INTEGRATION OF A VISION-GUIDED ROBOT
INTO A
RECONFIGURABLE COMPONENT-HANDLING PLATFORM

VERNON VILJOEN

Dissertation submitted in fulfilment of the requirements for the

MAGISTER TECHNOLOGIAE:

ENGINEERING ELECTRICAL

In the

School of Electrical and Computer Systems Engineering

of the

Faculty of Engineering, Information and Communication Technology

at the

Central University of Technology, Free State

Supervisor: Prof HJ Vermaak PhD

DECLARATION

I, VERNON VILJOEN, identity number [redacted] and student number 20030681, do hereby declare that this research project which has been submitted to the Central University of Technology, Free State, for the Degree Magister Technologiae: Engineering Electrical, is my own independent work and complies with the Code of Academic Integrity, as well as other relevant policies, procedures, rules and regulations of the Central University of Technology, Free State, and has not been submitted before by any person in fulfilment (or partial fulfilment) of the requirements for the attainment of any qualifications.

……………………………… ……………………

SIGNATURE OF STUDENT DATE

ACKNOWLEDGEMENTS

I would like to thank the following people who contributed towards the completion of this dissertation~~:~~

- First and foremost, thanks to the LORD, for through him all is possible.
- Dr H Vermaak, for his guidance and inspiration. Thank you for giving me the chance to prove myself when nobody else would.
- Eugene from JENDAMARK, thank you for allowing me to make such liberal use of your inbox. Without your input I would still be on page one.
- Brian and Mike from WESTPLEX, thank you for your advice.
- Prof. GD Jordaan, thank you for all your advice. Although at times it felt more like criticism, it was just what was needed.
- The Central University of Technology, Free State, for providing the research facilities and financial support.
- I would like to thank my parents for always believing in me and giving me the opportunity to study.
- To my fiancé's parents, thank you for all your support and constant love.
- To all my colleagues and individuals not mentioned here who helped or supported me in any way, thank you.

Finally, thanks to my fiancé, Annelle, for all your love, continuous support, patience, encouragement and understanding during this time.

SUMMARY

The latest technological trend in manufacturing worldwide is automation. Reducing human labour by using robots to do the work is purely a business decision. The reasons for automating a plant include:

- Improving productivity
- Reducing labour and equipment costs
- Reducing product damage
- Monitoring system reliability
- Improving plant safety.

The use of robots in the automation sector adds value to the production line because of their versatility. They can be programmed to follow specific paths when moving material from one point to another and their biggest advantage is that they can operate for twenty-four hours a day while delivering consistent quality and accuracy.

Vision-Guided Robots (VGRs) are developed for many different applications and therefore many different combinations of VGR systems are available. All VGRs are equipped with vision sensors which are used to locate and inspect various objects. In this study a robot and a vision system were combined for a pick-and-place application. Research was done on the design of a robot for locating, inspecting and picking selected components from a moving conveyor system.

OPSOMMING

Outomatisering van vervaardigings prosesse is die nuutste neiging wereldwyd. Deur hande arbeid te verminder en eerder robotte te gebruik om die werk te doen, is definitief ‘n besigheidsbesluit. Redes vir die automatisering van ‘n aanleg kan van die volgende, of selfs almal insluit:

- Verbeter produktiwiteit
- Verlaag arbeids en toerustingkostes
- Verminder produkbeskadiging
- Stelselbetroubaarheid kan gemonitor word
- Verbeter aanlegveiligheid

Die gebruik van robotte in die outomatiseringssektor het waarde gevoeg tot die produksielyn omrede dit so veelsydig is. Dit kan geprogrammeer word om ‘n spesifieke taak uit te voer terwyl materiaal van een punt na ‘n ander vervoer word. Die grootste voordeel is dat hulle vir die volle vier-en-twintig uur van ‘n dag kan werk teen ‘n konstante kwaliteisvlak en akkurate tempo.

Visuele Geleide Robotte (VGR) word vir verskillende toepassings ontwikkel en daarom is daar baie verskillende combinasies van VGRs beskikbaar. Alle VGRs beskik van visuele sensors wat gebruik word om verskillende komponente te identifiseer en te inspekteer. In hierdie studie word ‘n robot en visuele stelsel gekombineer om ‘n optel-en-plaas toepassing te realiseer. ‘n Deeglike navorsingstudie is gedoen om komponente te vind, inspekteer en selektief komponente op te tel vanaf ‘n bewegende vervoerband.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AGV	Automatic Guided Vehicle
CGOS	Computer vision Graphic Operating System
CGP	Computer vision Graphics Processor
CPU	Central Programming Unit
FOV	Field of View
GM	General Motors
GPS	Global Positioning System
HMI	Human Machine Interface
INI	Initialisation
IP	Internet Protocol
KCP	KUKA Control Panel
KRC	KUKA Robot Controller
KRL	KUKA Robot Logic
LCD	Liquid Crystal Display
LED	Light-Emitting Diode
MTF	Modulation Transfer Function
NASA	US National Aeronautics and Space Administration
OLE	Object Linking and Embedding

PC	Personal Computer
PLC	Programmable Logic Controller
SCADA	Supervisory Control and Data Acquisition
SCARA	Selective Compliant Assembly Robot Arm
SMTP	Simple Mail Transfer Protocol
TCP	Tool Centre Point
TCP/IP	Transmission Control Protocol / Internet Protocol
UAV	Unmanned Aerial Vehicle
VGR	Vision-Guided Robotics
CCTV	Closed-Circuit Television
FTP	File Transfer Protocol
OPC	Open Process Control

Chapter 1: Introduction to Vision-Guided Robotics

1.1 Introduction

Automation with mechanical components is costly and demands high levels of precision. This has led to a shift towards more flexible, sensor-based solutions. A machine vision system can successfully handle unfamiliar situations while supplying accurate measurements. It can greatly improve mechanical inaccuracies when equipped with the relevant components, and tends towards greater productivity and less downtime.

However, why would the replacement of human beings with machine vision systems be preferred? According to the Wikipedia [1], a human being's degree of success is not constant, and the degree of failure among humans is classically high due to distraction, illness or other circumstances. It is clear that machine vision systems are favoured for use in visual inspections that require precision as well as high-speed, 24-hour operation and repeatability of measurements.

This has led to the integration of robots and machine vision, commonly referred to as Vision-Guided Robotics (VGR). Guiding robots via machine vision is an enabling technology for flexible manufacturing, allowing production lines to readily accommodate product changes. The VGR demonstrates its versatile capabilities for rapid product identification, inspection and orientation by recognising a specific object of interest from among different objects and displaying the image on a human machine interface (HMI) terminal.

VGR performance is significantly limited when a vision sensor cannot repeatably locate parts because of variations in the appearance of the part due to changes in orientation, size, contrast, lighting, overlapping parts and/or image focus [2].

Increasingly, robotic applications require machine vision for guiding robot movement for automated component handling and for quality control. Machine vision using video cameras and specialised computer programs can replace human vision, and can improve on human vision where precise and repeatable visual measurements and inspections are required.

For automated component handling, components are typically selected and placed on a transportation system by a robot or motion mechanism. A visionless robot requires accurately positioned of components to be able to find them. This requires costly and often unique fixtures to position each component type and assembly. A robot with vision capabilities can use less costly and more general fixtures and can be taught to find and place components on the assembly. Visual guidance can also compensate for some variations in the components, permitting tasks to be carried out which would be impossible with blind placement. Such vision systems can be either fixed or mounted on the robot arm, which allows a greater degree of reconfigurability of the vision system.

This project involved the development of a VGR that can pick and place components from a moving conveyor system, as opposed to a conveyor that must stop before any interaction between the robot and the components can take place. The VGR must also be able to carry out quality assurance inspection of components on an inspection table irrespective of their orientation.

1.2 Problem Statement

In the case of traditional industrial conveyor systems, either humans or costly and complex positioning systems are used to position objects to allow robots to pick the objects repeatedly at a predefined location. The obvious inflexibility of this method seriously limits the operation of such systems.

1.3 Aim of the Study

The aim of the study is to integrate machine vision with an industrial robot to allow objects to be selectively picked and placed from a moving conveyor.

1.4 Hypothesis

A VGR can be used to identify, inspect and pick objects from a moving conveyor. A vision system can relay the location data to the robot to enable it to selectively pick objects according to their orientation on the conveyor.

1.5 Research Methodology

A vision system was integrated on a robot arm to produce a VGR by means of the following:

- Objects to be inspected and picked are examined, which will be the basis of the design implementation.
- Robots and vision systems suitable for the specific task were reviewed and identified for implementation in a pick-and-place environment as stated above.

- Various interfacing software for communication between the robot and vision system was reviewed.
- The vision system was calibrated.
- A tool was specifically designed to pick and place the objects.
- The tool was calibrated for the robot to use.
- The vision system was programmed to locate and inspect specific objects.
- The robot was programmed to pick and place objects based on the data received from the vision system.
- The best lighting set-up was constructed for illuminating the objects.

1.6 References

[1] [Online] Available from: http://en.wikipedia.org/wiki/Machine_vision:MachineVision [Accessed 1 May 2008].

[2] COGNEX VGR Brochure [Online] Available from: http://cognex.com [Accessed 1 May 2008].

2 Chapter 2: Literature Review

2.1 Robots

Robots have been at the forefront of flexible automation for many years [1]. Much progress has been made since a radio-controlled robot boat was patented by Nikola Tesla in 1898 [2]. In the past, humans only fantasised about robots, but nowadays many types of robots have become a reality. For example, robots are used in industry, space exploration, the medical field and agriculture. There are toy robots for entertainment, and to an increasing extent humanoid robots are being created for such tasks as helping with chores in the home or caring for the elderly and the handicapped [3].

Joe Engelberger started the industrial ball rolling in 1961 when his firm Unimation delivered General Motors' (GM’s) first robot. Engelberger is regarded as the father of the industrial robot [4].

The world's first working robot joined the assembly line at the GM plant in Ewing Township in the spring of 1961 [5]. It was nothing like the robots of the present day.

It was an automated die-casting mould that dropped red-hot door handles and other such car parts into pools of cooling liquid on a line that moved them along to workers for trimming and buffing. Its most distinctive feature was a grip on a steel armature that eliminated the need for a man to touch car parts just cast from molten steel [5].

GM executives did not publicise it at the time because, as the creator of the die-caster said, it was an experimental technology and they were afraid it would not work [5].

2.2 Computer Vision

Computervision was a company started in 1969 by Marty Allen and Philippe Villers, headquartered in Massachusetts, USA. The company's early products were built on a Data General Nova platform. Starting around 1975, the company built its own Computer vision Graphics Processor (CGP) Nova-compatible 16-bit computer with added instructions optimised for graphics applications and using its own operating system known as Computer vision Graphic Operating System (CGOS) [6] .

Computer vision is the science and technology of the building of artificial systems that obtain data from images. Images are acquired from many different devices such as multiple cameras, video cameras or singular cameras.

Computer vision can also be described as complementary to (but not necessarily the opposite of) biological vision. In biological vision, the visual perception of humans and various animals is studied, resulting in models of how these systems operate in terms of physiological processes [7]. Computer vision is considered more as the study and description of artificial vision systems and their implementation.

Computer vision as a whole can be subdivided into image analysis, robot vision, machine vision and image processing as the applications and techniques have a significant overlap, i.e. these fields are more or less identical and can be interpreted as different elements of one specific field of study with different names or subdivisions.

On the other hand, it appears to be necessary for research groups, scientific journals, conferences and companies to present or market themselves as specifically working in one of these fields, hence various characterisations which distinguish the fields from each other have been presented [7].

The following characterisations appear relevant but should not be taken as universally accepted:

- Image processing and image analysis tend to focus on 2D images, and how to transform one image into another, e.g. by pixel-wise operations such as contrast enhancement, local operations such as edge extraction or noise removal, or geometric transformations such as rotating the image. This characterisation implies that image processing or analysis neither require assumptions nor produce interpretations about the image content.
- Computer vision tends to focus on the 3D scene projected onto one or several images, e.g. how to reconstruct the structure of other information about the 3D scene from one or several images. Computer vision often relies on more or less complex assumptions about the scene depicted in an image.
- Machine vision tends to focus on applications, mainly in industry, e.g. vision-based autonomous robots and systems for vision-based inspection or measurement. This implies that image sensor technologies and control theory are often integrated with the processing of image data to control a robot and that real-time processing is emphasised by means of efficient implementations in hardware and software. It also implies that the external conditions such as lighting can be and often are more controlled in machine vision than they are in general computer vision, which can allow different algorithms to be used.
- There is also a field called imaging which primarily focuses on the process of producing images, but sometimes also deals with the processing and analysis of images. For example, medical imaging involves much analysis of image data.

- Finally, pattern recognition uses various methods to extract information from signals in general, mainly based on statistical approaches. A significant part of this field is devoted to applying these methods to image data [7].

A consequence of this state of affairs is that someone can be working in a laboratory related to one of these fields, apply methods from a second field to solve a problem in a third field, and present the results at a conference related to a fourth field [7].

2.2.1 Applications of Vision

The medical field is the most prominent field of application for vision. This area is characterised by the extraction of information from image data for diagnosing a patient [7]. Vision is also used to aid surgeons during operations – robots are used to aid the delicate manoeuvring of surgical instruments inside the patient.

Vision is applied in industry to extract information to support material-handling and/or manufacturing processes [7]. Application areas include quality control, where products are automatically inspected to detect defective products, and positioning to guide a robot with respect to the orientation and location of the product to be picked.

Military applications of vision are probably the largest. A modern military concept is "battlefield awareness", which requires various sensors, including image sensors [7], to be used to provide vital and current and information on the battle scene. Military action has proven to be the driving force behind many technological advances. Until fairly recently, Global Positioning Systems (GPS) were limited to military use until their benefits for civilians was realised.

One of the latest applications is Unmanned Aerial Vehicles (UAVs). Space exploration is already being done with autonomous vehicles using computer vision, e.g. the National Aeronautics and Space Administration's (NASA's) Mars Exploration Rover shown in Figure 2-1 [7].

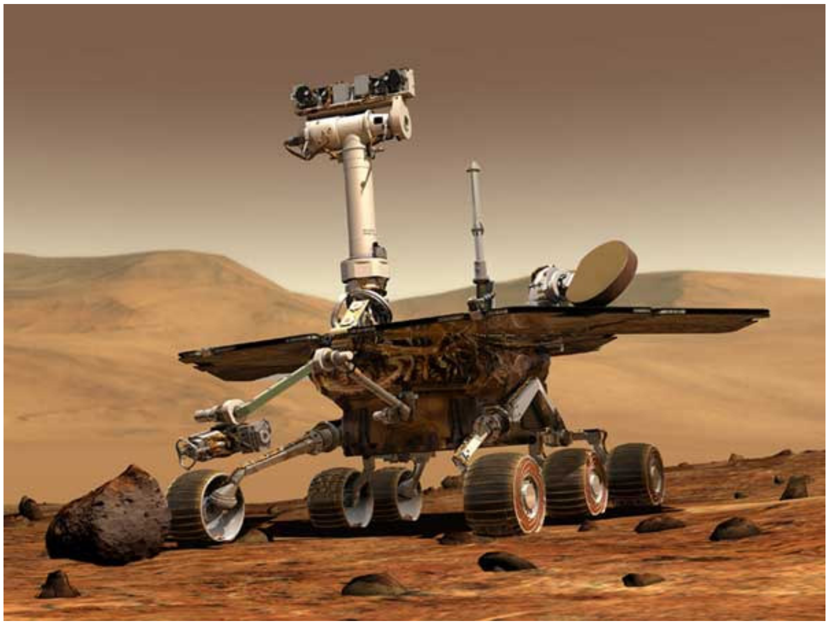


Figure 2-1 Mars Exploration Rover [7]

Figure 2.2 shows the launch by US Navy SEAL frogman-commandoes of a Boeing Insitu ScanEagle drone incorporating vision abilities using a pneumatic catapult. The UAV was subsequently recovered using the "Skyhook" system, in which the aircraft snags a line hanging from a 50-foot pole [8].

Figure 2-2 ScanEagle UAV [8]

These are just a few of the wide range of applications in many different sectors where vision is currently being applied.

2.3 Vision-Guided Robotics

Since the 1980s, vision-guided robotics has been a difficult goal for engineers to achieve, but since its development and improved implementation, users can now push the technology even further than previously thought possible. New operational systems can be trained with ease, and it is no longer necessary to use humans to unload, load or sort incoming or outgoing parts. On machining and production lines, randomised parts can be piled in bins and costly parts feeders used for part orientation can be replaced [9]. This leads to cost savings in countries where labour overheads are particularly high.

2.3.1 Various Implementations of Vision-Guided Robots

An Automatic Guided Vehicle (AGV) equipped with an omni-directional imaging camera has the ability to view a 360-degree area [10]. This gives the AGV the ability to view its surrounding and, based on the data received, it is able to navigate around obstacles. The AGV can then operate continuously, lifting and moving heavy objects.

Future space missions will be able to land robot space vehicles in areas of interest to scientists which were previously avoided due to the often hazardous terrain. To enable this capability, a fully autonomous onboard system that identifies and avoids hazardous features such as steep slopes and large rocks is required [11]. An algorithm has been applied on an autonomous helicopter test bed and demonstrated four times, enabling the first autonomous landing of an unmanned helicopter in unknown and hazardous terrain [11].

Robotics is a growing field of application in many different sectors. Robots are usually required to carry out pre-programmed tasks of motion. These tasks are then repeated continuously while the robot is in operation. The advantage of providing such a robot with vision capabilities is that the robot can automatically adapt to the environment, i.e. be able to randomly pick objects of interest. In one such application a batch of a few thousand small particles can be prepared as a monolayer, and then presented to an imaging system. Pattern recognition techniques are then applied to the data to identify target particles within the particle bed [12]. A vacuum nozzle is used on a Selective Compliant Assembly Robot Arm (SCARA) to extract the target particles one at a time and sort them into concentrate bins according to their mineral classifications [12].

Due to global competition and the continual advances in technology, more flexible solution-based systems are required in the field of robotics. The integration of robots and machine vision can be of great use in areas where humans were commonly used for quality control, as this type of automated system does not suffer from fatigue and can operate continuously. In addition, it will save time when components have to be picked and placed from a moving conveyor [13]. This will do away with the need for costly object-positioning systems and the need to stop the conveyor to pick objects.

2.4 References

[1] [Online] Available from: http://www.densorobotics.com/PR/17/Assembly%20-%20Vision%20Guided%20Robotics.pdf [Accessed 1 May 2009].

[2] [Online] Available from: http://www.teslasociety.com/robotics.htm [Accessed 1 May 2009].

[3] [Online] Available from: http://www.buzzle.com/articles/types-of-robots.html [Accessed 1 May 2009].

[4] [Online] Available from: http://www.industryweek.com/articles/new_roles_for_robots_10443.aspx [Accessed 1 May 2009].

[5] [Online] Available from: http://www.capitalcentury.com/1961.html [Accessed 1 May 2009].

[6] [Online] Available from: http://en.wikipedia.org/wiki/Computervision [Accessed 1 May 2009].

[7] [Online] Available from: http://en.wikipedia.org/wiki/Computer_vision [Accessed 1 May 2009].

[8] [Online] Available from: http://www.theregister.co.uk/2008/02/12/seal_robot_aircraft_carriers/ [Accessed 1 May 2009].

[9] [Online] Available from: http://www.automationworld.com/feature-1878 [Accessed 1 May 2009].

[10] Swanepoel, P et al. Omni-Directional Image Sensing for Automated Guided Vehicle. *Robotics & Mechatronics Symposium,* 2008, 2, pp41–44, ISBN: 978-0-620-42463-9.

[11] . Johnson, A, Montgomery, J, Matthies L et al. Vision Guided Landing of an Autonomous Helicopter in Hazardous Terrain, *Robotics and Automation*, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference**,** pp3966–3971, ISBN: 0-7803-8914-X.

[12] Green, Jeremy J et al. The Robotic Handling of small Mineral Grains with a Vacuum Nozzle, *Robotics & Mechatronics Symposium,* 2008, 2, pp49–54, ISBN: 978-0-620-42463-9.

[13] Viljoen, V et al. Vision Guide Robotics in a Reconfigured Environment for an Integrated Component-Handling Platform, *Robotics & Mechatronics Symposium,* 2008, 2, pp45–48, ISBN: 978-0-620-42463-9.

3 Chapter 3: Methods and Techniques

3.1 System Overview

The system overview is illustrated in Figure 3-1. Components are travelling in the direction indicated by the red arrow. The following actions take place at the numbered locations in Figure 3-1:

1 Components are randomly loaded onto conveyor 1 and travel along the conveyor towards the robot.

2 The robot inspects and selectively picks components from conveyor 1 and places them on conveyor 2.

3 Components travel further along conveyor 2 to be inspected and processed further.

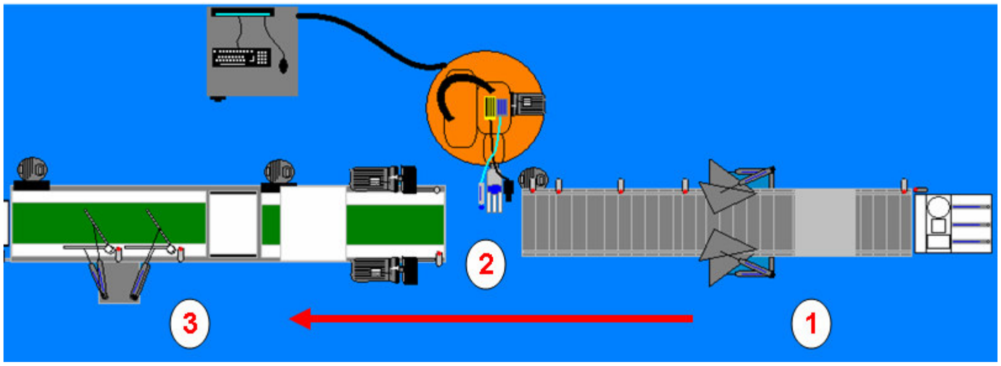


Figure 3-1 Top view of system

3.2 Introduction to system hierarchy

The systems hierarchy is shown in

Figure 3-2. It consists of the following:

- Robot and vision hardware used for the physical application of the system
- Robot software or KUKA robot logic (KRL) to program the robot's motions
- Interfacing software for communication between the robot and vision hardware
- Vision software or In-Sight Explorer used to program the vision system to detect and inspect components
- Lighting used to enhance and improve the image quality of components inspected by the vision hardware
- Lenses
- External environment – factors involved in the successful functioning of the system.

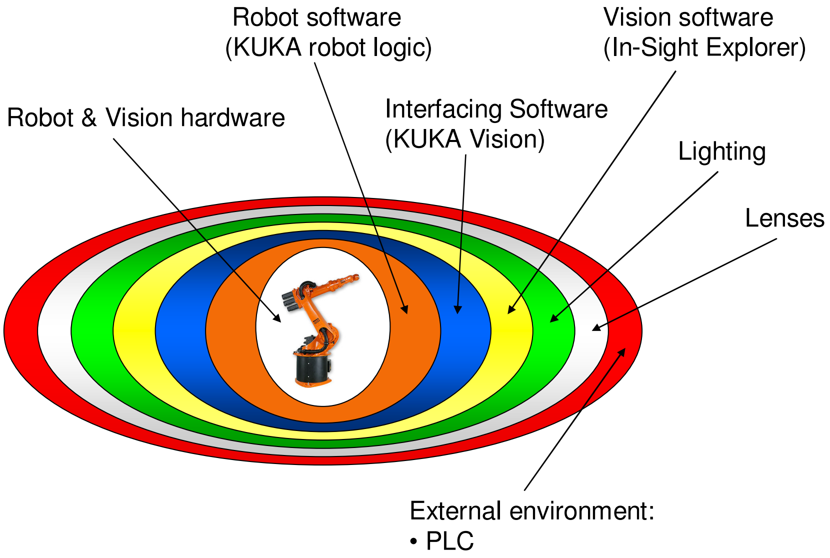


Figure 3-2 System hierarchy

3.3 Robot Hardware

3.3.1 KUKA KR6 Robot

As with the vision sensors, there are a number of robot manufacturers who supply products of high quality. It is even more difficult to select a robot than a vision sensor due to the physical and cost constraints of an application. A KUKA KR6 robot was selected for the project due to its flexibility and unique motion capabilities, which were ideal for the environment in which it was required to function.

The robot described in Figure 3-3 is a six-axis industrial robot with jointed-arm kinematics for all point-to-point and continuous-path controlled tasks.

