NELSON MANDELA
UNIVERSITY

Modular Omni-directional AGV Developmental Platform with Integrated Suspension, Power-plant and Control Systems

Alternatively:

Novel Powered Castor Arrangement for use on Omni-directional Holonomic Automatic Guided Vehicles; Compliant with SIL and Industry 4.0

THESIS

Submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy (Mechatronics)
in the Faculty of **Engineering, Built Environment & Information Technology (EBEIT)**
at the **Nelson Mandela University**
by

Alexander B. S. Macfarlane

Supervisor: Prof. Theo van Niekerk	[Nelson Mandela University]
Co-supervisor: Udo Becker	[Ostfalia University]

December 2022

Alexander Blair Stuart Macfarlane
1 University Way
Summerstrand
Port Elizabeth
-SOUTH AFRICA-

I, Alexander Blair Stuart Macfarlane, hereby declare that this thesis for the degree Doctor of Philosophy in Engineering is my own work and that it has not previously been submitted for assessment or completion of any postgraduate qualification to another University or for another qualification.

November 4, 2022

Alexander B.S. Macfarlane

Copyright Statement

NELSON MANDELA
UNIVERSITY

PERMISSION TO SUBMIT FINAL COPIES OF TREATISE/DISSERTATION/THESIS TO THE EXAMINATION OFFICE

Please type or complete in black ink

FACULTY: Engineering, the Built Environment and Technology

SCHOOL/DEPARTMENT: Mechatronics

I, (surname and initials of supervisor) Theo van Niekerk

and (surname and initials of co-supervisor) Udo Becker

the supervisor and co-supervisor respectively for (surname and initials of candidate) A. B. S. Macfarlane

(student number) 211117994 a candidate for the (full description of qualification)

Doctor of Philosophy in Engineering (Mechatronics) [CODE 76002]

with a treatise/dissertation/thesis entitled (full title of treatise/dissertation/thesis):

Modular Omni-directional AGV Developmental Platform with Integrated Suspension, Power-plant and Control Systems

It is hereby certified that the proposed amendments to the treatise/dissertation/thesis have been effected and that **permission is granted to the candidate to submit** the final copies of his/her treatise/dissertation/thesis to the examination office.

Theo van Niekerk

18 July 2022

SUPERVISOR **DATE**

And

Prof. Dr.-Ing. U. Becker

14-07-2022

CO-SUPERVISOR **DATE**

Abstract

The thesis focuses on the development of an industrial automatic guided vehicle (AGV) with omni-directional capabilities. The omni-directional strategy used was the "swerve drive" system, a system whereby a wheel can be rotated about both its y axis (rolling axis) and z axis (vertical axis). Unlike most commonly used swerve drive systems that have swerve capabilities on each wheel attached to the body of the vehicle, this research seeks to reduce cost by only having swerve capabilities on two diagonal wheels. The remaining two wheels will act as castor units. AC drives are used on the system in place of more traditional DC drives, due to their cost vs capability advantage over DC and their prevalence in the industrial environment. Since an AGV is a mobile platform any power source found on it is usually derived from batteries, a DC source. Usage of DC introduces several limitations including difficulty transforming voltage levels for different systems, inability to run AC drives directly from the power source and comparably larger conduction wires. These limitations were overcome by adding a stand-alone power-plant on the AGV in the form of an inverter. The inverter transformed the DC power supplied by a battery bank from 48 volts DC to 230 volts AC. Thus, the primary focus of this research is on the development and validation of a novel two wheel omni-directional drive system that makes use of inexpensive and readily available components that have already been proven to work in industry.

Acknowledgments

There have been many people whom have contributed to this thesis, be it time, effort or advice whom I would like to thank.

Prof Theo van Niekerk, my research supervisor. I would like to thank you for your advice and mentorship during the thesis, without you this project would not have been possible. Your ability to secure funding and industry support was invaluable.

Prof Udo Becker, my co-superior. I would like to thank you for your input on my early mechanical design proposals and being the bridge between my work and Ostfalia University. Unfortunately I did not get to meet with you as often as I would have liked due to the Covid pandemic.

Karl du Preez, the big boss at AMTC; thank you for the resources, funding and space that AMTC provided for this thesis.

Dale Flynn, Amish Lalla, Sherwin Casling, Jacobus Van Der Mescht and Jaromir Cizek, the workshop crew, for assistance with the manufacture or welding of complex components, that were beyond my capabilities to manufacture.

Joshna Daya and Saadiqah Pandie, for assistance on the administrative side, finances and procurement of parts.

Izak du Preez, account manager at Siemens. For sourcing the best priced Siemens parts for academic use.

Keenan Coverly, application's engineer at Festo. For sourcing the best priced Festo parts for academic use.

My Family, thank you for your patience.

The institutions that have contributed to this thesis:

CSIR, financial contribution through the ROSSA initiative

NMU, the university that enabled this research

Contents

List of Tables

List of Figures

Nomenclature

Symbols

Symbol	***Unit***	***Meaning***
$\#$	$[unitless]$	Number of ...
α	$[{}^{rad}/{}_{s^2}]$	Angular Acceleration
ε	$[V]$	Electromotive Force (EMF)
ζ	$[unitless]$	Dampening effect
θ	$[rad]$	Angle in Radians
θ	$[^\circ]$	Angle in Degrees
μ	$[unitless]$	Coefficient of Rolling Friction
ρ	$[{}^{kg}/{}_{m^3}]$	Density
τ	$[Nm]$	Mechanical Torque
Φ	$[Wb]$	Magnetic Flux (Webers)
ω	$[{}^{rad}/{}_{s}]$	Angular Velocity
ω_n	$[{}^{rad}/{}_{s}]$	Natural Frequency
A	$[m^2]$	Area, Normally Cross-sectional
a	$[{}^{m}/{}_{s^2}]$	Acceleration
B	$[{}^{N\cdot s}/{}_{m}]$	Dampening Co-efficient
B	$[T]$	Magnetic Field Strength (Teslas)
E_E	$[{}^{Wh}/{}_{h}]$	Electrical Energy
C	$[Ah]$	Electrical Capacity
F	$[N]$	Force
f	$[Hz]$	Frequency (Hertz)
g	$[{}^{m}/{}_{s}]$	Gravitational Constant ($9.81{}^{m}/{}_{s}$)
I	$[A]$	Electrical Current
k	$[{}^{N}/{}_{m}]$	Spring Constant

K	$[J]$	Mechanical Kinetic Energy
L	$[H]$	Inductance (Henry)
m	$[kg]$	Mass
N	$[N]$	Normal Force
P	$[W]$	Power
RPM	$[{}^{r}/_{min}]$	Angular Velocity in Revolutions per Minute
r	$[mm]$	Radius
t	$[s]$	Time (Seconds)
t_h	$[h]$	Time (Hours)
t_m	$[min]$	Time(Minutes)
U	$[J]$	Mechanical Potential Energy
v	$[{}^{m}/_{s}]$	Velocity
V	$[V]$	Electrical Voltage
W	$[J]$	Mechanical Work
z	$[m]$	Vertical Displacement

Explanation of abbreviations

Shortcut	***Meaning***
AC	Alternating Current
AGV	Automated Guided Vehicle
AGC	Automated Guided Carts
AI	Analogue Input
AMTC	Advanced Mechatronic Technology Centre
AP	Access Point (Network)
AQ	Analogue Output
ASTPM	Association of Steel Tube & Pipe Manufacturers of South Africa
BLDC	Brushless DC Motor
CAD	Computer Aided Design
CO_2	Carbon Dioxide
CNC	Computer Numerical Control
CSIR	Council for Scientific and Industrial Research
DC	Direct Current
DDoS	Distributed Denial-of-Service
DHCP	Dynamic Host Configuration Protocol
DI	Digital Input
DoD	Depth of Discharge
DPDT	Double Pole Double Throw Relay
DPST	Double Pole Single Throw Relay
DQ	Digital Output
EU	European Union
EV	Electric Vehicle
FNB	First National Battery
F-DI	Fail-safe (Safety) Digital Input
F-DQ	Fail-safe (Safety) Digital Output
FOSS	Free and Open Source
GUI	Graphical User Interface

IC	Internal Combustion
IP	Internet Protocol Address
IoT	Internet of Things
IPC	Industrial Personal Computer
MAC	Media Access Control
MIG	Metal Inert Gas (Welding)
MMA	Manual Metal Arc (Welding)
NTP	Network Time Protocol
NAT	Network Address Translation
NC	Numerical Control
NUT	Network UPS Tools
OS	Operation System
NIC	Network Interface Card
PC	Computer (Personal Computer)
PCD	Pitch Circle Diameter
PL	Performance Level
PM	Permanent Magnet
PMDC	Permanent Magnet DC
PSU	Power Supply Unit
PTO	Power Take-Off
PWM	Pulse Width Modulation
REE	Rare Earth Element
REDOX	Oxidation-Reduction Reaction
ROS	Robot Operating System
RT	Real-time Communication
SIL	Safety Integrated Level
SPWM	Sinewave Pulse Width Modulation
SR	Switched Reluctance
SoC	State of Charge
SSH	Secure Shell
STO	Safe Torque Off
TIG	Tungsten Inert Gas (Welding)
UPS	Uninterrupted Power Supply
VM	Virtual Machine
WINE	WINE is Not an Emulator

1 Introduction

The introduction contains academic goals, technical objectives, funding sources and a list of publications related to this thesis.

1.1 Research Objectives

Primary research goal:

"Creation of the First Industry Compilation Two Wheel Swerve Drive AGV with a corresponding algorithm capable of controlling it"

Auxiliary research goals:

- Creating a PLC based AGV capable of utilising ROS that fully conforms to SIL standards
- Integrating a suspension system into a swerve drive system

Four-wheeled swerve drive AGVs are common in the academic space and have seen limited usage in industry. A swerve drive mechanism allows omnidirectional holonomic motions by rotating standard wheels on an axis perpendicular to the ground plane [1]. The limitation of the adoption of swerve-drive AGVs in industry is due to their relatively high cost when compared to the other omnidirectional strategies currently available (for information on other systems, see section 2.3). The cost can theoretically be optimised by reducing the number of driven units of a traditional swerve drive AGV from four to two, thus reducing the total number of motors needed from eight to four. The issue with reducing the number of driven wheels in a swerve drive system to less than four is that the system becomes inherently unstable. The primary research goal for this thesis is therefore to design the mechanical system and algorithm necessary to prove the viability of a two-wheeled swerve drive system. The secondary research goal is to ensure that the system is designed and coded in such a way as to adhere to SIL safety standards, making the AGV usable in the "real world". Finally, it is necessary to implement a suspension system on the swerve drive units due to the unique conditions found in South Africa. Research on adding suspension systems to swerve drive systems is not readily available.

1.2 Technical Objectives

It is the objective of this research to produce an electric, modular, scalable omnidirectional AGC type AGV that uses a two swerve drive unit system (with the other two wheels being unpowered casters). This research requires that a kinematic model of this AGV be created, with an appropriate control system, to allow for efficient control of the AGV when performing omnidirectional manoeuvres.

1.2.1 Specifications

The specifications of the AGV are listed in the itemised list:

1. The gross mass of the AGV (including payload) will be between 500 kg and 1 000 kg
2. The AGV will have 4 wheels (two powered and two unpowered)
3. The powered units will have their own drive train and steering mechanism
4. A basic suspension system is to be implemented
5. The wheel/drive train combos must be modular and removable from the AGV body
6. The maximum speed of the AGV should be 1.3 $^{m}/_{s}$
7. The AGV should be able to climb an incline of 5°

1.2.2 Related Publications

Publications related to this research are listed in table 1.1:

Table 1.1: List of Related Publications

Publication/Conference	**Year**
Robotics and Mechatronics Conference (RobMech)	2016
Robotics and Mechatronics Conference (RobMech)	2017
ICCMAT'19	2019
SAUPEC 2020	2020

1.3 Delimitation

The following aspects will be beyond the scope of the project described in the thesis:

- Design and implementation of the navigational system
- Design of the battery management system and automated battery changing systems
- Design of any expansion modules to increase the capability of the AGV beyond that of a AGC

1.4 Funding and Resources

The author would like to thank the following entities for funding this research:

- **CSIR**: (Council for Scientific and Industrial Research) for their financial contribution to the research over a three year period via the ROSSA initiative
- **The Rupert's fund**: which was awarded for the purchase or research textbooks, journals and other texts
- **AMTC**: (Advanced Mechatronic Technology Centre), a research entity at the Nelson Mandela University (NMU) which the author was a member of for the duration of this research; which provided tools, equipment, space and additional funding.
- **NMU**: (Nelson Mandela University, formally Nelson Mandela Metropolitan University) for facilitating this degree

1.5 Completed AGV Pictures

As the name implies, this section is dedicated to photographs of the real world AGV. The photos in figure 1.1, figure 1.2 and figure 1.3 show the completed AGV. Note in these photos that the acrylic side panels and the SCADA screen on the sensor stalk have been removed.

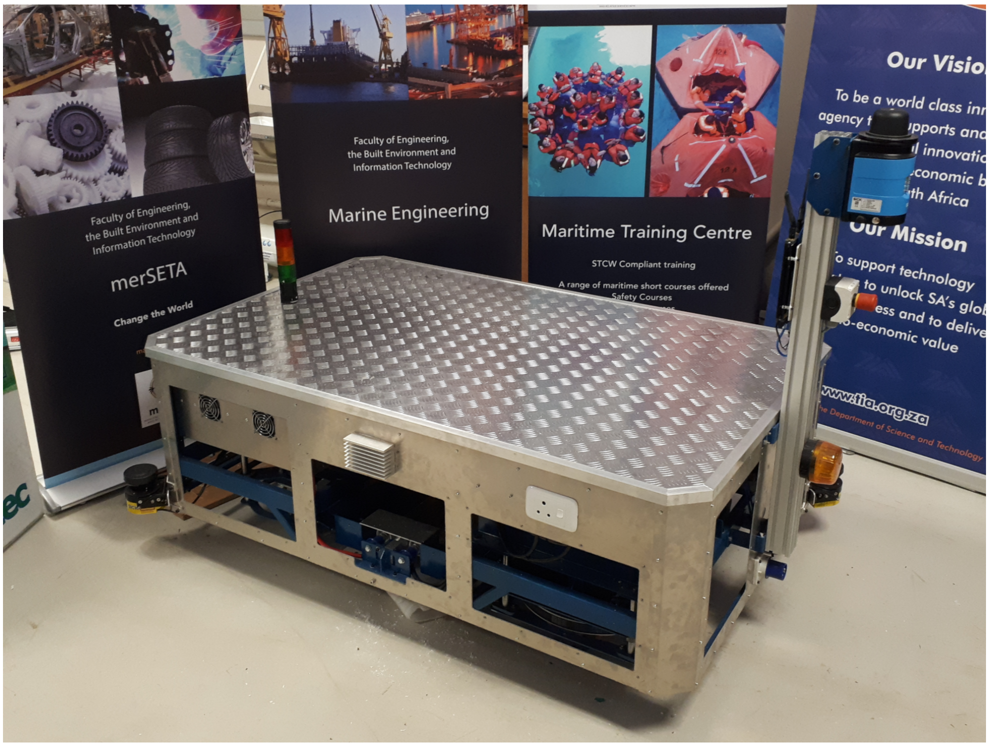

Faculty of Engineering, the Built Environment and Information Technology
merSETA
Change the World
Faculty of Engineering, the Built Environment and Information Technology
Marine Engineering
Maritime Training Centre
STCW Compliant training
A range of maritime short courses offered
Safety Courses
Our Visio
Our Mission
www.tia.org.za
The Department of Science and Technology

Figure 1.1: AGV Photo: Side View

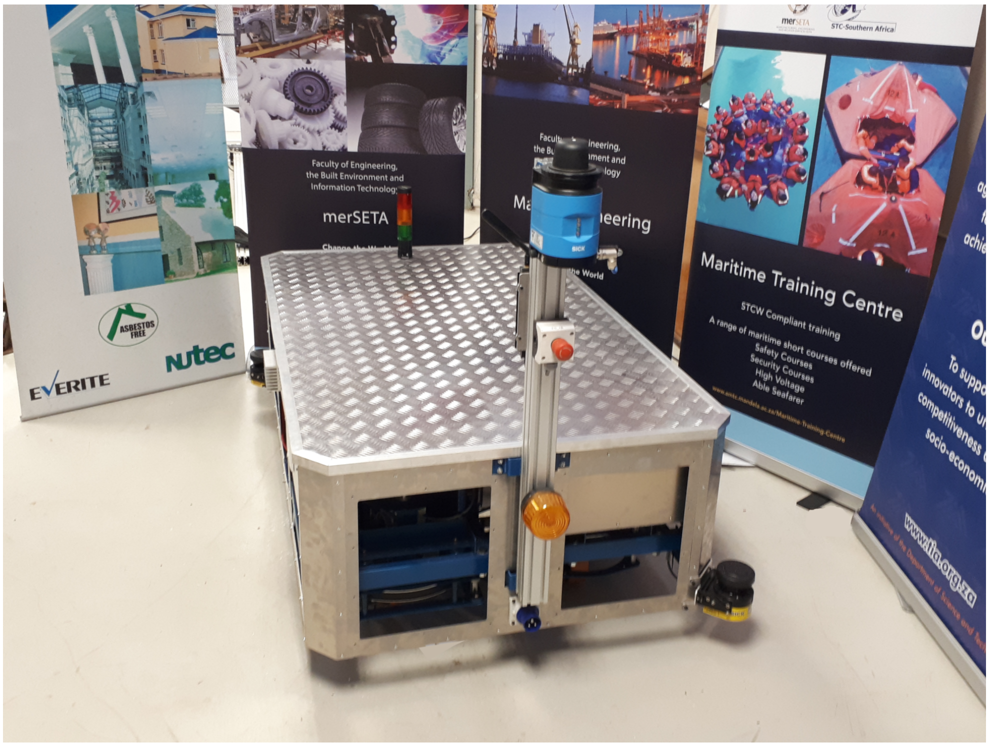


Figure 1.2: AGV Photo: Front View

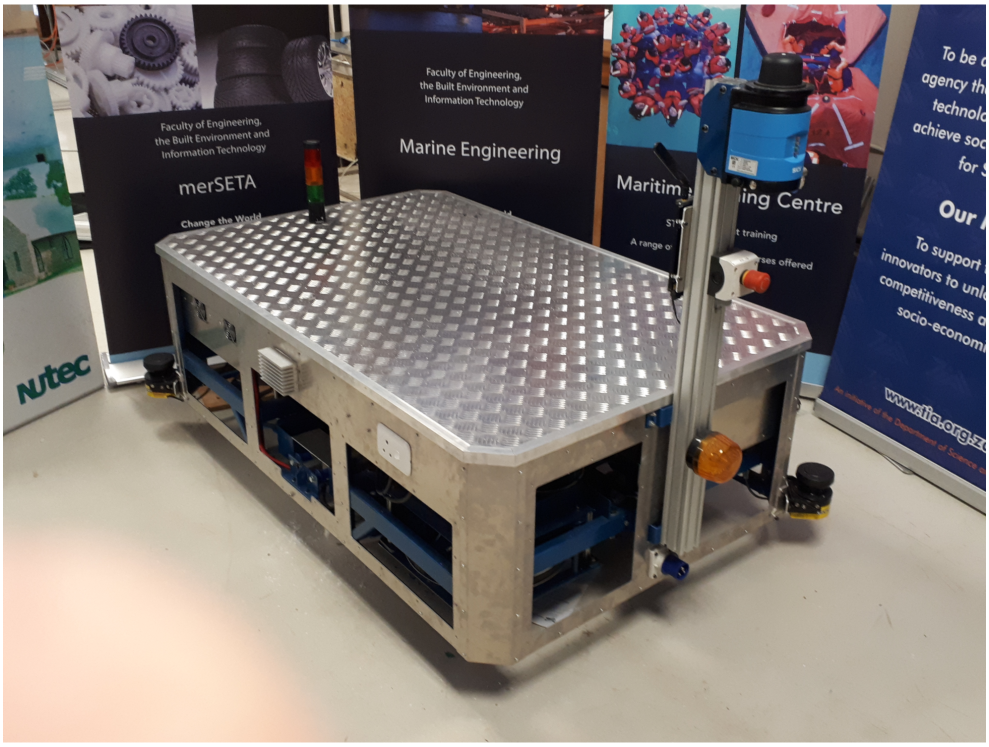

Faculty of Engineering, the Built Environment and Information Technology
merSETA
Change the World
Marine Engineering
nutec
www.tia.org.za

Figure 1.3: AGV Photo: Isometric View

Figure 1.4 shows the AGV during maintenance with the lid removed. The SCADA screen has been reattached to the sensor stalk, and the acrylic side panels are in place in this photo.

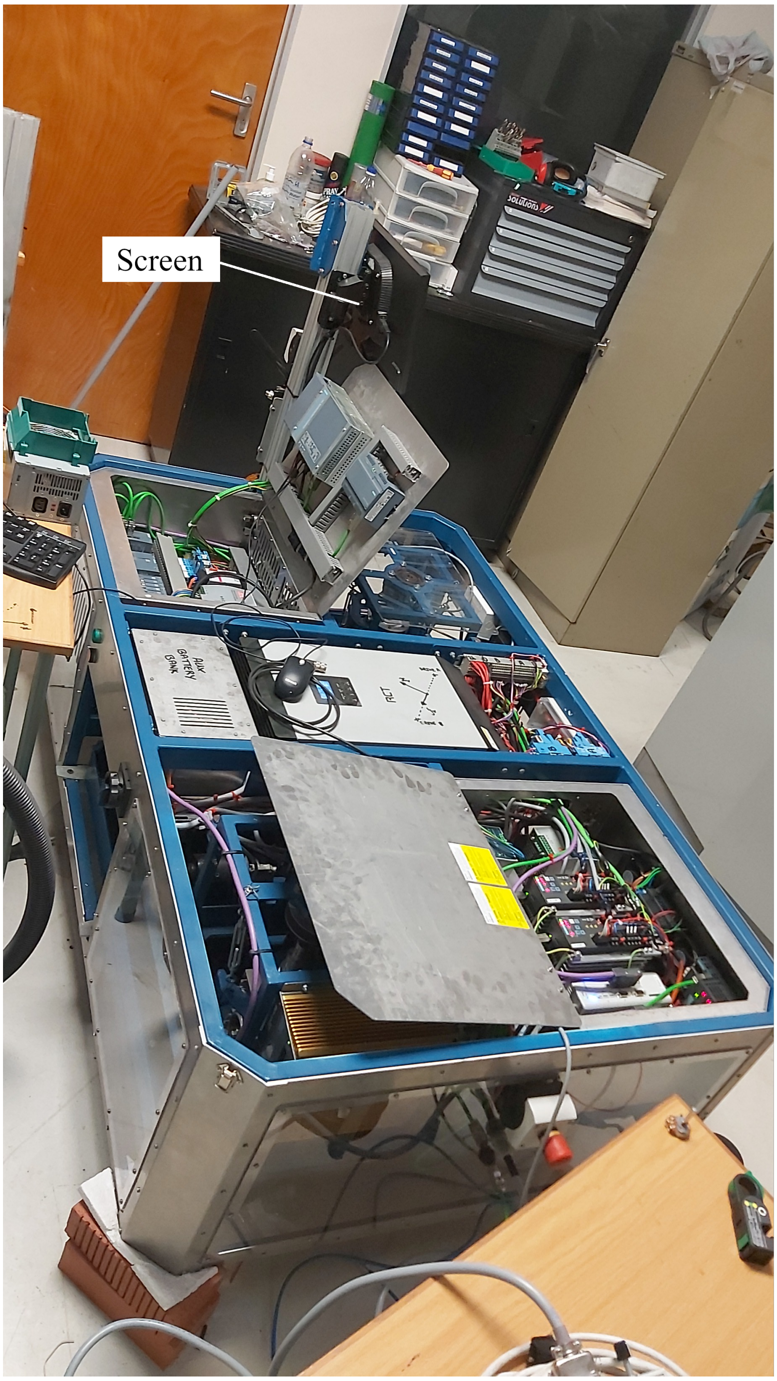


Figure 1.4: AGV Photo: Opened Up

1.6 Layout of Chapters

The chapters of the thesis are set out as follows:

1. Literature Survey:
   A critical examination of the current (circa. 2020) AGV market, justification for the use of AGVs and identification of the knowledge gap that exits in regard to swerve drive AGVs.

2. Theoretical Basis:
   Theoretical knowledge that will be needed to comprehend the research project. Not relevant to those whom are well versed in the field.

3. AGV Design Tool:
   A chapter about a MATLAB based tool the author created to aid with the development and implementation of the swerve-drive based AGV.

4. Mechanical Design of the AGV :
   A complete record of the vehicle's mechanical design, including a record of the design methodology and decisions. This chapter spans from mechanical conceptual design through computer aid design to manufacturing of the AGV.

5. Electrical Design of the AGV:
   A record of both the power distribution systems and control systems. Listing power requirements, electrical layouts, networking, I/O assignment, etc.

6. Safety Evaluation:
   An evaluation of the AGV's level of safety per the SIL safety standards.

7. Kinematics:
   A description of how the kinematic model of the AGV was developed, including the completed model.

8. Programming and Future ROS Integration:
   A description of the code architecture for the AGV. Including the PLC code, IPC code, SCADA code and drive configurations.

9. Testing
   Results of testing the AGV results include both calibration tests and validation tests.

10. Conclusion
    Conclusions are drawn on the behaviour of the AGV along with possible future

improvements.

1.7 Chapter Conclusion

The research aim is stated along with the rationale and motivation behind the original proposal. The desired outcomes of the research are specified in the form of an objectives list. This chapter includes a list of delimitations for the research and a layout of chapters for the thesis.

2 Literature Review

The literature survey will systematically justify the pursuit of the research project.

2.1 Background

An Automatic Guided Vehicle (AGV) is an intelligent self-driving mobile device that has become a popular means of material handling and distribution in automated factories (Industry 4.0 Compliant) [2]. Although there is no standard definition of AGV categories, manufacturers of these devices often group them into the following categories [1]:

- Automated Guided Carts (AGC's)

  These vehicles have no method of loading or offloading payloads; they are either loaded manually or by an external device [4]. The AGC is comparable to a conveyor belt.

- Mobile Robots

  Mobile robots are essentially autonomous platforms onto which an industrial robot is mounted; this extends the robot's work area beyond the limit of its arm's reach [5].

- Autonomous Forklifts

  The "ISO Standard 6780: Flat pallets for intercontinental materials handling" is a ubiquitous standard used for material handling. The manipulation of these pallets is done mainly through the use of operator driven forklifts. However, as of 2009, many of these forklifts have been replaced with autonomous variants [6].

- Tugger AGVs

[1] List compiled from Savant Automation [3]

Tugger AGVs are similar to AGCs; however, unlike AGCs, tugger AGVs tow their payload rather than carry it [7].

- Unit Load AGVs

  Unit load AGVs are AGCs with an onboard method for loading and offloading parts [8].

2.2 Motivation for AGV Research in South Africa

AGVs have begun to replace traditional fixed production lines, consisting of conveyor belts and large fixtures (called monuments). Replacement is due to the high cost, size, inflexibility and lack of scalability of these production lines compared to AGV based production lines [9]. It is often far more cost-effective and flexible to place all tooling and fixtures on mobile robotic systems such as AGVs [10].

The intelligence of the AGV can vary from the simple line following up to swarm logic, but regardless of the level of intelligence of the given device, they all have one thing in common, they can work independently of human guidance [11]. Almost all AGVs require a connection to a more extensive network to operate efficiently since the AGV needs to "know" what is happening in the factory as a whole (i.e. where parts need to be collected, what inventory slots are open, etc.) [12]. The high level of interconnectivity is dubbed "The Internet of Things (IoT)". The "Internet of Things" or "IoT" forms a cornerstone of "Industry 4.0". Industry 4.0 originated in Germany as a new ideological practice aimed at industry and focuses on the following elements [13]:

- Value Creation of Products and Quality of Life Improvement of Workers
- Networking and Transfer of Intelligent Systems
- Improved Pace of Innovation in Industry
- Creation of an Innovation Friendly Framework
- Transparency and Participation of Industrial Stakeholders

Since Germany is pushing for Industry 4.0 and the AGVs that go hand in hand with it, it will not be long before the automotive industry in South Africa follows suit. South Africa is following Germany's lead primarily due to two of the largest auto manufactures in South Africa being German. Namely, Volkswagen and Mercedes [14], whom are aligning their goals with the industry 4.0 narrative. Industry 4.0 has also been at the forefront of South Africa's current (circa 2019 - 2022) economic strategy; thus, becoming Industry 4.0 compliant will require locally produced AGVs to do so [15]. The global trend toward IoT necessitates the uptake of AGVs in the manufacturing sector [16].

The AGV researched in this thesis is intended for usage in the South African market; therefore, it should be viewed through such optics. It was estimated by Danielle Le Roux that up to 35% of South African jobs would be displaced by automation [17]. This replacement includes cashiers, tellers, construction workers, mine workers, factory workers and maintenance workers [18]. In South Africa, a large majority of automation is linked to the automotive sector or "auto industry", whose GDP contribution to South Africa has increased, on average, by 1.26% annually (0.8% in 2018, 1.2% in 2019 and 1.8% in 2020). This profitability increase occurred alongside a gradual decrease in the employment numbers by 2.6% on average (between the years 2018 to 2020) [19]. This increase can be directly attributed to the increased proliferation of automation in the automotive sector. Whether this is socially a positive or a negative, it is an inevitability. Thus, research into AGVs (especially in South Africa) is more relevant than ever.

AGVs also have a place in older factories, replacing workers performing repetitive, mundane tasks. This replacement improves efficiency of the factory as a whole since errors and safety lapses that are inherent with humans performing repetitive tasks are eliminated [20]. Other benefits that AGVs have over manual labour include lower labour costs, increased productivity and keeping humans away from potentially dangerous activities [21].

2.3 Motivation for the use of Omnidirectional AGVs

AGVs that have omnidirectional capabilities and thus increased flexibility are of great interest to the manufacturing industry, especially when compared to their non-holonomic counterparts [22]. An "omnidirectional" vehicle can move in any direction instantaneously from its current location and orientation in 2D space [23], in many regards similar to how a human being would negotiate their surroundings. Omnidirectional capabilities reduce the footprint required for vehicle manoeuvres, especially the large turning radii characteristic of Ackerman steering or differential steering [24]. This footprint reduction reduces the size of the manufacturing space overall, reducing monthly maintenance costs and capital costs associated with expansion. Omnidirectional AGVs can better function in complex working environments and older factories.

These older factories were never intended to host mobile robots or AGVs and were instead designed around human kinesiology, and ergonomics [25].

There exist two strategies to achieve omnidirectional motion in a vehicle [26]. The first strategy is to use a custom wheel capable of holonomic motion. Currently (circa 2022), three standard holonomic wheels exist, namely the mecanum wheel [27], the omniwheel [28] and the spherical wheel [29]. The second strategy is to use a standard wheel and rotate it about its z-axis; this is often referred to as a swerve drive system [30].

2.3.1 Mecanum Wheels

Mecanum wheels were first developed in Sweden in 1975 [31]. A mecanum wheeled system does not require a traditional steering system but instead can steer thanks to the unique characteristics of these wheels. A mecanum wheel consists of a hub that carries freely rotating rollers that are angled at 45° to the hub's rotational axis [32], traction is generated at 90° to the rollers and thus 45° to the AGV's body. A pair of left and right-handed mecanum wheels are illustrated in figure 2.1.

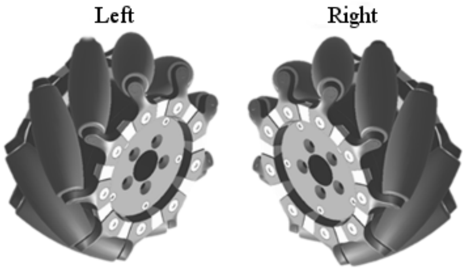


Figure 2.1: Left & Right Mecanum Wheels

Since the traction generated by a mecanum wheel is at 45° to the body of the AGV, the driving force of the mecanum wheel (resultant force) is also at 45° to the AGV's body. An illustration of this effect can be seen in figure 2.2.

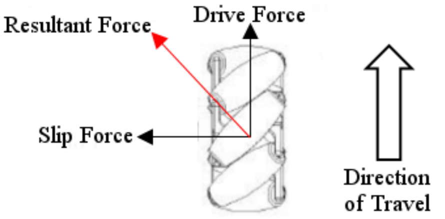


Figure 2.2: Forces on a Left Hand Mecanum Wheel

When left and right-handed mecanum wheels are combined, as illustrated in figure 2.3, the slip forces of all the wheels cancel (when the wheels are all rotating the same way and at the same speed); thus, the AGV can drive in a forward in a straight line. Omnidirectional motion is achieved by disturbing this balance by changing the wheels' speed or rotation direction relative to each other. In figure 2.3, the traction force of each wheel is represented by a red arrow, while black arrows represent the components of these forces.

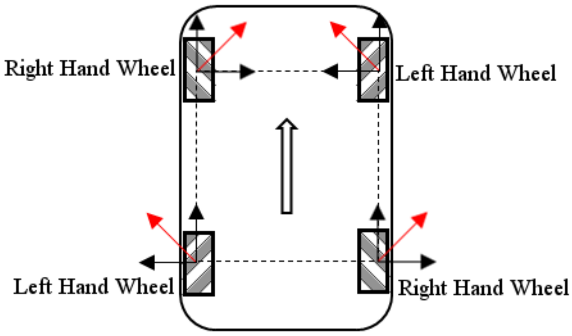


Figure 2.3: Typical 4 Mecanum Wheel Layout

Mecanum wheels are inherently unstable; this instability results from the balancing act that the system must continuously perform to operate. If at any time the contact profile of one of the AGVs wheels changes due to suspension influences or traction problems, the AGV will begin to move in an erroneous motion as the balancing act has been disturbed [33]. This erroneous motion is complicated to compensate for [31] since the wheel slippage cannot be detected using shaft encoders. Thus some form of high-level absolute positioning system must be used. An attempt to use the SICK NAV 350 laser scanner for this purpose was made by G. Scott [8] in 2015. The sensor proved ineffective for this task as both the resolution and update frequency proved insufficient to correct for the mecanum wheels' slippage fast enough. This slow update cycle resulted in the AGV vibrating as it moved under slippage conditions. A second significant drawback is that one component force of the wheel is always cancelled. Therefore, this cancelled force does not contribute to the overall motion of the AGV. These cancelled forces require energy to produce, energy which is essentially used by the AGV to work against itself or is lost through the rollers [34]. Excessive energy expenditure is not ideal since this is a battery-powered mobile platform.

2.3.2 Holonomic Wheels

Like the mecanum drive system, the holonomic drive system requires no steering mechanism. Steering is done by unbalancing the outputs of the wheels attached to the device. A holonomic wheel consists of a hub around which rollers are attached. The rollers have a rotational axis 90° to the hubs axis of rotation, see figure 2.4.

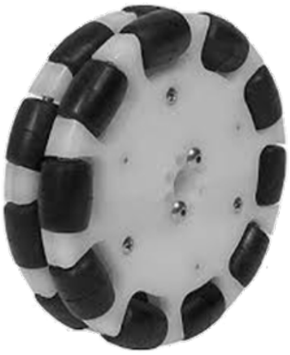


Figure 2.4: Holonomic Wheel[2]

Holonomic wheels are usually attached at 45° angles to the AGV's body, as illustrated in figure 2.5. The wheels provide traction in a direction normal to their rotational axis (like conventional wheels), while the rollers attached around the hub of the wheel allow "slip" in the direction normal to the desired direction of travel [36]. In figure 2.5, the red arrow represents the traction forces each wheel produces, while the black arrows represent the components of these forces.

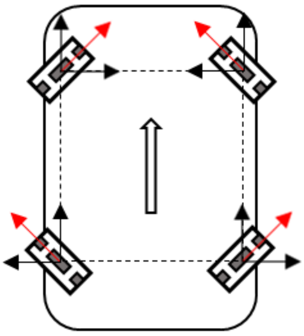


Figure 2.5: Holonomic Wheel Layout

As can be seen in figure 2.5, for the holonomic drive to achieve forward motion, the lateral components of the wheel's traction forces must cancel. Similarly to the mecanum

[2] Image adapted from Pitsco [35].

wheel drive, all the wheels must maintain the same traction profile and angular velocity for these forces to cancel. This system is often rejected for industrial usage as, in addition to having all the drawbacks of the mecanum wheel, these wheels are notoriously fragile [37]. This fragility makes mecanum wheels a better choice when compared.

2.3.3 Spherical Wheels

These wheels are the most costly to produce compared to their holonomic and mecanum cousins. Spherical drive systems consist of three or more spherical wheels; each of these spheres is actuated by two traditional wheels that make tangential contact with the spheres surface [38]. This system requires a minimum of 6 motors to operate as two motors are needed for each of the three spheres. This system is illustrated in figure 2.6.

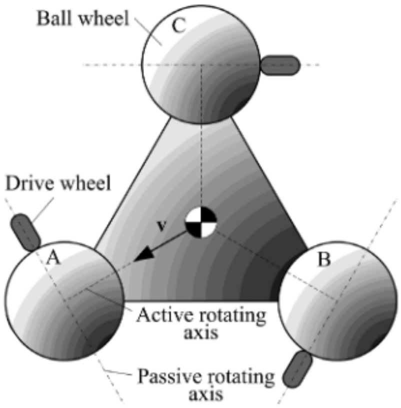


Figure 2.6: Spherical Wheel System[3]

This system has the disadvantage of being expensive to machine, is relatively fragile and is temperamental on rough ground [34]; hence why no commercial AGV manufacturer has ever attempted to use this system.

2.3.4 Traditional Swerve Drive Systems

Swerve drive systems are also known as powered castors or castor drives. Swerve steering is an omnidirectional strategy based on four powered castor wheels. A powered castor wheel has powered rotation about the y axis (rolling rotation of the wheel) and z-axis (steering axis perpendicular to the ground plane). The powered rotation about the z-axis of the castor wheels allows for omnidirectional capabilities. Two layouts are

[3] Image adapted from Diegel et al. [34].

commonly used for powered castors in swerve-drive systems. The first is an in-line system where the z-axis and y-axis share a common origin point. The second is the offset system, where the z-axis and y-axis are still perpendicular to each other, but their origin points are separated by a distance called the castor offset. The in-line system is illustrated in figure 2.7a , while the offset system is illustrated in figure 2.7b.

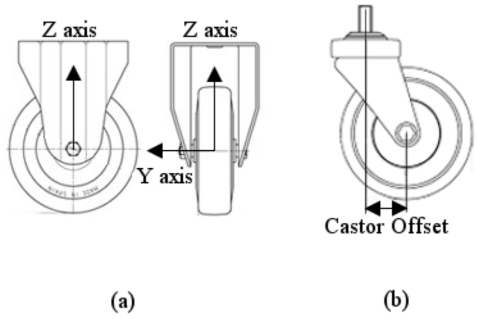


Figure 2.7: Power Castor Wheel Axes

The "castor offset" ensures that the wheels do not scuff when the wheel is rotated about its steering axis (z-axis) as this offset forces a steering arc. Without the offset, compensation must be performed by rotating the wheel about its rolling axis (y-axis) while steering occurs to form a turning arc.

For a four-wheel swerve drive system to achieve complete omnidirectional motion, the z-axis of the four wheels must be placed on the corners of a virtual square. If this is not done, rotation about the centroid of the vehicle will be impossible without scuffing the wheels. See figure 2.8 for an illustration of this effect; note the orientations of the wheels, which form a tangent to a virtual circle whose centre point coincides with the centroid of the AGV's body. The direction of motion of each wheel in figure 2.8 is illustrated with a black arrow. The overall motion of the AGV's body is also indicated.

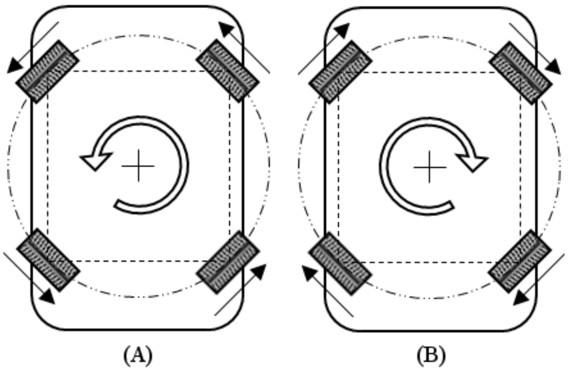


Figure 2.8: Swerve Drive Rotational Operation

The remaining omnidirectional motions of the AGV can be achieved by orienting the wheels of the AGV as shown in figure 2.9. All of the motions swerve drive is capable of are summarised in table 2.1

Table 2.1: Swerve Drive Omnidirectional Motions Summary

Figure	**Figure Letter**	**Description**
2.8	A	Anti-clockwise rotation about the centroid
2.8	B	Clockwise rotation about the centroid
2.9	C	Forward / Reverse motion
2.9	D	Left / Right motion
2.9	E	Forward diagonal motion
2.9	F	Reverse diagonal motion

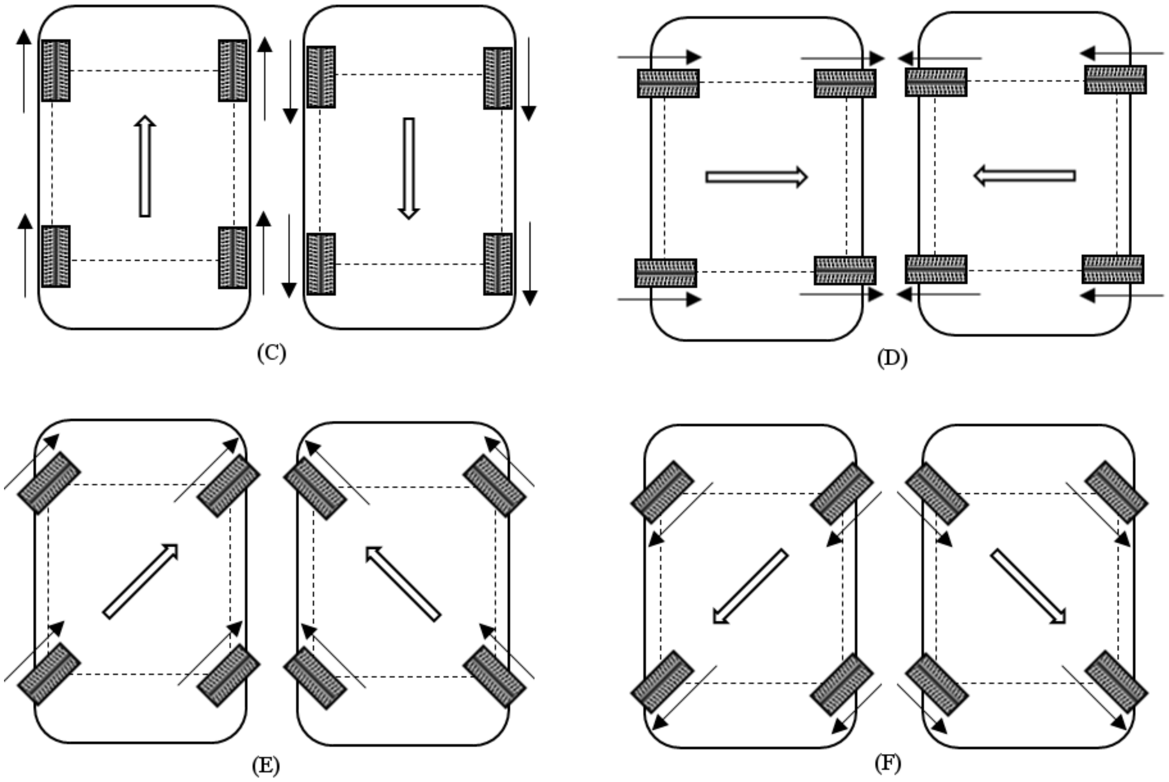


Figure 2.9: Swerve Drive Omnidirectional Operation

The original patent for a swerve drive system is titled "Powered Castor Wheel Module for use on omnidirectional drive systems" [39]. It was first filed in 1998. The original system developed called for four in-line powered castor units to be placed on a pitch circle diameter (PCD) of an arbitrary radius. See figure 2.10.

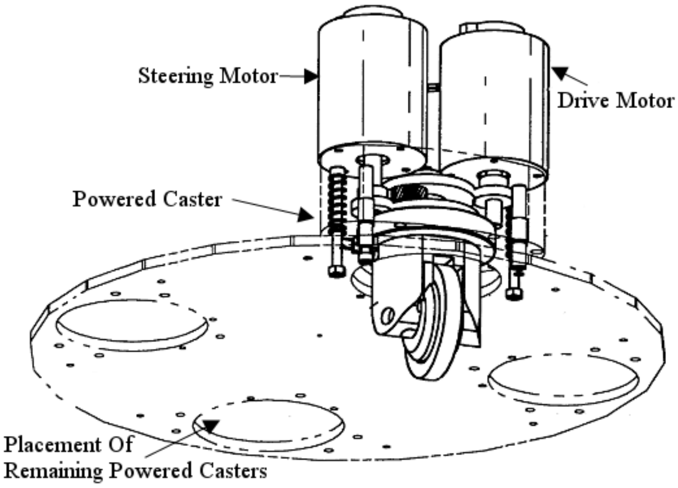


Figure 2.10: Holmberg & Slater's Powered Castor Concept[4]

One of the primary concerns when using a swerve drive system based on Holmberg et al.'s [39] work is the sheer number of motors that the system would require. Since each unit needs steering and drive motors. The number of motors for a four-wheeled system will be eight individual motors. One approach commonly used to reduce the number of motors is to couple all the drives and steering motors together. This idea was explored by Wada et al. [40], their design is illustrated in figure 2.11.

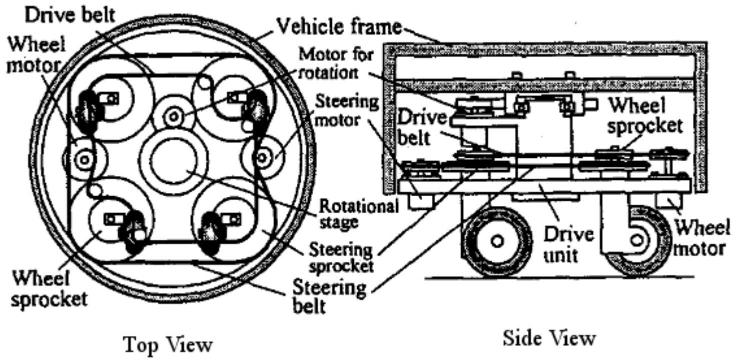


Figure 2.11: Wada, Takagi & Mori's Synchronous Powered Castor Concept[5]

[4] Image adapted from Holmberg et al. [39].

[5] Image adapted from Wada et al. [40].

Wada et al. [40] coupled all four wheels to a single drive motor and a single steering motor using a belting system. One crucial feature that must be noted with this system is that the steering sprockets of one diagonal wheel pair must be wound opposite to the remaining diagonal pair to ensure the steering works correctly.

This design effectively reduced the number of motors from eight to two. There are, however, two significant drawbacks to this system. The first is when the steering is coupled, as shown in figure 2.11, the AGV can no longer rotate about its centroid (as done in figure 2.8A and 2.8B). The system is thus no longer truly omnidirectional or holonomic. The second drawback is that with all of the drive motors coupled, Ackerman turning the AGV will cause the wheels to scuff without any form of a differential gear. The need for a differential, when Ackerman turning, is described in figure 2.12

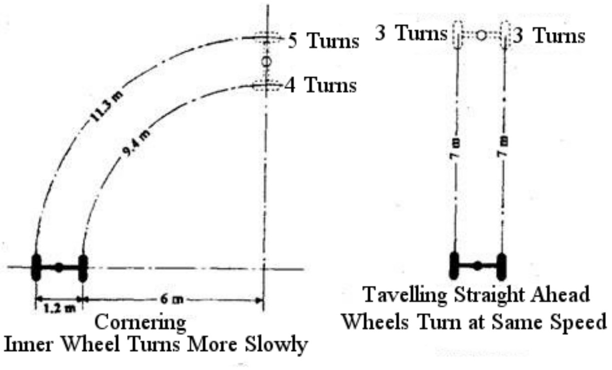


Figure 2.12: Differential Gear Justification[6]

The swerve drive system does, however, have the advantage of using traditional wheels rather than specialised holonomic wheels. The use of traditional wheels is significant as the wearing component of any AGV is the wheel, and as such cheaper wheels will ensure a cheaper running cost, even if the upfront cost is greater with this system.

2.4 Justifying Research: Two Wheel Swerve Drive System

The research aims to produce an omnidirectional AGV for use in industry; thus, this system must take industrial demands into account. The omniwheel and spherical wheel system can therefore be immediately discarded for industrial use due to their relative fragility [37] [38]. Discarding these systems would leave only mecanum wheels and swerve drive systems as viable options.

[6] Image adpated from Patel [41].

Since the mecanum wheel is the wearing part of the mecanum wheel system, maintenance will be costly. Swerve drive systems do not suffer from this issue as they use conventional wheels that are cost-effective to replace. Mecanum wheels have difficulty with stability as the change in traction of a single wheel can affect the motion of the whole AGV [33]. This issue is compounded by the fact that wheel encoders cannot be easily implemented to determine slippage and compensate appropriately due to the nature of the mecanum wheel [31]. Swerve drive systems can directly compensate for slippage with an encoder and torque feedback as they use traditional wheels [42]. The final issue with mecanum wheels, when compared to the traditional wheels of the swerve drive system, is the vibration that all mecanum wheels produce as the load is transferred from one peripheral roller to the next [34].

With all of the mentioned disadvantages of mecanum wheels, it would appear that the traditional swerve drive system should be completely dominant in the manufacturing industry. Swerve drive system are, however, not dominant. For all their benefits, swerve drive systems are less common than mecanum systems in industry(when a holonomic omnidirectional vehicle is required). This lack of dominance is solely due to the initial cost of a traditional swerve drive system compared to a mecanum system. The high initial cost of swerve-drive AGVs is directly attributable to the eight motors that are needed to produce a functional four-wheeled system [37] when compared to the four required for an equivalent mecanum wheeled system.

This research proposes alleviating the high initial cost of the traditional swerve drive system by removing two diagonal swerve units and replacing them with unconstrained castors. This removal will reduce the number of motors from eight to four, in line with a typical mecanum wheel system.

2.5 Justifying Research: Conforming to SIL Standards

Any machine that operates in an industrial environment must, by law, conform to appropriate legally binding safety standards. In South Africa, this is the SANS standard, which is based on a combination of the international standards ISO and IEC [43]. Most academic research fails to take this into account and often produces machines and prototypes that are marvels of research but need to be completely redesigned from the ground up when commercialised. This redesign is necessary as preliminary design choices made in a "safety vacuum" would not allow safety laws to be adhered to retroactively.

The EU Machinery directive 2006/42/EC ratifies the SIL standard [43], which is synergised with both ISO and IEC standards. This research intends to strictly adhere to the SIL standard from initialisation, making this, most likely (as far as the author's research has found), the very first academic swerve-drive AGV to have an intrinsic safety rating. The methodology to achieve this, documented in this research thesis, will allow future authors to make their AGV's safety compliant and intrinsically safe easily.

2.6 Justifying Research: Intergeneration of a Suspension System in Swerve Drive

Suspension systems on factory AGVs are a rarity since most AGVs of this type are expected to work on a floor with flat ground [44]. This expectation is a common doctrine, especially in the more industrialised nations of Europe and Mainland China. This doctrine, however, cannot be extended to the South African environment. Many buildings that house factories in South Africa have floors that no longer conform to SANS 10400 [45] [33]. This unevenness results from adding and removing machines over the years (leaving behind remnants of anchor points) or general wear and tear. Given the harsher conditions an AGV would likely experience in South Africa, it is advantageous to incorporate a cost-effective suspension system. This implementation has yet to be done on a swerve drive AGV.

3 Theoretical Basis

This section explains the theoretical knowledge needed to comprehend the research contained in this thesis.

3.1 Forward and Reverse Kinematics

This section discusses the kinematics of the AGV.

3.1.1 Basic of Terrestrial Kinematics

To determine the orientation of an object, in this case, an AGV, in 3D space, a minimum of six axes is needed to describe position fully. This description is called the six-tuple space and is referred to as a coordinate frame system. This thesis will use the Cartesian frame whose representations are listed in table 3.1.

Table 3.1: Six-Tuple Axis in Cartesian Space

Representation	**Description**
x	First axis of the 3D frame
y	Second axis of the 3D frame
z	Third axis of the 3D frame
α	Rotation about the x fame axis (pitch)
β	Rotation about the y fame axis (roll)
γ	Rotation about the z fame axis (yaw)

In order to move an AGV (or any system capable of motion) to a desired position and orientation, it is necessary to actuate the "joints" of the system in a specific manner. These "joints" are not necessarily physical; in the case of an AGV, they include the wheel's interaction with the driven surface. When the resultant position of the AGV's body is determined using known joint positions, this is referred to as forward kinematics. When the desired position of the AGV's body is known while the necessary position of the joints is to be determined, this is known as reverse kinematics.

3.1.2 Frames and Coordinate References

If the AGV were to move between two points in 3D space (see figure 3.1), the coordinate system of the AGV would change as shown in the schematic diagram 3.1. The AGV will move from the coordinate system "Frame A" to the coordinate system "Frame B" as it travels from position A to

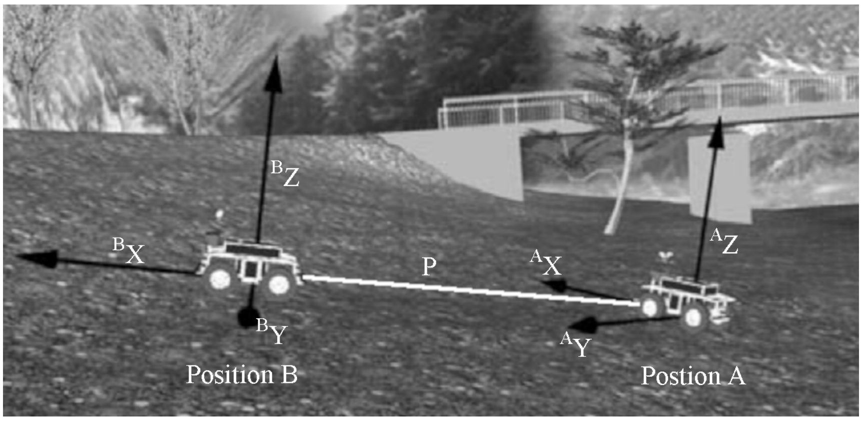


Figure 3.1: Pictorial Representation of an AGV Moving in 3D Space[1]

If a fixed coordinate system frame is defined, it is possible to define the position of the AGV in 3D space relative to this frame. To describe the position of the AGV, at position B relative to A, the translation vector $^{A}\vec{P}$[2]is used. Vector $^{A}\vec{P}$ in the coordinate system $\{A\}$ and consists of the elements given in equation 3.3.

$$^{A}\vec{P} = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T = p_x \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + p_y \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + p_z \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{3.1}$$

For an AGV, it is insufficient only to describe the vehicle's position in 3D space; it is also necessary to describe the orientation. Fully describing the position is done by creating the second coordinate frame $\{B\}$. If the orientation of the AGV in position

[1] Image adapted from Ullrich et al. [11].

[2] The preceding super-script in $^{A}\vec{P}$, demonstrates that this vector is with reference to coordinate frame system $\{A\}$

A is taken as the fixed coordinate system orientation for frame $\{A\}$, the orientation of the AGV in Position B, given by frame system $\{B\}$, can be described as the unit vectors $\hat{X}_B$, $\hat{Y}_B$ and $\hat{Z}_B$. These unit vectors will need to be expressed in the coordinate system frame $\{A\}$ by projecting them from frame $\{B\}$ to frame $\{A\}$. This projection is made using dot products as shown in equation 3.2 for the x-axis.

$$^{A}\hat{X}_B = \begin{bmatrix} \hat{X}_B \cdot \hat{X}_A \\ \hat{X}_B \cdot \hat{Y}_A \\ \hat{X}_B \cdot \hat{Z}_A \end{bmatrix} \tag{3.2}$$

Where $\hat{X}_B$ is projected onto $\hat{X}_A$, $\hat{Y}_A$ and $\hat{Z}_A$. Similar operations can be done for $\hat{Y}_B$ and $\hat{Z}_B$. Since dot products are used and since unit vectors by definition have a length of $|1|$:

$$\boldsymbol{A} \cdot \boldsymbol{B} = ||\boldsymbol{A}||||\boldsymbol{B}|| \cos\alpha = \cos\alpha \tag{3.3}$$

If this is done for all three vectors that span the coordinate system frame $\{B\}$ then the rotational matrix given in equation 3.4 will be derived. This matrix describes the coordinate frame system $\{B\}$ relative to $\{A\}$.

$${}^{A}_{B}\boldsymbol{R} = \begin{bmatrix} {}^{A}\hat{X}_B & {}^{A}\hat{Y}_B & {}^{A}\hat{Z}_B \end{bmatrix} \tag{3.4}$$

Since ${}^{A}_{B}\boldsymbol{R}$ consists of three unit vectors, the rotational matrix is orthonormal; this implies that the inverse matrix of ${}^{A}_{B}\boldsymbol{R}$ is the transpose matrix or ${}^{A}_{B}\boldsymbol{R}^T$.

With a method to describe the displacement of the AGV from position A to position B in frame $\{A\}$ using equation 3.3 and a method to describe the rotation of $\{B\}$ relative to $\{A\}$ using equation 3.4. It is possible to create a full transform from $\{A\}$ to $\{B\}$ using equation 3.5.

$$\{B\} = \{{}^{A}_{B}\boldsymbol{R},^{A} \vec{P}\} \tag{3.5}$$

3.1.3 Mapping Frames Onto Each Other

If frames $\{A\}$ and $\{B\}$ had the same orientation and the position of frame $\{B\}$ with reference to frame $\{A\}$ was given as a translation ${}^{A}\vec{P}$ apart; then it is possible to reference the point ${}^{B}\vec{Q}$ in frame $\{A\}$ as shown in equation 3.6:

$$^{A}\vec{Q} =^{B} \vec{Q} +^{A} \vec{P} \tag{3.6}$$

Since it is only possible to add vectors that share the same coordinate reference frame when the frames differ in orientation, it is necessary to rotate one frame to match the orientation of the other. This can be done using the rotational frame developed in equation 3.4 to generate equation 3.7:

$$^{A}\vec{Q} =^{A}_{B} \boldsymbol{R}^{B}\vec{Q} +^{A} \vec{P} \tag{3.7}$$

To simplify notation the translation and rotation operations can be combined to form a transformation matrix, such that equation 3.7 becomes equation 3.8 and equation 3.9:

$$^{A}\vec{Q} =^{A}_{B} \boldsymbol{T}^{B}\vec{Q} \tag{3.8}$$

Where:

$$\left[^{A}Q\right] = \left[\begin{array}{ccc|c} & ^{A}_{B}\boldsymbol{R} & & ^{A}P \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \left[\begin{array}{c} ^{B}Q \\ \hline 1 \end{array}\right] \tag{3.9}$$

$^{A}_{B}\boldsymbol{T}$ is know as a homogeneous transform, it can be inverse'd as shown in equation 3.10

$$\left[\begin{array}{ccc|c} & ^{A}_{B}\boldsymbol{R} & & ^{A}P \\ \hline 0 & 0 & 0 & 1 \end{array}\right]^{-1} = \left[\begin{array}{ccc|c} & ^{A}_{B}\boldsymbol{R}^{T} & & -^{A}_{B}\boldsymbol{R}^{T} \cdot^{A} P \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3.10}$$

In equation 3.8 to equation 3.10 a method to map coordinates from one frame to another was developed. The kinematic rules illustrated here are not only useful for demonstrating how the AGV can move from one frame to another in space via the wheel-floor "joint"[3]. They can also describe how individual components of the AGV must move to ensure the AGV's macro motion of the wheel-floor joint can be achieved as desired. That is to say, how the steering and velocity of the wheels will affect the entire system's motion.

Thus often, more than two frames will have to be mapped onto each other. This mapping can be done by stringing, in order, the transform matrices from the last

[3] "joint" refers to a point between two structures of fixed geometry that allow these structures to move relative to each other

frame of the system to the origin frame. For example, a three-frame system (say an arm with two "links"[4]and three "joints") will have frame $\{A\}$, $\{B\}$ and $\{C\}$. To get the position and orientation of any point in frame $\{C\}$ with reference to point $\{A\}$, the transform matrix from $\{C\}$ relative to frame $\{A\}$ is generated as illustrated in equation 3.11:

$$\begin{aligned} {}^{A}\vec{P} &= {}^{A}_{B}\boldsymbol{T} \cdot {}^{B}_{C}\boldsymbol{T} \cdot {}^{C}\vec{P} \\ &= {}^{A}_{C}\boldsymbol{T} \cdot {}^{C}\vec{P} \end{aligned} \tag{3.11}$$

Where the transformation matrix ${}^{A}_{C}\boldsymbol{T}$ is:

$${}^{A}_{C}\boldsymbol{T} = \left[\begin{array}{ccc|c} \multicolumn{3}{c|}{{}^{A}_{B}\boldsymbol{R} \cdot {}^{B}_{C}\boldsymbol{R}} & {}^{A}_{B}\boldsymbol{R} \cdot {}^{B}\vec{P}_{C} + {}^{A}\vec{P}_{B} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3.12}$$

$$\begin{aligned} {}^{A}\vec{P}_{B} &= \text{Translation from } \{A\} \text{ to } \{B\} \\ {}^{B}\vec{P}_{C} &= \text{Translation from } \{B\} \text{ to } \{C\} \end{aligned}$$

3.1.4 Simplifying Kinematics Using Roll-Pitch-Yaw

In the previous section, the rotation of the AGV from A to B is described using a 3x3 matrix. The column vectors represent the orthogonal unit vectors for the coordinate frame system in this matrix. Using a 3x3 matrix means that nine different values are used to describe the rotation. However, these nine values are not necessary as it is possible to use only three values to represent the orientation of the frame. This reduction is possible since the matrix columns are orthogonal, and their dot product is zero as they are unit vectors with a magnitude of 1. Reducing the matrix will impose six constraints on the orientation of the frame, and as such only three remain. Thus, it is sufficient to represent the orientation of the AGV in frame B as a rotation about specific angles around the x, y and z axis of frame A, see figure 3.2.

[4] "links" refer to any structure between joints that will retain a fixed geometry

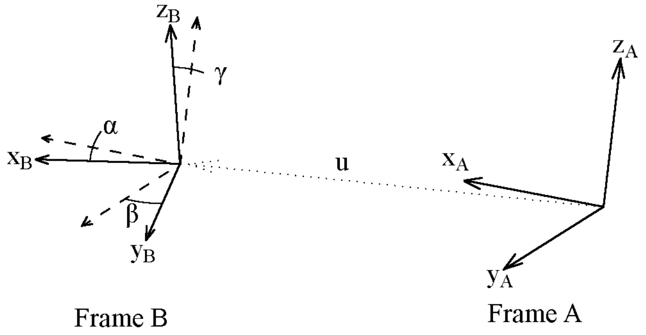


Figure 3.2: Transform Free-Body Diagram of AGV from Frame A to Frame B

Hence, to describe the linked rotations from position A to position B, the Roll-Pitch-Yaw model is used in this thesis. Where the AGV first rotated about the z-axis, then the y axis and finally the x-axis, with each rotation using a variable rotation axis. The matrix to describe this rotation is given in equation 3.13:

$$\boldsymbol{R} = \boldsymbol{R}_z(\alpha) \cdot \boldsymbol{R}_y(\beta) \cdot \boldsymbol{R}_x(\gamma) \tag{3.13}$$

Where:

$$\begin{aligned} \boldsymbol{R}_z(\alpha) &= \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ \boldsymbol{R}_y(\beta) &= \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \\ \boldsymbol{R}_x(\gamma) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\gamma) & -\sin(\gamma) \\ 0 & \sin(\gamma) & \cos(\gamma) \end{bmatrix} \end{aligned} \tag{3.14}$$

When the matrices in equation 3.14 are multiplied as defined in equation 3.14, the rotational matrix is described as:

$$\boldsymbol{R} = \begin{bmatrix} \cos(\alpha)\cos(\beta) & \cos(\alpha)\sin(\beta)\sin(\gamma) - \sin(\alpha)\cos(\gamma) & \cos(\alpha)\sin(\beta)\cos(\gamma) + \sin(\alpha)\sin(\gamma) \\ \sin(\alpha)\cos(\beta) & \sin(\alpha)\sin(\beta)\sin(\gamma) + \cos(\alpha)\cos(\gamma) & \sin(\alpha)\sin(\beta)\cos(\gamma) - \cos(\alpha)\sin(\gamma) \\ -\sin(\beta) & \cos(\beta)\sin(\gamma) & \cos(\beta)\cos(\gamma) \end{bmatrix} \tag{3.15}$$

Since the rotation matrix in equation 3.15 described the state of the AGV from A to B, it can be said that the transform matrix is of B relative to A or ${}^{A}_{B}\boldsymbol{R}$. To complete the transform from orientation or 'frame' A to orientation or 'frame' B the translation matrix $\vec{u}$ (movement vector, see figure 3.1) must be included. This is done in equation 3.16:

$$\boldsymbol{B} =^{A}_{B} \boldsymbol{R} \cdot \vec{u} \tag{3.16}$$

3.2 General Vehicle Modelling

Since the AGV described in this thesis will use traditional wheels and not exotic wheels such as mecanum wheels or holonomic wheels, traditional modelling equations can be used for calculations related to tractive effort. In order to design an effective AGV drive system, all forces acting on the machine need to be considered. These forces are:

- The rolling resistance of the wheels on the driving surface
- The effects of the vehicles weight on a slope due to gravity
- The force needed to accelerate the vehicle
- The aerodynamic drag of the vehicle due to the atmosphere

Rolling Resistance with Constant Velocity

The AGV's rolling resistance is related to the force of friction between the tyres/wheels of the AGV and the surface on which the AGV is driving. This force is typically modelled as constant and is independent of the vehicle's weight. The rolling resistance is calculated using equation 3.17 from Serway [46]. It is important to note that this force is always in the opposite direction to the object's direction of motion.

$$F_{rr} = \mu_{rr} N \tag{3.17}$$

F_{rr}	$=$	Force of rolling friction	N
μ_{rr}	$=$	Rolling co-efficient of friction	
N	$=$	Normal force	N

Since the normal force for an object on a flat surface is equal to the gravitational force, this is illustrated in figure 3.3, equation 3.17 can be expanded to equation 3.18:

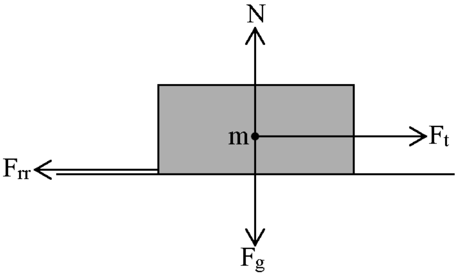


Figure 3.3: Force of Friction for an Object on a Flat Plane

$$F_{rr} = \mu_{rr} F_g = \mu_{rr} m g \tag{3.18}$$

F_t	$=$	Applied force of traction	N
F_g	$=$	Gravitational force	N
m	$=$	Mass	kg
g	$=$	Gravitational constant (9.8 $^{\mathrm{m}}/_{\mathrm{s}}$)	$^{m}/_{s}$

The only force action acting against the force of traction, "F_t", of the vehicle is "F_{rr}", thus:

$$\begin{aligned} F_t &= F_{rr} \\ &= \mu_{rr} m g \end{aligned} \tag{3.19}$$

Rolling Resistance with Acceleration

For the AGV to accelerate on a flat surface, extra tractive force must be applied beyond that needed to overcome rolling friction. This extra force required is defined by "Newtons' First Law" and is given in equation 3.20.

$$\begin{aligned} F_a &= ma \\ &= m\left(\frac{v_f - v_i}{t}\right) \end{aligned} \tag{3.20}$$

F_a	=	Force required for acceleration	N
a	=	Acceleration	$^{m}/_{s^2}$
v_f	=	Final velocity of AGV	$^{m}/_{s}$
v_i	=	Initial velocity of AGV	$^{m}/_{s}$
t	=	Time to reach final velocity from initial velocity	s

Using the frictional force calculated in equation 3.18, the applied traction force needed to accelerate the vehicle between two speeds in a given time frame will be:

$$\begin{aligned} F_t &= F_a + F_{rr} \\ &= m\left[\left(\frac{v_f - v_i}{t}\right) + \mu_{rr} g\right] \end{aligned} \tag{3.21}$$

Inclined Surface with Constant Velocity

When a vehicle is moving up an incline, gravity comes into play; this means a higher traction force will be necessary to overcome the force of friction and the force of gravity combined. In order to calculate the needed traction force "F_t" the force diagram in figure 3.4 is used.

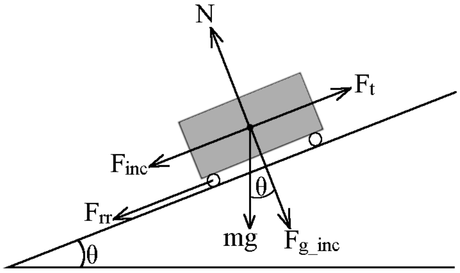


Figure 3.4: Force Diagram for a Vehicle on an Incline at Constant Velocity

Using figure 3.4, it is possible to deduce that the force of rolling friction will change form equation 3.18 to equation 3.22.

$$\begin{aligned} F_{rr} &= \mu_{rr} N \\ &= \mu_{rr} F_{g\_inc} \\ &= \mu_{rr} mg\cos(\theta) \end{aligned} \tag{3.22}$$

F_{g_inc} = Component force of F_g co-linear to the normal force N

θ = Angle of incline

The force of gravity acting on the vehicle body is represented by "F_{inc}" and is calculated using equation 3.23.

$$F_{inc} = mg\sin(\theta) \tag{3.23}$$

Thus the applied traction force "F_t" will be:

$$\begin{aligned} F_t &= F_{inc} + F_{rr} \\ &= mg\left[\sin(\theta) + \mu_{rr}\cos(\theta)\right] \end{aligned} \tag{3.24}$$

Acceleration on an Incline

The equation that describes the traction force that will be required to accelerate a vehicle up an incline can be generated by adding an acceleration component (see equation 3.20) to equation 3.24. This is done in equation 3.25

$$\begin{aligned} F_t &= F_{inc} + F_{rr} + F_a \\ &= m\left(g\left[\sin(\theta) + \mu_{rr}\cos(\theta)\right] + \left[\frac{v_f - v_i}{t}\right]\right) \end{aligned} \tag{3.25}$$

Notes on Aerodynamic Drag

For this research project, the effects of aerodynamic drag will be negligible. It is negligible because fluid drag is proportional to velocity, and as the AGV described in this thesis will have its top speed clamped to 1.3 $^{m}/_{s}$, for "safe" operation in a factory environment, the force created by fluid drag will be an order of magnitude less than the other tractive forces acting on the AGV. Excluding aerodynamic drag is justified using the estimation given in equation 3.26:

$$F_D = C_D A \frac{\rho v}{2} \tag{3.26}$$

F_D	=	Atmospheric drag force	N
C_D	=	Co-efficient of AGV's drag	
A	=	Frontal area of the AGV	m^2
ρ	=	Density of the fluid (air)	$^{kg}/_{m^3}$
v	=	Velocity of the fluid (air)	$^{m}/_{s^2}$

The value of C_D is estimated to be 0.76, which was taken from table 15-2 in Cengel [47]. The frontal area of the AGV is estimated at $0.4m^2$. The density of air at $25°C$ and 1 atmosphere of pressure is $1.184^{kg}/_{m^3}$ (taken from table A-22 in Cengel [47]. Since the top speed of the AGV is $1.3^{m}/_{s^2}$ the "worst-case" force of atmospheric drag on the AGV will be:

$$F_D = 0.76 \cdot 0.4 \left(\frac{1.184 \cdot 1.3}{2}\right) = 0.23N \tag{3.27}$$

Since this is less than a single Newton on a system whose forces will be measured in the hundreds of Newtons (see later chapters), this force is negligible.

3.3 Chapter Conclusion

This chapter established the mathematical basis needed to comprehend this thesis.

4 AGV Design Tool

The AGV design tool is a software tool written in Matlab to aid with the mechanical calculations needed to design the AGV. Most of the values and results found in chapter 5, were determined iteratively using this tool.

4.1 Introduction to the Design Tool

Traditional design processes are sequential in nature as illustrated in figure 4.1. Where a project typically starts with identifying a problem, then follows a step by step process to end at a documented solution [48].

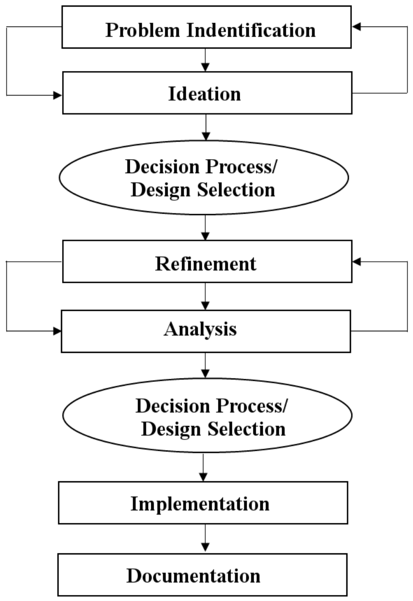


Figure 4.1: Traditional Iterative Design Process

As illustrated in figure 4.1, there is a cyclic sequence of steps around refinement and analysis of possible solutions that could solve the identified problem. This back and forth between refinement and analysis of a possible solution is highly costly to the project in both time and effort as each time this cycle occurs, the majority of the design calculations have to be redone as the idea is tweaked. In order to reduce the number of iterative calculations that were needed, it was decided to automate the design process calculations as far as reasonable.

4.2 AGV Design Tool Program Environment

This automation would come in the form of a design tool built in the Matlab code environment. The tool would use both Matlab's "quick and dirty" coding language and Simulink, a simulation package available as part of the Matlab environment. Using Matlab ensures that the code can easily be changed or manipulated by engineers, as the software is aimed at engineers and not software developers, while Simulink gives access to tools that would be difficult to implement in code. Since the AGV in this thesis is a swerve drive AGV, the design tool will be focused on creating an AGV of this type.

4.3 Software Requirements

This section deals with the inputs the tool will require from the user and the outputs that the tool will produce.

4.3.1 Required User Inputs

Listed in the section that follows are the required user inputs needed to generate a usable result. Further information on the listed input parameters can be found in the following section.

1. Net AGV mass (Kg)
2. Maximum speed of the AGV ($^m/_s$)
3. Desired time to reach maximum speed (t)
4. Maximum inclination that the AGV should be able to climb ()
5. Wheel diameter (mm)
6. Rolling friction of wheels
7. Number of drive wheels

8. Number of wheels used during power conservation mode
9. Spring constant of suspension system ($^{N}/_{m}$)
10. Dampening co-efficient of suspension system ($^{N \cdot s}/_{m}$)

The net mass of the AGV refers to the combination of the AGV's unloaded weight (i.e. the weight of the AGV's body only) and the payload weight. Essentially it is the maximum weight the AGV is allowed to reach. The weight of the AGV is used in conjunction with the maximum speed and time taken to reach this speed to determine the force required to accelerate the AGV. The maximum allowable inclination is used to determine the force of gravity acting against the AGV, while the rolling friction co-efficient is used to determine the force of friction against the AGV. In conjunction with the wheels diameter and number of drive wheels, these forces are used to determine the total tractive power requirements for the AGV and the power and torque requirements for each drive motor.

The "conservation mode" was dropped from the research project but was never purged from the code. The idea behind the conservation mode was that when the AGV moves on a flat surface (no force of gravity acting against the tractive force), some of the drive motors could be shut down and the drive train disconnected so that these wheels were "freewheeling". This "freewheeling" could be theoretically done as the remaining motors would have generated sufficient power to drive the AGV without the gravitational force working against the vehicle.

The spring constant and the dampening co-efficient are used to determine the effectiveness of the suspension system by generating the theoretical response to a given obstacle. The response can then be iteratively improved by tweaking the spring constant and dampening co-efficient.

4.3.2 Generated Program Outputs

The outputs generated by design tool are itemised:

1. Net forces acting on the AGV's body for:
    - Constant velocity on s flat surface
    - Acceleration on a flat surface
    - Constant velocity on an inclined surface
    - Acceleration on an inclined surface

2. The resulting net torques for each of the above conditions
3. Maximum required torque, RPM and power of each drive wheel
4. Selection of an appropriate gearbox from a list of of all considered manufacturer's data sheets; given the characteristics of the chosen motor, gearbox efficiencies, starting conditions, loading conditions and duty cycle
5. Selection of a pair of belt drives to connect the wheel to the gearbox and the gearbox to the drive motor from a list of considered manufacturers; while calculating the geometrical constrains (max pulley size, centre distances, belt widths, belt-pulley contact arc, etc.)
6. Displacement, velocity and acceleration response graphs for the suspension system

4.4 Operational Process

The design tool will work in an iterative fashion, this operation is illustrated in the flow diagram in figure 4.2.

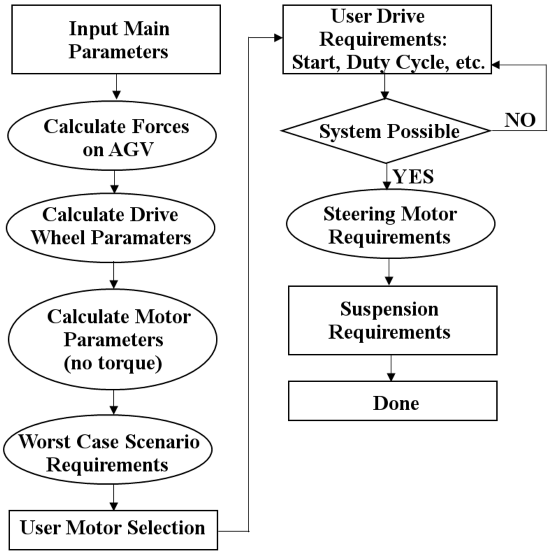

Input Main Parameters
Calculate Forces on AGV
Calculate Drive Wheel Paramaters
Calculate Motor Parameters (no torque)
Worst Case Scenario Requirements
User Motor Selection
User Drive Requirements: Start, Duty Cycle, etc.
System Possible
NO
YES
Steering Motor Requirements
Suspension Requirements
Done

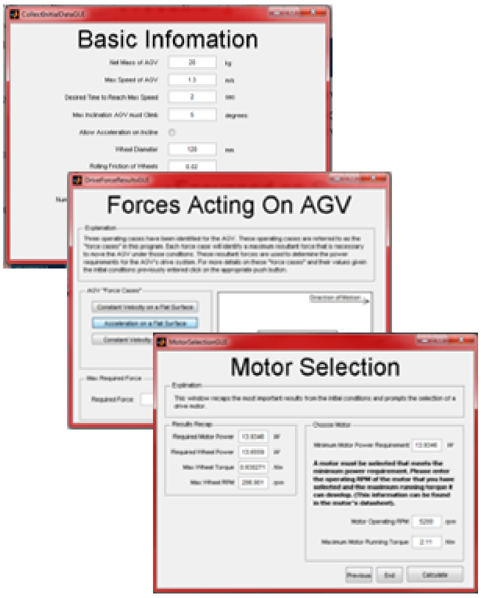
Figure 4.2: Process Diagram Describing Program Operation

4.5 Software Graphical User Interface

In an attempt to improve the ease of use of the design tool, a graphical user interface (GUI) was incorporated into the program. This GUI makes using the software far easier for non-computer literate personnel as a command-line interface can often be unwieldy for individuals not familiar with such an interface. The GUI also allows for the inclusion of informatics and diagrams into the program interface, making understanding the values required by the program easier to comprehend (a picture is worth a thousand words). There are 16 windows in total that make up the GUI, which are listed:

1. Start
2. Request Basic Information
3. Forces Acting on AGV
4. Drive Wheels Parameters
5. Drive Motor Parameters
6. Request Drive Motor Selection
7. Request AGV Operating Conditions (loading conditions, duty cycle, etc.)
8. Drive Train Development (after each part of the drive is defined, this GUI is reopened)
9. Belt Drive 2 Development
10. Bevel Gearbox Unit Selection
11. Belt Drive 1 Development
12. Request Steering Parameters
13. Steering System and Steering Motor Requirements
14. Steering Train Development GUI (after each part of the steering train is defined, this GUI is reopened)
15. Belt Drive Development GUI (Steering Train)
16. Gearbox Selection GUI (Steering Train)

A couple of screens are shown in figure 4.3 as examples. A complete compendium of all of the GUI can be found in appendix G.



CollectInitialDataGUI

Basic Infomation

Net Mass of AGV 25 kg

Max Speed of AGV 1.5 m/s

Desired Time to Reach Max Speed 2 sec

Max Inclination AGV must Climb 5 degrees

Allow Acceleration on Incline ○

Wheel Diameter 120 mm

Rolling Friction of Wheels 0.02

DriveForceResultsGUI

Forces Acting On AGV

Explanation

Three operating cases have been identified for the AGV. These operating cases are referred to as the "force cases" in this program. Each force case will identify a maximum resultant force that is necessary to move the AGV under those conditions. These resultant forces are used to determine the power requirements for the AGV's drive system. For more details on these "force cases" and their values given the initial conditions previously entered click on the appropriate push button.

MotorSelectionGUI

Motor Selection

Explanation

This window recaps the most important results from the initial conditions and prompts the selection of a drive motor.

Results Recap

Required Motor Power 13.8346 W

Required Wheel Power 13.6559 W

Max Wheel Torque 0.839271 Nm

Max Wheel RPM [illegible] rpm

Choose Motor

Minimum Motor Power Requirement 13.8346 W

A motor must be selected that meets the minimum power requirement. Please enter the operating RPM of the motor that you have selected and the maximum running torque it can develop. (This information can be found in the motor's datasheet).

Motor Operating RPM 5200 rpm

Maximum Motor Running Torque 2.11 Nm

Figure 4.3: Design Tool Example GUIs

4.6 Evaluating the Validity of The Software Tool

Before the software tool could be trusted for use to design an AGV, it first had to be validated; this was done using three validation tests:

- Test 1: Compare a set of calculated results to results the author generated by hand
- Test 2: Compared to existing in-house AGV, for this Chikosi's [49] AGV was used
- Test 3: Compared to an externally built AGV, for this the AGV built by Wada et al. [50], in the thesis "Holonomic and Omnidirectional Vehicle with Conventional Tires" was used.

A list of all the input data used for the three tests is given in table 4.1.

Table 4.1: Software Tool Testing Inputs

Input Data	**Test 1**	**Test 2**	**Test 3**
Net Mass	600 kg	300 kg	300 kg
Max Speed	1.3 ${}^{m}/_{s}$	1.5 ${}^{m}/_{s}$	1.3 ${}^{m}/_{s}$
Inclination Climb	2 s	3 s	2 s
Wheel Diameter	150 mm	482.6 mm	150 mm
Tyre Rolling Friction Co-efficient	0.02	0.25	0.02
Tyre Dynamic Friction Co-efficient	0.6	0.6	0.6
Number of Drive Wheels	2	4	4
Turning Speed	80 ${}^{r}/_{min}$	60 ${}^{r}/_{min}$	80 ${}^{r}/_{min}$
Castor Offset	45 mm	10 ${}^{r}/_{min}$	140 ${}^{r}/_{min}$

Once each test was run with the input values listed in table 4.1, a set of results was generated as listed in table 4.2.

Table 4.2: Software Tool Testing Outputs

Output Data	**Test 1**	**Test 2**	**Test 3**
Wheel Power Required	663.17 W	332.16 W	165.01 W
Wheel RPM	165.52 ${}^{r}/_{min}$	59.36 ${}^{r}/_{min}$	165.52 ${}^{r}/_{min}$
Max Wheel Torque	38.26 ${}^{n}/_{m^2}$	53.42 ${}^{n}/_{m^2}$	9.52 ${}^{n}/_{m^2}$
Drive Monitor Power	676.71 W	338.935 W	168.38 W
Turning Torque	36.69 Nm	4.41 Nm	61.74 Nm
Turning Power	309.63 W	27.71 W	517.23 W

4.7 Evaluation Validation Results of the Software Tool

The results delivered in table 4.2 can now be compared to real-world results to test the validity of this calculator. This comparison is made in the sections that follow.

4.7.1 Test 1 Results - 600 kg AGV

This test was done against hand calculations. The values chosen as inputs for this test match the preliminary values of the swerve drive AGV described in this thesis. This test shows that the calculator and hand calculations exactly match, but this is to be expected as the calculator is an automated version of the hand calculations with some optimisation algorithms to select the best fit from catalogue components. This result cannot be used alone to validate the design tool.

4.7.2 Test 2 Results - 300 kg AGV

Since Chikosi's [49] AGV was built internally at the same research group (AMTC at the Nelson Mandela University) as the AGV described in this report, a detailed specification sheet could be sourced for this machine. When Chikosi's [49] report was compared to the results generated in test 2, the following was noticed. The wheel torque of Chikosi's [49] AGV was specified in his report as 53.45 Nm driven by a 300 W motor. When the torque value is compared to the design programs' results, it is noticed that the required torque is almost an exact match. There is a discrepancy in the wattage of the motor; Chikosi's [49] motor is 11.49% smaller than that of the design program. Chikosi [49] later clarifies that the motor used was fractionally underpowered as the next available motor in the range he had chosen was a 500 W unit. The steering motor chosen by Chikosi [49] was rated at 30 W, while the design program determined that a power of 27.71 W was required. This difference is a discrepancy of 7.63%.

4.7.3 Test 3 Results - 250 kg AGV

A 250 kg swerve drive AGV was developed by Wada et al. [50] in the thesis titled "Castor Drive Mechanisms for Holonomic and Omnidirectional Mobile Platforms with no Over Constraint". The thesis outlined a couple of crucial details of the design, the most important being the size of the drive motors compared to the physical characteristics of the AGV. It was stated by Wada et al.[50] that two 300 W motors powered the AGV. This statement means that the motors on this system are 43.87% larger than are supposedly required according to the design software. This motor size could be due to the availability of motors or over-specification. However, this result could also suggest that the design software is inaccurate, though this is unlikely due to tests 1 and 2 performing so favourably. No detail was available about the sizing of the steering mechanism.

4.8 Evaluation Conclusion

It can be concluded that the results produced by the design software/ calculator are accurate enough to be used in the creation of the swerve drive AGV. This confirmation is a result of the following. Firstly, the hand calculations exactly match the results of the calculator. Secondly, in test 2, Chikosi's [49] AGV and the calculator's results closely matched, with the worst deviation being 11.49%. Finally, due to the lack of transparent design information provided by Wada et al. [50], test 3 cannot be reasonably used for confirmation or disapproval of validity.

4.9 Chapter Conclusion

A Matlab based tool was developed that allows for the rapid iterative design of swerve-drive AGVs; this tool was proven against existing systems for validation. With this tool, the iterative design process involving the selection of standardised parts was fast-tracked compared to traditional hand calculations.

5 Mechanical Design of the AGV

As alluded to in the previous chapter, the design of this AGV will be based upon the swerve drive architecture with two drive units and two freewheeling castor units. The drive units are responsible for generating the tractive effort of the AGV, while the castor units are solely to support the vehicle, capable of neither steering nor traction. Since swerve drives are used, the AGV is omnidirectional (for more information on omnidirectional systems, refer to chapter 2).

5.1 Vehicle Axis System

The roll-pitch-yaw vehicle axis system will be used throughout this thesis whenever describing the orientation of the AGV. This system is illustrated graphically in figure 5.1.

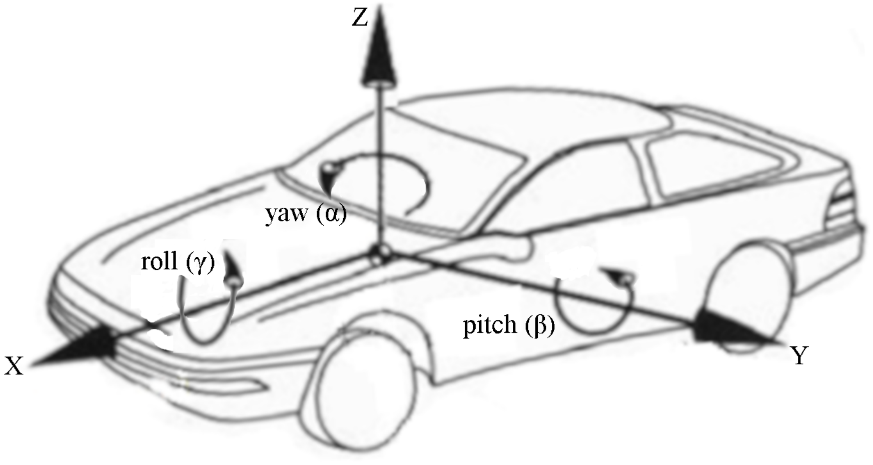


Figure 5.1: The Roll-Pitch-Yaw Vehicle Axis System[1]

[1] Image adapted from Gillespie [51]

5.2 Initial Layout & Component Placement

Before any CAD design was conducted, it was first necessary to generate an initial layout of the AGV. This initial layout allowed for the macro planning of component placement. The initial plan for the AGV is illustrated in figure 5.2.

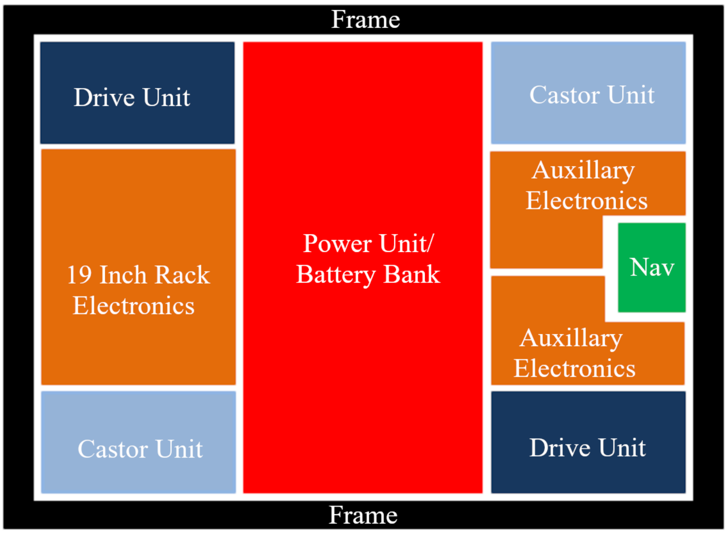


Figure 5.2: Initial Layout of the AGV

The layout shown in figure 5.2, was also the layout presented to ROSSA (the main capital contributor) during the preliminary stages of this research. However, the electrical component layout had to be changed due to size and component limitations to the layout illustrated in figure 5.5.

The original layout of the AGV shown in figure 5.2 made use of a "19 inch" rack form factor for electronics (see figure 5.3). The sliding action of the rack was to be orientated vertically (along the z-axis of the AGV). This concept was originally intended to add a modularity system by splitting the electronics into separate hot-swappable boxes. This system proved to be too inflexible to incorporate into the AGV, given the desired final dimensions of the vehicle.

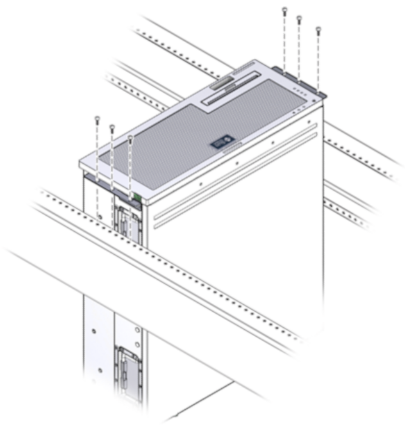


Figure 5.3: 19 Inch Rack Hot Swap Concept[2]

The primary idea was always to have the drive units diagonal to each other, as shown in figure 5.2. This orientation ensured that regardless of the AGV's direction of travel, the centroid of the AGV would experience no net torque. This lack of net torque is due to the moments generated by the drive units about the centroid, cancelling each other since they will always have equal magnitude but opposite directions. In practice, this is not always the case, as effects such as wheel slip will disrupt this balance, but it provides a stable starting point that allows easier correction of these issues in software. Which simplifies control of the AGV significantly. This concept is illustrated in figure 5.4:

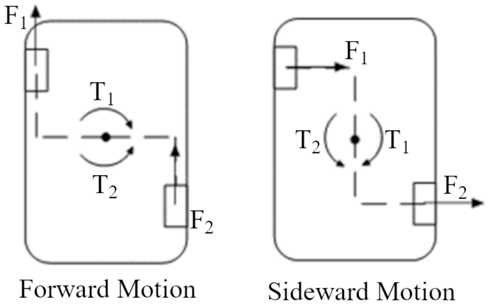


Figure 5.4: Cancellation of Net Torques about the Centroid

The equations that validate the effect shown in figure 5.4 are given in equation 5.1 and equation 5.2.

$$\sum M = T_1 + T_2 \tag{5.1}$$

[2] Image adapted from public domain

Since, in equation 5.1, $T = r \times F$:

$$\sum M = F_1 r_1 + F_2(-r_2) = 0 \tag{5.2}$$

As $F_1 = F_2$ and $|r_1| = |r_2|$ in equation 5.2.

$\sum M$	=	Sum of Moments about the Centroid	Nm
T_i	=	Torque Generated by Tractive Force i	Nm
F_i	=	Tractive Force Generated by the i'th Drive Unit	N
r_i	=	Perpendicular Distance of the i'th Force from the Centroid	m

Since the 19-inch rack proved to be unsuitable for this AGV, the layout of the AGV was changed from what is shown in figure 5.2 to figure 5.5.

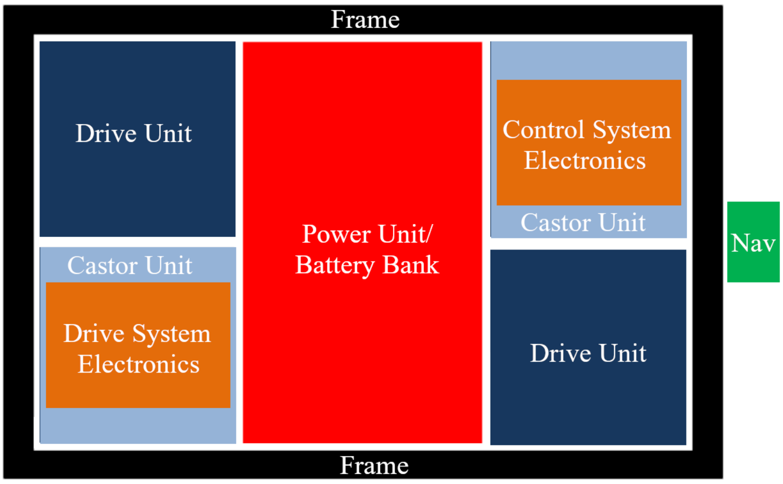


Figure 5.5: Final Block Diagram Layout of The AGV

The main power pack or battery pack of the AGV remained in the centre of the vehicle (see figure 5.5); this was to balance the weight within the system as evenly as possible over the four wheels.

The navigation system of the AGV (a lidar scanner) was moved to the exterior of the AGV. It was initially intended to be embedded within the AGV with a "mailbox" slot to allow the 2D lidar beam to see the outside world. The idea behind this was to protect the lidar from possible external damage, but it was deemed infeasible due to the inflexibility of this mounting and the constriction of the lidar's viewing angle to 180 degrees. The exterior mounting allowed the height of the lidar to be changed easily and, when mounted above the highest point of the AGV's body, gave the sensor a 350° viewing arc.

Since the castor units have no drive train or controlled steering mechanism, they are relatively compact compared to the drive units. This compact size allowed the electronics to be mounted in hot-swappable boxes above them, as illustrated in figure 5.5.

5.3 Drive Unit Design

The drive units of the AGV presented the most significant mechanical design problem of the entire research project since the units had to be mechanically compact and modular. This modularity meant that the frame of the AGV could not be used as a structure to build upon, but rather the drive units had to contain their own skeletal frame for support. The basic layout of the drive unit is shown in figure 5.6; from this basic template, concepts were fleshed out to achieve a functional unit.

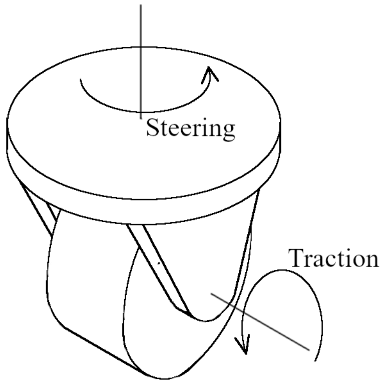


Figure 5.6: Basic Layout of the Drive Unit

The drive units consist of 2 motors, one for producing the tractive effort and one for steering. Since the unpowered castor units will need a castor offset to operate, a castor offset of equal magnitude must be incorporated into the drive units.

Three design ideas were generated, all of which used a sizeable slewing bearing, similar to those found in tank turret or the rotational interface between the cab and tracks of an excavator. This approach was chosen due to lessons learned from Chikosi [49], who attempted to use two large (40mm bore) bearings (not a slew bearing) and found that the system wobbled excessively. The use of two bearings also meant that the AGV stood very tall, raising the centre of gravity. Figure 5.7 illustrates this raised centre of gravity. When a slewing bearing is used, the distance between the wheel centre and bearing interface can be drastically reduced, and the large diameter of the slewing bearing (400mm in this case) makes it far more resilient to wobble about the x-axis and y-axis.

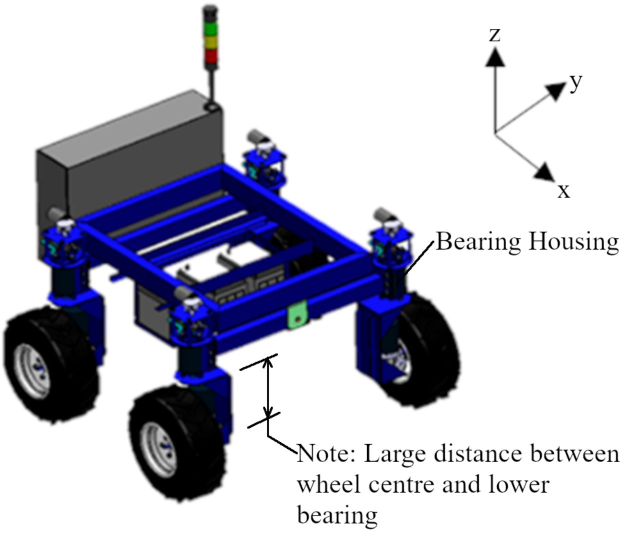


Figure 5.7: Chikosi's AGV[3]

5.3.1 Choosing a Slewing Bearing

All three concepts were based on the same slew bearing, proving to be one of the more problematic items to procure. This unavailability is because slewing bearings of the size required by the research project are relatively rare in the author's experience since slew bearings are predominately produced for much larger applications such as tank turrets and excavator track-cab joints (see figure 5.8). It would be easier to acquire if a slew bearing of that scale were needed. The only off-the-self slew bearings the author could find were produced as part of a truck trailer's steering mechanism, specifically the Jost KLK 400L, which had a rated radial and axial force of 7.5 kN or 765 kg.

[3] Image adapted from Chikosi [49].

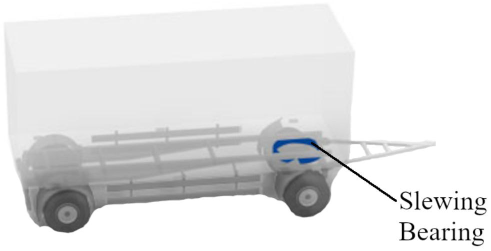


Figure 5.8: Slewing Bearing Intended Usage[4]

5.3.2 Drive Unit Conceptual Designs

Three conceptual designs for the drive unit will be discussed in the paragraphs to follow.

5.3.2.a Drive Unit Design 1 Concept

Design 1 is the simplest conceptually. In design 1, the wheel is mounted directly to the traction motor via an angled gearbox while the steering motor is mounted in line with the axis of rotation of the slewing bearing. Figure 5.9 illustrates an example of this design, which was taken from another AGV the author is working on unrelated to this research.

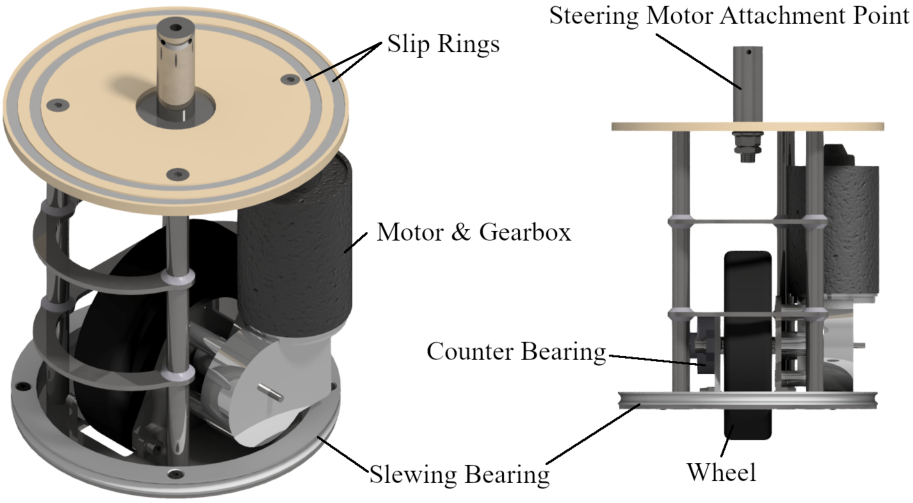


Figure 5.9: Drive Unit Concept 1 - Direct Drive

This idea was abandoned very early on for a couple of reasons, and as such, no initial

[4] Image adapted from JOST [52].

CAD was generated, nor was a suspension system designed. These drawbacks and concerns are listed:

1. Half of the weight supported by wheel is on the gearbox bearings
2. Difficult to balance system as when wheel is centred relative to slew bearing the gearbox and motor assembly will hang off the side
3. Difficult to power motor as a slip ring will be needed to transfer power and control signals to the motor through the slew ring rotational interface
4. Potentially expensive due to high amperage slip ring and gearbox with high radial load bearings

However, this idea is not without merits, especially for smaller AGVs where the overall load is smaller than the load of the AGV in this thesis. The "lob-sidedness" of the system has a negligible effect on these small AGVs, while the smaller loads also translate to smaller motors that draw less current. This lower current makes sourcing slip rings easier than the AGV in this thesis. These smaller AGVs also tend to have simpler motors reducing the number of slip ring contacts needed. Some of the perceived advantages of this system, especially for smaller AGVs, are listed:

1. Extremely simple mechanically
2. Machining of custom parts kept to a minimum and what parts are machined tend to be very simple

5.3.2.b Drive Unit Design 2 Concept

Design 2, in figure 5.10, is based on a similar idea, but the gearbox and traction motor is moved above the wheel. This relocation was done to balance the system better. This design also attempted to integrate the suspension system directly with the drive mechanism. Integrating the suspension system was done using the trailing-arm suspension system philosophy, utilising a torsion spring near the gearbox.

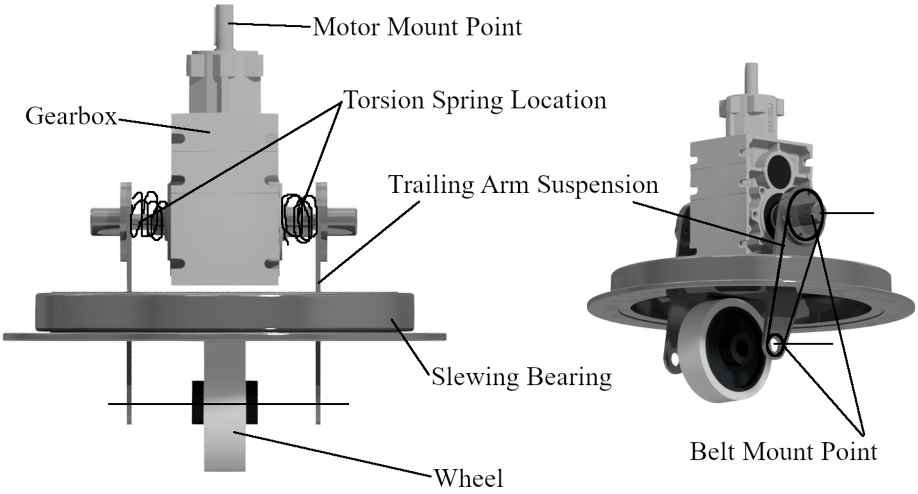


Figure 5.10: Drive Unit Concept 2 - Integrated Trailing Arm Suspension

As seen in figure 5.10 the belt is used to attach the gearbox to the wheel axle. This system has the following drawbacks; firstly, the trailing arm length is constrained by the length of standard timing belts as there is not enough space to add a separate idler. Secondly, there is also the torsion spring issue; no available spring could be found that would produce the necessary torque to support the vehicle and fit in the desired dimensions. Advantages include:

1. Evenly balanced system in terms of weight
2. Integrated suspension system

The inclusion of the trailing arm system has another advantage; namely, the author is familiar with this system, having already built it in the research project "Modular Electric Automatic Guided Vehicle Suspension-Drive Unit" [33]. This system, like concept 1, will also require a high current slip ring; other issues that inhibit this system include:

1. Fixed incremental length of trailing arms due to belt size limitations
2. Difficulty sourcing a torsion spring to meet the needs of this design
3. Variable castor offset due to suspension system

The variable castor offset presents issues when creating a kinematic model. The castor offset distance is random in this configuration(wholly dependent on the driving surface) and will have to be continuously measured to allow the control system to compensate for it in kinematic calculations.

This design was not easily written off, and in figure 5.11 a later iteration of the same concept can be found. This system fixed the problem of variable castor offset by incorporating an idler shaft at the suspension actuation point. This revision also reduced the overall height of the system by reducing the trailing arm angle relative to the ground plane, reducing the castor offset fluctuation. However, this necessitated the use of two separate belt systems. One belt was needed to transfer power from the main gearbox to the idler and the second to transmit power from the idler to the wheel.

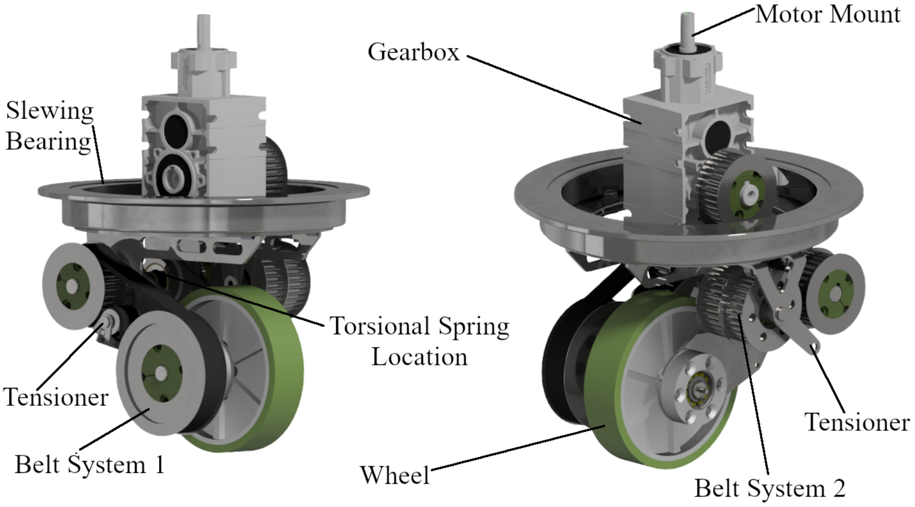


Figure 5.11: Drive Unit Concept 2 - Later Iteration of the Integrated Trailing Arm Suspension

As can be seen in figure 5.11, the system was fast becoming exceptionally mechanically complex. The inclusion of a dual belt system increases the number of required tensioners due to the non-standard distance between pulley centre points. Belt system 2 was made even more complex due to the geometrical constraints imposed by the slewing bearing, which necessitated that belt system 2 essentially made a 90° turn. This 90° turn was achieved by including a second idler shaft and splitting belt system 2 into two subsystems. This design still used a torsional spring to produce the spring effect needed by the suspension system.

Since an appropriate torsion spring could not be found, the design in figure 5.11 was again modified, this time to make use of a compression spring as shown in figure 5.12:

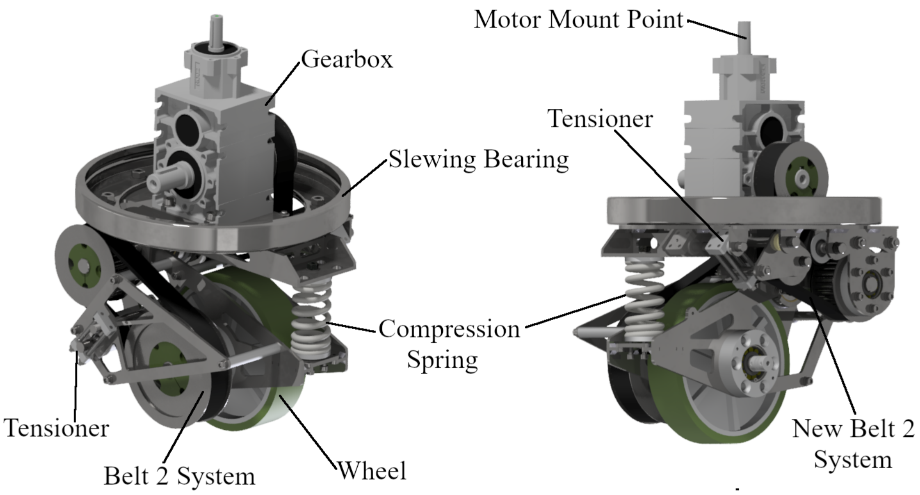


Figure 5.12: Drive Unit Concept 2 - Compression Spring Adaptation of the Integrated Trailing Arm Suspension Drive Unit

Note in figure 5.12 the second belts system was modified to use two smooth idlers to bend a single belt 90° instead of an idler shaft with toothed pulleys and two separate sub belts as seen in figure 5.11. This modification was done as a cost reduction effort as the smooth pulleys were much cheaper to produce than the toothed pulleys.

5.3.2.c Drive Unit Design 3 Concept

The third and final design replaced the trailing arm suspension system with the inline-suspension system. Exchanging the suspension system type alleviated several issues encountered with the design in concept 2 (see figure 5.12), which was extremely complex and had the major disadvantage of a randomly variable castor offset. This new drive unit is illustrated in figure 5.13.

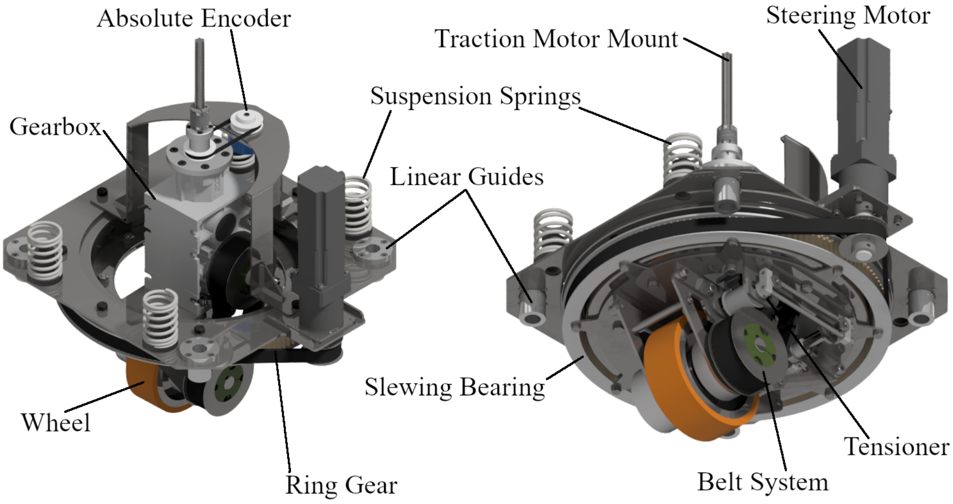


Figure 5.13: Drive Unit Concept 3 - Drive Unit with In-Line Suspension System

The third concept (see figure 5.13) has a lower profile than the second concept (figure 5.12); this helps with the overall height of the final AGV. As can be seen from figure 5.13 the inline suspensions system makes use of 4 springs in parallel rather than the one found in concept 2. Increasing the number of springs reduces the spring constant each spring needs, making them easier to obtain or manufacture due to their lower specifications. This design also significantly reduces the complexity of the belting system needed to transfer power from the gearbox to the wheel as the idler shaft (at the actuation point of the trailing point suspension system) is not needed. Thus, a simple two pulley system is sufficient with a single belt tensioning system.

In figure 5.13 the position of the steering motor and the feedback encoder is shown. The feedback encoder is an absolute encoder that uses a low-cost, precision potentiometer. A potentiometer can be used in this application only due to the steering system's slow and infrequent actuation; the potentiometer's accuracy can be leveraged against the relative encoder built into the steering motor to improve accuracy further.

This system has the following advantages when compared to the other two concepts explored:

1. Evenly balanced system in terms of weight
2. Constant castor offset
3. Integrated suspension system
4. Simple belt system

The disadvantages are as follows:

1. No camber conformity in suspension
2. Still more complex than the direct drive system of concept 1
3. Linear guides for suspension system present a weak point

Not to be underestimated in this system is the constant castor offset, which dramatically improves the stability of the control system.

5.3.3 Drive Final Design

The final design was determined from the previous concepts using a cost-benefit matrix.

5.3.3.a Drive Unit Selection Matrix

The cost-benefit evaluation matrix can be found in table 5.1.

Table 5.1: Comparison of Drive Unit Concepts

Concept	**Advantages**	**Disadvantages**
Concept 1	·Extremely simple ·Comparably inexpensive to build ·Steering and traction divorced ·Constant Castor Offset	·High current slip rings required ·Lob-sided weight loading ·Gearbox supports large portion of weight
Concept 2	·Symmetrical weight distribution ·Integrated suspension system ·Suspension system has good camber conformity	·Variable castor offset ·Exotic springs needed ·Extremely mechanically complex ·Trailing arm length restrictions
Concept 3	·Symmetrical weight distribution ·Constant Castor Offset ·Integrated suspension system ·Mechanically simpler than concept 2	·No camber conformity ·Linear guides of suspension are a weak point

As mentioned in the previous section, concept 1 can be immediately be excluded due to the need for high current slip rings that proved hard for the author to find at reasonable prices that would fit within the project budget. The uneven weight distribution would also put unnecessary strain on the slewing bearing. That left only concepts 2 and concept 3. Of the two systems, concept 2 was both more expensive and more mechanically complex due to the multiple idler shafts and belting systems needed. Not to forget, concept 2 also had the fatal flaw of a randomly variable castor offset, making control of the AGV more difficult. Therefore concept 3 proved to be the only viable choice for this application.

5.3.3.b Completion of the Drive Unit CAD Design

Since concept 3 was chosen, it could be further refined to produce a manufacturable unit. Discussion of that refinement is done in this section.

Modularity Skeleton

The first refinement performed was the creation of a framework that allowed the drive unit to be "stand-alone" relative to the main body (for modularity purposes). This skeleton is illustrated in figure 5.14:

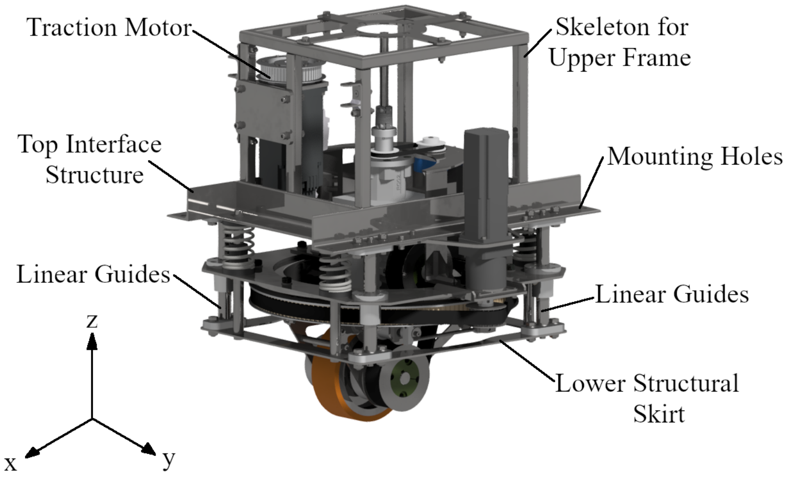


Figure 5.14: Final Drive Unit with Modular Skeleton

The skeleton used to support the drive units was primarily made from standard ISO steel sections, chiefly square tube and angle iron. This skeleton is attached to the main body of the AGV via the "Top Interface Structure" and "Lower Structural Skirt" (see figure 5.14). The entirety of the AGV's weight along the z-axis is transferred between the main body and drive unit via the "Top Interface Structure"; the "Lower Structural Skirt supports no weight". The "Lower Structural Skirt" aims solely to assist with force experienced radially (i.e. in the x and y directions). The "Lower Structural Skirt" adds much-needed rigidity to the linear guides and counters moments of rotation developed around the x and y axes.

Traction Motor Linkage

As can be seen in figure 5.14, the traction motor forms part of the "sprung load" (i.e. on the body side of the AGV) while the gearbox it attaches to forms part of the "unsprung load" (wheel side of the suspension system). This separation presented the unique problem of transferring rotational power across the suspension system barrier. Typically in automotive vehicles, this is done using a CV joint. A similar solution was developed for this AGV and is called the "vertical compromiser", illustrated in figure 5.15.

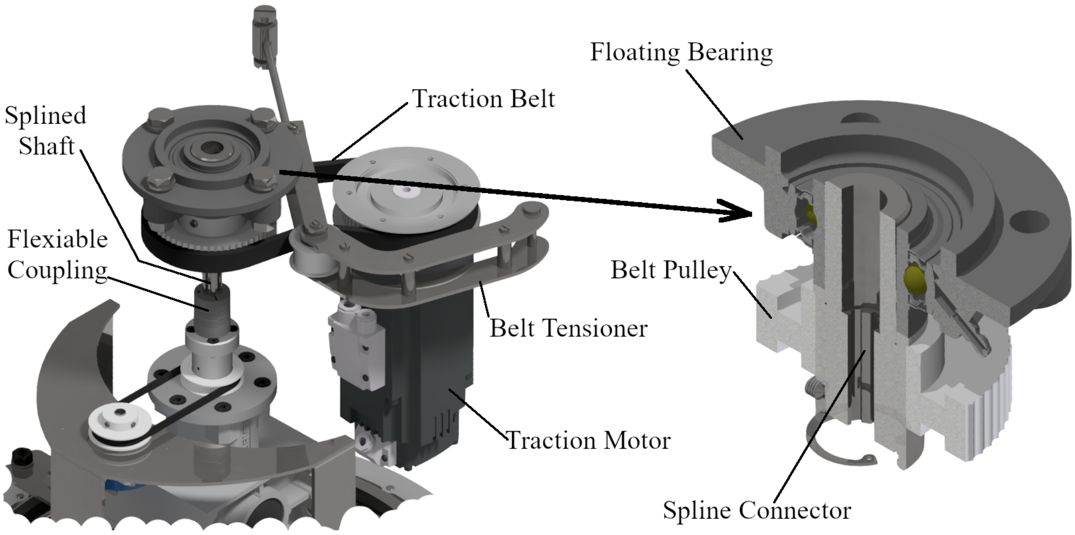


Figure 5.15: Final Drive Unit with Vertical Compromiser

Wheel

The wheels used for this design were a cast polyurethane tyre on a cast iron core. According to Castor and Ladder [53], cast polyurethane wheels create minimal noise, have low rolling resistance (when compared to other wheel materials), do not readily damage the floor and are resistant to both cuts and tearing. These characteristics make cast polyurethane ideal for an industrial environment. Specifications for the chosen wheel (WPU 062) can be found in table 5.2.

Table 5.2: WPU 062 Wheel Specifications

Specification	**Value**
Wheel Size	150×50 mm
Pilot Bore	17 mm
Load Rating	300 kg

Final Design

The final design of the drive unit is shown in figure 5.16; this includes the vertical compromiser, modular connection interface and chosen design concept.

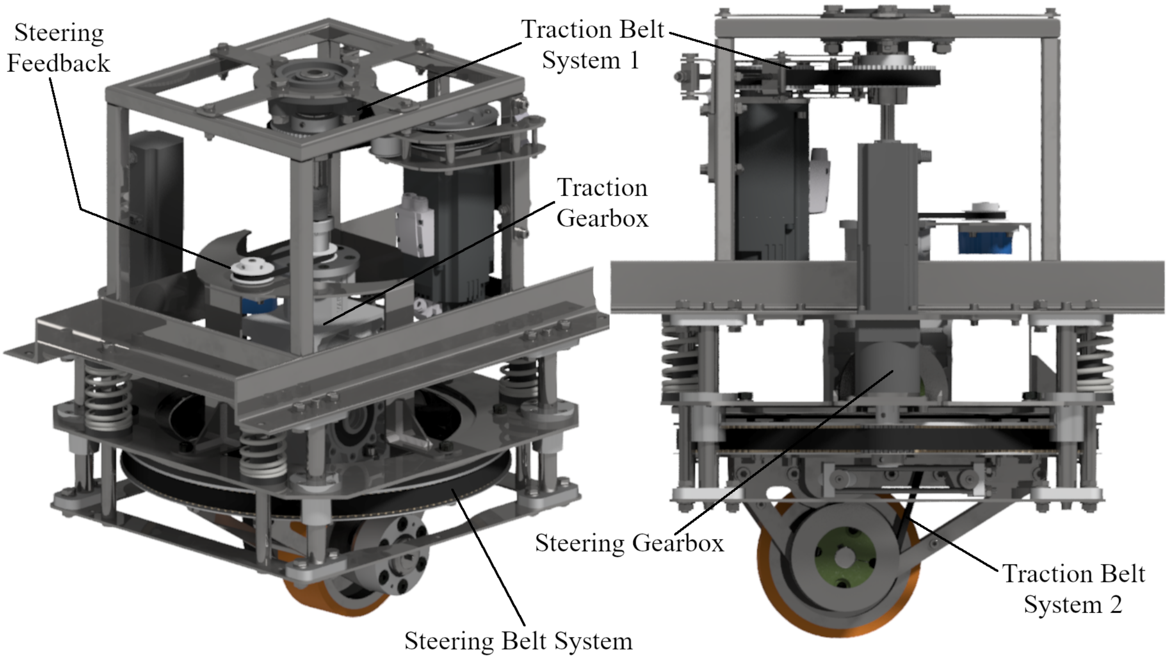


Figure 5.16: Finalised CAD Model of the AGV Drive Unit

5.3.4 Drive Unit: Traction System Detailed Analysis

This section deals with explaining the detailed mathematics of the drive unit. It also expands on minor mechanical components that were glossed over in the previous section.

5.3.4.a Tractive System Overview

The tractive drive train consists of three parts; the upper/ traction motor belts system (called belt system 1 from here on), the main bevel gearbox and the wheel side belt system (called belt system 2 from here on). Instead of the more common worm gearbox, a bevel gear type gearbox was used for the primary gearing system. The choice of a bevel gearbox was made to allow bi-directional torque transfer. That is to say, when the motor is running, it can turn the wheel, but when the motor is off, the inertia of the wheel can travel back through the gearbox to turn the motor. This bidirectional motion transfer will allow for regenerative braking since instead of dumping the braking current generated by the motor's drive into a braking resistor, it can be used to recharge the battery bank. The layout of the tractive drive train is described in figure 5.17.

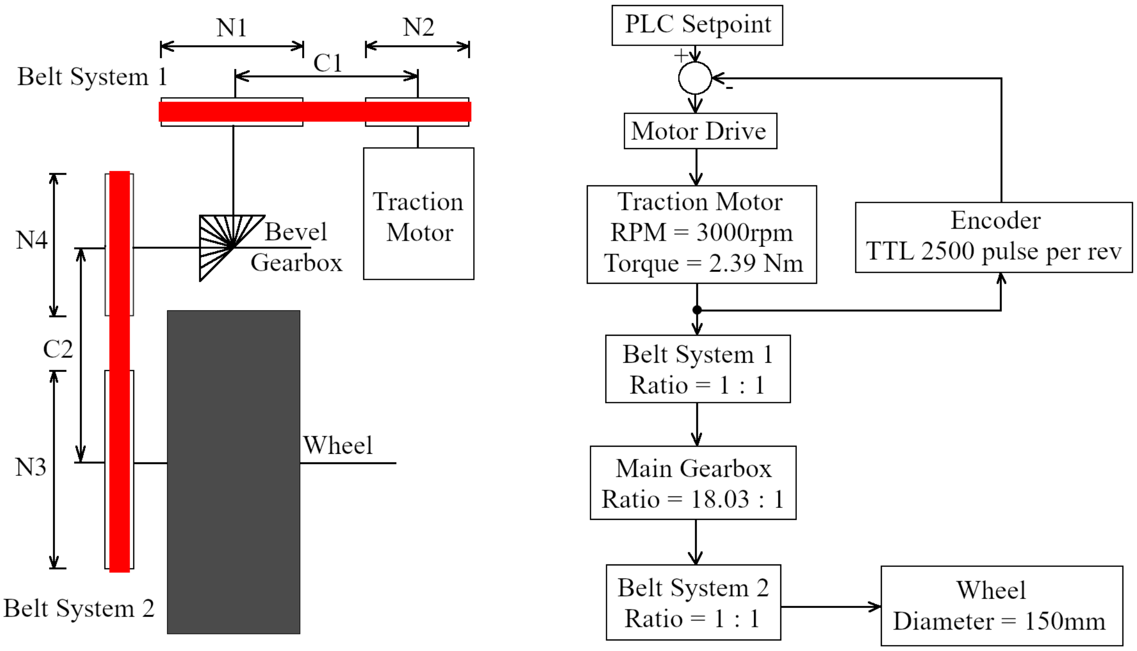

N1
N2
C1
Belt System 1
Traction Motor
Bevel Gearbox
N4
C2
N3
Wheel
Belt System 2
PLC Setpoint
+
-
Motor Drive
Traction Motor RPM = 3000rpm Torque = 2.39 Nm
Encoder TTL 2500 pulse per rev
Belt System 1 Ratio = 1 : 1
Main Gearbox Ratio = 18.03 : 1
Belt System 2 Ratio = 1 : 1
Wheel Diameter = 150mm

Figure 5.17: Tractive Belt System

It should be noted that there is only one feedback loop in this system (as shown in figure 5.17), which corrects for the traction-motor output speed relative to the setpoint speed given by the primary kinematic control system (a PLC in this case).

5.3.4.b AGV Tractive Requirements

Before the drive train described in figure 5.17 could be developed, it was first necessary to calculate the various specifications required of this system. These values are calculated with the aid of the equations developed in section 3.2 and the software tool described in chapter 4. The global requirements of the AGV are summarised in table 5.3, many of these requirements are listed as design specifications in section 1.2.1.

Table 5.3: Tractive Global Requirements

Specification	**Value**
Net AGV Mass	$600kg$
Maximum Desired Speed	$1.3^{m}/_{s^2}$
Time to Reach Max Speed	$2s$
Maximum Climbable Incline	5
Allow Acceleration on Incline	No
Wheel Diameter	$150mm$
Rolling Friction of Wheels (on concrete)	0.02
Number of Drive Units	2

5.3.4.c Net Forces on Centroid of AGV

Since acceleration up an incline is forbidden, the three cases that will be evaluated with regards to the net force required to drive an AGV of 600 kg are:

- Constant velocity on a flat surface
- Acceleration on a flat surface
- Constant velocity up an incline

Constant Velocity on a Flat Surface

When the AGV is travelling at a fixed velocity on a flat surface, the net force required for this motion, as directed at the centroid, can be calculated using equation 3.19 and is:

$$F_t = \mu_{rr} mg = 0.02(600)(9.8) = 117.76N \tag{5.3}$$

Acceleration on a Flat Surface

When the AGV is accelerating from rest to $1.3^{m}/_{s^2}$ on a flat surface, the net force required for this motion, as directed at the centroid, can be calculated using equation

3.21 and is:

$$F_t = m\left[\left(\frac{v_f - v_i}{t}\right) + \mu_{rr}g\right] = 600\left[\left(\frac{1.3-0}{2}\right) + 0.02(9.8)\right] = 507.72N \tag{5.4}$$

Constant Velocity up an Incline

When the AGV climbs an incline of 5°, at a constant velocity, the net force required for this motion, as directed at the centroid, can be calculated using equation 3.25 and is:

$$\begin{aligned} F_t &= m\left(g\left[\sin(\theta) + \mu_{rr}\cos(\theta)\right] + \left[\frac{v_f - v_i}{t}\right]\right) \\ &= 600\left(9.8\left[\sin(5) + 0.02\cos(5)\right] + \left[\frac{1.3-0}{2}\right]\right) \\ &= 630.27N \end{aligned} \tag{5.5}$$

Worst Case Scenario

When comparing the force required in equation 5.3, equation 5.4 and equation 5.5 it can be seen that the action "constant velocity up an incline" is the largest. Therefore, this is the worst case the drive train will have to deal with at 630.27 N.

5.3.4.d Tractive Requirements at Each Wheel

Each wheel's tractive requirements will include the following: the RPM of each wheel, the torque required by each wheel and the RPM of each wheel.

RPM Required by the AGV's Wheels

No slippage is considered here; as such, the AGV wheels' RPM will be equal. Since the desired top speed of the AGV is 1.3 $^{m}/_{s^2}$, the wheel RPM to match this can be calculated using equation 5.6 derived from Serway [46].

$$RPM_{wheel} = \frac{2\pi}{60}\left(\frac{v_{AGV}}{r_{wheel}}\right) \tag{5.6}$$

RPM_{wheel}	=	Wheel revolutions per second	$^{r}/_{min}$
v_{AGV}	=	Velocity of the AGV	$^{m}/_{s^2}$
r_{wheel}	=	Radius of the AGV's Wheels	m

$$RPM_{wheel} = \frac{2\pi}{60}\left(\frac{1.3}{\left(\frac{150\times10^{-3}}{2}\right)}\right) = 165.521\ {}^{r}/_{min} \tag{5.7}$$

Torque Required by the AGV's Wheels

Since there are three operational modes for the AGV (constant velocity on a flat surface, acceleration on a flat surface and constant velocity up an incline), the torque requirements of each of these manoeuvrers will be calculated. For simplicity, no slippage will be assumed, and the load distribution on the AGV will be equal. Torque can be estimated using equation 5.8 adapted from Serway [46].

$$\tau_{wheel} = \frac{r_{wheel} \times F_t}{n_{drive\ units}} \tag{5.8}$$

τ_{wheel}	=	Required wheel torque	Nm
$n_{drive\ units}$	=	Number of drive units on the AGV	

Thus, for constant velocity on a flat surface:

$$\tau_{wheel} = \frac{(75 \times 10^{-3}) \times 117.76}{2} = 4.415\ Nm \tag{5.9}$$

Acceleration on a flat surface:

$$\tau_{wheel} = \frac{(75 \times 10^{-3}) \times 507.72}{2} = 19.040\ Nm \tag{5.10}$$

Constant velocity up an incline of 5:

$$\tau_{wheel} = \frac{(75 \times 10^{-3}) \times 630.27}{2} = 23.635\ Nm \tag{5.11}$$

Power Required at Each Drive Wheel

Only the overall power is required for each wheel; therefore, only the case "constant velocity up an incline" will be evaluated since it requires the highest torque. The required wheel torque is calculated using equation 5.13 from Shigley [54].

$$P_{wheel} = \tau_{wheel}\omega_{wheel} = \tau_{wheel}\left(\frac{v_{agv}}{r_{wheel}}\right) \tag{5.12}$$

P_{wheel}	=	Required wheel power	W

Thus the maximum required wheel power is:

$$P_{wheel} = 23.635\left(\frac{1.3}{75 \times 10^{-3}}\right) = 409.676\ W \tag{5.13}$$

Power Required at Each Traction Motor

Intuitively, one may think that the traction motor power equals the needed wheel power. This assumption, however, does not take into account the drive train's inefficiency, which according to Shigley [54], is likely around 94%. Thus the power required of the motor will be:

$$P_{motor} = P_{wheel} + (1 - 0.98)P_{wheel} = 417.87W \tag{5.14}$$

P_{motor}	=	Required motor power	W

5.3.4.e Traction Motor Selection

Since the power of the traction motor is known thanks to the previous section, an appropriate motor can be found. The motor selection was made before selecting the gearbox and belting system, as the motor choices (at the time of writing) had fewer options available than the gearing system. It is easier to match a gear train to the motor than to match a motor to the gear train.

The motor selected was a Siemens Simotics S-1FL6 1FL6042-2AF21-1AH1. This motor is an induction type motor that uses 3 phase AC. The advantages of this motor type over other motor types include; its relative cost-effectiveness, its abundance as an off-the-shelf item and its steady torque over a sizable speed range.

This motor has a power rating of 750 W, which is more than sufficient for this application. The motor is 55% larger than is required by the system. The specifications for this motor can be found in table 5.4:

Table 5.4: 1FL6042-2AF21-1AH1 Traction Motor Specifications

Specification	**Value**
Power	750 W
Nominal Torque	2.39 Nm
Nominal Speed	3000 $^{r}/_{min}$
Nominal Current Draw	4.65 A
Peak Torque	2.39 Nm
Peak Speed	5000 $^{r}/_{min}$
Peak Current Draw	4.7 A
Nominal Voltage	111V

It is important to note that this motor has a built-in parking brake and feedback encoder. The parking brake can only be used to lock the rotor when the motor is stopped. If the motor is braked while running, the braking is done using a braking resistor. The parking brake is "normally on", which is ideal for this application, as when the AGV is off or unpowered, it should not be able to move for safety reasons. The encoder included in the motor is a relative encoder. The motor drive electronics use this encoder as feedback to ensure that the motor is spinning at the setpoint speed desired by the central controller. The control system can also benefit by using this value as part of its control loop structure. The encoder is a TTL 2500 pulse per revolution encoder. Essentially this encoder generates a 5 V square wave (TTL) which will take 2500 pulses to complete 360 of rotation. The speed of the rotor can be determined by differentiating the angle (number of pulses) relative to time.

5.3.4.f Tractive Gear Train

The tractive gear train system, as illustrated in figure 5.17, consists of three sections. The first belt system, the main bevel gearbox and the second belt system. Although using an induction motor means that the torque produced by the motor is relatively steady over the motor's speed range, it is still good practice to run the motor at its nominal speed to get the best power efficiency and largest torque. Thus the gear train was designed primarily such that a traction-motor speed of 3000 $^{r}/_{min}$ translated to a vehicle body speed of 1.3 $^{m}/_{s^2}$ (or wheel speed of 165.521 $^{r}/_{min}$).The necessary gear ratio to make this possible is calculated in equation 5.15.

$$i = \frac{\omega_{driven}}{\omega_{drive}} = \frac{3000}{165.521} \approx 18.12 \quad (18.12 : 1) \tag{5.15}$$

If a ratio of 18.12 : 1 is used, then the maximum torque produced by the motor at the wheel end will be:

$$\tau_{wheel} = i(\tau_{motor})e_{gear\ train} = 18.12(2.39)0.94 = 40.708\ Nm \tag{5.16}$$

$e_{gear\ train}$ = Predicted efficiency of the gear train

Where the predicted efficiency is the product of the efficiency of the sub-components as given in equation 5.17, these values were estimated using Shigley's [54] lookup tables.

$$\begin{aligned} e_{gear\ train} &= e_{belt1}e_{gearbox}e_{belt2} \\ &= 0.98 \times 0.98 \times 0.98 \approx 0.94 \end{aligned} \tag{5.17}$$

Since the torque required at the wheel is only 23.635 Nm, which is less than the potential 40.708 Nm a 18.12 : 1 gearbox can produce, this gearbox ratio choice is valid.

The next step in the design sequence was to find a bevel gearbox and belt arrangement that could close to the desired ratio of 18.12 : 1. This selection was made using the program developed in chapter 4 which performed a brute force optimisation using a list of known belts, pulleys and bevel gearboxes. It did this by working out all the possible gear ratios given the known pulleys, then sequentially multiplied them with know bevel gearbox ratios. These results were ordered according to their closest match to the desired ratio. The system also eliminated belt systems and gearboxes unable to withstand the required loads (i.e. a 5mm wide belt in belt system 2 would most likely break under the applied tension due to the high torque but would be fine in belt system 1, where the tension is less system 2, but the speed is faster).

The rather unsurprising result was that the system selected a ratio of 1 : 1 for belt system 1, 18.03 : 1 for the bevel gearbox and 1 : 1 for belt system 2. Which, when multiplied together, gives a total gear train ratio of:

$$i_{gear\ train} = i_{belt\ 1}(i_{bevel\ gearbox})i_{belt\ 2} = 18.03 \quad (18.03 : 1) \tag{5.18}$$

Thus using the actual gear ratio available, in equation 5.18, the top speed of the AGV will be (when the motor RPM = 3000 $^{r}/_{min}$):

$$\begin{aligned} v_{AGV} &= r_{wheel}\left(\frac{2\pi}{60}\left(\frac{RPM_{motor}}{i_{gear\ train}}\right)\right) \\ &= 75\times10^{-3}\left(\frac{2\pi}{60}\left(\frac{3000}{18.03}\right)\right) = 1.306\ ^{m}/_{s^2} \end{aligned} \tag{5.19}$$

The maximum torque that the AGV could develop per wheel will be (using equation 5.16 as a template):

$$\tau_{wheel\ max} = 18.03(2.39)0.94 = 40.506\ Nm \tag{5.20}$$

The required torque from the traction motor is calculated in equation 5.21, provided that the required maximum wheel torque (as given in equation 5.11) is 23.635 *Nm*.

$$\begin{aligned} \tau_{motor\ required} &= \left(\frac{\tau_{wheel}}{i_{gear\ train}}\right)(1 + (1 - e_{gear\ train})) \\ &= \left(\frac{23.635}{18.03}\right)(1 + 0.06) = 1.390\ Nm \end{aligned} \tag{5.21}$$

5.3.4.g Tractive Gear Train Additional Values

With the ratios of each section of the gear train calculated, along with the torque and RPM of the motor calculated, the torque and RPM of each segment of the gear train can be found as illustrated in the equations that follow:

Belt System 1 Values

This section will calculate the torque and RPM values along with the force of tension in the belt. Also included in this section is the dimensional layout of the system, which is illustrated in figure 5.18.

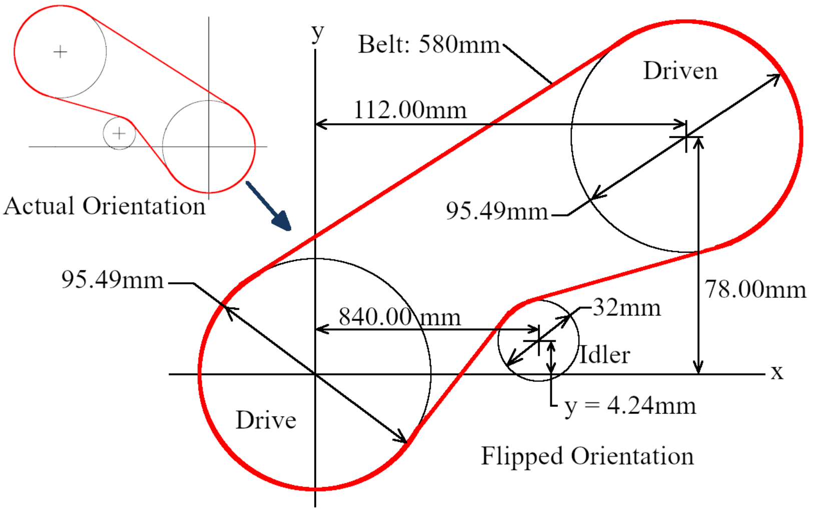


Figure 5.18: Belt System 1 Actual Layout

In figure 5.18, the belt system is flipped from its "actual orientation" (the orientation that would be recognisable on the AGV when viewed from the outside) to a "flipped orientation" this was done to ensure that the dimensional values matched the orientation expected by the AGV calculator given in chapter 4.

The AGV calculator is used to find the missing y-axis dimension for the idler pulley (in the case of this system, it is 4.24 mm). Since the belt sizes are standardised, and the geometry of the drive and driven pulley are fixed relative to each other, the only way to tension the belt is by making one of the component axes of the idler pulley settable. This settable dimension is calculated as the missing y value (the idler x dimension is arbitrarily chosen by the designer, which is used for the y value calculation). The AGV calculation tool also eliminated any belts systems that could not transfer the required torque and belt tension.

The RPM value of the drive pulley is directly attached to the traction motor and, as such, will run at 3000 $^{r}/_{min}$. The torque that this pulley experiences at full AGV load will be the same as the motor torque (calculated in equation 5.21) and is 1.390 Nm.

Since the drive and driven pulleys have identical diameters, the speed of the driven pulley will also be 3000 $^{r}/_{min}$. However, it was stated that this entire gear train is expected to have an efficiency of 94% (with each of the three sections having an efficiency of 0.98%); thus, the torque on this pulley is expected to be:

$$\begin{aligned}\tau_{belt1\ driven} &= \tau_{belt1\ drive}(e_{belt1}) \\ &= 1.390(0.98) = 1.362\ Nm\end{aligned} \tag{5.22}$$

The force of tension in the belt (F_T) can be calculated using equation 5.23. Note for this calculation, the highest torque the traction motor can produce is used and not the maximum operating torque is given in equation 5.21. Calculating using the highest torque rather than the operating torque was done for safety reasons.

$$\begin{aligned}|\vec{F_T}| &= \frac{\tau_{motor\ max}}{r_{belt1\ drive}} \\ &= \frac{2.39}{47.745 \times 10^{-3}} = 50.06\ N\end{aligned} \tag{5.23}$$

$|\vec{F_T}|$	=	Maximum tension force magnitude in belt	N
$\tau_{motor\ max}$	=	Maximum torque the traction motor is capable of	Nm
$r_{belt1\ drive}$	=	Pitch radius of traction motor drive pulley	m

Bevel Gearbox Values

Since the input to the bevel gearbox unit is directly attached to the driven pulley of belt system 1, the torque and RPM will be equivalent. Thus 1.362 Nm and 3000 $^{r}/_{min}$ respectively.

The gearbox was chosen using the AGV design program. It chose the gearbox from a list of Varvel gearboxes of varying ratios and torque allowances. Since the chosen bevel gearbox unit (BGU) has a gear ratio of 18.03 : 1. The torque and RPM at the output of this gearbox are given in equation 5.24 and equation 5.25.

$$\begin{aligned}\omega_{gearbox\ out} &= \frac{\omega_{gearbox\ in}}{i_{gearbox}} \\ &= \frac{3000}{18.03} = 166.389\ ^{r}/_{min}\end{aligned} \tag{5.24}$$

$$\begin{aligned}\tau_{gearbox\ out} &= \tau_{gearbox\ in}(i_{gearbox})e_{gearbox} \\ &= 1.362(18.03)0.98 = 24.066\ Nm\end{aligned} \tag{5.25}$$

The maximum torque that can be developed on the output of the gearbox (when the traction motor reaches its torque saturation point) is calculated in equation 5.26. This result is used to determine the tension force in belt system 2.

$$\begin{aligned}\tau_{gearbox\ maxout} &= \tau_{motor\ max}(i_{gearbox})(e_{gearbox} \times e_{belt1}) \\ &= 2.39(18.03)(0.98 \times 0.98) = 41.385\ Nm\end{aligned} \tag{5.26}$$

Belt System 2 Values

The output torque and RPM of the bevel gearbox will be equivalent to the drive pulley of belt system 2 due to their direct mechanical coupling, which is 166.389 $^{r}/_{min}$ and 24.066 Nm respectively. The geometry of belt system 2 is illustrated in figure 5.19.

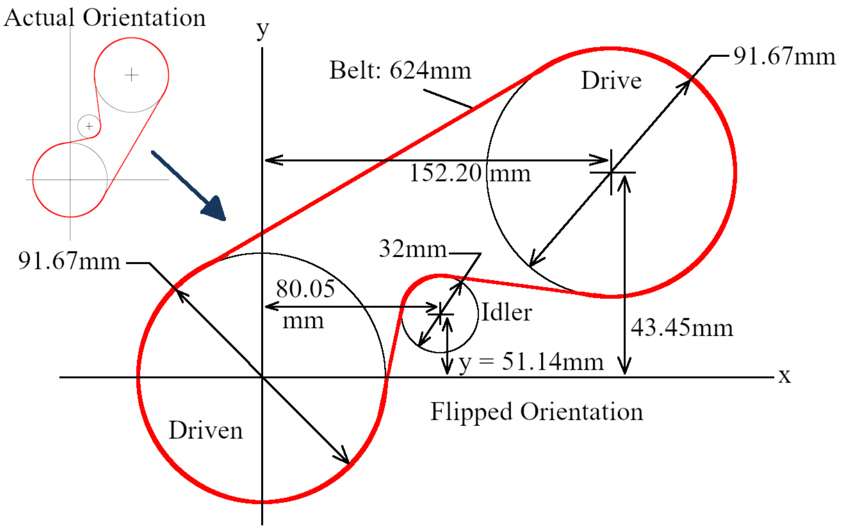


Figure 5.19: Belt System 2 Actual Layout

Once again, the belt system is flipped to make the dimensions directions match with what the AGV calculator in chapter 4 expects. Similarly to belt system 1, the AGV calculator was used to select the belt type and y distance of the idler pulley.

Since the ratio of belt system 2 is 1 : 1, the RPM of the driven pulley will be 166.389 $^{r}/_{min}$, and the torque will be as calculated in equation 5.27

$$\begin{aligned}\tau_{belt2\ driven} &= \tau_{belt2\ drive}(e_{belt2}) \\ &= 24.066(0.98) = 23.585\ Nm \approx 23.635\ Nm \ \text{ from equation 5.11}\end{aligned} \tag{5.27}$$

The tension in the belt used in belt system 2 was calculated using the maximum torque value possible out of the bevel gearbox (from equation 5.26) for safety reasons and is given in equation 5.28.

$$
\begin{aligned}
|\vec{F_T}| &= \frac{\tau_{gearbox\ maxout}}{r_{belt2\ drive}} \\
&= \frac{41.358}{45.835 \times 10^{-3}} = 902.32\ N
\end{aligned}
\tag{5.28}
$$

$\lvert\vec{F_T}\rvert$	=	Maximum tension force magnitude in belt	N
$\tau_{gearbox\ maxout}$	=	Maximum torque the bevel gearbox can output	Nm
$r_{belt2\ drive}$	=	Pitch radius of the drive pulley	m

5.3.4.h Idler Pulley Considerations

This section calculates the resultant forces present on the idler pulleys in the upper and lower belt systems to select the appropriate bearings using SKF's bearing selection tool[5].

Belt System 1

The belt orientation and force diagram for the idler pulley found in belt system 1 is illustrated in figure 5.20.

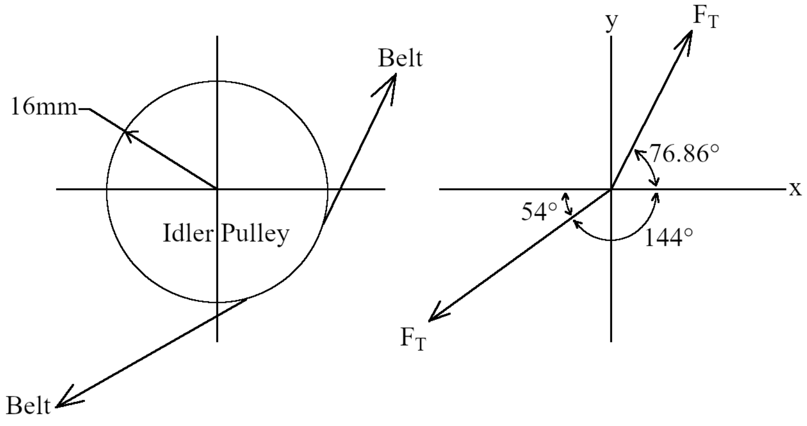


Figure 5.20: Belt System 1 Idler Pulley Force Diagram

[5] The bearing selection tool can be found at "https://www.skfbearingselect.com/#/bearing-selection-start"

The top speed of this idler pulley will occur when the traction motor is spinning at 3000 $^{r}/_{min}$. This speed value is calculated in equation 5.29.

$$\begin{aligned} RPM_{idler\ 1} &= \left(\frac{r_{belt1\ drive}}{r_{idler}}\right) RPM_{motor} \\ &= \left(\frac{47.745}{16}\right) 3000 = 8\ 952.12\ ^{r}/_{min} \end{aligned} \tag{5.29}$$

Using figure 5.20 and the belt tension calculated in equation 5.23, it is possible to calculate the x-component and y-component forces acting on the idler pulley.

In the x-direction:

$$\begin{aligned} \vec{F_x} &= |\vec{F_T}| \left[\cos(76.86)\hat{\imath} - \cos(54)\hat{\imath}\right] \\ &= 50.06 \left[\cos(76.86)\hat{\imath} - \cos(54)\hat{\imath}\right] = -18.04\hat{\imath}\ N \end{aligned} \tag{5.30}$$

In the y-direction:

$$\begin{aligned} \vec{F_y} &= |\vec{F_T}| \left[\sin(76.86)\hat{\jmath} - \sin(54)\hat{\jmath}\right] \\ &= 50.06 \left[\sin(76.86)\hat{\jmath} - \sin(54)\hat{\jmath}\right] = 8.25\hat{\jmath}\ N \end{aligned} \tag{5.31}$$

Thus the magnitude of the radial force acting on the bearings in the idler pulley will be:

$$\begin{aligned} |\vec{F_r}| &= \sqrt{\vec{F_x}^2 + \vec{F_y}^2} \\ &= \sqrt{(-18.04)^2 + (8.25)^2} = 19.84\ N \end{aligned} \tag{5.32}$$

The radial force discovered in equation 5.32 and the RPM of the idler pulley, in equation 5.29, were plugged into the SKF calculator along with the desired bearing type and dimensions. From these results, a pair of 61900-2Z bearings were selected. These bearings are estimated by the SKF calculation tool to have a lifespan of $> 2 \times 10^5$ hours (L_{10mh}). A full report for these bearings can be found in appendix A.

Belt System 2

Like belt system 1, belt system 2 also contains an idler pulley, and as such, it is necessary to calculate the RPM and radial forces present to find a suitable bearing pair. The force diagram for this idler pulley is given in figure 5.21

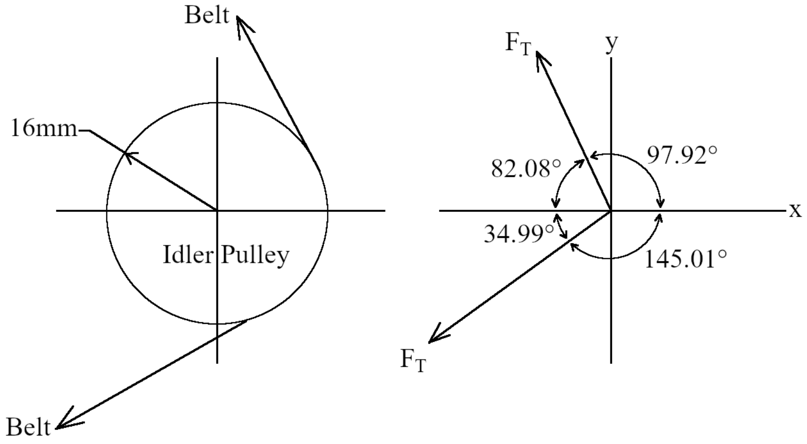


Figure 5.21: Belt System 2 Idler Pulley Force Diagram

The RPM of the idler pulley in belt system 2 can be calculated using equation 5.33 using the values taken from equation 5.24 and figure 5.18.

$$\begin{aligned} RPM_{idler\ 2} &= \left(\frac{r_{belt2\ drive}}{r_{idler}}\right) RPM_{belt2\ drive} \\ &= \left(\frac{45.835}{16}\right) 166.389 = 476.652\ {}^{r}\!/_{min} \end{aligned} \tag{5.33}$$

Using figure 5.21 and the belt tension for this system (calculated in equation 5.28), it is possible to find the component forces for this system as follows.

In the x-direction:

$$\begin{aligned} \vec{F_x} &= |\vec{F_T}| \left[-\cos(82.08)\hat{\imath} - \cos(34.99)\hat{\imath}\right] \\ &= 902.32 \left[-\cos(82.08)\hat{\imath} - \cos(34.99)\hat{\imath}\right] = -863.56\hat{\imath}\ N \end{aligned} \tag{5.34}$$

In the y-direction:

$$\begin{aligned} \vec{F_y} &= |\vec{F_T}| \left[\sin(82.08)\hat{\jmath} - \sin(34.99)\hat{\jmath}\right] \\ &= 902.32 \left[\sin(82.08)\hat{\jmath} - \sin(34.99)\hat{\jmath}\right] = 376.29\hat{\jmath}\ N \end{aligned} \tag{5.35}$$

Utilising equation 5.34 and equation 5.35, the magnitude of the radial force acting on the idler pulley's bearing will be:

$$
\begin{aligned}
|\vec{F_r}| &= \sqrt{\vec{F_x}^{\,2} + \vec{F_y}^{\,2}} \\
&= \sqrt{(-863.56)^2 + (376.29)^2} = 941.98\ N
\end{aligned}
\tag{5.36}
$$

The radial force calculated in equation 5.36 and RPM calculated in equation 5.33 were used to select appropriate bearings using the SKF bearing selection tool. The bearings chosen were a pair of HK 1616.2RS needle bearings. Needle bearings were used in this application due to space limitations. The results of the SKF bearing selection tool can be found in appendix B. These bearings are estimated to have a lifespan of 161 000 hours (L_{10mh}).

5.3.4.i Wheel Bearings Selection

The bearings used for the wheel shaft were sourced using SKF's bearing selection tool. These bearings had to withstand the tensional force created by the belt connection and had to support one quarter the weight of the AGV.

First, it is necessary to calculate the radial force created by tension in the belt on the wheel axle; this was done with the aid of figure 5.23.

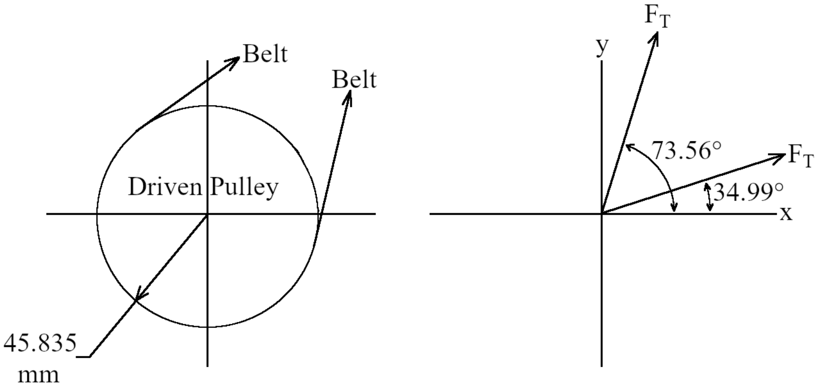


Figure 5.22: Wheel Axle Belt System Force Diagram

Since the force of tension, F_T, is 902.32 N (from equation 5.28), the component forces acting on the bearing will be:

In the x-direction:

$$
\begin{aligned}
\vec{F_x} &= |\vec{F_T}| \left[\cos(73.56)\hat{\imath} + \cos(34.99)\hat{\imath}\right] \\
&= 902.32 \left[\cos(73.56)\hat{\imath} - \cos(34.99)\hat{\imath}\right] = 994.59\hat{\imath}\ N
\end{aligned}
\tag{5.37}
$$

In the y-direction:

$$\begin{aligned} \vec{F_y} &= |\vec{F_T}|\left[\sin(73.56)\hat{j} - \sin(34.99)\hat{j}\right] \\ &= 902.32\left[\sin(73.56)\hat{j} - \sin(34.99)\hat{j}\right] = 1\ 382.85\hat{j}\ N \end{aligned} \tag{5.38}$$

Thus the radial force is:

$$\begin{aligned} |\vec{F_r}| &= \sqrt{\vec{F_x}^2 + \vec{F_y}^2} \\ &= \sqrt{(994.59)^2 + (1\ 382.85)^2} = 1\ 703.37\ N \end{aligned} \tag{5.39}$$

The force on the wheel axle caused by the weight of the AGV is calculated by taking $^1/_4$ of the vehicle's weight as shown in equation 5.40.

$$\begin{aligned} \vec{F}_{agv\ weight} &= \frac{1}{4}m_{agv}g\ \hat{j} \\ &= \frac{1}{4}(600)(9.81) = 1\ 471.5\hat{j}\ N \end{aligned} \tag{5.40}$$

$\vec{F}_{agv\ weight}$	=	Force resultant from AGV mass	N
m_{agv}	=	Mass of the AGV	kg
g	=	Gravitational constant	$^m/_{s^2}$

The force on the AGV's drive wheels, which are a reactionary force that results from the tractive force given in equation 5.5, will be equal to half of the tractive force due to there being two drive wheels.

$$\begin{aligned} \vec{F}_{tractive\ reaction} &= \frac{1}{2}F_t\hat{i} \\ &= \frac{1}{2}(630.27) = 315.14\hat{i}\ N \end{aligned} \tag{5.41}$$

$\vec{F}_{tractive\ reaction}$	=	Tractive reaction force	N
m_{agv}	=	Mass of the AGV	kg
g	=	Gravitational constant	$^m/_{s^2}$

The forces acting on the wheel axle are summarised in figure 5.23

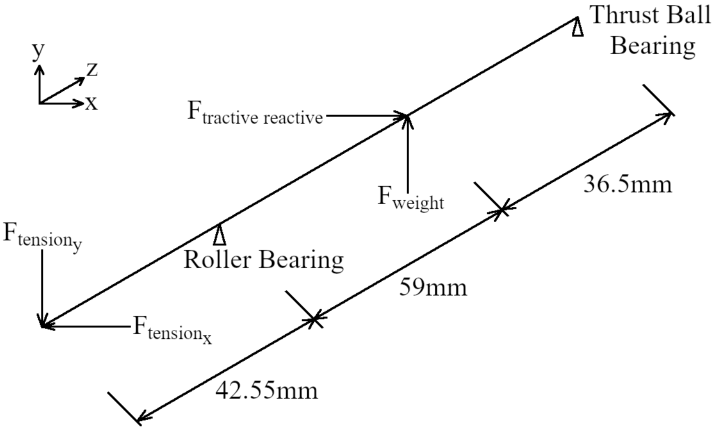

y
z
x
Thrust Ball Bearing
F_{tractive reactive}
F_{weight}
36.5mm
F_{tension_y}
Roller Bearing
59mm
F_{tension_x}
42.55mm

Figure 5.23: Wheel Axle Belt System Force Diagram

The forces are shown in figure 5.23, along with the RPM value of 166.389 $^{r}/_{min}$ was fed into the SKF bearing calculator[6]to select appropriate bearings. The chosen bearings were a NU 204 ECP bearing for the roller bearing shown in figure 5.23 and a 3205 A-2Z bearing for the thrust ball bearing shown in figure 5.23. The roller bearing is a free-floating bearing (meaning that it cannot take thrust forces); this allowed for play due to the shaft elongation and retraction during thermal cycles. Thus, if any thrust force occurs on the shaft, it will be compensated for by the thrust ball bearing, which can also handle radial loads despite the name. The SKF calculator determined that the NU 204 ECP roller bearing would have a lifespan of 16800 hours (L_{10mh}), while the 3205 A-2Z angular contact ball bearing will have a lifespan of 2×10^5 hours (L_{10mh}). The full results of SKF's bearing calculator can be found in appendix C.

5.3.5 Drive Unit: Steering System Detailed Analysis

The steering system within the drive unit is responsible for controlling the angle of the drive wheel along the z-axis (refer to figure 5.1 for axis system), which acts as the AGV's steering mechanism.

5.3.5.a Steering System Overview

The steering drive system consists of a motor side planetary gearbox and a belt drive. Feedback in this system is done using an endless potentiometer as an absolute encoder.

[6] The bearing selection tool can be found at "https://www.skfbearingselect.com/#/bearing-selection-start"

These systems are described in figure 5.24.

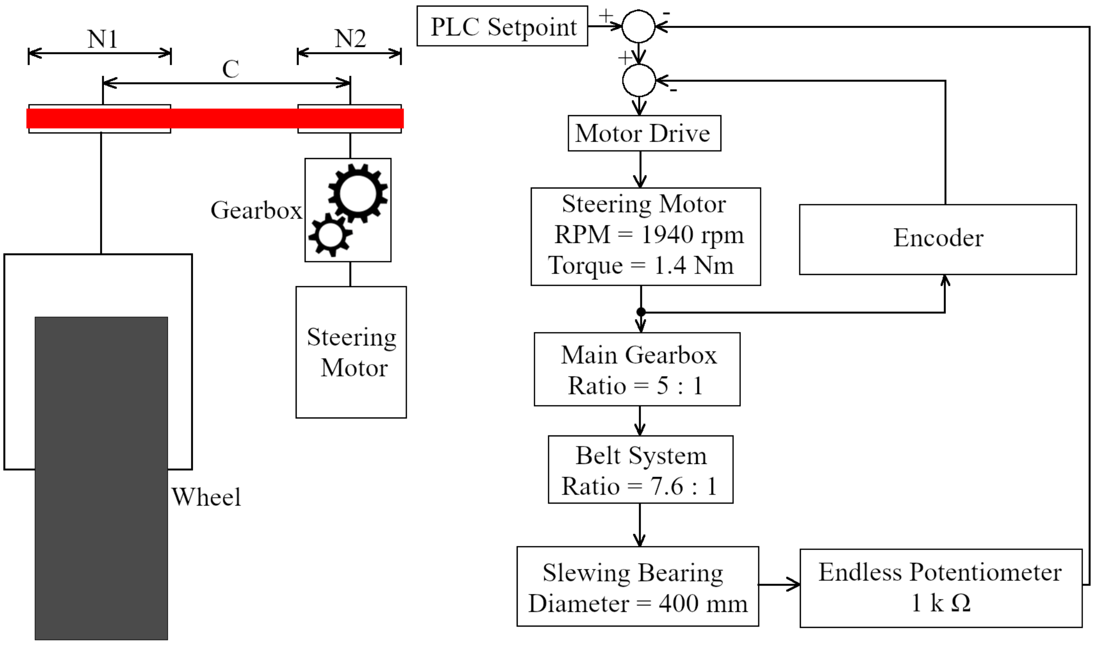


Figure 5.24: Steering System

There are two gear systems, namely the planetary gearbox and the belt gear ratio. The total gear ratio of this system will be:

$$\begin{aligned} i_{steering} &= (i_{planetary\ gearbox})(i_{belt}) \\ &= 5(7.6) = 38\ (1:38) \end{aligned} \tag{5.42}$$

5.3.5.b AGV Steering Torque Requirements

The torque on the steering assembly will be greatest when the wheels are not turning (i.e. the AGV is stationary), since when the wheels are not turning, but the steering is actuated, the wheels will scuff on the driving surface producing dynamic friction rather than rolling friction. Various parameters relating to the steering mechanism are summarised in table 5.5.

Table 5.5: Steering Global Requirements

Specification	**Value**
Net AGV Mass	600 kg
Castor Offset	45 mm
Dynamic Friction of Wheels (on concrete)	0.6
Number of Wheels	4
Number of Drive Units	2

Since the system has a castor offset of 45 mm for the AGV's wheel system, the force of friction the steering mechanism will have to overcome can be calculated as illustrated in equation 5.43.

$$\begin{aligned} F_f &= \mu \left(\frac{1}{4} mg \right) \\ &= 0.6 \left(\frac{1}{4}(600)9.8 \right) = 882\ N \end{aligned} \tag{5.43}$$

F_f	=	Force of friction on the wheel	N
μ	=	Dynamic friction co-efficient of polyurethane on concrete	
g	=	Gravitational constant	$^{m}/_{s^2}$
m	=	Mass of AGV	kg

Thus, using the diagram illustrated in figure 5.25 , the torque about the rotational axis of the steering mechanism (slewing bearing) can be found as determined in equation 5.44.

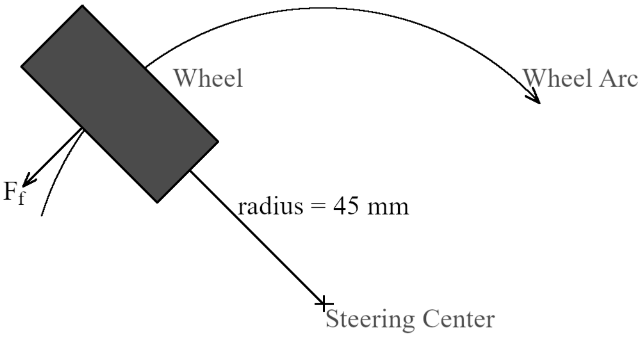


Figure 5.25: Steering Mechanism Force Diagram

$$\begin{aligned} T_{steering} &= (r_{castor\ offset})F_f \\ &= 45 \times 10^{-3}(882) = 39.69\ Nm \end{aligned} \tag{5.44}$$

$T_{steering}$	=	Torque require for steering	Nm
$r_{castor\ offset}$	=	Castor offset distance	m

5.3.5.c Steering Motor Validation

The motor chosen to actuate the steering mechanism was a Festo EMMS-ST-57-M-SEB-G2 stepper motor. A stepper motor was chosen for this application since it has high holding torque and is easy to implement in a system where a defined distance or arc needs to be traversed. (They have excellent open-loop control when compared to other motor types). The specifications associated with this motor are listed in table 5.6.

Table 5.6: EMMS-ST-57-M-SEB-G2 Steering Motor Specifications

Specification	**Value**
Nominal Torque	1.4 Nm
Nominal Speed	1940 $^{r}/_{min}$
Nominal Current Draw	5 A
Nominal Voltage	48V

5.3.5.d Steering Motor Gear train

Since the steering motor is connected directly to the planetary gearbox, the RPM of the drive pulley for the belt system will be five times slower than the motor, thus:

$$\begin{aligned} \omega_{drive\ pulley} &= \frac{\omega_{stepper}}{i_{planetary\ gearbox}} \\ &= \frac{1\ 940}{5} = 388\ {}^{r}\!/_{min} \end{aligned} \tag{5.45}$$

Since the belt system has a ratio of 7.6 : 1; the maximum speed of the slewing bearing, and thus the steering mechanism, will be:

$$\begin{aligned} \omega_{slewing\ bearing} &= \frac{\omega_{drive\ pulley}}{i_{belt\ system}} \\ &= \frac{388}{7.6} = 51.05\ {}^{r}\!/_{min} \end{aligned} \tag{5.46}$$

The torque out of the planetary gearbox, at the drive pulley of the belt system, can be calculated similarly to the speed (as done in equation 5.45). However, in this case, the in-efficiency of the gearbox must be taken into account. The result of this calculation can be found in equation 5.47.

$$\begin{aligned} \tau_{drive\ pulley} &= i_{planetary\ gearbox}(\tau_{stepper})(e_{planetary\ gearbox}) \\ &= 5(1.4)(0.98) = 6.86\ Nm \end{aligned} \tag{5.47}$$

The final torque for the steering system on the slewing bearing can be calculated from the result of equation 5.47 and is given in equation 5.48.

$$\begin{aligned} \tau_{slewing\ bearing} &= i_{belt\ system}(\tau_{drive\ pulley})(e_{belt\ system}) \\ &= 7.6(6.86)(0.98) = 51.09\ Nm \end{aligned} \tag{5.48}$$

The maximum torque that the steering mechanism can experience when scuffing the wheels is 39.69 Nm. Since the maximum possible torque that this system can develop is 51.09 Nm, the steering mechanism should not stall.

5.3.5.e Steering-Traction Coupling Effect

Since the steering mechanism and traction system share a common axis, there will be interference between the two systems. This common axis is illustrated in the block diagram shown in figure 5.26.

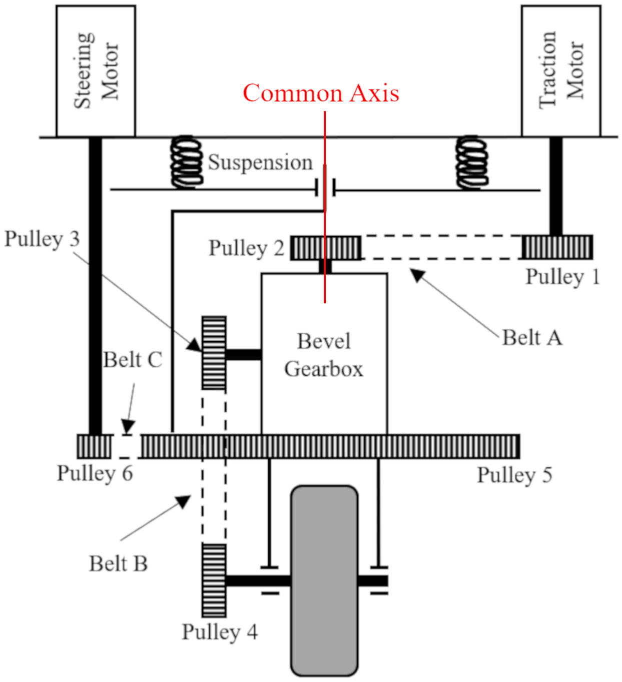


Figure 5.26: Block Diagram of Drive Unit

Due to the system's design, the steering mechanism can affect the traction system; however, the traction system cannot affect the steering system. Thus, only the traction system must compensate for motions in the steering mechanism. Since the coupling effect occurs before the main bevel gearbox, any compensations must be first reduced by the gear ratio. This reduction is explained in equation 5.49.

$$\omega_{parasitic} = \frac{\omega_{steering}}{i_{bevel\ gearbox} \times i_{traction\ belt2}} \tag{5.49}$$

$\omega_{parasitic}$ = Parasitic velocity as viewed at the wheel $^{r}/_{min}$

$\omega_{steering}$ = RPM of the slewing bearing of the steering system $^{r}/_{min}$

$i_{bevel\ gearbox}$ = Gear ratio of the traction system bevel gearbox

$i_{traction\ belt2}$ = Gear ratio of belt system 2 of the traction system (Belt B in figure 5.26)

With the reduction ratio calculated in equation 5.49, the parasitic compensation calculation that the control system will have to implement is:

$$\omega_{wheel\ adjusted} = \begin{cases} \omega_{wheel} + \omega_{parasitic} & \text{if} \quad \omega_{steering} \cdot \omega_{wheel} < 0 \\ \omega_{wheel} - \omega_{parasitic} & \text{if} \quad \omega_{steering} \cdot \omega_{wheel} > 0 \\ \omega_{wheel} & \text{if} \quad \omega_{steering} \cdot \omega_{wheel} = 0 \end{cases} \tag{5.50}$$

$\omega_{wheel\ adjusted}$	=	The adjusted RPM of the wheel	$^{r}/_{min}$
ω_{wheel}	=	The setpoint speed of the wheel before adjustment	$^{r}/_{min}$
$\omega_{parasitic}$	=	The parasitic velocity as developed by the steering mechanism	$^{r}/_{min}$

In equation 5.50, the parasitic velocity either is added or subtracted from the wheel's desired velocity in order to compensate. The addition or subtraction effect is determined using the product of the steering and traction angular velocities. If they are in the same direction, the product will be positive (> 0); thus, the parasitic velocity will attempt to add to the wheel velocity. An equal magnitude must be subtracted from the wheel's setpoint to achieve the correct final velocity. The opposite is true when the product is negative (< 0). The last case occurs when there is no steering motion, and as such, no compensation needs to occur.

To determine the traction motor RPM from the compensated wheel RPM value (equation 5.50), equation 5.51 can be used.

$$\omega_{traction\ motor\ adjusted} = \omega_{wheel\ adjusted}(i_{traction\ belt1} \times i_{bevel\ gearbox} \times i_{traction\ belt2}) \tag{5.51}$$

$i_{traction\ belt1}$ = Gear ratio of belt system 1 of the traction system (Belt A in figure 5.26)

5.3.6 Standard Parts Used for Drive Unit

Listed in the section that follows are some of the key components to be used in the drive unit. To identify components refer to figure 5.16 and figure 5.26.

Table 5.7: Key Drive Unit Components

Part Chosen	**Manufacturer**	**Component Description**
WPU 062	Ladder & Castor	Drive Wheel
3205 A-2Z	SKF	Wheel Axle Thrust Bearing
NU 204 ECP	SKF	Wheel Axle Roller Bearing
KLK 400L	JOST	Slew Bearing
S-1FL6 1FL6042-2AF21-1AH1	Siemens Simotics	Traction System Motor
PHG 624-8M-30	SKF	Belt B
PHP 36-8M-30TB	SKF	Belt B Drive Pulley
PHF TB1615X20MM	SKF	Belt B Drive Pulley Core
PHP 36-8M-30TB	SKF	Belt B Driven Pulley
PHF TB1615X25MM	SKF	Belt B Driven Pulley Core
HK 1616 2RS	SKF	Belt B Idler Pulley Bearing
R02 02 B3 18.03 AS 25X45 LH H1	Varvel	Traction System Gearbox
PHG 580-5M-15	SKF	Belt A
PHP 60-5M-15-RSB	SKF	Belt A Drive Pulley
PHP 60-5M-15-RSB	SKF	Belt A Driven Pulley
61900-2Z	SKF	Belt A Idler Pulley Bearing
EMMS-ST-57-M-SEB-G2	Festo	Steering System Motor
EMGA-60-P-G5-SST-57	Festo	Steering System Gearbox
PHG 1280-8M-20	SKF	Belt C
PHP 45-8M-20-RSB	SKF	Belt C Drive Pulley
KBF 20	MCL	Suspension Linear Bearings
GT2 60T W6mm	Banggood	Encoder Driven Pulley 6mm
GT20 60T B8mm W6mm	Banggood	Encoder Drive Pulley 6mm
8mm to 6.35mm adapter	Banggood	Encoder to Pulley Adapter
GT2 W6 L300	Banggood	Encoder Belt
FYC 506	SKF	Compromiser Bearing Flange
YAR 206-2F	SKF	Compromiser Bearing
14mm Spline Nut	MCL	14mm Spline Nut
14mm Spline Rod	MCL	14mm Spline Shaft
1kΩ Endless Potentiometer	Unknown	Encoder Potentiometer

5.4 Castor Unit Design

There are two unpowered castor units on the AGV (see figure 5.5). These castors solely provide support for the AGV's weight and are not for steering or traction.

5.4.1 Castor Unit Conceptual Designs

Three conceptual designs for the castor unit will be discussed in the paragraphs to follow:

5.4.1.a Castor Unit Design 1

Design 1 is based on the first design of the drive units. It is essentially the drive unit in figure 5.9 with the motors and slip-rings removed. The system did not progress past the initial idea phase due to the corresponding drive unit being rejected.

5.4.1.b Castor Unit Design 2

This castor unit corresponded to the design of the drive unit shown in figure 5.12. It was based on the idea of a cantilever suspension system. This system is illustrated in figure 5.27.

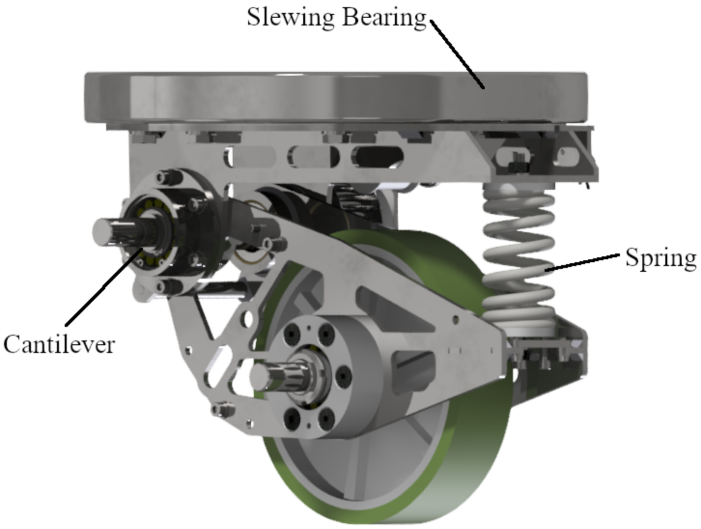


Figure 5.27: Castor Unit Concept 2 - Cantilever Suspension System

This system has the advantage of using a cantilever suspension system which is more cost-effective than the design chosen in concept 3.

5.4.1.c Castor Unit Design 3

Concept 3, the chosen concept, is based on an in-line suspension system. This design is the same design type used in the third concept design for the drive unit. The design also used a pre-built castor wheel unit from Ladder and Castor to simplify the design. This system is illustrated in figure 5.28.

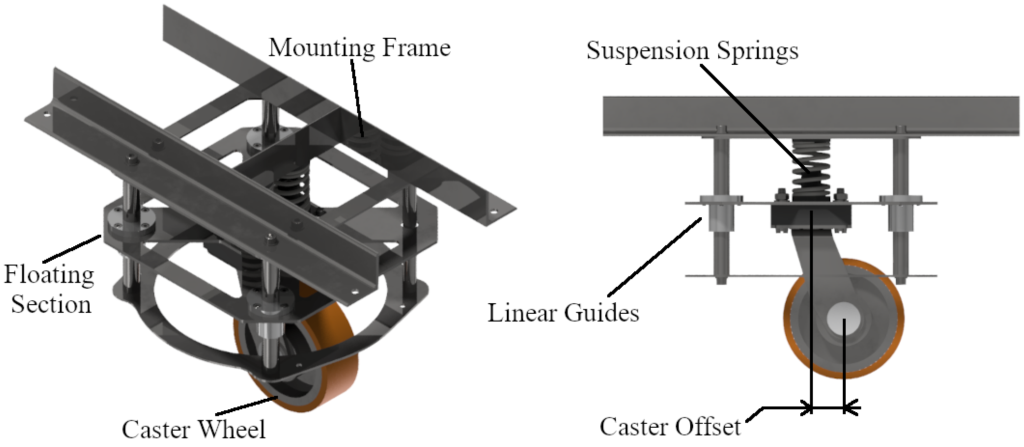


Figure 5.28: Castor Unit Concept 3 - In-Line Suspension System

The Castor unit in concept 3 makes use of two springs in the centre of the design, unlike concept 3 of the drive units (see figure 5.16). The use of two springs was done to reduce the structure needed to support the floating section of the castor unit by concentrating the load directly above the castor wheel. More reinforcing would be required if the springs were arranged around the edge of the floating section, which would unnecessarily increase the system's total weight. The drawback of doing this was that each spring in the castor unit would have to have a "K" factor or spring constant double that of the concept 3 drive unit to maintain a balanced system.

5.4.2 Castor Final Design

The final design of the castor units for the AGV is discussed in the following section.

5.4.2.a Castor Unit Selection Matrix

The chosen final design for the castor unit was heavily influenced by the chosen drive unit design. The advantages and disadvantages, however, are stilled listed in the cost-benefit matrix 5.8.

Table 5.8: Comparison of Castor Unit Concepts

Concept	Advantages	Disadvantages
Concept 1	· Extremely simple · Comparably inexpensive to build · Constant Castor Offset	· Paired with drive unit 1 · No suspension
Concept 2	· Symmetrical weight distribution · Integrated suspension system · Suspension system has good camber conformity	· Variable castor offset · Exotic springs needed · Extremely mechanically complex · Trailing arm length restrictions
Concept 3	· Symmetrical weight distribution · Constant Castor Offset · Integrated suspension system · Mechanically simpler than concept 2 · Uses off-the-shelf castor	· No camber conformity · Linear guides of suspension are a weak point

Since concept 1 for the drive unit was disqualified, so was the castor unit concept 1. As concept 1 was disqualified, only the cantilever suspension style of concept 2 and the in-line suspension style of concept 3 remained. Concept 2 has some disadvantages when compared to concept 3. The first and most prevalent disadvantage is the variable castor offset introduced by the suspension system. It also had the disadvantage of using an exotic spring for the suspension system that could not readily be found on the market. Finally, it was more expensive and complex to build than concept 3, as concept 3 used an off-the-shelf castor wheel solution. Concept 2 did have one advantage when compared to concept 3 as it had better camber conformity (see 5.5.1.b), though this advantage would be lost when using drive unit concept 3 with castor unit concept 2. Thus, concept 3 would be used for the final concept for the AGV's castor unit.

5.4.2.b Completion of the Castor Unit CAD Design

Since the castor unit is so mechanically simple, minimal "fine-tuning" needed to be done after selection. What little was done is documented in the sections that follow.

Modularity Skeleton

Like the drive unit, the castor unit had to be modular. This modularity allows it to be easily removed from the main AGV for replacement or maintenance. A mounting framework was designed to allow interfacing from the main AGV to the castor unit. This is illustrated in figure 5.29.

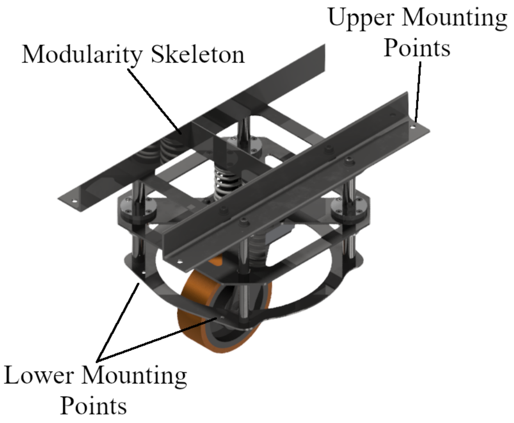


Figure 5.29: Castor Unit Modularity Frame

Cost Optimisation

Only three linear bearings were used to decrease the cost of this design. The corner without a linear guide was labelled in figure 5.30 as "No Linear Guide".

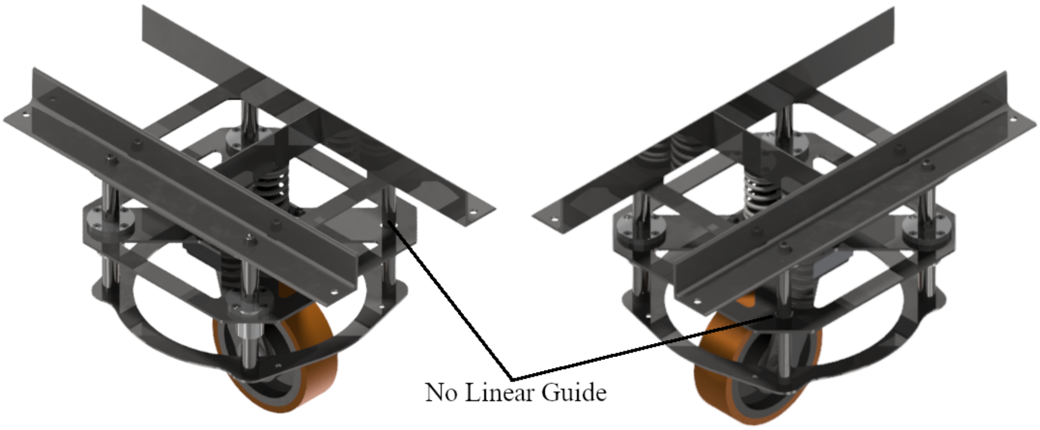


Figure 5.30: Castor Unit Cost Optimisation

Castor Wheel

As a prebuilt castor wheel was used for the castor unit, the castor offset of this unit determined the castor offset of the drive unit and was 45 *mm*. It was purchased from Ladder and Castor, designated TS 6 PUBM. The specifications for the prebuilt castor wheel are listed in table 5.9.

Table 5.9: TS 6 PUBM Castor Wheel Specifications

Specification	**Value**
Wheel Size	$150 \times 50\ mm$
Castor Height	$190\ mm$
Castor Offset	$45\ mm$
Load Rating	$300\ kg$

Like the wheel used for the drive unit, the castor wheel is made of a cast polyurethane tyre wrapped around a cast iron core. This material creates minimal noise, has low rolling resistance, does not damage the floor and is highly resistant to cuts and tearing [53].

Final Design

The final design for the castor unit is shown in figure 5.31.

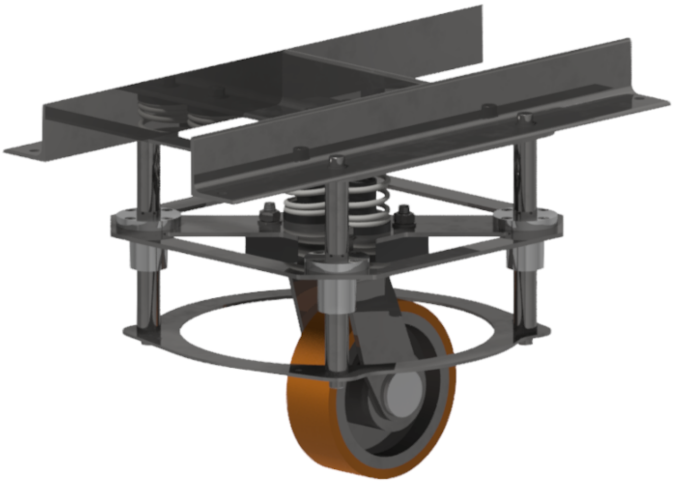


Figure 5.31: Finalised CAD Model of the AGV Castor Unit

5.4.3 Standard Parts Used for Castor Unit

Table 5.10 lists all the standard parts relevant to the castor unit.

Table 5.10: Key Castor Unit Components

Part Chosen	**Manufacturer**	**Component Description**
TS 6 PUBM	Ladder & Castor	Castor Wheel
KBF 20	MCL	Linear Ball Bearings

5.5 Suspension System

The type of suspension system for the AGV was chosen based on the drive and castor unit selection design. In this case, it was an inline type suspension system. The operation of this system is outlined in section 5.5.1.d.

5.5.1 Suspension System Theory

This section contains the theory needed to comprehend the suspension system design.

5.5.1.a Suspension System Definition

A suspension is any system that isolates a given body, in motion, from the surface on which it is driving. Suspension systems are used to limit the transmission of vibrations from the driving surface to the body of the vehicle [55]. All suspension systems make use of the spring-spring dampener effect, where the spring "compensates" for the unevenness of the driving surface by either extending or contracting to keep the vehicle's body moving in a fixed horizontal plane while the dampener dissipates energy that builds up in the spring through the "compensation" process. If a dampening factor is not included, the system will, theoretically, oscillate indefinitely. The spring and dampener in a suspension system do not necessarily have to consist of an actual physical spring and dampener; many strategies employ other methods such as fluids, magnetics or electrical fields. However, in each of these systems, components exist that mimic the actions of a spring and a dampener [56].

Some more straightforward suspension systems consist solely of a physical spring, where the dampening effect is produced by introducing sufficient friction to the system. High friction, damper-less systems such as this were used in the AGV.

5.5.1.b Terms used in Suspension Systems

Listed in the following sections are some standard terms relating to suspensions systems. These terms will be used later in this text.

Sprung and Unsprung Mass

The sprung mass refers to the vehicle's body and is isolated from the vibrations of the driven surface via the suspension system. The un-sprung mass consists of components not isolated by the suspension system, such as the wheels and axles.

Body Roll

Body roll refers to the rotation of the AGV about the x-axis and is a result of the vehicle's inertia when changing direction. Convention dictates that roll in the clockwise direction when viewed from the rear of the vehicle is positive and anti-clockwise in negative [57].

Camber Angle

Camber angle refers to the angle between the plane normal to the wheels axis of rotation and the plane normal to the driving surface [33]. This concept is illustrated in figure 5.32.

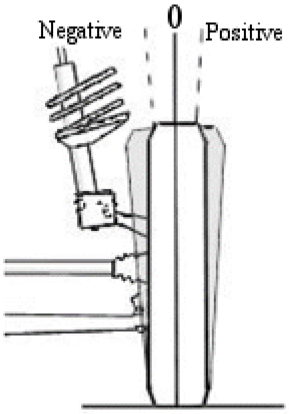


Figure 5.32: Camber Angle[7]

As illustrated in figure 5.32, when the top of the wheel slopes inwards, it is referred to as a negative camber angle. When the top of the wheel slopes outwards, it is known

[7] Image adapted from Collins et al. [55].

as a positive camber angle.

Camber Control & Camber Conformity

Camber control refers to how the camber of the wheel changes (with reference to the horizontal plane) as the suspension system is actuated. Camber conformity refers to how well a suspensions system can keep the wheel parallel to an uneven driving surface, which can vary in the x, y and z dimensions [33].

Toe-In/ Toe-Out

A vehicle's toe-in or toe-out refers to the direction the wheels, sharing a common axis of rotation (x-axis), are pointing when viewed from above (along the y-axis). An example of a toe-in wheel setup is shown in figure 5.33.

Toe-out would be the opposite of what is illustrated in figure 5.33. The leading edge of the tyres would be splayed outwards, and the trialling edge would point inwards.

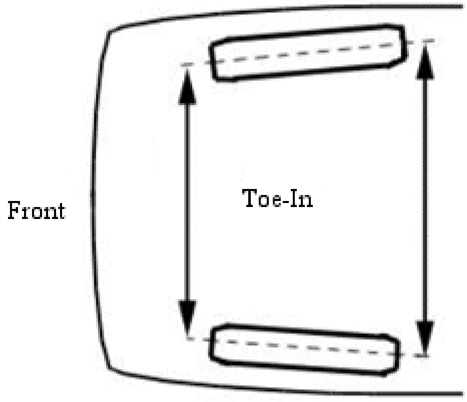


Figure 5.33: Toe-In[8]

Ideally, for the swerve drive envisioned for use on the AGV described in this report, a toe-in/ toe-out angle of zero is required, i.e. the wheels are parallel when sharing a common virtual axis.

5.5.1.c The Suspension Model

Suspensions systems are generally designed using one of two philosophies. These are the "half car model" and "quarter car model". The half-car model is relevant when

[8] Image adapted from Collins et al. [55].

two wheels on a vehicle share a suspension system, known as a dependent suspension system. These systems are standard on many motor vehicle types that use leaf springs on their rear axle. As each wheel of the AGV in this thesis works independently, a dependent suspension system would not suffice. Thus only independent suspension strategies, where the quarter car model is used, will be considered in this research.

The quarter-car model is a mathematical strategy used to determine the suspension system's behaviour by reducing the suspension system into a spring and dampener model that can be described mathematically. A graphical representation of this idea is illustrated in figure 5.34A.

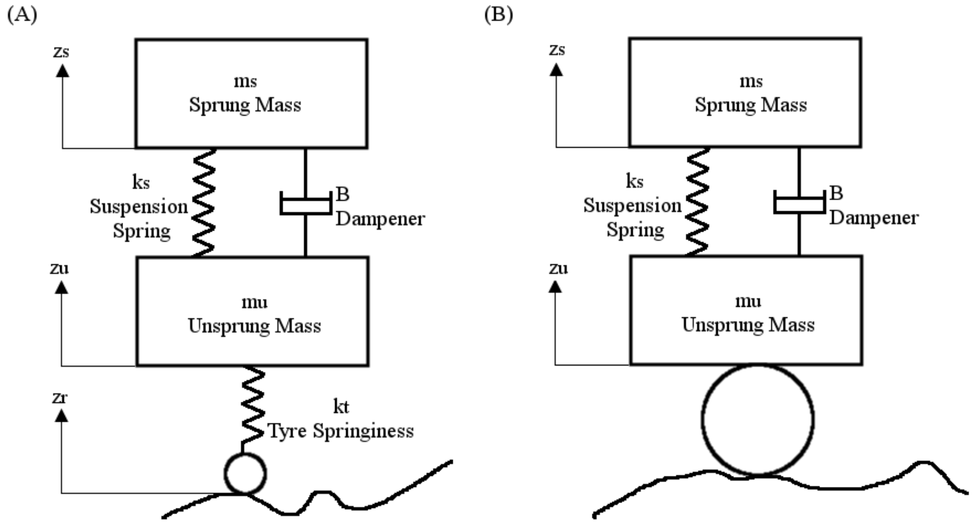


Figure 5.34: "Quater Car" Suspension Representation

Since the wheels used on the AGV are hard plastic, the inclusion of a tyre springiness factor (k_t) is not needed. The model is therefore simplified to the one illustrated in figure 5.34B.

The system behaviour shown in 5.34B can be described mathematically as illustrated in equation 5.52.

$$m_s z_s'' = -B\left(z_s' - z_u'\right) - k_s\left(z_s - z_u\right) \tag{5.52}$$

m_s	=	sprung mass	kg
B	=	viscous friction coefficient	$N\cdot s/m$
k_s	=	spring constant	N/m
z_s	=	sprung mass displacement	m
z_u	=	unsprung mass displacement	m

Equation 5.52 can be re-written as shown in equation 5.53:

$$m_s z_s'' + B z_s' + k_s z_s = B z_u' + k_s z_u \tag{5.53}$$

If z is defined as $z = z_s - z_u$ then:

$$m_s \left(z'' + z_u'' \right) + k_s z + B z' = 0 \tag{5.54}$$

Thus:

$$\begin{aligned} z'' + \left(\frac{B}{m_s} \right) z' + \left(\frac{k_s}{m_s} \right) z &= -z_u'' \\ &= -a_u \end{aligned} \tag{5.55}$$

a_u	=	generalised unsprung mass acceleration	m/s^2

If the co-efficients of equation 5.55 are non-dimensionalized, equation 5.56 results:

$$\begin{aligned} z'' + 2\zeta\omega_n z' + \omega_n^2 z &= -z_u'' \\ &= -a_u \end{aligned} \tag{5.56}$$

ζ	=	dampening of the system	*unitless*
ω_n	=	natural frequency of the system	rad/s

Taking the Laplace transform of equation 5.56 yields equation 5.57:

$$\frac{Z(s)}{A(s)} = \frac{-1}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{5.57}$$

$A(s)$ = acceleration of the unsprung mass $^m/_{s^2}$

Equation 5.57 can be used to predict the acceleration and thus the vibrations of the sprung mass given a known driving surface profile.

5.5.1.d Suspension Geometries

Suspension system geometry refers to the physical layout of the spring dampener system. There are several commonly used suspension system geometries; a brief description of each will be discussed in this section.

Swing Axle Suspension

The wheels in the swing axle suspension system rotate about the z-axis of the vehicle. The major drawback of this system is that camber is introduced when the system actuates. The layout of this system is illustrated in figure 5.35.

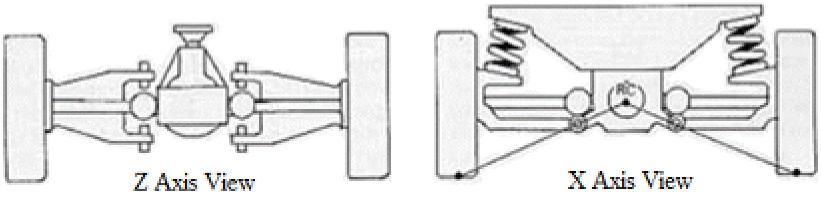


Figure 5.35: Swing Axle Suspension System

Trailing Arm Suspension

Figure 5.36 illustrates the trailing suspension system. It is similar to the swing axle suspension system in many ways, except it actuates the vehicle's x-axis; this eliminates many of the camber problems associated with the swing axle suspension system.

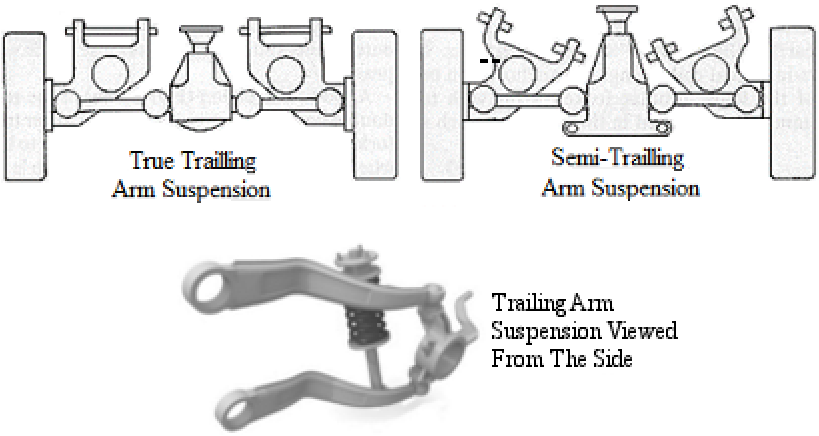


Figure 5.36: Trailing Arm Suspension System[9]

MacPherson Strut Suspension

The MacPherson strut suspension consists of an in-line spring and dampener, often connected to the body of the vehicle with a single base plate, see figure 5.37.

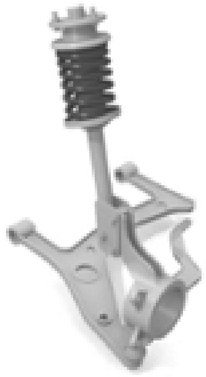


Figure 5.37: MacPherson Strut Suspension System

Inline Suspension

The inline suspension is not a suspension system readily found on any vehicle that travels above 10 $^{km}/_{h}$; this is due to its poor adhesion to the driving surface's camber. However, it is often found on large, slow-moving systems due to its relative simplicity and robustness. An example of such a system is illustrated in figure 5.38.

[9] Image adapted from Collins et al. [55].

[9] Image adapted from Collins et al. [55].

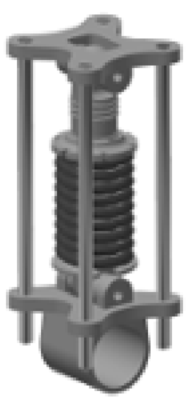


Figure 5.38: In Line Suspension System

Double Wishbone Suspension

The double-wishbone suspension system consists of two "A-frames" stacked above each other and connected to the wheel via a set of ball joints. The spring and dampener are found between the two A-frames; this is illustrated in figure 5.39.

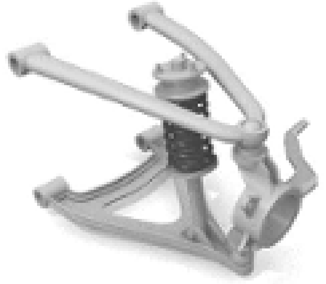


Figure 5.39: Double Wishbone Suspension System[10]

Multi-link Suspension

The multi-link suspension system is similar to the double-wishbone, except that the struts of the "A-frames" are separate items that can move independently. This suspension system is illustrated in figure 5.40.

[10] Image adapted from Collins et al. [55].

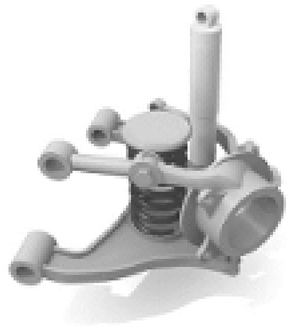


Figure 5.40: Multi-link Suspension System[11]

5.5.1.e Comparison of Suspension Systems

Table 5.11 gives a brief comparison of the different suspension systems previously listed, noting their perceived advantages and disadvantages.

[11] Image adapted from Collins et al. [55].

Table 5.11: Comparison of Suspension Systems[12]

Suspension System	Advantages	Disadvantages
Swing Axle	Simple to assemble Inexpensive takes lateral space only	Camber control variation Poor camber conformity
Pure Trailing Arm	Simple to assemble Good camber control Limited camber variation	Takes space longitudinally Poor camber conformity
Semi-Trailing Arm	Simple to assemble Decent camber conformity	Takes space longitudinally Camber control variation
MacPherson Strut	Simple to assemble Light weight Small size laterally Small size longitudinally Good camber conformity	Large vertical size
In-Line	Very Simple to assemble Light weight Good camber control	No camber conformity
Double Wishbone	Simple to assemble	Complex design work Very large
Multi-link	Excellent camber control Excellent camber conformity	Complex assembly Very complex designing Costly

5.5.2 Forces on the Suspension System

Thus a mathematical model for this system can be developed, as illustrated in the sections that follow, with the aid of figure 5.41

[12] Data extrapolated from Dixion [57], Collins [55] and Macfarlane [33]

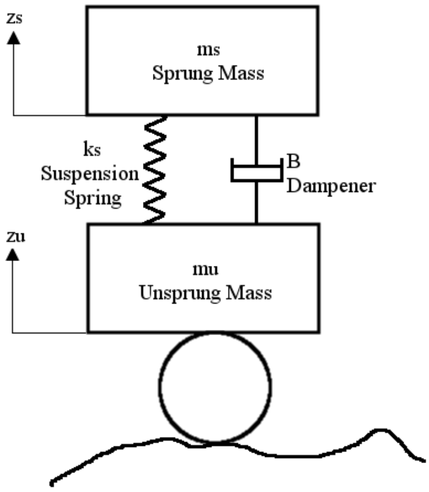


Figure 5.41: Inline Spring Dampener Model

Forces on the spring dampener system include weight, dampener reactionary forces and spring reaction forces. These forces are calculated in the sections to follow.

Force 1: Inertia of The Sprung Mass

The inertia of the AGV's body will produce a force that resists a change in motion. The resulting force of this inertia is listed in equation 5.58.

$$F_1 = m_s z_s'' \tag{5.58}$$

m_s	=	Sprung mass of the AGV	kg
z_s''	=	Acceleration of the sprung mass	$^{m}/_{s^2}$

Force 2: Dampening Force

No active dampener was included in the suspension system, and as such, the dampening is created through friction between the linear rails that form part of the suspension system. When designing this system, the exclusion of the dampener was based on the author's experience. Thus, if the dampening of the system proved insufficient, using just viscous friction, a 100N "gaslift" cylinder would be used to add appropriate dampening. After construction, the allocation for the "gaslift" cylinder proved unnecessary as the viscous friction of the linear rails provided sufficient dampening. The equation for viscous frictional dampening is given in equation 5.59.

$$F_2 = B(z_s' - z_u') \tag{5.59}$$

B	=	Dampener's dampening co-efficient	$^{N\cdot s}/_{m}$
z_s'	=	Velocity of the sprung mass	$^{m}/_{s^2}$
z_u'	=	Velocity of the unsprung mass	$^{m}/_{s^2}$

B can be experimentally determined from the linear rods and is a constant value.

Force 3: Force of the Spring

The force generated by the spring/s in the suspension system is calculated using equation 5.60.

$$F_3 = k(z_s - z_u) \tag{5.60}$$

k	=	Spring Constant	$^{N}/_{m}$
z_s	=	Displacement of the sprung mass	m
z_u	=	Displacement of the unsprung mass	m

5.5.3 Transfer Equation of The Suspension System

The transfer equation of the suspension system can be calculated utilising the maths developed in section 5.5.1.c. The first step involves calcualting the sum of forces as illustrated in equation 5.61.

$$\begin{gathered} F_1 + F_2 + F_3 = 0 \\ \implies m_s z_s'' + B(z_s' - z_u') + k(z_s - z_u) = 0 \end{gathered} \tag{5.61}$$

If $a_{wheel} = z_u$, $z = z_s - z_u$ and $z_s'' = z'' + z_u''$, then equation 5.59 becomes equation 5.62. In this system a_{wheel} is the acceleration of the wheel.

$$m_s z'' + Bz' + kz = -m_s a_{wheel} \tag{5.62}$$

Taking the Laplace of equation 5.62 will yield:

$$\begin{aligned} \mathscr{L}\{(m_s z'' + Bz' + kz)\} &= \mathscr{L}\{(-m_s a_{wheel})\} \\ \implies Z\left[m_s s^2 + Bs + k\right] &= -m_s A_{wheel} \end{aligned} \tag{5.63}$$

Thus equation 5.63 reordered gives:

$$Z = -m_s \left[\frac{1}{m_s s^2 + Bs + k}\right] A_{wheel} \tag{5.64}$$

5.5.4 Simulation Values and Suspension Simulation

Designing a suspension system for the AGV involves first creating a simulation. This simulation can be used to determine the system's reaction to various driving surfaces. For this system, a "worst-case" bump is used to determine the best parameters for the spring. The dimensions of this bump are illustrated in figure 5.42.

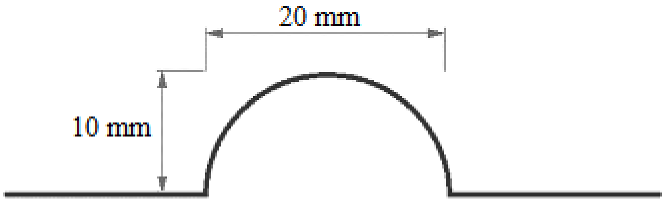


Figure 5.42: Bump Used in Simulation Test

The bump shown in figure 5.42 is consistent with small obstructions on the floor, such as cables or bolts, which makes this an ideal test for an AGV that is to operate in an industrial environment. In order to test the system using Matlab, a close approximation to this bump needed to be produced using an array of z-axis height values vs time. Creating this approximation was done using a URP function (described in appendix D) which calculated the bump height vs time using the velocity of the AGV. The height array vs time is graphically illustrated in figure 5.43.

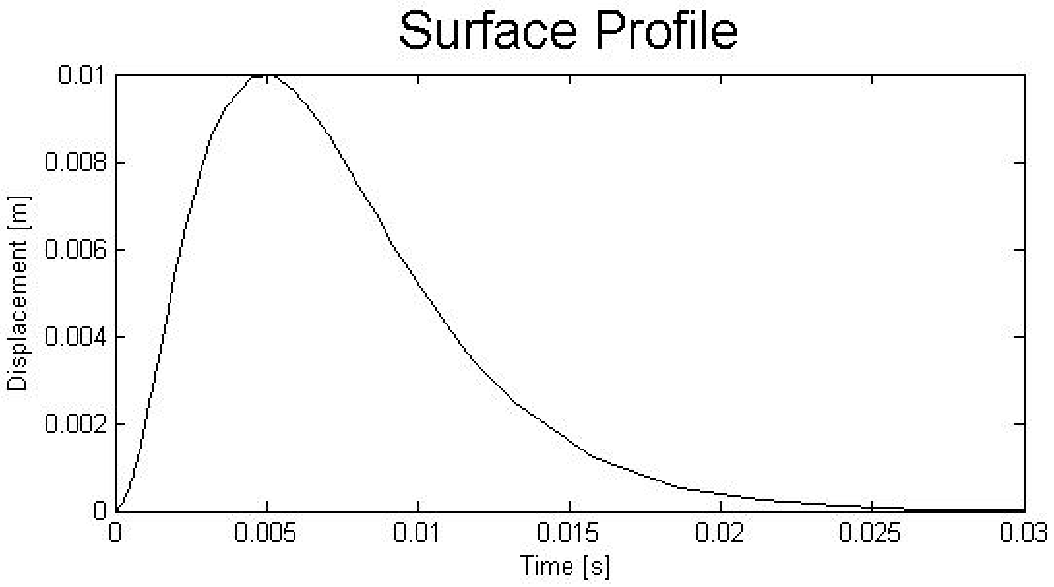


Figure 5.43: URP Function of Bump Height Vs Time

The quarter AGV weight and AGV speed set for this test are listed in table 5.12.

Table 5.12: AGV Simulation Parameters

Parameter	**Value**
Quarter AGV Sprung Mass	150 kg
AGV Speed	1.3 $^{m}/_{s}$

5.5.5 Matlab Model

The Matlab and Simulink model for the suspension system can be found in appendix D. This model, along with the parameters in table 5.13 were used to validate the feasibility of the suspension system. The values contained in table 5.13 were arrived at through an iterative design process, where a desired set of values were chosen and then compared to available mechanical components.

Table 5.13: Suspension System Component Specifications

Parameter	**Value**
Spring Constant	53 848 $^{N}/_{m}$
Dampening Co-efficient	220 $^{N\cdot s}/_{m}$
Sprung Mass (Quater AGV Weight)	150 kg

The parameters for the various components above yielded the following simulated suspension results. The suspension system displacement reaction to the bump described in section 5.5.4 is illustrated in figure 5.44.

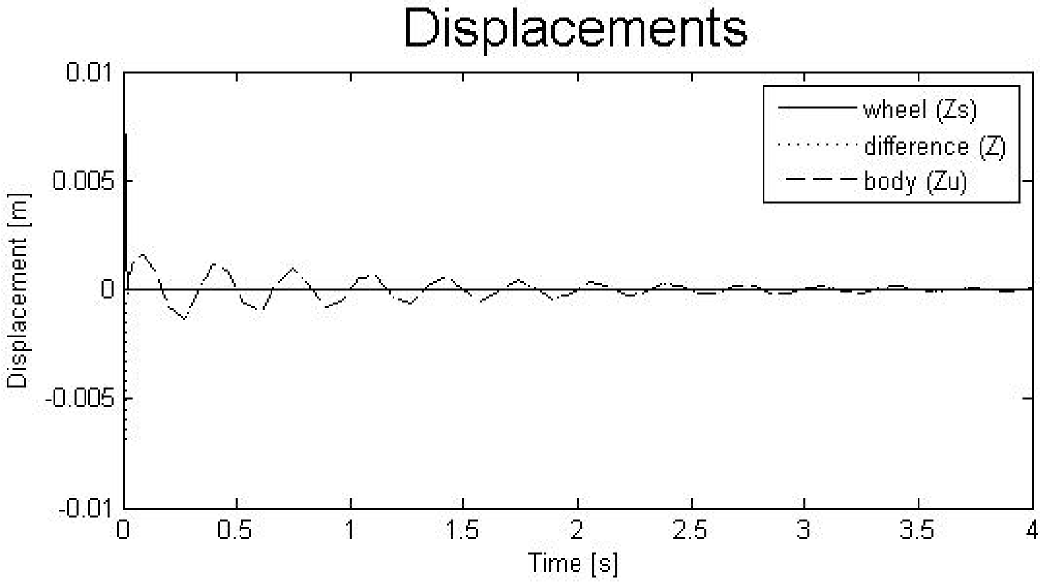


Figure 5.44: Displacement Results of Suspension System

From figure 5.44, it can be determined that it takes the suspension system approximately 3.5 seconds to fully settle after hitting a 10 mm high bump at 1.3 $^{m}/_{s}$ when fully loaded (600 kg or 150 kg quarter weight). It can be stated that after 2 seconds, the oscillations fall below 0.5 mm thus are unlikely to be felt. A close-up of the first 0.1 seconds of the displacement reaction is illustrated in figure 5.45.

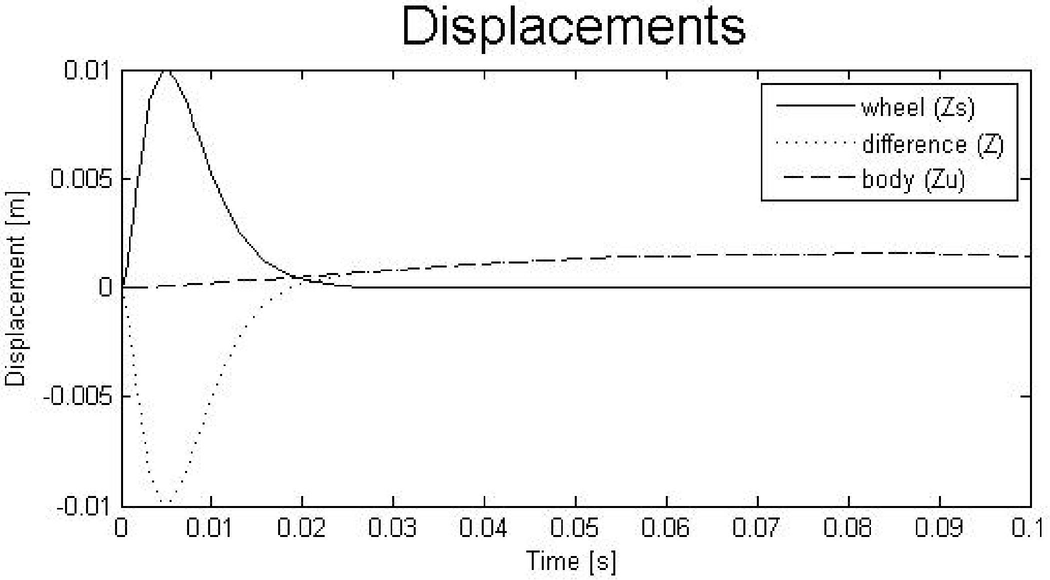


Figure 5.45: Zoomed Displacement Results of Suspension System

In figure 5.45, the displacement of the wheel follows the displacement of the 10 *mm* obstacle. It peaks at 10 *mm* as well. The difference between the wheel and body (Z) mirrors the wheel's displacement as it compensates, attempting to keep the body from moving. The velocity reaction of the system is illustrated in figure 5.46.

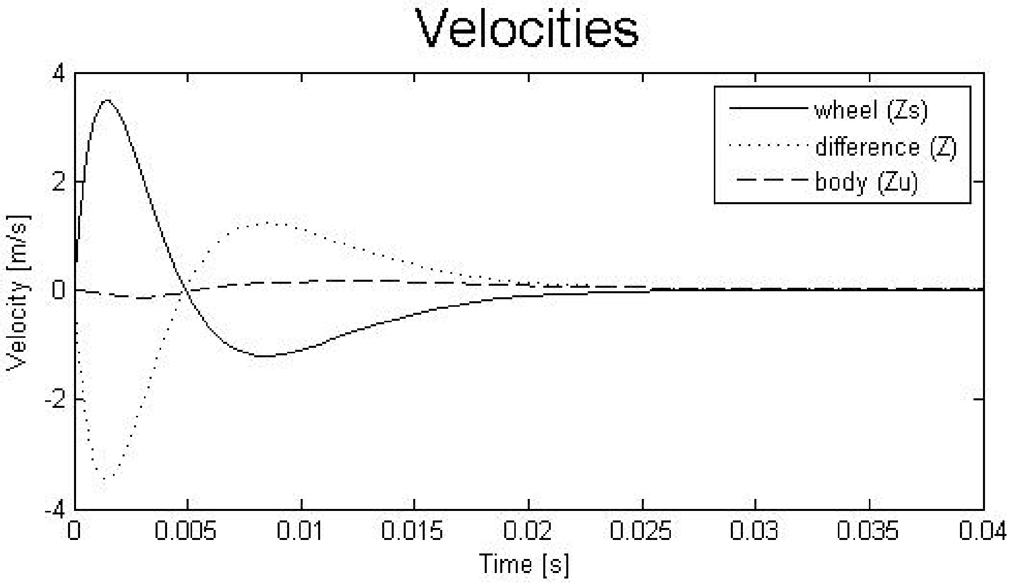


Figure 5.46: Velocity Results of Suspension System

With the parameters selected in table 5.13 a workable suspension system was created. This solution is illustrated in the results shown in figure 5.46, where the wheels of the AGV peak at $\sim 3.48\ ^{m}/_{s}$ while the body of the AGV has a velocity peak of only $\sim 0.125\ ^{m}/_{s}$. Thus, the difference in the motion of the wheels and body (Z) is an almost perfect mirror of the wheel's velocity as it almost entirely cancels it. The accelerations felt by the various components of the suspension system are illustrated in figure 5.47.

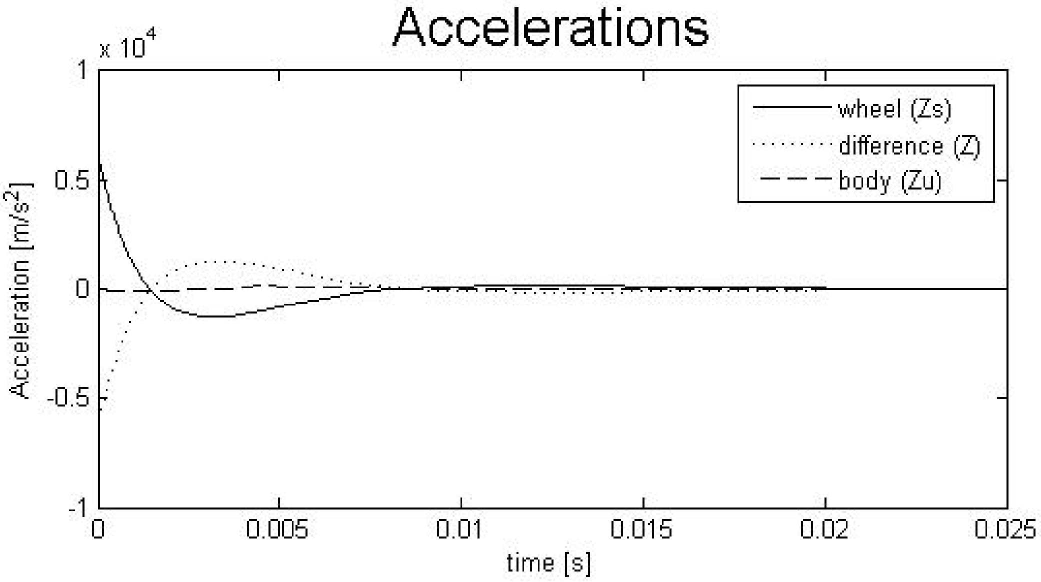


Figure 5.47: Acceleration Results of Suspension System

Accelerations are the primary concern for suspension systems, as a vehicle's body can have a significant change in velocity, but if the acceleration is low, it will not be "felt" by the components on the sprung mass of the AGV. Conversely, a small Δvelocity with large accelerations can be detrimental, especially if the frequency of the displacement oscillations is high. These high frequencies will lead to "vibrations" in the body of the AGV, which could potentially shake loose parts and fatigue wear components.

In figure 5.47 it can be seen that the AGV's body will experience a peak acceleration of $\sim 105\ ^{m}/_{s}$. The peak acceleration was reduced by the suspension system from a peak of $\sim 6000\ ^{m}/_{s}$. This attenuation is confirmed by the difference (Z) acceleration (difference in acceleration between the body and wheel), an almost perfect mirror image of the wheel acceleration.

For the peak oscillation of the AGV's body, the kinetic energy transferred from the sprung to unsprung mass can be calculated using equation 5.65 from Serway [46].

$$\begin{gathered} K = {}^{1}/_{2}mv^2 \\ K = {}^{1}/_{2}(600)(0.125)^2 = 15.63 \times 10^{-3} J \end{gathered} \tag{5.65}$$

According to the work-kinetic energy theorem:

"When work is done on a system and the only change in the system is in its speed, the net work done on the system equals the change in kinetic energy of the system"

Thus the total work that the oscillation described in equation 5.65 would produce is also $15.63 \times 10^{-3} J$.

5.5.6 Drive Unit Suspension

In section 5.5.5, the desired spring constant for this AGV was determined incrementally. In section 5.3.3, the drive unit was designed around a four spring configuration. Using four springs means that the spring constant determined in section 5.5.5 must be divided between the four springs used in the actual design; this is done according to equation 5.66 [58].

$$k_{total} = \sum_{i=1}^{n} k_n \tag{5.66}$$

k_{total}	=	Spring constant of springs in parallel	$^N/_m$
k_i	=	Spring Constant of the i'th Spring	$^N/_m$
n	=	Total number of springs in parallel	

It is important to note that equation 5.66 is only valid for springs in parallel; springs in series require a different formula.

Using equation 5.66, if four identical springs are used in the drive unit suspension system, then the spring constant of each spring will be 13 462 $^N/_m$ as calculated in equation 5.67.

$$\begin{aligned} k_{actual\ springs} &= \frac{k_{total}}{4} \\ \implies k_{actual\ springs} &= \frac{53\ 848}{4} = 13\ 462\ ^N/_m \end{aligned} \tag{5.67}$$

$k_{actual\ springs}$	=	Spring constant of the individual springs	$^N/_m$

Using the value calculated in equation 5.67 and the spring design tool in AutoDesk Inventor, a spring was designed that fulfilled the geometry and behavioural requirements of the suspension system. The full results of the spring design tool can be found in appendix E. Some notable values for this spring are listed in table 5.14.

Table 5.14: Drive Unit Spring Specifications

Specification	**Value**
Spring Constant	13 462 $^{N}/_{m}$
Working Load Length	66.175 mm
Working Stroke	$\sim$ 43 mm
Wire Diameter	5 mm
Spring Outer Diameter	48 mm
Number of Active Coils	5

5.5.7 Castor Unit Suspension

In much the same way as with the drive unit suspension system, the spring constant (k) of the castor unit's suspension system springs was determined by dividing the total desired spring constant by the number of parallel springs in the suspension system. In the case of the castor unit, there were two springs in parallel; thus, equation 5.67 is adapted as shown in equation 5.68.

$$\begin{aligned} k_{actual\ springs} &= \frac{k_{total}}{2} \\ \implies k_{actual\ springs} &= \frac{53\ 848}{2} = 26\ 924\ ^{N}/_{m} \end{aligned} \tag{5.68}$$

Using the value determined in equation 5.68 of 26 924 $^{N}/_{m}$ and Autodesk Inventor's Spring Design Tool, an appropriate spring was created for use in the suspension system. The detailed results of the spring tool can be found in appendix F. A summary of notable specifications are listed in table 5.15.

Table 5.15: Castor Unit Spring Specifications

Specification	**Value**
Spring Constant	34 329 $^{N}/_{m}$
Working Load Length	66.522 mm
Working Stroke	$\sim$ 17.478 mm
Wire Diameter	6 mm
Spring Outer Diameter	48 mm
Number of Active Coils	5

It is important to note that the castor unit springs are not exactly the desired 26 924 $^{N}/_{m}$, but rather 34 329 $^{N}/_{m}$. This deviation resulted from the manufacturability of the springs since the spring's dimensions were matched to drive unit spring dimensions. The matched dimensions included the working length and outer diameter. Values that could be massaged included the spring material type, number of active coils and wire diameter. However, it is to be noted that the wire diameter is restricted to a set of standard cross-sections (i.e. a diameter of 5.862 mm would not be possible to manufacture, even if it would give a value closer to the desired spring constant).

5.6 Body Design

This section deals with the construction of the body of the AGV. Standardised steel "Structural Hollow Section" would be used for the research project. Specifically, steel tube that adheres to ASTPM's Grade 355 tube standard [59]. This material was chosen due to its low cost and ease of manufacturability, being cheaper than standardised aluminium profiles of equivalent dimensions and requiring simple welding techniques (either MMA, MIG or DC TIG). Conversely, aluminium is harder to manufacture frames from as only AC TIG can be reliably used to join this material, which requires a much more expensive welder (which the author did not have access to) and greater skill. A list of chemical and mechanical specifications for Grade 355 tube can be found in appendix H.

5.6.1 AGV Body Conceptual Designs

Four concepts were explored for the AGV's body; the merits and shortfalls of each concept are listed in the following sections.

5.6.1.a AGV Body Design 1

The first concept was based on the idea of making a hexagonal AGV; this idea is illustrated in figure 5.48. The castor and drive units are placed within the square central part of the AGV. There are two half-hexagon parts attached to the main box frame; these structures would contain the electronic boxes.

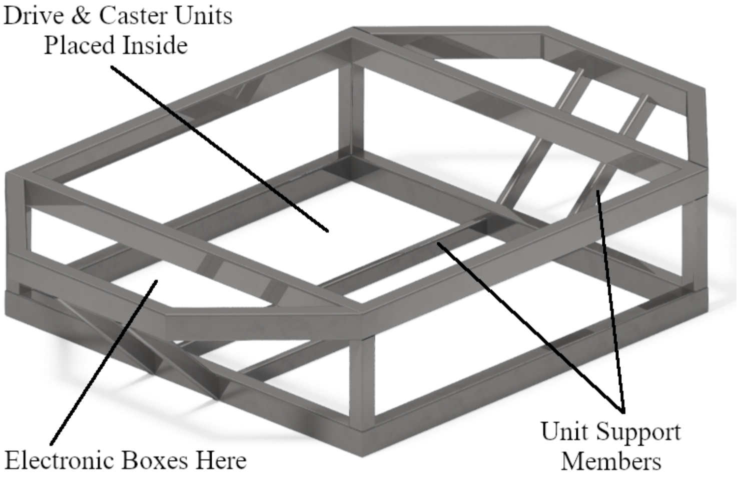


Figure 5.48: Concept 1 AGV Frame

-

The primary issue with this design is the large overhang of the body over the wheelbase, with minimal space available for the electronic components. There was also no allocation made in this design for battery placement. One advantage of this design is that the wheelbase is square, which simplifies rotation about the AGV's centroid.

5.6.1.b AGV Body Design 2

The second design is illustrated in figure 5.49; this design was heavily based on design 1 but with the corner chamfers removed.

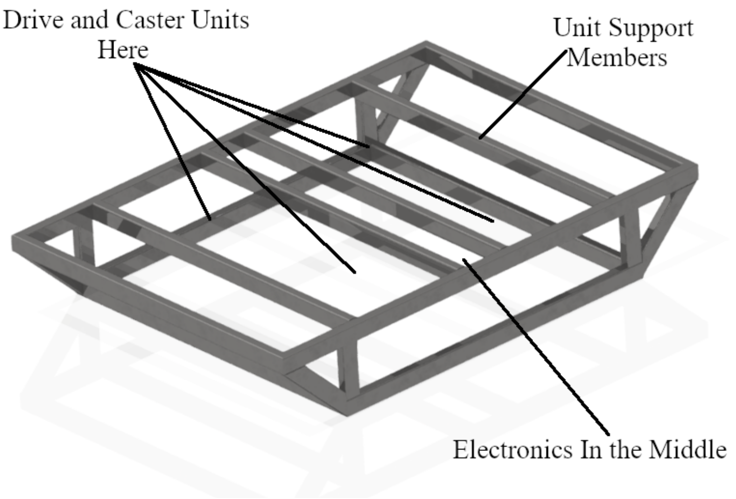


Figure 5.49: Concept 2 AGV Frame

Removing the corner chamfers created more space for components. However, removing the corner chamfers reduced the visual appeal of the AGV. Other changes included flipping structural members supporting the drive and castor units 90°and placing them at the top of the AGV's structure rather than along the bottom (as in design 1). This design's advantage over design 1 is that it has additional space for electrical components. It, like design 1, has no allocation for battery placement and has a square wheelbase.

5.6.1.c AGV Body Design 3

Design 3 marked a move away from the square wheelbase of designs 1 and 2 to a rectangular one, as illustrated in figure 5.50.

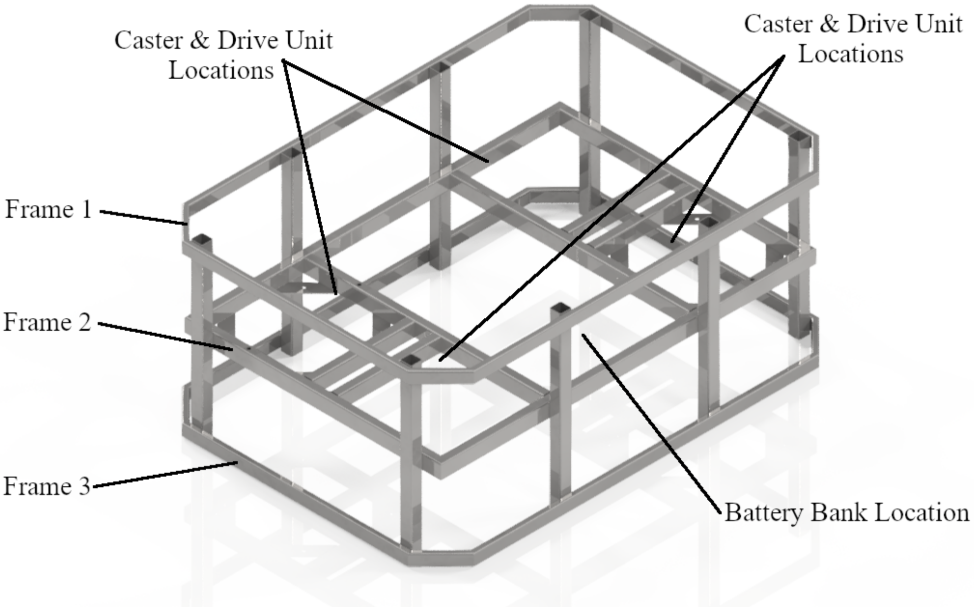


Figure 5.50: Concept 3 AGV Frame

This design has three frames (labelled frames 1, 2 and 3 in figure 5.50) sandwiched together with eight verticals. The centre frame, frame 2, attaches the drive and castor units to the vehicle. Frame 1 and 3 support the exterior cladding and provide rigidity to the vehicle. This AGV frame was elongated from the square designs of concepts 1 and 2 to make room for a battery bank for power.

The main advantage of this system is the inclusion of room for the battery bank, which designs 1 and 2 did not have. The tall nature of this design also meant that the electronics could easily be placed above the castor and drive unit mounting points. The additional height came at the cost of the vehicle's stability since this AGV has the smallest height to wheelbase ratio, with the height of the AGV being almost equal to the width of the AGV. Another disadvantage of this design was using three structural frames instead of the two present in the other concepts. The extra frame adds a significant amount of weight with minimal rigidity gain.

5.6.1.d AGV Body Design 4

Design 4 took the best features of design 3 and improved upon them. These improvements can be seen in figure 5.51. Like design 3, this design does not have a square wheelbase either.

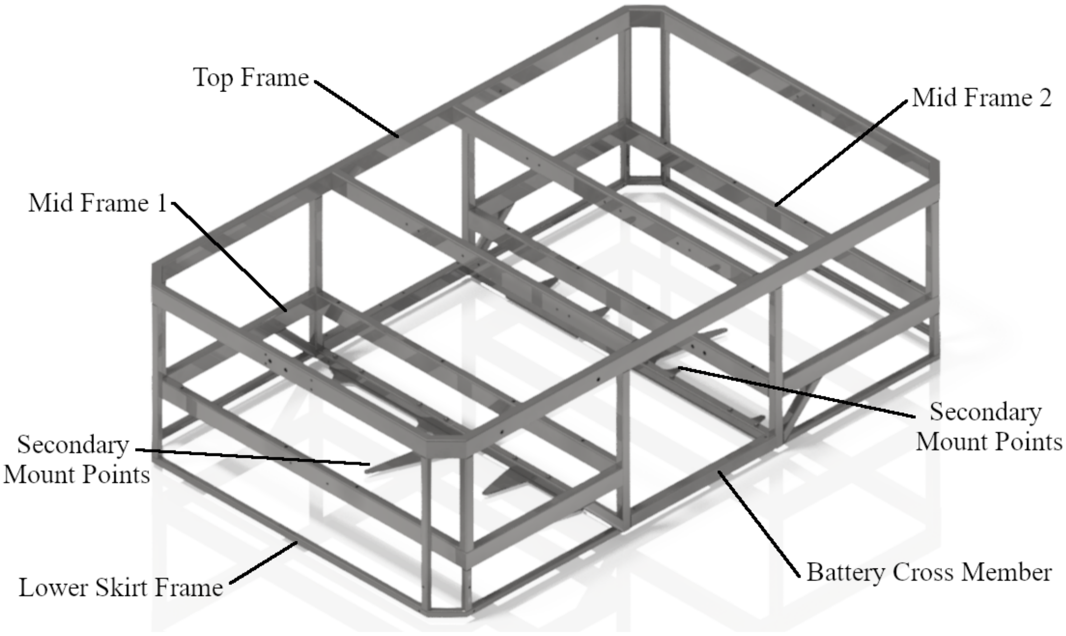


Figure 5.51: Concept 4 AGV Frame

As illustrated in figure 5.51, concept 4 still technically has three horizontal frames like concept 3. However, the size of the members varies drastically depending on whether the member is structural or only to support cladding. The top frame is still made from a larger 50 x 25 mm rectangular tube, but the second frame is now split into two separate segments called Mid Frame 1 and Mid Frame 2. These mid frames hold one castor unit and one drive unit. The split was to open up the sides of the AGV to allow the rapid exchange of the battery bank. This split weakened the AGV's frame, and the midsection of the lower frame was increased ("Battery Cross Member") to compensate for this. The rest of the lower frame was made from 19 mm square tube as it served no structural purpose besides supporting the cladding on the AGV. Also included in this design, and not in any other concept, is the addition of a second mounting point ("Secondary Mount Points") along the lower frame.

The advantages of this design include the allocation of space for the battery bank, the lowering of the overall height of the AGV compared to concept 3, the optimisation of structural member size vs loading requirements, and the secondary mounting points for the drive and castor units. The disadvantages include weakening the frame by removing the cross member in the mid-frame to allow the batteries to be easily removed and the non-square wheelbase.

5.6.2 AGV Body Final Design

The AGV body's final design chapter is broken up into two parts. The first is selecting the final design from the four conceptual designs, while the second is optimisation and additions made to the design after it was selected.

5.6.2.a AGV Body Selection Matrix

The four conceptual designs for the AGV's body, discussed in the previous sections, are not wholly independent designs but rather iterative improvements. It can already be surmised that concept 4 was the chosen concept; this is not to say the other designs are not without merit. The advantages and disadvantages of each design are listed in table 5.16.

Table 5.16: Comparison of AGV Body Concepts

Concept	Advantages	Disadvantages
Concept 1	· Small overall size · Visually appealing · Square wheelbase	· No battery space allocation · Limited electronics space
Concept 2	· Medium electronics space · Simple to build · Square wheelbase	· Ugly design · No battery space allocation
Concept 3	· Battery space allocation · Large electronics space · Extremely Rigid	· Very tall design · High centre of gravity · Heavier than other concept · Complex to build
Concept 4	· Lighter than concept 3 · Battery space allocation · Large electronics space · Removable battery bank · Visually appealing · Two mounting points for for drive and castor units	· Rectangular wheelbase · Larger and heavier than concepts 1 and 2

As can be seen from table 5.16, concept 4 is the best choice, with the most advantages and the most negligible disadvantages.

5.6.2.b Completion of the AGV Body

Concept 4 was the chosen concept for the AGV's design. Two subsystems were added to this frame to improve it. The first was a rail system to allow the battery bank to be easily added and removed. The second was a mounting system for the AC inverter(the inverter will be discussed in chapter 6).

Battery Box Rails

As mentioned in the previous sections, concept 4 of the AGV's body made allowance for a removable battery bank. A rail system is needed for this battery bank to be exchanged quickly. This rail system, ideally, will be used with an automated battery exchange system (hereafter referred to as the "dock") and thus should have a reasonable tolerance built into it to allow for misalignment during the docking process, see figure 5.52.

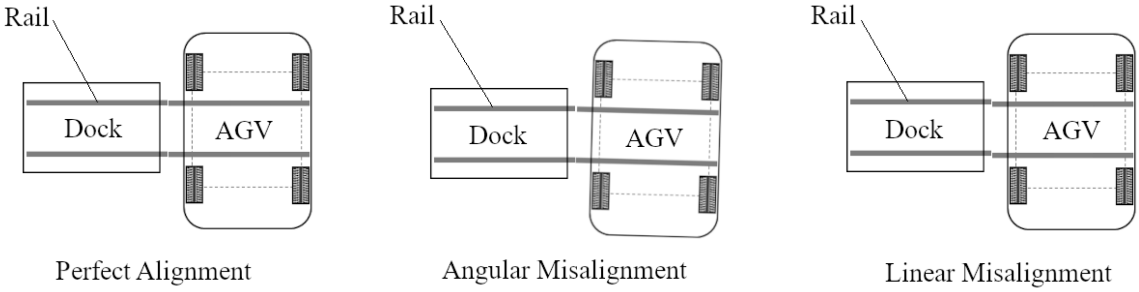


Figure 5.52: AGV and Battery Dock Alignments

Since there can be misalignments between the dock and AGV, more traditional machine rail methods such as SBR or MGN rails were out of the question (see figure 5.53) due to their high tolerance requirements.

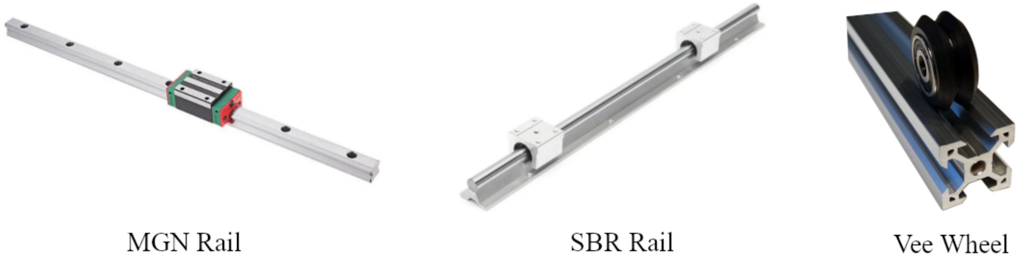


Figure 5.53: Standard Machine Rail Solutions[13]

Another option for the rail system was "Vee" wheels and aluminium extrusion. This system would be forgiving enough that the system should still function when dock and AGV rails are misaligned. Since the V-shaped nature of the wheels would cause self-alignment under the force of gravity. However, most Vee bearings are not designed for the high loads that would result from the weight of a lead-acid battery bank. Another concern is that Vee wheels are meant to be used in pairs, one on either face of an extrusion segment. Since they are paired, they do not fit very deeply into the extrusion's V slot, relying on the opposing wheel to keep them in the slot. In the proposed battery rail system, they will only be used on one side and could "jump" out of their V track if the AGV were to traverse an aggressive bump.

The last option available and the one chosen as the solution to the problem was the use of a traditional train track system, where mirrored flanged wheels are used to align the battery bank to the rails, see figure 5.54.

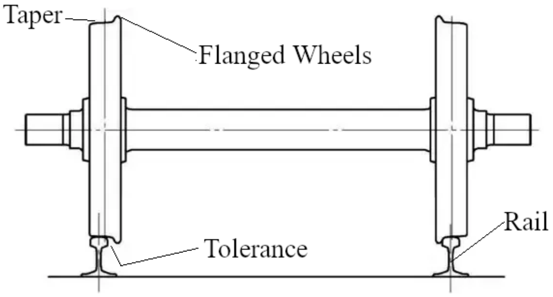


Figure 5.54: Traditional Train Rail System[14]

In addition to having flanges, locomotive wheels are also tapered along with the rail's top surface. This aids with the alignment of the train to the track and assists with turning [61]. Since the taper was omitted, it was opted to use standard angle iron for the battery bank rails while the wheels were hand lathed. The modified train rail system used in the AGV battery bank is illustrated in figure 5.55.

[13] Images found in public domain

[14] Image adapted from Sun [60].

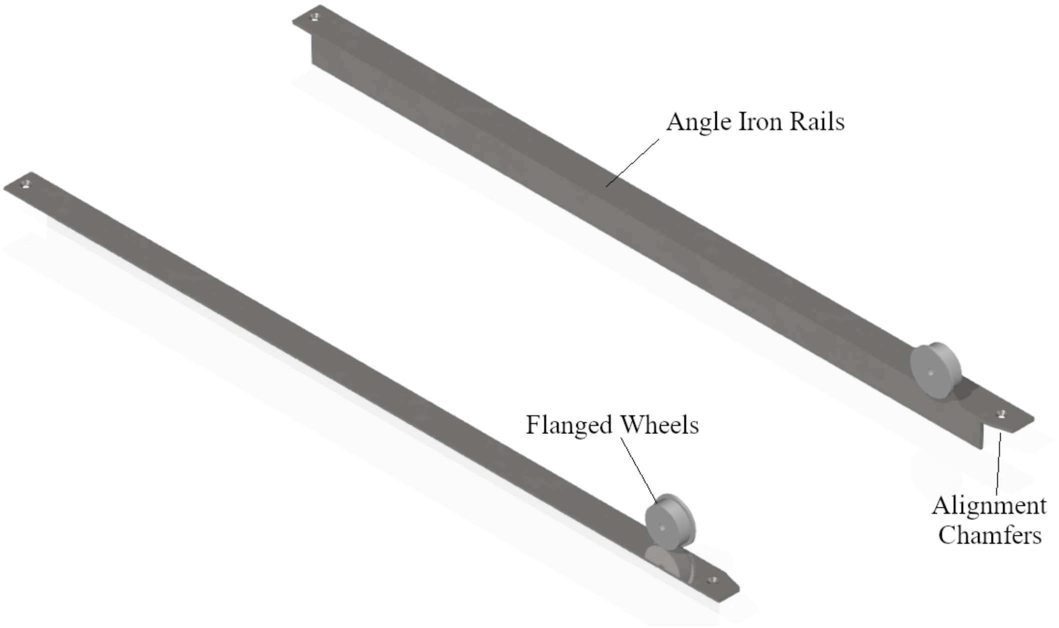


Figure 5.55: Battery Rail System

A chamfer cut into the interface portion of the rails aided in the alignment of the dock and AGV rails. The depth of the chamfer is less than the thickness of the flanged wheels to prevent the battery bank from falling off the rails. The rails are attached to the AGV via four countersunk bolts. The battery bank rail system installed in the main AGV body is illustrated in figure 5.56.

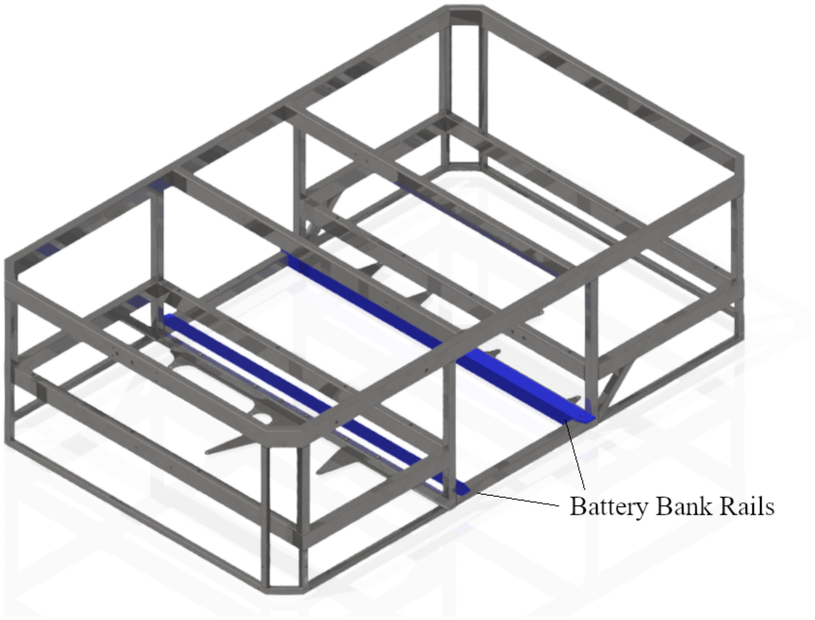


Figure 5.56: Battery Rail System in AGV Body

The battery system rails are illustrated in blue in figure 5.56. The rails will be galvanised with zinc to make them corrosion-resistant and wear-resistant.

Inverter Mounting Mechanism

Since an inverter is needed in this AGV to power some electronics, a mounting system had to be created for the RCT Axpert 5K MKS inverter. Since space was available above the battery bank, it was decided that the inverter would be mounted there. This mounting system is illustrated in figure 5.57.

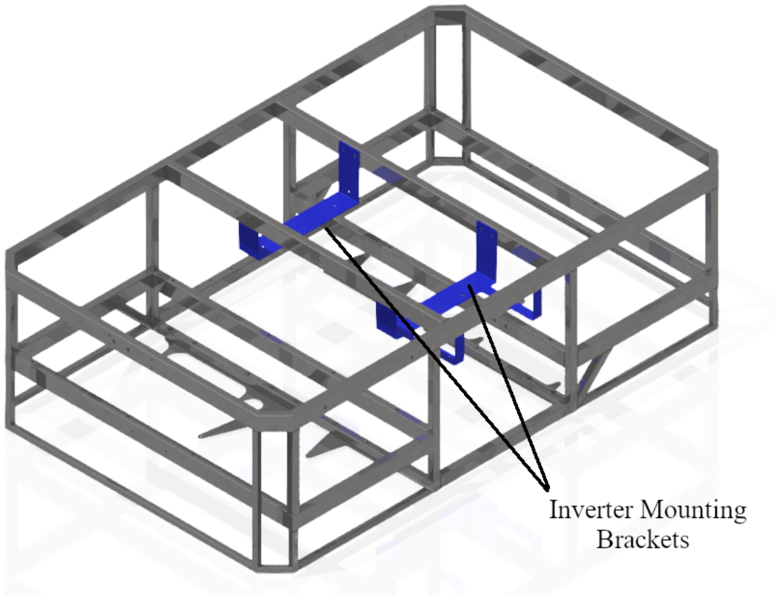


Figure 5.57: Inverter Mounting System

The inverter mounting brackets are highlighted in blue in figure 5.57.

5.6.3 Standard Parts Used in The AGV Body

The standard components used in the construction of the AGV's body are listed in table 5.17. For more details on which size steel tubes are used in what location, refer to the working drawings in appendix I.

Table 5.17: AGV Body Standard Parts List

Part Chosen	**Manufacturer**	**Component Description**
SABS 50x25x2 rect tube	Steel, Pipes & Fittings	Body structural steel
SABS 25x25x2 square tube	Steel, Pipes & Fittings	Body structural steel
SABS 38x25x2 rect tube	Steel, Pipes & Fittings	Body structural steel
SABS 19x19x2 square tube	Steel, Pipes & Fittings	Body structural steel
RCT Axpert 5K MKS	Rubicon	5kW DC to AC inverter

5.6.4 AGV Body Final Design

The final design comprises concept 4 and the additions and modifications listed in the previous sections. The final design is illustrated in figure 5.58.

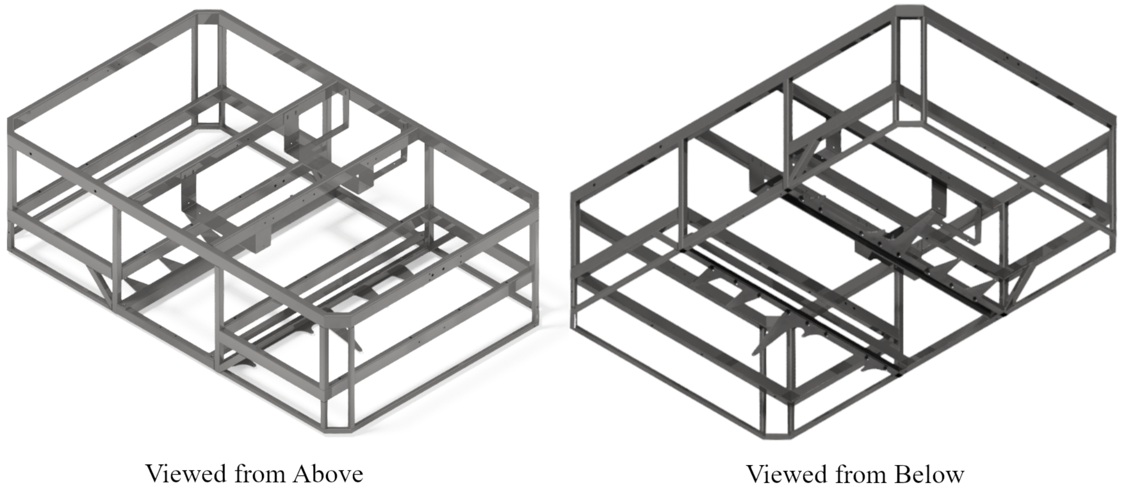


Figure 5.58: Final Design of the AGV Body

5.7 Electrical Component Boxes

This section describes the mechanical design of the electronic systems, for the electrical design, refer to Chapter 6.

Two aluminium boxes were constructed to house the electronics of the AGV. Aluminium was chosen as the material of construction for two reasons. Firstly aluminium is lighter than steel of equivalent stiffness, and second, using aluminium would negate the need for an additional corrosion resistance process (painting, galvanising or electroplating).

These boxes are located above the castor wheels in the AGV. This position was chosen as the castor units do not need the additional headroom that the drive units used, and as such, a void was available at this location within the AGV's body. The two electronics boxes are shown in figure 5.59.

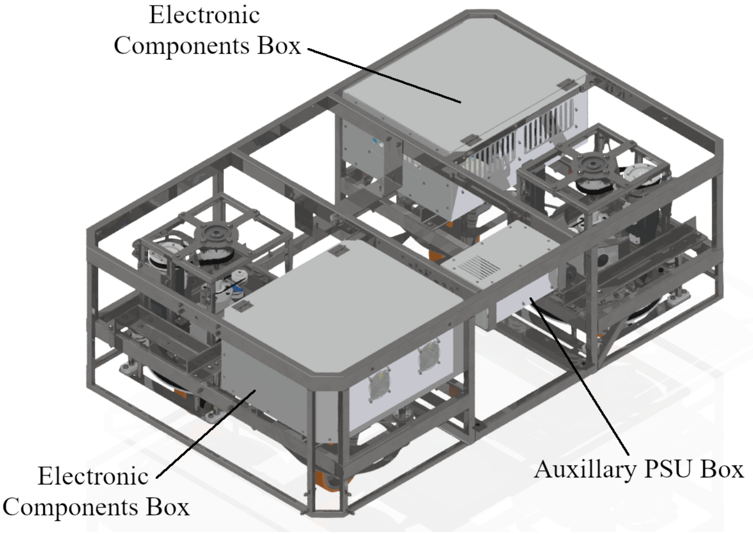


Figure 5.59: Location of Electronics Boxes

5.7.1 Control Systems Box

One of the two electric components boxes is the control system box. As the name implies, the control system box contains the AGV's control processors and some additional components such as relays and PSUs. The design of the control unit is illustrated in figure 5.60.

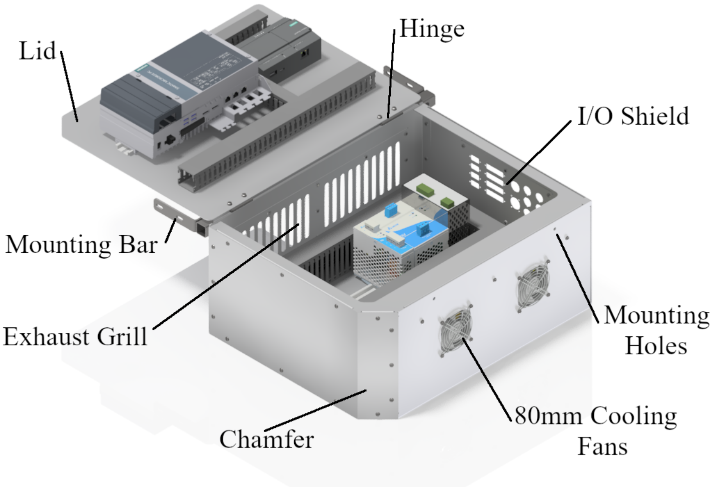


Figure 5.60: Control System Box

The corners of the electronics box are chamfered, as illustrated in figure 5.60, to fit in the corner of the AGVs frame. The control box is mounted at two points. The first is along the "front", where two mounting holes are available to bolt the box to the structural frame, see figure 5.60. The second is a mounting bar along the rear of the box. Cooling is generated by two 80mm fans that draw cool air from the exterior of the AGV, pass it over the electrical components, and out of the exhaust grill into the centre of the AGV (see figure 5.60). A detachable I/O shield was created to hold the plugs that would connect the control box to the rest of the AGV.

The layout of the significant electrical components is illustrated in figure 5.61 for the control systems box.

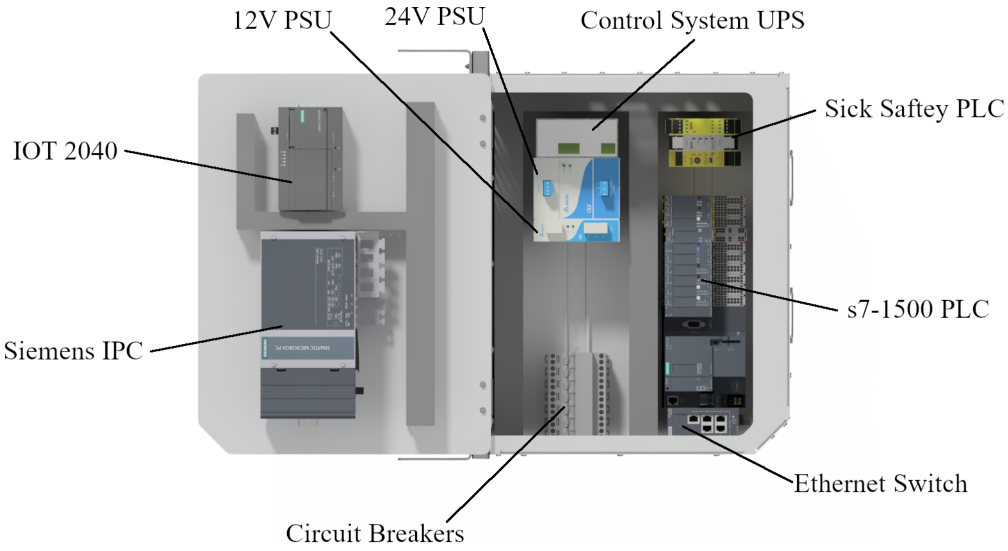


Figure 5.61: Control System Box Electrical Layout

A detailed explanation of the electrical components and their functions can be found in chapter 6.

5.7.2 Drive Systems Box

The second electronic component box contains the electric motor drives and motor control systems. This box will be referred to as the "Drive Systems Box". The drives were separated from the rest of the electrical components for two reasons. The first is size limitations; there is only so much space available in one of the boxes of the electronic components. The second is electrical interference; motor drives create interference in the form of induction, especially drives that produce AC waveforms (such as the ones used in this AGV).

Like the control systems box, the drive systems box is cooled by two 80mm fans that source cool air from the exterior of the AGV, push it over the electrical components, and out of the rear of the drive systems box and into the interior of the AGV. The mounting of this box is identical to the control system box and has a detachable I/O shield for quick detachment from the main AGV. The drive systems box is illustrated in figure 5.62.

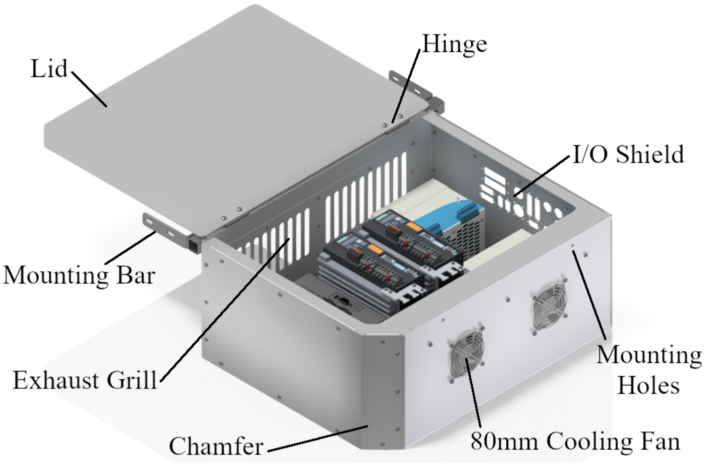


Figure 5.62: Drive System Box

Like the control systems box, the drive systems box has a chamfer on one corner to correctly fit into the AGV's body. The layout of the electrical components within the box is illustrated in figure 5.63.

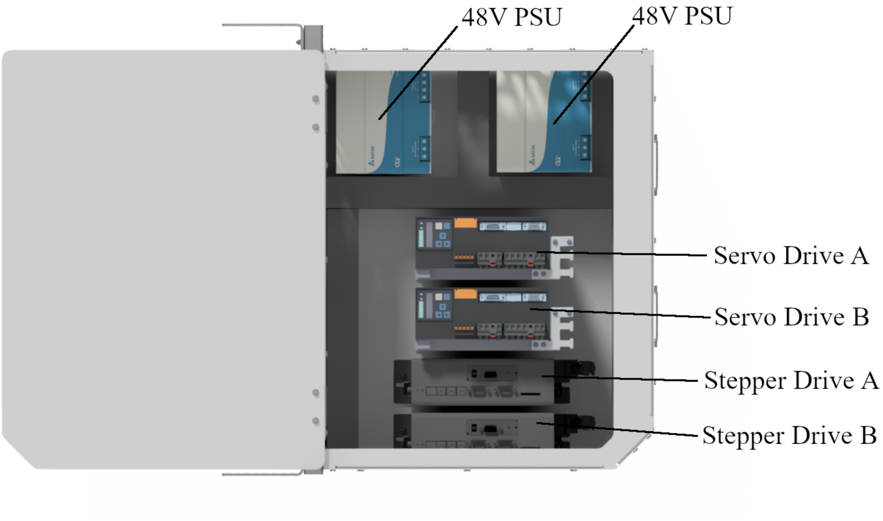


Figure 5.63: Drive System Box Electrical Layout

Two 48V DC power supplies are needed to power each stepper motor drive separately. Using two supplies was done due to the high current requirements of these drives. Additional information on the wiring and operation of the components and their functions can be found in chapter 6.

5.7.3 Standard Parts Used in The Electric Component Boxes

The standard components used to build the electric component boxes are listed in table 5.18.

Table 5.18: AGV Electric Components Box Standard Parts

Part Chosen	Manufacturer	Component Description
Siemens IPC457E	Siemens	Siemens industrial computer
Siemens 1512SP	Siemens	Siemens s7-1500 PLC
SICK FX3-CPU320002	SICK	SICK safety PLC
Siemens IoT2040	Siemens	IoT Controller/Gateway
LOGO!Power	Siemens	12V DC 4.5A power supply
Traco Power TSP-BCMU360	Rubicon	24V DC UPS switchover
Meanwell DRP-240-24	Meanwell	24V DC 10A power supply
Scalance XB005G	Siemens	5 port Ethernet switch
Meanwell SDR-480-48	Meanwell	48V DC 10A power supply
Festo CMMS-ST	Festo	Stepper motor drive
Siemens V90 (s-1FL6)	Siemens	Servo motor drive

5.8 Scanner and Safety Sensor Mounting

This section has two subsystems, namely the "LiDAR scanner stalk" and the "safety sensor mounts".

5.8.1 LiDAR Scanner Stalk

The LiDAR scanner stalk supports the Sick NAV 350 LiDAR scanner, a WIFI access point (to allow the AGV to communicate with the surrounding factory), an E-Stop and a flashing front light to warn people of the AGV's presence.

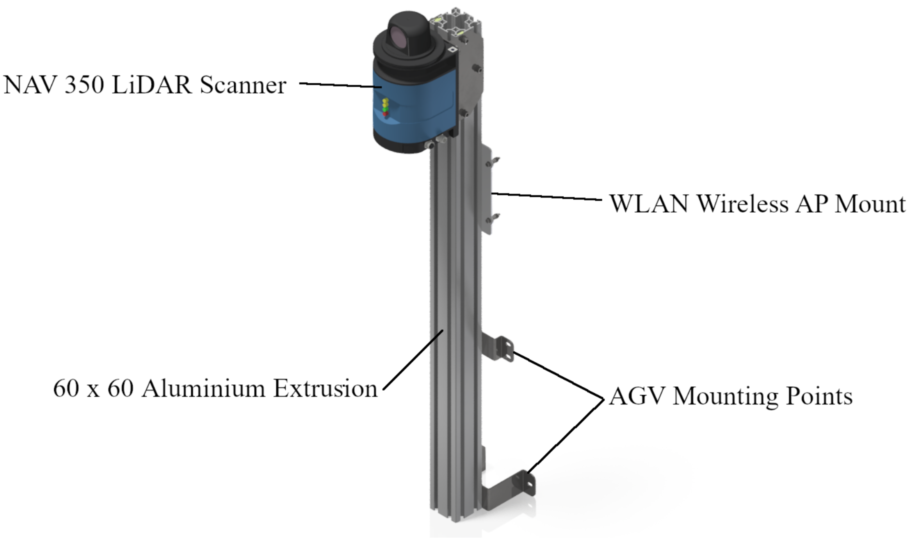


Figure 5.64: LiDAR Scanner Stalk

The warning light and E-Stop are not shown in figure 5.64, as they were not included in the original CAD design. They were added later during the construction and can be found in the photographs of the final design in the conclusion chapter. The sensor wiring was routed inside the aluminium extrusion via holes drilled from the exterior to keep the system neat. The LiDAR scanner stalk's position relative to the AGV's body can be found in figure 5.65.

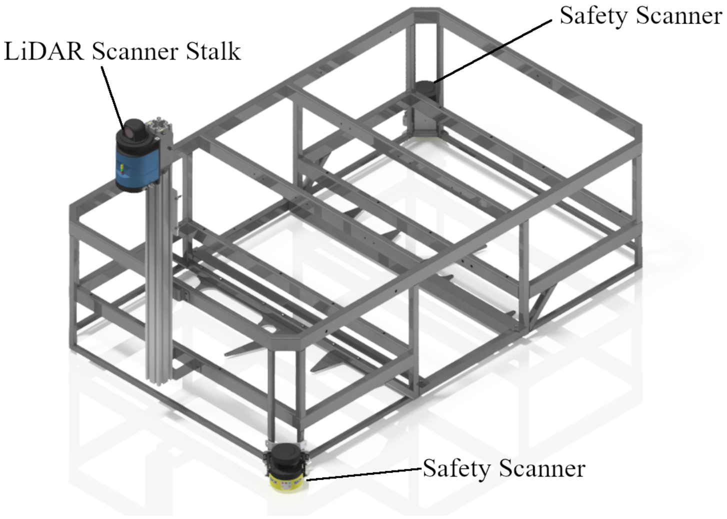


Figure 5.65: LiDAR Scanner Stalk and Safety Sensors Mounted on AGV

5.8.2 Safety Sensor Mounts

There are two "SICK S300 Mini Remote" safety sensors placed on diagonal corners of the AGV. The diagonal placement was done to minimise the total number of safety scanners needed as these sensors have a viewing angle of 270 °. As such, when placed on diagonal corners of the AGV, each sensor can easily detect objects that approach the AGV from two sides; see figure 5.66.

The SICK S300 mini remote can be configured with three detection zones layered like an ogre. The outermost zone is configured to limit the AGVs top speed when triggered, the second zone will bring the AGV to a controlled stop, and the innermost zone will emergency stop the AGV. Under normal conditions, when the AGV encounters a moving obstacle (such as a person), the obstacle should enter the controlled stop zone before the emergency stop zone. Thus the emergency stop system should not be triggered as the AGV would have already been halted before the obstacle entered the emergency zone.

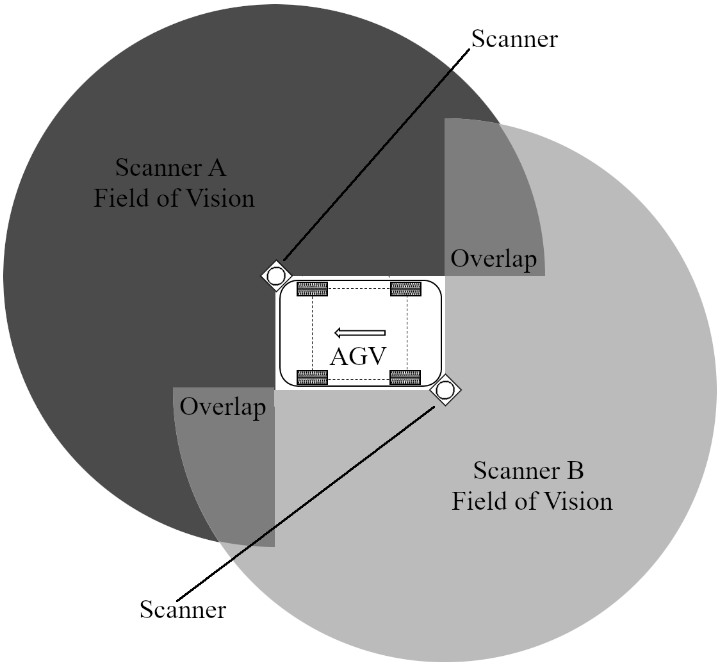


Figure 5.66: SICK S300 Mini Remote Safety Scanner Layout

The mounting for the Sensors is illustrated in figure 5.67, the position of these sensors on the AGV's frame can be found in figure 5.65.

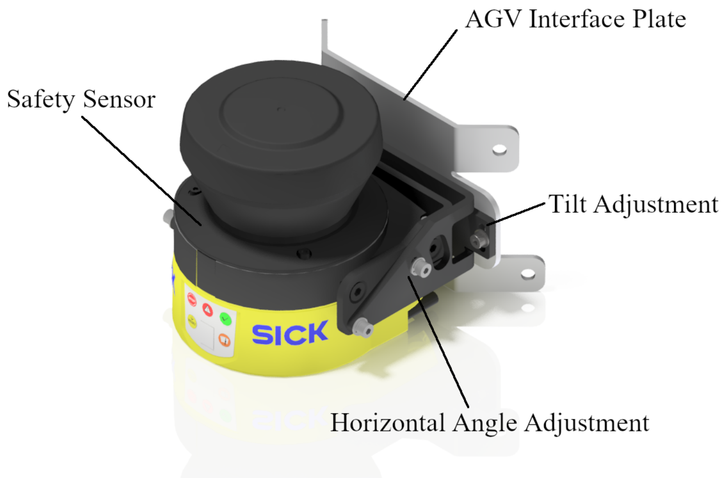


Figure 5.67: Safety Sensor Mounting Hardware

The mounting hardware has two angle adjustments for levelling the sensor beam relative to the ground; see figure 5.67.

5.8.3 Standard Parts used in The Scanner and Safety Sensor Mountings

A list of all of the standard components used in the LiDAR scanner stalk and the safety sensor mounts can be found in table 5.19.

Table 5.19: Scanner and Safety Sensor Mountings Standard Parts

Part Chosen	Manufacturer	Component Description
SICK NAV350-3232	SICK	Navigation LiDAR
Scalance W746-1	Siemens	Wireless network client
SICK S300 Mini Remote	SICK	Safety scanners

5.9 Main & Auxiliary Battery Unit

There are two battery banks in the AGV; the first is the large "Main Battery Unit", which provides the operational power for the AGV, including power for the motors. This main battery bank is exchangeable (i.e. a depleted battery unit can be exchanged with a charged one to minimise AGV downtime due to charging time).

Removing the main battery bank causes an issue since the AGV's control system will shut down when the power unit is removed. The battery unit exchange cannot be performed without an active control system. This issue was solved by adding an auxiliary battery bank that provides just enough standby power to run the control system (the drives, sensors, etc. will be shut down) during the battery exchange operation. The auxiliary battery bank is recharged from the main battery bank during regular operation.

5.9.1 Main Battery Unit

As mentioned, the main battery unit is removable. It interfaces with the AGV via the rails described in section 5.6.2.b. The main battery unit is more than a simple battery pack; it contains a fully standalone battery management system based on the Siemens IoT 2040 controller. This controller can perform Columb counting to determine the state of charge (SoC) and battery temperatures. The assembled battery management system is shown in figure 5.68.

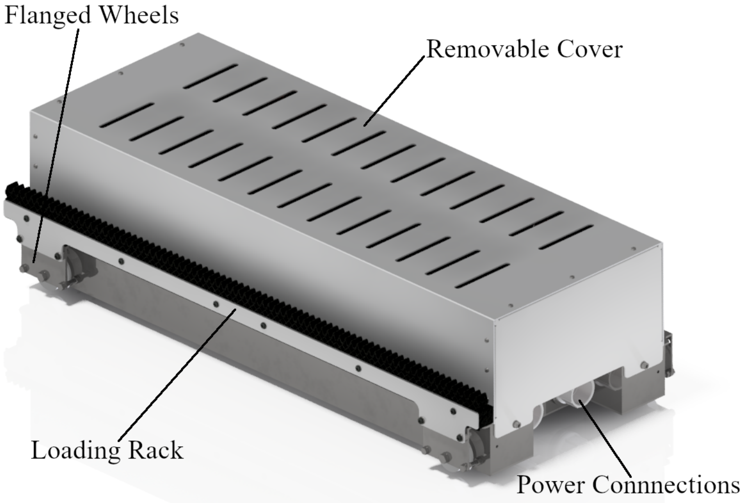


Figure 5.68: Main Battery Unit Fully Assembled

As illustrated in figure 5.68, the battery unit has four flanged wheels to mate with the rails implemented in section 5.6.2.b. A rack gear is attached to one side of the battery box to exchange the battery bank. This rack is driven by a DC motor inside the AGV with an appropriate pinion. Although it may seem counterintuitive to have the motor on the vehicle body rather than on the battery bank (from a mechanical standpoint, since the battery bank is the moving object), it will significantly reduce scaling costs. Scaling cost reduction occurs because for each AGV in operation, there could be four or five times as many battery banks in circulation; thus, having one motor per AGV rather than one motor per battery bank makes economic sense.

The major issue with having the rack's pinion on the AGV is how to transfer the battery bank beyond the bounds of the AGV since the rack gear is the same length as the battery bank. There are two solutions to this issue. The first is to have a second pinion motor on the charging station; the issue with this is synchronising the pinion teeth to the rack teeth to "hand-off" the battery bank from the AGV to the charging station. The easy solution for this problem is to add at least 180°of "slop" to the second opinion. Thus the pinion gear will be in mesh before the motor can "bite". Option 2 is to have the charging station rails pivot-able using a pneumatic cylinder; the pivot would only have to be about 1°-2°, just enough to keep the battery bank rolling under its own weight. See figure 5.69 for this concept.

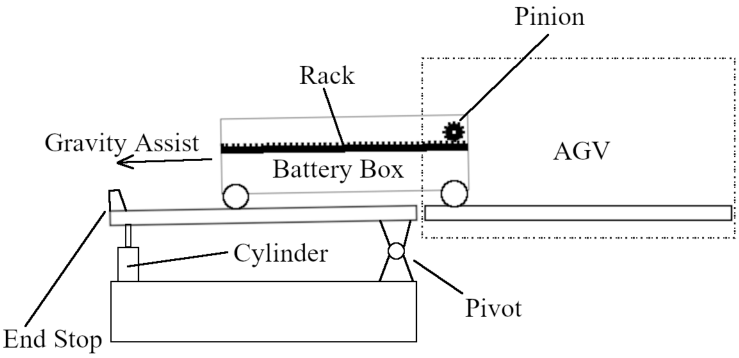


Figure 5.69: Charging Station Pivot Concept

When the battery bank is to be loaded, the angle of the charging station rails is inverted. The main battery bank will then attempt to roll into the AGV. The pinion can then "bite" the rack gear and pull the main battery unit into the AGV.

The main battery unit contains four "First National Battery" silver calcium lead-acid batteries. These batteries are rated at 12V DC and 102Ah and are wired in series to give a voltage of 48V DC at 102Ah. The battery layout and placement of the electronics backplate are illustrated in figure 5.70.

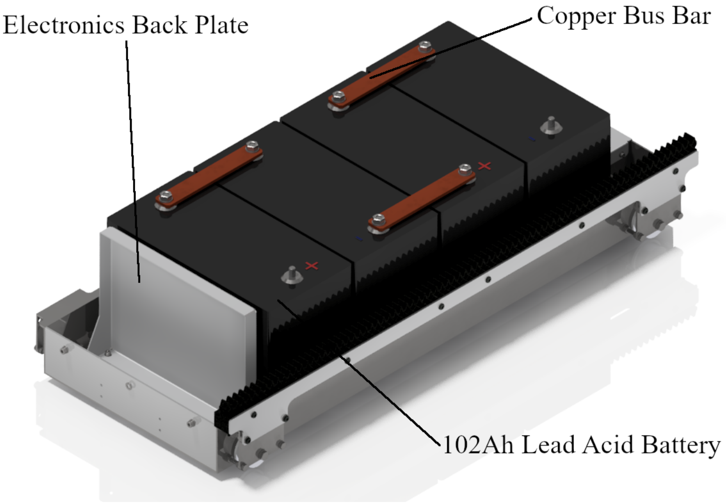


Figure 5.70: Main Battery Unit with Cover Removed

The docking interface that transfers power to the AGV from the docked main power unit is illustrated in figure 5.71.

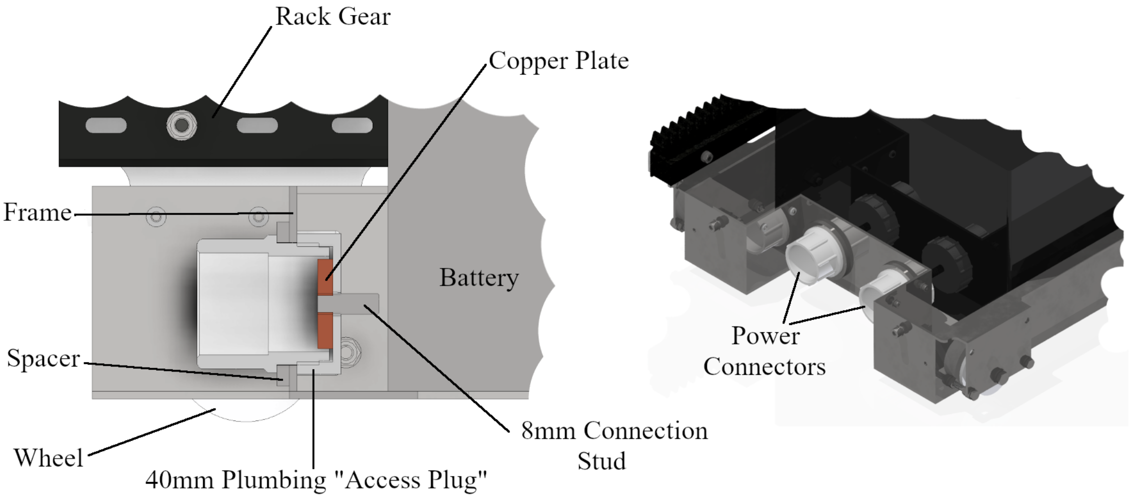


Figure 5.71: Main Battery Unit Electrical Connections

The connector housing was made from a modified PVC plastic access plug intended for use with 40 *mm* plumbing parts. Using an access plug provided electrical insulation for the copper plate that formed the contact for the connection. The copper plate is attached to the batteries via an M8 stud passing through the access plug's rear. The electrical connection was made using a set of cables terminated in ring lugs secured by a nut (not shown). In addition to securing the lug, the stud nut mechanically fastened the copper plate assembly inside of the access plug. Since the batteries were directly attached to the copper connection plates, the large recess that the access plug provided served well to prevent accidental shorting of the connection points as they are permanently "live". An unmodified access plug is illustrated in figure 5.72 for clarity.

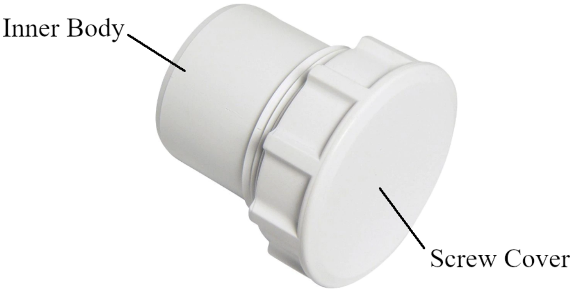


Figure 5.72: An Unmodified Access Plug

5.9.2 Auxiliary Battery Unit

The auxiliary battery unit cannot be removed from the AGV; this battery unit is nothing more than a simple box containing three 12V DC lead-acid batteries, each rated at 7Ah. These batteries are wired in parallel, creating a combined voltage of 12 VDC while increasing the capacity to 21Ah. The auxiliary unit is illustrated in figure 5.73.

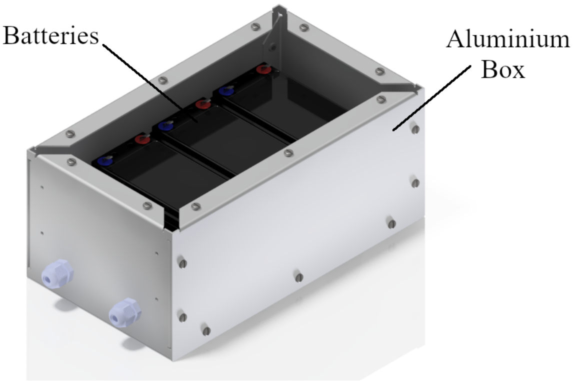


Figure 5.73: Auxiliary Battery Unit

The batteries are secured in place using polystyrene inserts (not shown). The location of the auxiliary battery unit relative to the AGV's body is illustrated in figure 5.74.

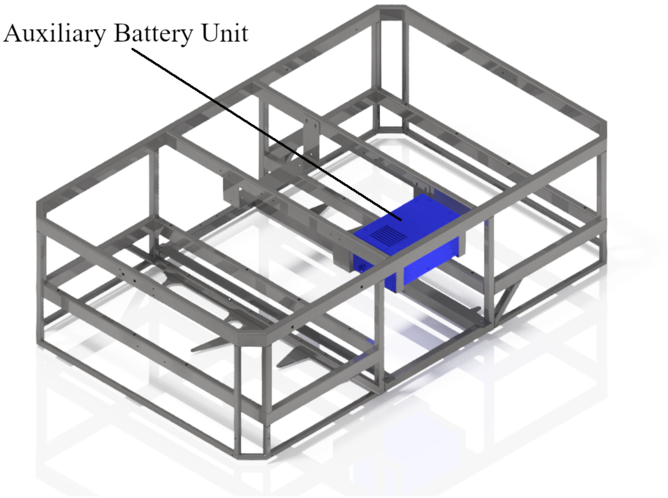


Figure 5.74: Auxiliary Battery Unit Location

5.9.3 Standard Parts used in The Main & Auxiliary Battery Unit

A list of all standard components used in the construction of the main and auxiliary battery units can be found in table 5.20.

Table 5.20: Main & Auxiliary Battery Unit Standard Parts

Part Chosen	**Manufacturer**	**Component Description**
Siemens IoT2040	Siemens	IoT Controller/Gateway
FNB SMF100	First National Battery	102Ah 12V lead acid battery
Gate Rack	Builders Warehouse	Standard POM Gate Rack
Gate Pinion	Builders Warehouse	Standard POM Gate Pinion
Access Plug	Aubrey Sacks Plumbing	Standard 40mm PVC access plug
RT1270B	Rubicon	7Ah 12V lead acid alarm battery

5.10 Cladding

The skin or cladding of the AGV is comprised of a combination of aluminium and transparent acrylic panels. The aluminium panelling forms most of the AGV's skin, providing additional structural rigidity, while the transparent acrylic panels are placed to provide maintenance access and visual inspection points. The AGV with its cladding is illustrated in figure 5.75.

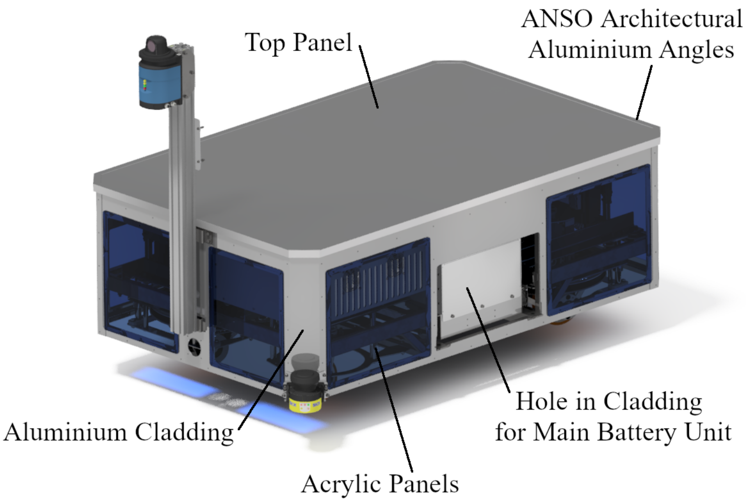

Top Panel
ANSO Architectural Aluminium Angles
Aluminium Cladding
Hole in Cladding for Main Battery Unit
Acrylic Panels

Figure 5.75: AGV Cladding

The top panel of the AGV has to be significantly more rigid than the sides, as any load the AGV carries would be placed here, and possibly a future COBOT (collaborative robot). Thus a different strategy had to be adopted to create this cladding section. In order to keep this panel lightweight, rigid and resistant to surface gouging, a composite structure was used. This composite consisted of a sheet of aluminium tread plate laminated to a layer of "engineered" plywood board using epoxy. A cross-section of this laminate can be found in figure 5.76.

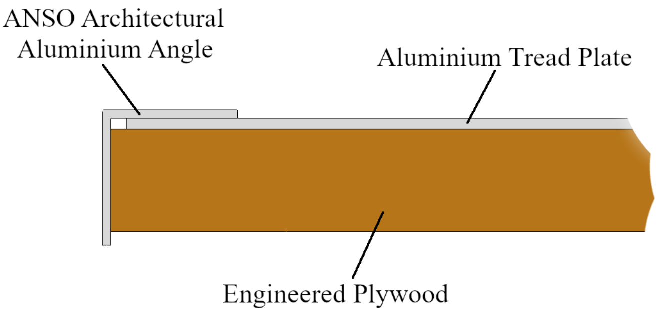


Figure 5.76: AGV Top Panel Laminate

To hide the top panel's laminations from view, it was finished using ANSO architectural aluminium angles. Not shown in the renderings was the addition of a stacker lamp near the rear of the top panel.

5.11 Miscellaneous Assemblies

This chapter contains all information relating to assemblies that are too small to have their own subsection.

5.11.1 Charging Port

The inverter (RCT Axpert 5K MKS) used in this research project contains a battery charger, and as such, provision was made in the AGV's design for a discreet charging port that could be used to charge the main battery unit while it is attached to the AGV. This charging port took the form of a panel mounted 16A 2P+E industrial plug, see figure 5.77.

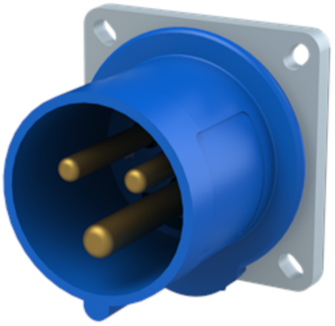


Figure 5.77: 16A 2P+E Panel Mount Industrial Plug

This plug is mounted on the front of the AGV, below the LiDAR scanner stalk, as illustrated in figure 5.78.

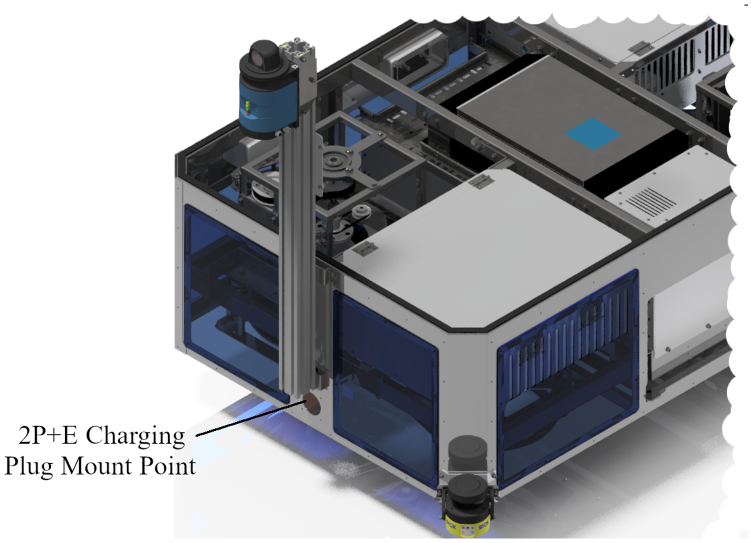


Figure 5.78: Charging Plug Location

5.11.2 Main Battery Unit Pinion System

The pinion system of the rack and pinion used to load and unload the AGV's removable battery bank is discussed in this section. The location of the pinion motor inside of the AGV is illustrated in figure 5.79.

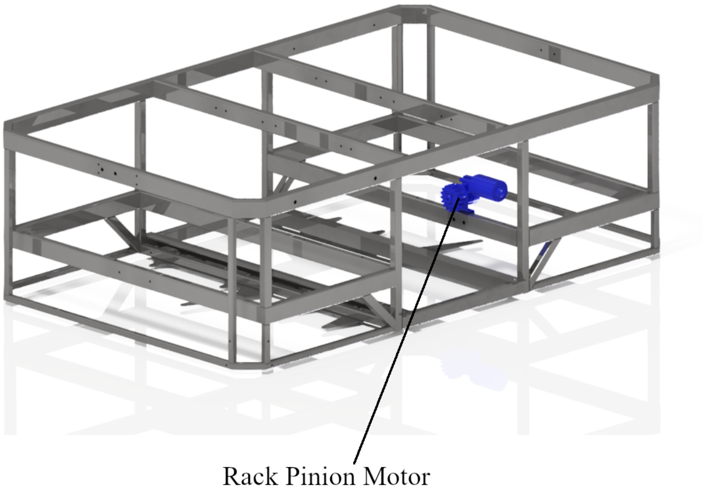


Figure 5.79: Main Battery Unit Pinion Motor Location

The pinion motor mechanism consists of a right-angle worm gearbox 12 VDC motor and a standard house gate pinion gear. The motor used is a car windscreen wiper motor and was bought new as a spare part. This motor is a spare for a Toyota Hilux and carries the part number ELP.WM10412. There were two reasons that this motor was chosen. Firstly as it is a mass-produced car part, it is cheaper than a motor marketed toward new designs. Secondly, it natively runs off of 12 VDC, the same voltage as the auxiliary battery unit from where it will draw power. The design of this mechanism is shown in figure 5.80.

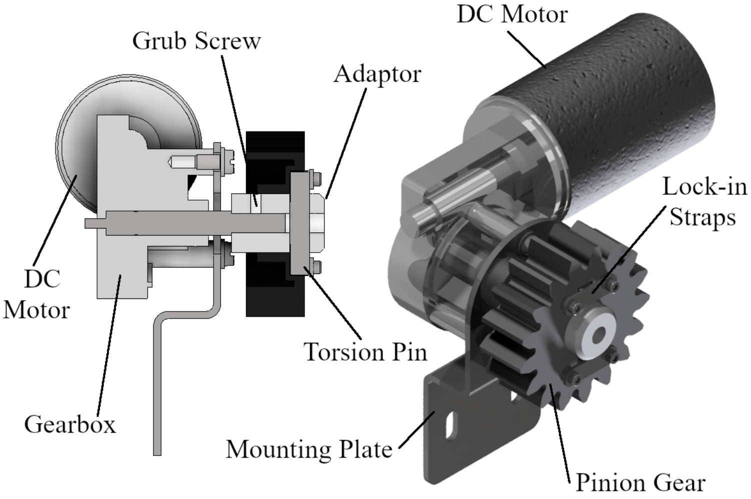


Figure 5.80: Main Battery Unit Pinion Mechanism

The pinion motor mechanism's height can be adjusted using a set of slotted bolt holes, which are used to secure this mechanism to the AGV's frame.

5.11.3 Main Battery Unit Connector

An electrical interface had to be designed to connect the removable main battery unit that supplies power to the AGV. It was insufficient to use an off the shelf connector as the high tolerance for these connectors would make them prohibitively difficult to use in an automated system. Thus a low tolerance spring-loaded system was designed as shown in figure 5.81.

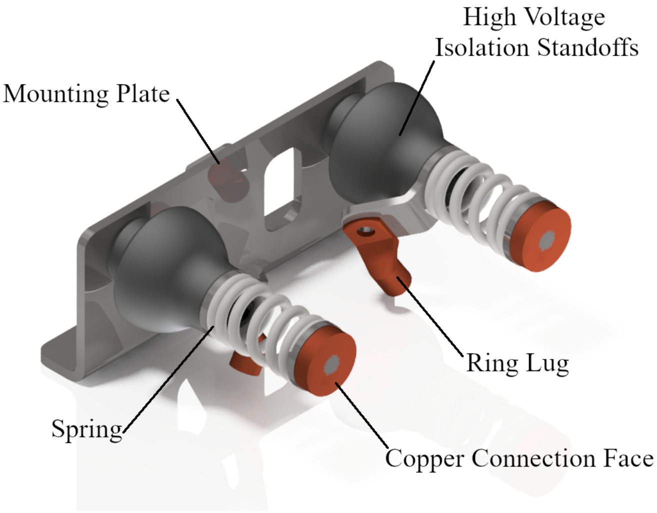


Figure 5.81: Main Battery Unit Electrical Interface

As illustrated in figure 5.81, there are two spring-loaded connectors, one for 48 VDC and one for 0 VDC. These connections are designed to transfer up to 100A; unfortunately, the springs are relatively stiff due to being thick enough to carry this current. High voltage polymer standoffs were used to insulate the contact mechanism from the metal frame of the AGV, thus preventing a dead short. These standoffs are typically used to support busbars in electrical installations.

The copper connection faces of this mechanism connects to the "copper plate" shown in figure 5.71 of the main battery unit described in section 5.9.1. When connected, the electrical interface (in figure 5.81) is wholly enclosed within the PVC access plug shown in figure 5.71. Enclosing the mechanism like this serves to prevent any possible shorting. The location of the main battery unit electrical interface is illustrated, in blue, in figure 5.82.

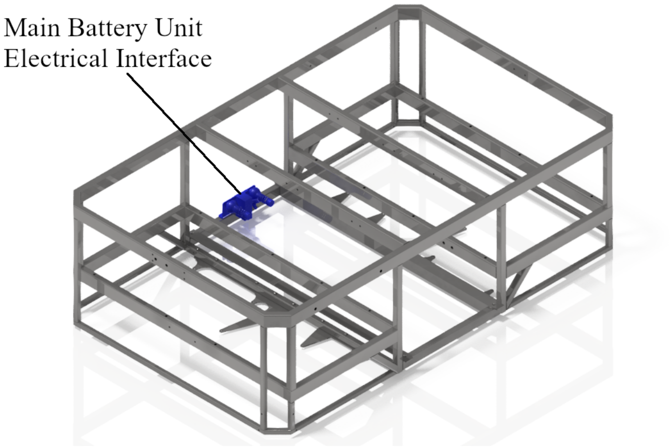


Figure 5.82: Main Battery Unit Electrical Interface Location

Located between the two spring connectors of the electrical interface is a limit switch used to detect if the battery unit is in place so that the rack pinion motor can be shut down. This assembly is shown in figure 5.83.

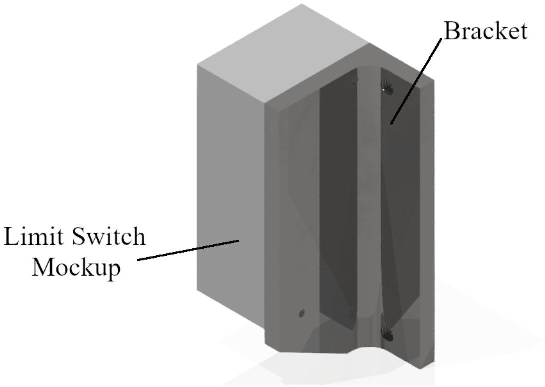


Figure 5.83: Main Battery Unit in Place Sensor

5.11.4 Cable Routing

Since cables have to be routed between the two electrical component boxes, the drive units, inverter and battery units, a cable tray system was installed in the AGV. This cable tray system used off-the-shelf Legrand 150x25 mm and 50x25 mm standard industrial cable trays. The location and layout of these trays within the AGV, are illustrated in figure 5.84.

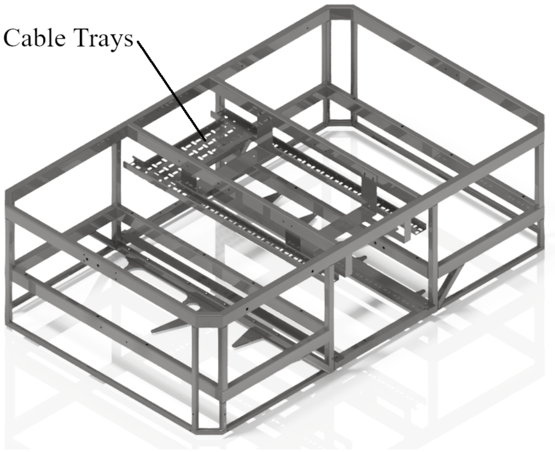


Figure 5.84: Cable Tray Layout Within The AGV

5.11.5 10V DC Steering Potentiometer Power Supply

Since the absolute steering angle of the AGV's wheels is determined by an endless potentiometer, an analogue voltage signal had to be deciphered by the control system (see figure 5.24 for control system diagram). Ingesting the analogue signal was done using an analogue input card on the Siemens 1500 PLC. Analogue voltage processing by a Siemens PLC is usually done between 0 VDC and 10 VDC; thus, a separate 10 VDC source is required. The 10 VDC source is generated using a DC-DC converter based on the XL4005 IC. This converter sourced its voltage from the 12V DC rail found in the AGV's control system. The location of this DC-DC converter is illustrated in figure 5.85.

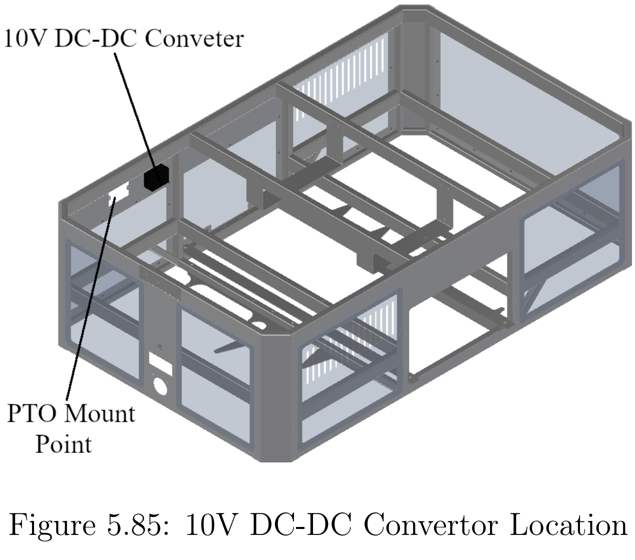


Figure 5.85: 10V DC-DC Convertor Location

5.11.6 Electrical Single Phase PTO

Allocation was made on the AGV for electrical power take-off (PTO) in the form of a standard South African single phase plug socket on the side of the AGV. This plug was included to ensure that any future equipment added to the AGV could easily source their power without destructive modifications to the AGV. This PTO mounting point is illustrated in figure 5.85, while the PTO itself is illustrated in figure 5.86.

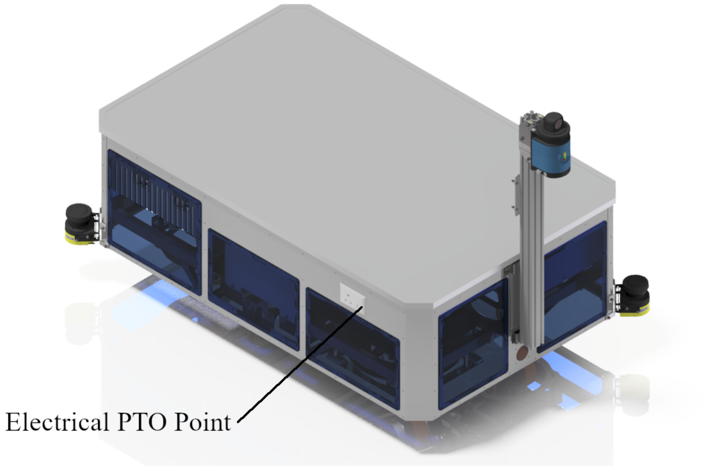


Figure 5.86: Electrical PTO Point

5.11.7 Main Battery Box Solid State Relay

The last minor mechanical assembly in this section is the solid-state relay responsible for connecting and disconnecting electrical power from the main battery bank unit. A solid-state relay was implemented to prevent arcing at the contacts described in section 5.11.3. The location of this unit is illustrated in figure 5.87.

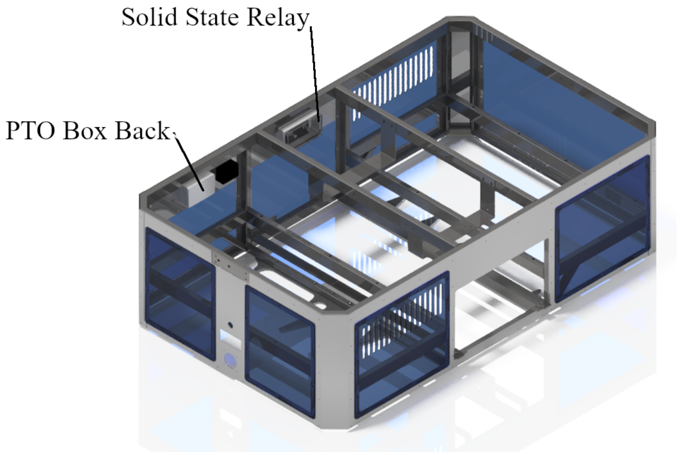


Figure 5.87: Solid State Main Battery Relay

5.11.8 Standard Parts Used in The Various Miscellaneous Assemblies

A list of all standard components used in the miscellaneous assemblies can be found in table 5.21.

Table 5.21: Miscellaneous Assemblies Standard Parts

Part Chosen	Manufacturer	Component Description
2P+E Industrial Plug	Rubicon	1 Phase industrial power connection
Toyota Hilux ELP.WM10412	Autozone Spares	Windscreen wiper motor
gate pinion	Builders Warehouse	Standard gate pinion
LV2FF-BK	ACDC Dynamics	High voltage polymer stand-off
Legrand 150x25	Legrand	Industrial cable tray
Legrand 50x25	Legrand	Industrial cable tray
DSN5000	Banggood	XL4005 DC to DC converter
SANS 164-1	Rubicon	Standard South African Plug
Crydom HDC60D120	Rubicon	120A 48VDC Solid State Relay

5.12 Chapter Conclusion

This chapter served two purposes. The first was the selection of appropriate mechanical designs that would satisfy the design requirements of this research. The second was the design and explanation of all these various components and assemblies. For the design of the drive units, design 3 was chosen. This design was built around an inline suspension system. For the castor units, design 3 was also chosen. Body design number 4 was chosen for the AGV's body. The remaining components, such as the electric component boxes, scanner mountings, battery units and cladding, did not go through a multiple design process and are only listed and described.

6 Electrical Design of the AGV

6.1 Electrical Overview

The electrical systems of the AGV are roughly split into two classes, the power electronics and the control systems. A block diagram explaining the overview of the AGV's electrical system is given in figure 6.1.

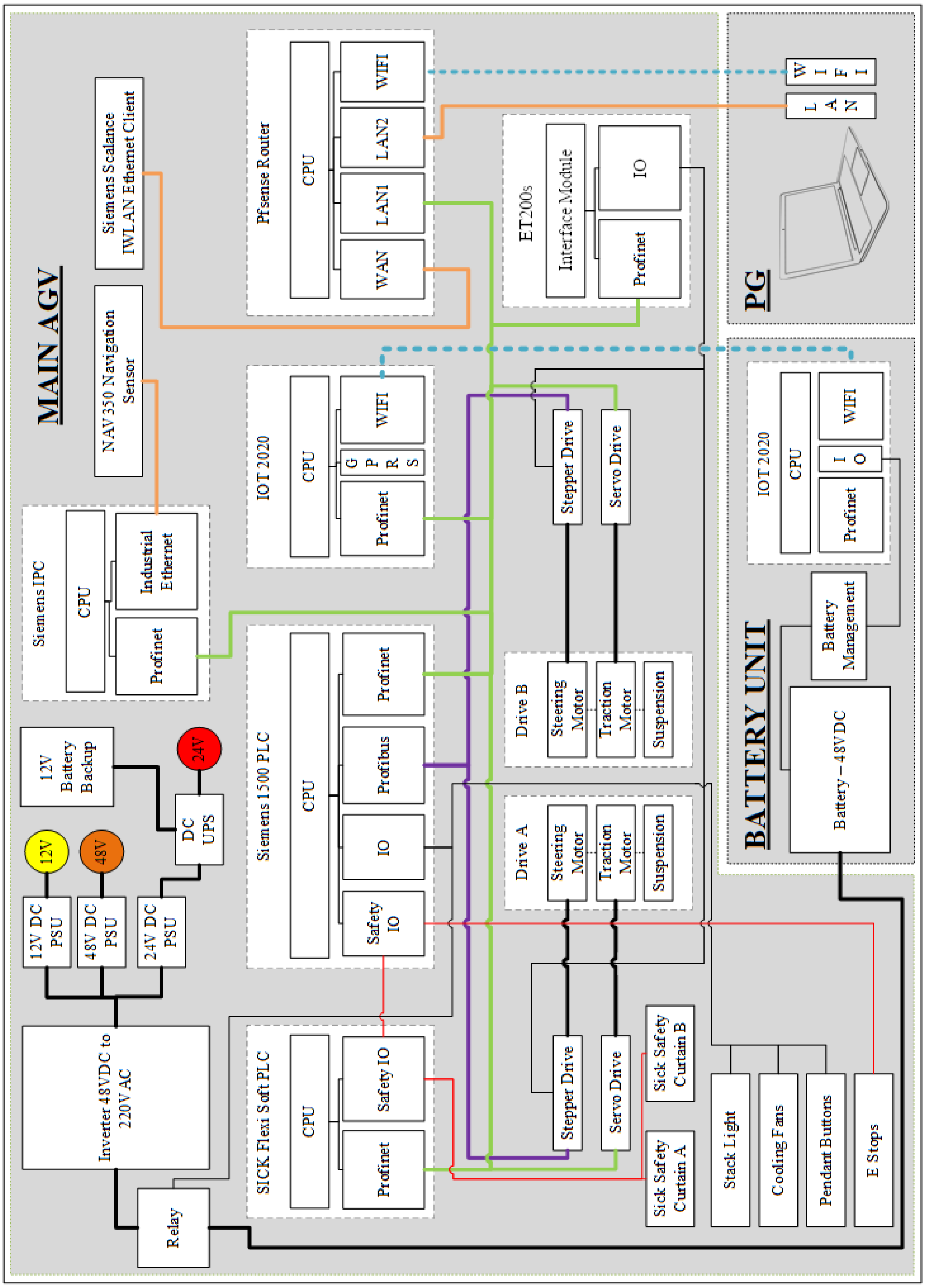


Figure 6.1: AGV Electrical System Block Diagram

From diagram 6.1 it can be seen that the AGV is split into three separate physical components. These are the main AGV, the battery unit and the programming device (PG). When reading the diagram, the connector colours have the following meanings:

- Thin Black: Boolean I/O data
- Thick Black: Power Buss
- Red: Boolean I/O Safety Data
- Orange: Industrial Ethernet (No Profinet)
- Green: Profinet real time connection (Can also support industrial ethernet and safety telegrams)
- Purple: Profibus connection (RS485 based communication)
- Dotted Cyan: WIFI connection

As seen in diagram 6.1 the AGV's body has a total of 6 discrete controllers, all performing specific tasks. The role of each of these controllers is defined in table 6.1.

Table 6.1: Summarised Controller List

Diagram Label	**Full Name**	**Component Role**
Siemens IPC	Siemens IPC427E	High level controller
Siemens 1500 PLC	Siemens 1512SP	Low level controller
IoT2020 (Main AGV)	Siemens IoT 2040	IoT gateway
SICK Flexi-Soft PLC	SICK FX-CPU320002	Safety curtain scanner controller
PFsense Router		Network controller and router
IoT2020 (Battery Unit)	Siemens IoT 2040	IoT gateway for battery unit

6.2 Power Electronics

This section deals with the power electronics in the AGV. Power electronics include the main battery unit, the auxiliary battery unit used for the DC UPS, the main power inverter and all the AC to DC power supplies.

6.2.1 Battery Unit

The main battery unit that supplies power for the AGV's tractive effort is removable. Removability facilitates speedy "recharge" of the AGV by exchanging a depleted battery unit with a charged one. The mechanical design of this unit was discussed in section 5.9.1.

The current draw of the system's major components was tabulated to size appropriate power supplies. It should be noted that these values will constitute worst-case scenario values or peak values. The actual power draw during normal operation is significantly less.

Table 6.2: Power Draw of Various Components

Component	**Power Draw [W]**	**Current Draw @ VDC [V]**
Siemens IPC427E	96 W	4A @ 24 VDC
Siemens 1512SP	24.55 W	1.232 @ 24 VDC
Siemens IoT 2040 (×2)	67.2 W	2.8 @ 24 VDC
SICK FX-CPU320002	7.1 W	0.3A @ 24 VDC
PFsense Router	60 W	2.5A @ 24 VDC
SICK NAV350-3232	36 W	1.5A @ 24 VDC
Scalance W746-1	6 W	0.23 A @ 24 VDC
Sick S300 Mini Remote 2011EA (×2)	10.6 W	0.44 A @ 24 VDC
Scalance XB005G Switch (×2)	21 W	0.88 A @ 24 VDC
Festo CMMS-ST Drive (×2) [1]	960 W	10 A @ 48 VDC
Siemens V90 Drive (×2) [1]	835.74 W	3.8 A @ 220 VAC
Traco Power TSP-BCMU360	16.3 W	1.2A @ 13.6 VDC
Battery Unit Eject Motor[1]	72 W	6 A @ 12 VDC
Cooling Fans (×4)	14.4 W	0.6A @ 12 VDC
XL4005 10V Analog Source	0.4 W	0.04 A @ 10 VDC
Mecer A5026 Screen	242 W	1.1 A @ 220 VAC
TOTAL	2469.29 W	51.44 A @ 48 VDC

Since all pilot lights (indicators) are directly connected to the s7-1500 PLC, they will

[1] The power listed here will be the maximum power required by the system and not the maximum power the motor can produce.

draw their power from the PLC. Thus, the current required for these devices is included in the PLC's power bill for table 6.2.

Although the battery eject motor is included in the power bill listed in table 6.2, it can be removed when looking at the power bill from the inverter/ main battery banks perspective. Removal is possible because this system will source its current directly from the 12 VDC auxiliary battery bank. The auxiliary battery bank is charged using the Traco Power TSP-BCMU360 from the main battery bank. The Traco Power TSP-BCMU360 charges at a current of 1.2 A at 13.6 VDC. Thus, the current drawn by the 'battery unit eject motor' can be included in the TSP-BCMU360's power bill. The same is true for the second Siemens IoT 2040, which sources its power directly from the 48 VDC battery bank and does not use the inverter. Thus the power requirements for the inverter will be reduced to **2 363.69 W**.

6.2.1.a Main Battery Bank

The main battery bank needs to produce 48V DC to power the AC inverter; as such, it was decided to use four deep-cycle lead-acid batteries. These batteries, made by "First National Battery", are rated at 12 VDC each and have a capacity of 102 Ah. The four batteries were wired in series to produce the desired 48 VDC with a capacity of 102 Ah. The batteries themselves have an M10 post for both the positive and negative terminal and were connected with a flexible copper bush bar as illustrated in figure 5.70. A bus bar was used in place of cable and ring lugs to reduce the resistance between the batteries, especially under high current draw.

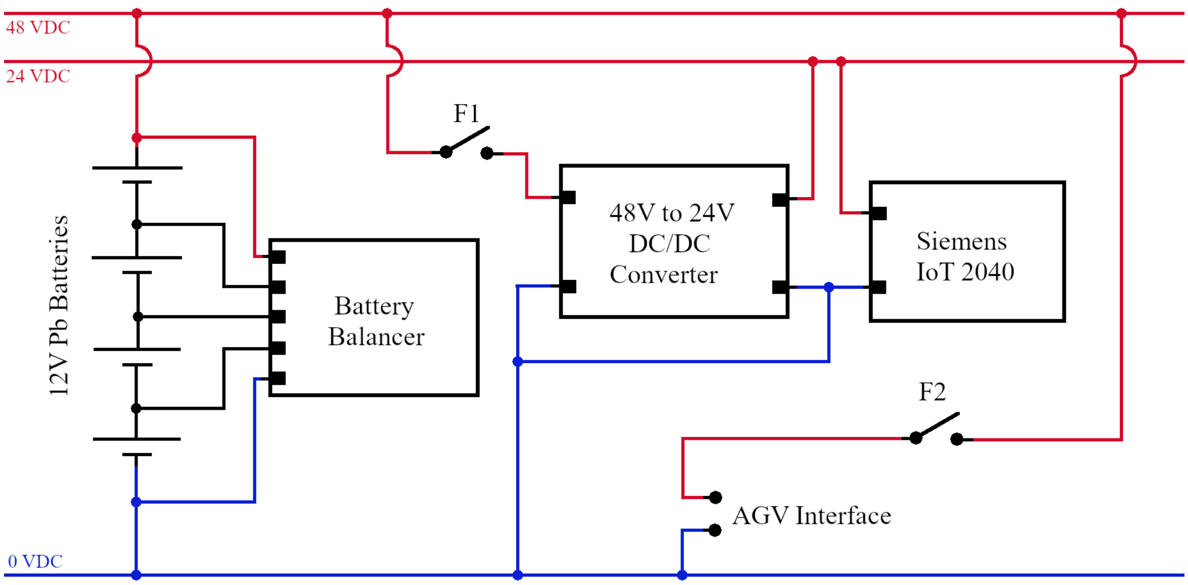


Figure 6.2: Wiring Diagram of Battery Unit

In figure 6.2 F1 and F2 are circuit breakers used to protect the wiring of the AGV from over-current. F1 has a current rating of 63A, and F2 has a current rating of 2A. BT1 - BT4 are the four deep cycle batteries from FNB (SMF100). The BAT BAL is a Banggod.com battery equalizer rated for a 48 VDC battery pack. The components used in this system are detailed in table 6.3.

The Siemens IoT 2040 was included to perform battery management and is the core of the BMS system. Since the IoT 2040 needs 24 VDC to operate and the battery bank is 48 VDC, a DC to DC converter is needed to step down the voltage of the main battery bank.

Table 6.3: Main Battery Unit Electrical Components

Part Chosen	**Manufacturer**	**Component Description**
DC/DC Converter	Unknown	48V to 24V DC/DC Converter
Siemens IoT2040	Siemens	IoT Controller/Gateway
BE48	Banggood.com	48V Battery Equalizer
FNB SMF100	First National Battery	102Ah 12V lead acid battery
5SJ71 Circuit Breaker	Siemens	63A Class C Circuit Breaker
MN 110Z Circuit Breaker	Hager	2A Class C Circuit Breaker

6.2.1.b Battery Management System

Battery management was not within the scope of the research and, as such, was given as an undergraduate project. This project entailed using Columb counting biased with a voltage measurement to determine the SoC of the battery bank. The brief for this project can be found in appendix J. Unfortunately, the final design for this BMS was unstable for this task and, therefore, will not be included in this report. Instead of a full BMS, a cheap battery equalizer from banggood.com was included (BE48), see figure 6.2.

6.2.2 Auxiliary Power Unit

The auxiliary power unit consists of two sub-sections: the auxiliary battery bank and the DC UPS system. Its purpose is to provide power to the AGV's control systems when the main battery bank is removed during a battery exchange operation. Since the control systems will have to change over from the main battery bank unit to the auxiliary battery bank unit seamlessly, it is imperative that the changeover act as an uninterrupted power supply (UPS).

The main control system must be active when the AGV exchanges batteries for the following reasons. Firstly, the AGV must continually communicate to a higher control network (Fleet Management System). Secondly, the AGV takes part in the main battery exchange process by locking the battery bank in place, moving the battery bank and closing a solid-state relay (this feature has depreciated). Finally, the AGV has an onboard Industrial PC (IPC); these systems take time to reboot (5 to 10 minutes) and can easily be damaged if power is suddenly cut [62].

6.2.2.a Auxiliary Battery Bank

The auxiliary battery bank has a nominal voltage of 12 VDC and consists of three batteries rated at 7 Ah each. Since the battery bank is rated at 12 VDC, the total capacity of the battery will be 21 Ah since the batteries are linked in a parallel configuration, see figure 6.3. The batteries used were sourced from Rubicon, with the part name RITAR RT1270B.

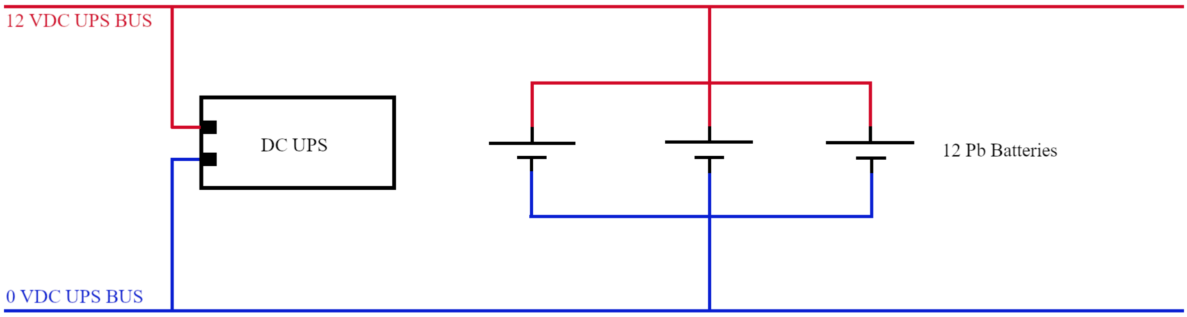


Figure 6.3: Parallel Wiring of Auxiliary Battery Bank

6.2.2.b DC Uninterrupted Power Supply

For the DC UPS, there were two options, a custom-built solution and an off-the self-solution. The off-the-shelf solution was chosen for the final AGV. This solution consisted of a TRACOPOWER TSP-BCMU360. A custom-built solution was fully developed as, at the time of development, the existence of the TRACOPOWER unit was unknown to the author. Implementation of the off-the-shelf unit is illustrated in figure 6.4.

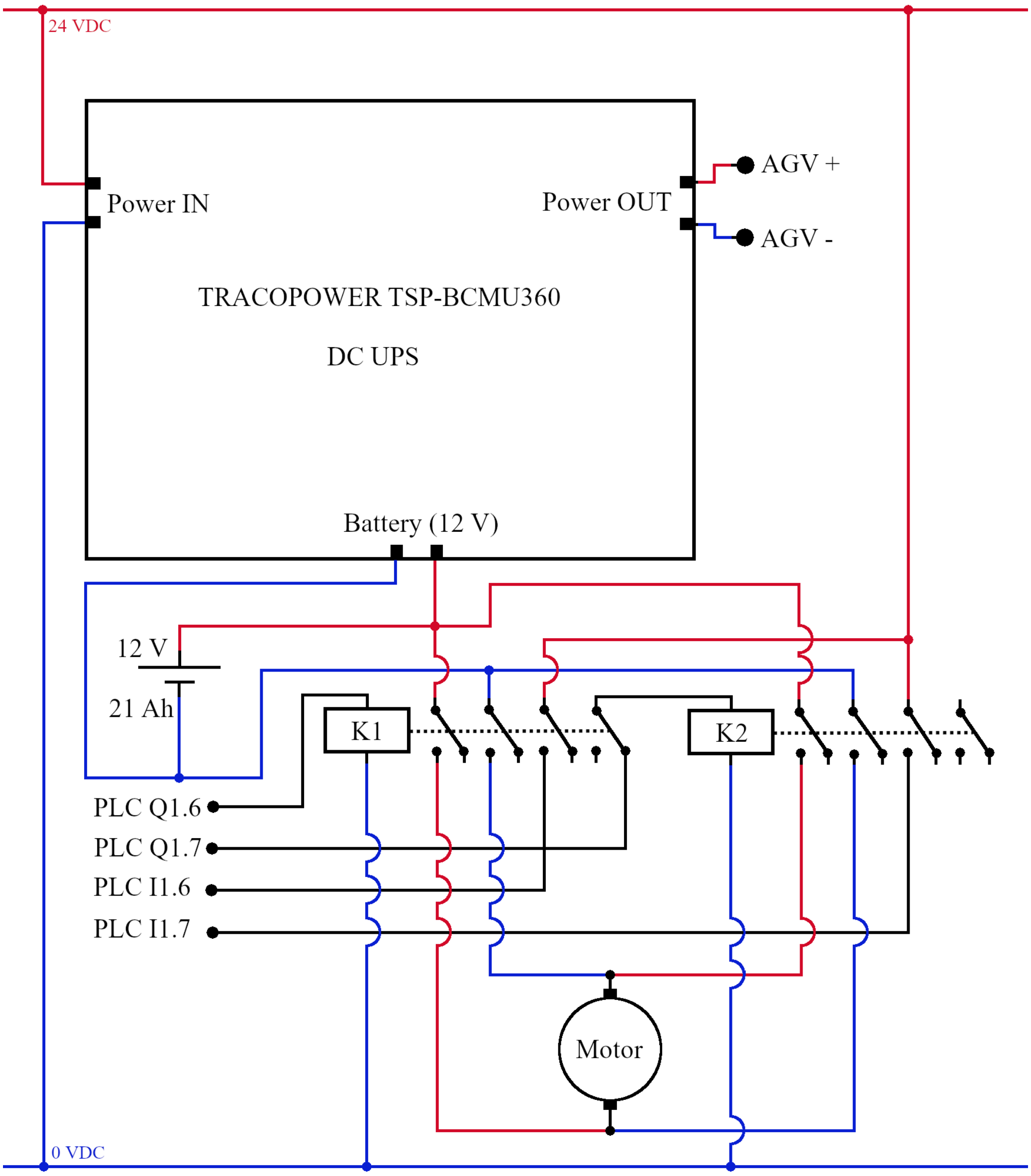


Figure 6.4: Off-The-Shelf Auxiliary PSU Wiring

In figure 6.4 the 24 VDC is provided by a standard AC to DC power source (PSU) that converts the main inverter's 220 VAC to 24 VDC. The TSP-BCMU360 sits between the AC to DC power supply and the load equipment. During regular operation, the UPS functions in "by-pass" mode. This mode passes current from the AC to DC power supply directly to the load equipment while simultaneously charging the auxiliary battery bank (BT1). When power to the input is cut, the UPS will source its current from the auxiliary battery bank.

Details of the custom built UPS and its implementation are included in the thesis

Macfarlane [63] and is included in appendix K.

The bill of electrical materials can be found in table 6.4.

Table 6.4: Auxiliary Battery Unit & DC UPS Electrical Components

Part Chosen	**Manufacturer**	**Component Description**
TRACOPOWER TSP-BCMU360	TRACOPOWER	DC UPS System
RITAR RT1270B	Rubicon	7Ah Alarm Battery
DPDT 16A Relay	Hager	16A Relay
Toyota Hilux ELP.WM10412	Autozone Spares	Windscreen Wiper Motor

6.2.3 Power Inverter

The main power inverter in the AGV is responsible for converting the 48 VDC of the main battery unit into a single phase 220 VAC. The inverter is necessary to run the traction motors, which require single-phase AC. The inclusion of the power inverter in the design had a secondary advantage above its necessity to power the motors. It allowed various DC voltages to be obtained relatively cost-effectively through AC to DC conversion rather than DC to DC conversion. Since,all-in-all, the system required the following voltages:

- 12 VDC
- 24 VDC
- 48 VDC
- 220 VAC

Of the available power inverters on the market, there are three types, namely square wave, modified sine wave, and SPWM (Sinewave Pulse Width Modulation) [64]. Square wave PWM inverters are the simplest by far, switching on and off the output at the line frequency. These inverters are extremely primitive and are often not suitable for anything but the most basic applications, such as incandescent bulbs.

Modified sine wave inverters are common in the small inverter market ($<$ 1 kW) due to their cost-effectiveness. They operate similarly to square wave inverters but have a dead state (off-state) at the zero volt point between the negative and positive peaks; this helps to eliminate the harmonics 3, 9, and 27. Modified sine wave inverters can be used for most applications.

SPWM inverters make use of PWM to very closely approximate a true sine wave and are suitable for all sine wave applications [64].

From table 6.2, it can be seen that a total of 2 363.69 W (with battery eject pinion motor and the second IoT 2040 removed) will be needed; thus, the inverter should be sized to meet this minimum. For this AGV, a 4 kW SPWM sine wave inverter was used; specifically, the RCT Axpert 5K MKS, see figure 6.5. This inverter was purchased from Rubicon.

This inverter has the added advantage of having a built-in battery charger that can produce a charging current of up to 60A (Hence the choice of the 63A breaker in table 6.3). Using an inverter with an included charger allows the AGV to be charged (if a battery unit exchange is not desired) directly from any standard SANS 164-1 or SANS 164-2 plug socket.

Figure 6.5: RCT Axpert 5K MKS Inverter

A summary of the electrical requirements of the inverter can be found in table 6.5.

Table 6.5: Inverter Electrical Requirements

Specification	**Value**
Required Wattage	1 907 W
48 VDC Input Current	39.73 A
220 VAC Output Current	8.67 A

The current required at the input and output of the inverter to match the required load was calculated using equation 6.1:

$$I = \frac{P}{V} \tag{6.1}$$

I	=	Electrical Current	A
P	=	Electrical Power	W
V	=	Electrical Voltage	V

Since the required nominal power of the system is 2 363.69W, and the inverter is rated at 4kW, the inverter is oversized by 41%. Choosing this size inverter was done for three reasons. The first is to accommodate any surges expected during startup (a surge current can often be 2x the rated current). The second was to account for unforeseen extra power demands or future expansions (especially since a PTO was included via a SANS 164-1 socket). Finally, the third reason was that inverters in this range and form factor (at the time of purchase circa 2017) were only available in 1kW, 2kW, 4kW and 6kW from the preferred supplier Rubicon.

6.2.4 Power Supplies

As mentioned in section 6.2.3, four primary voltages will be required: 12 VDC, 24 VDC, 48 VDC and 220 VAC. An additional 10 VDC bus is needed as a supply for any analogue sensors. This additional voltage is generated using a cheap XL4005 Banggood DC-DC converter to convert the 12 VDC bus to 10 VDC as the current draw was minimal.

6.2.4.a 12 VDC Power Supply

The 12 VDC bus is used to supply power to the cooling fans and as a supply source for the DC-DC converter. Due to the low current requirements, this supply could be relatively small. The power cost is listed in table 6.6.

Table 6.6: 12 *VDC* Power Requirements

Component	**Power Draw [W]**	**Current Draw @ 12V DC [V]**
Cooling Fans (×4)	14.4 W	1.2 A
XL4005 10V Analog Source	0.4 W	0.034 A
TOTAL	14.8 W	1.23 A

Since the total maximum power required from the 12 *VDC* bus will be 54 W, a Siemens LOGO! Power 6EP1322-1SH03 supply was chosen. This supply can provide a current of 4.5 A at 12 *VDC*. The 12 *VDC* power bus has 73.3% free capacity or can provide an additional 3.27 A before saturation. This is illustrated in figure 6.6.

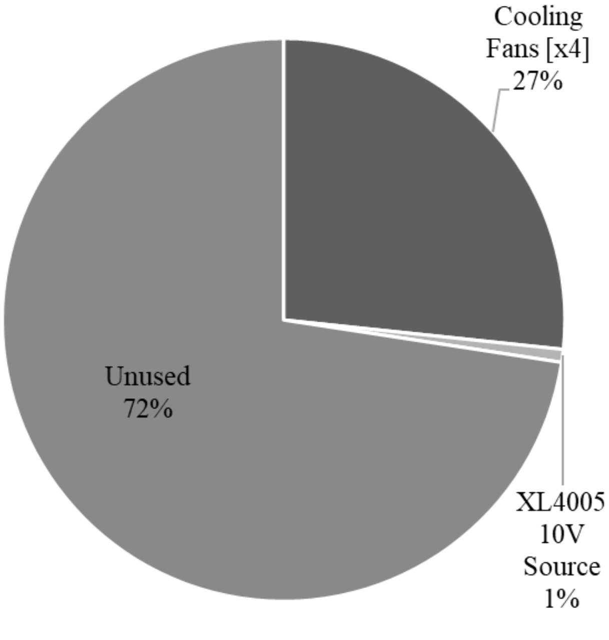


Figure 6.6: 12 VDC Power Consumption Pie Chart

6.2.4.b 24 VDC Power Supply

The 24 *VDC* is the primary voltage for the AGV's various control systems; this includes the IPC, PLCs and various indicators, light and actuators. This voltage was chosen since it is the standard voltage used in industrial controls and applications.

The AGV has two discrete 24 VDC power supplies, one rated at 10 A and one rated at 4 A. The 10 A power supply is a Meanwell DRP-240-24, while the 4 A power supply is a Siemens LOGO! 6EP1332-1SH52. Two separate power supplies were used rather than a single larger one since the electromagnetic brakes of the traction motors must have an isolated PSU due to interference concerns [65].

The power cost for the 24 VDC system is listed in table 6.7 and table 6.8.

Table 6.7: 24 VDC Power Requirements Control System (Meanwell DRP-240-24)

Component	**Power Draw [W]**	**Current Draw @ 24V DC [V]**
Siemens IPC427E	96 W	4 A
Siemens 1512SP	24.55 W	1.232 A
Siemens IoT 2040	33.6 W	1.4 A
Siemens ET200	5.86 W	0.25 A
SICK FX-CPU320002	7.1 W	0.3 A
PFsense Router	60 W	2.5 A
SICK NAV350-3232	36 W	1.5 A
Scalance W746-1	6 W	0.23 A
Sick S300 Mini Remote 2011EA (×2)	10.6 W	0.44 A
Scalance XB005G Switch (×2)	21 W	0.88 A
Traco Power TSP-BCMU360 [2]	16.3 W	0.68 A
TOTAL	317.01 W	13.41 A

According to table 6.7 the power required from the 10A PSU is 13.41 A. Essentially, the power supply was overloaded by 32 %. However, this current draw was not realistic during testing, and the system performed fine. The author hypothesises that the current draw is inflated beyond what is realistically drawn by the two IPCs in the system, the Siemens IPC427E and PFsense Router, which give absolute maximum current draw rather than typical operational current draws. Figure 6.7 graphically illustrates the power usage by each device on the 24 VDC bus.

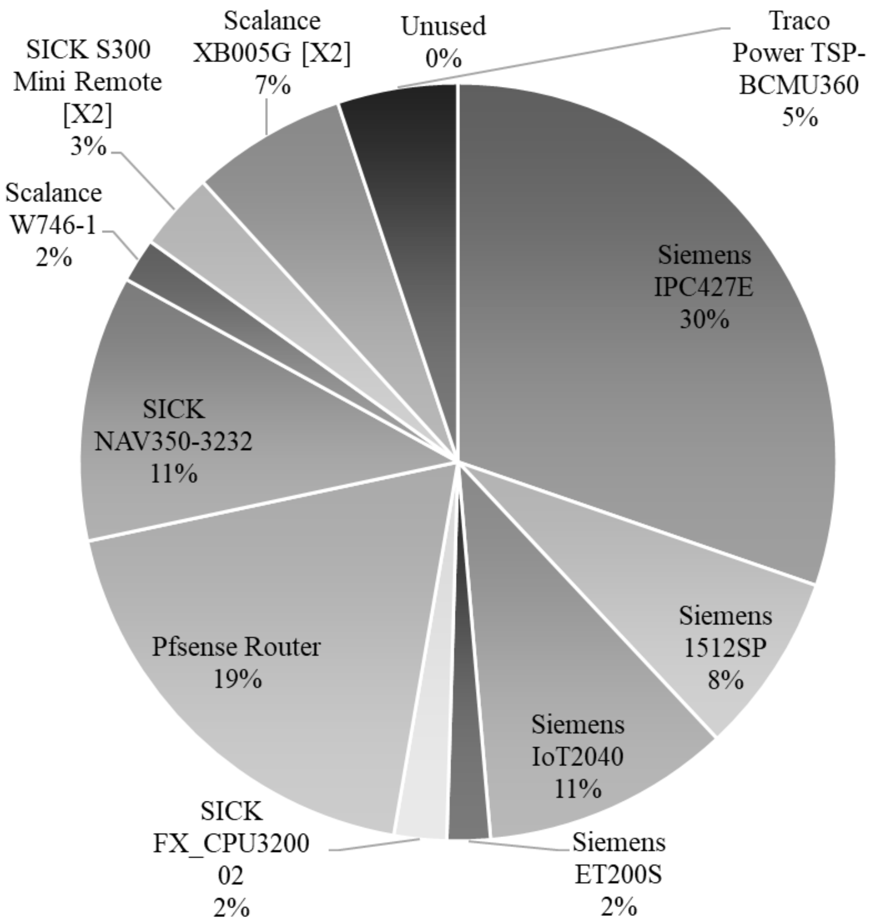


Figure 6.7: 24 VDC Power Consumption Pie Chart

Table 6.8: 24 VDC Power Requirements Parking Brakes (Siemens LOGO! 6EP1332-1SH52)

Component	**Power Draw [W]**	**Current Draw @ 24V DC [V]**
Siemens V90 Drive Brake (×2)	Unknown	Unknown
TOTAL	Unknown	Unknown

As stated in table 6.8, the exact power requirements for the holding brakes for the two traction motors were not specified in the application notes for the V90 drive. Thus an

[2] In table 6.2 the power requirements of the DC ups are listed at 13.6 VDC, as this is the charging current of the UPS battery. Since it acquires its power from the 24 VDC bus, it is appropriate to include it in table 6.7 while adjusting the current draw for 24 VDC.

"on-hand" 4A PSU was used (Siemens LOGO! 6EP1332-1SH52), which seems sufficient for the task.

6.2.4.c 48 VDC Power Supply

The 24 VDC power bus is responsible for powering the two Festo stepper steering motors, rated at a nominal 48 VDC. Although the main battery bank is rated at a nominal 48 VDC, it was decided by the author that sourcing the required power directly from the battery bank would be unwise. This decision was made since lead-acid batteries are not stable at their nominal voltage, but rather their voltage varies according to their state of charge (SoC). For a 48 VDC AGM battery bank, the voltage can vary from a minimum of 47.2 VDC (25% SoC) to a maximum of 51.2 VDC (100% SoC). It would be even worse if the battery were being charged (say through optimistic charging); under these conditions, the battery banks voltage could be as high as 58 VDC [66]. This voltage range could potentially damage the stepper drive responsible for controlling the steering motors. Thus a regulated 48 VDC bus was generated from the 220 VAC bus.

Since the stepper motors require a relatively large current, it was decided to place each steering motor on its own 48 VDC power bus. This decision required two separate discrete power supplies. The power supplies used for these motors were two Meanwell SDR-480-48 power supplies, each capable of producing 10 A. The power cost of each 48 VDC power bus is listed in table 6.9.

Table 6.9: 48 VDC Power Requirements

Component	**Power Draw [W]**	**Current Draw @ 48V DC [V]**
Festo CMMS-ST Drive	480 W	10 A
TOTAL	480 W	10 A

6.2.4.d 220 VAC Power Supply

There are very few components on the AGV that use the 220 VAC bus directly; instead, most components draw their current through one of the AC to DC power supplies. The only two systems that make use of the single-phase AC directly are the main Siemens traction motors and the LCD screen used as a SCADA and management screen for the Siemens IPC427E PC. The power cost for the 220 VAC power bus is listed in table

6.10 and table 6.11. Table 6.10 includes the expected maximum current draw from all AGV components when operated normally. Table 6.11 lists the power required on the 220 VAC bus if all AC to DC PSUs were saturated; this is not normal operating conditions.

Table 6.10: 220 VAC Power Requirements

Component	**Power Draw [W]**	**Current Draw @ 220V AC [V]**
Siemens V90 Drive (×2)	835.74 W	3.8 A
Mecer A5026 LCD Screen	242 W	1.1 A
LOGO! power 6EP1322-1SH03 (12 VDC PSU)	14.8 W	0.068 A
Meanwell DRP-240-24 (24 VDC PSU)[3]	240 W	1.09 A
LOGO! power 6EP1332-1SH52 (24 VDC PSU) [4]	24 W	0.109 A
Meanwell SDR-480-48 (×2)	960 W	4.36 A
TOTAL	2 316.54 W	10.53 A

Table 6.11: 220 VAC Fully Saturated Power Requirements

Component	**Power Draw [W]**	**Current Draw @ 220V AC [V]**
Siemens V90 Drive (×2)	835.74 W	3.8 A
Mecer A5026 LCD Screen	242 W	1.1 A
LOGO! power 6EP1322-1SH03 (12 VDC PSU)	54 W	0.245 A
Meanwell DRP-240-24	240 W	1.09A
LOGO! power 6EP1332-1SH52 (24 VDC PSU)	96 W	0.436 A
Meanwell SDR-480-48 (×2)	960 W	4.36 A
TOTAL	2 427.74 W	11.031 A

[3] The power supply is assumed to be at saturation see section 6.2.4.b

[4] It is assumed that at worst case scenario each of the traction motor's parking/ holding brakes draw 500 mA

The power drawn from the 220 VAC bus, when all the DC PSU's are saturated is illustrated in figure 6.8.

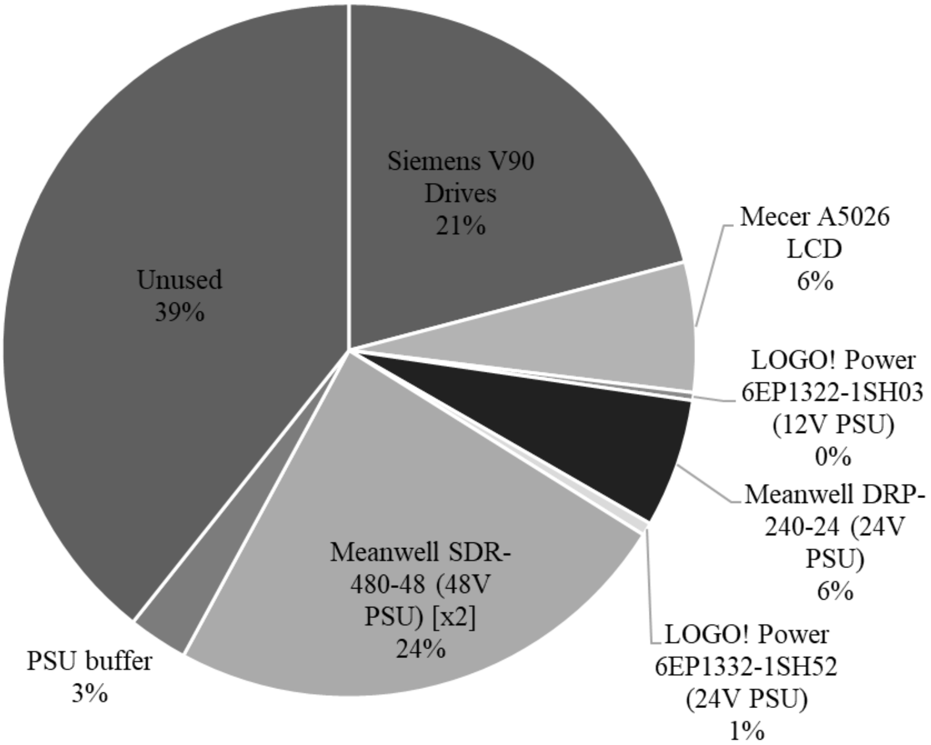


Figure 6.8: 220 VAC Power Consumption Pie Chart

In figure 6.8, the difference between the total nominal power draw (table 6.10) and the maximum potential power draw (table 6.11) is listed as "PSU buffer". The remaining "unused" power is the capacity that the power inverter can produce but is not consumed in the AGV's current configuration and, therefore, can be used via the PTO.

6.2.4.e 48 VDC to 24 VDC DC/DC converter

Included in the 24 VDC system is the second Siemens IoT 2040, which can be found in the Main Battery Unit, see section 5.9.1. This controller does not take power directly from the main 24 VDC bus but rather from a DC/DC converter found in the Main Battery Unit. The chosen DC/DC converter is a CALEX 48S24.6HCM, which outputs 24 VDC at a maximum 6.26 A current. This DC/DC converter can convert any DC voltage between 18 VDC and 75 VDC to a stable 24 VDC, making it ideal for use with a lead-acid battery bank (see section6.2.4.c for explanation). The current and power draw of the Siemens IoT 2040 is listed in table 6.7.

6.3 Control System

The control system of the AGV refers to the components that are responsible for performing the high-level navigation operations as well as the components that perform the kinematic calculations (which are outlined in chapter 8. A network diagram of the AGV's control system is illustrated in figure 6.9.

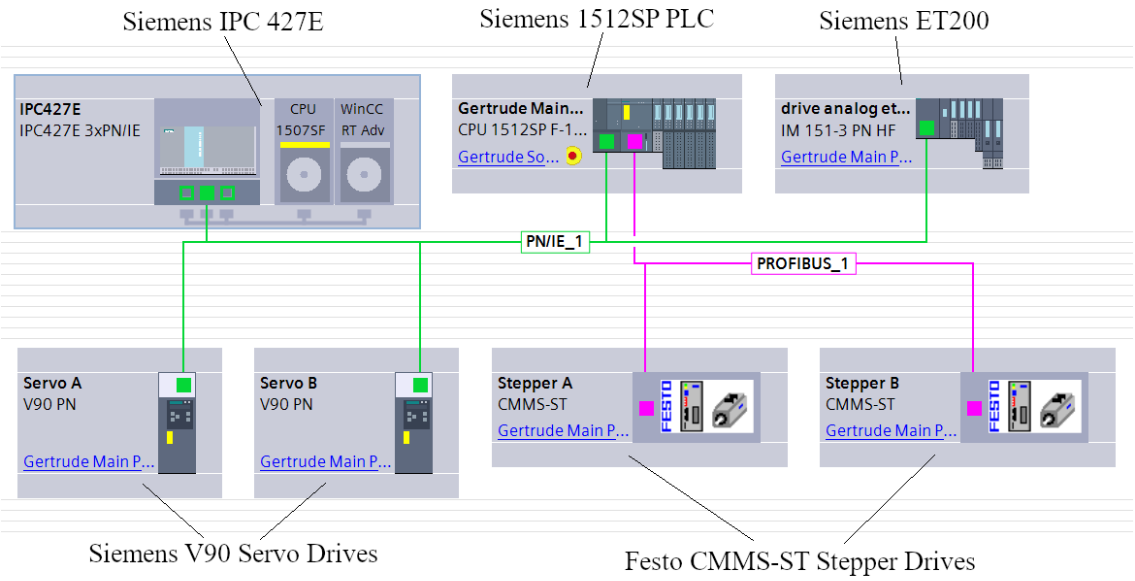


Figure 6.9: Network Diagram of the AGV's Control System

6.3.1 General Control System

The "general control system" constists of a Siemens IPC427E industrial computer and a Siemens 1512SP PLC. These two devices work in tandem to form the backbone of the AGV's control system.

6.3.1.a IPC

The IPC (Siemens IPC427E) is a proprietary x86 computer made by Siemens. The system makes use of a 6th generation Intel i5-6442EQ CPU that can boost to 2.7 GHz (nominal frequency is 1.9 GHz); this processor is a quad-core processor lacking hyper-threading [67]. The host operating system of this system is a proprietary Siemens OS called "Simatic s-7-1500 Software Controller", which uses one of the four cores in the i5 CPU and a negligible amount of RAM to perform two tasks. Firstly the Software Controller acts as a fully SIL (Safety Integrated Level) compliant PLC. Secondly, the controller acts as a hypervisor. The hypervisor hosts a full install of Windows 7 embedded, containing WinCC SCADA. This VM uses the remaining three cores and the majority of the RAM [68]. A Debian based Linux VM was later implemented that took over two of the windows 7 cores and half of the RAM to run ROS. A technical drawing of the IPC is shown in figure 6.10.

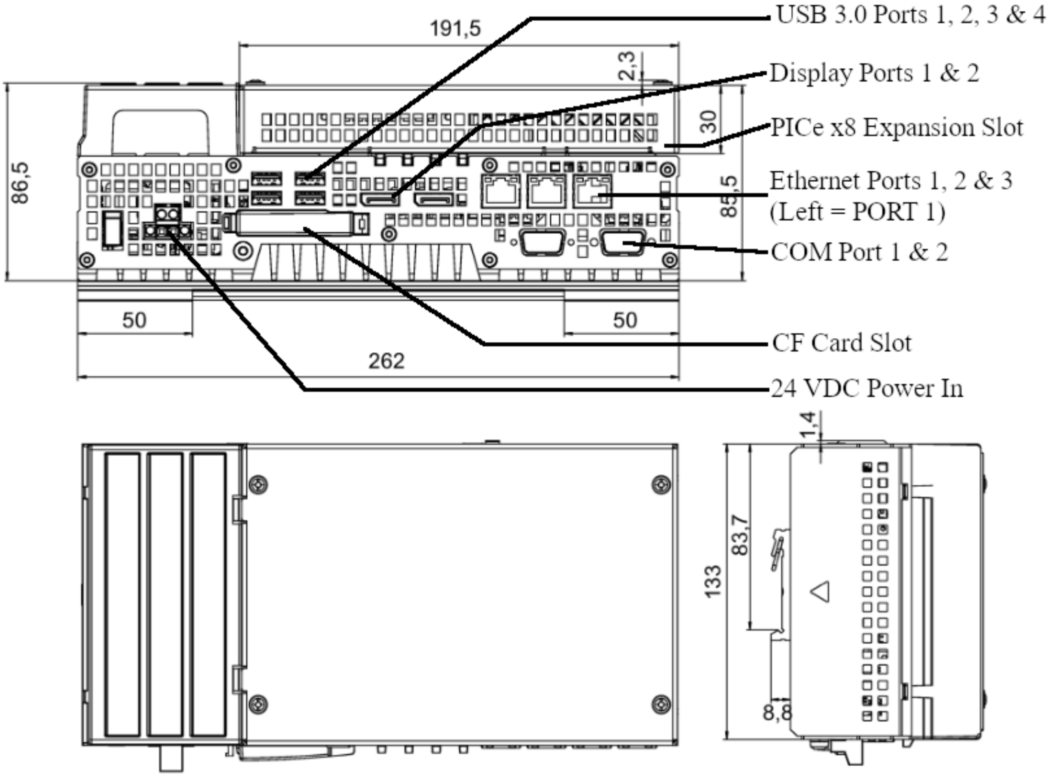


Figure 6.10: Siemens Simantic IPC 427E Technical Drawing[5]

The technical specs for the IPC 427E vary depending on customer selection. The hardware specifications for the specific IPC used in the research are given in table 6.12.

[5] Image adpated from Siemens [68]

Table 6.12: Technical Specifications for IPC 472E

Specification	**Value**
Processor	Intel I5-6442EQ (6MB cache, 2.7GHz boost)
RAM	x1 16 GB DDR4-SDRAM (SODIMM)
Buffer Memory	512 kB NVRAM (Soft PLC retentivity)
Expansion	x1 PCIe x8 (physical x16 slot)
SSD	2.5” 240GB SATA
CFast (CF flash)	Unpopulated (up to 30GB) (Soft PLC data log use)
Graphics Processor	Intel HD Graphics P530
Serial	COM1 and COM2
Monitor Interface	x2 Standard Display Ports
USB	x4 USB 3.0 Ports
Ethernet	x3 10/100/1000 Mbps RJ45 Ports

The standard IPC described in table 6.12 was upgraded by the author with the following components:

- 16GB CF flash card
- Nvidia Quadro 400 GPU
- Display Port to VGA active converter

These upgrades were done as after-market additions for two reasons, either the addition was not available as an option from Siemens, or the cost of pre-implementation was exorbitant to the point that self-installation was justifiable.

The additional CF card can be used for data logging on the PLC or as an additional drive for the windows 7 VM. The IPC did not come with a CF card installed as pre-implementation by siemens was excessively expensive. Instead, the author included a generic 16 GB card.

The Nvidia Quadro 400 was added by the author to validate the possibility of adding a GPU to the IPC 427E by checking if the PCIe expansion slot had been software locked to vendor-approved cards and what the rationale behind the 8W power limited [68] for the PCIe x4 slot was. The Quadro 400, by today’s standards (circa 2020), is woefully underpowered compared to contemporary hardware, being produced in 2011. However, it was the only single-slot card the author had on hand and had a trivial power draw of 35W (by GPU standards). It also did not require an external Molex connector for additional power (as most contemporary GPUs do). The added GPU performed well without issue. The functioning GPU confirmed that Siemens had not software locked the PCIe expansion slot and that the 8W limitation was not an electrical limitation but rather a thermal one as the IPC 427E is passively cooled. The author suspected this power limitation to be a thermal one as the PCIe standard specifies that for a x4 slot, a minimum power availability of 25W should be provided [69]. It was also hinted at this in Industrial PC SIMATIC IPC427E [68], on page 89. Since the GPU has a fan for cooling, the metallic cover for the expansion slot can be left off. The aggressive cooling provided by the component box fans also aided in keeping the GPU thermals under control. Thus, thermal limitations are not an issue. Leaving the metallic cover off the expansion slot has two other advantages. It allows for two-slot high PCIe cards to be used on the system. It also allows an additional PSU to supply a high power card via an external Molex connector.

The addition of a GPU to the IPC stemmed from the desire to run ROS on the AGV in the future. Since processing navigational data in SLAM or SLAM-like algorithms becomes nearly impossible without a GPU. If the inclusion of a GPU on the IPC were not possible, the navigational processing would have had to be done on a remote server connected to the AGV via WIFI or by using an Intel "Neural Compute Stick 2" unit connected via USB (https://software.intel.com/content/www/us/en/develop/hardware/neural-compute-stick.html). When ROS is implemented in the future on the AGV, a single slot GPU should be used with an external Molex connected PSU (powered by the 220 VAC bus). Using an externally powered GPU will take the strain off the 24 VDC power supply (that powers the IPC 427E) and the PCIe power delivery of the IPC 427E.

The purpose of the Display Port to VGA active converter allowed the author to use an already on-hand Mecer A5026 screen that only had VGA and DVI inputs. The display output is done via the integrated Intel HD P530 GPU and not the PCIe mounted Nvidia Quadro 400 (The PCIe mounted GPU is reserved for computational tasks, not graphical processing). The screen has two purposes; It allows for easy access to the windows 7 embedded desktop, where the author will include backups of all necessary code, CAD and documentation, as more often than not, this information is quickly lost as the research project is handed over to the next researcher. Secondly, it allows for management of the various components on the AGV, which make use of either a webpage or windows based software tool; this includes the PFsense router and RCT Axpert 5K MKS Inverter.

6.3.1.b PLC

A Siemens Simantic s7-1512SP PLC is used to control the I/O operations (switches, LEDs, indicators, etc.) and the drives on the two bus networks (Profinet and Profibus). This PLC adheres to the Siemens ET200SP form factor, making it very compact compared to other Siemens alternatives. However, it is computationally weaker than a traditional s7-1500, though this can be worked around by assigning computationally expensive tasks to the software PLC on the IPC. The layout of the s7-1512SP is shown in figure 6.11, further information can be found in appendix L.

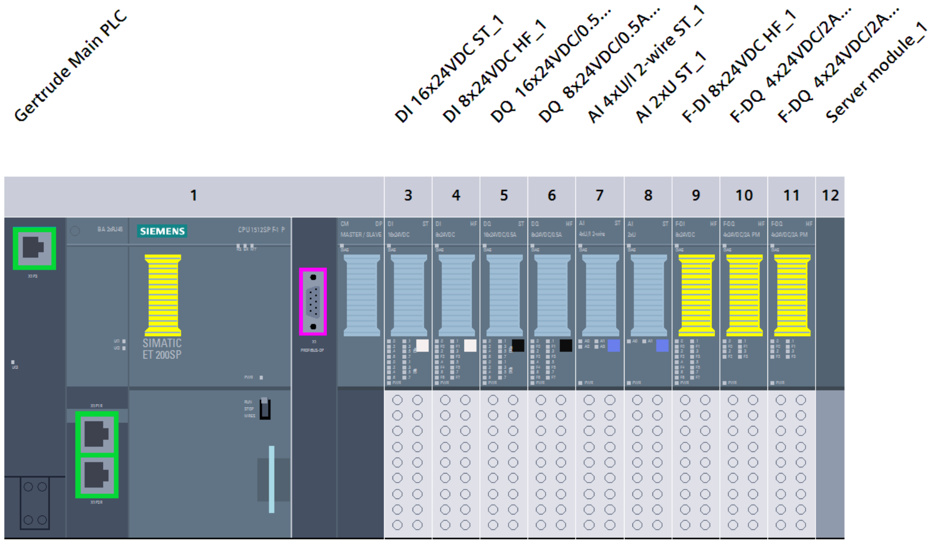


Figure 6.11: Actual Configuration of the s7-1512SP PLC

Table 6.13 details the purpose of the components illustrated in figure 6.11, note that not all I/O are necessarily used, a good portion is reserved for future use.

Table 6.13: Table of s7-1500 PLC Components

Slot	Type	Description
1	Gertrude Main PLC (CPU 1512SP F-1 PN)	Main s7-1500 Safety Central Processing Unit (CPU)
2	DP interface	Profibus communication processor
3	DI 16x24VDC ST	16 channel digital input card
4	DI 8x24VDC HF	8 channel digital input card
5	DQ 16x24VDC/0.5A ST	16 channel digital output card
6	DQ 8x24VDC/0.5A HF	8 channel high feature output card
7	AI 4xU/I 2-wire ST	4 channel analogue input card (16 bit ADC)
8	AI 2xU ST	2 channel analogue input card (16 bit ADC)
9	F-DI 8x24VDC HF	8 channel safety digital input card
10	F-DQ 4x24VDC/2A PM HF	4 channel safety digital output card
11	F-DQ 4x24VDC/2A PM HF	4 channel safety digital output card
12	Server module	Rack bus terminator

In table 6.13, reference is made to standard (ST) and high feature cards (HF). The difference between these two families of signal modules is how their commons function. Standard cards share a common (usually a global power bus); in the case of input cards, their shared common is usually the 24 VDC bus; for output cards, it is the 24 VDC GND bus. High feature cards provide a separate common per channel, where each input has its own 24 VDC source, and each output has its own GND sink (This is useful for error detection by the PLC)

The assignment of the I/O for the PLC (channel addressing and channel functions) is listed appendix M.

The s7-1512SP is configured in what is known as i-device mode. In i-device mode, the PLC acts as a slave device ("IO Device" in Siemens nomenclature), allowing it to easily communicate with another master device ("IO Controller" in Siemens nomenclature) without having to resort to TCP, UDP or ISO-on-TCP communications. The master controller in this research project is the software PLC located on the IPC427E industrial PC. The i-device functionality is critical due to the "real-time" requirement of the control system in this research project as TCP, UDP, or ISO-on-TCP communications are not real-time communication protocols while the Profinet communication between an IO Device and IO Controller is. This real-time communication (RT) has a cycle time of precisely 1 ms and an allowable jitter of 1 μs [70]. The real-time behaviour also allows two mechanisms to function correctly, the first is Profidrive, and the second is Profisafe. Prodrive is necessary to allow real-time speed and torque control of the drives, while Profisafe must allow safety signals to be passed between devices over a network (i.e. a safety compliment signal can be passed from the E-stops attached to PLC's physical inputs to the software controller).

6.3.1.c ET200

The Siemens ET200 is used for digitising the analogue values from the two Festo stepper drives. The analogue values generated by the stepper drives represent each drive unit's steering angle. Originally these analogue signals were attached to one of the analogue input cards of the PLC. However, it was found that these values were volatile as the signal wires had to run near the main power inverter and, as such, picked up interference. These signals were not simply sent over the Profibus network directly due to an artificial limitation in the Festo firmware of the CMMS-ST drives.

The actual hardware configuration of the ET200 is illustrated in figure 6.12.

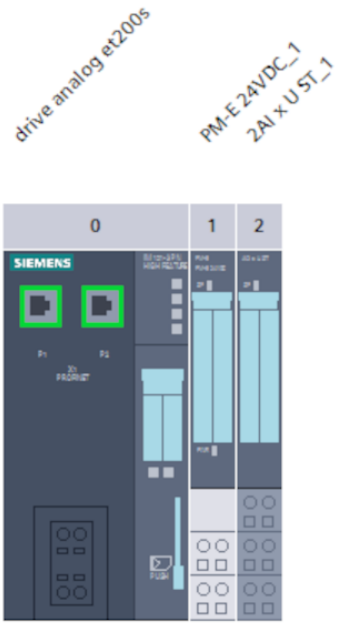


Figure 6.12: Actual Configuration of the ET200S

The purpose of each module shown in figure 6.12 is listed in table 6.14.

Table 6.14: Table of ET200S Components

Slot	Type	Description
0	Drive analog et200s	Network interface module for the ET200s
1	PM-E 24VDC	Power module for signal modules
2	2AI x U ST	Two channel analog input module

The ET200 only supports two analogue channels as listed in table 6.14. The tag-table for these channels can be found in appendix N. The ET200s is designated as a slave device to the s7-1512SP, and as such, its addresses are part of the S7-1512SP's range and not the Software PLC's address range.

6.3.2 Naviagtion

Primary navigation is done via the SICK NAV350-3232. This LiDAR uses a 905 nm laser with an angular resolution of 0.25°. The update frequency of the unit is 8 Hz and has a 360° viewing angle with an effective range of 35m @ 10% remission (10% laser light scattered) and 100m @ 90% remission. The LiDAR unit and scanning envelope are illustrated in figure 6.13.

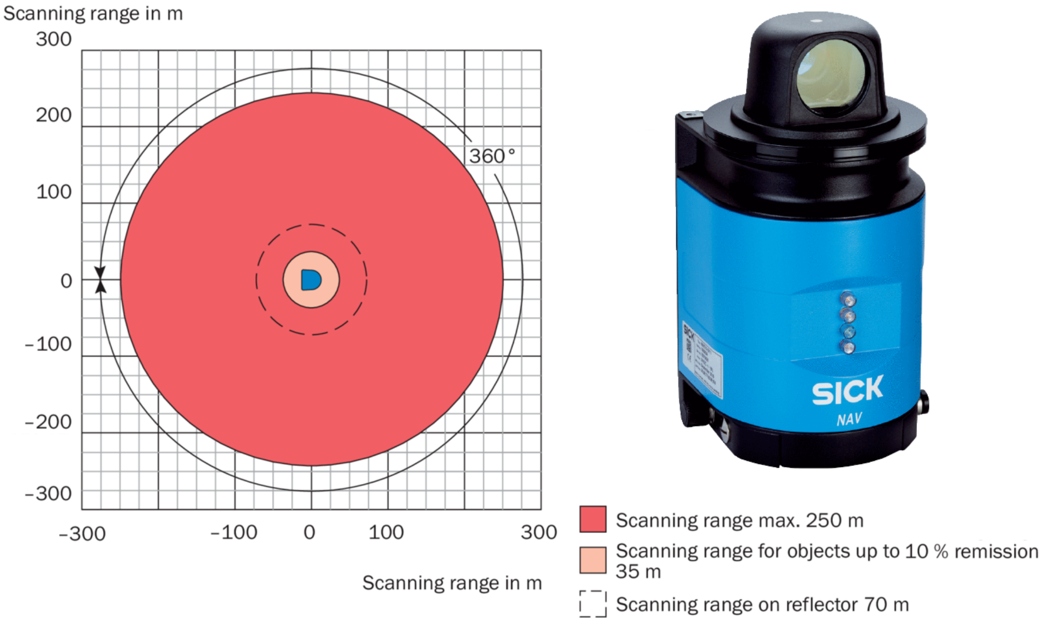


Figure 6.13: NAV350-3232 3D model & Scanning Profile[6]

Specifications for the NAV350 can be found in appendix O.

This unit is a 2D LiDAR scanner that maps the surrounding environment by creating a point cloud map. This point cloud is generated using time-of-flight calculations to measure the distance a beam of light must travel between the LiDAR unit and the object that reflects the beam to the LiDAR. An example of a point cloud is illustrated in figure 6.14.

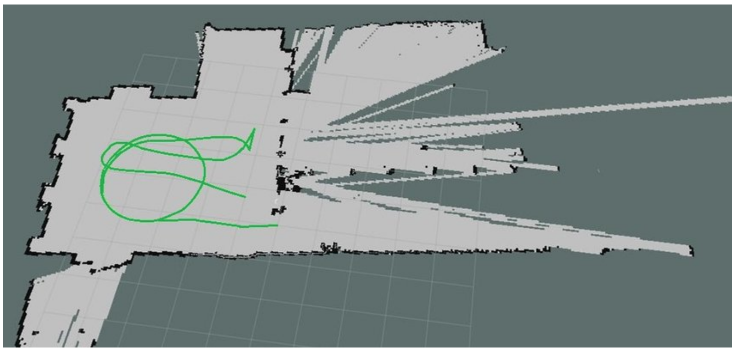


Figure 6.14: 2D Point Cloud Map of a Room [7]

The primary operating mode of the NAV350 is to utilise retro-reflector waypoints as landmarks. The position of the NAV350 is triangulated using these waypoints. The triangulated position is then passed to the high-level controller as coordinate data. The NAV350 uses telegrams on industrial ethernet to provide the AGV with its current position; this can then be compared to a "virtual path" that the AGV can follow as if it were a physical line. The high-level controller requests the AGV's position; in the case of this AGV, that would be the IPC. Position requests are made cyclically with a max stable cycle time of 125 ms. This process is illustrated graphically in figure 6.15.

[6] Image adapted from SICK [71]

[7] Image courtesy of Sokolov [72]

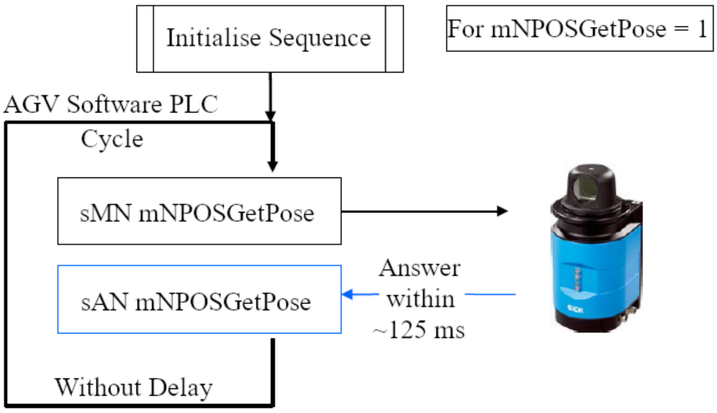


Figure 6.15: NAV350 Position Update Behaviour

6.3.3 Safety Systems

The safety system of the AGV can be activated via either of the two E-stop buttons or the light curtain safety system. When the AGV is placed in "safety mode", the STOs of the four drives (the two traction motors and two steering motors) are activated via safety channels from the s7-1500 PLC. Activating the STO input disables the drives and prevents the AGV from functioning. In order to release the AGV from the safety mode, the E-stops must be released, or the light curtain cleared, and the acknowledge must be registered using either the pendant acknowledge (Pendant SW7 in table M.1) or the red push button located at the front of the AGV (RED Pushbutton in table M.1). An analysis of the safety concerns and required conformance can be found in chapter 7.

6.3.3.a E-stop Safety System

The E-stop safety system is a manual safety system requiring the operator to press either of the two E-stop buttons. One button is located on the front of the AGV, while the second is located on the rear. These E-stop buttons use two "equivalent" channels to ensure high redundancy. "Equivalent" means each E-stop contains two electrically independent normally closed contacts, which get broken when the E-stop is pressed. In addition to providing redundancy, since the same mechanical action triggers both contacts, the trigger time between the two contacts can be used to evaluate the health of the E-stop. This type of E-stop is known as a 1oo2 system or "Read-in process signal via two channels" [73]. In the E-stop configuration used in the AGV, each E-stop received its own independent supply from the safety input card. The layout of this system is illustrated in figure 6.16.

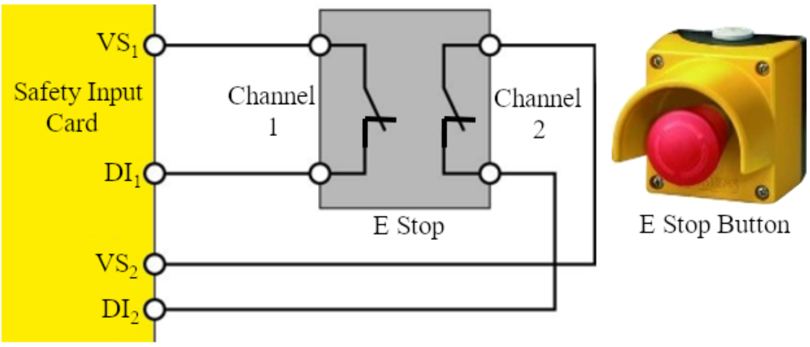


Figure 6.16: 1oo2 Equivalent E-stop Circuit[8]

Note that to use 1oo2 configuration on the Siemens F-DI 8x24VDC HF, the appropriate inputs must be paired as listed in table 6.15.

Table 6.15: Channel Pairing for 1oo2 using the F-DI 8x24VDC HF

1oo2 Pair	**F-DI Channels**	**F-DI Source**
α	Channel 0 (terminal 0)	terminal 8
	Channel 4 (terminal 4)	terminal 12
β	Channel 1 (terminal 1)	terminal 9
	Channel 5 (terminal 5)	terminal 13
γ	Channel 2 (terminal 2)	terminal 10
	Channel 6 (terminal 6)	terminal 14
δ	Channel 3 (terminal 3)	terminal 11
	Channel 7 (terminal 7)	terminal 15

[8] Image adapted from Siemens [43].

6.3.3.b Light Curtain Safety Sensor

The light curtain safety sensor prevents object collisions with the AGV. Collisions could occur if a static object is in the path of the AGV or because a moving object (such as a walking person) is on a collision course. Either way, it is necessary to avoid this collision or mitigate the possible damage/ injury should the collision be unavoidable. It was decided that the best way to do this was to bring the AGV to a complete stop by using the AGV's STO safety system. Since this light curtain barrier is responsible for ensuring the AGV operates in a "safe" manner (per the evaluation in chapter 7), it too must be safety rated.

In order to conform to the desired safety integrity level, a pair of SICK S300 Mini Remotes were used on opposite corners of the AGV. This configuration was previously discussed in the mechanical design chapter in section 5.8.2. The layout of the safety sensors is illustrated in figure 5.66 and is repeated in figure 6.17.

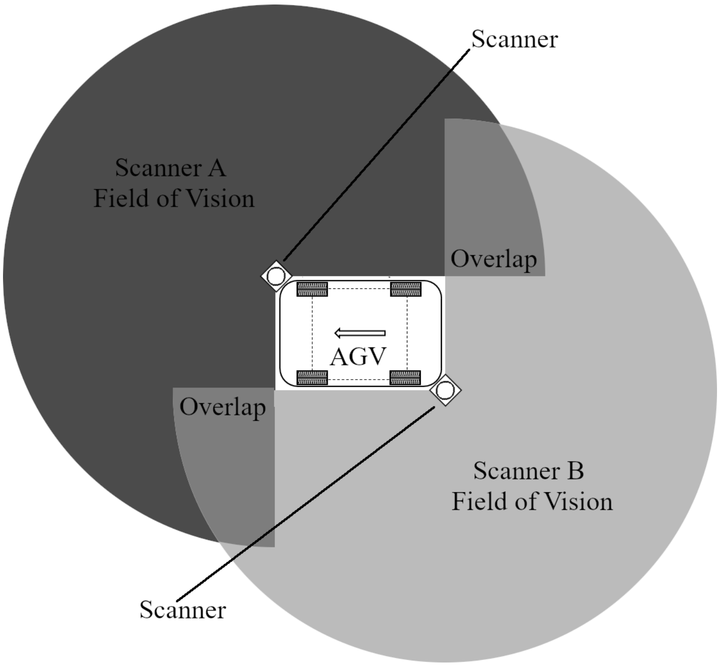


Figure 6.17: SICK S300 Mini Remote Safety Scanner Layout Repeated

The SICK S300 Mini Remote's LiDAR data is not available to the end user, however what is available is 3 zone triggers. That is to say, when an obstruction is at a set distance from the sensor it can be designated as in a certain zone. There are three zones available to the AGV, these are the protective zone, warning zone 1 and warning zone 2. When an object is detected in warning zone 2 the speed of the AGV is clamped from it maximum of 3 $^{m}/_{s}$ to 20 % of this (0.6 $^{\mathrm{m}}/_{\mathrm{s}}$). If the object enters warning zone

1 the AGV will be brought to a non-safe state stop. Finally is the object enters the closest zone to the AGV, the protective zone, the drives will be shut down and placed in a safe state using the hardware STO's on the drives. These zones are illustrated in figure 6.18.

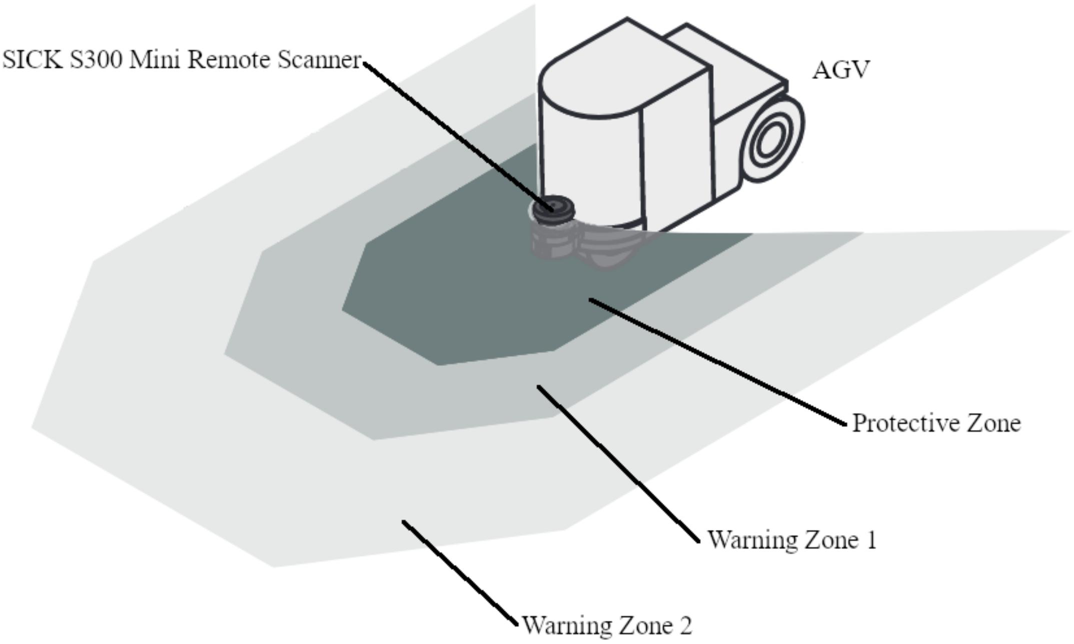


Figure 6.18: SICK S300 Mini Remote Safety Scanner Zones[9]

Note, as shown in figure 6.18, the zone boundaries do not have to be a fixed radius from the sensor but can be contoured to best suit the machine shape. This configuration is done on the sensor using the commissioning port and SICK's CDS (Configuration & Diagnostic Software) software. The generalised specifications for the SICK S300 Mini Remotes are discussed in table 6.16.

[9] Image adapted from SICK [71]

Table 6.16: Technical Specifications for SICK S300 Mini Remote

Specification	**Value**
Usage	Can only be used with SICK EFI network
Protective Zone	$<$ 3m
Warning Zones	$<$ 8m (@ 15 % reflectivity)
Distance Measuring Range	30 m
Type of Zones	Triple field set
Scanning Angle	270°
Resolution (configurable)	30 mm, 40mm, 50mm, 70mm, 150mm
Angular Resolution	0.5°
Response Time	80 ms
Safety Type	3
Safety Rating	SIL2 (IEC 61508)
	SILCL2 (EN 62061)
Safety Category	Category 3 (EN ISO 13849)
Safety Performance Level	PL d (EN ISO 13849)

As mentioned in table 6.16, the SICK S300 can only talk to other SICK devices via the EFI network. Thus, a SICK safety PLC had to be included in the AGV; the PLC used was the SICK FX3-CPU320002, which can support two EFI connections (enough for the two S300 remote minis used on the AGV). The SICK PLC was expanded with the following signal modules, a FX3-XTIO84002 I/O module and a FX0-GPNT00000 industrial ethernet gateway.

The FX3-XTIO84002 I/O module was necessary since this was the only way to pass safety-rated signals to the s7-1500 PLC since SICK does not (at the time of writing circa 2021) have a Profi-safe ethernet gateway.

Four F-DQ outputs of the s7-1200 PLC were wired to four F-DI inputs of the SICK PLC to send safety data from the s7-1200 PLC to the SICK PLC.

Four F-DQ outputs of the SICK PLC were wired to four F-DI inputs of the s7-1200 PLC to send safety data from the SICK PLC to the s7-1200 PLC.

Joining the I/O in such a manner allows four boolean safety signals to be sent in either direction. Since the safety signals sent between the devices are boolean, very little information can be conveyed besides the STO state (STO activated/ deactivated). To gather more in-depth information about the safety condition, such as which of the two scanners were triggered or the obstruction's coordinates in the laser field, a non-safety rated communication (black channel communication) is sent via industrial ethernet using the FX0-GPNT00000 industrial Ethernet gateway. The hardware configuration of the SICK PLC is shown in figure 6.19.

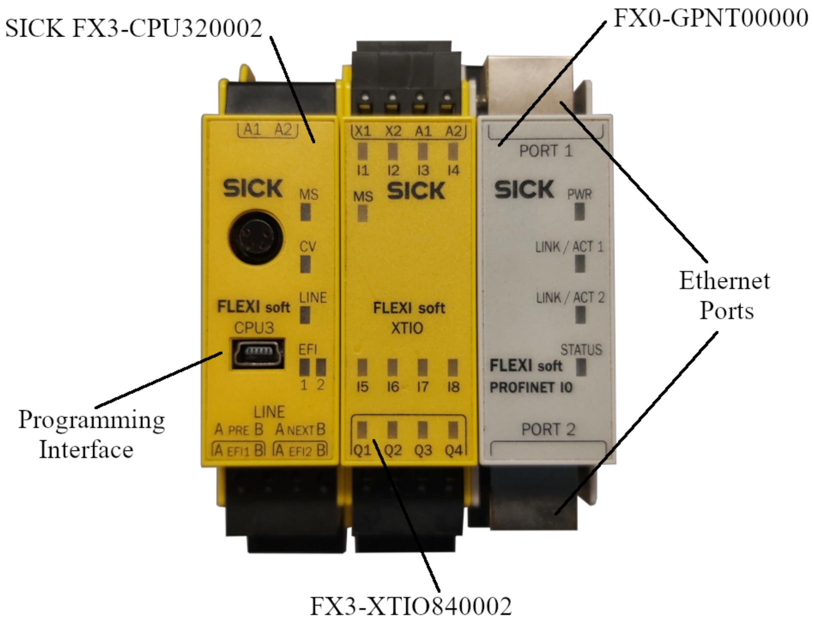


Figure 6.19: SICK Safety PLC Layout

The modules used in the SICK PLC, that are shown in figure 6.19 are listed in table 6.17.