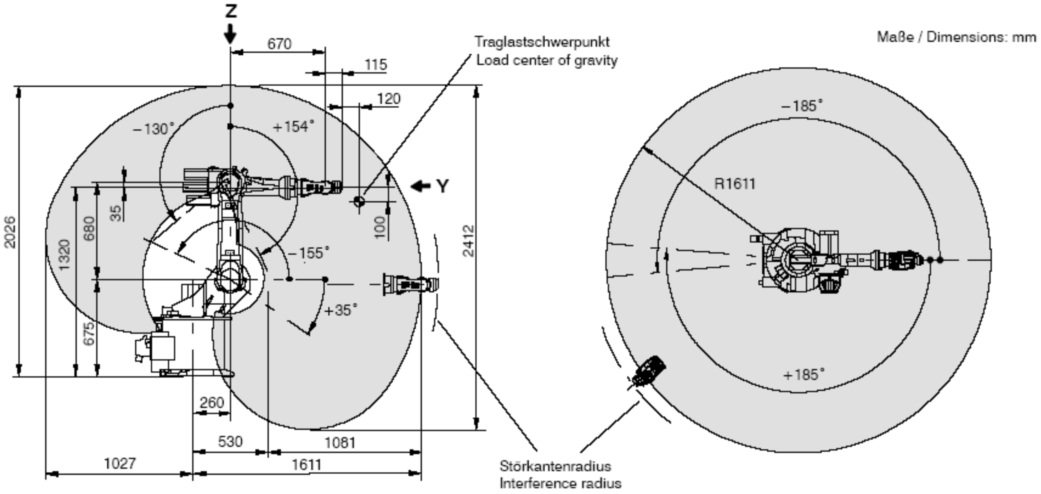


Figure 3-3 KUKA KR6 robot's range of movement [1]

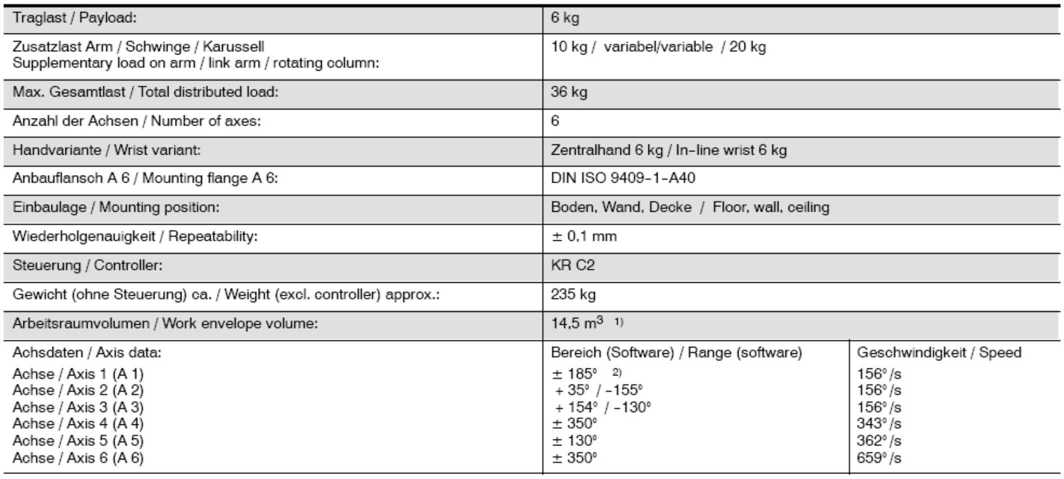
The main areas of application of this particular type of robot are:

- Handling of components
- Assembly of components

- Application of adhesives, sealants and preservatives as an element of assembly
- Machining through the attachment of suitable dynamic end-effectors.

The KR6 robot is installed on the floor for optimal functionality. Robots can be installed on the floor, the wall or the ceiling depending on the application. The rated payloads, i.e. the maximum permissible weight that the robot arm (as specified by KUKA) can handle at full speed while the arm is fully extended are 6 kg [1].

Table 1: KUKA technical data [1]



Traglast / Payload:	6 kg	
Zusatzlast Arm / Schwinge / Karussell Supplementary load on arm / link arm / rotating column:	10 kg / variabel/variable / 20 kg	
Max. Gesamtlast / Total distributed load:	36 kg	
Anzahl der Achsen / Number of axes:	6	
Handvariante / Wrist variant:	Zentralhand 6 kg / In-line wrist 6 kg	
Anbauflansch A 6 / Mounting flange A 6:	DIN ISO 9409-1-A40	
Einbaulage / Mounting position:	Boden, Wand, Decke / Floor, wall, ceiling	
Wiederholgenauigkeit / Repeatability:	± 0,1 mm	
Steuerung / Controller:	KR C2	
Gewicht (ohne Steuerung) ca. / Weight (excl. controller) approx.:	235 kg	
Arbeitsraumvolumen / Work envelope volume:	14,5 m^3 1)	
Achsdaten / Axis data:	Bereich (Software) / Range (software)	Geschwindigkeit / Speed
Achse / Axis 1 (A 1)	± 185° 2)	156°/s
Achse / Axis 2 (A 2)	+ 35° / -155°	156°/s
Achse / Axis 3 (A 3)	+ 154° / -130°	156°/s
Achse / Axis 4 (A 4)	± 350°	343°/s
Achse / Axis 5 (A 5)	± 130°	362°/s
Achse / Axis 6 (A 6)	± 350°	659°/s

All moving parts are covered and the joints and gears are almost free from backlash. All the axes are powered by brushless AC servomotors of plug-in design, which require no maintenance and offer reliable protection against overload [1].

The main axes are lifetime-lubricated, i.e. an oil change is necessary after 20 000 operating hours at the earliest. All the robot components are of intentionally simple and straightforward configuration. The robot can also be quickly replaced as a complete unit without any major program corrections being required. Overhead motion is possible [1].

These and numerous other design details make the KR6 robot fast, reliable and easy to maintain, with minimal maintenance requirements. It occupies very little floor space and can be located very close to the work piece on account of the special structural geometry. Like all KUKA robots, the KR6 has an average service life of 10 to 15 years [1].

Each robot is equipped with a controller whose control and power electronics are installed in a common cabinet. The controller is compact, user-friendly and easy to service [1].

The connecting cables between the robot and the control cabinet contain all the required energy supply and signal lines. The cable connections on the robot are of the plug-in type, as too are the power cables for the operation of end effectors [1].

The wrist of the robot has a flange for mounting end effectors such as grippers, suction pneumatics or welding tools [1].

3.3.2 I/O Modules for KUKA

A Beckhoff DeviceNet coupler is used for the I/O module needs of KUKA. The bus terminal system is the universal interface between a fieldbus system and the sensor/actuator level. The unit shown in Figure 3-4 consists of a bus coupler as the head station, and up to 64 electronic series terminals, the last one being an end terminal. Terminals are available for each technical signal, each having two I/O channels, and these can be mixed in any order [2]. The Beckhoff DeviceNet coupler was used for external wiring of inputs from the sensors and the programmable logic controller (PLC) to the KUKA robot.

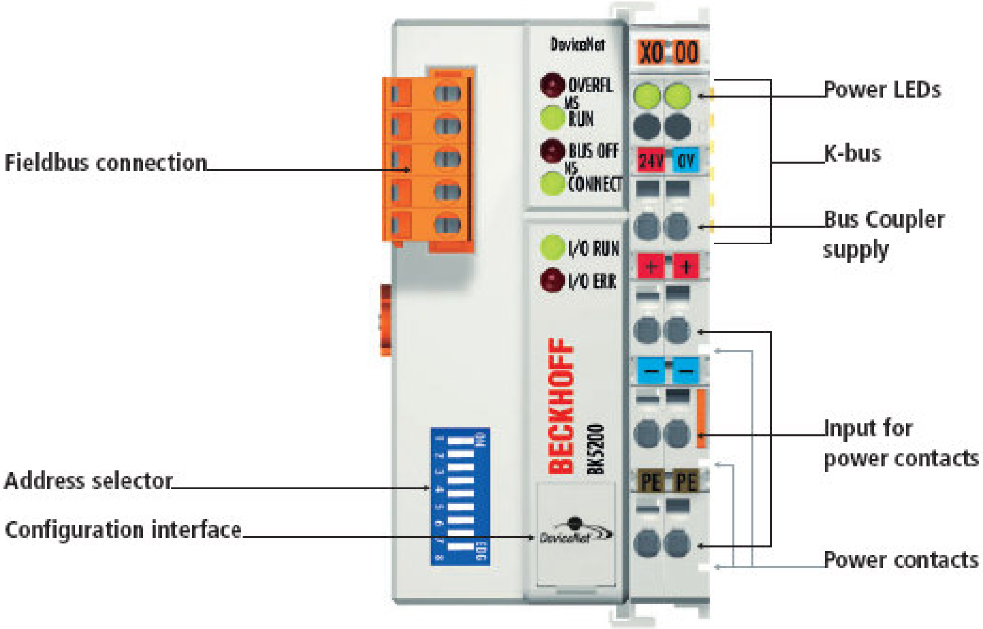


Figure 3-4 Beckhoff DeviceNet couplers [2]

3.3.3 End Effector

The hardware was mainly based on everything that is designed to fit on the end of the flange, also known as the end effector (e.g. gripper or tool). As the end effector, a three-fingered pneumatic gripper was mounted onto the end of the robot's flange. Additional tools had to be specially manufactured for the picking of components. It was decided that an extendable suction cup and camera bracket would be fixed onto the gripper. Figure 3-5 shows the extendable suction cup model 1, the first design attempt. The design required it to be able to be retracted when the grippers were used, and to extend past the grippers when they were needed again.

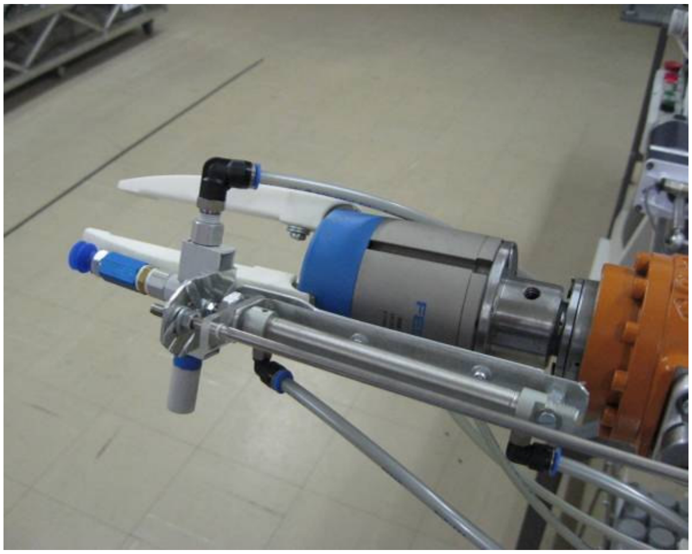


Figure 3-5 Extendable suction cup model 1

This first design worked with great success, but there were some design problems which had to be taken into consideration – there were motion limitations of the robot in the picking environment.

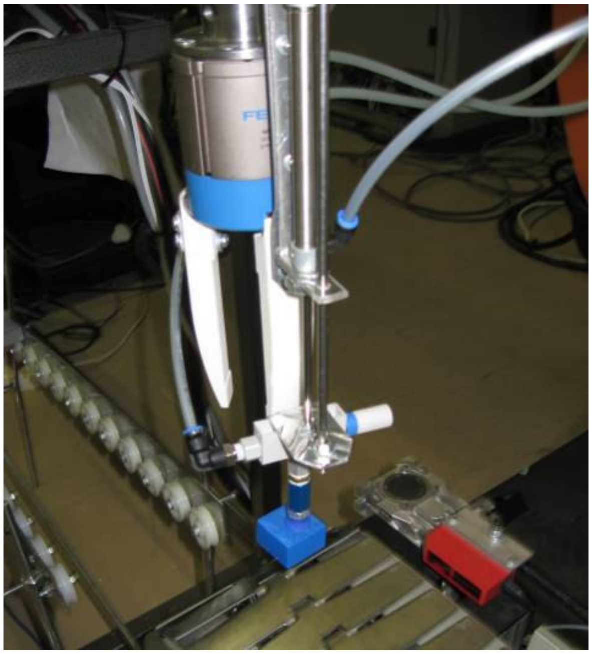


Figure 3-6 Suction cup mechanism extended

The first notable problems of model 1 were as follows:

- The design was too bulky as shown in Figure 3-6.
- The mounting bracket in Figure 3-7 allowed too much flexing and lateral movement as it was made of 2 mm aluminium plate. This meant that the calibrated picking point of the suction cup could be moved out of position accidentally if bumped or tampered with.
- The stabilising rod in Figure 3-7 was extended well beyond axis A5 of the robot when retracted. This caused the problem of a possible collision with axis A5 when model 1 was rotated on axis 6.

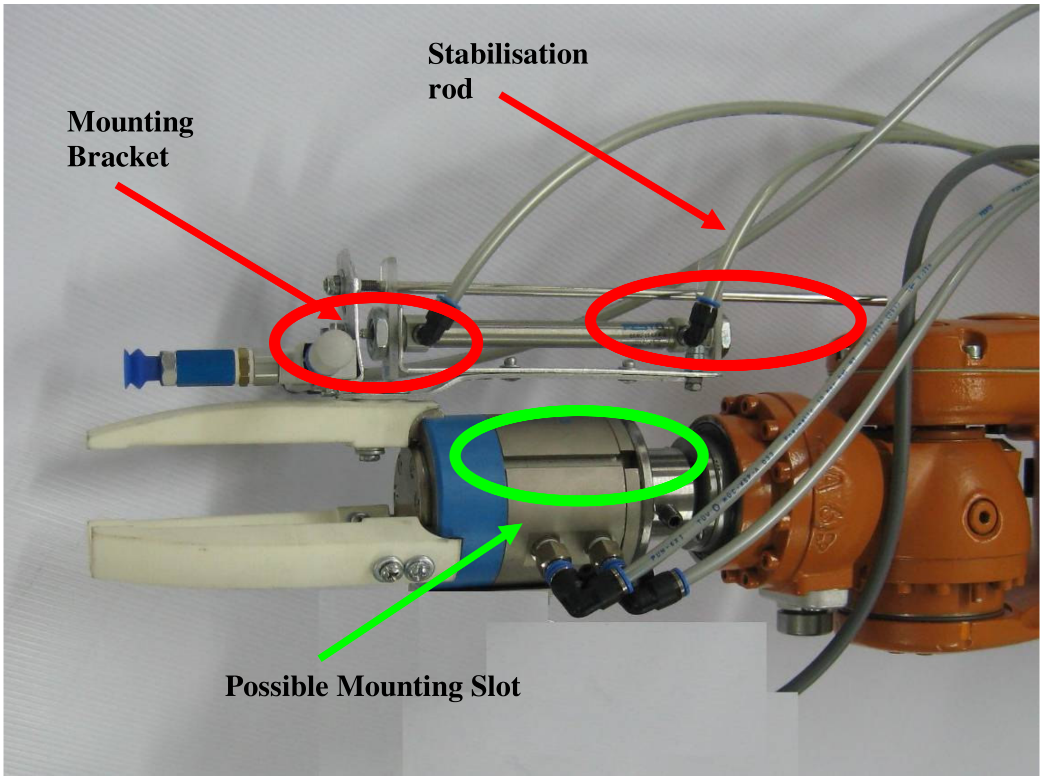


Figure 3-7 Extendable suction cup model 1, problems

After testing it was decided to try and improve the design of model 1. The consideration was to find a more suitable mounting area on the gripper for the new model as well as a sturdier design for the extendable suction cup.

In Figure 3-7, indicated in green, a possible solution was found for a stable mounting area for the extendable suction cup located on the gripper, namely a slot. The new design would now be mounted using the slot on the gripper as the fixing point. The design should also be such that it would not cause any motion limitations of the robot on axis A5 – or on any other axis for that matter.

The new extendable suction cup model 2, as shown in Figure 3-8, was manufactured from 3-mm-thick angle aluminium. This proved to be a much better design choice for the following reasons:

- The mounting bracket, which is fixed with three bolts to one of the slots on the gripper, will not move even if bumped or tampered with. This ensures that the suction cup will always be at the calibrated point when a pick action takes place.
- As can be seen in Figure 3-8, the stabilisation rod has been shortened, preventing any limitation of motion on any axis of the robot.

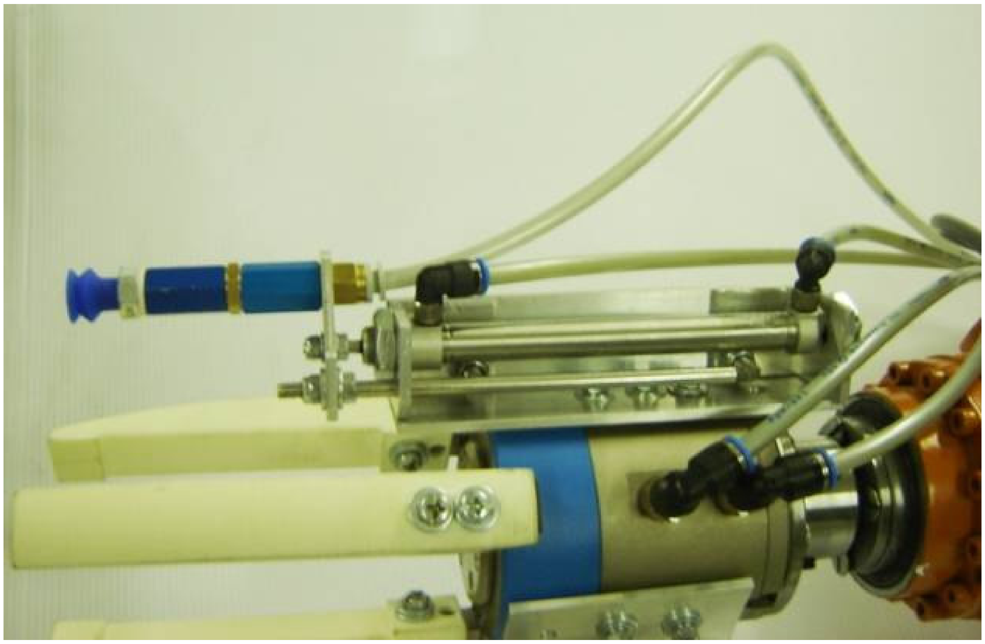


Figure 3-8 Extendable suction cup model 2

The final model design proved to be a big improvement over the previous design, and was successfully tested numerous times.

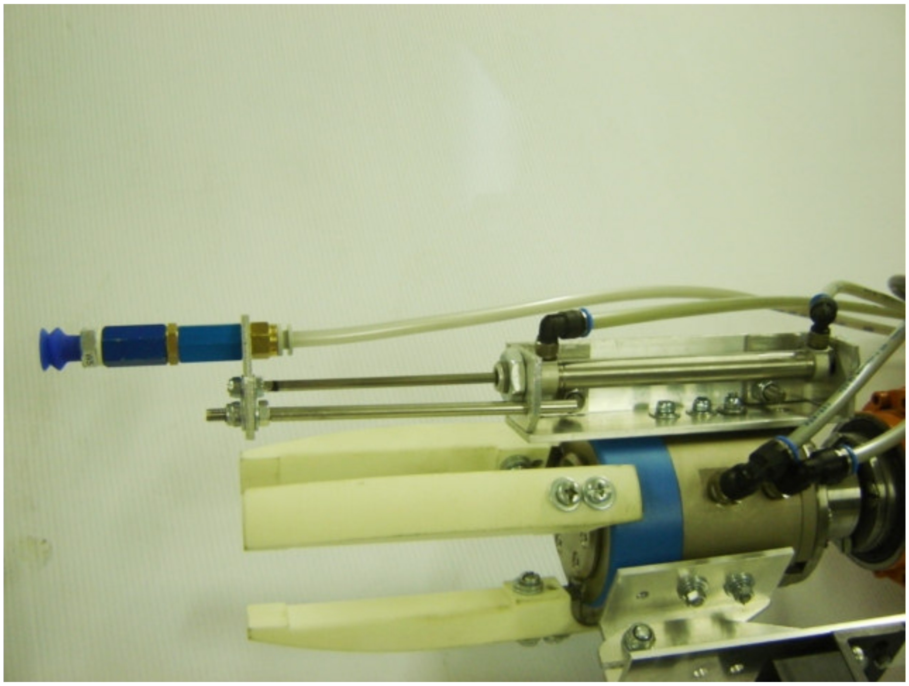


Figure 3-9 Model 2 extended

Any new tool that is mounted on the flange of the robot or on any other tool located on the flange must be calibrated. This means that the robot will know where the tool is located within the co-ordinates of its world, i.e. within the area of motion of the robot. To be able to calibrate the extendable model 1 or 2 suction cup, a tool centre point had to be established. This was done by machining a spike from brass to represent the tool centre point (TCP) of the suction cup during calibration. The spike is shown in Figure 3-10.

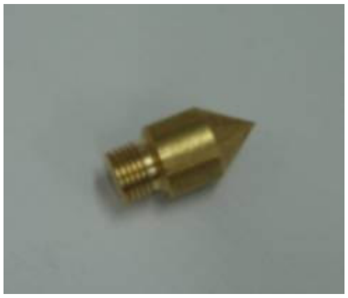


Figure 3-10 TCP calibration tip

The brass spike representing the centre point of the suction cup proved to be a great aid in the calibration of the extendable suction cup. Figure 3-11 illustrates the comparison of the brass spike and the suction cup.

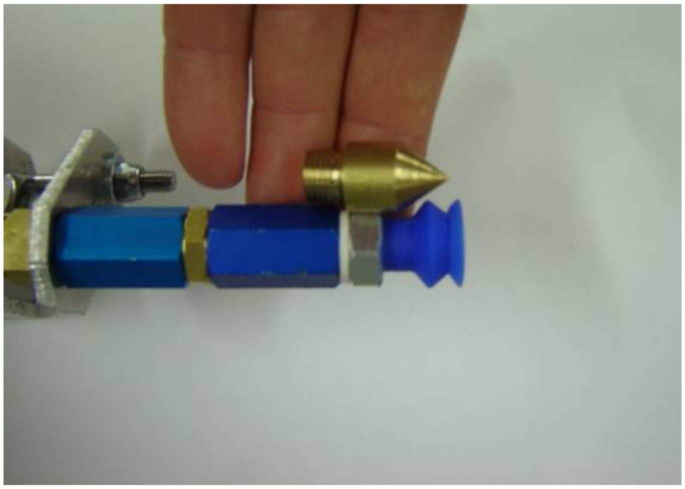


Figure 3-11 Brass spike and suction cup comparison

To prepare the extendable suction cup for calibration, it merely has to be unscrewed and the brass spike screwed on as shown in Figure 3-12.

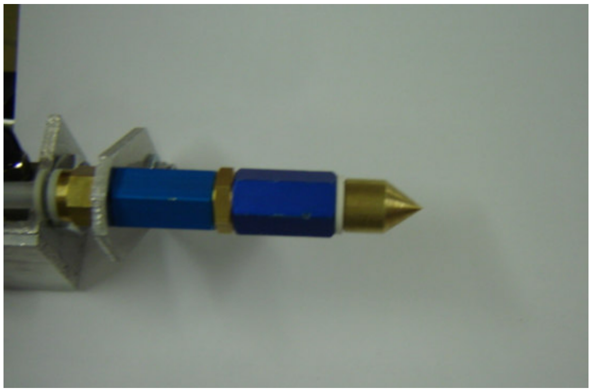


Figure 3-12 Brass spike as TCP

A fixed point in the robot's operational space was also needed to be able to calibrate the TCP. A second spike was machined which was used as the fixed point in space. It was then possible to calibrate the TCP by using the second spike as a reference point. The second spike used for calibration is shown in Figure 3-13.

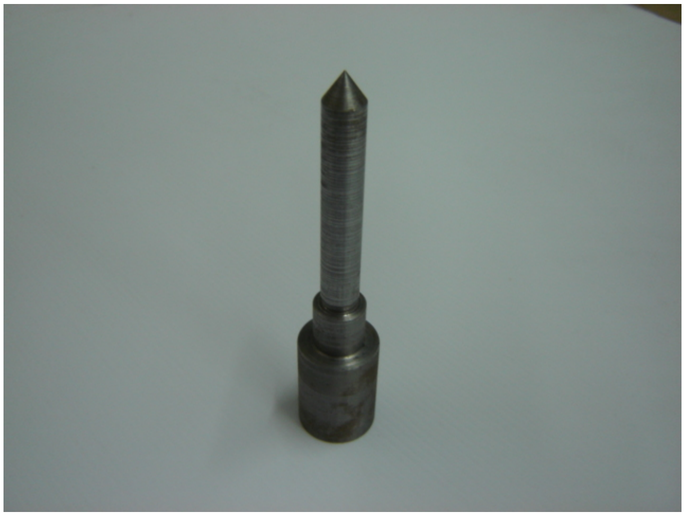


Figure 3-13 Second spike

3.3.4 Pneumatic Hardware

A pneumatic expansion module from FESTO was used to control the grippers, suction cup and cylinder. The expansion module's I/Os were hardwired to the robot's I/O module, namely the Beckhoff module. This gave the robot full functional control of the device. An air pressure of 6 psi (pounds per square inch) was established as a successful functional platform for all the pneumatic devices.