Table 6.17: Table of SICK PLC Components

Slot	Type	Description
1	FX3-CPU320002	SICK safety CPU with 2 EFI channels
2	FX3-XTIO8480002	8 F-DI, 4 F-DQ digital safety card
3	FX0-GPNT00000	2 port industrial Ethernet gateway

The wiring between the Siemens and SICK PLCs is shown in figure 6.20

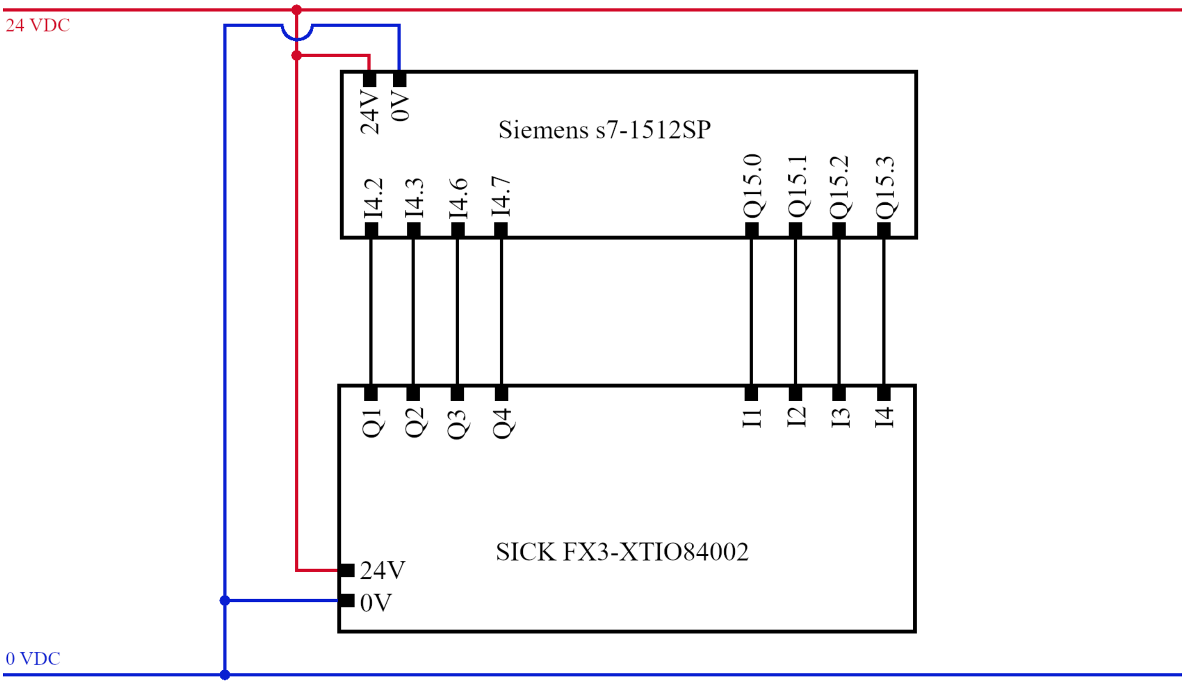


Figure 6.20: SICK Safety PLC to Siemens 1500 Safety I/O Connections

6.3.3.c STO Circuits for Servo and Stepper Drives

The Safe Torque Off (STO) circuits that control the Siemens v90 servo drives (traction) and the Festo CMMS-ST stepper drives (steering) are described in this section. These STO circuits are driven by the fail-safe digital outputs of the F-DQ 4x24VDC/2A PM HF card in slot 10 of the Siemens s7-1500 PLC.

A noted behaviour of the PLC safety outputs is that they will self-test approximately every hour. This self-test involves switching safety output on and off rapidly while measuring the residual current on the return path. When the voltage to the relay (connected to the safety output) is cut, the current should reduce to near zero; if this does not occur, relay failure is suspected. The STO state will be entered if relay failure is suspected, and a "disable run" command is sent via the bus network. The switching is fast enough that the attached device does not register the switching as an STO stop command [74].

The wiring between the Siemens s1500 PLC and the Siemens V90 servo drives (used for traction) is illustrated in figure 6.21 . A DTDP relay is used here, with each throw controlling one of the two STO inputs of the drive for added redundancy should one of the contacts weld. The SIL level of the STO implementation, according to the drive manufacturer (Siemens), is SIL 2 per EN61800-5-2 (PL d Equivalent) [75].

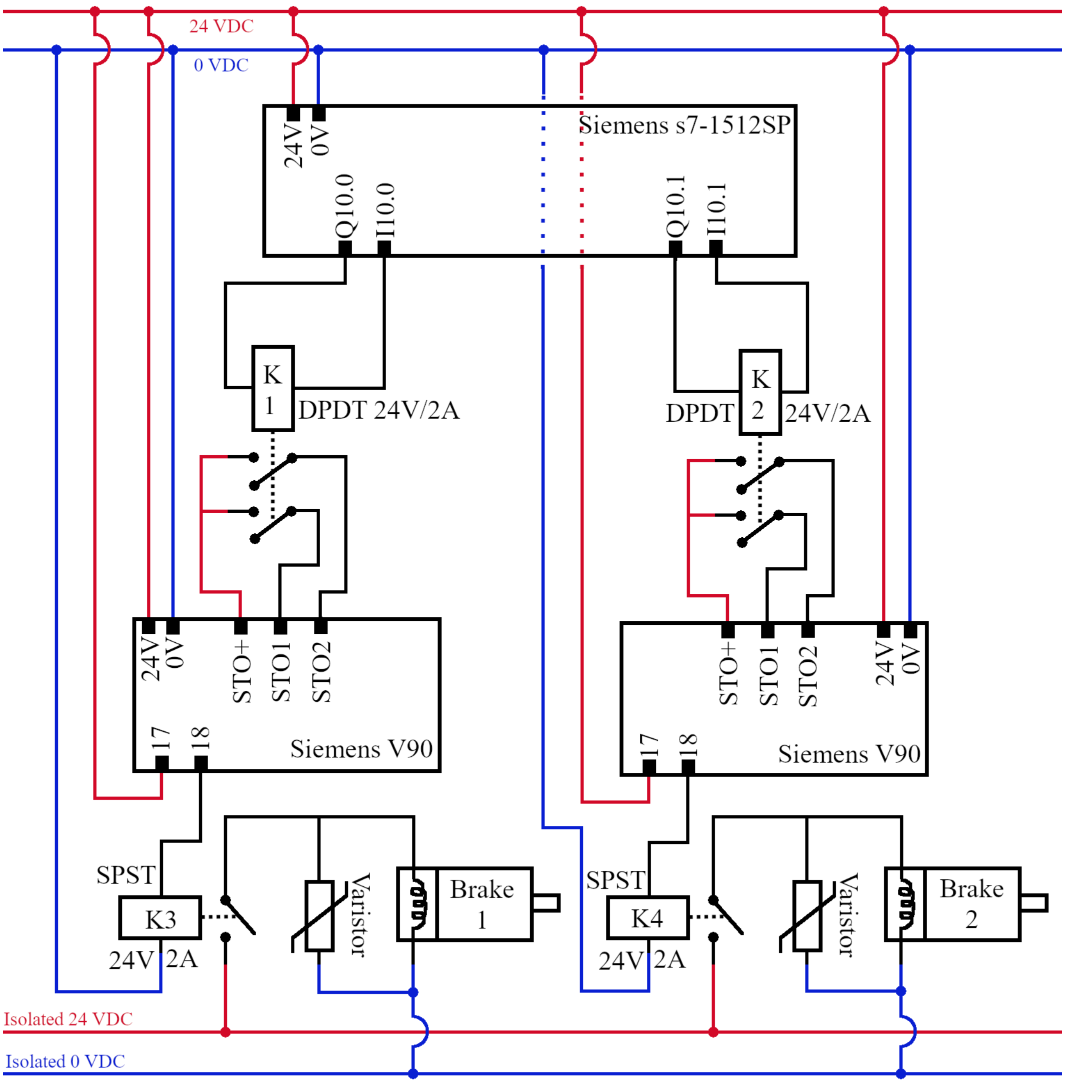


Figure 6.21: Siemens V90 Servo Motor STO Circuit

In figure 6.21, the wiring of the motor's brake is illustrated. This brake is not powered by the drive but rather by an external 24 VDC power source and is actuated via one of the drive's outputs (connection 18). Connection 18 draws power from the supply connected to connection 17. In the application notes for this drive, it is insisted that the supply for the brake be separate from the supply for other electronics, most likely due to the brake coil acting as a giant inductor. It is also recommended that an appropriate varistor be connected across the brake coil as illustrated in figure 6.21. The chosen (and recommended by Siemens) varistor implemented on the AGV was the EPCOS S20K20.

The wiring between the Siemens s7-1500 PLC and the Festo CMMS-ST stepper drives (used for steering) is illustrated in figure 6.22. A DTDP relay was used here; the first throw of the relay controlled the STO input of the drive (labelled REL on the drive), while the second throw controlled the "output stage enable" of the drive acting as a second level redundancy to shut down the drive. The Festo CMMS-ST drives have a 3rd enable signal, directly controlled from a non-safe output of the PLC (Q1.2 & Q1.3); this enables the drive's control board. The third enable signal was not done over a safety channel as there were none left in the F-DQ cards, and this enables itself not rated for safety applications. Safety switching the two inputs mentioned should rate the stepper drive at PL d, according to the manufacturer (Festo), per EN ISO 13849-1 (SIL 2 Equivalent) [76].

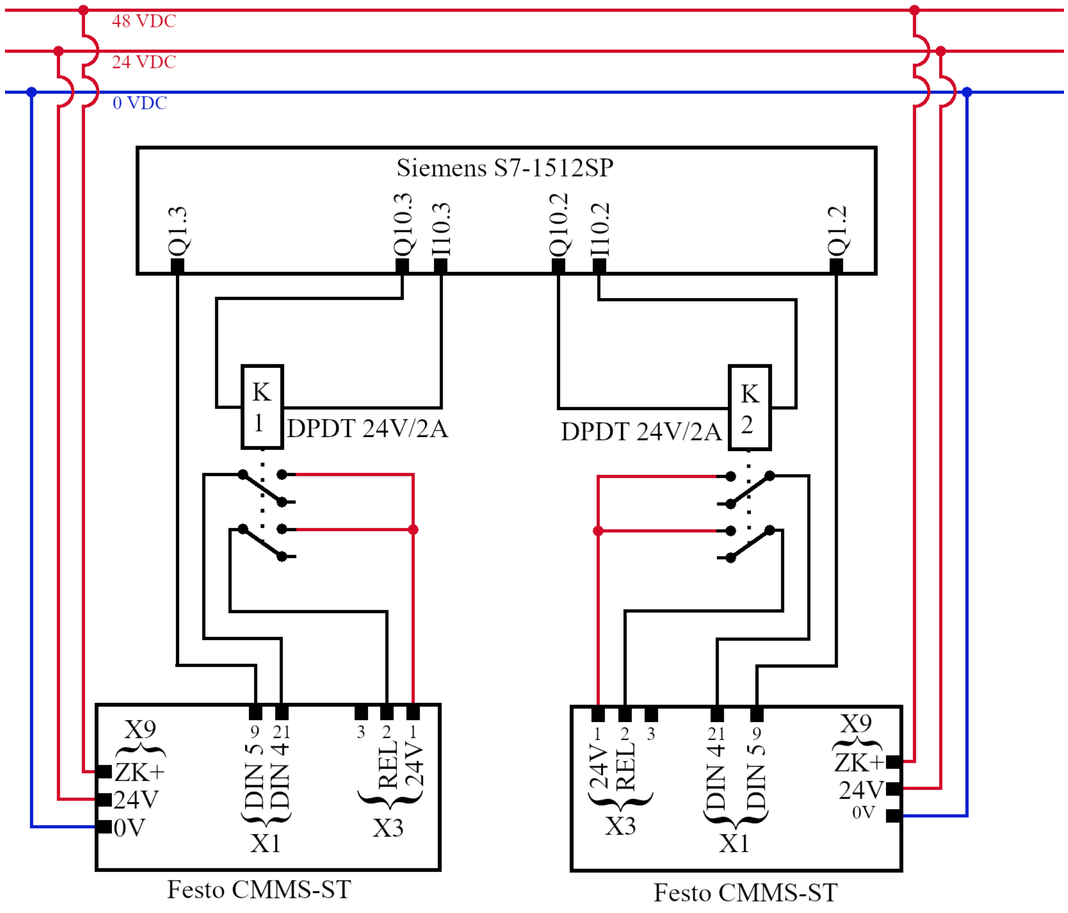


Figure 6.22: Festo CMMS-ST Stepper Motor STO Circuit

Unlike the Siemens V90 Servo motors, the Festo CMMS-ST stepper motors do not have external braking hardware; the brakes are implemented directly by the drive.

6.3.4 Battery Unit Eject Motor

The battery eject motor is an ELP.WM10412 Hilux windscreen wiper motor. This motor is a simple permanent magnet DC motor with a worm gearbox. Since this motor is responsible for both ejecting and loading the main battery unit, the motor must be able to spin both clockwise and counter-clockwise.

For a DC motor to change direction, the polarity of the voltage across it needs to be inverted. Swapping the polarity presents two issues in the current application of the AGV. Firstly, since this is an automotive part, the motor's negative terminal is tied to the chassis ground (in normal usage, the motor only spins one way). Having one of the terminals tied to the chassis ground is an issue since when the polarity is swapped, the +12 VDC would be connected to the chassis ground shorting out the auxiliary battery bank and other PSUs which are also tied to the chassis ground. Secondly, a circuit needs to be designed to switch the polarity or the motor using digital signals from the Siemens s7-1500 PLC; this system must also contain the appropriate hardware-level interlocking to prevent the system from receiving a counter-clockwise and clockwise command at the same time. Since this will cause a dead short across the auxiliary battery bank, possibly causing a fire.

The negative terminal tied to the ground is fixed by opening the motor and severing the connection of the second brush to the motor body. The severed connection is replaced by a new cable of appropriate gauge that runs alongside the first brush cable. Thus, neither brush/terminal of the motor is tied to the chassis ground.

The polarity switching circuit was implemented with relay logic, with appropriate hardware interlocking. A diagram of the relay system is illustrated in figure 6.23.

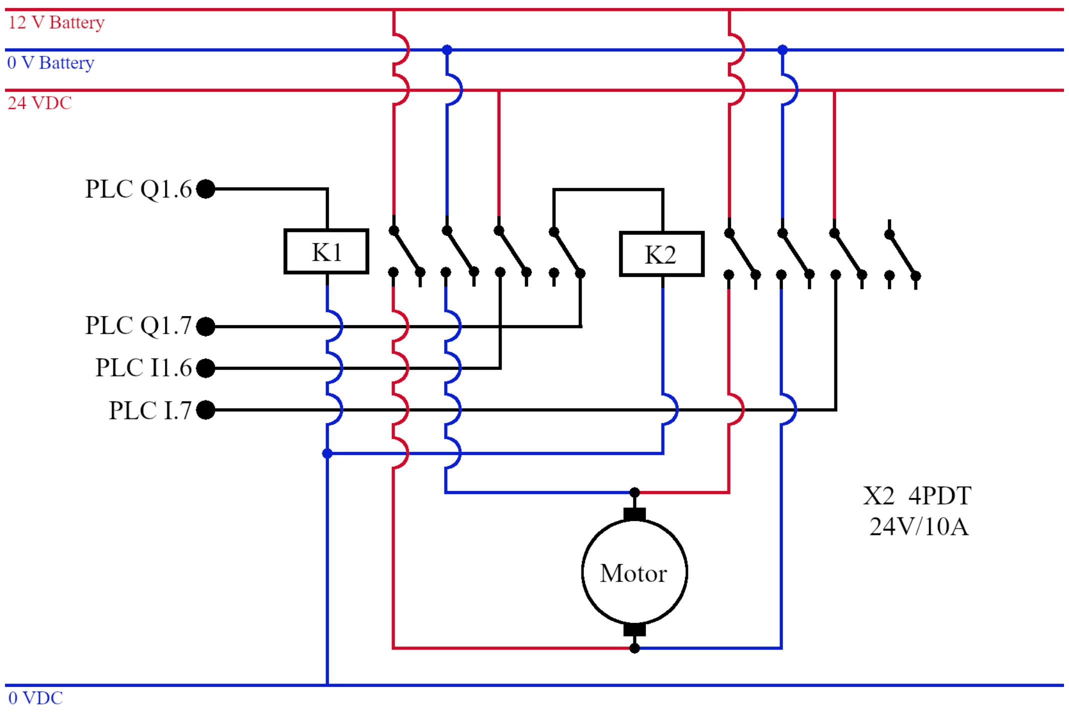


Figure 6.23: Battery Eject System System using Relays

In figure 6.23, it can seen that feedback was implemented via inputs I1.6 and I1.7.

6.3.5 Networking

A PFsense router controls the network on the AGV. PFsense is an enterprise-grade routing software package designed by Netgear to run on x86 hardware. This router is a client to a high-level plant network and an AP (access point) for programming devices. The inclusion of a full-blown router in the AGV was done for the following reasons:

- IoT Security

  The router and its firewall sit physically between the AGV's automation network and any external connections. These external networks include the interface a PG/PC would use to program the AGV and the high-level network the AGV would connect to as a client device. Using a PFsense box adds a layer of IoT security to the AGV since the firewall makes it much harder for malicious actors to access the AGV's systems even if the network that the AGV is connected to was compromised.

- IP Consolidation

When the AGV is viewed from the plant network perspective, the entire AGV is registered as one IP address. Reducing the AGV to a single IP saves IP addresses on the plant network and adds a layer of obfuscation. With such a system, the plant network engineers need not worry about assigning static IP addresses to each component in each AGV. The data sent from the plant network can still be directed to the appropriate device on the AGV's internal network via port forwarding in the router.

- DHCP Client

  Since the AGV has a router, the plant level network can use a DHCP server to assign IP addresses to the AGVs in the facility automatically. Using a plant DHCP allows AGVs that need servicing to be removed from circulation easily, with a backup AGV slotting into its place. Data is sent to the appropriate device on the AGV's network via port addressing rather than IP addressing. The use of a DHCP client system for the AGVs is also helpful regarding AGVs moving between different isolated networks/subnets as the AGVs physically move through a large factory and enter the range of different automation networks.

- DHCP Server

  The programming AP of the AGV can be accessed via either a physical ethernet cable (higher speed and reliability) or via a separate WIFI network generated by the AGV. Either way, the AGV will use a private DHCP server on the Pfsense router to assign the programming device an IP address. Assigning the programming device an IP address prevents issues where a programmer might accidentally set a static IP address on their programming device, which conflicts with a device on the AGV. It also makes connecting to the AGV easier as documentation will not need to be consulted regarding the subnet that the AGV.

- Trusted MAC Addresses

  The router can be configured to reject any MAC address (a hexadecimal hardcoded address unique to each network-capable device in the world) that does not appear on a whitelist contained in the router. Thus, malicious actors would have to guess the MAC address of an approved programming device (There are approximately 281 trillion possible MAC addresses [77]). If the attacker were to bypass this hurdle, they would still have to bypass the cryptography of the firewall to compromise the AGV. MAC filtering can also be implemented on the plant network, where the AGV rejects any WIFI access points that are not approved in its whitelist.

- DDoS Resilience and Network Chatter Reduction

  DDoS (distributed denial-of-service) is a malicious attack strategy whereby a device is bombarded by an inordinate amount of irrelevant data in the hope to overwhelm a device and drown out legitimate data packets. A firewall does not strictly prevent a DDoS attack but will prevent the attack from directly targeting the AGV's internal network. That is to say, the AGV will lose connection to the plant network as the firewall is overwhelmed and locks down, but the devices inside the AGV will still be able to communicate with each other freely. Thus the internal navigation data and Profisafe data will not be interrupted, and the AGV can make a controlled halt. Since the firewall in the router only allows relevant data through, any data not meant to cross networks will be blocked. This segregation reduces background chatter on both the AGV's internal and plant networks, especially when UDP protocols are employed. UDP is used heavily by the LiDAR scanner and ROS nodes on the AGV; since this protocol does not use handshaking, it can proliferate to multiple devices rather than just the intended recipient causing unnecessary chatter on the network.

A diagram of the AGVs network is shown in figure 6.24.

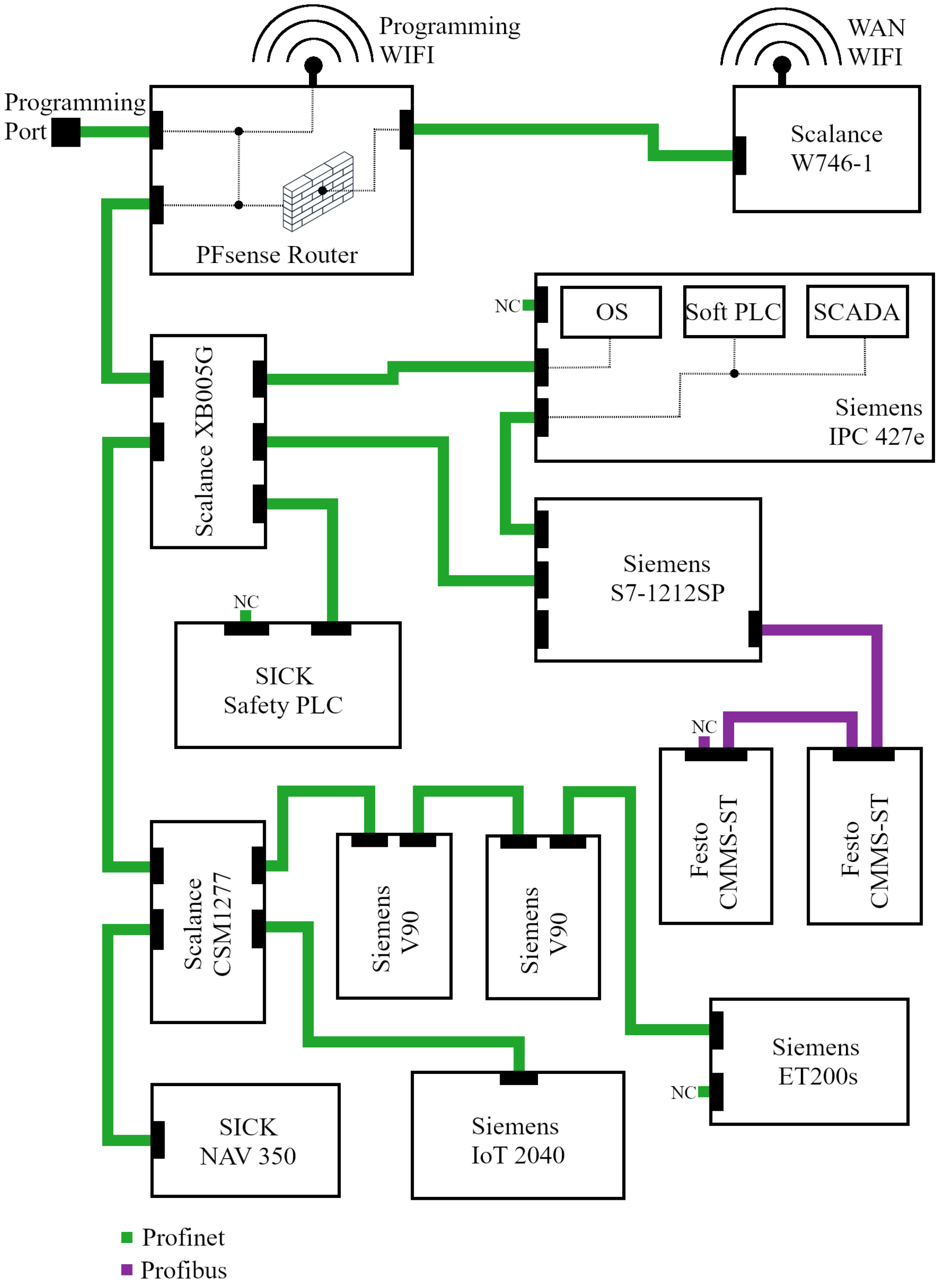


Figure 6.24: Network Topography

A list of the IP addresses used by the AGV's internal Ethernet/Profinet network is listed in table 6.18, while a list of device addresses for the Profibus network is listed in table 6.19.

Table 6.18: AGV IP Addresses List

Device	IP Address	NIC Purpose
Pfsense Router NIC 1	192.168.18.1	AGV Ethernet
Pfsense Router NIC 2	192.168.2.1	W746-1 Ethernet up-link
Pfsense Router NIC 3	192.168.18.1	Ethernet programming
Pfsense Router NIC 4	192.168.18.1	WIFI programming
Siemens IPC 427E NIC 1	192.168.18.51 NAT[10]	Windows 7 VM
	192.168.18.51 NAT	Future ROS linux VM
Siemens IPC 427E NIC 2	192.168.18.52 NAT	PLC Profinet Master
	192.168.18.52 NAT	WinCC SCADA
Siemens IPC 427E NIC 3	192.168.18.53	Unused Ethernet
Siemens V90 Drive A	192.168.18.57	Profinet Slave
Siemens V90 Drive B	192.168.18.56	Profinet Slave
Siemens 1512SP PLC	192.168.18.55	Profinet Master
SICK Safety PLC (FX-CPU32002)	192.168.18.61	Profinet Master
Siemens IoT 2040 AGV NIC 1	192.168.18.60	AGV Ethernet
Siemens IoT 2040 AGV NIC 2	192.168.60.4	IoT WIFI
Siemens IoT 2040 Battery NIC 1	disabled	Unused Ethernet
Siemens IoT 2040 Battery NIC 2	192.168.60.3	IoT 2040 WIFI
Siemens W746-1 NIC 1	192.168.2.1	PFsense up-link
Siemens W746-1 NIC 2	Plant DHCP	Plant WIFI
SICK NAV350-3232	192.168.18.59	Profinet Slave
Siemens ET200S	192.168.18.58	Profinet Slave

The DHCP server uses IP addresses between 192.168.18.100 and 192.168.18.254 as freely assignable IP addresses for devices that are added ad hoc. This category includes programming devices attached to the system. During testing, the pseudo plant network DHCP assigned the AGV an IP address (Siemens W746-1 NIC 2 IP address) of 192.168.2.5.

[10] NAT (network address translation) in this implementation allows virtual devices to use the same IP address when on the same physical hardware

Table 6.19: AGV Profibus Addresses List

Device	**Profibus Address**	**Interface Purpose**
Siemens 1512SP PLC	2	Profibus Master
Festo CMMS-ST Drive A	3	Profibus Slave
Festo CMMS-ST Drive B	4	Profibus Slave

6.3.6 IoT Integration

IoT is not the focus of this research; however, the allocation has been made for future implementation. This implimetation can be done in three possible ways:

1. IoT 2040

   As eluded to in the name, these devices are aimed at IoT applications. Since these are Siemens devices, they can relatively easily be connected to Mindsphere; a Siemens hosted cloud service. Likewise, a third-party service such as Ubidots can also be used. Data is sent as outbound connections only to a cloud-based host. This strategy has a relatively small attack surface due to IoT interaction being done via a cloud-based dash panel rather than the AGV itself.

2. Windows/Linux API

   The Windows or Linux VM on the IPC can host a service that sends data to a cloud service via an API. Either Mindsphere or Ubidots could be used. Data is sent outbound connections only to a cloud-based host. This strategy has a relatively small attack surface due to IoT interaction being done via a cloud-based dash panel rather than the AGV itself.

3. S7-1500 Webserver

   Siemens s7-1500 range PLCs can host a web server, with either a custom web page or a prebuilt Siemens one. When data is accessed over the internet, the actor in question will be accessing services on the physical AGV. This strategy means that, to some extent, inbound connections and port-forwarding on the Pfsense firewall will have to be enabled. Such a system makes this type of IoT interaction very risky as an improperly configured firewall that allows inbound connections can provide an avenue of attack for malicious actors. This approach is not recommended due to its large attack surface.

Whether the device is directly accessed from the internet or if the information is accessed via a cloud-based dash panel, appropriate rules and port forwarding must be implemented in the AGV's Pfsense router.

6.3.7 Drive System

As mentioned in the previous sections and chapters, the AGV has two traction motors and two steering motors. The traction motors are two Siemens V90 servo motors, while the steering motors are a pair of Festo CMMS-ST stepper drivers. The electrical configuration of these drives is explained in the sections that follow.

6.3.7.a Traction Motors Electrical Configuration

The wiring of the traction motors is shown in figure 6.25. Figure 6.25 shows both drives in the system along with essential connections. Connections not used in the AGV's installation were omitted to simplify the diagram. For example, only pin 17 and 18 are illustrated in the I/O cable connections (X8) as these were the only connections implemented.

Pin 17 and 18 of the I/O cable were used to actuate the third party braking solution mentioned in figure P.1, the installation of this system is illustrated in figure 6.21 as part of the STO circuit hence it was not repeated in figure 6.25.

For clarity, striped cables in figure 6.25 represent bus or standardised multi-core cables.

F1 and F2 in diagram 6.25 are both 6A class C circuit breakers used to cut the 220 VAC power to the drives should an electrical fault occur.

The wiring diagram shown in figure 6.25 was developed using the Siemens application notes titled "SINAMICS V90, SIMOTICS S-1FL6 PROFINET (PN) interface: Getting Started" [78]. The wiring diagram from the application notes can be found in appendix P.

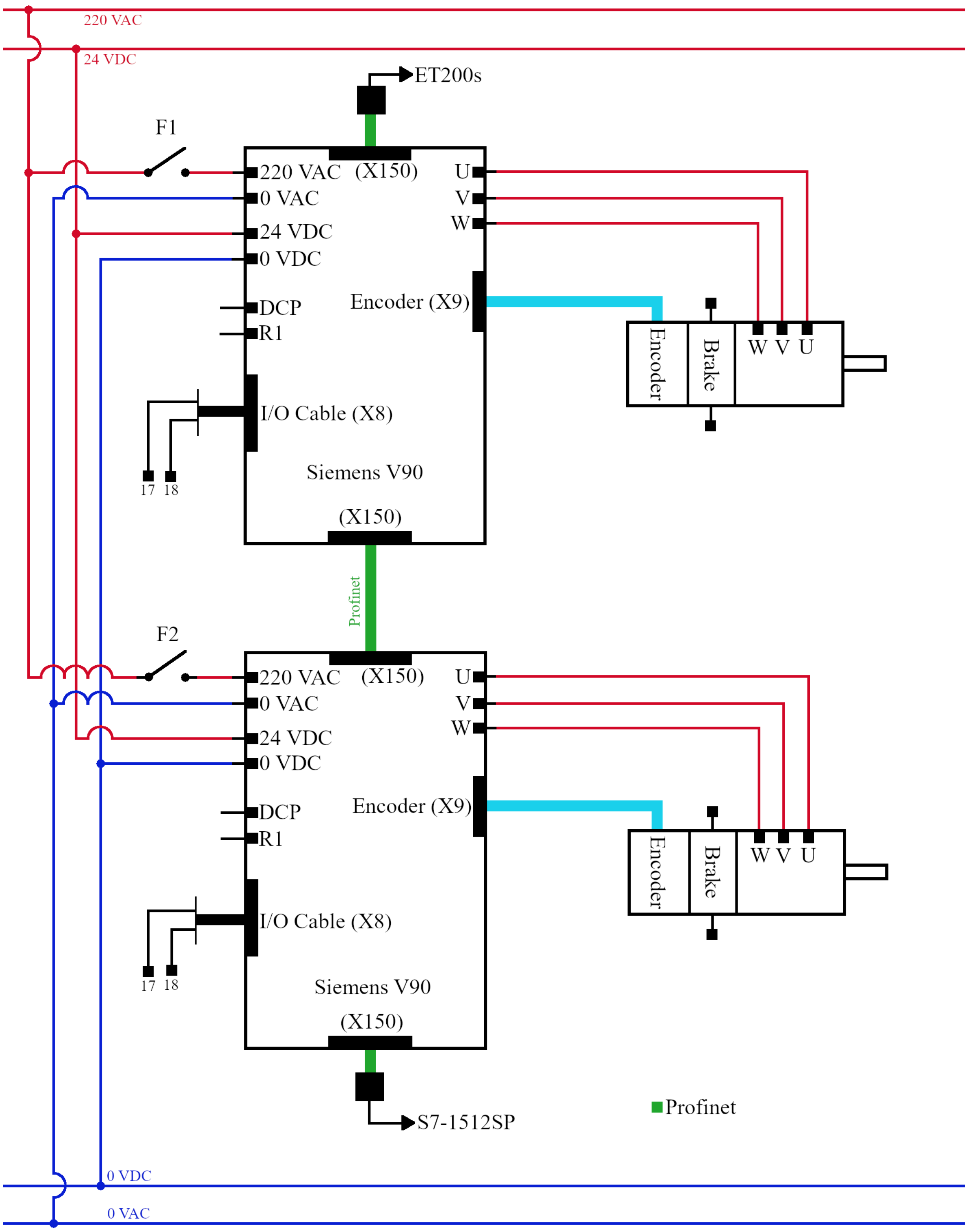


Figure 6.25: Electrical Installation of the Siemens V90 Servo Drives

6.3.7.b Steering Motors Electrical Configuration

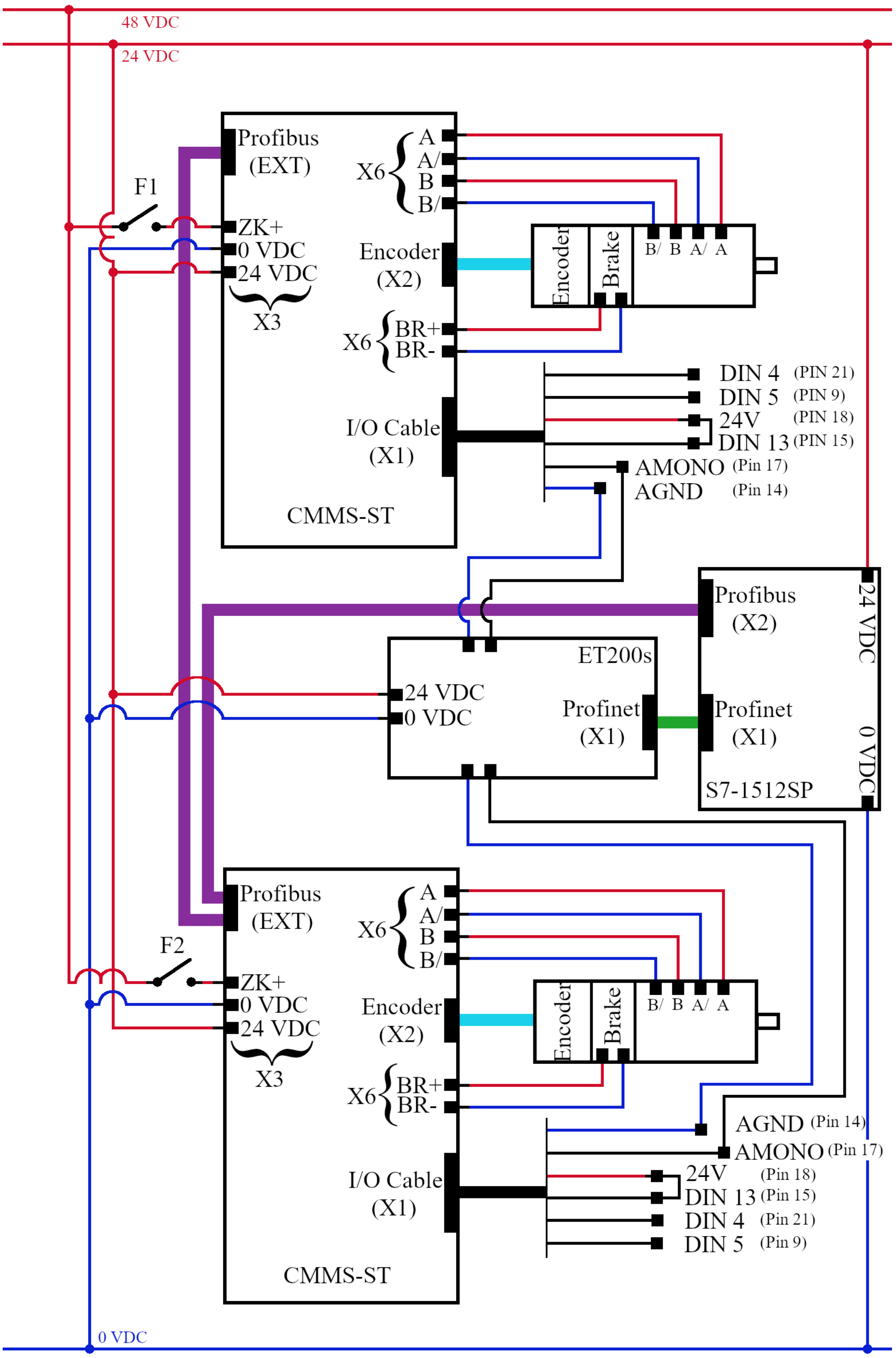


Figure 6.26: Electrical Installation of the Festo CMMST Stepper Drives

The wiring of the Festo CMMST stepper drive used for the AGV's steering is illustrated in figure 6.26.

The temperature sensor CAN bus, and master/slave connectors are not shown in figure 6.26 as they were not implemented in the AGV. Bus networks are represented as a stripped line. Note that only relevant connections implemented in the AGV are broken out of the bus connections in figure 6.26.

The Festo CMMST drives do not natively support Profinet or Profibus; however, Profibus connectivity can be added using a daughterboard fitted to the Drive's expansion slot as illustrated in figure 6.27 .The boards are called CAMC boards by Festo and support a variety of protocols; although a Profinet CAMC board is available from Festo, the CMMST range of drives does not support it hence, why the older Profibus network was employed.

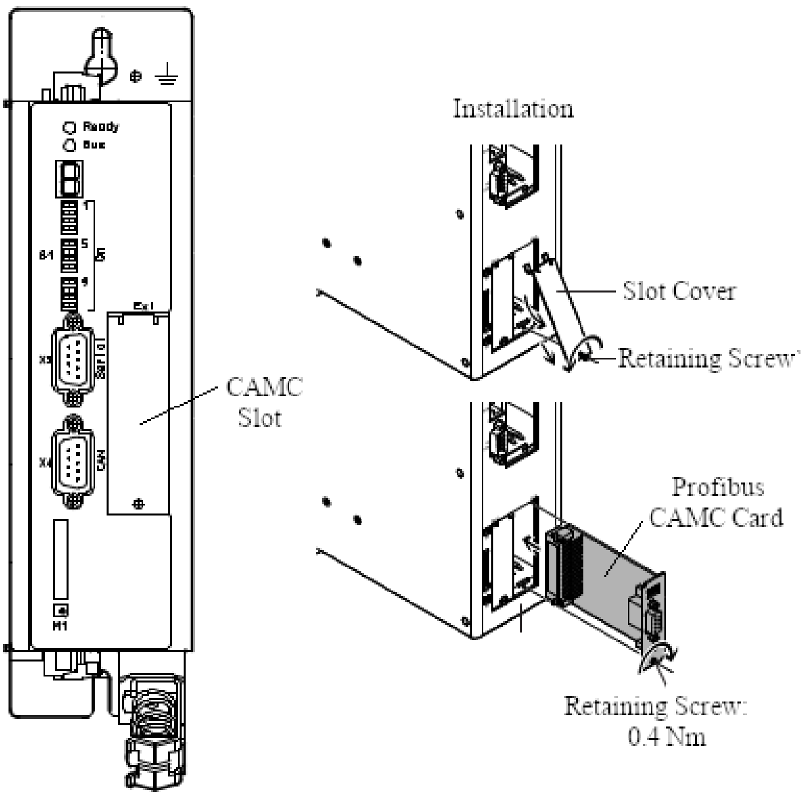


Figure 6.27: Electrical Installation of the Festo CMMST Stepper Drives[11]

The location and purpose of all the connections shown in figure 6.26 are illustrated on the physical drive in figure 6.28.

[11] Image adapted from Festo [76].

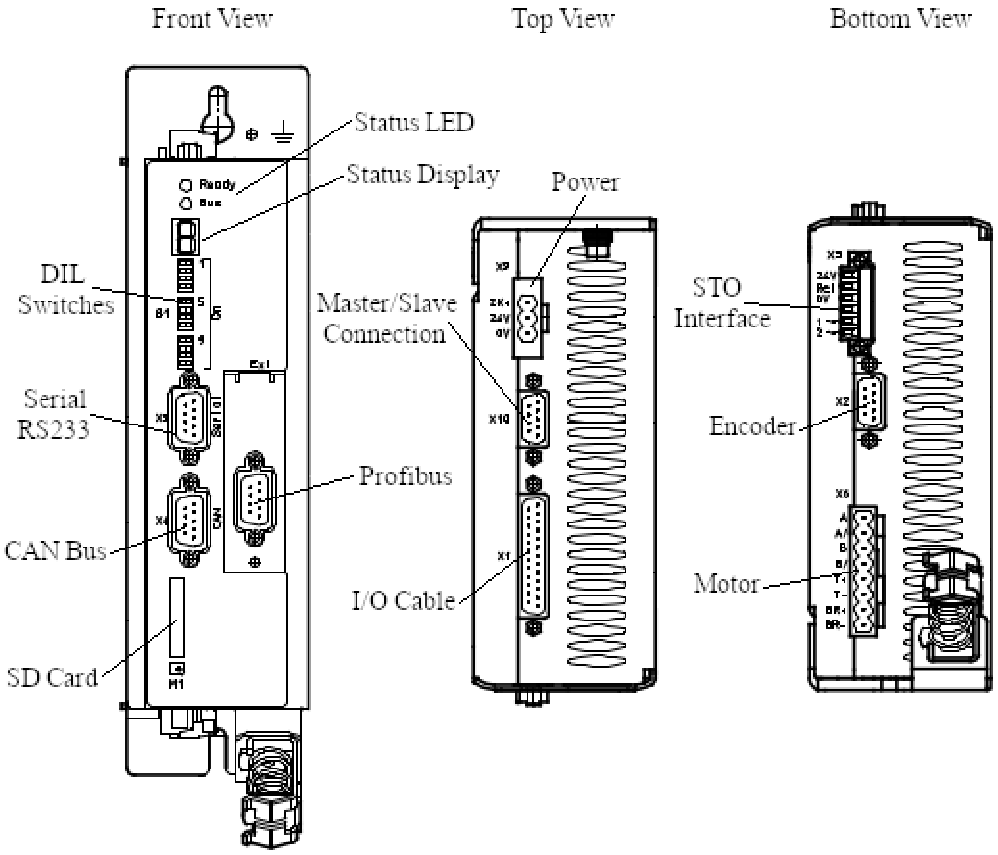


Figure 6.28: Electrical Connections of the Festo CMMST Drives[12]

The RS232 serial connection is used to configure the drives since configuration cannot be done through Profibus. The IPC is connected directly to this the drive via a serial cable for this purpose. Note, due to a software bug from Festo, the configuration of the CMMST drives can only be done via COM1 on any PC. The SD card on the drive stores configured parameters between power cycles (retentive memory). Thus, it is essential to manually save the configuration in RAM to the SD card (this is done via Festo's FCT software when the drive is connected to COM1).

The configuration and pinout of the connectors illustrated in figure 6.28 are listed in appendix Q.

6.4 Chapter Conclusion

This chapter outlined the electrical configuration of the AGV, along with specifying why certain electrical design decisions were made. This chapter was broken into two

[12] Image adapted from Festo [76].

primary sections. Firstly, the power delivery section which is responsible for powering the control systems and supplying power for the tractive effort. Secondly, the control systems section which specified all of the electronics used for the AGV's control system.

7 Safety Evaluation

7.1 Introduction

Any machine that operates near people should be designed in such a way to eliminate or reduce, as much as feasibly possible, the danger it presents to human beings. Thus, this chapter will outline steps taken by the author to ensure that the AGV is "safe" to be used amongst people and what laws and directives were followed to ensure its compliance.

7.2 Standards and Directives

Safety evaluation of the AGV designed in this report was done according to the EU Machinery directive 2006/42/EC. Although the AGV in this report was developed in South Africa, the EU directive was used for the following reasons. Firstly, the South African standards, SANS (South African National Standards), which are administered by SABS (South African Bureau of Standards), are often of poor quality and outdated, with the majority of SANS standards simply being "borrowed" by SABS from the ISO (International Organisation for Standardization) or IEC (International Electrotechnical Commission) standards. Secondly, since the EU Machinery Directive is synergised with the ISO and IEC standards, any machines built according to this directive will adhere to the appropriate SANS standard when SABS copies the ISO and IEC standards. Lastly, any machine built to EU standards will not have issues being imported into any country in the world as the EU standards are some of the most strict in the world, and as such, the AGV will likely pass the standards in other countries, such as [43]:

- EN (Europe)
- DIN (Germany)
- JIS (Japan)

- C-Tick, A-Tick (Australia)
- ANSI (United States of America)
- UL(C), CSA (Canada)
- IEC (International)
- ISO (International)

There are three steps to implementing the EU Machinery directive 2006/42/EC on the AGV. These steps are shown in the process chain, in figure 7.1.

Risk Assessment
Risk Management
Verification

Figure 7.1: Implementation of the Machinery Directive

7.3 Risk Assessment

The risk assessment step for the AGV has four steps as defined by the EU machinery directive [43]:

1. Define machinery boundaries
2. Identify hazards
3. Estimate risk
4. Assess risk

7.3.1 Step 1: Define Machinery Boundaries

The machinery boundaries are listed in accordance with ISO 12100 in the paragraphs that follow:

Intended Use

- AGV intended for mobility
- AGV intended for goods transportation
- AGV intended for both manual control and autonomous movement

- AGV intended for use in the vicinity of humans
- AGV size 1000 mm x 1700mm

Application Boundaries

- Maximum AC power present: 220 VAC @ 50Hz
- Maximum DC power present: 48 VDC
- Battery operation only, not to be operated with charging cable attached
- Machine intended for factory use (IP54)
- Temperature range: -5 °C to 50 °C
- Maximum Speed: 1.3 $^{m}/_{s}$
- Maximum Net Weight: 1000 kg
- Maximum incline: 5°

User Groups

- Operation only by specialist personnel, no laypersons
- Trainees are only to operate the AGV under supervision by specialists
- Operates in the vicinity of factory workers whom must be given safety training on how to interact with the machine

Time Boundaries

- Not to be operated longer than 3 months between services

Physical Boundaries

- Machine intended for operation only in forklift demarcated lanes
- Not to be operated in areas demarcated for machine operators

7.3.2 Step 2: Identity Hazards

According to ISO 12100, the following risks must be evaluated to ensure compliance: Cutting, Dropping, Motion, Gravity, Approach and Rotation. This process is summarised in figure 7.2.

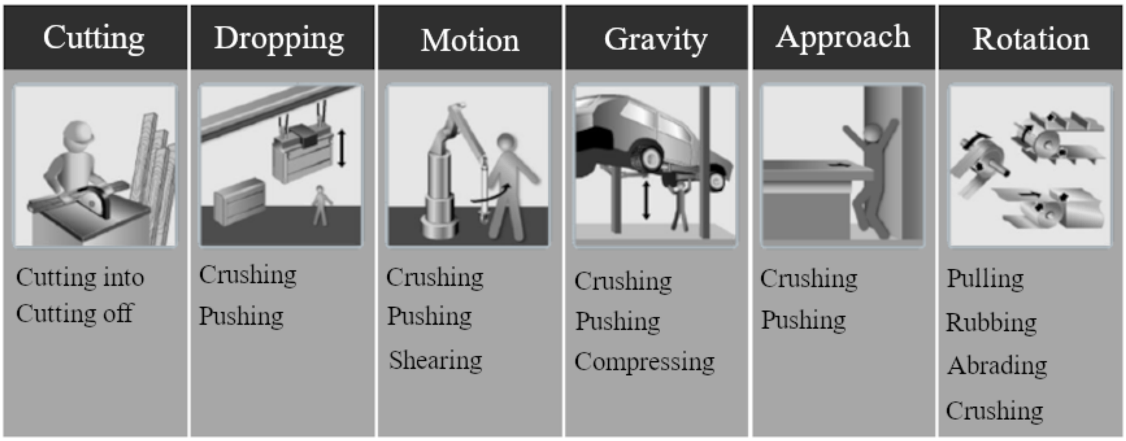


Figure 7.2: Possible Hazards According to ISO 12100 [1]

The following risk was identified when using the AGV contained in this thesis:

- Crushing risk when payload is load or offloaded onto AGV
- Crushing risk if AGV pins personal against a wall or object
- Pushing risk if AGV collides with person
- Shearing risk if appendages get caught in AGV when moving
- Pulling risk if appendages or clothing caught in AGV when moving
- Abrading risk if individual dragged by AGV
- Cutting risk if appendages caught in drive train
- Shearing risk if appendages caught in drive train
- Crushing risk if appendages caught in drive train
- Electrocution risk if individual touches exposed terminals
- Abrading risk if appendages caught in machinery

7.3.3 Step 3: Estimate Risk

Estimating the risk of a hazard is broken into two parts, namely, the severity of the risk and the probability of it occurring. This concept is illustrated in figure 7.3.

[1] Image adapted from ISO 12100 Standard

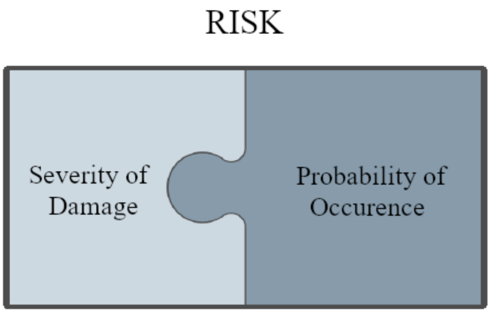


Figure 7.3: Severity of Risk Components

Severity of Risk

The severity of risk can be summarised as shown in the tiered list:

1. Reversible, first aid necessary
2. Reversible, treat by doctor necessary
3. Broken limbs, loss of fingers
4. Irreversible, death, loss of eye or arm

The Probability of Occurrence

Frequency and Duration of Exposure

- Is there a need to access a hazardous area
- Type of access required and duration of exposure
- Number of people accessing the hazardous area and the frequency of access

Probability of a Hazardous Event Occurring

- Low
- Medium
- High

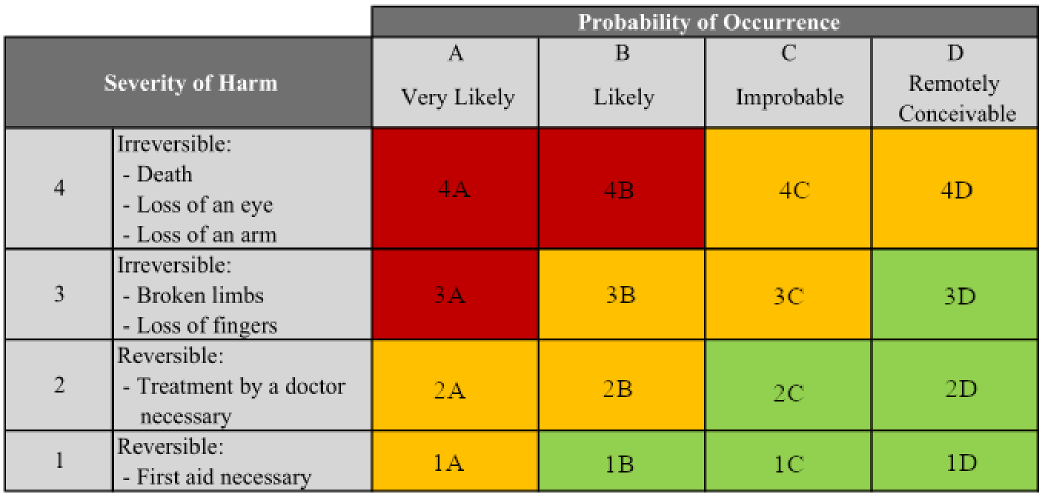
Probability of Avoiding or Limiting Damage

- Type of movement: sudden, fast or slow
- Qualifications of people
- Risk awareness

- Reflexes, practical experience
- Mobility, possibility of escape

7.3.4 Step 4: Assessment of Risk

Using the severity of risk along in conjunction with the probability of occurrence, from section 7.3.3, a matrix can be developed per the EU machinery directive 2006/42/EC and compliant with ISO 12100 as shown in figure 7.4.



Severity of Harm		Probability of Occurrence			
		A Very Likely	B Likely	C Improbable	D Remotely Conceivable
4	Irreversible: - Death - Loss of an eye - Loss of an arm	4A	4B	4C	4D
3	Irreversible: - Broken limbs - Loss of fingers	3A	3B	3C	3D
2	Reversible: - Treatment by a doctor necessary	2A	2B	2C	2D
1	Reversible: - First aid necessary	1A	1B	1C	1D

Figure 7.4: Risk Assessment Matrix[2]

The risks identified in section 7.3.2 were evaluated using the matrix in figure 7.4 to create table 7.1. Table 7.1 contains the results of the risk assessments done by the author.

[2] Table adapted from Siemens TIA Safety [43]

Table 7.1: Results of Risk Assessment

Hazard	**Risk Rating**
Crushing risk when payload is load or offloaded onto AGV	3A
Crushing risk if AGV pins personal against a wall or object	4B
Pushing risk if AGV collides with person	4B
Shearing risk if appendages get caught in AGV when moving	4B
Abrading risk if individual dragged by AGV	4B
Cutting risk if appendages caught in drive train	3B
Shearing risk if appendages caught in drive train	3B
Crushing risk if appendages caught in drive train	3B
Electrocution risk if individual touches exposed terminals	4B
Abrading risk if appendages caught in machinery	3B

Although the severity of harm is relatively high for many hazards, the probability of occurrence is moderate. The author made this assessment due to the low speed of the AGV. The vehicle's top speed is 1.3 $^{m}/_{s}$ (which is the walking speed of an average human), which gives people ample time to realise the AGV is present and then move out of the way.

7.4 Risk Mitigation

Risk mitigation on the AGV involves reducing the risk assessment rating in table 7.1 by applying safety devices and strategies to the AGV per ISO 12100 [43]. The steps to be taken are illustrated in figure 7.5.

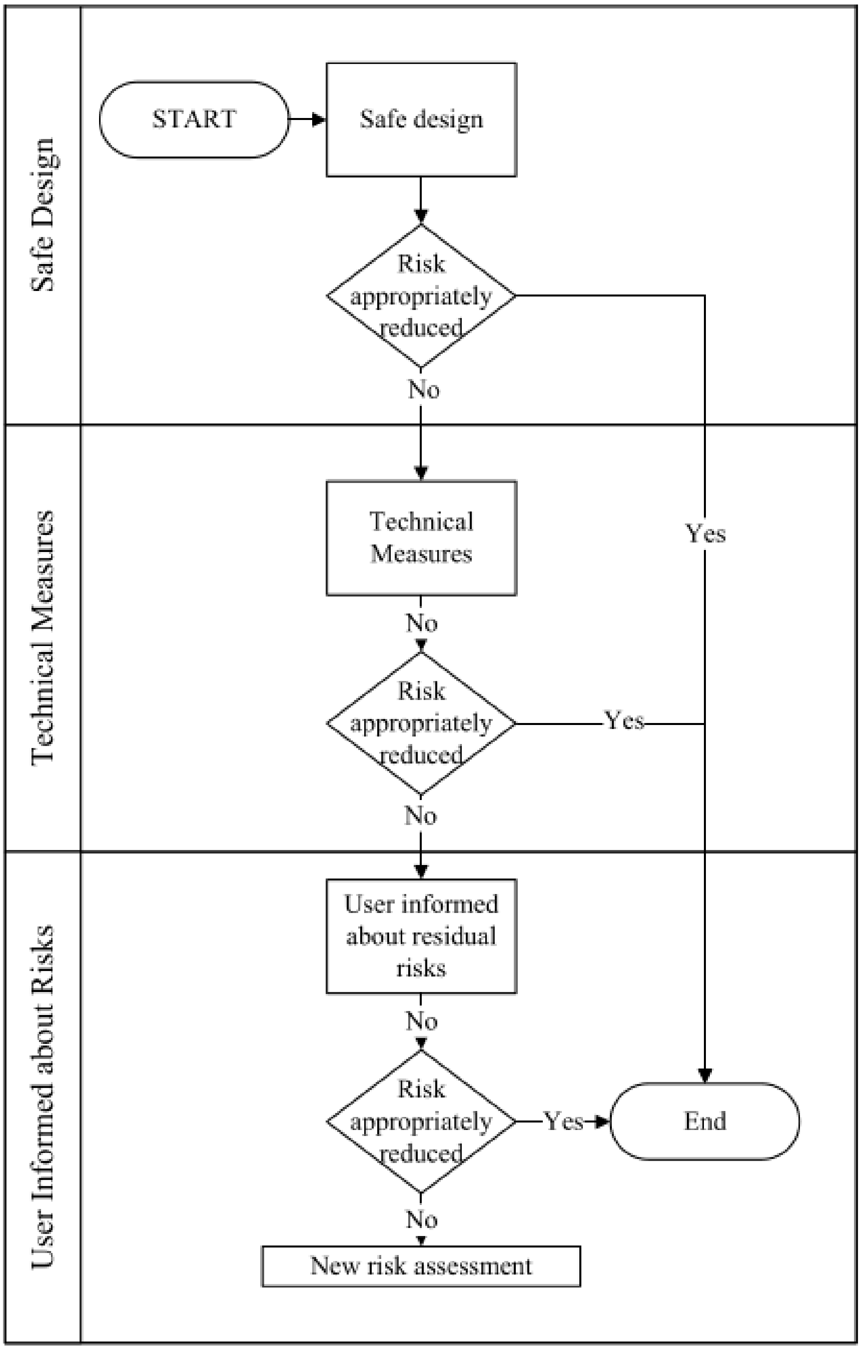

Safe Design
Technical Measures
User Informed about Risks
START
Safe design
Risk appropriately reduced
No
Yes
Technical Measures
No
Risk appropriately reduced
Yes
No
User informed about residual risks
No
Risk appropriately reduced
Yes
End
No
New risk assessment

Figure 7.5: Risk Mitigation in Accordance with ISO 12100

7.4.1 Safe Design

Safe design reduces the likelihood of injury to personnel by ensuring the machine is built in such a way as to minimise access to dangerous sections of the machine. For the AGV, the risk rating is reduced by applying safe design practices. These practices are listed in table 7.2 ,along with an adjusted safety rating.

Table 7.2: Safe Design: Results of Risk Assessment

Hazard	**Risk Rating**
Crushing risk when payload is load or offloaded onto AGV • none	3A
Crushing risk if AGV pins personal against a wall or object • none	4B
Pushing risk if AGV collides with person • none	4B
Shearing risk if appendages get caught in AGV when moving • Ensure there are no snag points on AGV cowling	***4C***
Abrading risk if individual dragged by AGV • Ensure there are no snag points on AGV cowling	***4C***
Cutting risk if appendages caught in drive train • Enclose all moving parts inside AGV cowling	***3C***
Shearing risk if appendages caught in drive train • Enclose all moving parts inside AGV cowling	***3C***
Crushing risk if appendages caught in drive train • Enclose all moving parts inside AGV cowling	***3C***
Electrocution risk if individual touches exposed terminals • Enclose all moving parts inside AGV cowling • Enclose all electronics in electronic boxes	***4D***
Abrading risk if appendages caught in machinery • Enclose all moving parts inside AGV cowling	***3C***

In table 7.2, the risk ratings that were lowered are bolded. The reason many of the risks are reduced to C (improbable) rather than D (remotely conceivable), even when there is a physical barrier between the individual and the dangerous components, is because when the access panels are removed, the machine does not automatically shut down.

7.4.2 Technical Measures

Technical preventative measures interact with the machine's behaviour rather than being a simple barrier or guard. On this AGV, three technical measures were introduced to reduce the risk rating of the AGV:

1. Installing an E-stop on either end of the AGV to shut down the system
2. Installing two SICK S300 Mini Remote scanners to shut down the AGV if a person or object enters the "danger" zone
3. Implementing a deadman switch on the pendant for use when the AGV is in manual mode (i.e. when a person is in the vicinity of the AGV)

The new risk ratings of the AGV, when both the "safe design" mitigations and the "technical measures" of this section are implemented, is given in table 7.3.

Table 7.3: Technical Measures: Results of Risk Assessment

Hazard	**Risk Rating**
Crushing risk when payload is load or offloaded onto AGV • none	3A
Crushing risk if AGV pins personal against a wall or object • safety scanner shuts down motion when person enters danger zone • E-stop shuts down motion when triggered	***2D***
Pushing risk if AGV collides with person • safety scanner shuts down motion when person enters danger zone • E-stop shuts down motion when triggered	***2D***
Shearing risk if appendages get caught in AGV when moving • safety scanner shuts down motion when person enters danger zone • E-stop shuts down motion when triggered	***2D***
Abrading risk if individual dragged by AGV • safety scanner shuts down motion when person enters danger zone • E-stop shuts down motion when triggered	***2D***

cont...

Hazard	Risk Rating
Cutting risk if appendages caught in drive train • safety scanner shuts down motion when person enters danger zone • E-stop shuts down motion when triggered	*2D*
Shearing risk if appendages caught in drive train • safety scanner shuts down motion when person enters danger zone • E-stop shuts down motion when triggered	*2D*
Crushing risk if appendages caught in drive train • safety scanner shuts down motion when person enters danger zone • E-stop shuts down motion when triggered	*2D*
Electrocution risk if individual touches exposed terminals • All circuits are fused appropriately with circuit breakers • 220 VAC bus contains earth leakage protection	*2D*
Abrading risk if appendages caught in machinery • safety scanner shuts down motion when person enters danger zone • E-stop shuts down motion when triggered	*2D*

The risk rating of being injured by internal components of the AGV was reduced in table 7.3. Since none of the moving parts will be active if someone enters the safety scanners range, thus, even if a maintenance panel were to be removed and someone were to stick their hand into the opening, the machine itself would be dead due to the safety scanners detecting that person's presence.

7.4.3 User Informed about Residual Risks

The only risk that has yet to be reduced to a green zone (as illustrated in figure 7.1) is the risk of crushing fingers or toes when the payload is loaded or offloaded. Mitigating this is relatively difficult since if a person is allowed to load or offload items manually, there will always be a risk that they can hurt themselves. The only way to mitigate this risk is by enforcing operational workplace safety. That includes ensuring workers wear safety boots and only manoeuvre heavy loads with the appropriate tools (forklifts, block and tackles, or lift carts).

Ensuring that workers take these precautions is directly tied to informing the workers (or, more likely, the safety officer) about this residual risk. If this is done, then the final risk ratings will be as given in table 7.4.

Table 7.4: User Informed: Results of Risk Assessment

Hazard	**Risk Rating**
Crushing risk when payload is load or offloaded onto AGV • Occupational health and safety observed	***2C***
Crushing risk if AGV pins personal against a wall or object • Personnel informed to stay out of forklift lanes	2D
Pushing risk if AGV collides with person • Personnel informed to stay out of forklift lanes	2D
Shearing risk if appendages get caught in AGV when moving • Personnel informed to stay out of forklift lanes	2D
Abrading risk if individual dragged by AGV • Personnel informed to stay out of forklift lanes	2D
Cutting risk if appendages caught in drive train • only qualified technicians allowed to open AGV	2D
Shearing risk if appendages caught in drive train • only qualified technicians allowed to open AGV	2D
Crushing risk if appendages caught in drive train • only qualified technicians allowed to open AGV	2D
Electrocution risk if individual touches exposed terminals • only qualified technicians allowed to open AGV	2D
Abrading risk if appendages caught in machinery • only qualified technicians allowed to open AGV	2D

7.5 Architecture of Safety Functions

Once the AGV has been made appropriately "safe" by implementing the aforementioned risk mitigation techniques, the techniques themselves have to be analysed to ensure that they can be "trusted". This analysis is done according to ISO 13849-1, or IEC 62061 [43].

ISO 13849-1 designates "Performance Levels" (PL a to PL e) to safety systems, while

IEC 62061 assigns "Safety Integrity Levels" (SIL 1 to SIL 3) to safety systems. Although these are two different standards, they can be interchanged as shown in table 7.5. Note SIL 3/ PL e is the highest reliability rating [43].

Table 7.5: PL to SIL Conversion

Performance Levels (ISO 13849-1)	**Safety Integrity Level (IEC 62061)**	**Reliability** $[{}^{failures}/_{hour}]$
PL a		10^{-5} to 10^{-4}
PL b	SIL 1	3×10^{-6} to 10^{-5}
PL c	SIL 1	10^{-6} to 3×10^{-6}
PL d	SIL 2	10^{-7} to 10^{-6}
PL e	SIL 3	10^{-8} to 10^{-7}

The methodology for determining the PL and SIL rates are is given in figure 7.6 and figure 7.7 respectively.

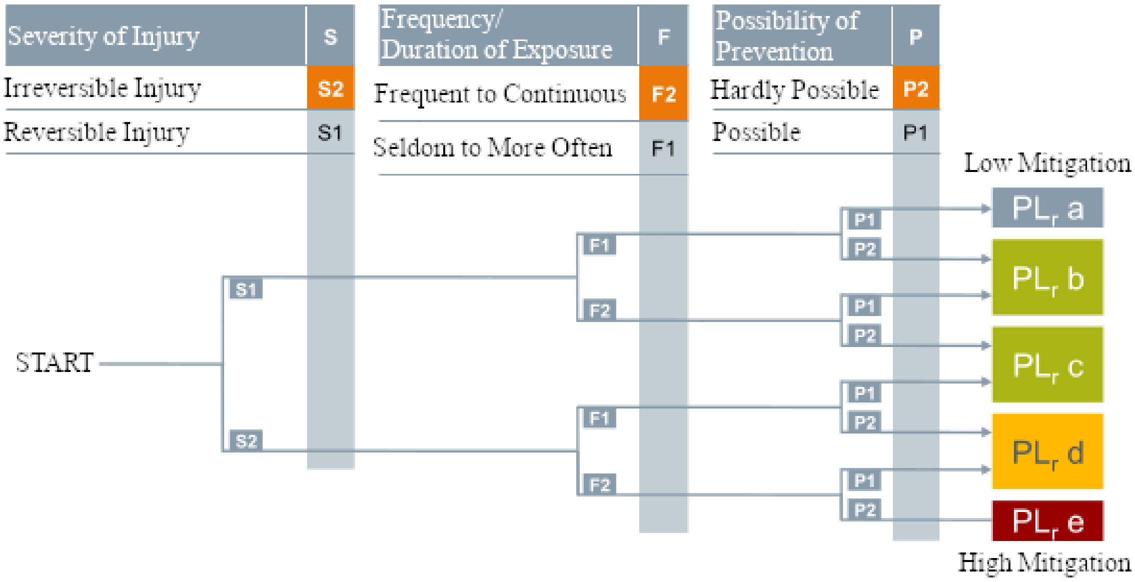
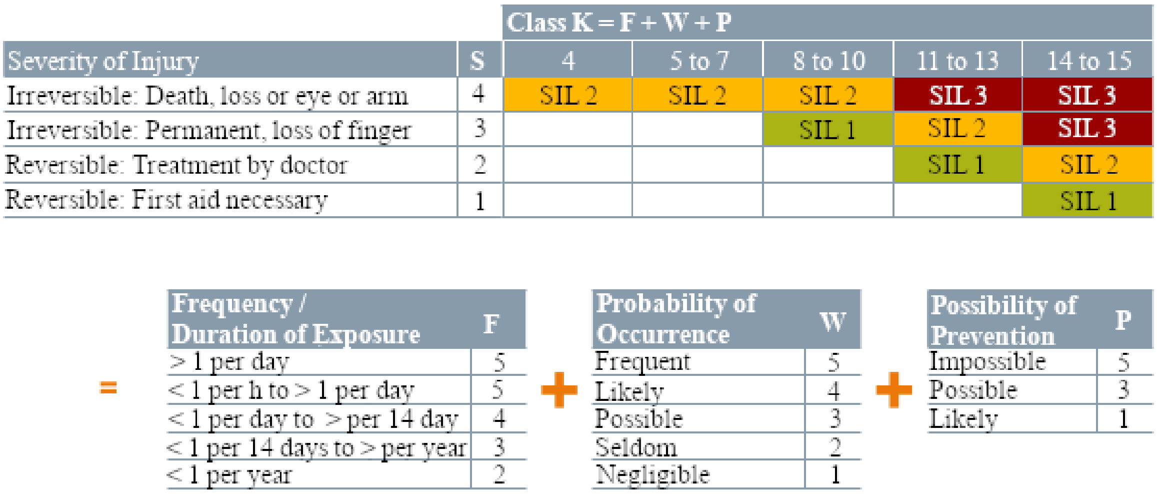


Figure 7.6: Requirements According to ISO 13849-1[3]

[3] figure adapted from Siemens TIA Safety [43]



		Class K = F + W + P				
Severity of Injury	S	4	5 to 7	8 to 10	11 to 13	14 to 15
Irreversible: Death, loss or eye or arm	4	SIL 2	SIL 2	SIL 2	SIL 3	SIL 3
Irreversible: Permanent, loss of finger	3			SIL 1	SIL 2	SIL 3
Reversible: Treatment by doctor	2				SIL 1	SIL 2
Reversible: First aid necessary	1					SIL 1

=

Frequency / Duration of Exposure	F
> 1 per day	5
< 1 per h to > 1 per day	5
< 1 per day to > per 14 day	4
< 1 per 14 days to > per year	3
< 1 per year	2

+

Probability of Occurrence	W
Frequent	5
Likely	4
Possible	3
Seldom	2
Negligible	1

+

Possibility of Prevention	P
Impossible	5
Possible	3
Likely	1

Figure 7.7: Requirements According to IEC 62061[4]

There are two safety systems on the AGV, listed in section 7.4.2, namely the E-Stop system and the SICK S300 Mini Remote scanners. The SICK safety scanner has a SIL rating of SIL 2 and a performance level of PL d. While the E-Stops, which were wired in 1oo2, have a SIL rating of SIL 3 and thus an equivalent performance level of PL e.

Although a deadman switch was included in the AGV pendant for manual mode, this switch is not connected to a safety input on the PLC and, as such, cannot be considered "safe". Thus, it does not count towards the safety rating of the AGV.

Table 7.6 describes the SIL rating required from the risk rating in table 7.2, this was done to ensure that the technical measures taken in section 7.4.2 were done with components of an appropriate SIL level.

[4] figure adapted from Siemens TIA Safety [43]

Table 7.6: Component SIL Level Requirements

Hazard	**Safe Design Risk Rating**	**SIL Requir.**
Crushing risk when payload is load or offloaded onto AGV	3A	SIL 2
Crushing risk if AGV pins personal against a wall or object	4B	SIL 2
Pushing risk if AGV collides with person	4B	SIL 2
Shearing risk if appendages get caught in AGV when moving	4C	SIL 2
Abrading risk if individual dragged by AGV	4C	SIL 2
Cutting risk if appendages caught in drive train	3C	SIL 2
Shearing risk if appendages caught in drive train	3C	SIL 2
Crushing risk if appendages caught in drive train	3C	SIL 2
Electrocution risk if individual touches exposed terminals	4D	SIL 1
Abrading risk if appendages caught in machinery	3C	SIL 2

As illustrated in table 7.6, the SIL rating of most components falls into SIL category 2; there are no SIL 3 requirements. This distribution stems from the low exposure level (i.e. people should not be in the forklift lanes) and the ease of prevention (i.e. enforcing the rules where humans are not allowed to walk or stand in forklift lanes). The riskiest interaction is when the AGV must be loaded or offloaded. The machine must be stopped to do this at a docking zone. So long as workers say in the appropriate sections of the dock (areas with physical rails that prevent the AGV from entering there), they should be safe.

7.6 TIA Safety Report

TIA portal (Siemens PLC programming software) can generate a safety report that confirms whether or not the fail-safe programming on the AGV meets basic safety requirements. This report is attached in appendix R.

7.7 Chapter Conclusion

This chapter analysed the risks associated with the AGV and then mitigated these risks to a safe level using a combination of safe design, technical mitigation measures and user operation restrictions. This analysis was done per ISO 12100. The technical mitigation components were then analysed to ensure that they met the minimum performance level or SIL level to be used as technical mitigation. The technical mitigation was validated per ISO 13849-1 for the performance level evaluation and IEC 62061 for the SIL evaluation.

8 Kinematics

Since the mechanical design of this vehicle drive system is so unique, very little previous work exists to describe the kinematic model, and as such, the kinematic will have to be generated from first principles. Added design features such as the castor offset on the drive wheels further complicated this endeavour.

8.1 Introduction to the Drive Philosophy

In order to produce a holonomic vehicle that is not over-contained (i.e. has more degrees of freedom than is predicted by the mobility formula), using the swerve drive philosophy, at least three motors would need to be used. Using three motors would allow for the creation of two 2 DOF system.

8.2 Forward Kinematics & Drive Unit Considerations

The first task to solve in creating a kinematic model is to create the kinematics of an individual drive unit. Since the traction motors and steering motors share a common axis, the issue of parasitic motion rears its head as described in section 5.3.5.e. If only the traction motor operates at ω_t and the steering motor is held still (ω_s), then the AGV will move forward. However, if the steering motor were to be actuated ($\omega_s \neq 0$), then the traction wheel will rotate as its orientation changes.

There are two strategies for solving this issue. The first is to use a feed-forward controller where an offset is applied to the traction motor angular velocity (described in section 5.3.5.e). The second decoupling method is done using a mechanical compensator in the form of a decoupling gear. This idea is described by Yang et al. [79] in the thesis titled "Decoupled Powered Castor Wheel for Omnidirectional Mobile Platforms". For this, AGV feed-forward compensation in the software will be used.

Using figure 8.1 it is possible to create an equation that describes the behaviour of the coordinate vector of the drive unit (x, y and rotational velocity of the system) from the drive unit's mechanical behaviour (speed of drive wheel and angular velocity of the steering mechanism), so-called "forward kinematics". These vectors will be described as $\dot{\vec{x_w}}$ and $\dot{\vec{u_w}}$ respectively. The formula for this relationship is given in equation 8.1.

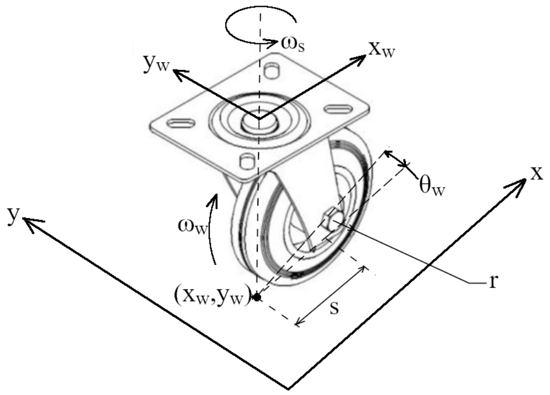


Figure 8.1: Coordinates of a Single Drive Unit Castor

$$\dot{\vec{x_w}} = \boldsymbol{B_w}\dot{\vec{u_w}} \tag{8.1}$$

$\dot{\vec{x_w}}$ = Coordinate Vector of the Drive Unit

$\boldsymbol{B_w}$ = Kinematic Matrix of a Drive Unit

$\dot{\vec{u_w}}$ = Mechanical Behaviour Vector of the Drive Unit

Equation 8.1 can be expanded to:

$$\begin{bmatrix} \dot{x_w} \\ \dot{y_w} \\ \dot{\theta_w} \end{bmatrix} = \begin{bmatrix} r\cos(\theta_w) & -r\sin(\theta_w) \\ r\sin(\theta_w) & s\cos(\theta_w) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_w \\ \omega_s \end{bmatrix} \tag{8.2}$$

x_w	$=$	X Axis Position of the Steering Axis	m
y_w	$=$	Y Axis Position of the Steering Axis	m
r	$=$	Radius of the Wheel	m
s	$=$	Castor Offset of the Wheel	m
θ_w	$=$	Angular Orientation of the Wheel	rad
$\dot{x_w}$	$=$	Velocity of a Drive Unit in the x Direction	$^{m}\!/_{s}$
$\dot{y_w}$	$=$	Velocity of a Drive Unit in the y Direction	$^{m}\!/_{s}$
ω_w	$=$	Angular Velocity of the Wheel	$^{rad}\!/_{s}$
ω_s	$=$	Angular Velocity of the Steering	$^{rad}\!/_{s}$

Equation 8.2 gives the following component equations:

$$\dot{x_w} = r\cos{(\theta_w)}\omega_w - s\sin{(\theta_w)}\omega_s \tag{8.3}$$

$$\dot{y_w} = r\sin{(\theta_w)}\omega_w + s\cos{(\theta_w)}\omega_s \tag{8.4}$$

$$\dot{\theta_w} = \omega_w \tag{8.5}$$

It is to be noted in equations 8.3, 8.4 and 8.5 that there exist common variables between all three equations. Thus if equation 8.3 were to be rearranged to give the formula in terms of "r":

$$r = \frac{\dot{x_w}}{\omega_w\cos{(\theta_w)}} + \frac{s\omega_s\sin{(\theta_w)}}{\omega_s\cos{(\theta_w)}} \tag{8.6}$$

Substituting equation 8.6 into equation 8.4 will yield:

$$\dot{y_w} = \left(\frac{\dot{x_w}}{\omega_w\cos{(\theta_w)}} + \frac{s\omega_s\sin{(\theta_w)}}{\omega_w\cos{(\theta_w)}}\right)\omega_w\sin{(\theta_w)} + s\omega_s\cos{(\theta_w)} \tag{8.7}$$

Simplified:

$$\begin{aligned} \dot{y_w} &= \frac{\dot{x_w}\sin(\theta_w)}{\cos\theta_w} + \frac{s\omega_s\sin^2(\theta_w)}{\cos(\theta_w)} + s\omega_s\cos(\theta_w) \\ \therefore \dot{y_w}\cos(\theta_w) &= \dot{x_w}\sin(\theta_w) + s\omega_s\sin^2(\theta_w) + s\omega_s\cos^2(\theta_w) \\ \therefore \dot{y_w}\cos(\theta_w) &= \dot{x_w}\sin(\theta_w) + s\omega_s \end{aligned} \tag{8.8}$$

Substituting equation 8.5 into equation 8.8 will yield:

$$\dot{x_w}\sin(\theta_w) - \dot{y_w}\cos(\theta_w) + s\dot{\theta_w} = 0 \tag{8.9}$$

If the Pfaffian contraint matrix is generated from equation 8.9 as shown in equation 8.10, where q is the configuration $q = (x_w,\ y_w,\ \theta_w)$ of the point shown in figure 8.1 which can be assumed to be derived from a set of loop closure equations that state the final position of the point q must coincide with the starting point.

$$\begin{aligned} \boldsymbol{A(q)}\dot{q} &= 0 \\ \therefore \boldsymbol{A(q)}\dot{q} &= \boldsymbol{A(q)}\begin{bmatrix}\dot{x_w}\\ \dot{y_w}\\ \dot{\theta_w}\end{bmatrix} = 0 \\ \implies \boldsymbol{A(q)}\dot{q} &= \begin{bmatrix}\sin(\theta_w) & -\cos(\theta_w) & s\end{bmatrix}\begin{bmatrix}\dot{x_w}\\ \dot{y_w}\\ \dot{\theta_w}\end{bmatrix} = 0 \end{aligned} \tag{8.10}$$

The velocity constraint given in equation 8.10 $\left(\begin{bmatrix}\sin(\theta_w) & -\cos(\theta_w) & s\end{bmatrix}\right)$ cannot be integrated to give a an equivalent configuration constraint matrix. Thus, the system will be non-holonomic in nature, with equation 8.9 representing the non-holonomic constraint of the system. This has the effect of reducing the space of the possible velocities of the drive unit but does not reduce the space of possible configurations (i.e. positions in real-space).

8.3 Inverse Kinematics of the Drive Units

In order to control the final behaviour of the drive unit, values need to be calculated for the steering and traction values. This calculation can be done by creating an inverse kinematic model for the drive unit from equation 8.1. Thus:

$$\dot{\vec{u_w}} = (\boldsymbol{B_w})^{-1}\,\dot{\vec{x_w}} \tag{8.11}$$

To determine the inverse of $\boldsymbol{B_w}$ in equation 8.11 (${\boldsymbol{B_w}}^{-1}$), according to Niku [80] the following steps are taken:

1. Calculate the determinate of the matrix
2. Transpose the matrix
3. Replace each element of the transposed matrix by its own minor
4. Divide the converted matrix by the determinate

However, $\boldsymbol{B_w}$ is not a square matrix, and as such, the determinate cannot be easily found. Thus a new strategy must be used called the pseudo-inverse or "Moore-Penrose inverse" method must be used [50]. According to Dresden [81] the pseudo-inverse matrix for $\boldsymbol{B_w} \in \Bbbk^{m \times n}$ can be defined as $\boldsymbol{B_w^+}$, where:

$$\boldsymbol{B_w^+} = \left(\boldsymbol{B_w^*}\boldsymbol{B_w}\right)^{-1}\boldsymbol{B_w^*} \tag{8.12}$$

If $\Bbbk = \mathbb{R}$ then the Hermitian transpose $\boldsymbol{B_w^*}$ is equivalent to the standard transpose $\boldsymbol{B_w^T}$, Thus equation 8.12 becomes:

$$\boldsymbol{B_w^+} = \left(\boldsymbol{B_w^T}\boldsymbol{B_w}\right)^{-1}\boldsymbol{B_w^T} \tag{8.13}$$

Thus using the pseudo-inverse matrix in equation 8.11 in place of the true inverse will yield equation 8.14:

$$\begin{aligned} \dot{\vec{u_w}} &= \left[\left(\boldsymbol{B_w^T}\boldsymbol{B_w}\right)^{-1}\boldsymbol{B_w^T}\right]\dot{\vec{x_w}} \\ &= \begin{bmatrix} \frac{1}{r}\cos\theta_w & \frac{1}{r}\sin\theta_w & 0 \\ -\frac{s}{s^2+1}\sin\theta_w & \frac{s}{s^2+1}\cos\theta_w & \frac{1}{s^2+1} \end{bmatrix} \begin{bmatrix} \dot{x_w} \\ \dot{y_w} \\ \dot{\theta_w} \end{bmatrix} \end{aligned} \tag{8.14}$$

As the system is overdetermined [50], it is very difficult to determine values for mechanical behaviour vector ($\dot{\vec{u_w}}$). Thus to generate a stable solution for the desired coordinate vector ($\dot{\vec{x_w}}$), we can reduce the number of controlled coordinates of the coordinate vector by the number of non-holonomic constraints [50]. Since equation 8.9 has only one non-holonomic constraint (i.e. one formula), the coordinate vector can be reduced by one term to give a system that is not overdetermined.

As euclidean space is easier to work with the rotational term in equation 8.2 will be released (i.e. control of the term $\dot{\theta_w}$ will be given up). This will change equation 8.2 into the form given in equation 8.15:

$$\begin{bmatrix}\dot{x_w}\\ \dot{y_w}\end{bmatrix} = \begin{bmatrix} r\cos(\theta_w) & -s\sin(\theta_w)\\ r\sin(\theta_w) & s\cos(\theta_w)\end{bmatrix}\begin{bmatrix}\omega_w\\ \omega_s\end{bmatrix} \tag{8.15}$$

Thus,

$$\begin{aligned}\dot{\vec{x_{w0}}} &= \boldsymbol{B_{w0}}\dot{\vec{u_w}}\\ \dot{\theta_w} &= -\frac{1}{s}\dot{x_w}\sin(\theta_w) + \frac{1}{s}\dot{y_w}\cos(\theta_w)\end{aligned} \tag{8.16}$$

Where in equation 8.16, $\dot{\theta_w}$ is an internal variable that is described in terms of $\dot{x_w}$ and $\dot{y_w}$ thanks to the non-holonomic constraint discovered in equation 8.9. If the inverse kinematics for the new equation described in equation 8.16 is evaluated utilising the four rules defined by Niku [80] and given previously in this section, then equation 8.17 will result:

$$\begin{aligned}\dot{\vec{u_w}} &= \boldsymbol{B_{w0}}^{-1}\dot{\vec{x_{w0}}}\\ &= \begin{bmatrix}\frac{1}{r}\cos(\theta_w) & \frac{1}{r}\sin(\theta_w)\\ -\frac{1}{s}\sin(\theta_w) & \frac{1}{s}\cos(\theta_w)\end{bmatrix}\begin{bmatrix}\dot{x_w}\\ \dot{y_w}\end{bmatrix}\end{aligned} \tag{8.17}$$

If equation 8.17 is used, the drive unit should accurately reproduce the behaviour of a castor wheel; this was also proposed by Wada et al. [50]. Thus, there should be no conflicts between the behaviour of the driven casters in the "drive units" and the uncontrolled casters in the "castor units".

8.4 Kinematics of the Swerve Drive AGV

The kinematic model can be expanded to control the entire AGV using the kinematic models developed for the drive units. Since the AGV only has two drive units, only these units' wheels will be considered in the model. The uncontrolled castor units will be ignored. The AGV has a total of 3 DOF (degrees of freedom), two in translation (in the x and y direction) and one in rotation (about the z-axis); see section 5.1 for more clarity. Given the available 3 DOF, the AGV will need at least two drive units to control the body of the AGV relative to the universal origin. As previously mentioned in this thesis, the AGV will have the bare minimum of 2 controlled drive units.

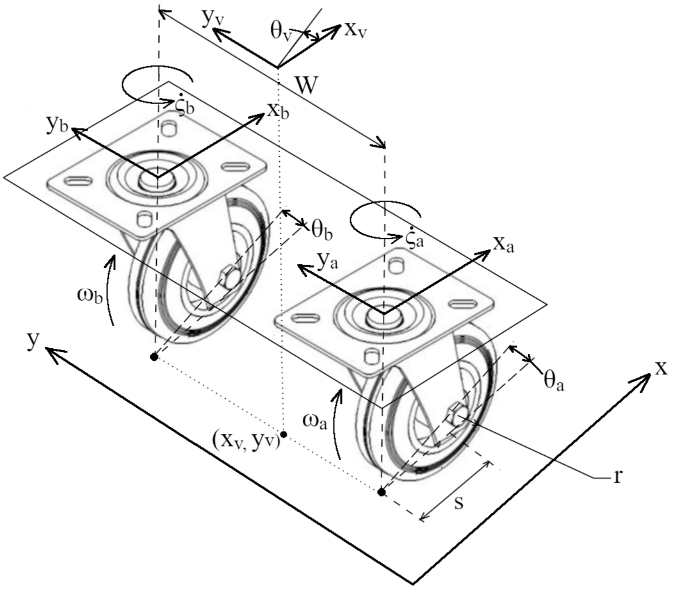


Figure 8.2: Coordinates of the AGV and It's Two Drive Units

The AGV's body and two drive units are represented in figure 8.2. Note that this diagram does not show the actual physical layout of the vehicle but rather a simplified layout with the drive units assumed to share a common axis, in this case, the y-axis. This simplification is done to simplify the mathematics as the distance between the wheels will not need to be broken into x and y-components. This choice can be justified as the AGV is an omnidirectional vehicle whose front or positive x-axis is entirely arbitrary; thus, a front is designated later (which it will be). The vehicle axis can be rotated by a determined amount about the z-axis using a transform. This simplification was proposed by Wada et al. [50]

In figure 8.2, the two drive units are labelled A and B; the origin of the AGV is set at the midpoint between the two drive units, which are spaced a distance "W" apart. Since each drive unit receives an $\dot{x_w}$ and $\dot{y_w}$ component of a control coordinate vector, it is necessary to differentiate between the two coordinate vectors. This adaptation is defined in equation 8.18.

$$
\begin{aligned}
\dot{x_{w_a}} &= \dot{x_a} \\
\dot{x_{w_b}} &= \dot{x_b} \\
\dot{y_{w_a}} &= \dot{y_a} \\
\dot{y_{w_b}} &= \dot{y_b} \\
\omega_{w_a} &= \xi_a \\
\omega_{w_b} &= \xi_b
\end{aligned}
\tag{8.18}
$$

$\dot{x_{w_a}}$	=	Velocity of Drive Unit A in the x Direction	$^{m}/_{s}$
$\dot{x_{w_b}}$	=	Velocity of Drive Unit B in the x Direction	$^{m}/_{s}$
$\dot{y_{w_a}}$	=	Velocity of Drive Unit A in the y Direction	$^{m}/_{s}$
ω_{w_a}	=	Steering Angular Velocity Drive Unit A	$^{rad}/_{s}$
ω_{w_b}	=	Steering Angular Velocity Drive Unit B	$^{rad}/_{s}$

The forward kinematic equation to describe the AGV body, as shown in figure 8.2, is given in equation 8.19:

$$\dot{\vec{x_v}} = \boldsymbol{B_v}\dot{\vec{u_v}} \tag{8.19}$$

The equations of motion for the AGV's body, which are necessary to develop matrix $\boldsymbol{B_v}$ in equation 8.19 are developed with the aid of the free body diagram shown in figure 8.3.

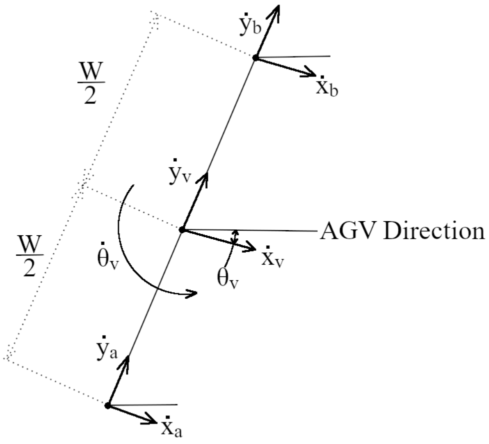


Figure 8.3: Free Body Diagram of the AGV and It's Two Drive Units

Equation of motion for translation in the x-direction:

$$\dot{x_v} = \dot{x_a} + \dot{x_b} \tag{8.20}$$

Equation of motion for translation in the y-direction:

$$\dot{y_v} = \dot{y_a} + \dot{y_b} \tag{8.21}$$

Equation of motion for rotation about the z-direction:

$$\dot{\theta_v} = \dot{\theta_{v_{x_a}}} + \dot{\theta_{v_{y_a}}} - \dot{\theta_{v_{x_b}}} - \dot{\theta_{v_{y_b}}} \tag{8.22}$$

$\dot{\theta_{v_{x_a}}}$ = AGV centroidal angular velocity due to x-component velocity of drive unit A

$\dot{\theta_{v_{y_a}}}$ = AGV centroidal angular velocity due to y-component velocity of drive unit A

$\dot{\theta_{v_{x_b}}}$ = AGV centroidal angular velocity due to x-component velocity of drive unit B

$\dot{\theta_{v_{y_b}}}$ = AGV centroidal angular velocity due to y-component velocity of drive unit B

Where:

$$\begin{aligned}
\dot{\theta_{v_{x_a}}} &= \dot{x_a}\cos(\theta_v)\left(\frac{2}{W}\right) \\
\dot{\theta_{v_{y_a}}} &= \dot{y_a}\sin(\theta_v)\left(\frac{2}{W}\right) \\
\dot{\theta_{v_{x_b}}} &= -\dot{x_b}\cos(\theta_v)\left(\frac{2}{W}\right) \\
\dot{\theta_{v_{y_b}}} &= -\dot{y_b}\sin(\theta_v)\left(\frac{2}{W}\right)
\end{aligned} \tag{8.23}$$

The equations of motion developed in equation 8.20, equation 8.21, equation 8.22 and 8.23 are utilised to generate matrix $\boldsymbol{B_v}$ in equation 8.19, to give equation 8.24.

$$\begin{bmatrix} \dot{x_v} \\ \dot{y_v} \\ \dot{\theta_v} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{W}(\cos\theta_v) & \frac{1}{W}\sin(\theta_v) & -\frac{1}{W}\cos(\theta_v) & -\frac{1}{W}\sin(\theta_v) \end{bmatrix} \begin{bmatrix} \dot{x_a} \\ \dot{y_a} \\ \dot{x_b} \\ \dot{y_b} \end{bmatrix} \tag{8.24}$$

The inverse matrix for $\boldsymbol{B_v}$ is not square, so once again, the pseudo inverse will have to be calculated as described in equation 8.12 and equation 8.13. Thus:

$$\begin{aligned} \dot{\vec{u_v}} &= \left(\boldsymbol{B_v}\right)^{-1} \dot{\vec{x_v}} \\ &= \left[\left(\boldsymbol{B_v^T B_v}\right)^{-1} \boldsymbol{B_v^T}\right] \dot{\vec{x_v}} \end{aligned} \tag{8.25}$$

Using the online mathematical tool Wolfram|Alpha [82] the pseudo-inverse matrix can efficiently be found, as recorded in equation 8.26. An extract from the Wolfram|Alpha Website can be found in the appendix S.

$$\dot{\vec{u_v}} = \begin{bmatrix} \dot{x_a} \\ \dot{y_a} \\ \dot{x_b} \\ \dot{y_b} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \frac{W}{2}\cos(\theta_v) \\ 0 & 1 & \frac{W}{2}\sin(\theta_v) \\ 1 & 0 & -\frac{W}{2}\cos(\theta_v) \\ 0 & 1 & -\frac{W}{2}\sin(\theta_v) \end{bmatrix} \begin{bmatrix} \dot{x_v} \\ \dot{y_v} \\ \dot{\theta_v} \end{bmatrix} \tag{8.26}$$

8.5 Control System Strategy

To control the AGV using the previously developed kinematics, it is necessary first to develop a strategy to implement them.

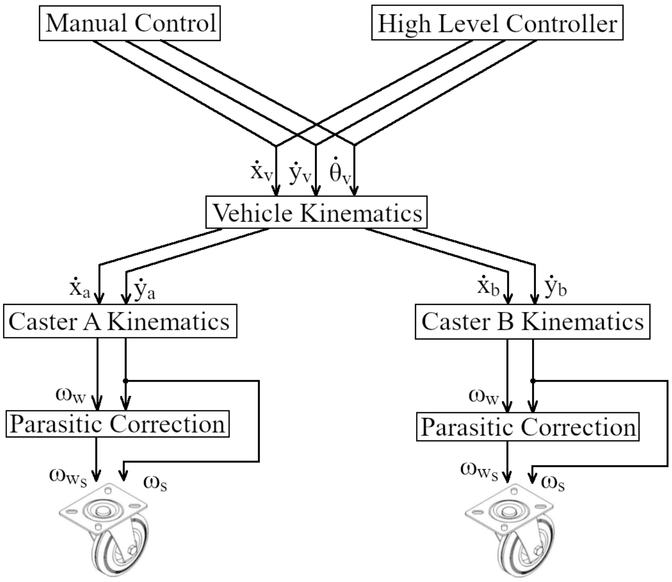


Figure 8.4: Control System Strategy

From equation 8.26 it is possible to generate the velocity vectors for drive units A and

B from the desired AGV's x, y velocities and angular velocity. These values can be fed into the appropriate drive unit to control accurately each unit's desired wheel angular velocity and steering angular velocity. This idea was graphically explained in figure 8.4.

8.6 Chapter Conclusion

This chapter outlined the kinematic model needed to make the AGV function. The forward kinematics allows the high-level control system or manual control system to drive the AGV by sending an x-component velocity, y-component velocity and yaw rate for the centroid of the AGV. These values are translated into a steering speed (not position) and traction speed of each of the two drive units. A reverse kinematic model was also created to deduce the position and orientation of the AGV through integration.

9 Programming and Future ROS Integration

This section mainly focuses on the PLC code needed to operate the AGV. Included in this chapter is a small section on how it attaches a higher level ROS system to the AGV, but this implementation is outside the scope of this research.

9.1 Programming Overview

The control loop strategy described in figure 8.4 is expanded upon in this section to make this system functional with real-world code. The control kinematics in section 8 will also be reduced down to a set of formulae that can be implemented in a PLC. The control loop in the AGV is illustrated in figure 9.1.

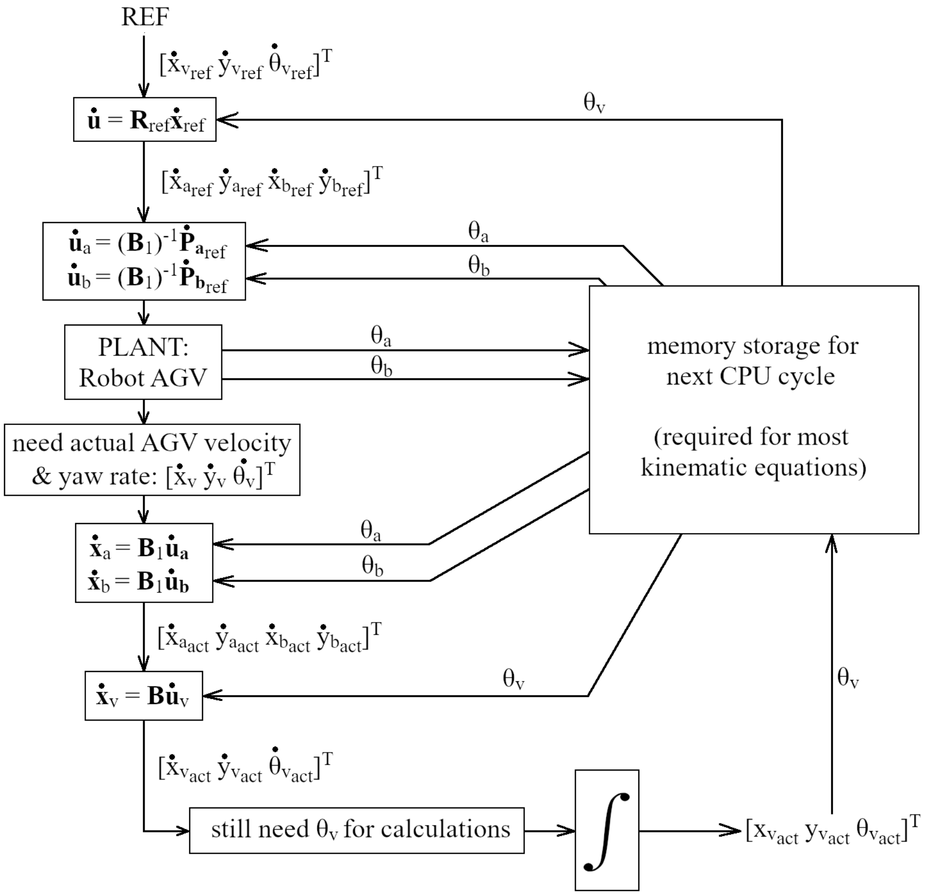

REF
$[\dot{x}_{v_{ref}}\ \dot{y}_{v_{ref}}\ \dot{\theta}_{v_{ref}}]^T$
$\dot{\mathbf{u}} = \mathbf{R}_{ref}\dot{\mathbf{x}}_{ref}$
θ_v
$[\dot{x}_{a_{ref}}\ \dot{y}_{a_{ref}}\ \dot{x}_{b_{ref}}\ \dot{y}_{b_{ref}}]^T$
$\dot{\mathbf{u}}_a = (\mathbf{B}_1)^{-1}\dot{\mathbf{P}}_{\mathbf{a}_{ref}}$
$\dot{\mathbf{u}}_b = (\mathbf{B}_1)^{-1}\dot{\mathbf{P}}_{\mathbf{b}_{ref}}$
θ_a
θ_b
PLANT:
Robot AGV
θ_a
θ_b
memory storage for
next CPU cycle
(required for most
kinematic equations)
need actual AGV velocity
& yaw rate: $[\dot{x}_v\ \dot{y}_v\ \dot{\theta}_v]^T$
$\dot{\mathbf{x}}_a = \mathbf{B}_1\dot{\mathbf{u}}_\mathbf{a}$
$\dot{\mathbf{x}}_b = \mathbf{B}_1\dot{\mathbf{u}}_\mathbf{b}$
θ_a
θ_b
$[\dot{x}_{a_{act}}\ \dot{y}_{a_{act}}\ \dot{x}_{b_{act}}\ \dot{y}_{b_{act}}]^T$
$\dot{\mathbf{x}}_v = \mathbf{B}\dot{\mathbf{u}}_v$
θ_v
θ_v
$[\dot{x}_{v_{act}}\ \dot{y}_{v_{act}}\ \dot{\theta}_{v_{act}}]^T$
still need θ_v for calculations
$\int$
$[x_{v_{act}}\ y_{v_{act}}\ \theta_{v_{act}}]^T$

Figure 9.1: Kinematic Control Startergy

Since the AGV is a complex machine with multiple controllers, each section will revolve around a different controller. The primary controllers in this system are:

- Siemens s7-1512SP PLC
- Software PLC on the IPC427E
- WinCC SCADA system on IPC427E
- Siemens Sinamics V90 Servo Drives
- Fetso CMMS-ST Stepper Drives

Note: due to the complexity of the code, no code will be included in this chapter. Instead, it will be included as appendices for each controller with the description of operation included in this chapter.

The code used on this AGV is written in five languages; however, the number of languages used on this PLC will likely increase to seven when ROS is implemented. A breakdown of the languages used is listed in the sections that follow:

Ladder (LAD)

Ladder code was used on the IPC and PLC for basic code operations. Ladder is a straightforward language to troubleshoot, so it was used for much of the basic coding. It also works on an assignment principle rather than a set/reset principle like most higher-level text-based languages making it appropriate for use with a PLC.

Structure Control Language (SCL)

SCL is a Siemens proprietary language similar to C++ in syntax and functionality. This language was used in the PLC and software PLC whenever complicated code needed to be implemented. SCL has the advantage of implementing complex structures and algorithms; however, it is difficult to troubleshoot (compared to Ladder) and is considered a dark art by most technicians (only really be understood by programmers and engineers). This complexity makes routine maintenance on machines that only use SCL complex for most factories without a qualified code jockey; thus, SCL was used sparingly in the research project.

S7-GRAPH

S7-GRAPH is another Siemens proprietary language (other PLC manufacturers have similar languages). GRAPH is a language developed to control machines with a defined sequential process. When a machine enters a defined step in the process, only the variables associated with that step will be read or written; any variables outside of that step are ignored even if they change behaviour. Essentially the language is a finite state machine that can be compiled. This language was used on the IPC and PLC for wheel alignment, booting the system, and shutting down the system. All these processes have a definite sequence of steps to be executed in a known order. Although this language is foreign to programmers, it is well known by engineers and technicians, making it appropriate for factory machines.

VBScript

VBScript is a Microsoft language. This language is the preferred language for SCADA scripting, and as such, it was used primarily in the WinCC SCADA system that runs on the IPC. The author also used the language to run custom commands on the Windows 7 VM on the IPC.

BATCH script

BATCH scripting is a Microsoft language used only on the Windows 7 VM of the IPC. The author used this language in conjunction with VBScript to control the Windows OS via Powershell.

Python

A high-level interpreter language. This language is to be used (not implemented at the time of the report) to create ROS nodes. Thus, it is to be used with a future Debian-based Linux VM (the author would recommend either Kubuntu or PopOS) that runs alongside Windows 7.

C++

C++ is a compiled language that can be used in place of Python to create some ROS nodes. Since C++ is compiled, it will be more efficient than equivalent Python code. However, it will take longer to implement due to do the fact that it needs to be compiled.

The AGV has seven operation modes; these are summarised in table 9.1.

Table 9.1: AGV Modes

Mode	**Description**
Auto Mode	Receives centroidal data from ROS system
Manual Mode	Calculates centroidal data from analog "accelerator", analog "steering wheel" and analog "strafe direction lever" on pendant
Commissioning Mode	Allows for jog of individual motors using dip switches on pendant & accelerator for direction/ speed
Homing Mode	Cannot be controller by user, evoked when AGV boots or when "home wheels" selected from HMI
Move Units (HMI only)	Similar to "commissioning mode" but run from HMI rather than pendant
Move AGV (HMI only)	Similar to "manual mode" but run from HMI rather than pendant
Testing Mode	Cannot be controlled by user, uses recorded data from previous "manual mode" to control the AGV. This mode is used to repeat the exact input condition to the AGV in-order to repetitively test its response

"Centroidal data", in table 9.1, refers to the x-component velocity, y-component velocity and yaw rate of the centroid. The kinematic model calculates the behaviour of the steering and traction motors from this data.

9.2 PLC Code

The Siemens PLC s7-1512SP is the low-level controller for this system. It is responsible for actuating the drives per the desired motion of the centroid of the AGV.

9.2.1 Overview

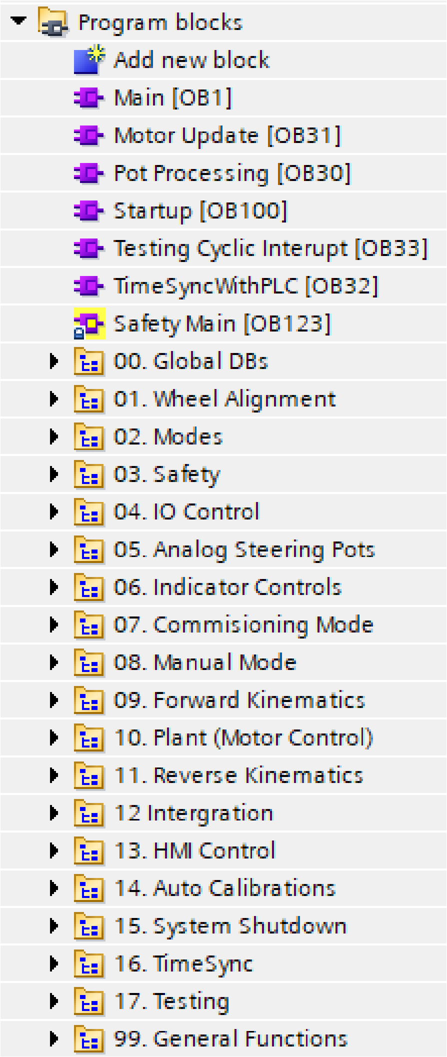
In Siemens' TIA Portal, there are four code blocks available. These are Organisational blocks (OBs), Functions (FCs), Function Blocks (FBs) and Data Blocks (DBs). OBs are used to control the behaviour of the PLC's operating system as, unlike microcontrollers, the programmer does not have access to edit the OS. These OBs include boot OBs, cycle OBs (runs user code) and interrupt OBs. FCs contain code but have no static memory; only temporary variables can be used. FBs contain static memory. Finally, DBs are blocks that only contain static variables.

The program was ordered into folders in TIA; this was done to simplify the code layout and make maintenance more manageable. These aforementioned folders are:

- **00. Global DBs** *Variables that must be accessed in multiple blocks*
- **01. Wheel Alignment** *Code that homes steering mechanism*
- **02. Modes** *Code to select modes as given in table 9.1*
- **03. Safety** *Safety compliant code*
- **04. IO Control** *FCs that pass/receive variables to the physical IO via DBs*
- **05. Analog Steering Pots** *Code to process & scale raw analog values*
- **06. Indicator Controls** *Controls for indicator LEDs, Stack Lamps, Sirens, etc.*
- **07. Commissioning Mode** *Generates control vectors for commissioning mode*
- **08. Manual Mode** *Generates control vectors for manual mode from pendant values*
- **09. Forward Kinematics** *Generates steering and traction speed values for motors*
- **10. Plant (Motor Control)** *Sends and receives data from motor control interrupt*
- **11. Reverse Kinematics** *Feedback from actual motor behaviour used to calculate actual centroidal behaviour*
- **12. Integration** *Integrates various speed values to get position values*
- **13. HMI Control** *FCs that pass/receive data from HMI via DBs*
- **14. Auto Calibrations** *Code to calibrate absolute potentiometers and encoders, activated from maintenance screen on HMI*

- **15. System Shutdown** *Code to gracefully shut down all systems on the AGV*
- **16. Time Sync** *Code to synchronise clocks between devices*
- **17. Testing** *Code to run testing mode*
- **18. Automatic Mode** *Code to run automatic mode*
- **99. General Functions** *General reusable functions such as degrees (°) to radians converter, etc.*

A screen capture of these code folders described previously is shown in figure 9.2.



Program blocks
Add new block
Main [OB1]
Motor Update [OB31]
Pot Processing [OB30]
Startup [OB100]
Testing Cyclic Interupt [OB33]
TimeSyncWithPLC [OB32]
Safety Main [OB123]
00. Global DBs
01. Wheel Alignment
02. Modes
03. Safety
04. IO Control
05. Analog Steering Pots
06. Indicator Controls
07. Commisioning Mode
08. Manual Mode
09. Forward Kinematics
10. Plant (Motor Control)
11. Reverse Kinematics
12 Intergration
13. HMI Control
14. Auto Calibrations
15. System Shutdown
16. TimeSync
17. Testing
99. General Functions

Figure 9.2: Screen Capture of Code Folders for the PLC

Not all of the code used in the PLC is run as part of the main code loop in OB1 (often referred to as the "main" in other IDEs). Several systems are run solely as cyclic interrupts. The can be seen in figure 9.2, they are:

- Motor Update [OB31] (50ms)
- Pot Processing [OB30] (50ms)
- Testing Cyclic Interrupt [OB33] (25ms)
- TimeSyncWithPLC [OB32] (60s)

"Main [OB1]" is the cycle (not to be confused with cyclic) block, and "Startup [OB100]" is the boot code. Since all of the interrupt OBs are cyclic OBs, their cyclic time was listed in curved brackets in the previous itemised list.

OB1 is the cycle OB and is responsible for running non-time deterministic code, which is the majority of the code. Since OB1 itself does not run on a deterministic time scale but will instead run at the fastest possible CPU cycle speed. This cycle time was approximately 2ms - 3ms.

"Motor Update [OB31]" is responsible for sending the Profidrive telegrams to the V90 servo and CMMS-ST stepper drives. This communication must be time deterministic and should be at a cycle time higher than the 2 ms to 3 ms of OB1. The cycle time for this block is 50 ms; due to the operation of the block, the Profi-drive telegram is sent every second execution. Thus, the update cycle of the drives will be every 100 ms.

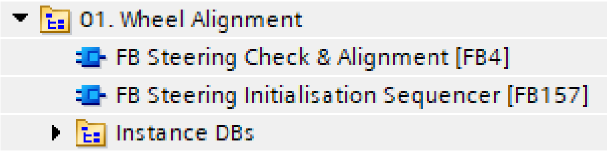
"Pot Processing [OB30]" is responsible for reading and scaling the analogue inputs. Four analogue inputs are read and scaled by this OB. They are the two "endless" potentiometers used for registering the absolute angles of the steering and the two analogue signals generated by the CMMS-ST drives, which give the relative steering angles. The execution cycle of this OB is done every 50 ms, while the sampling is every 100 ms since the sampling occurs every second execution cycle. The reason that there is both a relative and absolute measurement for each drive unit's steering angle is explained in section 9.2.13.

"Testing Cyclic Interrupt [OB33]" is used to either record or stream the control input data to the kinematic model of the AGV for repeatability of testing. I.e. the same input data is streamed to the AGV while the response is recorded to check for deviations in behaviour. This block has an execution interval of 25 ms, while the actual data is recorded or streamed every 50 ms (every second execution cycle).

"TimeSyncWithPLC [OB32]" runs every 60 seconds (1 minute) and at *each* execution syncs the time with the IPC. The IPC is the time master as it can sync its time using the windows VM and an internet-based NTP server. This block was depreciated but was not removed due to it causing issues when disabled (it will take time to disentangle it from the rest of the code properly).

9.2.2 Boot

Booting the AGV is a two-step process that makes use of "Startup [OB100]" and "Main [OB1]". OB100 is used to SET a startup latch. This latch, while true, disables all other functionality of the AGV except for code in OB1 that controls homing the steering. OB1 will execute the homing subroutine via the code contained in the folder "01. Wheel Alignment" (see figure 9.3). This code can be viewed in its entirety in appendix T. Once the wheel alignment has been completed, this subroutine's "done" signal is used to reset the startup latch. More boot actions can be added in the future with their "done" signal being "ANDed" with the wheel alignment "done" signal to reset the startup latch.



- 01. Wheel Alignment
  - FB Steering Check & Alignment [FB4]
  - FB Steering Initialisation Sequencer [FB157]
  - Instance DBs

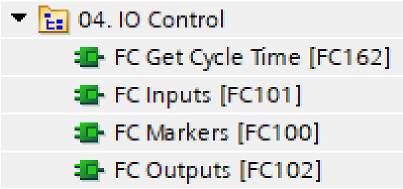
Figure 9.3: Wheel Alignment Code Blocks

The wheel alignment has two options set from a service screen on the HMI. The first option sets whether the wheels are orientated (steering) to the front (zero degree point) or whether the wheels should have the same orientation relative to each other but do not necessarily need to be oriented to the zero point. The second setting prevents the wheel alignment code from running a homing sequence. If homing is allowed and the wheels fail the alignment check, the AGV can move them into the appropriate position. If homing is not allowed and the wheels fail the alignment check, the AGV enters error mode, and the steering must be manually jogged into alignment. These settings were exposed to the user as it allows them to select the most appropriate behaviour for implementation in their factory.

Due to its relative complexity, the wheel alignment subroutine used a GRAPH language sequencer and various SCL code blobs, which were called by the sequencer.

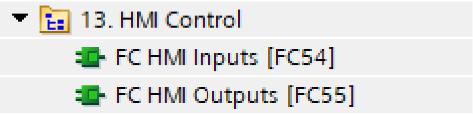
9.2.3 IO interfacing

IO interfacing is run using OB1 via the code contained in the folders "04. I/O Control" (see figure 9.4) and "13. HMI Control" (see figure 9.5). The purpose of the FCs in those folders is to provide a layer of abstraction between the actual hardware addresses in the case of the I/O control or the DB variables passed over the Profinet network in the case of the HMI. This abstraction layer allows for much easier re-wiring or porting of code should the physical hardware configuration change. Since the programmer needs not to hunt down every reference in the code to a specific hardware/HMI address but rather change it once in these FCs.



- 04. IO Control
  - FC Get Cycle Time [FC162]
  - FC Inputs [FC101]
  - FC Markers [FC100]
  - FC Outputs [FC102]

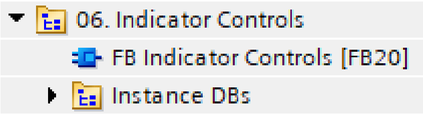
Figure 9.4: I/O Control Code Blocks



- 13. HMI Control
  - FC HMI Inputs [FC54]
  - FC HMI Outputs [FC55]

Figure 9.5: HMI Control Code Blocks

Interfacing with the indicators on the AGV, such as the moving light and stack lamps, is done using the code in the folder "06 Indicator Controls" (see figure 9.6).



- 06. Indicator Controls
  - FB Indicator Controls [FB20]
  - Instance DBs

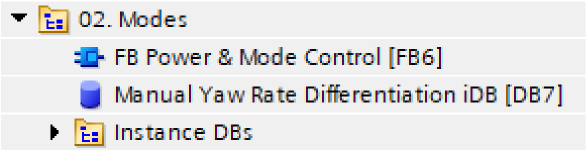
Figure 9.6: Indicator Control Code Blocks

The full code for the I/O interfacing can be found in the appendix T.

9.2.4 Mode Selection

Mode selection is run as part of the main cycle (OB1) and is executed by the code found in "02. Modes". This block is used to choose one of the operating listed in table 9.1. It is also responsible for placing the AGV into ON or OFF mode. The drive train has been disabled when the system is OFF, but the AGV is otherwise powered. The AGV will enter OFF mode after an STO issue has occurred. The system returns to ON mode once the STO fault has been cleared and the fault has been acknowledged.

Acknowledgement can be done via either the pendant or button on the AGV. The "02. Mode" folder is shown expanded in figure 9.7 and the full code can be found in appendix T.



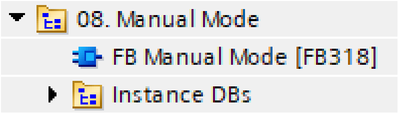
▾ 02. Modes
FB Power & Mode Control [FB6]
Manual Yaw Rate Differentiation iDB [DB7]
▸ Instance DBs

Figure 9.7: Mode Selection Blocks

Once a mode is selected, any code not relevant to that mode is jumped over in OB1, using the jump command. This jumping was done to save CPU cycle time, as returning a "false" would take the same cycle time as returning a "true". Hence, it is more efficient to ignore the code entirely rather than waste time generating the "false" logic result.

9.2.5 Manual Mode

Manual mode allows the operator to control the AGV via the pendant. Three potentiometers are used as control inputs in this mode. The first controls the centroidal speed of the AGV (this is a magnitude value). The second acts as a sort of steering wheel which controls the Ackerman steering of the AGV. Finally, the last potentiometer controls the strafe angle of the AGV (essentially, the heading offset). The code used to control the manual mode is located in the folder "08. Manual Mode" and is illustrated in figure 9.8. A full breakdown of the code can be found in appendix T.



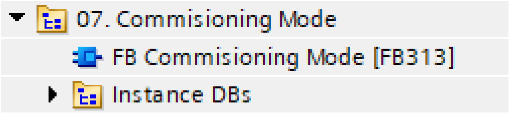
▾ 08. Manual Mode
FB Manual Mode [FB318]
▸ Instance DBs

Figure 9.8: Manual Mode Blocks

Both the strafe angle potentiometer and Ackerman steering potentiometer modulate the speed potentiometer. Combining these three potentiometer values generates three velocities of the centroid used by the kinematics to generate steering and traction speeds. These speeds are the x-component velocity, y-component velocity and yaw rate. See section 5.1 for clarity on these coordinates.

9.2.6 Commissioning Mode

Commissioning mode allows low-level user control of the four motors on the AGV (two steering motors and two traction motors) from the pendant. Control of a motor is selected via one of four DIP switches on the pendant; these switches are interlocked so that only one motor can be jogged at any given moment. Once a motor is selected, its speed and direction are controlled via the speed potentiometer normally used in manual mode. The potentiometer midpoint is designated (in this mode only) as 0 $^{r}/_{min}$, with the extremities of the pot designated as max speed counter-clockwise or max speed clockwise. The code used for the process is contained in the folder "07. Commissioning Mode", which is illustrated in figure 9.9 and is fully detailed in appendix T.



07. Commisioning Mode
FB Commisioning Mode [FB313]
Instance DBs

Figure 9.9: Commissioning Mode Blocks

The commissioning mode block is also used for the mode "Move Unit" (see table 9.1), an HMI only mode, as this mode performs essentially the same function as the commissioning mode that is run from the pendant. It acts as a stand-alone mode from the commissioning mode due to the differences in interaction between the HMI variables and the pendant variables with the AGV. Thus, it was simply easier to implement it as a separate mode.

When the commissioning mode is being run, the kinematics are disabled via jump commands, and the commissioning code directly controls the RPM of the selected motor.

9.2.7 Automatic Mode

Automatic mode is the default mode that the AGV will attempt to enter when the AGV is booted. For safety reasons, the AGV will not begin running immediately after boot but will enter STO mode and requires the system to be acknowledged before operations can begin. Acknowledgement can be performed using either a button on the pendant or the acknowledge push-button located on the AGV's front. An acknowledge button was included on the AGV since the pendant can be detached when the AGV is in automatic mode.

Automatic mode operates by grabbing control information from the software PLC (on the Siemens IPC) and streaming it to the kinematic model. The control information will be generated directly from the SICK NAV 350 to perform "virtual line following" or generated by the ROS system using SLAM. Either way, the control data will consist of:

- Centroidal x-component velocity $^{m}/_{s}$
- Centroidal y-component velocity $^{m}/_{s}$
- Centroidal yaw rate $^{rad}/_{s}$

9.2.8 Forward & Reverse Kinematics

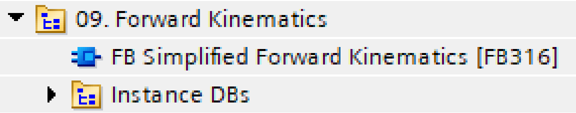
The forward kinematic block takes the following inputs that were generated by either the manual mode or automatic mode code:

- Centroidal x-component velocity [SETPOINT] $^{m}/_{s}$
- Centroidal y-component velocity [SETPOINT] $^{m}/_{s}$
- Centroidal yaw rate [SETPOINT] $^{rad}/_{s}$

and translates them to:

- Unit A steering motor speed [SETPOINT] $^{r}/_{min}$
- Unit B steering motor speed [SETPOINT] $^{r}/_{min}$
- Unit A traction motor speed [SETPOINT] $^{r}/_{min}$
- Unit B traction motor speed [SETPOINT] $^{r}/_{min}$

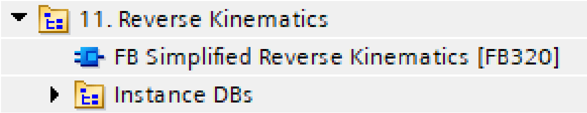
These RPM values are sent cyclically, via OB31, to the V90 and CMMS-ST drives. Note, both sets of drives accept RPM as the unit of measurement and not $^{rad}/_{s}$. The code to perform this function was developed using the mathematics from chapter 8 and is contained in the folder "09. Forward Kinematics", which is illustrated in figure 9.10.



09. Forward Kinematics
FB Simplified Forward Kinematics [FB316]
Instance DBs

Figure 9.10: Forward Kinematics Blocks

The reverse kinematics block uses the actual values returned from the motor encoders to develop the actual centroidal x-component velocity, y-component velocity and yaw rate. These values can then be integrated to find the AGV's actual position. They can also be used in the closed-loop necessary for automatic mode, where the error between the setpoint and actual values needs to be determined. The code for the reverse kinematics is contained in the folder called "11. Reverse Kinematics", which is illustrated in figure 9.11.



11. Reverse Kinematics

FB Simplified Reverse Kinematics [FB320]

Instance DBs

Figure 9.11: Reverse Kinematics Blocks

The full code for both forward and reverse kinematics can be found in appendix T.

9.2.9 Integration

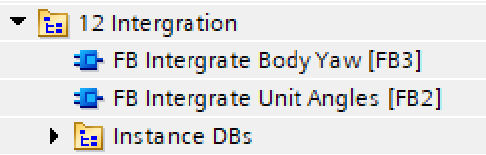
The integration system on the AGV used numeral analysis to approximate the integral of a set of variables. Since this system uses numerical analysis, any variable to be integrated must be sampled at a fixed time interval since this process is time deterministic. The two values sets that are integrated are the *actual* speed of the steppers and the *actual* and *setpoint* yaw rates of the AGV. When these values are integrated with reference to time, the angle of the steppers and the yaw angle is generated. These values are beneficial for closed-loop control and general system monitoring feedback systems. For clarity's sake, the variables and their integrals are listed in table 9.2.

Table 9.2: Integrated Variables

Variable	**Integral**
Setpoint yaw rate ($\dot{\theta}_v$ ACT)	Setpoint yaw angle (θ_v ACT)
Actual yaw rate ($\dot{\theta}_v$ SET)	Actual yaw angle (θ_v SET)
Actual unit A steering angle ($\dot{\theta}_a$ ACT)	Actual angle of the steering (θ_a ACT)
Actual unit B steering angle ($\dot{\theta}_b$ ACT)	Actual angle of the steering (θ_b ACT)

Although the steering angle was calculated using integration, this value was never used and was eventually disabled. It was disabled after a way was found, via analogue channels, to grab the actual steering angle directly from the CMMS-ST drives. Since this is a direct measurement rather than one mathematically deduced, it proved to be more reliable for use with kinematics. The issues with grabbing the actual steering angle are detailed in the section 9.2.13.

All code for integration was run via the cyclic interrupt "Motor Update [OB31]" and is contained in the folder "12. Integration" as shown in figure 9.12.



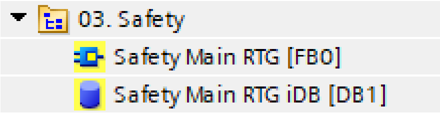
12 Intergration
FB Intergrate Body Yaw [FB3]
FB Intergrate Unit Angles [FB2]
Instance DBs

Figure 9.12: Integration Blocks

The code for these integration blocks can be found in appendix T.

9.2.10 Safety Systems

The safety code is linked to the safety analysis done in chapter 7. This code is run as a separate runtime to the standard code called "Safety Main [OB123]", see figure 9.2. Thus, if the standard code goes into stop mode, the safety systems remain active. OB123 is responsible for executing the safety code, most of which is contained in "Safety Main RTG [FB0]". This code interacts with the fail-safe IOs on the AGV; in this case, it is the inputs from the two E-stops, the SICK S300 Remote Minis (via bit-banging the SICK safety PLC) and the outputs to the drive's STO terminals. "Safety Main RTG [FB0]" can be found in the folder "03. Safety", as illustrated in figure 9.13. The complete code can be found in appendix T.



03. Safety
Safety Main RTG [FB0]
Safety Main RTG iDB [DB1]

Figure 9.13: Safety Code Blocks

9.2.11 System Shutdown

Faults can occur if the system is hard shutdown (power cut) as this can cause corruption of the Windows OS, issues with the PFsense routing tables and issues with the software PLC not properly storing retentive variables. To avoid these issues, the AGV needs to be a soft shutdown. A soft shutdown of the AGV is easier said than done as both the IPC and PFsense router needs to register a shutdown command.

The shutdown system works as follows:

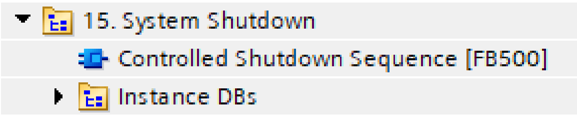
1. Main isolator on AGV turned to off position

    - 220VAC bus cut
    - Inverter told to shutdown via axillary contact 1 on switch
    - Control systems now running off of DC UPS
    - Auxiliary contact 2 on switch tells PLC that main isolator in off position

2. PLC flips router shutdown bit which is shared with SCADA system
    - Shutdown timer1 started
3. SCADA system uses VBScript to open SSH window using PuTTY on Windows 7 VM
4. SCADA logs into PFsense router via SSH using VBScript, sends "poweroff" command
5. Router begins shutdown sequence
6. Shutdown timer1 elapsed, PLC flips IPC shutdown bit which is shared with SCADA system
    - Shutdown timer2 started
7. SCADA uses VBScript to call windows batch file to shutdown IPC
8. SCADA, Windows 7 VM and Software PLC begin shutdown
9. Shutdown timer2 elapsed, IPC and router are fully shutdown at this point
10. PLC triggers UPS shutdown via DQ output
11. UPS shutdown, AGV now fully dead.

All systems are set in boot config or bios (device dependant) to boot "after power recovery," i.e. when power is restored via activating the inverter using the isolator.

The code for this system on the PLC side is contained in the folder "15. System Shutdown", which is illustrated in figure 9.14.



15. System Shutdown
Controlled Shutdown Sequence [FB500]
Instance DBs

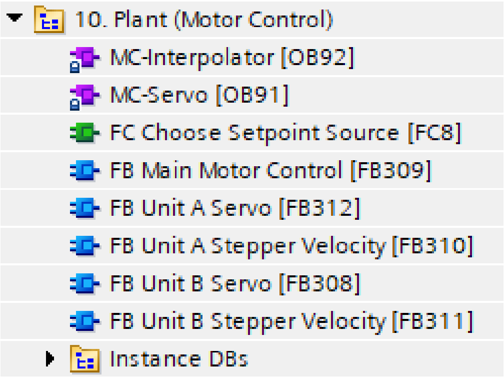
Figure 9.14: System Shutdown Code

The code for this operation was mainly written in GRAPH due to its sequential nature and can be found in appendix T. The VBScripts and batch files can be found in the sections dealing with the IPC.

9.2.12 Motor Control Interrupt

The "motor control interrupt" is responsible for sending the cyclic and acyclic Profidrive telegrams to the drives. This process needs to be time deterministic, and as such, a cyclic interrupt was used, namely, "Motor Update [OB31]" The cyclic interrupt is set to execute every 50 ms but only updates the telegram every 100 ms due to code operation.

These telegrams send/receive the acyclic words that change the drive's state. The telegrams also send the setpoint RPM to the drive and receive the actual RPM as a response using cyclic words. Though there is a shortfall with the Festo stepper drives, due to poor implementation by Festo, the relative position cannot be returned to the PLC from the drive when the drive is in velocity mode (as is the case with the AGV's implementation). The relative position is calculated and exists on the drive; only no provision was made to send this data. The only way the author could retrieve this value for use on the PLC was by passing it through an analogue output on the drive.



10. Plant (Motor Control)
MC-Interpolator [OB92]
MC-Servo [OB91]
FC Choose Setpoint Source [FC8]
FB Main Motor Control [FB309]
FB Unit A Servo [FB312]
FB Unit A Stepper Velocity [FB310]
FB Unit B Servo [FB308]
FB Unit B Stepper Velocity [FB311]
Instance DBs

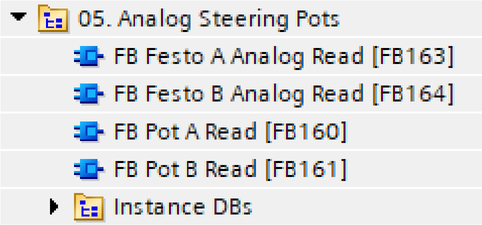
Figure 9.15: Motor Control Interrupt Code

The code of the motor control interrupt is called by OB31 and is contained in the folder "10 Plant (Motor Control)", which is illustrated in figure 9.15 and full described in appendix T.

9.2.13 Potentiometer Reading Interrupt

Since analogue to digital conversion is CPU intensive, performing this process every CPU cycle is unwise. Especially if the analogue value's rate of change is significantly slower than the CPU cycle time, thus, the processing and conversion of the analogue values generated by the potentiometer were only read cyclically, using the cyclic interrupt "Pot Processing [OB30]" every 50 ms. Measuring the pot values using a cyclic interrupt also makes these samples time deterministic, simplifying processes such as integration.

Four analogue values are measured using this cyclic interrupt; they are the two absolute steering position sensors and the two relative steering position sensors sent by the Festo CMMS-ST drives. All four channels are configured to use 0..10 VDC as their measurement range, with the real-world values (in this case, radians) being scaled to this analogue voltage range. The code for this system is contained in the folder "05. Analog Steering Pots", as illustrated in figure 9.16.



- ▼ 05. Analog Steering Pots
  - FB Festo A Analog Read [FB163]
  - FB Festo B Analog Read [FB164]
  - FB Pot A Read [FB160]
  - FB Pot B Read [FB161]
  - ▶ Instance DBs

Figure 9.16: Potentiometer Reading Interrupt Code

A question might be raised as to why it is necessary to capture the relative steering position when the absolute steering position is known. Capturing the relative position is necessary due to how the absolute position measurement was implemented. The absolute position is generated using an endless potentiometer; this was done to save cost as absolute encoders (when compared to relative encoders) are an order of magnitude more expensive.

Using a potentiometer as an absolute encoder has two drawbacks. The first is that there exists a relatively sizeable dead zone when the potentiometer crosses over from the 0V point to the 10V point. Secondly, it was found that the voltage value became unstable at higher RPMs making the values unreliable. These issues made using the absolute position unsuitable for feedback into the kinematic control loop.

Since the relative encoder does not suffer from either of these two shortfalls, it was ideal to use it for feedback on the kinematic calculations. Though there is also an issue with using relative encoders for position measurement, and that is the fact that they will lose their start position between power cycles and assume that the 0-degree position is where were when they are powered "on". This behaviour is not ideal for a system like the steering mechanism with a definite 0-degree position. The relative encoder is homed using the absolute encoder as a homing reference to solve this issue.

If the homing is done slow enough and the potentiometer is oriented so that the dead zone is not at the zero point, the issues associated with using a potentiometer as an absolute encoder are alleviated.

An issue encountered during testing occurred when the relative encoder's 0-degree position was scaled to 0 V, and the 360-degree position was scaled to 10 V. This scaling causes a crossover from 0 V to 10 V at the steering's neutral position. Since the crossover from 0 V to 10 V is not instantaneous, a ramp up/down occurs during the changeover due to real-world electrical behaviour. The control system interprets this as a high-speed 360° rotation which causes the steering to vibrate violently. The solve this issue, the 0-degree position was mapped to 5 V, while the 0 V position was mapped to the 180-degrees. Although the violent vibrations will still occur if the steering attempts to hover around the 180° position, this orientation can be locked out in software, and 180° travel can be achieved by reversing the rotation of the traction wheels when the steering is orientated to 0° instead.

The complete code for this interrupt can be found in appendix T.

9.2.14 Testing Cyclic Interrupt

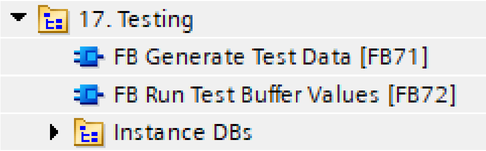
Repeatability is needed to generate useful data for analysis. Such data generation is tricky for a system such as an AGV as each motion, even over the same path, will be unique. The closest that a system such as this can be to "repeatable" is if the system is fed the same control inputs (centroidal x-component velocity, centroidal y-component velocity and yaw rate) over the same path.

This pseudo repeatability was achieved using a cyclic interrupt called "Testing Cyclic Interrupt [OB33]", which ran every 25 ms. This cyclic interrupt was used to perform two separate functions.

The first function was to record the raw analogue data from the pendant potentiometers for speed control, Ackerman steering and strafe angle. This recoding occurred every second cycle (50 ms interval), and the resulting sample was stored in an array. Since this recording mechanism records the pendant pots, the testing mode can only use data recorded from manual mode. The array has 6000 indices, and as such, a recording can last for a maximum of 300 seconds (5 minutes).

The second function is to stream the recoded data back into the manual mode block every 50 ms to spoof usage of the pendant. Since this data can be streamed multiple times into the AGV, the process is highly repeatable.

The code to run this system is contained in the folder "17. Testing" (see figure 9.17 and can be found in its entirety in the appendix T.



```
▼ 17. Testing
      FB Generate Test Data [FB71]
      FB Run Test Buffer Values [FB72]
   ▶ Instance DBs
```

Figure 9.17: Repeating Test Code

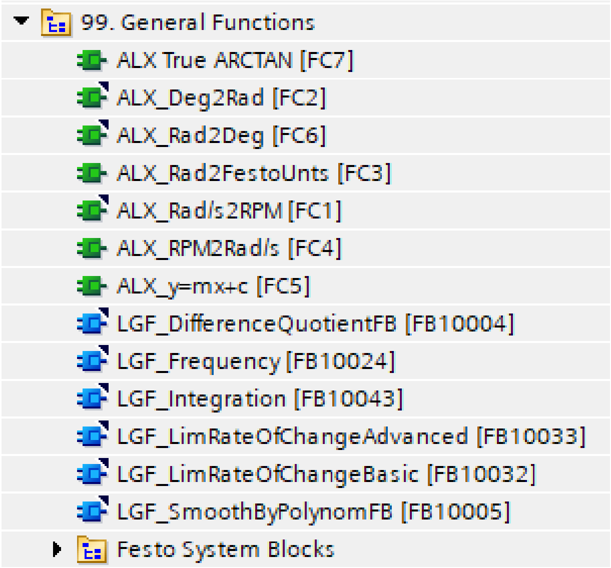
9.2.15 General Functions

The general functions folder in the PLC contains code reused multiple times in different portions of the control system. These functions are listed in table 9.3.

Table 9.3: Generalised Functions

Name	**Description**
ALX True ARCTAN [FC7]	Calculates the arc tan of a value, with the correct sign
ALX_Deg2Rad [FC2]	Converts degrees to radians
ALX_Rad2Deg [FC6]	Converts radians to degrees
ALX_Rad2FestoUnts [FC3]	Converts radians to Festo's measurement of angle (65536 ticks per revolution)
ALX_Rad/s2PRM [FC1]	Converts $^{radians}/_{s}$ to $^{r}/_{min}$
ALX_RPM2Rad/s [FC4]	Converts $^{r}/_{min}$ to $^{radians}/_{s}$
LGF_DifferenceQuotientFB [FB10004]	Preforms differentiation using known time delta
LGF_Frequency [FB10024]	Flips a boolean at a specified frequency
LGF_Integration [FB10043]	Uses numerical analysis to integrate variable sampled at a fixed rate
LGF_LimitRateOfChangeAdvanced [FB10033]	Limits the rate of change of variable (rate selectable for rise vs fall, + vs -)
LGF_LimitRateOfChangeBasic [FB10032]	Limits the rate of change of variable (rate fixed)
LGF_SmoothByPolynomFB [FB10005]	Smooths variable using higher level polynomial

Code contained in table 9.3 that has the prefix "ALX" was written by the author of this thesis, while code with the prefix "LGF" was written by Siemens and can be downloaded from the Siemens Industrial Mall. The contents of "99. General Functions" is illustrated in figure 9.18. The code for these blocks can be found in appendix T.



99. General Functions
- ALX True ARCTAN [FC7]
- ALX_Deg2Rad [FC2]
- ALX_Rad2Deg [FC6]
- ALX_Rad2FestoUnts [FC3]
- ALX_Rad/s2RPM [FC1]
- ALX_RPM2Rad/s [FC4]
- ALX_y=mx+c [FC5]
- LGF_DifferenceQuotientFB [FB10004]
- LGF_Frequency [FB10024]
- LGF_Integration [FB10043]
- LGF_LimRateOfChangeAdvanced [FB10033]
- LGF_LimRateOfChangeBasic [FB10032]
- LGF_SmoothByPolynomFB [FB10005]
- Festo System Blocks

Figure 9.18: General Functions Code

9.3 Software PLC code

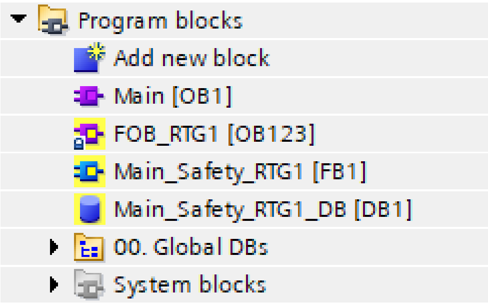
The software PLC currently does very little for the AGV's functionality. The functionality that will be implemented is listed in the section to follow and is not within the scope of this thesis.

There are two mutually exclusive modes that the software PLC will run in.

The first mode will interpret the data coming from the SICK NAV350 (x-position, y-position and yaw position telegram updated every 100 ms). These values will be used to run a closed-loop control system that generates an x-centroidal velocity component, y-centroidal velocity component and yaw rate to follow a predetermined path in memory (so-called "virtual" line following). The closed-loop part results from the fact that the low-level controller on the PLC will return an actual X centroidal velocity, actual Y centroidal velocity and actual yaw rate for error calculation.

The second mode passes the SICK NAV350 raw point cloud data to a Debian VM running ROS and SLAM for real-time navigation and object avoidance. This VM will pass the resultant x-centroidal velocity, y-centroidal velocity, and yaw rate data desired of the AGV to the high-level controller. The high-level controller passes the data un-edited to the low-level controller. The high-level controller will also echo the actual values from the low-level control system back to the ROS system to complete the control loop.

The code as the system stands (at the time of this thesis's creation) is listed in figure 9.19 and is detailed in appendix U.



- Program blocks
  - Add new block
  - Main [OB1]
  - FOB_RTG1 [OB123]
  - Main_Safety_RTG1 [FB1]
  - Main_Safety_RTG1_DB [DB1]
  - 00. Global DBs
  - System blocks

Figure 9.19: Screen Capture of Code Folders for the Software PLC

9.4 WinCC SCADA

The SCADA system is hosted on the Siemens IPC alongside the software PLC. The purpose of the SCADA system is primarily to host an HMI that is displayed on the Mecer A5026 LCD screen. The secondary function of the SCADA is to pass commands from the low-level controller to the Windows 7 VM. Such as the commands to run the scripts that gracefully shut down the various components on the AGV.

The Mecer A5026 screen is not a touchscreen and, as such, uses a generic "mini wireless keyboard & mouse" as its input device (see figure 9.20). However, this screen can be replaced in the future with any generic VESA mountable touchscreen.

Figure 9.20: Generic Mini Wireless Keyboard & Mouse[1]

The following screens were included in the HMI:

1. Alarms [Alarms]
2. Battery Unit [BatteryUnit]

[1] Image adapted from the public domain

3. Calibrate Analogs [CalibrateAnalogs]
4. Choose Orientation Mode [ChoseOrintMode]
5. Home [Home]
6. Jog Choice [JogChoice]
7. Jog System AGV [JogSysemAGV]
8. Jog System Units [JogSystemUnits]
9. Main [Main]
10. Router Webpage [RouterWebpage]
11. System [System]
12. Test Runner [TestRunner]
13. Wheel Orientations [WheelOrientation]

Note: The tags used are listed in square brackets next to the screen name. A screenshot of the folder containing these screens is shown in figure 9.21.

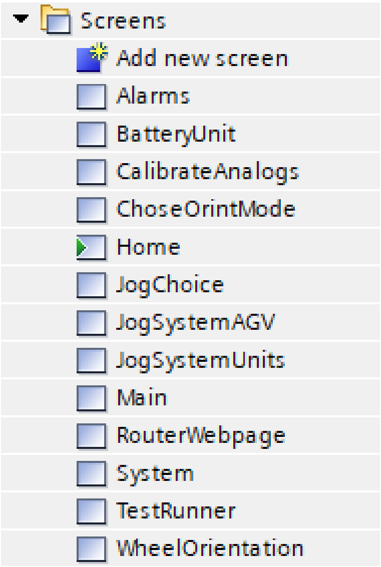


Figure 9.21: Screen Capture of Screens Folder for WinCC

The navigation of the screens is shown in the flow diagram contained in figure 9.22.

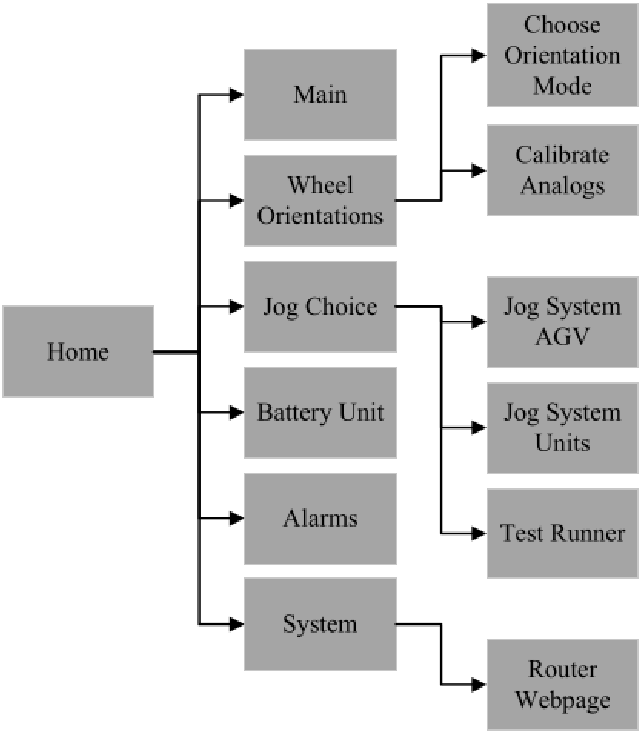


Figure 9.22: HMI Screen Navigation

9.4.1 HMI Screens

The function of each screen is listed in the sections that follow:

9.4.1.a Home

Home is the root screen for the system. I.e. the screen that the system boots on too. This screen is illustrated in figure 9.23. Since this is a landing screen, the information available on it is spartan. This screen is used to navigate to an appropriate screen using the buttons along the bottom taskbar.

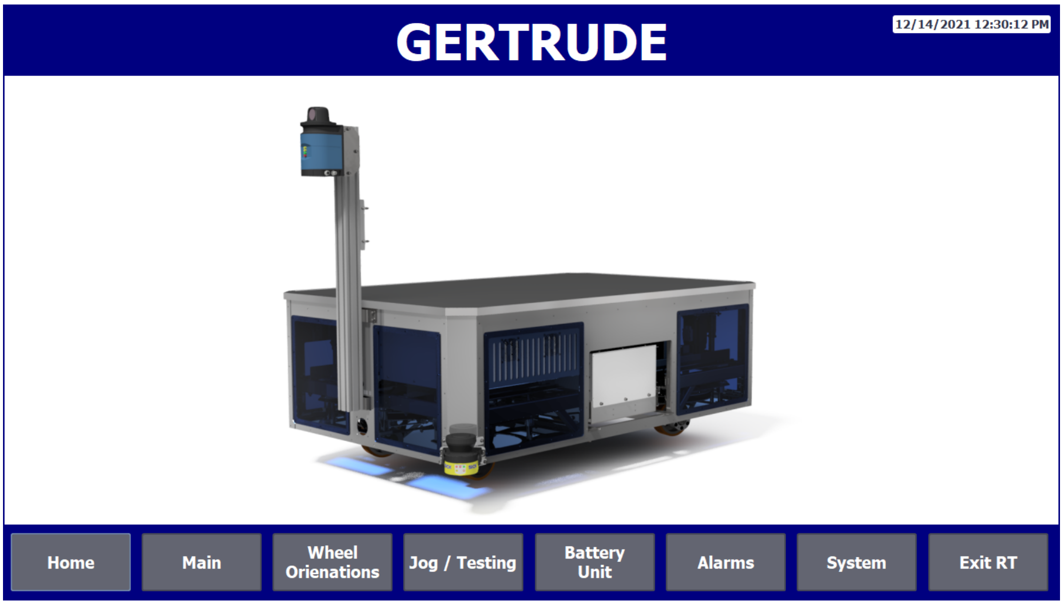
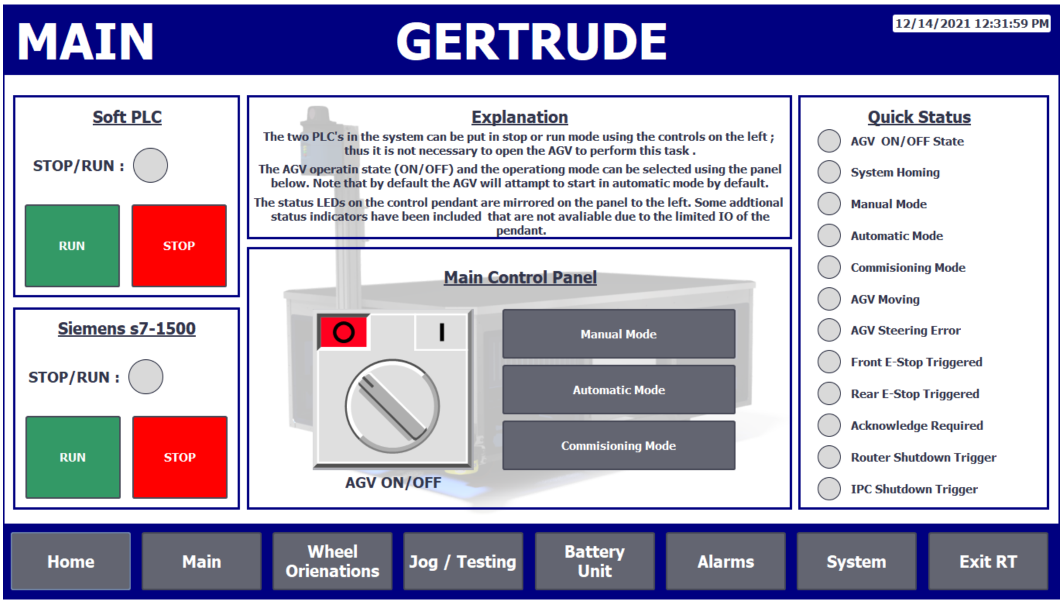


Figure 9.23: Home Screen

9.4.1.b Main

The main screen is the first screen available via the buttons arranged along the bottom taskbar. This screen is shown in figure 9.24. As illustrated in figure 9.24, it is possible to put both the physical PLC and Software PLC into either RUN or STOP mode using this screen. The mode of operation can also be set using this screen; the selectable modes are manual mode, automatic mode and commissioning mode. The remaining modes mentioned in the PLC code section (section 9.2) are activated when the user enters the appropriate screen and are not manually selectable. Finally, this screen includes "Quick Status" indicators that assist with the operation of the AGV. Many of these indicators are mirrored by the LEDs found on the pendant. As illustrated in figure 9.24, the AGV can be put into ON or OFF mode using the rotary switch. The position of this rotary switch will automatically update itself to the appropriate orientation should the push-button on the pendant be used to change the ON/OFF status of the AGV.



The two PLC's in the system can be put in stop or run mode using the controls on the left ; thus it is not necessary to open the AGV to perform this task .

The AGV operatin state (ON/OFF) and the operationg mode can be selected using the panel below. Note that by default the AGV will attampt to start in automatic mode by default.

The status LEDs on the control pendant are mirrored on the panel to the left. Some addtional status indicators have been included that are not avaliable due to the limited IO of the pendant.

Figure 9.24: Main Screen

9.4.1.c Wheel Orientations

The wheel orientation screen is primarily for visualisation. Due to space limitations, the heading for the screen was concatenated to "wheels", as illustrated in figure 9.25.

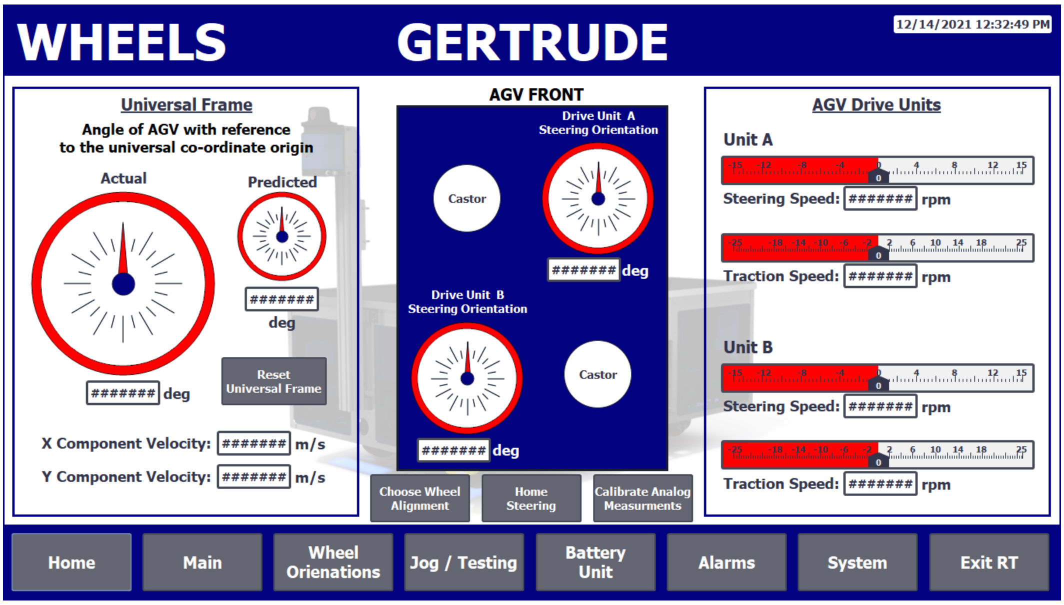

WHEELS
GERTRUDE
12/14/2021 12:32:49 PM
Universal Frame
Angle of AGV with reference to the universal co-ordinate origin
Actual
Predicted
####### deg
####### deg
Reset Universal Frame
X Component Velocity: ####### m/s
Y Component Velocity: ####### m/s
AGV FRONT
Drive Unit A Steering Orientation
Castor
####### deg
Drive Unit B Steering Orientation
Castor
####### deg
Choose Wheel Alignment
Home Steering
Calibrate Analog Measurments
AGV Drive Units
Unit A
Steering Speed: ####### rpm
Traction Speed: ####### rpm
Unit B
Steering Speed: ####### rpm
Traction Speed: ####### rpm
Home
Main
Wheel Orientations
Jog / Testing
Battery Unit
Alarms
System
Exit RT

Figure 9.25: Wheel Orientation Screen

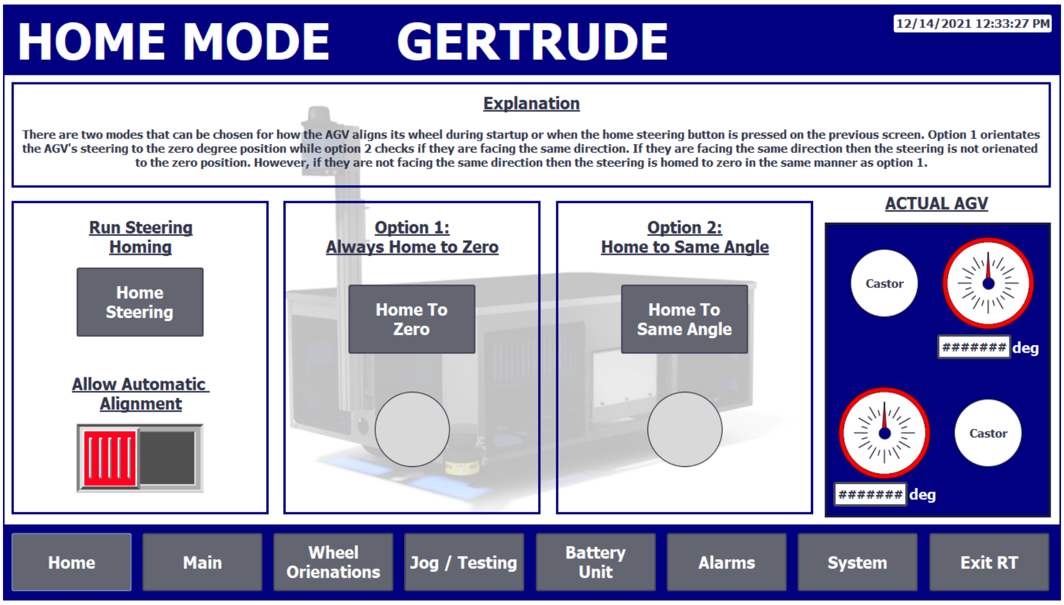
This screen shows the real-time angular orientation of the AGV with reference to the universal fame. Both a predicted (integrated from the set point or desired yaw rate) and actual angle (integrated from actual yaw rate) are shown. In addition, the actual x-component velocity and y-component velocity of the AGV are given. If the user needs to reset the universal coordinate system to the AGV's coordinate system, this can be done using the "Reset Universal Frame" button.

The centre of this screen contains a graphical representation of the AGV, viewed from above. This screen displays the real-time orientation of the drive unit's steering and a numerical degree value. The rightmost section of the screen lists the live speed of the traction and steering motors, both graphically and numerically, in RPM.

If the user wants to force the AGV's steering to home at any stage, this can be achieved by using the button "Home Steering". The button "Choose Wheel Alignment" takes the user to the "Choose Orientation Mode" screen, while the button " Calibrate Analog Measurement" takes the user to the "Calibrate Analogs" screen. Neither of these screens can be reached outside the "Wheel Orientation" screen.

9.4.1.d Choose Orientation Mode

The screen name is shortened to "HOME MODE" due to size limitations as illustrated in figure 9.26.



HOME MODE GERTRUDE

12/14/2021 12:33:27 PM

Explanation

There are two modes that can be chosen for how the AGV aligns its wheel during startup or when the home steering button is pressed on the previous screen. Option 1 orientates the AGV's steering to the zero degree position while option 2 checks if they are facing the same direction. If they are facing the same direction then the steering is not orienated to the zero position. However, if they are not facing the same direction then the steering is homed to zero in the same manner as option 1.

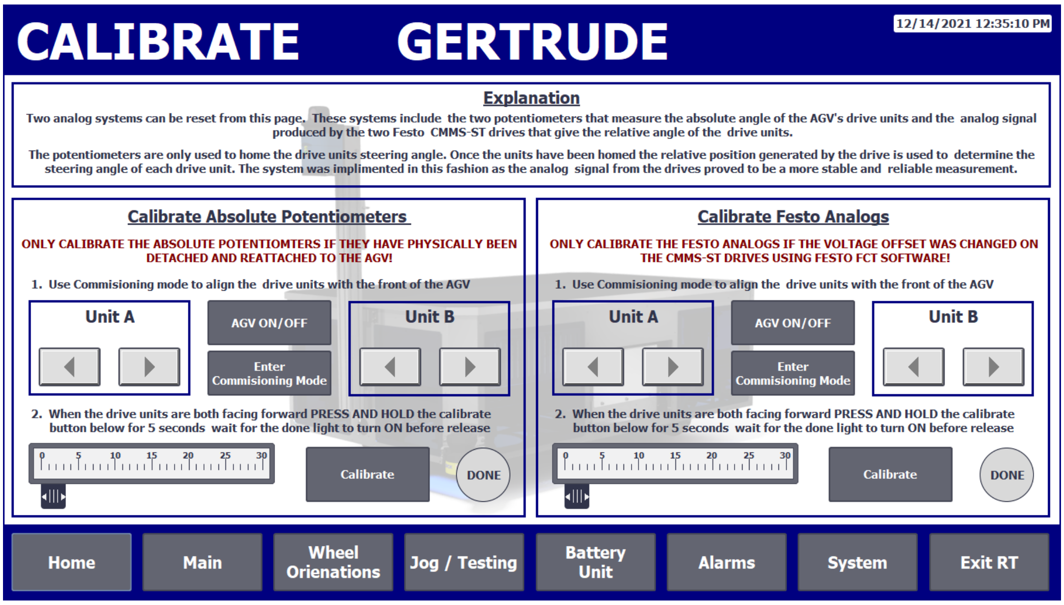
Figure 9.26: Choose Orientation Mode Screen

This screen is only accessible via the screen "Wheel Orientations". It is used to set the system's behaviour when the steering is homed. The home steering sub-routine is run when the AGV boots or can be manually invoked at any time using the button "Home Steering" on the screen Wheel Orientations".

Using this screen can determine the type of home the steering will do. The default is to home and orientates both wheels to zero degrees (rolling axis perpendicular to the x-axis of AGV, see section 5.1). The alternative mode still homes the wheels to the zero degree point of the AGV but orientates them with reference to each other (i.e. the system will move the wheels as little as possible from their current orientation until they share the same orientation angle). The second mode was implemented to reduce wheel scuffing. The homing can also be completely disabled from this window. Provided the AGV is not moved between power cycles, the relative angle stored in retentive memory should still be valid. However, keeping the homing disabled for a long time is not recommended as the cumulative error will eventually stack.

9.4.1.e Calibrate Analogs

The name of this screen was shortened to "Calibrate", as illustrated in figure 9.27. It is only accessible via the screen "Wheel Orientations".


CALIBRATE GERTRUDE

12/14/2021 12:35:10 PM

Explanation

Two analog systems can be reset from this page. These systems include the two potentiometers that measure the absolute angle of the AGV's drive units and the analog signal produced by the two Festo CMMS-ST drives that give the relative angle of the drive units.

The potentiometers are only used to home the drive units steering angle. Once the units have been homed the relative position generated by the drive is used to determine the steering angle of each drive unit. The system was implimented in this fashion as the analog signal from the drives proved to be a more stable and reliable measurement.

Calibrate Absolute Potentiometers

ONLY CALIBRATE THE ABSOLUTE POTENTIOMTERS IF THEY HAVE PHYSICALLY BEEN DETACHED AND REATTACHED TO THE AGV!

1. Use Commisioning mode to align the drive units with the front of the AGV

2. When the drive units are both facing forward PRESS AND HOLD the calibrate button below for 5 seconds wait for the done light to turn ON before release

Calibrate Festo Analogs

ONLY CALIBRATE THE FESTO ANALOGS IF THE VOLTAGE OFFSET WAS CHANGED ON THE CMMS-ST DRIVES USING FESTO FCT SOFTWARE!

1. Use Commisioning mode to align the drive units with the front of the AGV

2. When the drive units are both facing forward PRESS AND HOLD the calibrate button below for 5 seconds wait for the done light to turn ON before release

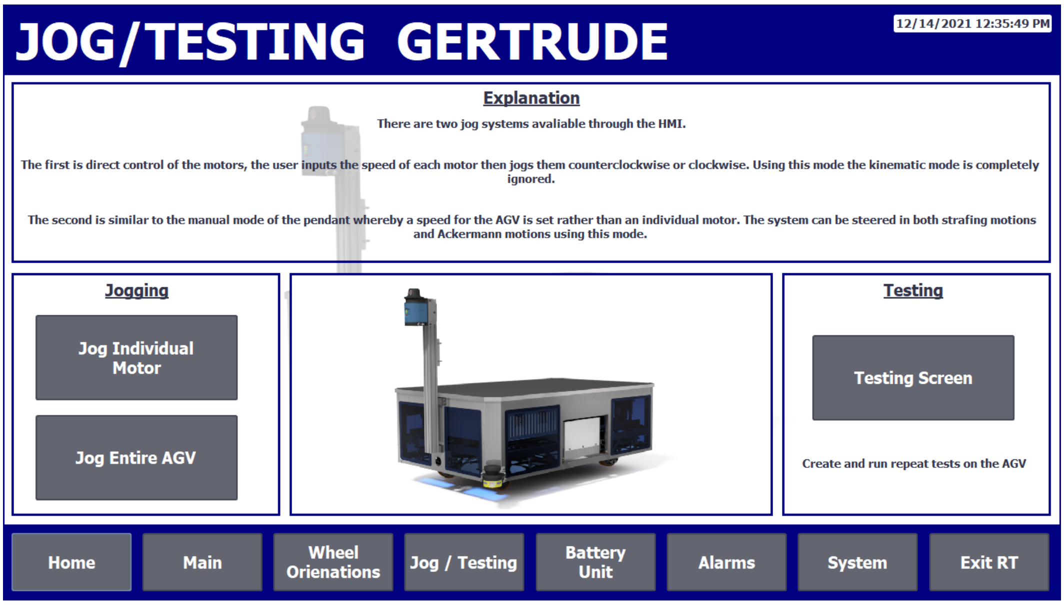
Figure 9.27: Calibrate Analogs Screen

This screen allows the user to perform two tasks. The first is calibrating how the PLC interprets the analogue voltage returned from the Festo drives (this voltage represents a relative angle). This calibration can set any arbitrary voltage returned by the Festo drive as the zero degree point. The second calibration option allows the user to set any arbitrary voltage returned by the absolute steering angle potentiometers as the zero degree point. This second calibration must be done each time the absolute potentiometers are removed and reattached, as it is improbable that they will be placed back on the AGV in the same position as before.

Both calibration tasks follow the same methodology. The steering systems are jogged to the desired position (and checked with a protractor) using the buttons included on the screen. The user then holds down the appropriate calibrate button until the done indicator changes to yellow. This procedure can take up to 5 seconds.

9.4.1.f Jog Choice

The "Jog Choice" screen (see figure 9.28), labelled "JOG/TESTING" in the heading of the screen, is used to direct the operator to three other screens.



JOG/TESTING GERTRUDE

12/14/2021 12:35:49 PM

Explanation

There are two jog systems avaliable through the HMI.

The first is direct control of the motors, the user inputs the speed of each motor then jogs them counterclockwise or clockwise. Using this mode the kinematic mode is completely ignored.

The second is similar to the manual mode of the pendant whereby a speed for the AGV is set rather than an individual motor. The system can be steered in both strafing motions and Ackermann motions using this mode.

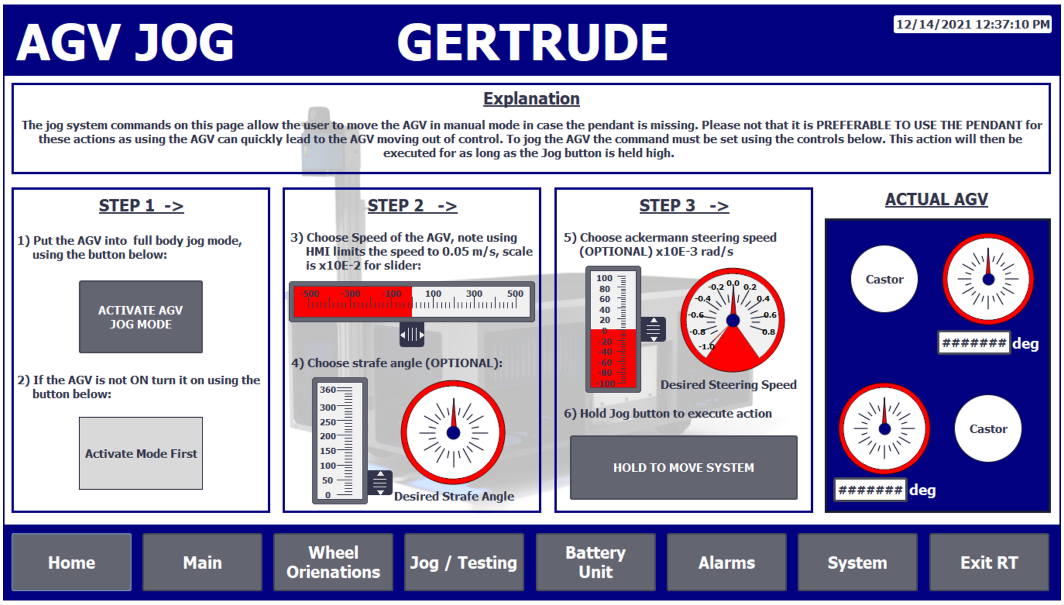
Create and run repeat tests on the AGV

Figure 9.28: Jog Choice Screen

The screens that can be accessed from here are the "Jog System AGV" via the button labelled "Jog Entire AGV", the "Jog System Units" via the button labelled "Jog Individual Motor", and the "Test Runner" screen via the button labelled "Testing Screen".

9.4.1.g Jog System AGV

The "Jog System AGV" (figure 9.29), labelled "AGV JOG", can be used as an alternative to manual mode using the pendant. In both cases, the AGV is fed a speed value, strafe value and Ackerman steering value and moves according to these desired inputs. This system is much more awkward to use when compared to the manual mode and pendant since, for safety reasons, the button "HOLD TO MOVE SYSTEM" must be pressed to move the system. As the HMI is not multi-touch, if the steering or strafe is to be changed, the user must first release the button "HOLD TO MOVE SYSTEM" (causing the AGV to stop), change the values and then press the button again. This interlocking means that the AGV cannot be steered while it is moving; but instead has to stop, change its steering behaviour, and move again.


AGV JOG GERTRUDE

12/14/2021 12:37:10 PM

Explanation

The jog system commands on this page allow the user to move the AGV in manual mode in case the pendant is missing. Please not that it is PREFERABLE TO USE THE PENDANT for these actions as using the AGV can quickly lead to the AGV moving out of control. To jog the AGV the command must be set using the controls below. This action will then be executed for as long as the Jog button is held high.

STEP 1 ->

1) Put the AGV into full body jog mode, using the button below:

2) If the AGV is not ON turn it on using the button below:

STEP 2 ->

3) Choose Speed of the AGV, note using HMI limits the speed to 0.05 m/s, scale is x10E-2 for slider:

4) Choose strafe angle (OPTIONAL):

STEP 3 ->

5) Choose ackermann steering speed (OPTIONAL) x10E-3 rad/s

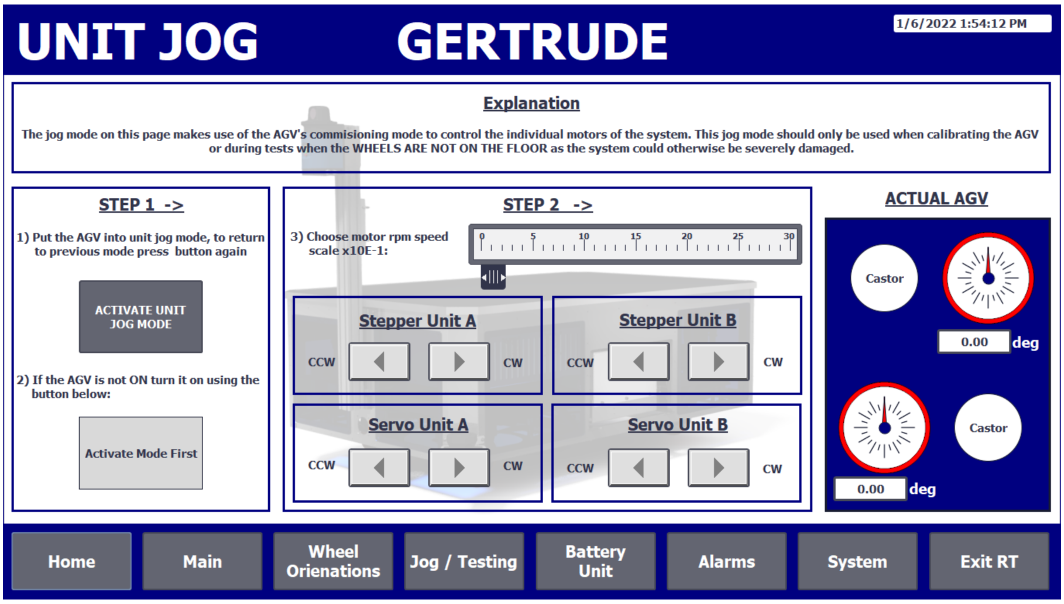
6) Hold Jog button to execute action

Figure 9.29: Jog System AGV Screen

This mode of operation should only be used if the pendant has been removed and is not available.

9.4.1.h Jog Unit AGV

The "Jog Unit AGV" (figure 9.30), labelled "Unit Jog", can be used as an alternative to the commissioning mode used with the pendant. This mode allows for low-level control of the individual steering and traction motors. To operate this mode, the AGV is first placed into the model using the button "ACTIVATE UNIT JOG MODE" once this is done, the AGV can be placed in run mode (the run mode button only becomes visible when the mode is active before this the button is locked out and displays the text: "Activate Mode First"). Once the AGV has been activated, the top slider can be used to select the desired motor speed; the appropriate motor is then jogged by holding down an appropriate direction button.



UNIT JOG GERTRUDE

1/6/2022 1:54:12 PM

Explanation

The jog mode on this page makes use of the AGV's commisioning mode to control the individual motors of the system. This jog mode should only be used when calibrating the AGV or during tests when the WHEELS ARE NOT ON THE FLOOR as the system could otherwise be severely damaged.

STEP 1 ->

1) Put the AGV into unit jog mode, to return to previous mode press button again

ACTIVATE UNIT JOG MODE

2) If the AGV is not ON turn it on using the button below:

Activate Mode First

STEP 2 ->

3) Choose motor rpm speed scale x10E-1:

Stepper Unit A: CCW ◀ ▶ CW

Stepper Unit B: CCW ◀ ▶ CW

Servo Unit A: CCW ◀ ▶ CW

Servo Unit B: CCW ◀ ▶ CW

ACTUAL AGV

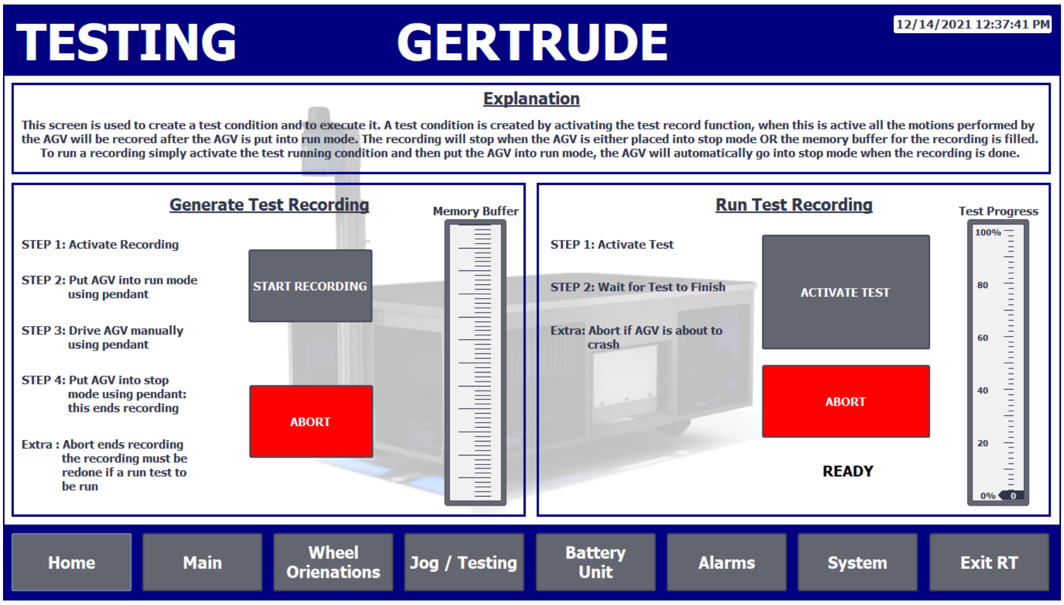
0.00 deg

0.00 deg

Figure 9.30: Jog Unit AGV Screen

9.4.1.i Test Runner

This screen is used to generate tests and execute them at a later stage. Once again, due to size limitations, the heading on the screen was changed to "TESTING" (as illustrated in figure 9.31).



TESTING GERTRUDE

12/14/2021 12:37:41 PM

Explanation

This screen is used to create a test condition and to execute it. A test condition is created by activating the test record function, when this is active all the motions performed by the AGV will be recored after the AGV is put into run mode. The recording will stop when the AGV is either placed into stop mode OR the memory buffer for the recording is filled. To run a recording simply activate the test running condition and then put the AGV into run mode, the AGV will automatically go into stop mode when the recording is done.

Generate Test Recording

STEP 1: Activate Recording

STEP 2: Put AGV into run mode using pendant

STEP 3: Drive AGV manually using pendant

STEP 4: Put AGV into stop mode using pendant: this ends recording

Extra : Abort ends recording the recording must be redone if a run test to be run

START RECORDING

ABORT

Memory Buffer

Run Test Recording

STEP 1: Activate Test

STEP 2: Wait for Test to Finish

Extra: Abort if AGV is about to crash

ACTIVATE TEST

ABORT

READY

Test Progress

Figure 9.31: Test Runner Screen

The left portion of the screen is used to generate a test. Test generation is done by activating the recording using the "START RECORDING" button. This action would turn the AGV off if it were previously on and places the AGV into manual mode. When the user turns the AGV back on using the push button on the pendant, the system will begin recording the relevant input values and will store them in a memory buffer. The recording will end when the "ABORT" button is pressed, the AGV is placed in OFF mode, or the memory buffer becomes saturated.

The right portion of the screen is used to run the previously recorded test. Running a test is done by pressing the "ACTIVATE TEST" button. The AGV will then drive, in manual mode, using the previously recorded data. It is recommended to first move the AGV to roughly the same starting point between tests to prevent obstacles or different driving surfaces from interfering with the test.

9.4.1.j Battery Unit

This system was not implemented at the time of writing this report; as such, a greyed out holding screen is shown in figure 9.32.

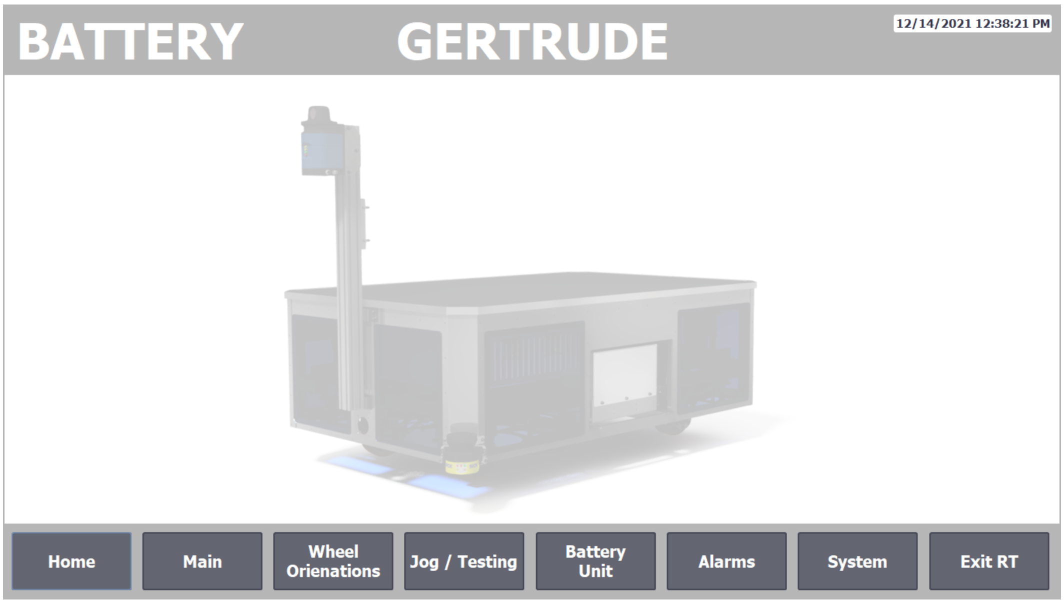
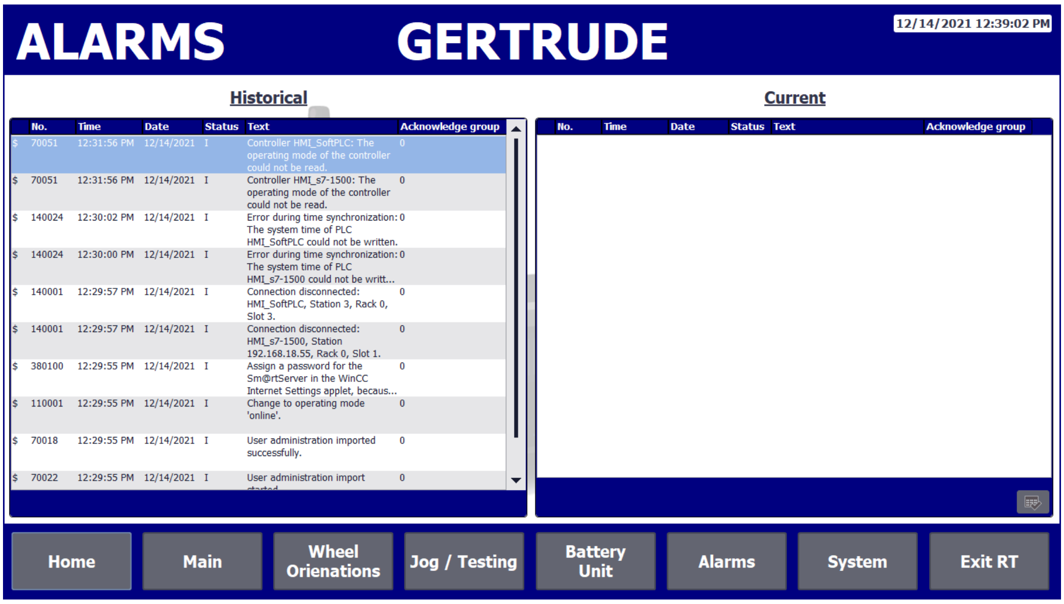


Figure 9.32: Battery Unit Screen

9.4.1.k Alarms

The alarms screen (see figure 9.33) shows the currently active alarms on the right-hand side and a historical view of the alarms that have occurred on the left-hand side. A complete list of all the alarms implemented on the AGV can be found in appendix V, section V.5.



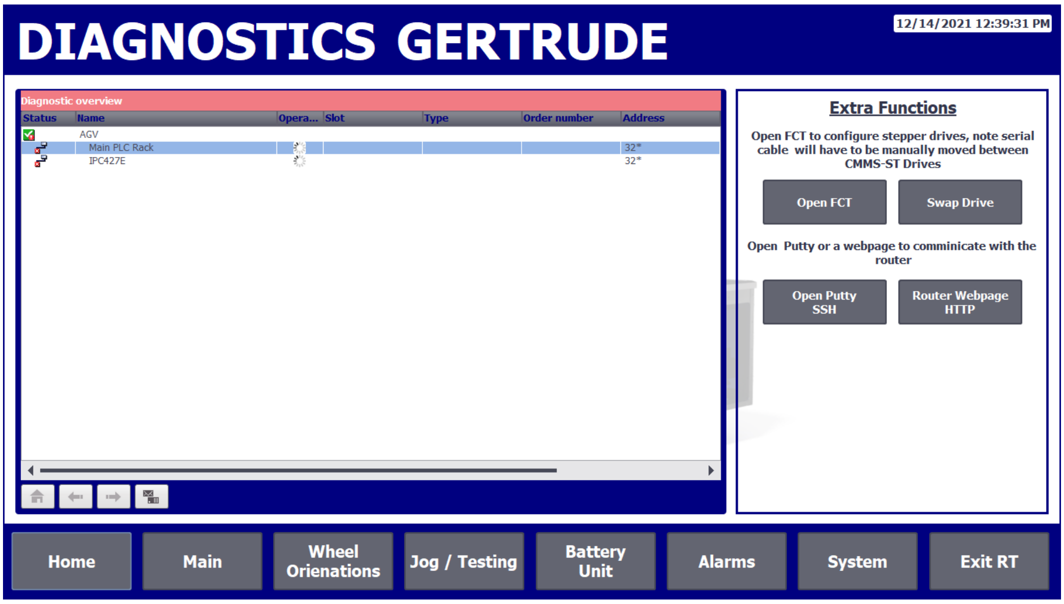
	No.	Time	Date	Status	Text	Acknowledge group
$	70051	12:31:56 PM	12/14/2021	I	Controller HMI_SoftPLC: The operating mode of the controller could not be read.	0
$	70051	12:31:56 PM	12/14/2021	I	Controller HMI_s7-1500: The operating mode of the controller could not be read.	0
$	140024	12:30:02 PM	12/14/2021	I	Error during time synchronization: The system time of PLC HMI_SoftPLC could not be written.	0
$	140024	12:30:00 PM	12/14/2021	I	Error during time synchronization: The system time of PLC HMI_s7-1500 could not be writt...	0
$	140001	12:29:57 PM	12/14/2021	I	Connection disconnected: HMI_SoftPLC, Station 3, Rack 0, Slot 3.	0
$	140001	12:29:57 PM	12/14/2021	I	Connection disconnected: HMI_s7-1500, Station 192.168.18.55, Rack 0, Slot 1.	0
$	380100	12:29:55 PM	12/14/2021	I	Assign a password for the Sm@rtServer in the WinCC Internet Settings applet, becaus...	0
$	110001	12:29:55 PM	12/14/2021	I	Change to operating mode 'online'.	0
$	70018	12:29:55 PM	12/14/2021	I	User administration imported successfully.	0
$	70022	12:29:55 PM	12/14/2021	I	User administration import	0

No.	Time	Date	Status	Text	Acknowledge group

Figure 9.33: Alarms Screen

9.4.1.1 System

The system screen displays the built-in diagnostics from the software PLC and s7-1512F PLC. This screen can also access either PLC's diagnostics buffer (a log system for important events in the PLC's runtime). In addition to hosting the diagnostics windows, there are also four buttons located on the rightmost side of the screen. The "Open FCT" button opens Festo's FCT software as a pop-up window to allow the user to change the configuration of the drives. Due to poor implementation by Festo, the configuration of the CMMS-ST drives can only be done through serial COM1. Thus, a mechanical switcher was needed to physically change over the cable even though the IPC has three serial ports. This switching was done via the PLC using a digital output and a relay bank black box.


DIAGNOSTICS GERTRUDE

12/14/2021 12:39:31 PM

Diagnostic overview

Status	Name	Opera...	Slot	Type	Order number	Address
✓	AGV					
☒	Main PLC Rack					32*
☒	IPC427E					32*

Extra Functions

Open FCT to configure stepper drives, note serial cable will have to be manually moved between CMMS-ST Drives

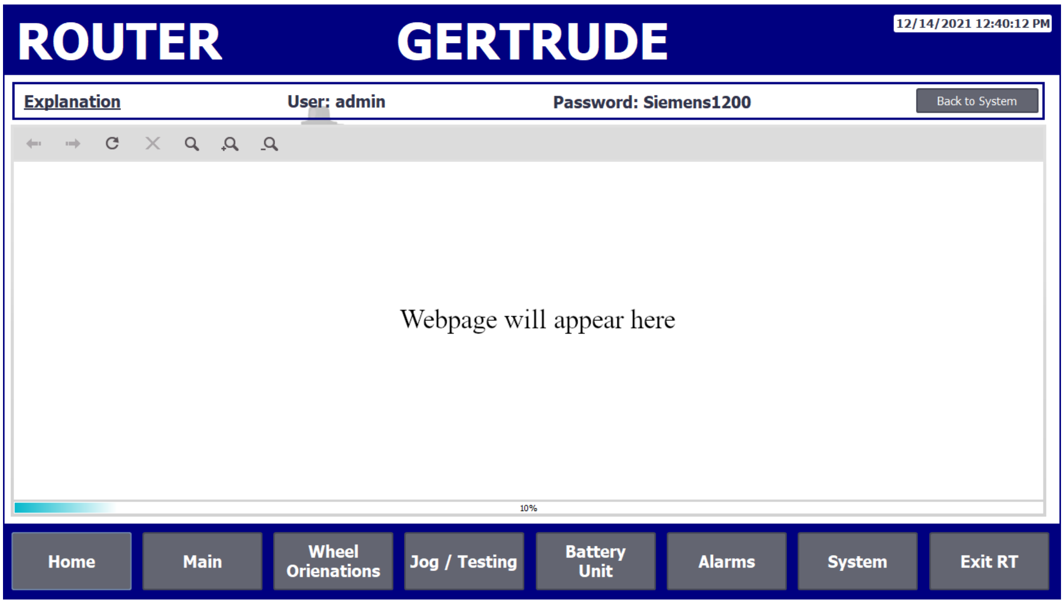
Open Putty or a webpage to comminicate with the router

Figure 9.34: System Screen

The "Open PuTTY SSH" button opens an SSH window to the router to adjust its configuration. The router runs PFsense, which is based on FreeBSD; thus, most Linux commands will not work, and the FreeBSD equivalent should be used. If using the terminal is beyond the user's capabilities, an embedded webpage for the router can be opened using the "Router Webpage HTTP" button.

9.4.1.m Router Webpage

This screen contains an embedded HTTP management webpage for the PFsense router, as illustrated in figure 9.35. Since this screen was printed from the TIA portal software, the live webpage is not visible.



ROUTER GERTRUDE

12/14/2021 12:40:12 PM

User: admin

Password: Siemens1200

Webpage will appear here

10%

Figure 9.35: Router Webpage Screen

To idiot-proof the system, the login credentials for the router are printed on the border of the webpage. Management is only allowed via the LAN; WAN access is blocked to ensure the system is still relatively safe. Thus even if a malicious actor were to acquire these credentials, they would not be able to access the system unless they attached their PC/PG to the LAN side, which requires physical access to the machine.

9.4.2 VBScripts

There are two VBScripts written on the WinCC system. They are the "ShutdownIPC" script and the "ShutdownRouter" script. These scripts act as brokers to activate scripts or batch files on the Windows 7 VM.

The "ShutdownIPC" script is used to call the batch file "IPCPowerOff.bat" located in the Windows 7 directory "C:\AlexBatchFiles". The called batch file is detailed in the section 9.5, while the raw code for the SCADA VBScript is listed in appendix V.6.

The "ShutdownIPC" VBScript script is used to call the VBScript on the Windows 7 VM called "RouterPowerOff.vbs" located in the directory "C:\AlexBatchFiles". Details on the Windows 7 VBScript are detailed in the section 9.5, while the SCADA script can be found in appendix V.6.

9.5 Windows 7 VM

The Windows 7 VM was included by Siemens on the IPC by default. Although it would have been ideal to have had only a Debian Linux machine running on the IPC (the author is partial to Kubuntu), the Windows 7 VM is necessary to manage the Software PLC and SCADA system, neither of which have Linux apps. Due to certification of safety issues, it is inadvisable to run these tools under something like WINE (compatibility layer for Linux).

The windows VM also hosts the Festo FCT tool for configuration and management of the Festo CMMS-ST drives along with "WatchPower", which is a proprietary tool that interacts with the RCT inverter and displays data such as State of Charge (SoC), current draw, power draw, and last charge time. It is recommended that WatchPower be depreciated later as the software is closed-source, and as such, there is no easy way to grab that data for use in the PLCs. There is an open-source tool called NUT (Network UPS Tool) which can be used on Debian Linux in its place, which provides the appropriate means to export the data from the inverter (section 9.9.1).

9.5.1 IPC Shutdown Code

The IPC shutdown code called by the WinCC VBScript is straightforward. It only contains the code "shutdown /s /t 20". When run on a windows system, this will cause the system to shut down after a time delay of 20 seconds. This code was included in appendix W. The batch file is named "IPCPowerOff.bat" and is stored in the Windows 7 directory "C:\AlexBatchFiles".

9.5.2 PFsense Router Shutdown

The PFsense router shutdown is slightly more complex than the IPC shutdown code as it has to use an SSH window to shut down the router remotely. Since Windows 7 does not natively support SSH (this feature was only introduced in Windows 10), a third-party application had to be used. The chosen application was PuTTY since PuTTY is lightweight, FOSS and can store pre-configured connection templates.

The pre-configured PuTTY template created by the author was called "GoldenRouter" and pointed the connection to the SSH channel using port TCP 22 and IP address 192.168.18.1 (LAN interface of the router). To open this template, a VBScript was called by the SCADA system named "RouterPowerOff.vbs", which was located in the directory "C:\AlexBatchFiles". This script, in turn, opened a batch file in the same directory, called "OpenRouterTerm.bat", which contained the code to open and run the PuTTY template. After the batch file opened the SSH window via the template, the "RouterPowerOff.vbs" VBScript then pushed the appropriate FreeBSD commands to the router to shut down the system using the SSH window.

The code for the batch files and VBScripts used in this system can be found in appendix W.

9.6 CMMS-ST Stepper Drive Configuration

The CMMS-ST drives were configured using Festo's FCT tool. These drives can only be commissioned using RS232 communication via a serial port; a shortfall of the FCT system is that it can only connect to a drive via COM1. Thus, communication cannot be established if something else is using that port. There is no sensible reason for this besides the extremely shoddy implementation by Festo. A serial switched box was used to alleviate this issue, which is actuated by an output from the PLC.

FCT was used to input the physical characteristics of the motor (gearbox ratio, motor size, etc.). The two significant deviations from a stock configuration of this type of system were to use the analogue output (0V - 10V) of the drives to register the relative angle of the steering with the zero degree point set to 5V (explained in section 9.2.13), and setting the homing method of the drive to its current position (as "homing" is done by the PLC and not the drive).

The complete configuration of the Festo stepper drives can be found in appendix X.

9.7 V90 Servo Drive Configuration

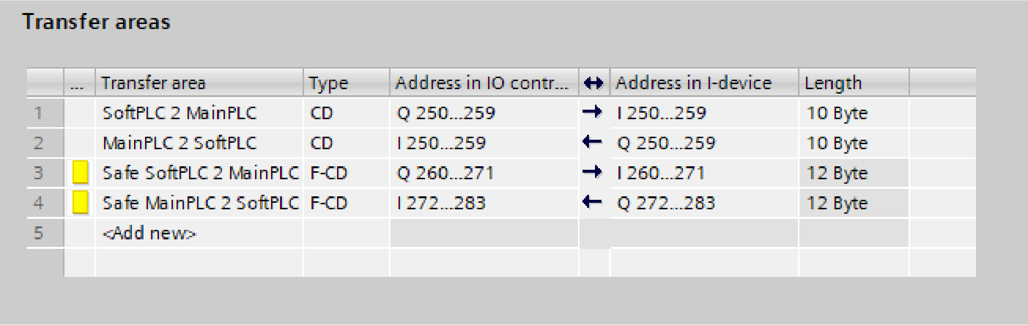
There was minimal configuration done on the V90 drives besides selecting the correct motor and limiting the maximum allowable speed to 3000 $^{r}/_{min}$ (translates to 1.3 $^{m}/_{s}$ after gear train). Configuration was done using the Siemens V-ASSISTANT tool. The results of the configuration can be found in appendix Y.

The only difference in the configuration of the two V90 drives (Unit A and Unit B) was the IP configuration. Since the IP configuration of the drives is not set in V-ASSISTANT but rather in TIA Portal, the configuration files from Drive A and Drive B were identical in V-ASSISTANT; thus, it is only shown once in appendix Y.

9.8 Interface between IPC and PLC

In order to pass safety data via ProfiSafe between the two controllers (PLC and Software PLC), it was necessary to set up the PLC as an "iDevice". Essentially an iDevice behaves as both a master and slave device simultaneously. Thus, the PLC will act as a slave device to the Software PLC and as a master device to the drives and other distributed I/O.

To pass safety signals and other data seamlessly between the two PLCs, the following peripheral I/O was dedicated, as illustrated in figure 9.36.



Transfer areas

	...	Transfer area	Type	Address in IO contr...	↔	Address in I-device	Length
1		SoftPLC 2 MainPLC	CD	Q 250...259	→	I 250...259	10 Byte
2		MainPLC 2 SoftPLC	CD	I 250...259	←	Q 250...259	10 Byte
3	🟨	Safe SoftPLC 2 MainPLC	F-CD	Q 260...271	→	I 260...271	12 Byte
4	🟨	Safe MainPLC 2 SoftPLC	F-CD	I 272...283	←	Q 272...283	12 Byte
5		<Add new>					

Figure 9.36: PLC to PLC iDevice Transfer Areas

9.9 Incomplete Code and Sections

Listed in the following sections are segments of code and objects on the AGV that still need to be completed to produce a genuinely automatic vehicle. This code and programming were outside the scope of the research, and as such, its incomplete state does not affect this thesis but will need to be completed to create a minimal viable product (MVP) for commercialisation.

9.9.1 Battery Eject

The battery eject code will need to handshake via the WAN to the battery loading/unloading device. When the power management system determines that the battery capacity is below a predefined threshold, the AGV will attempt to return to the docking station.

The easiest way to monitor the battery system capacity would be to use the built-in BMS on the inverter. Information is currently being sent to the Window's VM from the inverter BMS. NUT (Network UPS Tool) can be used to interpret this data for use with the AGV's control system. NUT will, however, have to be implemented on Debian Linux. Usage on Debian is beneficial as it can be hooked into the ROS system as a ROS node.

Once the AGV has moved into the operational range of the docking station, ROS can hand off the rest of the procedure to the low-level PLC controller. This controller will handshake with the docking station and actuate the eject motor as appropriate. Section 5.9.1 describes the docking procedure and mechanism.

9.9.2 AGV Side IoT 2040

There is an IoT2040 located in the control box of the AGV. This IoT can be used as part of the battery management system on the AGV by communicating wirelessly to the IoT2040 inside the main battery unit. This IoT can also act as an IoT gateway to Mindsphere, a Siemens cloud system targeted at the manufacturing industry.

9.9.3 Battery Management System

In addition to using the built-in BMS on the inverter, it would be advantageous to have an IoT 2040 inside the main battery unit acting as an advanced BMS. This additional BMS will greatly increase the overall lifespan of the batteries since, at the moment, the BMS system on the inverter only makes use of the battery voltage to estimate the SoC (State-of-Charge). This is explained in section **??**. The inverter BMS is unaware of the actual capacity of the battery bank, the true SoC and the temperature of the batteries.

A far more accurate SoC value can be deduced if the IoT2040 is used to perform Columb counting and battery temperature measurement. Advanced BMS can significantly improve the lifespan of the batteries by preventing over-discharge and overcharging the battery.

9.9.4 SICK NAV 350

The SICK NAV 350 was not implemented in the AGV as it was faulty. This NAV 350 was received second hand from another project; however, it was never used in that project as far as the author is aware. When the author opened the device, none of the daughter boards was attached to the mainboard via the appropriate ribbon cables. Whether this was delivered faulty from the factory or was the result of the previous user is unknown; unfortunately, the device was outside of warranty and could not be replaced.

The SICK NAV 350, when it can be afforded to replace, will be used in two modes. The first mode is virtual line following; the NAV 350 will stream its current coordinates to the Software PLC via telegrams using retro-reflector beacons in the factory; this is called automatic mode. ROS (on the Debian-based VM) will be used for true autonomous mode. In this mode, the raw point cloud will be sent to the ROS system to create a SLAM map for the environment.

9.9.5 ROS integration

Adding ROS to this system was beyond the scope of the thesis. It was only necessary to ensure that ROS could be implemented. Preparing for ROS integration was achieved using an IPC and ensuring that the PLCs could communicate with the ROS Master as a ROS node using UDP.

9.10 Time Synchronisation Addendum

As eluded to in the previous sections, the time synchronisation method was changed from using a DB (data block) to pass the current time for synchronisation to using a built-in method only available on Siemens s7-1500 and s7-1200 PLC.

Initially, the time was read from the "time masters" internal clock using a library code block called "Read System Time" this time was then passed every minute to the "time slaves" who read the time from the DB and then overwrote their internal clocks with this time using a library code block called "Write System Time". This method was the documented synchronisation process for S7-300 and s7-400 Siemens PLC considered the previous generation or "legacy PLCs".

The author only became aware of the new method after the code skeleton was written, hence why the previous system has not completely been removed but deactivated. The new synchronization method is illustrated in appendix Z, using the connections tab in WinCC SCADA.

9.11 Chapter Conclusion

This chapter fully described all of the code's functionality in the AGV, including the code on both the hardware PLC, software PLC, Windows 7 VM and WinCC SCADA system. Also described in this section is code outside this project's scope but would need to be implemented to create a minimally viable product.

10 Testing

This section contains the tests used to validate the two-wheel functionality of the two-wheel swerve-drive system. Calibration of various components had to be performed; these component calibrations can be found in appendix AA.

10.1 Test Methodology

10.1.1 Research Validation

The functionality of a four-wheeled swerve drive system has been proven by both Holmberg et al. [39], and Chikosi [49]. This research aims to prove that a novel two-wheeled variant works at least as well as traditional four-wheeled variants. To this end, a set of tests need to be conducted. These tests must prove that this swerve-drive system should perform the same tasks as a traditional four-wheeled system in a stable manner.

Previous work on a similar two-wheeled system was conducted by Wada et al. [50] in 1996. Wada et al.'s system lacked both a suspension system and SIL safety system, the inclusion of which adds significant complexity to the design and makes it utterly unique.

10.1.2 Test Operation

Four operational tests were performed to validate if the AGV in this research can perform equivalently to a traditional four-wheeled suspension-less AGV. These tests can be performed stably by a traditional swerve-drive AGV as validated by both Holmberg et al. [39] and Chikosi [49].

Thus if this thesis' AGV passes these tests, it proves this thesis's research goals. The tests are:

- Straight Line Test
- Strafe Test
- Ackerman Steering Test
- Combination Test

Each test was run five times, with the average of the five tests used for analysis. Each of these five repeats contains approximately 6000 samples each. As mentioned in the PLC programming section (section 9.2), the way a test is created is by recording a set of inputs used to perform movements in manual mode. This recording was then streamed five times into the control system of the AGV in order to get the system to reproduce a set of identical motions.

The variables streamed to the PLC were:

1. Pendant speed potentiometer value
2. Pendant Ackerman steering angle potentiometer value
3. Pendant strafe angle potentiometer value

The resultant values that were recorded during each test were:

1. Setpoint centroidal x-component velocity ($^m/_s$)
2. Setpoint centroidal y-component velocity ($^m/_s$)
3. Setpoint centroidal yaw rate ($^{rad}/_s$)
4. Unit A steering RPM ($^r/_{min}$)
5. Unit B steering RPM ($^r/_{min}$)
6. Unit A traction RPM ($^r/_{min}$)
7. Unit B traction RPM ($^r/_{min}$)
8. Actual centroidal x-component velocity ($^m/_s$)
9. Actual centroidal y-component velocity ($^m/_s$)
10. Actual centroidal yaw rate ($^{rad}/_s$)
11. Actual unit A steering angle (deg)
12. Actual unit B steering angle (deg)
13. Actual centroidal angle (deg)

The results of each test are listed in the sections that follow.

10.2 Straight Line Test

The straight-line test locks the steering (Ackerman and strafe) to zero degrees. I.e. the AGV is travelling in a straight line. Only the speed potentiometer on the pendant was used; the other two pots were left in the "neutral" position. This resulted in the centroid setpoint velocities shown in figure 10.1.

Note, in figure 10.1, the only value that changes is the AGV's setpoint centroidal x-component velocity (see figure 5.1 for explanation of the axes system). The setpoint centroidal y-component velocity and setpoint centroidal yaw rate remain zero. This behaviour is expected as the AGV is only moving in the forward (x-axis) direction.

This test was started at 0.2 $^{m}/_{s}$ as illustrated in figure 10.1. This starting point was chosen to saturate the AGV's acceleration, as it would have to jump from 0 $^{m}/_{s}$ to 0.2 $^{m}/_{s}$ immediately (obviously, acceleration limits were imposed on using the servo drives configuration). Next, the system was ramped down to near 0$^{m}/_{s}$, before being very gradually being ramped back up to 0.2 $^{m}/_{s}$. Following this, the speed of the AGV was ramped up and down three more times, with each successive ramp accelerating and de-accelerating faster than the last. These ramps are visible as the last three spikes between time 40s and 55s in figure 10.1. The AGV was run at a fixed speed between 55s and 60s. The test was stopped at approximately the 65s mark.

Figure on next page

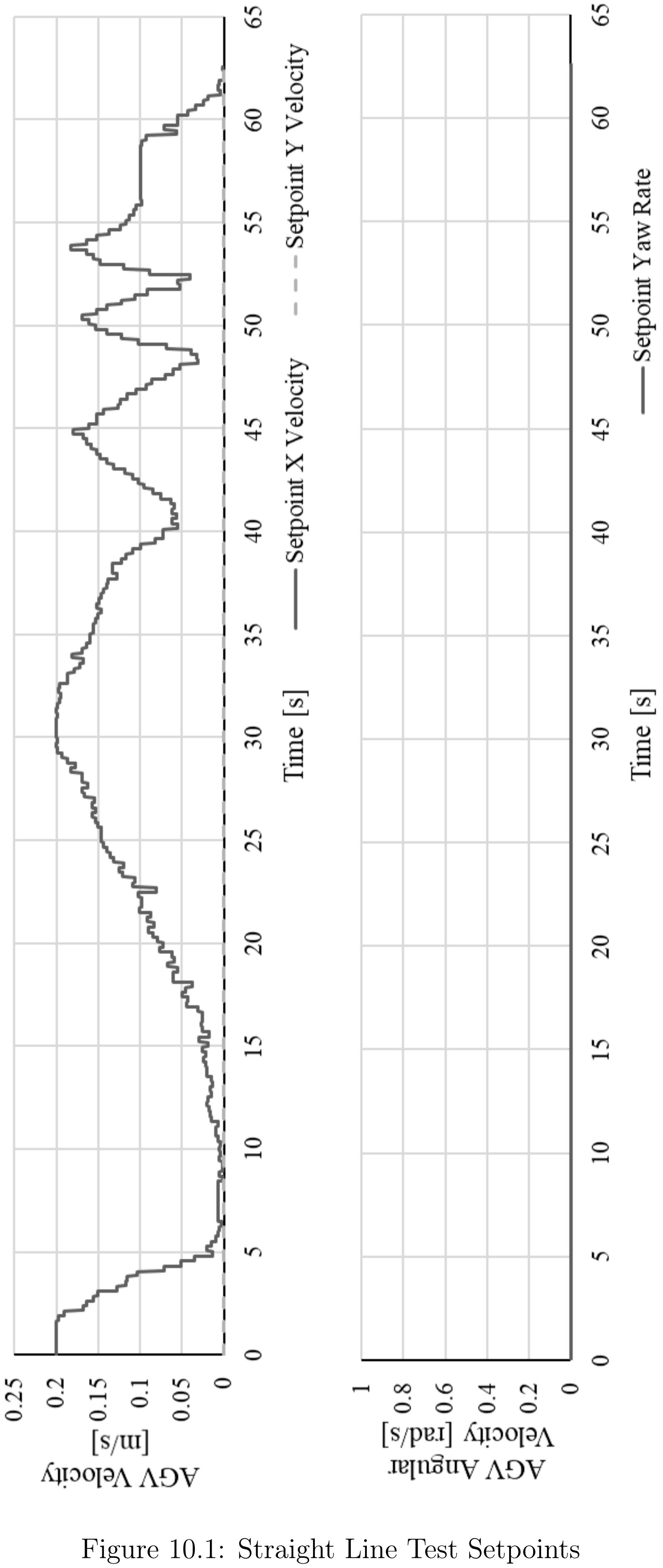


Figure 10.1: Straight Line Test Setpoints

From the setpoint values developed in figure 10.1. The kinematic model calculated the required traction and steering speeds to ensure that the centroid of the AGV moved per these desired setpoints.

Since there is no centroidal component velocity in the y-direction or a yaw rate caused by Ackerman steering, the steering motors remain at zero RPM throughout the entire test. Hence why, these graphs were not included in the results. The two traction motors, however, had non-zero RPMs as illustrated in figure 10.2.

The tractive RPM values shown in figure 10.2, represent the actual wheel RPM. This value was determined by reading the motor encoder's RPM and dividing this value by the gearbox ratio.

As illustrated in figure 10.2, the immediate jump from 0 $^{m}/_{s}$ to 0.2 $^{m}/_{s}$ at the $<$ 1s mark, results in the highest acceleration of the wheels. Both drive wheels go from 0 $^{r}/_{min}$ to 25 $^{r}/_{min}$ in under 1 second (as illustrated in figure 10.2). After 1 second, the acceleration of the traction motors will not be saturated, so for the most part, the curve of the RPM of the wheels can be matched to the centroidal setpoint x-velocity curve in figure 10.1. This behaviour is expected, as when the steering is not used, the speed potentiometer on the pendant essentially becomes the setpoint for the wheel speed.

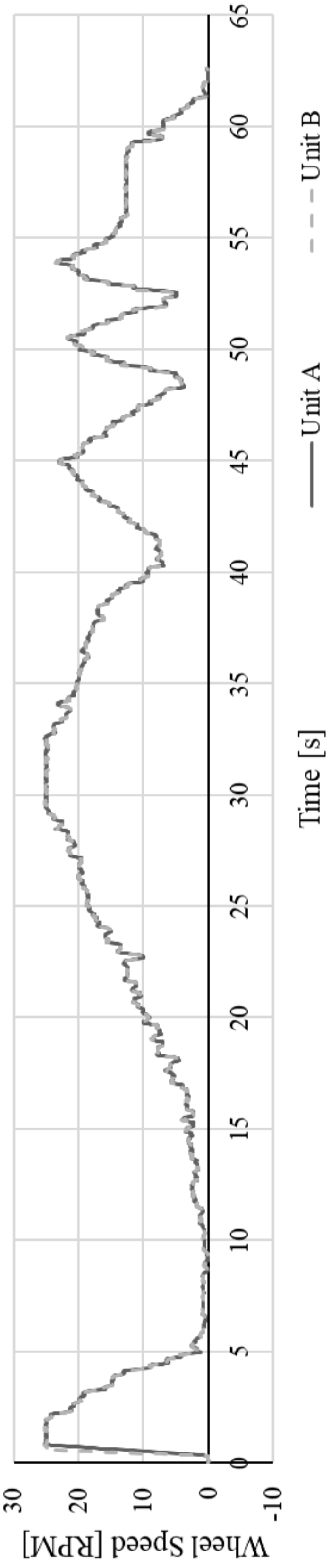


Figure 10.2: Straight Line Test Tractive Wheel RPMs

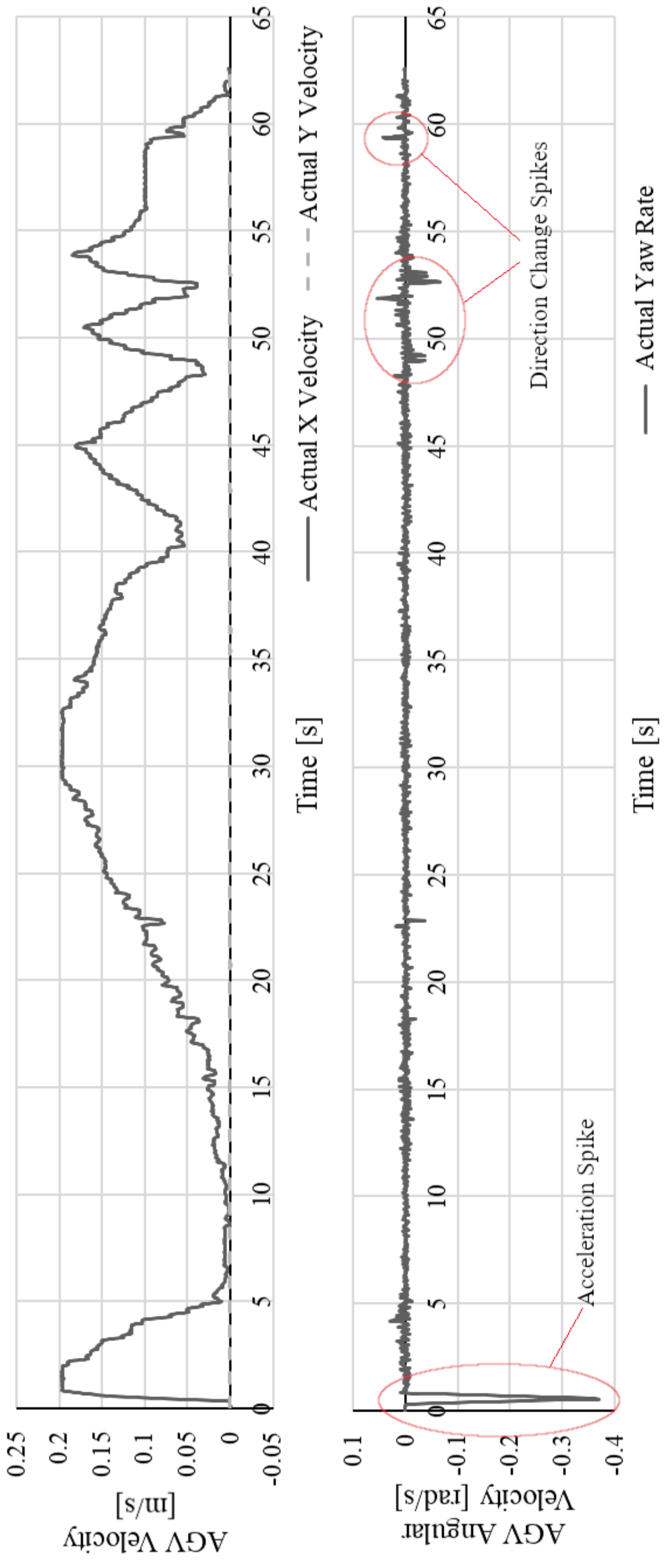


Figure 10.3: Straight Line Test Actual Values

From the actual RPM of the steering and traction motors, it was possible to use the kinematic model to calculate the actual centroidal x-component velocity, actual centroidal y-component velocity and actual centroidal yaw rate.

These results were plotted in the graphs contained in figure 10.3. In figure 10.3, it can be seen that the actual y-component velocity is zero throughout the test, which is entirely expected as the AGV travelled in a straight line during this test.

The centroidal x-component velocity is almost an exact match for the setpoint x-component velocity (shown in figure 10.1), which is to be expected as the actual value should closely follow the setpoint. However, there is a slight discrepancy between the two graphs at the <1s mark. This discrepancy was where the traction motor acceleration was saturated, and as a result, there was a delay between when the setpoint dictated that the AGV should be moving at 0.2 $^{m}/_{s}$ and when the AGV was moving at 0.2 $^{m}/_{s}$.

Also of note is the yaw rate graph shown in figure 10.3. One would expect this graph to be constant at zero, like the setpoint graph in figure 10.1 but this is not the case. Since the traction and steering mechanism share a common axis (as explained in section 5.3.5.e), rapid changes in the behaviour of the traction can cause the steering mechanism to shift from its setpoint position (zero in this case). The closed-loop control of the steering mechanism then will quickly bring the steering back to its correct setpoint position. This behaviour is visible as spikes on the actual yaw graph of the AGV that correspond time-wise to inflexions of the actual centroidal x-component velocity. These spikes are highlighted in red on the actual yaw rate graph in figure 10.3. The worst offender is the spike that occurs during acceleration saturation, named the "Acceleration Spike" the remaining spikes called "Direction Change Spikes" have minimal effect and are not noticeable in the real world.

10.3 Strafe Tests

During the strafe test, the setpoint speed of the AGV was kept as constant as possible. The speed pot was ramped up from zero to the point where the tractive wheels were rotating at 16 RPM (see figure 10.5). The strafe angle of the AGV (holonomic motion with AGV's reference frame remaining fixed relative to the universal frame) was then adjusted so that the machine was strafing either left or right. Strafing was done using the strafe potentiometer on the pendant. The sequence of movement is as follows:

- turn anticlockwise 90°quickly then back to 0°quickly

- turn clockwise 90°quickly quickly then back to 0°quickly
- turn anticlockwise 90°slowly then back to 0°slowly
- turn clockwise 90°slowly then back to 0°slowly
- turn anticlockwise 180°slowly then back to 0°slowly
- turn clockwise 180°slowly then back to 0°slowly

The motions described in the previous itemised list will change the ratio between the x-centroidal component velocity and y-centroidal component velocity. The angle describes the ratio change between these two developed as a result of the strafe potentiometer setting, which is illustrated in equation 10.1.

$$\begin{aligned} x_v &= v_{AGV} \cos(\xi_{strafe}) \\ y_v &= v_{AGV} \sin(\xi_{strafe}) \end{aligned} \tag{10.1}$$

x_v	=	X centroidal component velocity	${}^{m}\!/_{s}$
y_v	=	Y centroidal component velocity	${}^{m}\!/_{s}$
v_{AGV}	=	Setpoint velocity of AGV (from speed pot)	${}^{m}\!/_{s}$
ξ_{strafe}	=	Setpoint strafe angle (from strafe pot)	rad

The yaw rate of the AGV will remain zero for this test as no Ackerman steering takes place, and as such, the reference frame of the AGV will maintain its orientation with reference to the universal frame.

The setpoint x-centroidal component velocity and y-centroidal component velocity are illustrated in equation 10.1, along with the zero yaw rate throughout the test. The results of these measurements are illustrated in figure 10.4. Note that this test took 2 minutes (120s) to run.

The setpoint values shown in figure 10.4, when fed into the kinematic model, caused the wheels and steering to move, as shown in figure 10.5 to follow these setpoints.

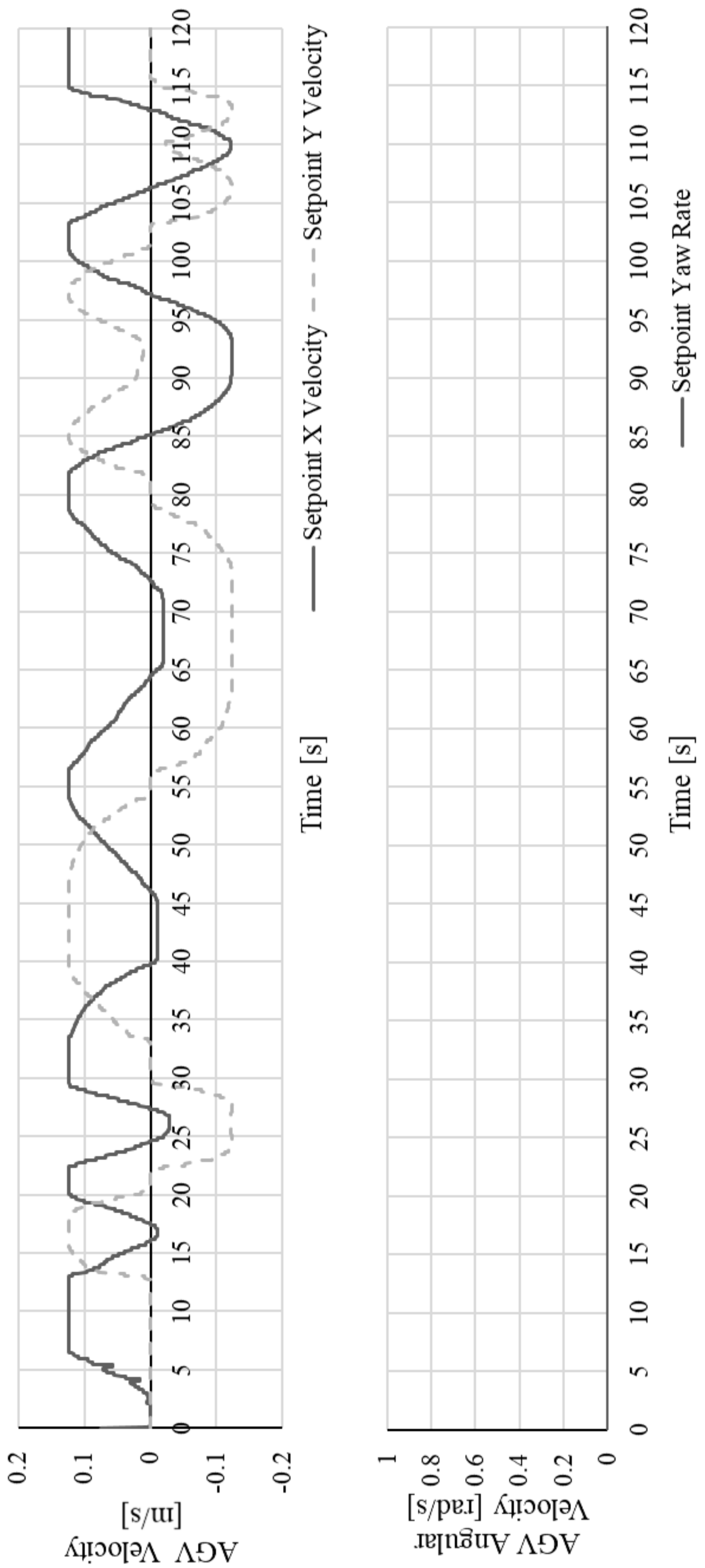


Figure 10.4: Strafe Test Setpoint Centroidal Values

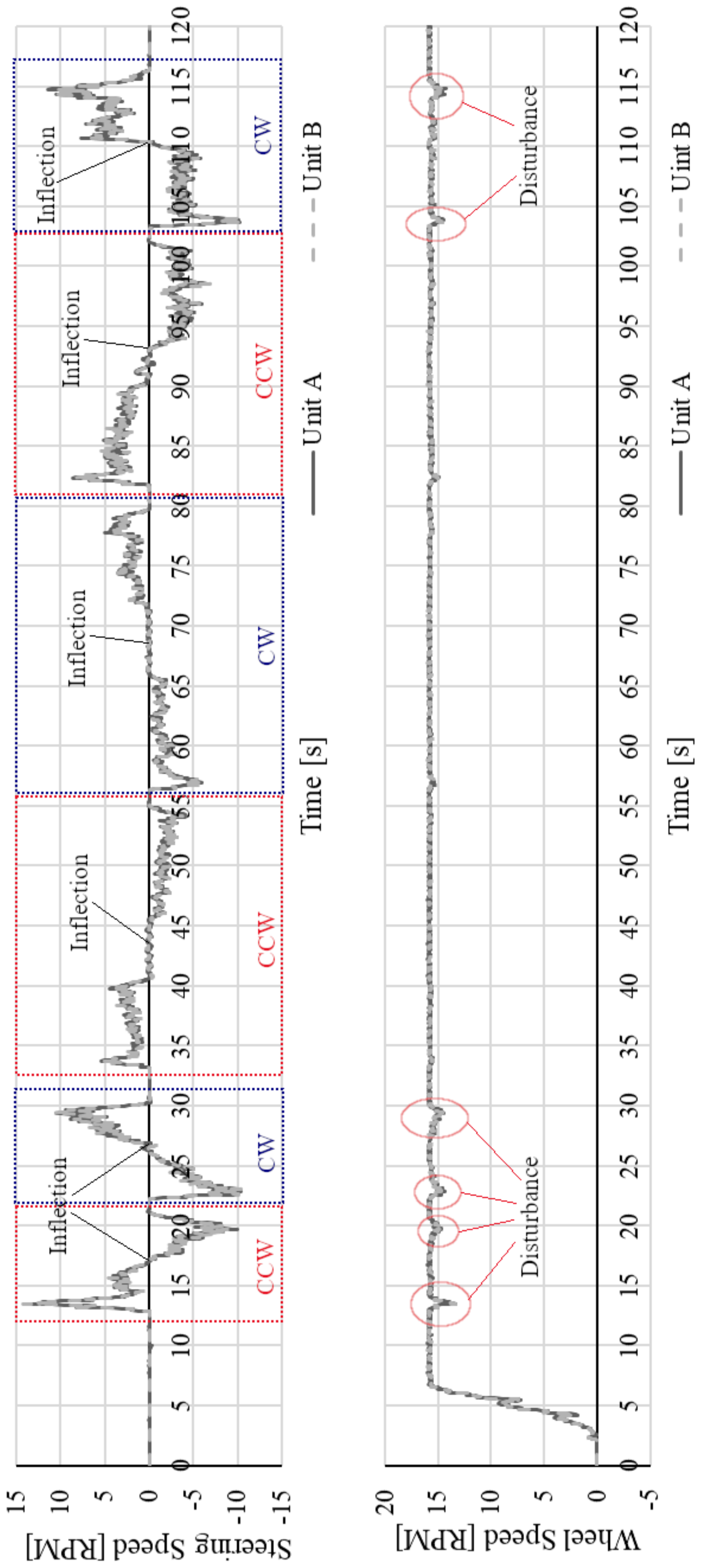


Figure 10.5: Strafe Test Tractive and Steering RPMs

As can be seen in figure 10.5, the tractive (wheel) RPM remains relatively constant after the AGV is ramped up from zero RPM, with only minor disturbances. These disturbances coincide with fast steering motions and are caused by the traction and steering mechanisms coupling due to their shared axis (see figure 5.3.5.e). Coupling only occurs at high steering speeds as it is at these speeds that the compensation algorithm's cycle time is too slow to compensate correctly. These spikes are temporary, and the compensation algorithm will return the speed to steady-state conditions given time. These disturbances are highlighted in red circles on the wheel speed graph in figure 10.5.

On the steering speed graph in figure 10.5, the steering motions of the AGV have been highlighted. When enclosed in a red dotted block, the steering is attempting to move the strafe direction of the AGV in a counter-clockwise direction and then back to zero. The direction is evident as the AGV's steering motion (for both drive units) increases in the positive direction (CCW id defined as positive), peaks, and then decreases to zero RPM. This Zero RPM point represents the steering angle zenith and the inflexion point where the steering angle will return to zero (the negative RPM after this point). The sections of the graph enclosed in a dotted blue line represent clockwise motion for the AGV, as the RPM accelerates in a negative direction, decelerates to the inflexion point (maximum steering angle

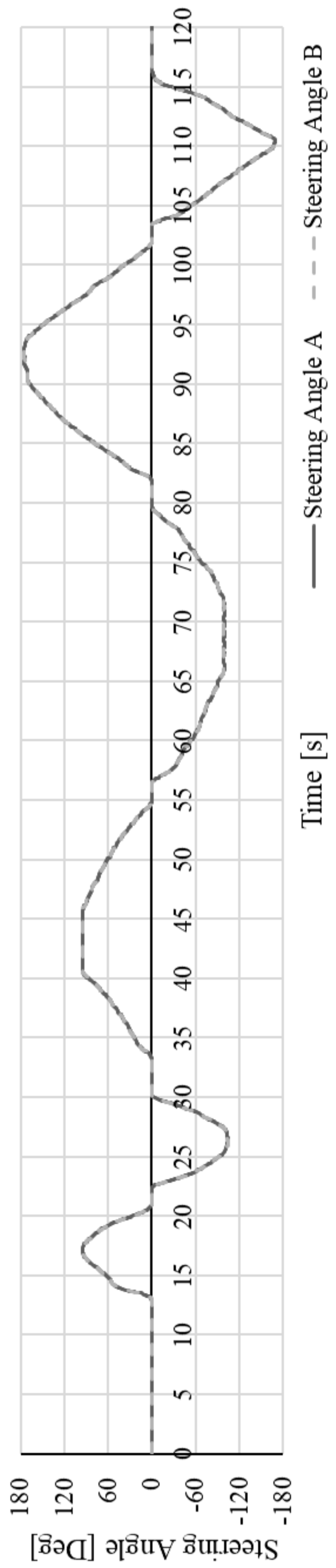


Figure 10.6: Strafe Test Drive Unit Steering Angles

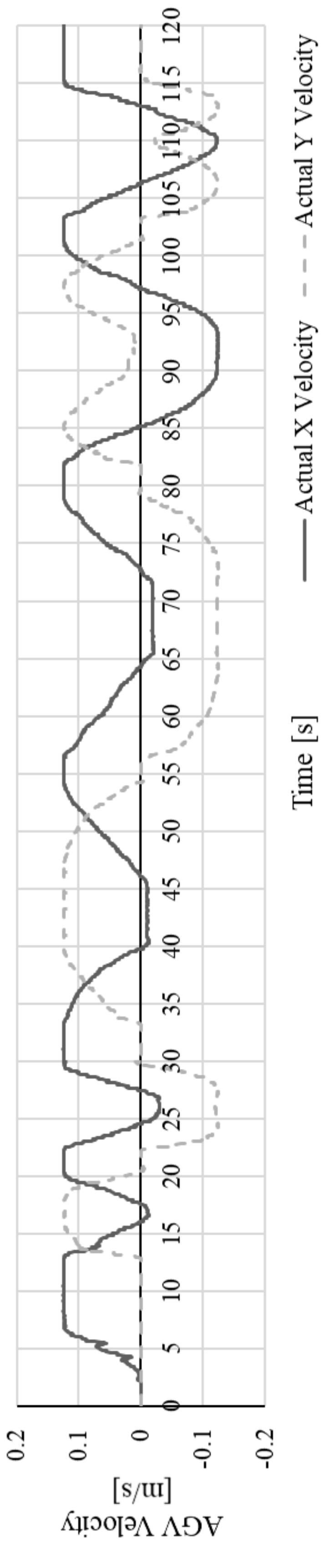


Figure 10.7: Strafe Test Actual Centroidal Values

point) then accelerates in the positive direction to the zero degrees angle point.

The motions in the previous paragraph can be more easily understood if viewed as an angle value rather than a velocity value. This analysis can be done with the aid of figure 10.6, which represents the steering angles of the drive units with reference to time. Figure 10.6 gives the angle in degrees as measured directly from the relative encoder on the stepper motors (multiplied by a gear ratio).

From the actual values produced by the encoders (both on the tractive system and steering system), the real-world values were captured and represented in figure 10.5. When these values are fed into the kinematic model in reverse, the actual centroidal x-component and y-component velocities can be determined along with the actual centroidal yaw rate as illustrated in figure 10.7. It is then possible to compare the actual values to the setpoint values given in figure 10.4 to determine the accuracy of the control system.

When comparing the actual (figure 10.7) and setpoint (figure 10.4) centroidal motion values it can be seen that the actual value very closely matched the setpoint value. This result was expected since the drives were not saturated and were able to keep their speed in sync with the setpoint speed.

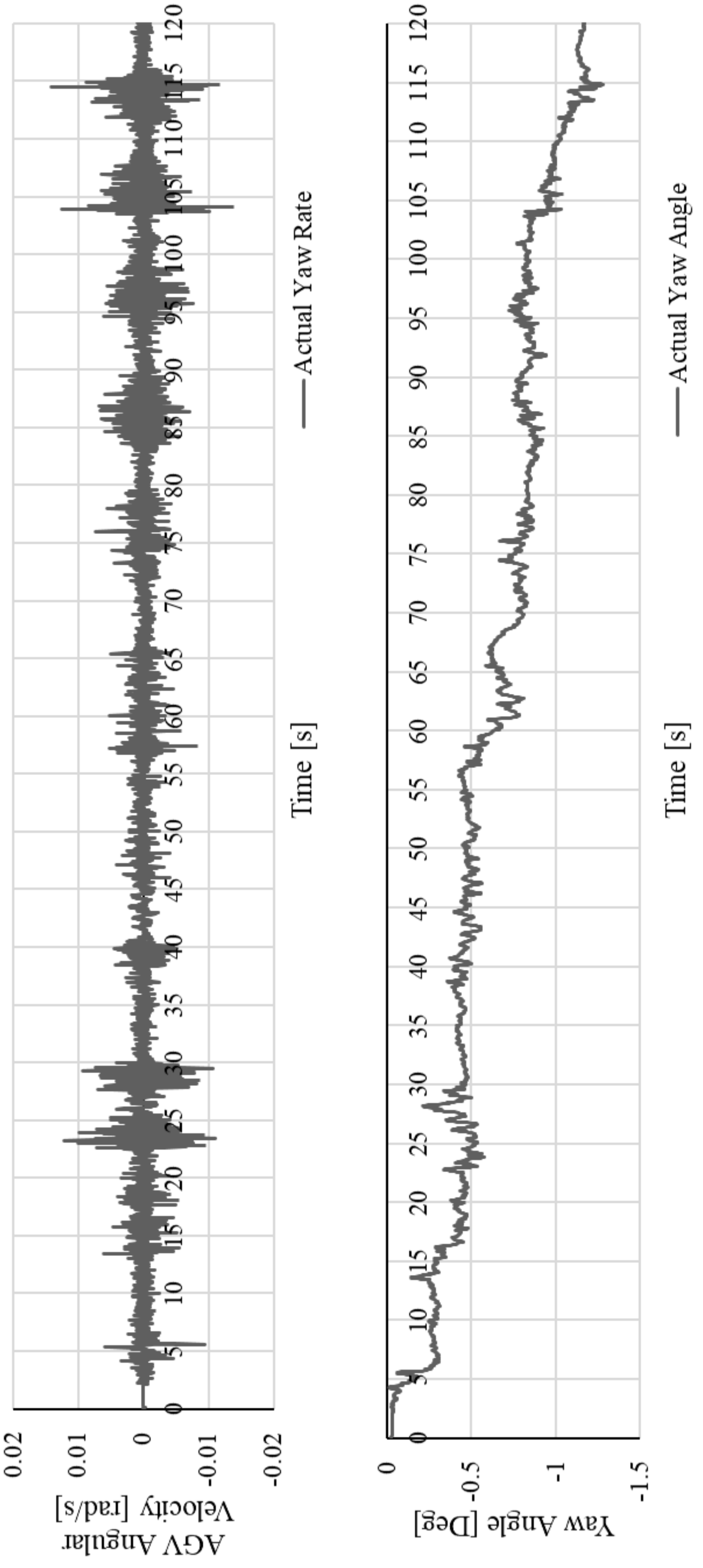


Figure 10.8: Strafe Test Yaw Rate and Yaw Angle

The last set of values that need to be examined is the behaviour of the yaw of the AGV. These values include the actual yaw rate and the actual yaw angle of the AGV. In figure 10.4 the setpoint yaw rate is shown to be zero throughout the test; however, the actual yaw rate does not remain at zero. This effect is illustrated in figure 10.8.

Although there is plenty of jitter about the actual centroidal yaw rate (as illustrated in figure 10.8), this jitter is centralised and balanced around the 0 $^{rad}/_{s}$ point. With the peaks extending to only $0.01^{rad}/_{s}$ (0.095 $^{r}/_{min}$). Thus, this jittery is negligible and not noticeable under normal observation for the most part.

Of concern, however, is the drift experienced by the yaw angle. This angle should have remained zero throughout the entirety of the test; however, as illustrated in figure 10.8, the yaw angle drifted consistently by approximately 1° every 110 seconds in the clockwise direction. The yaw angle is produced by integrating the actual yaw rate. Since the actual yaw rate is relatively consistent about the zero point (see yaw rate graph in figure 10.8), likely, the time interval used for the numerical analysis based integration (trapezoidal method) is not as stable as it should be. That is to say; there exists jittery about the sample point times. Stabilising the sample times is not easy to fix as the cyclic interrupt system (used for the sampling) on Siemens PLCs is closed-source; the programmer can only set an interval time.

The easiest solution to this issue would be to run the integrated value through an error correction control loop that used the navigation sensor data (from the NAV 350) to correct this drift or abandon the integration altogether and only use feedback from the NAV 350 sensor for the yaw angle.

10.4 Ackerman Steering Test

Ackerman steering involves changing the heading of the AGV (i.e. the AGV's frame will rotate relative to the universal frame) like a car. The sequence of motion for this 85 second test is listed in the itemised text:

- hard steer anti-clockwise (left) then return to zero
- hard steer clockwise (right) then return to zero
- gradual steer clockwise (right) then return to zero
- gradual steer anti-clockwise (left)
- continue to hard steer anti-clockwise (left) and return to zero

- extremely hard steer clockwise (right) and return to zero

During the steering test, the AGV's linear velocity was kept constant in the x-direction (relative to the AGV's reference frame) at $0.9\ ^{m}/_{s}$. This behaviour is illustrated in the setpoint centroidal velocity graph contained in figure 10.9. Since Ackerman steering is performed during this test, the setpoint yaw rate, for the first time, is non-zero. The yaw rate value (figure 10.9 is determined by the Ackerman steering pot found on the pendant.

The setpoint values described in figure 10.9 were fed into the kinematic model by the AGV's control system to produce a set of tractive and steering RPMs (shown in figure 10.10). In order to perform Ackerman steering correctly, both the tractive velocity of the wheels themselves and the steering angle they moved to had to be varied. If only the steering angle were to change and the tractive RPM of the wheels were to remain constant, scuffing would occur. Thus, the RPMs of the wheels were varied to produce a virtual differential effect, as can be seen in the wheels speed graph contained in figure 10.10. The need for a virtual differential is explained in figure 2.12.

To simplify understanding, the motions described in the previous itemised list (for this test) are highlighted by enclosing them in dotted boxes. Red dotted boxes represent clockwise steering relative to the universal frame (i.e. the AGV is turning right), while blue dotted boxes represent counter-clockwise steering relative to the universal frame (i.e. the AGV is turning left). When interpreting the steering speed, it is important to note that the speed will "ramp up" for a deterministic amount of time to move the steering to the desired steering angle. Once this angle has been reached, the steering speed will drop to zero RPM. The AGV will turn about the yaw until the steering is "ramped down" to an angle of zero. The "ramp up and "ramp down" is labelled in figure 10.10 for clarity.

The direction of yaw rotation in figure 10.10 can be determined by establishing which unit is rotating in a positive direction (for Ackerman steering, the other unit will always rotate counter to this, i.e. in a negative direction). If the AGV is turning anti-clockwise relative to the universal frame (i.e. turning left), then the "ramp up" would have been positive for unit A and negative for unit B. The opposite will be true if the AGV turns clockwise (i.e. right) relative to the universal frame.

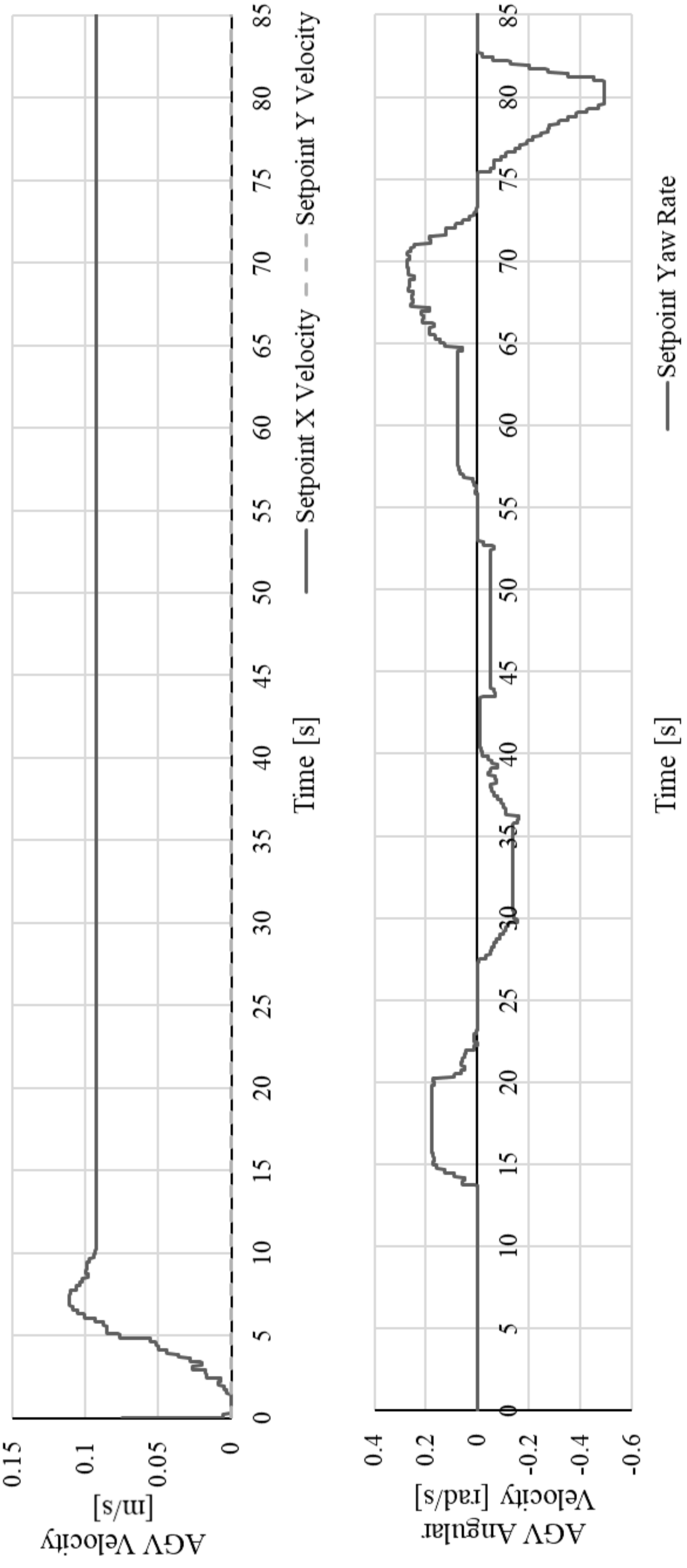


Figure 10.9: Ackerman Steering Setpoint Centroidal Speeds

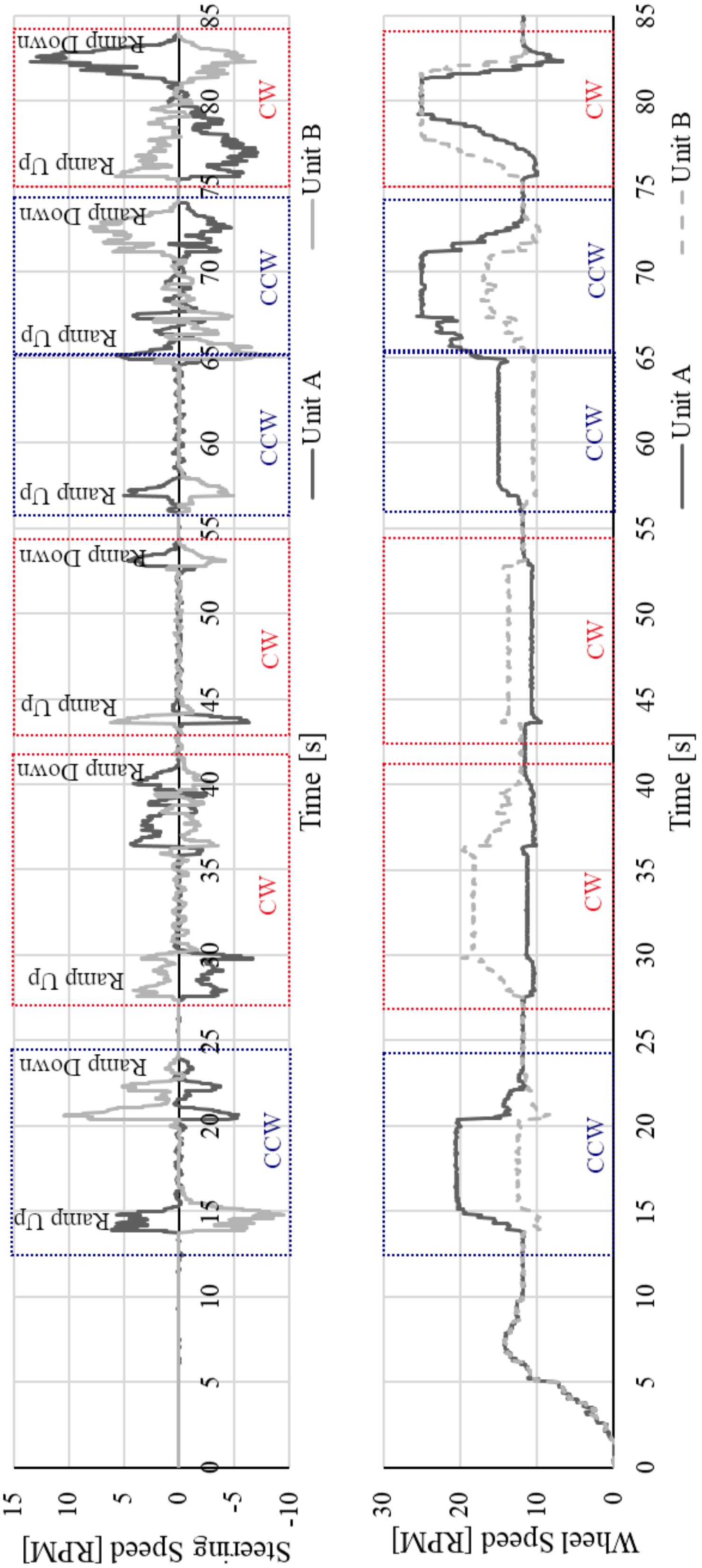


Figure 10.10: Ackerman Steering and Tactive System RPMs

This steering direction effect is also visible in the wheels speed graph in figure 10.10, where one of the wheel's speeds is boosted relative to the other (to create a virtual differential). The unboosted unit remains at the nominal speed of 12 $^{r}/_{min}$. For clarity:

Counter-clockwise (left) steering	Unit A speed boost
Clockwise (right) steering	Unit B speed boost

To prevent scuffing, the RPM of the wheels will have to change to replicate a virtual differential. This behaviour, however, breaks down when the boost speed reaches saturation (i.e. maximum allowable wheel RPM). When this happens, the non-boosted unit will also experience a speed boost, as can be seen in the last CCW and CW motions of the wheel speed graph in figure 10.10. Note that the last CW (counter-clockwise motion) attempts to turn the AGV left at such a sharp turning radius for the given speed that both units' speed becomes saturated, and the actual and setpoint values will deviate.

The actual wheel and steering RPM (described in figure 10.10) were fed into the reverse kinematic model to determine the actual centroidal x-component velocity, y-component velocity and yaw rate. These results are represent in figure 10.11.

For the most part, the actual speed values mirror the setpoint values given in figure 10.9. As previously mentioned, this relationship only breaks down when the traction speed is saturated. Between the 65 to 75 second mark on the AGV velocity graph in figure 10.11, minor instability occurs on the x-component velocity. This point was where the speed of Unit A was saturated, and as such, Unit B's speed had to be increased to maintain the desired steering arc. However, even with the minor instability, the actual yaw rate could be maintained to match the desired yaw rate shown in figure 10.9. However, the system falls apart when both units' speeds are saturated (between 77 and 83 seconds). At this point the actual yaw rate (AGV angular velocity graph in figure 10.11) deviates from the setpoint yaw rate in figure 10.9. This deviation is marked with a red deviation bar in figure 10.11.

The saturation of the steering speed between 77 and 83 seconds also caused the AGV to inadvertently strafe when it was not supposed to as it could not maintain the desired steering arc. This behaviour can be seen in figure 10.11 as the y-centroidal component deviating away from zero RPM and peaks at the 80-second mark.

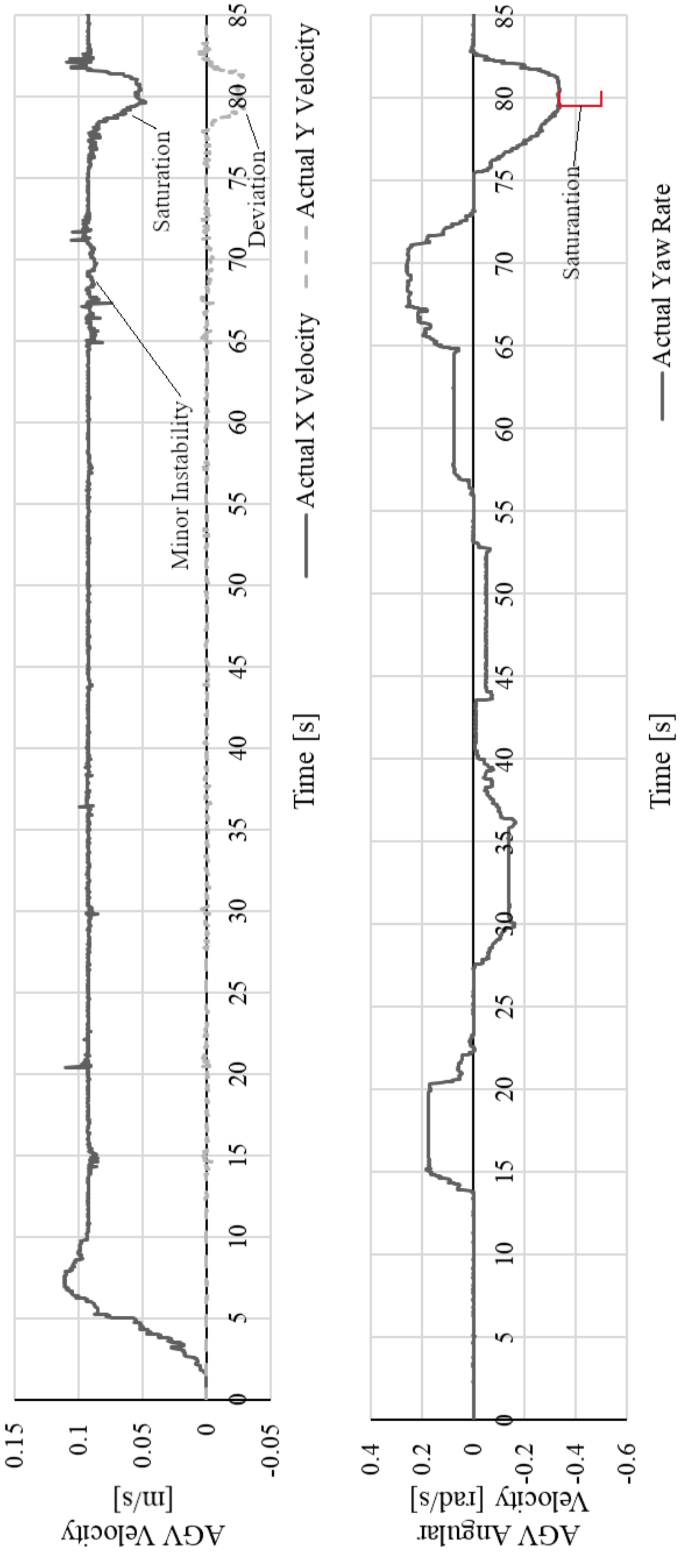


Figure 10.11: Ackerman Steering Actual Centroidal Speeds

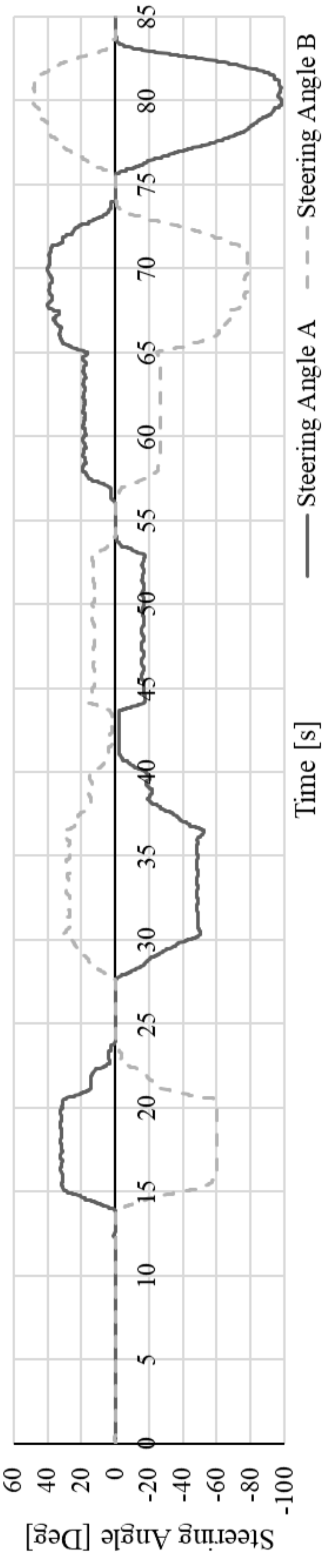


Figure 10.12: Ackerman Test Drive Unit Steering Angles

The steering angles of the drive units during this test are illustrated in figure 10.12. It is important to note that the steering angle does not have the same magnitude but opposite direction as would intuitively be expected. As can be seen in figure 10.12, the angles have different magnitudes. These magnitudes diverge according to a fixed constant resulting from the speed of the AGV and the desired yaw rate. The higher the desired yaw rate (and thus the tighter the turning arc), the larger this constant will be. For more information on why the angles should have opposing signs when performing Ackerman steering see section 2.3.4.

When the steering angle plateaus at a non-zero value in figure 10.12, the AGV will turn at a constant rate relative to the universal frame. That is to say, the steering arc of the AGV will remain constant, and if the steering angle does not change, the AGV will turn in a circle whose radius magnitude is determined by the steering arc/ yaw rate. This is illustrated in figure 10.13.

Nothing prevents the AGV from performing a full 360° rotation relative to the universal frame. For figure 10.13 a full 360° is mapped between -180° and 180°. Thus, whenever the AGV's yaw angle crosses over the 180° mark, it is mapped onto the opposite sign spectrum. This effect is visible in figure 10.13 as a vertical straight line that jumps from -180°to 180°or vice versa.

It must be noted that although the integration drift is not easily visible in figure 10.13, it still exists as described in figure 10.8. This drift is 1°every 110 seconds in the clockwise (negative) direction.

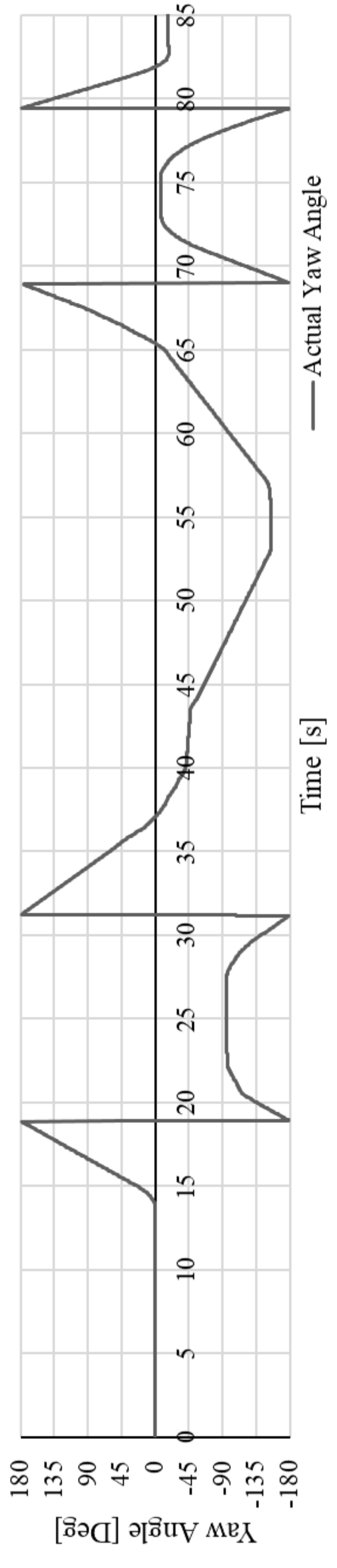


Figure 10.13: Ackerman Test AGV Yaw Angle

10.5 Combination Test

The combination test combines the three previous tests to check if there is any unexpected behaviour when running these different manoeuvrers together. The sequence of the test is as follows:

- Ramp up to constant velocity (50°max velocity)
- Strafe in the anticlockwise (left) direction then return to zero
- Strafe in the clockwise (right) direction then return to zero
- Ramp speed up to max speed then back down to 50°
- Ackerman steer anticlockwise (left) then return to zero
- Ackerman steer clockwise (right) then return to zero

Therefore, this test will involve all three control potentiometers on the pendant: the speed pot, strafe pot, and steering pot. The values of these pots with respect to time were used to develop the centroidal setpoint velocities and centroidal yaw rate of the AGV. These centroidal velocities and the yaw rate are given with respect to time in the graphs contained in figure 10.14. This test took 2 minutes and 10 seconds to complete (130s).

In figure 10.14 the change in linear speed of the AGV are enclosed in green blocks, the strafe manoeuvrers are enclosed in blue blocks, and Ackerman steering is enclosed in red blocks.

These three setpoint values (in figure 10.14), once fed into the kinematic models, would produce a set of steering and traction RPMs that the tractive and steering motors would run at. The resultant wheel speed and steering speed, as recorded by encoders, are given in figure 10.15.

Figure on next page

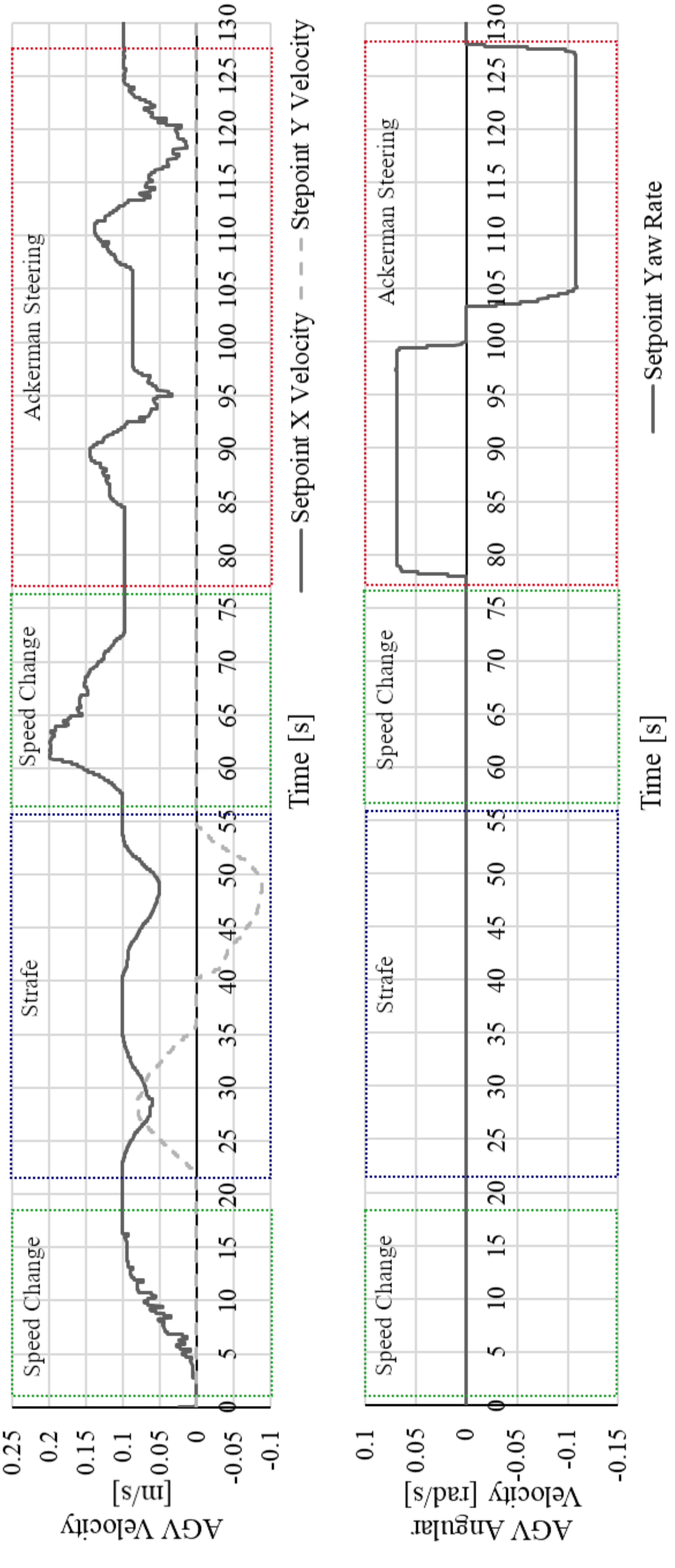


Figure 10.14: Combination Test Centroidal Setpoint Values

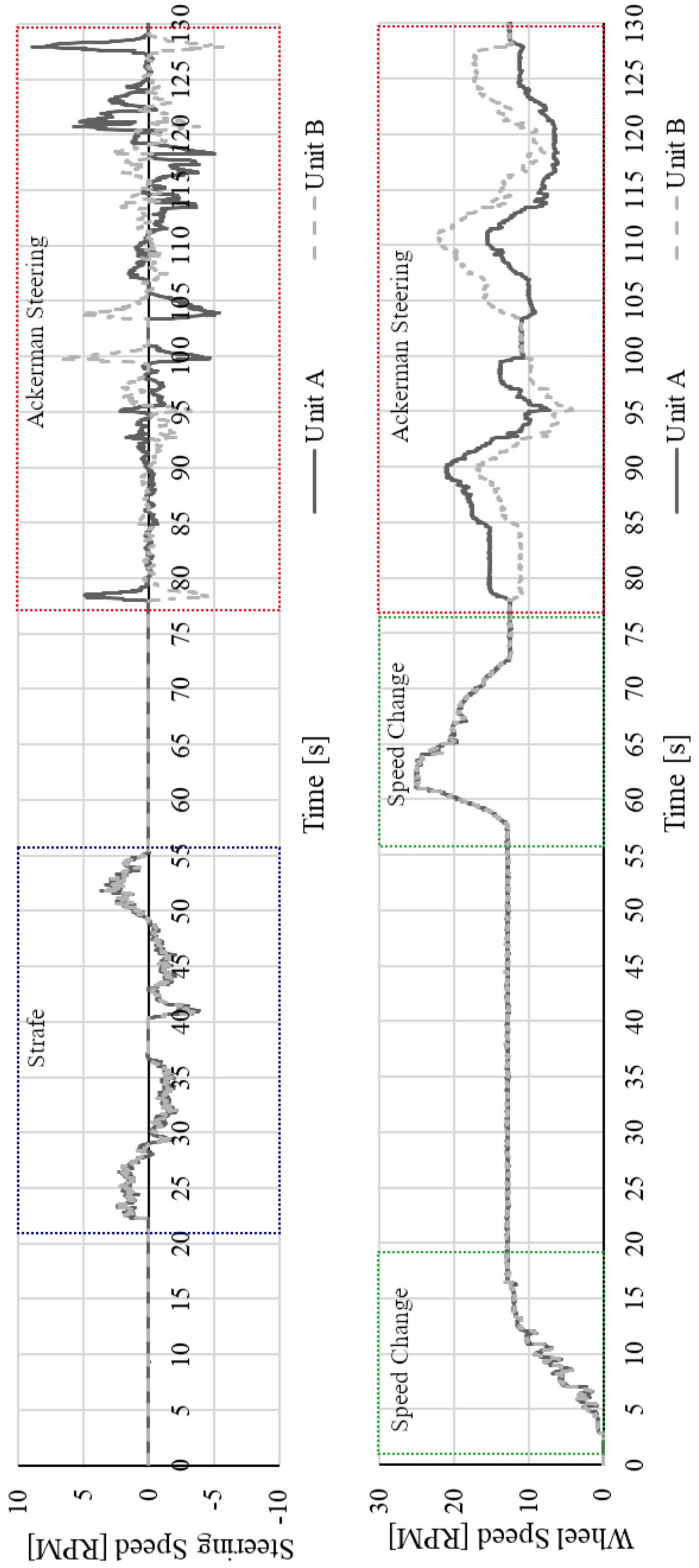


Figure 10.15: Combination Test Wheel and Steering RPMs

In figure 10.15, the different manoeuvrers are once again highlighted in different coloured blocks (green = speed change, blue = strafe and red = Ackerman steering). From this, it is possible to deduce that a speed change will only affect the wheel RPMs, with both units having the same wheels' RPM magnitude and sign (see figure 10.15), the steering RPMs remain constant. When the AGV is strafing to either the left or right, the wheel speed remains constant, and only the steering RPMs are affected. Both units' steering RPMs have the same magnitude and sign during this operation. Finally, both the steering and wheel RPMs are affected when a manoeuvrer is performed that uses Ackerman steering. Ackerman steering causes the steering RPMs to have opposite signs and whose magnitude is a constant scaling dependent on the setpoint yaw rate. The wheel RPMs during Ackerman steering have the same sign, but one of the unit's speeds will be boosted relative to the other to create a "virtual differential". The unit whose speed is boosted depends on whether the Ackerman steering turns the AGV left or right (left = unit A speed boost, right = unit B speed boost).

The reverse kinematic model, when used on the actual RPM values recorded in figure 10.15, will produce the actual centroidal x-component velocity, y-component velocity and actual centroidal yaw rate. These actual centroidal values are given in figure 10.16.

When the setpoint values, in figure 10.14, are compared to the actual values in figure 10.16 they are almost identical except for some minor noise. This effect was expected as at no point during this test was either the acceleration or velocity of any motors driven to saturation.

Figure on next page

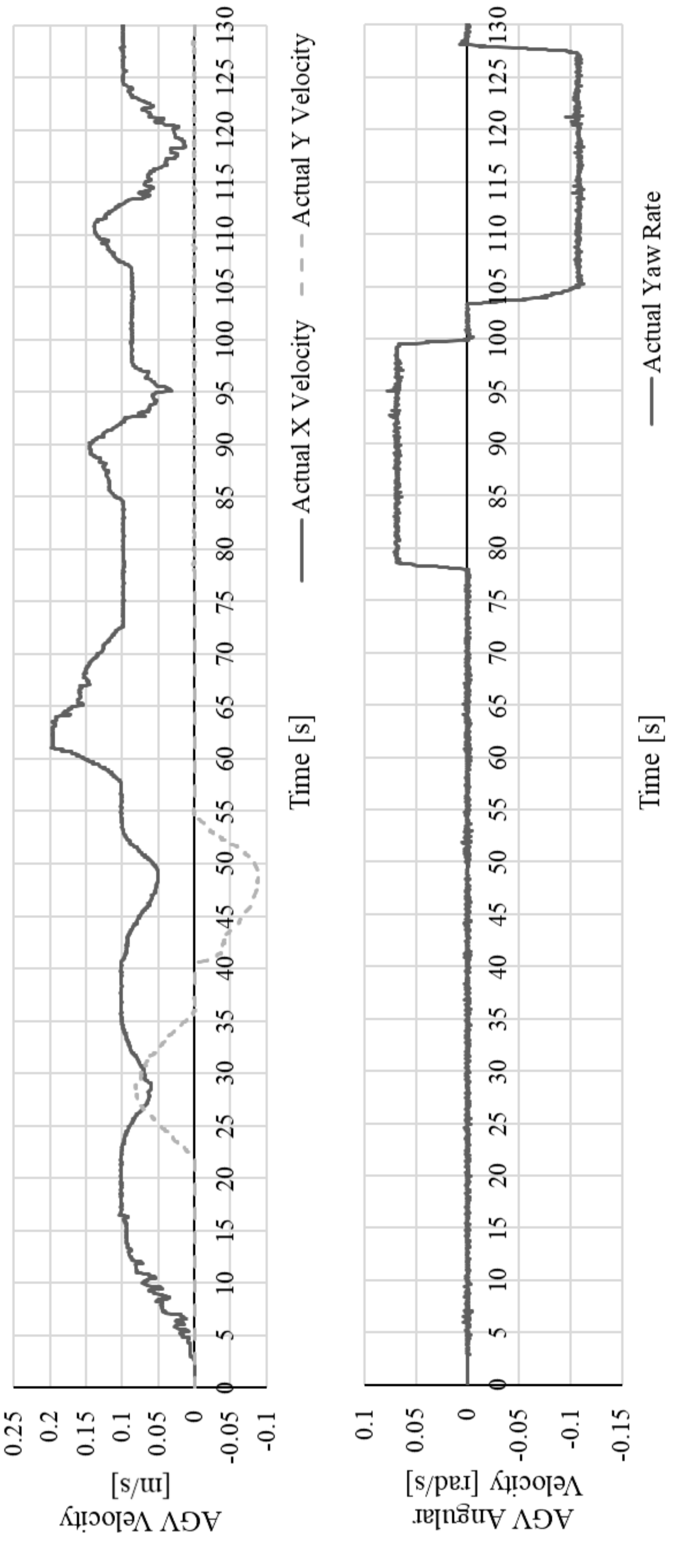


Figure 10.16: Combination Test Centroidal Actual Values

The steering angles of the drive units, captured using the stepper motor encoders and a gear ratio calculation, are given in figure 10.17. The angle of the wheels during a strafe manoeuvrer is enclosed in a blue box in figure 10.17, while during an Ackerman steering manoeuvrer, it is enclosed in a red box.

Note in figure 10.17 that when the AGV is strafing, the angles will have the same magnitude and sign. When the AGV is performing Ackerman steering, the angle of the wheels will have opposite signs and will be scaled relative to each other by a constant determined by the desired yaw rate, mirroring the behaviour seen in figure 10.15 for the steering RPMs.

A change in the setpoint speed of the AGV has negligible effects on the steering angle.

The yaw angle of the AGV is determined using integration of the actual yaw rate (from figure 10.16) and is given in figure 10.18. The yaw angle measures the angle between the AGV's frame and the universal frame; as such, it will only change from zero when Ackerman steering is used. This behaviour can be seen between the 75 to 130 second mark on figure 10.18 which corresponds to a set of Ackerman manoeuvrers during this test. It is important to note that the yaw angle drift of 1°every 110 seconds in the clockwise (negative) direction (discovered in

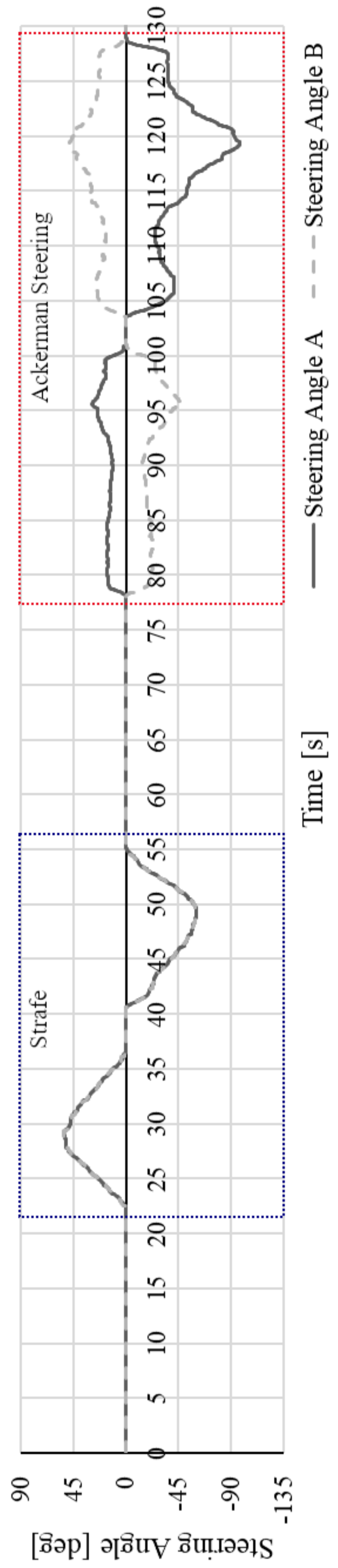


Figure 10.17: Combination Test Drive Unit Steering Angles

section 10.3) is still present but not noticeable due to the scale used in figure 10.18.

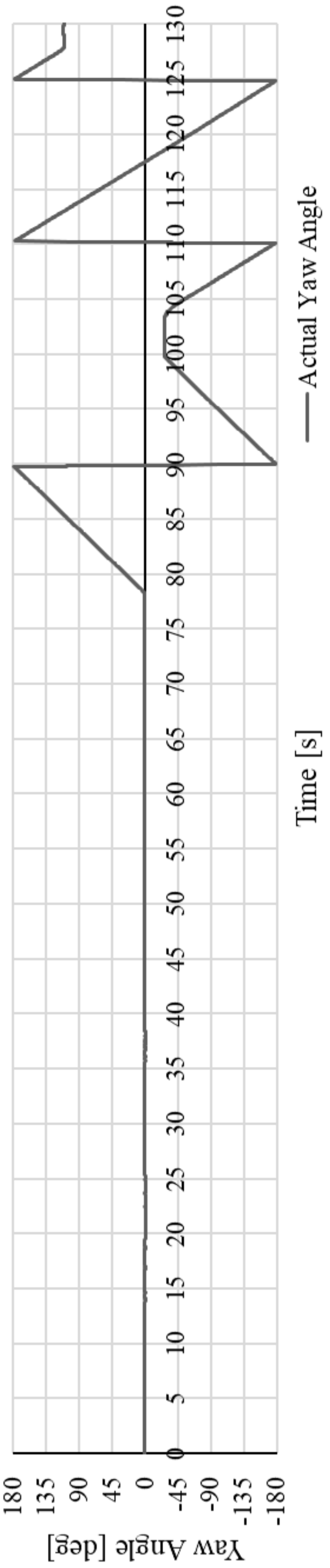


Figure 10.18: Combination Test AGV Yaw Angle

10.6 Chapter Conclusion

This chapter contains the calibration and validation tests performed on the AGV. The calibration tests were used to generate compensation algorithms to ensure that the measured values from a given sensor matched the real-world values. The validation tests confirmed that the modes of operation of the AGV performed as expected and validated the kinematic model used to achieve a two-drive unit swerve drive machine.

11 Conclusion

11.1 Discussion of Testing Results

This section discusses important observations made from the testing chapter (chapter 10). These tests were used to confirm the validity of the design, kinematics and control system. Confirmation of these tests validates the functionality of the two-wheel swerve-drive system with an integrated suspension system.

In terms of confirming validity, the AGV behaved as desired for the four tests: straight-line test, strafe test, Ackerman test and combination test; when the setpoint values of the AGV were calculated and updated every 50 ms with the speed of the AGV kept below the specified maximum of 1.3 $^{m}/_{s}$. The minimum number of control inputs that the AGV required to achieve omnidirectional operation was three. These control inputs were:

- Centroidal x-velocity
- Centroidal y-velocity
- Centroidal yaw rate

The system was controlled in the velocity domain, as seen from the listed control inputs. An attempt was made to control the steering in the position domain; however, this control strategy proved very jerky and unsuitable for stable operation.

The primary control principles for the system are as follows:

- Velocity in a straight line

  If the AGV is to be driven straight forward, then the magnitude of the centroidal x-velocity is set to the desired linear velocity. The centroidal y-velocity and centroidal yaw rate are kept at zero.

- Strafe at a given angle and speed

The strafe behaviour refers to the AGV moving linearly at an angle not aligned with the front of the AGV (the AGV's frame of reference will not change its orientation relative to the universal frame). This behaviour is achieved by setting the centroidal x-velocity and centroidal y-velocity to a calculated component magnitude of the desired linear velocity. The magnitude of these components is calculated using the desired strafe angle. During a purely strafe operation, the centroidal yaw rate of the AGV is kept at zero.

- Ackerman steering at a given speed

  For a purely Ackerman turn where the strafe angle is zero, the centroidal x-velocity is set to the desired linear velocity while the centroidal y-velocity is zero. The turning arc radius is determined by the yaw rate of the centroid coupled with the linear velocity of the AGV.

- Combination manoeuvres

  A combination of the previously mentioned operations can be achieved by superimposing these manoeuvres to achieve a centroidal x-velocity, centroidal y-velocity, and centroidal yaw rate set.

During the tests, it was found that when the acceleration of the AGV was purposefully saturated or rapid changes between deceleration to acceleration were performed (near acceleration saturation), the steering angle of the AGV would be pulled off of its desired angle. The closed-loop control corrected this effect; however, it illustrated that the shared axis between the tractive and steering systems could cause issues even with mathematical compensation. However, this issue is unlikely to affect normal operation as saturating the acceleration or rapidly changing between accelerating and deceleration in the manner needed to cause this effect will likely damage the payload long before the coupling effect is noticed. For more details on this issue refer to section 10.2.

The tractive system affects the steering system, but the steering system was observed to affect the tractive system; this was counter to expectations. This interaction was shown in the second test (strafe), where the tractive velocity was kept constant while the steering was actuated. It was observed, in section 10.3, that during a fast steering angle change, a minor disturbance occurred on the tractive speed. However, this disturbance was comparably minor compared to the effect that rapid acceleration of the tractive system can have on the steering system (as previously discussed). This behaviour is because the steering velocity's magnitude is significantly smaller than the tractive velocity. The only way the steering system could have a notable effect on the tractive system was when the Ackerman steering velocity was saturated, as observed in section 10.4. In this case, the AGV cannot turn at the desired turning arc relative to the AGV's linear speed. Thus to compensate for this and maintain the turning circle at the desired arc radius, the linear velocity decreases.

The last effect of prominence observed during the testing was the drift of the yaw angle. The yaw angle is determined using numerical analysis' trapezoidal method to approximate the integral of the yaw rate. If the sample time is not precisely constant, then drift can occur. This yaw drift is invisible for most tests because it is so small compared to the actual yaw angle change. However, during the swerve test (section 10.3), when the yaw angle remains zero for the entirety of the duration of the test, the drift can be observed. It is approximated to be 1° every 110 seconds. If the drift proves to be constant, it could be compensated for by using a compensation algorithm; however, an extended duration test would have to be run to determine if this drift is as linear as it appears to be. This issue could also be eliminated by removing the reliance of this value on integration and instead determining it by using feedback from the navigation sensor; this concept is discussed further in section 10.3.

11.2 Research Conclusion

To recap, the primary research goal of this project was to produce a novel two-wheeled swerve drive AGV with an integrated suspension system. Therefore to validate the research into this system, it must be proved to function in a manner equivalent to a traditional four-wheeled swerve drive system. Therefore the AGV had to pass the four tests denoted in the testing section (section 10).

As discussed in section 11.1, the AGV passed all four validation tests and, therefore, can act as a replacement for traditional four-wheeled swerve-drive AGVs such as Chikosi's [49] or Holmberg et al.'s [39] AGV. The AGV was still able to perform omnidirectional manoeuvres even with an included suspension system; this was due to the unique mechanism called the "vertical compromiser" (see section 5.3.3.b).

Since the AGV researched in this project has only four motors as opposed to the eight found in a traditional swerve-drive system, the cost of implementation of a swerve drive AGV could be reduced by as much as 50 %. However, this is closer to 40 % as the cost of implementing "castor units" in place of the two removed "drive units" is not negligible.

The AGV is entirely SIL safety compliant, which means this AGV can be directly implemented into the manufacturing industry, something Wada et al.'s [50] AGV (similar two-wheel swerve-drive idea) was not compliant with and therefore could not be directly implemented into industry.

Thus, in conclusion, the AGV researched in this thesis is lower cost than a traditional swerve-drive system while still performing the same tasks and having an integrated suspension system. These facts make the AGV researched in this paper an ideal candidate for replacing traditional swerve drive AGVs in the manufacturing industry, where omnidirectional capabilities are needed with poor floor conditions. This AGV has the benefit of being economically viable compared to mecanum wheel AGVs as this system has the same number of motors, four in total. Due to the added complexity of the mecanum wheel, any economic advantages that the system might have at initial capital outlay (due to a more straightforward gearing topography) are negated.

There are a couple of concerns about this system related to behaviour near motor speed saturation points. At these points, the control system performs erroneously, as discussed in section 11.1. This erroneous behaviour only occurs when the AGV is operated outside of normal operating conditions and, therefore, cannot invalidate this AGV as a replacement for the traditional four-wheel swerve-drive system. However, solutions to this issue are discussed in the improvements section 12.2.

11.3 Technical Objectives Conclusion

The section discusses if the research's aim, as described in section 1.2, has been fulfilled and to what degree.

One of the core concepts of this AGV was the idea that the drive units should be modular. This modularity would allow for easy exchange of faulty units and the ability to build more powerful AGVs in the future by stacking more of these units on the AGV. As described in the aim, this modularity goal was achieved since the drive units can be removed from the AGV as discrete components. They are held in place by a set of seven bolts, as described in section 5.3.3, with the electrical connections (be they communication lines or power lines) being terminated in plugs.

It would have been ideal for the control system box and drive unit box to have been removable. However, this was not a research aim or specification and thus beyond this project's scope. An attempt was made to achieve this, as described in the section 5.7, via a set of "I/O plates" that contained a multitude of plugs. This attempt, unfortunately, had to be abandoned as a short to ground was detected, and after three weeks of fault-finding, the short could not be found. Thus, the AGV was rewired from scratch, and due to time constraints, the multitude of plugs could not be re-soldered and were replaced with point-to-point connections. This issue does not affect the drive units that still have their detachment plugs.

The physical separation of the drive units meant that fly-by-wire was the only control methodology that could be implemented; mechanical linkages between the drive units would have made them too complex to be "removable". Thus the fly-by-wire requirement of the AGV was successfully implemented as described in section 6.3.7.

The inclusion of a suspension system ensures that the AGV should be able to work in factories with suboptimal floors as required in the aims; the complete validation of the suspension system along with its modelling can be found in section 5.5.

The conclusion to the aims for the kinematic model, control system and the two units swerve drive system requirements are discussed as part of the "Objective Achievement" section, section 11.4.

The last requirement of the project aim was for the AGV to fulfil the role of an AGC (automated guided cart) that could be expanded with any number of additional devices, such as perhaps a robot arm. To fulfil this AGC requirement, a large unobstructed deck was designed into the top of the AGV to which additional components can be bolted, see section 5.10. Additionally, the main 220 VAC inverter has an unused capacity of 1.5 kW (39%) (see section 6.2.4.d) for additional devices.

11.4 Objective Achievement

The fulfilment state of the specifications listed section 1.2.1, is listed in table 11.1.

Table 11.1: Fulfilment of Specifications

Specification	**Fulfilment Status**
The gross mass of the AGV (including payload) will be between 500 *kg* and 1 000 *kg*	YES
The AGV will have 4 wheels (two powered and two unpowered)	YES
The powered units will have their own drive train and steering mechanism	YES
A basic suspension system is to be implemented	YES
The wheel/drive train combos must be modular and removable from the AGV body	YES
The maximum speed of the AGV should be 1.3 $^m/_s$	YES
The AGV should be able to climb an incline of 5°	YES

How the specifications listed in table 11.1 were fulfilled is detailed in the sections that follow.

11.4.1 Specification: Gross Mass Between 500kg and 1000kg

The final gross weight of the AGV was 600 *kg* as developed in section 5.3.4 and listed in table 5.3. The gross mass of the AGV includes the weight of any potential payload. The curb/ unloaded weight (bare bones without a payload) is 389 *kg*. The weight distribution of the AGV is thus 65% AGV and 35% payload; this is represented in figure 11.1.

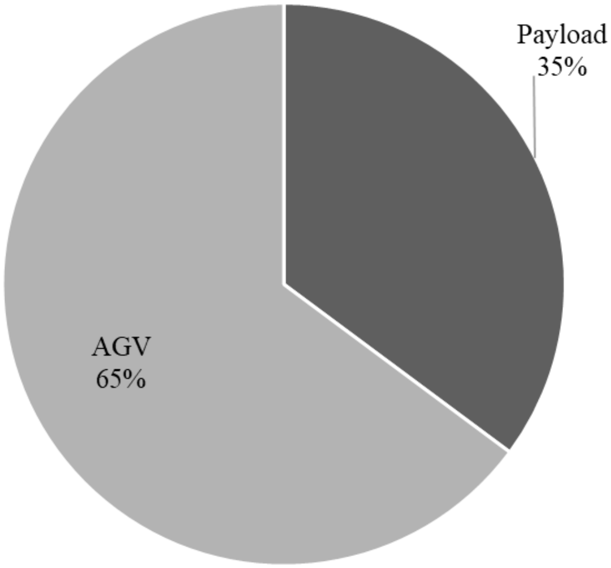


Figure 11.1: AGV Weight Allocation

Determination of the various motors sizes, gearboxes and bearings was done using the AGV design tool listed in chapter 4 and the calculations that can be found in chapter 5.

11.4.2 Specification: Four wheels, Two Powered & Two Unpowered

The AGV has two powered units called the "drive units" and two unpowered castor wheel based units called "castor units". The drive units were arranged diagonally to each other with reference to the AGV's frame as described in section 5.2. The arrangement is illustrated in the block diagram contained in figure 5.5 (which is repeated in figure 11.2 for ease of reference). The final mechanical design of this layout is contained in the section 5.6.

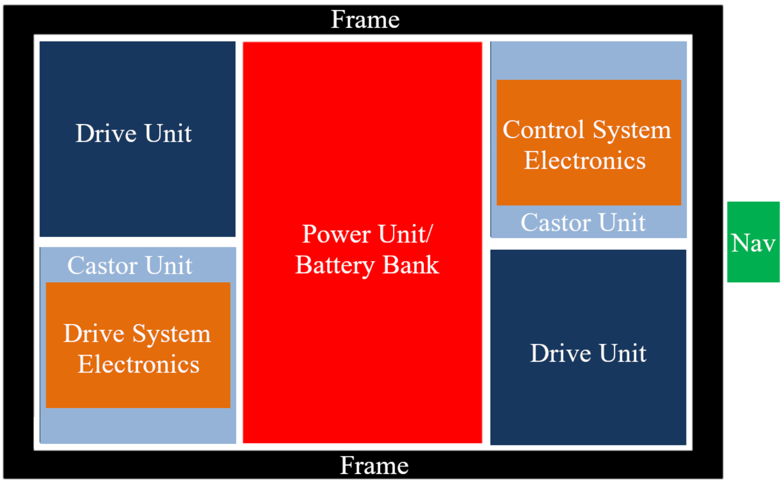


Figure 11.2: Final Block Diagram Layout of The AGV

The diagonal placement of the drive units was necessary to prevent the development of a moment about the centroid of the AGV during a strafing manoeuvre. This concept is described both linguistically and mathematically in section 5.2.

11.4.3 Specification: Basic Suspension System

The suspension system that was implemented on the AGV was the "Inline Suspension" system (see section 5.5.1.d). Since this AGV made use of two different wheel configurations, namely the "drive unit" and "castor unit" configuration, two different inline suspension systems were designed using a MATLAB model as described in section 5.5.5.

The drive unit had to use four springs due to design limitations, with each spring having a spring co-efficient of 13 462 $^N/_m$. With the castor unit, it was sufficient to use two larger springs, each having a spring co-efficient of 26 924 $^N/_m$. No discrete dampeners were implemented as the relatively high friction provided by the linear guides in this suspension system provided sufficient dampening.

Since this is a passive suspension system, it can only be optimised for one weight rating. The further the weight of the AGV deviates from the optimised weight value, the worse the suspension will behave. The suspension will become more bouncy with less weight and stiffer with more weight when compared to the optimised weight. The suspension system on the AGV was optimised for 600 kg (fully loaded), as the author felt that the AGV should be most stable when transporting its payload.

The worst-case scenario for the suspension system was also defined as a bump with a height of 10 mm maximum and width of 20 mm minimum (see section 5.5.4). Any bump or obstruction worse than this will cause the AGV to behave outside of desired bounds.

11.4.4 Specification: Modularity of Drive Units

As discussed in the previous section 11.3, the drive units are completely modular and thus can be removed from the AGV's body with the removal of seven bolts and six electrical plugs. Three of those plugs are for the traction motor (Siemens S1FL6 servo motors), two are for the steering motor (Festo CMMS-ST stepper motors), and the last is for the absolute encoder. The mechanical modularity is discussed in section 5.3.3, while the electrical connections and the plugs used are discussed in section 6.3.7.

11.4.5 Specification: AGV Max Speed = 1.3 $^m/_s$

The chosen gear ration, combined with the chosen motors wattage, allows the AGV to achieve a maximum velocity of 1.3 $^m/_s$ with a maximum acceleration of 0.65 $^m/_{s^2}$ (2s to go from 0 to 1.3 $^m/_s$). Although 1.3 $^m/_s$ is the maximum speed the AGV is capable of, it was found in practice to be rather quick. Thus, most manoeuvres are performed at 0.2 $^m/_s$ as shown in the tests done in chapter 10.

11.4.6 Specification: AGV Climb Incline of 5°

The AGV was designed to be able to climb an incline of 5°, when fully laden to 600 kg as discussed in section 5.3.4.a. Though the AGV can climb this incline, it was not designed to accelerate on such an incline but rather maintain a constant velocity. The AGV was designed to be able to perform the following operations:

- Constant velocity on a flat surface
- Acceleration on a flat surface
- Constant velocity up an incline

11.5 Chapter Conclusion

This chapter concludes the thesis. It analysed the results in the previous chapter, "Testing", to find possible issues, confirmed that the design brief has been met and suggested future improvements and spin-off fields of research.

12 Improvements and Future Research

12.1 To be Completed (Beyond Scope of Thesis)

This section will list tasks outside this thesis's scope but will eventually need to be implemented to complete the AGV fully. These are:

- Battery Eject System

  The battery eject system was completed mechanically and electrically but still requires programming and development of an interfacing protocol to communicate with a future battery loading system.

- Battery Management

  At a bare minimum, the AGV will need to know the charge of its batteries to trigger a battery change or charging request. The easiest way to achieve this is via reading the battery status data from the main inverter using RS232. The wiring for this system has been completed; however, code implementation remains unfinished; this could be done via NUTS (Network UPS tool).

- SICK Nav 350 Navigation

  To enable the automatic operation of the AGV, the IPC will need to grab position data from the NAV 350 via Profinet telegrams. A point cloud will need to be generated from the environment to enable autonomous operation, which will require time-of-flight from the NAV 350 LiDAR scanner.

- SICK s300 Remote Mini

  The two safety scanners responsible for the safety curtain around the AGV are not currently operational and need to be programmed.

- ROS integration

  Suppose the AGV is to be autonomous instead of simply automatic (virtual line following). A ROS system needs to be developed on the Debian Linux VM on the IPC.

12.2 Improvements

The following improvements to the AGV:

- Armour Safety Sensors

  Currently, the SICK s300 remote mini safety scanners are exposed on the corners of the AGV; see section 5.8.2. Although it is unlikely the AGV would damage these sensors since they would shut down the AGV before a collision. There is a high probability that they could be damaged by third parties when they operate in the vicinity of the shutdown AGV. To alleviate this issue, they should be "armoured" by enclosing them in a cage or sheet metal pillbox, with only the bare minimum exposed to possible damage.

- Main Battery Unit Electrical Interface Improvement

  The current electrical interface on the main battery unit is less than idea, shown in figure 12.1 and described in section 5.11.3, since it has relativity stiff springs. These stiff springs mean the pinion system motor has to work harder to compress them, requiring more current, which means larger wires, larger relays and an auxiliary battery unit that can produce more current.

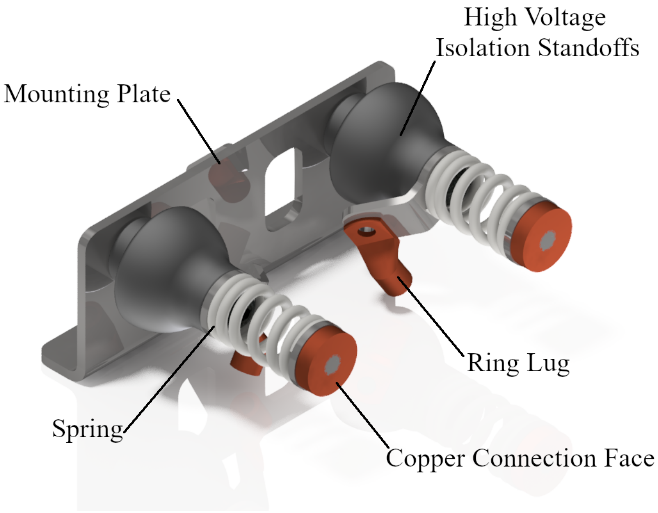


Figure 12.1: Main Battery Unit Electrical Interface

The reason these springs are so stiff is a result of the spring wire diameter, which as the sole current carrier between the battery unit and the inverter, had to be able to carry a maximum of 100A at 48 VDC. If these springs were made thinner (easier for the pinion system to compress), the high current drawn through them would cause them to heat up, damaging the spring temper and possibly melting or igniting surrounding insulators and components.

It may be possible to decrease the spring wire diameter by providing an alternative path for the current to flow. It is the author's idea that a braided mesh (similar to those used for flexible busbars, see figure 12.2) could be placed inside of the spring connecting the copper connection face (see figure 12.1) to the ring lug connection plate.

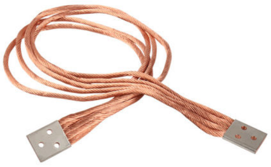


Figure 12.2: Flexible Copper Braided Busbar[1]

- SSR Cut-Out for AGV Battery Bank

[1] Image taken from public domain

The inverter is shut down via a digital I/O from the PLC (PLC powered by DC UPS) to exchange the battery unit. It would be ideal if, in addition to the inverter shutdown, the 48 VDC connection between the inverter and the battery unit was physically broken via a high amperage contact or Solid State Relay (SSR). To this end, the positive 48 VDC cable between the inverter and battery bank passes through an empty enclosure to house this contact. This housing also has its' rear exposed to the bare metal of the AGV's aluminium skin to provide a heat sink for an SSR.

12.3 Additional Research

Additional research projects that can be done as undergraduate or master's degrees:

- Direct Drive Wheels

  Currently, the AGV's traction and steering system share a common axis. This common axis presents issues (as seen in the testing section) where the two systems will influence each other even with an algorithm to compensate for this. The easiest way to eliminate this issue is to attach the tractive motor directly to the drive wheel, skipping the common axis. This design will require research into compact high ration/high torque gearing, and high current slip rings to power the motor. The gearbox issue could be solved using a cycloidal drive which has recently (circa 2021) become more accessible on the market thanks to its use in modern robotic arms such as the Universal Robotics UR10e COBOT and the Chinese Dobot CR series.

- Advanced Battery Management System

  There was previously an undergraduate project on this topic that failed to meet requirements (see appendix J); as such, it can be reissued. Although there is a primitive BMS system on the inverter that can be used as a stopgap to determine the SoC of the main battery bank, it is not ideal. It is not ideal since it only uses a voltage measurement to estimate the SoC. A more advanced BMS system will be needed to determine the actual SoC, such as Columb counting (the charge into and out of the battery). While simultaneously checking the max current draw and battery temperature, all of which affect a battery's capacity. In addition to battery management, this system can also perform advanced battery balancing (in place of the cheap Bangood battery balancer) and track and trace the factory's battery unit as it is a separate system from the AGV itself. A Siemens IoT2040 was pre-emptively installed in the battery bank for this purpose.

- Inertial Feedback

  Inertial feedback uses a set of accelerometers, gyroscopes and a magnetometer (compass) to determine the motion of a mobile machine. The inclusion of an inertial navigation system to supplement the LiDAR navigation and wheel feedback of the AGV would improve its accuracy. It could also help reduce wheel slip as it would be easier to detect through an inertial system. The inclusion of this system is relatively cheap as an Adafruit LLC 2472 BNO055 is around R500.00 ($31.65). The main issues with these systems are the extensive filters and data processing necessary to make them useful and reliable, thus becoming a research project.

- Floor Camera Feedback

  Like the inertial navigation unit, this could be used to supplement the navigation system on the AGV. It works by using a monochrome camera to take pictures of either the ceiling or floor and comparing landmarks it finds in these images to determine motion. This system is the primary operating principle of an optical mouse and has already been implemented by the Xiaomi Mi 1C robot vacuum cleaner.

- Docking Station & Solar

  Currently, the AGV can be "charged" in two ways. The first is by exchanging the entire battery bank, and the second is via a 220 VAC manual plug. It would be ideal if a docking station idea could also be implemented to allow the AGV to perform opportunistic charging when docked to load and offload its payload.

  A full 300W solar panel could be embedded in the payload deck; this would allow the AGV to trickle-charge its batteries when waiting or moving without a payload outside. This solar panel will not provide enough power to create a self-sustaining system, and the AGV will eventually need to dock to charge or exchange its batteries, but it could provide a couple of additional minutes of runtime, which, when spread over hundreds of AGVs could make economic sense. It would also make maintenance more manageable as the AGV can trickle charge when not in use to prevent self-discharge of the batteries. This system is relatively easy to implement as a solar charge controller is included in the inverter.

- Robot Operating System

  A ROS system will need to be implemented for full autonomous navigation and a SLAM system and path optimisation algorithm. This ROS system can all be implemented on the Debian Linux VM on the IPC.

- Fleet Management System

  If multiple AGVs operate together, a fleet or factory management system will need to be developed. This system will optimise the AGV movements within the factory serve to notify AGVs of what payload they must collect and where they need to deposit it.

Bibliography

[1] A. Backer and I. Mackenzie, "2008 FIRST Robotics Conference," in *Omnidirectional Drive Systems Kinematucs and Control*, (Atlanta), 2008.

[2] E. Guizzo, "Three engineers, hundreds of robots, one warehouse," *IEEE Spectrum*, vol. 45, no. 7, pp. 26–34, 2008.

[3] Savant, "Types of Automatic Guided Vehicle," 2015.

[4] T. P. Ferreira and I. A. Gorlach, "Development of an affordable Automated Guided Cart for material handling tasks," *International Journal of Circuits and Electronics*, vol. 1, pp. 145–150, 2016.

[5] R. Bostelman, J. Falco, and T. H. Hong, "Performance Measurements of Motion Capture Systems used for AGV and Robot Arm Evaluation," *Autonomous Industrial Vehicles: From the Laboratory to the Factory Floor*, no. May, pp. 1–7, 2016.

[6] T. A. Tamba, B. Hong, and K. S. Hong, "A path following control of an unmanned autonomous forklift," *International Journal of Control, Automation and Systems*, vol. 7, no. 1, pp. 113–122, 2009.

[7] B. D. Dziwis and S. O. Company, "Automated / Self Guided Vehicles (AGV / SGV) and System Design Considerations," *ACADEMIA*, pp. 1–3, 2005.

[8] G. D. Scott, *Design and development of an automatic guided vehicle.pdf.* Masters thesis, Nelson Mandela Metropolitain University, 2015.

[9] D. Bourne, H. Choset, H. Hu, G. Kantor, C. Niessl, Z. Rubinstein, R. Simmons, and S. Smith, "Mobile manufacturing of large structures," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 1565–1572, 2015.

[10] G. K. Kraetzschmar, N. Hochgeschwender, W. Nowak, F. Hegger, S. Schneider,

R. Dwiputra, J. Berghofer, and R. Bischoff, "RoboCup@Work: Competing for the factory of the future," *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)*, vol. 8992, pp. 171–182, 2015.

[11] G. Ullrich, *Automated Guided Vehicle Systems.* Germany: Springer-Verlag Berlin Heidelberg, 1 ed., 2015.

[12] L. Qiu, W. J. Hsu, S. Y. Huang, and H. Wang, "Scheduling and routing algorithms for AGVs: A survey," *International Journal of Production Research*, vol. 40, no. 3, pp. 745–760, 2002.

[13] BMBF, "The new High-Tech Strategy Innovations for Germany," *Federal Ministry of Education and Research (BMBF)*, pp. 1–58, 2014.

[14] Riana de Lange, "R256.7 billion; the vehicle industry's massive GDP contribution," mar 2017.

[15] Euronews, "South Africa wants to lead the Fourth Industrial Revolution," 2019.

[16] I. P. Vlachos, R. M. Pascazzi, G. Zobolas, P. Repoussis, and M. Giannakis, "Lean manufacturing systems in the area of Industry 4.0: a lean automation plan of AGVs/IoT integration," *Production Planning and Control*, vol. 0, no. 0, pp. 1–14, 2021.

[17] D. B. le Roux, "Automation and employment: The case of South Africa," 2018.

[18] M. Chikwete, "Automation: Is South Africa Ready?," 2020.

[19] B. I. Chigbu and F. H. Nekhwevha, "The extent of job automation in the automobile sector in South Africa," *Economic and Industrial Democracy*, 2020.

[20] L. Kromann, J. R. Skaksen, and A. Sørensen, "Automation, labor productivity and employment-a cross country comparison," *Aim-Projektet.Dk*, no. March 2021, pp. 1–16, 2007.

[21] Kollmorgen, "Automated Guided Vehicles (AGVs)," 2016.

[22] S. Bøgh, C. Schou, T. Rühr, Y. Kogan, A. Dömel, M. Brucker, C. Eberst, R. Tornese, C. Sprunk, G. D. Tipaldi, and T. Hennessy, "Integration and assessment of multiple mobile manipulators in a real-world industrial production facility," *Proceedings for the Joint Conference of ISR 2014 - 45th International Symposium on Robotics and Robotik 2014 - 8th German Conference on Robotics, ISR/ROBOTIK 2014*, no. 260026, pp. 305–312, 2014.

[23] H. Qian, T. L. Lam, W. Li, C. Xia, and Y. Xu, “System and design of an omni-directional vehicle,” *2008 IEEE International Conference on Robotics and Biomimetics, ROBIO 2008*, vol. 2008-Janua, pp. 389–394, 2008.

[24] J. J. Zeng, R. Q. Yang, W. J. Zhang, X. H. Weng, and Q. Jun, “Research on semi-automatic bomb fetching for an EOD robot,” *International Journal of Advanced Robotic Systems*, vol. 4, no. 2, pp. 247–252, 2007.

[25] J. Qian, B. Zi, D. Wang, Y. Ma, and D. Zhang, “The design and development of an Omni-Directional mobile robot oriented to an intelligent manufacturing system,” *Sensors (Switzerland)*, vol. 17, no. 9, 2017.

[26] J. A. Batlle and A. Barjau, “Holonomy in mobile robots,” *Robotics and Autonomous Systems*, vol. 57, no. 4, pp. 433–440, 2009.

[27] J. Efendi, M. Salih, M. Rizon, and S. Yaacob, “Designing Omni-Directional Mobile Robot with Mecanum Wheel Abdul Hamid Adom and 2 Mohd Rozailan Mamat School of Mechatronics Engineering , Kolej Universiti Kejuruteraan Utara Malaysia Terengganu Advanced Technical Institute , Jalan Panchor , Teluk Kalong,” *American Journal of Applied Sciences*, vol. 3, no. 5, pp. 1831–1835, 2006.

[28] M. Komori, K. Matsuda, T. Terakawa, F. Takeoka, H. Nishihara, and H. Ohashi, “Active omni wheel capable of active motion in arbitrary direction and omnidirectional vehicle,” *Journal of Advanced Mechanical Design, Systems and Manufacturing*, vol. 10, no. 6, pp. 1–20, 2016.

[29] C. W. Wu, K. S. Huang, and C. K. Hwang, “A novel spherical wheel driven by chains with guiding wheels,” *Proceedings of the 2009 International Conference on Machine Learning and Cybernetics*, vol. 6, no. July, pp. 3242–3245, 2009.

[30] E. H. Binugroho, A. Setiawan, Y. Sadewa, P. H. Amrulloh, K. Paramasastra, and R. W. Sudibyo, “Position and Orientation Control of Three Wheels Swerve Drive Mobile Robot Platform,” *International Electronics Symposium 2021: Wireless Technologies and Intelligent Systems for Better Human Lives, IES 2021 - Proceedings*, pp. 669–674, 2021.

[31] P. Xu, “Mechatronics Design of a Mecanum Wheeled Mobilr Robot,” in *Cutting Edge Robotics* (K. Vedran, A. Lazinica, and M. Merdan, eds.), ch. Modelling, pp. 61–74, Mammendorf: pIV pro literatur Verlag, 1 ed., 2005.

[32] N. Tlale and M. de Villiers, “Kinematics and Dynamics Modelling of a Mecanum Wheeled Mobile Platform,” *Mechatronics and Machine Vision in Practice, 2008.*

M2VIP 2008. 15th International Conference on, no. 3, pp. 657–662, 2008.

[33] A. B. S. Macfarlane, *Modular Electric Automatic Guided Vehicle Suspension-Drive Unit.* PhD thesis, 2016.

[34] O. Diegel, A. Badve, G. Bright, J. Potgieter, and S. Tlale, "Improved Mecanum Wheel Design for Omni-directional Robots," *Shinji Kamiuchi and Shoichi Maeyama, "A Novel Human Interface of an Omni-directional Wheelchair", Int. Workshop on Robot and Human Interactive Communication, 2004, pp. 101-106.*, no. November, pp. 27–29, 2002.

[35] Pitsco Education, "TETRIX MAX Omni Wheel Packs," 2022.

[36] R. Rojas and A. Förster, "Holonomic control of a robot with an omnidirectional drive," *KI-Künstliche Intelligenz*, p. 7, 2006.

[37] I. Mackenzie, "Omnidirectional drive system," in *2006 FIRST Robotics Conference*, 2006.

[38] C. H. Chiu and W. R. Tsai, "Design and implementation of an omnidirectional spherical mobile platform," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 3, pp. 1619–1628, 2015.

[39] R. Holmberg and J. C. Slater, "Powered caster wheel module for use on omnidirectional drive systems," 2002.

[40] M. Wada, A. Takagi, and S. Mori, "Caster drive mechanisms for holonomic and omnidirectional mobile platforms with no over constraint," *IEEE International Conference on Robotics and Automation*, vol. 2, no. April, pp. 1531–1538, 2000.

[41] Darshit Patel, "Automobile Transmission," 2017.

[42] A. Siravuru, S. V. Shah, and K. M. Krishna, "An optimal wheel-torque control on a compliant modular robot for wheel-slip minimization," *Robotica*, vol. 35, no. 2, pp. 463–482, 2017.

[43] Siemens AG, *TIA Safety.* Germany: Siemens AG, 15 ed., 2019.

[44] Y. Wang, X. Lei, G. Zhang, S. Li, H. Qian, and Y. Xu, "Design of dual-spring shock absorption system for outdoor AGV," *2017 IEEE International Conference on Information and Automation, ICIA 2017*, no. July, pp. 159–164, 2017.

[45] Janek, "SANS10400," 2013.

[46] R. A. Serway and J. W. Jewett, *Physics for Scientist and Engineers with Modern*

Physics 9th Edition. Boston: Brooks/Cole Cengage Learning, 9th ed., 2014.

[47] Y. A. Çengel, J. M. Cimbala, and R. H. Turner, *Fundamentals of Thermal-Fluid Sciences.* United States of America: McGraw-Hill Higher Education, 5th ed., 2017.

[48] F. E. Giessecke, A. Mitchell, H. C. Spencer, I. L. Hill, J. T. Dygdon, J. E. Novak, and S. Lockhart, *Modern Graphics Communication.* United States of America: Pearson, 4th ed., 2010.

[49] G. Chikosi, "Autonomous Guided Vehicle For Agricultural Applications," tech. rep., 2014.

[50] M. Wada and S. Mori, "Holonomic and omnidirectional vehicle with conventional tires," *Robotics and Automation, 1996. Proceedings. ...*, no. April, pp. 3671–3676, 1996.

[51] T. D. Gillespie, *Fundamentals of vehicle dynamics.* Warrendale: USA:Society of Automotive Engineers Inc., 1992.

[52] Jost, "Ball bearing turntabls and slewing rings: KLK L und N," 2019.

[53] Castor & Ladder, "Castor and Ladder," 2020.

[54] R. G. Budynas and J. K. Nisbett, *Shigley's Mechanical Engineering Design.* United States of America: McGraw-Hill Education, 10th ed., 2015.

[55] D. Collins, "Complete Guide to Car Suspension," 2018.

[56] Hariharan, "Suspension systems and components," 2012.

[57] J. C. Dixion, *Tires, Suspension and Handling.* United States of America: ARNOLD, second ed., 2001.

[58] Flinn Scientific, "Spring Combinations," *Flinn Scientific: Physical Science Fax*, 2016.

[59] Steel Pipes & Fittings, "Steel Pipes & Fittings: Mechanical & Structural Tubes," 2020.

[60] Y. Q. Sun, "Mitigating Train Derailments Due to Sharp Curve and Overspeed," *Frontiers in Mechanical Engineering*, vol. 4, no. August, pp. 1–12, 2018.

[61] W. L. Webb, *Railroad Construction: Theory and Practice.* New York: John Wiley & Sons, 1st ed., 1900.

[62] S. Tanaka, N. Fukujyu, and S. Kaneda, "Configurations of Micro EV Quick-Charging System with EDLC Storage," *Proceedings - International Computer Software and Applications Conference*, vol. 2, pp. 294–299, 2016.

[63] A. B. Macfarlane, T. van Niekerk, and U. Becker, "Low Cost PLC Uninterrupted Power Supply for use on AGVs with a Removable Battery Banks," in *2020 International SAUPEC/RobMech/PRASA Conference*, pp. 1–6, IEEE, jan 2020.

[64] Ned Mohan; Tore M. Undelande; William P. Robbins, *Power Electronics: Converters, Applications and Design*, vol. 33. John Wiley & Sons, 3rd ed., 2002.

[65] Siemens, "SINAMICS V90 PROFINET, SIMOTICS S-1FL6 Operating Instructions," tech. rep., Siemens, Germany, 2017.

[66] D. Pavlov, *Lead-Acid Batteries: Science and Technology.* Sofia: Elsevier B.V., 2nd ed., 2017.

[67] Intel, "Intel Core i5-6442EQ Processor," 2021.

[68] Siemens, "Industrial PC SIMATIC IPC427E," tech. rep., Germany, 2016.

[69] Z. Schoenborn, "Board Design Guidelines for PCI Express Architecture," tech. rep., PCI-SIG, 2004.

[70] Siemens Sitrain, *Industrial Communication PROFINET with Industrial Ethernet in TIA Portal.* Germany: Siemens AG, v15.01.00 ed., 2018.

[71] SICK, "SICK Sensor Intelligence," 2010.

[72] M. Sokolov, O. Bulichev, and I. Afanasyev, "Analysis of ROS-based visual and lidar odometry for a teleoperated crawler-type robot in indoor environment," *ICINCO 2017 - Proceedings of the 14th International Conference on Informatics in Control, Automation and Robotics*, vol. 2, no. July, pp. 316–321, 2017.

[73] Siemens Sitrain, *Safety-related programming with STEP 7 Safety in TIA Portal.* Germany: Siemens AG, v15.01.00 ed., 2019.

[74] Siemens AG, "Digital output module ET 200SP Digital output module," tech. rep., Germany, 2013.

[75] Siemens, "SINAMICS v90, SIMOTICS S-1 FL6 Operating Instructions," tech. rep., Siemens AG, Germany, 2017.

[76] Festo, "CMMS-ST-C8-7-G2 STO safety function (Safe Torque Off)," tech. rep., Festo AG, Germany, 2016.

[77] IEEE, "IEEE Registration Authority," 2021.

[78] Siemens AG, *SINAMICS V90, SIMOTICS S-1FL6 PROFINET (PN) interface Getting Started*. Germany: Siemens AG, 3 ed., 2017.

[79] G. Yang, Y. Li, T. M. Lim, and C. W. Lim, "Decoupled Powered Caster Wheel for omnidirectional mobile platforms," *Proceedings of the 2014 9th IEEE Conference on Industrial Electronics and Applications, ICIEA 2014*, pp. 954–959, 2014.

[80] S. B. Niku, *Introduction to Robotics; Analysis, Control, Applications*. United States of America: John Wiley & Sons, second ed., 2011.

[81] A. Dresden, "The fourteenth western meeting of the american mathematical society," *Bulletin of the American Mathematical Society*, vol. 26, no. 9, pp. 385–396, 1920.

[82] Wolfram, "WolframAlpha," 2021.

[83] Y. C. Lee, D. V. Lee, J. H. Chung, and S. A. Velinsky, "Control of a redundant, reconfigurable ball wheel drive mechanism for an omnidirectional mobile platform," *Robotica*, vol. 25, no. 4, pp. 385–395, 2007.

[84] Admin@studyelectrical.com, "Types of Stepper Motor," 2019.

[85] Gates Energy Products, *Rechargeable Batteries: Applications Handbook*. Stonham: Butterworth-Heinemann, 1st ed., 1991.

[86] Mantech, "Mantech Electronics," 2019.

[87] Y. Li, T. Zielinska, M. Ang, and W. Lin, "Vehicle Dynamics of Redundant Mobile Robots with Powered Caster Wheels," *Romansy 16*, vol. 487, pp. 221–228, 2006.

[88] L. Chang, "Comparison of AC drives for electric vehicles - a report on experts' opinion survey," *IEEE Aerospace and Electronic Systems Magazine*, vol. 9, no. 8, pp. 7–11, 1994.

[89] Z. Xian and G. Wang, "Optimal dispatch of electric vehicle batteries between battery swapping stations and charging stations," *IEEE Power and Energy Society General Meeting*, vol. 2016-Novem, pp. 1–5, 2016.

[90] R. Baran and L. F. L. Legey, "The introduction of electric vehicles in Brazil: Impacts on oil and electricity consumption," *Technological Forecasting and Social Change*, vol. 80, no. 5, pp. 907–917, 2013.

[91] A. Fitzgerald, *Electric Machinery, Sixth Edition*. McGraw-Hill Higher Education,

6th ed., 2003.

[92] V. S. Bagotsky, A. M. Skundin, and Y. M. Volfkovich, *Electrochemical Power Sources.* United States of America: John Wiley & Sons, 1st ed., 2015.

[93] J. Larminie and J. Lowry, *Electric Vehicle Technology Explained: Second Edition.* Chichester: John Wiley & Sons, 2nd ed., 2012.

[94] A. Rossouw, "The accelerated life cycle testing and modelling of Li-ion cells used in electric vehicle applications," Tech. Rep. January, Nelson Mandela Metropolitain University, 2012.

[95] M. Wada, Y. Tominaga, and S. Mori, "Omnidirectional holonomic mobile robot using nonholonomic wheels," *IEEE International Conference on Intelligent Robots and Systems*, vol. 3, pp. 446–453, 1995.

[96] A. C. R. Teixeira and J. R. Sodré, "Impacts of replacement of engine powered vehicles by electric vehicles on energy consumption and CO 2 emissions," *Transportation Research Part D: Transport and Environment*, vol. 59, pp. 375–384, 2018.

[97] K. Doubleday, A. Meintz, and T. Markel, "An opportunistic wireless charging system design for an on-demand shuttle service," *2016 IEEE Transportation Electrification Conference and Expo, ITEC 2016*, pp. 1–6, 2016.

[98] A. Kilian, "German forklift manufacturer establishes local presence," 2016.

[99] Y. Hori, "Novel EV society based on motor/ capacitor/ wireless - Application of electric motor, supercapacitors, and wireless power transfer to enhance operation of future vehicles," *2012 IEEE MTT-S International Microwave Workshop Series on Innovative Wireless Power Transmission: Technologies, Systems, and Applications, IMWS-IWPT 2012 - Proceedings*, pp. 3–8, 2012.

[100] T. J. E. Miller, *Brushless Permanent-Magnet and Reluctance Motor Drives (Monographs in Electrical and Electronic Engineering 21).* No. 21, Clarendon Press, 1989.

[101] W. Rippel, "Tesla: Induction Versus DC Brushless Motors," 2007.

[102] RS Components, "RS-Online," 2019.

[103] Siemens, "Siemens Industrial Mall," 2019.

[104] K. T. Gradin, S. Poulikidou, A. Björklund, and C. Luttropp, "Scrutinising the

electric vehicle material backpack," *Journal of Cleaner Production*, vol. 172, pp. 1699–1710, 2018.

[105] P. Gallo, "Charging electric vehicles using opportunistic stopovers," *Conference Proceedings - 2017 17th IEEE International Conference on Environment and Electrical Engineering and 2017 1st IEEE Industrial and Commercial Power Systems Europe, EEEIC / I and CPS Europe 2017*, pp. 1–5, 2017.

[106] G. Albright, J. Edie, and S. Al-Hallaj, "A Comparison of Lead Acid to Lithium-ion in Stationary Storage Applications." 2012.

[107] H. Shareef, S. N. Khalid, M. W. Mustafa, and A. Mohamed, "Modeling and simulation of overvoltage surges in low voltage systems," *PECon 2008 - 2008 IEEE 2nd International Power and Energy Conference*, no. PECon 08, pp. 357–361, 2008.

[108] S. M. Church, "Energy management system for the diagnosis and control of an automatic guided vehicle," tech. rep., Nelson Mandela University, Port Elizabeth, 2016.

[109] M. Rouse and I. Wigmore, "Internet of Things (IoT)," 2016.

[110] Cisco, "Internet of Things (IoT)," 2016.

[111] S. Anuphappharadorn, S. Sukchai, C. Sirisamphanwong, and N. Ketjoy, "Comparison the economic analysis of the battery between lithium-ion and lead-acid in PV stand-alone application," *Energy Procedia*, vol. 56, no. C, pp. 352–358, 2014.

[112] D. Berndt, *Maintenance Free Batteries.* Great Britain: John Wiley & Sons, 1st ed., 1993.

[113] Institute of Electrical and Electronics Engineers, *IEEE Recommended Practice for Emergency and Standby Power Systems for Industrial and Commercial Applications*, vol. 1995. 1996.

[114] Engineers Edge LLC, "Battery Appication & Technology," 2019.

[115] Y. E. Abu Eldahab, N. H. Saad, and A. Zekry, "Enhancing the design of battery charging controllers for photovoltaic systems," *Renewable and Sustainable Energy Reviews*, vol. 58, no. September 2018, pp. 646–655, 2016.

[116] ASTPM, "Standards for Hot Strip for Tube and Pipe," 2020.

[117] SHENZHEN RITAR POWER CO LTD, "RT1270B Datasheet," 2014.

[118] A. Kloeblen, “Construction and integration of a battery pack and management system into a solar car,” Tech. Rep. June, Nelson Mandela University, Port Elizabeth, 2013.

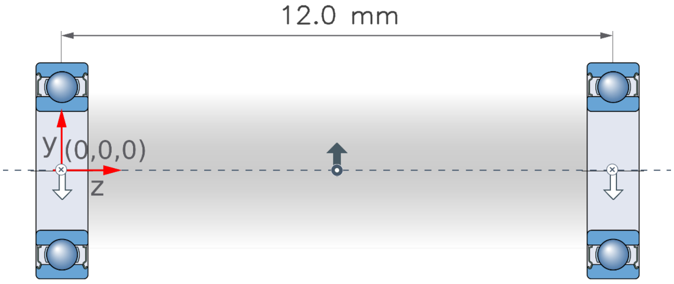
[119] C. A. Vincent and B. Scrosati, *Modern Batteries: An Introduction to Electrochemical Power Sources.* London: John Wiley & Sons, 2nd ed., 1997.

A Appendix - Belt System 1 Idler Pulley Bearing SKF Calculations

SKF Bearing Select v1.2-45

Belt System 1 Idler Bearings

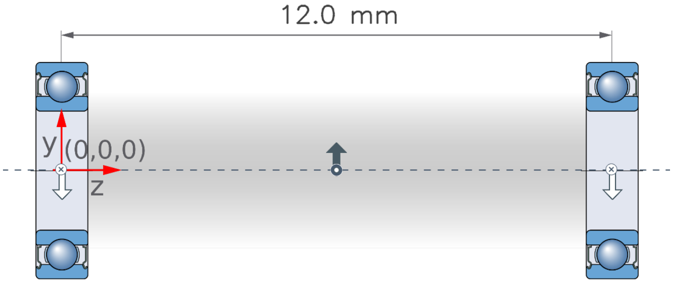
Calculations for bearings used in the idler pulley found in the upper belt system of the drive unit



Alex Macfarlane
NMU

August 12, 2020

1. Abstract



Deep groove ball bearing

Locating

Deep groove ball bearing

Non-Locating

🟦 SKF Explorer ▶ Popular item

	Designation	Life model		Grease	Static safety factor	Frictional moment	Power loss
		Basic	SKF life	Catalogue grease life		Total	
		L_{10h}	L_{10mh}	L_{10}	S_0	M	P_{loss}
		h				*Nmm*	*W*
Left	61900-2Z	> 2x10^5	> 2x10^5	28900	139	0.21	0.2
Right	61900-2Z	> 2x10^5	> 2x10^5	28900	139	0.21	0.2

** SKF rating life (L_{10mh}) for steel-steel bearings; GBLM load based life (L_{10GMh}) for hybrid bearings*

Left bearing

! Low viscosity ratio k, reduced asperity contact. It is recommended to select a higher viscosity lubricant or improve cooling. It is not appropriate to look at basic rating life only. Instead use SKF rating life method. Recommended to use anti-wear (AW) or extreme pressure (EP) additives to reduce wear More info

! For rating life results above 100000 hours, other failure modes than those included in the current rating life models will dominate and limit the life of the bearing. More info

! The grease life / relubrication interval is reduced depending on the contamination level. Higher cleanliness will improve the duration.

Right bearing

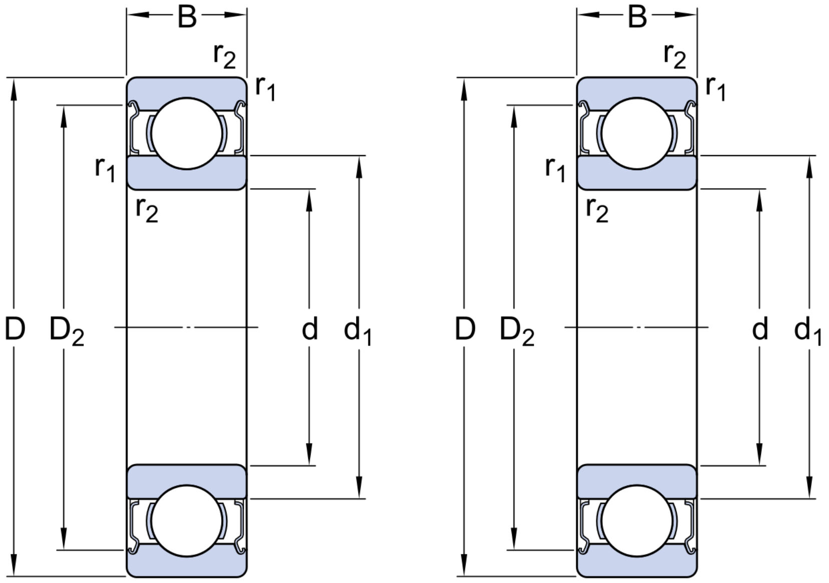
! Low viscosity ratio k, reduced asperity contact. It is recommended to select a higher viscosity lubricant or improve cooling. It is not appropriate to look at basic rating life only. Instead use SKF rating life method. Recommended to use anti-wear (AW) or extreme pressure (EP) additives to reduce wear More info

! The grease life / relubrication interval is reduced depending on the contamination level. Higher cleanliness will improve the duration.

! For rating life results above 100000 hours, other failure modes than those included in the current rating life models will dominate and limit the life of the bearing. More info

2. Input

2.1. Bearing data



	Designation	Bearing type	Principal dimensions			Basic load ratings		Fatigue load limit
						Dynamic	Static	
			d	D	B	C	C_0	P_u
			mm			*kN*		
Left	61900-2Z	Deep groove ball bearing	10	22	6	2.7	1.27	0.054
Right	61900-2Z	Deep groove ball bearing	10	22	6	2.7	1.27	0.054

	Designation	Speed ratings		Clearance class
		Reference	Limiting	
		n_{ref}	n_{lim}	
		r/min		
Left	61900-2Z	70000	36000	Normal
Right	61900-2Z	70000	36000	Normal

2.2. Loads & Speed

Locating bearing	None (axial load ignored)	
Bearing distance	12.0	*mm*
Shaft orientation	Vertical	
Rotating ring	Inner ring rotation	

	Load	**Coordinate system**	**Coordinates**			**Forces**			**Speed**	**Case weight**
			x|r	**y|θ**	**Z**	**Fx|Fr**	**Fy|Fθ**	**Fz**		
			mm	*mm|deg*	*mm*	*kN*	*kN*	*kN*	*r/min*	
LC1	F1	Cartesian	0.0	0.0	6.0	-0.018	0.003	0.0	8952.12	1

2.3. Temperature

Load Cases	**Left**		**Right**	
	Inner ring	Outer ring	Inner ring	Outer ring
	°C			
LC1	70	65	70	65

- *Maximum temperature is used for calculating the actual viscosity, kappa, a_{SKF} and SKF rating life.*
- *Mean temperature is used for calculating bearing friction and power loss.*

2.4. Lubrication

	Designation	Lubricant	Effective EP additives
		Name	
Left	61900-2Z	LT10	False
Right	61900-2Z	LT10	False

	Designation	Contamination
		Method
Left	61900-2Z	Detailed guidelines
Right	61900-2Z	Detailed guidelines

2.5. Fits and tolerances

	Designation	Requirements	Tolerance Class		Calculated interference	Include Smoothing
		Guidance	Housing	Shaft		
Left	61900-2Z	False	H7	k6	True	True
Right	61900-2Z	False	H7	k6	True	True

3. Results

3.1. Loads & static safety

	Designation	Load ratio	Static safety factor	Equivalent dynamic load	Equivalent static load
		C/P	S_0	P	P_0
				kN	
Left	61900-2Z	296.5	139	0.01	0.00911
Right	61900-2Z	296.5	139	0.01	0.00911

3.2. Bearing minimum load

	Designation	Reaction forces		Minimum load	
		Radial	Axial		met?
		F_r	F_a	$F_{r\ m}$	
		kN			
Left	61900-2Z	0.00911	0	0.00705	yes
Right	61900-2Z	0.00911	0	0.00705	yes

3.3. Adjusted reference speed

	Designation	Adjusted reference speed	Adjustment factors	
			For bearing load P	For oil viscosity
		n_{ar}	f_p	f_v
		r/min		
Left	**61900-2Z**	151000	1.0	2.16
Right	**61900-2Z**	151000	1.0	2.16

3.4. Lubrication conditions

	Designation	Operating viscosity			Viscosity ratio
		Actual	Rated	Rated @ 40 °C	
		ν	ν_1	ν_{ref}	K
		mm²/s			
Left	**61900-2Z**	5.7	10.0	28.6	0.56
Right	**61900-2Z**	5.7	10.0	28.6	0.56

This report was produced with SKF Bearing Select software. Please note the limited warranty, shown at the bottom of the last page of this report

3.5. Grease life and relubrication interval

	Designation	Catalogue grease life L_{10} *h*	Speed factor Speed x mean diameter nd_m *mm/min*
Left	61900-2Z	28900	143000
Right	61900-2Z	28900	143000

Left bearing

! The grease life / relubrication interval is reduced depending on the contamination level. Higher cleanliness will improve the duration.

Right bearing

! The grease life / relubrication interval is reduced depending on the contamination level. Higher cleanliness will improve the duration.

3.6. Bearing rating life

	Designation	Bearing rating life Basic L_{10h}	 SKF L_{10mh}	SKF life modification factor a_{skf}	Contaminati on factor η_c
		h			
Left	61900-2Z	> 2x10^5	> 2x10^5	3.86	0.07
Right	61900-2Z	> 2x10^5	> 2x10^5	3.86	0.07

** SKF rating life (L_{10mh}) for steel-steel bearings; GBLM load based life (L_{10GMh}) for hybrid bearings*

Left bearing

! Low viscosity ratio k, reduced asperity contact. It is recommended to select a higher viscosity lubricant or improve cooling. It is not appropriate to look at basic rating life only. Instead use SKF rating life method. Recommended to use anti-wear (AW) or extreme pressure (EP) additives to reduce wear More info

! For rating life results above 100000 hours, other failure modes than those included in the current rating life models will dominate and limit the life of the bearing. More info

Right bearing

! Low viscosity ratio k, reduced asperity contact. It is recommended to select a higher viscosity lubricant or improve cooling. It is not appropriate to look at basic rating life only. Instead use SKF rating life method. Recommended to use anti-wear (AW) or extreme pressure (EP) additives to reduce wear More info

! For rating life results above 100000 hours, other failure modes than those included in the current rating life models will dominate and limit the life of the bearing. More info

3.7. Bearing friction & power loss

	Designation	Frictional moment		Friction Sources				Power loss
		Total	At start 20-30°C and zero speed	Rolling	Sliding	Seals	Drag loss	
		M	M_{start}	M_{rr}	M_{sl}	M_{seal}	M_{drag}	P_{loss}
		Nmm						*W*
Left	**61900-2Z**	0.21	0.01	0.21	0.0	0	0	0.2
Right	**61900-2Z**	0.21	0.01	0.21	0.0	0	0	0.2

3.8. Bearing frequencies

	Designation	Rotational frequencies			
		Inner ring	Outer ring	Rolling element set & cage	Rolling element about its axis
		f_i	f_e	f_c	f_r
		Hz			
Left	**61900-2Z**	149	0	59.7	361
Right	**61900-2Z**	149	0	59.7	361

	Designation	Frequency of over-rolling		
		Point on inner ring	Point on outer ring	Rolling element
		f_{ip}	f_{ep}	f_{rp}
		Hz		
Left	**61900-2Z**	804	538	722
Right	**61900-2Z**	804	538	722

3.9. Fits and tolerances

3.9.1. Tolerances

	Designation	Shaft outer diameter		Bearing bore		Bearing outer diameter	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
		µm					
Left	61900-2Z	1	10	-8	0	-9	0
Right	61900-2Z	1	10	-8	0	-9	0

	Designation	Housing bore		Smoothing	
		Minimum	Maximum	Shaft and bearing bore	Bearing outer ring and housing
		µm			
Left	61900-2Z	0	21	7	12
Right	61900-2Z	0	21	7	12

- *For the tolerances calculation, the normal tolerance for the bearing bore and outer diameter is used.*

3.9.2. Fits, Probable Interference (+) / Clearance (-)

	Designation	Shaft			Housing		
		Probable minimum	Middle	Probable maximum	Probable minimum	Middle	Probable maximum
		µm					
Left	61900-2Z	-4	2	9	-38	-27	-16
Right	61900-2Z	-4	2	9	-38	-27	-16

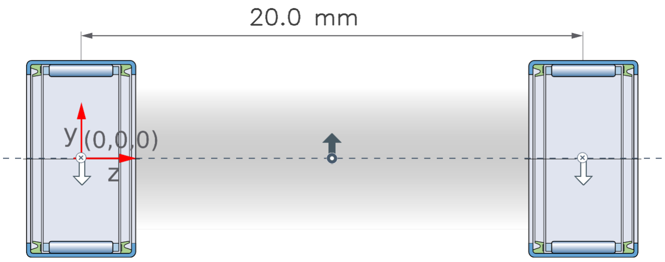
LIMITED WARRANTY : The SKF Bearing Select software tool of the SKF Company supports the calculation and selection of bearings.
This software is provided 'as is' with out any warranty of any sort, implicitly as well as explicitly.
Please note that the obtained results can be affected by many external parameters and/or the quality of the assumptions taken into account.
The results obtained using this software must be validated by the user who accepts the fact that the use of this software and the exploitation of the obtained results are under the user's entire and sole responsibility.

B Appendix - Belt System 2 Idler Pulley Bearing SKF Calculations

SKF Bearing Select v1.2-45

Belt System 2 Idler Bearings

Calculations for bearings used in the idler pulley found in the lower belt system of the drive unit

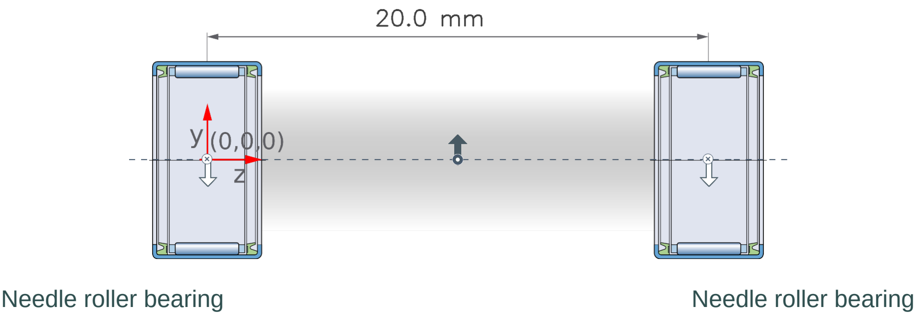


Alex Macfarlane

NMU

August 19, 2020

1. Abstract



■ SKF Explorer ▶ Popular item

	Designation	Life model		Static safety factor
		Basic	SKF life	
		L_{10h}	L_{10mh}	S_0
		h		
Left	**HK 1616.2RS**	> 2x10^5	161000	20.8
Right	**HK 1616.2RS**	> 2x10^5	161000	20.8

** SKF rating life (L_{10mh}) for steel-steel bearings; GBLM load based life (L_{10GMh}) for hybrid bearings*

Left bearing

! Low viscosity ratio k, reduced asperity contact. It is recommended to select a higher viscosity lubricant or improve cooling. It is not appropriate to look at basic rating life only. Instead use SKF rating life method. Recommended to use anti-wear (AW) or extreme pressure (EP) additives to reduce wear More info

! For rating life results above 100000 hours, other failure modes than those included in the current rating life models will dominate and limit the life of the bearing. More info

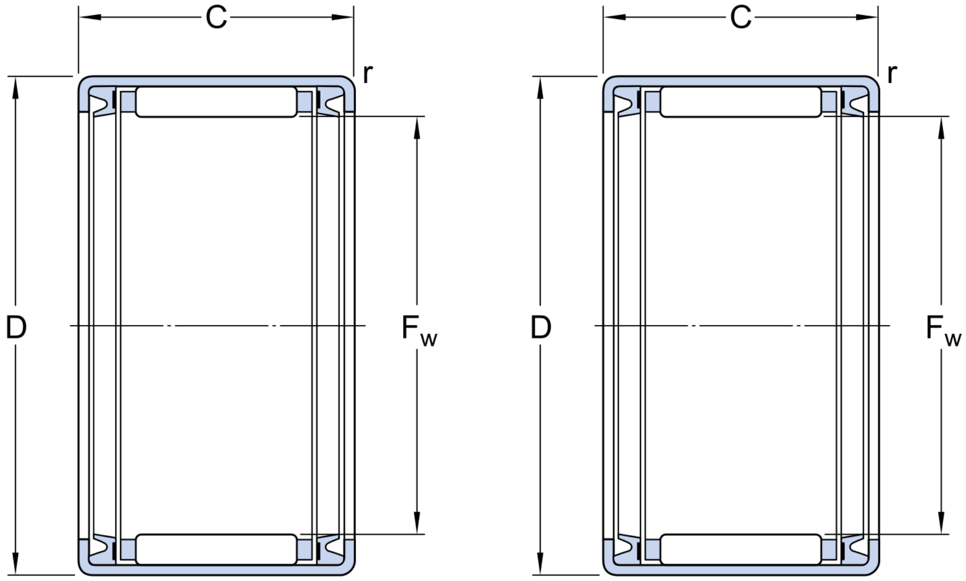
Right bearing

! Low viscosity ratio k, reduced asperity contact. It is recommended to select a higher viscosity lubricant or improve cooling. It is not appropriate to look at basic rating life only. Instead use SKF rating life method. Recommended to use anti-wear (AW) or extreme pressure (EP) additives to reduce wear More info

! For rating life results above 100000 hours, other failure modes than those included in the current rating life models will dominate and limit the life of the bearing. More info

2. Input

2.1. Bearing data



	Designation	Bearing type	Principal dimensions			Basic load ratings		Fatigue load limit
						Dynamic	Static	
			d	D	B	C	C_0	P_u
			mm			*kN*		
Left	**HK 1616.2RS**	Needle roller bearing	16	22	16	7.37	9.8	1.12
Right	**HK 1616.2RS**	Needle roller bearing	16	22	16	7.37	9.8	1.12

	Designation	Speed ratings	
		Reference	Limiting
		n_{ref}	n_{lim}
		r/min	
Left	**HK 1616.2RS**		9000
Right	**HK 1616.2RS**		9000

2.2. Loads & Speed

Locating bearing	None (axial load ignored)	
Bearing distance	20.0	*mm*
Rotating ring	Inner ring rotation	

	Load	**Coordinate system**	**Coordinates**			**Forces**			**Speed**	**Case weight**
			x|r	**y|θ**	**z**	**Fx|Fr**	**Fy|Fθ**	**Fz**		
			mm	*mm|deg*	*mm*	*kN*	*kN*	*kN*	*r/min*	
LC1	F1	Cartesian	0.0	0.0	10.0	-0.864	0.376	0.0	476.65	1

2.3. Temperature

Load Cases	**Left**		**Right**	
	Inner ring	Outer ring	Inner ring	Outer ring
	°C			
LC1	70	65	70	65

- *Maximum temperature is used for calculating the actual viscosity, kappa, a_{SKF} and SKF rating life.*
- *Mean temperature is used for calculating bearing friction and power loss.*

2.4. Lubrication

	Designation	Lubricant	Effective EP additives
		Name	
Left	HK 1616.2RS	LGWA2	False
Right	HK 1616.2RS	LGWA2	False

	Designation	Contamination
		Method
Left	HK 1616.2RS	Detailed guidelines
Right	HK 1616.2RS	Detailed guidelines

2.5. Fits and tolerances

	Designation	Requirements	Tolerance Class		Calculated interference	Include Smoothing
		Guidance	Housing	Shaft		
Left	HK 1616.2RS	False	H7	N/A	True	True
Right	HK 1616.2RS	False	H7	N/A	True	True

3. Results

3.1. Loads & static safety

	Designation	Load ratio	Static safety factor	Equivalent dynamic load	Equivalent static load
		C/P	S_0	P	P_0
				kN	
Left	**HK 1616.2RS**	15.64	20.8	0.47	0.471
Right	**HK 1616.2RS**	15.64	20.8	0.47	0.471

3.2. Bearing minimum load

	Designation	Reaction forces		Minimum load	
		Radial	Axial		met?
		F_r	F_a	$F_{r\ m}$	
		kN			
Left	**HK 1616.2RS**	0.471	0	0.147	yes
Right	**HK 1616.2RS**	0.471	0	0.147	yes

3.3. Lubrication conditions

	Designation	Operating viscosity			Viscosity ratio
		Actual	Rated	Rated @ 40 °C	
		ν	ν_1	ν_{ref}	K
		mm²/s			
Left	**HK 1616.2RS**	41.4	54.2	231	0.76
Right	**HK 1616.2RS**	41.4	54.2	231	0.76

3.4. Bearing rating life

	Designation	Bearing rating life		SKF life modification factor	Contamination factor
		Basic	SKF		
		L_{10h}	L_{10mh}	a_{skf}	η_c
		h			
Left	**HK 1616.2RS**	> 2x10^5	161000	0.48	0.1
Right	**HK 1616.2RS**	> 2x10^5	161000	0.48	0.1

** SKF rating life (L_{10mh}) for steel-steel bearings; GBLM load based life (L_{10GMh}) for hybrid bearings*

Left bearing

! Low viscosity ratio k, reduced asperity contact. It is recommended to select a higher viscosity lubricant or improve cooling. It is not appropriate to look at basic rating life only. Instead use SKF rating life method. Recommended to use anti-wear (AW) or extreme pressure (EP) additives to reduce wear More info

! For rating life results above 100000 hours, other failure modes than those included in the current rating life models will dominate and limit the life of the bearing. More info

Right bearing

! Low viscosity ratio k, reduced asperity contact. It is recommended to select a higher viscosity lubricant or improve cooling. It is not appropriate to look at basic rating life only. Instead use SKF rating life method. Recommended to use anti-wear (AW) or extreme pressure (EP) additives to reduce wear More info

! For rating life results above 100000 hours, other failure modes than those included in the current rating life models will dominate and limit the life of the bearing. More info

3.5. Bearing frequencies

	Designation	Rotational frequencies			
		Inner ring	Outer ring	Rolling element set & cage	Rolling element about its axis
		f_i	f_e	f_c	f_r
		Hz			
Left	**HK 1616.2RS**	7.94	0	3.53	35.3
Right	**HK 1616.2RS**	7.94	0	3.53	35.3

	Designation	Frequency of over-rolling		
		Point on inner ring	Point on outer ring	Rolling element
		f_{ip}	f_{ep}	f_{rp}
		Hz		
Left	**HK 1616.2RS**	79.4	63.5	70.6
Right	**HK 1616.2RS**	79.4	63.5	70.6

3.6. Fits and tolerances

3.6.1. Tolerances

	Designation	Shaft outer diameter		Bearing bore		Bearing outer diameter	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
		µm					
Left	**HK 1616.2RS**	N/A	N/A	N/A	N/A	-9	0
Right	**HK 1616.2RS**	N/A	N/A	N/A	N/A	-9	0

	Designation	Housing bore		Smoothing	
		Minimum	Maximum	Shaft and bearing bore	Bearing outer ring and housing
		µm			
Left	**HK 1616.2RS**	0	21	N/A	12
Right	**HK 1616.2RS**	0	21	N/A	12

- *For the tolerances calculation, the normal tolerance for the bearing bore and outer diameter is used.*

3.6.2. Fits, Probable Interference (+) / Clearance (-)

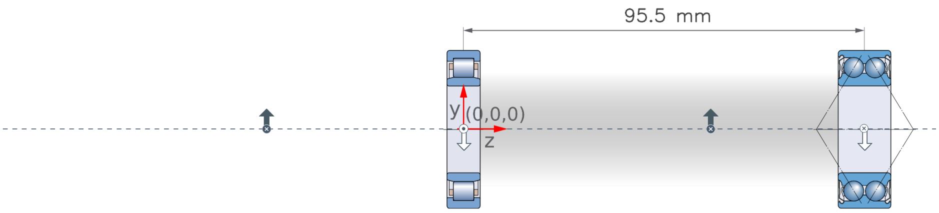
	Designation	Shaft			Housing		
		Probable minimum	Middle	Probable maximum	Probable minimum	Middle	Probable maximum
		µm					
Left	**HK 1616.2RS**	N/A	N/A	N/A	-38	-27	-16
Right	**HK 1616.2RS**	N/A	N/A	N/A	-38	-27	-16

C Appendix - Drive Unit Wheel Axle Bearing Calculations

SKF Bearing Select v1.2-45

Drive Unit Wheel Axle Bearings

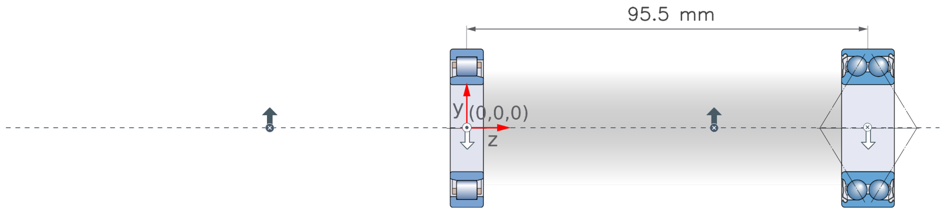
Calculations for the bearings found on the wheel axle of the drive unit. The left bearing is a roller bearing and the right bearing is an angular contact ball bearing



Alex Macfarlane
NMU

August 19, 2020

1. Abstract



Cylindrical roller bearing

Non-Locating

Angular contact ball bearing

Locating

🟦 SKF Explorer ▶ Popular item

	Designation	Life model		Grease		Static safety factor	Frictional moment	Power loss
		Basic	SKF life	Relubrication interval	Catalogue grease life		Total	
		L_{10h}	L_{10mh}	t_f	L_{10}	S_0	M	P_{loss}
		h					*Nmm*	*W*
Left	▶ NU 204 ECP	119000	16800	5920		6.47	29.0	0.51
Right	▶ 3205 A-2Z	> 2x10^5	> 2x10^5		100000	52.7	8.42	0.15

** SKF rating life (L_{10mh}) for steel-steel bearings; GBLM load based life (L_{10GMh}) for hybrid bearings*

Left bearing

! Results are based on default operating conditions. Please, review and adjust operating conditions where needed!

! The grease life / relubrication interval is reduced depending on the contamination level. Higher cleanliness will improve the duration.

! The grease life / relubrication interval is halved for bearings fitted with J, JA, JB, MA, MB, ML, MP and PHA cages.

! Low viscosity ratio k, reduced asperity contact. It is recommended to select a higher viscosity lubricant or improve cooling. It is not appropriate to look at basic rating life only. Instead use SKF rating life method. Recommended to use anti-wear (AW) or extreme pressure (EP) additives to reduce wear More info

Right bearing

! Low viscosity ratio k, reduced asperity contact. It is recommended to select a higher viscosity lubricant or improve cooling. It is not appropriate to look at basic rating life only. Instead use SKF rating life method. Recommended to use anti-wear (AW) or extreme pressure (EP) additives to reduce wear More info

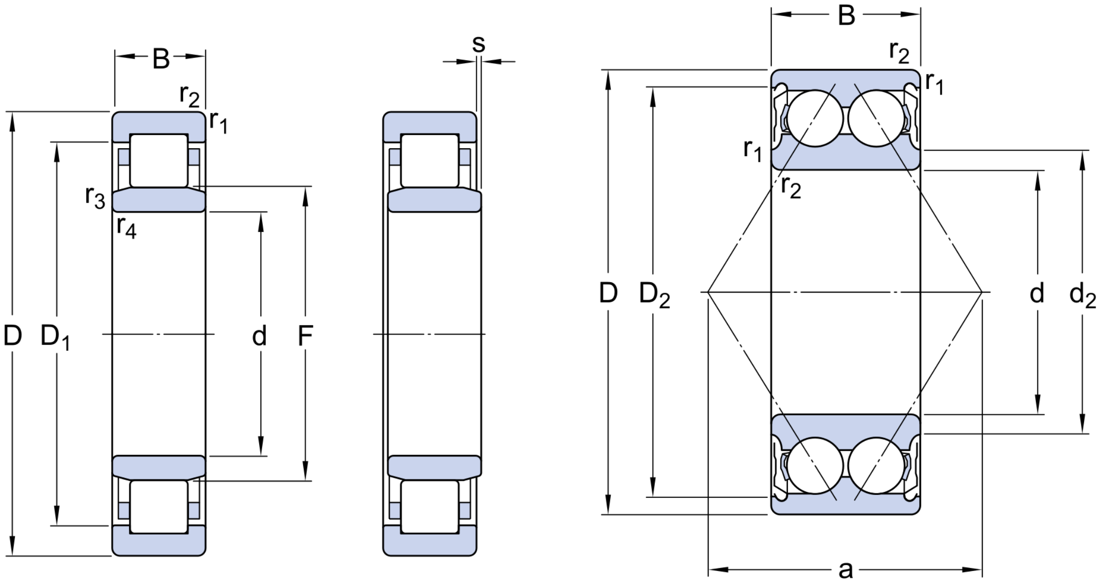
! Results are based on default operating conditions. Please, review and adjust operating conditions where needed!

! The grease life / relubrication interval is reduced depending on the contamination level. Higher cleanliness will improve the duration.

! For rating life results above 100000 hours, other failure modes than those included in the current rating life models will dominate and limit the life of the bearing. More info

2. Input

2.1. Bearing data



Designation		Bearing type	Principal dimensions			Basic load ratings		Fatigue load limit
						Dynamic	Static	
			d	D	B	C	C_0	P_u
			mm			*kN*		
Left	▶ NU 204 ECP	Cylindrical roller bearing	20	47	14	28.5	22	2.75
Right	▶ 3205 A-2Z	Angular contact ball bearing	25	52	20.6	22	15.3	0.64

Designation		Speed ratings	
		Reference	Limiting
		n_{ref}	n_{lim}
		r/min	
Left	▶ NU 204 ECP	17000	19000
Right	▶ 3205 A-2Z	12000	12000

2.2. Loads & Speed

Locating bearing	Right	
Bearing distance	95.5	*mm*
Shaft orientation	Horizontal	
Rotating ring	Inner ring rotation	

	Load	**Coordinate system**	**Coordinates**			**Forces**			**Speed**	**Case weight**
			x|r	**y|θ**	**Z**	**Fx|Fr**	**Fy|Fθ**	**Fz**		
			mm	*mm|deg*	*mm*	*kN*	*kN*	*kN*	*r/min*	
LC1	F1	Cartesian	0.0	0.0	-42.55	0.995	1.703	0.0	166.39	1
	F2	Cartesian	0.0	0.0	59.0	0.315	1.472	0.0		

2.3. Temperature

Load Cases	**Left**		**Right**	
	Inner ring	Outer ring	Inner ring	Outer ring
	°C			
LC1	70	65	70	65

- *Maximum temperature is used for calculating the actual viscosity, kappa, a_{SKF} and SKF rating life.*
- *Mean temperature is used for calculating bearing friction and power loss.*

2.4. Lubrication

	Designation	Lubricant			Effective EP additives
		Type	method	Name	
Left	▶ NU 204 ECP	Grease	SKF grease	LGMT 2: all purpose industrial and automotive	False
Right	▶ 3205 A-2Z			GJN	False

	Designation	Contamination
		Method
Left	▶ NU 204 ECP	Detailed guidelines
Right	▶ 3205 A-2Z	Detailed guidelines

2.5. Fits and tolerances

	Designation	Requirements	Tolerance Class		Calculated interference	Include Smoothing
		Guidance	Housing	Shaft		
Left	▶ NU 204 ECP	False	H7	k6	True	True
Right	▶ 3205 A-2Z	False	H7	k6	True	True

3. Results

3.1. Loads & static safety

	Designation	Load ratio	Static safety factor	Equivalent dynamic load	Equivalent static load
		C/P	S_0	P	P_0
				kN	
Left	▶ NU 204 ECP	8.38	6.47	3.4	3.4
Right	▶ 3205 A-2Z	75.78	52.7	0.29	0.29

3.2. Bearing minimum load

	Designation	Reaction forces		Minimum load	
		Radial	Axial		met?
		F_r	F_a	$F_{r\ m}$	
		kN			
Left	▶ NU 204 ECP	3.4	0	0.105	yes
Right	▶ 3205 A-2Z	0.29	0	0.0263	yes

3.3. Adjusted reference speed

	Designation	Adjusted reference speed	Adjustment factors	
			For bearing load P	For oil viscosity
		n_{ar}	f_p	f_v
		r/min		
Left	▶ NU 204 ECP	11600	0.69	1.0
Right	▶ 3205 A-2Z	12000	1.0	1.0

3.4. Lubrication conditions

	Designation	Operating viscosity			Viscosity ratio
		Actual	Rated	Rated @ 40 °C	
		ν	ν_1	ν_{ref}	K
		mm²/s			
Left	▶ NU 204 ECP	28.0	97.3	475	0.28
Right	▶ 3205 A-2Z	30.5	91.2	439	0.33

3.5. Grease life and relubrication interval

	Designation	Catalogue grease life	Relubrication interval	Grease quantity	Speed factor
				Side	Speed x mean diameter
		L_{10}	t_f	G_p	nd_m
		h		*g*	*mm/min*
Left	▶ NU 204 ECP		5920	3	5650
Right	▶ 3205 A-2Z	100000			6400

Left bearing

! The grease life / relubrication interval is reduced depending on the contamination level. Higher cleanliness will improve the duration.

! The grease life / relubrication interval is halved for bearings fitted with J, JA, JB, MA, MB, ML, MP and PHA cages.

Right bearing

! The grease life / relubrication interval is reduced depending on the contamination level. Higher cleanliness will improve the duration.

3.6. Bearing rating life

	Designation	Bearing rating life		SKF life modification factor	Contaminati on factor
		Basic	SKF		
		L_{10h}	L_{10mh}	a_{skf}	η_c
		h			
Left	▶ NU 204 ECP	119000	16800	0.14	0.08
Right	▶ 3205 A-2Z	> 2x10^5	> 2x10^5	0.54	0.1

** SKF rating life (L_{10mh}) for steel-steel bearings; GBLM load based life (L_{10GMh}) for hybrid bearings*

Left bearing

! Low viscosity ratio k, reduced asperity contact. It is recommended to select a higher viscosity lubricant or improve cooling. It is not appropriate to look at basic rating life only. Instead use SKF rating life method. Recommended to use anti-wear (AW) or extreme pressure (EP) additives to reduce wear More info

Right bearing

! Low viscosity ratio k, reduced asperity contact. It is recommended to select a higher viscosity lubricant or improve cooling. It is not appropriate to look at basic rating life only. Instead use SKF rating life method. Recommended to use anti-wear (AW) or extreme pressure (EP) additives to reduce wear More info

! For rating life results above 100000 hours, other failure modes than those included in the current rating life models will dominate and limit the life of the bearing. More info

3.7. Bearing friction & power loss

	Designation	Frictional moment		Friction Sources				Power loss
		Total	At start 20-30°C and zero speed	Rolling	Sliding	Seals	Drag loss	
		M	M_{start}	M_{rr}	M_{sl}	M_{seal}	M_{drag}	P_{loss}
		Nmm						*W*
Left	▶ **NU 204 ECP**	29.0	25.6	10.7	18.3	0	0	0.51
Right	▶ **3205 A-2Z**	8.42	8.04	2.58	5.83	0	0	0.15

3.8. Bearing frequencies

	Designation	Rotational frequencies			
		Inner ring	Outer ring	Rolling element set & cage	Rolling element about its axis
		f_i	f_e	f_c	f_r
		Hz			
Left	▶ **NU 204 ECP**	2.77	0	1.08	5.97
Right	▶ **3205 A-2Z**	2.77	0	1.13	6.51

	Designation	Frequency of over-rolling		
		Point on inner ring	Point on outer ring	Rolling element
		f_{ip}	f_{ep}	f_{rp}
		Hz		
Left	▶ **NU 204 ECP**	18.6	11.8	11.9
Right	▶ **3205 A-2Z**	14.7	10.2	13.0

3.9. Fits and tolerances

3.9.1. Tolerances

	Designation	Shaft outer diameter		Bearing bore		Bearing outer diameter	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
		µm					
Left	▶ NU 204 ECP	2	15	-10	0	-11	0
Right	▶ 3205 A-2Z	2	15	-10	0	-13	0

	Designation	Housing bore		Smoothing	
		Minimum	Maximum	Shaft and bearing bore	Bearing outer ring and housing
		µm			
Left	▶ NU 204 ECP	0	25	7	12
Right	▶ 3205 A-2Z	0	30	7	12

- *For the tolerances calculation, the normal tolerance for the bearing bore and outer diameter is used.*

3.9.2. Fits, Probable Interference (+) / Clearance (-)

	Designation	Shaft			Housing		
		Probable minimum	Middle	Probable maximum	Probable minimum	Middle	Probable maximum
		µm					
Left	▶ NU 204 ECP	-2	6	15	-44	-30	-16
Right	▶ 3205 A-2Z	-2	6	15	-50	-34	-17

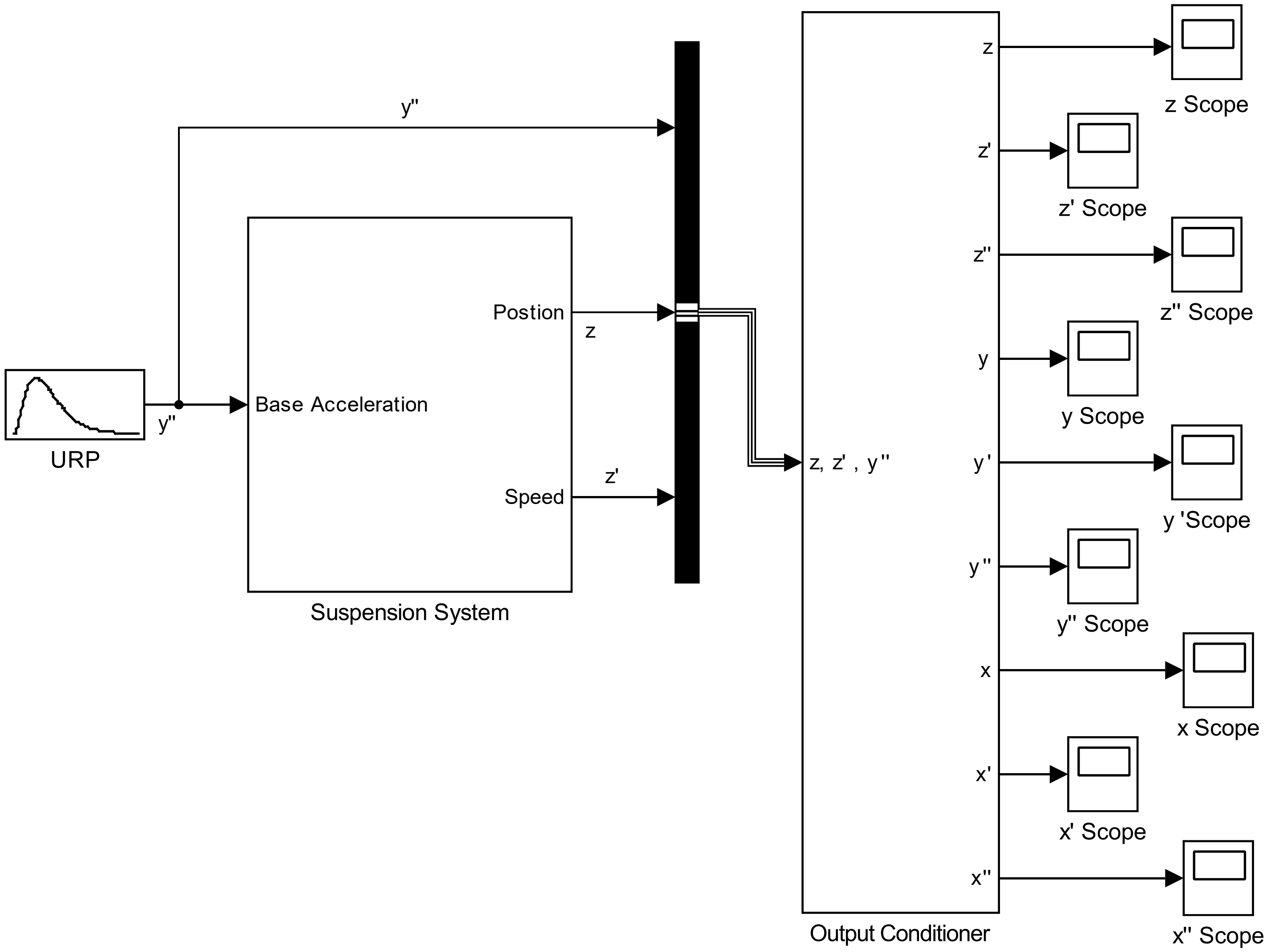
D Appendix - Suspension System Model

```
function [sys,x0,str,ts] = URPFcn(t, x, u, flag, X, gw, Output, SampleTime)
% X = pulse magnitude
% gw = intensity factor*frequency
% Output = where 1 - position, 2 - velocity, 3 - acceleration
switch flag,
  %%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes(SampleTime);

  %%%%%%%%%%
  % Output %
  %%%%%%%%%%
  case 3,
    sys=mdlOutputs(t,x,u, X, gw, Output);

  %%%%%%%%%%%%%%%%
  % Unused flags %
  %%%%%%%%%%%%%%%%
  case { 1, 2, 4, 9 },
    sys = [];

  %%%%%%%%%%%%%%%%%%%%
  % Unexpected flags %
  %%%%%%%%%%%%%%%%%%%%
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end


%===========================================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function
%===========================================================================
%
function [sys,x0,str,ts]=mdlInitializeSizes(SampleTime)

%
% call simsizes for a sizes structure, fill it in and convert it to a
% sizes array.
%
sizes = simsizes;

sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 0;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;


sys = simsizes(sizes);

%
% initialize the initial conditions
%
x0  = [];

%
% str is always an empty matrix
%
str = [];

%
% inherited ts = [-1 0], continuous is [0 0]
%
ts  = [SampleTime 0];




%==========================================================================
% mdlOutputs
% Output the signal
%==========================================================================
%
function sys=mdlOutputs(t,x,u,X,gw,Output)
%  x = X*(e^2/4)*(g*w*t)^2*e^-g*w*t
%  x'' = X*(e^2/4)*g^2*w^2*(2-4*g*w*t+w^2*g^2*t^2)*e^(-g*w*t)
if Output == 1
    sys = X*(exp(2)/4)*(gw*t)^2*exp(-gw*t);
elseif Output ==2
    sys = X*(exp(2)/4)*(gw)^2*t*(2 - gw*t)*exp(-gw*t);
elseif Output == 3
    sys = X*exp(2)/4*gw^2*(2-4*gw*t+gw^2*t^2)*exp(-gw*t);
end

% end mdlUpdate
```

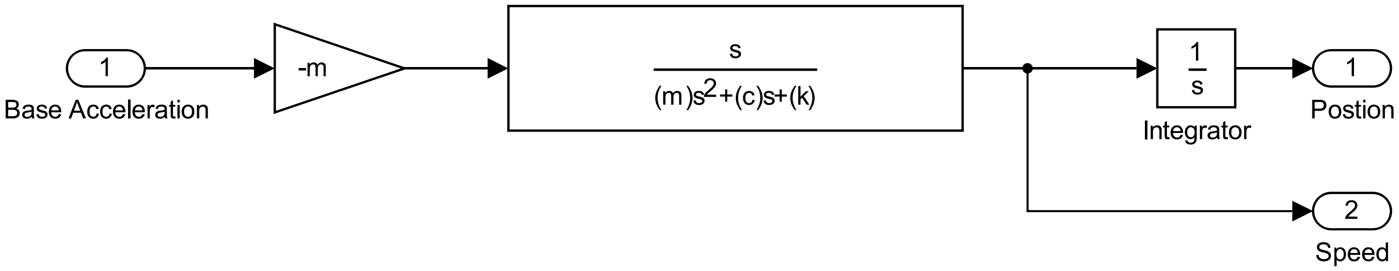
```
%Simulation Parameters
%*************************************************************************
clear;
clc;

%User Defined Variables
%*************************************************************************
        STime  = 4;         %simulation time (s)
        OpenModel  = 'y';   %open simulink model (y/n)
        xAxisLim = 0.3;

        %Bump Input Conditions____________________________________________
        Am = 0.01;          %amplitude of bump (m)
        Dbump = 0.02;       %lenght of bump (m)
        Speed = 1.3;        %horizontal speed (m/s)


        %Physical Suspension Parameters___________________________________
        c = 220;            %dampening co-efficient (Ns/m)
        k = 53848;          %spring co-efficient (N/m)
        m = 150;            %quater mass of system (kg)

%Calculations
%*************************************************************************
Tm  = Dbump/Speed;
GamOm = (2*pi())/Tm;

%Run Simulation
%*************************************************************************
if OpenModel == 'y'
    open('MechanicalSuspension.mdl');
end
sim('MechanicalSuspension.mdl');

%Print Object Graphs
%*************************************************************************

%input____________________________________________________________________
InG = figure('Position',[100, 100, 604, 302]);
plot(y_disp(:,1),y_disp(:,2),'k');
Ftitle = title('Surface Profile');
set(Ftitle,'FontSize',20);
xlabel('Time [s]');
ylabel('Displacement [m]');
xlim([0,STime]);


%displacement_____________________________________________________________
dispG=figure('Position',[100, 100, 604, 302]);
```

```
plot(y_disp(:,1),y_disp(:,2),'k',z_disp(:,1),z_disp(:,2),'k:',x_disp(:,1),x_disp(:,↙
2),'k--');

Gtitle = title('Displacements');
set(Gtitle,'FontSize',20);
xlabel('Time [s]');
ylabel('Displacement [m]');
legend('wheel (Zs)','difference (Z)','body (Zu)');
xlim([0,0.1]);

%velocity_________________________________________________________________
velG = figure('Position',[100, 100, 604, 302]);
plot(y_vel(:,1),y_vel(:,2),'k',z_vel(:,1),z_vel(:,2),'k:',x_vel(:,1),x_vel(:,2),'k--');
Htitle = title('Velocities');
set(Htitle,'FontSize',20);
xlabel('Time [s]');
ylabel('Velocity [m/s]');
legend('wheel (Zs)','difference (Z)','body (Zu)');
xlim([0,0.04]);

%acceleration_____________________________________________________________
accG= figure('Position',[100, 100, 604, 302]);
plot(y_acc(:,1),y_acc(:,2),'k',z_acc(:,1),z_acc(:,2),'k:',x_acc(:,1),x_acc(:,2),'k--');
Ititle = title('Accelerations');
set(Ititle,'FontSize',20);
xlabel('time [s]');
ylabel('Acceleration [m/s^2]');
legend('wheel (Zs)','difference (Z)','body (Zu)');
xlim([0,0.025]);
```

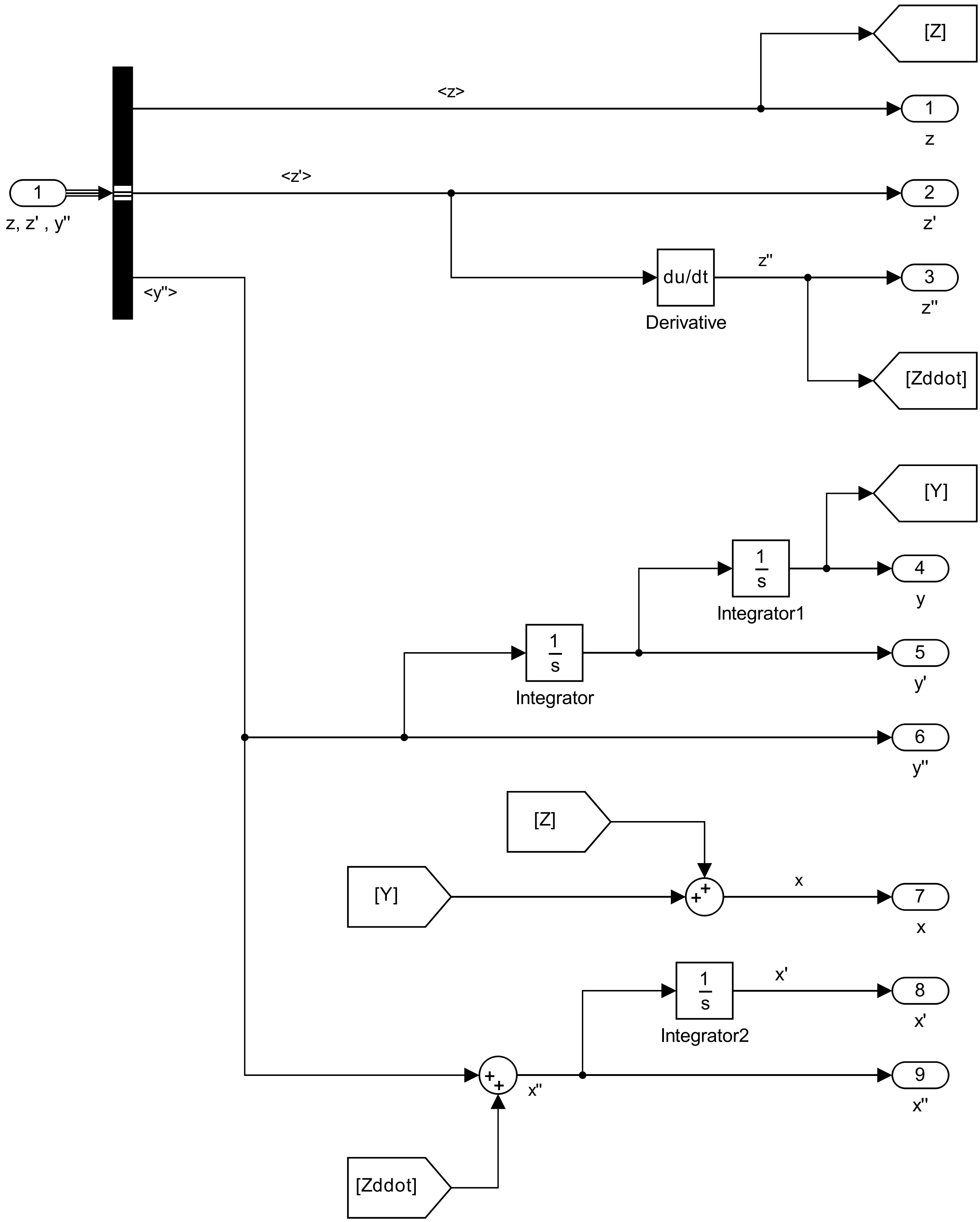
Main Simulink Model



Suspension System



Output Conditioner



E Appendix - Drive Unit Suspension Spring Autodesk Inventor Calculator

Drive Unit Spring Design Results

22/10/2020

Project Info

Summary: Drive unit suspension system spring for swerve drive AGV

Project: Swerve drive AGV

Status: **Completed**

Guide

Spring Type	Guided mounting - parallel ground ends
Spring Strength Calculation	Compression Spring Design
Design Type	F, D --> d, L_0, n, Assembly Dimensions
Method of Stress Curvature Correction	No Correction

Spring Load

Min. Load	F_1	320.000 N
Max. Load	F_8	900.000 N
Working Load	F	600.000 N

Spring Dimensions

Loose Spring Length	L_0	110.746 mm
Wire Diameter	d	5.000 mm
Pitch of Free Spring	t	19.899 mm
Outside Spring Diameter	D_1	48.000 mm
Mean Spring Diameter	D	43.000 mm
Inside Spring Diameter	D_2	38.000 mm
Spring Index	c	8.600 ul

Spring Coils

Active Coils	n	5.000 ul		
Rounding of Coils Number	1			
Coil Direction	right			
Spring Ends				
Params	Start		End	
Closed End Coils	n_{z1}	1.500 ul	n_{z2}	1.000 ul
Transition Coils	n_{t1}	1.000 ul	n_{t2}	0.750 ul
Ground Coils	z_{o1}	0.750 ul	z_{o2}	0.500 ul

Assembly Dimensions

Min. Load Length	L_1	86.975 mm

Max. Load Length	L_8	43.890 mm
Working Stroke	H	43.085 mm
Working Load Length	L_W	66.175 mm
Installed Length	L	66.175 mm

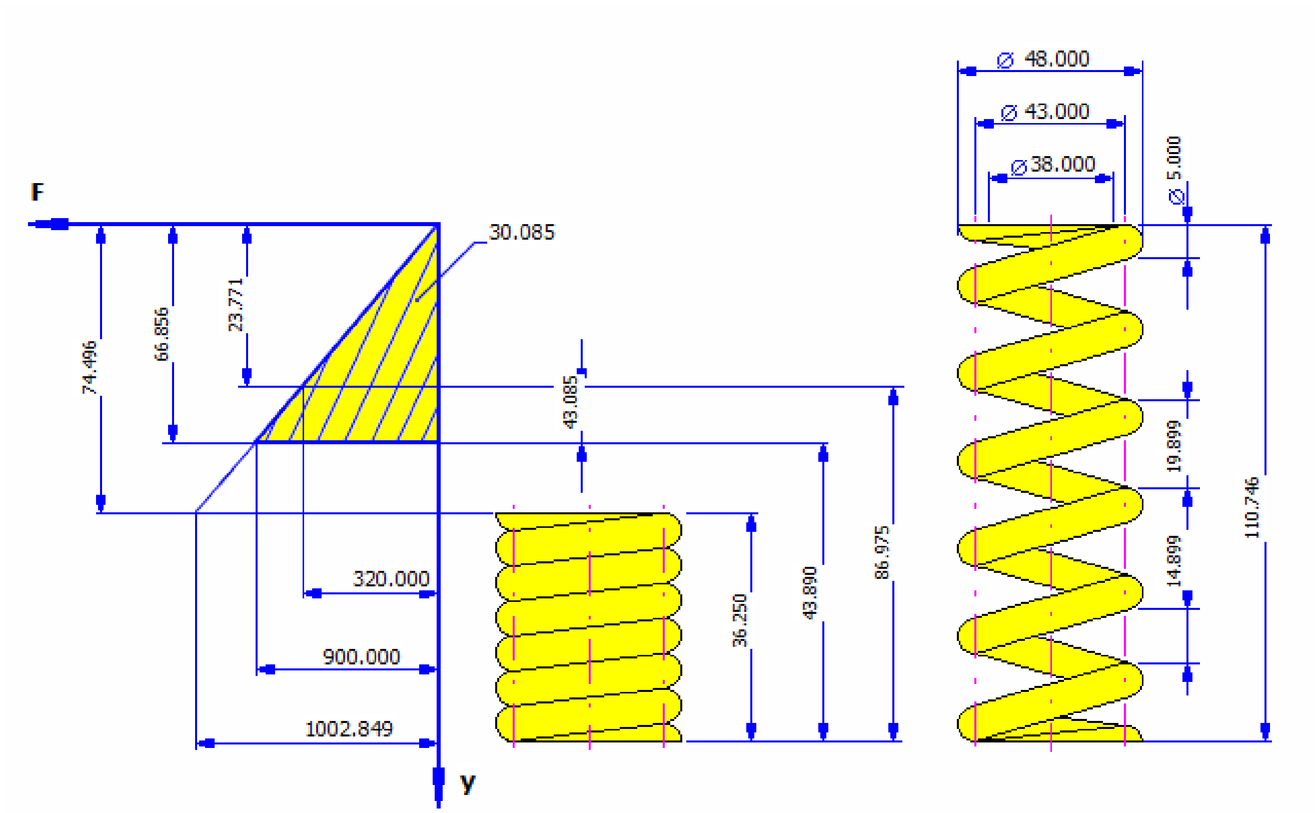
Design of Working Deflection

Not Specified

Spring Material

User material		
Ultimate Tensile Stress	σ_{ult}	1860.000 MPa
Allowable Torsional Stress	τ_A	930.000 MPa
Modulus of Elasticity in Shear	G	68500.000 MPa
Density	ρ	7850 kg/m^3
Utilization Factor of Spring Material	us	0.900 ul

Working Diagram



Results

Space between Coils of Free Spring	a	14.899 mm
Pitch of Free Spring	t	19.899 mm
Stress Concentration Factor	K_W	1.000 ul
Spring Constant	k	13.462 N/mm
Min. Load Spring Deflection	s_1	23.771 mm
Total Spring Deflection	s_8	66.856 mm

Limit Spring Deflection	s_9	74.496 mm
Working Spring Deflection	s_{work}	0.604 ul
Max. Allowable Spring Deflection	s_{max}	0.677 ul
Limit Test Length of Spring	L_{minf}	41.800 mm
Theoretic Limit Length of Spring	L_9	36.250 mm
Spring Limit Force	F_9	1002.849 N
Min. Load Stress	τ_1	280.316 MPa
Max. Load Stress	τ_8	788.390 MPa
Solid Length Stress	τ_9	878.484 MPa
Critical Speed of Spring	v	2.747 mps
Natural Frequency of Spring Surge	f	179.795 Hz
Deformation Energy	W_8	30.085 J
Wire Length	l	1032.000 mm
Spring Mass	m	0.159 kg
Spring Check Result		**Positive**

Summary of Messages

01:41:03 PM : Calculation indicates design compliance!