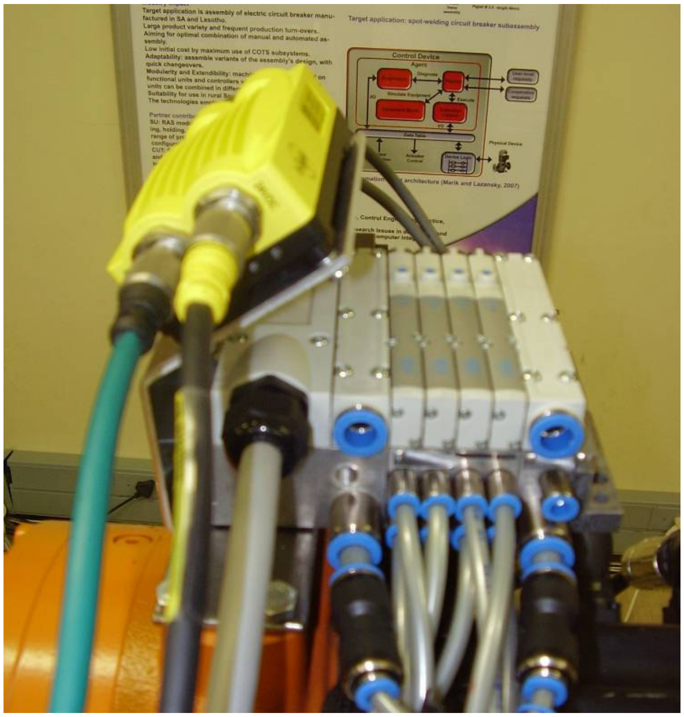

Target application: spot-welding circuit breaker subassembly
Control Device
Agent

Figure 3-14 Pneumatic valve terminal

3.4 Vision Hardware

3.4.1 Introduction to Vision Hardware

There are currently a great number of vision sensors (cameras) on the market and it is often difficult to select which sensor to use.

The remote In-Sight 5400R was chosen as it complied with the vision and network requirement. It is a very robust and compact instrument as shown in Figure 3-15, which is still able to deliver the required resolution at the maximum required trigger rate. The remote camera head was mounted onto the front of the KUKA robot, as the objective was to be able to position it very near the object being inspected, or in any orientation in relation to the object.

In the case of a robot-mounted sensor, any number of object features can be measured as long as the object can be approached by the single sensor head through the appropriate motions of the robot arm. Thus it is vital that the robot should be able to guide the sensor head to the target measuring position.

The robot then triggers the In-Sight sensor to take a snapshot of the object in question as soon as it passes directly beneath the camera head. Through KUKA Vision the robot receives the position data from the camera and is able to act accordingly, based on the requirements.

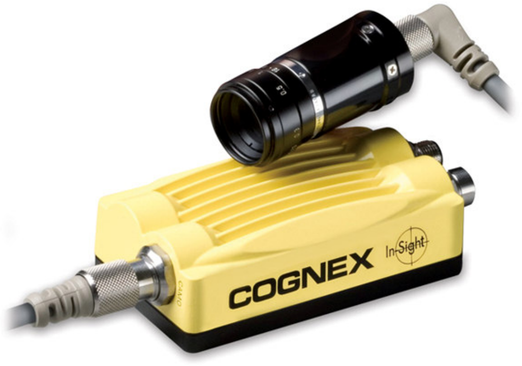


Figure 3-15 In-Sight remote camera head [3]

The advantages of a single robot-mounted sensor as shown in Figure 3-15, rather than fixed sensors, are greater flexibility and the use of fewer sensors, thus reducing costs and allowing operation where space is limited.

3.4.2 Vision Hardware Configuration

Figure 3-16 shows the entire hardware interface configuration of the COGNEX In-Sight system, which consists of:

A. COGNEX In-Sight 5400R sensor which does all the processing.
B. Closed-circuit television (CCTV) lens or any other lens that meets the requirements.
C. Extension ring 5 mm in width based on requirements.
D. The remote head camera comes in black and white or colour.
E. Camera cable which is attached to the In-Sight sensor.
F. Camera housing.
G. Transmission control protocol / Internet protocol (TCP/IP) cable allowing direct connection to the network.
H. TCP/IP cable for direct connection to a single personal computer (PC).
I. Connection to the network switch environment.
J. Access to the sensor over the network via other PCs.
K. Open cable connection for miscellaneous use.
L. Colour code of the cable and 24 V dc power connections.
M. Cable for connecting to the breakout or I/O module.
N. Breakout or I/O module.
O. Power supply connection to the breakout board.

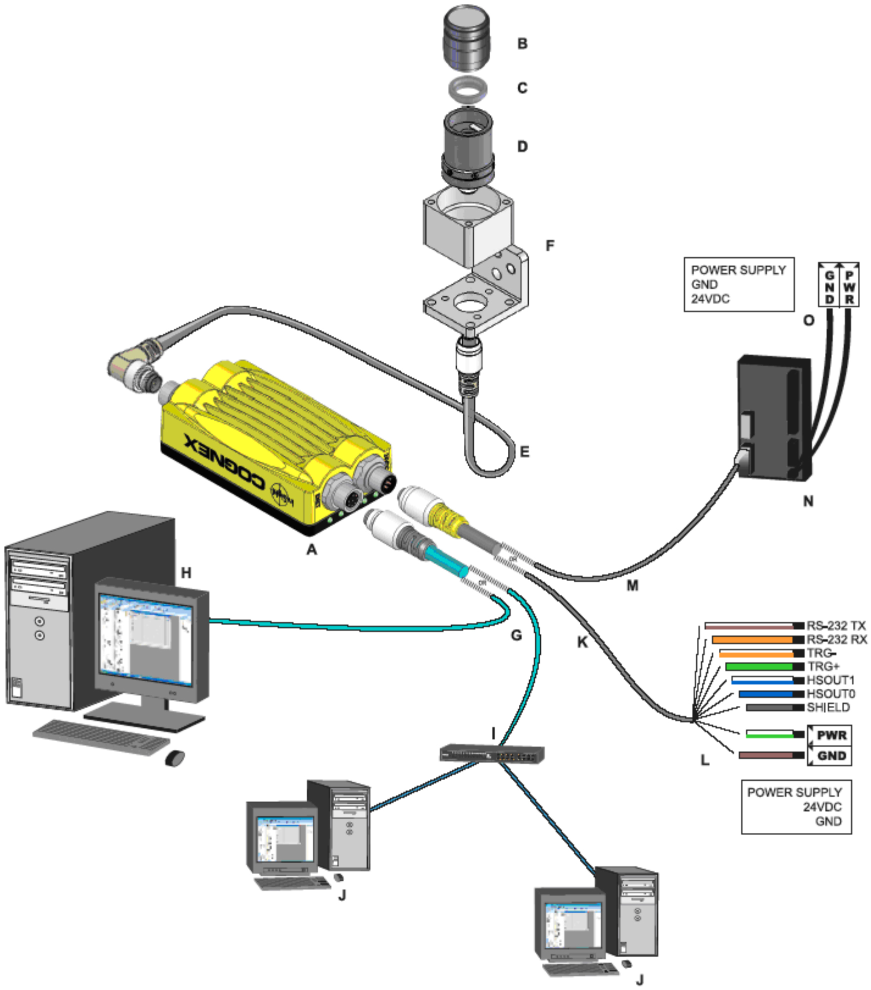


Figure 3-16 COGNEX hardware configurations [3]

3.4.3 Camera and Sensor Bracket

A mounting bracket had to be designed to mount the COGNEX In-Sight camera head on the robot. A mounting bracket for the camera head was manufactured using 3 mm angle aluminium. The mounting bracket was fixed onto the gripper to enable synchronised movement of the gripper and the camera to any possible inspection point by moving the robot to a new inspection position. The aluminium bracket shown in Figure 3-17 was designed in such a way that the camera head can be located closer to or further from the object it is inspecting by means of the mounting holes indicated by A. B in Figure 3-17 indicates a rotational point on the bracket to facilitate adjustment of the camera's view with respect to the relevant end effector.

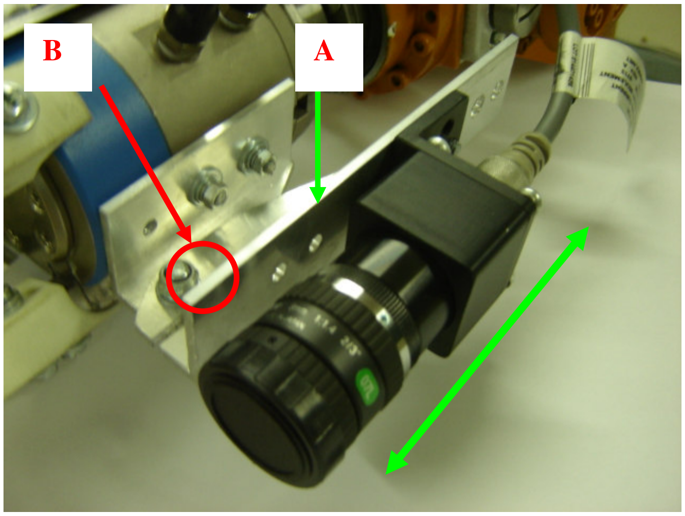


Figure 3-17 Camera head bracket

The COGNEX In-Sight sensor bracket was manufactured from a 3 mm aluminium plate which was bent to an angle of approximately 30 degrees as shown in Figure 3-18 and mounted on the KUKA robot. It was very important to ensure that the bracket would not limit the robot's motion.

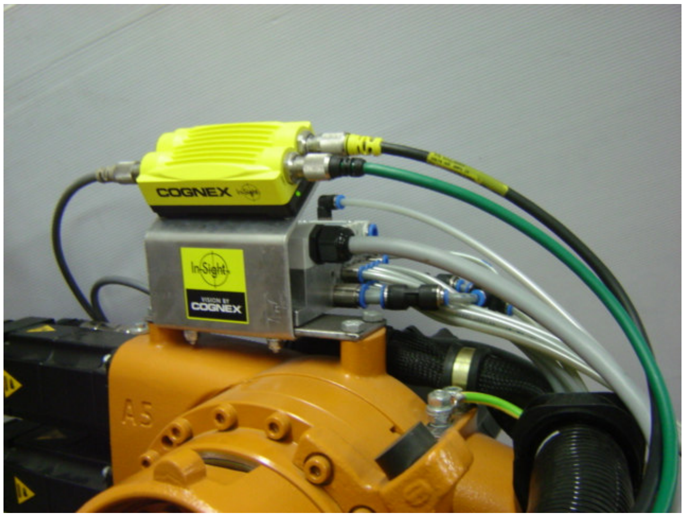


Figure 3-18 COGNEX In-Sight sensor bracket

3.4.4 Vision network environment

In-Sight 5000 series vision sensors are configured and their operation is monitored remotely from a networked PC running In-Sight Explorer using mouse and keyboard input [4]. Figure 3-19 illustrates the layout of the applications network environment used.

Figure 3-19

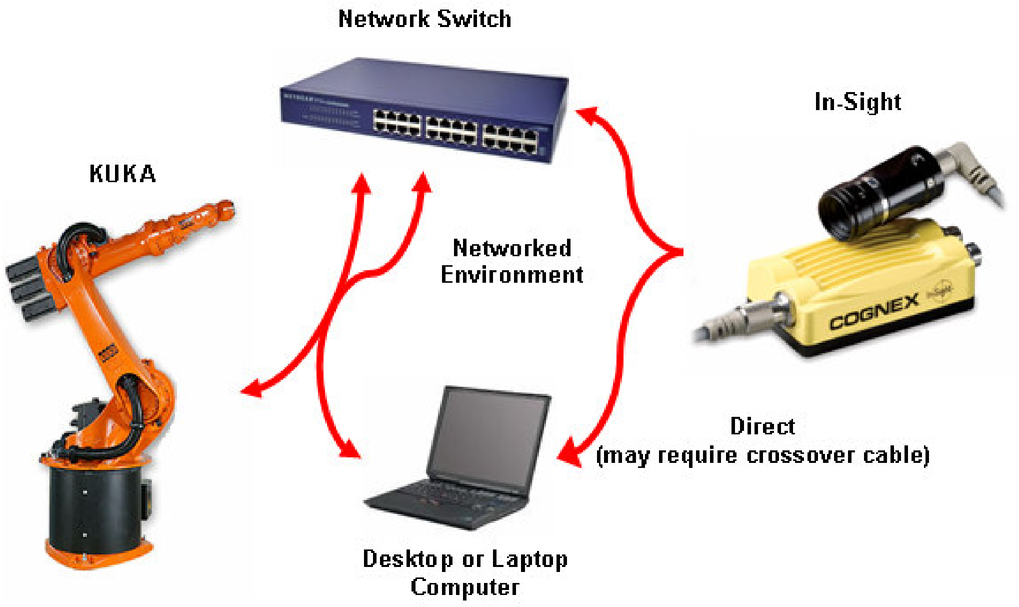
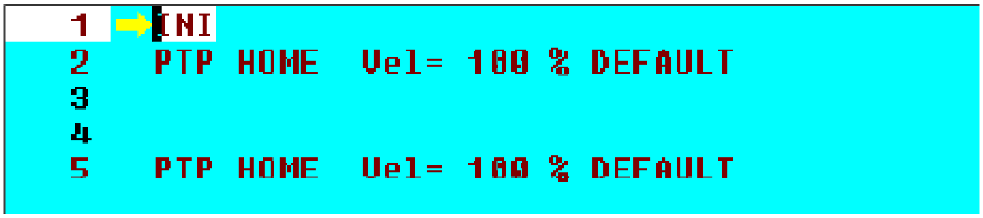


Figure 3-19 Network environment

3.5 Robot Software

3.5.1 KUKA Robot Control (KRC)

KUKA Robot Control is the software used to program the KUKA Robot Logic (KRL) to move the robot tool under program control to a point. It is also possible for logic commands and wait timers to be added to enhance the functionality of the robot program [5]. The standard structure of a new KRL program consists of an Initialisation (INI) section for declaring variables and two home positions at the beginning and end of the motion program (see Figure 3-20). The robot will thus start and end at the position from which it started executing the motion program.



```
INI
PTP HOME  Vel= 100 % DEFAULT


PTP HOME  Vel= 100 % DEFAULT
```

Figure 3-20 KRL [5]

The KUKA Control Panel (KCP) Figure 3-21 forms the Human Machine Interface (HMI) and is used to operate the KRL. All the elements required for programming and operator control of the robot system, with the exception of the main switch, are located directly on the KCP. The Liquid Crystal Display (LCD) screen is used to visualise operator and programming actions [5].

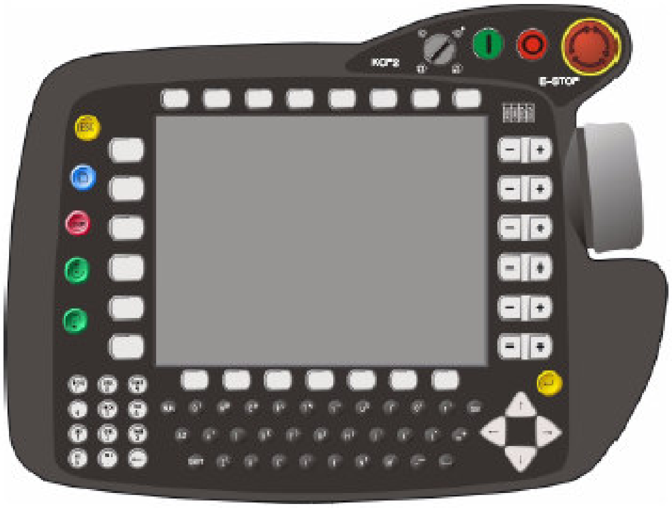


Figure 3-21 KCP [5]

3.5.2 KUKA Software Packages

KUKA provides a wide range of software packages that can assist in the successful implementation of robotic functions. A few options that were considered but not implemented were the following:

1 The KUKA OPC server is a software option for the KUKA controller which makes it possible to access system and user variables of the controller externally from anywhere in the network [6].

2 The technology package ConveyorTech is used for programming translational and rotational conveyor applications in the KUKA robot controller [7]. This allows robot motions to be co-ordinated or synchronised with linear or circular motions of conveyor systems.

3 KUKA Sim can be used to train employees safely and cost effectively right up to the expert programming level [8]. The optimal layout for manufacturing cells can be found, which allows verification of drafts and designs of a working cell at a development stage.

3.6 Interfacing Software

KUKA Vision is a software interface between KUKA HMI and a COGNEX In-Sight sensor [9]. The COGNEX In-Sight sensor is a vision sensor used for VGR applications. With KUKA Vision, the robot is able to control up to two COGNEX In-Sight sensors either through robot control or external triggering, and allows the following actions to be carried out:

- Trigger the COGNEX In-Sight sensors to acquire an image
- Select and edit jobs, i.e. the programs to be executed in the COGNEX In-Sight sensors
- Receive data from the COGNEX In-Sight sensors
- Read and write to any cell in a spreadsheet format.

Up to 120 variables (of data type Real) can be passed from each In-Sight sensor to the robot, allowing the robot to be programmed in such a way that the data received can be integrated with the robot logic.

The capability to read and/or write to any cell in the spreadsheet from the robot logic to the In-Sight sensor allows greater flexibility of the software.

The hand-held KUKA control panel (KCP) used to program the KUKA robot also allows the operator to tab between programmes, view and acquire images, search results, manually trigger the In-Sight sensor and edit routines in the In-sight sensor.

KUKA Vision supports the following In-Sight sensors with firmware 3.3 or above:
5100, 5400, 5400R, 5400S, 5403S, 5401 and 5403 [9].

KUKA uses a standardised interface for all vision systems, allowing simple handling during configuration and ease of use. Figure 3-22 shows this field of interfacing which KUKA has established.

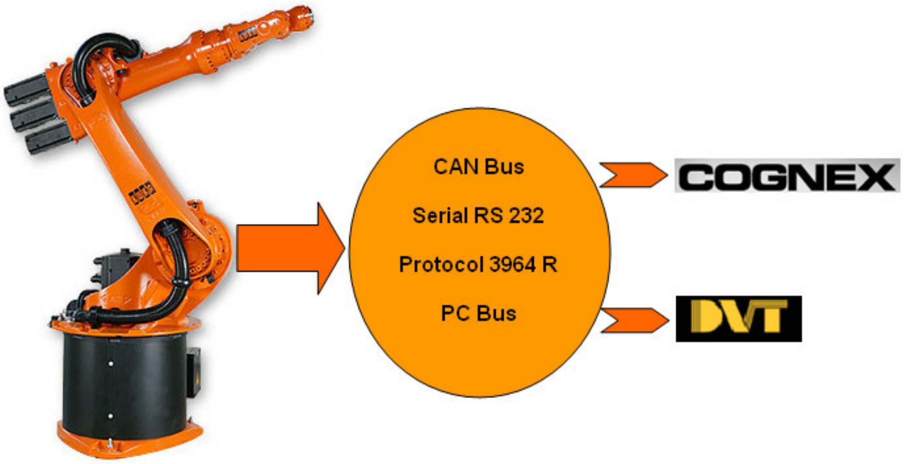


Figure 3-22 KUKA standard interface [9]

To obtain the fastest communication transmission between the KUKA robot and the In-Sight camera, a serial connection was initially chosen. However, under test conditions it was established that the TCP/IP connection would be the best choice to read and write data strings to and from the KUKA robot and the In-Sight module as there is a minuscule difference in the connection speed, and to a slightly larger degree ease of use. The interface options to the vision system are shown in Figure 3-23.

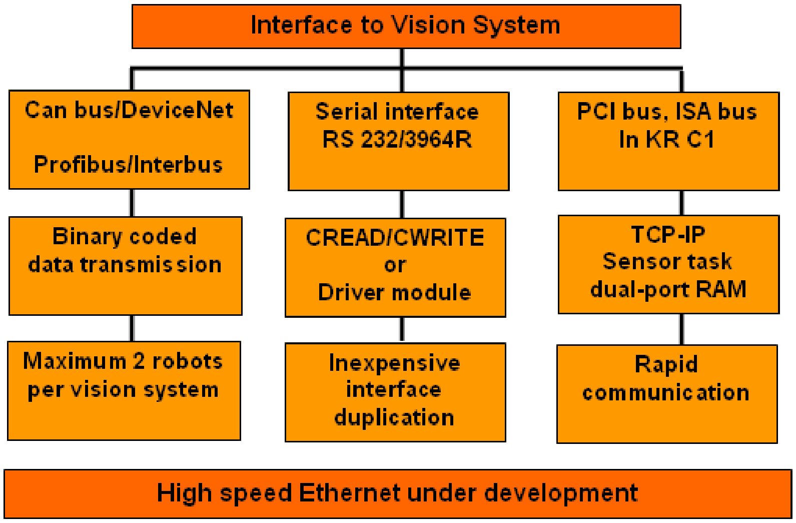
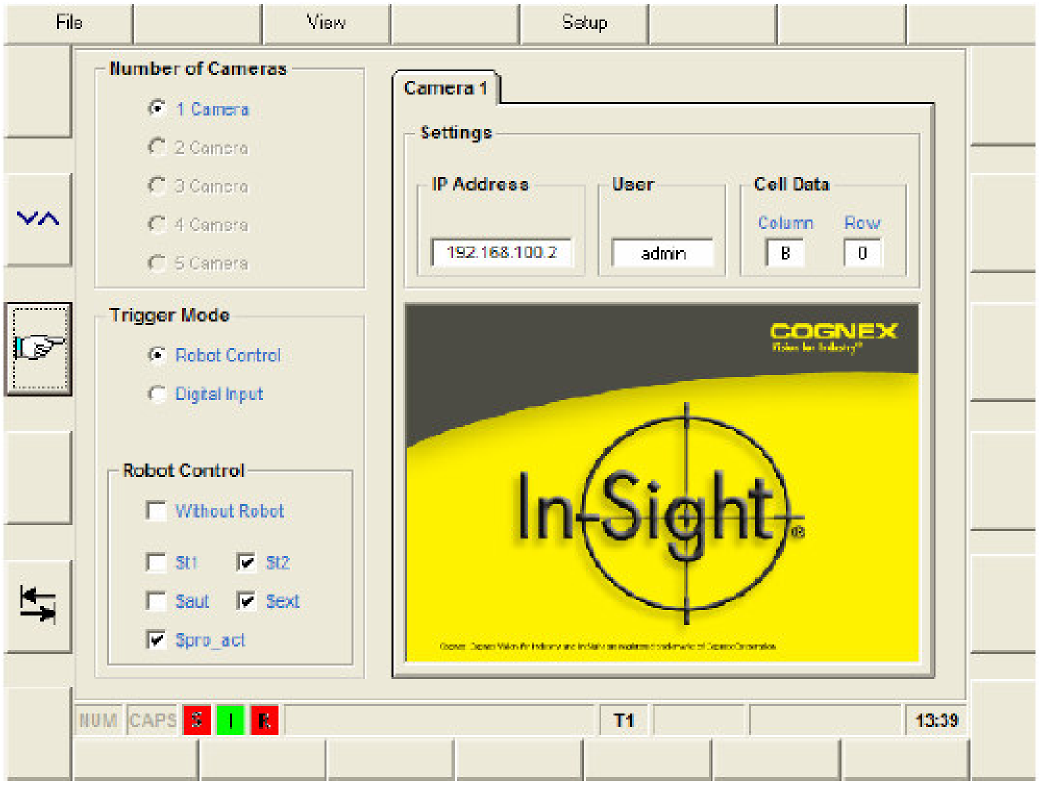


Figure 3-23 KUKA Interface to vision system [9]

It is important when running KUKA vision for the first time, as shown in Figure 3-24, to select the number of cameras and the triggering mode or robot control as the robot will trigger the camera or digital input in the way that an external trigger would be used to trigger the camera. The IP address of the camera specifies the user type and the cell location from where the required data can be accessed after inspection.



Number of Cameras
● 1 Camera
○ 2 Camera
○ 3 Camera
○ 4 Camera
○ 5 Camera

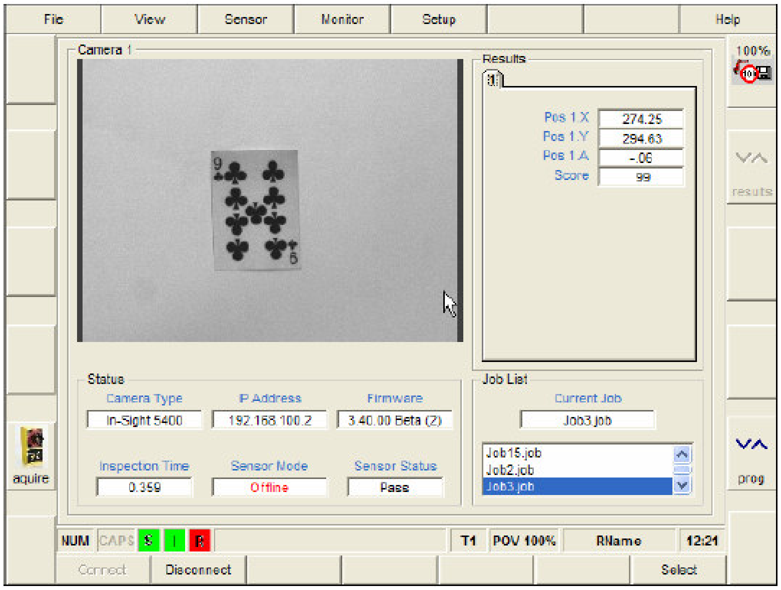
Trigger Mode
● Robot Control
○ Digital Input

Robot Control
□ Without Robot
□ St1 ☑ St2
□ Saut ☑ Sext
☑ Spro_act

Settings
IP Address: 192.168.100.2
User: admin
Cell Data: Column B, Row 0

Figure 3-24 KUKA Vision set-up program [9]

Once KUKA vision has been successfully set up and both the COGNEX In-Sight sensor and the KUKA robot have been correctly interfaced, a KUKA vision window will display the image captured as well as the co-ordinates of the object, the inspection time, pass or fail status and online/offline status as shown in Figure 3-25.



Results

Pos 1.X 274.25
Pos 1.Y 294.63
Pos 1.A -.06
Score 99

Status

Camera Type	IP Address	Firmware
In-Sight 5400	192.168.100.2	3.40.00 Beta (2)
Inspection Time	**Sensor Mode**	**Sensor Status**
0.359	Offline	Pass

Job List

Current Job: Job3.job

Job15.job
Job2.job
Job3.job

Figure 3-25 KUKA Vision displayed on KCP [9]

KUKA Vision was chosen as the interface between the KUKA HMI and the COGNEX In-Sight sensor. These two different product developers realised the need for interfacing software between the robot and the vision system and developed it. This assisted in the selection of the robot and the vision sensor. Figure 3-26 illustrates how the operator would view the images acquired by the COGNEX In-Sight sensor on the KCP.

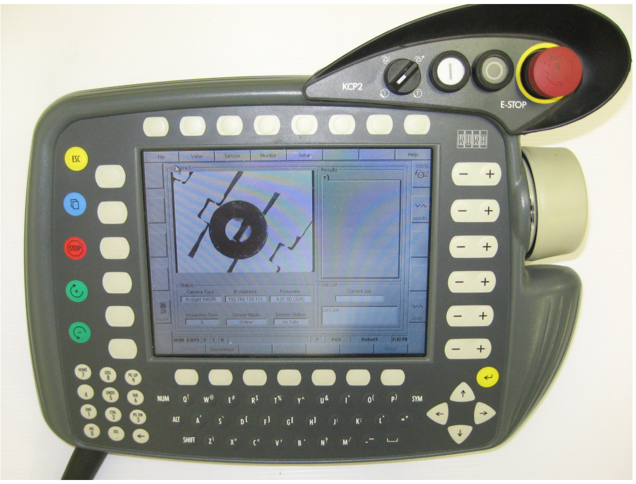


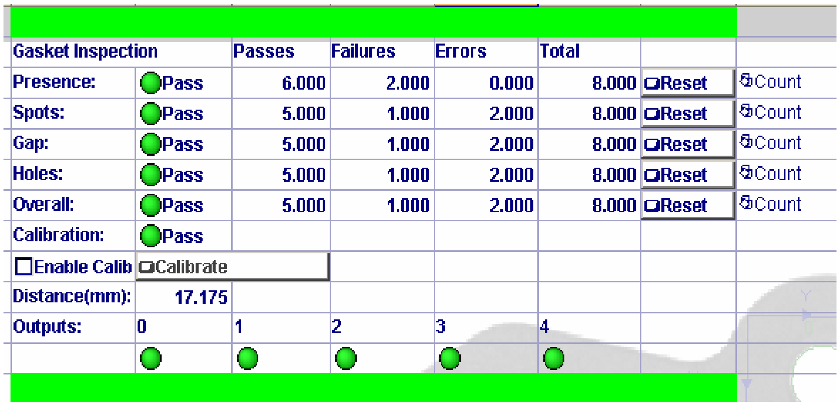
Figure 3-26 KUKA's HMI interface using KCP

3.7 Vision Software

3.7.1 In-Sight Explorer

COGNEX's vision software, In-Sight Explorer, is a spreadsheet development environment used for pattern recognition. A new and even more user-friendly product, In-Sight Easy Builder [10], was released in 2008.

In-Sight Explorer was used to locate and inspect parts on the moving or stationary conveyor. It is possible to write a program and design a user interface for the software to allow the operator quick and easy understanding of the vision sensor's actions which are taking place. The focus is therefore kept on areas of interest in the inspection, while all the complex activities taking place in the background are masked. Figure 3-27 displays a typical user interface.



Gasket Inspection		Passes	Failures	Errors	Total		
Presence:	● Pass	6.000	2.000	0.000	8.000	Reset	Count
Spots:	● Pass	5.000	1.000	2.000	8.000	Reset	Count
Gap:	● Pass	5.000	1.000	2.000	8.000	Reset	Count
Holes:	● Pass	5.000	1.000	2.000	8.000	Reset	Count
Overall:	● Pass	5.000	1.000	2.000	8.000	Reset	Count
Calibration:	● Pass						
☐ Enable Calib	Calibrate						
Distance(mm):	17.175						
Outputs:	0	1	2	3	4		
	●	●	●	●	●		

Figure 3-27 In-Sight Explorer User Interface [10]

Once image acquisition has been completed and if the object has passed inspection, the X, Y and Z location of the object is transferred to the robot via the KUKA Vision interface software, allowing the robot to move to the picking point wherever the object is located on the conveyor.

The user interface and images can then be viewed on the HMI on the control panel of the KUKA unit by means of the KUKA Vision interface software.

3.7.2 COGNEX OPC Capabilities

The In-Sight OPC server provides a simple way to configure a tag and determine the tags to be published to other OPC programs, such as common HMI and Supervisory Control and Data Acquisition (SCADA) packages [10].

The In-Sight OPC server provides tag configuration and publishing for any In-Sight vision sensor with declared job tags on a network attached to the PC. The server also provides a convenient way to monitor tag status and event logging for any attached vision sensors from a single program [10].

3.7.3 Sensor Calibration

Calibration is a vital part of commissioning the vision sensor. Not only is it vital for image acquisition, but also for axis-specific direction, which is in turn related to robot guidance. When acquiring an image through any lens, there will always be some distortion inherent in the specific lens. The most common distortion is radial distortion, as shown in Figure 3-28. It has the same effect as looking through a peephole. This distortion has to be removed from the image to enable reliable measurements to be made and to measure real-world units (see Figure 3-29).

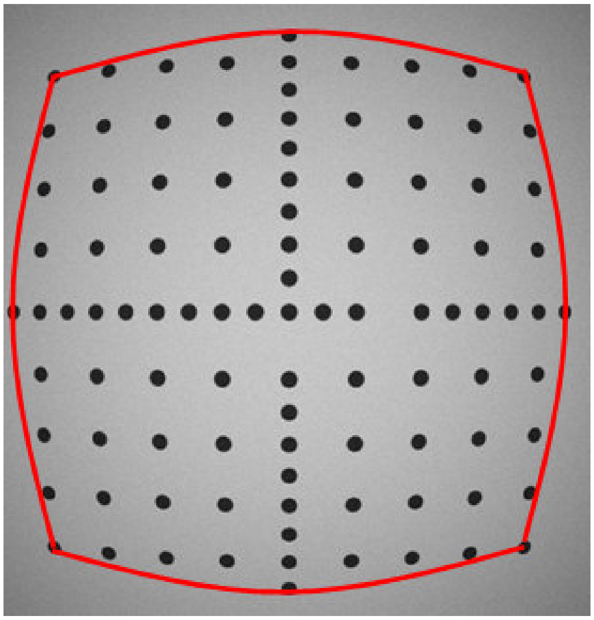


Figure 3-28 Radial distortion [11]

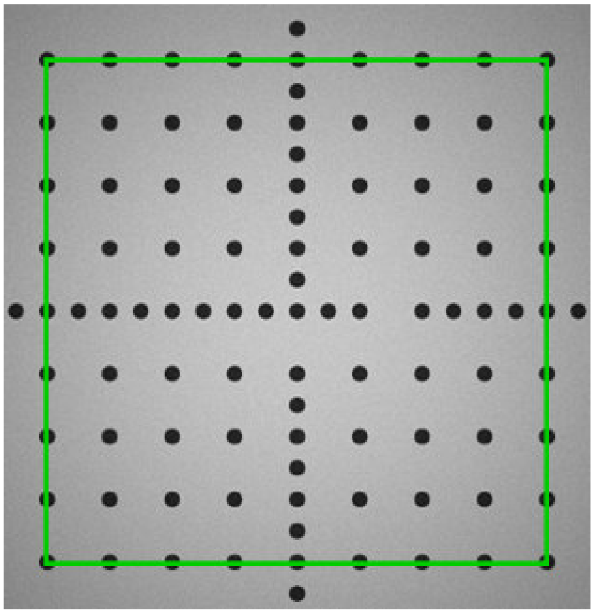


Figure 3-29 Undistorted image [11]

Another reason why calibration may be required is that if the camera is mounted perpendicular to the object being inspected, the object may be viewed from an angle, causing perspective distortion as shown in [11].

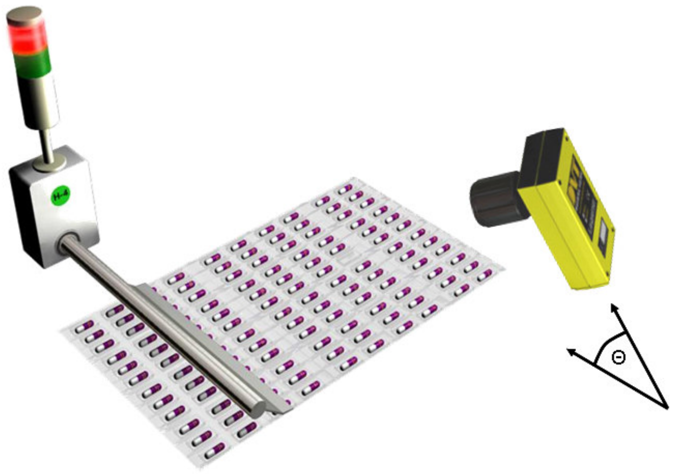


Figure 3-30 Perspective distortion [11]

To calibrate the sensor, various grids can be used, e.g. a grid of dots, a checkerboard pattern or a custom grid of dots. In prototyping a grid can be printed by a laser or inkjet printer, but for highly accurate calibration grids a professional organisation must be found that specialises in grids.

The In-Sight calibration wizard was used to calibrate the sensor. The checkerboard grid with a fiducial is a grid with a graphic design to help specify the X and Y co-ordinates automatically. The checkerboard grid with a fiducial in Figure 3-31 is used by the vision sensor to specify the co-ordinates automatically.

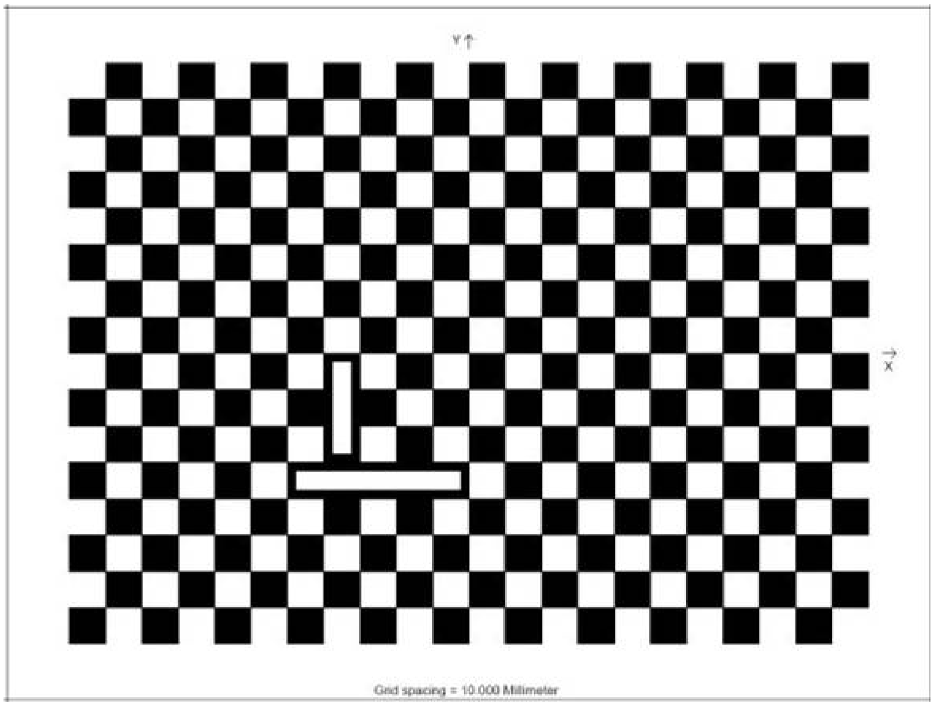


Figure 3-31 Checkerboard grid [11]

3.8 Lighting

3.8.1 Introduction to Lighting for Photographic purposes

Lighting is one of the most critical factors when it comes to image recognition. Without the correct light intensity or angle, components are very difficult to identify or recognise. Without a good image, the vision task may be very difficult, if not impossible, to carry out. Lighting is considered to be an art as there are various ways of illuminating an object for photographic purposes.

3.8.2 Purpose of lighting

The purpose of lighting is to:

- Enhance the contrast of the features of interest of an object
- Make the foreground and background distinctly different grey values

- Minimise shadows
- Freeze the motion of moving objects by taking snapshots with pulsed light
- Increase the sharpness of edges
- Remove glare.

This is done with the type of lighting selected and the way in which it is used [12].

3.8.3 Lighting Terminology

Specular reflection is caused when light rays are redirected at the same angle as they strike a shiny surface, making it difficult to view the object's features. There are three main problems to avoid when working with light [12]:

- Collimated light: the light rays are parallel, which cause glare on highly reflective surfaces
- Diffuse light: the light rays come from many different angles
- Direct light: this is somewhere between collimated and diffused light [12]

3.8.4 Types of Light Sources

There are many different types of light sources available today, including natural light, which can help to eliminate problematic optical characteristics in the working cell environment. The problem is to ensure that the right light source at the correct level of lighting is applied to the working cell environment at the correct angle. In Table 2 some of the different options available are shown.

Table 2: Various lighting options

Type	Halogen	Incandescent	Fluorescent	Laser	LED	Xenon flash
Direct light source	X					
Diffuse light source		X	X			
Structured light source				X		
Can be strobed				X	X	X
Cannot be strobed	X	X	X			
Expensive				X		
Cheap		X				
Safety issue				X		
Degrades over Time	X		X			X

3.8.5 Lighting Hardware Applied

3.8.5.1 Introduction to the Application of Lighting

As lighting is a very important consideration when acquiring images, and since total enclosure of the robot's environment was not a consideration at all, some time was spent determining which lighting method would be suitable for this application. The aim was to establish a flexible lighting method which could be applied in industry, and which would allow the system to function correctly without being influenced by ambient light.

3.8.5.2 Ring Light

A 12 V dc diffused ring light was the first choice for testing as it can illuminate the object to be inspected through 360 degrees from an elevated position. The camera acquires the image through the centre hole of the ring light as shown in Figure 3-32.

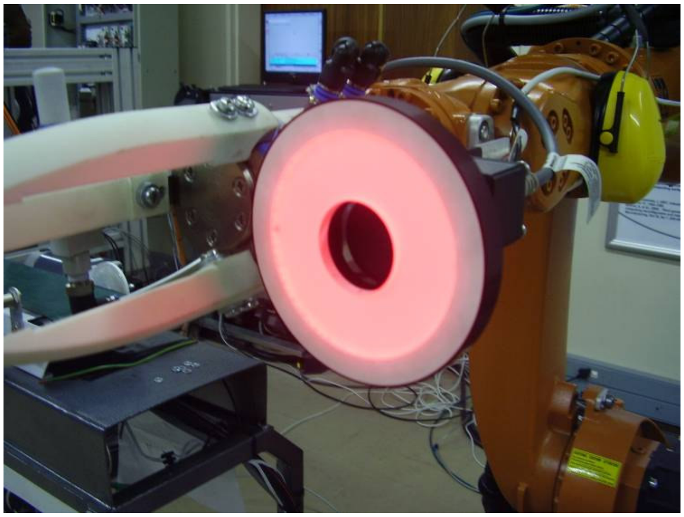


Figure 3-32 Ring light

A bracket was made for the ring light from 2 mm aluminium plate Figure 3-33 to enable it to be mounted on the camera bracket. The reason is that when the camera position is adjusted, the ring light position is adjusted simultaneously as it is fixed to the camera bracket which is mounted on the gripper.

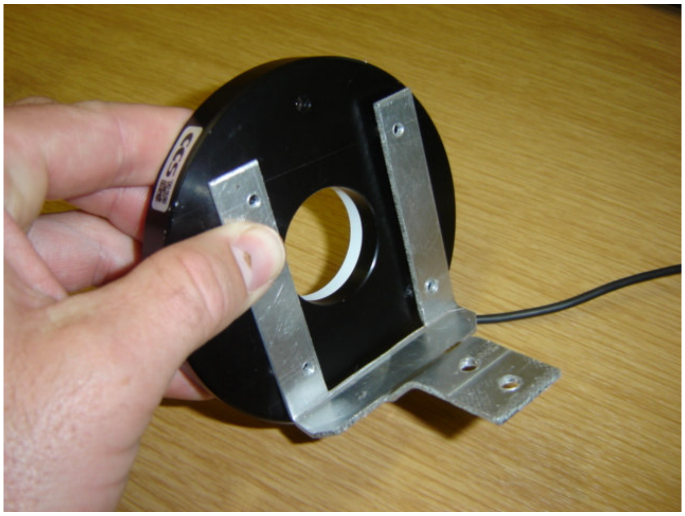


Figure 3-33 Ring light Bracket

It was soon discovered that light was being reflected off the glossy surface of the conveyor back into the lens Figure 3-34. This led to the following problems with image acquisition:

- The object was over-illuminated.
- The acquired images were difficult to process.

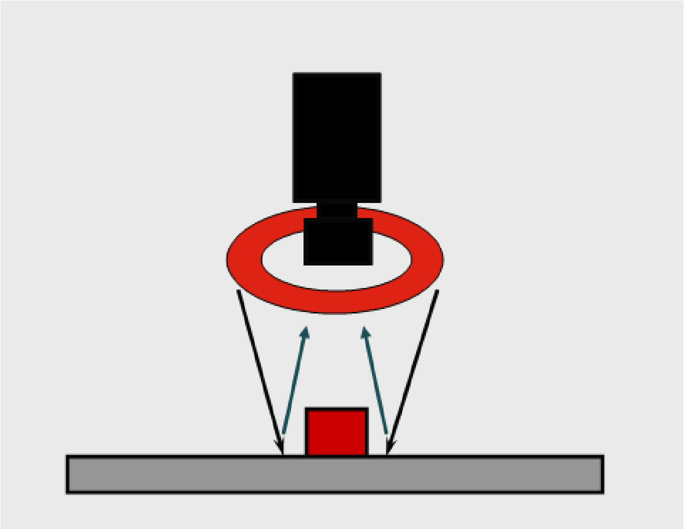


Figure 3-34 Ring light effect

3.8.5.3 Linear Direct Side-Lighting

A system of linear direct side-lighting was subsequently developed and assessed as it illuminated the edge features of objects and allowed a crisper image of the object to be acquired. It also resulted in the total removal of shadows as shown in Figure 3-35, while at the same time reducing reflected light off the shiny surface of the conveyor back into the lens. For this application one hundred super-bright red light-emitting diodes (LEDs) were used, fifty of which were fitted on either side of the conveyor system.

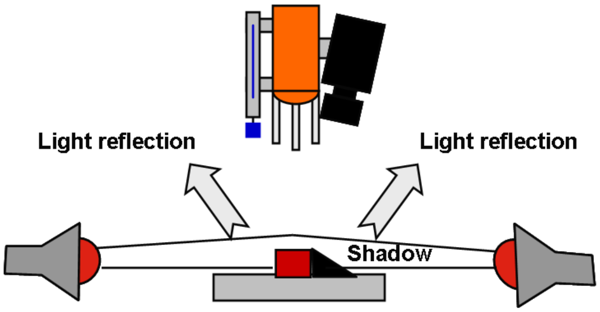


Figure 3-35 Linear direct side-lighting

Fifty LEDs were soldered in parallel and mounted onto a side bracket which could be fixed to the side of the conveyor. This greatly improved the quality of the acquired image and enhanced the edge features of components as shown in Figure 3-36. It gave a crisper image for processing.

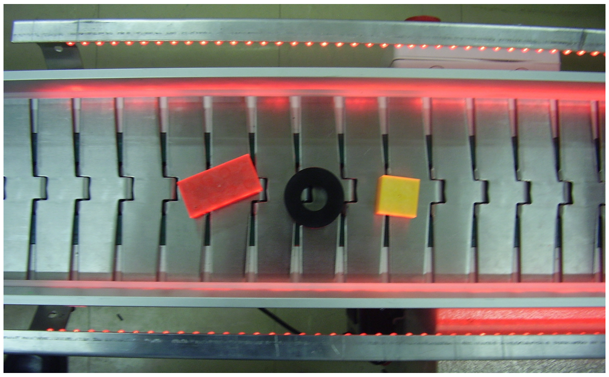


Figure 3-36 Direct lighted objects

3.9 Lenses

3.9.1 Introduction to Lenses

The choice of lens will depend on the specific application. For example, magnifying lenses may be needed for inspecting tiny parts. Different working distances (the distance between the camera lens and the inspection surface) will require different focal length lenses. Telecentric lenses may be needed to correct any perspective errors. Major optics and lens manufacturing companies provide a range of lenses.

3.9.2 Factors in Lens Selection

The following factors must be taken into consideration when a lens is selected for an application:

- Field of view:

  Dependent on focal length of lens

  Dependent on size of part to be inspected

- Depth of field:

  A factor when inspecting three-dimensional parts

- Object distance:

  May be pre-determined by machine design

- Feature contrast required:

  Indicated by the number of alternating black and white lines per millimetre a lens can distinguish, i.e. Modulation Transfer Function (MTF)

- Overall lens quality (ability to produce undistorted images across most of the field of view (FOV) [13].

3.9.3 Types of Lenses

Once the factors in lens selection have been considered, the type of lens to be used for the application must be looked at.

- Wide angle lens:

  Focal length is less than the standard 50 mm, e.g. 16 mm

  Great depth of field, closer focusing distance

- Telephoto lens:

  Focal length greater than the standard 50 mm, e.g. 75 mm

  Shallow depth of field; enlarges distant objects

- Zoom lens:

  Variable focal lengths, e.g. 35–70 mm

- Macro lens:

  Size of image = size of object (1:1) [13].

3.9.4 CCTV Lenses

CCTV Lenses are the preferred type, shown in Figure 3-37, as they have the following characteristics:

- Low cost
- May require extension tubes for machine vision
- Available in a variety of focal lengths
- Good for low light level applications
- Good for part identification
- Not good for high-accuracy gauging.

Figure 3-37: Various CCTV lenses [13]

3.9.5 Telecentric Lenses

Telecentric lenses are the preferred choice when very high-quality image acquisition is required and if any of the following lens characteristics are a factor:

- Medium priced to costly
- Uniform image illumination
- Optically invariant magnification
- Constant perspective across field of view
- Final lens element is size of field of view
- Good for measuring at varying distances.

Figure 3-38: Telecentric lenses [13]

3.9.6 Conventional versus Telecentric lenses

Conventional lenses have a perspective error: if there is a change in the distance to the object, there has to be a change in the magnification as well, as shown in Figure 3-39.

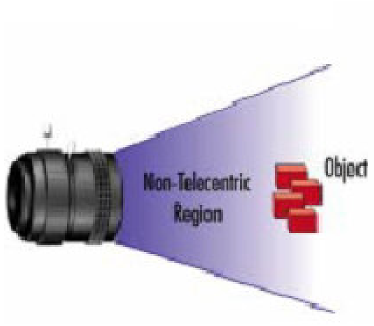


Figure 3-39: Conventional lens [13]

Telecentric lenses (Figure 3-40), on the other hand, provide constant magnification when there is a variation in distance to the object with no perspective error.

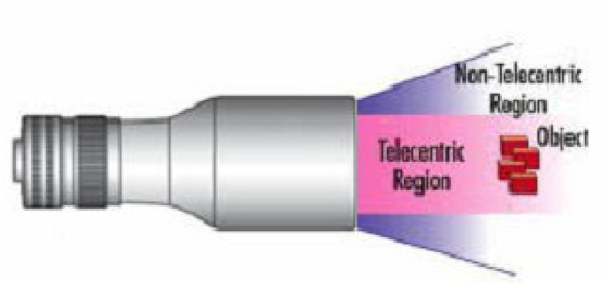


Figure 3-40: Telecentric lens [13]

3.9.7 Lens selected

A CCTV lens was selected for this project as the pick-and-place application did not require a telecentric lens and the CCTV lens functioned adequately.

3.10 External Environment

3.10.1 Programmable Logic Controller (PLC)

3.10.1.1 Introduction to PLC

Control engineering has developed over time. In the past, humans were the main controllers of a system. In more recent times electricity began to be used for control. Early electrical control was based on relays, which allow the power to be switched on and off without a mechanical switch. Relays are commonly used to make simple logical control decisions. The decrease in the cost of computers has enabled the development and widespread use of the Programmable Logic Controller (PLC). The PLC began to be used in the 1970s, and has become the most common choice for manufacturing equipment controls [14].

3.10.1.2 Methods of Communication with the PLC

The following methods were considered for communication to and from the PLC:

- Point-to-Point – RS-232 communication port configurable for direct connection to the programming device.

- Ethernet/IP – RS-232 communication port configurable for Ethernet/IP communication through an EtherNet/IP-to-RS-232 interface. The EtherNet/IP-to-RS-232 interface module supports program monitoring and upload/download, data collection from PCs or mainframes, and controller peer-to-peer communication. It can also send e-mail messages via SMTP (Simple Mail Transfer Protocol), and can display status and configuration information through a built-in Internet server.

- DeviceNet – RS-232 communication port configurable for communication through a DeviceNet interface allows the controller to function as a slave node in a DeviceNet network or as a peer to other controllers.

- DF1, half-duplex – RS-232 communication port configurable for DF1 half-duplex slave protocol for connection to a modem in SCADA applications [14].

3.11 References

[1] KUKA kr_6.PDF [Online]. Available from: http://www.kuka.com [Accessed 4 April 2008].