F Appendix - Caster Unit Suspension Spring Autodesk Inventor Calculator

Caster Unit Spring Design Results

27/10/2020

Project Info

Summary: Caster unit suspension system spring for swerve drive AGV

Project: Swerve drive AGV

Status: **Completed**

Guide

Spring Type	Guided mounting - parallel ground ends
Spring Strength Calculation	Spring Check Calculation
Method of Stress Curvature Correction	Correction by Wahl

Spring Load

Min. Load	F_1	300.000 N
Max. Load	F_8	900.000 N
Working Load	F	600.000 N

Spring Dimensions

Loose Spring Length	L_0	84.000 mm
Wire Diameter	d	6.000 mm
Pitch of Free Spring	t	14.100 mm
Outside Spring Diameter	D_1	48.000 mm
Mean Spring Diameter	D	42.000 mm
Inside Spring Diameter	D_2	36.000 mm
Spring Index	c	7.000 ul

Spring Coils

Active Coils	n	5.000 ul		
Coil Direction	right			
Spring Ends				
Params	Start		End	
Closed End Coils	n_{z1}	1.500 ul	n_{z2}	1.000 ul
Transition Coils	n_{t1}	1.000 ul	n_{t2}	0.750 ul
Ground Coils	z_{o1}	0.750 ul	z_{o2}	0.500 ul

Assembly Dimensions

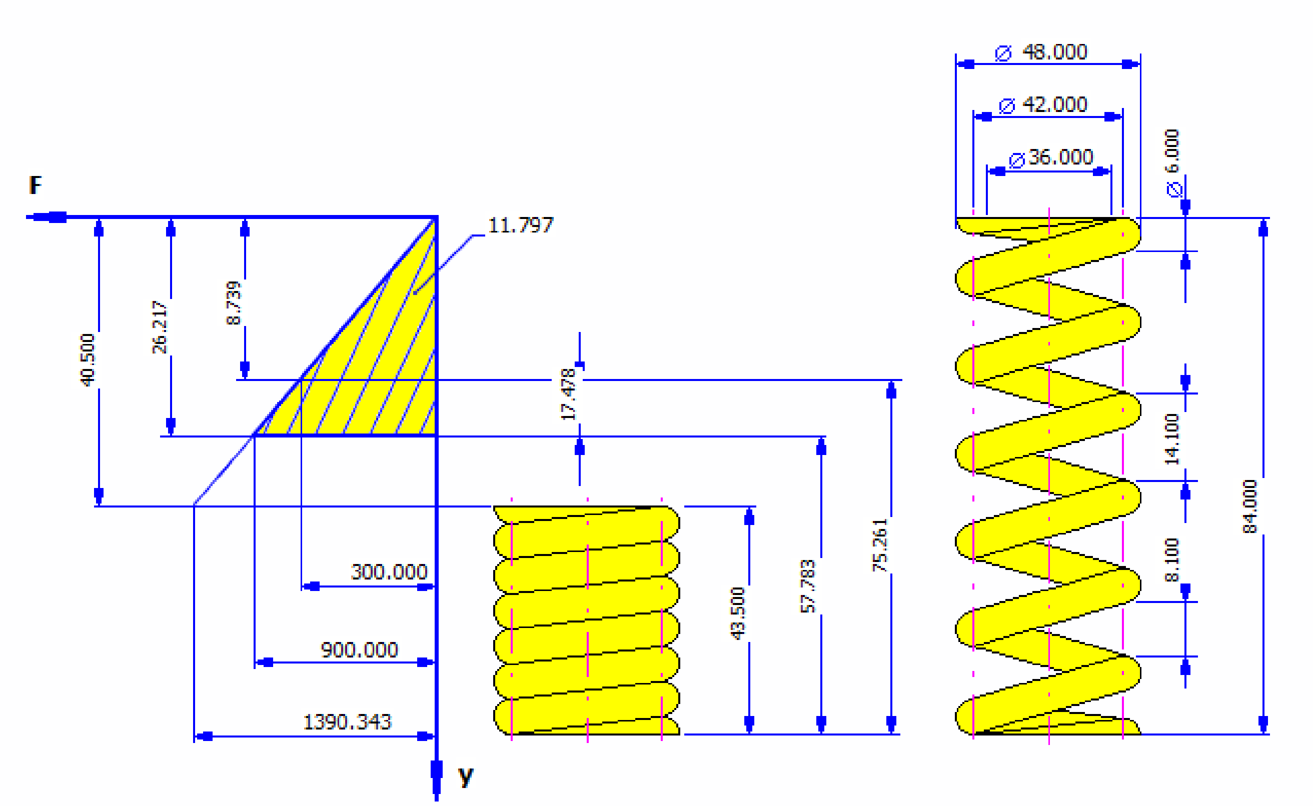
Min. Load Length	L_1	75.261 mm
Max. Load Length	L_8	57.783 mm
Working Stroke	H	17.478 mm

Working Load Length	L_W	66.522 mm
Installed Length	L	75.261 mm

Spring Material

Heat treated wire carbon steel		
Ultimate Tensile Stress	σ_{ult}	1260.000 MPa
Allowable Torsional Stress	τ_A	756.000 MPa
Modulus of Elasticity in Shear	G	78500.000 MPa
Density	ρ	7850 kg/m^3
Utilization Factor of Spring Material	us	0.900 ul

Working Diagram



Results

Space between Coils of Free Spring	a	8.100 mm
Pitch of Free Spring	t	14.100 mm
Stress Concentration Factor	K_W	1.213 ul
Spring Constant	k	34.329 N/mm
Min. Load Spring Deflection	s_1	8.739 mm
Total Spring Deflection	s_8	26.217 mm
Limit Spring Deflection	s_9	40.500 mm
Working Spring Deflection	s_{work}	0.312 ul
Max. Allowable Spring Deflection	s_{max}	0.700 ul
Limit Test Length of Spring	L_{minf}	49.200 mm
Theoretic Limit Length of Spring	L_9	43.500 mm

Spring Limit Force	F_9	1390.343 N
Min. Load Stress	τ_1	180.163 MPa
Max. Load Stress	τ_8	540.490 MPa
Solid Length Stress	τ_9	834.963 MPa
Critical Speed of Spring	v	8.388 mps
Natural Frequency of Spring Surge	f	242.096 Hz
Deformation Energy	W_8	11.797 J
Wire Length	l	1008.000 mm
Spring Mass	m	0.224 kg
Spring Check Result		**Positive**

Summary of Messages

09:24:36 AM : Calculation indicates design compliance!

G Appendix - AGV Design Tool Code and GUI

G.1 Main & Start GUI

The order of execution of the GUI windows and flow of the program:

1. Start Program
2. Initialise Start GUI
3. Start GUI Exited
4. Initialise Initial Data Collection GUI
5. Next button pressed
6. Initial Data Collection GUI Exited
7. Initialise drive force results GUI
8. Next button pressed
9. Initialise wheel results GUI
10. Next button pressed
11. Initialise motor requirements GUI
12. Next button pressed
13. Initialise motor selection GUI
14. Calculate button pressed
15. Initialise system conditions GUI
16. Calculate button pressed

17. Initialise gear train development GUI
18. Refine belt 2 pressed
19. Initialise belt 2 refinement GUI Part A
20. Done PB pressed
21. Initialise belt 2 refinement GUI Part B
22. Done PB pressed
23. Initialise gear train development GUI
24. Refine BGU pressed
25. Initialise BGU GUI
26. Done button pressed
27. Initialise gear train development GUI
28. Refine belt 1 pressed
29. Initialise belt 1 refinement GUI
30. Done PB pressed
31. Initialise gear train development GUI
32. Calculate button pressed
33. FIN

```
%AGV Designer V1.1
%8888888888888888888888888888888888888888888888888888888888888888888888888
%Swerve Drive AGV Design Calculator
%Developed by Alex Macfarlane





%8888888888888888888888888888888888888888888888888888888888888888888888888
%Initialise Program
%8888888888888888888888888888888888888888888888888888888888888888888888888
%clear memory and console
clc;
clear;
display('Start Program');





%8888888888888888888888888888888888888888888888888888888888888888888888888
%Open Start GUI
%8888888888888888888888888888888888888888888888888888888888888888888888888

%   Run GUI1
%   *********************************************************************
%   Write status to command window
    display('Initilaise Start GUI (GUI1)');

    StartGUI;

%   Write status to command window
    display('Start GUI (GUI1) Exited');





%8888888888888888888888888888888888888888888888888888888888888888888888888
%Global Variables
%8888888888888888888888888888888888888888888888888888888888888888888888888
%These variables must be accesable to all the GUIs

    %Check if matlab file version exists of raw data
    %********************************************************************

        %cheack if matlab file containing all variables exists
        MatFileExist = exist('RawData.mat');

        %set up trigger to force read of excel files
        NoMatlabFile = 0;

        if MatFileExist == 2
            %file called 'RawData.mat' exists
            %load data from 'RawData.mat' to workspace
            load('RawData.mat');

            %prevent read from raw ecel files
            NoMatlabFile = 1;

        end

    %Read from excel files ONLY IF MATLAB FILE NOT FOUND
    %*********************************************************************

        if NoMatlabFile ==  0
            %read from excel files

            %{
            %Check if value already read from previous run
            valueExists = zeros(1,4);
            valueExists(1,1) = exist('HTD5M');
            valueExists(1,2) = exist('HTD8M');
            valueExists(1,3) = exist('HTD14M');
            valueExists(1,4) = exist('ROBGU');
            ForceRead = 1;  %set value to zero to force read every time

            %Choose what to do if run
            if (valueExists(1,1)*valueExists(1,2)*valueExists(1,3)*valueExists(1,4)↙
*ForceRead) == 0
                %All data doesn't exist

                %display task being run
                display('Reading data from excel files');

                %Read all gear train data from xlsx files
                [ HTD5M, HTD8M, HTD14M, PowerConditions, ROBGU ] =↙
GearDataXLSXReader();

                %save to matlab variable for quick future access
                save('RawData.↙
mat','HTD5M','HTD8M','HTD14M','PowerConditions','ROBGU');

            end
            %}

            %display task being run
            display('Reading data from excel files');

            %Read all gear train data from xlsx files
            [ HTD5M, HTD8M, HTD14M, PowerConditions, ROBGU ] = GearDataXLSXReader();

            %save to matlab variable for quick future access
            save('RawData.mat','HTD5M','HTD8M','HTD14M','PowerConditions','ROBGU');

        end

    %Create list of avaliable gear types
    %********************************************************************

        %avaliable belt pitch vectors
        BeltsAvaliable = {'HTD5M' 'HTD8M' 'HTD14M'};


    %Set up conditions for gear train optimisation
    %*********************************************************************

        %Create empty gear train vector
        GearTrainVector = [0 0 0];
        %               = [Belt2_i BGU_i Belt1_i];

    %Set up status of gear train optimisation
    %*********************************************************************

        %If = 0 => gear ratio not set yet
        GearTrainStatus = 0;    %set to stage 1 (0)

    %Create list of avaliable Varvel bevel gear units
    %********************************************************************

        %avaliable belt pitch vectors
        BGUAvaliable = {'RO02' 'RO12' 'RO22' 'RO32'};





%8888888888888888888888888888888888888888888888888888888888888888888888888
%Start GUI Loop
%8888888888888888888888888888888888888888888888888888888888888888888888888

RunThisGUI = 1; %run GUI2 initially
LoopGUI = 1;    %loop through results until GUI breaks loop

while LoopGUI == 1
%This loops through the results GUIs until the user decides to move onto
%the next part of the program

    switch RunThisGUI

        case 1

%            8888888888888888888888888888888888888888888888888888888888888
%            Collect Initial Data
%            8888888888888888888888888888888888888888888888888888888888888

%               Write status to command window
                display('Initilaise Initial Data Collection GUI (GUI2)');

%               Run GUI2
```

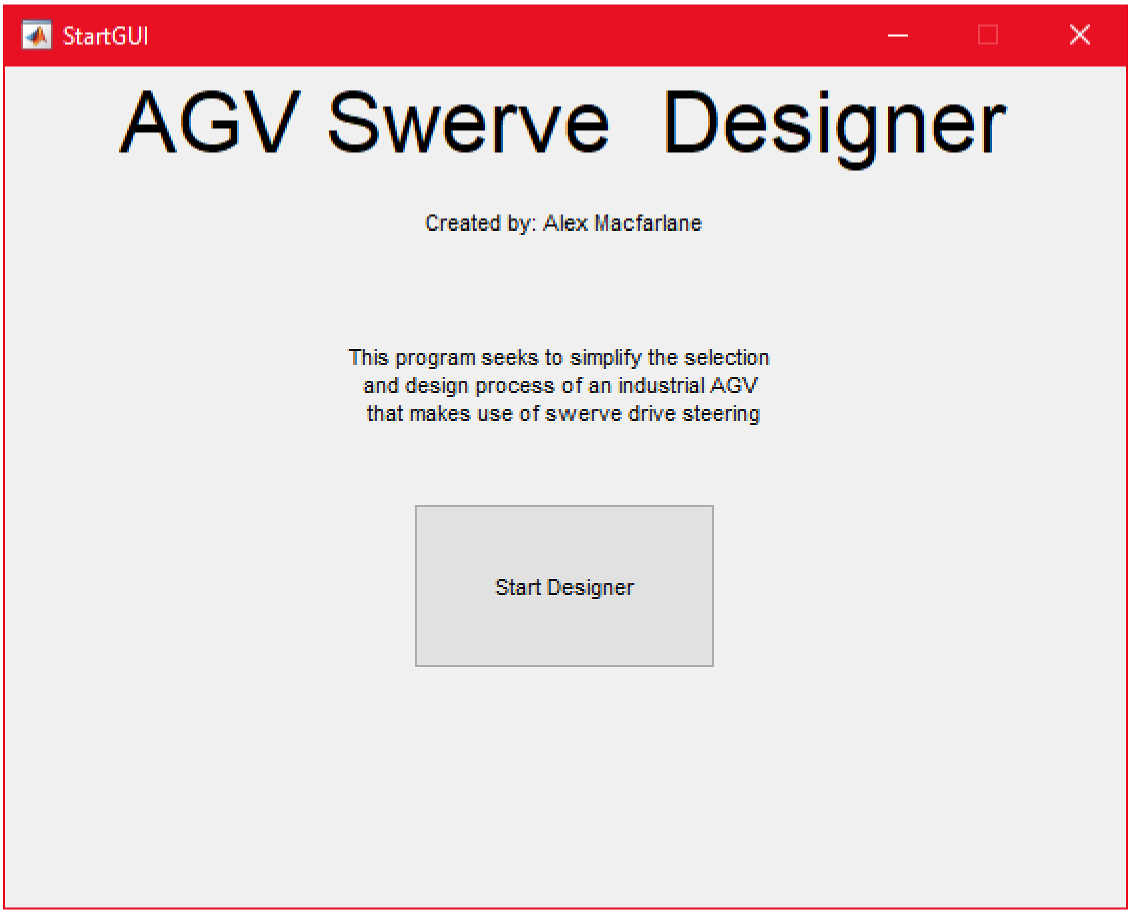
```
                [Mass, Speed, AccTime, MaxIncline, AllowAccInc, WheelRad, u,↙
NoMotors_Full, NoMotors_Eco, RunThisGUI, LoopGUI] = CollectInitialDataGUI;

%               Drive system efficiency -hardcoded- (value between 0 and 1)
                DriveEff = 0.98;

%               Write status to command window
                display('Initial Data Collection GUI (GUI2) Exited');





%            888888888888888888888888888888888888888888888888888888888888
%            Calculate Max Required Forces on AGV (Whole Vehicle)
%            888888888888888888888888888888888888888888888888888888888888

%               There are four conditions that the AGV will have to contend
%               with,namely:
%               1) constant velocity on a flat surface     (CF)
%               2) acceleration on a flat surface          (AF)
%               3) constant velocity up and incline        (CI)
%               4) acceleration up an incline              (AI)

%               ForceVec   = returns a 1x4 vector of forces required to
%                            move the AGV under the CF, AF, CI and AI
%                            conditions
%                          = [ CF, AF, CI, AI ]

%               Call "AGVRequiredForceCal" function
                [ForceVec] = AGVRequiredForceCal(Mass, u, MaxIncline, Speed, AccTime);





%            888888888888888888888888888888888888888888888888888888888888
%            Calculate Max Required Angular Velocity of Wheels
%            (Whole Vehicle)
%            888888888888888888888888888888888888888888888888888888888888

%               Call "AGVWheelAngVelCal" function
                [WheelOmega] = AGVWheelAngVelCal(Speed, WheelRad);





%            888888888888888888888888888888888888888888888888888888888888
%            Calculate Max Required Torque
%            888888888888888888888888888888888888888888888888888888888888

%               Calculates torque required for the following:
%               1) (WTotT) : total AGV torque required wheel side
%               2) (WFulT) : wheel torque for full drive motor compliment
%                            wheel side
%                3) (WEcoT) : wheel torque for eco drive motor compliment
%                            wheel side
%

%                TorqueVecWheel = returns a 3x4 vector with the WTotT
%                                torques for each force condition in row
%                                1,the WFulT torques for each force
%                                condition in row 2 and the WEcoT torques
%                                for each force condition in row 3 on the
%                                wheel side of the gearing.

%                              = [ WTotT_CF, WTotT_AF, WTotT_CI, WTotT_AI ]
%                                [ WFulT_CF, WFulT_AF, WFulT_CI, WFulT_AI ]
%                                [ WEcoT_CF, WEcoT_AF, WEcoT_CI, WEcoT_AI ]

%                Call "AGVTorqueCal" function
                 [TorqueVecWheel] = AGVTorqueCal(ForceVec, WheelRad, NoMotors_Full,↙
NoMotors_Eco);




%             8888888888888888888888888888888888888888888888888888888888888
%             Calculate Max Required Power
%             8888888888888888888888888888888888888888888888888888888888888

%                Calculates power for the following:
%                1) (WTotT) : total AGV power required wheel side
%                2) (WFulT) : wheel power required for full drive motor
%                            compliment
%                3) (WEcoT) : wheel power required for eco drive motor
%                            compliment
%                4) (MTotT) : total AGV power required motor side
%                5) (MFulT) : motor power required for full drive motor
%                            compliment
%                6) (MEcoT) : motor power required for eco drive motor
%                            compliment

%                PowerVecWheel  = returns a 3x4 vector with the WTotP power
%                                for each force condition in row 1, the
%                                WFulP power for each force condition in
%                                row 2 and the WEcoP power for each force
%                                condition in row 3

%                              = [ WTotP_CF, WTotP_AF, WTotP_CI, WTotP_AI ]
%                                [ WFulP_CF, WFulP_AF, WFulP_CI, WFulP_AI ]
%                                [ WEcoP_CF, WEcoP_AF, WEcoP_CI, WEcoP_AI ]

%                PowerVecMotor  = returns a 3x4 vector with the MTotP power
%                                for each force condition in row 1, the
%                                MFulP power for each force condition in
%                                row 2 and the MEcoP power for each force
%                                condition in row 3
%                              = [ MTotP_CF, MTotP_AF, MTotP_CI, mTotP_AI ]
%                                [ MFulP_CF, MFulP_AF, MFulP_CI, MFulP_AI ]
%                                [ MEcoP_CF, MEcoP_AF, MEcoP_CI, MEcoP_AI ]

%               Call "AGVPowerCal" function
                [PowerVecWheel, PowerVecMotor] = AGVPowerCal(TorqueVecWheel,↙
WheelOmega, DriveEff);




%            8888888888888888888888888888888888888888888888888888888888888
%            Determine Motor Power Reqmts and Conditions for Eco Motor
%            8888888888888888888888888888888888888888888888888888888888888

%               The AGV drive motor power will be specified according to
%               the largest power value found in the full motor compliment
%               conditions if the AGV is allowed to accelerate up and
%               incline. If the AGV is not allowed to accelerate up an
%               incline the drive motor power required will be found in the
%               full motor compliment conditions minus the power required
%               to accelerate up an incline value.

%               The ECO motor compliment system (when not all drives are
%               used to save power) CAN ONLY BE USED when the power
%               required by the eco motors is less than the maximum drive
%               motor power specification described in the paragraph above

%               Call "AGVMaxPowerReq" function
                [MaxWheelPower, MaxPowerReq, MaxWheelTorque ,EcoUseLimits] =↙
AGVMaxPowerReq(PowerVecMotor, PowerVecWheel, TorqueVecWheel ,AllowAccInc);




        case 2

%            8888888888888888888888888888888888888888888888888888888888888
%            Display Forces Acting on AGV Results
%            8888888888888888888888888888888888888888888888888888888888888

                %Write status to command window
                display('Initialise drive force results GUI (GUI3)');

                %Run GUI3
                [RunThisGUI, LoopGUI] = DriveForceResultsGUI(AllowAccInc, ForceVec,↙
MaxIncline);


                %RunThisGUI = 0;
        case 3

%           8888888888888888888888888888888888888888888888888888888888888
%           Display Torque and Power Reqmts of Each Drive Wheel
%           8888888888888888888888888888888888888888888888888888888888888

                %Write status to command window
                display('Initialise wheel results GUI (GUI4)');

                %Run GUI4
                [RunThisGUI, LoopGUI] = WheelPandTResultsGUI(MaxWheelPower,↙
WheelOmega, TorqueVecWheel, PowerVecWheel, AllowAccInc, EcoUseLimits);




        case 4

%           8888888888888888888888888888888888888888888888888888888888888
%           Display Each Drive Motor Power Requirements
%           8888888888888888888888888888888888888888888888888888888888888

                %Write status to command window
                display('Initialise motor requirements GUI (GUI5)');

                %Run GUI5
                [RunThisGUI, LoopGUI] = MotorPResultsGUI(MaxPowerReq, DriveEff,↙
PowerVecMotor, EcoUseLimits);




        case 5

%           8888888888888888888888888888888888888888888888888888888888888
%           Optain Information on Drive Motor (RPM and Max Torque)
%           8888888888888888888888888888888888888888888888888888888888888

                %Write status to command window
                display('Initialise motor selection GUI (GUI6)');

                %Run GUI6
                [RunThisGUI, LoopGUI, MotorOmega, MotorRateTorque] = MotorSelectionGUI↙
(MaxPowerReq, MaxWheelPower, MaxWheelTorque, WheelOmega);



%           8888888888888888888888888888888888888888888888888888888888888
%           Calculate allowable gear ratio
%           8888888888888888888888888888888888888888888888888888888888888

                [GearRatio] = AGVGearRatioCal(WheelOmega, MotorOmega);
        case 6

%           8888888888888888888888888888888888888888888888888888888888888
%           Get System Conditions of Operation
%           8888888888888888888888888888888888888888888888888888888888888

                %Write status to command window
                display('Initialise system conditions GUI');

                %Run GUI7a
                [RunThisGUI, LoopGUI, ConditionsVec] = OperationalConditionsGUI↙
(PowerConditions);



        case 7

%           8888888888888888888888888888888888888888888888888888888888888
%           Find RPMs and Torques of each stage in the gear train
%           8888888888888888888888888888888888888888888888888888888888888

                %Call stage RPM & Torque Calculator
                [GTrpm , GTtor] = AGVGearTrainRatioSolver(GearTrainVector, WheelOmega,↙
MaxWheelTorque);




%           8888888888888888888888888888888888888888888888888888888888888
%           Run gear train development GUI
%           8888888888888888888888888888888888888888888888888888888888888

                %Write status to command window
                display('Initialise gear train development GUI');

                %Run GUI7b
                [RunThisGUI, LoopGUI] = GearDevelopmentGUI(GearRatio, GearTrainVector,↙
GTrpm, GTtor, GearTrainStatus, MotorOmega, MotorRateTorque);



        case 8

%           8888888888888888888888888888888888888888888888888888888888888
%           Refine Belt 2 Part A
%           8888888888888888888888888888888888888888888888888888888888888
```

```
                %Write status to command window
                display('Initialise belt 2 refinement GUI Part A');

                %Check if Belt2AResultsVector already exists
                if GearTrainStatus >= 10
                    %This has already been run at least once

                    DataExists = 1;
                else
                    %This has not been run at all

                    DataExists = 0;
                end

                %Run GUI8
                [RunThisGUI, LoopGUI, Temp] = Belt2AOptimisationGUI(HTD5M, HTD8M, ↙
HTD14M, BeltsAvaliable, WheelOmega, MaxPowerReq, ConditionsVec, PowerConditions, ↙
DataExists);

                %Belt2AResultsVector = Temp = [ BeltPitch,
                %                               DesiredBeltWidth,
                %                               GearRatio,
                %                               DrivePulleyTeeth,
                %                               DrivePulleyDia,
                %                               DrivenPulleyTeeth,
                %                               DrivenPulleyDia,
                %                               X1,
                %                               Y1,
                %                               X2,
                %                               Y2,
                %                               BeltTeeth,
                %                               BeltLength          ]


                if  Temp ~= 0
                    %Data was entered => update (i.e. was not forced to
                    %zero

                    Belt2AResultsVector = Temp;

                    %Update gear train development stage
                    GearTrainStatus = 10;

                    %Only update GearTrainStatus if belt 2 done previously
                    if GearTrainStatus >= 20

                        %Update Gear Train vector
                        GearTrainVector(1,1) = Belt2AResultsVector(1,3)↙
*Belt2BResultsVector(1,3);
                        %                    = [Belt2_i BGU_i Belt1_i];

                    end

                end
```

```
        case 9

%           8888888888888888888888888888888888888888888888888888888888888
%           Refine Belt 2 Part B
%           8888888888888888888888888888888888888888888888888888888888888

                %Write status to command window
                display('Initialise belt 2 refinement GUI Part B');

                %Calculate output RPM for second stage (rad/sec)
                Belt2BOmega = WheelOmega*Belt2AResultsVector(1,3);

                %Check if Belt2AResultsVector already exists
                if GearTrainStatus >= 20
                    %This has already been run at least once

                    DataExists = 1;
                else
                    %This has not been run at all

                    DataExists = 0;
                end

                %Run GUI9
                [RunThisGUI, LoopGUI, Temp] = Belt2BOptimisationGUI(HTD5M, HTD8M, ↙
HTD14M, BeltsAvaliable, Belt2BOmega, MaxPowerReq, ConditionsVec, PowerConditions, ↙
DataExists);

                if Temp ~= 0
                    %Data was entered => update (i.e. was not forced to
                    %zero

                    Belt2BResultsVector = Temp;

                    %Update gear train development stage
                    GearTrainStatus = 20;

                    %Update Gear Train vector
                    GearTrainVector(1,1) = Belt2AResultsVector(1,3)↙
*Belt2BResultsVector(1,3);
                    %                    = [Belt2_i BGU_i Belt1_i];

                end




        case 10

%           8888888888888888888888888888888888888888888888888888888888888
%           Refine BGU
%           8888888888888888888888888888888888888888888888888888888888888
```

```
                %Write status to command window
                display('Initialise BGU GUI');

                %Calculate output angular velocity for third stage (rad/sec)
                BGUOmega = Belt2BOmega*Belt2BResultsVector(1,3);

                %Find BGU output torque (input torque for Belt2 system)
                BGUOutTorque = GTtor(1,2);

                %Run GUI10
                [RunThisGUI, LoopGUI, BGUResultsVector] = BGUOptimisationGUI(ROBGU, ↙
BGUAvaliable, BGUOutTorque , MaxPowerReq);

                %BGUResultsVector = [   BGUUnitIndex,
                %                       GearRatioGiven,
                %                       GearRatioActual,
                %                       OuterShaftDia,
                %                       InputShaftDia,
                %                       InputShaftRadForce,
                %                       OutputShaftRadForce,
                %                       OutputShaftThrstForce   ];

                %Update gear train vector
                GearTrainVector(1,2) = BGUResultsVector(1,3);
                %                    = [Belt2_i BGU_i Belt1_i];

                %Update gear train development stage
                GearTrainStatus = 30;




        case 11

%           88888888888888888888888888888888888888888888888888888888888888
%           Refine Belt 1
%           88888888888888888888888888888888888888888888888888888888888888

                %Write status to command window
                display('Initialise belt 1 refinement GUI');

                %Calculate output angular velocity for 4th stage (rad/sec)
                Belt1Omega = BGUOmega*BGUResultsVector(1,3);

                %Run GUI11
                [RunThisGUI, LoopGUI, Belt1ResultsVector] = Belt1OptimisationGUI↙
(HTD5M, HTD8M, HTD14M, BeltsAvaliable, Belt1Omega, MaxPowerReq, ConditionsVec, ↙
PowerConditions);

                %Belt1ResultsVector = [ BeltPitch,
                %                       DesiredBeltWidth,
                %                       GearRatio,
                %                       DrivePulleyTeeth,
                %                       DrivePulleyDia,
                %                       DrivenPulleyTeeth,
                %                       DrivenPulleyDia,
                %                       X1,
                %                       Y1,
                %                       X2,
                %                       Y2,
                %                       BeltTeeth,
                %                       BeltLength          ]


                %Update gear train vector
                GearTrainVector(1,3) = Belt1ResultsVector(1,3);

                %Update gear train development stage
                GearTrainStatus = 40;

        case 12

                %Call Belt
                LoopGUI = 0;

    end
end

        %}


%   Run GUI3
%   *********************************************************************









    %display('Preliminary Drive Results GUI (GUI3) Exited');





display('FIN');



%call initial GUI
```

Start GUI:



StartGUI

AGV Swerve Designer

Created by: Alex Macfarlane

This program seeks to simplify the selection
and design process of an industrial AGV
that makes use of swerve drive steering

```
function varargout = StartGUI(varargin)
% STARTGUI M-file for StartGUI.fig
%      STARTGUI, by itself, creates a new STARTGUI or raises the existing
%      singleton*.
%
%      H = STARTGUI returns the handle to a new STARTGUI or the handle to
%      the existing singleton*.
%
%      STARTGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in STARTGUI.M with the given input arguments.
%
%      STARTGUI('Property','Value',...) creates a new STARTGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before StartGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to StartGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help StartGUI

% Last Modified by GUIDE v2.5 11-Aug-2016 15:47:21

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @StartGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @StartGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before StartGUI is made visible.
function StartGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to StartGUI (see VARARGIN)

% Choose default command line output for StartGUI
```

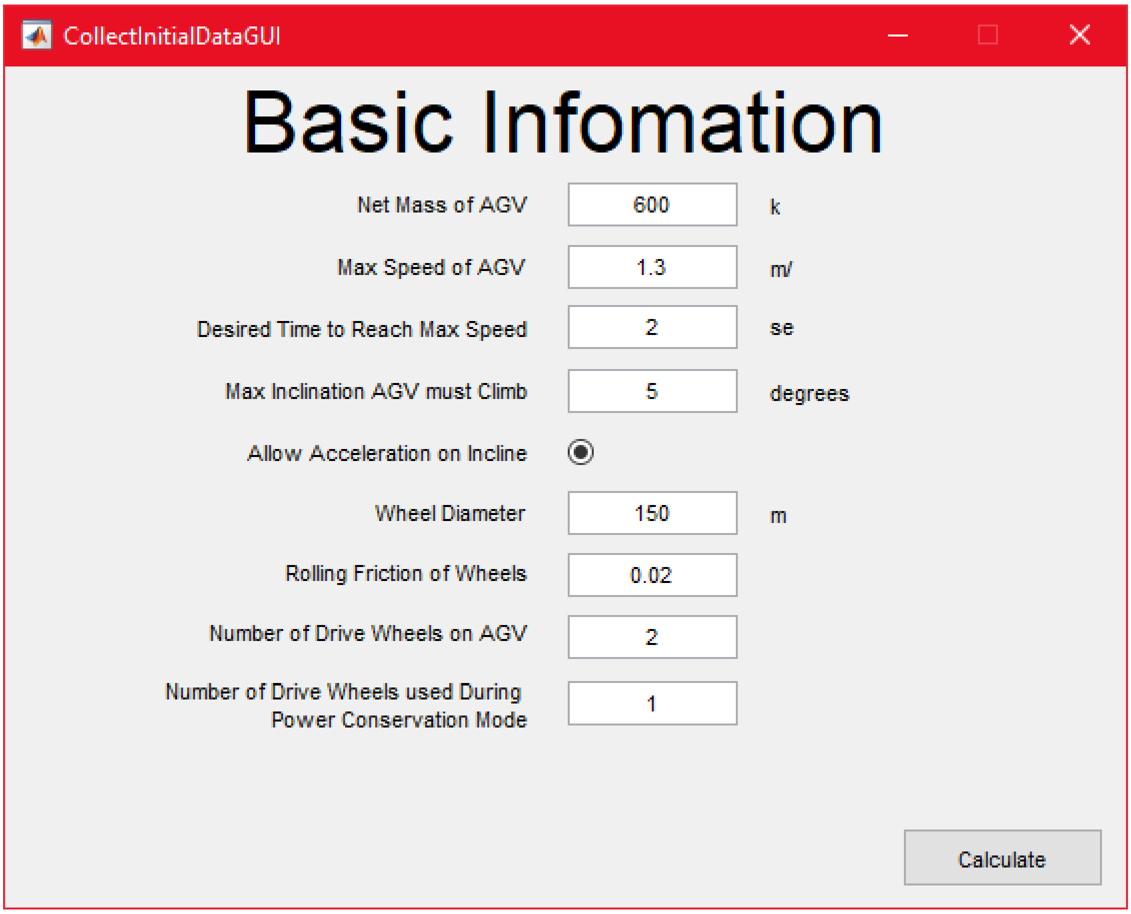
```
handles.output = hObject;
handles.mass = 0;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes StartGUI wait for user response (see UIRESUME)
uiwait(handles.GUI1);


% --- Outputs from this function are returned to the command line.
function varargout = StartGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

%The figure can now be deleted
delete(handles.GUI1);


% --- Executes on button press in StartDesigner.
function StartDesigner_Callback(hObject, eventdata, handles)
% hObject    handle to StartDesigner (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%SC = true;
%setappdata(0,'StartComplete',SC);

close;


% --- Executes when user attempts to close GUI1.
function GUI1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to GUI1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if isequal(get(hObject, 'waitstatus'),'waiting')
    %The GUI is still in UIWAIT, use UIRESUME
    uiresume(hObject);
else
    %The GUI is nolonger waiting and can be closed
    delete(hObject);
end
```

```
function [ HTD5M, HTD8M, HTD14M, PowerConditions, ROBGU ] = GearDataXLSXReader( ~ )
%GearDataXLSXReader
%888888888888888888888888888888888888888888888888888888888888888888888888
%   This function reads relevent data for the belt gear trains and bevel
%   gear boxes from the appropriate xlsx files

%   Outputs
%   *********************************************************************

%   HTD5M                   = structure of HTD5M's data
%   HTD8M                   = structure of HTD8M's data
%   HTD14M                  = structure of HTD14M's data
%   PowerConditions         = power conditions table

%888888888888888888888888888888888888888888888888888888888888888888888888
%Get list data
%888888888888888888888888888888888888888888888888888888888888888888888888

%   Read in HTD 5M data
%   *********************************************************************

    tempA = xlsread('BeltLookUpTables.xlsx','HTD_5M_Belt');
    display('read HTD_5M_Belt');
    tempB = xlsread('BeltLookUpTables.xlsx','HTD_5M_Pulley');
    display('read HTD_5M_Pulley');
    tempC = xlsread('BeltLookUpTables.xlsx','5M_Power');
    display('read 5M_Power');
    tempD = xlsread('BeltLookUpTables.xlsx','5M_PowerMultipliers');
    display('read 5M_PowerMultipliers');
    tempE = xlsread('BeltLookUpTables.xlsx','5M_BeltLengthMultiplier');
    display('read 5M_BeltLengthMultiplier');

    HTD5M = struct('BeltCenter',tempA,'Pulley',tempB,'Power',tempC,'PowerMult',↙
tempD,'LengthMult',tempE);

%   Read in HTD 8M data
%   *********************************************************************

    tempA = xlsread('BeltLookUpTables.xlsx','HTD_8M_Belt');
    display('read HTD_8M_Belt');
    tempB = xlsread('BeltLookUpTables.xlsx','HTD_8M_Pulley');
    display('read HTD_8M_Pulley');
    tempC = xlsread('BeltLookUpTables.xlsx','8M_Power');
    display('read 8M_Power');
    tempD = xlsread('BeltLookUpTables.xlsx','8M_PowerMultipliers');
    display('read 8M_PowerMultipliers');
    tempE = xlsread('BeltLookUpTables.xlsx','8M_BeltLengthMultiplier');
    display('read 8M_BeltLengthMultiplier');

    HTD8M = struct('BeltCenter',tempA,'Pulley',tempB,'Power',tempC,'PowerMult',↙
tempD,'LengthMult',tempE);

%   Read in HTD 14M data
%   *********************************************************************
```

```
    tempA = xlsread('BeltLookUpTables.xlsx','HTD_14M_Belt');
    display('read HTD_14M_Belt');
    tempB = xlsread('BeltLookUpTables.xlsx','HTD_14M_Pulley');
    display('read HTD_14M_Pulley');
    tempC = xlsread('BeltLookUpTables.xlsx','14M_Power');
    display('read 14M_Power');
    tempD = xlsread('BeltLookUpTables.xlsx','14M_PowerMultipliers');
    display('read 14M_PowerMultipliers');
    tempE = xlsread('BeltLookUpTables.xlsx','14M_BeltLengthMultiplier');
    display('read 14M_BeltLengthMultiplier');

    HTD14M = struct('BeltCenter',tempA,'Pulley',tempB,'Power',tempC,'PowerMult',↙
tempD,'LengthMult',tempE);

%   Read in power condition multiplier table
%   **********************************************************************

    PowerConditions = xlsread('BeltLookUpTables.xlsx','PowerConditions');
    display('read PowerConditions');

%   Read in Varvel bevel gear unit data
%   **********************************************************************

    tempA = xlsread('BGULookUpTables.xlsx','RO02_Ratios');
    display('read RO02_Ratios');
    tempB = xlsread('BGULookUpTables.xlsx','RO12_Ratios');
    display('read RO02_Ratios');
    tempC = xlsread('BGULookUpTables.xlsx','RO22_Ratios');
    display('read RO22_Ratios');
    tempD = xlsread('BGULookUpTables.xlsx','RO32_Ratios');
    display('read RO32_Ratios');

    ROBGU = struct('RO02', tempA, 'RO12', tempB, 'RO22', tempC, 'RO32', tempD);

end
```

G.2 Initial Data Collection GUI

Initial Data Collection GUI:



Basic Infomation

Net Mass of AGV	600	k
Max Speed of AGV	1.3	m/
Desired Time to Reach Max Speed	2	se
Max Inclination AGV must Climb	5	degrees
Allow Acceleration on Incline	●	
Wheel Diameter	150	m
Rolling Friction of Wheels	0.02	
Number of Drive Wheels on AGV	2	
Number of Drive Wheels used During Power Conservation Mode	1	

```
function varargout = CollectInitialDataGUI(varargin)
%varargout
% GUI2 M-file for GUI2.fig
%      GUI2, by itself, creates a new GUI2 or raises the existing
%      singleton*.
%
%      H = GUI2 returns the handle to a new GUI2 or the handle to
%      the existing singleton*.
%
%      GUI2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI2.M with the given input arguments.
%
%      GUI2('Property','Value',...) creates a new GUI2 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before CollectInitialDataGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to CollectInitialDataGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI2

% Last Modified by GUIDE v2.5 15-Aug-2016 16:39:49

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @CollectInitialDataGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @CollectInitialDataGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT




%88888888888888888888888888888888888888888888888888888888888888888888888888
%Initialise
%88888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes just before GUI2 is made visible.
```

```
function CollectInitialDataGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GUI2 (see VARARGIN)

% Choose default command line output for GUI2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% Setup initial conditions
set(handles.MassTB,'String','600');
set(handles.SpeedTB,'String','1.3');
set(handles.AccTimeTB,'String','2');
set(handles.InclineTB,'String','5');
set(handles.WheelDiaTB,'String','150');
set(handles.RollingFricTB,'String','0.02');
set(handles.NoDriveTB,'String','2');
set(handles.NoEcoTB,'String','1');

% Diable error message
set(handles.ErrorMessage,'Visible','off');

% UIWAIT makes GUI2 wait for user response (see UIRESUME)
uiwait(handles.GUI2);




%88888888888888888888888888888888888888888888888888888888888888888888888
%Output Function
%88888888888888888888888888888888888888888888888888888888888888888888888

% --- Outputs from this function are returned to the command line.
function varargout = CollectInitialDataGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Set outputs
%***********************************************************************

    %Net mass of AGV (kg)
    varargout{1} = str2double(get(handles.MassTB,'String'));

    %Top speed of the AGV (m/s)
    varargout{2} = str2double(get(handles.SpeedTB,'String'));

    %Time taken to reach top speed (sec)
    varargout{3} = str2double(get(handles.AccTimeTB,'String'));
    %Max angle of inclination of floor (degrees)
    varargout{4} = str2double(get(handles.InclineTB,'String'));

    %Allow AGV to accelerate on an incline (True or False)
    varargout{5} = get(handles.AllowAccRB,'Value');

    %Radius of driving wheels (m)
    varargout{6} = ((str2double(get(handles.WheelDiaTB,'String')))/1000)/2;

    %Rolling friction co-efficient of wheels (~)
    varargout{7} = str2double(get(handles.RollingFricTB,'String'));

    %number of drive wheels (#)
    varargout{8} = str2double(get(handles.NoDriveTB,'String'));

    %number of wheels used during ECO mode (#)
    varargout{9} = str2double(get(handles.NoEcoTB,'String'));

 %Get navigation data stored in pushbuttons
%************************************************************************

    %Collect data for Next PB
    hpp1 = get(handles.CalculatePB,'UserData');

    %Collect data for End PB
    hpp2 = 0; %get(handles.End_PB,'UserData');

    %Collect data for Previous PB
    hpp3 = 0; %get(handles.Prev_PB,'UserData');

%Determine which PB was pressed and update outputs
%(navigation data non-zero)
%************************************************************************

    %varargout{10} = Next GUI to be run
    %varargout{11} = (if = 1)=> Keep Looping through results GUI else don't

    if hpp1 ~= 0
        %Next PB was pressed to close GUI
        display('Next button pressed');
        varargout{10} = 2;
        varargout{11} = 1;

    elseif hpp2 ~= 0
        %End PB was pressed to close GUI
        display('End button pressed');
        varargout{10} = 1;
        varargout{11} = 1;

    elseif hpp3 ~= 0
        %Previous PB was pressd to close GUI
        display('Previous button pressed');
        varargout{10} = 2;
        varargout{11} = 1;
    else
        %Unknown PB was pressed or user terminated
        display('Error: program closed unexpectedly')
    end

%Delete GUI
%************************************************************************

    %The figure can now be deleted
    delete(handles.GUI2);






%8888888888888888888888888888888888888888888888888888888888888888888888888
%Prev, Next, Home Pushbuttons
%888888888888888888888888888888888888888888888888888888888888888888888888

%Empty






%8888888888888888888888888888888888888888888888888888888888888888888888888
%Create Textboxes and Objects
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes during object creation, after setting all properties.
function MassTB_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MassTB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function SpeedTB_CreateFcn(hObject, eventdata, handles)
% hObject    handle to SpeedTB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes during object creation, after setting all properties.
function AccTimeTB_CreateFcn(hObject, eventdata, handles)
% hObject    handle to AccTimeTB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function InclineTB_CreateFcn(hObject, eventdata, handles)
% hObject    handle to InclineTB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function WheelDiaTB_CreateFcn(hObject, eventdata, handles)
% hObject    handle to WheelDiaTB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function RollingFricTB_CreateFcn(hObject, eventdata, handles)
% hObject    handle to RollingFricTB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function NoDriveTB_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to NoDriveTB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function NoEcoTB_CreateFcn(hObject, eventdata, handles)
% hObject    handle to NoEcoTB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function ErrorWindow_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ErrorWindow (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end





%8888888888888888888888888888888888888888888888888888888888888888888888888
%UI Code Interaction Code(Pushbuttons, Radio Buttons, etc.)
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes on button press in CalculatePB.
function CalculatePB_Callback(hObject, eventdata, handles)
% hObject    handle to CalculatePB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Initialise values
%************************************************************************

    %When stop = false GUI can close + calculations can start, else =
    %mistake in input thus let user change input (initialised to false)
    stop = false;

    %Diable error message
    set(handles.ErrorMessage,'Visible','off');

%Mass input processing
%************************************************************************

    %copy mass to variable "holdstring"
    holdstring = get(handles.MassTB,'String');

    %check if each character in "holdstring" is a digits and store result
    %as vector "test1"
    test1 = isstrprop(holdstring,'digit');

    %check if any charaters in "holdstring" are decimal points and store
    %results as vector "test2"
    test2 = zeros(1,length(holdstring));    %initialise test2 as zeros

    for i = 1:length(test1)
        %for each character in "holdstring"

        if holdstring(1,i) == '.'
            %check if character is a decimal point

            test2(1,i) = 1;

        end
    end

    %add "test1" and "test2", resultant vector will contain a 1 for any
    %character that is either a decimal point or digit and 0 if it is not.
    %Eg [1 1 1 1] = valid number
    %   [1 1 0 1] = string input contains one non digit/decimal point,
    %               invalid number
    test = test1+test2;

    %if product of test is zero = invalid number else number is valid
    isDigit = prod([1,test]);

    %check if number is valid and non-zero
    if (str2double(get(handles.MassTB,'String')) == 0) || (isDigit == 0)
        %input invalid or zero

        %change textbox colour to red
        set(handles.MassTB,'BackgroundColor',[0.85 0.19 0]);

        %stop = true = GUI not allowed to close
        stop = true;

    else
        %input valid

        %reset backround color to white
        set(handles.MassTB,'BackgroundColor',[1 1 1]);

        %stop = false (initialised conditon not changed) = GUI can close

    end

%Max speed input processing
%*************************************************************************

    %copy max speed to variable "holdstring"
    holdstring = get(handles.SpeedTB,'String');

    %check if each character in "holdstring" is a digits and store result
    %as vector "test1"
    test1 = isstrprop(holdstring,'digit');

    %check if any charaters in "holdstring" are decimal points and store
    %results as vector "test2"
    test2 = zeros(1,length(holdstring));

    for i = 1:length(test1)
        %for each character in "holdstring"

        if holdstring(1,i) == '.'
            %check if charater is a decimal point

            test2(1,i) = 1;

        end
    end

    %add "test1" and "test2", resultant vector will contain a 1 for any
    %character that is either a decimal point or digit and 0 if it is not.
    %Eg [1 1 1 1] = valid number
    %   [1 1 0 1] = string input contains one non digit/decimal point,
    %               invalid number
    test = test1+test2;

    %if product of test is zero = invalid number else number is valid
    isDigit = prod([1,test]);

    %check if number is valid and non-zero
    if (str2double(get(handles.SpeedTB,'String')) == 0) || (isDigit == 0)
        %input invalid or zero

        %change textbox colour to red
        set(handles.SpeedTB,'BackgroundColor',[0.85 0.19 0]);

        %stop = true = GUI not allowed to close
        stop = true;

    else
        %input valid

        %reset backround color to white
        set(handles.SpeedTB,'BackgroundColor',[1 1 1]);

        %stop = false (initialised conditon not changed) = GUI can close

    end

%Acceleration time input processing
%************************************************************************

    %copy acceleration time to variable "holdstring"
    holdstring = get(handles.AccTimeTB,'String');

    %check if each character in "holdstring" is a digits and store result
    %as vector "test1"
    test1 = isstrprop(holdstring,'digit');

    %check if any charaters in "holdstring" are decimal points and store
    %results as vector "test2"
    test2 = zeros(1,length(holdstring));

    for i = 1:length(test1)
        %for each character in "holdstring"

        if holdstring(1,i) == '.'
            %check if charater is a decimal point

            test2(1,i) = 1;

        end
    end

    %add "test1" and "test2", resultant vector will contain a 1 for any
    %character that is either a decimal point or digit and 0 if it is not.
    %Eg [1 1 1 1] = valid number
    %   [1 1 0 1] = string input contains one non digit/decimal point,
    %               invalid number
    test = test1+test2;

    %if product of test is zero = invalid number else number is valid
    isDigit = prod([1,test]);

    %check if number is valid and non-zero
    if (str2double(get(handles.AccTimeTB,'String')) == 0) || (isDigit == 0)
        %input invalid or zero

        %change textbox colour to red
        set(handles.AccTimeTB,'BackgroundColor',[0.85 0.19 0]);

        %stop = true = GUI not allowed to close
        stop = true;

    else
        %input valid

        %reset backround color to white
        set(handles.AccTimeTB,'BackgroundColor',[1 1 1]);

        %stop = false (initialised conditon not changed) = GUI can close

    end

%Incline angle input processing
%************************************************************************

    %copy incline angle to variable "holdstring"
    holdstring = get(handles.InclineTB,'String');

    %check if each character in "holdstring" is a digits and store result
    %as vector "test1"
    test1 = isstrprop(holdstring,'digit');

    %check if any charaters in "holdstring" are decimal points and store
    %results as vector "test2"
    test2 = zeros(1,length(holdstring));

    for i = 1:length(test1)
        %for each character in "holdstring"

        if holdstring(1,i) == '.'
            %check if charater is a decimal point

            test2(1,i) = 1;

        end
    end

    %add "test1" and "test2", resultant vector will contain a 1 for any
    %character that is either a decimal point or digit and 0 if it is not.
    %Eg [1 1 1 1] = valid number
    %   [1 1 0 1] = string input contains one non digit/decimal point,
    %               invalid number
    test = test1+test2;

    %if product of test is zero = invalid number else number is valid
    isDigit = prod([1,test]);

    %check if number is valid (can be zero in this case)
    if (isDigit == 0)
       %input invalid

        %change textbox colour to red
        set(handles.InclineTB,'BackgroundColor',[0.85 0.19 0]);

        %stop = true = GUI not allowed to close
        stop = true;

    else
        %input valid

        %reset backround color to white
        set(handles.InclineTB,'BackgroundColor',[1 1 1]);

        %stop = false (initialised conditon not changed) = GUI can close

    end

%Wheel diameter input processing
%*************************************************************************

    %copy wheel diameter to variable "holdstring"
    holdstring = get(handles.WheelDiaTB,'String');

    %check if each character in "holdstring" is a digits and store result
    %as vector "test1"
    test1 = isstrprop(holdstring,'digit');

    %check if any charaters in "holdstring" are decimal points and store
    %results as vector "test2"
    test2 = zeros(1,length(holdstring));

    for i = 1:length(test1)
        %for each character in "holdstring"

        if holdstring(1,i) == '.'
            %check if charater is a decimal point

            test2(1,i) = 1;

        end
    end

    %add "test1" and "test2", resultant vector will contain a 1 for any
    %character that is either a decimal point or digit and 0 if it is not.
    %Eg [1 1 1 1] = valid number
    %   [1 1 0 1] = string input contains one non digit/decimal point,
    %               invalid number
    test = test1+test2;

    %if product of test is zero = invalid number else number is valid
    isDigit = prod([1,test]);

    %check if number is valid and non-zero
    if (str2double(get(handles.WheelDiaTB,'String')) == 0) || (isDigit == 0)
       %input invalid

        %change textbox colour to red
        set(handles.WheelDiaTB,'BackgroundColor',[0.85 0.19 0]);

        %stop = true = GUI not allowed to close
        stop = true;

    else
        %input valid

        %reset backround color to white
        set(handles.WheelDiaTB,'BackgroundColor',[1 1 1]);

        %stop = false (initialised conditon not changed) = GUI can close

    end

%Rolling friction input processing
%***********************************************************************

    %copy rolling friction to variable "holdstring"
    holdstring = get(handles.RollingFricTB,'String');

    %check if each character in "holdstring" is a digits and store result
    %as vector "test1"
    test1 = isstrprop(holdstring,'digit');

    %check if any charaters in "holdstring" are decimal points and store
    %results as vector "test2"
    test2 = zeros(1,length(holdstring));

    for i = 1:length(test1)
        %for each character in "holdstring"

        if holdstring(1,i) == '.'
            %check if charater is a decimal point

            test2(1,i) = 1;

        end
    end

    %add "test1" and "test2", resultant vector will contain a 1 for any
    %character that is either a decimal point or digit and 0 if it is not.
    %Eg [1 1 1 1] = valid number
    %   [1 1 0 1] = string input contains one non digit/decimal point,
    %               invalid number
    test = test1+test2;

    %if product of test is zero = invalid number, else number is valid
    isDigit = prod([1,test]);

    %check if number is valid, non-zero and between the values of 0 and 1
    if (str2double(get(handles.RollingFricTB,'String')) == 0) || (isDigit == 0) ||↙
(str2double(get(handles.RollingFricTB,'String')) >= 1)
       %input invalid

        %change textbox colour to red
        set(handles.RollingFricTB,'BackgroundColor',[0.85 0.19 0]);

        %stop = true = GUI not allowed to close
        stop = true;

    else
        %input valid
        %reset backround color to white
        set(handles.RollingFricTB,'BackgroundColor',[1 1 1]);

        %stop = false (initialised conditon not changed) = GUI can close

    end

%Number of drive wheels input (normal mode)
%***********************************************************************

    %check if each character in "holdstring" is a digits and store result
    %as vector "test"
    test = isstrprop(get(handles.NoDriveTB,'String'),'digit');

    %if product of test is zero = invalid number, else number is valid
    %(don't have to check for decimal places as number of wheels must be a
    %whole number)
    isDigit = prod([1,test]);

    %check if number is valid and non-zero
    if (str2double(get(handles.NoDriveTB,'String')) == 0) || (isDigit == 0)
       %input invalid

        %change textbox colour to red
        set(handles.NoDriveTB,'BackgroundColor',[0.85 0.19 0]);

        %stop = true = GUI not allowed to close
        stop = true;

    else
        %input valid

        %reset backround color to white
        set(handles.NoDriveTB,'BackgroundColor',[1 1 1]);

        %stop = false (initialised conditon not changed) = GUI can close

    end

%Number of Eco drive wheels (Eco made)
%***********************************************************************

    %check if each character in "holdstring" is a digits and store result
    %as vector "test"
    test = isstrprop(get(handles.NoEcoTB,'String'),'digit');

    %if product of test is zero = invalid number, else number is valid
    %(don't have to check for decimal places as number of wheels must be a
    %whole number)
    isDigit = prod([1,test]);

    %check if number is valid, non-zero and less than number of drive
    %wheels
    if (str2double(get(handles.NoEcoTB,'String')) == 0) || (isDigit == 0) ||↙
(str2double(get(handles.NoEcoTB,'String'))> str2double(get(handles.↙
NoDriveTB,'String')))
       %input invalid

        %change textbox colour to red
        set(handles.NoEcoTB,'BackgroundColor',[0.85 0.19 0]);

        %stop = true = GUI not allowed to close
        stop = true;

    else
        %input valid

        %reset backround color to white
        set(handles.NoEcoTB,'BackgroundColor',[1 1 1]);

        %stop = false (initialised conditon not changed) = GUI can close

    end

%Display or Reset Error Message and Pass Variables
%*************************************************************************

    %check if any inputs flagged "stop" by setting it to true
    if stop == true
        %One or more errors have occured = turn on error message
        set(handles.ErrorMessage,'Visible','on');
    else
        %No errors have occured = turn off error message
        set(handles.ErrorMessage,'Visible','off');

        %Set contitions to call next GUI
        set(handles.CalculatePB,'UserData',1);

        %Close GUI2
        close;
    end

function AllowAccRB_Callback(hObject, eventdata, handles)
% hObject    handle to AllowAccRB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

 function MassTB_Callback(hObject, eventdata, handles)
% hObject    handle to MassTB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function SpeedTB_Callback(hObject, eventdata, handles)
% hObject    handle to SpeedTB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function AccTimeTB_Callback(hObject, eventdata, handles)
% hObject    handle to AccTimeTB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function InclineTB_Callback(hObject, eventdata, handles)
% hObject    handle to InclineTB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function WheelDiaTB_Callback(hObject, eventdata, handles)
% hObject    handle to WheelDiaTB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function RollingFricTB_Callback(hObject, eventdata, handles)
% hObject    handle to RollingFricTB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function NoDriveTB_Callback(hObject, eventdata, handles)
% hObject    handle to NoDriveTB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function NoEcoTB_Callback(hObject, eventdata, handles)
% hObject    handle to NoEcoTB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)




%888888888888888888888888888888888888888888888888888888888888888888888888
%Close GUI
%888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes when user attempts to close GUI2.
function GUI2_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to GUI2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if isequal(get(hObject, 'waitstatus'),'waiting')
    %The GUI is still in UIWAIT, use UIRESUME
    uiresume(hObject);
else
    %The GUI is nolonger waiting and can be closed
    delete(hObject);
end
```

```
function [ ForceVec ] = AGVRequiredForceCal( m, u, zeta, vf, tDelta )
%AGVRequiredForceCal
%88888888888888888888888888888888888888888888888888888888888888888888888888

%   This function calculated the force required to move an AGV under the
%   following conditions:
%   1) constant velocity on a flat surface      (CF)
%   2) acceleration on a flat surface           (AF)
%   3) constant velocity up and incline         (CI)
%   4) acceleration up an incline               (AI)

    %Inputs
    %*********************************************************************

    %m      = mass of AGV                            (kg)
    %u      = coefficient of rolling friction        (~)
    %zeta   = angle of inclination of floor          (degrees)
    %vf     = desired top speed                      (m/s)
    %tDelta = time taken to reach top speed          (t)

    %Outputs
    %*********************************************************************

    %ForceVec = vetor containing the requird forces for each case

%Force Calculations
%888888888888888888888888888888888888888888888888888888888888888888888888

    %CASE 1: constant velocity on a flat surface (CF)
    %*********************************************************************
    %only force of friction is present

    %Frictional force on flat surface
    Ff_flat = u*(m*9.81);

    %Required force to transverse flat surface
    CF = Ff_flat;

    %CASE 2: acceleration on a flat surface (AF)
    %*********************************************************************
    %force of friction and force of acceleration are present

    %Force of acceleration
    Fa = m*(vf/tDelta);

    %Required force to accelerate on flat surface
    AF = Ff_flat + Fa;

    %CASE 3: constant velocity up and incline (CI)
    %*********************************************************************
    %force of friction and force of inclination present (note the force of
    %friction will be different to that of a flat floor)

    %Force of friction on inclination
    Ff_inc = u*(m*9.81*cos(deg2rad(zeta)));
    %Force due to inclination
    F_inc = m*9.81*sin(deg2rad(zeta));

    %Required force to transverse inclination at constant speed
    CI = Ff_inc + F_inc;

    %CASE 3: acceleration up an incline (AI)
    %********************************************************************
    %force of friction, inclination and acceleration present

    %Required force to accelerate on an incline
    AI = Ff_inc + F_inc + Fa;

%Create Force Vector
%888888888888888888888888888888888888888888888888888888888888888888888888

    %Force vetor
    ForceVec = [CF, AF, CI, AI];


end
```

```
function [ WheelOmega ] = AGVWheelAngVelCal( Vel, WheelRad )
%AGVWheelAngVelCal
%   This function calculates the max angular velocity of the AGV's wheels
%   given the desired top speed of the AGV and radius of the wheels

    %Inputs
    %*********************************************************************

    %Vel            = max linear velocity of the AGV        (m/s)
    %WheelRia       = radius of the AGVs wheels             (m)

    %Outputs
    %*********************************************************************

    %WheelOmega     = angular velocity of the AGV's wheels  (rad/s)

%Calcualte Angular Velocity of AGV Wheels
%8888888888888888888888888888888888888888888888888888888888888888888888888

    %Wheels angular velocity
    WheelOmega = Vel/WheelRad;
end
```

```
function [ TorqueVecWheel] = AGVTorqueCal( ForceVec, WheelRad, NoMotors_Ful,↙
NoMotors_Eco )
%AGVTorqueCal Info
%8888888888888888888888888888888888888888888888888888888888888888888888888
%   This function calculates the torque required by the AGV for the 4 force
%   cases (CF, AF, CI and AI) for the total torque required by the AGV, the
%   wheel torque required given the full drive motor compliment and the
%   wheel torque required given the eco compiment of drive motors

%   Inputs
%   *********************************************************************

%   ForceVec        = A vector forces containing [CF, AF, CI, AI]   (N)
%   WheelRad        = Radius of AGV wheels                          (m)
%   NoMotors_Ful    = No. of drive motors used for full compliment  (#)
%   NoMotors_Eco    = No. of drive motors used for eco compliment   (#)
%   DriveEff        = Efeciency of the gearing system               (0<x<1)

%   Outputs
%   *********************************************************************

%   TorqueVecWheel  = returns a 3x4 vector with the WTotT torques for each
%                     force condition in row 1, the WFulT torques for each
%                     force condition in row 2 and the WEcoT torques for
%                     each force condition in row 3 on the wheel side of
%                     the gearing
%                   = [ WTotT_CF, WTotT_AF, WTotT_CI, WTotT_AI ]
%                     [ WFulT_CF, WFulT_AF, WFulT_CI, WFulT_AI ]
%                     [ WEcoT_CF, WEcoT_AF, WEcoT_CI, WEcoT_AI ]    (Nm)

%Create Output Torque Vector
%88888888888888888888888888888888888888888888888888888888888888888888888

%   Initialise output torque vectors with all values zero
    TorqueVecWheel = zeros(3,4);

%Torque Requirement Calculations on Wheel Side of Gearbox
%888888888888888888888888888888888888888888888888888888888888888888888888

%   Calculate Total Torques
%   **********************************************************************

    for i = 1:4
        %Create total torque values for AGV (row 1 of TorqueVec) by looping
        %through ForceVec
        TorqueVecWheel(1,i) = WheelRad*ForceVec(1,i);
    end

%   Calculate Wheel Torques for Full Drive Motor Compliment Condition
%   **********************************************************************

    for i = 1:4
        %Create wheel torque values for AGV (row 2 of TorqueVec) by looping
        %through ForceVec
        TorqueVecWheel(2,i) = (WheelRad*ForceVec(1,i))/NoMotors_Ful;
    end

%   Calculate Wheel Torques for Eco Drive Motor Compliment Condition
%   **********************************************************************

    if NoMotors_Ful == NoMotors_Eco
        %number of Eco drive motors matches number of Full drive motors
        %thus torque for the eco condition will match the torques for the
        %full conditions
        TorqueVecWheel(3,:) = TorqueVecWheel(2,:);
    else
        %number of Eco drive motors doesn't matches number of Full drive
        %motors thus torque values must be calculated
        for i = 1:4
            %Create wheel torque values for AGV under Eco conditions(row 3
            %of TorqueVec) by looping through ForceVec
            TorqueVecWheel(3,i) = (WheelRad*ForceVec(1,i))/NoMotors_Eco;
        end
    end
end
```

```
function [ PowerVecWheel, PowerVecMotor ] = AGVPowerCal( TorqueVecWheel, WheelOmega,↙
DriveEff )
%AGVPowerCal Info
%8888888888888888888888888888888888888888888888888888888888888888888888888

%   This function calculates the power requirements for the AGV given the
%   four required force cases CF, AF, CI and AI for total power required,
%   power required by each wheel when the full drive compliment is used and
%   when the eco drive compilment is used

%   Inputs
%   **********************************************************************

%   TorqueVecWheel   = [ Tot_CF, Tot_AF, Tot_CI, Tot_AI ]
%                      [ Ful_CF, Ful_AF, Ful_CI, Ful_AI ]
%                      [ Eco_CF, Eco_AF, Eco_CI, Eco_AI ]       (Nm)
%   WheelOmega      = Angular velocity of the AGV's wheels      (rad/s)
%   DriveEff        = Efficiency of the gear system             (0<x<1)

%   Outputs
%   **********************************************************************

%   PowerVecWheel   = returns a 3x4 vector with the WTotP power for each
%                     force condition in row 1, the WFulP power for each
%                     force condition in row 2 and the WEcoP power for each
%                     force condition in row 3
%                   = [ WTotP_CF, WTotP_AF, WTotP_CI, WTotP_AI ]
%                     [ WFulP_CF, WFulP_AF, WFulP_CI, WFulP_AI ]
%                     [ WEcoP_CF, WEcoP_AF, WEcoP_CI, WEcoP_AI ]    (W)

%   PowerVecMotor   = returns a 3x4 vector with the MTotP power for each
%                     force condition in row 1, the MFulP power for each
%                     force condition in row 2 and the MEcoP power for each
%                     force condition in row 3
%                   = [ MTotP_CF, MTotP_AF, MTotP_CI, mTotP_AI ]
%                     [ MFulP_CF, MFulP_AF, MFulP_CI, MFulP_AI ]
%                     [ MEcoP_CF, MEcoP_AF, MEcoP_CI, MEcoP_AI ]    (W)

%Create Output Power Vector
%8888888888888888888888888888888888888888888888888888888888888888888888888

    PowerVecWheel = zeros(3,4);
    PowerVecMotor = zeros(3,4);

%Wheel Side Power Requirements Calculations
%8888888888888888888888888888888888888888888888888888888888888888888888888

%   Calculate Total Power Requirements
%   **********************************************************************

    for i = 1:4
        %Create total power values for AGV (row 1 of PowerVec) by looping
        %through TorqueVec
        PowerVecWheel(1,i) = TorqueVecWheel(1,i)*WheelOmega;
    end
```

```
%   Calculate Wheel Power for Full Drive Motor Compliment Condition
%   **********************************************************************

    for i = 1:4
        %Create wheel power values for AGV (row 2 of PowerVec) by looping
        %through TorqueVec
        PowerVecWheel(2,i) = TorqueVecWheel(2,i)*WheelOmega;
    end

%   Calculate Wheel Power for Eco Drive Motor Compliment Condition
%   **********************************************************************

    for i = 1:4
        %Create wheel power values for AGV under Eco conditions (row 3 of
        %PowerVec) by looping through TorqueVec
        PowerVecWheel(3,i) = TorqueVecWheel(3,i)*WheelOmega;
    end

%Motor Side Power Requirements Calculations
%8888888888888888888888888888888888888888888888888888888888888888888888888

%   Assuming an inefficiency of DriveEff is present in the gear system
%   that links the drive motors to the drive wheels the power required by
%   the AGV drive system is increased as follows

    for i = 1:3
        %for each motor compliment
        for j = 1:4
            %for each force case
            PowerVecMotor(i,j) = PowerVecWheel(i,j)/DriveEff;
        end
    end

end
```

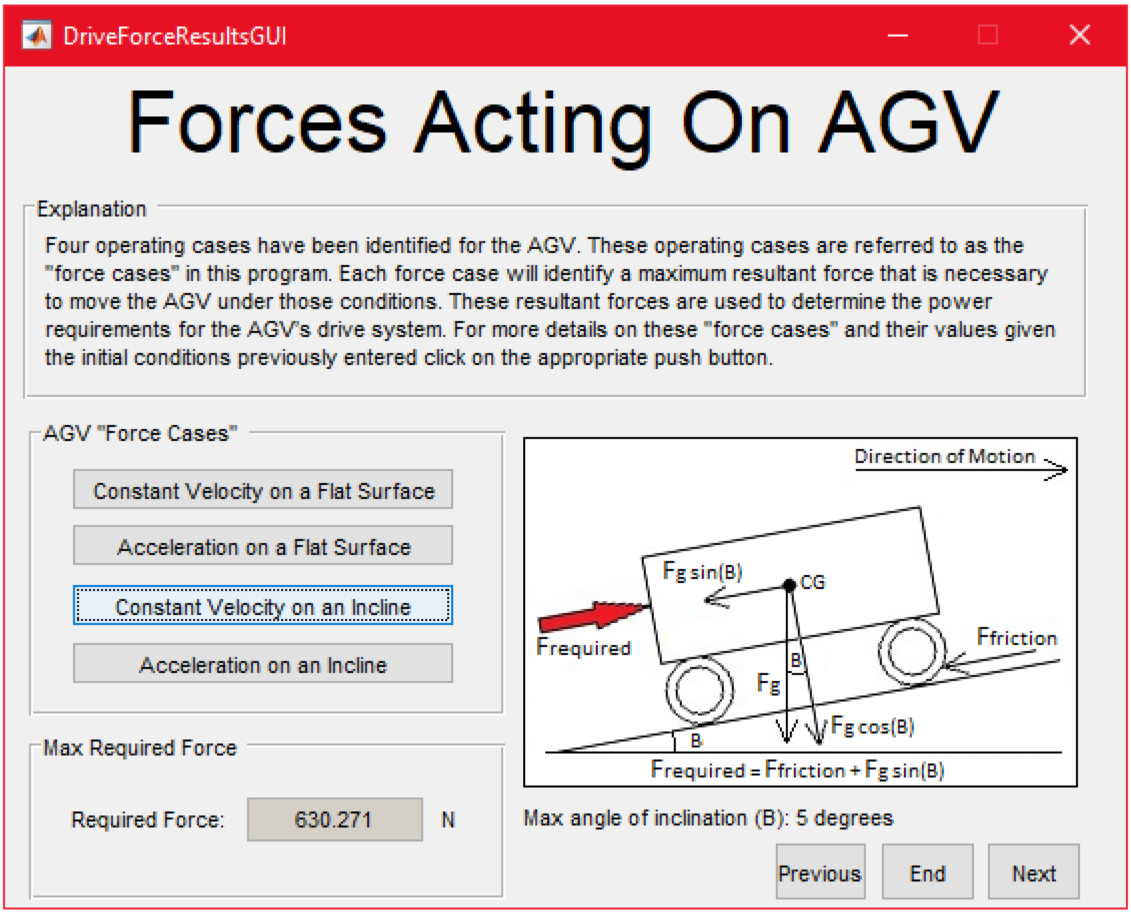
```
function [MaxWheelPower, MaxPowerReq, MaxTorq ,EcoUseLimits] = AGVMaxPowerReq(↙
PowerVecMotor, PowerVecWheel, TorqueVecWheel ,AllowAccInc )
%AGVMaxPowerReq
%8888888888888888888888888888888888888888888888888888888888888888888888888

%   This function calculates the maximum power requirement of the drive
%   motors of the AGV along with the force cases where the AGV can use the
%   Eco motor compliment system

%   Inputs
%   *********************************************************************

%   PowerVecMotor   = contains a 3x4 vector with the MTotP power for each
%                     force condition in row 1, the MFulP power for each
%                     force condition in row 2 and the MEcoP power for each
%                     force condition in row 3
%                   = [ MTotP_CF, MTotP_AF, MTotP_CI, mTotP_AI ]
%                     [ MFulP_CF, MFulP_AF, MFulP_CI, MFulP_AI ]
%                     [ MEcoP_CF, MEcoP_AF, MEcoP_CI, MEcoP_AI ]    (W)

%    PowerVecWheel  = returns a 3x4 vector with the WTotP power for each
%                     force condition in row 1, the WFulP power for each
%                     force condition in row 2 and the WEcoP power for each
%                     force condition in row 3
%                   = [ WTotP_CF, WTotP_AF, WTotP_CI, WTotP_AI ]
%                     [ WFulP_CF, WFulP_AF, WFulP_CI, WFulP_AI ]
%                     [ WEcoP_CF, WEcoP_AF, WEcoP_CI, WEcoP_AI ]    (W)

%   TorqueVecWheel  = returns a 3x4 vector with the WTotT torques for each
%                     force condition in row 1, the WFulT torques for each
%                     force condition in row 2 and the WEcoT torques for
%                     each force condition in row 3 on the wheel side of
%                     the gearing
%                   = [ WTotT_CF, WTotT_AF, WTotT_CI, WTotT_AI ]
%                     [ WFulT_CF, WFulT_AF, WFulT_CI, WFulT_AI ]
%                     [ WEcoT_CF, WEcoT_AF, WEcoT_CI, WEcoT_AI ]    (Nm)

%   AllowAccInc     = True/False allow acceleration on an incline   (T/F)

%   Outputs
%   *********************************************************************

%   MaxWheelPower   = Maximum power requirement for one drive wheel (W)

%   MaxPowerReq     = Maximum power requirement of the AGV's motors (W)

%   MaxTorq         = Maximum torque requirement for the AGV's wheels (Nm)

%   EcoUseLimits    = A true/false vector explaining under which
%                     force cases the Eco motor compliment system
%                     can be used                                   (T/F)
```

```
%8888888888888888888888888888888888888888888888888888888888888888888888888
%Create Output Eco Motor Compliment System Use Vector
%8888888888888888888888888888888888888888888888888888888888888888888888888

    EcoUseLimits = zeros(1,4);




%8888888888888888888888888888888888888888888888888888888888888888888888888
%Calculate Max Wheel Power
%8888888888888888888888888888888888888888888888888888888888888888888888888

%   The AGV wheel power will be specified according to the largest
%   power value found in the full motor compliment conditions if the AGV is
%   allowed to accelerate up and incline. If the AGV is not allowed to
%   accelerate up an incline the wheel power required will be found
%   in the full motor compliment conditions minus the power required to
%   accelerate up an incline value.

    if AllowAccInc == 0
        %False: dont allow acceleration on an incline
        MaxWheelPower = max([PowerVecWheel(2,1), PowerVecWheel(2,2), PowerVecWheel↙
(2,3)]);
    else
        %True: allow acceleration on an incline
        MaxWheelPower = max([PowerVecWheel(2,:)]);
    end




%8888888888888888888888888888888888888888888888888888888888888888888888888
%Calculate Max Motor Size
%8888888888888888888888888888888888888888888888888888888888888888888888888

%   The AGV drive motor power will be specified according to the largest
%   power value found in the full motor compliment conditions if the AGV is
%   allowed to accelerate up and incline. If the AGV is not allowed to
%   accelerate up an incline the drive motor power required will be found
%   in the full motor compliment conditions minus the power required to
%   accelerate up an incline value.

    if AllowAccInc == 0
        %False: dont allow acceleration on an incline
        MaxPowerReq = max([PowerVecMotor(2,1), PowerVecMotor(2,2), PowerVecMotor↙
(2,3)]);
    else
        %True: allow acceleration on an incline
        MaxPowerReq = max([PowerVecMotor(2,:)]);
    end
```

```
%8888888888888888888888888888888888888888888888888888888888888888888888888
%Calculate Max Torque
%8888888888888888888888888888888888888888888888888888888888888888888888888

%   This calculates the maximum torque that will be required on the AGV's
%   drive wheels. If acceleration is allowed to occur on an incline, the
%   max torque will be the maximum value found under full motor compliment
%   conditions. If acceleration is not allowed on an incline then the max
%   torque will exclude the resultant torque needed to accelerate up an
%   incline

    if AllowAccInc == 0
        %False: dont allow acceleration on an incline
        MaxTorq = max([TorqueVecWheel(2,1), TorqueVecWheel(2,2), TorqueVecWheel↙
(2,3)]);
    else
        %True: allow acceleration on an incline
        MaxTorq = max([TorqueVecWheel(2,:)]);
    end




%8888888888888888888888888888888888888888888888888888888888888888888888888
%Find Force Cases Where Eco Motor Compliment System Can be Used
%8888888888888888888888888888888888888888888888888888888888888888888888888

    for i = 1:4
        %for each Eco system force case
        if PowerVecMotor(3,i) < MaxPowerReq
            EcoUseLimits(1,i) = true;
        else
            EcoUseLimits(1,i) = false;
        end
    end

end
```

G.3 Drive Force Results GUI

Drive Force Results GUI:



DriveForceResultsGUI

Forces Acting On AGV

Explanation

Four operating cases have been identified for the AGV. These operating cases are referred to as the "force cases" in this program. Each force case will identify a maximum resultant force that is necessary to move the AGV under those conditions. These resultant forces are used to determine the power requirements for the AGV's drive system. For more details on these "force cases" and their values given the initial conditions previously entered click on the appropriate push button.

AGV "Force Cases"

Constant Velocity on a Flat Surface
Acceleration on a Flat Surface
Constant Velocity on an Incline
Acceleration on an Incline

Max Required Force

Required Force: 630.271 N

Max angle of inclination (B): 5 degrees

```
function varargout = DriveForceResultsGUI(varargin)
% GUI3 M-file for GUI3.fig
%      GUI3, by itself, creates a new GUI3 or raises the existing
%      singleton*.
%
%      H = GUI3 returns the handle to a new GUI3 or the handle to
%      the existing singleton*.
%
%      GUI3('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI3.M with the given input arguments.
%
%      GUI3('Property','Value',...) creates a new GUI3 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before DriveForceResultsGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to DriveForceResultsGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI3

% Last Modified by GUIDE v2.5 16-Aug-2016 16:28:38

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @DriveForceResultsGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @DriveForceResultsGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT




%8888888888888888888888888888888888888888888888888888888888888888888888888
%Initialise
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes just before GUI3 is made visible.
function DriveForceResultsGUI_OpeningFcn(hObject, eventdata, handles, varargin)
```

```
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GUI3 (see VARARGIN)

% Choose default command line output for GUI3
handles.output = hObject;
handles.ForceVec = varargin{2};
temp = num2str(varargin{3});
handles.MaxIncline = strcat({'Max angle of inclination (B): '},temp,{' degrees'});
%handles.RunThisGUI = 'Value';

% Update handles structure
guidata(hObject, handles);

% Setup initial conditions
if varargin{1} == 1
    %True: allow acceleration on an incline
    set(handles.InfoText,'String','Four operating cases have been identified for the↙
AGV. These operating cases are referred to as the "force cases" in this program. Each↙
force case will identify a maximum resultant force that is necessary to move the AGV↙
under those conditions. These resultant forces are used to determine the power↙
requirements for the AGV’s drive system. For more details on these "force cases" and↙
their values given the initial conditions previously entered click on the appropriate↙
push button.');
elseif varargin{1} == 0
    %False: don't allow accleration on incline
    set(handles.InfoText,'String','Three operating cases have been identified for the↙
AGV. These operating cases are referred to as the "force cases" in this program. Each↙
force case will identify a maximum resultant force that is necessary to move the AGV↙
under those conditions. These resultant forces are used to determine the power↙
requirements for the AGV’s drive system. For more details on these "force cases" and↙
their values given the initial conditions previously entered click on the appropriate↙
push button.');
    set(handles.AI_PB,'Visible','off');
else
    display('Error: incline acceleration state unknown');
end

set(handles.Angle_Txt,'String',handles.MaxIncline);

% Setup initial image
imshow('Blankforces.png');

% UIWAIT makes GUI3 wait for user response (see UIRESUME)
uiwait(handles.GUI3);



%888888888888888888888888888888888888888888888888888888888888888888888888
%Output Function
%888888888888888888888888888888888888888888888888888888888888888888888888
```

```
% --- Outputs from this function are returned to the command line.
function varargout = DriveForceResultsGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Get navigation data stored in pushbuttons
%*************************************************************************

    %Collect data for Next PB
    hpp1 = get(handles.Next_PB,'UserData');

    %Collect data for End PB
    hpp2 = get(handles.End_PB,'UserData');

    %Collect data for Previous PB
    hpp3 = get(handles.Prev_PB,'UserData');

%Determine which PB was pressed and update outputs
%(navigation data non-zero)
%*************************************************************************

    %varargout{1} = Next GUI to be run
    %varargout{2} = (if = 1)=> Keep Looping through results GUI else don't

    if hpp1 ~= 0
        %Next PB was pressed to close GUI
        display('Next button pressed');
        varargout{1} = 3;
        varargout{2} = 1;

    elseif hpp2 ~= 0
        %End PB was pressed to close GUI
        display('End button pressed');
        varargout{1} = 1;
        varargout{2} = 1;

    elseif hpp3 ~= 0
        %Previous PB was pressd to close GUI
        display('Previous button pressed');
        varargout{1} = 1;
        varargout{2} = 1;

    else
        %Unknown PB was pressed or user terminated
        display('Error: program closed unexpectedly')
    end

%Delete GUI
%*************************************************************************

    %The figure can now be deleted
    delete(handles.GUI3);
```

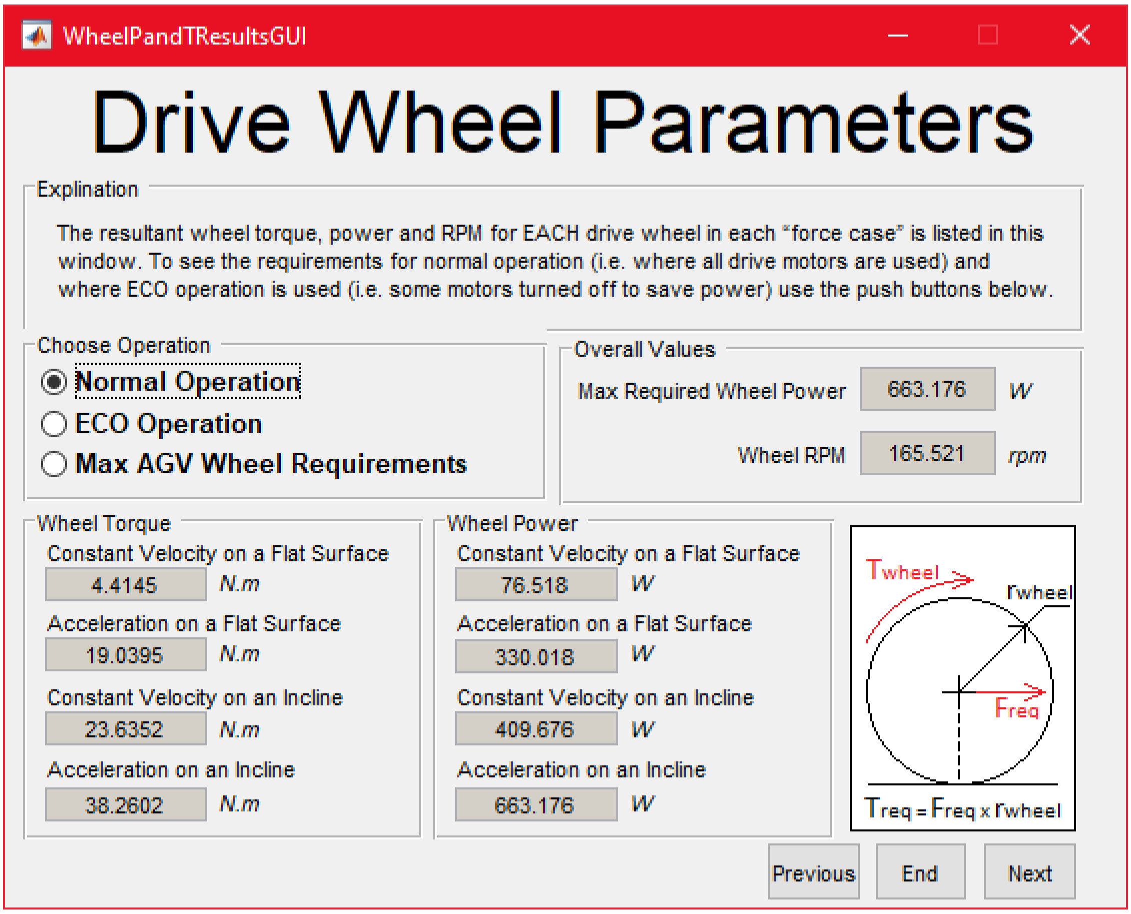
```
%888888888888888888888888888888888888888888888888888888888888888888888888
%Prev, Next, Home Pushbuttons
%888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes on button press in Next_PB.
function Next_PB_Callback(hObject, eventdata, handles)
% hObject    handle to Next_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.Next_PB,'UserData',1);

close;

% --- Executes on button press in End_PB.
function End_PB_Callback(hObject, eventdata, handles)
% hObject    handle to End_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.End_PB,'UserData',1);

close;

% --- Executes on button press in Prev_PB.
function Prev_PB_Callback(hObject, eventdata, handles)
% hObject    handle to Prev_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.Prev_PB,'UserData',1);

close;




%8888888888888888888888888888888888888888888888888888888888888888888888888
%Create Textboxes and Objects
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes1
axes(hObject);
```

```
set(hObject,'Tag','axes1');

% --- Executes during object creation, after setting all properties.
function RQForce_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to RQForce_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end






%888888888888888888888888888888888888888888888888888888888888888888888888
%UI Code Interaction Code(Pushbuttons, Radio Buttons, etc.)
%888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes on button press in CF_PB.
function CF_PB_Callback(hObject, eventdata, handles)
% hObject    handle to CF_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Update Required Force
%***********************************************************************

    %Update textbox
    set(handles.RQForce_Txt,'String',handles.ForceVec(1,1));

%Update Force Picture
%***********************************************************************

    %Update image
    axes(handles.axes1);
    imshow('CFforces.png');

    %Hide angle info text
    set(handles.Angle_Txt,'Visible','off');


% --- Executes on button press in AF_PB.
function AF_PB_Callback(hObject, eventdata, handles)
% hObject    handle to AF_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Update Required Force
%***********************************************************************
```

```
    %Update textbox
    set(handles.RQForce_Txt,'String',handles.ForceVec(1,2));

%Update Force Picture
%************************************************************************

    %Update image
    axes(handles.axes1);
    imshow('AFforces.png');

    %Hide angle info text
    set(handles.Angle_Txt,'Visible','off');


% --- Executes on button press in CI_PB.
function CI_PB_Callback(hObject, eventdata, handles)
% hObject    handle to CI_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Update Required Force
%************************************************************************

    %Update textbox
    set(handles.RQForce_Txt,'String',handles.ForceVec(1,3));

%Update Force Picture
%************************************************************************

    %Update image
    axes(handles.axes1);
    imshow('CIforces.png');

    %Show angle info text
    set(handles.Angle_Txt,'Visible','on');

% --- Executes on button press in AI_PB.
function AI_PB_Callback(hObject, eventdata, handles)
% hObject    handle to AI_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Update Required Force
%************************************************************************

    %Update textbox
    set(handles.RQForce_Txt,'String',handles.ForceVec(1,4));

%Update Force Picture
%************************************************************************

    %Update image
    axes(handles.axes1);
    imshow('AIforces.png');
    %Show angle info text
    set(handles.Angle_Txt,'Visible','on');




%8888888888888888888888888888888888888888888888888888888888888888888888888
%Close GUI
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes when user attempts to close GUI3.
function GUI3_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to GUI3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if isequal(get(hObject, 'waitstatus'),'waiting')
    %The GUI is still in UIWAIT, use UIRESUME
    uiresume(hObject);
else
    %The GUI is nolonger waiting and can be closed
    delete(hObject);
end
```

G.4 Wheel Power and Torque Results GUI

Wheel Power and Torque Results GUI:



WheelPandTResultsGUI

Drive Wheel Parameters

Explination

The resultant wheel torque, power and RPM for EACH drive wheel in each "force case" is listed in this window. To see the requirements for normal operation (i.e. where all drive motors are used) and where ECO operation is used (i.e. some motors turned off to save power) use the push buttons below.

Choose Operation

● Normal Operation

○ ECO Operation

○ Max AGV Wheel Requirements

Overall Values

Max Required Wheel Power: 663.176 W

Wheel RPM: 165.521 rpm

Wheel Torque

Constant Velocity on a Flat Surface: 4.4145 N.m

Acceleration on a Flat Surface: 19.0395 N.m

Constant Velocity on an Incline: 23.6352 N.m

Acceleration on an Incline: 38.2602 N.m

Wheel Power

Constant Velocity on a Flat Surface: 76.518 W

Acceleration on a Flat Surface: 330.018 W

Constant Velocity on an Incline: 409.676 W

Acceleration on an Incline: 663.176 W

```
function varargout = WheelPandTResultsGUI(varargin)
% WHEELPANDTRESULTSGUI M-file for WheelPandTResultsGUI.fig
%      WHEELPANDTRESULTSGUI, by itself, creates a new WHEELPANDTRESULTSGUI or raises↙
the existing
%      singleton*.
%
%      H = WHEELPANDTRESULTSGUI returns the handle to a new WHEELPANDTRESULTSGUI or↙
the handle to
%      the existing singleton*.
%
%      WHEELPANDTRESULTSGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in WHEELPANDTRESULTSGUI.M with the given input↙
arguments.
%
%      WHEELPANDTRESULTSGUI('Property','Value',...) creates a new WHEELPANDTRESULTSGUI↙
or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before WheelPandTResultsGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to WheelPandTResultsGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help WheelPandTResultsGUI

% Last Modified by GUIDE v2.5 16-Aug-2016 16:53:11

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @WheelPandTResultsGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @WheelPandTResultsGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT




%8888888888888888888888888888888888888888888888888888888888888888888888888
%Initialise
```

```
%888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes just before WheelPandTResultsGUI is made visible.
function WheelPandTResultsGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to WheelPandTResultsGUI (see VARARGIN)

% Choose default command line output for WheelPandTResultsGUI
handles.output = hObject;
handles.TorqueVec = varargin{3};
handles.PowerVec = varargin{4};
handles.EcoUse = varargin{6};

% Update handles structure
guidata(hObject, handles);

% Setup initial conditions
% ************************************************************************

if varargin{5} == 0
    %False: don't allow acceleration on an incline

    %set incline info in wheel torque UI box to invisiable
    set(handles.AITHead_Txt,'Visible','off');
    set(handles.AIT_Txt,'Visible','off');
    set(handles.AITUnit_Txt,'Visible','off');

    %set incline info in wheel power UI box to invisiable
    set(handles.AIPHead_Txt,'Visible','off');
    set(handles.AIP_Txt,'Visible','off');
    set(handles.AIPUnit_Txt,'Visible','off');

end

set(handles.Warn_Txt,'Visible','off');
set(handles.MaxP_Txt,'String',varargin{1});

MaxRPM = varargin{2}*(60/(2*pi()));
set(handles.WheelRPM_Txt,'String',MaxRPM);

imshow('WheelParameters.png');

% UIWAIT makes WheelPandTResultsGUI wait for user response (see UIRESUME)
uiwait(handles.GUI4);




%88888888888888888888888888888888888888888888888888888888888888888888888
%Output Function
%88888888888888888888888888888888888888888888888888888888888888888888888
```

```
% --- Outputs from this function are returned to the command line.
function varargout = WheelPandTResultsGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Get navigation data stored in pushbuttons
%************************************************************************

    %Collect data for Next PB
    hpp1 = get(handles.Next_PB,'UserData');

    %Collect data for End PB
    hpp2 = get(handles.End_PB,'UserData');

    %Collect data for Previous PB
    hpp3 = get(handles.Prev_PB,'UserData');

%Determine which PB was pressed and update outputs
%(navigation data non-zero)
%************************************************************************

    %varargout{1} = Next GUI to be run
    %varargout{2} = (if = 1)=> Keep Looping through results GUI else don't

    if hpp1 ~= 0
        %Next PB was pressed to close GUI
        display('Next button pressed');
        varargout{1} = 4;
        varargout{2} = 1;

    elseif hpp2 ~= 0
        %End PB was pressed to close GUI
        display('End button pressed');
        varargout{1} = 1;
        varargout{2} = 1;

    elseif hpp3 ~= 0
        %Previous PB was pressd to close GUI
        display('Previous button pressed');
        varargout{1} = 2;
        varargout{2} = 1;

    else
        %Unknown PB was pressed or user terminated
        display('Error: program closed unexpectedly')
    end

%Delete GUI
%************************************************************************

    %The figure can now be deleted
    delete(handles.GUI4);
```

```
%8888888888888888888888888888888888888888888888888888888888888888888888888
%Prev, Next, Home Pushbuttons
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes on button press in End_PB.
function End_PB_Callback(hObject, eventdata, handles)
% hObject    handle to End_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.End_PB,'UserData',1);

close;

% --- Executes on button press in Next_PB.
function Next_PB_Callback(hObject, eventdata, handles)
% hObject    handle to Next_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.Next_PB,'UserData',1);

close;

% --- Executes on button press in Prev_PB.
function Prev_PB_Callback(hObject, eventdata, handles)
% hObject    handle to Prev_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.Prev_PB,'UserData',1);

close;





%888888888888888888888888888888888888888888888888888888888888888888888888
%Create Textboxes and Objects
%888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% --- Executes during object creation, after setting all properties.
function CFP_Txt_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to CFP_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function AFP_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to AFP_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function CIP_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to CIP_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function AIP_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to AIP_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function CFT_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to CFT_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function AFT_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to AFT_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function CIT_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to CIT_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function AIT_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to AIT_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function MaxP_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MaxP_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

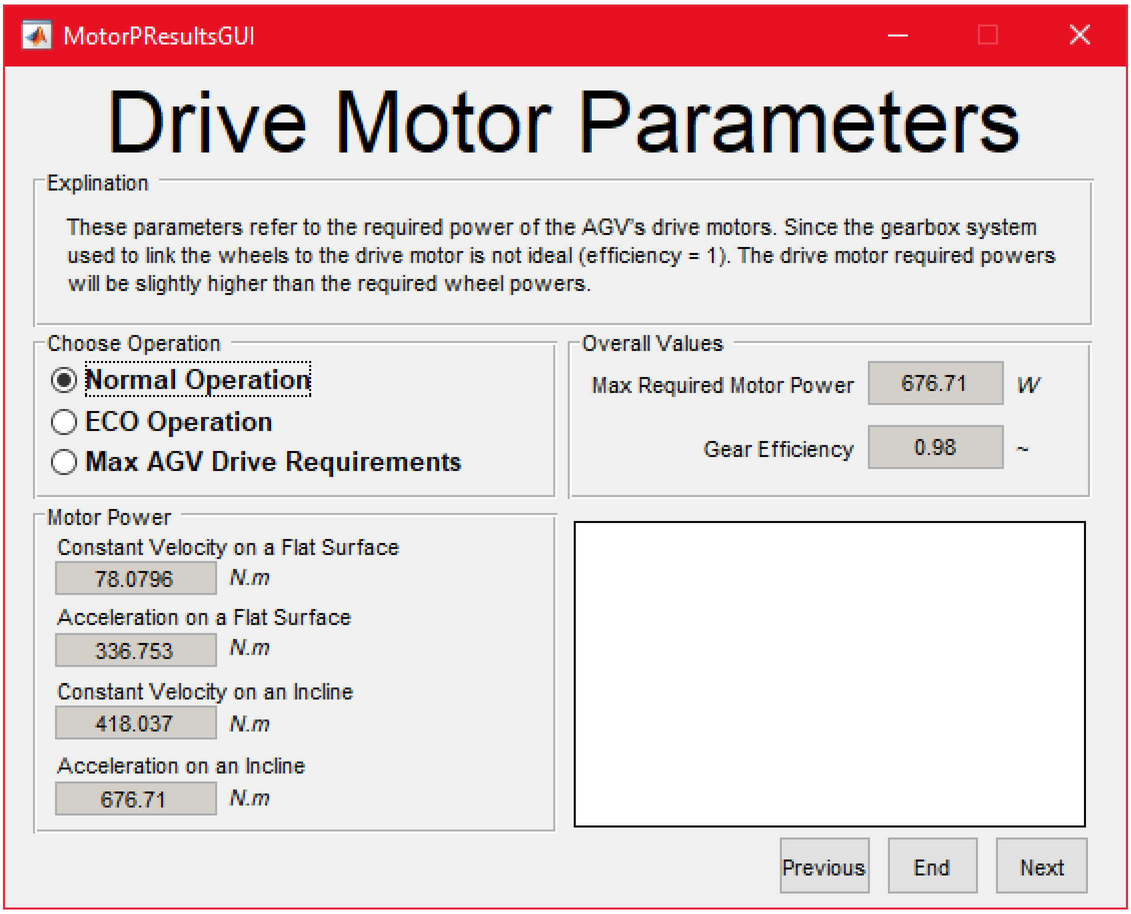
```
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function WheelRPM_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to WheelRPM_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




%888888888888888888888888888888888888888888888888888888888888888888888888
%UI Code Interaction Code(Pushbuttons, Radio Buttons, etc.)
%888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes when selected object is changed in OPChoose_UI.
function OPChoose_UI_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in OPChoose_UI
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue: handle of the previously selected object or empty if none was selected
%   NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)

% Initial Paramaters
%************************************************************************

    %On selection of radio button pass tag of selected radio button to
    %variable "SelectedRB"
    SelectedRB = get(eventdata.NewValue,'Tag');

    %Disable warning label
    set(handles.Warn_Txt,'Visible','off');

    %Set all box text back to black
    set(handles.CFT_Txt,'ForegroundColor',[0 0 0]);
    set(handles.CFP_Txt,'ForegroundColor',[0 0 0]);
    set(handles.AFT_Txt,'ForegroundColor',[0 0 0]);
    set(handles.AFP_Txt,'ForegroundColor',[0 0 0]);
    set(handles.CIT_Txt,'ForegroundColor',[0 0 0]);
    set(handles.CIP_Txt,'ForegroundColor',[0 0 0]);
    set(handles.AIT_Txt,'ForegroundColor',[0 0 0]);
    set(handles.AIP_Txt,'ForegroundColor',[0 0 0]);
```

```
%Populate UI Boxes
%*************************************************************************

    if strcmp(SelectedRB,'NO_RB') == 1
        % Normal operation results selected using radio button

        %set parameters for "Wheel Torque" UI box
        set(handles.CFT_Txt,'String',handles.TorqueVec(2,1));
        set(handles.AFT_Txt,'String',handles.TorqueVec(2,2));
        set(handles.CIT_Txt,'String',handles.TorqueVec(2,3));
        set(handles.AIT_Txt,'String',handles.TorqueVec(2,4));

        %set parameters for "Wheel Power" UI box
        set(handles.CFP_Txt,'String',handles.PowerVec(2,1));
        set(handles.AFP_Txt,'String',handles.PowerVec(2,2));
        set(handles.CIP_Txt,'String',handles.PowerVec(2,3));
        set(handles.AIP_Txt,'String',handles.PowerVec(2,4));

    elseif strcmp(SelectedRB,'ECO_RB') == 1
        % Eco operation results selected using radio button

        %Show wanrning lable yes/no
        ShowWarn = 0; %don't show

            %Results: constant velocity on a flat surface (CF)
            if handles.EcoUse(1,1) == 1
                %ECO conditions can be used under CF conditions
                set(handles.CFT_Txt,'String',handles.TorqueVec(3,1));
                set(handles.CFP_Txt,'String',handles.PowerVec(3,1));
            else
                %ECO canditions cannot be used under CF conditions
                set(handles.CFT_Txt,'ForegroundColor',[0.85 0.19 0]);
                set(handles.CFP_Txt,'ForegroundColor',[0.85 0.19 0]);
                set(handles.CFT_Txt,'String','NotA');
                set(handles.CFP_Txt,'String','NotA');

                %Show warning label
                ShowWarn = 1;
            end

            %Results: acceleration on a flat surface (AF)
            if handles.EcoUse(1,2) == 1
                %ECO conditions can be used under AF conditions
                set(handles.AFT_Txt,'String',handles.TorqueVec(3,2));
                set(handles.AFP_Txt,'String',handles.PowerVec(3,2));
            else
                %ECO canditions cannot be used under AF conditions
                set(handles.AFT_Txt,'ForegroundColor',[0.85 0.19 0]);
                set(handles.AFP_Txt,'ForegroundColor',[0.85 0.19 0]);
                set(handles.AFT_Txt,'String','NotA');
                set(handles.AFP_Txt,'String','NotA');

                %Show warning label
                ShowWarn = 1;
            end
            %Results: constant velocity up an incline (CI)
            if handles.EcoUse(1,3) == 1
                %ECO conditions can be used under CI conditions
                set(handles.CIT_Txt,'String',handles.TorqueVec(3,3));
                set(handles.CIP_Txt,'String',handles.PowerVec(3,3));
            else
                %ECO canditions cannot be used under CI conditions
                set(handles.CIT_Txt,'ForegroundColor',[0.85 0.19 0]);
                set(handles.CIP_Txt,'ForegroundColor',[0.85 0.19 0]);
                set(handles.CIT_Txt,'String','NotA');
                set(handles.CIP_Txt,'String','NotA');

                %Show warning label
                ShowWarn = 1;
            end

            %Results: acceleration up an incline (AI)
            if handles.EcoUse(1,4) == 1
                %ECO conditions can be used under AI conditions
                set(handles.AIT_Txt,'String',handles.TorqueVec(3,4));
                set(handles.AIP_Txt,'String',handles.PowerVec(3,4));
            else
                %ECO canditions cannot be used under AI conditions
                set(handles.AIT_Txt,'ForegroundColor',[0.85 0.19 0]);
                set(handles.AIP_Txt,'ForegroundColor',[0.85 0.19 0]);
                set(handles.AIT_Txt,'String','NotA');
                set(handles.AIP_Txt,'String','NotA');

                %Show warning label
                ShowWarn = 1;
            end

        %Write warning to screen if "ShowWarn" true
        if ShowWarn == 1
            set(handles.Warn_Txt,'Visible','on');
        end

    elseif strcmp(SelectedRB,'Sum_RB') == 1
        % Show sum of agv requirements selected using radio button

            %Results: constant velocity on a flat surface (CF)
            if handles.EcoUse(1,1) == 1
                %ECO conditions can be used under CF conditions
                set(handles.CFT_Txt,'String',handles.TorqueVec(3,1));
                set(handles.CFP_Txt,'String',handles.PowerVec(3,1));
            else
                %ECO canditions cannot be used under CF conditions
                set(handles.CFT_Txt,'String',handles.TorqueVec(2,1));
                set(handles.CFP_Txt,'String',handles.PowerVec(2,1));
            end

            %Results: acceleration on a flat surface (AF)
            if handles.EcoUse(1,2) == 1
                %ECO conditions can be used under AF conditions
                set(handles.AFT_Txt,'String',handles.TorqueVec(3,2));
                set(handles.AFP_Txt,'String',handles.PowerVec(3,2));
            else
                %ECO canditions cannot be used under AF conditions
                set(handles.AFT_Txt,'String',handles.TorqueVec(2,2));
                set(handles.AFP_Txt,'String',handles.PowerVec(2,2));
            end

            %Results: constant velocity up an incline (CI)
            if handles.EcoUse(1,3) == 1
                %ECO conditions can be used under CI conditions
                set(handles.CIT_Txt,'String',handles.TorqueVec(3,3));
                set(handles.CIP_Txt,'String',handles.PowerVec(3,3));
            else
                %ECO canditions cannot be used under CI conditions
                set(handles.CIT_Txt,'String',handles.TorqueVec(2,3));
                set(handles.CIP_Txt,'String',handles.PowerVec(2,3));
            end

                        %Results: acceleration up an incline (AI)
            if handles.EcoUse(1,4) == 1
                %ECO conditions can be used under AI conditions
                set(handles.AIT_Txt,'String',handles.TorqueVec(3,4));
                set(handles.AIP_Txt,'String',handles.PowerVec(3,4));
            else
                %ECO canditions cannot be used under AI conditions
                set(handles.AIT_Txt,'String',handles.TorqueVec(2,4));
                set(handles.AIP_Txt,'String',handles.PowerVec(2,4));
            end
%{
        %set parameters for "Wheel Torque" UI box
        set(handles.CFT_Txt,'String',handles.TorqueVec(1,1));
        set(handles.AFT_Txt,'String',handles.TorqueVec(1,2));
        set(handles.CIT_Txt,'String',handles.TorqueVec(1,3));
        set(handles.AIT_Txt,'String',handles.TorqueVec(1,4));

        %set parameters for "Wheel Power" UI box
        set(handles.CFP_Txt,'String',handles.PowerVec(1,1));
        set(handles.AFP_Txt,'String',handles.PowerVec(1,2));
        set(handles.CIP_Txt,'String',handles.PowerVec(1,3));
        set(handles.AIP_Txt,'String',handles.PowerVec(1,4));
%}
    else
        %requested result unknown
        display('Error: requested radio button outside avaliable options');
    end



%8888888888888888888888888888888888888888888888888888888888888888888888888
%Close GUI
%8888888888888888888888888888888888888888888888888888888888888888888888888
```

```
% --- Executes when user attempts to close GUI4.
function GUI4_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to GUI4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if isequal(get(hObject, 'waitstatus'),'waiting')
    %The GUI is still in UIWAIT, use UIRESUME
    uiresume(hObject);
else
    %The GUI is nolonger waiting and can be closed
    delete(hObject);
end
```

G.5 Motor Power Results

Motor Power Results:



Drive Motor Parameters

Explination

These parameters refer to the required power of the AGV's drive motors. Since the gearbox system used to link the wheels to the drive motor is not ideal (efficiency = 1). The drive motor required powers will be slightly higher than the required wheel powers.

Choose Operation

● Normal Operation
○ ECO Operation
○ Max AGV Drive Requirements

Overall Values

Max Required Motor Power 676.71 W

Gear Efficiency 0.98 ~

Motor Power

Constant Velocity on a Flat Surface 78.0796 N.m

Acceleration on a Flat Surface 336.753 N.m

Constant Velocity on an Incline 418.037 N.m

Acceleration on an Incline 676.71 N.m

```
function varargout = MotorPResultsGUI(varargin)
% MOTORPRESULTSGUI M-file for MotorPResultsGUI.fig
%      MOTORPRESULTSGUI, by itself, creates a new MOTORPRESULTSGUI or raises the↙
existing
%      singleton*.
%
%      H = MOTORPRESULTSGUI returns the handle to a new MOTORPRESULTSGUI or the handle↙
to
%      the existing singleton*.
%
%      MOTORPRESULTSGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MOTORPRESULTSGUI.M with the given input arguments.
%
%      MOTORPRESULTSGUI('Property','Value',...) creates a new MOTORPRESULTSGUI or↙
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before MotorPResultsGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to MotorPResultsGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help MotorPResultsGUI

% Last Modified by GUIDE v2.5 15-Aug-2016 14:25:16

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @MotorPResultsGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @MotorPResultsGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT




%8888888888888888888888888888888888888888888888888888888888888888888888888
%Initialise
%8888888888888888888888888888888888888888888888888888888888888888888888888
```

```
% --- Executes just before MotorPResultsGUI is made visible.
function MotorPResultsGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to MotorPResultsGUI (see VARARGIN)

% Choose default command line output for MotorPResultsGUI
handles.output = hObject;
handles.PowerVec = varargin{3};
handles.EcoUse = varargin{4};

% Update handles structure
guidata(hObject, handles);

% Setup initial conditions
set(handles.MaxMotP_Txt,'String',varargin{1});
set(handles.Eff_Txt,'String',varargin{2});
set(handles.Warn_Txt,'Visible','off');

% Setup initial image
imshow('MotorParameters.png');

% UIWAIT makes MotorPResultsGUI wait for user response (see UIRESUME)
uiwait(handles.GUI5);





%8888888888888888888888888888888888888888888888888888888888888888888888888
%Output Function
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Outputs from this function are returned to the command line.
function varargout = MotorPResultsGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Get navigation data stored in pushbuttons
%************************************************************************

    %Collect data for Next PB
    hpp1 = get(handles.Next_PB,'UserData');

    %Collect data for End PB
    hpp2 = get(handles.End_PB,'UserData');

    %Collect data for Previous PB
    hpp3 = get(handles.Prev_PB,'UserData');
```

```
%Determine which PB was pressed and update outputs
%(navigation data non-zero)
%*************************************************************************

    %varargout{1} = Next GUI to be run
    %varargout{2} = (if = 1)=> Keep Looping through results GUI else don't

    if hpp1 ~= 0
        %Next PB was pressed to close GUI
        display('Next button pressed');
        varargout{1} = 5;
        varargout{2} = 1;

    elseif hpp2 ~= 0
        %End PB was pressed to close GUI
        display('End button pressed');
        varargout{1} = 1;
        varargout{2} = 1;

    elseif hpp3 ~= 0
        %Previous PB was pressd to close GUI
        display('Previous button pressed');
        varargout{1} = 3;
        varargout{2} = 1;

    else
        %Unknown PB was pressed or user terminated
        display('Error: program closed unexpectedly')
    end

%Delete GUI
%*************************************************************************

    %The figure can now be deleted
    delete(handles.GUI5);





%888888888888888888888888888888888888888888888888888888888888888888888888
%Prev, Next, Home Pushbuttons
%888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes on button press in End_PB.
function End_PB_Callback(hObject, eventdata, handles)
% hObject    handle to End_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.End_PB,'UserData',1);

close;

% --- Executes on button press in Next_PB.
```

```
function Next_PB_Callback(hObject, eventdata, handles)
% hObject    handle to Next_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.Next_PB,'UserData',1);

close;

% --- Executes on button press in Prev_PB.
function Prev_PB_Callback(hObject, eventdata, handles)
% hObject    handle to Prev_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.Prev_PB,'UserData',1);

close;





%8888888888888888888888888888888888888888888888888888888888888888888888888
%Create Textboxes and Objects
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes1

% --- Executes during object creation, after setting all properties.
function CFP_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to CFP_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function AFP_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to AFP_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function CIP_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to CIP_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function AIP_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to AIP_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function MaxMotP_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MaxMotP_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function Eff_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Eff_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
```

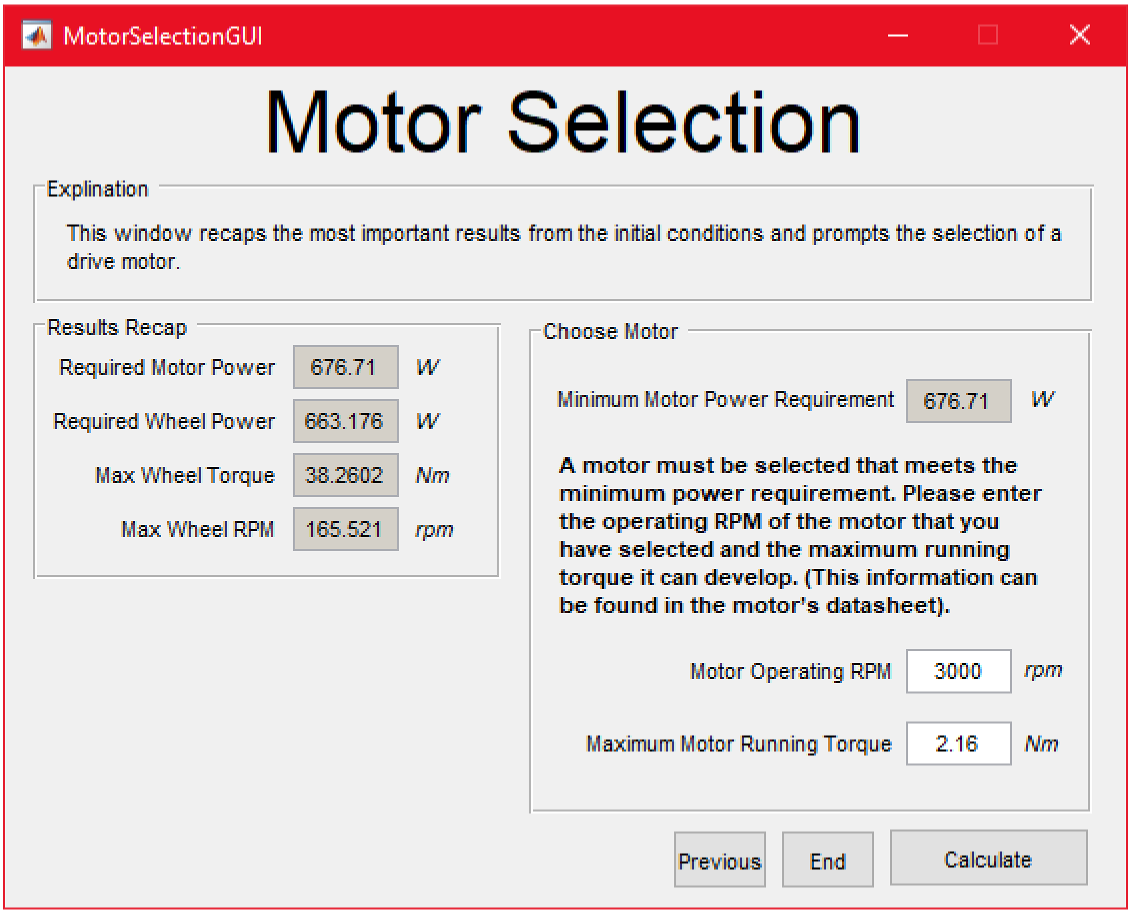
```
    set(hObject,'BackgroundColor','white');
end





%8888888888888888888888888888888888888888888888888888888888888888888888888
%UI Code Interaction Code(Pushbuttons, Radio Buttons, etc.)
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes when selected object is changed in ChooseOP_UI.
function ChooseOP_UI_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in ChooseOP_UI
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue: handle of the previously selected object or empty if none was selected
%   NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)

% Initial Paramaters
%*************************************************************************

    %On selection of radio button pass tag of selected radio button to variable
    %"SelectedRB"
    SelectedRB = get(eventdata.NewValue,'Tag');

    %Disable warning label
    set(handles.Warn_Txt,'Visible','off');

    %Set all box text back to black
    set(handles.CFP_Txt,'ForegroundColor',[0 0 0]);
    set(handles.AFP_Txt,'ForegroundColor',[0 0 0]);
    set(handles.CIP_Txt,'ForegroundColor',[0 0 0]);
    set(handles.AIP_Txt,'ForegroundColor',[0 0 0]);

%Populate UI Boxes
%*************************************************************************

    if strcmp(SelectedRB, 'NO_RB') == 1
        % Normal operation results selected using radio button, set
        % parameters for "Motor Power" UI box

        set(handles.CFP_Txt,'String',handles.PowerVec(2,1));
        set(handles.AFP_Txt,'String',handles.PowerVec(2,2));
        set(handles.CIP_Txt,'String',handles.PowerVec(2,3));
        set(handles.AIP_Txt,'String',handles.PowerVec(2,4));

    elseif strcmp(SelectedRB,'ECO_RB') == 1
        %Eco operation results selected using radio button, set parameters
        %for "Motor Power" UI box

        %Show wanrning lable yes/no
        ShowWarn = 0; %don't show
```

```
            %Results: constant velocity on a flat surface (CF)
            if handles.EcoUse(1,1) == 1
                %ECO conditions can be used under CF conditions
                set(handles.CFP_Txt,'String',handles.PowerVec(3,1));
            else
                %ECO canditions cannot be used under CF conditions
                set(handles.CFP_Txt,'ForegroundColor',[0.85 0.19 0]);
                set(handles.CFP_Txt,'String','NotA');

                %Show warning label
                ShowWarn = 1;
            end

            %Results: acceleration on a flat surface (AF)
            if handles.EcoUse(1,2) == 1
                %ECO conditions can be used under AF conditions
                set(handles.AFP_Txt,'String',handles.PowerVec(3,2));
            else
                %ECO canditions cannot be used under AF conditions
                set(handles.AFP_Txt,'ForegroundColor',[0.85 0.19 0]);
                set(handles.AFP_Txt,'String','NotA');

                %Show warning label
                ShowWarn = 1;
            end

            %Results: constant velocity up an incline (CI)
            if handles.EcoUse(1,3) == 1
                %ECO conditions can be used under CI conditions
                set(handles.CIP_Txt,'String',handles.PowerVec(3,3));
            else
                %ECO canditions cannot be used under CI conditions
                set(handles.CIP_Txt,'ForegroundColor',[0.85 0.19 0]);
                set(handles.CIP_Txt,'String','NotA');

                %Show warning label
                ShowWarn = 1;
            end

            %Results: acceleration up an incline (AI)
            if handles.EcoUse(1,4) == 1
                %ECO conditions can be used under AI conditions
                set(handles.AIP_Txt,'String',handles.PowerVec(3,4));
            else
                %ECO canditions cannot be used under AI conditions
                set(handles.AIP_Txt,'ForegroundColor',[0.85 0.19 0]);
                set(handles.AIP_Txt,'String','NotA');

                %Show warning label
                ShowWarn = 1;
            end

        %Write warning to screen if "ShowWarn" true
        if ShowWarn == 1
            set(handles.Warn_Txt,'Visible','on');
        end

    elseif strcmp(SelectedRB,'Sum_RB') == 1
        % Show sum of agv requirements selected using radio button, set
        % parameters for "Motor Power" UI box

            %Results: constant velocity on a flat surface (CF)
            if handles.EcoUse(1,1) == 1
                %ECO conditions can be used under CF conditions
                set(handles.CFP_Txt,'String',handles.PowerVec(3,1));
            else
                %ECO canditions cannot be used under CF conditions
                set(handles.CFP_Txt,'String',handles.PowerVec(2,1));
            end

            %Results: acceleration on a flat surface (AF)
            if handles.EcoUse(1,2) == 1
                %ECO conditions can be used under AF conditions
                set(handles.AFP_Txt,'String',handles.PowerVec(3,2));
            else
                %ECO canditions cannot be used under AF conditions
                set(handles.AFP_Txt,'String',handles.PowerVec(2,2));
            end

            %Results: constant velocity up an incline (CI)
            if handles.EcoUse(1,3) == 1
                %ECO conditions can be used under CI conditions
                set(handles.CIP_Txt,'String',handles.PowerVec(3,3));
            else
                %ECO canditions cannot be used under CI conditions
                set(handles.CIP_Txt,'String',handles.PowerVec(2,3));
            end

            %Results: acceleration up an incline (AI)
            if handles.EcoUse(1,4) == 1
                %ECO conditions can be used under AI conditions
                set(handles.AIP_Txt,'String',handles.PowerVec(3,4));
            else
                %ECO canditions cannot be used under AI conditions
                set(handles.AIP_Txt,'String',handles.PowerVec(2,4));
            end

            %{
        set(handles.CFP_Txt,'String',handles.PowerVec(1,1));
        set(handles.AFP_Txt,'String',handles.PowerVec(1,2));
        set(handles.CIP_Txt,'String',handles.PowerVec(1,3));
        set(handles.AIP_Txt,'String',handles.PowerVec(1,4));
            %}

    else
        %requested result unknown
        display('Error: requested radio button outside avaliable options');
    end
```

```
%8888888888888888888888888888888888888888888888888888888888888888888888888
%Close GUI
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes when user attempts to close GUI5.
function GUI5_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to GUI5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if isequal(get(hObject, 'waitstatus'),'waiting')
    %The GUI is still in UIWAIT, use UIRESUME
    uiresume(hObject);
else
    %The GUI is nolonger waiting and can be closed
    delete(hObject);
end
```

G.6 Motor Selection GUI

Motor Selection GUI:



Motor Selection

Explination

This window recaps the most important results from the initial conditions and prompts the selection of a drive motor.

Results Recap

Required Motor Power 676.71 *W*

Required Wheel Power 663.176 *W*

Max Wheel Torque 38.2602 *Nm*

Max Wheel RPM 165.521 *rpm*

Choose Motor

Minimum Motor Power Requirement 676.71 *W*

A motor must be selected that meets the minimum power requirement. Please enter the operating RPM of the motor that you have selected and the maximum running torque it can develop. (This information can be found in the motor's datasheet).

Motor Operating RPM 3000 *rpm*

Maximum Motor Running Torque 2.16 *Nm*

```
function varargout = MotorSelectionGUI(varargin)
% MOTORSELECTIONGUI M-file for MotorSelectionGUI.fig
%      MOTORSELECTIONGUI, by itself, creates a new MOTORSELECTIONGUI or raises the↙
existing
%      singleton*.
%
%      H = MOTORSELECTIONGUI returns the handle to a new MOTORSELECTIONGUI or the↙
handle to
%      the existing singleton*.
%
%      MOTORSELECTIONGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MOTORSELECTIONGUI.M with the given input arguments.
%
%      MOTORSELECTIONGUI('Property','Value',...) creates a new MOTORSELECTIONGUI or↙
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before MotorSelectionGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to MotorSelectionGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help MotorSelectionGUI

% Last Modified by GUIDE v2.5 16-Aug-2016 15:14:24

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @MotorSelectionGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @MotorSelectionGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT




%8888888888888888888888888888888888888888888888888888888888888888888888888
%Initialise
%8888888888888888888888888888888888888888888888888888888888888888888888888
```

```matlab
% --- Executes just before MotorSelectionGUI is made visible.
function MotorSelectionGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to MotorSelectionGUI (see VARARGIN)

% Choose default command line output for MotorSelectionGUI
handles.output = hObject;
handles.MotorPow = varargin{1};

% Update handles structure
guidata(hObject, handles);

% Setup initial conditions
set(handles.ReqPowM_Txt,'String',varargin{1});
set(handles.ReqMotorP_Txt,'String',varargin{1});
set(handles.ReqPowW_Txt,'String',varargin{2});
set(handles.MaxTorq_Txt,'String',varargin{3});

MaxRPM = varargin{4}*(60/(2*pi()));
set(handles.MaxRPM_Txt,'String',MaxRPM);

% Disable warning message
set(handles.ErrorMessage,'Visible','off');
set(handles.MotorFlag_Txt,'Visible','off');

% Add suggested values to user input
set(handles.MotorRPM_Txt,'String','3000');
set(handles.MotorTorq_Txt,'String','2.16');

% UIWAIT makes MotorSelectionGUI wait for user response (see UIRESUME)
uiwait(handles.GUI6);





%8888888888888888888888888888888888888888888888888888888888888888888888888
%Output Function
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Outputs from this function are returned to the command line.
function varargout = MotorSelectionGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%(navigation data non-zero)
%************************************************************************

    %varargout{1} = Next GUI to be run
    %varargout{2} = (if = 1)=> Keep Looping through results GUI else don't

    if get(handles.Calculate_PB,'UserData') ~= 0
        %Calculate PB was pressed to close GUI
        display('Calculate button pressed');
        varargout{1} = 6;
        varargout{2} = 1;

    elseif get(handles.End_PB,'UserData') ~= 0
        %End PB was pressed to close GUI
        display('End button pressed');
        varargout{1} = 1;
        varargout{2} = 1;

    elseif get(handles.Prev_PB,'UserData') ~= 0
        %Previous PB was pressd to close GUI
        display('Previous button pressed');
        varargout{1} = 4;
        varargout{2} = 1;

    else
        %Unknown PB was pressed or user terminated
        display('Error: program closed unexpectedly')
    end

%Set outputs
%*************************************************************************

    %Motor operating omega (rad/sec)
    varargout{3} = (str2double(get(handles.MotorRPM_Txt,'String')))*((2*pi())/60);

    %Motor max running torque (Nm)
    varargout{4} = str2double(get(handles.MotorTorq_Txt,'String'));

%Delete GUI
%*************************************************************************

    %The figure can now be deleted
    delete(handles.GUI6);





%888888888888888888888888888888888888888888888888888888888888888888888888
%Prev, Next, Home Pushbuttons
%888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes on button press in End_PB.
function End_PB_Callback(hObject, eventdata, handles)
% hObject    handle to End_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.End_PB,'UserData',1);
```

```
close;

% --- Executes on button press in Prev_PB.
function Prev_PB_Callback(hObject, eventdata, handles)
% hObject    handle to Prev_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.Prev_PB,'UserData',1);

close;





%8888888888888888888888888888888888888888888888888888888888888888888888888
%Create Textboxes and Objects
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes during object creation, after setting all properties.
function ReqMotorP_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ReqMotorP_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function MotorRPM_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MotorRPM_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function MotorTorq_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MotorTorq_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function ReqPowM_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ReqPowM_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function ReqPowW_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ReqPowW_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function MaxTorq_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MaxTorq_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function MaxRPM_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MaxRPM_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end





%8888888888888888888888888888888888888888888888888888888888888888888888888
%UI Code Interaction Code(Pushbuttons, Radio Buttons, etc.)
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes on button press in Calculate_PB.
function Calculate_PB_Callback(hObject, eventdata, handles)
% hObject    handle to Calculate_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Initialise values
%************************************************************************

    %When stop = false GUI can close + calculations can start, else =
    %mistake in input thus let user change input (initialised to false)
    stop = false;
    rpmCorrect = false;
    flagMotor = false;

    %Diable error message
    set(handles.ErrorMessage,'Visible','off');
    set(handles.MotorFlag_Txt,'Visible','off');


%Motor operating RPM processing
%*************************************************************************

    %copy operating RPM to "holdstring"
    holdstring = get(handles.MotorRPM_Txt,'String');

    %check if each character in "holdstring" is a digits and store result
    %as vector "test1"
    test1 = isstrprop(holdstring,'digit');

    %check if any charaters in "holdstring" are decimal points and store
    %results as vector "test2"
    test2 = zeros(1,length(holdstring));

    for i = 1:length(test1)
        %for each character in "holdstring"

        if holdstring(1,i) == '.'
            %check if character is a decimal point

            test2(1,i) = 1;

        end
    end
    %add "test1" and "test2", resultant vector will contain a 1 for any
    %character that is either a decimal point or digit and 0 if it is not.
    %Eg [1 1 1 1] = valid number
    %   [1 1 0 1] = string input contains one non digit/decimal point,
    %               invalid number

    test = test1 + test2;

    %if product of test is zero = invalid number else number is valid
    isDigit = prod([1,test]);

    %check if number is valid and non-zero
    if (str2double(get(handles.MotorRPM_Txt,'String')) == 0) || (isDigit == 0)
        %input invalid or zero

        %change textbox colour to red
        set(handles.MotorRPM_Txt,'BackgroundColor',[0.85 0.19 0]);

        %stop = true = GUI not allowed to close
        stop = true;

    else
        %input valid

        %reset backround color to white
        set(handles.MotorRPM_Txt,'BackgroundColor',[1 1 1]);

        %stop = false (initialised conditon not changed) = GUI can close

        %change RPM correct status to true
        rpmCorrect = true;
    end

%Max motor running torque input processing
%*************************************************************************

    %copy motor running torque to variable "holdstring"
    holdstring = get(handles.MotorTorq_Txt,'String');

    %check if each character in "holdstring" is a digits and store result
    %as vector "test1"
    test1 = isstrprop(holdstring,'digit');

    %check if any charaters in "holdstring" are decimal points and store
    %results as vector "test2"
    test2 = zeros(1,length(holdstring));

    for i = 1:length(test1)
        %for each character in "holdstring"

        if holdstring(1,i) == '.'
            %check if charater is a decimal point

            test2(1,i) = 1;
        end
    end

    %add "test1" and "test2", resultant vector will contain a 1 for any
    %character that is either a decimal point or digit and 0 if it is not.
    %Eg [1 1 1 1] = valid number
    %   [1 1 0 1] = string input contains one non digit/decimal point,
    %               invalid number
    test = test1+test2;

    %if product of test is zero = invalid number else number is valid
    isDigit = prod([1,test]);

    %check if number is valid and non-zero
    if (str2double(get(handles.MotorTorq_Txt,'String')) == 0) || (isDigit == 0)
        %input invalid or zero

        %change textbox colour to red
        set(handles.MotorTorq_Txt,'BackgroundColor',[0.85 0.19 0]);

        %stop = true = GUI not allowed to close
        stop = true;

    else
        %input valid

        %reset backround color to white
        set(handles.MotorTorq_Txt,'BackgroundColor',[1 1 1]);

        %calculate power of chosen motor (user rpm * user torq)
        MotorRadSec = (str2double(get(handles.MotorTorq_Txt,'String')))*((2*pi())/60);
        MotorActualPower = (str2double(get(handles.MotorRPM_Txt,'String')))↙
*MotorRadSec;

        if (MotorActualPower < (handles.MotorPow)) && (rpmCorrect == true)
            %User chosen motor is less than required power, user input
            %error

            %change textbox colour to red
            set(handles.MotorTorq_Txt,'BackgroundColor',[0.85 0.19 0]);
            set(handles.MotorRPM_Txt,'BackgroundColor',[0.85 0.19 0]);

            %flagMotor = true = GUI not allowed to close
            flagMotor = true;
        end

        %stop = false (initialised conditon not changed) = GUI can close

    end

%Display or Reset Error Message and Pass Variables
%************************************************************************

    %check if any inputs flagged "stop" by setting it to true
    if stop == true
```

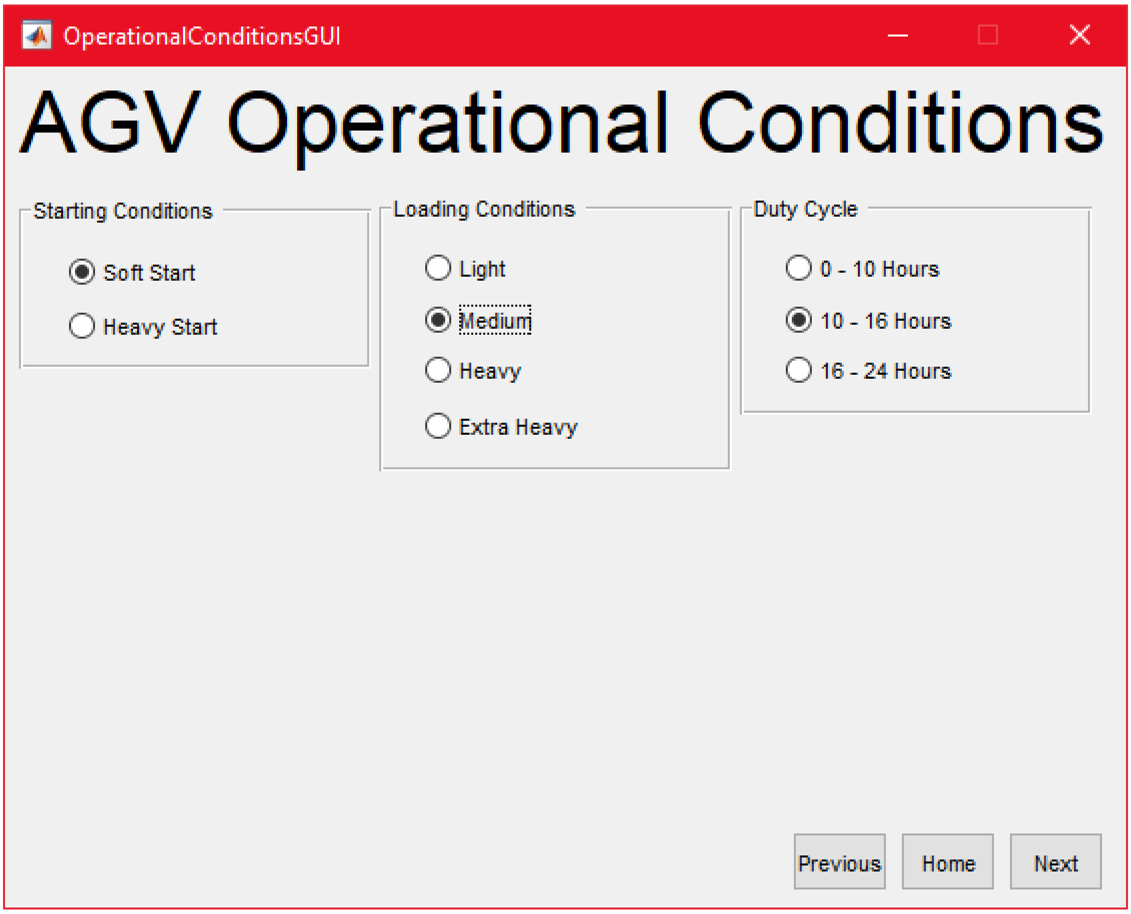
```
        %One or more errors have occured = turn on error message
        set(handles.ErrorMessage,'Visible','on');

    elseif flagMotor == true
        %Motor size incorrect
        set(handles.MotorFlag_Txt,'Visible','on');

    else
        %No errors have occured = turn off error message
        set(handles.ErrorMessage,'Visible','off');

        %Set contitions to call next GUI
        set(handles.Calculate_PB,'UserData',1);

        %Close GUI2
        close;
    end

function ReqMotorP_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to ReqMotorP_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function MotorRPM_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to MotorRPM_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function MotorTorq_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to MotorTorq_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function ReqPowM_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to ReqPowM_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function ReqPowW_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to ReqPowW_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function MaxTorq_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to MaxTorq_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function MaxRPM_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to MaxRPM_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
%8888888888888888888888888888888888888888888888888888888888888888888888888
%Close GUI
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes when user attempts to close GUI6.
function GUI6_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to GUI6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if isequal(get(hObject, 'waitstatus'),'waiting')
    %The GUI is still in UIWAIT, use UIRESUME
    uiresume(hObject);
else
    %The GUI is nolonger waiting and can be closed
    delete(hObject);
end
```

```
function [ GearRatio ] = AGVGearRatioCal( WheelOmega, MotorOmega )
%AGVGearRatioCal Info
%888888888888888888888888888888888888888888888888888888888888888888888888
%   This function calculates the optimum gear ratio required for the user
%   chosen drive motor to power the AGV given the wheels speed and torque
%   requirements

%   Inputs
%   **********************************************************************

%   WheelOmega      = angular velocity of the AGV's drive wheels  (rad/sec)
%   MotorOmega      = angular velocity of the chosen drive motor  (rad/sec)
%   MaxWheelTorque  = drive wheel required torque                 (Nm)
%   MotorRateTorque = maximum rated torque of chosen drive motors (Nm)

%   Outputs
%   **********************************************************************

%   GearRatio       = optimum gear ratio for AGV drive system     (~)




%888888888888888888888888888888888888888888888888888888888888888888888888
%Find Gear Ratio (Angular Velocity Dependent)
%888888888888888888888888888888888888888888888888888888888888888888888888

    %find gear ratio dependent on angular velocities (e1)
    GearRatio = WheelOmega/MotorOmega;

end
```

G.7 Operational Conditions GUI

Operational Conditions GUI:



OperationalConditionsGUI

AGV Operational Conditions

Starting Conditions

● Soft Start

○ Heavy Start

Loading Conditions

○ Light

● Medium

○ Heavy

○ Extra Heavy

Duty Cycle

○ 0 - 10 Hours

● 10 - 16 Hours

○ 16 - 24 Hours

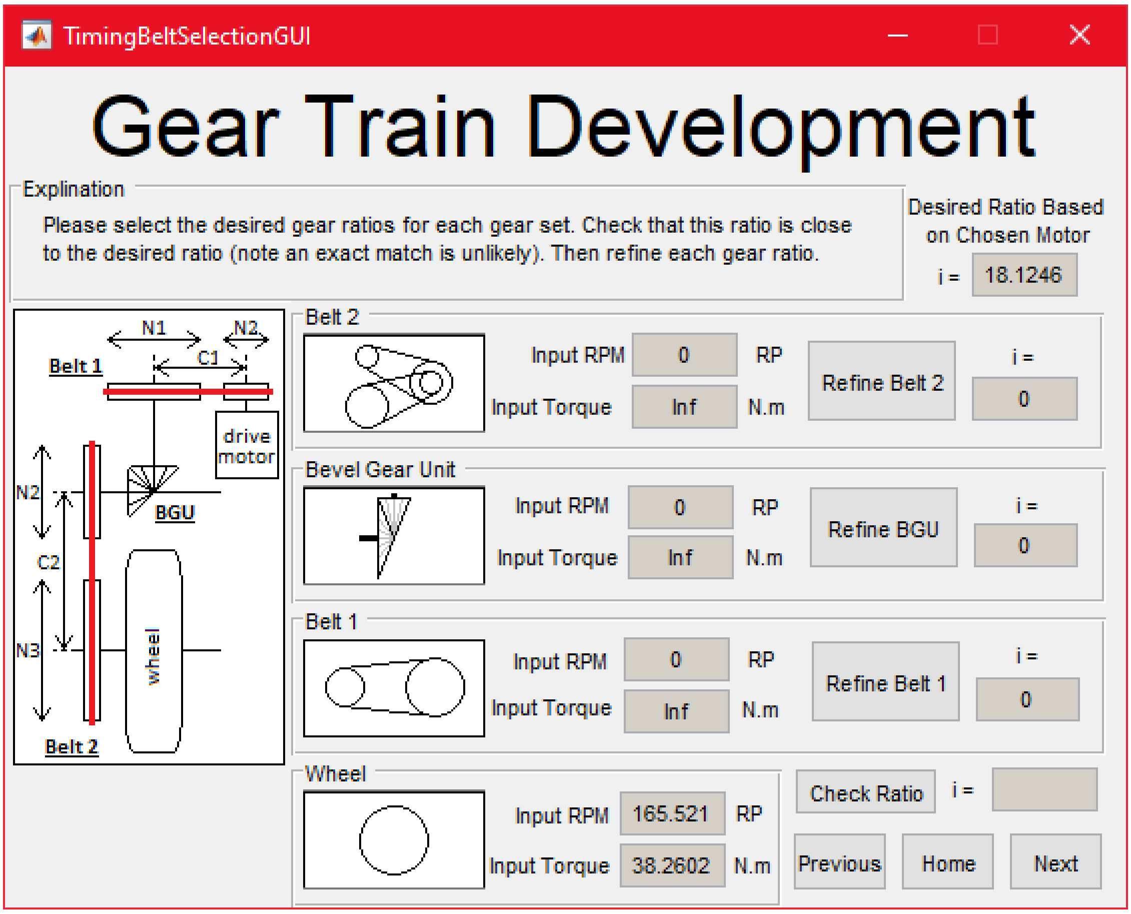
```
function varargout = OperationalConditionsGUI(varargin)
% OPERATIONALCONDITIONSGUI M-file for OperationalConditionsGUI.fig
%      OPERATIONALCONDITIONSGUI, by itself, creates a new OPERATIONALCONDITIONSGUI or↙
raises the existing
%      singleton*.
%
%      H = OPERATIONALCONDITIONSGUI returns the handle to a new↙
OPERATIONALCONDITIONSGUI or the handle to
%      the existing singleton*.
%
%      OPERATIONALCONDITIONSGUI('CALLBACK',hObject,eventData,handles,...) calls the↙
local
%      function named CALLBACK in OPERATIONALCONDITIONSGUI.M with the given input↙
arguments.
%
%      OPERATIONALCONDITIONSGUI('Property','Value',...) creates a new↙
OPERATIONALCONDITIONSGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before OperationalConditionsGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to OperationalConditionsGUI_OpeningFcn via↙
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help OperationalConditionsGUI

% Last Modified by GUIDE v2.5 13-Apr-2017 15:11:59

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @OperationalConditionsGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @OperationalConditionsGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before OperationalConditionsGUI is made visible.
function OperationalConditionsGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to OperationalConditionsGUI (see VARARGIN)

% Choose default command line output for OperationalConditionsGUI
handles.output = hObject;
handles.PowCon = varargin{1};

% Update handles structure
guidata(hObject, handles);

%Force initial conditions
set(handles.StartConUI,'UserData',1);       %Initial soft start
set(handles.LoadingConsUI,'UserData',1);    %Initial light loading
set(handles.DutyCycleUI,'UserData',2);      %Initial 10-16 hour duty cycle

% UIWAIT makes OperationalConditionsGUI wait for user response (see UIRESUME)
uiwait(handles.GUI8a);





%888888888888888888888888888888888888888888888888888888888888888888888888
%Output Function
%888888888888888888888888888888888888888888888888888888888888888888888888

% --- Outputs from this function are returned to the command line.
function varargout = OperationalConditionsGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%(navigation data non-zero)
%************************************************************************

    %varargout{1} = Next GUI to be run
    %varargout{2} = (if = 1) => keep looping through results GUI else don't

    ConditionsVector = [get(handles.StartConUI,'UserData'), get(handles.↙
LoadingConsUI,'UserData'), get(handles.DutyCycleUI,'UserData')];

    if get(handles.Next_PB,'UserData') ~= 0
        %Next PB was pressed to close GUI
        display('Calculate button pressed');
        varargout{1} = 7;
        varargout{2} = 1;
        varargout{3} = ConditionsVector;

    elseif get(handles.Home_PB,'UserData') ~= 0
        %Home PB was pressed to close GUI
        display('End button pressed');
        varargout{1} = 1;
        varargout{2} = 1;
        varargout{3} = 0;

    elseif get(handles.Prev_PB,'UserData') ~= 0
        %Previous PB was pressd to close GUI
        display('Previous button pressed');
        varargout{1} = 5;
        varargout{2} = 1;
        varargout{3} = 0;

    else
        %Unknown PB was pressed or user terminated
        display('Error: program closed unexpectedly')
    end

%Delete GUI
%************************************************************************

    %The figure can now be deleted
    delete(handles.GUI8a);




%888888888888888888888888888888888888888888888888888888888888888888888888
%Prev, Next, Home Pushbuttons
%888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes on button press in Home_PB.
function Home_PB_Callback(hObject, eventdata, handles)
% hObject    handle to Home_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.Home_PB,'UserData',1);

close;

% --- Executes on button press in Next_PB.
function Next_PB_Callback(hObject, eventdata, handles)
% hObject    handle to Next_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.Next_PB,'UserData',1);

close;

% --- Executes on button press in Prev_PB.
function Prev_PB_Callback(hObject, eventdata, handles)
% hObject    handle to Prev_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
set(handles.Prev_PB,'UserData',1);

close;





%888888888888888888888888888888888888888888888888888888888888888888888888
%UI Code Interaction Code(Pushbuttons, Radio Buttons, etc.)
%888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes when selected object is changed in StartConUI.
function StartConUI_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in StartConUI
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue: handle of the previously selected object or empty if none was selected
%   NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)

%Find selected radio button
%************************************************************************

    %On selection of radio button pass tag of selected radio button to
    %variable "SelectedRB"
    SelectedRB = get(eventdata.NewValue,'Tag');

    if strcmp(SelectedRB, 'SoftStartRB') == 1
        %Soft start selected

        %Write chosen value to UI "user data"
        set(handles.StartConUI,'UserData',1);

    elseif strcmp(SelectedRB, 'HeavyStartRB') == 1
        %Heavy start selected

        %Write chosen value to UI "user data"
        set(handles.StartConUI,'UserData',2);

    else
        %Throw error to console
        display('Error in starting conditions selection');

    end


% --- Executes when selected object is changed in LoadingConsUI.
function LoadingConsUI_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in LoadingConsUI
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue: handle of the previously selected object or empty if none was selected
%   NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)

%Find selected radio button
%************************************************************************

    %On selection of radio button pass tag of selected radio button to
    %variable "SelectedRB"
    SelectedRB = get(eventdata.NewValue,'Tag');

    if strcmp(SelectedRB, 'LightRB') == 1
        %Light loading condition selected

        %Write chosen value to UI "user data"
        set(handles.LoadingConsUI,'UserData',1);

    elseif strcmp(SelectedRB, 'MediumRB') == 1
        %Medium loading conditions selected

        %Write chosen value to UI "user data"
        set(handles.LoadingConsUI,'UserData',2);

    elseif strcmp(SelectedRB, 'HeavyRB') == 1
        %Heavy loading conditions selected

        %Write chosen value to UI "user data"
        set(handles.LoadingConsUI,'UserData',3);

    elseif strcmp(SelectedRB, 'ExtraHeavyRB') == 1
        %Extra heavy loading conditions selected

        %Write chosen value to UI "user data"
        set(handles.LoadingConsUI,'UserData',4);

    else
        %Throw error to console
        display('Error in loading conditions selection');

    end


% --- Executes when selected object is changed in DutyCycleUI.
function DutyCycleUI_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in DutyCycleUI
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue: handle of the previously selected object or empty if none was selected
%   NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)

%Find selected radio button
%************************************************************************

    %On selection of radio button pass tag of selected radio button to
    %variable "SelectedRB"
    SelectedRB = get(eventdata.NewValue,'Tag');
    if strcmp(SelectedRB, 'ZeroToTenRB')
        %0 to 10 hour duty cycle selected

        %Write chosen value to UI "user data"
        set(handles.DutyCycleUI,'UserData',1);

    elseif strcmp(SelectedRB, 'TenToSixteenRB')
        %10 to 16 hour duty cycle selected

        %Write chosen value to UI "user data"
        set(handles.DutyCycleUI,'UserData',2);

    elseif strcmp(SelectedRB, 'SixteenToTwentyFourRB')
        %16 to 24 hour duty cycle selected

        %Write chosen value to UI "user data"
        set(handles.DutyCycleUI,'UserData',3);

    else
        %Throw error to console
        display('Error in duty cycle selection');

    end




%8888888888888888888888888888888888888888888888888888888888888888888888888
%Close GUI
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes when user attempts to close GUI8a.
function GUI8a_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to GUI8a (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if isequal(get(hObject, 'waitstatus'),'waiting')
    %The GUI is still in UIWAIT, use UIRESUME
    uiresume(hObject);
else
    %The GUI is nolonger waiting and can be closed
    delete(hObject);
end
```

G.8 AGV Gear Train Ratio Solver

AGV Gear Train Ratio Solver Before:

The figure below illustrates what the GUI looks like when first opened.



TimingBeltSelectionGUI

Gear Train Development

Explination

Please select the desired gear ratios for each gear set. Check that this ratio is close to the desired ratio (note an exact match is unlikely). Then refine each gear ratio.

Desired Ratio Based on Chosen Motor i = 18.1246

Belt 2
Input RPM 0 RP
Input Torque Inf N.m
i = 0

Bevel Gear Unit
Input RPM 0 RP
Input Torque Inf N.m
i = 0

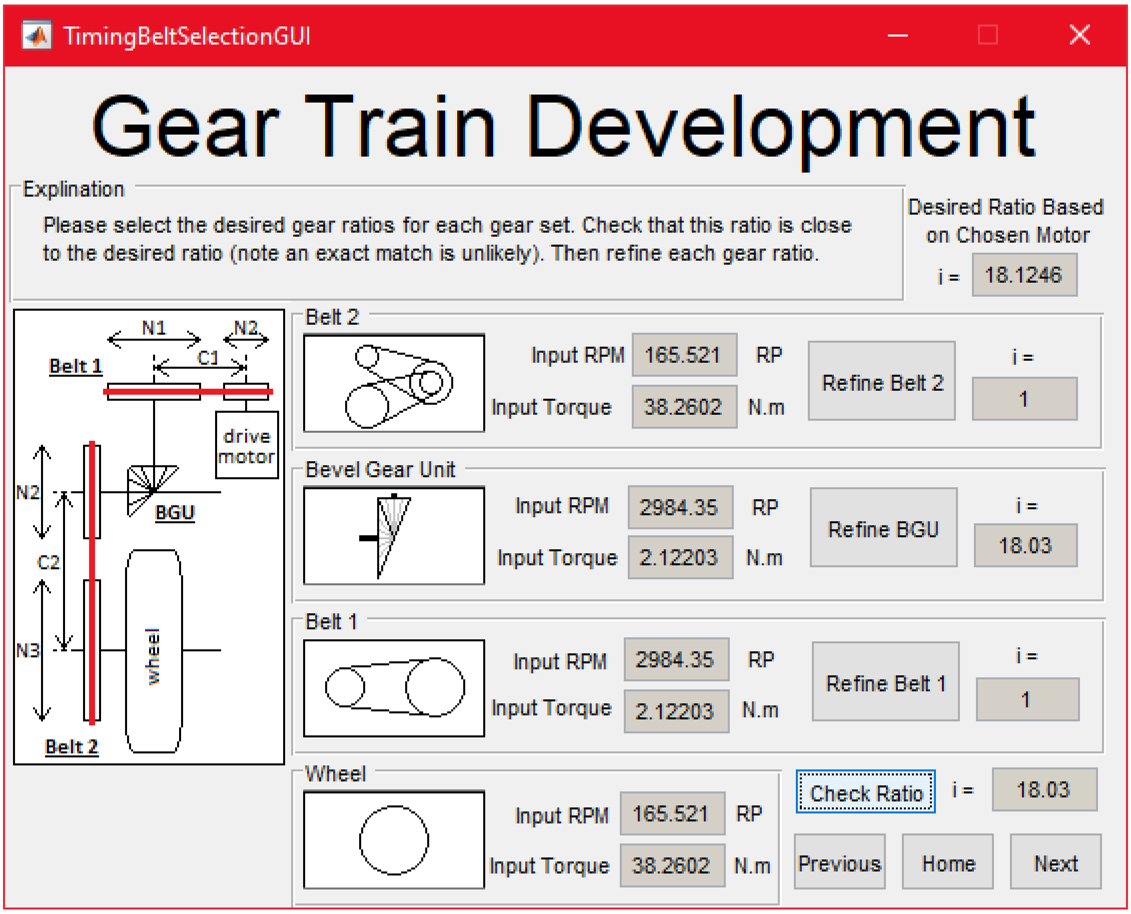
Belt 1
Input RPM 0 RP
Input Torque Inf N.m
i = 0

Wheel
Input RPM 165.521 RP
Input Torque 38.2602 N.m

i =

AGV Gear Train Ratio Solver After:

The figure below illustrates what the GUI looks like after Belt 2, Bevel Gear Unit and Belt 1 refinements have been completed.



TimingBeltSelectionGUI

Gear Train Development

Explination

Please select the desired gear ratios for each gear set. Check that this ratio is close to the desired ratio (note an exact match is unlikely). Then refine each gear ratio.

Desired Ratio Based on Chosen Motor

i = 18.1246

Belt 2

Input RPM 165.521 RP

Input Torque 38.2602 N.m

i = 1

Bevel Gear Unit

Input RPM 2984.35 RP

Input Torque 2.12203 N.m

i = 18.03

Belt 1

Input RPM 2984.35 RP

Input Torque 2.12203 N.m

i = 1

Wheel

Input RPM 165.521 RP

Input Torque 38.2602 N.m

i = 18.03

```
function [ GTOmegaVec, GTtorqueVec ] = AGVGearTrainRatioSolver( GearRatioVector,↙
WheelOmega, WheelTorque )
%AGVGearTrainRatioSolver
%88888888888888888888888888888888888888888888888888888888888888888888888888
%   This function calculates the input RPM and torque for each stage of the
%   gear train given the gear ratio 'i' of the previous gear train and
%   previous RPM and torque

%   Inputs
%   ***********************************************************************

%   GearRatioVector     = an array of the gear ratios for step (unitless)
%   WheelOmega          = max RPM of the AGV's wheels          (rad/s)
%   WheelTorque         = max required wheel torque            (Nm)

%   Outputs
%   **********************************************************************

%   GTomegaVec          = a vector of all the input rpms for stage
%                       =[wheel_Om, belt2_Om, BGU_Om, belt1_Om]     (rad/s)

%   GTtorqueVec         = a vector of all the input Nm for each stage
%                       =[wheel_Tor, belt2_Tor, BGU_Tor, belt1_Tor] (Nm)



%88888888888888888888888888888888888888888888888888888888888888888888888888
%Find input RPMS
%88888888888888888888888888888888888888888888888888888888888888888888888888

    %initialise vector
    Lek = length(GearRatioVector)+1;
    GTOmegaVec = zeros(1,Lek);

    %find input rpm for wheel
    GTOmegaVec(1,1) = WheelOmega;

    %find input rpm for Belt 2, BGU, Belt 1
    for i = 1:3
        %for each previous gear rpm (ith) multiply by ith gear ratio
        GTOmegaVec(1,i+1) = GearRatioVector(1,i)*GTOmegaVec(1,i);
    end




%88888888888888888888888888888888888888888888888888888888888888888888888888
%Find input Torques
%88888888888888888888888888888888888888888888888888888888888888888888888888

    %initialise vector
    GTtorqueVec = zeros(1,Lek);
    %find input torque of wheel
    GTtorqueVec(1,1) = WheelTorque;

    %find input torque for Belt 2, BGU, Belt 1
    for i = 1:3
        %%for each previous gear torque (ith) multiply by ith gear ratio
        GTtorqueVec(1,i+1) = (1/GearRatioVector(1,i))*GTtorqueVec(1,i);
    end

end
```

```
function varargout = GearDevelopmentGUI(varargin)
%GEARDEVELOPMENTGUI M-file for GearDevelopmentGUI.fig
%      GEARDEVELOPMENTGUI, by itself, creates a new GEARDEVELOPMENTGUI or raises the↙
existing
%      singleton*.
%
%      H = GEARDEVELOPMENTGUI returns the handle to a new GEARDEVELOPMENTGUI or the↙
handle to
%      the existing singleton*.
%
%      GEARDEVELOPMENTGUI('Property','Value',...) creates a new GEARDEVELOPMENTGUI↙
using the
%      given property value pairs. Unrecognized properties are passed via
%      varargin to GearDevelopmentGUI_OpeningFcn.  This calling syntax produces a
%      warning when there is an existing singleton*.
%
%      GEARDEVELOPMENTGUI('CALLBACK') and GEARDEVELOPMENTGUI('CALLBACK',hObject,...)↙
call the
%      local function named CALLBACK in GEARDEVELOPMENTGUI.M with the given input
%      arguments.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GearDevelopmentGUI

% Last Modified by GUIDE v2.5 06-Apr-2017 11:41:55

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GearDevelopmentGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @GearDevelopmentGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT




%8888888888888888888888888888888888888888888888888888888888888888888888888
%Initialise
```

```
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes just before GearDevelopmentGUI is made visible.
function GearDevelopmentGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   unrecognized PropertyName/PropertyValue pairs from the
%            command line (see VARARGIN)

% Choose default command line output for GearDevelopmentGUI
handles.output = hObject;
handles.GearRatio = varargin{1};
handles.GearTrainVector = varargin{2};
handles.GTrpm = varargin{3};
handles.GTtor = varargin{4};
handles.GearTrainStatus = varargin{5};
handles.MotorOmega = varargin{6};
handles.MotorTorque = varargin{7};

% Update handles structure
guidata(hObject, handles);

% Setup initial images
imshow('TimingBeltDiagram.png','Parent', handles.Axis1);
imshow('Belt1.png','Parent', handles.Axis2);
imshow('BGU.png','Parent', handles.Axis3);
imshow('Belt2.png','Parent', handles.Axis4);
imshow('Wheel.png','Parent', handles.Axis5);

% Disable Warning Messages
%set(handles.Warning_Txt,'Visible','off');

%Update all txt boxes
set(handles.WheelRPM_Txt,'String',handles.GTrpm(1,1)*(60/(2*pi())));
set(handles.WheelTor_Txt,'String',handles.GTtor(1,1));

set(handles.Belt2I_Txt,'String',handles.GearTrainVector(1,1));
set(handles.Belt2RPM_Txt,'String',handles.GTrpm(1,2)*(60/(2*pi())));
set(handles.Belt2Tor_Txt,'String',handles.GTtor(1,2));

set(handles.BGUI_Txt,'String',handles.GearTrainVector(1,2));
set(handles.BGURPM_Txt,'String',handles.GTrpm(1,3)*(60/(2*pi())));
set(handles.BGUTor_Txt,'String',handles.GTtor(1,3));

set(handles.Belt1I_Txt,'String',handles.GearTrainVector(1,3));
set(handles.Belt1RPM_Txt,'String',handles.GTrpm(1,4)*(60/(2*pi())));
set(handles.Belt1Tor_Txt,'String',handles.GTtor(1,4));

set(handles.ActualRatio_Txt,'String',handles.GearRatio^-1);

%Reset "has ratio been checked"

set(handles.CheckRatio_PB,'UserData',0);
```

```
% UIWAIT makes GearDevelopmentGUI wait for user response (see UIRESUME)
 uiwait(handles.GUI7);




%8888888888888888888888888888888888888888888888888888888888888888888888888
%Output Function
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Outputs from this function are returned to the command line.
function varargout = GearDevelopmentGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%(navigation data non-zero)
%*************************************************************************

    %varargout{1} = Next GUI to be run
    %varargout{2} = (if = 1) => keep looping through results GUI else don't

    if get(handles.Next_PB,'UserData') ~= 0
        %Next PB was pressed to close GUI
        display('Calculate button pressed');
        varargout{1} = 12;
        varargout{2} = 1;

    elseif get(handles.Home_PB,'UserData') ~= 0
        %Home PB was pressed to close GUI
        display('End button pressed');
        varargout{1} = 1;
        varargout{2} = 1;

    elseif get(handles.Prev_PB,'UserData') ~= 0
        %Previous PB was pressd to close GUI
        display('Previous button pressed');
        varargout{1} = 6;
        varargout{2} = 1;

    elseif get(handles.RefineBelt2_PB,'UserData') ~= 0
        %Refine Belt 2 PB was pressed to close GUI
        display('Refine belt 2 pressed');
        varargout{1} = 8;
        varargout{2} = 1;

    elseif get(handles.RefineBGU_PB,'UserData') ~= 0
        %Refine BGU PB was pressed to close GUI
        display('Refine BGU pressed');
        varargout{1} = 10;
        varargout{2} = 1;
    elseif get(handles.RefineBelt1_PB,'UserData') ~= 0
        %Refine Belt 1 PB was pressed to close GUI
        display('Refine belt 1 pressed');
        varargout{1} = 11;
        varargout{2} = 1;

    else
        %Unknown PB was pressed or user terminated
        display('Error: program closed unexpectedly')
    end

%Delete GUI
%*************************************************************************

    %The figure can now be deleted
    delete(handles.GUI7);





%888888888888888888888888888888888888888888888888888888888888888888888888
%Prev, Next, Home Pushbuttons
%888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes on button press in Home_PB.
function Home_PB_Callback(hObject, eventdata, handles)
% hObject    handle to Home_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.Home_PB,'UserData',1);

close;

% --- Executes on button press in Next_PB.
function Next_PB_Callback(hObject, eventdata, handles)
% hObject    handle to Next_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.GearTrainStatus >= 40
    %All gear stages complete

    if get(handles.CheckRatio_PB,'UserData') == 1
        %Ratio check has been run

        if get(handles.CheckRatio_Txt,'UserData') == 1
            %Torque requirements met

            set(handles.Next_PB,'UserData',1);

            close;

        else
```

```
            %Torque requirements not met
            
            %Write warning message to window
            WarnMessage = 'Correct torque requirement first';
            set(handles.Warning_Txt,'String',WarnMessage);
            
        end
        
    else
        %Ratio check has not been run
        
        %Write warning message to window
        WarnMessage = 'Run ratio check first';
        set(handles.Warning_Txt,'String',WarnMessage);
        
    end
    
else
    %All gear stages not complete
    
    %Write warning message to window
    WarnMessage = 'Please fully define belt 1, belt 2 & BGU first';
    set(handles.Warning_Txt,'String',WarnMessage);
    
end



% --- Executes on button press in Prev_PB.
function Prev_PB_Callback(hObject, eventdata, handles)
% hObject    handle to Prev_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.Prev_PB,'UserData',1);

close;





%888888888888888888888888888888888888888888888888888888888888888888888888
%Create Textboxes and Objects
%888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes during object creation, after setting all properties.
function Belt2RPM_txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Belt2RPM_txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function CheckRatio_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to CheckRatio_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function ActualRatio_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ActualRatio_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function Belt2Tor_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Belt2Tor_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function Belt2I_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Belt2I_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes during object creation, after setting all properties.
function BGURPM_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to BGURPM_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function BGUTor_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to BGUTor_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function BGUI_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to BGUI_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function Belt1RPM_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Belt1RPM_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function WheelRPM_Txt_CreateFcn(hObject, eventdata, handles)
```

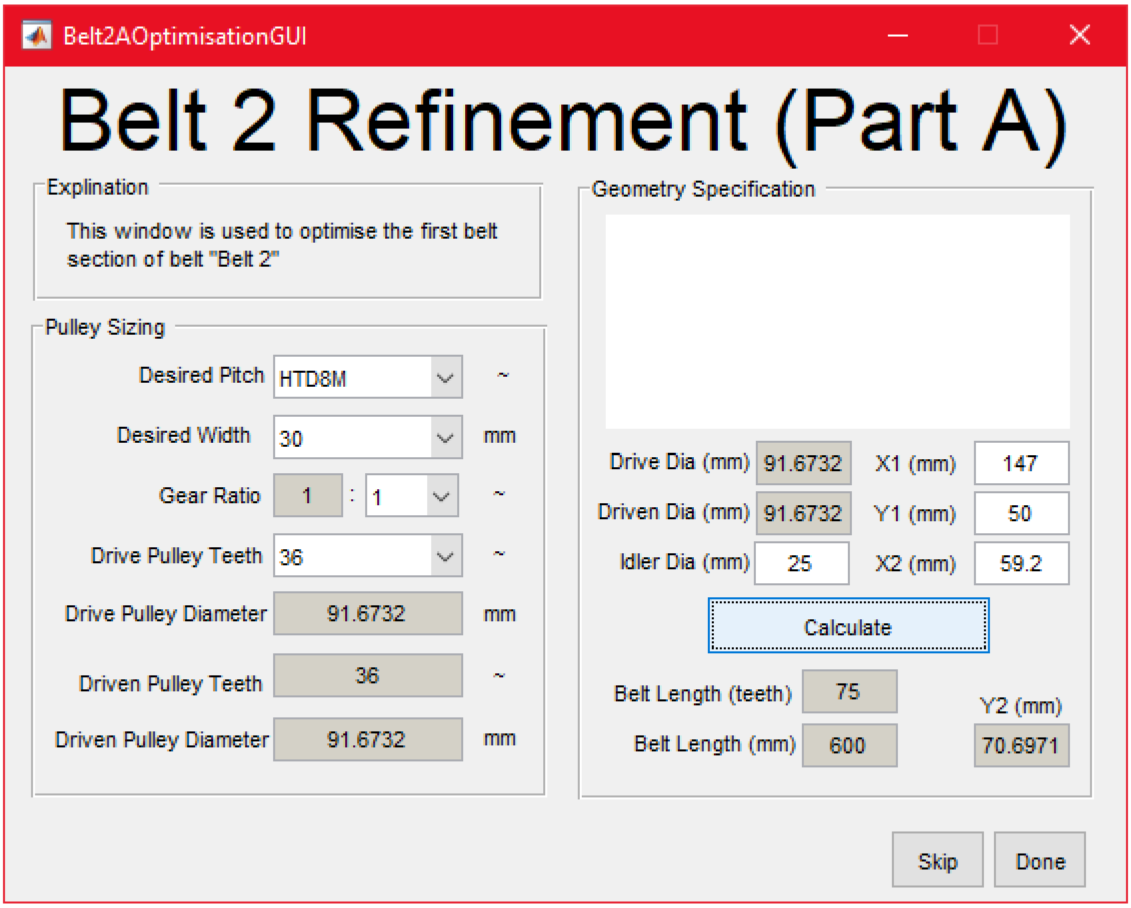
```
% hObject    handle to WheelRPM_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function Belt1Tor_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Belt1Tor_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function Belt1I_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Belt1I_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function WheelTor_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to WheelTor_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function Belt2RPM_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Belt2RPM_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end





%8888888888888888888888888888888888888888888888888888888888888888888888888
%UI Code Interaction Code(Pushbuttons, Radio Buttons, etc.)
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes on button press in RefineBelt2_PB.
function RefineBelt2_PB_Callback(hObject, eventdata, handles)
% hObject    handle to RefineBelt2_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Disable Warning Messages
set(handles.Warning_Txt,'String','');

if handles.GearTrainStatus >= 0
    %Can complete this stage

    set(handles.RefineBelt2_PB,'UserData',1);

    close;
end



% --- Executes on button press in RefineBGU_PB.
function RefineBGU_PB_Callback(hObject, eventdata, handles)
% hObject    handle to RefineBGU_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Disable Warning Messages
set(handles.Warning_Txt,'String','');

if handles.GearTrainStatus >= 20

    set(handles.RefineBGU_PB,'UserData',1);

    close;
else

    %Write Warning Message
    set(handles.Warning_Txt,'String','Please fully define belt 2 first');

end
```

```
% --- Executes on button press in RefineBelt1_PB.
function RefineBelt1_PB_Callback(hObject, eventdata, handles)
% hObject    handle to RefineBelt1_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Disable Warning Messages
set(handles.Warning_Txt,'String','');

if handles.GearTrainStatus >= 30

    set(handles.RefineBelt1_PB,'UserData',1);

    close;

else

    %Write Wanring Mesage
    set(handles.Warning_Txt,'String','Please fully define belt 2 & BGU first');

end

% --- Executes on button press in CheckRatio_PB.
function CheckRatio_PB_Callback(hObject, eventdata, handles)
% hObject    handle to CheckRatio_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Disable Warning Messages
set(handles.Warning_Txt,'String',' ');

if handles.GearTrainStatus >= 40

    %set ratio has been checked to true
    set(handles.CheckRatio_PB,'UserData',1);

    ActualGearRatio = handles.GearTrainVector(1,1) * handles.GearTrainVector(1,2) *↙
handles.GearTrainVector(1,3);

    set(handles.CheckRatio_Txt,'String',ActualGearRatio);

    if handles.GTtor(1,4) <= handles.MotorTorque
        %Input torque less than the max the motor can supply

        %Set backround of txt box to static grey
        set(handles.CheckRatio_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);

        %Allow next button to exit window
        set(handles.CheckRatio_Txt,'UserData',1);

    else
        %Input torque exceedes what motor is capable of
        %Set backround of txt box to red
        set(handles.CheckRatio_Txt,'BackgroundColor','red');

        % Enable Warning Messages
        set(handles.Warning_Txt,'Visible','on');

        %Create warning message
        txt1 = 'Torque (';
        txt2 = handles.MotorTorque;
        txt3 = ' N.m) provided by the chosen motor is less than the needed torque (';
        txt4 = handles.GTtor(1,4);
        txt5 = ' N.m)';

        WarnMessage = strcat(txt1, txt2, txt3, txt4, txt5);

        %Write warning message to window
        set(handles.Warning_Txt,'String',WarnMessage);

        %Prevent next button from exiting window
        set(handles.CheckRatio_Txt,'UserData',0);

    end

else

    %Write Wanring Mesage
    set(handles.Warning_Txt,'String','Please fully define belt 1, belt 2 & BGU↙
first');

end

function CheckRatio_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to CheckRatio_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of CheckRatio_Txt as text
%        str2double(get(hObject,'String')) returns contents of
%        CheckRatio_Txt as a double

function ActualRatio_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to ActualRatio_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ActualRatio_Txt as text
%        str2double(get(hObject,'String')) returns contents of ActualRatio_Txt as a↙
double



function Belt2RPM_txt_Callback(hObject, eventdata, handles)
% hObject    handle to Belt2RPM_txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Belt2RPM_txt as text
%        str2double(get(hObject,'String')) returns contents of Belt2RPM_txt as a↙
double

function Belt2Tor_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to Belt2Tor_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Belt2Tor_Txt as text
%        str2double(get(hObject,'String')) returns contents of Belt2Tor_Txt as a↙
double



function Belt2I_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to Belt2I_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Belt2I_Txt as text
%        str2double(get(hObject,'String')) returns contents of Belt2I_Txt as a double

function BGURPM_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to BGURPM_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of BGURPM_Txt as text
%        str2double(get(hObject,'String')) returns contents of BGURPM_Txt as a double

function BGUTor_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to BGUTor_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of BGUTor_Txt as text
%        str2double(get(hObject,'String')) returns contents of BGUTor_Txt as a double

function BGUI_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to BGUI_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of BGUI_Txt as text
%        str2double(get(hObject,'String')) returns contents of BGUI_Txt as a double

function Belt1RPM_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to Belt1RPM_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Belt1RPM_Txt as text
%        str2double(get(hObject,'String')) returns contents of Belt1RPM_Txt as a↙
double

function Belt1Tor_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to Belt1Tor_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Belt1Tor_Txt as text
%        str2double(get(hObject,'String')) returns contents of Belt1Tor_Txt as a↙
double



function Belt1I_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to Belt1I_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Belt1I_Txt as text
%        str2double(get(hObject,'String')) returns contents of Belt1I_Txt as a double

function Belt2RPM_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to Belt2RPM_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Belt2RPM_Txt as text
%        str2double(get(hObject,'String')) returns contents of Belt2RPM_Txt as
%        a double

function WheelRPM_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to WheelRPM_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of WheelRPM_Txt as text
%        str2double(get(hObject,'String')) returns contents of WheelRPM_Txt as a↙
double

function WheelTor_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to WheelTor_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of WheelTor_Txt as text
%        str2double(get(hObject,'String')) returns contents of WheelTor_Txt as a↙
double




%8888888888888888888888888888888888888888888888888888888888888888888888888
%Close GUI
```

```
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes when user attempts to close GUI7.
function GUI7_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to GUI7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if isequal(get(hObject, 'waitstatus'),'waiting')
    %The GUI is still in UIWAIT, use UIRESUME
    uiresume(hObject);
else
    %The GUI is nolonger waiting and can be closed
    delete(hObject);
end
```

G.9 Belt 2A Optimisation GUI

Belt 2A Optimisation GUI:



Belt2AOptimisationGUI

Belt 2 Refinement (Part A)

Explination

This window is used to optimise the first belt section of belt "Belt 2"

Pulley Sizing

Desired Pitch HTD8M ~

Desired Width 30 mm

Gear Ratio 1 : 1 ~

Drive Pulley Teeth 36 ~

Drive Pulley Diameter 91.6732 mm

Driven Pulley Teeth 36 ~

Driven Pulley Diameter 91.6732 mm

Geometry Specification

Drive Dia (mm) 91.6732 X1 (mm) 147

Driven Dia (mm) 91.6732 Y1 (mm) 50

Idler Dia (mm) 25 X2 (mm) 59.2

Belt Length (teeth) 75

Belt Length (mm) 600

Y2 (mm) 70.6971

```
function varargout = Belt2AOptimisationGUI(varargin)
% BELT2AOPTIMISATIONGUI M-file for Belt2AOptimisationGUI.fig
%      BELT2AOPTIMISATIONGUI, by itself, creates a new BELT2AOPTIMISATIONGUI or raises↙
the existing
%      singleton*.
%
%      H = BELT2AOPTIMISATIONGUI returns the handle to a new BELT2AOPTIMISATIONGUI or↙
the handle to
%      the existing singleton*.
%
%      BELT2AOPTIMISATIONGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in BELT2AOPTIMISATIONGUI.M with the given input↙
arguments.
%
%      BELT2AOPTIMISATIONGUI('Property','Value',...) creates a new↙
BELT2AOPTIMISATIONGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Belt2AOptimisationGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Belt2AOptimisationGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Belt2AOptimisationGUI

% Last Modified by GUIDE v2.5 26-Apr-2017 11:27:35

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Belt2AOptimisationGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @Belt2AOptimisationGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



%888888888888888888888888888888888888888888888888888888888888888888888888
%Initialise
%888888888888888888888888888888888888888888888888888888888888888888888888
```

```
% --- Executes just before Belt2AOptimisationGUI is made visible.
function Belt2AOptimisationGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Belt2AOptimisationGUI (see VARARGIN)

% Choose default command line output for Belt2AOptimisationGUI
handles.output = hObject;
handles.HTD5M = varargin{1};
handles.HTD8M = varargin{2};
handles.HTD14M = varargin{3};
handles.BeltPitch = varargin{4};
handles.RPM = varargin{5};
handles.Power = varargin{6};
handles.ConditionsVec = varargin{7};
handles.PowerConditions = varargin{8};
handles.DataExists = varargin{9};

% Update handles structure
guidata(hObject, handles);

% Populate list boxes
set(handles.BeltPitch_Pop,'String',varargin{4});

%set initial list box colours
set(handles.BeltWidth_Pop,'BackgroundColor',[0.94, 0.94, 0.94]);
set(handles.DrivePTeeth_Pop,'BackgroundColor',[0.94, 0.94, 0.94]);
set(handles.DrivePDia_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.One_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.GRatio_Pop,'BackgroundColor',[0.94, 0.94, 0.94]);
set(handles.DrivenPTeeth_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.DrivenPDia_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.DA_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.DB_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.BeltTeeth_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.BeltLength_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.ActualY2_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);

% Setup initial images
imshow('Belt2PartA.png','Parent', handles.BeltGeo_Axis);

%Initialise warning text
set(handles.WarningTxt, 'ForegroundColor', 'red');

%set data stage for pulley sizing
set(handles.PulleySize_UI,'UserData',0);

% UIWAIT makes Belt2AOptimisationGUI wait for user response (see UIRESUME)
uiwait(handles.GUI8);
```

```
%888888888888888888888888888888888888888888888888888888888888888888888888
%Output Function
%888888888888888888888888888888888888888888888888888888888888888888888888

% --- Outputs from this function are returned to the command line.
function varargout = Belt2AOptimisationGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%(navigation data non-zero)
%************************************************************************

    %varargout{1} = Next GUI to be run
    %varargout{2} = (if = 1) => keep looping through results GUI else don't

    if get(handles.Done_PB,'UserData') ~= 0
        %Done PB was pressed to close GUI
        display('Done PB pressed');
        varargout{1} = 9;
        varargout{2} = 1;

        %Results Vector Data Creation
        %*****************************************************************

            %Get belt pitch index (index)
            index  = get(handles.BeltPitch_Pop,'Value');

            if index == 1
                %HTD5M belt selected
                BeltPitch = 5;

            elseif index == 2
                %HTD8M belt selected
                BeltPitch = 8;

            elseif index == 3
                %HTD14M belt selected
                BeltPitch = 14;

            else
                %throw error to console
                display('Output data creation error');
            end

            %Get belt width value (mm)
            contents = get(handles.BeltWidth_Pop,'String');
            index = get(handles.BeltWidth_Pop,'Value');
            DesiredBeltWidth = str2num(contents(index,:));

            %Get belt ratio (unitless)
            contents = get(handles.GRatio_Pop,'String');
            index = get(handles.GRatio_Pop,'Value');
            GearRatio = str2num(contents(index,:));

            %Get drive pulley teeth (#)
            contents = get(handles.DrivePTeeth_Pop,'String');
            index = get(handles.DrivePTeeth_Pop,'Value');
            DrivePulleyTeeth = str2num(contents(index,:));

            %Get drive pulley diameter (mm)
            DrivePulleyDia = str2num(get(handles.DrivePDia_Txt,'String'));

            %Get driven pulley teeth (#)
            DrivenPulleyTeeth = str2num(get(handles.DrivenPTeeth_Txt,'String'));

            %Get driven pulley diameter (mm)
            DrivenPulleyDia = str2num(get(handles.DrivenPDia_Txt,'String'));

            %Get X1 (mm)
            X1 = str2num(get(handles.X1_Txt,'String'));

            %Get Y1 (mm)
            Y1 = str2num(get(handles.Y1_Txt,'String'));

            %Get X2 (mm)
            X2 = str2num(get(handles.X2_Txt,'String'));

            %Get Y2 (mm)
            Y2 = str2num(get(handles.ActualY2_Txt,'String'));

            %Get belt length (teeth)
            BeltTeeth = str2num(get(handles.BeltTeeth_Txt,'String'));

            %Get belt length (mm)
            BeltLength = str2num(get(handles.BeltLength_Txt,'String'));

        %Create Output Vector Containing Results
        %*****************************************************************

            Belt2AResultsVector = [BeltPitch, DesiredBeltWidth, GearRatio,↙
DrivePulleyTeeth, DrivePulleyDia, DrivenPulleyTeeth, DrivenPulleyDia, X1, Y1, X2, Y2,↙
BeltTeeth, BeltLength];

            varargout{3} = Belt2AResultsVector;

    elseif get(handles.Skip_PB,'UserData') ~= 0
        %Skip PB was pressed to close GUI
        display('Skip PB pressed');
        varargout{1} = 9;
        varargout{2} = 1;
        varargout{3} = 0;   %This prevents data being overwritten

    else
        %Unknown PB was pressed or user terminated
        display('Error: program closed unexpectedly')
    end
```

```
%Delete GUI
%*************************************************************************

    %The figure can now be deleted
    delete(handles.GUI8);




%888888888888888888888888888888888888888888888888888888888888888888888888
%Done Pushbutton
%888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes on button press in Done_PB.
function Done_PB_Callback(hObject, eventdata, handles)
% hObject    handle to Done_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if get(handles.PulleySize_UI,'UserData') >= 50
    %All stages complete
    set(handles.Done_PB,'UserData',1);

    close;

else
    %Not all stages complete

    %Write error to gui error txt
    WarningMessage = 'Not all data entered or calculated';
    set(handles.WarningTxt, 'String', WarningMessage);

end

% --- Executes on button press in Skip_PB.
function Skip_PB_Callback(hObject, eventdata, handles)
% hObject    handle to Skip_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.DataExists == 1
    %Data already exists

    set(handles.Skip_PB,'UserData',1);

    close;

else
    %Data doesn't exist

    %Write error to gui error txt
    WarningMessage = 'Data does not exist yet, cannot skip this step';
    set(handles.WarningTxt, 'String', WarningMessage);

end
```

```
%888888888888888888888888888888888888888888888888888888888888888888888888
%Create Textboxes and Objects
%888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function popupmenu3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function GRatio_Pop_CreateFcn(hObject, eventdata, handles)
% hObject    handle to GRatio_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function BeltWidth_Pop_CreateFcn(hObject, eventdata, handles)
% hObject    handle to BeltWidth_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function BeltPitch_Pop_CreateFcn(hObject, eventdata, handles)
% hObject    handle to BeltPitch_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function DrivePTeeth_Pop_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DrivePTeeth_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function DrivenPTeeth_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DrivenPTeeth_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function DrivePDia_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DrivePDia_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function DrivenPDia_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DrivenPDia_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function One_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to One_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function WarningTxt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to WarningTxt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% --- Executes during object creation, after setting all properties.
function DA_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DA_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function DB_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DB_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function DC_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DC_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function X1_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to X1_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function Y1_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Y1_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function X2_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to X2_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function Y2_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Y2_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function ActualY2_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ActualY2_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all properties.
function BeltTeeth_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to BeltTeeth_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function BeltLength_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to BeltLength_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

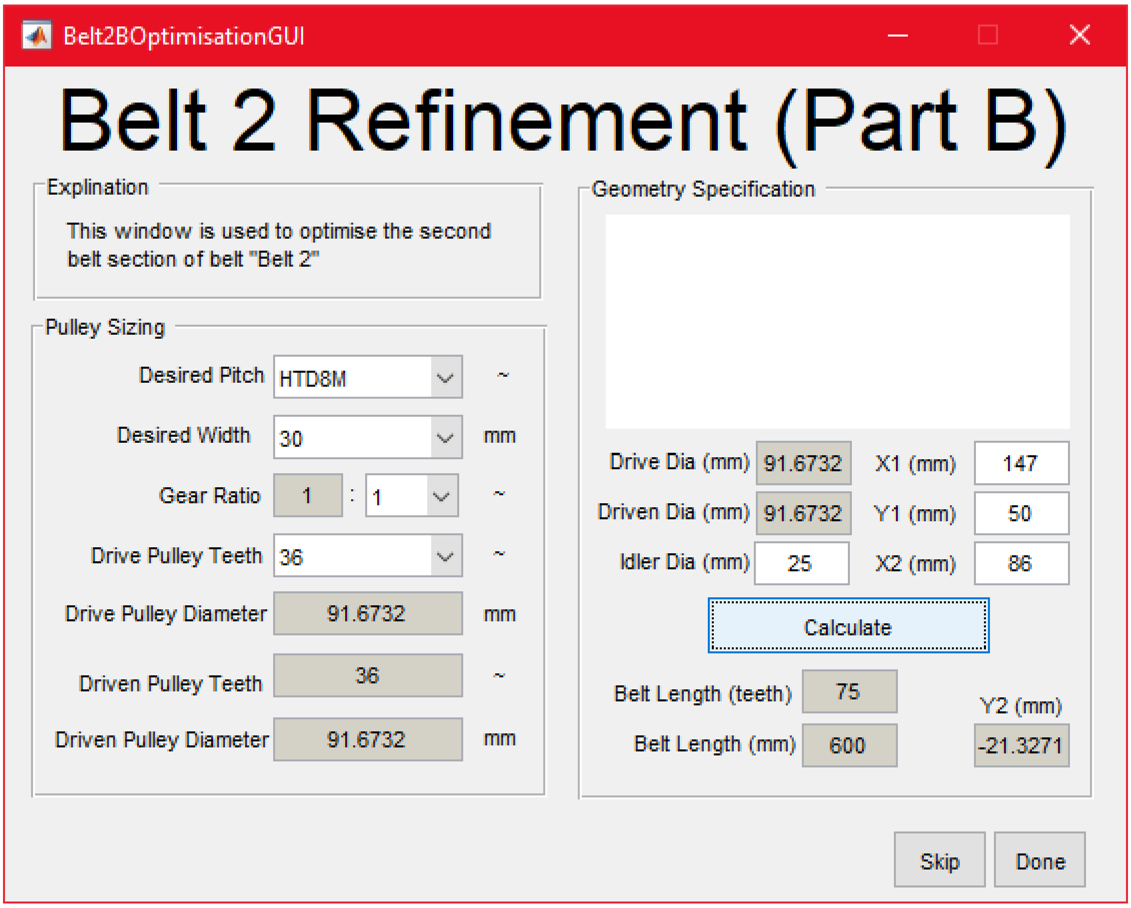
```
%888888888888888888888888888888888888888888888888888888888888888888888888
%UI Code Interaction Code(Pushbuttons, Radio Buttons, etc.)
%888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes on selection change in BeltPitch_Pop.
function BeltPitch_Pop_Callback(hObject, eventdata, handles)
% hObject    handle to BeltPitch_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    %set correct colours (grey)
    %empty = {'Empty','Empty'};
    set(handles.GRatio_Pop,'BackgroundColor',[0.94, 0.94, 0.94]);
    set(handles.DrivePTeeth_Pop,'BackgroundColor',[0.94, 0.94, 0.94]);

    Checkvalue = 0;

    %read what value chosen from derired pitch pop up
    Checkvalue = get(handles.BeltPitch_Pop,'Value');
    %          = 1 = HTD5M
    %          = 2 = HTD8M
    %          = 3 = HTD14M

    %Run task to impliment avaliable widths
    if Checkvalue == 1
        %HTD5M was selected

        temp = handles.HTD5M.Pulley(1,:);

        %set correct colours (white)
        set(handles.BeltWidth_Pop,'BackgroundColor','white');

        %Write list to pop menu
        set(handles.BeltWidth_Pop,'String',temp);

    elseif Checkvalue == 2
        %HTD8M was selected

        temp = handles.HTD8M.Pulley(1,:);

        %set correct colours (white)
        set(handles.BeltWidth_Pop,'BackgroundColor','white');

        %Write list to pop menu
        set(handles.BeltWidth_Pop,'String',temp);

    elseif Checkvalue == 3
        %HTD14M was selected
        temp = handles.HTD14M.Pulley(1,:);

        %set correct colours (white)
        set(handles.BeltWidth_Pop,'BackgroundColor','white');

        %Write list to pop menu
        set(handles.BeltWidth_Pop,'String',temp);

    else
        %write error to console
        display('Selected belt does not mach any in data files');
    end

    %reset warning
    set(handles.WarningTxt, 'String', ' ');

    %Set correct data stage
    set(handles.PulleySize_UI,'UserData',10);


% --- Executes on selection change in BeltWidth_Pop.
function BeltWidth_Pop_Callback(hObject, eventdata, handles)
% hObject    handle to BeltWidth_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    %Run belt width selection only if previous stages complete
    if get(handles.PulleySize_UI,'UserData') >= 10
        %can proceede

        %set correct colours (grey)
        set(handles.DrivePTeeth_Pop,'BackgroundColor',[0.94, 0.94, 0.94]);

        %Find which pitch chosen
        Checkvalue = get(handles.BeltPitch_Pop,'Value');

        %Retrieve list of avaliable gear ratios

        if Checkvalue == 1
            %HTD5M was selected

            [RatioTable] = AGVBeltRatioSorter(handles.HTD5M.BeltCenter);

        elseif Checkvalue == 2
            %HTD8M was selected

            [RatioTable] = AGVBeltRatioSorter(handles.HTD8M.BeltCenter);

        elseif Checkvalue == 3
            %HTD14M was selected

            [RatioTable] = AGVBeltRatioSorter(handles.HTD14M.BeltCenter);

        else
            %Write error to console
            display('An error has occured in the ratio determination');

        end

        %reset warning
        set(handles.WarningTxt, 'String', ' ');

        %Set correct colours (white)
        set(handles.GRatio_Pop,'BackgroundColor',[1, 1, 1]);

        %Set correct data stage
        set(handles.PulleySize_UI,'UserData',20);

        %Write gear ratios to pop up menu
        set(handles.GRatio_Pop,'String',RatioTable);

    end

function One_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to One_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on selection change in GRatio_Pop.
function GRatio_Pop_Callback(hObject, eventdata, handles)
% hObject    handle to GRatio_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


    %Run belt width selection only if previous stages complete
    if get(handles.PulleySize_UI,'UserData') >= 20
        %can proceede

        %Get belt pitch index
        BeltPitch = get(handles.BeltPitch_Pop,'Value');

        %Get belt width value (mm)
        contents = get(handles.BeltWidth_Pop,'String');
        index = get(handles.BeltWidth_Pop,'Value');
        BeltWidth = str2num(contents(index,:));

        %Get belt ratio (unitless)
        contents = get(handles.GRatio_Pop,'String');
        index = get(handles.GRatio_Pop,'Value');
        GearTrainI = str2num(contents(index,:));

        %Get RPM (rad/sec)
        RPM = handles.RPM;

        %Get power (Watts)
        Power = handles.Power;

        %Run Drive Pulley Finder
        if BeltPitch == 1
```

```
            %HTD5M was selected

            [ PowerDrivesAccpt ] = AGVDrivePulleyPowerChecker(RPM, Power, BeltWidth,↙
(handles.HTD5M.Pulley), (handles.HTD5M.Power), (handles.HTD5M.PowerMult) ,GearTrainI,↙
(handles.ConditionsVec), (handles.PowerConditions));
            [ AcceptableDrives ] = AGVDrivePulleyRatioChecker(GearTrainI, (handles.↙
HTD5M.BeltCenter), PowerDrivesAccpt );


        elseif BeltPitch == 2
            %HTD8M was selected

            [ PowerDrivesAccpt ] = AGVDrivePulleyPowerChecker(RPM, Power, BeltWidth,↙
(handles.HTD8M.Pulley), (handles.HTD8M.Power), (handles.HTD8M.PowerMult), GearTrainI,↙
(handles.ConditionsVec), (handles.PowerConditions));
            [ AcceptableDrives ] = AGVDrivePulleyRatioChecker(GearTrainI, (handles.↙
HTD8M.BeltCenter), PowerDrivesAccpt );

        elseif BeltPitch == 3
            %HTD14M was selected

            [ PowerDrivesAccpt ] = AGVDrivePulleyPowerChecker(RPM, Power, BeltWidth,↙
(handles.HTD14M.Pulley), (handles.HTD14M.Power), (handles.HTD14M.PowerMult),↙
GearTrainI, (handles.ConditionsVec), (handles.PowerConditions));
            [ AcceptableDrives ] = AGVDrivePulleyRatioChecker(GearTrainI, (handles.↙
HTD14M.BeltCenter), PowerDrivesAccpt );

        end

        %Check if any drive pulleys are avaliable for chosen gear ratio

        if AcceptableDrives(1,1) ~= 0
            %The first acceptable drive pulley isn't zero => it exists

            %reset warning
            set(handles.WarningTxt, 'String', ' ');

            %Set correct colours (white)
            set(handles.DrivePTeeth_Pop,'BackgroundColor',[1, 1, 1]);

            %Write avaliable drivers to pop up menu
            set(handles.DrivePTeeth_Pop,'String',AcceptableDrives(1,:));

            %Write matching driven vector to UserData of "DrivenPTeeth_Txt"
            set(handles.DrivenPTeeth_Txt,'UserData',AcceptableDrives(2,:));

            %Set correct data stage
            set(handles.PulleySize_UI,'UserData',30);

        elseif AcceptableDrives(1,1) == 0
            %The first acceptable drive pulley doesn't exist => no pulleys
            %=> ask user to reconsider selections

            set(handles.WarningTxt, 'String', 'Drive pulley could not be found for↙
this belt pitch, power requirements and gear ratio combination. Please reselect either↙
the belt pitch or gear ratio');

        end
    end


% --- Executes on selection change in DrivePTeeth_Pop.
function DrivePTeeth_Pop_Callback(hObject, eventdata, handles)
% hObject    handle to DrivePTeeth_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    %Run this stage only if previous stage complete
    if get(handles.PulleySize_UI,'UserData') >= 30

        %Determine drive pulley size (diameter) in mm, given number of teeth
        %and pitch of pulley
        %*****************************************************************

        %Get pulley size
        contents = get(handles.DrivePTeeth_Pop,'String');
        index = get(handles.DrivePTeeth_Pop,'Value');
        DriveTeeth = str2num(contents(index,:));

        %Get belt pitch index
        BeltPitch = get(handles.BeltPitch_Pop,'Value');


        %Perform diameter calculation ( = (pitch*teeth) / pi)
        if BeltPitch == 1
            %HTD5M was selected

            DpitchDrive = ((5*DriveTeeth) / pi());

        elseif BeltPitch == 2
            %HTD8M was selected

            DpitchDrive = ((8*DriveTeeth) / pi());

        elseif BeltPitch == 3
            %HTD14M was selected

            DpitchDrive = ((14*DriveTeeth) / pi());

        else
            %Write error to console
            display('Unknown belt pitch when calculatin drive pulley diameter');

        end

        %Write drive diameter to txt window
        set(handles.DrivePDia_Txt,'String',DpitchDrive);
        %Determine size of driven pulley given gear ratio
        %*****************************************************************

        index = get(handles.DrivePTeeth_Pop,'Value');
        DrivenTeethArray = get(handles.DrivenPTeeth_Txt,'UserData');
        DrivenTeeth = DrivenTeethArray(1,index);

        %Write drive diameter to txt window
        set(handles.DrivenPTeeth_Txt,'String',DrivenTeeth);



        %Determine diameter of drive pulley
        %*****************************************************************

        %Perform diameter calculation ( = (pitch*teeth) / pi)
        if BeltPitch == 1
            %HTD5M was selected

            DpitchDriven = ((5*DrivenTeeth) / pi());

        elseif BeltPitch == 2
            %HTD8M was selected

            DpitchDriven = ((8*DrivenTeeth) / pi());

        elseif BeltPitch == 3
            %HTD14M was selected

            DpitchDriven = ((14*DrivenTeeth) / pi());

        else
            %Write error to console
            display('Unknown belt pitch when calculatin drive pulley diameter');

        end

        %Write drive diameter to txt window
        set(handles.DrivenPDia_Txt,'String',DpitchDriven);


        %Write pulley diameters to geometry window
        %****************************************************************

        %DA pulley (Driver)
        set(handles.DA_Txt,'String',DpitchDrive);

        %DB pulley (Driven)
        set(handles.DB_Txt,'String',DpitchDriven);


        %Set Correct Data Stage
        %*****************************************************************

        %Set correct data stage
        set(handles.PulleySize_UI,'UserData',40);


    end

function DrivePDia_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to DrivePDia_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function DrivenPTeeth_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to DrivenPTeeth_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function DrivenPDia_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to DrivenPDia_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function DA_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to DA_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function DB_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to DB_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function DC_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to DC_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function X1_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to X1_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function Y1_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to Y1_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function X2_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to X2_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function Y2_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to Y2_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in GeoCal_PB.
function GeoCal_PB_Callback(hObject, eventdata, handles)
% hObject    handle to GeoCal_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    %Run this stage only if previous stage complete
    if get(handles.PulleySize_UI,'UserData') >= 40

        %Run desired belt length finder (non - standard belt)

        %Get pulley diameters from textboxes
        DA = str2num(get(handles.DrivePDia_Txt,'String'));  %calculated
        DB = str2num(get(handles.DrivenPDia_Txt,'String')); %calculated
        DC = str2num(get(handles.DC_Txt,'String'));         %user data

        %Get distances
        X1 = str2num(get(handles.X1_Txt,'String'));         %user data
        Y1 = str2num(get(handles.Y1_Txt,'String'));         %user data
        X2 = str2num(get(handles.X2_Txt,'String'));         %user data

        %Check if centre distance between pulleys is acceptable
        %****************************************************************

        [ Accept, MinC ] = AGVPulleyCentreDistanceChecker(X1, Y1, DA, DB);


        %Calculate desired belt length
        %****************************************************************
        if  Accept == 1
            %Center distance requirements met

            %Find ideal case, i.e. idler pulley just tangent to two pulley
            %system
            [ DesiredBL, Y2Initial ] = AGVDesiredBeltLengthCal(DA, DB, DC, X1, Y1, ↙
X2);




            %Determine the closest real life belt that matched the closen belt
            %****************************************************************

            %Get belt pitch index
            BeltPitch = get(handles.BeltPitch_Pop,'Value');

            if BeltPitch == 1
                %HTD5M was selected

                [ ChoosenBeltLength ] = AGVClosestActualBeltToDesiredBeltFinder(↙
DesiredBL, handles.HTD5M.BeltCenter );
                ChoosenBeltTeeth = ChoosenBeltLength/5;

            elseif BeltPitch == 2
                %HTD8M was selected
                [ ChoosenBeltLength ] = AGVClosestActualBeltToDesiredBeltFinder(↙
DesiredBL, handles.HTD8M.BeltCenter );
                ChoosenBeltTeeth = ChoosenBeltLength/8;

            elseif BeltPitch == 3
                %HTD14M was selected

                [ ChoosenBeltLength ] = AGVClosestActualBeltToDesiredBeltFinder(↙
DesiredBL, handles.HTD14M.BeltCenter );
                ChoosenBeltTeeth = ChoosenBeltLength/14;

            else
                %Write error to console
                display('Error when attempting to match real belt to desired belt');

            end




            %Write Belt lengths to GUI & Find Y2 value
            %*****************************************************************

            if ChoosenBeltLength ~= 0
                %Belt exist, ie doesn't equal zero

                %Set output value for belt teeth number
                set(handles.BeltTeeth_Txt,'String',ChoosenBeltTeeth);

                %Set output value for belt length
                set(handles.BeltLength_Txt,'String',ChoosenBeltLength);

                %Find Y2 value
                [Y2BestMatch, ValFound] = AGVFindIdlerPulleyYDistance↙
(ChoosenBeltLength, Y2Initial, DA, DB, DC, X1, Y1, X2);

                if ValFound == 1
                %Y2 could be found
                    %Set output value for belt length
                    set(handles.ActualY2_Txt,'String',Y2BestMatch);

                    %Set correct stage
                    set(handles.PulleySize_UI,'UserData',50);

                else
                %Y2 could not be found

                    %Write warning to window
                    WarningMessage = 'Iterative search for Y2 yielded no useful↙
results, consider changing geometry';

                    %Write error to gui error txt
                    set(handles.WarningTxt, 'String', WarningMessage);
                end

            elseif ChoosenBeltLength == 0
                %Belt doesn't exist

                %Set output value for belt teeth number to zero
                set(handles.BeltTeeth_Txt,'String','0');

                %Set output value for belt length to zero
                set(handles.BeltLength_Txt,'String','0');

                WarningMessage = 'No exists that is long enough, please change drive↙
geometry of belt pitch';

                %Write error to gui error txt
                set(handles.WarningTxt, 'String', WarningMessage);
            end



        elseif Accept == 0
            %Center distance requirements not met

            WarningPt1 = 'Minimum center distance requirements not met, Minimum center↙
distance is: ';
            WarningPt2 = num2str(MinC, 2);
            WarningPt3 = ' mm. Please edit drive geometry';

            WarningMessage = strcat(WarningPt1, WarningPt2, WarningPt3);

            %Write error to gui error txt
            set(handles.WarningTxt, 'String', WarningMessage);

        else
            %throw error to console
            display('Unknown state for calculating desired belt length');

        end

    else
        %Display error message
        set(handles.WarningTxt, 'String', 'Complete pulley sizing first');

    end

function BeltTeeth_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to BeltTeeth_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function BeltLength_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to BeltLength_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```matlab
function ActualY2_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to ActualY2_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)








%8888888888888888888888888888888888888888888888888888888888888888888888888
%Close GUI
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes when user attempts to close GUI8.
function GUI8_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to GUI8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if isequal(get(hObject, 'waitstatus'),'waiting')
    %The GUI is still in UIWAIT, use UIRESUME
    uiresume(hObject);
else
    %The GUI is nolonger waiting and can be closed
    delete(hObject);
end

% --- Executes during object deletion, before destroying properties.
function GUI8_DeleteFcn(hObject, eventdata, handles)
% hObject    handle to GUI8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles = rmfield(handles, 'HTD5M');
handles = rmfield(handles, 'HTD8M');
handles = rmfield(handles, 'HTD14M');
```

G.10 Belt 2B Optimisation

Belt 2B Optimisation:



Belt2BOptimisationGUI

Belt 2 Refinement (Part B)

Explination

This window is used to optimise the second belt section of belt "Belt 2"

Pulley Sizing

Desired Pitch: HTD8M ~

Desired Width: 30 mm

Gear Ratio: 1 : 1 ~

Drive Pulley Teeth: 36 ~

Drive Pulley Diameter: 91.6732 mm

Driven Pulley Teeth: 36 ~

Driven Pulley Diameter: 91.6732 mm

Geometry Specification

Drive Dia (mm): 91.6732

Driven Dia (mm): 91.6732

Idler Dia (mm): 25

X1 (mm): 147

Y1 (mm): 50

X2 (mm): 86

Belt Length (teeth): 75

Belt Length (mm): 600

Y2 (mm): -21.3271

```
function varargout = Belt2BOptimisationGUI(varargin)
% BELT2BOPTIMISATIONGUI M-file for Belt2BOptimisationGUI.fig
%      BELT2BOPTIMISATIONGUI, by itself, creates a new BELT2BOPTIMISATIONGUI or raises↙
the existing
%      singleton*.
%
%      H = BELT2BOPTIMISATIONGUI returns the handle to a new BELT2BOPTIMISATIONGUI or↙
the handle to
%      the existing singleton*.
%
%      BELT2BOPTIMISATIONGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in BELT2BOPTIMISATIONGUI.M with the given input↙
arguments.
%
%      BELT2BOPTIMISATIONGUI('Property','Value',...) creates a new↙
BELT2BOPTIMISATIONGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Belt2BOptimisationGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Belt2BOptimisationGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Belt2BOptimisationGUI

% Last Modified by GUIDE v2.5 26-Apr-2017 11:48:53

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Belt2BOptimisationGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @Belt2BOptimisationGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



%888888888888888888888888888888888888888888888888888888888888888888888888
%Initialise
%888888888888888888888888888888888888888888888888888888888888888888888888
```

```
% --- Executes just before Belt2BOptimisationGUI is made visible.
function Belt2BOptimisationGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Belt2BOptimisationGUI (see VARARGIN)

% Choose default command line output for Belt2BOptimisationGUI
handles.output = hObject;
handles.HTD5M = varargin{1};
handles.HTD8M = varargin{2};
handles.HTD14M = varargin{3};
handles.BeltPitch = varargin{4};
handles.RPM = varargin{5};
handles.Power = varargin{6};
handles.ConditionsVec = varargin{7};
handles.PowerConditions = varargin{8};
handles.DataExists = varargin{9};

% Update handles structure
guidata(hObject, handles);

% Populate list boxes
set(handles.BeltPitch_Pop,'String',varargin{4});

%set initial list box colours
set(handles.BeltWidth_Pop,'BackgroundColor',[0.94, 0.94, 0.94]);
set(handles.DrivePTeeth_Pop,'BackgroundColor',[0.94, 0.94, 0.94]);
set(handles.DrivePDia_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.One_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.GRatio_Pop,'BackgroundColor',[0.94, 0.94, 0.94]);
set(handles.DrivenPTeeth_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.DrivenPDia_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.DA_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.DB_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.BeltTeeth_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.BeltLength_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.ActualY2_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);

% Setup initial images
imshow('Belt2PartA.png','Parent', handles.BeltGeo_Axis);

%Initialise warning text
set(handles.WarningTxt, 'ForegroundColor', 'red');

%set data stage for pulley sizing
set(handles.PulleySize_UI,'UserData',0);

% UIWAIT makes Belt2BOptimisationGUI wait for user response (see UIRESUME)
uiwait(handles.GUI9);
```

```
%8888888888888888888888888888888888888888888888888888888888888888888888888
%Output Function
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Outputs from this function are returned to the command line.
function varargout = Belt2BOptimisationGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%(navigation data non-zero)
%************************************************************************

    %varargout{1} = Next GUI to be run
    %varargout{2} = (if = 1) => keep looping through results GUI else don't

    if get(handles.Done_PB,'UserData') ~= 0
        %Done PB was pressed to close GUI
        display('Done PB pressed');
        varargout{1} = 7;
        varargout{2} = 1;

        %Results Vector Data Creation
        %*****************************************************************

            %Get belt pitch index (index)
            index  = get(handles.BeltPitch_Pop,'Value');

            if index == 1
                %HTD5M belt selected
                BeltPitch = 5;

            elseif index == 2
                %HTD8M belt selected
                BeltPitch = 8;

            elseif index == 3
                %HTD14M belt selected
                BeltPitch = 14;

            else
                %throw error to console
                display('Output data creation error');
            end

            %Get belt width value (mm)
            contents = get(handles.BeltWidth_Pop,'String');
            index = get(handles.BeltWidth_Pop,'Value');
            DesiredBeltWidth = str2num(contents(index,:));

            %Get belt ratio (unitless)
            contents = get(handles.GRatio_Pop,'String');
            index = get(handles.GRatio_Pop,'Value');
            GearRatio = str2num(contents(index,:));

            %Get drive pulley teeth (#)
            contents = get(handles.DrivePTeeth_Pop,'String');
            index = get(handles.DrivePTeeth_Pop,'Value');
            DrivePulleyTeeth = str2num(contents(index,:));

            %Get drive pulley diameter (mm)
            DrivePulleyDia = str2num(get(handles.DrivePDia_Txt,'String'));

            %Get driven pulley teeth (#)
            DrivenPulleyTeeth = str2num(get(handles.DrivenPTeeth_Txt,'String'));

            %Get driven pulley diameter (mm)
            DrivenPulleyDia = str2num(get(handles.DrivenPDia_Txt,'String'));

            %Get X1 (mm)
            X1 = str2num(get(handles.X1_Txt,'String'));

            %Get Y1 (mm)
            Y1 = str2num(get(handles.Y1_Txt,'String'));

            %Get X2 (mm)
            X2 = str2num(get(handles.X2_Txt,'String'));

            %Get Y2 (mm)
            Y2 = str2num(get(handles.ActualY2_Txt,'String'));

            %Get belt length (teeth)
            BeltTeeth = str2num(get(handles.BeltTeeth_Txt,'String'));

            %Get belt length (mm)
            BeltLength = str2num(get(handles.BeltLength_Txt,'String'));

        %Create Output Vector Containing Results
        %*****************************************************************

            Belt2AResultsVector = [BeltPitch, DesiredBeltWidth, GearRatio, ↙
DrivePulleyTeeth, DrivePulleyDia, DrivenPulleyTeeth, DrivenPulleyDia, X1, Y1, X2, Y2, ↙
BeltTeeth, BeltLength];

            varargout{3} = Belt2AResultsVector;

    elseif get(handles.Skip_PB,'UserData') ~= 0
        %Skip PB was pressed to close GUI
        display('Done PB pressed');
        varargout{1} = 7;
        varargout{2} = 1;
        varargout{3} = 0;


    else
        %Unknown PB was pressed or user terminated
        display('Error: program closed unexpectedly')
    end
```

```
%Delete GUI
%*************************************************************************

    %The figure can now be deleted
    delete(handles.GUI9);




%888888888888888888888888888888888888888888888888888888888888888888888888
%Done Pushbutton
%888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes on button press in Done_PB.
function Done_PB_Callback(hObject, eventdata, handles)
% hObject    handle to Done_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if get(handles.PulleySize_UI,'UserData') >= 50
    %All stages complete
    set(handles.Done_PB,'UserData',1);

    close;

else
    %Not all stages complete

    %Write error to gui error txt
    WarningMessage = 'Not all data entered or calculated';
    set(handles.WarningTxt, 'String', WarningMessage);

end

% --- Executes on button press in Skip_PB.
function Skip_PB_Callback(hObject, eventdata, handles)
% hObject    handle to Skip_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.DataExists == 1
    %Data already exists

    set(handles.Skip_PB,'UserData',1);

    close;

else
    %Data doesn't exist => cannot skip

    %Write error to gui error txt
    WarningMessage = 'Data does not exist yet, cannot skip this step';
    set(handles.WarningTxt, 'String', WarningMessage);
```

```
end







%888888888888888888888888888888888888888888888888888888888888888888888888
%Create Textboxes and Objects
%888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function popupmenu3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function GRatio_Pop_CreateFcn(hObject, eventdata, handles)
% hObject    handle to GRatio_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function BeltWidth_Pop_CreateFcn(hObject, eventdata, handles)
% hObject    handle to BeltWidth_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function BeltPitch_Pop_CreateFcn(hObject, eventdata, handles)
% hObject    handle to BeltPitch_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function DrivePTeeth_Pop_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DrivePTeeth_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function DrivenPTeeth_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DrivenPTeeth_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function DrivePDia_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DrivePDia_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function DrivenPDia_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DrivenPDia_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function One_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to One_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function WarningTxt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to WarningTxt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% --- Executes during object creation, after setting all properties.
function DA_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DA_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function DB_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DB_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function DC_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DC_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function X1_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to X1_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function Y1_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Y1_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function X2_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to X2_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function Y2_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Y2_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function ActualY2_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ActualY2_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all properties.
function BeltTeeth_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to BeltTeeth_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function BeltLength_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to BeltLength_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
```

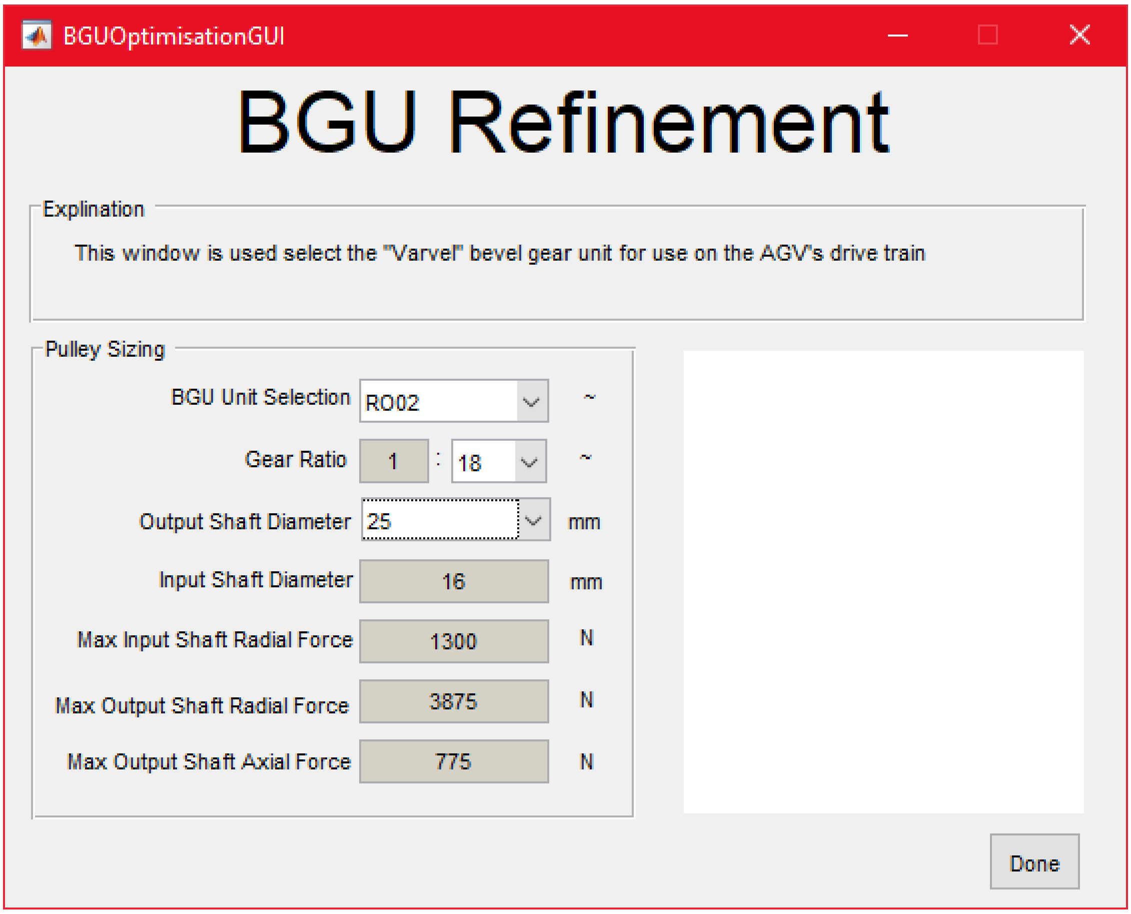
```
    set(hObject,'BackgroundColor','white');
end







%88888888888888888888888888888888888888888888888888888888888888888888888888
%UI Code Interaction Code(Pushbuttons, Radio Buttons, etc.)
%88888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes on selection change in BeltPitch_Pop.
function BeltPitch_Pop_Callback(hObject, eventdata, handles)
% hObject    handle to BeltPitch_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    %set correct colours (grey)
    %empty = {'Empty','Empty'};
    set(handles.GRatio_Pop,'BackgroundColor',[0.94, 0.94, 0.94]);
    set(handles.DrivePTeeth_Pop,'BackgroundColor',[0.94, 0.94, 0.94]);

    Checkvalue = 0;

    %read what value chosen from derired pitch pop up
    Checkvalue = get(handles.BeltPitch_Pop,'Value');
    %          = 1 = HTD5M
    %          = 2 = HTD8M
    %          = 3 = HTD14M

    %Run task to impliment avaliable widths
    if Checkvalue == 1
        %HTD5M was selected

        temp = handles.HTD5M.Pulley(1,:);

        %set correct colours (white)
        set(handles.BeltWidth_Pop,'BackgroundColor','white');

        %Write list to pop menu
        set(handles.BeltWidth_Pop,'String',temp);

    elseif Checkvalue == 2
        %HTD8M was selected

        temp = handles.HTD8M.Pulley(1,:);

        %set correct colours (white)
        set(handles.BeltWidth_Pop,'BackgroundColor','white');

        %Write list to pop menu
        set(handles.BeltWidth_Pop,'String',temp);

    elseif Checkvalue == 3
```

```
        %HTD14M was selected

        temp = handles.HTD14M.Pulley(1,:);

        %set correct colours (white)
        set(handles.BeltWidth_Pop,'BackgroundColor','white');

        %Write list to pop menu
        set(handles.BeltWidth_Pop,'String',temp);

    else
        %write error to console
        display('Selected belt does not mach any in data files');
    end

    %reset warning
    set(handles.WarningTxt, 'String', ' ');

    %Set correct data stage
    set(handles.PulleySize_UI,'UserData',10);


% --- Executes on selection change in BeltWidth_Pop.
function BeltWidth_Pop_Callback(hObject, eventdata, handles)
% hObject    handle to BeltWidth_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    %Run belt width selection only if previous stages complete
    if get(handles.PulleySize_UI,'UserData') >= 10
        %can proceede

        %set correct colours (grey)
        set(handles.DrivePTeeth_Pop,'BackgroundColor',[0.94, 0.94, 0.94]);

        %Find which pitch chosen
        Checkvalue = get(handles.BeltPitch_Pop,'Value');

        %Retrieve list of avaliable gear ratios

        if Checkvalue == 1
            %HTD5M was selected

            [RatioTable] = AGVBeltRatioSorter(handles.HTD5M.BeltCenter);

        elseif Checkvalue == 2
            %HTD8M was selected

            [RatioTable] = AGVBeltRatioSorter(handles.HTD8M.BeltCenter);

        elseif Checkvalue == 3
            %HTD14M was selected

            [RatioTable] = AGVBeltRatioSorter(handles.HTD14M.BeltCenter);
        else
            %Write error to console
            display('An error has occured in the ratio determination');

        end

        %reset warning
        set(handles.WarningTxt, 'String', ' ');

        %Set correct colours (white)
        set(handles.GRatio_Pop,'BackgroundColor',[1, 1, 1]);

        %Set correct data stage
        set(handles.PulleySize_UI,'UserData',20);

        %Write gear ratios to pop up menu
        set(handles.GRatio_Pop,'String',RatioTable);

    end

function One_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to One_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on selection change in GRatio_Pop.
function GRatio_Pop_Callback(hObject, eventdata, handles)
% hObject    handle to GRatio_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


    %Run belt width selection only if previous stages complete
    if get(handles.PulleySize_UI,'UserData') >= 20
        %can proceede

        %Get belt pitch index
        BeltPitch = get(handles.BeltPitch_Pop,'Value');

        %Get belt width value (mm)
        contents = get(handles.BeltWidth_Pop,'String');
        index = get(handles.BeltWidth_Pop,'Value');
        BeltWidth = str2num(contents(index,:));

        %Get belt ratio (unitless)
        contents = get(handles.GRatio_Pop,'String');
        index = get(handles.GRatio_Pop,'Value');
        GearTrainI = str2num(contents(index,:));

        %Get RPM (rad/sec)
        RPM = handles.RPM;

        %Get power (Watts)
        Power = handles.Power;
        %Run Drive Pulley Finder
        if BeltPitch == 1
            %HTD5M was selected

            [ PowerDrivesAccpt ] = AGVDrivePulleyPowerChecker(RPM, Power, BeltWidth,↙
(handles.HTD5M.Pulley), (handles.HTD5M.Power), (handles.HTD5M.PowerMult) ,GearTrainI,↙
(handles.ConditionsVec), (handles.PowerConditions));
            [ AcceptableDrives ] = AGVDrivePulleyRatioChecker(GearTrainI, (handles.↙
HTD5M.BeltCenter), PowerDrivesAccpt );


        elseif BeltPitch == 2
            %HTD8M was selected

            [ PowerDrivesAccpt ] = AGVDrivePulleyPowerChecker(RPM, Power, BeltWidth,↙
(handles.HTD8M.Pulley), (handles.HTD8M.Power), (handles.HTD8M.PowerMult), GearTrainI,↙
(handles.ConditionsVec), (handles.PowerConditions));
            [ AcceptableDrives ] = AGVDrivePulleyRatioChecker(GearTrainI, (handles.↙
HTD8M.BeltCenter), PowerDrivesAccpt );

        elseif BeltPitch == 3
            %HTD14M was selected

            [ PowerDrivesAccpt ] = AGVDrivePulleyPowerChecker(RPM, Power, BeltWidth,↙
(handles.HTD14M.Pulley), (handles.HTD14M.Power), (handles.HTD14M.PowerMult),↙
GearTrainI, (handles.ConditionsVec), (handles.PowerConditions));
            [ AcceptableDrives ] = AGVDrivePulleyRatioChecker(GearTrainI, (handles.↙
HTD14M.BeltCenter), PowerDrivesAccpt );

        end


        %Check if any drive pulleys are avaliable for chosen gear ratio

        if AcceptableDrives(1,1) ~= 0
            %The first acceptable drive pulley isn't zero => it exists

            %reset warning
            set(handles.WarningTxt, 'String', ' ');

            %Set correct colours (white)
            set(handles.DrivePTeeth_Pop,'BackgroundColor',[1, 1, 1]);

            %Write avaliable drivers to pop up menu
            set(handles.DrivePTeeth_Pop,'String',AcceptableDrives(1,:));

            %Write matching driven vector to UserData of "DrivenPTeeth_Txt"
            set(handles.DrivenPTeeth_Txt,'UserData',AcceptableDrives(2,:));

            %Set correct data stage
            set(handles.PulleySize_UI,'UserData',30);

        elseif AcceptableDrives(1,1) == 0
            %The first acceptable drive pulley doesn't exist => no pulleys
            %=> ask user to reconsider selections
            set(handles.WarningTxt, 'String', 'Drive pulley could not be found for↙
this belt pitch, power requirements and gear ratio combination. Please reselect either↙
the belt pitch or gear ratio');

        end
    end


% --- Executes on selection change in DrivePTeeth_Pop.
function DrivePTeeth_Pop_Callback(hObject, eventdata, handles)
% hObject    handle to DrivePTeeth_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    %Run this stage only if previous stage complete
    if get(handles.PulleySize_UI,'UserData') >= 30

        %Determine drive pulley size (diameter) in mm, given number of teeth
        %and pitch of pulley
        %*****************************************************************

        %Get pulley size
        contents = get(handles.DrivePTeeth_Pop,'String');
        index = get(handles.DrivePTeeth_Pop,'Value');
        DriveTeeth = str2num(contents(index,:));

        %Get belt pitch index
        BeltPitch = get(handles.BeltPitch_Pop,'Value');


        %Perform diameter calculation ( = (pitch*teeth) / pi)
        if BeltPitch == 1
            %HTD5M was selected

            DpitchDrive = ((5*DriveTeeth) / pi());

        elseif BeltPitch == 2
            %HTD8M was selected

            DpitchDrive = ((8*DriveTeeth) / pi());

        elseif BeltPitch == 3
            %HTD14M was selected

            DpitchDrive = ((14*DriveTeeth) / pi());

        else
            %Write error to console
            display('Unknown belt pitch when calculatin drive pulley diameter');

        end

        %Write drive diameter to txt window
        set(handles.DrivePDia_Txt,'String',DpitchDrive);
        %Determine size of driven pulley given gear ratio
        %*****************************************************************
        
        index = get(handles.DrivePTeeth_Pop,'Value');
        DrivenTeethArray = get(handles.DrivenPTeeth_Txt,'UserData');
        DrivenTeeth = DrivenTeethArray(1,index);
        
        %Write drive diameter to txt window
        set(handles.DrivenPTeeth_Txt,'String',DrivenTeeth);
        
        
        
        %Determine diameter of drive pulley
        %*****************************************************************
        
        %Perform diameter calculation ( = (pitch*teeth) / pi)
        if BeltPitch == 1
            %HTD5M was selected
            
            DpitchDriven = ((5*DrivenTeeth) / pi());
            
        elseif BeltPitch == 2
            %HTD8M was selected
            
            DpitchDriven = ((8*DrivenTeeth) / pi());
            
        elseif BeltPitch == 3
            %HTD14M was selected
            
            DpitchDriven = ((14*DrivenTeeth) / pi());
            
        else
            %Write error to console
            display('Unknown belt pitch when calculatin drive pulley diameter');
            
        end
        
        %Write drive diameter to txt window
        set(handles.DrivenPDia_Txt,'String',DpitchDriven);
        
        
        %Write pulley diameters to geometry window
        %****************************************************************
        
        %DA pulley (Driver)
        set(handles.DA_Txt,'String',DpitchDrive);
        
        %DB pulley (Driven)
        set(handles.DB_Txt,'String',DpitchDriven);
        
        
        %Set Correct Data Stage
        %*****************************************************************

        %Set correct data stage
        set(handles.PulleySize_UI,'UserData',40);


    end

function DrivePDia_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to DrivePDia_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function DrivenPTeeth_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to DrivenPTeeth_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function DrivenPDia_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to DrivenPDia_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function DA_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to DA_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function DB_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to DB_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function DC_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to DC_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function X1_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to X1_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function Y1_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to Y1_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function X2_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to X2_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function Y2_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to Y2_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in GeoCal_PB.
function GeoCal_PB_Callback(hObject, eventdata, handles)
% hObject    handle to GeoCal_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    %Run this stage only if previous stage complete
    if get(handles.PulleySize_UI,'UserData') >= 40

        %Run desired belt length finder (non - standard belt)

        %Get pulley diameters from textboxes
        DB = str2num(get(handles.DrivePDia_Txt,'String'));  %calculated
        DA = str2num(get(handles.DrivenPDia_Txt,'String')); %calculated
        DC = str2num(get(handles.DC_Txt,'String'));         %user data

        %Get distances
        X1 = str2num(get(handles.X1_Txt,'String'));         %user data
        Y1 = str2num(get(handles.Y1_Txt,'String'));         %user data
        X2_dash = str2num(get(handles.X2_Txt,'String'));    %user data
        X2 = X1 - X2_dash;  %instead of measuring from DB to DC measure
                            %from DA to DC

        %Check if centre distance between pulleys is acceptable
        %****************************************************************

        [ Accept, MinC ] = AGVPulleyCentreDistanceChecker(X1, Y1, DA, DB);


        %Calculate desired belt length
        %****************************************************************
        if  Accept == 1
            %Center distance requirements met

            %Find ideal case, i.e. idler pulley just tangent to two pulley
            %system
            [ DesiredBL, Y2Initial ] = AGVDesiredBeltLengthCal(DA, DB, DC, X1, Y1, ↙
X2);




            %Determine the closest real life belt that matched the closen belt
            %****************************************************************

            %Get belt pitch index
            BeltPitch = get(handles.BeltPitch_Pop,'Value');

            if BeltPitch == 1
                %HTD5M was selected

                [ ChoosenBeltLength ] = AGVClosestActualBeltToDesiredBeltFinder(↙
DesiredBL, handles.HTD5M.BeltCenter );
                ChoosenBeltTeeth = ChoosenBeltLength/5;

            elseif BeltPitch == 2
                %HTD8M was selected

                [ ChoosenBeltLength ] = AGVClosestActualBeltToDesiredBeltFinder(↙
DesiredBL, handles.HTD8M.BeltCenter );
                ChoosenBeltTeeth = ChoosenBeltLength/8;

            elseif BeltPitch == 3
                %HTD14M was selected

                [ ChoosenBeltLength ] = AGVClosestActualBeltToDesiredBeltFinder(↙
DesiredBL, handles.HTD14M.BeltCenter );
                ChoosenBeltTeeth = ChoosenBeltLength/14;

            else
                %Write error to console
                display('Error when attempting to match real belt to desired belt');

            end




            %Write Belt lengths to GUI & Find Y2 value
            %*****************************************************************

            if ChoosenBeltLength ~= 0
                %Belt exist, ie doesn't equal zero

                %Set output value for belt teeth number
                set(handles.BeltTeeth_Txt,'String',ChoosenBeltTeeth);

                %Set output value for belt length
                set(handles.BeltLength_Txt,'String',ChoosenBeltLength);

                %Find Y2 value
                [Y2BestMatch, ValFound] = AGVFindIdlerPulleyYDistance↙
(ChoosenBeltLength, Y2Initial, DA, DB, DC, X1, Y1, X2);

                if ValFound == 1
                    %Y2 could be found
                    %convert form measuring from DA to DB
                    Y2BestMatch_dash = Y1 - Y2BestMatch;

                    %Set output value for belt length
                    set(handles.ActualY2_Txt,'String',Y2BestMatch_dash);

                    %Set correct stage
                    set(handles.PulleySize_UI,'UserData',50);

                else
                %Y2 could not be found
```

```
                    %Write warning to window
                    WarningMessage = 'Iterative search for Y2 yielded no useful↙
results, consider changing geometry';

                    %Write error to gui error txt
                    set(handles.WarningTxt, 'String', WarningMessage);

                end

            elseif ChoosenBeltLength == 0
                %Belt doesn't exist

                %Set output value for belt teeth number to zero
                set(handles.BeltTeeth_Txt,'String','0');

                %Set output value for belt length to zero
                set(handles.BeltLength_Txt,'String','0');

                WarningMessage = 'No exists that is long enough, please change drive↙
geometry of belt pitch';

                %Write error to gui error txt
                set(handles.WarningTxt, 'String', WarningMessage);
            end



        elseif Accept == 0
            %Center distance requirements not met

            WarningPt1 = 'Minimum center distance requirements not met, Minimum center↙
distance is: ';
            WarningPt2 = num2str(MinC, 2);
            WarningPt3 = ' mm. Please edit drive geometry';

            WarningMessage = strcat(WarningPt1, WarningPt2, WarningPt3);

            %Write error to gui error txt
            set(handles.WarningTxt, 'String', WarningMessage);

        else
            %throw error to console
            display('Unknown state for calculating desired belt length');

        end

    else
        %Display error message
        set(handles.WarningTxt, 'String', 'Complete pulley sizing first');
    end

function BeltTeeth_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to BeltTeeth_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function BeltLength_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to BeltLength_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function ActualY2_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to ActualY2_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)








%888888888888888888888888888888888888888888888888888888888888888888888888
%Close GUI
%888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes when user attempts to close GUI9.
function GUI9_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to GUI9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if isequal(get(hObject, 'waitstatus'),'waiting')
    %The GUI is still in UIWAIT, use UIRESUME
    uiresume(hObject);
else
    %The GUI is nolonger waiting and can be closed
    delete(hObject);
end

% --- Executes during object deletion, before destroying properties.
function GUI9_DeleteFcn(hObject, eventdata, handles)
% hObject    handle to GUI9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles = rmfield(handles, 'HTD5M');
handles = rmfield(handles, 'HTD8M');
handles = rmfield(handles, 'HTD14M');
```

G.11 Bevel Gear Unit Optimisation

Bevel Gear Unit Optimisation:



BGU Refinement

Explination

This window is used select the "Varvel" bevel gear unit for use on the AGV's drive train

Pulley Sizing

BGU Unit Selection	RO02	~
Gear Ratio	1 : 18	~
Output Shaft Diameter	25	mm
Input Shaft Diameter	16	mm
Max Input Shaft Radial Force	1300	N
Max Output Shaft Radial Force	3875	N
Max Output Shaft Axial Force	775	N

```
function varargout = BGUOptimisationGUI(varargin)
% BGUOPTIMISATIONGUI M-file for BGUOptimisationGUI.fig
%      BGUOPTIMISATIONGUI, by itself, creates a new BGUOPTIMISATIONGUI or raises the↙
existing
%      singleton*.
%
%      H = BGUOPTIMISATIONGUI returns the handle to a new BGUOPTIMISATIONGUI or the↙
handle to
%      the existing singleton*.
%
%      BGUOPTIMISATIONGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in BGUOPTIMISATIONGUI.M with the given input arguments.
%
%      BGUOPTIMISATIONGUI('Property','Value',...) creates a new BGUOPTIMISATIONGUI or↙
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before BGUOptimisationGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to BGUOptimisationGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one_txt
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help BGUOptimisationGUI

% Last Modified by GUIDE v2.5 25-Apr-2017 09:19:15

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @BGUOptimisationGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @BGUOptimisationGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT




%8888888888888888888888888888888888888888888888888888888888888888888888888
%Initialise
%8888888888888888888888888888888888888888888888888888888888888888888888888
```

```
% --- Executes just before BGUOptimisationGUI is made visible.
function BGUOptimisationGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to BGUOptimisationGUI (see VARARGIN)

% Choose default command line output for BGUOptimisationGUI
handles.output = hObject;
handles.ROBGU = varargin{1};
handles.AvaliableBGU = varargin{2};
handles.Torque = varargin{3};
handles.Power = varargin{4};

% Update handles structure
guidata(hObject, handles);

% Populate list boxes
set(handles.Unit_Pop,'String',varargin{2});

% Set initial list box colours
set(handles.Ratio_Pop,'BackgroundColor',[0.94, 0.94, 0.94]);
set(handles.One_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.OutShaft_Pop,'BackgroundColor',[0.94, 0.94, 0.94]);
set(handles.InShaft_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.RFInShaft_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.RFOutShaft_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.AFShaft_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);

% Setup initial images
imshow('VarvelBGU.png','Parent', handles.BGU_Pic);

% UIWAIT makes BGUOptimisationGUI wait for user response (see UIRESUME)
uiwait(handles.GUI10);





%8888888888888888888888888888888888888888888888888888888888888888888888888
%Output Function
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Outputs from this function are returned to the command line.
function varargout = BGUOptimisationGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%(navigation data non-zero)
%************************************************************************
```

```
    %varargout{1} = Next GUI to be run
    %varargout{2} = (if = 1) => keep looping through results GUI else don't
    %varargout{3} = results vector

    if get(handles.Done_PB,'UserData') ~= 0
        %Next PB was pressed to close GUI
        display('Done button pressed');
        varargout{1} = 7;
        varargout{2} = 1;

        %Get chosen BGU index
        BGUUnitIndex = get(handles.Unit_Pop,'Value');

        %Get BGU gear ratio (shown value)
        contents = get(handles.Ratio_Pop,'String');
        index = get(handles.Ratio_Pop,'Value');
        GearRatioGiven = str2num(contents(index,:));

        %Get BGU gear ratio (actual value)
            %Get Contents of appropriate ROBGU
            if BGUUnitIndex == 1
                %RO02 selected

                contents = handles.ROBGU.RO02;

            elseif BGUUnitIndex == 2
                %RO12 selected

                contents = handles.ROBGU.RO12;

            elseif BGUUnitIndex == 3
                %RO22 selected

                contents = handles.ROBGU.RO22;

            elseif BGUUnitIndex == 4
                %RO32 selected

                contents = handles.ROBGU.RO32;

            else
                %Unknown state

                %Write error to console
                display('Unknown unit size when selecting shaft data');

            end

            GearRatioActual = contents(index,2);

        %Get output shaft diameter
        contents = get(handles.OutShaft_Pop,'String');
        index = get(handles.OutShaft_Pop,'Value');
        OuterShaftDia = str2num(contents(index,:));
        %Get input shaft diameter
        InputShaftDia = str2num(get(handles.InShaft_Txt,'String'));

        %Get max allowable radial force on input shaft
        InputShaftRadForce = str2num(get(handles.RFInShaft_Txt,'String'));

        %Get max allowable radial force on output shaft
        OutputShaftRadForce = str2num(get(handles.RFOutShaft_Txt,'String'));

        %Get max allowable thrust force on output shaft
        OutputShaftThrstForce = str2num(get(handles.AFShaft_Txt,'String'));

        %Return results vector
        varargout{3} = [BGUUnitIndex, GearRatioGiven, GearRatioActual, OuterShaftDia,↙
InputShaftDia, InputShaftRadForce, OutputShaftRadForce, OutputShaftThrstForce];

    end

%Delete GUI
%*************************************************************************

    %The figure can now be deleted
    delete(handles.GUI10);




%88888888888888888888888888888888888888888888888888888888888888888888888
%Prev, Next, Home Pushbuttons
%88888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes on button press in Done_PB.
function Done_PB_Callback(hObject, eventdata, handles)
% hObject    handle to Done_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if get(handles.GUI10,'UserData') >= 20
    %All data processed

    set(handles.Done_PB,'UserData',1);

    close;

else
    %Not all data recieved

    %Clear error message
    set(handles.Warning_Txt,'String','Please ensure all user data is selected');

end


%888888888888888888888888888888888888888888888888888888888888888888888888
```

```
%Create Textboxes and Objects
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes during object creation, after setting all properties.
function Ratio_Pop_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Ratio_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function One_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to One_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function AFShaft_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to AFShaft_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function RFInShaft_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to RFInShaft_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes during object creation, after setting all properties.
function RFOutShaft_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to RFOutShaft_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function Unit_Pop_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Unit_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function InShaft_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to InShaft_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function OutShaft_Pop_CreateFcn(hObject, eventdata, handles)
% hObject    handle to OutShaft_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
%8888888888888888888888888888888888888888888888888888888888888888888888888
%UI Code Interaction Code(Pushbuttons, Radio Buttons, etc.)
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes on selection change in Unit_Pop.
function Unit_Pop_Callback(hObject, eventdata, handles)
% hObject    handle to Unit_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    %Clear error message
    set(handles.Warning_Txt,'String',' ');

    %Set initial colours (grey)
    set(handles.Ratio_Pop,'BackgroundColor',[0.94, 0.94, 0.94]);
    set(handles.OutShaft_Pop,'BackgroundColor',[0.94, 0.94, 0.94]);

    %Reset chosen gera ratio and output = false
    set(handles.Ratio_Pop,'UserData', 0);
    set(handles.OutShaft_Pop,'UserData',0);

    BGUSize = get(handles.Unit_Pop,'Value');

    %Run BGU ratio solver
    [ RatiosList, InputListDia, OutputListDia] = AGVBGUGearRatioFinder(BGUSize,↙
handles.ROBGU, handles.Power, handles.Torque);

    if isempty(RatiosList) ~= 1
        %There exist at least one ratio

        %Add Data to ratio pop menu
        %*****************************************************************
            %Change color of pop menu
            set(handles.Ratio_Pop,'BackgroundColor','white');

            %Add data to pop up menu
            set(handles.Ratio_Pop,'String',RatiosList(1,:));

        %Add data to output shaft pop menu
        %*****************************************************************

            %Change color of pop menu
            set(handles.OutShaft_Pop,'BackgroundColor','white');

            %Add data to pop menu
            set(handles.OutShaft_Pop,'String',OutputListDia);

        %Add data to input shaft txt box
        %*****************************************************************

            %Add data to txt box
            set(handles.InShaft_Txt,'String',InputListDia(1,1));

        %Set correct data stage
        %*****************************************************************

            %Incriment data stage
            set(handles.GUI10,'UserData',10);

    else
        %There is no acceptable ratios

        %Write error message to window
        set(handles.Warning_Txt,'String','No acceptable ratios found, try increasing↙
BGU unit size');

    end

% --- Executes on selection change in Ratio_Pop.
function Ratio_Pop_Callback(hObject, eventdata, handles)
% hObject    handle to Ratio_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if get(handles.GUI10,'UserData') >= 10
    %Chosen BGU size

    %Clear error message
    set(handles.Warning_Txt,'String',' ');

    %Set chosen gear ratio = true
    set(handles.Ratio_Pop,'UserData', 1);

    if (get(handles.Ratio_Pop,'UserData') == 1) && (get(handles.↙
OutShaft_Pop,'UserData') == 1)
        %Both gear ratio and output shaft diameter selected

        %Incriment data stage
        set(handles.GUI10,'UserData',20);

        %Get BGU unit index
        indexUnit = get(handles.Unit_Pop,'Value');

        %Get Contents of appropriate ROBGU
        if indexUnit == 1
            %RO02 selected

            contents = handles.ROBGU.RO02;

        elseif indexUnit == 2
            %RO12 selected

            contents = handles.ROBGU.RO12;

        elseif indexUnit == 3
            %RO22 selected

            contents = handles.ROBGU.RO22;
        elseif indexUnit == 4
            %RO32 selected

            contents = handles.ROBGU.RO32;

        else
            %Unknown state

            %Write error to console
            display('Unknown unit size when selecting shaft data');

        end

        %Get gear ratio index
        indexGear = get(handles.Ratio_Pop,'Value');

        %Get & Set maximum allowable radial force on input shaft
        temp = contents(indexGear,6);
        set(handles.RFInShaft_Txt,'String',temp);

        %Get & Set maximum allowable radial force on output shaft
        Kl = 1;     %Placement factor (middle of output shaft)
        Kt = 1.25;  %Timing belt used factor
        temp = contents(indexGear,7)*Kl*Kt;
        set(handles.RFOutShaft_Txt,'String',temp);

        %Get & Set maximum allowable axial force on output shaft
        temp = temp*0.2;
        set(handles.AFShaft_Txt,'String',temp);

    end

else

    %Write error to window
    set(handles.Warning_Txt,'String','Please select a BGU size first');

end

function One_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to One_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on selection change in OutShaft_Pop.
function OutShaft_Pop_Callback(hObject, eventdata, handles)
% hObject    handle to OutShaft_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if get(handles.GUI10,'UserData') >= 10
    %Chosen BGU size

    %Clear error message
    set(handles.Warning_Txt,'String',' ');
    %Set chosen gear ratio = true
    set(handles.OutShaft_Pop,'UserData', 1);

    if (get(handles.Ratio_Pop,'UserData') == 1) && (get(handles.↙
OutShaft_Pop,'UserData') == 1)
        %Both gear ratio and output shaft diameter selected

        %Incriment data stage
        set(handles.GUI10,'UserData',20);

        %Get BGU unit index
        indexUnit = get(handles.Unit_Pop,'Value');

        %Get Contents of appropriate ROBGU
        if indexUnit == 1
            %RO02 selected

            contents = handles.ROBGU.RO02;

        elseif indexUnit == 2
            %RO12 selected

            contents = handles.ROBGU.RO12;

        elseif indexUnit == 3
            %RO22 selected

            contents = handles.ROBGU.RO22;

        elseif indexUnit == 4
            %RO32 selected

            contents = handles.ROBGU.RO32;

        else
            %Unknown state

            %Write error to console
            display('Unknown unit size when selecting shaft data');

        end

        %Get gear ratio index
        indexGear = get(handles.Ratio_Pop,'Value');

        %Get & Set maximum allowable radial force on input shaft
        temp = contents(indexGear,6);
        set(handles.RFInShaft_Txt,'String',temp);

        %Get & Set maximum allowable radial force on output shaft
        Kl = 1;     %Placement factor (middle of output shaft)
        Kt = 1.25;  %Timing belt used factor
        temp = contents(indexGear,7)*Kl*Kt;
        set(handles.RFOutShaft_Txt,'String',temp);
        %Get & Set maximum allowable axial force on output shaft
        temp = temp*0.2;
        set(handles.AFShaft_Txt,'String',temp);

    end

else

    %Write error to window
    set(handles.Warning_Txt,'String','Please select a BGU size first');

end

function AFShaft_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to AFShaft_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function RFInShaft_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to RFInShaft_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function RFOutShaft_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to RFOutShaft_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function InShaft_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to InShaft_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)







%8888888888888888888888888888888888888888888888888888888888888888888888888
%Close GUI
%8888888888888888888888888888888888888888888888888888888888888888888888888


% --- Executes when user attempts to close GUI10.
function GUI10_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to GUI10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if isequal(get(hObject, 'waitstatus'),'waiting')
    %The GUI is still in UIWAIT, use UIRESUME
    uiresume(hObject);
else
```

```
    %The GUI is nolonger waiting and can be closed
    delete(hObject);
end
```

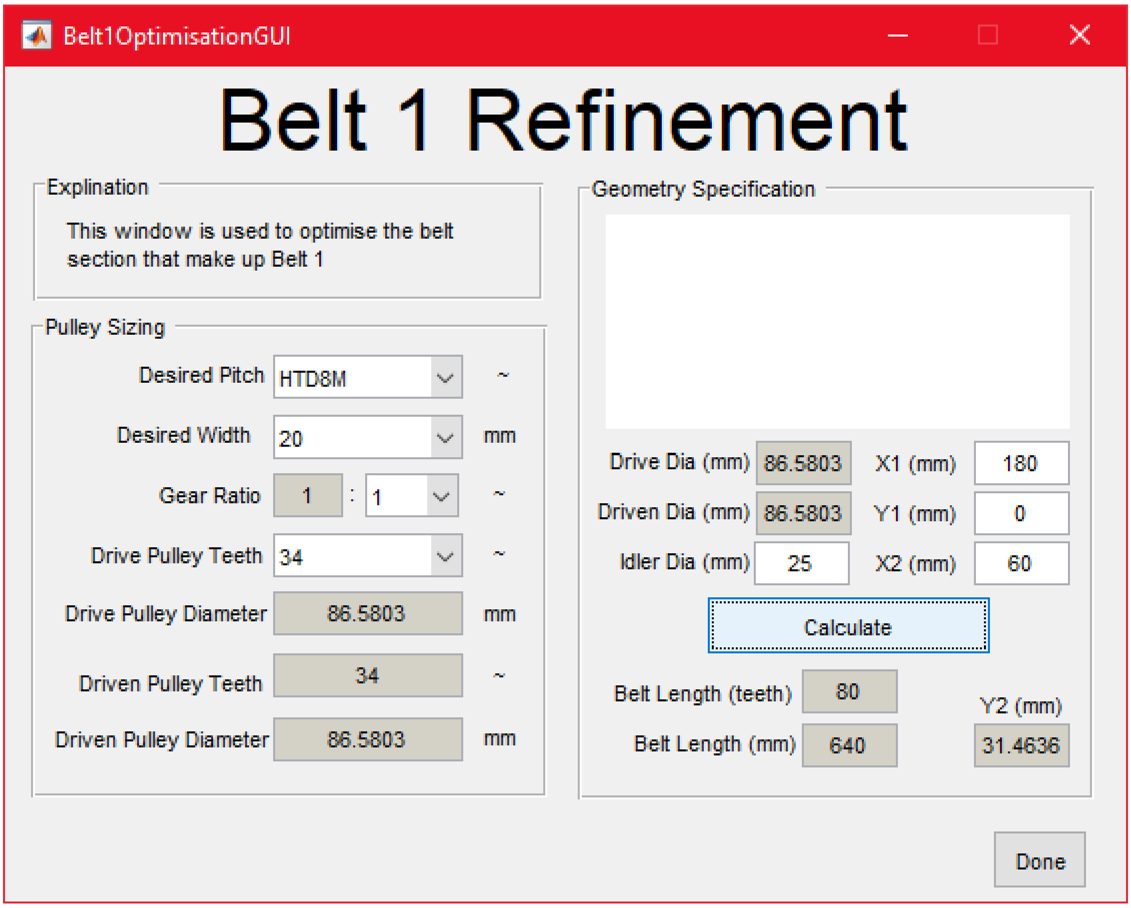
```
function [ RatiosList, InputListDia, OutputListDia ] = AGVBGUGearRatioFinder(BGUSize,↙
BGUStruct ,DesiredPower, DesiredTorque)
%AGVBGUGearRatioFinder
%888888888888888888888888888888888888888888888888888888888888888888888888

%   This calculator determines the avaliable bevel gear units ratios given
%   the power and speed limitations

%   Inputs
%   **********************************************************************

%   BGUSize       = size of of the BGU (RO02, RO12, RO22 or RO32) (~)
%   BGUStruct     = a structure containing all types of BGU       (~)
%   DesiredPower  = the desired power transfer through the BGU    (W)
%   DesiredTorque = the desired torque out of the BGU             (Nm)

%   Outputs
%   **********************************************************************

%   RatiosList    = a list of acceptable ratios                     (~)
%   InputListDia  = returns a list of avaliable input shaft sizes   (mm)
%   OutputListDia = returns a list of avaliable outputs shaft sizes (mm)




%8888888888888888888888888888888888888888888888888888888888888888888888888
%Isolate Desired Unit Size From Main Table
%8888888888888888888888888888888888888888888888888888888888888888888888888

    if BGUSize == 1
        %RO02 chosen for unit size

        TempTable = BGUStruct.RO02;

    elseif BGUSize == 2
        %RO12 chosen for unit size

        TempTable = BGUStruct.RO12;

    elseif BGUSize == 3
        %RO22 chosen for unit size

        TempTable = BGUStruct.RO22;

    elseif BGUSize == 4
        %RO32 chosen for unit size

        TempTable = BGUStruct.RO32;

    else
        %Unknown unit size

        %Write error to console
        display('Unknown BGU unit size selected whem attempting to find gear ratios');
    end

    %Find size of table 'TempTable'
    [n m] = size(TempTable);




%8888888888888888888888888888888888888888888888888888888888888888888888888
%Isolate Desired Unit Size
%8888888888888888888888888888888888888888888888888888888888888888888888888

    %Create empty vector to store data
    TempInputDia = zeros(1,n);      %Empty input shaft diameter table


    %Write data from struct 'BGUStruct' to 'TempInputDia'
    %*********************************************************************
        InputCnt = 0;
        Stop = 0;

        while (InputCnt <= n) && (Stop == 0)
            %for each row in column m-1

            %incriment loop counter
            InputCnt = InputCnt + 1;

            %find current value on table
            Current = TempTable(InputCnt,m-1);

            if isnan(Current) ~= 1
                %value exists

                TempInputDia(1,InputCnt) = Current;

            elseif isnan(Current) == 1
                %value doesn't exist => no more values exit

                %terminate loop
                Stop = 1;

            end
        end

    %Remove zero vectors from 'TempInputDia' to create 'InputListDia'
    %*********************************************************************

        InputListDia = zeros(1,InputCnt-1);

        for i = 1:InputCnt-1

            InputListDia(1,i) = TempInputDia(1,i);

        end
```

```
%8888888888888888888888888888888888888888888888888888888888888888888888888
%Create Shaft Output Diameter Vectors
%8888888888888888888888888888888888888888888888888888888888888888888888888

    %Create empty vector to store data
    TempOutDia = zeros(1,n);        %Empty output shaft diamater table

    %Write data from "BGUStruct' to 'TempOutDia'
    %*********************************************************************
        OutputCnt = 0;
        Stop = 0;

        while (OutputCnt <= n) && (Stop == 0)
            %for each row in column m-1

            %incriment loop counter
            OutputCnt = OutputCnt + 1;

            %find current value on table
            Current = TempTable(OutputCnt,m);

            if isnan(Current) ~= 1
                %value exists

                TempOutDia(1,OutputCnt) = Current;

            elseif isnan(Current) == 1
                %value doesn't exist => no more values exit

                %terminate loop
                Stop = 1;
            end
        end

    %Remove zero vectors from 'TempOutDia' to create'OutputListDia'
    %*********************************************************************
        OutputListDia = zeros(1,OutputCnt-1);

        for i = 1:OutputCnt-1

            OutputListDia(1,i) = TempOutDia(1,i);

        end



%888888888888888888888888888888888888888888888888888888888888888888888888
```

```
%Create acceptable ratio list
%888888888888888888888888888888888888888888888888888888888888888888888888

    %Find all ratios that meet power and torque requirements
    %********************************************************************
        TempRatioList = zeros(2,n); %row 1: Given ratio
                                    %row 2: true ratio

        %Accepted ratio counter
        AcptRatioCnt = 0;

        for i = 1:n
            %for each row in table temp

            CurrentPower = TempTable(i,5)*1000; %convert kW to W
            CurrentTorque = TempTable(i,4);

            if (CurrentTorque >= DesiredTorque) && (CurrentPower >= DesiredPower)
                %Torque, Power meet min requirements

                %Incriment acceptable ratio counter
                AcptRatioCnt = AcptRatioCnt + 1;

                %Write given ratio to 'TempRatioList'
                TempRatioList(1,AcptRatioCnt) = TempTable(i,1);

                %Write true ratio to 'TempRatioList'
                TempRatioList(2,AcptRatioCnt) = TempTable(i,2);

            end

        end

    %Remove zeros from returned vector 'RatiosList'
    %*********************************************************************
        RatiosList = zeros(2,AcptRatioCnt);


        for i = 1:AcptRatioCnt

            RatiosList(1,i) = TempRatioList(1,i);
            RatiosList(2,i) = TempRatioList(2,i);

        end

end
```

G.12 Belt 1 Optimisation

Belt 1 Optimisation:



Belt1OptimisationGUI

Belt 1 Refinement

Explination

This window is used to optimise the belt section that make up Belt 1

Pulley Sizing

Desired Pitch	HTD8M	~
Desired Width	20	mm
Gear Ratio	1 : 1	~
Drive Pulley Teeth	34	~
Drive Pulley Diameter	86.5803	mm
Driven Pulley Teeth	34	~
Driven Pulley Diameter	86.5803	mm

Geometry Specification

Drive Dia (mm)	86.5803	X1 (mm)	180
Driven Dia (mm)	86.5803	Y1 (mm)	0
Idler Dia (mm)	25	X2 (mm)	60

Belt Length (teeth)	80	Y2 (mm)
Belt Length (mm)	640	31.4636

```
function varargout = Belt1OptimisationGUI(varargin)
% BELT1OPTIMISATIONGUI M-file for Belt1OptimisationGUI.fig
%      BELT1OPTIMISATIONGUI, by itself, creates a new BELT1OPTIMISATIONGUI or raises↙
the existing
%      singleton*.
%
%      H = BELT1OPTIMISATIONGUI returns the handle to a new BELT1OPTIMISATIONGUI or↙
the handle to
%      the existing singleton*.
%
%      BELT1OPTIMISATIONGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in BELT1OPTIMISATIONGUI.M with the given input↙
arguments.
%
%      BELT1OPTIMISATIONGUI('Property','Value',...) creates a new BELT1OPTIMISATIONGUI↙
or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Belt1OptimisationGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Belt1OptimisationGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Belt1OptimisationGUI

% Last Modified by GUIDE v2.5 25-Apr-2017 13:47:29

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Belt1OptimisationGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @Belt1OptimisationGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



%888888888888888888888888888888888888888888888888888888888888888888888888
%Initialise
%888888888888888888888888888888888888888888888888888888888888888888888888
```

```
% --- Executes just before Belt1OptimisationGUI is made visible.
function Belt1OptimisationGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Belt1OptimisationGUI (see VARARGIN)

% Choose default command line output for Belt1OptimisationGUI
handles.output = hObject;
handles.HTD5M = varargin{1};
handles.HTD8M = varargin{2};
handles.HTD14M = varargin{3};
handles.BeltPitch = varargin{4};
handles.RPM = varargin{5};
handles.Power = varargin{6};
handles.ConditionsVec = varargin{7};
handles.PowerConditions = varargin{8};

% Update handles structure
guidata(hObject, handles);

% Populate list boxes
set(handles.BeltPitch_Pop,'String',varargin{4});

%set initial list box colours
set(handles.BeltWidth_Pop,'BackgroundColor',[0.94, 0.94, 0.94]);
set(handles.DrivePTeeth_Pop,'BackgroundColor',[0.94, 0.94, 0.94]);
set(handles.DrivePDia_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.One_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.GRatio_Pop,'BackgroundColor',[0.94, 0.94, 0.94]);
set(handles.DrivenPTeeth_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.DrivenPDia_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.DA_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.DB_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.BeltTeeth_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.BeltLength_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);
set(handles.ActualY2_Txt,'BackgroundColor',[0.83, 0.82, 0.78]);

% Setup initial images
imshow('Belt2PartA.png','Parent', handles.BeltGeo_Axis);

%Initialise warning text
set(handles.WarningTxt, 'ForegroundColor', 'red');

%set data stage for pulley sizing
set(handles.PulleySize_UI,'UserData',0);

% UIWAIT makes Belt1OptimisationGUI wait for user response (see UIRESUME)
uiwait(handles.GUI11);
```

```
%8888888888888888888888888888888888888888888888888888888888888888888888888
%Output Function
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Outputs from this function are returned to the command line.
function varargout = Belt1OptimisationGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%(navigation data non-zero)
%*************************************************************************

    %varargout{1} = Next GUI to be run
    %varargout{2} = (if = 1) => keep looping through results GUI else don't

    if get(handles.Done_PB,'UserData') ~= 0
        %Done PB was pressed to close GUI
        display('Done PB pressed');
        varargout{1} = 7;
        varargout{2} = 1;

        %Results Vector Data Creation
        %*****************************************************************

            %Get belt pitch index (index)
            index  = get(handles.BeltPitch_Pop,'Value');

            if index == 1
                %HTD5M belt selected
                BeltPitch = 5;

            elseif index == 2
                %HTD8M belt selected
                BeltPitch = 8;

            elseif index == 3
                %HTD14M belt selected
                BeltPitch = 14;

            else
                %throw error to console
                display('Output data creation error');
            end

            %Get belt width value (mm)
            contents = get(handles.BeltWidth_Pop,'String');
            index = get(handles.BeltWidth_Pop,'Value');
            DesiredBeltWidth = str2num(contents(index,:));

            %Get belt ratio (unitless)
            contents = get(handles.GRatio_Pop,'String');
            index = get(handles.GRatio_Pop,'Value');
            GearRatio = str2num(contents(index,:));
            %Get drive pulley teeth (#)
            contents = get(handles.DrivePTeeth_Pop,'String');
            index = get(handles.DrivePTeeth_Pop,'Value');
            DrivePulleyTeeth = str2num(contents(index,:));

            %Get drive pulley diameter (mm)
            DrivePulleyDia = str2num(get(handles.DrivePDia_Txt,'String'));

            %Get driven pulley teeth (#)
            DrivenPulleyTeeth = str2num(get(handles.DrivenPTeeth_Txt,'String'));

            %Get driven pulley diameter (mm)
            DrivenPulleyDia = str2num(get(handles.DrivenPDia_Txt,'String'));

            %Get X1 (mm)
            X1 = str2num(get(handles.X1_Txt,'String'));

            %Get Y1 (mm)
            Y1 = str2num(get(handles.Y1_Txt,'String'));

            %Get X2 (mm)
            X2 = str2num(get(handles.X2_Txt,'String'));

            %Get Y2 (mm)
            Y2 = str2num(get(handles.ActualY2_Txt,'String'));

            %Get belt length (teeth)
            BeltTeeth = str2num(get(handles.BeltTeeth_Txt,'String'));

            %Get belt length (mm)
            BeltLength = str2num(get(handles.BeltLength_Txt,'String'));

        %Create Output Vector Containing Results
        %*****************************************************************

            Belt2AResultsVector = [BeltPitch, DesiredBeltWidth, GearRatio, ↙
DrivePulleyTeeth, DrivePulleyDia, DrivenPulleyTeeth, DrivenPulleyDia, X1, Y1, X2, Y2, ↙
BeltTeeth, BeltLength];

            varargout{3} = Belt2AResultsVector;

    else
        %Unknown PB was pressed or user terminated
        display('Error: program closed unexpectedly')
    end

%Delete GUI
%*************************************************************************

    %The figure can now be deleted
    delete(handles.GUI11);
```

```
%888888888888888888888888888888888888888888888888888888888888888888888888
%Done Pushbutton
%888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes on button press in Done_PB.
function Done_PB_Callback(hObject, eventdata, handles)
% hObject    handle to Done_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if get(handles.PulleySize_UI,'UserData') >= 50
    %All stages complete
    set(handles.Done_PB,'UserData',1);

    close;

else
    %Not all stages complete

    %Write error to gui error txt
    WarningMessage = 'Not all data entered or calculated';
    set(handles.WarningTxt, 'String', WarningMessage);

end




%888888888888888888888888888888888888888888888888888888888888888888888888
%Create Textboxes and Objects
%888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function popupmenu3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function GRatio_Pop_CreateFcn(hObject, eventdata, handles)
% hObject    handle to GRatio_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function BeltWidth_Pop_CreateFcn(hObject, eventdata, handles)
% hObject    handle to BeltWidth_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function BeltPitch_Pop_CreateFcn(hObject, eventdata, handles)
% hObject    handle to BeltPitch_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function DrivePTeeth_Pop_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DrivePTeeth_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end

% --- Executes during object creation, after setting all properties.
function DrivenPTeeth_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DrivenPTeeth_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function DrivePDia_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DrivePDia_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function DrivenPDia_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DrivenPDia_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function One_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to One_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
```

```
function WarningTxt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to WarningTxt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% --- Executes during object creation, after setting all properties.
function DA_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DA_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function DB_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DB_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function DC_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DC_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function X1_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to X1_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end

% --- Executes during object creation, after setting all properties.
function Y1_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Y1_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function X2_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to X2_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function Y2_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Y2_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function ActualY2_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ActualY2_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes during object creation, after setting all properties.
function BeltTeeth_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to BeltTeeth_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function BeltLength_Txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to BeltLength_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end





%888888888888888888888888888888888888888888888888888888888888888888888888
%UI Code Interaction Code(Pushbuttons, Radio Buttons, etc.)
%888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes on selection change in BeltPitch_Pop.
function BeltPitch_Pop_Callback(hObject, eventdata, handles)
% hObject    handle to BeltPitch_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    %set correct colours (grey)
    %empty = {'Empty','Empty'};
    set(handles.GRatio_Pop,'BackgroundColor',[0.94, 0.94, 0.94]);
    set(handles.DrivePTeeth_Pop,'BackgroundColor',[0.94, 0.94, 0.94]);

    Checkvalue = 0;

    %read what value chosen from derired pitch pop up
    Checkvalue = get(handles.BeltPitch_Pop,'Value');
    %          = 1 = HTD5M
    %          = 2 = HTD8M
    %          = 3 = HTD14M

    %Run task to impliment avaliable widths
    if Checkvalue == 1
        %HTD5M was selected

        temp = handles.HTD5M.Pulley(1,:);

        %set correct colours (white)
        set(handles.BeltWidth_Pop,'BackgroundColor','white');

        %Write list to pop menu
        set(handles.BeltWidth_Pop,'String',temp);

    elseif Checkvalue == 2
        %HTD8M was selected

        temp = handles.HTD8M.Pulley(1,:);

        %set correct colours (white)
        set(handles.BeltWidth_Pop,'BackgroundColor','white');

        %Write list to pop menu
        set(handles.BeltWidth_Pop,'String',temp);

    elseif Checkvalue == 3
        %HTD14M was selected

        temp = handles.HTD14M.Pulley(1,:);

        %set correct colours (white)
        set(handles.BeltWidth_Pop,'BackgroundColor','white');

        %Write list to pop menu
        set(handles.BeltWidth_Pop,'String',temp);

    else
        %write error to console
        display('Selected belt does not mach any in data files');
    end

    %reset warning
    set(handles.WarningTxt, 'String', ' ');

    %Set correct data stage
    set(handles.PulleySize_UI,'UserData',10);


% --- Executes on selection change in BeltWidth_Pop.
function BeltWidth_Pop_Callback(hObject, eventdata, handles)
% hObject    handle to BeltWidth_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    %Run belt width selection only if previous stages complete
    if get(handles.PulleySize_UI,'UserData') >= 10
        %can proceede
        %set correct colours (grey)
        set(handles.DrivePTeeth_Pop,'BackgroundColor',[0.94, 0.94, 0.94]);

        %Find which pitch chosen
        Checkvalue = get(handles.BeltPitch_Pop,'Value');

        %Retrieve list of avaliable gear ratios

        if Checkvalue == 1
            %HTD5M was selected

            [RatioTable] = AGVBeltRatioSorter(handles.HTD5M.BeltCenter);

        elseif Checkvalue == 2
            %HTD8M was selected

            [RatioTable] = AGVBeltRatioSorter(handles.HTD8M.BeltCenter);

        elseif Checkvalue == 3
            %HTD14M was selected

            [RatioTable] = AGVBeltRatioSorter(handles.HTD14M.BeltCenter);

        else
            %Write error to console
            display('An error has occured in the ratio determination');

        end

        %reset warning
        set(handles.WarningTxt, 'String', ' ');

        %Set correct colours (white)
        set(handles.GRatio_Pop,'BackgroundColor',[1, 1, 1]);

        %Set correct data stage
        set(handles.PulleySize_UI,'UserData',20);

        %Write gear ratios to pop up menu
        set(handles.GRatio_Pop,'String',RatioTable);

    end

function One_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to One_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on selection change in GRatio_Pop.
function GRatio_Pop_Callback(hObject, eventdata, handles)
% hObject    handle to GRatio_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
    %Run belt width selection only if previous stages complete
    if get(handles.PulleySize_UI,'UserData') >= 20
        %can proceede
        
        %Get belt pitch index
        BeltPitch = get(handles.BeltPitch_Pop,'Value');
        
        %Get belt width value (mm)
        contents = get(handles.BeltWidth_Pop,'String');
        index = get(handles.BeltWidth_Pop,'Value');
        BeltWidth = str2num(contents(index,:));
        
        %Get belt ratio (unitless)
        contents = get(handles.GRatio_Pop,'String');
        index = get(handles.GRatio_Pop,'Value');
        GearTrainI = str2num(contents(index,:));
        
        %Get RPM (rad/sec)
        RPM = handles.RPM;
        
        %Get power (Watts)
        Power = handles.Power;
        
        %Run Drive Pulley Finder
        if BeltPitch == 1
            %HTD5M was selected
            
            [ PowerDrivesAccpt ] = AGVDrivePulleyPowerChecker(RPM, Power, BeltWidth,↙
(handles.HTD5M.Pulley), (handles.HTD5M.Power), (handles.HTD5M.PowerMult) ,GearTrainI,↙
(handles.ConditionsVec), (handles.PowerConditions));
            [ AcceptableDrives ] = AGVDrivePulleyRatioChecker(GearTrainI, (handles.↙
HTD5M.BeltCenter), PowerDrivesAccpt );
            
            
        elseif BeltPitch == 2
            %HTD8M was selected
            
            [ PowerDrivesAccpt ] = AGVDrivePulleyPowerChecker(RPM, Power, BeltWidth,↙
(handles.HTD8M.Pulley), (handles.HTD8M.Power), (handles.HTD8M.PowerMult), GearTrainI,↙
(handles.ConditionsVec), (handles.PowerConditions));
            [ AcceptableDrives ] = AGVDrivePulleyRatioChecker(GearTrainI, (handles.↙
HTD8M.BeltCenter), PowerDrivesAccpt );
            
        elseif BeltPitch == 3
            %HTD14M was selected
            
            [ PowerDrivesAccpt ] = AGVDrivePulleyPowerChecker(RPM, Power, BeltWidth,↙
(handles.HTD14M.Pulley), (handles.HTD14M.Power), (handles.HTD14M.PowerMult),↙
GearTrainI, (handles.ConditionsVec), (handles.PowerConditions));
            [ AcceptableDrives ] = AGVDrivePulleyRatioChecker(GearTrainI, (handles.↙
HTD14M.BeltCenter), PowerDrivesAccpt );
            
        end
        %Check if any drive pulleys are avaliable for chosen gear ratio

        if AcceptableDrives(1,1) ~= 0
            %The first acceptable drive pulley isn't zero => it exists

            %reset warning
            set(handles.WarningTxt, 'String', ' ');

            %Set correct colours (white)
            set(handles.DrivePTeeth_Pop,'BackgroundColor',[1, 1, 1]);

            %Write avaliable drivers to pop up menu
            set(handles.DrivePTeeth_Pop,'String',AcceptableDrives(1,:));

            %Write matching driven vector to UserData of "DrivenPTeeth_Txt"
            set(handles.DrivenPTeeth_Txt,'UserData',AcceptableDrives(2,:));

            %Set correct data stage
            set(handles.PulleySize_UI,'UserData',30);

        elseif AcceptableDrives(1,1) == 0
            %The first acceptable drive pulley doesn't exist => no pulleys
            %=> ask user to reconsider selections

            set(handles.WarningTxt, 'String', 'Drive pulley could not be found for↙
this belt pitch, power requirements and gear ratio combination. Please reselect either↙
the belt pitch or gear ratio');

        end
    end


% --- Executes on selection change in DrivePTeeth_Pop.
function DrivePTeeth_Pop_Callback(hObject, eventdata, handles)
% hObject    handle to DrivePTeeth_Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    %Run this stage only if previous stage complete
    if get(handles.PulleySize_UI,'UserData') >= 30

        %Determine drive pulley size (diameter) in mm, given number of teeth
        %and pitch of pulley
        %*****************************************************************

        %Get pulley size
        contents = get(handles.DrivePTeeth_Pop,'String');
        index = get(handles.DrivePTeeth_Pop,'Value');
        DriveTeeth = str2num(contents(index,:));

        %Get belt pitch index
        BeltPitch = get(handles.BeltPitch_Pop,'Value');


        %Perform diameter calculation ( = (pitch*teeth) / pi)
        if BeltPitch == 1
            %HTD5M was selected

            DpitchDrive = ((5*DriveTeeth) / pi());

        elseif BeltPitch == 2
            %HTD8M was selected

            DpitchDrive = ((8*DriveTeeth) / pi());

        elseif BeltPitch == 3
            %HTD14M was selected

            DpitchDrive = ((14*DriveTeeth) / pi());

        else
            %Write error to console
            display('Unknown belt pitch when calculatin drive pulley diameter');

        end

        %Write drive diameter to txt window
        set(handles.DrivePDia_Txt,'String',DpitchDrive);



        %Determine size of driven pulley given gear ratio
        %*****************************************************************

        index = get(handles.DrivePTeeth_Pop,'Value');
        DrivenTeethArray = get(handles.DrivenPTeeth_Txt,'UserData');
        DrivenTeeth = DrivenTeethArray(1,index);

        %Write drive diameter to txt window
        set(handles.DrivenPTeeth_Txt,'String',DrivenTeeth);



        %Determine diameter of drive pulley
        %*****************************************************************

        %Perform diameter calculation ( = (pitch*teeth) / pi)
        if BeltPitch == 1
            %HTD5M was selected

            DpitchDriven = ((5*DrivenTeeth) / pi());

        elseif BeltPitch == 2
            %HTD8M was selected

            DpitchDriven = ((8*DrivenTeeth) / pi());

        elseif BeltPitch == 3
            %HTD14M was selected
            DpitchDriven = ((14*DrivenTeeth) / pi());

        else
            %Write error to console
            display('Unknown belt pitch when calculatin drive pulley diameter');

        end

        %Write drive diameter to txt window
        set(handles.DrivenPDia_Txt,'String',DpitchDriven);


        %Write pulley diameters to geometry window
        %****************************************************************

        %DA pulley (Driver)
        set(handles.DA_Txt,'String',DpitchDrive);

        %DB pulley (Driven)
        set(handles.DB_Txt,'String',DpitchDriven);


        %Set Correct Data Stage
        %*****************************************************************

        %Set correct data stage
        set(handles.PulleySize_UI,'UserData',40);


    end

function DrivePDia_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to DrivePDia_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function DrivenPTeeth_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to DrivenPTeeth_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function DrivenPDia_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to DrivenPDia_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function DA_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to DA_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function DB_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to DB_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
function DC_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to DC_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function X1_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to X1_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function Y1_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to Y1_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function X2_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to X2_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function Y2_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to Y2_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in GeoCal_PB.
function GeoCal_PB_Callback(hObject, eventdata, handles)
% hObject    handle to GeoCal_PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    %Run this stage only if previous stage complete
    if get(handles.PulleySize_UI,'UserData') >= 40

        %Run desired belt length finder (non - standard belt)

        %Get pulley diameters from textboxes
        DA = str2num(get(handles.DrivePDia_Txt,'String'));  %calculated
        DB = str2num(get(handles.DrivenPDia_Txt,'String')); %calculated
        DC = str2num(get(handles.DC_Txt,'String'));         %user data

        %Get distances
        X1 = str2num(get(handles.X1_Txt,'String'));         %user data
        Y1 = str2num(get(handles.Y1_Txt,'String'));         %user data
        X2 = str2num(get(handles.X2_Txt,'String'));         %user data

        %Check if centre distance between pulleys is acceptable
        %****************************************************************

        [ Accept, MinC ] = AGVPulleyCentreDistanceChecker(X1, Y1, DA, DB);


        %Calculate desired belt length
        %****************************************************************
        if  Accept == 1
            %Center distance requirements met

            %Find ideal case, i.e. idler pulley just tangent to two pulley
            %system
            [ DesiredBL, Y2Initial ] = AGVDesiredBeltLengthCal(DA, DB, DC, X1, Y1,↙
X2);


            %Determine the closest real life belt that matched the closen belt
            %*****************************************************************

            %Get belt pitch index
            BeltPitch = get(handles.BeltPitch_Pop,'Value');

            if BeltPitch == 1
                %HTD5M was selected

                [ ChoosenBeltLength ] = AGVClosestActualBeltToDesiredBeltFinder(↙
DesiredBL, handles.HTD5M.BeltCenter );
                ChoosenBeltTeeth = ChoosenBeltLength/5;

            elseif BeltPitch == 2
                %HTD8M was selected

                [ ChoosenBeltLength ] = AGVClosestActualBeltToDesiredBeltFinder(↙
DesiredBL, handles.HTD8M.BeltCenter );
                ChoosenBeltTeeth = ChoosenBeltLength/8;

            elseif BeltPitch == 3
                %HTD14M was selected

                [ ChoosenBeltLength ] = AGVClosestActualBeltToDesiredBeltFinder(↙
DesiredBL, handles.HTD14M.BeltCenter );
                ChoosenBeltTeeth = ChoosenBeltLength/14;

            else
                %Write error to console
                display('Error when attempting to match real belt to desired belt');

            end



            %Write Belt lengths to GUI & Find Y2 value
            %*****************************************************************

            if ChoosenBeltLength ~= 0
                %Belt exist, ie doesn't equal zero

                %Set output value for belt teeth number
                set(handles.BeltTeeth_Txt,'String',ChoosenBeltTeeth);

                %Set output value for belt length
                set(handles.BeltLength_Txt,'String',ChoosenBeltLength);

                %Find Y2 value
                [Y2BestMatch, ValFound] = AGVFindIdlerPulleyYDistance↙
(ChoosenBeltLength, Y2Initial, DA, DB, DC, X1, Y1, X2);

                if ValFound == 1
                    %Y2 could be found

                    %Set output value for belt length
                    set(handles.ActualY2_Txt,'String',Y2BestMatch);

                    %Set correct stage
                    set(handles.PulleySize_UI,'UserData',50);

                else
                    %Y2 could not be found

                    %Write warning to window
                    WarningMessage = 'Iterative search for Y2 yielded no useful↙
results, consider changing geometry';

                    %Write error to gui error txt
                    set(handles.WarningTxt, 'String', WarningMessage);

                end

            elseif ChoosenBeltLength == 0
                %Belt doesn't exist

                %Set output value for belt teeth number to zero
                set(handles.BeltTeeth_Txt,'String','0');

                %Set output value for belt length to zero
                set(handles.BeltLength_Txt,'String','0');

                WarningMessage = 'No exists that is long enough, please change drive↙
geometry of belt pitch';

                %Write error to gui error txt
                set(handles.WarningTxt, 'String', WarningMessage);
            end




        elseif Accept == 0
            %Center distance requirements not met

            WarningPt1 = 'Minimum center distance requirements not met, Minimum center↙
distance is: ';
            WarningPt2 = num2str(MinC, 2);
            WarningPt3 = ' mm. Please edit drive geometry';

            WarningMessage = strcat(WarningPt1, WarningPt2, WarningPt3);
            %Write error to gui error txt
            set(handles.WarningTxt, 'String', WarningMessage);

        else
            %throw error to console
            display('Unknown state for calculating desired belt length');

        end

    else
        %Display error message
        set(handles.WarningTxt, 'String', 'Complete pulley sizing first');

    end

function BeltTeeth_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to BeltTeeth_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function BeltLength_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to BeltLength_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function ActualY2_Txt_Callback(hObject, eventdata, handles)
% hObject    handle to ActualY2_Txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)




%8888888888888888888888888888888888888888888888888888888888888888888888888
%Close GUI
%8888888888888888888888888888888888888888888888888888888888888888888888888

% --- Executes when user attempts to close GUI11.
function GUI11_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to GUI11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if isequal(get(hObject, 'waitstatus'),'waiting')
    %The GUI is still in UIWAIT, use UIRESUME
    uiresume(hObject);
else
    %The GUI is nolonger waiting and can be closed
    delete(hObject);
end

% --- Executes during object deletion, before destroying properties.
function GUI11_DeleteFcn(hObject, eventdata, handles)
```

```
% hObject    handle to GUI11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles = rmfield(handles, 'HTD5M');
handles = rmfield(handles, 'HTD8M');
handles = rmfield(handles, 'HTD14M');
```

G.13 Generalised Reused Code

This code is used by multiple different GUIs, hence it is not listed under a specific GUI.

```matlab
function [  RatioTable ] = AGVBeltRatioSorter( BeltCentreTable )
%AGVBeltRatioSorter
%8888888888888888888888888888888888888888888888888888888888888888888888888
%   This function creates a list of avaliable gear ratios given the centre
%   distance table of a specific pitch

%   Inputs
%   **********************************************************************

%   BeltCentreTable     = the centre distance table of a given belt system

%   Outputs
%   **********************************************************************

%   RatioTable          = a list of all avaliable ratios for a given belt
%                         pitch (unitless)


%8888888888888888888888888888888888888888888888888888888888888888888888888
%Create Extended Ratio Table (duplicate ratios will be present)
%8888888888888888888888888888888888888888888888888888888888888888888888888

    [n m] = size(BeltCentreTable);

    RatioExtendedTable = zeros(1,n-3);

    for i = 4:n
        %for each ith row (ratio)

        RatioExtendedTable(1,i-3) = BeltCentreTable(i,1);

    end

%8888888888888888888888888888888888888888888888888888888888888888888888888
%Remove Duplicate ratio values from table
%8888888888888888888888888888888888888888888888888888888888888888888888888

    temp = zeros(1,n-3);
    NoUniqueRatios = 0;
    PreviousValue = 0;

    for i = 1 : (n-3)
        %For each ratio in the ratio table

        if RatioExtendedTable(1,i) ~= PreviousValue
            %Does not match previous value

            %Write unique value to vector
            temp(1,NoUniqueRatios+1) = RatioExtendedTable(1,i);

            %Write unique value to PreviousValue variable
            PreviousValue = RatioExtendedTable(1,i);

            %Incriment the number of unique ratios
            NoUniqueRatios = NoUniqueRatios + 1;
        end
    end

%888888888888888888888888888888888888888888888888888888888888888888888888
%Remove Empty Values on Vetor "Temp" to Create RatioTable
%888888888888888888888888888888888888888888888888888888888888888888888888

    RatioTable = zeros(1,NoUniqueRatios);

    for i = 1 : NoUniqueRatios
        %for each unique ratio in "temp"

        RatioTable(1,i) = temp(1,i);

    end
end
```

```
function [ AcceptableDrive ] = AGVDrivePulleyPowerChecker( OmegaDriven, Power,↙
BeltWidth, DriverList, PowerTable, PowerMult ,GearRatio, ConditionsVec,↙
PowerConditions)
%AGVDrivePulleyPowerChecker
%88888888888888888888888888888888888888888888888888888888888888888888888888
%   This function checks if the pulleys in the supplied driver list are
%   able to handle to power that is to be transfered

%   Inputs
%   *********************************************************************

%   OmegaDriven     = the desired driven RPM (rad/sec)
%   Power           = the power to be transfered through the system (W)
%   BeltWidth       = the width of belt we are using (mm)
%   DriverList      = list of all pulleys in the pitch & belt size
%   PowerTable      = pulley power table
%   GearRatio       = the desired gear ratio for this stage

%   Outputs
%   *********************************************************************

%   AcceptableDrive = a list of all acceptable driver pulleys that meet
%                     power requirements

%   RPM_OK          = states whether the RPM value is in the tables range
%                     (bool)

%   Width_Fac_OK   = states whether the width factor exists
%                    (bool)


%8888888888888888888888888888888888888888888888888888888888888888888888888
%Initialise Values
%8888888888888888888888888888888888888888888888888888888888888888888888888

%   Set RPM_OK to 0(i.e. RPM out of range, set to 1 later in code if found)
%   *********************************************************************
    RPM_OK = 0;

%   Set Width_Fac_OK to 0 (i.e. out of range, set in code to 1)
%   *********************************************************************
    Width_Fac_OK = 0;

%   Find the RPM of the driver (rpm)
%   *********************************************************************
    RPMDriver = (OmegaDriven*(60/(2*pi()))) * GearRatio;



%8888888888888888888888888888888888888888888888888888888888888888888888888
%Find Power Vector From Power Table (Given Driver RPM) & Isolate Pulleys
%8888888888888888888888888888888888888888888888888888888888888888888888888

%   Find RPM on power table
%   *********************************************************************

    stop = 0;   %loop terminator
    i = 2;      %loop incrimentor (start at 3; 1st, 2nd position = heading)

    %find the number of RPMs that must be searched through
    [n m] = size(PowerTable);

    %Initialise index for RPM vlaues
    UpperRPMIndex = 0;
    LowerRPMIndex = 0;

    %Create loop to search rpm values and find either an exact match or
    %interpolate a value between given RPMs

    while (i <= n) && (stop == 0)
        %while number of rows not exceeded and stop not triggered (i.e.
        %value not found)

        %find the current RPM value from the table
        CurrentRPM = PowerTable(i,1);    %column 1 = rpm values

        if RPMDriver == CurrentRPM
            %absolute match found (unlikly)

            UpperRPMIndex = i;
            LowerRPMIndex = i;

            %value found = terminate search loop
            stop = 1;

            %value found = set RPM_OK = 1
            RPM_OK = 1;

        elseif CurrentRPM > RPMDriver
            %RPM found just larger than desired RPM

            UpperRPMIndex = i;
            LowerRPMIndex = i-1;

            %value isolated = terminate search loop
            stop = 1;

            %value isolated = set RPM_OK = 1
            RPM_OK = 1;

        end

        %value not found => incriment search index
        i = i + 1;

    end


%   Using upper and lower RPM indexes develop a power vector relating to
%   each driving gear
%   *********************************************************************

    %Create empty power vector
    PowerVector = zeros(1, (m-1));

    %Create pulley teeth vector
    PowerPulleyVector = zeros(1, (m-1));

    if RPM_OK == 1
        %RPM exists within the tables bounds

        if (UpperRPMIndex == LowerRPMIndex) && (UpperRPMIndex > 2)
            %if upper index and lower index match an absolute match was found,
            %this match must be non-zero and as such the upper index cannot be
            %1 as 1 = headings OR 2 as 2 = 0 RPM

            for i = 2:m
                % for each column in the power table except i=1 as 1 = heading

                temp = PowerTable(UpperRPMIndex, i);

                if temp ~= -99
                    %Value exists

                    %Copy ith power value
                    PowerVector(1,i-1) = temp;

                    %Copy ith pulley to vector
                    PowerPulleyVector(1,i-1) = PowerTable(1,i);

                elseif temp == -99
                    %Value doesnt exist

                    %Copy zero to power table
                    PowerVector(1,i-1) = 0;

                    %Copy 0 to vector
                    PowerPulleyVector(1,i-1) = 0;

                end

            end

        elseif (UpperRPMIndex ~= LowerRPMIndex) && (UpperRPMIndex > 2)
            %if upper index does not match lower index then a value needs to be
            %interpolated provided that the upper index is no zero thus it must
            %sit at an index > 2

            %find RPM values for linear interpolation
            x1 = PowerTable(LowerRPMIndex, 1);
            x2 = RPMDriver;
            x3 = PowerTable(UpperRPMIndex, 1);

            for i = 2:m
```

```
                %for each column in the power table interpolate the power value

                %find power values for interpolation
                y1 = PowerTable(LowerRPMIndex,i);
                y3 = PowerTable(UpperRPMIndex,i);

                if (y1 ~= -99) && (y3 ~= -99)
                    %neither y1 OR y2 equal -99, thus values exist

                    %Interpolate ith power value
                    PowerVector(1,i-1) = (((x2 - x1)*(y3 - y1))/(x3 - x1)) + y1;

                    %Copy ith pulley to vector
                    PowerPulleyVector(1,i-1) = PowerTable(1,i);

                elseif (y1 == -99) || (y3 == -99)
                    % Either y1 OR y3 are = -99, thus value doesn't exist

                    %set power vector to zero
                    PowerVector(1,i-1) = 0;

                    %Copy 0 to vector
                    PowerPulleyVector(1,i-1) = 0;

                end
            end
        end
    end




%888888888888888888888888888888888888888888888888888888888888888888888888
%Correct Power Vector Using Multipliers
%888888888888888888888888888888888888888888888888888888888888888888888888

%   Convert table from kW to W
%   *********************************************************************
    PowerVector = 1000*PowerVector;



%   Apply Belt Width Multiplier
%   *********************************************************************

%   Find size of power multiplication table
    [n m] = size(PowerMult);

%   Search for matching belt width index

    stop = 0;   %force loop termination = 1
    i = 1;      %loop incrimentor

%   Initialise index value to zero
```

```
    WidthFactor = 0;

    while (i <= m) && (stop == 0)
        %for each column i

        if PowerMult(1,i) == BeltWidth
            %ith column matches chosen belt

            %Find width factor
            WidthFactor = PowerMult(2,i);

            %terminate loop
            stop = 1;

            %set width factor found to OK
            Width_Fac_OK = 1;

        end

        i = i + 1;

    end

%   Multiply PowerVector by Width factor
    PowerVector = WidthFactor*PowerVector;




%888888888888888888888888888888888888888888888888888888888888888888888888
%Correct Required Power Using Multipliers
%888888888888888888888888888888888888888888888888888888888888888888888888

%   Find operating conditions service factor
%   *********************************************************************

%   Determine service factor

    StartCon = ConditionsVec(1,1);
    LoadCon = ConditionsVec(1,2);
    DutyCycle = ConditionsVec(1,3);

    if StartCon == 1
        %Soft start selected for starting condition

        ServiceFactor = PowerConditions(LoadCon+2, DutyCycle);

    elseif StartCon == 2
        %Heavy Start selected for starting conditions

        ServiceFactor = PowerConditions(LoadCon+2, DutyCycle+3);

    end
```

```
%   Find extra idle pulley correction factor
%   *********************************************************************

%   Determine idle pulley correction factor
    IdlePulleyCorrrection = 0.2;


%   Sum correction factors and multiply with desired power
%   *********************************************************************

    Power = (ServiceFactor + IdlePulleyCorrrection) * Power;





%888888888888888888888888888888888888888888888888888888888888888888888888
%Identify Suitable Pulley from Pulley List Based on Power Requirements
%888888888888888888888888888888888888888888888888888888888888888888888888

    %Determine max possible number of pulleys
    l = length(PowerVector);

    %Create an empty vector for pulleys that meet the power requirements
    AcceptablePulleysPow = zeros(1,l);

    %Create an acceptable pulley counter
    AccptPulleyCnt = 1;

    for i = 1:l
        %for each pulley in the power vector

        if PowerVector(1,i) >= Power
            %ith pulley >= desired power requirements

            %Write size (teeth) of accepted pulley to
            %"AcceptablePulleysPow"
            AcceptablePulleysPow(1,AccptPulleyCnt) = PowerPulleyVector(1,i);

            %Incriment acceptable pulley counter
            AccptPulleyCnt = AccptPulleyCnt + 1;
        end
    end




%888888888888888888888888888888888888888888888888888888888888888888888888
%Compare Avaliable Driver Pulleys to Pulley that Meet Power Requirements
%888888888888888888888888888888888888888888888888888888888888888888888888

    %Identify correct column in pulley table
    %********************************************************************
    [n m] = size(DriverList);

    %Set up loop conditions
    stop = 0;
    i = 1;

    while (i <= m) && (stop == 0)
        %for each cloumn in DriverList

        if DriverList(1,i) == BeltWidth
            %belt heading on column matches chosen belt

            %record column index to extract pulley sizes later
            BeltIndex = i;

            %terminate loop on match
            stop = 1;

        end

        %Incriment counter
        i = i + 1;

    end

    %Compare tables
    %********************************************************************

    %Create temporary vector to hold output pulleys
    temp = zeros(1, max( [l ,n-1] ));

    %Count number of values added to temp
    tempCnt = 0;

    for i = 1:l
        %for each possible index in "AcceptablePulleysPow"

        CurrentPulley1 = AcceptablePulleysPow(1,i);

        if CurrentPulley1 ~= 0
            %current pulley is non-zero

            for j = 2:n
                %for each possible index in "DriverList"s chosen column,
                %ignore first row => heading

                if CurrentPulley1 == DriverList(j,BeltIndex)
                    %Match found between tables

                    %Incriment temp entry counter
                    tempCnt = tempCnt + 1;

                    %Add pulley to temp vector
                    temp(1,tempCnt) = CurrentPulley1;
                end
            end
        end
    end





%888888888888888888888888888888888888888888888888888888888888888888888888
%Create Output Vector: "AcceptableDrive"
%888888888888888888888888888888888888888888888888888888888888888888888888


    if tempCnt ~= 0

        %Initialise Vector
        AcceptableDrive = zeros(1,tempCnt);

        %Record values from temp to "AcceptableDrive" without zeros
        for i = 1:tempCnt

            AcceptableDrive(1,i) = temp(1,i);

        end

    else
        %No acceptable pulleys found

        AcceptableDrive = 0;

    end

end
```

```
function [ AcceptablePulleys ] = AGVDrivePulleyRatioChecker( GearRatio, BeltTable, ↙
PulleyTable )
%AGVDrivePulleyRatioChecker
%8888888888888888888888888888888888888888888888888888888888888888888888888
%   This function checks if the choosen pulleys can be used for the desired
%   gear ratio. That is to say does the appropriate driven gear exist to
%   achieve the ratio

%   Inputs
%   **********************************************************************

%   GearRatio       = the desired gear ratio (unitless)
%   BeltTable       = belt table contains a list of all possible
%                     combinations for each ratio
%   PulleyTable     = all avaliable pulleys list

%   Output
%   **********************************************************************

%   AcceptablePulleys   = a list of all pulleys that fit the ratio requirements




%8888888888888888888888888888888888888888888888888888888888888888888888888
%Find list of all drive pulleys for given ratio
%8888888888888888888888888888888888888888888888888888888888888888888888888

    [n m] = size(BeltTable);

    %create empty drive pulley vector (for given ratio)
    AcceptableRatioDriver = zeros(2,n); %row 1: Driver
                                        %row 2: Driven

    %Loop initial conditions
    Stop = 0;        %force loop stop condition
    i = 4;           %loop index (start at 4 as 1-3 are headings)
    PulleyCnt = 0;   %pulley counter

    %Find starting index of avaliable pulleys
     while (i <= n) && (Stop == 0);
         %loop through all avaliable pulleys and have not passed desired
         %ratio

         %Current value (gear ratio) to be evalauted
         current = BeltTable(i,1);

         if current == GearRatio
             %current ratio a match to desired ratio

             %Incriment pulley counter
             PulleyCnt = PulleyCnt + 1;

             %Write current pullet size to list of accepted ratio pulleys
             AcceptableRatioDriver(1,PulleyCnt) = BeltTable(i,2);
```

```
            AcceptableRatioDriver(2,PulleyCnt) = BeltTable(i,3);

        elseif (PulleyCnt ~= 0) && (current ~= GearRatio)
            %Moved past current gear ratios and pulleys nolonger match

            %Force loop termination
            Stop = 1;

        end

        %Incriment counter
        i = i + 1;

    end

    %Remove zero values from "AcceptableRatioDriver"
    AcceptableRatioDriver2 = zeros(2,PulleyCnt);

    for i = 1:PulleyCnt
        %write all non-zero values to "AcceptableRatioDriver2" from
        %"AcceptableRatioDriver"

        AcceptableRatioDriver2(1,i) = AcceptableRatioDriver(1,i);
        AcceptableRatioDriver2(2,i) = AcceptableRatioDriver(2,i);

    end




%888888888888888888888888888888888888888888888888888888888888888888888888
%Match Drive Pulleys that Match Power Requirements with Drive Pulleys
%that Match Power Requirements
%888888888888888888888888888888888888888888888888888888888888888888888888

m = length(PulleyTable);
n = min( [PulleyCnt, m] );

%Number of matches found set to zero
MatchCnt = 0;

    if m ~= 0
        %"PulleyTable" is not empty

        %Initialise empty vector to store matched values
        OutputVector = zeros(2,n);

        for i = 1:m
            %For each pulley in "PulleyTable" attempt to find match

            current = PulleyTable(1,i);

            %Set up initial conditions
            Stop = 0;
            j = 1;

            while (j <= PulleyCnt) && (Stop == 0)
                %While in bounds and match not found (stop = 1 when matched)

                if current == AcceptableRatioDriver2(1,j)
                    %match found

                    %Incriment match counter
                    MatchCnt = MatchCnt + 1;

                    %Write drive & driven pulleys to output vector
                    OutputVector(1,MatchCnt) = AcceptableRatioDriver2(1,j);
                    OutputVector(2,MatchCnt) = AcceptableRatioDriver2(2,j);

                    %Match found = force loop stop
                    Stop = 1;

                end

                %Incriment loop counter
                j = j + 1;

            end
        end
    end




%888888888888888888888888888888888888888888888888888888888888888888888888
%Remove Zeros From Output Vector
%888888888888888888888888888888888888888888888888888888888888888888888888

    if MatchCnt ~= 0
        %At least one pulley match found

        %Create empty return vector
        AcceptablePulleys = zeros(2,MatchCnt);

        for i = 1:MatchCnt
            %for each non-zero value in match count => write to return vector

            AcceptablePulleys(1,i) = OutputVector(1,i);
            AcceptablePulleys(2,i) = OutputVector(2,i);

        end

    elseif MatchCnt == 0
        %No matches found

        %return empty vector
        AcceptablePulleys = zeros(2,1);
    else
        %Throw error to console
        display('Unknown state when removing zeros from pulley size ratio vector');

    end


end
```

```
function [ Result, MinCenterD ] = AGVPulleyCentreDistanceChecker( X1, Y1, DA, DB )
%AGVPulleyCentreDistanceChecker
%888888888888888888888888888888888888888888888888888888888888888888888888

%   Given X1 and Y1, the center distance between the pulleys is calculated,
%   this value is then compared to the minimun possible value.

%   Inputs
%   *********************************************************************

%   X1      = horizontal distance between driver and driven pulley  (mm)
%   Y1      = vertical distance between driver and driven pulley    (mm)

%   DA      = diameter of driver pulley                             (mm)
%   DB      = diameter of driven pulley                             (mm)

%   Outputs
%   *********************************************************************

%   Result  = boolean output, centre distance acceptable = 1, not
%             acceptable = 0

%   MinC    = minumum centre distance                           (mm)




%8888888888888888888888888888888888888888888888888888888888888888888888888
%Desired Centre Distance
%8888888888888888888888888888888888888888888888888888888888888888888888888

    if (Y1 * X1) ~= 0
        %Both values non-zero

        WantedCenterD = Y1 /  sin( atan(Y1/X1) );

    elseif Y1 == 0

        WantedCenterD = X1;

    elseif X1 == 0

        WantedCenterD = Y1;

    end




%8888888888888888888888888888888888888888888888888888888888888888888888888
%Minimun Centre Distance
%888888888888888888888888888888888888888888888888888888888888888888888888

    MinCenterD = (DA/2)+(DB/2);
```

```
%8888888888888888888888888888888888888888888888888888888888888888888888888
%Compare Desired Centre Distance and Minimum Centre Distance
%8888888888888888888888888888888888888888888888888888888888888888888888888

    if WantedCenterD >= MinCenterD
        %Desired centre distance meets minimum length requirements

        Result = 1;

    else
        %Desired centre distance doesn't meet minimum length requirements

        Result = 0;

    end

end
```

```
function [ DesiredBeltLength, Y2initial ] = AGVDesiredBeltLengthCal( DA, DB, DC, X1,↙
Y1, X2 )
%AGVDesiredBeltLengthCal
%888888888888888888888888888888888888888888888888888888888888888888888888

%   This calculator finds the belt length of a three pulley system if the
%   diameters of each pulley are known along with thier center points. The
%   belt length returned will most likly not match a standard belt length.

%   Inputs
%   **********************************************************************

%   DA  = diameter of pulley A (driver pulley)          (mm)
%   DB  = diameter of pulley B (driven pulley)          (mm)
%   DC  = diameter of pulley C (idler pulley)           (mm)

%   X1  = horizontal distance between A and B           (mm)
%   Y1  = vertical distance between A and B             (mm)
%   X2  = horizontal distance between A and C           (mm)

%   Outputs
%   *********************************************************************

%   DesiredBeltLength   = the desired belt length       (mm)
%   Y2initial           = starting point for Y2 value   (mm)




%888888888888888888888888888888888888888888888888888888888888888888888888
%Find BL of Two Pulley System
%888888888888888888888888888888888888888888888888888888888888888888888888

    %Find centre distance of two belt system (mm)

    if (X1*Y1) ~= 0
        %Both X1 and Y1 non-zero

        TBC = X1/(cos(atan(Y1/X1)));

    elseif X1 == 0
        %X1 is zero

        TBC = Y1;

    elseif Y1 == 0
        %Y1 is zero

        TBC = X1;

    end

    %Theta of pulley A (radians)
    TBTheta = 2*acos((DB-DA)/(2*TBC));
    %TBC' belt length (mm)
    TBCL = TBC*sin(TBTheta/2);

    %Find arc lengths of belt (mm)
    SaTB = TBTheta*(DA/2);
    SbTB = (2*pi() - TBTheta)*(DB/2);

    %Ideal belt length for two puley system (mm)
    DesiredBeltLength = (2*TBCL) + SaTB + SbTB;




%888888888888888888888888888888888888888888888888888888888888888888888888
%Find Tangent Point of 20mm Pulley
%888888888888888888888888888888888888888888888888888888888888888888888888

    %Find Theta Prime
    %********************************************************************
        %Theta 1
        if (Y1*X1) ~= 0
            Theta1 = atan(Y1/X1);

        elseif X1 == 0
            Theta1 = 90;

        elseif Y1 == 0
            Theta1 = 0;

        end


        %Theta 2
        Theta2 = TBTheta/2;

        %Theta 3
        Theta3 = Theta2 - Theta1;

        %ThetaPrime
        ThetaPrime = (pi()/2) - Theta3;

    %Find Xcor & Ycor
    %********************************************************************

        %These are the correction fators from the centre point of the idle
        %pulley to the tangent point

        Xcor = (DC/2)*sin(ThetaPrime);

        Ycor = (DC/2)*cos(ThetaPrime);

    %Find initial Q position for the idler pulley given an x position
    %*********************************************************************
```

```
        Yq = (X2 + Xcor)*tan(ThetaPrime) + Ycor;

    %Find length AQ (from pulley A centre point to point Q)
    %*********************************************************************

        %Find include angle
        ThetaQii = (pi()/2) - Theta2 + Theta1;

        %Find AQ
        lAQ = (DA/2)/cos(ThetaQii);

    %Calculate initial Y2
    %*********************************************************************

        Y2initial = Yq + lAQ;

end
```

```
function [ BestMatchLength ] = AGVClosestActualBeltToDesiredBeltFinder( DesiredLength,↙
BeltTable )
%AGVClosestActualBeltToDesiredBeltFinder
%8888888888888888888888888888888888888888888888888888888888888888888888888

%   Given the length of the desired belt find the closest matching real
%   life belt.

%   Input
%   **********************************************************************

%   DesiredLength   = desired length of the belt            (mm)
%   BeltTable       = table of avaliable belts

%   Output
%   **********************************************************************

%   BestMatchLength = the length of the best matching belt  (mm)


%8888888888888888888888888888888888888888888888888888888888888888888888888
%Extract Belts from Centre Distance Table
%8888888888888888888888888888888888888888888888888888888888888888888888888

%   Find center distance table size
    [n m] = size(BeltTable);

%   Create empty vector to hold belt lengths
    AllBeltLengths = zeros(1,(m-3));


    for i = 4:m     %start at 4 as 1-3 => headings
        %for each column containing a belt length

        %Write belt length to list
        AllBeltLengths(1,i-3) = BeltTable(1,i);

    end

%8888888888888888888888888888888888888888888888888888888888888888888888888
%Find Best Matching Belt
%8888888888888888888888888888888888888888888888888888888888888888888888888

%   Initialise best match to zero
    BestMatchLength = 0;

%   Initialise loop conditions
    Stop = 0;   %forces loop termination
    i = 1;      %loop incrimentor

    while (i <= (m-3)) && (Stop == 0)

        CurrentLength = AllBeltLengths(1,i);

        %Check if current belt is just larger than desired belt
        if CurrentLength >= DesiredLength

            %Terminate loop
            Stop = 1;

            %Write current length to "BestMatchLength"
            BestMatchLength = CurrentLength;

        end

        %Incriment loop counter
        i = i + 1;

    end

end
```

```
function [ Y2, ValFound ] = AGVFindIdlerPulleyYDistance( ActualBeltLength, Y2Initial,↙
DA, DB, DC, X1, Y1, X2 )
%AGVFindIdlerPulleyYDistance
%888888888888888888888888888888888888888888888888888888888888888888888888

%   Iterrative calculator that seeks to find the Y2 position of the idler
%   pulley given the length of the chosen standard belt

%   Inputs
%   ********************************************************************

%   ActualBeltLength    = The length of the actual belt chosen      (mm)
%   Y2Initial           = Intial vertical height of idler pulley    (mm)

%   DA                  = Diameter of pulley A (Driver)             (mm)
%   DB                  = Diameter of pulley B (Driven)             (mm)
%   DC                  = Diameter of pulley C (Ideler)             (mm)

%   X1                  = AB horizontal distance                    (mm)
%   Y1                  = AB vertical distance                      (mm)
%   X2                  = AC horizonatal distance                   (mm)

    tollerence          = 0.01;
    LoopLimit           = 300;

%   Outputs
%   ********************************************************************

%   Y2                  = best match for Y2                         (mm)
%   ValFound            = was Y2 value found                        (bool)




%888888888888888888888888888888888888888888888888888888888888888888888888
%Positive quadrant calculation
%888888888888888888888888888888888888888888888888888888888888888888888888

    %Initialisation
    %*******************************************************************

        %Loop parameters
        Stop = 0;                   %Loop force stop
        Hold = Y2Initial;           %Initial space element of
                                    %(0,Y2Initial)
        loopsafe = 0;               %Loop counter

         %Create empty BL results vector
         BL = zeros(1,2);

         %Initialise current Y2 value
         Y2 = Y2Initial-0.0001;

         %Initialise "ValFound" to zero
         ValFound = 0;

    %Begin optimatation iteration
    %*********************************************************************

        while (Stop == 0) && (loopsafe < LoopLimit)
            %Loop while stop conditions not met

            %Nested Loop initialisation and update
            %*************************************************************

                %incriment loop counter
                loopsafe = loopsafe + 1;

                pointer = [ Y2, (Y2 - 0.5*Hold) ];
                    %pointer1 = Y2;
                    %pointer2 = Y2 - 0.5*Hold

            %Find Belt length given current pointers
            %*************************************************************

                for i = 1:2 %Run equation for both pointers

                    %Find the belt length given the current Y2
                    BL(1,i) = ThreePulBeltLoopLengthCal(X1, Y1, X2, pointer(1,i), DA,↙
DB, DC);

                end

            %Identify Which Faction is Closer to True Result
            %*************************************************************
                %FACTIONS:
                %DifLowerFact = ActualBeltLength - BL(1,1);
                %DifUpperFact = ActualBeltLength - BL(1,2);


                %Y2 in the lower faction
                %*********************************************************

                    if ActualBeltLength < BL(1,2)
                        %lower faction has better match

                        DifLowerFact = ActualBeltLength - BL(1,1);

                        %Check if termination conditions met
                        if DifLowerFact <= tollerence
                            %termintion conditions met

                            %terminate loop
                            Stop = 1;
                            Y2 = pointer(1,1);

                            %Value found
                            ValFound = 1;
                        else
                            %termination conditions not met

                            %pointer = [ Y2, (Y2 - 0.5*Hold) ];
                            Y2 = pointer(1,1);
                            Hold = Hold/2;

                        end

                %Y2 in the upper faction
                %*********************************************************

                    elseif ActualBeltLength > BL(1,2)
                        %upper faction match

                        DifUpperFact = ActualBeltLength - BL(1,2);

                        %Check if termination conditions met
                        if DifUpperFact <= tollerence
                            %termination conditions met

                            %terminate loop
                            Stop = 1;
                            Y2 = pointer(1,2);

                            %Value found
                            ValFound = 1;

                        else
                            %termination conditions not met

                            %pointer = [ Y2, (Y2 - 0.5*Hold) ];
                            Y2 = pointer(1,2);
                            Hold = Hold/2;

                        end

                %Y2 in unknown fation
                %*********************************************************

                    else
                        %Unknown state = write to console
                        display('Y2 value in neither region');

                    end
        end



%8888888888888888888888888888888888888888888888888888888888888888888888888
%Negative quadrant calculation
%8888888888888888888888888888888888888888888888888888888888888888888888888
```

```
    %Check if optimisation occurs in this quadrant as it didn't occur in
    %previous quadrant
    if ValFound == 0

        %Write to console (for debuging)
        display('Running negative quadrant check');

        %Initialisation
        %*****************************************************************

            %Loop parameters
            Stop = 0;                   %Loop force stop
            Hold = Y2Initial;           %Initial space element of
                                        %(0,Y2Initial)
            loopsafe = 0;               %Loop counter

             %Create empty BL results vector
             BL = zeros(1,2);

             %Initialise current Y2 value
             Y2 = -(Y2Initial-0.0001);

        %Begin optimatation iteration
        %*****************************************************************

            while (Stop == 0) && (loopsafe < LoopLimit)
                %Loop while stop conditions not met

                %Nested Loop initialisation and update
                %*********************************************************

                    %incriment loop counter
                    loopsafe = loopsafe + 1;

                    pointer = [ Y2, (Y2 + 0.5*Hold) ];
                        %pointer1 = -Y2;
                        %pointer2 = -Y2 + 0.5*Hold

                %Find Belt length given current pointers
                %*********************************************************

                    for i = 1:2 %Run equation for both pointers

                        %Find the belt length given the current Y2
                        BL(1,i) = ThreePulBeltLoopLengthCal(X1, Y1, X2, pointer(1,i),↙
DA, DB, DC);

                    end

                %Identify Which Faction is Closer to True Result
                %*********************************************************
                    %FACTIONS:
                    %DifLowerFact = ActualBeltLength - BL(1,1);
                    %DifUpperFact = ActualBeltLength - BL(1,2);
                    %Y2 in the lower faction
                    %******************************************************

                        if ActualBeltLength > BL(1,2)
                            %lower faction has better match

                            DifLowerFact = abs(ActualBeltLength - BL(1,1));

                            %Check if termination conditions met
                            if DifLowerFact <= tollerence
                                %termintion conditions met

                                %terminate loop
                                Stop = 1;
                                Y2 = pointer(1,1);

                                %Value found
                                ValFound = 1;

                            else
                                %termination conditions not met

                                %pointer = [ Y2, (Y2 - 0.5*Hold) ];
                                Y2 = pointer(1,1);
                                Hold = Hold/2;

                            end

                    %Y2 in the upper faction
                    %******************************************************

                        elseif ActualBeltLength < BL(1,2)
                            %upper faction match

                            DifUpperFact = abs(ActualBeltLength - BL(1,2));

                            %Check if termination conditions met
                            if DifUpperFact <= tollerence
                                %termination conditions met

                                %terminate loop
                                Stop = 1;
                                Y2 = pointer(1,2);

                                %Value found
                                ValFound = 1;

                            else
                                %termination conditions not met

                                %pointer = [ Y2, (Y2 - 0.5*Hold) ];
                                Y2 = pointer(1,2);
                                Hold = Hold/2;
                            end

                    %Y2 in unknown fation
                    %****************************************************

                        else
                            %Unknown state = write to console
                            display('Y2 value in neither region');

                        end
            end
    end

end
```

H Appendix - Grade 355 Structural Tube Specifications

Table H.1: Grade 355 Structural Tube Chemical Specifications[1]

Specification	**Value**
Carbon	0.14 %
Manganese	0.15 %
Phosphorous	0.035 %
Sulphur	0.03 %
Silicon	$0.15 - 0.25$ %
Niobium	0.05 %
Titanium	0.05 %

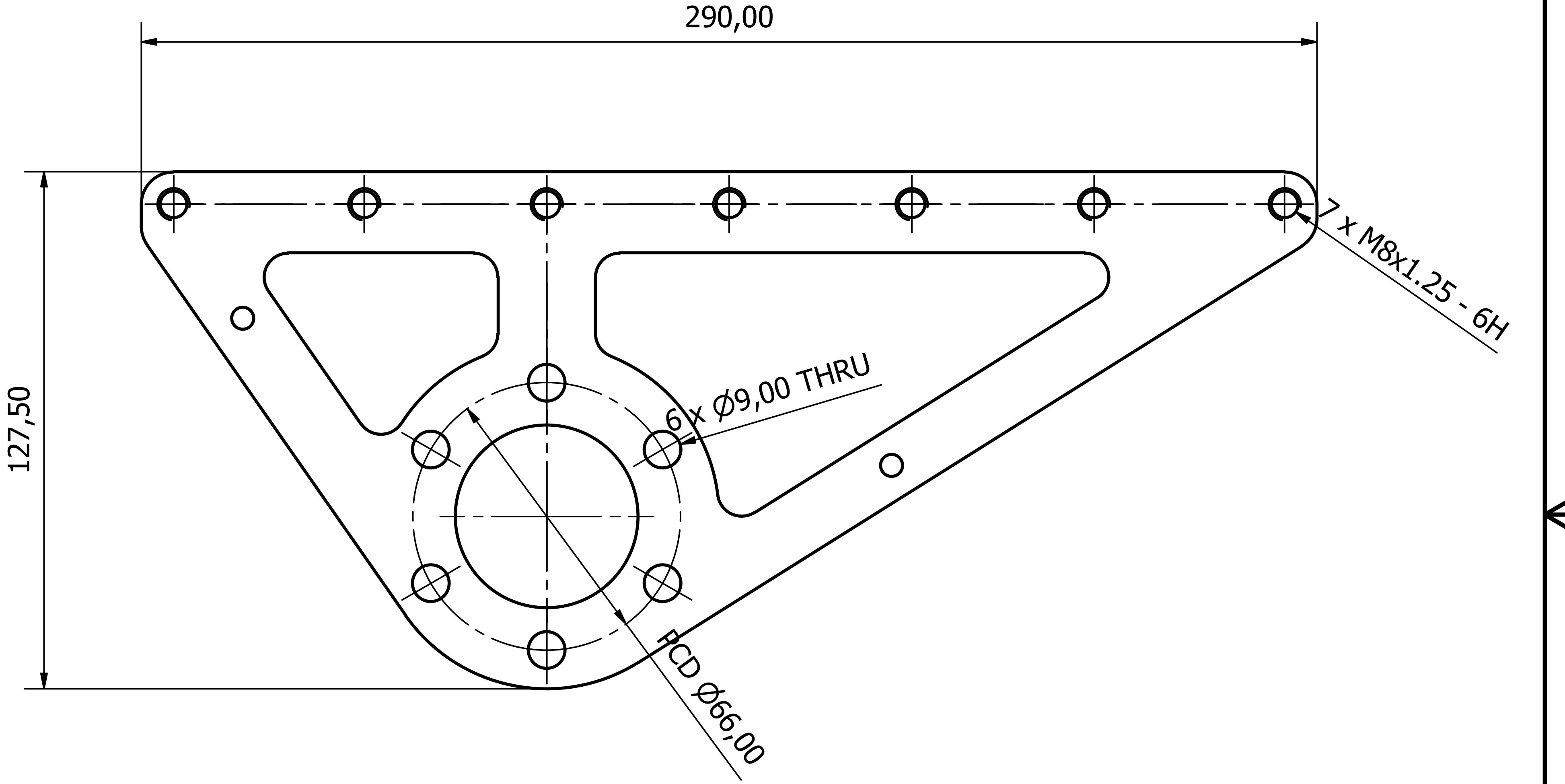
Table H.2: Grade 355 Structural Tube Mechanical Specifications[2]

Specification	**Value**
Yield Strength	$355 - 475\ MPa$
Ultimate Tensile Strength	$450 - 550\ MPa$
Minimum Elongation A5	22 %
Impact Test	$27\ J\ at\ 0°C$

[1] Chemical composition dictated by ASTPM[116]

[2] Mechanical specification dictated by ASTPM[116], testing methodology can be found here: ASTPM[116]

I Appendix - Working Drawings

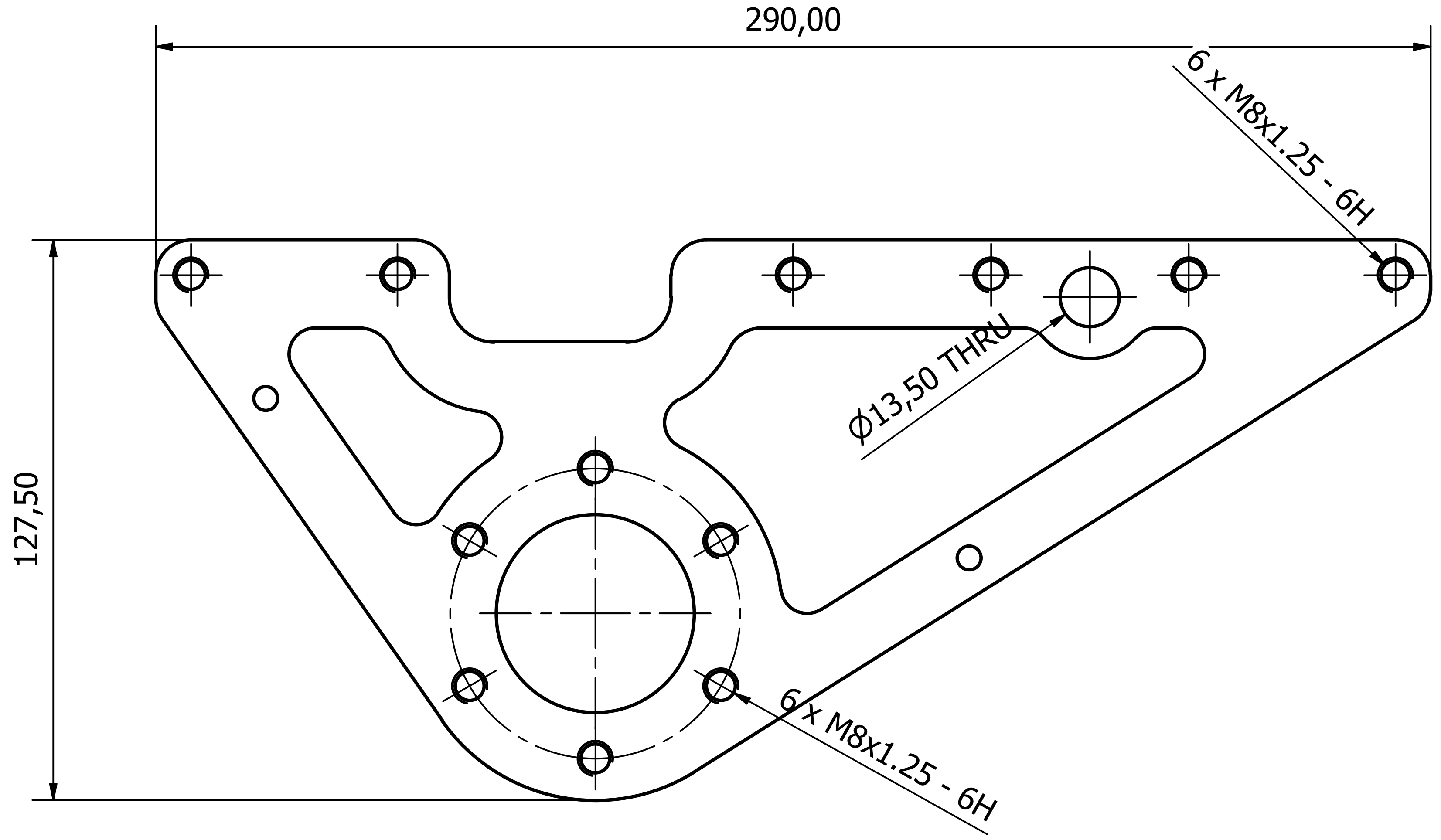


Sheet Thickness: 4,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Axle Support Plate Type 1		
Swerve Drive AGV	Edition	Sheet 1 / 207



Sheet Thickness: 4,00 mm

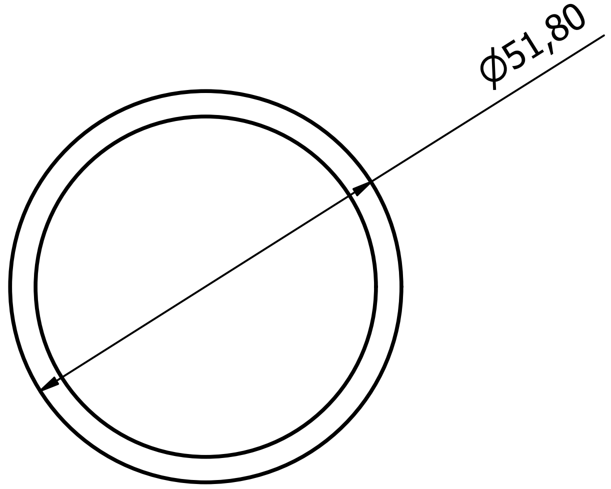
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Alex Support Plate Type 2

Swerve Drive AGV

Edition	Sheet
	2 / 207



Sheet Thickness: 4,00 mm

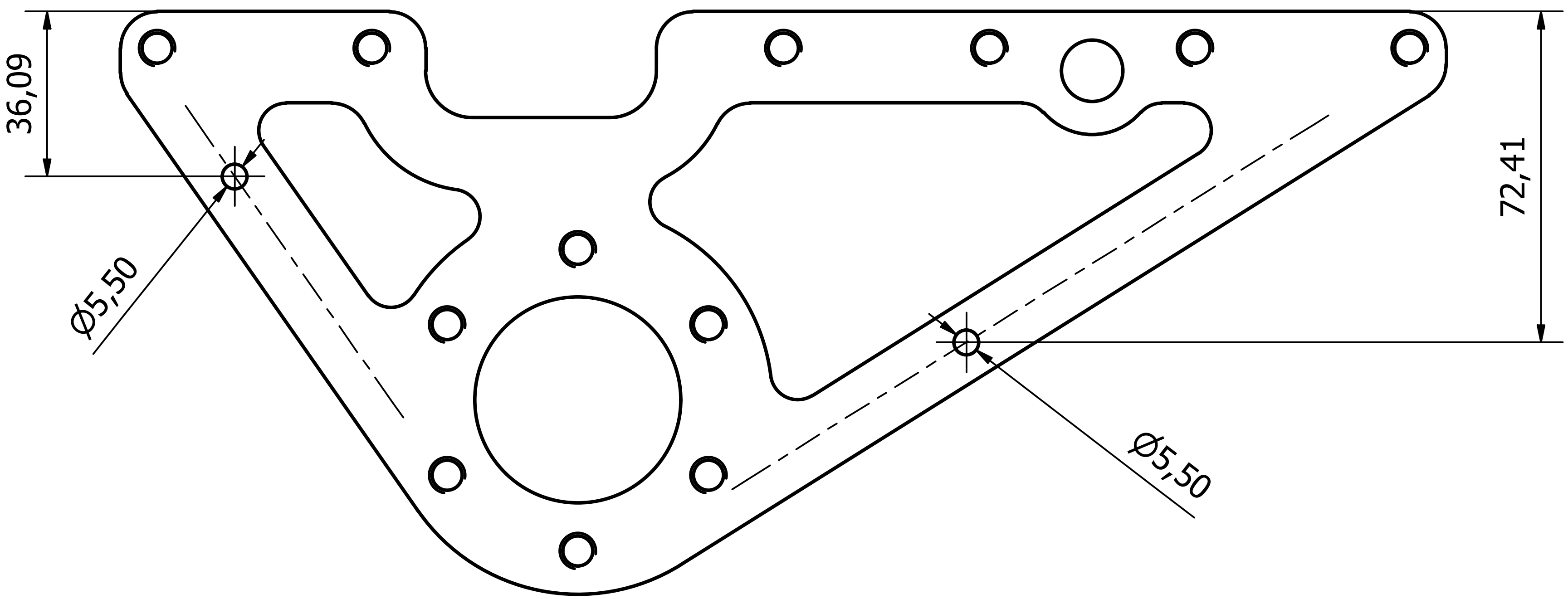
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Thrust Washer Seat Ring

Swerve Drive AGV

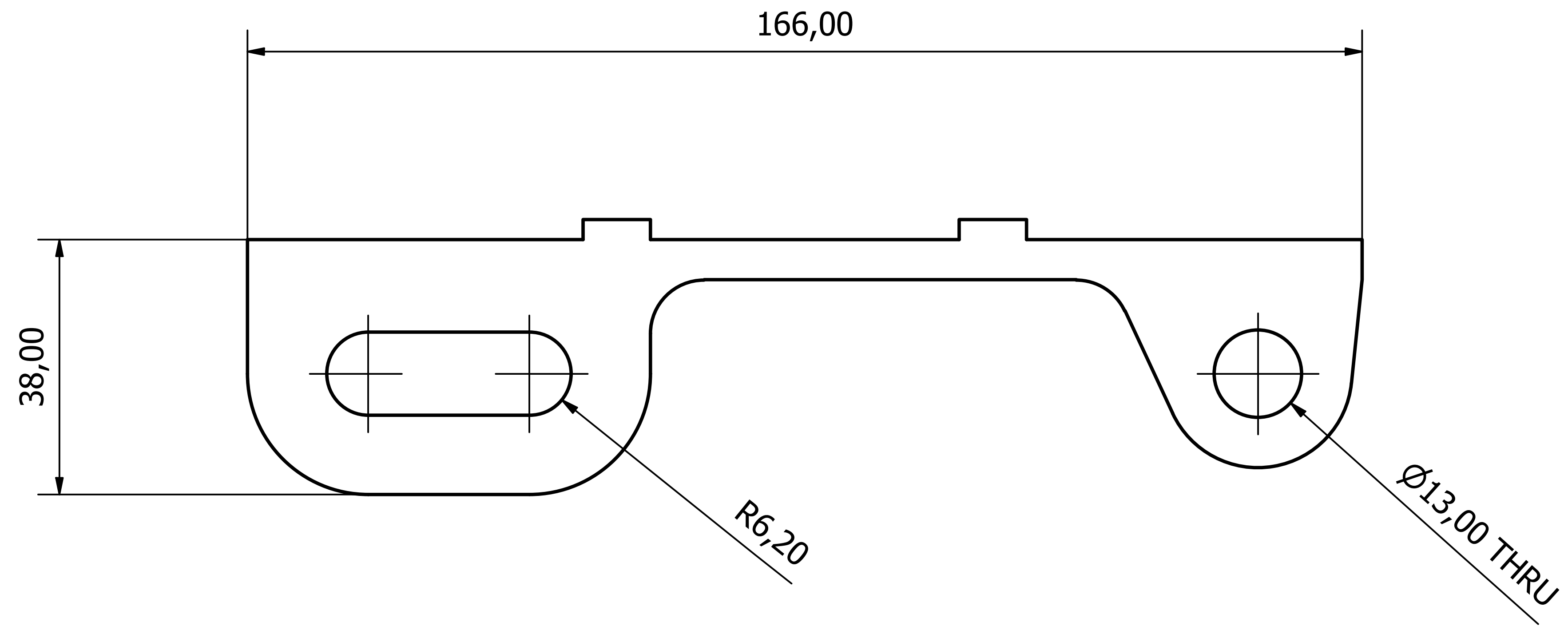
Edition	Sheet
	3 / 207



Drill diagram for missing holes. These holes are responsible for attaching the spacer beams

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY	Alex Support Plate Type 2		
	Swerve Drive AGV	Edition	Sheet 4 / 207



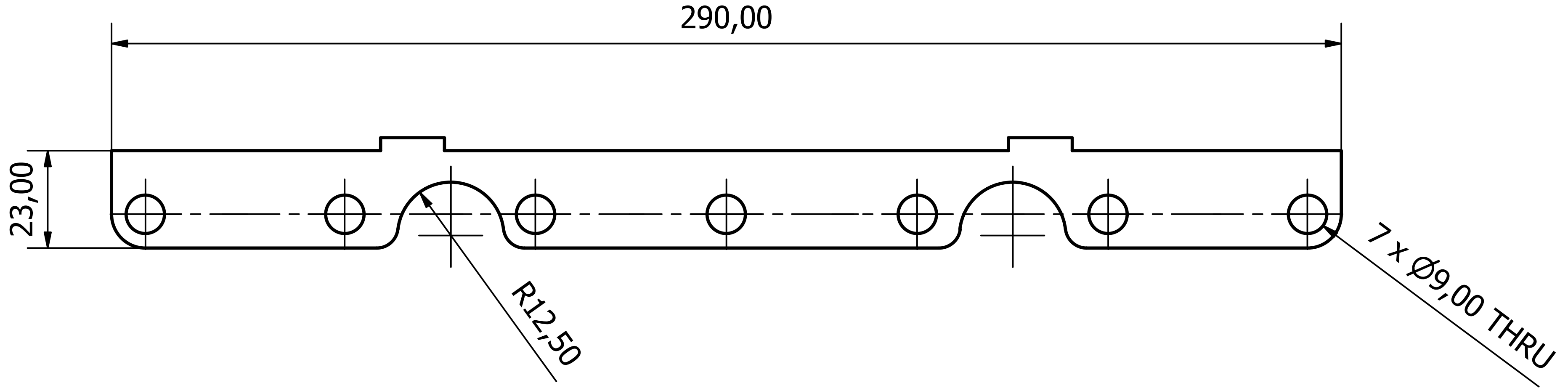
Sheet Thickness: 4,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Idler Pulley Independent Plate

	Edition	Sheet
Swerve Drive AGV		5 / 207



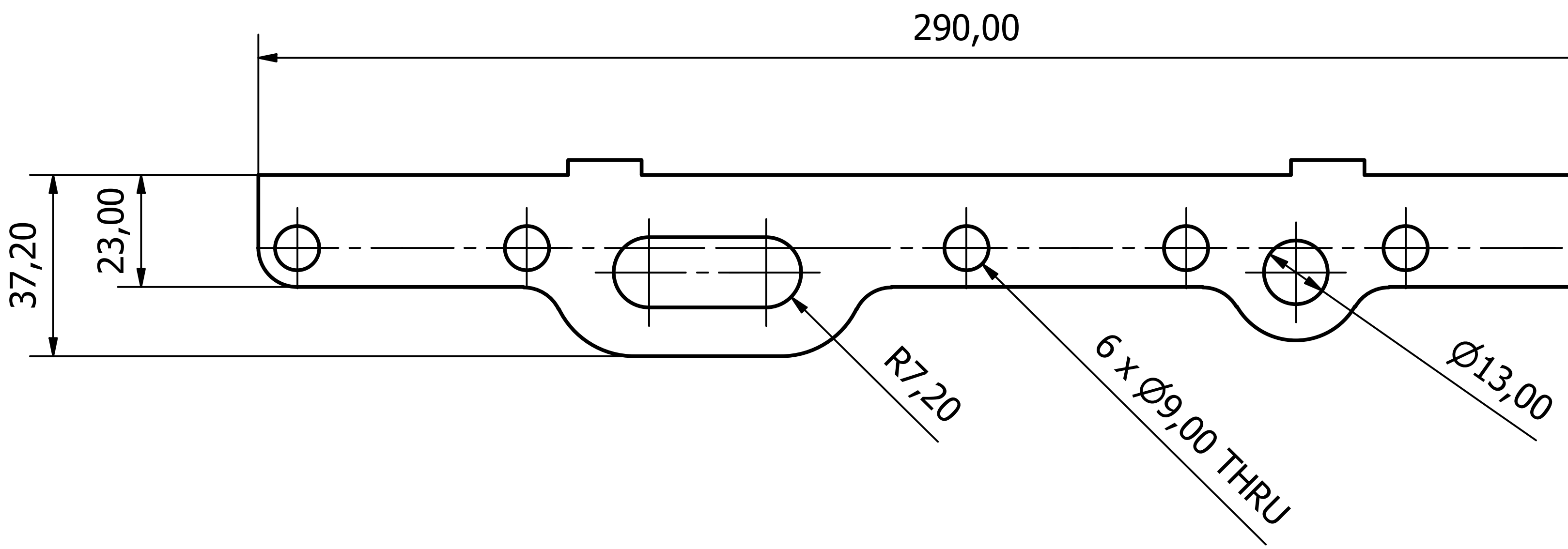
Sheet Thickness: 5,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

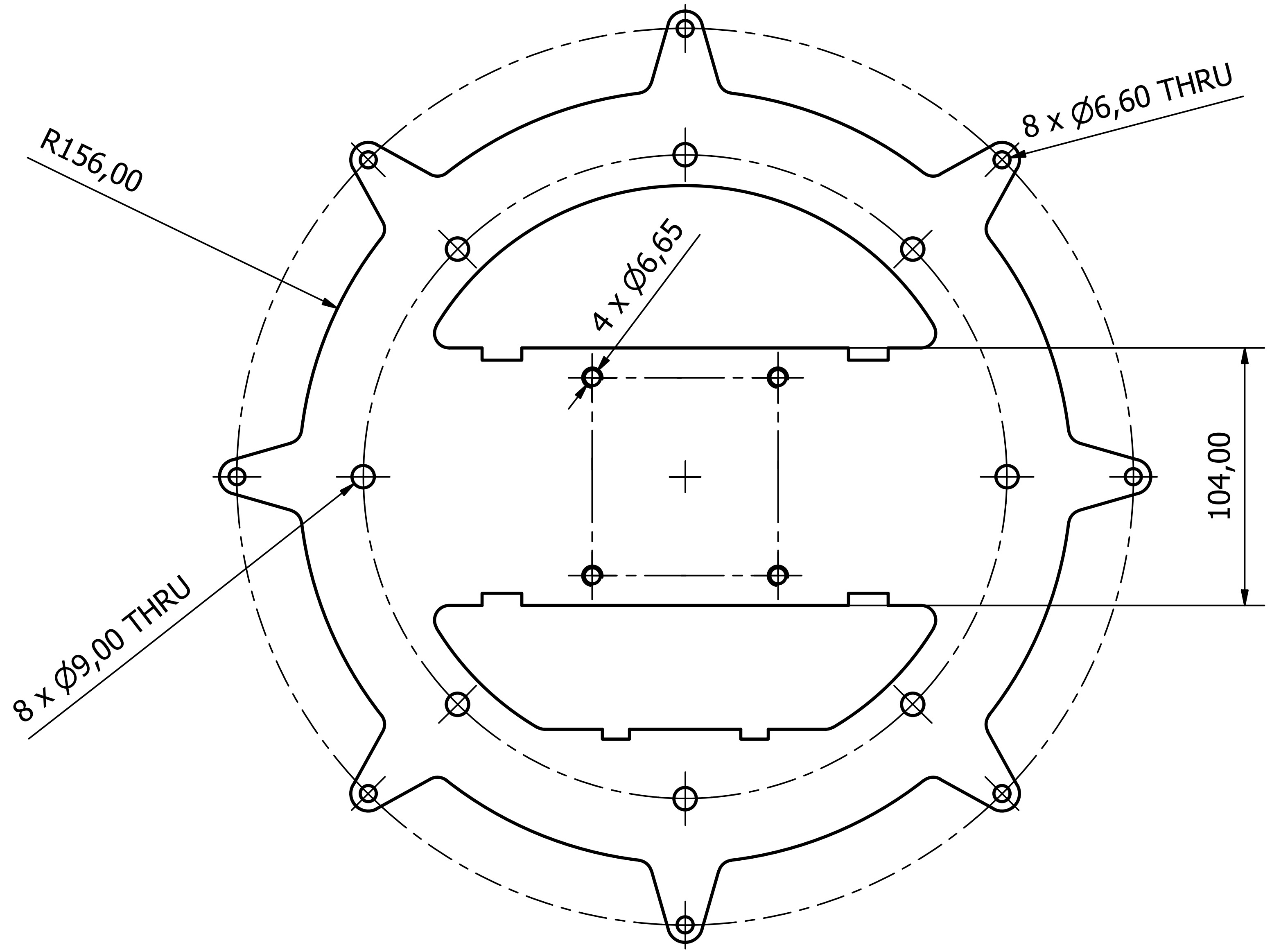
Interface Flange Brace Type 1

Swerve Drive AGV	Edition	Sheet
		6 / 207



Sheet Thickness: 5,00 mm

Designed by AlexM	Quantity 2	Manufactured by External	Material Steel, Mild	Date 07/03/2019
NELSON MANDELA UNIVERSITY	Interface Flange Brace Type 2			
	Swerve Drive AGV	Edition	Sheet 7 / 207	



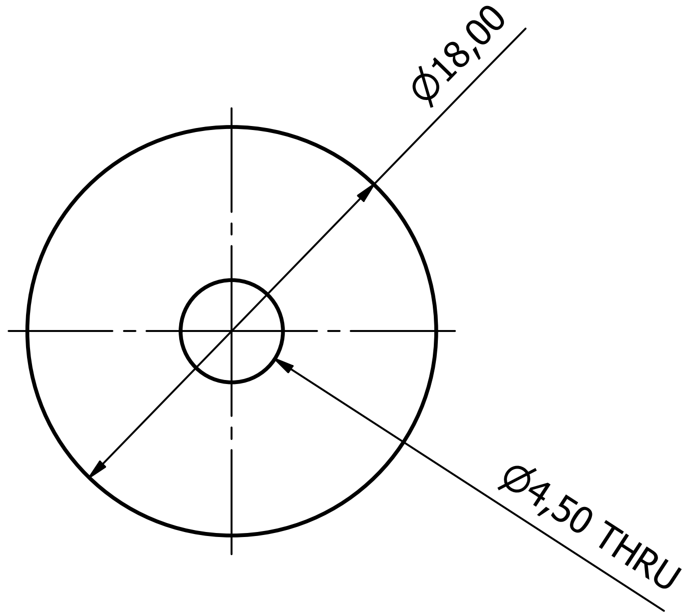
Sheet Thickness: 3,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Swivel Side Interface Flange

Swerve Drive AGV	Edition	Sheet
		8 / 207



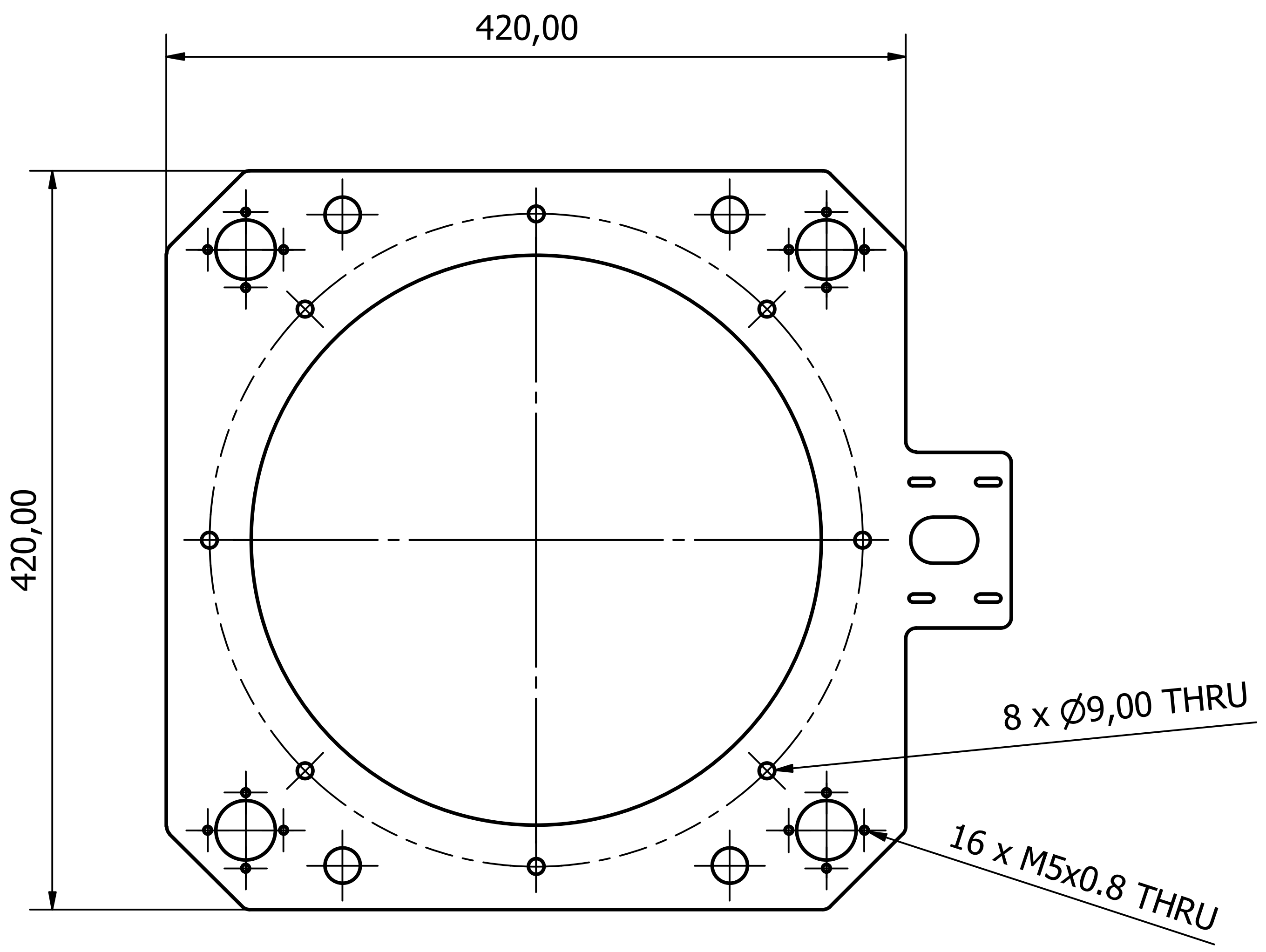
Sheet Thickness: 2,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Tensioning Rod End Stop

	Edition	Sheet
Swerve Drive AGV		9 / 207



Sheet Thickness: 4,00 mm

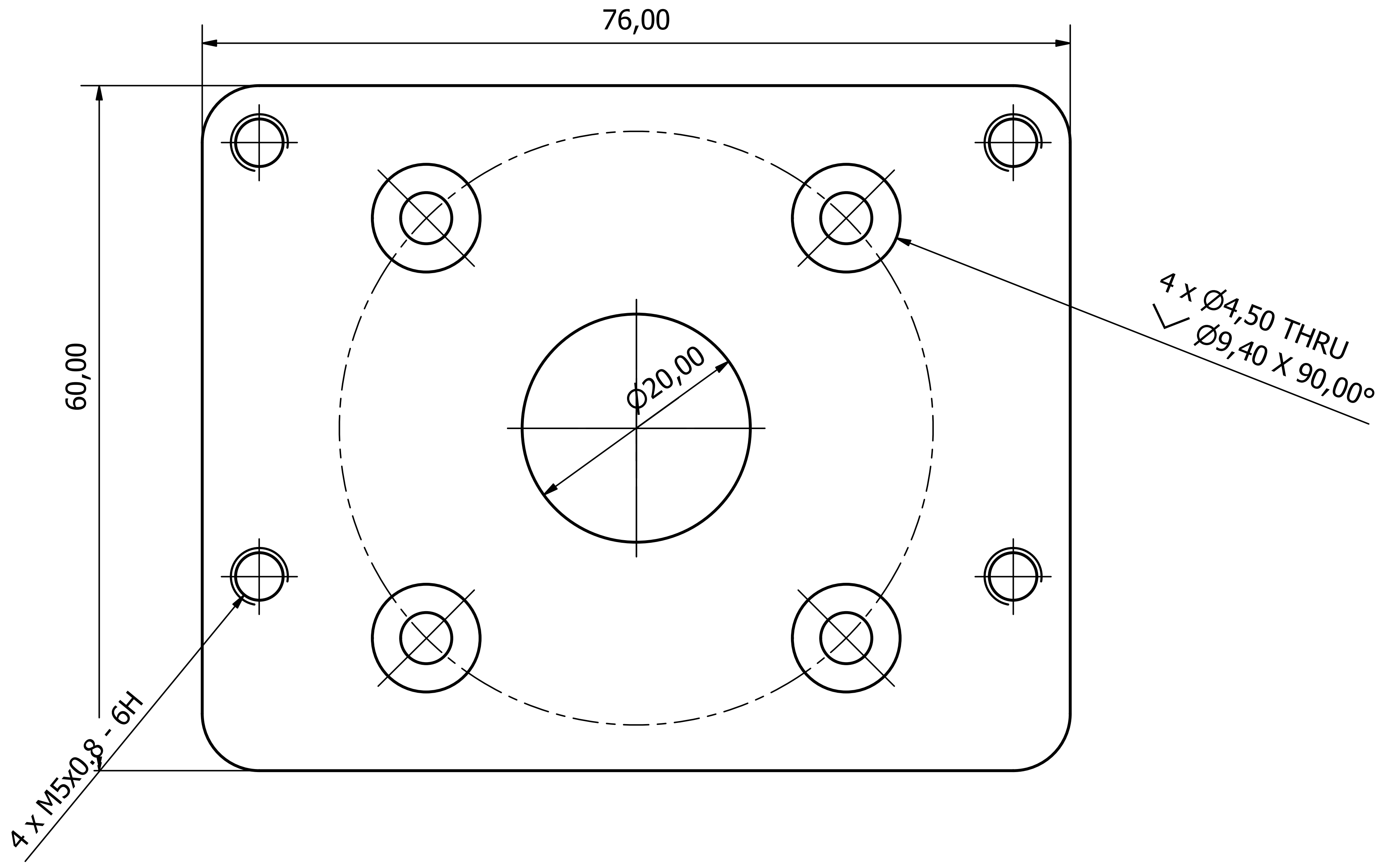
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Slew Ring Stationary Plate

Swerve Drive AGV

Edition	Sheet
	10 / 207



Sheet Thickness: 4,00 mm

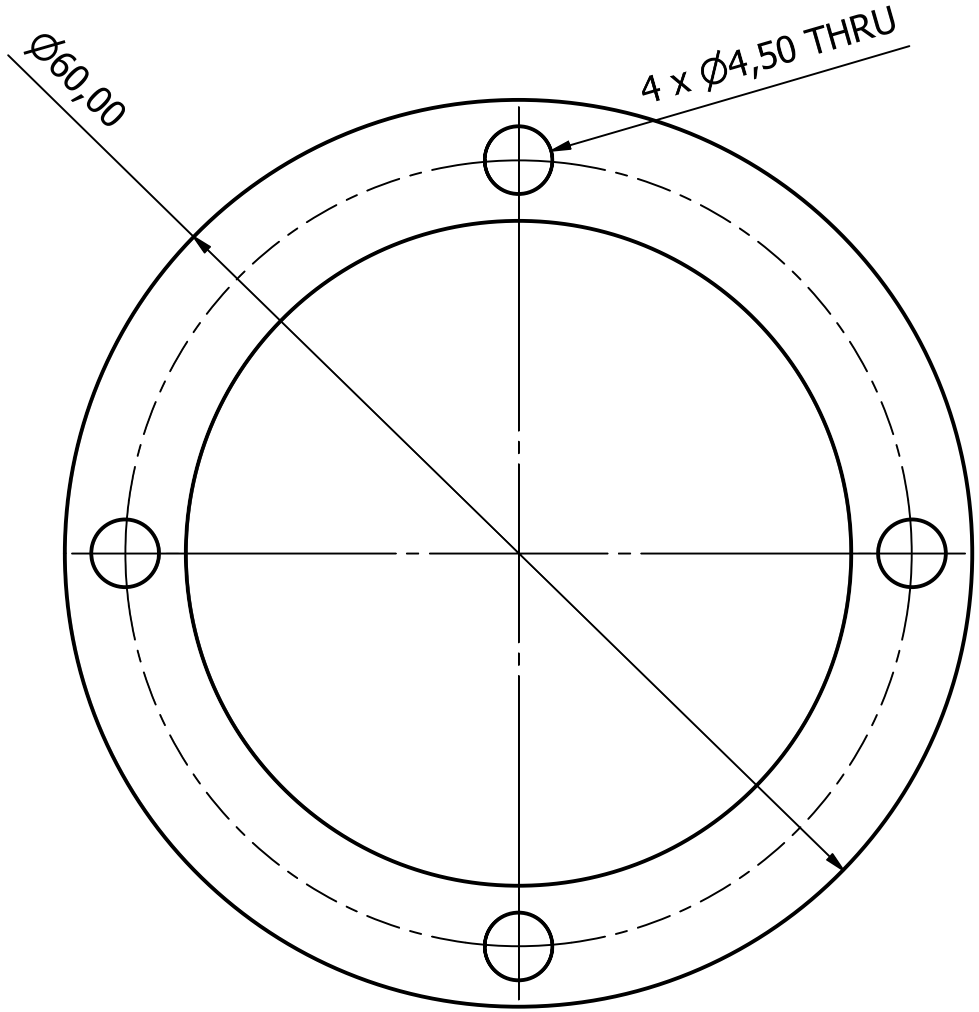
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Turning Motor Spacer

Swerve Drive AGV

Edition	Sheet
	11 / 207

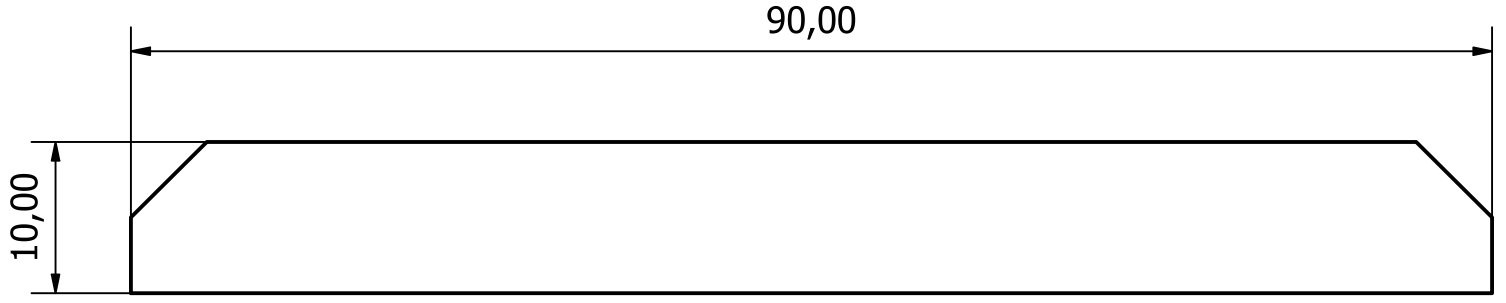


Sheet Thickness: 4,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Turning Motor Ring		
Swerve Drive AGV	Edition	Sheet 12 / 207



Sheet Thickness: 4,00 mm

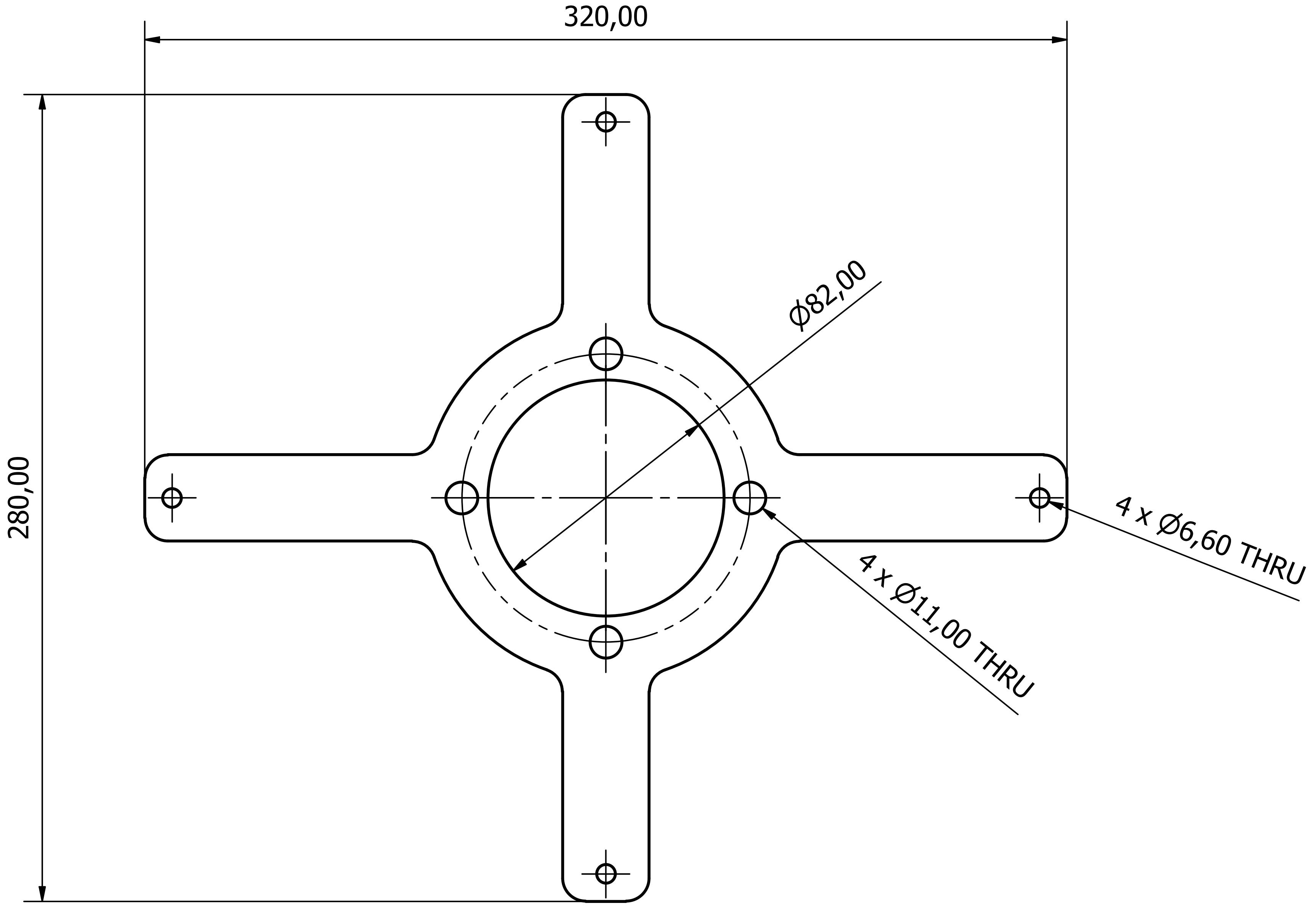
Designed by	Quantity	Manufactured by	Material	Date
AlexM	4	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Turning Motor Webs

Swerve Drive AGV

Edition	Sheet
	13 / 207



Sheet Thickness: 3,00 mm

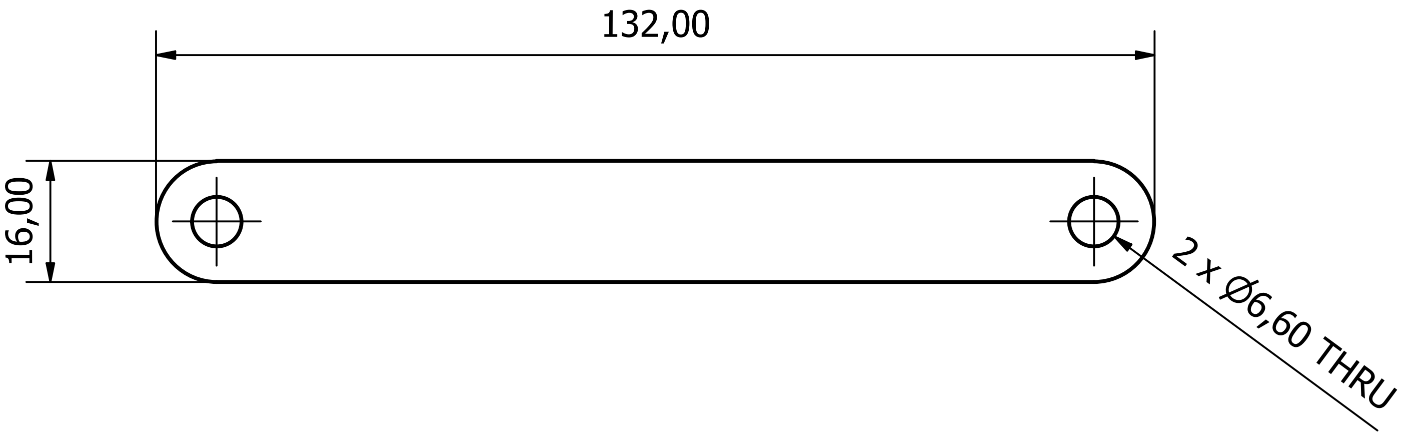
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Compensator Mount Plate

Swerve Drive AGV

Edition	Sheet
	14 / 207



Sheet Thickness: 2,00 mm

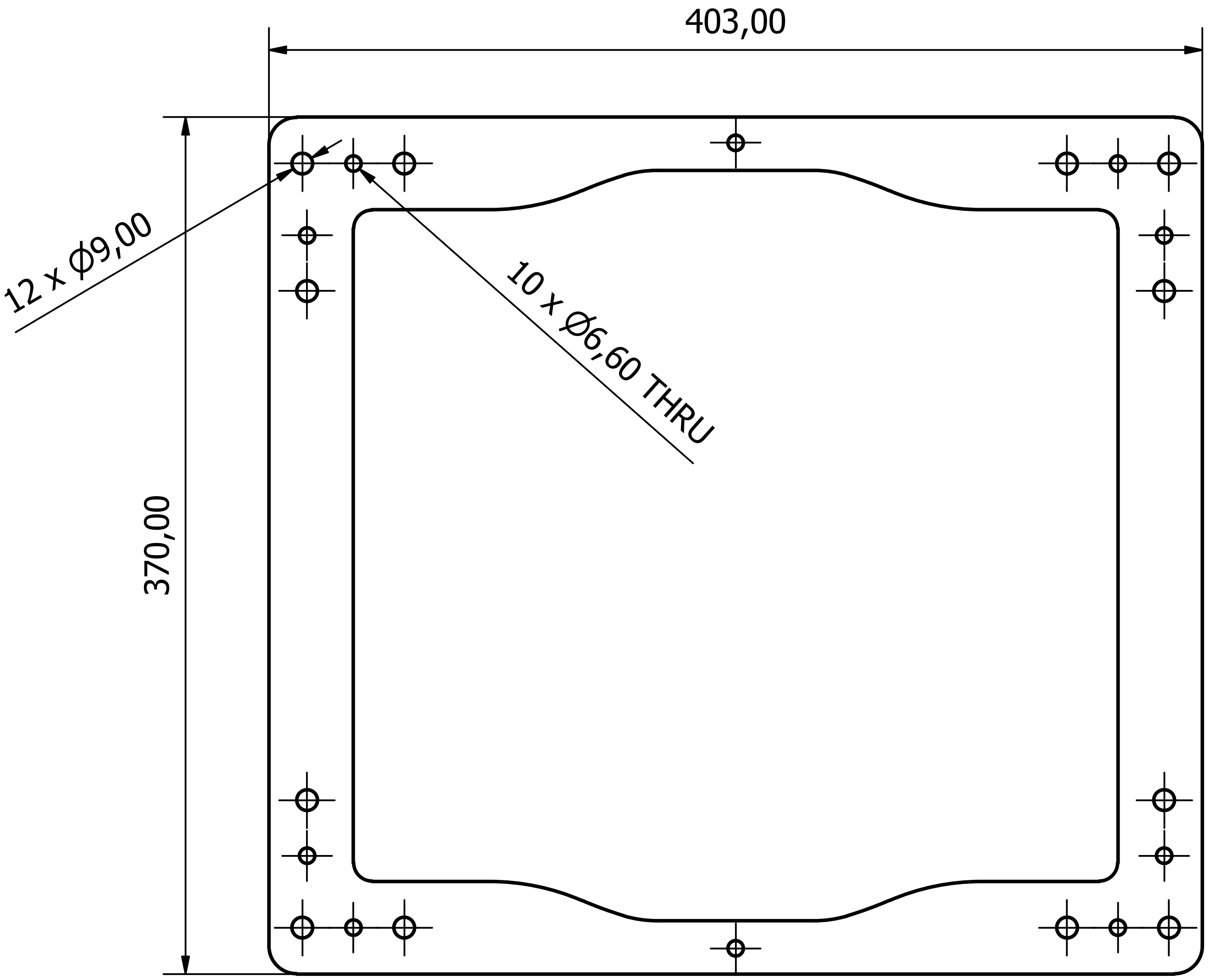
Designed by	Quantity	Manufactured by	Material	Date
AlexM	4	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Long Plate

Swerve Drive AGV

Edition	Sheet
	15 / 207



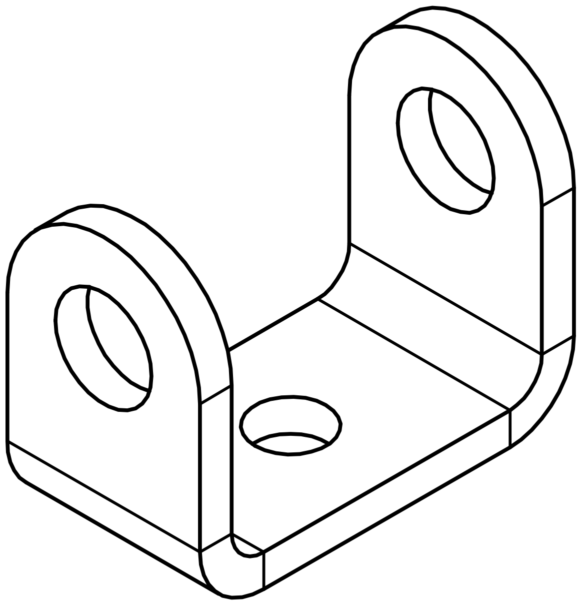
Sheet Thickness: 3,00 mm

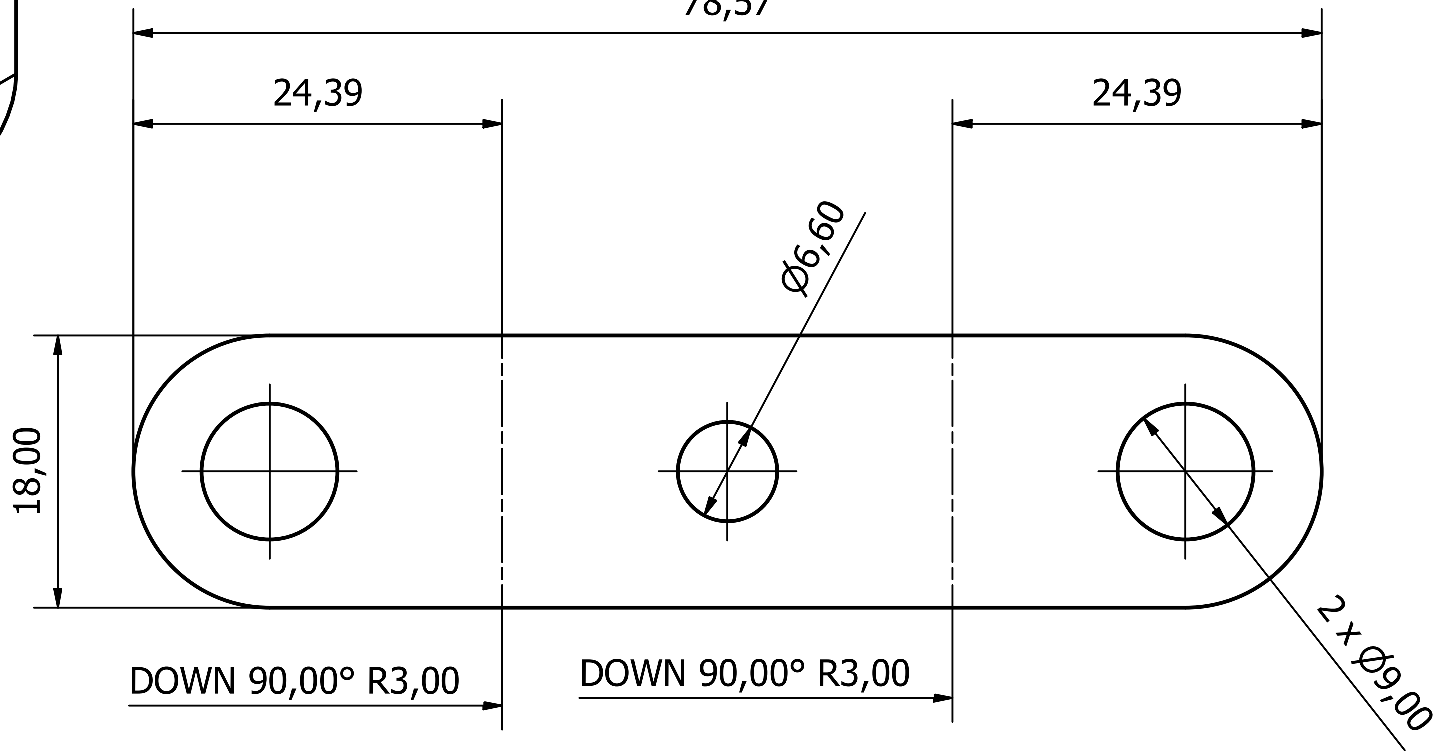
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Lower Structure Brace

	Edition	Sheet
Swerve Drive AGV		16 / 207





Sheet Thickness: 3,00 mm

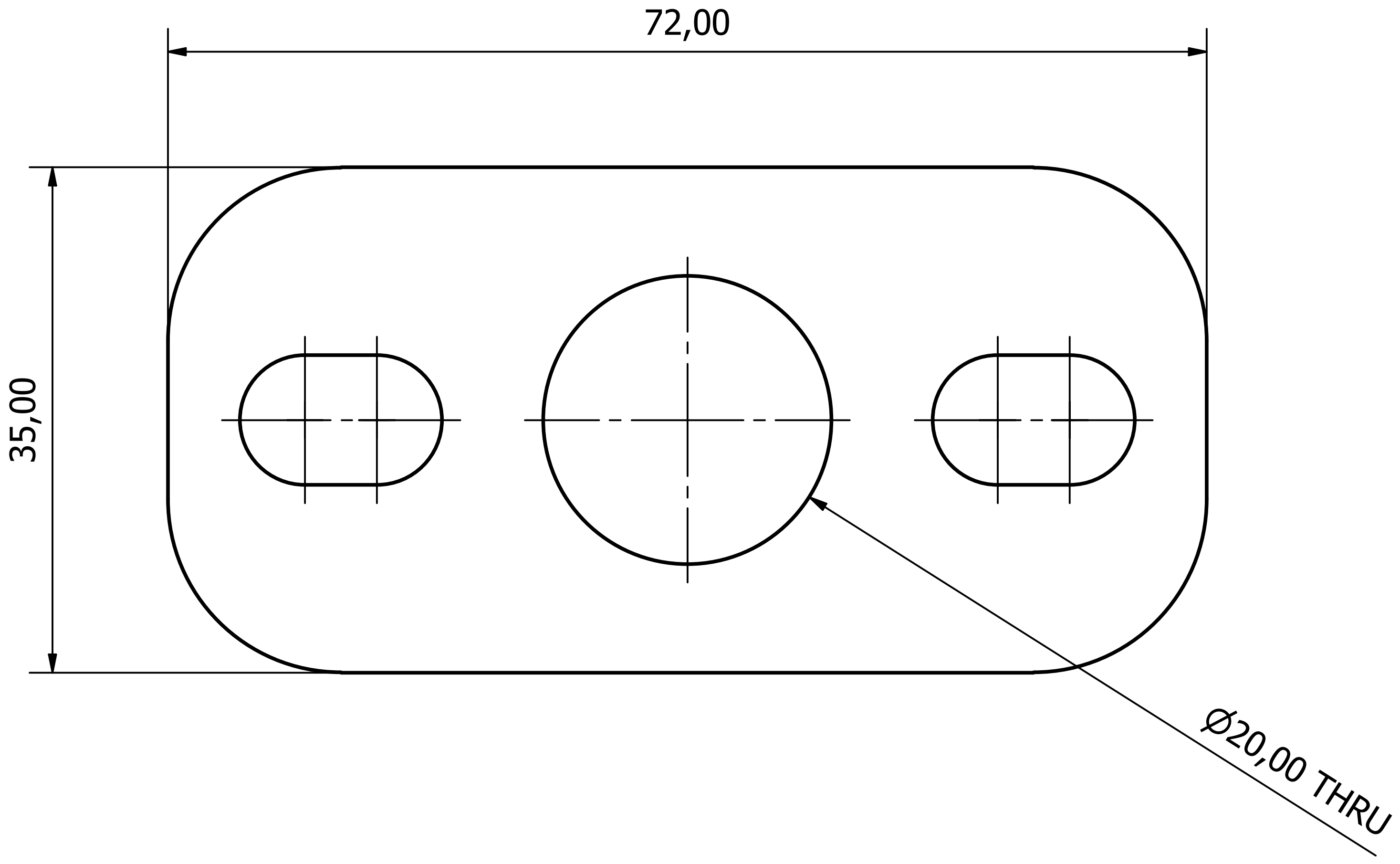
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Shackle Buckle

Swerve Drive AGV

Edition	Sheet
	17 / 207



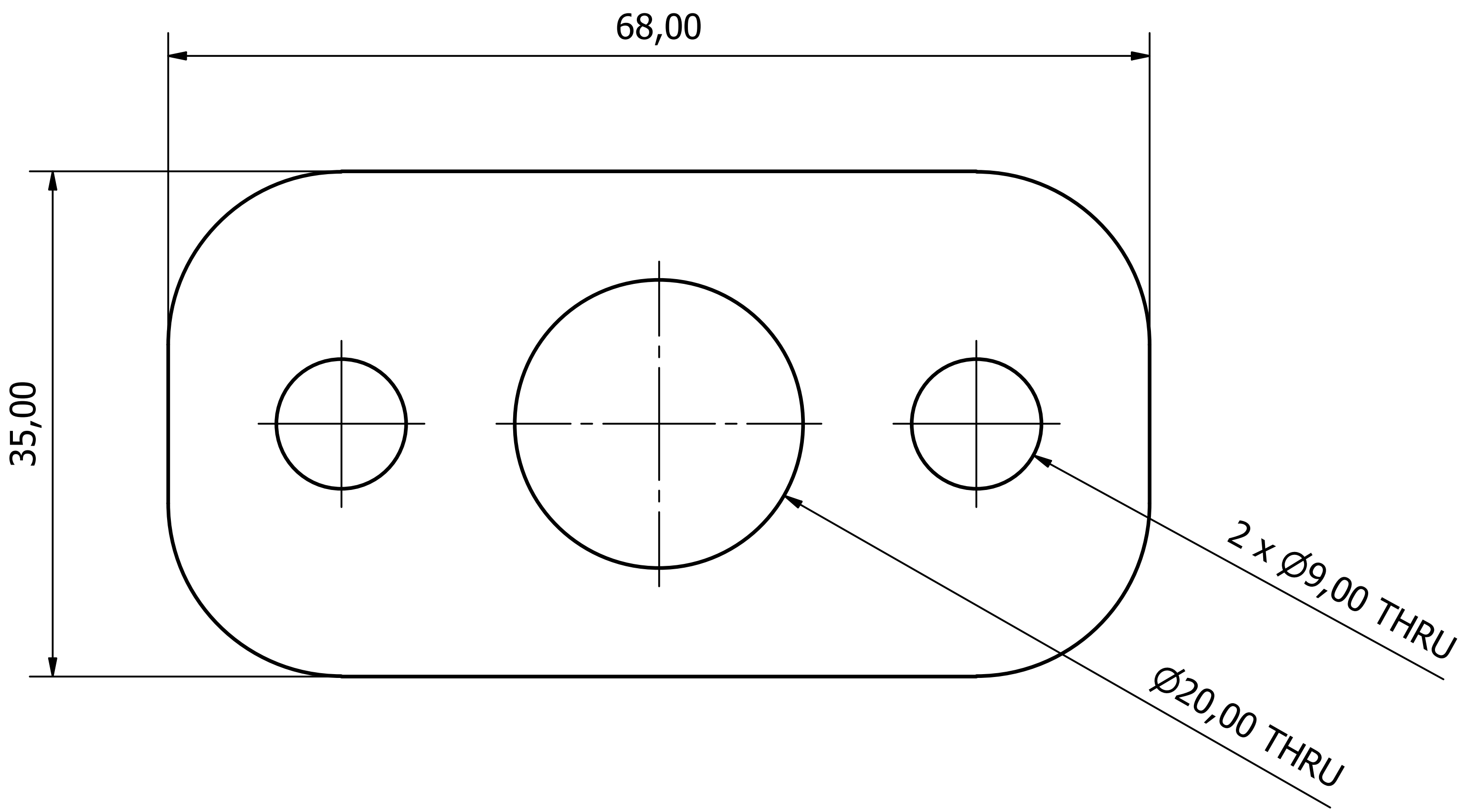
Sheet Thickness: 10,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	8	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Sliding Rod Mount Type 1

	Edition	Sheet
Swerve Drive AGV		18 / 207

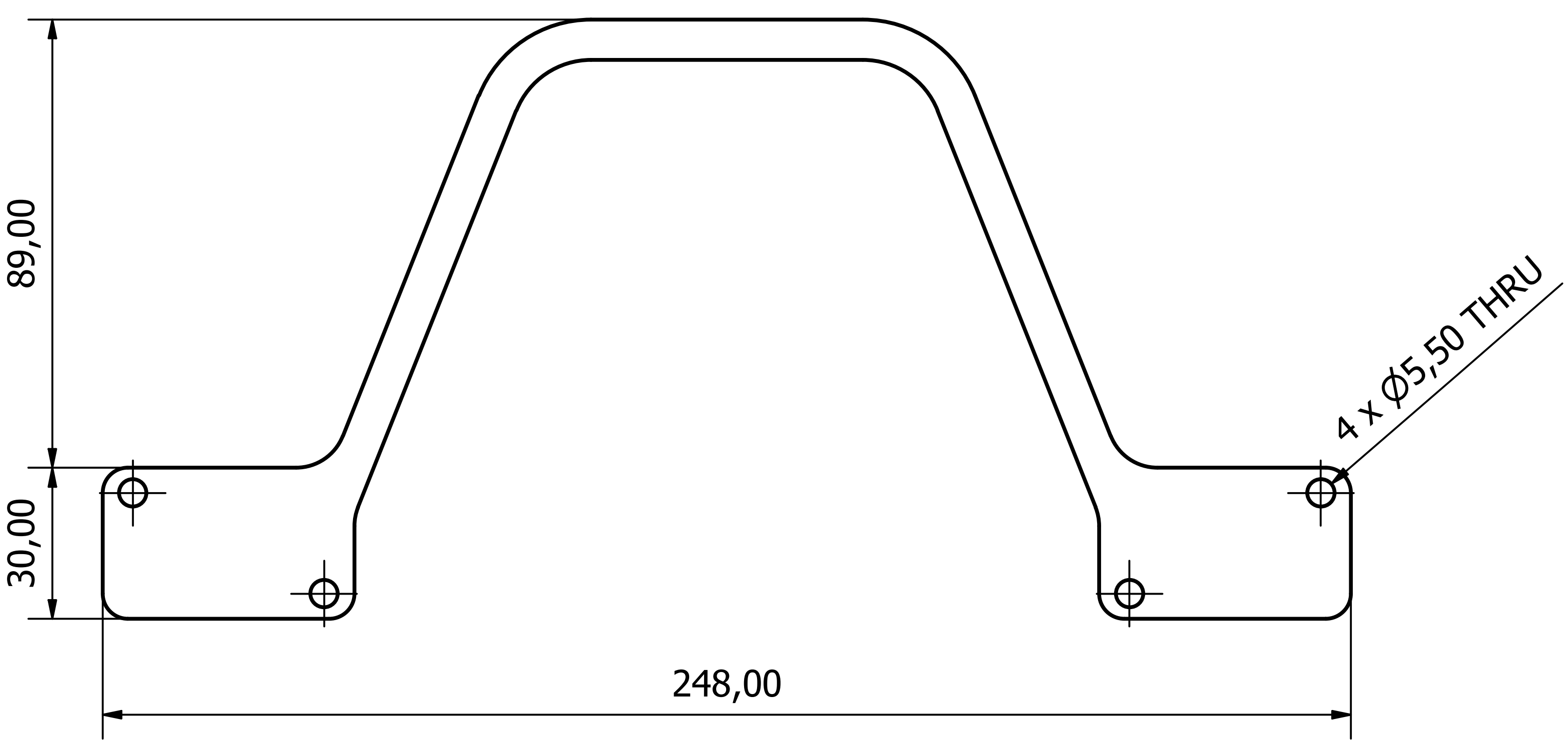


Sheet Thickness: 10,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	8	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Sliding Rod Mount Type 2		
Swerve Drive AGV	Edition	Sheet 19 / 207



Sheet Thickness: 5,00 mm

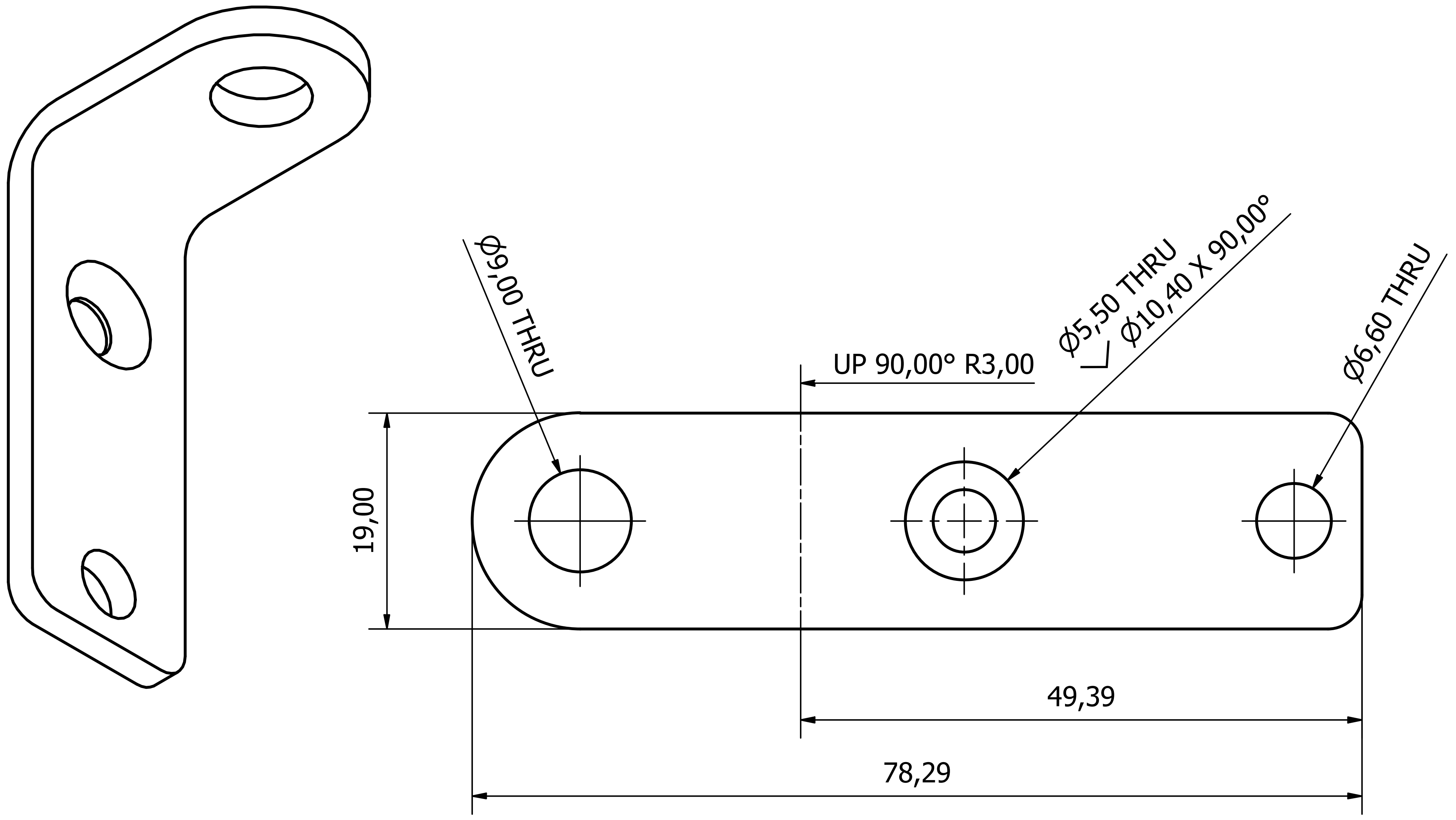
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Steering Motor Protection Brace Sheet