[2] Beckhoff, DeviceNet Coupler Technical Document, Version 1.3, 2006, p6.

[3] COGNEX Vision Systems [Online]. Available from: http://www.cognex.com/ProductsServices/VisionSystems/InSight [Accessed 4 March 2008].

[4] Installing In-Sight 5000 Series Vision Sensors [Online] Available from: http://www.cognex.com [Accessed 4 April 2008].

[5] KUKA\ Software_KSS.PDF [Online] Available from: http://www.kuka.com [Accessed 4 April 2008].

[6] KUKA OPC server.PDF [Online] Available from: http://www.kuka.com [Accessed 4 April 2008].

[7] KUKA Software conveyor_r31.PDF [Online] Available from: http://www.kuka.com [Accessed 4 April 2008].

[8] KUKA Sim.PDF [Online] Available from: http://www.kuka.com [Accessed 4 April 2008].

[9] KUKA [1].Vision V8.PDF [Online] Available from: http://www.kuka.com [Accessed 4 April 2008].

[10] COGNEX In-Sight_Brochure.PDF [Online] Available from: http://www.cognex.com [Accessed 4 March 2008].

[11] COGNEX CALIBRATION [Online] Available from: http://www.cognex.com [Accessed 4 April 2008].

[12] COGNEX In-Sight-lighting [Online] Available from: http://www.cognex.com [Accessed 4 March 2008].

[13] COGNEX In-Sight-lens [Online] Available from: http://www.cognex.com [Accessed 4 March 2008].

[14] HUGH JACK. Automating Manufacturing Systems with PLCs, Version 4.7, 14 April 2005, p20.

4 Chapter 4: System Calibration and Set-up

4.1 Vision Sensor

4.1.1 Installing the Sensor

This section describes how the camera head was connected to the in-sight vision sensor hardware:

1 The remote camera head was mounted as indicated (see Figure 4-1).

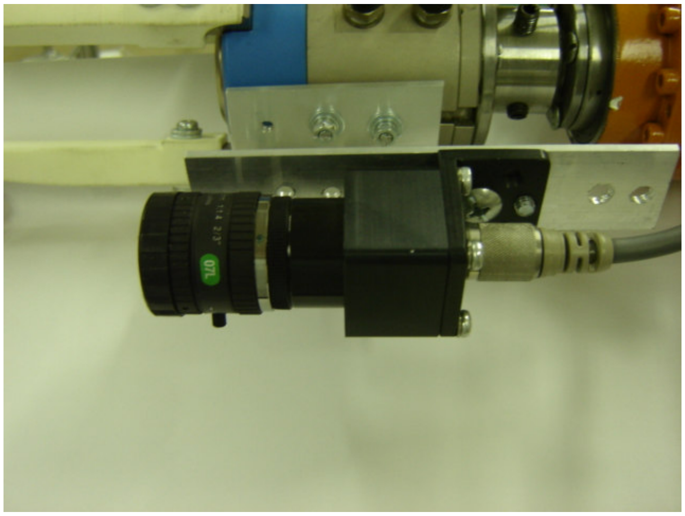


Figure 4-1 Remote Camera Head

2 A male M12 camera cable was connected to the In-sight sensor's CAMO connector.

3 The Ethernet cable for network communication was connected to the sensor's female M12 ENET connector shown in Figure 4-2 and then to the network switch or directly to a PC.

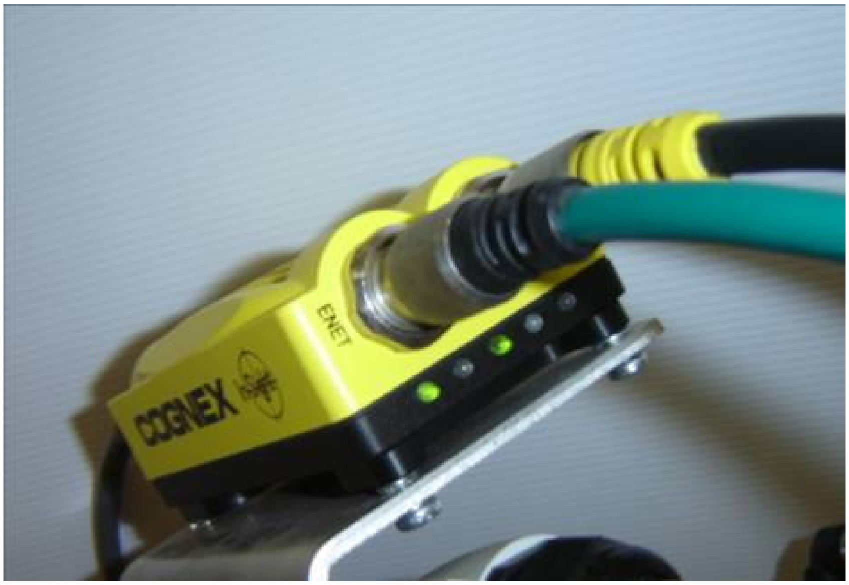


Figure 4-2 ENET and 24 V dc connector

4 The power supply breakout cable was connected to the 1350 breakout module shown in Figure 4-3. This module is the conventional connection method for power, serial communication and I/O lines.

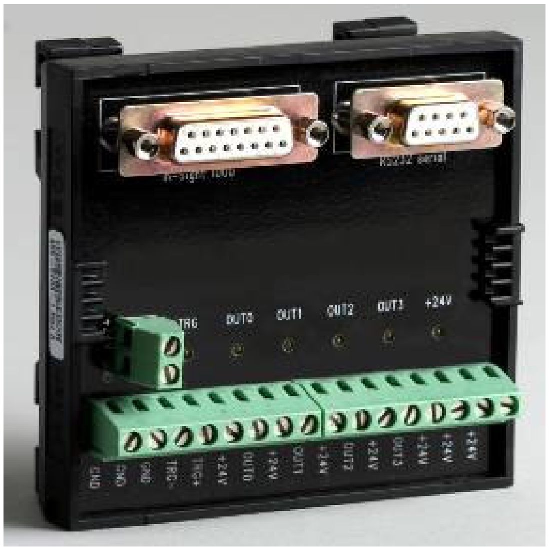


Figure 4-3 Breakout 1350 module [1]

5 When the sensor is powered up (Figure 4-4), User 0 LED and User 1 LED both turn on momentarily. Then User 0 LED turns off and User 1 LED stays on. Next, User 0 LED turns on and User 1 LED turns off. Finally, both LEDs momentarily light up and then turn off [1].

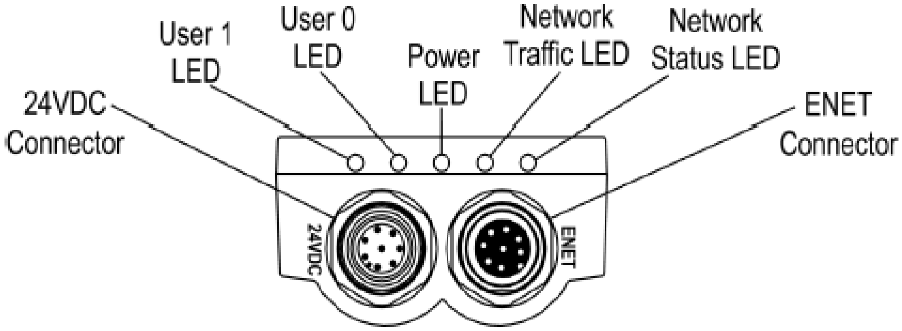


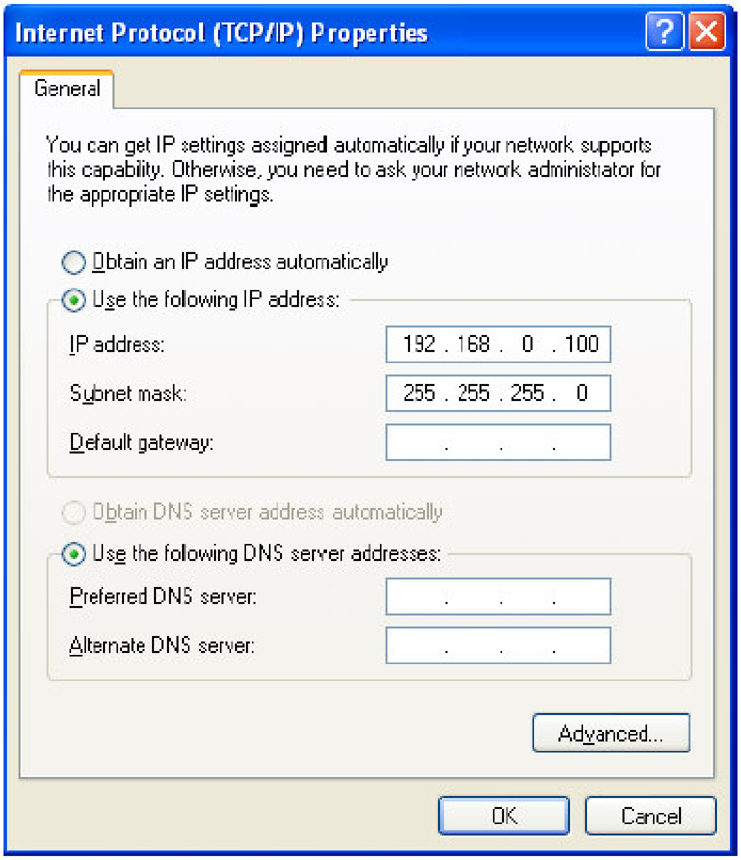
Figure 4-4 Vision sensor indicator LEDS [1]

Table 3 Vision sensor indicators and functions [1]

Indicator	Function
User 1 LED	Green when active. User configurable using Discrete Output Line 4 (Line 10 when using I/O expansion module)
User 0 LED	Red when active. User configurable using Discrete Output Line 5 (Line 11 when using I/O expansion module)
Power LED	Green when power is applied
Network Traffic LED	Flashes green while transmitting and receiving data
Network Status LED	Green when network connection is detected

4.1.2 Configuring Network Settings

Once the COGNEX in-sight explorer software had been installed, the network settings were configured to connect to the In-Sight 5000. An appropriate IP address and an appropriate subnet mask had to be entered. The subnet mask defines which part of the system's IP address refers to the network and which part refers to the host. The network part of the IP address is the same for all hosts on the same subnet, and the remainder is unique to each host as shown in Figure 4-5 [1].



You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administiator for the appropriate IP settings.

○ Obtain an IP address automatically

● Use the following IP address:

IP address: 192 . 168 . 0 . 100

Subnet mask: 255 . 255 . 255 . 0

Default gateway: . . .

○ Obtain DNS server address automatically

● Use the following DNS server addresses:

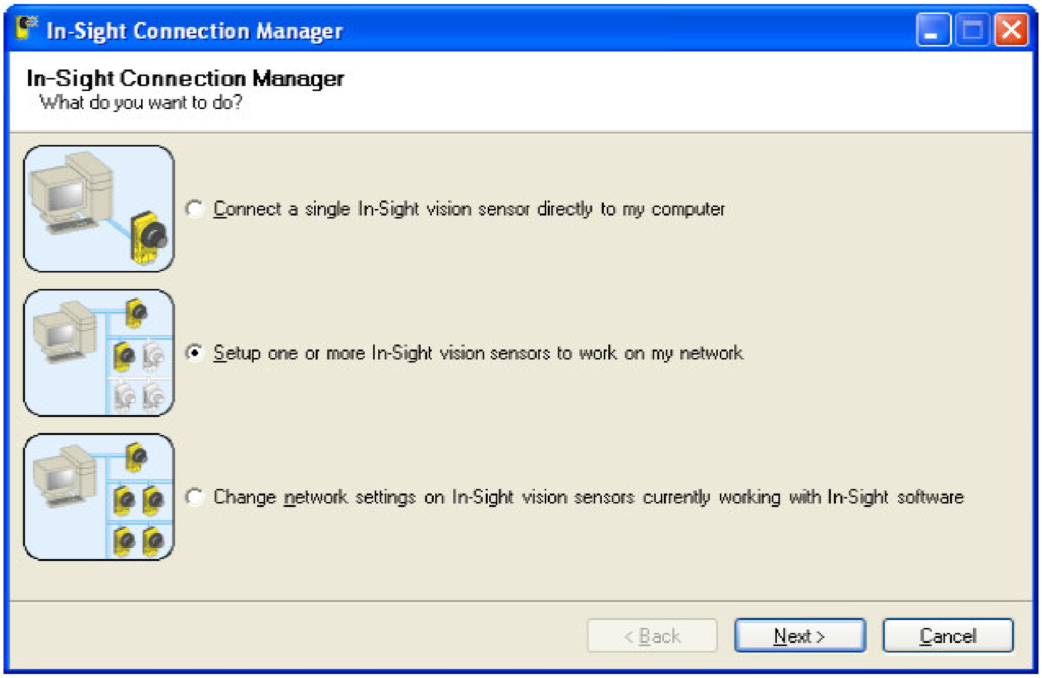
Preferred DNS server: . . .

Alternate DNS server: . . .

Figure 4-5 Network protocol dialog [1]

4.1.3 Adding the Sensor to the Network

After the in-sight connection manager is used to configure the sensor's network settings, the vision sensor must be connected to the network and powered up. The in-sight connection manager will prompt the user to follow the configuration procedure as shown in Figure 4-6.



In-Sight Connection Manager

In-Sight Connection Manager
What do you want to do?

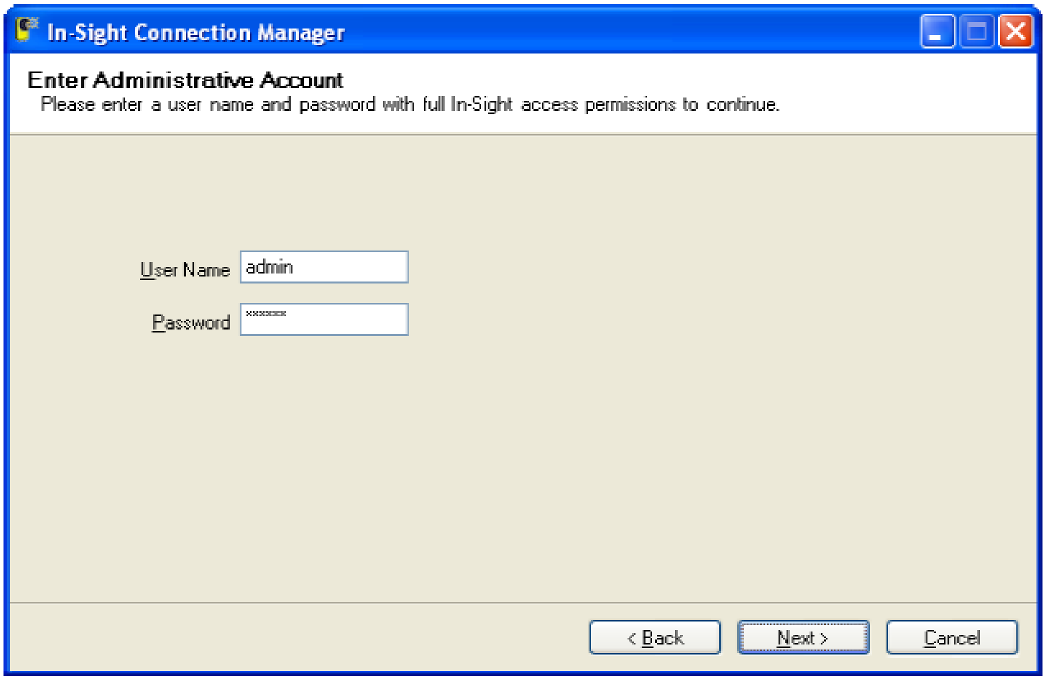
○ Connect a single In-Sight vision sensor directly to my computer

● Setup one or more In-Sight vision sensors to work on my network

○ Change network settings on In-Sight vision sensors currently working with In-Sight software

Figure 4-6 In-Sight Connection Manager [1]

It is important to add an administrative account to allow full access to the sensor when logging on as shown in Figure 4-7.



In-Sight Connection Manager

Enter Administrative Account

Please enter a user name and password with full In-Sight access permissions to continue.

User Name: admin

Password: *******

Figure 4-7 Administrative Account [1]

4.1.4 Logging on to the Sensor

After the sensor has been added to the network it is possible to log on by using the valid User Name and Password.

Each sensor is pre-configured with three User Names: Admin, Monitor and Operator; each of these accounts is configured with a blank password. Each User Name is assigned a specific Access Level. The Access Level controls how much interaction is allowed for the current user to prevent inadvertent or unauthorised changes to the configuration as follows [1]:

- **Admin Level** (Full): The user has complete, unrestricted access to the In-Sight sensor
- **Operator Level** (Protected): The user has limited access to the sensor

- **Monitor Level** (Locked): The most restricted level of access available. A user in Locked mode can only monitor the operation of the current sensor [1].

4.1.5 Sensor Calibration

For the robot application, the X, Y and origin co-ordinates were specified manually as this method makes calibration much more flexible. Thus a grid of dots was used with no fiducial (which indicated the X, Y and origin for the sensor). Using the calibration wizard, the co-ordinates were specified for the vision sensor as shown in Figure 4-8.

Once the sensor was calibrated it was very important to calibrate the robot according to the same co-ordinates as those of the sensor. If the same co-ordinates were not used, calibration of the sensor and robot would differ, causing orientation problems during operation.

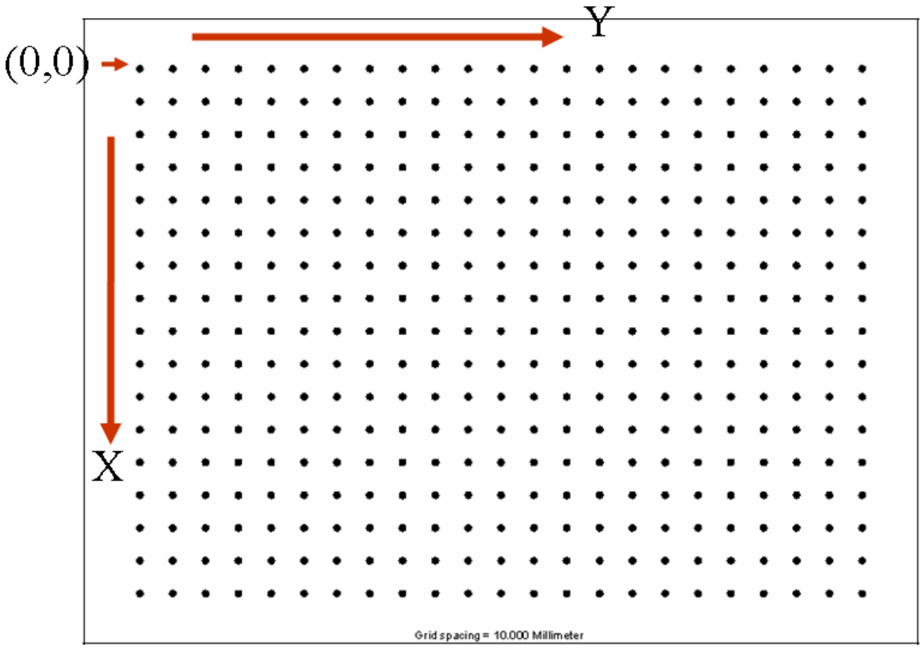


Figure 4-8 Grid of dots [2]

Figure 4-9 illustrates the axis directions for the pick-and-place application. A view of the system environment from above is shown. This method of calibration helps to relate the axis to real-world co-ordinates since it is then easier for people to understand and relate to it.

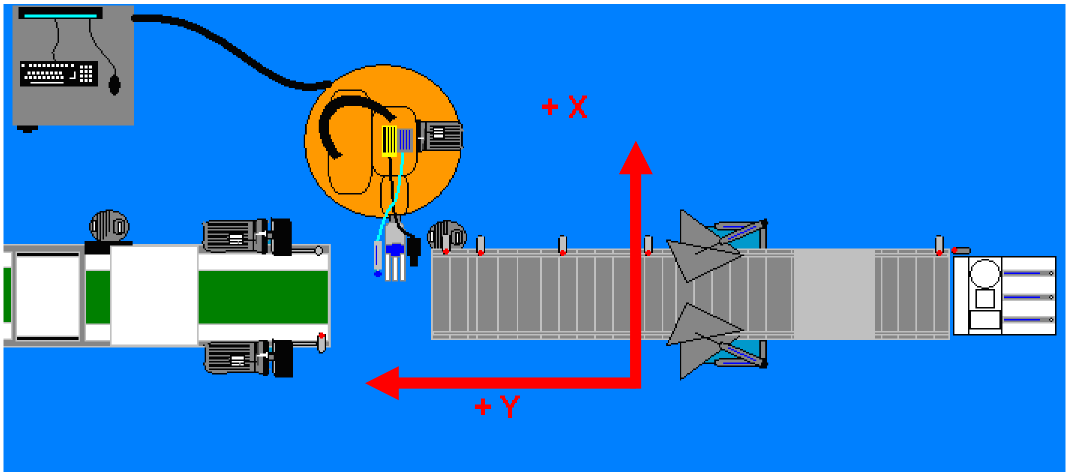


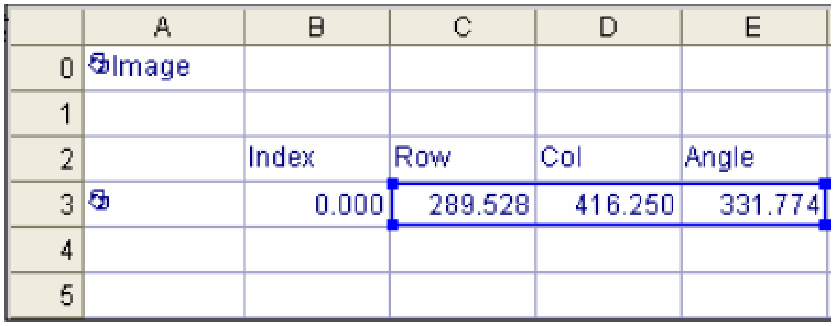
Figure 4-9 View of system from above

4.1.6 FormatString

A FormatString dialog was used to effortlessly format referenced cell data and generate an Output String that can be sent to the robot via the interfacing software from the vision sensor. Within the FormatString dialog, each referenced cell is listed as an individual item. Each item can be formatted by assigning to it a Label, Data Type, number of Decimal Places, Fixed Width and Pad. To format an item, it must first be highlighted in the list. The entire Output String can be formatted by adding Leading Text, Trailing Text, Terminators and Delimiters [3].

The Find Patterns function was used to find the position and angle of the object, first on a stationary conveyor and later while the conveyor was in motion (Figure 4-10). Cell B19 was selected as the empty cell in the In-Sight Explorer software where the

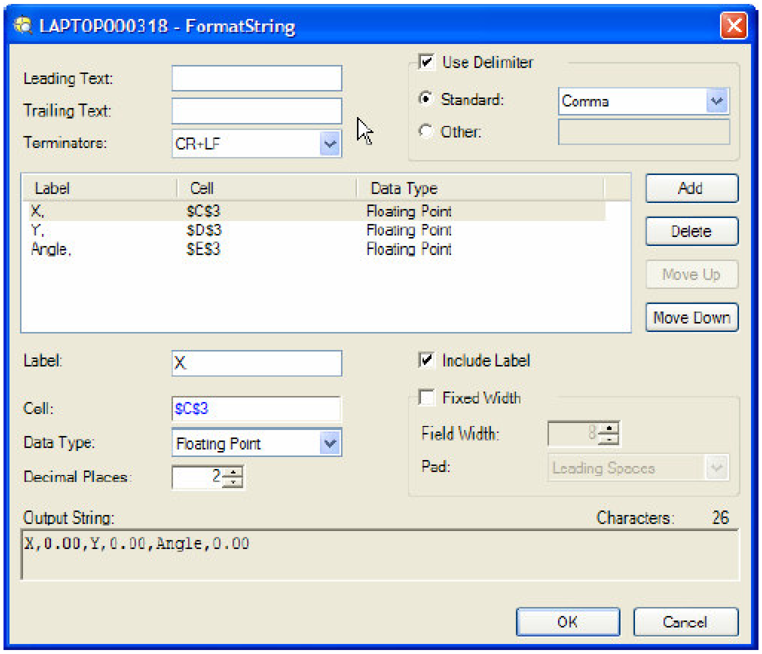
FormatString function references this data and returns an Output String that can be read by the KUKA Vision interfacing software and converted to data readable by the robot.



	A	B	C	D	E
0	Image				
1					
2		Index	Row	Col	Angle
3		0.000	289.528	416.250	331.774
4					
5					

Figure 4-10 In-Sight Explorer program

Once the reference cell B19 has been accepted, the row (X), column (Y) and angle (Z) cells are referenced as shown in Figure 4-11. Labels can be added and must be specified as data-type floating points.



LAPTOP000318 - FormatString

Leading Text:

Trailing Text:

Terminators: CR+LF

☑ Use Delimiter

● Standard: Comma

○ Other:

Label	Cell	Data Type
X,	C3	Floating Point
Y,	D3	Floating Point
Angle,	E3	Floating Point

Label: X

Cell: C3

Data Type: Floating Point

Decimal Places: 2

☑ Include Label

☐ Fixed Width

Field Width: 8

Pad: Leading Spaces

Output String: Characters: 26

```
X,0.00,Y,0.00,Angle,0.00
```

Figure 4-11 FormatString configuration

When the FormatString has been completed, the output string will be displayed in cell B19, which can be read by KUKA Vision to relay the string data to the KUKA robot.

4.2 Interfacing Software Set-up

4.2.1 KUKA Vision Set-up

Before KUKA Vision could be run, the set-up had to be completed. Refer to Figure 4-12 when reading the set-up explanation. Under Number of Cameras only one was selected.

The triggering mode could be varied between either Robot Control, where the robot triggers the vision sensor, or Digital Input, which triggers the vision sensor externally via the breakout board. For this application, Robot Control was selected as a triggering sensor would trigger the vision sensor via the KRL.

For Robot Control, different constant modes are selected. Without Robot Control is used when no KRC is present or a stand-alone PC is used. As Robot Control is active in this application, the following constant modes were selected:

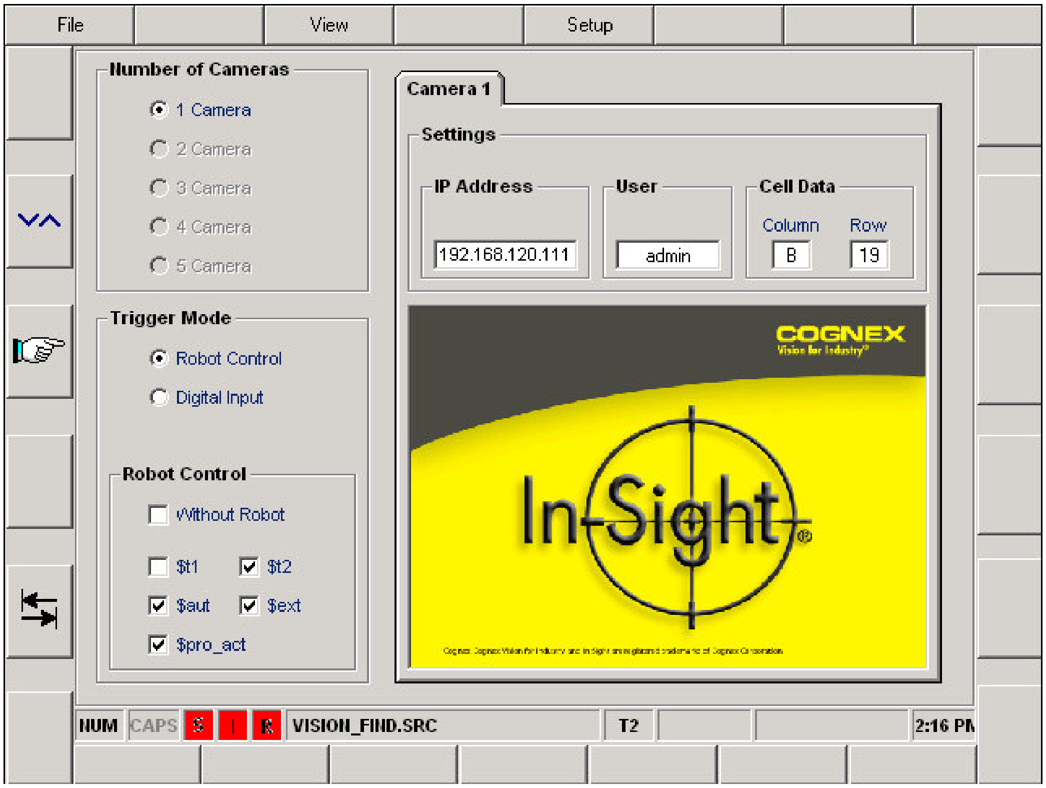
- $t1 Mode $t1 required for Auto Mode was left unchecked
- $t2 Mode $t2 required for Auto Mode
- $aut Mode $aut required for Auto Mode
- $ext Mode $ext required for Auto Mode
- $pro_act Mode $pro_act required for Auto Mode [3].

The IP address 192.168.120.111 is the actual IP address given to the vision sensor. By specifying the IP address, KUKA Vision has an IP location from where the cell data can be retrieved.

The User List maintains the access level and File Transfer Protocol (FTP) read/write privileges for authorised users of In-Sight sensors and emulators. The User List settings determine which users may log on to a particular In-Sight sensor through the Log On/Off dialog, as well as the types of changes they can make to the active job. Each In-Sight sensor has its own User List, separate from every other sensor on the network. If a user needs access to a particular sensor, that person must have a user name and password that already exists in that sensor's User List [3].

A cell is a location in the spreadsheet, identified by its column letter and row number. For example, cell A2 is located at the intersection of column A and row 2. For this application cell B19, referring to column B and row 19, was selected to be used by a format string to load location data for KUKA Vision to retrieve.

The set-up application is then saved and closed. The KUKA Vision setting is stored in the registry. Changes will only be effective if they are saved and KUKA Vision is restarted [3].



Number of Cameras

● 1 Camera
○ 2 Camera
○ 3 Camera
○ 4 Camera
○ 5 Camera

Trigger Mode

● Robot Control
○ Digital Input

Robot Control

□ Without Robot

□ $t1 ☑ $t2
☑ $aut ☑ $ext
☑ $pro_act

Camera 1

Settings

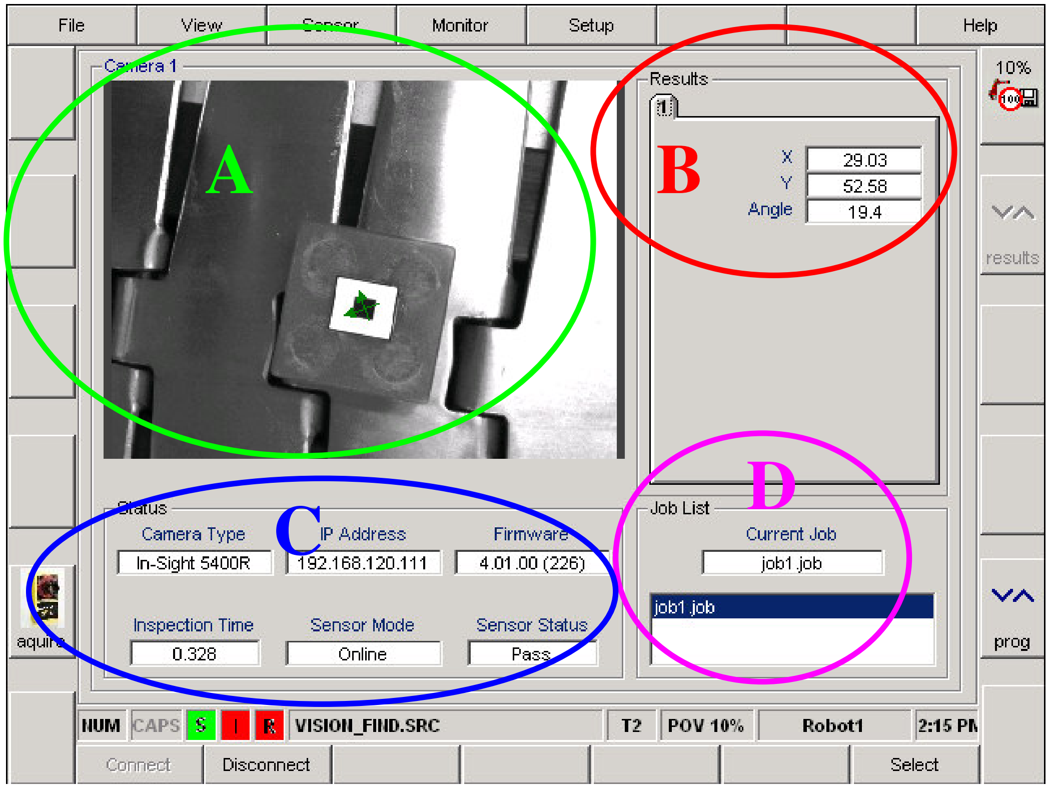
IP Address: 192.168.120.111
User: admin
Cell Data: Column B Row 19

NUM CAPS S I R VISION_FIND.SRC T2 2:16 PM

Figure 4-12 KUKA Vision set-up

4.2.2 Starting KUKA Vision

KUKA Vision will try to automatically connect to the In-Sight Vision sensor each time it comes online. When the COGNEX In-Sight sensor is triggered via the robot, it runs the selected Job, which in this case is job1.job and outputs the results via the FormatString function to a defined cell B19 in the spreadsheet. KUKA Vision then extracts this data and writes it to the variable Results [n, m] (see Figure 4-13) indicated as (B) to the robot's HMI which can be viewed on the interfacing software KUKA Vision. Referring to Figure 4-13, the image captured by the camera is indicated by (A), the job selected to be run by (D) which contains the vision program, and the status of the sensor is indicated by (C), showing time of inspection, sensor mode (On/OFF) and sensor status (Pass/No Data).



X 29.03
Y 52.58
Angle 19.4

Camera Type	IP Address	Firmware
In-Sight 5400R	192.168.120.111	4.01.00 (226)

Inspection Time	Sensor Mode	Sensor Status
0.328	Online	Pass

Current Job job1.job

job1.job

NUM CAPS S I R VISION_FIND.SRC T2 POV 10% Robot1 2:15 PM

Figure 4-13 KUKA Vision running

4.2.3 KUKA Vision Programs

As there are a great many possible vision applications, KUKA Vision does NOT come with robot programs. It must be remembered that KUKA Vision is the interface between a COGNEX In-Sight sensor and the robot. It is up to individuals to write their own controls [3].

This is a simple example to show how to write a simple KRL program that can process the data received from the camera in the robot program. The ";" indicates comments which are not executed when the KRL program is run.

```
;Reset Capture Image Trigger Flag
        CAMERA[1].CAPTURE_IMAGE=FALSE
```

```
;Reset Data_Ready Flag
        CAMERA[1].DATA_READY=FALSE
        WAIT SEC 0.1
;Repeat until data received from camera
        REPEAT
;Trigger an inspection in camera 1
        CAMERA[1].CAPTURE_IMAGE=TRUE
;Wait for inspection to finish
        WHILE (CAMERA[1].CAPTURE_IMAGE==TRUE)
        WAIT SEC 0.0
        ENDWHILE
        UNTIL (CAMERA[1].DATA_READY==TRUE)
;Reset Pick Position
        PICK_POS=$NULLFRAME
;Load Pick Position with data from camera
        PICK_POS.X=RESULT[1,1]
        PICK_POS.Y=RESULT[1,2]
        PICK_POS.A=RESULT[1,3]
        LIN PICK_POS
```

4.3 KUKA Robot

4.3.1 KUKA Tool Calibration

To calibrate the suction cup the calibration program “XYZ-4 POINT” was selected. In the “4-Point” method, the TCP of the tool is moved to a reference point from four different directions (hence “4-Point” method) as shown in Figure 4-14. The position of the TCP is then calculated on the basis of the various positions and orientations of the robot flange [4].

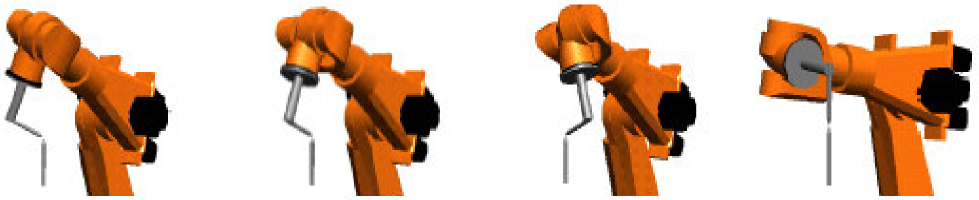


Figure 4-14 Four-point calibration method [4]

To carry out the calibration, the calibration link in Figure 4-15 was followed.

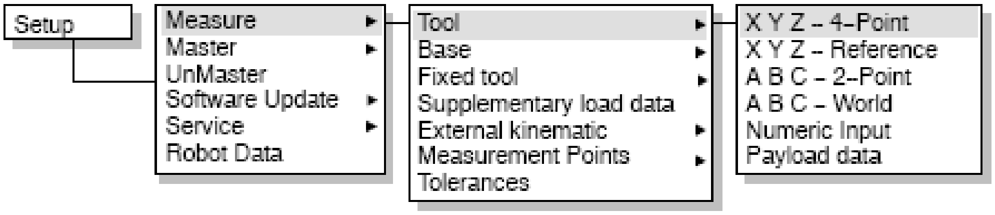


Figure 4-15 Calibration link [4]

A dialog window for the 4-point calibration method is opened. Here the tool number and tool name must be specified as they will be required later for motion programming. Once this information has been accepted, the user is prompted to jog the robot to any four reference points around the spiked base as shown in Figure 4-16.

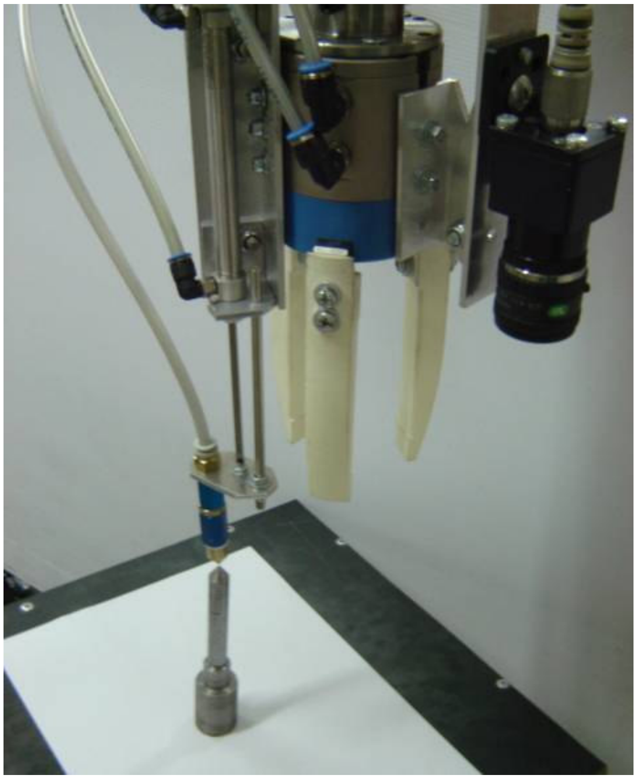


Figure 4-16 Calibrating the TCP

Once the TCP has been calibrated, the BASE must be done. The reason for this is that the robot must have the same BASE co-ordinates as the vision sensor. When the vision sensor was calibrated, a grid of dots was used and the origin and the X and Y co-ordinate directions were specified. Now the robot's BASE must be calibrated according to the grid used. The BASE calibration link in Figure 4-17 was followed**.**

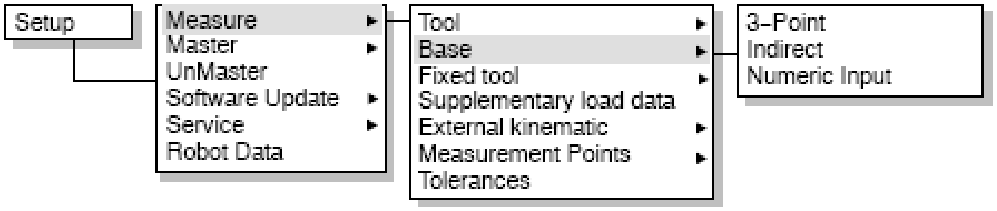


Figure 4-17 BASE calibration link [4]

The 3-point method was used to calibrate the BASE. The TCP was first moved to the origin and calibrated, and then it was moved in the positive X direction and calibrated. Next, it was moved along the XY plane in the positive Y direction, basically in an L shape, and calibrated. Figure 4-18 illustrates the L shape calibration motion. Once the TCP and BASE have been calibrated, it is possible to program the motion using the two calibrations.

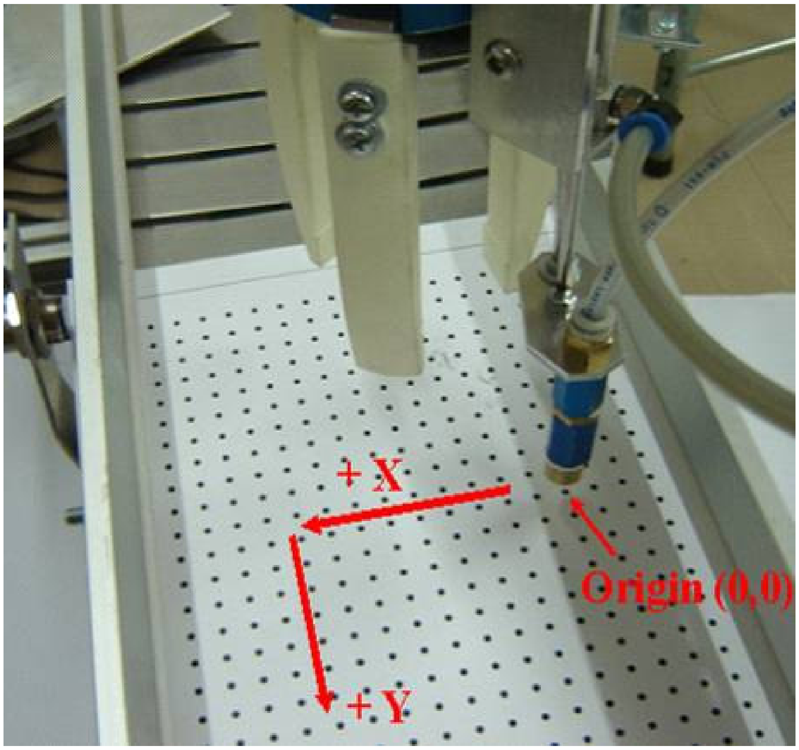
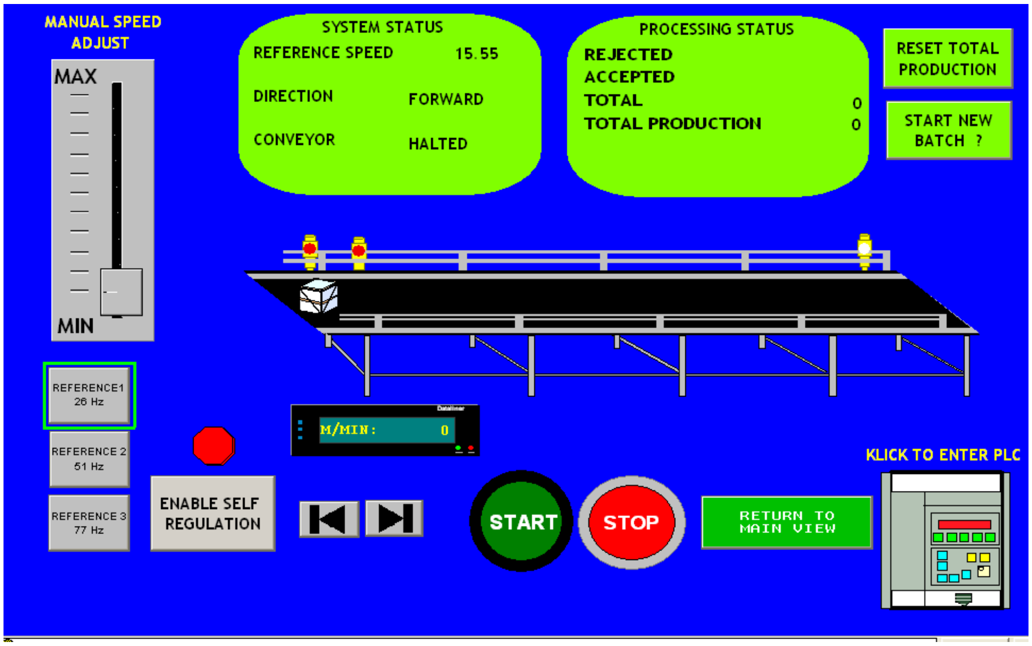


Figure 4-18 BASE calibration

4.4 PLC Program

An ALLEN BRADLEY PLC was used for controlling the speed of the conveyor via a SCADA interface shown in Figure 4-19 which has three different speed levels for the operator to select from. These different speed levels are used to indicate to the robot, by means of the number of bits being active, at what speed the conveyor is moving.

Based on the speed level selected from the SCADA, the robot is able to synchronise its speed to that of the conveyor. This will allow an accurate pick at different speeds from the conveyor.



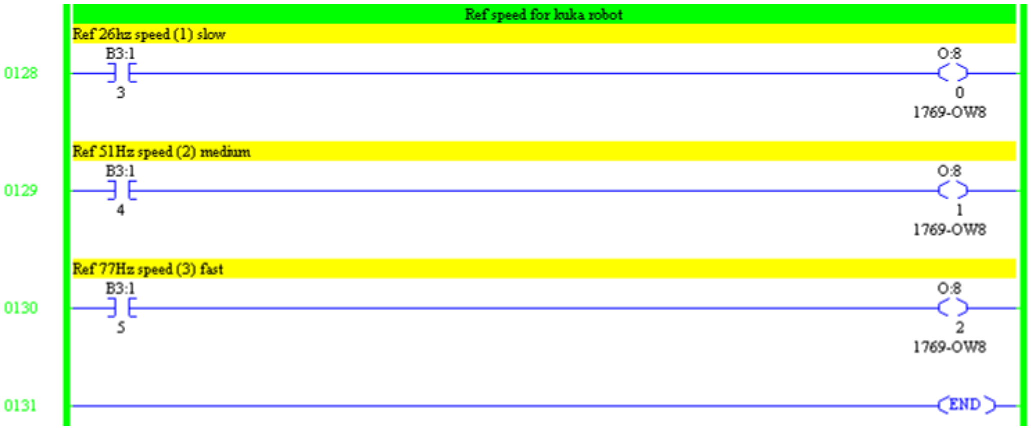
SYSTEM STATUS
REFERENCE SPEED 15.55
DIRECTION FORWARD
CONVEYOR HALTED

PROCESSING STATUS
REJECTED
ACCEPTED
TOTAL 0
TOTAL PRODUCTION 0

M/MIN: 0

Figure 4-19 SCADA controller

A small program was written to give the robot three hardwired inputs into its I/O module. The full system program can be viewed in Appendix A.



Ref speed for kuka robot

0128 Ref 26hz speed (1) slow — B3:1 3 — O:8 0 1769-OW8

0129 Ref 51Hz speed (2) medium — B3:1 4 — O:8 1 1769-OW8

0130 Ref 77Hz speed (3) fast — B3:1 5 — O:8 2 1769-OW8

0131 END

Figure 4-20 Speed program

4.5 References

[1] Installing In-Sight 5000 Series Vision Sensors [Online] Available from: http://www.cognex.com [Accessed 4 April 2008].

[2] COGNEX Calibration [Online] Available from: http://www.cognex.com [Accessed 4 April 2008].

[3] KUKA [1].Vision V8.PDF [Online] Available from: http://www.kuka.com [Accessed 4 April 2008].

[4] KUKA Start-Up .PDF [Online] Available from: http://www.kuka.com [Accessed 4 April 2008].

5 Implementation and Results

5.1 TEST 1

5.1.1 Introduction to Vision Program

Using the Vision program, three different objects were inspected as shown in Figure 5-1. To aid with inspection, a direct lighting method was used to illuminate the edges of the objects. As the objects had different shapes and very large grey scale differences (three different colours giving a gray scale difference of 0 to 255), this led to complexities in the Vision program. One of the major difficulties was that the square objects were very similar in size to the slats which formed part of the conveyor belt. It was therefore decided to simplify the inspection and try to establish a working platform first before trying to accomplish too many tasks at once.

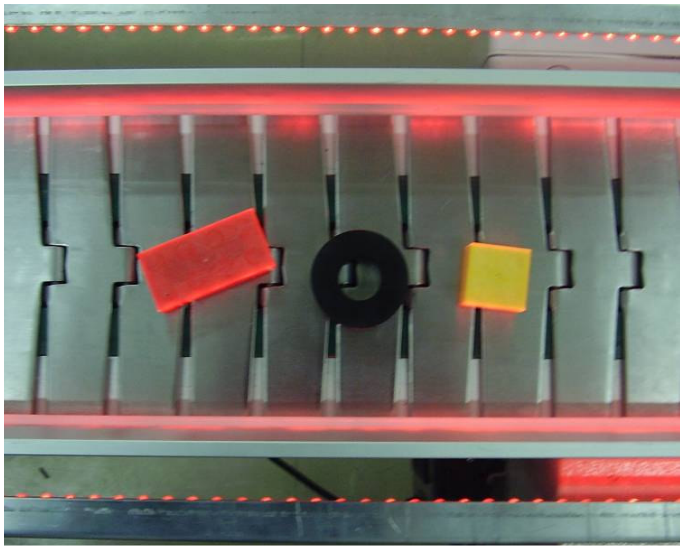


Figure 5-1 Different objects as they appear on the conveyor

For the new Vision program, a printed pattern pasted onto the squares Figure 5-2 was used. This would still be considered as quality control as the squares would be picked and placed while the circle and rectangle would be left. This printed pattern proved to be such an improvement that the direct lights were no longer required. The other positive aspect was that the fluorescent lighting in the laboratory, which pulsed at 50 Hz, had no effect on the images captured, which proved that the Vision software was indeed very powerful.