Swerve Drive AGV

Edition	Sheet
	20 / 207



Sheet Thickness: 3,00 mm

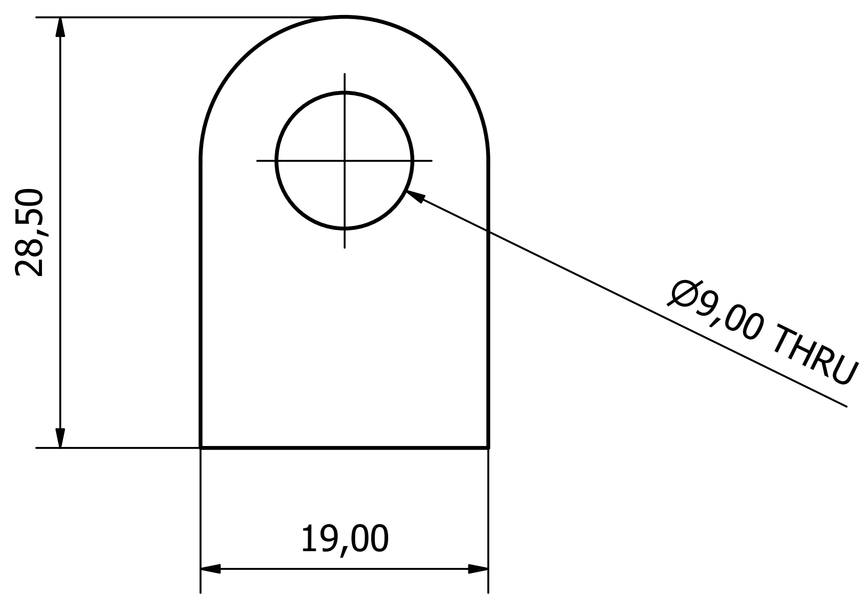
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Tensioner Butress Plate

Swerve Drive AGV

Edition	Sheet
	21 / 207



Sheet Thickness: 3,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

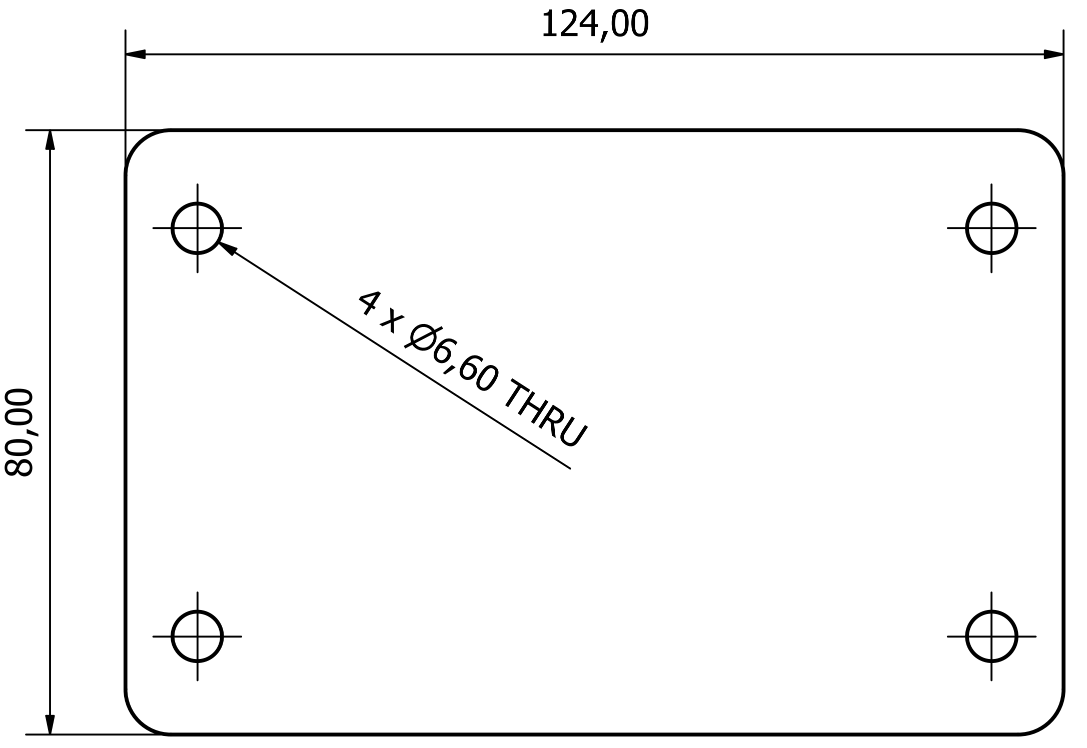
NELSON MANDELA UNIVERSITY

Tensioner Butress

Swerve Drive AGV

Edition

Sheet 22 / 207

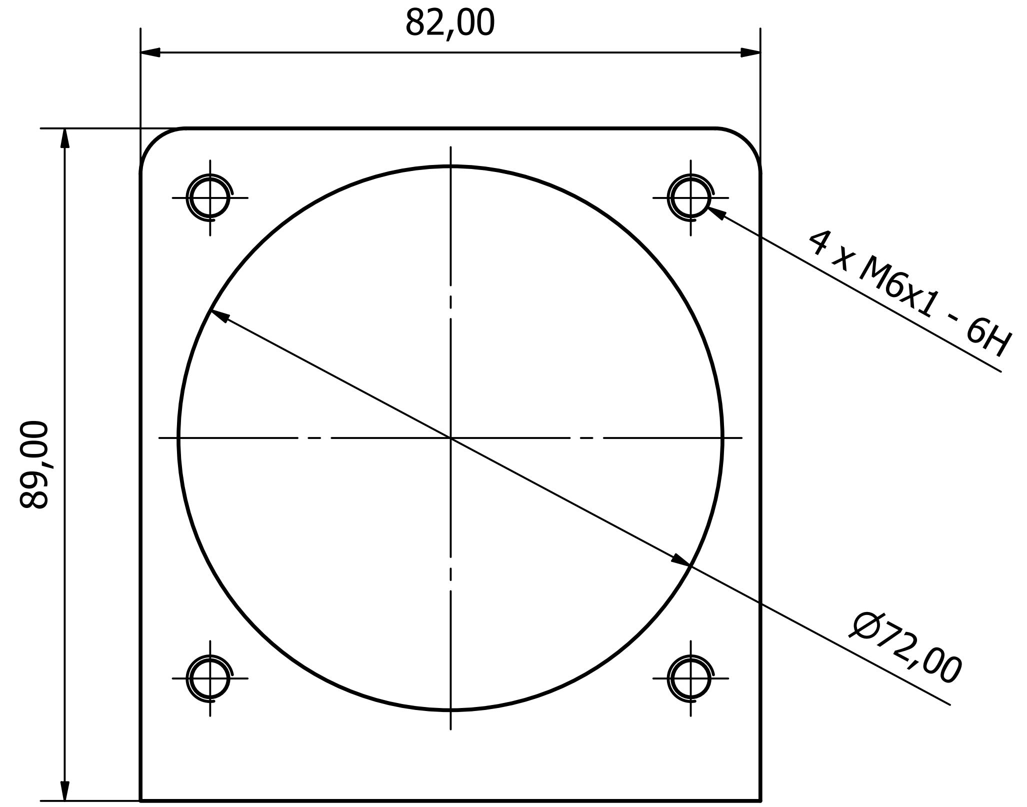


Sheet Thickness: 6,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Traction Motor Mount Base		
Swerve Drive AGV	Edition	Sheet 23 / 207

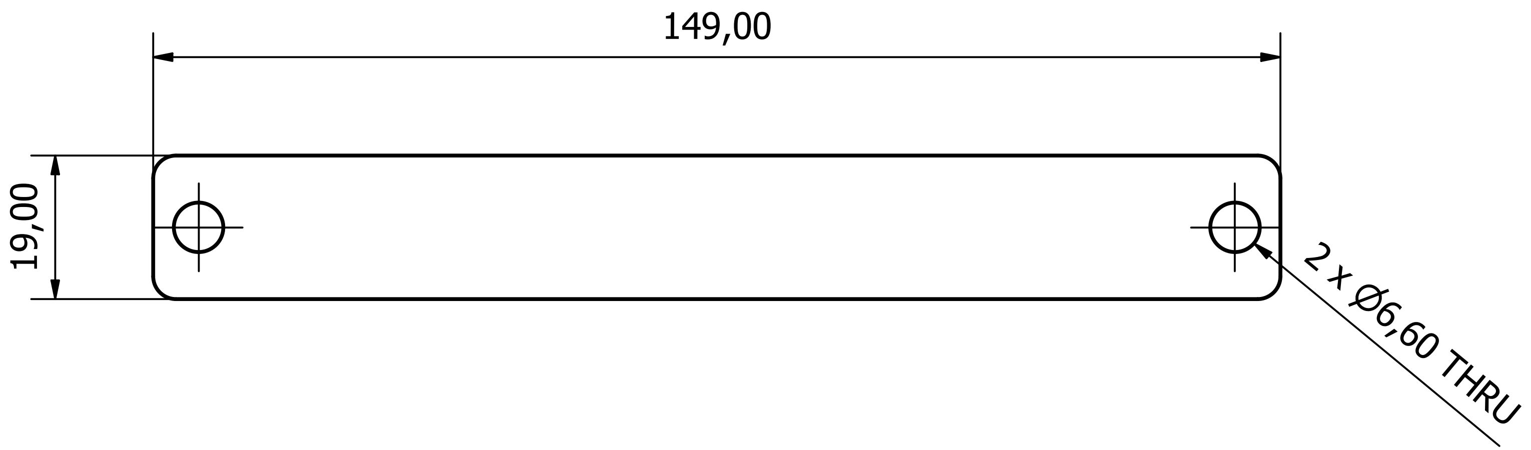


Sheet Thickness: 6,00 mm

Designed by AlexM	Quantity 2	Manufactured by External	Material Steel, Mild	Date 07/03/2019

Traction Motor Mount Flange

Swerve Drive AGV	Edition	Sheet 24 / 207



Sheet Thickness: 3,00 mm

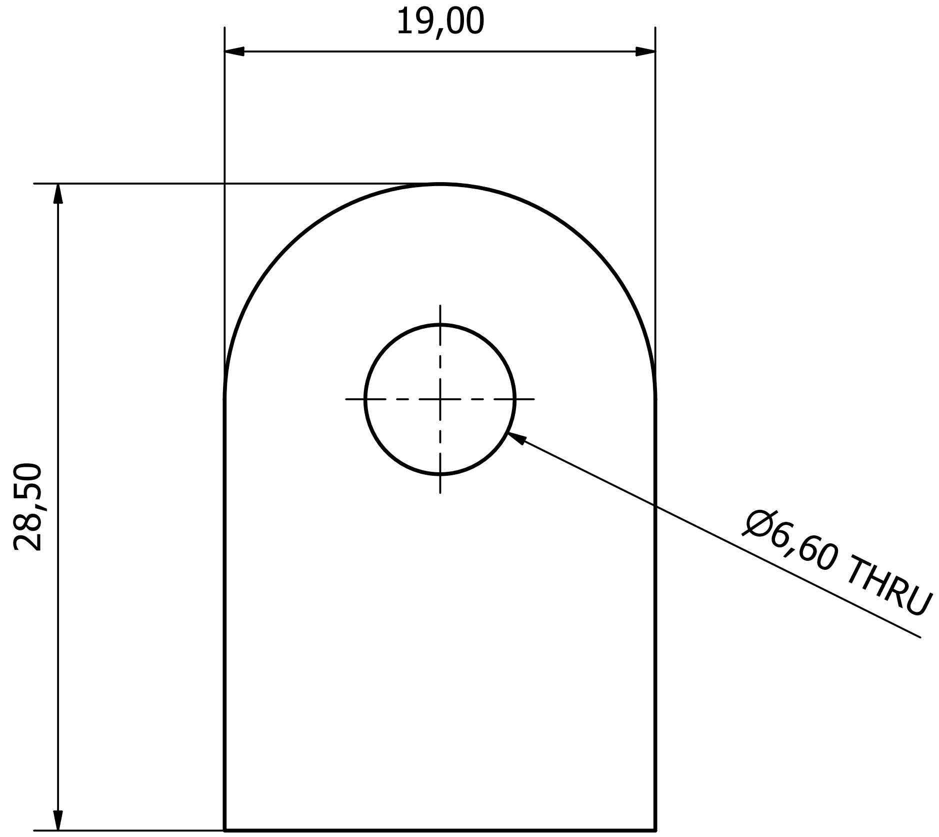
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Upper Idler Pulley Butress Plate

Swerve Drive AGV

Edition	Sheet
	25 / 207



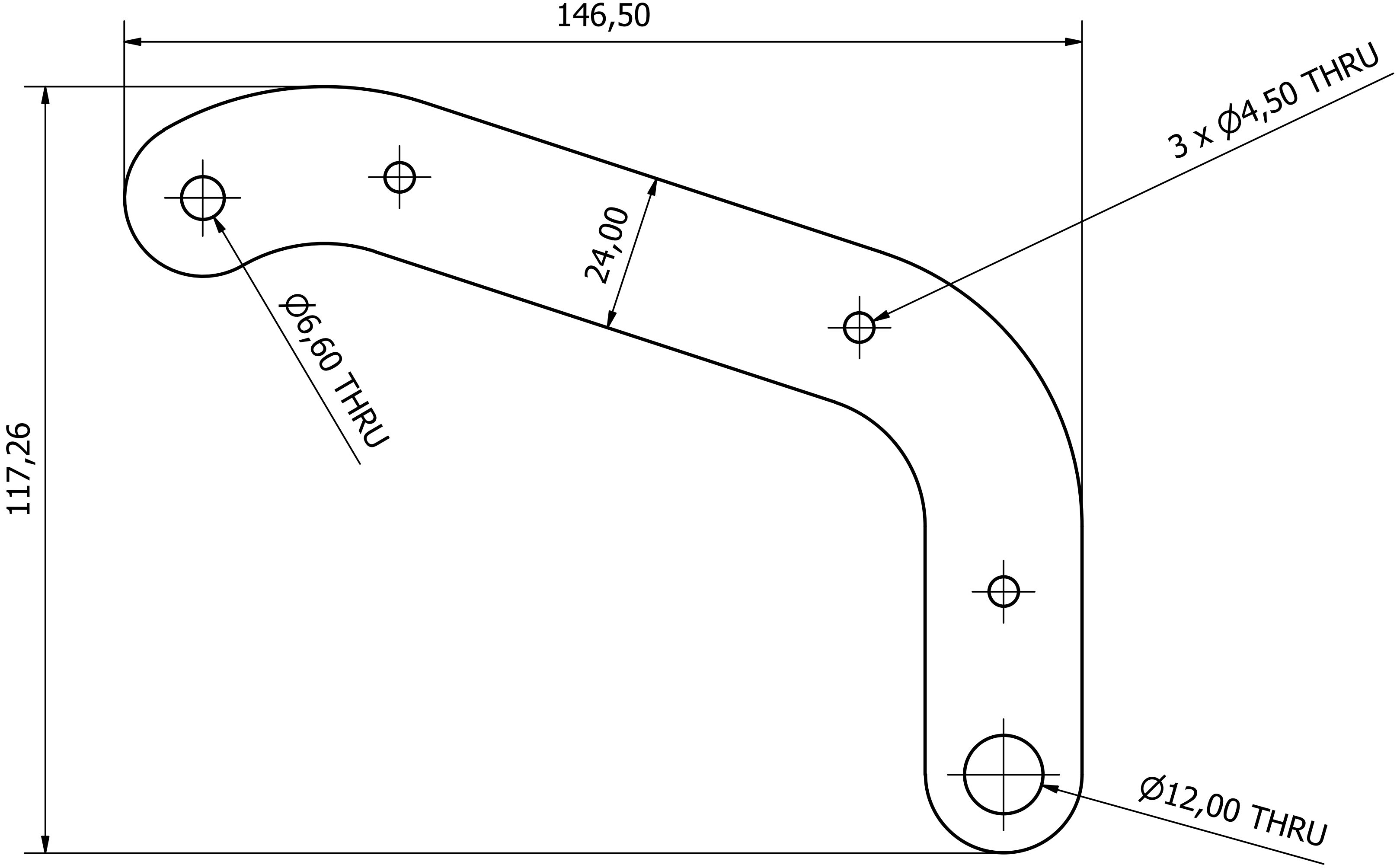
Sheet Thickness: 3,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	4	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Upper Idler Pulley Butress

	Edition	Sheet
Swerve Drive AGV		26 / 207



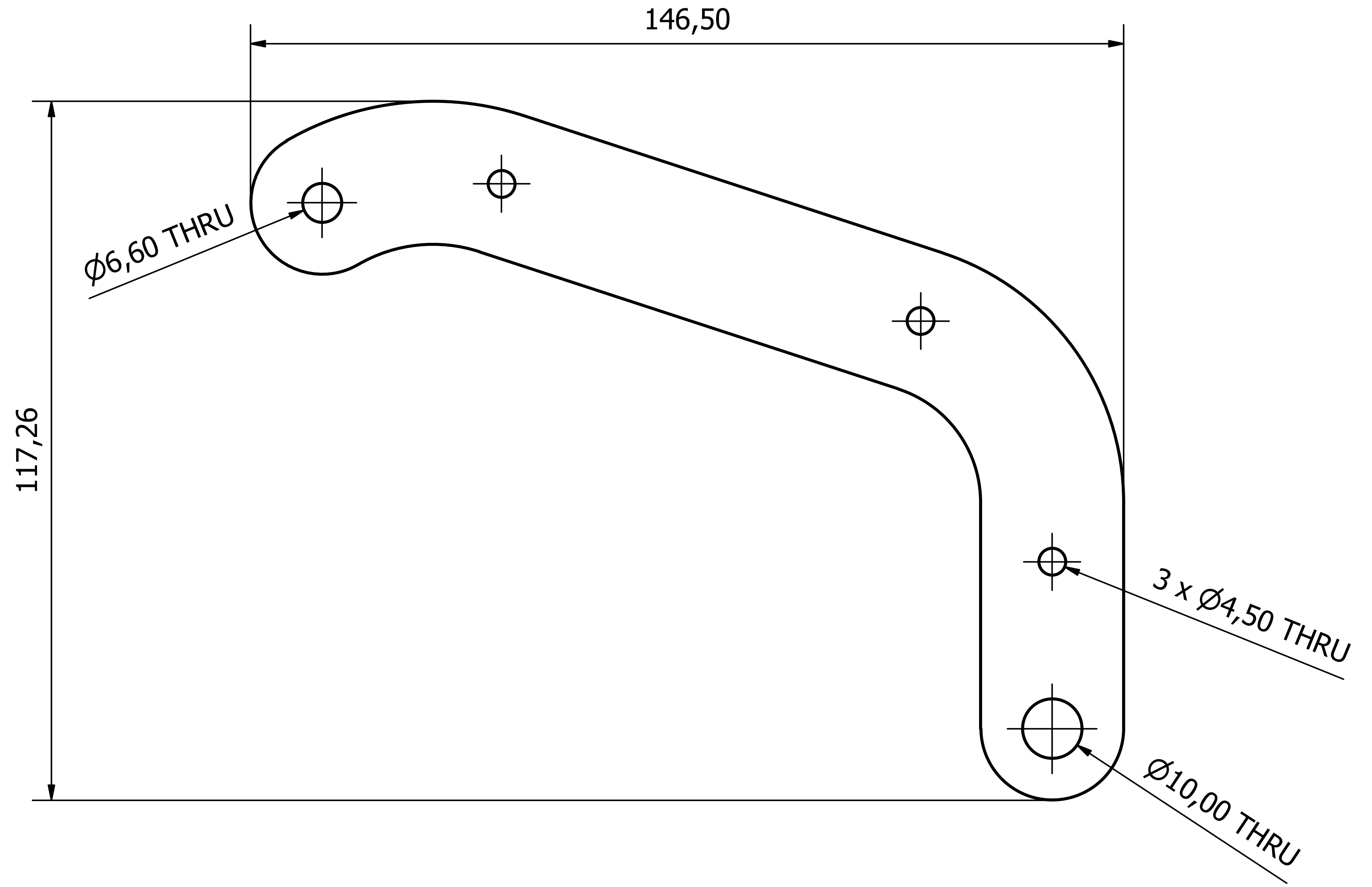
Sheet Thickness: 3,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Upper Idler Pulley Cantilever Plate 1

	Edition	Sheet
Swerve Drive AGV		27 / 207



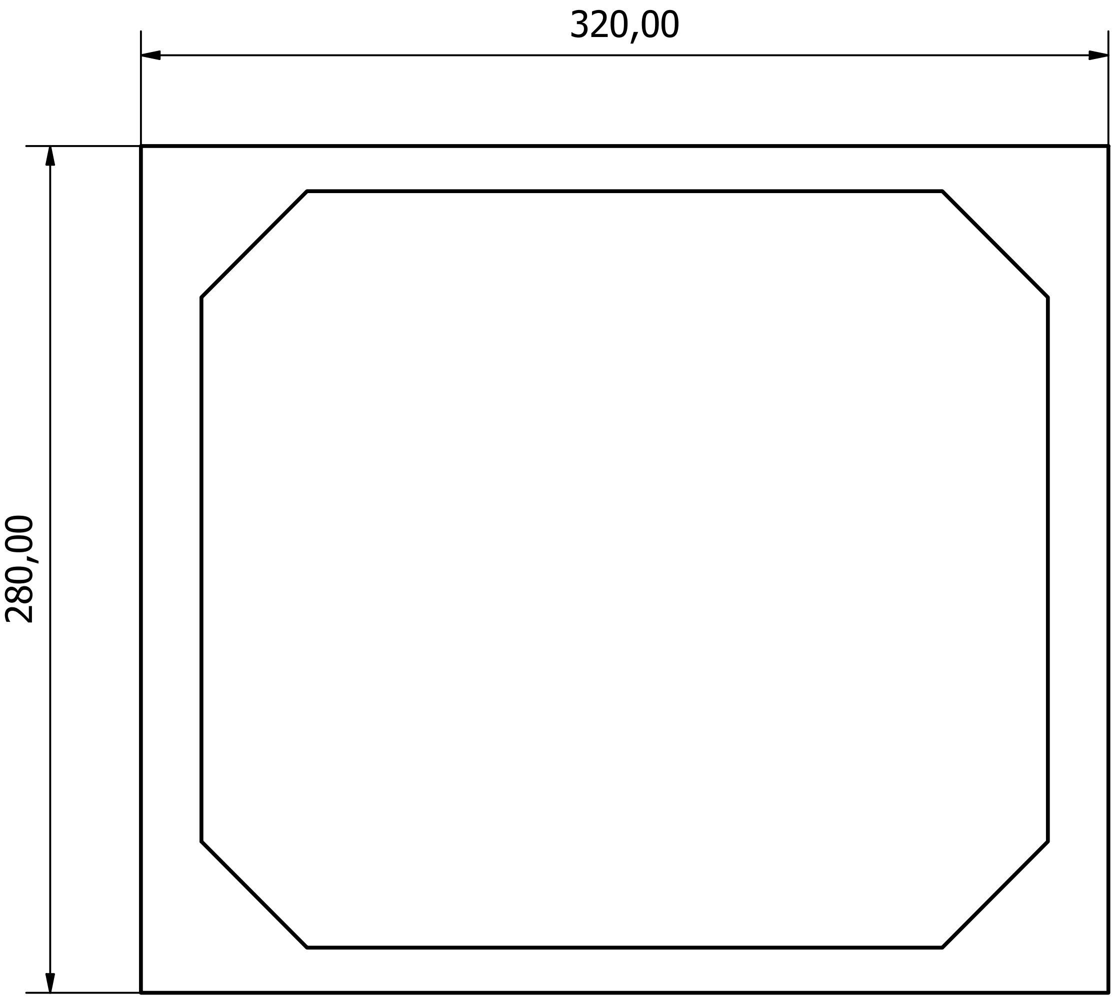
Sheet Thickness: 3,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Upper Idler Plulley Cantilever Plate 2

Swerve Drive AGV	Edition	Sheet
		28 / 207



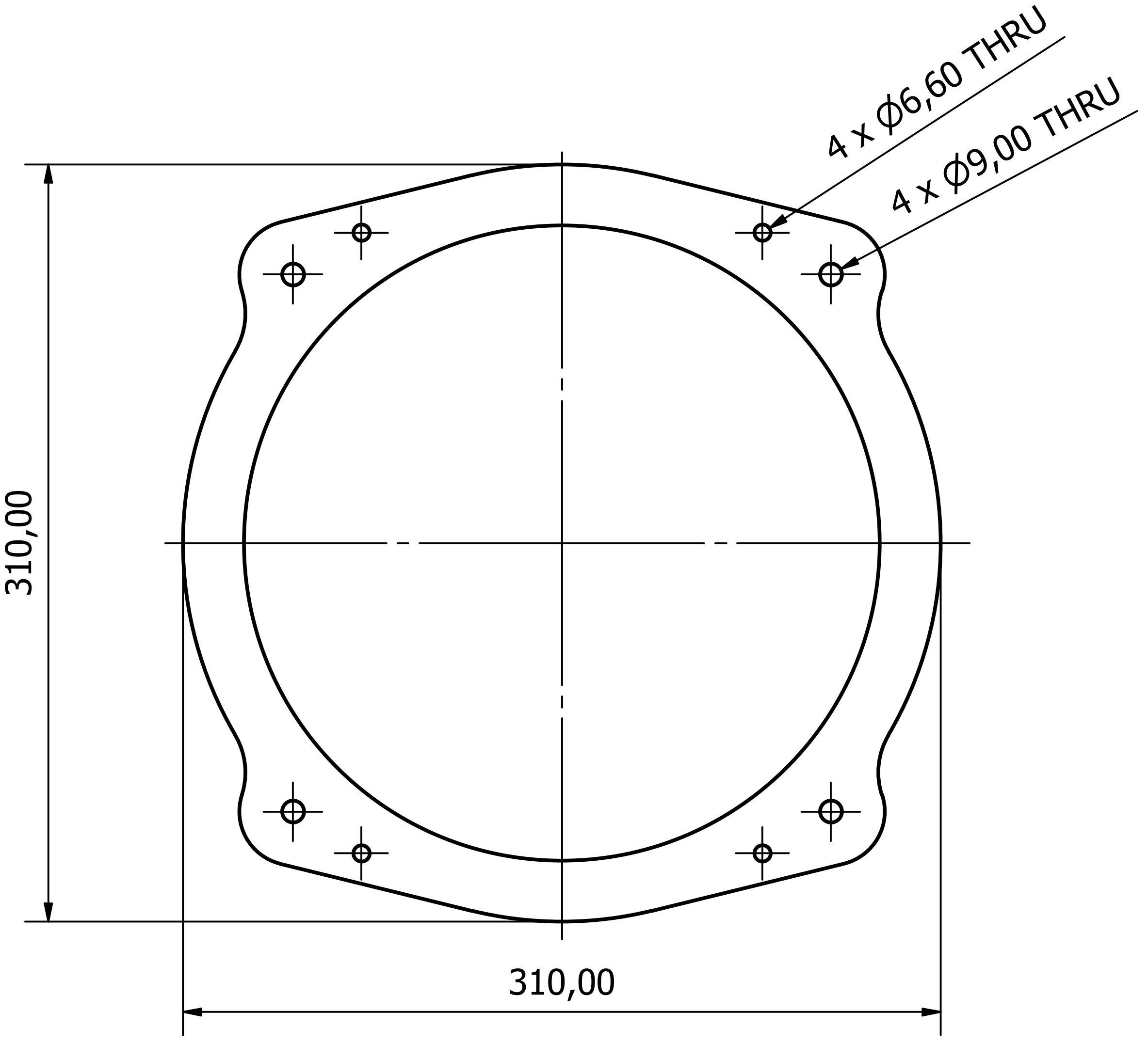
Sheet Thickness: 2,50 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Upper Structure Brace

Swerve Drive AGV	Edition	Sheet
		29 / 207



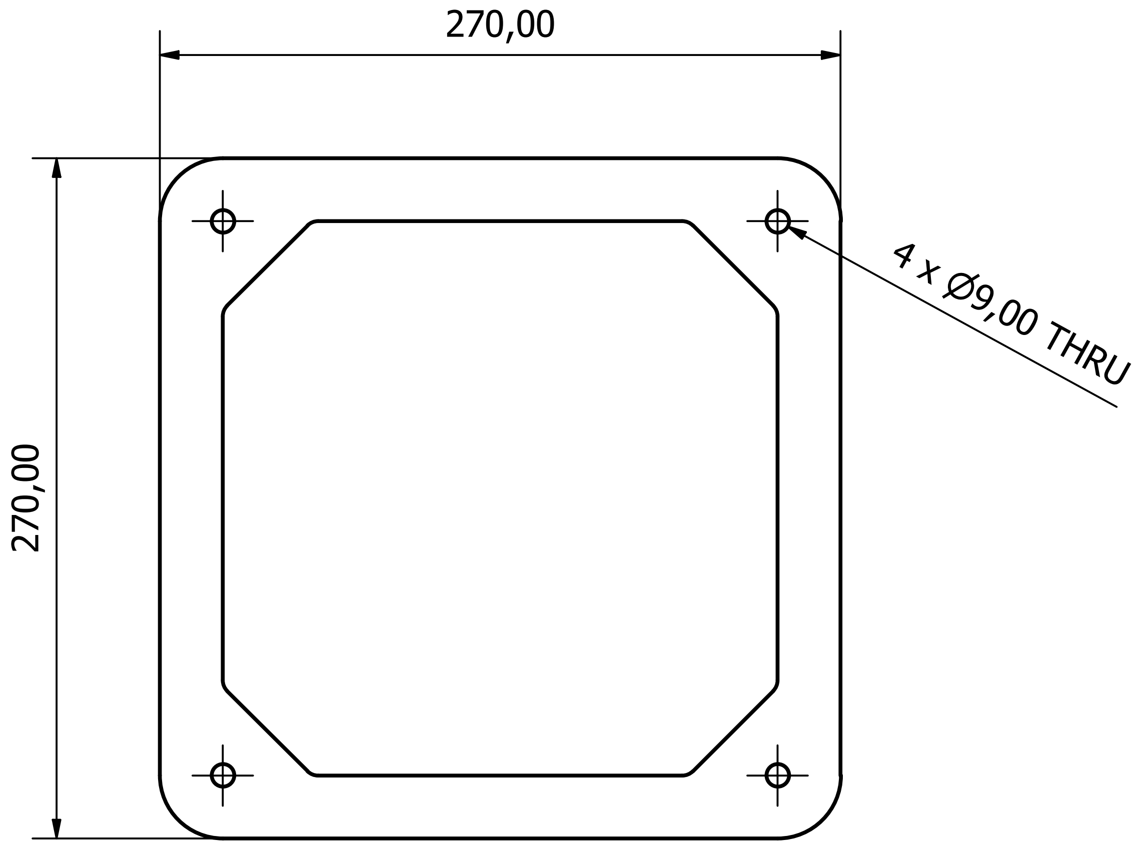
Sheet Thickness: 3,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Caster Suspension Bottom Brace

	Edition	Sheet
Swerve Drive AGV		30 / 207



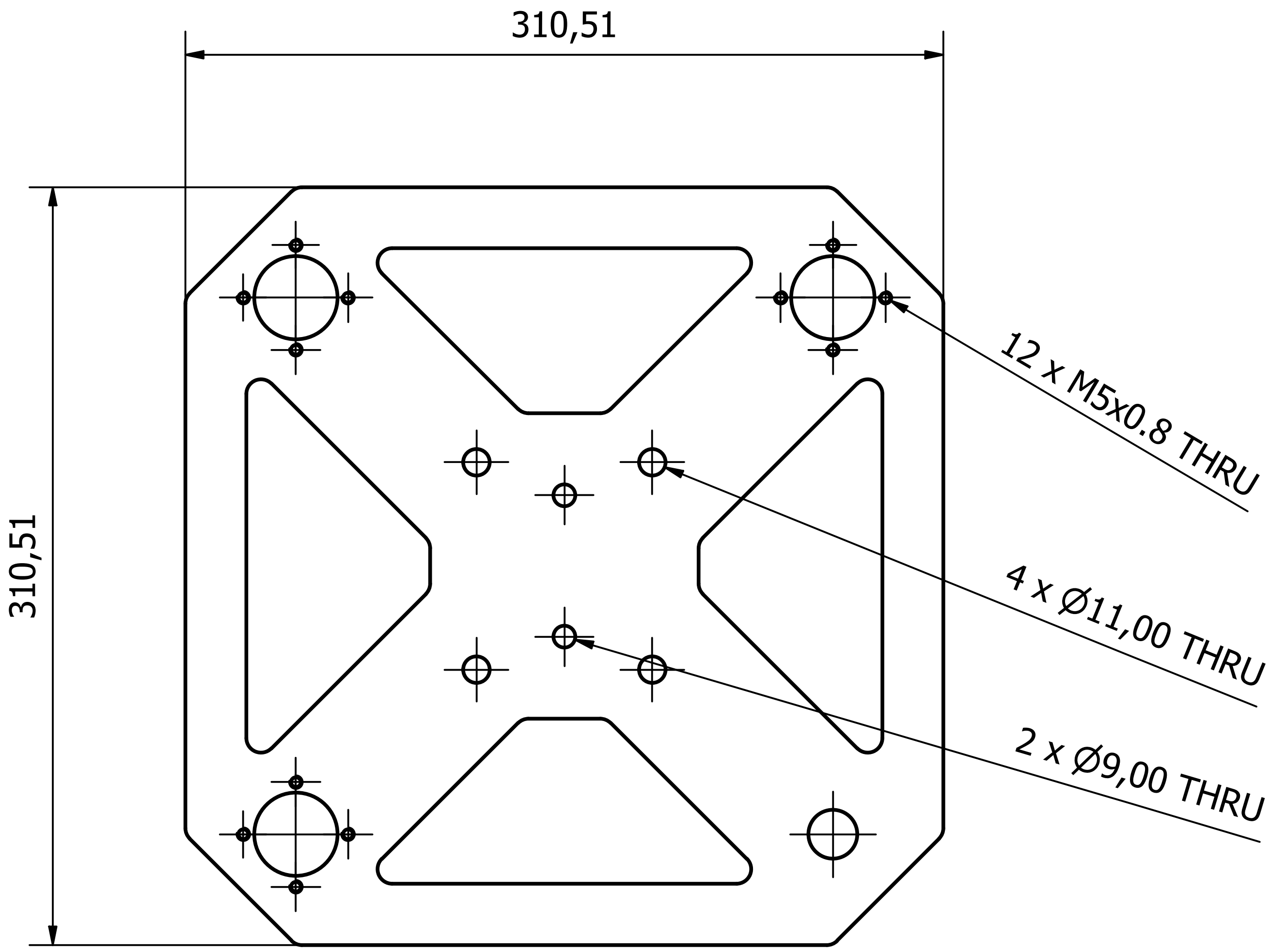
Sheet Thickness: 3,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Caster Suspension Top Brace

Swerve Drive AGV	Edition	Sheet
		31 / 207



Sheet Thickness: 4,00 mm

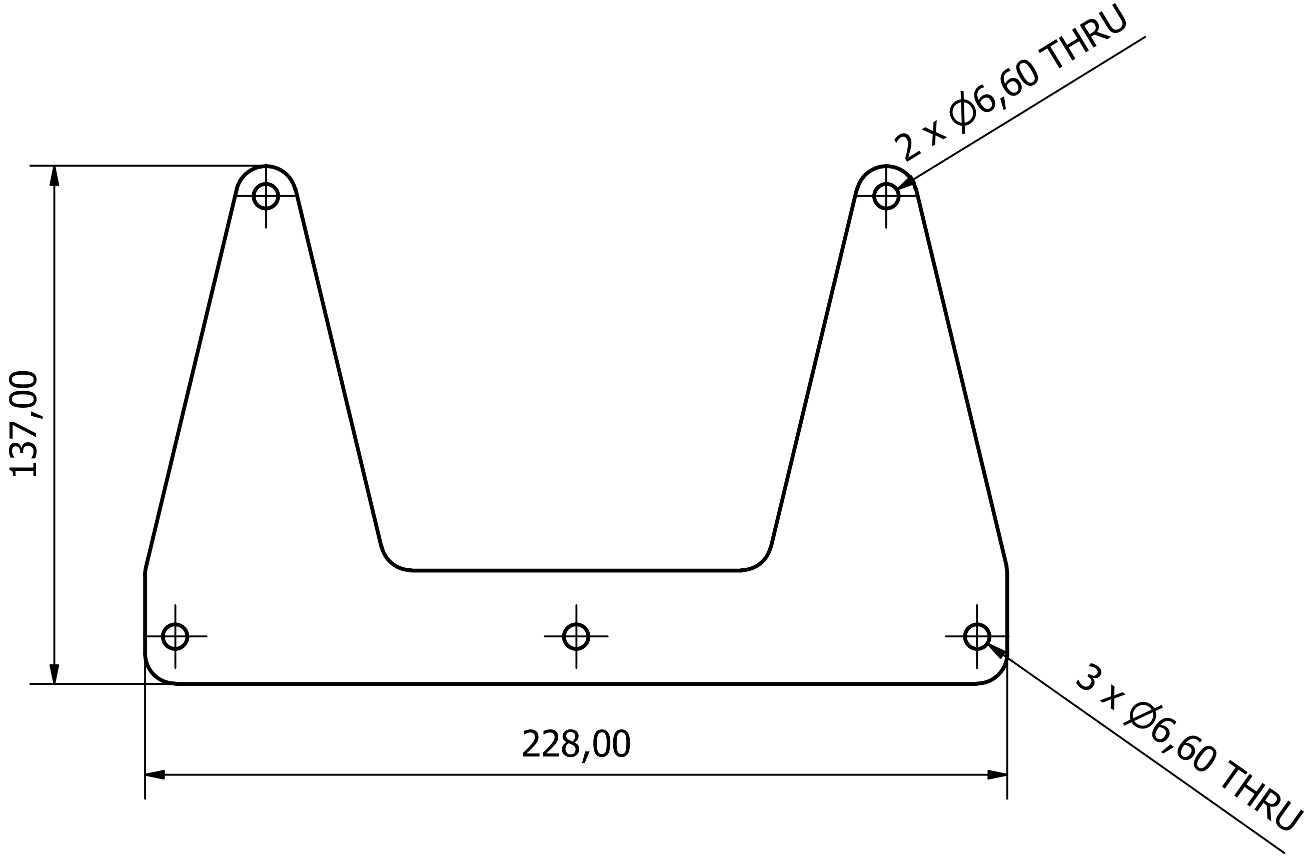
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Floating Caster Mount

Swerve Drive AGV

Edition	Sheet
	32 / 207



Sheet Thickness: 4,00 mm

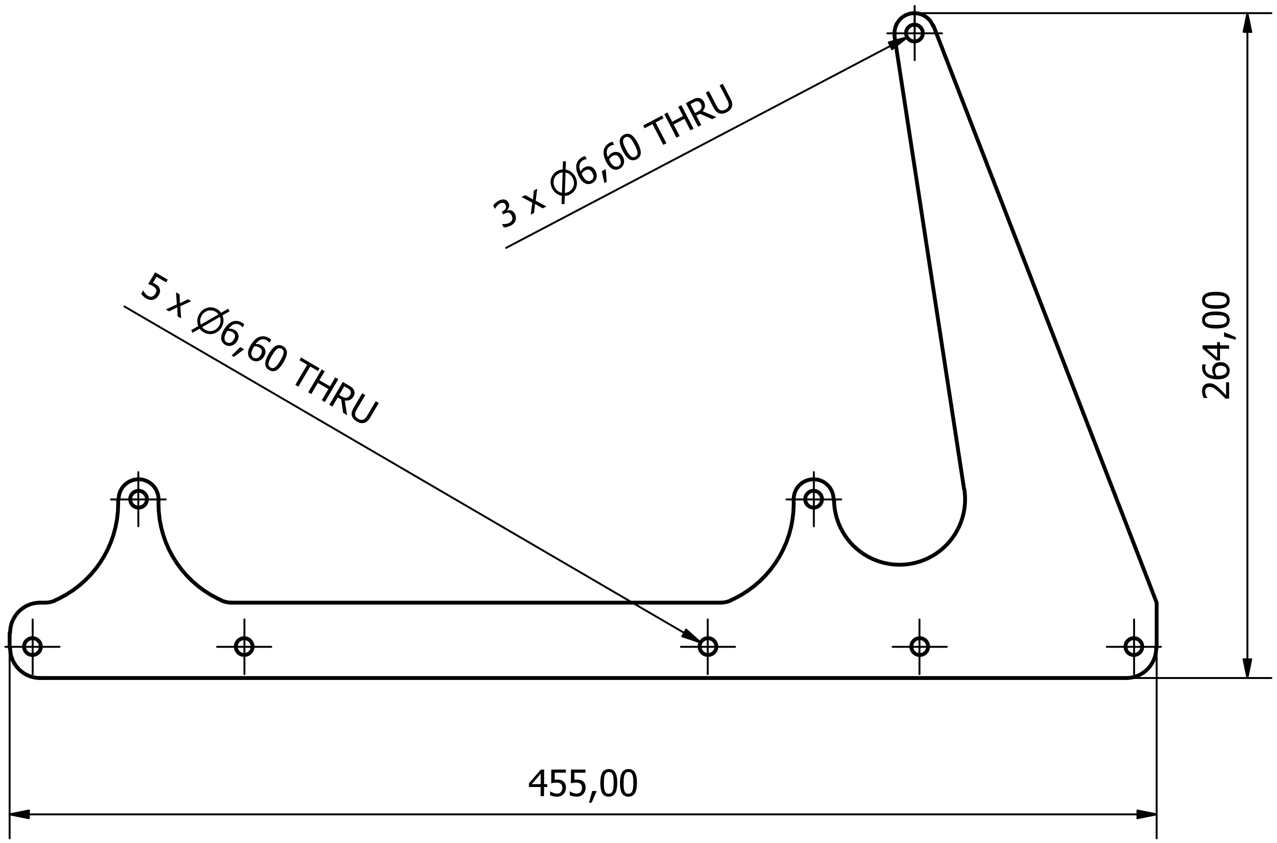
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Lower Caster Unit Stabalisation Flange

Swerve Drive AGV

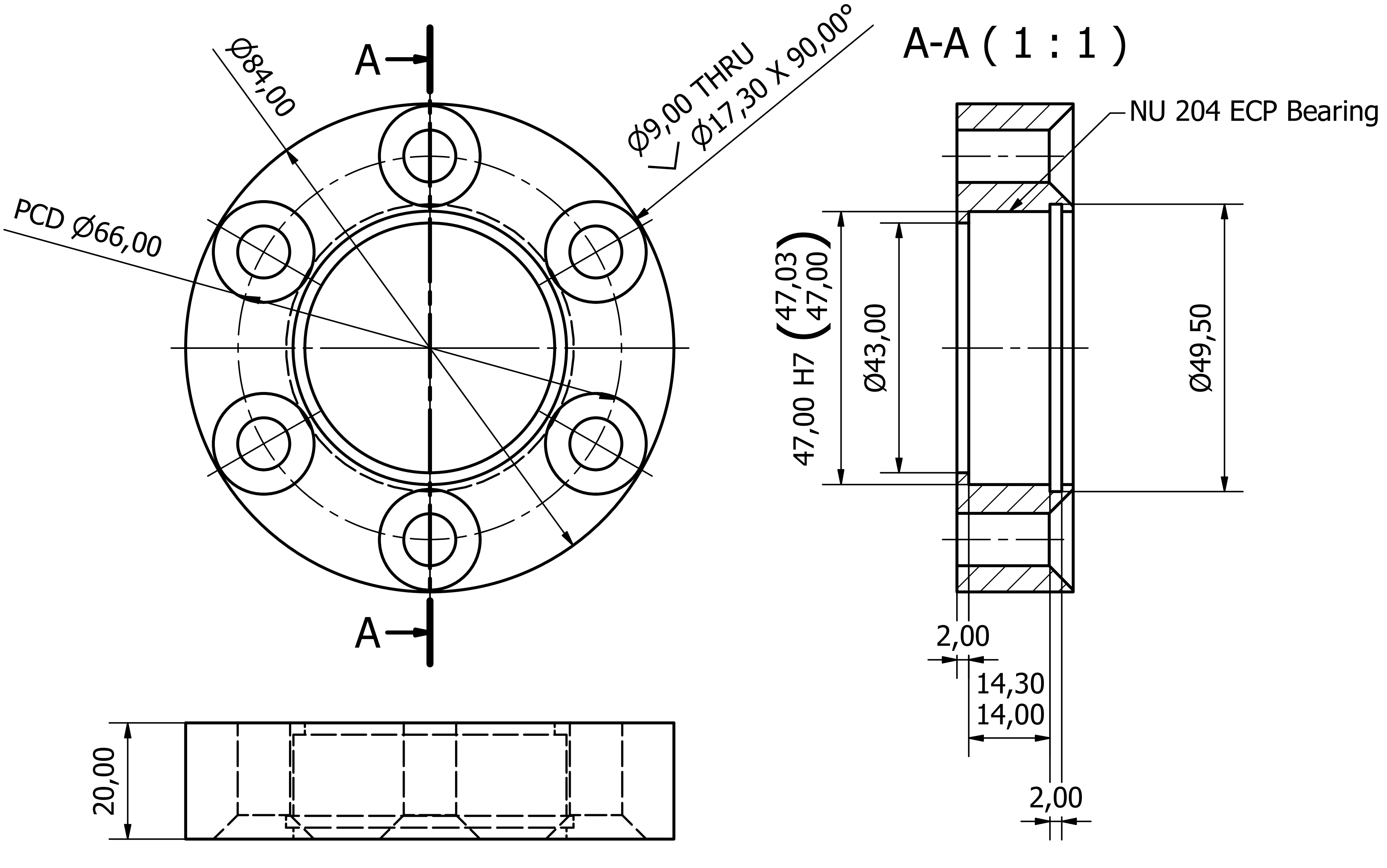
Edition	Sheet
	33 / 207



Sheet Thickness: 4,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY	Lower Drive Unit Stabalisation Flange		
	Swerve Drive AGV	Edition	Sheet 34 / 207



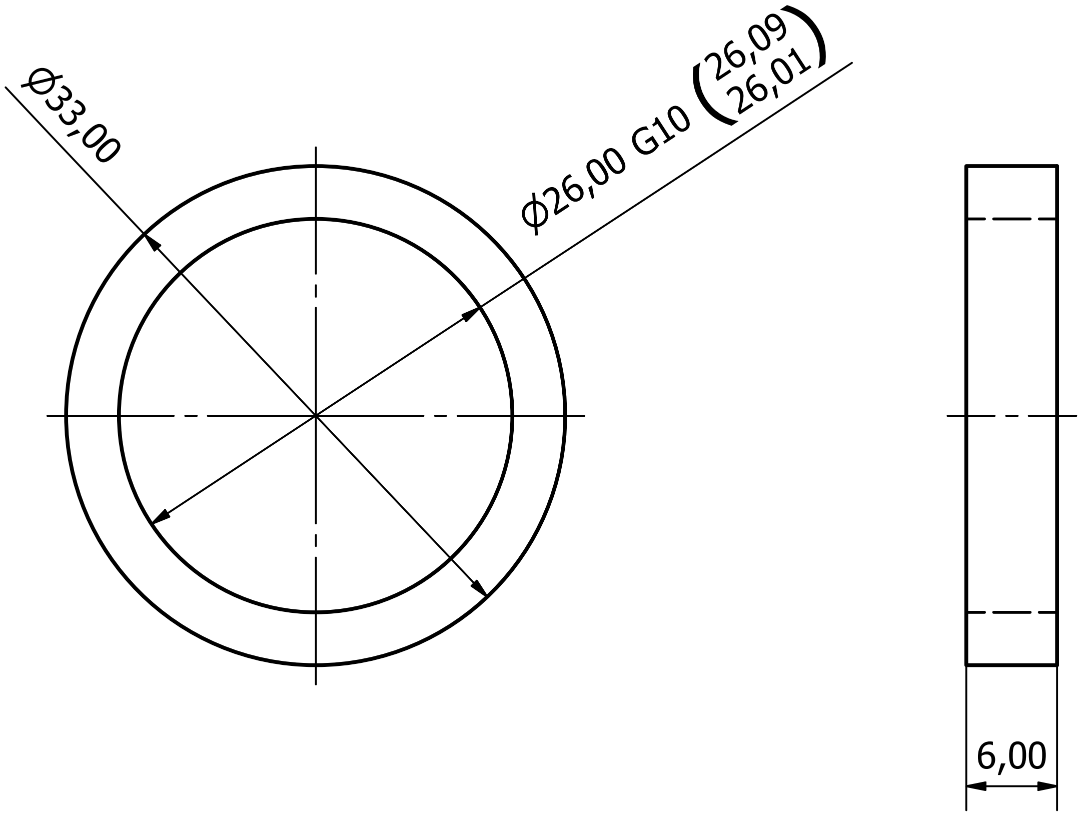
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Radial Bearing Housing

Swerve Drive AGV

Edition	Sheet
	35 / 207



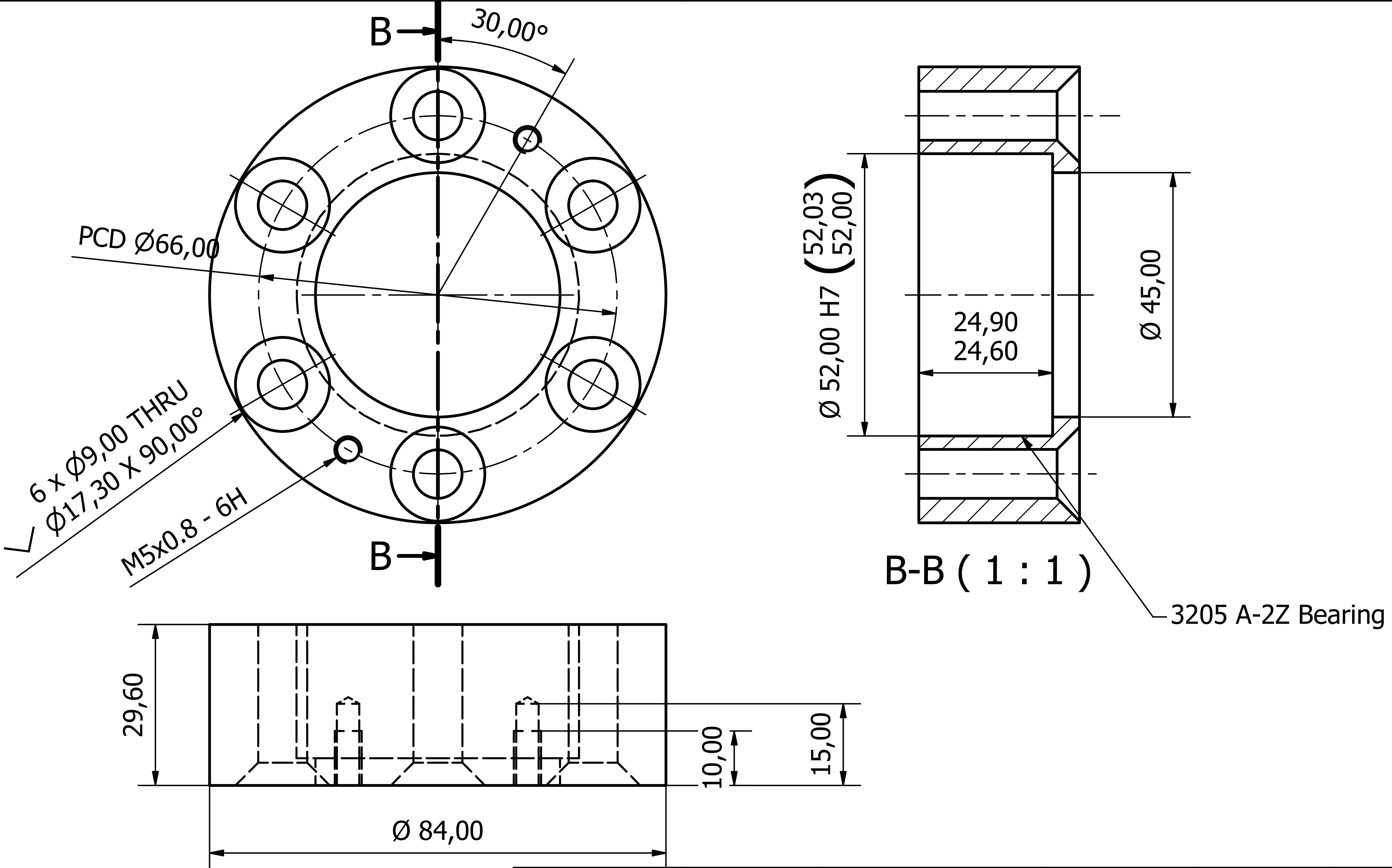
Designed by AlexM	Quantity 2	Manufactured by Dale Flynn	Material Aluminum 6061	Date 07/03/2019

NELSON MANDELA UNIVERSITY

Shaft Locknut Spacer

Swerve Drive AGV

Edition	Sheet
	36 / 207

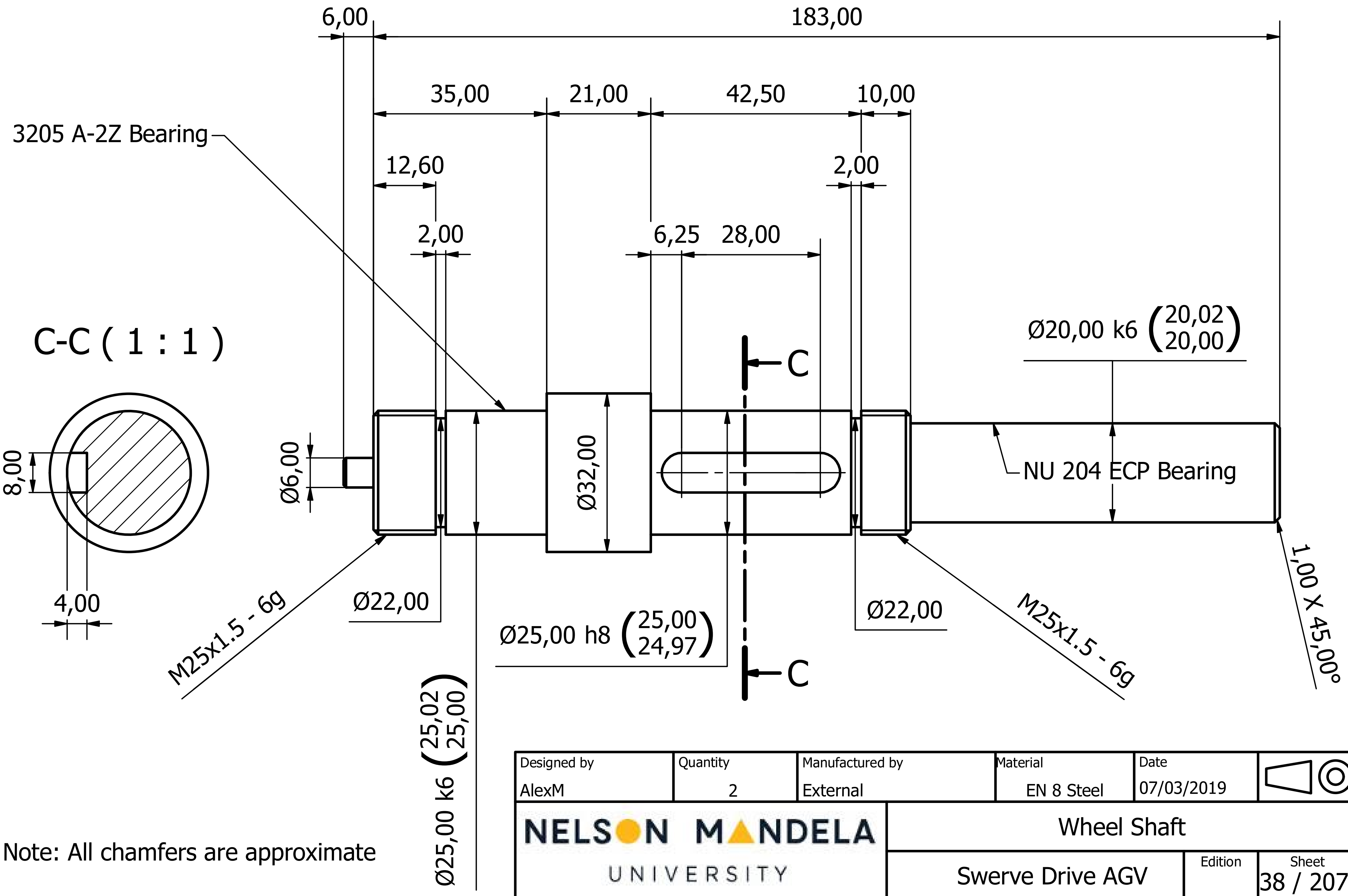


Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Thrust Bearing Housing

	Edition	Sheet
Swerve Drive AGV		37 / 207



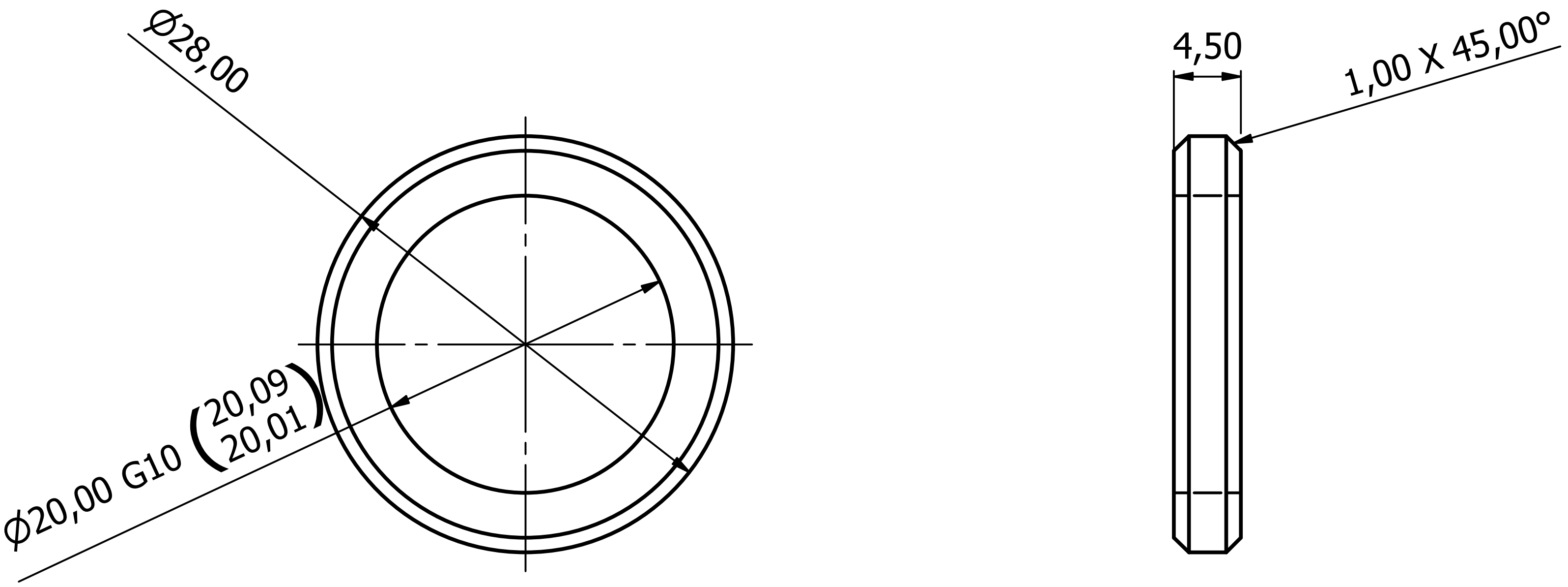
Note: All chamfers are approximate

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	EN 8 Steel	07/03/2019

NELSON MANDELA UNIVERSITY

Wheel Shaft

Swerve Drive AGV	Edition	Sheet
		38 / 207



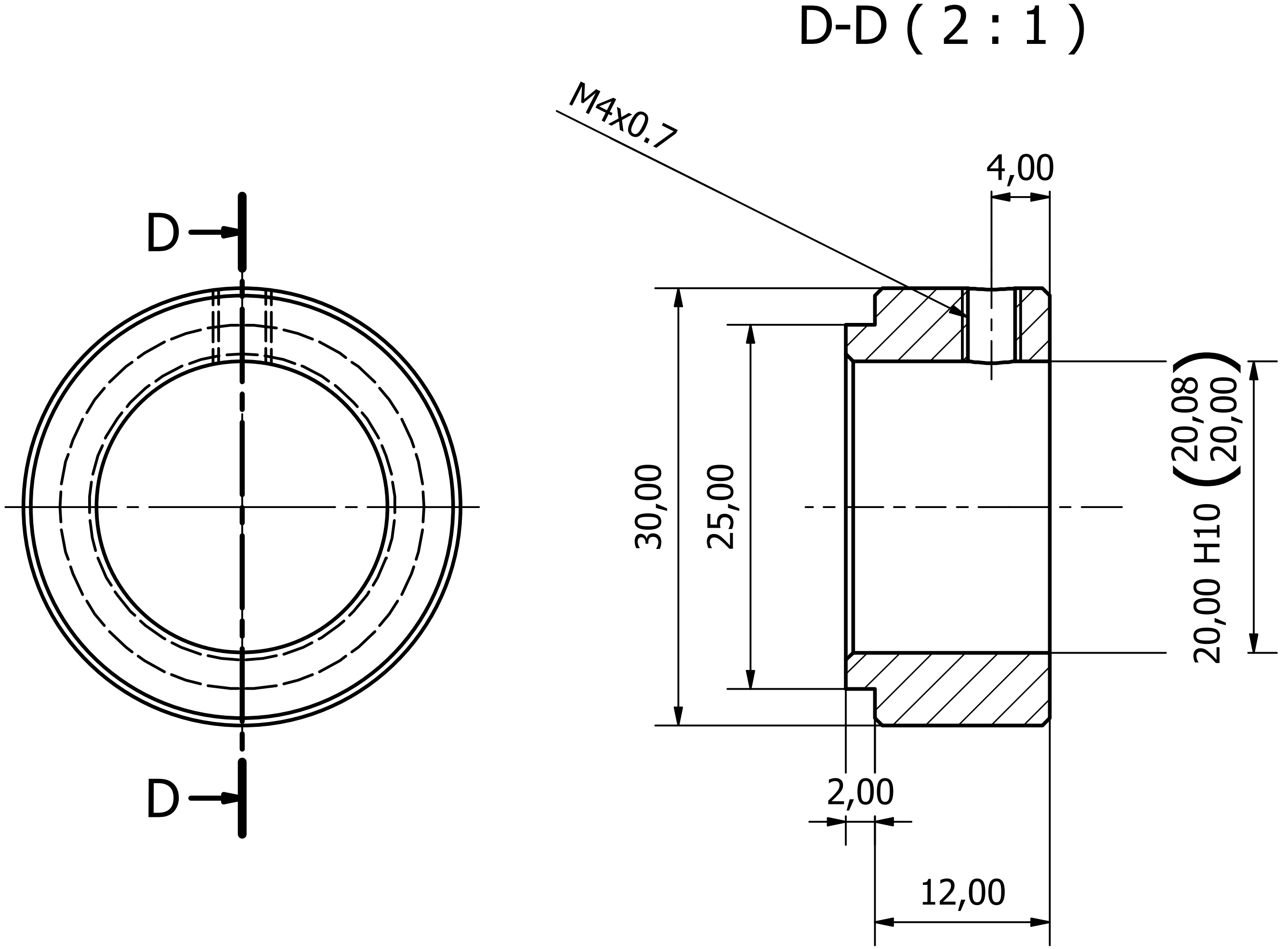
All chamfers are approximate

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	Dale Flynn	Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Floating Bearing Spacer

	Edition	Sheet
Swerve Drive AGV		39 / 207



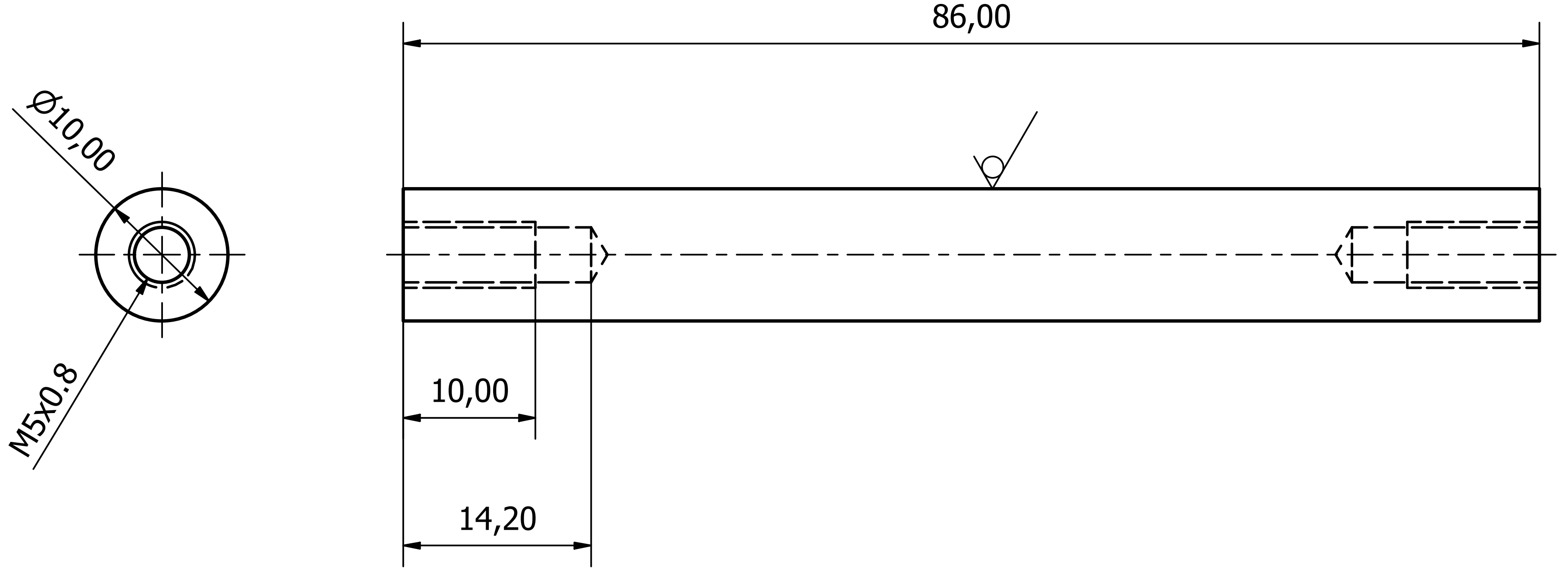
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	Dale Flynn	Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Floating Bearing Lock Ring

Swerve Drive AGV

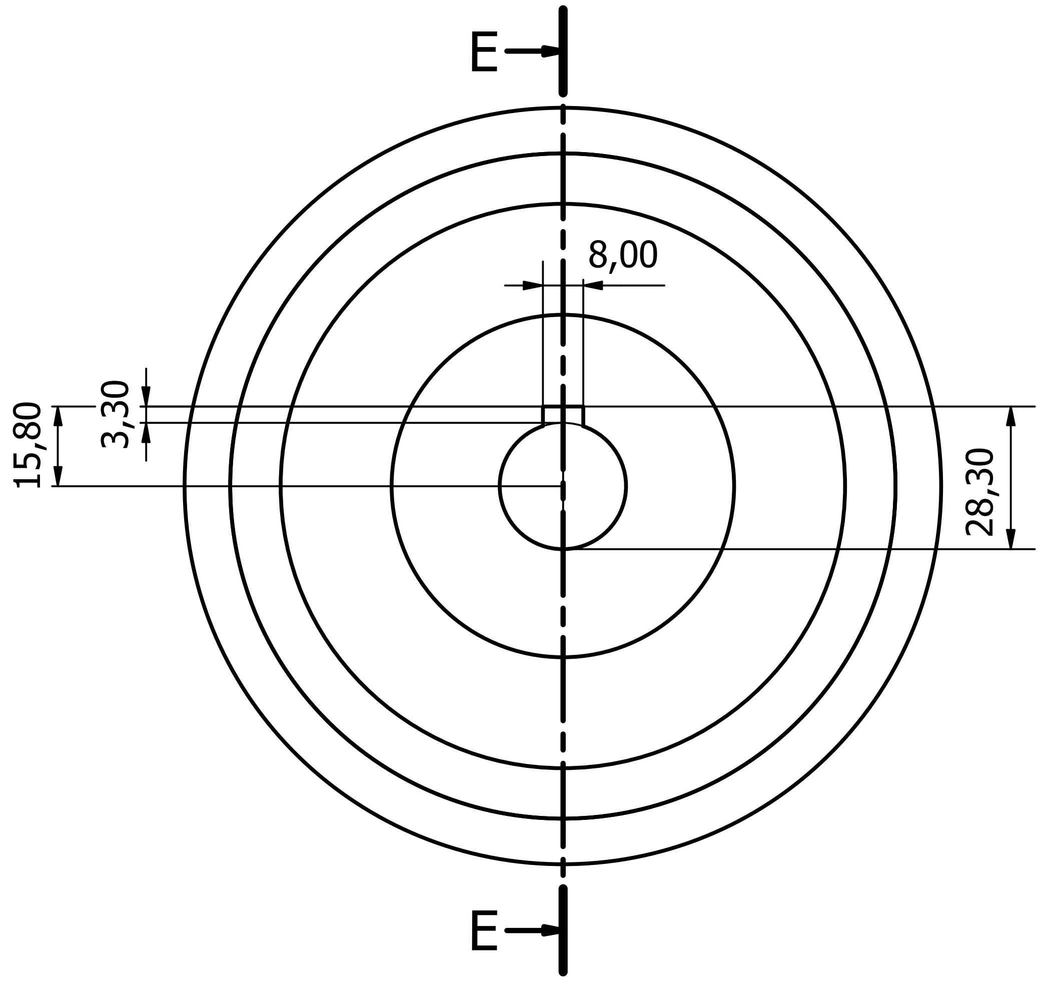
Edition	Sheet
	40 / 207



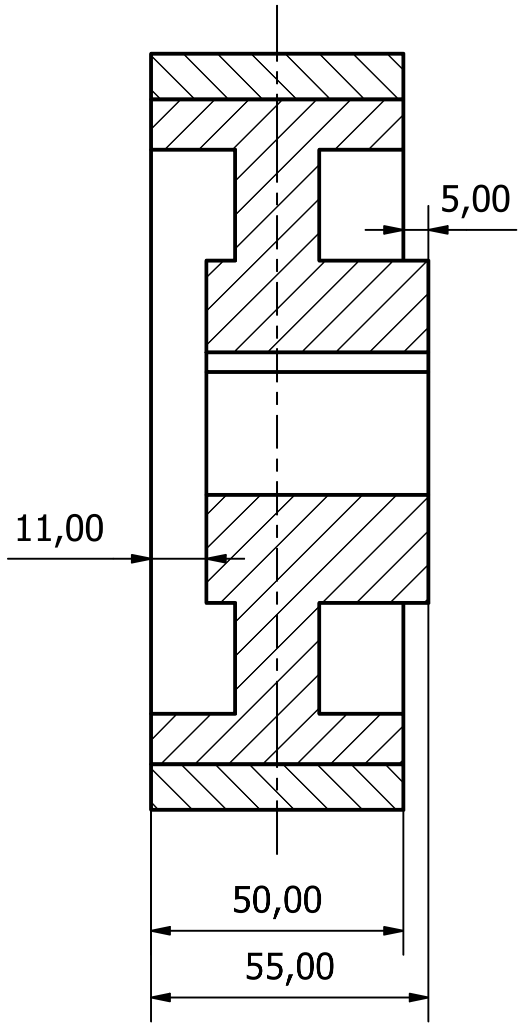
Designed by	Quantity	Manufactured by	Material	Date
AlexM	4	Alex Macfarlane	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Spacer Beams		
Swerve Drive AGV	Edition	Sheet 41 / 207



E-E (1 :1.5)

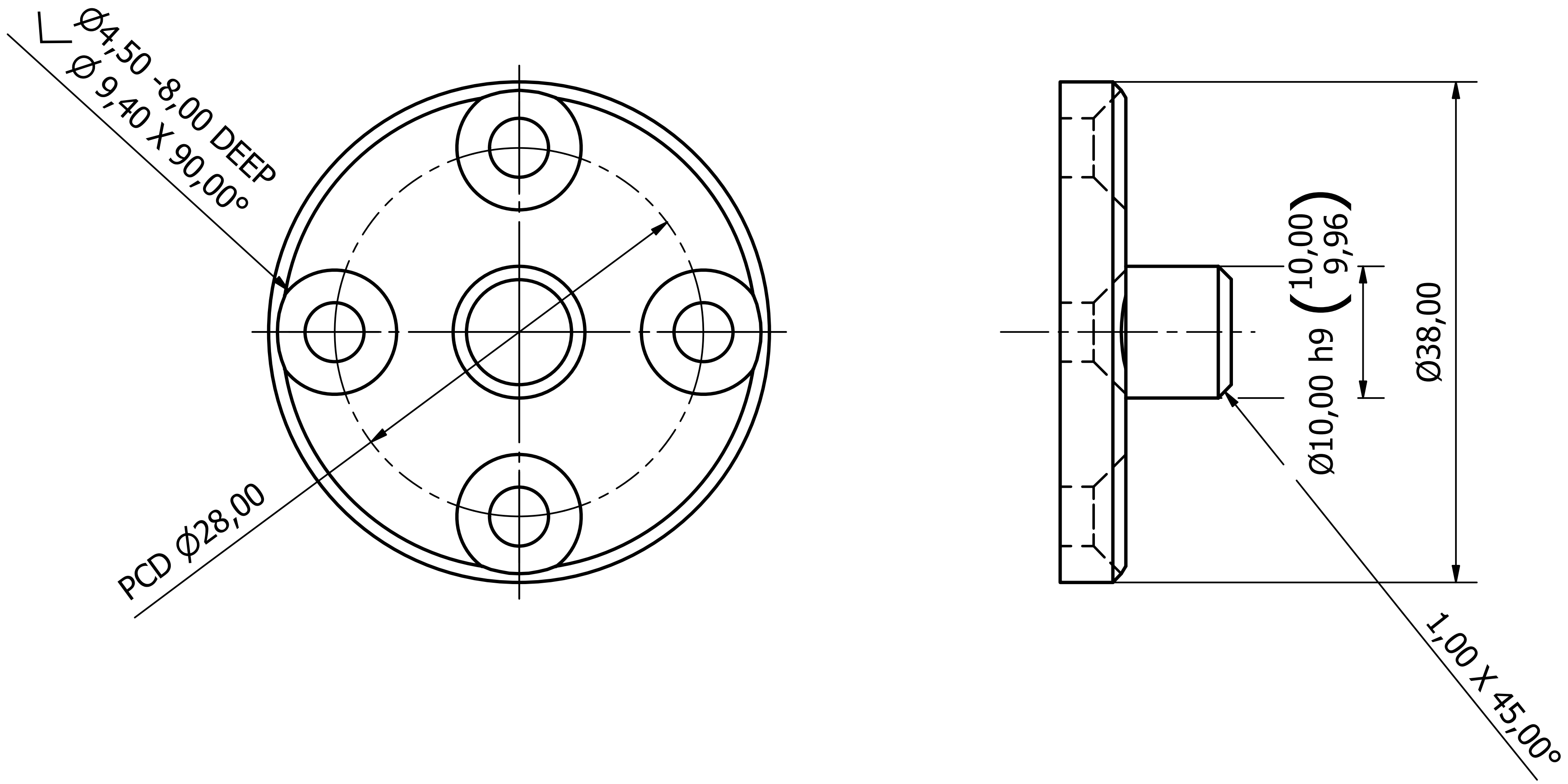


Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	Ladder and Caster		07/03/2019

NELSON MANDELA UNIVERSITY

Drive Wheel

	Edition	Sheet
Swerve Drive AGV		42 / 207

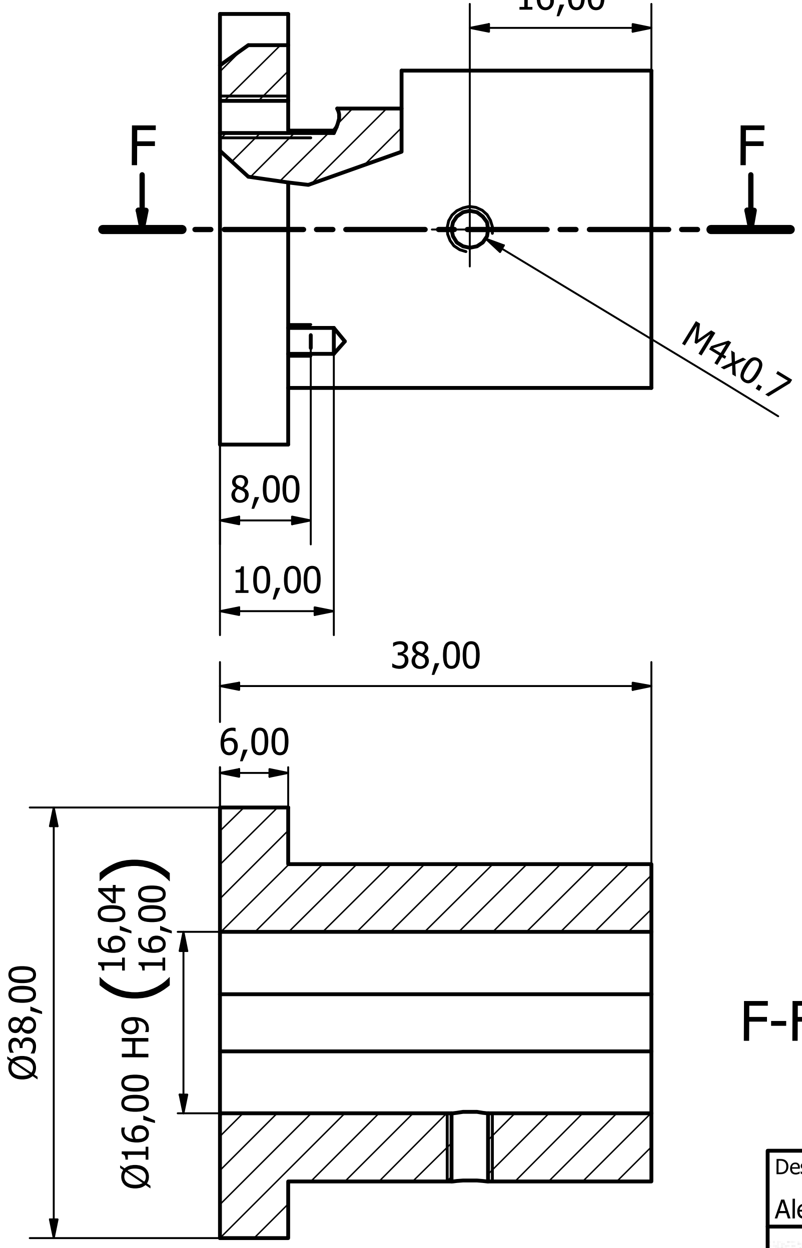


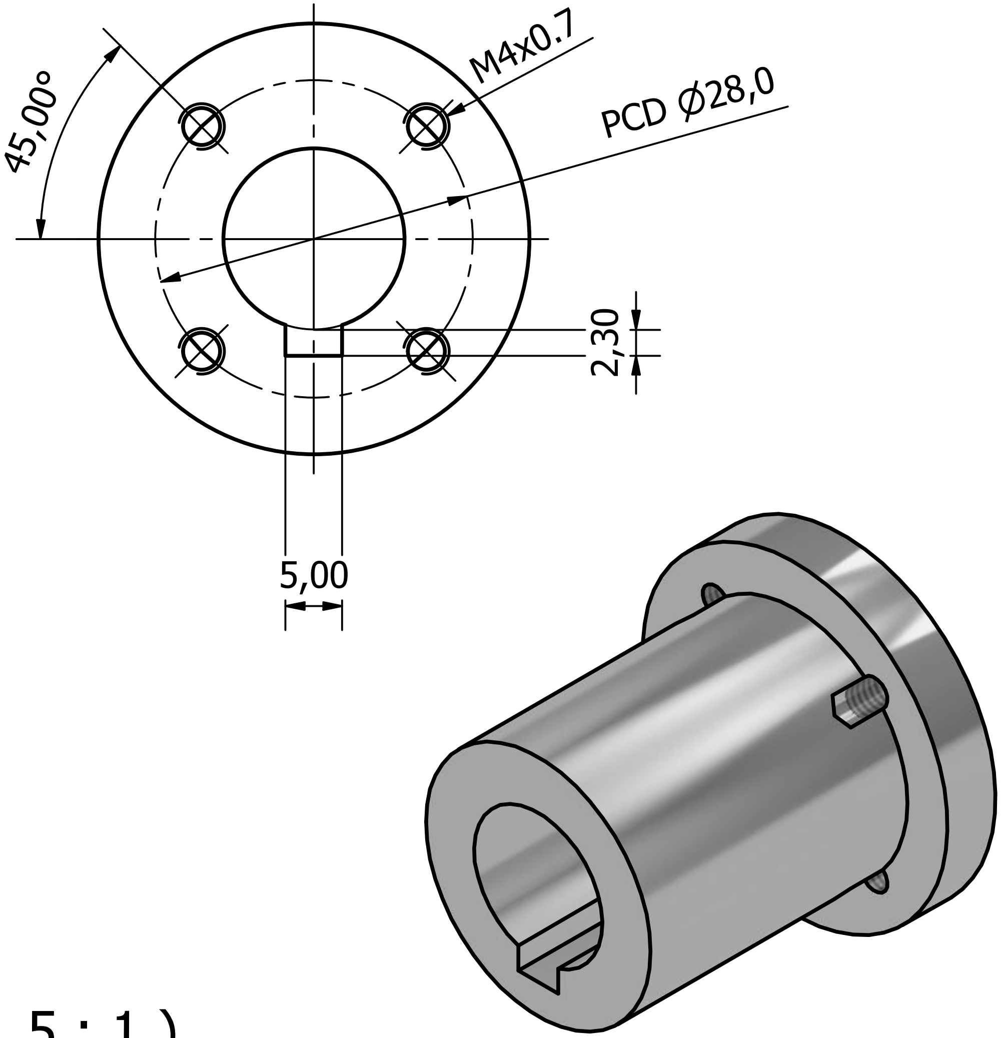
Note: Chamfers are approximate

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Gearbox Connection Unit Part A		
Swerve Drive AGV	Edition	Sheet 43 / 207





F-F (1.5 : 1)

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	Dale Flynn	Aluminum 6061	07/03/2019

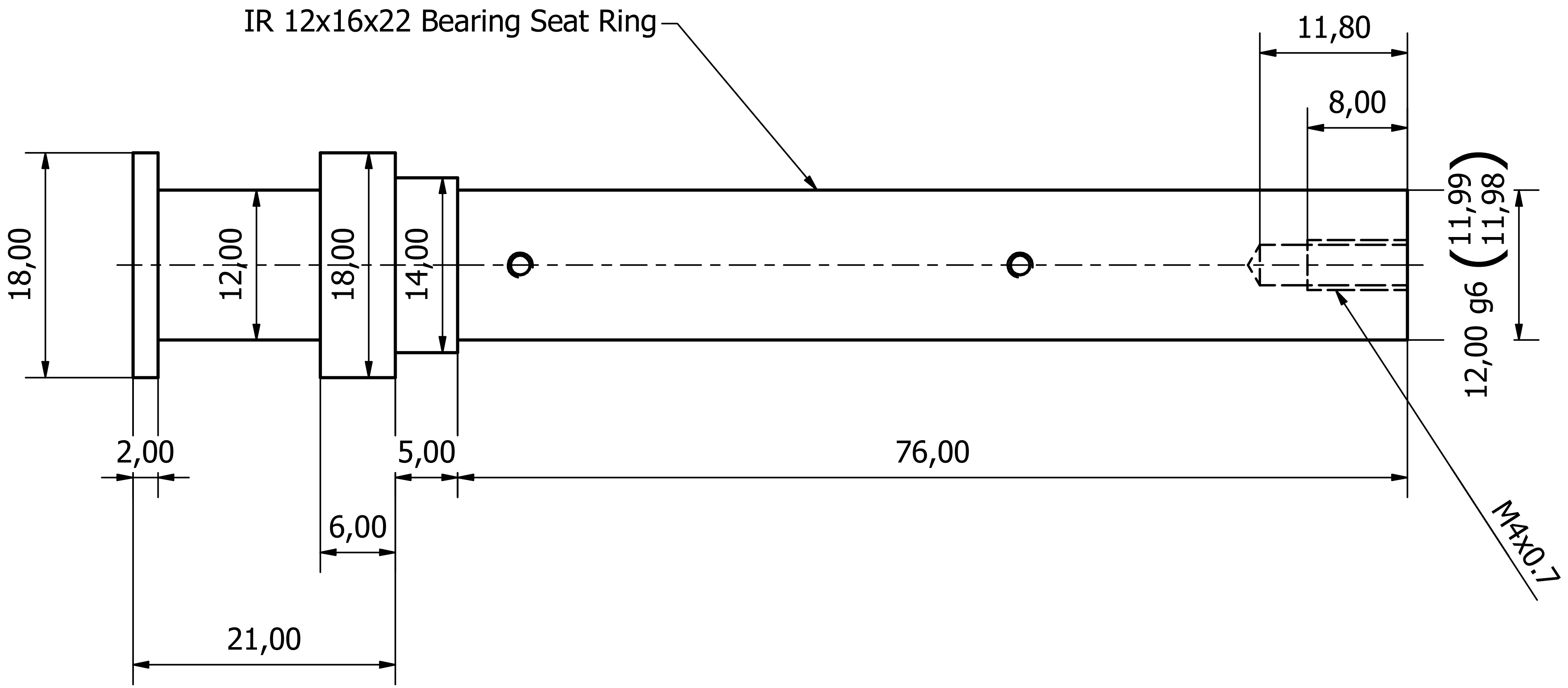
NELSON MANDELA UNIVERSITY

Gearbox Connection Unit Part B

Swerve Drive AGV

Edition

Sheet 44 / 207

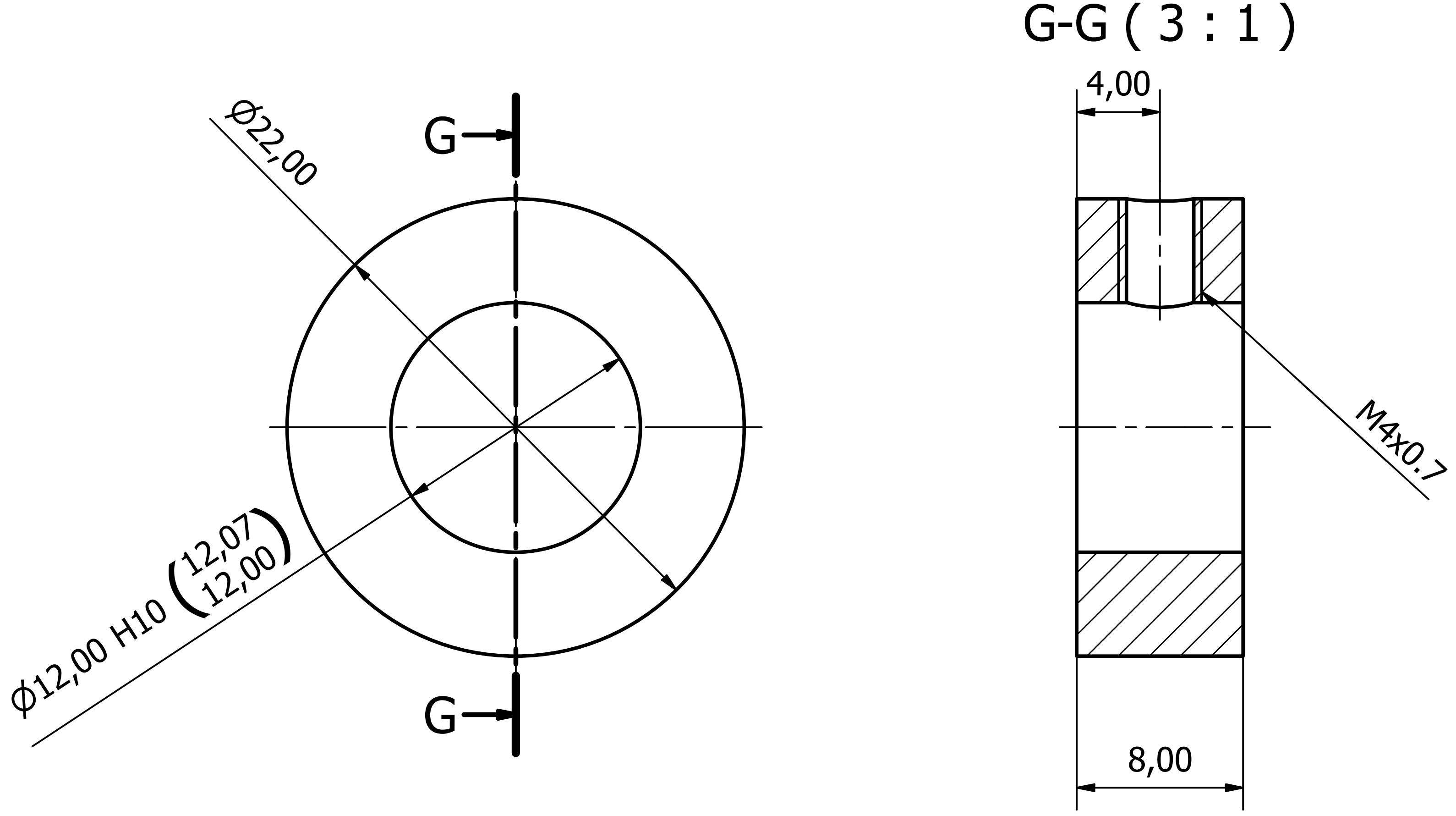


Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	EN 8 Steel	07/03/2019

NELSON MANDELA UNIVERSITY

Smooth Idler Pulley Bearing Shaft

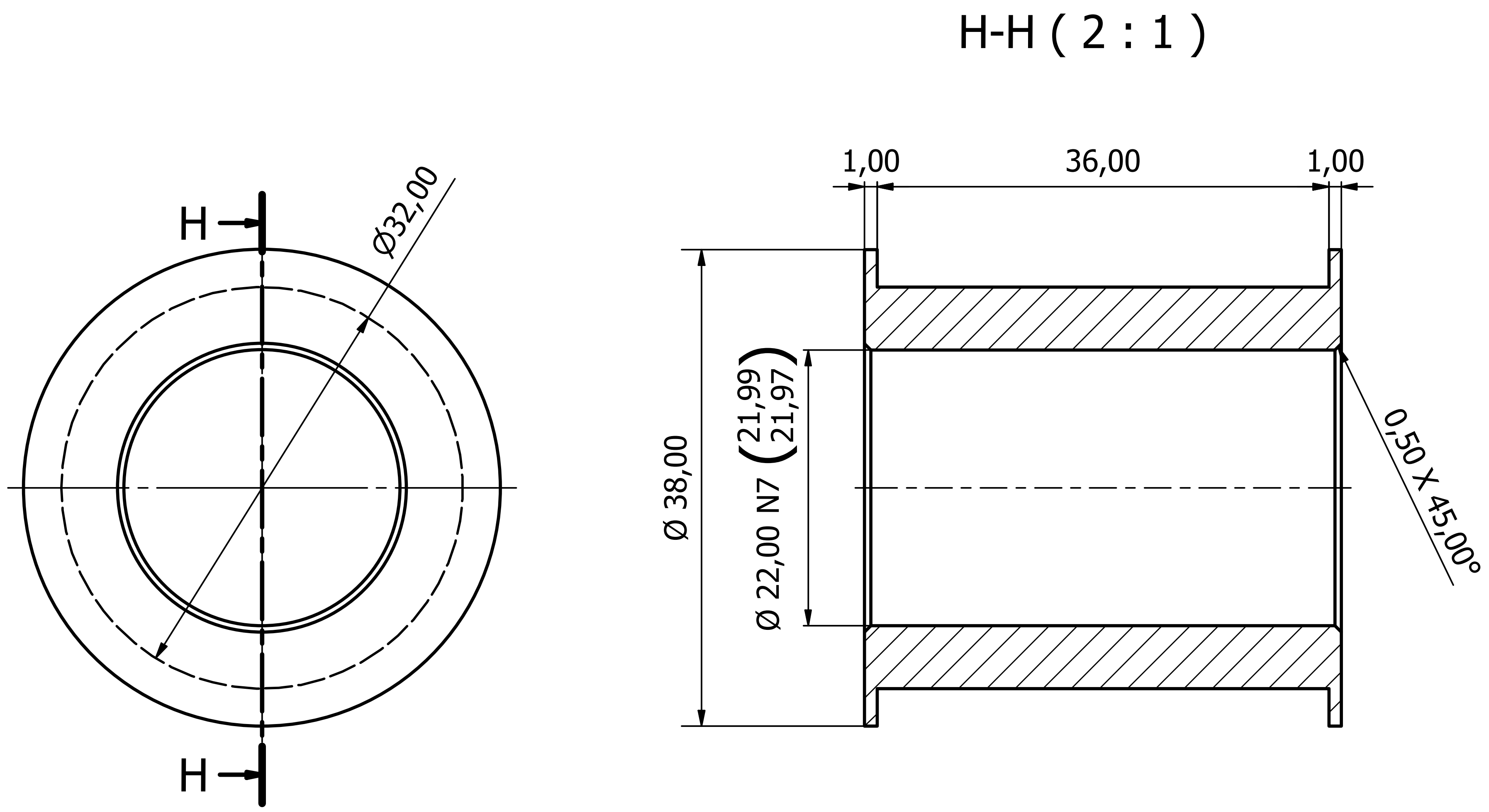
Swerve Drive AGV	Edition	Sheet 45 / 207



Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	Dale Flynn	Aluminum 6061	07/03/2019

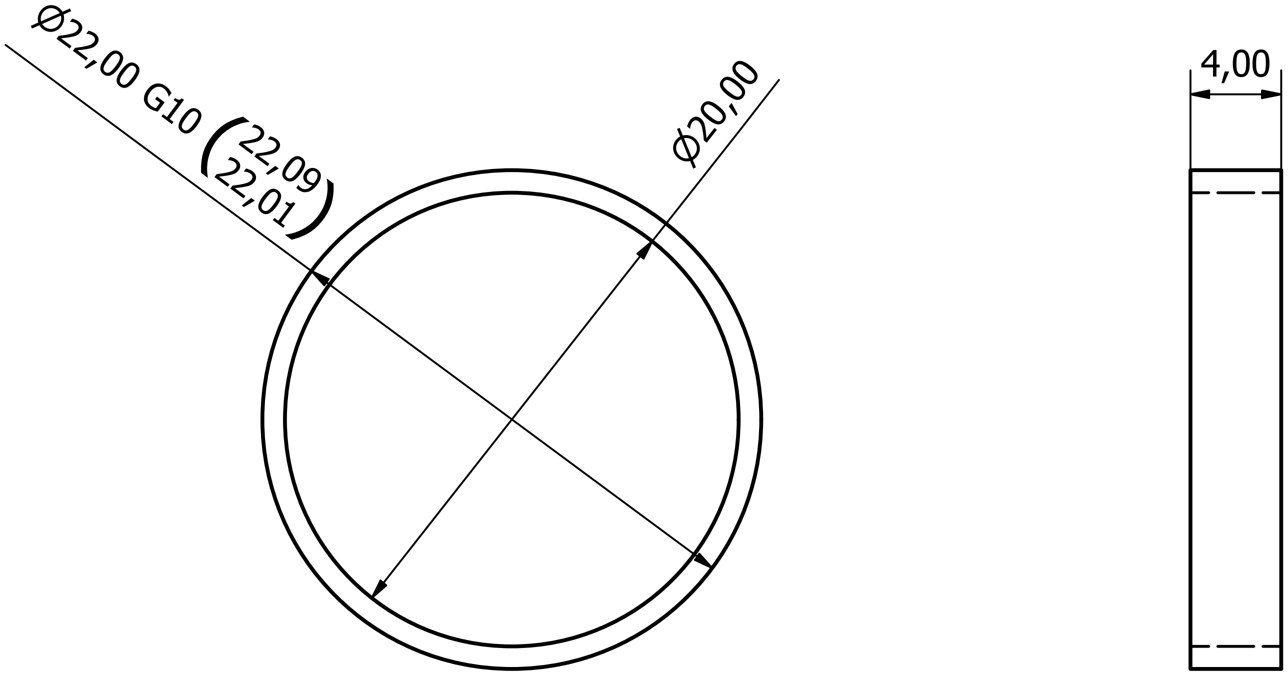
NELSON MANDELA UNIVERSITY

Smooth Idler Pulley Shaft Collar		
Swerve Drive AGV	Edition	Sheet 46 / 207



Note: Chamfers are approximate

Designed by AlexM	Quantity 2	Manufactured by External	Material Aluminum 6061	Date 07/03/2019
NELSON MANDELA UNIVERSITY	Smooth Idler Pulley			
	Swerve Drive AGV		Edition	Sheet 47 / 207

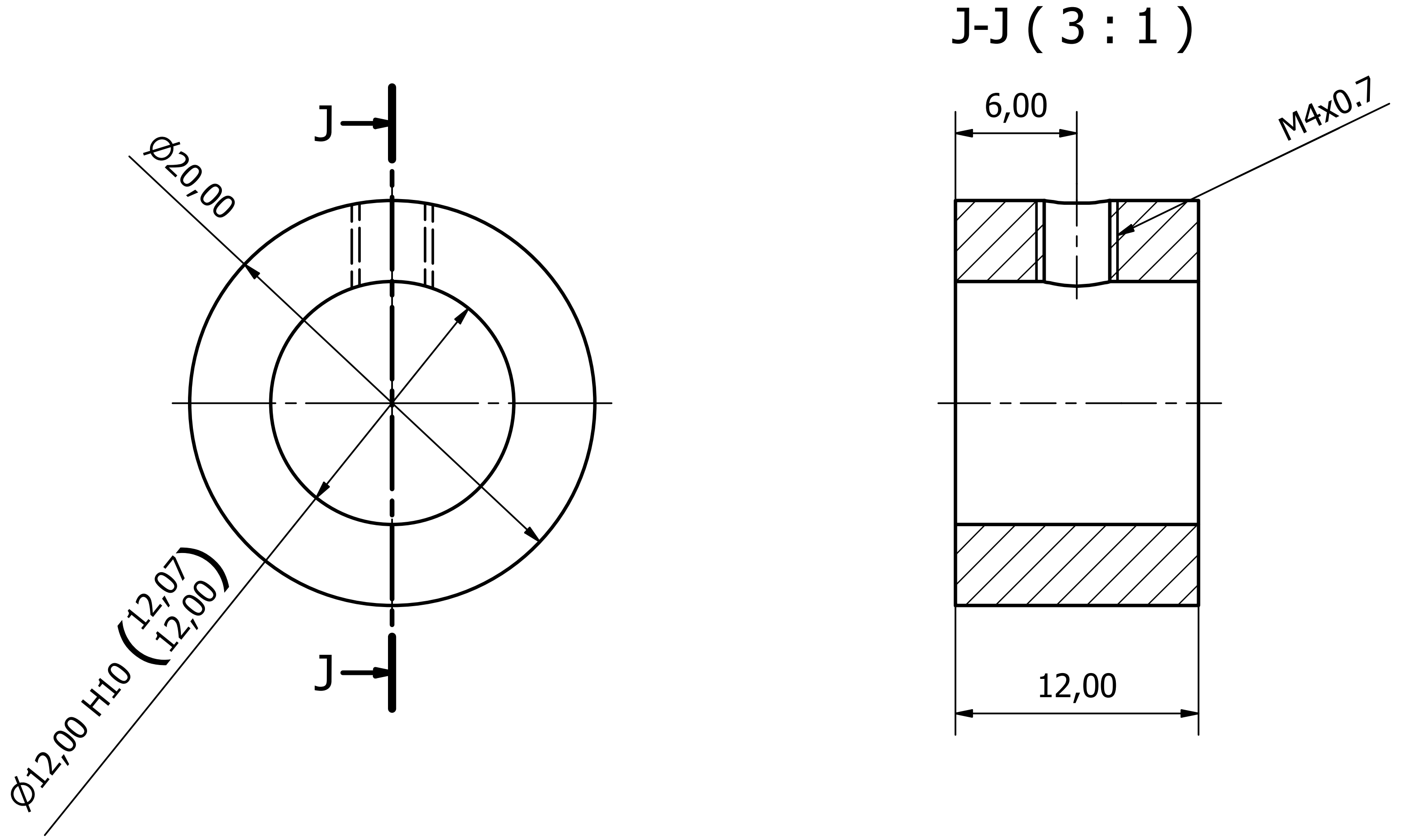


Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Aluminum 6061	07/03/2019

NELSON MANDELA
UNIVERSITY

Smooth Pulley Bearing Spacer

	Edition	Sheet
Swerve Drive AGV		48 / 207

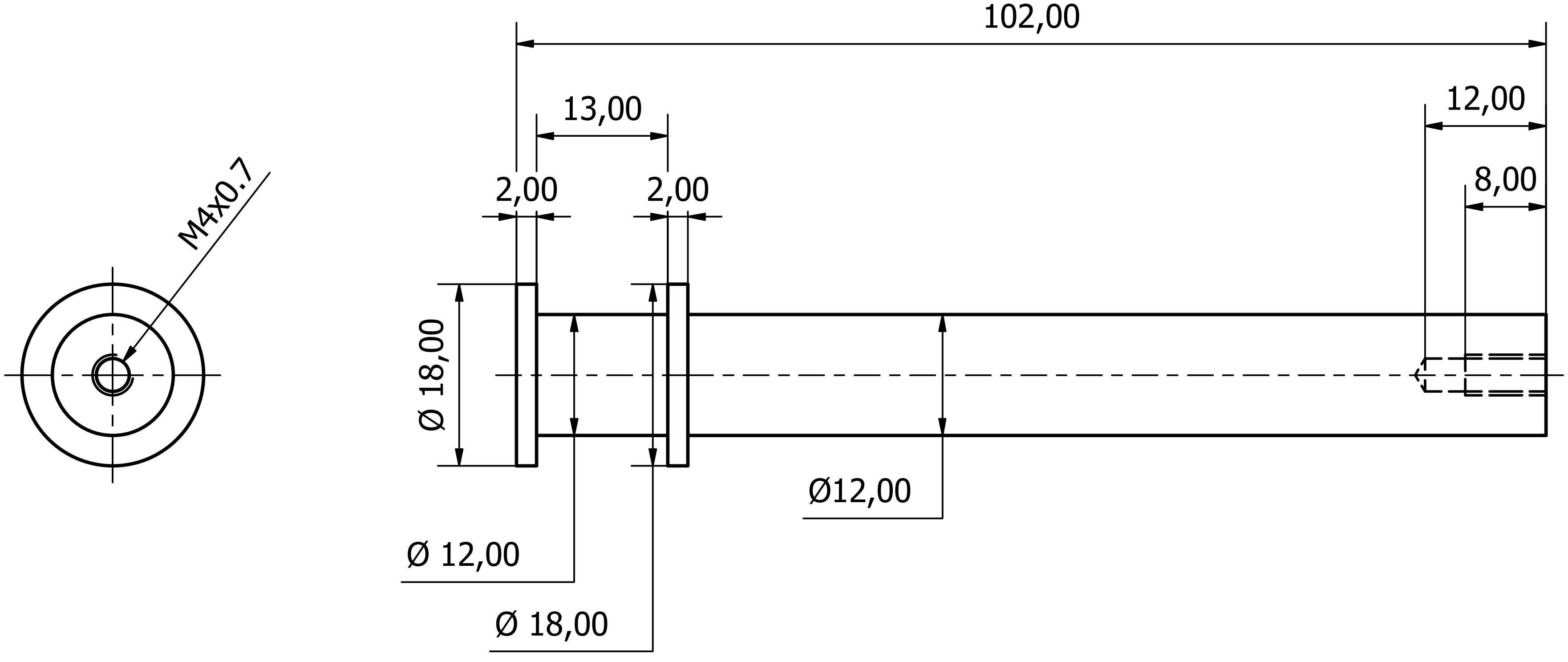


Designed by AlexM	Quantity 2	Manufactured by Dale Flynn	Material Aluminum 6061	Date 07/03/2019

NELSON MANDELA UNIVERSITY

Stationary Tensioning Rod Positioning Collar

Swerve Drive AGV	Edition	Sheet 49 / 207



Designed by	Quantity	Manufactured by	Material	Date
AlexM	EN 8 Steel	External	2	07/03/2019

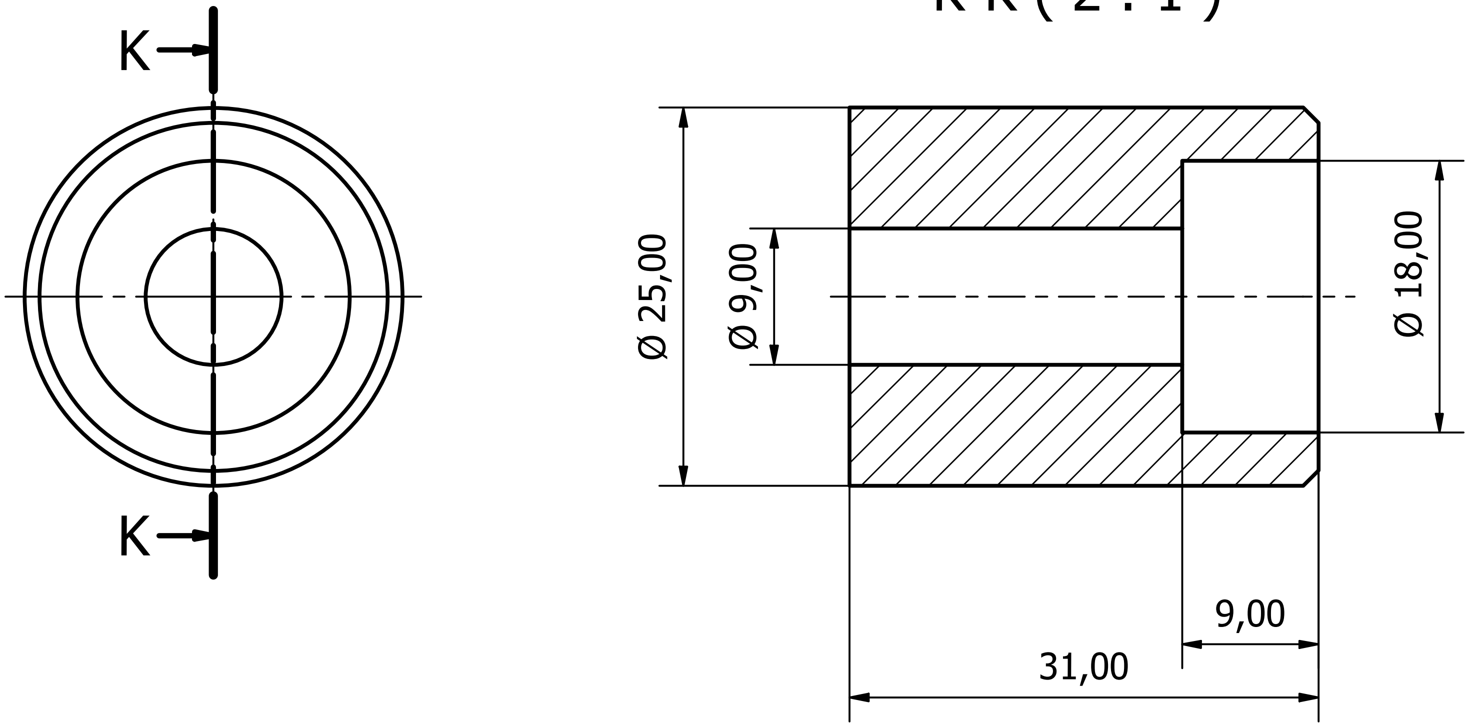
NELSON MANDELA UNIVERSITY

Stationary Tensioning Rod

Swerve Drive AGV

Edition	Sheet
	50 / 207

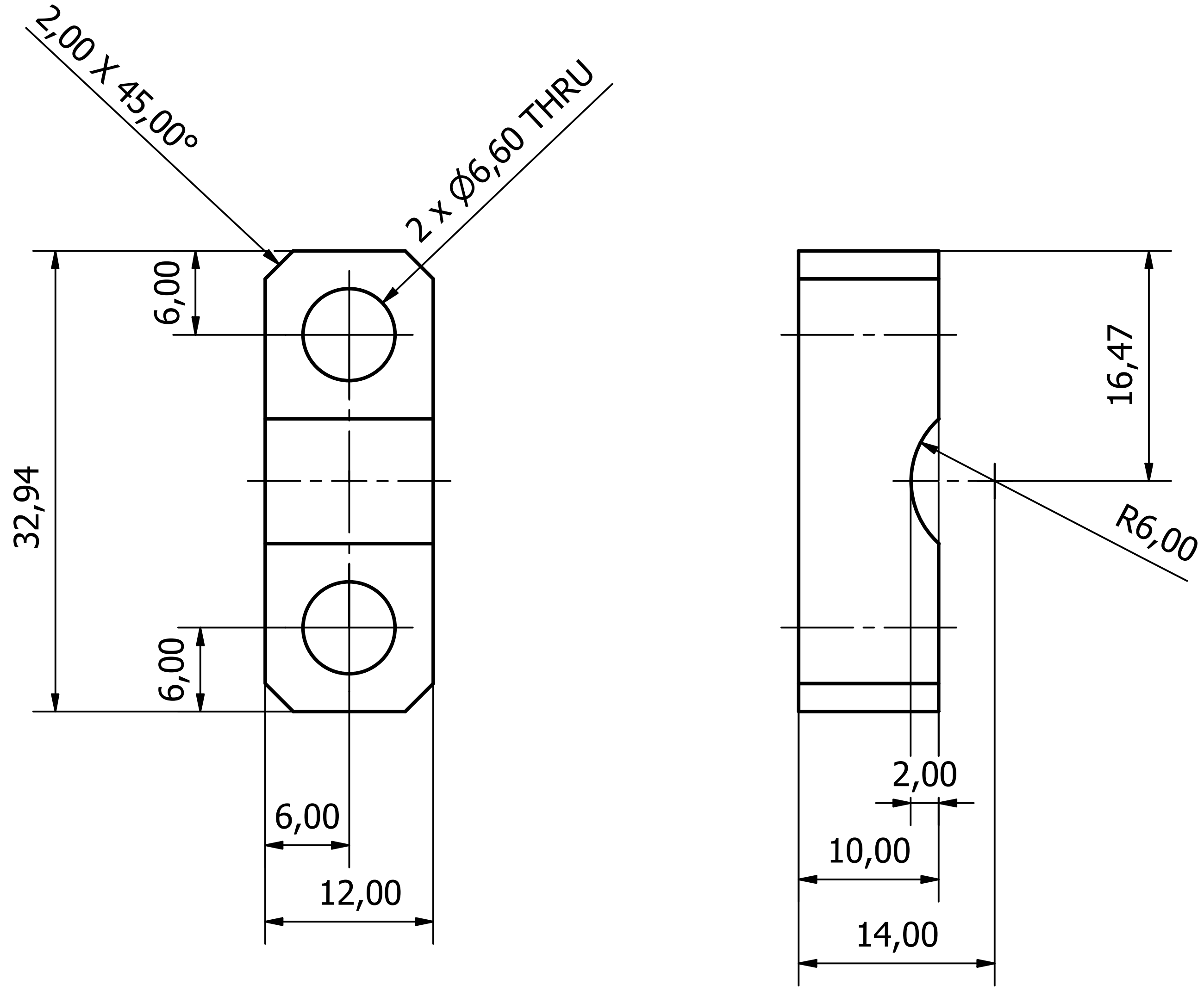
K-K (2 : 1)



Designed by	Quantity	Manufactured by	Material	Date
AlexM	8	Dale Flynn	ABS Plastic	07/03/2019

NELSON MANDELA UNIVERSITY

Suspension Stops		
Swerve Drive AGV	Edition	Sheet 51 / 207



All Chamfers are Approximate

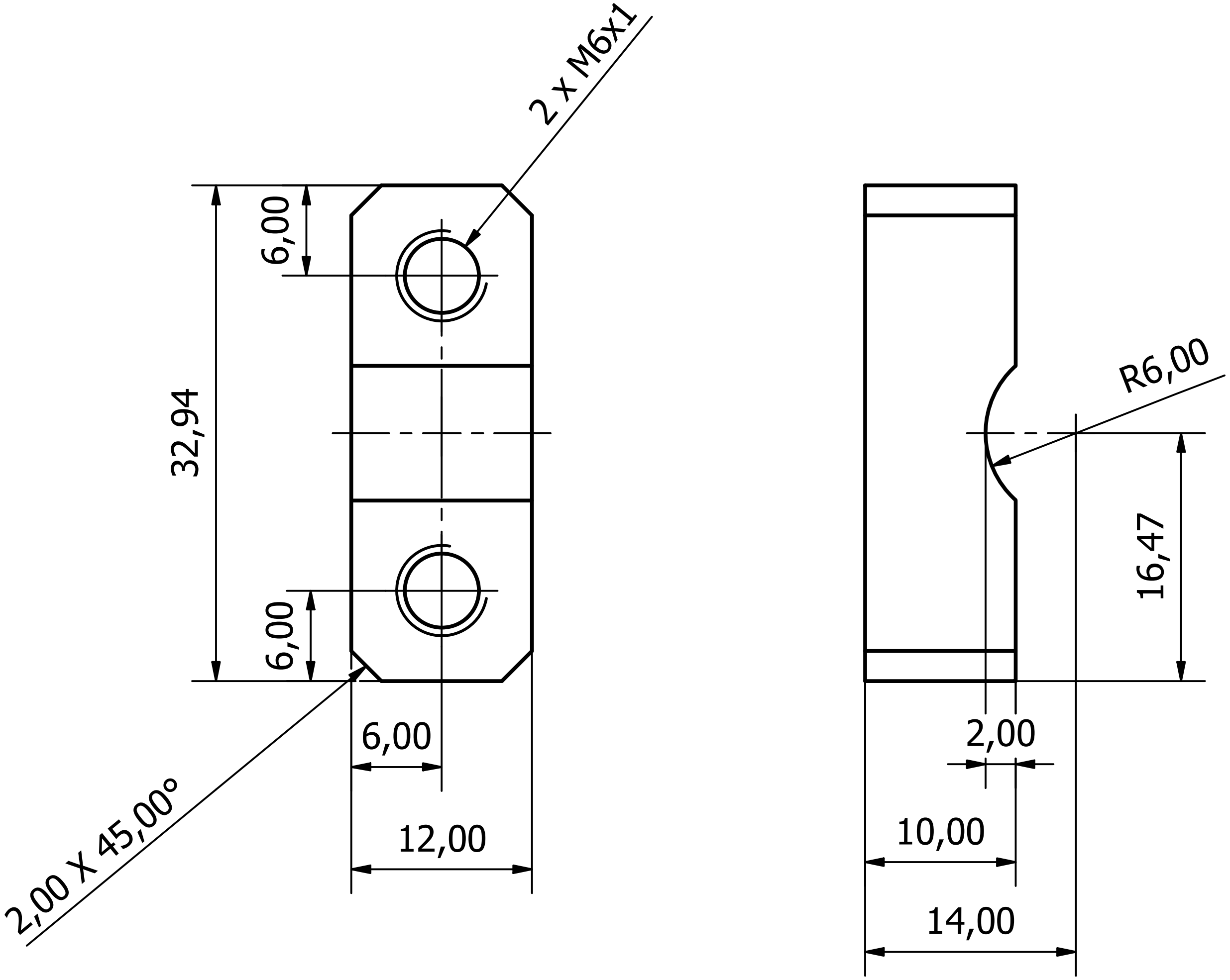
Designed by	Quantity	Manufactured by	Material	Date
AlexM	4	External	Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Tension Half Trough (Clearence) x 2H

Swerve Drive AGV

Edition	Sheet
	52 / 207



Designed by	Quantity	Manufactured by	Material	Date
AlexM	4	External	Aluminum 6061	07/03/2019

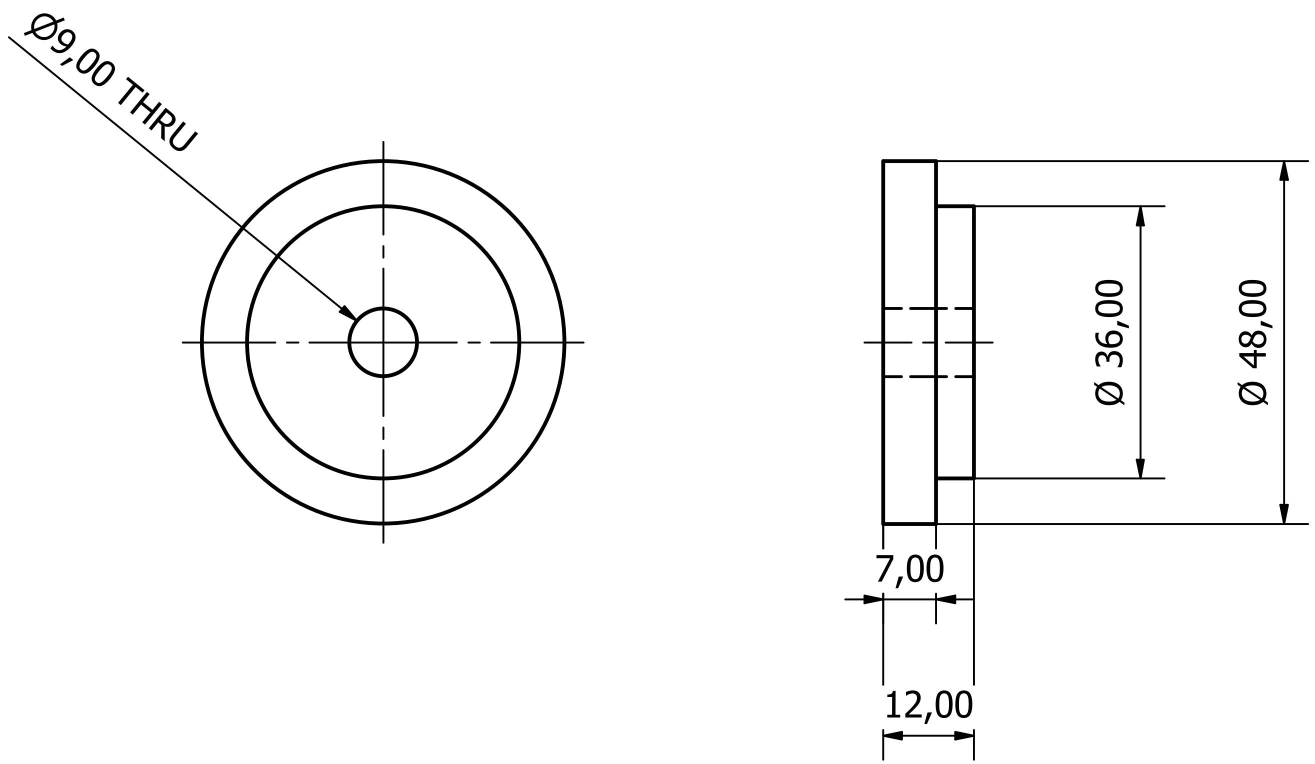
NELSON MANDELA UNIVERSITY

Tension Half Trough (Threaded) x 2H

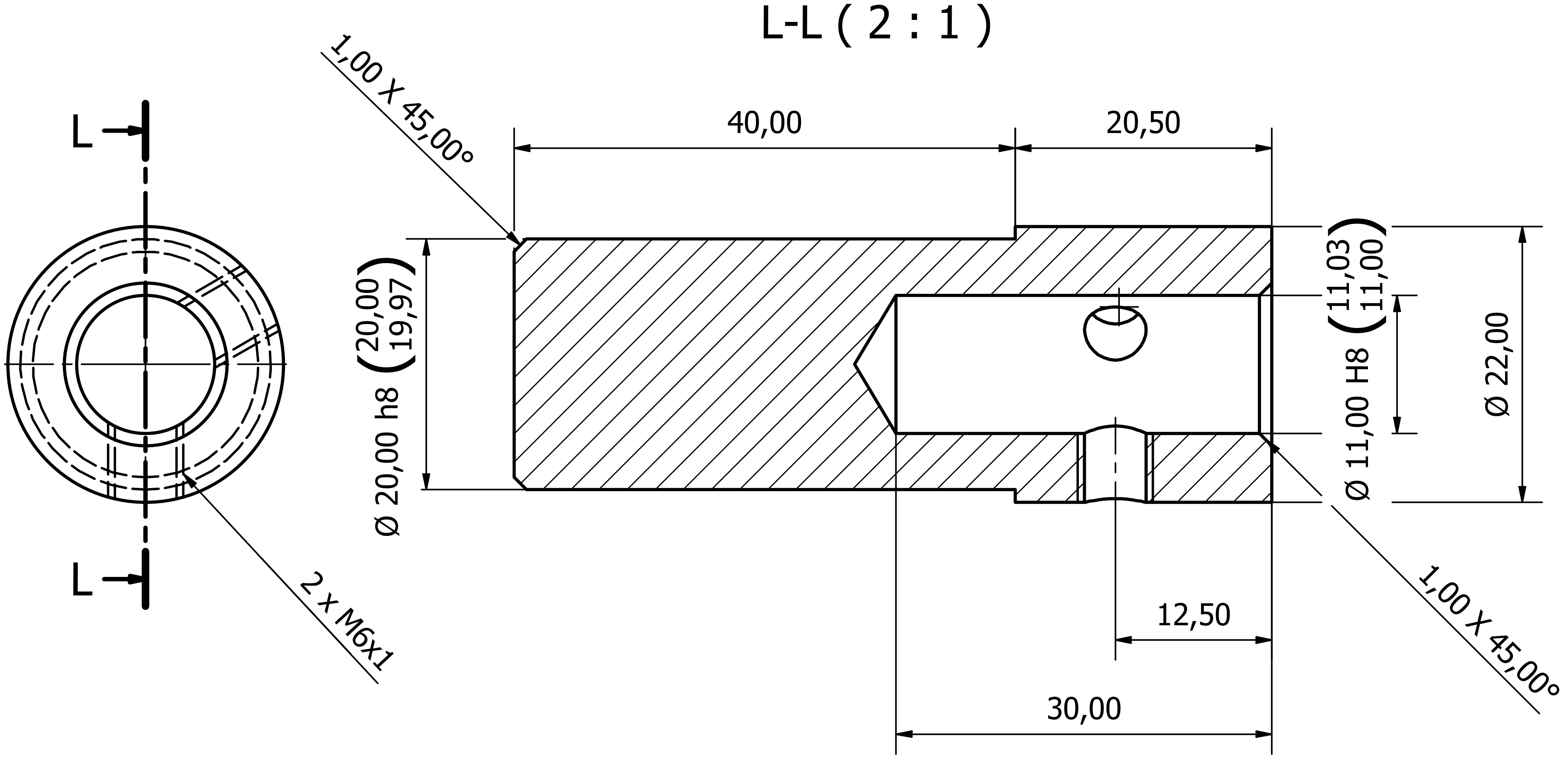
Swerve Drive AGV

Edition

Sheet 53 / 207



Designed by AlexM	Quantity 8	Manufactured by Dale Flynn	Material ABS Plastic	Date 07/03/2019
NELSON MANDELA UNIVERSITY		Spring Rest Bottom		
		Swerve Drive AGV	Edition	Sheet 54 / 207



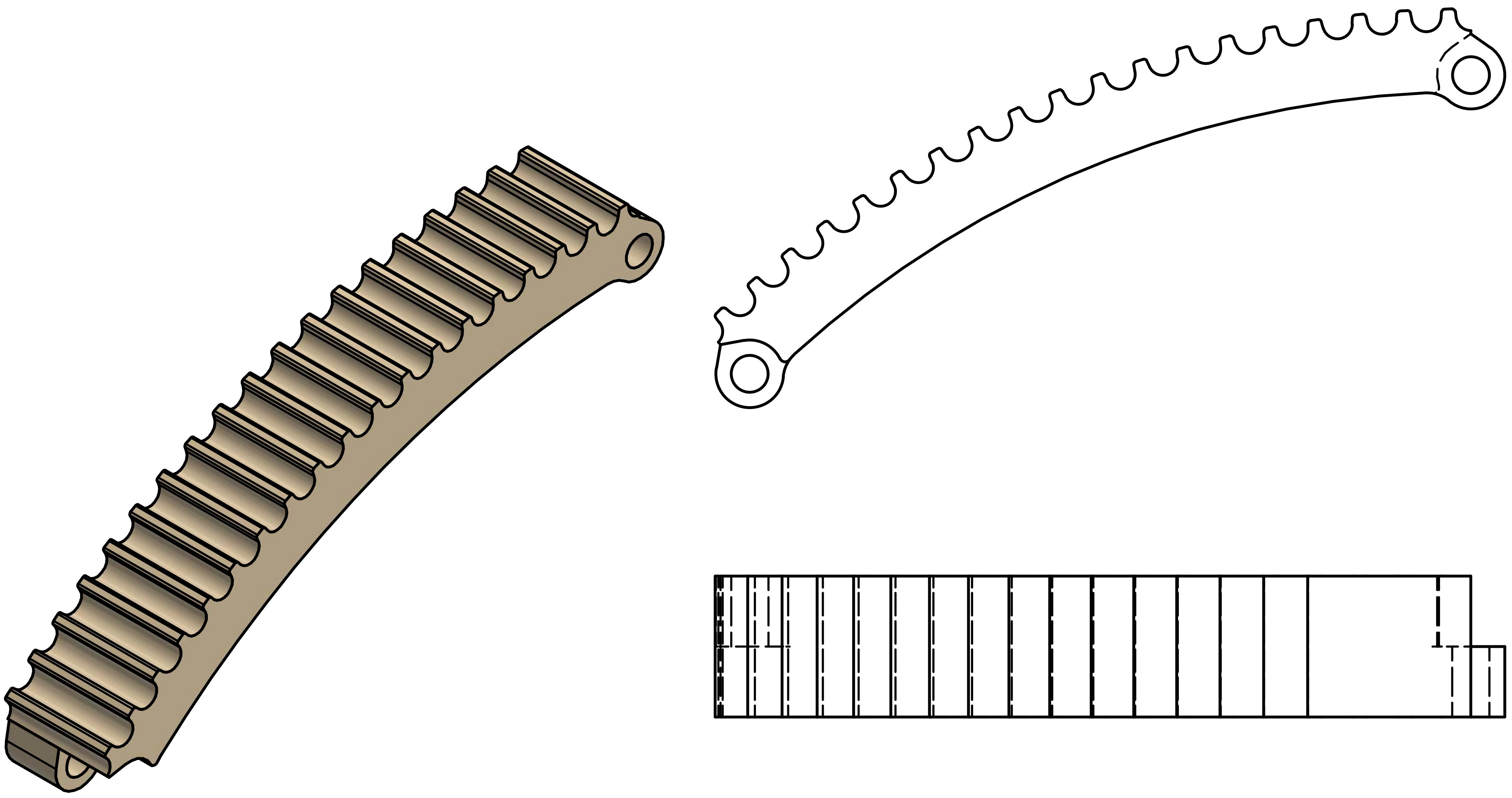
Note: Chamfers are approximate

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	Dale Flynn	Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Steering Pinion Converter

Swerve Drive AGV	Edition	Sheet
		55 / 207



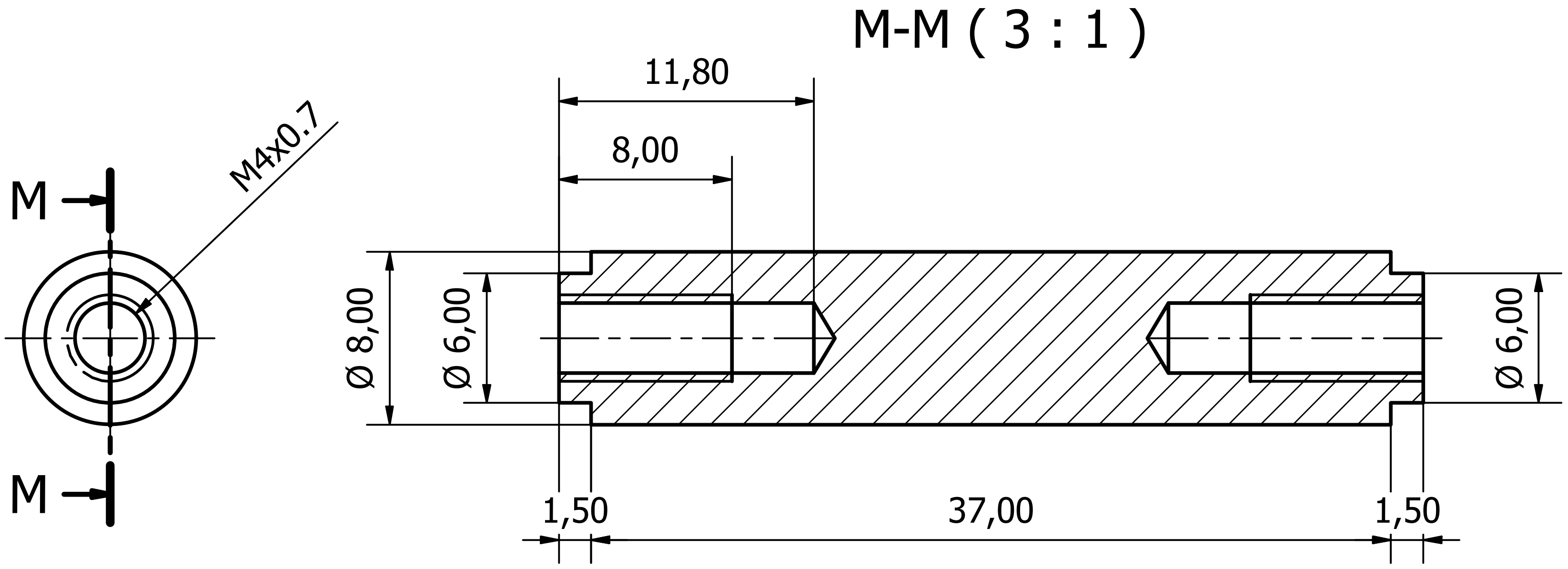
Designed by	Quantity	Manufactured by	Material	Date
AlexM	16	External	Polyethylene, High Density	07/03/2019

NELSON MANDELA UNIVERSITY

Ring Timing Belt Gear VER2

Swerve Drive AGV

Edition	Sheet
	56 / 207



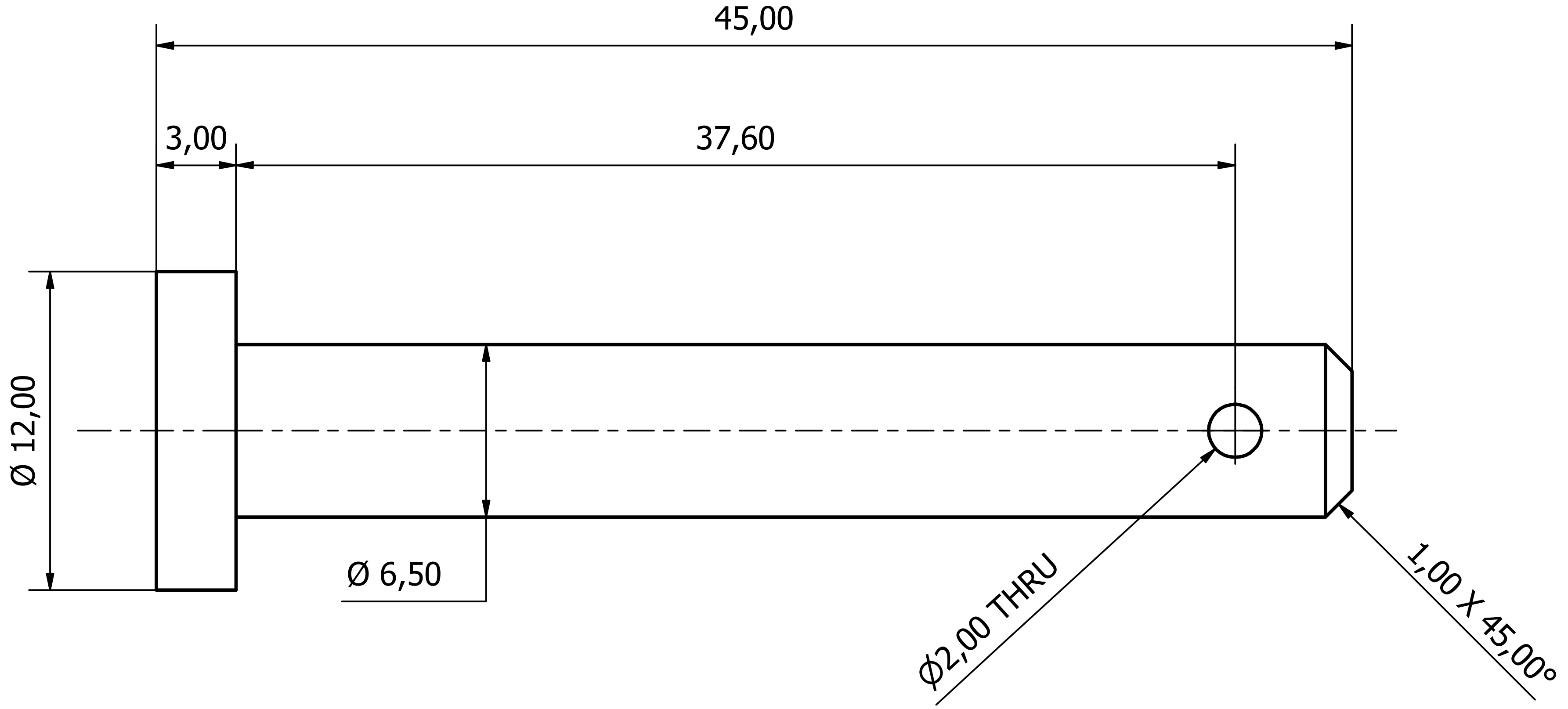
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	Alex Macfarlane	Steel, Mild	07/03/2019

NELSON MANDELA
UNIVERSITY

Bootlace Pin

Swerve Drive AGV

Edition	Sheet
	57 / 207



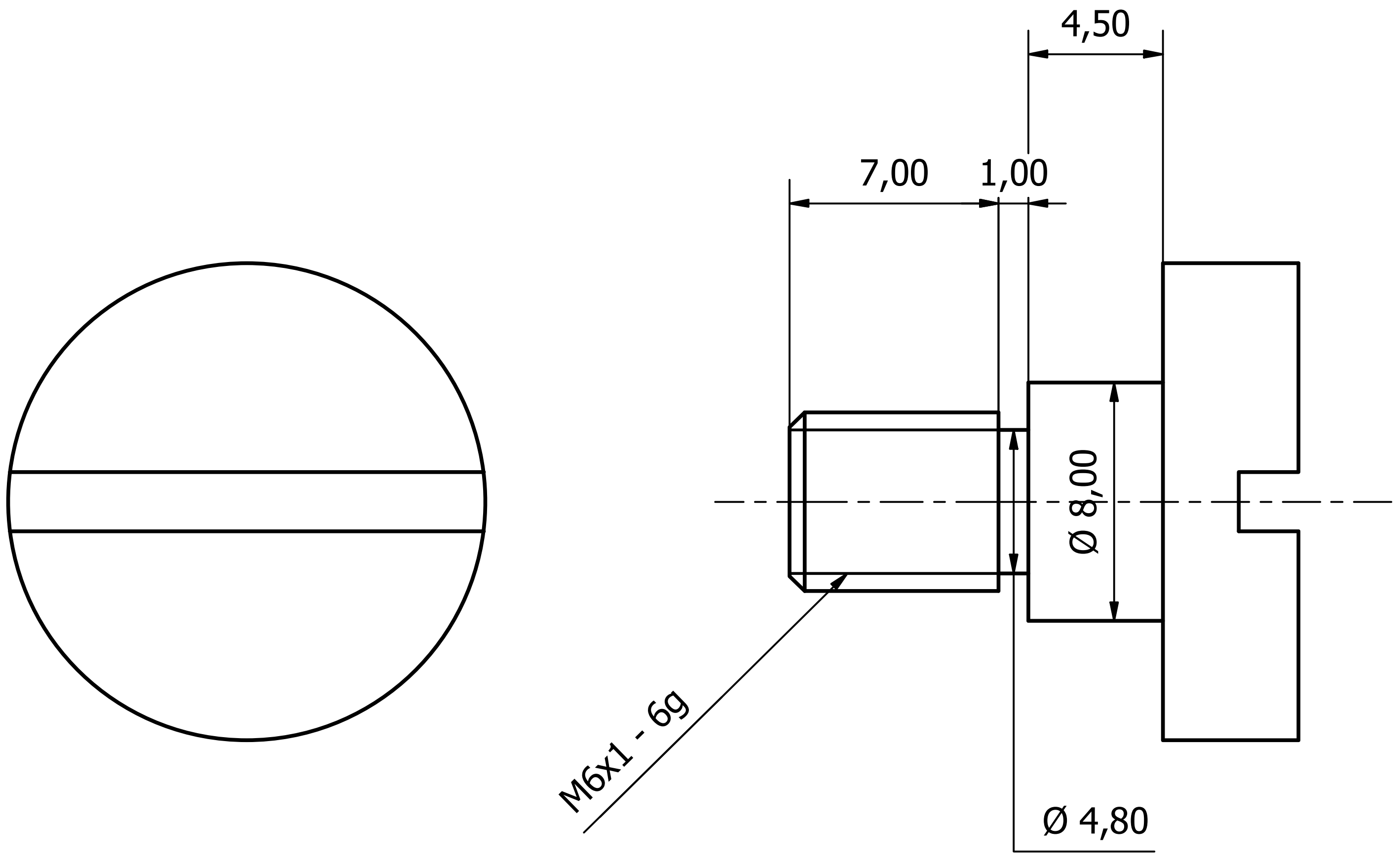
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	Alex Macfarlane	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Hinge Pin

	Edition	Sheet
Swerve Drive AGV		58 / 207

Made from machining down an ISO 1207 M10 x 50 standard machine screw

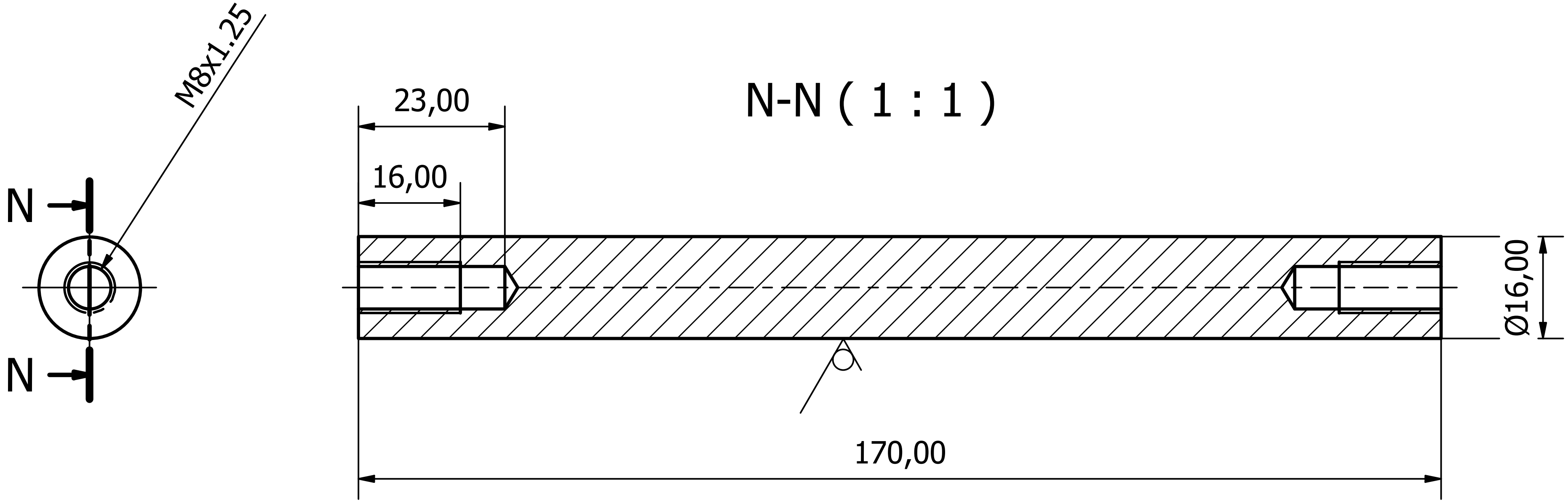


Designed by	Quantity	Manufactured by	Material	Date
AlexM	4	External	Steel, Mild	07/03/2019

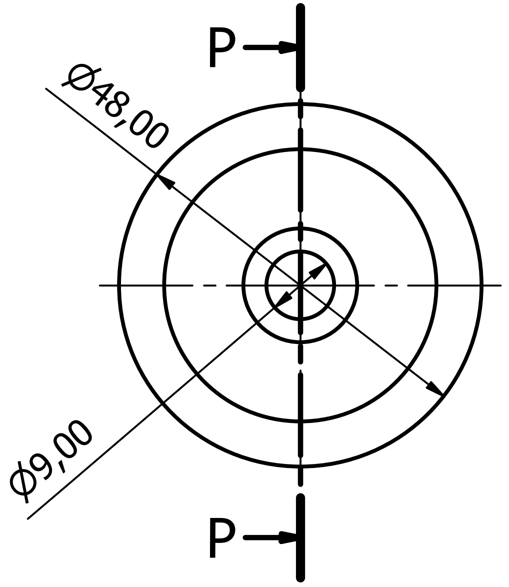
NELSON MANDELA UNIVERSITY

M6 Modified Bolt

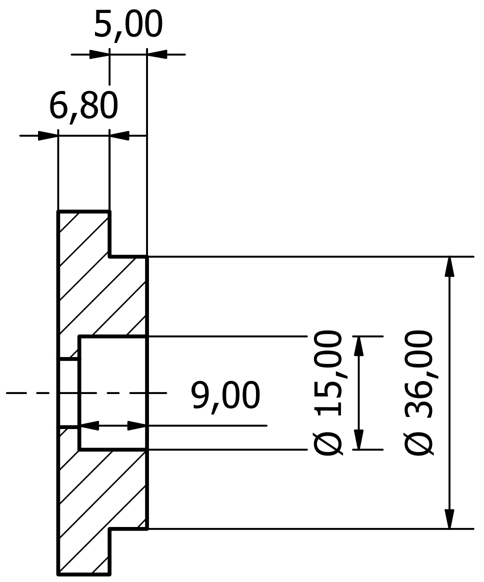
	Edition	Sheet
Swerve Drive AGV		59 / 207



Designed by AlexM	Quantity 8	Manufactured by Alex Macfarlane	Material Steel, Mild	Date 07/03/2019
NELSON MANDELA UNIVERSITY	Spacer Rods			
	Swerve Drive AGV		Edition	Sheet 60 / 207



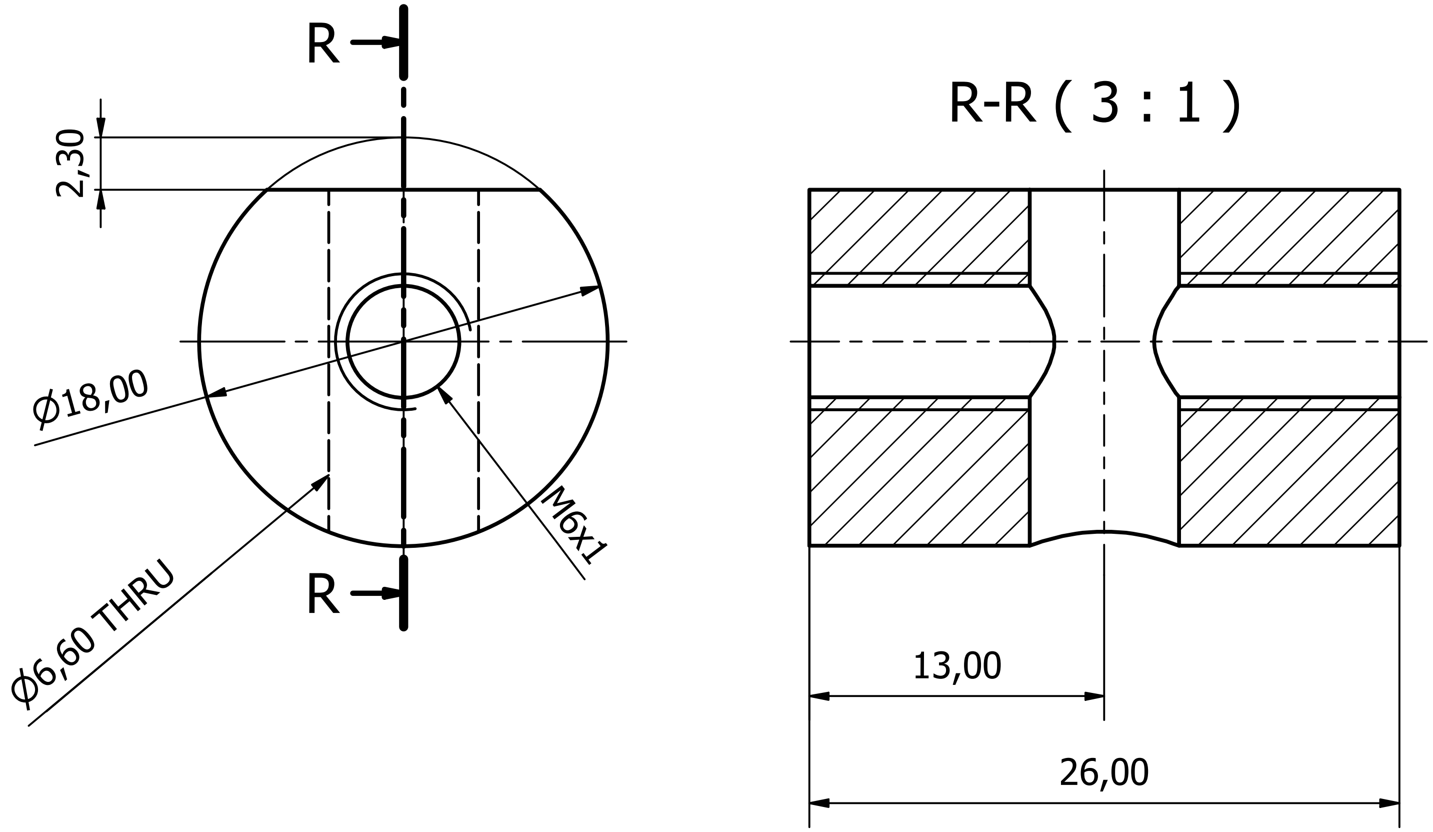
P-P (1 : 1)



Designed by	Quantity	Manufactured by	Material	Date
AlexM	8	Dale Flynn	ABS Plastic	07/03/2019

NELSON MANDELA UNIVERSITY

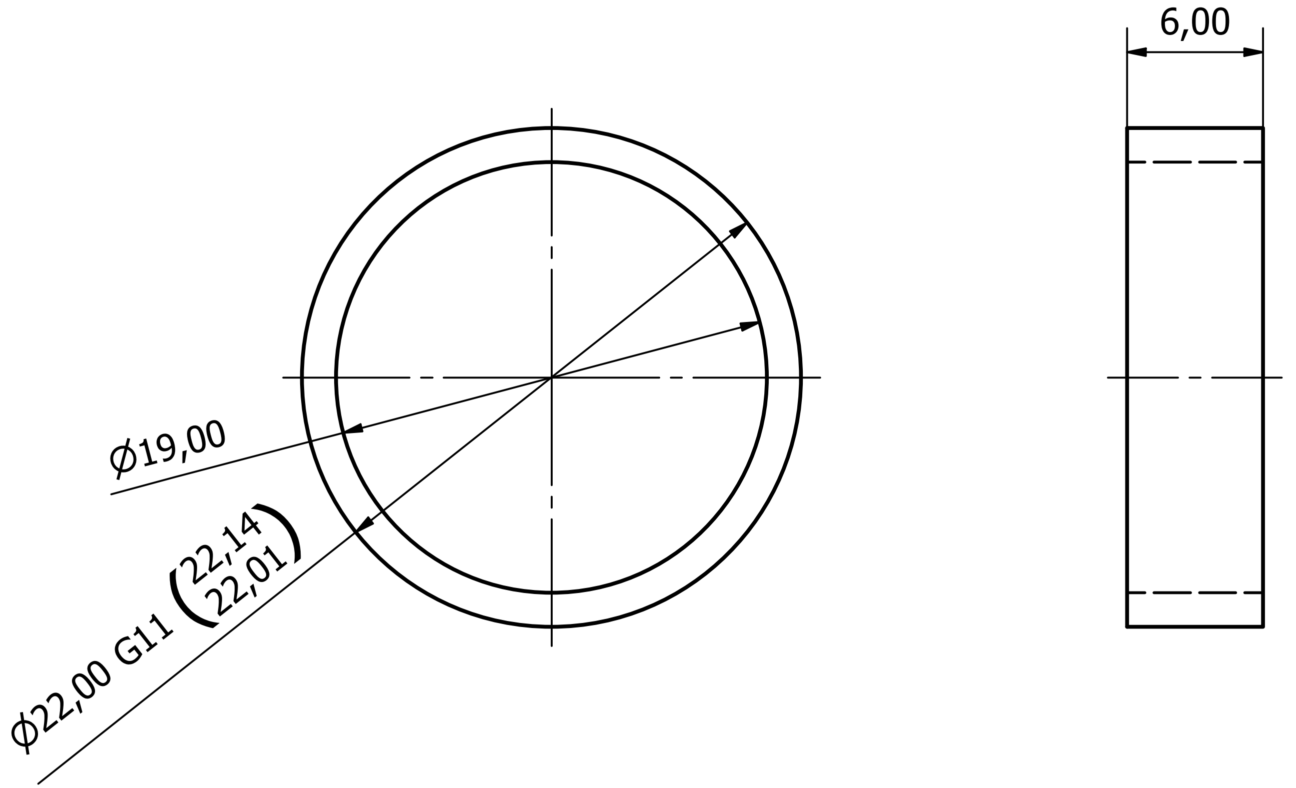
Spring Rest Top		
Swerve Drive AGV	Edition	Sheet 61 / 207



Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Tension Swivel		
Swerve Drive AGV	Edition	Sheet 62 / 207

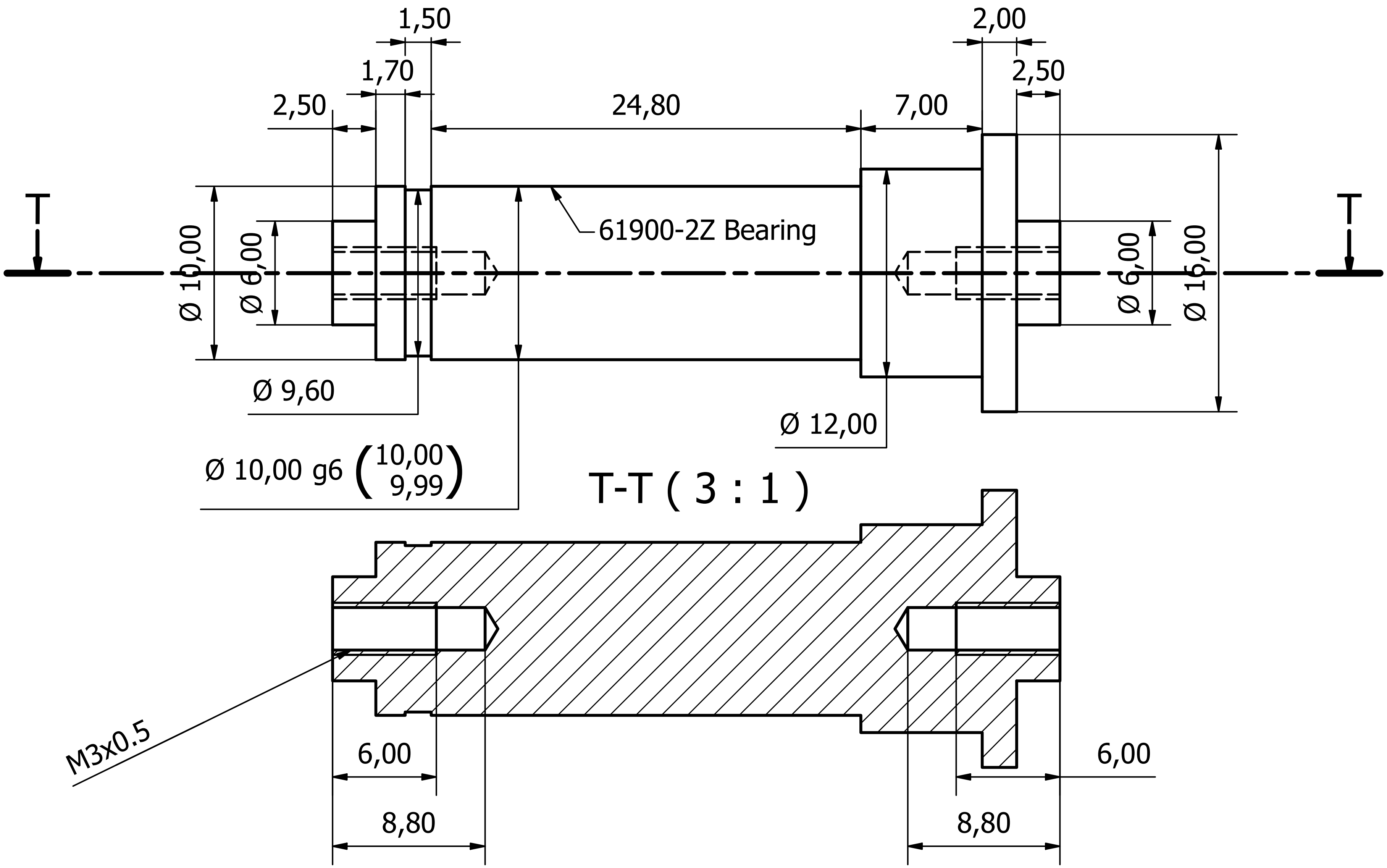


Designed by AlexM	Quantity 2	Manufactured by External	Material Aluminum 6061	Date 07/03/2019

NELSON MANDELA UNIVERSITY

Upper Smooth Idler Pulley Bearing Spacer

Swerve Drive AGV	Edition	Sheet 63 / 207

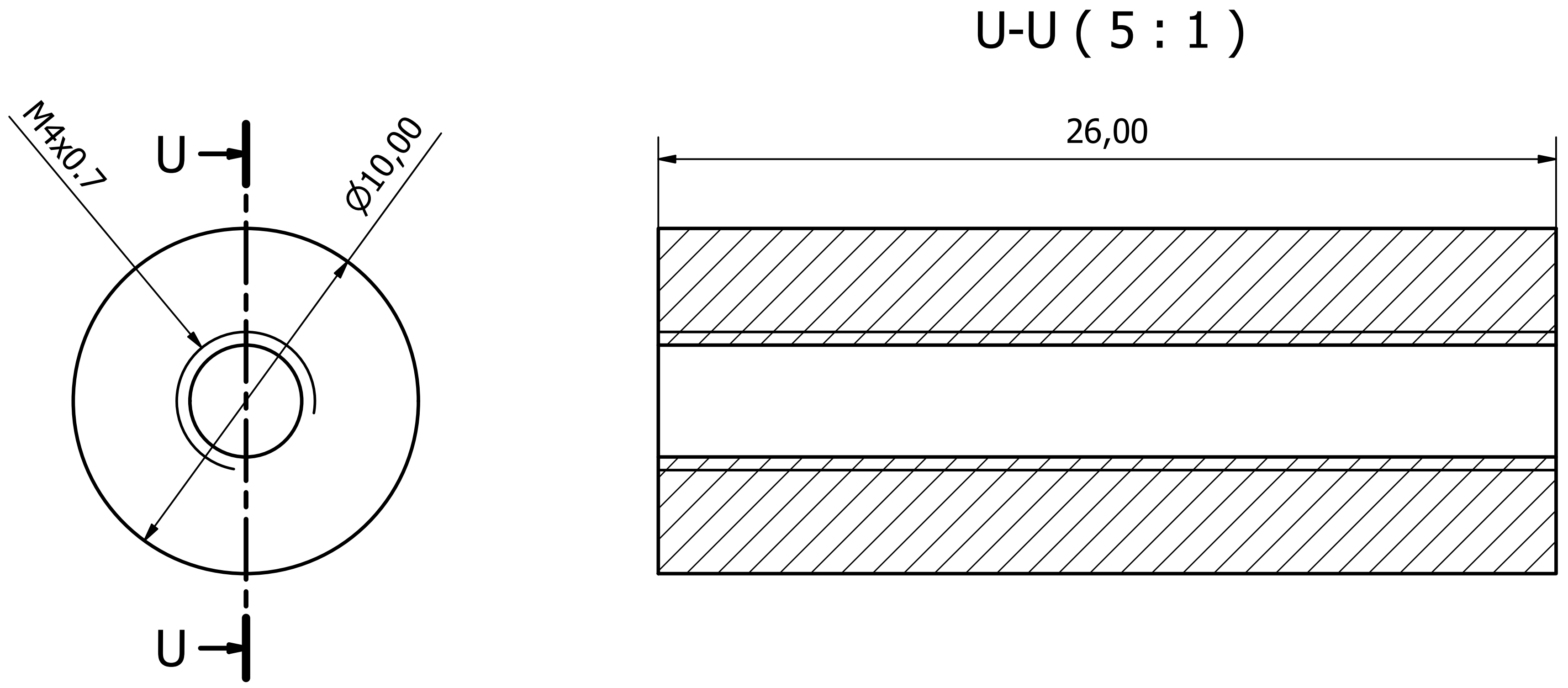


Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	En 8 Steel	07/03/2019

NELSON MANDELA UNIVERSITY

Upper Smooth Idler Pulley Shaft

	Edition	Sheet
Swerve Drive AGV		64 / 207

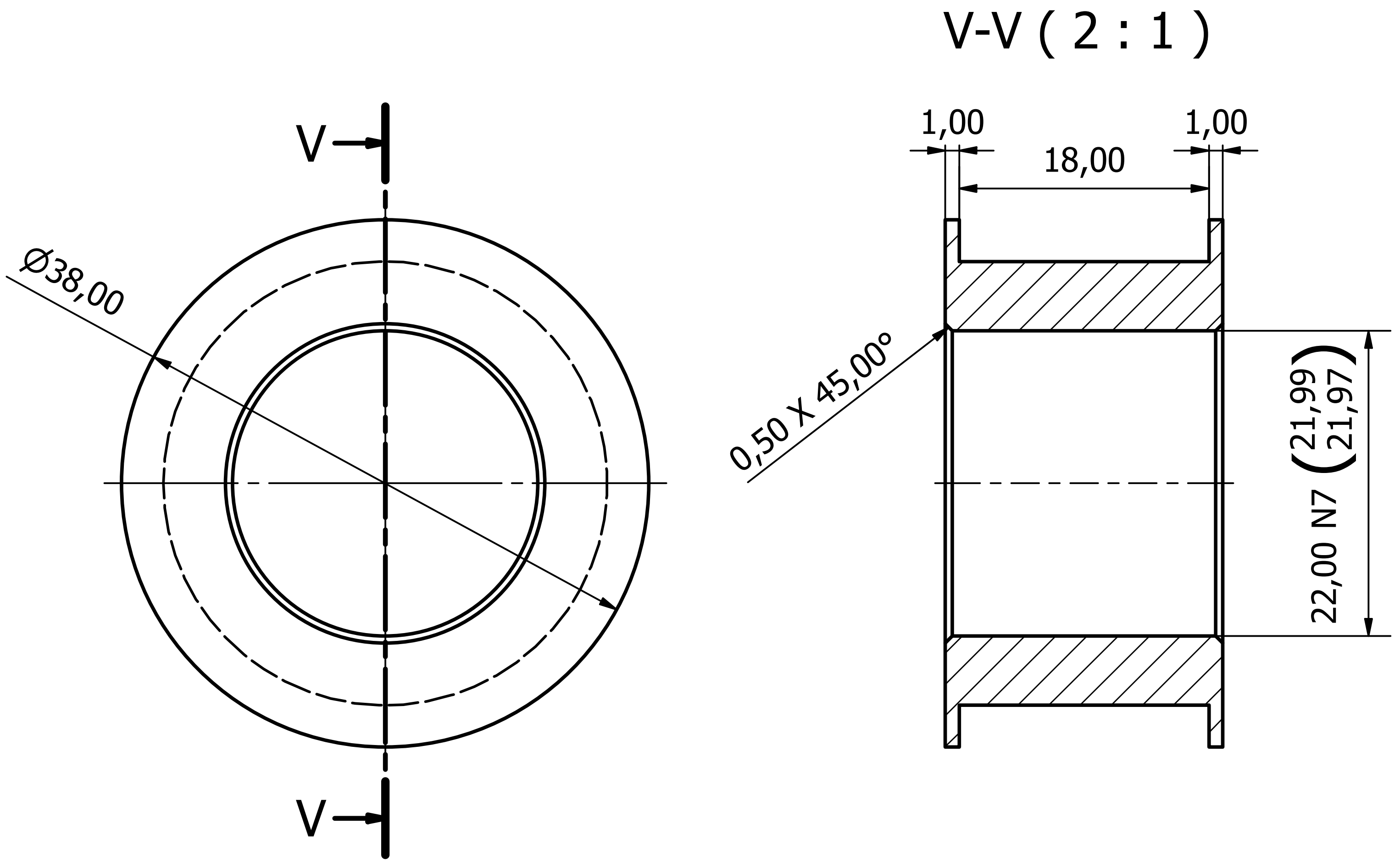


Designed by	Quantity	Manufactured by	Material	Date
AlexM	8	Alex Macfarlane	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Upper Smooth Idler Pulley Spacers

	Edition	Sheet
Swerve Drive AGV		65 / 207



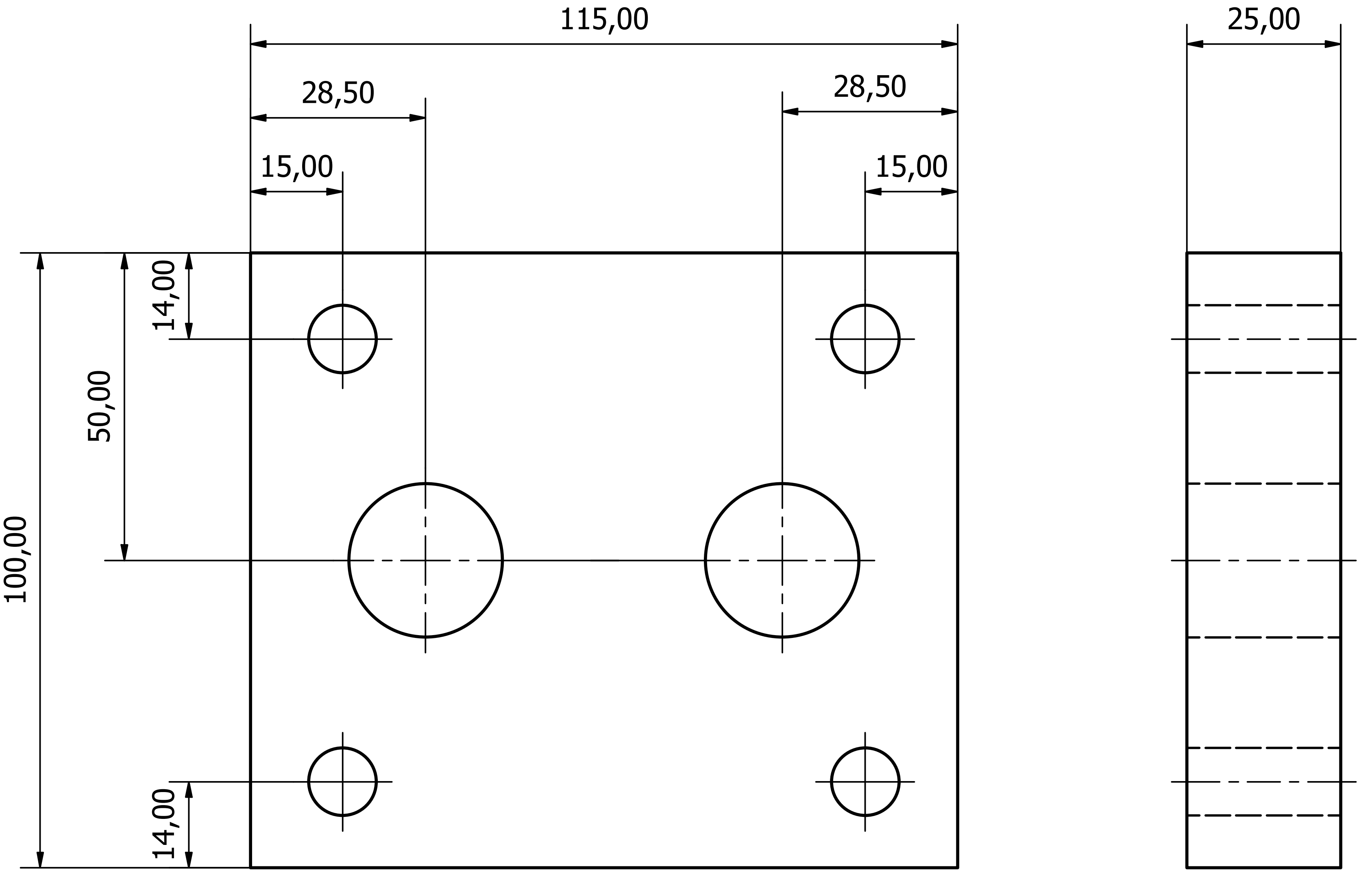
Note: Chamfers are approximate

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Upper Smooth Idler Pulley

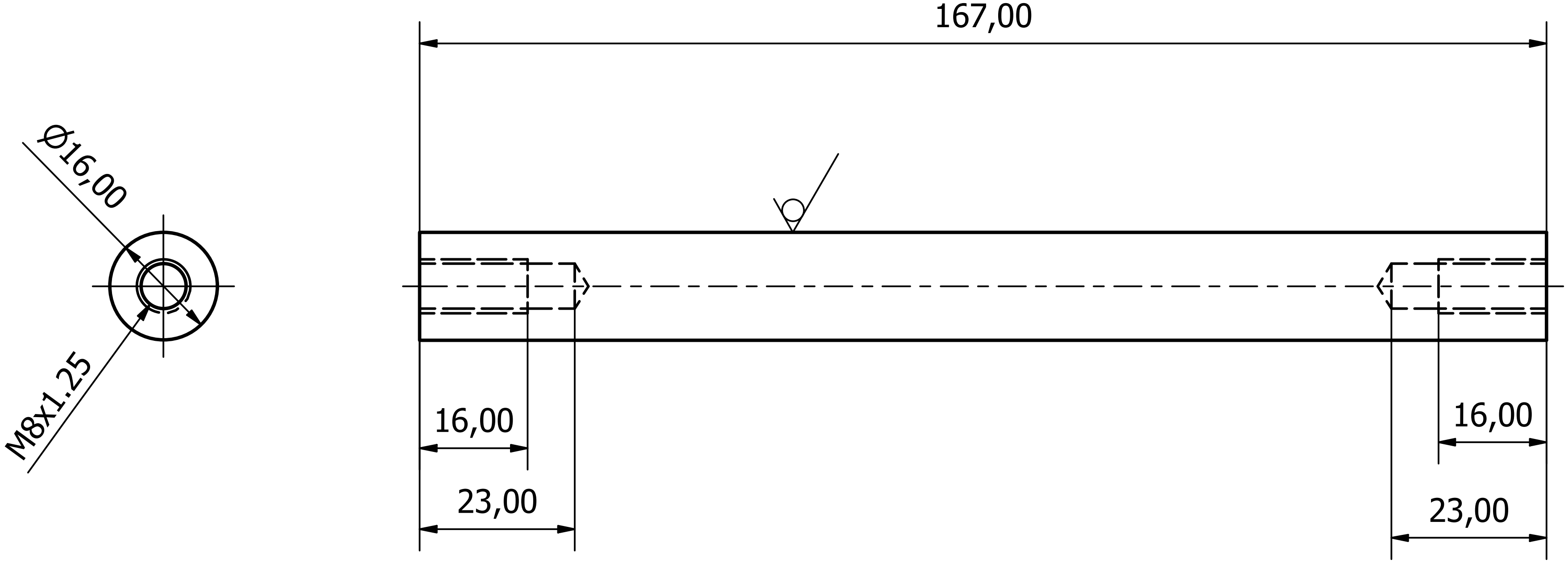
	Edition	Sheet
Swerve Drive AGV		66 / 207



Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Polyethylene, High Density	07/03/2019

NELSON MANDELA UNIVERSITY

Caster Unit Spacer Block		
Swerve Drive AGV	Edition	Sheet 67 / 207



Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	Alex Macfarlane	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

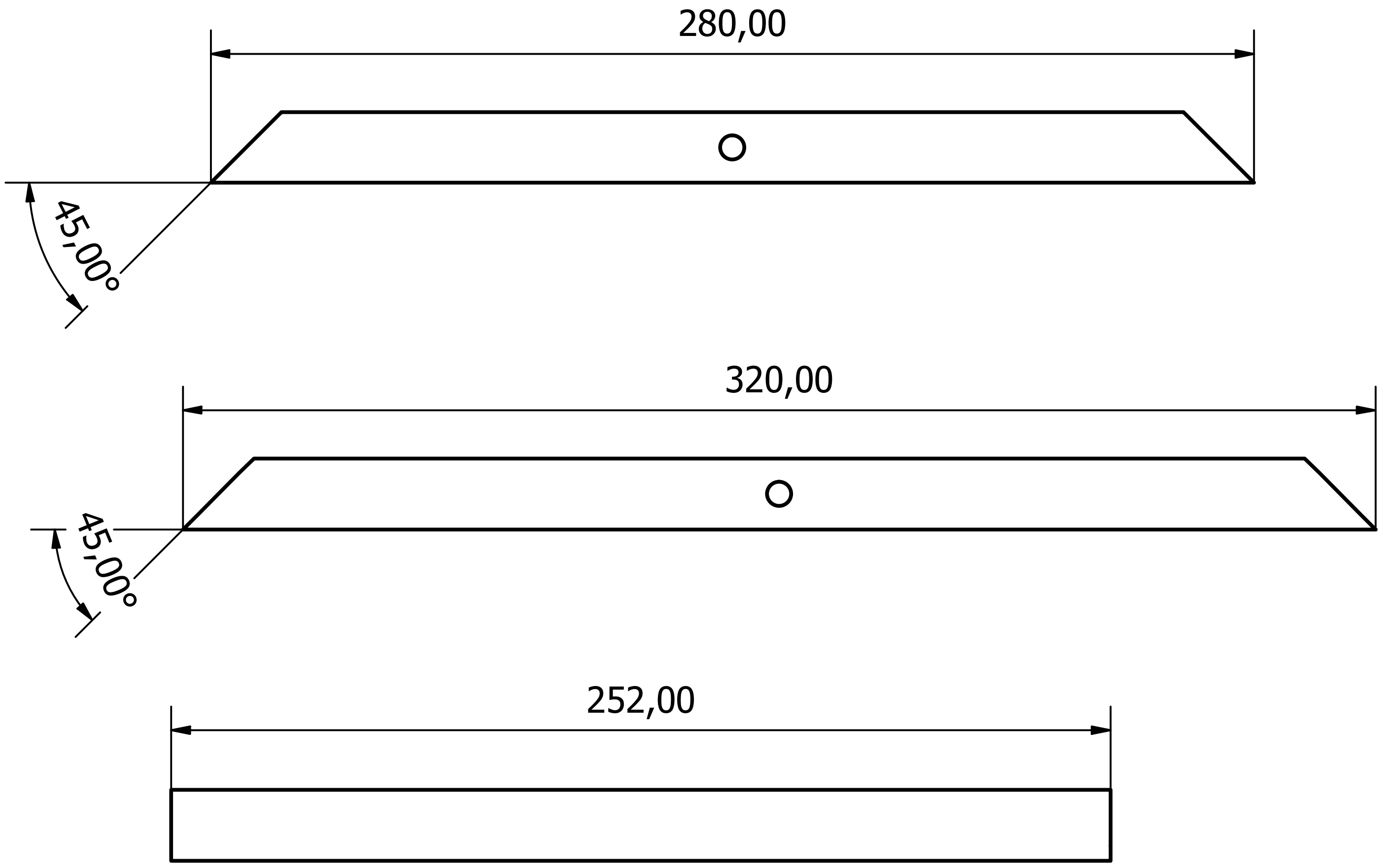
Caster Unit Spacer Rod

Swerve Drive AGV	Edition	Sheet
		68 / 207

SABS 19x19x2 000010
SABS 19x19x2 000012

SABS 19x19x2 000011
SABS 19x19x2 000013

SABS 19x19x2 000014
SABS 19x19x2 000015
SABS 19x19x2 000016
SABS 19x19x2 000017
SABS 19x19x2 000018



Square Tube 19mm x 2mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	Alex Macfarlane	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

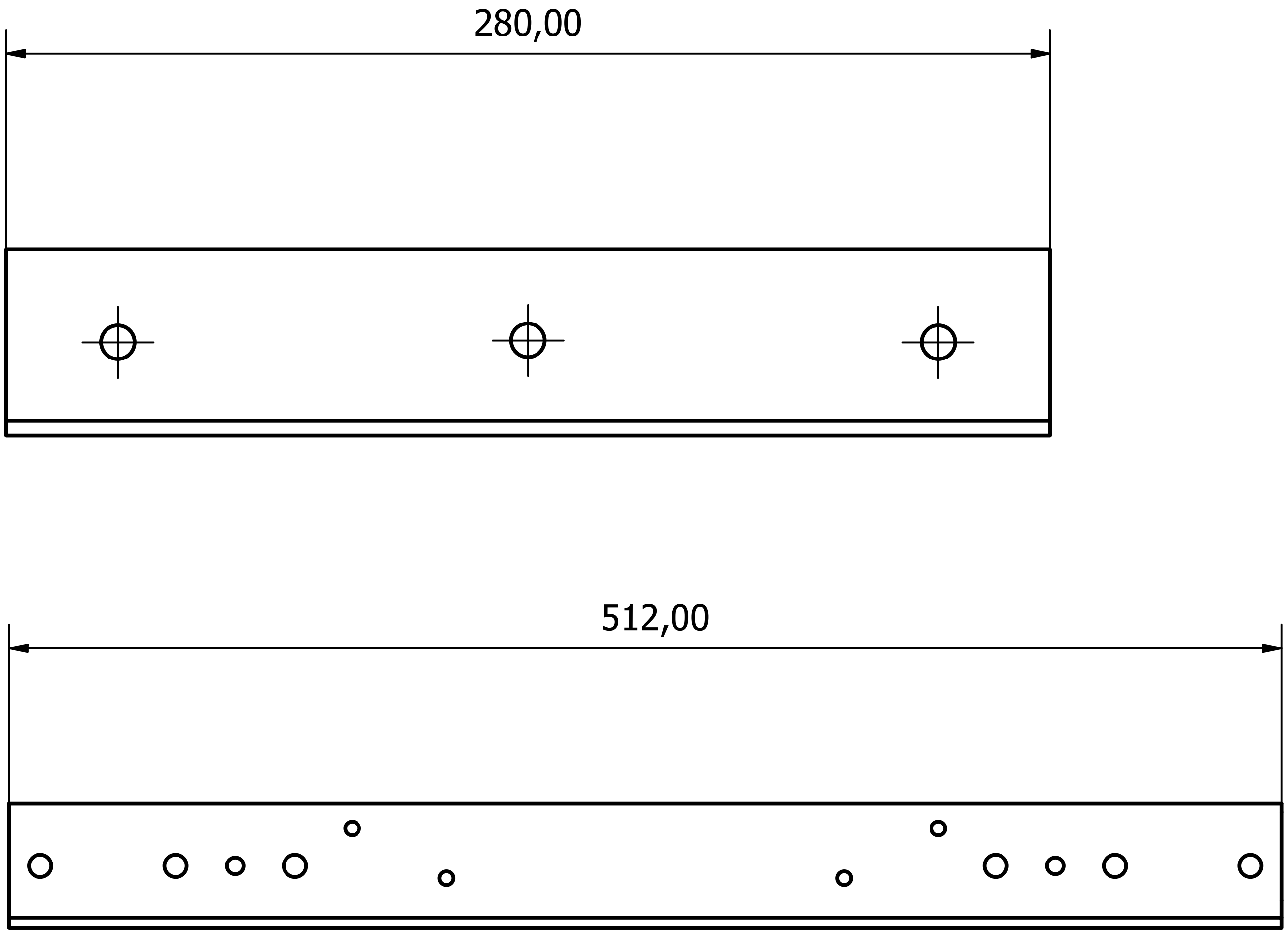
Drive Unit: Square Sections

Swerve Drive AGV

Edition	Sheet
	69 / 207

ISO L50x50x4 00008
ISO L50x50x4 00009

ISO L50x50x4 000010
ISO L50x50x4 000011



Equal Angle Iron 50mm x 4mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	Alex Macfarlane	Steel, Mild	07/03/2019

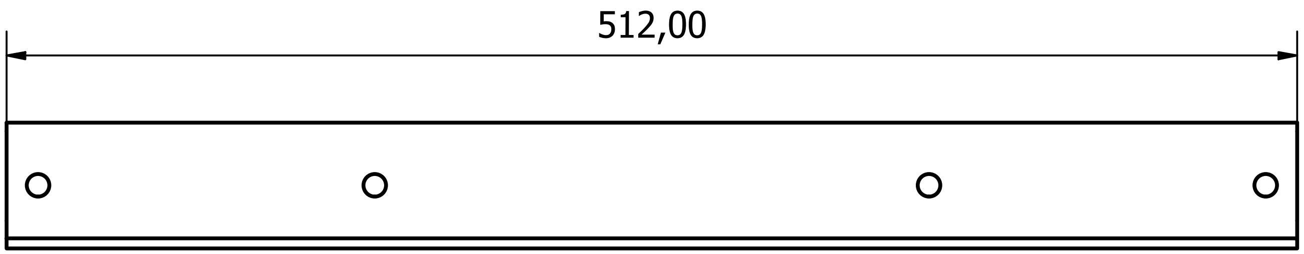
NELSON MANDELA UNIVERSITY

Drive Unit: Angle Sections

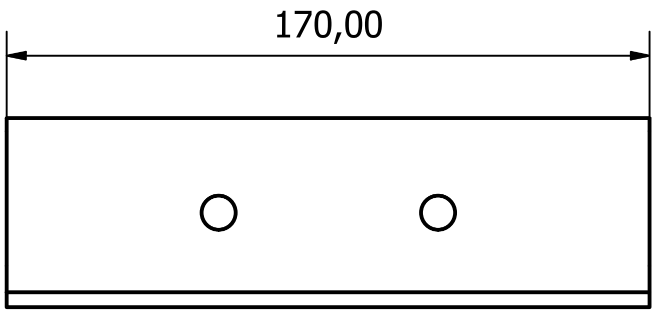
Swerve Drive AGV

Edition	Sheet
	70 / 207

ISO L50x50x4 00001
ISO L50x50x4 00002



ISO L50x50x4 000002



Equal Angle Iron 50mm x 4mm

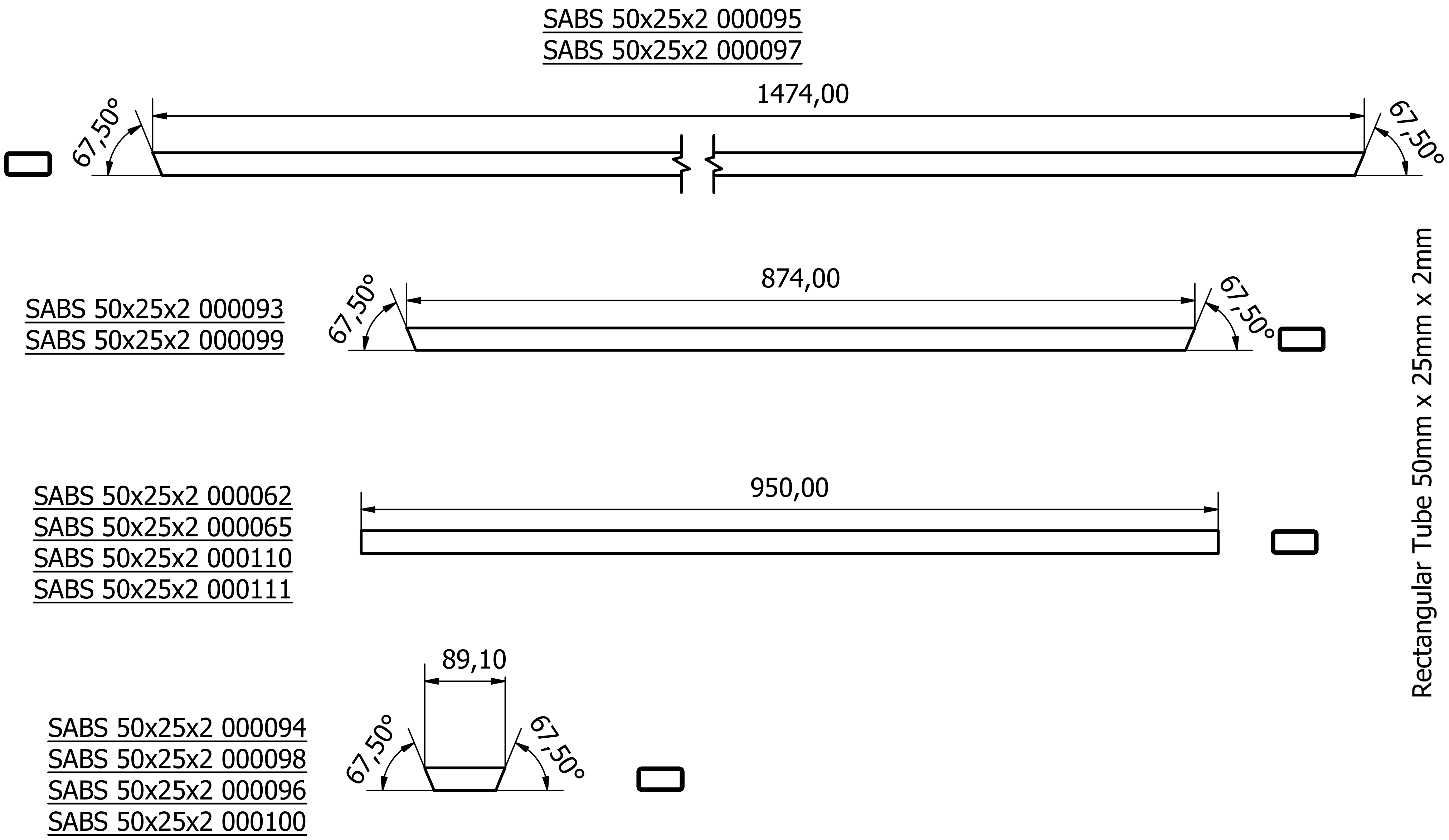
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	Alex Macfarlane	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Caster Unit: Angle Sections

Swerve Drive AGV

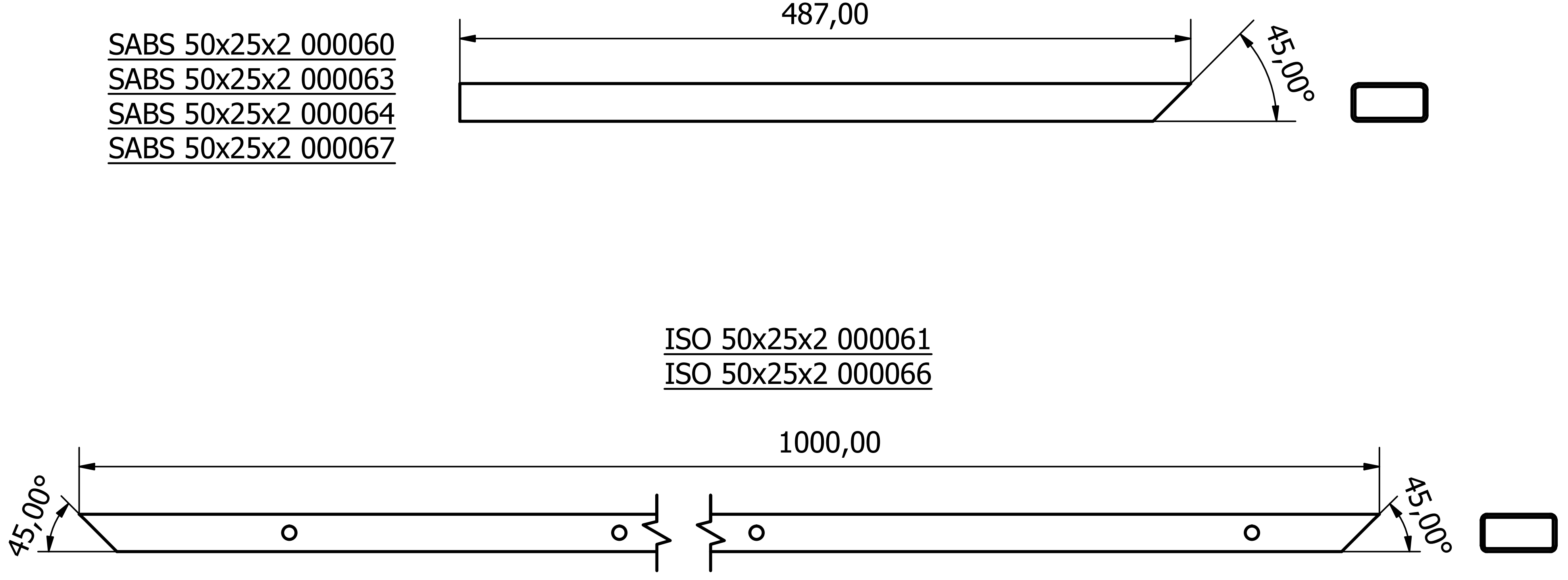
Edition	Sheet
	71 / 207



Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	Alex Macfarlane	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Body: 50x25 Rectangular Sections 1		
Swerve Drive AGV	Edition	Sheet 72 / 207



SABS 50x25x2 000060
SABS 50x25x2 000063
SABS 50x25x2 000064
SABS 50x25x2 000067

ISO 50x25x2 000061
ISO 50x25x2 000066

Rectangular Tube 50mm x 25mm x 2mm

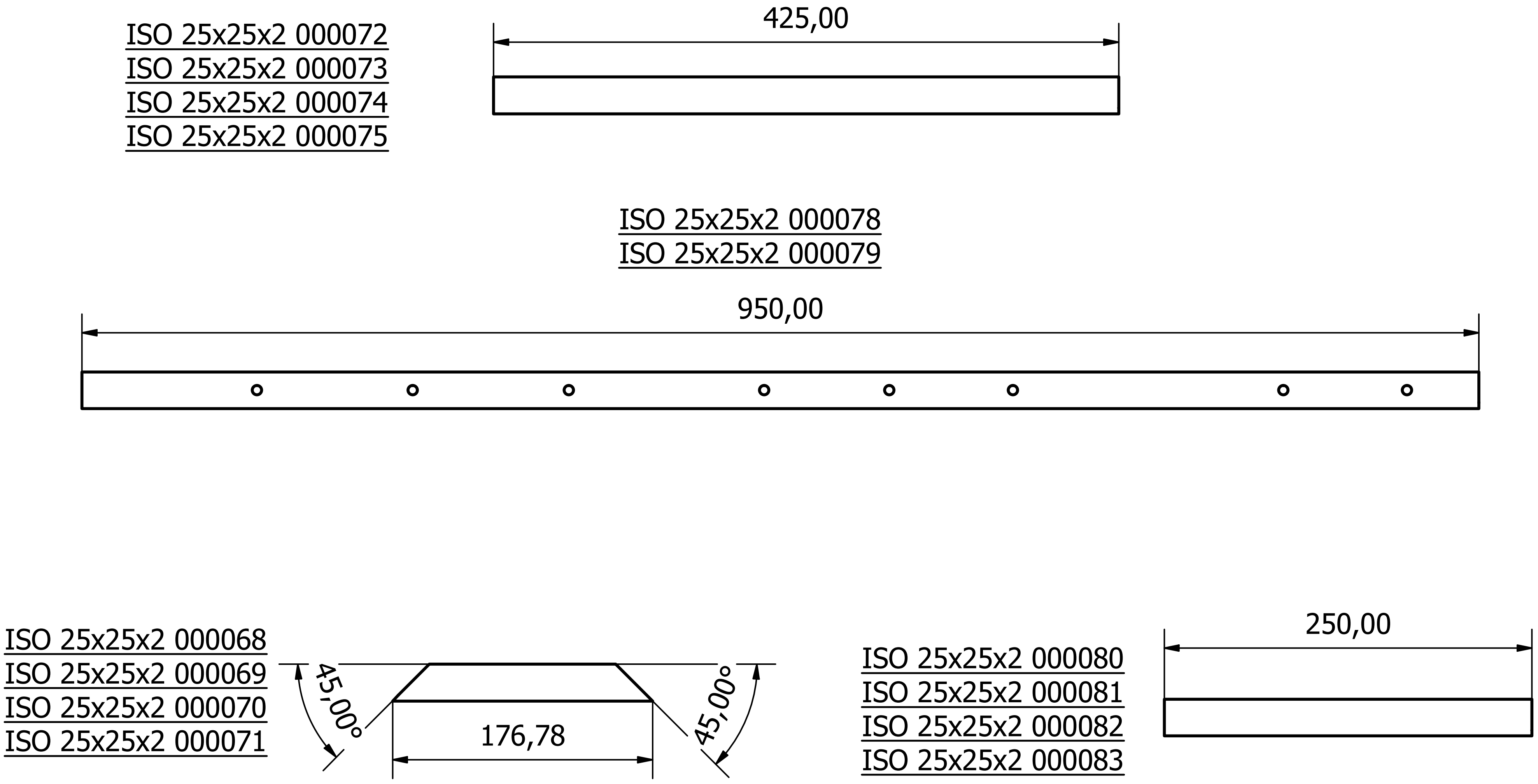
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	Alex Macfarlane	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Body: 50x25 Rectangular Sections 2

Swerve Drive AGV

Edition	Sheet
	73 / 207



Square Tube 25mm x 25mm x 2mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	Alex Macfarlane	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

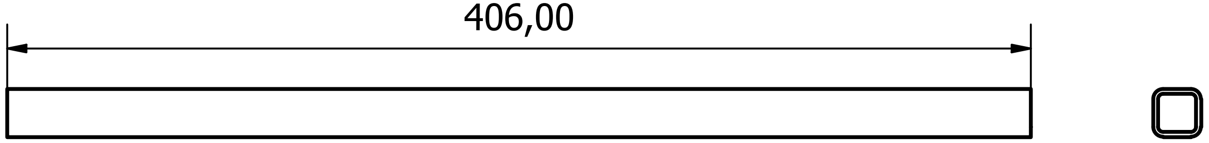
Body: 25x25 Square Sections

Swerve Drive AGV

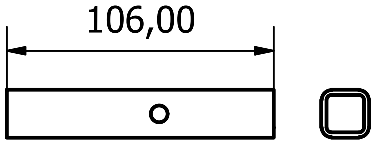
Edition

Sheet 74 / 207

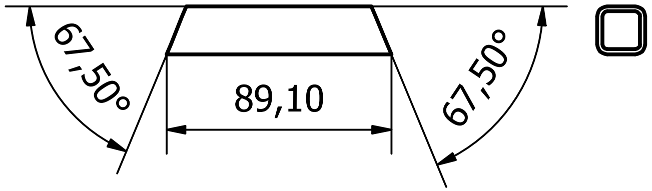
SABS 19x19x2 000091
SABS 19x19x2 000092
SABS 19x19x2 000108
SABS 19x19x2 000109



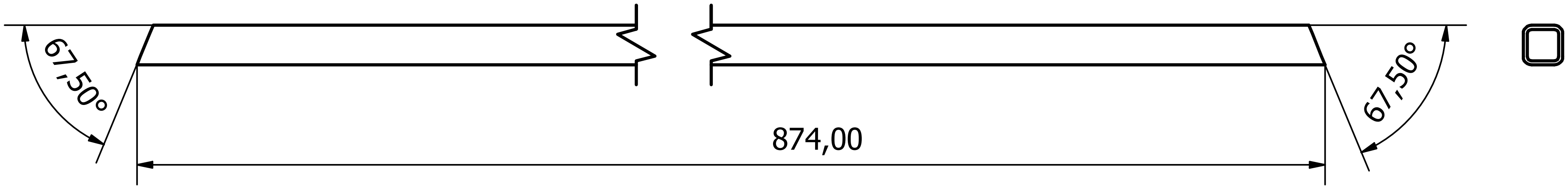
SABS 19x19x2 000089
SABS 19x19x2 000090
SABS 19x19x2 000106
SABS 19x19x2 000107



SABS 19x19x2 000085
SABS 19x19x2 000087
SABS 19x19x2 000102
SABS 19x19x2 000104



SABS 19x19x2 000086
SABS 19x19x2 000103



Square Tube 19mm x 19mm x 2mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	Alex Macfarlane	Steel, Mild	07/03/2019

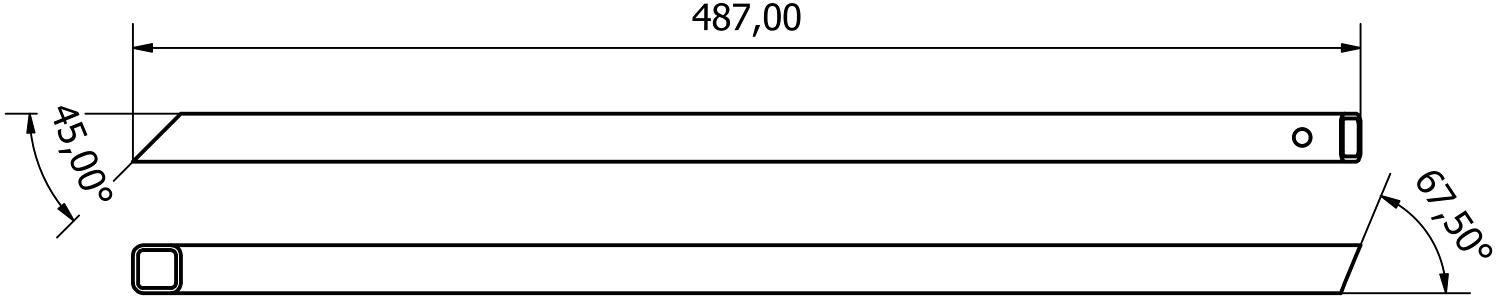
NELSON MANDELA UNIVERSITY

Body: 19x19 Square Sections 1

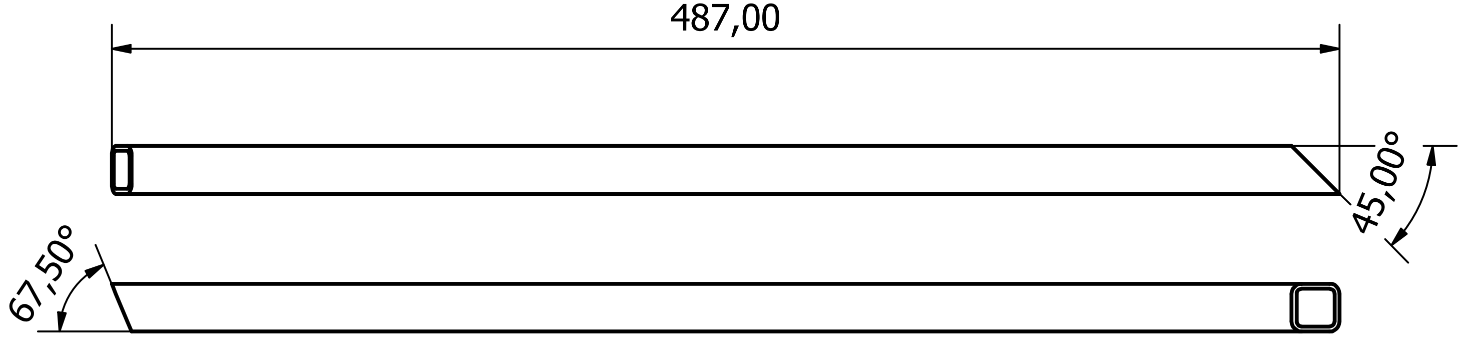
Swerve Drive AGV

Edition	Sheet
	75 / 207

SABS 19x19x2 000088
SABS 19x19x2 000101



SABS 19x19x2 000084
SABS 19x19x2 000105



Square Tube 19mm x 19mm x 2mm

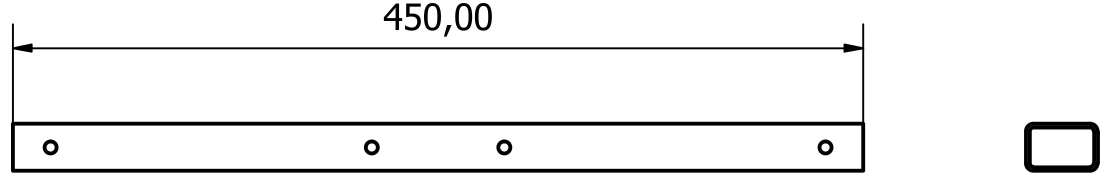
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	Alex Macfarlane	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Body: 19x19 Square Sections 2

	Edition	Sheet
Swerve Drive AGV		76 / 207

SABS 38x25x2 000114
SABS 38x25x2 000115



Square Tube 38mm x 25mm x 2mm

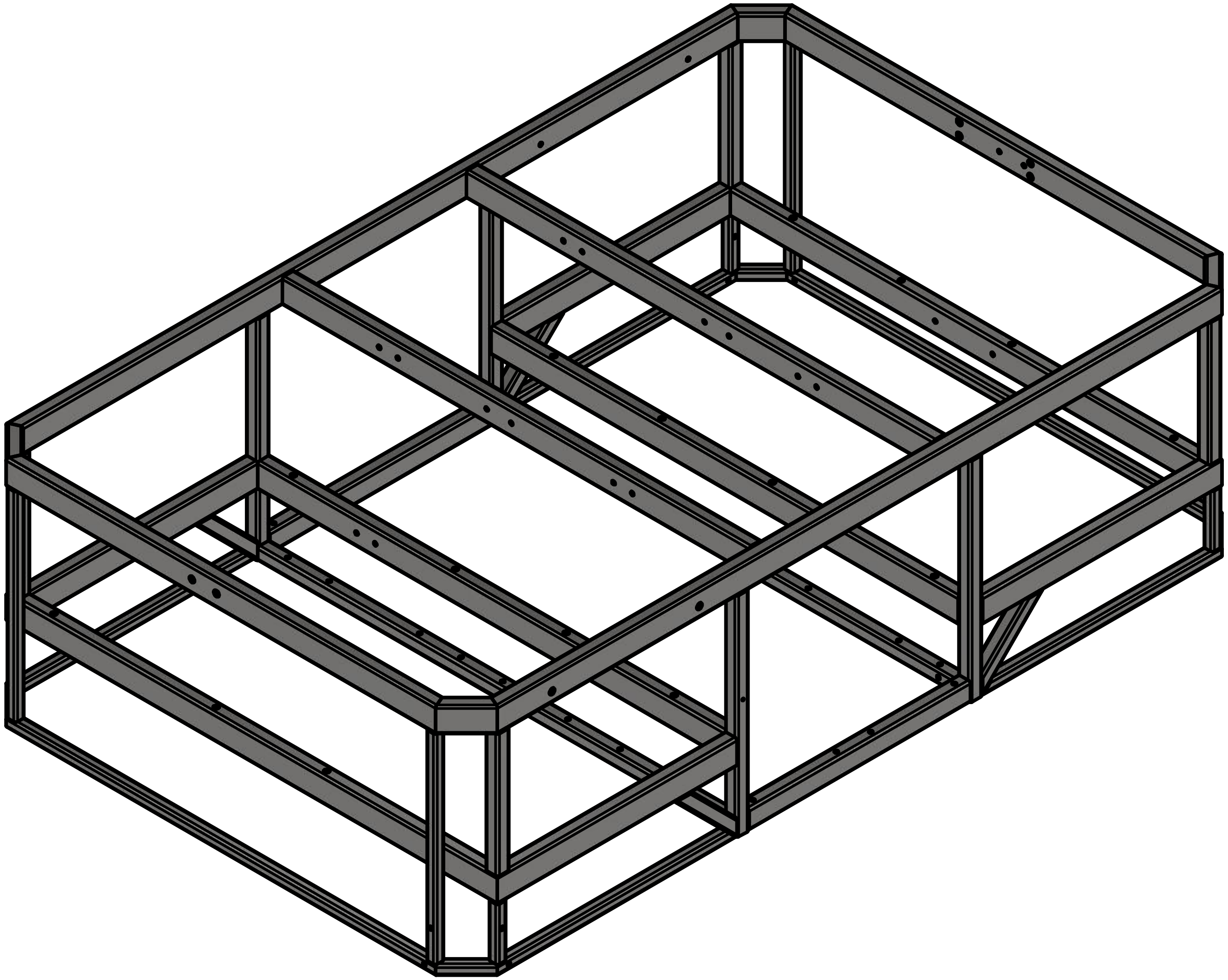
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	Alex Macfarlane	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Body: 38x25 Rectangular Sections

Swerve Drive AGV

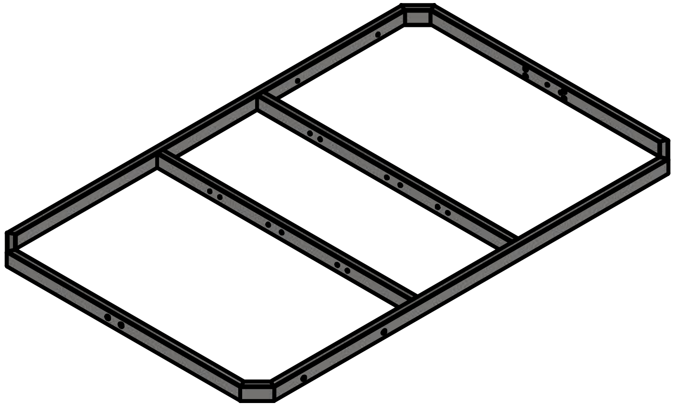
Edition	Sheet
	77 / 207

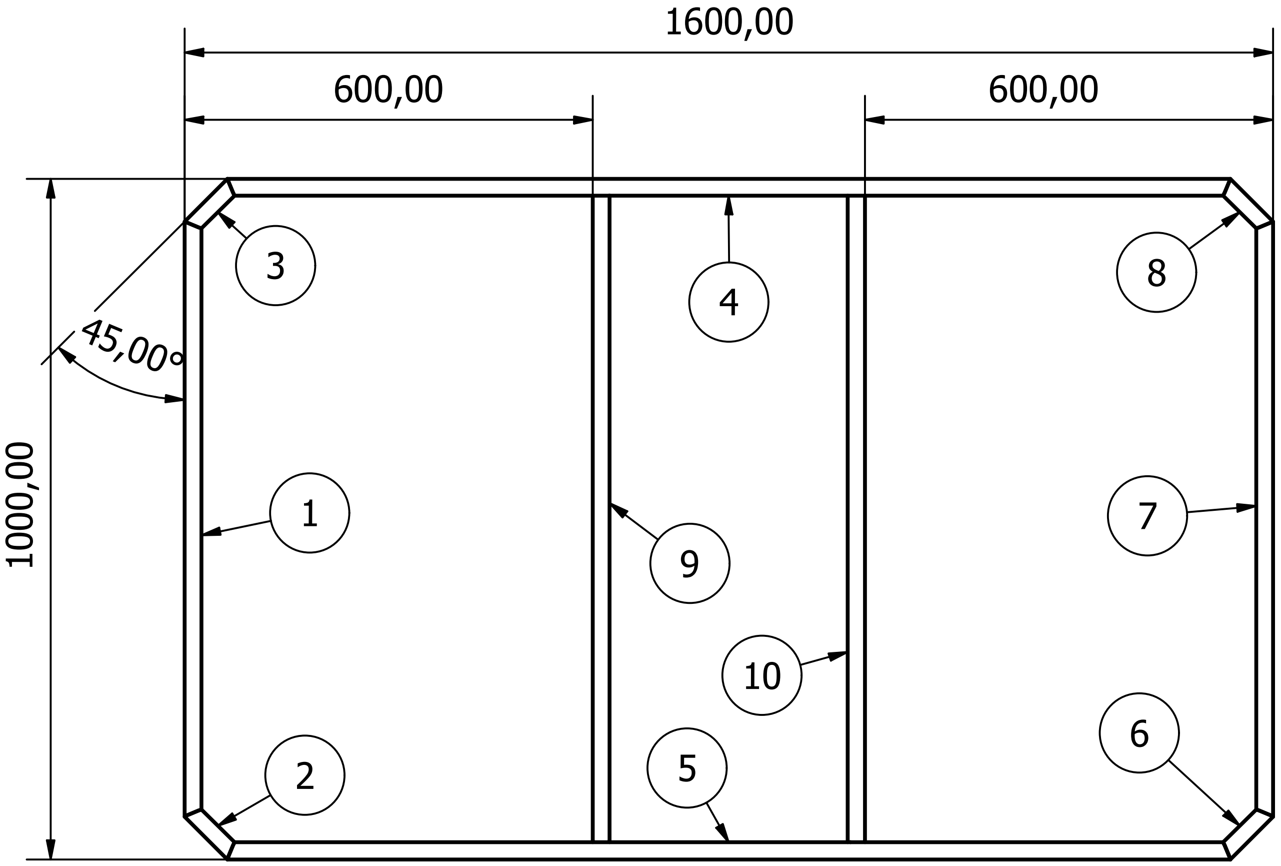


Designed by	Quantity	Manufactured by	Material	Date
AlexM			Steel, Mild, Welded	07/03/2019

NELSON MANDELA UNIVERSITY

	Edition	Sheet
Swerve Drive AGV		78 / 207





PARTS LIST		
ITEM	QTY	TITLE
1	874.000 mm	SABS 50x25x2 00099
2	89.095 mm	SABS 50x25x2 00100
3	89.095 mm	SABS 50x25x2 00098
4	1474.000 mm	SABS 50x25x2 00097
5	1474.000 mm	SABS 50x25x2 00095
6	89.095 mm	SABS 50x25x2 00094
7	874.000 mm	SABS 50x25x2 00093
8	89.095 mm	SABS 50x25x2 00096
9	950.000 mm	SABS 50x25x2 00110
10	950.000 mm	SABS 50x25x2 00111

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	Internal		07/03/2019

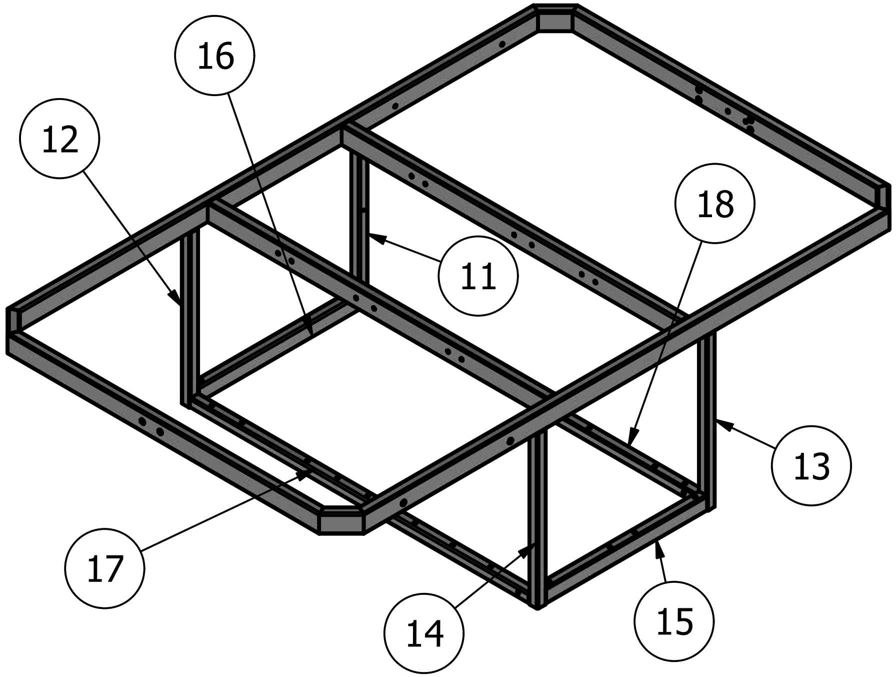
NELSON MANDELA UNIVERSITY

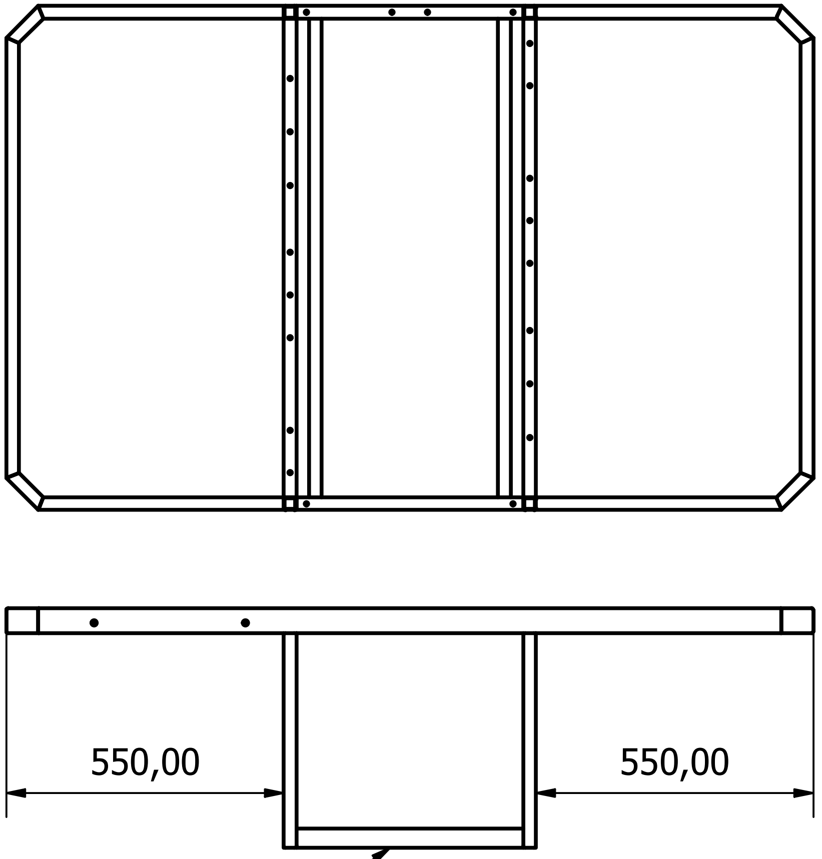
Body Weldmatt 1

Swerve Drive AGV

Edition

Sheet 79 / 207





This structure is centred about the middle of the AGV frame giving approximatly 550mm on each side

PARTS LIST		
ITEM	QTY	TITLE
11	425.000 mm	SABS 25x25x2 000075
12	425.000 mm	SABS 25x25x2 000074
13	425.000 mm	SABS 25x25x2 000072
14	425.000 mm	SABS 25x25x2 000073
15	450.000 mm	SABS 38x25x2 000114
16	450.000 mm	SABS 38x25x2 000115
17	950.000 mm	SABS 25x25x2 000079
18	950.000 mm	SABS 25x25x2 000078

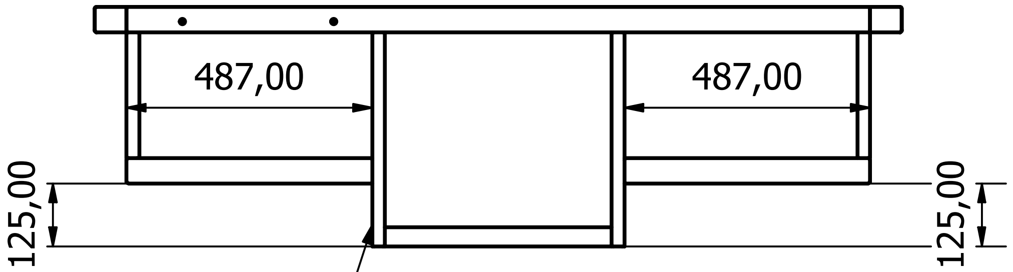
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	Internal		07/03/2019

NELSON MANDELA UNIVERSITY

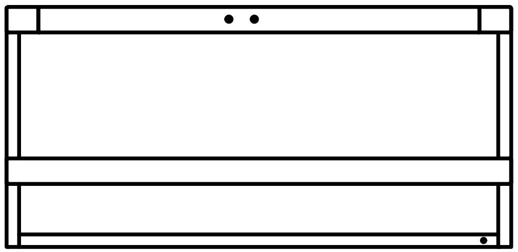
Body Weldmatt 2

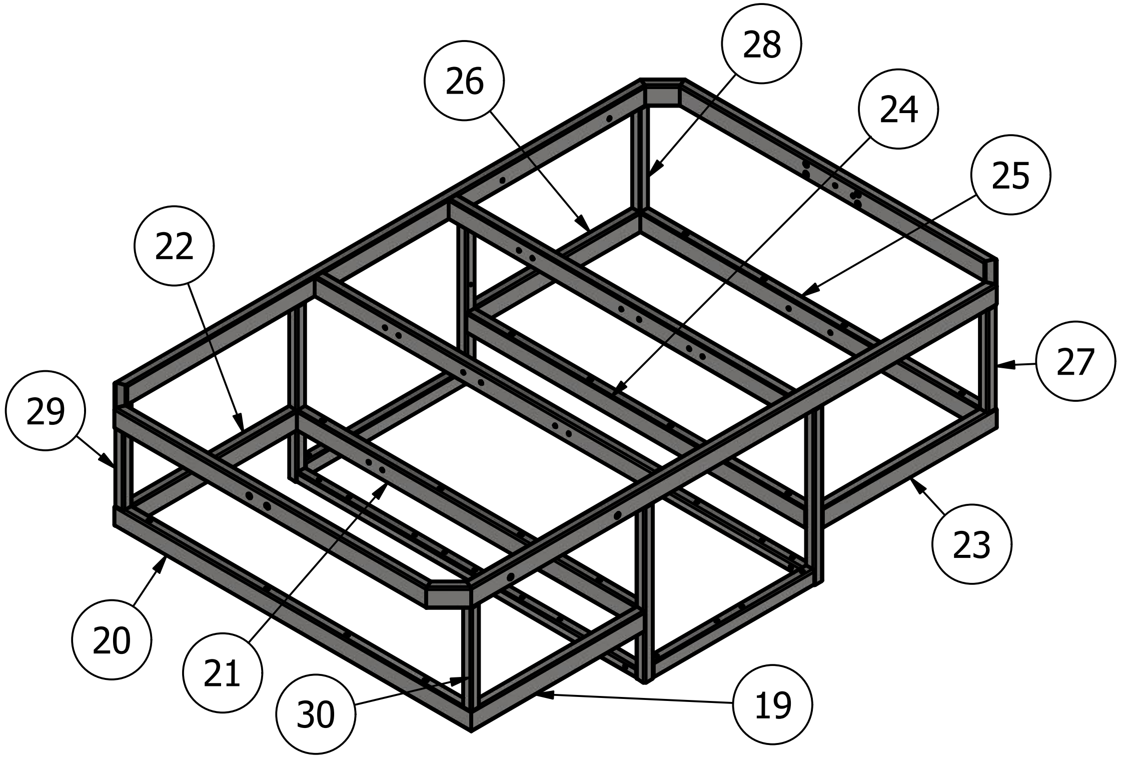
Swerve Drive AGV

Edition	Sheet
	80 / 207



45 degree braces are placed here 125mm should be set by them see "Body Weldmatt 4"





PARTS LIST		
ITEM	QTY	TITLE
19	487.000 mm	SABS 50x25x2 000060
20	1000.000 mm	SABS 50x25x2 000061
21	950.000 mm	SABS 50x25x2 000062
22	487.000 mm	SABS 50x25x2 000063
23	487.000 mm	SABS 50x25x2 000064
24	950.000 mm	SABS 50x25x2 000065
25	1000.000 mm	SABS 50x25x2 000066
26	487.000 mm	SABS 50x25x2 000067
27	250.000 mm	SABS 50x25x2 000080
28	250.000 mm	SABS 50x25x2 000081
29	250.000 mm	SABS 50x25x2 000082
30	250.000 mm	SABS 25x25x2 000083

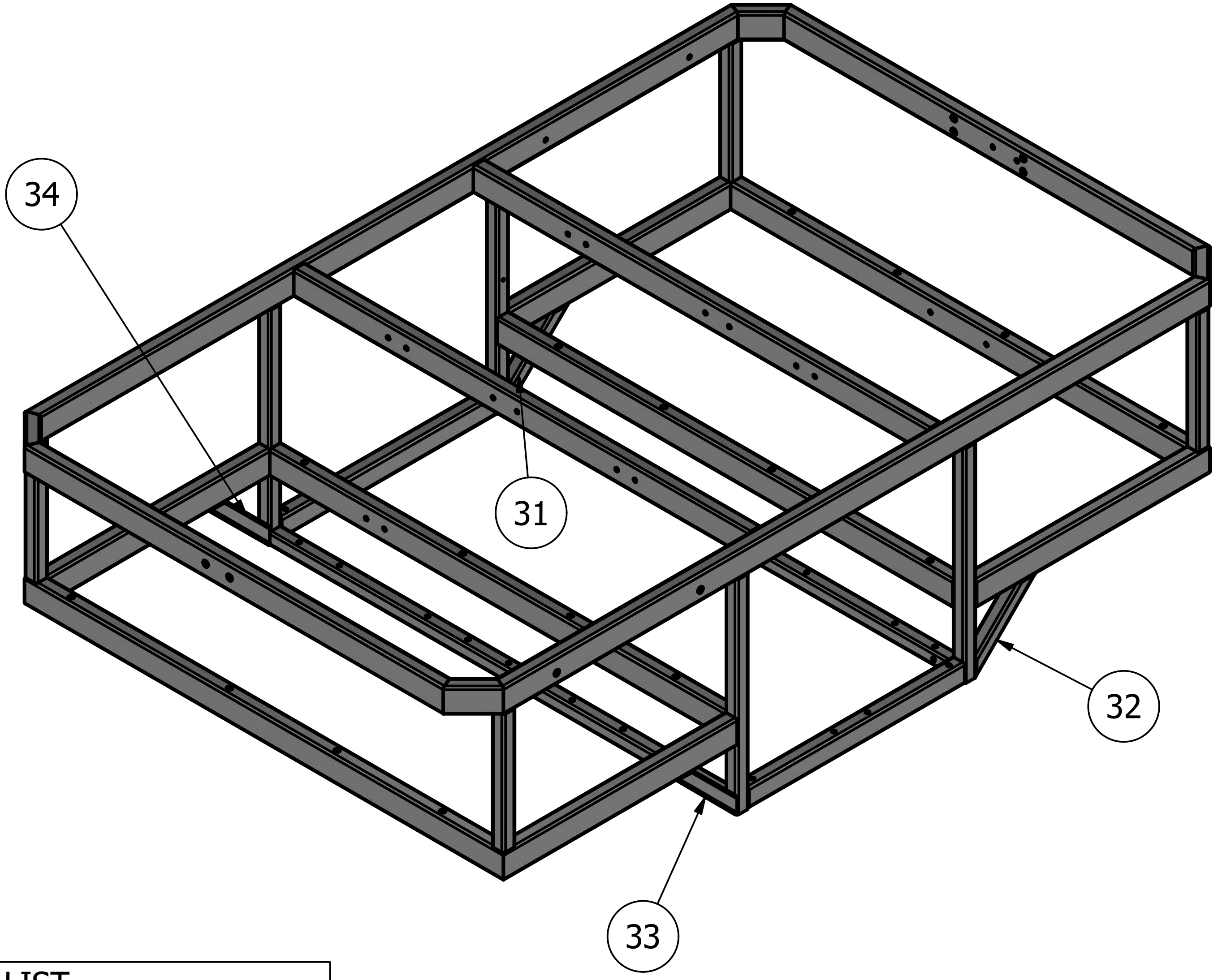
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	Internal		07/03/2019

NELSON MANDELA UNIVERSITY

Body Weldmatt 3

Swerve Drive AGV

Edition	Sheet
	81 / 207



PARTS LIST		
ITEM	QTY	TITLE
31	176.777 mm	SABS 25x25x2 000068
32	176.777 mm	SABS 25x25x2 000069
33	176.777 mm	SABS 25x25x2 000070
34	176.777 mm	SABS 25x25x2 000071

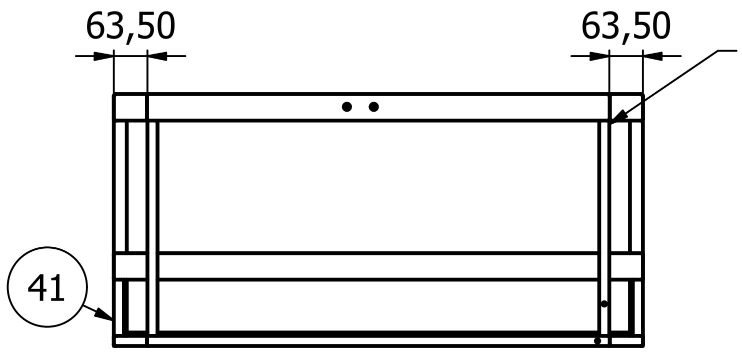
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	Internal		07/03/2019

NELSON MANDELA UNIVERSITY

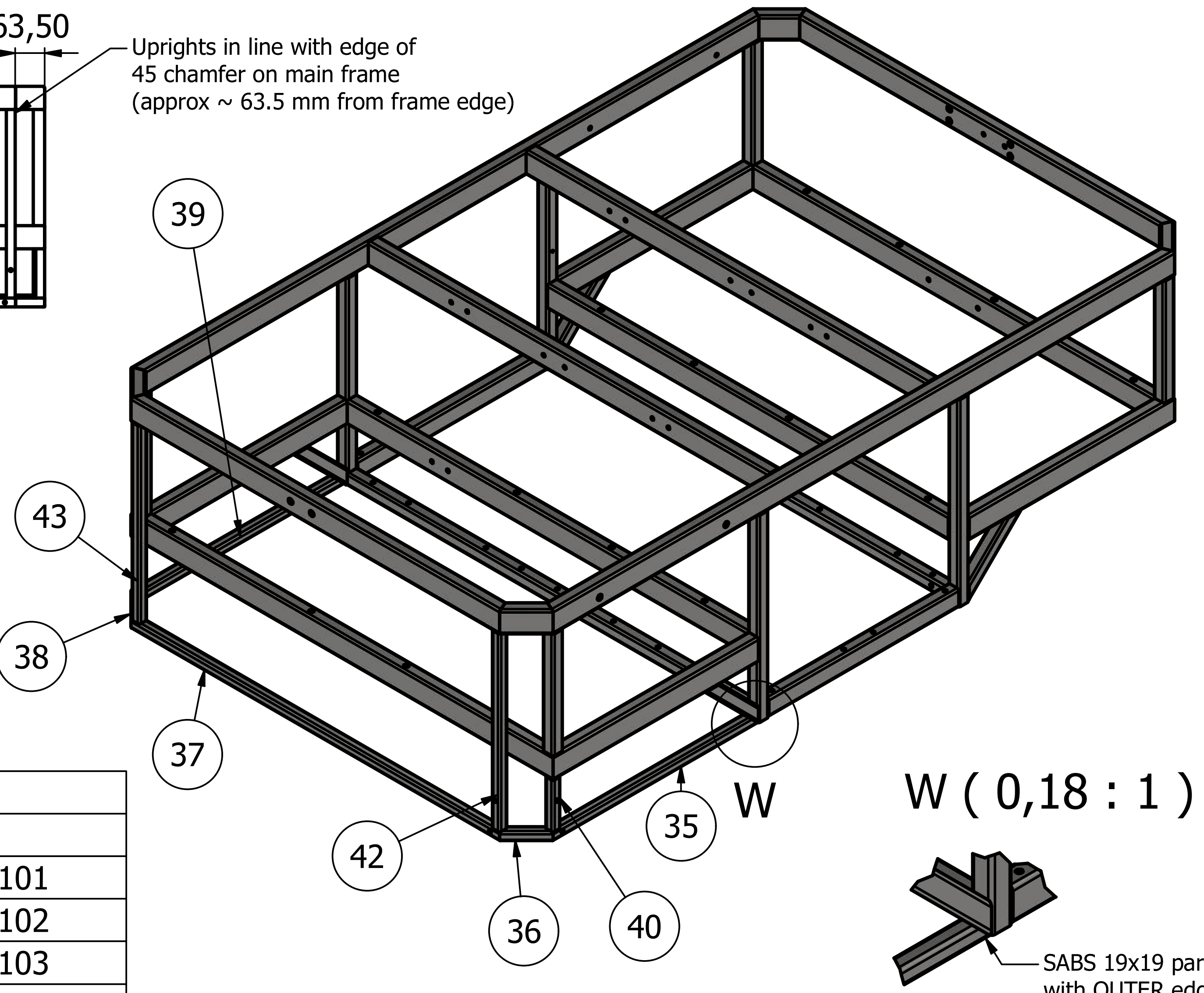
Body Weldmatt 4

Swerve Drive AGV

Edition	Sheet
	82 / 207



Uprights in line with edge of 45 chamfer on main frame (approx ~ 63.5 mm from frame edge)



SABS 19x19 parrallel with OUTER edge of main structure

PARTS LIST

ITEM	QTY	TITLE
35	487.000 mm	SABS 19x19x2 000101
36	89.095 mm	SABS 19x19x2 000102
37	874.000 mm	SABS 19x19x2 000103
38	89.095 mm	SABS 19x19x2 000104
39	487.000 mm	SABS 19x19x2 000105
40	106.000 mm	SABS 19x19x2 000106
41	106.000 mm	SABS 19x19x2 000107
42	406.000 mm	SABS 19x19x2 000108
43	406.000 mm	SABS 19x19x2 000109

Designed by	Quantity	Manufactured by	Material	Date
AlexM				07/03/2019

NELSON MANDELA UNIVERSITY

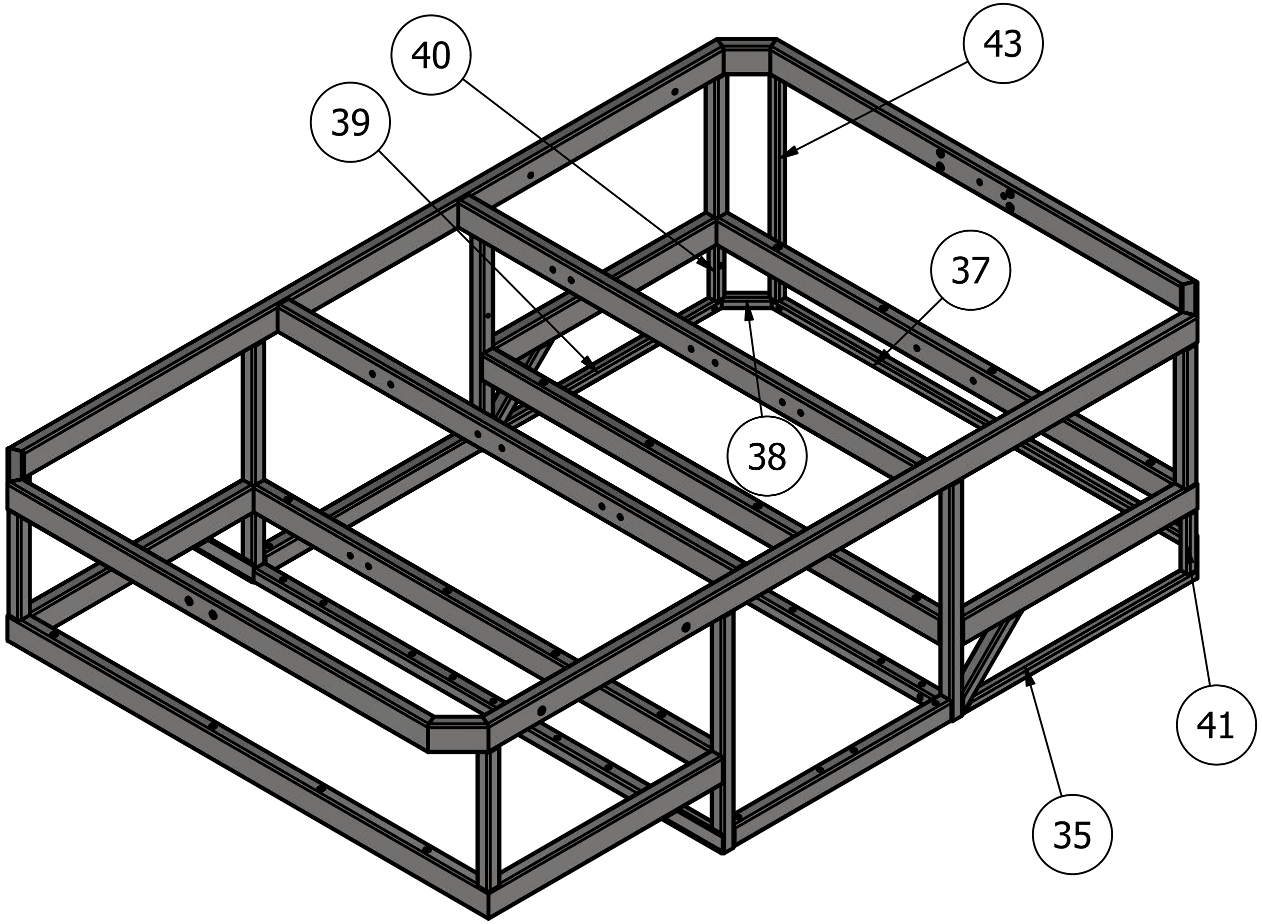
Body Welmatt 5

Swerve Drive AGV

Edition

Sheet 83 / 207

Rear skirt is a mirror image of the
front skirt which can be found on
"Body Wledmatt 5"



PARTS LIST		
ITEM	QTY	TITLE
35	487.000 mm	SABS 19x19x2 000084
36	89.095 mm	SABS 19x19x2 000085
37	874.000 mm	SABS 19x19x2 000086
38	89.095 mm	SABS 19x19x2 000087
39	487.000 mm	SABS 19x19x2 000088
40	106.000 mm	SABS 19x19x2 000089
41	106.000 mm	SABS 19x19x2 000090
42	406.000 mm	SABS 19x19x2 000091
43	406.000 mm	SABS 19x19x2 000092

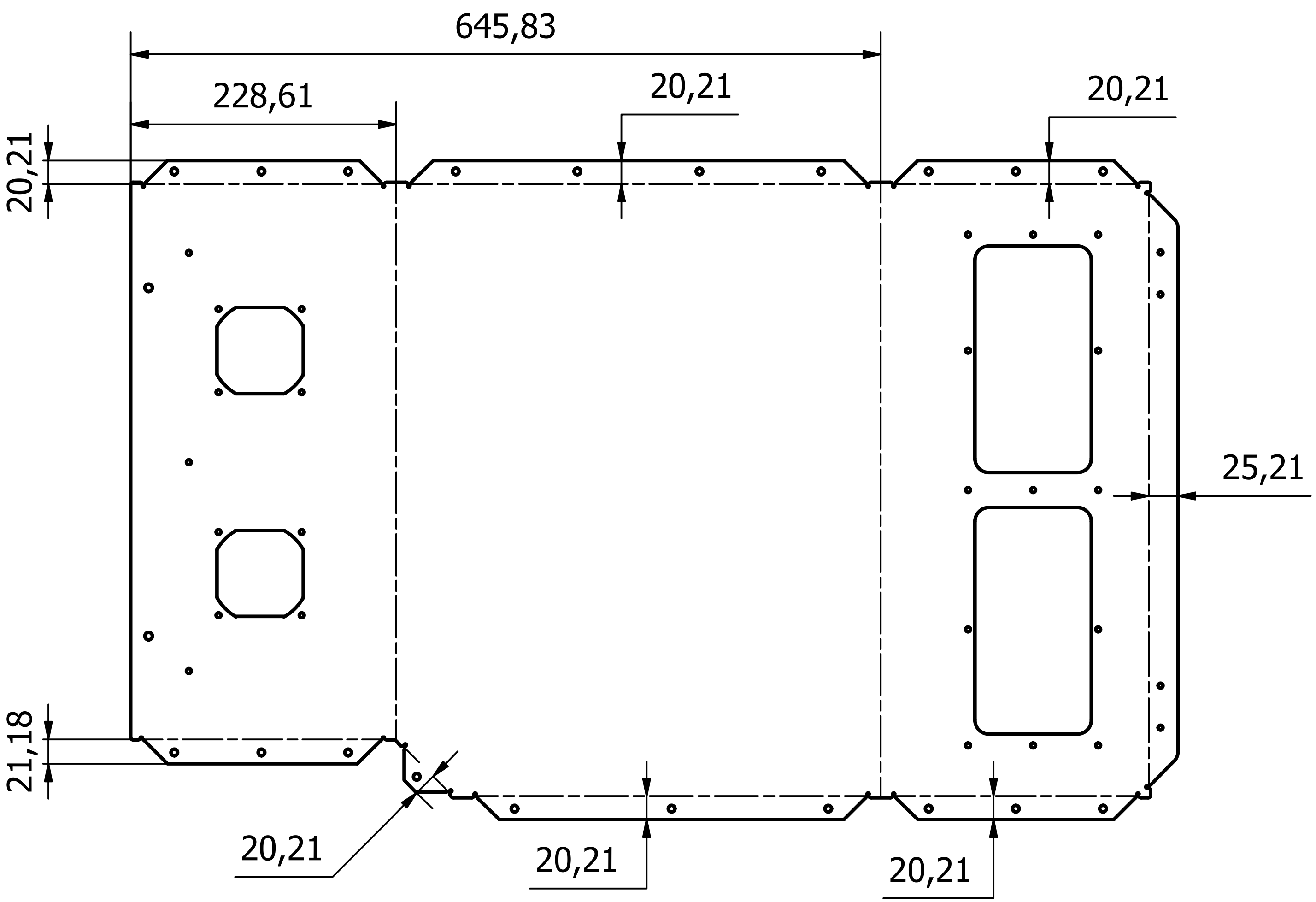
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	Internal		07/03/2019

NELSON MANDELA UNIVERSITY

Body Weldmatt 6

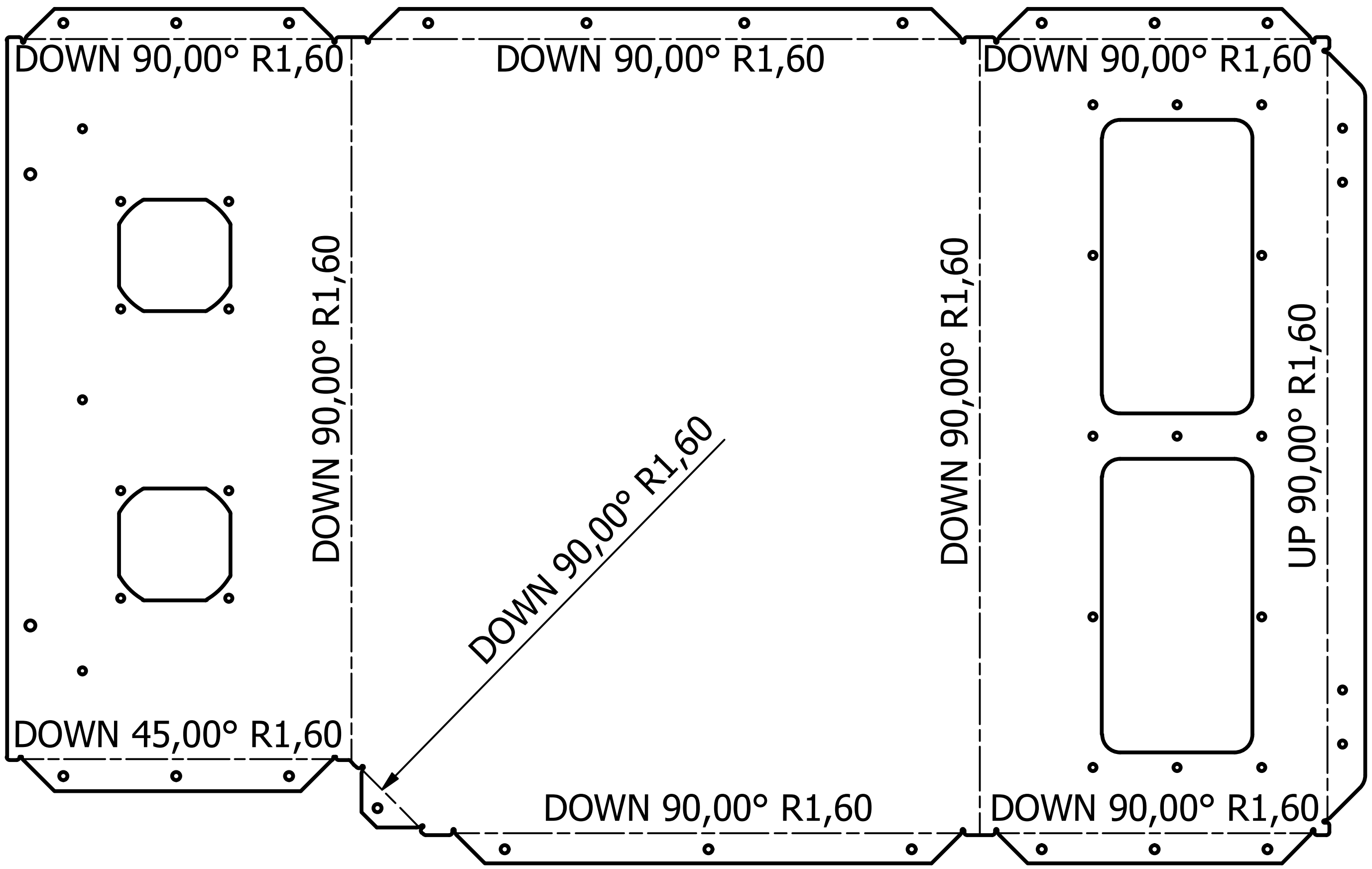
Swerve Drive AGV

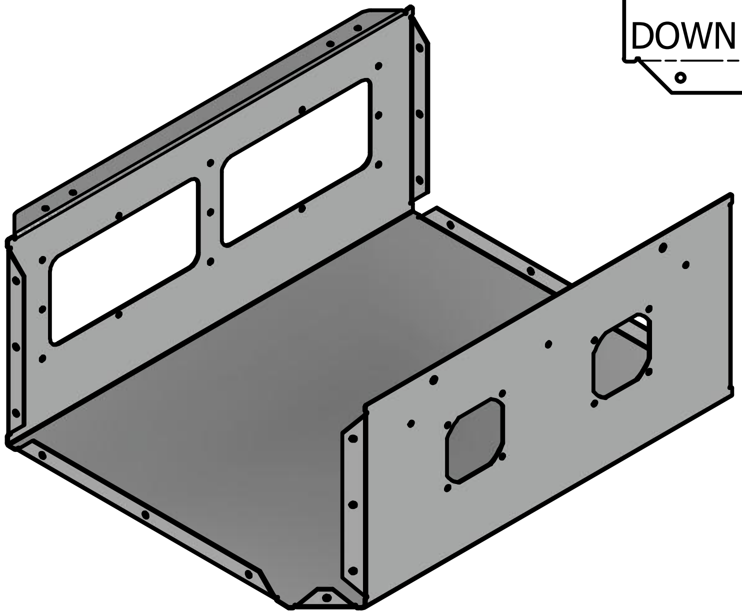
Edition	Sheet
	84 / 207



Sheet Thickness: 1,60mm

Designed by	Quantity	Manufactured by	Material	Date	
AlexM	2	External	Aluminum 6061	07/03/2019	
NELSON MANDELA UNIVERSITY		CB Main Panel - Dimensions			
		Swerve Drive AGV		Edition	Sheet 85 / 207





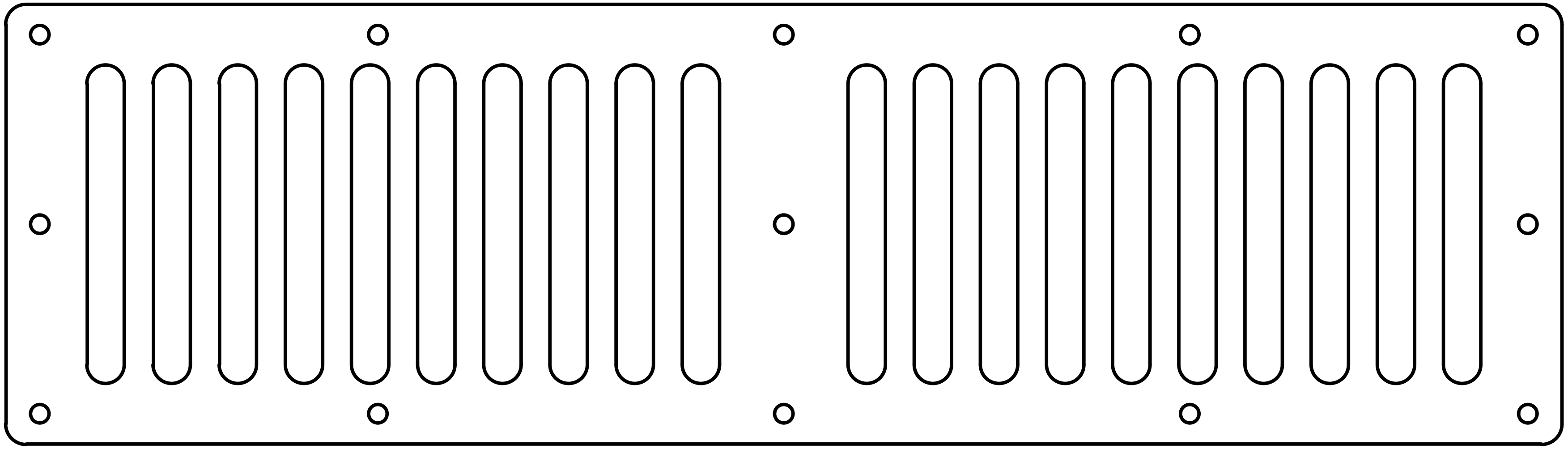
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

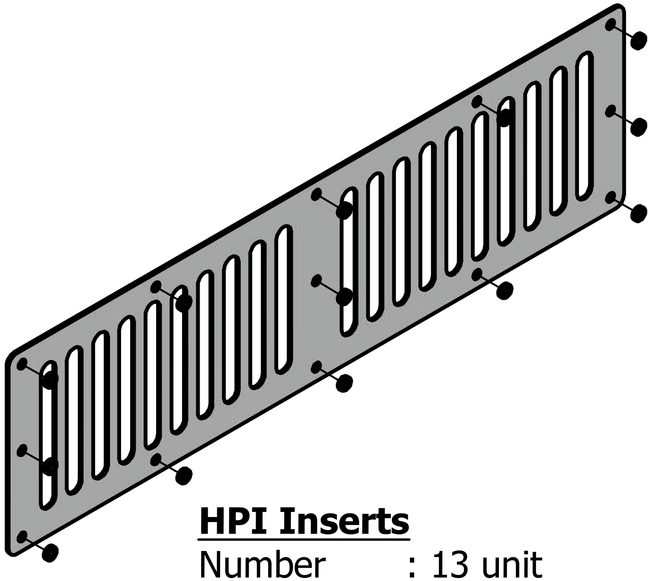
CB Main Panel - Bends

Swerve Drive AGV

Edition	Sheet
	86 / 207



Sheet Thickness: 1,60 mm



HPI Inserts

Number : 13 unit

Designation : CM4-0 (SCN M4)

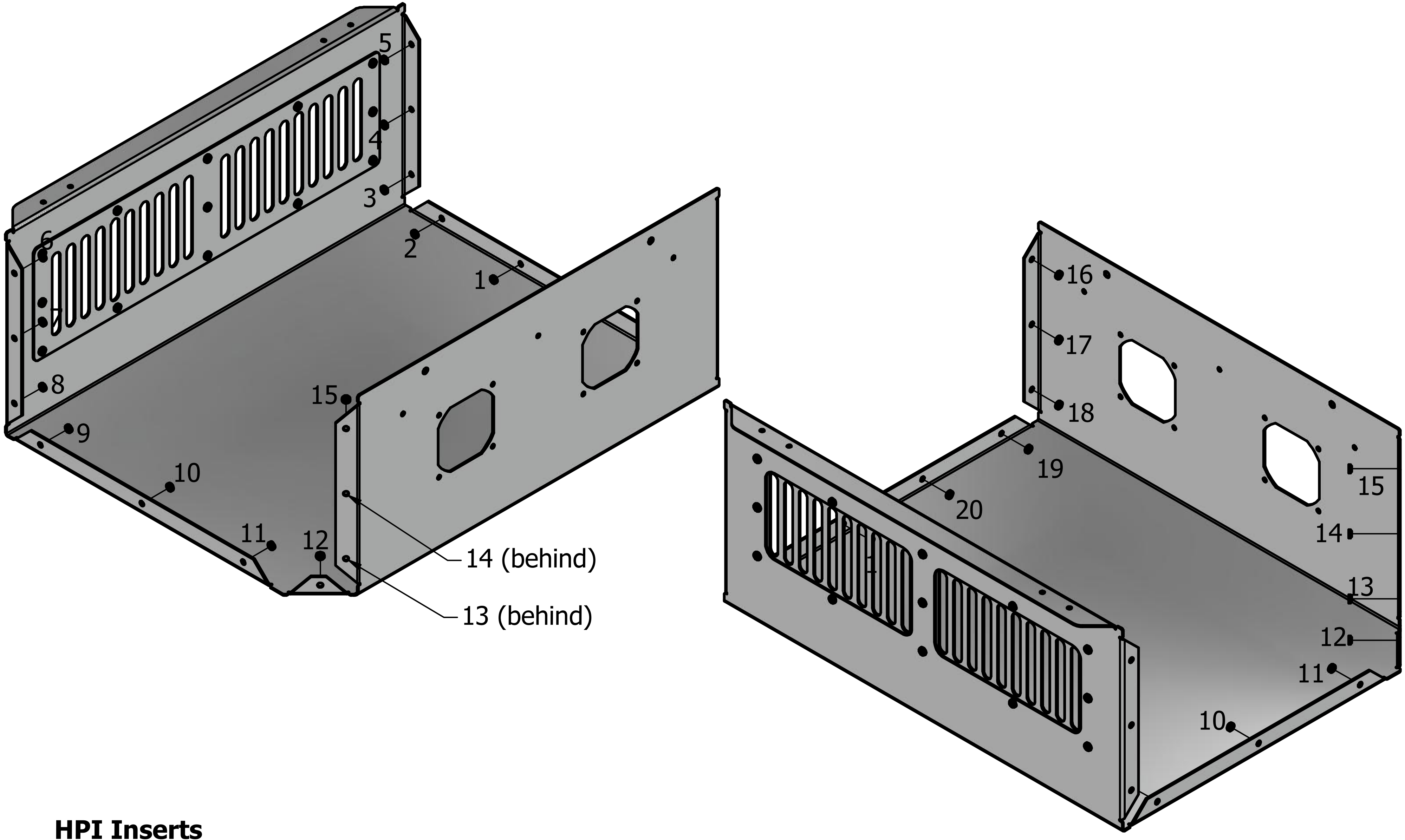
PARTS LIST		
ITEM	QTY	PART NUMBER
1	1	Air Vent (Cut Ali Sheet)
2	13	SCN M4 - CM4-0

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2		Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Electronics Box Air Vent

	Edition	Sheet
Swerve Drive AGV		87 / 207



HPI Inserts

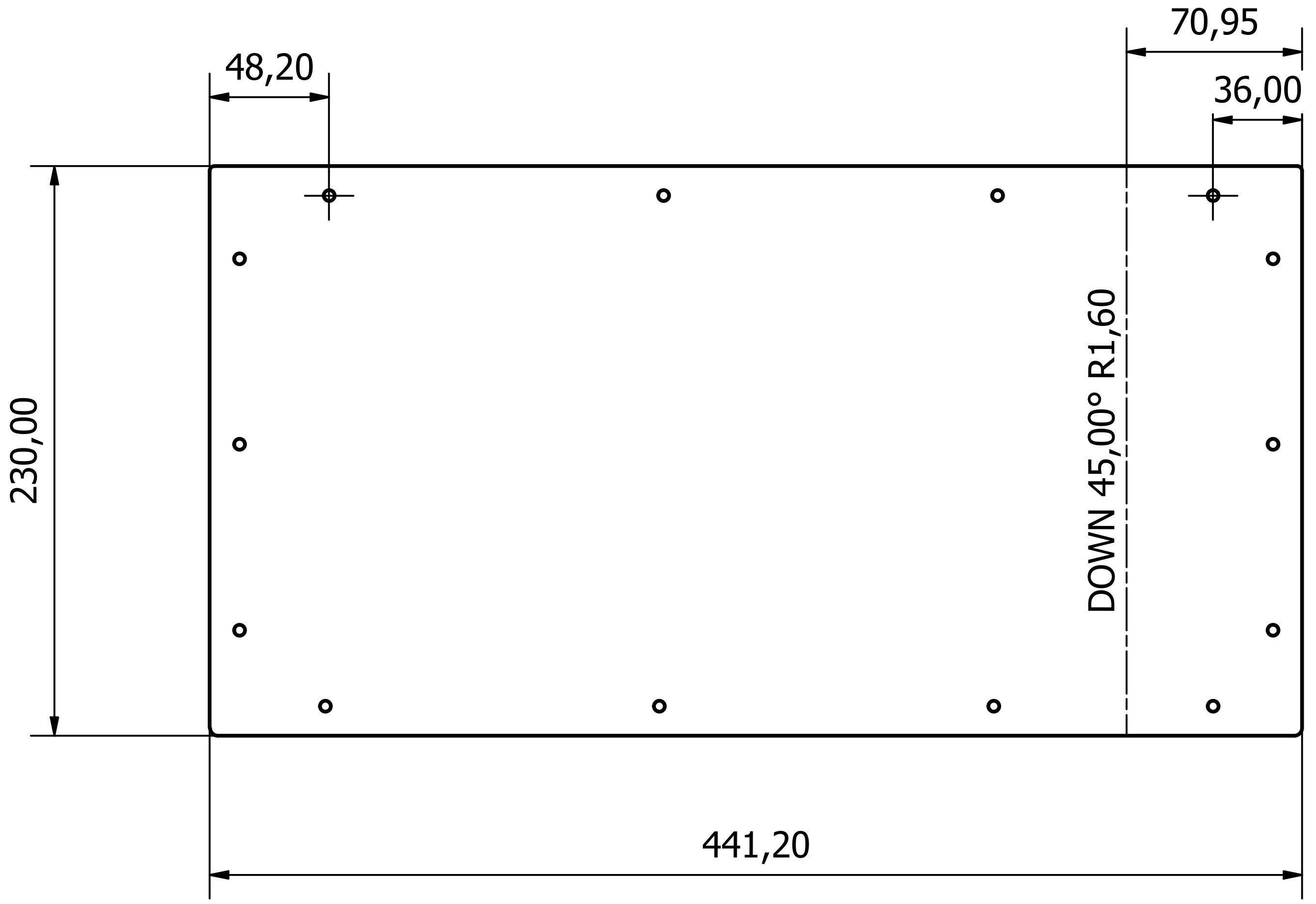
Number	: 20 unit
Designation	: CM4-0 (SCN M4)

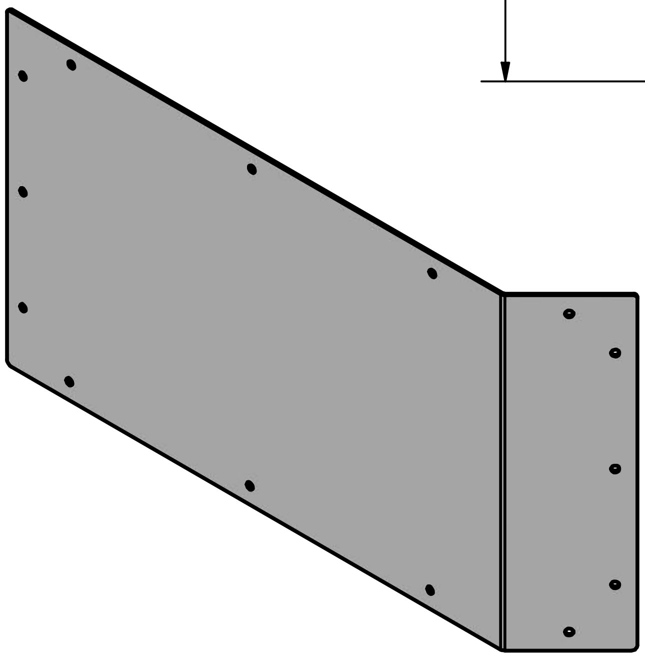
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External		07/03/2019

NELSON MANDELA UNIVERSITY

CB Main Panel - Inserts

	Edition	Sheet
Swerve Drive AGV		88 / 207





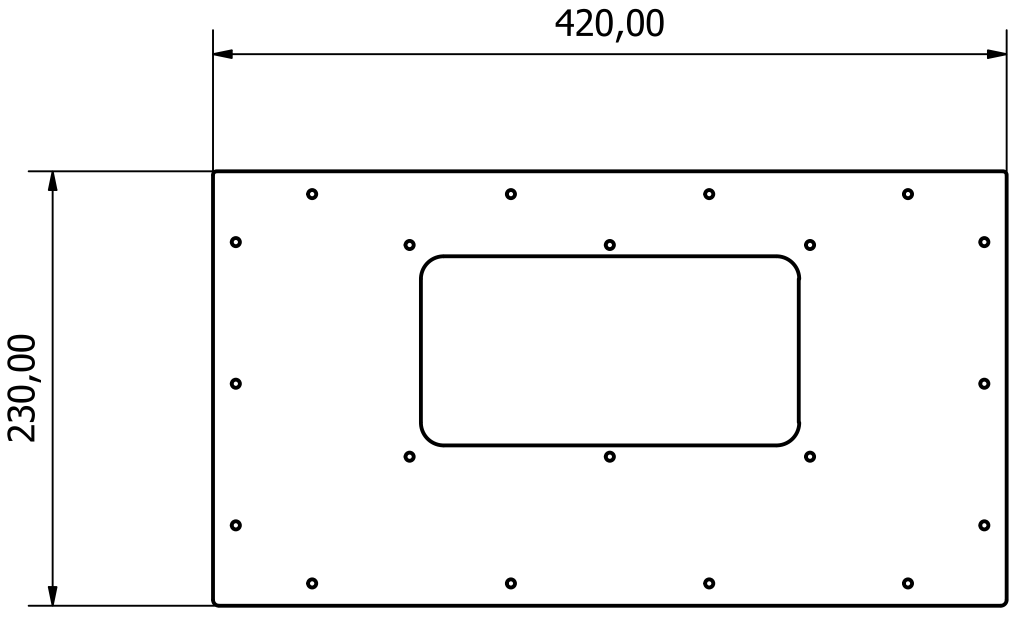
Sheet Thickness: 1,60mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Side Panel 1

	Edition	Sheet
Swerve Drive AGV		89 / 207



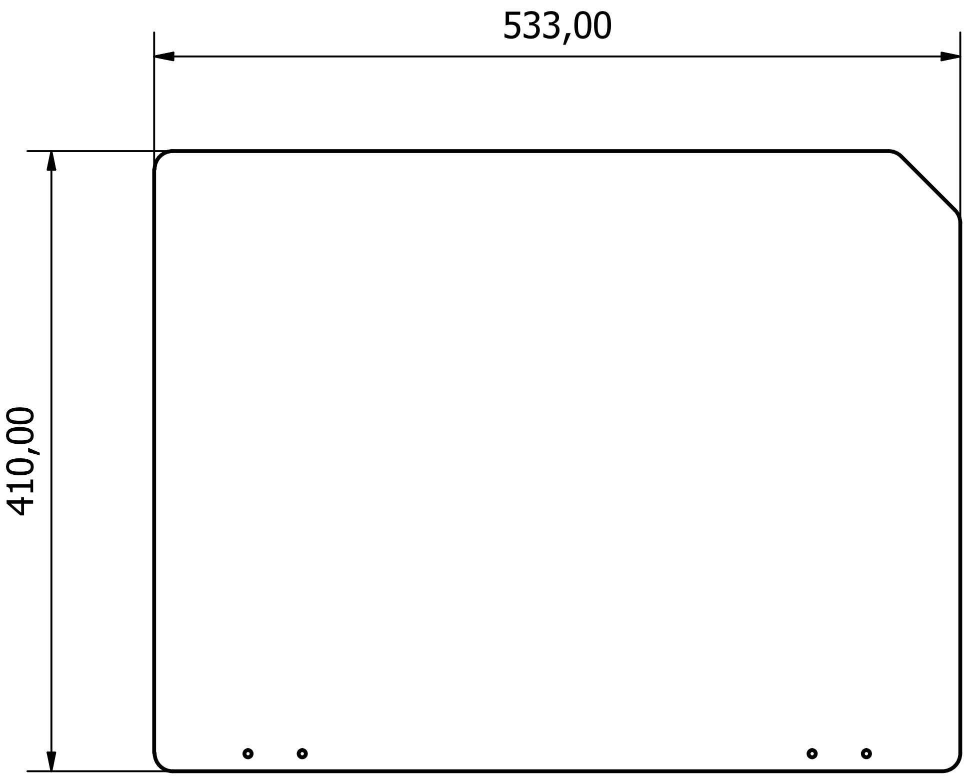
Sheet Thickness: 1,60mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Side Panel 2

Swerve Drive AGV	Edition	Sheet
		90 / 207



Sheet Thickness: 2,00mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Box Door

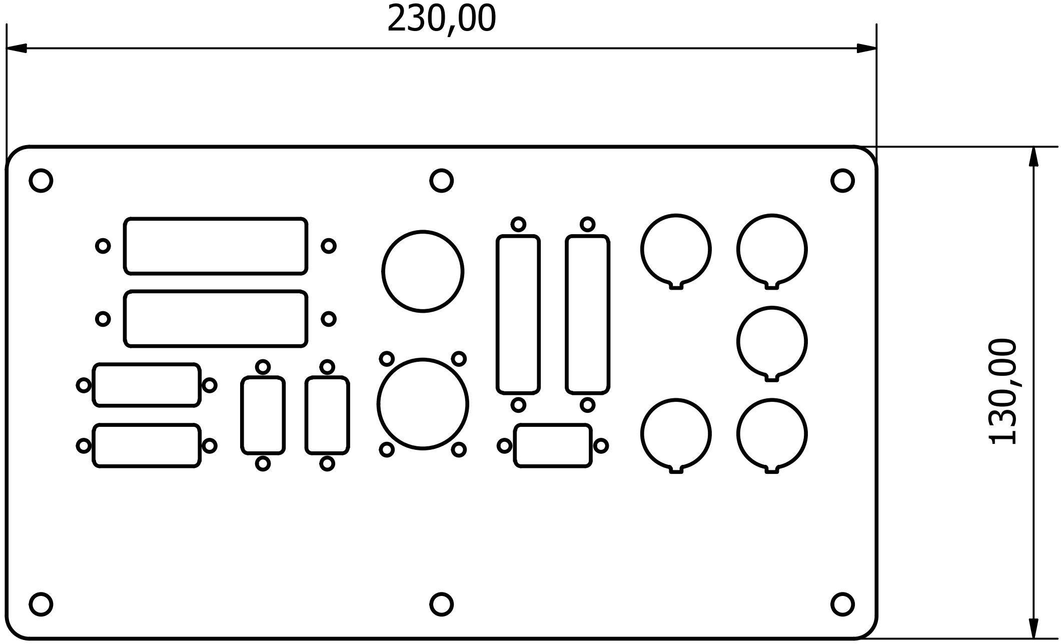
Swerve Drive AGV

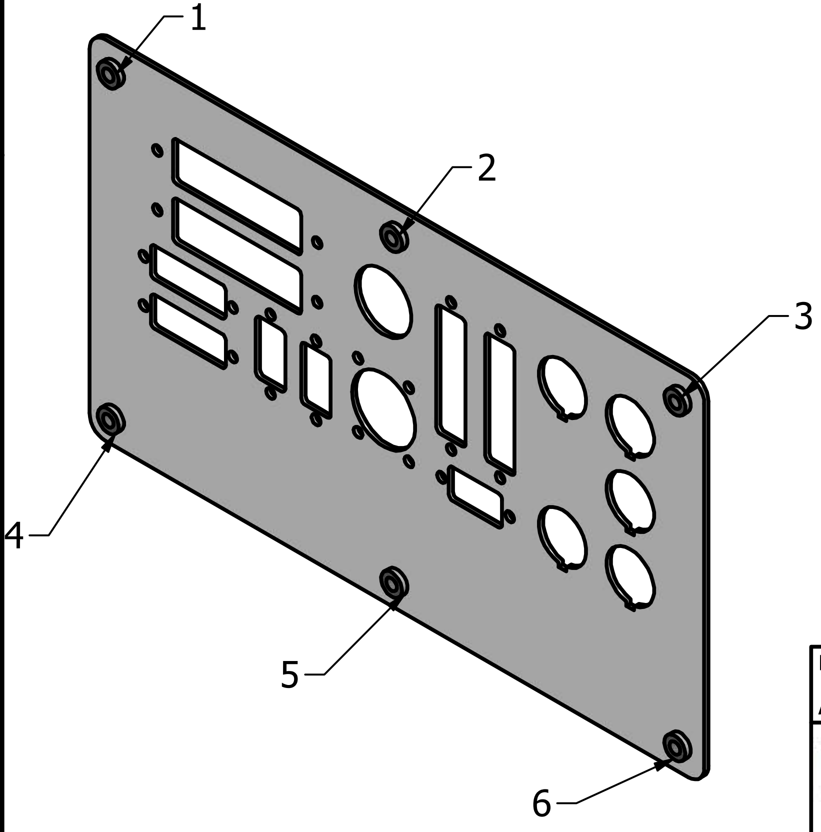
Edition	Sheet
	91 / 207

HPI Inserts

Number : 6 unit

Designation : CM4-0 (SCN M4)





Sheet Thickness: 1,60mm

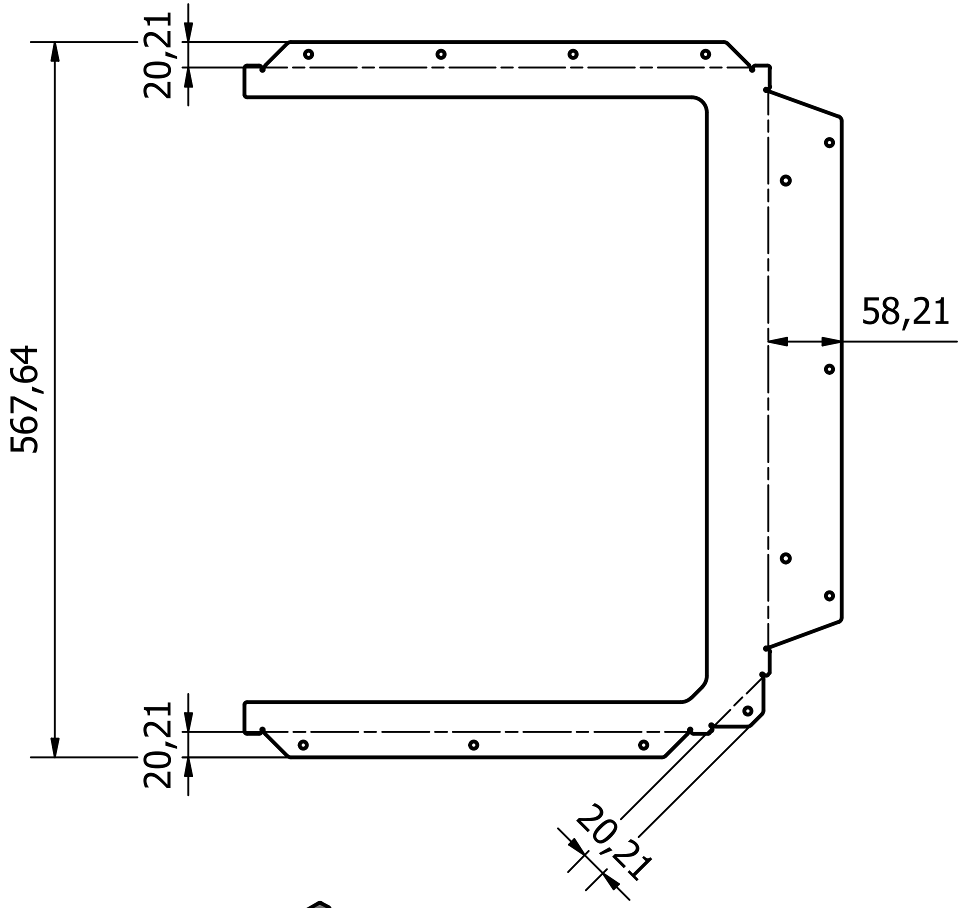
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Aluminium 6061	07/03/2019

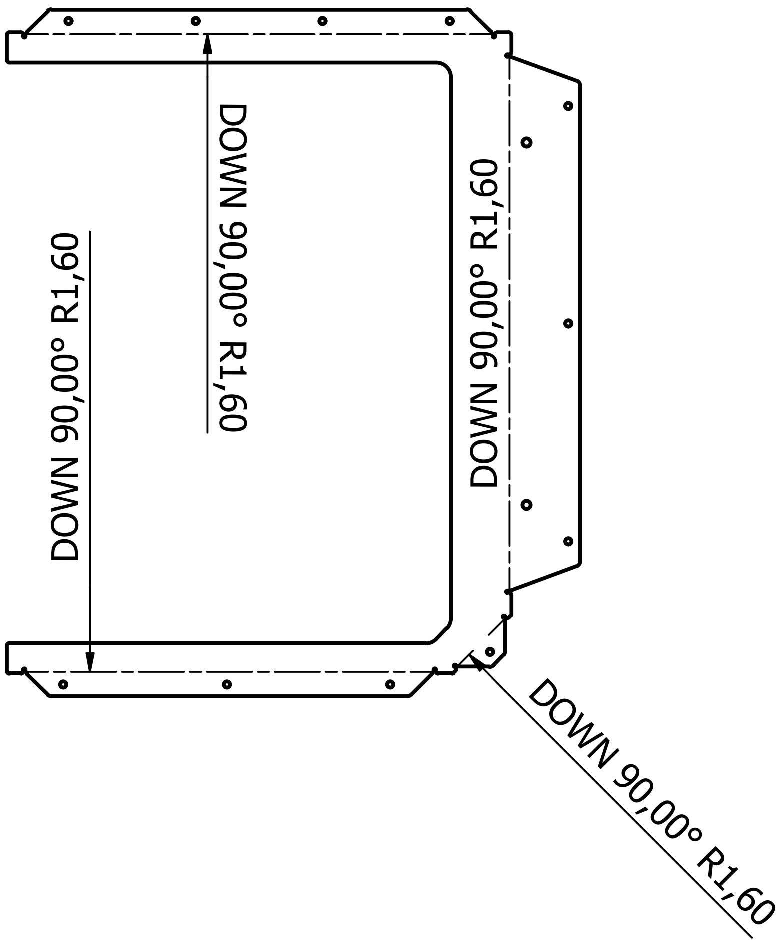
NELSON MANDELA UNIVERSITY

Electrical Connection Plate Type B

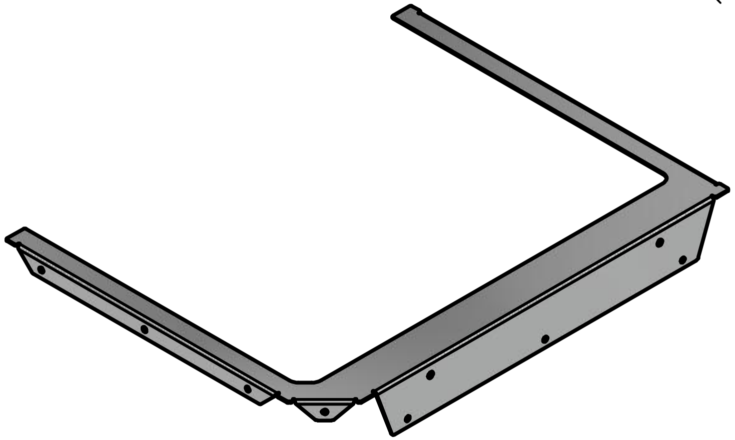
Swerve Drive AGV

Edition	Sheet
	92 / 207





Sheet Thickness: 1,60mm



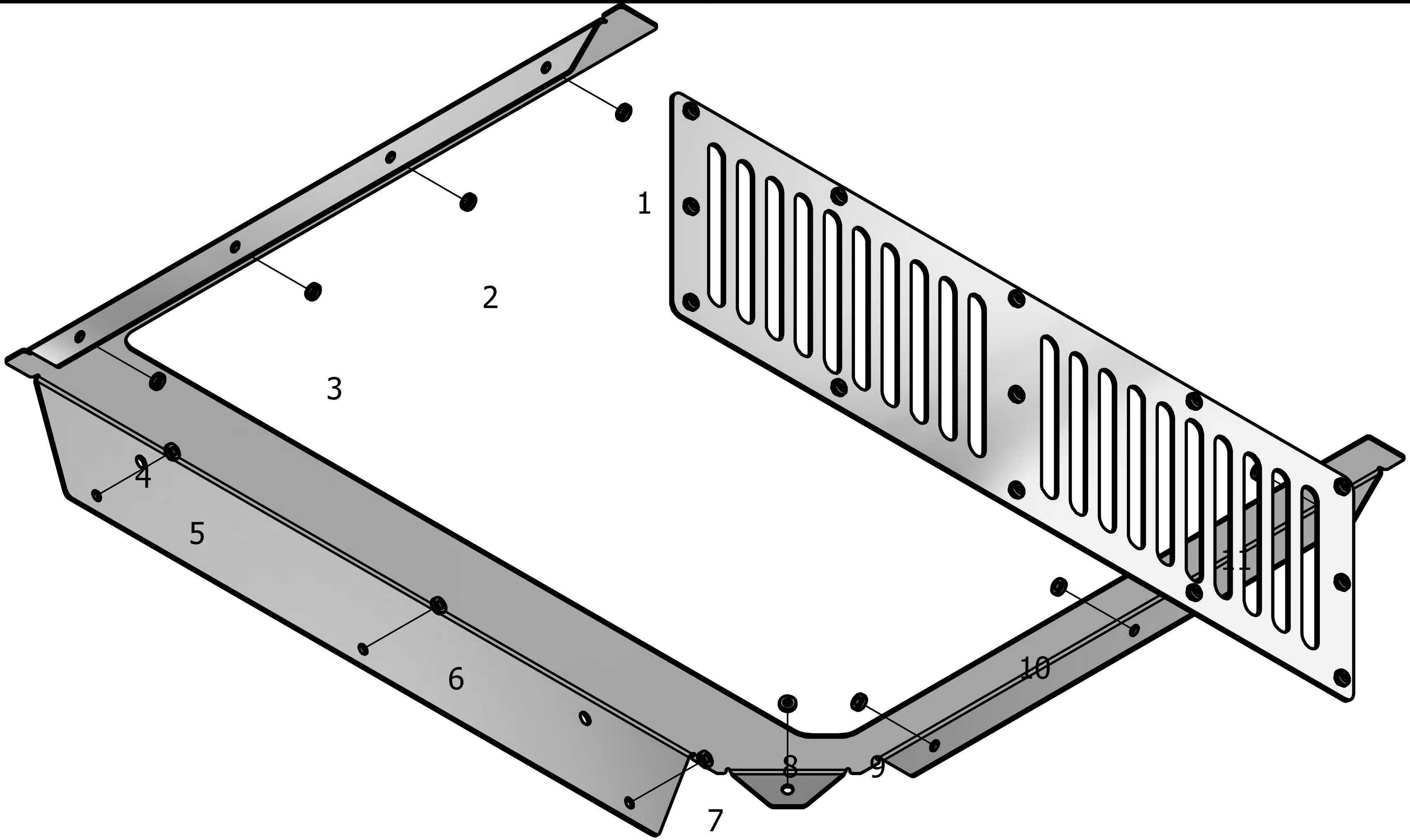
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Box Door Frame - Dimensions & Bend

Swerve Drive AGV

Edition	Sheet
	93 / 207



HPI Inserts

Number : 11 unit
Designation : CM4-0 (SCN M4)

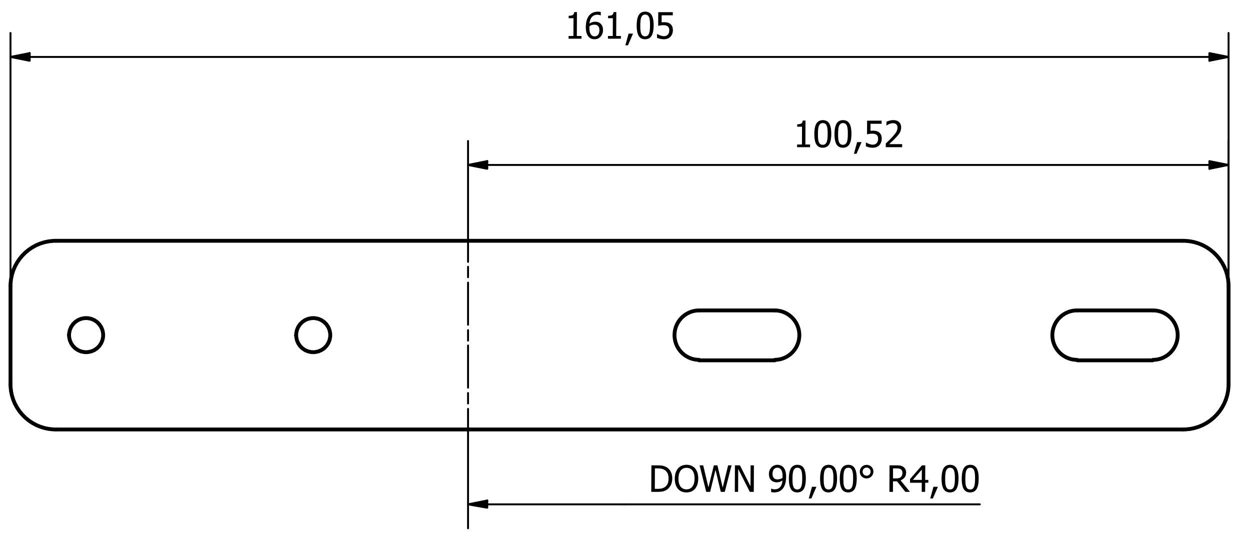
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External		07/03/2019

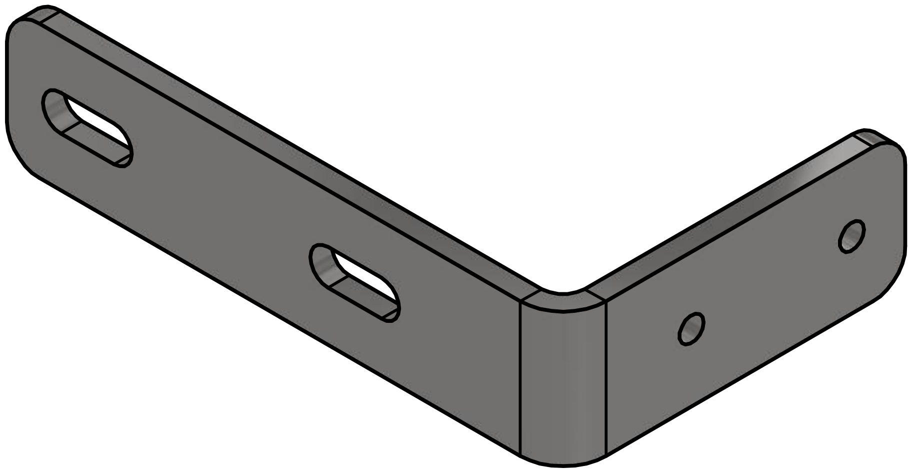
NELSON MANDELA UNIVERSITY

Box Door Frame - Inserts

Swerve Drive AGV

Edition	Sheet
	94 / 207





Sheet Thickness: 4,00mm

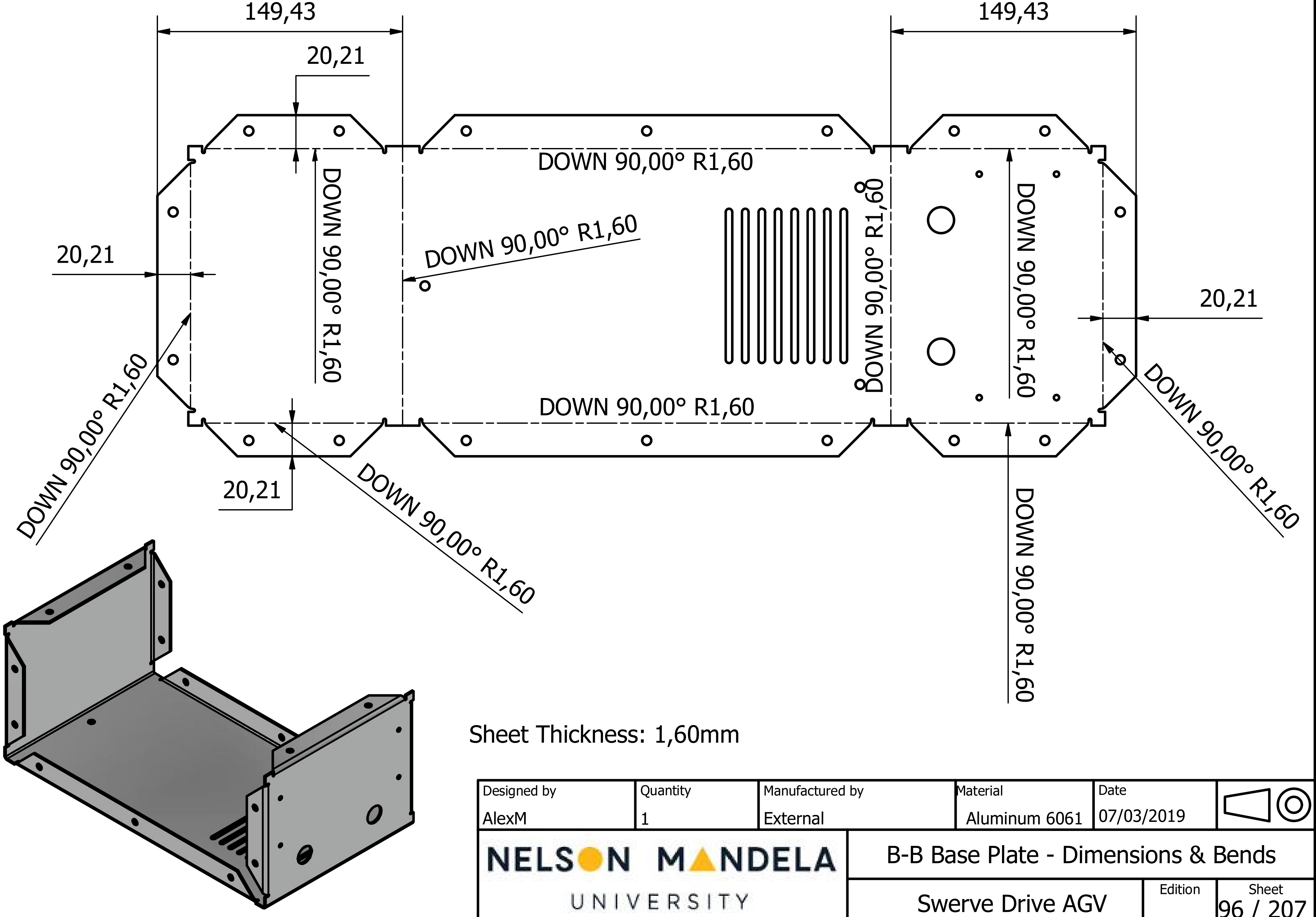
Designed by	Quantity	Manufactured by	Material	Date
AlexM	4	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Support Bar Brackets

Swerve Drive AGV

Edition	Sheet
	95 / 207

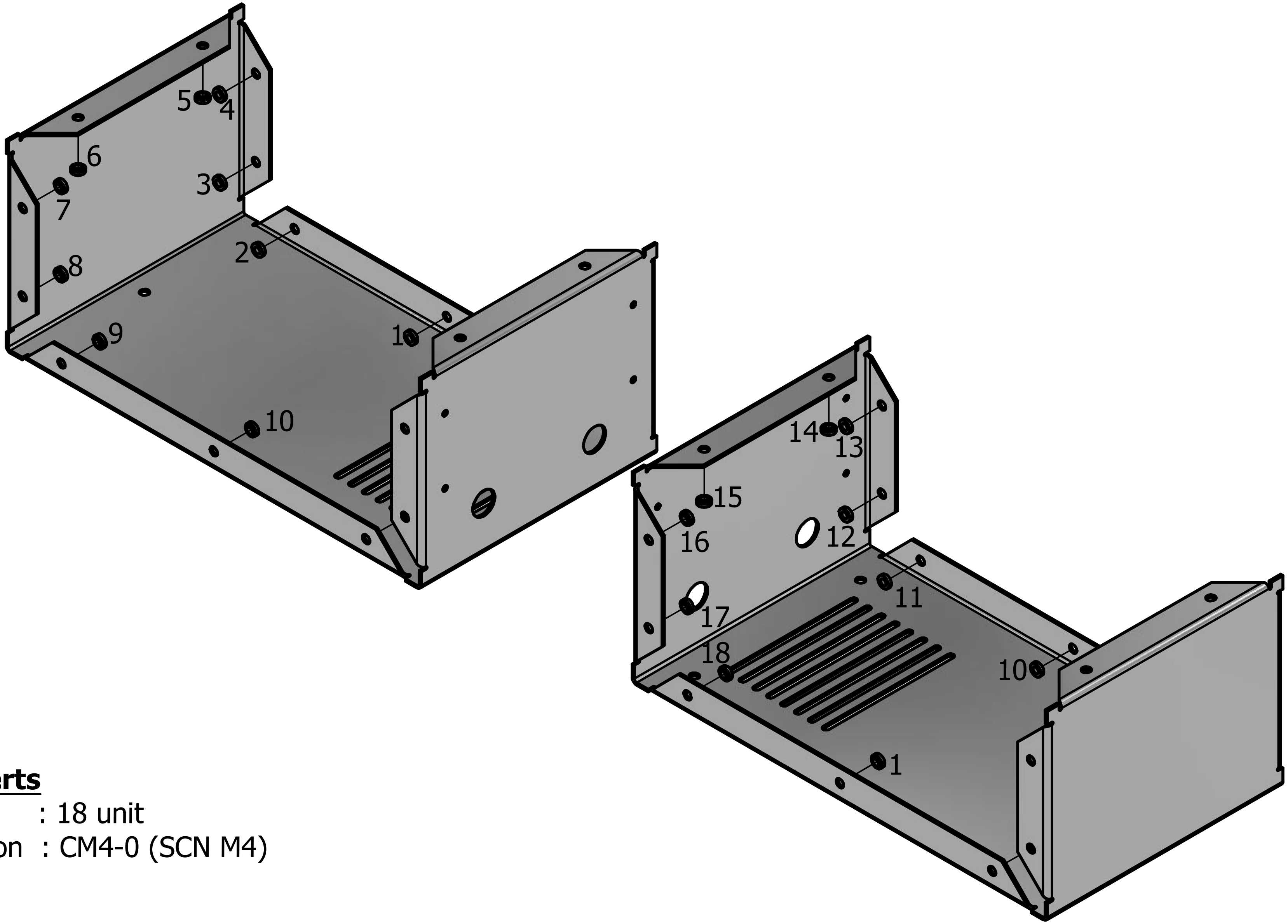


Sheet Thickness: 1,60mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	External	Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

B-B Base Plate - Dimensions & Bends		
Swerve Drive AGV	Edition	Sheet 96 / 207



HPI Inserts

Number : 18 unit

Designation : CM4-0 (SCN M4)

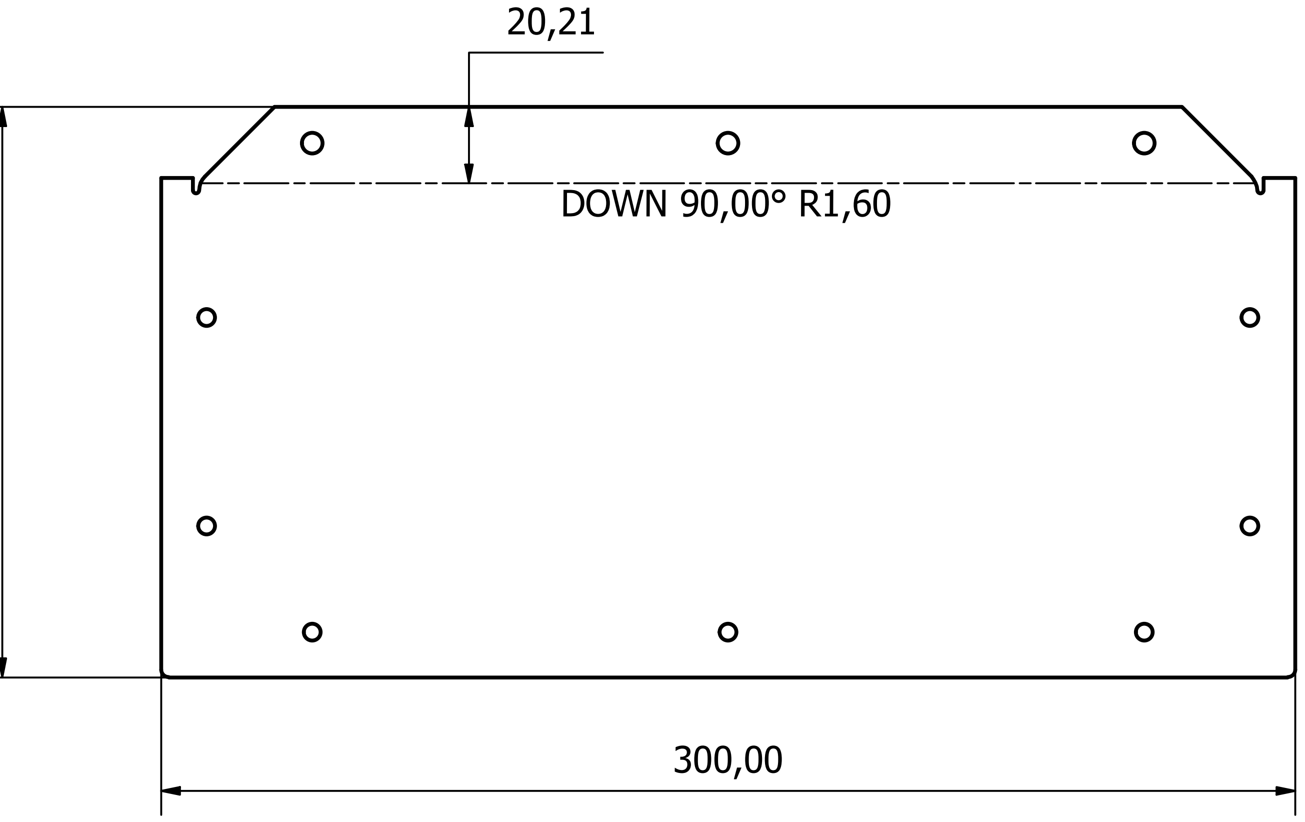
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	External		07/03/2019

NELSON MANDELA UNIVERSITY

B-B Base Plate - Inserts

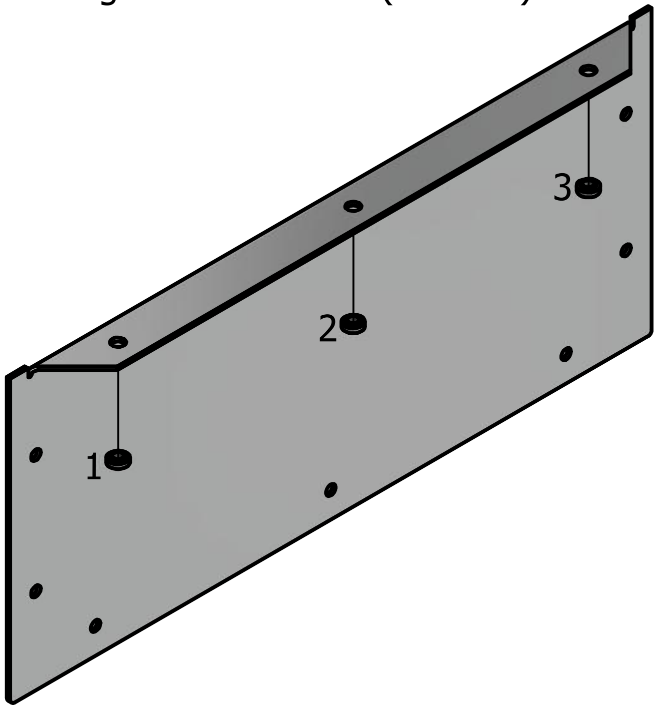
Swerve Drive AGV

Edition	Sheet
	97 / 207



Sheet Thickness: 1,60mm

HPI Inserts
Number : 3 unit
Designation : CM4-0 (SCN M4)

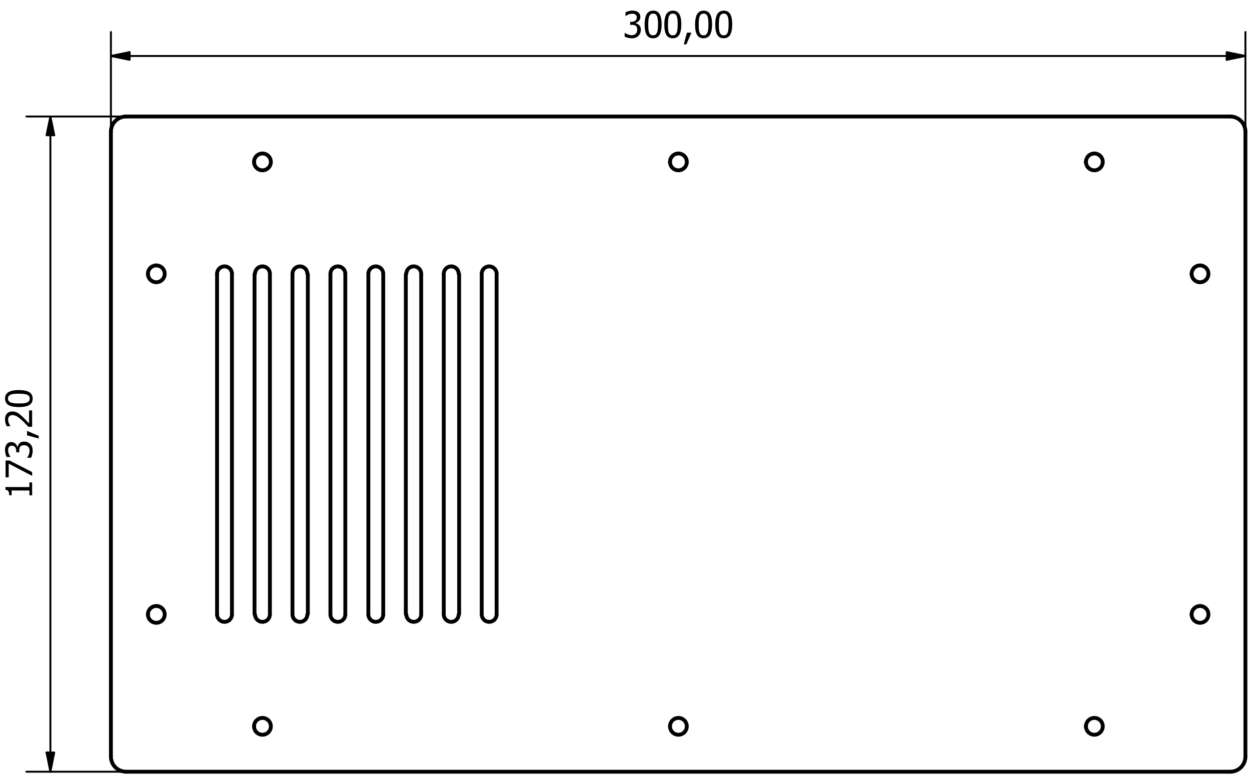


Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

B-B Side Plate Type A

	Edition	Sheet
Swerve Drive AGV		98 / 207



Sheet Thickness: 1,60mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	External	Aluminum 6061	07/03/2019

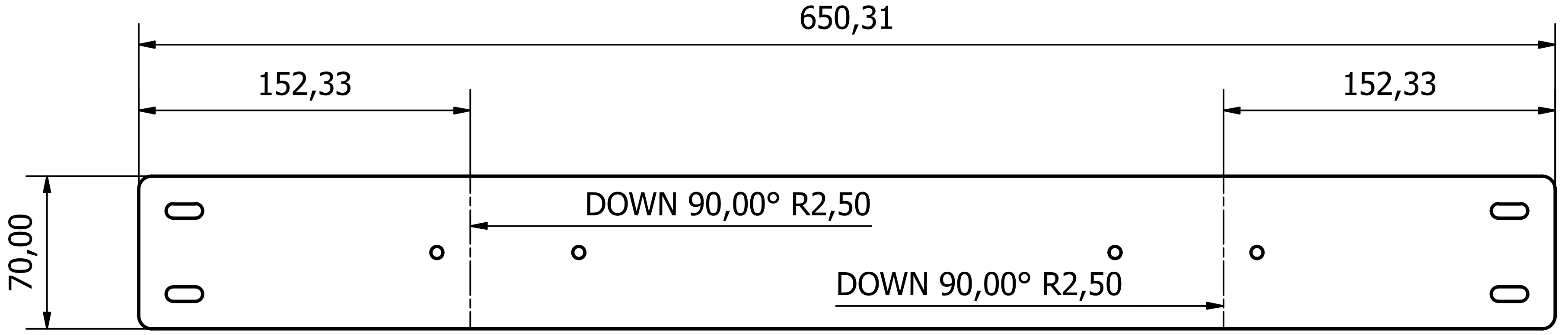
NELSON MANDELA UNIVERSITY

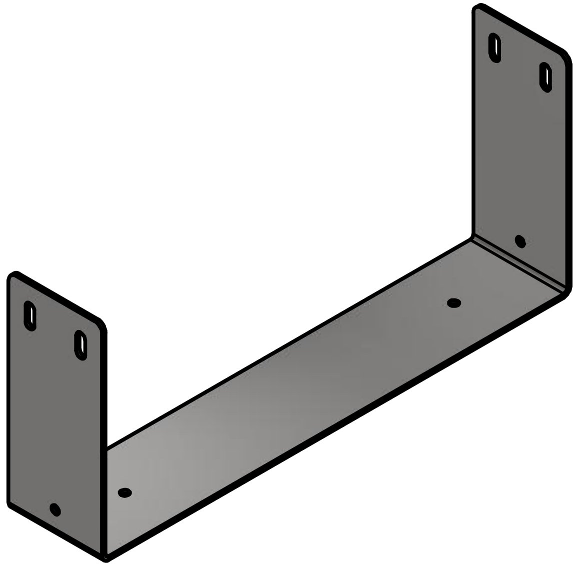
B-B Lid Plate

Swerve Drive AGV

Edition

Sheet 99 / 207





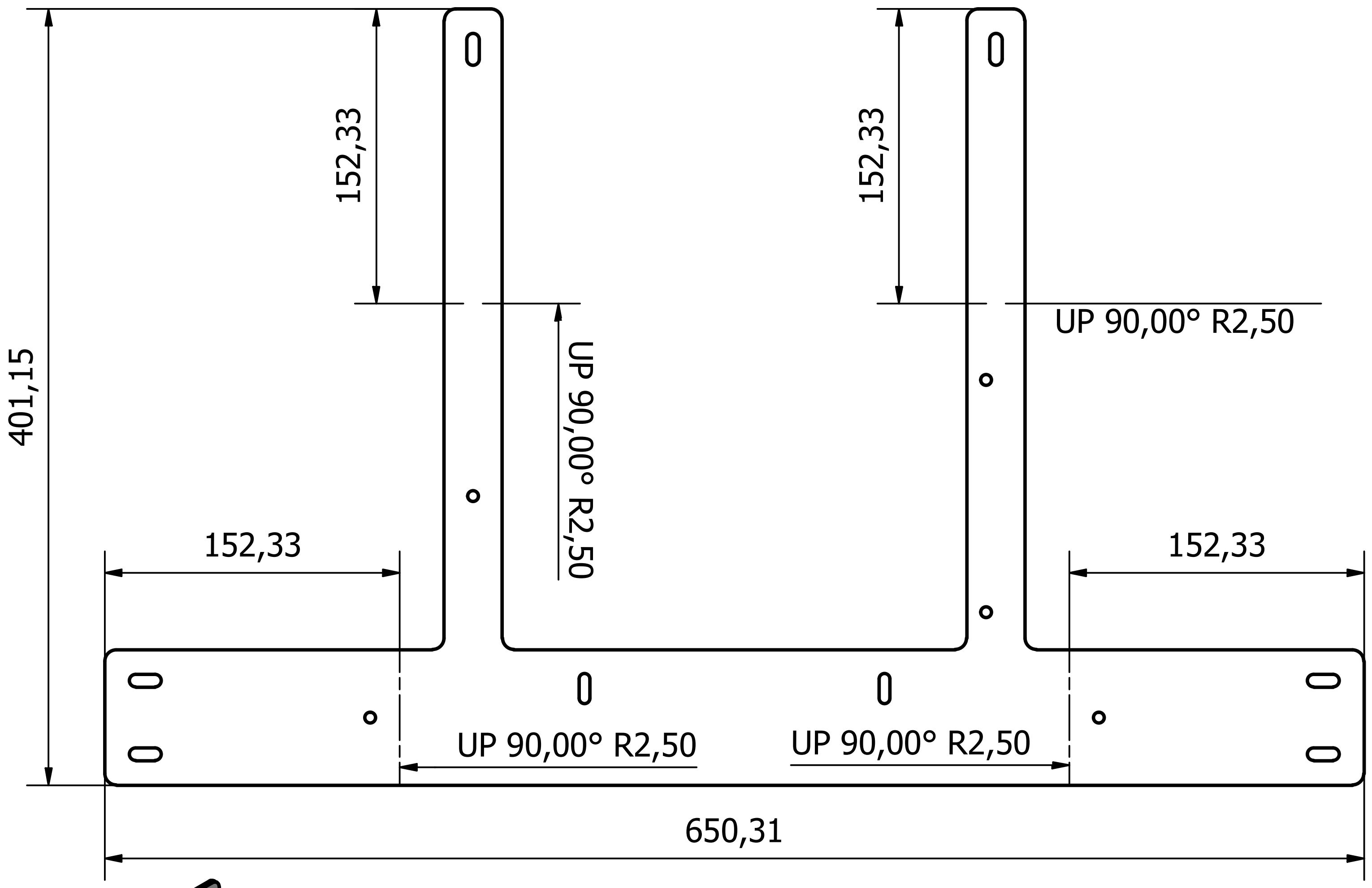
Sheet Thickness: 2,50mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	External	Steel, Mild	07/03/2019

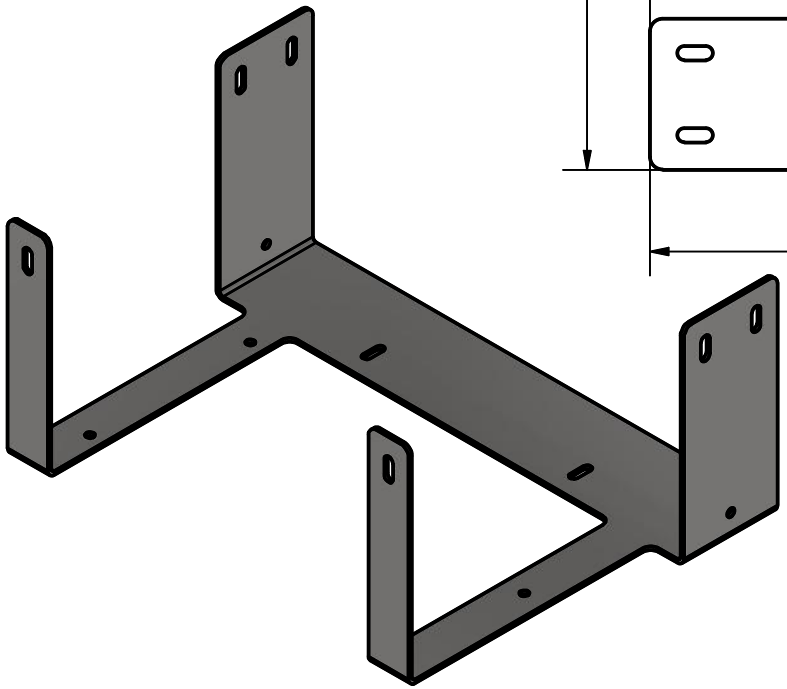
NELSON MANDELA UNIVERSITY

Inverter Support Type A

Swerve Drive AGV	Edition	Sheet
		100 / 207



Sheet Thickness: 2,50mm

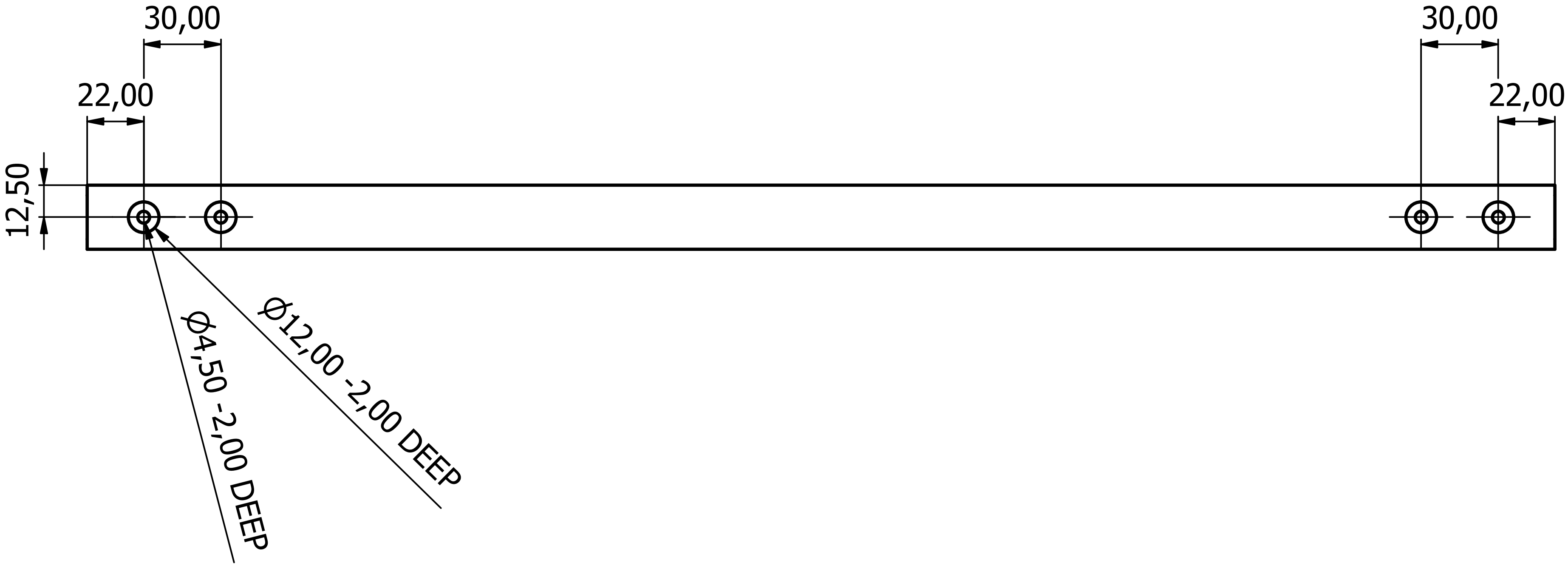


Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	External	Steel, Mild	07/03/2019

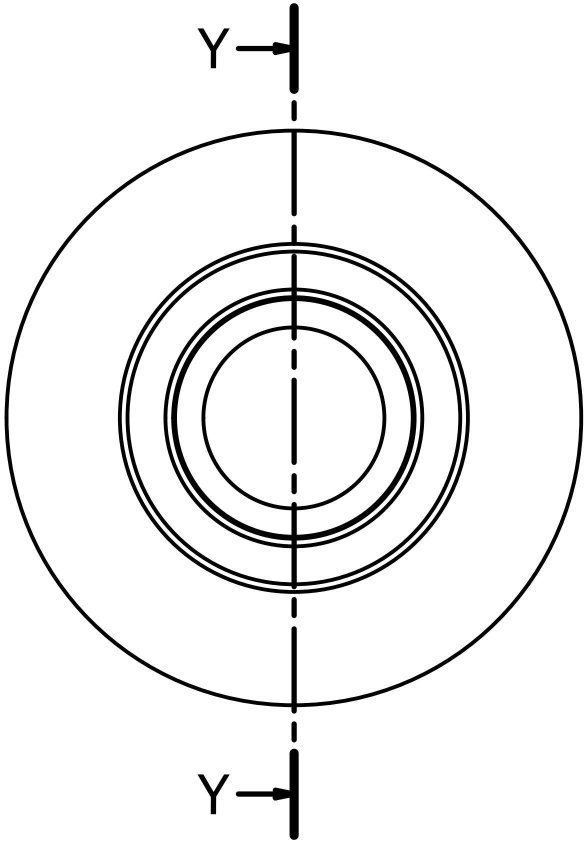
NELSON MANDELA UNIVERSITY

Inveter Support Type B

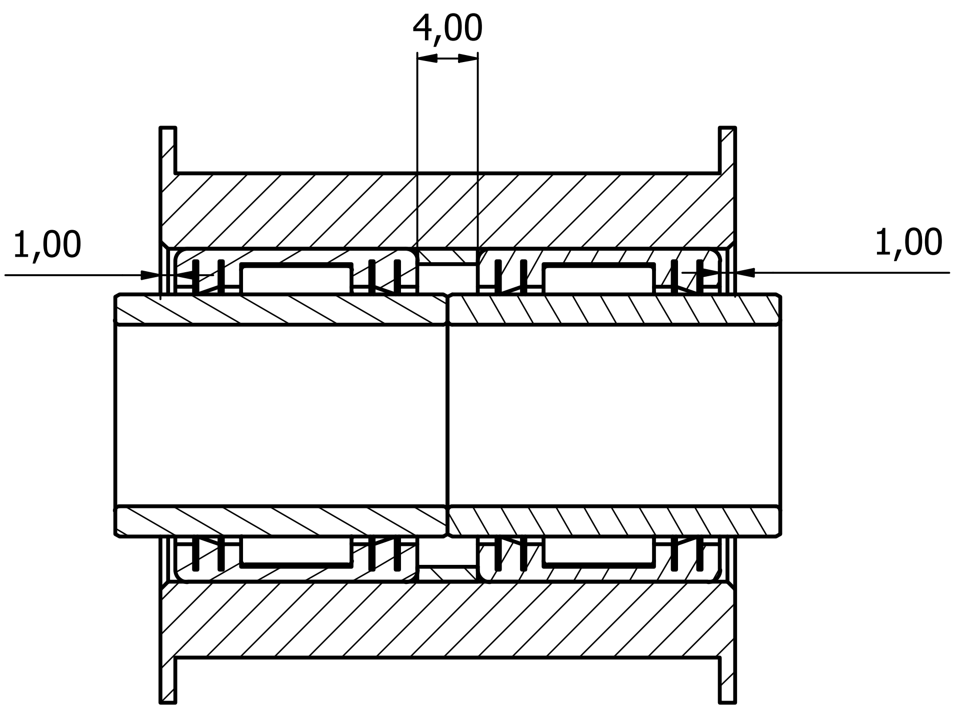
Swerve Drive AGV	Edition	Sheet
		101 / 207



Designed by AlexM	Quantity 2	Manufactured by Alex Macfarlane	Material	Date 07/03/2019
NELSON MANDELA UNIVERSITY	Support Bar			
	Swerve Drive AGV		Edition	Sheet 102 / 207



Y-Y (2 : 1)

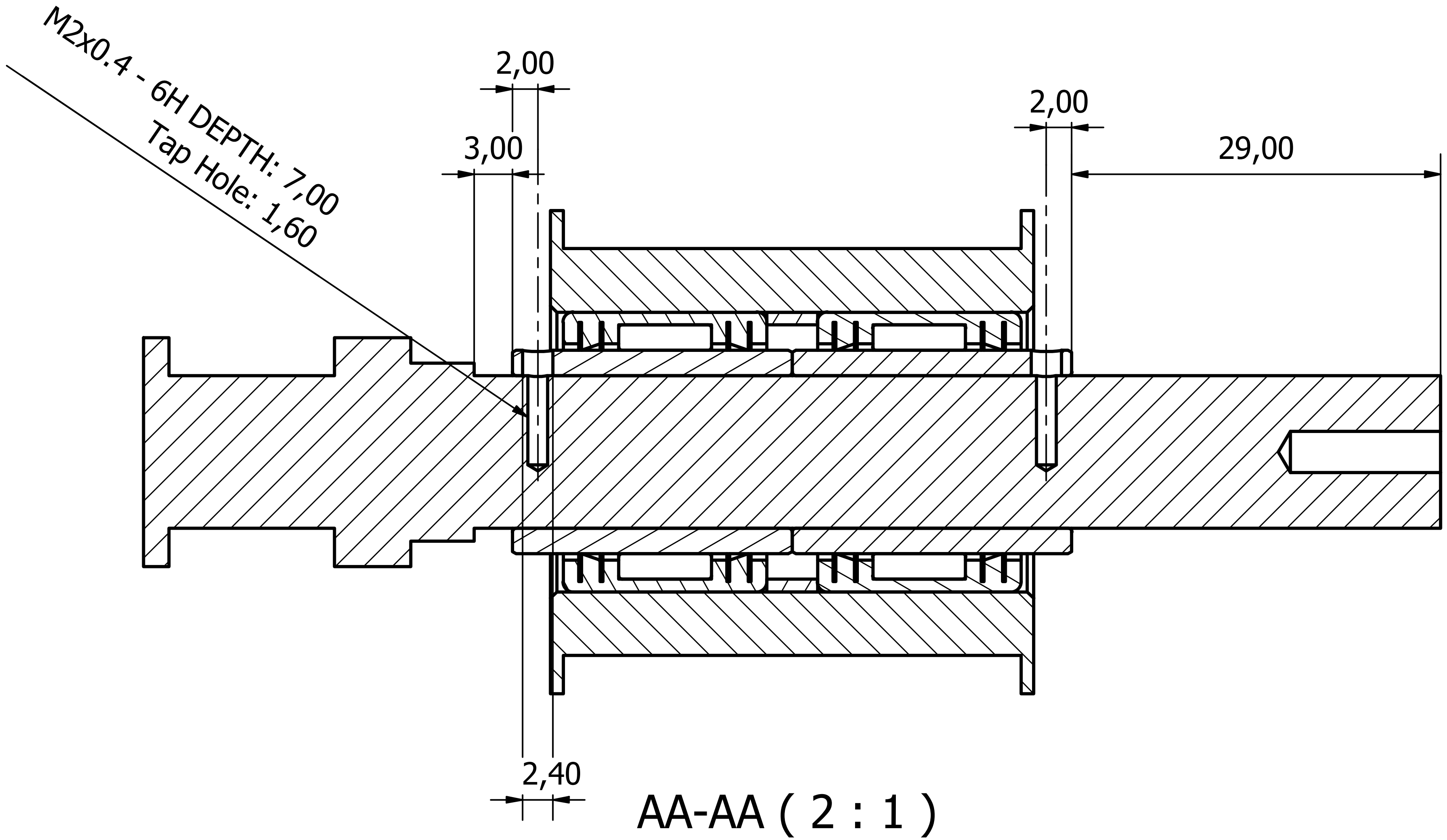


Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	Alex Macfarlane		07/03/2019

NELSON MANDELA UNIVERSITY

Smooth Idler Pulley

	Edition	Sheet
Swerve Drive AGV		103 / 207



AA-AA (2 : 1)

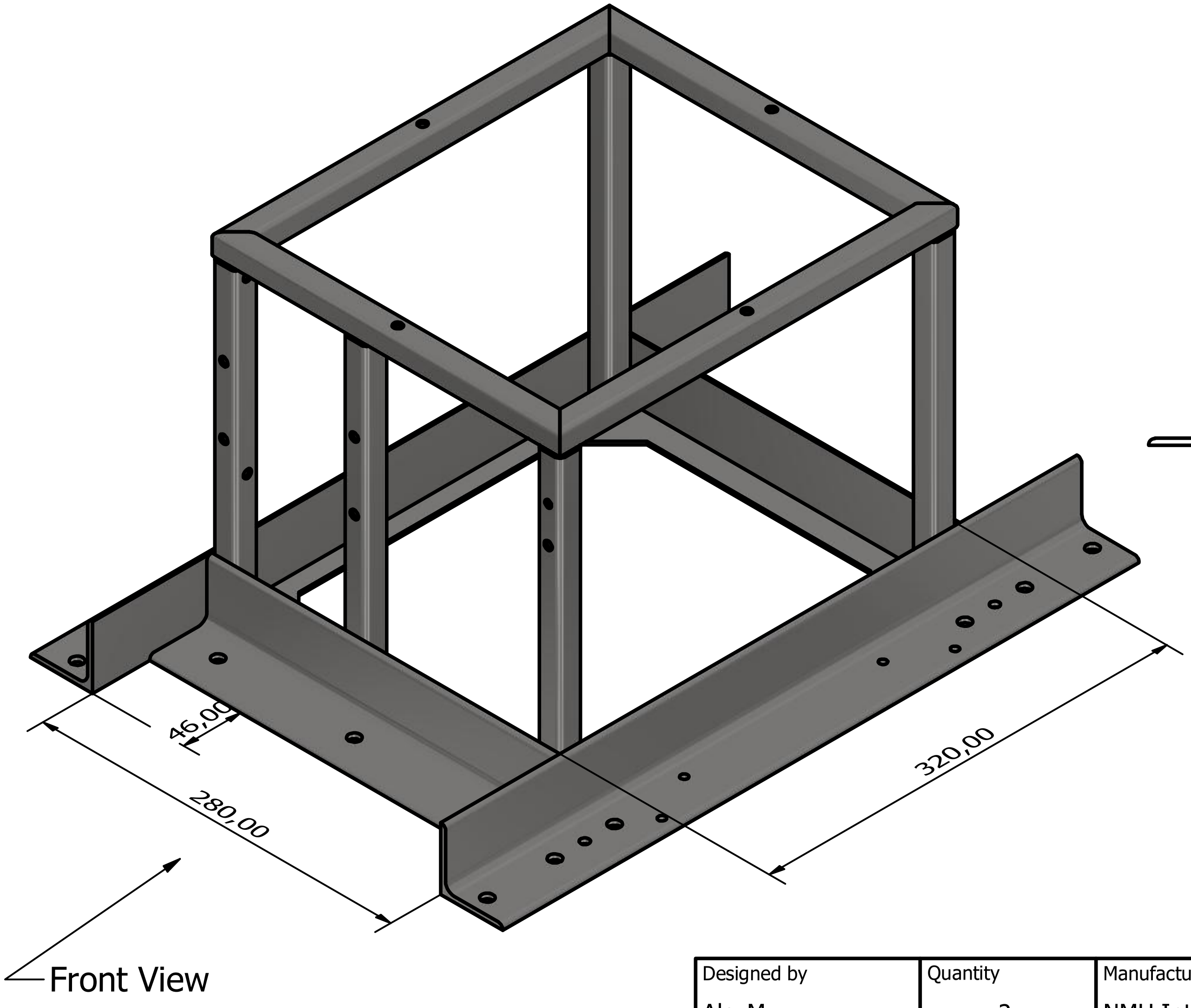
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	Alex Macfarlane		07/03/2019

NELSON MANDELA UNIVERSITY

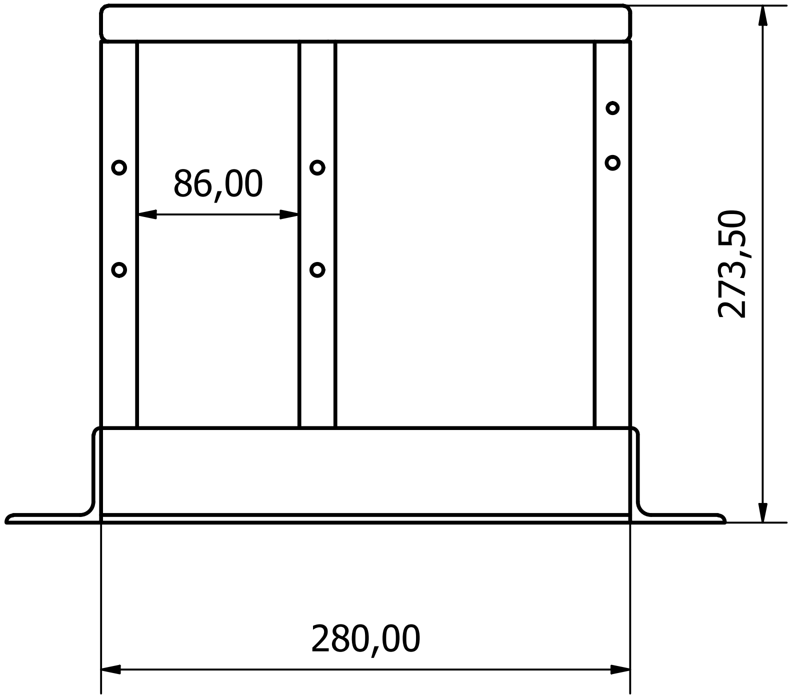
Lower Idler Pulley Assembly

Swerve Drive AGV

Edition	Sheet
	104 / 207



Front View



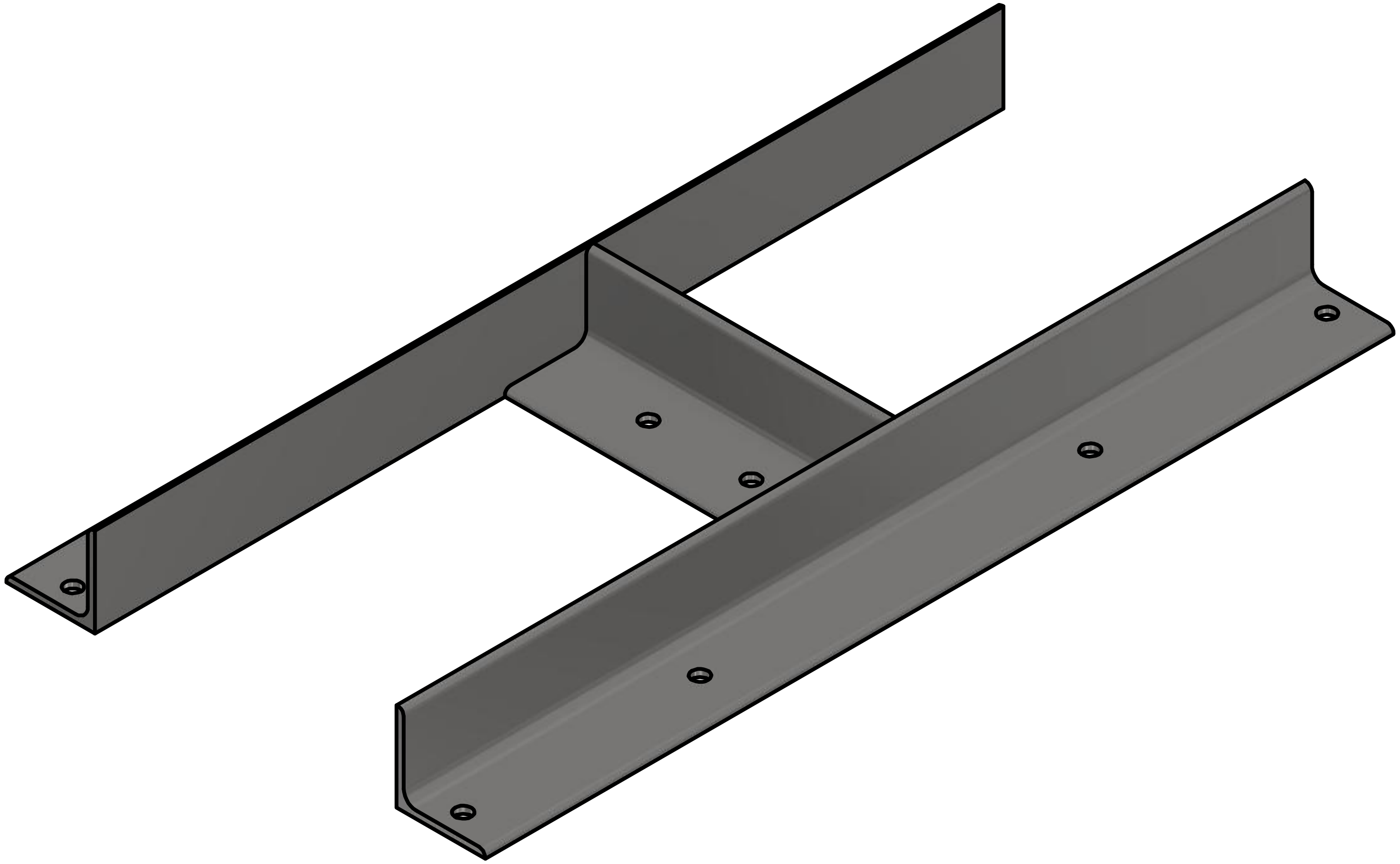
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	NMU Internal	Steel, Mild, Welded	07/03/2019

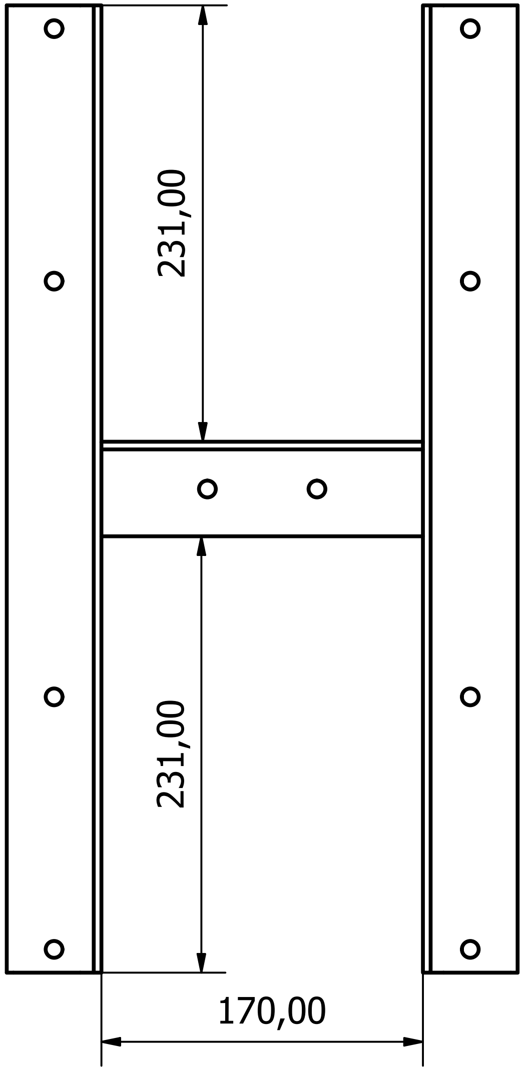
NELSON MANDELA UNIVERSITY

Static Interface Frame Upper Section

Swerve Drive AGV

Edition	Sheet
	105 / 207





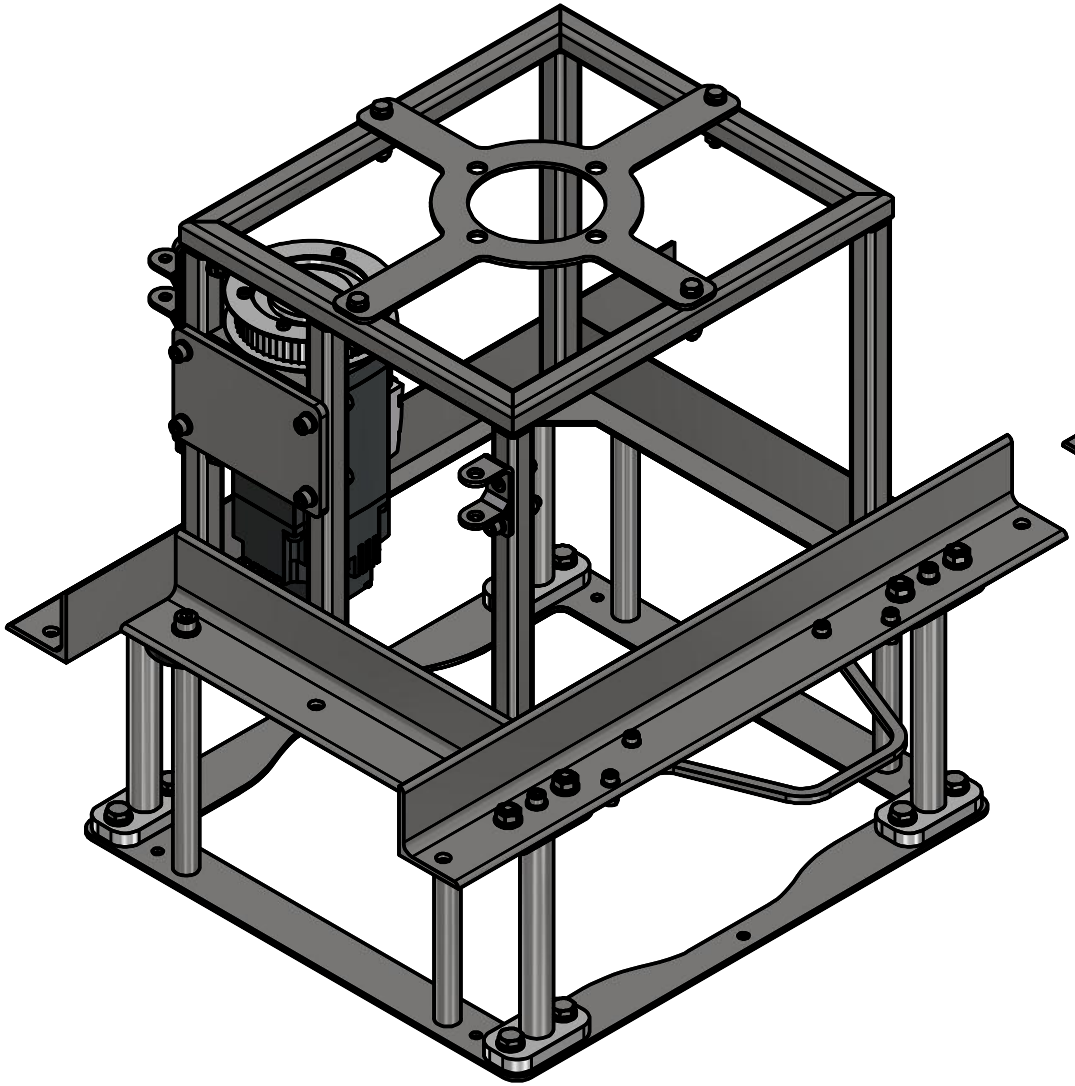
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	NMU Internal	Steel, Mild, Welded	07/03/2019

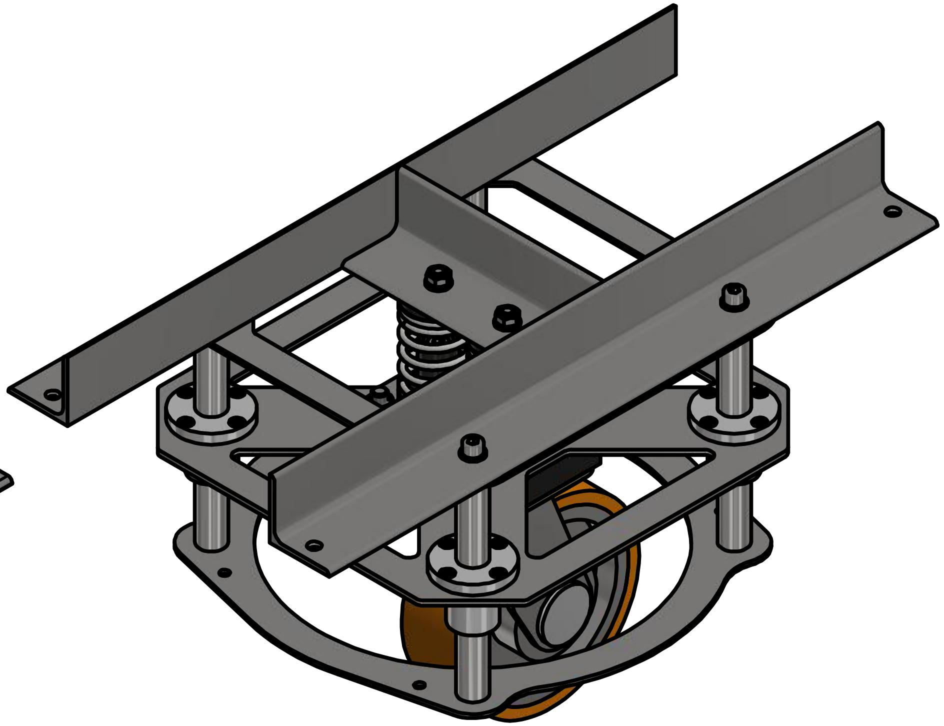
NELSON MANDELA UNIVERSITY

Caster Connection Flange

Swerve Drive AGV

Edition	Sheet
	106 / 207





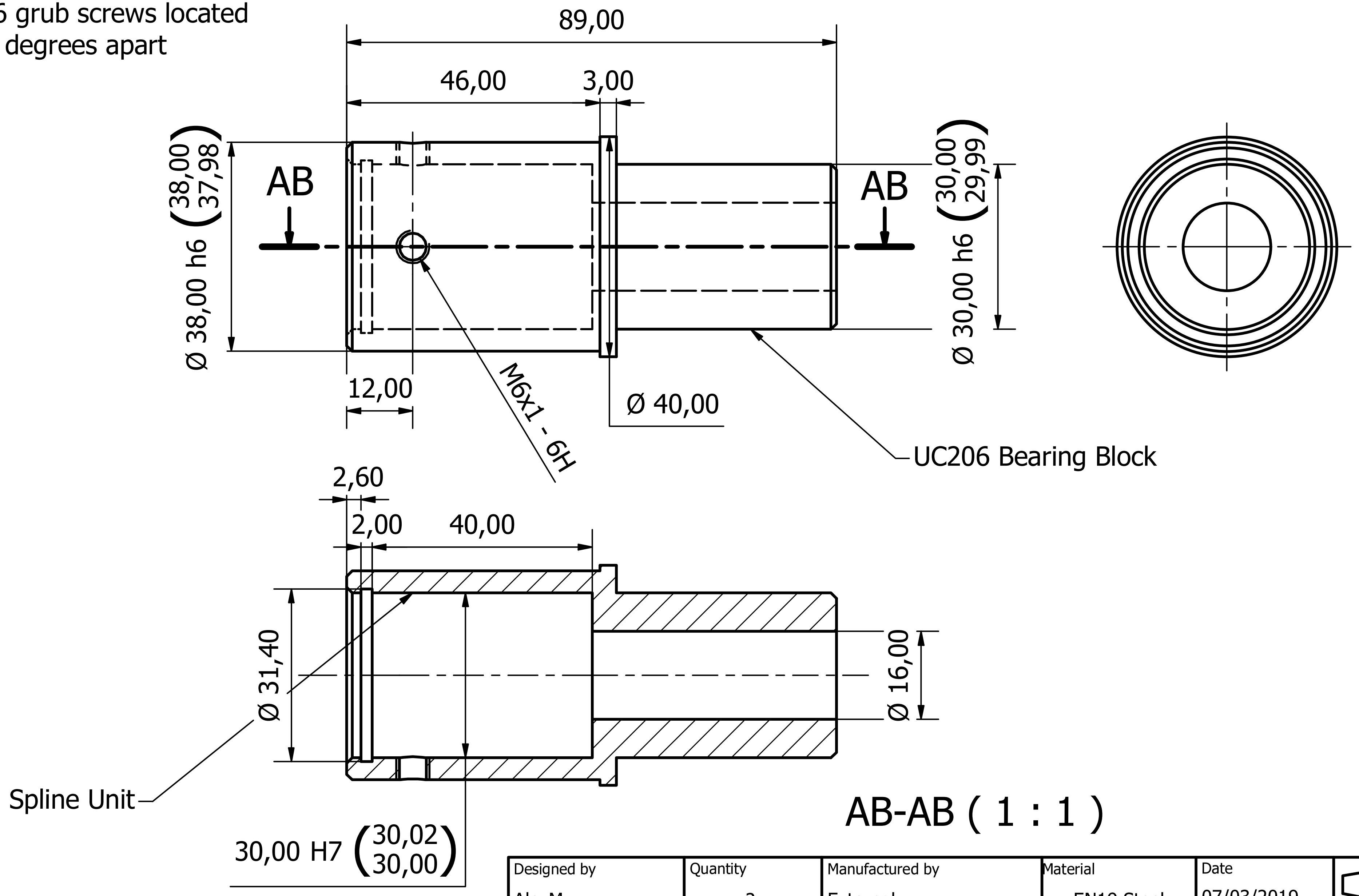
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	Alex Macfarlane		07/03/2019

NELSON MANDELA UNIVERSITY

Unit Assemblies

Swerve Drive AGV	Edition	Sheet
		107 / 207

2x M6 grub screws located
at 90 degrees apart



AB-AB (1 : 1)

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	EN19 Steel	07/03/2019

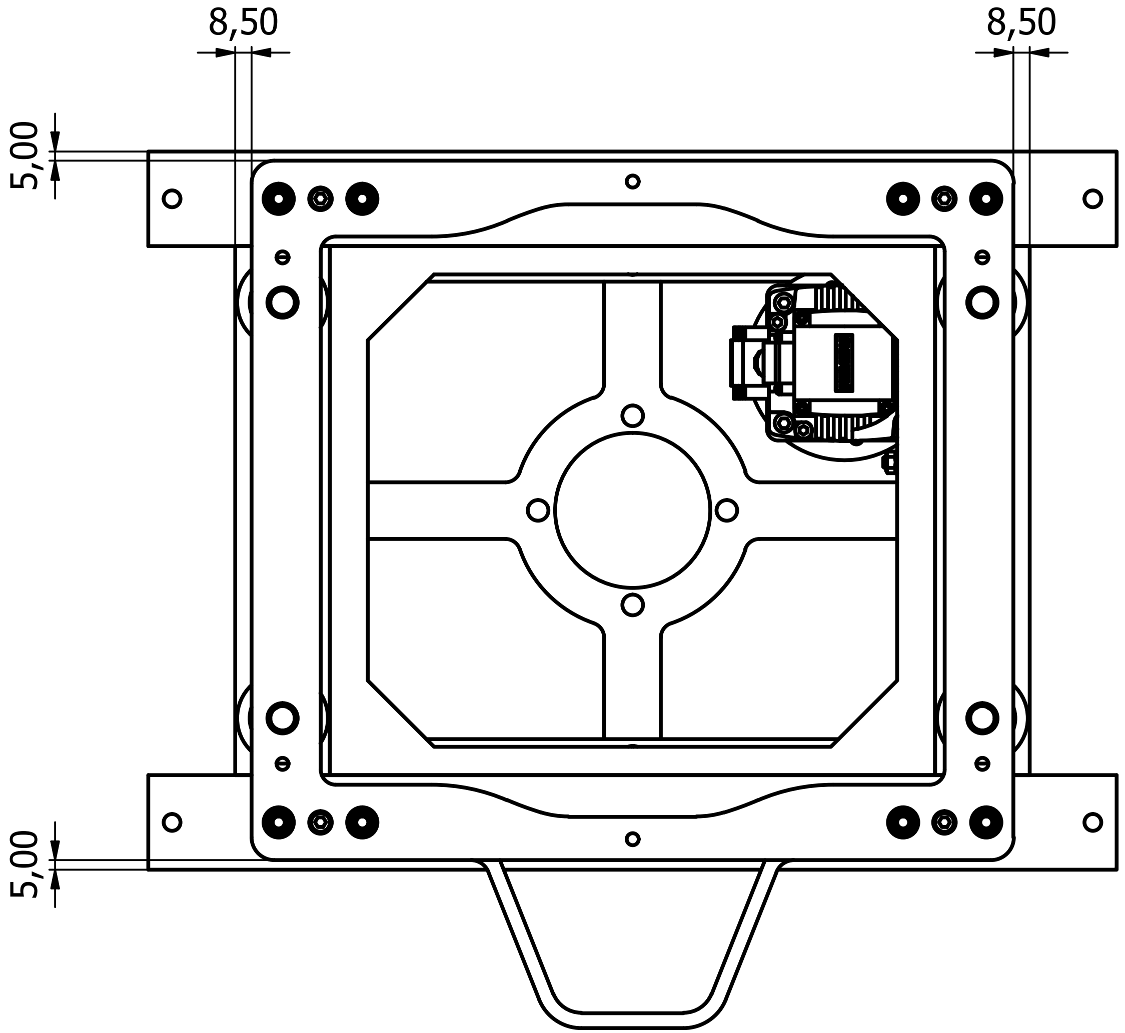
NELSON MANDELA UNIVERSITY

Compensator Main Housing

Swerve Drive AGV

Edition	Sheet
	108 / 207

Chamfers: 1mm x 45°



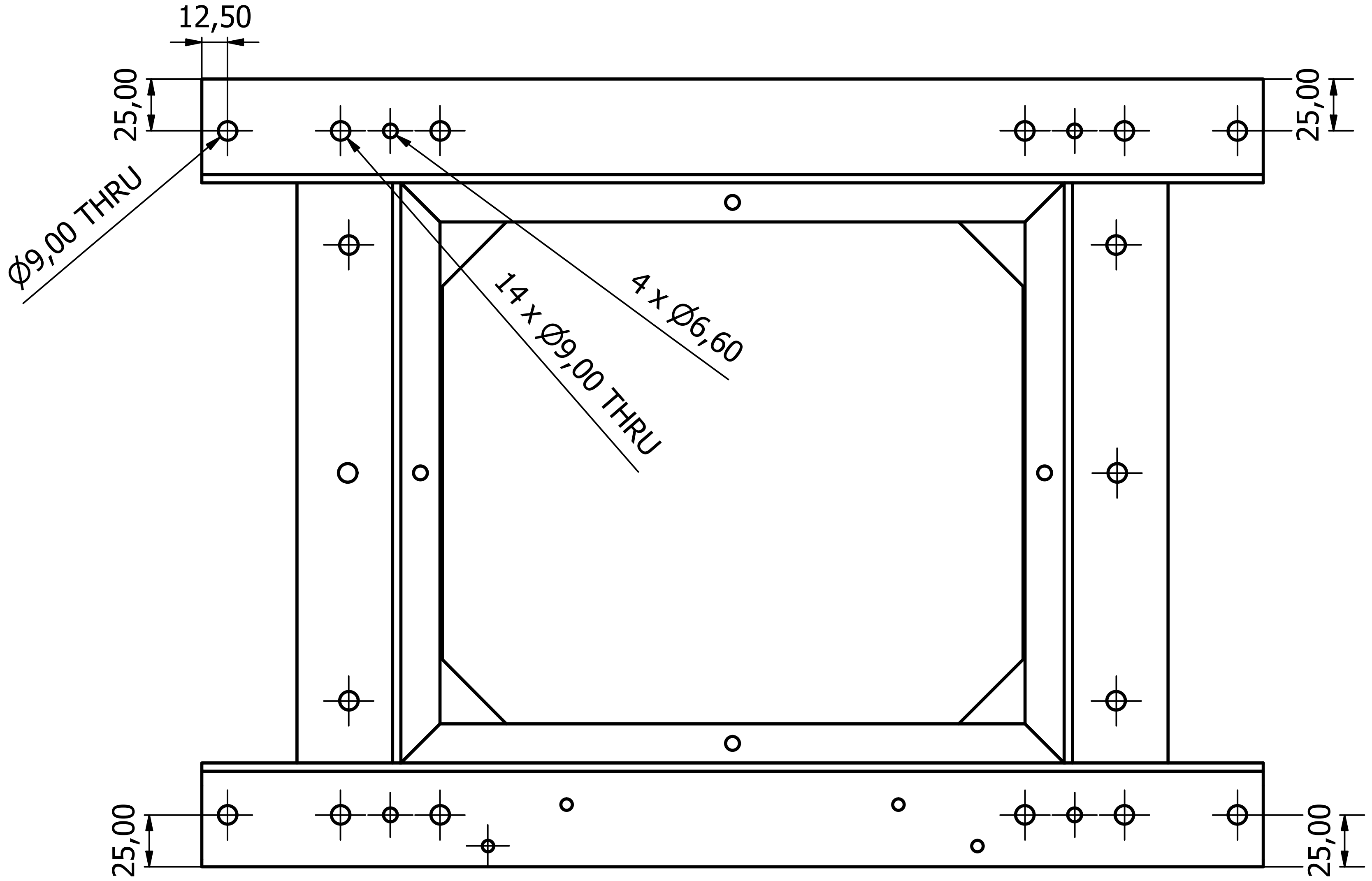
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	Alex Macfarlane		07/03/2019

NELSON MANDELA UNIVERSITY

Static Interface Frame Section Assembly

Swerve Drive AGV

Edition	Sheet
	109 / 207



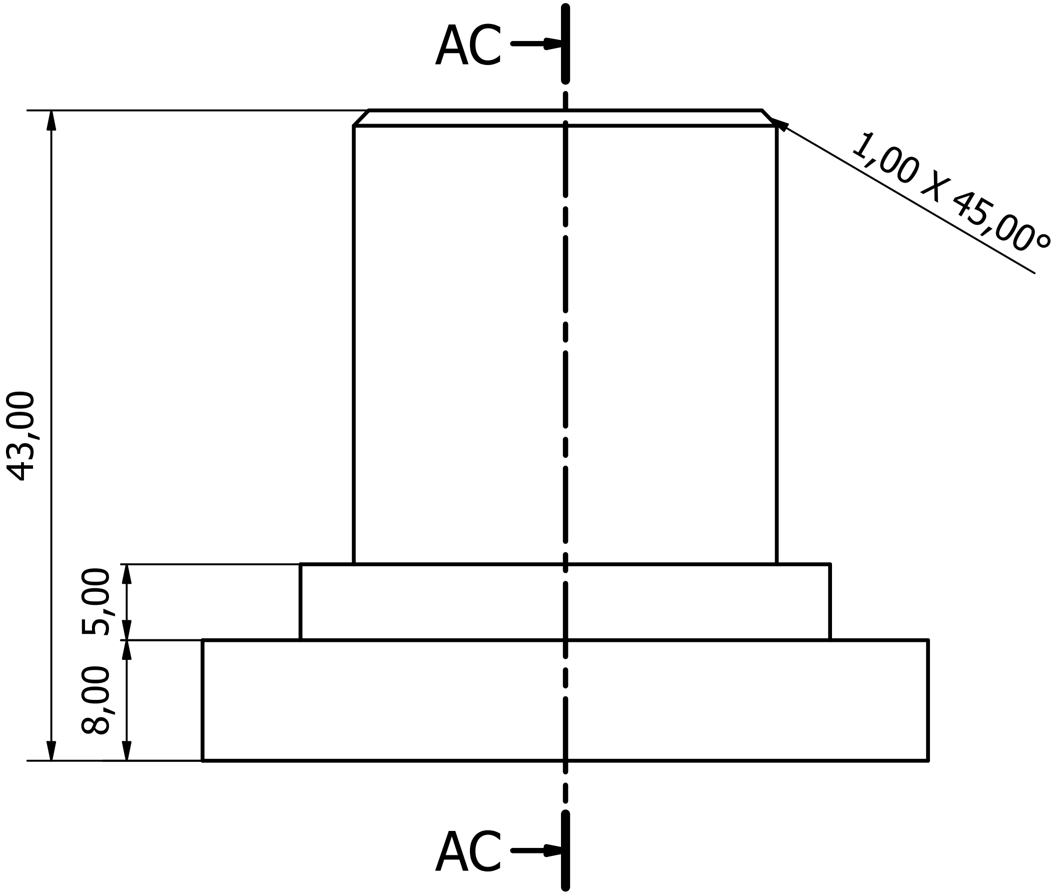
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	NMU Internal	Steel, Mild, Welded	07/03/2019

NELSON MANDELA UNIVERSITY

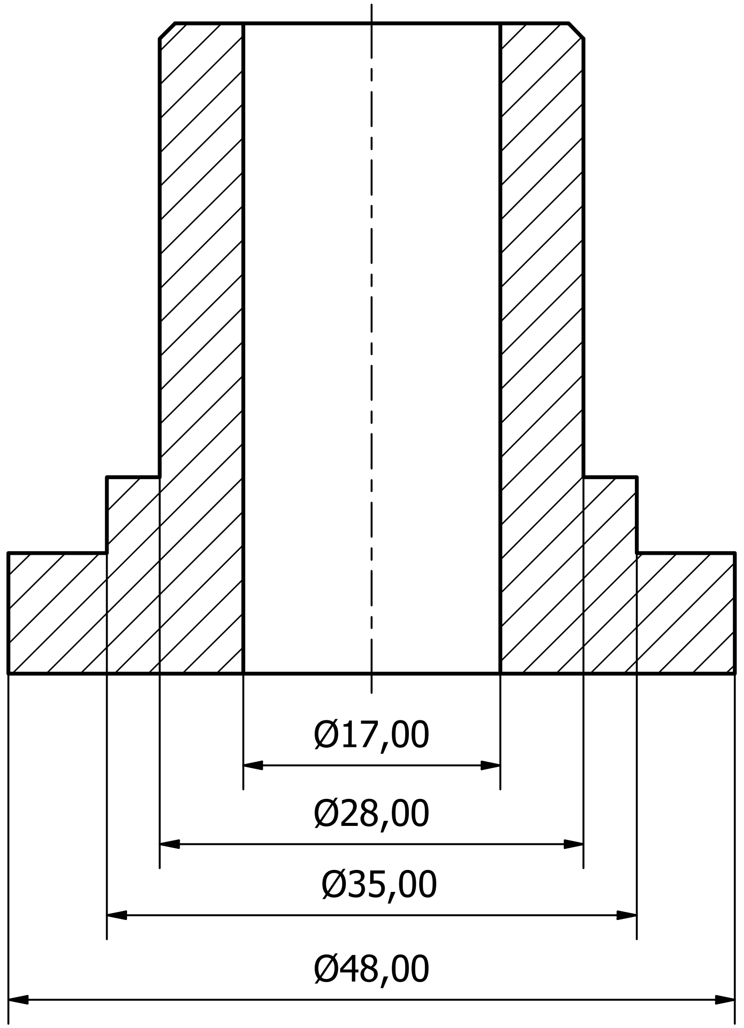
Static Interface Frame Upper Section

Swerve Drive AGV

Edition	Sheet
	110 / 207



AC-AC (2 : 1)



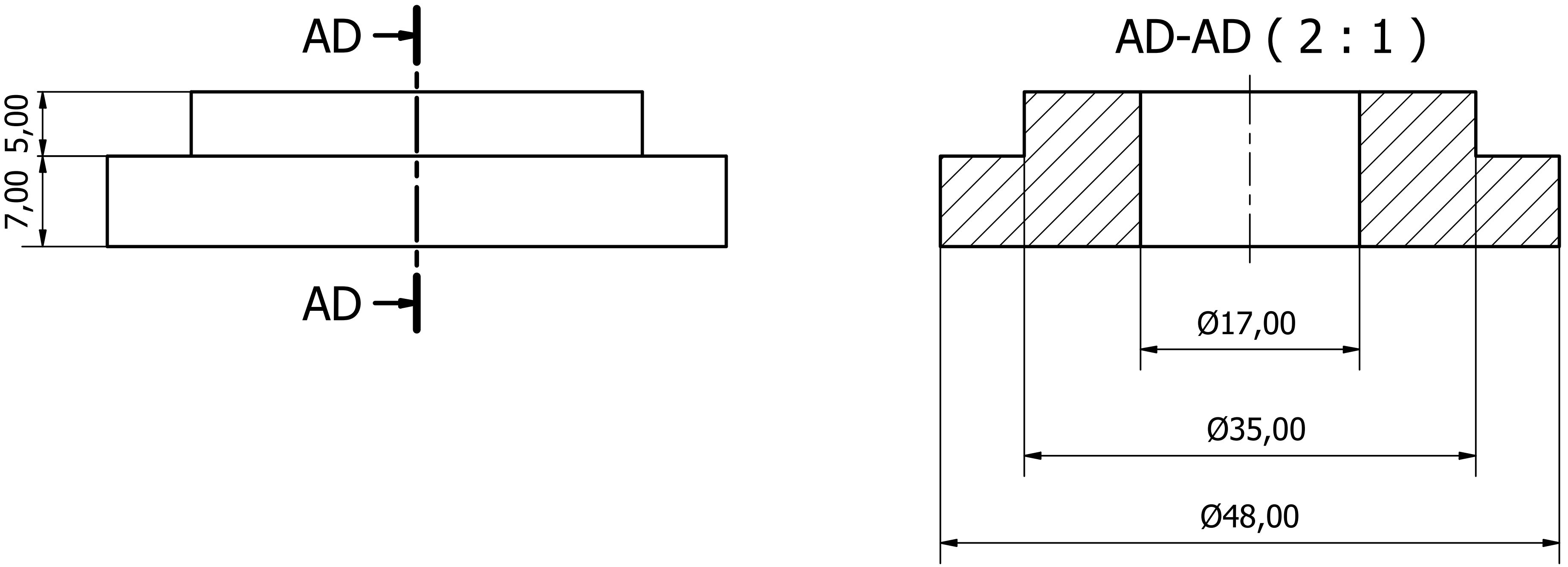
Chamfer is approximate

Designed by	Quantity	Manufactured by	Material	Date
AlexM	8		Polyethylene, High Density	07/03/2019

NELSON MANDELA UNIVERSITY

Swerve Drive AGV

Edition	Sheet
	111 / 207



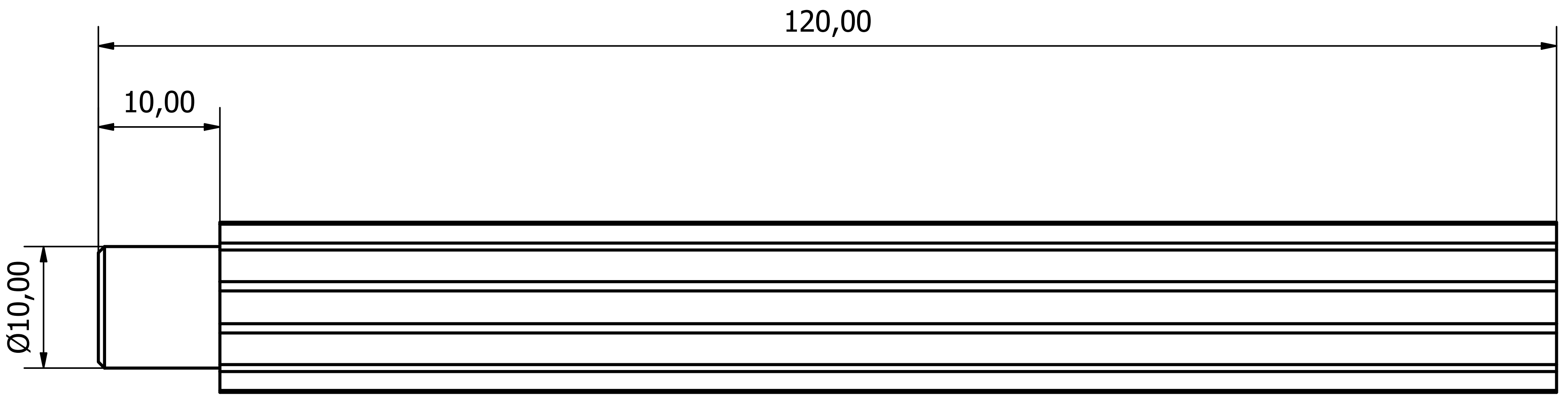
Designed by	Quantity	Manufactured by	Material	Date
AlexM	8		Polyethylene, High Density	07/03/2019

NELSON MANDELA UNIVERSITY

Spring Rest Top

Swerve Drive AGV

Edition	Sheet
	112 / 207



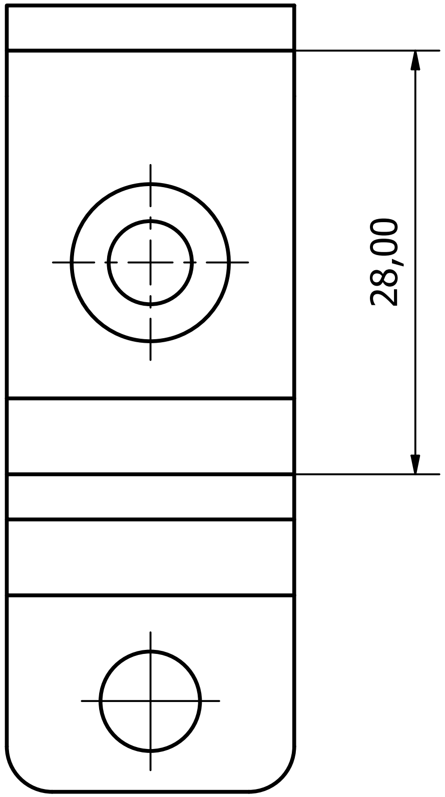
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	Alex Macfarlane	Steel, High Strength, Low Alloy	07/03/2019

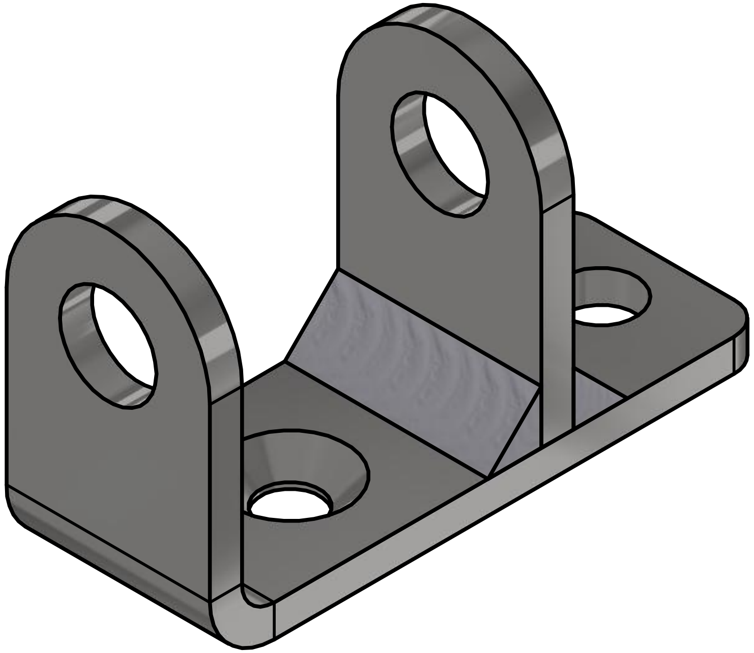
NELSON MANDELA UNIVERSITY

Plain Splined Shaft

Swerve Drive AGV

Edition	Sheet
	113 / 207

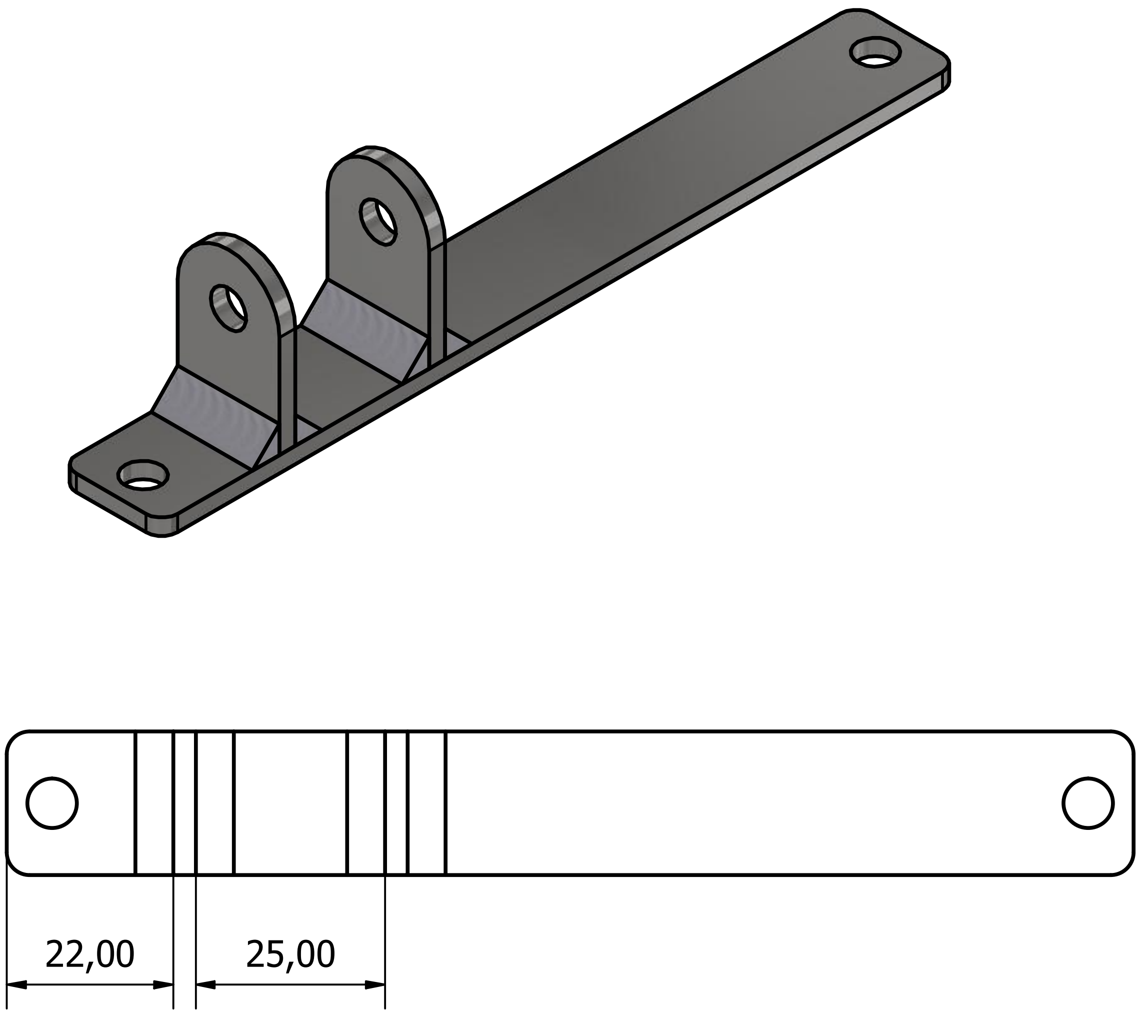




Designed by	Quantity	Manufactured by	Material	Date
AlexM	2		Welded Aluminum-6061	07/03/2019

NELSON MANDELA
UNIVERSITY

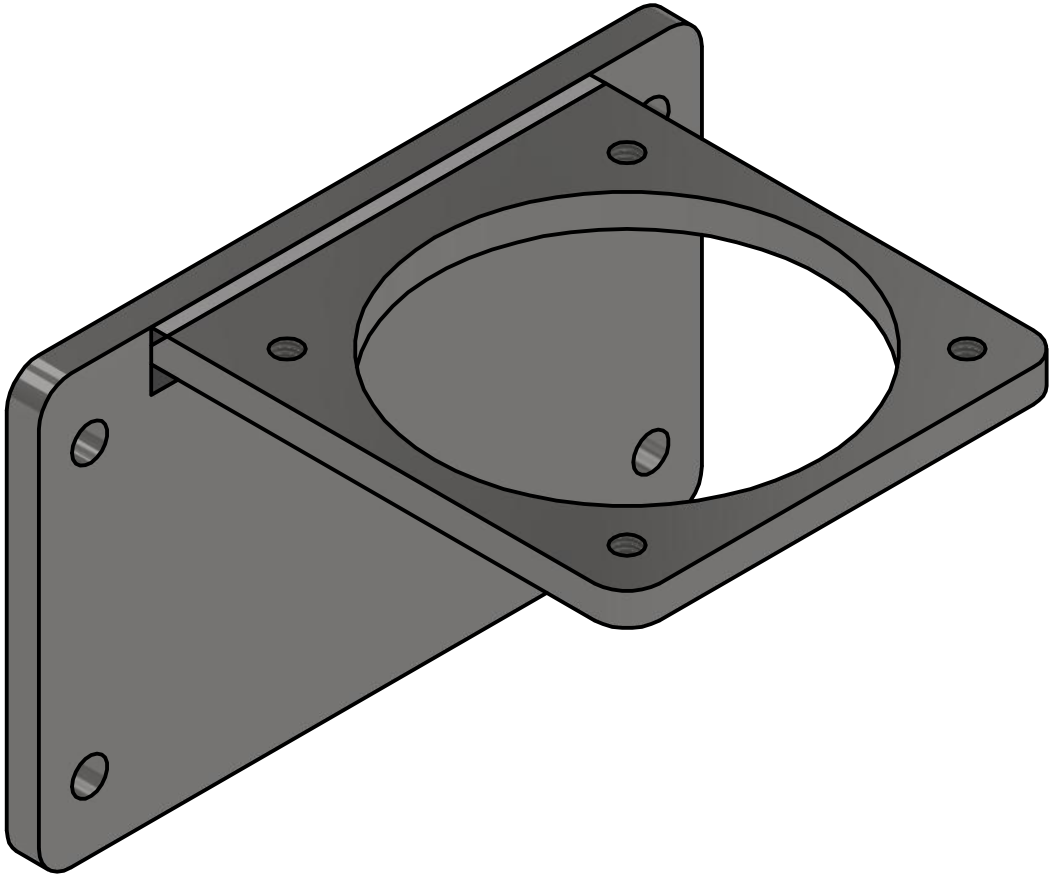
	Edition	Sheet
Swerve Drive AGV		114 / 207

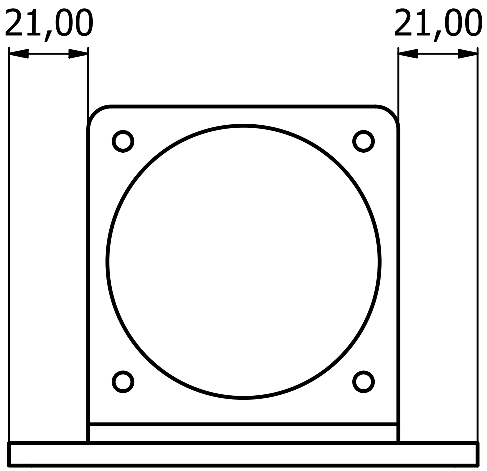


Designed by	Quantity	Manufactured by	Material	Date
AlexM	2		Welded Aluminum-6061	07/03/2019

NELSON MANDELA UNIVERSITY

	Edition	Sheet
Swerve Drive AGV		115 / 207





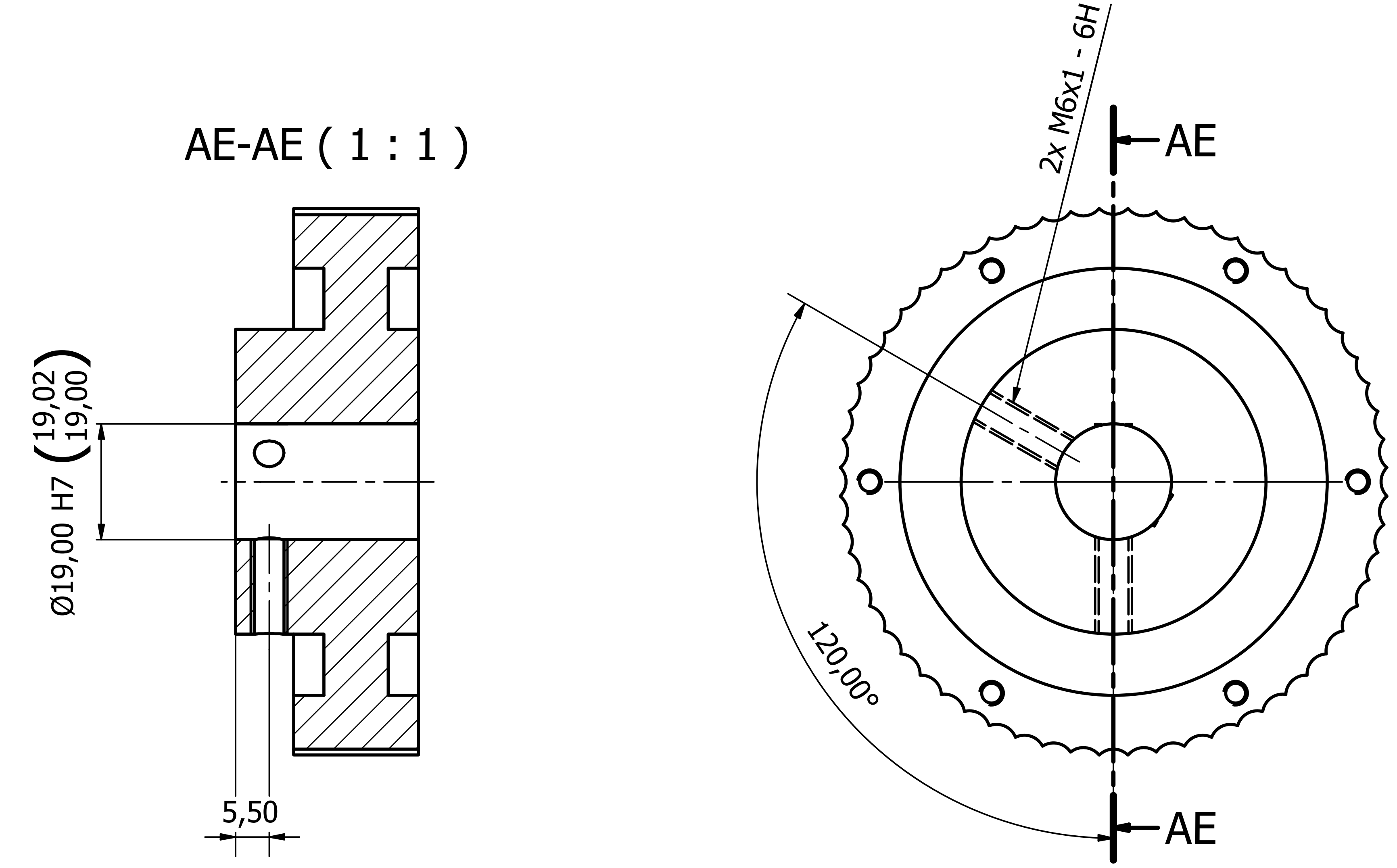
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	LPB	Steel, Mild, Welded	07/03/2019

NELSON MANDELA UNIVERSITY

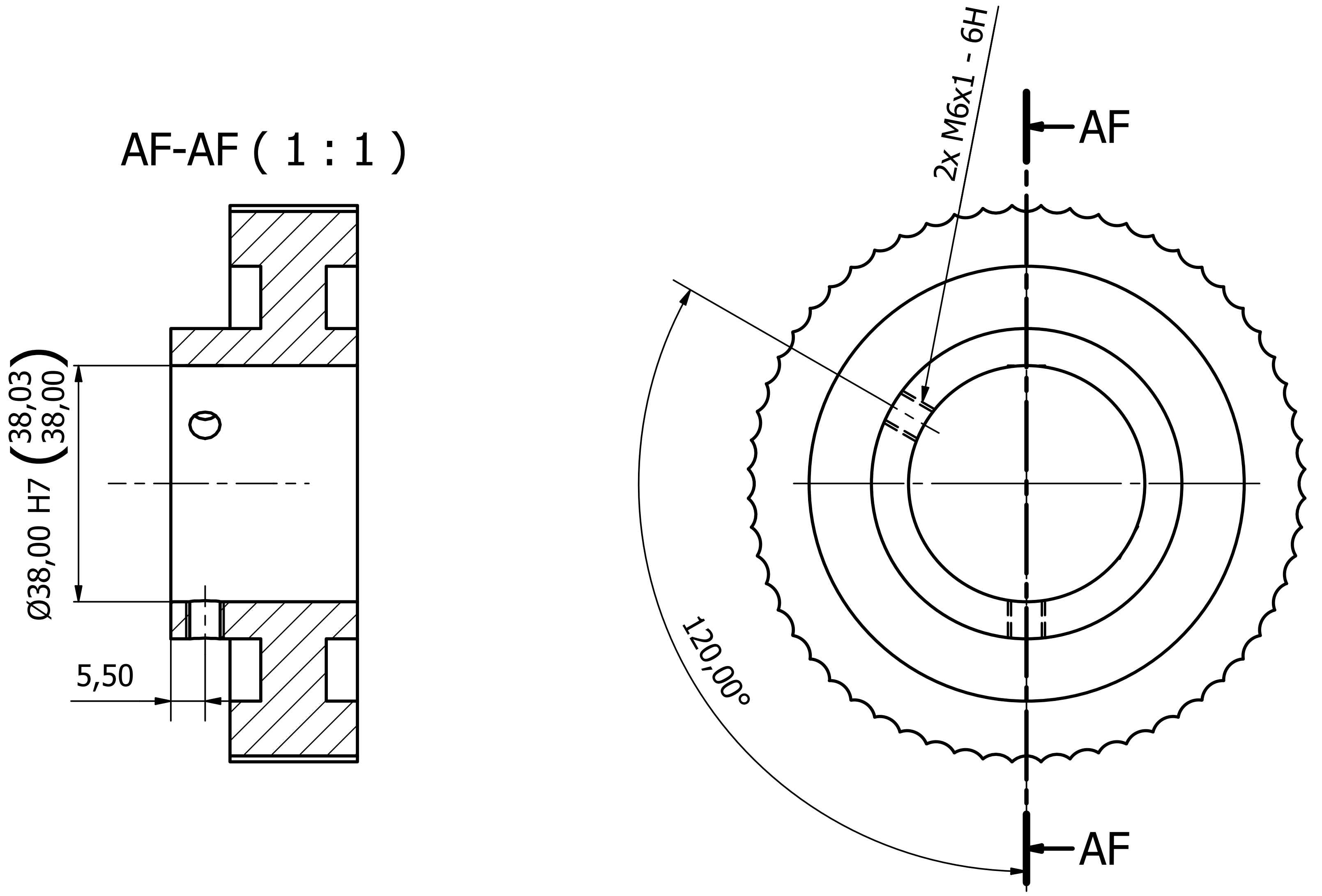
Traction Motor Mount Weldmatt

Swerve Drive AGV

Edition	Sheet
	116 / 207



Designed by AlexM	Quantity 2	Manufactured by	Material Aluminum 6061	Date 07/03/2019
NELSON MANDELA UNIVERSITY	PHP 60-5m-15-RSB Motor Side			
	Swerve Drive AGV		Edition	Sheet 117 / 207



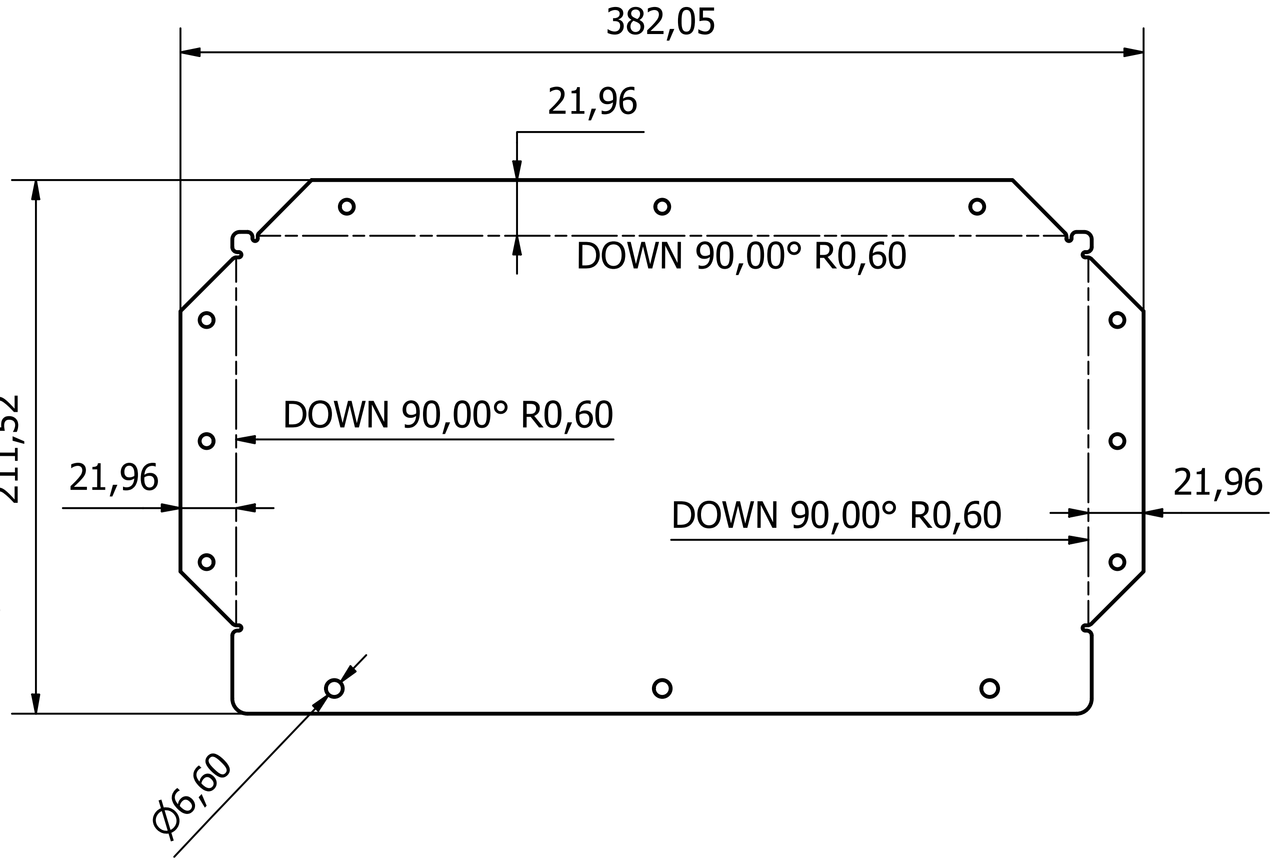
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2		Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

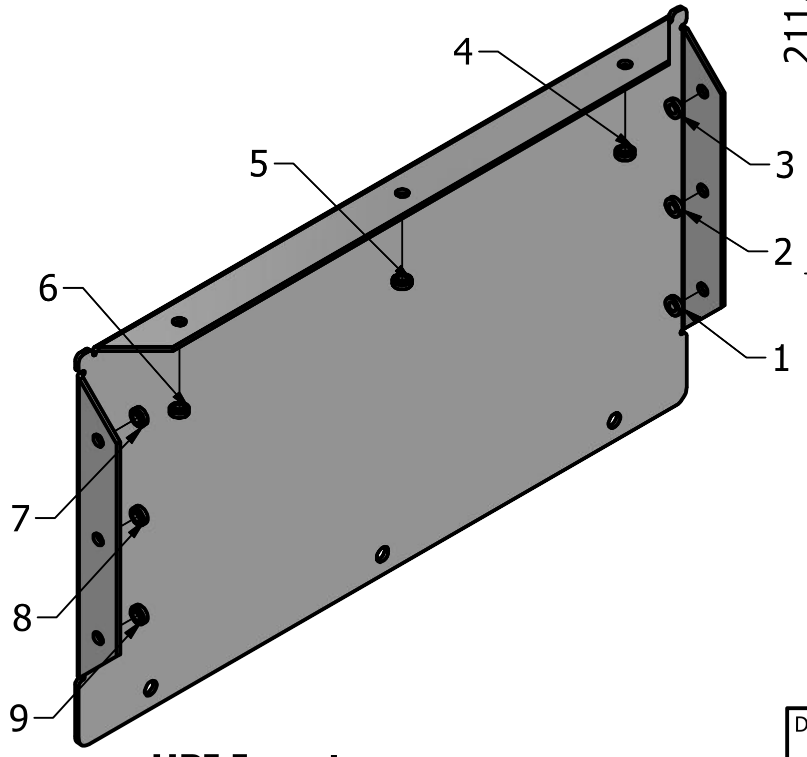
PHP 60-5M-15-RSB Gear Side

Swerve Drive AGV

Edition	Sheet
	118 / 207



Sheet Thickness: 2,00mm



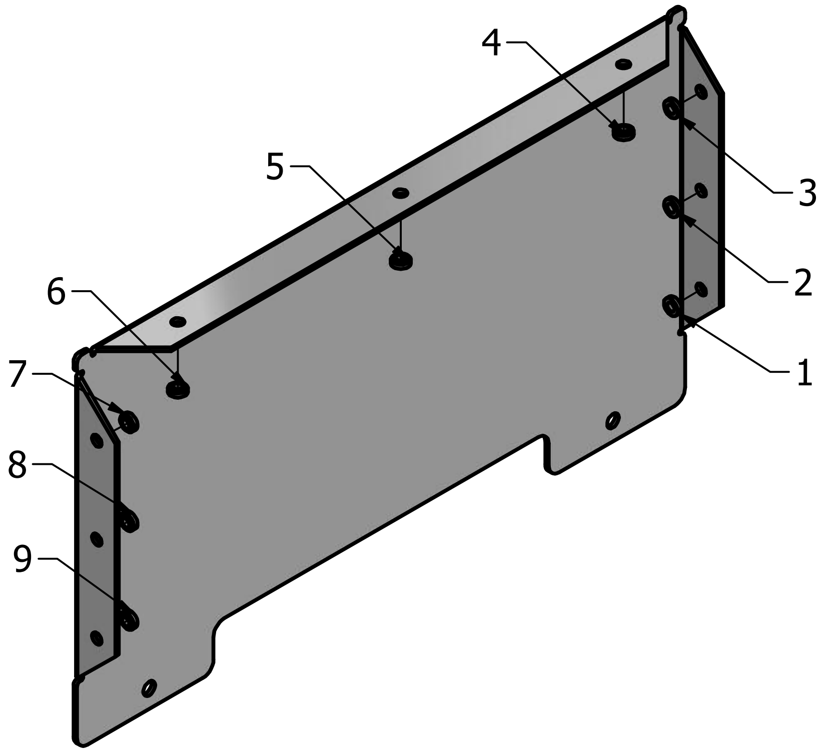
HPI Inserts
Number : 9 unit
Designation : CM4-0 (SCN M4)

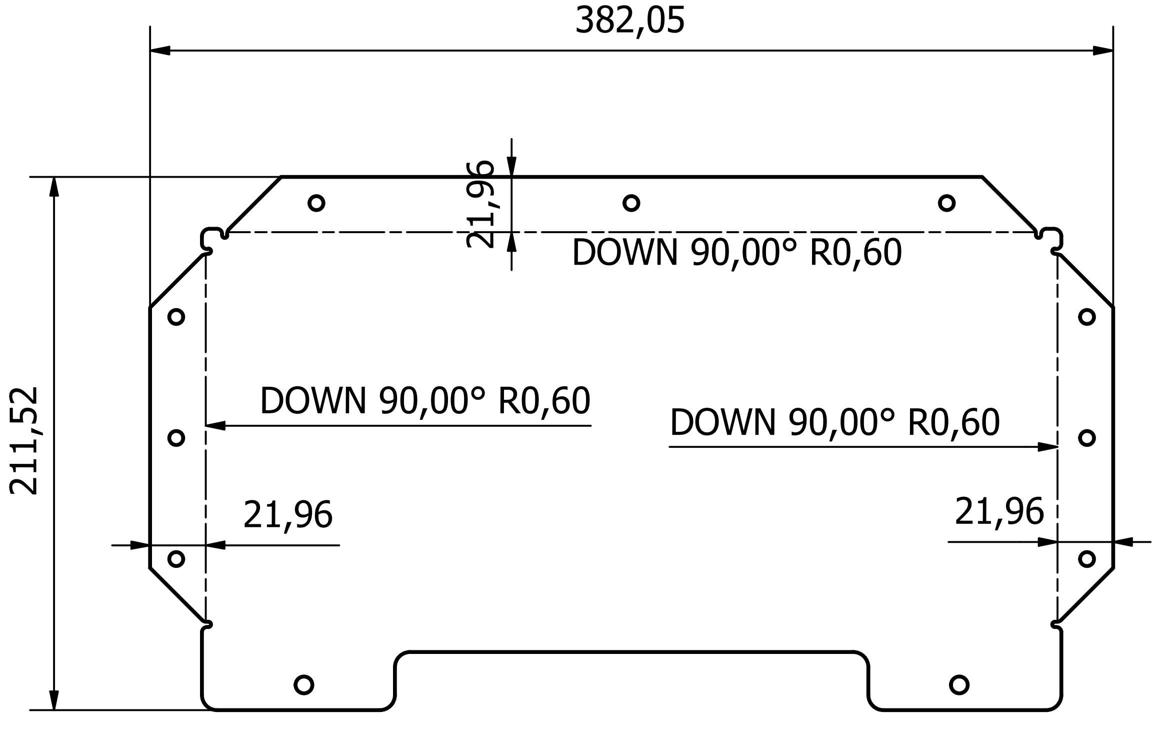
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	LPB	Aluminum	07/03/2019

NELSON MANDELA UNIVERSITY

Battery Bank Lid Side A

	Edition	Sheet
Swerve Drive AGV		119 / 207





Sheet Thickness: 2,00mm

HPI Inserts

Number : 9 unit

Designation : CM4-0 (SCN M4)

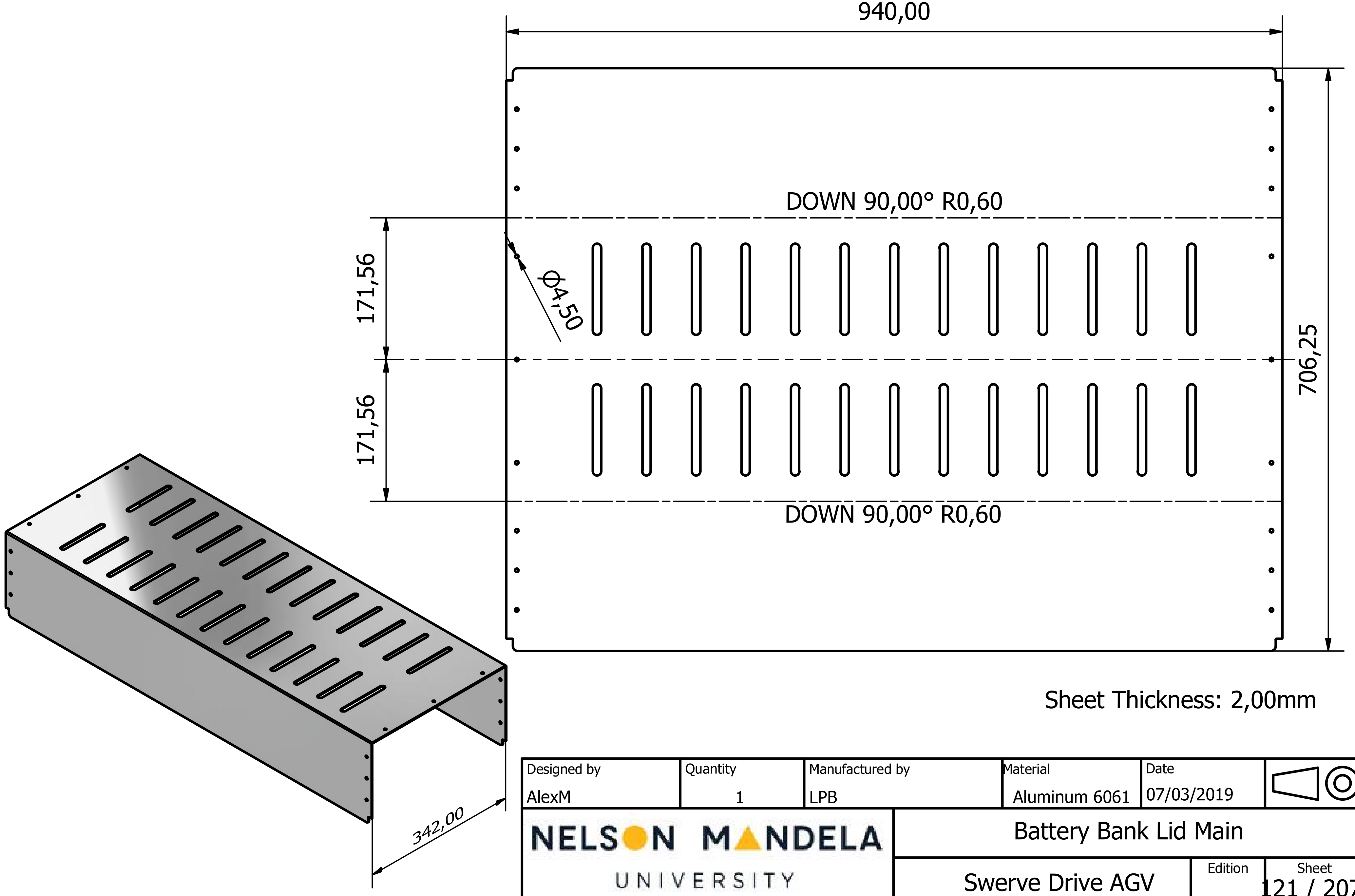
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	LPB	Aluminum	07/03/2019

NELSON MANDELA UNIVERSITY

Battery Bank Lid Side B

Swerve Drive AGV

Edition	Sheet
	120 / 207



Sheet Thickness: 2,00mm

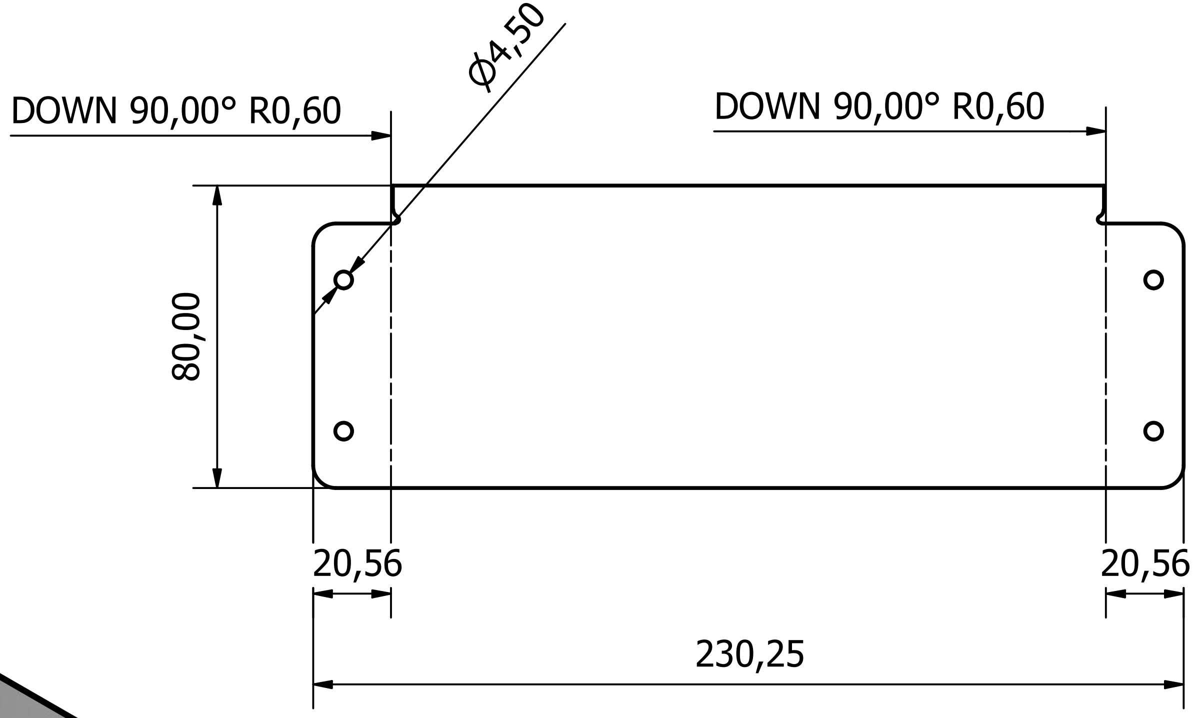
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	LPB	Aluminum 6061	07/03/2019

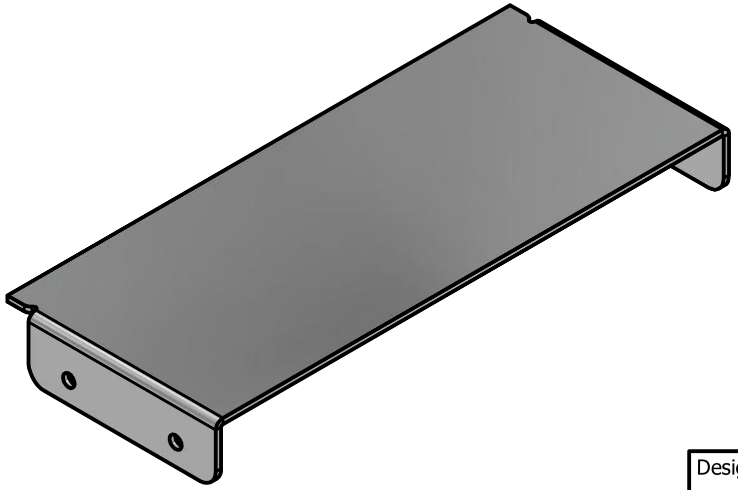
NELSON MANDELA UNIVERSITY

Battery Bank Lid Main

Swerve Drive AGV

Edition	Sheet
	121 / 207





Sheet Thickness: 2,00mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	LPB	Aluminum 6061	07/03/2019

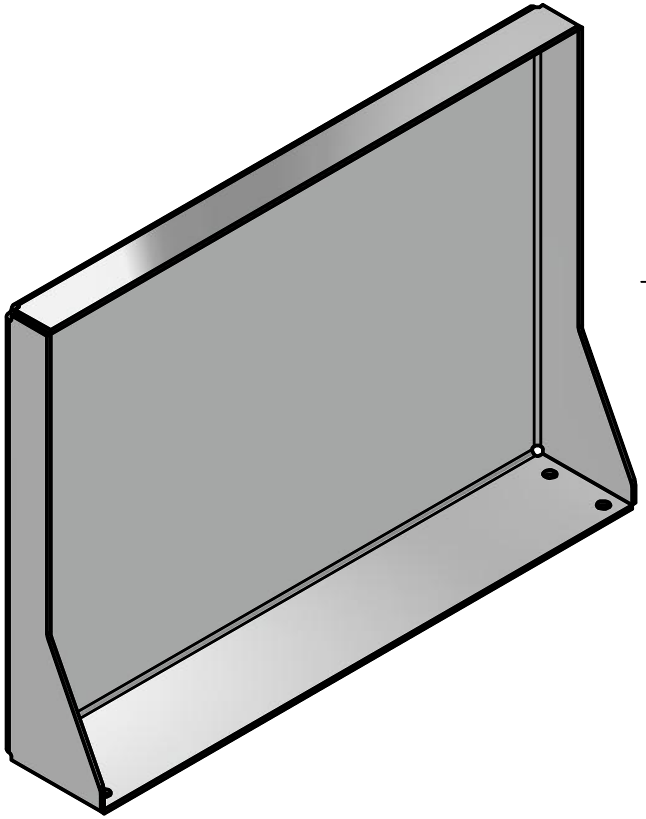
NELSON MANDELA UNIVERSITY

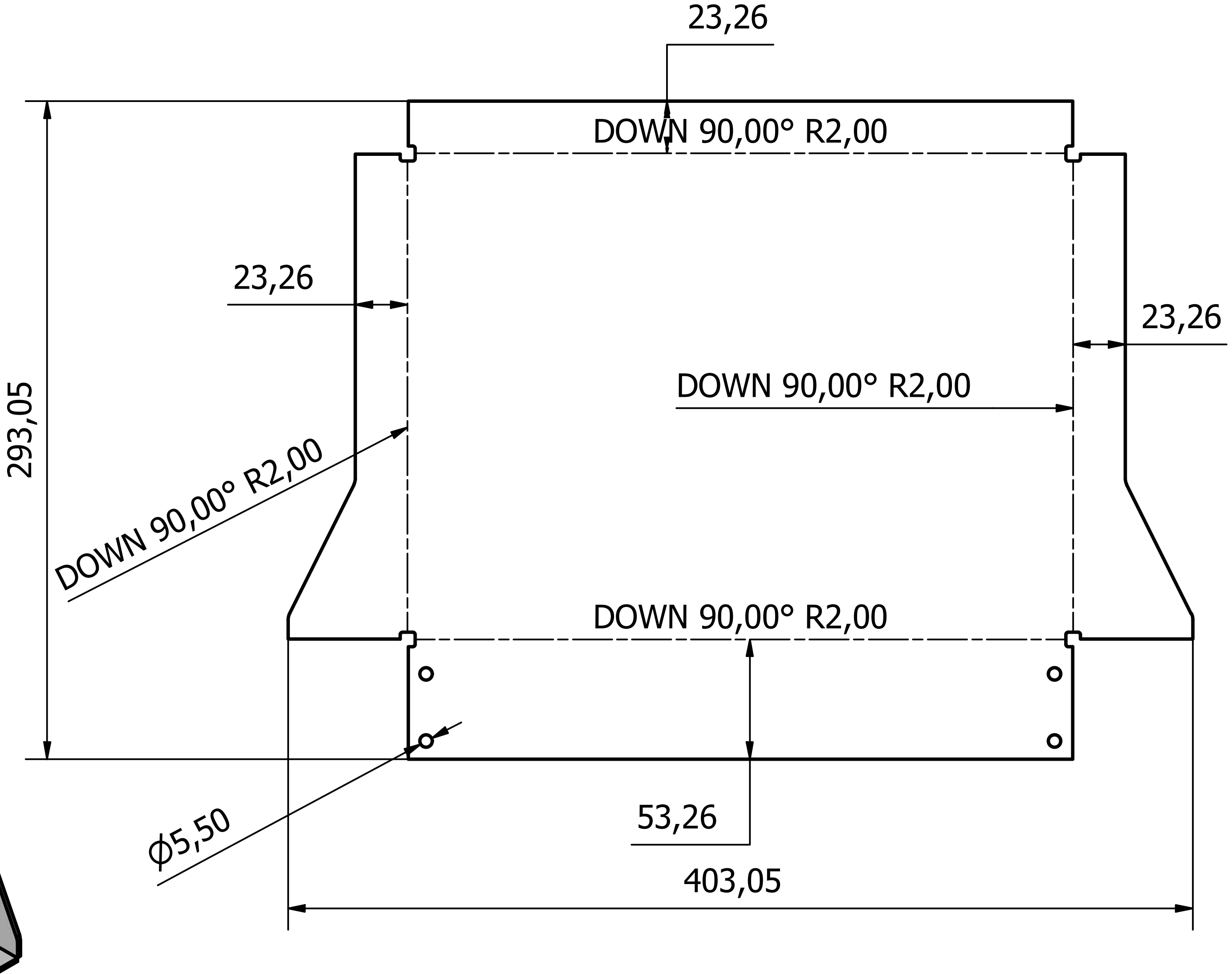
Cutout Lid

Swerve Drive AGV

Edition	Sheet
	122 / 207

Sheet Thickness: 2,00mm





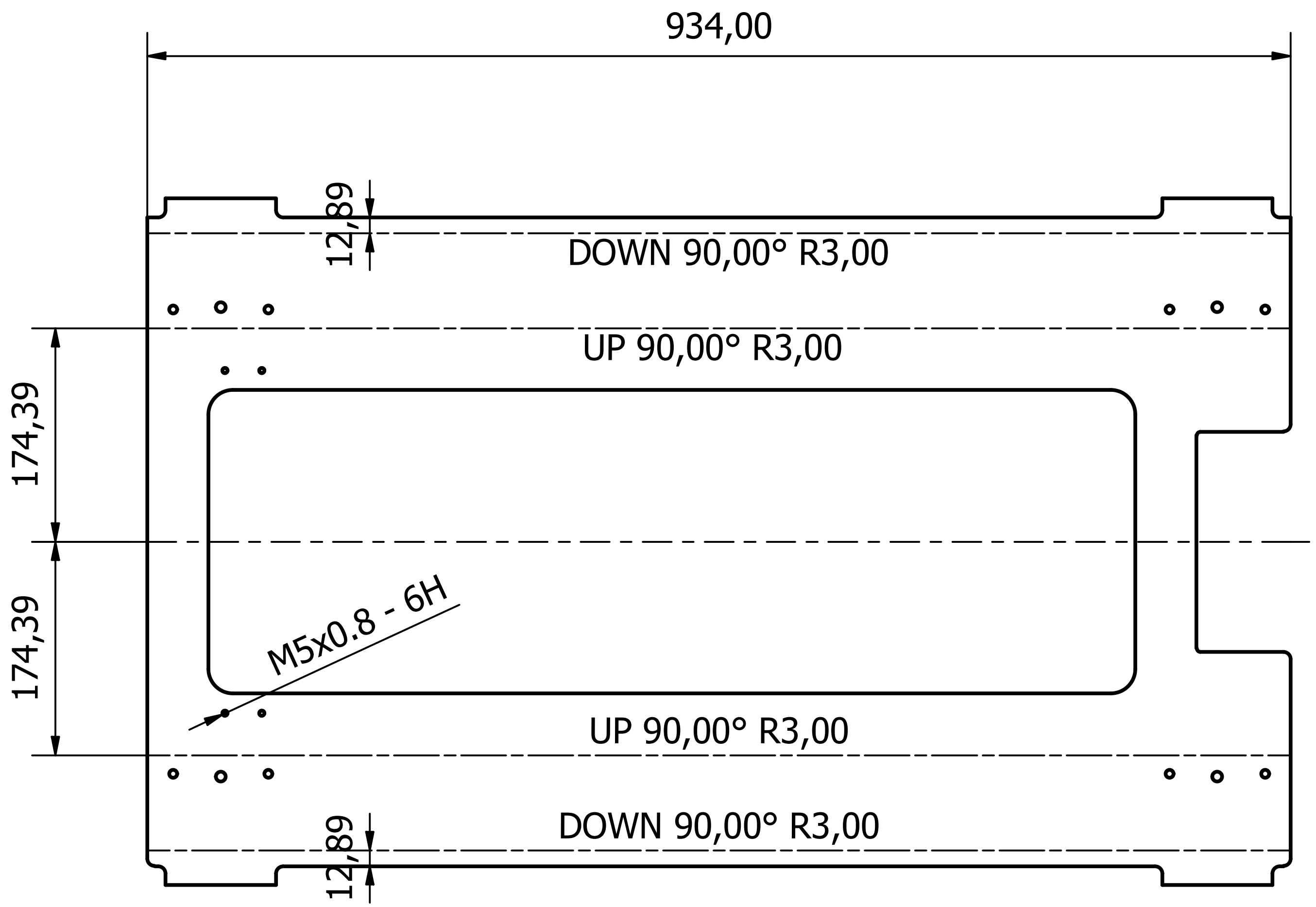
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	LPB	Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Battery Bank Control Mount Plate

Swerve Drive AGV

Edition	Sheet
	123 / 207



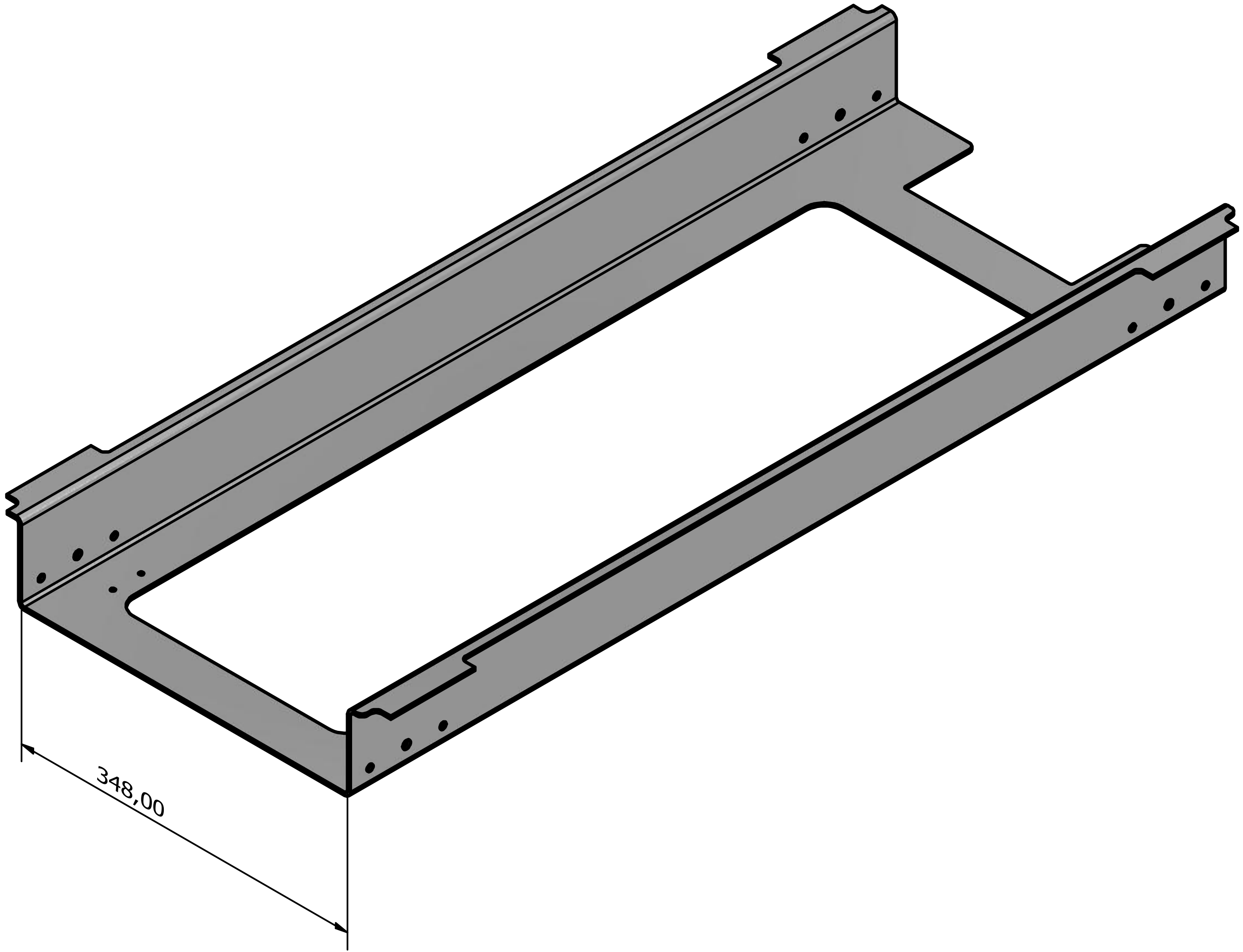
Sheet Thickness: 3,00mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	LPB	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Battery Box 3 Base Plate

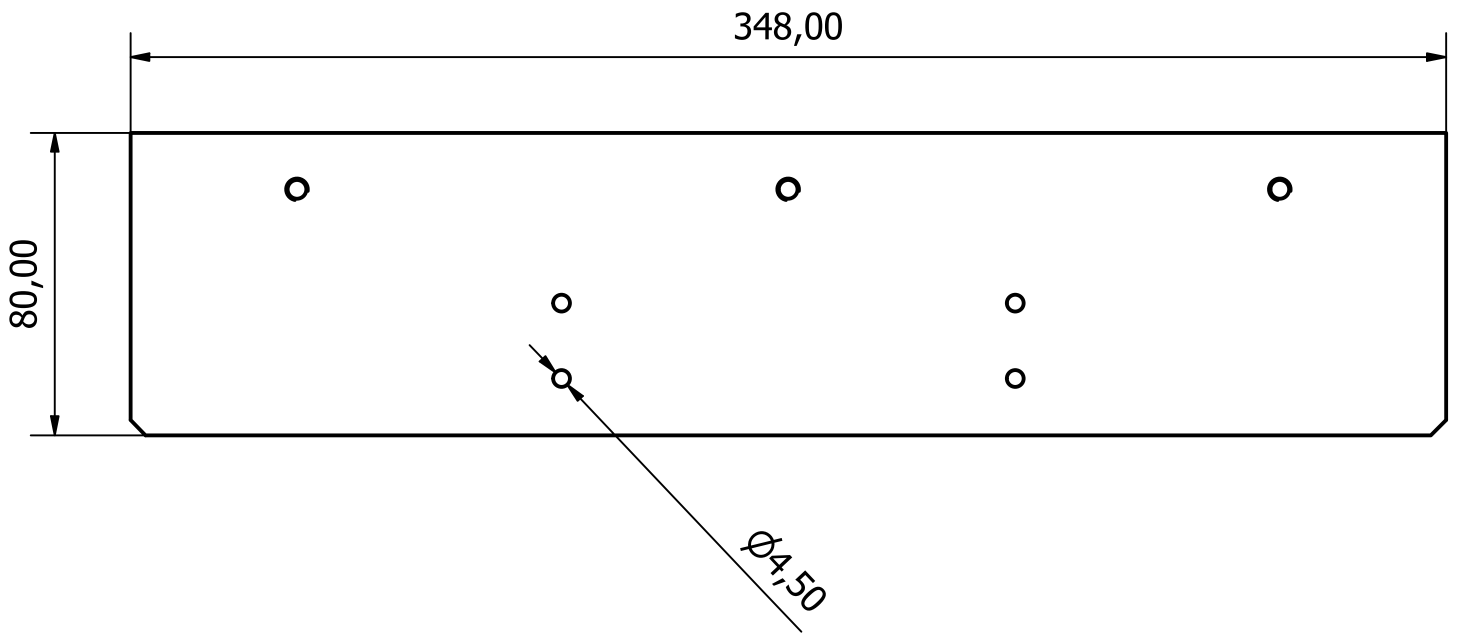
Swerve Drive AGV	Edition	Sheet
		124 / 207



Sheet Thickness: 3,00mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	LPB	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY	Battery Box 3 Base Plate		
	Swerve Drive AGV	Edition	Sheet 125 / 207



Sheet Thickness: 3,00mm

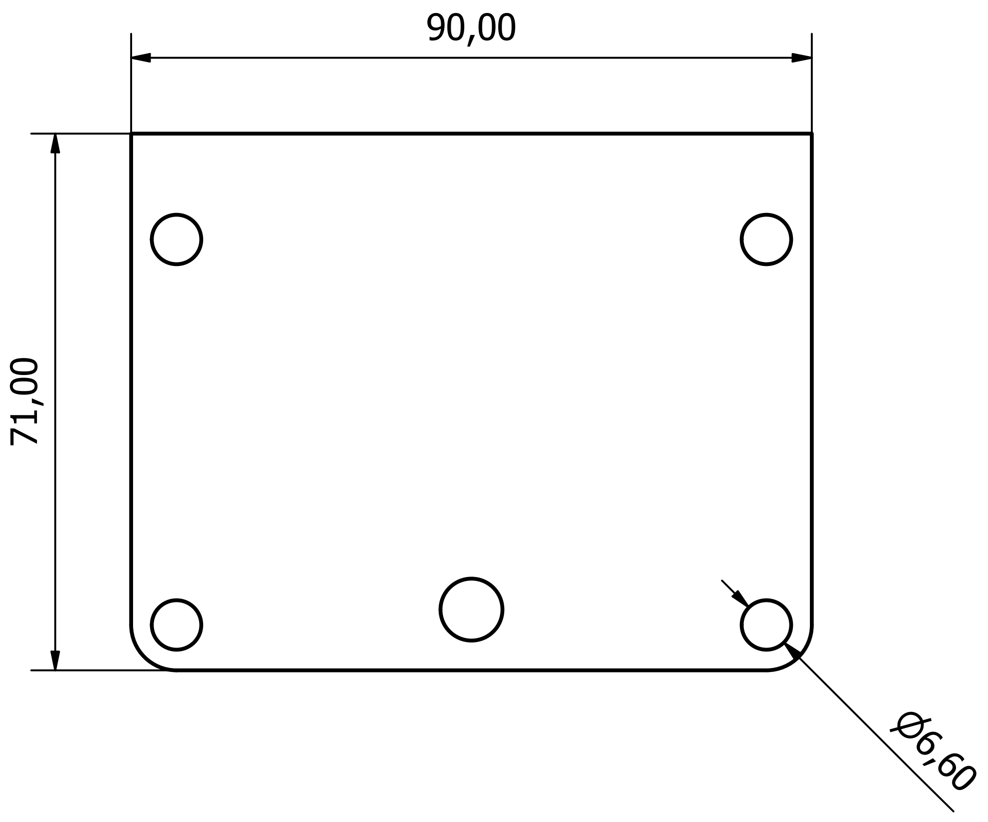
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	LPB	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Battery Box Base End Plate A

Swerve Drive AGV

Edition	Sheet
	126 / 207



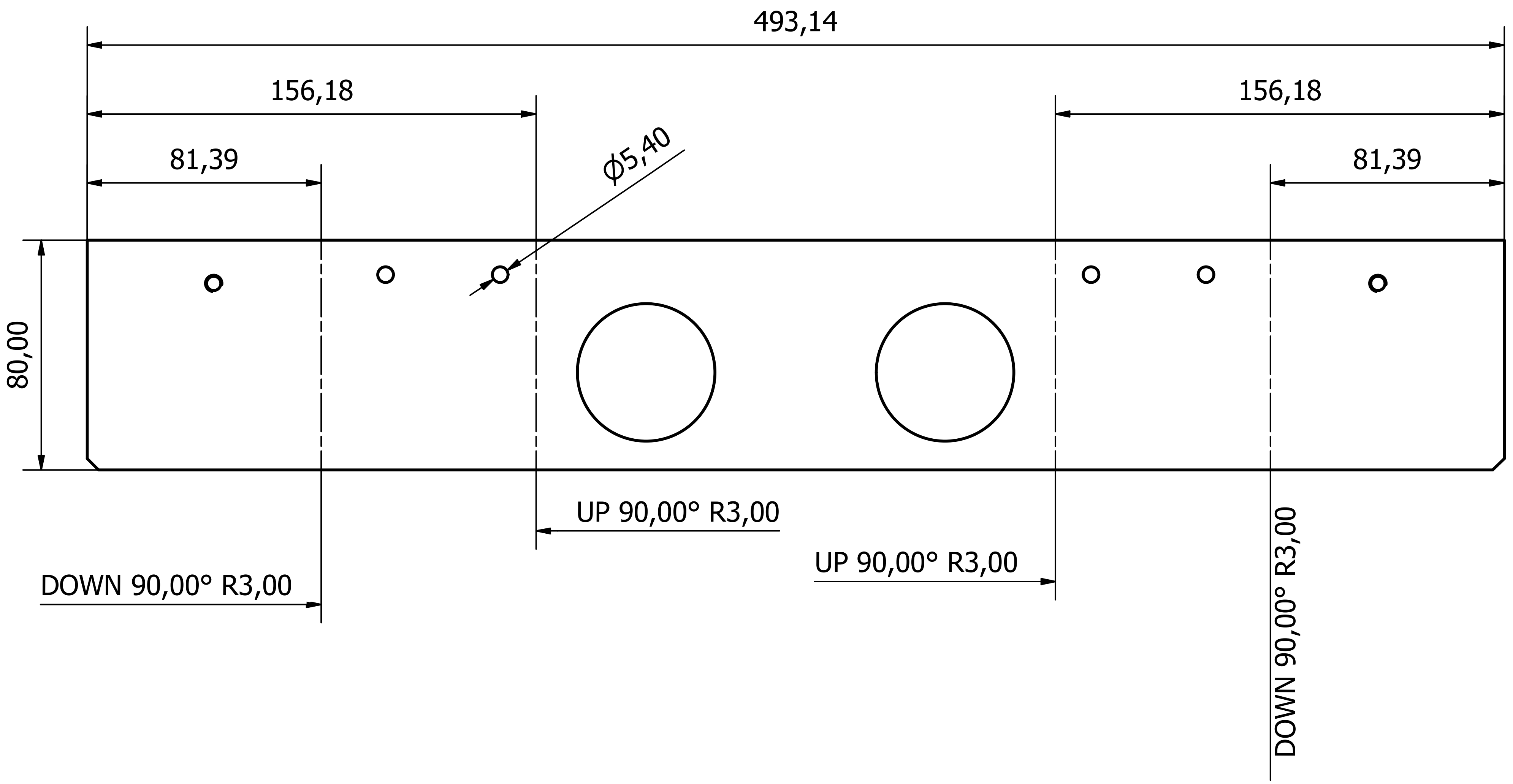
Sheet Thickness: 3,00mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	4	LPB	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Battery Box 3 Flanges

Swerve Drive AGV	Edition	Sheet
		127 / 207



Sheet Thickness: 3,00mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	LPB	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

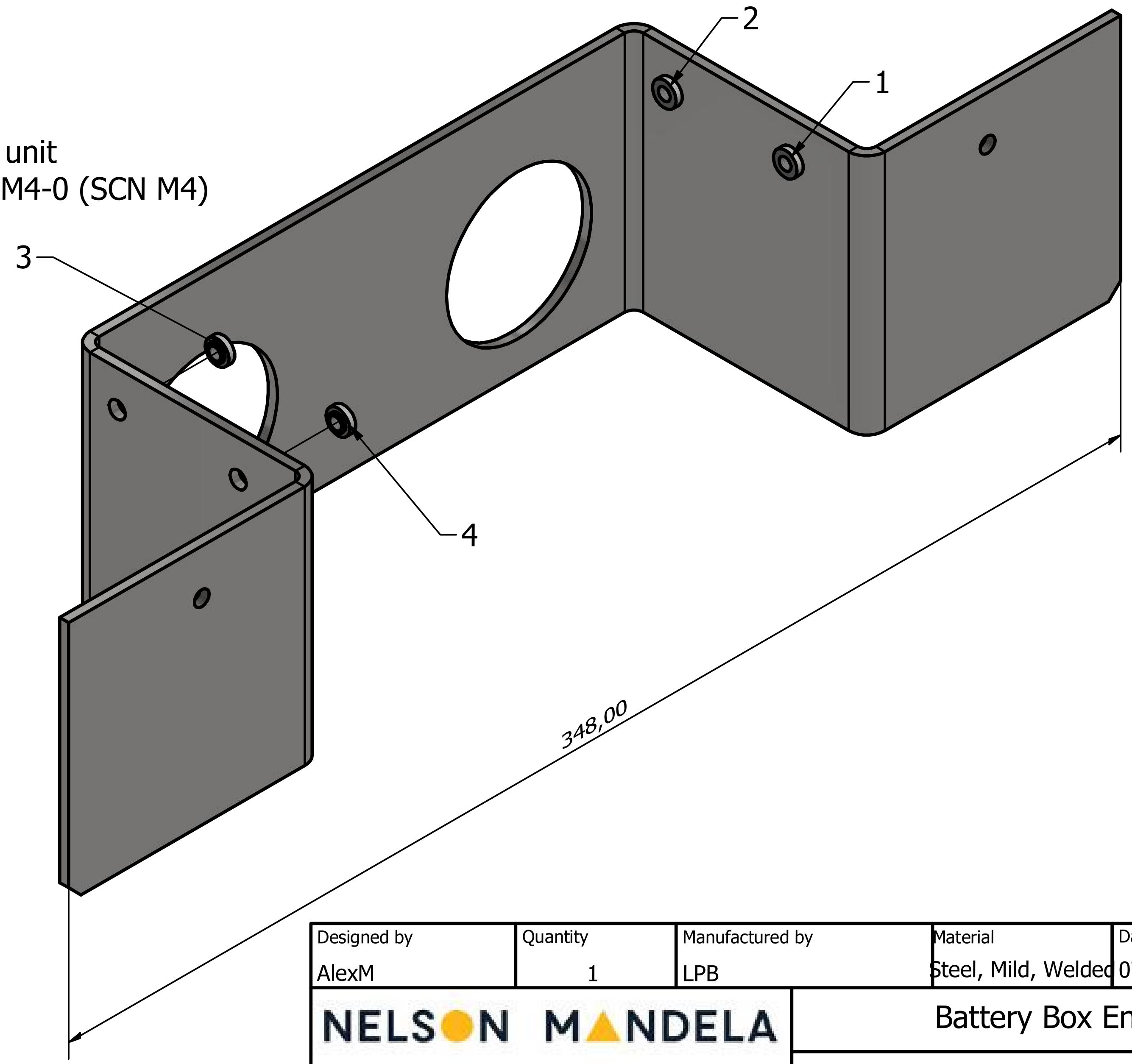
Battery Box Base End Plate B

Swerve Drive AGV	Edition	Sheet
		128 / 207

HPI Inserts

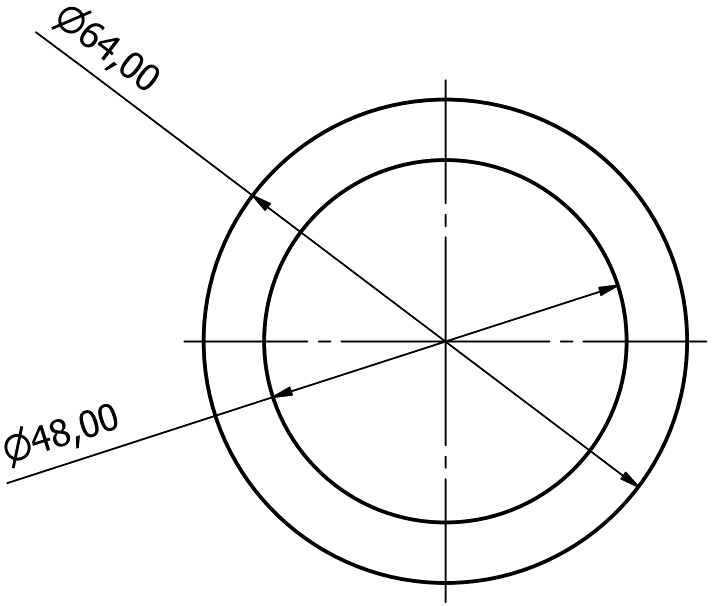
Number : 4 unit

Designation : CM4-0 (SCN M4)



Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	LPB	Steel, Mild, Welded	07/03/2019

NELSON MANDELA UNIVERSITY	Battery Box End Plate B		
	Swerve Drive AGV	Edition	Sheet 129 / 207



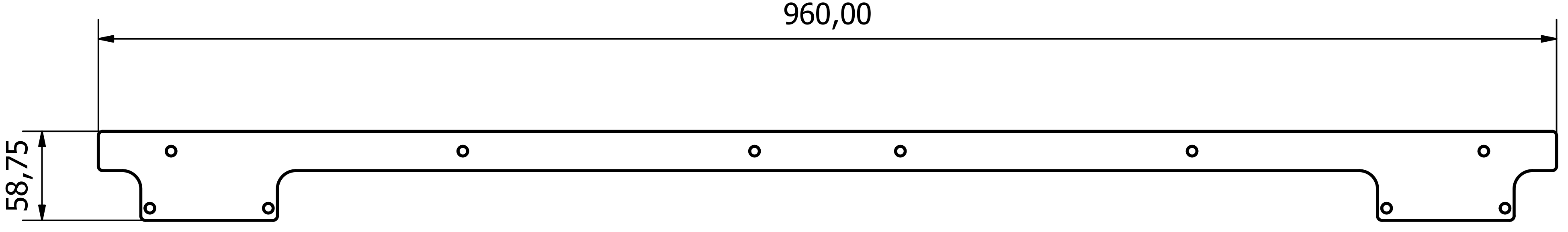
Sheet Thickness: 5,00mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	LPB	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Electrode Spacer

	Edition	Sheet
Swerve Drive AGV		130 / 207



Sheet Thickness: 3,00mm

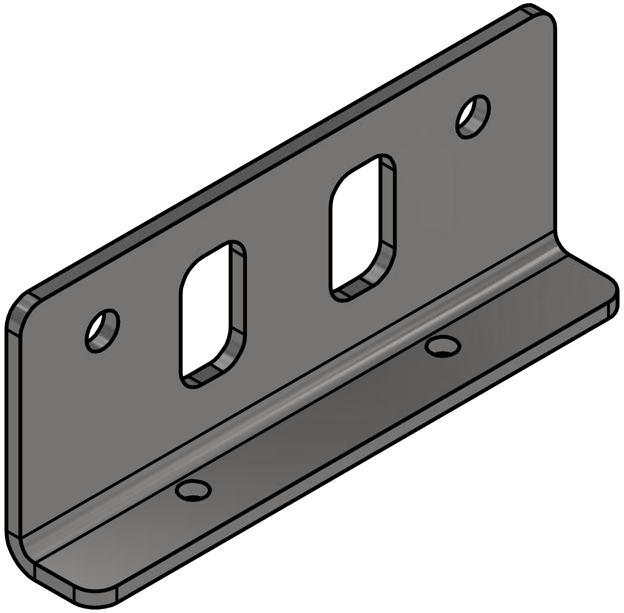
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	LPB	Steel, Mild	07/03/2019

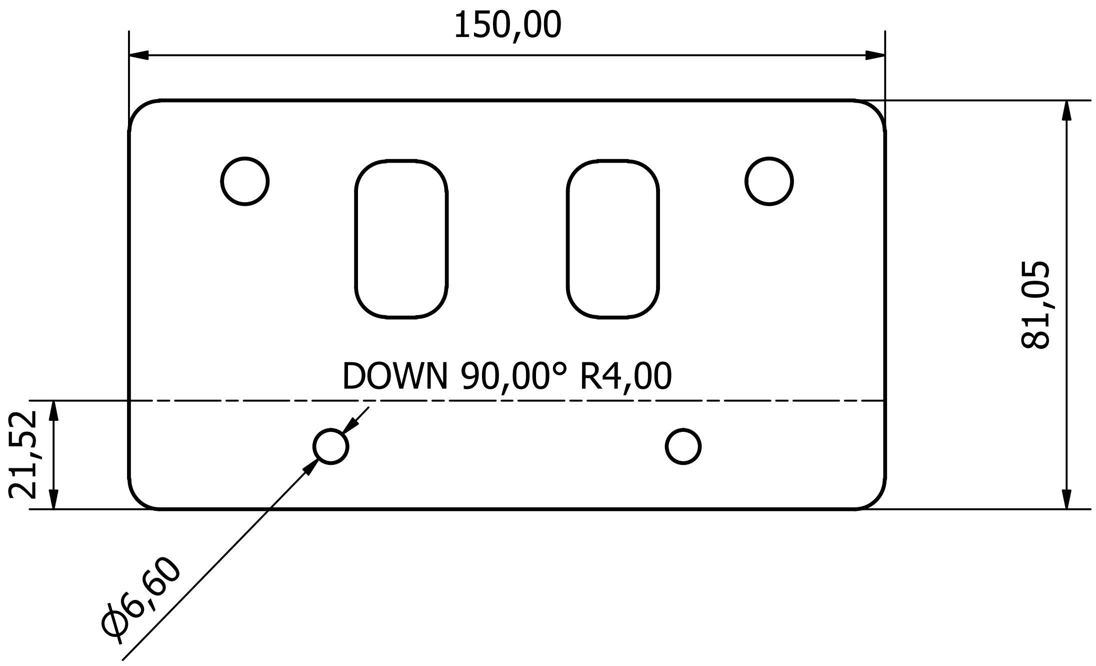
NELSON MANDELA UNIVERSITY

Battery Box Rack Attachment Plate

Swerve Drive AGV

Edition	Sheet
	131 / 207





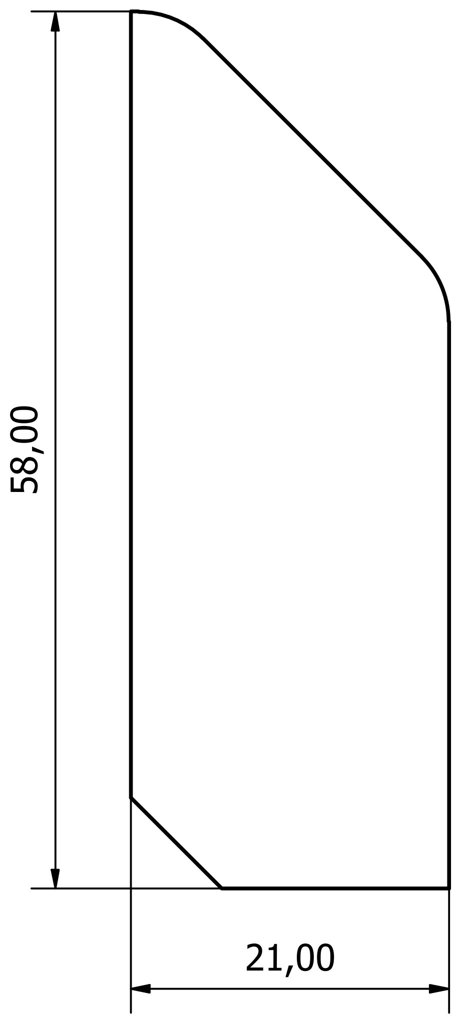
Sheet Thickness: 4,00mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	LPB	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Contactor Plate

	Edition	Sheet
Swerve Drive AGV		132 / 207



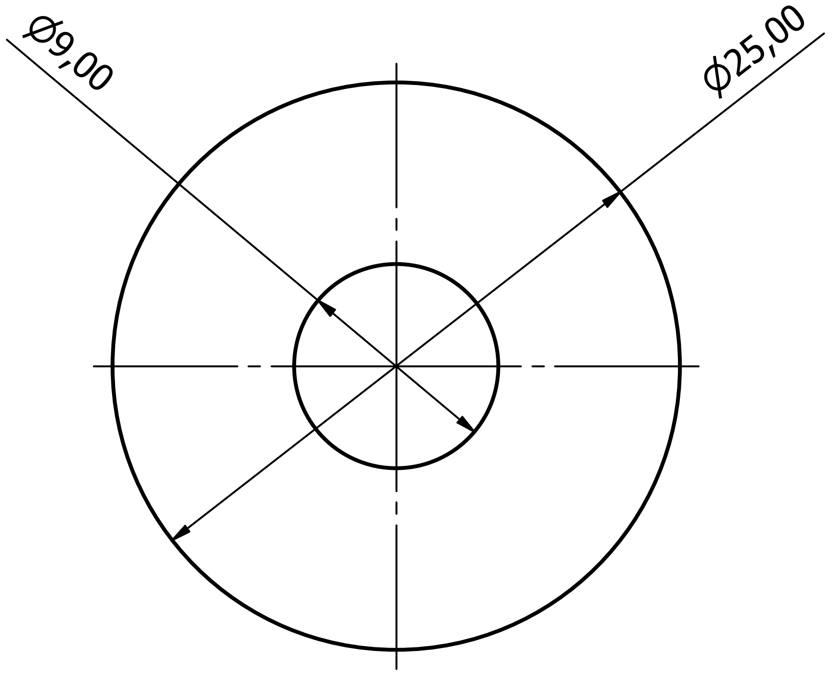
Sheet Thickness: 3,00mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	LPB	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Contactor Plate Web

	Edition	Sheet
Swerve Drive AGV		133 / 207



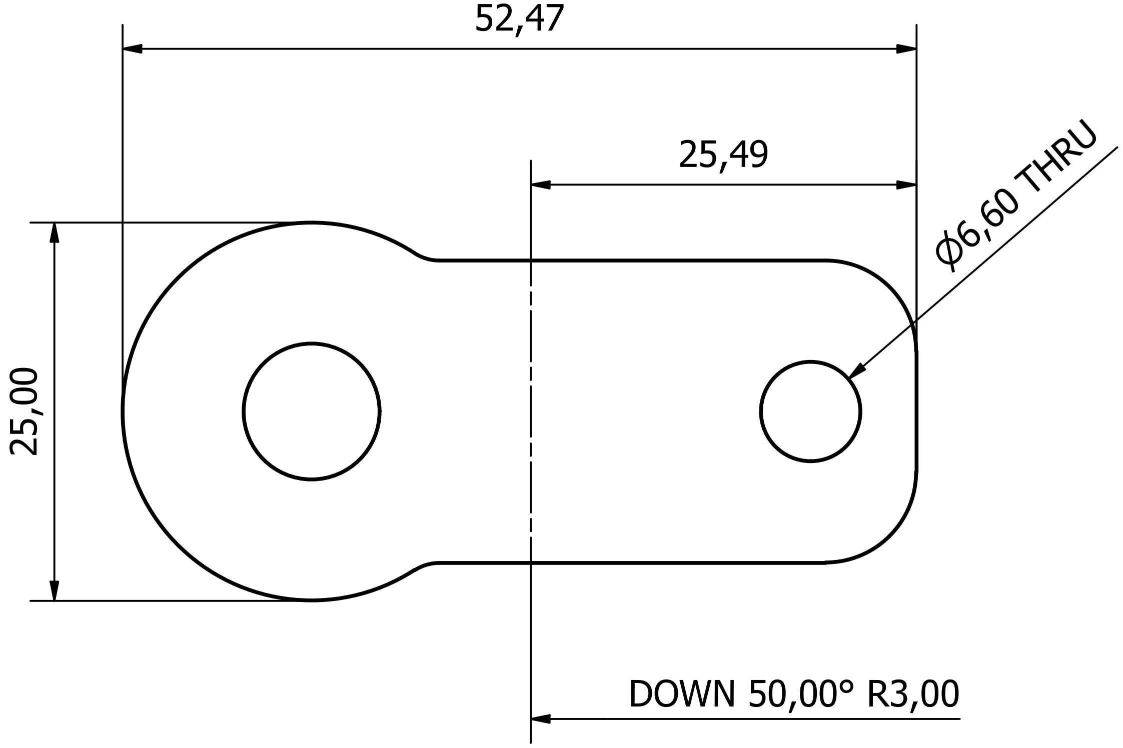
Sheet Thickness: 3,00mm

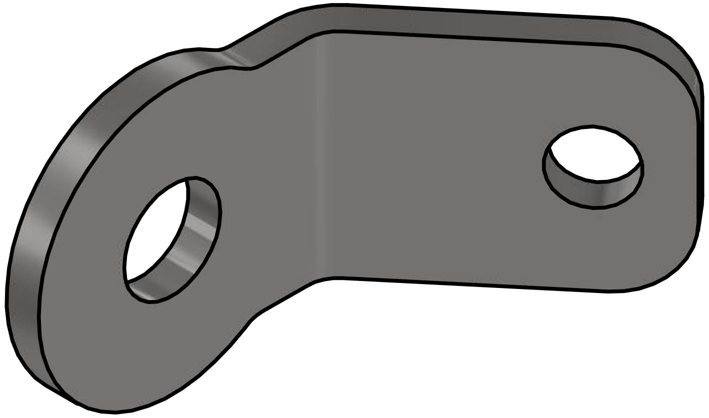
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	LPB	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Spring Ends A

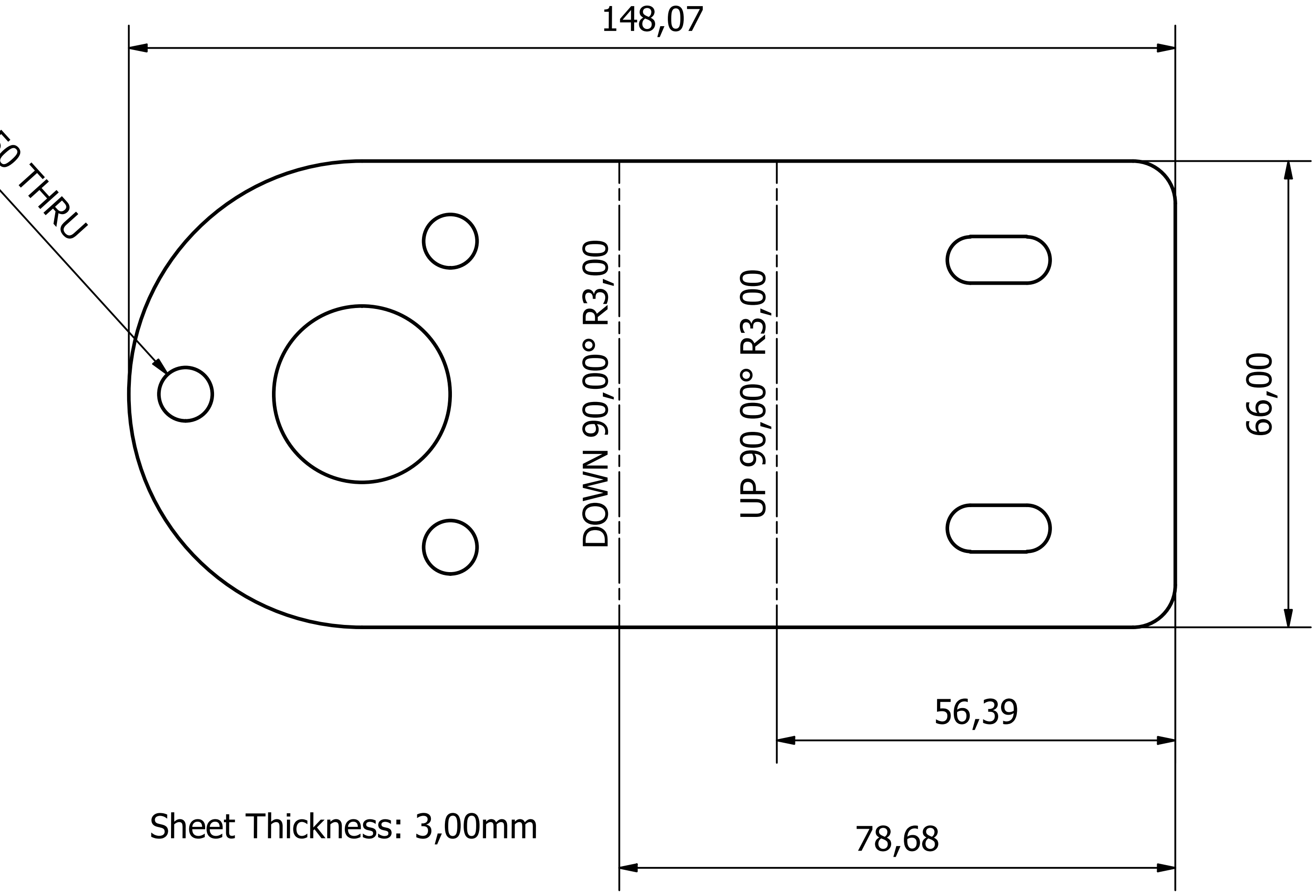
	Edition	Sheet
Swerve Drive AGV		134 / 207



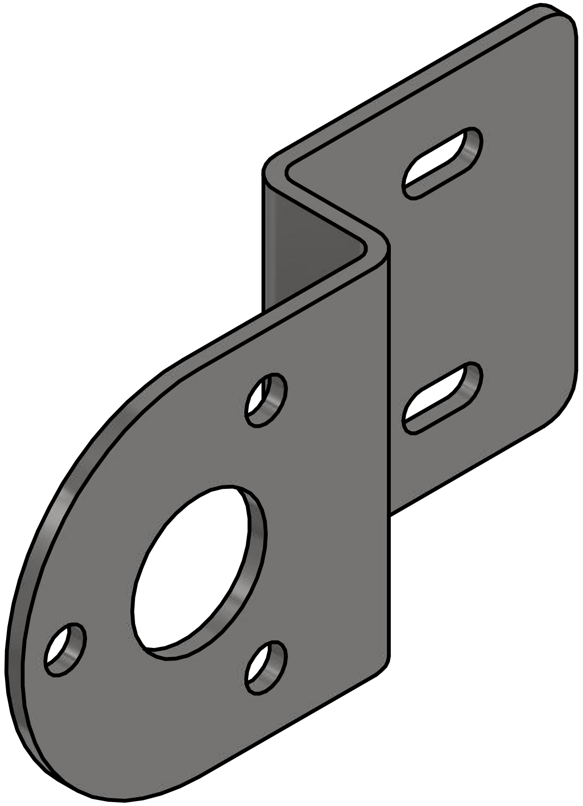


Sheet Thickness: 3,00mm

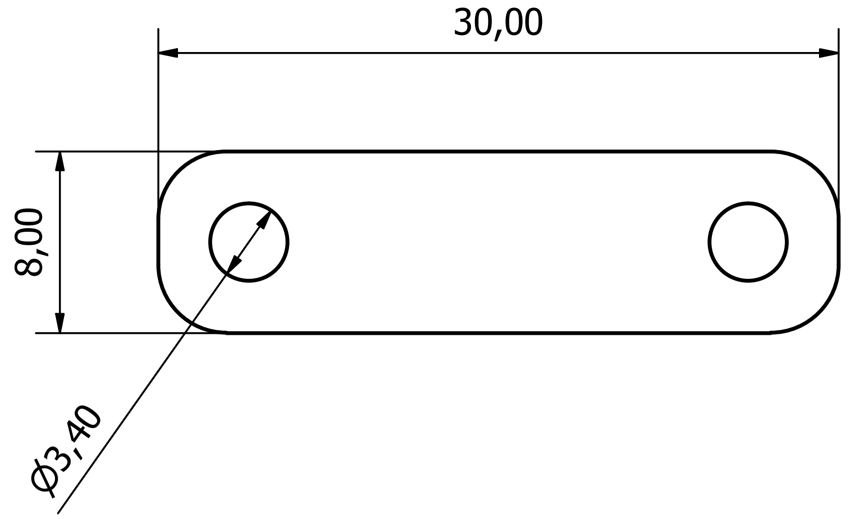
Designed by AlexM	Quantity 2	Manufactured by LPB	Material Steel, Mild	Date 07/03/2019
NELSON MANDELA UNIVERSITY	Spring Ends B			
	Swerve Drive AGV		Edition	Sheet 135 / 207



Sheet Thickness: 3,00mm



Designed by AlexM	Quantity 1	Manufactured by LBP	Material Steel, Mild	Date 07/03/2019	
NELSON MANDELA UNIVERSITY	Rack Motor Mount				
	Swerve Drive AGV			Edition	Sheet 136 / 207



Sheet Thickness: 1,60mm

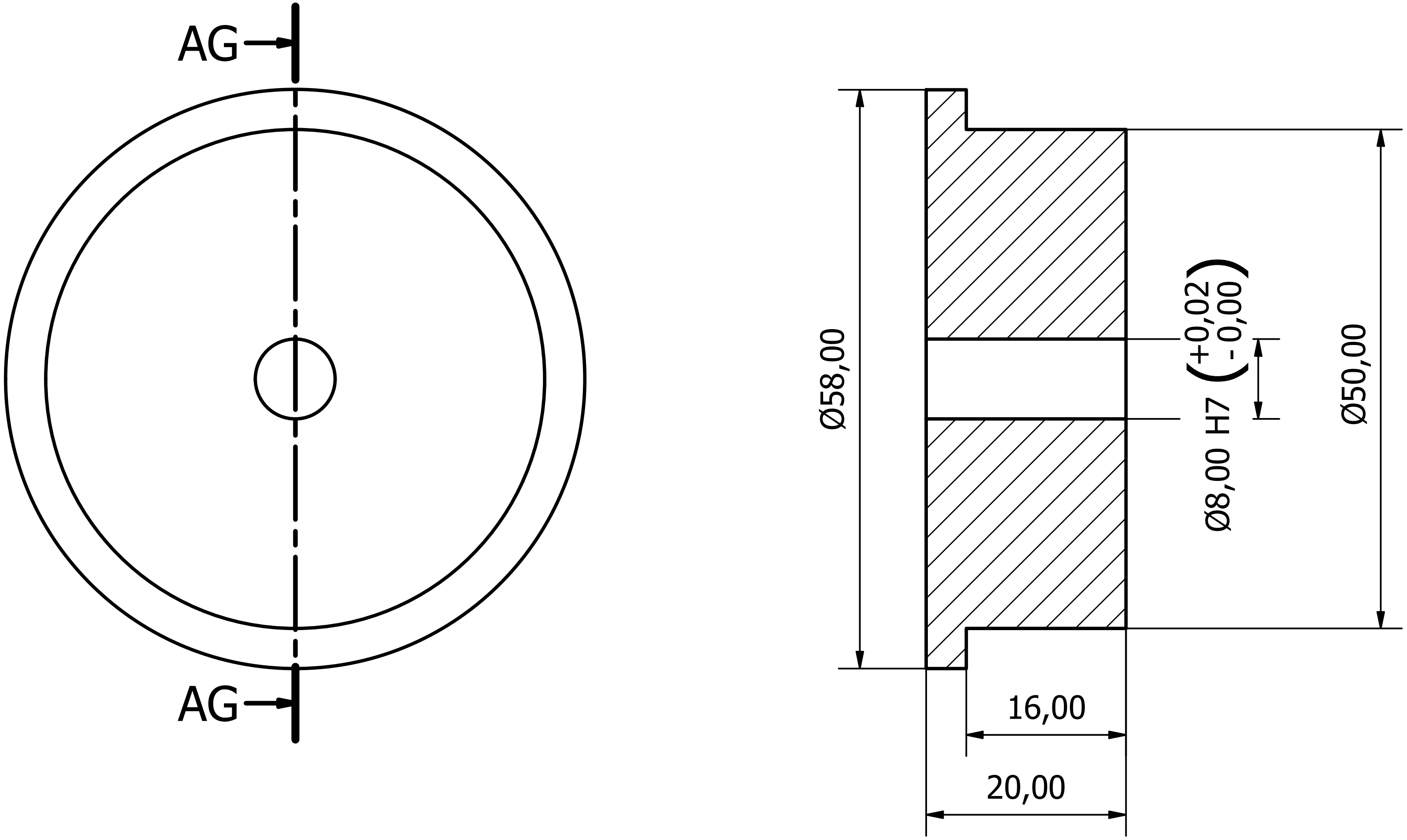
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	LPB	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Pinion Pin Strap

	Edition	Sheet
Swerve Drive AGV		137 / 207

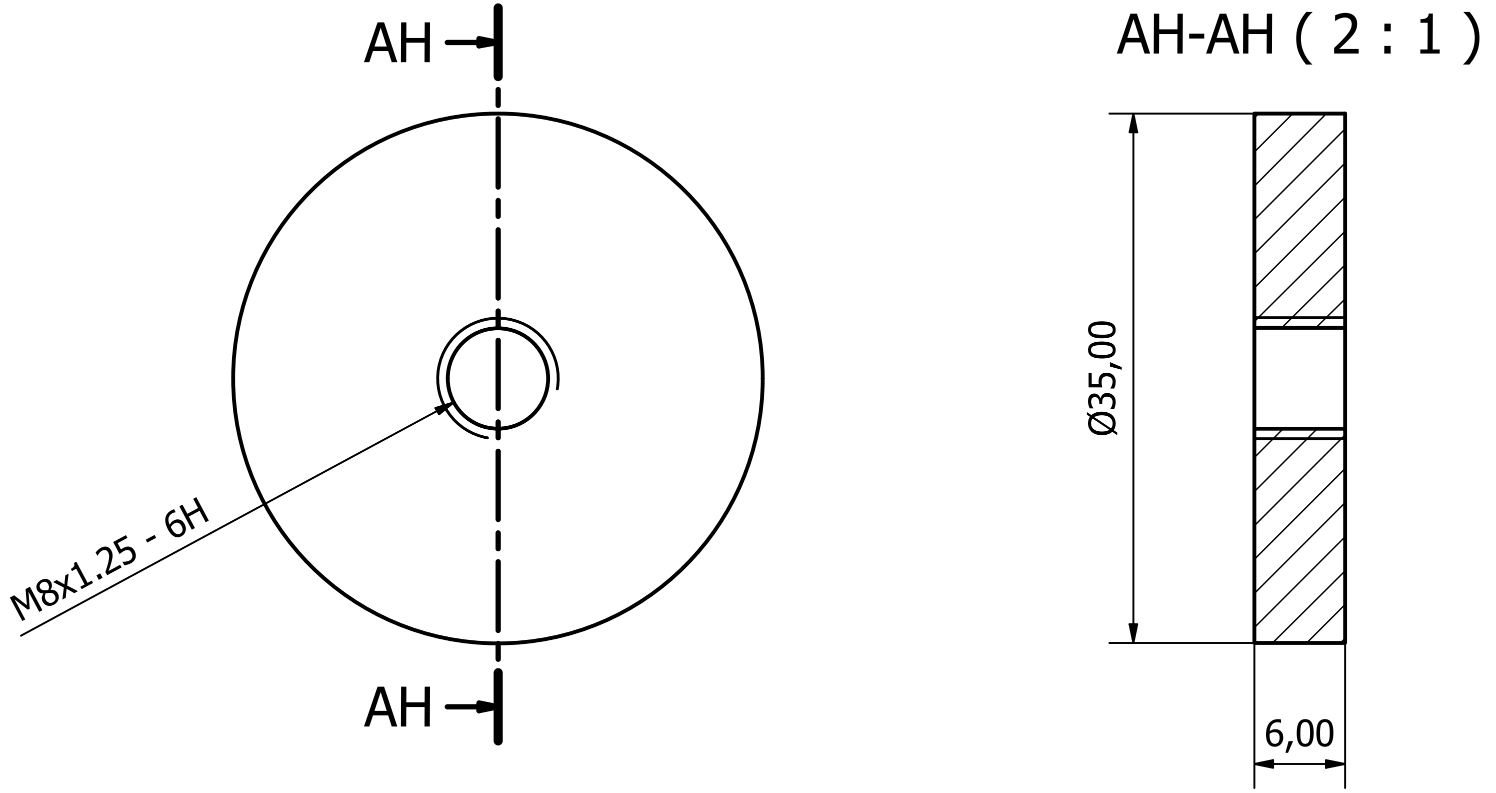
AG-AG (1.5 : 1)



Designed by	Quantity	Manufactured by	Material	Date
AlexM	4	Retrac	Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Flanged Wheel		
Swerve Drive AGV	Edition	Sheet 138 / 207

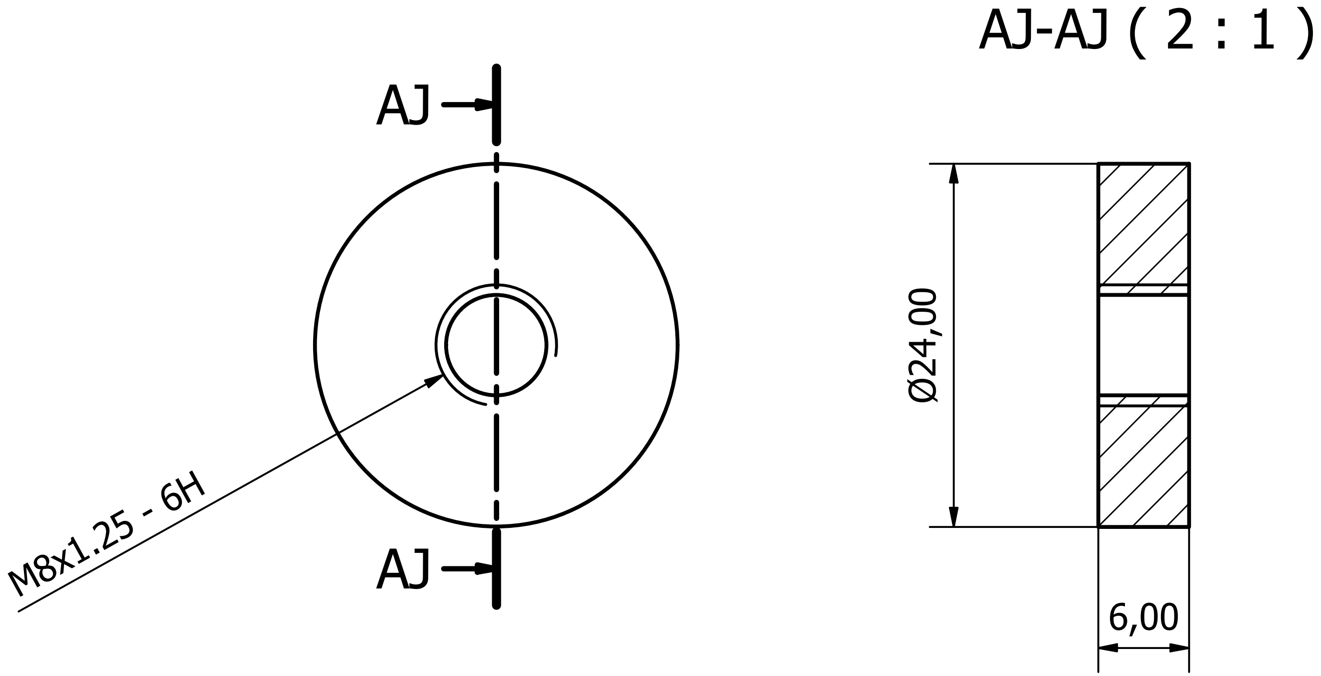


Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	Retrac	Copper	07/03/2019

NELSON MANDELA UNIVERSITY

Copper Plate B

	Edition	Sheet
Swerve Drive AGV		139 / 207

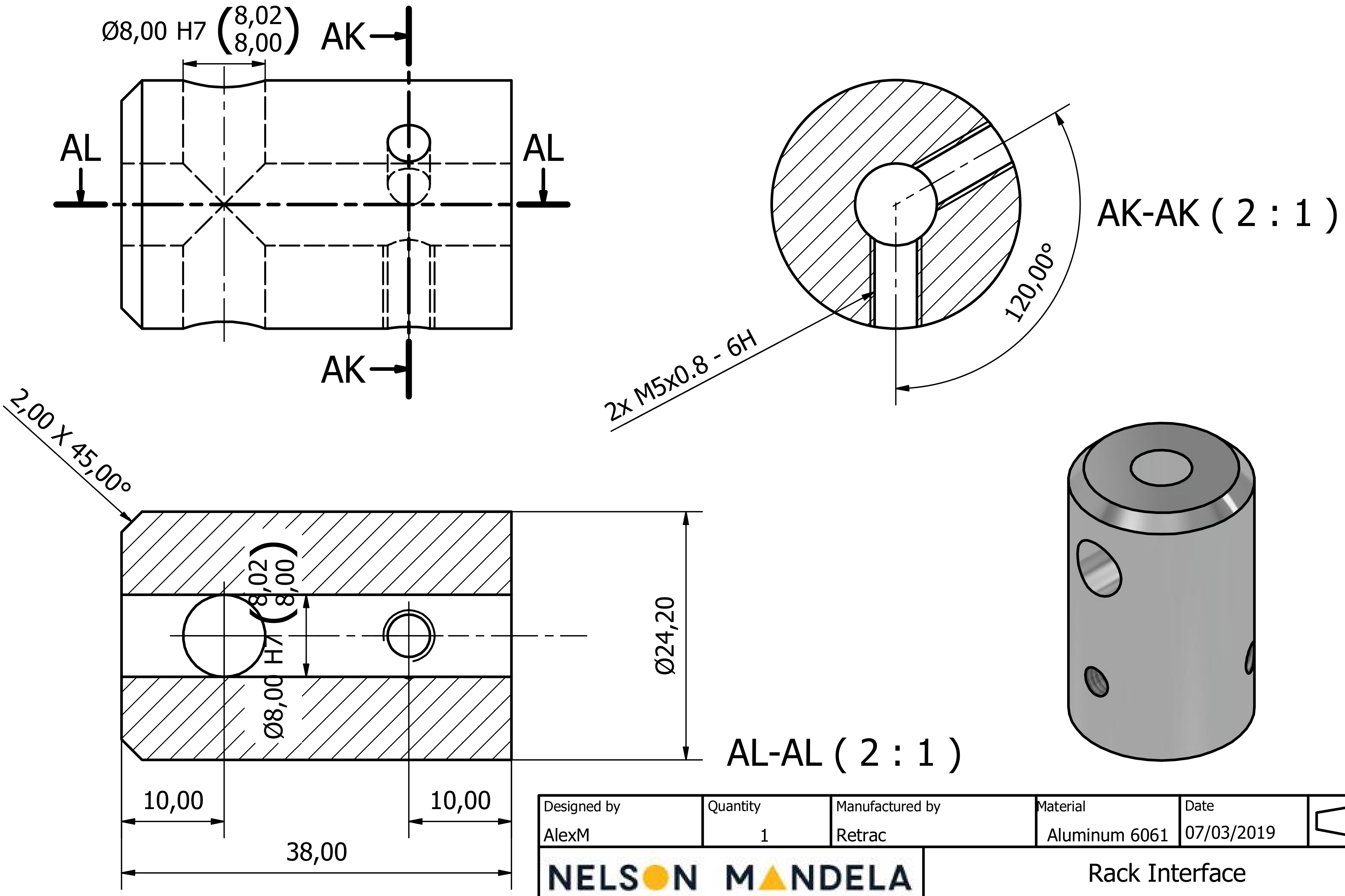


Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	Retrac	Copper, Alloy	07/03/2019

NELSON MANDELA UNIVERSITY

Copper Plate A

Swerve Drive AGV	Edition	Sheet
		140 / 207



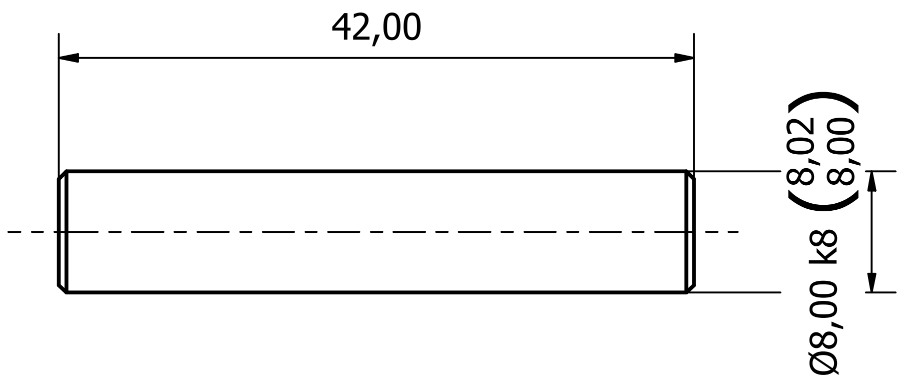
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	Retrac	Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Rack Interface

Swerve Drive AGV

Edition	Sheet
	141 / 207

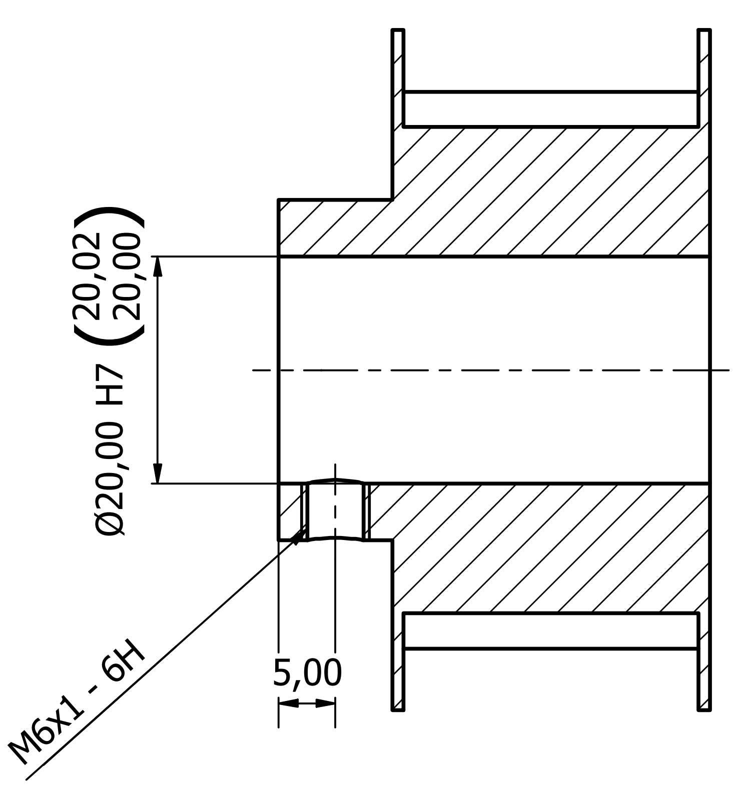


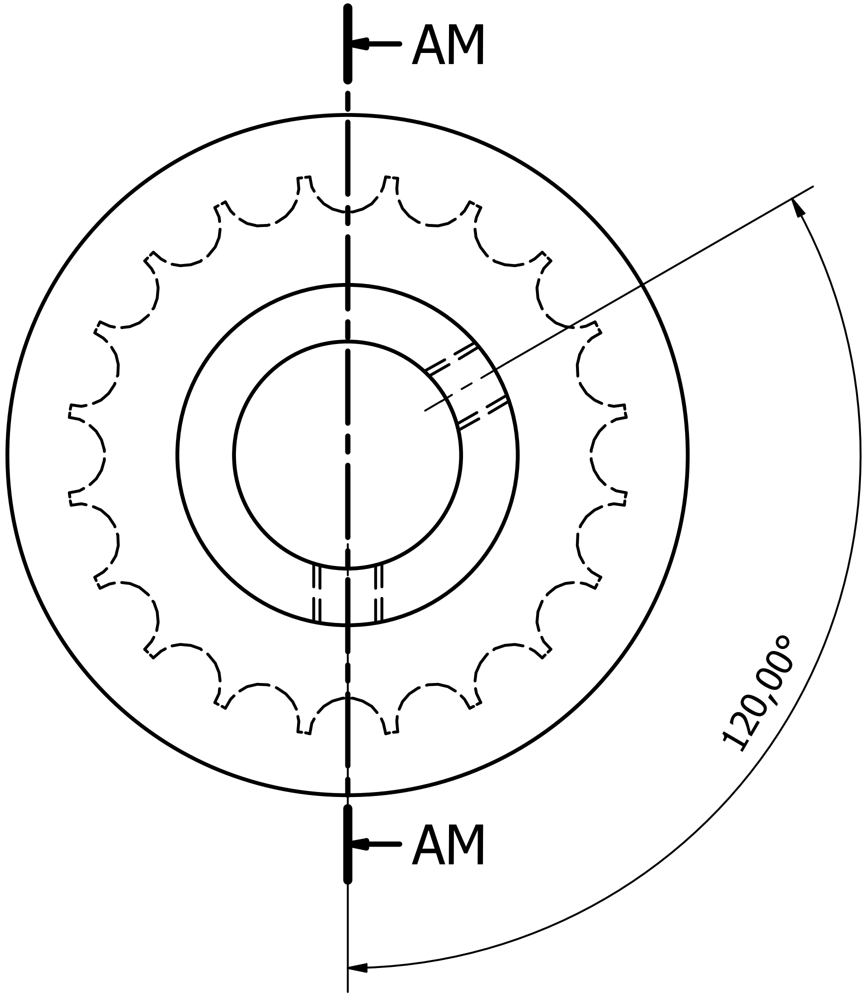
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1		Stainless Steel	07/03/2019

NELSON MANDELA UNIVERSITY

Swerve Drive AGV	Edition	Sheet
		142 / 207

AM-AM (1.5:1)



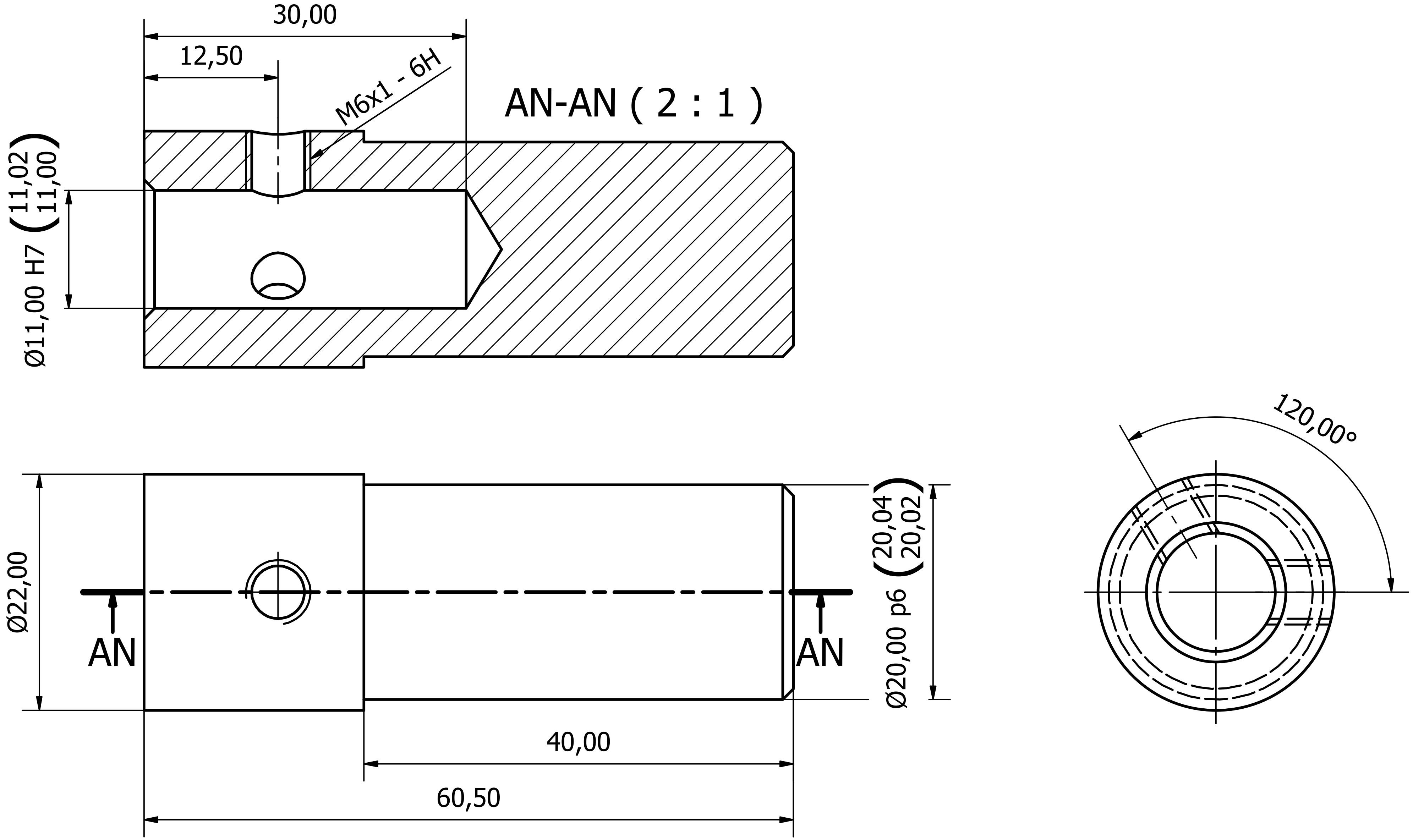


Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	Internal	Steel, Galvanized	07/03/2019

NELSON MANDELA UNIVERSITY

PHP 45-8M-20-RSB

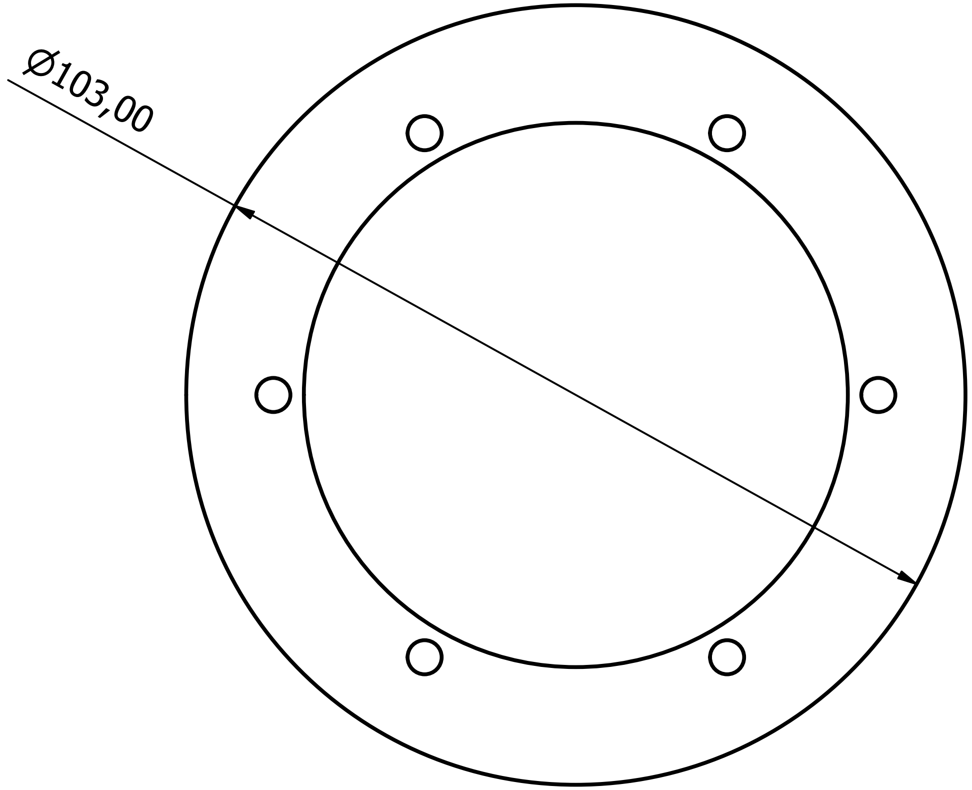
	Edition	Sheet
Swerve Drive AGV		143 / 207



Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	Dale Flynn	Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Steering Pinion Converter		
Swerve Drive AGV	Edition	Sheet 144 / 207



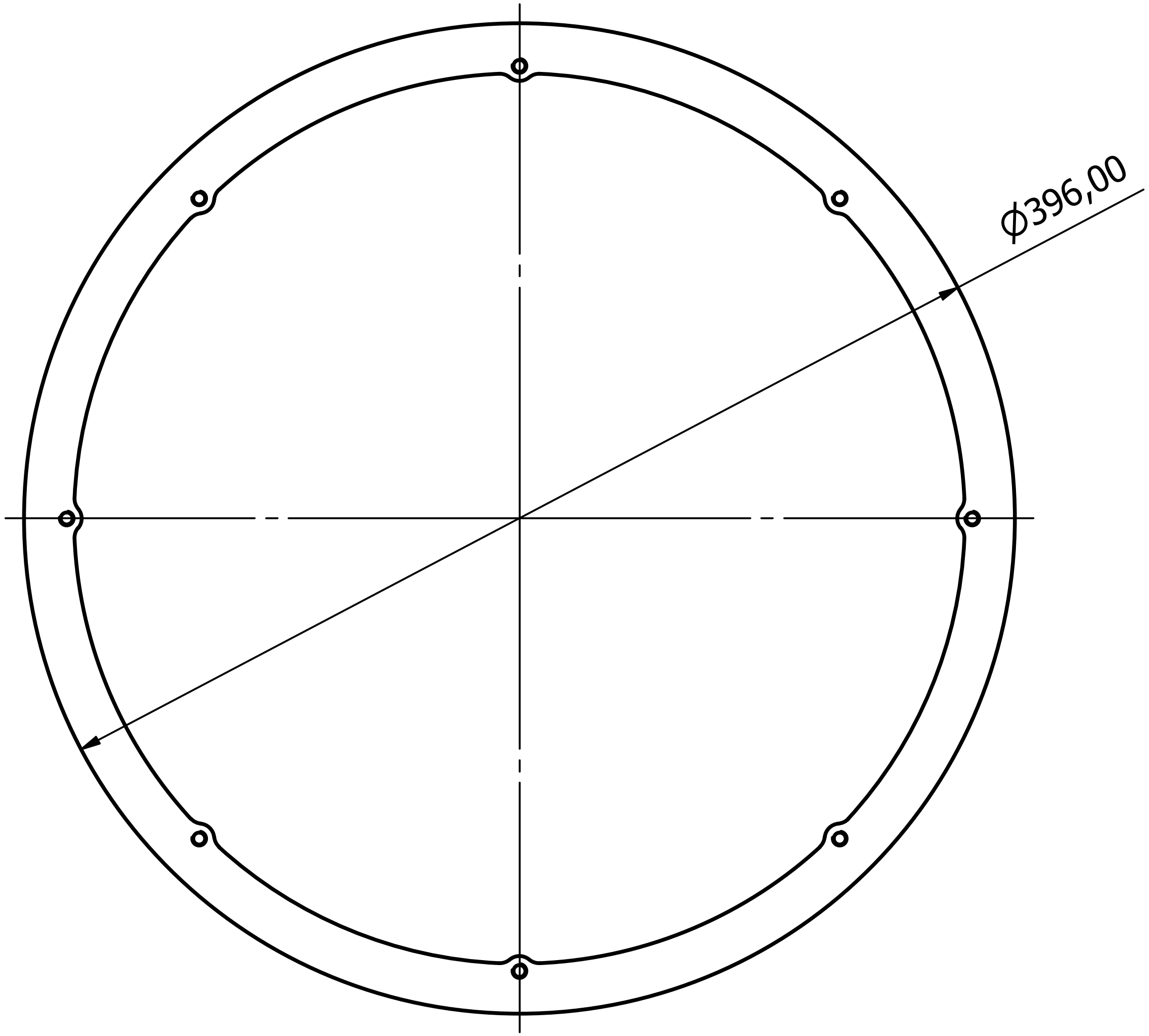
Sheet Thickness: 1,20 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	4	LPB	Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Drive Pulley Flange

	Edition	Sheet
Swerve Drive AGV		145 / 207



Sheet Thickness: 3,00 mm

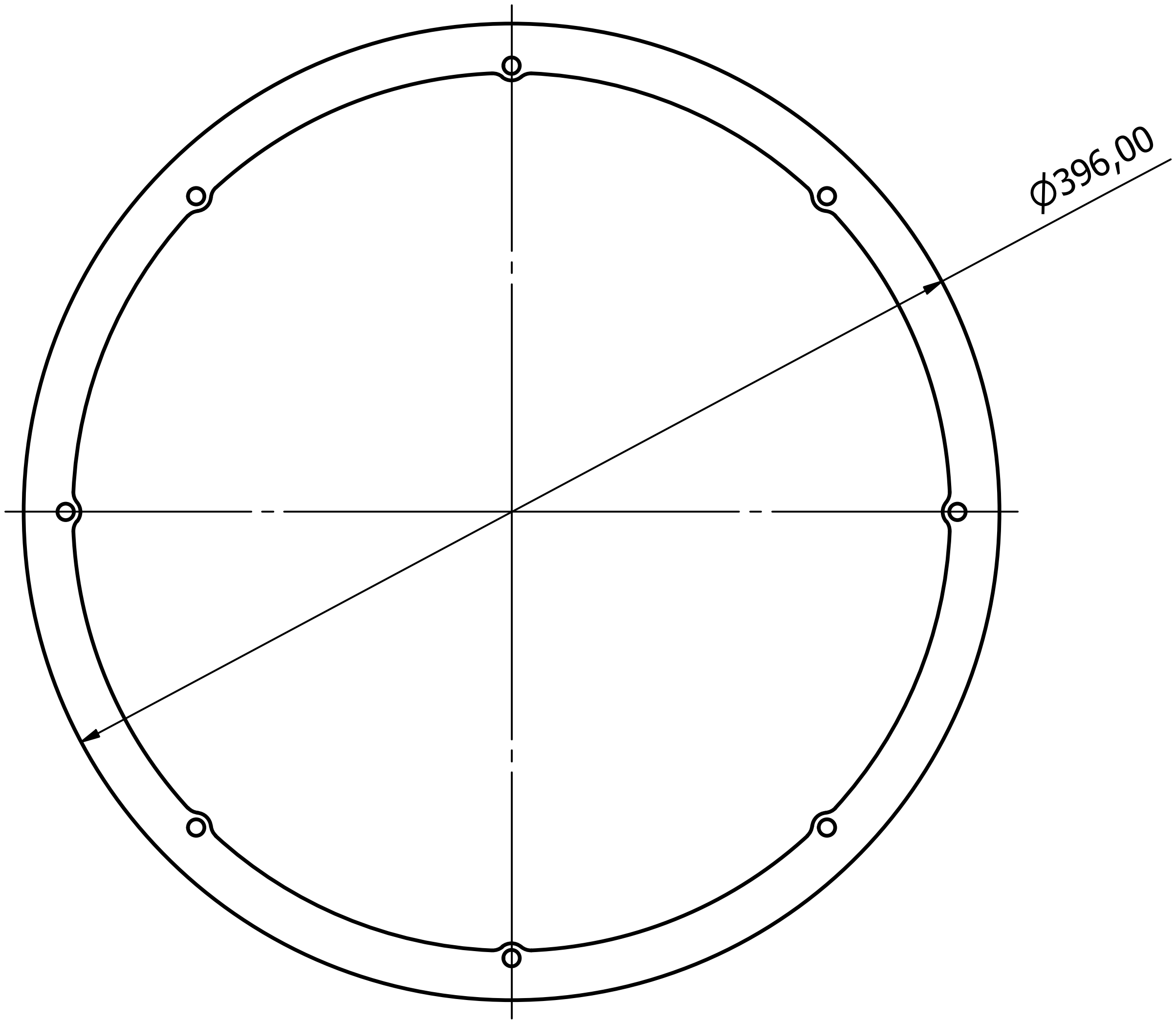
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	LPB	Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Large Gear Flange - Threaded

Swerve Drive AGV

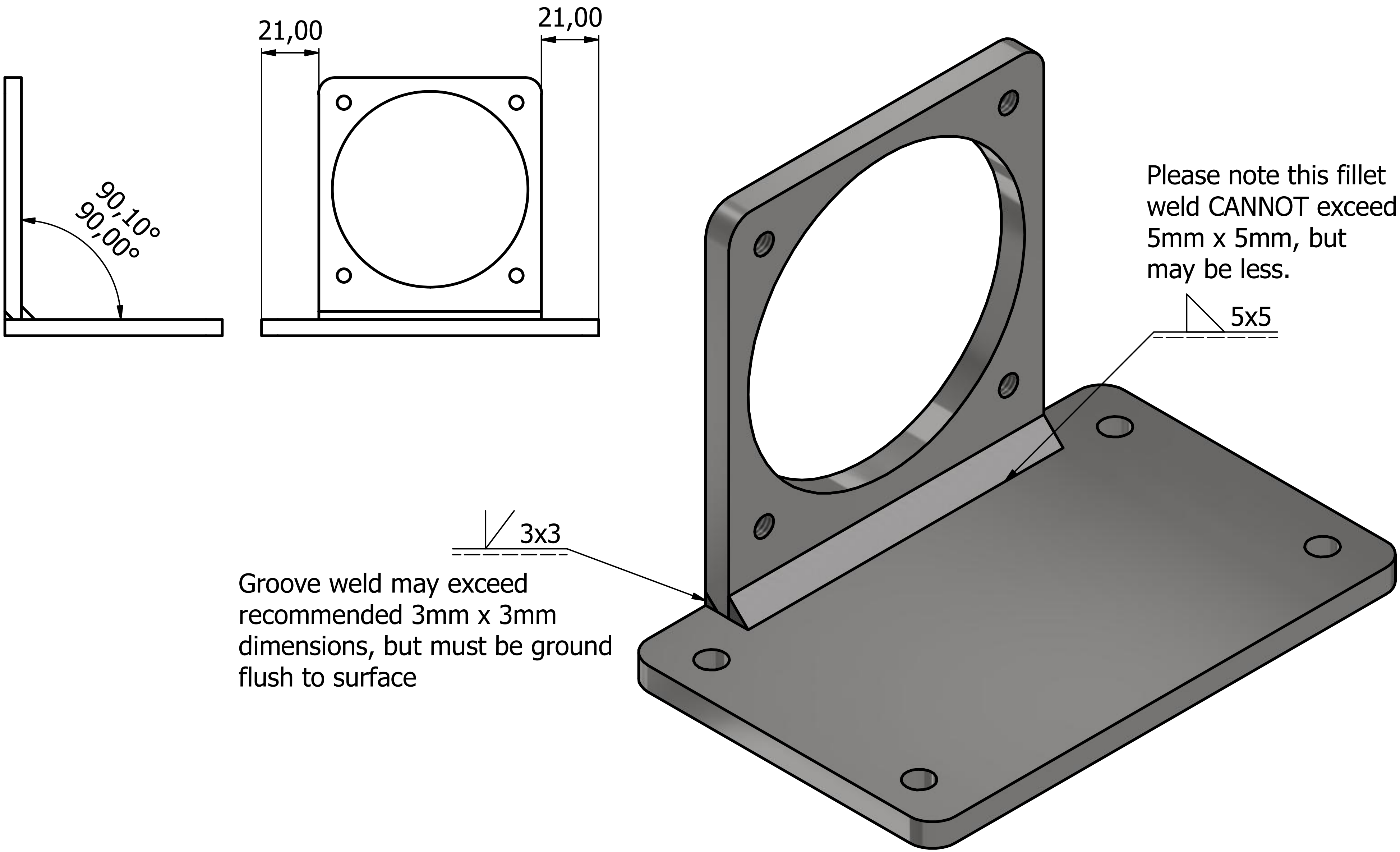
Edition	Sheet
	146 / 207



Sheet Thickness: 1,60 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	LPB	Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY	Large Gear Flange - Clearence		
	Swerve Drive AGV	Edition	Sheet 147 / 207



Please note this fillet weld CANNOT exceed 5mm x 5mm, but may be less.

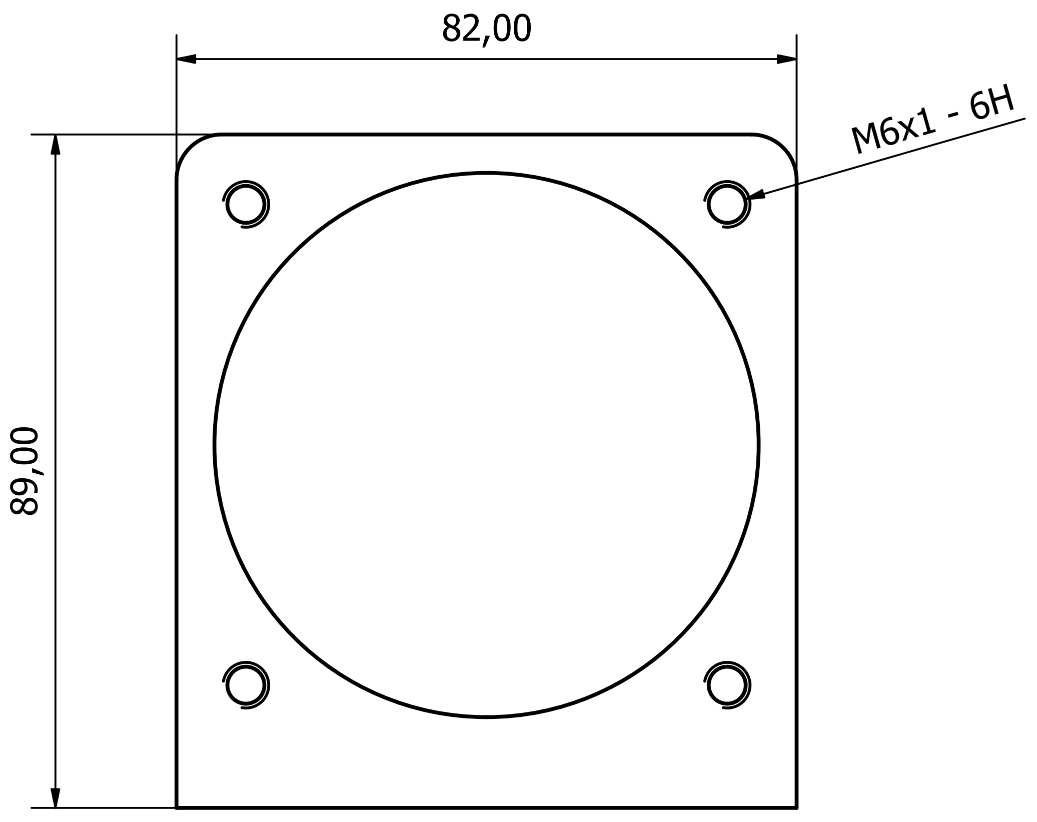
Groove weld may exceed recommended 3mm x 3mm dimensions, but must be ground flush to surface

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	LPB	Steel, Mild, Welded	07/03/2019

NELSON MANDELA UNIVERSITY

Traction Motor Mount Weldmatt

Swerve Drive AGV	Edition	Sheet
		148 / 207



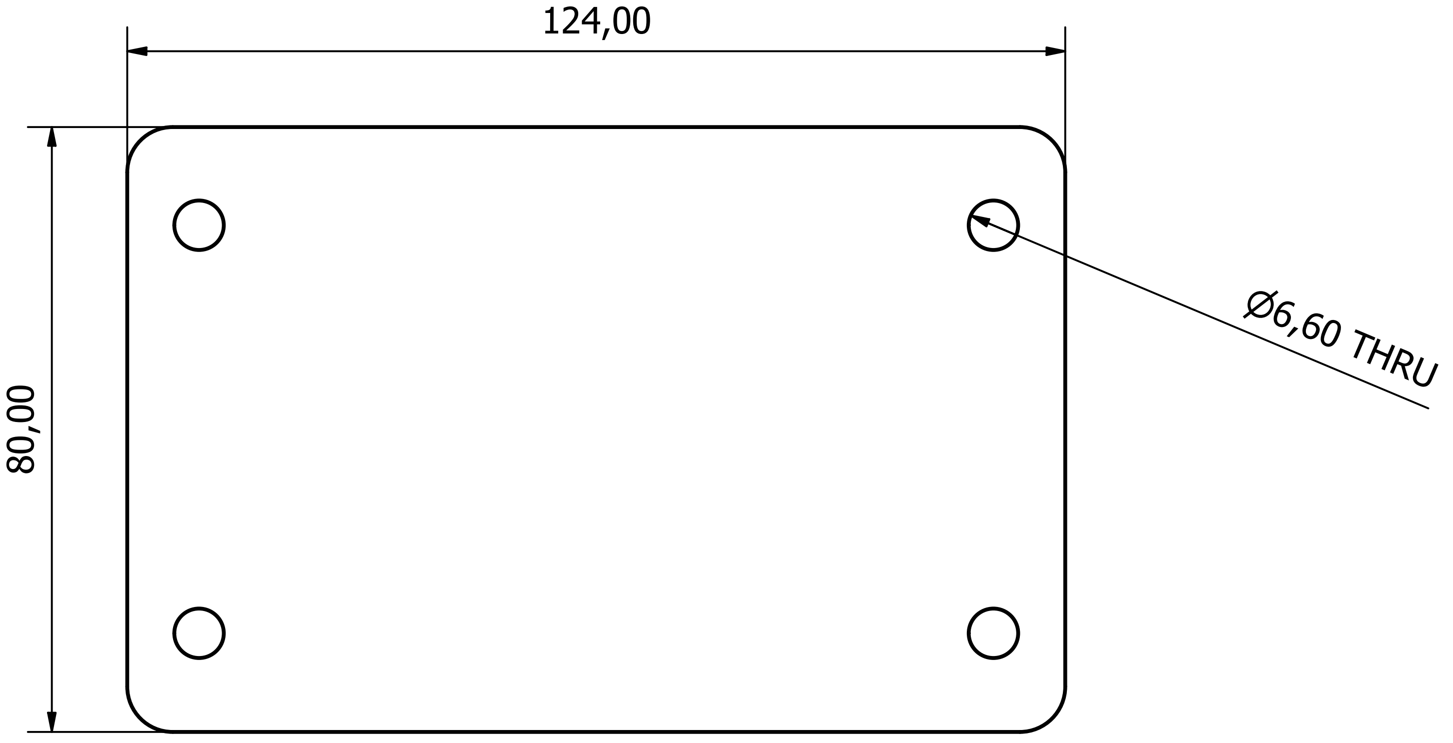
Sheet Thickness: 6,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Traction Motor Mount Flange

Swerve Drive AGV	Edition	Sheet
		149 / 207



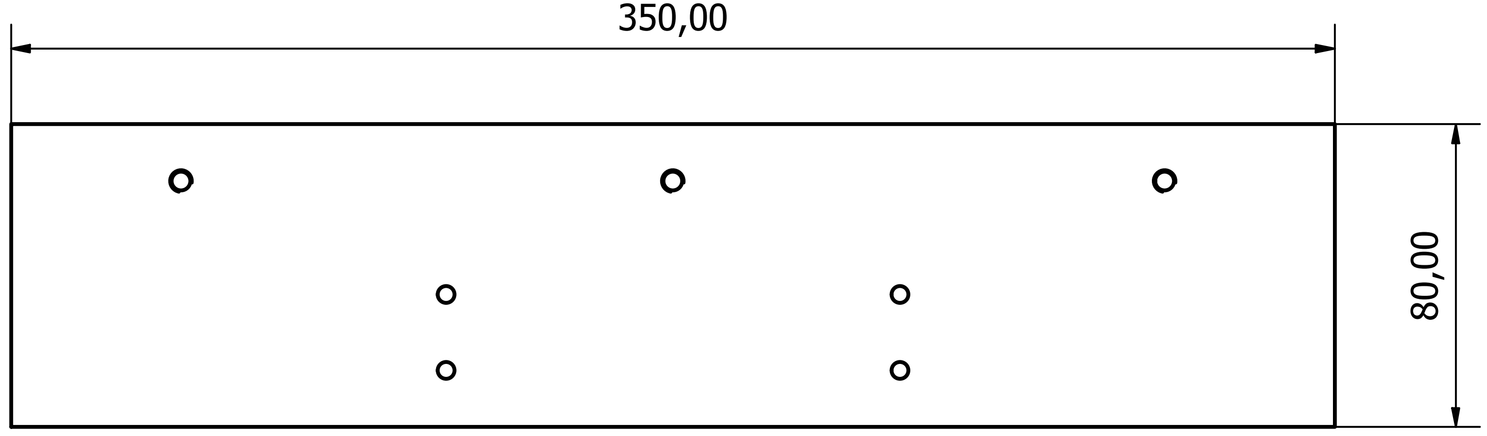
Sheet Thickness: 6,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	External	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Traction Motor Mount Base

Swerve Drive AGV	Edition	Sheet
		150 / 207

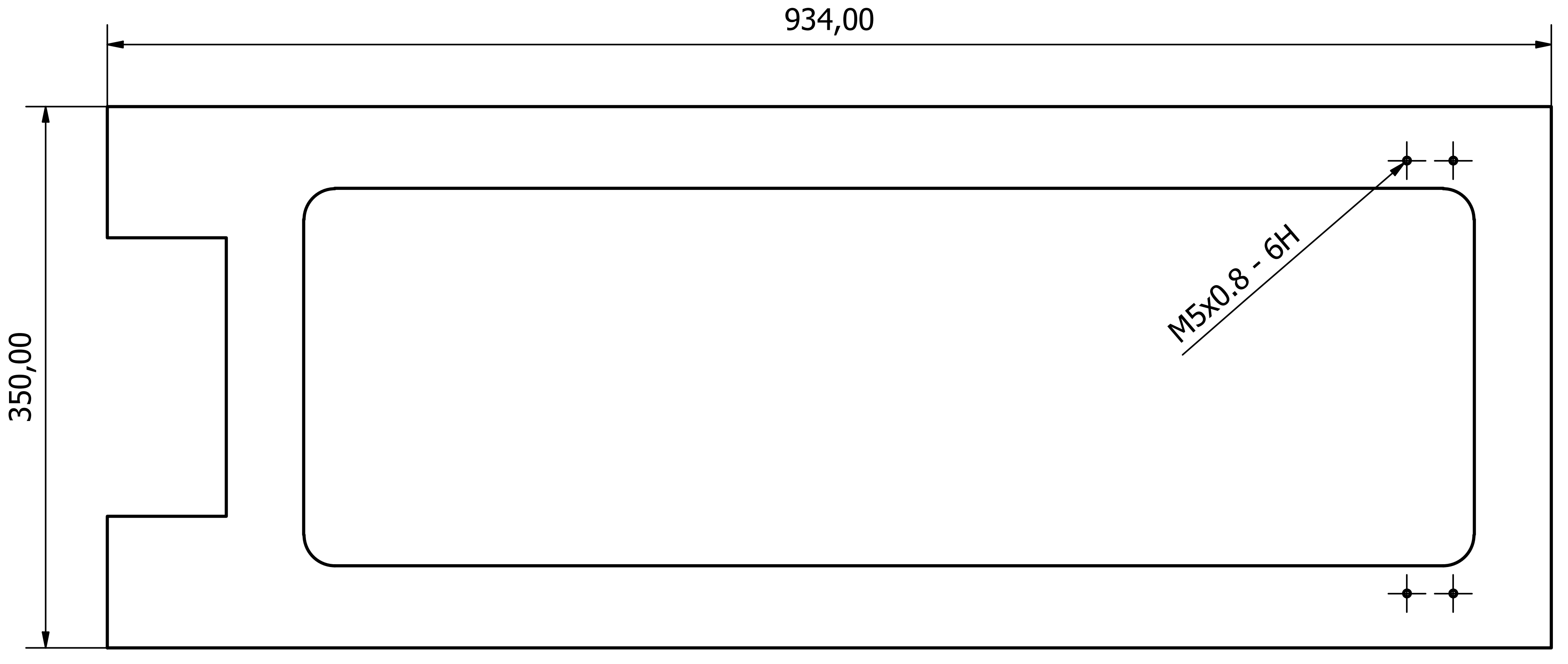


Sheet Thickness: 3,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	LPB	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Battery Box Base End Plate A		
Swerve Drive AGV	Edition	Sheet 151 / 207



Sheet Thickness: 3,00 mm

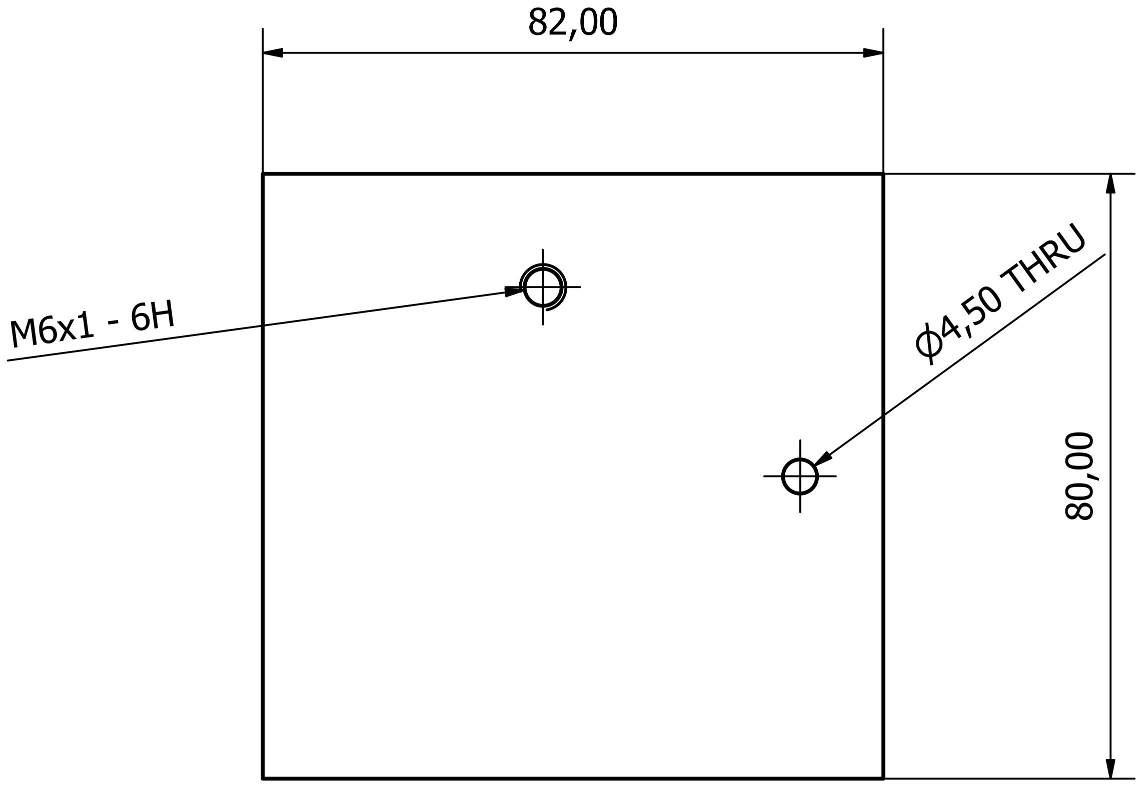
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	LPB	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Base Plate

Swerve Drive AGV

Edition	Sheet
	152 / 207



Sheet Thickness: 3,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	LPB	Steel, Mild	07/03/2019

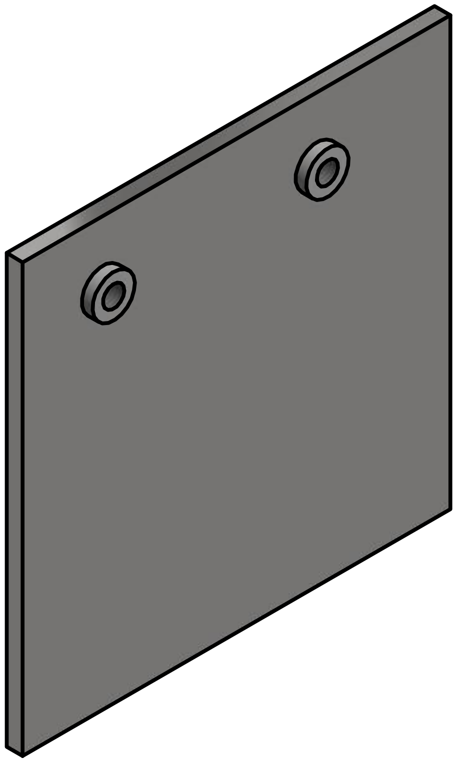
NELSON MANDELA UNIVERSITY

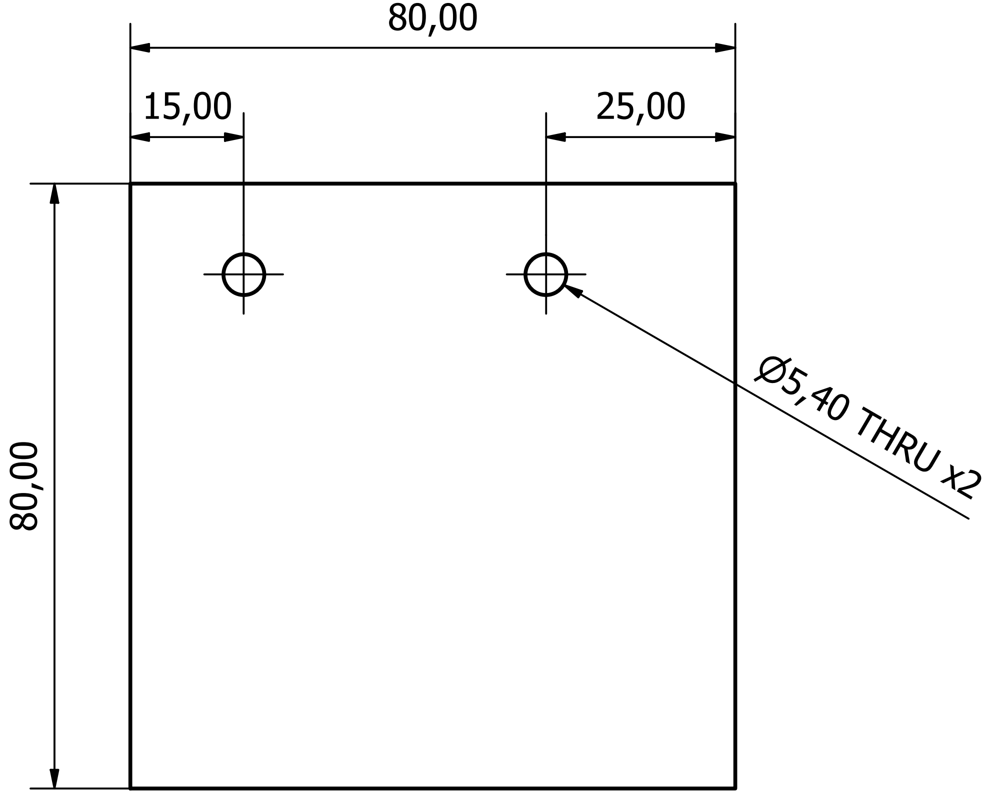
Base Side Corner B

Swerve Drive AGV

Edition	Sheet
	153 / 207

PARTS LIST		
ITEM	QTY	PART NUMBER
1	1	Base Side Corner A (Left)
2	2	SCN M4 - CM4-0





Sheet Thickness: 3,00 mm

HPI Inserts
Number : 2 unit
Designation : CM4-0 (SCN M4)

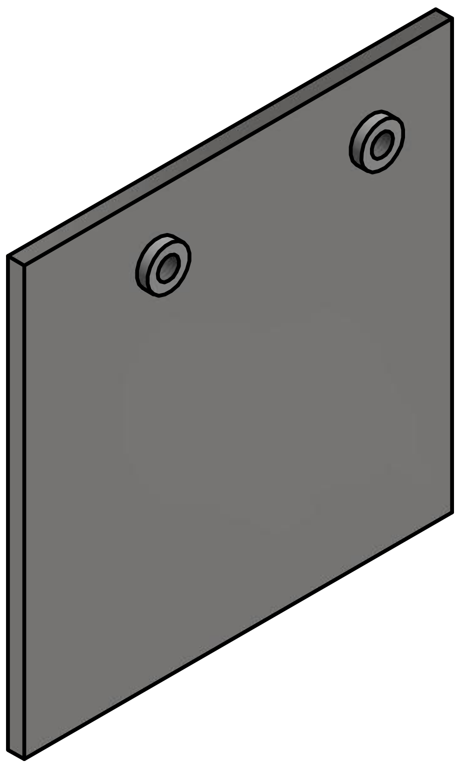
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	LPB	Mild Steel	07/03/2019

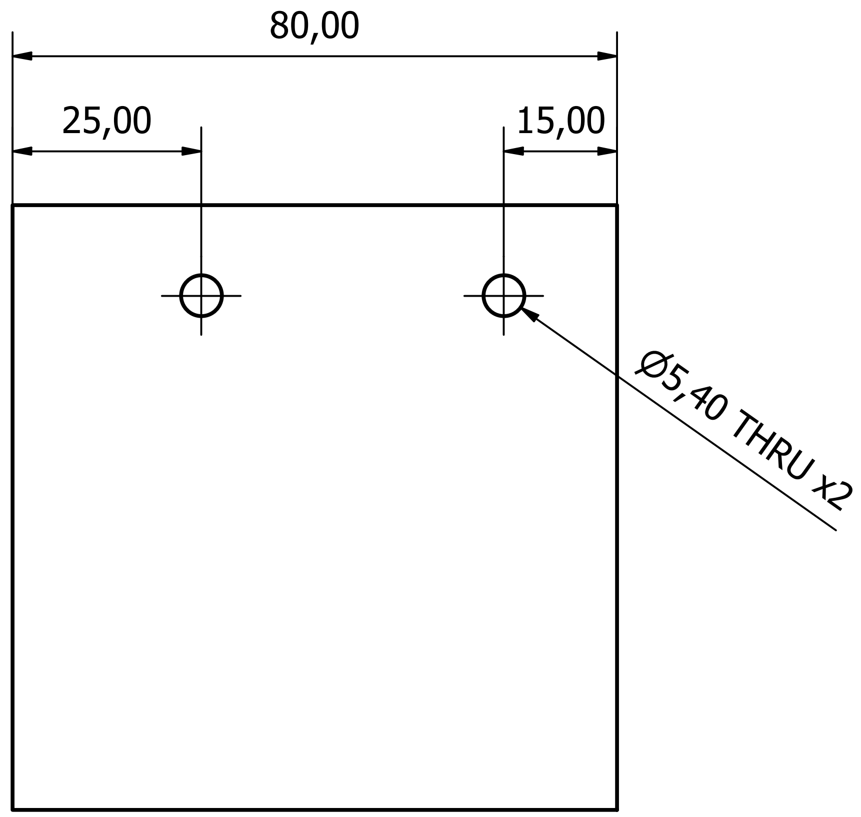
NELSON MANDELA UNIVERSITY

Base Side Corner A (Left)

	Edition	Sheet
Swerve Drive AGV		154 / 207

PARTS LIST		
ITEM	QTY	PART NUMBER
1	1	Base Side Corner A (Right)
2	2	SCN M4 - CM4-0





Sheet Thickness: 3,00 mm

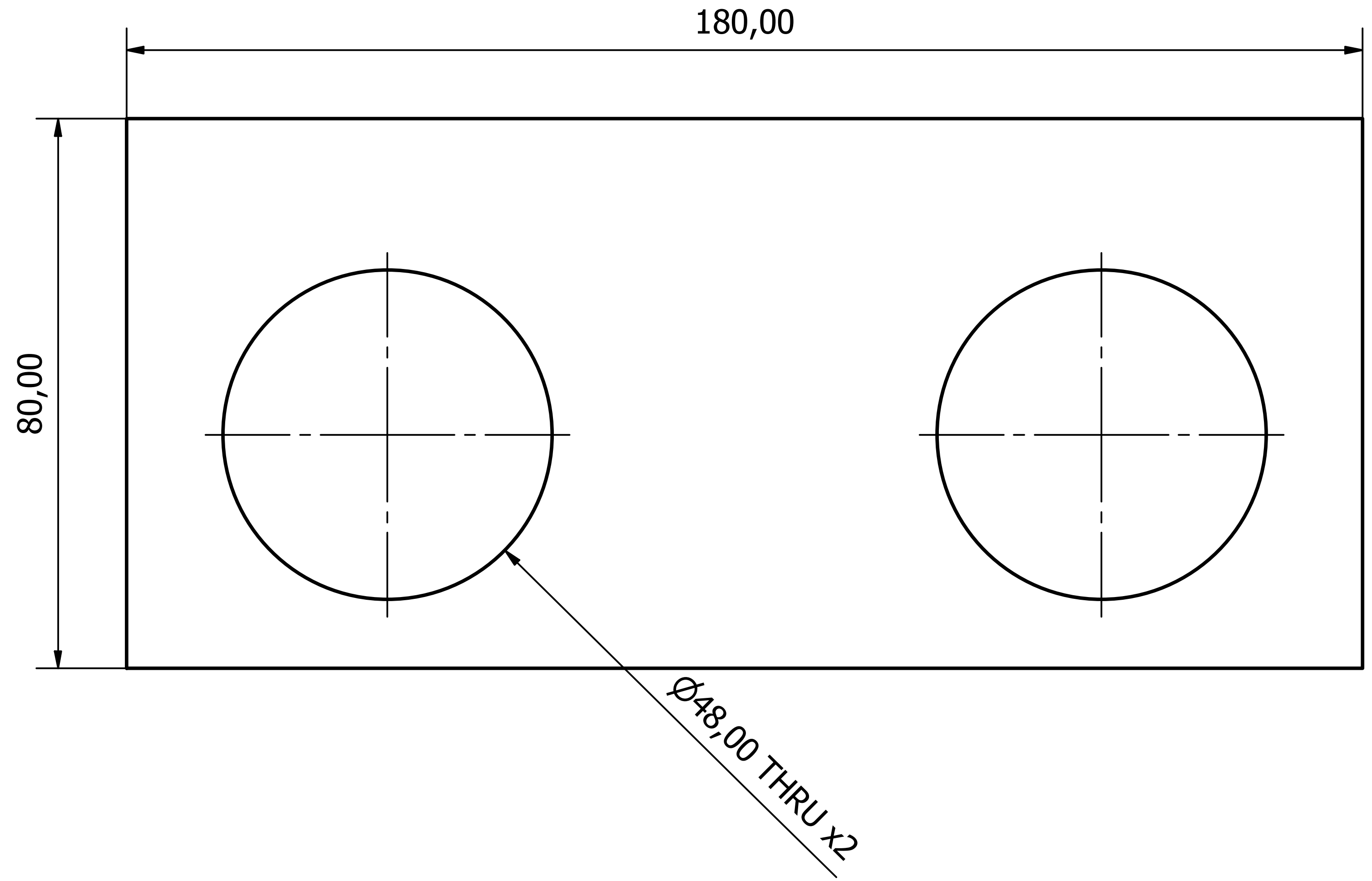
HPI Inserts
Number : 2 unit
Designation : CM4-0 (SCN M4)

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	LPB	Mild Steel	07/03/2019

NELSON MANDELA UNIVERSITY

Base Side Corner A (Right)

	Edition	Sheet
Swerve Drive AGV		155 / 207



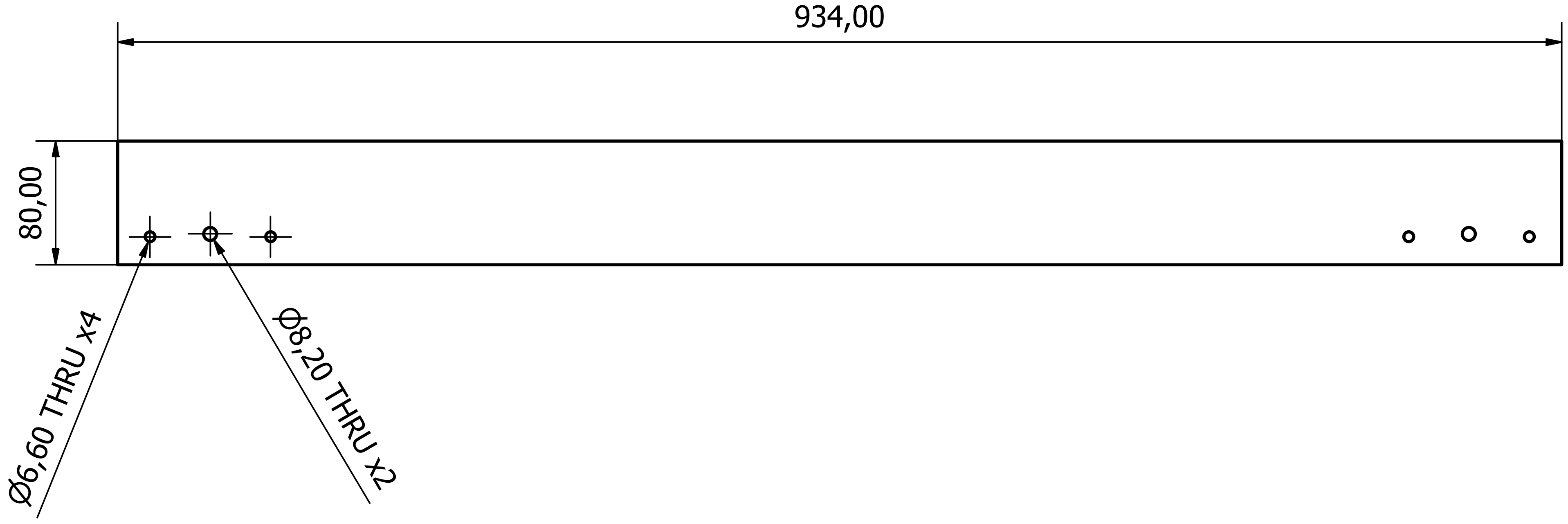
Sheet Thickness: 3,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	LPB	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Base Side Corner C

Swerve Drive AGV	Edition	Sheet
		156 / 207



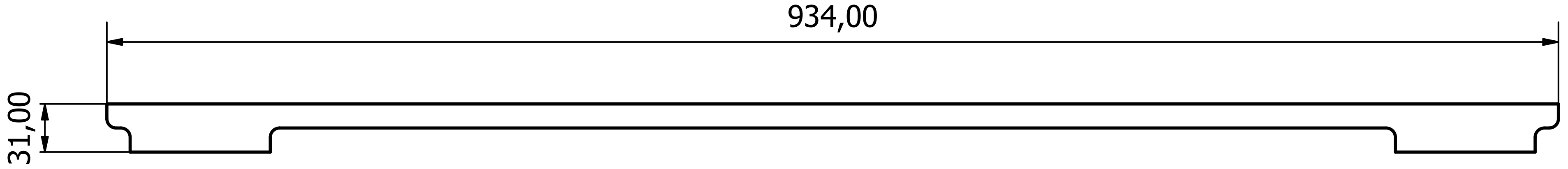
Sheet Thickness: 3,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	LPB	Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Base Side Long Edge

Swerve Drive AGV	Edition	Sheet
		157 / 207



Sheet Thickness: 3,00 mm

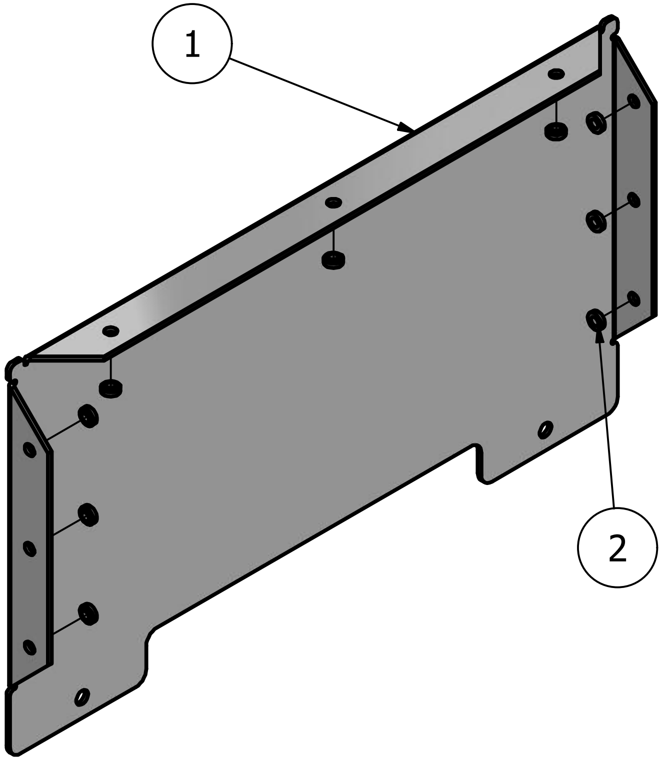
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2	LPB	Steel, Mild	07/03/2019

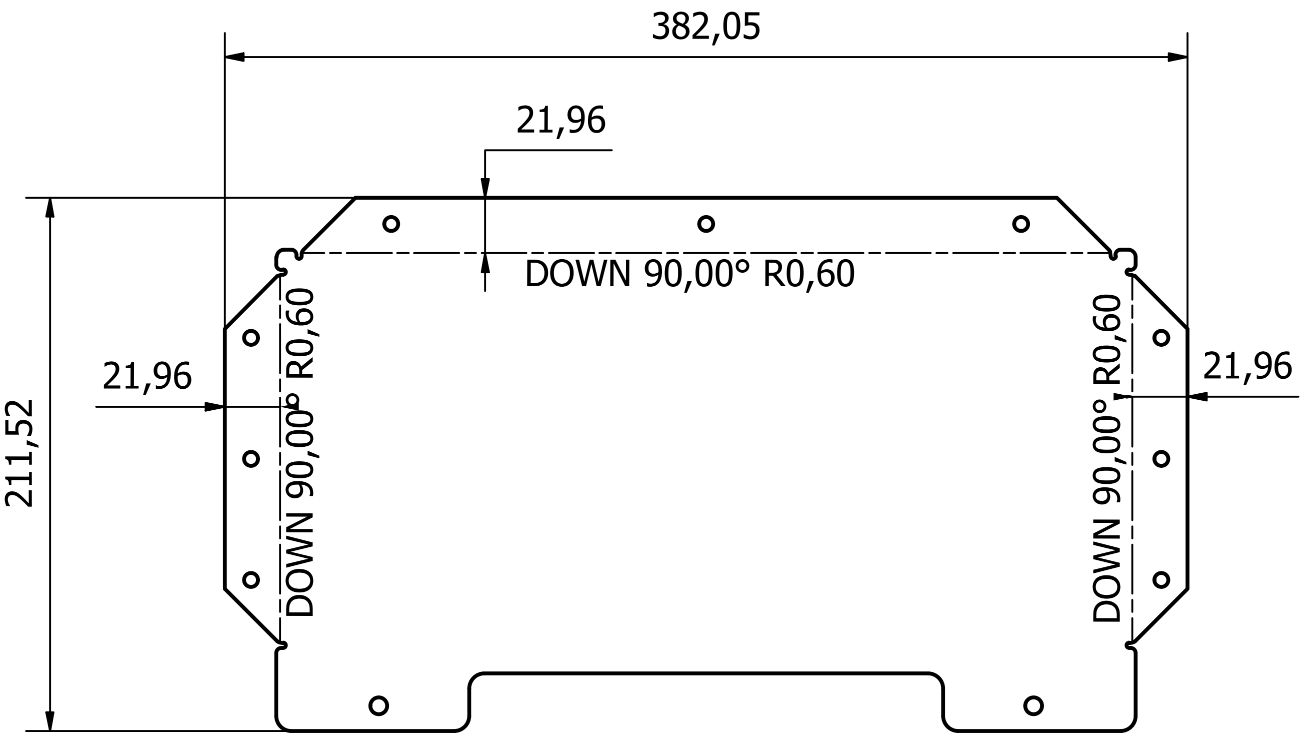
NELSON MANDELA UNIVERSITY

Base Side Flange

Swerve Drive AGV

Edition	Sheet
	158 / 207





HPI Inserts
Number : 9 unit
Designation : CM4-0 (SCN M4)

Sheet Thickness: 2,00 mm

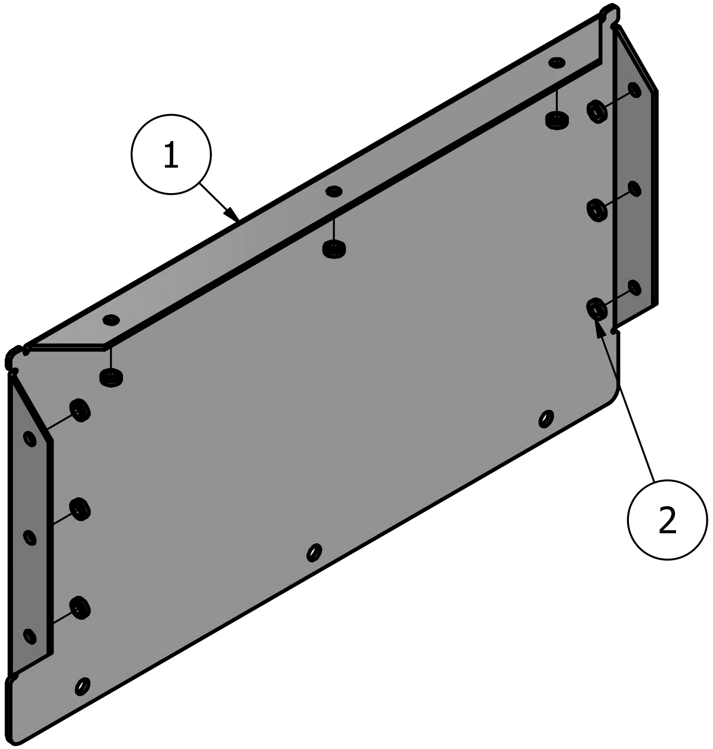
PARTS LIST		
ITEM	QTY	PART NUMBER
1	1	Battery Bank Lid Side B
2	9	SCN M4 - CM4-0

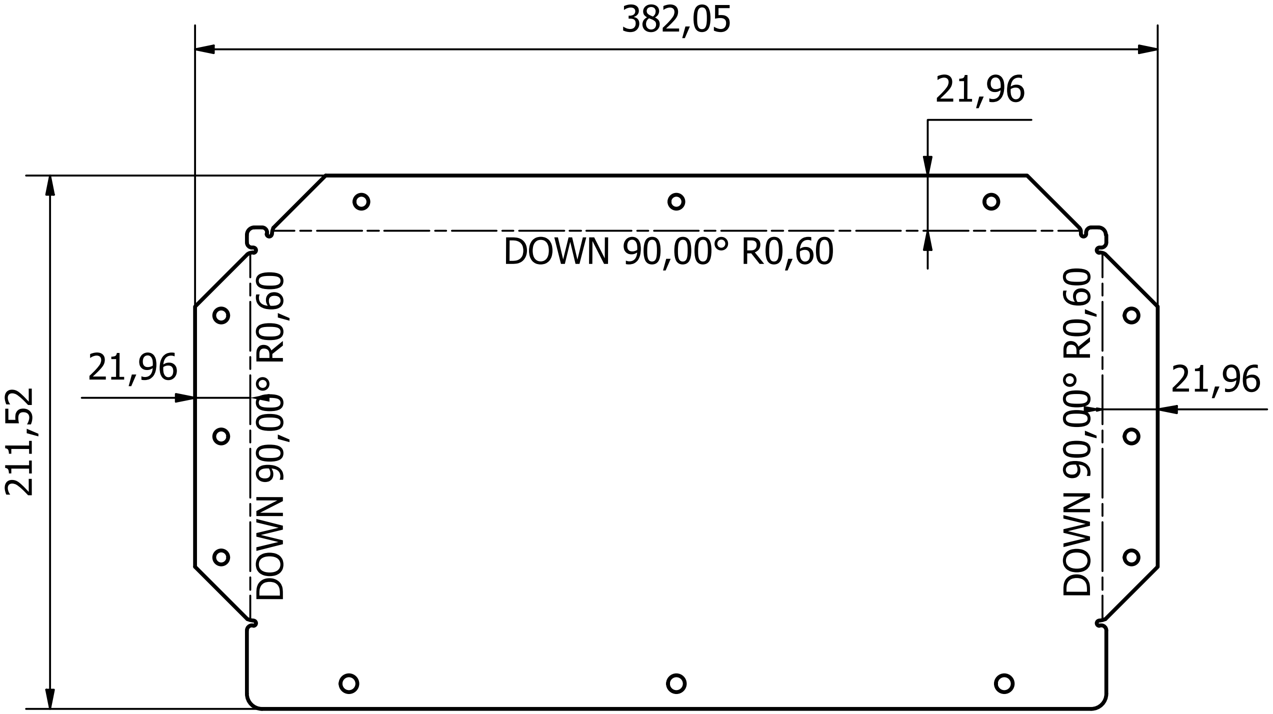
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	LPB	Aluminum	07/03/2019

NELSON MANDELA UNIVERSITY

Battery Bank Lid Side B

Swerve Drive AGV	Edition	Sheet
		159 / 207





HPI Inserts
Number : 9 unit
Designation : CM4-0 (SCN M4)

Sheet Thickness: 2,00 mm

PARTS LIST

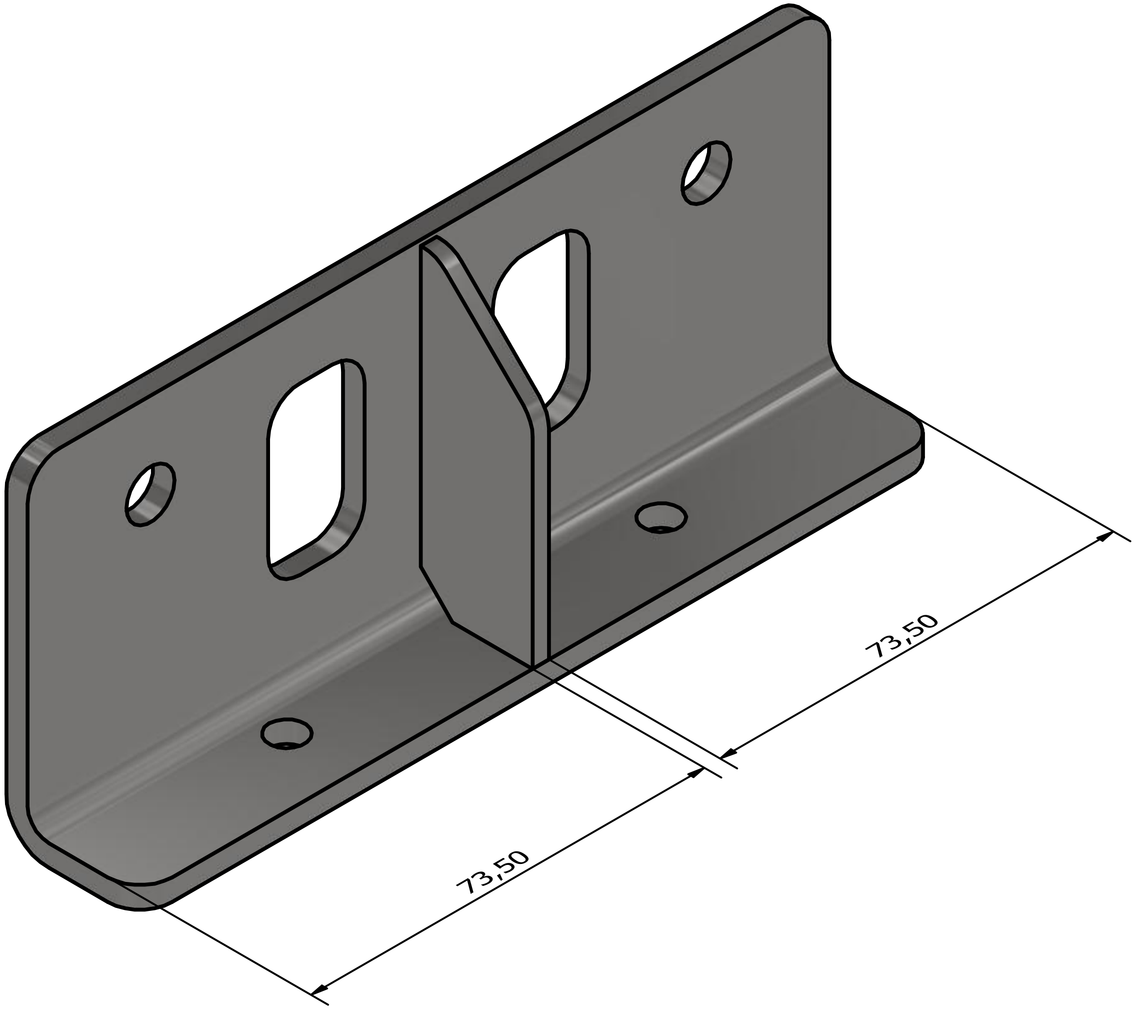
ITEM	QTY	PART NUMBER
1	1	Battery Bank Lid Side A
2	9	SCN M4 - CM4-0

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1	LPB	Aluminum	07/03/2019

NELSON MANDELA UNIVERSITY

Battery Bank Lid Side A

	Edition	Sheet
Swerve Drive AGV		160 / 207



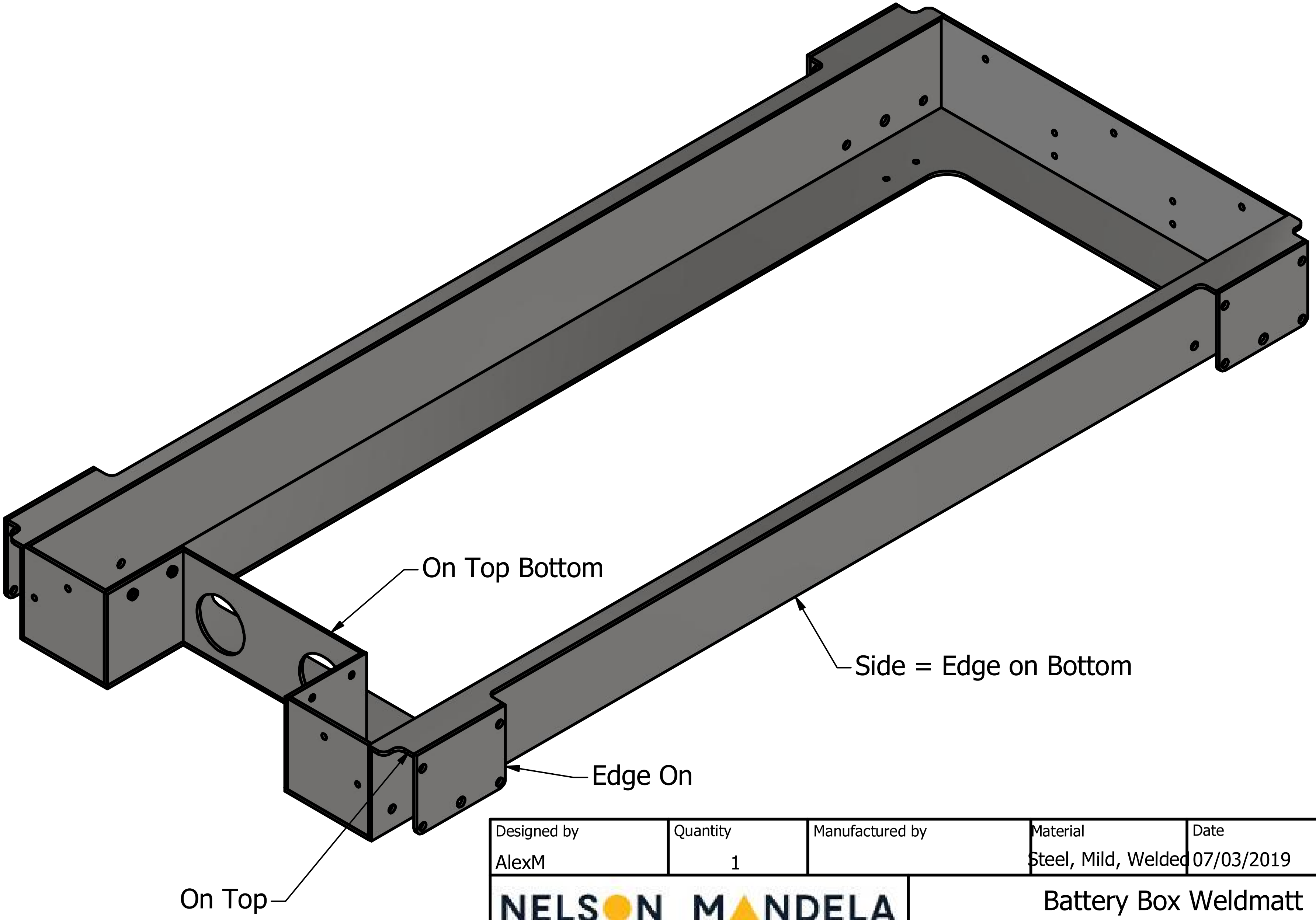
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1		Steel, Mild, Welded	07/03/2019

NELSON MANDELA UNIVERSITY

Battery Contactor Weldmatt

Swerve Drive AGV

Edition	Sheet
	161 / 207

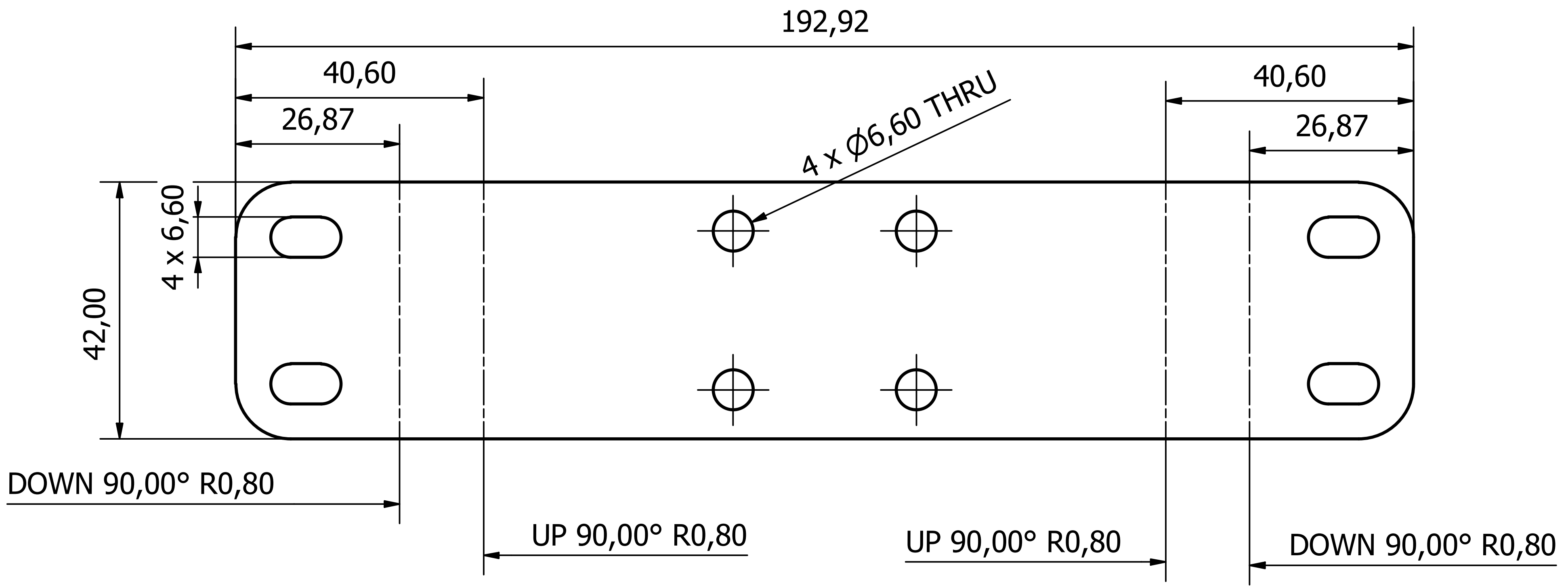


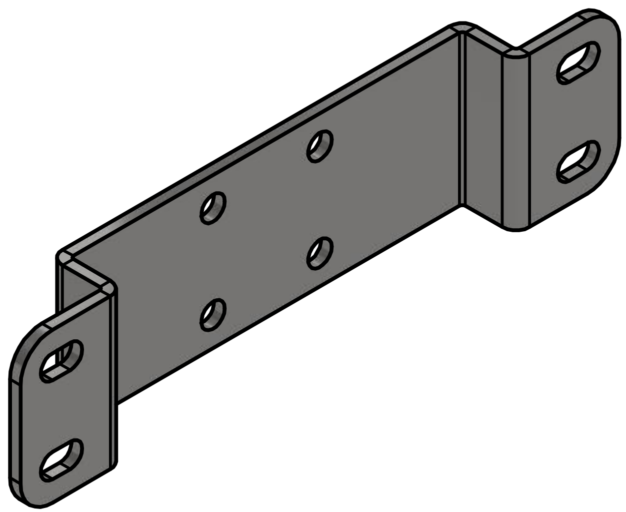
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1		Steel, Mild, Welded	07/03/2019

NELSON MANDELA UNIVERSITY

Battery Box Weldmatt

	Edition	Sheet
Swerve Drive AGV		162 / 207





Sheet Thickness: 3,00mm

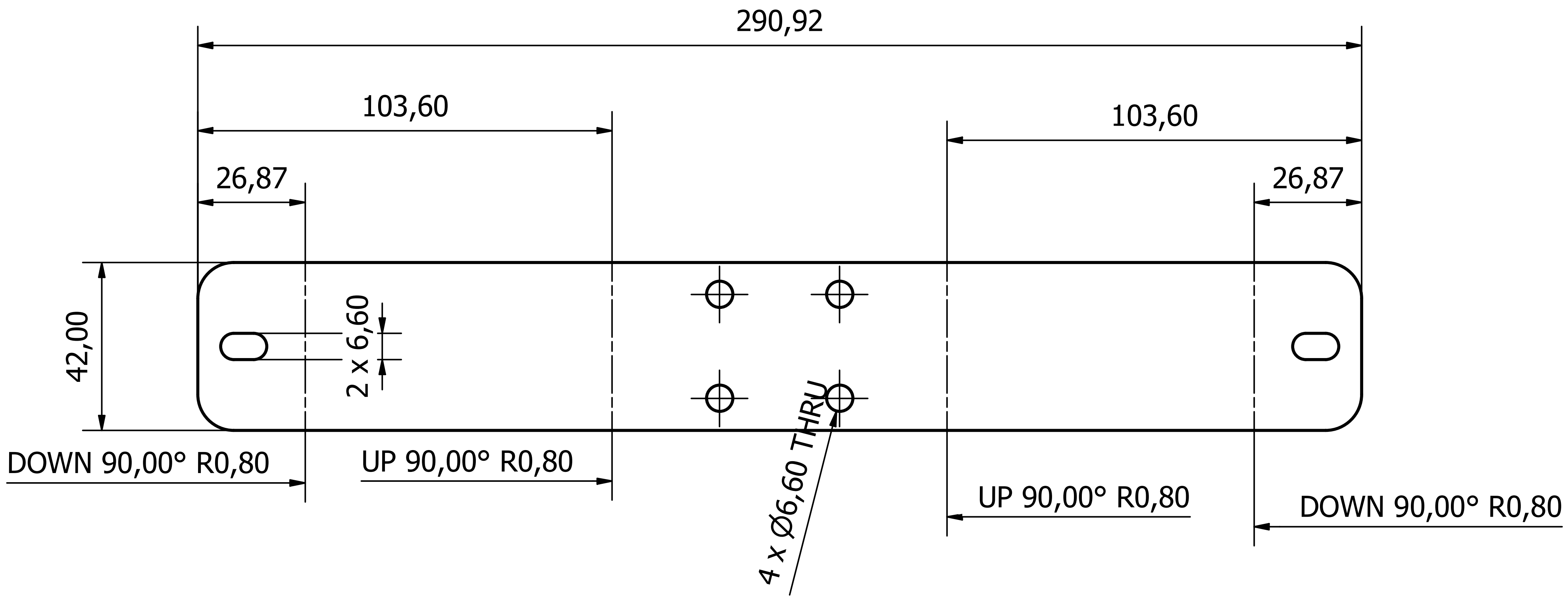
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1		Steel, Mild	07/03/2019

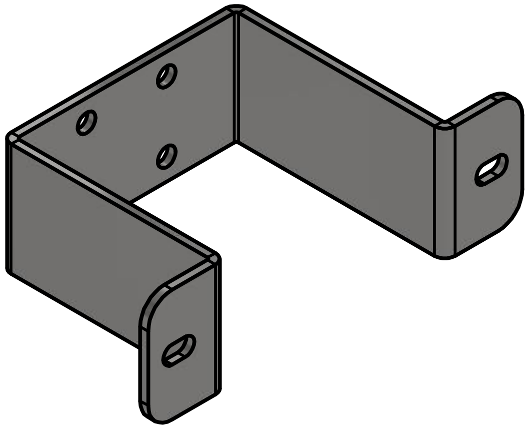
NELSON MANDELA UNIVERSITY

LiDAR Top Support Bracket

Swerve Drive AGV

Edition	Sheet
	163 / 207





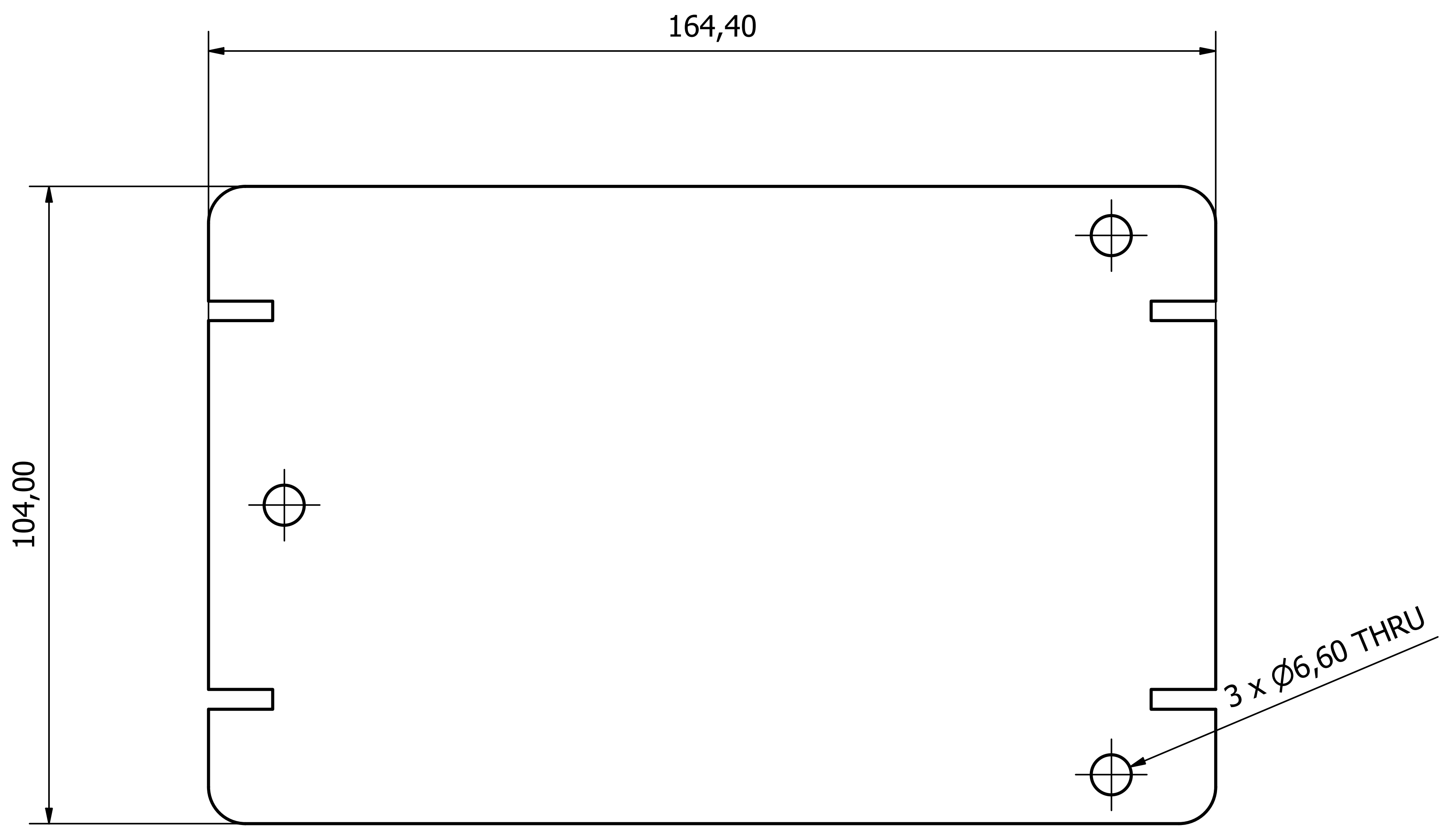
Sheet Thickness: 3,00mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1		Steel, Mild	07/03/2019

NELSON MANDELA
UNIVERSITY

LiDAR Bottom Support Bracket

	Edition	Sheet
Swerve Drive AGV		164 / 207



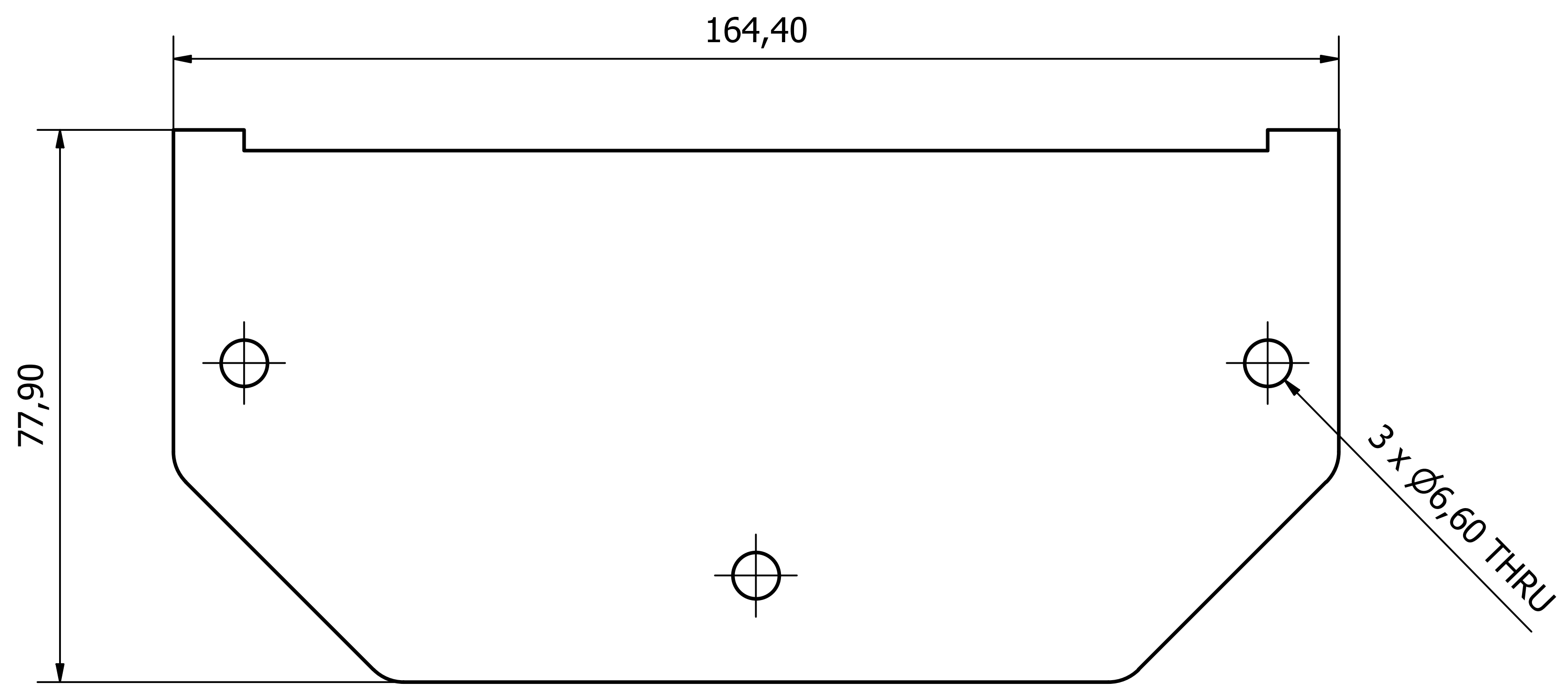
Sheet Thickenss 3,00mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1		Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

LiDAR Backing Plate Main

Swerve Drive AGV	Edition	Sheet
		165 / 207



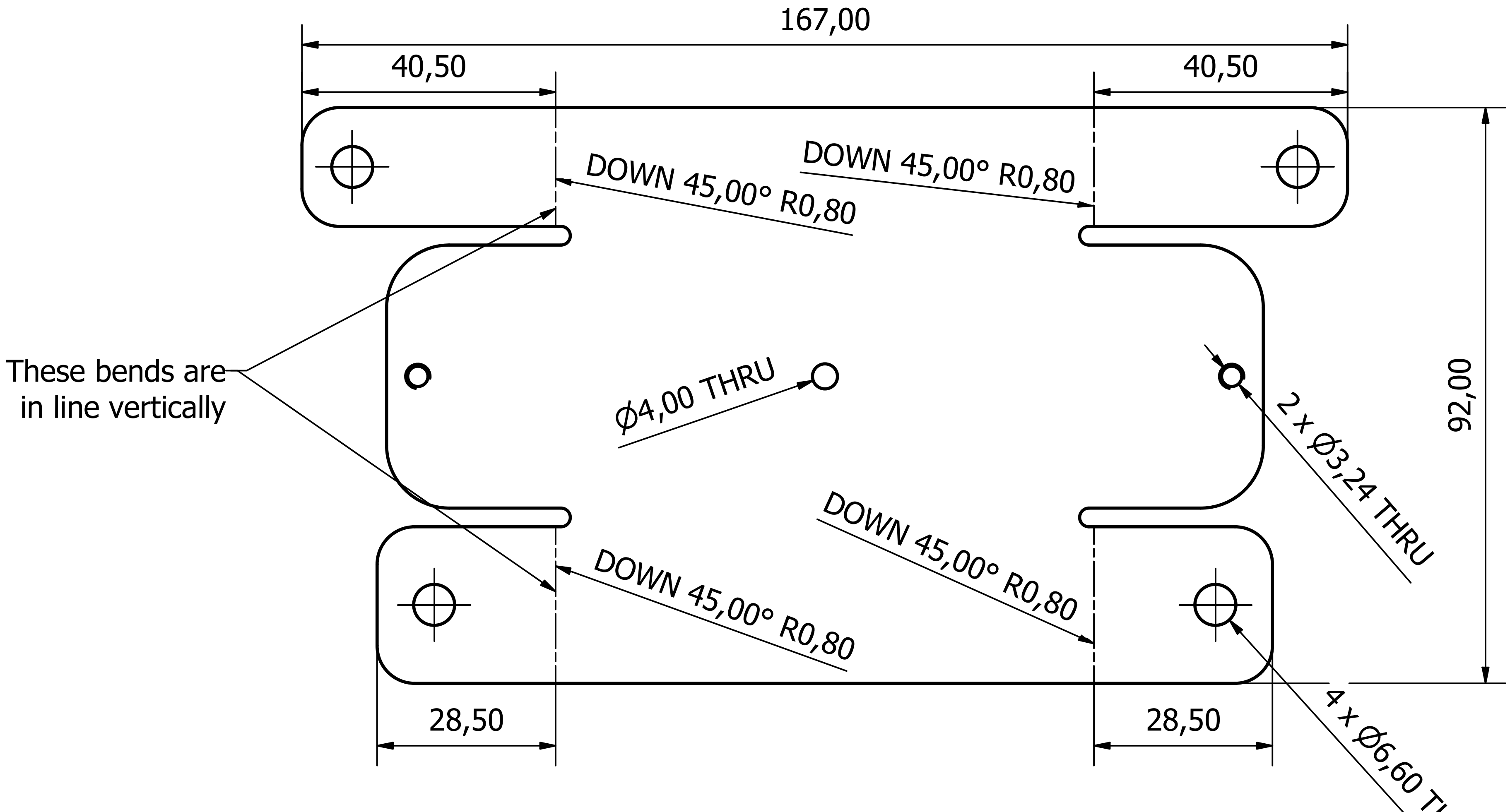
Sheet Thickness: 3,00mm

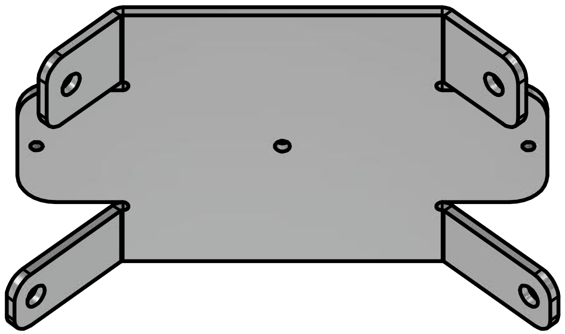
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2		Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

LiDAR Backing Plate Web

	Edition	Sheet
Swerve Drive AGV		166 / 207





Sheet Thickness: 3,00mm

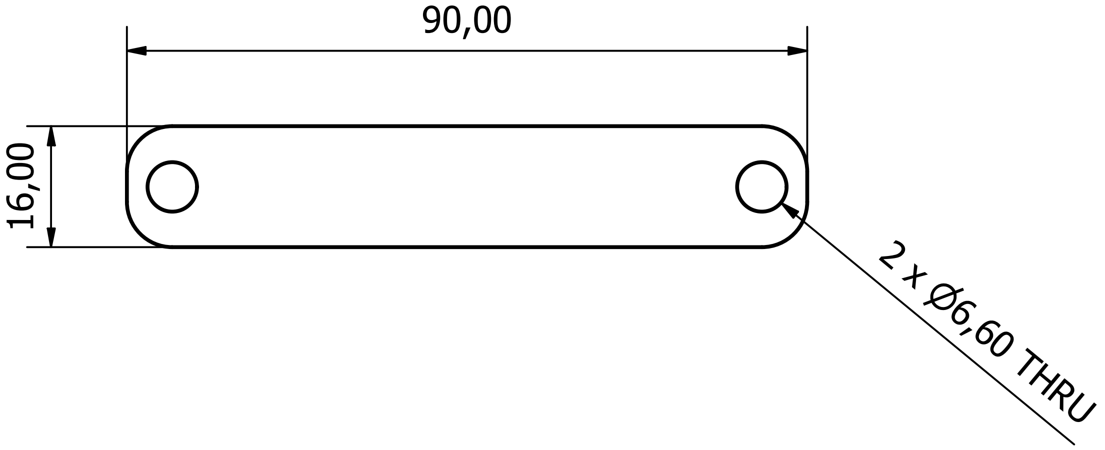
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2		Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

S300 Anchor Bracket

Swerve Drive AGV

Edition	Sheet
	167 / 207



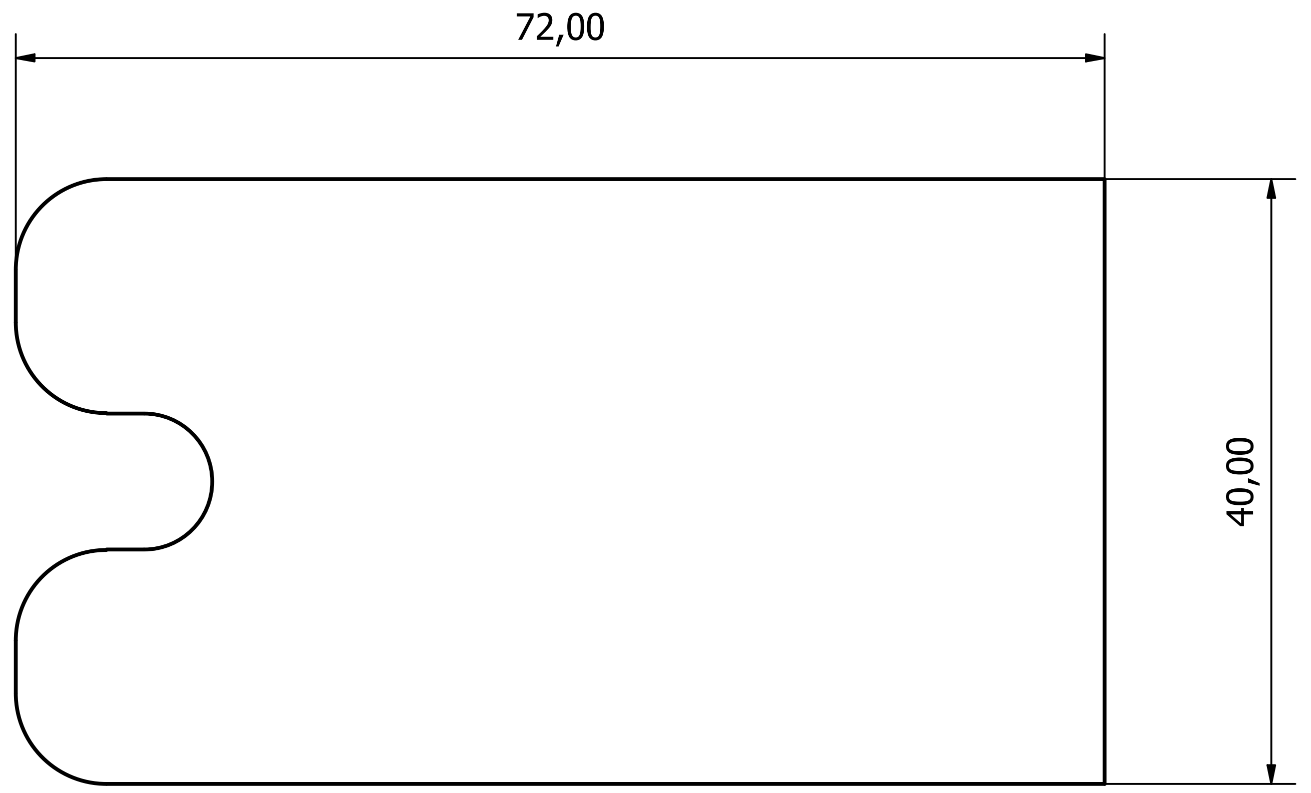
Sheet Thickness: 5,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2		Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Rack Shim Plate

	Edition	Sheet
Swerve Drive AGV		168 / 207



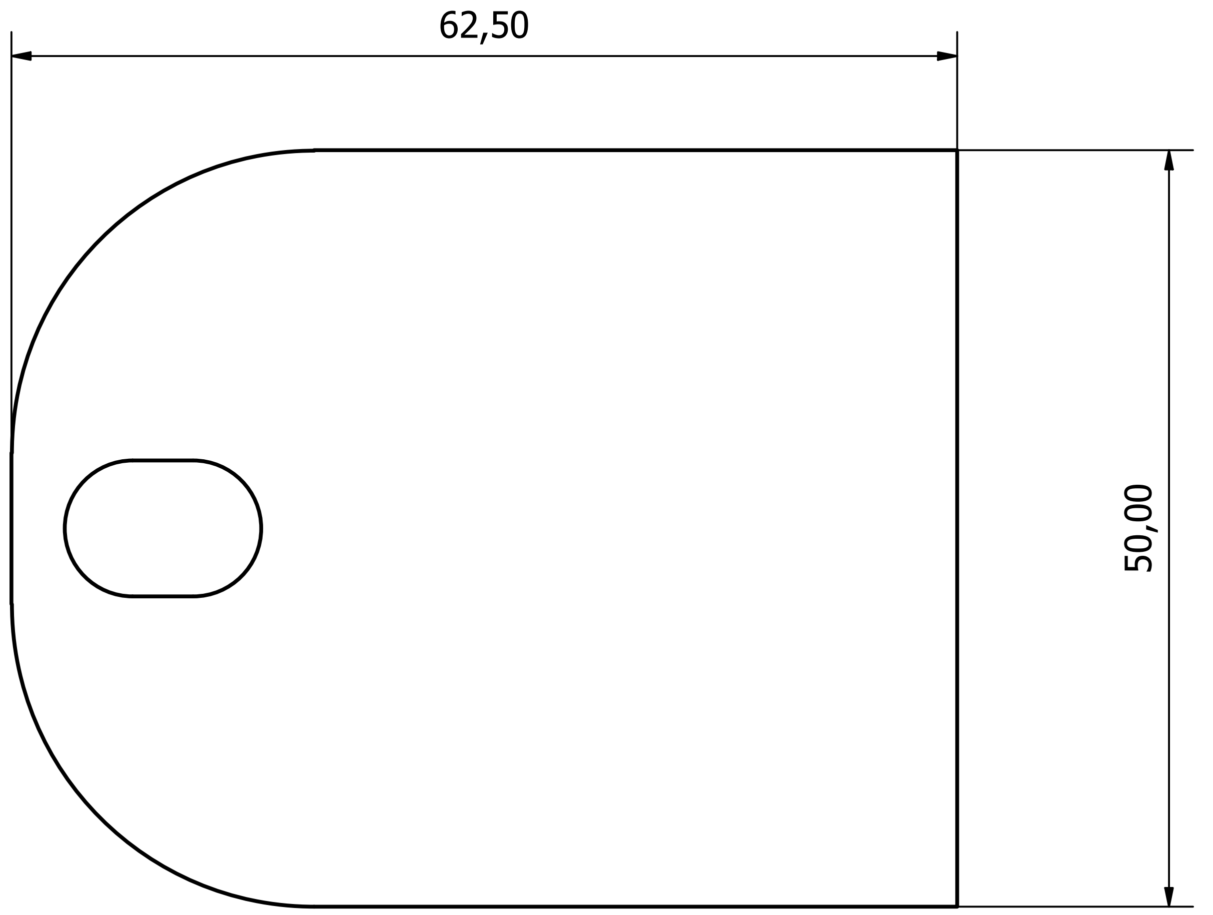
Sheet Thickness: 3,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	4		Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Encoder Foot Plate Narrow

Swerve Drive AGV	Edition	Sheet
		169 / 207



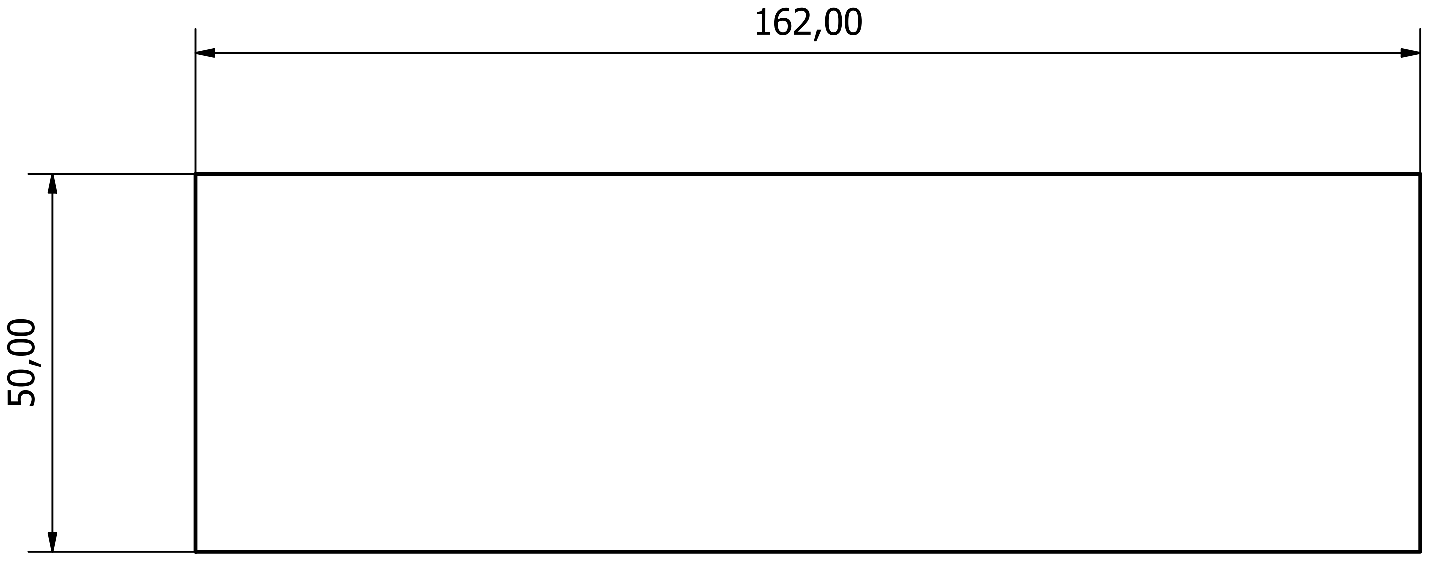
Sheet Thickness: 3,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2		Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Encoder Foot Plate Wide

Swerve Drive AGV	Edition	Sheet
		170 / 207



Sheet Thickness: 3,00 mm

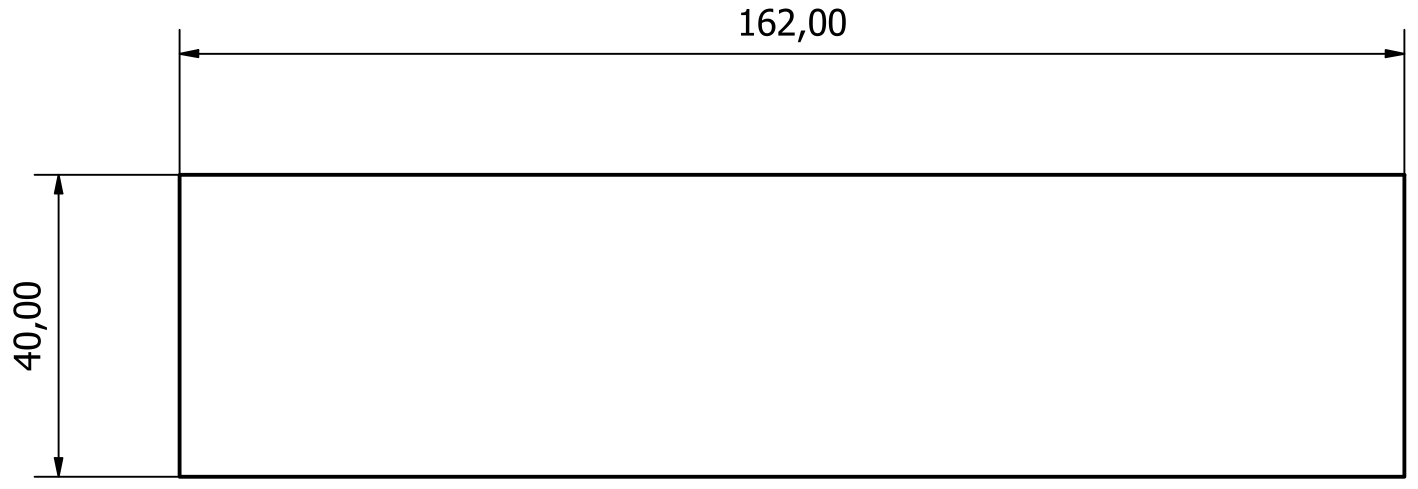
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2		Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Wide Encoder Pillar

Swerve Drive AGV

Edition	Sheet
	171 / 207



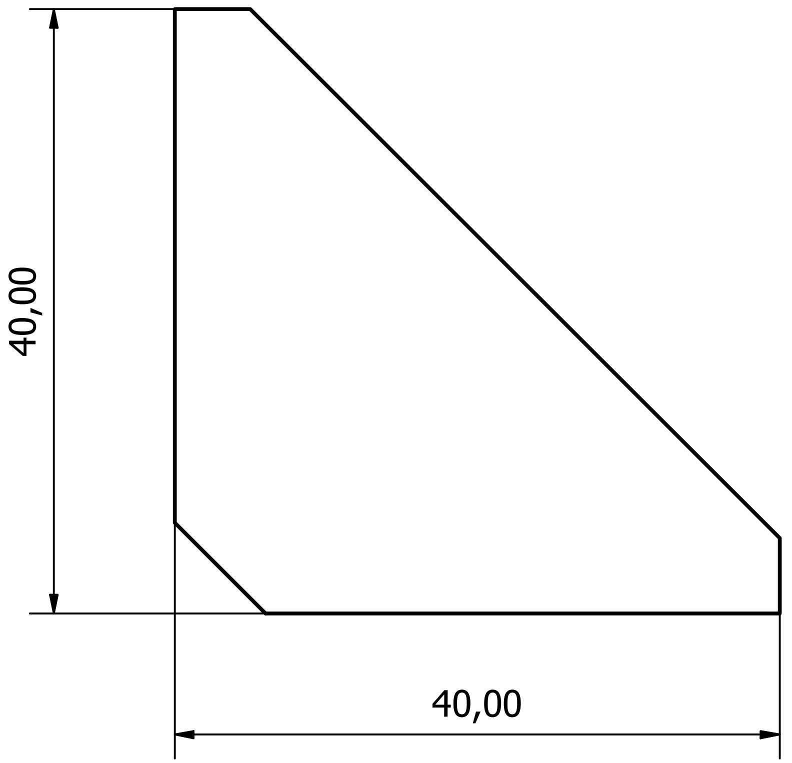
Sheet Thickness: 3,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	4		Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Narrow Encoder Pillar

Swerve Drive AGV	Edition	Sheet
		172 / 207



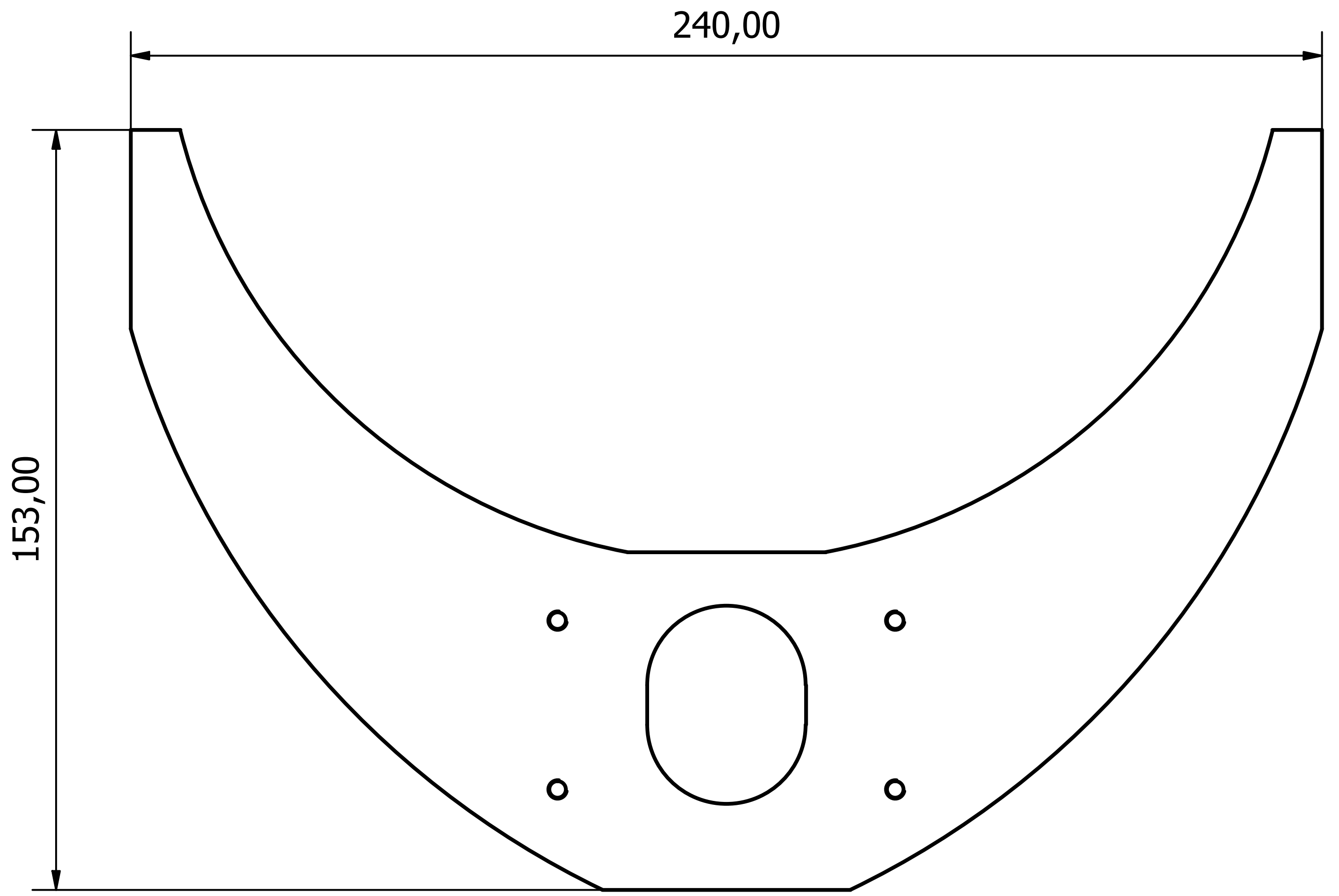
Sheet Thickness: 3,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	6		Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Encoder Brace Corner

Swerve Drive AGV	Edition	Sheet
		173 / 207



Sheet Thickness: 3,00 mm

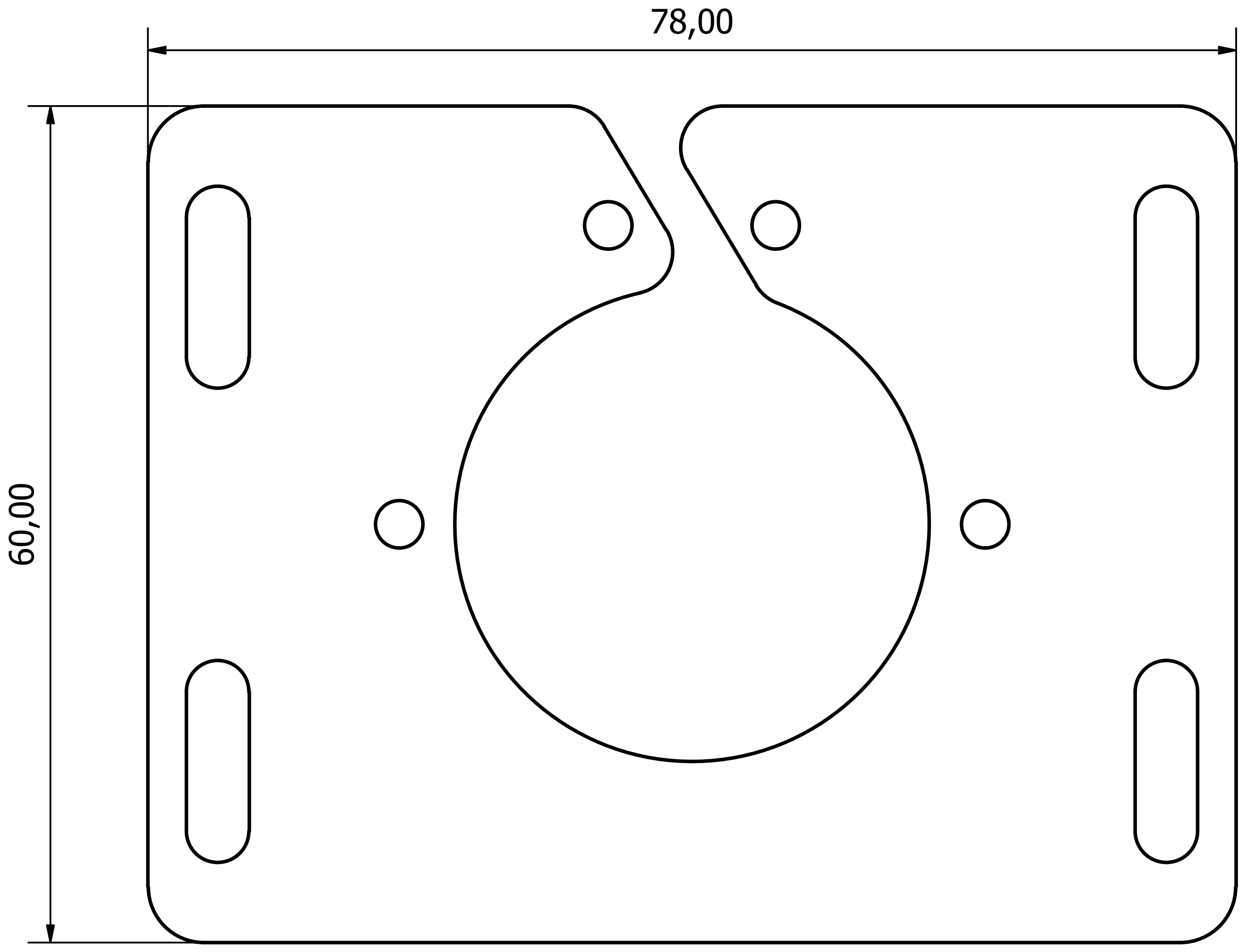
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2		Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

Half Moon Encoder Mount

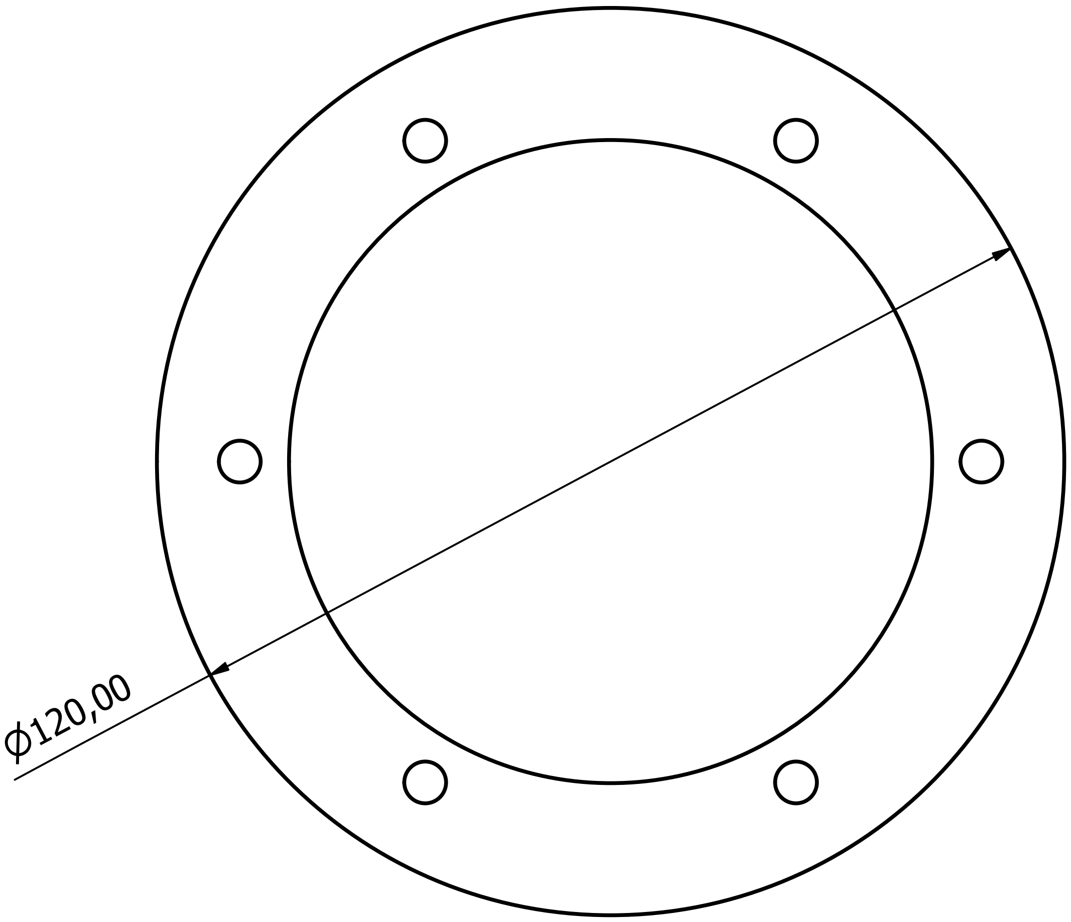
Swerve Drive AGV

Edition	Sheet
	174 / 207



Sheet Thickness: 3,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2		Aluminum 6061	07/03/2019
NELSON MANDELA UNIVERSITY	Encoder Slide Plate			
	Swerve Drive AGV		Edition	Sheet 175 / 207



Sheet Thickness: 1,50 mm

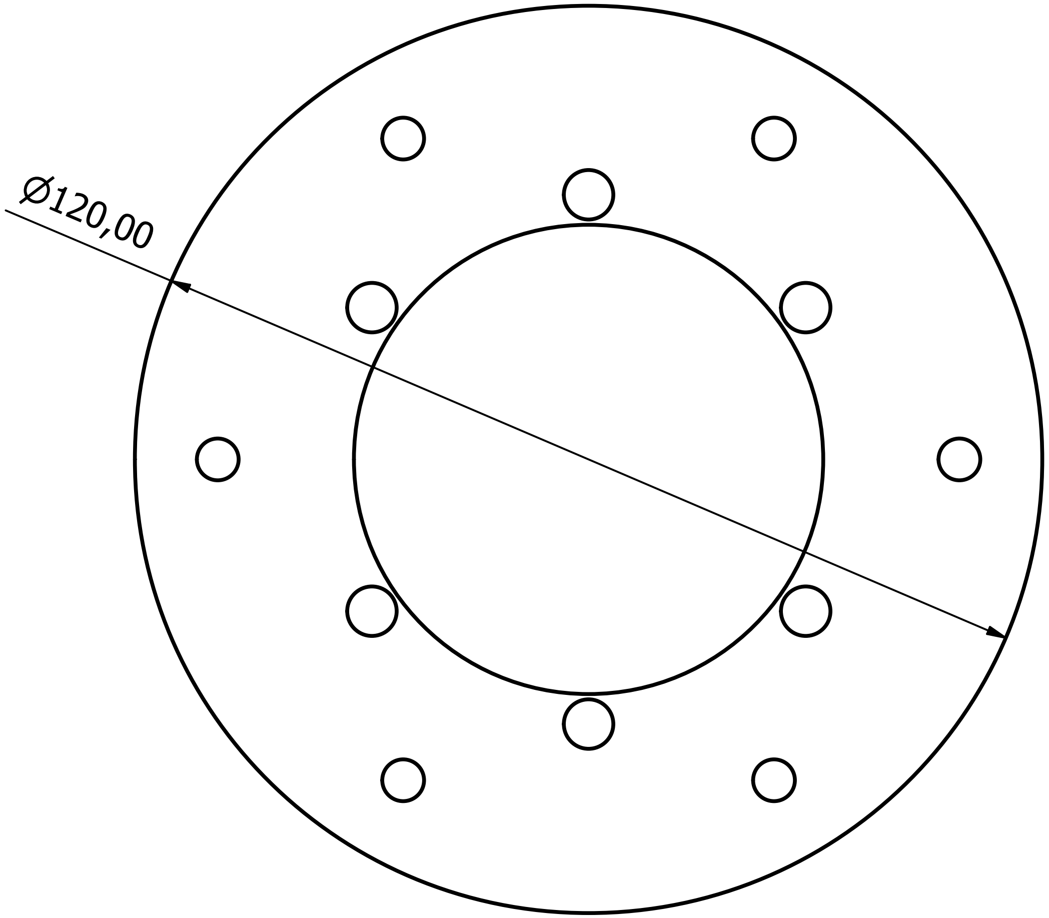
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2		Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Encoder Driven Gear Aux Flange

Swerve Drive AGV

Edition	Sheet
	176 / 207



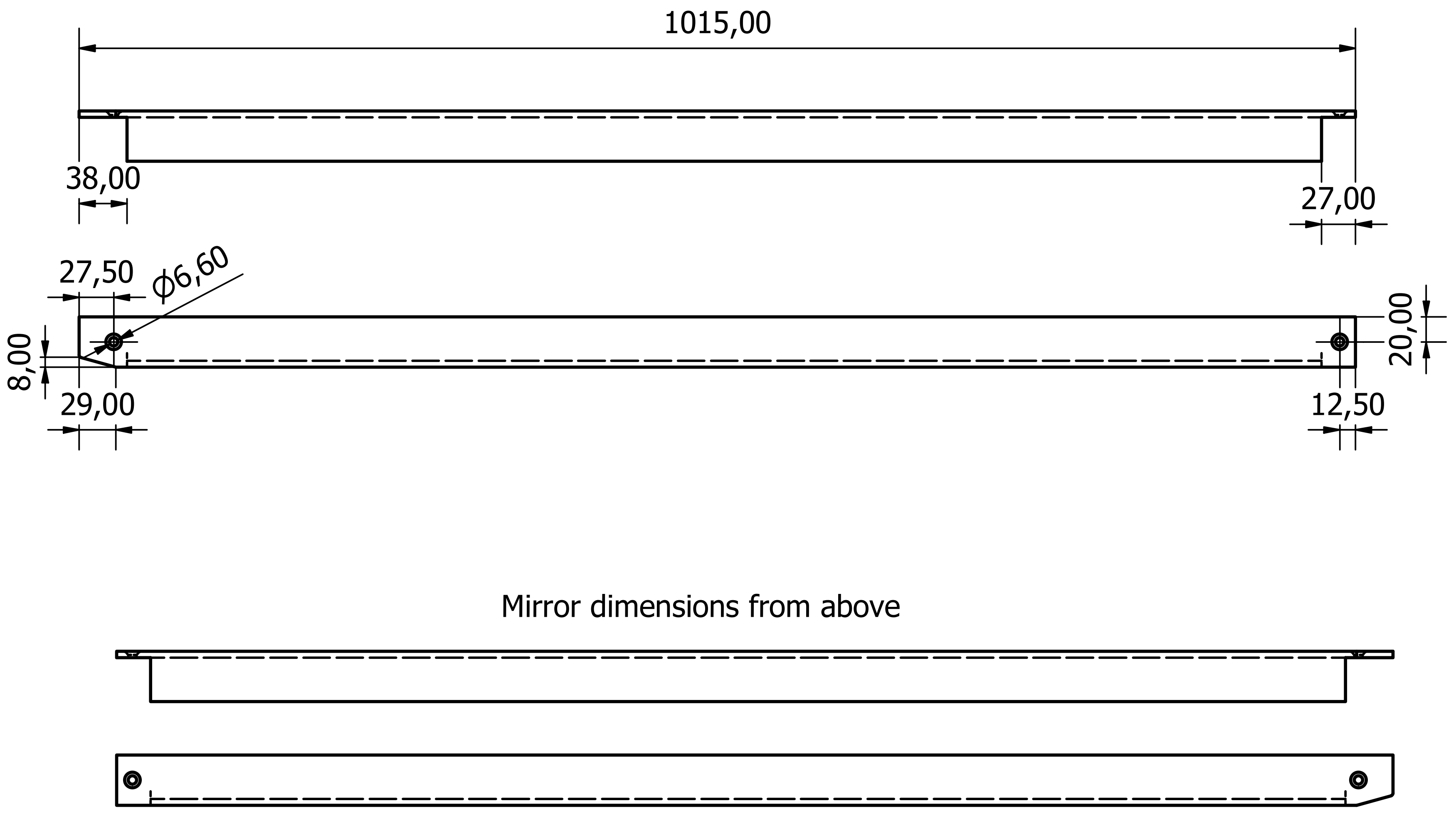
Sheet Thickness: 3,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2		Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Encoder Driven Gear Main Flange

Swerve Drive AGV	Edition	Sheet
		177 / 207

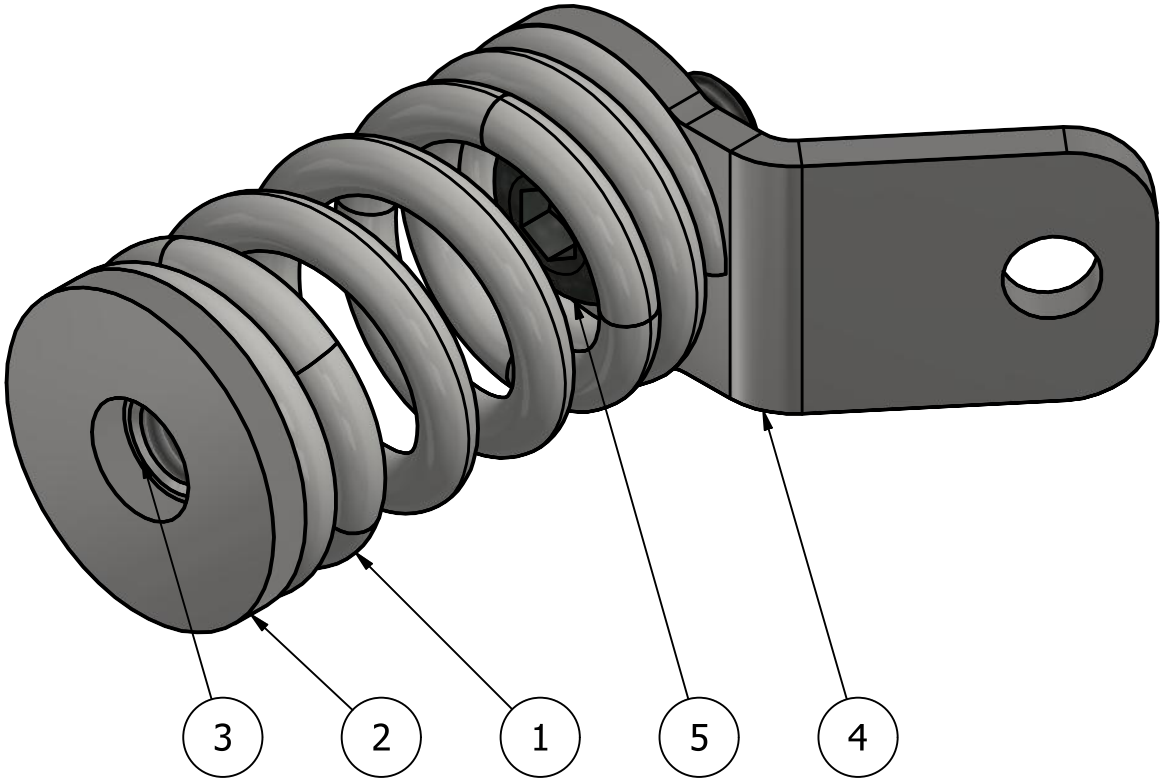


Mirror dimensions from above

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1		Steel, Mild	07/03/2019

NELSON MANDELA UNIVERSITY

	Edition	Sheet
Swerve Drive AGV		178 / 207



PARTS LIST		
ITEM	QTY	PART NUMBER
1	1	Compress Spring1
2	1	Spring Ends A
3	1	ISO 4032 - M8
4	1	Spring Ends B
5	1	ISO 7380-1 - M8 x 12

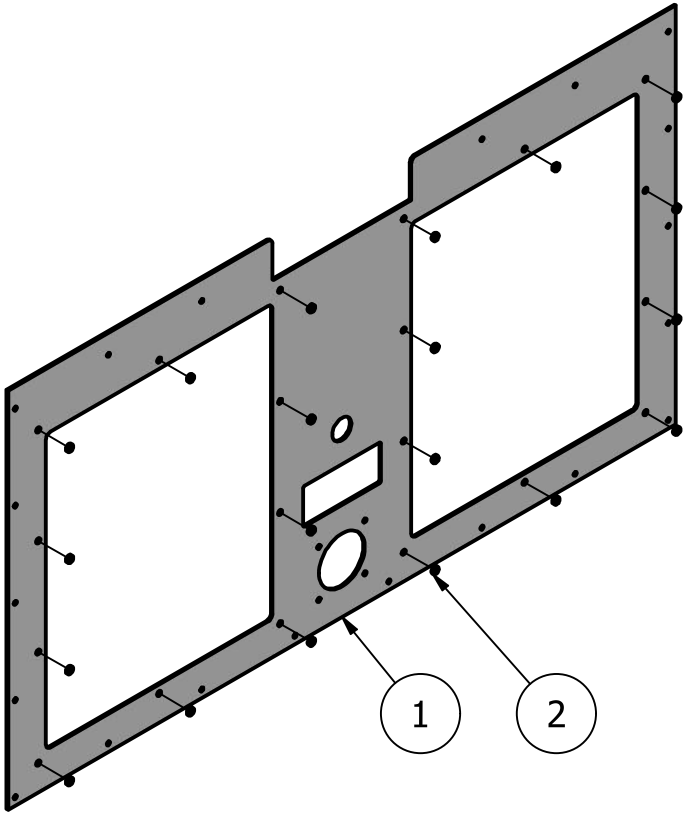
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2			07/03/2019

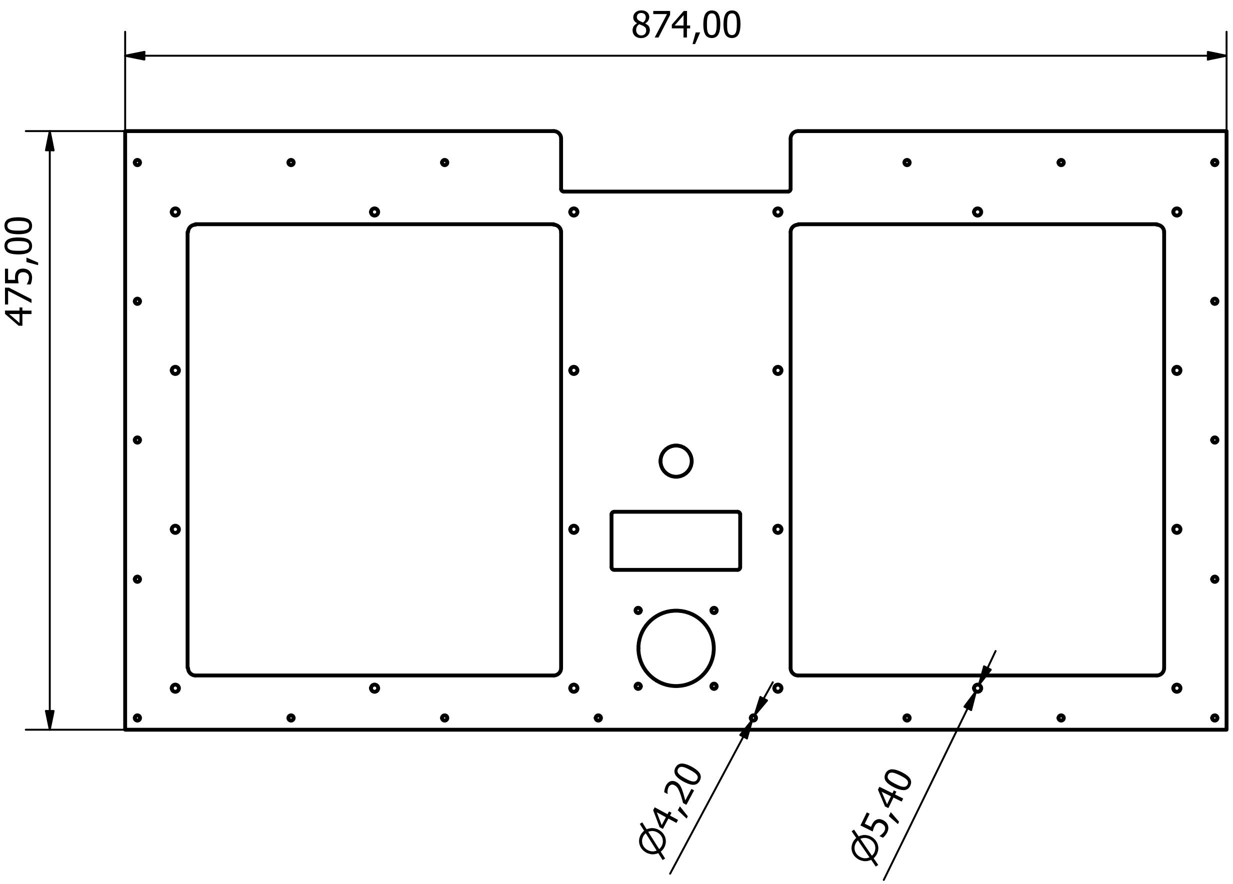
NELSON MANDELA UNIVERSITY

Spring Assembly Weldmatt

	Edition	Sheet
Swerve Drive AGV		179 / 207

HPI Inserts
Number : 20 unit
Designation : CM4-0 (SCN M4)





*HPI inserts in holes bordering cutouts only

Sheet Thickness 2,00mm

PARTS LIST		
ITEM	QTY	PART NUMBER
1	1	Front Cover
2	20	SCN M4 - CM4-0

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1		Aluminium	07/03/2019

NELSON MANDELA UNIVERSITY

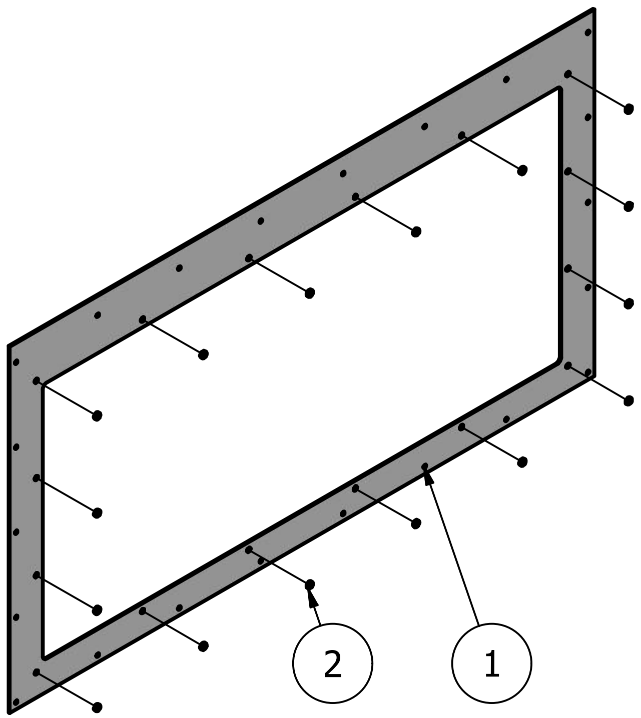
Front Cover

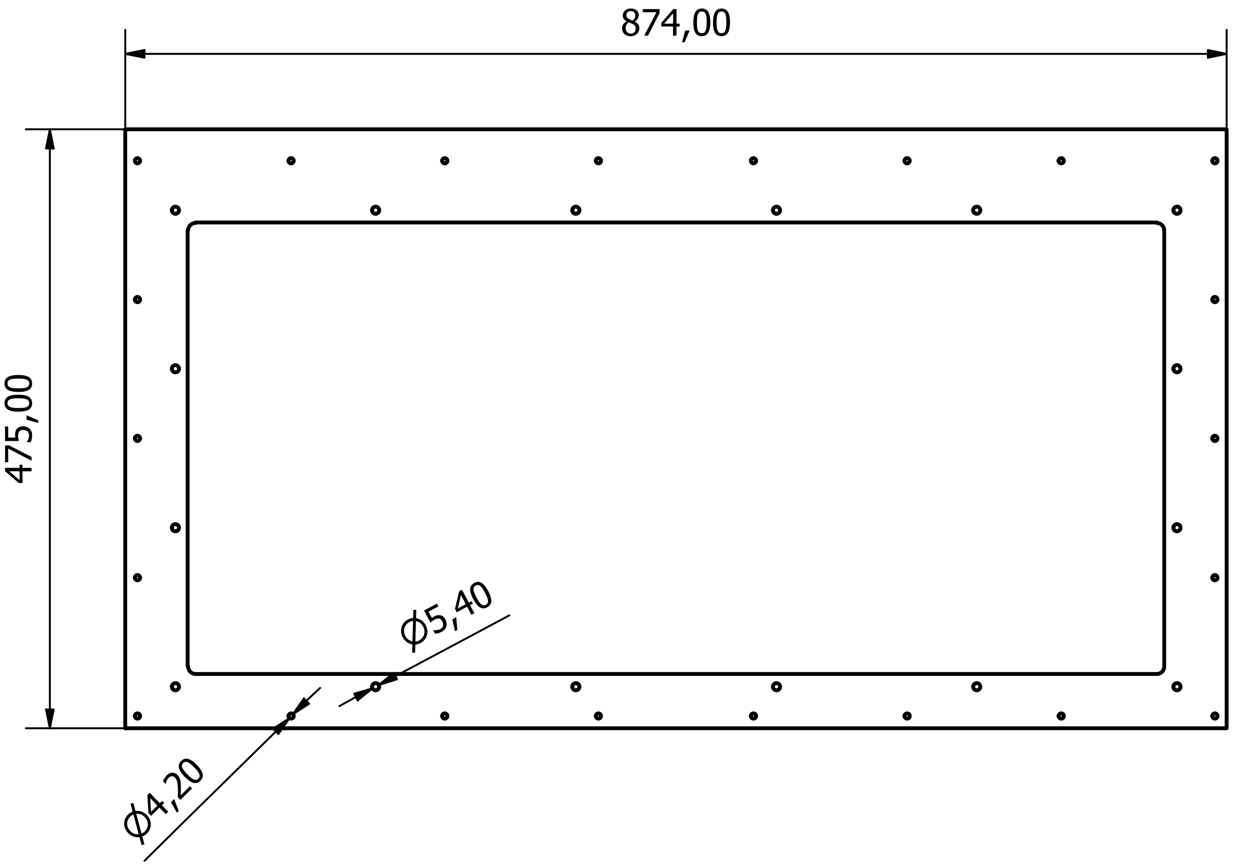
	Edition	Sheet
Swerve Drive AGV		180 / 207

HPI Inserts

Number : 16 unit

Designation : CM4-0 (SCN M4)





*HPI inserts in holes bordering cutouts only

Sheet Thickness 2,00mm

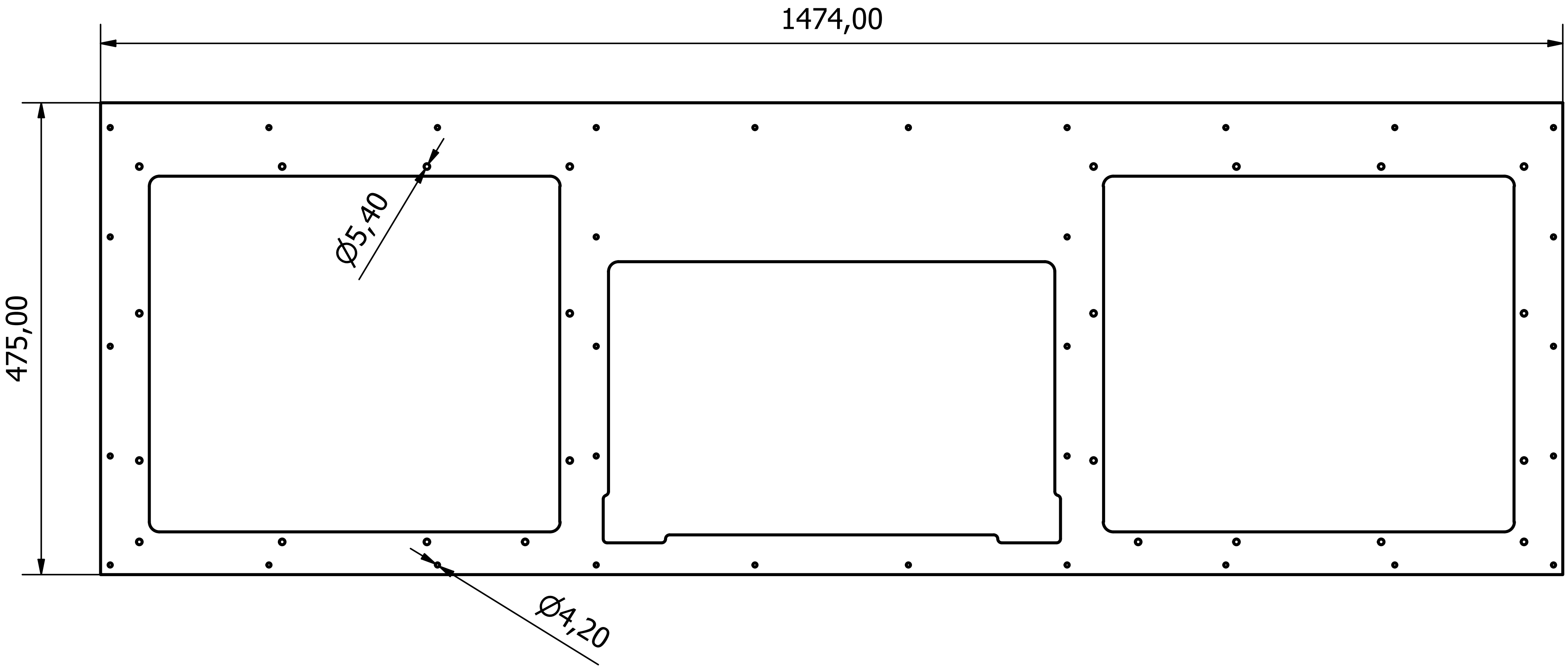
PARTS LIST		
ITEM	QTY	PART NUMBER
1	1	Rear Cover
2	16	SCN M4 - CM4-0

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1		Aluminium	07/03/2019

NELSON MANDELA UNIVERSITY

Rear Cover

	Edition	Sheet
Swerve Drive AGV		181 / 207



Sheet Thickness 2,00mm

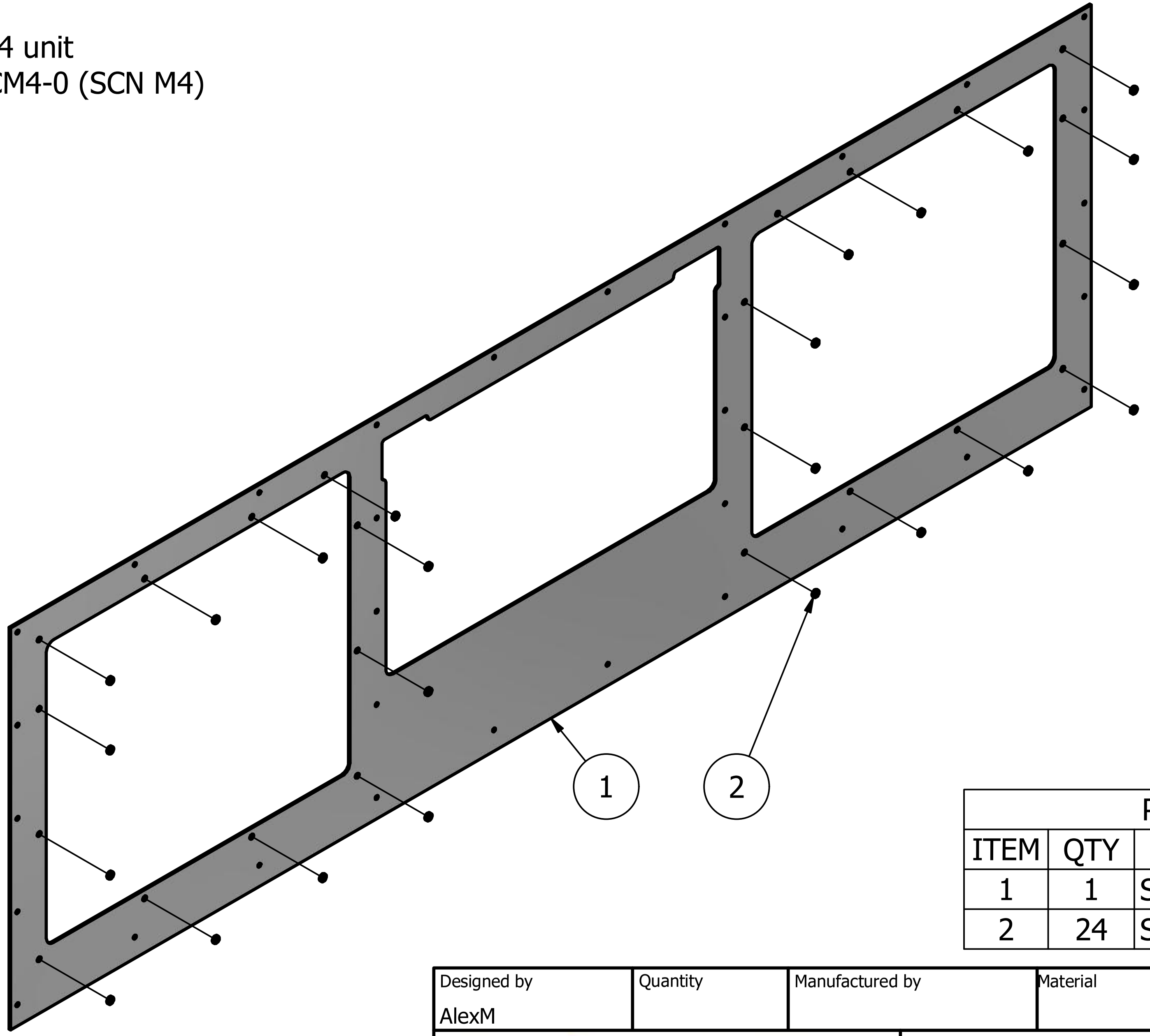
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1		Aluminium	07/03/2019

NELSON MANDELA UNIVERSITY

Side Cover A

	Edition	Sheet
Swerve Drive AGV		182 / 207

HPI Inserts
Number : 24 unit
Designation : CM4-0 (SCN M4)



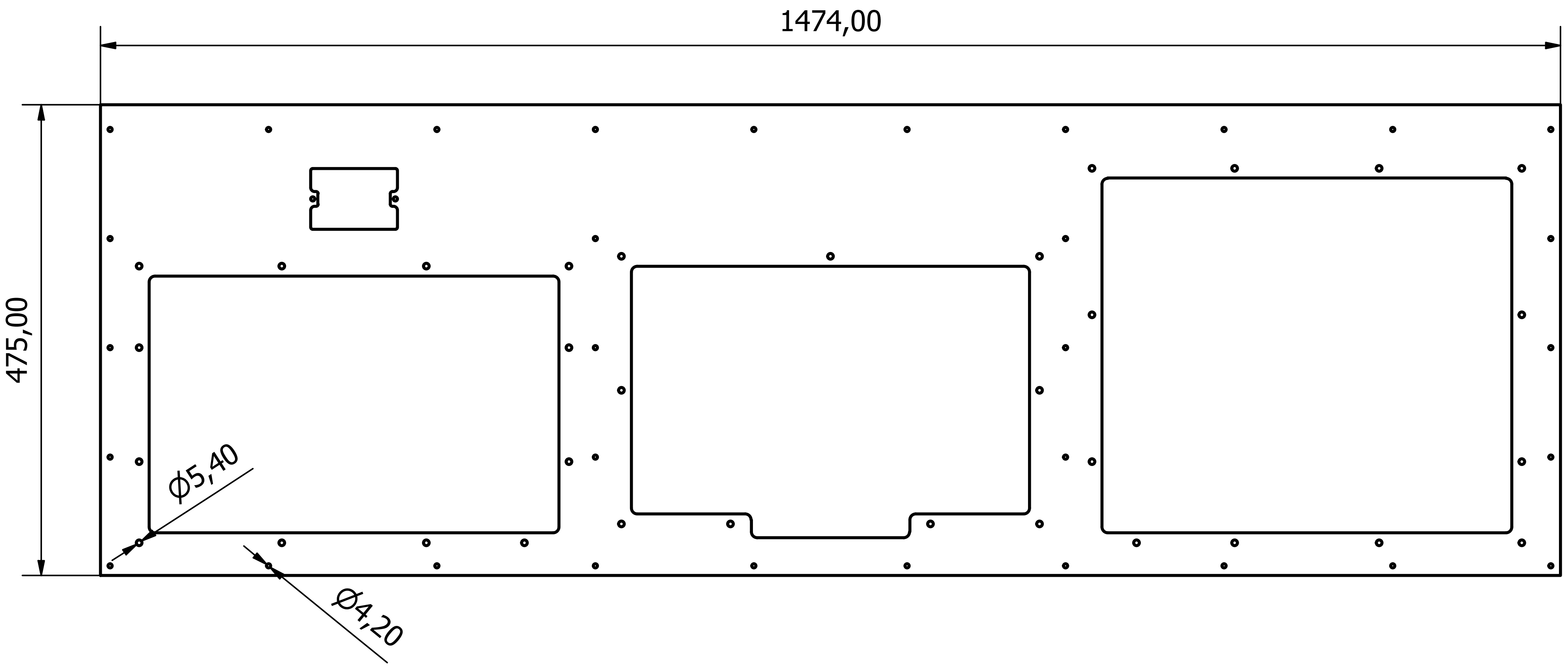
PARTS LIST		
ITEM	QTY	PART NUMBER
1	1	Side Cover A
2	24	SCN M4 - CM4-0

*HPI inserts in holes bordering cutouts only

Designed by	Quantity	Manufactured by	Material	Date
AlexM				07/03/2019

NELSON MANDELA UNIVERSITY

	Edition	Sheet
Swerve Drive AGV		183 / 207



Sheet Thickness 2,00mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1		Aluminium	07/03/2019

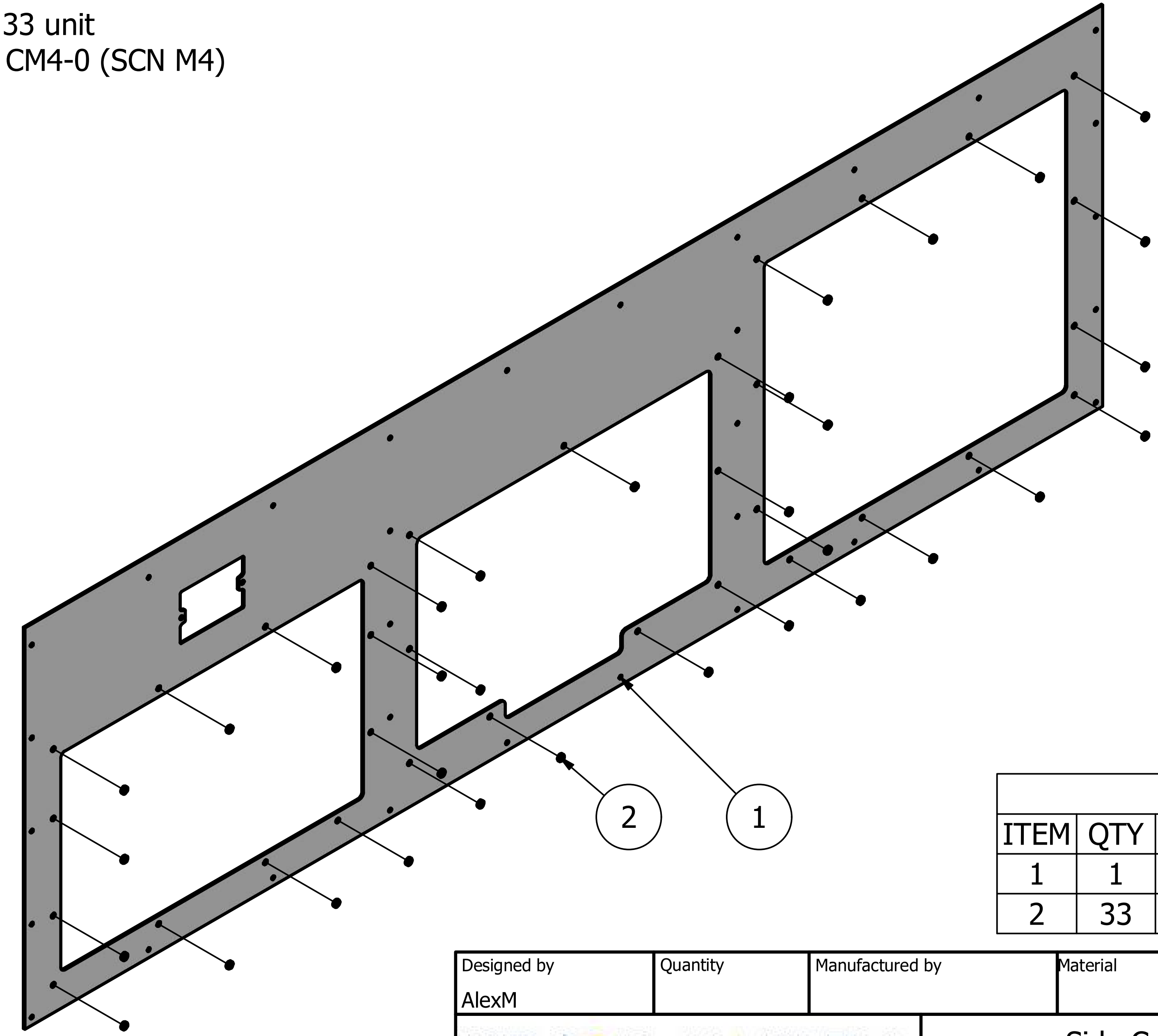
NELSON MANDELA UNIVERSITY

Side Cover B		
Swerve Drive AGV	Edition	Sheet 184 / 207

HPI Inserts

Number : 33 unit

Designation : CM4-0 (SCN M4)



PARTS LIST		
ITEM	QTY	PART NUMBER
1	1	Side Cover B
2	33	SCN M4 - CM4-0

*HPI inserts in holes bordering cutouts only

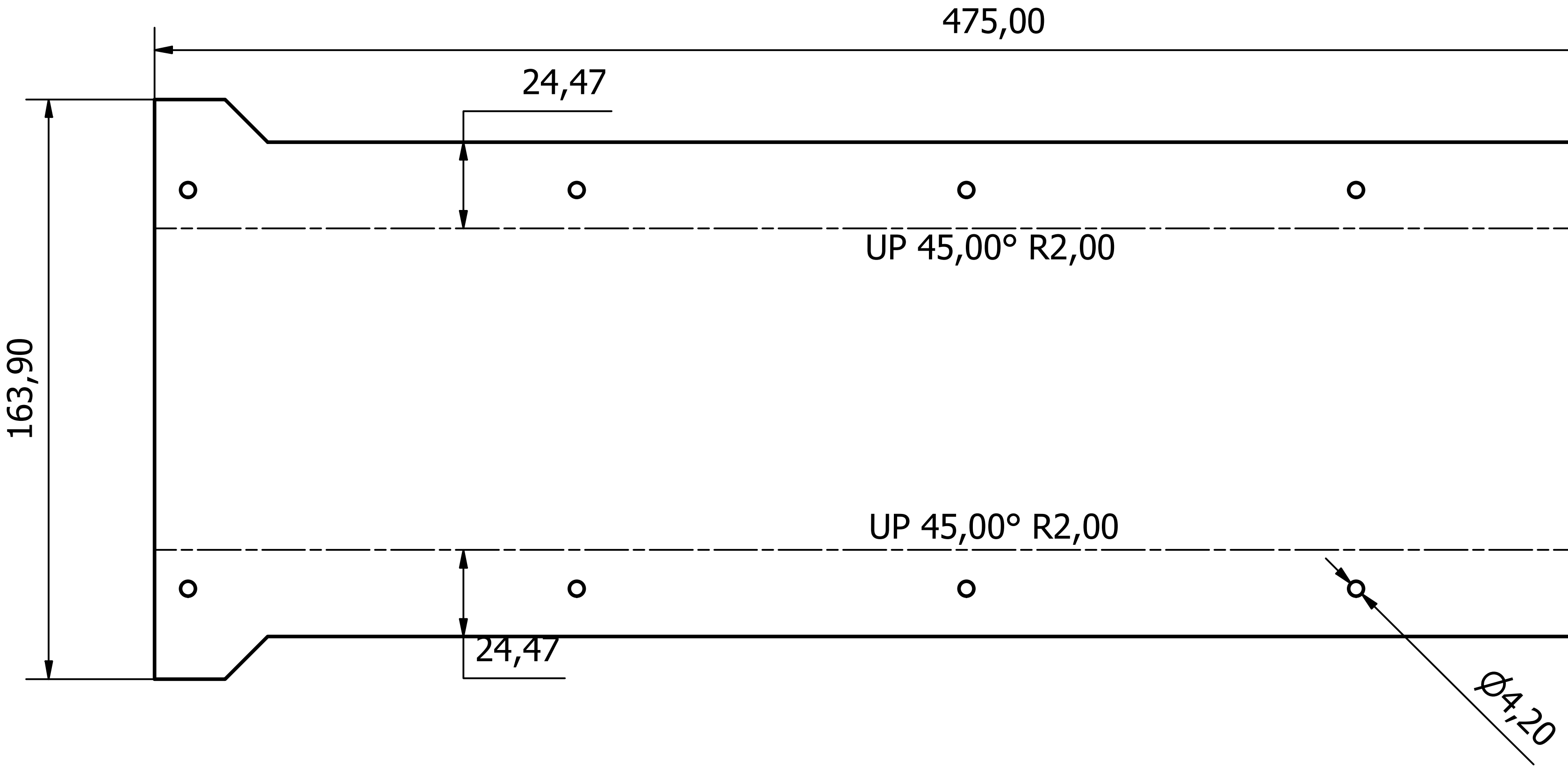
Designed by	Quantity	Manufactured by	Material	Date
AlexM				07/03/2019

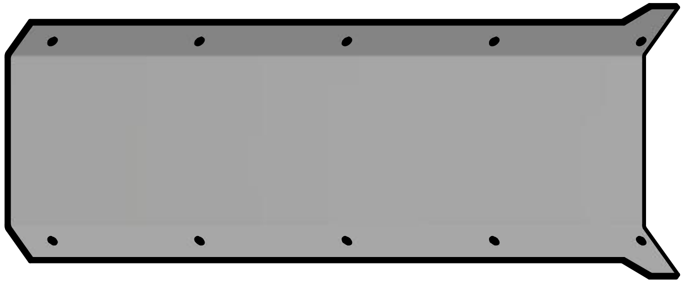
NELSON MANDELA UNIVERSITY

Side Cover B Assembly

Swerve Drive AGV

Edition	Sheet
	185 / 207





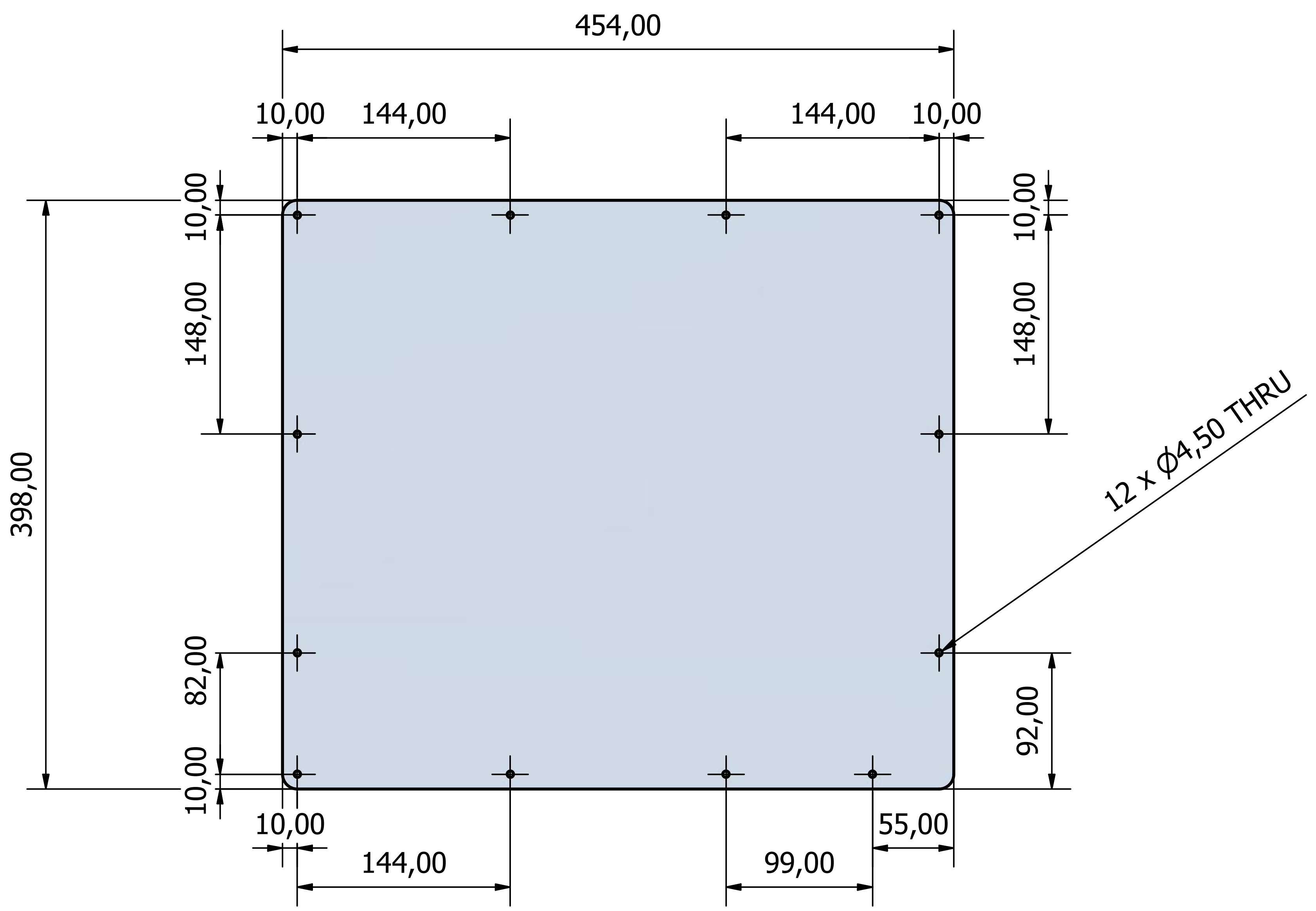
Sheet Thickness 2,00mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	4		Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Corner Cover

	Edition	Sheet
Swerve Drive AGV		186 / 207



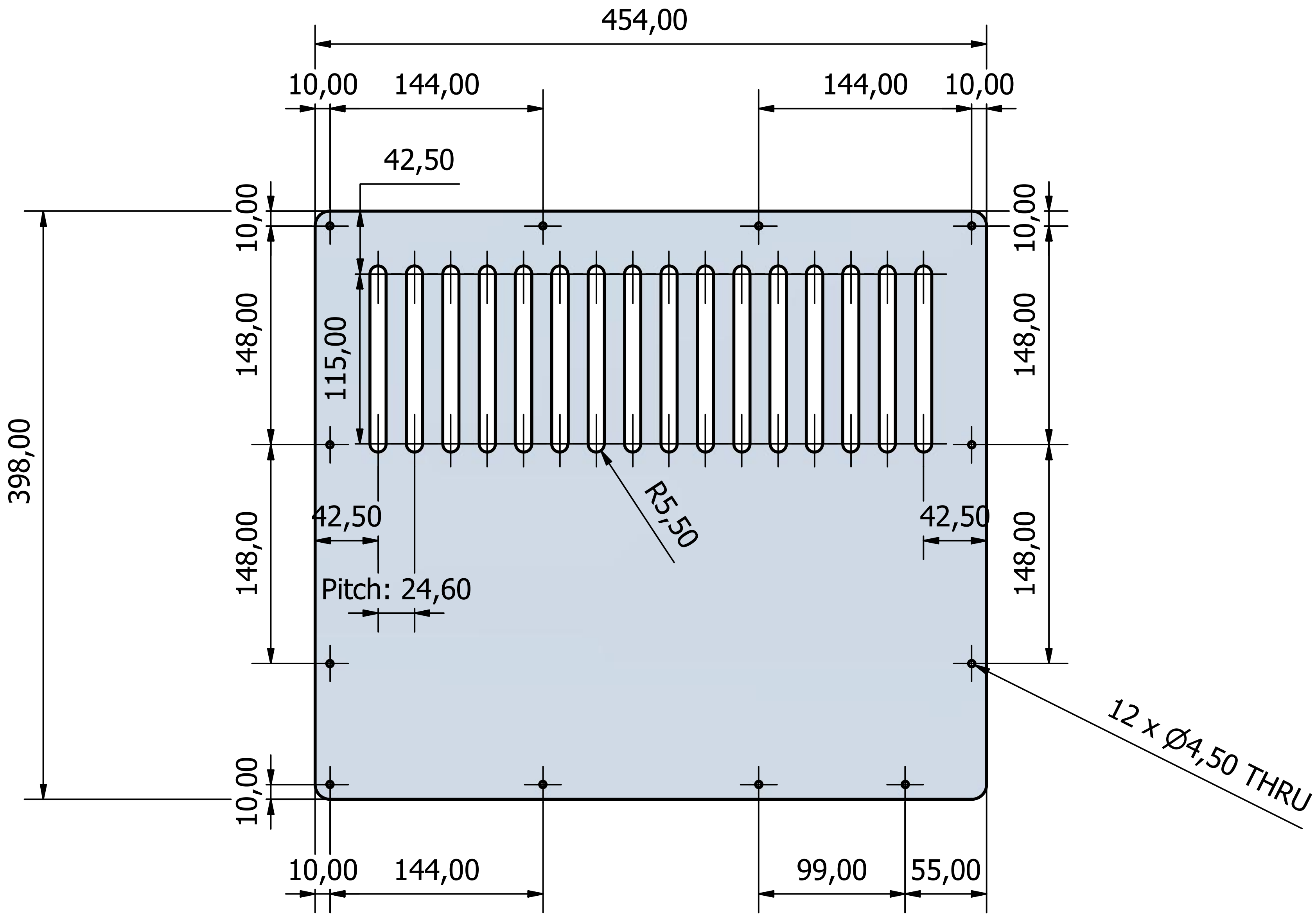
Colour: Clear

Sheet Thickness: 3,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1		Clear Perspex	07/03/2019

Access Panel Type A1

Swerve Drive AGV	Edition	Sheet
		187 / 207



Colour: Clear

Sheet Thickness: 3,00 mm

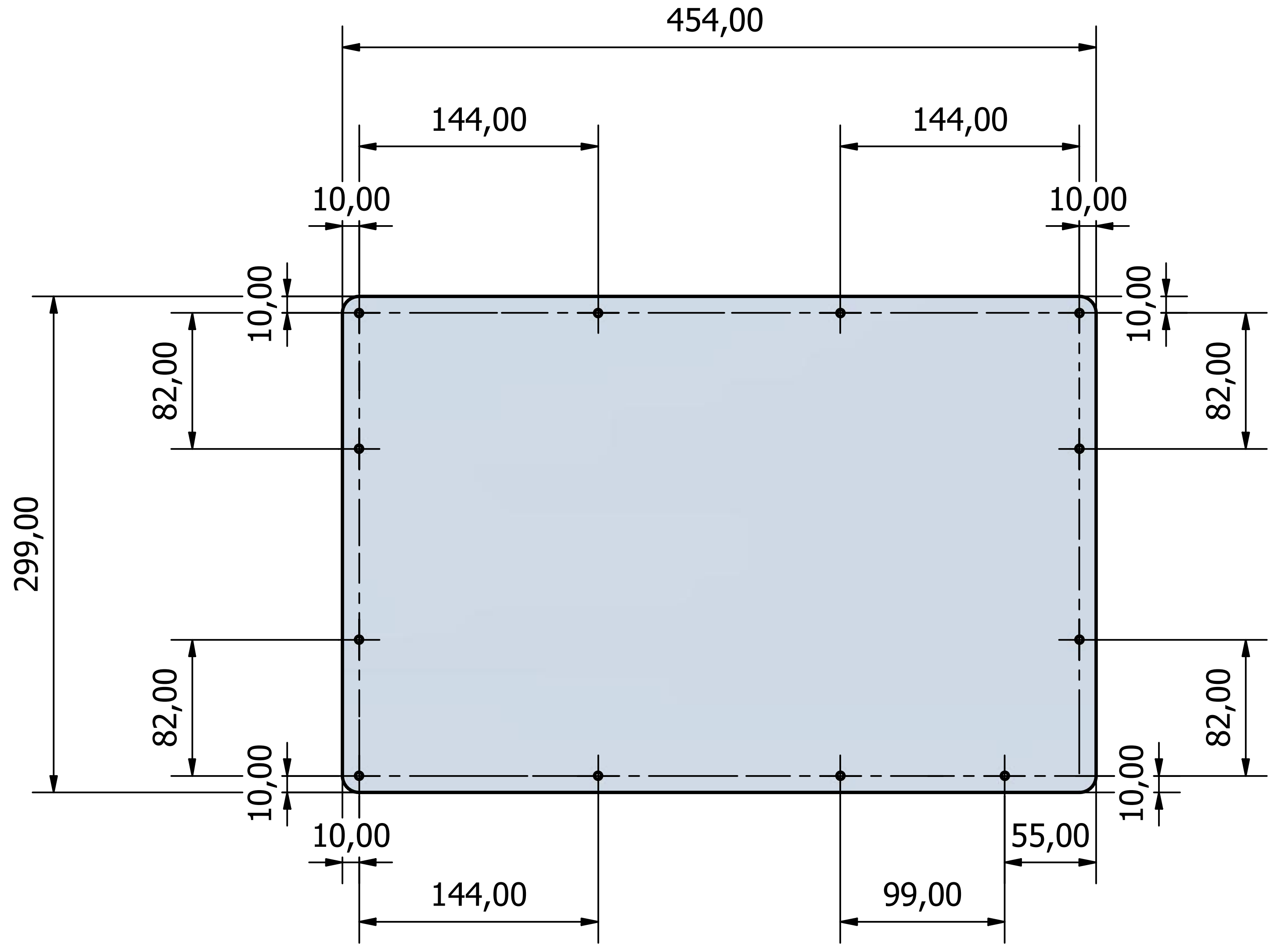
Designed by	Quantity	Manufactured by	Material	Date
AlexM	2		Clear Perspex	07/03/2019

NELSON MANDELA UNIVERSITY

Access Panel Type A2

Swerve Drive AGV

Edition	Sheet
	188 / 207



Colour: Clear

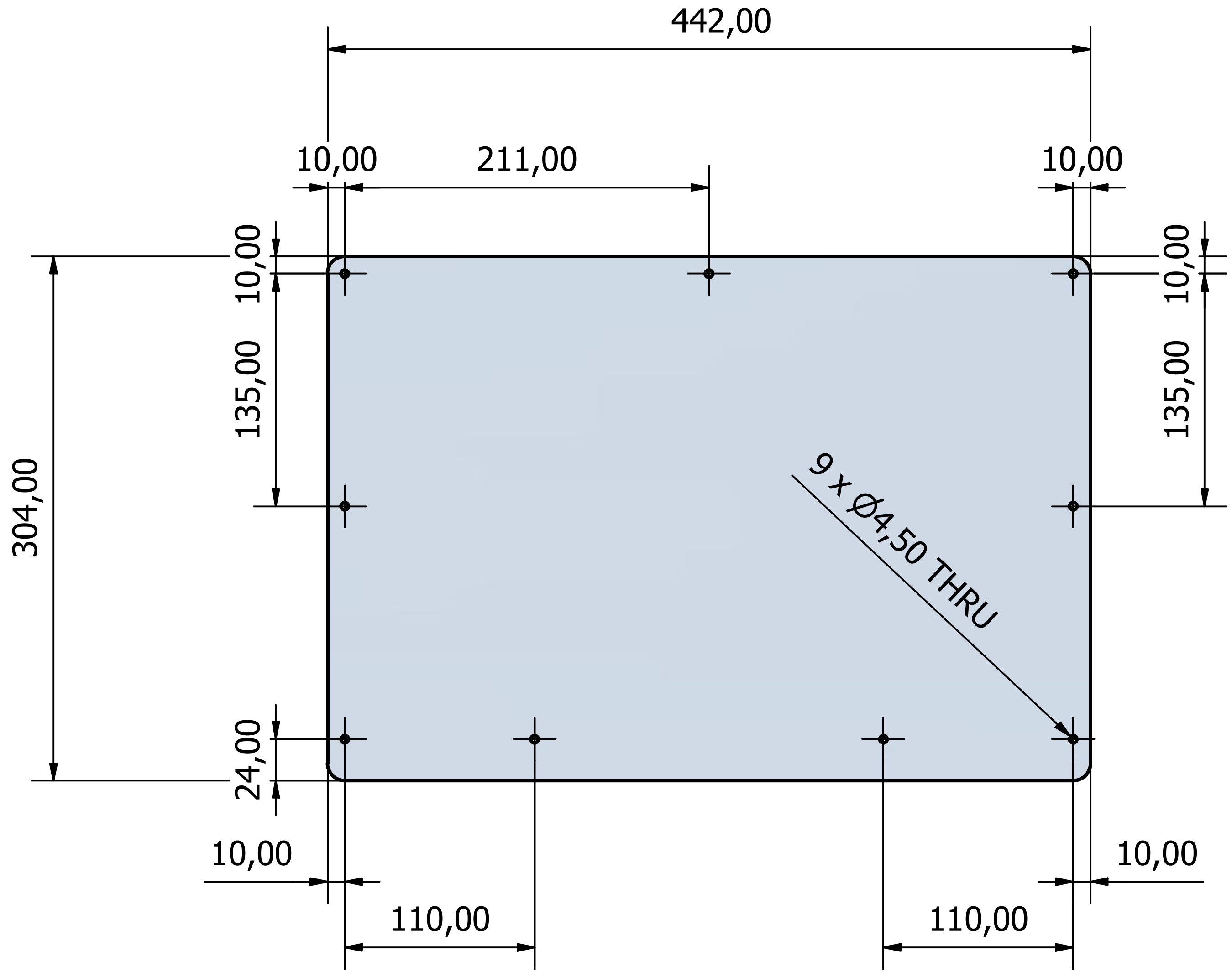
Sheet Thickness: 3,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1		Clear Perspex	07/03/2019

NELSON MANDELA UNIVERSITY

Access Panel Type A3

	Edition	Sheet
Swerve Drive AGV		189 / 207



Colour: Clear

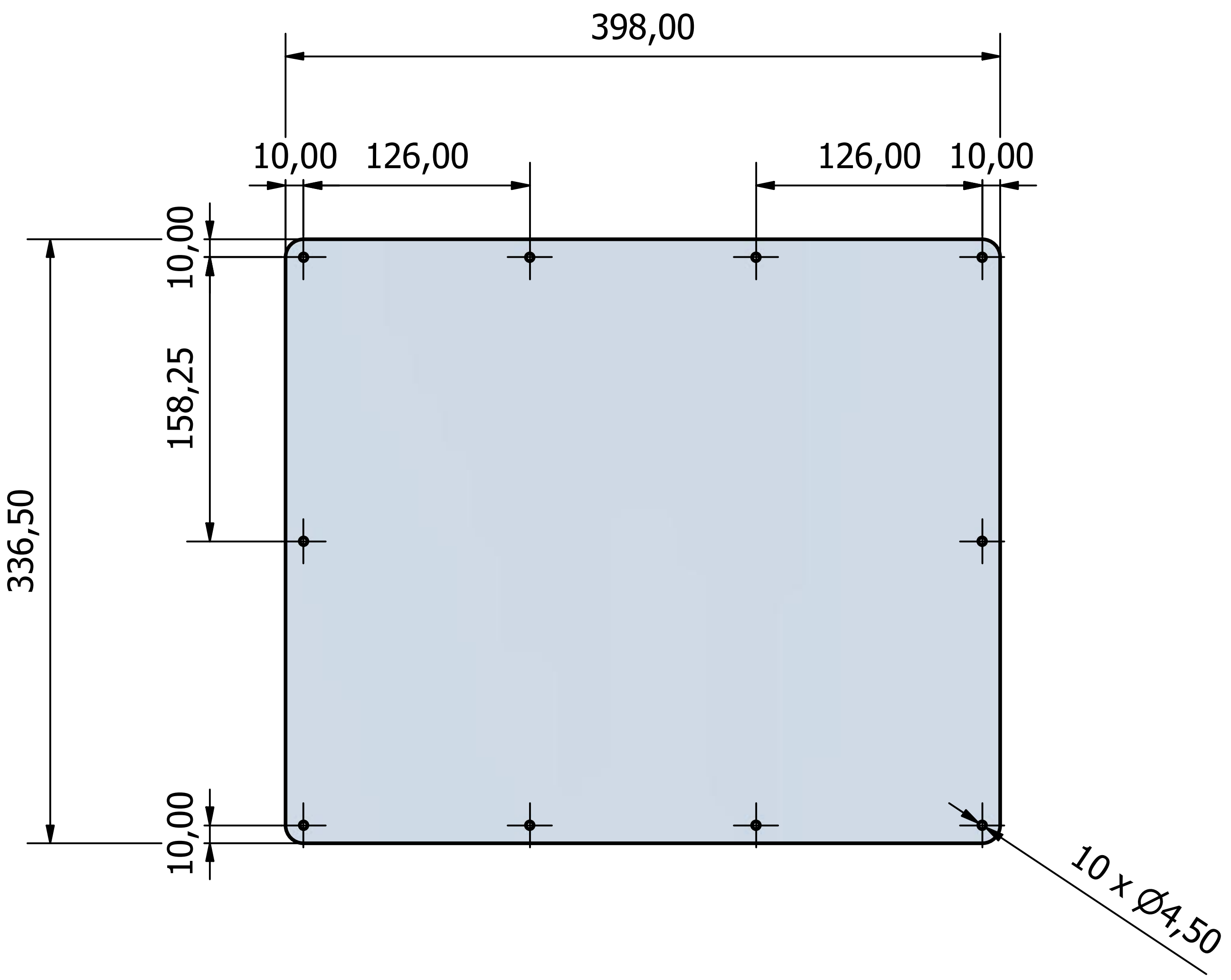
Sheet Thickness: 3,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1		Clear Perspex	07/03/2019

NELSON MANDELA UNIVERSITY

Access Panel Type B

	Edition	Sheet
Swerve Drive AGV		190 / 207



Colour: Clear

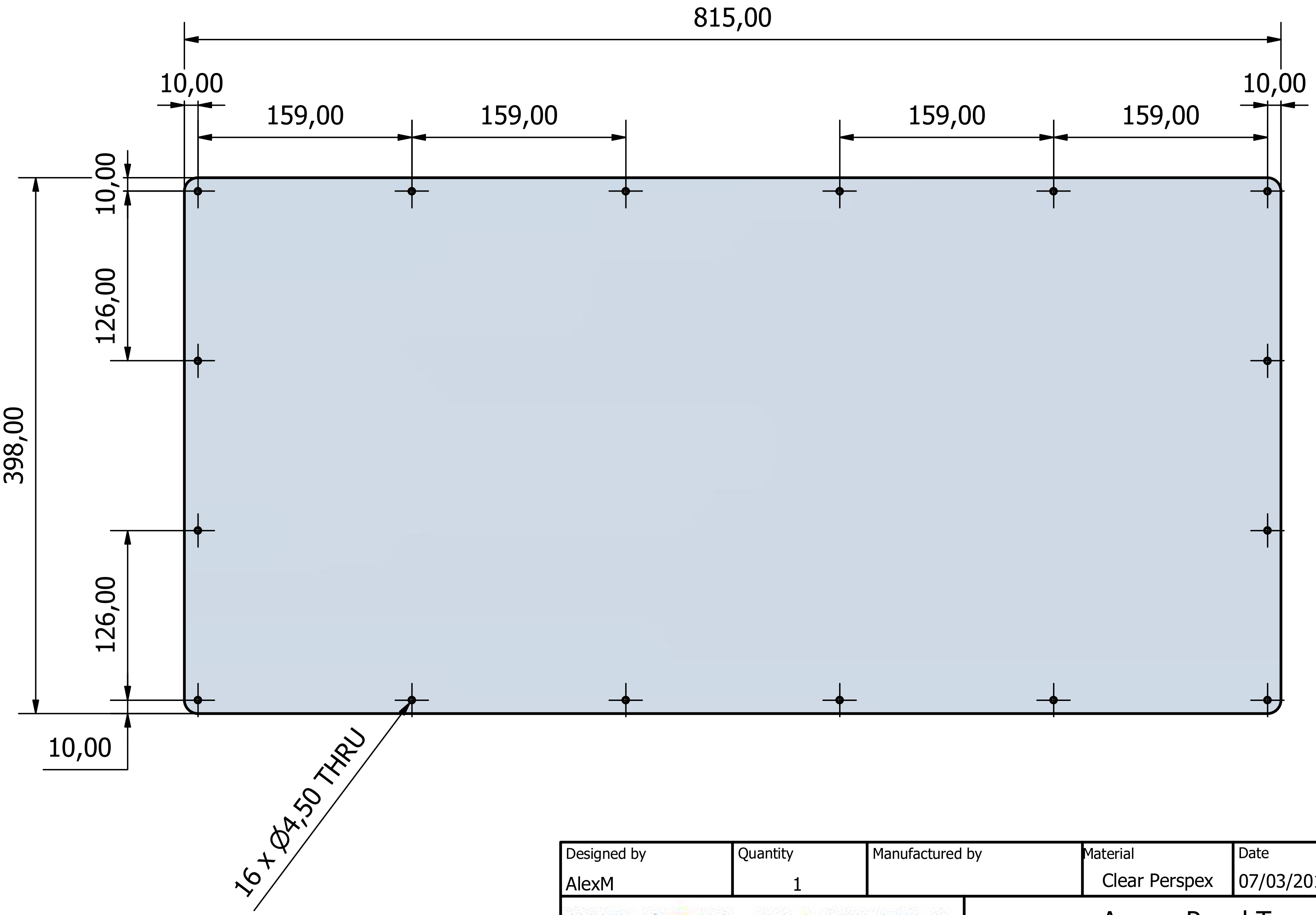
Sheet Thickness: 3,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2		Clear Perspex	07/03/2019

NELSON MANDELA UNIVERSITY

Access Panel Type C

Swerve Drive AGV	Edition	Sheet
		191 / 207



Colour: Clear

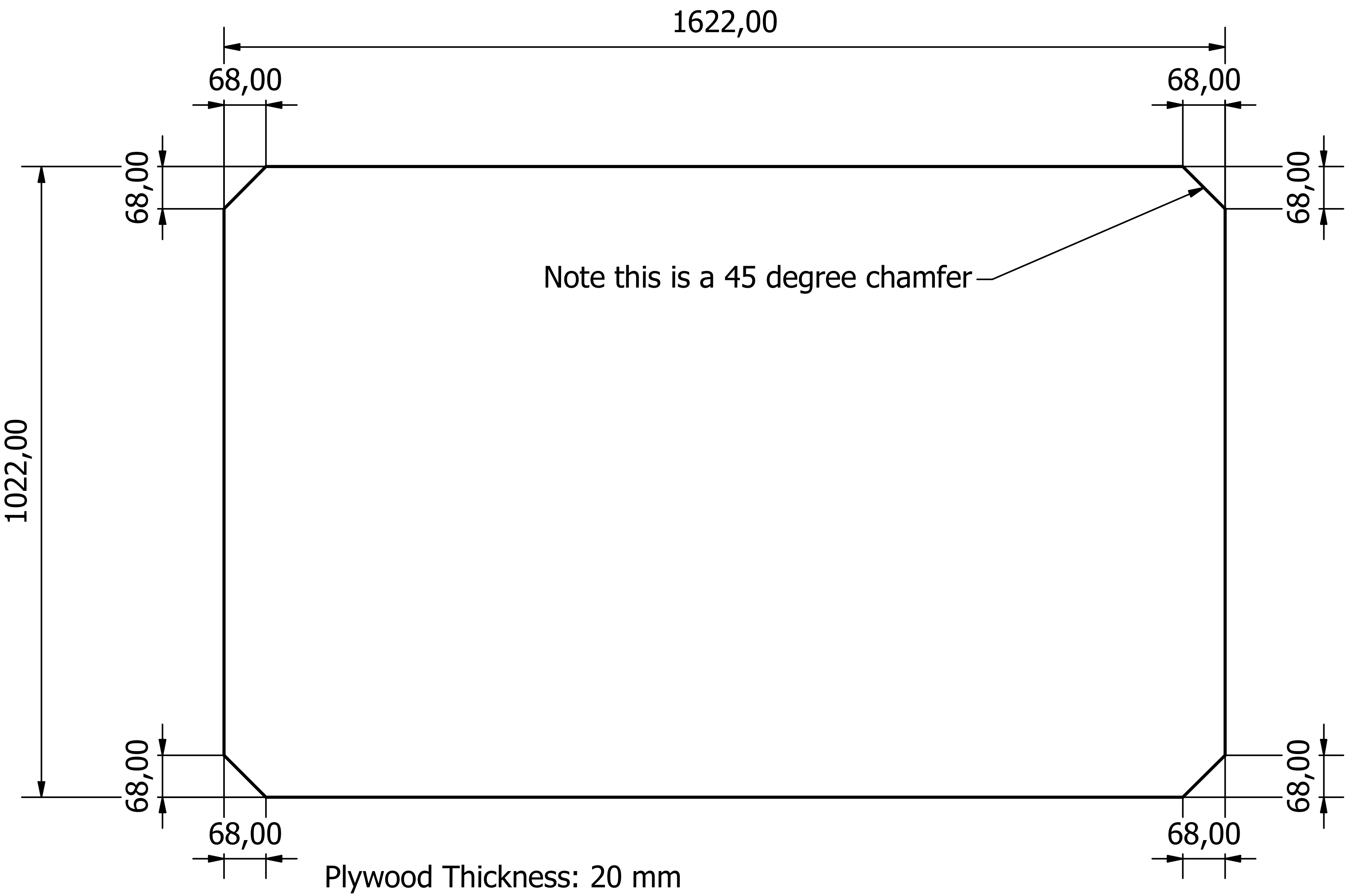
Sheet Thickness: 3,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1		Clear Perspex	07/03/2019

NELSON MANDELA UNIVERSITY

Access Panel Type D

	Edition	Sheet
Swerve Drive AGV		192 / 207



Plywood Thickness: 20 mm
(thickness deviation allowable within range 18mm to 22mm)

Tolerance conformity to 1mm required

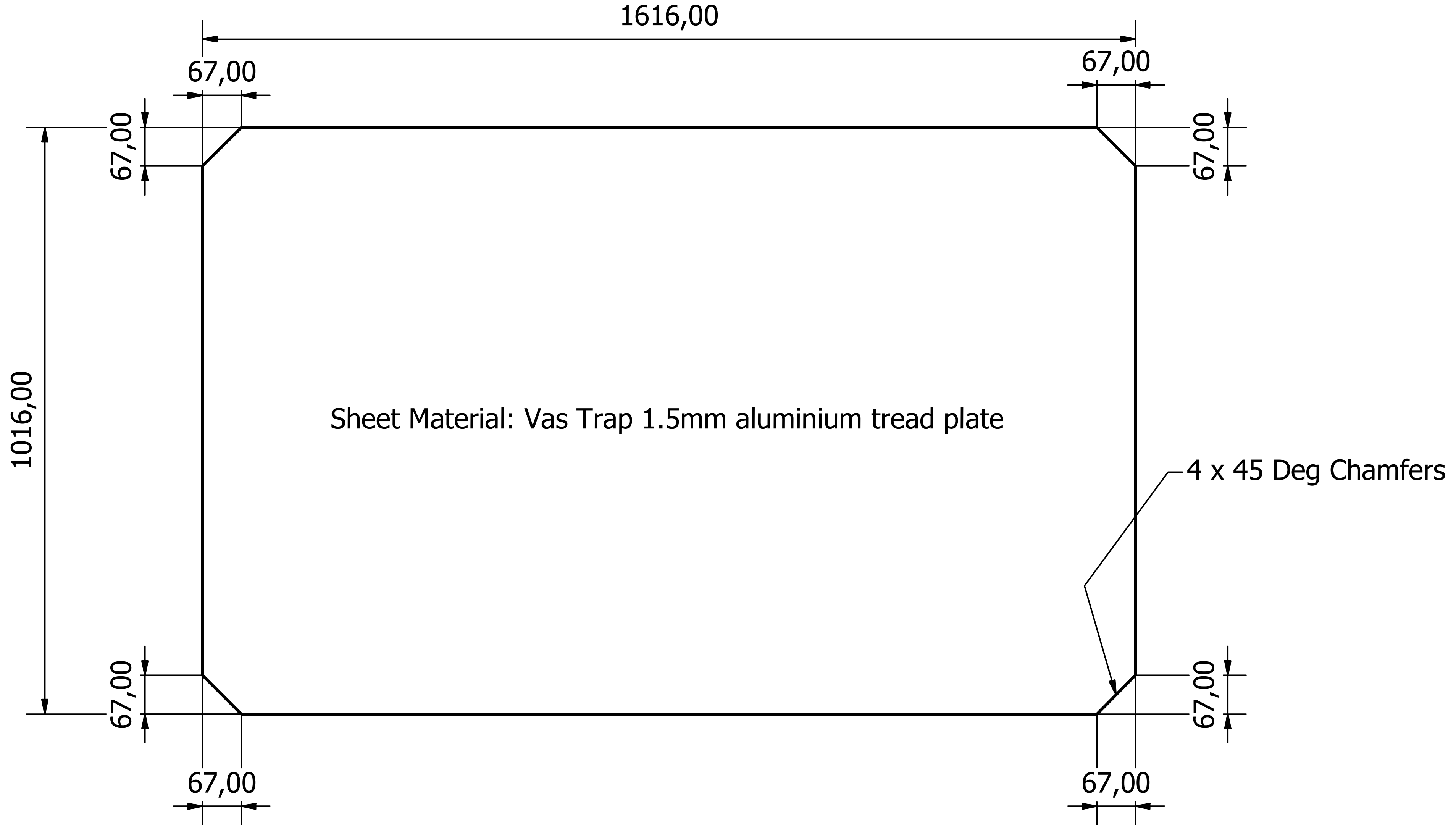
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1		Shutterply Plywood	07/03/2019

NELSON MANDELA UNIVERSITY

AGV Checker Plate Backing Board

Swerve Drive AGV

Edition	Sheet
	193 / 207



Sheet Material: Vas Trap 1.5mm aluminium tread plate

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1		Aluminium 1050 & 4017	07/03/2019

NELSON MANDELA UNIVERSITY

AGV Lid Chequer Plate

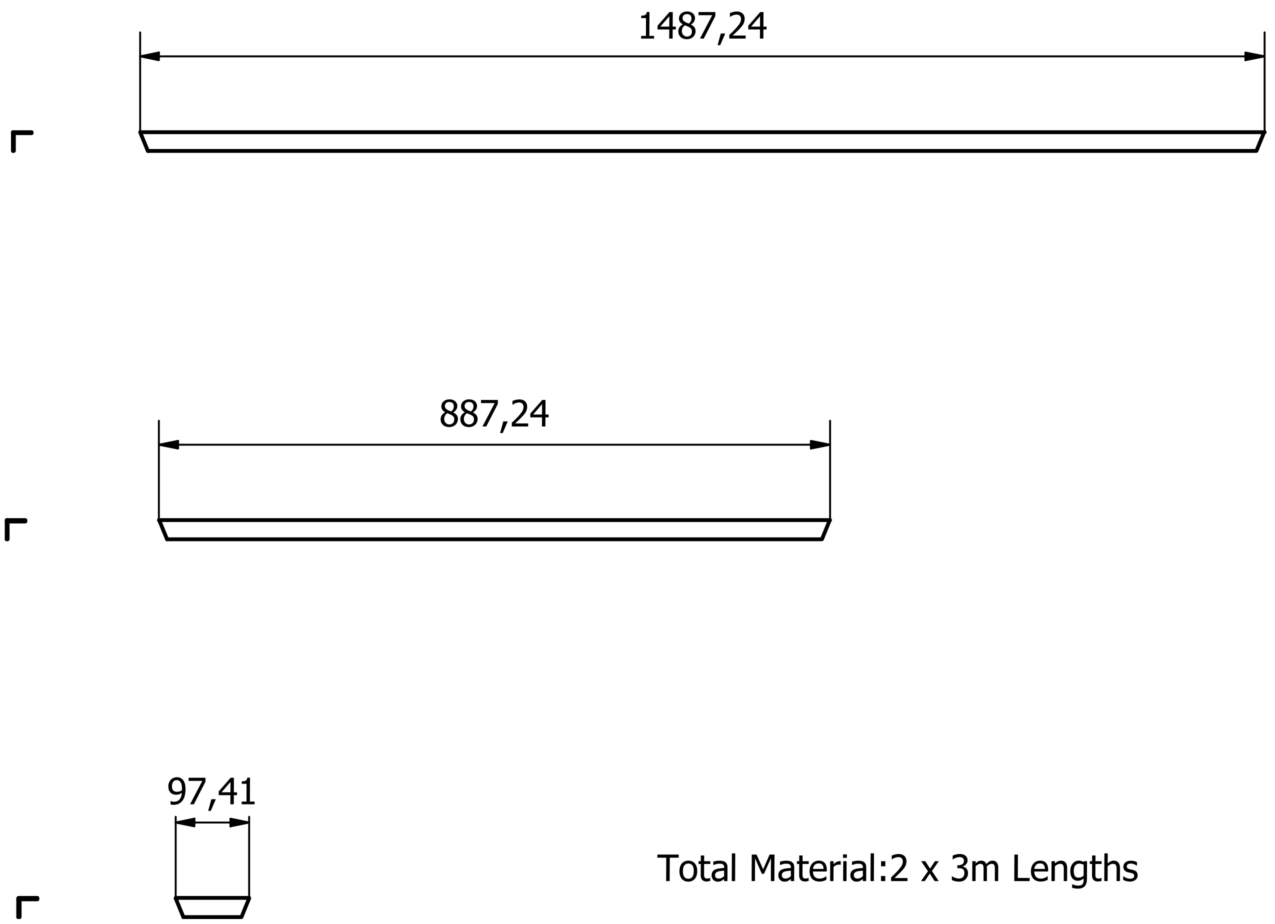
Swerve Drive AGV

Edition	Sheet
	194 / 207

WINSPECO L25x25x1.5 000003
WINSPECO L25x25x1.5 000004

WINSPECO L25x25x1.5 000001
WINSPECO L25x25x1.5 000002

WINSPECO L25x25x1.5 000005
WINSPECO L25x25x1.5 000006
WINSPECO L25x25x1.5 000007
WINSPECO L25x25x1.5 000008

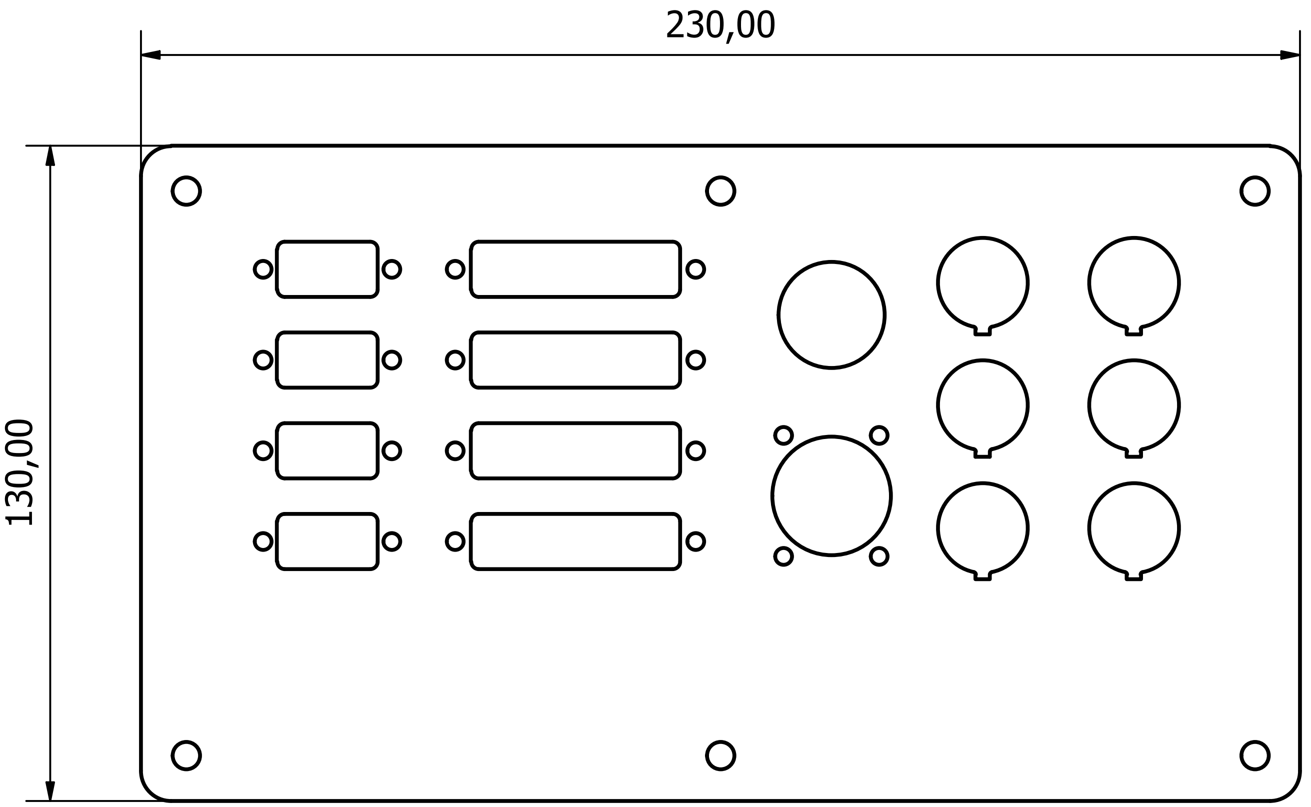


Total Material:2 x 3m Lengths

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1		Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

AGV Lid WINSPECO Edging		
Swerve Drive AGV	Edition	Sheet 195 / 207



Sheet Thickness 1,6mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1		Aluminum 6061	07/03/2019

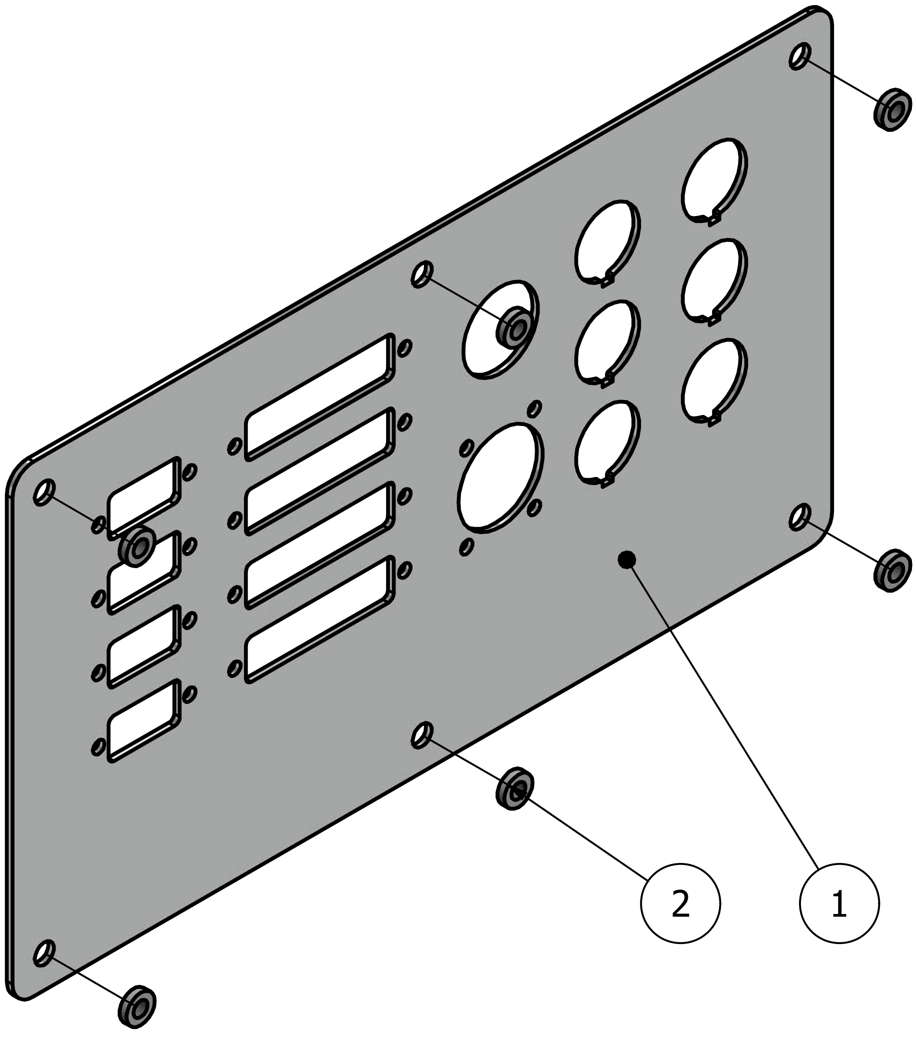
NELSON MANDELA UNIVERSITY

Electrical Connection Plate Type A

Swerve Drive AGV

Edition	Sheet
	196 / 207

HPI Inserts
Number : 6 unit
Designation : CM4-0 (SCN M4)



PARTS LIST		
ITEM	QTY	PART NUMBER
1	1	Electric Connection Plate Type A
2	6	SCN M4 - CM4-0

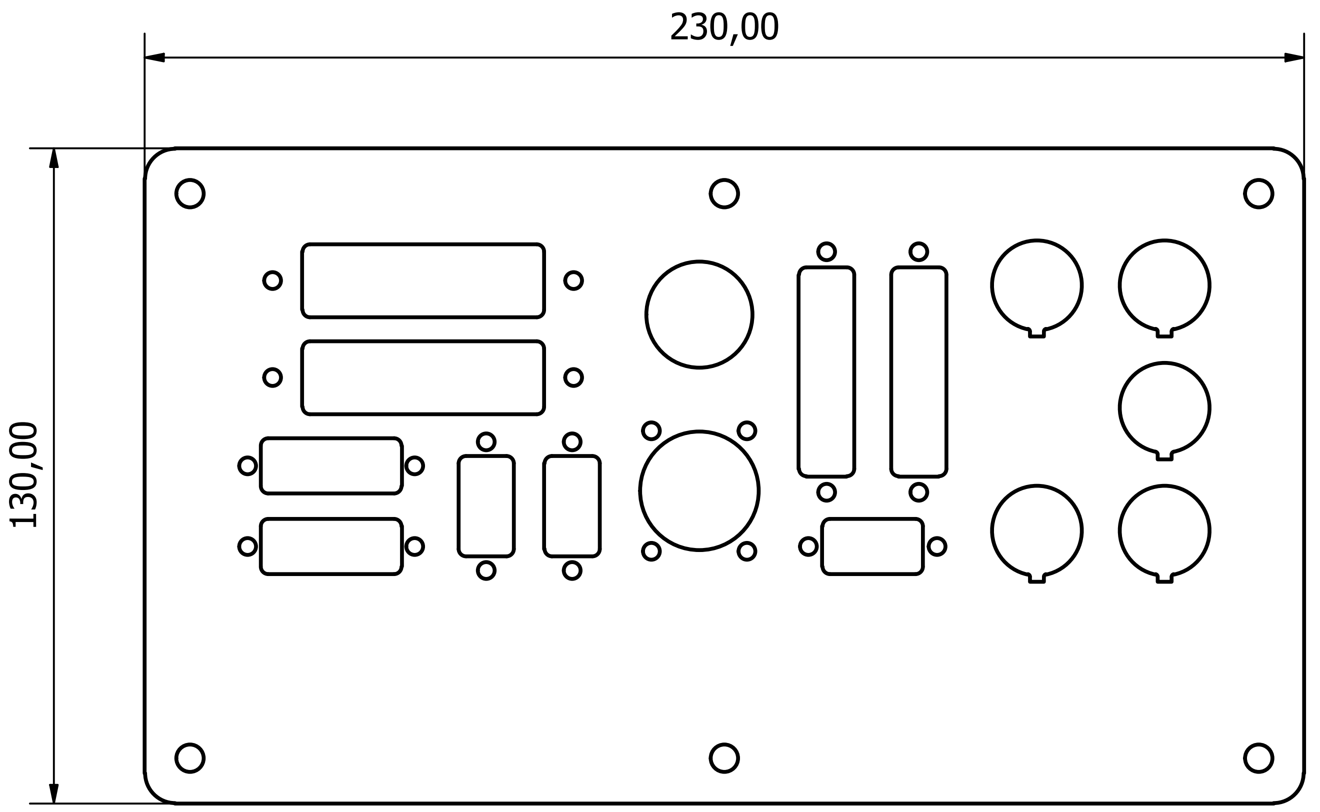
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1			07/03/2019

NELSON MANDELA UNIVERSITY

Electrical Connection Plate Type A

Swerve Drive AGV

Edition	Sheet
	197 / 207



Sheet Thickness 1,6mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1		Aluminum 6061	07/03/2019

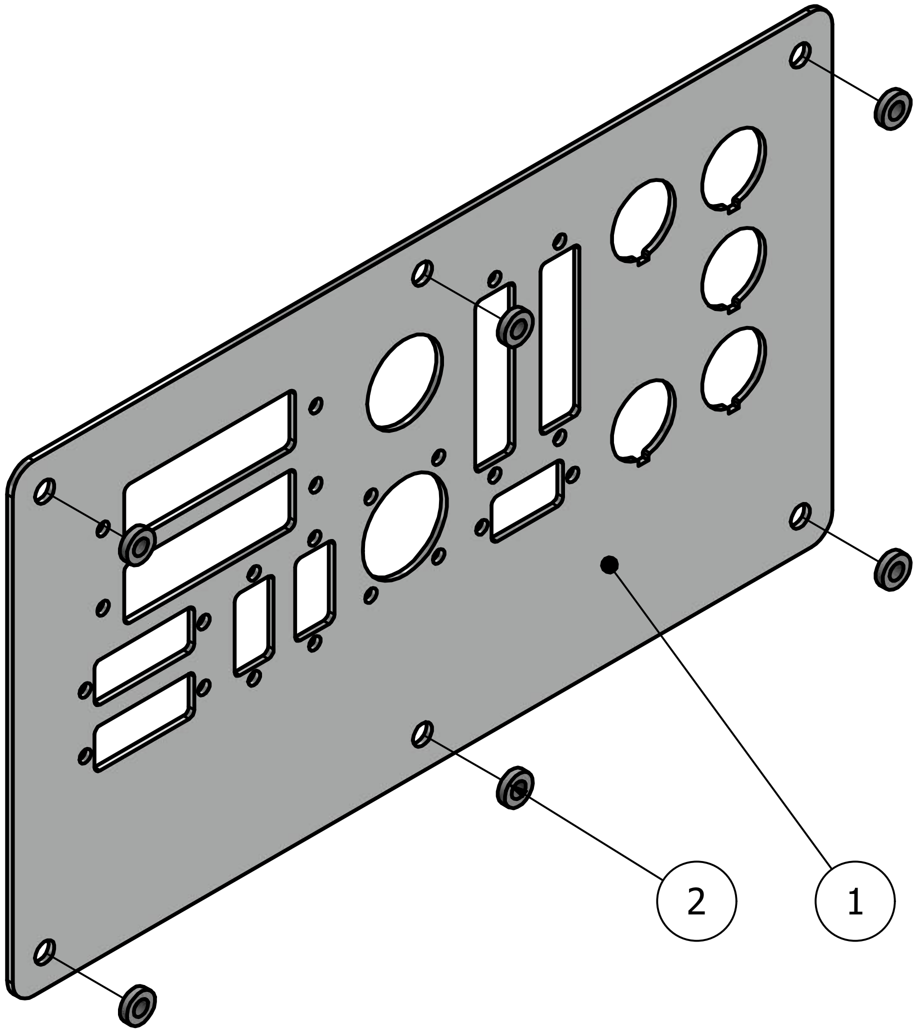
NELSON MANDELA UNIVERSITY

Electrical Connection Plate Type B

Swerve Drive AGV	Edition	Sheet
		198 / 207

HPI Inserts

Number : 6 unit
Designation : CM4-0 (SCN M4)



PARTS LIST		
ITEM	QTY	PART NUMBER
1	1	Electric Connection Plate Type B
2	6	SCN M4 - CM4-0

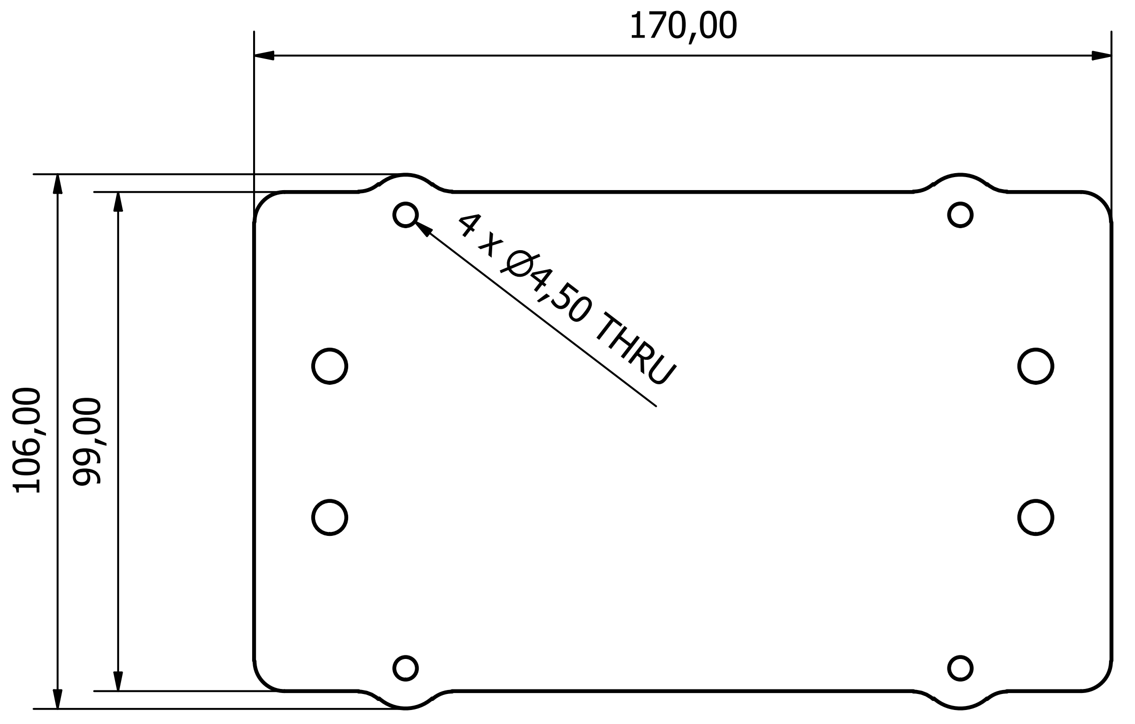
Designed by	Quantity	Manufactured by	Material	Date
AlexM	1			07/03/2019

NELSON MANDELA UNIVERSITY

Electrical Connection Plate Type B

Swerve Drive AGV

Edition	Sheet
	199 / 207



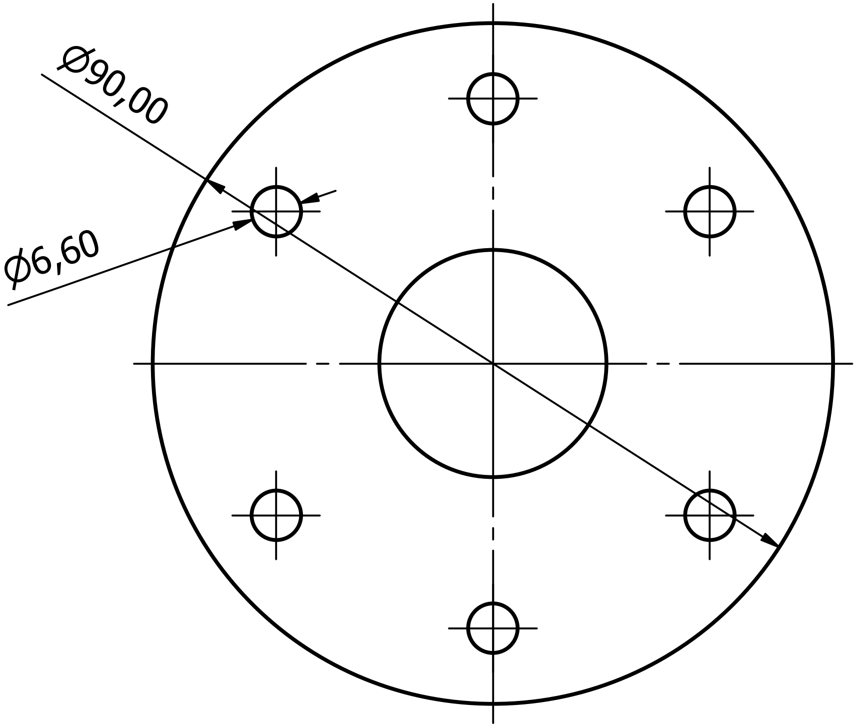
Sheet Thickness 3,0mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	1		Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

WLAN Holding Plate

Swerve Drive AGV	Edition	Sheet
		200 / 207



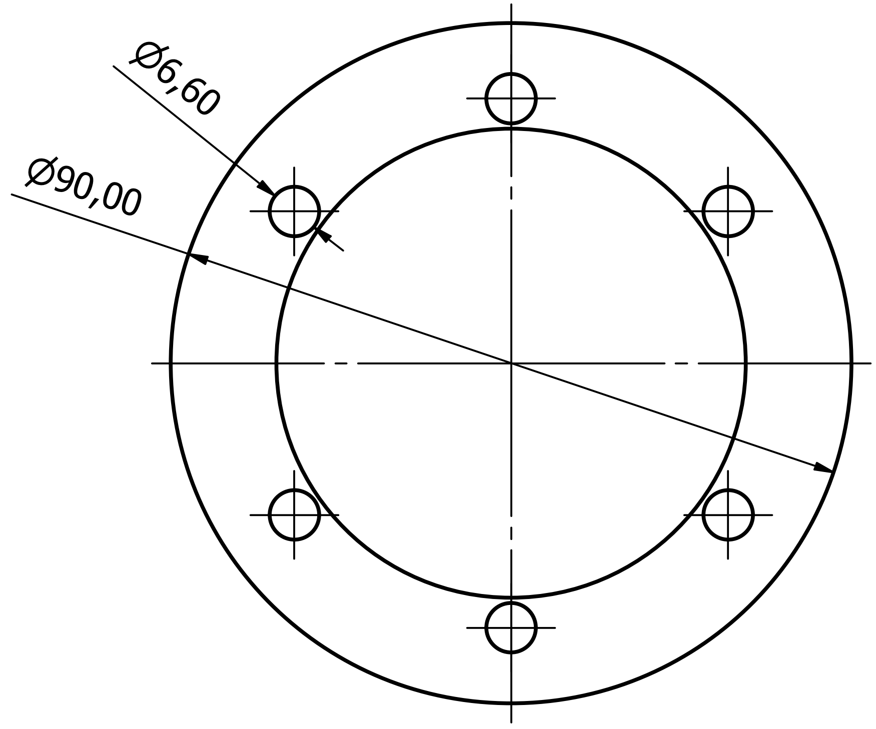
Sheet Metal Thickness: 4,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2		Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

GT2 Pulley Attachement Plate

	Edition	Sheet
Swerve Drive AGV		201 / 207



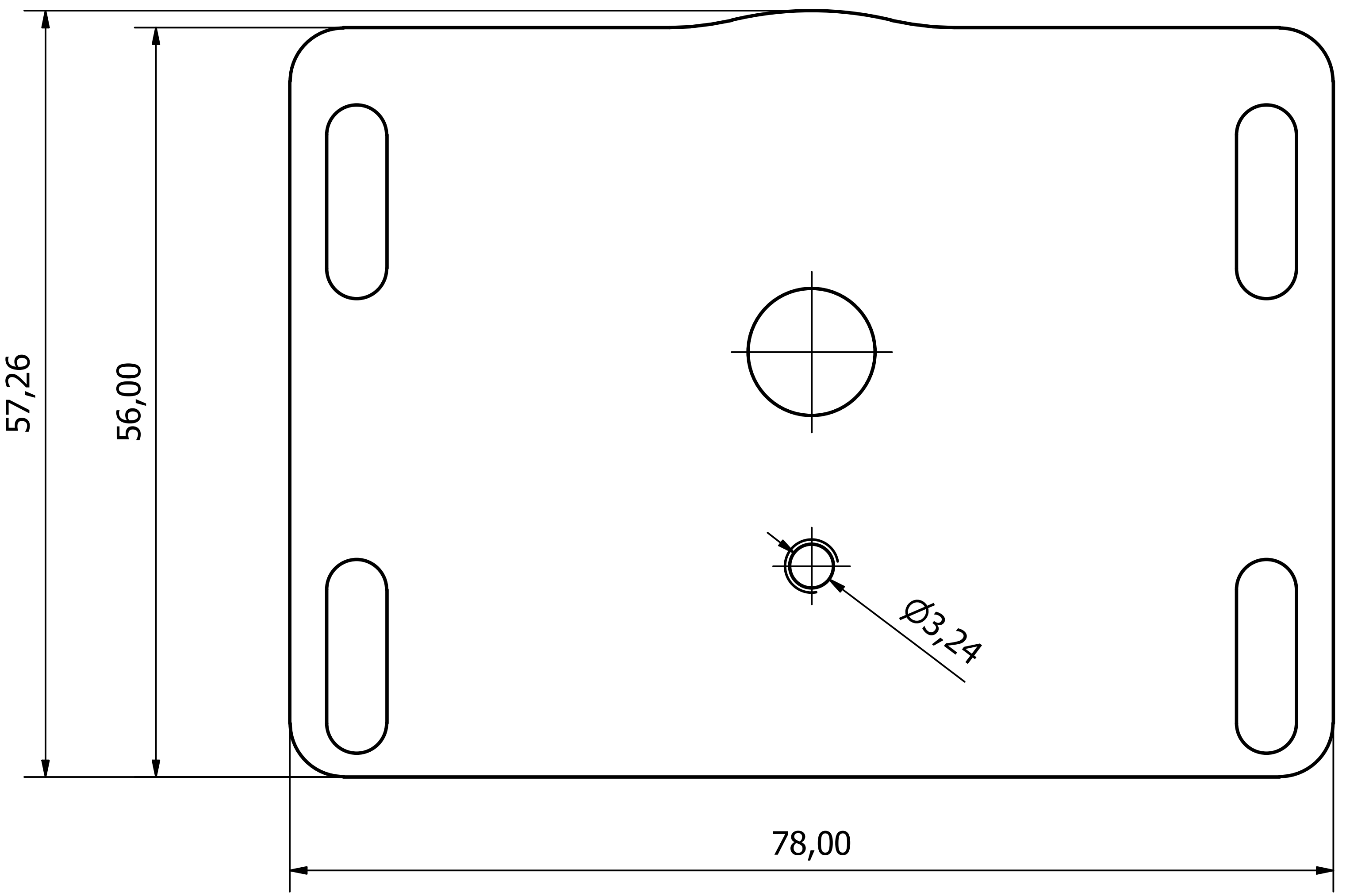
Sheet Metal Thickness: 4,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2		Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Gearbox Side Ring Spacer

	Edition	Sheet
Swerve Drive AGV		202 / 207



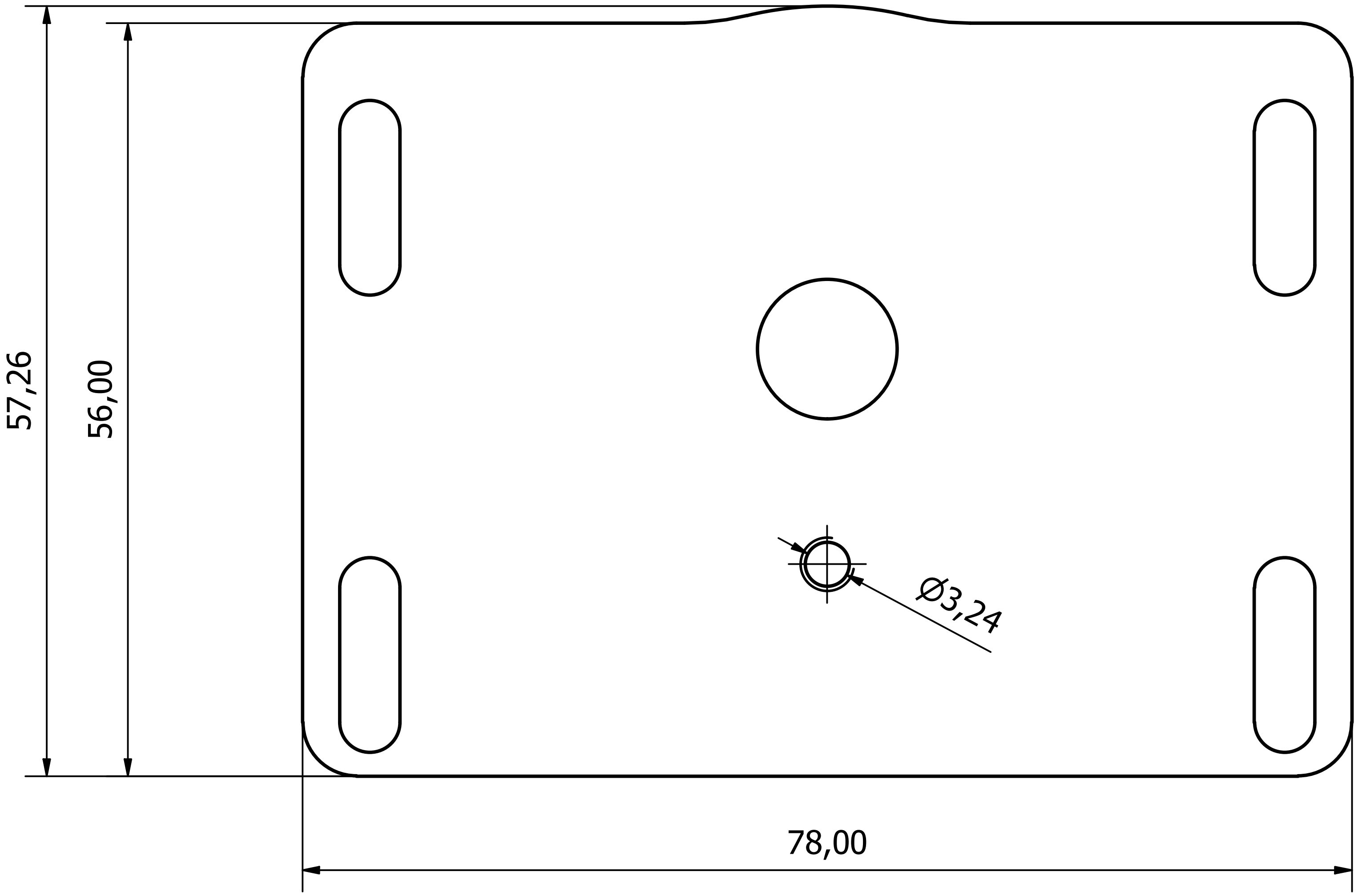
Sheet Metal Thickness: 3,00 mm

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2		Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

Encoder Plate Type A

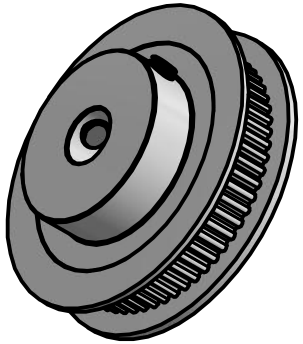
	Edition	Sheet
Swerve Drive AGV		203 / 207



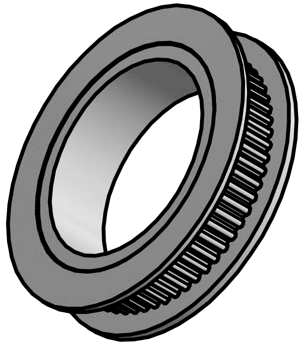
Sheet Metal Thickness: 3,00 mm

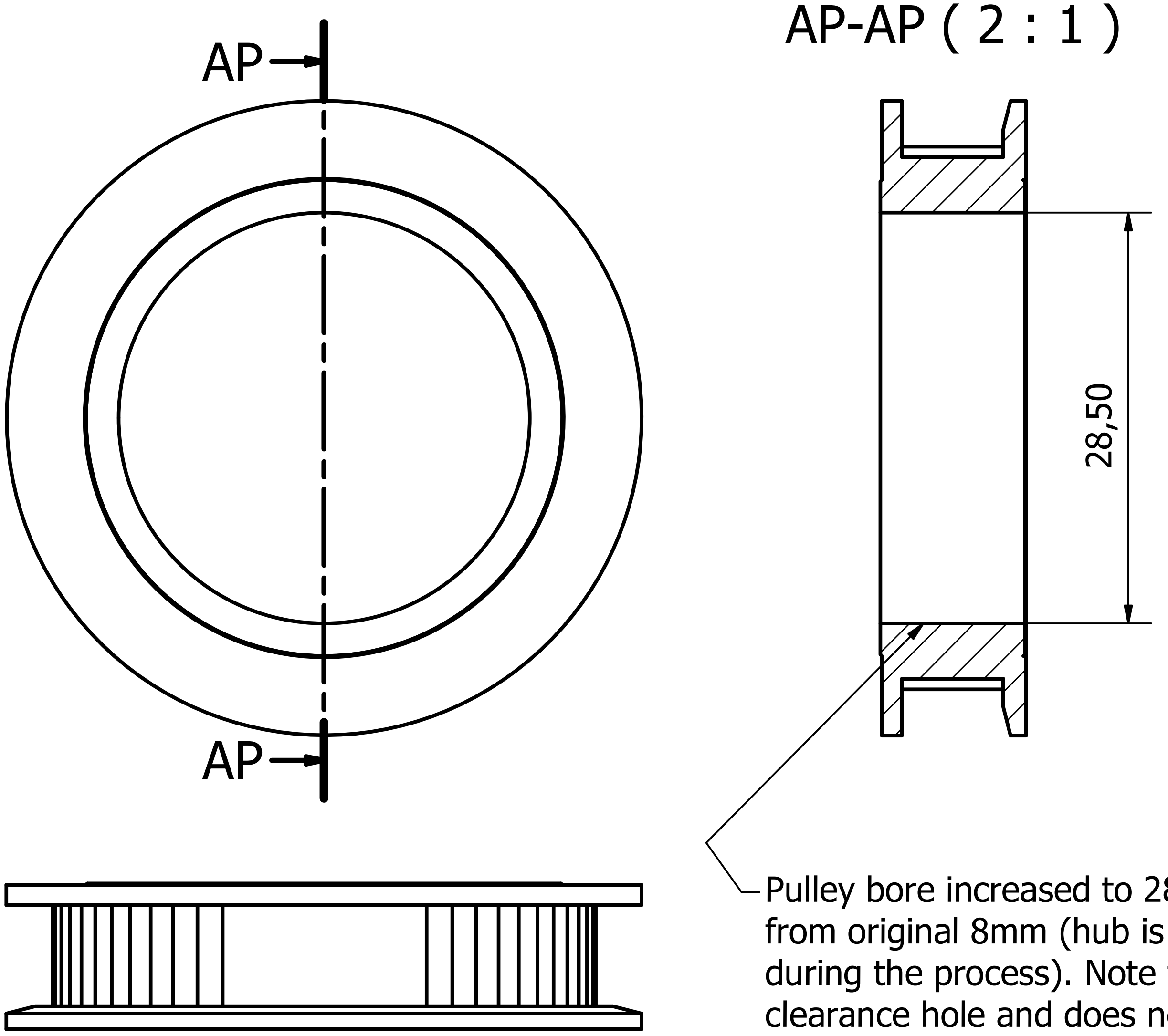
Designed by AlexM	Quantity 2	Manufactured by	Material Aluminum 6061	Date 07/03/2019
NELSON MANDELA UNIVERSITY		Encoder Plate Type B		
		Swerve Drive AGV	Edition	Sheet 204 / 207

Original Part



Modified Part





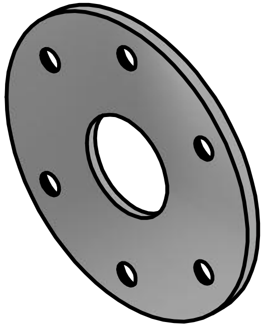
Pulley bore increased to 28.5mm from original 8mm (hub is removed during the process). Note this is clearance hole and does not mate to another part.

Designed by	Quantity	Manufactured by	Material	Date
AlexM	2		Aluminum 6061	07/03/2019

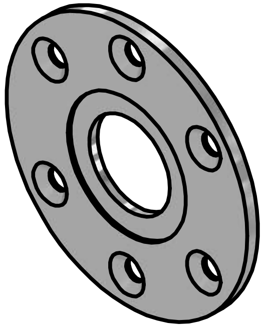
NELSON MANDELA UNIVERSITY

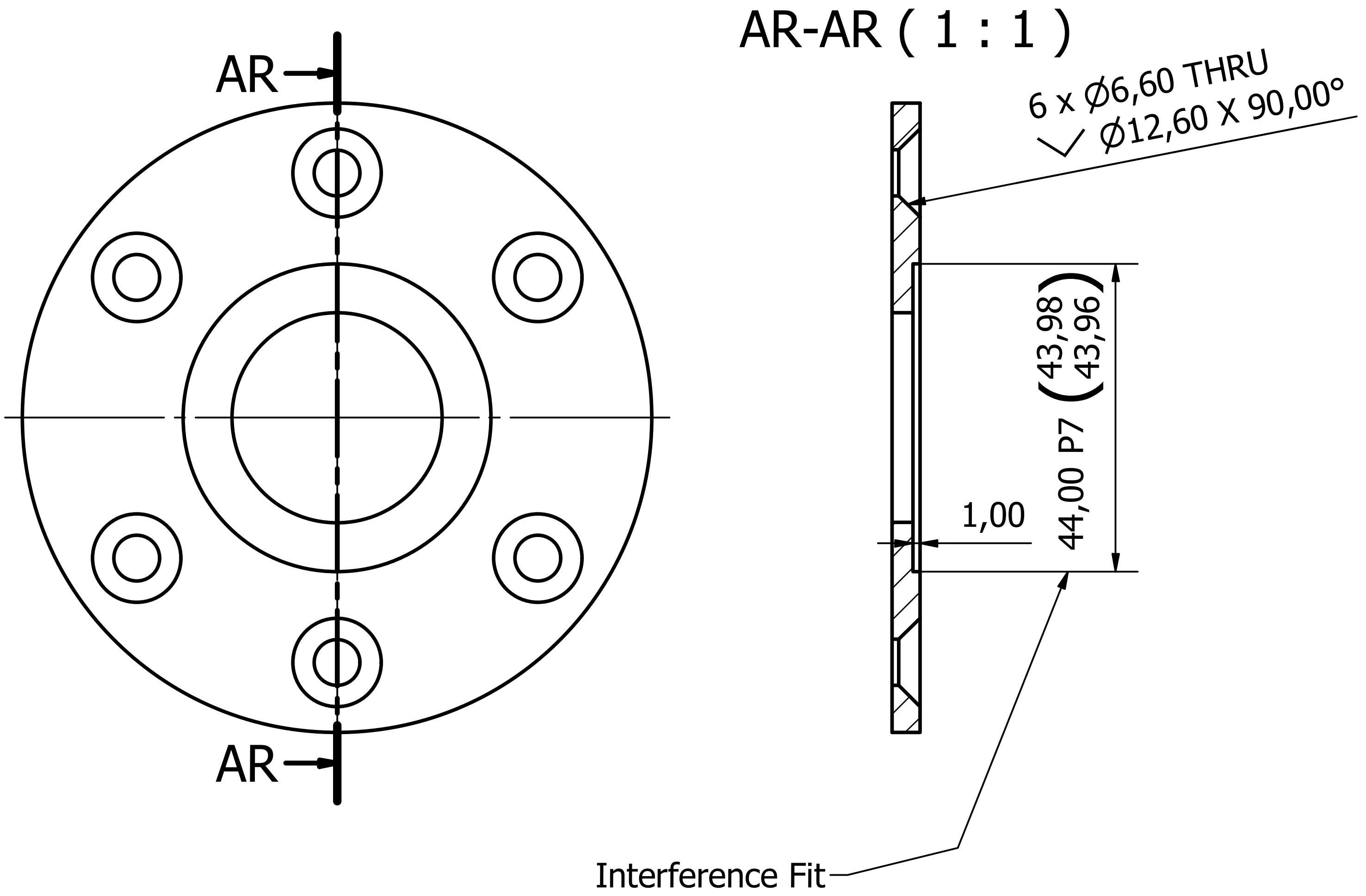
Machined GT2 60T Pulley		
Swerve Drive AGV	Edition	Sheet 205 / 207

Original



Machined

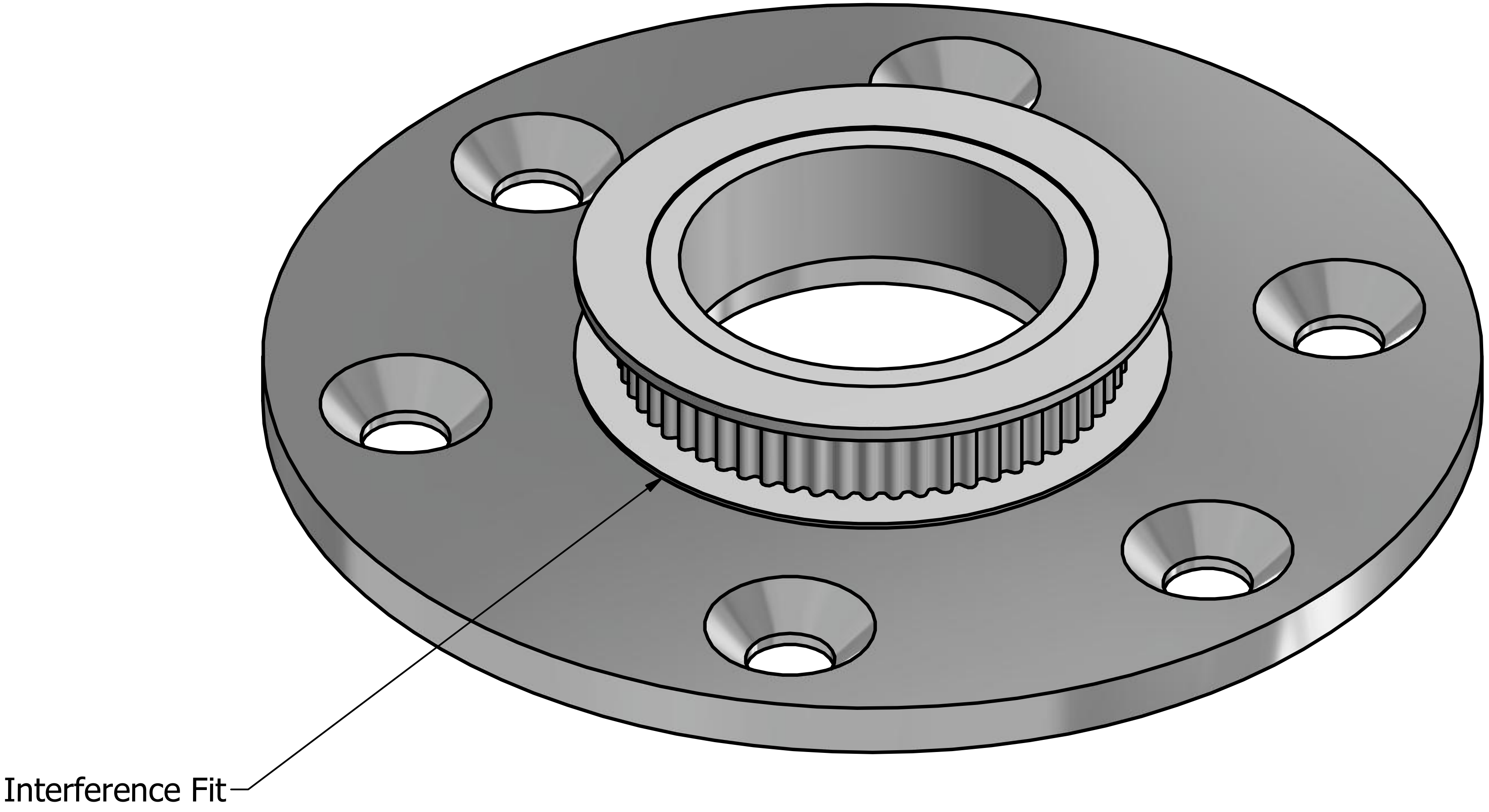




Designed by	Quantity	Manufactured by	Material	Date
AlexM	2		Aluminum 6061	07/03/2019

NELSON MANDELA UNIVERSITY

GT2 Pulley Attachement Plate		
Swerve Drive AGV	Edition	Sheet 206 / 207



Designed by AlexM	Quantity 2	Manufactured by	Material	Date 07/03/2019

NELSON MANDELA UNIVERSITY

Gearbox Encoder Pulley System

Swerve Drive AGV	Edition	Sheet 207 / 207

J Appendix - AGV Battery Management Proposal

1.1 Topic

Battery Management System for Implementation on Automatic Guided Vehicles

1.2 Background

AGVs are becoming increasingly popular in production, especially when taking over jobs previously done by forklifts and tugger vehicles. However, unlike forklifts and tugger vehicles most modern AGVs rely on batteries as their source of power in place of more traditional internal combustion engines. This presents a whole new set of challenges, the primary being battery management. Accurate monitoring of the battery is required to ensure that the state of charge of the battery is known at all times and the charging cycles of the battery are optimised to ensure longevity of the battery. The battery banks found in these systems are often removable and attached to the AGV by "hold in place pins". Power is thus transferred trough contactors rather than permanent fixings to ensure easy removal.

1.3 Aim/Objectives

The overall aim of this project is to develop a battery management system to monitor the state of charge of a 48V battery bank. The management system must make use of "Coulomb Counting" to measure state of charge and employ a siemens "IoT 2000" board as the controller. The following objectives must be achieved.

- Literature research on "Coulomb Counting" and battery management (including "Depth of Discharge")
- Use and programming of an IoT 2000 board (provided)
- Implementation of a "Coulomb Counting" circuit
- Development of PCB boards for final circuits
- Communication of relevant battery states and conditions from the IoT 2000 board to a remote PLC via Profinet
- Mechanical design on battery box (could include contactors/ hold in place pins/ ect)
- Testing and Validation of the system

The battery box and batteries will be provided for the student.

1.4 Performance Requirements

- "Coulomb Counting" to determine battery status
- "Battery Low", "Battery Empty Status" and "Battery Approaching End of Lifecycle" warnings communicated to remote PLC via Profinet
- Approximate battery life left based on current power consumption written to PLC via Profinet at user defined intervals
- The system must be safe (i.e. relay to disengage battery bank from contacts when removed from AGV and other safety systems)
- Use of industrial components (24Vdc logic)

1.5 Budget

????????

Promoter : Theo van Niekerk	Student :
Signature :	Signature :
Date :	Date :

K Appendix - Low Cost Uninterrupted Power Supply

Low Cost PLC Uninterrupted Power Supply for use on AGVs with a Removable Battery Banks

Alexander B.S. Macfarlane
EBEIT (Engineering, Built Environment & IT)
Nelson Mandela University
Port Elizabeth, South Africa
alex.macfarlane@mandela.ac.za

Theo van Niekerk
EBEIT (Engineering, Built Environment & IT)
Nelson Mandela University
Port Elizabeth, South Africa
theo.vanniekerk@mandela.ac.za

Udo Becker
line 2: *IFA, Faculty of Automotive Engineering*
line 3: *Ostfalia UAS*
Wolfsburg, Germany
u.becker@ostfalia.de

***Abstract*—When it comes to electrical AGVs, downtime due to charging batteries is a significant hurdle. This is often dealt with in numerous ways; including opportunistic charging, mobile charging, quick charging, etc. This paper will focus on the difficulties of using exchangeable "power units", which entails making the entire battery system (including the battery management system) removable from the host machine. This eliminates downtime, due to charging, as the entire power plant is replaced with a "fresh" one. The depleted power unit can then be recharged at a sustainable (from a battery lifespan perspective) rate. The major disadvantage of this strategy is that with the power plant removed from the AGV, all control systems are unpowered thus deactivated. Hence the focus of this paper on creating a "control system UPS" to keep the control systems powered even when the main power unit is down and thus solving one of the major problems with a removable battery system.**

Keywords—DC UPS, UPS, Uninterrupted Power Supply, AGV, Automatic Guided Vehicle, Lead Acid, Battery

I. Introduction

An Automatic Guided Vehicle (AGV) is an intelligent self-driving vehicle used to perform repetitive fetch, carry and deposit tasks in industry. These machines often use electrical power trains as opposed to more traditional internal combustion engines (ICs), due to concerns about greenhouse gasses and newer European Union laws[1]. The use of an electrical power train has quite a few advantages over IC systems such as a lower carbon footprint (provided the electricity is cleanly sourced) and fewer mechanical components (most electrical motors consist of primarily a rotor and some form of commutation ring as their only moving parts, compared to the hundreds of moving parts of an IC engine)[2]. There are however some disadvantages, the most prominent and the one focused on in this paper, is that of the charge time of the system's batteries[3]. The recharge time of such a system using batteries is excessively long when compared to the near instantaneous "recharge" of an IC engine (which simply requires its fuel supply to be topped up). There are numerous proposed solutions to deal with this issue some include:

- Opportunistic charging – Charging while the AGV is performing a stationary task [3]
- Mobile charging – Using special lanes/tracks between nodes that have a charging system integrated for in-motion charging [4]
- Quick charging – Using super capacitors to store a massive instantaneous charge and slowly charge the batteries with this energy during operation [5]
- Wireless Charging – A possible future technology (currently range to small ~1m in ideal circumstances) [6]
- Exchangeable power units – Battery and battery management system removable from host machine [7]

The AGV that this paper will focus on will implement a number of these strategies however this paper will specifically focus on the last point "exchangeable power units" and the difficulties they introduce; specifically the loss of power the host system, AGV, will experience when the main power unit is removed.

II. Justification

The implementation of the removable "power module" (battery and battery management system (BMS)) has already been implement on the subject AGV. The problem is keeping the control system active when the main power bank is removed. The main control system must be active when the AGV is exchanging batteries for the following reasons. Firstly, the AGV must continually communicate to a higher control network. Secondly, the AGV takes part in the "power module" exchange action; by locking the battery bank in place and closing a solid-state relay that allows power to flow to the rest of the system. Lastly, the AGV has an onboard Industrial PC (IPC), these systems state time to reboot (5 to 10 minutes) and can easily be damaged it power is suddenly cut [8].

III. Syetem Overview

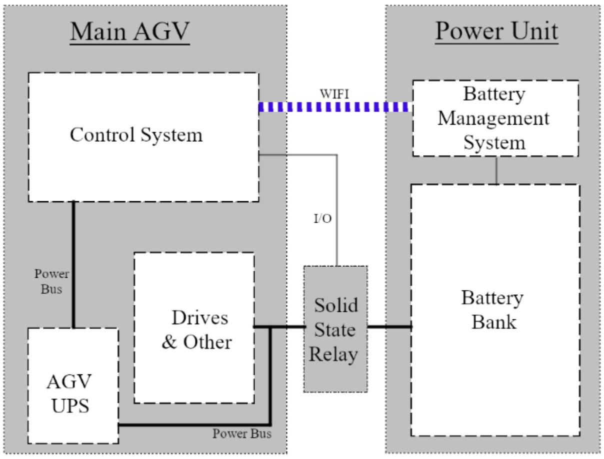


fig. 1. AGV and Power Unit Layout

As illustrated in fig. 1, the AGV is broken into two parts, the "main AGV" and the "power unit. Power is transferred to the "main AGV" from the "power unit" via a solid-state relay. The purpose of the solid ate relay is to prevent arcing on the power unit's terminals during battery exchange. The relay is controlled by the AGV's control system. All communication between the battery management system (BMS) and AGV control system is done via WIFI communication to reduce the number of physical connections. fig. 1 also clearly indicates the position of the uninterrupted power supply (UPS) on the main AGV's body. It is to be noted that when the AGV is exchanging batteries the main drive chain and any auxiliary electronics will be unpowered.

IV. AGV PLC UPS Requirements

Since this UPS system is supply power to a Programmable Logic Controller (PLC), Siemens Industrial PC (IPC) and industry standard distributed peripherals (DP) The UPS will need to provide a nominal voltage of 24 Vdc however this can vary between 19.2 Vdc and 28.8 Vdc [9], which is the tolerable range of the Siemens range of equipment used for the AGV control system.

The largest concern for this UPS is cost. Ideally this system should be as cost effective as possible. Originally an off-the shelf unit was to be used, an example of this is the TRACOPOWER TSP-BCMU360, which is available from RS Components. At the time of creating this design the cost of this unit was R 3,347.86, this price excludes batteries [10].

Ideally this system should have only one battery attached to it. Since 24 Vdc batteries do exist, this battery would seem like the ideal choice; however, they tend to be cost prohibitive to implement when compared to using two 12 Vdc batteries, this is illustrated in Table I.

Table I. Battery Cost from Nelson Mandela University Suppliers

Battery Type	**Battery Cost Analysis**			
	Supplier	***Number of Batteries Needed to make 24V @ 7Ah***	***Single Battery Cost***[a]	***Total Cost to Implement 24VDC @ 7Ah***
24VDC 7Ah	RS Components[10]	1	R 6,364.58	R 6,364.58
12VDC 7Ah	RS Components	2	R 435.33	R 870.66
24VDC 4.5 Ah	Mantech[11]	2	R 487.36	R 974.72
12VDC 7Ah	Mantech	2	R 179.52	R 359.04

[a.] Prices taken September 2019

Thus, it was decided to use two 12 Vdc batteries rated at 7 Ah to give at total rating of 24 Vdc @ 7 Ah when in series as opposed to the 24 Vdc equivalent. Although it is possible to use a 12 Vdc battery and boost the DC voltage up to the required 24 Vdc, this strategy was decided against due to the added complexity a buck-boost converter would add.

The next concern for the system, is the total load drawn from the battery and the length of time the backup battery will last for. This was determined by first calculating the total load of the system when in the low powered "power module" removal state (i.e. what systems are active during this state) then comparing maximum allowable current draw of the batteries.

Table II below excludes any system not active or **not in use** during the power module removal state as many of the distributed I/O output systems can still be theoretically driven during this state from the UPS, however would quickly max out the UPS's current limit. These I/O will be software interlocked to prevent operation during the "power module" removal state.

Table II. Power Consumption During Low Power State[B]

Item	**Power Usage**		
	Current Draw (A)	***Quantity***	***Total Current***
Siemens S7-1500 PLC	0.6A	1	0.6A
Siemens Ditrabuted I/O Inputs	0.09A	16 total (4 modules)	0.36A
Siemens Distrabuted I/O Outputs	0.5A	6 total (4 modules)	3A
SICK Saftey PLC	0.3	1	0.3A
Siemens IPC (max)	4A	1	4A
Battery Bank Eject Motor **(not connected through UPS only to UPS battery)**	4A	1	4A

[B.] Data taken from Siemens[12] and Sick[13] website

Thus, from Table II the total power drawn by the control system during the low power state through the UPS is 8.26 A, while an additional 4 A will be drawn directly from the UPS batteries for the "power module" eject system motor. The eject motor does not need to have uninterrupted power characteristics as it will only operate when the AGV's main "power module" is disconnected.

It was decided that in a worst-case scenario, where a manual battery exchange occurs, the maximum amount of time that the main "power module would be disconnected from the core AGV would be no longer than 5 minutes. Thus, the UPS battery should be able the sustain a maximum current of 12.26 A (sum of UPS draw of 8.26 A and eject motor draw of 4 A) for at least 5 minutes.

The battery selected to make up the UPS battery bank was thus a RITAR RT1270B. Two of these batteries in series would produce a 24 V battery bank of 7 Ah. Since these are lead acid batteries it is not recommended to discharge the batteries at a C_{rate} value higher than 4 [14]. The C_{rate} value of a battery relates its discharge current to its total capacity, see (1):

$$I_{discharge} = C_{rate} C \tag{1}$$

Where $I_{discharge}$ is the discharge current in amperes (A) and C is the capacity of the battery in amp-hours (Ah).

The RT1270B can provide the needed current of 12.26 A if a C_{rate} of 2 is used (At 2C_{rate} the discharge is 14 A). Shown in fig. 2 (Note the 2C_{rate} is estimated as a line of best fit) the battery will reach a voltage of 11.5 Vdc at approximately 7.02 minute mark. 11.5 Vdc is chosen as the cutoff voltage (i.e. battery is "flat") as this corresponds to a depth of discharge (DoD) of 50% for this battery (see fig. 3), which for lead acid batteries is often stated as the best compromise between total lifespan and usable capacity[15].

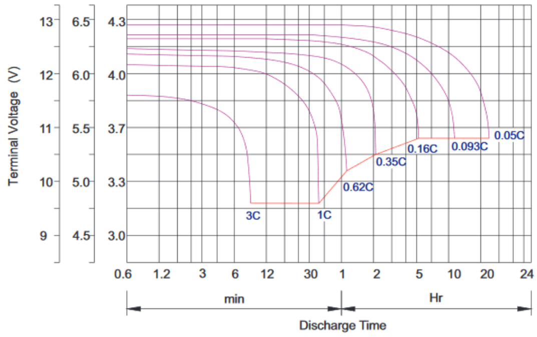


fig. 2. RT1270B Discharge Characteristic Curve[16]

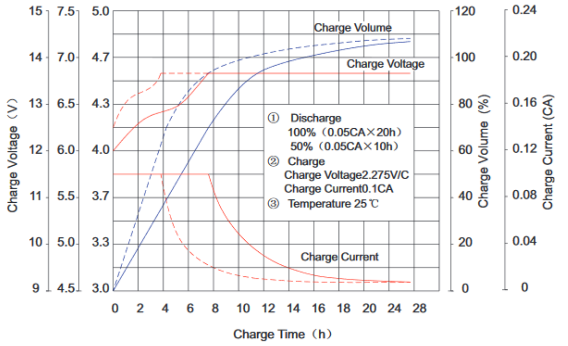


fig. 3. RT1270B Charge Characteristic Curve[16]

As for the UPS the industry standard for switching speed seems to be between 3 ms and 10 ms. Though there are "high efficiency double conversion" UPS's that take between 1 ms and 3 ms to switch in [17]. Thus, ideally the UPS should be able to switch over faster than 8 ms.

V. Working Principal Of The UPS

Since this UPS is operating on a DC system a redundant DC power supply topography [18] can be used. The principal of operation of this system is shown below, in fig. 4.

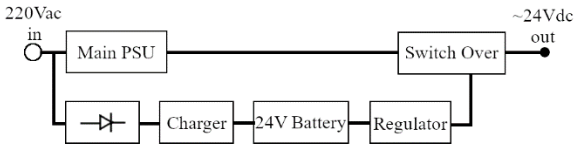


fig. 4. Block Diagram Operation of UPS System

As illustrated in fig. 4, the AGV does contain a 220 Vac bus. This is generated by an inverter from the main "power module". The inclusion of this AC bus simplifies matters considerably when generating multiple, high current, DC voltages, eliminating the need for DC-DC converters. When the main "power module" is present in the AGV the 24 Vdc is generated for the control systems via a standard AC to DC power supply unit (Main PSU) at the same time the backup battery for the system (24 V Battery) is charged. When the "power module" is removed from the AGV the 230 Vac with shut off, at this point the backup battery will be switched in (Switch Over) automatically. The voltage generated by the backup battery will be regulated by a regulator, see fig. 4, to ensure it voltage is stable.

VI. Working Principal Of The UPS

The UPS system was designed with the aid of National Instruments Multisim 12.0 and consists of the following parts:

- Charger
- Regulator
- Switch Over

A. Charger

The charger is a constant current type charger and. This means that the charger's current fed to the battery is uniform regardless of the battery's voltage. This strategy is ideal as a low-cost strategy for charging battery's that are cyclically discharged and have a relatively long recharge time available [19]. Constant current charging also has the advantage of significantly reducing the imbalance charge state of cells, when compared to other charging strategies. This also reduces the need for a BMS on the backup battery bank which would add extra cost and complexity to the system [20].

To achieve the afore mentioned goals the LM317 variable voltage regulator from Texas Instruments was used with a Fairchild BC547B NPN transistor. A reference circuit for a constant current lead acid battery is provided for in the LM317 documentation and is illustrated in fig. 5.

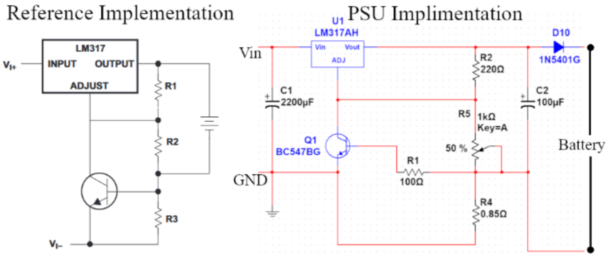


fig. 5. Implementation of The LM317 as a Current-Limited Charger

fig. 5 shows the reference implementation of the LM317 as current limited charger along with the implementation in the AGVs PSU circuit. In the PSU Implementation (all component labels will refer to the PSU implementation) resistor R2 and potentiometer R5 are used to set the charging voltage for the system. Ideally this will be calibrated to 30 Vdc, the battery will only see 28.8 Vdc due to the 1.2 V drop over D10, this will correspond to a cell charge voltage of 2.4 V (for a 24 V system).

The constant current value is set using resistor R4 which at 0.45Ω corresponds to a charging current of 1.37 A. This charging current is approximately equal to the 0.2C (or 0.2*7Ah = 1.4A) recommended by the RT1270B data sheet.

Constant current charging is achieved as follows. If the current through R4 increases, the resultant voltage over the resistor will increase. As this voltage approaches 0.7 V, the base to emitter junction of transistor Q1 will begin to conduct. This effectively reduces the resistance of R5 by sinking extra current from the adjustment pin of the LM317 voltage regulator (U1) and thus lowers the output voltage of the regulator to maintain a constant current over the battery.

The capacitor C1 is used as smoothing capacitor from the bridge rectifier, it is also needed by the voltage regulator to maintain stability. C2 is used to improve transient response.

B. *Regulator*

The regulator portion of the UPS is used to maintain a constant voltage output to the control system. This is necessary as the voltage of the battery will vary with its level of charge.

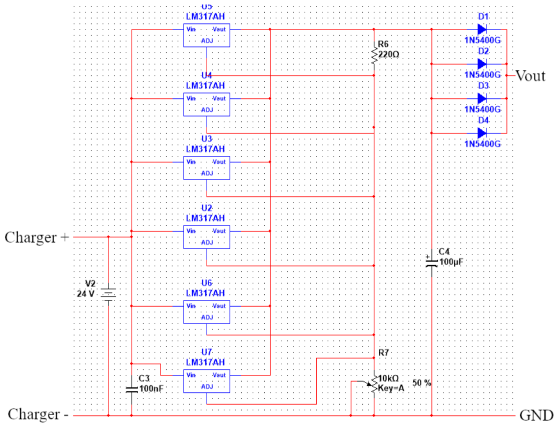


fig. 6. UPS Regulation Circuit

The regulation circuit of UPS system, fig. 6, relies on a parallel array of LM317 variable voltage regulators. The parallelization is done to increase the maximum current that can be drawn from the system the LM317 has a maximum output current rating of 1.5 A. Thus 6 in parallel will be able to provide 9 A in total. This exceeds the 8.26 A needed by the system by 8.2%. The output voltage of the regulator bank and is set to 21 V using resistors R6 and R7. This value corresponds to 2 V less than the minimum battery voltage (when at 50% DoD) of 23 V. The regulators cannot be set at 23 V as the regulators have a voltage minimum differential of 2 V between Vin and Vout. Once the current has flown through Schottky diodes D1 to D4, the voltage will drop to 20.475 V. This is still within the tolerable range of the components in the control system.

C. *Switch Over*

Due to the inclusion of the Schottky diodes D1 to D4 in fig. 6, the UPS can be directly attached to the control system, with the diodes acting as the changeover as the 21 V at the diodes will always be reverse biased against the main PSU's 24 V. Thus, changeover will occur when the voltage from the main PSU drop to zero and diodes D1 to D4 become forward biased.

D. *Full Schemtic*

The full schematic diagram of the AGV UPS system can be found in fig. 7. The AC to DC rectification is done using a 230 Vac to 35 Vac (T1), full bridge rectifier consisting of four 1N5401 (D6 to D9) which can provide 3 A to the battery charger circuit. Smoothing of the half wave AC produced by the bridge rectifier is done using a 2200 uF electrolytic capacitor. This produces peak of ~47.94 V a minimum of ~47.91 V with a ripple of 3 mV.

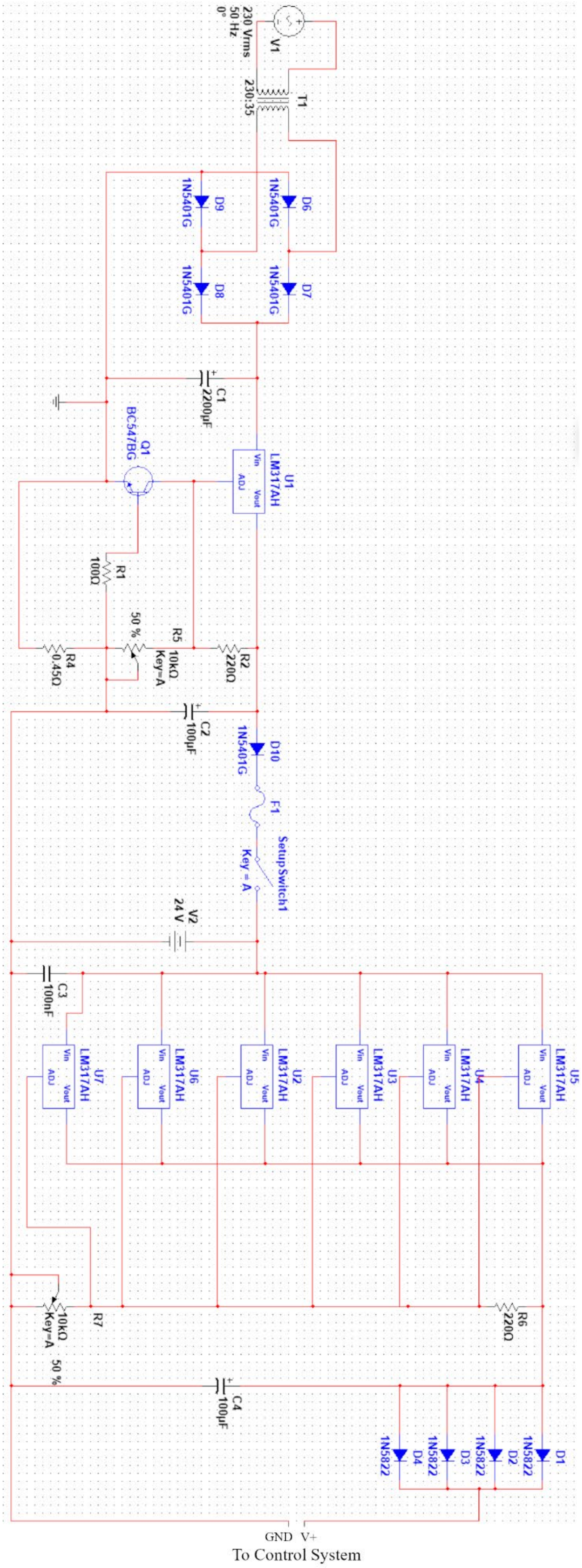


fig. 7. Full Mutism Schematic of AGV UPS System

Also, of note in the main schematic, fig. 7, is the inclusion of a 1.5 A fuse (F1) between the battery charger and battery. This is to prevent an accidental overcurrent draw if the battery is depleted, the charger is still connected to the powered AC bus and the main DC PSU, see fig. 4, is not running. Since control system would then attempt to draw the full 12.26 A from the charger, which is only capable of supplying maximum 1.5 A. (LM317 limitation).

The "SetupSwitch1", in fig. 7, is used to calibrate the charge voltage of the charger portion of the UPS. When calibrating the charger, the switch is opened. The open voltage is measured using a multimeter connected between ground and the output of D10. This voltage should be set to 28.8 Vdc using potentiometer R5 (50% will be the ideal starting point).

VII. Printed Circuit Board Design

The design of the UPS was built such that the entire system fits on one Printed Circuit Board (PCB). The PCB is illustrated in fig. 8.

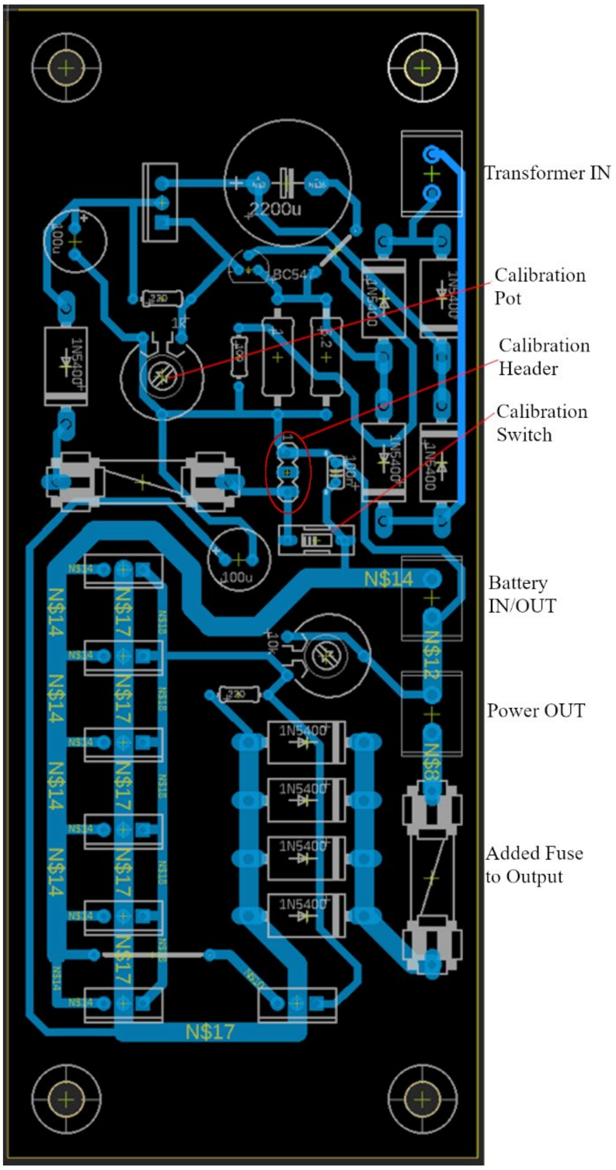


fig. 8. Printed Circuit Board for PLC UPS

The PCB is a single sided PCB to save cost, with the components mounted on the opposite face to the traces. Components that cannot be attached to the PCB include the battery bank and transformer. The transformer is attached directly to the AC bus and its output is attached to the "transformer IN" screw terminal of the PCB. The battery bank is attached via the "Battery IN/OUT" screw terminal and finally the "Power OUT" is connected to the same DC bus the main 24V dc PSU of the AGV is connected to.

The "Calibration Header" in fig. 8 can be used to attach a multimeter to set the correct charge voltage for the charging circuit of the UPS. When this is done "calibration switch" must be switched to the off position and the "Calibration Pot" is used to set the correct charging voltage of 28.8 Vdc.

There is one discrepancy between the schematic diagram shown in fig. 7 and the PCB shown in fig. 8, that is the inclusion of a second fuse for the output of the UPS, this is labeled as "added fuse to output" on fig. 8. This fuse is rated at 9A and prevent overloading the voltage regulator or more commonly shorting out the leads when connecting the UPS in, when the battery is attached.

VIII. Conclusion & Future Improvements

This AGV PSU was very hastily put together, to get a low-cost PSU system for an existing AGV. However, it works surprisingly well. Some improved that could be made to future iteration of this system include the removal of the LM317 based voltage regulation segment and replacing this with a true buck-boost converter able to produce an output voltage of approximately 23.48 V due to the voltage drop over the Schottky diodes used for switching in the UPS.

This System should not be implemented on any electrical system that requires more stringent 24 Vdc adhesion due to the manner it does the switching, via biased diodes. Even with the afore mentioned improved voltage regulation segment the voltage between using the AGV's main PSU and back up UPS will differ by 0.52 V. As the system stands, without the improved voltage regulation system, the voltage drop between the main PSU and UPS is 3.525 V or a 14.7% voltage drop.

IX. Bibliography

[1] A. C. R. Teixeira and J. R. Sodré, "Impacts of replacement of engine powered vehicles by electric vehicles on energy consumption and CO 2 emissions," *Transp. Res. Part D Transp. Environ.*, vol. 59, pp. 375–384, 2018.

[2] R. Baran and L. F. L. Legey, "The introduction of electric vehicles in Brazil: Impacts on oil and electricity consumption," *Technol. Forecast. Soc. Change*, vol. 80, no. 5, pp. 907–917, 2013.

[3] P. Gallo, "Charging electric vehicles using opportunistic stopovers," *Conf. Proc. - 2017 17th IEEE Int. Conf. Environ. Electr. Eng. 2017 1st IEEE Ind. Commer. Power Syst. Eur. EEEIC / I CPS Eur. 2017*, pp. 1–5, 2017.

[4] K. Doubleday, A. Meintz, and T. Markel, "An opportunistic wireless charging system design for an on-demand shuttle service," *2016 IEEE Transp. Electrif. Conf. Expo, ITEC 2016*, pp. 1–6, 2016.

[5] S. Tanaka, N. Fukujyu, and S. Kaneda, "Configurations of Micro EV Quick-Charging

System with EDLC Storage," *Proc. - Int. Comput. Softw. Appl. Conf.*, vol. 2, pp. 294–299, 2016.

[6] Y. Hori, "Novel EV society based on motor/ capacitor/ wireless - Application of electric motor, supercapacitors, and wireless power transfer to enhance operation of future vehicles," *2012 IEEE MTT-S Int. Microw. Work. Ser. Innov. Wirel. Power Transm. Technol. Syst. Appl. IMWS-IWPT 2012 - Proc.*, pp. 3–8, 2012.

[7] Z. Xian and G. Wang, "Optimal dispatch of electric vehicle batteries between battery swapping stations and charging stations," *IEEE Power Energy Soc. Gen. Meet.*, vol. 2016-November, pp. 1–5, 2016.

[8] H. Shareef, S. N. Khalid, M. W. Mustafa, and A. Mohamed, "Modeling and simulation of overvoltage surges in low voltage systems," *PECon 2008 - 2008 IEEE 2nd Int. Power Energy Conf.*, no. PECon 08, pp. 357–361, 2008.

[9] Siemens, "SIMATIC ET200SP CPU 1512SP-1 PN Manual," *Siemens Answears for Industry*, 2014. [Online]. Available: https://cache.industry.siemens.com/dl/files/013/90157013/att_83463/v1/et200sp_cpu1512sp_1_pn_manual_en-US_en-US.pdf. [Accessed: 05-Oct-2019].

[10] RS Components, "RS-Online," *Online Catalogue*, 2019. [Online]. Available: https://za.rs-online.com/web/.

[11] Mantech, "Mantech Electronics," *Online Catalogue*, 2019. [Online]. Available: https://www.mantech.co.za/. [Accessed: 10-May-2019].

[12] Siemens, "Siemens Industrial Mall," *Siemens Industrial Mall*, 2019. [Online]. Available: https://mall.industry.siemens.com/goos/WelcomePage.aspx?regionUrl=/za&language=en. [Accessed: 05-Oct-2019].

[13] SICK, "SICK Sensor Intelligence," *SICK Sensor Intelligence*, 2010. [Online]. Available: https://www.sick.com/ag/en/. [Accessed: 05-Oct-2019].

[14] D. Berndt, *Maintenance Free Batteries*, 1st ed. Great Britain: John Wiley & Sons, 1993.

[15] Gates Energy Products, *Rechargeable Batteries: Applications Handbook*, 1st ed. Stonham: Butterworth-Heinemann, 1991.

[16] SHENZHEN RITAR POWER CO LTD, "RT1270B Datasheet," *RITAR Datasheets*, 2014. [Online]. Available: https://www.mantech.co.za/Datasheets/Products/RT1270B-180215A.pdf. [Accessed: 05-Oct-2019].

[17] Institute of Electrical and Electronics Engineers, *IEEE Recommended Practice for Emergency and Standby Power Systems for Industrial and Commercial Applications*, vol. 1995. 1996.

[18] R. S. Malik, D. N. Ho, and V. V. Oak, "Low Cost Redundant AC to DC Power Supply," US 6,630,753 B2, 2003.

[19] Y. E. Abu Eldahab, N. H. Saad, and A. Zekry, "Enhancing the design of battery charging controllers for photovoltaic systems," *Renew. Sustain. Energy Rev.*, vol. 58, no. September 2018, pp. 646–655, 2016.

[20] Engineers Edge LLC, "Battery Appication & Technology," 2019. [Online]. Available: https://www.engineersedge.com/battery/constant_current_charging.htm. [Accessed: 06-Oct-2019].

L Appendix - Siemens Simantic s7-1512SP PLC

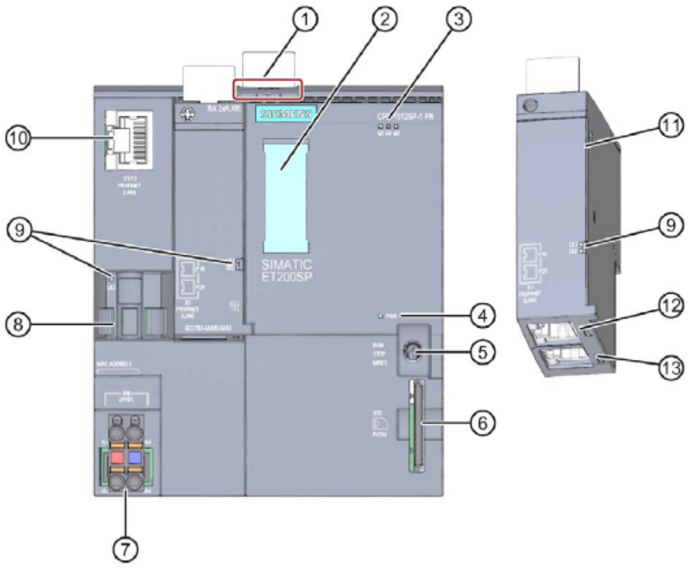


Figure L.1: s7-1512SP PLC CPU Labelled Diagram[1]

The items labelled in figure L.1 are defined:

1. Mounting rail release
2. Label strip
3. Status and error indicator LEDs
4. Power LED
5. Mode selection switch

[1] Image adapted from Siemens [68].

6. SIMATIC memory card slot
7. 24 VDC infeed point (double terminal for piggybacking)
8. Profinet cable support point
9. Profinet port activity LEDs
10. Profinet port 3
11. Buss adaptor when removed from CPU
12. Profinet port 1
13. Profinet port 2

M Appendix - PLC I/O Assignment List (Tag Tables)

Table M.1: s7-1512SP Signal Module Tag Table Part 1

Address	Tag	Description
	DI 16x24VDC ST	
I0.0	unused	no function
I0.1	unused	no function
I0.2	unused	no function
I0.3	unused	no function
I0.4	unused	no function
I0.5	unused	no function
I0.6	unused	no function
I0.7	unused	no function
I1.0	unused	no function
I1.1	AGVPowerSW	Signals to PLC to shut down AGV
I1.2	BLK Pushbutton	Reset Steering Angle Potentiometers
I1.3	RED Pushbutton	Error Acknowledge
I1.4	Battery PIP NO	Battery unit in place sensor (NO)
I1.5	Battery PIP NC	Battery unit in place sensor (NC)
I1.6	Eject Motor Feedback Eject	Battery unit motor CW contactor status feedback
I1.7	Eject Motor Feedback Insert	Battery unit motor CCW contactor status feedback

cont...

Table M.2: s7-1512SP Signal Module Tag Table Part 2

Address	Tag	Description
		DI 8x24VDC HF
I2.0	Pendant SW0	AGV ON/OFF
I2.1	Pendant SW1	Increment mode
I2.2	Pendant SW2	Manual Mode Deadman SW
I2.3	Pendant SW3	Jog Servo A
I2.4	Pendant SW4	Jog Servo B
I2.5	Pendant SW5	Jog Stepper A
I2.6	Pendant SW6	Jog Stepper B
I2.7	Pendant SW7	Acknowledge faults
		DQ 16x24VDC/0.5A ST
Q0.0	unused	no function
Q0.1	unused	no function
		DQ 16x24VDC/0.5A ST
Q0.2	unused	no function
Q0.3	unused	no function
Q0.4	unused	no function
Q0.5	unused	no function
Q0.6	AGVSelfKillSW	Shutdown DC UPS to kill AGV
Q0.7	Stack Light 1	Red stack light
Q1.0	Stack Light 2	Yellow stack light
Q1.1	Stack Light 3	Green stack light
Q1.2	Stepper B Enable	Enable stepper motor B drive
Q1.3	Stepper A Enable	Enable stepper motor A drive
Q1.4	Front Light	Activate front flashing drive light
Q1.5	Disable Cooling Fans	Override cooling fans to OFF state
Q1.6	Eject Motor Eject	Battery unit motor CW contact
Q1.7	Eject Motor Insert	Battery unit motor CCW contact
		DQ 8x24VDC/0.5A HF
Q2.0	Pendant LED0	AGV ON/OFF status
Q2.1	Pendant LED1	Wheel alignment error
Q2.2	Pendant LED2	Manual mode active

cont...

Table M.3: s7-1512SP Signal Module Tag Table Part 3

Address	**Tag**	**Description**
Q2.3	Pendant LED3	Auto mode active
Q2.4	Pendant LED4	Commissioning mode active
Q2.5	Pendant LED5	Homing steering active
Q2.6	Pendant LED6	Testing mode active
Q2.7	Pendant LED7	Error acknowledge required
	AI 4xU/I 2-wire ST	
IW30	Speed Pot	Pendant speed potentiometer
IW32	Strafe Pot	Pendant strafe angle potentiometer
IW34	Unit A Steering Pot	Pendant steering angle potentiometer
IW36	unused	no function
	AI 4xU/I 2-wire ST	
IW20	Unit A Steering Pot	Unit A absolute steering angle
IW22	Unit B Steering Pot	Unit B absolute steering angle
	F-DI 8x24VDC HF	
I4.0	Rear E Stop CH0	Rear E Stop Channel 0
I4.1	Front E Stop CH1	Front E Stop Channel 1
I4.2	SICK IO Q1	Safety bit coms Q1 SICK PLC
I4.3	SICK IO Q2	Safety bit coms Q2 SICK PLC
I4.4	*Rear E Stop CH4*	Rear E Stop Channel 4 (redundancy)
I4.5	*Front E Stop CH5*	Front E Stop Channel 5 (redundancy)
I4.6	SICK IO Q3	Safety bit coms Q3 SICK PLC
I4.7	SICK IO Q4	Safety bit coms Q4 SICK PLC
I5.0	Rear E Stop Source CH0	Rear E Stop Channel 0 Source
I5.1	Front E Stop Source CH1	Front E Stop Channel 1 Source
I5.2	unused	no function
I5.3	unused	no function
I5.4	*Rear E Stop Source CH4*	Rear E Stop Channel 4 Source (redundancy)
I5.5	*Front E Stop Source CH5*	Front E Stop Channel 5 Source (redundancy)

cont...

Table M.4: s7-1512SP Signal Module Tag Table Part 4

Address	**Tag**	**Description**
I5.6	unused	no function
I5.7	unused	no function
	F-DQ 4x24VDC/2A PM HF	
Q10.0	Servo A STO PWR	Servo A SIL safety stop
Q10.1	Servo B STO PWR	Servo B SIL safety stop
Q10.2	Stepper B STO PWR	Stepper B SIL safety stop
Q10.3	Stepper A STO PWR	Stepper A SIL safety stop
I10.0	Servo A STO RTN	Servo A SIL safety stop return
I10.1	Servo B STO RTN	Servo B SIL safety stop return
I10.2	Stepper B STO RTN	Stepper B SIL safety stop return
I10.3	Stepper A STO RTN	stepper A SIL safety stop return
	F-DQ 4x24VDC/2A PM HF	
Q15.0	SICK IO I1	Safety bit coms I1 SICK PLC
Q15.1	SICK IO I2	Safety bit coms I2 SICK PLC
Q15.2	SICK IO I3	Safety bit coms I3 SICK PLC
Q15.3	SICK IO I4	Safety bit coms I4 SICK PLC
I15.0	unused	no function
I15.1	unused	no function
I15.2	unused	no function
I15.3	unused	no function

N Appendix - ET200s I/O Assignment List (Tag Tables)

Table N.1: ET200 Signal Module Tag Table

Address	**Tag**	**Description**
		DI 16x24VDC ST
IW24	Stepper A Drive Analog	Unit A relative steering angle
IW26	Stepper B Drive Analog	Unit B relative steering angle

O Appendix - NAV350 Specifications

A specification list can be found for the NAV350 in table O.1.

Table O.1: Technical Specifications for NAV350-3232

Specification	**Value**
Application	Indoor
Wavelength	905 nm
Laser Class	1 (IEC 60825-1:2014, EN 60825-1:2014)
Scanning Envelope	360 °
Scanning Frequency	8 Hz
Angular Resolution	0.25°
Working Range	0.5 m to 250m (0.5 m to 70 m using reflectors)
Scanning Range	35m @ 10% remission
	100m @ 90 % remission
Enclosure Rating	IP65
Communication	TCP/IP @ 100 Mbit/s
Reflector memory (waypoints)	12 000
Positional Accuracy	4 mm

P Appendix - Wiring the Siemens V90 Drive

A wiring diagram extracted from these notes is illustrated in figure P.1.

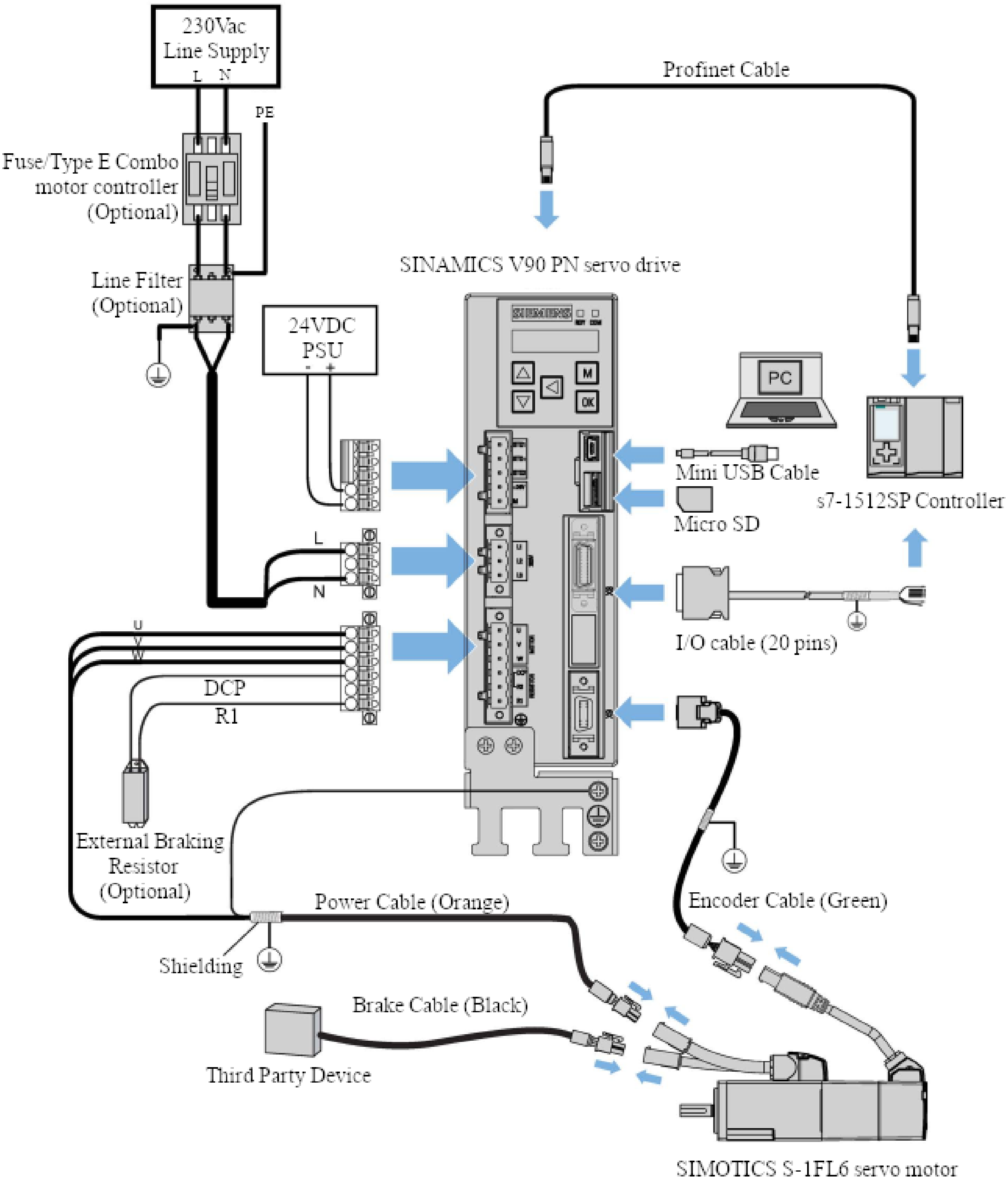


Figure P.1: Reference installation for the Siemens V90 Drives[1]

Note that in figure P.1 allocation is made for the implementation of a line-side filter and

[1] Image adapted from Siemens [75].

external braking resistor, neither of these were implemented on the AGV. As for the Fuse/Type E combo motor controller, a 6A class C circuit breaker was used. However, the 6A circuit breaker only operates on the live line and not both the live and neutral line, as illustrated in figure P.1. Commissioning of the drive in the AGV is done via Profinet, and as such, the Mini-USB was never used. The I/O cable in figure P.1 does not give feedback to the PLC as illustrated in the application notes but rather only feeds pins 17 and 18 to a relay that actuates the brake on the motor (see figure 6.21 for this implementation).