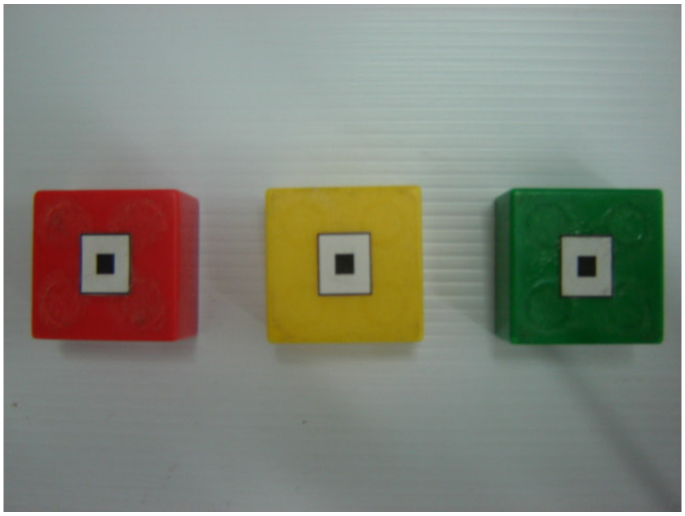
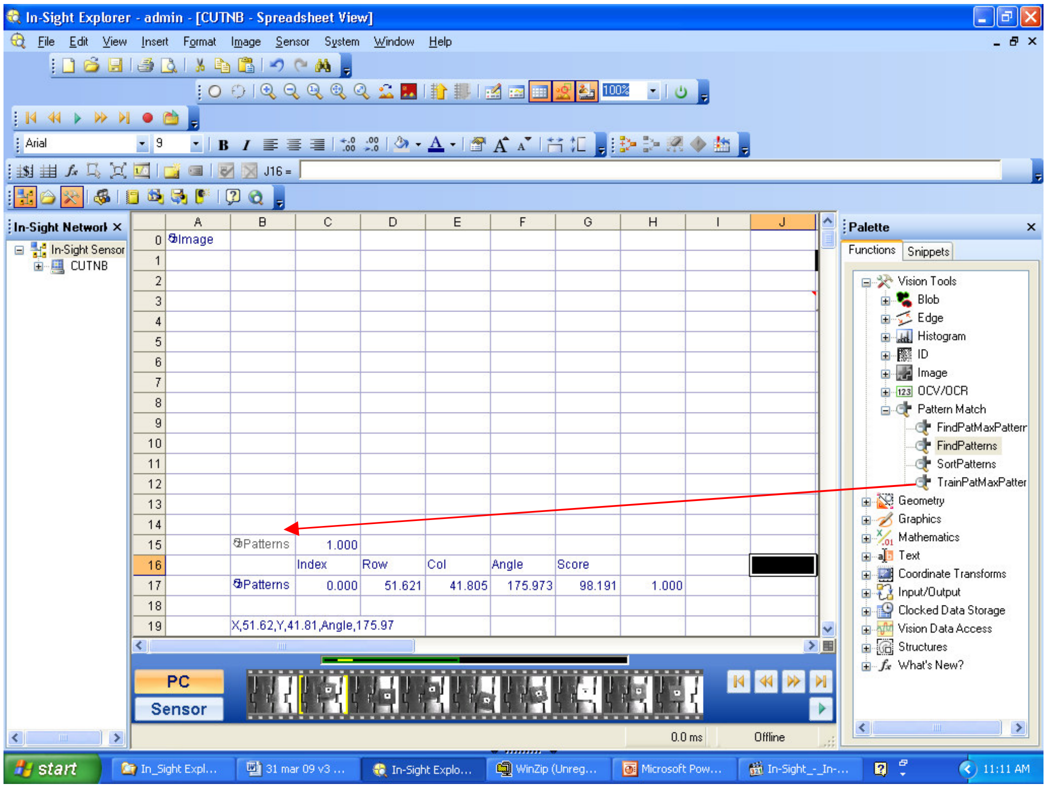


Figure 5-2 Squares with printed pattern

PatMax, a feature-based pattern location tool, was used to train the pattern pasted on top of the squares. PatMax was chosen as it has very high accuracy and repeatability. Pixel values can also be ignored altogether with PatMax as patterns that were dark are now light and light patterns are now dark.

5.1.2 Vision Program V1.0

The TrainPatMaxPattern function is dragged and dropped into the spreadsheet as shown in Figure 5-3 to train the pattern that we wanted the program to look for.



	A	B	C	D	E	F	G	H	I	J
0	Image									
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										
11										
12										
13										
14										
15		Patterns	1.000							
16			Index	Row	Col	Angle	Score			
17		Patterns	0.000	51.621	41.805	175.973	98.191	1.000		
18										
19		X,51.62,Y,41.81,Angle,175.97								

Figure 5-3 TrainPatMaxPattern

Once the TrainPatMaxPattern function has been added to the spreadsheet, the Property Sheet, as shown in Figure 5-4, pops up.

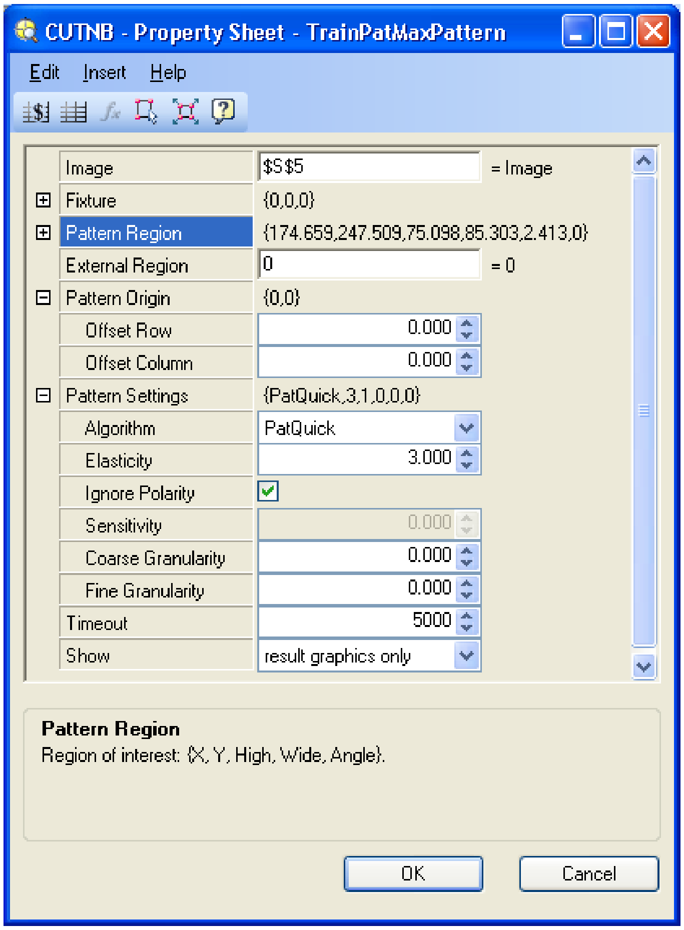


Image	S5	= Image
⊞ Fixture	{0,0,0}	
⊞ Pattern Region	{174.659,247.509,75.098,85.303,2.413,0}	
External Region	0	= 0
⊟ Pattern Origin	{0,0}	
Offset Row	0.000	
Offset Column	0.000	
⊟ Pattern Settings	{PatQuick,3,1,0,0,0}	
Algorithm	PatQuick	
Elasticity	3.000	
Ignore Polarity	☑	
Sensitivity	0.000	
Coarse Granularity	0.000	
Fine Granularity	0.000	
Timeout	5000	
Show	result graphics only	

Pattern Region

Region of interest: {X, Y, High, Wide, Angle}.

Figure 5-4 TrainPatMaxPattern Property Sheet

Refer to the Property Sheet in Figure 5-4 for the explanation that follows:

- **Image:** Is referenced to the target image cell, which in this case is the calibrated image cell S5. See appendix C, Figure 8-1 and Figure 8-2 for dependency levels.
- **Fixture:** Where the tool should fix itself on the image.
- **Pattern region:** Region specifying the feature to be trained. The pattern is dragged around the square printed region as this is the region to be trained

Figure 5-5. PatMax will then extract features of the image which will be searched for when the objects are inspected, shown in green Figure 5-6.

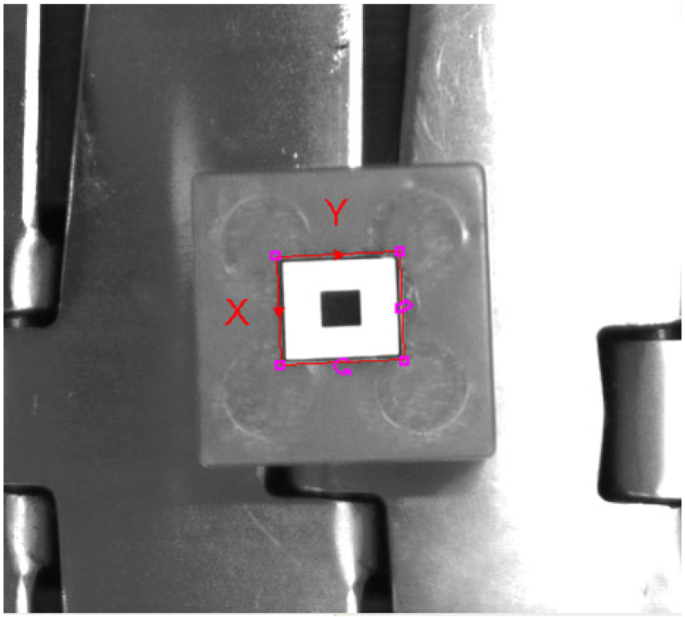


Figure 5-5 Pattern region

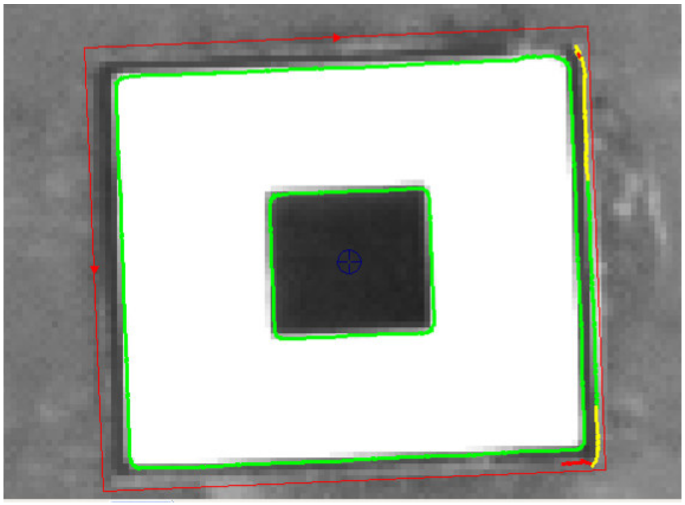


Figure 5-6 Features of interest

- **Pattern origin:** The location that must be reported within the pattern. This location can sometimes be very specific when a pick application is used, and it can be adjusted as required. The location of interest for the pick application is specified as Origin {0,0} on the image as this is the centre point co-ordinate where the suction cup is to be placed Figure 5-7.

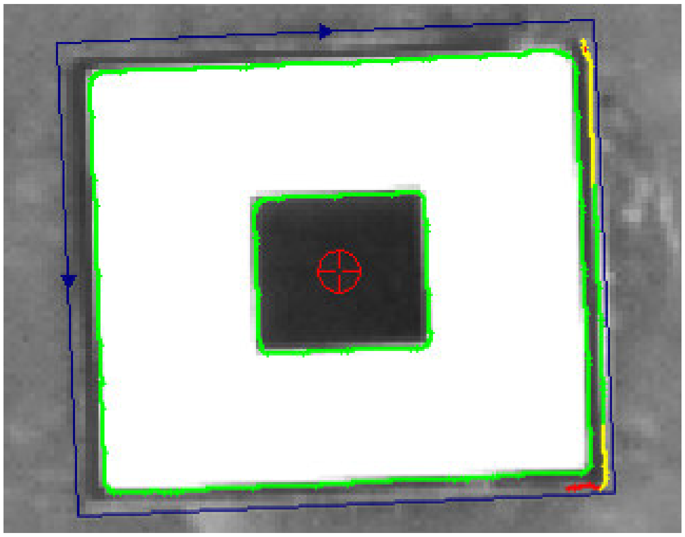
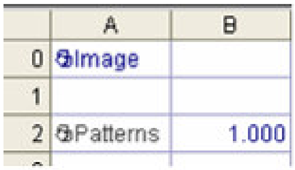


Figure 5-7 Pattern origin

- **Pattern settings:** Specialised settings based on the pattern. The algorithm selected was PatQuick as it is the best for speed, tolerates more image variation and is perfect for pick-and-place applications as inspection time must be kept to a minimum. The elasticity was set to 3 to allow some tolerance for nonlinear geometric changes of the printed image that is inspected. Ignore Polarity is checked as both polarities are checked for the pattern. Granularity is left at 0.0 as PatQuick will automatically choose the best accuracy.

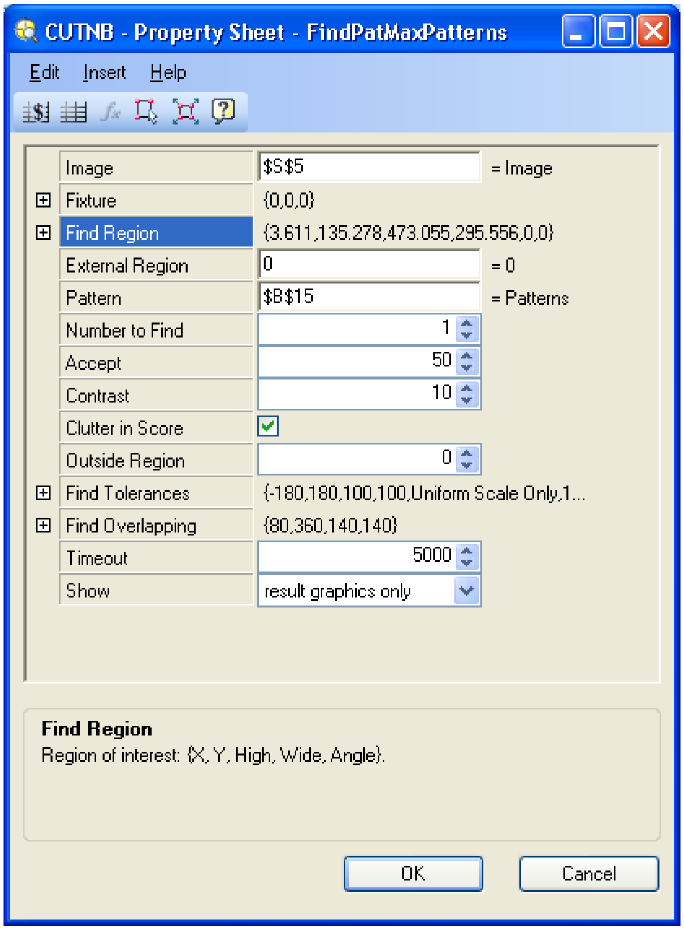
Once the TrainPatMaxPattern settings have been configured and "OK'ed", the result is a trained PatMaxPattern structure which is indicated by a 1 if trained successfully or a 0 if not. See the example in Figure 5-8



	A	B
0	Image	
1		
2	Patterns	1.000

Figure 5-8 Image trained

Once the pattern has been trained the next step is to find it. The FindPatMaxPattern function is dragged and dropped into the spreadsheet just below the TrainPatMaxPattern. The property sheet for the shown in Figure 5-9 pops up.



CUTNB - Property Sheet - FindPatMaxPatterns

Image	S5	= Image
⊞ Fixture	{0,0,0}	
⊞ Find Region	{3.611,135.278,473.055,295.556,0,0}	
External Region	0	= 0
Pattern	B15	= Patterns
Number to Find	1	
Accept	50	
Contrast	10	
Clutter in Score	☑	
Outside Region	0	
⊞ Find Tolerances	{-180,180,100,100,Uniform Scale Only,1...	
⊞ Find Overlapping	{80,360,140,140}	
Timeout	5000	
Show	result graphics only	

Find Region
Region of interest: {X, Y, High, Wide, Angle}.

Figure 5-9 FindPatMaxPattern Property Sheet

Refer to the property sheet in Figure 5-9 for the explanation that follows:

- **Find region:** Region specifying the search area for the pattern within the image. The entire field of view can be used, but to decrease processing time the region has been made smaller Figure 5-10.

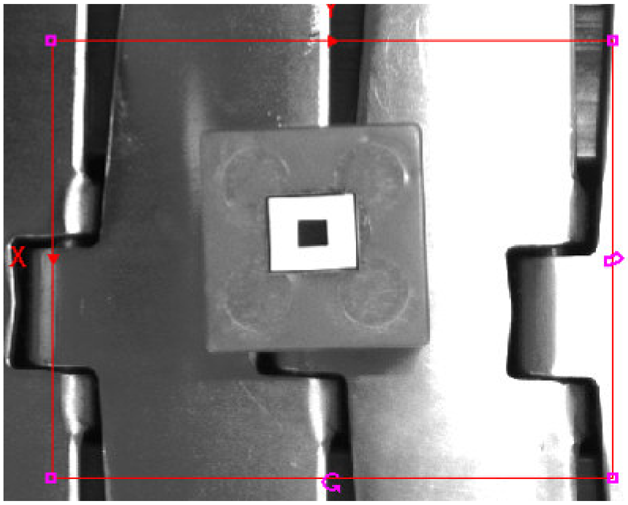
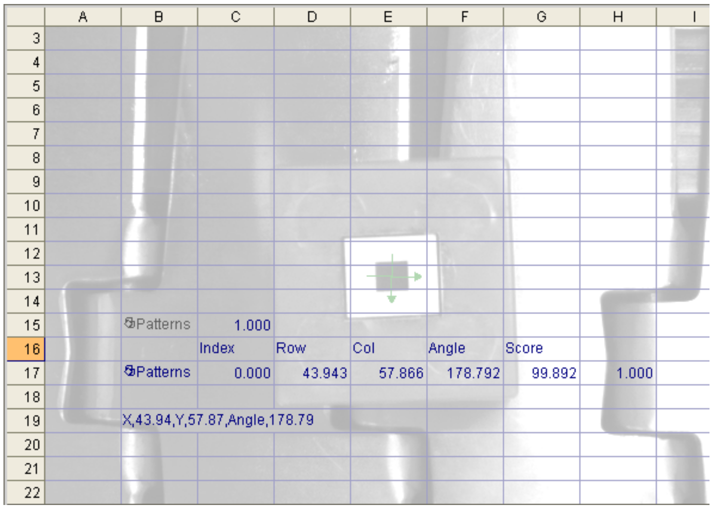


Figure 5-10 Find region

- **Pattern:** It is very important to reference the Trained PatMaxPattern $B$15 that is being searched.
- **Find tolerances:** Used to set the allowable rotation of the object. In this case the allowable rotation is -180 to 180. Anything outside this rotation will not be allowed.
- **Find overlapping:** Sets the allowable overlapping between matches found**.**

Now that the pattern has been trained and we can find the pattern, a logic 1 will indicate if the pattern has been found or not. The next step is to relay the location information via KUKA Vision to the robot. This is done by means of the FormatString. The FormatString is located in cell B19 Figure 5-11. The naming convention of the program is critical. To allow Vision Program v1.0 to be selected using the KUKA Robot Logic (KRL) to run the program, it was saved as job1.job. Refer to Appendix B for full the vision program with the dependencies indicated.



	A	B	C	D	E	F	G	H	I
15		Patterns	1.000						
16			Index	Row	Col	Angle	Score		
17		Patterns	0.000	43.943	57.866	178.792	99.892	1.000	
18									
19		X,43.94,Y,57.87,Angle,178.79							

Figure 5-11 Vision Program V1.0

5.1.3 Robot Program V1.0

Now that the vision program is complete and the information on the object's location can be read by KUKA Vision during inspection, the KUKA Robot Logic (KRL) program can start. Note that the first KRL program is executed while the conveyor is stationary.

The robot will move from its home position, i.e. the position from where the motion program starts and stops, to the inspection position at sensor 1 (Figure 5-12), placing the vision sensor at the inspection position. At this position the robot will wait for input 4 to equal logic 1 (WAIT FOR $IN [4] == TRUE) which is sensor 1 in Figure 5-12. For testing purposes sensor 1 was positioned manually: the object was placed under the vision sensor at the inspection position while conveyor 1 was stationary.

Once sensor 1 has been positioned and input 4 is equal to logic 1, the KRL vision is activated. First job1.job is selected and the Capture Image and Data Ready flags are reset to clear any previous data that may have been stored in the memory locations. The reset is repeated until the image is captured.

When the image is captured (CAMERA [1].CAPTURE_IMAGE=TRUE), a repeat until loop is executed until the Data Ready flag is set to true, i.e. the image location information has been received. The pick position is reset to clear any previous data.

The pick position data of the image is then loaded with the data received from the camera, giving the robot the X, Y and Z locations of the pattern found. The robot then moves the suction cup linearly to the centre point directly above the pattern found. Once the motion is completed, the robot moves back to its home position, i.e. the position from where it started.

A flow chart of the KRL is shown in Figure 5-13. Refer to Appendix C for the full KRL program.

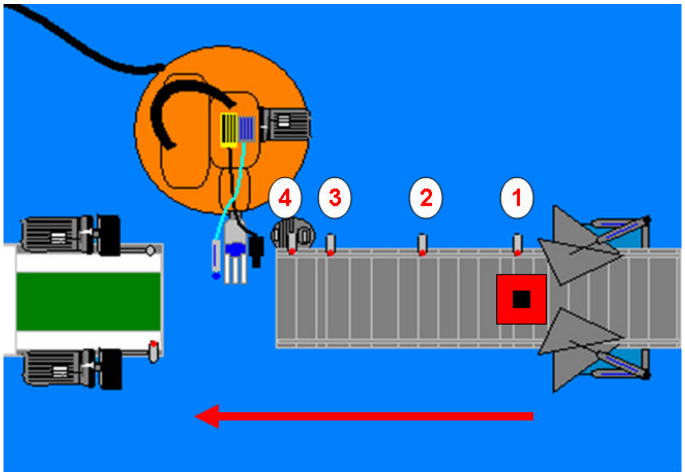


Figure 5-12 Illustration of robot inspection

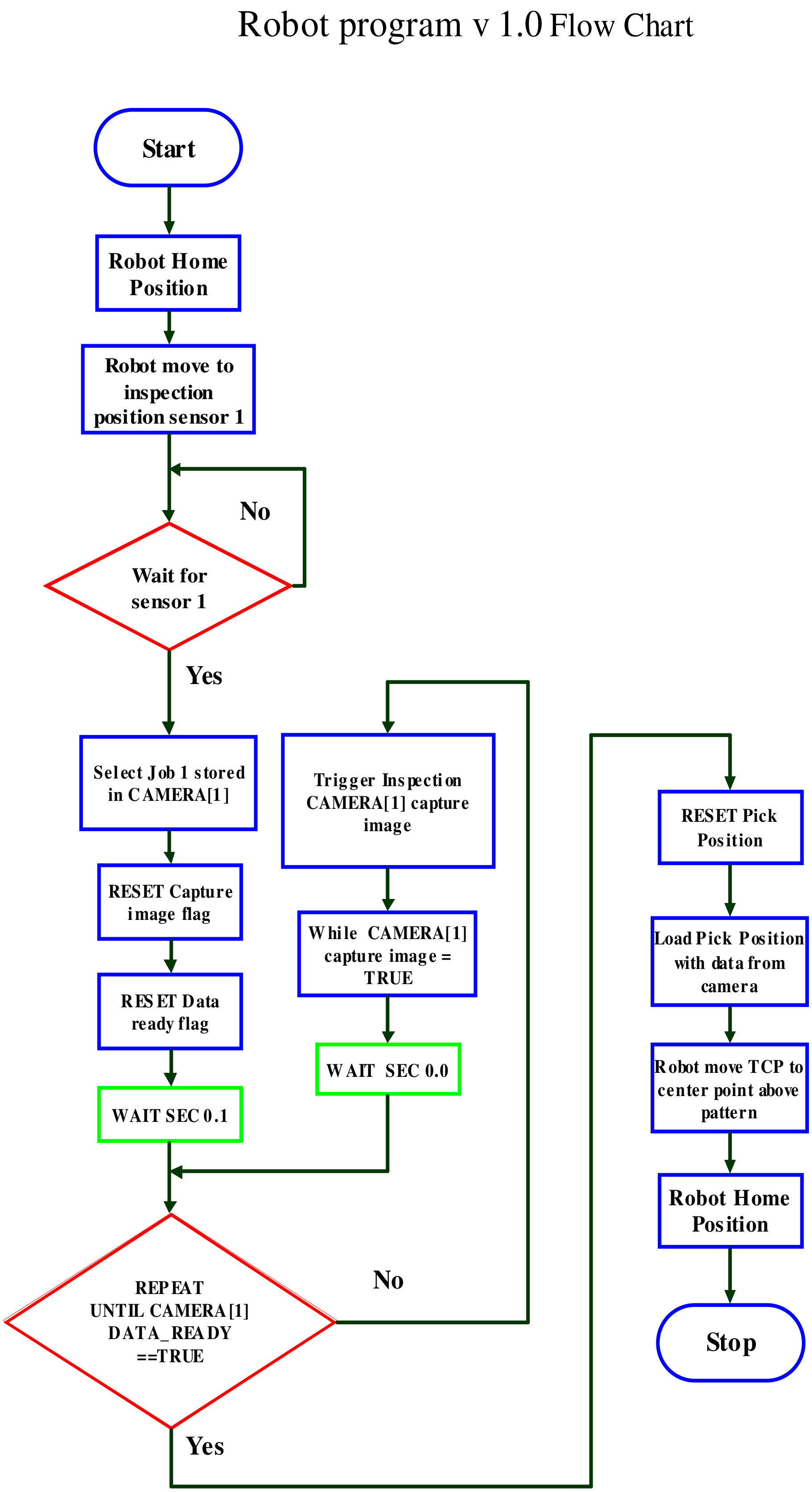

Robot program v 1.0 Flow Chart
Start
Robot Home Position
Robot move to inspection position sensor 1
No
Wait for sensor 1
Yes
Select Job 1 stored in CAMERA[1]
RESET Capture image flag
RESET Data ready flag
WAIT SEC 0.1
Trigger Inspection CAMERA[1] capture image
While CAMERA[1] capture image = TRUE
WAIT SEC 0.0
REPEAT UNTIL CAMERA[1] DATA_READY ==TRUE
No
Yes
RESET Pick Position
Load Pick Position with data from camera
Robot move TCP to center point above pattern
Robot Home Position
Stop

Figure 5-13 Test 1 flow chart

5.1.4 Summary

The combination of vision program v1.0 and robot program v1.0 proves that the two separately written programs and interfacing software are functioning correctly. The next test now commenced, which consisted of an actual pick of an object from stationary conveyor 1 and placing it on conveyor 2.