The bus connections X8, X9, and X150 shown in figure 6.25 are listed in the sections that follow.

X8: I/O Cable

The I/O cable attached at connection X8 uses a 20 pin SCSI connector (20-pin MDR socket), whose pinout is shown in figure P.2 and descriptions are listed in table P.1.

Table P.1: Siemens V90 connections: X8

Pin	Signal	Description
1	DI 1	Digital input 1
2	DI 2	Digital input 2
3	DI 3	Digital input 3
4	DI 4	Digital input 4
6	DI COM	Digital inputs common terminal
7	DI COM	Digital inputs common terminal
11	DQ 1+	Digital output 1, positive
12	DQ 1-	Digital output 1, negative
13	DQ 2+	Digital output 2, positive
14	DQ 2-	Digital output 2, negative
17	BK+	Motor holding brake signal, positive
18	BK-	Motor holding brake signal, negative

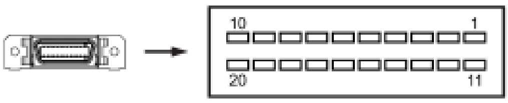


Figure P.2: Servo I/O Cable Connector

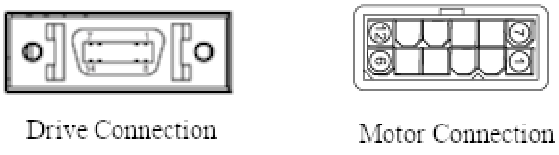
Pins 5, 8, 9, 10, 15, 16, 19 and 20 of the I/O cable serve no purpose and as such were omitted from table P.1.

X9: Encoder Interface

The Encoder interface cable comes from Siemens as a complete cable; the pinout is only listed here for completeness' sake. The connector are illustrated in figure P.3 , while the pinout is listed in table P.2. The connector used on the drive is a UNC 4-40 plug, while the connector used on the motor is a Siemens proprietary connector.

Table P.2: Siemens V90 connections: X9 Drive Side

Pin	**Signal**	**Description**
1	Biss DataP	Absolute encoder data signal, positive
2	Biss DataN	Absolute encoder data signal, negative
3	Biss ClockN	Absolute encoder clock signal, negative
4	Biss ClockP	Absolute encoder clock signal, positive
5	P5V	Encoder power 5V
6	P5V	Encoder power 5V
7	M	Encoder power GND
8	M	Encoder Power GND
9	Rp	Encoder R phase positive signal
10	Rn	Encoder R phase negative signal
11	Bn	Encoder B phase negative signal
12	Bp	Encoder B phase negative signal
13	An	Encoder A phase negative signal
14	Ap	Encoder A phase positive signal



Drive Connection

Motor Connection

Figure P.3: Servo Encoder Interface Connector

The pinout of the encoder cable attached to the motor illustrated in figure P.3 is listed in table P.3.

Table P.3: Siemens V90 connections: X9 Motor Side

Pin	Signal	Description
1	P Supply	Encoder power 5V
2	M	Encoder power GND
3	A+	Encoder A phase positive
4	B+	Encoder B phase positive
5	R+	Encoder C phase positive
6	n.c.	Not connected
7	P Supply	Encoder power 5V
8	M	Encoder power GND
9	A-	Encoder A phase negative
10	B-	Encoder B phase negative
11	R-	Encoder C phase negative
12	Shielding	Cable shield ground

X150: Profinet Interface

The Profinet interface is standardised, so this document will not include the pinout.

Q Appendix - Wiring the Festo CMMS-ST Stepper Drive

X1: I/O Interface

The pinout of the I/O interface is designated in table Q.1; this connection uses a standard DB25 (colloquially called a "parallel port").

Table Q.1: Festo CMMST connections: X1

Pin	Signal	Description
1	SGND	Shielding for analog signals
2	DIN12 / AIN0	Mode bit 0 / analog setpoint
3	DIN10	Record selection 4
4	+VREF	10V analog reference
5	n.c.	Not connected
6	GND24	Digital inputs GND
7	DIN1	Record selection bit 1
8	DIN3	Record selection bit 2
9	**DIN5**	**Controller enable**
10	DIN7	Limit switch 1
11	DIN9	Mode bit 1
12	DOUT1	Motion Complete
13	DOUT3	Common Error
14	**AGND**	**0V analog reference**
15	**DIN13**	**Stop (active low)**
16	DIN11	Record selection 5
17	**AMON0**	**Analog output 0**
18	**24 VDC**	**24V source**
19	DIN0	Record selection bit 0
20	DIN2	Record selection bit 2
21	**DIN4**	**Output stage enable**
22	DIN6	Limit switch 0
23	DIN8	Start positioning
24	DOUT0	Controller ready
25	DOUT2	Start acknowledged

The connections implemented in the AGV for X1 are listed in bold in table Q.1; the rest are not connected. The analogue connection, AMON0, sends the degree position of the drives to the s7-1500 plc as this could not be done over the Profibus connection due to poor implementation by Festo. DIN13 (pin 15) was connected to 24 VDC to ensure the drive left STOP mode. The 24 VDC reference signal is provided by pin 18. DIN4 (pin 21) and DIN5 (pin 9) were implemented as part of the STO and are illustrated in figure 6.22.

X2: Encoder

The encoder connection uses a standard DE9 connector ("serial port"). The pinout is listed in table Q.2.

Table Q.2: Festo CMMST connections: X1

Pin	Signal	Description
1	A+	Encoder A phase positive
2	B+	Encoder B phase positive
3	N+	Encoder N phase positive
4	GND	Encoder power GND
5	VCC	Encoder power 5V
6	A-	Encoder A phase negative
7	B-	Encoder B phase negative
8	N-	Encoder N phase negative
9	GND	Encoder power GND

X3 STO Interface

The STO interface is implimented as illustrated in figure 6.22. The screw terminal connector is illustrated in figure Q.1 while the pinout is listed in table Q.3.

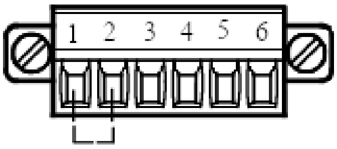


Figure Q.1: Stepper STO Connector

Table Q.3: Festo CMMST connections: X1

Pin	Signal	Description
1	24V	Power 24V output
2	Rel	Drive supply relay control
3	0V	Power 0V
4	n.c.	Not connected
5	STO1	STO input 1
6	STO2	STO input 2

X5 Serial Interface

The serial interface uses the RS233 protocol for commissioning and parametrizing the drive. Since RS232 serial connections are a standard connection with a standard pinout, the pinout will not be listed in this thesis. This plug uses a DE9 connector (sometimes erroneously called a DB9).

X6 Motor Connection

The motor connection for the Festo stepper is made drive side via a Phoenix Contact MSTB 2.5/8-G.08 BK socket and Phoenix Contact MSTB 2.5/8-ST5.08 BK plug. The wiring and pinout are illustrated in figure Q.2, while the pinout description is listed in table Q.4.

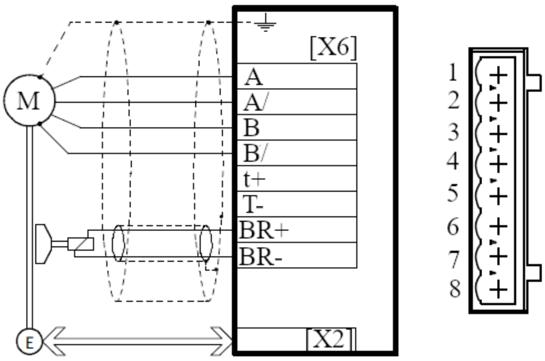


Figure Q.2: Stepper Motor Connector

Table Q.4: Festo CMMST connections: X6

Pin	Signal	Description
1	A	Stepper phase A +
2	A/	Stepper phase A -
3	B	Stepper phase B +
4	B/	Stepper phase B -
5	t+	Temperature sensor power
6	T-	Temperature sensor signal
7	BR+	Brake +
8	BR-	Brake -

X9 Power Connections

Although the power connector pinout is self-explanatory, it was explicitly included in this report due to the unique way Festo implemented the 48VDC and 24VDC connections. The connector used for the drive's power is a Phoenix Contact - MSTB 2.5/3-G-5.08 BK socket and Phoenix Contact - MSTB 2.5/3-ST-5.08 BK plug pair. This connection is illustrated in figure Q.4 and has its pinouts listed in table Q.5.

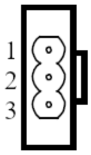


Figure Q.3: Stepper Power Connector

Table Q.5: Festo CMMST connections: X9

Pin	Signal	Description
1	ZK+	48V power
2	24V	24V power
3	0V	Common DC ground

As can be seen in figure Q.4, there are two positive DC sources required; however, there is only one connection allocated to 0V/ ground. Thus, the two power sources will share a common ground, as illustrated in figure Q.4.

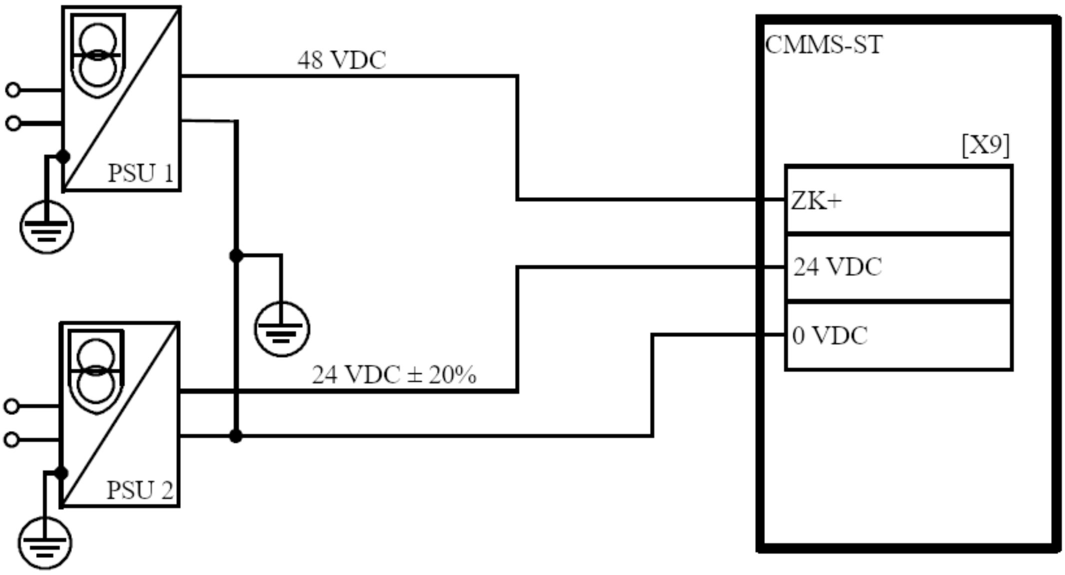


Figure Q.4: CMMST Common Ground Configuration[1]

[1] Image adapted from Festo [76].

R Appendix - TIA Portal Safety Report

Table of contents

Safety Administration

Safety summary

General information

F-signatures	
Collective F-signature	42AB4D7A
Software F-signature	502B7B35
Hardware F-signature	F27FD245
Communication address F-signature	----
Current compilation	
Safety program state	The offline safety program is consistent.
Compilation time	11/26/2021 12:20:51 PM (UTC +2:00)
Used versions	
STEP 7	STEP 7 Professional V15.1 Update 5
Safety	STEP 7 Safety V15.1
Access protection	
Safety program	The safety program is protected by password
F-CPU	Full access with fail-safe (no protection)

Notes

Location	Note	Additional info
Hardware configuration	The 'F-DQ 4x24VDC/2A PM HF_2' F-module has been configured in the hardware configuration but is not used in the safety program.	
General information	The response time of your safety function also depends on the cycle time of the F-OB and the runtime of the F-runtime group. When using distributed F-I/O modules, the response time also depends on the PROFINET/PROFIBUS parameter assignment. The configuration and parameter assignment of the standard system also has an effect on the response time of your safety function. Note that the configuration and parameter assignment of the standard system is not subject to the access protection of the safety program and does not change the F collective signature.	Note the warning "S085" in the manual and in the STEP 7 Safety online help.

Safety program settings

Safety mode can be disabled	No
Assignment of F-system block numbers	F-system managed
Safety system version	V2.2
Variable F-communication IDs enabled	No

System library elements used in safety program

Instructions (optional package STEP 7 Safety)	
Name	**Used version**
ESTOP1	V1.6

Information on F-runtime group

RTG1

Fail-safe organization block	
Name	Safety Main [OB123]
Event class	Cyclic interrupt
Cycle time	100000 µs
Phase shift	0 µs
Priority	12

Main safety block	
Name	Safety Main RTG [FB0]
I-DB for main safety block	Safety Main RTG iDB [DB1]

F-runtime group parameters	
Name	F-runtime group 1
Warn cycle time of the F-runtime group	130000 µs
Maximum cycle time of the F-runtime group	140000 µs
DB for F-runtime group communication	--
F-runtime group information DB	RTG1SysInfo

Pre/Post processing	
FC for pre processing	--
FC for post processing	--

F-blocks in safety program

Block name [Block number]	Function in safety program	Used and compiled in F-RTG	Signature
Safety Main [OB123]	F-OB [system-protected]	RTG1	42C4BC03
▼ 03. Safety			
Safety Main RTG [FB0]	F-FB	RTG1	246B2EE4
Safety Main RTG iDB [DB1]	F-IDB	RTG1	86EB4BB1

Know-how protected F-blocks in the safety program

The safety program does not include know-how protected F-blocks.

F-compliant PLC data types in the safety program

The safety program contains no F-compliant PLC data types (UDT).

Data from the standard user program

Absolute address	Symbolic operand	F-runtime group	Block name [Block number]	Network
--	"Inputs GDB"."Ack Pendant"	RTG1	Safety Main RTG [FB0]	1
--	"Inputs GDB"."Ack AGV"	RTG1	Safety Main RTG [FB0]	1

Parameters for safety-related CPU-CPU communications via RCV_DP, SEND_DP

No safety-related CPU-CPU communication via RCV_DP, SEND_DP is configured.

Communications via Flexible F-Link

No communications via Flexible F-Link are defined for the F-Program.

Hardware configuration of F-I/O

F-CPU information	
Short designation	CPU 1512SP F-1 PN
Article number	6ES7 512-1SK01-0AB0
Firmware version	V2.1
Central F-source address	1
F-destination address range (PROFIsafe address type 1)	--
F-destination address range (PROFIsafe address type 2)	65532 .. 65534

Central periphery

Rail - Slot	Module	Start address	F-destination address	F-monitoring time	Parameter signature (w/o addresses)
Rack_0-9	6ES7 136-6BA00-0CA0 F-DI 8x24VDC HF_1	4	65534	150 ms	0x36AA (13994)
Rack_0-10	6ES7 136-6DB00-0CA0 F-DQ 4x24VDC/2A PM HF_1	10	65533	150 ms	0xB33 (2867)
Rack_0-11	6ES7 136-6DB00-0CA0 F-DQ 4x24VDC/2A PM HF_2	15	65532	150 ms	0x61BD (25021)

F-DI 8x24VDC HF_1 : Central I/O Rack_0, Slot 9

General parameters

Hardware	
Name	F-DI 8x24VDC HF_1
Slot	9
Short designation	F-DI 8x24VDC HF
Article number	6ES7 136-6BA00-0CA0
Start address input	4
Start address output	4
Hardware identifier	263
F-monitoring time	150 ms
F-source address	1
F-destination address	65534
F-parameter signature (without addresses)	0x36AA (13994)
F-parameter signature (with addresses)	0xCE35 (52789)
Behavior after channel fault	Passivate channel
RIOforFA-Safety	No
PROFIsafe mode	V2 mode
PROFIsafe protocol version	Loop-back extension (LP)
Firmware version	V1.0
Software	
F-I/O DB number	30002
F I/O DB name	F00004_F-DI8x24VDCHF_1
Used in F-runtime group	RTG1

Specific Parameters

Sensor supply 0	
Short-circuit test	Yes
Time for short-circuit test	4.2 ms
Startup time of sensor after short-circuit test	4.2 ms
Sensor supply 1	
Short-circuit test	Yes
Time for short-circuit test	4.2 ms
Startup time of sensor after short-circuit test	4.2 ms
Sensor supply 2	
Short-circuit test	Yes
Time for short-circuit test	4.2 ms
Startup time of sensor after short-circuit test	4.2 ms
Sensor supply 3	
Short-circuit test	Yes
Time for short-circuit test	4.2 ms
Startup time of sensor after short-circuit test	4.2 ms
Sensor supply 4	
Short-circuit test	Yes
Time for short-circuit test	4.2 ms
Startup time of sensor after short-circuit test	4.2 ms
Sensor supply 5	
Short-circuit test	Yes
Time for short-circuit test	4.2 ms
Startup time of sensor after short-circuit test	4.2 ms
Sensor supply 6	
Short-circuit test	Yes
Time for short-circuit test	4.2 ms
Startup time of sensor after short-circuit test	4.2 ms
Sensor supply 7	
Short-circuit test	Yes
Time for short-circuit test	4.2 ms
Startup time of sensor after short-circuit test	4.2 ms
Channel 0, 4	
Sensor evaluation	1oo2 evaluation, equivalent
Discrepancy behavior	Supply value 0
Discrepancy time	50 ms
Reintegration after discrepancy error	Test 0-Signal not necessary
Channel 0	
Activated	Yes
Sensor supply	Sensor supply 0
Input delay	3,2 ms
Chatter monitoring	No
Number of signal changes	5
Monitoring window	2 sec
Channel 4	
Activated	Yes
Sensor supply	Sensor supply 4
Input delay	3,2 ms
Chatter monitoring	No
Number of signal changes	5
Monitoring window	2 sec
Channel 1, 5	
Sensor evaluation	1oo2 evaluation, equivalent
Discrepancy behavior	Supply value 0
Discrepancy time	50 ms
Reintegration after discrepancy error	Test 0-Signal not necessary
Channel 1	
Activated	Yes
Sensor supply	Sensor supply 1
Input delay	3,2 ms
Chatter monitoring	No
Number of signal changes	5
Monitoring window	2 sec
Channel 5	
Activated	Yes
Sensor supply	Sensor supply 5
Input delay	3,2 ms
Chatter monitoring	No
Number of signal changes	5
Monitoring window	2 sec
Channel 2, 6	
Sensor evaluation	1oo2 evaluation, equivalent
Discrepancy behavior	Supply value 0
Discrepancy time	50 ms

General parameters		Specific Parameters	
		Reintegration after discrepancy error	Test 0-Signal not necessary
		Channel 2	
		Activated	Yes
		Sensor supply	Sensor supply 2
		Input delay	3,2 ms
		Chatter monitoring	No
		Number of signal changes	5
		Monitoring window	2 sec
		Channel 6	
		Activated	Yes
		Sensor supply	Sensor supply 6
		Input delay	3,2 ms
		Chatter monitoring	No
		Number of signal changes	5
		Monitoring window	2 sec
		Channel 3, 7	
		Sensor evaluation	1oo2 evaluation, equivalent
		Discrepancy behavior	Supply value 0
		Discrepancy time	50 ms
		Reintegration after discrepancy error	Test 0-Signal not necessary
		Channel 3	
		Activated	Yes
		Sensor supply	Sensor supply 3
		Input delay	3,2 ms
		Chatter monitoring	No
		Number of signal changes	5
		Monitoring window	2 sec
		Channel 7	
		Activated	Yes
		Sensor supply	Sensor supply 7
		Input delay	3,2 ms
		Chatter monitoring	No
		Number of signal changes	5
		Monitoring window	2 sec

F-DQ 4x24VDC/2A PM HF_1 : Central I/O Rack_0, Slot 10

General parameters		Specific Parameters	
Hardware		Maximum test period	1000 sec
Name	F-DQ 4x24VDC/2A PM HF_1	**Channel 0**	
Slot	10	Activated	Yes
Short designation	F-DQ 4x24VDC/2A PM HF	Max. readback time dark test	1.0 ms
Article number	6ES7 136-6DB00-0CA0	Max. readback time switch on test	2.0 ms
Start address input	10	Activated light test	No
Start address output	10	Diagnosis: Wire break	No
Hardware identifier	264	**Channel 1**	
F-monitoring time	150 ms	Activated	Yes
F-source address	1	Max. readback time dark test	1.0 ms
F-destination address	65533	Max. readback time switch on test	2.0 ms
F-parameter signature (without addresses)	0xB33 (2867)	Activated light test	No
F-parameter signature (with addresses)	0x9E08 (40456)	Diagnosis: Wire break	No
Behavior after channel fault	Passivate channel	**Channel 2**	
RIOforFA-Safety	No	Activated	Yes
PROFIsafe mode	V2 mode	Max. readback time dark test	1.0 ms
PROFIsafe protocol version	Loop-back extension (LP)	Max. readback time switch on test	2.0 ms
Firmware version	V1.0	Activated light test	No
Software		Diagnosis: Wire break	No
F-I/O DB number	30003	**Channel 3**	
F I/O DB name	F00010_F-DQ4x24VDC/2APMHF_1	Activated	Yes
Used in F-runtime group	RTG1	Max. readback time dark test	1.0 ms
		Max. readback time switch on test	2.0 ms
		Activated light test	No
		Diagnosis: Wire break	No

F-DQ 4x24VDC/2A PM HF_2 : Central I/O Rack_0, Slot 11

General parameters

Hardware	
Name	F-DQ 4x24VDC/2A PM HF_2
Slot	11
Short designation	F-DQ 4x24VDC/2A PM HF
Article number	6ES7 136-6DB00-0CA0
Start address input	15
Start address output	15
Hardware identifier	265
F-monitoring time	150 ms
F-source address	1
F-destination address	65532
F-parameter signature (without addresses)	0x61BD (25021)
F-parameter signature (with addresses)	0x746F (29807)
Behavior after channel fault	Passivate channel
RIOforFA-Safety	No
PROFIsafe mode	V2 mode
PROFIsafe protocol version	Loop-back extension (LP)
Firmware version	V1.0
Software	
F-I/O DB number	30004
F I/O DB name	F00015_F-DQ4x24VDC/2APMHF_2
Used in F-runtime group	No

Specific Parameters

Maximum test period	1000 sec
Channel 0	
Activated	No
Max. readback time dark test	1.0 ms
Max. readback time switch on test	0.6 ms
Activated light test	No
Diagnosis: Wire break	No
Channel 1	
Activated	No
Max. readback time dark test	1.0 ms
Max. readback time switch on test	0.6 ms
Activated light test	No
Diagnosis: Wire break	No
Channel 2	
Activated	No
Max. readback time dark test	1.0 ms
Max. readback time switch on test	0.6 ms
Activated light test	No
Diagnosis: Wire break	No
Channel 3	
Activated	No
Max. readback time dark test	1.0 ms
Max. readback time switch on test	0.6 ms
Activated light test	No
Diagnosis: Wire break	No

Supplementary information

Print created on	12/8/2021 8:38:56 AM (UTC +2:00)	**Page numbers for safety summary**	From 3 - 1 to 3 - 9

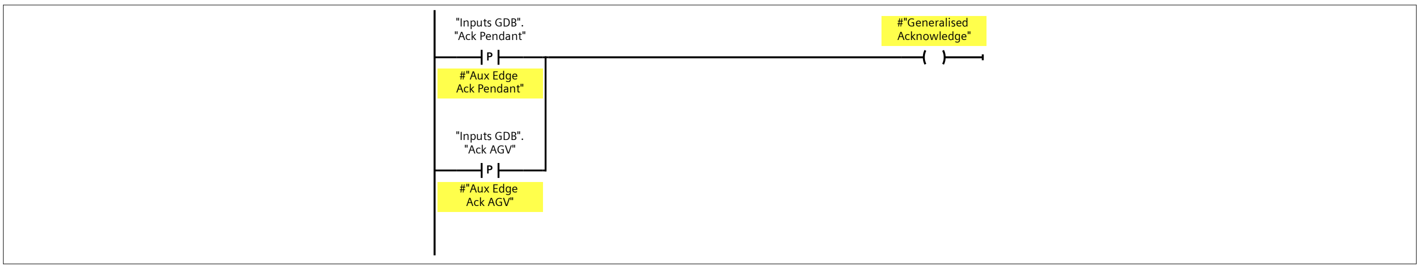
Safety Administration / Fail-safe user blocks

Safety Main RTG

Safety Main RTG Properties							
General							
Name	Safety Main RTG	**Number**	0	**Type**	FB	**Language**	LAD
Numbering	Manual						
Information							
Title		**Author**		**Comment**	Password: "Siemens1200"	**Family**	
Version	0.1	**User-defined ID**					

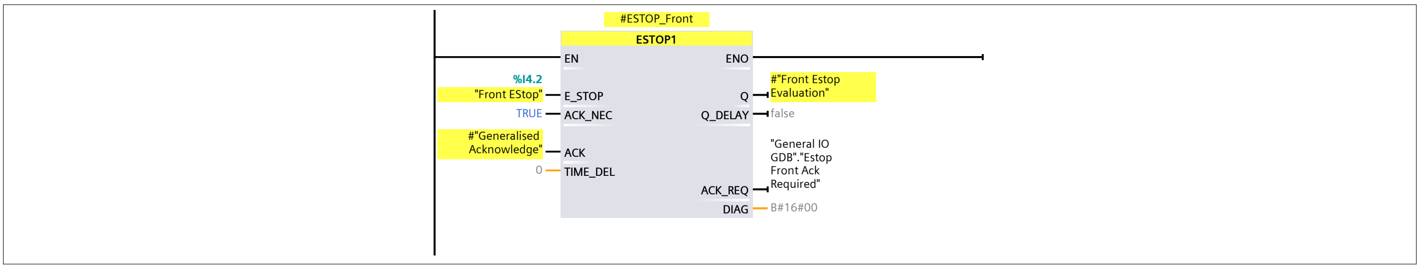
Name	Data type	Default value	Retain	Accessible from HMI/OPC UA	Writable from HMI/OPC UA	Visible in HMI engineering	Setpoint	Supervision	Comment
Input									
Output									
InOut									
▼ Static									
▼ ESTOP_Front	ESTOP1			True	True	True	True		
▼ Input									
E_STOP	Bool	false	Non-retain	True	True	True	False		Emercency STOP
ACK_NEC	Bool	true	Non-retain	True	True	True	False		1=Acknowledgment necessary
ACK	Bool	false	Non-retain	True	True	True	False		1=Acknowledgment
TIME_DEL	Time	0	Non-retain	True	True	True	False		Time delay
▼ Output									
Q	Bool	false	Non-retain	True	True	True	False		1=Enable
Q_DELAY	Bool	false	Non-retain	True	True	True	False		Enable is OFF delayed
ACK_REQ	Bool	false	Non-retain	True	True	True	False		1=acknowledgment request
DIAG	Byte	B#16#00	Non-retain	True	True	True	False		Service information
InOut									
Static									
▼ ESTOP_Rear	ESTOP1			True	True	True	True		
▼ Input									
E_STOP	Bool	false	Non-retain	True	True	True	False		Emercency STOP
ACK_NEC	Bool	true	Non-retain	True	True	True	False		1=Acknowledgment necessary
ACK	Bool	false	Non-retain	True	True	True	False		1=Acknowledgment
TIME_DEL	Time	0	Non-retain	True	True	True	False		Time delay
▼ Output									
Q	Bool	false	Non-retain	True	True	True	False		1=Enable
Q_DELAY	Bool	false	Non-retain	True	True	True	False		Enable is OFF delayed
ACK_REQ	Bool	false	Non-retain	True	True	True	False		1=acknowledgment request
DIAG	Byte	B#16#00	Non-retain	True	True	True	False		Service information
InOut									
Static									
Front Estop Evaluation	Bool	false	Non-retain	True	True	True	True		
Rear Estop Evaluation	Bool	false	Non-retain	True	True	True	False		
Aux Edge Ack Pendant	Bool	false	Non-retain	True	True	True	False		
Aux Edge Ack AGV	Bool	false	Non-retain	True	True	True	False		
Generalised Acknowledge	Bool	false	Non-retain	True	True	True	False		
▼ Temp									
test	Bool								
Constant									

Network 1: Acknowledge Faults

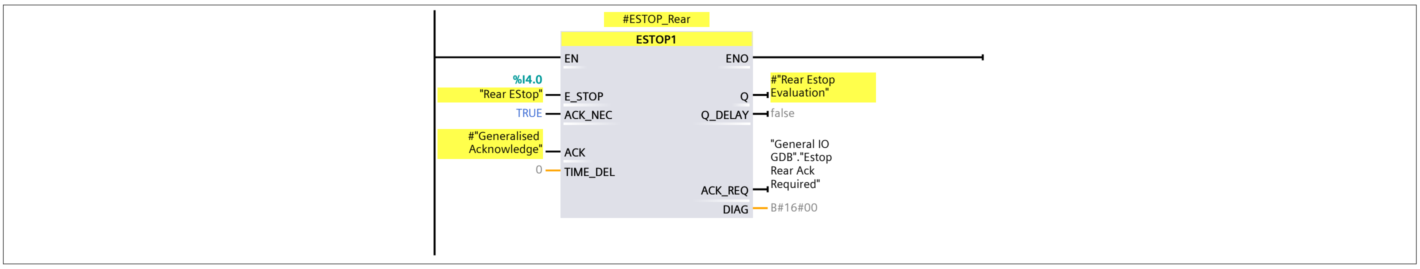


Network 2: Front Estop System

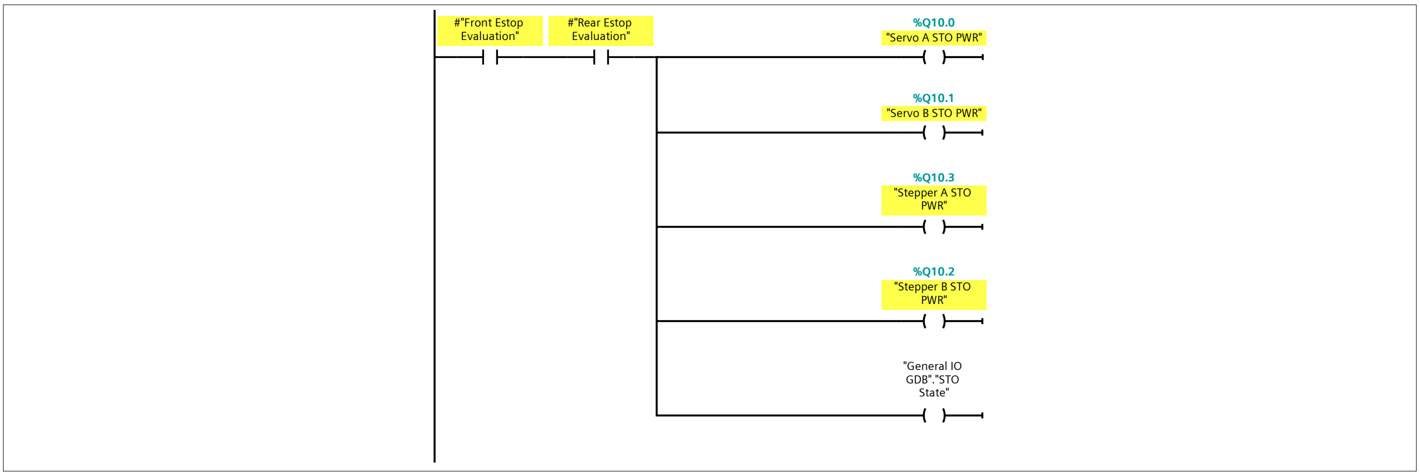
Safety information: 42AB4D7A Consistent; STEP 7 Safety V15.1;



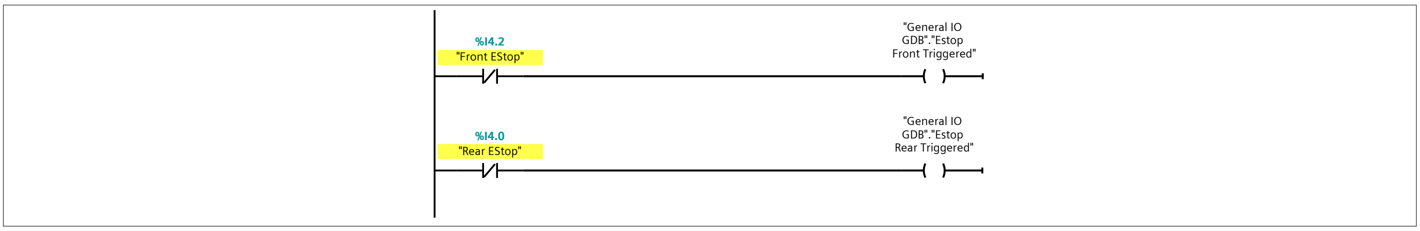
Network 3: Rear Estop System



Network 4: Servo & Stepper Motor STO



Network 5: Register E-Stop Trigger State


%I4.2
"Front EStop"
"General IO GDB"."Estop Front Triggered"
%I4.0
"Rear EStop"
"General IO GDB"."Estop Rear Triggered"

Safety Administration / Fail-safe user blocks

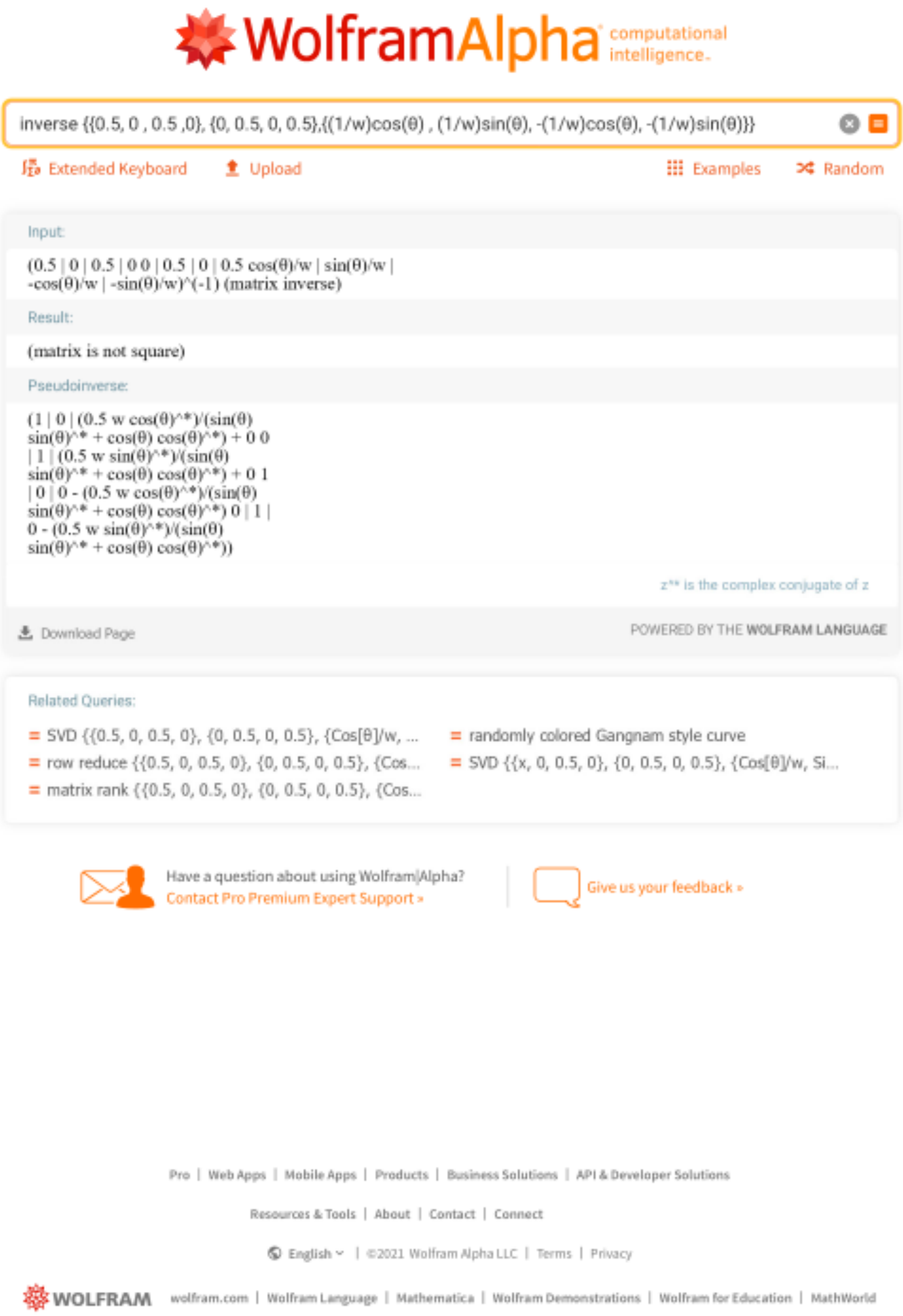
Safety Main RTG iDB

Safety Main RTG iDB Properties							
General							
Name	Safety Main RTG iDB	**Number**	1	**Type**	DB	**Language**	DB
Numbering	Automatic						
Information							
Title		**Author**		**Comment**		**Family**	
Version	0.1	**User-defined ID**	FUSI				

Name	Data type	Start value	Retain	Accessible from HMI/OPC UA	Writable from HMI/OPC UA	Visible in HMI engineering	Setpoint	Supervision	Comment
Input									
Output									
InOut									
▼ Static									
▼ ESTOP_Front	ESTOP1		False	True	True	True	True		
▼ Input									
E_STOP	Bool	false	False	True	True	True	False		Emercency STOP
ACK_NEC	Bool	true	False	True	True	True	False		1=Acknowledgment necessary
ACK	Bool	false	False	True	True	True	False		1=Acknowledgment
TIME_DEL	Time	0	False	True	True	True	False		Time delay
▼ Output									
Q	Bool	false	False	True	True	True	False		1=Enable
Q_DELAY	Bool	false	False	True	True	True	False		Enable is OFF delayed
ACK_REQ	Bool	false	False	True	True	True	False		1=acknowledgment request
DIAG	Byte	B#16#00	False	True	True	True	False		Service information
InOut									
Static									
▼ ESTOP_Rear	ESTOP1		False	True	True	True	True		
▼ Input									
E_STOP	Bool	false	False	True	True	True	False		Emercency STOP
ACK_NEC	Bool	true	False	True	True	True	False		1=Acknowledgment necessary
ACK	Bool	false	False	True	True	True	False		1=Acknowledgment
TIME_DEL	Time	0	False	True	True	True	False		Time delay
▼ Output									
Q	Bool	false	False	True	True	True	False		1=Enable
Q_DELAY	Bool	false	False	True	True	True	False		Enable is OFF delayed
ACK_REQ	Bool	false	False	True	True	True	False		1=acknowledgment request
DIAG	Byte	B#16#00	False	True	True	True	False		Service information
InOut									
Static									
Front Estop Evaluation	Bool	false	False	True	True	True	True		
Rear Estop Evaluation	Bool	false	False	True	True	True	False		
Aux Edge Ack Pendant	Bool	false	False	True	True	True	False		
Aux Edge Ack AGV	Bool	false	False	True	True	True	False		
Generalised Acknowledge	Bool	false	False	True	True	True	False		

Safety information: 42AB4D7A Consistent; STEP 7 Safety V15.1;

S Appendix - Wolfram|Alpha Captures


WolframAlpha computational intelligence.
Extended Keyboard
Upload
Examples
Random

inverse {{0.5, 0 , 0.5 ,0}, {0, 0.5, 0, 0.5},{(1/w)cos(θ) , (1/w)sin(θ), -(1/w)cos(θ), -(1/w)sin(θ)}}

Input:

(0.5 | 0 | 0.5 | 0 0 | 0.5 | 0 | 0.5 cos(θ)/w | sin(θ)/w |
-cos(θ)/w | -sin(θ)/w)^(-1) (matrix inverse)

Result:

(matrix is not square)

Pseudoinverse:

(1 | 0 | (0.5 w cos(θ)^*)/(sin(θ)
sin(θ)^* + cos(θ) cos(θ)^*) + 0 0
| 1 | (0.5 w sin(θ)^*)/(sin(θ)
sin(θ)^* + cos(θ) cos(θ)^*) + 0 1
| 0 | 0 - (0.5 w cos(θ)^*)/(sin(θ)
sin(θ)^* + cos(θ) cos(θ)^*) 0 | 1 |
0 - (0.5 w sin(θ)^*)/(sin(θ)
sin(θ)^* + cos(θ) cos(θ)^*))

z^* is the complex conjugate of z

Download Page
POWERED BY THE WOLFRAM LANGUAGE

Related Queries:

= SVD {{0.5, 0, 0.5, 0}, {0, 0.5, 0, 0.5}, {Cos[θ]/w, ...
= row reduce {{0.5, 0, 0.5, 0}, {0, 0.5, 0, 0.5}, {Cos...
= matrix rank {{0.5, 0, 0.5, 0}, {0, 0.5, 0, 0.5}, {Cos...
= randomly colored Gangnam style curve
= SVD {{x, 0, 0.5, 0}, {0, 0.5, 0, 0.5}, {Cos[θ]/w, Si...

Have a question about using Wolfram|Alpha?
Contact Pro Premium Expert Support »

Give us your feedback »

Pro | Web Apps | Mobile Apps | Products | Business Solutions | API & Developer Solutions
Resources & Tools | About | Contact | Connect
English | ©2021 Wolfram Alpha LLC | Terms | Privacy
WOLFRAM wolfram.com | Wolfram Language | Mathematica | Wolfram Demonstrations | Wolfram for Education | MathWorld

T Appendix - Siemens s7-1512SP PLC Code

T.1 Program Blocks

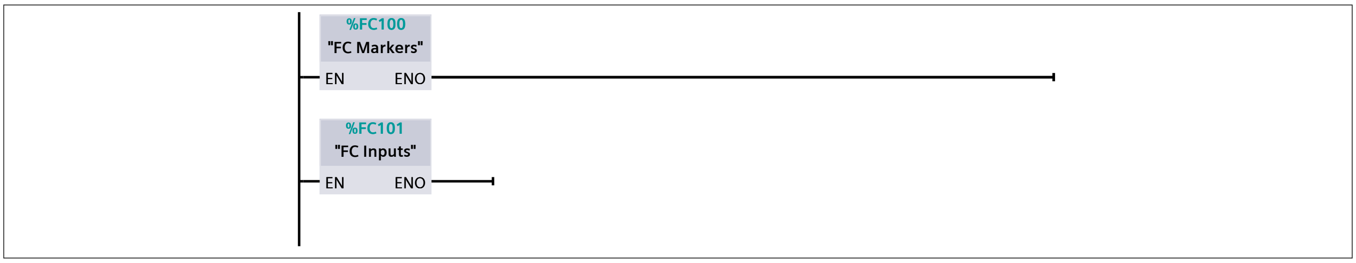
Code can be found on next page

Main [OB1]

Main Properties					
General					
Name	Main	**Number**	1	**Type**	OB
Language	LAD	**Numbering**	Automatic		
Information					
Title	"Main Program Sweep (Cycle)"	**Author**		**Comment**	Safety Block Password = "Siemens1200"
Family		**Version**	0.1	**User-defined ID**	

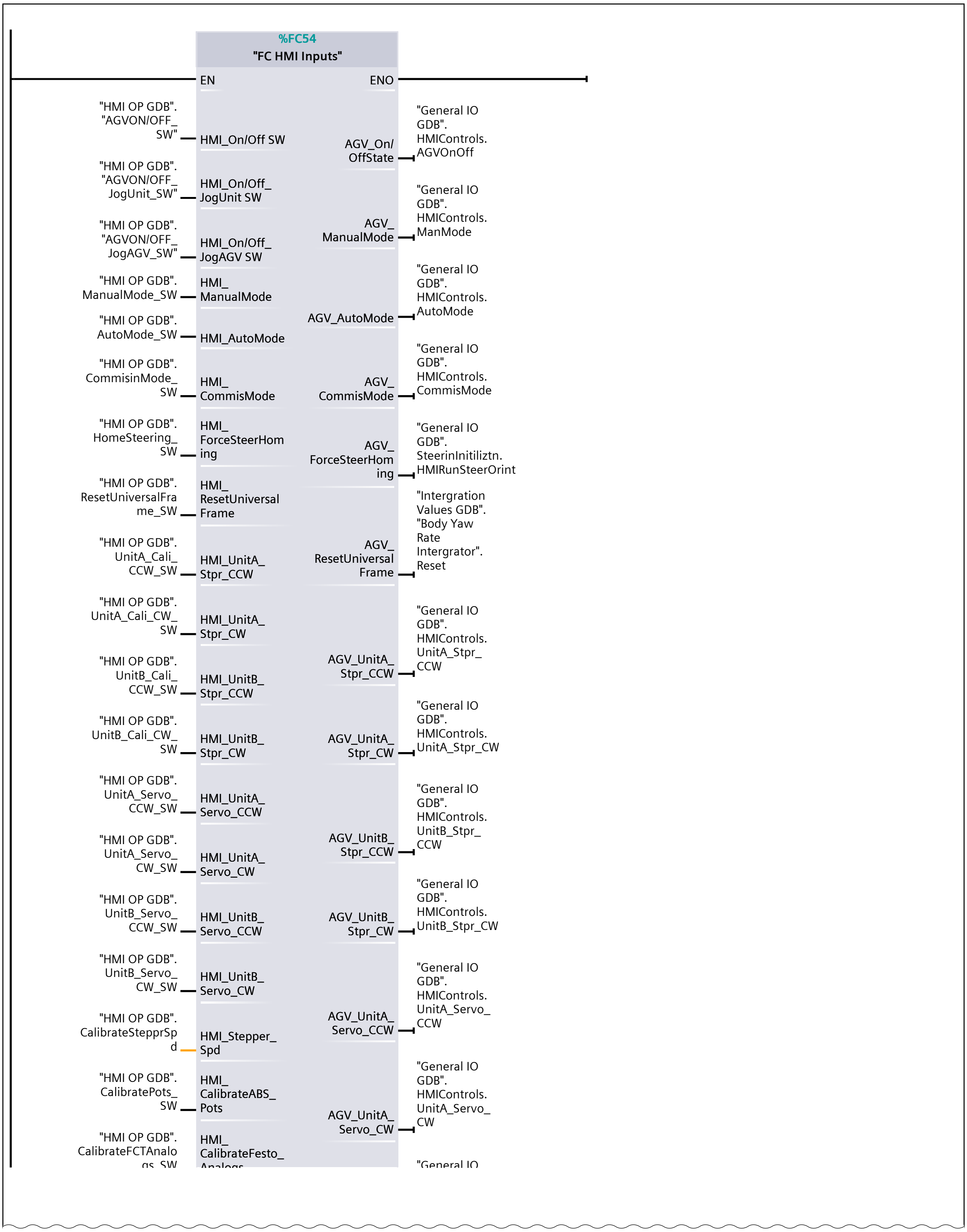
Name	Data type	Default value
▼ Input		
Initial_Call	Bool	
Remanence	Bool	
Temp		
Constant		

Network 1: Inputs Control



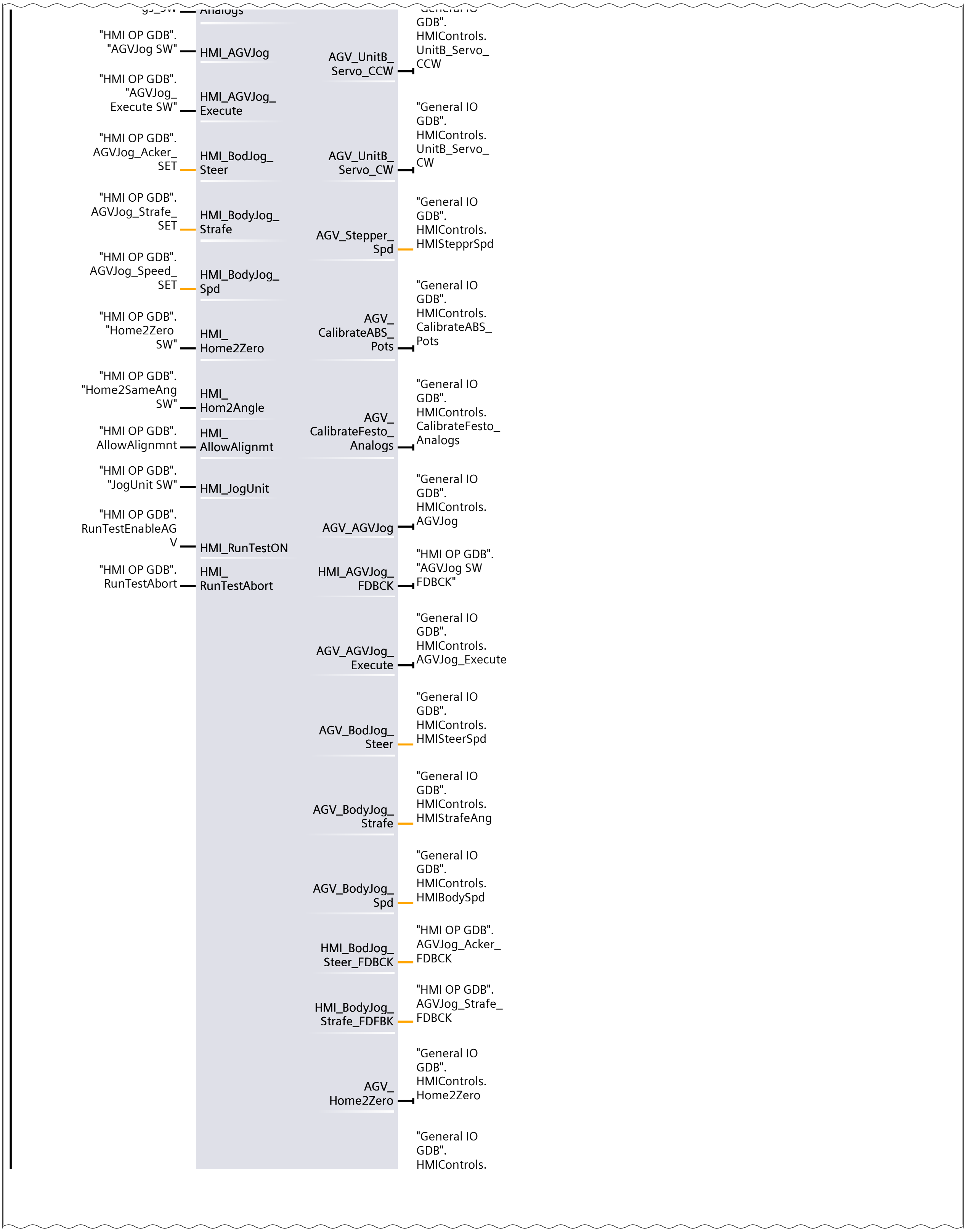
Network 2: HMI Inputs HMI ---> AGV

Network 2: HMI Inputs HMI ---> AGV (1.1 / 3.1)


%FC54
"FC HMI Inputs"
EN
ENO
"HMI OP GDB". "AGVON/OFF_ SW"
HMI_On/Off SW
"General IO GDB". HMIControls. AGVOnOff
AGV_On/ OffState
"HMI OP GDB". "AGVON/OFF_ JogUnit_SW"
HMI_On/Off_ JogUnit SW
"General IO GDB". HMIControls. ManMode
AGV_ ManualMode
"HMI OP GDB". "AGVON/OFF_ JogAGV_SW"
HMI_On/Off_ JogAGV SW
"HMI OP GDB". ManualMode_SW
HMI_ ManualMode
"General IO GDB". HMIControls. AutoMode
AGV_AutoMode
"HMI OP GDB". AutoMode_SW
HMI_AutoMode
"General IO GDB". HMIControls. CommisMode
"HMI OP GDB". CommisinMode_ SW
HMI_ CommisMode
AGV_ CommisMode
"General IO GDB". SteerinInitiliztn. HMIRunSteerOrint
"HMI OP GDB". HomeSteering_ SW
HMI_ ForceSteerHom ing
AGV_ ForceSteerHom ing
"Intergration Values GDB". "Body Yaw Rate Intergrator". Reset
"HMI OP GDB". ResetUniversalFra me_SW
HMI_ ResetUniversal Frame
AGV_ ResetUniversal Frame
"HMI OP GDB". UnitA_Cali_ CCW_SW
HMI_UnitA_ Stpr_CCW
"HMI OP GDB". UnitA_Cali_CW_ SW
HMI_UnitA_ Stpr_CW
"General IO GDB". HMIControls. UnitA_Stpr_ CCW
AGV_UnitA_ Stpr_CCW
"HMI OP GDB". UnitB_Cali_ CCW_SW
HMI_UnitB_ Stpr_CCW
"General IO GDB". HMIControls. UnitA_Stpr_CW
"HMI OP GDB". UnitB_Cali_CW_ SW
HMI_UnitB_ Stpr_CW
AGV_UnitA_ Stpr_CW
"General IO GDB". HMIControls. UnitB_Stpr_ CCW
"HMI OP GDB". UnitA_Servo_ CCW_SW
HMI_UnitA_ Servo_CCW
AGV_UnitB_ Stpr_CCW
"HMI OP GDB". UnitA_Servo_ CW_SW
HMI_UnitA_ Servo_CW
"General IO GDB". HMIControls. UnitB_Stpr_CW
"HMI OP GDB". UnitB_Servo_ CCW_SW
HMI_UnitB_ Servo_CCW
AGV_UnitB_ Stpr_CW
"General IO GDB". HMIControls. UnitA_Servo_ CCW
"HMI OP GDB". UnitB_Servo_ CW_SW
HMI_UnitB_ Servo_CW
AGV_UnitA_ Servo_CCW
"HMI OP GDB". CalibrateSteppr Sp d
HMI_Stepper_ Spd
"General IO GDB". HMIControls. UnitA_Servo_ CW
"HMI OP GDB". CalibratePots_ SW
HMI_ CalibrateABS_ Pots
AGV_UnitA_ Servo_CW
"HMI OP GDB". CalibrateFCTAnalo gs_SW
HMI_ CalibrateFesto_ Analogs
"General IO

Network 2: HMI Inputs HMI ---> AGV (2.1 / 3.1)

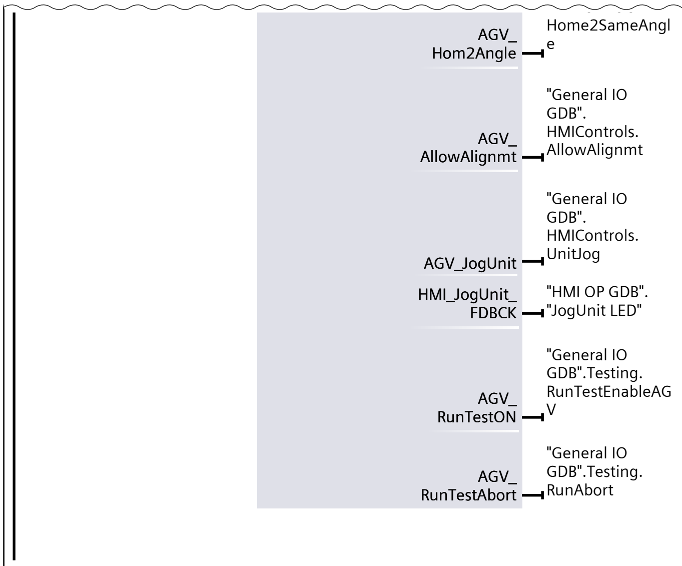
1.1 (Page2 - 2)


gs_SW"
Analogs
"HMI OP GDB". "AGVJog SW"
HMI_AGVJog
"HMI OP GDB". "AGVJog_ Execute SW"
HMI_AGVJog_ Execute
"HMI OP GDB". AGVJog_Acker_ SET
HMI_BodJog_ Steer
"HMI OP GDB". AGVJog_Strafe_ SET
HMI_BodyJog_ Strafe
"HMI OP GDB". AGVJog_Speed_ SET
HMI_BodyJog_ Spd
"HMI OP GDB". "Home2Zero SW"
HMI_ Home2Zero
"HMI OP GDB". "Home2SameAng SW"
HMI_ Hom2Angle
"HMI OP GDB". AllowAlignmnt
HMI_ AllowAlignmt
"HMI OP GDB". "JogUnit SW"
HMI_JogUnit
"HMI OP GDB". RunTestEnableAG V
HMI_RunTestON
"HMI OP GDB". RunTestAbort
HMI_ RunTestAbort
"General IO GDB". HMIControls. UnitB_Servo_ CCW
AGV_UnitB_ Servo_CCW
"General IO GDB". HMIControls. UnitB_Servo_ CW
AGV_UnitB_ Servo_CW
"General IO GDB". HMIControls. HMISteprSpd
AGV_Stepper_ Spd
"General IO GDB". HMIControls. CalibrateABS_ Pots
AGV_ CalibrateABS_ Pots
"General IO GDB". HMIControls. CalibrateFesto_ Analogs
AGV_ CalibrateFesto_ Analogs
"General IO GDB". HMIControls. AGVJog
AGV_AGVJog
"HMI OP GDB". "AGVJog SW FDBCK"
HMI_AGVJog_ FDBCK
"General IO GDB". HMIControls. AGVJog_Execute
AGV_AGVJog_ Execute
"General IO GDB". HMIControls. HMISteerSpd
AGV_BodJog_ Steer
"General IO GDB". HMIControls. HMIStrafeAng
AGV_BodyJog_ Strafe
"General IO GDB". HMIControls. HMIBodySpd
AGV_BodyJog_ Spd
"HMI OP GDB". AGVJog_Acker_ FDBCK
HMI_BodJog_ Steer_FDBCK
"HMI OP GDB". AGVJog_Strafe_ FDBCK
HMI_BodyJog_ Strafe_FDFBK
"General IO GDB". HMIControls. Home2Zero
AGV_ Home2Zero
"General IO GDB". HMIControls.

3.1 (Page2 - 4)

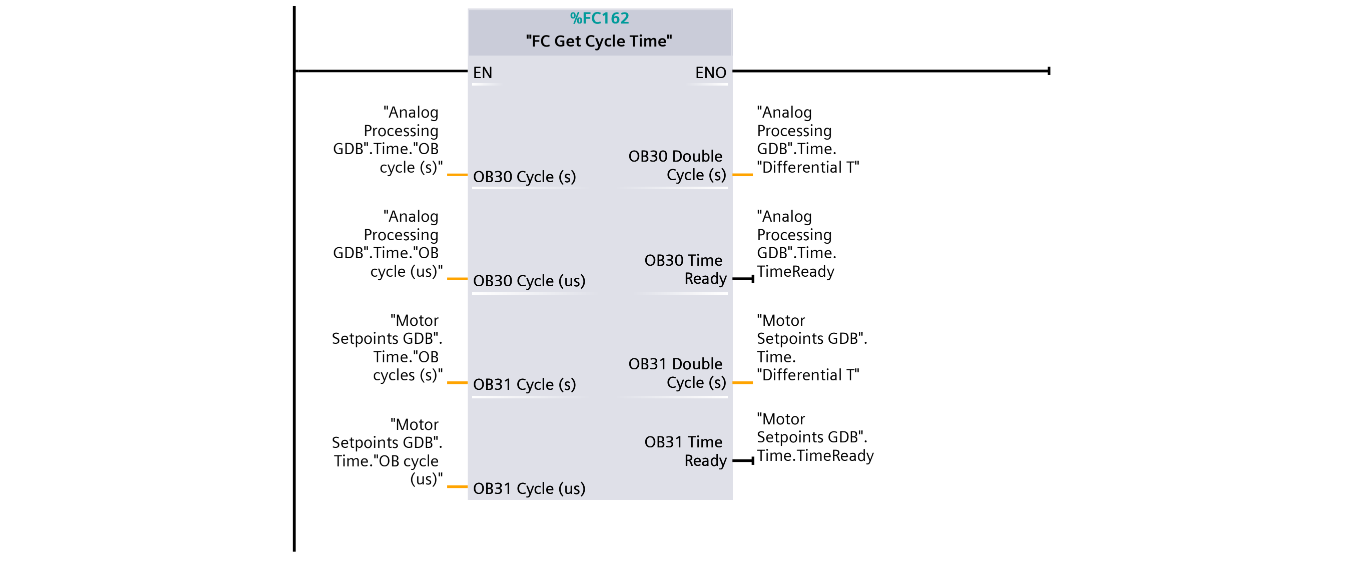
Network 2: HMI Inputs HMI ---> AGV (3.1 / 3.1)

2.1 (Page2 - 3)

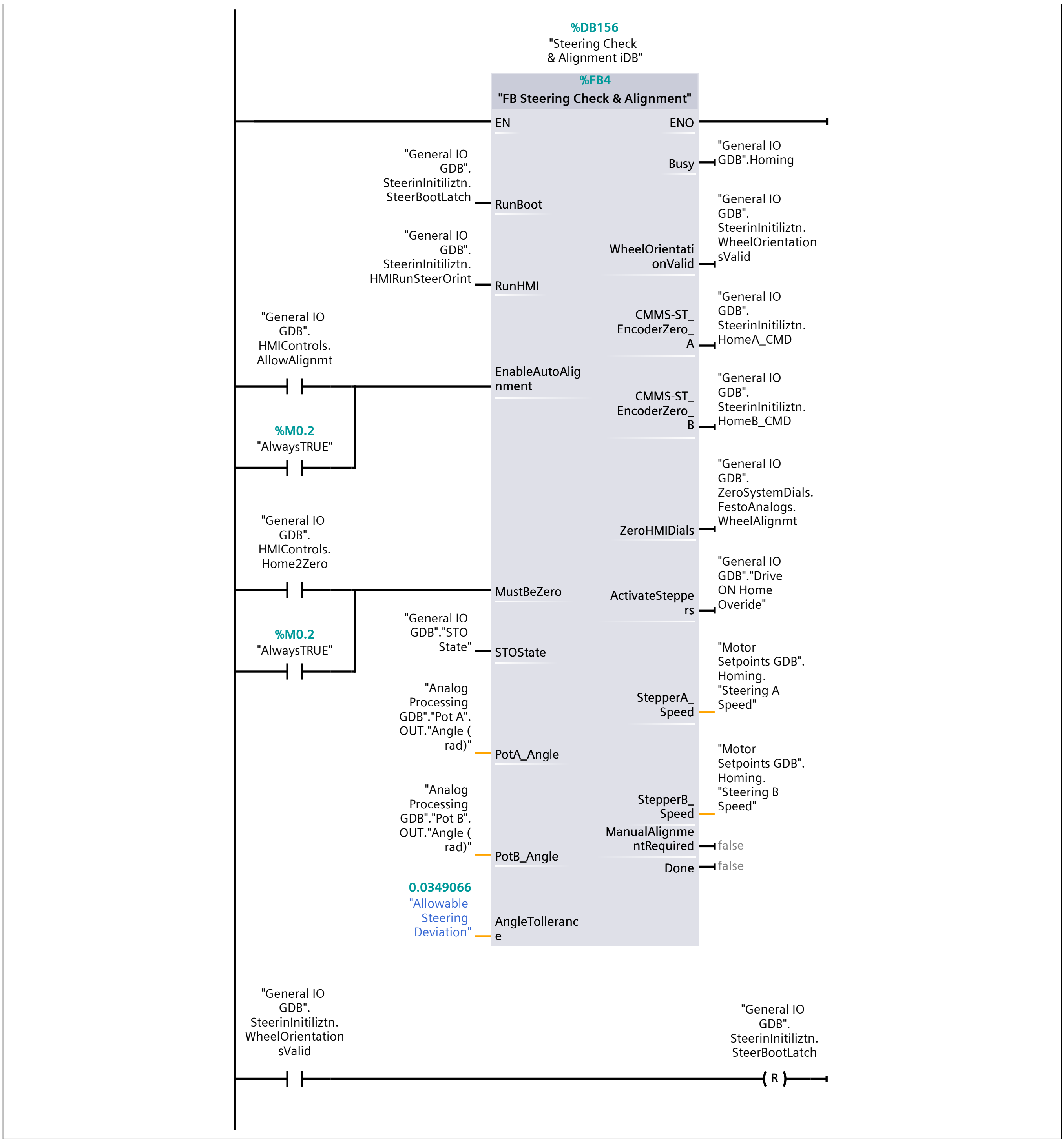


Network 3: Read Cycle time of cyclic interupts

Read the cycle time of the OB that samples the potentiometers, this is done incase the cyceletime is changed at a later state and the programmer forgets to change the time interval in the numerical anaylsis that performs differentiation in the pot OB

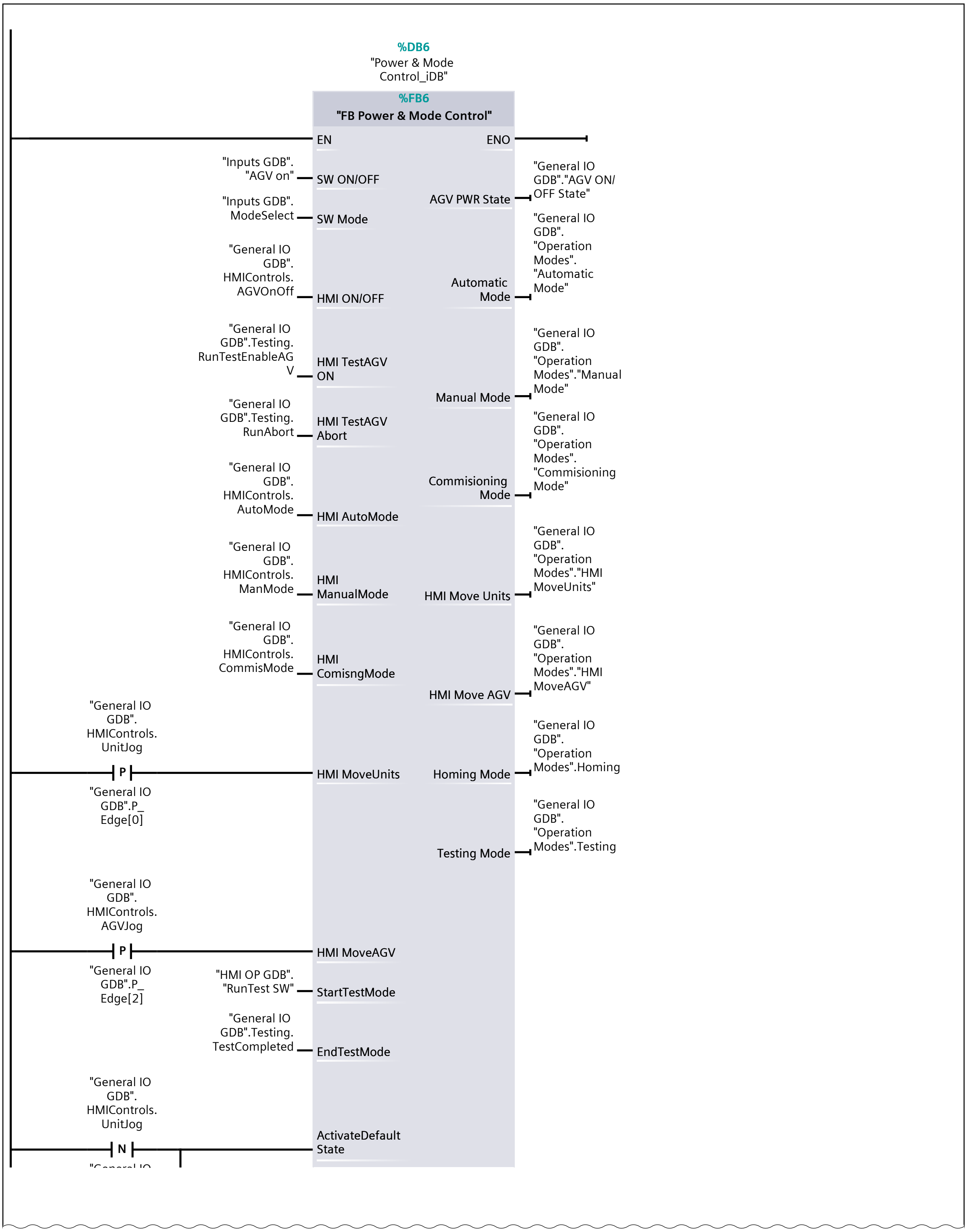

%FC162
"FC Get Cycle Time"
EN
ENO
"Analog Processing GDB".Time."OB cycle (s)"
OB30 Cycle (s)
OB30 Double Cycle (s)
"Analog Processing GDB".Time."Differential T"
"Analog Processing GDB".Time."OB cycle (us)"
OB30 Cycle (us)
OB30 Time Ready
"Analog Processing GDB".Time.TimeReady
"Motor Setpoints GDB".Time."OB cycles (s)"
OB31 Cycle (s)
OB31 Double Cycle (s)
"Motor Setpoints GDB".Time."Differential T"
"Motor Setpoints GDB".Time."OB cycle (us)"
OB31 Cycle (us)
OB31 Time Ready
"Motor Setpoints GDB".Time.TimeReady

Network 4: Home Steering System


%DB156
"Steering Check & Alignment iDB"
%FB4
"FB Steering Check & Alignment"
EN
ENO
"General IO GDB".Homing
Busy
"General IO GDB". SteerinInitiliztn. SteerBootLatch
RunBoot
"General IO GDB". SteerinInitiliztn. WheelOrientationsValid
WheelOrientationValid
"General IO GDB". SteerinInitiliztn. HMIRunSteerOrint
RunHMI
"General IO GDB". SteerinInitiliztn. HomeA_CMD
CMMS-ST_EncoderZero_A
"General IO GDB". HMIControls. AllowAlignmt
EnableAutoAlignment
%M0.2
"AlwaysTRUE"
"General IO GDB". SteerinInitiliztn. HomeB_CMD
CMMS-ST_EncoderZero_B
"General IO GDB". ZeroSystemDials. FestoAnalogs. WheelAlignmt
ZeroHMIDials
"General IO GDB". HMIControls. Home2Zero
MustBeZero
"General IO GDB"."Drive ON Home Overide"
ActivateSteppers
%M0.2
"AlwaysTRUE"
"General IO GDB"."STO State"
STOState
"Motor Setpoints GDB". Homing. "Steering A Speed"
StepperA_Speed
"Analog Processing GDB"."Pot A". OUT."Angle (rad)"
PotA_Angle
"Motor Setpoints GDB". Homing. "Steering B Speed"
StepperB_Speed
ManualAlignmentRequired
false
"Analog Processing GDB"."Pot B". OUT."Angle (rad)"
PotB_Angle
Done
false
0.0349066
"Allowable Steering Deviation"
AngleTollerance
"General IO GDB". SteerinInitiliztn. WheelOrientationsValid
"General IO GDB". SteerinInitiliztn. SteerBootLatch
R

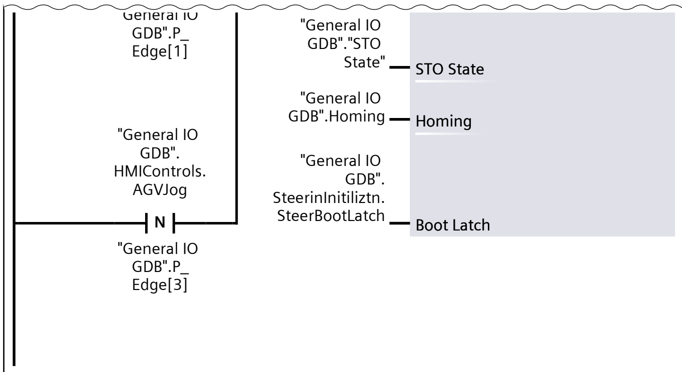
Network 5: Mode Selection (AGV in ON/OFF state?)

Network 5: Mode Selection (AGV in ON/OFF state?) (1.1 / 2.1)

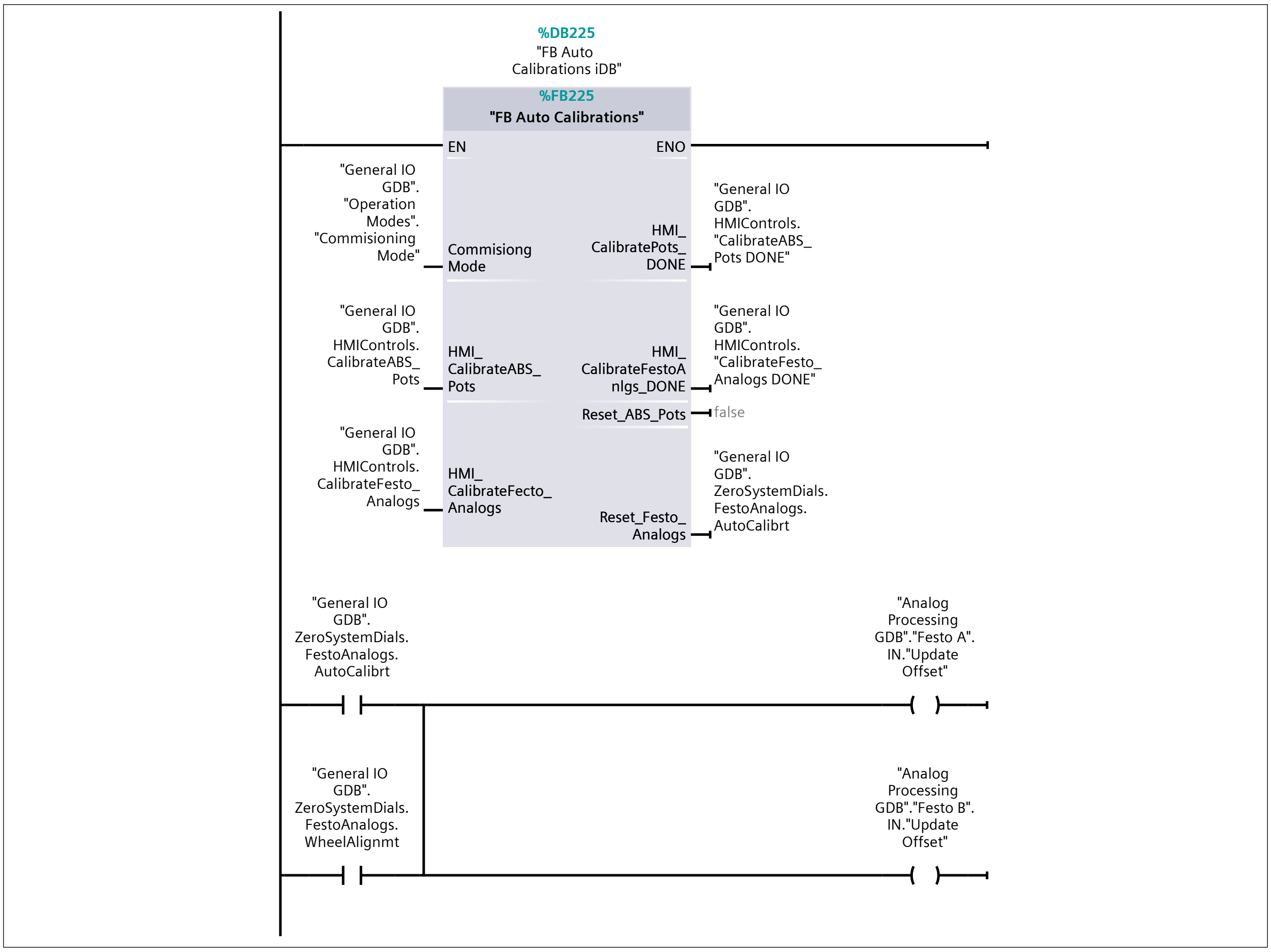

%DB6
"Power & Mode Control_iDB"
%FB6
"FB Power & Mode Control"
EN
ENO
"Inputs GDB". "AGV on"
SW ON/OFF
"Inputs GDB". ModeSelect
SW Mode
"General IO GDB". HMIControls. AGVOnOff
HMI ON/OFF
"General IO GDB".Testing. RunTestEnableAGV
HMI TestAGV ON
"General IO GDB".Testing. RunAbort
HMI TestAGV Abort
"General IO GDB". HMIControls. AutoMode
HMI AutoMode
"General IO GDB". HMIControls. ManMode
HMI ManualMode
"General IO GDB". HMIControls. CommisMode
HMI ComisngMode
"General IO GDB". HMIControls. UnitJog
P
"General IO GDB".P_ Edge[0]
HMI MoveUnits
"General IO GDB". HMIControls. AGVJog
P
"General IO GDB".P_ Edge[2]
HMI MoveAGV
"HMI OP GDB". "RunTest SW"
StartTestMode
"General IO GDB".Testing. TestCompleted
EndTestMode
"General IO GDB". HMIControls. UnitJog
N
ActivateDefault State
AGV PWR State
"General IO GDB"."AGV ON/ OFF State"
Automatic Mode
"General IO GDB". "Operation Modes". "Automatic Mode"
Manual Mode
"General IO GDB". "Operation Modes"."Manual Mode"
Commisioning Mode
"General IO GDB". "Operation Modes". "Commisioning Mode"
HMI Move Units
"General IO GDB". "Operation Modes"."HMI MoveUnits"
HMI Move AGV
"General IO GDB". "Operation Modes"."HMI MoveAGV"
Homing Mode
"General IO GDB". "Operation Modes".Homing
Testing Mode
"General IO GDB". "Operation Modes".Testing

Network 5: Mode Selection (AGV in ON/OFF state?) (2.1 / 2.1)

1.1 (Page2 - 7)

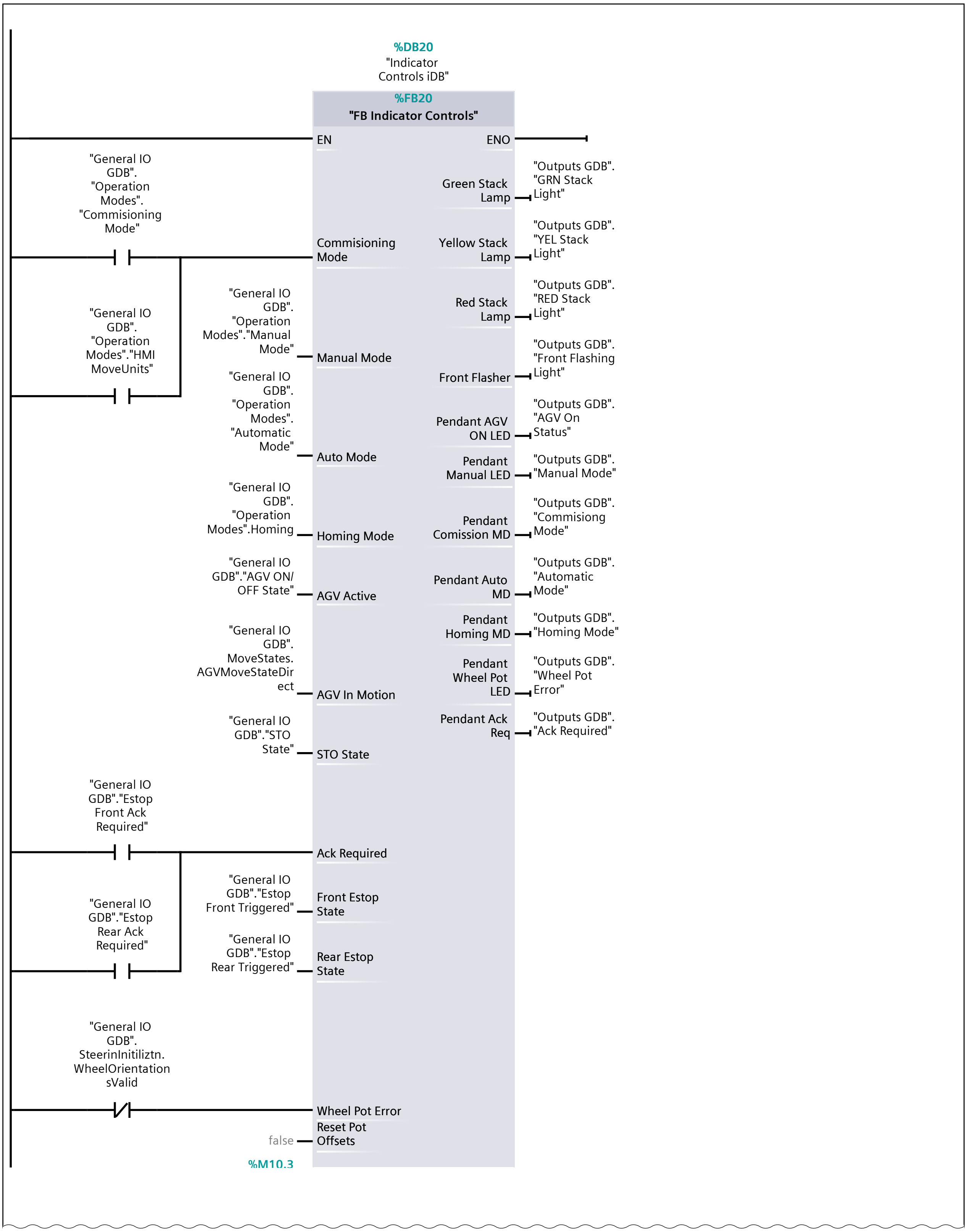


Network 6: Calibrate Absolute Pots OR Festo Analogs


%DB225
"FB Auto Calibrations iDB"
%FB225
"FB Auto Calibrations"
EN
ENO
"General IO GDB". "Operation Modes". "Commisioning Mode"
Commisiong Mode
HMI_ CalibratePots_ DONE
"General IO GDB". HMIControls. "CalibrateABS_ Pots DONE"
"General IO GDB". HMIControls. CalibrateABS_ Pots
HMI_ CalibrateABS_ Pots
HMI_ CalibrateFestoA nlgs_DONE
"General IO GDB". HMIControls. "CalibrateFesto_ Analogs DONE"
Reset_ABS_Pots
false
"General IO GDB". HMIControls. CalibrateFesto_ Analogs
HMI_ CalibrateFecto_ Analogs
Reset_Festo_ Analogs
"General IO GDB". ZeroSystemDials. FestoAnalogs. AutoCalibrt
"General IO GDB". ZeroSystemDials. FestoAnalogs. AutoCalibrt
"Analog Processing GDB"."Festo A". IN."Update Offset"
"General IO GDB". ZeroSystemDials. FestoAnalogs. WheelAlignmt
"Analog Processing GDB"."Festo B". IN."Update Offset"

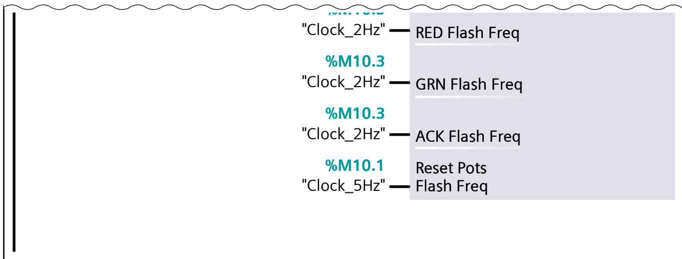
Network 7: Indicator, Stacklight and LED Controls

Network 7: Indicator, Stacklight and LED Controls (1.1 / 2.1)

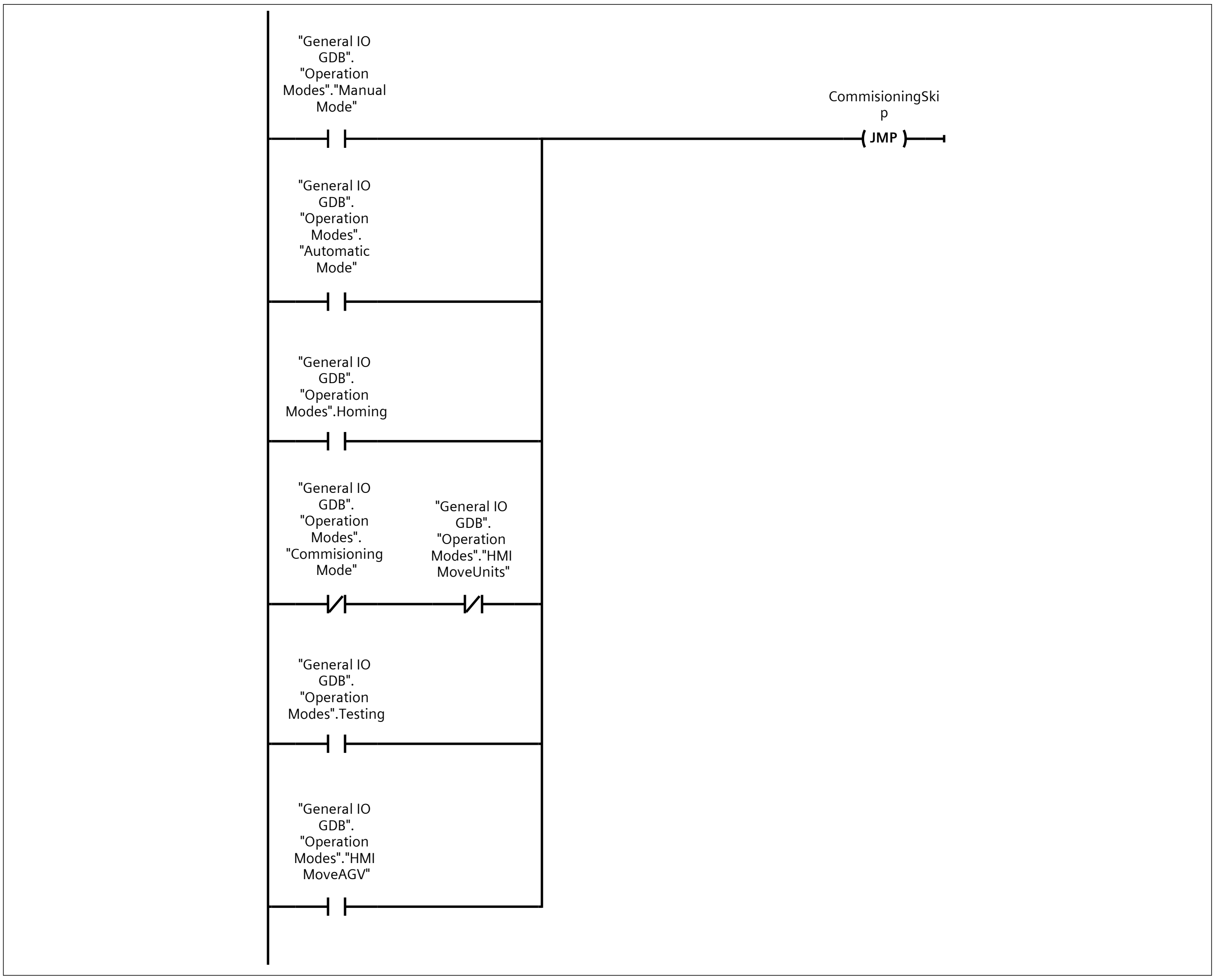


Network 7: Indicator, Stacklight and LED Controls (2.1 / 2.1)

1.1 (Page2 - 10)



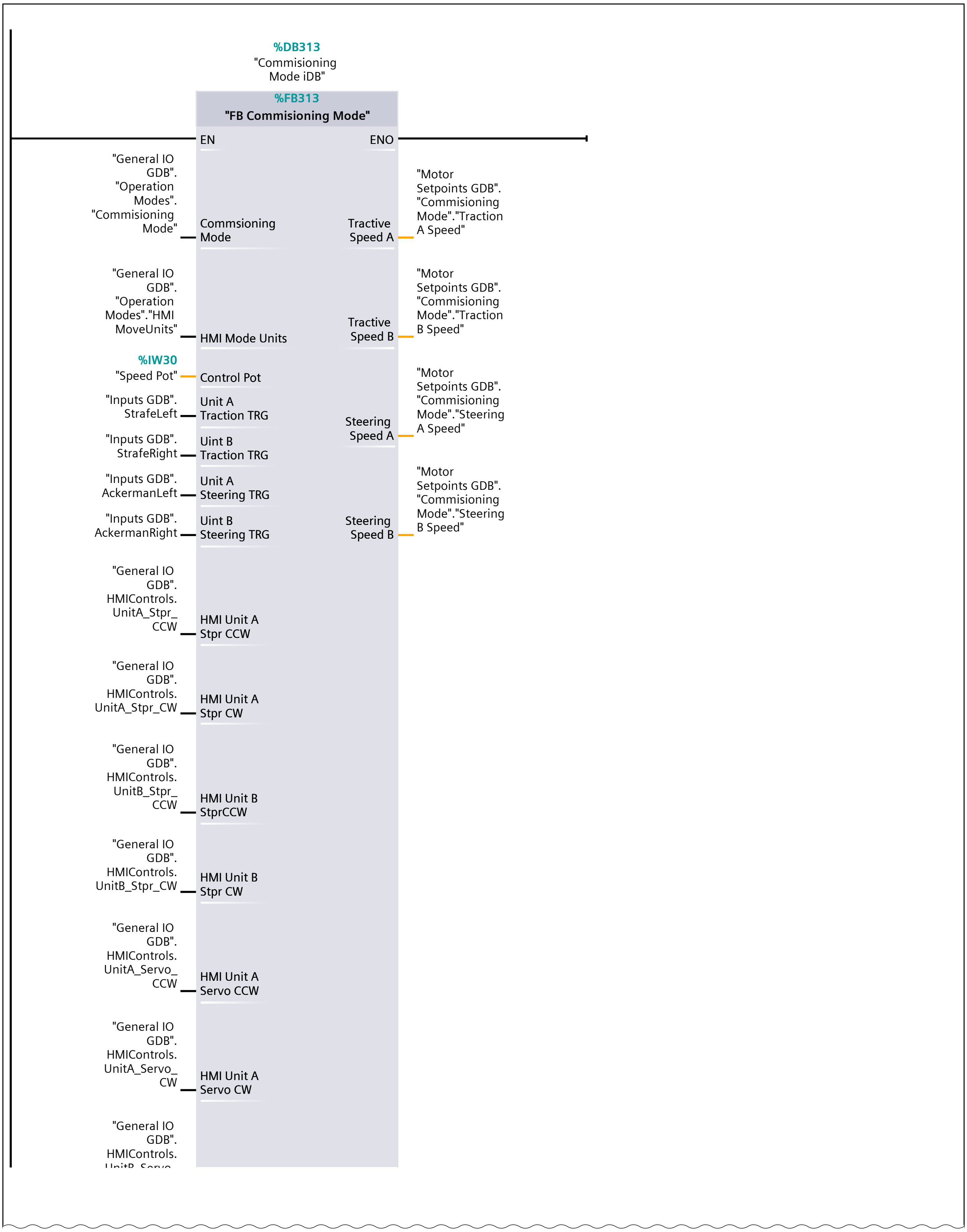
Network 8: ***JUMP*** over commisioning Mode


"General IO GDB". "Operation Modes"."Manual Mode"
CommisioningSkip
JMP
"General IO GDB". "Operation Modes". "Automatic Mode"
"General IO GDB". "Operation Modes".Homing
"General IO GDB". "Operation Modes". "Commisioning Mode"
"General IO GDB". "Operation Modes"."HMI MoveUnits"
"General IO GDB". "Operation Modes".Testing
"General IO GDB". "Operation Modes"."HMI MoveAGV"

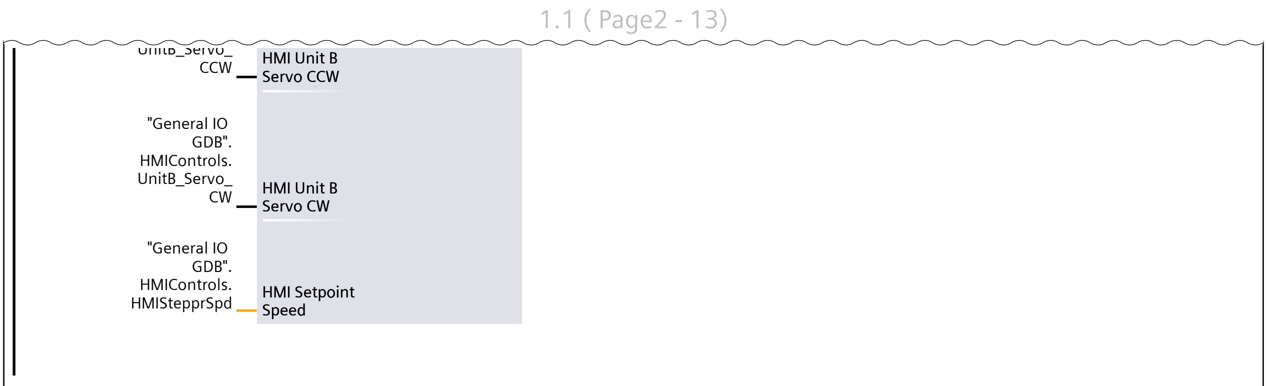
Network 9: Commisioning Mode

Bow Left = Unit A Traction
Bow Right = Unit B Traction
Turn Left = Unit A Steering
Turn Right = Unit B steering

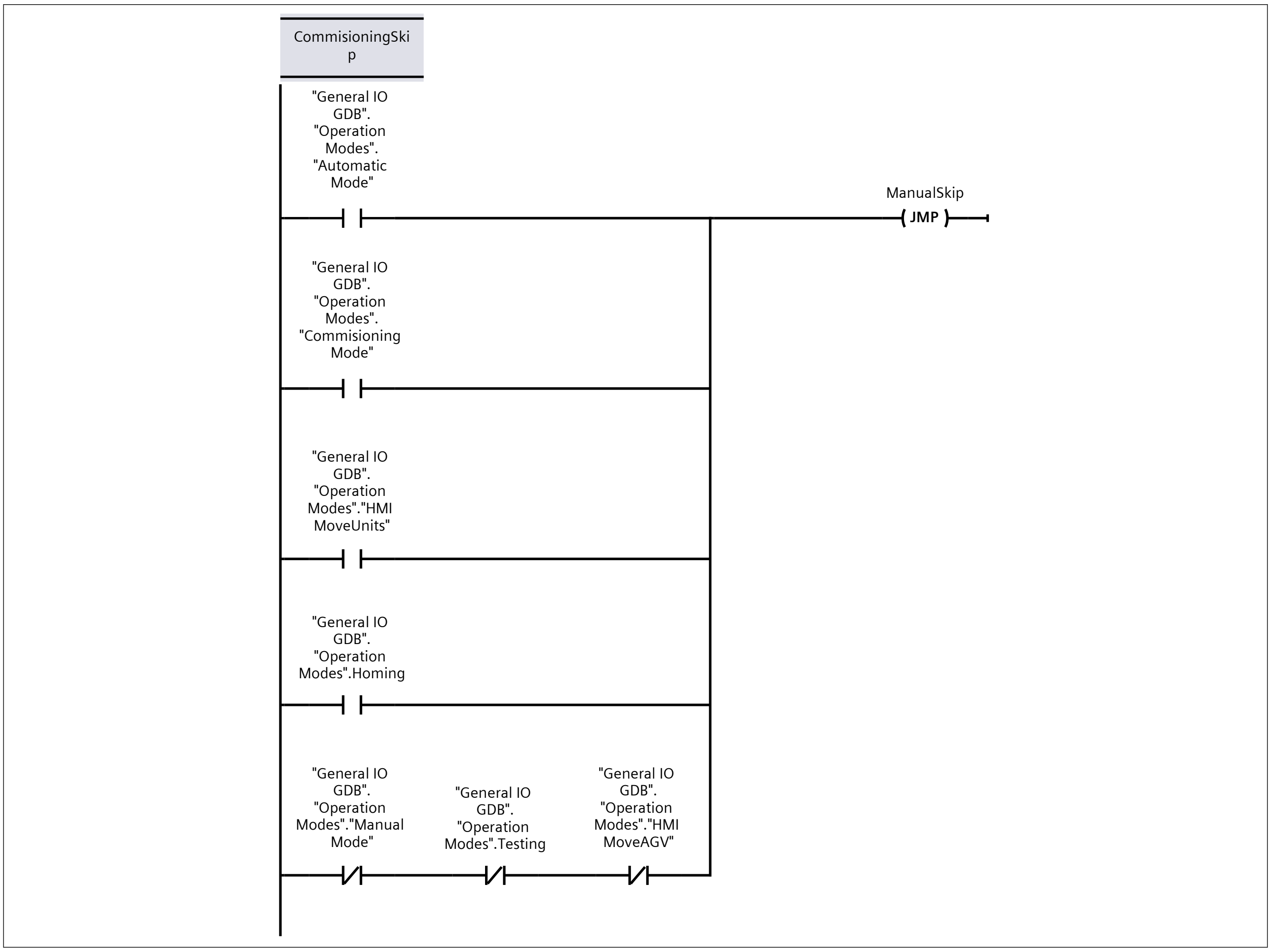
Network 9: Commisioning Mode (1.1 / 2.1)


%DB313
"Commisioning Mode iDB"
%FB313
"FB Commisioning Mode"
EN
ENO
"General IO GDB". "Operation Modes". "Commisioning Mode"
Commsioning Mode
Tractive Speed A
"Motor Setpoints GDB". "Commisioning Mode"."Traction A Speed"
"General IO GDB". "Operation Modes"."HMI MoveUnits"
HMI Mode Units
Tractive Speed B
"Motor Setpoints GDB". "Commisioning Mode"."Traction B Speed"
%IW30
"Speed Pot"
Control Pot
"Inputs GDB". StrafeLeft
Unit A Traction TRG
Steering Speed A
"Motor Setpoints GDB". "Commisioning Mode"."Steering A Speed"
"Inputs GDB". StrafeRight
Uint B Traction TRG
"Inputs GDB". AckermanLeft
Unit A Steering TRG
Steering Speed B
"Motor Setpoints GDB". "Commisioning Mode"."Steering B Speed"
"Inputs GDB". AckermanRight
Uint B Steering TRG
"General IO GDB". HMIControls. UnitA_Stpr_ CCW
HMI Unit A Stpr CCW
"General IO GDB". HMIControls. UnitA_Stpr_CW
HMI Unit A Stpr CW
"General IO GDB". HMIControls. UnitB_Stpr_ CCW
HMI Unit B StprCCW
"General IO GDB". HMIControls. UnitB_Stpr_CW
HMI Unit B Stpr CW
"General IO GDB". HMIControls. UnitA_Servo_ CCW
HMI Unit A Servo CCW
"General IO GDB". HMIControls. UnitA_Servo_ CW
HMI Unit A Servo CW
"General IO GDB". HMIControls. UnitB_Servo

Network 9: Commisioning Mode (2.1 / 2.1)



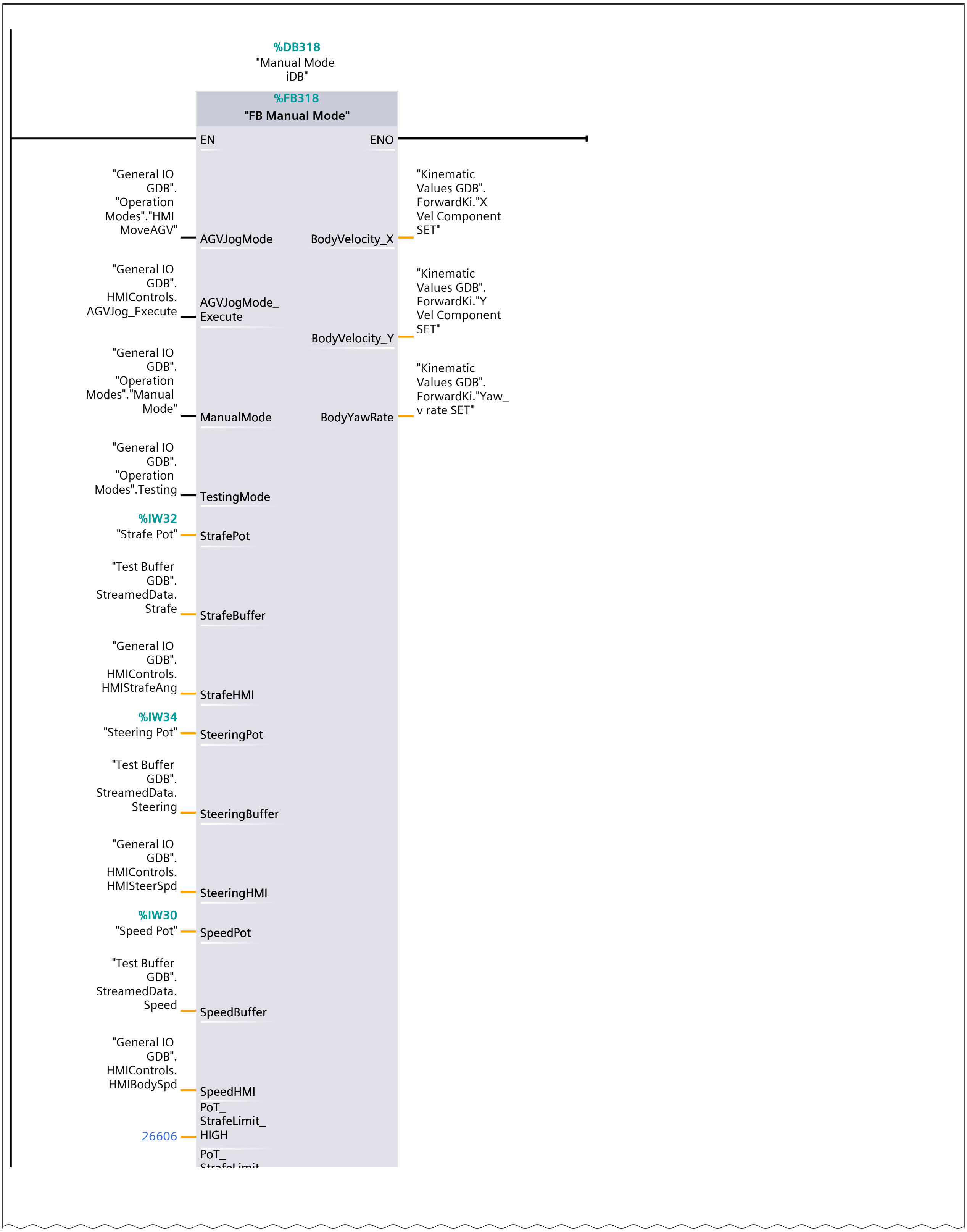
Network 10: ***JUMP*** over manual mode


CommisioningSkip
"General IO GDB"."Operation Modes"."Automatic Mode"
ManualSkip
JMP
"General IO GDB"."Operation Modes"."Commisioning Mode"
"General IO GDB"."Operation Modes"."HMI MoveUnits"
"General IO GDB"."Operation Modes".Homing
"General IO GDB"."Operation Modes"."Manual Mode"
"General IO GDB"."Operation Modes".Testing
"General IO GDB"."Operation Modes"."HMI MoveAGV"

Network 11: Manual Mode Data Generation

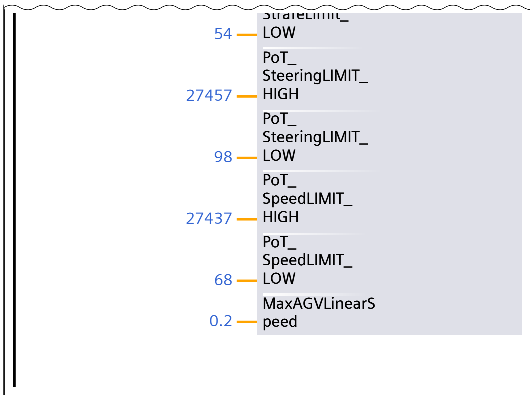
Note: Jump command skips this network if system is in auto mode (used to save cycle time)

Network 11: Manual Mode Data Generation (1.1 / 2.1)

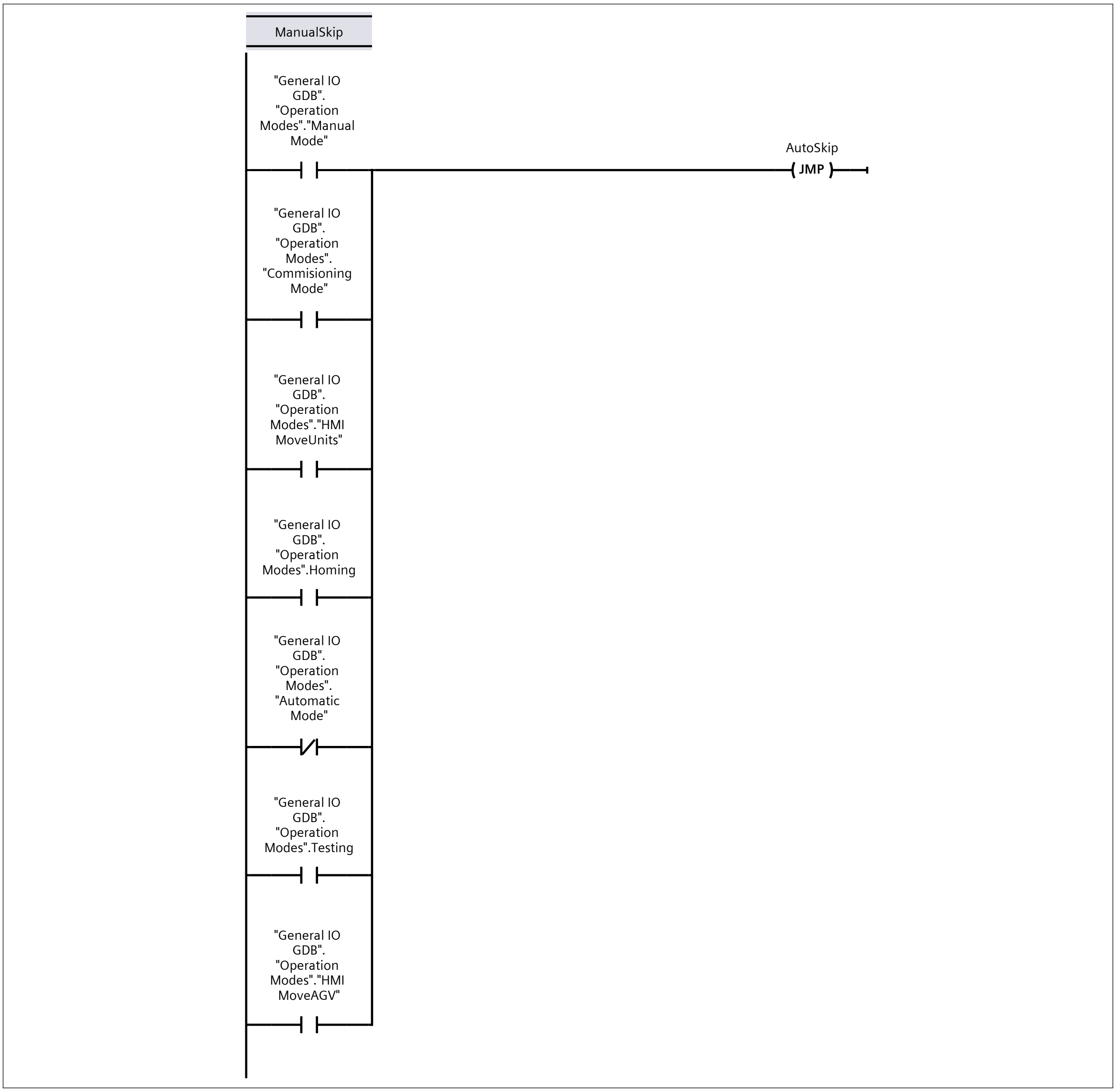


Network 11: Manual Mode Data Generation (2.1 / 2.1)

1.1 (Page2 - 16)

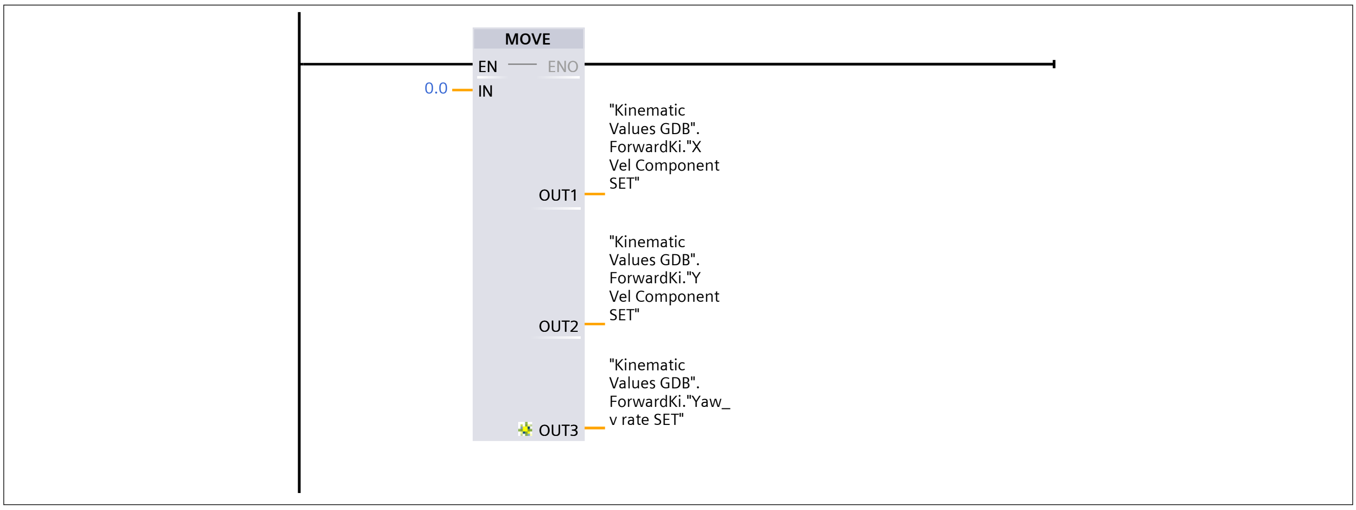


Network 12: ***JUMP*** over Automatic Mode

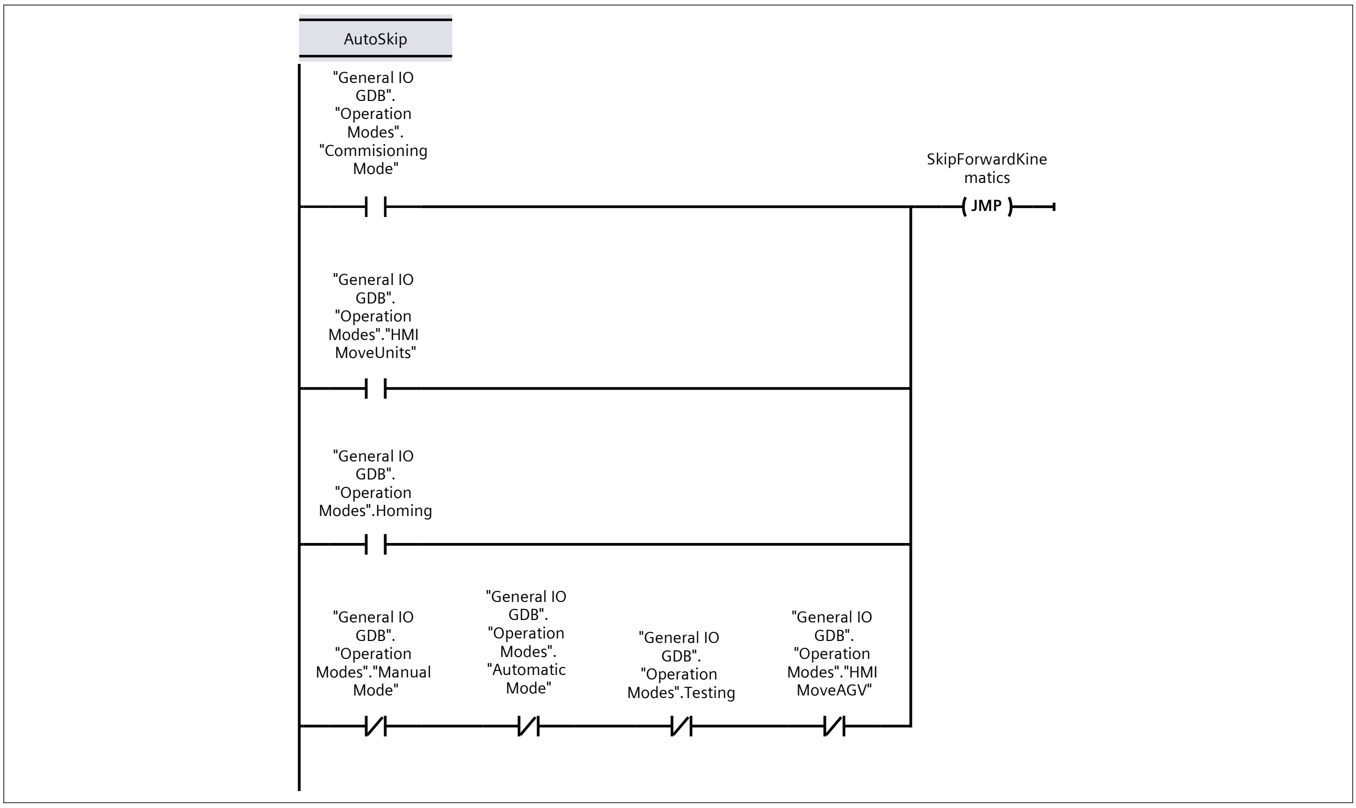

ManualSkip
"General IO GDB". "Operation Modes"."Manual Mode"
AutoSkip
JMP
"General IO GDB". "Operation Modes". "Commisioning Mode"
"General IO GDB". "Operation Modes"."HMI MoveUnits"
"General IO GDB". "Operation Modes".Homing
"General IO GDB". "Operation Modes". "Automatic Mode"
"General IO GDB". "Operation Modes".Testing
"General IO GDB". "Operation Modes"."HMI MoveAGV"

Network 13: Automatic Mode Data Generation

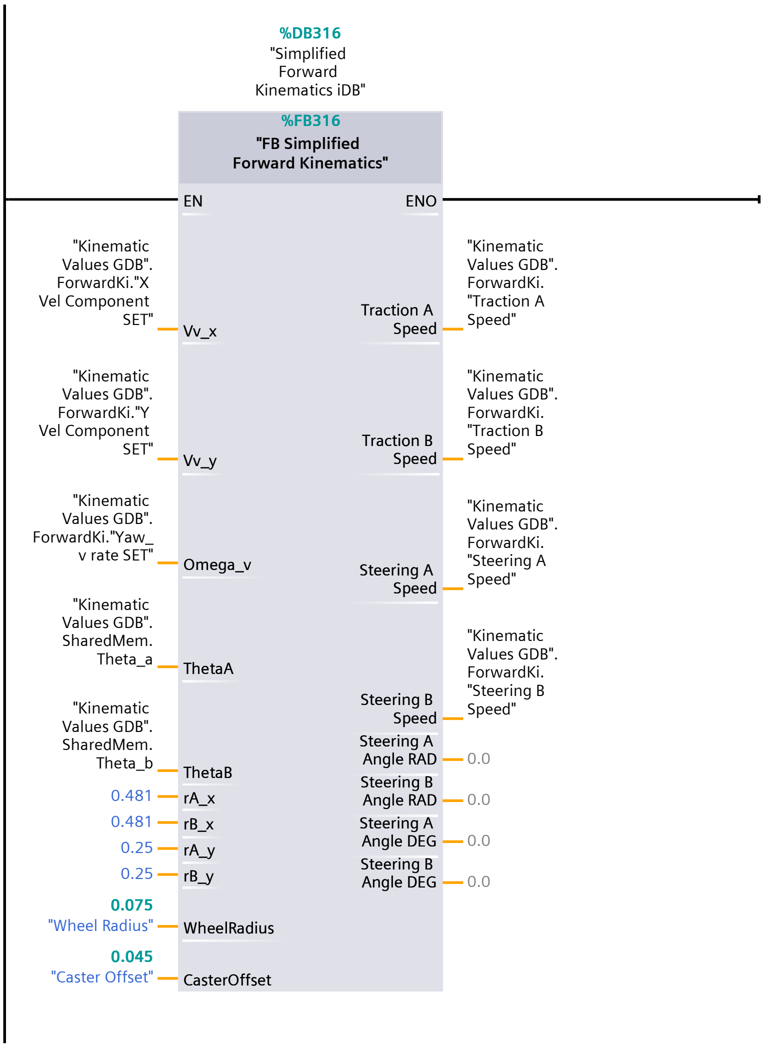
Note: Skips auto mode execution if in manual mode (save system cycle time)


MOVE
EN
ENO
0.0
IN
"Kinematic Values GDB". ForwardKi."X Vel Component SET"
OUT1
"Kinematic Values GDB". ForwardKi."Y Vel Component SET"
OUT2
"Kinematic Values GDB". ForwardKi."Yaw_ v rate SET"
OUT3

Network 14: ***JUMP*** over Forward Kinematics if in Commisioning Mode/Homing


AutoSkip
"General IO GDB". "Operation Modes". "Commisioning Mode"
SkipForwardKinematics
JMP
"General IO GDB". "Operation Modes"."HMI MoveUnits"
"General IO GDB". "Operation Modes".Homing
"General IO GDB". "Operation Modes"."Manual Mode"
"General IO GDB". "Operation Modes". "Automatic Mode"
"General IO GDB". "Operation Modes".Testing
"General IO GDB". "Operation Modes"."HMI MoveAGV"

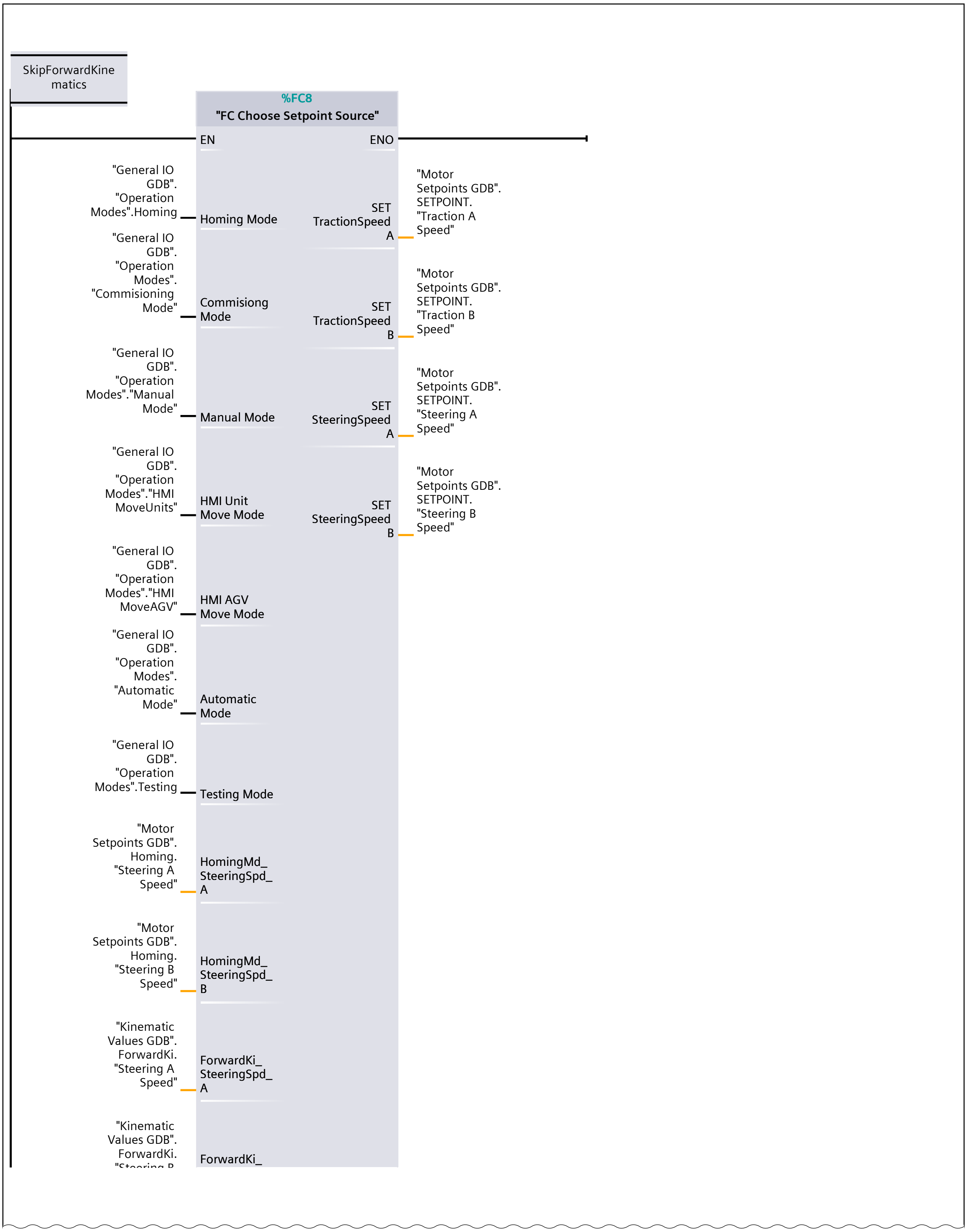
Network 15: Forward Kinematics


%DB316
"Simplified Forward Kinematics iDB"
%FB316
"FB Simplified Forward Kinematics"
EN
ENO
"Kinematic Values GDB". ForwardKi."X Vel Component SET"
Vv_x
Traction A Speed
"Kinematic Values GDB". ForwardKi. "Traction A Speed"
"Kinematic Values GDB". ForwardKi."Y Vel Component SET"
Vv_y
Traction B Speed
"Kinematic Values GDB". ForwardKi. "Traction B Speed"
"Kinematic Values GDB". ForwardKi."Yaw_v rate SET"
Omega_v
Steering A Speed
"Kinematic Values GDB". ForwardKi. "Steering A Speed"
"Kinematic Values GDB". SharedMem. Theta_a
ThetaA
Steering B Speed
"Kinematic Values GDB". ForwardKi. "Steering B Speed"
"Kinematic Values GDB". SharedMem. Theta_b
ThetaB
Steering A Angle RAD
0.0
Steering B Angle RAD
0.0
0.481
rA_x
0.481
rB_x
Steering A Angle DEG
0.0
0.25
rA_y
0.25
rB_y
Steering B Angle DEG
0.0
0.075
"Wheel Radius"
WheelRadius
0.045
"Caster Offset"
CasterOffset

Network 16: Choose Setpoint Source

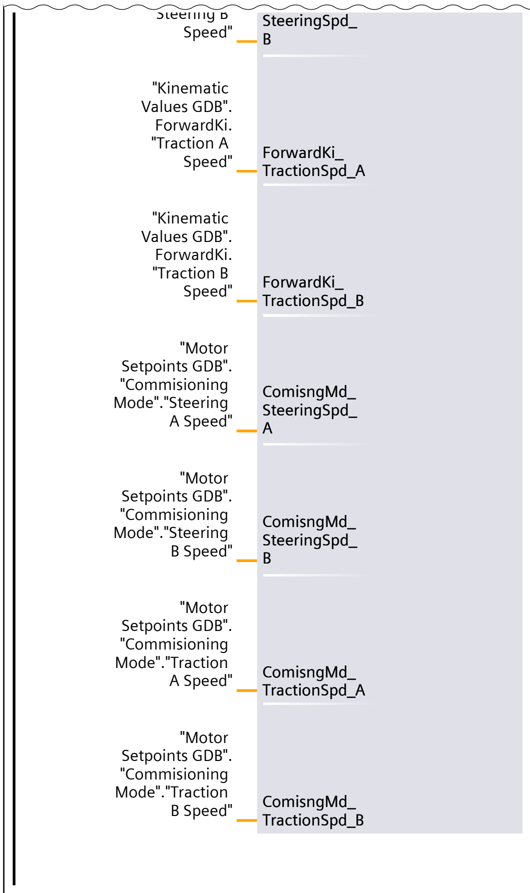
(Auto || Manual) && !Commisioning => Drives get forward kinematic rad/s values
(!Auto && !Manual) && Commisioning => Drives get rad/s values direct from commisioning block
!Auto && !Manual && !Commisioning => All drives get 0 rad/s

Network 16: Choose Setpoint Source (1.1 / 2.1)

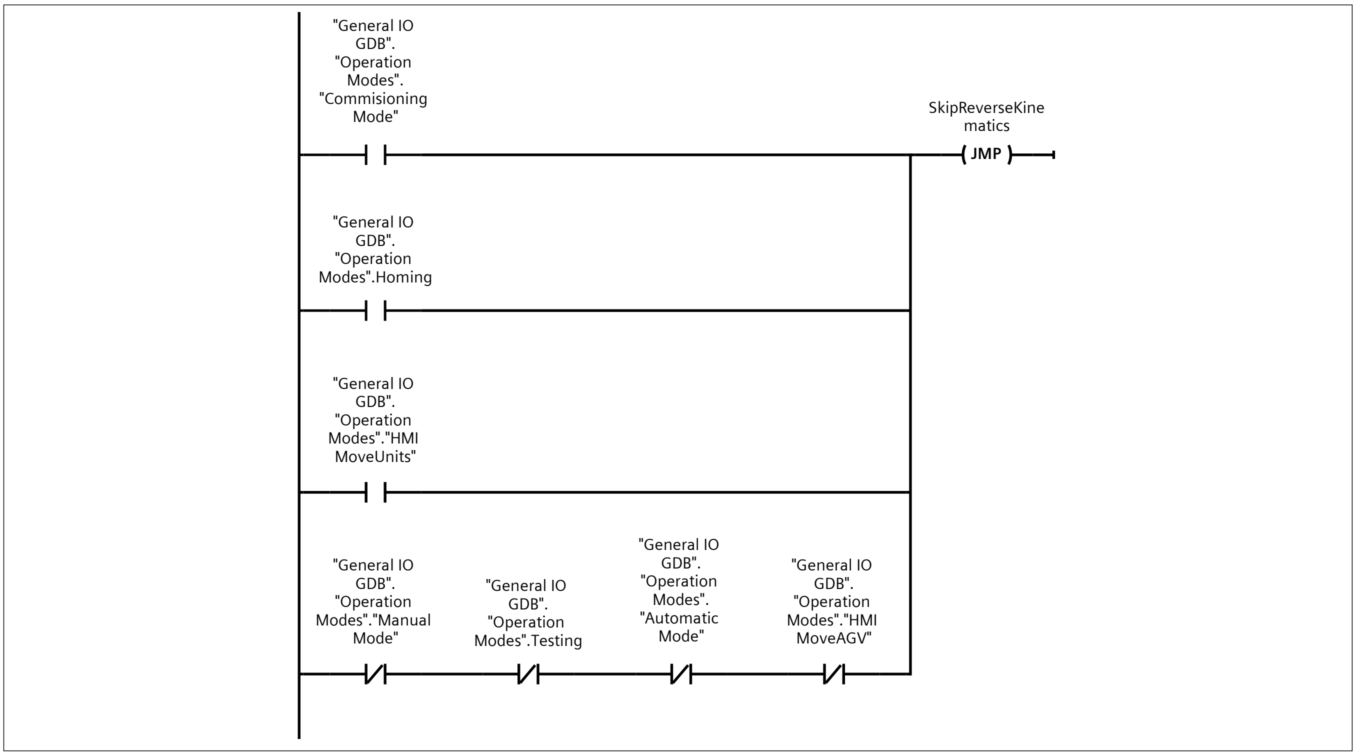

SkipForwardKinematics
%FC8
"FC Choose Setpoint Source"
EN
ENO
"General IO GDB". "Operation Modes".Homing
Homing Mode
"General IO GDB". "Operation Modes". "Commisioning Mode"
Commisiong Mode
"General IO GDB". "Operation Modes"."Manual Mode"
Manual Mode
"General IO GDB". "Operation Modes"."HMI MoveUnits"
HMI Unit Move Mode
"General IO GDB". "Operation Modes"."HMI MoveAGV"
HMI AGV Move Mode
"General IO GDB". "Operation Modes". "Automatic Mode"
Automatic Mode
"General IO GDB". "Operation Modes".Testing
Testing Mode
"Motor Setpoints GDB". Homing. "Steering A Speed"
HomingMd_SteeringSpd_A
"Motor Setpoints GDB". Homing. "Steering B Speed"
HomingMd_SteeringSpd_B
"Kinematic Values GDB". ForwardKi. "Steering A Speed"
ForwardKi_SteeringSpd_A
"Kinematic Values GDB". ForwardKi. "Steering B
ForwardKi_
SET TractionSpeed A
"Motor Setpoints GDB". SETPOINT. "Traction A Speed"
SET TractionSpeed B
"Motor Setpoints GDB". SETPOINT. "Traction B Speed"
SET SteeringSpeed A
"Motor Setpoints GDB". SETPOINT. "Steering A Speed"
SET SteeringSpeed B
"Motor Setpoints GDB". SETPOINT. "Steering B Speed"

Network 16: Choose Setpoint Source (2.1 / 2.1)

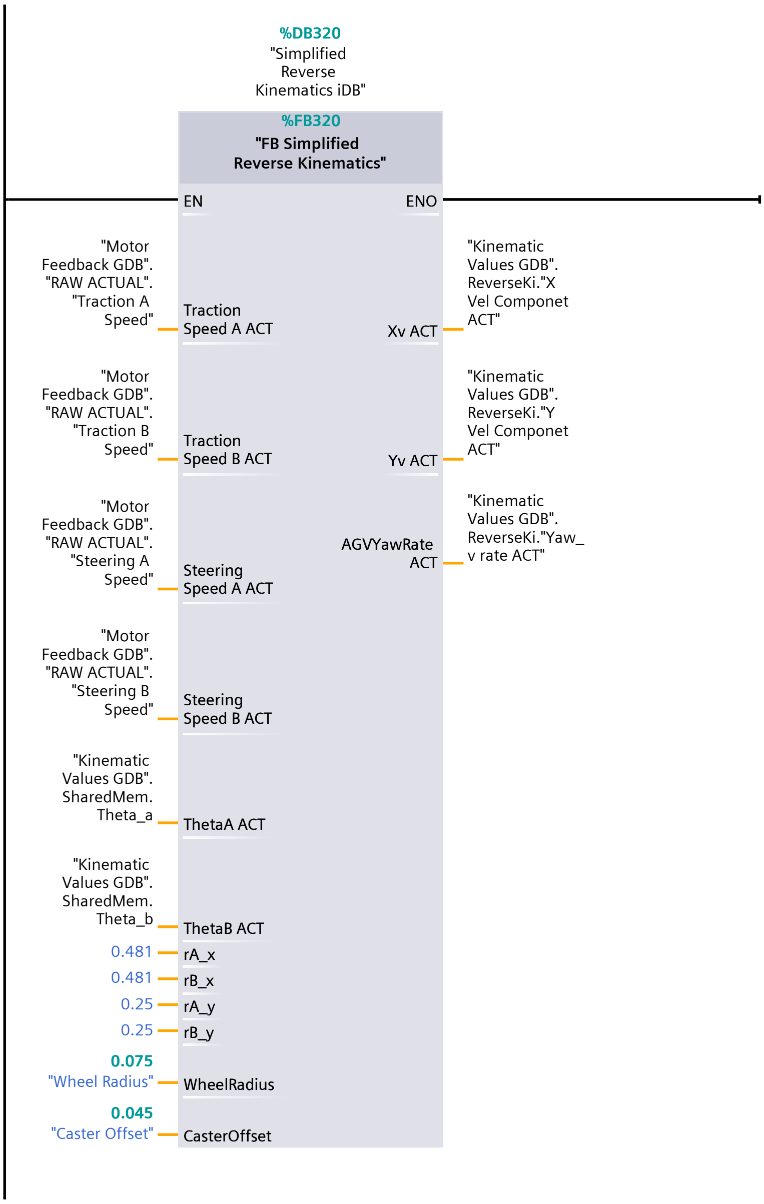
1.1 (Page2 - 21)


Steering B Speed"
SteeringSpd_B
"Kinematic Values GDB". ForwardKi. "Traction A Speed"
ForwardKi_TractionSpd_A
"Kinematic Values GDB". ForwardKi. "Traction B Speed"
ForwardKi_TractionSpd_B
"Motor Setpoints GDB". "Commisioning Mode"."Steering A Speed"
ComisngMd_SteeringSpd_A
"Motor Setpoints GDB". "Commisioning Mode"."Steering B Speed"
ComisngMd_SteeringSpd_B
"Motor Setpoints GDB". "Commisioning Mode"."Traction A Speed"
ComisngMd_TractionSpd_A
"Motor Setpoints GDB". "Commisioning Mode"."Traction B Speed"
ComisngMd_TractionSpd_B

Network 17: ***JUMP*** over Reverse Kinematics if in Commisioning Mode


"General IO GDB". "Operation Modes". "Commisioning Mode"
SkipReverseKinematics
JMP
"General IO GDB". "Operation Modes".Homing
"General IO GDB". "Operation Modes"."HMI MoveUnits"
"General IO GDB". "Operation Modes"."Manual Mode"
"General IO GDB". "Operation Modes".Testing
"General IO GDB". "Operation Modes". "Automatic Mode"
"General IO GDB". "Operation Modes"."HMI MoveAGV"

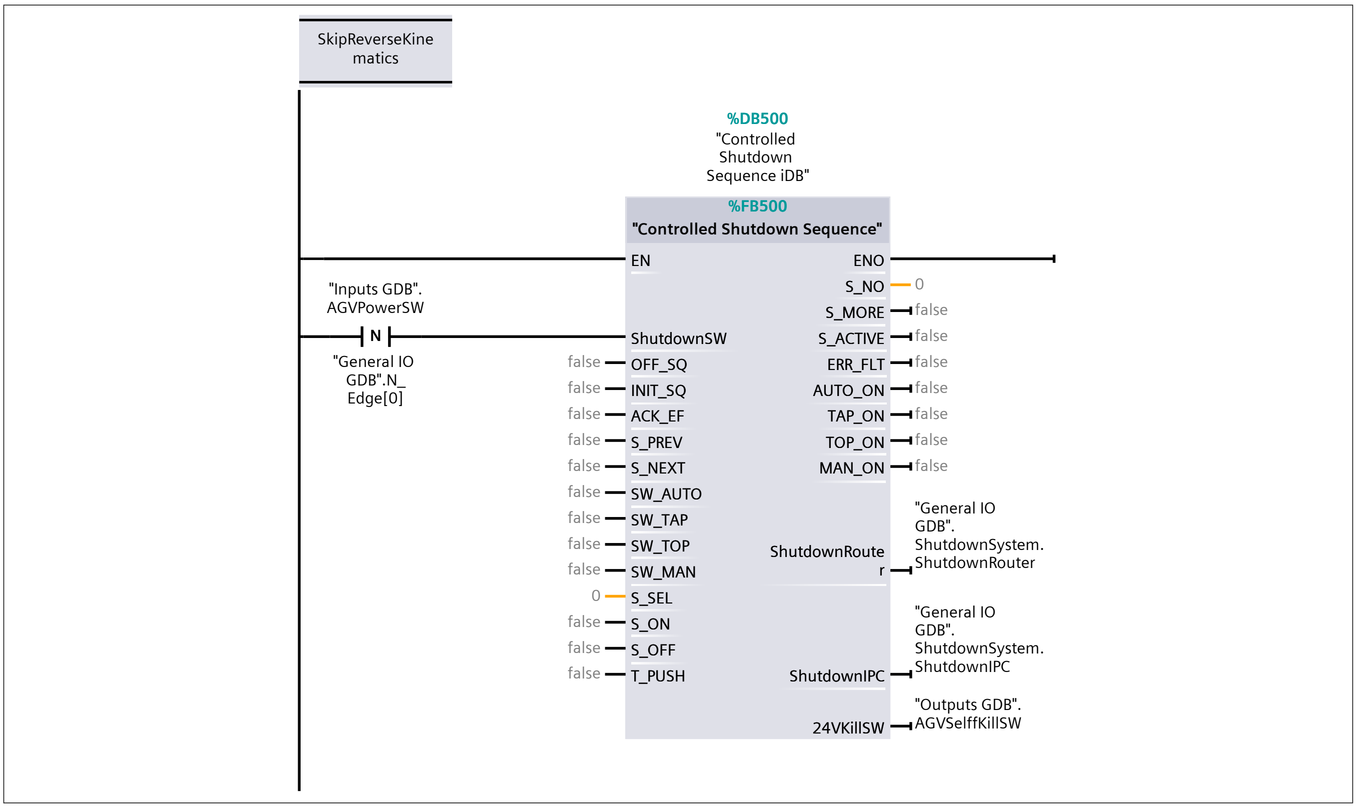
Network 18: Reverse Kinematics


%DB320
"Simplified Reverse Kinematics iDB"
%FB320
"FB Simplified Reverse Kinematics"
EN
ENO
"Motor Feedback GDB". "RAW ACTUAL". "Traction A Speed"
Traction Speed A ACT
Xv ACT
"Kinematic Values GDB". ReverseKi."X Vel Componet ACT"
"Motor Feedback GDB". "RAW ACTUAL". "Traction B Speed"
Traction Speed B ACT
Yv ACT
"Kinematic Values GDB". ReverseKi."Y Vel Componet ACT"
"Motor Feedback GDB". "RAW ACTUAL". "Steering A Speed"
Steering Speed A ACT
AGVYawRate ACT
"Kinematic Values GDB". ReverseKi."Yaw_ v rate ACT"
"Motor Feedback GDB". "RAW ACTUAL". "Steering B Speed"
Steering Speed B ACT
"Kinematic Values GDB". SharedMem. Theta_a
ThetaA ACT
"Kinematic Values GDB". SharedMem. Theta_b
ThetaB ACT
0.481
rA_x
0.481
rB_x
0.25
rA_y
0.25
rB_y
0.075
"Wheel Radius"
WheelRadius
0.045
"Caster Offset"
CasterOffset

Network 19: Controlled Shutdown of System

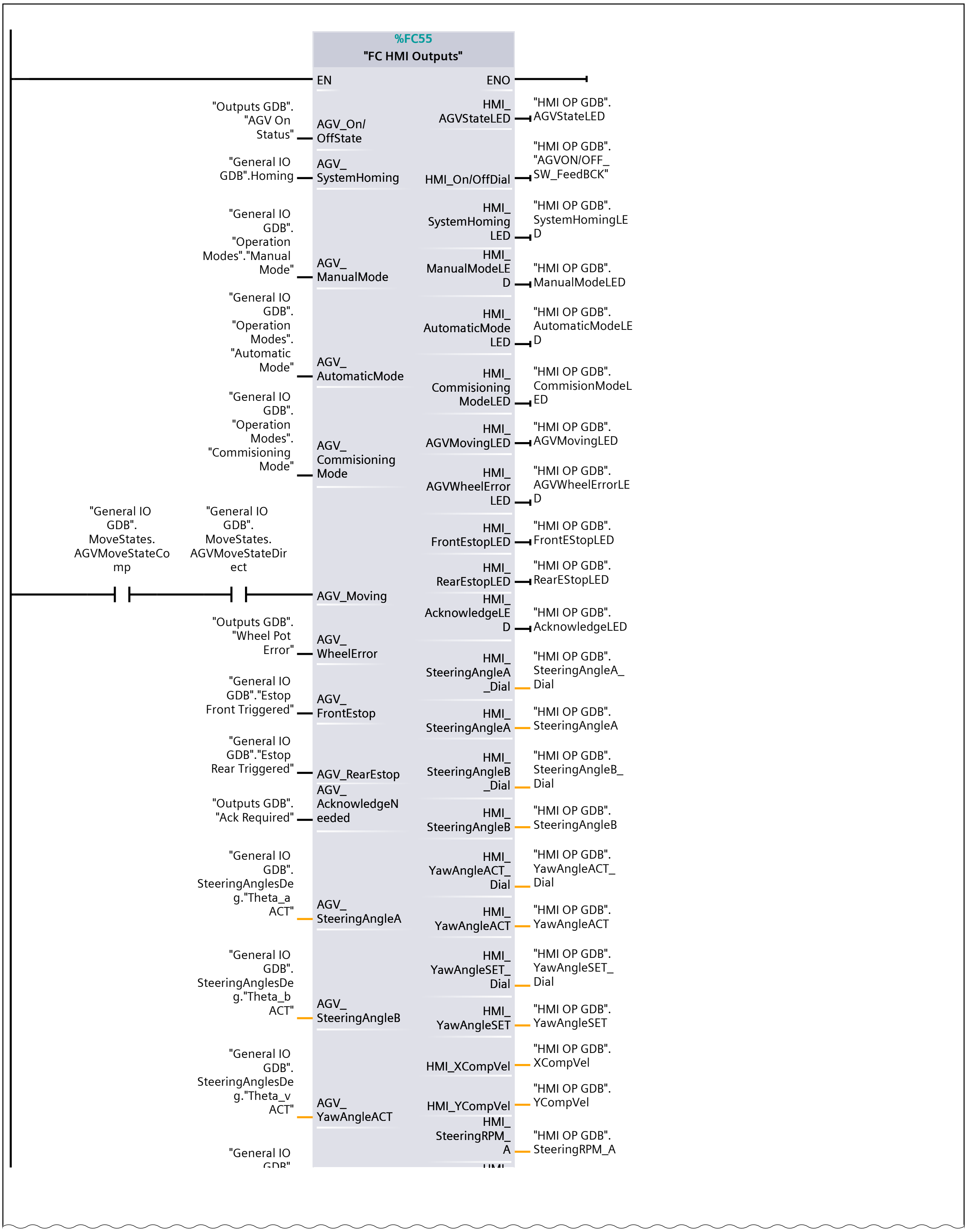
The devices on the system need to be shutdown in a cotrolled fashion, simply cutting the power is not acceptable. The two items that need a controlled shutdown are:
1) the IPC
2) the router

Especially the router, if power is simply cut a corrupted ARP tabe can result, this requires the operator to SSH into the router and manually delete the corrupted ARP table: https://dannyda.com/2020/12/20/how-to-fix-pfsense-boot-loop-issue/



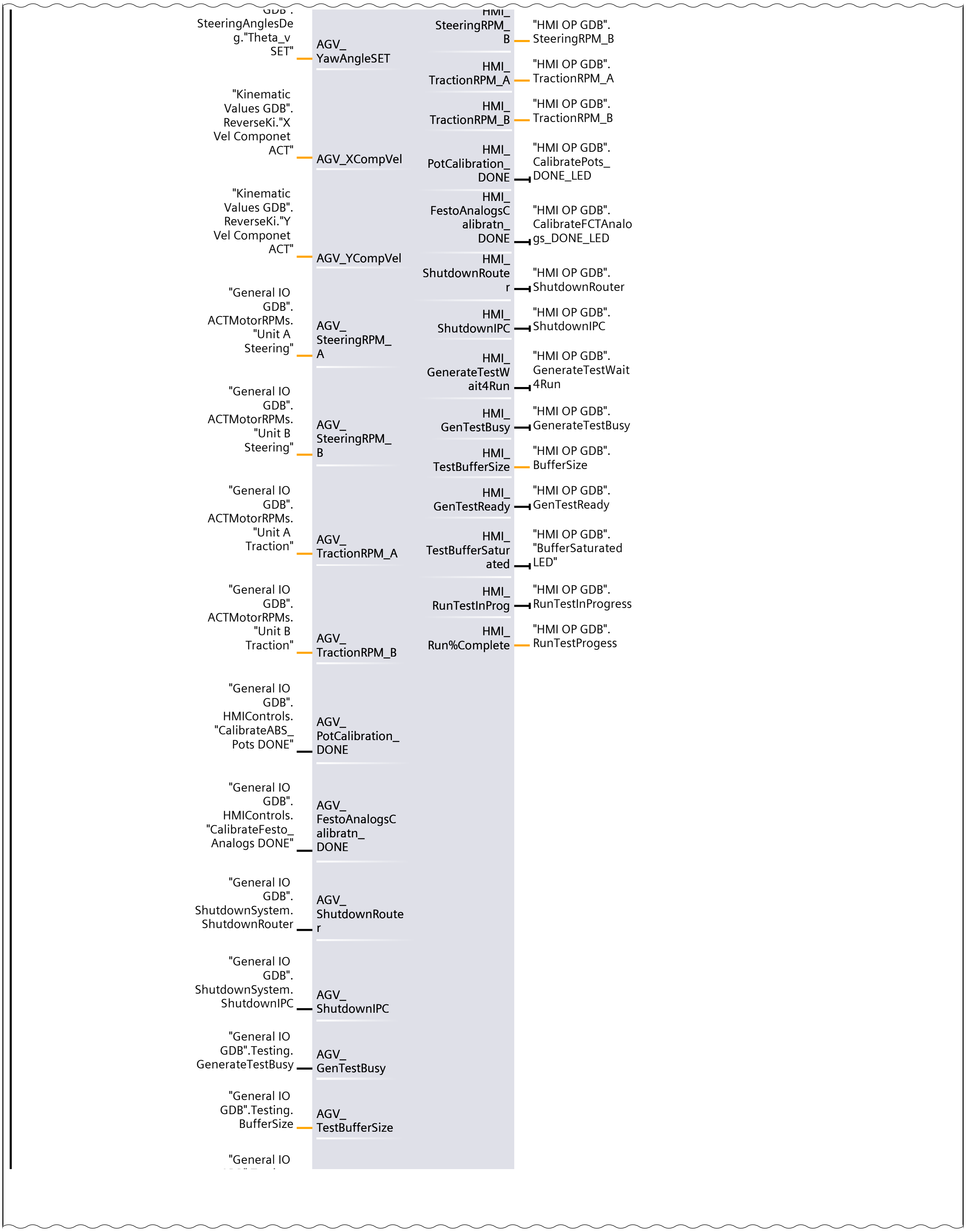
Network 20: HMI Outputs AGV ---> HMI

Network 20: HMI Outputs AGV ---> HMI (1.1 / 3.1)



Network 20: HMI Outputs AGV ---> HMI (2.1 / 3.1)

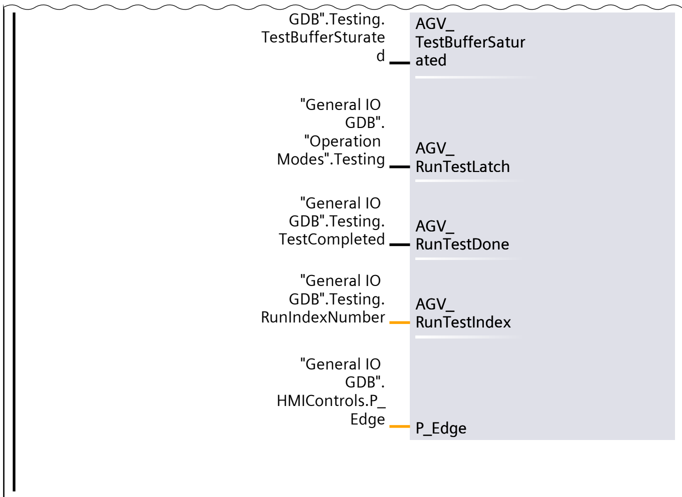
1.1 (Page2 - 26)


GDB".
SteeringAnglesDeg."Theta_v SET"
AGV_YawAngleSET
"Kinematic Values GDB".ReverseKi."X Vel Componet ACT"
AGV_XCompVel
"Kinematic Values GDB".ReverseKi."Y Vel Componet ACT"
AGV_YCompVel
"General IO GDB".ACTMotorRPMs."Unit A Steering"
AGV_SteeringRPM_A
"General IO GDB".ACTMotorRPMs."Unit B Steering"
AGV_SteeringRPM_B
"General IO GDB".ACTMotorRPMs."Unit A Traction"
AGV_TractionRPM_A
"General IO GDB".ACTMotorRPMs."Unit B Traction"
AGV_TractionRPM_B
"General IO GDB".HMIControls."CalibrateABS_Pots DONE"
AGV_PotCalibration_DONE
"General IO GDB".HMIControls."CalibrateFesto_Analogs DONE"
AGV_FestoAnalogsCalibratn_DONE
"General IO GDB".ShutdownSystem.ShutdownRouter
AGV_ShutdownRouter
"General IO GDB".ShutdownSystem.ShutdownIPC
AGV_ShutdownIPC
"General IO GDB".Testing.GenerateTestBusy
AGV_GenTestBusy
"General IO GDB".Testing.BufferSize
AGV_TestBufferSize
"General IO
HMI_SteeringRPM_B
"HMI OP GDB".SteeringRPM_B
HMI_TractionRPM_A
"HMI OP GDB".TractionRPM_A
HMI_TractionRPM_B
"HMI OP GDB".TractionRPM_B
HMI_PotCalibration_DONE
"HMI OP GDB".CalibratePots_DONE_LED
HMI_FestoAnalogsCalibratn_DONE
"HMI OP GDB".CalibrateFCTAnalogs_DONE_LED
HMI_ShutdownRouter
"HMI OP GDB".ShutdownRouter
HMI_ShutdownIPC
"HMI OP GDB".ShutdownIPC
HMI_GenerateTestWait4Run
"HMI OP GDB".GenerateTestWait4Run
HMI_GenTestBusy
"HMI OP GDB".GenerateTestBusy
HMI_TestBufferSize
"HMI OP GDB".BufferSize
HMI_GenTestReady
"HMI OP GDB".GenTestReady
HMI_TestBufferSaturated
"HMI OP GDB"."BufferSaturated LED"
HMI_RunTestInProg
"HMI OP GDB".RunTestInProgress
HMI_Run%Complete
"HMI OP GDB".RunTestProgess

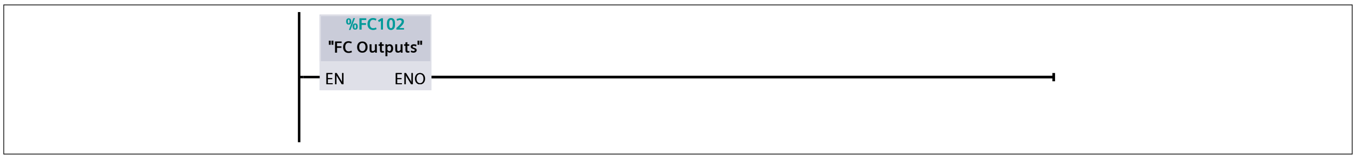
3.1 (Page2 - 28)

Network 20: HMI Outputs AGV ---> HMI (3.1 / 3.1)

2.1 (Page2 - 27)



Network 21: Outputs Control (Last Network)



Motor Update [OB31]

Motor Update Properties					
General					
Name	Motor Update	**Number**	31	**Type**	OB
Language	LAD	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value
▼ Input		
Initial_Call	Bool	
Event_Count	Int	
Temp		
Constant		

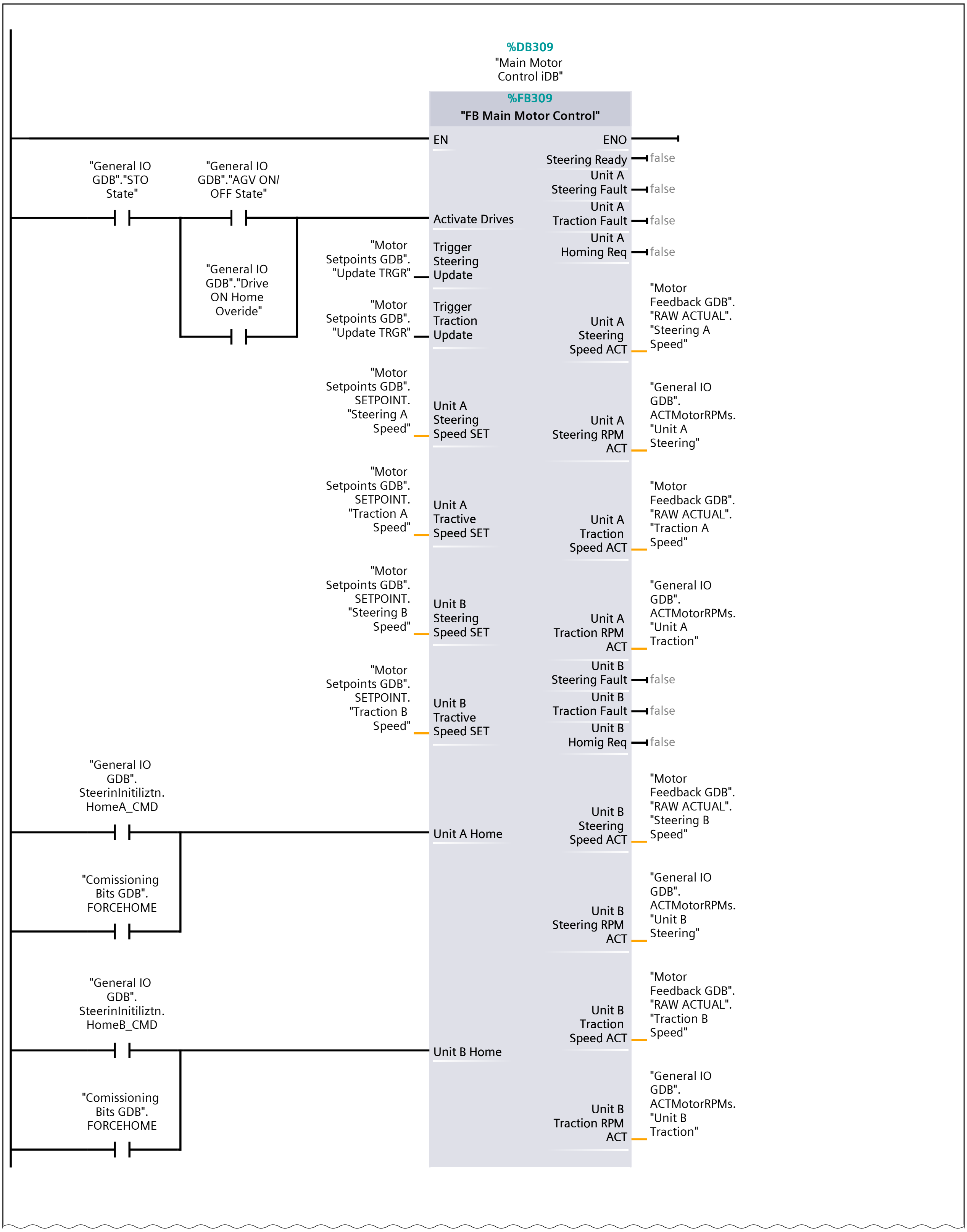
Network 1: Toggle Update Trigger

```
IF "Motor Setpoints GDB"."Update TRGR" = TRUE THEN
    // If bit true => flip bit to false
    "Motor Setpoints GDB"."Update TRGR" := FALSE;
ELSIF "Motor Setpoints GDB"."Update TRGR" = FALSE THEN
    // If bit true => flip bit to true
    "Motor Setpoints GDB"."Update TRGR" := TRUE;
ELSE
    // Should never have this state => do nothing
    ;
END_IF;


```

Network 2: Motors (Traction & Steering)

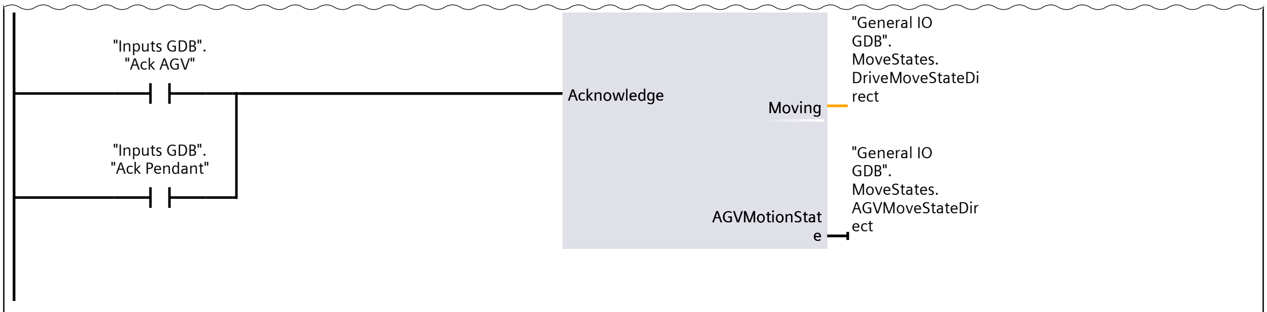
Control motors cyclically

Network 2: Motors (Traction & Steering) (1.1 / 2.1)

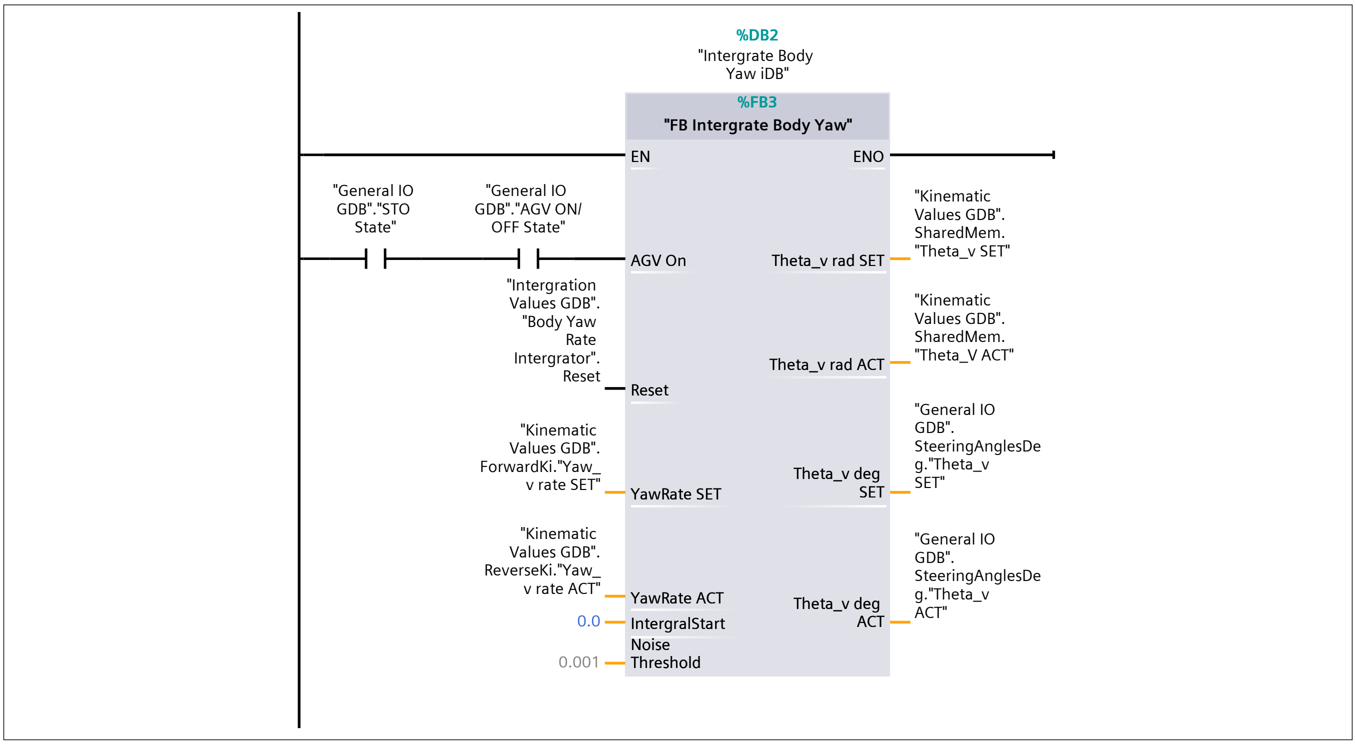


Network 2: Motors (Traction & Steering) (2.1 / 2.1)

1.1 (Page3 - 2)

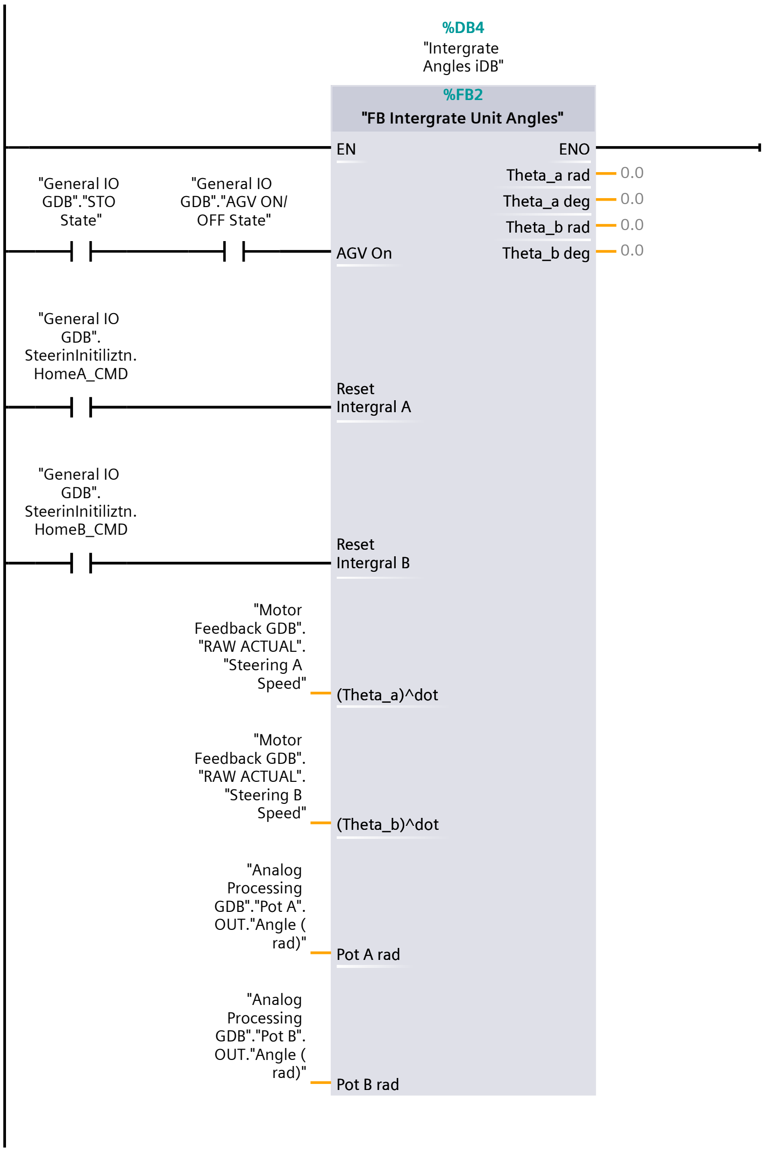


Network 3: Intergrate the Body Yaw Rate to get the body Angle


%DB2
"Intergrate Body Yaw iDB"
%FB3
"FB Intergrate Body Yaw"
EN
ENO
"General IO GDB"."STO State"
"General IO GDB"."AGV ON/OFF State"
AGV On
Theta_v rad SET
"Kinematic Values GDB".SharedMem."Theta_v SET"
"Intergration Values GDB"."Body Yaw Rate Intergrator".Reset
Reset
Theta_v rad ACT
"Kinematic Values GDB".SharedMem."Theta_V ACT"
"Kinematic Values GDB".ForwardKi."Yaw_v rate SET"
YawRate SET
Theta_v deg SET
"General IO GDB".SteeringAnglesDeg."Theta_v SET"
"Kinematic Values GDB".ReverseKi."Yaw_v rate ACT"
YawRate ACT
Theta_v deg ACT
"General IO GDB".SteeringAnglesDeg."Theta_v ACT"
0.0
IntergralStart
0.001
Noise Threshold

Network 4: Use Intergration to Calculate Steering Angles

Calculate Theta_a and Theta_b


%DB4
"Intergrate
Angles iDB"
%FB2
"FB Intergrate Unit Angles"
EN
ENO
Theta_a rad
0.0
Theta_a deg
0.0
Theta_b rad
0.0
Theta_b deg
0.0
"General IO
GDB"."STO
State"
"General IO
GDB"."AGV ON/
OFF State"
AGV On
"General IO
GDB".
SteerinInitiliztn.
HomeA_CMD
Reset
Intergral A
"General IO
GDB".
SteerinInitiliztn.
HomeB_CMD
Reset
Intergral B
"Motor
Feedback GDB".
"RAW ACTUAL".
"Steering A
Speed"
(Theta_a)^dot
"Motor
Feedback GDB".
"RAW ACTUAL".
"Steering B
Speed"
(Theta_b)^dot
"Analog
Processing
GDB"."Pot A".
OUT."Angle (
rad)"
Pot A rad
"Analog
Processing
GDB"."Pot B".
OUT."Angle (
rad)"
Pot B rad

Pot Processing [OB30]

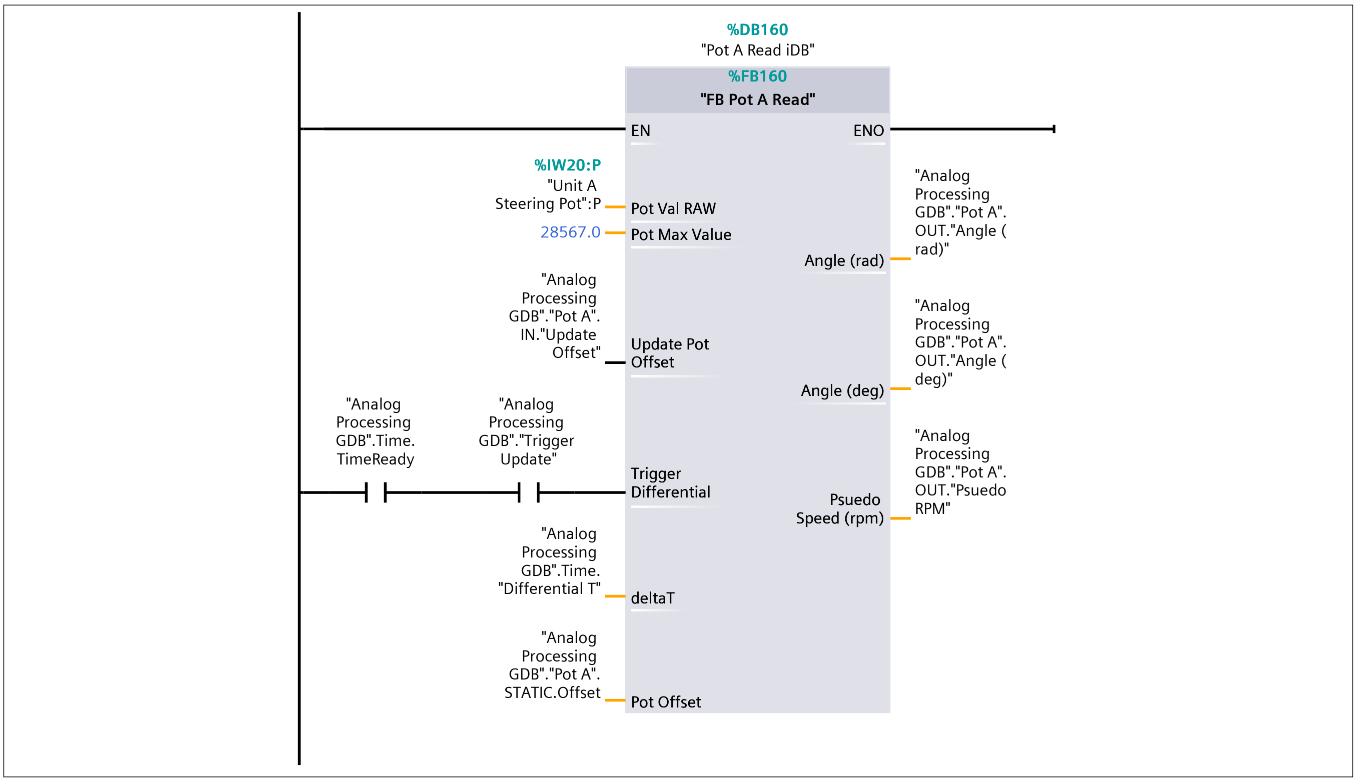
Pot Processing Properties					
General					
Name	Pot Processing	**Number**	30	**Type**	OB
Language	LAD	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value
▼ Input		
Initial_Call	Bool	
Event_Count	Int	
Temp		
Constant		

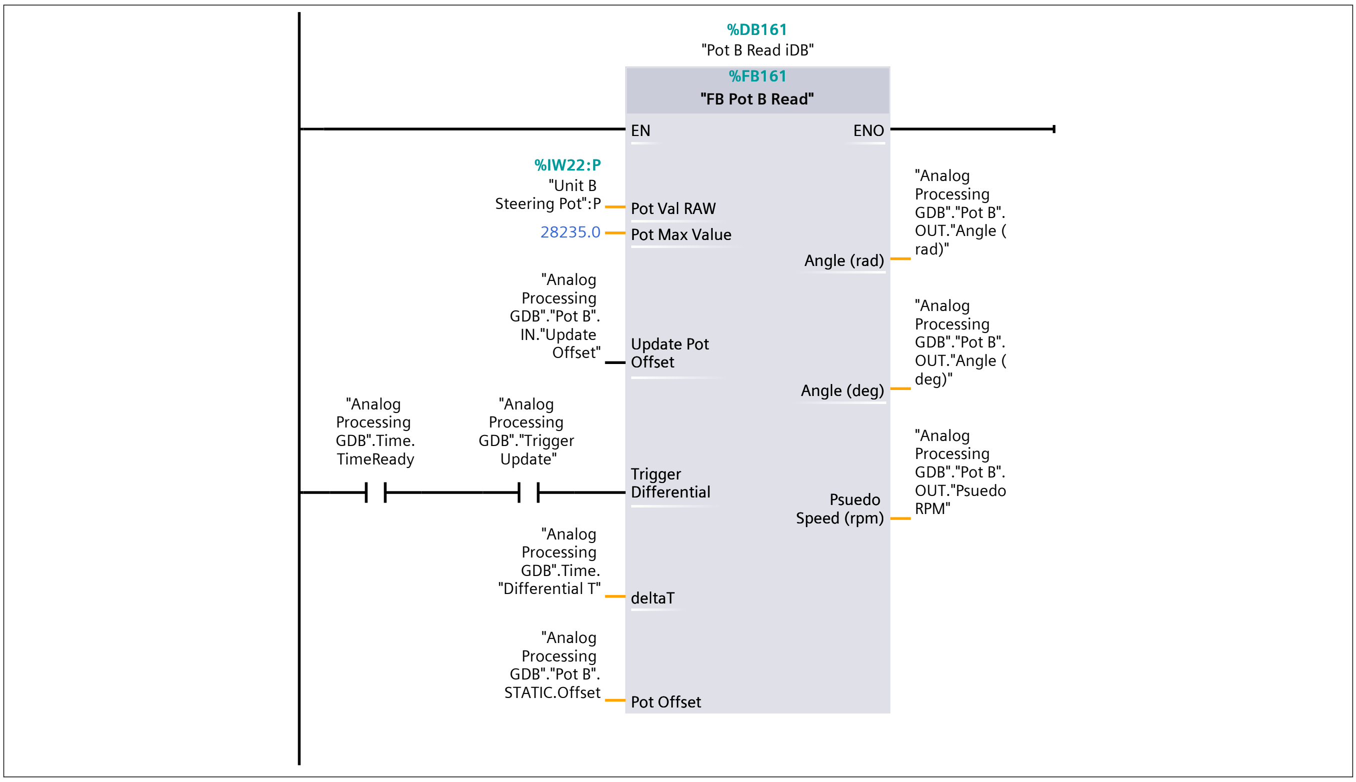
Network 1: Flip bit

```
IF "Analog Processing GDB"."Trigger Update" = FALSE THEN
    // If false flip to true
    "Analog Processing GDB"."Trigger Update" := TRUE;
ELSIF "Analog Processing GDB"."Trigger Update" = TRUE THEN
    // If true flip to flase
    "Analog Processing GDB"."Trigger Update" := FALSE;
ELSE
    // Should not occur = do nothing
    ;
END_IF;

```

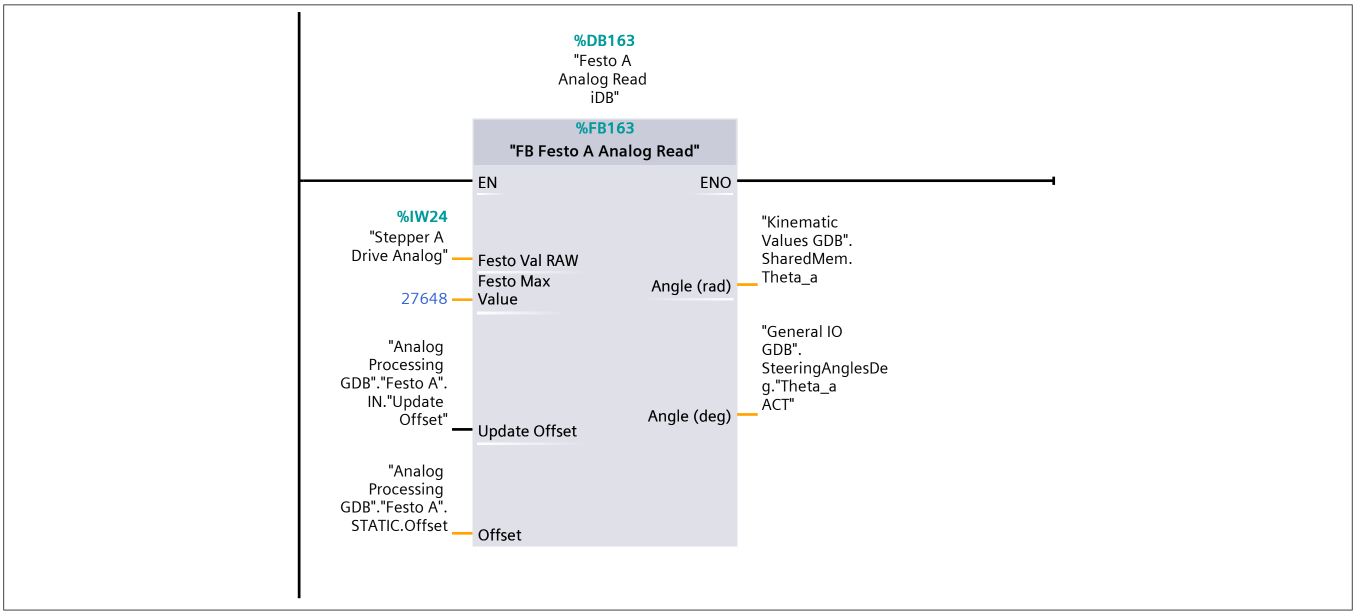
Network 2: Read Potentiometer Readings Pot A


%DB160
"Pot A Read iDB"
%FB160
"FB Pot A Read"
EN
ENO
%IW20:P
"Unit A Steering Pot":P
Pot Val RAW
28567.0
Pot Max Value
Angle (rad)
"Analog Processing GDB"."Pot A".OUT."Angle (rad)"
"Analog Processing GDB"."Pot A".IN."Update Offset"
Update Pot Offset
Angle (deg)
"Analog Processing GDB"."Pot A".OUT."Angle (deg)"
"Analog Processing GDB".Time.TimeReady
"Analog Processing GDB"."Trigger Update"
Trigger Differential
Psuedo Speed (rpm)
"Analog Processing GDB"."Pot A".OUT."Psuedo RPM"
"Analog Processing GDB".Time."Differential T"
deltaT
"Analog Processing GDB"."Pot A".STATIC.Offset
Pot Offset

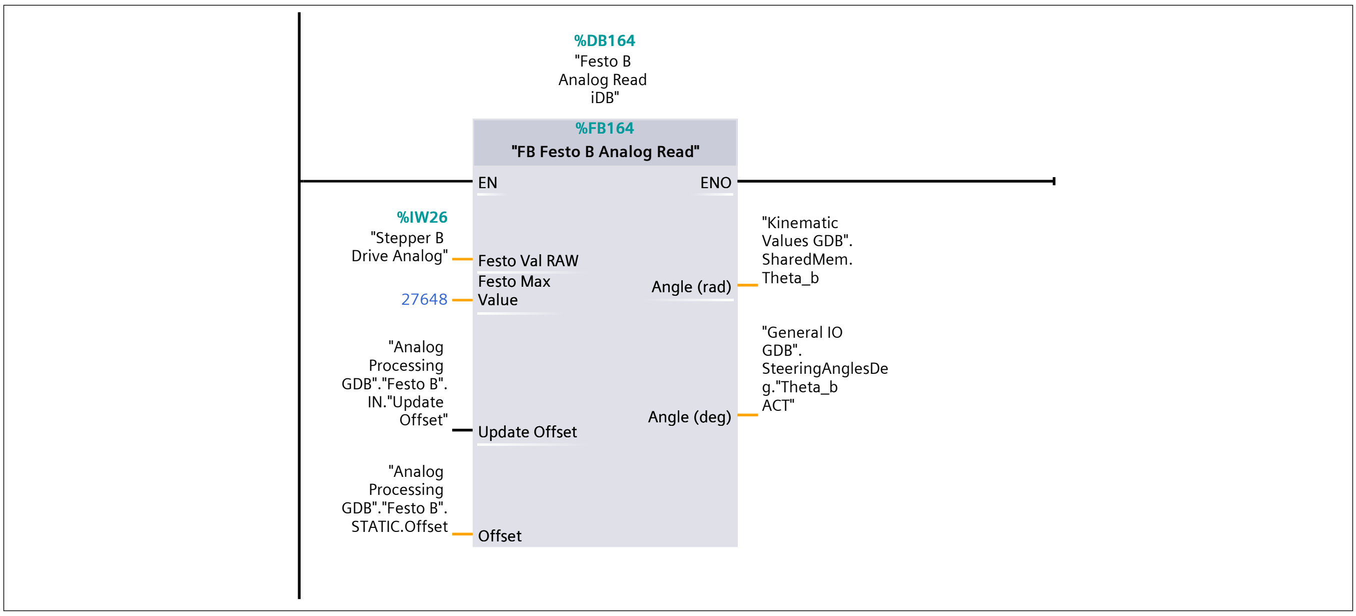
Network 3: Read Potentiometer Readings Pot B


%DB161
"Pot B Read iDB"
%FB161
"FB Pot B Read"
EN
ENO
%IW22:P
"Unit B Steering Pot":P
Pot Val RAW
28235.0
Pot Max Value
Angle (rad)
"Analog Processing GDB"."Pot B".OUT."Angle (rad)"
"Analog Processing GDB"."Pot B".IN."Update Offset"
Update Pot Offset
Angle (deg)
"Analog Processing GDB"."Pot B".OUT."Angle (deg)"
"Analog Processing GDB".Time.TimeReady
"Analog Processing GDB"."Trigger Update"
Trigger Differential
Psuedo Speed (rpm)
"Analog Processing GDB"."Pot B".OUT."Psuedo RPM"
"Analog Processing GDB".Time."Differential T"
deltaT
"Analog Processing GDB"."Pot B".STATIC.Offset
Pot Offset

Network 4: Read analog value from festo drive A



Network 5:



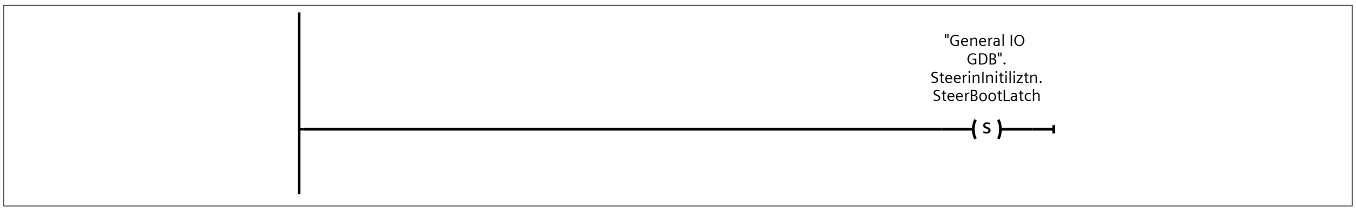
Startup [OB100]

Startup Properties					
General					
Name	Startup	**Number**	100	**Type**	OB
Language	LAD	**Numbering**	Automatic		
Information					
Title	"Complete Restart"	**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value
▼ Input		
LostRetentive	Bool	
LostRTC	Bool	
▼ Temp		
OB31 RET_VAL	Int	
OB31 STATUS	Word	
OB31 PHASE	UDInt	
OB31 CYCLE	UDInt	
Constant		

Network 1: Set First Run Latch

As part of the boot procedure set the diagnostics latch to a high state

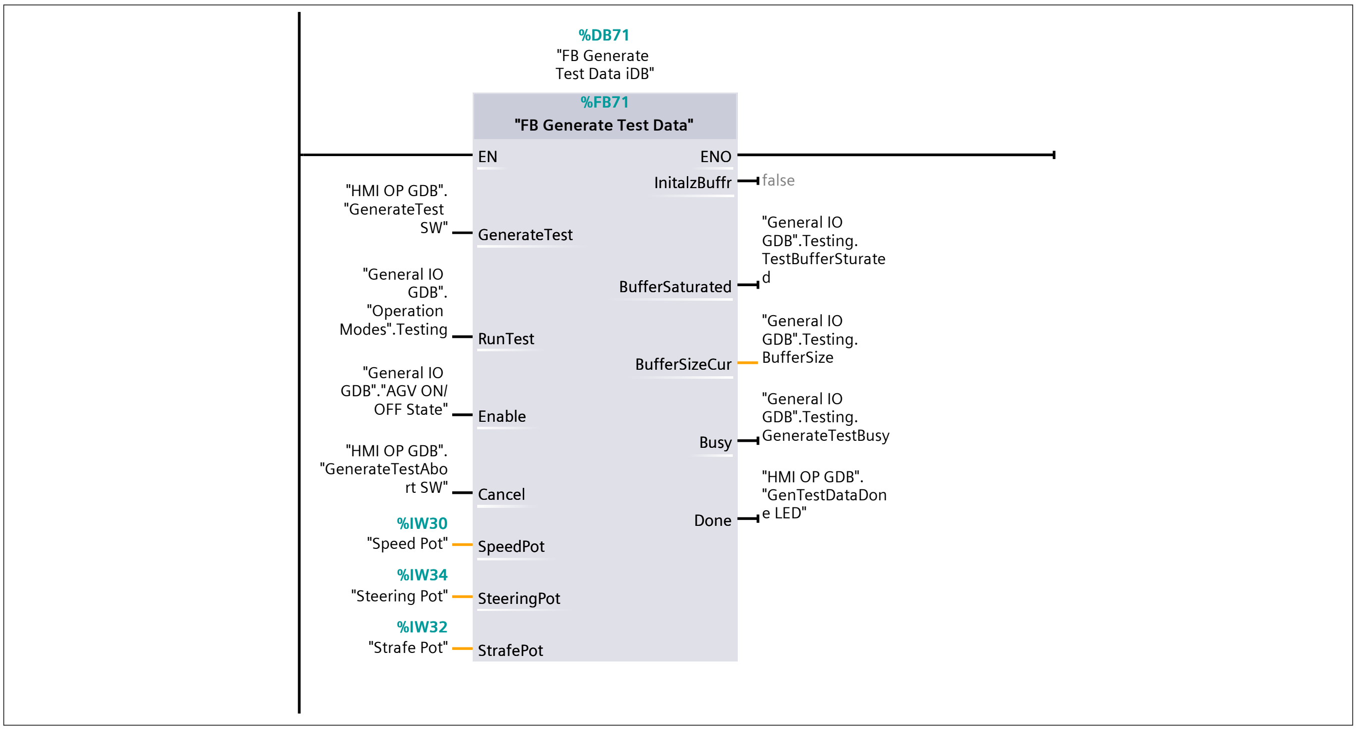


Testing Cyclic Interupt [OB33]

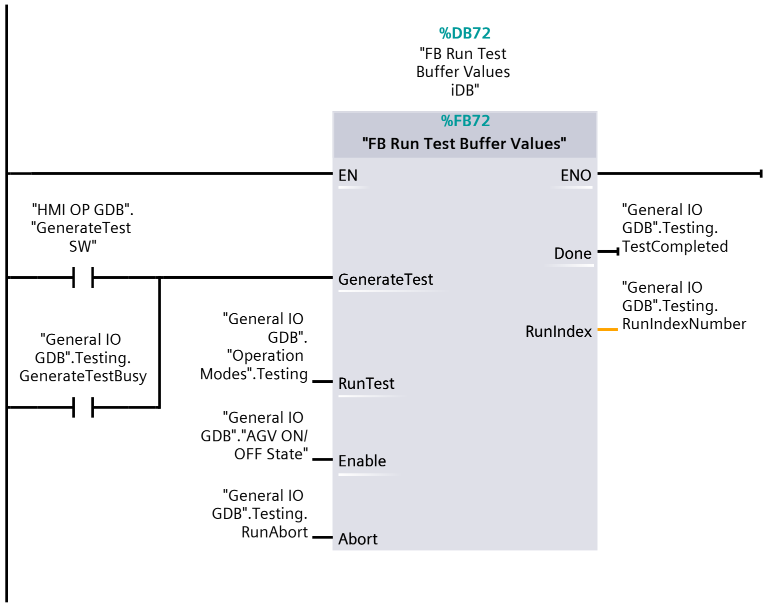
Testing Cyclic Interupt Properties					
General					
Name	Testing Cyclic Interupt	**Number**	33	**Type**	OB
Language	LAD	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value
▼ Input		
Initial_Call	Bool	
Event_Count	Int	
Temp		
Constant		

Network 1: Record Movements for Test Repeat



Network 2: Stream Stored pot data to Manual Mode Block


%DB72
"FB Run Test Buffer Values iDB"
%FB72
"FB Run Test Buffer Values"
EN
ENO
"HMI OP GDB". "GenerateTest SW"
"General IO GDB".Testing. GenerateTestBusy
GenerateTest
"General IO GDB". "Operation Modes".Testing
RunTest
"General IO GDB"."AGV ON/ OFF State"
Enable
"General IO GDB".Testing. RunAbort
Abort
Done
"General IO GDB".Testing. TestCompleted
RunIndex
"General IO GDB".Testing. RunIndexNumber

TimeSyncWithPLC [OB32]

TimeSyncWithPLC Properties					
General					
Name	TimeSyncWithPLC	**Number**	32	**Type**	OB
Language	LAD	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value
▼ Input		
Initial_Call	Bool	
Event_Count	Int	
Temp		
Constant		

Safety Main [OB123]

Safety Main Properties					
General					
Name	Safety Main	**Number**	123	**Type**	OB
Language	SCL	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value
▼ Input		
Initial_Call	Bool	
Event_Count	Int	

00. Global DBs

Comissioning Bits GDB [DB103]

Comissioning Bits GDB Properties					
General					
Name	Comissioning Bits GDB	**Number**	103	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Static			
Enable Drive	Bool	false	False
Enable Drive 2	Bool	false	False
Stop	Bool	false	False
Halt	Bool	false	False
Brake State	Bool	false	False
Execute Task	Bool	false	False
Acknowledge Faults	Bool	false	False
Home Drive	Bool	false	False
Jog Pos	Bool	false	False
Jog Neg	Bool	false	False
Setpoint Pos	DInt	0	False
Max Vel Percentage	Byte	16#14	False
Servo Testing	Struct		False
Wheel Orientation Test Bit	Bool	false	False
Update PolylineA	Bool	false	False
reset Intergrators	Bool	false	False
home drive A	Bool	false	False
home drive b	Bool	false	False
Test SteerStartUp	Bool	false	False
ackTest	Bool	false	False
temp stepper Angle	DInt	0	False
steeringhomeAux	Bool	false	False
ShutdownTest	Bool	false	False
FORCEHOME	Bool	false	False

00. Global DBs

General IO GDB [DB100]

General IO GDB Properties					
General					
Name	General IO GDB	**Number**	100	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Static			
Steering Pot Values	Struct		False
Operation Modes	Struct		False
SteerinInitiliztn	Struct		False
Wheel Pot Error	Bool	false	False
STO State	Bool	false	False
Estop Front Triggered	Bool	false	False
Estop Front Ack Required	Bool	false	False
Estop Rear Triggered	Bool	false	False
Estop Rear Ack Required	Bool	false	False
AGV ON/OFF State	Bool	false	False
Drive ON Home Overide	Bool	false	False
Homing	Bool	false	False
Battery Exchange State	Bool	false	False
N_Edge	Array[0..1] of Bool		False
P_Edge	Array[0..4] of Bool		False
MoveStates	Struct		False
SteeringAnglesDeg	Struct		False
ACTMotorRPMs	Struct		False
HMIControls	Struct		False
ZeroSystemDials	Struct		False
ShutdownSystem	Struct		False
Testing	Struct		False

00. Global DBs

HMI OP GDB [DB105]

HMI OP GDB Properties					
General					
Name	HMI OP GDB	**Number**	105	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Static			
AGVStateLED	Bool	false	False
SystemHomingLED	Bool	false	False
ManualModeLED	Bool	false	False
AutomaticModeLED	Bool	false	False
CommisionModeLED	Bool	false	False
AGVMovingLED	Bool	false	False
AGVWheelErrorLED	Bool	false	False
FrontEStopLED	Bool	false	False
RearEStopLED	Bool	false	False
AcknowledgeLED	Bool	false	False
SteeringAngleA_Dial	Real	0.0	False
SteeringAngleA	Real	0.0	False
SteeringAngleB_Dial	Real	0.0	False
SteeringAngleB	Real	0.0	False
YawAngleACT_Dial	Real	0.0	False
YawAngleACT	Real	0.0	False
YawAngleSET_Dial	Real	0.0	False
YawAngleSET	Real	0.0	False
XCompVel	Real	0.0	False
YCompVel	Real	0.0	False
SteeringRPM_A	Real	0.0	False
SteeringRPM_B	Real	0.0	False
TractionRPM_A	Real	0.0	False
TractionRPM_B	Real	0.0	False
AGVON/OFF_SW_FeedBCK	Bool	false	False
AGVON/OFF_SW	Bool	false	False
AGVON/OFF_JogUnit_SW	Bool	false	False
AGVON/OFF_JogAGV_SW	Bool	false	False
ManualMode_SW	Bool	false	False
AutoMode_SW	Bool	false	False
CommisinMode_SW	Bool	false	False
HomeSteering_SW	Bool	false	False
ResetUniversalFrame_SW	Bool	false	False
UnitA_Cali_CCW_SW	Bool	false	False
UnitA_Cali_CW_SW	Bool	false	False

Name	Data type	Start value	Retain
UnitB_Cali_CCW_SW	Bool	false	False
UnitB_Cali_CW_SW	Bool	false	False
CalibratePots_SW	Bool	false	False
CalibratePots_DONE_LED	Bool	false	False
CalibrateFCTAnalogs_SW	Bool	false	False
CalibrateFCTAnalogs_DONE_LED	Bool	false	False
CalibrateStepprSpd	Int	10	False
JogUnit SW	Bool	false	False
JogUnit LED	Bool	false	False
AGVJog SW	Bool	false	False
AGVJog SW FDBCK	Bool	false	False
AGVJog_Execute SW	Bool	false	False
AGVJog_Speed_SET	Int	200	False
AGVJog_Strafe_SET	Int	0	False
AGVJog_Strafe_FDBCK	Real	0.0	False
AGVJog_Acker_SET	Int	0	False
AGVJog_Acker_FDBCK	Real	0.0	False
UnitA_Servo_CCW_SW	Bool	false	False
UnitA_Servo_CW_SW	Bool	false	False
UnitB_Servo_CCW_SW	Bool	false	False
UnitB_Servo_CW_SW	Bool	false	False
ShutdownRouter	Bool	false	False
ShutdownIPC	Bool	false	False
Home2Zero SW	Bool	True	False
Home2SameAng SW	Bool	false	False
AllowAlignmnt	Bool	True	False
GenerateTest SW	Bool	false	False
GenerateTestAbort SW	Bool	false	False
GenerateTestWait4Run	Bool	false	False
GenerateTestBusy	Bool	false	False
GenTestReady	Bool	false	False
BufferSize	Int	0	False
RunTest SW	Bool	false	False
InitialisingBuffr LED	Bool	false	False
BufferSaturated LED	Bool	false	False
GenTestDataDone LED	Bool	false	False
RunTestDone LED	Bool	false	False
RunTestInProgress	Bool	false	False
RunTestProgess	Int	0	False
RunTestEnableAGV	Bool	false	False
RunTestAbort	Bool	false	False

00. Global DBs

Inputs GDB [DB101]

Inputs GDB Properties					
General					
Name	Inputs GDB	**Number**	101	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Static			
Battery PIP NO	Bool	false	False
Battery PIP NC	Bool	false	False
Eject Motor Feedback Eject	Bool	false	False
Eject Motor Feedback Insert	Bool	false	False
AGV on	Bool	false	False
ModeSelect	Bool	false	False
AckermanZero	Bool	false	False
StrafeLeft	Bool	false	False
StrafeRight	Bool	false	False
AckermanLeft	Bool	false	False
AckermanRight	Bool	false	False
Ack Pendant	Bool	false	False
Ack AGV	Bool	false	False
Reset Analog Pot	Bool	false	False
AGVPowerSW	Bool	false	False

00. Global DBs

Kinematic Values GDB [DB107]

Kinematic Values GDB Properties					
General					
Name	Kinematic Values GDB	**Number**	107	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Static			
SharedMem	Struct		True
ForwardKi	Struct		False
ReverseKi	Struct		False

00. Global DBs

Motor Feedback GDB [DB98]

Motor Feedback GDB Properties					
General					
Name	Motor Feedback GDB	**Number**	98	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Static			
RAW ACTUAL	Struct		False
SMOOTHED ACTUAL	Struct		False

00. Global DBs

Motor Setpoints GDB [DB99]

Motor Setpoints GDB Properties					
General					
Name	Motor Setpoints GDB	**Number**	99	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Static			
Update TRGR	Bool	false	False
Time	Struct		False
Homing	Struct		False
Manual Mode	Struct		False
Auto Mode	Struct		False
Commisioning Mode	Struct		False
SETPOINT	Struct		False

00. Global DBs

Outputs GDB [DB102]

Outputs GDB Properties					
General					
Name	Outputs GDB	**Number**	102	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Static			
RED Stack Light	Bool	false	False
YEL Stack Light	Bool	false	False
GRN Stack Light	Bool	false	False
Stepper A Enable	Bool	false	False
Stepper B Enable	Bool	false	False
Front Flashing Light	Bool	false	False
Disable Cooling	Bool	false	False
Eject Motor Eject	Bool	false	False
Eject Motor Insert	Bool	false	False
AGV On Status	Bool	false	False
Manual Mode	Bool	false	False
Automatic Mode	Bool	false	False
Wheel Pot Error	Bool	false	False
Commisiong Mode	Bool	false	False
Homing Mode	Bool	false	False
Unused OUT 3	Bool	false	False
Ack Required	Bool	false	False
AGVSelffKillSW	Bool	false	False

00. Global DBs

Analog Processing GDB [DB106]

Analog Processing GDB Properties					
General					
Name	Analog Processing GDB	**Number**	106	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Static			
Trigger Update	Bool	false	False
Time	Struct		True
Pot A	Struct		True
Pot B	Struct		True
Festo A	Struct		True
Festo B	Struct		True

00. Global DBs

Intergration Values GDB [DB109]

Intergration Values GDB Properties					
General					
Name	Intergration Values GDB	**Number**	109	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Static			
Body Yaw Rate Intergrator	Struct		False
Unit Angle Intergrators	Struct		False

00. Global DBs

Alarms GDB [DB120]

Alarms GDB Properties					
General					
Name	Alarms GDB	**Number**	120	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Static			
StepperAlarmWord	Word	16#0	False
ServoAlarmWord	Word	16#0	False
STOAlarmWord	Word	16#0	False
WheelAlignAlarmWord	Word	16#0	False
GeneralAlarmsWord	Word	16#0	False

00. Global DBs

Test Buffer GDB [DB70]

Test Buffer GDB Properties					
General					
Name	Test Buffer GDB	**Number**	70	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Static			
TestBuffer	Struct		False
StreamedData	Struct		False

Instance DBs

Steering Initialisation Sequencer iDB [DB157]

Steering Initialisation Sequencer iDB Properties					
General					
Name	Steering Initialisation Sequencer iDB	**Number**	157	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
RunBoot	Bool	false	False
RunHMI	Bool	false	False
EnableAutoAlignment	Bool	false	False
MustBeZero	Bool	false	False
STOState	Bool	false	False
OFF_SQ	Bool	false	False
INIT_SQ	Bool	false	False
ACK_EF	Bool	false	False
S_PREV	Bool	false	False
S_NEXT	Bool	false	False
SW_AUTO	Bool	false	False
SW_TAP	Bool	false	False
SW_TOP	Bool	false	False
SW_MAN	Bool	false	False
S_SEL	Int	0	False
S_ON	Bool	false	False
S_OFF	Bool	false	False
T_PUSH	Bool	false	False
▼ Output			
Busy	Bool	false	False
WheelOrientationValid	Bool	false	False
Done	Bool	false	False
ManuallyAlignWheels	Bool	false	False
ActivateSteppers	Bool	false	False
RunDeviationTest_Zero	Bool	false	False
RunWheelAlignment	Bool	false	False
RunDeviationTest_Align	Bool	false	False
ZeroCMMS-ST_Encoders_A	Bool	false	False
ZeroCMMS-ST_Encoders_B	Bool	false	False
ZeroHMIDials	Bool	false	False
S_NO	Int	0	False
S_MORE	Bool	false	False
S_ACTIVE	Bool	false	False

Name	Data type	Start value	Retain
ERR_FLT	Bool	false	False
AUTO_ON	Bool	false	False
TAP_ON	Bool	false	False
TOP_ON	Bool	false	False
MAN_ON	Bool	false	False
▼ InOut			
DeviationTest_ZeroPass	Bool	false	False
DeviationTest_ZeroFail	Bool	false	False
DeviationTest_AlignPass	Bool	false	False
DeviationTest_AlignFail	Bool	false	False
AlignmentComplete	Bool	false	False
HomingComplete	Bool	false	False
▼ Static			
RT_DATA	G7_RTDataPlus_V6		False
Trans1	G7_TransitionPlus_V6		False
Trans2	G7_TransitionPlus_V6		False
Trans3	G7_TransitionPlus_V6		False
Trans4	G7_TransitionPlus_V6		False
Trans5	G7_TransitionPlus_V6		False
Trans6	G7_TransitionPlus_V6		False
Trans7	G7_TransitionPlus_V6		False
Trans8	G7_TransitionPlus_V6		False
Trans9	G7_TransitionPlus_V6		False
Trans10	G7_TransitionPlus_V6		False
Trans11	G7_TransitionPlus_V6		False
Trans12	G7_TransitionPlus_V6		False
Trans13	G7_TransitionPlus_V6		False
Trans15	G7_TransitionPlus_V6		False
Trans16	G7_TransitionPlus_V6		False
Trans17	G7_TransitionPlus_V6		False
Trans19	G7_TransitionPlus_V6		False
Trans20	G7_TransitionPlus_V6		False
Trans21	G7_TransitionPlus_V6		False
Trans22	G7_TransitionPlus_V6		False
Trans23	G7_TransitionPlus_V6		False
Trans25	G7_TransitionPlus_V6		False
Trans26	G7_TransitionPlus_V6		False
Trans27	G7_TransitionPlus_V6		False
Trans28	G7_TransitionPlus_V6		False
Trans29	G7_TransitionPlus_V6		False
Trans30	G7_TransitionPlus_V6		False
Trans31	G7_TransitionPlus_V6		False
Trans32	G7_TransitionPlus_V6		False
Null	G7_StepPlus_V6		False
DeviationTest Zero'd Test1	G7_StepPlus_V6		False
DeviationTest Aligned Test1	G7_StepPlus_V6		False
DeviationTest_Zero's Pass	G7_StepPlus_V6		False
DeviationTest Zero'd Test2	G7_StepPlus_V6		False

Name	Data type	Start value	Retain
Initialise	G7_StepPlus_V6		False
DeviationTest Zero'd Test3	G7_StepPlus_V6		False
Initalise Alignment	G7_StepPlus_V6		False
Auto Alignment	G7_StepPlus_V6		False
ManualAlignment	G7_StepPlus_V6		False
ZeroEncoders2	G7_StepPlus_V6		False
HomingDone	G7_StepPlus_V6		False
Zero Encoders	G7_StepPlus_V6		False
DeactivateSteppers	G7_StepPlus_V6		False
AlignmentDone	G7_StepPlus_V6		False
DeviationTest Aligned Test2	G7_StepPlus_V6		False
DeviationTest Aligned Test3	G7_StepPlus_V6		False
InitializeEncoders	G7_StepPlus_V6		False
Step21	G7_StepPlus_V6		False
Zero HMI Dials 2	G7_StepPlus_V6		False
ZeroHMIDIals 1	G7_StepPlus_V6		False

Instance DBs

Steering Check & Alignment iDB [DB156]

Steering Check & Alignment iDB Properties					
General					
Name	Steering Check & Alignment iDB	**Number**	156	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
RunBoot	Bool	false	False
RunHMI	Bool	false	False
EnableAutoAlignment	Bool	false	False
MustBeZero	Bool	false	False
STOState	Bool	false	False
PotA_Angle	Real	0.0	False
PotB_Angle	Real	0.0	False
AngleTollerance	Real	0.0	False
▼ Output			
Busy	Bool	false	False
WheelOrientationValid	Bool	false	False
CMMS-ST_EncoderZero_A	Bool	false	False
CMMS-ST_EncoderZero_B	Bool	false	False
ZeroHMIDials	Bool	false	False
ActivateSteppers	Bool	false	False
StepperA_Speed	Real	0.0	False
StepperB_Speed	Real	0.0	False
ManualAlignmentRequired	Bool	false	False
Done	Bool	false	False
InOut			
▼ Static			
UnitA_OriTestPass	Bool	false	False
UnitB_OriTestPass	Bool	false	False
ABSWheelDiff	Real	0.0	False
AlignmentDoneA	Bool	false	False
AlignmentDoneB	Bool	false	False
ErrorA	Real	0.0	False
ErrorB	Real	0.0	False
DeviationTest_ZeroPass	Bool	false	False
DeviationTest_ZeroFail	Bool	false	False
AlignmentComplete	Bool	false	False
DeviationTest_AlignPass	Bool	false	False
DeviationTest_AlignFail	Bool	false	False

Name	Data type	Start value	Retain
HomingComplete	Bool	false	False
IEC_Timer_0_Instance	TON_TIME		False
IEC_Timer_0_Instance_1	TP_TIME		False
IEC_Timer_0_Instance_2	TON_TIME		False
StepperEnable_Internal_1	Bool	false	False
StepperEnable_Internal_2	Bool	false	False
IEC_Timer_0_Instance_3	TON_TIME		False
SequencerError	Bool	false	False

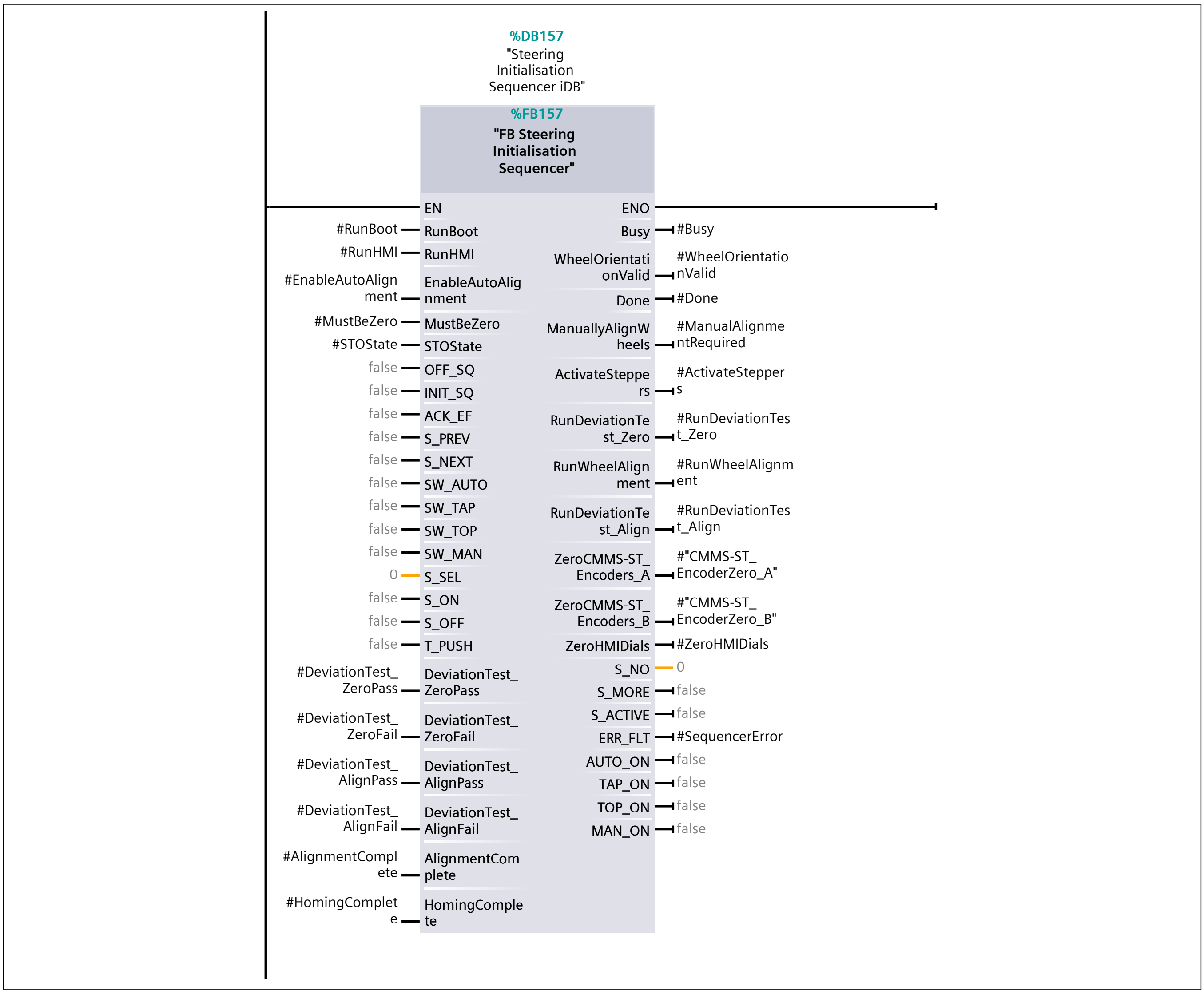
FB Steering Check & Alignment [FB4]

FB Steering Check & Alignment Properties					
General					
Name	FB Steering Check & Alignment	**Number**	4	**Type**	FB
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
▼ Input			
RunBoot	Bool	false	Non-retain
RunHMI	Bool	false	Non-retain
EnableAutoAlignment	Bool	false	Non-retain
MustBeZero	Bool	false	Non-retain
STOState	Bool	false	Non-retain
PotA_Angle	Real	0.0	Non-retain
PotB_Angle	Real	0.0	Non-retain
AngleTollerance	Real	0.0	Non-retain
▼ Output			
Busy	Bool	false	Non-retain
WheelOrientationValid	Bool	false	Non-retain
CMMS-ST_EncoderZero_A	Bool	false	Non-retain
CMMS-ST_EncoderZero_B	Bool	false	Non-retain
ZeroHMIDials	Bool	false	Non-retain
ActivateSteppers	Bool	false	Non-retain
StepperA_Speed	Real	0.0	Non-retain
StepperB_Speed	Real	0.0	Non-retain
ManualAlignmentRequired	Bool	false	Non-retain
Done	Bool	false	Non-retain
InOut			
▼ Static			
UnitA_OriTestPass	Bool	false	Non-retain
UnitB_OriTestPass	Bool	false	Non-retain
ABSWheelDiff	Real	0.0	Non-retain
AlignmentDoneA	Bool	false	Non-retain
AlignmentDoneB	Bool	false	Non-retain
ErrorA	Real	0.0	Non-retain
ErrorB	Real	0.0	Non-retain
DeviationTest_ZeroPass	Bool	false	Non-retain
DeviationTest_ZeroFail	Bool	false	Non-retain
AlignmentComplete	Bool	false	Non-retain
DeviationTest_AlignPass	Bool	false	Non-retain
DeviationTest_AlignFail	Bool	false	Non-retain
HomingComplete	Bool	false	Non-retain

Name	Data type	Default value	Retain
IEC_Timer_0_Instance	TON_TIME		Non-retain
IEC_Timer_0_Instance_1	TP_TIME		Non-retain
IEC_Timer_0_Instance_2	TON_TIME		Non-retain
StepperEnable_Internal_1	Bool	false	Non-retain
StepperEnable_Internal_2	Bool	false	Non-retain
IEC_Timer_0_Instance_3	TON_TIME		Non-retain
SequencerError	Bool	false	Non-retain
▼ Temp			
RunDeviationTest_Zero	Bool		
RunWheelAlignment	Bool		
RunDeviationTest_Align	Bool		
RunDrivesHoming	Bool		
Constant			

Network 1: Sequencer


%DB157
"Steering Initialisation Sequencer iDB"
%FB157
"FB Steering Initialisation Sequencer"
EN
ENO
#RunBoot — RunBoot
Busy — #Busy
#RunHMI — RunHMI
WheelOrientationValid — #WheelOrientationValid
#EnableAutoAlignment — EnableAutoAlignment
Done — #Done
#MustBeZero — MustBeZero
ManuallyAlignWheels — #ManualAlignmentRequired
#STOState — STOState
false — OFF_SQ
ActivateSteppers — #ActivateSteppers
false — INIT_SQ
false — ACK_EF
RunDeviationTest_Zero — #RunDeviationTest_Zero
false — S_PREV
false — S_NEXT
RunWheelAlignment — #RunWheelAlignment
false — SW_AUTO
false — SW_TAP
RunDeviationTest_Align — #RunDeviationTest_Align
false — SW_TOP
false — SW_MAN
ZeroCMMS-ST_Encoders_A — #"CMMS-ST_EncoderZero_A"
0 — S_SEL
false — S_ON
ZeroCMMS-ST_Encoders_B — #"CMMS-ST_EncoderZero_B"
false — S_OFF
false — T_PUSH
ZeroHMIDials — #ZeroHMIDials
S_NO — 0
#DeviationTest_ZeroPass — DeviationTest_ZeroPass
S_MORE — false
S_ACTIVE — false
#DeviationTest_ZeroFail — DeviationTest_ZeroFail
ERR_FLT — #SequencerError
AUTO_ON — false
#DeviationTest_AlignPass — DeviationTest_AlignPass
TAP_ON — false
TOP_ON — false
#DeviationTest_AlignFail — DeviationTest_AlignFail
MAN_ON — false
#AlignmentComplete — AlignmentComplete
#HomingComplete — HomingComplete

Network 2: DeviationTest_Zero Subroutine

Subroutine that finds out if both units's steering is at zero degrees and returns the result to the sequencer

```
//This condition occurs with the orientation of the drives must line up with the zero degree point of the system
//Each unit will have to be tested and the orientation between (0 -> deviation/2) and ([360-deviation/2] -> 360)
//
    //Check if sequencer wants test run
    IF #RunDeviationTest_Zero = TRUE THEN
        
        //check if Unit A passes test
        //
        IF (#PotA_Angle < (#AngleTollerance / 2)) OR (#PotA_Angle > ("2pi" - (#AngleTollerance / 2))) THEN
            //Angle between tollerances, thus wheel orientation valid
            #UnitA_OriTestPass := TRUE;
        ELSE
            #UnitA_OriTestPass := FALSE;
        END_IF;
        
        //Check if Unit B passes test
        //
        IF (#PotB_Angle < (#AngleTollerance / 2)) OR (#PotB_Angle > ("2pi" - (#AngleTollerance / 2))) THEN
            //Angle is between tollerance limits, thus wheel orienation valid
            #UnitB_OriTestPass := TRUE;
        ELSE
            #UnitB_OriTestPass := FALSE;
        END_IF;
        
        //Check if both tests were a pass
        //
        IF (#UnitA_OriTestPass = TRUE) AND (#UnitB_OriTestPass = TRUE) THEN
            //Both tests were a pass
            //
            #DeviationTest_ZeroPass := TRUE;
            #DeviationTest_ZeroFail := FALSE;
        ELSE
            //Both tests were not a pass
            #DeviationTest_ZeroPass := FALSE;
            #DeviationTest_ZeroFail := TRUE;
            
        END_IF;
        
    END_IF;
```

Network 3: DeviationTest_Align Subroutine

Subroutine finds out if the wheels are facting the same direction and not necessarily the zero degree direction

```
// Wheels only need to have the same orienation, the they do not need to face the zero degree angle
    //
    IF #RunDeviationTest_Align = TRUE THEN
```

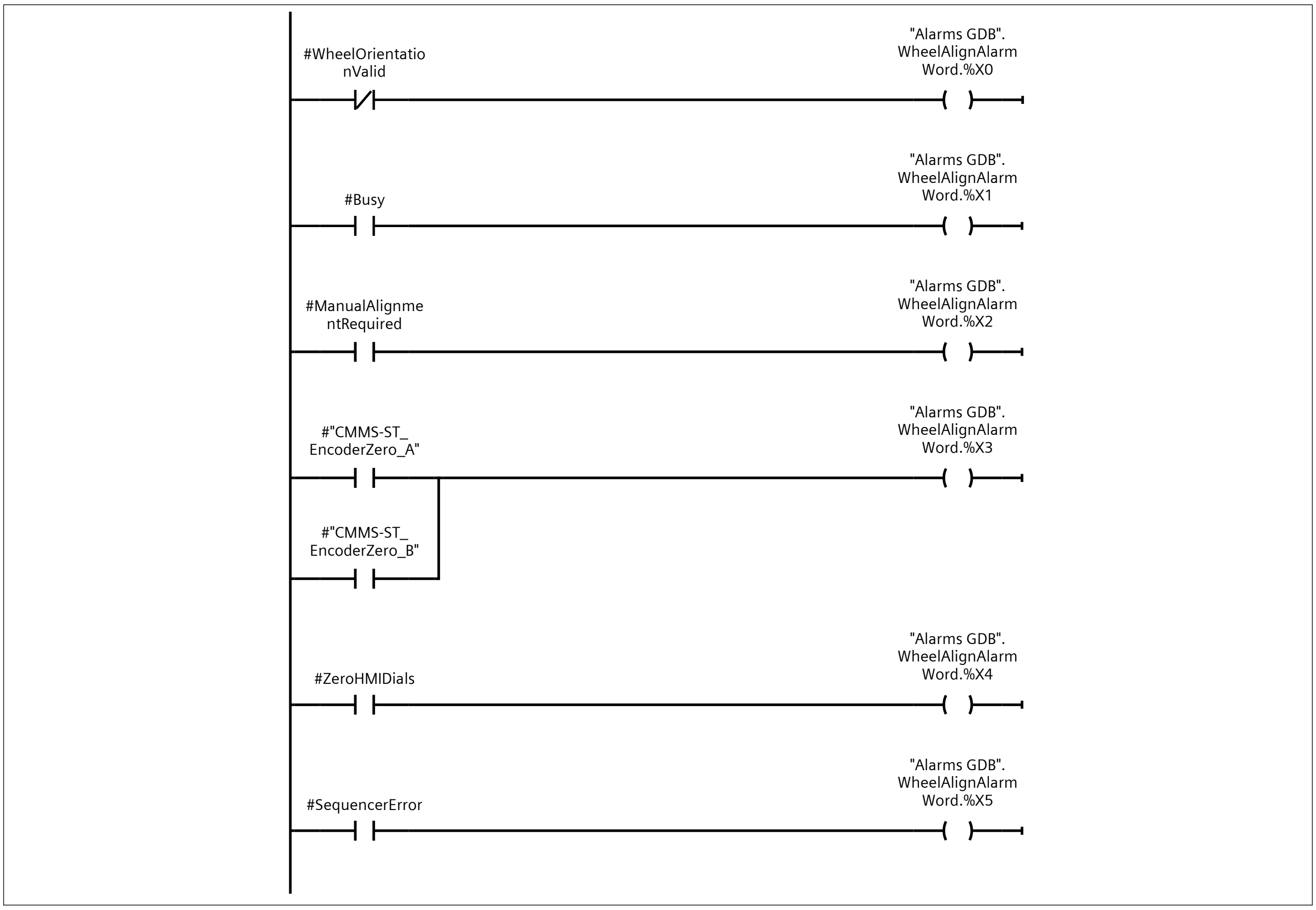
```

        //Find the absolute difference between the wheels
        #ABSWheelDiff := ABS(#PotA_Angle - #PotB_Angle);

        //Check is absolute differenc is within the angle deviation tollerance
        //
        IF #ABSWheelDiff <= #AngleTollerance THEN
            //Wheel deviation less than allowable tollerance, thus a pass
            //
            #DeviationTest_AlignPass := TRUE;
            #DeviationTest_AlignFail := FALSE;

        ELSE
            //Test was not a pass
            //
            #DeviationTest_AlignPass := FALSE;
            #DeviationTest_AlignFail := TRUE;

        END_IF;
    END_IF;
```

Network 4: WheelAlignmet Subroutine

Subroutine that aligns the wheel of the AGV to the zero degree position

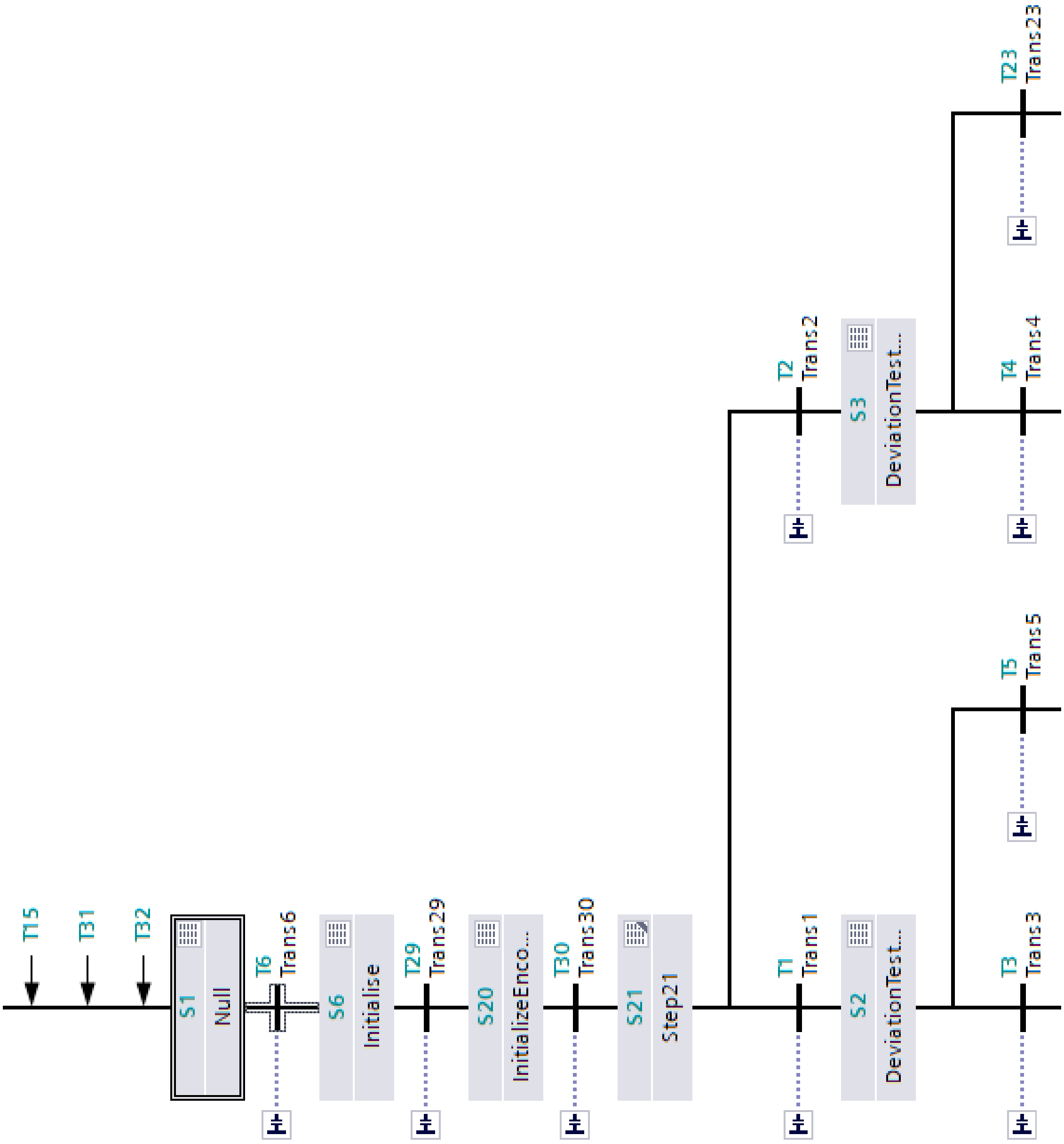
```
//Subroutine that aligns the wheel of the AGV to the zero degree position
    //

    IF #RunWheelAlignment = TRUE THEN
        //sequencer requires aligment to be run

        //UNIT A
        //
*********************************************************************************
************************************
        IF #PotA_Angle <= "pi" THEN
            //Pot angle is between 0 and 180 degrees

            //Determine error to zero point
            #ErrorA := ABS(#PotA_Angle);

            IF #ErrorA > (#AngleTollerance / 2) THEN
                //Error is larger than allowed, thus move system to reduce error
value

                IF #ErrorA < 0.0872665 THEN
                    // Error is less than 5 degrees from desired location = slow
speed
                    #StepperA_Speed := -0.1;
                ELSE
                    // Error is greater than 5 degrees = fast speed
                    #StepperA_Speed := -0.3;
                END_IF;

                #AlignmentDoneA := FALSE;

            ELSIF #ErrorA <= (#AngleTollerance / 2) THEN
                //Error is within allowance
                #StepperA_Speed := 0.0;
                #AlignmentDoneA := TRUE;
            END_IF;

        ELSIF #PotA_Angle > "pi" THEN
            //Pot angle is between 180 and 360 degrees

            //Determine error to zero point
            #ErrorA := ABS("2pi" - #PotA_Angle);

            IF #ErrorA > (#AngleTollerance / 2) THEN
                //Error is larger than allowed, thus move system to reduce error value

                IF #ErrorA < 0.0872665 THEN
                    // Error is less than 5 degrees from desired location = slow speed
                    #StepperA_Speed := 0.1;
                ELSE
                    // Error is greater than 5 degrees = fast speed
                    #StepperA_Speed := 0.3;
                END_IF;

                #AlignmentDoneA := FALSE;

            ELSIF #ErrorA <= (#AngleTollerance / 2) THEN
                //Error is within allowance
                #StepperA_Speed := 0;
                #AlignmentDoneA := TRUE;
            END_IF;
        END_IF;



        //UNIT B
        //******************************************************************************************************************
        IF #PotB_Angle <= "pi" THEN
            //Pot angle is between 0 and 180 degrees
            //
            #ErrorB := ABS(#PotB_Angle);

            IF #ErrorB > (#AngleTollerance / 2) THEN
                //Error is larger than allowed, thus move system to reduce error value

                IF #ErrorB < 0.0872665 THEN
                    // Error is less than 5 degrees from desired location = slow speed
                    #StepperB_Speed := -0.1;
                ELSE
                    // Error is greater than 5 degrees = fast speed
                    #StepperB_Speed := -0.3;
                END_IF;
                
                #AlignmentDoneB := FALSE;
                
            ELSIF #ErrorB <= (#AngleTollerance / 2) THEN
                //Error is within allowance
                #StepperB_Speed := 0.0;
                #AlignmentDoneB := TRUE;
            END_IF;
            
        ELSIF #PotB_Angle > "pi" THEN
            //Pot angle is between 180 and 360 degrees
            
            //Determine error to zero point
            #ErrorB := ABS("2pi" - #PotB_Angle);
            
            IF #ErrorB > (#AngleTollerance / 2) THEN
                //Error is larger than allowed, thus move system to reduce error value
                
                IF #ErrorB < 0.0872665 THEN
                    // Error is less than 5 degrees from desired location = slow speed
                    #StepperB_Speed := 0.1;
                ELSE
                    // Error is greater than 5 degrees = fast speed
                    #StepperB_Speed := 0.3;
                END_IF;
                
                #AlignmentDoneB := FALSE;
                
            ELSIF #ErrorB <= (#AngleTollerance / 2) THEN
                //Error within allowance
                #StepperB_Speed := 0.0;
                #AlignmentDoneB := TRUE;
            END_IF;
        END_IF;
        
        
        //CHECK ALIGNMENT OF UNIT A AND B
        //*******************************************************************************************************************
        IF #AlignmentDoneA AND #AlignmentDoneB THEN
            //Both units have completed alignmnet
            #AlignmentComplete := TRUE;
        ELSE
            #AlignmentComplete := FALSE;
        END_IF;
    END_IF;
    
```

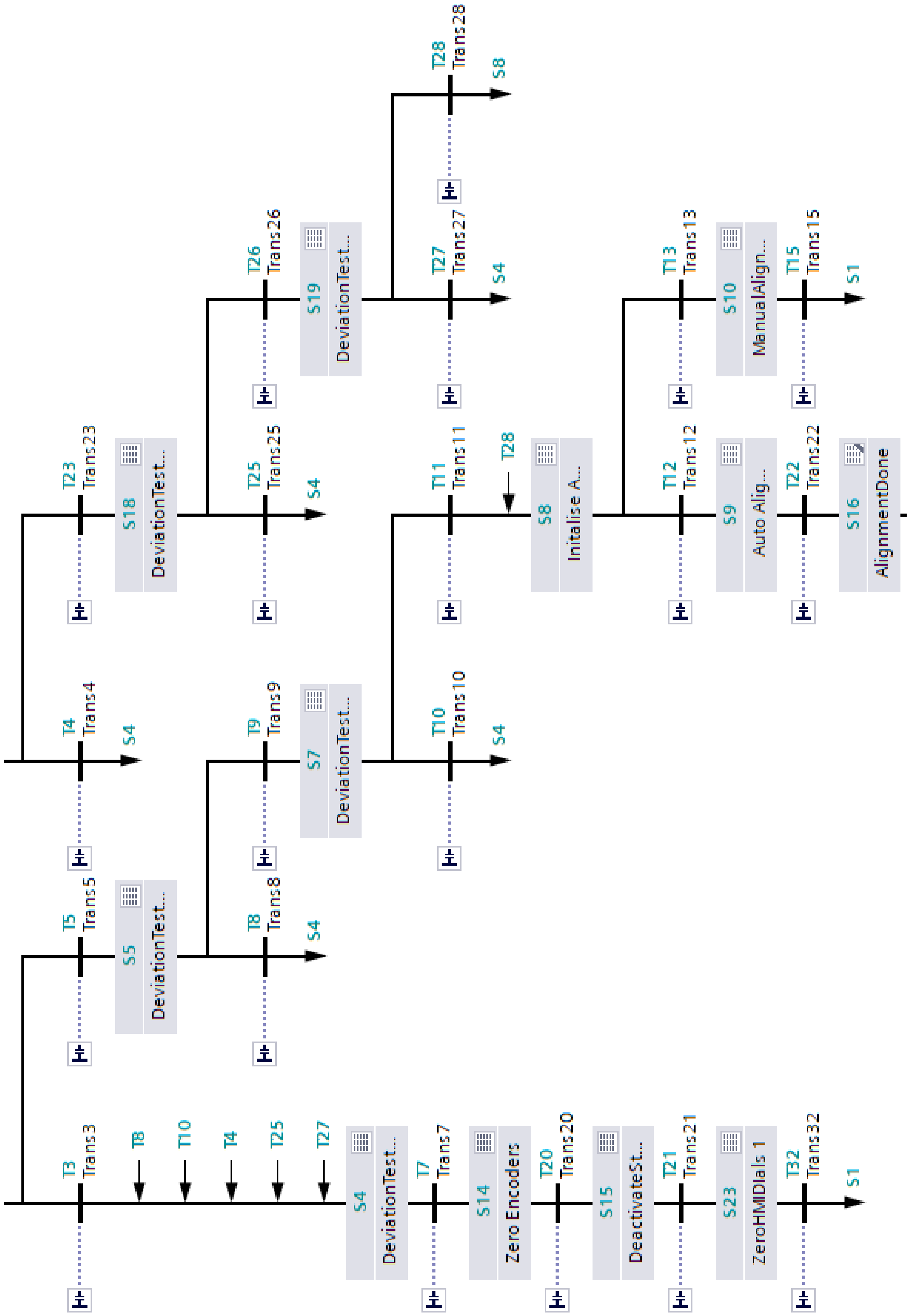
Network 5: Alarm Words

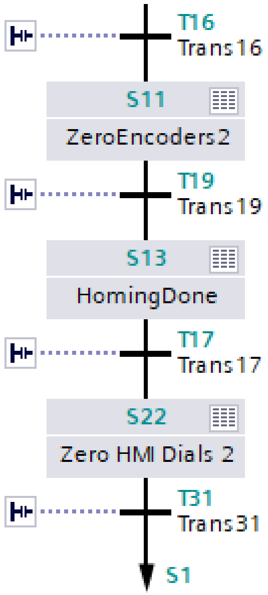

#WheelOrientationValid
"Alarms GDB".WheelAlignAlarmWord.%X0
#Busy
"Alarms GDB".WheelAlignAlarmWord.%X1
#ManualAlignmentRequired
"Alarms GDB".WheelAlignAlarmWord.%X2
#"CMMS-ST_EncoderZero_A"
"Alarms GDB".WheelAlignAlarmWord.%X3
#"CMMS-ST_EncoderZero_B"
#ZeroHMIDials
"Alarms GDB".WheelAlignAlarmWord.%X4
#SequencerError
"Alarms GDB".WheelAlignAlarmWord.%X5

FB Steering Initialisation Sequencer [FB157]

This had to be imported manually as print function in TIA failed to render this FB, instead TIA crashed.




T3
Trans3
T8
T10
T4
T25
T27
S4
DeviationTest...
T7
Trans7
S14
Zero Encoders
T20
Trans20
S15
DeactivateSt...
T21
Trans21
S23
ZeroHMIDIals 1
T32
Trans32
S1
T5
Trans5
S5
DeviationTest...
T8
Trans8
S4
T9
Trans9
S7
DeviationTest...
T10
Trans10
S4
T4
Trans4
S4
T23
Trans23
S18
DeviationTest...
T25
Trans25
S4
T26
Trans26
S19
DeviationTest...
T27
Trans27
S4
T28
Trans28
S8
T11
Trans11
T28
S8
Initalise A...
T12
Trans12
S9
Auto Alig...
T22
Trans22
S16
AlignmentDone
T13
Trans13
S10
ManualAlign...
T15
Trans15
S1


T16
Trans16
S11
ZeroEncoders2
T19
Trans19
S13
HomingDone
T17
Trans17
S22
Zero HMI Dials 2
T31
Trans31
S1

Doctoral AGV / Low Level Systems / Gertrude Main PLC [CPU 1512SP F-1 PN] / Program blocks / 01. Wheel Alignment

FB Steering Initialisation Sequencer [FB157]

FB Steering Initialisation Sequencer Properties					
General					
Name	FB Steering Initialisation Sequencer	**Number**	157	**Type**	FB
Language	GRAPH	**Numbering**	Manual	**Network language**	LAD
Block version	V6.0				
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

FB Steering Initialisation Sequencer			
Name	**Data type**	**Default value**	**Retain**
▼ Input			
RunBoot	Bool	false	Non-retain
RunHMI	Bool	false	Non-retain
EnableAutoAlignment	Bool	false	Non-retain
MustBeZero	Bool	false	Non-retain
STOState	Bool	false	Non-retain
OFF_SQ	Bool	false	Non-retain
INIT_SQ	Bool	false	Non-retain
ACK_EF	Bool	false	Non-retain
S_PREV	Bool	false	Non-retain
S_NEXT	Bool	false	Non-retain
SW_AUTO	Bool	false	Non-retain
SW_TAP	Bool	false	Non-retain
SW_TOP	Bool	false	Non-retain
SW_MAN	Bool	false	Non-retain
S_SEL	Int	0	Non-retain
S_ON	Bool	false	Non-retain
S_OFF	Bool	false	Non-retain
T_PUSH	Bool	false	Non-retain
▼ Output			
Busy	Bool	false	Non-retain
WheelOrientationValid	Bool	false	Non-retain
Done	Bool	false	Non-retain
ManuallyAlignWheels	Bool	false	Non-retain
ActivateSteppers	Bool	false	Non-retain
RunDeviationTest_Zero	Bool	false	Non-retain
RunWheelAlignment	Bool	false	Non-retain
RunDeviationTest_Align	Bool	false	Non-retain
ZeroCMMS-ST_Encoders_A	Bool	false	Non-retain
ZeroCMMS-ST_Encoders_B	Bool	false	Non-retain
ZeroHMIDials	Bool	false	Non-retain

Name	Data type	Default value	Retain
S_NO	Int	0	Non-retain
S_MORE	Bool	false	Non-retain
S_ACTIVE	Bool	false	Non-retain
ERR_FLT	Bool	false	Non-retain
AUTO_ON	Bool	false	Non-retain
TAP_ON	Bool	false	Non-retain
TOP_ON	Bool	false	Non-retain
MAN_ON	Bool	false	Non-retain
▼ InOut			
DeviationTest_ZeroPass	Bool	false	Non-retain
DeviationTest_ZeroFail	Bool	false	Non-retain
DeviationTest_AlignPass	Bool	false	Non-retain
DeviationTest_AlignFail	Bool	false	Non-retain
AlignmentComplete	Bool	false	Non-retain
HomingComplete	Bool	false	Non-retain
▼ Static			
RT_DATA	G7_RTDataPlus_V6		Non-retain
Trans1	G7_TransitionPlus_V6		Non-retain
Trans2	G7_TransitionPlus_V6		Non-retain
Trans3	G7_TransitionPlus_V6		Non-retain
Trans4	G7_TransitionPlus_V6		Non-retain
Trans5	G7_TransitionPlus_V6		Non-retain
Trans6	G7_TransitionPlus_V6		Non-retain
Trans7	G7_TransitionPlus_V6		Non-retain
Trans8	G7_TransitionPlus_V6		Non-retain
Trans9	G7_TransitionPlus_V6		Non-retain
Trans10	G7_TransitionPlus_V6		Non-retain
Trans11	G7_TransitionPlus_V6		Non-retain
Trans12	G7_TransitionPlus_V6		Non-retain
Trans13	G7_TransitionPlus_V6		Non-retain
Trans15	G7_TransitionPlus_V6		Non-retain
Trans16	G7_TransitionPlus_V6		Non-retain
Trans17	G7_TransitionPlus_V6		Non-retain
Trans19	G7_TransitionPlus_V6		Non-retain
Trans20	G7_TransitionPlus_V6		Non-retain

Name	Data type	Default value	Retain
Trans21	G7_Transition-Plus_V6		Non-retain
Trans22	G7_Transition-Plus_V6		Non-retain
Trans23	G7_Transition-Plus_V6		Non-retain
Trans25	G7_Transition-Plus_V6		Non-retain
Trans26	G7_Transition-Plus_V6		Non-retain
Trans27	G7_Transition-Plus_V6		Non-retain
Trans28	G7_Transition-Plus_V6		Non-retain
Trans29	G7_Transition-Plus_V6		Non-retain
Trans30	G7_Transition-Plus_V6		Non-retain
Trans31	G7_Transition-Plus_V6		Non-retain
Trans32	G7_Transition-Plus_V6		Non-retain
Null	G7_StepPlus_V6		Non-retain
DeviationTest Zero'd Test1	G7_StepPlus_V6		Non-retain
DeviationTest Aligned Test1	G7_StepPlus_V6		Non-retain
DeviationTest_Zero's Pass	G7_StepPlus_V6		Non-retain
DeviationTest Zero'd Test2	G7_StepPlus_V6		Non-retain
Initialise	G7_StepPlus_V6		Non-retain
DeviationTest Zero'd Test3	G7_StepPlus_V6		Non-retain
Initalise Alignment	G7_StepPlus_V6		Non-retain
Auto Alignment	G7_StepPlus_V6		Non-retain
ManualAlignment	G7_StepPlus_V6		Non-retain
ZeroEncoders2	G7_StepPlus_V6		Non-retain
HomingDone	G7_StepPlus_V6		Non-retain
Zero Encoders	G7_StepPlus_V6		Non-retain
DeactivateSteppers	G7_StepPlus_V6		Non-retain
AlignmentDone	G7_StepPlus_V6		Non-retain
DeviationTest Aligned Test2	G7_StepPlus_V6		Non-retain
DeviationTest Aligned Test3	G7_StepPlus_V6		Non-retain
InitializeEncoders	G7_StepPlus_V6		Non-retain
Step21	G7_StepPlus_V6		Non-retain
Zero HMI Dials 2	G7_StepPlus_V6		Non-retain
ZeroHMIDIals 1	G7_StepPlus_V6		Non-retain
Temp			
Constant			

Alarms

Enable alarms	True

Category	Category enabler	Display class
Error		0
Warning		0

Category	Category enabler	Display class
Info		0
Category 4		0
Category 5		0
Category 6		0
Category 7		0
Category 8		0

Category for interlocks	Error	**Subcategory 1 for interlocks**		**Subcategory 2 for interlocks**	

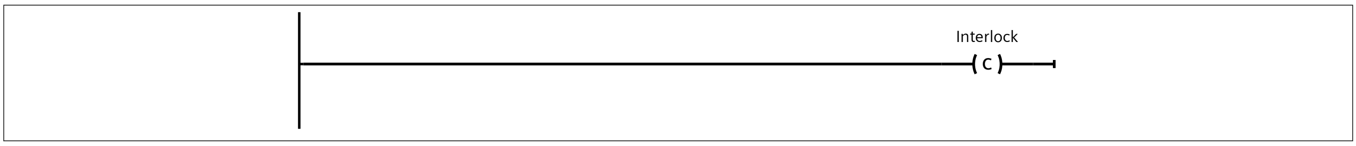
Category for supervisions	Error	**Subcategory 1 for supervisions**		**Subcategory 2 for supervisions**	

Category for GRAPH warnings	Warning	**Subcategory 1 for GRAPH warnings**		**Subcategory 2 for GRAPH warnings**	

S1 - [Initial step]:Null

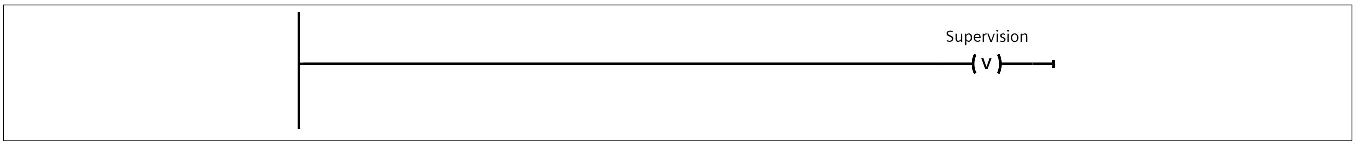
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

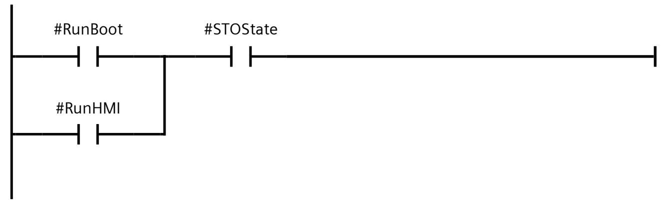
Supervision alarm	
Alarm text	



Actions:

Interlock	Event	Qualifier	Action
		R	#Busy
		S	#Done

T6:Trans6



Doctoral AGV / Low Level Systems / Gertrude Main PLC [CPU 1512SP F-1 PN] / Program blocks / 01. Wheel Alignment

FB Steering Initialisation Sequencer [FB157]

FB Steering Initialisation Sequencer Properties					
General					
Name	FB Steering Initialisation Sequencer	**Number**	157	**Type**	FB
Language	GRAPH	**Numbering**	Manual	**Network language**	LAD
Block version	V6.0				
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Alarms

Enable alarms	True

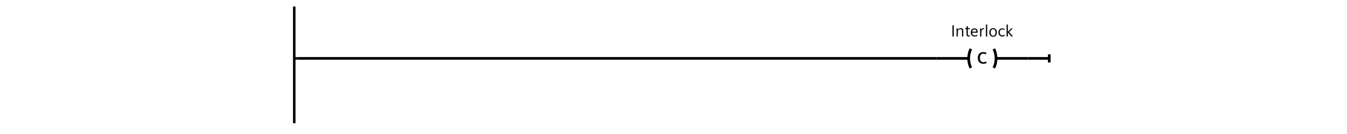
Category	Category enabler	Display class
Error		0
Warning		0
Info		0
Category 4		0
Category 5		0
Category 6		0
Category 7		0
Category 8		0

Category for interlocks	Error	**Subcategory 1 for interlocks**		**Subcategory 2 for interlocks**	
Category for supervisions	Error	**Subcategory 1 for supervisions**		**Subcategory 2 for supervisions**	
Category for GRAPH warnings	Warning	**Subcategory 1 for GRAPH warnings**		**Subcategory 2 for GRAPH warnings**	

S6:Initialise

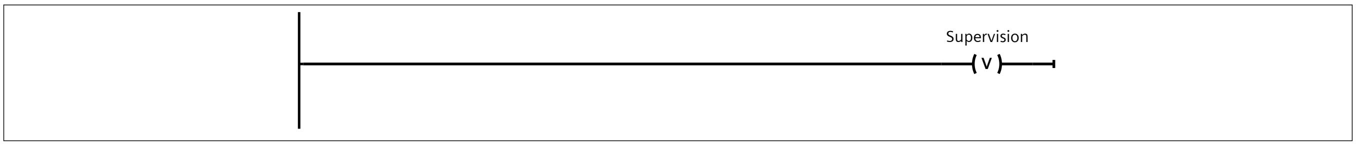
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

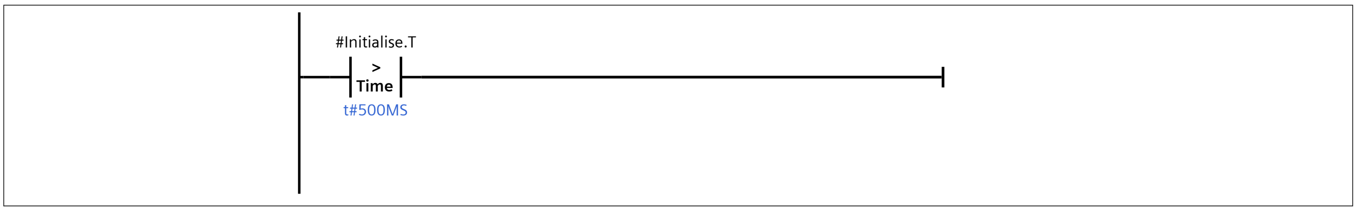
Supervision alarm	
Alarm text	



Actions:

Actions:			
Interlock	**Event**	**Qualifier**	**Action**
		R	#Done
		S	#Busy
		R	#WheelOrientationValid
		R	#ManuallyAlignWheels
		S	#ActivateSteppers

T29:Trans29



Doctoral AGV / Low Level Systems / Gertrude Main PLC [CPU 1512SP F-1 PN] / Program blocks / 01. Wheel Alignment

FB Steering Initialisation Sequencer [FB157]

FB Steering Initialisation Sequencer Properties					
General					
Name	FB Steering Initialisation Sequencer	**Number**	157	**Type**	FB
Language	GRAPH	**Numbering**	Manual	**Network language**	LAD
Block version	V6.0				
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Alarms

Enable alarms	True

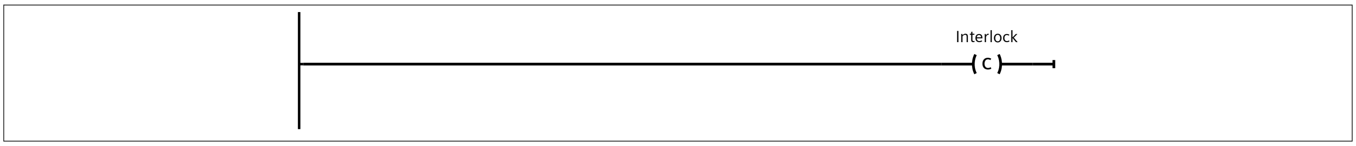
Category	Category enabler	Display class
Error		0
Warning		0
Info		0
Category 4		0
Category 5		0
Category 6		0
Category 7		0
Category 8		0

Category for interlocks	Error	**Subcategory 1 for interlocks**		**Subcategory 2 for interlocks**	
Category for supervisions	Error	**Subcategory 1 for supervisions**		**Subcategory 2 for supervisions**	
Category for GRAPH warnings	Warning	**Subcategory 1 for GRAPH warnings**		**Subcategory 2 for GRAPH warnings**	

S20:InitializeEncoders

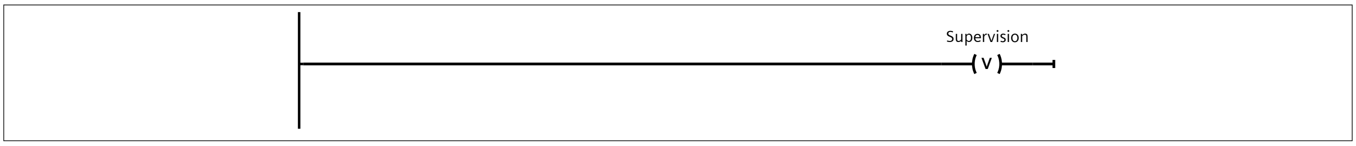
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

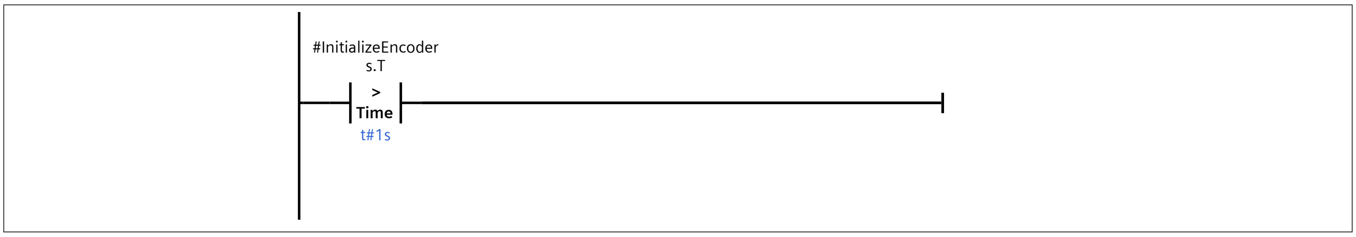
Supervision alarm	
Alarm text	



Actions:

Actions:			
Interlock	Event	Qualifier	Action
		N	#"ZeroCMMS-ST_Encoders_A"
		N	#"ZeroCMMS-ST_Encoders_B"

T30:Trans30



Doctoral AGV / Low Level Systems / Gertrude Main PLC [CPU 1512SP F-1 PN] / Program blocks / 01. Wheel Alignment

FB Steering Initialisation Sequencer [FB157]

FB Steering Initialisation Sequencer Properties					
General					
Name	FB Steering Initialisation Sequencer	**Number**	157	**Type**	FB
Language	GRAPH	**Numbering**	Manual	**Network language**	LAD
Block version	V6.0				
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Alarms

Enable alarms	True

Category	Category enabler	Display class
Error		0
Warning		0
Info		0
Category 4		0
Category 5		0
Category 6		0
Category 7		0
Category 8		0

Category for interlocks	Error	**Subcategory 1 for interlocks**		**Subcategory 2 for interlocks**	

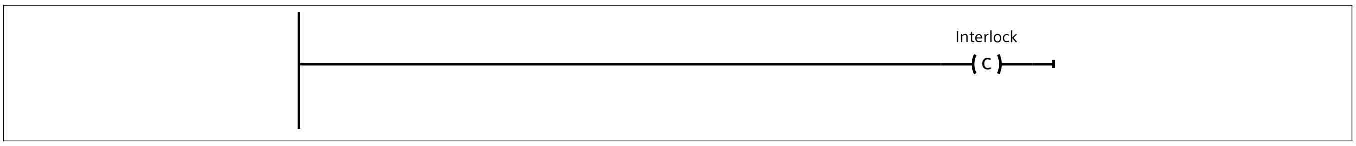
Category for supervisions	Error	**Subcategory 1 for supervisions**		**Subcategory 2 for supervisions**	

Category for GRAPH warnings	Warning	**Subcategory 1 for GRAPH warnings**		**Subcategory 2 for GRAPH warnings**	

S21:Step21

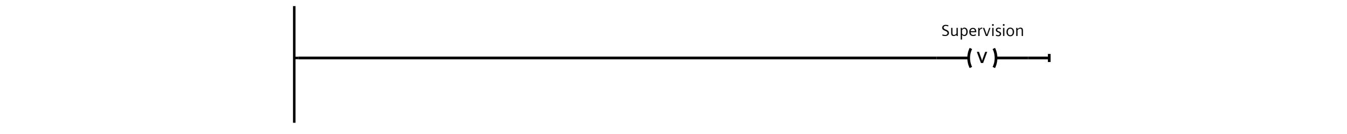
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

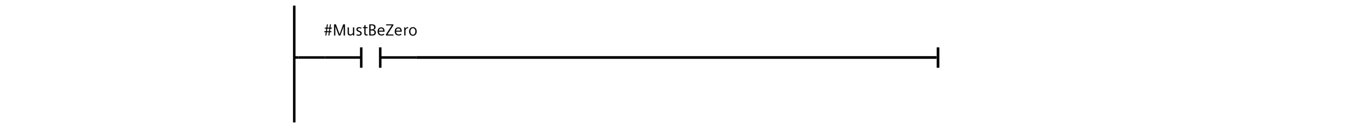
Supervision alarm	
Alarm text	



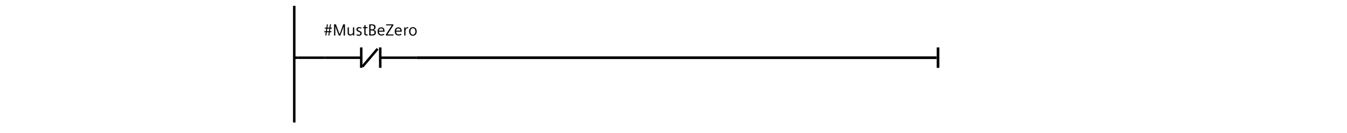
Actions:

Actions:			
Interlock	Event	Qualifier	Action

T1:Trans1



T2:Trans2



Doctoral AGV / Low Level Systems / Gertrude Main PLC [CPU 1512SP F-1 PN] / Program blocks / 01. Wheel Alignment

FB Steering Initialisation Sequencer [FB157]

FB Steering Initialisation Sequencer Properties					
General					
Name	FB Steering Initialisation Sequencer	**Number**	157	**Type**	FB
Language	GRAPH	**Numbering**	Manual	**Network language**	LAD
Block version	V6.0				
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Alarms

Enable alarms	True

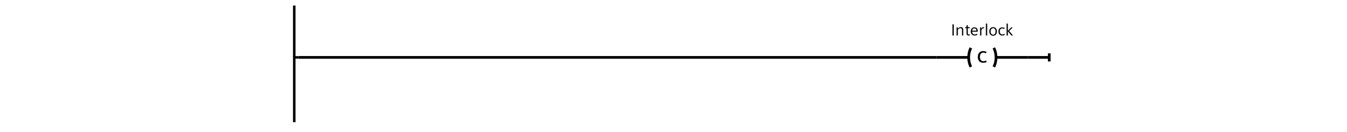
Category	Category enabler	Display class
Error		0
Warning		0
Info		0
Category 4		0
Category 5		0
Category 6		0
Category 7		0
Category 8		0

Category for interlocks	Error	**Subcategory 1 for interlocks**		**Subcategory 2 for interlocks**	
Category for supervisions	Error	**Subcategory 1 for supervisions**		**Subcategory 2 for supervisions**	
Category for GRAPH warnings	Warning	**Subcategory 1 for GRAPH warnings**		**Subcategory 2 for GRAPH warnings**	

S2:DeviationTest Zero'd Test1

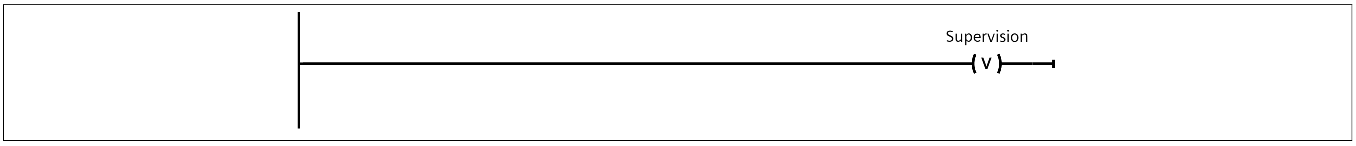
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

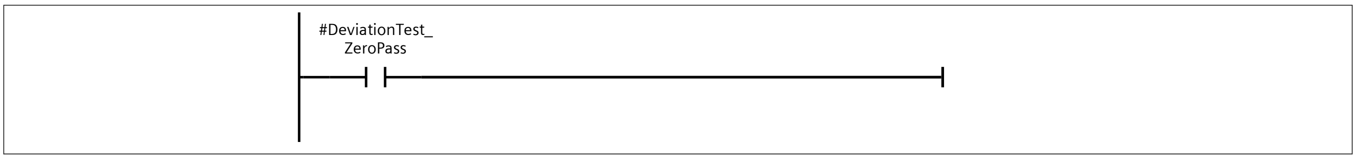
Supervision alarm	
Alarm text	



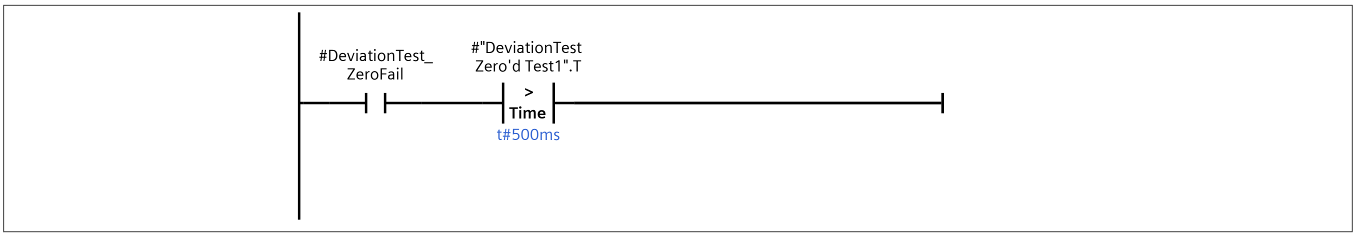
Actions:

Actions:			
Interlock	Event	Qualifier	Action
		N	#RunDeviationTest_Zero

T3:Trans3



T5:Trans5



Doctoral AGV / Low Level Systems / Gertrude Main PLC [CPU 1512SP F-1 PN] / Program blocks / 01. Wheel Alignment

FB Steering Initialisation Sequencer [FB157]

FB Steering Initialisation Sequencer Properties					
General					
Name	FB Steering Initialisation Sequencer	**Number**	157	**Type**	FB
Language	GRAPH	**Numbering**	Manual	**Network language**	LAD
Block version	V6.0				
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Alarms

Enable alarms	True

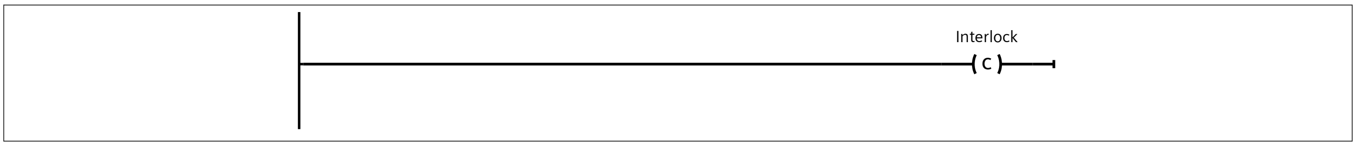
Category	Category enabler	Display class
Error		0
Warning		0
Info		0
Category 4		0
Category 5		0
Category 6		0
Category 7		0
Category 8		0

Category for interlocks	Error	**Subcategory 1 for interlocks**		**Subcategory 2 for interlocks**	
Category for supervisions	Error	**Subcategory 1 for supervisions**		**Subcategory 2 for supervisions**	
Category for GRAPH warnings	Warning	**Subcategory 1 for GRAPH warnings**		**Subcategory 2 for GRAPH warnings**	

S3:DeviationTest Aligned Test1

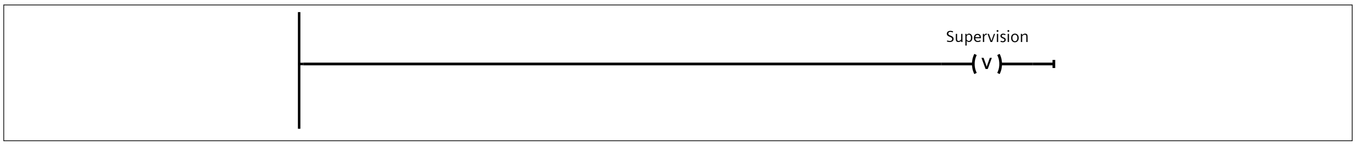
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

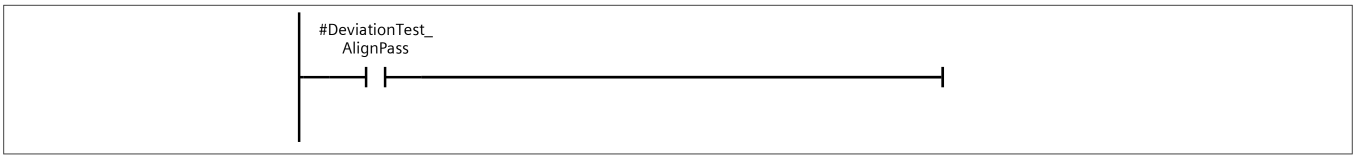
Supervision alarm	
Alarm text	



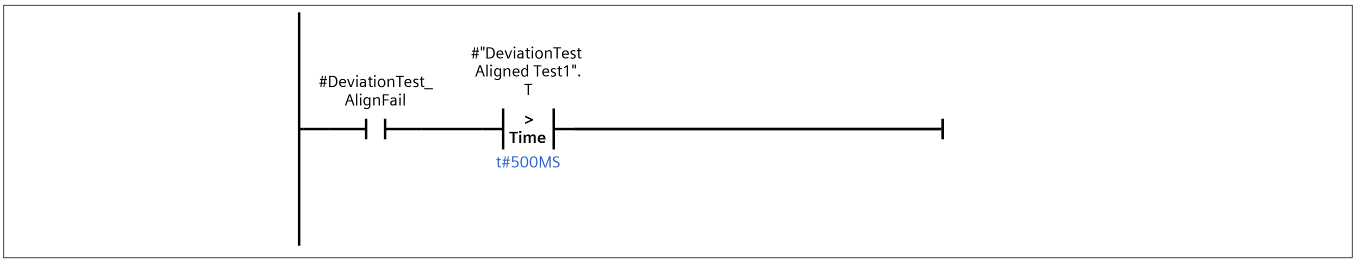
Actions:

Actions:			
Interlock	**Event**	**Qualifier**	**Action**
		N	#RunDeviationTest_Align

T4:Trans4



T23:Trans23



Doctoral AGV / Low Level Systems / Gertrude Main PLC [CPU 1512SP F-1 PN] / Program blocks / 01. Wheel Alignment

FB Steering Initialisation Sequencer [FB157]

FB Steering Initialisation Sequencer Properties					
General					
Name	FB Steering Initialisation Sequencer	**Number**	157	**Type**	FB
Language	GRAPH	**Numbering**	Manual	**Network language**	LAD
Block version	V6.0				
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Alarms

Enable alarms	True

Category	Category enabler	Display class
Error		0
Warning		0
Info		0
Category 4		0
Category 5		0
Category 6		0
Category 7		0
Category 8		0

Category for interlocks	Error	**Subcategory 1 for interlocks**		**Subcategory 2 for interlocks**	

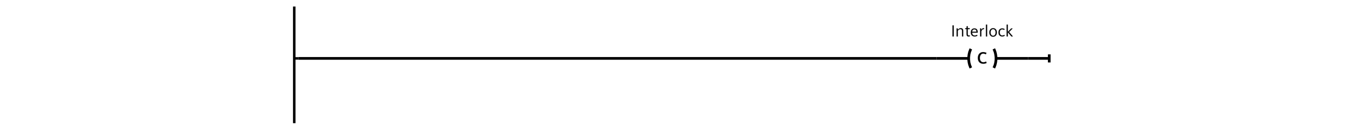
Category for supervisions	Error	**Subcategory 1 for supervisions**		**Subcategory 2 for supervisions**	

Category for GRAPH warnings	Warning	**Subcategory 1 for GRAPH warnings**		**Subcategory 2 for GRAPH warnings**	

S5:DeviationTest Zero'd Test2

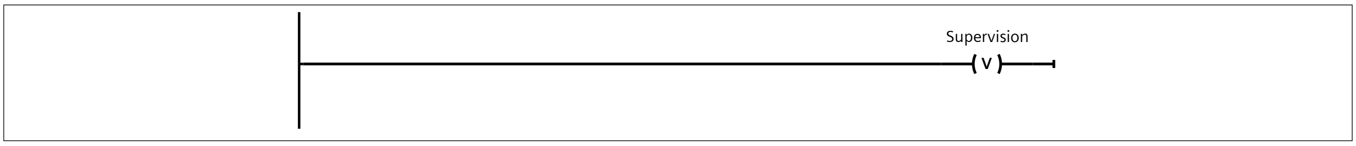
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

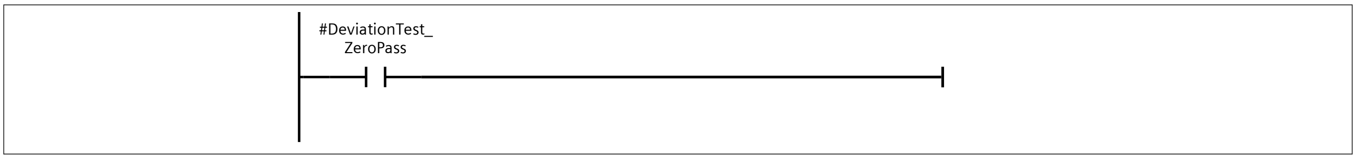
Supervision alarm	
Alarm text	



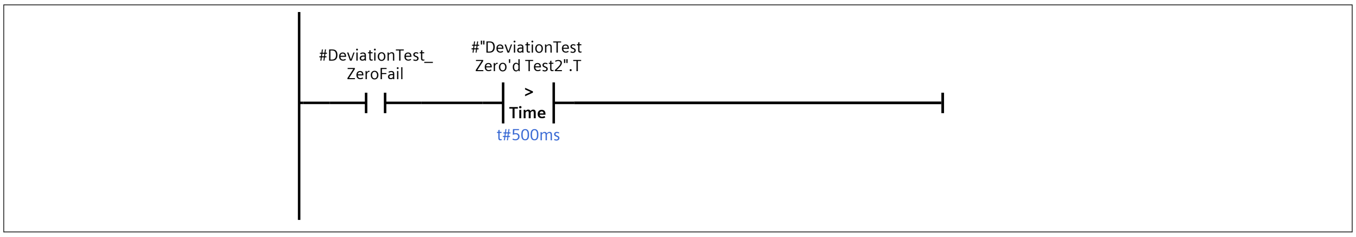
Actions:

Actions:			
Interlock	Event	Qualifier	Action
		R	#DeviationTest_ZeroFail
		N	#RunDeviationTest_Zero

T8:Trans8



T9:Trans9



Doctoral AGV / Low Level Systems / Gertrude Main PLC [CPU 1512SP F-1 PN] / Program blocks / 01. Wheel Alignment

FB Steering Initialisation Sequencer [FB157]

FB Steering Initialisation Sequencer Properties					
General					
Name	FB Steering Initialisation Sequencer	**Number**	157	**Type**	FB
Language	GRAPH	**Numbering**	Manual	**Network language**	LAD
Block version	V6.0				
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Alarms

Enable alarms	True

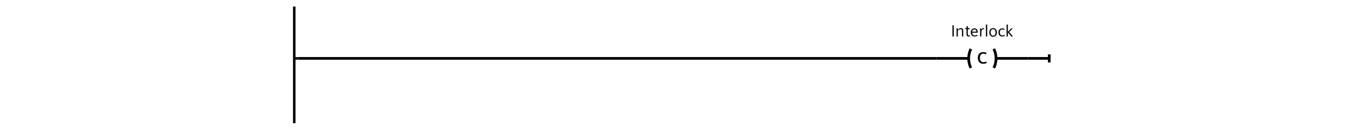
Category	Category enabler	Display class
Error		0
Warning		0
Info		0
Category 4		0
Category 5		0
Category 6		0
Category 7		0
Category 8		0

Category for interlocks	Error	**Subcategory 1 for interlocks**		**Subcategory 2 for interlocks**	
Category for supervisions	Error	**Subcategory 1 for supervisions**		**Subcategory 2 for supervisions**	
Category for GRAPH warnings	Warning	**Subcategory 1 for GRAPH warnings**		**Subcategory 2 for GRAPH warnings**	

S18:DeviationTest Aligned Test2

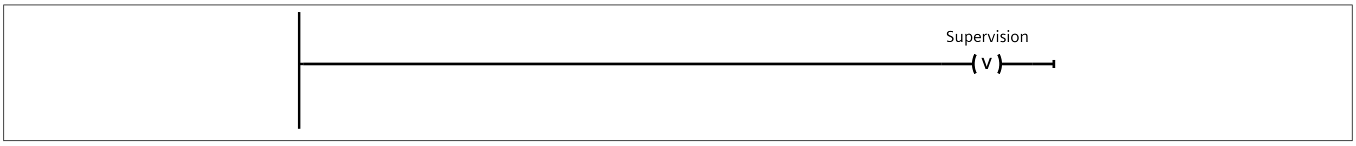
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

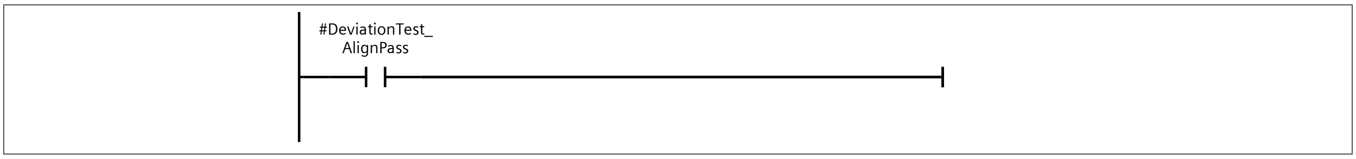
Supervision alarm	
Alarm text	



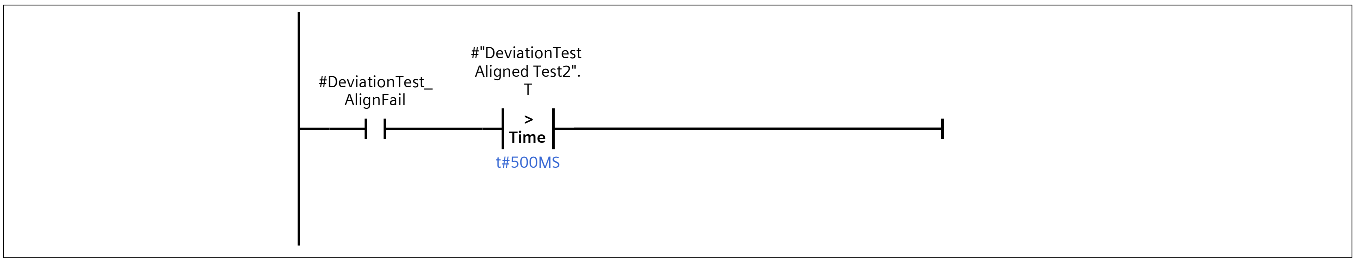
Actions:

Actions:			
Interlock	Event	Qualifier	Action
		R	#DeviationTest_AlignFail
		N	#RunDeviationTest_Align

T25:Trans25



T26:Trans26



Doctoral AGV / Low Level Systems / Gertrude Main PLC [CPU 1512SP F-1 PN] / Program blocks / 01. Wheel Alignment

FB Steering Initialisation Sequencer [FB157]

FB Steering Initialisation Sequencer Properties					
General					
Name	FB Steering Initialisation Sequencer	**Number**	157	**Type**	FB
Language	GRAPH	**Numbering**	Manual	**Network language**	LAD
Block version	V6.0				
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Alarms

Enable alarms	True

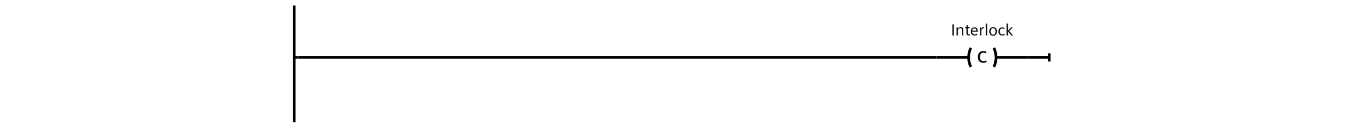
Category	Category enabler	Display class
Error		0
Warning		0
Info		0
Category 4		0
Category 5		0
Category 6		0
Category 7		0
Category 8		0

Category for interlocks	Error	**Subcategory 1 for interlocks**		**Subcategory 2 for interlocks**	
Category for supervisions	Error	**Subcategory 1 for supervisions**		**Subcategory 2 for supervisions**	
Category for GRAPH warnings	Warning	**Subcategory 1 for GRAPH warnings**		**Subcategory 2 for GRAPH warnings**	

S4:DeviationTest_Zero's Pass

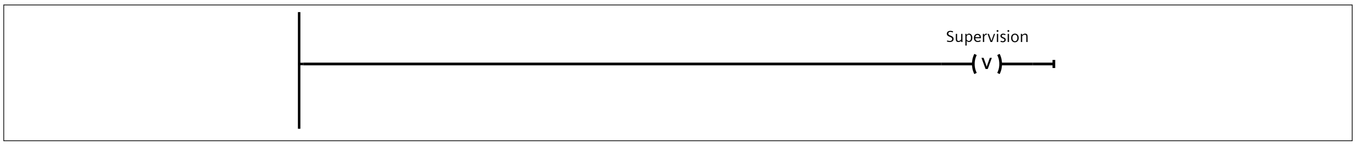
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

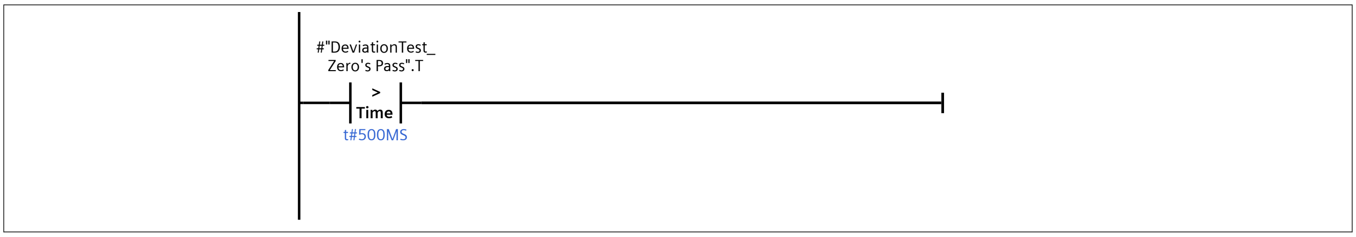
Supervision alarm	
Alarm text	



Actions:

Actions:			
Interlock	Event	Qualifier	Action
		R	#DeviationTest_ZeroPass
		R	#DeviationTest_AlignPass

T7:Trans7



Doctoral AGV / Low Level Systems / Gertrude Main PLC [CPU 1512SP F-1 PN] / Program blocks / 01. Wheel Alignment

FB Steering Initialisation Sequencer [FB157]

FB Steering Initialisation Sequencer Properties					
General					
Name	FB Steering Initialisation Sequencer	**Number**	157	**Type**	FB
Language	GRAPH	**Numbering**	Manual	**Network language**	LAD
Block version	V6.0				
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Alarms

Enable alarms	True

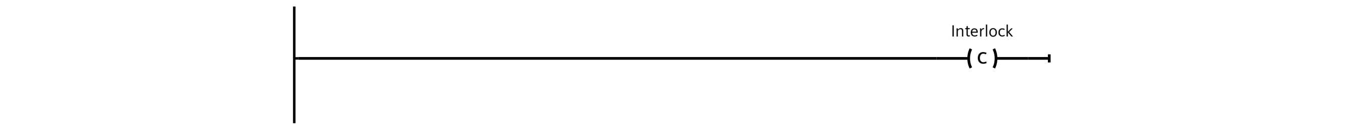
Category	Category enabler	Display class
Error		0
Warning		0
Info		0
Category 4		0
Category 5		0
Category 6		0
Category 7		0
Category 8		0

Category for interlocks	Error	**Subcategory 1 for interlocks**		**Subcategory 2 for interlocks**	
Category for supervisions	Error	**Subcategory 1 for supervisions**		**Subcategory 2 for supervisions**	
Category for GRAPH warnings	Warning	**Subcategory 1 for GRAPH warnings**		**Subcategory 2 for GRAPH warnings**	

S7:DeviationTest Zero'd Test3

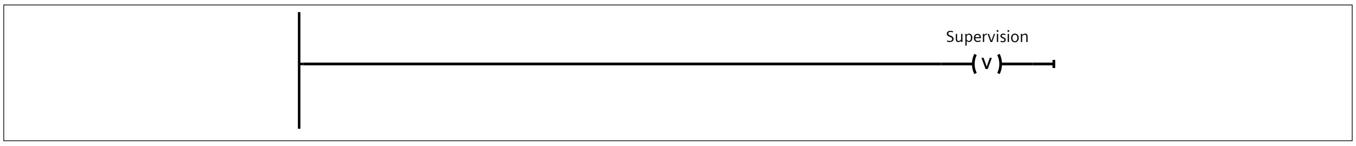
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

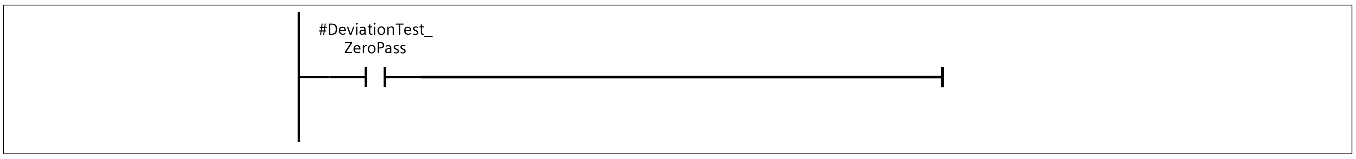
Supervision alarm	
Alarm text	



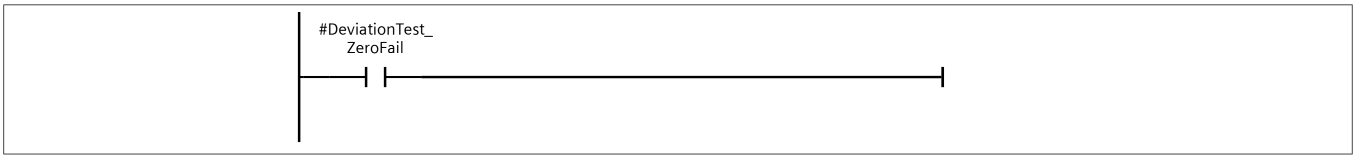
Actions:

Actions:			
Interlock	Event	Qualifier	Action
		R	#DeviationTest_ZeroFail
		N	#RunDeviationTest_Zero

T10:Trans10



T11:Trans11



Doctoral AGV / Low Level Systems / Gertrude Main PLC [CPU 1512SP F-1 PN] / Program blocks / 01. Wheel Alignment

FB Steering Initialisation Sequencer [FB157]

FB Steering Initialisation Sequencer Properties					
General					
Name	FB Steering Initialisation Sequencer	**Number**	157	**Type**	FB
Language	GRAPH	**Numbering**	Manual	**Network language**	LAD
Block version	V6.0				
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Alarms

Enable alarms	True

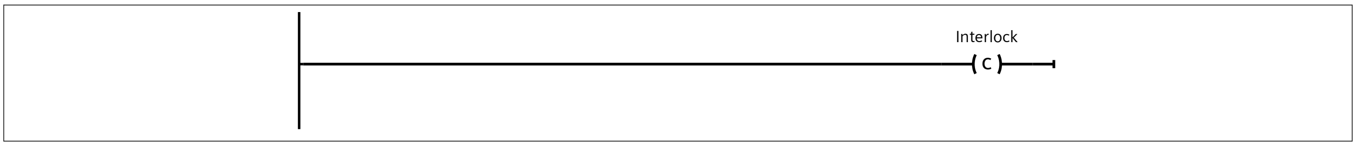
Category	Category enabler	Display class
Error		0
Warning		0
Info		0
Category 4		0
Category 5		0
Category 6		0
Category 7		0
Category 8		0

Category for interlocks	Error	**Subcategory 1 for interlocks**		**Subcategory 2 for interlocks**	
Category for supervisions	Error	**Subcategory 1 for supervisions**		**Subcategory 2 for supervisions**	
Category for GRAPH warnings	Warning	**Subcategory 1 for GRAPH warnings**		**Subcategory 2 for GRAPH warnings**	

S19:DeviationTest Aligned Test3

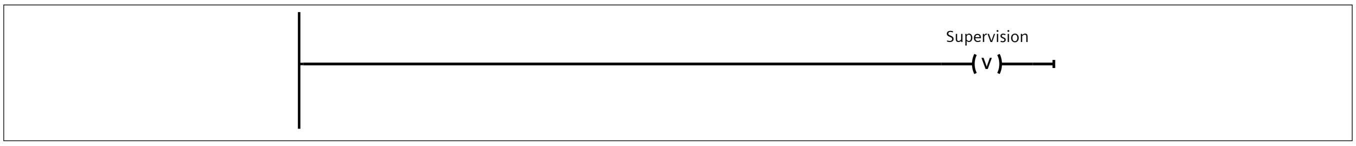
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

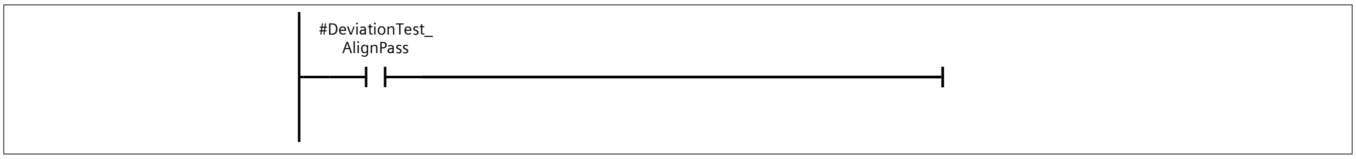
Supervision alarm	
Alarm text	



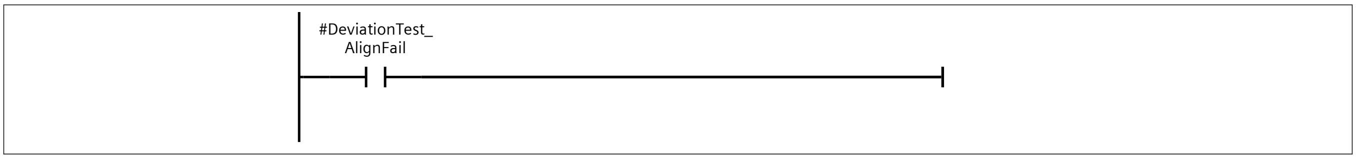
Actions:

Actions:			
Interlock	Event	Qualifier	Action
		R	#DeviationTest_AlignFail
		N	#RunDeviationTest_Align

T27:Trans27



T28:Trans28



Doctoral AGV / Low Level Systems / Gertrude Main PLC [CPU 1512SP F-1 PN] / Program blocks / 01. Wheel Alignment

FB Steering Initialisation Sequencer [FB157]

FB Steering Initialisation Sequencer Properties					
General					
Name	FB Steering Initialisation Sequencer	**Number**	157	**Type**	FB
Language	GRAPH	**Numbering**	Manual	**Network language**	LAD
Block version	V6.0				
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Alarms

Enable alarms	True

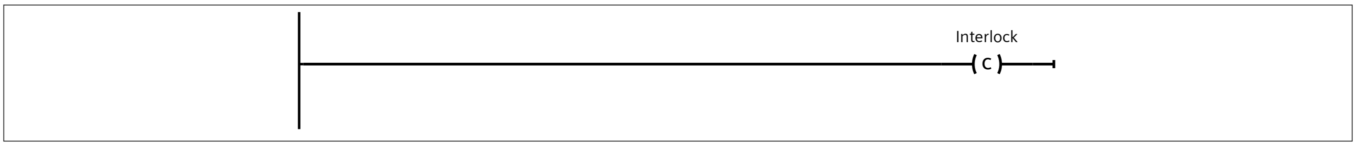
Category	Category enabler	Display class
Error		0
Warning		0
Info		0
Category 4		0
Category 5		0
Category 6		0
Category 7		0
Category 8		0

Category for interlocks	Error	**Subcategory 1 for interlocks**		**Subcategory 2 for interlocks**	
Category for supervisions	Error	**Subcategory 1 for supervisions**		**Subcategory 2 for supervisions**	
Category for GRAPH warnings	Warning	**Subcategory 1 for GRAPH warnings**		**Subcategory 2 for GRAPH warnings**	

S14:Zero Encoders

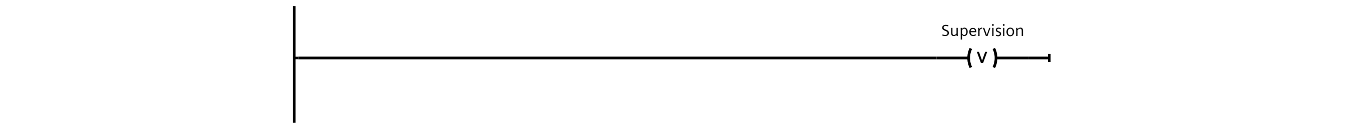
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

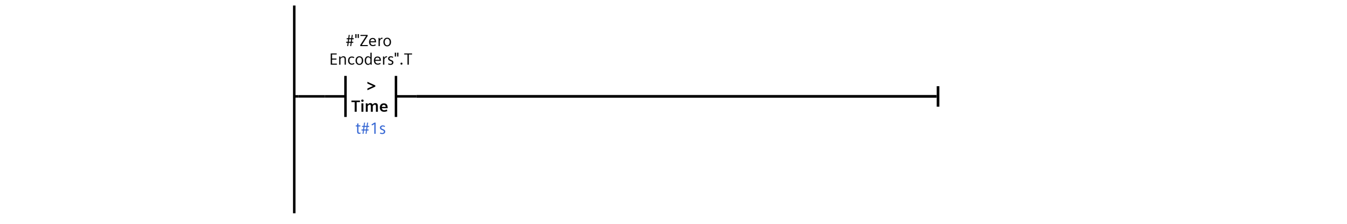
Supervision alarm	
Alarm text	



Actions:

Actions:			
Interlock	**Event**	**Qualifier**	**Action**
		N	#"ZeroCMMS-ST_Encoders_A"
		N	#"ZeroCMMS-ST_Encoders_B"

T20:Trans20



Doctoral AGV / Low Level Systems / Gertrude Main PLC [CPU 1512SP F-1 PN] / Program blocks / 01. Wheel Alignment

FB Steering Initialisation Sequencer [FB157]

FB Steering Initialisation Sequencer Properties					
General					
Name	FB Steering Initialisation Sequencer	**Number**	157	**Type**	FB
Language	GRAPH	**Numbering**	Manual	**Network language**	LAD
Block version	V6.0				
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Alarms

Enable alarms	True

Category	Category enabler	Display class
Error		0
Warning		0
Info		0
Category 4		0
Category 5		0
Category 6		0
Category 7		0
Category 8		0

Category for interlocks	Error	**Subcategory 1 for interlocks**		**Subcategory 2 for interlocks**	

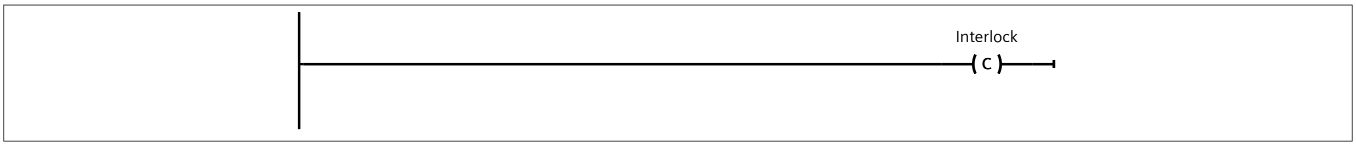
Category for supervisions	Error	**Subcategory 1 for supervisions**		**Subcategory 2 for supervisions**	

Category for GRAPH warnings	Warning	**Subcategory 1 for GRAPH warnings**		**Subcategory 2 for GRAPH warnings**	

S8:Initalise Alignment

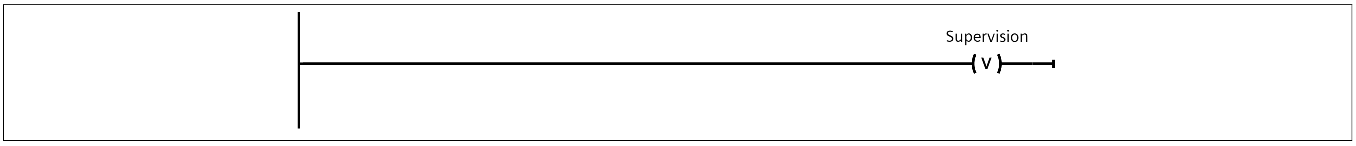
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

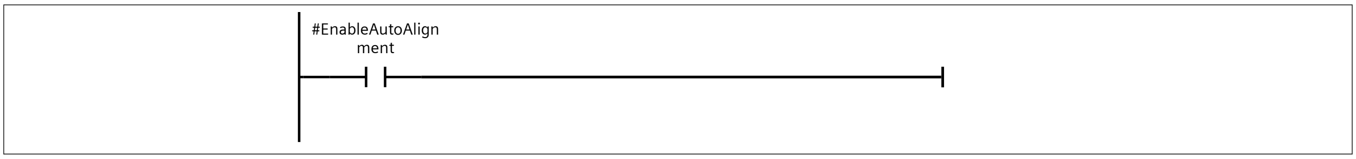
Supervision alarm	
Alarm text	



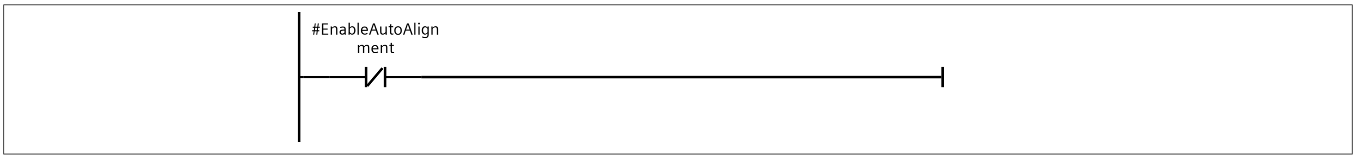
Actions:

Actions:			
Interlock	Event	Qualifier	Action
		R	#DeviationTest_ZeroFail
		R	#DeviationTest_AlignFail

T12:Trans12



T13:Trans13



Doctoral AGV / Low Level Systems / Gertrude Main PLC [CPU 1512SP F-1 PN] / Program blocks / 01. Wheel Alignment

FB Steering Initialisation Sequencer [FB157]

FB Steering Initialisation Sequencer Properties					
General					
Name	FB Steering Initialisation Sequencer	**Number**	157	**Type**	FB
Language	GRAPH	**Numbering**	Manual	**Network language**	LAD
Block version	V6.0				
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Alarms

Enable alarms	True

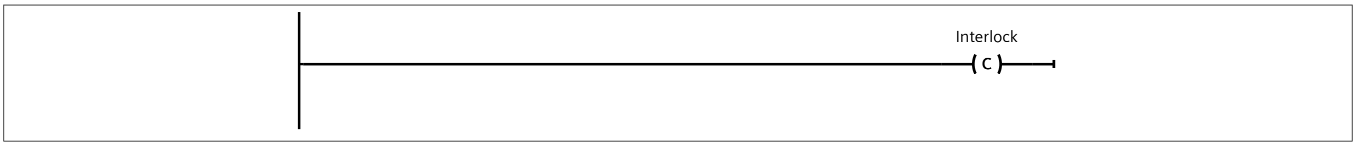
Category	Category enabler	Display class
Error		0
Warning		0
Info		0
Category 4		0
Category 5		0
Category 6		0
Category 7		0
Category 8		0

Category for interlocks	Error	**Subcategory 1 for interlocks**		**Subcategory 2 for interlocks**	
Category for supervisions	Error	**Subcategory 1 for supervisions**		**Subcategory 2 for supervisions**	
Category for GRAPH warnings	Warning	**Subcategory 1 for GRAPH warnings**		**Subcategory 2 for GRAPH warnings**	

S15:DeactivateSteppers

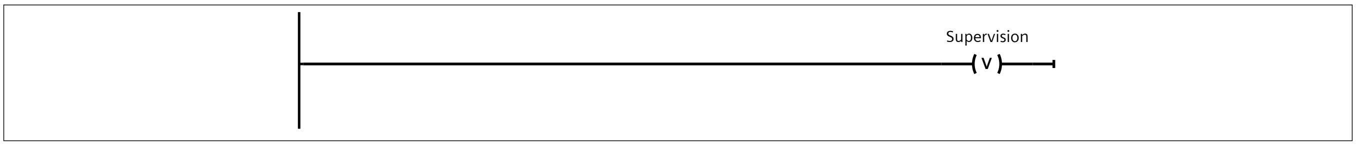
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

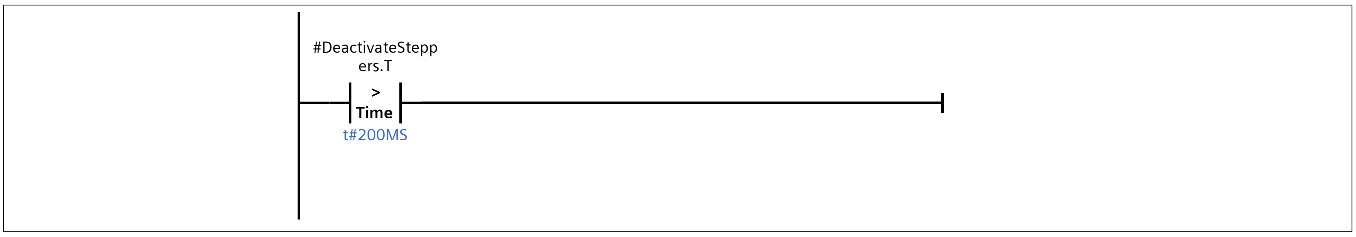
Supervision alarm	
Alarm text	



Actions:

Actions:			
Interlock	**Event**	**Qualifier**	**Action**
		R	#ActivateSteppers
		S	#WheelOrientationValid

T21:Trans21



Doctoral AGV / Low Level Systems / Gertrude Main PLC [CPU 1512SP F-1 PN] / Program blocks / 01. Wheel Alignment

FB Steering Initialisation Sequencer [FB157]

FB Steering Initialisation Sequencer Properties					
General					
Name	FB Steering Initialisation Sequencer	**Number**	157	**Type**	FB
Language	GRAPH	**Numbering**	Manual	**Network language**	LAD
Block version	V6.0				
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Alarms

Enable alarms	True

Category	Category enabler	Display class
Error		0
Warning		0
Info		0
Category 4		0
Category 5		0
Category 6		0
Category 7		0
Category 8		0

Category for interlocks	Error	**Subcategory 1 for interlocks**		**Subcategory 2 for interlocks**	

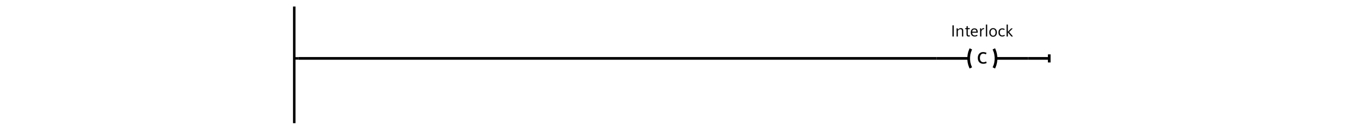
Category for supervisions	Error	**Subcategory 1 for supervisions**		**Subcategory 2 for supervisions**	

Category for GRAPH warnings	Warning	**Subcategory 1 for GRAPH warnings**		**Subcategory 2 for GRAPH warnings**	

S9:Auto Alignment

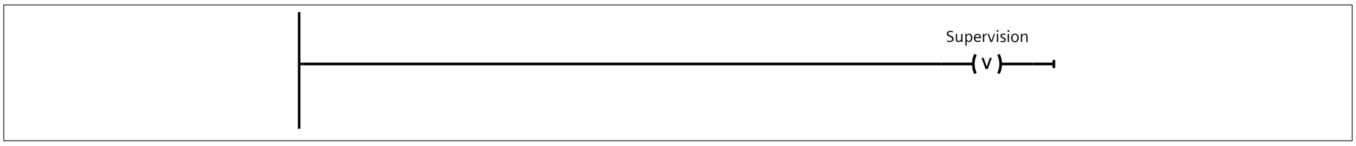
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

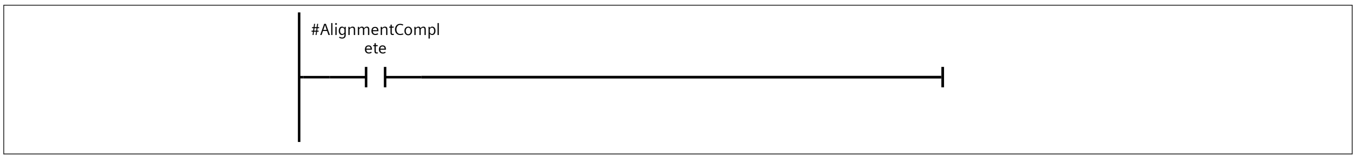
Supervision alarm	
Alarm text	



Actions:

Actions:			
Interlock	Event	Qualifier	Action
		N	#RunWheelAlignment

T22:Trans22



Doctoral AGV / Low Level Systems / Gertrude Main PLC [CPU 1512SP F-1 PN] / Program blocks / 01. Wheel Alignment

FB Steering Initialisation Sequencer [FB157]

FB Steering Initialisation Sequencer Properties					
General					
Name	FB Steering Initialisation Sequencer	**Number**	157	**Type**	FB
Language	GRAPH	**Numbering**	Manual	**Network language**	LAD
Block version	V6.0				
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Alarms

Enable alarms	True

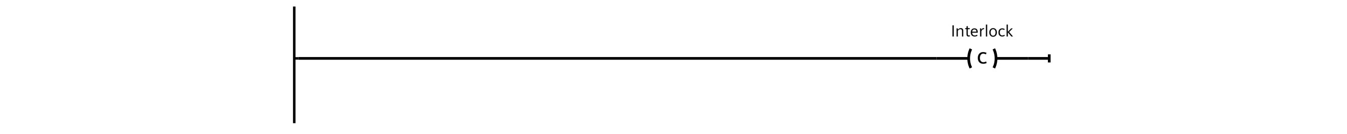
Category	Category enabler	Display class
Error		0
Warning		0
Info		0
Category 4		0
Category 5		0
Category 6		0
Category 7		0
Category 8		0

Category for interlocks	Error	**Subcategory 1 for interlocks**		**Subcategory 2 for interlocks**	
Category for supervisions	Error	**Subcategory 1 for supervisions**		**Subcategory 2 for supervisions**	
Category for GRAPH warnings	Warning	**Subcategory 1 for GRAPH warnings**		**Subcategory 2 for GRAPH warnings**	

S10:ManualAlignment

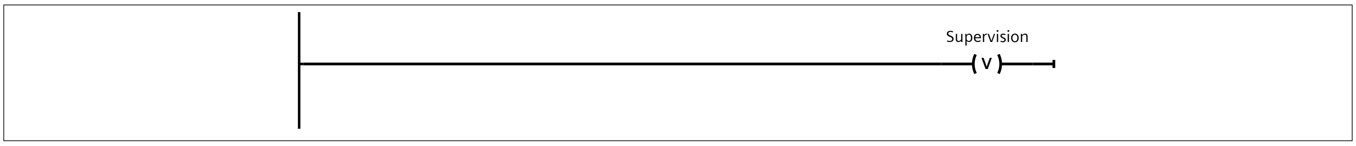
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

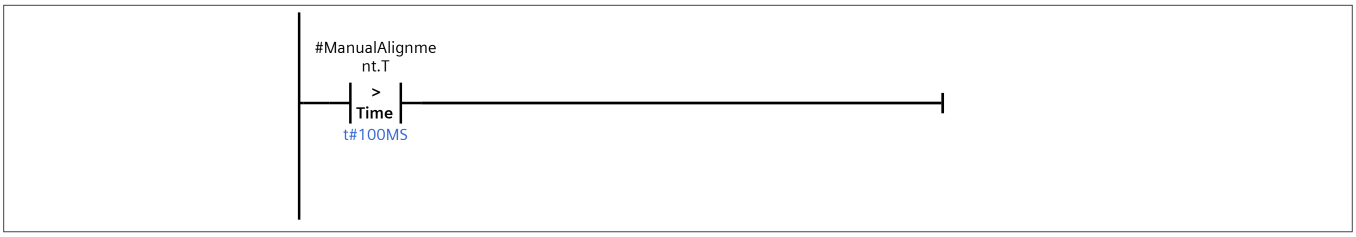
Supervision alarm	
Alarm text	



Actions:

Actions:			
Interlock	Event	Qualifier	Action
		S	#ManuallyAlignWheels

T15:Trans15



Doctoral AGV / Low Level Systems / Gertrude Main PLC [CPU 1512SP F-1 PN] / Program blocks / 01. Wheel Alignment

FB Steering Initialisation Sequencer [FB157]

FB Steering Initialisation Sequencer Properties					
General					
Name	FB Steering Initialisation Sequencer	**Number**	157	**Type**	FB
Language	GRAPH	**Numbering**	Manual	**Network language**	LAD
Block version	V6.0				
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Alarms

Enable alarms	True

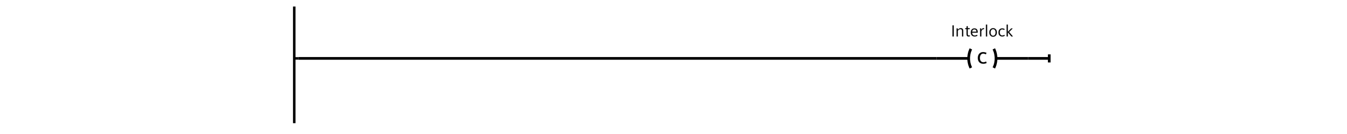
Category	Category enabler	Display class
Error		0
Warning		0
Info		0
Category 4		0
Category 5		0
Category 6		0
Category 7		0
Category 8		0

Category for interlocks	Error	**Subcategory 1 for interlocks**		**Subcategory 2 for interlocks**	
Category for supervisions	Error	**Subcategory 1 for supervisions**		**Subcategory 2 for supervisions**	
Category for GRAPH warnings	Warning	**Subcategory 1 for GRAPH warnings**		**Subcategory 2 for GRAPH warnings**	

S23:ZeroHMIDIals 1

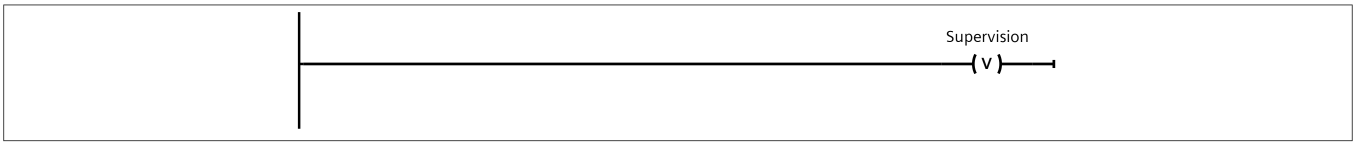
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

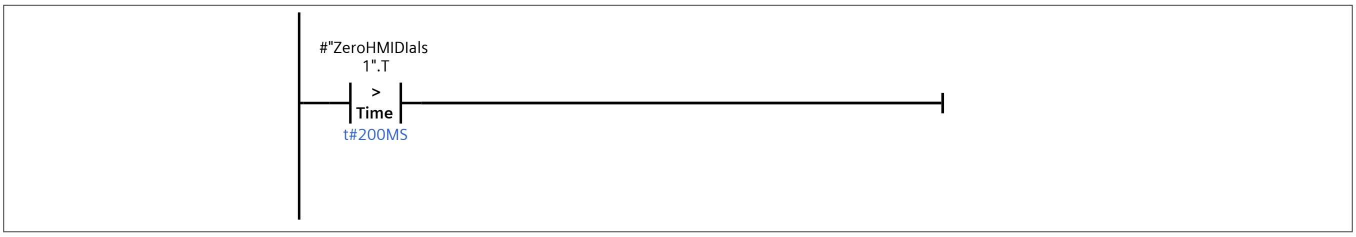
Supervision alarm	
Alarm text	



Actions:

Actions:			
Interlock	Event	Qualifier	Action
		N	#ZeroHMIDials

T32:Trans32



Doctoral AGV / Low Level Systems / Gertrude Main PLC [CPU 1512SP F-1 PN] / Program blocks / 01. Wheel Alignment

FB Steering Initialisation Sequencer [FB157]

FB Steering Initialisation Sequencer Properties					
General					
Name	FB Steering Initialisation Sequencer	**Number**	157	**Type**	FB
Language	GRAPH	**Numbering**	Manual	**Network language**	LAD
Block version	V6.0				
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Alarms

Enable alarms	True

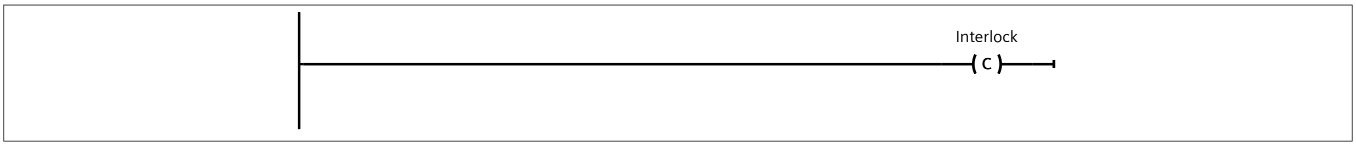
Category	Category enabler	Display class
Error		0
Warning		0
Info		0
Category 4		0
Category 5		0
Category 6		0
Category 7		0
Category 8		0

Category for interlocks	Error	**Subcategory 1 for interlocks**		**Subcategory 2 for interlocks**	
Category for supervisions	Error	**Subcategory 1 for supervisions**		**Subcategory 2 for supervisions**	
Category for GRAPH warnings	Warning	**Subcategory 1 for GRAPH warnings**		**Subcategory 2 for GRAPH warnings**	

S16:AlignmentDone

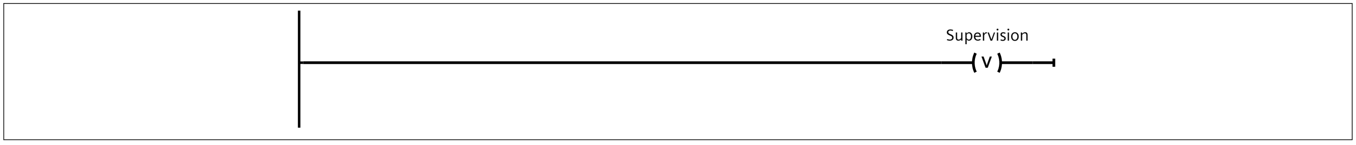
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

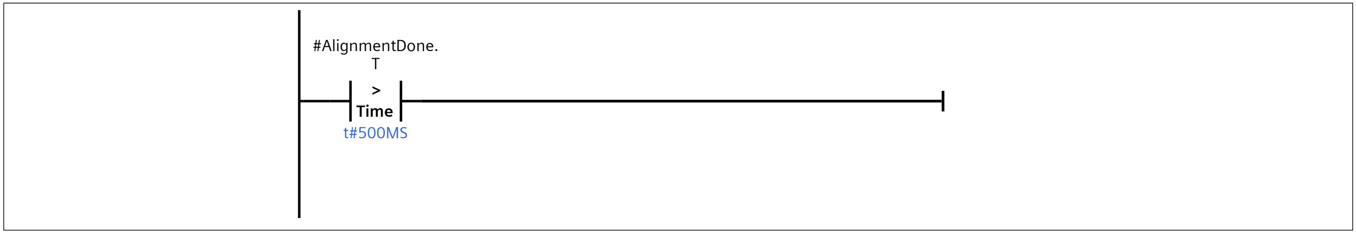
Supervision alarm	
Alarm text	



Actions:

Actions:			
Interlock	Event	Qualifier	Action

T16:Trans16



Doctoral AGV / Low Level Systems / Gertrude Main PLC [CPU 1512SP F-1 PN] / Program blocks / 01. Wheel Alignment

FB Steering Initialisation Sequencer [FB157]

FB Steering Initialisation Sequencer Properties					
General					
Name	FB Steering Initialisation Sequencer	**Number**	157	**Type**	FB
Language	GRAPH	**Numbering**	Manual	**Network language**	LAD
Block version	V6.0				
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Alarms

Enable alarms	True

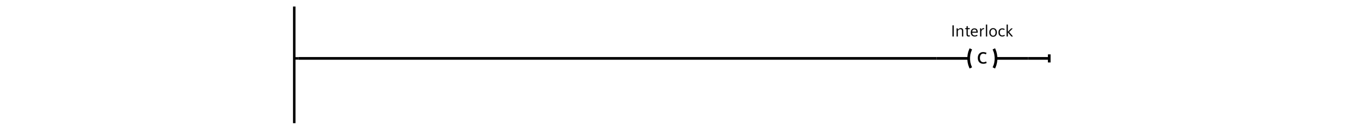
Category	Category enabler	Display class
Error		0
Warning		0
Info		0
Category 4		0
Category 5		0
Category 6		0
Category 7		0
Category 8		0

Category for interlocks	Error	**Subcategory 1 for interlocks**		**Subcategory 2 for interlocks**	
Category for supervisions	Error	**Subcategory 1 for supervisions**		**Subcategory 2 for supervisions**	
Category for GRAPH warnings	Warning	**Subcategory 1 for GRAPH warnings**		**Subcategory 2 for GRAPH warnings**	

S11:ZeroEncoders2

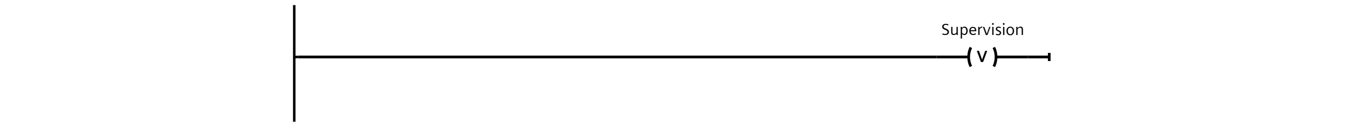
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

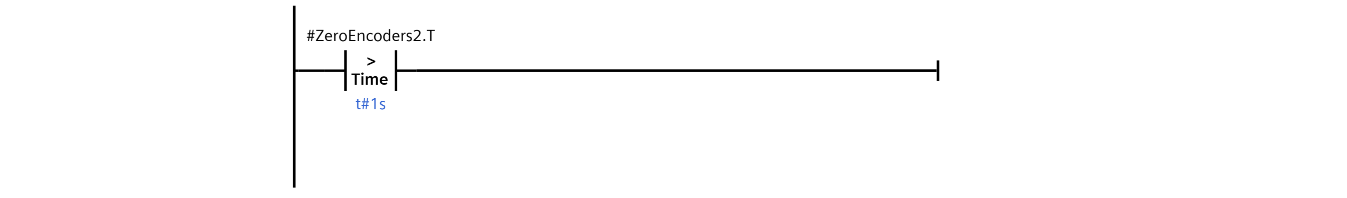
Supervision alarm	
Alarm text	



Actions:

Actions:			
Interlock	**Event**	**Qualifier**	**Action**
		N	#"ZeroCMMS-ST_Encoders_A"
		N	#"ZeroCMMS-ST_Encoders_B"

T19:Trans19



Doctoral AGV / Low Level Systems / Gertrude Main PLC [CPU 1512SP F-1 PN] / Program blocks / 01. Wheel Alignment

FB Steering Initialisation Sequencer [FB157]

FB Steering Initialisation Sequencer Properties					
General					
Name	FB Steering Initialisation Sequencer	**Number**	157	**Type**	FB
Language	GRAPH	**Numbering**	Manual	**Network language**	LAD
Block version	V6.0				
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Alarms

Enable alarms	True

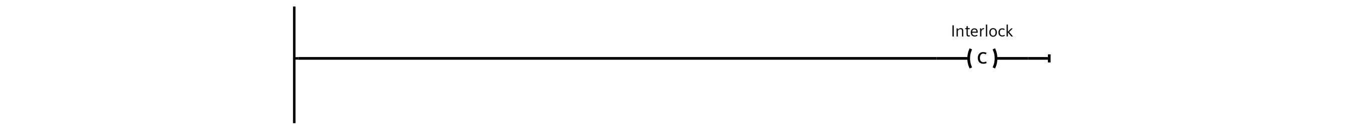
Category	Category enabler	Display class
Error		0
Warning		0
Info		0
Category 4		0
Category 5		0
Category 6		0
Category 7		0
Category 8		0

Category for interlocks	Error	**Subcategory 1 for interlocks**		**Subcategory 2 for interlocks**	
Category for supervisions	Error	**Subcategory 1 for supervisions**		**Subcategory 2 for supervisions**	
Category for GRAPH warnings	Warning	**Subcategory 1 for GRAPH warnings**		**Subcategory 2 for GRAPH warnings**	

S13:HomingDone

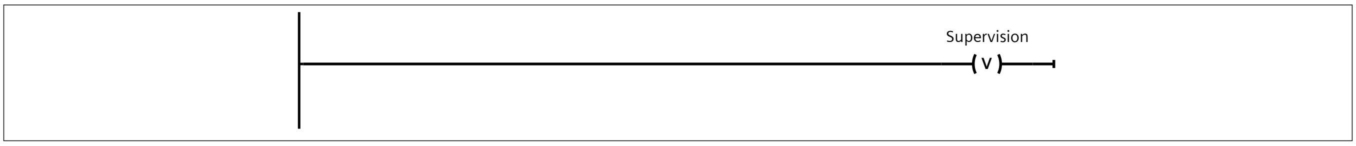
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

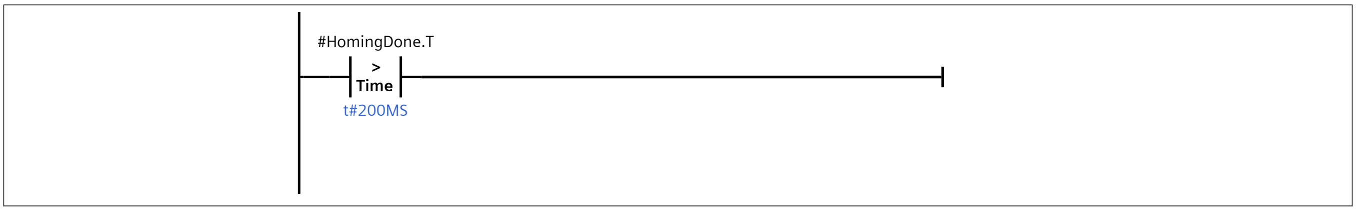
Supervision alarm	
Alarm text	



Actions:

Actions:			
Interlock	Event	Qualifier	Action
		S	#WheelOrientationValid
		R	#ActivateSteppers

T17:Trans17



Doctoral AGV / Low Level Systems / Gertrude Main PLC [CPU 1512SP F-1 PN] / Program blocks / 01. Wheel Alignment

FB Steering Initialisation Sequencer [FB157]

FB Steering Initialisation Sequencer Properties					
General					
Name	FB Steering Initialisation Sequencer	**Number**	157	**Type**	FB
Language	GRAPH	**Numbering**	Manual	**Network language**	LAD
Block version	V6.0				
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Alarms

Enable alarms	True

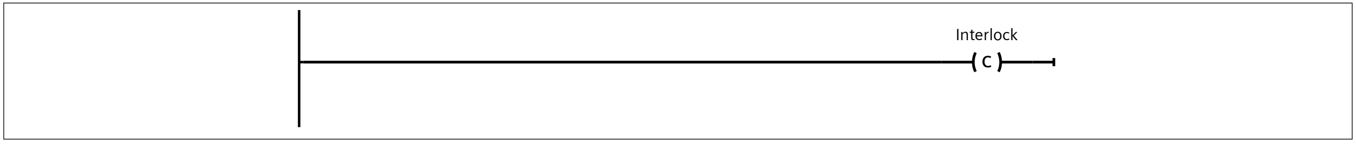
Category	Category enabler	Display class
Error		0
Warning		0
Info		0
Category 4		0
Category 5		0
Category 6		0
Category 7		0
Category 8		0

Category for interlocks	Error	**Subcategory 1 for interlocks**		**Subcategory 2 for interlocks**	
Category for supervisions	Error	**Subcategory 1 for supervisions**		**Subcategory 2 for supervisions**	
Category for GRAPH warnings	Warning	**Subcategory 1 for GRAPH warnings**		**Subcategory 2 for GRAPH warnings**	

S22:Zero HMI Dials 2

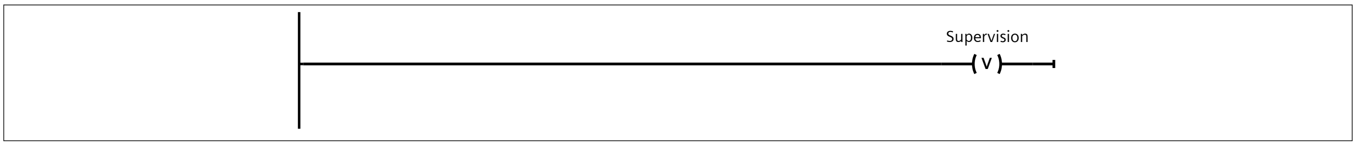
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

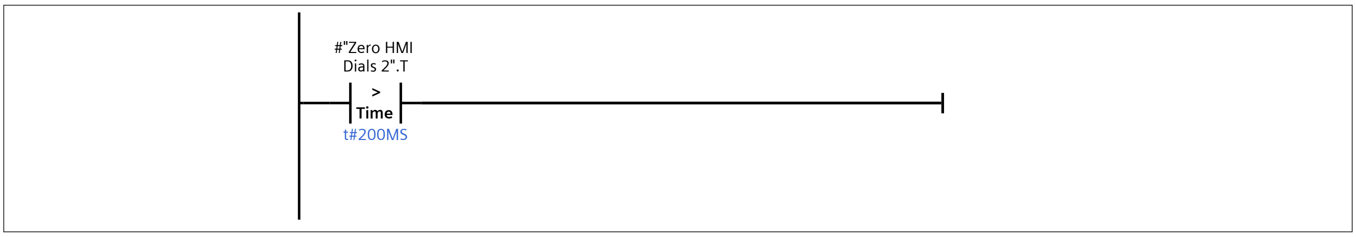
Supervision alarm	
Alarm text	



Actions:

Actions:			
Interlock	**Event**	**Qualifier**	**Action**
		N	#ZeroHMIDials

T31:Trans31



02. Modes

FB Power & Mode Control [FB6]

FB Power & Mode Control Properties					
General					
Name	FB Power & Mode Control	**Number**	6	**Type**	FB
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
▼ Input			
SW ON/OFF	Bool	false	Non-retain
SW Mode	Bool	false	Non-retain
HMI ON/OFF	Bool	false	Non-retain
HMI TestAGV ON	Bool	false	Non-retain
HMI TestAGV Abort	Bool	false	Non-retain
HMI AutoMode	Bool	false	Non-retain
HMI ManualMode	Bool	false	Non-retain
HMI ComisngMode	Bool	false	Non-retain
HMI MoveUnits	Bool	false	Non-retain
HMI MoveAGV	Bool	false	Non-retain
StartTestMode	Bool	false	Non-retain
EndTestMode	Bool	false	Non-retain
ActivateDefaultState	Bool	false	Non-retain
STO State	Bool	false	Non-retain
Homing	Bool	false	Non-retain
Boot Latch	Bool	false	Non-retain
▼ Output			
AGV PWR State	Bool	false	Non-retain
Automatic Mode	Bool	false	Non-retain
Manual Mode	Bool	false	Non-retain
Commisioning Mode	Bool	false	Non-retain
HMI Move Units	Bool	false	Non-retain
HMI Move AGV	Bool	false	Non-retain
Homing Mode	Bool	false	Non-retain
Testing Mode	Bool	false	Non-retain
InOut			
▼ Static			
PowerLatch	Bool	false	Non-retain
R_TRIG_ON/OFF_SW	R_TRIG		
SW ON/OFF Pulse	Bool	false	Non-retain
ModeNo	UInt	0	Non-retain
SW Mode Pulse	Bool	false	Non-retain
R_TRIG_Mode_SW	R_TRIG		
R_TRIG_ON/OFF_Latch	R_TRIG		

Name	Data type	Default value	Retain
PWR Latch Pulse	Bool	false	Non-retain
HMI_PWR_Pulse	Bool	false	Non-retain
R_TRIG_HMI_On/Off_Pulse	R_TRIG		
StoreLastState	UInt	0	Non-retain
ManualModeRLO	Bool	false	Non-retain
CommisiongModeRLO	Bool	false	Non-retain
AutoModeRLO	Bool	false	Non-retain
SWModeRLO	Bool	false	Non-retain
TestingModeLatch	Bool	false	Non-retain
P_Edge	Array[0..2] of Bool		Non-retain
P_StartTestMode	R_TRIG		
StartTestMode Pulse	Bool	false	Non-retain
P_EndTestMode	R_TRIG		
EndTestMode Pulse	Bool	false	Non-retain
N_HomingMode	F_TRIG		
Homing Pulse	Bool	false	Non-retain
P_ActivateDefaultState	R_TRIG		
ActivateDefaultState Pulse	Bool	false	Non-retain
P_TestAbort	R_TRIG		
TestAbort_Pulse	Bool	false	Non-retain
Temp			
Constant			

Network 1: ON / OFF State of AGV via Pendant SW

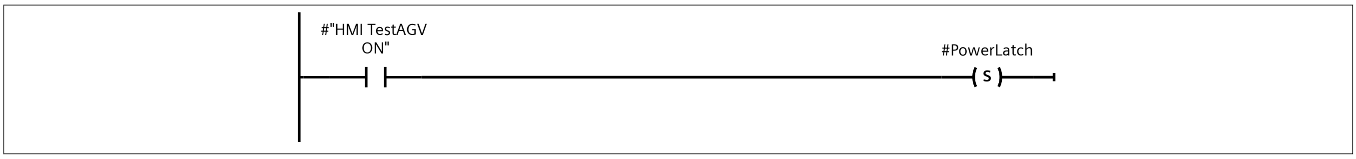
```
// Using a physical switch toggle AGV ON/OFF
//
    //Generate a pulse on rising edge of switch (incase it is left in on position)
    //
    #"R_TRIG_ON/OFF_SW"(CLK:=#"SW ON/OFF",
                     Q=>#"SW ON/OFF Pulse");

    //Main power latch
    //
    IF (#"SW ON/OFF Pulse" = TRUE) AND (#PowerLatch = TRUE) THEN
        // Rising edge at switch and PowerLatch is true, this indicates that the system is ON
        // Thus the system must be turned OFF
        //
        #PowerLatch := FALSE;

    ELSIF (#"SW ON/OFF Pulse" = TRUE) AND (#PowerLatch = FALSE) THEN
        // Rising edge at switch and PowerLatch is false, this indicates that the system is OFF
        // Thus the system must be turned ON
        //
        IF (#"STO State" = TRUE) AND (#Homing = FALSE) AND (#"Boot Latch" = FALSE) THEN
```

```
            // STO is not triggered, and the system is not homing nor is it booting
            // System can be activated
            //
            #PowerLatch := TRUE;
            
        ELSE
            // System is not ready to be activated, set system power state to OFF
            //
            #PowerLatch := FALSE;
            
        END_IF;
        
    ELSE
        // ON/OFF switch not triggered => do nothing
        ;
    END_IF;
```

Network 2: ON/OFF state via HMI

```
//The HMI will give an on/off signal this must be translated to a pulse for the PowerLatch
    //
    
    //Generate a positive edge at going high of signal
    //
    #"R_TRIG_HMI_On/Off_Pulse"(CLK:=#"HMI ON/OFF",
                               Q=>#HMI_PWR_Pulse);
    
    //Control main power latch
    //
    
    IF (#HMI_PWR_Pulse = TRUE) AND (#PowerLatch = TRUE) THEN
        // Rising edge at HMI switch and PowerLatch is true, this indicates that the system is ON
        // Thus the system must be turned OFF
        
        #PowerLatch := FALSE;
        
    ELSIF (#HMI_PWR_Pulse = TRUE) AND (#PowerLatch = FALSE) THEN
        // Rising edge at HMI switch and PowerLatch is false, this indicates that the system is OFF
        // Thus the system must be turned ON
        
        IF (#"STO State" = TRUE) AND (#Homing = FALSE) AND (#"Boot Latch" = FALSE) THEN
            // STO is not triggered, and the system is not homing nor is it booting
            // System can be activated
            //
            #PowerLatch := TRUE;
            
        ELSE
            //System is not ready to be activated
            //
            #PowerLatch := FALSE;
        END_IF;

    ELSE
        //do nothing
        ;
    END_IF;

```

Network 3: ON/OFF via testing dialog screen



Network 4: Cancel Latch automatically if STO, homing or boot occurs and system was ON

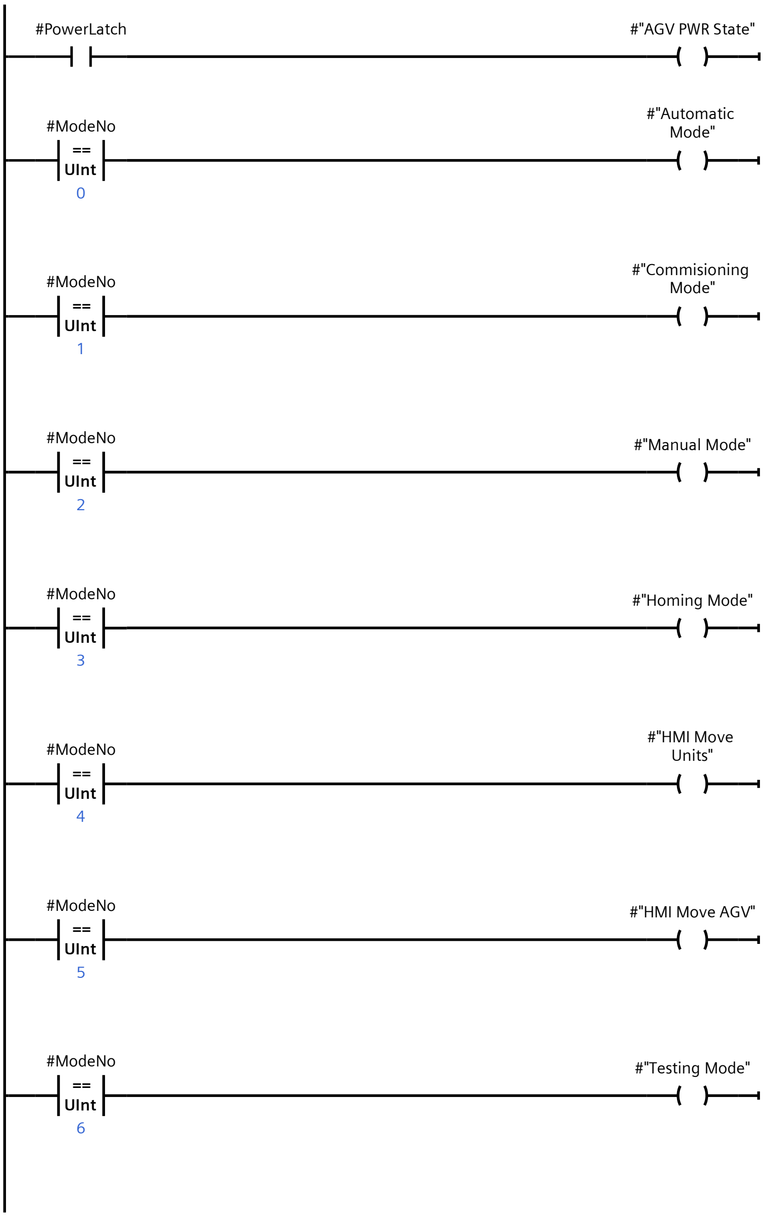
```
//An STO error has occured the system must move to OFF state
    //
    IF #"STO State" = FALSE THEN
        // When STO = FALSE, system has enterd a safety condition
        //
        #PowerLatch := FALSE;

    END_IF;

//The system is attempting to home the steering set the AGV state to OFF
    //
    IF #Homing = TRUE THEN
        // System is currently trying to home, this is done in the OFF state as the homing subroutine
        // directly interfaces with the drives and doesnt activated them via this block
        //
        #PowerLatch := FALSE;

    END_IF;

//The system is booting thus the AGV must be OFF
    //
    IF #"Boot Latch" = TRUE THEN
        // AGV is still in boot process, thus should be turned off for safety
        // This segment of code should never activate, only here if memory buffer not properly cleard during reboot process
        // and the ON state from the last power cycle is retained
        //
        #PowerLatch := FALSE;

    END_IF;

```

Network 5: Mode Selection via Pendant SW

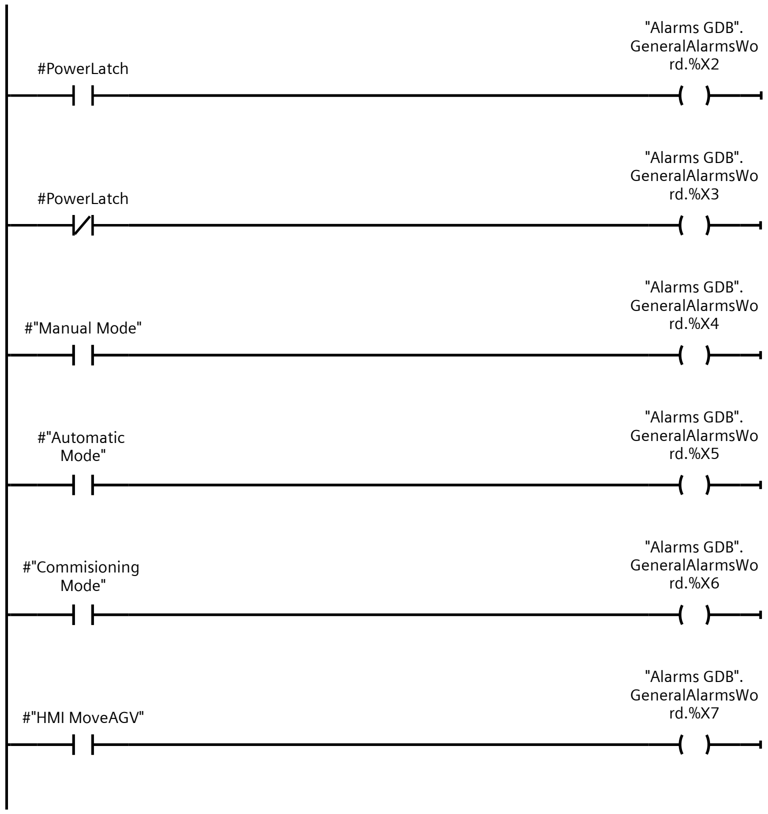
Mode selection done via one button on the pendant

```
//Pendant can be used to toggle between automatic mode, manual mode and commisioning mode
//Homing mode is a higher level mode that activates at boot or when a homing command is given via the HMI
//
    //Generate positive edge for mode selection SW
    #R_TRIG_Mode_SW(CLK := #"SW Mode",
                    Q => #"SW Mode Pulse");
    
    //Generate positive edge for start test mode
    ///#P_StartTestMode(CLK := #StartTestMode,
                     //Q => #"StartTestMode Pulse");
    
    //Generate positive edge for stop test mode
    ///#P_EndTestMode(CLK := #EndTestMode,
                   //Q => #"EndTestMode Pulse");
    
    //Generate negative edge when exiting homing mode
    #N_HomingMode(CLK := #Homing,
                  Q => #"Homing Pulse");
    
    //Generate positive edge for activate default state
    #P_ActivateDefaultState(CLK := #ActivateDefaultState,
                            Q => #"ActivateDefaultState Pulse");
    
    
    IF (#Homing = FALSE) AND (#TestingModeLatch = FALSE) THEN
        //System is not attempting to home
        //
        
        IF #"ActivateDefaultState Pulse" = TRUE THEN
            //External call for store last state
            #ModeNo := #StoreLastState;
            #PowerLatch := FALSE;
        END_IF;
        
        IF #"Homing Pulse" = TRUE THEN
            //On negative edge of leaving homeing mode
            #ModeNo := #StoreLastState;
            #PowerLatch := FALSE;
        END_IF;
        
        //CONTROL FROM PENDANT
        //
*********************************************************************************************
        
            IF #"SW Mode Pulse" = TRUE THEN
                // Positive edge of mode selection switch on pendant detected

                #PowerLatch := FALSE;

                IF (#ModeNo >= 3) THEN // 3 = homing system mode
                    #ModeNo := #StoreLastState;
                END_IF;

                #ModeNo := #ModeNo + 1; //Incriment mode number by one

                IF (#ModeNo > 2) THEN
                    //Mode has reached a value higher than the selectable values, thus it needs to be rolled 
                    //over back to mode 1
                    #ModeNo := 0;
                END_IF;

                #StoreLastState := #ModeNo;

            END_IF;

        //CONTROL FROM HMI
       //***********************************************************************************************

            IF #"HMI AutoMode" = TRUE THEN
                //Automatic mode PB pressed
                //
                #ModeNo := 0;
                #StoreLastState := 0;
                #PowerLatch := FALSE;
            END_IF;

            IF #"HMI ComisngMode" = TRUE THEN
                //Commisioning mode PB pressed
                //
                #ModeNo := 1;
                #StoreLastState := 1;
                #PowerLatch := FALSE;
            END_IF;

            IF #"HMI ManualMode" = TRUE THEN
                //Manual mode PB pressed
                //
                #ModeNo := 2;
                #StoreLastState := 2;
                #PowerLatch := FALSE;
            END_IF;

            //MODE 3 SELECTED BY OUTERLOOP

            IF #"HMI MoveUnits" = TRUE THEN
                //HMI commisioining mode selected (JOG INDIVIDUAL MOTORS)
                //
                #ModeNo := 4;
            END_IF;

            IF #"HMI MoveAGV" = TRUE THEN
                //HMI jog mode selected (ENTIRE AGV JOG)
                //
                #ModeNo := 5;
            END_IF;

            IF #StartTestMode = TRUE THEN
                //System wants to run a test using streamed buffer data
                //
                #ModeNo := 6;
                #TestingModeLatch := TRUE;
            END_IF;

    ELSIF (#Homing = TRUE) AND (#TestingModeLatch = FALSE) THEN
        //system is homing, set mode to 3 (cannot enter homing mode if test is being run)
        #ModeNo := 3;

    ELSE

        IF (#TestingModeLatch = TRUE) AND (#EndTestMode = TRUE) THEN
            //System is currently in test mode
            //
            #TestingModeLatch := FALSE;
            #PowerLatch := FALSE;
            #ModeNo := #StoreLastState;
        END_IF;

        IF (#"HMI TestAGV Abort"= TRUE) AND (#TestingModeLatch = TRUE) THEN
            //Return to last selected state if test aborted
            #TestingModeLatch := FALSE;
            #ModeNo := #StoreLastState;
            #PowerLatch := FALSE;
        END_IF;

    END_IF;



```

Network 6: Set Outputs



Network 7: Alarms


#PowerLatch
"Alarms GDB".
GeneralAlarmsWo
rd.%X2
#PowerLatch
"Alarms GDB".
GeneralAlarmsWo
rd.%X3
#"Manual Mode"
"Alarms GDB".
GeneralAlarmsWo
rd.%X4
#"Automatic
Mode"
"Alarms GDB".
GeneralAlarmsWo
rd.%X5
#"Commisioning
Mode"
"Alarms GDB".
GeneralAlarmsWo
rd.%X6
#"HMI MoveAGV"
"Alarms GDB".
GeneralAlarmsWo
rd.%X7

02. Modes

Manual Yaw Rate Differentiation iDB [DB7]

Manual Yaw Rate Differentiation iDB Properties					
General					
Name	Manual Yaw Rate Differentiation iDB	**Number**	7	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**	Siemens_Digital_Industry	**Comment**	
Family	LGF	**Version**	0.1	**User-defined ID**	LGF_DifferenceQuotientFB

Name	Data type	Start value	Retain
▼ Input			
enable	Bool	false	False
insert	Bool	false	False
value	LReal	0.0	False
deltaT	LReal	0.0	False
▼ Output			
derivatedValue	LReal	0.0	False
error	Bool	false	False
status	Word	16#0	False
InOut			
▼ Static			
statValues	Array[0..4] of LReal		False
statCount	Int	0	False
statDerivatedValue	LReal	0.0	False
statStatus	Word	16#0	False
statEnableOld	Bool	false	False
statInsertOld	Bool	false	False

02. Modes / Instance DBs

Power & Mode Control_iDB [DB6]

Power & Mode Control_iDB Properties					
General					
Name	Power & Mode Control_iDB	**Number**	6	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
SW ON/OFF	Bool	false	False
SW Mode	Bool	false	False
HMI ON/OFF	Bool	false	False
HMI TestAGV ON	Bool	false	False
HMI TestAGV Abort	Bool	false	False
HMI AutoMode	Bool	false	False
HMI ManualMode	Bool	false	False
HMI ComisngMode	Bool	false	False
HMI MoveUnits	Bool	false	False
HMI MoveAGV	Bool	false	False
StartTestMode	Bool	false	False
EndTestMode	Bool	false	False
ActivateDefaultState	Bool	false	False
STO State	Bool	false	False
Homing	Bool	false	False
Boot Latch	Bool	false	False
▼ Output			
AGV PWR State	Bool	false	False
Automatic Mode	Bool	false	False
Manual Mode	Bool	false	False
Commisioning Mode	Bool	false	False
HMI Move Units	Bool	false	False
HMI Move AGV	Bool	false	False
Homing Mode	Bool	false	False
Testing Mode	Bool	false	False
InOut			
▼ Static			
PowerLatch	Bool	false	False
R_TRIG_ON/OFF_SW	R_TRIG		False
SW ON/OFF Pulse	Bool	false	False
ModeNo	UInt	0	False
SW Mode Pulse	Bool	false	False
R_TRIG_Mode_SW	R_TRIG		False

Name	Data type	Start value	Retain
R_TRIG_ON/OFF_Latch	R_TRIG		False
PWR Latch Pulse	Bool	false	False
HMI_PWR_Pulse	Bool	false	False
R_TRIG_HMI_On/Off_Pulse	R_TRIG		False
StoreLastState	UInt	0	False
ManualModeRLO	Bool	false	False
CommisiongModeRLO	Bool	false	False
AutoModeRLO	Bool	false	False
SWModeRLO	Bool	false	False
TestingModeLatch	Bool	false	False
P_Edge	Array[0..2] of Bool		False
P_StartTestMode	R_TRIG		False
StartTestMode Pulse	Bool	false	False
P_EndTestMode	R_TRIG		False
EndTestMode Pulse	Bool	false	False
N_HomingMode	F_TRIG		False
Homing Pulse	Bool	false	False
P_ActivateDefaultState	R_TRIG		False
ActivateDefaultState Pulse	Bool	false	False
P_TestAbort	R_TRIG		False
TestAbort_Pulse	Bool	false	False

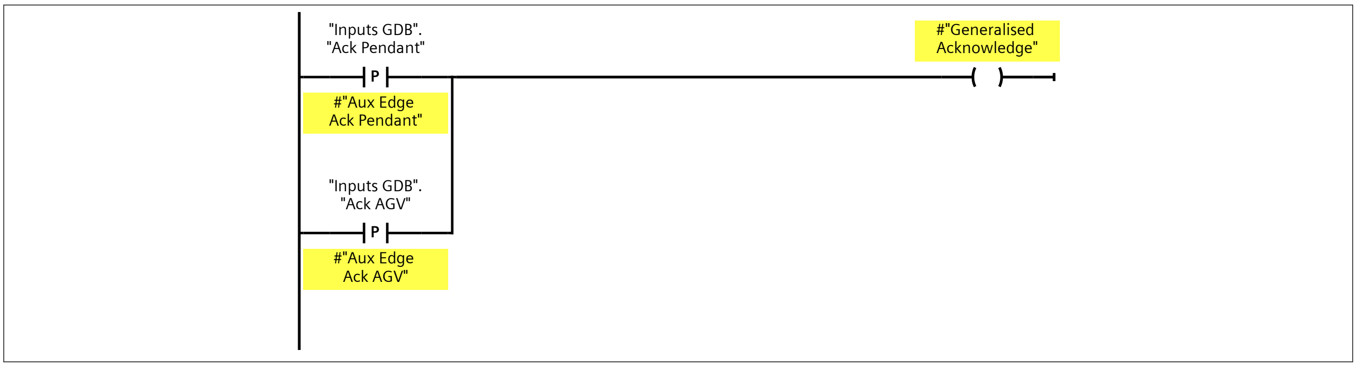
03. Safety

Safety Main RTG [FB0]

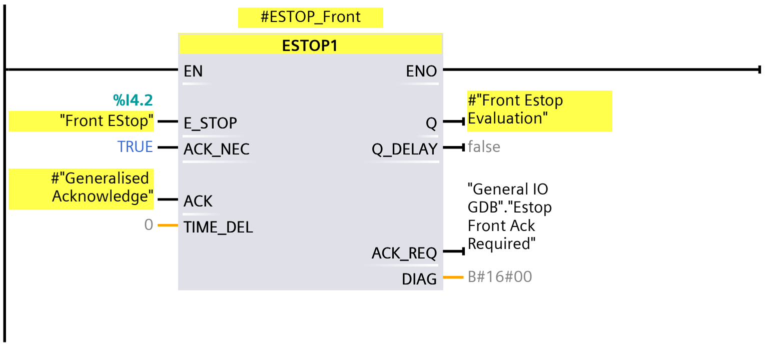
Safety Main RTG Properties					
General					
Name	Safety Main RTG	**Number**	0	**Type**	FB
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	Password: "Siemens1200"
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
Input			
Output			
InOut			
▼ Static			
ESTOP_Front	ESTOP1		
ESTOP_Rear	ESTOP1		
Front Estop Evaluation	Bool	false	Non-retain
Rear Estop Evaluation	Bool	false	Non-retain
Aux Edge Ack Pendant	Bool	false	Non-retain
Aux Edge Ack AGV	Bool	false	Non-retain
Generalised Acknowledge	Bool	false	Non-retain
▼ Temp			
test	Bool		
Constant			

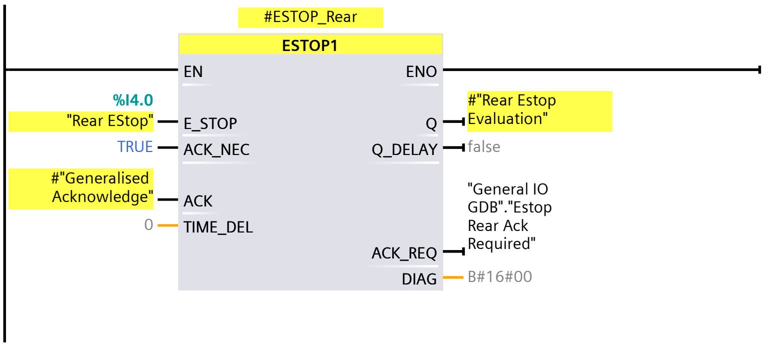
Network 1: Acknowledge Faults



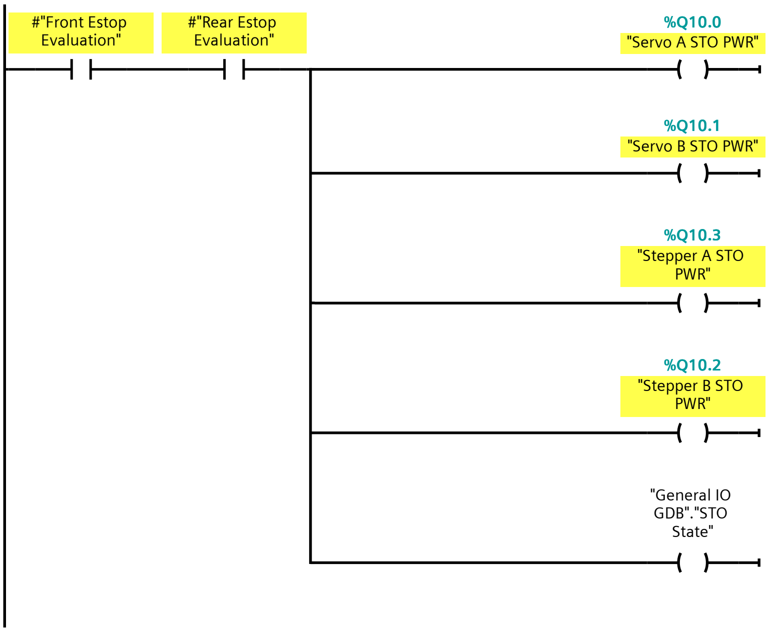
Network 2: Front Estop System



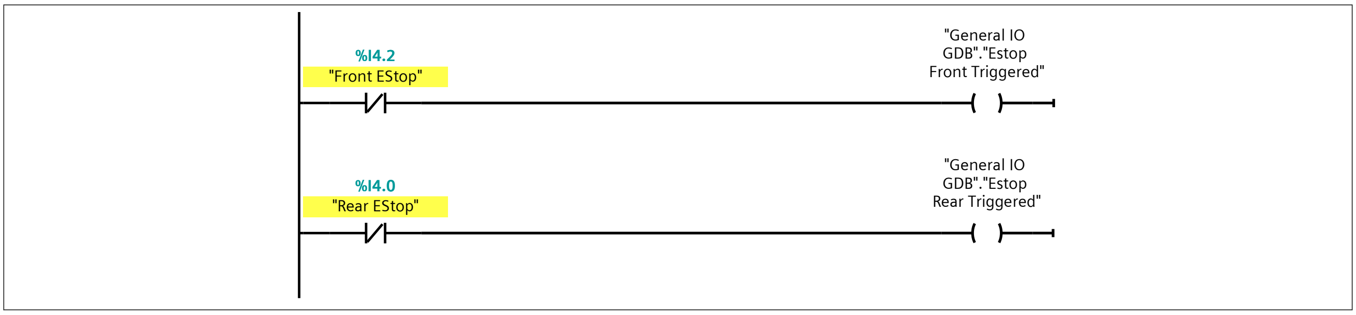
Network 3: Rear Estop System



Network 4: Servo & Stepper Motor STO



Network 5: Register E-Stop Trigger State


%I4.2
"Front EStop"
"General IO GDB"."Estop Front Triggered"
%I4.0
"Rear EStop"
"General IO GDB"."Estop Rear Triggered"

03. Safety

Safety Main RTG iDB [DB1]

Safety Main RTG iDB Properties					
General					
Name	Safety Main RTG iDB	**Number**	1	**Type**	DB
Language	DB	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	FUSI

Name	Data type	Start value	Retain
Input			
Output			
InOut			
▼ Static			
ESTOP_Front	ESTOP1		False
ESTOP_Rear	ESTOP1		False
Front Estop Evaluation	Bool	false	False
Rear Estop Evaluation	Bool	false	False
Aux Edge Ack Pendant	Bool	false	False
Aux Edge Ack AGV	Bool	false	False
Generalised Acknowledge	Bool	false	False

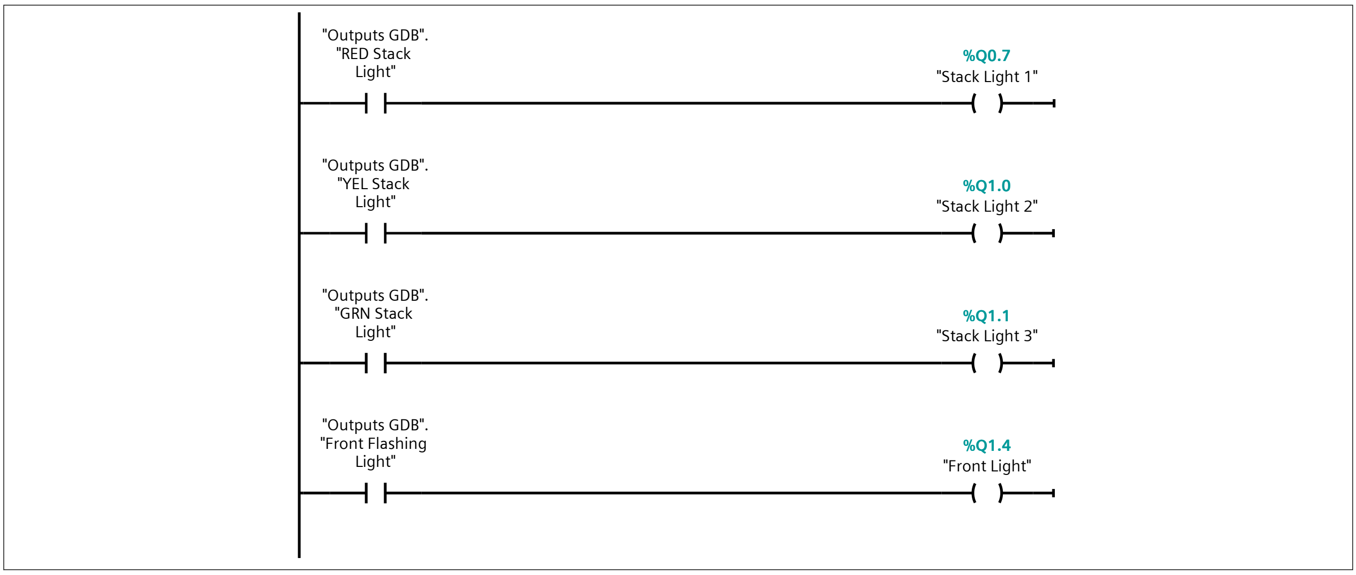
04. IO Control

FC Outputs [FC102]

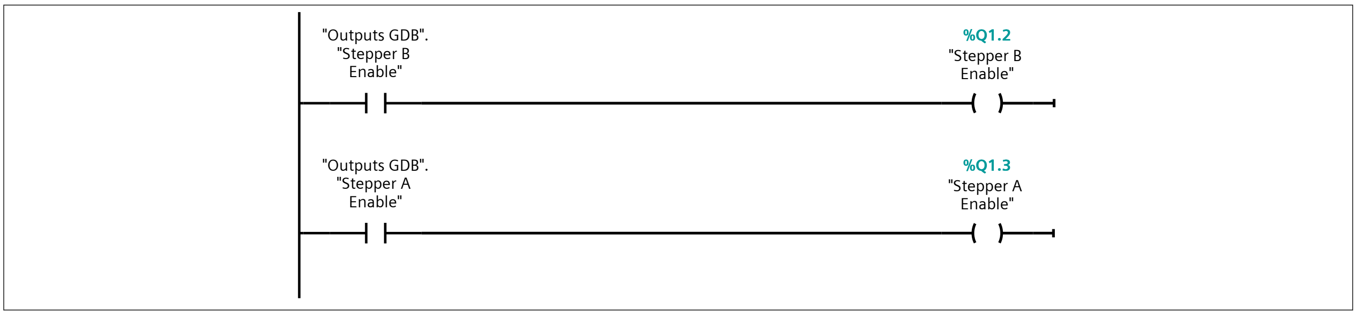
FC Outputs Properties					
General					
Name	FC Outputs	**Number**	102	**Type**	FC
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value
Input		
Output		
InOut		
Temp		
Constant		
▼ Return		
FC Outputs	Void	

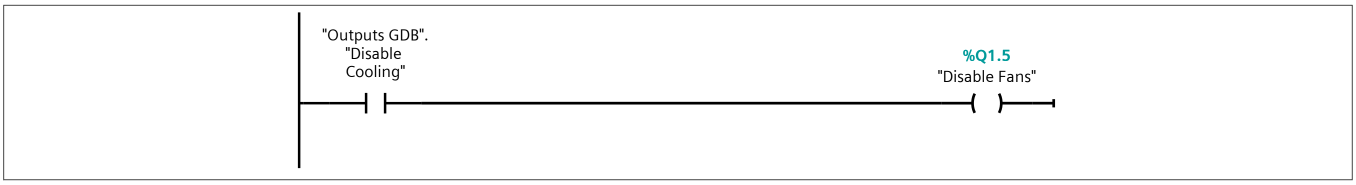
Network 1: Connect DB items to Physical Outputs (Lights)



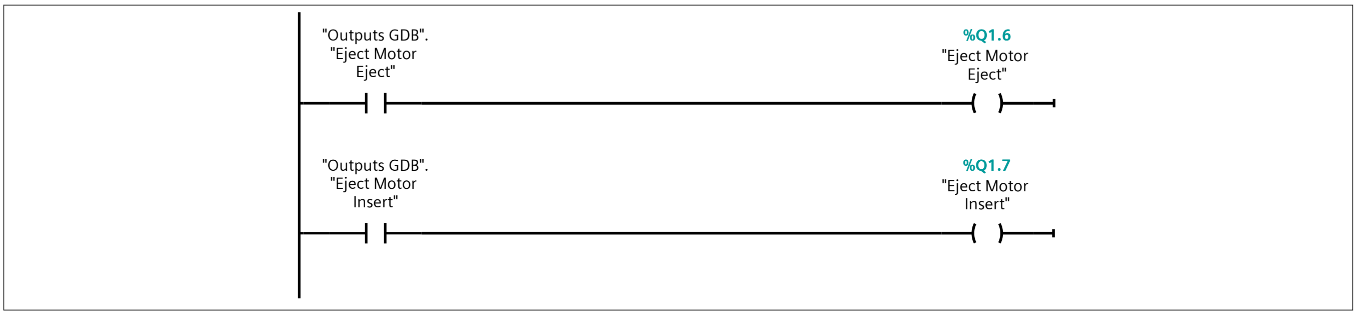
Network 2: Connect DB items to Physical Outputs (Stepper Motor Non Safe IO)



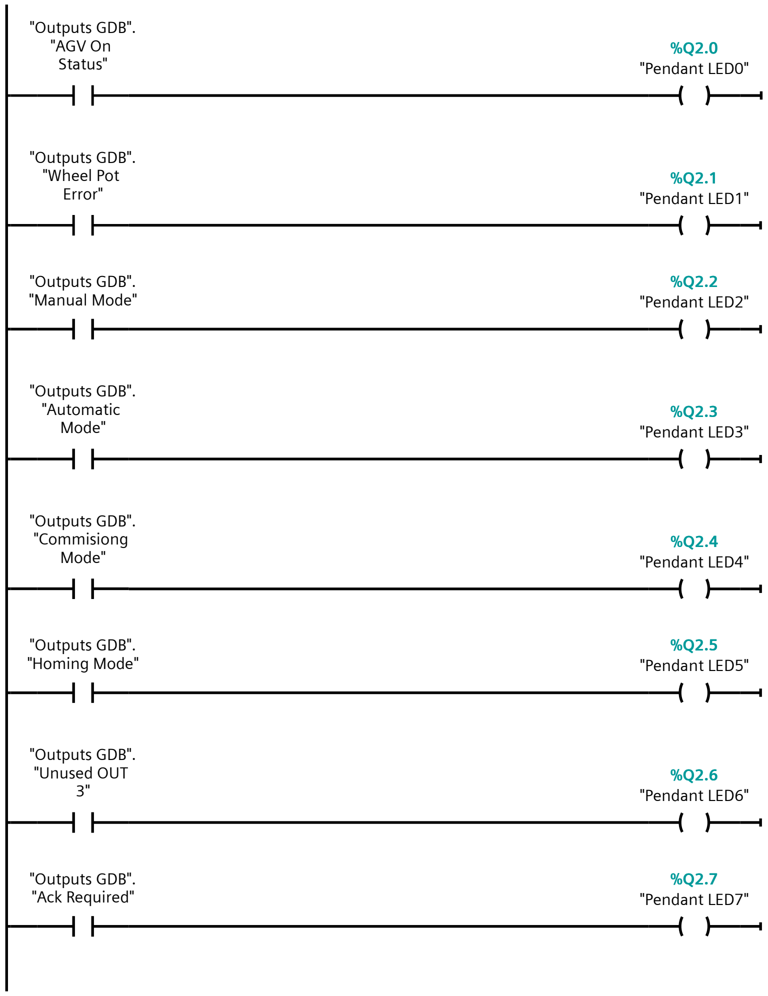
Network 3: Connect DB items to Physical Outputs (Cooling System)



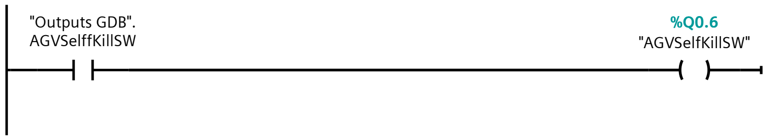
Network 4: Connect DB items to Physical Outputs (Battery Bank Eject)



Network 5: Connect DB items to Physical Outputs (Festo Pendant)



Network 6: Connect DB items to Physical Outputs (Shutdown)



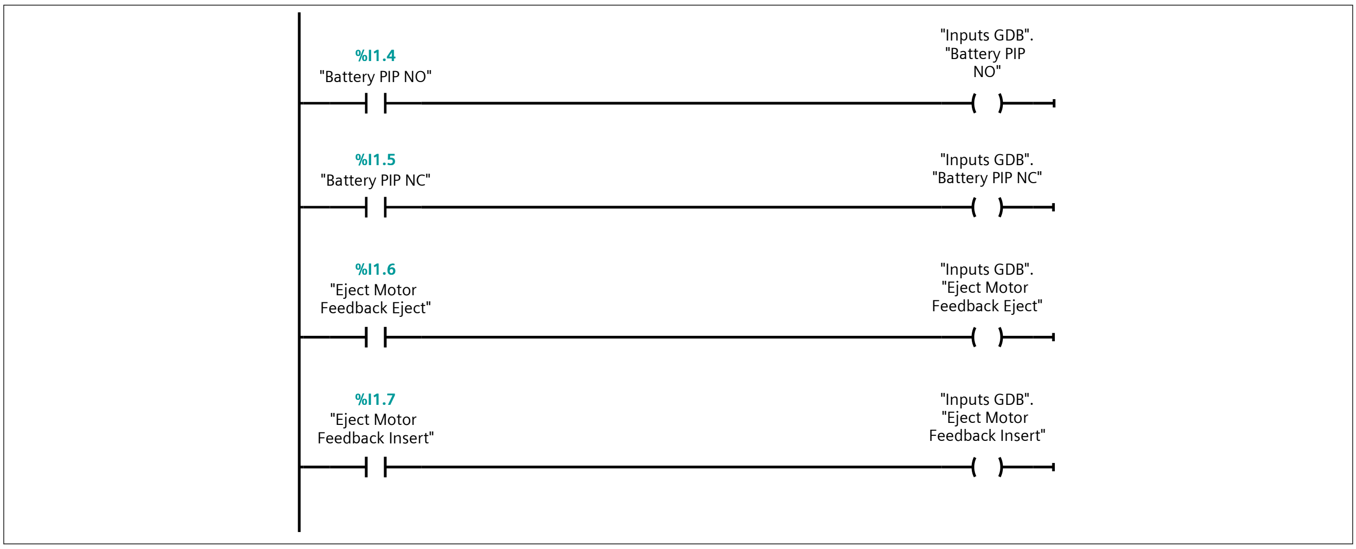
04. IO Control

FC Inputs [FC101]

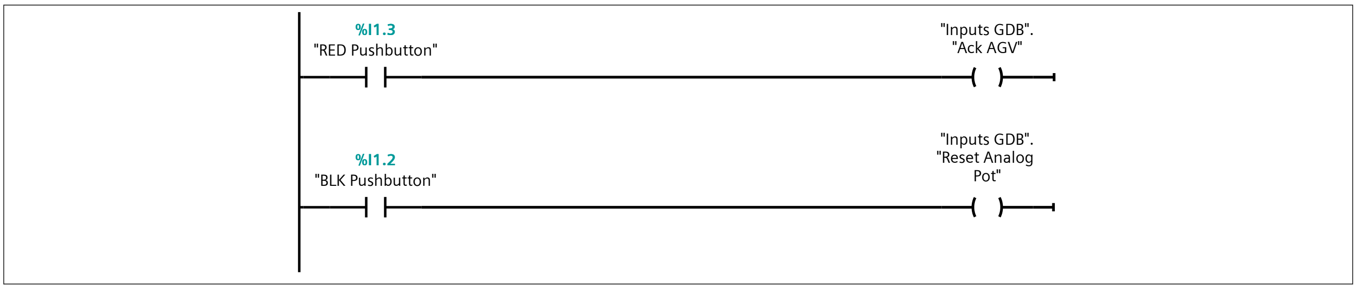
FC Inputs Properties					
General					
Name	FC Inputs	**Number**	101	**Type**	FC
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value
Input		
Output		
InOut		
Temp		
Constant		
▼ Return		
FC Inputs	Void	

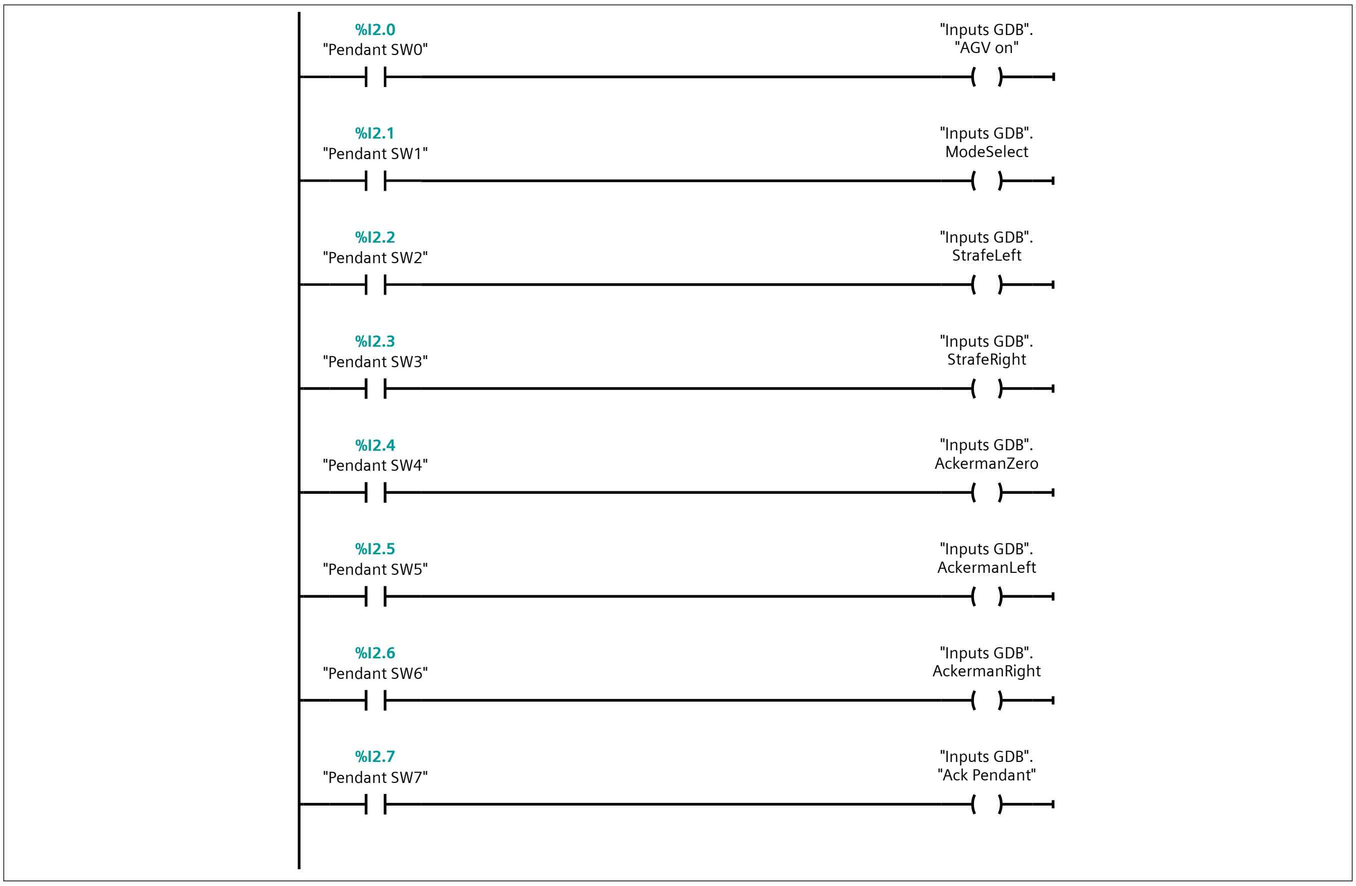
Network 1: Connect physical inputs to DB items (Battery System)



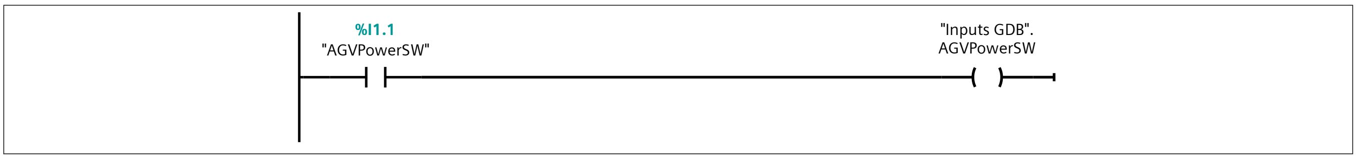
Network 2: Connect physcial inputs to DB items (Reset Pendant AGV Front)



Network 3: Connect physical inputs to DB items (Festo Pendant)



Network 4: Shutdown



04. IO Control

FC Markers [FC100]

FC Markers Properties					
General					
Name	FC Markers	**Number**	100	**Type**	FC
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value
Input		
Output		
InOut		
Temp		
Constant		
▼ Return		
FC Markers	Void	

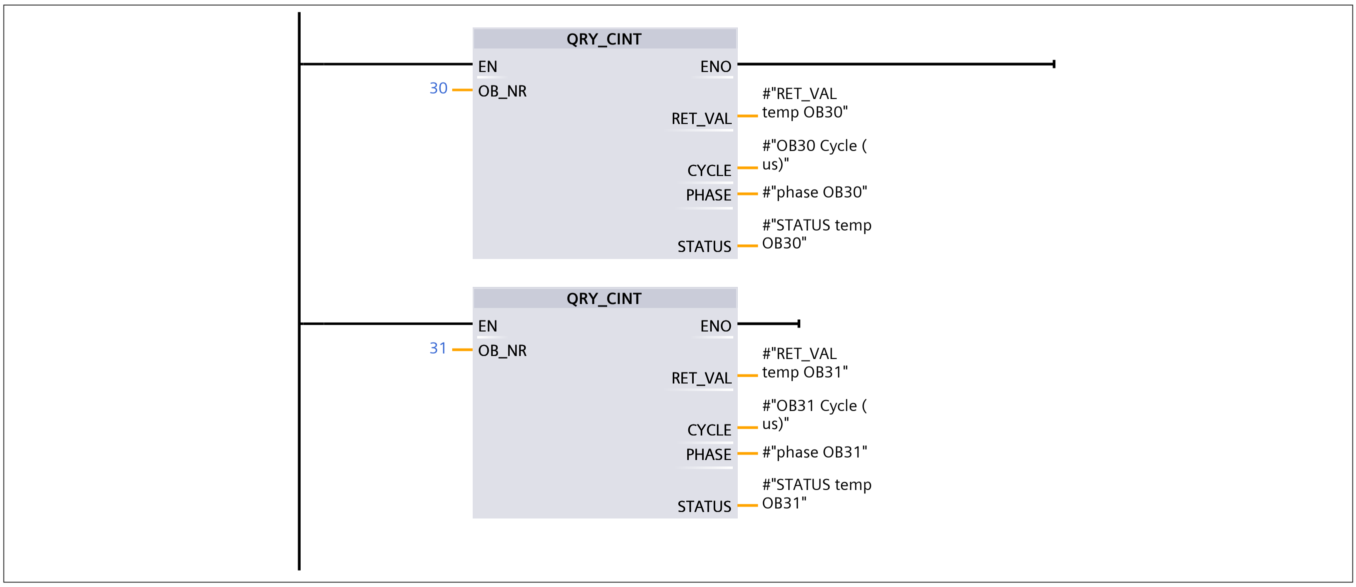
04. IO Control

FC Get Cycle Time [FC162]

FC Get Cycle Time Properties					
General					
Name	FC Get Cycle Time	**Number**	162	**Type**	FC
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

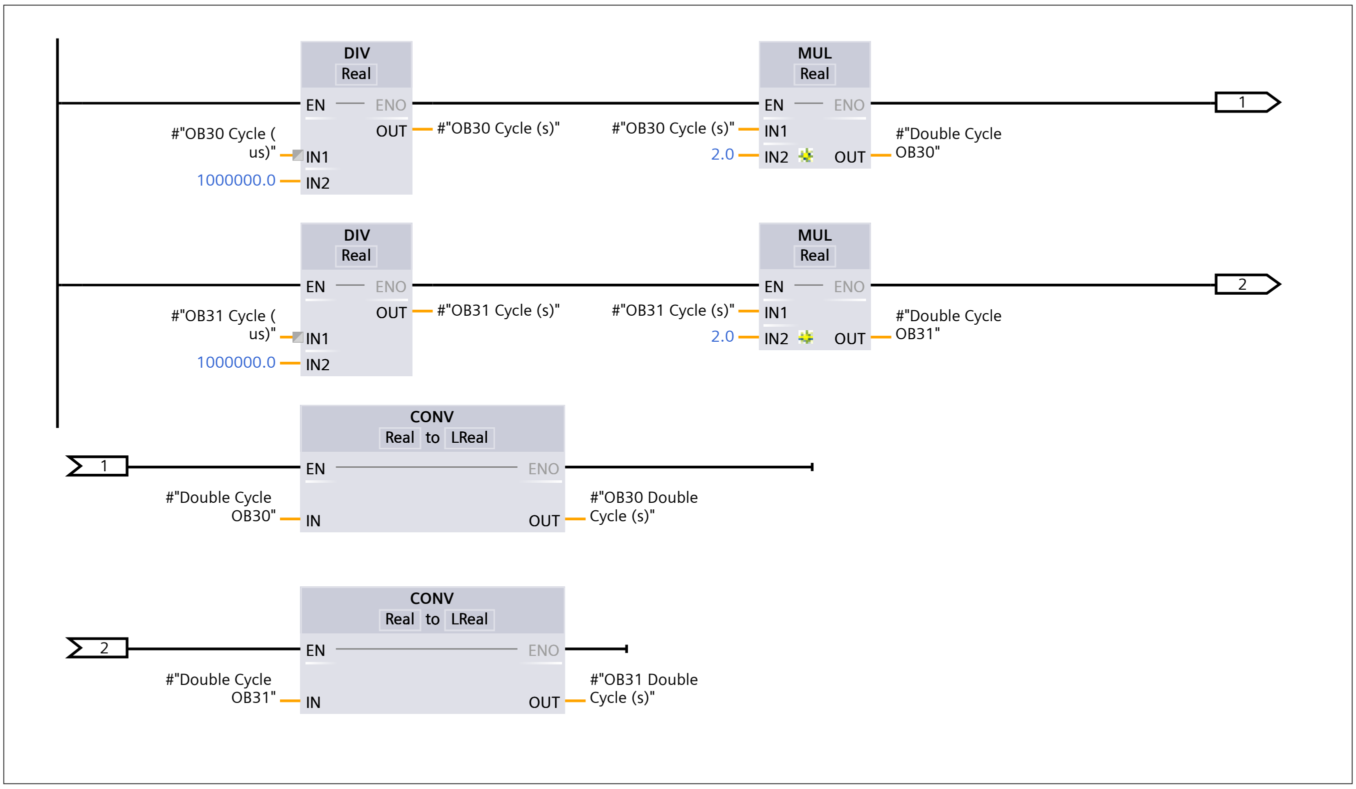
Name	Data type	Default value
Input		
▼ Output		
OB30 Double Cycle (s)	LReal	
OB30 Time Ready	Bool	
OB31 Double Cycle (s)	LReal	
OB31 Time Ready	Bool	
▼ InOut		
OB30 Cycle (s)	Real	
OB30 Cycle (us)	UDInt	
OB31 Cycle (s)	Real	
OB31 Cycle (us)	UDInt	
▼ Temp		
RET_VAL temp OB30	Int	
RET_VAL temp OB31	Int	
phase OB30	UDInt	
phase OB31	UDInt	
STATUS temp OB30	Word	
STATUS temp OB31	Word	
Double Cycle OB30	Real	
Double Cycle OB31	Real	
Constant		
▼ Return		
FC Get Cycle Time	Void	

Network 1: Query the cycletime of OB30

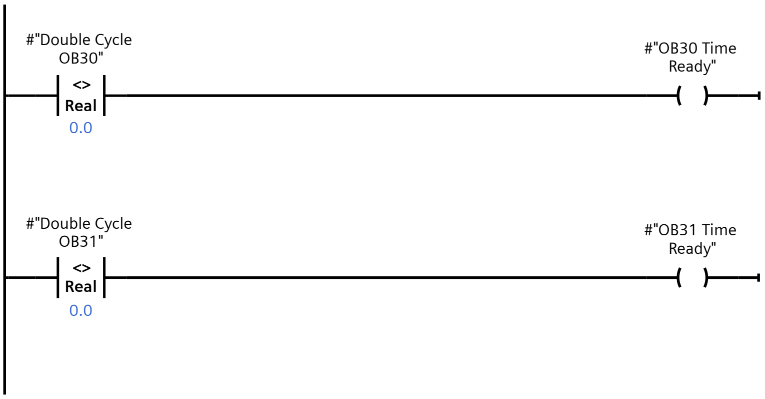


Network 2: Convert from microseconds to seconds

Network 2: Convert from microseconds to seconds



Network 3:


#"Double Cycle OB30"
<>
Real
0.0
#"OB30 Time Ready"
#"Double Cycle OB31"
<>
Real
0.0
#"OB31 Time Ready"

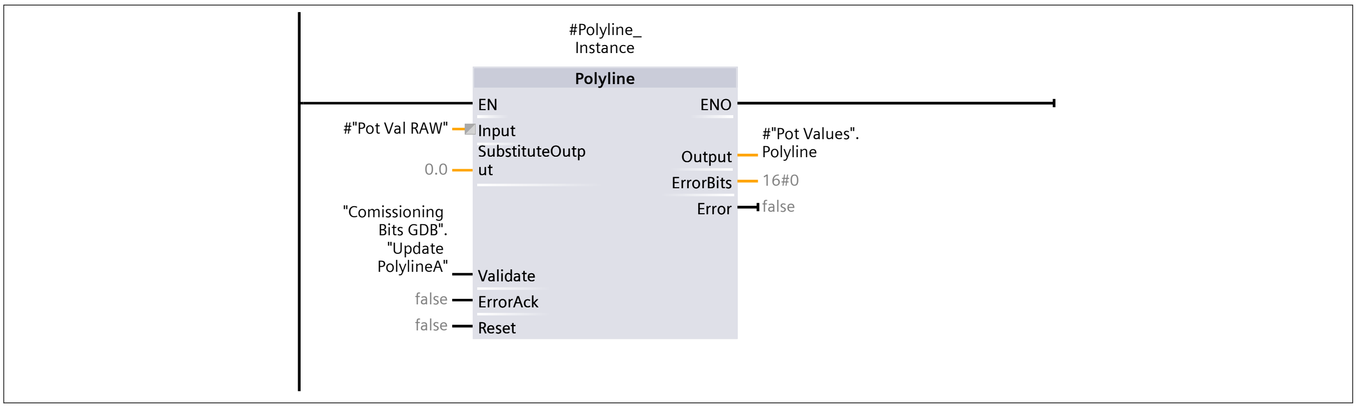
05. Analog Steering Pots

FB Pot A Read [FB160]

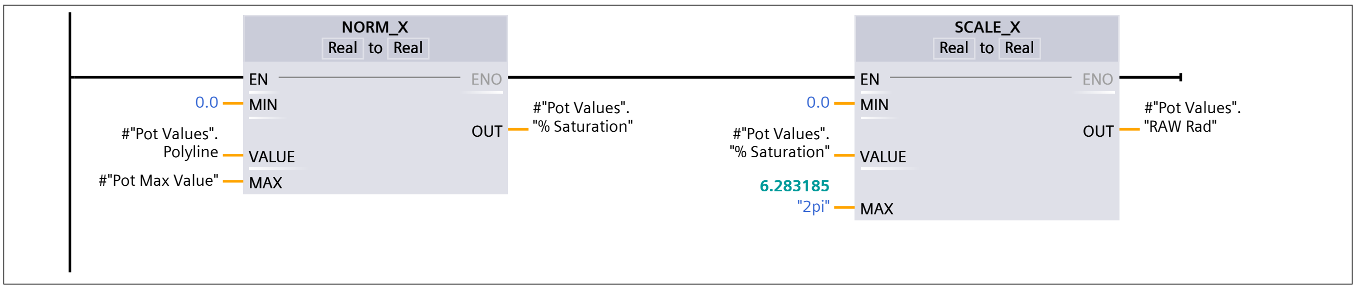
FB Pot A Read Properties					
General					
Name	FB Pot A Read	**Number**	160	**Type**	FB
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
▼ Input			
Pot Val RAW	Int	0	Non-retain
Pot Max Value	Real	0.0	Non-retain
Update Pot Offset	Bool	false	Non-retain
Trigger Differential	Bool	false	Non-retain
deltaT	LReal	0.0	Non-retain
▼ Output			
Angle (rad)	Real	0.0	Non-retain
Angle (deg)	Real	0.0	Non-retain
Psuedo Speed (rpm)	Real	0.0	Non-retain
▼ InOut			
Pot Offset	Real	0.0	Non-retain
▼ Static			
PotValues	Struct		Non-retain
Polyline_Instance	Polyline		
Pot Values	Struct		Non-retain
LGF_DifferenceQuotientFB_Instance	"LGF_DifferenceQuotientFB"		
last cycle CCW	Array[0..4] of Real		Non-retain
values	Struct		Non-retain
AuxPurgeZero	Int	0	Non-retain
LockoutPurge	Bool	false	Non-retain
Temp			
▼ Constant			
RPM noise Lim +	Real	0.09	
RPM noise Lim -	Real	-0.09	

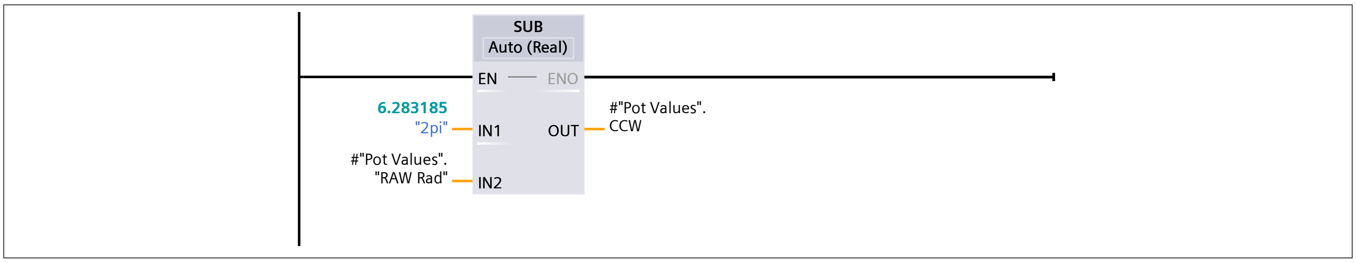
Network 1: Read and Scale RAW pot value



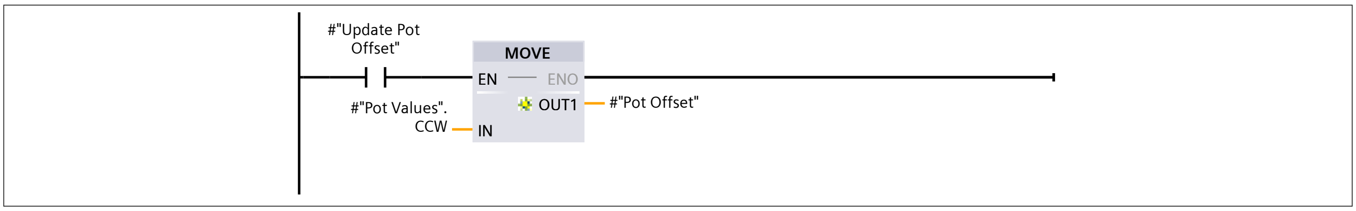
Network 2: Normalise Polyangle to get saturation percentage



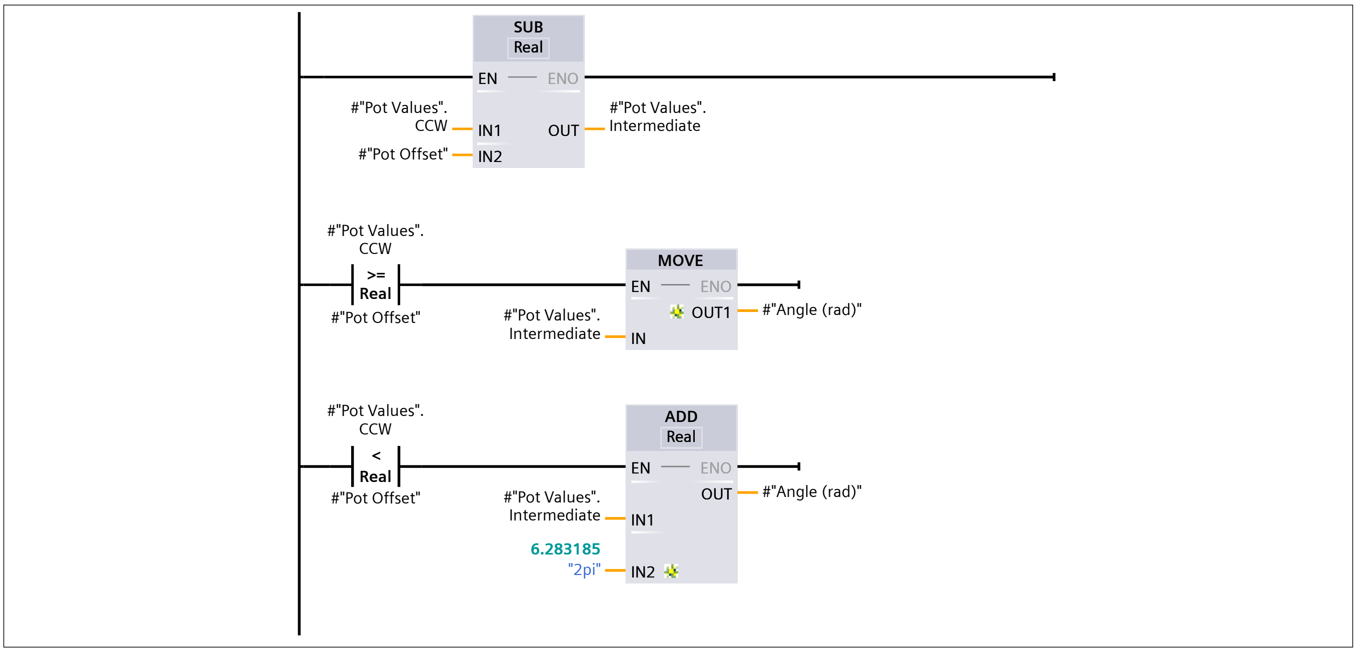
Network 3: invert from CW to CCW



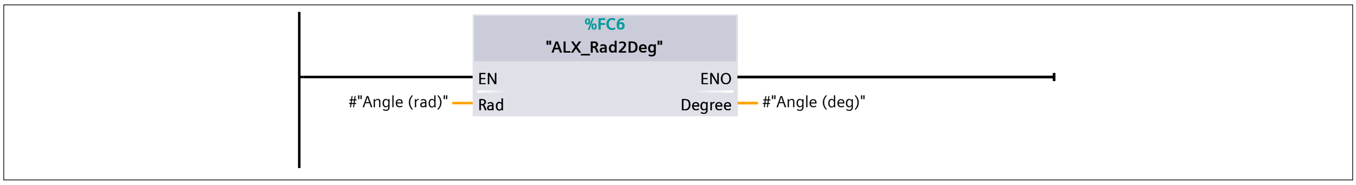
Network 4: Update Pot Offset



Network 5: remove potentiometer offset



Network 6: convert to degrees

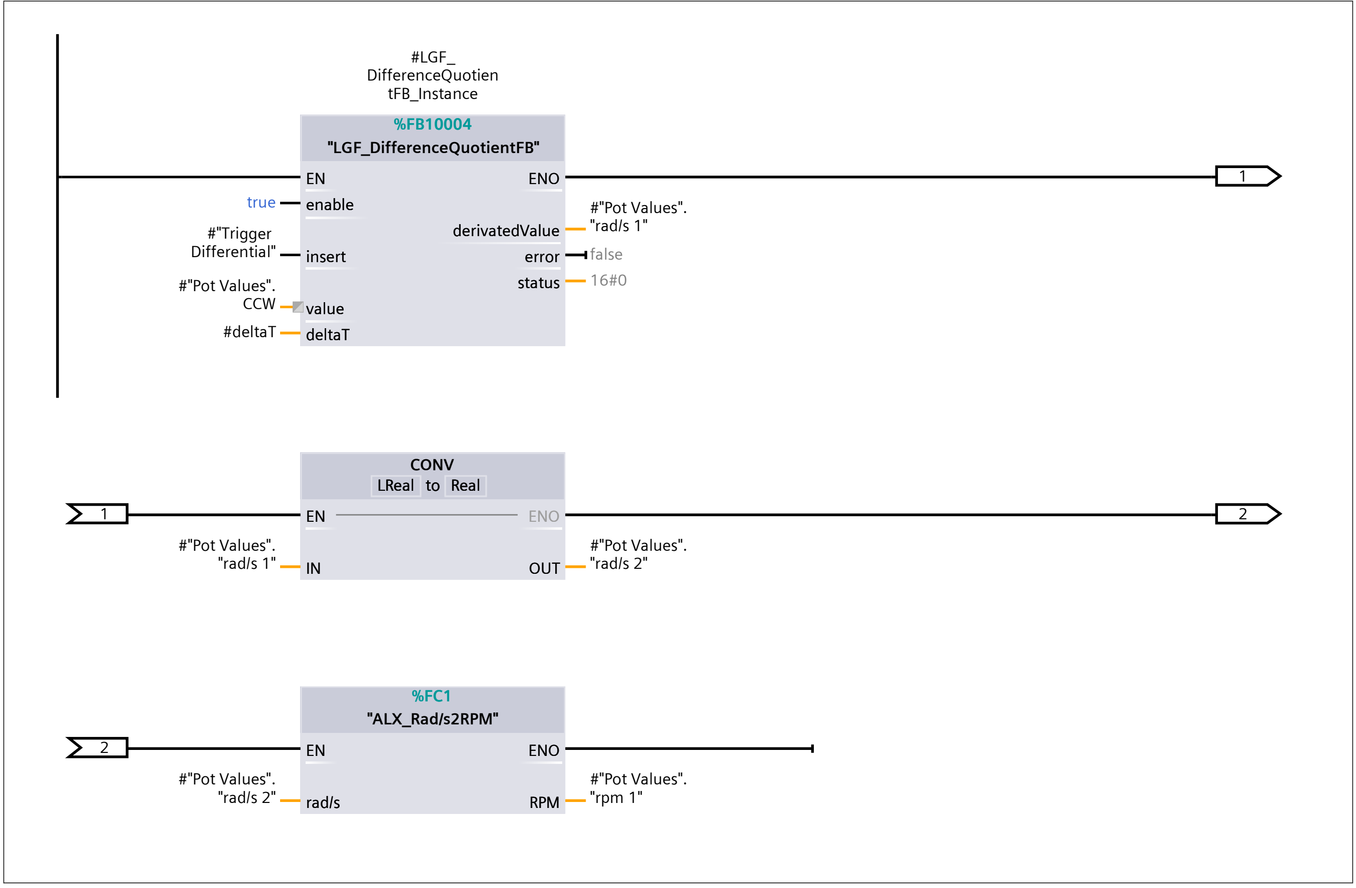


Network 7: check if differentiated values will use swap over from 360 -> 0 or 0 -> 360, in which case this value should be purged

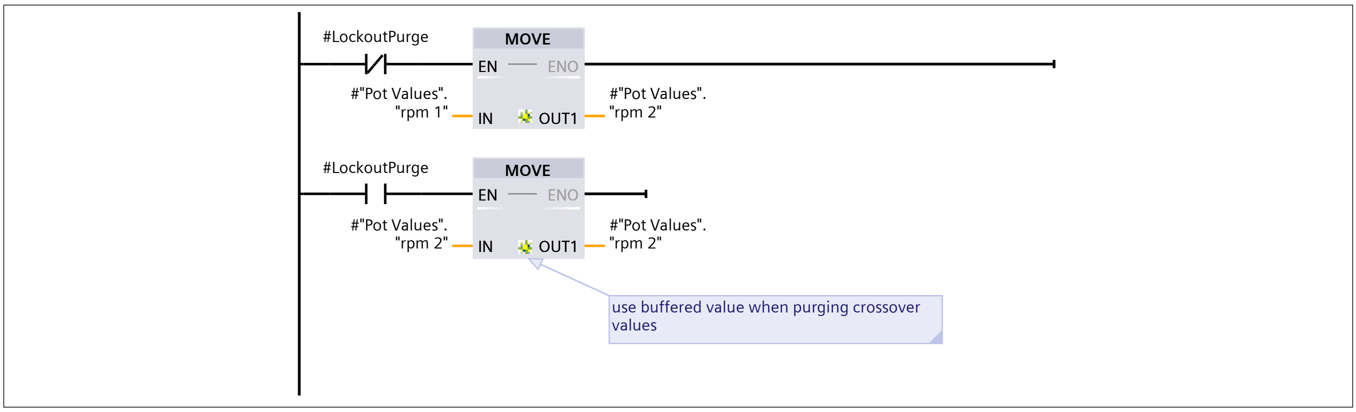
```
//shift values by one place if differrentiator triggered
//
IF #"Trigger Differential" THEN
    // Statement section IF
    #"last cycle CCW"[0] := #"last cycle CCW"[1];
    #"last cycle CCW"[1] := #"last cycle CCW"[2];
    #"last cycle CCW"[2] := #"last cycle CCW"[3];
    #"last cycle CCW"[3] := #"last cycle CCW"[4];
    #"last cycle CCW"[4] := #"Pot Values".CCW;

    #values.temp_diff1 := #"last cycle CCW"[4] - #"last cycle CCW"[3];
    #values.temp_diff := ABS(#values.temp_diff1);

    IF #values.temp_diff > 5.49779 THEN
        // if difference bigger than 315 degrees (5.49779 radians) then most likly crossed over zero point
        #AuxPurgeZero := 5;
    END_IF;

    IF #AuxPurgeZero <> 0 THEN
        // need to purge the buffer of crossover values

        #LockoutPurge := TRUE;

        #AuxPurgeZero := #AuxPurgeZero - 1;

    ELSIF #AuxPurgeZero <= 0 THEN
        // purge of lockout values completed

        #LockoutPurge := FALSE;

    ELSE
        // Statement section ELSE
        ;
    END_IF;

END_IF;
```

Network 8: Defferentiate to get rpm

Network 8: Defferentiate to get rpm

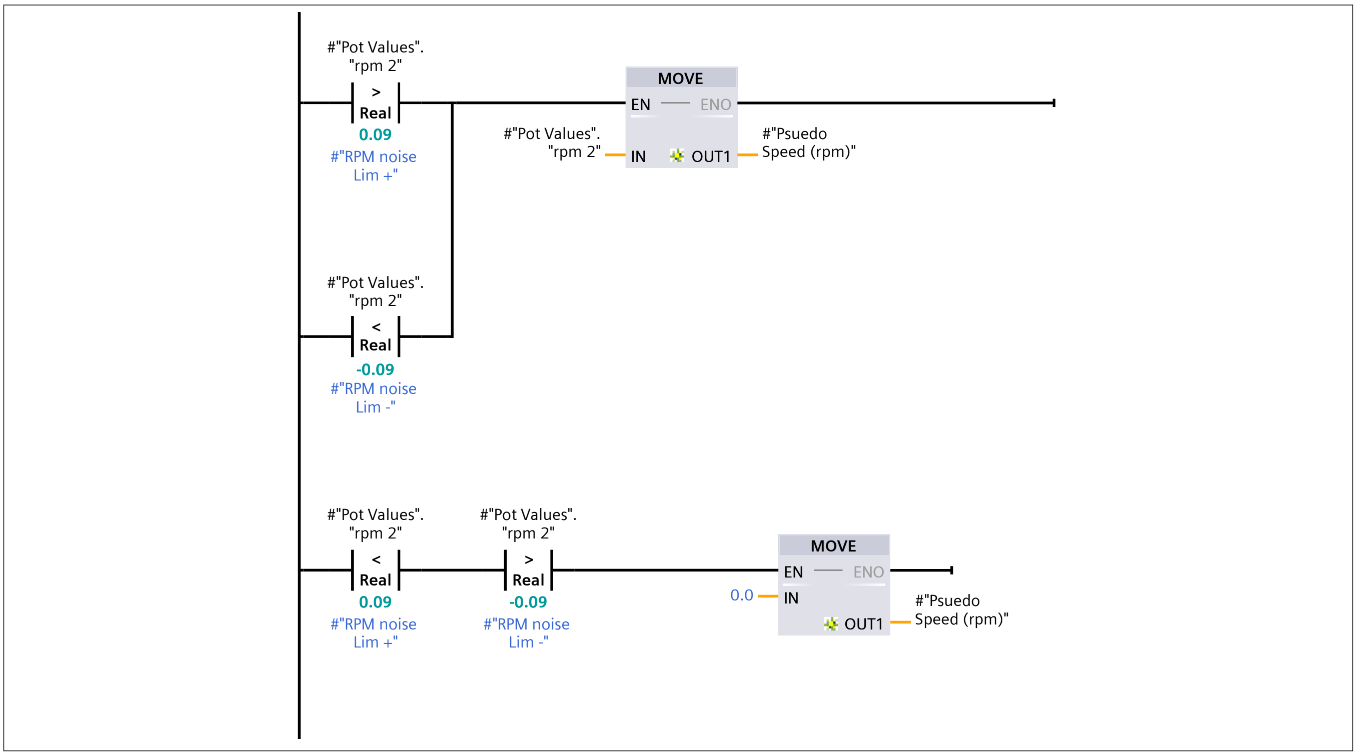


Network 9: Ignore values at crossover



use buffered value when purging crossover values

Network 10: Is moving OR noise?



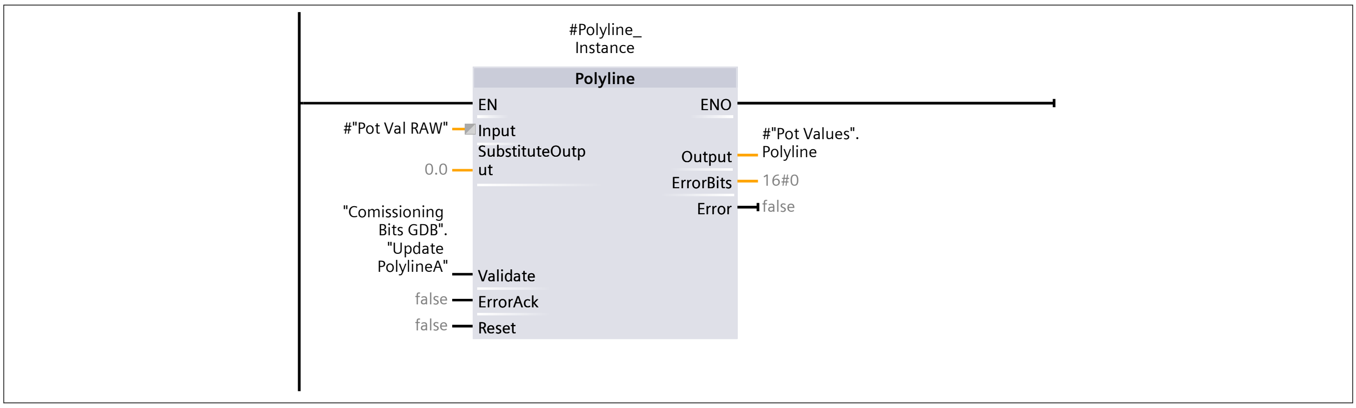
05. Analog Steering Pots

FB Pot B Read [FB161]

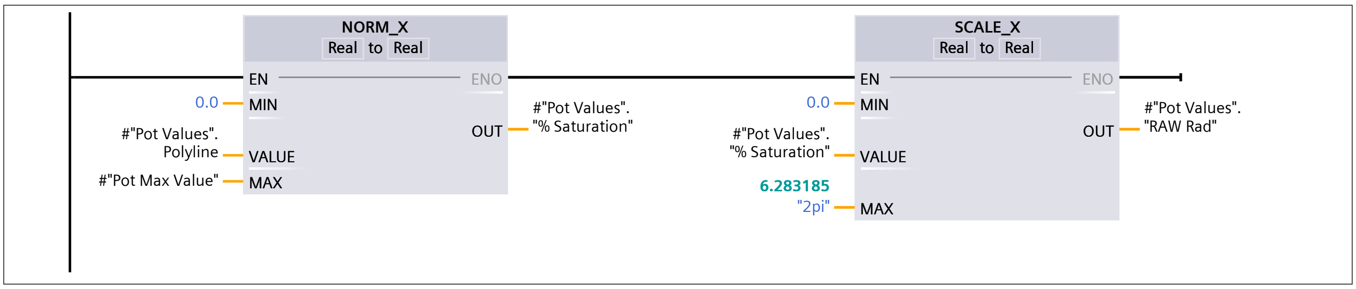
FB Pot B Read Properties					
General					
Name	FB Pot B Read	**Number**	161	**Type**	FB
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
▼ Input			
Pot Val RAW	Int	0	Non-retain
Pot Max Value	Real	0.0	Non-retain
Update Pot Offset	Bool	false	Non-retain
Trigger Differential	Bool	false	Non-retain
deltaT	LReal	0.0	Non-retain
▼ Output			
Angle (rad)	Real	0.0	Non-retain
Angle (deg)	Real	0.0	Non-retain
Psuedo Speed (rpm)	Real	0.0	Non-retain
▼ InOut			
Pot Offset	Real	0.0	Non-retain
▼ Static			
PotValues	Struct		Non-retain
Pot Values	Struct		Non-retain
LGF_DifferenceQuotientFB_Instance	"LGF_Difference-QuotientFB"		
Polyline_Instance	Polyline		
last cycle CCW	Array[0..4] of Real		Non-retain
values	Struct		Non-retain
AuxPurgeZero	Int	0	Non-retain
LockoutPurge	Bool	false	Non-retain
Temp			
▼ Constant			
RPM noise Lim +	Real	0.09	
RPM noise Lim -	Real	-0.09	

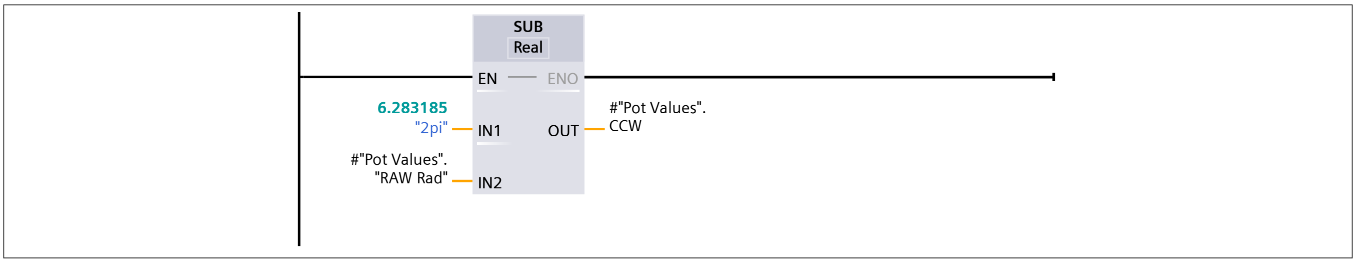
Network 1: Read and Scale RAW pot value



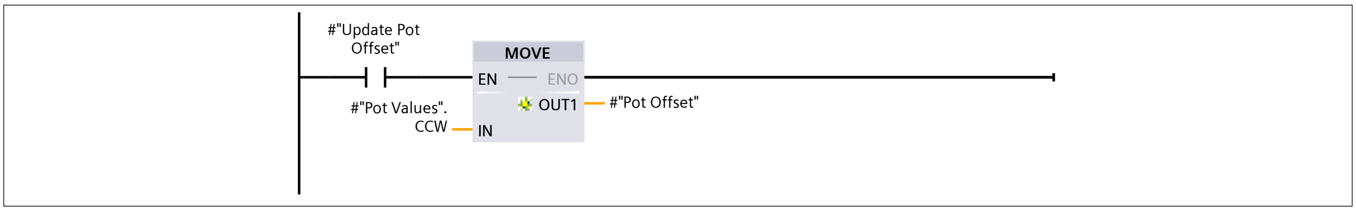
Network 2: Normalise Polyangle to get saturation percentage



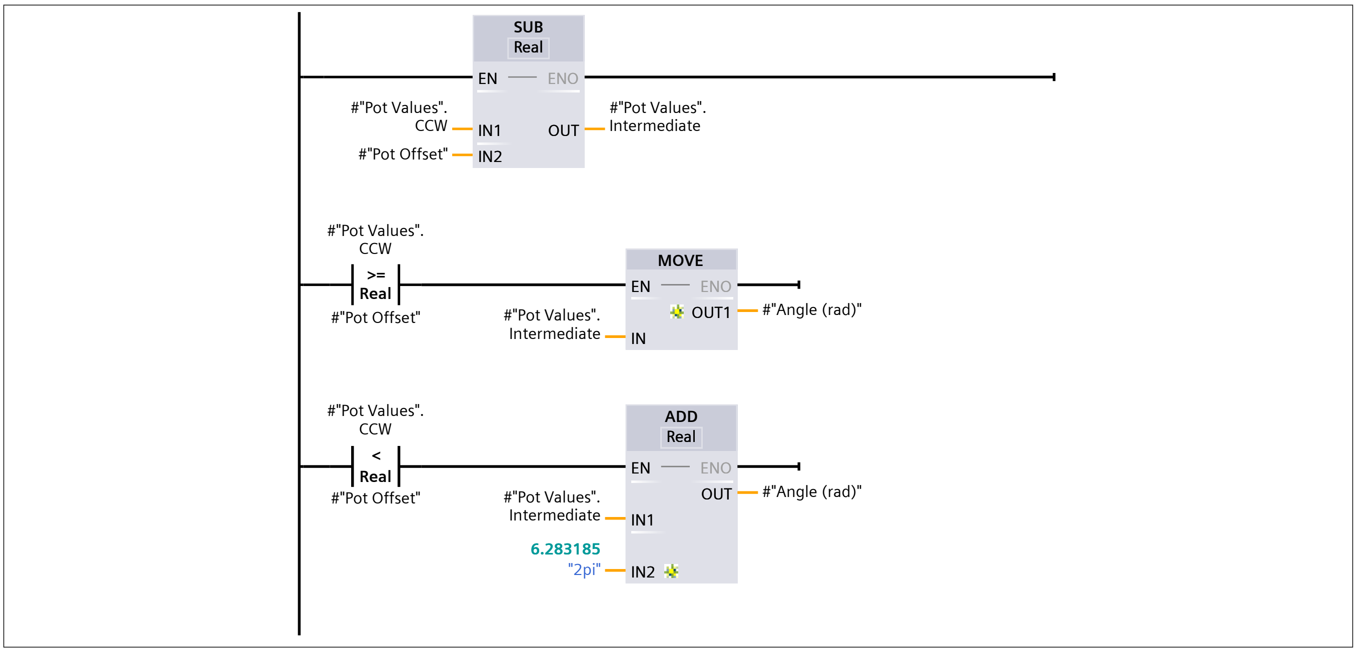
Network 3: invert from CW to CCW



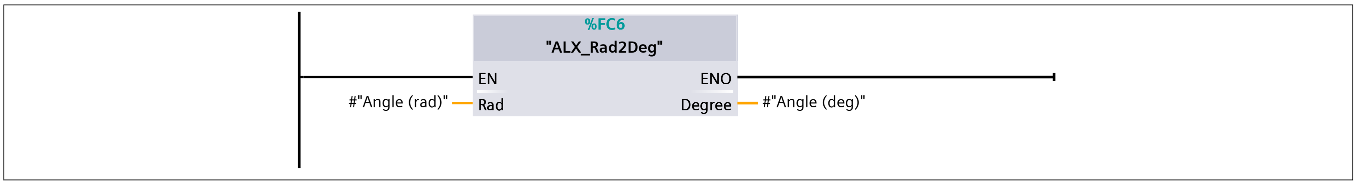
Network 4: Update Pot Offset



Network 5: Remove potentiometer offset



Network 6: convert to degrees

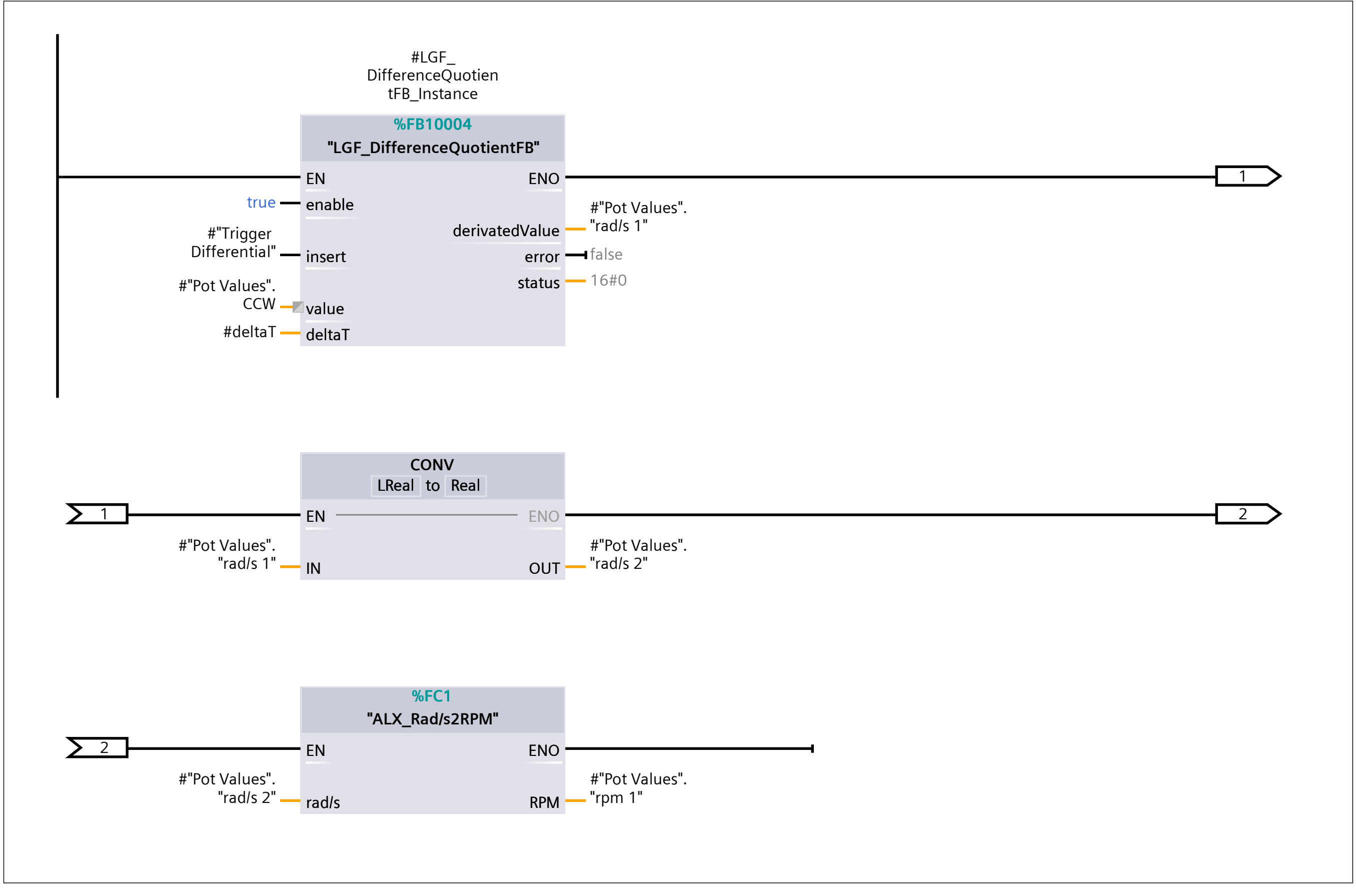


Network 7: check if differentiated values will use swap over from 360 -> 0 or 0 -> 360, in which case this value should be purged

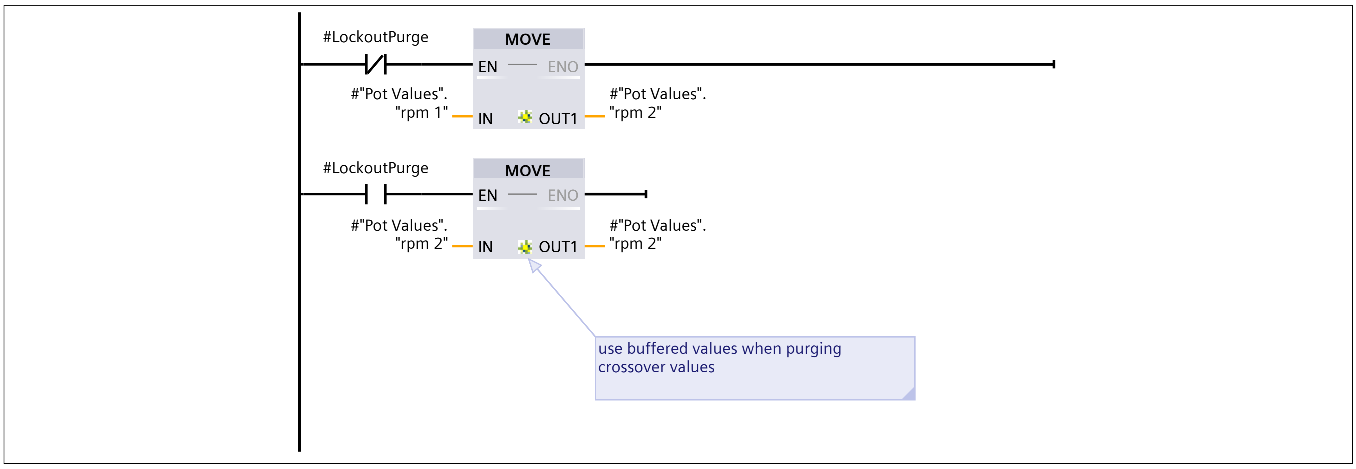
```
//shift values by one place if differrentiator triggered
//
IF #"Trigger Differential" THEN
    // Statement section IF
    #"last cycle CCW"[0] := #"last cycle CCW"[1];
    #"last cycle CCW"[1] := #"last cycle CCW"[2];
    #"last cycle CCW"[2] := #"last cycle CCW"[3];
    #"last cycle CCW"[3] := #"last cycle CCW"[4];
    #"last cycle CCW"[4] := #"Pot Values".CCW;

    #values.temp_diff1 := #"last cycle CCW"[4] - #"last cycle CCW"[3];
    #values.temp_diff := ABS(#values.temp_diff1);

    IF #values.temp_diff > 5.49779 THEN
        // if difference bigger than 315 degrees (5.49779 radians) then most likly crossed over zero point
        #AuxPurgeZero := 5;
    END_IF;

    IF #AuxPurgeZero <> 0 THEN
        // need to purge the buffer of crossover values

        #LockoutPurge := TRUE;

        #AuxPurgeZero := #AuxPurgeZero - 1;

    ELSIF #AuxPurgeZero <= 0 THEN
        // purge of lockout values completed

        #LockoutPurge := FALSE;

    ELSE
        // Do nothing
        ;
    END_IF;

END_IF;
```

Network 8: Defferentiate to get rpm

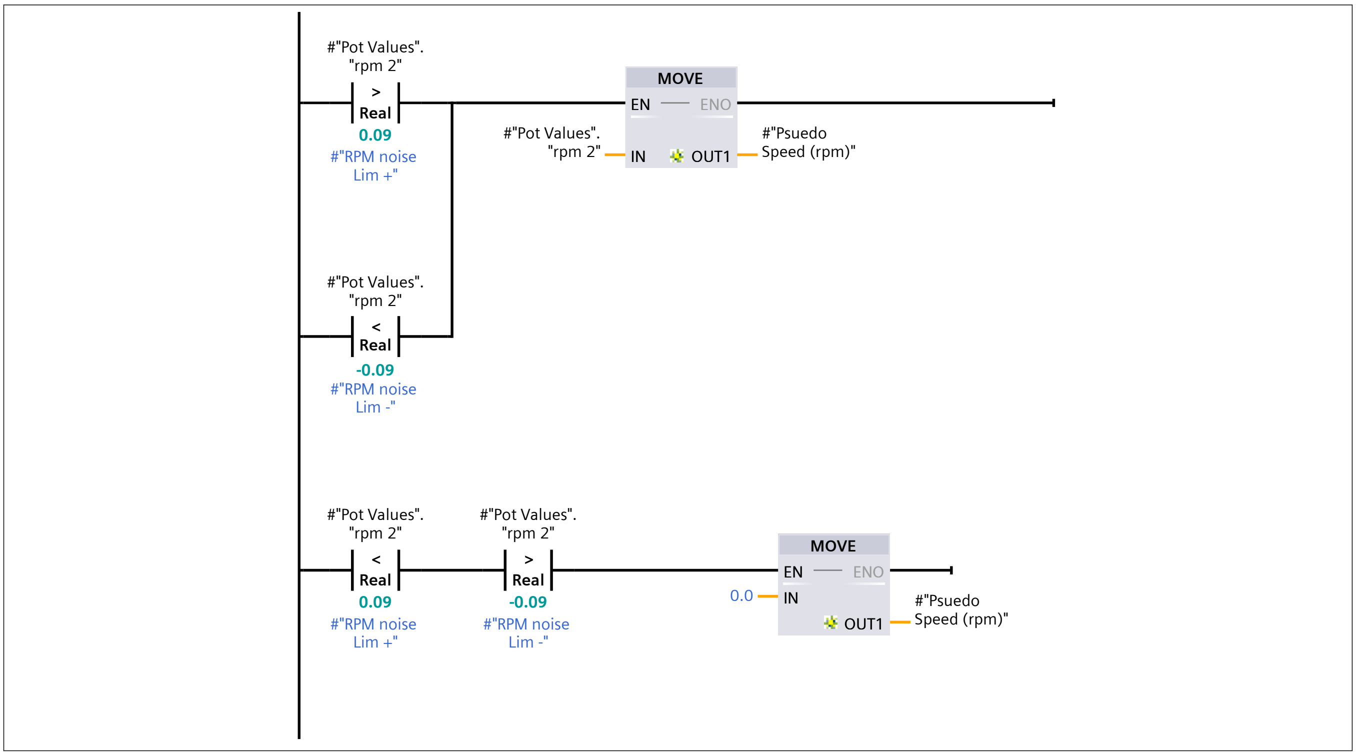
Network 8: Defferentiate to get rpm



Network 9: Ignore values at crossover



Network 10: Is moving OR noise?



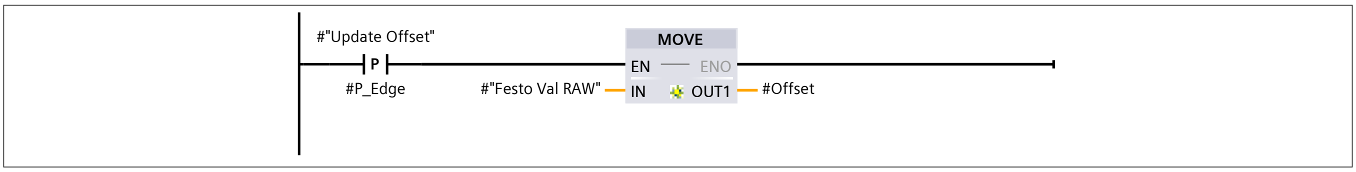
05. Analog Steering Pots

FB Festo A Analog Read [FB163]

FB Festo A Analog Read Properties					
General					
Name	FB Festo A Analog Read	**Number**	163	**Type**	FB
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

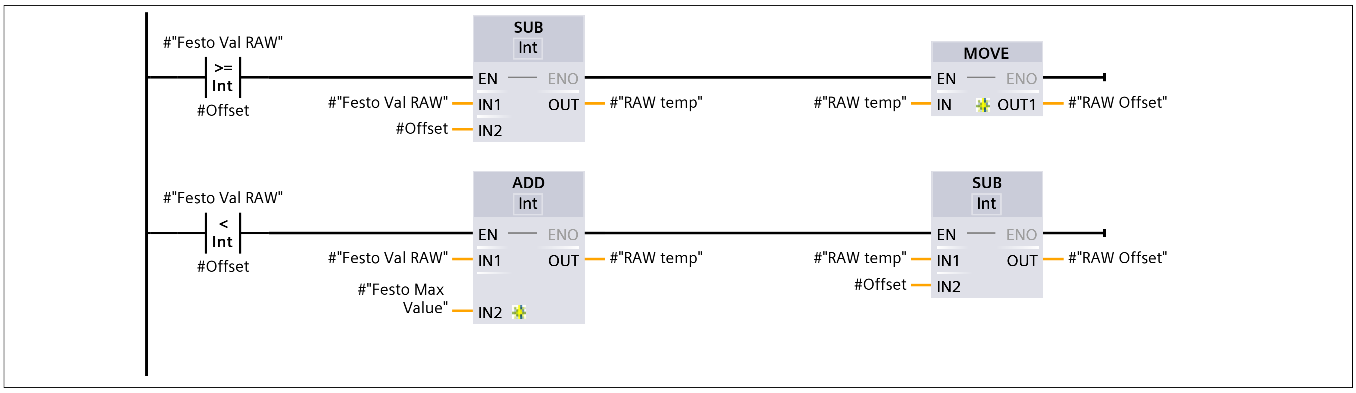
Name	Data type	Default value	Retain
▼ Input			
Festo Val RAW	Int	0	Non-retain
Festo Max Value	Int	27648	Non-retain
Update Offset	Bool	false	Non-retain
▼ Output			
Angle (rad)	Real	0.0	Non-retain
Angle (deg)	Real	0.0	Non-retain
▼ InOut			
Offset	Int	13904	Non-retain
▼ Static			
P_Edge	Bool	false	Non-retain
RAW Offset	Int	0	Non-retain
RAW temp	Int	0	Non-retain
Festo Val %	Real	0.0	Non-retain
Festo RAW rad	Real	0.0	Non-retain
Festo Value CCW	Real	0.0	Non-retain
Temp			
Constant			

Network 1: Update Analog Offset

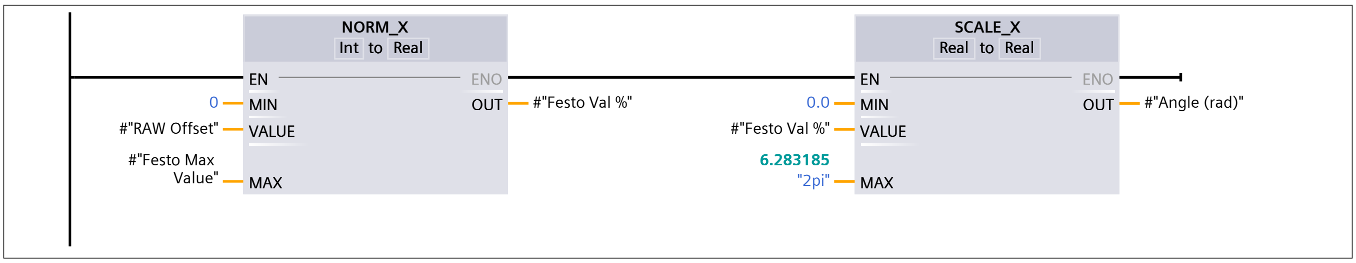


Network 2: Remove Offset from RAW pot value

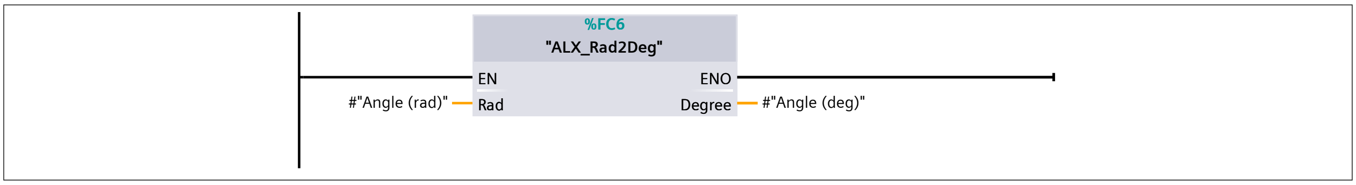
A 5V offset is provided to the analog value thus 5V = 0 deg. This was done to move the 0V -> 10V changeover to the 180 deg position. Since this changeover between voltages causes a volateg ramp that causes the system to erroneously measure the current angle as changing even if it is not moving. 180 degress is full reverse, where it is thought that this instablity will have the least effect on te system.



Network 3: Normalise and scale incoming analog value



Network 4: Convert to degrees



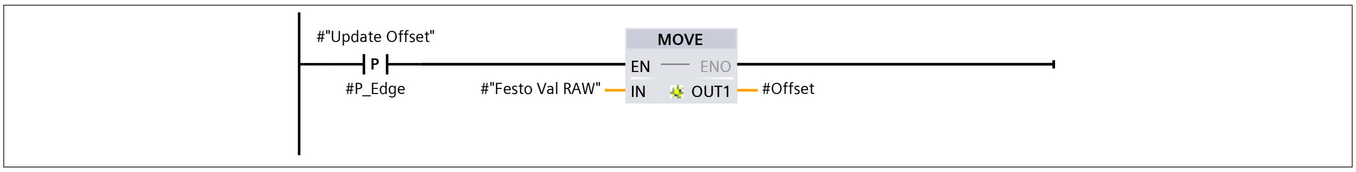
05. Analog Steering Pots

FB Festo B Analog Read [FB164]

FB Festo B Analog Read Properties					
General					
Name	FB Festo B Analog Read	**Number**	164	**Type**	FB
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

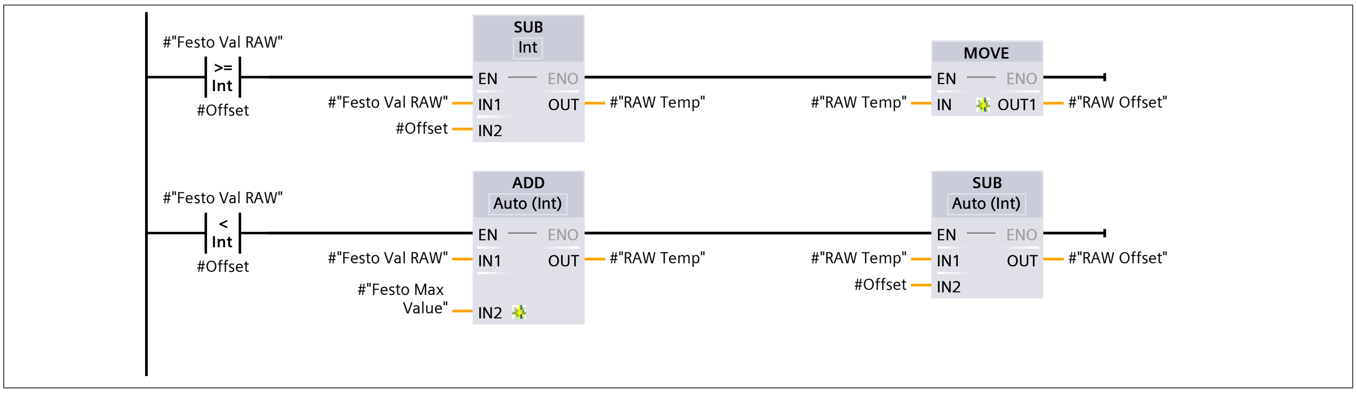
Name	Data type	Default value	Retain
▼ Input			
Festo Val RAW	Int	0	Non-retain
Festo Max Value	Int	27648	Non-retain
Update Offset	Bool	false	Non-retain
▼ Output			
Angle (rad)	Real	0.0	Non-retain
Angle (deg)	Real	0.0	Non-retain
▼ InOut			
Offset	Int	13772	Non-retain
▼ Static			
P_Edge	Bool	false	Non-retain
RAW Offset	Int	0	Non-retain
RAW Temp	Int	0	Non-retain
Festo Val %	Real	0.0	Non-retain
Festo RAW rad	Real	0.0	Non-retain
Festo Value CCW	Real	0.0	Non-retain
Temp			
Constant			

Network 1: Update Analog Offset

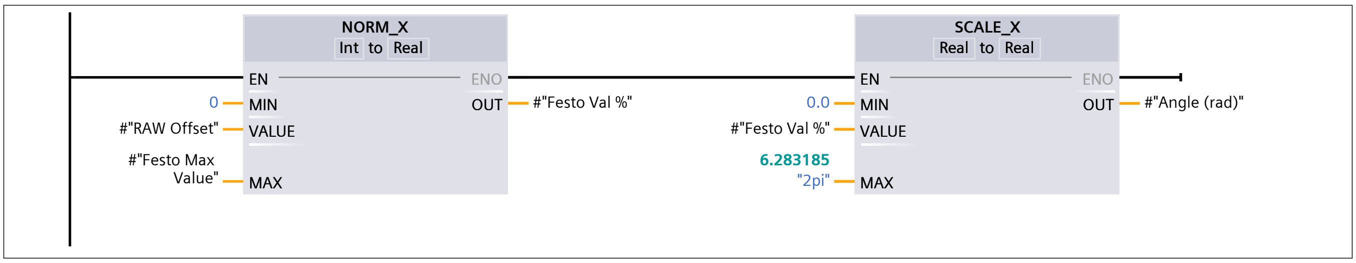


Network 2: Remove Offset from RAW pot value

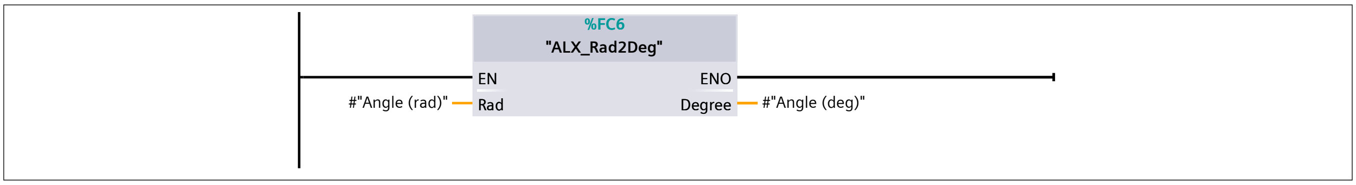
A 5V offset is provided to the analog value thus 5V = 0 deg. This was done to move the 0V -> 10V changeover to the 180 deg position. Since this changeover between voltages causes a volateg ramp that causes the system to erroneously measure the current angle as changing even if it is not moving. 180 degress is full reverse, where it is thought that this instablity will have the least effect on te system.



Network 3: Normalise and scale incoming analog value



Network 4: Convert to degrees



05. Analog Steering Pots / Instance DBs

Pot A Read iDB [DB160]

Pot A Read iDB Properties					
General					
Name	Pot A Read iDB	**Number**	160	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
Pot Val RAW	Int	0	False
Pot Max Value	Real	0.0	False
Update Pot Offset	Bool	false	False
Trigger Differential	Bool	false	False
deltaT	LReal	0.0	False
▼ Output			
Angle (rad)	Real	0.0	False
Angle (deg)	Real	0.0	False
Psuedo Speed (rpm)	Real	0.0	False
▼ InOut			
Pot Offset	Real	0.0	False
▼ Static			
PotValues	Struct		False
Polyline_Instance	Polyline		False
Pot Values	Struct		False
LGF_DifferenceQuotientFB_Instance	"LGF_DifferenceQuotientFB"		False
last cycle CCW	Array[0..4] of Real		False
values	Struct		False
AuxPurgeZero	Int	0	False
LockoutPurge	Bool	false	False

05. Analog Steering Pots / Instance DBs

Pot B Read iDB [DB161]

Pot B Read iDB Properties					
General					
Name	Pot B Read iDB	**Number**	161	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
Pot Val RAW	Int	0	False
Pot Max Value	Real	0.0	False
Update Pot Offset	Bool	false	False
Trigger Differential	Bool	false	False
deltaT	LReal	0.0	False
▼ Output			
Angle (rad)	Real	0.0	False
Angle (deg)	Real	0.0	False
Psuedo Speed (rpm)	Real	0.0	False
▼ InOut			
Pot Offset	Real	0.0	False
▼ Static			
PotValues	Struct		False
Pot Values	Struct		False
LGF_DifferenceQuotientFB_Instance	"LGF_DifferenceQuotientFB"		False
Polyline_Instance	Polyline		False
last cycle CCW	Array[0..4] of Real		False
values	Struct		False
AuxPurgeZero	Int	0	False
LockoutPurge	Bool	false	False

05. Analog Steering Pots / Instance DBs

Festo A Analog Read iDB [DB163]

Festo A Analog Read iDB Properties					
General					
Name	Festo A Analog Read iDB	**Number**	163	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
Festo Val RAW	Int	0	False
Festo Max Value	Int	27648	False
Update Offset	Bool	false	False
▼ Output			
Angle (rad)	Real	0.0	False
Angle (deg)	Real	0.0	False
▼ InOut			
Offset	Int	13904	False
▼ Static			
P_Edge	Bool	false	False
RAW Offset	Int	0	False
RAW temp	Int	0	False
Festo Val %	Real	0.0	False
Festo RAW rad	Real	0.0	False
Festo Value CCW	Real	0.0	False

05. Analog Steering Pots / Instance DBs

Festo B Analog Read iDB [DB164]

Festo B Analog Read iDB Properties					
General					
Name	Festo B Analog Read iDB	**Number**	164	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
Festo Val RAW	Int	0	False
Festo Max Value	Int	27648	False
Update Offset	Bool	false	False
▼ Output			
Angle (rad)	Real	0.0	False
Angle (deg)	Real	0.0	False
▼ InOut			
Offset	Int	13772	False
▼ Static			
P_Edge	Bool	false	False
RAW Offset	Int	0	False
RAW Temp	Int	0	False
Festo Val %	Real	0.0	False
Festo RAW rad	Real	0.0	False
Festo Value CCW	Real	0.0	False

06. Indicator Controls

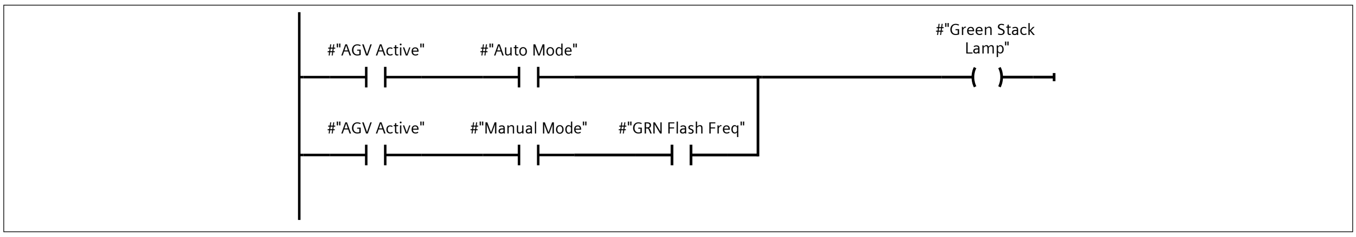
FB Indicator Controls [FB20]

FB Indicator Controls Properties					
General					
Name	FB Indicator Controls	**Number**	20	**Type**	FB
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
▼ Input			
Commisioning Mode	Bool	false	Non-retain
Manual Mode	Bool	false	Non-retain
Auto Mode	Bool	false	Non-retain
Homing Mode	Bool	false	Non-retain
AGV Active	Bool	false	Non-retain
AGV In Motion	Bool	false	Non-retain
STO State	Bool	false	Non-retain
Ack Required	Bool	false	Non-retain
Front Estop State	Bool	false	Non-retain
Rear Estop State	Bool	false	Non-retain
Wheel Pot Error	Bool	false	Non-retain
Reset Pot Offsets	Bool	false	Non-retain
RED Flash Freq	Bool	false	Non-retain
GRN Flash Freq	Bool	false	Non-retain
ACK Flash Freq	Bool	false	Non-retain
Reset Pots Flash Freq	Bool	false	Non-retain
▼ Output			
Green Stack Lamp	Bool	false	Non-retain
Yellow Stack Lamp	Bool	false	Non-retain
Red Stack Lamp	Bool	false	Non-retain
Front Flasher	Bool	false	Non-retain
Pendant AGV ON LED	Bool	false	Non-retain
Pendant Manual LED	Bool	false	Non-retain
Pendant Comission MD	Bool	false	Non-retain
Pendant Auto MD	Bool	false	Non-retain
Pendant Homing MD	Bool	false	Non-retain
Pendant Wheel Pot LED	Bool	false	Non-retain
Pendant Ack Req	Bool	false	Non-retain
InOut			
▼ Static			
Aux Red Lamp 1	Bool	false	Non-retain
Temp			
Constant			

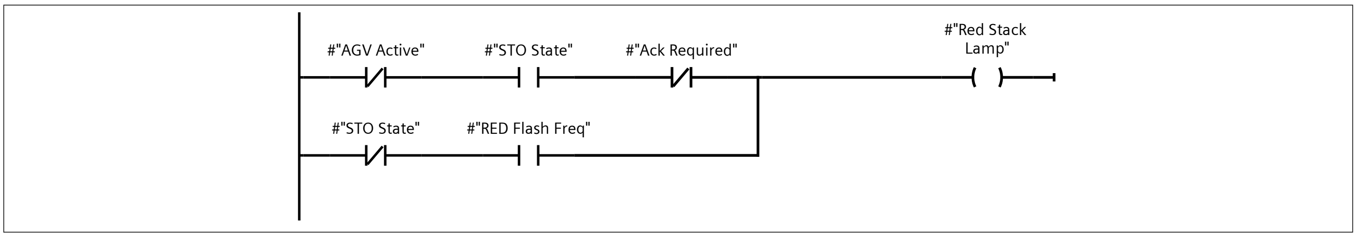
Network 1: Green Stack Lamp Control

GRN STACK STEADY ON = AGV is in run mode & automatic mode
GRN STACK FLASHING = AGV is in run mode & manual mode



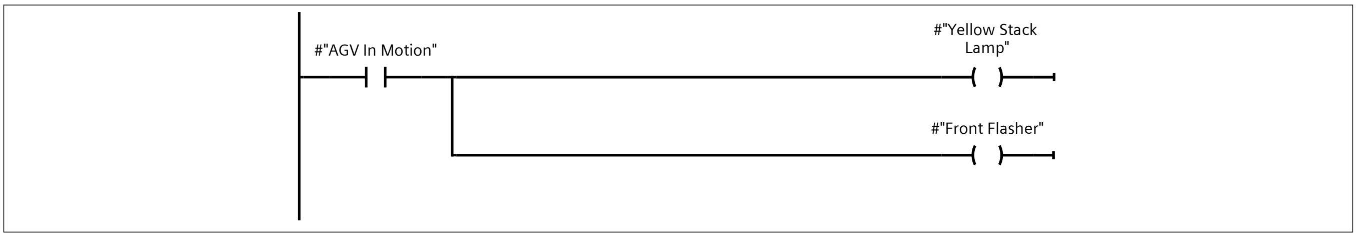
Network 2: Red Stack Lamp Control

STEADY ON = AGV is in stop mode (manual or automatic)
FLASHING = AGV is in STO/Ack required Mode (Estop Mode)



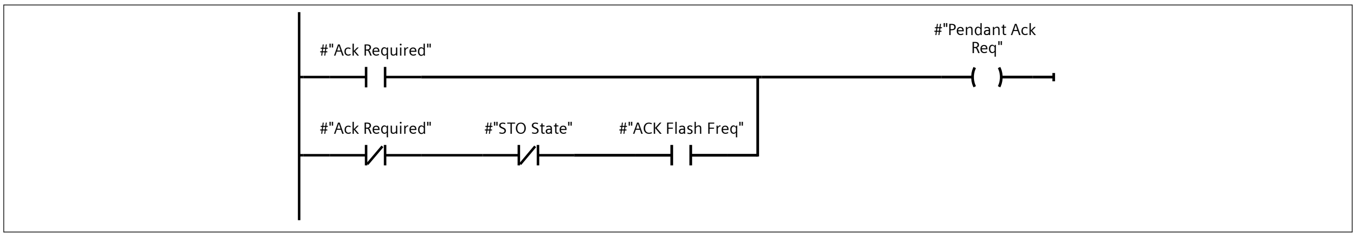
Network 3: Yellow Lamp & Front Flasher Control (AGV Motion Warning)

Both of these lights have physical flashing circuits DO NOT FLASH IN CODE

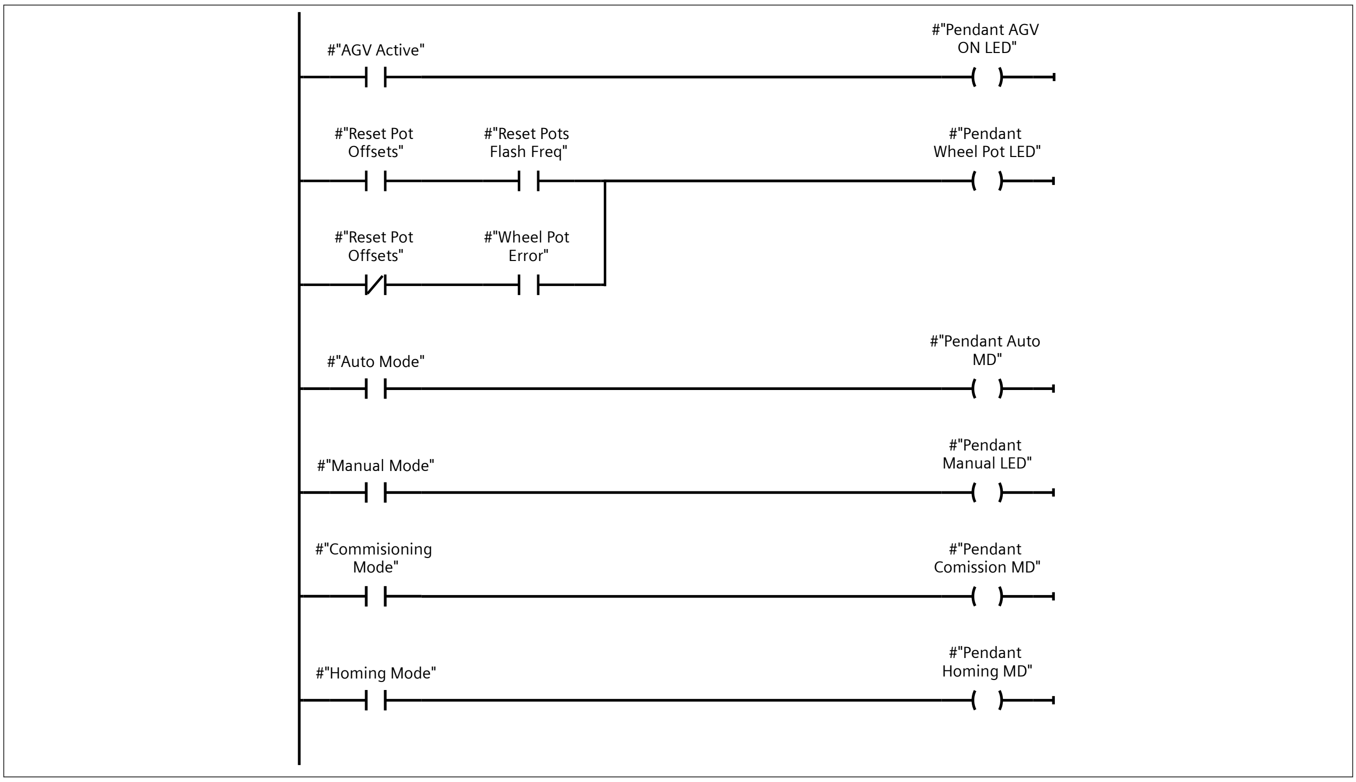


Network 4: Acknowledge Required

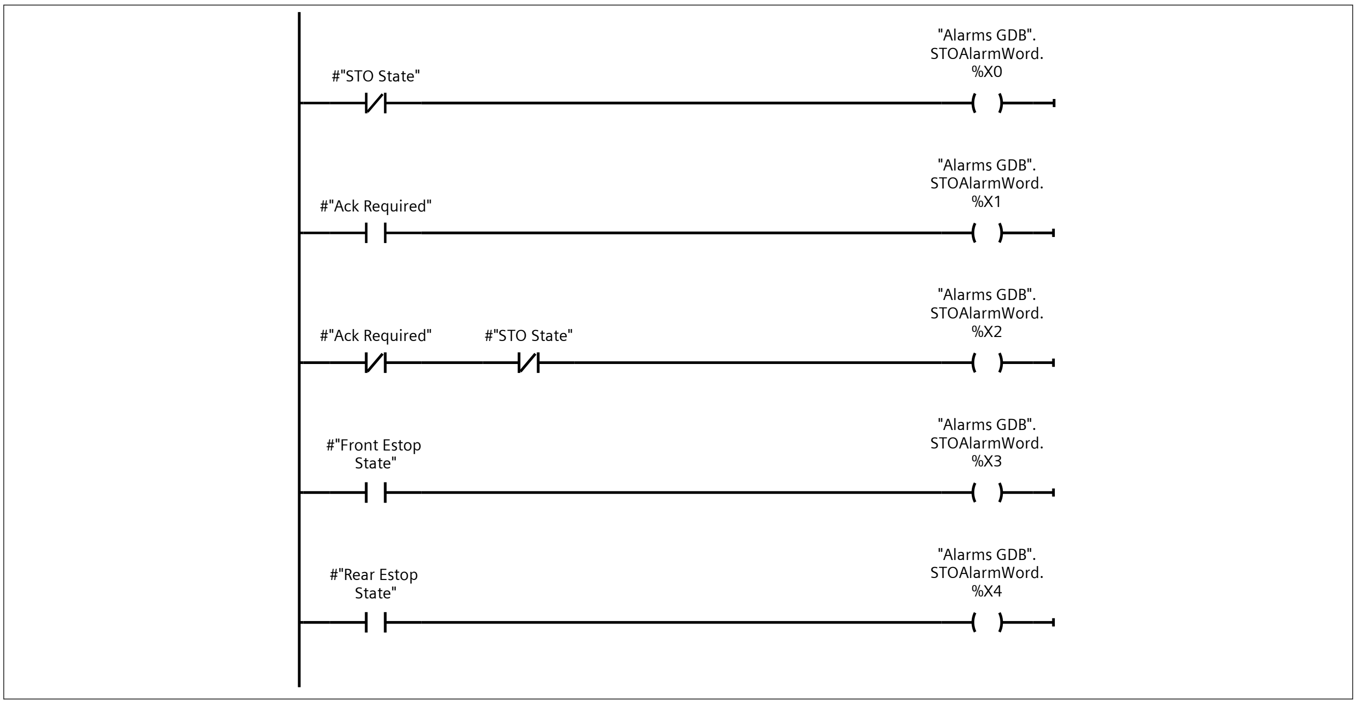
STEADY ON = Acknowledge Required
FLASHING = In STO mode (First release Estop)



Network 5: Pendant Status LEDs



Network 6: Alarm Word Controls


#"STO State"
"Alarms GDB". STOAlarmWord. %X0
#"Ack Required"
"Alarms GDB". STOAlarmWord. %X1
#"Ack Required"
#"STO State"
"Alarms GDB". STOAlarmWord. %X2
#"Front Estop State"
"Alarms GDB". STOAlarmWord. %X3
#"Rear Estop State"
"Alarms GDB". STOAlarmWord. %X4

06. Indicator Controls / Instance DBs

Indicator Controls iDB [DB20]

Indicator Controls iDB Properties					
General					
Name	Indicator Controls iDB	**Number**	20	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
Commisioning Mode	Bool	false	False
Manual Mode	Bool	false	False
Auto Mode	Bool	false	False
Homing Mode	Bool	false	False
AGV Active	Bool	false	False
AGV In Motion	Bool	false	False
STO State	Bool	false	False
Ack Required	Bool	false	False
Front Estop State	Bool	false	False
Rear Estop State	Bool	false	False
Wheel Pot Error	Bool	false	False
Reset Pot Offsets	Bool	false	False
RED Flash Freq	Bool	false	False
GRN Flash Freq	Bool	false	False
ACK Flash Freq	Bool	false	False
Reset Pots Flash Freq	Bool	false	False
▼ Output			
Green Stack Lamp	Bool	false	False
Yellow Stack Lamp	Bool	false	False
Red Stack Lamp	Bool	false	False
Front Flasher	Bool	false	False
Pendant AGV ON LED	Bool	false	False
Pendant Manual LED	Bool	false	False
Pendant Comission MD	Bool	false	False
Pendant Auto MD	Bool	false	False
Pendant Homing MD	Bool	false	False
Pendant Wheel Pot LED	Bool	false	False
Pendant Ack Req	Bool	false	False
InOut			
▼ Static			
Aux Red Lamp 1	Bool	false	False

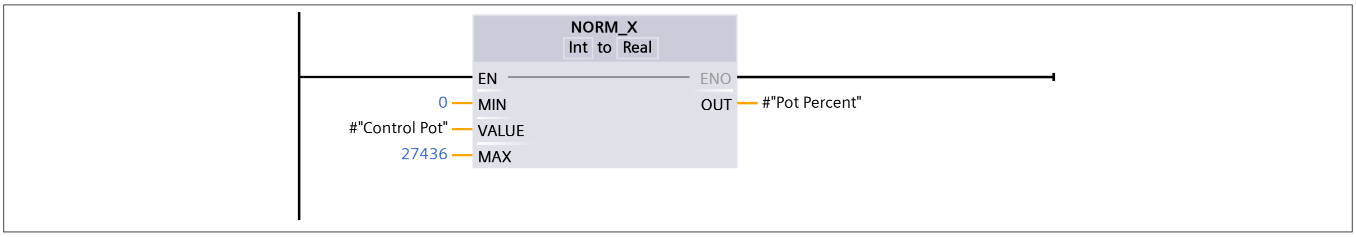
07. Commisioning Mode

FB Commisioning Mode [FB313]

FB Commisioning Mode Properties					
General					
Name	FB Commisioning Mode	**Number**	313	**Type**	FB
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

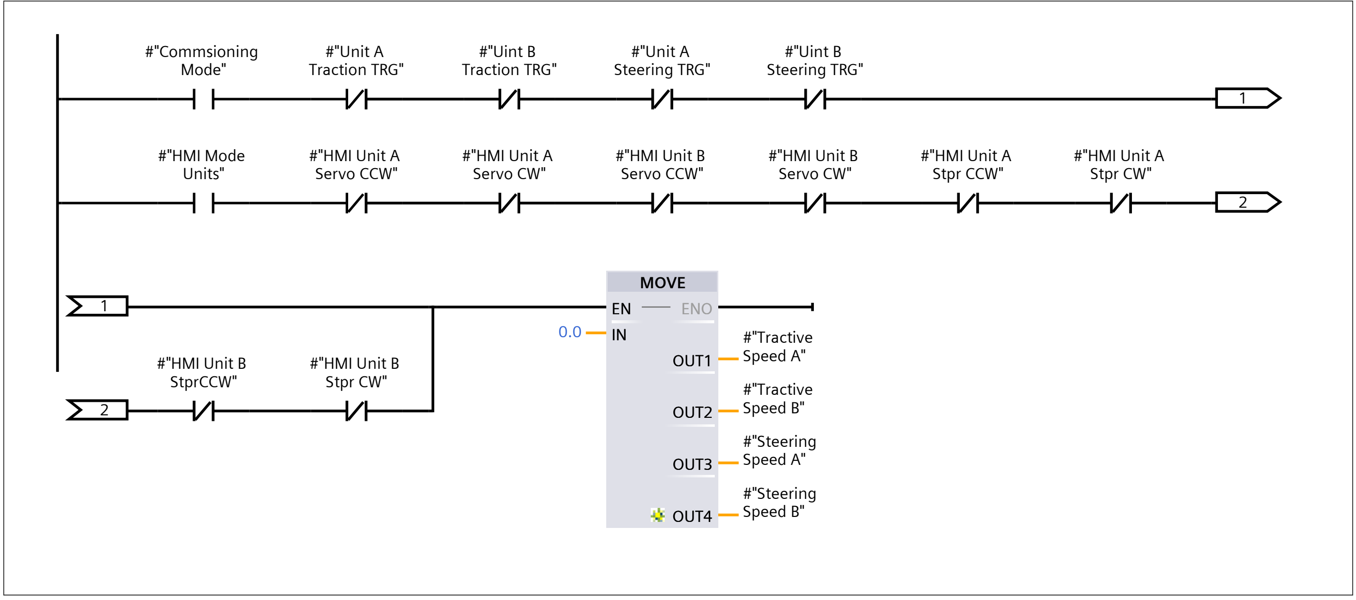
Name	Data type	Default value	Retain
▼ Input			
Commsioning Mode	Bool	false	Non-retain
HMI Mode Units	Bool	false	Non-retain
Control Pot	Int	0	Non-retain
Unit A Traction TRG	Bool	false	Non-retain
Uint B Traction TRG	Bool	false	Non-retain
Unit A Steering TRG	Bool	false	Non-retain
Uint B Steering TRG	Bool	false	Non-retain
HMI Unit A Stpr CCW	Bool	false	Non-retain
HMI Unit A Stpr CW	Bool	false	Non-retain
HMI Unit B StprCCW	Bool	false	Non-retain
HMI Unit B Stpr CW	Bool	false	Non-retain
HMI Unit A Servo CCW	Bool	false	Non-retain
HMI Unit A Servo CW	Bool	false	Non-retain
HMI Unit B Servo CCW	Bool	false	Non-retain
HMI Unit B Servo CW	Bool	false	Non-retain
HMI Setpoint Speed	Real	0.0	Non-retain
▼ Output			
Tractive Speed A	Real	0.0	Non-retain
Tractive Speed B	Real	0.0	Non-retain
Steering Speed A	Real	0.0	Non-retain
Steering Speed B	Real	0.0	Non-retain
InOut			
Static			
▼ Temp			
Pot Percent	Real		
Tractive A RPM	Real		
Tractive B RPM	Real		
Steering A RPM	Real		
Steering B RPM	Real		
▼ Constant			
Tractive Speed LIMITS +	Real	3.0	
Tractive Speed LIMITS -	Real	-3.0	
Steering Speed LIMITS +	Real	3.0	
Steering Speed LIMITS -	Real	-3.0	

Network 1: Normalise Pot Value



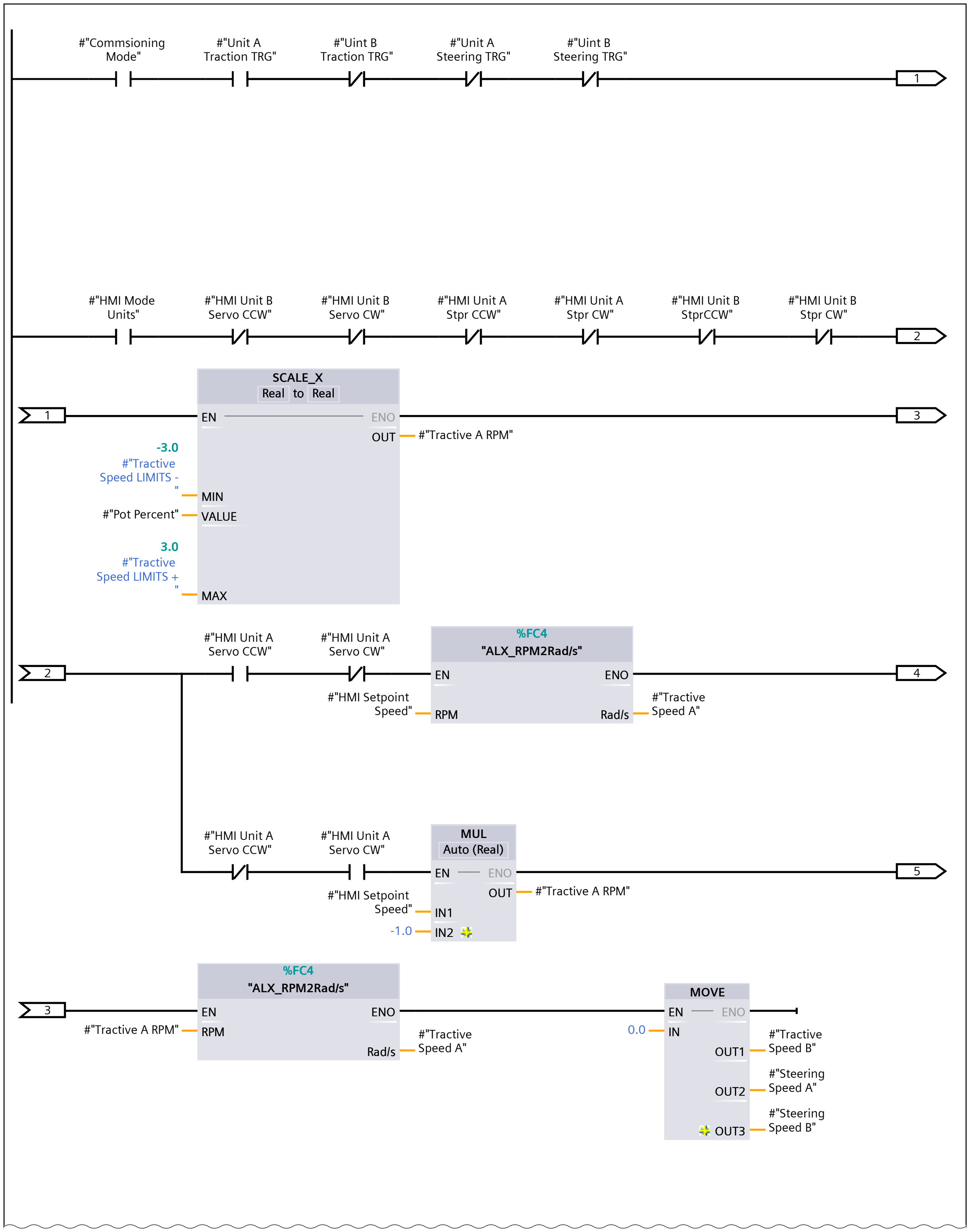
Network 2: In commisiong mode but no motor selected

Network 2: In commisiong mode but no motor selected



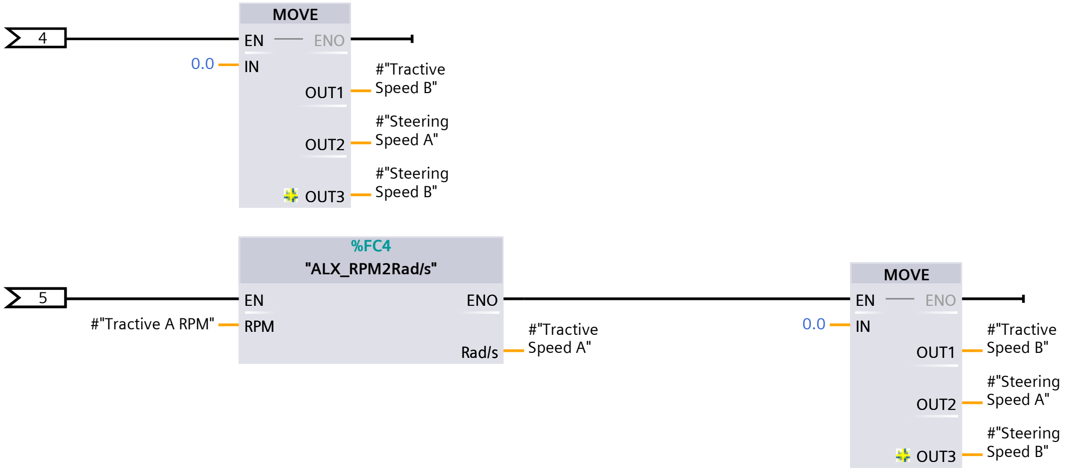
Network 3: Direct control of unit A traction

Network 3: Direct control of unit A traction (1.1 / 2.1)



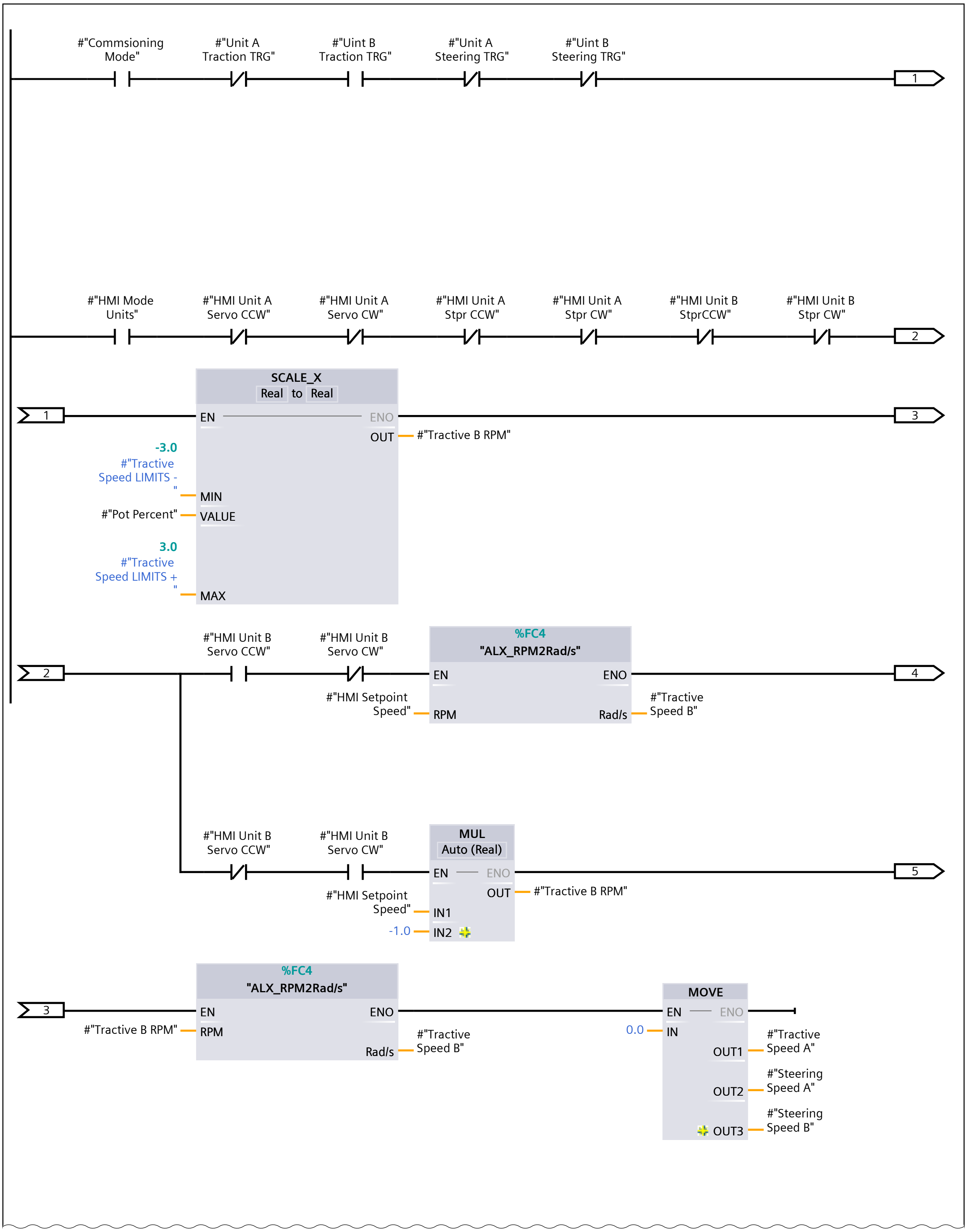
Network 3: Direct control of unit A traction (2.1 / 2.1)

1.1 (Page2 - 3)



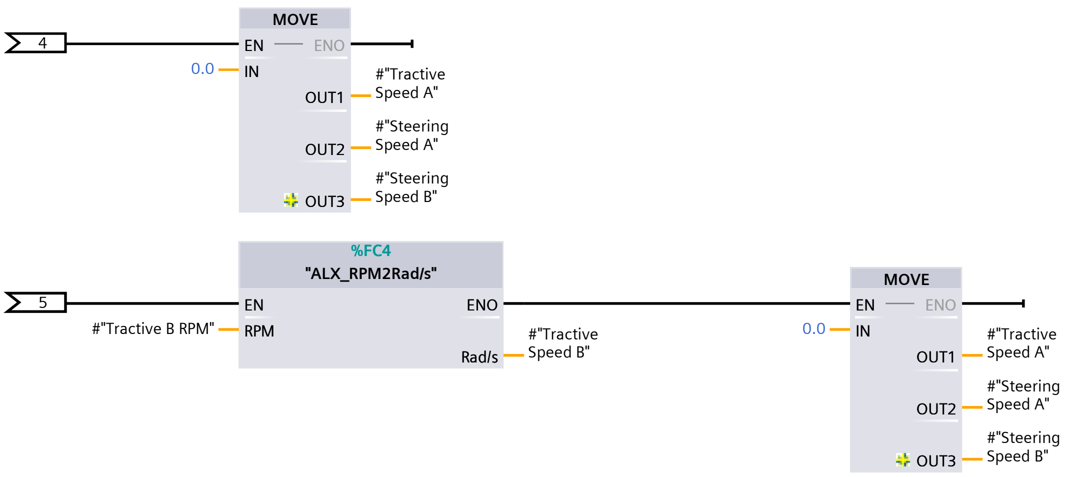
Network 4: Direct control of unit B traction

Network 4: Direct control of unit B traction (1.1 / 2.1)



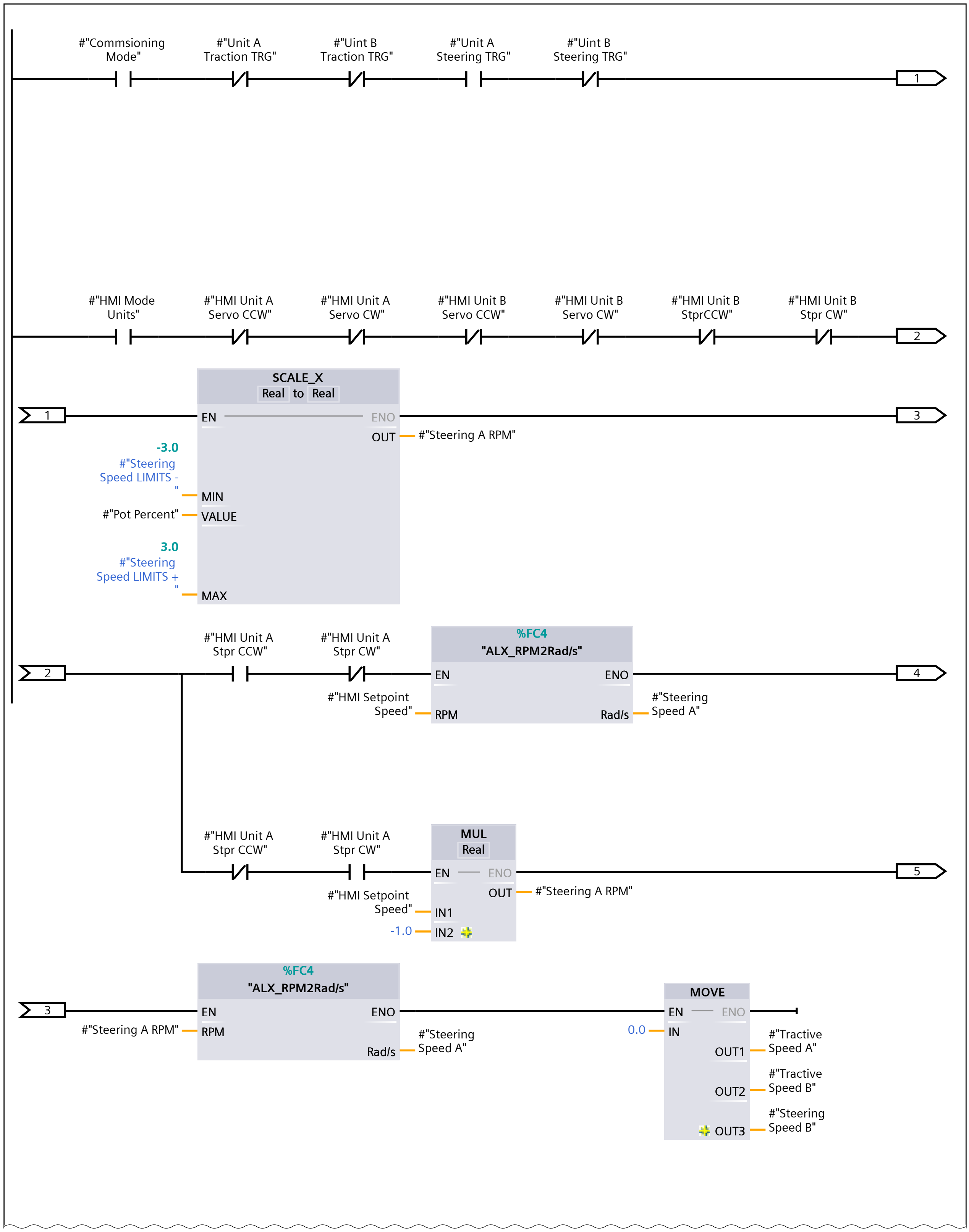
Network 4: Direct control of unit B traction (2.1 / 2.1)

1.1 (Page2 - 6)



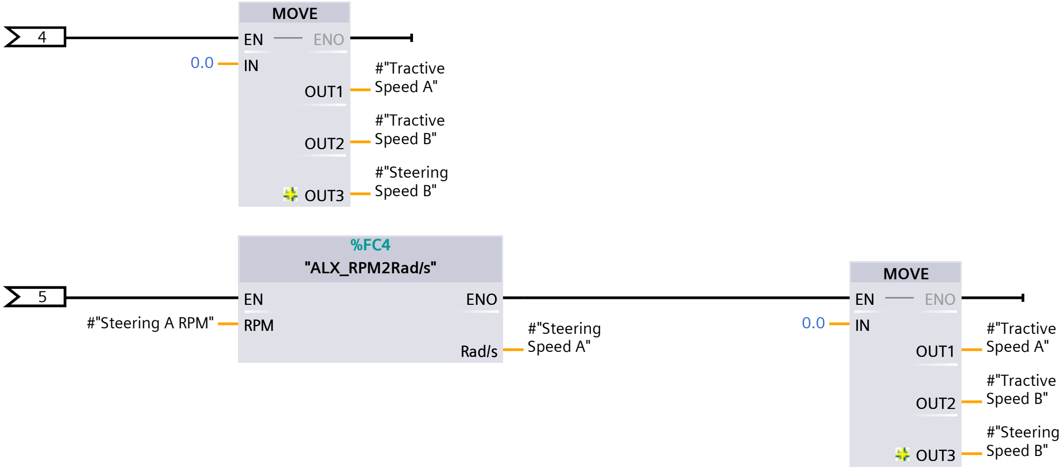
Network 5: Direct control of unit A steering

Network 5: Direct control of unit A steering (1.1 / 2.1)



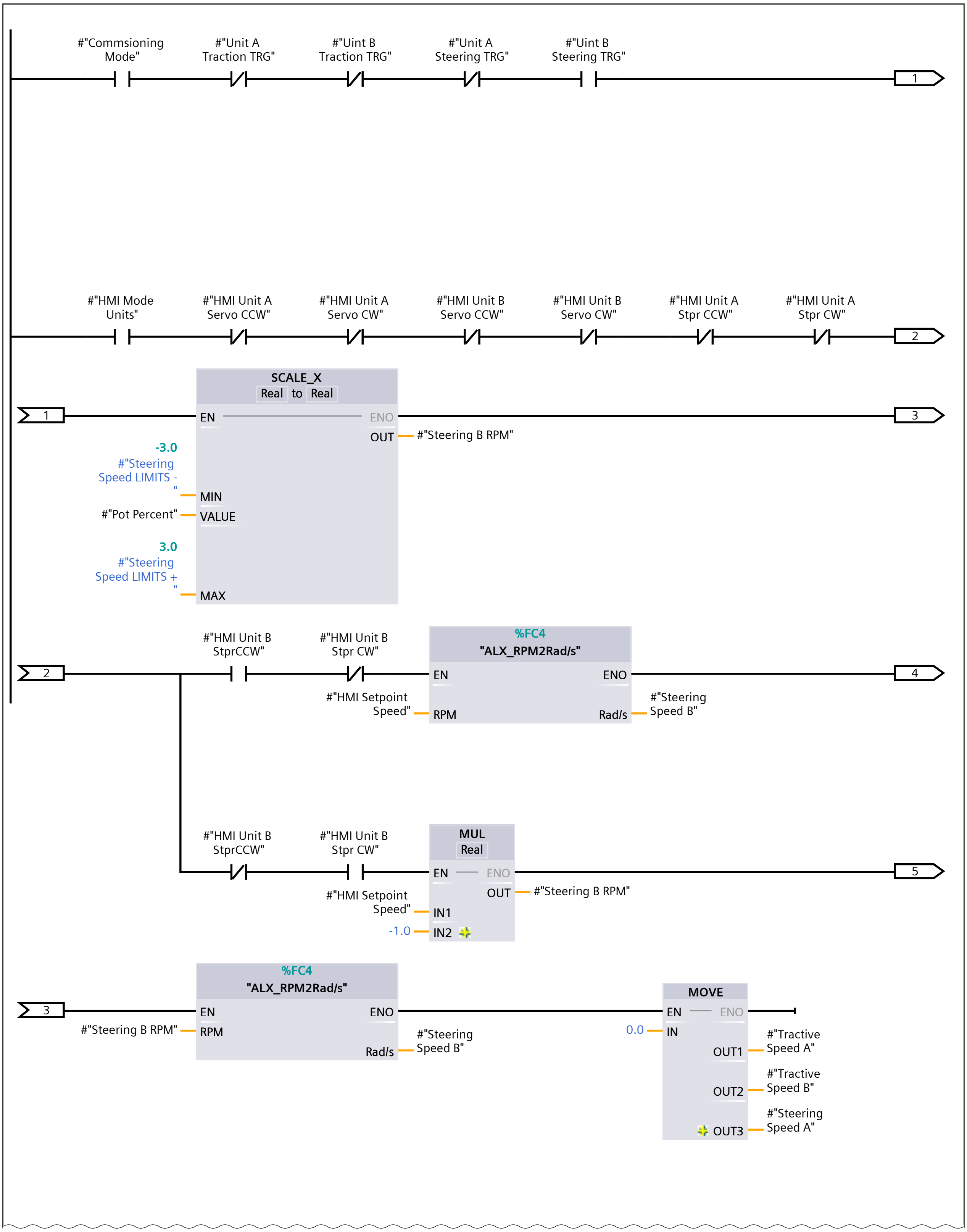
Network 5: Direct control of unit A steering (2.1 / 2.1)

1.1 (Page2 - 9)



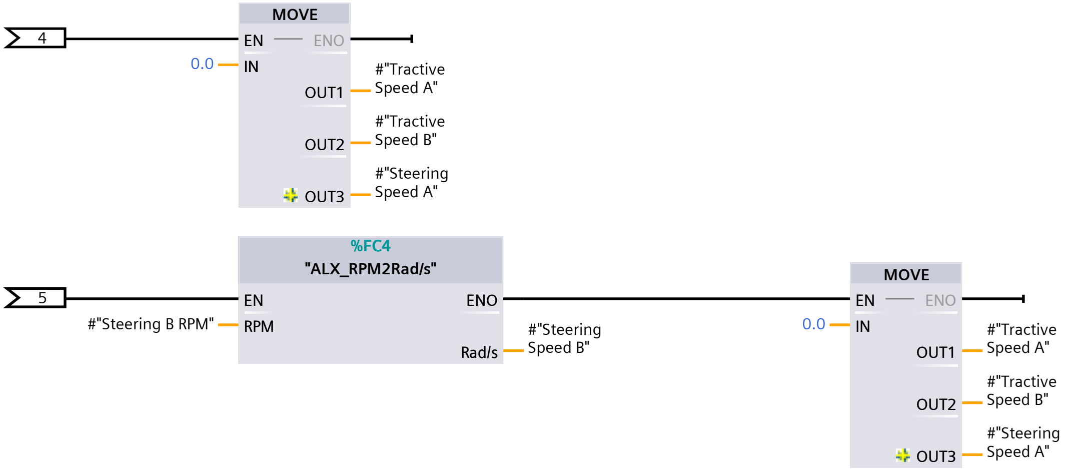
Network 6: Direct control of unit B steering

Network 6: Direct control of unit B steering (1.1 / 2.1)



Network 6: Direct control of unit B steering (2.1 / 2.1)

1.1 (Page2 - 12)



07. Commisioning Mode / Instance DBs

Commisioning Mode iDB [DB313]

Commisioning Mode iDB Properties					
General					
Name	Commisioning Mode iDB	**Number**	313	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
Commsioning Mode	Bool	false	False
HMI Mode Units	Bool	false	False
Control Pot	Int	0	False
Unit A Traction TRG	Bool	false	False
Uint B Traction TRG	Bool	false	False
Unit A Steering TRG	Bool	false	False
Uint B Steering TRG	Bool	false	False
HMI Unit A Stpr CCW	Bool	false	False
HMI Unit A Stpr CW	Bool	false	False
HMI Unit B StprCCW	Bool	false	False
HMI Unit B Stpr CW	Bool	false	False
HMI Unit A Servo CCW	Bool	false	False
HMI Unit A Servo CW	Bool	false	False
HMI Unit B Servo CCW	Bool	false	False
HMI Unit B Servo CW	Bool	false	False
HMI Setpoint Speed	Real	0.0	False
▼ Output			
Tractive Speed A	Real	0.0	False
Tractive Speed B	Real	0.0	False
Steering Speed A	Real	0.0	False
Steering Speed B	Real	0.0	False
InOut			
Static			

08. Manual Mode

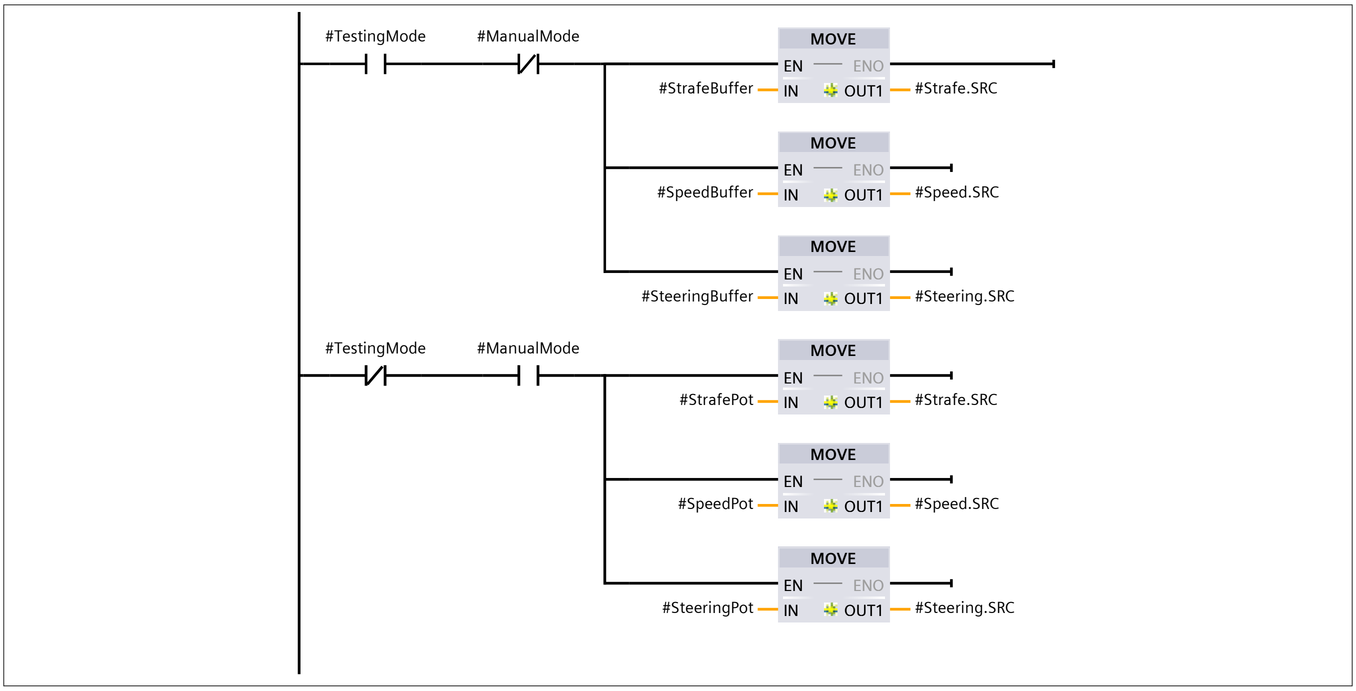
FB Manual Mode [FB318]

FB Manual Mode Properties					
General					
Name	FB Manual Mode	**Number**	318	**Type**	FB
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

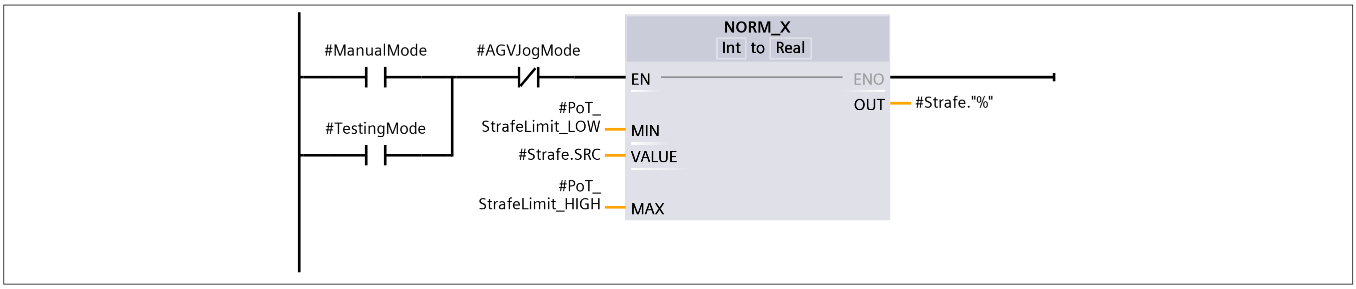
Name	Data type	Default value	Retain
▼ Input			
AGVJogMode	Bool	false	Non-retain
AGVJogMode_Execute	Bool	false	Non-retain
ManualMode	Bool	false	Non-retain
TestingMode	Bool	false	Non-retain
StrafePot	Int	0	Non-retain
StrafeBuffer	Int	0	Non-retain
StrafeHMI	Real	0.0	Non-retain
SteeringPot	Int	0	Non-retain
SteeringBuffer	Int	0	Non-retain
SteeringHMI	Real	0.0	Non-retain
SpeedPot	Int	0	Non-retain
SpeedBuffer	Int	0	Non-retain
SpeedHMI	Real	0.0	Non-retain
PoT_StrafeLimit_HIGH	Int	26606	Non-retain
PoT_StrafeLimit_LOW	Int	54	Non-retain
PoT_SteeringLIMIT_HIGH	Int	27451	Non-retain
PoT_SteeringLIMIT_LOW	Int	98	Non-retain
PoT_SpeedLIMIT_HIGH	Int	27437	Non-retain
PoT_SpeedLIMIT_LOW	Int	68	Non-retain
MaxAGVLinearSpeed	Real	0.05	Non-retain
▼ Output			
BodyVelocity_X	Real	0.0	Non-retain
BodyVelocity_Y	Real	0.0	Non-retain
BodyYawRate	Real	0.0	Non-retain
InOut			
▼ Static			
Strafe	Struct		Non-retain
Speed	Struct		Non-retain
Steering	Struct		Non-retain
LGF_LimRateOfChangeBasic_Strafe	"LGF_LimRateOfChangeBasic"		
LGF_LimRateOfChangeBasic_Steering	"LGF_LimRateOfChangeBasic"		
LGF_LimRateOfChangeBasic_Speed	"LGF_LimRateOfChangeBasic"		

Name	Data type	Default value	Retain
Temp			
▼ Constant			
StrafePotOffset	Real	0.0	
InvertStrafe	Bool	TRUE	
InvertSteering	Bool	TRUE	
SpeedNoiseSupression	Int	150	
DeadzoneUpper	Real	0.55	
DeadzoneLower	Real	0.45	

Network 1: Choose Interger Source (Pot or Buffer)

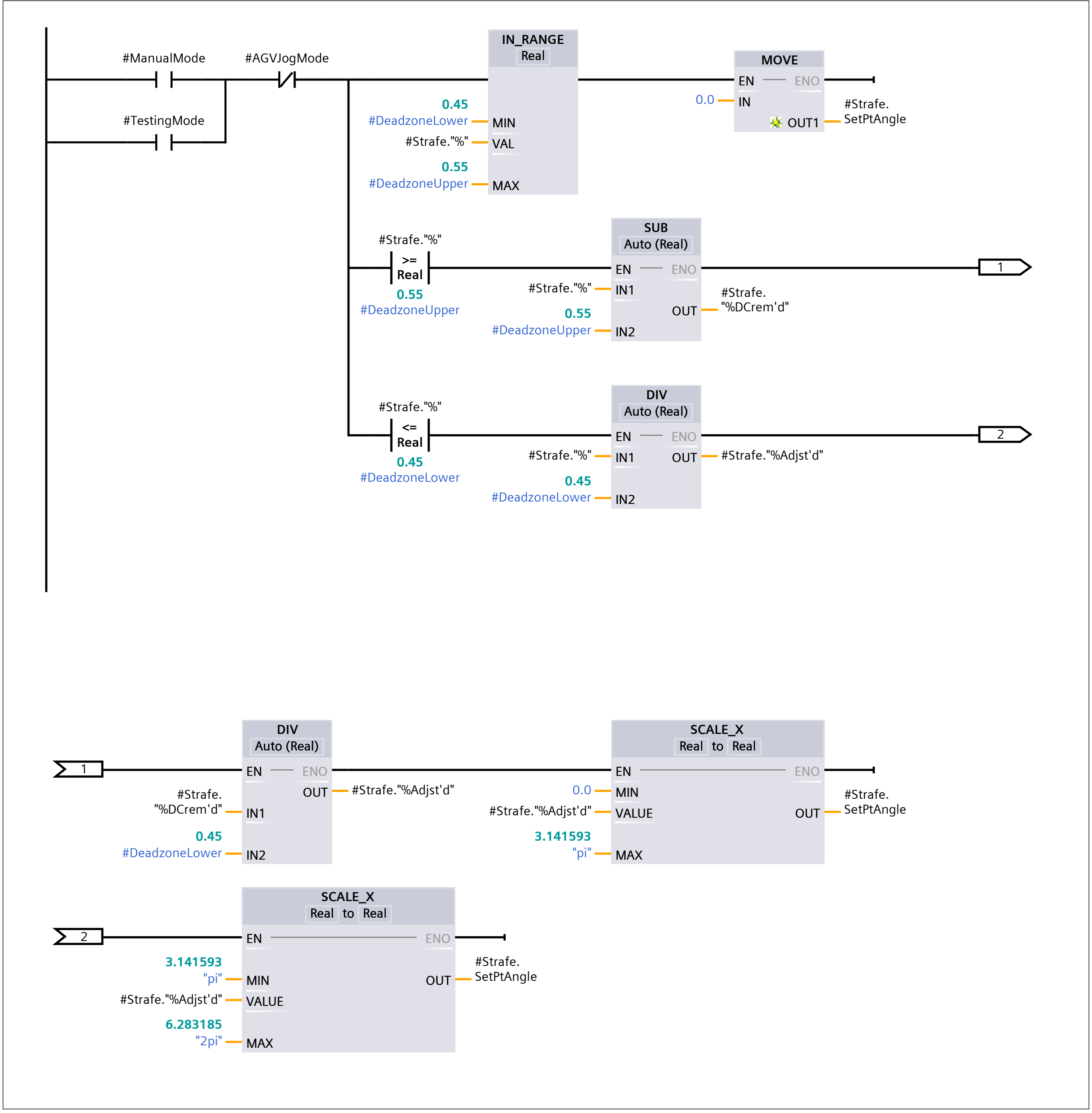


Network 2: Strafe Pot Processing

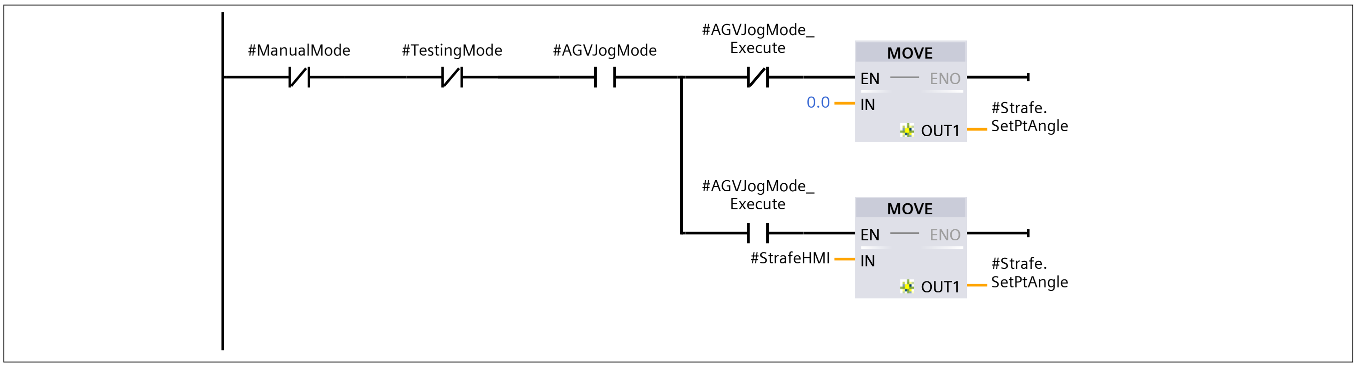


Network 3: Strafe Value Processing

Network 3: Strafe Value Processing

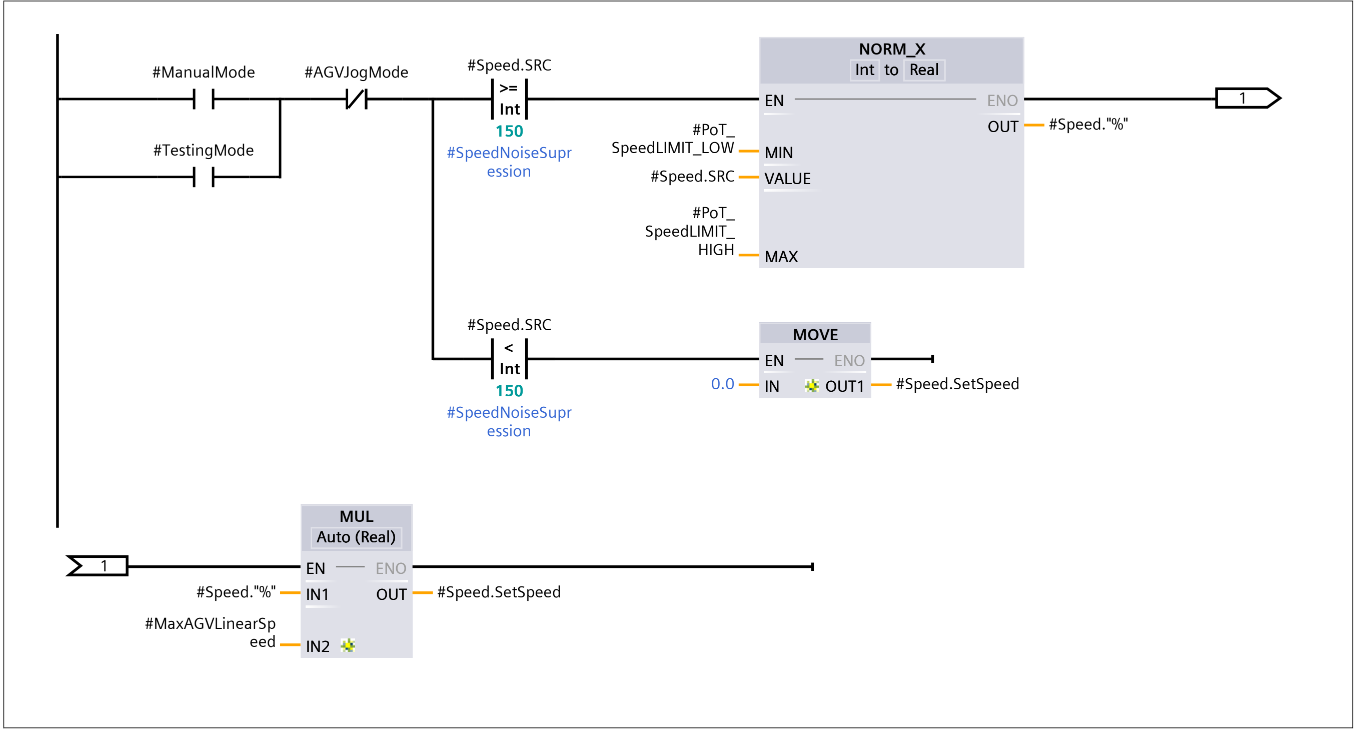


Network 4: Strafe HMI Processing

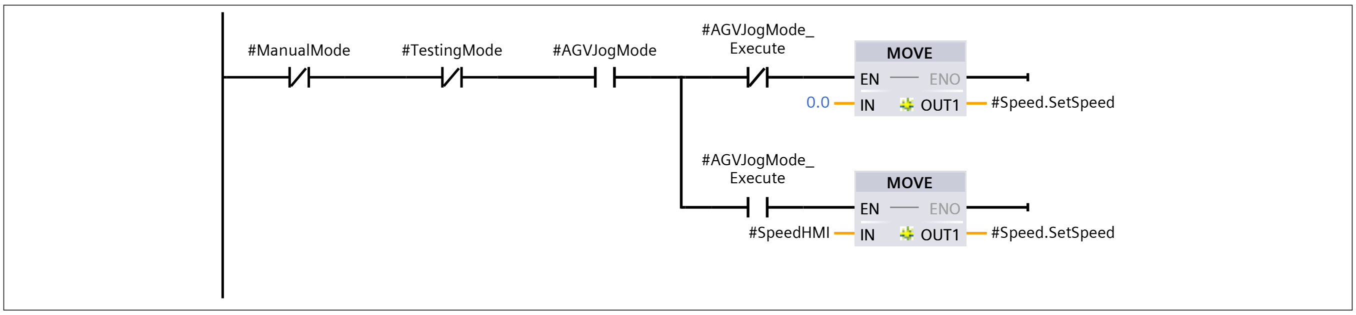


Network 5: Speed Pot Processing

Network 5: Speed Pot Processing



Network 6: HMI Speed Processing

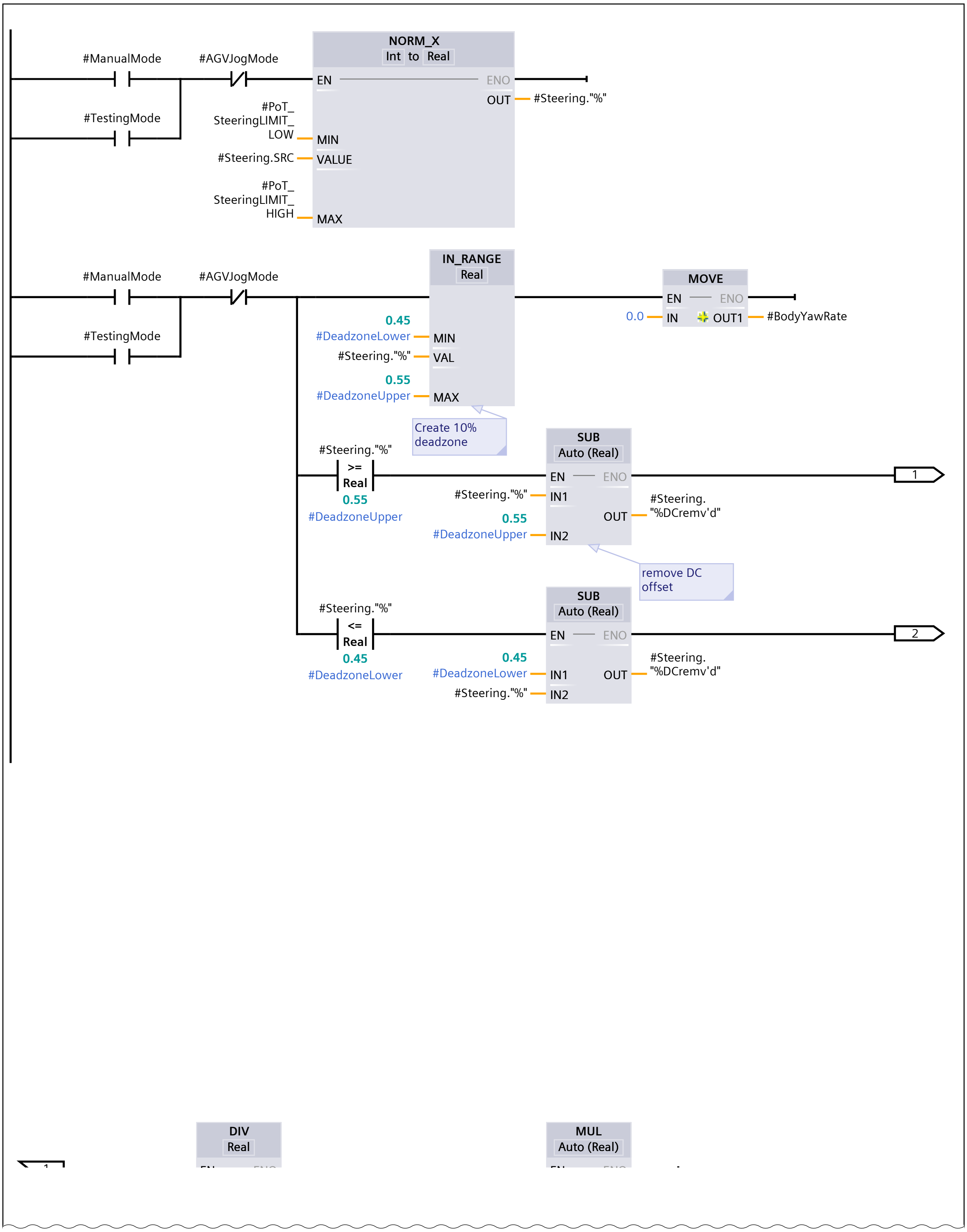


Network 7: Find X and Y Centrodial Velocity Components

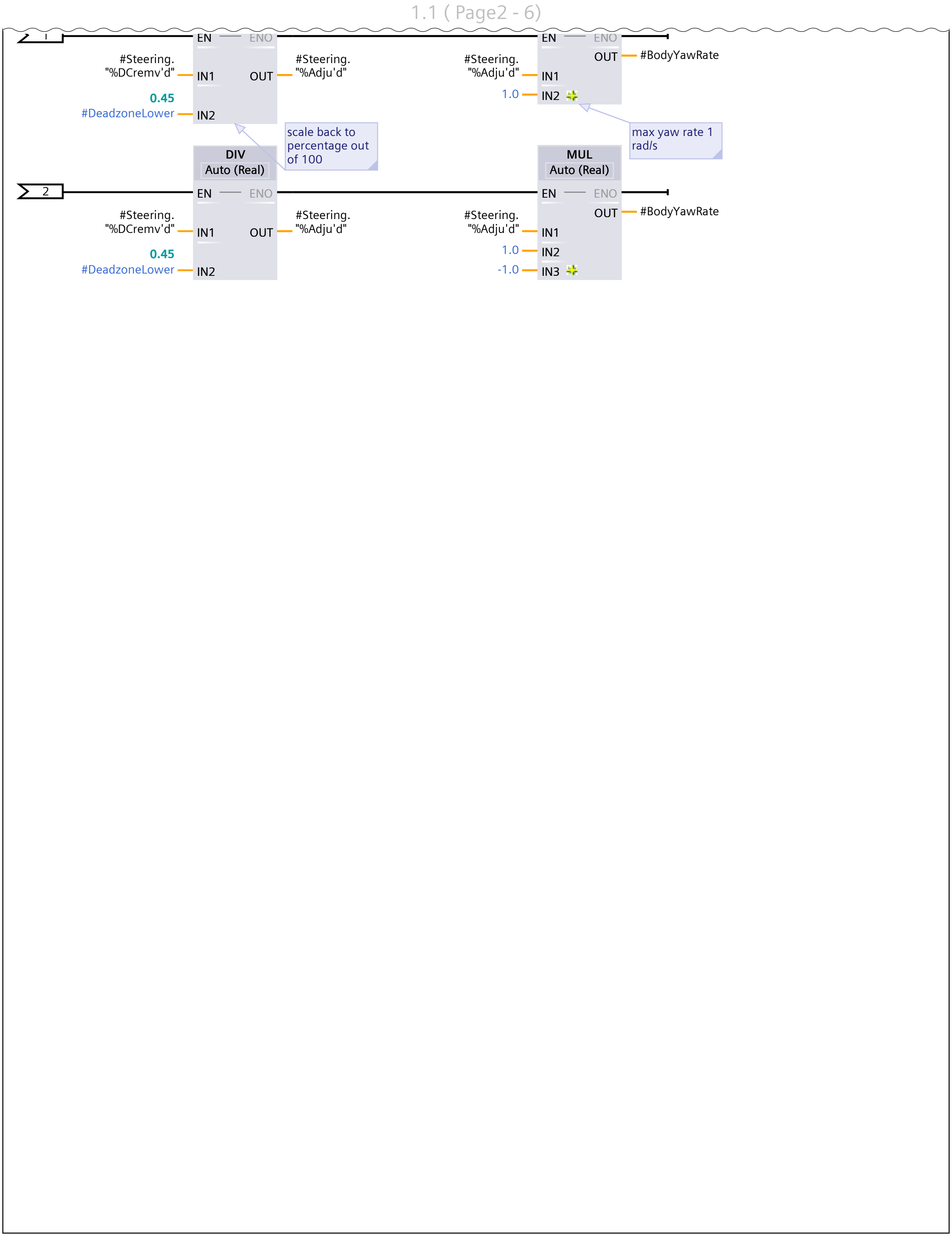
```
// Centroidal X component velocity
//
#BodyVelocity_X := #Speed.SetSpeed * COS(#Strafe.SetPtAngle);

// Centroidal Y Component Velocity
//
#BodyVelocity_Y := #Speed.SetSpeed * SIN(#Strafe.SetPtAngle);
```

Network 8: Steering Pot Processing

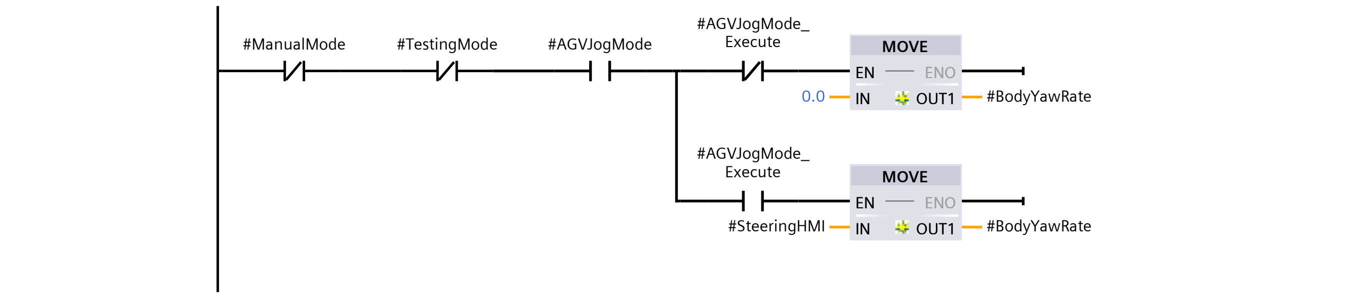
Network 8: Steering Pot Processing (1.1 / 2.1)



Network 8: Steering Pot Processing (2.1 / 2.1)



Network 9: HMI Steering Value Processing



08. Manual Mode / Instance DBs

Manual Mode iDB [DB318]

Manual Mode iDB Properties					
General					
Name	Manual Mode iDB	**Number**	318	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
AGVJogMode	Bool	false	False
AGVJogMode_Execute	Bool	false	False
ManualMode	Bool	false	False
TestingMode	Bool	false	False
StrafePot	Int	0	False
StrafeBuffer	Int	0	False
StrafeHMI	Real	0.0	False
SteeringPot	Int	0	False
SteeringBuffer	Int	0	False
SteeringHMI	Real	0.0	False
SpeedPot	Int	0	False
SpeedBuffer	Int	0	False
SpeedHMI	Real	0.0	False
PoT_StrafeLimit_HIGH	Int	26606	False
PoT_StrafeLimit_LOW	Int	54	False
PoT_SteeringLIMIT_HIGH	Int	27451	False
PoT_SteeringLIMIT_LOW	Int	98	False
PoT_SpeedLIMIT_HIGH	Int	27437	False
PoT_SpeedLIMIT_LOW	Int	68	False
MaxAGVLinearSpeed	Real	0.05	False
▼ Output			
BodyVelocity_X	Real	0.0	False
BodyVelocity_Y	Real	0.0	False
BodyYawRate	Real	0.0	False
InOut			
▼ Static			
Strafe	Struct		False
Speed	Struct		False
Steering	Struct		False
LGF_LimRateOfChangeBasic_Strafe	"LGF_LimRateOfChangeBasic"		False
LGF_LimRateOfChangeBasic_Steering	"LGF_LimRateOfChangeBasic"		False

Name	Data type	Start value	Retain
LGF_LimRateOfChangeBasic_Speed	"LGF_LimRateOfChangeBasic"		False

09. Forward Kinematics

FB Simplified Forward Kinematics [FB316]

FB Simplified Forward Kinematics Properties					
General					
Name	FB Simplified Forward Kinematics	**Number**	316	**Type**	FB
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
▼ Input			
Vv_x	Real	0.0	Non-retain
Vv_y	Real	0.0	Non-retain
Omega_v	Real	0.0	Non-retain
ThetaA	Real	0.0	Non-retain
ThetaB	Real	0.0	Non-retain
rA_x	Real	0.0	Non-retain
rB_x	Real	0.0	Non-retain
rA_y	Real	0.0	Non-retain
rB_y	Real	0.0	Non-retain
WheelRadius	Real	0.0	Non-retain
CasterOffset	Real	0.0	Non-retain
▼ Output			
Traction A Speed	Real	0.0	Non-retain
Traction B Speed	Real	0.0	Non-retain
Steering A Speed	Real	0.0	Non-retain
Steering B Speed	Real	0.0	Non-retain
Steering A Angle RAD	Real	0.0	Non-retain
Steering B Angle RAD	Real	0.0	Non-retain
Steering A Angle DEG	Real	0.0	Non-retain
Steering B Angle DEG	Real	0.0	Non-retain
InOut			
▼ Static			
VA_x	Real	0.0	Non-retain
VA_y	Real	0.0	Non-retain
VB_x	Real	0.0	Non-retain
VB_y	Real	0.0	Non-retain
ThetaACal	Real	0.0	Non-retain
ThetaBCal	Real	0.0	Non-retain
VA	Real	0.0	Non-retain
VB	Real	0.0	Non-retain
LastCPU_VA_x	Real	0.0	Non-retain
LastCPU_VA_y	Real	0.0	Non-retain
LastCPU_VB_x	Real	0.0	Non-retain

Name	Data type	Default value	Retain
LastCPU_VB_y	Real	0.0	Non-retain
BufferdTheta_A	Real	0.0	Non-retain
BufferdTheta_B	Real	0.0	Non-retain
Temp			
▼ Constant			
JitterThreshold	Real	0.0	

Network 1: Unit A find X and Y component velocities

```
// Component velocity of unit A
//
    // X componet velocity
    #VA_x := #Vv_x + (#Omega_v * #rA_y);

    // Y component velocity
    #VA_y := #Vv_y + (#Omega_v * #rA_x);
```

Network 2: Unit B X and Y component velocities

```
// Component velocity of unit B
//
    // X component velocity
    #VB_x := #Vv_x - (#Omega_v * #rB_y);

    // Y component velocity
    #VB_y := #Vv_y - (#Omega_v * #rB_x);
```

Network 3: Unit A Steering & Wheel Velocities

```
//Rotational velocities unit A
//
    //Traction rotational velocity unit A
    #"Traction A Speed" := ((1 / #WheelRadius) * COS(#ThetaA) * #VA_x) + ((1 / #WheelRadius) * SIN(#ThetaA) * #VA_y);

    //Steering roatational velocity unit A
    #"Steering A Speed" := ((-1 / #CasterOffset) * SIN(#ThetaA) * #VA_x) + ((1 / #CasterOffset) * COS(#ThetaA) * #VA_y);

```

Network 4: Unit B Steering & Wheel Velocities

```
//Rotational velocities unit B
//
    //Traction rotational velocity unit B
    #"Traction B Speed" := ((1 / #WheelRadius) * COS(#ThetaB) * #VB_x) + ((1 / #WheelRadius) * SIN(#ThetaB) * #VB_y);

    //Steering rotational velocity unit B
    #"Steering B Speed" := ((-1 / #CasterOffset) * SIN(#ThetaB) * #VB_x) + ((1 / #CasterOffset) * COS(#ThetaB) * #VB_y);
```

09. Forward Kinematics / Instance DBs

Simplified Forward Kinematics iDB [DB316]

Simplified Forward Kinematics iDB Properties					
General					
Name	Simplified Forward Kinematics iDB	**Number**	316	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
Vv_x	Real	0.0	False
Vv_y	Real	0.0	False
Omega_v	Real	0.0	False
ThetaA	Real	0.0	False
ThetaB	Real	0.0	False
rA_x	Real	0.0	False
rB_x	Real	0.0	False
rA_y	Real	0.0	False
rB_y	Real	0.0	False
WheelRadius	Real	0.0	False
CasterOffset	Real	0.0	False
▼ Output			
Traction A Speed	Real	0.0	False
Traction B Speed	Real	0.0	False
Steering A Speed	Real	0.0	False
Steering B Speed	Real	0.0	False
Steering A Angle RAD	Real	0.0	False
Steering B Angle RAD	Real	0.0	False
Steering A Angle DEG	Real	0.0	False
Steering B Angle DEG	Real	0.0	False
InOut			
▼ Static			
VA_x	Real	0.0	False
VA_y	Real	0.0	False
VB_x	Real	0.0	False
VB_y	Real	0.0	False
ThetaACal	Real	0.0	False
ThetaBCal	Real	0.0	False
VA	Real	0.0	False
VB	Real	0.0	False
LastCPU_VA_x	Real	0.0	False
LastCPU_VA_y	Real	0.0	False
LastCPU_VB_x	Real	0.0	False

Name	Data type	Start value	Retain
LastCPU_VB_y	Real	0.0	False
BufferdTheta_A	Real	0.0	False
BufferdTheta_B	Real	0.0	False

10. Plant (Motor Control)

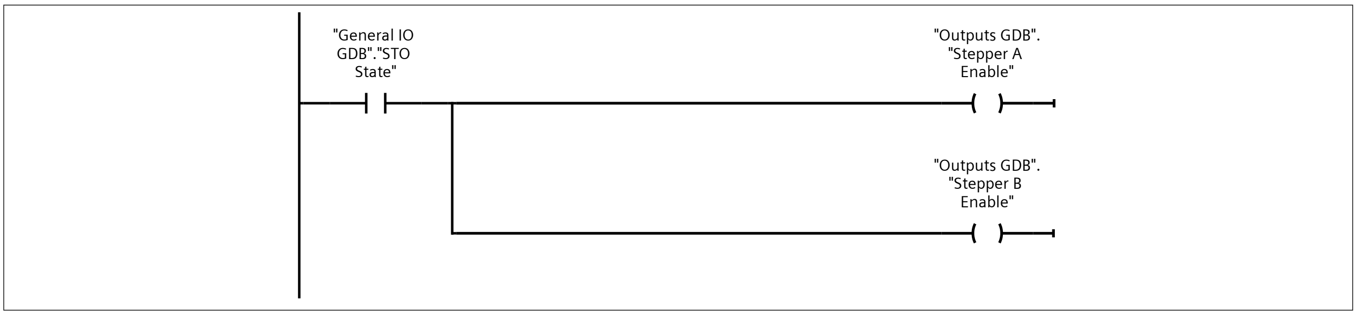
FB Main Motor Control [FB309]

FB Main Motor Control Properties					
General					
Name	FB Main Motor Control	**Number**	309	**Type**	FB
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

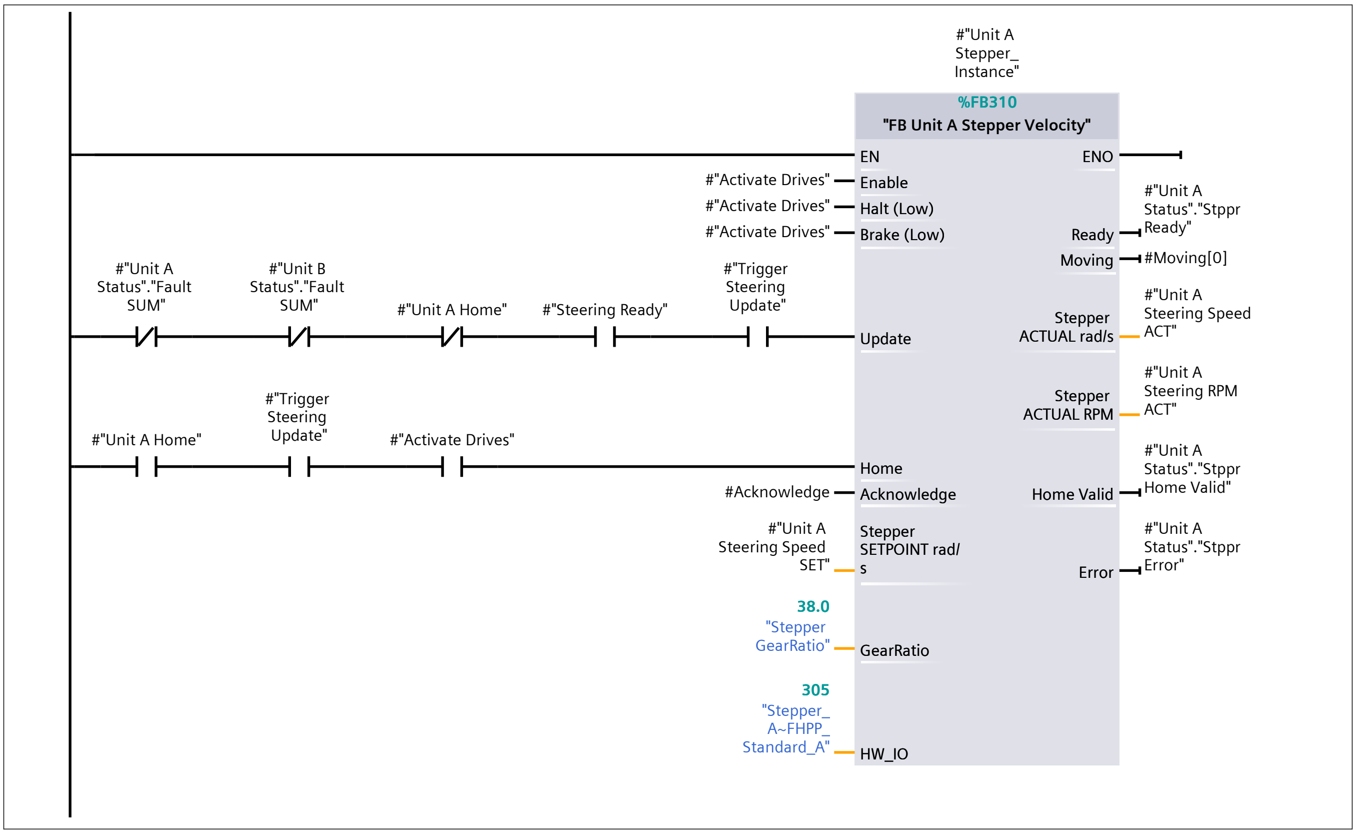
Name	Data type	Default value	Retain
▼ Input			
Activate Drives	Bool	false	Non-retain
Trigger Steering Update	Bool	false	Non-retain
Trigger Traction Update	Bool	false	Non-retain
Unit A Steering Speed SET	Real	0.0	Non-retain
Unit A Tractive Speed SET	Real	0.0	Non-retain
Unit B Steering Speed SET	Real	0.0	Non-retain
Unit B Tractive Speed SET	Real	0.0	Non-retain
Unit A Home	Bool	false	Non-retain
Unit B Home	Bool	false	Non-retain
Acknowledge	Bool	false	Non-retain
▼ Output			
Steering Ready	Bool	false	Non-retain
Unit A Steering Fault	Bool	false	Non-retain
Unit A Traction Fault	Bool	false	Non-retain
Unit A Homing Req	Bool	false	Non-retain
Unit A Steering Speed ACT	Real	0.0	Non-retain
Unit A Steering RPM ACT	Real	0.0	Non-retain
Unit A Traction Speed ACT	Real	0.0	Non-retain
Unit A Traction RPM ACT	Real	0.0	Non-retain
Unit B Steering Fault	Bool	false	Non-retain
Unit B Traction Fault	Bool	false	Non-retain
Unit B Homig Req	Bool	false	Non-retain
Unit B Steering Speed ACT	Real	0.0	Non-retain
Unit B Steering RPM ACT	Real	0.0	Non-retain
Unit B Traction Speed ACT	Real	0.0	Non-retain
Unit B Traction RPM ACT	Real	0.0	Non-retain
Moving	Array[0..3] of Bool		Non-retain
AGVMotionState	Bool	false	Non-retain
InOut			
▼ Static			
Unit A Stepper_Instance	"FB Unit A Stepper Velocity"		
Unit B Stepper_Instance	"FB Unit B Stepper Velocity"		
Unit A Status	Struct		Non-retain

Name	Data type	Default value	Retain
Unit B Status	Struct		Non-retain
Unit A Servo_Instance	"FB Unit A Servo"		
Unit B Servo_Instance	"FB Unit B Servo"		
Temp			
Constant			

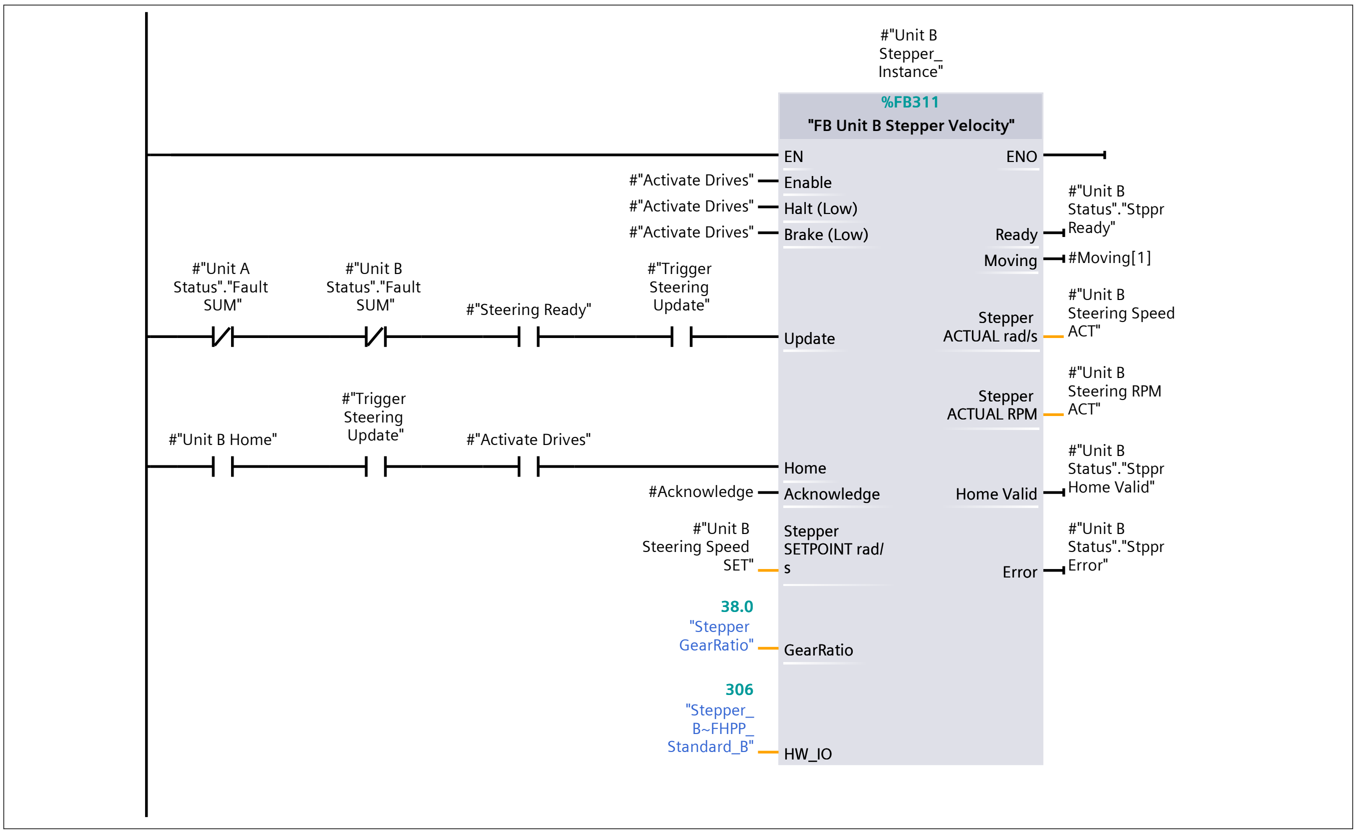
Network 1: Enable Steppers


"General IO GDB"."STO State"
"Outputs GDB". "Stepper A Enable"
"Outputs GDB". "Stepper B Enable"

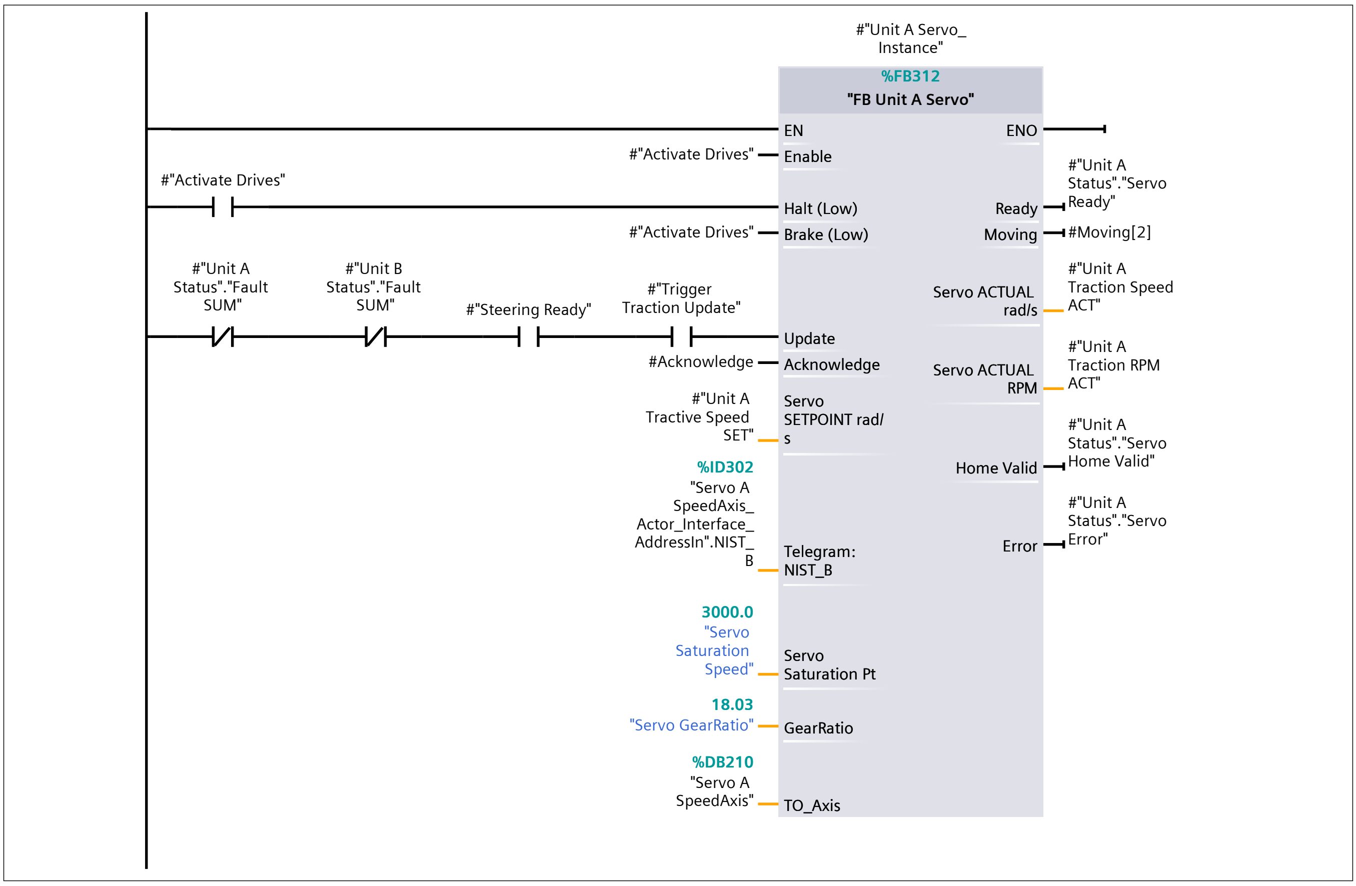
Network 2: Stepper A Control


#"Unit A Stepper_ Instance"
%FB310
"FB Unit A Stepper Velocity"
EN
ENO
#"Activate Drives"
Enable
#"Activate Drives"
Halt (Low)
#"Activate Drives"
Brake (Low)
Ready
#"Unit A Status"."Stppr Ready"
Moving
#Moving[0]
#"Unit A Status"."Fault SUM"
#"Unit B Status"."Fault SUM"
#"Unit A Home"
#"Steering Ready"
#"Trigger Steering Update"
Update
Stepper ACTUAL rad/s
#"Unit A Steering Speed ACT"
Stepper ACTUAL RPM
#"Unit A Steering RPM ACT"
#"Unit A Home"
#"Trigger Steering Update"
#"Activate Drives"
Home
#Acknowledge
Acknowledge
Home Valid
#"Unit A Status"."Stppr Home Valid"
#"Unit A Steering Speed SET"
Stepper SETPOINT rad/ s
Error
#"Unit A Status"."Stppr Error"
38.0
"Stepper GearRatio"
GearRatio
305
"Stepper_ A~FHPP_ Standard_A"
HW_IO

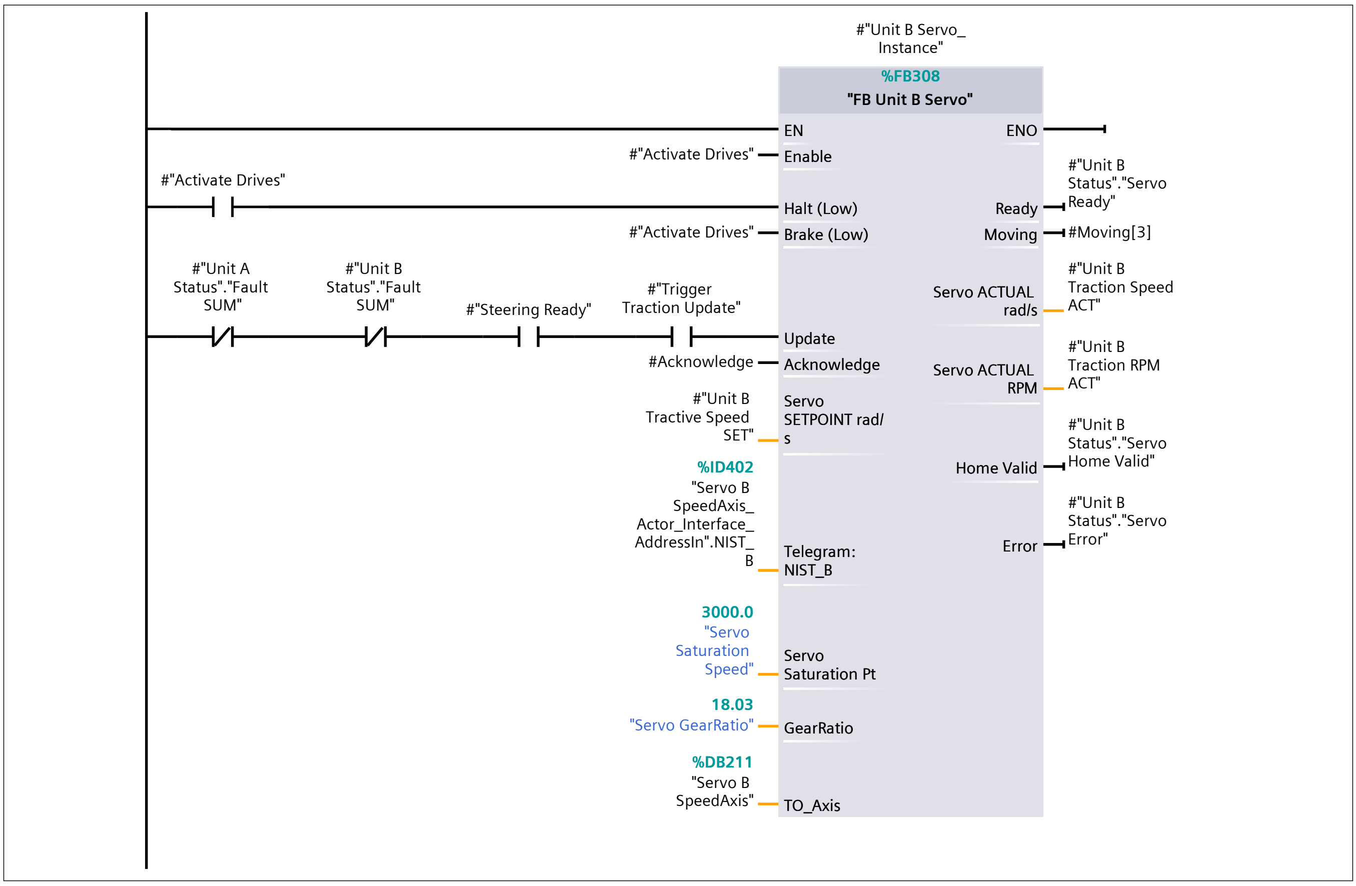
Network 3: Stepper B Control



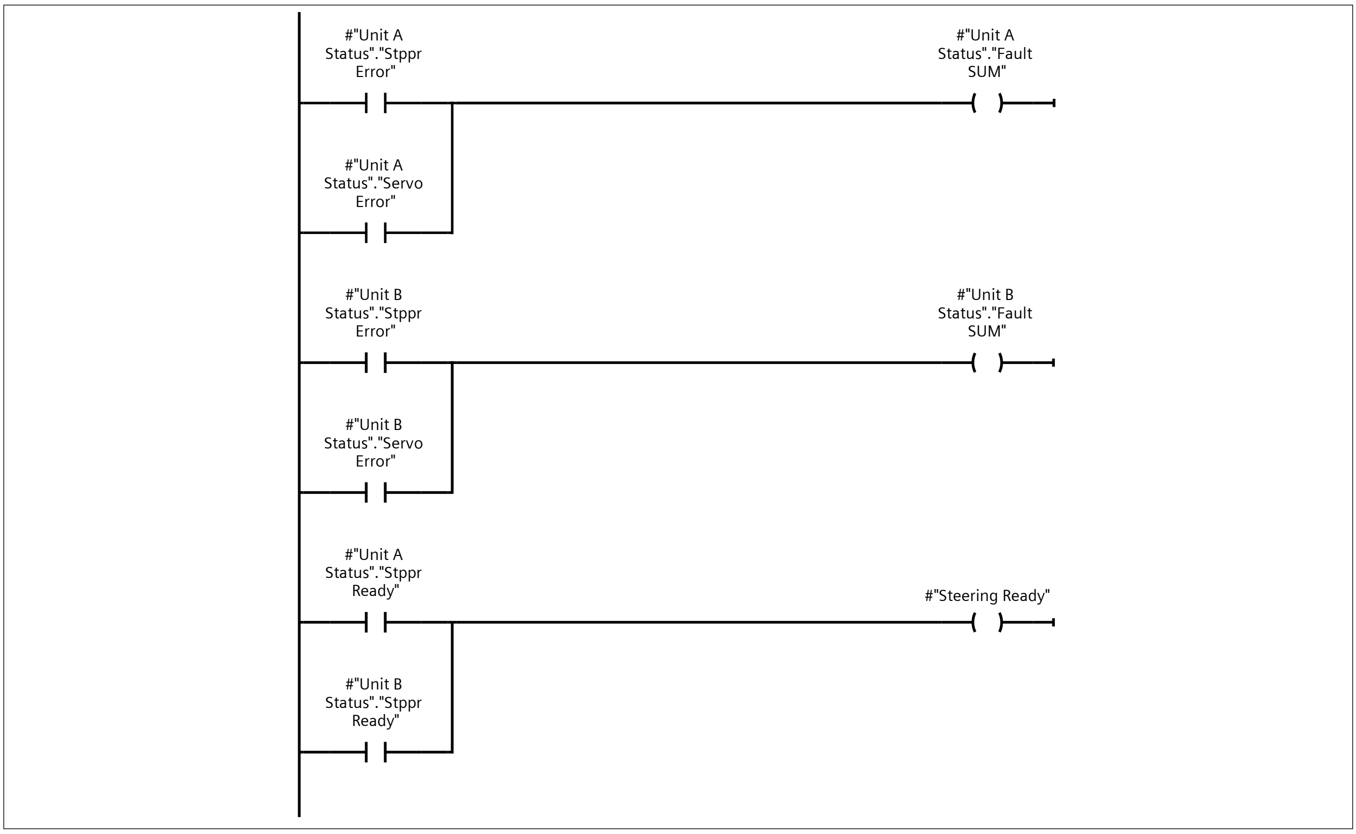
Network 4: Servo A Control



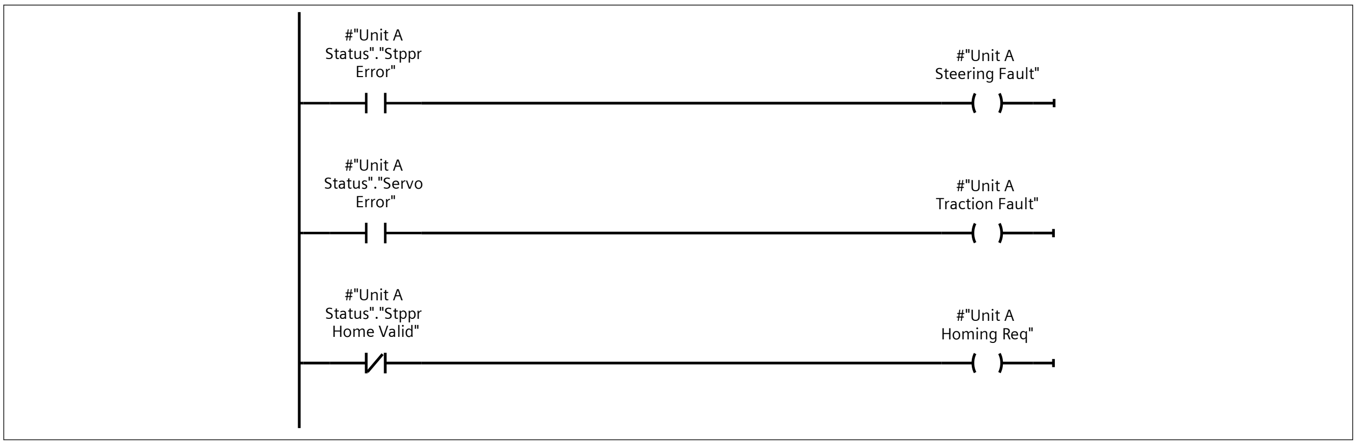
Network 5: Servo B Control



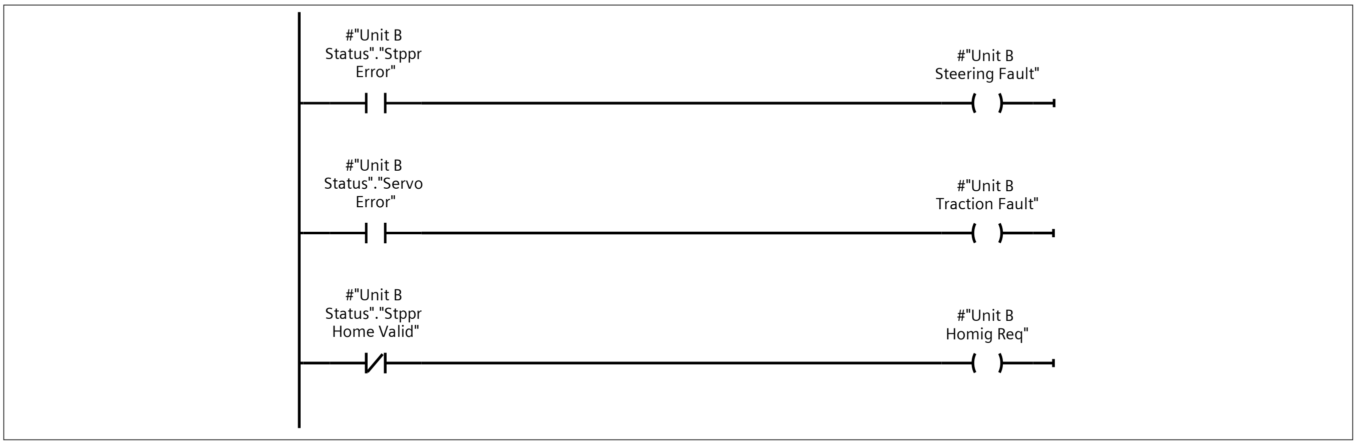
Network 6: Sum Of Faults


#"Unit A Status"."Stppr Error"
#"Unit A Status"."Fault SUM"
#"Unit A Status"."Servo Error"
#"Unit B Status"."Stppr Error"
#"Unit B Status"."Fault SUM"
#"Unit B Status"."Servo Error"
#"Unit A Status"."Stppr Ready"
#"Steering Ready"
#"Unit B Status"."Stppr Ready"

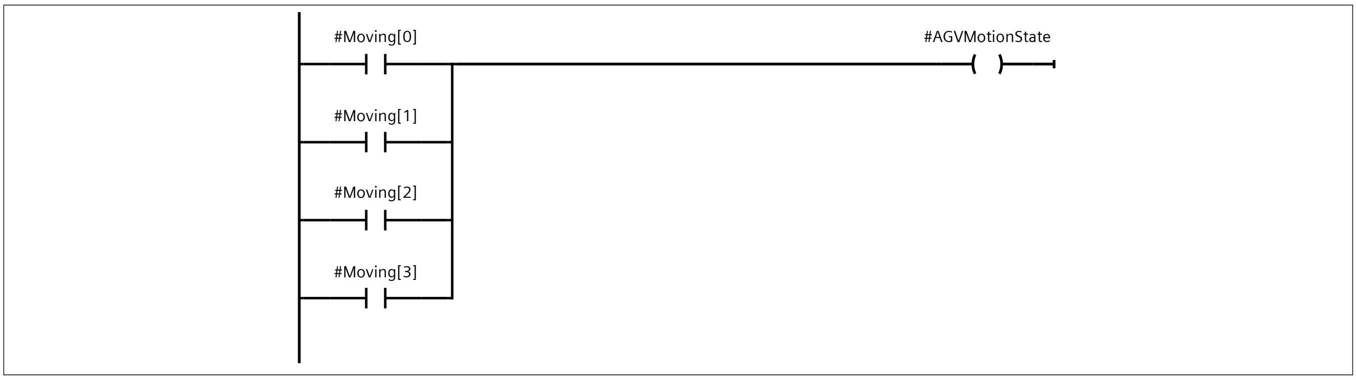
Network 7: Pass statuses to outputs unit A


#"Unit A Status"."Stppr Error"
#"Unit A Steering Fault"
#"Unit A Status"."Servo Error"
#"Unit A Traction Fault"
#"Unit A Status"."Stppr Home Valid"
#"Unit A Homing Req"

Network 8: Pass statuses to outputs unit B


#"Unit B Status"."Stppr Error"
#"Unit B Steering Fault"
#"Unit B Status"."Servo Error"
#"Unit B Traction Fault"
#"Unit B Status"."Stppr Home Valid"
#"Unit B Homig Req"

Network 9: Check if any drives are moving



10. Plant (Motor Control)

MC-Interpolator [OB92]

MC-Interpolator Properties					
General					
Name	MC-Interpolator	**Number**	92	**Type**	OB
Language	LAD	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	1.0	**User-defined ID**	

Name	Data type	Default value
▼ Input		
Initial_Call	Bool	
PIP_Input	Bool	
PIP_Output	Bool	
IO_System	USInt	
Event_Count	Int	
Reduction	UInt	

10. Plant (Motor Control)

MC-Servo [OB91]

MC-Servo Properties					
General					
Name	MC-Servo	**Number**	91	**Type**	OB
Language	LAD	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	1.0	**User-defined ID**	

Name	Data type	Default value
▼ Input		
Initial_Call	Bool	
PIP_Input	Bool	
PIP_Output	Bool	
IO_System	USInt	
Event_Count	Int	
Synchronous	Bool	

10. Plant (Motor Control)

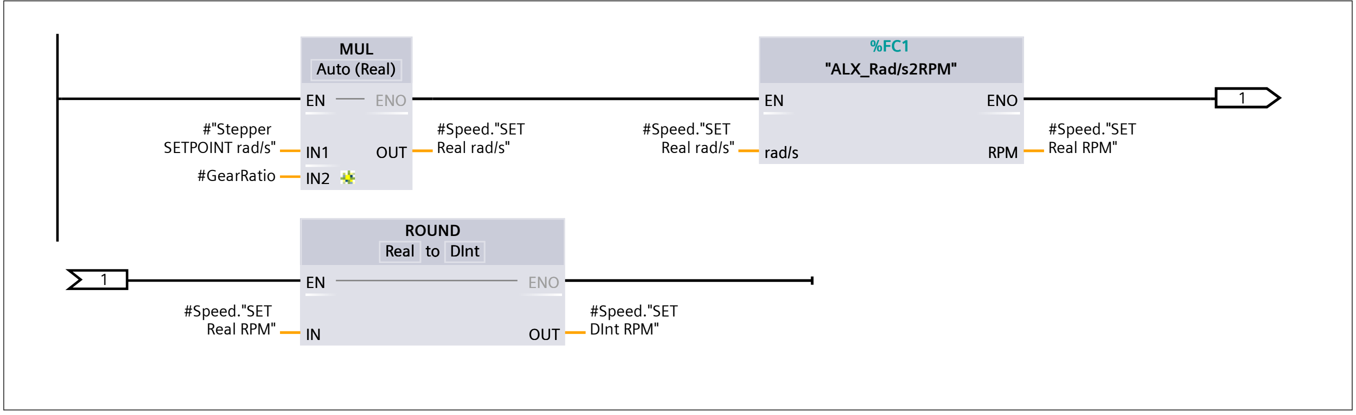
FB Unit A Stepper Velocity [FB310]

FB Unit A Stepper Velocity Properties					
General					
Name	FB Unit A Stepper Velocity	**Number**	310	**Type**	FB
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
▼ Input			
Enable	Bool	false	Non-retain
Halt (Low)	Bool	false	Non-retain
Brake (Low)	Bool	false	Non-retain
Update	Bool	false	Non-retain
Home	Bool	false	Non-retain
Acknowledge	Bool	false	Non-retain
Stepper SETPOINT rad/s	Real	0.0	Non-retain
GearRatio	Real	0.0	Non-retain
HW_IO	HW_IO	0	Non-retain
▼ Output			
Ready	Bool	false	Non-retain
Moving	Bool	false	Non-retain
Stepper ACTUAL rad/s	Real	0.0	Non-retain
Stepper ACTUAL RPM	Real	0.0	Non-retain
Home Valid	Bool	false	Non-retain
Error	Bool	false	Non-retain
InOut			
▼ Static			
FML_REF_Instance	"DT_FML_REF"		Non-retain
FHPP_DPRD_DAT_Instance	"FHPP_DPRD_DAT"		
FHPP_CTRL_Instance	"FHPP_CTRL"		
FHPP_DPWR_DAT_Instance	"FHPP_DPWR_DAT"		
Errors	Struct		Non-retain
Speed	Struct		Non-retain
P_Edge	Bool	false	Non-retain
Temp			
Constant			

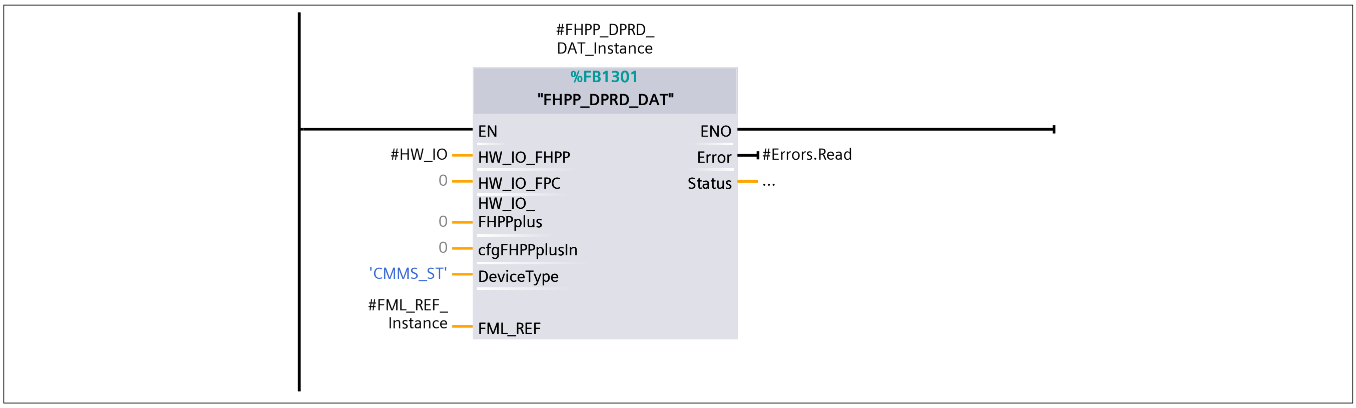
Network 1: Format incomming actual speed

Network 1: Format incomming actual speed



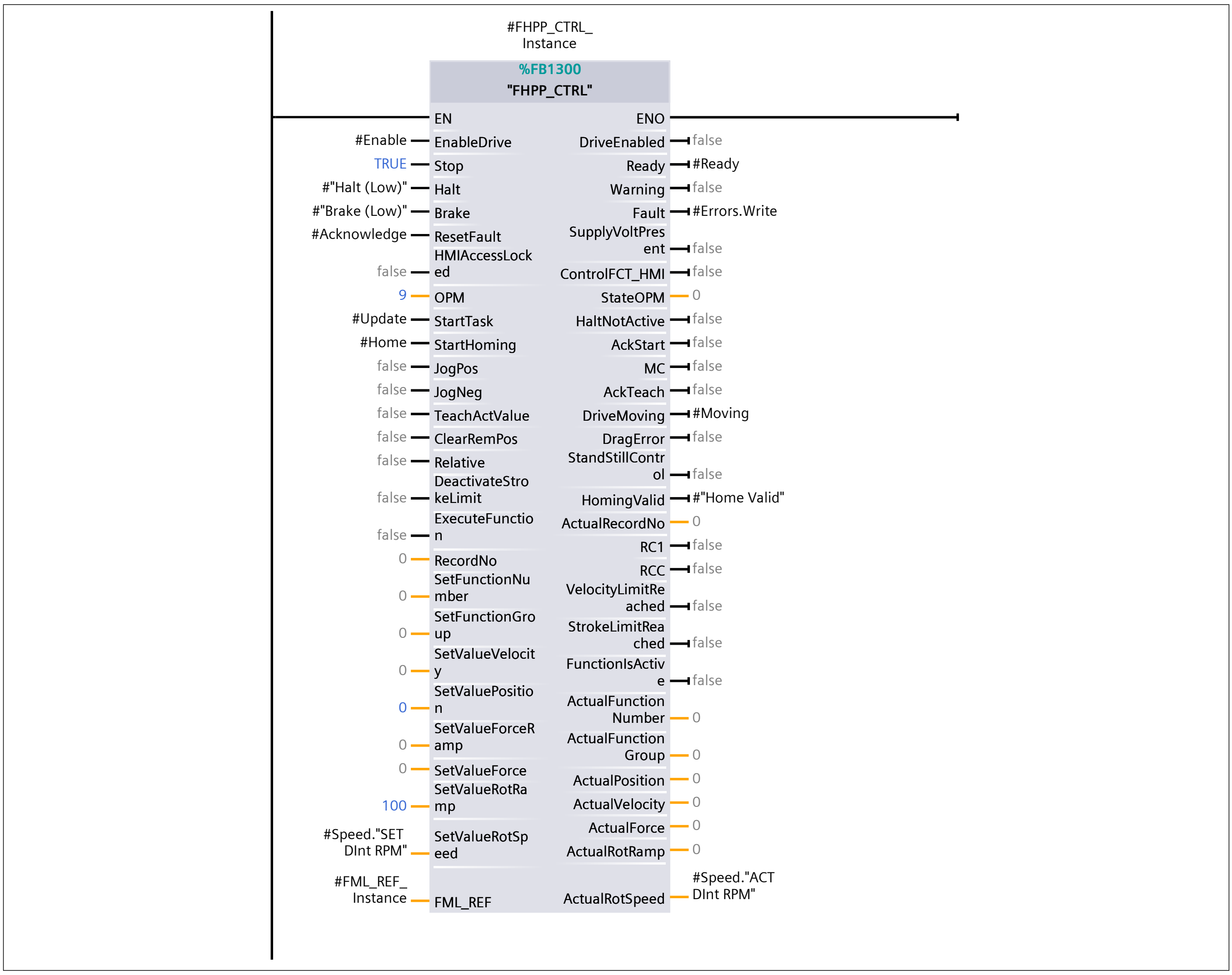
Network 2: Read Data From Stepper FHPP

THIS NETWORK MUST BE FIRST



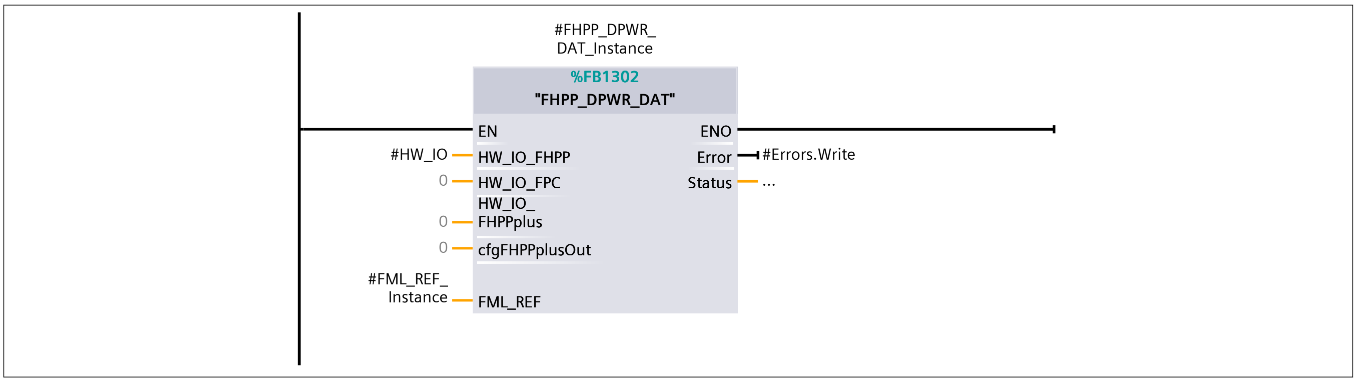
Network 3: Control Data Stepper FHPP

THIS NETWORK MUST BE SECOND



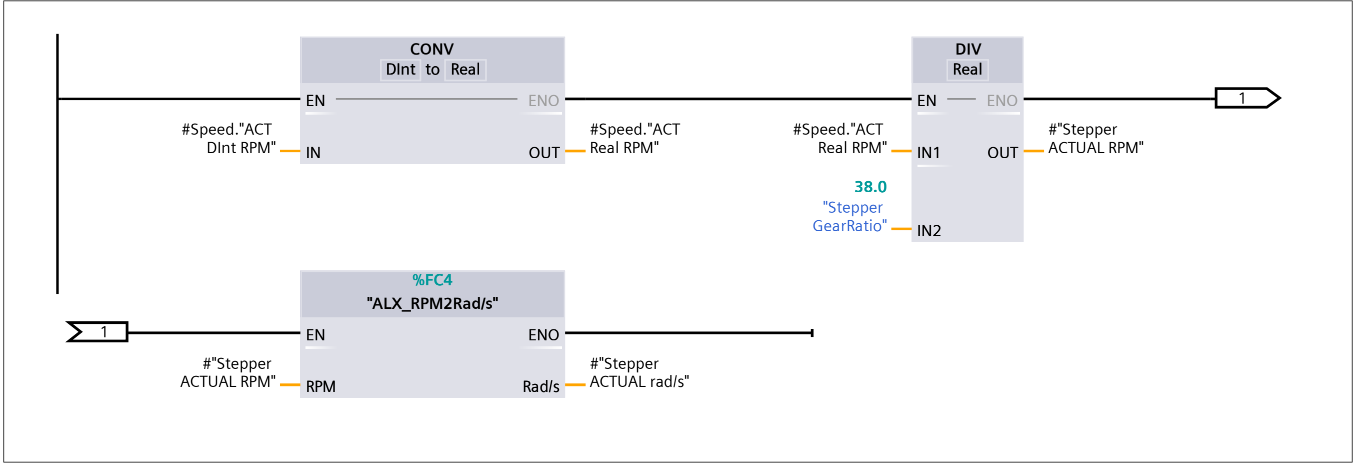
Network 4: Write Data Stepper FHPP

THIS NETWORK MUST BE THIRD

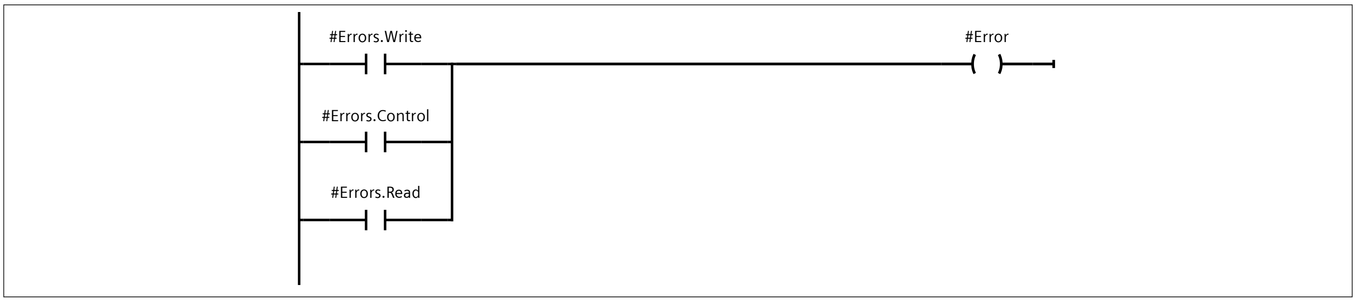


Network 5: Format outgoing actual speed

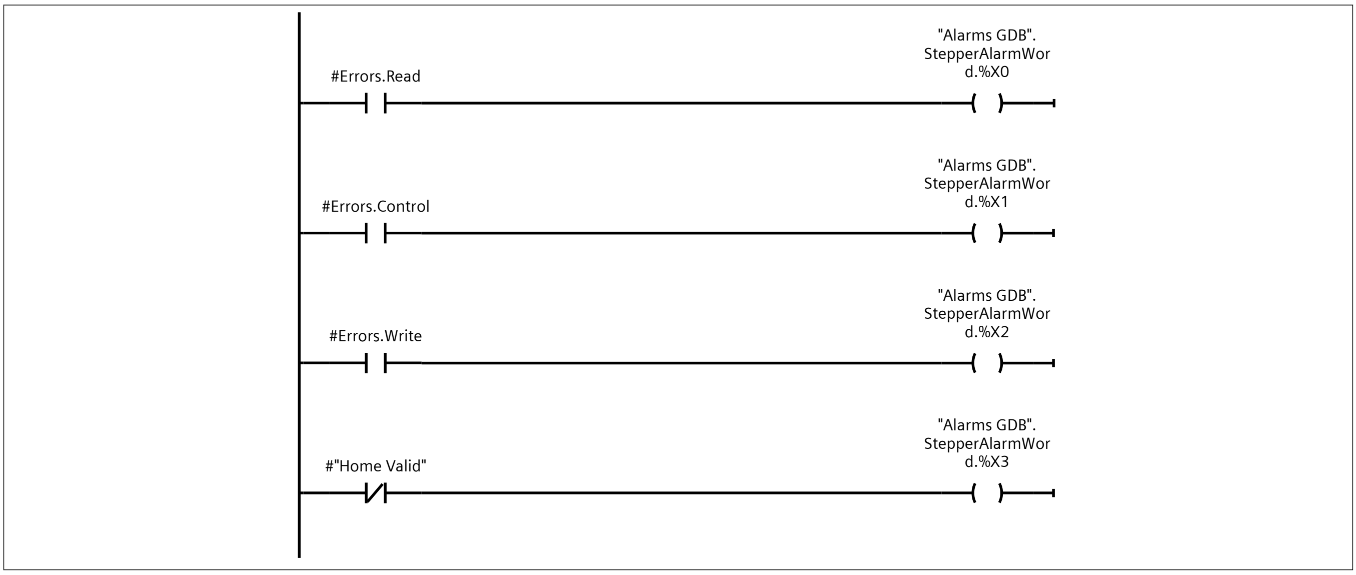
Network 5: Format outgoing actual speed



Network 6: Error Status Update



Network 7: Alarms



10. Plant (Motor Control)

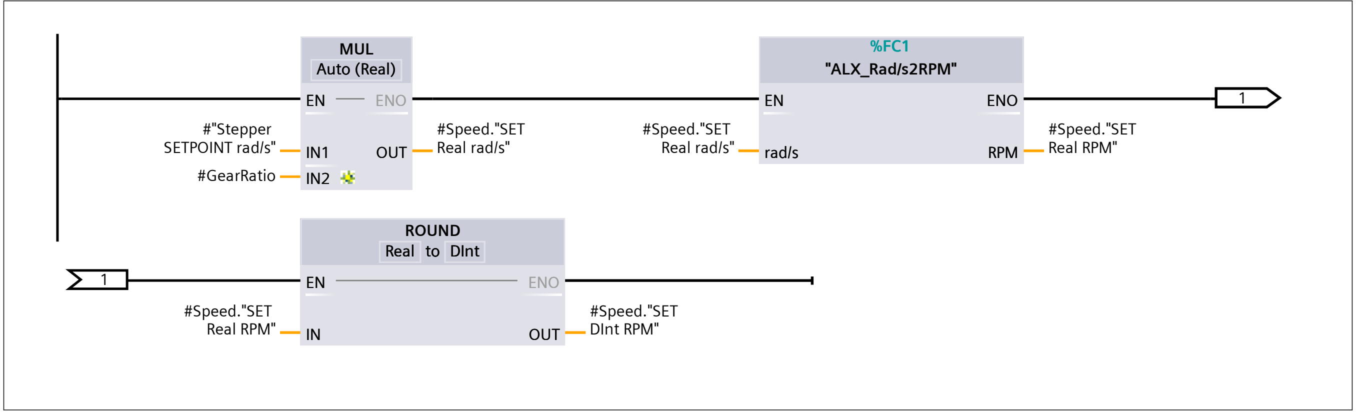
FB Unit B Stepper Velocity [FB311]

FB Unit B Stepper Velocity Properties					
General					
Name	FB Unit B Stepper Velocity	**Number**	311	**Type**	FB
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
▼ Input			
Enable	Bool	false	Non-retain
Halt (Low)	Bool	false	Non-retain
Brake (Low)	Bool	false	Non-retain
Update	Bool	false	Non-retain
Home	Bool	false	Non-retain
Acknowledge	Bool	false	Non-retain
Stepper SETPOINT rad/s	Real	0.0	Non-retain
GearRatio	Real	0.0	Non-retain
HW_IO	HW_IO	0	Non-retain
▼ Output			
Ready	Bool	false	Non-retain
Moving	Bool	false	Non-retain
Stepper ACTUAL rad/s	Real	0.0	Non-retain
Stepper ACTUAL RPM	Real	0.0	Non-retain
Home Valid	Bool	false	Non-retain
Error	Bool	false	Non-retain
InOut			
▼ Static			
FML_REF_Instance	"DT_FML_REF"		Non-retain
FHPP_DPRD_DAT_Instance	"FHPP_DPRD_DAT"		
FHPP_CTRL_Instance	"FHPP_CTRL"		
FHPP_DPWR_DAT_Instance	"FHPP_DPWR_DAT"		
Errors	Struct		Non-retain
Speed	Struct		Non-retain
P_Edge	Bool	false	Non-retain
Temp			
Constant			

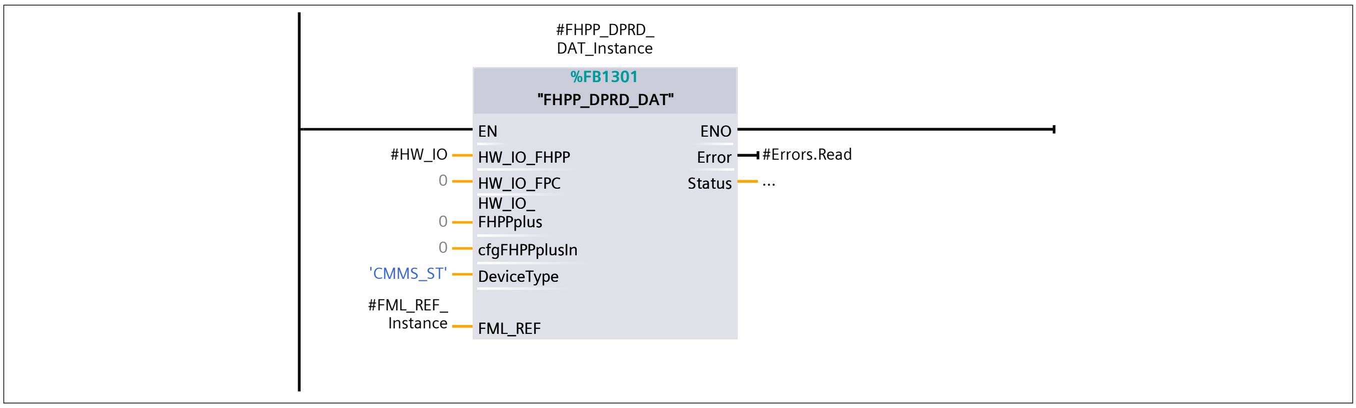
Network 1: Format incomming actual speed

Network 1: Format incomming actual speed



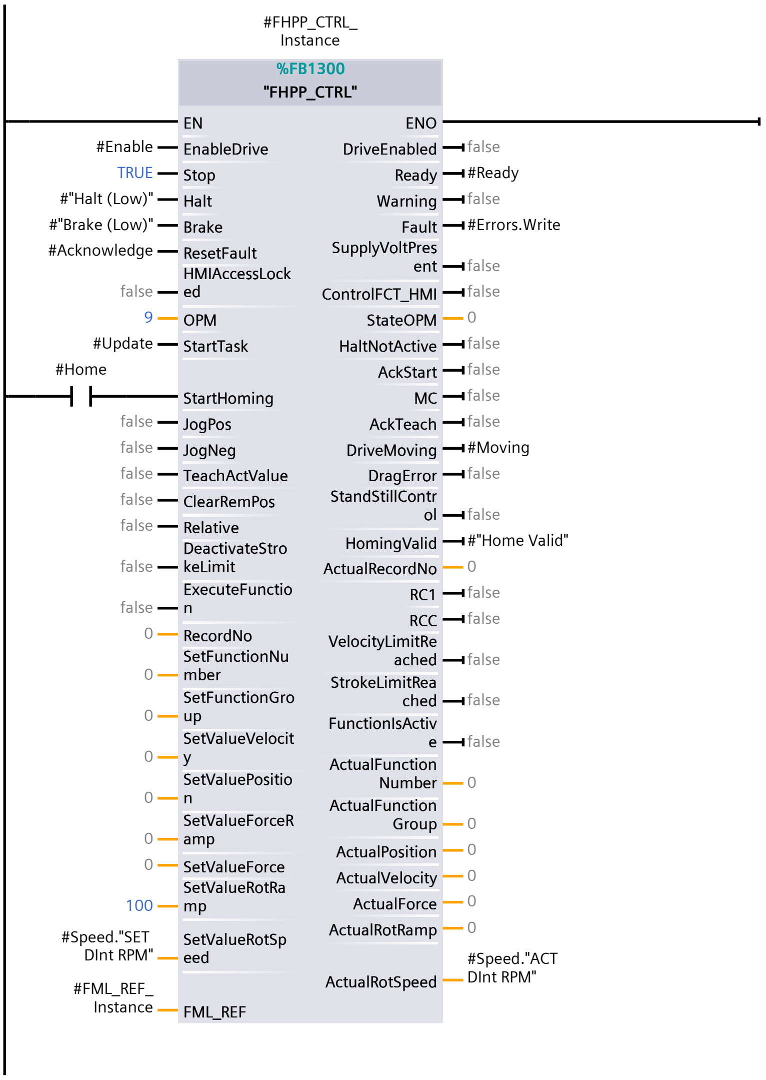
Network 2: Read Data From Stepper FHPP

THIS NETWORK MUST BE FIRST



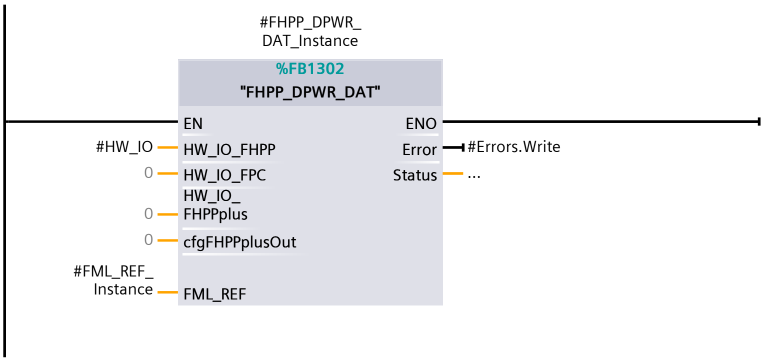
Network 3: Control Data Stepper FHPP

THIS NETWORK MUST BE SECOND



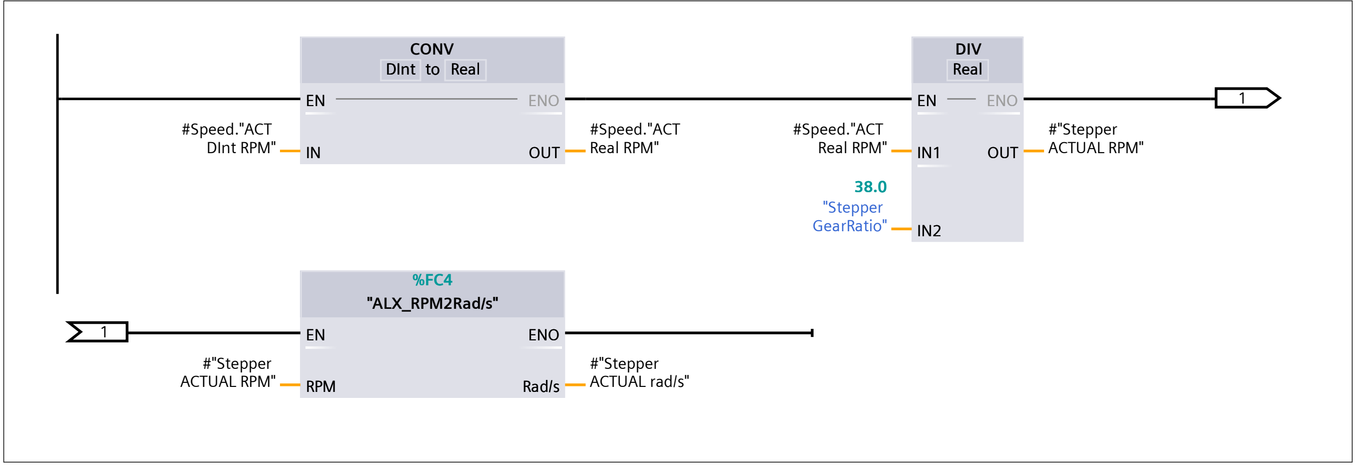
Network 4: Write Data Stepper FHPP

THIS NETWORK MUST BE THIRD

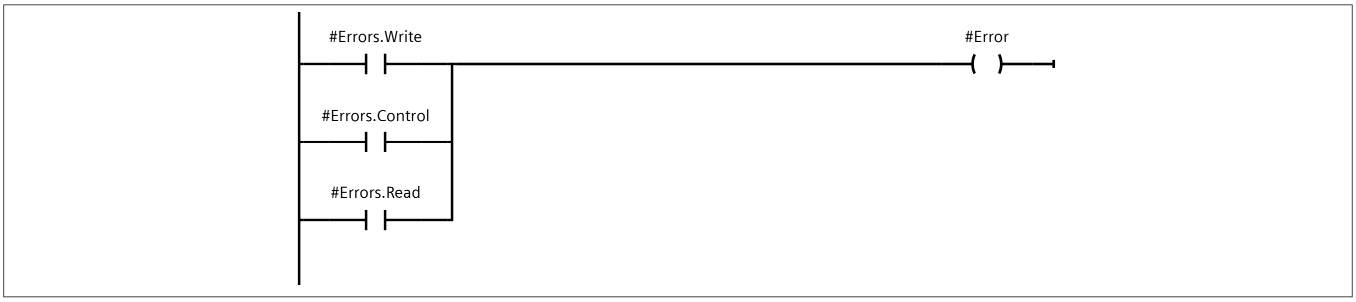


Network 5: Format outgoing actual speed

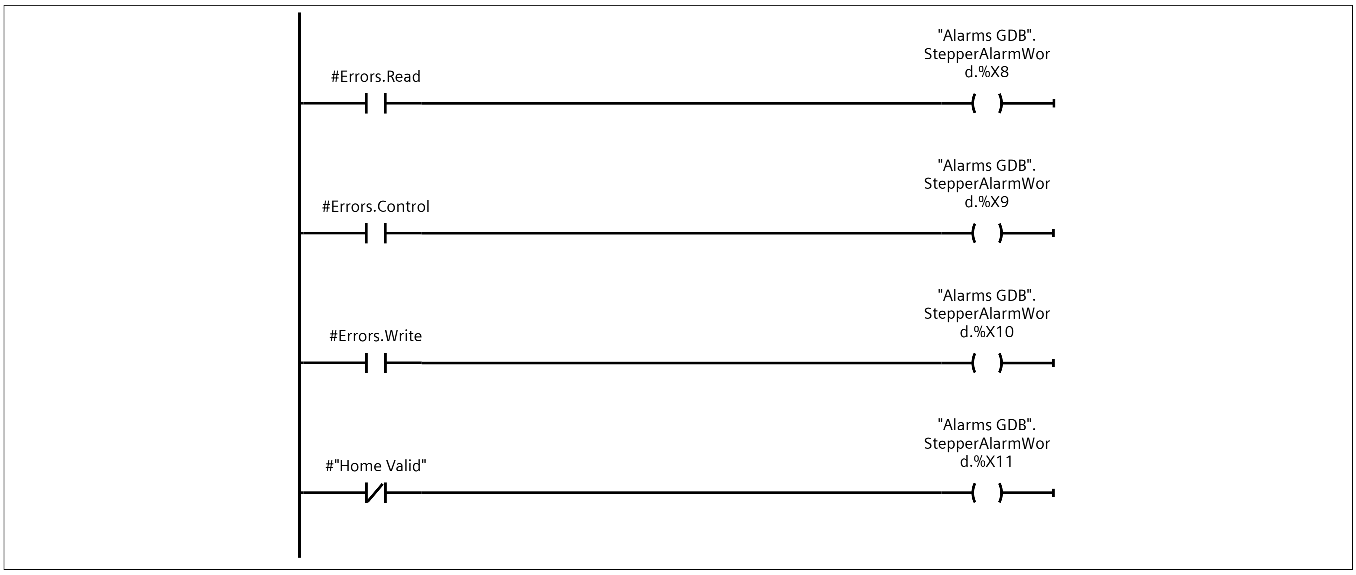
Network 5: Format outgoing actual speed



Network 6: Error Status Update



Network 7: Alarms



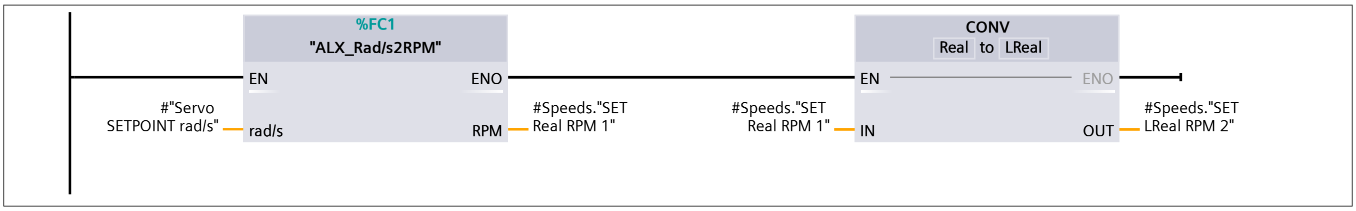
10. Plant (Motor Control)

FB Unit A Servo [FB312]

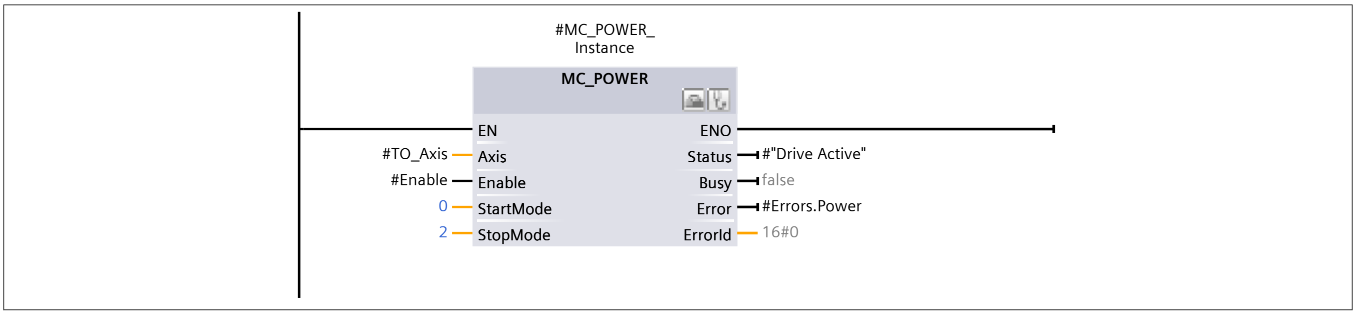
FB Unit A Servo Properties					
General					
Name	FB Unit A Servo	**Number**	312	**Type**	FB
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
▼ Input			
Enable	Bool	false	Non-retain
Halt (Low)	Bool	false	Non-retain
Brake (Low)	Bool	false	Non-retain
Update	Bool	false	Non-retain
Acknowledge	Bool	false	Non-retain
Servo SETPOINT rad/s	Real	0.0	Non-retain
Telegram: NIST_B	DInt	0	Non-retain
Servo Saturation Pt	Real	0.0	Non-retain
GearRatio	Real	0.0	Non-retain
▼ Output			
Ready	Bool	false	Non-retain
Moving	Bool	false	Non-retain
Servo ACTUAL rad/s	Real	0.0	Non-retain
Servo ACTUAL RPM	Real	0.0	Non-retain
Home Valid	Bool	TRUE	Non-retain
Error	Bool	false	Non-retain
▼ InOut			
TO_Axis	TO_SpeedAxis		
▼ Static			
MC_POWER_Instance	MC_POWER		
MC_MOVEVELOCITY_Instance	MC_MOVEVELOCITY		
MC_HALT_Instance	MC_HALT		
MC_RESET_Instance	MC_RESET		
Errors	Struct		Non-retain
Speeds	Struct		Non-retain
Drive Active	Bool	false	Non-retain
Temp			
▼ Constant			
NIST_B Positive Limit	DInt	1073741824	
NIST_B Negative Limit	DInt	-1073741824	

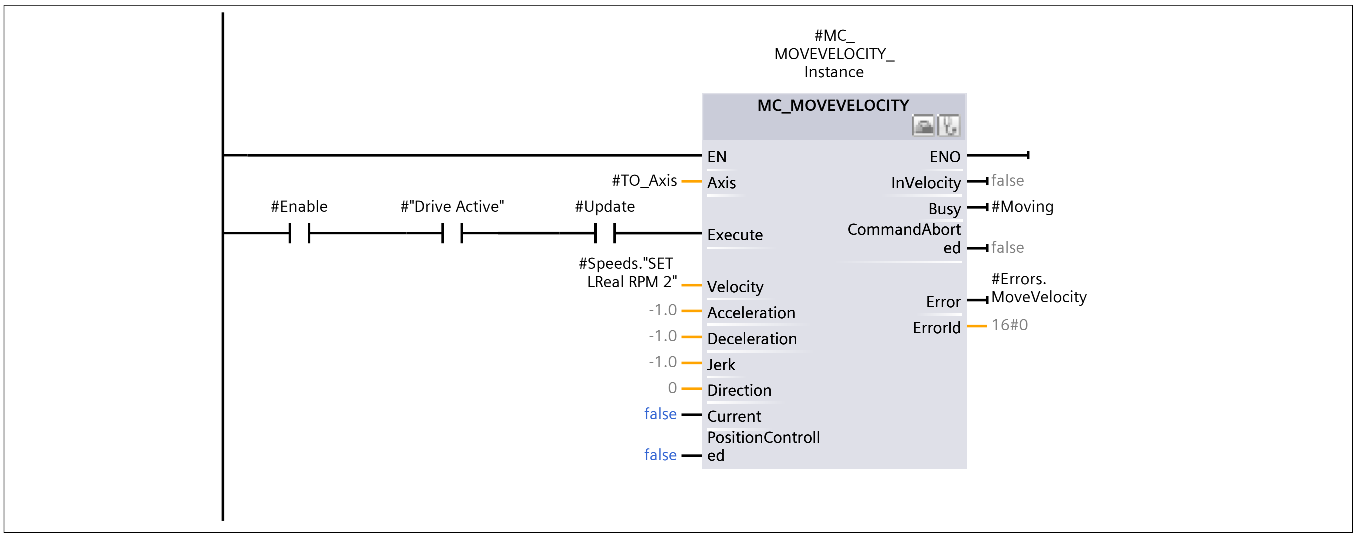
Network 1: Format in-comming speed value



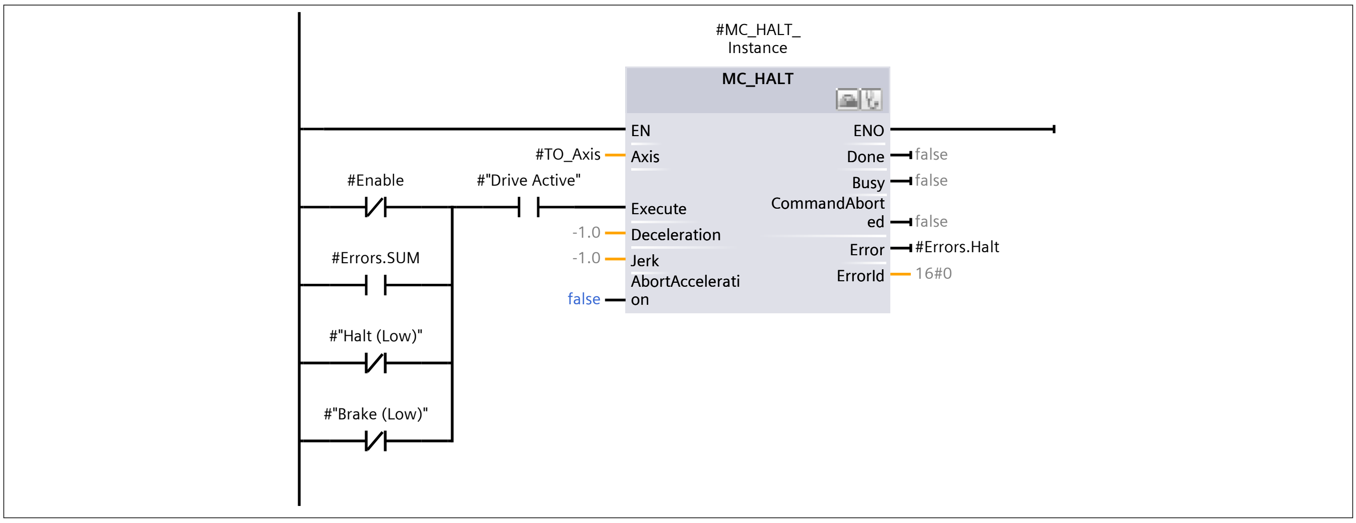
Network 2: Enable drive



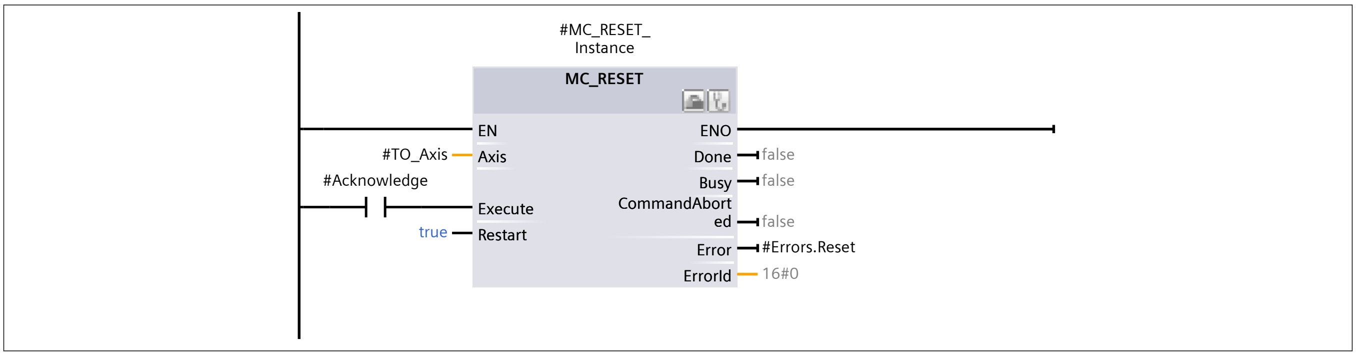
Network 3: Move drive with velocity control



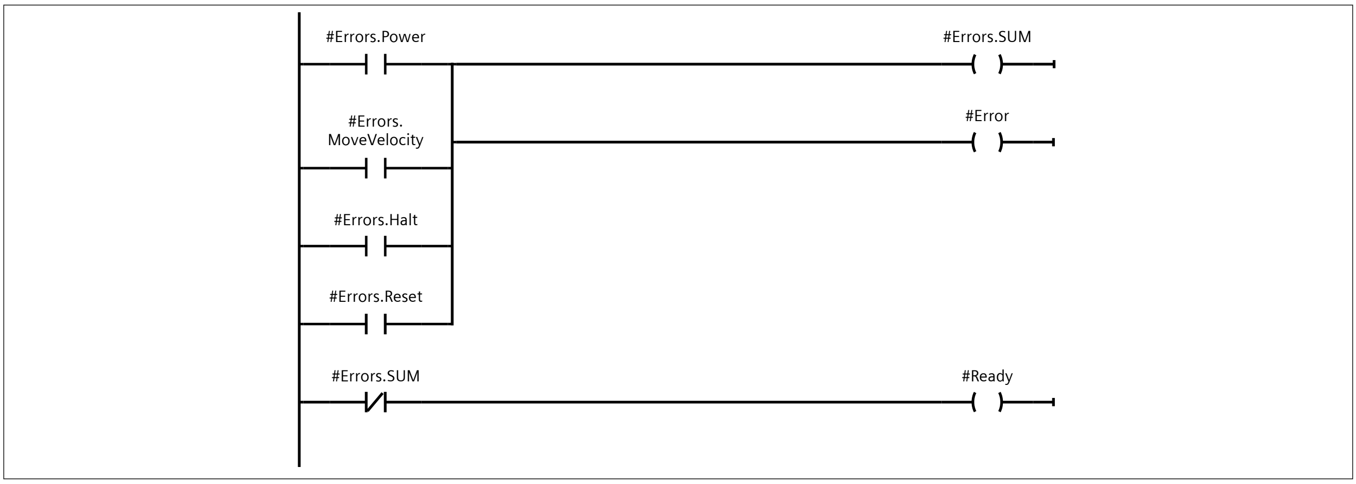
Network 4: Halt drive



Network 5: Reset drive using acknowledge if fault occured



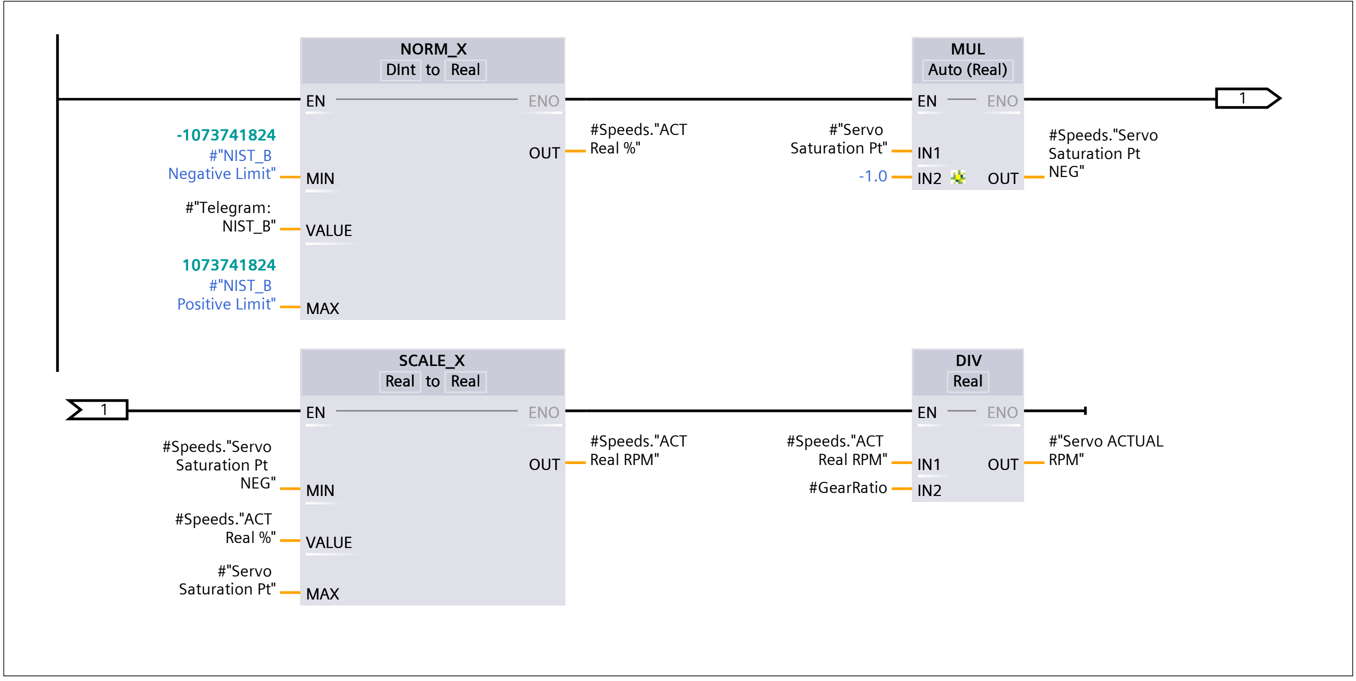
Network 6: Sum of faults



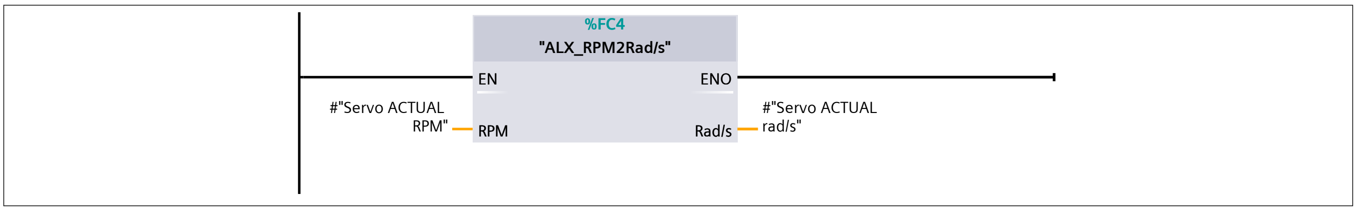
Network 7: Find actual speed of servo

Since Siemens are assholes the only way to gather ther actual speed from the drive is via bitbanging the appropriate memory spaceused in the profinet telegram.
This memory space is called NIST_B

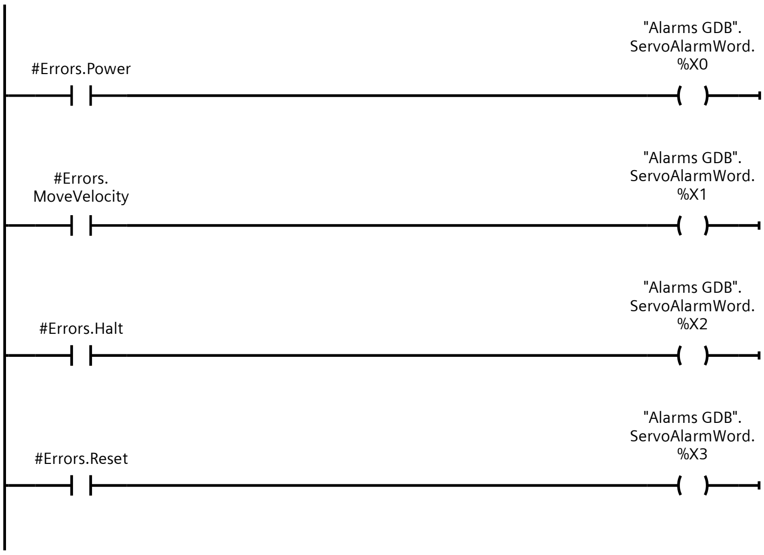
Network 7: Find actual speed of servo



Network 8: Convert actual RPM to rad/s



Network 9: Alarm Messages


"Alarms GDB".
ServoAlarmWord.
%X0
#Errors.Power
"Alarms GDB".
ServoAlarmWord.
%X1
#Errors.
MoveVelocity
"Alarms GDB".
ServoAlarmWord.
%X2
#Errors.Halt
"Alarms GDB".
ServoAlarmWord.
%X3
#Errors.Reset

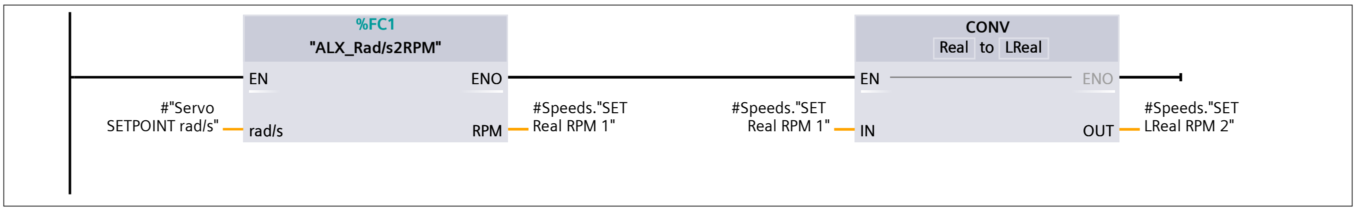
10. Plant (Motor Control)

FB Unit B Servo [FB308]

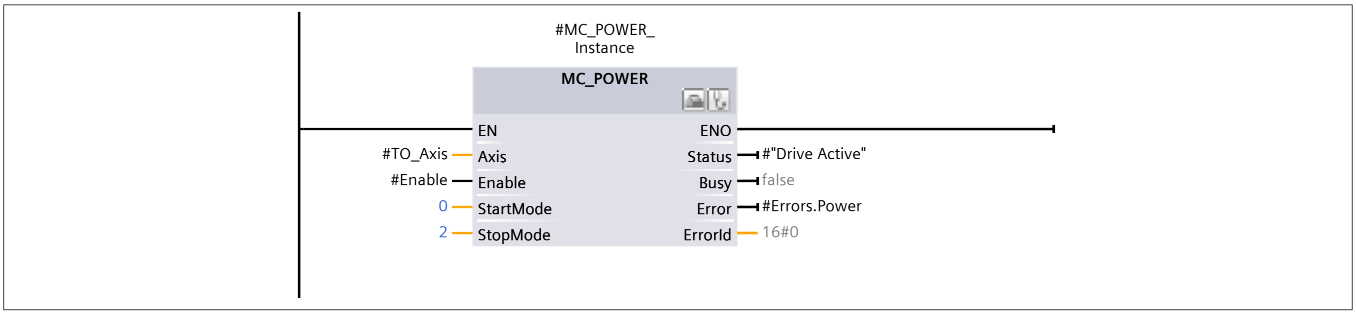
FB Unit B Servo Properties					
General					
Name	FB Unit B Servo	**Number**	308	**Type**	FB
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
▼ Input			
Enable	Bool	false	Non-retain
Halt (Low)	Bool	false	Non-retain
Brake (Low)	Bool	false	Non-retain
Update	Bool	false	Non-retain
Acknowledge	Bool	false	Non-retain
Servo SETPOINT rad/s	Real	0.0	Non-retain
Telegram: NIST_B	DInt	0	Non-retain
Servo Saturation Pt	Real	0.0	Non-retain
GearRatio	Real	0.0	Non-retain
▼ Output			
Ready	Bool	false	Non-retain
Moving	Bool	false	Non-retain
Servo ACTUAL rad/s	Real	0.0	Non-retain
Servo ACTUAL RPM	Real	0.0	Non-retain
Home Valid	Bool	TRUE	Non-retain
Error	Bool	false	Non-retain
▼ InOut			
TO_Axis	TO_SpeedAxis		
▼ Static			
MC_POWER_Instance	MC_POWER		
MC_MOVEVELOCITY_Instance	MC_MOVEVELOCITY		
MC_HALT_Instance	MC_HALT		
MC_RESET_Instance	MC_RESET		
Errors	Struct		Non-retain
Speeds	Struct		Non-retain
Drive Active	Bool	false	Non-retain
Temp			
▼ Constant			
NIST_B Positive Limit	DInt	1073741824	
NIST_B Negative Limit	DInt	-1073741824	

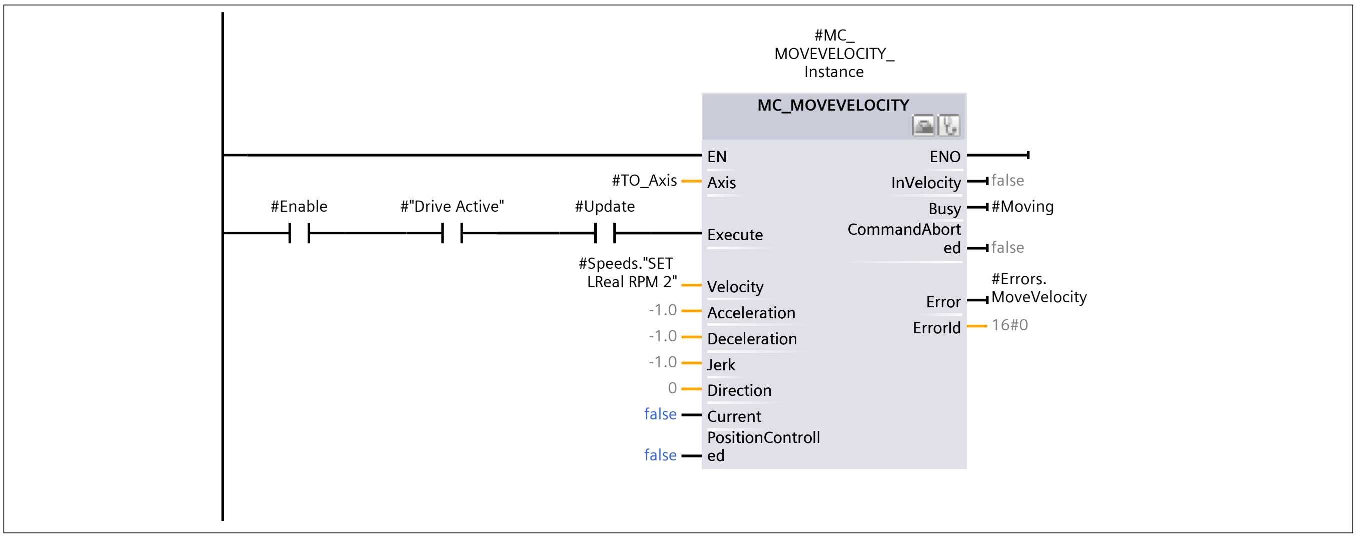
Network 1: Format in-comming speed value



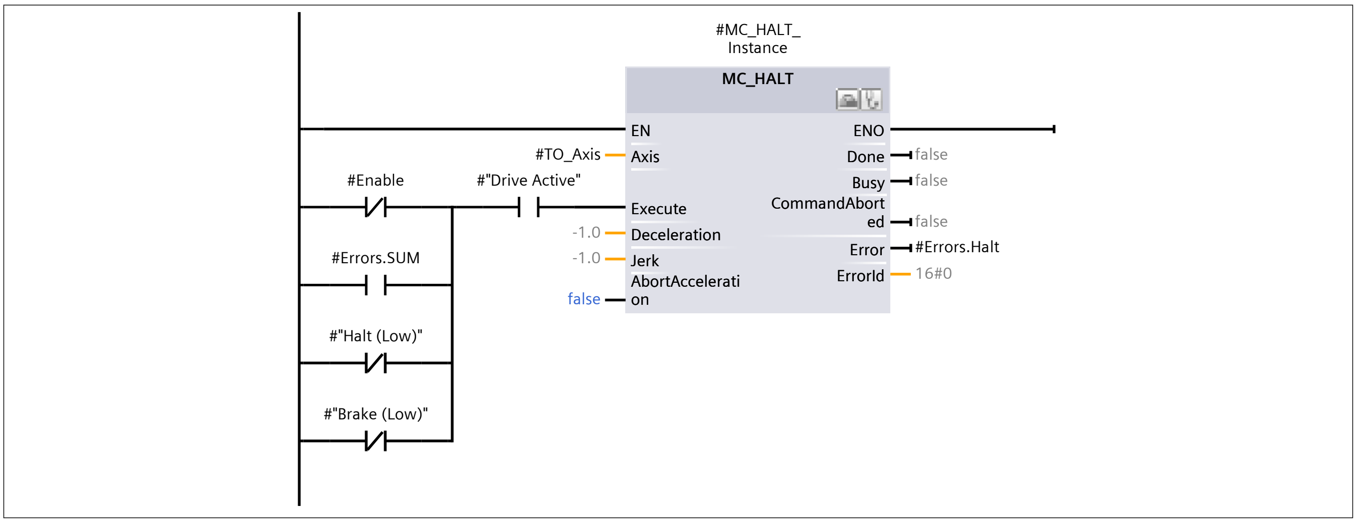
Network 2: Enable drive



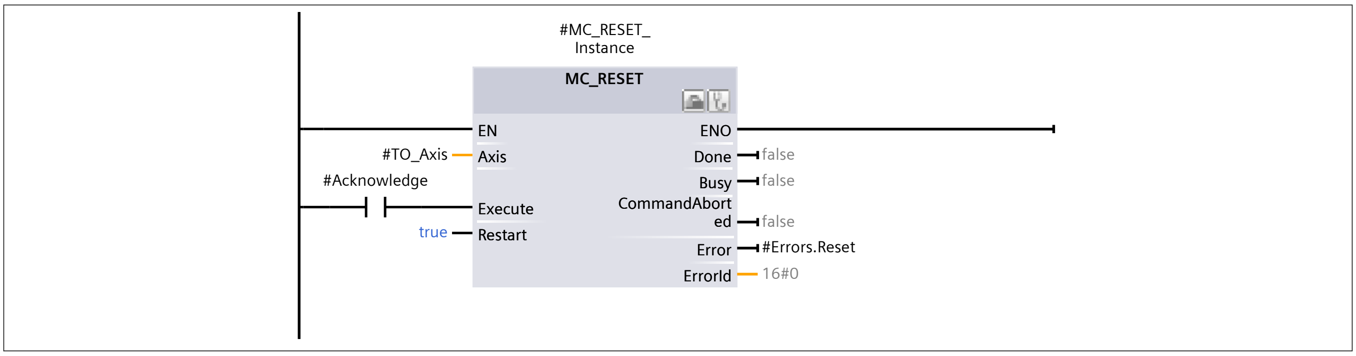
Network 3: Move drive with velocity control



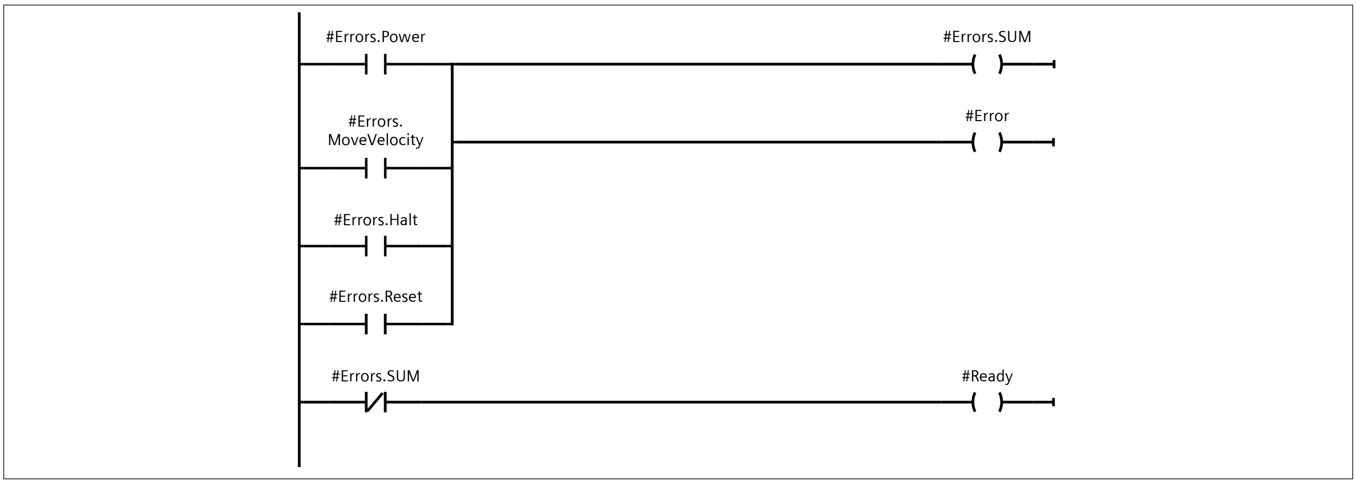
Network 4: Halt drive



Network 5: Reset drive using acknowledge if fault occured



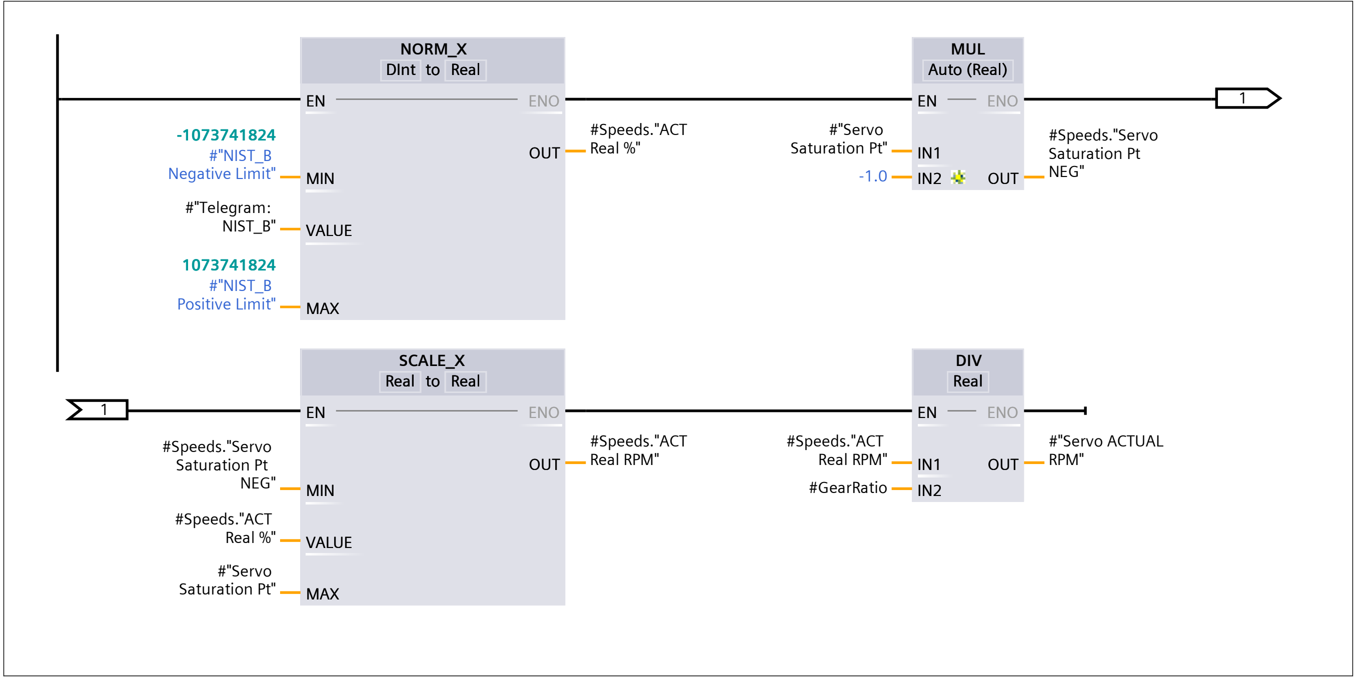
Network 6: Sum of faults



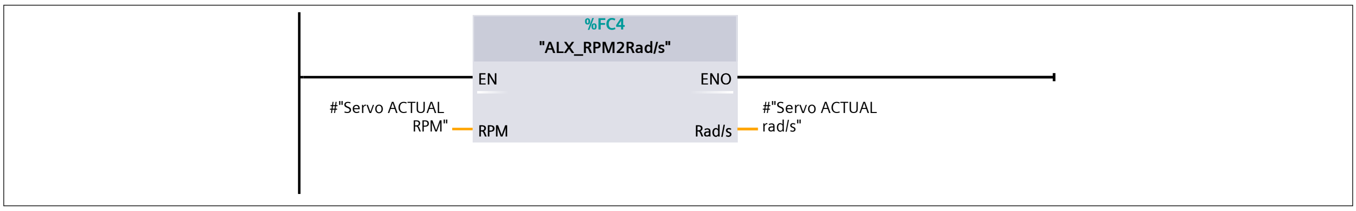
Network 7: Find actual speed of servo

Since Siemens are assholes the only way to gather ther actual speed from the drive is via bitbanging the appropriate memory spaceused in the profinet telegram.
This memory space is called NIST_B

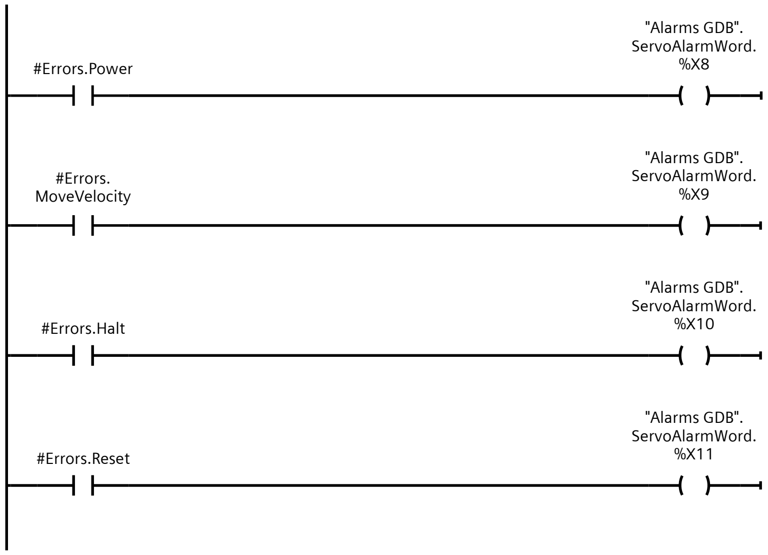
Network 7: Find actual speed of servo



Network 8: Convert actual RPM to rad/s



Network 9: Alarm Messages


"Alarms GDB".
ServoAlarmWord.
%X8
#Errors.Power
"Alarms GDB".
ServoAlarmWord.
%X9
#Errors.
MoveVelocity
"Alarms GDB".
ServoAlarmWord.
%X10
#Errors.Halt
"Alarms GDB".
ServoAlarmWord.
%X11
#Errors.Reset

10. Plant (Motor Control)

FC Choose Setpoint Source [FC8]

FC Choose Setpoint Source Properties					
General					
Name	FC Choose Setpoint Source	**Number**	8	**Type**	FC
Language	LAD	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value
▼ Input		
Homing Mode	Bool	
Commisiong Mode	Bool	
Manual Mode	Bool	
HMI Unit Move Mode	Bool	
HMI AGV Move Mode	Bool	
Automatic Mode	Bool	
Testing Mode	Bool	
HomingMd_SteeringSpd_A	Real	
HomingMd_SteeringSpd_B	Real	
ForwardKi_SteeringSpd_A	Real	
ForwardKi_SteeringSpd_B	Real	
ForwardKi_TractionSpd_A	Real	
ForwardKi_TractionSpd_B	Real	
ComisngMd_SteeringSpd_A	Real	
ComisngMd_SteeringSpd_B	Real	
ComisngMd_TractionSpd_A	Real	
ComisngMd_TractionSpd_B	Real	
▼ Output		
SET TractionSpeed A	Real	
SET TractionSpeed B	Real	
SET SteeringSpeed A	Real	
SET SteeringSpeed B	Real	
InOut		
Temp		
Constant		
▼ Return		
FC Choose Setpoint Source	Void	

Network 1: Homing Mode Selected

```
0001 IF (#"Homing Mode" = TRUE) AND (#"Commisiong Mode" = FALSE) AND (#"HMI Unit Move
     Mode" = FALSE) AND
```

```
    (#"Manual Mode" = FALSE) AND (#"Automatic Mode" = FALSE) AND (#"Testing Mode" = FALSE) AND
    (#"HMI AGV Move Mode" = FALSE) THEN
    // Homing mode == true, thus steering will attempt to align to zero, note that tractive speeds are forced to zero
    #"SET TractionSpeed A" := 0.0;
    #"SET TractionSpeed B" := 0.0;
    #"SET SteeringSpeed A" := #HomingMd_SteeringSpd_A;
    #"SET SteeringSpeed B" := #HomingMd_SteeringSpd_B;
END_IF;

```

Network 2: Commisiong Mode Selected

```
IF (#"Homing Mode" = FALSE) AND (#"Manual Mode" = FALSE) AND (#"Automatic Mode" = FALSE) AND
    (#"Testing Mode" = FALSE) AND (#"HMI AGV Move Mode" = FALSE) THEN

    IF (#"HMI Unit Move Mode" = TRUE) OR (#"Commisiong Mode" = TRUE) THEN
        // Commisiong mode = true or HMI UnitMode = true
        #"SET TractionSpeed A" := #ComisngMd_TractionSpd_A;
        #"SET TractionSpeed B" := #ComisngMd_TractionSpd_B;
        #"SET SteeringSpeed A" := #ComisngMd_SteeringSpd_A;
        #"SET SteeringSpeed B" := #ComisngMd_SteeringSpd_B;
    END_IF;

END_IF;
```

Network 3: Manual Mode or Automatic Mode Selected

```
//If either automatic mode or manual mode 

IF (#"Homing Mode" = FALSE) AND (#"Commisiong Mode" = FALSE) AND (#"HMI Unit Move Mode" = FALSE) THEN
    // Not homing nor comissioning
    //
    IF (#"Manual Mode" = TRUE) OR (#"Automatic Mode" = TRUE) OR
        (#"Testing Mode" = TRUE) OR (#"HMI AGV Move Mode" = TRUE) THEN
        // Either in manual or automatic mode, either way setpoint RPMs come from the forward kinematics
        #"SET TractionSpeed A" := #ForwardKi_TractionSpd_A;
        #"SET TractionSpeed B" := #ForwardKi_TractionSpd_B;
        #"SET SteeringSpeed A" := #ForwardKi_SteeringSpd_A;
        #"SET SteeringSpeed B" := #ForwardKi_SteeringSpd_B;
    END_IF;

END_IF;


```

Network 4: No Mode Selected Catch All

```
IF (#"Homing Mode" = FALSE) AND (#"Commisiong Mode" = FALSE) AND (#"Manual Mode" = FALSE) AND
    (#"Automatic Mode" = FALSE) AND (#"Testing Mode" = FALSE) AND (#"HMI Unit Move Mode" = FALSE)
    AND (#"HMI AGV Move Mode" = FALSE) THEN
    // No mode selected = set all RPM to 0 to prevent erroneous running
    #"SET TractionSpeed A" := 0.0;
    #"SET TractionSpeed B" := 0.0;
    #"SET SteeringSpeed A" := 0.0;
    #"SET SteeringSpeed B" := 0.0;
END_IF;
```

10. Plant (Motor Control) / Instance DBs

Main Motor Control iDB [DB309]

Main Motor Control iDB Properties					
General					
Name	Main Motor Control iDB	**Number**	309	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
Activate Drives	Bool	false	False
Trigger Steering Update	Bool	false	False
Trigger Traction Update	Bool	false	False
Unit A Steering Speed SET	Real	0.0	False
Unit A Tractive Speed SET	Real	0.0	False
Unit B Steering Speed SET	Real	0.0	False
Unit B Tractive Speed SET	Real	0.0	False
Unit A Home	Bool	false	False
Unit B Home	Bool	false	False
Acknowledge	Bool	false	False
▼ Output			
Steering Ready	Bool	false	False
Unit A Steering Fault	Bool	false	False
Unit A Traction Fault	Bool	false	False
Unit A Homing Req	Bool	false	False
Unit A Steering Speed ACT	Real	0.0	False
Unit A Steering RPM ACT	Real	0.0	False
Unit A Traction Speed ACT	Real	0.0	False
Unit A Traction RPM ACT	Real	0.0	False
Unit B Steering Fault	Bool	false	False
Unit B Traction Fault	Bool	false	False
Unit B Homig Req	Bool	false	False
Unit B Steering Speed ACT	Real	0.0	False
Unit B Steering RPM ACT	Real	0.0	False
Unit B Traction Speed ACT	Real	0.0	False
Unit B Traction RPM ACT	Real	0.0	False
Moving	Array[0..3] of Bool		False
AGVMotionState	Bool	false	False
InOut			
▼ Static			
Unit A Stepper_Instance	"FB Unit A Stepper Velocity"		False
Unit B Stepper_Instance	"FB Unit B Stepper Velocity"		False
Unit A Status	Struct		False

Name	Data type	Start value	Retain
Unit B Status	Struct		False
Unit A Servo_Instance	"FB Unit A Servo"		False
Unit B Servo_Instance	"FB Unit B Servo"		False

10. Plant (Motor Control) / Instance DBs / Stepper A Drive Control depreciated

DT_FML_REF Stepper A iDB [DB1300]

DT_FML_REF Stepper A iDB Properties					
General					
Name	DT_FML_REF Stepper A iDB	**Number**	1300	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Static			
InData	Struct		False
OutData	Struct		False
Modbus	Bool	false	False
DeviceType	String[16]	''	False
MemberID	Int	0	False
MemberIDmax	Int	0	False
Done	Bool	false	False
Err	Bool	false	False
PNU	Word	16#0	False
ActPNU	Word	16#0	False
ReqID	Int	0	False
RespID	Int	0	False
FPCC_Modus	Byte	16#0	False
FPCS_Modus	Byte	16#0	False
FPCC_Packet	"DT_FML_PRM_FILE"		False
FPCS_Packet	"DT_FML_PRM_FILE"		False
Subindex	Byte	16#0	False
ActSubindex	Byte	16#0	False
DatatypeWR	Byte	16#0	False
DatatypeRD	Byte	16#0	False
ParamValueWR	DInt	0	False
ParamValueRD	DInt	0	False

10. Plant (Motor Control) / Instance DBs / Stepper A Drive Control depreciated

FHPP_CTRL Stepper A iDB [DB1302]

FHPP_CTRL Stepper A iDB Properties					
General					
Name	FHPP_CTRL Stepper A iDB	**Number**	1302	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**	FESTO	**Comment**	
Family	E-DRIVE	**Version**	5.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
EnableDrive	Bool	false	False
Stop	Bool	false	False
Halt	Bool	false	False
Brake	Bool	false	False
ResetFault	Bool	false	False
HMIAccessLocked	Bool	false	False
OPM	Int	0	False
StartTask	Bool	false	False
StartHoming	Bool	false	False
JogPos	Bool	false	False
JogNeg	Bool	false	False
TeachActValue	Bool	false	False
ClearRemPos	Bool	false	False
Relative	Bool	false	False
DeactivateStrokeLimit	Bool	false	False
ExecuteFunction	Bool	false	False
RecordNo	Int	0	False
SetFunctionNumber	Int	0	False
SetFunctionGroup	Int	0	False
SetValueVelocity	DInt	0	False
SetValuePosition	DInt	0	False
SetValueForceRamp	DInt	0	False
SetValueForce	DInt	0	False
SetValueRotRamp	DInt	0	False
SetValueRotSpeed	DInt	0	False
▼ Output			
DriveEnabled	Bool	false	False
Ready	Bool	false	False
Warning	Bool	false	False
Fault	Bool	false	False
SupplyVoltPresent	Bool	false	False
ControlFCT_HMI	Bool	false	False
StateOPM	Int	0	False

Name	Data type	Start value	Retain
HaltNotActive	Bool	false	False
AckStart	Bool	false	False
MC	Bool	false	False
AckTeach	Bool	false	False
DriveMoving	Bool	false	False
DragError	Bool	false	False
StandStillControl	Bool	false	False
HomingValid	Bool	false	False
ActualRecordNo	Int	0	False
RC1	Bool	false	False
RCC	Bool	false	False
VelocityLimitReached	Bool	false	False
StrokeLimitReached	Bool	false	False
FunctionIsActive	Bool	false	False
ActualFunctionNumber	Int	0	False
ActualFunctionGroup	Int	0	False
ActualPosition	DInt	0	False
ActualVelocity	DInt	0	False
ActualForce	DInt	0	False
ActualRotRamp	DInt	0	False
ActualRotSpeed	DInt	0	False
▼ InOut			
FML_REF	"DT_FML_REF"		False
▼ Static			
Converted_BYTE5_8_IN	DInt	0	False
Converted_BYTE5_8_OUT	DInt	0	False
OperationModeB1	Bool	false	False
OperationModeB2	Bool	false	False
ControlModeB1	Bool	false	False
ControlModeB2	Bool	false	False
ContinuousMode	Bool	false	False
StateOperationModeB1	Bool	false	False
StateOperationModeB2	Bool	false	False
StateControlModeB1	Bool	false	False
StateControlModeB2	Bool	false	False
StateContinuousMode	Bool	false	False

10. Plant (Motor Control) / Instance DBs / Stepper A Drive Control depreciated

FHPP_DPRD_DAT Stepper A iDB [DB1304]

FHPP_DPRD_DAT Stepper A iDB Properties					
General					
Name	FHPP_DPRD_DAT Stepper A iDB	**Number**	1304	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**	FESTO	**Comment**	
Family	E-DRIVE	**Version**	5.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
HW_IO_FHPP	HW_IO	0	False
HW_IO_FPC	HW_IO	0	False
HW_IO_FHPPplus	HW_IO	0	False
cfgFHPPplusIn	DInt	0	False
DeviceType	String[16]	''	False
▼ Output			
Error	Bool	false	False
Status	"DT_FML_STATUS"		False
▼ InOut			
FML_REF	"DT_FML_REF"		False
▼ Static			
EnableFHPPplusInOld	DInt	0	False
FHPPplusSize	DInt	0	False
tmpStatus	"DT_FML_STATUS"		False
FHPPplusElementSize	Array[1..9] of DInt		False

10. Plant (Motor Control) / Instance DBs / Stepper A Drive Control depreciated

FHPP_DPWR_DAT Stepper A iDB [DB1306]

FHPP_DPWR_DAT Stepper A iDB Properties					
General					
Name	FHPP_DPWR_DAT Stepper A iDB	**Number**	1306	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**	FESTO	**Comment**	
Family	E-DRIVE	**Version**	5.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
HW_IO_FHPP	HW_IO	0	False
HW_IO_FPC	HW_IO	0	False
HW_IO_FHPPplus	HW_IO	0	False
cfgFHPPplusOut	DInt	0	False
▼ Output			
Error	Bool	false	False
Status	"DT_FML_STATUS"		False
▼ InOut			
FML_REF	"DT_FML_REF"		False
▼ Static			
EnableFHPPplusOutOld	DInt	0	False
FHPPplusSize	DInt	0	False
FHPPplusElementSize	Array[1..9] of DInt		False

10. Plant (Motor Control) / Instance DBs / Stepper B Drive Control depreciated

FHPP_DPRD_DAT Stepper B iDB [DB1305]

FHPP_DPRD_DAT Stepper B iDB Properties					
General					
Name	FHPP_DPRD_DAT Stepper B iDB	**Number**	1305	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**	FESTO	**Comment**	
Family	E-DRIVE	**Version**	5.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
HW_IO_FHPP	HW_IO	0	False
HW_IO_FPC	HW_IO	0	False
HW_IO_FHPPplus	HW_IO	0	False
cfgFHPPplusIn	DInt	0	False
DeviceType	String[16]	''	False
▼ Output			
Error	Bool	false	False
Status	"DT_FML_STATUS"		False
▼ InOut			
FML_REF	"DT_FML_REF"		False
▼ Static			
EnableFHPPplusInOld	DInt	0	False
FHPPplusSize	DInt	0	False
tmpStatus	"DT_FML_STATUS"		False
FHPPplusElementSize	Array[1..9] of DInt		False

10. Plant (Motor Control) / Instance DBs / Stepper B Drive Control depreciated

FHPP_CTRL Stepper B iDB [DB1303]

FHPP_CTRL Stepper B iDB Properties					
General					
Name	FHPP_CTRL Stepper B iDB	**Number**	1303	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**	FESTO	**Comment**	
Family	E-DRIVE	**Version**	5.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
EnableDrive	Bool	false	False
Stop	Bool	false	False
Halt	Bool	false	False
Brake	Bool	false	False
ResetFault	Bool	false	False
HMIAccessLocked	Bool	false	False
OPM	Int	0	False
StartTask	Bool	false	False
StartHoming	Bool	false	False
JogPos	Bool	false	False
JogNeg	Bool	false	False
TeachActValue	Bool	false	False
ClearRemPos	Bool	false	False
Relative	Bool	false	False
DeactivateStrokeLimit	Bool	false	False
ExecuteFunction	Bool	false	False
RecordNo	Int	0	False
SetFunctionNumber	Int	0	False
SetFunctionGroup	Int	0	False
SetValueVelocity	DInt	0	False
SetValuePosition	DInt	0	False
SetValueForceRamp	DInt	0	False
SetValueForce	DInt	0	False
SetValueRotRamp	DInt	0	False
SetValueRotSpeed	DInt	0	False
▼ Output			
DriveEnabled	Bool	false	False
Ready	Bool	false	False
Warning	Bool	false	False
Fault	Bool	false	False
SupplyVoltPresent	Bool	false	False
ControlFCT_HMI	Bool	false	False
StateOPM	Int	0	False

Name	Data type	Start value	Retain
HaltNotActive	Bool	false	False
AckStart	Bool	false	False
MC	Bool	false	False
AckTeach	Bool	false	False
DriveMoving	Bool	false	False
DragError	Bool	false	False
StandStillControl	Bool	false	False
HomingValid	Bool	false	False
ActualRecordNo	Int	0	False
RC1	Bool	false	False
RCC	Bool	false	False
VelocityLimitReached	Bool	false	False
StrokeLimitReached	Bool	false	False
FunctionIsActive	Bool	false	False
ActualFunctionNumber	Int	0	False
ActualFunctionGroup	Int	0	False
ActualPosition	DInt	0	False
ActualVelocity	DInt	0	False
ActualForce	DInt	0	False
ActualRotRamp	DInt	0	False
ActualRotSpeed	DInt	0	False
▼ InOut			
FML_REF	"DT_FML_REF"		False
▼ Static			
Converted_BYTE5_8_IN	DInt	0	False
Converted_BYTE5_8_OUT	DInt	0	False
OperationModeB1	Bool	false	False
OperationModeB2	Bool	false	False
ControlModeB1	Bool	false	False
ControlModeB2	Bool	false	False
ContinuousMode	Bool	false	False
StateOperationModeB1	Bool	false	False
StateOperationModeB2	Bool	false	False
StateControlModeB1	Bool	false	False
StateControlModeB2	Bool	false	False
StateContinuousMode	Bool	false	False

10. Plant (Motor Control) / Instance DBs / Stepper B Drive Control depreciated

FHPP_DPWR_DAT Stepper B iDB [DB1307]

FHPP_DPWR_DAT Stepper B iDB Properties					
General					
Name	FHPP_DPWR_DAT Stepper B iDB	**Number**	1307	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**	FESTO	**Comment**	
Family	E-DRIVE	**Version**	5.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
HW_IO_FHPP	HW_IO	0	False
HW_IO_FPC	HW_IO	0	False
HW_IO_FHPPplus	HW_IO	0	False
cfgFHPPplusOut	DInt	0	False
▼ Output			
Error	Bool	false	False
Status	"DT_FML_STATUS"		False
▼ InOut			
FML_REF	"DT_FML_REF"		False
▼ Static			
EnableFHPPplusOutOld	DInt	0	False
FHPPplusSize	DInt	0	False
FHPPplusElementSize	Array[1..9] of DInt		False

10. Plant (Motor Control) / Instance DBs / Stepper B Drive Control depreciated

DT_FML_REF Stepper B iDB [DB1301]

DT_FML_REF Stepper B iDB Properties					
General					
Name	DT_FML_REF Stepper B iDB	**Number**	1301	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Static			
InData	Struct		False
OutData	Struct		False
Modbus	Bool	false	False
DeviceType	String[16]	''	False
MemberID	Int	0	False
MemberIDmax	Int	0	False
Done	Bool	false	False
Err	Bool	false	False
PNU	Word	16#0	False
ActPNU	Word	16#0	False
ReqID	Int	0	False
RespID	Int	0	False
FPCC_Modus	Byte	16#0	False
FPCS_Modus	Byte	16#0	False
FPCC_Packet	"DT_FML_PRM_FILE"		False
FPCS_Packet	"DT_FML_PRM_FILE"		False
Subindex	Byte	16#0	False
ActSubindex	Byte	16#0	False
DatatypeWR	Byte	16#0	False
DatatypeRD	Byte	16#0	False
ParamValueWR	DInt	0	False
ParamValueRD	DInt	0	False

11. Reverse Kinematics

FB Simplified Reverse Kinematics [FB320]

FB Simplified Reverse Kinematics Properties					
General					
Name	FB Simplified Reverse Kinematics	**Number**	320	**Type**	FB
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
▼ Input			
Traction Speed A ACT	Real	0.0	Non-retain
Traction Speed B ACT	Real	0.0	Non-retain
Steering Speed A ACT	Real	0.0	Non-retain
Steering Speed B ACT	Real	0.0	Non-retain
ThetaA ACT	Real	0.0	Non-retain
ThetaB ACT	Real	0.0	Non-retain
rA_x	Real	0.0	Non-retain
rB_x	Real	0.0	Non-retain
rA_y	Real	0.0	Non-retain
rB_y	Real	0.0	Non-retain
WheelRadius	Real	0.0	Non-retain
CasterOffset	Real	0.0	Non-retain
▼ Output			
Xv ACT	Real	0.0	Non-retain
Yv ACT	Real	0.0	Non-retain
AGVYawRate ACT	Real	0.0	Non-retain
InOut			
▼ Static			
VA_x ACT	Real	0.0	Non-retain
VA_y ACT	Real	0.0	Non-retain
VB_x ACT	Real	0.0	Non-retain
VB_y ACT	Real	0.0	Non-retain
OmegaAComp	Real	0.0	Non-retain
OmegaBComp	Real	0.0	Non-retain
Temp			
Constant			

Network 1: Unit A actual components

```
//Actual component velocities from angular speeds - unit A
//
    // X Component velocity
    #"VA_x ACT" := (#WheelRadius * COS(#"ThetaA ACT") * #"Traction Speed A ACT")
- (#CasterOffset * SIN(#"ThetaA ACT") * #"Steering Speed A ACT");

    // Y Component velocity
    #"VA_y ACT" := (#WheelRadius * SIN(#"ThetaA ACT") * #"Traction Speed A ACT")
+ (#CasterOffset * COS(#"ThetaA ACT") * #"Steering Speed A ACT");

```

Network 2: Unit B actual components

```
//Actual components from angular speeds - unit B
//
    // X Component velocity
    #"VB_x ACT" := (#WheelRadius * COS(#"ThetaB ACT") * #"Traction Speed B ACT")
- (#CasterOffset * SIN(#"ThetaB ACT") * #"Steering Speed B ACT");

    // Y Component velocity
    #"VB_y ACT" := (#WheelRadius * SIN(#"ThetaB ACT") * #"Traction Speed B ACT")
+ (#CasterOffset * COS(#"ThetaB ACT") * #"Steering Speed B ACT");

```

Network 3: Find the Centrodial Components

```
//Find actual centrodial components from unit components
//
    // X Component
    #"Xv ACT" := #"VA_x ACT" + #"VB_x ACT";

    // Y Component
    #"Yv ACT" := #"VA_y ACT" + #"VB_y ACT";

//Find the body Yaw rate from the unit components
//

    //Yaw rate due to Unit A's influence
    #OmegaAComp := (#"VA_x ACT" / #rA_y) + (#"VA_y ACT" / #rA_x);

    //Yaw rate due to Unit B's influence
    #OmegaBComp := (#"VB_x ACT" / #rB_y) + (#"VB_y ACT" / #rB_x);

    //Sum of the units yaw rate effect on the centroid
    #"AGVYawRate ACT" := #OmegaAComp - #OmegaBComp;



```

11. Reverse Kinematics / Instance DBs

Simplified Reverse Kinematics iDB [DB320]

Simplified Reverse Kinematics iDB Properties					
General					
Name	Simplified Reverse Kinematics iDB	**Number**	320	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
Traction Speed A ACT	Real	0.0	False
Traction Speed B ACT	Real	0.0	False
Steering Speed A ACT	Real	0.0	False
Steering Speed B ACT	Real	0.0	False
ThetaA ACT	Real	0.0	False
ThetaB ACT	Real	0.0	False
rA_x	Real	0.0	False
rB_x	Real	0.0	False
rA_y	Real	0.0	False
rB_y	Real	0.0	False
WheelRadius	Real	0.0	False
CasterOffset	Real	0.0	False
▼ Output			
Xv ACT	Real	0.0	False
Yv ACT	Real	0.0	False
AGVYawRate ACT	Real	0.0	False
InOut			
▼ Static			
VA_x ACT	Real	0.0	False
VA_y ACT	Real	0.0	False
VB_x ACT	Real	0.0	False
VB_y ACT	Real	0.0	False
OmegaAComp	Real	0.0	False
OmegaBComp	Real	0.0	False

12 Intergration

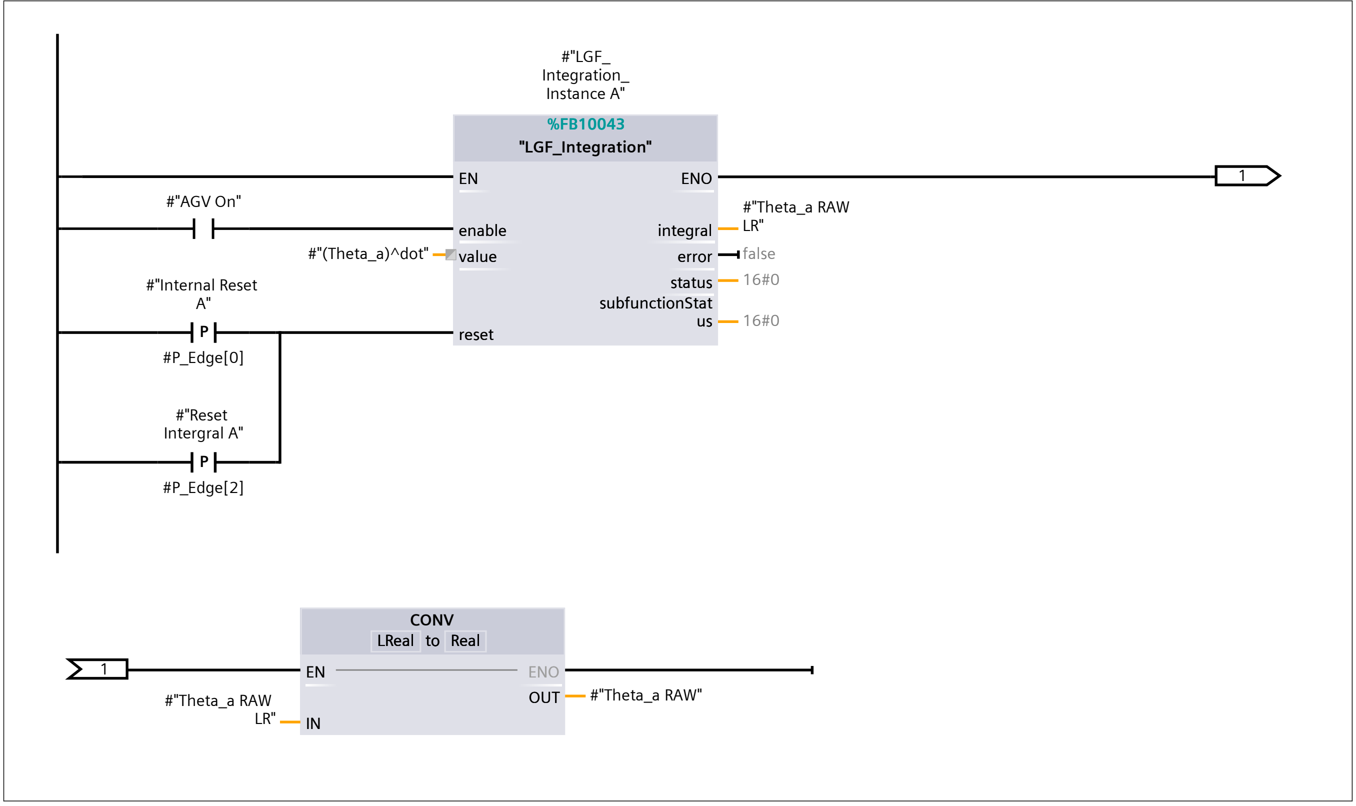
FB Intergrate Unit Angles [FB2]

FB Intergrate Unit Angles Properties					
General					
Name	FB Intergrate Unit Angles	**Number**	2	**Type**	FB
Language	LAD	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

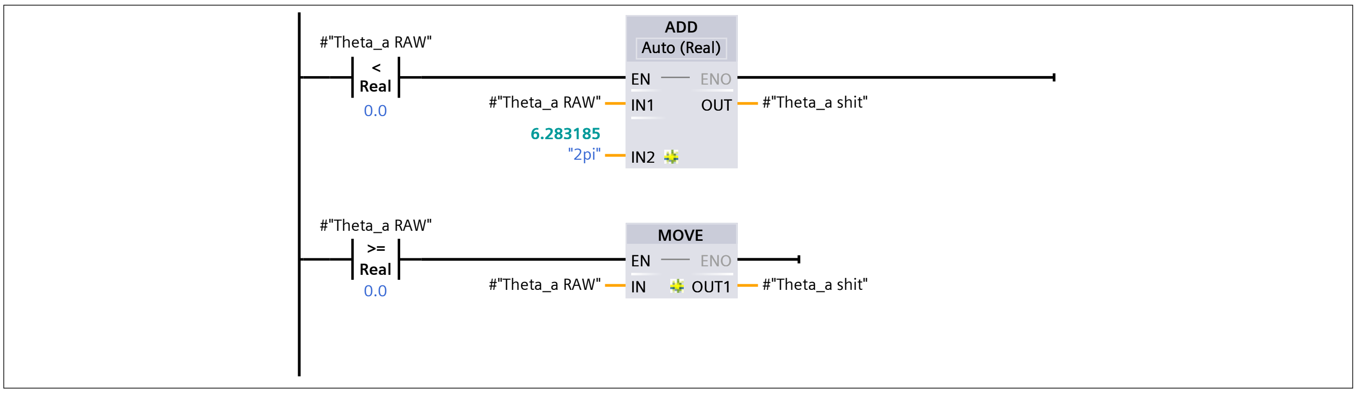
Name	Data type	Default value	Retain
▼ Input			
AGV On	Bool	false	Non-retain
Reset Intergral A	Bool	false	Non-retain
Reset Intergral B	Bool	false	Non-retain
(Theta_a)^dot	Real	0.0	Non-retain
(Theta_b)^dot	Real	0.0	Non-retain
Pot A rad	Real	0.0	Non-retain
Pot B rad	Real	0.0	Non-retain
▼ Output			
Theta_a rad	Real	0.0	Non-retain
Theta_a deg	Real	0.0	Non-retain
Theta_b rad	Real	0.0	Non-retain
Theta_b deg	Real	0.0	Non-retain
InOut			
▼ Static			
Theta_a RAW LR	LReal	0.0	Non-retain
Theta_a RAW	Real	0.0	Non-retain
Theta_b RAW LR	LReal	0.0	Non-retain
Theta_b RAW	Real	0.0	Non-retain
Theta_a shit	Real	0.0	Non-retain
Theta_b shit	Real	0.0	Non-retain
Internal Reset A	Bool	false	Non-retain
Internal Reset B	Bool	false	Non-retain
LGF_Integration_Instance A	"LGF_Integration"		
LGF_Integration_Instance B	"LGF_Integration"		
P_Edge	Array[0..3] of Bool		Non-retain
Temp			
▼ Constant			
Theta_a rad fuzzyfactor	Real	0.174533	
Pot a fuzzyfactor	Real	0.0174533	
Theta_b rad fuzzyfactor	Real	0.174533	
Pot b fuzzyfactor	Real	0.0174533	

Network 1: Intergrate unit A speed to get unit A position

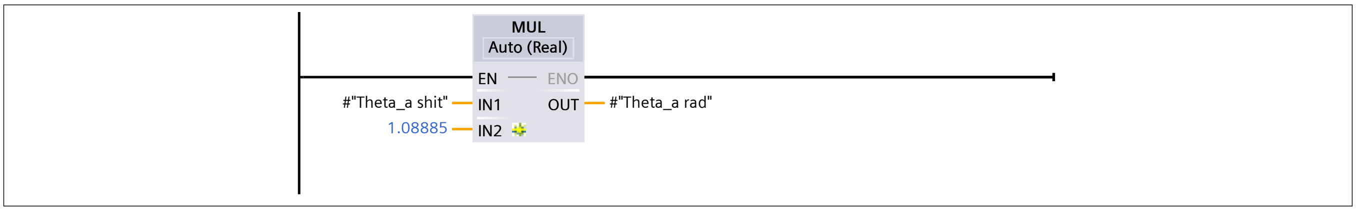
Network 1: Intergrate unit A speed to get unit A position



Network 2: If negative = map to positive spectrum unit A

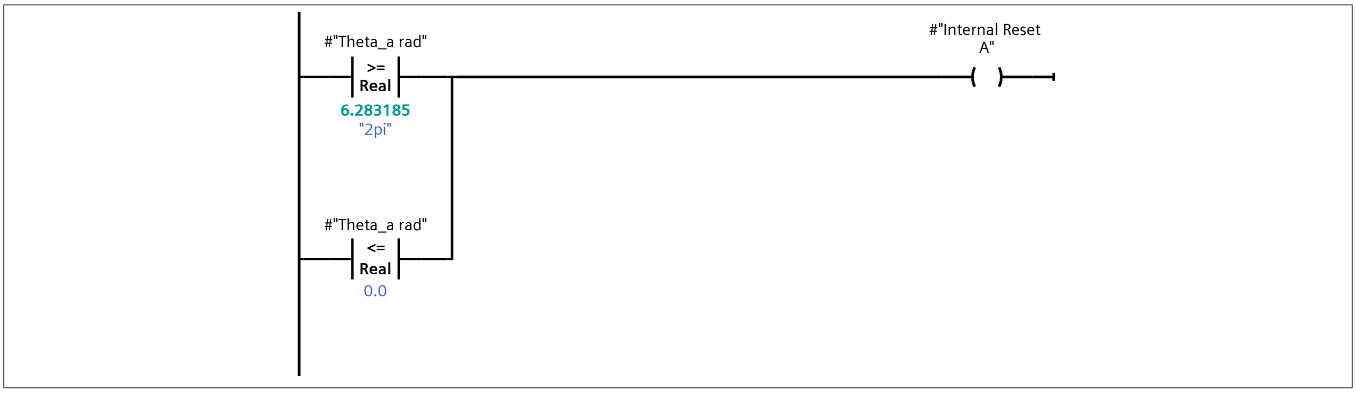


Network 3: Scale for non-linear behaviour of shit gear



Network 4: Reset when full rotation i.e. 360 = 0

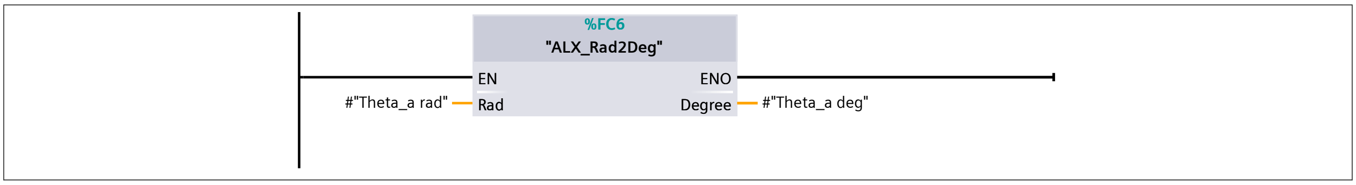
This has depreciated as network 5 presents a better reset system.



Network 5: Unit A Reset intergral using potentiometer value

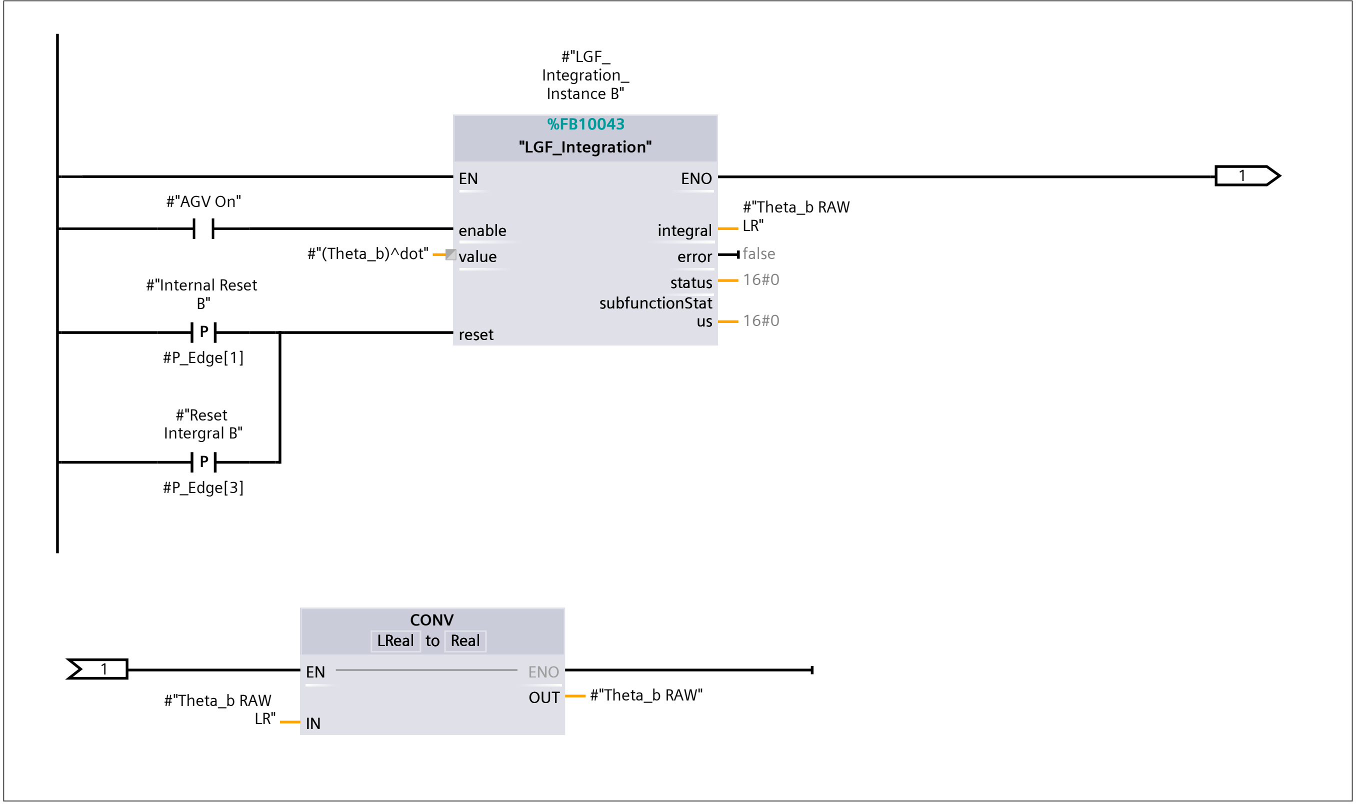
```
// Resetting the intergrated value only using the intergrated value, i.e. when intergrated value >= 360 deg then reset to zero
// or when the intergral becomes lower than 0 degrees having done at least one CW rotation is dangerous since accumulated errors will begin to stack.
//
// 1) Its a better idea to use the potentiometer to reset the intergrator. I.e. when the pot value is zero reset the intergrator
// 2) Also only allow the pot to reset the intergratoe if the intergrated value is near 360 or 0 degrees. This will prevent the issue where deadzones
// on the pot are erroneously able to reset the intergrator (this is especially true with unit A's pot
// 3) If an AGV drive unit has been disassembled please ensure the deadzone is not near the zero point by physically rotating the pot
//

IF "AlwaysFALSE" = TRUE THEN //Disable as strange results

    //Reset angle intergrator for Unit A
    //
    IF #"Theta_a rad" >= ("2pi" - #"Theta_a rad fuzzyfactor") THEN
        // intergrator is measuring a value > (360-fuzzyfactor), thus if pot triggers zero it is likly a true zero and not a deadzone
        //
        IF (#"Pot A rad" < #"Pot a fuzzyfactor") OR (#"Pot A rad" > ("2pi" - #"Pot a fuzzyfactor")) THEN
            // physcial pot is between "pot fuzzy factor" and 360-"pot fuzzy factor" (default < 1 deg and > 359 deg), thus pot is very close to zero
            // fuzzy factor needed here as an absolute zero value comming from a pot is extremely unlikly.
            #"Internal Reset A" := TRUE;
        ELSE
            // intergrator in reset zone but pot not near zero
            #"Internal Reset A" := FALSE;
        END_IF;

    ELSIF #"Theta_a rad" <= (0.0 + #"Theta_a rad fuzzyfactor") THEN
        // intergrator is measuring a value < (0+fuzzyfactor), thus if pot triggers zero it is likly a true zero and not a deadzone
        //
        IF (#"Pot A rad" < #"Pot a fuzzyfactor") OR (#"Pot A rad" > ("2pi" - #"Pot a fuzzyfactor")) THEN
            // physcial pot is between "pot fuzzy factor" and 360-"pot fuzzy factor" (default < 1 deg and > 359 deg), thus pot is very close to zero
            // fuzzy factor needed here as an absolute zero value comming from a pot is extremely unlikly.
            #"Internal Reset A" := TRUE;
        ELSE
            // intergrator in reset zone but pot not near zero
            #"Internal Reset A" := FALSE;
        END_IF;

    ELSE
        // Do nothing as intergrator not near zero zone or pot zero
        #"Internal Reset A" := FALSE;
    END_IF;

END_IF;

```

Network 6: Convert angle to degrees unit a

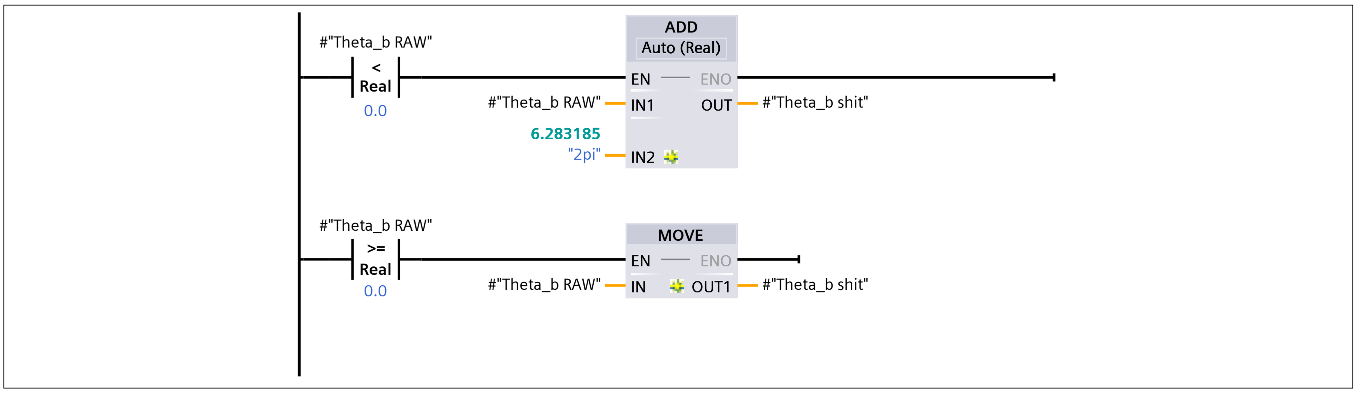


Network 7: Intergrate unit B speed to get unit B position

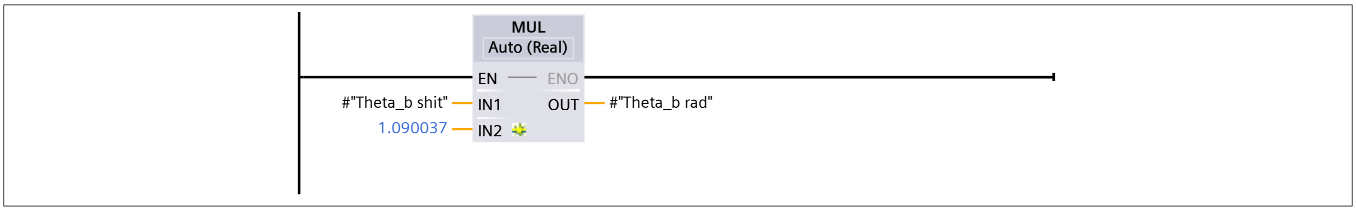
Network 7: Intergrate unit B speed to get unit B position



Network 8: If negative = map to positive spectrum unit B

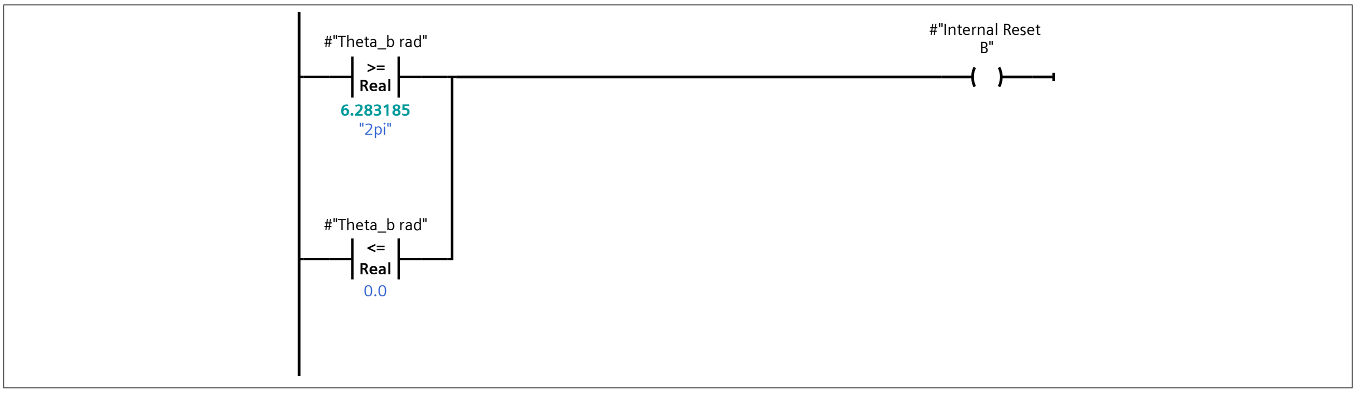


Network 9: Scale for non-linear behaviour of shit gear



Network 10: Rest when full rotation i.e. 360 = 0

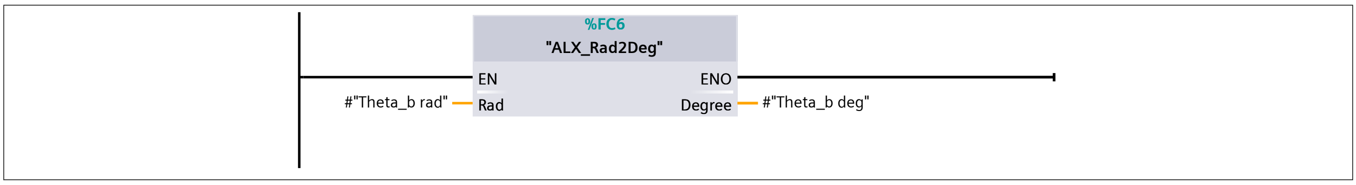
This has depreciated as network 11 presents a better reset system.



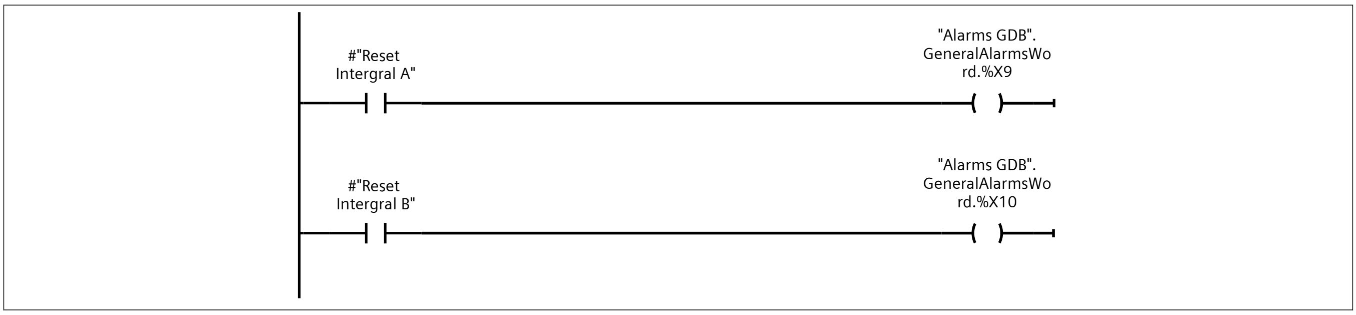
Network 11: Unit B Reset intergral using potentiometer value

```
// Resetting the intergrated value only using the intergrated value, i.e. when intergrated value >= 360 deg then reset to zero
// or when the intergral becomes lower than 0 degrees having done at least one CW rotation is dangerous since accumulated errors will begin to stack.
//
// 1) Its a better idea to use the potentiometer to reset the intergrator. I.e. when the pot value is zero reset the intergrator
// 2) Also only allow the pot to reset the intergratoe if the intergrated value is near 360 or 0 degrees. This will prevent the issue where deadzones
// on the pot are erroneously able to reset the intergrator (this is especially true with unit A's pot
// 3) If an AGV drive unit has been disassembled please ensure the deadzone is not near the zero point by physically rotating the pot
//

IF "AlwaysFALSE" = TRUE THEN

    //Reset angle intergrator for Unit A
    //
    IF #"Theta_b rad" >= ("2pi" - #"Theta_b rad fuzzyfactor") THEN
        // intergrator is measuring a value > (360-fuzzyfactor), thus if pot triggers zero it is likly a true zero and not a deadzone
        //
        IF (#"Pot B rad" < #"Pot b fuzzyfactor") OR (#"Pot B rad" > ("2pi" - #"Pot b fuzzyfactor")) THEN
            // physcial pot is between "pot fuzzy factor" and 360-"pot fuzzy factor" (default < 1 deg and > 359 deg), thus pot is very close to zero
            // fuzzy factor needed here as an absolute zero value comming from a pot is extremely unlikly.
            #"Internal Reset B" := TRUE;
        ELSE
            // intergrator in reset zone but pot not near zero
            #"Internal Reset B" := FALSE;
        END_IF;

    ELSIF #"Theta_b rad" <= (0.0 + #"Theta_b rad fuzzyfactor") THEN
        // intergrator is measuring a value < (0+fuzzyfactor), thus if pot triggers zero it is likly a true zero and not a deadzone
        //
        IF (#"Pot B rad" < #"Pot b fuzzyfactor") OR (#"Pot B rad" > ("2pi" - #"Pot b fuzzyfactor")) THEN
            // physcial pot is between "pot fuzzy factor" and 360-"pot fuzzy factor" (default < 1 deg and > 359 deg), thus pot is very close to zero
            // fuzzy factor needed here as an absolute zero value comming from a pot is extremely unlikly.
            #"Internal Reset B" := TRUE;
        ELSE
            // // intergrator in reset zone but pot not near zero
            #"Internal Reset B" := FALSE;
        END_IF;

    ELSE
        // Do nothing as intergrator not near zero zone or pot zero
        #"Internal Reset B" := FALSE;
    END_IF;

END_IF;


```

Network 12: Convert angle to degrees unit a



Network 13: Alarms



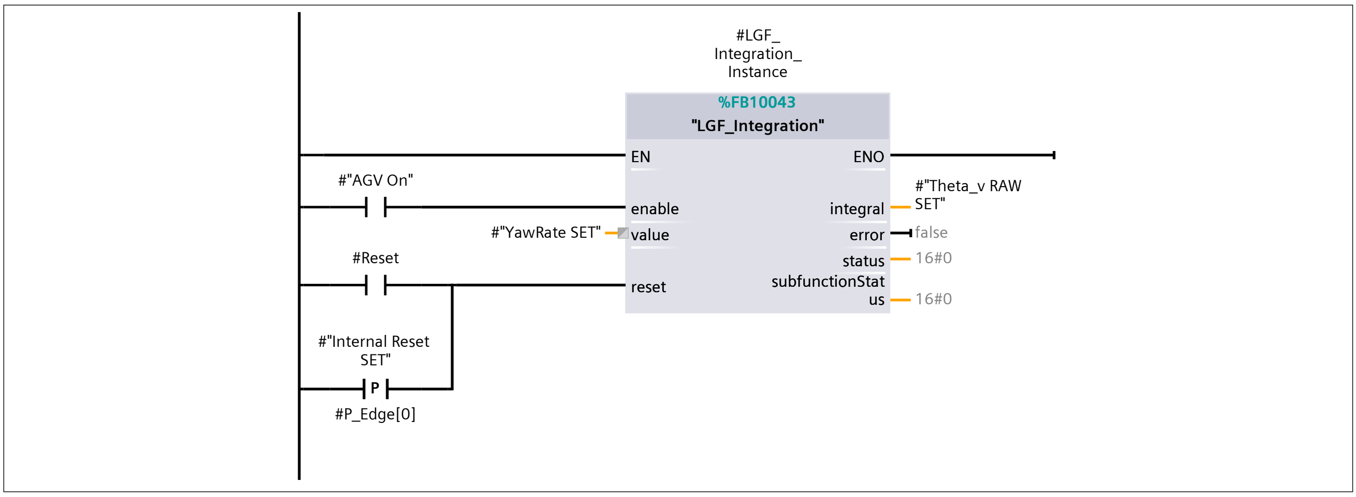
12 Intergration

FB Intergrate Body Yaw [FB3]

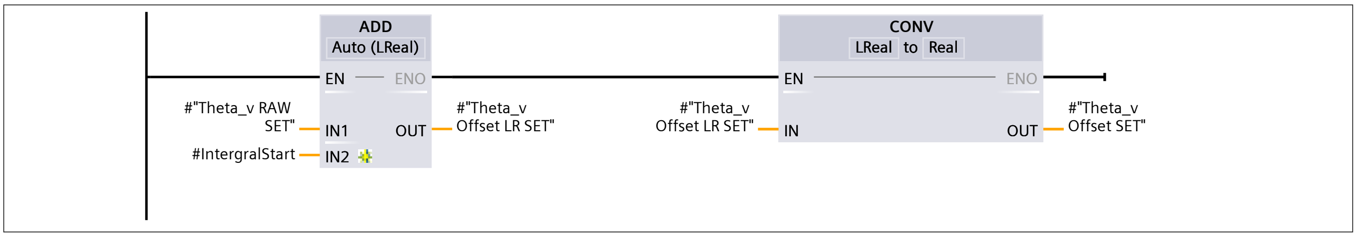
FB Intergrate Body Yaw Properties					
General					
Name	FB Intergrate Body Yaw	**Number**	3	**Type**	FB
Language	LAD	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
▼ Input			
AGV On	Bool	false	Non-retain
Reset	Bool	false	Non-retain
YawRate SET	Real	0.0	Non-retain
YawRate ACT	Real	0.0	Non-retain
IntergralStart	LReal	0.0	Non-retain
Noise Threshold	Real	0.001	Non-retain
▼ Output			
Theta_v rad SET	Real	0.0	Non-retain
Theta_v rad ACT	Real	0.0	Non-retain
Theta_v deg SET	Real	0.0	Non-retain
Theta_v deg ACT	Real	0.0	Non-retain
InOut			
▼ Static			
LGF_Integration_Instance	"LGF_Integration"		
Theta_v RAW SET	LReal	0.0	Non-retain
Theta_v Offset LR SET	LReal	0.0	Non-retain
Theta_v Offset SET	Real	0.0	Non-retain
Internal Reset SET	Bool	false	Non-retain
LGF_Integration_Instance_1	"LGF_Integration"		
Theta_v RAW Buffer	LReal	0.0	Non-retain
Theta_v RAW ACT	LReal	0.0	Non-retain
Theta_v Offset LR ACT	LReal	0.0	Non-retain
Theta_v Offset ACT	Real	0.0	Non-retain
Internal Reset ACT	Bool	false	Non-retain
P_Edge	Array[0..2] of Bool		Non-retain
Temp			
Constant			

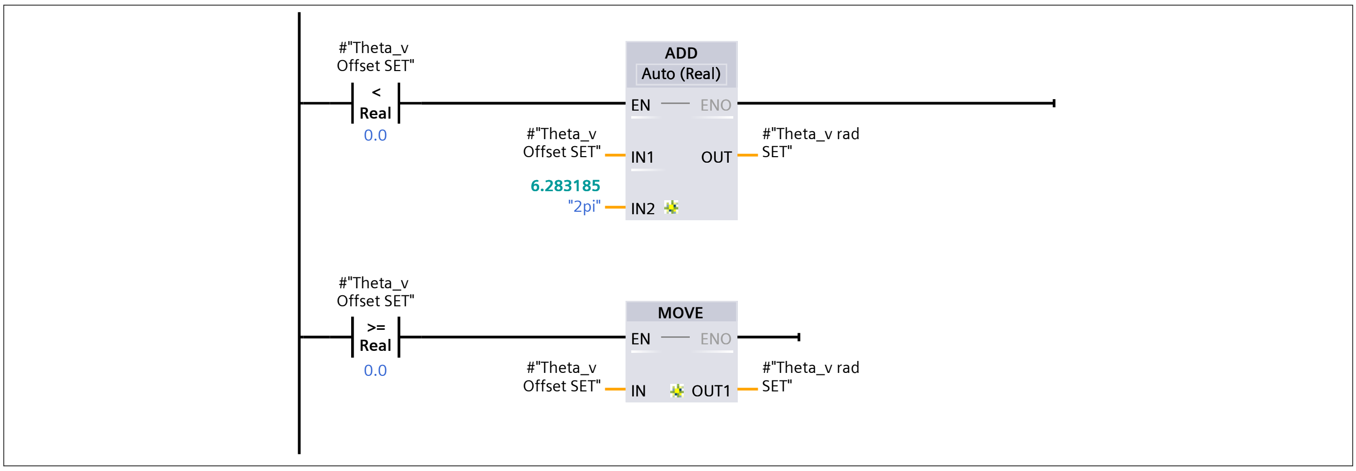
Network 1: SETPOINT: Intergrate the desired yaw rate to get angle SET



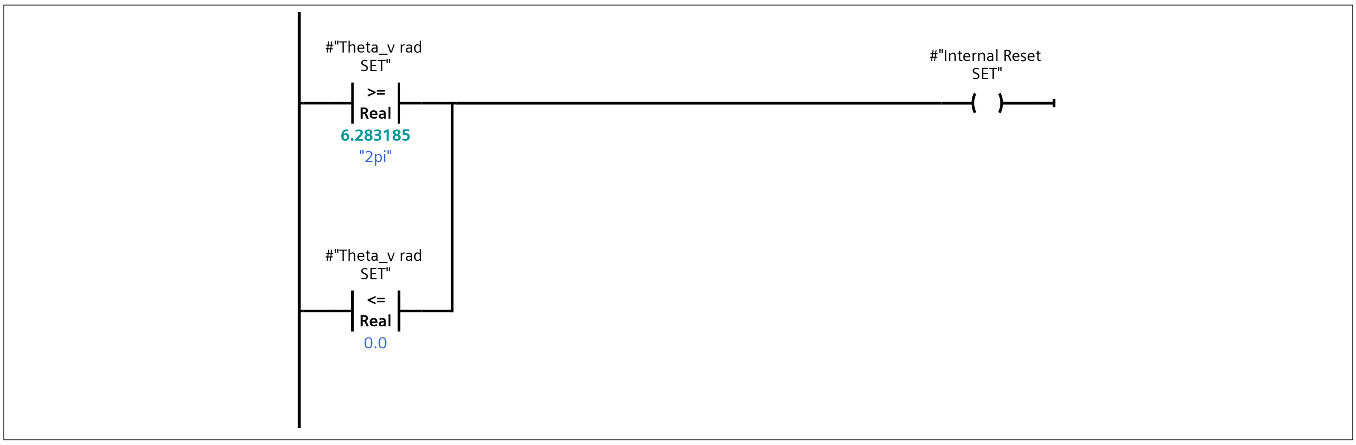
Network 2: SETPOINT: Add an offset if there is one (in automatic mode this will likely be the current univesal angle)



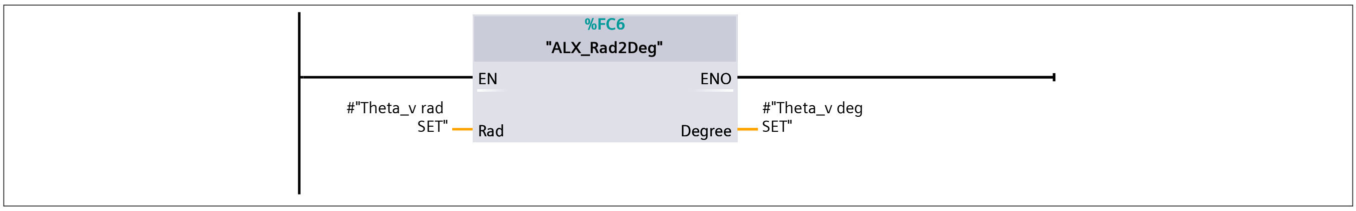
Network 3: SETPOINT: If negative = map value onto the positive spectrum



Network 4: SETPOINT: Reset at full revolution i.e. 360 deg == 0 deg

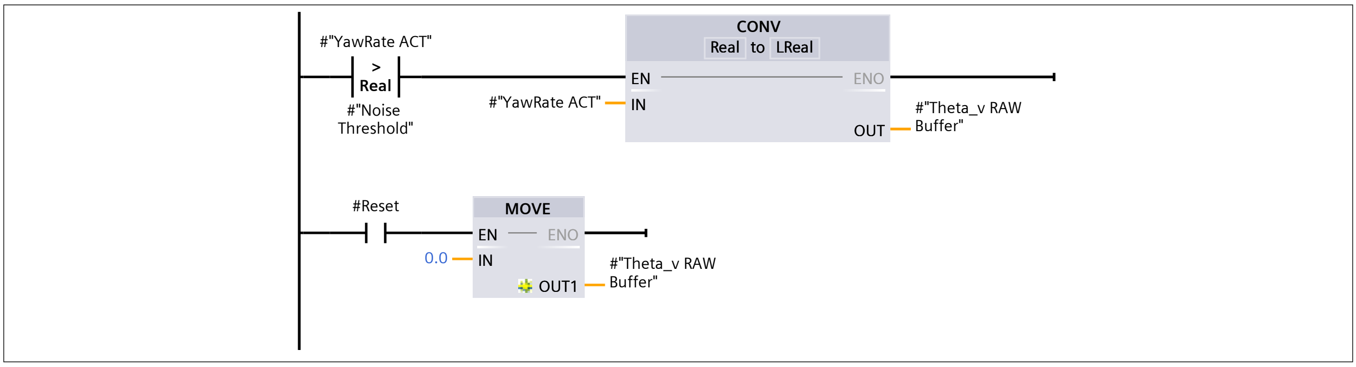


Network 5: SETPOINT: Convert Radians to degrees

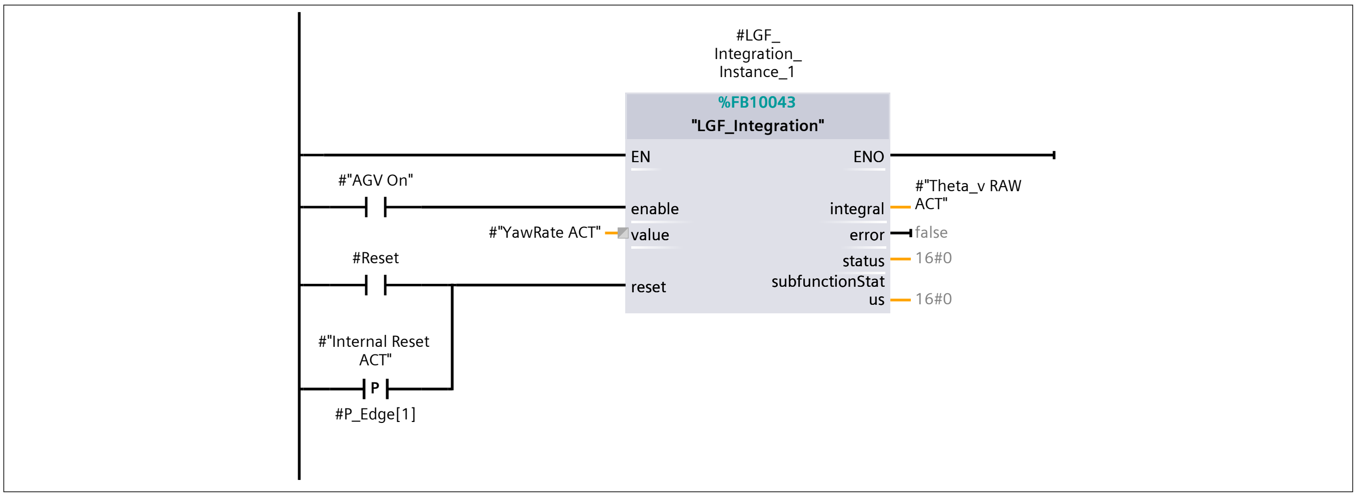


Network 6: ACTUAL: Noise Rejection

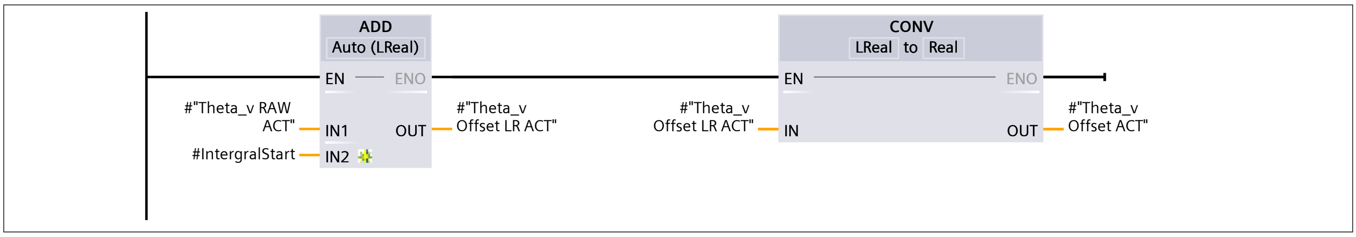
Since there is a small amount of noise on the input the noise threshold removesit when the system is at standstill



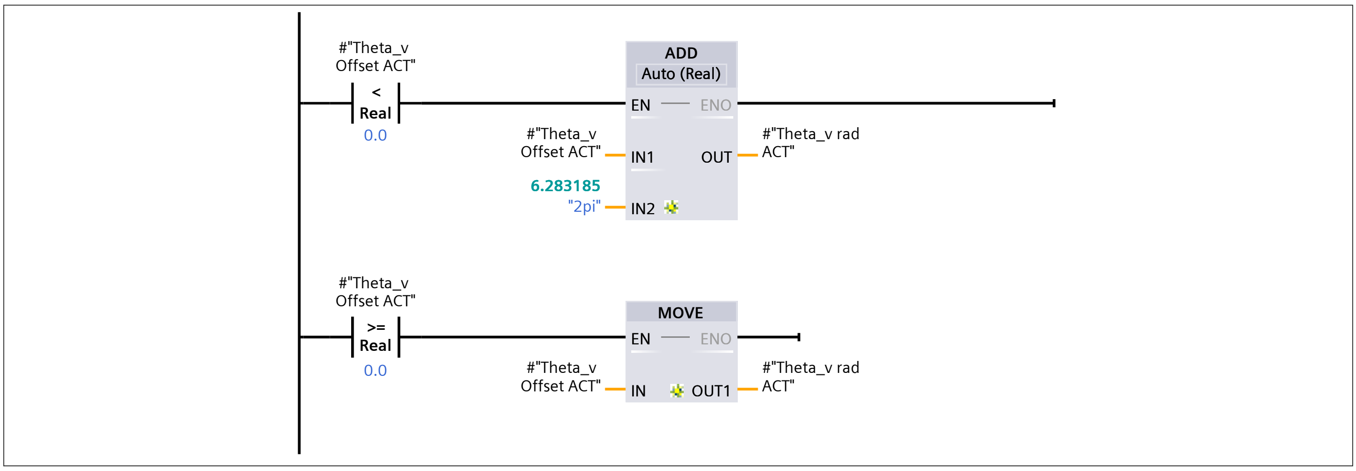
Network 7: ACT UAL: Intergrate the desired yaw rate to get angle



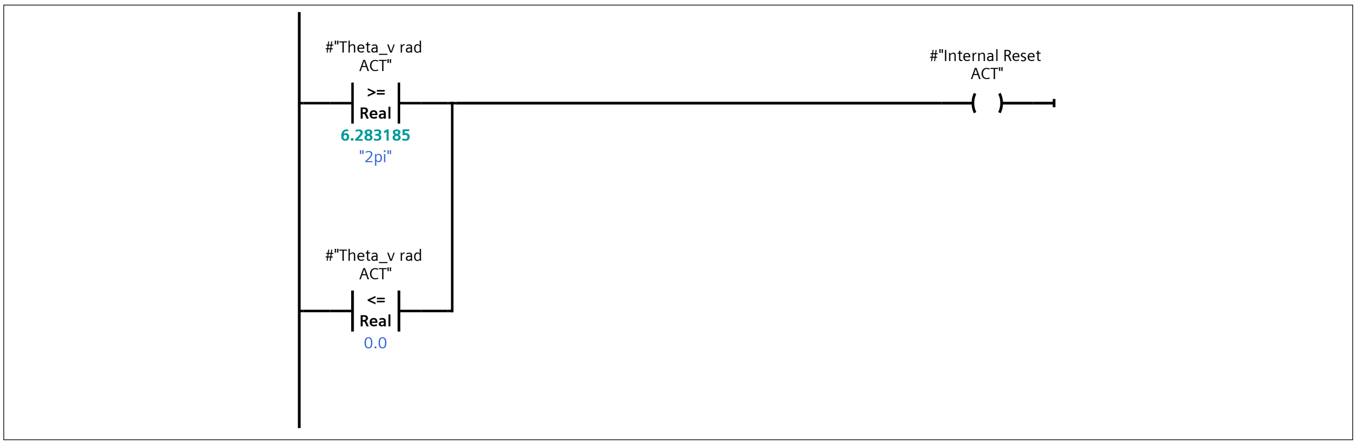
Network 8: ACTUAL: Add an offset if there is one (in automatic mode this will likely be the current univesal angle)



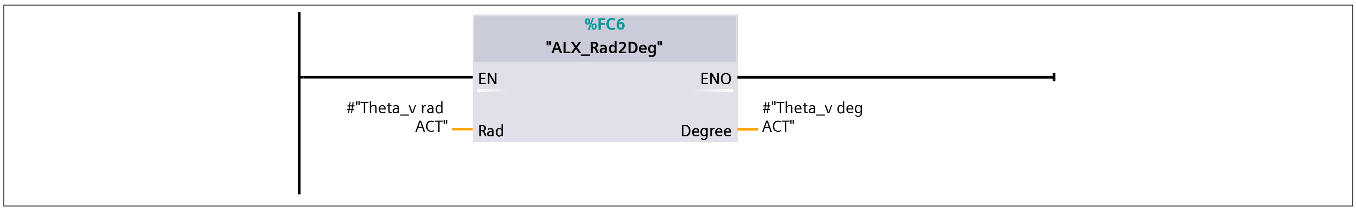
Network 9: ACTUAL: If negative = map value onto the positive spectrum



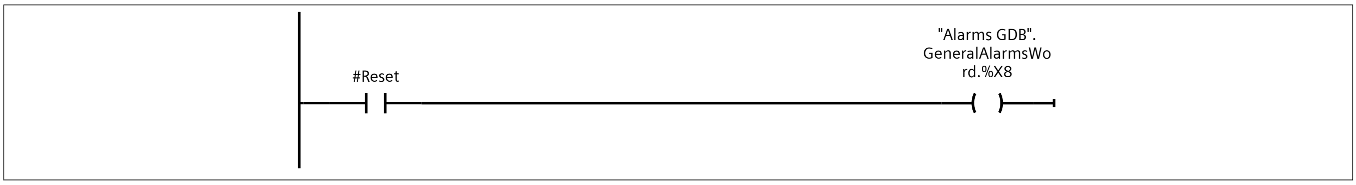
Network 10: ACTUAL: Reset at full revolution i.e. 360 deg == 0 deg



Network 11: ACTUAL: Convert Radians to degrees



Network 12: Alarms



12 Intergration / Instance DBs

Intergrate Angles iDB [DB4]

Intergrate Angles iDB Properties					
General					
Name	Intergrate Angles iDB	**Number**	4	**Type**	DB
Language	DB	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
AGV On	Bool	false	False
Reset Intergral A	Bool	false	False
Reset Intergral B	Bool	false	False
(Theta_a)^dot	Real	0.0	False
(Theta_b)^dot	Real	0.0	False
Pot A rad	Real	0.0	False
Pot B rad	Real	0.0	False
▼ Output			
Theta_a rad	Real	0.0	False
Theta_a deg	Real	0.0	False
Theta_b rad	Real	0.0	False
Theta_b deg	Real	0.0	False
InOut			
▼ Static			
Theta_a RAW LR	LReal	0.0	False
Theta_a RAW	Real	0.0	False
Theta_b RAW LR	LReal	0.0	False
Theta_b RAW	Real	0.0	False
Theta_a shit	Real	0.0	False
Theta_b shit	Real	0.0	False
Internal Reset A	Bool	false	False
Internal Reset B	Bool	false	False
LGF_Integration_Instance A	"LGF_Integration"		False
LGF_Integration_Instance B	"LGF_Integration"		False
P_Edge	Array[0..3] of Bool		False

12 Intergration / Instance DBs

Intergrate Body Yaw iDB [DB2]

Intergrate Body Yaw iDB Properties					
General					
Name	Intergrate Body Yaw iDB	**Number**	2	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
AGV On	Bool	false	False
Reset	Bool	false	False
YawRate SET	Real	0.0	False
YawRate ACT	Real	0.0	False
IntergralStart	LReal	0.0	False
Noise Threshold	Real	0.001	False
▼ Output			
Theta_v rad SET	Real	0.0	False
Theta_v rad ACT	Real	0.0	False
Theta_v deg SET	Real	0.0	False
Theta_v deg ACT	Real	0.0	False
InOut			
▼ Static			
LGF_Integration_Instance	"LGF_Integration"		False
Theta_v RAW SET	LReal	0.0	False
Theta_v Offset LR SET	LReal	0.0	False
Theta_v Offset SET	Real	0.0	False
Internal Reset SET	Bool	false	False
LGF_Integration_Instance_1	"LGF_Integration"		False
Theta_v RAW Buffer	LReal	0.0	False
Theta_v RAW ACT	LReal	0.0	False
Theta_v Offset LR ACT	LReal	0.0	False
Theta_v Offset ACT	Real	0.0	False
Internal Reset ACT	Bool	false	False
P_Edge	Array[0..2] of Bool		False

13. HMI Control

FC HMI Outputs [FC55]

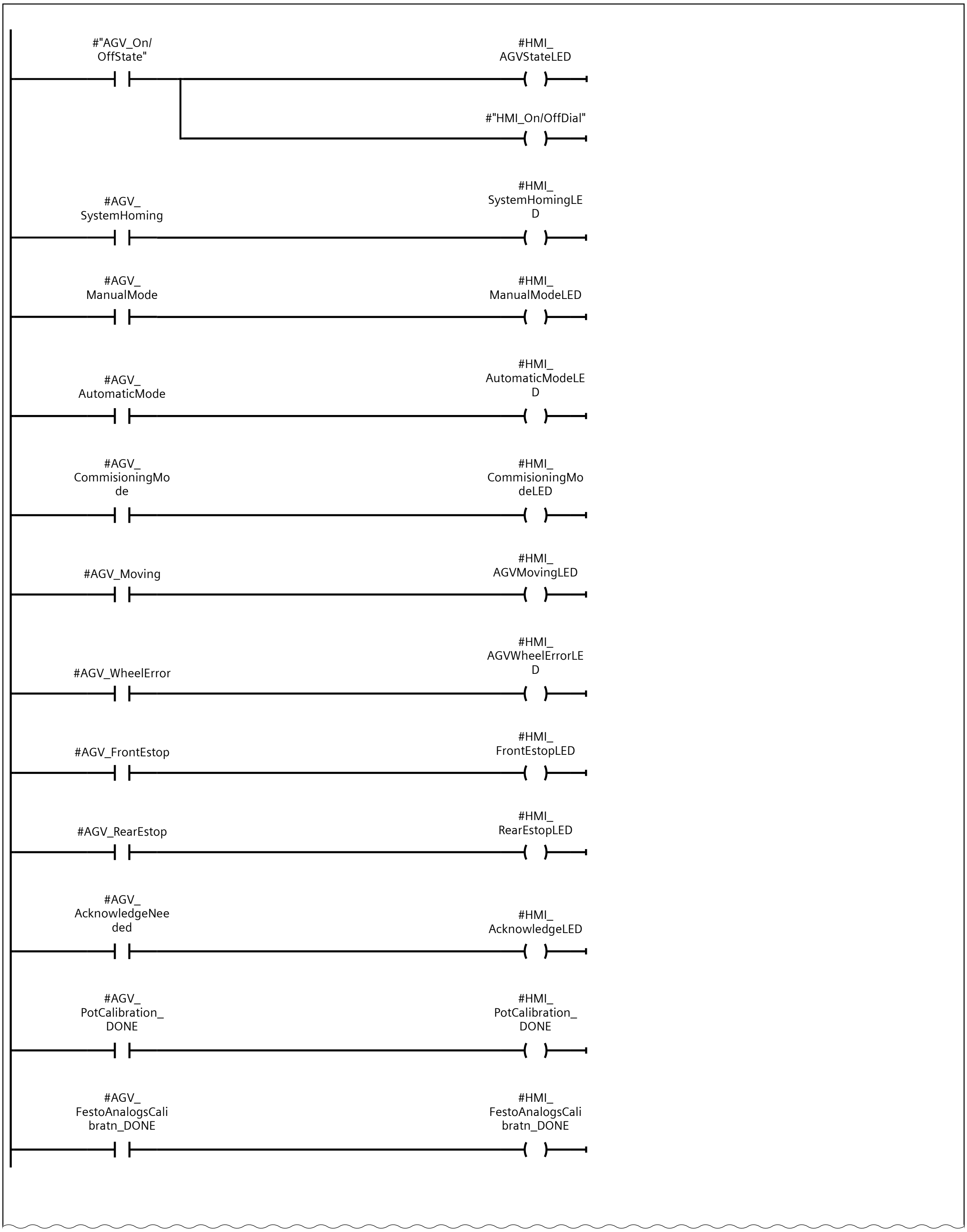
FC HMI Outputs Properties					
General					
Name	FC HMI Outputs	**Number**	55	**Type**	FC
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value
▼ Input		
AGV_On/OffState	Bool	
AGV_SystemHoming	Bool	
AGV_ManualMode	Bool	
AGV_AutomaticMode	Bool	
AGV_CommisioningMode	Bool	
AGV_Moving	Bool	
AGV_WheelError	Bool	
AGV_FrontEstop	Bool	
AGV_RearEstop	Bool	
AGV_AcknowledgeNeeded	Bool	
AGV_SteeringAngleA	Real	
AGV_SteeringAngleB	Real	
AGV_YawAngleACT	Real	
AGV_YawAngleSET	Real	
AGV_XCompVel	Real	
AGV_YCompVel	Real	
AGV_SteeringRPM_A	Real	
AGV_SteeringRPM_B	Real	
AGV_TractionRPM_A	Real	
AGV_TractionRPM_B	Real	
AGV_PotCalibration_DONE	Bool	
AGV_FestoAnalogsCalibratn_DONE	Bool	
AGV_ShutdownRouter	Bool	
AGV_ShutdownIPC	Bool	
AGV_GenTestBusy	Bool	
AGV_TestBufferSize	Int	
AGV_TestBufferSaturated	Bool	
AGV_RunTestLatch	Bool	
AGV_RunTestDone	Bool	
AGV_RunTestIndex	Int	
▼ Output		
HMI_AGVStateLED	Bool	
HMI_On/OffDial	Bool	
HMI_SystemHomingLED	Bool	

Name	Data type	Default value
HMI_ManualModeLED	Bool	
HMI_AutomaticModeLED	Bool	
HMI_CommisioningModeLED	Bool	
HMI_AGVMovingLED	Bool	
HMI_AGVWheelErrorLED	Bool	
HMI_FrontEstopLED	Bool	
HMI_RearEstopLED	Bool	
HMI_AcknowledgeLED	Bool	
HMI_SteeringAngleA_Dial	Real	
HMI_SteeringAngleA	Real	
HMI_SteeringAngleB_Dial	Real	
HMI_SteeringAngleB	Real	
HMI_YawAngleACT_Dial	Real	
HMI_YawAngleACT	Real	
HMI_YawAngleSET_Dial	Real	
HMI_YawAngleSET	Real	
HMI_XCompVel	Real	
HMI_YCompVel	Real	
HMI_SteeringRPM_A	Real	
HMI_SteeringRPM_B	Real	
HMI_TractionRPM_A	Real	
HMI_TractionRPM_B	Real	
HMI_PotCalibration_DONE	Bool	
HMI_FestoAnalogsCalibratn_DONE	Bool	
HMI_ShutdownRouter	Bool	
HMI_ShutdownIPC	Bool	
HMI_GenerateTestWait4Run	Bool	
HMI_GenTestBusy	Bool	
HMI_TestBufferSize	Int	
HMI_GenTestReady	Bool	
HMI_TestBufferSaturated	Bool	
HMI_RunTestInProg	Bool	
HMI_Run%Complete	Int	
▼ InOut		
P_Edge	Array[0..9] of Bool	
▼ Temp		
Wait4RunLink	Bool	
RunTest%Comp_Temp	Real	
Constant		
▼ Return		
FC HMI Outputs	Void	

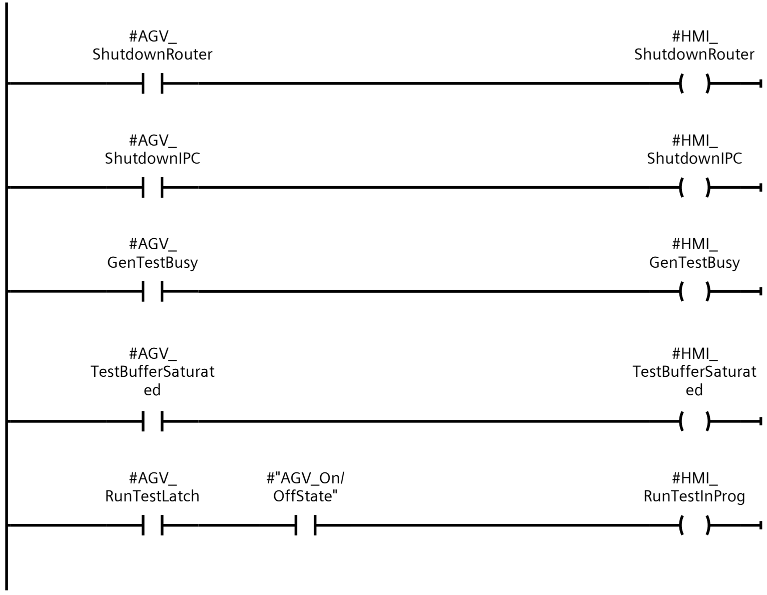
Network 1: Booleans AGV ---> HMI

Network 1: Booleans AGV ---> HMI (1.1 / 2.1)



Network 1: Booleans AGV ---> HMI (2.1 / 2.1)

1.1 (Page2 - 3)

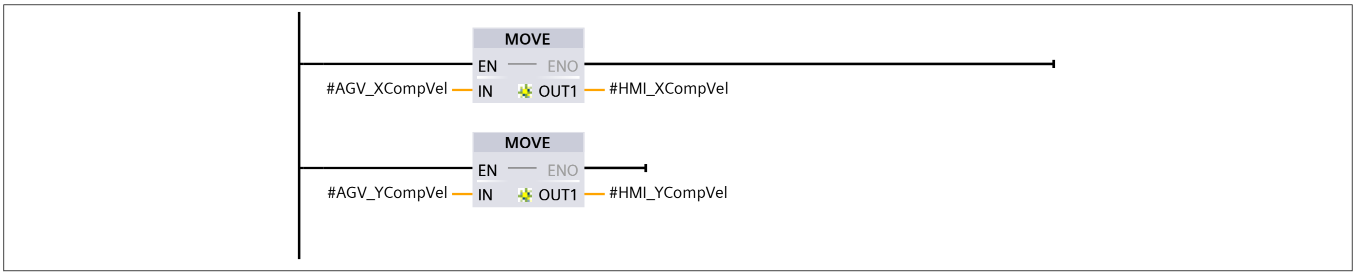


Network 2: SteeringAngles AGV ---> HMI

```
//Pass steering angles to HMI (degrees)
    //
    #HMI_SteeringAngleA := #AGV_SteeringAngleA;
    #HMI_SteeringAngleB := #AGV_SteeringAngleB;
    #HMI_YawAngleACT := #AGV_YawAngleACT; //reference to the universal frame ACTUAL VALUE
    #HMI_YawAngleSET := #AGV_YawAngleSET; //reference to the universal frame SETPOINT VALUE

// Generate counterclockwise angles to work with stupid Siemens dials that can only rotate CW and not CCW
    //
    #HMI_SteeringAngleA_Dial := -1*#AGV_SteeringAngleA;
    #HMI_SteeringAngleB_Dial := -1*#AGV_SteeringAngleB;
    #HMI_YawAngleACT_Dial := -1*#AGV_YawAngleACT;   //reference to the universal frame ACTUAL VALUE
    #HMI_YawAngleSET_Dial := -1 * #AGV_YawAngleSET; //reference to the universal frame SETPOINT VALUE

```

Network 3: Centroid X and Y component velocities (vehicle frame)

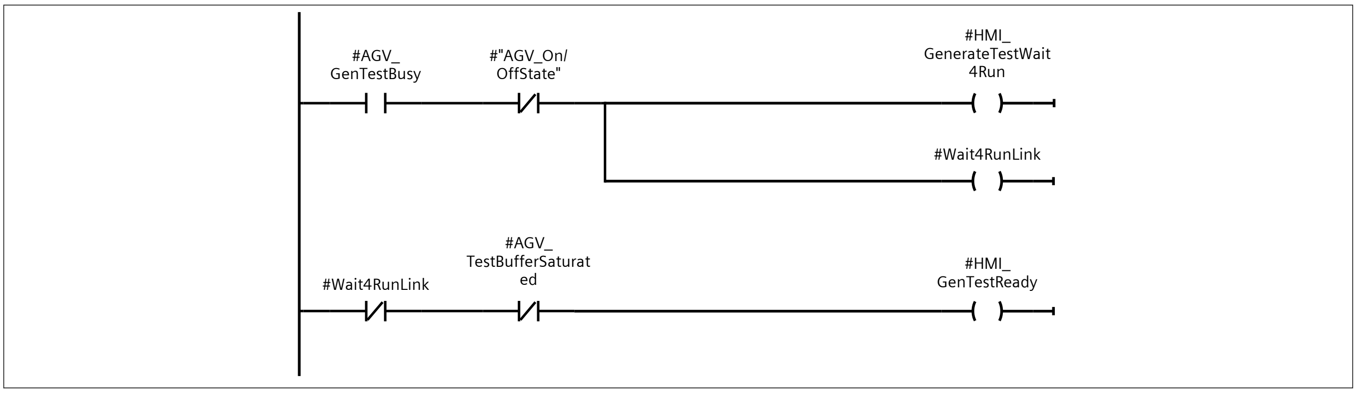
This is with reference to the AGV's body and not the universal frame



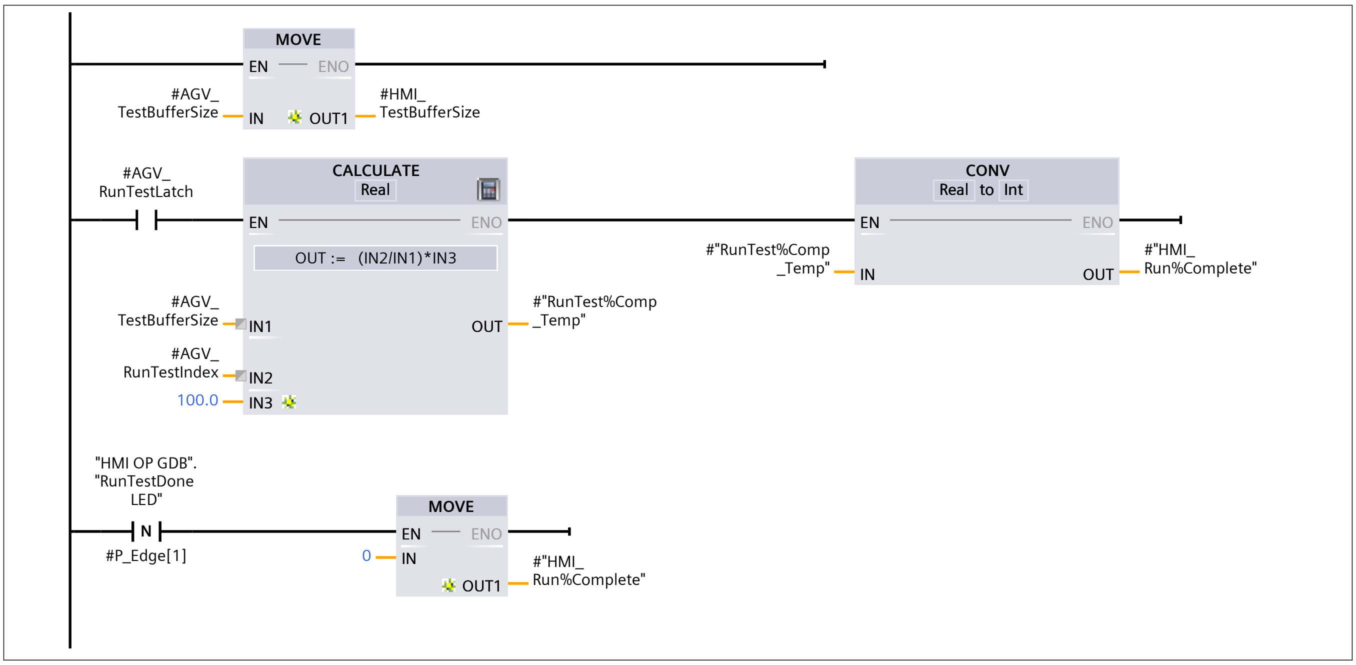
Network 4: RPM values of the motors

```
//Send Real world RPM values to HMI
    //
    #HMI_SteeringRPM_A := #AGV_SteeringRPM_A;
    #HMI_SteeringRPM_B := #AGV_SteeringRPM_B;
    #HMI_TractionRPM_A := #AGV_TractionRPM_A;
    #HMI_TractionRPM_B := #AGV_TractionRPM_B;

```

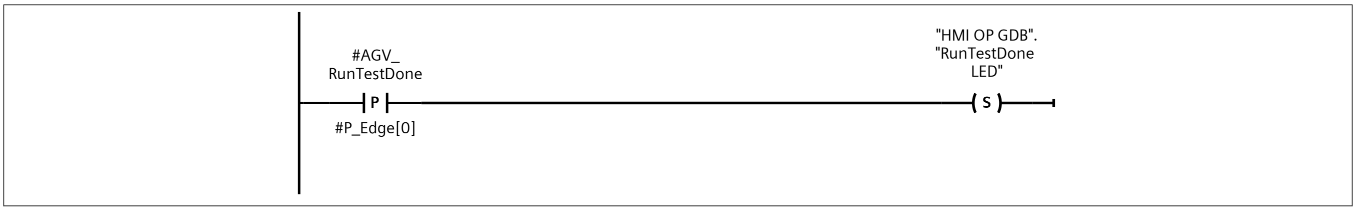
Network 5: Generate test wait for run indicator



Network 6: Move digital data



Network 7: Test Done Latch



13. HMI Control

FC HMI Inputs [FC54]

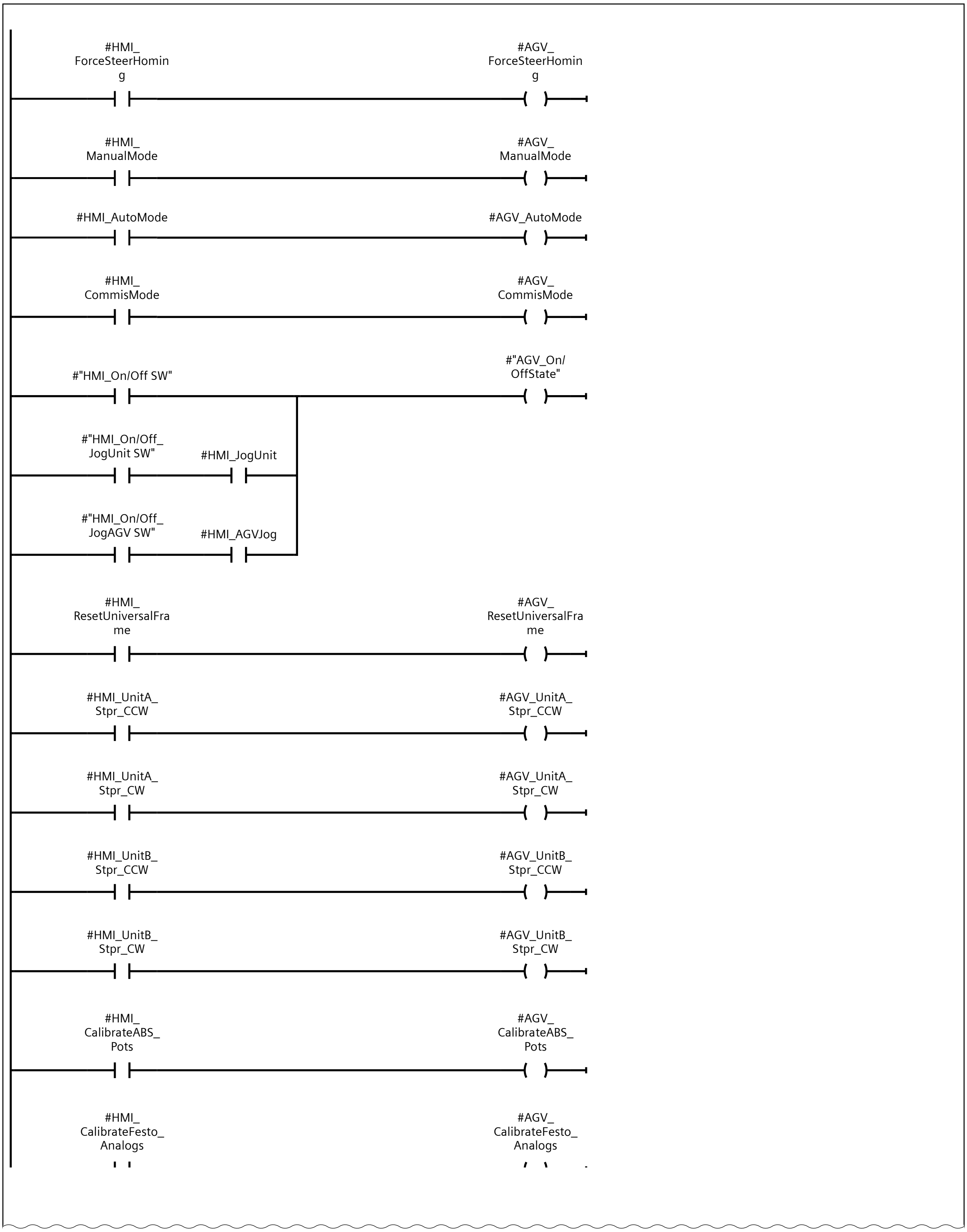
FC HMI Inputs Properties					
General					
Name	FC HMI Inputs	**Number**	54	**Type**	FC
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value
▼ Input		
HMI_On/Off SW	Bool	
HMI_On/Off_JogUnit SW	Bool	
HMI_On/Off_JogAGV SW	Bool	
HMI_ManualMode	Bool	
HMI_AutoMode	Bool	
HMI_CommisMode	Bool	
HMI_ForceSteerHoming	Bool	
HMI_ResetUniversalFrame	Bool	
HMI_UnitA_Stpr_CCW	Bool	
HMI_UnitA_Stpr_CW	Bool	
HMI_UnitB_Stpr_CCW	Bool	
HMI_UnitB_Stpr_CW	Bool	
HMI_UnitA_Servo_CCW	Bool	
HMI_UnitA_Servo_CW	Bool	
HMI_UnitB_Servo_CCW	Bool	
HMI_UnitB_Servo_CW	Bool	
HMI_Stepper_Spd	Int	
HMI_CalibrateABS_Pots	Bool	
HMI_CalibrateFesto_Analogs	Bool	
HMI_AGVJog	Bool	
HMI_AGVJog_Execute	Bool	
HMI_BodJog_Steer	Int	
HMI_BodyJog_Strafe	Int	
HMI_BodyJog_Spd	Int	
HMI_Home2Zero	Bool	
HMI_Hom2Angle	Bool	
HMI_AllowAlignmt	Bool	
HMI_JogUnit	Bool	
HMI_RunTestON	Bool	
HMI_RunTestAbort	Bool	
▼ Output		
AGV_On/OffState	Bool	
AGV_ManualMode	Bool	
AGV_AutoMode	Bool	

Name	Data type	Default value
AGV_CommisMode	Bool	
AGV_ForceSteerHoming	Bool	
AGV_ResetUniversalFrame	Bool	
AGV_UnitA_Stpr_CCW	Bool	
AGV_UnitA_Stpr_CW	Bool	
AGV_UnitB_Stpr_CCW	Bool	
AGV_UnitB_Stpr_CW	Bool	
AGV_UnitA_Servo_CCW	Bool	
AGV_UnitA_Servo_CW	Bool	
AGV_UnitB_Servo_CCW	Bool	
AGV_UnitB_Servo_CW	Bool	
AGV_Stepper_Spd	Real	
AGV_CalibrateABS_Pots	Bool	
AGV_CalibrateFesto_Analogs	Bool	
AGV_AGVJog	Bool	
HMI_AGVJog_FDBCK	Bool	
AGV_AGVJog_Execute	Bool	
AGV_BodJog_Steer	Real	
AGV_BodyJog_Strafe	Real	
AGV_BodyJog_Spd	Real	
HMI_BodJog_Steer_FDBCK	Real	
HMI_BodyJog_Strafe_FDFBK	Real	
AGV_Home2Zero	Bool	
AGV_Hom2Angle	Bool	
AGV_AllowAlignmt	Bool	
AGV_JogUnit	Bool	
HMI_JogUnit_FDBCK	Bool	
AGV_RunTestON	Bool	
AGV_RunTestAbort	Bool	
InOut		
▼ Temp		
StepperSpd_Temp	Real	
SteeringJog_Temp	Real	
StrafeJog_Temp	Real	
SpeedJog_Temp	Real	
Constant		
▼ Return		
FC HMI Inputs	Void	

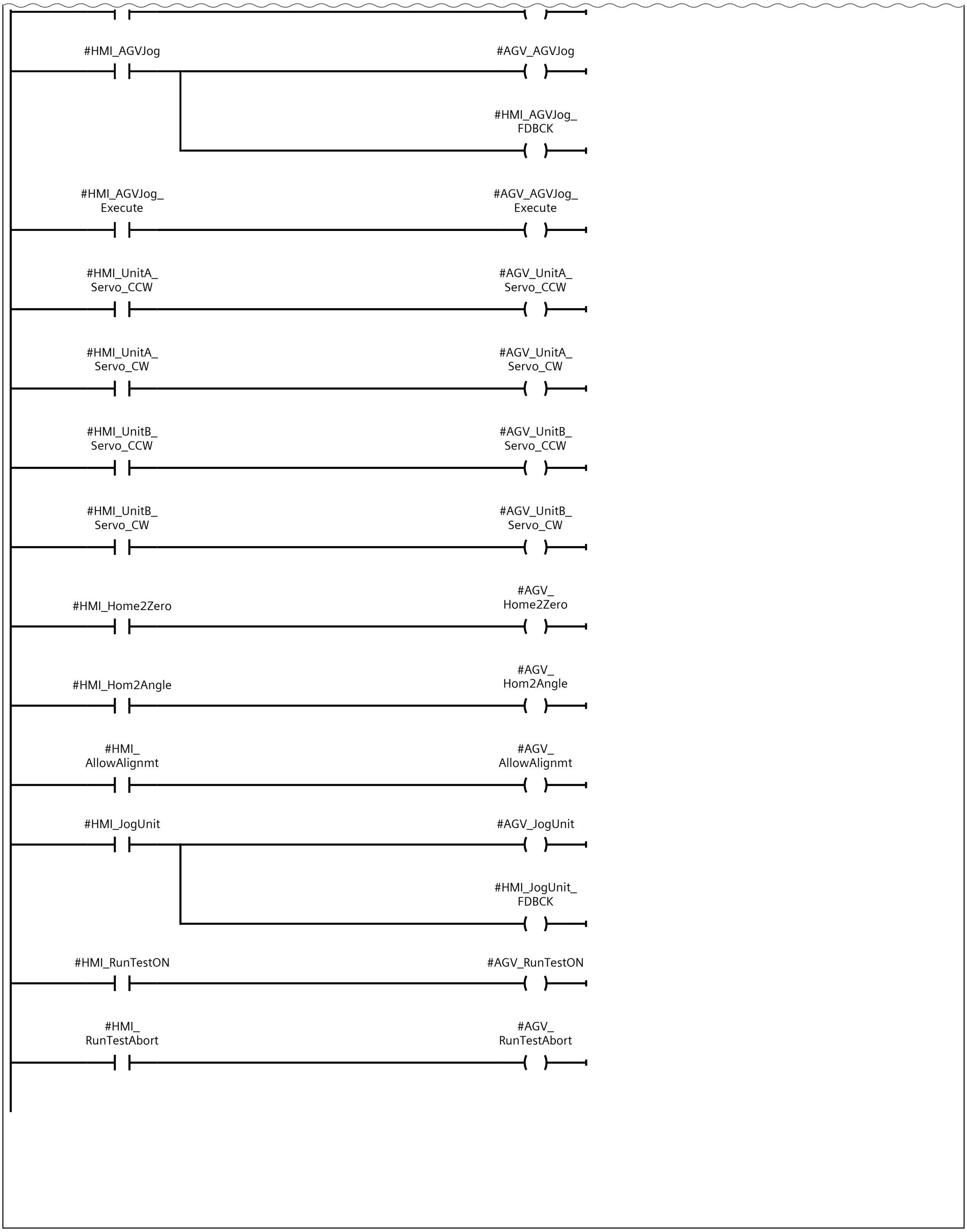
Network 1: Boolean passthrough

Network 1: Boolean passthrough (1.1 / 2.1)

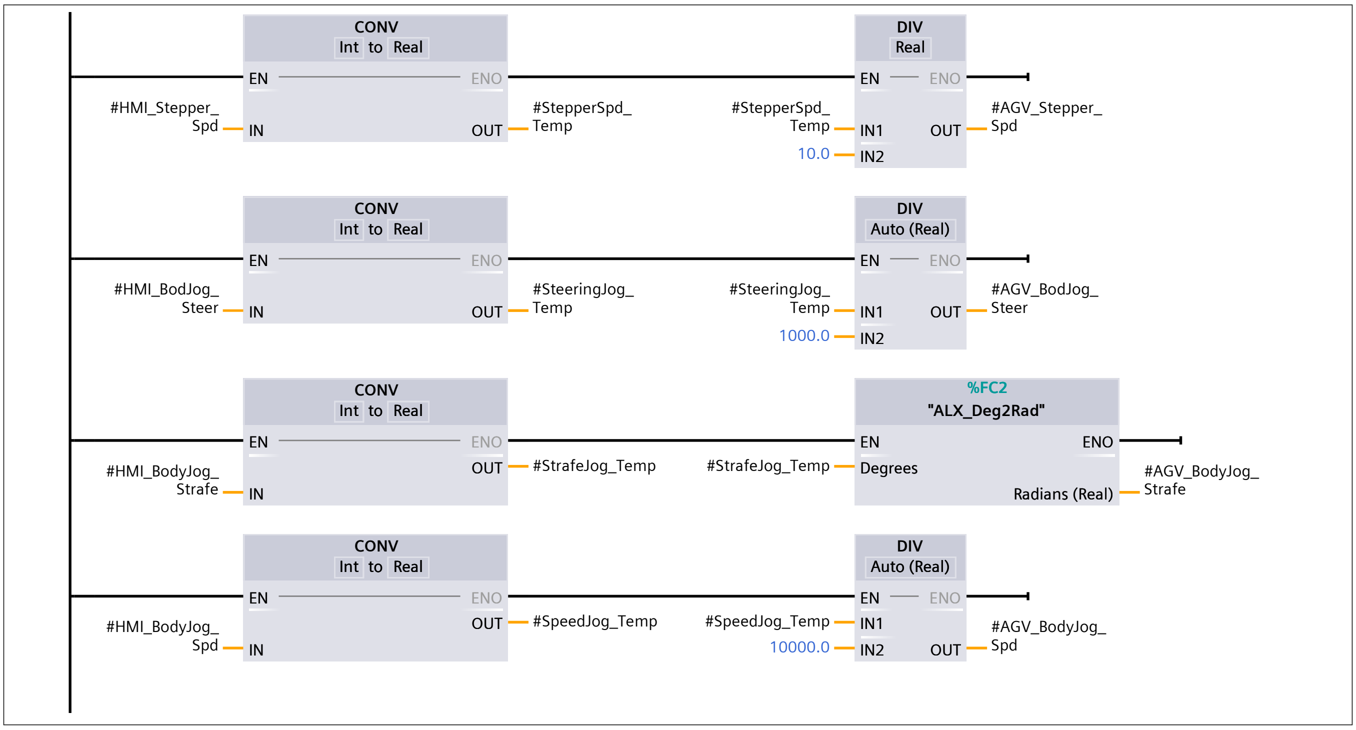


Network 1: Boolean passthrough (2.1 / 2.1)

1.1 (Page3 - 3)



Network 2: Digital Data passthrough



Network 3: HMI Feedback

These are done here rather than the HMI write block to save on reversing the calculations ie deg ->rad ->deg

```
// Jog system body feedback
    //
    #HMI_BodJog_Steer_FDBCK := #SteeringJog_Temp / 100.0;
    #HMI_BodyJog_Strafe_FDFBK := #StrafeJog_Temp * -1;
```

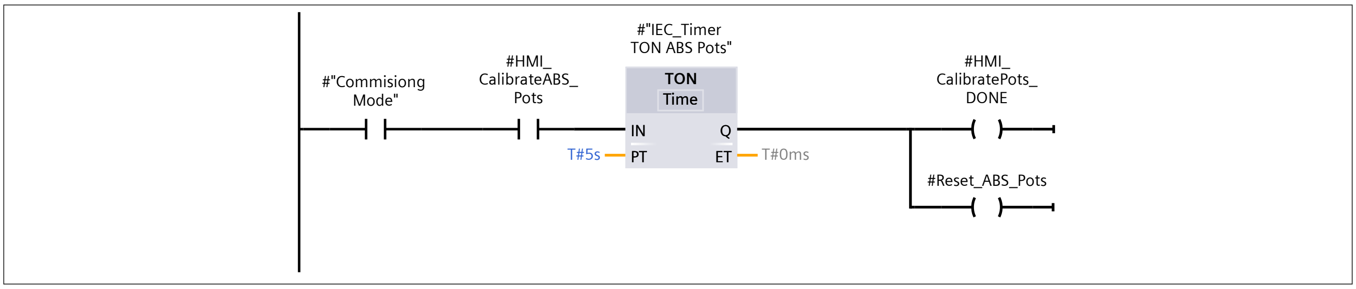
14. Auto Calibrations

FB Auto Calibrations [FB225]

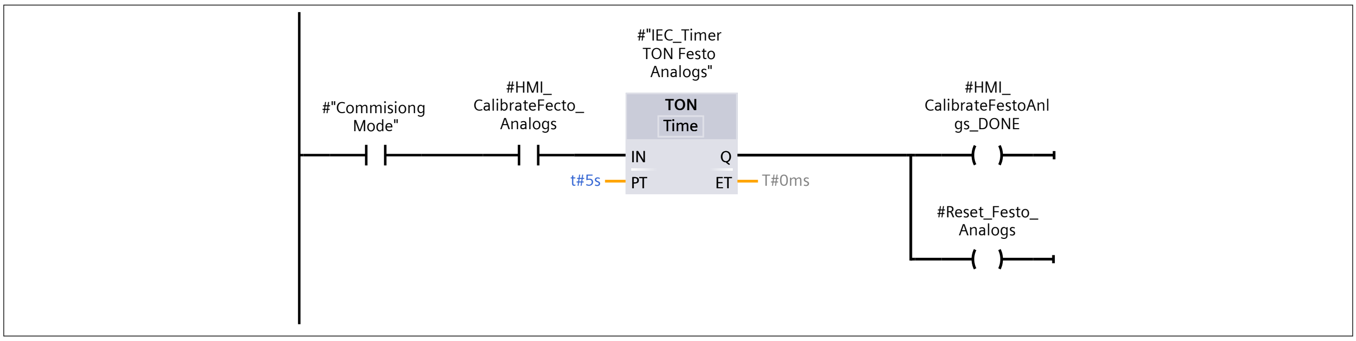
FB Auto Calibrations Properties					
General					
Name	FB Auto Calibrations	**Number**	225	**Type**	FB
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
▼ Input			
Commisiong Mode	Bool	false	Non-retain
HMI_CalibrateABS_Pots	Bool	false	Non-retain
HMI_CalibrateFecto_Analogs	Bool	false	Non-retain
▼ Output			
HMI_CalibratePots_DONE	Bool	false	Non-retain
HMI_CalibrateFestoAnlgs_DONE	Bool	false	Non-retain
Reset_ABS_Pots	Bool	false	Non-retain
Reset_Festo_Analogs	Bool	false	Non-retain
InOut			
▼ Static			
IEC_Timer TON ABS Pots	TON_TIME		Non-retain
IEC_Timer TON Festo Analogs	TON_TIME		Non-retain
Temp			
Constant			

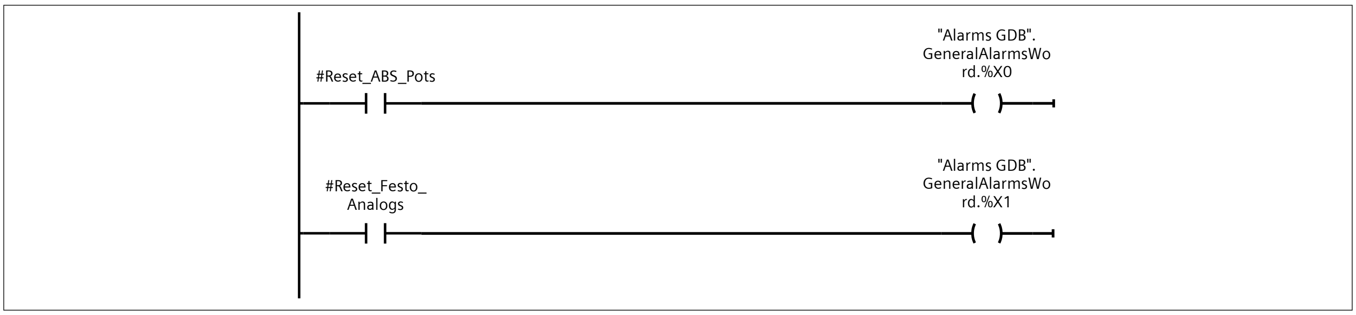
Network 1: Calibrate absolute potetiometers delay timer



Network 2: Calibrate festo analog values



Network 3: Alarms



14. Auto Calibrations / Instance DBs

FB Auto Calibrations iDB [DB225]

FB Auto Calibrations iDB Properties					
General					
Name	FB Auto Calibrations iDB	**Number**	225	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
Commisiong Mode	Bool	false	False
HMI_CalibrateABS_Pots	Bool	false	False
HMI_CalibrateFecto_Analogs	Bool	false	False
▼ Output			
HMI_CalibratePots_DONE	Bool	false	False
HMI_CalibrateFestoAnlgs_DONE	Bool	false	False
Reset_ABS_Pots	Bool	false	False
Reset_Festo_Analogs	Bool	false	False
InOut			
▼ Static			
IEC_Timer TON ABS Pots	TON_TIME		False
IEC_Timer TON Festo Analogs	TON_TIME		False

15. System Shutdown

Controlled Shutdown Sequence [FB500]

Controlled Shutdown Sequence Properties					
General					
Name	Controlled Shutdown Sequence	**Number**	500	**Type**	FB
Language	GRAPH	**Numbering**	Manual	**Network language**	LAD
Block version	V6.0				
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
▼ Input			
ShutdownSW	Bool	false	Non-retain
OFF_SQ	Bool	false	Non-retain
INIT_SQ	Bool	false	Non-retain
ACK_EF	Bool	false	Non-retain
S_PREV	Bool	false	Non-retain
S_NEXT	Bool	false	Non-retain
SW_AUTO	Bool	false	Non-retain
SW_TAP	Bool	false	Non-retain
SW_TOP	Bool	false	Non-retain
SW_MAN	Bool	false	Non-retain
S_SEL	Int	0	Non-retain
S_ON	Bool	false	Non-retain
S_OFF	Bool	false	Non-retain
T_PUSH	Bool	false	Non-retain
▼ Output			
S_NO	Int	0	Non-retain
S_MORE	Bool	false	Non-retain
S_ACTIVE	Bool	false	Non-retain
ERR_FLT	Bool	false	Non-retain
AUTO_ON	Bool	false	Non-retain
TAP_ON	Bool	false	Non-retain
TOP_ON	Bool	false	Non-retain
MAN_ON	Bool	false	Non-retain
ShutdownRouter	Bool	false	Non-retain
ShutdownIPC	Bool	false	Non-retain
24VKillSW	Bool	false	Non-retain
InOut			
▼ Static			
RT_DATA	G7_RTDataPlus_V6		Non-retain
Trans1	G7_TransitionPlus_V6		Non-retain

Name	Data type	Default value	Retain
Trans2	G7_Transition-Plus_V6		Non-retain
Trans3	G7_Transition-Plus_V6		Non-retain
Trans4	G7_Transition-Plus_V6		Non-retain
Initialise	G7_StepPlus_V6		Non-retain
RouterShutdownStep	G7_StepPlus_V6		Non-retain
IPCShutdownStep	G7_StepPlus_V6		Non-retain
24V UPS Killswitch	G7_StepPlus_V6		Non-retain
Temp			
Constant			

Alarms

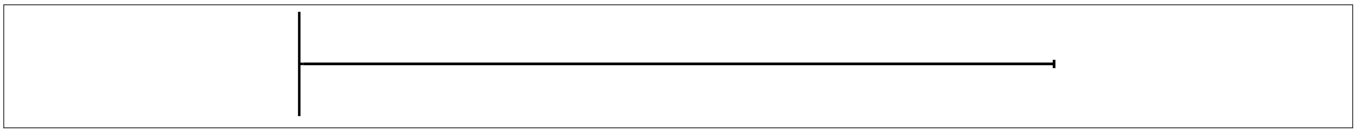
Enable alarms	True

Category	Category enabler	Display class
Error		0
Warning		0
Info		0
Category 4		0
Category 5		0
Category 6		0
Category 7		0
Category 8		0

Category for interlocks	Error	**Subcategory 1 for interlocks**		**Subcategory 2 for interlocks**	
Category for supervisions	Error	**Subcategory 1 for supervisions**		**Subcategory 2 for supervisions**	
Category for GRAPH warnings	Warning	**Subcategory 1 for GRAPH warnings**		**Subcategory 2 for GRAPH warnings**	

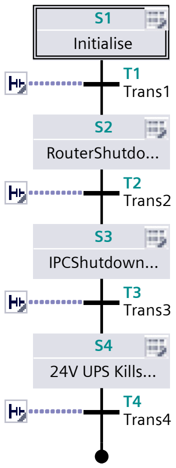
Permanent pre-instructions

1:



Sequences (1)

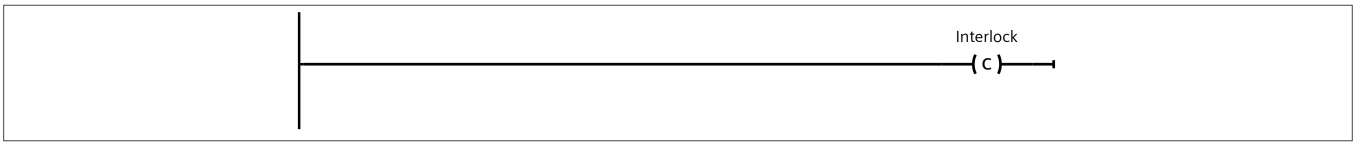
1:



S1 - [Initial step]:Initialise

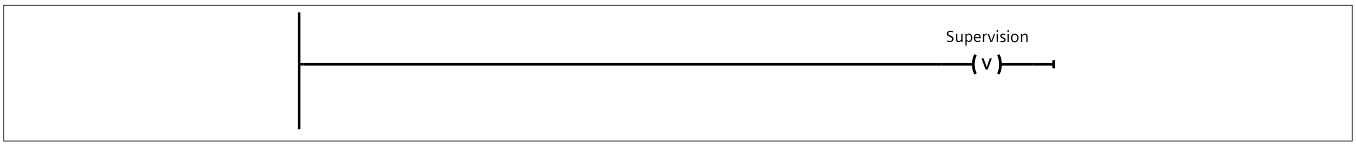
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

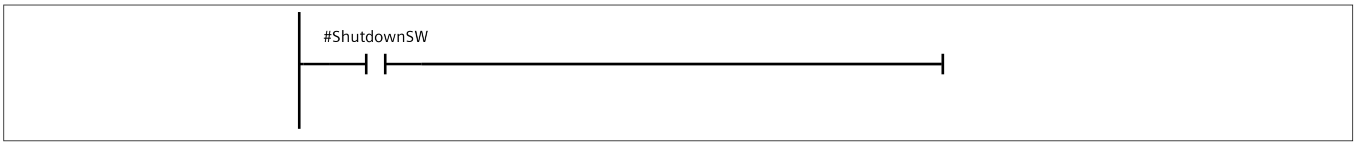
Supervision alarm	
Alarm text	



Actions:

Actions:			
Interlock	Event	Qualifier	Action
		R	#ShutdownRouter

T1:Trans1

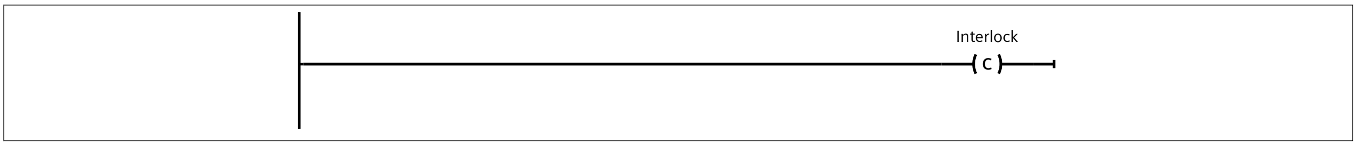


S2:RouterShutdownStep

1) PLC sends a shutdown command to the SCADA via DB item
2) On positive edge change of DB item SCADA runs a winCC VBScript that call a VSscript program on the windows partition of the IPC
3) Windows VBScript uses batch file to SSH (with password) into router using the program PuTTY
4) Once SSH established to router via PuTTY, the windows VBScript send the appropriate linux commands to shut down the router

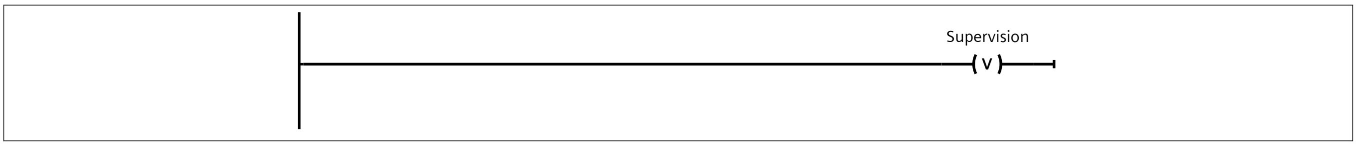
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

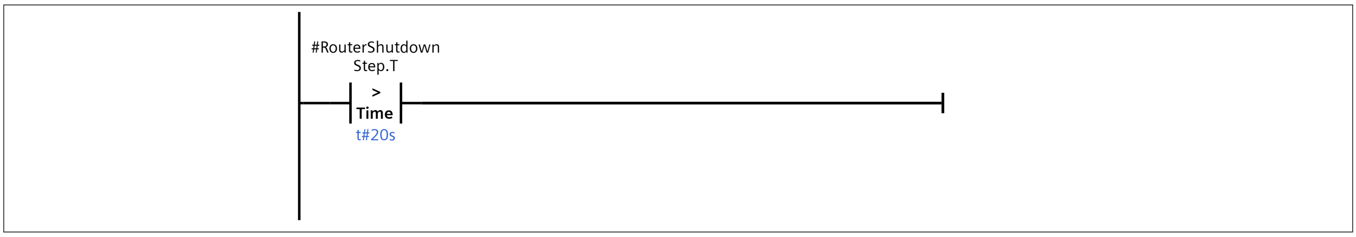
Supervision alarm	
Alarm text	



Actions:

Actions:			
Interlock	Event	Qualifier	Action
		S	#ShutdownRouter

T2:Trans2

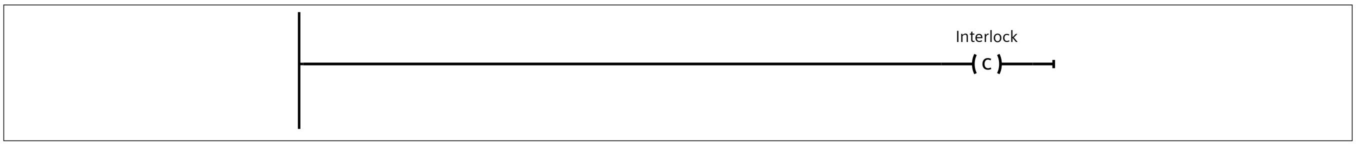


S3:IPCShutdownStep

1) PLC sends a shutdown command to the SCADA via DB item
2) On positive edge change of DB item SCADA runs a winCC VBScript that runs a batch file on the windows partition of the IPC
3) Batch file contains the command "Shutdown \s \t 10s" to shut down the IPC in 10 seconds after the batch file has been run

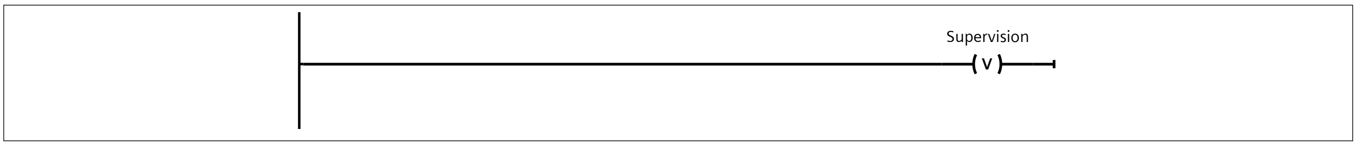
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

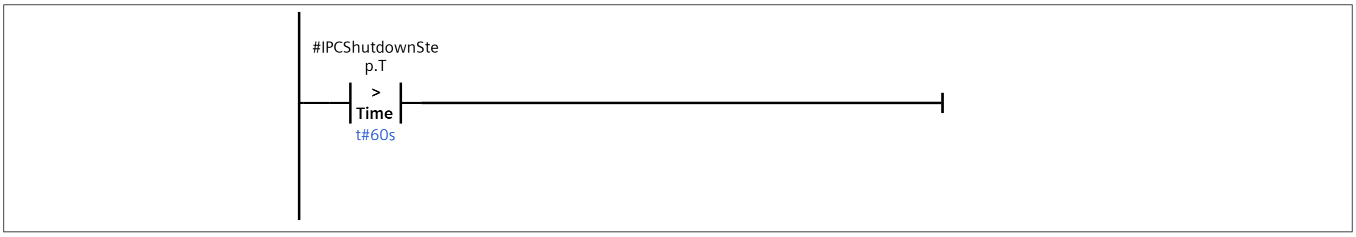
Supervision alarm	
Alarm text	



Actions:

Actions:			
Interlock	Event	Qualifier	Action
		S	#ShutdownIPC

T3:Trans3

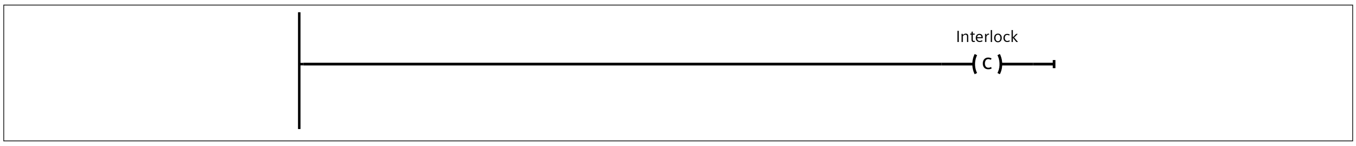


S4:24V UPS Killswitch

PLC kills itself by telling the 24VDC UPS to shutdown, system is now completely offline

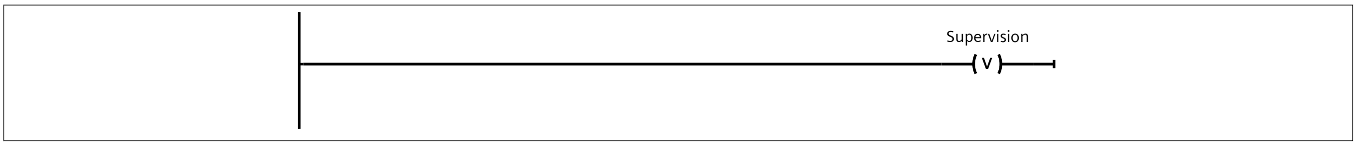
Interlock -(c)-:

Interlock alarm	
Alarm text	



Supervision -(v)-:

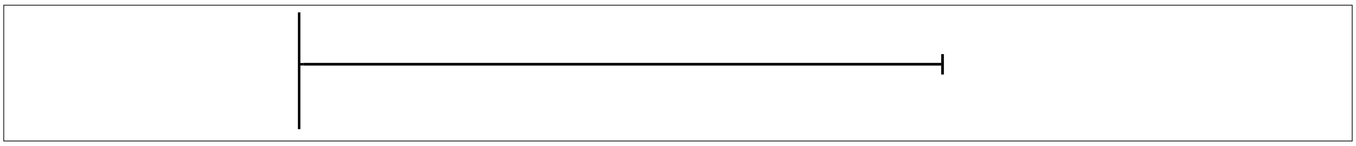
Supervision alarm	
Alarm text	



Actions:

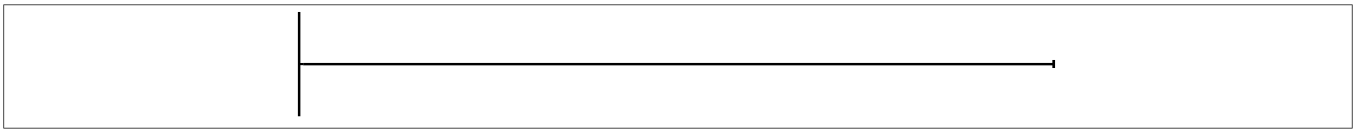
Actions:			
Interlock	**Event**	**Qualifier**	**Action**
		S	#"24VKillSW"

T4:Trans4



Permanent post-instructions

1:



15. System Shutdown / Instance DBs

Controlled Shutdown Sequence iDB [DB500]

Controlled Shutdown Sequence iDB Properties					
General					
Name	Controlled Shutdown Sequence iDB	**Number**	500	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
ShutdownSW	Bool	false	False
OFF_SQ	Bool	false	False
INIT_SQ	Bool	false	False
ACK_EF	Bool	false	False
S_PREV	Bool	false	False
S_NEXT	Bool	false	False
SW_AUTO	Bool	false	False
SW_TAP	Bool	false	False
SW_TOP	Bool	false	False
SW_MAN	Bool	false	False
S_SEL	Int	0	False
S_ON	Bool	false	False
S_OFF	Bool	false	False
T_PUSH	Bool	false	False
▼ Output			
S_NO	Int	0	False
S_MORE	Bool	false	False
S_ACTIVE	Bool	false	False
ERR_FLT	Bool	false	False
AUTO_ON	Bool	false	False
TAP_ON	Bool	false	False
TOP_ON	Bool	false	False
MAN_ON	Bool	false	False
ShutdownRouter	Bool	false	False
ShutdownIPC	Bool	false	False
24VKillSW	Bool	false	False
InOut			
▼ Static			
RT_DATA	G7_RTDataPlus_V6		False
Trans1	G7_TransitionPlus_V6		False
Trans2	G7_TransitionPlus_V6		False
Trans3	G7_TransitionPlus_V6		False
Trans4	G7_TransitionPlus_V6		False

Name	Data type	Start value	Retain
Initialise	G7_StepPlus_V6		False
RouterShutdownStep	G7_StepPlus_V6		False
IPCShutdownStep	G7_StepPlus_V6		False
24V UPS Killswitch	G7_StepPlus_V6		False

16. TimeSync

This folder is empty.

THIS CODE WAS DEPRECIATED IN FAVOUR OF USING WinCC TIME SYNCRONISATION

17. Testing

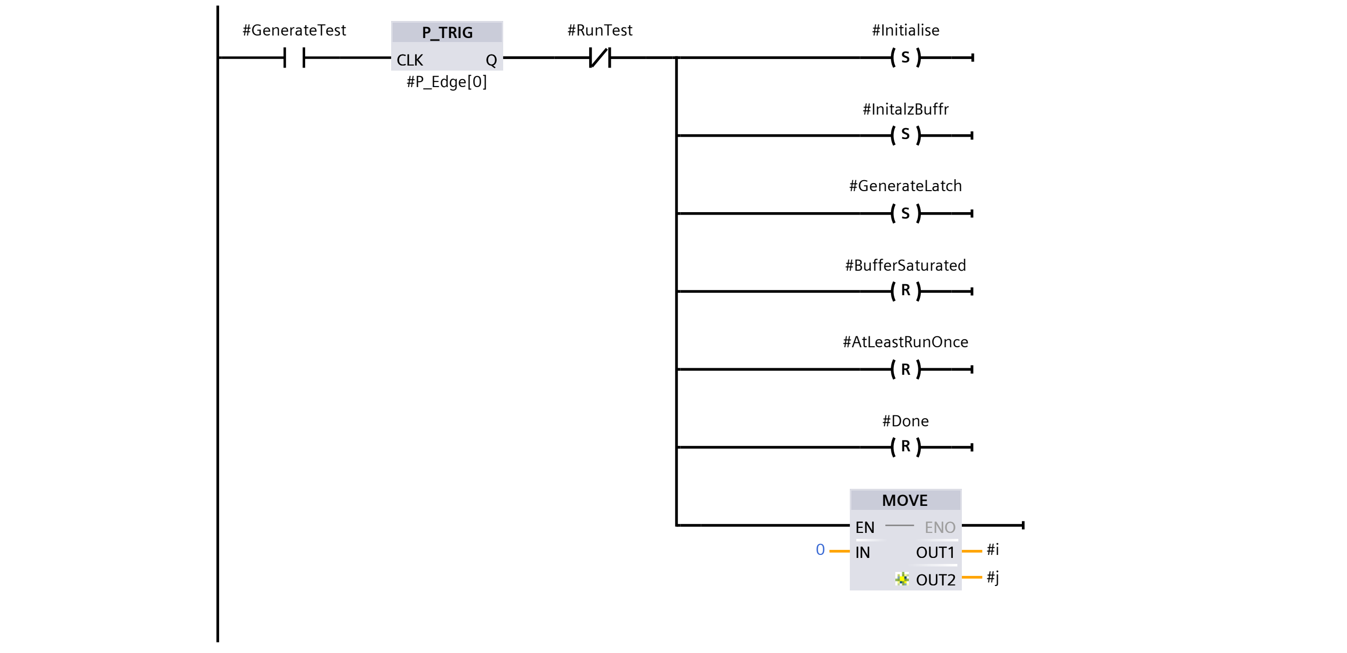
FB Generate Test Data [FB71]

FB Generate Test Data Properties					
General					
Name	FB Generate Test Data	**Number**	71	**Type**	FB
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
▼ Input			
GenerateTest	Bool	false	Non-retain
RunTest	Bool	false	Non-retain
Enable	Bool	false	Non-retain
Cancel	Bool	false	Non-retain
SpeedPot	Int	0	Non-retain
SteeringPot	Int	0	Non-retain
StrafePot	Int	0	Non-retain
▼ Output			
InitalzBuffr	Bool	false	Non-retain
BufferSaturated	Bool	false	Non-retain
BufferSizeCur	Int	0	Non-retain
Busy	Bool	false	Non-retain
Done	Bool	false	Non-retain
InOut			
▼ Static			
P_Edge	Array[0..2] of Bool		Non-retain
Initialise	Bool	false	Non-retain
GenerateLatch	Bool	false	Non-retain
i	Int	0	Non-retain
j	Int	0	Non-retain
AtLeastRunOnce	Bool	false	Non-retain
N_Enable	Bool	false	Non-retain
Temp			
▼ Constant			
BufferSize	Int	6009	

Network 1: Check if generate test data was requested

Using the positive edge of the generate test data to initialise the speed of the AGV with -1, this is done to set a "STOP" condition when the test is run:
when speed = -1 => end test
maximun test duration = 10min as 6000 samples can be generated and each sample is taken every 100ms

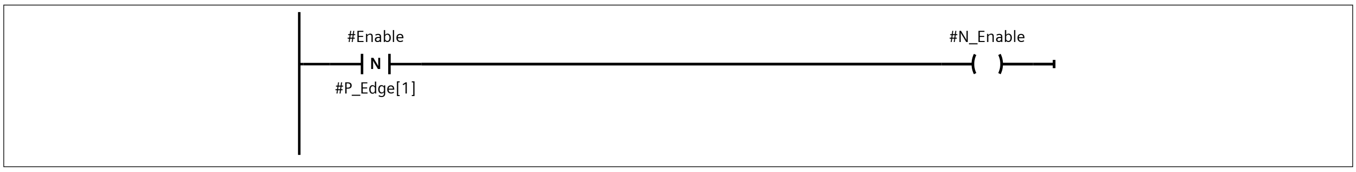


Network 2: Initialise Speed value with -1

runs a loop that re-initialises all of the speed values to -1, since this is a while loop the initialisation should be completed in one of the OB32's cycles. This could cause issues with the watchdog timer if the buffer becomes too large and takes a while to initialise

```
IF (#Initialise = TRUE) THEN
    // Positive edge generated test startup => run initialise loop for speed values

    WHILE (#i <= 6009) DO
        // while buffer index is smaller than buffer size
        "Test Buffer GDB".TestBuffer.Speed[#i] := -1;
        "Test Buffer GDB".TestBuffer.Steering[#i] := 0;
        "Test Buffer GDB".TestBuffer.Strafe[#i] := 0;

        //incriment buffer counter
        #i := #i + 1;
    END_WHILE;

    IF (#i >= 6009) THEN
        //Initialisation is completed
        #Initialise := FALSE;
        #InitalzBuffr := FALSE;
    END_IF;

END_IF;
```

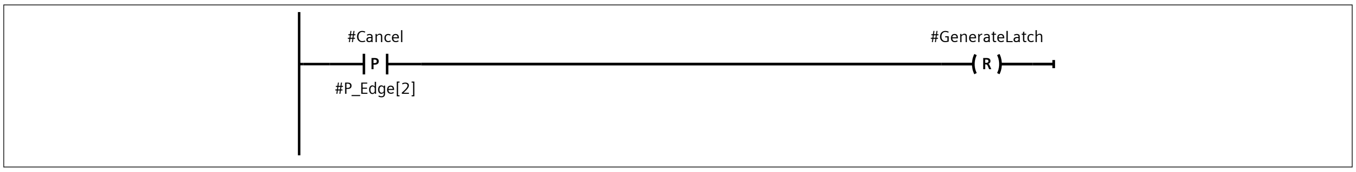
Network 3:



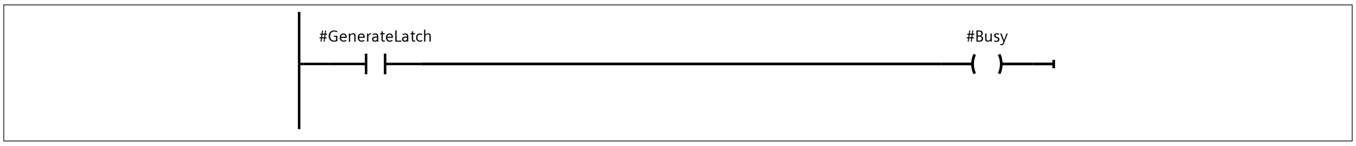
Network 4: Take sample every time this block is run, i.e. every 100ms

```
IF (#Initialise = FALSE) AND (#RunTest = FALSE) AND (#GenerateLatch = TRUE) AND (#Enable = TRUE) THEN
    // IF the system is not inisialising the buffer, the test is not being run using the buffer and the
    // generate buffer data latch is set then each cycle of this block a sample will be taken and placed
    // in the buffer
    
    //take samples from pots
    IF #SpeedPot >= 0 THEN
        "Test Buffer GDB".TestBuffer.Speed[#j] := #SpeedPot;
    ELSE
        "Test Buffer GDB".TestBuffer.Speed[#j] := 0;
    END_IF;
    "Test Buffer GDB".TestBuffer.Steering[#j] := #SteeringPot;
    "Test Buffer GDB".TestBuffer.Strafe[#j] := #StrafePot;
    
    //Incriment buffer index after value stored
    #j := #j + 1;
    
    IF #j > 6000 THEN
        //reached end of buffer (index 6001 to 6009 used as stopping values
        #GenerateLatch := FALSE;
        #BufferSaturated := TRUE;
        #Done := TRUE;
    END_IF;
    
    //Use this to ensure at least one value was taken before system tuened off
    #AtLeastRunOnce := TRUE;
    
ELSIF (#Initialise = FALSE) AND (#GenerateLatch = TRUE) THEN
    
    IF (#N_Enable = TRUE) AND (#AtLeastRunOnce = TRUE) THEN
        //system turned off = stop taking values
        #Done := TRUE;
        #GenerateLatch := FALSE;
        
    ELSIF (#N_Enable = TRUE) AND (#AtLeastRunOnce = FALSE) THEN
        //system turned off and no values taken
        #GenerateLatch := FALSE;
    END_IF;
    
END_IF;

```

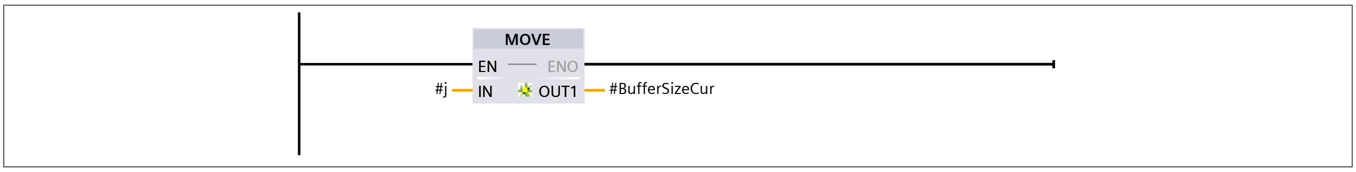
Network 5: overide and cancel test creation



Network 6:



Network 7: Update Current Buffer Size



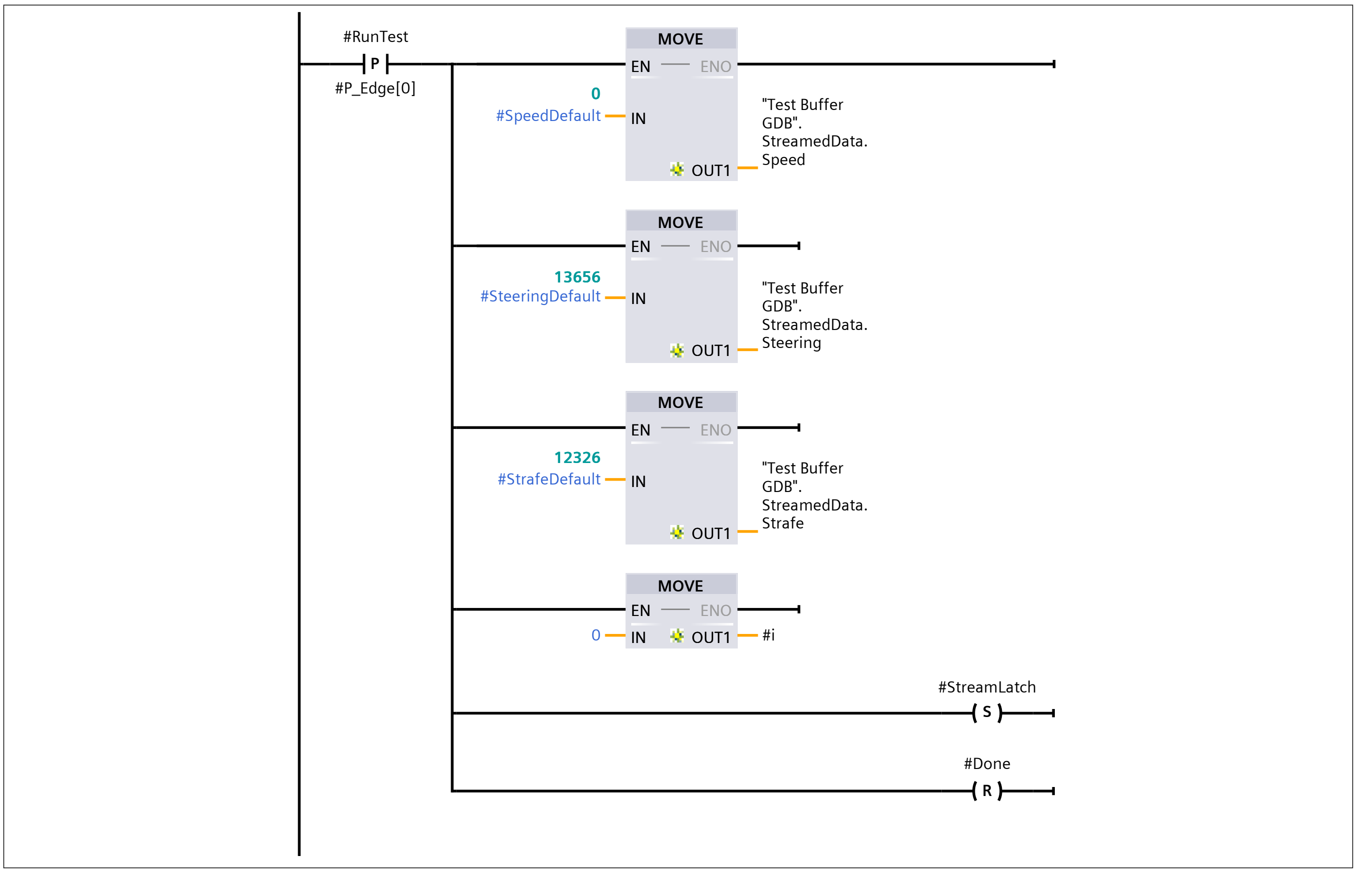
17. Testing

FB Run Test Buffer Values [FB72]

FB Run Test Buffer Values Properties					
General					
Name	FB Run Test Buffer Values	**Number**	72	**Type**	FB
Language	LAD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
▼ Input			
GenerateTest	Bool	false	Non-retain
RunTest	Bool	false	Non-retain
Enable	Bool	false	Non-retain
Abort	Bool	false	Non-retain
▼ Output			
Done	Bool	false	Non-retain
RunIndex	Int	0	Non-retain
InOut			
▼ Static			
P_Edge	Array[0..1] of Bool		Non-retain
i	Int	0	Non-retain
StreamLatch	Bool	false	Non-retain
Temp			
▼ Constant			
SpeedDefault	Int	0	
SteeringDefault	Int	13656	
StrafeDefault	Int	12326	

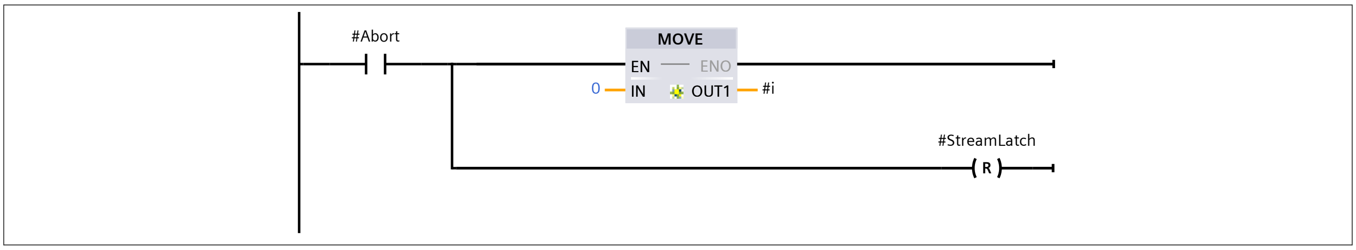
Network 1: Initialise



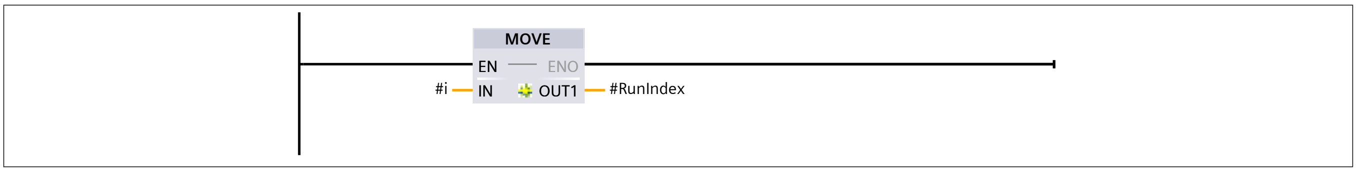
Network 2: Stream Data from Buffer to Manual Block

```
IF (#GenerateTest = FALSE) AND (#StreamLatch = TRUE) AND (#RunTest = TRUE) AND (#Enable = TRUE) THEN
    // Not generating test data, test is ready to be run, AGV is enabled
    //

    IF ("Test Buffer GDB".TestBuffer.Speed[#i] <> -1) AND (#i <= 6000) THEN

        //Stream current buffer index data to manual mode linking valiables
        "Test Buffer GDB".StreamedData.Speed := "Test Buffer GDB".TestBuffer.Speed[#i];
        "Test Buffer GDB".StreamedData.Steering := "Test Buffer GDB".TestBuffer.Steering[#i];
        "Test Buffer GDB".StreamedData.Strafe := "Test Buffer GDB".TestBuffer.Strafe[#i];

        //incriment index for data buffer
        #i := #i + 1;

    ELSIF ("Test Buffer GDB".TestBuffer.Speed[#i] = -1) OR (#i > 6000) THEN

        #StreamLatch := FALSE;
        #Done := TRUE;

        //"Test Buffer GDB".StreamedData.Speed := #SpeedDefault;
        //"Test Buffer GDB".StreamedData.Steering := #SteeringDefault;
        //"Test Buffer GDB".StreamedData.Strafe := #StrafeDefault;

    END_IF;

END_IF;

```

Network 3: Abort



Network 4: Pass run index number to output



17. Testing / Instance DBs

FB Generate Test Data iDB [DB71]

FB Generate Test Data iDB Properties					
General					
Name	FB Generate Test Data iDB	**Number**	71	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
GenerateTest	Bool	false	False
RunTest	Bool	false	False
Enable	Bool	false	False
Cancel	Bool	false	False
SpeedPot	Int	0	False
SteeringPot	Int	0	False
StrafePot	Int	0	False
▼ Output			
InitalzBuffr	Bool	false	False
BufferSaturated	Bool	false	False
BufferSizeCur	Int	0	False
Busy	Bool	false	False
Done	Bool	false	False
InOut			
▼ Static			
P_Edge	Array[0..2] of Bool		False
Initialise	Bool	false	False
GenerateLatch	Bool	false	False
i	Int	0	False
j	Int	0	False
AtLeastRunOnce	Bool	false	False
N_Enable	Bool	false	False

17. Testing / Instance DBs

FB Run Test Buffer Values iDB [DB72]

FB Run Test Buffer Values iDB Properties					
General					
Name	FB Run Test Buffer Values iDB	**Number**	72	**Type**	DB
Language	DB	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Start value	Retain
▼ Input			
GenerateTest	Bool	false	False
RunTest	Bool	false	False
Enable	Bool	false	False
Abort	Bool	false	False
▼ Output			
Done	Bool	false	False
RunIndex	Int	0	False
InOut			
▼ Static			
P_Edge	Array[0..1] of Bool		False
i	Int	0	False
StreamLatch	Bool	false	False

ALX True ARCTAN [FC7]

ALX True ARCTAN Properties					
General					
Name	ALX True ARCTAN	**Number**	7	**Type**	FC
Language	SCL	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value
▼ Input		
X	Real	
Y	Real	
▼ Output		
Angle	Real	
InOut		
Temp		
Constant		
▼ Return		
ALX True ARCTAN	Void	

```

//since issue will occur using tan at 0, 90, 180, 270 as tan is a piece of shit,
//an IF statement will need to be used to generate the angles for these conditions

IF #X = 0 THEN
  // If X = 0 then angle is either 90 or 270 dependant on whether Y is pos or neg
  //
  IF #Y > 0 THEN
    // If Y > 0 then angle is 90 degrees
    #Angle := ("pi" / 2);    // pi/2 radians == 90 degrees

  ELSIF #Y < 0 THEN
    // If Y < 0 then angle is 270 degrees
    #Angle := ((3/2)*"pi"); // 3pi/2 == 270 degrees

  ELSE
    // Y = 0, not possible to find angle => use last angle value (i.e. do not update value)
    ;
  END_IF;

ELSIF #Y = 0 THEN
  // If Y = 0 then angle is either 0 or 180 dependant on whether X is pos or neg
  //
  IF #X > 0 THEN
    //If X > 0 then angle is 0 degrees
    #Angle := 0; // 0 radians = 0 degrees

  ELSIF #X < 0 THEN
    // If X < 0 then angle is 180 degrees
    #Angle := "pi"; // pi radians == 180 degrees
  ELSE
    // X = 0, not possible to find angle => use last angle value (i.e. do not update value)
    ;
  END_IF;

ELSE
  // Niether X nor Y is zero thus the angle of the resultant vector can be found using tan, quadrants shown below:
  // X
  // ^
  // |
  // Q1 | Q4
  // Y <.......|..........
  // |
  // Q2 | Q3
  // |

  IF (#X > 0) AND (#Y > 0) THEN
    // Value is in First Quadrant (+X, +Y)
    #Angle := ATAN((#Y/#X));
  END_IF;

  IF (#X < 0) AND (#Y > 0) THEN
    // Value in the second quadrant (-X, +Y), thus add 180 to the NEGATIVE result
    #Angle := ATAN((#Y/#X)) + "pi";
  END_IF;

  IF (#X < 0) AND (#Y < 0) THEN
    // Value in the second quadrant (-X, -Y), thus add 180 to the POSITIVE result
    #Angle := ATAN((#Y / #X)) + "pi";
  END_IF;

  IF (#X > 0) AND (#Y < 0) THEN
    // Value in the second quadrant (-X, -Y), thus add 360 to the NEGATIVE result
    #Angle := ATAN((#Y / #X)) + (2*"pi");
  END_IF;

END_IF;
```

ALX_Deg2Rad [FC2] [ALX_Deg2Rad V 0.0.1]

ALX_Deg2Rad Properties					
General					
Name	ALX_Deg2Rad	**Number**	2	**Type**	FC
Language	SCL	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value
▼ Input		
Degrees	Real	
▼ Output		
Radians (Real)	Real	
InOut		
Temp		
Constant		
▼ Return		
ALX_Deg2Rad	Void	

```
// Convert degrees to radians since SIEMENS math blocks use radians and not degrees

#"Radians (Real)" := ("pi" / 180.0) * #Degrees;

//Round off value to a DInt

//#"Radians (DInt)" := ROUND(#"Radians (Real)");
```

ALX_Rad2Deg [FC6] [ALX_Rad2Deg V 0.0.1]

ALX_Rad2Deg Properties					
General					
Name	ALX_Rad2Deg	**Number**	6	**Type**	FC
Language	SCL	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value
▼ Input		
Rad	Real	
▼ Output		
Degree	Real	
InOut		
Temp		
Constant		
▼ Return		
ALX_Rad2Deg	Void	

```
#Degree := (180 / "pi") * #Rad;
```

ALX_Rad2FestoUnts [FC3]

ALX_Rad2FestoUnts Properties					
General					
Name	ALX_Rad2FestoUnts	**Number**	3	**Type**	FC
Language	SCL	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value
▼ Input		
Radians	Real	
▼ Output		
Festo Incriments	DInt	
InOut		
▼ Temp		
Fractional Incriments	Real	
FestoRatio	Real	
Constant		
▼ Return		
ALX_Rad2FestoUnts	Void	

```
// Convert radians into festo stepper incriments , there is 65536 incriments per revolution according to the documentation

#"FestoRatio" := DINT_TO_REAL("Festo Incriments Rev");
#"Fractional Incriments" := (#FestoRatio/ (2 * "pi")) * #Radians;

//Since the Festo incriments factor is in DInt rounding off is needed of the real value that results from the equation above

#"Festo Incriments" := ROUND(#"Fractional Incriments");
```

ALX_Rad/s2RPM [FC1] [ALX_Rad/s2RPM V 0.0.1]

ALX_Rad/s2RPM Properties					
General					
Name	ALX_Rad/s2RPM	**Number**	1	**Type**	FC
Language	SCL	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value
▼ Input		
rad/s	Real	
▼ Output		
RPM	Real	
InOut		
Temp		
Constant		
▼ Return		
ALX_Rad/s2RPM	Void	

```
//Convert radians per second to revolutions per minute

#RPM := (60.0 / (2 * "pi")) * #"rad/s";
```

ALX_RPM2Rad/s [FC4]

ALX_RPM2Rad/s Properties					
General					
Name	ALX_RPM2Rad/s	**Number**	4	**Type**	FC
Language	SCL	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value
▼ Input		
RPM	Real	
▼ Output		
Rad/s	Real	
InOut		
Temp		
Constant		
▼ Return		
ALX_RPM2Rad/s	Void	

```
//Convert RPM to Rad/s

#"Rad/s" := ((2 * "pi") / 60.0) * #RPM;
```

ALX_y=mx+c [FC5]

ALX_y=mx+c Properties					
General					
Name	ALX_y=mx+c	**Number**	5	**Type**	FC
Language	SCL	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value
▼ Input		
m	Real	
x	Real	
c	Real	
▼ Output		
y	Real	
InOut		
Temp		
Constant		
▼ Return		
ALX_y=mx+c	Void	

```
#y := (#m * #x) + #c;
```

T.2 Technology Objects

Code can be found on next page

Technology objects

Servo A SpeedAxis [DB210]

Servo A SpeedAxis properties					
General					
Name	Servo A SpeedAxis	**Number**	210	**Type**	DB
Language	Motion_DB	**Numbering**	manual		
Information					
Title		**Author**	SIMATIC	**Comment**	
Family	MC_1500	**Version**	3.0	**User-defined ID**	

Name	Data type	Project start value	Unit of measure	Comment
Velocity	LREAL	0	1/min	
Acceleration	LREAL	0	1/s²	
ActualSpeed	LREAL	0	1/min	
▼ Actor	Struct			
Type	DINT	PROFIDRIVE		
InverseDirection	BOOL	False		
DataAdaption	DINT	ACTOR_DATA_ADAPTION_YES		
Efficiency	LREAL	1		
▼ Interface	Struct			
▼ AddressIn	Struct			
RID	DWORD	34080550		
AREA	BYTE	129		
DB_NUMBER	UINT	0		
OFFSET	UDINT	2400		
▼ AddressOut	Struct			
RID	DWORD	34080551		
AREA	BYTE	130		
DB_NUMBER	UINT	0		
OFFSET	UDINT	2400		
EnableDriveOutput	BOOL	False		
▼ EnableDriveOutputAddress	Struct			
RID	DWORD	0		
AREA	BYTE	0		
DB_NUMBER	UINT	0		
OFFSET	UDINT	0		
DriveReadyInput	BOOL	False		
▼ DriveReadyInputAddress	Struct			
RID	DWORD	0		
AREA	BYTE	0		
DB_NUMBER	UINT	0		
OFFSET	UDINT	0		
▼ DriveParameter	Struct			

Name	Data type	Project start value	Unit of measure	Comment
ReferenceSpeed	LREAL	3000	1/min	
MaxSpeed	LREAL	3000	1/min	
ReferenceTorque	LREAL	3.2	Nm	
▼ TorqueLimiting	Struct			
LimitBase	DINT	TORQUELIMIT_LIMIT-BASE_LOAD		
PositionBasedMonitorings	DINT	TORQUELIMIT_POS_BASED_MONITORING_NO		
▼ LimitDefaults	Struct			
Torque	LREAL	25	Nm	
Force	LREAL	0	N	
▼ LoadGear	Struct			
Numerator	UDINT	1803		
Denominator	UDINT	100		
▼ DynamicLimits	Struct			
MaxVelocity	LREAL	25	1/min	
MaxAcceleration	LREAL	4.16666666666667	1/s²	
MaxDeceleration	LREAL	4.16666666666667	1/s²	
MaxJerk	LREAL	83.3333333333335	1/s³	
▼ DynamicDefaults	Struct			
Velocity	LREAL	5	1/min	
Acceleration	LREAL	0.833333333333334	1/s²	
Deceleration	LREAL	0.833333333333334	1/s²	
Jerk	LREAL	16.6666666666667	1/s³	
EmergencyDeceleration	LREAL	4.16666666666667	1/s²	
▼ Override	Struct			
Velocity	LREAL	100	%	
▼ Units	Struct			
LengthUnit	UDINT	0		
VelocityUnit	UDINT	0		
TimeUnit	UDINT	0		
TorqueUnit	UDINT	0		
ForceUnit	UDINT	0		
▼ StatusDrive	Struct			
InOperation	BOOL	False		
CommunicationOK	BOOL	False		
Error	BOOL	False		
AdaptionState	DINT	NOT_ADAPTED		
StatusWord	DWORD	0		
ErrorWord	DWORD	0		
▼ ErrorDetail	Struct			
Number	UDINT	0		
Reaction	DINT	NONE		
WarningWord	DWORD	0		
▼ ControlPanel	Struct			

Name	Data type	Project start value	Unit of measure	Comment
▼ Input	Struct			
▼ Command	Array[]			
▼ Command[1]	Struct			
ReqCounter	UDINT	0		
Type	UDINT	0		
Position	LREAL	0		
Velocity	LREAL	0	1/min	
Acceleration	LREAL	0	$1/s^2$	
Deceleration	LREAL	0	$1/s^2$	
Jerk	LREAL	0	$1/s^3$	
Param	LREAL	0		
▼ Command[2]	Struct			
ReqCounter	UDINT	0		
Type	UDINT	0		
Position	LREAL	0		
Velocity	LREAL	0	1/min	
Acceleration	LREAL	0	$1/s^2$	
Deceleration	LREAL	0	$1/s^2$	
Jerk	LREAL	0	$1/s^3$	
Param	LREAL	0		
TimeOut	LREAL	2000	ms	
EsLifeSign	UDINT	0		
▼ Output	Struct			
▼ Command	Array[]			
▼ Command[1]	Struct			
AckCounter	UDINT	0		
Error	BOOL	False		
ErrorID	WORD	0		
Done	BOOL	False		
Aborted	BOOL	False		
▼ Command[2]	Struct			
AckCounter	UDINT	0		
Error	BOOL	False		
ErrorID	WORD	0		
Done	BOOL	False		
Aborted	BOOL	False		
RtLifeSign	UDINT	0		
▼ InternalToTrace	Array[]			
▼ InternalToTrace[1]	Struct			
Id	DINT	0		
Value	LREAL	0		
▼ InternalToTrace[2]	Struct			
Id	DINT	0		
Value	LREAL	0		
▼ InternalToTrace[3]	Struct			

Name	Data type	Project start value	Unit of measure	Comment
Id	DINT	0		
Value	LREAL	0		
▼ InternalToTrace[4]	Struct			
Id	DINT	0		
Value	LREAL	0		
▼ VirtualAxis	Struct			
Mode	UDINT	0		
▼ Simulation	Struct			
Mode	UDINT	0		

Technology objects

Servo B SpeedAxis [DB211]

Servo B SpeedAxis properties					
General					
Name	Servo B SpeedAxis	**Number**	211	**Type**	DB
Language	Motion_DB	**Numbering**	manual		
Information					
Title		**Author**	SIMATIC	**Comment**	
Family	MC_1500	**Version**	3.0	**User-defined ID**	

Name	Data type	Project start value	Unit of measure	Comment
Velocity	LREAL	0	1/min	
Acceleration	LREAL	0	1/s²	
ActualSpeed	LREAL	0	1/min	
▼ Actor	Struct			
Type	DINT	PROFIDRIVE		
InverseDirection	BOOL	False		
DataAdaption	DINT	ACTOR_DATA_ADAPTION_YES		
Efficiency	LREAL	1		
▼ Interface	Struct			
▼ AddressIn	Struct			
RID	DWORD	34080550		
AREA	BYTE	129		
DB_NUMBER	UINT	0		
OFFSET	UDINT	3200		
▼ AddressOut	Struct			
RID	DWORD	34080551		
AREA	BYTE	130		
DB_NUMBER	UINT	0		
OFFSET	UDINT	3200		
EnableDriveOutput	BOOL	False		
▼ EnableDriveOutputAddress	Struct			
RID	DWORD	0		
AREA	BYTE	0		
DB_NUMBER	UINT	0		
OFFSET	UDINT	0		
DriveReadyInput	BOOL	False		
▼ DriveReadyInputAddress	Struct			
RID	DWORD	0		
AREA	BYTE	0		
DB_NUMBER	UINT	0		
OFFSET	UDINT	0		
▼ DriveParameter	Struct			

Name	Data type	Project start value	Unit of measure	Comment
ReferenceSpeed	LREAL	3000	1/min	
MaxSpeed	LREAL	3000	1/min	
ReferenceTorque	LREAL	3.2	Nm	
▼ TorqueLimiting	Struct			
LimitBase	DINT	TORQUELIMIT_LIMIT-BASE_LOAD		
PositionBasedMonitorings	DINT	TORQUELIMIT_POS_BASED_MONITORING_NO		
▼ LimitDefaults	Struct			
Torque	LREAL	25	Nm	
Force	LREAL	0	N	
▼ LoadGear	Struct			
Numerator	UDINT	1803		
Denominator	UDINT	100		
▼ DynamicLimits	Struct			
MaxVelocity	LREAL	25	1/min	
MaxAcceleration	LREAL	8.33333333333333	1/s²	
MaxDeceleration	LREAL	8.33333333333333	1/s²	
MaxJerk	LREAL	166.666666666667	1/s³	
▼ DynamicDefaults	Struct			
Velocity	LREAL	5	1/min	
Acceleration	LREAL	1.66666666666667	1/s²	
Deceleration	LREAL	1.66666666666667	1/s²	
Jerk	LREAL	33.3333333333334	1/s³	
EmergencyDeceleration	LREAL	8.33333333333333	1/s²	
▼ Override	Struct			
Velocity	LREAL	100	%	
▼ Units	Struct			
LengthUnit	UDINT	0		
VelocityUnit	UDINT	0		
TimeUnit	UDINT	0		
TorqueUnit	UDINT	0		
ForceUnit	UDINT	0		
▼ StatusDrive	Struct			
InOperation	BOOL	False		
CommunicationOK	BOOL	False		
Error	BOOL	False		
AdaptionState	DINT	NOT_ADAPTED		
StatusWord	DWORD	0		
ErrorWord	DWORD	0		
▼ ErrorDetail	Struct			
Number	UDINT	0		
Reaction	DINT	NONE		
WarningWord	DWORD	0		
▼ ControlPanel	Struct			

Name	Data type	Project start value	Unit of measure	Comment
▼ Input	Struct			
▼ Command	Array[]			
▼ Command[1]	Struct			
ReqCounter	UDINT	0		
Type	UDINT	0		
Position	LREAL	0		
Velocity	LREAL	0	1/min	
Acceleration	LREAL	0	$1/s^2$	
Deceleration	LREAL	0	$1/s^2$	
Jerk	LREAL	0	$1/s^3$	
Param	LREAL	0		
▼ Command[2]	Struct			
ReqCounter	UDINT	0		
Type	UDINT	0		
Position	LREAL	0		
Velocity	LREAL	0	1/min	
Acceleration	LREAL	0	$1/s^2$	
Deceleration	LREAL	0	$1/s^2$	
Jerk	LREAL	0	$1/s^3$	
Param	LREAL	0		
TimeOut	LREAL	2000	ms	
EsLifeSign	UDINT	0		
▼ Output	Struct			
▼ Command	Array[]			
▼ Command[1]	Struct			
AckCounter	UDINT	0		
Error	BOOL	False		
ErrorID	WORD	0		
Done	BOOL	False		
Aborted	BOOL	False		
▼ Command[2]	Struct			
AckCounter	UDINT	0		
Error	BOOL	False		
ErrorID	WORD	0		
Done	BOOL	False		
Aborted	BOOL	False		
RtLifeSign	UDINT	0		
▼ InternalToTrace	Array[]			
▼ InternalToTrace[1]	Struct			
Id	DINT	0		
Value	LREAL	0		
▼ InternalToTrace[2]	Struct			
Id	DINT	0		
Value	LREAL	0		
▼ InternalToTrace[3]	Struct			

Name	Data type	Project start value	Unit of measure	Comment
Id	DINT	0		
Value	LREAL	0		
▼ InternalToTrace[4]	Struct			
Id	DINT	0		
Value	LREAL	0		
▼ VirtualAxis	Struct			
Mode	UDINT	0		
▼ Simulation	Struct			
Mode	UDINT	0		

T.3 PLC Tages

Code can be found on next page

PLC tags / Default tag table [110]

PLC tags

PLC tags

	Name	Data type	Address	Retain
	Commisioning Dump	Bool	%M0.5	False
	Commisioning Toggle	Bool	%M100.1	False

PLC tags / Battery System [7]

PLC tags

PLC tags

	Name	Data type	Address	Retain
	Battery PIP NO	Bool	%I1.4	False
	Battery PIP NC	Bool	%I1.5	False
	Eject Motor Feedback Eject	Bool	%I1.6	False
	Eject Motor Feedback Insert	Bool	%I1.7	False
	Eject Motor Eject	Bool	%Q1.6	False
	Eject Motor Insert	Bool	%Q1.7	False
	TestNoCounter	Counter	%C0	False

PLC tags / Clocks [9]

PLC tags

PLC tags				
	Name	Data type	Address	Retain
	Clock_Byte	Byte	%MB10	False
	Clock_10Hz	Bool	%M10.0	False
	Clock_5Hz	Bool	%M10.1	False
	Clock_2.5Hz	Bool	%M10.2	False
	Clock_2Hz	Bool	%M10.3	False
	Clock_1.25Hz	Bool	%M10.4	False
	Clock_1Hz	Bool	%M10.5	False
	Clock_0.625Hz	Bool	%M10.6	False
	Clock_0.5Hz	Bool	%M10.7	False

PLC tags / Constants [20]

User constants

User constants

	Name	Data type	Value
▤	Yaw Angle Update Freq	Real	20.0
▤	Turning Radius	Real	0.50
▤	Wheel 2 Wheel Center Dist	Real	1.08418
▤	Wheel Radius	Real	0.075
▤	Caster Offset	Real	0.045
▤	Festo Incriments Rev	DInt	65536
▤	Festo Motor Scale Speed	Real	25.527
▤	Servo Saturation Speed	Real	3000.0
▤	pi	Real	3.1415926536
▤	2pi	Real	6.2831853072
▤	Allowable Steering Deviation	Real	0.0349066
▤	Stepper GearRatio	Real	38.0
▤	Servo GearRatio	Real	18.03
▤	Kinematic Offset Rad	Real	1.091473064
▤	POT_SteeringLIMITS_L	Int	98
▤	POT_SteeringLIMITS_H	Int	27451
▤	POT_StrafeLIMITS_L	Int	54
▤	POT_StrafeLIMITS_H	Int	26606
▤	POT_SpeedLIMITS_L	Int	68
▤	POT_SpeedLIMITS_H	Int	27437

PLC tags / EStops [2]

PLC tags

PLC tags

	Name	Data type	Address	Retain
	Rear EStop	Bool	%I4.0	False
	Front EStop	Bool	%I4.2	False

PLC tags / Fans [1]

PLC tags

PLC tags

	Name	Data type	Address	Retain
	Disable Fans	Bool	%Q1.5	False

PLC tags / Front PBs [2]

PLC tags

PLC tags				
	Name	Data type	Address	Retain
01	RED Pushbutton	Bool	%I1.3	False
01	BLK Pushbutton	Bool	%I1.2	False

PLC tags / Lamp & Indicators [4]

PLC tags

PLC tags

	Name	Data type	Address	Retain
	Stack Light 1	Bool	%Q0.7	False
	Stack Light 2	Bool	%Q1.0	False
	Stack Light 3	Bool	%Q1.1	False
	Front Light	Bool	%Q1.4	False

PLC tags / Motor STO [10]

PLC tags

PLC tags

	Name	Data type	Address	Retain
	Stepper A Enable	Bool	%Q1.3	False
	Stepper B Enable	Bool	%Q1.2	False
	Servo A STO PWR	Bool	%Q10.0	False
	Servo B STO PWR	Bool	%Q10.1	False
	Stepper B STO PWR	Bool	%Q10.2	False
	Stepper A STO PWR	Bool	%Q10.3	False
	Servo A STO RTN	Bool	%I10.0	False
	Servo B STO RTN	Bool	%I10.1	False
	Stepper B STO RTN	Bool	%I10.2	False
	Stepper A STO RTN	Bool	%I10.3	False

PLC tags / Pendant [19]

PLC tags

PLC tags				
	Name	Data type	Address	Retain
	Speed Pot	Int	%IW30	False
	Pendant SW0	Bool	%I2.0	False
	Pendant SW1	Bool	%I2.1	False
	Pendant SW2	Bool	%I2.2	False
	Pendant SW3	Bool	%I2.3	False
	Pendant SW4	Bool	%I2.4	False
	Pendant SW5	Bool	%I2.5	False
	Pendant SW6	Bool	%I2.6	False
	Pendant SW7	Bool	%I2.7	False
	Pendant LED0	Bool	%Q2.0	False
	Pendant LED1	Bool	%Q2.1	False
	Pendant LED2	Bool	%Q2.2	False
	Pendant LED3	Bool	%Q2.3	False
	Pendant LED4	Bool	%Q2.4	False
	Pendant LED5	Bool	%Q2.5	False
	Pendant LED6	Bool	%Q2.6	False
	Pendant LED7	Bool	%Q2.7	False
	Strafe Pot	Int	%IW32	False
	Steering Pot	Int	%IW34	False

PLC tags / Servo Telegrams [4]

PLC tags

PLC tags

	Name	Data type	Address	Retain
	▼ Servo A SpeedAxis_Actor_Interface_AddressIn	"PD_TEL102_IN"	%I300.0	False
	▼ ZSW1	PD_ZSW1	%I300.0	
	NoSpeedDeviation	Bool	%I300.0	
	ControlRequested	Bool	%I300.1	
	SpeedComparisonValusReachedExeeded	Bool	%I300.2	
	TorqueLimitNotReached	Bool	%I300.3	
	OpenHoldingBrake	Bool	%I300.4	
	NoMotorOvertemperature	Bool	%I300.5	
	ActualSpeedPositive	Bool	%I300.6	
	NoPowerUnitOvertemperature	Bool	%I300.7	
	ReadyToSwitchOn	Bool	%I301.0	
	ReadyToOperate	Bool	%I301.1	
	OperationEnabled	Bool	%I301.2	
	FaultPresent	Bool	%I301.3	
	NoCoastStopActivated	Bool	%I301.4	
	NoQuickStopActivated	Bool	%I301.5	
	SwitchingOnInhibited	Bool	%I301.6	
	AlarmPresent	Bool	%I301.7	
	NIST_B	DWord	%ID302	
	▼ ZSW2	PD_ZSW2	%I306.0	
	TravelToFixedEndStopActive	Bool	%I306.0	
	Reserved_Bit09	Bool	%I306.1	
	PulsesEnabled	Bool	%I306.2	
	MotorDataSetChangeoverActive	Bool	%I306.3	
	SlaveSignOfLifeBit0	Bool	%I306.4	
	SlaveSignOfLifeBit1	Bool	%I306.5	
	SlaveSignOfLifeBit2	Bool	%I306.6	
	SlaveSignOfLifeBit3	Bool	%I306.7	
	DriveDataSetEffectiveBit0	Bool	%I307.0	
	DriveDataSetEffectiveBit1	Bool	%I307.1	
	DriveDataSetEffectiveBit2	Bool	%I307.2	
	DriveDataSetEffectiveBit3	Bool	%I307.3	
	DriveDataSetEffectiveBit4	Bool	%I307.4	
	AlarmClassBit0	Bool	%I307.5	

	Name	Data type	Address	Retain
	AlarmClassBit1	Bool	%I307.6	
	ParkingAxisActive	Bool	%I307.7	
	▼ MELDW	PD_MELDW	%I308.0	
	SpeedTolerance	Bool	%I308.0	
	Reserved_Bit09	Bool	%I308.1	
	Reserved_Bit10	Bool	%I308.2	
	ControllerEnable	Bool	%I308.3	
	DriveReady	Bool	%I308.4	
	PulsesEnabled	Bool	%I308.5	
	Reserved_Bit14	Bool	%I308.6	
	Reserved_Bit15	Bool	%I308.7	
	RampUpDownCompleted	Bool	%I309.0	
	TorqueUtilizationOk	Bool	%I309.1	
	ActSpeedValue3Ok	Bool	%I309.2	
	ActSpeedValue2Ok	Bool	%I309.3	
	Reserved_Bit04	Bool	%I309.4	
	VariableFunctionality	Bool	%I309.5	
	NoWarningMotorTemperature	Bool	%I309.6	
	NoPowerUnitOvertemperature	Bool	%I309.7	
	▼ G1_ZSW	PD_Gx_ZSW	%I310.0	
	Probe1Deflected	Bool	%I310.0	
	Probe2Deflected	Bool	%I310.1	
	Reserved_Bit10	Bool	%I310.2	
	EncoderFaultAcknowledgeActive	Bool	%I310.3	
	HomePositionExecuted	Bool	%I310.4	
	AbsoluteValueCyclicallyExecuted	Bool	%I310.5	
	ParkingSensorExecuted	Bool	%I310.6	
	SensorError	Bool	%I310.7	
	Function1Active	Bool	%I311.0	
	Function2Active	Bool	%I311.1	
	Function3Active	Bool	%I311.2	
	Function4Active	Bool	%I311.3	
	Value1Available	Bool	%I311.4	
	Value2Available	Bool	%I311.5	
	Value3Available	Bool	%I311.6	
	Value4Available	Bool	%I311.7	
	G1_XIST1	DWord	%ID312	
	G1_XIST2	DWord	%ID316	
	▼ Servo A SpeedAxis_Actor_Interface_AddressOut	"PD_TEL102_OUT"	%Q300.0	False

	Name	Data type	Address	Retain
	▼ STW1	PD_STW1	%Q300.0	
	Reserved_Bit08	Bool	%Q300.0	
	Reserved_Bit09	Bool	%Q300.1	
	ControlByPlc	Bool	%Q300.2	
	SetpointInversion	Bool	%Q300.3	
	OpenHoldingBrake	Bool	%Q300.4	
	RaiseMotorizedPotentiometerSetpoint	Bool	%Q300.5	
	LowerMotorizedPotentiometerSetpoint	Bool	%Q300.6	
	Reserved_Bit15	Bool	%Q300.7	
	On	Bool	%Q301.0	
	NoCoastStop	Bool	%Q301.1	
	NoQuickStop	Bool	%Q301.2	
	EnableOperation	Bool	%Q301.3	
	EnableRampGenerator	Bool	%Q301.4	
	UnfreezeRampGenerator	Bool	%Q301.5	
	EnableSetpoint	Bool	%Q301.6	
	FaultAcknowledge	Bool	%Q301.7	
	NSOLL_B	DWord	%QD302	
	▼ STW2	PD_STW2	%Q306.0	
	TravelToFixedEndstop	Bool	%Q306.0	
	Reserved_Bit09	Bool	%Q306.1	
	Reserved_Bit10	Bool	%Q306.2	
	MotorSwitchoverFinished	Bool	%Q306.3	
	MasterSignOfLifeBit0	Bool	%Q306.4	
	MasterSignOfLifeBit1	Bool	%Q306.5	
	MasterSignOfLifeBit2	Bool	%Q306.6	
	MasterSignOfLifeBit3	Bool	%Q306.7	
	DriveDataSetSelectionBit0	Bool	%Q307.0	
	DriveDataSetSelectionBit1	Bool	%Q307.1	
	DriveDataSetSelectionBit2	Bool	%Q307.2	
	DriveDataSetSelectionBit3	Bool	%Q307.3	
	DriveDataSetSelectionBit4	Bool	%Q307.4	
	Reserved_Bit05	Bool	%Q307.5	
	Reserved_Bit06	Bool	%Q307.6	
	ParkingAxisSelection	Bool	%Q307.7	
	MOMRED	Word	%QW308	
	▼ G1_STW	PD_Gx_STW	%Q310.0	
	Reserved_Bit08	Bool	%Q310.0	
	Reserved_Bit09	Bool	%Q310.1	

	Name	Data type	Address	Retain
	Reserved_Bit10	Bool	%Q310.2	
	Reserved_Bit11	Bool	%Q310.3	
	Reserved_Bit12	Bool	%Q310.4	
	AbsoluteValueCyclically	Bool	%Q310.5	
	RequestParkingEncoder	Bool	%Q310.6	
	AcknowledgeError	Bool	%Q310.7	
	Function1Request	Bool	%Q311.0	
	Function2Request	Bool	%Q311.1	
	Function3Request	Bool	%Q311.2	
	Function4Request	Bool	%Q311.3	
	Command0Request	Bool	%Q311.4	
	Command1Request	Bool	%Q311.5	
	Command2Request	Bool	%Q311.6	
	Mode	Bool	%Q311.7	
	▼ Servo B SpeedAxis_Actor_Interface_AddressIn	"PD_TEL102_IN"	%I400.0	False
	▼ ZSW1	PD_ZSW1	%I400.0	
	NoSpeedDeviation	Bool	%I400.0	
	ControlRequested	Bool	%I400.1	
	SpeedComparisonValusReachedExeeded	Bool	%I400.2	
	TorqueLimitNotReached	Bool	%I400.3	
	OpenHoldingBrake	Bool	%I400.4	
	NoMotorOvertemperature	Bool	%I400.5	
	ActualSpeedPositive	Bool	%I400.6	
	NoPowerUnitOvertemperature	Bool	%I400.7	
	ReadyToSwitchOn	Bool	%I401.0	
	ReadyToOperate	Bool	%I401.1	
	OperationEnabled	Bool	%I401.2	
	FaultPresent	Bool	%I401.3	
	NoCoastStopActivated	Bool	%I401.4	
	NoQuickStopActivated	Bool	%I401.5	
	SwitchingOnInhibited	Bool	%I401.6	
	AlarmPresent	Bool	%I401.7	
	NIST_B	DWord	%ID402	
	▼ ZSW2	PD_ZSW2	%I406.0	
	TravelToFixedEndStopActive	Bool	%I406.0	
	Reserved_Bit09	Bool	%I406.1	
	PulsesEnabled	Bool	%I406.2	
	MotorDataSetChangeoverActive	Bool	%I406.3	
	SlaveSignOfLifeBit0	Bool	%I406.4	

	Name	Data type	Address	Retain
	SlaveSignOfLifeBit1	Bool	%I406.5	
	SlaveSignOfLifeBit2	Bool	%I406.6	
	SlaveSignOfLifeBit3	Bool	%I406.7	
	DriveDataSetEffectiveBit0	Bool	%I407.0	
	DriveDataSetEffectiveBit1	Bool	%I407.1	
	DriveDataSetEffectiveBit2	Bool	%I407.2	
	DriveDataSetEffectiveBit3	Bool	%I407.3	
	DriveDataSetEffectiveBit4	Bool	%I407.4	
	AlarmClassBit0	Bool	%I407.5	
	AlarmClassBit1	Bool	%I407.6	
	ParkingAxisActive	Bool	%I407.7	
	▼ MELDW	PD_MELDW	%I408.0	
	SpeedTolerance	Bool	%I408.0	
	Reserved_Bit09	Bool	%I408.1	
	Reserved_Bit10	Bool	%I408.2	
	ControllerEnable	Bool	%I408.3	
	DriveReady	Bool	%I408.4	
	PulsesEnabled	Bool	%I408.5	
	Reserved_Bit14	Bool	%I408.6	
	Reserved_Bit15	Bool	%I408.7	
	RampUpDownCompleted	Bool	%I409.0	
	TorqueUtilizationOk	Bool	%I409.1	
	ActSpeedValue3Ok	Bool	%I409.2	
	ActSpeedValue2Ok	Bool	%I409.3	
	Reserved_Bit04	Bool	%I409.4	
	VariableFunctionality	Bool	%I409.5	
	NoWarningMotorTemperature	Bool	%I409.6	
	NoPowerUnitOvertemperature	Bool	%I409.7	
	▼ G1_ZSW	PD_Gx_ZSW	%I410.0	
	Probe1Deflected	Bool	%I410.0	
	Probe2Deflected	Bool	%I410.1	
	Reserved_Bit10	Bool	%I410.2	
	EncoderFaultAcknowledgeActive	Bool	%I410.3	
	HomePositionExecuted	Bool	%I410.4	
	AbsoluteValueCyclicallyExecuted	Bool	%I410.5	
	ParkingSensorExecuted	Bool	%I410.6	
	SensorError	Bool	%I410.7	
	Function1Active	Bool	%I411.0	
	Function2Active	Bool	%I411.1	

	Name	Data type	Address	Retain
	Function3Active	Bool	%I411.2	
	Function4Active	Bool	%I411.3	
	Value1Available	Bool	%I411.4	
	Value2Available	Bool	%I411.5	
	Value3Available	Bool	%I411.6	
	Value4Available	Bool	%I411.7	
	G1_XIST1	DWord	%ID412	
	G1_XIST2	DWord	%ID416	
	▼ Servo B SpeedAxis_Actor_Interface_AddressOut	"PD_TEL102_OUT"	%Q400.0	False
	▼ STW1	PD_STW1	%Q400.0	
	Reserved_Bit08	Bool	%Q400.0	
	Reserved_Bit09	Bool	%Q400.1	
	ControlByPlc	Bool	%Q400.2	
	SetpointInversion	Bool	%Q400.3	
	OpenHoldingBrake	Bool	%Q400.4	
	RaiseMotorizedPotentiometerSetpoint	Bool	%Q400.5	
	LowerMotorizedPotentiometerSetpoint	Bool	%Q400.6	
	Reserved_Bit15	Bool	%Q400.7	
	On	Bool	%Q401.0	
	NoCoastStop	Bool	%Q401.1	
	NoQuickStop	Bool	%Q401.2	
	EnableOperation	Bool	%Q401.3	
	EnableRampGenerator	Bool	%Q401.4	
	UnfreezeRampGenerator	Bool	%Q401.5	
	EnableSetpoint	Bool	%Q401.6	
	FaultAcknowledge	Bool	%Q401.7	
	NSOLL_B	DWord	%QD402	
	▼ STW2	PD_STW2	%Q406.0	
	TravelToFixedEndstop	Bool	%Q406.0	
	Reserved_Bit09	Bool	%Q406.1	
	Reserved_Bit10	Bool	%Q406.2	
	MotorSwitchoverFinished	Bool	%Q406.3	
	MasterSignOfLifeBit0	Bool	%Q406.4	
	MasterSignOfLifeBit1	Bool	%Q406.5	
	MasterSignOfLifeBit2	Bool	%Q406.6	
	MasterSignOfLifeBit3	Bool	%Q406.7	
	DriveDataSetSelectionBit0	Bool	%Q407.0	
	DriveDataSetSelectionBit1	Bool	%Q407.1	
	DriveDataSetSelectionBit2	Bool	%Q407.2	

	Name	Data type	Address	Retain
	DriveDataSetSelectionBit3	Bool	%Q407.3	
	DriveDataSetSelectionBit4	Bool	%Q407.4	
	Reserved_Bit05	Bool	%Q407.5	
	Reserved_Bit06	Bool	%Q407.6	
	ParkingAxisSelection	Bool	%Q407.7	
	MOMRED	Word	%QW408	
	▼ G1_STW	PD_Gx_STW	%Q410.0	
	Reserved_Bit08	Bool	%Q410.0	
	Reserved_Bit09	Bool	%Q410.1	
	Reserved_Bit10	Bool	%Q410.2	
	Reserved_Bit11	Bool	%Q410.3	
	Reserved_Bit12	Bool	%Q410.4	
	AbsoluteValueCyclically	Bool	%Q410.5	
	RequestParkingEncoder	Bool	%Q410.6	
	AcknowledgeError	Bool	%Q410.7	
	Function1Request	Bool	%Q411.0	
	Function2Request	Bool	%Q411.1	
	Function3Request	Bool	%Q411.2	
	Function4Request	Bool	%Q411.3	
	Command0Request	Bool	%Q411.4	
	Command1Request	Bool	%Q411.5	
	Command2Request	Bool	%Q411.6	
	Mode	Bool	%Q411.7	

PLC tags / Shutdown [2]

PLC tags

PLC tags

	Name	Data type	Address	Retain
	AGVPowerSW	Bool	%I1.1	False
	AGVSelfKillSW	Bool	%Q0.6	False

PLC tags / Stepper IO [4]

PLC tags

PLC tags

	Name	Data type	Address	Retain
	Unit A Steering Pot	Int	%IW20	False
	Unit B Steering Pot	Int	%IW22	False
	Stepper A Drive Analog	Int	%IW24	False
	Stepper B Drive Analog	Int	%IW26	False

PLC tags / System Bits [5]

PLC tags

PLC tags

Name	Data type	Address	Retain
System_Byte	Byte	%MB0	False
FirstScan	Bool	%M0.0	False
DiagStatusUpdate	Bool	%M0.1	False
AlwaysTRUE	Bool	%M0.2	False
AlwaysFALSE	Bool	%M0.3	False

T.4 Trace Functions

Code can be found on next page

Traces / Measurements

SteeringTest1

Signals

Name	Data type	Address	Unit	Comment
"Kinematic Values GDB".ForwardKi."X Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Y Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Yaw_v rate SET"	Real			SET rad/s
"HMI OP GDB".SteeringRPM_A	Real			ACT rpm
"HMI OP GDB".SteeringRPM_B	Real			ACT rpm
"HMI OP GDB".TractionRPM_A	Real			ACT rpm
"HMI OP GDB".TractionRPM_B	Real			ACT rpm
"HMI OP GDB".XCompVel	Real			ACT m/s
"HMI OP GDB".YCompVel	Real			ACT m/s
"Kinematic Values GDB".ReverseKi."Yaw_v rate ACT"	Real			ACT rad/s
"HMI OP GDB".SteeringAngleA	Real			ACT deg
"HMI OP GDB".SteeringAngleB	Real			ACT deg
"HMI OP GDB".YawAngleACT	Real			ACT deg
"HMI OP GDB".RunTestInProgress	Bool			

Snapshots

Name	Time stamp	Comment

Traces / Measurements

SteeringTest2

Signals

Name	Data type	Address	Unit	Comment
"Kinematic Values GDB".ForwardKi."X Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Y Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Yaw_v rate SET"	Real			SET rad/s
"HMI OP GDB".SteeringRPM_A	Real			ACT rpm
"HMI OP GDB".SteeringRPM_B	Real			ACT rpm
"HMI OP GDB".TractionRPM_A	Real			ACT rpm
"HMI OP GDB".TractionRPM_B	Real			ACT rpm
"HMI OP GDB".XCompVel	Real			ACT m/s
"HMI OP GDB".YCompVel	Real			ACT m/s
"Kinematic Values GDB".ReverseKi."Yaw_v rate ACT"	Real			ACT rad/s
"HMI OP GDB".SteeringAngleA	Real			ACT deg
"HMI OP GDB".SteeringAngleB	Real			ACT deg
"HMI OP GDB".YawAngleACT	Real			ACT deg
"HMI OP GDB".RunTestInProgress	Bool			

Snapshots

Name	Time stamp	Comment

Traces / Measurements

SteeringTest3

Signals

Name	Data type	Address	Unit	Comment
"Kinematic Values GDB".ForwardKi."X Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Y Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Yaw_v rate SET"	Real			SET rad/s
"HMI OP GDB".SteeringRPM_A	Real			ACT rpm
"HMI OP GDB".SteeringRPM_B	Real			ACT rpm
"HMI OP GDB".TractionRPM_A	Real			ACT rpm
"HMI OP GDB".TractionRPM_B	Real			ACT rpm
"HMI OP GDB".XCompVel	Real			ACT m/s
"HMI OP GDB".YCompVel	Real			ACT m/s
"Kinematic Values GDB".ReverseKi."Yaw_v rate ACT"	Real			ACT rad/s
"HMI OP GDB".SteeringAngleA	Real			ACT deg
"HMI OP GDB".SteeringAngleB	Real			ACT deg
"HMI OP GDB".YawAngleACT	Real			ACT deg
"HMI OP GDB".RunTestInProgress	Bool			

Snapshots

Name	Time stamp	Comment

Traces / Measurements

SteeringTest4

Signals

Name	Data type	Address	Unit	Comment
"Kinematic Values GDB".ForwardKi."X Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Y Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Yaw_v rate SET"	Real			SET rad/s
"HMI OP GDB".SteeringRPM_A	Real			ACT rpm
"HMI OP GDB".SteeringRPM_B	Real			ACT rpm
"HMI OP GDB".TractionRPM_A	Real			ACT rpm
"HMI OP GDB".TractionRPM_B	Real			ACT rpm
"HMI OP GDB".XCompVel	Real			ACT m/s
"HMI OP GDB".YCompVel	Real			ACT m/s
"Kinematic Values GDB".ReverseKi."Yaw_v rate ACT"	Real			ACT rad/s
"HMI OP GDB".SteeringAngleA	Real			ACT deg
"HMI OP GDB".SteeringAngleB	Real			ACT deg
"HMI OP GDB".YawAngleACT	Real			ACT deg
"HMI OP GDB".RunTestInProgress	Bool			

Snapshots

Name	Time stamp	Comment

Traces / Measurements

SteeringTest5

Signals

Name	Data type	Address	Unit	Comment
"Kinematic Values GDB".ForwardKi."X Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Y Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Yaw_v rate SET"	Real			SET rad/s
"HMI OP GDB".SteeringRPM_A	Real			ACT rpm
"HMI OP GDB".SteeringRPM_B	Real			ACT rpm
"HMI OP GDB".TractionRPM_A	Real			ACT rpm
"HMI OP GDB".TractionRPM_B	Real			ACT rpm
"HMI OP GDB".XCompVel	Real			ACT m/s
"HMI OP GDB".YCompVel	Real			ACT m/s
"Kinematic Values GDB".ReverseKi."Yaw_v rate ACT"	Real			ACT rad/s
"HMI OP GDB".SteeringAngleA	Real			ACT deg
"HMI OP GDB".SteeringAngleB	Real			ACT deg
"HMI OP GDB".YawAngleACT	Real			ACT deg
"HMI OP GDB".RunTestInProgress	Bool			

Snapshots

Name	Time stamp	Comment

Traces / Measurements

StraightLineTest1

Signals				
Name	**Data type**	**Address**	**Unit**	**Comment**
"Kinematic Values GDB".ForwardKi."X Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Y Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Yaw_v rate SET"	Real			SET rad/s
"HMI OP GDB".SteeringRPM_A	Real			ACT rpm
"HMI OP GDB".SteeringRPM_B	Real			ACT rpm
"HMI OP GDB".TractionRPM_A	Real			ACT rpm
"HMI OP GDB".TractionRPM_B	Real			ACT rpm
"HMI OP GDB".XCompVel	Real			ACT m/s
"HMI OP GDB".YCompVel	Real			ACT m/s
"Kinematic Values GDB".ReverseKi."Yaw_v rate ACT"	Real			ACT rad/s
"HMI OP GDB".SteeringAngleA	Real			ACT deg
"HMI OP GDB".SteeringAngleB	Real			ACT deg
"HMI OP GDB".YawAngleACT	Real			ACT deg
"HMI OP GDB".RunTestInProgress	Bool			

Snapshots		
Name	**Time stamp**	**Comment**

Traces / Measurements

StraightLineTest2

Signals				
Name	**Data type**	**Address**	**Unit**	**Comment**
"Kinematic Values GDB".ForwardKi."X Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Y Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Yaw_v rate SET"	Real			SET rad/s
"HMI OP GDB".SteeringRPM_A	Real			ACT rpm
"HMI OP GDB".SteeringRPM_B	Real			ACT rpm
"HMI OP GDB".TractionRPM_A	Real			ACT rpm
"HMI OP GDB".TractionRPM_B	Real			ACT rpm
"HMI OP GDB".XCompVel	Real			ACT m/s
"HMI OP GDB".YCompVel	Real			ACT m/s
"Kinematic Values GDB".ReverseKi."Yaw_v rate ACT"	Real			ACT rad/s
"HMI OP GDB".SteeringAngleA	Real			ACT deg
"HMI OP GDB".SteeringAngleB	Real			ACT deg
"HMI OP GDB".YawAngleACT	Real			ACT deg
"HMI OP GDB".RunTestInProgress	Bool			

Snapshots		
Name	**Time stamp**	**Comment**

Traces / Measurements

StraightLineTest3

Signals				
Name	**Data type**	**Address**	**Unit**	**Comment**
"Kinematic Values GDB".ForwardKi."X Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Y Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Yaw_v rate SET"	Real			SET rad/s
"HMI OP GDB".SteeringRPM_A	Real			ACT rpm
"HMI OP GDB".SteeringRPM_B	Real			ACT rpm
"HMI OP GDB".TractionRPM_A	Real			ACT rpm
"HMI OP GDB".TractionRPM_B	Real			ACT rpm
"HMI OP GDB".XCompVel	Real			ACT m/s
"HMI OP GDB".YCompVel	Real			ACT m/s
"Kinematic Values GDB".ReverseKi."Yaw_v rate ACT"	Real			ACT rad/s
"HMI OP GDB".SteeringAngleA	Real			ACT deg
"HMI OP GDB".SteeringAngleB	Real			ACT deg
"HMI OP GDB".YawAngleACT	Real			ACT deg
"HMI OP GDB".RunTestInProgress	Bool			

Snapshots		
Name	**Time stamp**	**Comment**

Traces / Measurements

StraightLineTest4

Signals

Name	Data type	Address	Unit	Comment
"Kinematic Values GDB".ForwardKi."X Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Y Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Yaw_v rate SET"	Real			SET rad/s
"HMI OP GDB".SteeringRPM_A	Real			ACT rpm
"HMI OP GDB".SteeringRPM_B	Real			ACT rpm
"HMI OP GDB".TractionRPM_A	Real			ACT rpm
"HMI OP GDB".TractionRPM_B	Real			ACT rpm
"HMI OP GDB".XCompVel	Real			ACT m/s
"HMI OP GDB".YCompVel	Real			ACT m/s
"Kinematic Values GDB".ReverseKi."Yaw_v rate ACT"	Real			ACT rad/s
"HMI OP GDB".SteeringAngleA	Real			ACT deg
"HMI OP GDB".SteeringAngleB	Real			ACT deg
"HMI OP GDB".YawAngleACT	Real			ACT deg
"HMI OP GDB".RunTestInProgress	Bool			

Snapshots

Name	Time stamp	Comment

Traces / Measurements

StraightLineTest5

Signals

Name	Data type	Address	Unit	Comment
"Kinematic Values GDB".ForwardKi."X Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Y Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Yaw_v rate SET"	Real			SET rad/s
"HMI OP GDB".SteeringRPM_A	Real			ACT rpm
"HMI OP GDB".SteeringRPM_B	Real			ACT rpm
"HMI OP GDB".TractionRPM_A	Real			ACT rpm
"HMI OP GDB".TractionRPM_B	Real			ACT rpm
"HMI OP GDB".XCompVel	Real			ACT m/s
"HMI OP GDB".YCompVel	Real			ACT m/s
"Kinematic Values GDB".ReverseKi."Yaw_v rate ACT"	Real			ACT rad/s
"HMI OP GDB".SteeringAngleA	Real			ACT deg
"HMI OP GDB".SteeringAngleB	Real			ACT deg
"HMI OP GDB".YawAngleACT	Real			ACT deg
"HMI OP GDB".RunTestInProgress	Bool			

Snapshots

Name	Time stamp	Comment

Traces / Measurements

SwerveTest1

Signals				
Name	**Data type**	**Address**	**Unit**	**Comment**
"Kinematic Values GDB".ForwardKi."X Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Y Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Yaw_v rate SET"	Real			SET rad/s
"HMI OP GDB".SteeringRPM_A	Real			ACT rpm
"HMI OP GDB".SteeringRPM_B	Real			ACT rpm
"HMI OP GDB".TractionRPM_A	Real			ACT rpm
"HMI OP GDB".TractionRPM_B	Real			ACT rpm
"HMI OP GDB".XCompVel	Real			ACT m/s
"HMI OP GDB".YCompVel	Real			ACT m/s
"Kinematic Values GDB".ReverseKi."Yaw_v rate ACT"	Real			ACT rad/s
"HMI OP GDB".SteeringAngleA	Real			ACT deg
"HMI OP GDB".SteeringAngleB	Real			ACT deg
"HMI OP GDB".YawAngleACT	Real			ACT deg
"HMI OP GDB".RunTestInProgress	Bool			

Snapshots		
Name	**Time stamp**	**Comment**

Traces / Measurements

SwerveTest2

Signals

Name	Data type	Address	Unit	Comment
"Kinematic Values GDB".ForwardKi."X Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Y Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Yaw_v rate SET"	Real			SET rad/s
"HMI OP GDB".SteeringRPM_A	Real			ACT rpm
"HMI OP GDB".SteeringRPM_B	Real			ACT rpm
"HMI OP GDB".TractionRPM_A	Real			ACT rpm
"HMI OP GDB".TractionRPM_B	Real			ACT rpm
"HMI OP GDB".XCompVel	Real			ACT m/s
"HMI OP GDB".YCompVel	Real			ACT m/s
"Kinematic Values GDB".ReverseKi."Yaw_v rate ACT"	Real			ACT rad/s
"HMI OP GDB".SteeringAngleA	Real			ACT deg
"HMI OP GDB".SteeringAngleB	Real			ACT deg
"HMI OP GDB".YawAngleACT	Real			ACT deg
"HMI OP GDB".RunTestInProgress	Bool			

Snapshots

Name	Time stamp	Comment

Traces / Measurements

SwerveTest3

Signals

Name	Data type	Address	Unit	Comment
"Kinematic Values GDB".ForwardKi."X Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Y Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Yaw_v rate SET"	Real			SET rad/s
"HMI OP GDB".SteeringRPM_A	Real			ACT rpm
"HMI OP GDB".SteeringRPM_B	Real			ACT rpm
"HMI OP GDB".TractionRPM_A	Real			ACT rpm
"HMI OP GDB".TractionRPM_B	Real			ACT rpm
"HMI OP GDB".XCompVel	Real			ACT m/s
"HMI OP GDB".YCompVel	Real			ACT m/s
"Kinematic Values GDB".ReverseKi."Yaw_v rate ACT"	Real			ACT rad/s
"HMI OP GDB".SteeringAngleA	Real			ACT deg
"HMI OP GDB".SteeringAngleB	Real			ACT deg
"HMI OP GDB".YawAngleACT	Real			ACT deg
"HMI OP GDB".RunTestInProgress	Bool			

Snapshots

Name	Time stamp	Comment

Traces / Measurements

SwerveTest4

Signals

Name	Data type	Address	Unit	Comment
"Kinematic Values GDB".ForwardKi."X Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Y Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Yaw_v rate SET"	Real			SET rad/s
"HMI OP GDB".SteeringRPM_A	Real			ACT rpm
"HMI OP GDB".SteeringRPM_B	Real			ACT rpm
"HMI OP GDB".TractionRPM_A	Real			ACT rpm
"HMI OP GDB".TractionRPM_B	Real			ACT rpm
"HMI OP GDB".XCompVel	Real			ACT m/s
"HMI OP GDB".YCompVel	Real			ACT m/s
"Kinematic Values GDB".ReverseKi."Yaw_v rate ACT"	Real			ACT rad/s
"HMI OP GDB".SteeringAngleA	Real			ACT deg
"HMI OP GDB".SteeringAngleB	Real			ACT deg
"HMI OP GDB".YawAngleACT	Real			ACT deg
"HMI OP GDB".RunTestInProgress	Bool			

Snapshots

Name	Time stamp	Comment

Traces / Measurements

SwerveTest5

Signals				
Name	**Data type**	**Address**	**Unit**	**Comment**
"Kinematic Values GDB".ForwardKi."X Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Y Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Yaw_v rate SET"	Real			SET rad/s
"HMI OP GDB".SteeringRPM_A	Real			ACT rpm
"HMI OP GDB".SteeringRPM_B	Real			ACT rpm
"HMI OP GDB".TractionRPM_A	Real			ACT rpm
"HMI OP GDB".TractionRPM_B	Real			ACT rpm
"HMI OP GDB".XCompVel	Real			ACT m/s
"HMI OP GDB".YCompVel	Real			ACT m/s
"Kinematic Values GDB".ReverseKi."Yaw_v rate ACT"	Real			ACT rad/s
"HMI OP GDB".SteeringAngleA	Real			ACT deg
"HMI OP GDB".SteeringAngleB	Real			ACT deg
"HMI OP GDB".YawAngleACT	Real			ACT deg
"HMI OP GDB".RunTestInProgress	Bool			

Snapshots		
Name	**Time stamp**	**Comment**

Traces / Measurements

ComboTest1

Signals				
Name	**Data type**	**Address**	**Unit**	**Comment**
"Kinematic Values GDB".ForwardKi."X Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Y Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Yaw_v rate SET"	Real			SET rad/s
"HMI OP GDB".SteeringRPM_A	Real			ACT rpm
"HMI OP GDB".SteeringRPM_B	Real			ACT rpm
"HMI OP GDB".TractionRPM_A	Real			ACT rpm
"HMI OP GDB".TractionRPM_B	Real			ACT rpm
"HMI OP GDB".XCompVel	Real			ACT m/s
"HMI OP GDB".YCompVel	Real			ACT m/s
"Kinematic Values GDB".ReverseKi."Yaw_v rate ACT"	Real			ACT rad/s
"HMI OP GDB".SteeringAngleA	Real			ACT deg
"HMI OP GDB".SteeringAngleB	Real			ACT deg
"HMI OP GDB".YawAngleACT	Real			ACT deg
"HMI OP GDB".RunTestInProgress	Bool			

Snapshots		
Name	**Time stamp**	**Comment**

Traces / Measurements

ComboTest2

Signals

Name	Data type	Address	Unit	Comment
"Kinematic Values GDB".ForwardKi."X Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Y Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Yaw_v rate SET"	Real			SET rad/s
"HMI OP GDB".SteeringRPM_A	Real			ACT rpm
"HMI OP GDB".SteeringRPM_B	Real			ACT rpm
"HMI OP GDB".TractionRPM_A	Real			ACT rpm
"HMI OP GDB".TractionRPM_B	Real			ACT rpm
"HMI OP GDB".XCompVel	Real			ACT m/s
"HMI OP GDB".YCompVel	Real			ACT m/s
"Kinematic Values GDB".ReverseKi."Yaw_v rate ACT"	Real			ACT rad/s
"HMI OP GDB".SteeringAngleA	Real			ACT deg
"HMI OP GDB".SteeringAngleB	Real			ACT deg
"HMI OP GDB".YawAngleACT	Real			ACT deg
"HMI OP GDB".RunTestInProgress	Bool			

Snapshots

Name	Time stamp	Comment

Traces / Measurements

ComboTest3

Signals				
Name	**Data type**	**Address**	**Unit**	**Comment**
"Kinematic Values GDB".ForwardKi."X Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Y Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Yaw_v rate SET"	Real			SET rad/s
"HMI OP GDB".SteeringRPM_A	Real			ACT rpm
"HMI OP GDB".SteeringRPM_B	Real			ACT rpm
"HMI OP GDB".TractionRPM_A	Real			ACT rpm
"HMI OP GDB".TractionRPM_B	Real			ACT rpm
"HMI OP GDB".XCompVel	Real			ACT m/s
"HMI OP GDB".YCompVel	Real			ACT m/s
"Kinematic Values GDB".ReverseKi."Yaw_v rate ACT"	Real			ACT rad/s
"HMI OP GDB".SteeringAngleA	Real			ACT deg
"HMI OP GDB".SteeringAngleB	Real			ACT deg
"HMI OP GDB".YawAngleACT	Real			ACT deg
"HMI OP GDB".RunTestInProgress	Bool			

Snapshots		
Name	**Time stamp**	**Comment**

Traces / Measurements

ComboTest4

Signals

Name	Data type	Address	Unit	Comment
"Kinematic Values GDB".ForwardKi."X Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Y Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Yaw_v rate SET"	Real			SET rad/s
"HMI OP GDB".SteeringRPM_A	Real			ACT rpm
"HMI OP GDB".SteeringRPM_B	Real			ACT rpm
"HMI OP GDB".TractionRPM_A	Real			ACT rpm
"HMI OP GDB".TractionRPM_B	Real			ACT rpm
"HMI OP GDB".XCompVel	Real			ACT m/s
"HMI OP GDB".YCompVel	Real			ACT m/s
"Kinematic Values GDB".ReverseKi."Yaw_v rate ACT"	Real			ACT rad/s
"HMI OP GDB".SteeringAngleA	Real			ACT deg
"HMI OP GDB".SteeringAngleB	Real			ACT deg
"HMI OP GDB".YawAngleACT	Real			ACT deg
"HMI OP GDB".RunTestInProgress	Bool			

Snapshots

Name	Time stamp	Comment

Traces / Measurements

ComboTest5

Signals				
Name	**Data type**	**Address**	**Unit**	**Comment**
"Kinematic Values GDB".ForwardKi."X Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Y Vel Component SET"	Real			SET m/s
"Kinematic Values GDB".ForwardKi."Yaw_v rate SET"	Real			SET rad/s
"HMI OP GDB".SteeringRPM_A	Real			ACT rpm
"HMI OP GDB".SteeringRPM_B	Real			ACT rpm
"HMI OP GDB".TractionRPM_A	Real			ACT rpm
"HMI OP GDB".TractionRPM_B	Real			ACT rpm
"HMI OP GDB".XCompVel	Real			ACT m/s
"HMI OP GDB".YCompVel	Real			ACT m/s
"Kinematic Values GDB".ReverseKi."Yaw_v rate ACT"	Real			ACT rad/s
"HMI OP GDB".SteeringAngleA	Real			ACT deg
"HMI OP GDB".SteeringAngleB	Real			ACT deg
"HMI OP GDB".YawAngleACT	Real			ACT deg
"HMI OP GDB".RunTestInProgress	Bool			

Snapshots		
Name	**Time stamp**	**Comment**

U Appendix - Siemens Software 1507S F PLC Code

U.1 Program Blocks

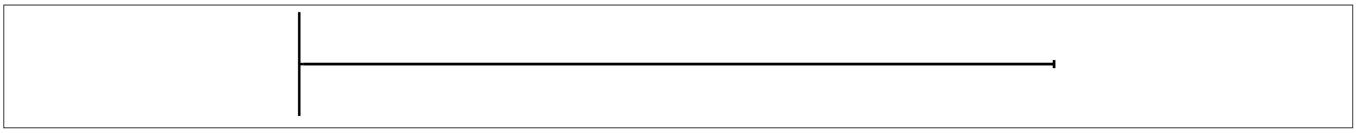
Code can be found on next page

Main [OB1]

Main Properties					
General					
Name	Main	**Number**	1	**Type**	OB
Language	LAD	**Numbering**	Automatic		
Information					
Title	"Main Program Sweep (Cycle)"	**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	**Data type**	**Default value**
▼ Input		
Initial_Call	Bool	
Remanence	Bool	
▼ Temp		
RD_SYS_T_RET_VAL	Word	
Constant		

Network 1: Send Data to Low Level PLC



FOB_RTG1 [OB123]

FOB_RTG1 Properties					
General					
Name	FOB_RTG1	**Number**	123	**Type**	OB
Language	SCL	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value
▼ Input		
Initial_Call	Bool	
Event_Count	Int	

Main_Safety_RTG1 [FB1]

Main_Safety_RTG1 Properties					
General					
Name	Main_Safety_RTG1	**Number**	1	**Type**	FB
Language	FBD	**Numbering**	Manual		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Retain
Input			
Output			
InOut			
Static			
Temp			
Constant			

Main_Safety_RTG1_DB [DB1]

Main_Safety_RTG1_DB Properties					
General					
Name	Main_Safety_RTG1_DB	**Number**	1	**Type**	DB
Language	DB	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	FUSI

Name	Data type	Start value	Retain
Input			
Output			
InOut			
Static			

Safety information: 9A0773BD Consistent; STEP 7 Safety V15.1;

00. Global DBs

This folder is empty.

U.2 PLC Tags

Code can be found on next page

PLC tags / Default tag table [71]

PLC tags

PLC tags

	Name	Data type	Address	Retain
	System_Byte	Byte	%MB0	False
	FirstScan	Bool	%M0.0	False
	DiagStatusUpdate	Bool	%M0.1	False
	AlwaysTRUE	Bool	%M0.2	False
	AlwaysFALSE	Bool	%M0.3	False
	Clock_Byte	Byte	%MB10	False
	Clock_10Hz	Bool	%M10.0	False
	Clock_5Hz	Bool	%M10.1	False
	Clock_2.5Hz	Bool	%M10.2	False
	Clock_2Hz	Bool	%M10.3	False
	Clock_1.25Hz	Bool	%M10.4	False
	Clock_1Hz	Bool	%M10.5	False
	Clock_0.625Hz	Bool	%M10.6	False
	Clock_0.5Hz	Bool	%M10.7	False

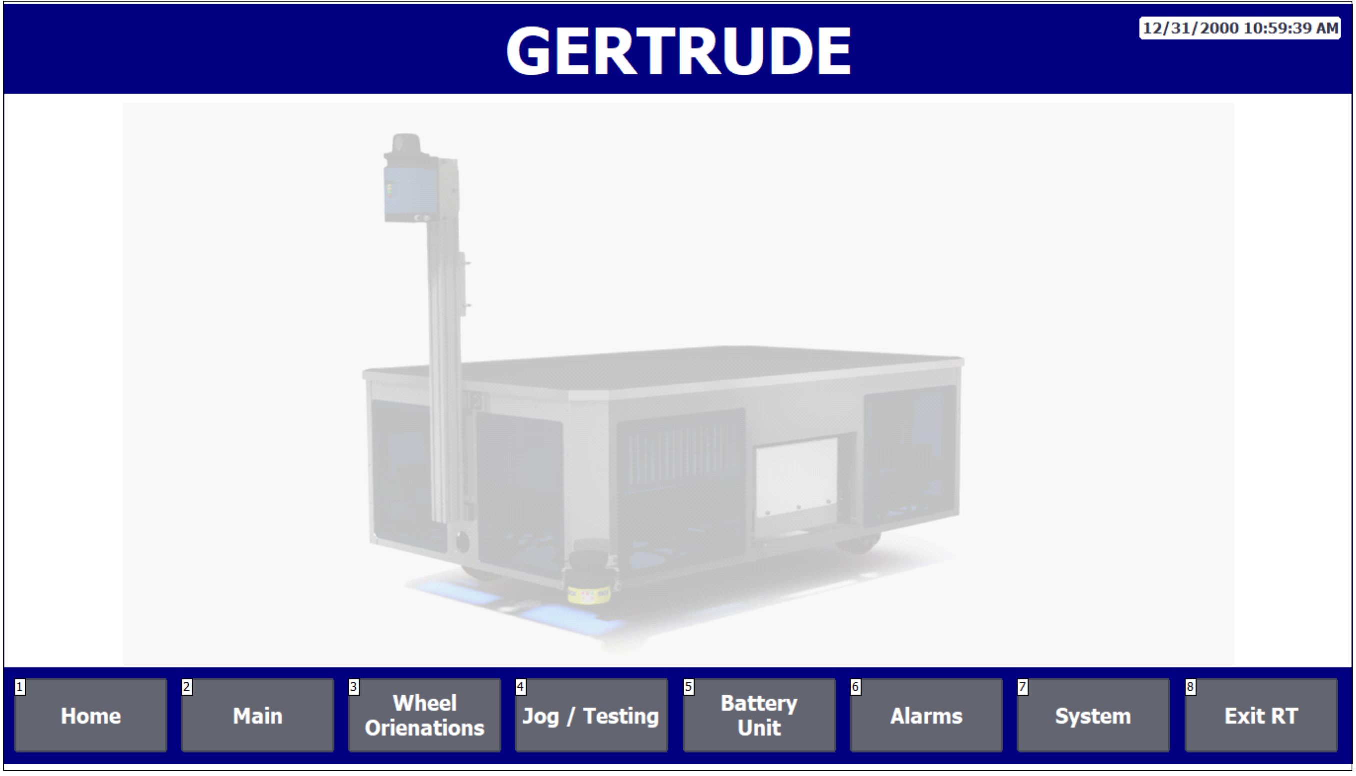
V Appendix - Siemens SCADA System

V.1 Screen Templates

Code can be found on next page

Alex Temp A

Hardcopy of Alex Temp A



Name	Alex Temp A	**Background color**	0, 0, 128
Grid color	0, 0, 0	**Tab sequence in foreground**	Enabled
Active layer	0		

Template_Rectangle_1

Type	Rectangle	**Name**	Template_Rectangle_1
X position	1	**Y position**	91
Width	1364	**Height**	579
Layer	0 - Layer_0	**Background color**	255, 255, 255
Border color	24, 28, 49		

Template_Home

Type	Button	**Name**	Template_Home
X position	11	**Y position**	681
Width	155	**Height**	75
Mode	Text	**Text OFF**	Home
Text ON	Text		

Dynamizations\Event	
Event name	Click

Function list\ResetBit

Tag	HMI OP GDB_AGVJog_SW

Function list\ResetBit

Tag	HMI OP GDB_RunTestDone LED

Function list\ActivateScreen

Screen name	Home	**Object number**	0

Button_Main

Type	Button	**Name**	Button_Main
X position	180	**Y position**	681
Width	155	**Height**	75
Mode	Text	**Text OFF**	Main
Text ON	Text		

Dynamizations\Event	
Event name	Click

Function list\ResetBit

Tag	HMI OP GDB_AGVJog_SW

Function list\ResetBit

Tag	HMI OP GDB_RunTestDone LED

Function list\GetPLCMode

Connection	HMI_SoftPLC	**Mode**	SoftPLC_Status

Function list\GetPLCMode

Connection	HMI_s7-1500	**Mode**	s7-1500_Status

Function list\ActivateScreen

Screen name	Main	**Object number**	0

Button_Wheel_Alignment

Type	Button	**Name**	Button_Wheel_Alignment
X position	349	**Y position**	681
Width	155	**Height**	75
Mode	Text	**Text OFF**	Wheel Orienations
Text ON	Text		

Dynamizations\Event	
Event name	Click

Function list\ResetBit

Tag	HMI OP GDB_AGVJog_SW

Function list\ResetBit

Tag	HMI OP GDB_RunTestDone LED

Function list\ActivateScreen

Screen name	WheelOrientation	**Object number**	0

Button_Jog/Testing

Type	Button	**Name**	Button_Jog/Testing
X position	518	**Y position**	681
Width	155	**Height**	75
Mode	Text	**Text OFF**	Jog / Testing
Text ON	Text		

Dynamizations\Event	
Event name	Click

Function list\ResetBit

Tag	HMI OP GDB_AGVJog_SW

Function list\ResetBit

Tag	HMI OP GDB_RunTestDone LED

Function list\ActivateScreen

Screen name	JogChoice	**Object number**	0

Button_BatteryUnit

Type	Button	**Name**	Button_BatteryUnit
X position	688	**Y position**	681
Width	155	**Height**	75
Mode	Text	**Text OFF**	Battery Unit
Text ON	Text		

Dynamizations\Event	
Event name	Click

Function list\ResetBit

Tag	HMI OP GDB_AGVJog_SW

Function list\ResetBit

Tag	HMI OP GDB_RunTestDone LED

Function list\ActivateScreen

Screen name	BatteryUnit	**Object number**	0

Button_Alarms

Type	Button	**Name**	Button_Alarms
X position	857	**Y position**	681

Width	155	**Height**	75
Mode	Text	**Text OFF**	Alarms
Text ON	Text		

Dynamizations\Event	
Event name	Click

Function list\ResetBit

Tag	HMI OP GDB_AGVJog_SW

Function list\ResetBit

Tag	HMI OP GDB_RunTestDone LED

Function list\ActivateScreen

Screen name	Alarms	**Object number**	0

Button_System

Type	Button	**Name**	Button_System
X position	1026	**Y position**	681
Width	155	**Height**	75
Mode	Text	**Text OFF**	System
Text ON	Text		

Dynamizations\Event	
Event name	Click

Function list\ResetBit

Tag	HMI OP GDB_AGVJog_SW

Function list\ResetBit

Tag	HMI OP GDB_RunTestDone LED

Function list\ActivateScreen

Screen name	System	**Object number**	0

Button_Exit_RT

Type	Button	**Name**	Button_Exit_RT
X position	1196	**Y position**	681
Width	155	**Height**	75
Mode	Text	**Text OFF**	Exit RT
Text ON	Text		

Dynamizations\Event	
Event name	Click

Function list\ResetBit

Tag	HMI OP GDB_AGVJog_SW

Function list\ResetBit

Tag	HMI OP GDB_RunTestDone LED

Function list\StopRuntime

Mode	Runtime

HeadingText_Gertrude

Type	Text field	**Name**	HeadingText_Gertrude
X position	505	**Y position**	5
Width	357	**Height**	81
Layer	0 - Layer_0	**Font**	Tahoma, 64px, style=Bold
Text	GERTRUDE		

Template_Date/time field_1

Type	Date/time field	**Name**	Template_Date/time field_1
X position	1150	**Y position**	13
Width	204	**Height**	23
Font	Tahoma, 15px, style=Bold	**Mode**	Output

AGV_WatermarkPicture

Type	Graphic view	**Name**	AGV_WatermarkPicture
X position	121	**Y position**	100
Width	1125	**Height**	568
Layer	0 - Layer_0	**Graphic**	Complete AGV 06102021 transparent
Fit graphic to size	Stretch graphic		

Template_Circle_1

Type	Circle	**Name**	Template_Circle_1
X position	2	**Y position**	103
Width	21	**Height**	21
Radius	10	**Background color**	217, 217, 217
Border color	24, 28, 49		
Dynamizations\Appearance			
Tag - Cycle	ShutdownIPC_Trigger -	**Data type**	Range
Range	0..0	**Foreground color**	24, 28, 49
Background color	217, 217, 217	**Flashing**	No
Range	1..1	**Foreground color**	24, 28, 49
Background color	255, 255, 0	**Flashing**	No

Template_Circle_2

Type	Circle	**Name**	Template_Circle_2
X position	28	**Y position**	103
Width	21	**Height**	21
Radius	10	**Background color**	217, 217, 217
Border color	24, 28, 49		
Dynamizations\Appearance			
Tag - Cycle	ShutdownRouter_Trigger -	**Data type**	Range
Range	0..0	**Foreground color**	24, 28, 49
Background color	217, 217, 217	**Flashing**	No

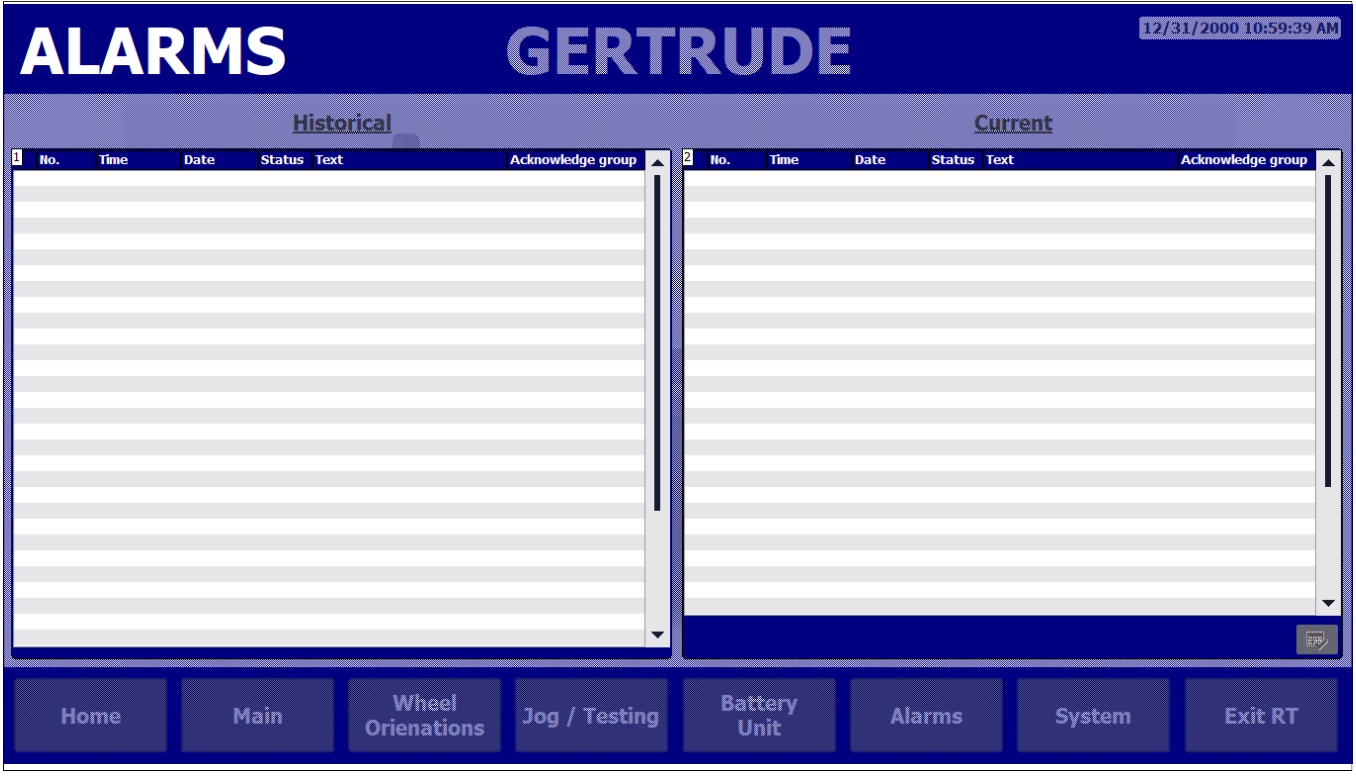
Range	1..1	**Foreground color**	24, 28, 49
Background color	255, 255, 0	**Flashing**	No

V.2 HMI Screens

Code can be found on next page

Alarms

Hardcopy of Alarms



Name	Alarms	**Background color**	0, 0, 128
Grid color	0, 0, 0	**Number**	7
Template	Alex Temp A	**Tooltip**	

HistoricalAlarmView

Type	Alarm view	**Name**	HistoricalAlarmView
X position	8	**Y position**	146
Width	669	**Height**	516
Layer	0 - Layer_0	**Source of alarms**	AlarmBuffer
Table font	Tahoma, 13px		

CurrentAlarmView

Type	Alarm view	**Name**	CurrentAlarmView
X position	687	**Y position**	146
Width	669	**Height**	516
Layer	0 - Layer_0	**Source of alarms**	Alarms
Table font	Tahoma, 13px		

HeadingText_Historical

Type	Text field	**Name**	HeadingText_Historical
X position	290	**Y position**	105
Width	105	**Height**	30

Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Historical		

HeadinagText_Current

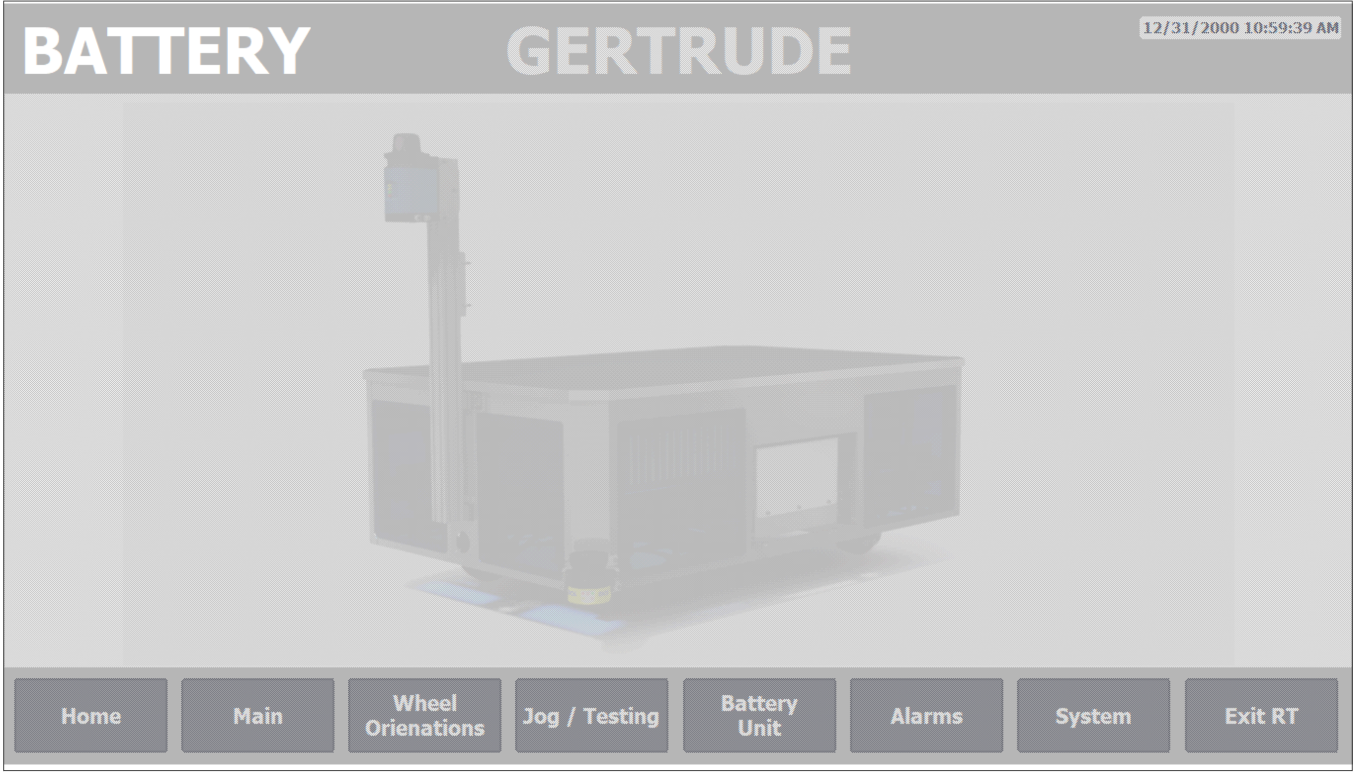
Type	Text field	**Name**	HeadinagText_Current
X position	980	**Y position**	105
Width	84	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Current		

HeadingText_ALARMS

Type	Text field	**Name**	HeadingText_ALARMS
X position	14	**Y position**	5
Width	274	**Height**	81
Layer	0 - Layer_0	**Font**	Tahoma, 64px, style=Bold
Text	ALARMS		

BatteryUnit

Hardcopy of BatteryUnit



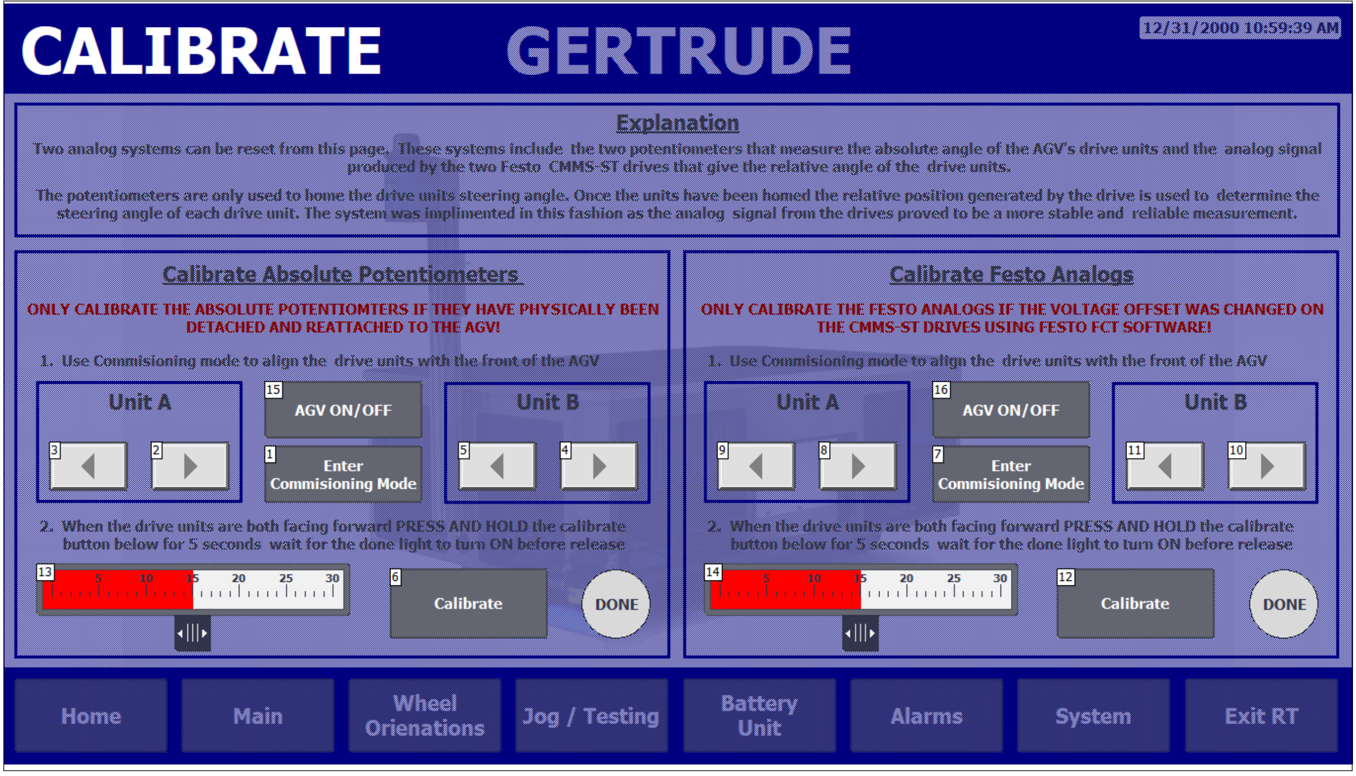
Name	BatteryUnit	**Background color**	182, 182, 182
Grid color	0, 0, 0	**Number**	8
Template	Alex Temp A	**Tooltip**	

Template_Text field_1

Type	Text field	**Name**	Template_Text field_1
X position	14	**Y position**	5
Width	299	**Height**	81
Layer	0 - Layer_0	**Font**	Tahoma, 64px, style=Bold
Text	BATTERY		

CalibrateAnalogs

Hardcopy of CalibrateAnalogs



Explanation

Two analog systems can be reset from this page. These systems include the two potentiometers that measure the absolute angle of the AGV's drive units and the analog signal produced by the two Festo CMMS-ST drives that give the relative angle of the drive units.

The potentiometers are only used to home the drive units steering angle. Once the units have been homed the relative position generated by the drive is used to determine the steering angle of each drive unit. The system was implimented in this fashion as the analog signal from the drives proved to be a more stable and reliable measurement.

Calibrate Absolute Potentiometers

ONLY CALIBRATE THE ABSOLUTE POTENTIOMTERS IF THEY HAVE PHYSICALLY BEEN DETACHED AND REATTACHED TO THE AGV!

1. Use Commisioning mode to align the drive units with the front of the AGV

2. When the drive units are both facing forward PRESS AND HOLD the calibrate button below for 5 seconds wait for the done light to turn ON before release

Calibrate Festo Analogs

ONLY CALIBRATE THE FESTO ANALOGS IF THE VOLTAGE OFFSET WAS CHANGED ON THE CMMS-ST DRIVES USING FESTO FCT SOFTWARE!

1. Use Commisioning mode to align the drive units with the front of the AGV

2. When the drive units are both facing forward PRESS AND HOLD the calibrate button below for 5 seconds wait for the done light to turn ON before release

Name	CalibrateAnalogs	**Background color**	0, 0, 128
Grid color	0, 0, 0	**Number**	5
Template	Alex Temp A	**Tooltip**	

Rectangle_1

Type	Rectangle	**Name**	Rectangle_1
X position	11	**Y position**	100
Width	1342	**Height**	136
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

HeadingText_Explination

Type	Text field	**Name**	HeadingText_Explination
X position	617	**Y position**	105
Width	130	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Explanation		

ExplinationText_1

Type	Text field	**Name**	ExplinationText_1
X position	27	**Y position**	135
Width	1310	**Height**	40

Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Two analog systems can be reset from this page. These systems include the two potentiometers that measure the absolute angle of the AGV's drive units and the analog signal produced by the two Festo CMMS-ST drives that give the relative angle of the drive units.		

ExplinationText_2

Type	Text field	**Name**	ExplinationText_2
X position	30	**Y position**	182
Width	1304	**Height**	40
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	The potentiometers are only used to home the drive units steering angle. Once the units have been homed the relative position generated by the drive is used to determine the steering angle of each drive unit. The system was implimented in this fashion as the analog signal from the drives proved to be a more stable and reliable measurement.		

CalibrateAbsolutePotentiometers_Step1_Text

Type	Text field	**Name**	CalibrateAbsolutePotentiometers_Step1_Text
X position	33	**Y position**	349
Width	572	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	1. Use Commisioning mode to align the drive units with the front of the AGV		

Rectangle_2

Type	Rectangle	**Name**	Rectangle_2
X position	11	**Y position**	249
Width	664	**Height**	412
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

Rectangle_3

Type	Rectangle	**Name**	Rectangle_3
X position	688	**Y position**	249
Width	664	**Height**	412
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

HeadingText_Calibrate_Absolute_Potentiometers

Type	Text field	**Name**	HeadingText_Calibrate_Absolute_Potentiometers
X position	158	**Y position**	258

Width	371	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Calibrate Absolute Potentiometers		

CalibrateAbsolutePotentiometes_RED_text

Type	Text field	**Name**	CalibrateAbsolutePotentiometes_RED_text
X position	21	**Y position**	297
Width	645	**Height**	40
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	ONLY CALIBRATE THE ABSOLUTE POTENTIOMTERS IF THEY HAVE PHYSICALLY BEEN DETACHED AND REATTACHED TO THE AGV!		

Button_CAP_Enter_Commisioning_Mode

Type	Button	**Name**	Button_CAP_Enter_Commisioning_Mode
X position	264	**Y position**	446
Width	160	**Height**	58
Mode	Text	**Text OFF**	Enter Commisioning Mode
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_CommisinMode_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_CommisinMode_SW

Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_CommisionModeLED -	**Data type**	Range
Range	0..0	**Foreground color**	255, 255, 255
Background color	99, 101, 113	**Flashing**	No
Range	1..1	**Foreground color**	255, 255, 255
Background color	51, 153, 102	**Flashing**	No

Rectangle_4

Type	Rectangle	**Name**	Rectangle_4
X position	33	**Y position**	381
Width	209	**Height**	123
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

Rectangle_5

Type	Rectangle	**Name**	Rectangle_5
X position	446	**Y position**	382
Width	209	**Height**	123
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

HeadingText_CAP_Unit_A

Type	Text field	**Name**	HeadingText_CAP_Unit_A
X position	103	**Y position**	387
Width	69	**Height**	29
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold
Text	Unit A		

HeadingText_CAP_UnitB

Type	Text field	**Name**	HeadingText_CAP_UnitB
X position	516	**Y position**	387
Width	69	**Height**	29
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold
Text	Unit B		

Button_Unit_A_CAP_Right_Arrow

Type	Button	**Name**	Button_Unit_A_CAP_Right_Arrow
X position	149	**Y position**	442
Width	80	**Height**	50
Mode	Check back with graphic	**Text OFF**	Text
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_UnitA_Cali_CW_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_UnitA_Cali_CW_SW

Button_Unit_A_CAP_Left_Arrow

Type	Button	**Name**	Button_Unit_A_CAP_Left_Arrow
X position	46	**Y position**	442
Width	80	**Height**	50
Mode	Check back with graphic	**Text OFF**	Text
Text ON	Text		

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_UnitA_Cali_CCW_SW

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_UnitA_Cali_CCW_SW

Button_Unit_B_CAP_Right_Arrow

Type	Button	**Name**	Button_Unit_B_CAP_Right_Arrow
X position	563	**Y position**	442
Width	80	**Height**	50
Mode	Check back with graphic	**Text OFF**	Text
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_UnitB_Cali_CW_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_UnitB_Cali_CW_SW

Button_Unit_B_CAP_Left_Arrow

Type	Button	**Name**	Button_Unit_B_CAP_Left_Arrow
X position	460	**Y position**	442
Width	80	**Height**	50
Mode	Check back with graphic	**Text OFF**	Text
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_UnitB_Cali_CCW_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_UnitB_Cali_CCW_SW

CalibrateAbsolutePotentiometers_Step2_Text

Type	Text field	**Name**	CalibrateAbsolutePotentiometers_Step2_Text

X position	33	**Y position**	516
Width	603	**Height**	40
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	2. When the drive units are both facing forward PRESS AND HOLD the calibrate button below for 5 seconds wait for the done light to turn ON before release		

Button_CAP_Calibrate

Type	Button	**Name**	Button_CAP_Calibrate
X position	391	**Y position**	570
Width	160	**Height**	71
Mode	Text	**Text OFF**	Calibrate
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_CalibratePots_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_CalibratePots_SW

Indictor_CAP_DONE

Type	Circle	**Name**	Indictor_CAP_DONE
X position	585	**Y position**	571
Width	70	**Height**	70
Radius	35	**Background color**	217, 217, 217
Border color	24, 28, 49		
Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_CalibratePots_DONE_LED -	**Data type**	Range
Range	0..0	**Foreground color**	24, 28, 49
Background color	217, 217, 217	**Flashing**	No
Range	1..1	**Foreground color**	24, 28, 49
Background color	255, 255, 0	**Flashing**	No

Text_CAP_DONE_Indicator

Type	Text field	**Name**	Text_CAP_DONE_Indicator
X position	596	**Y position**	595
Width	49	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	DONE		

HeadingText_Calibrate_Festo_Analogs

Type	Text field	**Name**	HeadingText_Calibrate_Festo_Analogs
X position	894	**Y position**	258
Width	252	**Height**	30

Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Calibrate Festo Analogs		

CalibrateFestoAnalogs_RED_Text

Type	Text field	**Name**	CalibrateFestoAnalogs_RED_Text
X position	703	**Y position**	297
Width	640	**Height**	40
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	ONLY CALIBRATE THE FESTO ANALOGS IF THE VOLTAGE OFFSET WAS CHANGED ON THE CMMS-ST DRIVES USING FESTO FCT SOFTWARE!		

CalibrateFestoAnalogs_Step1_Text

Type	Text field	**Name**	CalibrateFestoAnalogs_Step1_Text
X position	709	**Y position**	349
Width	572	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	1. Use Commisioning mode to align the drive units with the front of the AGV		

Button_CFA_Enter_Commisioning_Mode

Type	Button	**Name**	Button_CFA_Enter_Commisioning_Mode
X position	940	**Y position**	446
Width	160	**Height**	58
Mode	Text	**Text OFF**	Enter Commisioning Mode
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_CommisinMode_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_CommisinMode_SW

Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_CommisionModeLED -	**Data type**	Range
Range	0..0	**Foreground color**	255, 255, 255
Background color	99, 101, 113	**Flashing**	No
Range	1..1	**Foreground color**	255, 255, 255
Background color	51, 153, 102	**Flashing**	No

Rectangle_6

Type	Rectangle	**Name**	Rectangle_6

X position	709	**Y position**	381
Width	209	**Height**	123
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

Rectangle_7

Type	Rectangle	**Name**	Rectangle_7
X position	1122	**Y position**	382
Width	209	**Height**	123
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

HeadingText_CFA_Unit_A

Type	Text field	**Name**	HeadingText_CFA_Unit_A
X position	779	**Y position**	387
Width	69	**Height**	29
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold
Text	Unit A		

HeadingText_CFA_Unit_B

Type	Text field	**Name**	HeadingText_CFA_Unit_B
X position	1192	**Y position**	387
Width	69	**Height**	29
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold
Text	Unit B		

Button_Unit_A_CFA_Right_Arrow

Type	Button	**Name**	Button_Unit_A_CFA_Right_Arrow
X position	825	**Y position**	442
Width	80	**Height**	50
Mode	Check back with graphic	**Text OFF**	Text
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_UnitA_Cali_CW_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_UnitA_Cali_CW_SW

Button_Unit_A_CFA_Left_Arrow

Type	Button	**Name**	Button_Unit_A_CFA_Left_Arrow
X position	722	**Y position**	442
Width	80	**Height**	50
Mode	Check back with graphic	**Text OFF**	Text

Text ON	Text

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_UnitA_Cali_CCW_SW

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_UnitA_Cali_CCW_SW

Button_Unit_B_CFA_Right_Arrow

Type	Button	**Name**	Button_Unit_B_CFA_Right_Arrow
X position	1239	**Y position**	442
Width	80	**Height**	50
Mode	Check back with graphic	**Text OFF**	Text
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_UnitB_Cali_CW_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_UnitB_Cali_CW_SW

Button_Unit_B_CFA_Left_Arrow

Type	Button	**Name**	Button_Unit_B_CFA_Left_Arrow
X position	1136	**Y position**	442
Width	80	**Height**	50
Mode	Check back with graphic	**Text OFF**	Text
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_UnitB_Cali_CCW_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_UnitB_Cali_CCW_SW

CalibrateFestoAnalogs_Step2_Text

Type	Text field	**Name**	CalibrateFestoAnalogs_Step2_Text
X position	709	**Y position**	516
Width	603	**Height**	40
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	2. When the drive units are both facing forward PRESS AND HOLD the calibrate button below for 5 seconds wait for the done light to turn ON before release		

Button_CFA_Calibrate

Type	Button	**Name**	Button_CFA_Calibrate
X position	1066	**Y position**	570
Width	160	**Height**	71
Mode	Text	**Text OFF**	Calibrate
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_CalibrateFCTAnalogs_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_CalibrateFCTAnalogs_SW

Indictor_CFA_DONE

Type	Circle	**Name**	Indictor_CFA_DONE
X position	1261	**Y position**	571
Width	70	**Height**	70
Radius	35	**Background color**	217, 217, 217
Border color	24, 28, 49		
Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_CalibrateFCTAnalogs_DONE_LED -	**Data type**	Range
Range	0..0	**Foreground color**	24, 28, 49
Background color	217, 217, 217	**Flashing**	No
Range	1..1	**Foreground color**	24, 28, 49
Background color	255, 255, 0	**Flashing**	No

Text_CFA_DONE_Indicator

Type	Text field	**Name**	Text_CFA_DONE_Indicator
X position	1272	**Y position**	595
Width	49	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	DONE		

Slider_CAP

Type	Slider	**Name**	Slider_CAP
X position	33	**Y position**	565
Width	318	**Height**	53
Layer	0 - Layer_0	**Minimum value**	0
Maximum value	30	**Process value**	15
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_CalibrateStepprSpd

Slider_CFA

Type	Slider	**Name**	Slider_CFA
X position	709	**Y position**	565
Width	318	**Height**	53
Layer	0 - Layer_0	**Minimum value**	0
Maximum value	30	**Process value**	15
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_CalibrateStepprSpd

Button_CAP_AGV_ON/OFF

Type	Button	**Name**	Button_CAP_AGV_ON/OFF
X position	264	**Y position**	381
Width	160	**Height**	58
Mode	Text	**Text OFF**	AGV ON/OFF
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_AGVON/OFF_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_AGVON/OFF_SW

Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_AGVON/OFF_SW_FeedBCK -	**Data type**	Range
Range	0..0	**Foreground color**	255, 255, 255
Background color	99, 101, 113	**Flashing**	No
Range	1..1	**Foreground color**	255, 255, 255
Background color	51, 153, 102	**Flashing**	No

Button_CFA_AGV_ON/OFF

Type	Button	**Name**	Button_CFA_AGV_ON/OFF
X position	940	**Y position**	381
Width	160	**Height**	58
Mode	Text	**Text OFF**	AGV ON/OFF
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_AGVON/OFF_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_AGVON/OFF_SW

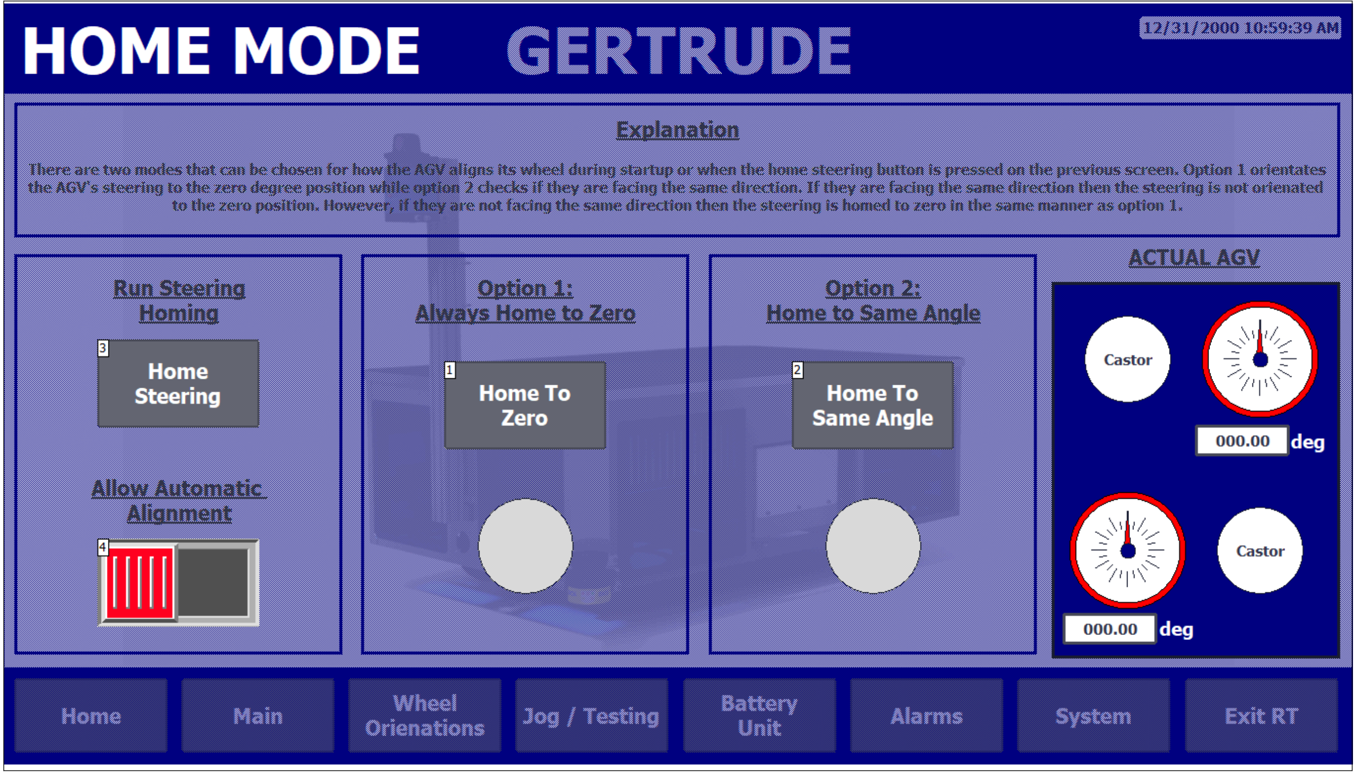
Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_AGVON/OFF_SW_FeedBCK -	**Data type**	Range
Range	0..0	**Foreground color**	255, 255, 255
Background color	99, 101, 113	**Flashing**	No
Range	1..1	**Foreground color**	255, 255, 255
Background color	51, 153, 102	**Flashing**	No

HeadingText_CALIBRATE

Type	Text field	**Name**	HeadingText_CALIBRATE
X position	14	**Y position**	5
Width	372	**Height**	81
Layer	0 - Layer_0	**Font**	Tahoma, 64px, style=Bold
Text	CALIBRATE		

ChoseOrintMode

Hardcopy of ChoseOrintMode



HOME MODE GERTRUDE

12/31/2000 10:59:39 AM

Explanation

There are two modes that can be chosen for how the AGV aligns its wheel during startup or when the home steering button is pressed on the previous screen. Option 1 orientates the AGV's steering to the zero degree position while option 2 checks if they are facing the same direction. If they are facing the same direction then the steering is not orienated to the zero position. However, if they are not facing the same direction then the steering is homed to zero in the same manner as option 1.

Name	ChoseOrintMode	**Background color**	0, 0, 128
Grid color	0, 0, 0	**Number**	12
Template	Alex Temp A	**Tooltip**	

Rectangle_1

Type	Rectangle	**Name**	Rectangle_1
X position	11	**Y position**	100
Width	1342	**Height**	136
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

HeadingText_Explanation

Type	Text field	**Name**	HeadingText_Explanation
X position	617	**Y position**	112
Width	130	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Explanation		

Explination_Text

Type	Text field	**Name**	Explination_Text
X position	22	**Y position**	156
Width	1320	**Height**	58

Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	There are two modes that can be chosen for how the AGV aligns its wheel during startup or when the home steering button is pressed on the previous screen. Option 1 orientates the AGV's steering to the zero degree position while option 2 checks if they are facing the same direction. If they are facing the same direction then the steering is not orienated to the zero position. However, if they are not facing the same direction then the steering is homed to zero in the same manner as option 1.		

Rectangle_2

Type	Rectangle	**Name**	Rectangle_2
X position	11	**Y position**	253
Width	332	**Height**	404
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

Rectangle_5

Type	Rectangle	**Name**	Rectangle_5
X position	1061	**Y position**	281
Width	292	**Height**	380
Layer	0 - Layer_0	**Background color**	0, 0, 128
Border color	24, 28, 49		

HeadingText_Actual_AGV

Type	Text field	**Name**	HeadingText_Actual_AGV
X position	1136	**Y position**	241
Width	138	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	ACTUAL AGV		

Gauge_UnitB_Angle

Type	Gauge	**Name**	Gauge_UnitB_Angle
X position	1078	**Y position**	492
Width	120	**Height**	120
Minimum value	-360	**Maximum value**	0
Process value	50		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_SteeringAngleB_Dial

I/O_Field_UnitB_Angle

Type	I/O field	**Name**	I/O_Field_UnitB_Angle
X position	1072	**Y position**	615
Width	96	**Height**	32
Layer	0 - Layer_0	**Mode**	Output
Font	Tahoma, 15px, style=Bold		

Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_SteeringAngleB

Text_deg1

Type	Text field	**Name**	Text_deg1
X position	1167	**Y position**	617
Width	40	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	deg		

Gauge_UnitA_Angle

Type	Gauge	**Name**	Gauge_UnitA_Angle
X position	1212	**Y position**	299
Width	120	**Height**	120
Minimum value	-360	**Maximum value**	0
Process value	50		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_SteeringAngleA_Dial

I/O_Field_UnitA_Angle

Type	I/O field	**Name**	I/O_Field_UnitA_Angle
X position	1206	**Y position**	425
Width	96	**Height**	32
Layer	0 - Layer_0	**Mode**	Output
Font	Tahoma, 15px, style=Bold		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_SteeringAngleA

Text_deg2

Type	Text field	**Name**	Text_deg2
X position	1300	**Y position**	428
Width	40	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	deg		

Circle_1

Type	Circle	**Name**	Circle_1
X position	1094	**Y position**	315
Width	88	**Height**	88
Radius	44	**Background color**	255, 255, 255
Border color	24, 28, 49		

Text_Castor2

Type	Text field	**Name**	Text_Castor2
X position	1111	**Y position**	348
Width	54	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Castor		

Circle_2

Type	Circle	**Name**	Circle_2
X position	1228	**Y position**	508
Width	88	**Height**	88
Radius	44	**Background color**	255, 255, 255
Border color	24, 28, 49		

Text_Castor1

Type	Text field	**Name**	Text_Castor1
X position	1245	**Y position**	541
Width	54	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Castor		

Rectangle_3

Type	Rectangle	**Name**	Rectangle_3
X position	362	**Y position**	253
Width	332	**Height**	404
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

Rectangle_4

Type	Rectangle	**Name**	Rectangle_4
X position	714	**Y position**	253
Width	332	**Height**	404
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

HeadingText_Option1:...

Type	Text field	**Name**	HeadingText_Option1:...
X position	414	**Y position**	271
Width	228	**Height**	56
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Option 1: Always Home to Zero		

HeadingText_Option2:...

Type	Text field	**Name**	HeadingText_Option2:...
X position	769	**Y position**	271
Width	222	**Height**	56
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Option 2: Home to Same Angle		

HeadingText_Run_Steering_Home

Type	Text field	**Name**	HeadingText_Run_Steering_Home
X position	108	**Y position**	271
Width	139	**Height**	56
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline

Text	Run Steering Homing

Button_Home_To_Zero

Type	Button	**Name**	Button_Home_To_Zero
X position	446	**Y position**	361
Width	164	**Height**	89
Mode	Text	**Text OFF**	Home To Zero
Text ON	Text		

Dynamizations\Event	
Event name	Click

Function list\SetBit

Tag	HMI OP GDB_Home2Zero SW

Function list\ResetBit

Tag	HMI OP GDB_Home2SameAng SW

Button_Home_To_Same_Angle

Type	Button	**Name**	Button_Home_To_Same_Angle
X position	798	**Y position**	361
Width	164	**Height**	89
Mode	Text	**Text OFF**	Home To Same Angle
Text ON	Text		

Dynamizations\Event	
Event name	Click

Function list\SetBit

Tag	HMI OP GDB_Home2SameAng SW

Function list\ResetBit

Tag	HMI OP GDB_Home2Zero SW

Indiactor_Option1

Type	Circle	**Name**	Indiactor_Option1
X position	480	**Y position**	499
Width	97	**Height**	97
Radius	48	**Background color**	217, 217, 217
Border color	24, 28, 49		
Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_Home2Zero SW -	**Data type**	Range
Range	0..0	**Foreground color**	24, 28, 49
Background color	217, 217, 217	**Flashing**	No
Range	1..1	**Foreground color**	24, 28, 49
Background color	255, 255, 0	**Flashing**	No

Indicator_Option2

Type	Circle	**Name**	Indicator_Option2
X position	832	**Y position**	499
Width	97	**Height**	97
Radius	48	**Background color**	217, 217, 217
Border color	24, 28, 49		
Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_Home2SameAng SW -	**Data type**	Range
Range	0..0	**Foreground color**	24, 28, 49
Background color	217, 217, 217	**Flashing**	No
Range	1..1	**Foreground color**	24, 28, 49
Background color	255, 255, 0	**Flashing**	No

Button_Home_Steering

Type	Button	**Name**	Button_Home_Steering
X position	95	**Y position**	339
Width	164	**Height**	89
Mode	Text	**Text OFF**	Home Steering
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_HomeSteering_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_HomeSteering_SW

HeadingText_Allow_Automatic_Alignment

Type	Text field	**Name**	HeadingText_Allow_Automatic_Alignment
X position	86	**Y position**	473
Width	183	**Height**	56
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Allow Automatic Alignment		

Switch_Allow_Auto_Alignment

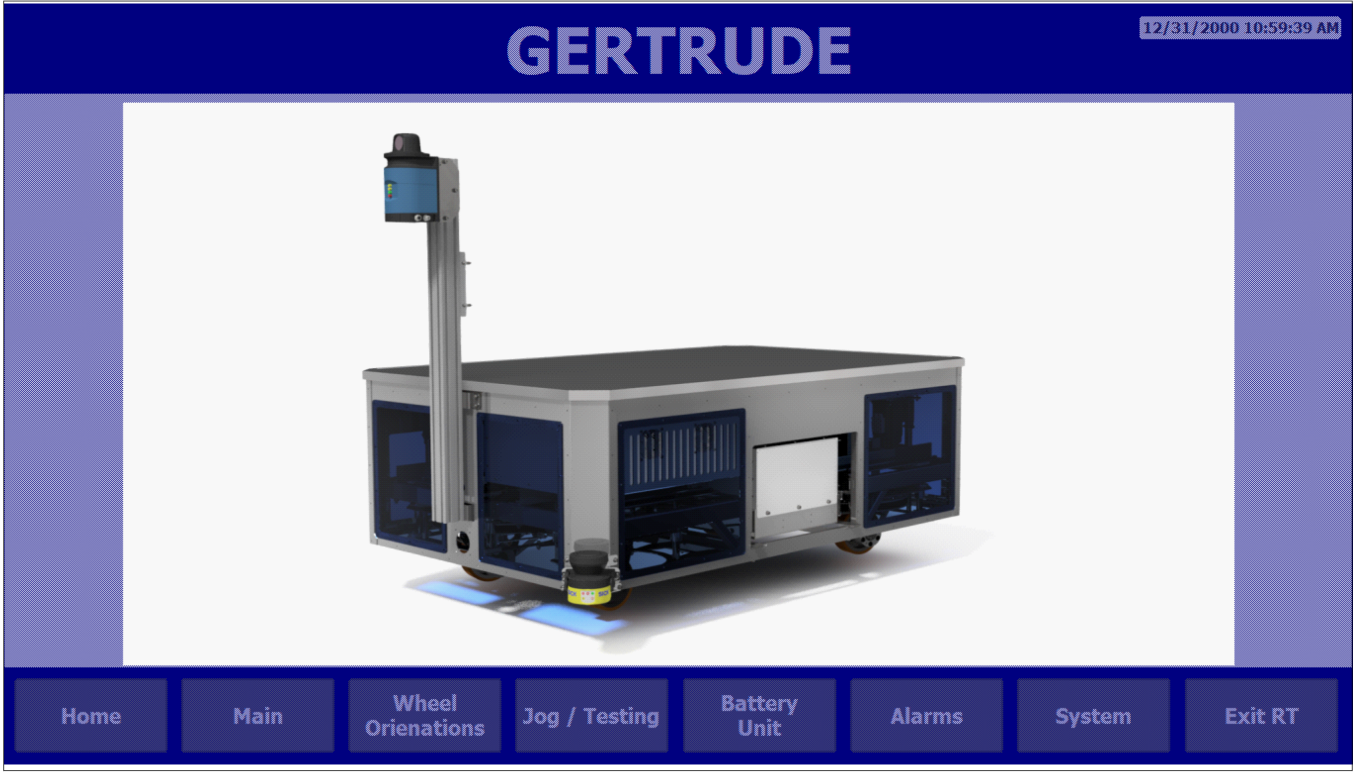
Type	Switch	**Name**	Switch_Allow_Auto_Alignment
X position	95	**Y position**	540
Width	164	**Height**	89
Layer	0 - Layer_0	**Mode**	Switch with graphic
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_AllowAlignmnt

HeadingText_HOME_MODE

Type	Text field	**Name**	HeadingText_HOME_MODE
X position	14	**Y position**	5
Width	411	**Height**	81
Layer	0 - Layer_0	**Font**	Tahoma, 64px, style=Bold
Text	HOME MODE		

Home

Hardcopy of Home



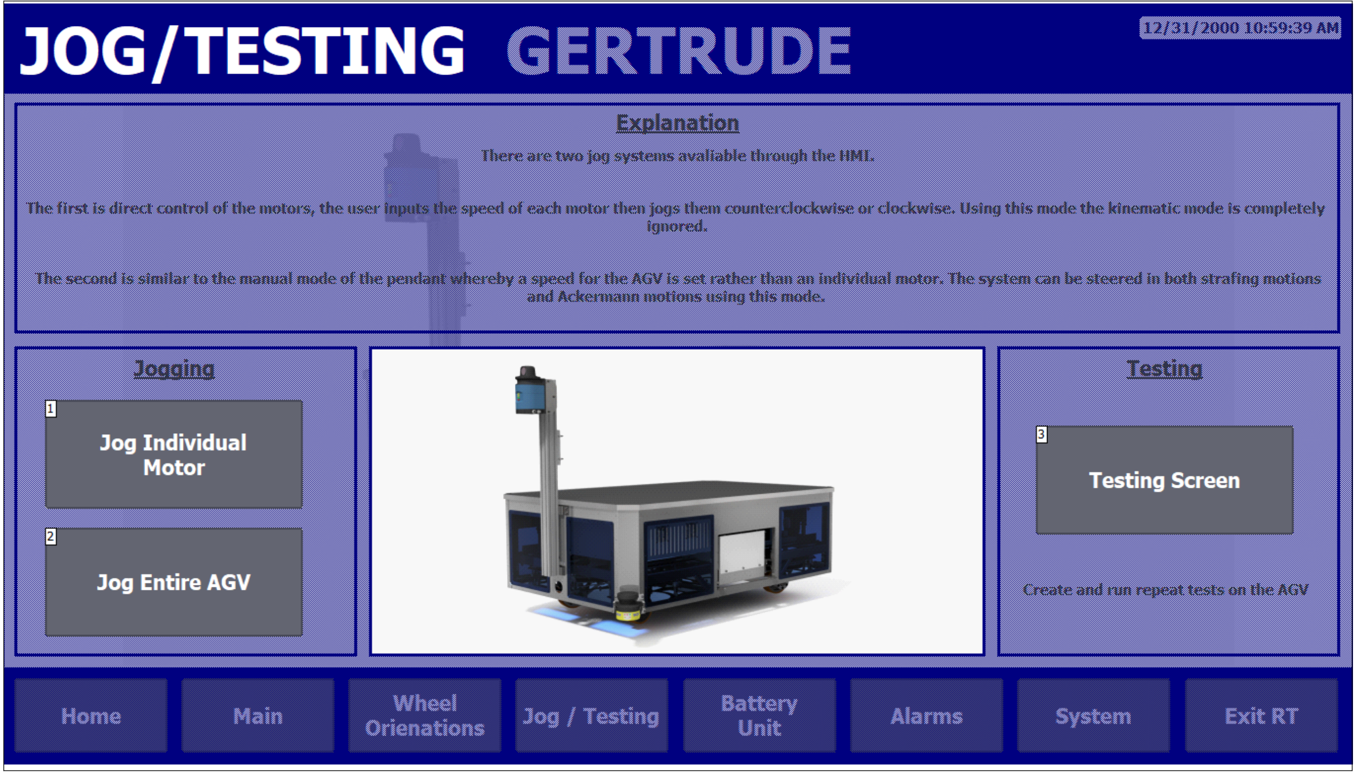
Name	Home	**Background color**	0, 0, 128
Grid color	0, 0, 0	**Number**	1
Template	Alex Temp A	**Tooltip**	

AGVPicture

Type	Graphic view	**Name**	AGVPicture
X position	121	**Y position**	100
Width	1125	**Height**	568
Layer	0 - Layer_0	**Graphic**	Complete AGV 06102021
Fit graphic to size	Stretch graphic		

JogChoice

Hardcopy of JogChoice



JOG/TESTING GERTRUDE

12/31/2000 10:59:39 AM

Explanation

There are two jog systems avaliable through the HMI.

The first is direct control of the motors, the user inputs the speed of each motor then jogs them counterclockwise or clockwise. Using this mode the kinematic mode is completely ignored.

The second is similar to the manual mode of the pendant whereby a speed for the AGV is set rather than an individual motor. The system can be steered in both strafing motions and Ackermann motions using this mode.

Create and run repeat tests on the AGV

Name	JogChoice	**Background color**	0, 0, 128
Grid color	0, 0, 0	**Number**	11
Template	Alex Temp A	**Tooltip**	

Rectangle_1

Type	Rectangle	**Name**	Rectangle_1
X position	11	**Y position**	100
Width	1342	**Height**	233
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

HeadingText_Explanation

Type	Text field	**Name**	HeadingText_Explanation
X position	617	**Y position**	105
Width	130	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Explanation		

Text_Explanation_Paragraph1

Type	Text field	**Name**	Text_Explanation_Paragraph1
X position	480	**Y position**	142
Width	404	**Height**	22

Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	There are two jog systems avaliable through the HMI.		

Text_Explanation_Paragraph2

Type	Text field	**Name**	Text_Explanation_Paragraph2
X position	20	**Y position**	195
Width	1324	**Height**	40
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	The first is direct control of the motors, the user inputs the speed of each motor then jogs them counterclockwise or clockwise. Using this mode the kinematic mode is completely ignored.		

Text_Explanation_Paragraph3

Type	Text field	**Name**	Text_Explanation_Paragraph3
X position	29	**Y position**	266
Width	1307	**Height**	40
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	The second is similar to the manual mode of the pendant whereby a speed for the AGV is set rather than an individual motor. The system can be steered in both strafing motions and Ackermann motions using this mode.		

Button_Jog_Individual_Motor

Type	Button	**Name**	Button_Jog_Individual_Motor
X position	42	**Y position**	400
Width	261	**Height**	110
Mode	Text	**Text OFF**	Jog Individual Motor
Text ON	Text		

Dynamizations\Event	
Event name	Click

Function list\ActivateScreen

Screen name	JogSystemUnits	**Object number**	0

Button_Jog_Entire_AGV

Type	Button	**Name**	Button_Jog_Entire_AGV
X position	42	**Y position**	529
Width	261	**Height**	110
Mode	Text	**Text OFF**	Jog Entire AGV
Text ON	Text		

Dynamizations\Event	
Event name	Click

Function list\ActivateScreen

Screen name	JogSystemAGV	**Object number**	0

Rectangle_2

Type	Rectangle	**Name**	Rectangle_2
X position	11	**Y position**	346
Width	347	**Height**	313
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

Rectangle_3

Type	Rectangle	**Name**	Rectangle_3
X position	1006	**Y position**	346
Width	347	**Height**	313
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

AGVPicture

Type	Graphic view	**Name**	AGVPicture
X position	370	**Y position**	346
Width	624	**Height**	313
Layer	0 - Layer_0	**Graphic**	Complete AGV 06102021
Fit graphic to size	Stretch graphic		

HeadingText_Jogging

Type	Text field	**Name**	HeadingText_Jogging
X position	129	**Y position**	353
Width	87	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Jogging		

HeadingText_Testing

Type	Text field	**Name**	HeadingText_Testing
X position	1134	**Y position**	353
Width	82	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Testing		

Button_Testing_Screen

Type	Button	**Name**	Button_Testing_Screen
X position	1045	**Y position**	426
Width	261	**Height**	110
Mode	Text	**Text OFF**	Testing Screen
Text ON	TestRunner		

Dynamizations\Event	
Event name	Click

Function list\ActivateScreen

Screen name	TestRunner	**Object number**	0

Text_Testing_create...

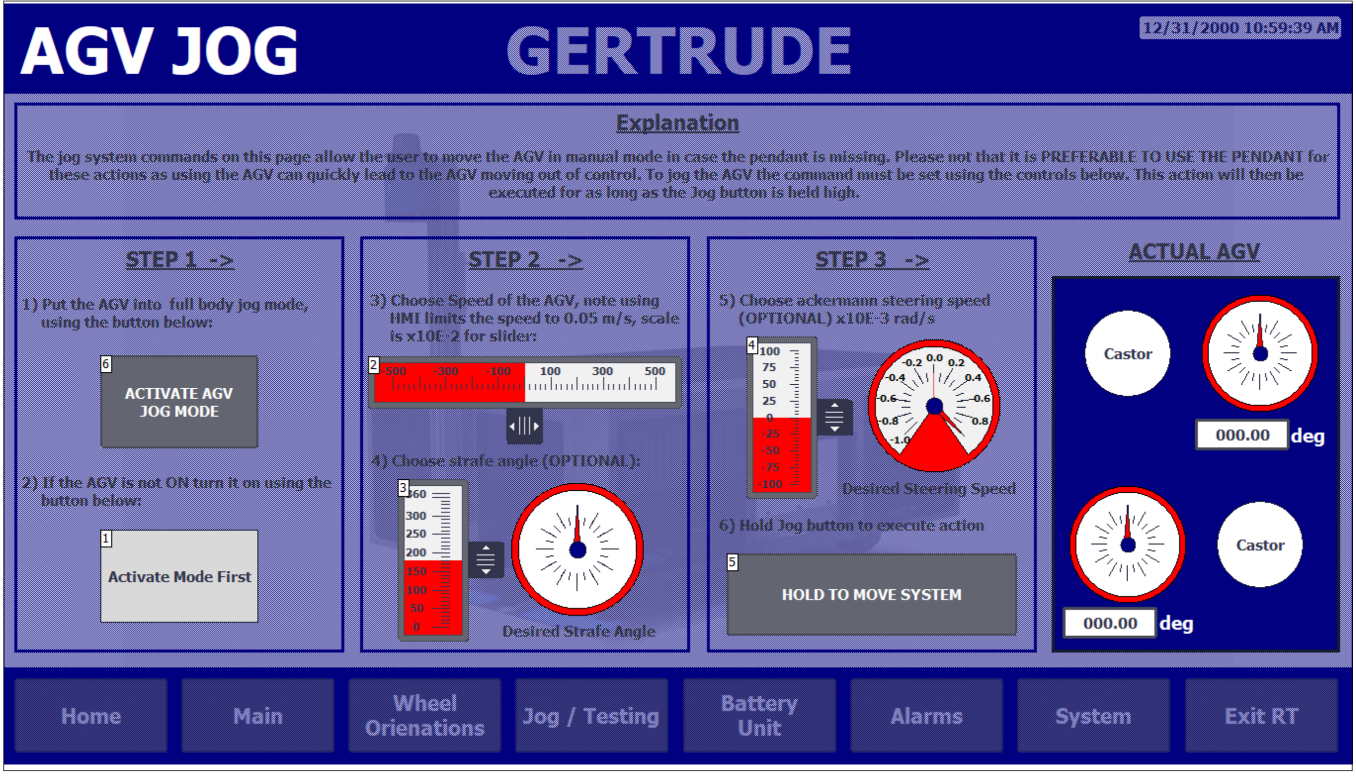
Type	Text field	**Name**	Text_Testing_create...
X position	1029	**Y position**	580
Width	294	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Create and run repeat tests on the AGV		

HeadingText_Jog/Testing

Type	Text field	**Name**	HeadingText_Jog/Testing
X position	14	**Y position**	5
Width	457	**Height**	81
Layer	0 - Layer_0	**Font**	Tahoma, 64px, style=Bold
Text	JOG/TESTING		

JogSystemAGV

Hardcopy of JogSystemAGV



Explanation

The jog system commands on this page allow the user to move the AGV in manual mode in case the pendant is missing. Please not that it is PREFERABLE TO USE THE PENDANT for these actions as using the AGV can quickly lead to the AGV moving out of control. To jog the AGV the command must be set using the controls below. This action will then be executed for as long as the Jog button is held high.

STEP 1 ->

1) Put the AGV into full body jog mode, using the button below:

2) If the AGV is not ON turn it on using the button below:

STEP 2 ->

3) Choose Speed of the AGV, note using HMI limits the speed to 0.05 m/s, scale is x10E-2 for slider:

4) Choose strafe angle (OPTIONAL):

STEP 3 ->

5) Choose ackermann steering speed (OPTIONAL) x10E-3 rad/s

6) Hold Jog button to execute action

Name	JogSystemAGV	**Background color**	0, 0, 128
Grid color	0, 0, 0	**Number**	6
Template	Alex Temp A	**Tooltip**	

Rectangle_1

Type	Rectangle	**Name**	Rectangle_1
X position	11	**Y position**	100
Width	1342	**Height**	118
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

HeadingText_Explanation

Type	Text field	**Name**	HeadingText_Explanation
X position	617	**Y position**	105
Width	130	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Explanation		

Text_Expalanation

Type	Text field	**Name**	Text_Expalanation
X position	21	**Y position**	143
Width	1323	**Height**	58

Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	The jog system commands on this page allow the user to move the AGV in manual mode in case the pendant is missing. Please not that it is PREFERABLE TO USE THE PENDANT for these actions as using the AGV can quickly lead to the AGV moving out of control. To jog the AGV the command must be set using the controls below. This action will then be executed for as long as the Jog button is held high.		

Rectangle_2

Type	Rectangle	**Name**	Rectangle_2
X position	11	**Y position**	235
Width	334	**Height**	420
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

HeadingText_STEP_1...

Type	Text field	**Name**	HeadingText_STEP_1...
X position	121	**Y position**	243
Width	115	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	STEP 1 ->		

Rectangle_3

Type	Rectangle	**Name**	Rectangle_3
X position	361	**Y position**	235
Width	334	**Height**	420
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

HeadingText_STEP_2...

Type	Text field	**Name**	HeadingText_STEP_2...
X position	468	**Y position**	243
Width	121	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	STEP 2 ->		

Rectangle_4

Type	Rectangle	**Name**	Rectangle_4
X position	712	**Y position**	235
Width	334	**Height**	420
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

HeadingText_STEP_3...

Type	Text field	**Name**	HeadingText_STEP_3...

X position	819	**Y position**	243
Width	121	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	STEP 3 ->		

Button_AGV_ON/OFF

Type	Button	**Name**	Button_AGV_ON/OFF
X position	98	**Y position**	531
Width	160	**Height**	94
Mode	Text	**Text OFF**	AGV ON/OFF
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_AGVON/OFF_JogAGV_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_AGVON/OFF_JogAGV_SW

Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_AGVON/OFF_SW_FeedBCK -	**Data type**	Range
Range	0..0	**Foreground color**	255, 255, 255
Background color	99, 101, 113	**Flashing**	No
Range	1..1	**Foreground color**	255, 255, 255
Background color	51, 153, 102	**Flashing**	No

Text_Instruction_2

Type	Text field	**Name**	Text_Instruction_2
X position	15	**Y position**	472
Width	319	**Height**	40
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	2) If the AGV is not ON turn it on using the button below:		

Text_Instruction_1

Type	Text field	**Name**	Text_Instruction_1
X position	15	**Y position**	292
Width	300	**Height**	40
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	1) Put the AGV into full body jog mode, using the button below:		

Text_Instruction_3

Type	Text field	**Name**	Text_Instruction_3
X position	368	**Y position**	288

Width	318	**Height**	58
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	3) Choose Speed of the AGV, note using HMI limits the speed to 0.05 m/s, scale is x10E-2 for slider:		

Slider_AGVSpeed

Type	Slider	**Name**	Slider_AGVSpeed
X position	369	**Y position**	356
Width	318	**Height**	53
Layer	0 - Layer_0	**Minimum value**	-500
Maximum value	500	**Process value**	0
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_JogSpd_SET

Text_Instruction_4

Type	Text field	**Name**	Text_Instruction_4
X position	369	**Y position**	450
Width	278	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	4) Choose strafe angle (OPTIONAL):		

Slider_AGVStrafeAngle

Type	Slider	**Name**	Slider_AGVStrafeAngle
X position	399	**Y position**	480
Width	72	**Height**	164
Layer	0 - Layer_0	**Minimum value**	0
Maximum value	360	**Process value**	180
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_JogStrafeAngle_SET

Gauge_DesiredStrafeAngle

Type	Gauge	**Name**	Gauge_DesiredStrafeAngle
X position	513	**Y position**	483
Width	137	**Height**	137
Minimum value	-360	**Maximum value**	0
Process value	50		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_JogStrafeAngle_FeedBCK

Text_Desired_Strafe...

Type	Text field	**Name**	Text_Desired_Strafe...
X position	502	**Y position**	622
Width	160	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Desired Strafe Angle		

Text_Instruction_5

Type	Text field	**Name**	Text_Instruction_5
X position	721	**Y position**	288

Width	280	**Height**	40
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	5) Choose ackermann steering speed (OPTIONAL) x10E-3 rad/s		

Slider_AckermanSteerAngle

Type	Slider	**Name**	Slider_AckermanSteerAngle
X position	752	**Y position**	336
Width	72	**Height**	164
Layer	0 - Layer_0	**Minimum value**	-100
Maximum value	100	**Process value**	0
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_JogAckrAngle_SET

Text_Desired_Streer...

Type	Text field	**Name**	Text_Desired_Streer...
X position	846	**Y position**	478
Width	181	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Desired Steering Speed		

Button_HOLD_TO_MOVE_SYSTEM

Type	Button	**Name**	Button_HOLD_TO_MOVE_SYSTEM
X position	732	**Y position**	555
Width	294	**Height**	83
Mode	Text	**Text OFF**	HOLD TO MOVE SYSTEM
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_AGVJog_Execute_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_AGVJog_Execute_SW

Text_Instruction_6

Type	Text field	**Name**	Text_Instruction_6
X position	721	**Y position**	515
Width	274	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	6) Hold Jog button to execute action		

Rectangle_5

Type	Rectangle	**Name**	Rectangle_5
X position	1061	**Y position**	275

Width	292	**Height**	380
Layer	0 - Layer_0	**Background color**	0, 0, 128
Border color	24, 28, 49		

HeadingText_ACTUAL_AGV

Type	Text field	**Name**	HeadingText_ACTUAL_AGV
X position	1136	**Y position**	235
Width	138	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	ACTUAL AGV		

Gauge_UnitB_Angle

Type	Gauge	**Name**	Gauge_UnitB_Angle
X position	1078	**Y position**	486
Width	120	**Height**	120
Minimum value	-360	**Maximum value**	0
Process value	50		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_SteeringAngleB_Dial

I/O_UnitB_Angle

Type	I/O field	**Name**	I/O_UnitB_Angle
X position	1072	**Y position**	609
Width	96	**Height**	32
Layer	0 - Layer_0	**Mode**	Output
Font	Tahoma, 15px, style=Bold		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_SteeringAngleB

Text_deg2

Type	Text field	**Name**	Text_deg2
X position	1167	**Y position**	611
Width	40	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	deg		

Gauge_UnitA_Angle

Type	Gauge	**Name**	Gauge_UnitA_Angle
X position	1212	**Y position**	293
Width	120	**Height**	120
Minimum value	-360	**Maximum value**	0
Process value	50		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_SteeringAngleA_Dial

I/O_Field_UnitA_Angle

Type	I/O field	**Name**	I/O_Field_UnitA_Angle
X position	1206	**Y position**	419
Width	96	**Height**	32
Layer	0 - Layer_0	**Mode**	Output

Font	Tahoma, 15px, style=Bold		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_SteeringAngleA

Text_deg1

Type	Text field	**Name**	Text_deg1
X position	1300	**Y position**	422
Width	40	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	deg		

Circle_1

Type	Circle	**Name**	Circle_1
X position	1094	**Y position**	309
Width	88	**Height**	88
Radius	44	**Background color**	255, 255, 255
Border color	24, 28, 49		

Text_Castor1

Type	Text field	**Name**	Text_Castor1
X position	1111	**Y position**	342
Width	54	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Castor		

Circle_2

Type	Circle	**Name**	Circle_2
X position	1228	**Y position**	502
Width	88	**Height**	88
Radius	44	**Background color**	255, 255, 255
Border color	24, 28, 49		

TextCastor2

Type	Text field	**Name**	TextCastor2
X position	1245	**Y position**	535
Width	54	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Castor		

Guage_DesiredAckermanSteerAngle

Type	Gauge	**Name**	Guage_DesiredAckermanSteerAngle
X position	874	**Y position**	338
Width	137	**Height**	137
Minimum value	-1	**Maximum value**	1
Process value	50		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_JogAckrAngle_FeedBCK

Button_ACTIVATE_AGV_JOG_MODE

Type	Button	**Name**	Button_ACTIVATE_AGV_JOG_MODE

X position	98	**Y position**	355
Width	160	**Height**	94
Mode	Text	**Text OFF**	ACTIVATE AGV JOG MODE
Text ON	Text		
Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_AGVJog SW FDBCK -	**Data type**	Range
Range	0..0	**Foreground color**	255, 255, 255
Background color	99, 101, 113	**Flashing**	No
Range	1..1	**Foreground color**	255, 255, 255
Background color	51, 153, 102	**Flashing**	No

Dynamizations\Event	
Event name	Click

Function list\InvertBit

Tag	HMI OP GDB_AGVJog_SW

Group_Hide_AGV_ON/OFF_Rectangle

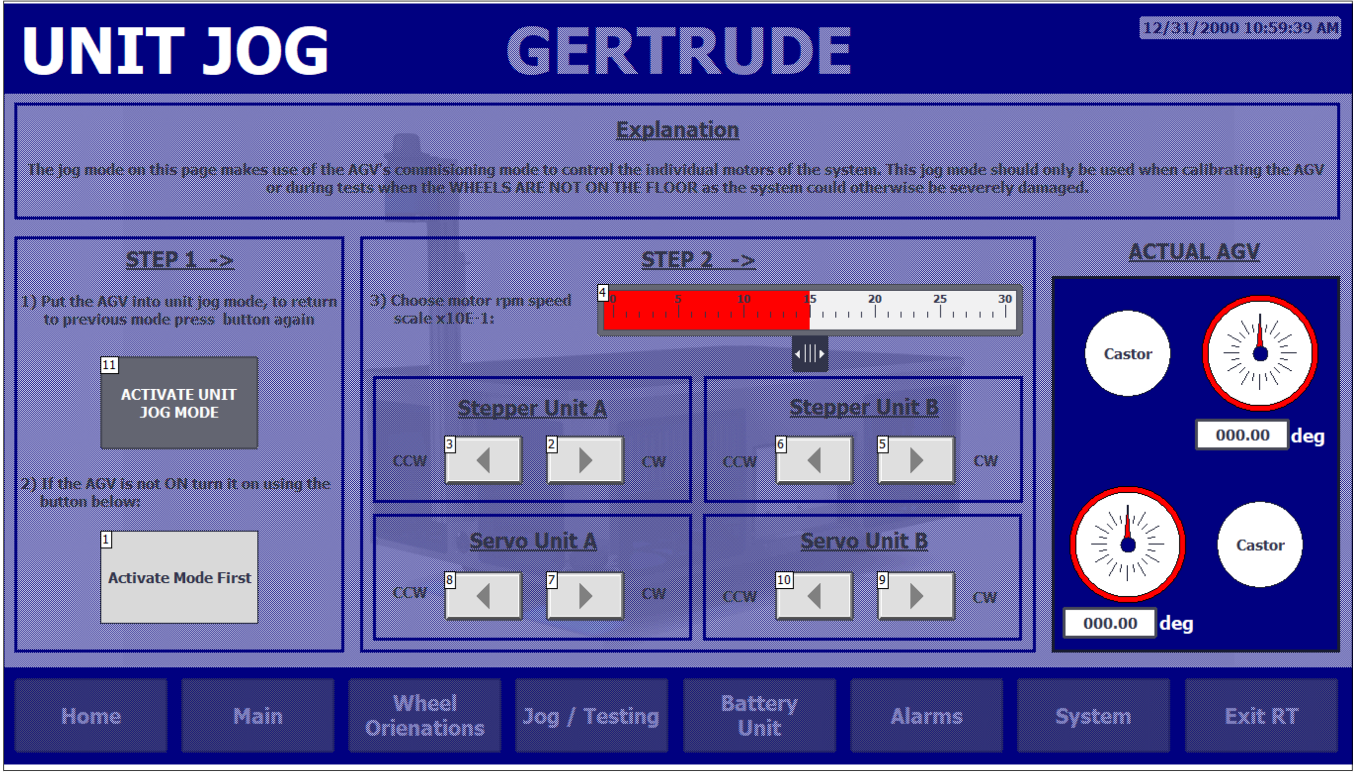
Type	Group	**Name**	Group_Hide_AGV_ON/OFF_Rectangle
X position	98	**Y position**	531
Width	160	**Height**	94
Layer	0 - Layer_0		

HeadingText_AGV_JOG

Type	Text field	**Name**	HeadingText_AGV_JOG
X position	14	**Y position**	5
Width	288	**Height**	81
Layer	0 - Layer_0	**Font**	Tahoma, 64px, style=Bold
Text	AGV JOG		

JogSystemUnits

Hardcopy of JogSystemUnits



The jog mode on this page makes use of the AGV's commisioning mode to control the individual motors of the system. This jog mode should only be used when calibrating the AGV or during tests when the WHEELS ARE NOT ON THE FLOOR as the system could otherwise be severely damaged.

1) Put the AGV into unit jog mode, to return to previous mode press button again

2) If the AGV is not ON turn it on using the button below:

3) Choose motor rpm speed scale x10E-1:

Name	JogSystemUnits	**Background color**	0, 0, 128
Grid color	0, 0, 0	**Number**	10
Template	Alex Temp A	**Tooltip**	

Rectangle_1

Type	Rectangle	**Name**	Rectangle_1
X position	11	**Y position**	100
Width	1342	**Height**	118
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

HeadingText_Explanation

Type	Text field	**Name**	HeadingText_Explanation
X position	617	**Y position**	112
Width	130	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Explanation		

Text_Explanation

Type	Text field	**Name**	Text_Explanation
X position	21	**Y position**	156
Width	1322	**Height**	40

Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	The jog mode on this page makes use of the AGV's commisioning mode to control the individual motors of the system. This jog mode should only be used when calibrating the AGV or during tests when the WHEELS ARE NOT ON THE FLOOR as the system could otherwise be severely damaged.		

Rectangle_2

Type	Rectangle	**Name**	Rectangle_2
X position	11	**Y position**	235
Width	334	**Height**	420
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

HeadingText_STEP_1...

Type	Text field	**Name**	HeadingText_STEP_1...
X position	121	**Y position**	243
Width	115	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	STEP 1 ->		

Button_AGV_ON/OFF

Type	Button	**Name**	Button_AGV_ON/OFF
X position	98	**Y position**	532
Width	160	**Height**	94
Mode	Text	**Text OFF**	AGV ON/OFF
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_AGVON/OFF_JogUnit_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_AGVON/OFF_JogUnit_SW

Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_AGVON/OFF_SW_FeedBCK -	**Data type**	Range
Range	0..0	**Foreground color**	255, 255, 255
Background color	99, 101, 113	**Flashing**	No
Range	1..1	**Foreground color**	255, 255, 255
Background color	51, 153, 102	**Flashing**	No

Text_Instruction_2

Type	Text field	**Name**	Text_Instruction_2
X position	14	**Y position**	473
Width	319	**Height**	40
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	2) If the AGV is not ON turn it on using the button below:		

Text_Instruction_1

Type	Text field	**Name**	Text_Instruction_1
X position	14	**Y position**	289
Width	326	**Height**	40
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	1) Put the AGV into unit jog mode, to return to previous mode press button again		

Rectangle_3

Type	Rectangle	**Name**	Rectangle_3
X position	361	**Y position**	235
Width	684	**Height**	420
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

HeadingText_STEP_2...

Type	Text field	**Name**	HeadingText_STEP_2...
X position	643	**Y position**	243
Width	121	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	STEP 2 ->		

Text_Instruction_3

Type	Text field	**Name**	Text_Instruction_3
X position	368	**Y position**	288
Width	209	**Height**	40
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	3) Choose motor rpm speed scale x10E-1:		

Rectangle_5

Type	Rectangle	**Name**	Rectangle_5
X position	1061	**Y position**	275
Width	292	**Height**	380
Layer	0 - Layer_0	**Background color**	0, 0, 128
Border color	24, 28, 49		

HeadingText_ACTUAL_AGV

Type	Text field	**Name**	HeadingText_ACTUAL_AGV
X position	1136	**Y position**	235

Width	138	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	ACTUAL AGV		

Gauge_UnitB_Angle

Type	Gauge	**Name**	Gauge_UnitB_Angle
X position	1078	**Y position**	486
Width	120	**Height**	120
Minimum value	-360	**Maximum value**	0
Process value	50		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_SteeringAngleB_Dial

I/O_Field_UnitB_Angle

Type	I/O field	**Name**	I/O_Field_UnitB_Angle
X position	1072	**Y position**	609
Width	96	**Height**	32
Layer	0 - Layer_0	**Mode**	Output
Font	Tahoma, 15px, style=Bold		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_SteeringAngleB

Text_deg2

Type	Text field	**Name**	Text_deg2
X position	1167	**Y position**	611
Width	40	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	deg		

Gauge_3

Type	Gauge	**Name**	Gauge_3
X position	1212	**Y position**	293
Width	120	**Height**	120
Minimum value	-360	**Maximum value**	0
Process value	50		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_SteeringAngleA_Dial

I/O_Field_UnitA_Angle

Type	I/O field	**Name**	I/O_Field_UnitA_Angle
X position	1206	**Y position**	419
Width	96	**Height**	32
Layer	0 - Layer_0	**Mode**	Output
Font	Tahoma, 15px, style=Bold		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_SteeringAngleA

Text_deg1

Type	Text field	**Name**	Text_deg1
X position	1300	**Y position**	422

Width	40	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	deg		

Circle_1

Type	Circle	**Name**	Circle_1
X position	1094	**Y position**	309
Width	88	**Height**	88
Radius	44	**Background color**	255, 255, 255
Border color	24, 28, 49		

TextCastor1

Type	Text field	**Name**	TextCastor1
X position	1111	**Y position**	342
Width	54	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Castor		

Circle_2

Type	Circle	**Name**	Circle_2
X position	1228	**Y position**	502
Width	88	**Height**	88
Radius	44	**Background color**	255, 255, 255
Border color	24, 28, 49		

Text_Castor2

Type	Text field	**Name**	Text_Castor2
X position	1245	**Y position**	535
Width	54	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Castor		

Rectangle_4

Type	Rectangle	**Name**	Rectangle_4
X position	709	**Y position**	376
Width	323	**Height**	128
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

Rectangle_6

Type	Rectangle	**Name**	Rectangle_6
X position	708	**Y position**	515
Width	323	**Height**	128
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

Rectangle_7

Type	Rectangle	**Name**	Rectangle_7
X position	374	**Y position**	376
Width	323	**Height**	128

Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

Rectangle_8

Type	Rectangle	**Name**	Rectangle_8
X position	374	**Y position**	515
Width	323	**Height**	128
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

HeadingText_Stepper_Unit_A

Type	Text field	**Name**	HeadingText_Stepper_Unit_A
X position	457	**Y position**	393
Width	156	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Stepper Unit A		

HeadingText_Stepper_Unit_B

Type	Text field	**Name**	HeadingText_Stepper_Unit_B
X position	793	**Y position**	392
Width	156	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Stepper Unit B		

HeadingText_Servo_Unit_A

Type	Text field	**Name**	HeadingText_Servo_Unit_A
X position	469	**Y position**	527
Width	134	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Servo Unit A		

HeadingText_Servo_Unit_B

Type	Text field	**Name**	HeadingText_Servo_Unit_B
X position	804	**Y position**	527
Width	134	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Servo Unit B		

Button_StepperA_Right

Type	Button	**Name**	Button_StepperA_Right
X position	549	**Y position**	436
Width	80	**Height**	50
Mode	Check back with graphic	**Text OFF**	Text
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_UnitA_Cali_CW_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_UnitA_Cali_CW_SW

Button_StepperA_Left

Type	Button	**Name**	Button_StepperA_Left
X position	446	**Y position**	436
Width	80	**Height**	50
Mode	Check back with graphic	**Text OFF**	Text
Text ON	Text		

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_UnitA_Cali_CCW_SW

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_UnitA_Cali_CCW_SW

Text_StepperA _CCW

Type	Text field	**Name**	Text_StepperA _CCW
X position	391	**Y position**	450
Width	40	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	CCW		

Text_StepperA _CW

Type	Text field	**Name**	Text_StepperA _CW
X position	643	**Y position**	451
Width	30	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	CW		

Slider_MotorRPM

Type	Slider	**Name**	Slider_MotorRPM
X position	601	**Y position**	283
Width	431	**Height**	53
Layer	0 - Layer_0	**Minimum value**	0
Maximum value	30	**Process value**	15
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_CalibrateSteppr Spd

Button_StepperB_Right

Type	Button	**Name**	Button_StepperB_Right

X position	884	**Y position**	436
Width	80	**Height**	50
Mode	Check back with graphic	**Text OFF**	Text
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_UnitB_Cali_CW_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_UnitB_Cali_CW_SW

Button_StepperB_Left

Type	Button	**Name**	Button_StepperB_Left
X position	781	**Y position**	436
Width	80	**Height**	50
Mode	Check back with graphic	**Text OFF**	Text
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_UnitB_Cali_CCW_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_UnitB_Cali_CCW_SW

Text_StepperB _CW

Type	Text field	**Name**	Text_StepperB _CW
X position	979	**Y position**	450
Width	30	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	CW		

Text_StepperB _CCW

Type	Text field	**Name**	Text_StepperB _CCW
X position	725	**Y position**	451
Width	40	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	CCW		

Button_ServoA_Right

Type	Button	**Name**	Button_ServoA_Right
X position	549	**Y position**	573
Width	80	**Height**	50
Mode	Check back with graphic	**Text OFF**	Text
Text ON	Text		

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_UnitA_Servo_CW_SW

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_UnitA_Servo_CW_SW

Button_ServoA_Left

Type	Button	**Name**	Button_ServoA_Left
X position	446	**Y position**	573
Width	80	**Height**	50
Mode	Check back with graphic	**Text OFF**	Text
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_UnitA_Servo_CCW_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_UnitA_Servo_CCW_SW

Button_ServoB_Right

Type	Button	**Name**	Button_ServoB_Right
X position	884	**Y position**	573
Width	80	**Height**	50
Mode	Check back with graphic	**Text OFF**	Text
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_UnitB_Servo_CW_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_UnitB_Servo_CW_SW

Button_ServoB_Left

Type	Button	**Name**	Button_ServoB_Left
X position	781	**Y position**	573
Width	80	**Height**	50
Mode	Check back with graphic	**Text OFF**	Text
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_UnitB_Servo_CCW_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_UnitB_Servo_CCW_SW

Text_ServoA _CCW

Type	Text field	**Name**	Text_ServoA _CCW
X position	391	**Y position**	583
Width	40	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	CCW		

Text_ServoA _CW

Type	Text field	**Name**	Text_ServoA _CW
X position	643	**Y position**	584
Width	30	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	CW		

Text_ServoB _CCW

Type	Text field	**Name**	Text_ServoB _CCW
X position	725	**Y position**	587
Width	40	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	CCW		

Text_ServoB _CW

Type	Text field	**Name**	Text_ServoB _CW
X position	978	**Y position**	588
Width	30	**Height**	22

Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	CW		

Button_ACTIVATE_UNIT_JOG_MODE

Type	Button	**Name**	Button_ACTIVATE_UNIT_JOG_MODE
X position	98	**Y position**	356
Width	160	**Height**	94
Mode	Text	**Text OFF**	ACTIVATE UNIT JOG MODE
Text ON	Text		
Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_JogUnit LED -	**Data type**	Range
Range	0..0	**Foreground color**	255, 255, 255
Background color	99, 101, 113	**Flashing**	No
Range	1..1	**Foreground color**	255, 255, 255
Background color	51, 153, 102	**Flashing**	No

Dynamizations\Event	
Event name	Click

Function list\InvertBit

Tag	HMI OP GDB_JogUnit SW

Group_Hide_AGV_ON/OFF_Rectangle

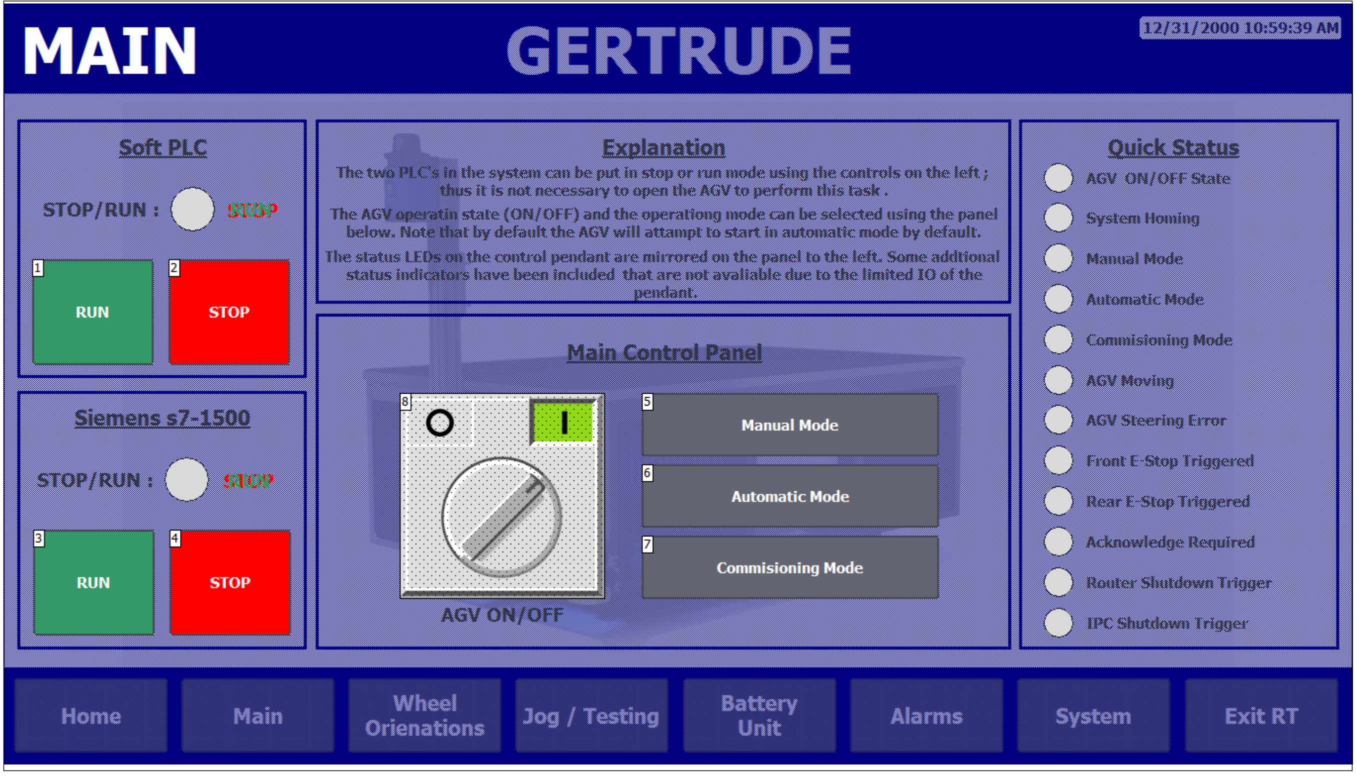
Type	Group	**Name**	Group_Hide_AGV_ON/OFF_Rectangle
X position	98	**Y position**	532
Width	160	**Height**	94
Layer	0 - Layer_0		

HeadingText_UNIT_JOG

Type	Text field	**Name**	HeadingText_UNIT_JOG
X position	14	**Y position**	5
Width	319	**Height**	81
Layer	0 - Layer_0	**Font**	Tahoma, 64px, style=Bold
Text	UNIT JOG		

Main

Hardcopy of Main



The two PLC's in the system can be put in stop or run mode using the controls on the left ; thus it is not necessary to open the AGV to perform this task .

The AGV operatin state (ON/OFF) and the operationg mode can be selected using the panel below. Note that by default the AGV will attampt to start in automatic mode by default.

The status LEDs on the control pendant are mirrored on the panel to the left. Some addtional status indicators have been included that are not avaliable due to the limited IO of the pendant.

Name	Main	**Background color**	0, 0, 128
Grid color	0, 0, 0	**Number**	2
Template	Alex Temp A	**Tooltip**	

Rectangle_1

Type	Rectangle	**Name**	Rectangle_1
X position	14	**Y position**	117
Width	293	**Height**	261
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

Rectangle_2

Type	Rectangle	**Name**	Rectangle_2
X position	14	**Y position**	391
Width	293	**Height**	261
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

Rectangle_3

Type	Rectangle	**Name**	Rectangle_3
X position	1028	**Y position**	117
Width	324	**Height**	535

Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

HeadingText_Soft_PLC

Type	Text field	**Name**	HeadingText_Soft_PLC
X position	114	**Y position**	130
Width	94	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Soft PLC		

HeadingText_Siemens_s7-1500

Type	Text field	**Name**	HeadingText_Siemens_s7-1500
X position	69	**Y position**	403
Width	183	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Siemens s7-1500		

Text_SoftPLC_STOP/RUN:

Type	Text field	**Name**	Text_SoftPLC_STOP/RUN:
X position	37	**Y position**	194
Width	123	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	STOP/RUN :		

Button_SoftPLC_RUN

Type	Button	**Name**	Button_SoftPLC_RUN
X position	29	**Y position**	258
Width	123	**Height**	107
Mode	Text	**Text OFF**	RUN
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetPLCMode

Connection	HMI_SoftPLC	**Mode**	RUN

Dynamizations\Event	
Event name	Release

Function list\GetPLCMode

Connection	HMI_SoftPLC	**Mode**	SoftPLC_Status

Button_SoftPLC_STOP

Type	Button	**Name**	Button_SoftPLC_STOP
X position	167	**Y position**	258
Width	123	**Height**	107
Mode	Text	**Text OFF**	STOP
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetPLCMode

Connection	HMI_SoftPLC	**Mode**	STOP

Dynamizations\Event	
Event name	Release

Function list\GetPLCMode

Connection	HMI_SoftPLC	**Mode**	SoftPLC_Status

Button_PLC_RUN

Type	Button	**Name**	Button_PLC_RUN
X position	30	**Y position**	531
Width	123	**Height**	107
Mode	Text	**Text OFF**	RUN
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetPLCMode

Connection	HMI_s7-1500	**Mode**	RUN

Dynamizations\Event	
Event name	Release

Function list\GetPLCMode

Connection	HMI_s7-1500	**Mode**	s7-1500_Status

Button_PLC_Stop

Type	Button	**Name**	Button_PLC_Stop
X position	168	**Y position**	531
Width	123	**Height**	107
Mode	Text	**Text OFF**	STOP
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetPLCMode

Connection	HMI_s7-1500	**Mode**	STOP

Dynamizations\Event	
Event name	Release

Function list\GetPLCMode

Connection	HMI_s7-1500	**Mode**	s7-1500_Status

Indicator_SoftPLC_ON/OFF

Type	Circle	**Name**	Indicator_SoftPLC_ON/OFF
X position	169	**Y position**	185
Width	45	**Height**	45
Radius	22	**Background color**	217, 217, 217
Border color	24, 28, 49		
Dynamizations\Appearance			
Tag - Cycle	SoftPLC_Status -	**Data type**	Range
Range	4..4	**Foreground color**	24, 28, 49
Background color	255, 0, 0	**Flashing**	No
Range	8..8	**Foreground color**	24, 28, 49
Background color	51, 153, 102	**Flashing**	No

Text_SoftPLC_RUN

Type	Text field	**Name**	Text_SoftPLC_RUN
X position	226	**Y position**	194
Width	48	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	RUN		
Dynamizations\Visibility			
Tag - Cycle	SoftPLC_Status -	**Data type**	Range
Start range	8	**End range**	8
Visibility	Visible		

Text_SoftPLC_STOP

Type	Text field	**Name**	Text_SoftPLC_STOP
X position	224	**Y position**	195
Width	56	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	STOP		
Dynamizations\Visibility			
Tag - Cycle	SoftPLC_Status -	**Data type**	Range
Start range	4	**End range**	4
Visibility	Visible		

Text_PLC_STOP/RUN:

Type	Text field	**Name**	Text_PLC_STOP/RUN:
X position	31	**Y position**	467
Width	123	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	STOP/RUN :		

Indicator_PLC_ON/OFF

Type	Circle	**Name**	Indicator_PLC_ON/OFF
X position	163	**Y position**	458
Width	45	**Height**	45
Radius	22	**Background color**	217, 217, 217
Border color	24, 28, 49		
Dynamizations\Appearance			
Tag - Cycle	s7-1500_Status -	**Data type**	Range

Range	4..4	**Foreground color**	24, 28, 49
Background color	255, 0, 0	**Flashing**	No
Range	8..8	**Foreground color**	24, 28, 49
Background color	51, 153, 102	**Flashing**	No

Text_PLC_STOP

Type	Text field	**Name**	Text_PLC_STOP
X position	220	**Y position**	468
Width	56	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	STOP		
Dynamizations\Visibility			
Tag - Cycle	s7-1500_Status -	**Data type**	Range
Start range	4	**End range**	4
Visibility	Visible		

Text_PLC_RUN

Type	Text field	**Name**	Text_PLC_RUN
X position	224	**Y position**	468
Width	48	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	RUN		
Dynamizations\Visibility			
Tag - Cycle	s7-1500_Status -	**Data type**	Range
Start range	8	**End range**	8
Visibility	Visible		

Indicator_AGV_ON/OFF_State

Type	Circle	**Name**	Indicator_AGV_ON/OFF_State
X position	1053	**Y position**	161
Width	30	**Height**	30
Radius	15	**Background color**	217, 217, 217
Border color	24, 28, 49		
Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_AGVStateLED -	**Data type**	Range
Range	0..0	**Foreground color**	24, 28, 49
Background color	217, 217, 217	**Flashing**	No
Range	1..1	**Foreground color**	24, 28, 49
Background color	255, 255, 0	**Flashing**	No

HeadingText_Quick_Status

Type	Text field	**Name**	HeadingText_Quick_Status
X position	1115	**Y position**	130
Width	138	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Quick Status		

Text_Indicator_AGV_ON/OFF_State

Type	Text field	**Name**	Text_Indicator_AGV_ON/OFF_State
X position	1093	**Y position**	165
Width	151	**Height**	22

Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	AGV ON/OFF State		

Indicator_System_Homing

Type	Circle	**Name**	Indicator_System_Homing
X position	1053	**Y position**	201
Width	30	**Height**	30
Radius	15	**Background color**	217, 217, 217
Border color	24, 28, 49		
Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_SystemHomingLED -	**Data type**	Range
Range	0..0	**Foreground color**	24, 28, 49
Background color	217, 217, 217	**Flashing**	No
Range	1..1	**Foreground color**	24, 28, 49
Background color	255, 255, 0	**Flashing**	No

Text_Indicator_System_Homing

Type	Text field	**Name**	Text_Indicator_System_Homing
X position	1093	**Y position**	205
Width	120	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	System Homing		

Indicator_Manual_Mode

Type	Circle	**Name**	Indicator_Manual_Mode
X position	1053	**Y position**	242
Width	30	**Height**	30
Radius	15	**Background color**	217, 217, 217
Border color	24, 28, 49		
Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_ManualModeLED -	**Data type**	Range
Range	0..0	**Foreground color**	24, 28, 49
Background color	217, 217, 217	**Flashing**	No
Range	1..1	**Foreground color**	24, 28, 49
Background color	255, 255, 0	**Flashing**	No

Text_Indicator_Manual_Mode

Type	Text field	**Name**	Text_Indicator_Manual_Mode
X position	1093	**Y position**	246
Width	103	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Manual Mode		

Indicator_Automatic_Mode

Type	Circle	**Name**	Indicator_Automatic_Mode
X position	1053	**Y position**	283
Width	30	**Height**	30
Radius	15	**Background color**	217, 217, 217
Border color	24, 28, 49		
Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_AutomaticModeLED -	**Data type**	Range

Range	0..0	**Foreground color**	24, 28, 49
Background color	217, 217, 217	**Flashing**	No
Range	1..1	**Foreground color**	24, 28, 49
Background color	255, 255, 0	**Flashing**	No

Text_Indicator_Automatic_Mode

Type	Text field	**Name**	Text_Indicator_Automatic_Mode
X position	1093	**Y position**	287
Width	124	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Automatic Mode		

Indicator_Commisioning_Mode

Type	Circle	**Name**	Indicator_Commisioning_Mode
X position	1053	**Y position**	324
Width	30	**Height**	30
Radius	15	**Background color**	217, 217, 217
Border color	24, 28, 49		
Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_CommisionModeLED -	**Data type**	Range
Range	0..0	**Foreground color**	24, 28, 49
Background color	217, 217, 217	**Flashing**	No
Range	1..1	**Foreground color**	24, 28, 49
Background color	255, 255, 0	**Flashing**	No

Text_Indicator_Commisioning_Mode

Type	Text field	**Name**	Text_Indicator_Commisioning_Mode
X position	1093	**Y position**	328
Width	153	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Commisioning Mode		

Indicator_AGV_Moving

Type	Circle	**Name**	Indicator_AGV_Moving
X position	1053	**Y position**	365
Width	30	**Height**	30
Radius	15	**Background color**	217, 217, 217
Border color	24, 28, 49		
Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_AGVMovingLED -	**Data type**	Range
Range	0..0	**Foreground color**	24, 28, 49
Background color	217, 217, 217	**Flashing**	No
Range	1..1	**Foreground color**	24, 28, 49
Background color	255, 255, 0	**Flashing**	No

Text_Indicator_AGV_Moving

Type	Text field	**Name**	Text_Indicator_AGV_Moving
X position	1093	**Y position**	369
Width	94	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	AGV Moving		

Indicator_AGV_Steering_Error

Type	Circle	**Name**	Indicator_AGV_Steering_Error
X position	1053	**Y position**	405
Width	30	**Height**	30
Radius	15	**Background color**	217, 217, 217
Border color	24, 28, 49		
Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_AGVWheelErrorLED -	**Data type**	Range
Range	0..0	**Foreground color**	24, 28, 49
Background color	217, 217, 217	**Flashing**	No
Range	1..1	**Foreground color**	24, 28, 49
Background color	255, 255, 0	**Flashing**	No

Text_Indicator_AGV_Steering_Error

Type	Text field	**Name**	Text_Indicator_AGV_Steering_Error
X position	1093	**Y position**	409
Width	147	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	AGV Steering Error		

Indicator_Acknowledge_Required

Type	Circle	**Name**	Indicator_Acknowledge_Required
X position	1053	**Y position**	528
Width	30	**Height**	30
Radius	15	**Background color**	217, 217, 217
Border color	24, 28, 49		
Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_AcknowledgeLED -	**Data type**	Range
Range	0..0	**Foreground color**	24, 28, 49
Background color	217, 217, 217	**Flashing**	No
Range	1..1	**Foreground color**	24, 28, 49
Background color	255, 255, 0	**Flashing**	No

Text_Indicator_Acknowledge_Required

Type	Text field	**Name**	Text_Indicator_Acknowledge_Required
X position	1093	**Y position**	532
Width	176	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Acknowledge Required		

Indicator_Front_E-Stop_Triggered

Type	Circle	**Name**	Indicator_Front_E-Stop_Triggered
X position	1053	**Y position**	446
Width	30	**Height**	30
Radius	15	**Background color**	217, 217, 217
Border color	24, 28, 49		
Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_FrontEStopLED -	**Data type**	Range
Range	0..0	**Foreground color**	24, 28, 49
Background color	217, 217, 217	**Flashing**	No

Range	1..1	**Foreground color**	24, 28, 49
Background color	255, 255, 0	**Flashing**	No

Text_Indicator_Front_E-Stop_Triggered

Type	Text field	**Name**	Text_Indicator_Front_E-Stop_Triggered
X position	1093	**Y position**	450
Width	175	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Front E-Stop Triggered		

Indicator_Rear_E-Stop_Triggered

Type	Circle	**Name**	Indicator_Rear_E-Stop_Triggered
X position	1053	**Y position**	487
Width	30	**Height**	30
Radius	15	**Background color**	217, 217, 217
Border color	24, 28, 49		
Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_RearEStopLED -	**Data type**	Range
Range	0..0	**Foreground color**	24, 28, 49
Background color	217, 217, 217	**Flashing**	No
Range	1..1	**Foreground color**	24, 28, 49
Background color	255, 255, 0	**Flashing**	No

Text_Indicator_Rear_E-Stop_Triggered

Type	Text field	**Name**	Text_Indicator_Rear_E-Stop_Triggered
X position	1093	**Y position**	491
Width	171	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Rear E-Stop Triggered		

Softkey_F6

Type	Function key	**Key code**	212
Global assignment	Enabled	**Graphic**	
Authorization		**LED tag**	
Bit in the LED tag	0		

Softkey_F7

Type	Function key	**Key code**	203
Global assignment	Enabled	**Graphic**	
Authorization		**LED tag**	
Bit in the LED tag	0		

Softkey_F5

Type	Function key	**Key code**	202
Global assignment	Enabled	**Graphic**	
Authorization		**LED tag**	
Bit in the LED tag	0		

Softkey_F8

Type	Function key	**Key code**	213

Global assignment	Enabled	**Graphic**	
Authorization		**LED tag**	
Bit in the LED tag	0		

Softkey_F12

Type	Function key	**Key code**	215
Global assignment	Enabled	**Graphic**	
Authorization		**LED tag**	
Bit in the LED tag	0		

Softkey_F3

Type	Function key	**Key code**	201
Global assignment	Enabled	**Graphic**	
Authorization		**LED tag**	
Bit in the LED tag	0		

Rectangle_4

Type	Rectangle	**Name**	Rectangle_4
X position	316	**Y position**	313
Width	704	**Height**	339
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

HeadingText_Main_Control_Panel

Type	Text field	**Name**	HeadingText_Main_Control_Panel
X position	567	**Y position**	337
Width	203	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Main Control Panel		

Text_MainControlPanel_AGV_ON/OFF

Type	Text field	**Name**	Text_MainControlPanel_AGV_ON/OFF
X position	440	**Y position**	603
Width	129	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	AGV ON/OFF		

Button_Manual_Mode

Type	Button	**Name**	Button_Manual_Mode
X position	646	**Y position**	393
Width	301	**Height**	64
Mode	Text	**Text OFF**	Manual Mode
Text ON	Text		
Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_ManualModeLED -	**Data type**	Range
Range	0..0	**Foreground color**	255, 255, 255
Background color	99, 101, 113	**Flashing**	No
Range	1..1	**Foreground color**	255, 255, 255
Background color	51, 153, 102	**Flashing**	No

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_ManualMode_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_ManualMode_SW

Button_Automatic_Mode

Type	Button	Name	Button_Automatic_Mode
X position	646	Y position	465
Width	301	Height	64
Mode	Text	Text OFF	Automatic Mode
Text ON	Text		
Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_AutomaticModeLED -	Data type	Range
Range	0..0	Foreground color	255, 255, 255
Background color	99, 101, 113	Flashing	No
Range	1..1	Foreground color	255, 255, 255
Background color	51, 153, 102	Flashing	No

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_AutoMode_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_AutoMode_SW

Button_Commisioning_Mode

Type	Button	Name	Button_Commisioning_Mode
X position	646	Y position	537
Width	301	Height	64
Mode	Text	Text OFF	Commisioning Mode
Text ON	Text		
Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_CommisionModeLED -	Data type	Range
Range	0..0	Foreground color	255, 255, 255
Background color	99, 101, 113	Flashing	No
Range	1..1	Foreground color	255, 255, 255
Background color	51, 153, 102	Flashing	No

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_CommisinMode_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_CommisinMode_SW

Graphic_Switch=OFF

Type	Graphic view	Name	Graphic_Switch=OFF
X position	401	Y position	393
Width	208	Height	208
Layer	0 - Layer_0	Graphic	Rotary_RNGN_Off_256c
Fit graphic to size	Stretch graphic		
Dynamizations\Visibility			
Tag - Cycle	HMI OP GDB_AGVON/OFF_SW_FeedBCK -	Data type	Bit
Specifies the bit to monitor.	0	Visibility	Invisible

Graphic_Switch=ON

Type	Graphic view	Name	Graphic_Switch=ON
X position	401	Y position	393
Width	208	Height	208
Layer	0 - Layer_0	Graphic	Rotary_RNGN_On_256c
Fit graphic to size	Stretch graphic		
Dynamizations\Visibility			
Tag - Cycle	HMI OP GDB_AGVON/OFF_SW_FeedBCK -	Data type	Bit
Specifies the bit to monitor.	0	Visibility	Visible

Button_GraphicOverlay_AGV_ON/OFF

Type	Button	Name	Button_GraphicOverlay_AGV_ON/OFF
X position	401	Y position	393
Width	208	Height	208
Mode	Invisible	Text OFF	Text
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_AGVON/OFF_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_AGVON/OFF_SW

Rectangle_5

Type	Rectangle	**Name**	Rectangle_5
X position	316	**Y position**	117
Width	704	**Height**	186
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

HeadingText_Explanation

Type	Text field	**Name**	HeadingText_Explanation
X position	603	**Y position**	130
Width	130	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Explanation		

Text_Explanation_Paragraph1

Type	Text field	**Name**	Text_Explanation_Paragraph1
X position	334	**Y position**	159
Width	669	**Height**	40
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	The two PLC's in the system can be put in stop or run mode using the controls on the left ; thus it is not necessary to open the AGV to perform this task .		

Text_Explanation_Paragraph2

Type	Text field	**Name**	Text_Explanation_Paragraph2
X position	328	**Y position**	201
Width	681	**Height**	40
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	The AGV operatin state (ON/OFF) and the operationg mode can be selected using the panel below. Note that by default the AGV will attampt to start in automatic mode by default.		

Text_Explanation_Paragraph3

Type	Text field	**Name**	Text_Explanation_Paragraph3
X position	322	**Y position**	244
Width	693	**Height**	58
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	The status LEDs on the control pendant are mirrored on the panel to the left. Some addtional status indicators have been included that are not avaliable due to the limited IO of the pendant.		

Indicator_IPC_Shtdown_Trigger

Type	Circle	**Name**	Indicator_IPC_Shtdown_Trigger
X position	1053	**Y position**	610
Width	30	**Height**	30
Radius	15	**Background color**	217, 217, 217
Border color	24, 28, 49		
Dynamizations\Appearance			
Tag - Cycle	ShutdownIPC_Trigger -	**Data type**	Range
Range	0..0	**Foreground color**	24, 28, 49
Background color	217, 217, 217	**Flashing**	No
Range	1..1	**Foreground color**	24, 28, 49
Background color	255, 255, 0	**Flashing**	No

Indicator_Router_Shtdown_Trigger

Type	Circle	**Name**	Indicator_Router_Shtdown_Trigger
X position	1053	**Y position**	569
Width	30	**Height**	30
Radius	15	**Background color**	217, 217, 217
Border color	24, 28, 49		
Dynamizations\Appearance			
Tag - Cycle	ShutdownRouter_Trigger -	**Data type**	Range
Range	0..0	**Foreground color**	24, 28, 49
Background color	217, 217, 217	**Flashing**	No
Range	1..1	**Foreground color**	24, 28, 49
Background color	255, 255, 0	**Flashing**	No

Text_Indicator_Router_Shtdown_Trigger

Type	Text field	**Name**	Text_Indicator_Router_Shtdown_Trigger
X position	1093	**Y position**	573
Width	193	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Router Shutdown Trigger		

Text_Indicator_IPC_Shtdown_Trigger

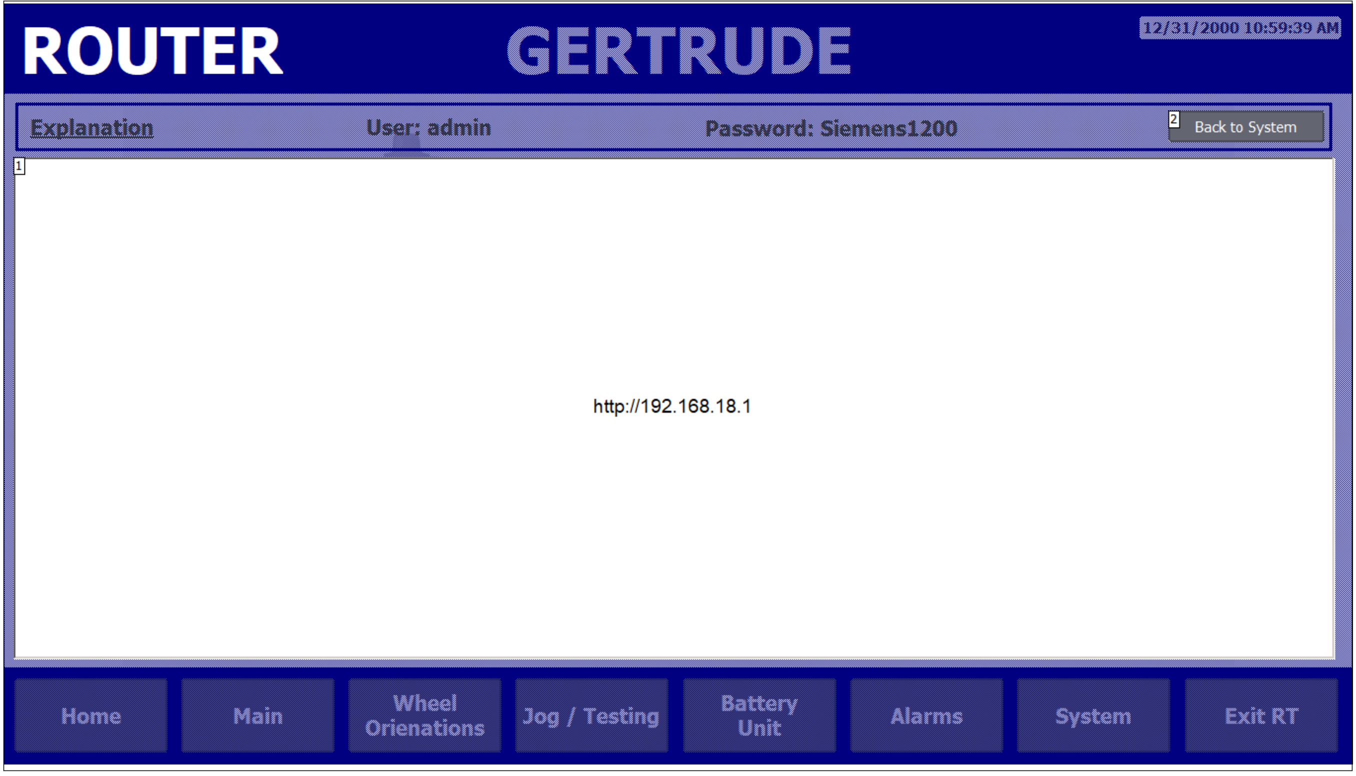
Type	Text field	**Name**	Text_Indicator_IPC_Shtdown_Trigger
X position	1093	**Y position**	614
Width	169	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	IPC Shutdown Trigger		

HeadingText_MAIN

Type	Text field	**Name**	HeadingText_MAIN
X position	14	**Y position**	5
Width	186	**Height**	81
Layer	0 - Layer_0	**Font**	Tahoma, 64px, style=Bold
Text	MAIN		

RouterWebpage

Hardcopy of RouterWebpage



ROUTER GERTRUDE

12/31/2000 10:59:39 AM

Explanation User: admin Password: Siemens1200

http://192.168.18.1

Name	RouterWebpage	**Background color**	0, 0, 128
Grid color	0, 0, 0	**Number**	9
Template	Alex Temp A	**Tooltip**	

HTML_RouterWebpage

Type	HTML browser	**Name**	HTML_RouterWebpage
X position	10	**Y position**	155
Width	1337	**Height**	506
Layer	0 - Layer_0	**Address**	http://192.168.18.1

Rectangle_1

Type	Rectangle	**Name**	Rectangle_1
X position	12	**Y position**	100
Width	1333	**Height**	49
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

HeadingText_Explanation

Type	Text field	**Name**	HeadingText_Explanation
X position	24	**Y position**	110
Width	130	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline

Text	Explanation

Text_User:admin

Type	Text field	**Name**	Text_User:admin
X position	364	**Y position**	110
Width	132	**Height**	29
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold
Text	User: admin		

Text_Password:...

Type	Text field	**Name**	Text_Password:...
X position	707	**Y position**	111
Width	261	**Height**	29
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold
Text	Password: Siemens1200		

Button_Back_to_System

Type	Button	**Name**	Button_Back_to_System
X position	1179	**Y position**	108
Width	158	**Height**	32
Mode	Text	**Text OFF**	Back to System
Text ON	Text		

Dynamizations\Event	
Event name	Click

Function list\ActivateScreen

Screen name	System	**Object number**	0

Text field_4

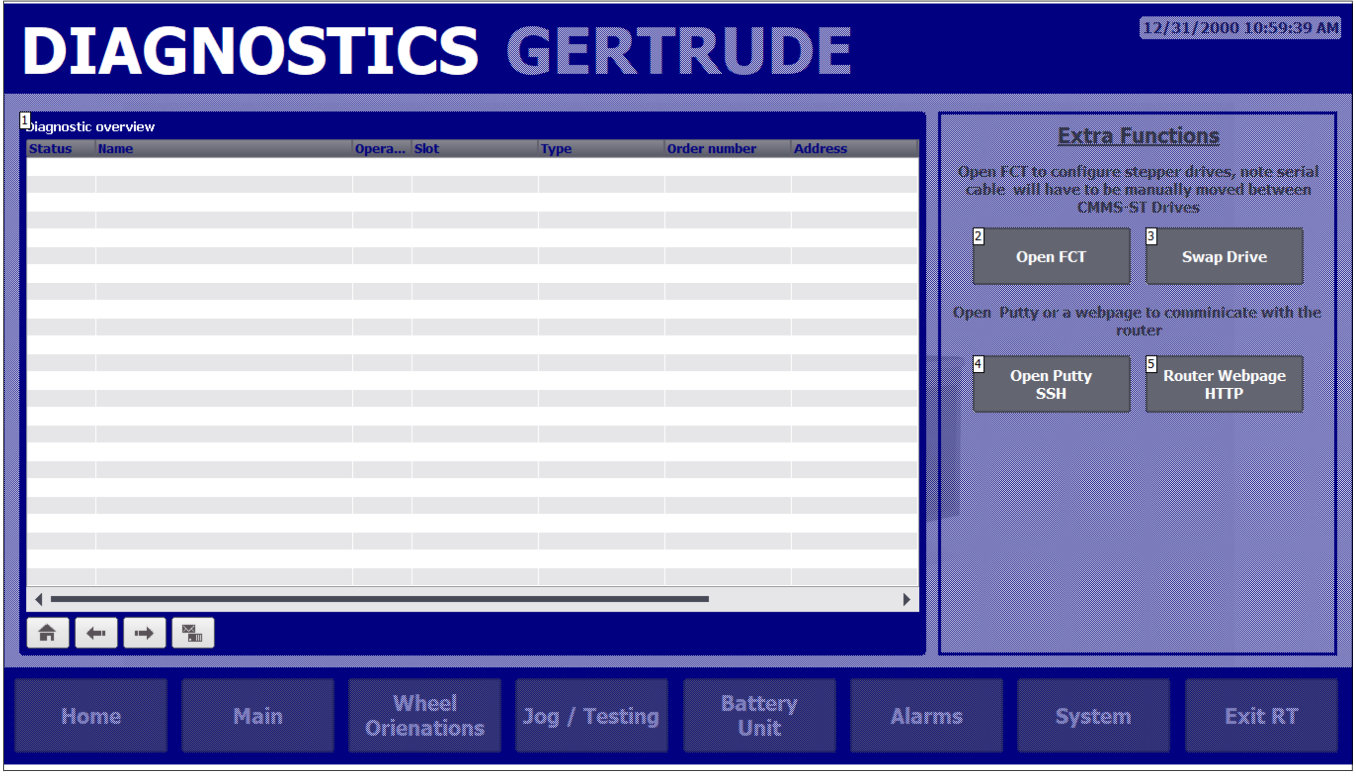
Type	Text field	**Name**	Text field_4
X position	583	**Y position**	391
Width	192	**Height**	34
Layer	0 - Layer_0	**Font**	Tahoma, 25px, style=Bold
Text	Loading		

HeadingText_ROUTER

Type	Text field	**Name**	HeadingText_ROUTER
X position	14	**Y position**	5
Width	271	**Height**	81
Layer	0 - Layer_0	**Font**	Tahoma, 64px, style=Bold
Text	ROUTER		

System

Hardcopy of System



Open FCT to configure stepper drives, note serial cable will have to be manually moved between CMMS-ST Drives

Open Putty or a webpage to comminicate with the router

Name	System	**Background color**	0, 0, 128
Grid color	0, 0, 0	**Number**	3
Template	Alex Temp A	**Tooltip**	

AGV_System_diagnostics_view

Type	System diagnostics view	**Name**	AGV_System_diagnostics_view
X position	16	**Y position**	109
Width	918	**Height**	549
Layer	0 - Layer_0		

Rectangle_1

Type	Rectangle	**Name**	Rectangle_1
X position	946	**Y position**	109
Width	404	**Height**	549
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

HeadingText_Extra_Functions

Type	Text field	**Name**	HeadingText_Extra_Functions
X position	1064	**Y position**	118
Width	169	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline

Text	Extra Functions

Text_ExtraFunctions_Paragraph1_CMMS-ST

Type	Text field	**Name**	Text_ExtraFunctions_Paragraph1_CMMS-ST
X position	963	**Y position**	158
Width	371	**Height**	58
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Open FCT to configure stepper drives, note serial cable will have to be manually moved between CMMS-ST Drives		

Button_Open_FCT

Type	Button	**Name**	Button_Open_FCT
X position	981	**Y position**	226
Width	160	**Height**	58
Mode	Text	**Text OFF**	Open FCT
Text ON	Text		

Dynamizations\Event	
Event name	Click

Function list\StartProgram

Program name	"C:\Users\Public\FCT.lnk"	**Program parameters**	
Display mode	Normal	**Wait for program to end**	No

Button_Swap_Drives

Type	Button	**Name**	Button_Swap_Drives
X position	1156	**Y position**	226
Width	160	**Height**	58
Mode	Text	**Text OFF**	Swap Drive
Text ON	Text		

Text_ExtraFunctions_Paragraph2_PuTTY

Type	Text field	**Name**	Text_ExtraFunctions_Paragraph2_PuTTY
X position	958	**Y position**	300
Width	382	**Height**	40
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Open Putty or a webpage to comminicate with the router		

Button_Open_PuTTY_SSH

Type	Button	**Name**	Button_Open_PuTTY_SSH
X position	981	**Y position**	355
Width	160	**Height**	58
Mode	Text	**Text OFF**	Open Putty SSH
Text ON	Text		

Dynamizations\Event	
Event name	Click

Function list\StartProgram

Program name	"C:\Program Files\PuTTY\putty.exe"	**Program parameters**	-load GoldenRouter -l admin -pw Siemens1200
Display mode	Normal	**Wait for program to end**	No

Button_Router_Webpage_HTTP

Type	Button	**Name**	Button_Router_Webpage_HTTP
X position	1156	**Y position**	355
Width	160	**Height**	58
Mode	Text	**Text OFF**	Router Webpage HTTP
Text ON	Text		

Dynamizations\Event	
Event name	Click

Function list\ActivateScreen

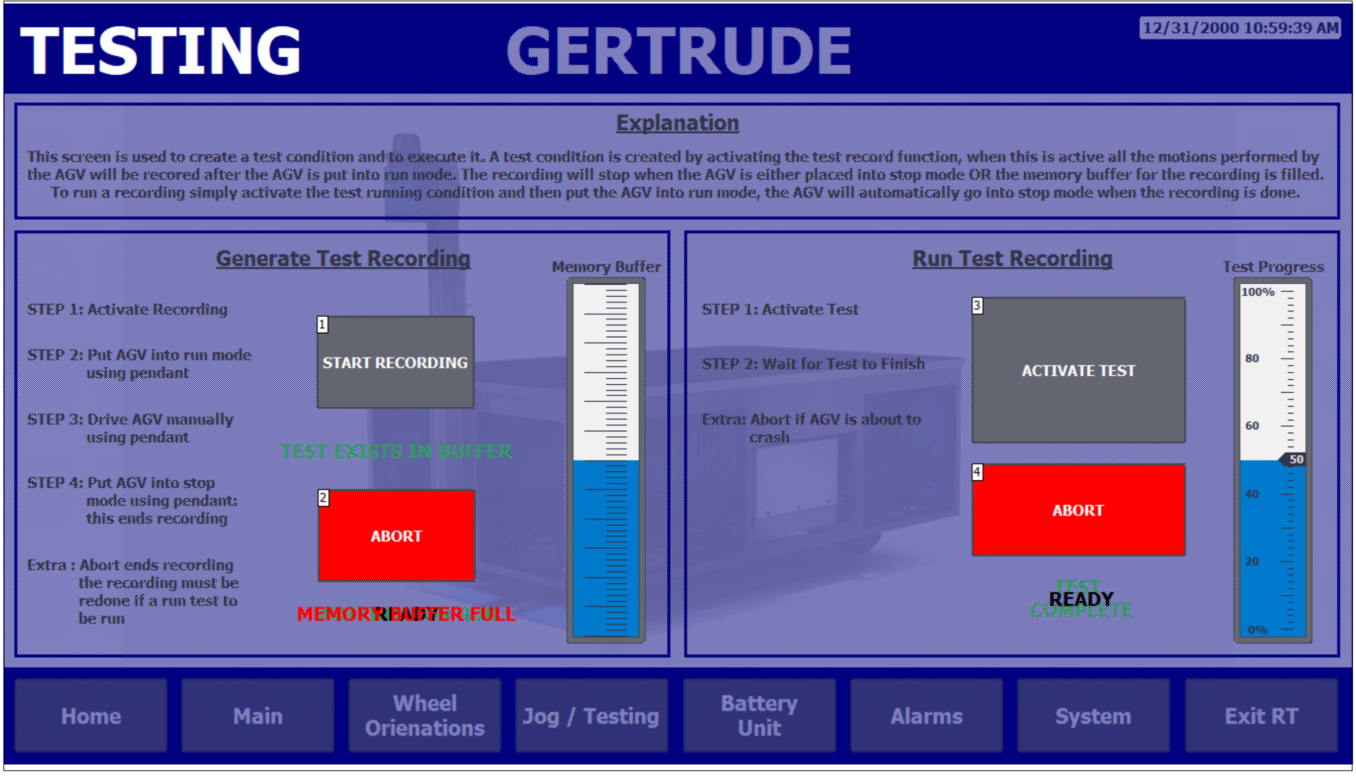
Screen name	RouterWebpage	**Object number**	0

HeadingText_DIAGNOSTICS

Type	Text field	**Name**	HeadingText_DIAGNOSTICS
X position	14	**Y position**	5
Width	469	**Height**	81
Layer	0 - Layer_0	**Font**	Tahoma, 64px, style=Bold
Text	DIAGNOSTICS		

TestRunner

Hardcopy of TestRunner



This screen is used to create a test condition and to execute it. A test condition is created by activating the test record function, when this is active all the motions performed by the AGV will be recored after the AGV is put into run mode. The recording will stop when the AGV is either placed into stop mode OR the memory buffer for the recording is filled. To run a recording simply activate the test running condition and then put the AGV into run mode, the AGV will automatically go into stop mode when the recording is done.

STEP 1: Activate Recording

STEP 2: Put AGV into run mode using pendant

STEP 3: Drive AGV manually using pendant

STEP 4: Put AGV into stop mode using pendant: this ends recording

Extra : Abort ends recording the recording must be redone if a run test to be run

STEP 1: Activate Test

STEP 2: Wait for Test to Finish

Extra: Abort if AGV is about to crash

Name	TestRunner	**Background color**	0, 0, 128
Grid color	0, 0, 0	**Number**	13
Template	Alex Temp A	**Tooltip**	

Rectangle_1

Type	Rectangle	**Name**	Rectangle_1
X position	11	**Y position**	100
Width	1342	**Height**	118
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

HeadingText_Explanation

Type	Text field	**Name**	HeadingText_Explanation
X position	617	**Y position**	105
Width	130	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Explanation		

Text_Explanation

Type	Text field	**Name**	Text_Explanation
X position	21	**Y position**	143
Width	1318	**Height**	58

Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	This screen is used to create a test condition and to execute it. A test condition is created by activating the test record function, when this is active all the motions performed by the AGV will be recored after the AGV is put into run mode. The recording will stop when the AGV is either placed into stop mode OR the memory buffer for the recording is filled. To run a recording simply activate the test running condition and then put the AGV into run mode, the AGV will automatically go into stop mode when the recording is done.		

Rectangle_2

Type	Rectangle	**Name**	Rectangle_2
X position	11	**Y position**	229
Width	664	**Height**	431
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

Rectangle_3

Type	Rectangle	**Name**	Rectangle_3
X position	689	**Y position**	229
Width	664	**Height**	431
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

HeadingText_Generate_Test_Recording

Type	Text field	**Name**	HeadingText_Generate_Test_Recording
X position	212	**Y position**	242
Width	263	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Generate Test Recording		

HeadingText_Run_TextRecording

Type	Text field	**Name**	HeadingText_Run_TextRecording
X position	917	**Y position**	242
Width	208	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Run Test Recording		

Button_StartRecording

Type	Button	**Name**	Button_StartRecording
X position	317	**Y position**	315
Width	160	**Height**	94
Mode	Text list	**Text OFF**	ACTIVATE RECORDING
Text ON	Text		

Dynamizations\Appearance

Tag - Cycle	HMI OP GDB_GenerateTestBusy -	**Data type**	Range
Range	0..0	**Foreground color**	255, 255, 255
Background color	99, 101, 113	**Flashing**	No
Range	1..1	**Foreground color**	0, 0, 0
Background color	255, 255, 0	**Flashing**	Yes

Dynamizations\Event

Event name	Press

Function list\SetBit

Tag	HMI OP GDB_GenerateTest SW

Dynamizations\Event

Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_GenerateTest SW

Dynamizations\Tag connection

Property name	Process value	**Tag**	HMI OP GDB_GenerateTestBusy

Button_ABORT_Recording

Type	Button	**Name**	Button_ABORT_Recording
X position	318	**Y position**	490
Width	160	**Height**	94
Mode	Text	**Text OFF**	ABORT
Text ON	Text		

Dynamizations\Event

Event name	Press

Function list\SetBit

Tag	HMI OP GDB_GenerateTestAbort SW

Dynamizations\Event

Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_GenerateTestAbort SW

Text_GenerateTestRecording_Step1

Type	Text field	**Name**	Text_GenerateTestRecording_Step1
X position	21	**Y position**	297
Width	208	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	STEP 1: Activate Recording		

Text_GenerateTestRecording_Step2

Type	Text field	**Name**	Text_GenerateTestRecording_Step2
X position	21	**Y position**	343

Width	232	**Height**	40
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	STEP 2: Put AGV into run mode using pendant		

Text_GenerateTestRecording_Step3

Type	Text field	**Name**	Text_GenerateTestRecording_Step3
X position	21	**Y position**	408
Width	214	**Height**	40
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	STEP 3: Drive AGV manually using pendant		

Text_GenerateTestRecording_Step4

Type	Text field	**Name**	Text_GenerateTestRecording_Step4
X position	21	**Y position**	472
Width	218	**Height**	58
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	STEP 4: Put AGV into stop mode using pendant: this ends recording		

Text_GenerateTestRecording_StepExtra

Type	Text field	**Name**	Text_GenerateTestRecording_StepExtra
X position	21	**Y position**	555
Width	223	**Height**	94
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Extra : Abort ends recording the recording must be redone if a run test to be run		

Bar_MemoryBufferUsage

Type	Bar	**Name**	Bar_MemoryBufferUsage
Y position	276	**X position**	570
Width	80	**Height**	370
Maximum value	6000	**Minimum value**	0
Process value	0		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_BufferSize

Text_MemoryBuffer

Type	Text field	**Name**	Text_MemoryBuffer
X position	552	**Y position**	254
Width	116	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Memory Buffer		

IndicatorText_MEMORY_BUFFER_FULL

Type	Text field	**Name**	IndicatorText_MEMORY_BUFFER_FULL
X position	294	**Y position**	602

Width	227	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	MEMORY BUFFER FULL		
Dynamizations\Visibility			
Tag - Cycle	HMI OP GDB_BufferSaturated LED -	**Data type**	Bit
Specifies the bit to monitor.	0	**Visibility**	Visible

IndicatorText_WAITING_FOR_RUN

Type	Text field	**Name**	IndicatorText_WAITING_FOR_RUN
X position	312	**Y position**	602
Width	192	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	WAITING FOR RUN		
Dynamizations\Visibility			
Tag - Cycle	HMI OP GDB_GenerateTestWait4Run -	**Data type**	Bit
Specifies the bit to monitor.	0	**Visibility**	Visible

IndicatorText_Test_Exists_In_Buffer

Type	Text field	**Name**	IndicatorText_Test_Exists_In_Buffer
X position	277	**Y position**	438
Width	240	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	TEST EXISTS IN BUFFER		
Dynamizations\Visibility			
Tag - Cycle	HMI OP GDB_GenTestDataDone LED -	**Data type**	Bit
Specifies the bit to monitor.	0	**Visibility**	Visible

HeadingText_TESTING

Type	Text field	**Name**	HeadingText_TESTING
X position	14	**Y position**	5
Width	291	**Height**	81
Layer	0 - Layer_0	**Font**	Tahoma, 64px, style=Bold
Text	TESTING		

Button_ACTIVATE_TEST

Type	Button	**Name**	Button_ACTIVATE_TEST
X position	980	**Y position**	296
Width	217	**Height**	148
Mode	Text list	**Text OFF**	RUN TEST FROM
Text ON	Text		
Dynamizations\Appearance			
Tag - Cycle	HMI OP GDB_RunTestInProgress -	**Data type**	Range
Range	0..0	**Foreground color**	255, 255, 255
Background color	99, 101, 113	**Flashing**	No
Range	1..1	**Foreground color**	0, 0, 0
Background color	255, 255, 0	**Flashing**	Yes
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_RunTestInProgress

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_RunTestEnableAGV

Function list\SetBit

Tag	HMI OP GDB_RunTest SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_RunTestEnableAGV

Function list\ResetBit

Tag	HMI OP GDB_RunTest SW

Bar_TestProgress

Type	Bar	**Name**	Bar_TestProgress
Y position	276	**X position**	1245
Width	80	**Height**	370
Maximum value	100	**Minimum value**	0
Process value	0		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_RunTestProgess

Text_Test_Progress

Type	Text field	**Name**	Text_Test_Progress
X position	1231	**Y position**	254
Width	108	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Test Progress		

IndicatorText_TEST_COMPLETE

Type	Text field	**Name**	IndicatorText_TEST_COMPLETE
X position	1036	**Y position**	575
Width	109	**Height**	50
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	TEST COMPLETE		
Dynamizations\Visibility			
Tag - Cycle	HMI OP GDB_RunTestDone LED -	**Data type**	Bit
Specifies the bit to monitor.	0	**Visibility**	Visible

Text field_16

Type	Text field	**Name**	Text field_16
X position	1055	**Y position**	587
Width	71	**Height**	27

Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	READY		
Dynamizations\Visibility			
Tag - Cycle	HMI OP GDB_RunTestDone LED -	**Data type**	Bit
Specifies the bit to monitor.	0	**Visibility**	Invisible

IndicatorText_Ready

Type	Text field	**Name**	IndicatorText_Ready
X position	372	**Y position**	602
Width	71	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	READY		
Dynamizations\Visibility			
Tag - Cycle	HMI OP GDB_GenTestReady -	**Data type**	Bit
Specifies the bit to monitor.	0	**Visibility**	Visible

Text_RunTestRecording_Step1

Type	Text field	**Name**	Text_RunTestRecording_Step1
X position	704	**Y position**	297
Width	164	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	STEP 1: Activate Test		

Text_RunTestRecording_Step2

Type	Text field	**Name**	Text_RunTestRecording_Step2
X position	704	**Y position**	352
Width	231	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	STEP 2: Wait for Test to Finish		

Text_RunTestRecording_StepExtra

Type	Text field	**Name**	Text_RunTestRecording_StepExtra
X position	704	**Y position**	408
Width	227	**Height**	40
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Extra: Abort if AGV is about to crash		

Button_ABORT_RunTest

Type	Button	**Name**	Button_ABORT_RunTest
X position	980	**Y position**	464
Width	217	**Height**	94
Mode	Text	**Text OFF**	ABORT
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_RunTestAbort

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_RunTestAbort

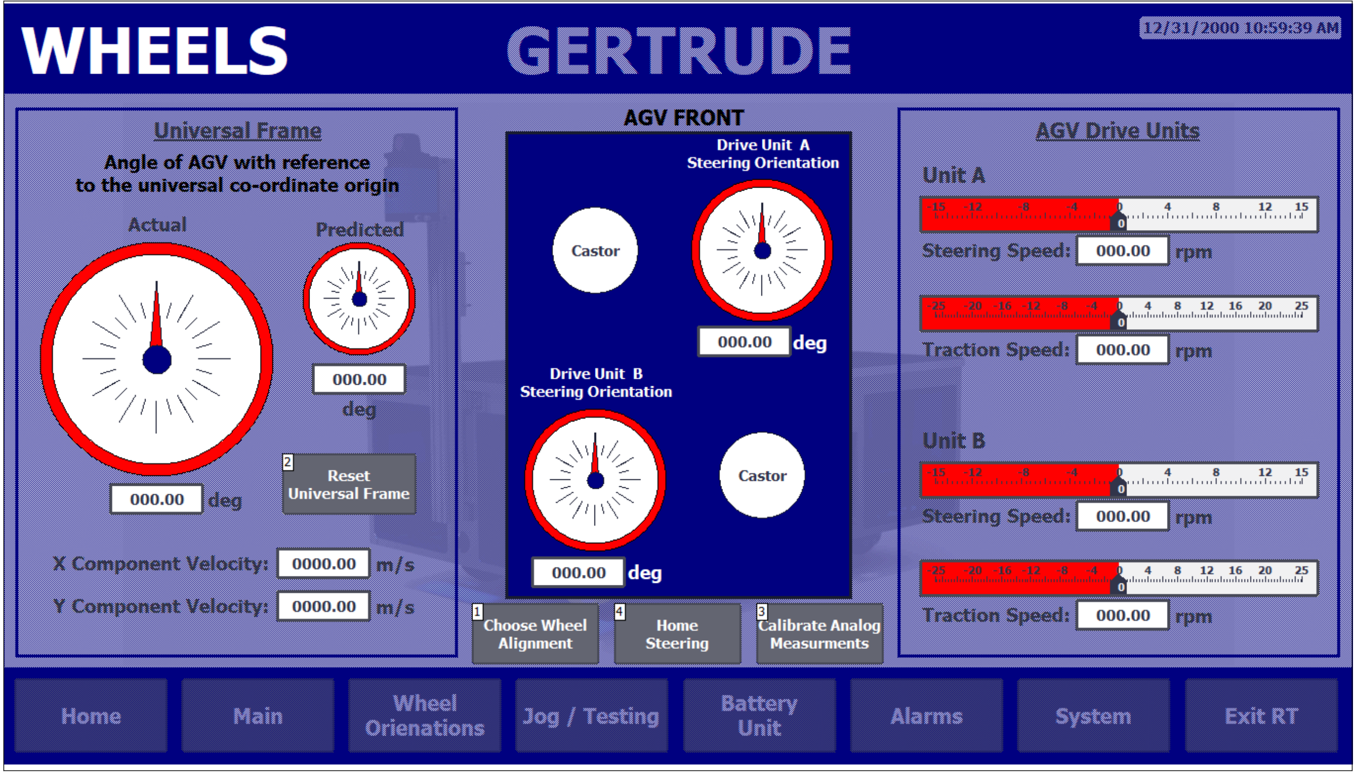
Dynamizations\Event	
Event name	Click

Function list\ResetBit

Tag	HMI OP GDB_RunTestDone LED

WheelOrientation

Hardcopy of WheelOrientation



Name	WheelOrientation	**Background color**	0, 0, 128
Grid color	0, 0, 0	**Number**	4
Template	Alex Temp A	**Tooltip**	

Gauge_UnitA_Angle

Type	Gauge	**Name**	Gauge_UnitA_Angle
X position	695	**Y position**	176
Width	147	**Height**	147
Minimum value	-360	**Maximum value**	0
Process value	50		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_SteeringAngleA_Dial

Rectangle_1

Type	Rectangle	**Name**	Rectangle_1
X position	508	**Y position**	129
Width	351	**Height**	472
Layer	0 - Layer_0	**Background color**	0, 0, 128
Border color	24, 28, 49		

Gauge_UnitB_Angle

Type	Gauge	**Name**	Gauge_UnitB_Angle

X position	526	**Y position**	408
Width	147	**Height**	147
Minimum value	-360	**Maximum value**	0
Process value	50		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_SteeringAngleB_Dial

Circle_1

Type	Circle	**Name**	Circle_1
X position	555	**Y position**	205
Width	88	**Height**	88
Radius	44	**Background color**	255, 255, 255
Border color	24, 28, 49		

Circle_2

Type	Circle	**Name**	Circle_2
X position	724	**Y position**	432
Width	88	**Height**	88
Radius	44	**Background color**	255, 255, 255
Border color	24, 28, 49		

I/O_Field_UnitA_Angle

Type	I/O field	**Name**	I/O_Field_UnitA_Angle
X position	702	**Y position**	325
Width	96	**Height**	32
Layer	0 - Layer_0	**Mode**	Output
Font	Tahoma, 15px, style=Bold		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_SteeringAngleA

I/O_Field_UnitB_Angle

Type	I/O field	**Name**	I/O_Field_UnitB_Angle
X position	534	**Y position**	558
Width	96	**Height**	32
Layer	0 - Layer_0	**Mode**	Output
Font	Tahoma, 15px, style=Bold		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_SteeringAngleB

HeadingText_AGV_FRONT

Type	Text field	**Name**	HeadingText_AGV_FRONT
X position	625	**Y position**	100
Width	127	**Height**	29
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold
Text	AGV FRONT		

Text_Drive_Unit_B_Steering_Orientation

Type	Text field	**Name**	Text_Drive_Unit_B_Steering_Orientation
X position	520	**Y position**	362

Width	159	**Height**	40
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Drive Unit B Steering Orientation		

Text_Drive_Unit_A_Steering_Orientation

Type	Text field	**Name**	Text_Drive_Unit_A_Steering_Orientation
X position	689	**Y position**	131
Width	159	**Height**	40
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Drive Unit A Steering Orientation		

Gauge_ActualAGVAngle

Type	Gauge	**Name**	Gauge_ActualAGVAngle
X position	36	**Y position**	240
Width	238	**Height**	238
Minimum value	-360	**Maximum value**	0
Process value	50		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_YawAngleACT_Dial

Text_UniversalFame_Explanation

Type	Text field	**Name**	Text_UniversalFame_Explanation
X position	70	**Y position**	146
Width	333	**Height**	50
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	Angle of AGV with reference to the universal co-ordinate origin		

I/O_Field_ActualAGVAngle

Type	I/O field	**Name**	I/O_Field_ActualAGVAngle
X position	107	**Y position**	484
Width	96	**Height**	32
Layer	0 - Layer_0	**Mode**	Output
Font	Tahoma, 15px, style=Bold		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_YawAngleACT

Text field_5

Type	Text field	**Name**	Text field_5
X position	19	**Y position**	535
Width	15	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text			

Rectangle_2

Type	Rectangle	**Name**	Rectangle_2
X position	12	**Y position**	105
Width	448	**Height**	555

Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

Text_Castorlabel_2

Type	Text field	**Name**	Text_Castorlabel_2
X position	741	**Y position**	465
Width	54	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Castor		

Text_Castorlabel_1

Type	Text field	**Name**	Text_Castorlabel_1
X position	572	**Y position**	238
Width	54	**Height**	22
Layer	0 - Layer_0	**Font**	Tahoma, 15px, style=Bold
Text	Castor		

HeadingText_Univeral_Frame

Type	Text field	**Name**	HeadingText_Univeral_Frame
X position	149	**Y position**	113
Width	175	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	Universal Frame		

Text_X_Component_Velocity:

Type	Text field	**Name**	Text_X_Component_Velocity:
X position	47	**Y position**	551
Width	230	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	X Component Velocity:		

Text_Y_Component_Velocity:

Type	Text field	**Name**	Text_Y_Component_Velocity:
X position	47	**Y position**	594
Width	230	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	Y Component Velocity:		

I/O_Field_XComponentVelocity

Type	I/O field	**Name**	I/O_Field_XComponentVelocity
X position	276	**Y position**	549
Width	96	**Height**	32
Layer	0 - Layer_0	**Mode**	Output
Font	Tahoma, 15px, style=Bold		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_XCompVel

I/O_Field_YComponentVelocity

Type	I/O field	**Name**	I/O_Field_YComponentVelocity
X position	276	**Y position**	592

Width	96	**Height**	32
Layer	0 - Layer_0	**Mode**	Output
Font	Tahoma, 15px, style=Bold		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_YCompVel

Text_m/sLabel_1

Type	Text field	**Name**	Text_m/sLabel_1
X position	374	**Y position**	552
Width	44	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	m/s		

Text_m/sLabel_2

Type	Text field	**Name**	Text_m/sLabel_2
X position	374	**Y position**	595
Width	44	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	m/s		

Text_deglabel_2

Type	Text field	**Name**	Text_deglabel_2
X position	204	**Y position**	487
Width	40	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	deg		

Text_deglabel_4

Type	Text field	**Name**	Text_deglabel_4
X position	629	**Y position**	560
Width	40	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	deg		

Text_deglabel_3

Type	Text field	**Name**	Text_deglabel_3
X position	796	**Y position**	328
Width	40	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	deg		

Rectangle_3

Type	Rectangle	**Name**	Rectangle_3
X position	905	**Y position**	105
Width	448	**Height**	555
Layer	0 - Layer_0	**Background color**	217, 217, 217
Border color	0, 0, 128		

HeadingText_AGV_DRIVE_UNITS

Type	Text field	**Name**	HeadingText_AGV_DRIVE_UNITS

X position	1042	**Y position**	113
Width	171	**Height**	30
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold, Underline
Text	AGV Drive Units		

HeadingText_UnitA

Type	Text field	**Name**	HeadingText_UnitA
X position	927	**Y position**	158
Width	69	**Height**	29
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold
Text	Unit A		

Text_A_Steering_Speed:

Type	Text field	**Name**	Text_A_Steering_Speed:
X position	927	**Y position**	235
Width	156	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	Steering Speed:		

Text_A_Traction_Speed

Type	Text field	**Name**	Text_A_Traction_Speed
X position	927	**Y position**	335
Width	155	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	Traction Speed:		

I/O_Field_SteeringSpeedA

Type	I/O field	**Name**	I/O_Field_SteeringSpeedA
X position	1085	**Y position**	233
Width	96	**Height**	32
Layer	0 - Layer_0	**Mode**	Output
Font	Tahoma, 15px, style=Bold		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_SteeringRPM_A

I/O_Field_TractionSpeedA

Type	I/O field	**Name**	I/O_Field_TractionSpeedA
X position	1085	**Y position**	333
Width	96	**Height**	32
Layer	0 - Layer_0	**Mode**	Output
Font	Tahoma, 15px, style=Bold		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_TractionRPM_A

Text_RPMlabel_1

Type	Text field	**Name**	Text_RPMlabel_1
X position	1183	**Y position**	236
Width	43	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	rpm		

Text_RPMlabel_2

Type	Text field	**Name**	Text_RPMlabel_2
X position	1183	**Y position**	336
Width	43	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	rpm		

Bar_SteeringSpeedA

Type	Bar	**Name**	Bar_SteeringSpeedA
Y position	194	**X position**	927
Width	405	**Height**	38
Maximum value	15	**Minimum value**	-15
Process value	0		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_SteeringRPM_A

Bar_TractionSpeedA

Type	Bar	**Name**	Bar_TractionSpeedA
Y position	294	**X position**	927
Width	405	**Height**	38
Maximum value	25	**Minimum value**	-25
Process value	0		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_TractionRPM_A

HeadingText_UnitB

Type	Text field	**Name**	HeadingText_UnitB
X position	927	**Y position**	426
Width	69	**Height**	29
Layer	0 - Layer_0	**Font**	Tahoma, 21px, style=Bold
Text	Unit B		

Text_B_Steering_Speed:

Type	Text field	**Name**	Text_B_Steering_Speed:
X position	927	**Y position**	503
Width	156	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	Steering Speed:		

Text_B_Traction_Speed

Type	Text field	**Name**	Text_B_Traction_Speed
X position	927	**Y position**	603
Width	155	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	Traction Speed:		

I/O_Field_SteeringSpeedB

Type	I/O field	**Name**	I/O_Field_SteeringSpeedB
X position	1085	**Y position**	501

Width	96	**Height**	32
Layer	0 - Layer_0	**Mode**	Output
Font	Tahoma, 15px, style=Bold		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_SteeringRPM_B

I/O_Field_TractionSpeedB

Type	I/O field	**Name**	I/O_Field_TractionSpeedB
X position	1085	**Y position**	601
Width	96	**Height**	32
Layer	0 - Layer_0	**Mode**	Output
Font	Tahoma, 15px, style=Bold		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_TractionRPM_B

Text_RPMlabel_3

Type	Text field	**Name**	Text_RPMlabel_3
X position	1183	**Y position**	504
Width	43	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	rpm		

Text_RPMlabel_4

Type	Text field	**Name**	Text_RPMlabel_4
X position	1183	**Y position**	604
Width	43	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	rpm		

Bar_SteeringSpeedB

Type	Bar	**Name**	Bar_SteeringSpeedB
Y position	462	**X position**	927
Width	405	**Height**	38
Maximum value	15	**Minimum value**	-15
Process value	0		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_SteeringRPM_B

Bar_TractionSpeedB

Type	Bar	**Name**	Bar_TractionSpeedB
Y position	561	**X position**	927
Width	405	**Height**	38
Maximum value	25	**Minimum value**	-25
Process value	0		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_TractionRPM_B

Button_Choose_Wheel_Alignment

Type	Button	**Name**	Button_Choose_Wheel_Alignment
X position	474	**Y position**	605

Width	129	**Height**	62
Mode	Text	**Text OFF**	Choose Wheel Alignment
Text ON	Text		

Dynamizations\Event	
Event name	Click

Function list\ActivateScreen

Screen name	ChoseOrintMode	**Object number**	0

Gauge_PredictedAGVAngle

Type	Gauge	**Name**	Gauge_PredictedAGVAngle
X position	302	**Y position**	240
Width	116	**Height**	116
Minimum value	-360	**Maximum value**	0
Process value	50		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_YawAngleSET_Dial

I/O_Field_PredictedAGVAngle

Type	I/O field	**Name**	I/O_Field_PredictedAGVAngle
X position	312	**Y position**	363
Width	96	**Height**	32
Layer	0 - Layer_0	**Mode**	Output
Font	Tahoma, 15px, style=Bold		
Dynamizations\Tag connection			
Property name	Process value	**Tag**	HMI OP GDB_YawAngleSET

Text field_27

Type	Text field	**Name**	Text field_27
X position	123	**Y position**	209
Width	65	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	Actual		

Text field_28

Type	Text field	**Name**	Text field_28
X position	313	**Y position**	214
Width	95	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	Predicted		

Button_Reset_Universal_Frame

Type	Button	**Name**	Button_Reset_Universal_Frame
X position	282	**Y position**	454
Width	136	**Height**	62
Mode	Text	**Text OFF**	Reset Universal Frame
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_ResetUniversalFrame_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_ResetUniversalFrame_SW

Text_deglabel_1

Type	Text field	**Name**	Text_deglabel_1
X position	340	**Y position**	395
Width	40	**Height**	27
Layer	0 - Layer_0	**Font**	Tahoma, 19px, style=Bold
Text	deg		

Button_Calibrate_Analog_Measurements

Type	Button	**Name**	Button_Calibrate_Analog_Measurements
X position	762	**Y position**	605
Width	129	**Height**	62
Mode	Text	**Text OFF**	Calibrate Analog Measurments
Text ON	Text		

Dynamizations\Event	
Event name	Click

Function list\ActivateScreen

Screen name	CalibrateAnalogs	**Object number**	0

Button_Home_Steering

Type	Button	**Name**	Button_Home_Steering
X position	618	**Y position**	605
Width	129	**Height**	62
Mode	Text	**Text OFF**	Home Steering
Text ON	Text		

Dynamizations\Event	
Event name	Press

Function list\SetBit

Tag	HMI OP GDB_HomeSteering_SW

Dynamizations\Event	
Event name	Release

Function list\ResetBit

Tag	HMI OP GDB_HomeSteering_SW

HeadingText_WHEELS

Type	Text field	**Name**	HeadingText_WHEELS
X position	14	**Y position**	5
Width	276	**Height**	81
Layer	0 - Layer_0	**Font**	Tahoma, 64px, style=Bold
Text	WHEELS		

V.3 HMI Tags

Code can be found on next page

HMI tags

Default tag table [81]

s7-1500_Status

Name	s7-1500_Status	**Display name**	
Address		**Connection**	<Internal tag>
Data type	SInt	**Length**	1

SoftPLC_Status

Name	SoftPLC_Status	**Display name**	
Address		**Connection**	<Internal tag>
Data type	SInt	**Length**	1

HMI OP GDB_AGVStateLED

Name	HMI OP GDB_AGVStateLED	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_SystemHomingLED

Name	HMI OP GDB_SystemHomingLED	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_ManualModeLED

Name	HMI OP GDB_ManualModeLED	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_AutomaticModeLED

Name	HMI OP GDB_AutomaticModeLED	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_CommisionModeLED

Name	HMI OP GDB_CommisionModeLED	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_AGVMovingLED

Name	HMI OP GDB_AGVMovingLED	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_AGVWheelErrorLED

Name	HMI OP GDB_AGVWheelErrorLED	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_FrontEStopLED

Name	HMI OP GDB_FrontEStopLED	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_RearEStopLED

Name	HMI OP GDB_RearEStopLED	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_AcknowledgeLED

Name	HMI OP GDB_AcknowledgeLED	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_SteeringAngleA_Dial

Name	HMI OP GDB_SteeringAngleA_Dial	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Real	**Length**	4

HMI OP GDB_SteeringAngleB_Dial

Name	HMI OP GDB_SteeringAngleB_Dial	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Real	**Length**	4

HMI OP GDB_SteeringAngleA

Name	HMI OP GDB_SteeringAngleA	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Real	**Length**	4

HMI OP GDB_SteeringAngleB

Name	HMI OP GDB_SteeringAngleB	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Real	**Length**	4

HMI OP GDB_YawAngleACT_Dial

Name	HMI OP GDB_YawAngleACT_Dial	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Real	**Length**	4

HMI OP GDB_YawAngleACT

Name	HMI OP GDB_YawAngleACT	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Real	**Length**	4

HMI OP GDB_XCompVel

Name	HMI OP GDB_XCompVel	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Real	**Length**	4

HMI OP GDB_YCompVel

Name	HMI OP GDB_YCompVel	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Real	**Length**	4

HMI OP GDB_SteeringRPM_A

Name	HMI OP GDB_SteeringRPM_A	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Real	**Length**	4

HMI OP GDB_TractionRPM_A

Name	HMI OP GDB_TractionRPM_A	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Real	**Length**	4

HMI OP GDB_SteeringRPM_B

Name	HMI OP GDB_SteeringRPM_B	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Real	**Length**	4

HMI OP GDB_TractionRPM_B

Name	HMI OP GDB_TractionRPM_B	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Real	**Length**	4

HMI OP GDB_AGVON/OFF_SW_FeedBCK

Name	HMI OP GDB_AGVON/OFF_SW_FeedBCK	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_ManualMode_SW

Name	HMI OP GDB_ManualMode_SW	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_AutoMode_SW

Name	HMI OP GDB_AutoMode_SW	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_CommisinMode_SW

Name	HMI OP GDB_CommisinMode_SW	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_HomeSteering_SW

Name	HMI OP GDB_HomeSteering_SW	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_AGVON/OFF_SW

Name	HMI OP GDB_AGVON/OFF_SW	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_YawAngleSET_Dial

Name	HMI OP GDB_YawAngleSET_Dial	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Real	**Length**	4

HMI OP GDB_YawAngleSET

Name	HMI OP GDB_YawAngleSET	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Real	**Length**	4

HMI OP GDB_ResetUniversalFrame_SW

Name	HMI OP GDB_ResetUniversalFrame_SW	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_UnitA_Cali_CW_SW

Name	HMI OP GDB_UnitA_Cali_CW_SW	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_UnitA_Cali_CCW_SW

Name	HMI OP GDB_UnitA_Cali_CCW_SW	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_UnitB_Cali_CCW_SW

Name	HMI OP GDB_UnitB_Cali_CCW_SW	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_UnitB_Cali_CW_SW

Name	HMI OP GDB_UnitB_Cali_CW_SW	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_CalibratePots_SW

Name	HMI OP GDB_CalibratePots_SW	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_CalibratePots_DONE_LED

Name	HMI OP GDB_CalibratePots_DONE_LED	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_CalibrateFCTAnalogs_SW

Name	HMI OP GDB_CalibrateFCTAnalogs_SW	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_CalibrateFCTAnalogs_DONE_LED

Name	HMI OP GDB_CalibrateFCTAnalogs_DONE_LED	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_CalibrateStepprSpd

Name	HMI OP GDB_CalibrateStepprSpd	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Int	**Length**	2

HMI OP GDB_AGVJog_SW

Name	HMI OP GDB_AGVJog_SW	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_JogSpd_SET

Name	HMI OP GDB_JogSpd_SET	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Int	**Length**	2

HMI OP GDB_JogStrafeAngle_SET

Name	HMI OP GDB_JogStrafeAngle_SET	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Int	**Length**	2

HMI OP GDB_JogStrafeAngle_FeedBCK

Name	HMI OP GDB_JogStrafeAngle_FeedBCK	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Real	**Length**	4

HMI OP GDB_JogAckrAngle_SET

Name	HMI OP GDB_JogAckrAngle_SET	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Int	**Length**	2

HMI OP GDB_JogAckrAngle_FeedBCK

Name	HMI OP GDB_JogAckrAngle_FeedBCK	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Real	**Length**	4

HMI OP GDB_AGVJog_Execute_SW

Name	HMI OP GDB_AGVJog_Execute_SW	**Display name**	
Address		**Connection**	HMI_s7-1500

Data type	Bool	**Length**	1

HMI OP GDB_UnitA_Servo_CCW_SW

Name	HMI OP GDB_UnitA_Servo_CCW_SW	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_UnitA_Servo_CW_SW

Name	HMI OP GDB_UnitA_Servo_CW_SW	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_UnitB_Servo_CCW_SW

Name	HMI OP GDB_UnitB_Servo_CCW_SW	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_UnitB_Servo_CW_SW

Name	HMI OP GDB_UnitB_Servo_CW_SW	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

Alarms GDB_StepperAlarmWord

Name	Alarms GDB_StepperAlarmWord	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Word	**Length**	2

Alarms GDB_ServoAlarmWord

Name	Alarms GDB_ServoAlarmWord	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Word	**Length**	2

Alarms GDB_STOAlarmWord

Name	Alarms GDB_STOAlarmWord	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Word	**Length**	2

Alarms GDB_WheelAlignAlarmWord

Name	Alarms GDB_WheelAlignAlarmWord	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Word	**Length**	2

Alarms GDB_GeneralAlarmsWord

Name	Alarms GDB_GeneralAlarmsWord	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Word	**Length**	2

HMI OP GDB_Home2Zero SW

Name	HMI OP GDB_Home2Zero SW	**Display name**	

Address		Connection	HMI_s7-1500
Data type	Bool	Length	1

HMI OP GDB_Home2SameAng SW

Name	HMI OP GDB_Home2SameAng SW	Display name	
Address		Connection	HMI_s7-1500
Data type	Bool	Length	1

HMI OP GDB_AllowAlignmnt

Name	HMI OP GDB_AllowAlignmnt	Display name	
Address		Connection	HMI_s7-1500
Data type	Bool	Length	1

HMI OP GDB_JogUnit SW

Name	HMI OP GDB_JogUnit SW	Display name	
Address		Connection	HMI_s7-1500
Data type	Bool	Length	1

HMI OP GDB_JogUnit LED

Name	HMI OP GDB_JogUnit LED	Display name	
Address		Connection	HMI_s7-1500
Data type	Bool	Length	1

HMI OP GDB_AGVJog SW FDBCK

Name	HMI OP GDB_AGVJog SW FDBCK	Display name	
Address		Connection	HMI_s7-1500
Data type	Bool	Length	1

HMI OP GDB_AGVON/OFF_JogUnit_SW

Name	HMI OP GDB_AGVON/OFF_JogUnit_SW	Display name	
Address		Connection	HMI_s7-1500
Data type	Bool	Length	1

HMI OP GDB_AGVON/OFF_JogAGV_SW

Name	HMI OP GDB_AGVON/OFF_JogAGV_SW	Display name	
Address		Connection	HMI_s7-1500
Data type	Bool	Length	1

HMI OP GDB_GenerateTestWait4Run

Name	HMI OP GDB_GenerateTestWait4Run	Display name	
Address		Connection	HMI_s7-1500
Data type	Bool	Length	1

HMI OP GDB_BufferSaturated LED

Name	HMI OP GDB_BufferSaturated LED	Display name	
Address		Connection	HMI_s7-1500
Data type	Bool	Length	1

HMI OP GDB_GenerateTest SW

Name	HMI OP GDB_GenerateTest SW	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_BufferSize

Name	HMI OP GDB_BufferSize	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Int	**Length**	2

HMI OP GDB_GenerateTestAbort SW

Name	HMI OP GDB_GenerateTestAbort SW	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_GenerateTestBusy

Name	HMI OP GDB_GenerateTestBusy	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_GenTestDataDone LED

Name	HMI OP GDB_GenTestDataDone LED	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_RunTestInProgress

Name	HMI OP GDB_RunTestInProgress	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_RunTest SW

Name	HMI OP GDB_RunTest SW	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_RunTestProgess

Name	HMI OP GDB_RunTestProgess	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Int	**Length**	2

HMI OP GDB_RunTestDone LED

Name	HMI OP GDB_RunTestDone LED	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_GenTestReady

Name	HMI OP GDB_GenTestReady	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_RunTestEnableAGV

Name	HMI OP GDB_RunTestEnableAGV	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI OP GDB_RunTestAbort

Name	HMI OP GDB_RunTestAbort	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

HMI tags

Temp [1]

IncrimentTemp

Name	IncrimentTemp	**Display name**	
Address		**Connection**	<Internal tag>
Data type	Int	**Length**	2

HMI tags

VBScript Triggers [2]

ShutdownRouter_Trigger

Name	ShutdownRouter_Trigger	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

Dynamizations\Event	
Event name	Value change

Function list\ShutdownRouter

ShutdownIPC_Trigger

Name	ShutdownIPC_Trigger	**Display name**	
Address		**Connection**	HMI_s7-1500
Data type	Bool	**Length**	1

Dynamizations\Event	
Event name	Value change

Function list\ShutdownIPC

V.4 Connections

Code can be found on next page

Connections

HMI_s7-1500

Name	HMI_s7-1500	**Communication driver**	SIMATIC S7 1500	**Comment**	

HMI_SoftPLC

Name	HMI_SoftPLC	**Communication driver**	SIMATIC S7 1500	**Comment**	

V.5 HMI Alarms

Code can be found on next page

HMI alarms

Discrete alarms

GnrlAGVoff

Name	GnrlAGVoff	**ID**	31
Alarm class	Warnings	**Alarm text**	AGV is OFF
Alarm group	<No alarm group>		

GeneralAlarmsWord

GnrlAGVon

Name	GnrlAGVon	**ID**	30
Alarm class	Warnings	**Alarm text**	AGV is ON
Alarm group	<No alarm group>		

GeneralAlarmsWord

GnrlAutoMode

Name	GnrlAutoMode	**ID**	33
Alarm class	Warnings	**Alarm text**	AGV is in automatic mode
Alarm group	<No alarm group>		

GeneralAlarmsWord

GnrlBodJog

Name	GnrlBodJog	**ID**	35
Alarm class	Warnings	**Alarm text**	AGV is in HMI body jog mode
Alarm group	<No alarm group>		

GeneralAlarmsWord

GnrlCalABSPot

Name	GnrlCalABSPot	**ID**	28
Alarm class	Warnings	**Alarm text**	Calibration of absolute angle encoders completed
Alarm group	<No alarm group>		

GeneralAlarmsWord

GnrlCalStrSclr

Name	GnrlCalStrSclr	**ID**	29
Alarm class	Warnings	**Alarm text**	Calibration of steering angle scaler completed
Alarm group	<No alarm group>		

GeneralAlarmsWord

GnrlCmisMode

Name	GnrlCmisMode	**ID**	34

Alarm class	Warnings	**Alarm text**	AGV is in commisioning mode
Alarm group	<No alarm group>		

GeneralAlarmsWord

GnrlIntgrARst

Name	GnrlIntgrARst	**ID**	37
Alarm class	Warnings	**Alarm text**	Unit A steering angle intergrated from steering speed reset to zero
Alarm group	<No alarm group>		

GeneralAlarmsWord

GnrlIntgrBRst

Name	GnrlIntgrBRst	**ID**	38
Alarm class	Warnings	**Alarm text**	Unit B steering angle intergrated from steering speed reset to zero
Alarm group	<No alarm group>		

GeneralAlarmsWord

GnrlManMode

Name	GnrlManMode	**ID**	32
Alarm class	Warnings	**Alarm text**	AGV is in manual mode
Alarm group	<No alarm group>		

GeneralAlarmsWord

GnrlUniFrmRst

Name	GnrlUniFrmRst	**ID**	36
Alarm class	Warnings	**Alarm text**	Universal reference reset to AGV's current location
Alarm group	<No alarm group>		

GeneralAlarmsWord

ServoAackError

Name	ServoAackError	**ID**	12
Alarm class	Errors	**Alarm text**	servo A did not reset from fault correctly
Alarm group	<No alarm group>		

ServoAlarmWord

ServoAHaltError

Name	ServoAHaltError	**ID**	11
Alarm class	Errors	**Alarm text**	servo A is not able to halt correctly
Alarm group	<No alarm group>		

ServoAlarmWord

ServoAmoveError

Name	ServoAmoveError	**ID**	10
Alarm class	Errors	**Alarm text**	servo A powered but unable to move
Alarm group	<No alarm group>		

ServoAlarmWord

ServoApwrError

Name	ServoApwrError	**ID**	9
Alarm class	Errors	**Alarm text**	servo A is not able to power on
Alarm group	<No alarm group>		

ServoAlarmWord

ServoBackError

Name	ServoBackError	**ID**	16
Alarm class	Errors	**Alarm text**	servo B did not reset from fault correctly
Alarm group	<No alarm group>		

ServoAlarmWord

ServoBHaltError

Name	ServoBHaltError	**ID**	15
Alarm class	Errors	**Alarm text**	servo B is not able to halt correctly
Alarm group	<No alarm group>		

ServoAlarmWord

ServoBmoveError

Name	ServoBmoveError	**ID**	14
Alarm class	Errors	**Alarm text**	servo B powered but unable to move
Alarm group	<No alarm group>		

ServoAlarmWord

ServoBpwrError

Name	ServoBpwrError	**ID**	13
Alarm class	Errors	**Alarm text**	servo B is not able to power on
Alarm group	<No alarm group>		

ServoAlarmWord

STOAckFailure

Name	STOAckFailure	**ID**	19
Alarm class	Warnings	**Alarm text**	STO acknowledge required: FAULT STILL PRESENT
Alarm group	<No alarm group>		

STOAlarmWord

STOAckReq

Name	STOAckReq	**ID**	18
Alarm class	Warnings	**Alarm text**	STO acknowledge required: FAULT NOT FRESENT
Alarm group	<No alarm group>		

STOAlarmWord

STOFrontEStop

Name	STOFrontEStop	**ID**	20
Alarm class	Warnings	**Alarm text**	Front E-Stop activated
Alarm group	<No alarm group>		

STOAlarmWord

STORearEStop

Name	STORearEStop	**ID**	21
Alarm class	Warnings	**Alarm text**	Rear E-Stop activated
Alarm group	<No alarm group>		

STOAlarmWord

STOState

Name	STOState	**ID**	17
Alarm class	Warnings	**Alarm text**	AGV is in Safe Torque Off (STO) state
Alarm group	<No alarm group>		

STOAlarmWord

StprAControlError

Name	StprAControlError	**ID**	2
Alarm class	Errors	**Alarm text**	unable to control stepper drive A
Alarm group	<No alarm group>		

StepperAlarmWord

StprAHomeError

Name	StprAHomeError	**ID**	4
Alarm class	Warnings	**Alarm text**	stepper drive A is not correctly homed
Alarm group	<No alarm group>		

StepperAlarmWord

StprAReadError

Name	StprAReadError	**ID**	1
Alarm class	Errors	**Alarm text**	cannot read data from stepper drive A
Alarm group	<No alarm group>		

StepperAlarmWord

StprAWriteError

Name	StprAWriteError	**ID**	3
Alarm class	Errors	**Alarm text**	cannot write data to stepper drive A
Alarm group	<No alarm group>		

StepperAlarmWord

StprBControlError

Name	StprBControlError	**ID**	6
Alarm class	Errors	**Alarm text**	unable to control stepper drive B
Alarm group	<No alarm group>		

StepperAlarmWord

StprBHomeError

Name	StprBHomeError	**ID**	8
Alarm class	Warnings	**Alarm text**	stepper drive A is not correctly homed
Alarm group	<No alarm group>		

StepperAlarmWord

StprBReadError

Name	StprBReadError	**ID**	5
Alarm class	Errors	**Alarm text**	cannot read data from stepper drive B
Alarm group	<No alarm group>		

StepperAlarmWord

StprBWriteError

Name	StprBWriteError	**ID**	7
Alarm class	Errors	**Alarm text**	cannot write data to stepper drive B
Alarm group	<No alarm group>		

StepperAlarmWord

WhlAlignInProgress

Name	WhlAlignInProgress	**ID**	23
Alarm class	Warnings	**Alarm text**	Wheels alignment is active
Alarm group	<No alarm group>		

WheelAlignAlarmWord

WhlAlignManReq

Name	WhlAlignManReq	**ID**	24
Alarm class	Warnings	**Alarm text**	Manaul alignment of teh wheels is necessary using the jog screen
Alarm group	<No alarm group>		

WheelAlignAlarmWord

WhlAlignNotAlignd

Name	WhlAlignNotAlignd	**ID**	22
Alarm class	Warnings	**Alarm text**	Wheels are not aligned correctly for startup
Alarm group	<No alarm group>		

WheelAlignAlarmWord

WhlAlignSeqFail

Name	WhlAlignSeqFail	**ID**	27
Alarm class	Warnings	**Alarm text**	Failure of graph sequencer for wheel alignment
Alarm group	<No alarm group>		

WheelAlignAlarmWord

WhlAlignZeroEnc

Name	WhlAlignZeroEnc	**ID**	25
Alarm class	Warnings	**Alarm text**	Zero of stepper drive encoders underway
Alarm group	<No alarm group>		

WheelAlignAlarmWord

WhlAlignZeroSclr

Name	WhlAlignZeroSclr	**ID**	26
Alarm class	Warnings	**Alarm text**	Zero of steering orientation scaler is underway
Alarm group	<No alarm group>		

WheelAlignAlarmWord

HMI alarms

Alarm groups

Alarm_group_1

Name	Alarm_group_1	**ID**	1

Alarm_group_10

Name	Alarm_group_10	**ID**	10

Alarm_group_11

Name	Alarm_group_11	**ID**	11

Alarm_group_12

Name	Alarm_group_12	**ID**	12

Alarm_group_13

Name	Alarm_group_13	**ID**	13

Alarm_group_14

Name	Alarm_group_14	**ID**	14

Alarm_group_15

Name	Alarm_group_15	**ID**	15

Alarm_group_16

Name	Alarm_group_16	**ID**	16

Alarm_group_2

Name	Alarm_group_2	**ID**	2

Alarm_group_3

Name	Alarm_group_3	**ID**	3

Alarm_group_4

Name	Alarm_group_4	**ID**	4

Alarm_group_5

Name	Alarm_group_5	**ID**	5

Alarm_group_6

Name	Alarm_group_6	**ID**	6

Alarm_group_7

Name	Alarm_group_7	**ID**	7

Alarm_group_8

Name	Alarm_group_8	**ID**	8

Alarm_group_9

Name	Alarm_group_9	**ID**	9

HMI alarms

Alarm classes

Acknowledgement

Name	Acknowledgement	**Display name**	A
ID	33	**Alarm log**	<No log>

Diagnosis events

Name	Diagnosis events	**Display name**	S7
ID	4	**Alarm log**	<No log>

Errors

Name	Errors	**Display name**	!
ID	1	**Alarm log**	<No log>

No Acknowledgement

Name	No Acknowledgement	**Display name**	NA
ID	34	**Alarm log**	<No log>

System

Name	System	**Display name**	$
ID	3	**Alarm log**	<No log>

Warnings

Name	Warnings	**Display name**	
ID	2	**Alarm log**	<No log>

V.6 Scripts

Code can be found on next page

Scripts / VB scripts

ShutdownIPC

Comment	

```


Sub ShutdownIPC()
'Tip:
' 1. Use the <CTRL+SPACE> or <CTRL+I> shortcut to open a list of all objects and functions
' 2. Write the code using the HMI Runtime object.
' Example: HmiRuntime.Screens("Screen_1").
' 3. Use the <CTRL+J> shortcut to create an object reference.
'Write the code as of this position:

Dim ShutdownIPCtmp 'create a local variable to mirror the DB item

ShutdownIPCtmp = SmartTags("ShutdownIPC_Trigger") 'write the DB item state to the local variable

If ShutdownIPCtmp = True Then
  'If the shutdown signal to the IPC is true shutdown the IPC via the windows command prompt
  'A sign of life bit will be used by the PLC to determine when the IPC has shut down so that it too can shutdown
  StartProgram "C:\AlexBatchFiles\IPCPowerOff.bat", "", hmiShowNormal, hmiNo

  'Send shutdown command to cmd via the "StartProgram" command
  ShowSystemAlarm "IPC shutdown command sent"


End If

End Sub
```

Scripts / VB scripts

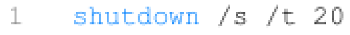
ShutdownRouter

Comment	

```


Sub ShutdownRouter()
'Tip:
' 1. Use the <CTRL+SPACE> or <CTRL+I> shortcut to open a list of all objects and functions
' 2. Write the code using the HMI Runtime object.
' Example: HmiRuntime.Screens("Screen_1").
' 3. Use the <CTRL+J> shortcut to create an object reference.
'Write the code as of this position:

'ShowSystemAlarm "Router shutdown command sent"

Dim ShutdownCMDtemp 'declare a local variable that will mirror the db item

ShutdownCMDtemp = SmartTags("ShutdownRouter_Trigger") 'write the state of the DB item to the temp variable

If ShutdownCMDtemp = True Then
  'send the poweroff command from the IPC via SSH to the router to shut it down using a .bat file

  'run "StartProgram" function here to run .bat file
  StartProgram "C:\AlexBatchFiles\RouterPowerOff.vbs", "", hmiShowNormal, hmiNo

  'echo that the shutdown command was given to the router
  ShowSystemAlarm "Router shutdown command sent"

End If

End Sub
```

W Appendix - Windows 7 Scripts

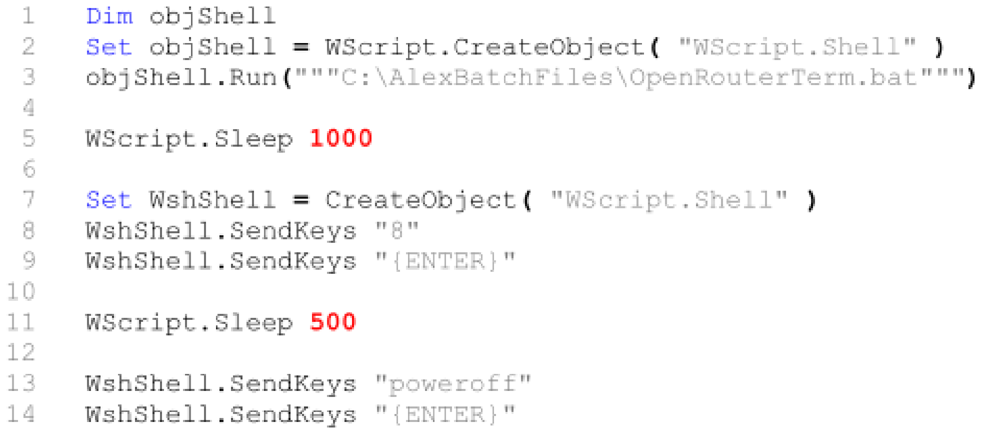
W.1 IPC Shutdown Code



```
shutdown /s /t 20
```

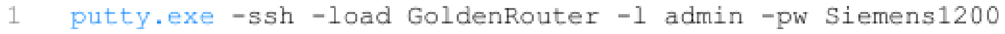
W.2 PFsense Router Shutdown Code

W.2.1 VBSscript



```
Dim objShell
Set objShell = WScript.CreateObject( "WScript.Shell" )
objShell.Run("""C:\AlexBatchFiles\OpenRouterTerm.bat""")

WScript.Sleep 1000

Set WshShell = CreateObject( "WScript.Shell" )
WshShell.SendKeys "8"
WshShell.SendKeys "{ENTER}"

WScript.Sleep 500

WshShell.SendKeys "poweroff"
WshShell.SendKeys "{ENTER}"
```

W.2.2 Batch File



```
putty.exe -ssh -load GoldenRouter -l admin -pw Siemens1200
```

X Appendix - Festo Stepper Drive Config

Project description

Project name:	Gertrude Steering Motor Config Bckup
Title:	
Creation date:	3/8/2021
Author:	alexm
Version:	V1.0.0
Last modification:	12/16/2021

Description

Componentlist

Componentname	Componentvendor	Componentfamily
Stepper A	Festo	CMMS-ST
Stepper B	Festo	CMMS-ST

Device Description

Device name:	Stepper A
Device family:	CMMS-ST
Vendor:	Festo
Plugin version:	V2.7.0

Configuration

Controller Type:	CMMS-ST-C8-7-G2
Option Slot:	CAMC-PB: PROFIBUS DP
Load Voltage:	48V

Motor Type:	EMMS-ST-57-M-SEB-G2
Gearbox:	5:1
Brake	Yes

Axis Type:	User Defined Rotative Axis (unlimited)
External Gearbox:	38:5
Parallel Mount:	No
Mechanical Structure:	Single Axis
Mounting kit:	Not Present

Application Data

Operating Mode Settings

Control Interface: PROFIBUS DP

Used Operating Modes

Profile Position Mode	Active
Homing Mode	Active
Profile Velocity Mode	Active
Profile Torque Mode	Inactive

Used Functions

X10 active	Inactive
Flying Measure	Inactive

Environment

Inverse Rotation Polarity	Inactive	
Mass moment of inertia:	2.500	kgcm²

Messages

Message "Target Reached"

Message Window: +/-	0.100	r
Window Time:	100	ms

Message "Following Error"

Message Window: +/-	0.500	r
Message Delay:	100	ms

Message "Velocity reached"

Declared Velocity:	0.000	rpm
Message Window: +/-	0	ms

Message "Remaining Distance"

Message Window:	0.000	r

Motor

Basic Parameter

Rated Current:	4.95	A
Boost Current:	5.00	A
Thermic Current:	4.95	A
Hold Current:	0.99	A
I²t Time Motor:	1000	ms
Undervoltage Level:	38.4	V
Brake Chopper Threshold:	58V	

Resonance Filter

	Velocity [rpm]	Range [rpm]
1	0.000	0.000
2	0.000	0.000
3	0.000	0.000

Brake Control

Lock Delay:	150	ms
Unlock Delay:	150	ms

Axis

Switch Types

Type Of Limit Switch:	NC - Normally Closed

General Limitations

Velocity:	40.842	rpm
Acceleration	7036.324	rpm/s

Stop Decelerations

Quick Stop:	7036.324	rpm/s
Monitoring time Quick Stop:	500	ms
Stop Input Signal:	7036.324	rpm/s
Limit switch:	7036.324	rpm/s

Homing

Homing Method

Method Description:	35: Actual position

Parameters

	Velocity [rpm]	Acceleration [rpm/s]
Search:	0.789	145.999
Crawl:	0.395	145.999
Running:	1.711	145.999
Axis Zero Point:	0.000 r	

Options

Go to the axis zero point after homing	No
Homing at controller enable	No

Measure

Axis Zero Point:	0.000	r
Project Zero Point	0.000	r

Controller

Closed Loop

Current Control

Gain:	1.67	
Time Constant:	2.18	ms

Velocity Control

Gain:	0.31	
Time Constant:	12.21	ms
Actual Velocity Filter:	2.00	ms

Position Control

Gain:	0.29	
Max. Correction Velocity:	13.158	rpm
Dead Range:	0.000	r

Application Data

Mass moment of inertia:	2.500	kgcm²
Inertia Ratio:	1.0	

Control Interface

Setpoint selection

Velocity Control

Setpoint:	Fieldbus

Digital I/O

Mode Selection over DIN9 and DIN12	Inactive

Digital Outputs

DOUT1:	Motion Complete

DOUT2:	Acknowledge Start
DOUT3:	Error

Offline View of the mode dependent I/O Configuration

Offline Mode:	Single position set

Analogue I/O

Analogue Output

Analogue Monitor:	Position Actual Value	
Scaling:	0.000	r
Offset:	5.0	V
Numeric Overflow Limitation	Inactive	

Direct Mode

Base value of velocity:	25.526	rpm
Acceleration:	281.400	rpm/s
Deceleration:	281.400	rpm/s
Smoothing:	0	%

Jog Mode

Crawling

Crawling Velocity:	0.395	rpm
Slow Moving Time:	2000	ms

Jog Parameters

Max. Velocity:	1.600	rpm
Acceleration:	281.400	rpm/s
Deceleration:	281.400	rpm/s
Smoothing:	100	%

Time To Ignore DINs After Teach

Ignore time:	100	ms

Position Set Table

Position Profiles

No.	Vel. [rpm]	Accel. [rpm/s]	Decel. [rpm/s]	Smooth [%]	Time [ms]	Start D. [ms]	Fin.Vel. [rpm]	Startcond.	Comment
0	1.605	281.400	281.400	0	0	0	0.000	Ignore	
1	1.605	281.400	281.400	0	0	0	0.000	Ignore	
2	1.605	281.400	281.400	0	0	0	0.000	Ignore	
3	1.605	281.400	281.400	0	0	0	0.000	Ignore	
4	1.605	281.400	281.400	0	0	0	0.000	Ignore	
5	1.605	281.400	281.400	0	0	0	0.000	Ignore	
6	1.605	281.400	281.400	0	0	0	0.000	Ignore	
7	1.605	281.400	281.400	0	0	0	0.000	Ignore	

Error Management

No	Error Text	Reaction
20	DC Bus undervoltage	Stop immediate with error
31	Overheating error (Motor)	Ignore
40	Overtemperature power stage	Stop immediate with error
122	CAN communication	Stop immediate with error
170	Following error	Stop immediate with error
180	Motor temperature 5°C below maximum	Ignore
181	Output stage temperature 5°C below maximum	Stop immediate with error
190	I^2t at 80%	Show warning
220	PROFIBUS init error	Stop immediate with error
290	No SD available	Show warning
291	SD initialisation	Stop immediate with error
292	SD parameter set	Stop immediate with error
310	I^2t-error motor (I^2t at 100%)	Show warning
311	I^2t-error power stage (I^2t at 100%)	Stop immediate with error
400	Softwarelimit negative	Show warning
401	Softwarelimit positive	Show warning
402	Target position behind softwarelimit negative	Show warning
403	Target position behind softwarelimit positive	Show warning
418	Record sequence: Unknown command	Stop immediate with error
419	Record sequence: Invalid branch destination	Stop immediate with error
421	Position precomputation	Stop immediate with error
424	Please enforce homing run!	Show warning
430	Limit switch negative	Stop controlled with error
431	Limit switch positive	Stop controlled with error
439	Both limit switches on	Stop controlled with error
650	DeviceNet assembly	Stop immediate with error
651	DeviceNet initialisation	Stop immediate with error
703	Operating mode	Stop immediate with error
790	RS232 communication error	Stop immediate with error

Device Description

Device name:	Stepper B
Device family:	CMMS-ST
Vendor:	Festo
Plugin version:	V2.7.0

Configuration

Controller Type:	CMMS-ST-C8-7-G2
Option Slot:	CAMC-PB: PROFIBUS DP
Load Voltage:	48V

Motor Type:	EMMS-ST-57-M-SEB-G2
Gearbox:	5:1
Brake	Yes

Axis Type:	User Defined Rotative Axis (unlimited)
External Gearbox:	38:5
Parallel Mount:	No
Mechanical Structure:	Single Axis
Mounting kit:	Not Present

Application Data

Operating Mode Settings

Control Interface: PROFIBUS DP

Used Operating Modes

Profile Position Mode	Active
Homing Mode	Active
Profile Velocity Mode	Active
Profile Torque Mode	Inactive

Used Functions

X10 active	Inactive
Flying Measure	Inactive

Environment

Inverse Rotation Polarity	Inactive	
Mass moment of inertia:	2.500	kgcm²

Messages

Message "Target Reached"

Message Window: +/-	0.100	r
Window Time:	100	ms

Message "Following Error"

Message Window: +/-	0.500	r
Message Delay:	100	ms

Message "Velocity reached"

Declared Velocity:	0.000	rpm
Message Window: +/-	0	ms

Message "Remaining Distance"

Message Window:	0.000	r

Motor

Basic Parameter

Rated Current:	4.95	A
Boost Current:	5.00	A
Thermic Current:	4.95	A
Hold Current:	0.99	A
I²t Time Motor:	1000	ms
Undervoltage Level:	38.4	V
Brake Chopper Threshold:	58V	

Resonance Filter

	Velocity [rpm]	Range [rpm]
1	0.000	0.000
2	0.000	0.000
3	0.000	0.000

Brake Control

Lock Delay:	150	ms
Unlock Delay:	150	ms

Axis

Switch Types

Type Of Limit Switch:	NC - Normally Closed

General Limitations

Velocity:	40.842	rpm
Acceleration	7036.324	rpm/s

Stop Decelerations

Quick Stop:	7036.324	rpm/s
Monitoring time Quick Stop:	500	ms
Stop Input Signal:	7036.324	rpm/s
Limit switch:	7036.324	rpm/s

Homing

Homing Method

Method Description:	35: Actual position

Parameters

	Velocity [rpm]	Acceleration [rpm/s]
Search:	0.789	145.999
Crawl:	0.395	145.999
Running:	1.711	145.999

Axis Zero Point:	0.000	r

Options

Go to the axis zero point after homing	No
Homing at controller enable	No

Measure

Axis Zero Point:	0.000	r
Project Zero Point	0.000	r

Controller

Closed Loop

Current Control

Gain:	1.67	
Time Constant:	2.18	ms

Velocity Control

Gain:	0.31	
Time Constant:	12.21	ms
Actual Velocity Filter:	2.00	ms

Position Control

Gain:	0.29	
Max. Correction Velocity:	13.158	rpm
Dead Range:	0.000	r

Application Data

Mass moment of inertia:	2.500	kgcm²
Inertia Ratio:	1.0	

Control Interface

Setpoint selection

Velocity Control

Setpoint:	Fieldbus

Digital I/O

Mode Selection over DIN9 and DIN12	Inactive

Digital Outputs

DOUT1:	Motion Complete

DOUT2:	Acknowledge Start
DOUT3:	Error

Offline View of the mode dependent I/O Configuration

Offline Mode:	Single position set

Analogue I/O

Analogue Output

Analogue Monitor:	Position Actual Value	
Scaling:	0.000	r
Offset:	5.0	V
Numeric Overflow Limitation	Inactive	

Direct Mode

Base value of velocity:	25.526	rpm
Acceleration:	281.400	rpm/s
Deceleration:	281.400	rpm/s
Smoothing:	0	%

Jog Mode

Crawling

Crawling Velocity:	0.395	rpm
Slow Moving Time:	2000	ms

Jog Parameters

Max. Velocity:	1.600	rpm
Acceleration:	281.400	rpm/s
Deceleration:	281.400	rpm/s
Smoothing:	100	%

Time To Ignore DINs After Teach

Ignore time:	100	ms

Position Set Table

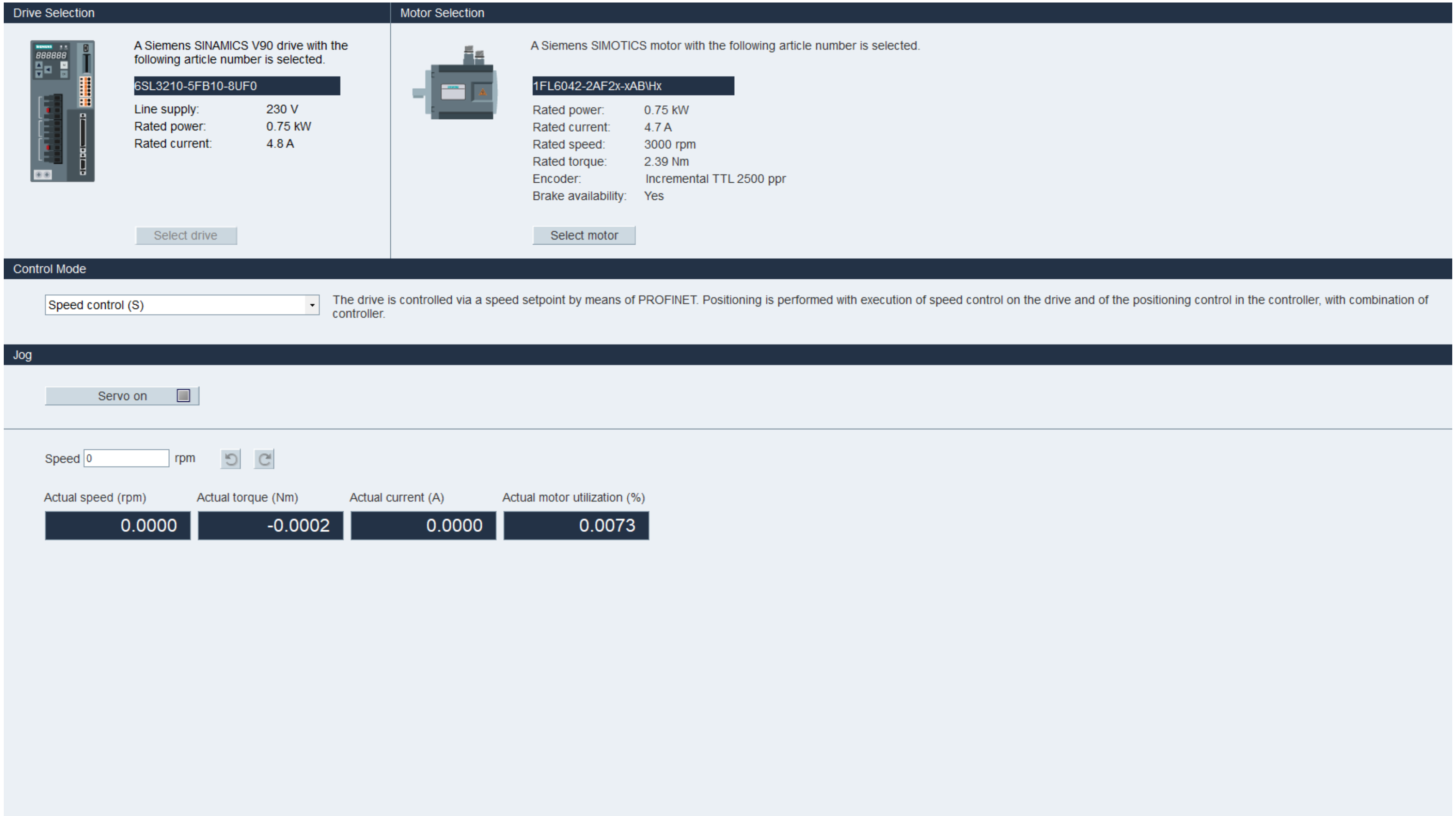
Position Profiles

No.	Vel. [rpm]	Accel. [rpm/s]	Decel. [rpm/s]	Smooth [%]	Time [ms]	Start D. [ms]	Fin.Vel. [rpm]	Startcond.	Comment
0	1.605	281.400	281.400	0	0	0	0.000	Ignore	
1	1.605	281.400	281.400	0	0	0	0.000	Ignore	
2	1.605	281.400	281.400	0	0	0	0.000	Ignore	
3	1.605	281.400	281.400	0	0	0	0.000	Ignore	
4	1.605	281.400	281.400	0	0	0	0.000	Ignore	
5	1.605	281.400	281.400	0	0	0	0.000	Ignore	
6	1.605	281.400	281.400	0	0	0	0.000	Ignore	
7	1.605	281.400	281.400	0	0	0	0.000	Ignore	

Error Management

No	Error Text	Reaction
20	DC Bus undervoltage	Stop immediate with error
31	Overheating error (Motor)	Ignore
40	Overtemperature power stage	Stop immediate with error
122	CAN communication	Stop immediate with error
170	Following error	Stop immediate with error
180	Motor temperature 5°C below maximum	Ignore
181	Output stage temperature 5°C below maximum	Stop immediate with error
190	I^2t at 80%	Show warning
220	PROFIBUS init error	Stop immediate with error
290	No SD available	Show warning
291	SD initialisation	Stop immediate with error
292	SD parameter set	Stop immediate with error
310	I^2t-error motor (I^2t at 100%)	Show warning
311	I^2t-error power stage (I^2t at 100%)	Stop immediate with error
400	Softwarelimit negative	Show warning
401	Softwarelimit positive	Show warning
402	Target position behind softwarelimit negative	Show warning
403	Target position behind softwarelimit positive	Show warning
418	Record sequence: Unknown command	Stop immediate with error
419	Record sequence: Invalid branch destination	Stop immediate with error
421	Position precomputation	Stop immediate with error
424	Please enforce homing run!	Show warning
430	Limit switch negative	Stop controlled with error
431	Limit switch positive	Stop controlled with error
439	Both limit switches on	Stop controlled with error
650	DeviceNet assembly	Stop immediate with error
651	DeviceNet initialisation	Stop immediate with error
703	Operating mode	Stop immediate with error
790	RS232 communication error	Stop immediate with error

Y Appendix - Siemens V90 Drive Config



Drive Selection

A Siemens SINAMICS V90 drive with the following article number is selected.

6SL3210-5FB10-8UF0

Line supply:	230 V
Rated power:	0.75 kW
Rated current:	4.8 A

Motor Selection

A Siemens SIMOTICS motor with the following article number is selected.

1FL6042-2AF2x-xAB\Hx

Rated power:	0.75 kW
Rated current:	4.7 A
Rated speed:	3000 rpm
Rated torque:	2.39 Nm
Encoder:	Incremental TTL 2500 ppr
Brake availability:	Yes

Control Mode

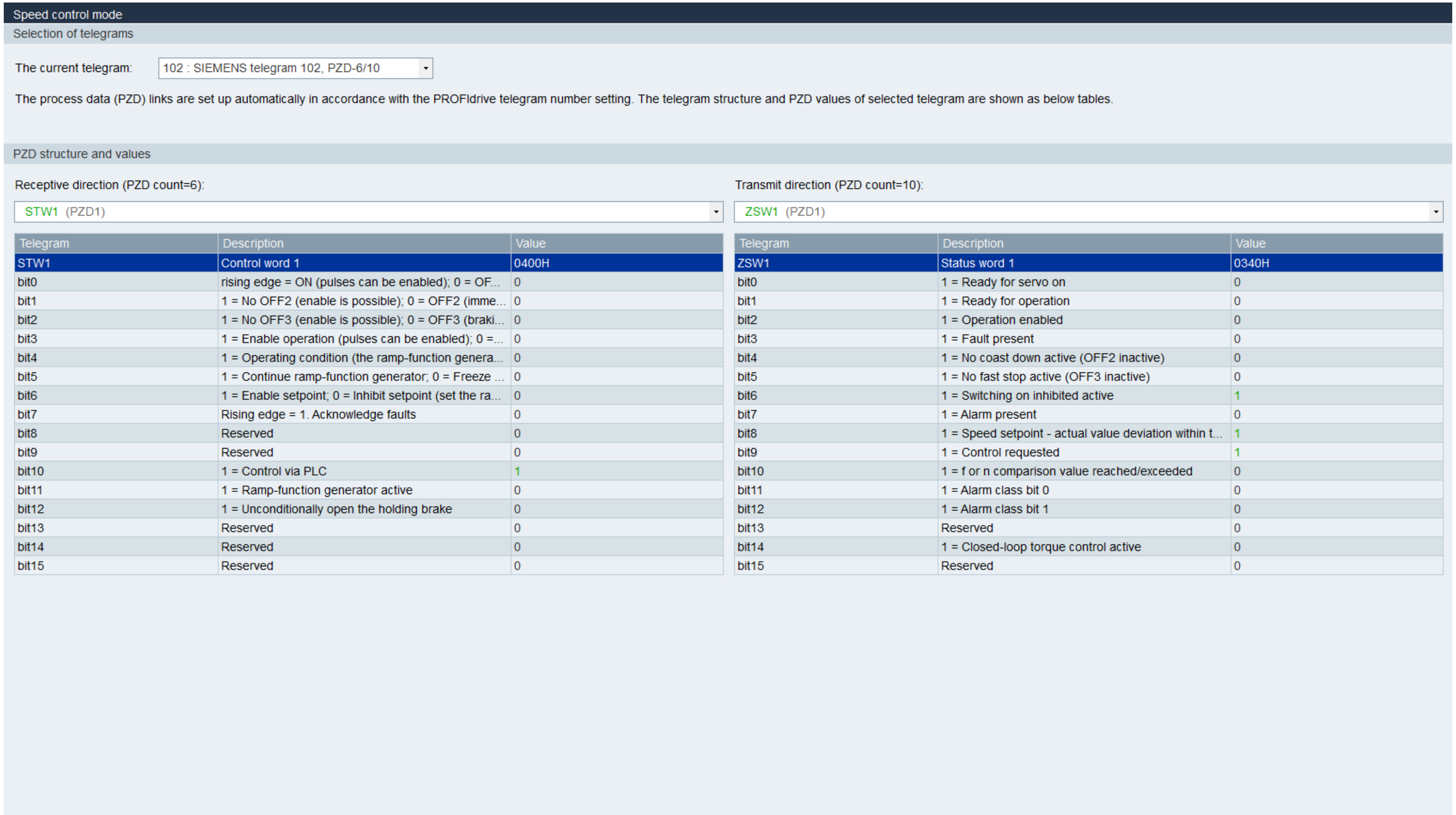
Speed control (S)

The drive is controlled via a speed setpoint by means of PROFINET. Positioning is performed with execution of speed control on the drive and of the positioning control in the controller, with combination of controller.

Jog

Speed 0 rpm

Actual speed (rpm)	Actual torque (Nm)	Actual current (A)	Actual motor utilization (%)
0.0000	-0.0002	0.0000	0.0073



Speed control mode

Selection of telegrams

The current telegram: 102 : SIEMENS telegram 102, PZD-6/10

The process data (PZD) links are set up automatically in accordance with the PROFIdrive telegram number setting. The telegram structure and PZD values of selected telegram are shown as below tables.

PZD structure and values

Receptive direction (PZD count=6):

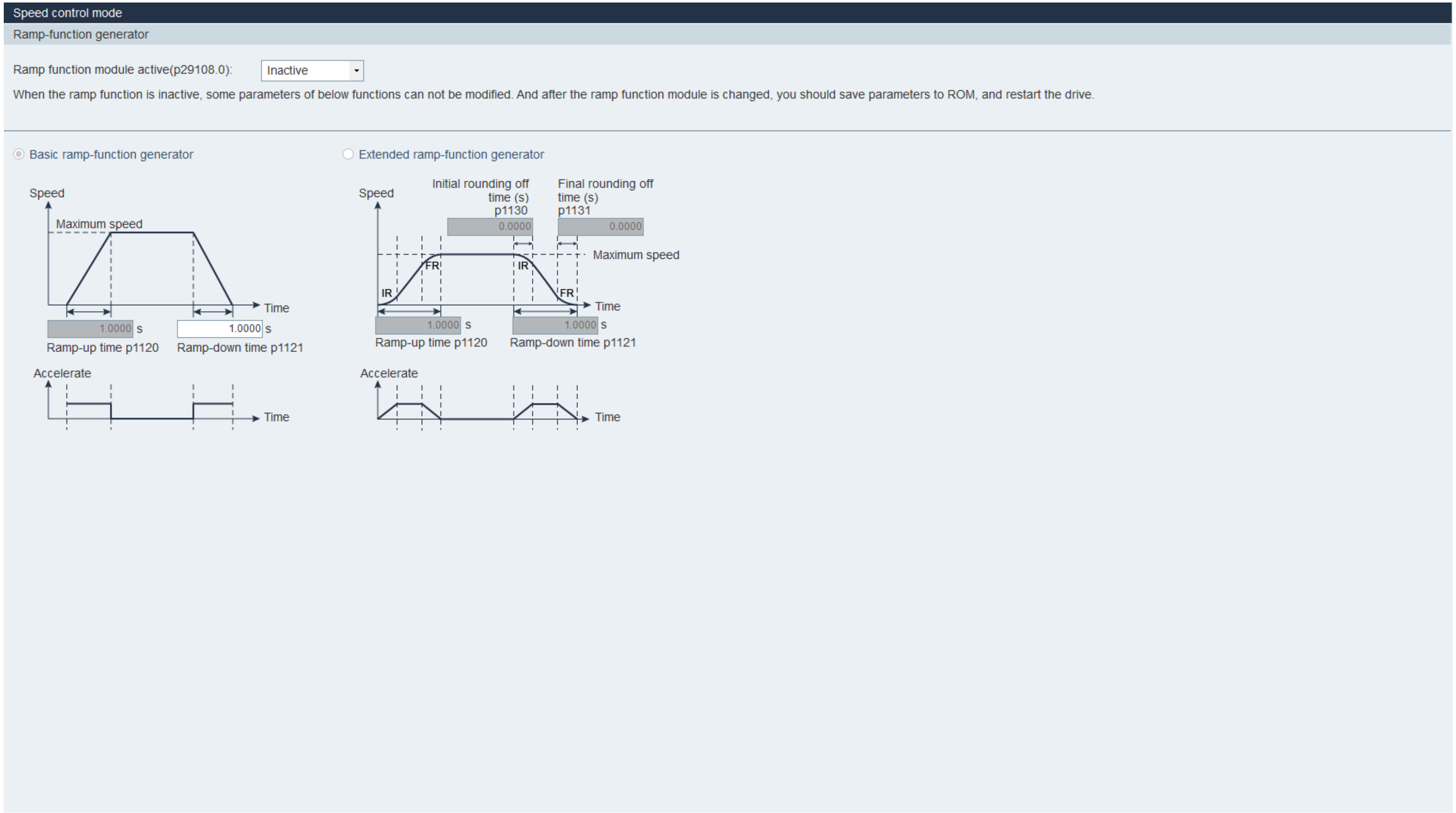
STW1 (PZD1)

Telegram	Description	Value
STW1	Control word 1	0400H
bit0	rising edge = ON (pulses can be enabled); 0 = OF...	0
bit1	1 = No OFF2 (enable is possible); 0 = OFF2 (imme...	0
bit2	1 = No OFF3 (enable is possible); 0 = OFF3 (braki...	0
bit3	1 = Enable operation (pulses can be enabled); 0 =...	0
bit4	1 = Operating condition (the ramp-function genera...	0
bit5	1 = Continue ramp-function generator; 0 = Freeze ...	0
bit6	1 = Enable setpoint; 0 = Inhibit setpoint (set the ra...	0
bit7	Rising edge = 1. Acknowledge faults	0
bit8	Reserved	0
bit9	Reserved	0
bit10	1 = Control via PLC	1
bit11	1 = Ramp-function generator active	0
bit12	1 = Unconditionally open the holding brake	0
bit13	Reserved	0
bit14	Reserved	0
bit15	Reserved	0

Transmit direction (PZD count=10):

ZSW1 (PZD1)

Telegram	Description	Value
ZSW1	Status word 1	0340H
bit0	1 = Ready for servo on	0
bit1	1 = Ready for operation	0
bit2	1 = Operation enabled	0
bit3	1 = Fault present	0
bit4	1 = No coast down active (OFF2 inactive)	0
bit5	1 = No fast stop active (OFF3 inactive)	0
bit6	1 = Switching on inhibited active	1
bit7	1 = Alarm present	0
bit8	1 = Speed setpoint - actual value deviation within t...	1
bit9	1 = Control requested	1
bit10	1 = f or n comparison value reached/exceeded	0
bit11	1 = Alarm class bit 0	0
bit12	1 = Alarm class bit 1	0
bit13	Reserved	0
bit14	1 = Closed-loop torque control active	0
bit15	Reserved	0



Speed control mode

Ramp-function generator

Ramp function module active(p29108.0): Inactive

When the ramp function is inactive, some parameters of below functions can not be modified. And after the ramp function module is changed, you should save parameters to ROM, and restart the drive.

● Basic ramp-function generator

Ramp-up time p1120: 1.0000 s

Ramp-down time p1121: 1.0000 s

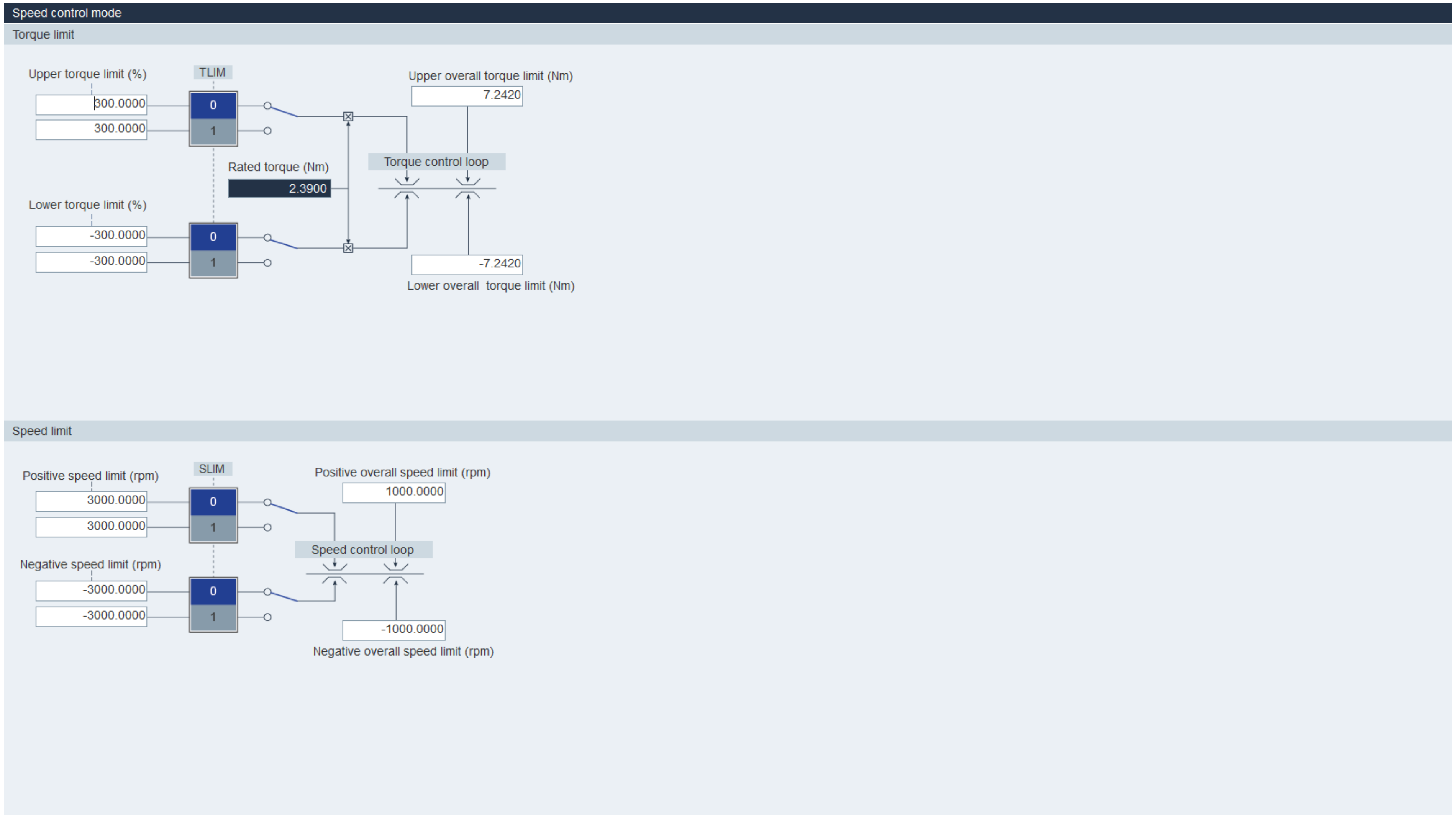
○ Extended ramp-function generator

Initial rounding off time (s) p1130: 0.0000

Final rounding off time (s) p1131: 0.0000

Ramp-up time p1120: 1.0000 s

Ramp-down time p1121: 1.0000 s



Speed control mode

Torque limit

Upper torque limit (%)

300.0000

300.0000

TLIM

0

1

Rated torque (Nm)

2.3900

Lower torque limit (%)

-300.0000

-300.0000

0

1

Upper overall torque limit (Nm)

7.2420

Torque control loop

-7.2420

Lower overall torque limit (Nm)

Speed limit

Positive speed limit (rpm)

3000.0000

3000.0000

SLIM

0

1

Negative speed limit (rpm)

-3000.0000

-3000.0000

0

1

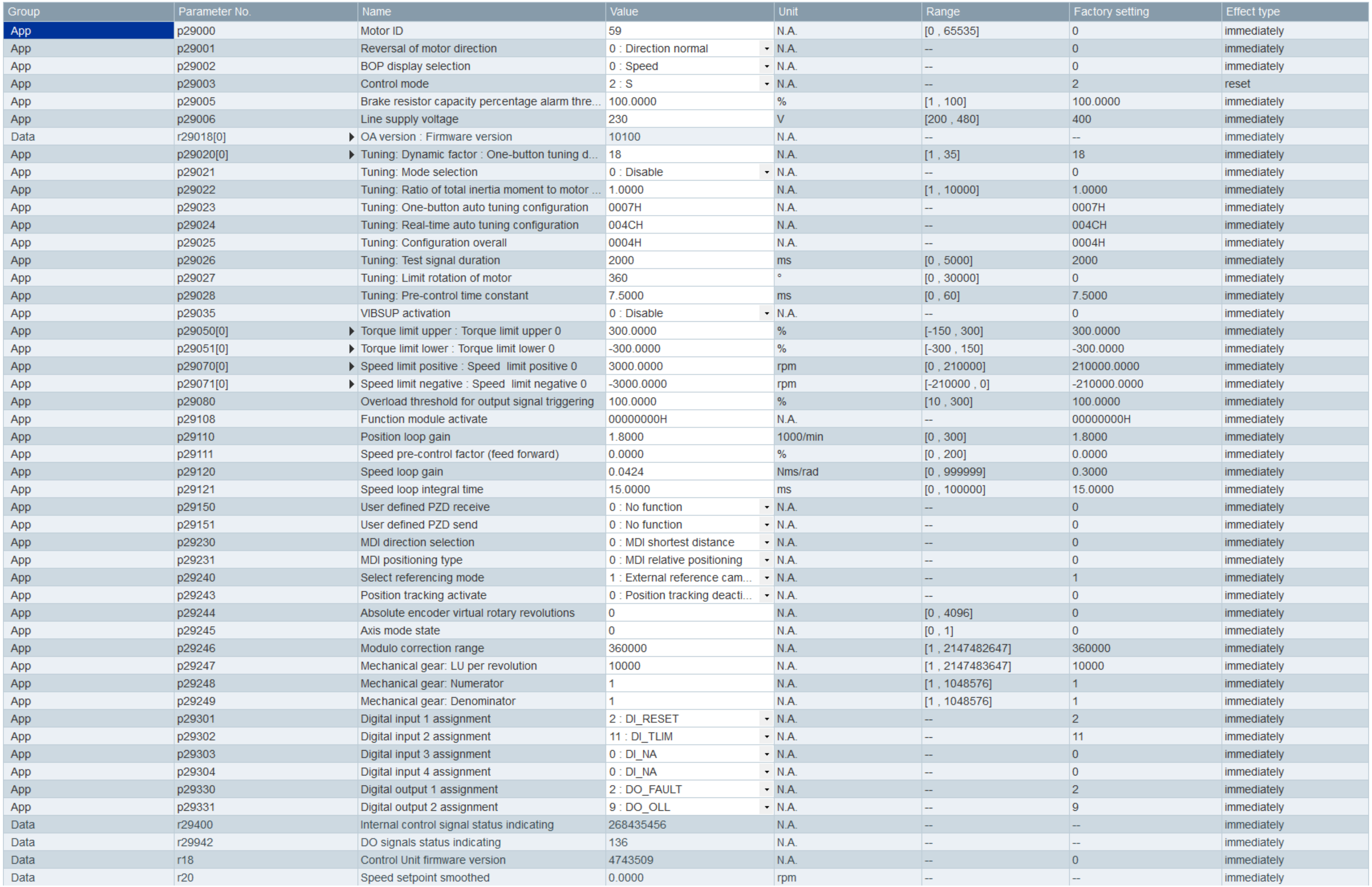
Positive overall speed limit (rpm)

1000.0000

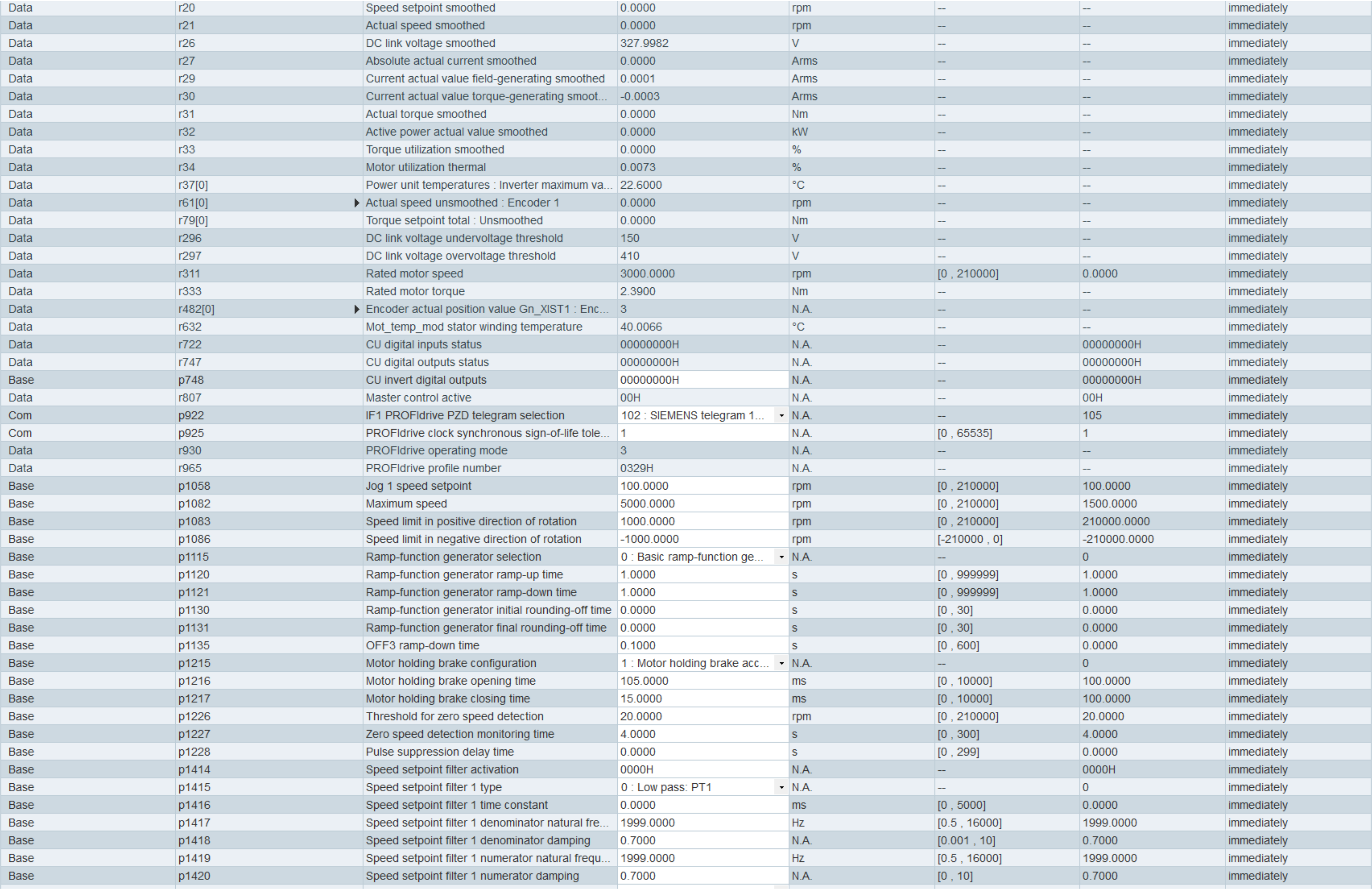
Speed control loop

-1000.0000

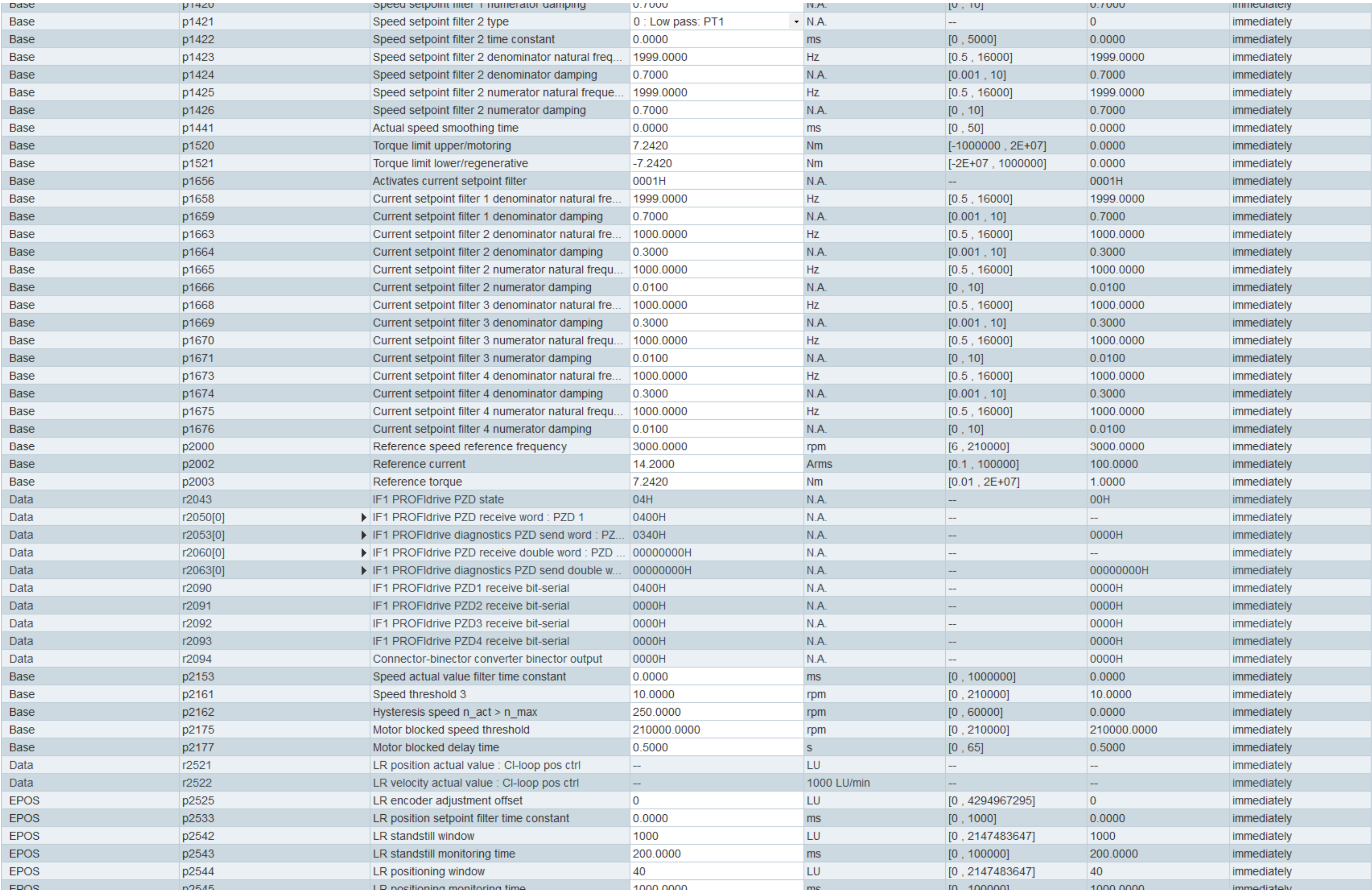
Negative overall speed limit (rpm)



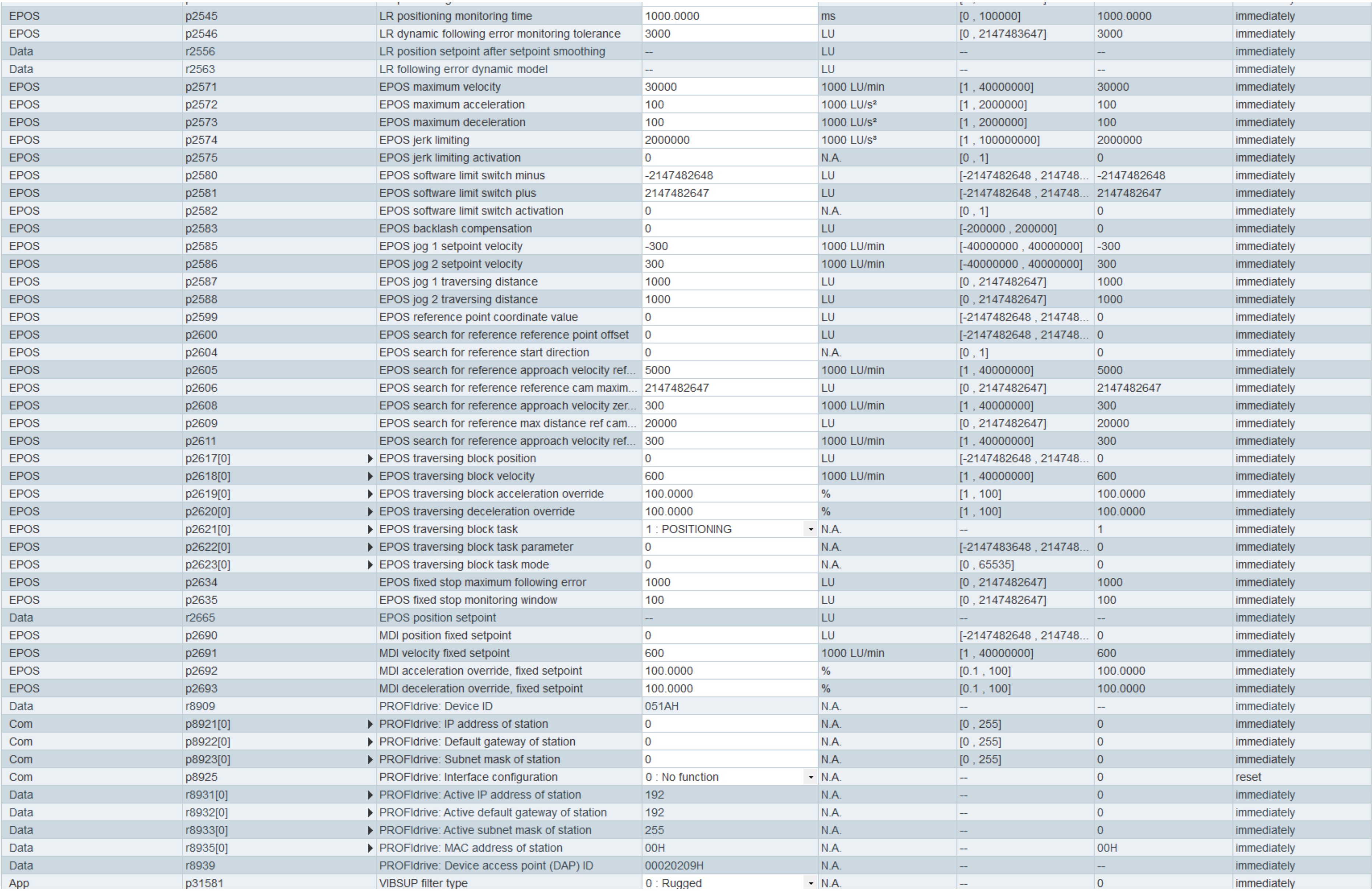
Group	Parameter No.	Name	Value	Unit	Range	Factory setting	Effect type
App	p29000	Motor ID	59	N.A.	[0 , 65535]	0	immediately
App	p29001	Reversal of motor direction	0 : Direction normal	N.A.	--	0	immediately
App	p29002	BOP display selection	0 : Speed	N.A.	--	0	immediately
App	p29003	Control mode	2 : S	N.A.	--	2	reset
App	p29005	Brake resistor capacity percentage alarm thre...	100.0000	%	[1 , 100]	100.0000	immediately
App	p29006	Line supply voltage	230	V	[200 , 480]	400	immediately
Data	r29018[0]	OA version : Firmware version	10100	N.A.	--	--	immediately
App	p29020[0]	Tuning: Dynamic factor : One-button tuning d...	18	N.A.	[1 , 35]	18	immediately
App	p29021	Tuning: Mode selection	0 : Disable	N.A.	--	0	immediately
App	p29022	Tuning: Ratio of total inertia moment to motor ...	1.0000	N.A.	[1 , 10000]	1.0000	immediately
App	p29023	Tuning: One-button auto tuning configuration	0007H	N.A.	--	0007H	immediately
App	p29024	Tuning: Real-time auto tuning configuration	004CH	N.A.	--	004CH	immediately
App	p29025	Tuning: Configuration overall	0004H	N.A.	--	0004H	immediately
App	p29026	Tuning: Test signal duration	2000	ms	[0 , 5000]	2000	immediately
App	p29027	Tuning: Limit rotation of motor	360	°	[0 , 30000]	0	immediately
App	p29028	Tuning: Pre-control time constant	7.5000	ms	[0 , 60]	7.5000	immediately
App	p29035	VIBSUP activation	0 : Disable	N.A.	--	0	immediately
App	p29050[0]	Torque limit upper : Torque limit upper 0	300.0000	%	[-150 , 300]	300.0000	immediately
App	p29051[0]	Torque limit lower : Torque limit lower 0	-300.0000	%	[-300 , 150]	-300.0000	immediately
App	p29070[0]	Speed limit positive : Speed limit positive 0	3000.0000	rpm	[0 , 210000]	210000.0000	immediately
App	p29071[0]	Speed limit negative : Speed limit negative 0	-3000.0000	rpm	[-210000 , 0]	-210000.0000	immediately
App	p29080	Overload threshold for output signal triggering	100.0000	%	[10 , 300]	100.0000	immediately
App	p29108	Function module activate	00000000H	N.A.	--	00000000H	immediately
App	p29110	Position loop gain	1.8000	1000/min	[0 , 300]	1.8000	immediately
App	p29111	Speed pre-control factor (feed forward)	0.0000	%	[0 , 200]	0.0000	immediately
App	p29120	Speed loop gain	0.0424	Nms/rad	[0 , 999999]	0.3000	immediately
App	p29121	Speed loop integral time	15.0000	ms	[0 , 100000]	15.0000	immediately
App	p29150	User defined PZD receive	0 : No function	N.A.	--	0	immediately
App	p29151	User defined PZD send	0 : No function	N.A.	--	0	immediately
App	p29230	MDI direction selection	0 : MDI shortest distance	N.A.	--	0	immediately
App	p29231	MDI positioning type	0 : MDI relative positioning	N.A.	--	0	immediately
App	p29240	Select referencing mode	1 : External reference cam...	N.A.	--	1	immediately
App	p29243	Position tracking activate	0 : Position tracking deacti...	N.A.	--	0	immediately
App	p29244	Absolute encoder virtual rotary revolutions	0	N.A.	[0 , 4096]	0	immediately
App	p29245	Axis mode state	0	N.A.	[0 , 1]	0	immediately
App	p29246	Modulo correction range	360000	N.A.	[1 , 2147482647]	360000	immediately
App	p29247	Mechanical gear: LU per revolution	10000	N.A.	[1 , 2147483647]	10000	immediately
App	p29248	Mechanical gear: Numerator	1	N.A.	[1 , 1048576]	1	immediately
App	p29249	Mechanical gear: Denominator	1	N.A.	[1 , 1048576]	1	immediately
App	p29301	Digital input 1 assignment	2 : DI_RESET	N.A.	--	2	immediately
App	p29302	Digital input 2 assignment	11 : DI_TLIM	N.A.	--	11	immediately
App	p29303	Digital input 3 assignment	0 : DI_NA	N.A.	--	0	immediately
App	p29304	Digital input 4 assignment	0 : DI_NA	N.A.	--	0	immediately
App	p29330	Digital output 1 assignment	2 : DO_FAULT	N.A.	--	2	immediately
App	p29331	Digital output 2 assignment	9 : DO_OLL	N.A.	--	9	immediately
Data	r29400	Internal control signal status indicating	268435456	N.A.	--	--	immediately
Data	r29942	DO signals status indicating	136	N.A.	--	--	immediately
Data	r18	Control Unit firmware version	4743509	N.A.	--	0	immediately
Data	r20	Speed setpoint smoothed	0.0000	rpm	--	--	immediately



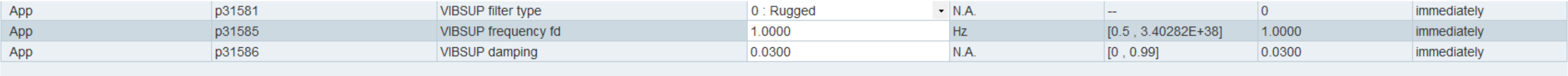
Data	r20	Speed setpoint smoothed	0.0000	rpm	--	--	immediately
Data	r21	Actual speed smoothed	0.0000	rpm	--	--	immediately
Data	r26	DC link voltage smoothed	327.9982	V	--	--	immediately
Data	r27	Absolute actual current smoothed	0.0000	Arms	--	--	immediately
Data	r29	Current actual value field-generating smoothed	0.0001	Arms	--	--	immediately
Data	r30	Current actual value torque-generating smoot...	-0.0003	Arms	--	--	immediately
Data	r31	Actual torque smoothed	0.0000	Nm	--	--	immediately
Data	r32	Active power actual value smoothed	0.0000	kW	--	--	immediately
Data	r33	Torque utilization smoothed	0.0000	%	--	--	immediately
Data	r34	Motor utilization thermal	0.0073	%	--	--	immediately
Data	r37[0]	Power unit temperatures : Inverter maximum va...	22.6000	°C	--	--	immediately
Data	r61[0]	▶ Actual speed unsmoothed : Encoder 1	0.0000	rpm	--	--	immediately
Data	r79[0]	Torque setpoint total : Unsmoothed	0.0000	Nm	--	--	immediately
Data	r296	DC link voltage undervoltage threshold	150	V	--	--	immediately
Data	r297	DC link voltage overvoltage threshold	410	V	--	--	immediately
Data	r311	Rated motor speed	3000.0000	rpm	[0 , 210000]	0.0000	immediately
Data	r333	Rated motor torque	2.3900	Nm	--	--	immediately
Data	r482[0]	▶ Encoder actual position value Gn_XIST1 : Enc...	3	N.A.	--	--	immediately
Data	r632	Mot_temp_mod stator winding temperature	40.0066	°C	--	--	immediately
Data	r722	CU digital inputs status	00000000H	N.A.	--	00000000H	immediately
Data	r747	CU digital outputs status	00000000H	N.A.	--	00000000H	immediately
Base	p748	CU invert digital outputs	00000000H	N.A.	--	00000000H	immediately
Data	r807	Master control active	00H	N.A.	--	00H	immediately
Com	p922	IF1 PROFIdrive PZD telegram selection	102 : SIEMENS telegram 1...	N.A.	--	105	immediately
Com	p925	PROFIdrive clock synchronous sign-of-life tole...	1	N.A.	[0 , 65535]	1	immediately
Data	r930	PROFIdrive operating mode	3	N.A.	--	--	immediately
Data	r965	PROFIdrive profile number	0329H	N.A.	--	--	immediately
Base	p1058	Jog 1 speed setpoint	100.0000	rpm	[0 , 210000]	100.0000	immediately
Base	p1082	Maximum speed	5000.0000	rpm	[0 , 210000]	1500.0000	immediately
Base	p1083	Speed limit in positive direction of rotation	1000.0000	rpm	[0 , 210000]	210000.0000	immediately
Base	p1086	Speed limit in negative direction of rotation	-1000.0000	rpm	[-210000 , 0]	-210000.0000	immediately
Base	p1115	Ramp-function generator selection	0 : Basic ramp-function ge...	N.A.	--	0	immediately
Base	p1120	Ramp-function generator ramp-up time	1.0000	s	[0 , 999999]	1.0000	immediately
Base	p1121	Ramp-function generator ramp-down time	1.0000	s	[0 , 999999]	1.0000	immediately
Base	p1130	Ramp-function generator initial rounding-off time	0.0000	s	[0 , 30]	0.0000	immediately
Base	p1131	Ramp-function generator final rounding-off time	0.0000	s	[0 , 30]	0.0000	immediately
Base	p1135	OFF3 ramp-down time	0.1000	s	[0 , 600]	0.0000	immediately
Base	p1215	Motor holding brake configuration	1 : Motor holding brake acc...	N.A.	--	0	immediately
Base	p1216	Motor holding brake opening time	105.0000	ms	[0 , 10000]	100.0000	immediately
Base	p1217	Motor holding brake closing time	15.0000	ms	[0 , 10000]	100.0000	immediately
Base	p1226	Threshold for zero speed detection	20.0000	rpm	[0 , 210000]	20.0000	immediately
Base	p1227	Zero speed detection monitoring time	4.0000	s	[0 , 300]	4.0000	immediately
Base	p1228	Pulse suppression delay time	0.0000	s	[0 , 299]	0.0000	immediately
Base	p1414	Speed setpoint filter activation	0000H	N.A.	--	0000H	immediately
Base	p1415	Speed setpoint filter 1 type	0 : Low pass: PT1	N.A.	--	0	immediately
Base	p1416	Speed setpoint filter 1 time constant	0.0000	ms	[0 , 5000]	0.0000	immediately
Base	p1417	Speed setpoint filter 1 denominator natural fre...	1999.0000	Hz	[0.5 , 16000]	1999.0000	immediately
Base	p1418	Speed setpoint filter 1 denominator damping	0.7000	N.A.	[0.001 , 10]	0.7000	immediately
Base	p1419	Speed setpoint filter 1 numerator natural frequ...	1999.0000	Hz	[0.5 , 16000]	1999.0000	immediately
Base	p1420	Speed setpoint filter 1 numerator damping	0.7000	N.A.	[0 , 10]	0.7000	immediately



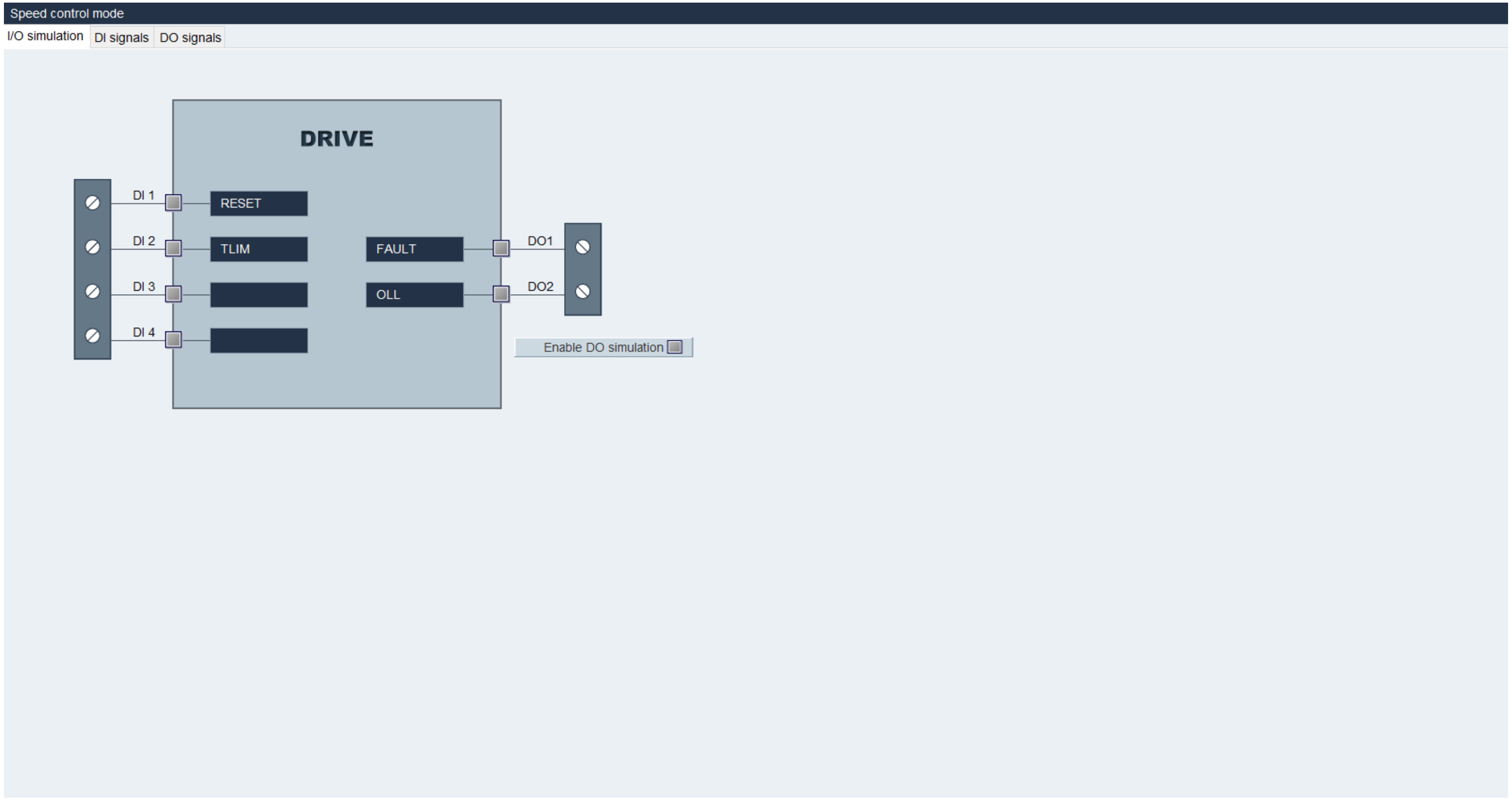
Base	p1420	Speed setpoint filter 1 numerator damping	0.7000	N.A.	[0 , 10]	0.7000	immediately
Base	p1421	Speed setpoint filter 2 type	0 : Low pass: PT1 ▾	N.A.	--	0	immediately
Base	p1422	Speed setpoint filter 2 time constant	0.0000	ms	[0 , 5000]	0.0000	immediately
Base	p1423	Speed setpoint filter 2 denominator natural freq...	1999.0000	Hz	[0.5 , 16000]	1999.0000	immediately
Base	p1424	Speed setpoint filter 2 denominator damping	0.7000	N.A.	[0.001 , 10]	0.7000	immediately
Base	p1425	Speed setpoint filter 2 numerator natural freque...	1999.0000	Hz	[0.5 , 16000]	1999.0000	immediately
Base	p1426	Speed setpoint filter 2 numerator damping	0.7000	N.A.	[0 , 10]	0.7000	immediately
Base	p1441	Actual speed smoothing time	0.0000	ms	[0 , 50]	0.0000	immediately
Base	p1520	Torque limit upper/motoring	7.2420	Nm	[-1000000 , 2E+07]	0.0000	immediately
Base	p1521	Torque limit lower/regenerative	-7.2420	Nm	[-2E+07 , 1000000]	0.0000	immediately
Base	p1656	Activates current setpoint filter	0001H	N.A.	--	0001H	immediately
Base	p1658	Current setpoint filter 1 denominator natural fre...	1999.0000	Hz	[0.5 , 16000]	1999.0000	immediately
Base	p1659	Current setpoint filter 1 denominator damping	0.7000	N.A.	[0.001 , 10]	0.7000	immediately
Base	p1663	Current setpoint filter 2 denominator natural fre...	1000.0000	Hz	[0.5 , 16000]	1000.0000	immediately
Base	p1664	Current setpoint filter 2 denominator damping	0.3000	N.A.	[0.001 , 10]	0.3000	immediately
Base	p1665	Current setpoint filter 2 numerator natural frequ...	1000.0000	Hz	[0.5 , 16000]	1000.0000	immediately
Base	p1666	Current setpoint filter 2 numerator damping	0.0100	N.A.	[0 , 10]	0.0100	immediately
Base	p1668	Current setpoint filter 3 denominator natural fre...	1000.0000	Hz	[0.5 , 16000]	1000.0000	immediately
Base	p1669	Current setpoint filter 3 denominator damping	0.3000	N.A.	[0.001 , 10]	0.3000	immediately
Base	p1670	Current setpoint filter 3 numerator natural frequ...	1000.0000	Hz	[0.5 , 16000]	1000.0000	immediately
Base	p1671	Current setpoint filter 3 numerator damping	0.0100	N.A.	[0 , 10]	0.0100	immediately
Base	p1673	Current setpoint filter 4 denominator natural fre...	1000.0000	Hz	[0.5 , 16000]	1000.0000	immediately
Base	p1674	Current setpoint filter 4 denominator damping	0.3000	N.A.	[0.001 , 10]	0.3000	immediately
Base	p1675	Current setpoint filter 4 numerator natural frequ...	1000.0000	Hz	[0.5 , 16000]	1000.0000	immediately
Base	p1676	Current setpoint filter 4 numerator damping	0.0100	N.A.	[0 , 10]	0.0100	immediately
Base	p2000	Reference speed reference frequency	3000.0000	rpm	[6 , 210000]	3000.0000	immediately
Base	p2002	Reference current	14.2000	Arms	[0.1 , 100000]	100.0000	immediately
Base	p2003	Reference torque	7.2420	Nm	[0.01 , 2E+07]	1.0000	immediately
Data	r2043	IF1 PROFIdrive PZD state	04H	N.A.	--	00H	immediately
Data	r2050[0]	▶ IF1 PROFIdrive PZD receive word : PZD 1	0400H	N.A.	--	--	immediately
Data	r2053[0]	▶ IF1 PROFIdrive diagnostics PZD send word : PZ...	0340H	N.A.	--	0000H	immediately
Data	r2060[0]	▶ IF1 PROFIdrive PZD receive double word : PZD ...	00000000H	N.A.	--	--	immediately
Data	r2063[0]	▶ IF1 PROFIdrive diagnostics PZD send double w...	00000000H	N.A.	--	00000000H	immediately
Data	r2090	IF1 PROFIdrive PZD1 receive bit-serial	0400H	N.A.	--	0000H	immediately
Data	r2091	IF1 PROFIdrive PZD2 receive bit-serial	0000H	N.A.	--	0000H	immediately
Data	r2092	IF1 PROFIdrive PZD3 receive bit-serial	0000H	N.A.	--	0000H	immediately
Data	r2093	IF1 PROFIdrive PZD4 receive bit-serial	0000H	N.A.	--	0000H	immediately
Data	r2094	Connector-binector converter binector output	0000H	N.A.	--	0000H	immediately
Base	p2153	Speed actual value filter time constant	0.0000	ms	[0 , 1000000]	0.0000	immediately
Base	p2161	Speed threshold 3	10.0000	rpm	[0 , 210000]	10.0000	immediately
Base	p2162	Hysteresis speed n_act > n_max	250.0000	rpm	[0 , 60000]	0.0000	immediately
Base	p2175	Motor blocked speed threshold	210000.0000	rpm	[0 , 210000]	210000.0000	immediately
Base	p2177	Motor blocked delay time	0.5000	s	[0 , 65]	0.5000	immediately
Data	r2521	LR position actual value : Cl-loop pos ctrl	--	LU	--	--	immediately
Data	r2522	LR velocity actual value : Cl-loop pos ctrl	--	1000 LU/min	--	--	immediately
EPOS	p2525	LR encoder adjustment offset	0	LU	[0 , 4294967295]	0	immediately
EPOS	p2533	LR position setpoint filter time constant	0.0000	ms	[0 , 1000]	0.0000	immediately
EPOS	p2542	LR standstill window	1000	LU	[0 , 2147483647]	1000	immediately
EPOS	p2543	LR standstill monitoring time	200.0000	ms	[0 , 100000]	200.0000	immediately
EPOS	p2544	LR positioning window	40	LU	[0 , 2147483647]	40	immediately
EPOS	p2545	LR positioning monitoring time	1000.0000	ms	[0 , 100000]	1000.0000	immediately

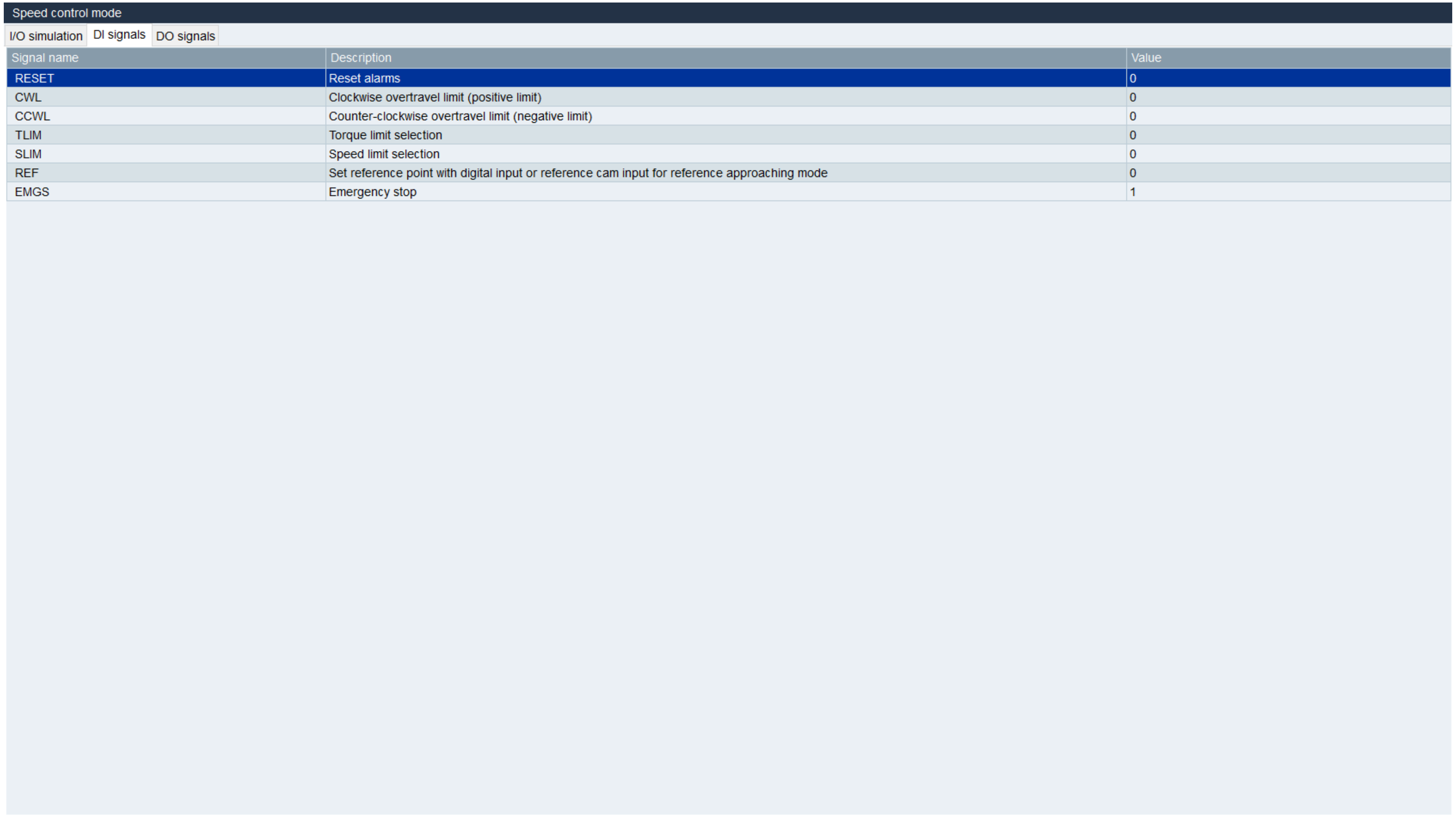


EPOS	p2545	LR positioning monitoring time	1000.0000	ms	[0 , 100000]	1000.0000	immediately
EPOS	p2546	LR dynamic following error monitoring tolerance	3000	LU	[0 , 2147483647]	3000	immediately
Data	r2556	LR position setpoint after setpoint smoothing	--	LU	--	--	immediately
Data	r2563	LR following error dynamic model	--	LU	--	--	immediately
EPOS	p2571	EPOS maximum velocity	30000	1000 LU/min	[1 , 40000000]	30000	immediately
EPOS	p2572	EPOS maximum acceleration	100	1000 LU/s²	[1 , 2000000]	100	immediately
EPOS	p2573	EPOS maximum deceleration	100	1000 LU/s²	[1 , 2000000]	100	immediately
EPOS	p2574	EPOS jerk limiting	2000000	1000 LU/s³	[1 , 100000000]	2000000	immediately
EPOS	p2575	EPOS jerk limiting activation	0	N.A.	[0 , 1]	0	immediately
EPOS	p2580	EPOS software limit switch minus	-2147482648	LU	[-2147482648 , 214748...	-2147482648	immediately
EPOS	p2581	EPOS software limit switch plus	2147482647	LU	[-2147482648 , 214748...	2147482647	immediately
EPOS	p2582	EPOS software limit switch activation	0	N.A.	[0 , 1]	0	immediately
EPOS	p2583	EPOS backlash compensation	0	LU	[-200000 , 200000]	0	immediately
EPOS	p2585	EPOS jog 1 setpoint velocity	-300	1000 LU/min	[-40000000 , 40000000]	-300	immediately
EPOS	p2586	EPOS jog 2 setpoint velocity	300	1000 LU/min	[-40000000 , 40000000]	300	immediately
EPOS	p2587	EPOS jog 1 traversing distance	1000	LU	[0 , 2147482647]	1000	immediately
EPOS	p2588	EPOS jog 2 traversing distance	1000	LU	[0 , 2147482647]	1000	immediately
EPOS	p2599	EPOS reference point coordinate value	0	LU	[-2147482648 , 214748...	0	immediately
EPOS	p2600	EPOS search for reference reference point offset	0	LU	[-2147482648 , 214748...	0	immediately
EPOS	p2604	EPOS search for reference start direction	0	N.A.	[0 , 1]	0	immediately
EPOS	p2605	EPOS search for reference approach velocity ref...	5000	1000 LU/min	[1 , 40000000]	5000	immediately
EPOS	p2606	EPOS search for reference reference cam maxim...	2147482647	LU	[0 , 2147482647]	2147482647	immediately
EPOS	p2608	EPOS search for reference approach velocity zer...	300	1000 LU/min	[1 , 40000000]	300	immediately
EPOS	p2609	EPOS search for reference max distance ref cam...	20000	LU	[0 , 2147482647]	20000	immediately
EPOS	p2611	EPOS search for reference approach velocity ref...	300	1000 LU/min	[1 , 40000000]	300	immediately
EPOS	p2617[0]	▶ EPOS traversing block position	0	LU	[-2147482648 , 214748...	0	immediately
EPOS	p2618[0]	▶ EPOS traversing block velocity	600	1000 LU/min	[1 , 40000000]	600	immediately
EPOS	p2619[0]	▶ EPOS traversing block acceleration override	100.0000	%	[1 , 100]	100.0000	immediately
EPOS	p2620[0]	▶ EPOS traversing deceleration override	100.0000	%	[1 , 100]	100.0000	immediately
EPOS	p2621[0]	▶ EPOS traversing block task	1 : POSITIONING ▾	N.A.	--	1	immediately
EPOS	p2622[0]	▶ EPOS traversing block task parameter	0	N.A.	[-2147483648 , 214748...	0	immediately
EPOS	p2623[0]	▶ EPOS traversing block task mode	0	N.A.	[0 , 65535]	0	immediately
EPOS	p2634	EPOS fixed stop maximum following error	1000	LU	[0 , 2147482647]	1000	immediately
EPOS	p2635	EPOS fixed stop monitoring window	100	LU	[0 , 2147482647]	100	immediately
Data	r2665	EPOS position setpoint	--	LU	--	--	immediately
EPOS	p2690	MDI position fixed setpoint	0	LU	[-2147482648 , 214748...	0	immediately
EPOS	p2691	MDI velocity fixed setpoint	600	1000 LU/min	[1 , 40000000]	600	immediately
EPOS	p2692	MDI acceleration override, fixed setpoint	100.0000	%	[0.1 , 100]	100.0000	immediately
EPOS	p2693	MDI deceleration override, fixed setpoint	100.0000	%	[0.1 , 100]	100.0000	immediately
Data	r8909	PROFIdrive: Device ID	051AH	N.A.	--	--	immediately
Com	p8921[0]	▶ PROFIdrive: IP address of station	0	N.A.	[0 , 255]	0	immediately
Com	p8922[0]	▶ PROFIdrive: Default gateway of station	0	N.A.	[0 , 255]	0	immediately
Com	p8923[0]	▶ PROFIdrive: Subnet mask of station	0	N.A.	[0 , 255]	0	immediately
Com	p8925	PROFIdrive: Interface configuration	0 : No function ▾	N.A.	--	0	reset
Data	r8931[0]	▶ PROFIdrive: Active IP address of station	192	N.A.	--	0	immediately
Data	r8932[0]	▶ PROFIdrive: Active default gateway of station	192	N.A.	--	0	immediately
Data	r8933[0]	▶ PROFIdrive: Active subnet mask of station	255	N.A.	--	0	immediately
Data	r8935[0]	▶ PROFIdrive: MAC address of station	00H	N.A.	--	00H	immediately
Data	r8939	PROFIdrive: Device access point (DAP) ID	00020209H	N.A.	--	--	immediately
App	p31581	VIBSUP filter type	0 : Rugged ▾	N.A.	--	0	immediately



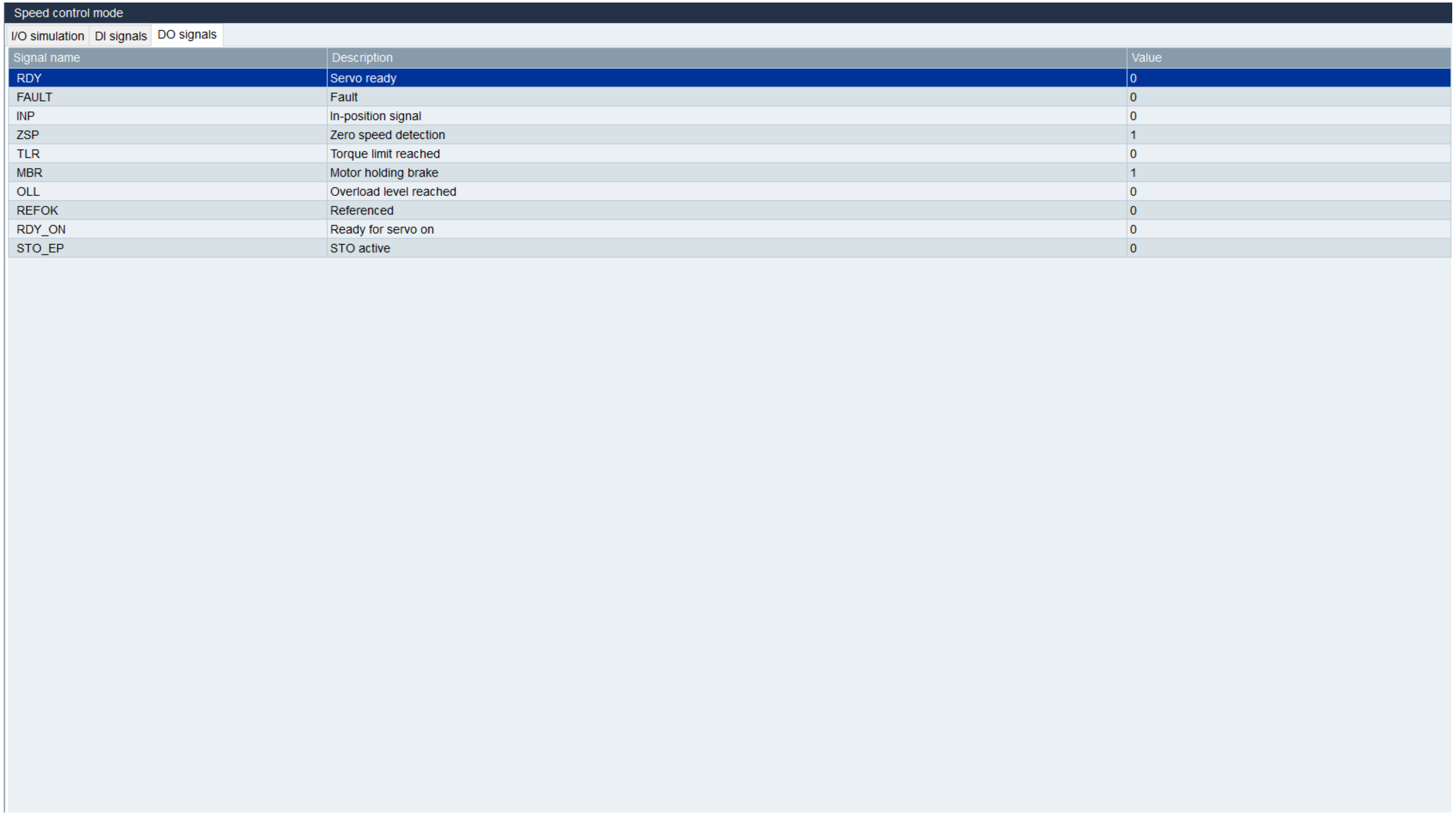
App	p31581	VIBSUP filter type	0 : Rugged	N.A.	--	0	immediately
App	p31585	VIBSUP frequency fd	1.0000	Hz	[0.5 , 3.40282E+38]	1.0000	immediately
App	p31586	VIBSUP damping	0.0300	N.A.	[0 , 0.99]	0.0300	immediately


Speed control mode
I/O simulation
DI signals
DO signals
DRIVE
DI 1
DI 2
DI 3
DI 4
RESET
TLIM
FAULT
OLL
DO1
DO2
Enable DO simulation



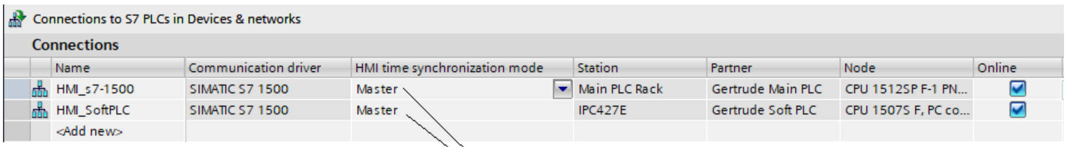
Speed control mode

Signal name	Description	Value
RESET	Reset alarms	0
CWL	Clockwise overtravel limit (positive limit)	0
CCWL	Counter-clockwise overtravel limit (negative limit)	0
TLIM	Torque limit selection	0
SLIM	Speed limit selection	0
REF	Set reference point with digital input or reference cam input for reference approaching mode	0
EMGS	Emergency stop	1



Signal name	Description	Value
RDY	Servo ready	0
FAULT	Fault	0
INP	In-position signal	0
ZSP	Zero speed detection	1
TLR	Torque limit reached	0
MBR	Motor holding brake	1
OLL	Overload level reached	0
REFOK	Referenced	0
RDY_ON	Ready for servo on	0
STO_EP	STO active	0

Z Appendix - Updated Time Sync



Connections to S7 PLCs in Devices & networks

Connections

Name	Communication driver	HMI time synchronization mode	Station	Partner	Node	Online
HMI_s7-1500	SIMATIC S7 1500	Master	Main PLC Rack	Gertrude Main PLC	CPU 1512SP F-1 PN...	☑
HMI_SoftPLC	SIMATIC S7 1500	Master	IPC427E	Gertrude Soft PLC	CPU 1507S F, PC co...	☑
<Add new>						

WinCC set as time master

AA Appendix - AGV Calibration

AA.1 Festo CMMS-ST Drive Analog Calibration

The Festo CMMS-ST stepper drives do not return an angle value via Profibus when the drives are in "velocity mode". The value is available on the drive itself, but Festo neglected to make it available through any network protocol due to poor implementation.

Since the current angle of the steering mechanism is needed for the kinematic calculations, this value needed to be obtained somehow. The first option was to deduce this value using numerical analysis based on the stepper drive's velocity. Velocity was one of the values that Festo decided to make available on the Profibus network, hence why this strategy was possible. The second was to convert it to an electrical signal and send it via an analogue output.

As it turned out, the integrated angle of the drive's velocity did not precisely match the real-world angle. The author suspected that the reason for this was that the speed values from the drives were being rounded off before being sent from the drives to the PLC via Profibus, or the time scaling used by the numerical method integrator was not as exact as it should have been. Either way, the integrated angle values diverged further from the real world angle the further the steering moved from zero; this divergence also appeared to be linear, see figure AA.1 and figure AA.2.

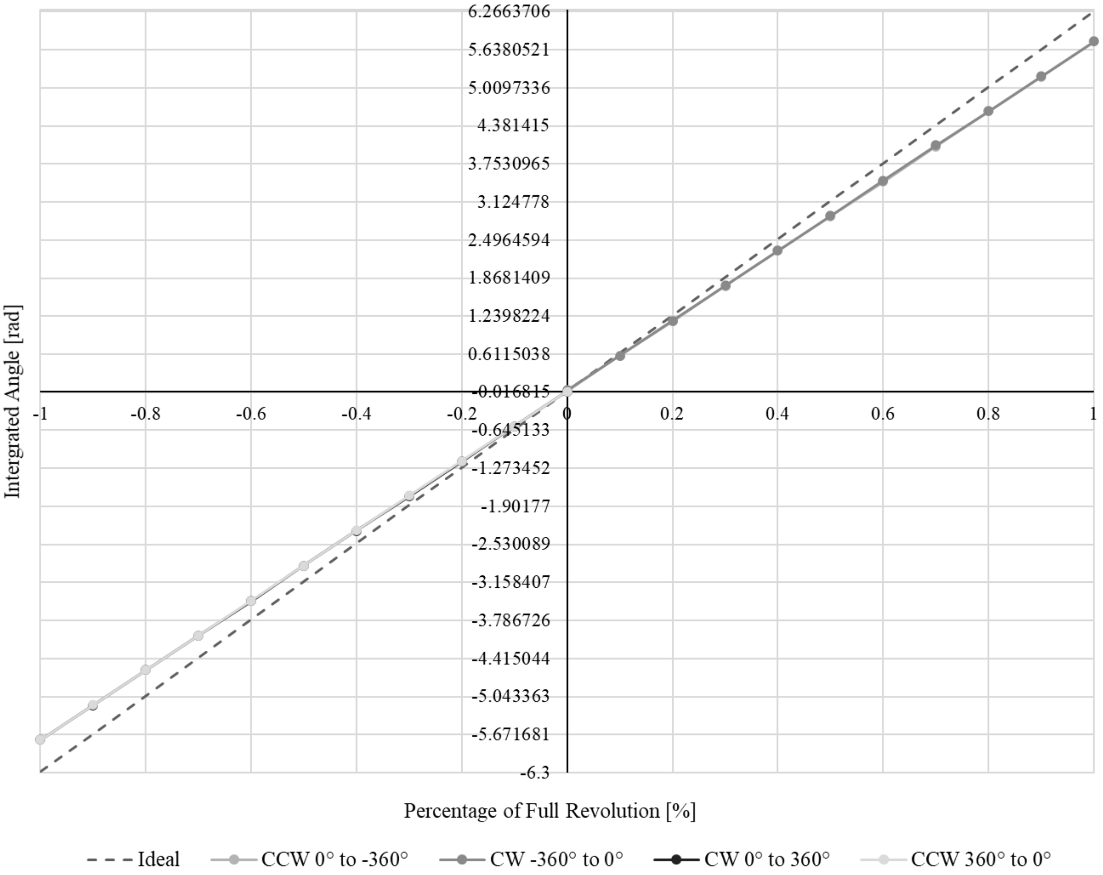


Figure AA.1: Integrated Angle vs. Real World Angle for Unit A

The author decided to implement a compensation algorithm to compensate for the divergent behaviour. The actual angle was recorded directly from the drives using the FCT software via serial communication; this value was then compared to the integrated value for the same position to calibrate the mitigation algorithm.

In total, four types of tests were carried out:

- Counter clockwise (CCW) rotation from 0°to -360°
- Counter clockwise (CCW) rotation from 360°to 0°
- Clockwise rotation (CW) from 0°to 360°
- Clockwise rotation (CW) from -360°to 0°

The results of these four test types (run five times each), with measurements taken in 36° intervals, were averaged out to a single set of values for each text type. The result of these four test are plotted onto a single graph, figure AA.1 for unit A and figure

AA.2 for unit B.

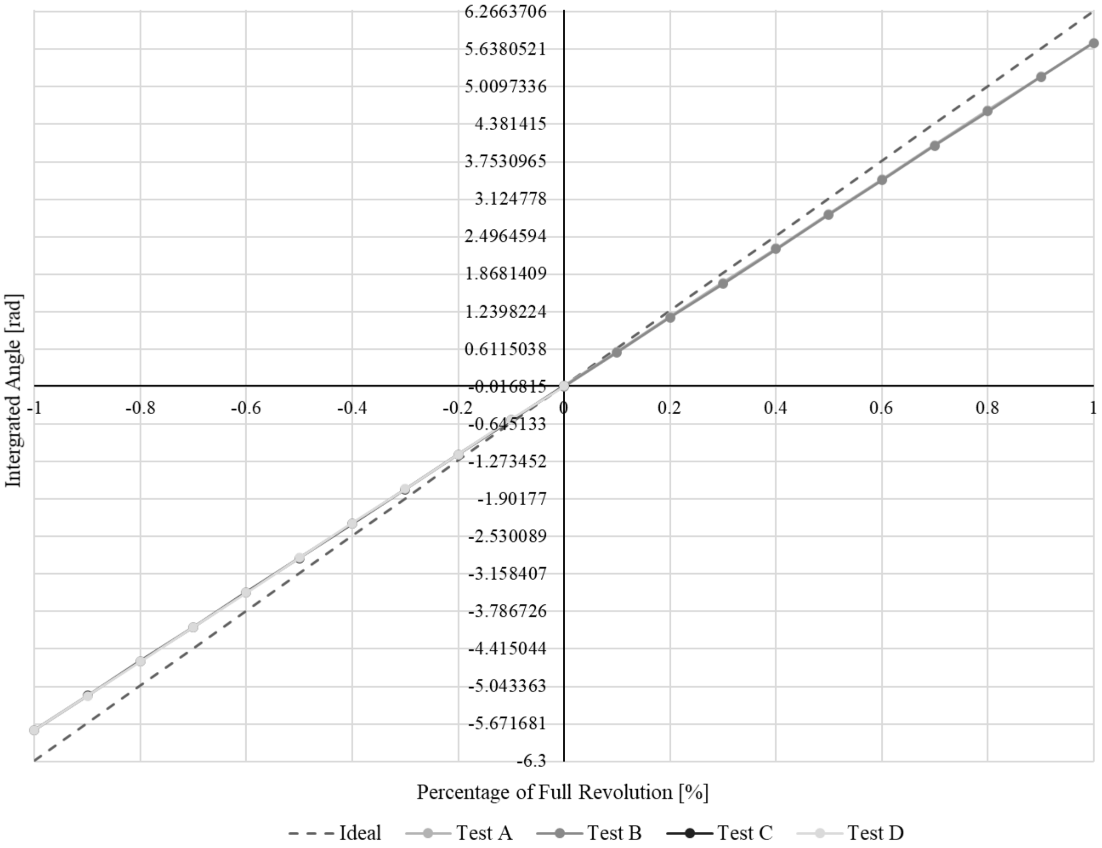


Figure AA.2: Integrated Angle vs. Real World Angle for Unit B

Note: Since the Festo FCT software does not use degrees or radians but rather a percentage of a complete revolution as its unit, this unit was also used as the independent variable unit for the graphs to keep consistency with the thesis.

As can be seen in both figure AA.1 and figure AA.2, the deviation appears to be relatively constant, increasing at a linear rate the further that the steering moves away from the zero point. There also appears to to minimal hysteresis as CCW 0° to -360° and CW -360° to 0° perfectly overlay each other and so does CW 0° to 360° and CCW 360° to 0°.

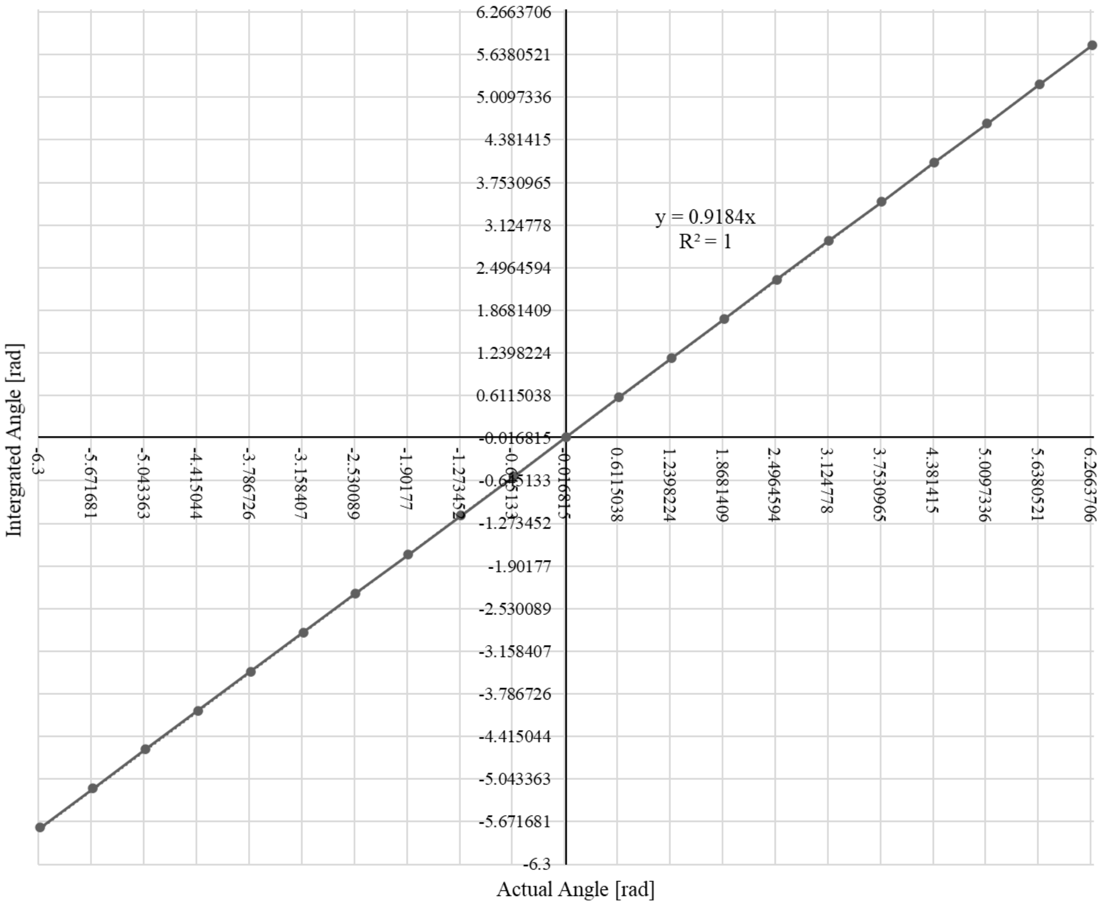


Figure AA.3: Unit A Integrator Compensation

Using the result developed in figure AA.1 and figure AA.2, it was possible to develop a formula for both integrators that compensated for the deviation. This compensation algorithm's results are shown in figure AA.3 for unit A and figure AA.4 for unit B.

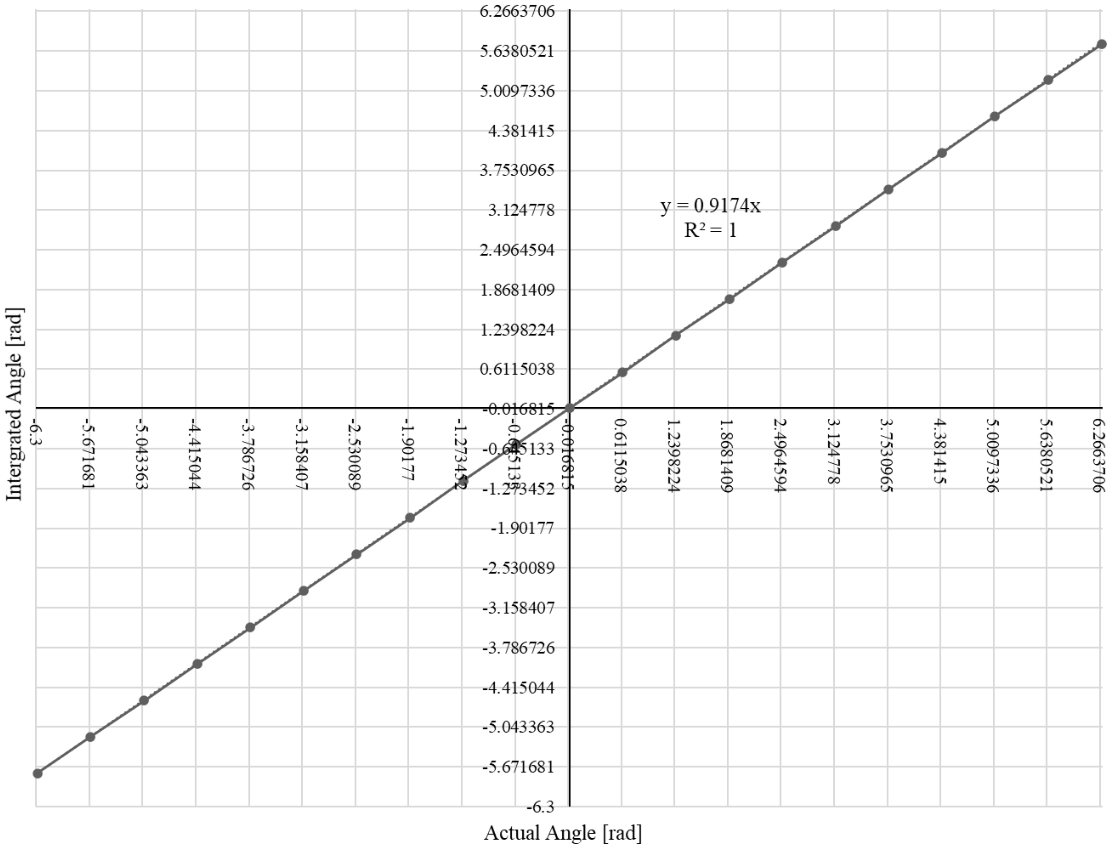


Figure AA.4: Unit B Integrator Compensation

Using figure AA.3 and a line of best fit (recorded in equation AA.1) was derived. Equation AA.1 can be used to compensate for the deviation the integrator generates to map the integrator angle value to the real world angle.

$$\theta_{actual} = \left(\frac{1}{0.9184}\right)\theta_{integrated} \tag{AA.1}$$

The same was done for unit B, whose resulting compensation equation can be found in equation AA.2.

$$\theta_{actual} = \left(\frac{1}{0.9174}\right)\theta_{integrated} \tag{AA.2}$$

θ_{actual} = Real world angle of steering *degrees*

$\theta_{integrated}$ = Integrator resultant angle *degrees*

AA.2 Absolute Encoder Calibrations

The potentiometers used as absolute encoders are linear; however, this does not mean that their behaviour is linear enough to be used as an encoder straight out of the box. During testing, it was found that the encoders are slightly non-linear.

The steering was manually homed to the zero degree point using a protractor to calibrate the absolute pots. The steering was incremented in 7.2° steps, with the resultant digitised analogue value on the PLC being recorded. For the Siemens range of PLCs, all 0V-10V analogue signals are digitised to 0-27648. The raw results of these tests are illustrated in figure AA.5 (unit A) and figure AA.6 (unit B).

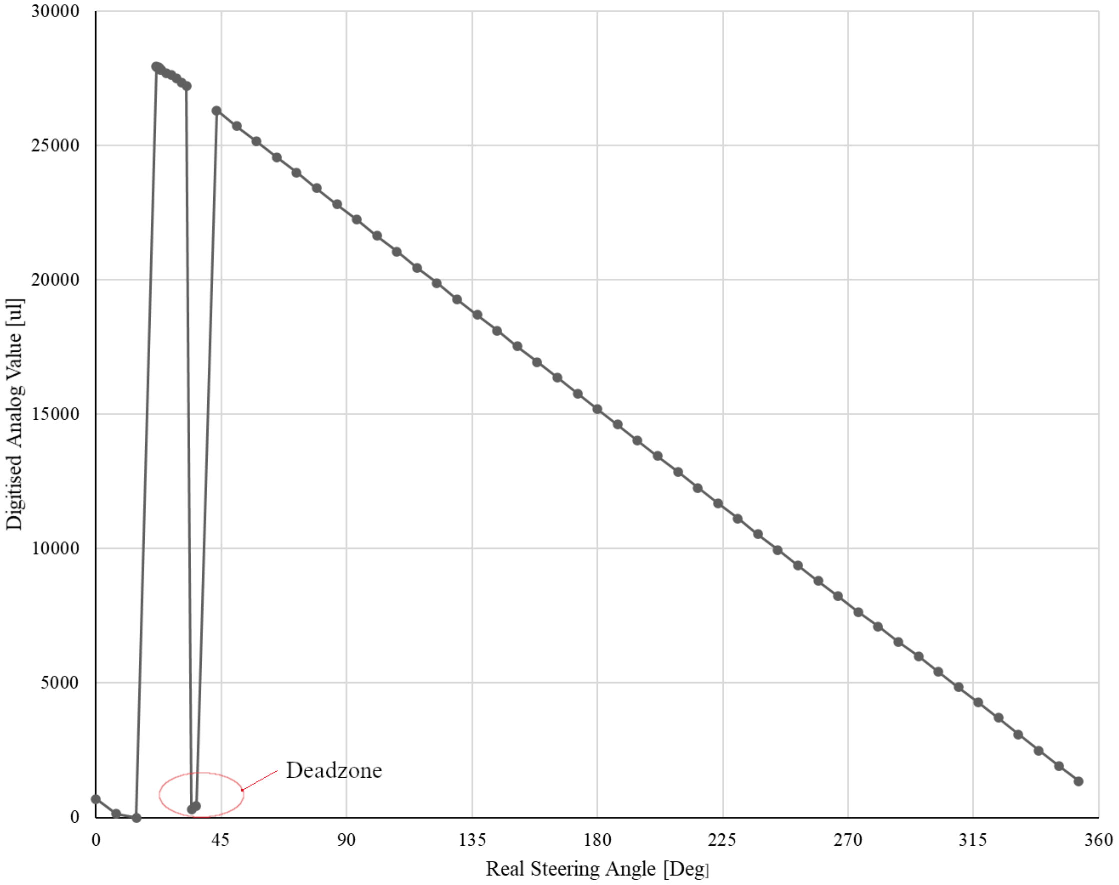


Figure AA.5: Unit A Raw Potentiometer Data vs Angle

As illustrated in figure AA.5, there exists a dead spot between 32.4° and 43.2 °. This dead spot was due to the previous owner's fix on the potentiometer since these potentiometers are second-hand. These potentiometers are specialised potentiometers, which, unlike "normal" potentiometers, have a full 360°endless rotation and a higher level of linear accuracy. Thus, it takes a long time to source them from the manufacturer, making replacement untenable. This issue was easy to work around in code, so it was not a major issue. For unit A, it can be seen that the zero degrees value (0°) corresponds to a potentiometer value of 686. If the pot were removed from the AGV and reseated, this value would change.

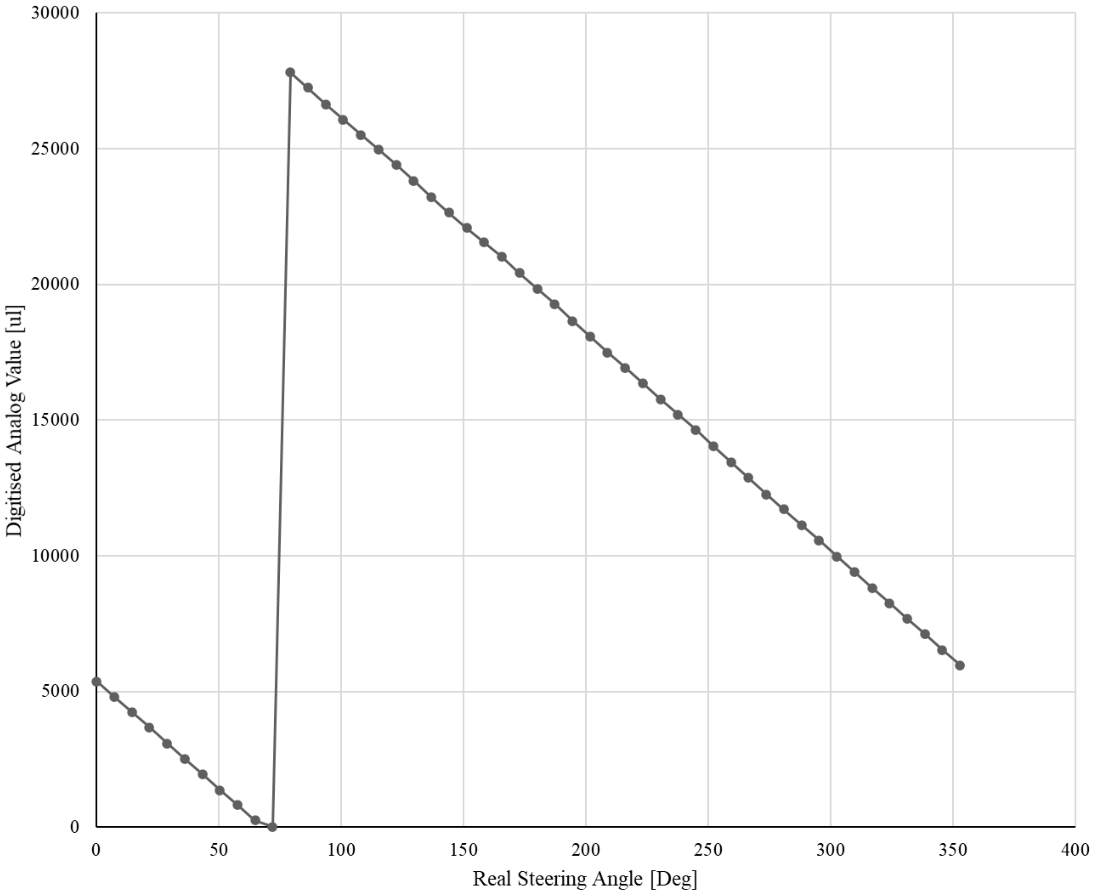


Figure AA.6: Unit B Raw Potentiometer Data vs Angle

Unit B's zero degree point (0°) corresponds to a digitised analogue value of 5374.

To simplify the graphs illustrated in figure AA.5 and figure AA.6. The independent variable zero point is shifted to generate figure AA.7 for unit A and figure AA.8 for unit B.

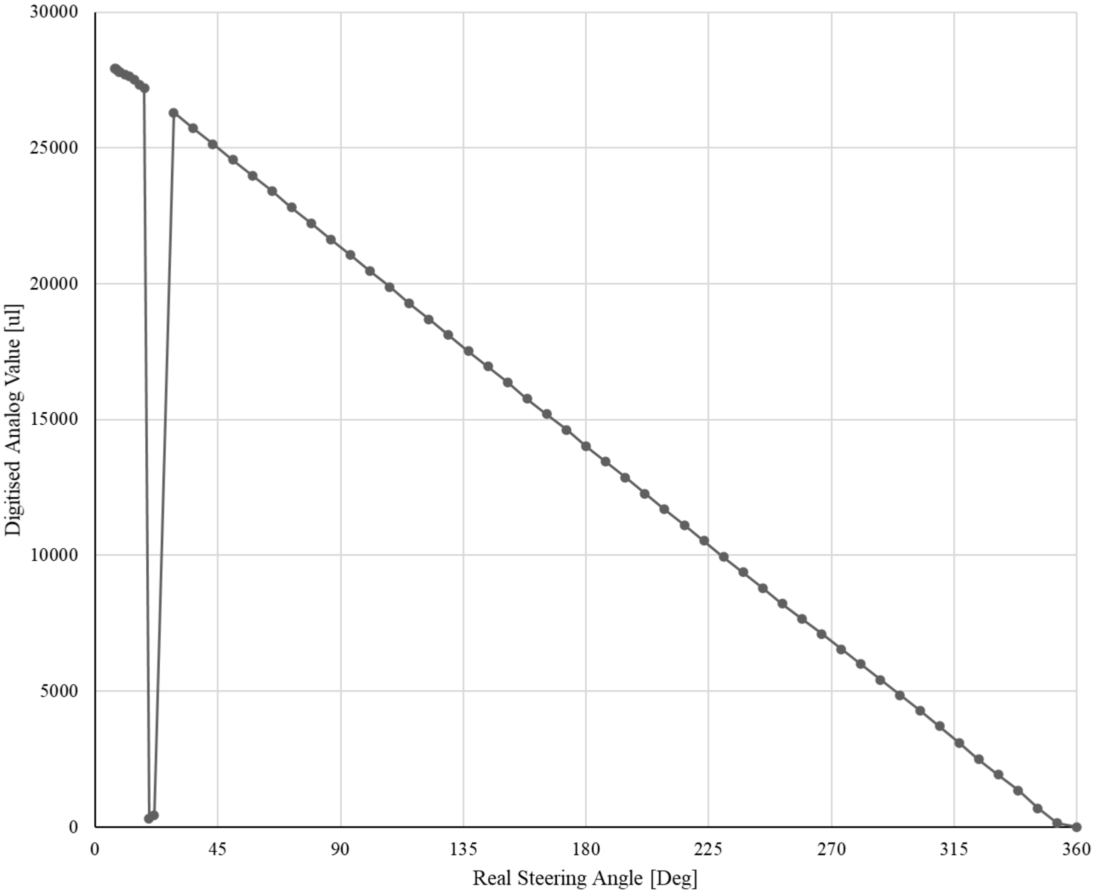


Figure AA.7: Unit A Raw Potentiometer Data vs Angle with Shifted Zero

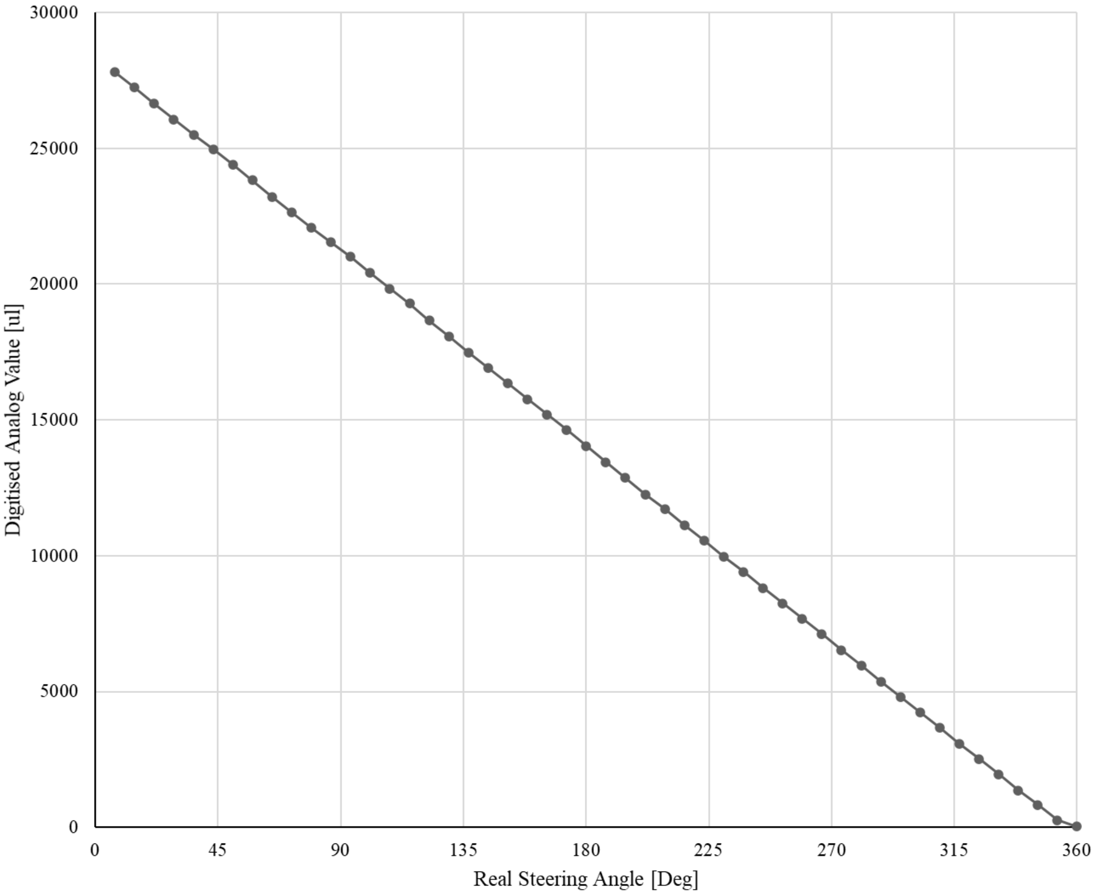


Figure AA.8: Unit B Raw Potentiometer Data vs Angle with Shifted Zero

By inverting the potentiometer measurement (which is possible since the 10 V and 0 V side can be arbitrarily chosen) a set of graphs can be generated from the shifted graphs illustrated in figure AA.7 and figure AA.8 that intersect the (0,0) point. This is useful to generate an equation that represents the relationship between the potentiometer values and real world angles. These (0,0) intersect graphs are illustrated in figure AA.9 for unit A and figure AA.10 for unit B.

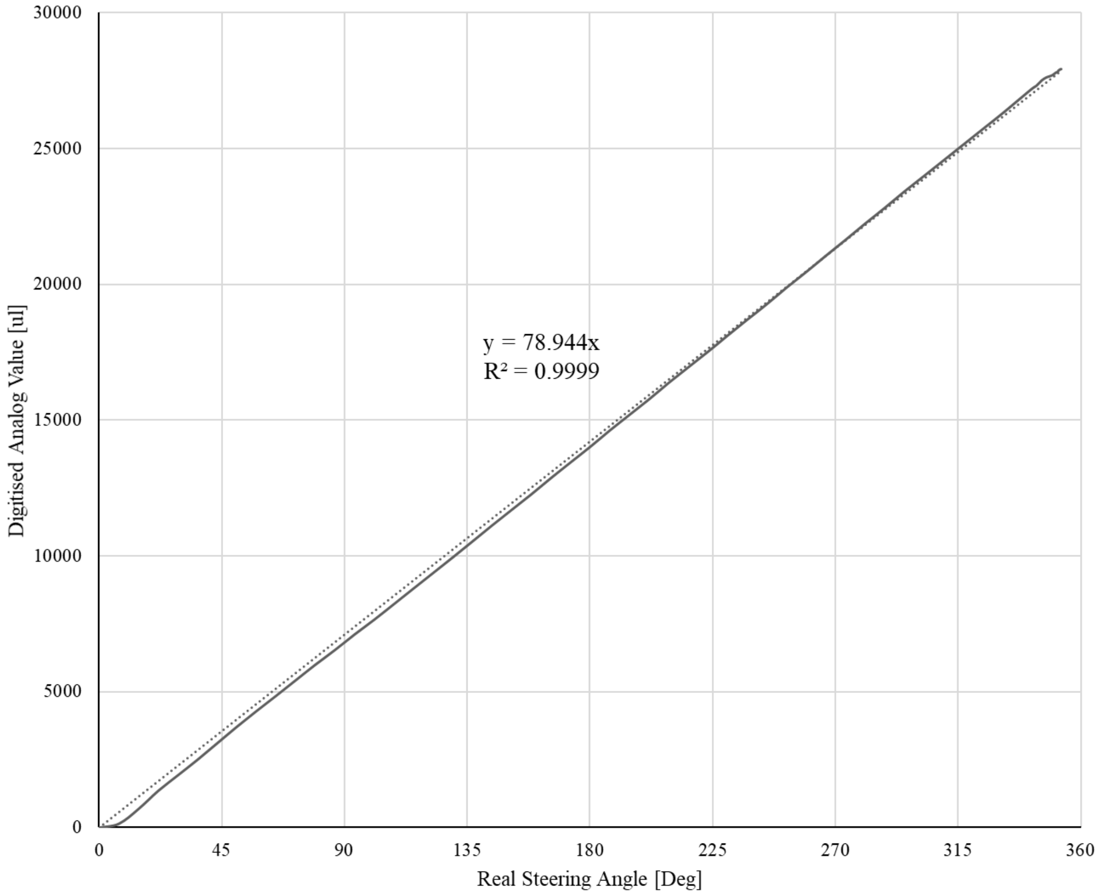


Figure AA.9: Inverted and Zero Shifted Unit A Pot Vs Angle

It should be noted that in figure AA.9, the dead spot values were removed from the graph, as these outliers would skew the relationship equation.

For unit A the equation that represents the relationship between the real-world angle and the potentiometer value, as calculated from graph AA.9 is:

$$\theta_{actual} = \left(\frac{1}{78.944}\right) x_{potA} \tag{AA.3}$$

θ_{actual}	=	Real world angle of steering	$degrees$
x_{potA}	=	Digitised Potentiometer Value	

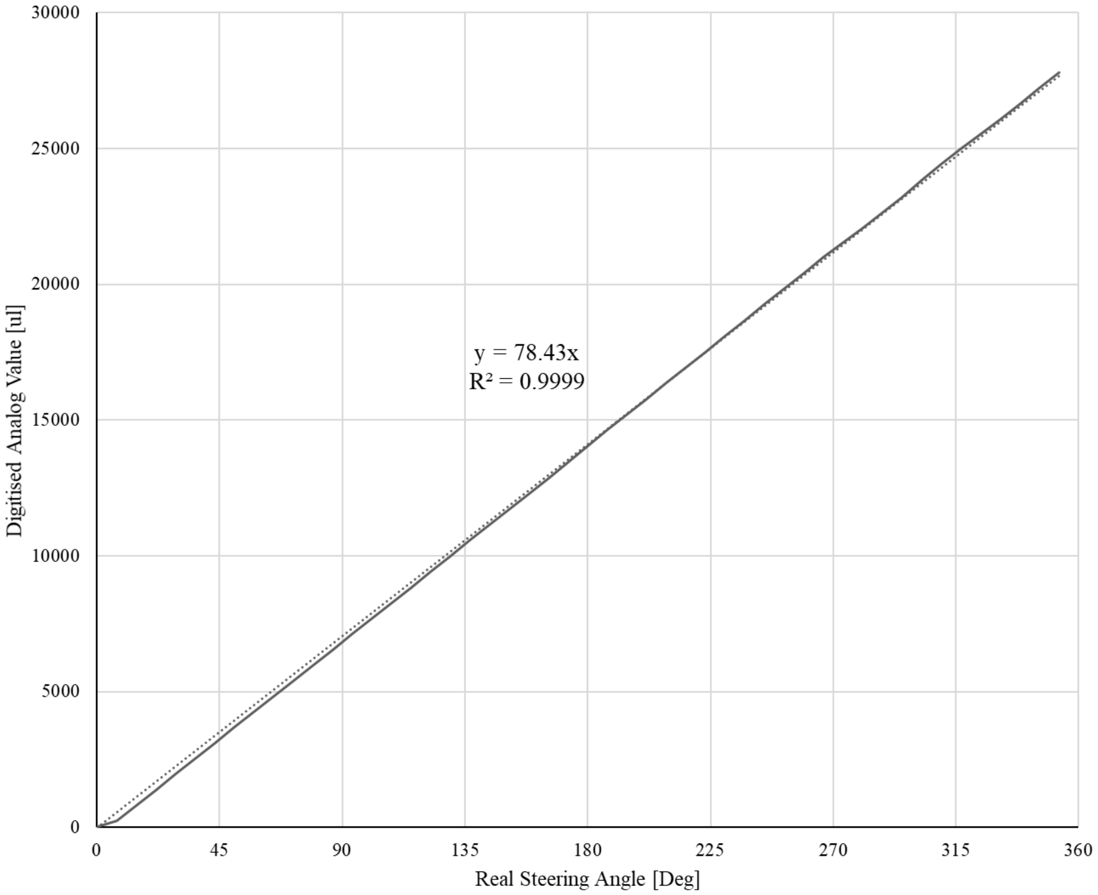


Figure AA.10: Inverted and Zero Shifted Unit B Pot Vs Angle

The relationship between the steering potentiometer for unit B and the real world angle, as derived from figure AA.10, is given in equation AA.4.

$$\theta_{actual} = \left(\frac{1}{78.43}\right) x_{potB} \tag{AA.4}$$

x_{potB} = Digitised Potentiometer Value