5.2 TEST 2

5.2.1 Vision Program V1.0

The vision program for test 2 was not altered in any way and was used as is.

5.2.2 Robot Program V1.1

The robot program is identical to v1.0. The only difference to v1.1 is that the robot must, after inspecting the object and moving the TCP directly above the centre point of the object, pick the object while conveyor 1 is stationary and place it on conveyor 2 (Figure 5-14). Sensors 3 and 4 are in place for possible speed testing. The flow chart in Figure 5-15 depicts the KRL program. Refer to Appendix D for the full KRL program.

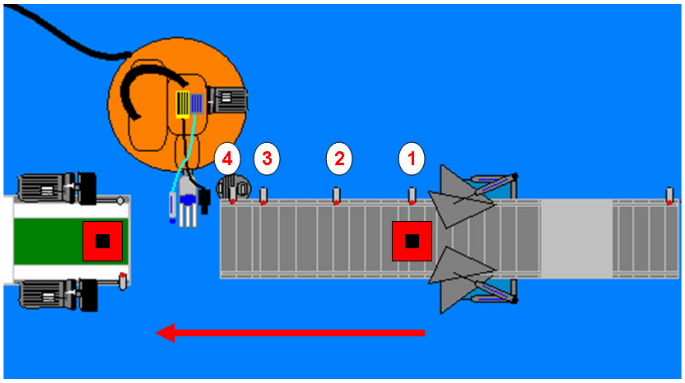


Figure 5-14 Stationary pick and place

Robot program v 1.1 Flow Chart

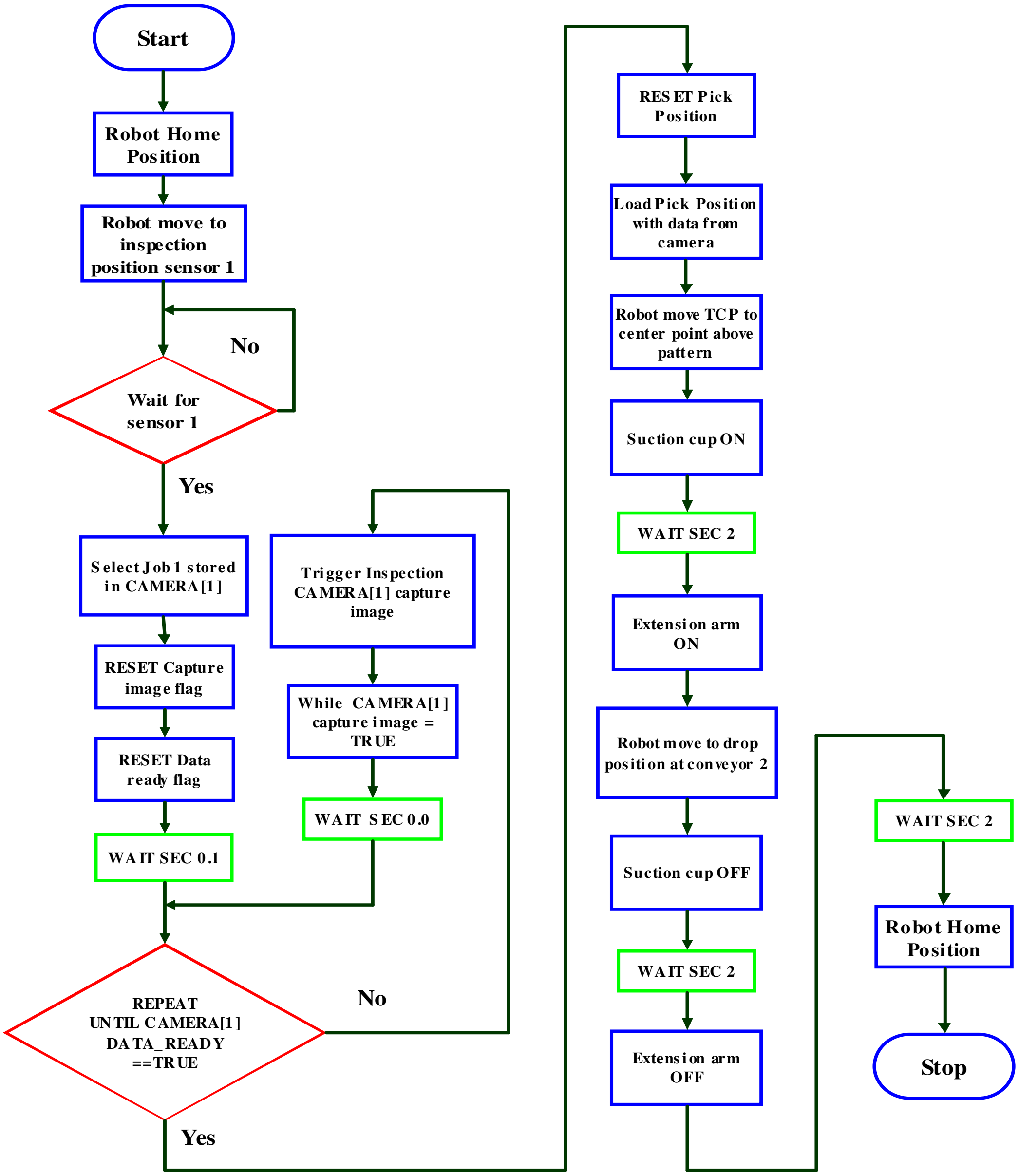

Start
Robot Home Position
Robot move to inspection position sensor 1
No
Wait for sensor 1
Yes
Select Job 1 stored in CAMERA[1]
RESET Capture image flag
RESET Data ready flag
WAIT SEC 0.1
Trigger Inspection CAMERA[1] capture image
While CAMERA[1] capture image = TRUE
WAIT SEC 0.0
No
REPEAT UNTIL CAMERA[1] DATA_READY ==TRUE
Yes
RESET Pick Position
Load Pick Position with data from camera
Robot move TCP to center point above pattern
Suction cup ON
WAIT SEC 2
Extension arm ON
Robot move to drop position at conveyor 2
Suction cup OFF
WAIT SEC 2
Extension arm OFF
WAIT SEC 2
Robot Home Position
Stop

Figure 5-15 Test 2 flow chart

5.2.3 Summary

The pick and place action from stationary conveyor 1 to conveyor 2 was successful. Based on this test, it was now possible to do the pick and place with conveyor 1 in motion.

5.3 TEST 3

5.3.1 Vision Program V1.0

The vision program for test 3 was not altered in any way and was used as is.

5.3.2 Robot Program V1.2

While conveyor 1 is in motion the robot moves to the inspection position and waits for sensor 1 to be activated. Once the object reaches sensor 1 the object is inspected and the robot then moves the TCP to the pick position at sensor 2 based on the pattern co-ordinate data received. The suction cup is switched on while the robot moves to the pick position at sensor 2 Figure 5-16. A message window displays the found pattern co-ordinates, which aids in further development of the program. When sensor 2 is reached, the extension arm moves out to make contact with the object. The robot moves 600 in the Z up from conveyor 1 and then across to conveyor 2. The suction cup is switched off to release the object onto conveyor 2 and the extension arm is retracted. The robot will move 20 in the Z up from conveyor 2 then back to the home position. The program is looped to run continuously. The program flow chart is shown in Figure 5-17. Refer to Appendix E for the KRL program.

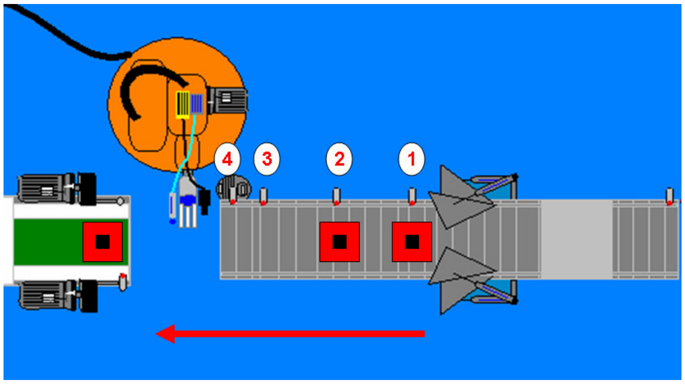


Figure 5-16 Pick and place

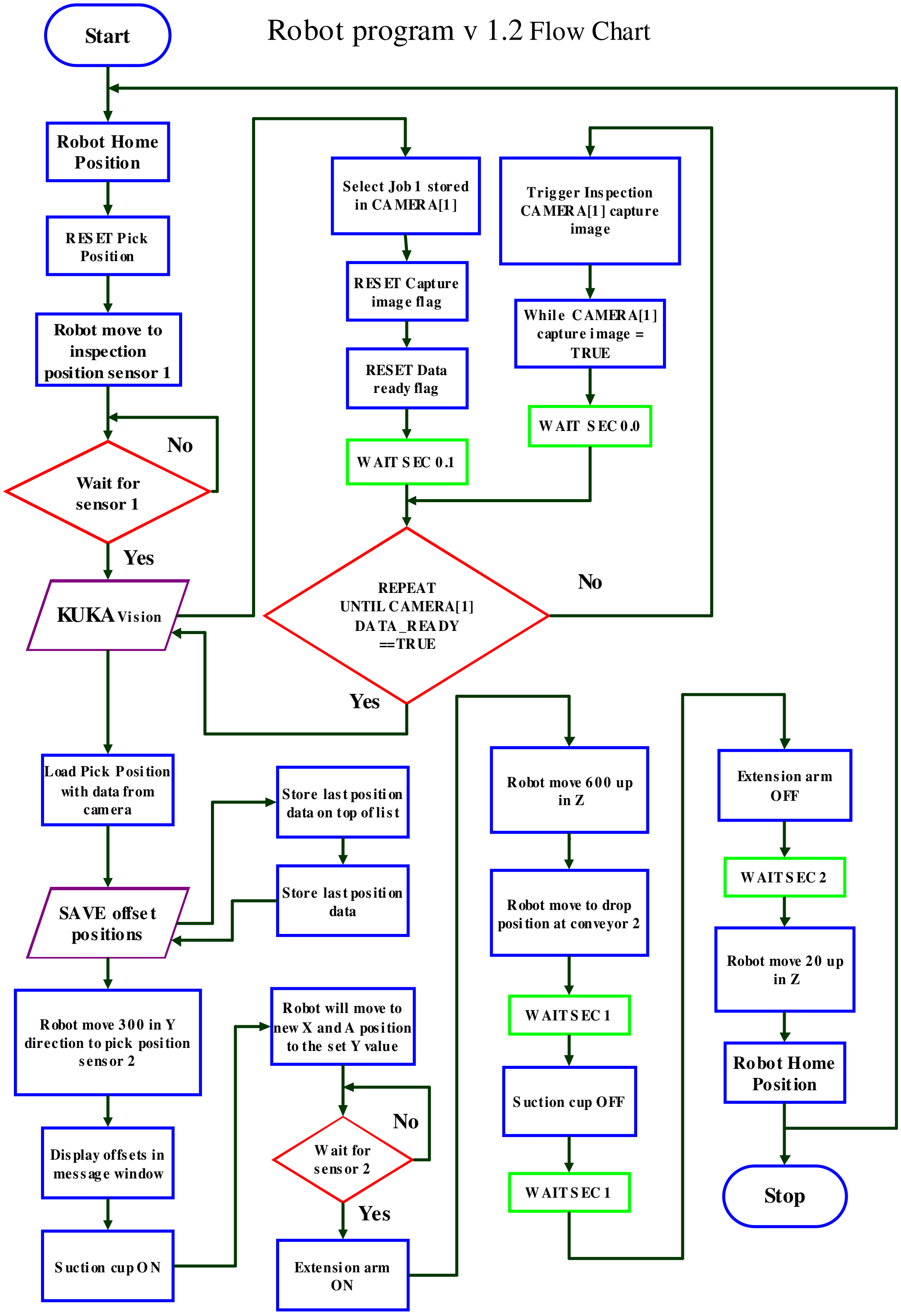

Robot program v 1.2 Flow Chart
Start
Robot Home Position
RESET Pick Position
Robot move to inspection position sensor 1
No
Wait for sensor 1
Yes
KUKA Vision
Select Job 1 stored in CAMERA[1]
RESET Capture image flag
RESET Data ready flag
WAIT SEC 0.1
Trigger Inspection CAMERA[1] capture image
While CAMERA[1] capture image = TRUE
WAIT SEC 0.0
REPEAT UNTIL CAMERA[1] DATA_READY ==TRUE
No
Yes
Load Pick Position with data from camera
SAVE offset positions
Store last position data on top of list
Store last position data
Robot move 300 in Y direction to pick position sensor 2
Display offsets in message window
Suction cup ON
Robot will move to new X and A position to the set Y value
No
Wait for sensor 2
Yes
Extension arm ON
Robot move 600 up in Z
Robot move to drop position at conveyor 2
WAIT SEC 1
Suction cup OFF
WAIT SEC 1
Extension arm OFF
WAIT SEC 2
Robot move 20 up in Z
Robot Home Position
Stop

Figure 5-17 Test 3 flow chart

5.3.3 Summary

The pick and place action from a moving conveyor worked with great success. Based on this test, it has now been proven that objects can be picked and placed by a VGR from a moving conveyor. The programs can be optimised to ensure improved functionality of the system, utilising the working platform that has been established.

5.4 TEST 4

5.4.1 Vision Program V1.1

The search for previous patterns by the vision program proved to be not as effective because of the accuracy of detecting the objects. Vision program V1.0 would recognise patterns of no interest in the slats of the conveyor belt as these images were very similar to the trained patterns. The location details were reported to the robot as pass when the inspection sensor triggered the camera. The problem areas are circled in red in Figure 5-18.

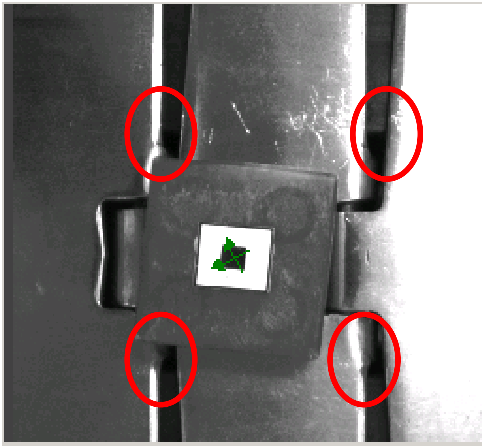


Figure 5-18 Image detection areas

It was decided to create a unique pattern for the camera to find, something which would not be identifiable anywhere else except on the part to be picked. A simple 6 x 6 mm pattern was drawn consisting of an outer square and a few simple images located within it as shown in Figure 5-19. The effectiveness of the unique pattern was established later with the aid of a benchmark test.

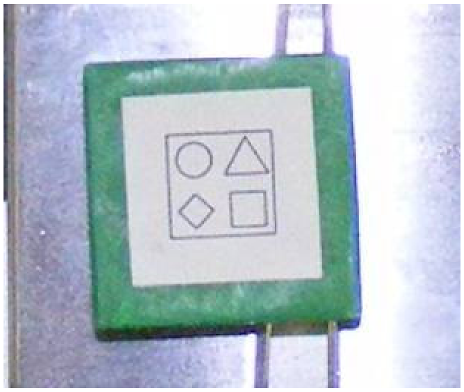
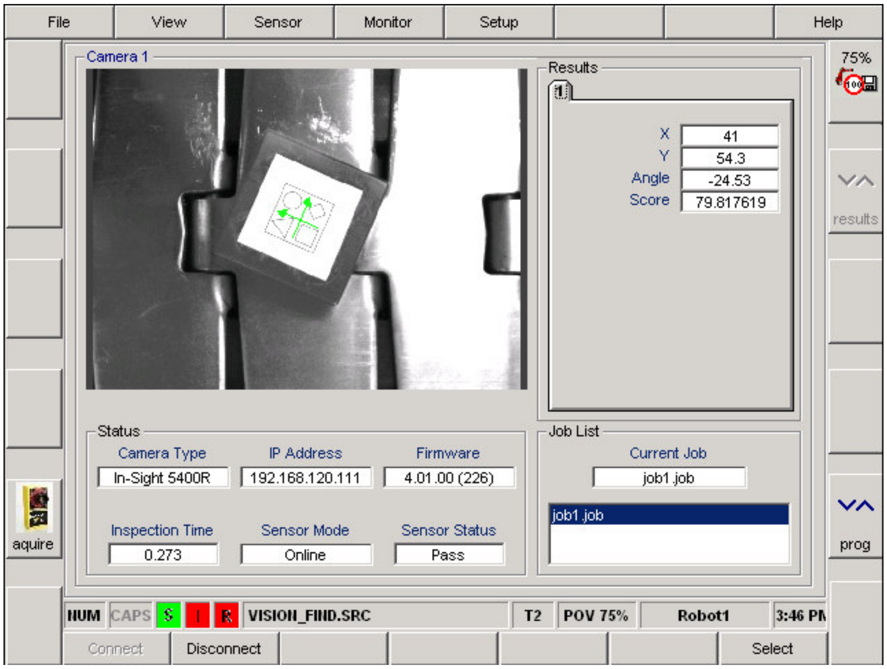


Figure 5-19 New image to be inspected

The new pattern that was inspected did not involve much alteration of the vision program. The program structure remained unchanged and only the new pattern to be inspected was trained. The score of the pattern determined in the vision software was added to KUKA Vision (see the results tab in Figure 5-20) via the format string which relays the pattern location details to the robot. It would be a great help to be able to view the various percentage ranges displayed. The acceptable percentage threshold was limited to 50%, thus everything above this figure would be picked and the rest discarded. The camera was programmed to run the new program job1.job of the In-Sight 5400R module on start-up. The new program was saved and put online. The new vision programs with its dependencies are shown in Appendix F.



Results

X	41
Y	54.3
Angle	-24.53
Score	79.817619

Status

Camera Type	IP Address	Firmware
In-Sight 5400R	192.168.120.111	4.01.00 (226)
Inspection Time	**Sensor Mode**	**Sensor Status**
0.273	Online	Pass

Job List

Current Job: job1.job

job1.job

```
NUM CAPS S I R VISION_FIND.SRC   T2   POV 75%   Robot1   3:46 PM
```

Figure 5-20 KUKA Vision display

5.4.2 Robot Program V1.3

Speed and accuracy were the two most important factors to take into consideration for the KRL program V1.3. The aim was to try and improve processing speed and accuracy during and after inspection of the object.

The first notable problem was that when parts of no interest passed under the camera, the vision subprogram would get stuck in an endless loop. The reason for this was that the vision subprogram was written only to handle parts of interest and not parts of no interest, thus causing an endless loop. This meant that an extra bit of code had to be added to the vision subprogram for parts of no interest to be handled. If the part is of no interest the vision subprogram is forced to exit and continue further in the main program. An "if" statement handles the condition if no data is received from the camera. If the condition is true, a "go to" is executed, meaning that the program is jumped to reset the pick position data while still at the inspection position. This causes the KRL program to wait for inspection sensor 1 to be triggered for the next inspection. This effectively solved the problem. The robot will now stay at the inspection position and let parts of no interest pass along the conveyor and will only pick parts of interest from the moving conveyor.

To improve the speed of the KRL execution systematically, the wait times were shortened to 0.5 seconds from 1 or 2 seconds. The actions were executed faster with a gain of 1.5 seconds, which is not much in human terms but is still very notable in the execution of the KRL program.

The two new programs were subjected to a benchmark test described later in this section to establish how well they worked and to determine whether any alterations would still have to be made.

The flow chart for the KRL program is shown in Figure 5-21 and Figure 5-22, and the entire KRL program is given in Appendix G.

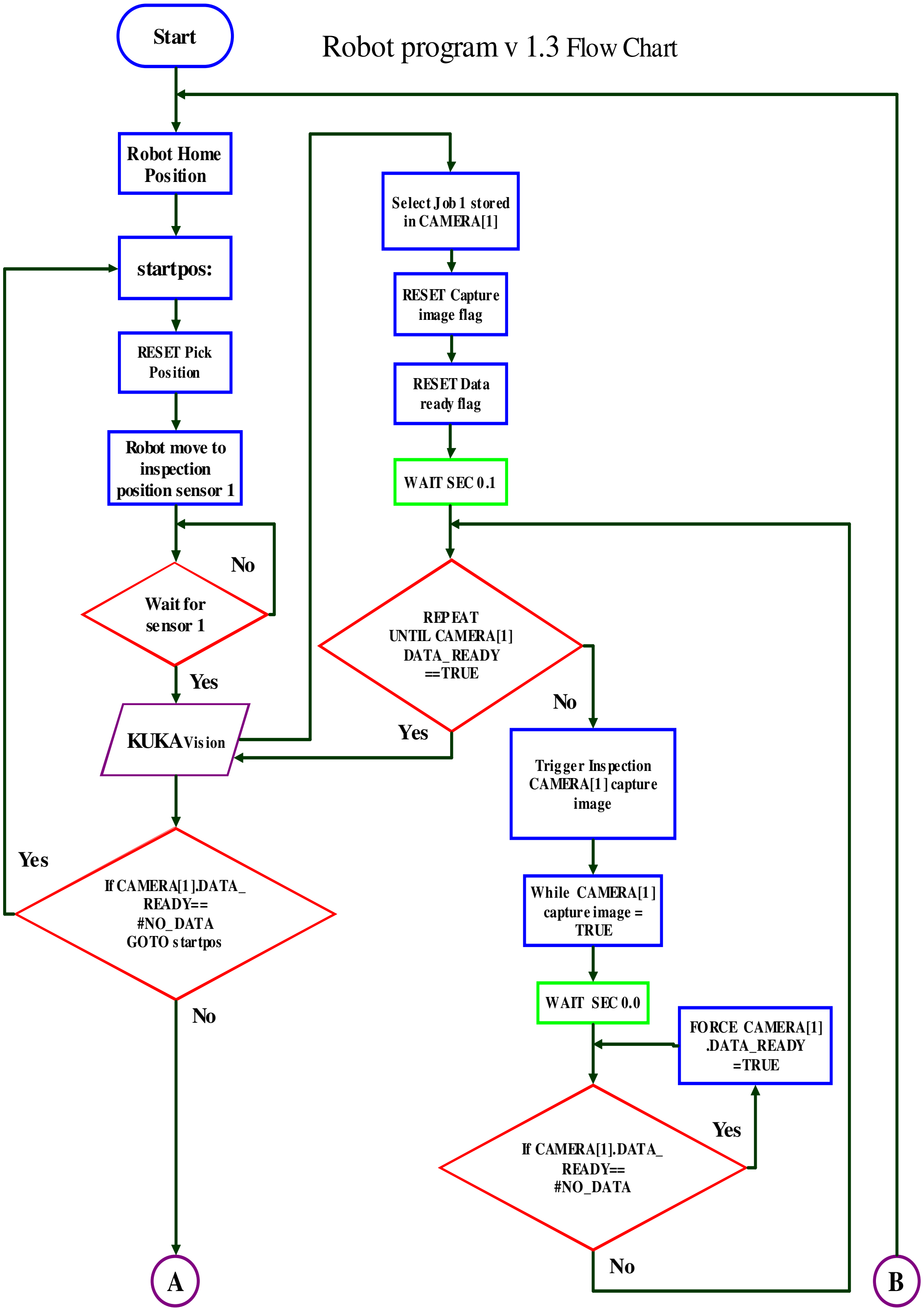

Robot program v 1.3 Flow Chart
Start
Robot Home Position
startpos:
RESET Pick Position
Robot move to inspection position sensor 1
No
Wait for sensor 1
Yes
KUKAVision
Yes
If CAMERA[1].DATA_ READY== #NO_DATA GOTO startpos
No
A
Select Job 1 stored in CAMERA[1]
RESET Capture image flag
RESET Data ready flag
WAIT SEC 0.1
REPEAT UNTIL CAMERA[1] DATA_READY ==TRUE
No
Yes
Trigger Inspection CAMERA[1] capture image
While CAMERA[1] capture image = TRUE
WAIT SEC 0.0
FORCE CAMERA[1] .DATA_READY =TRUE
Yes
If CAMERA[1].DATA_ READY== #NO_DATA
No
B

Figure 5-21 Test 4 flow chart 1

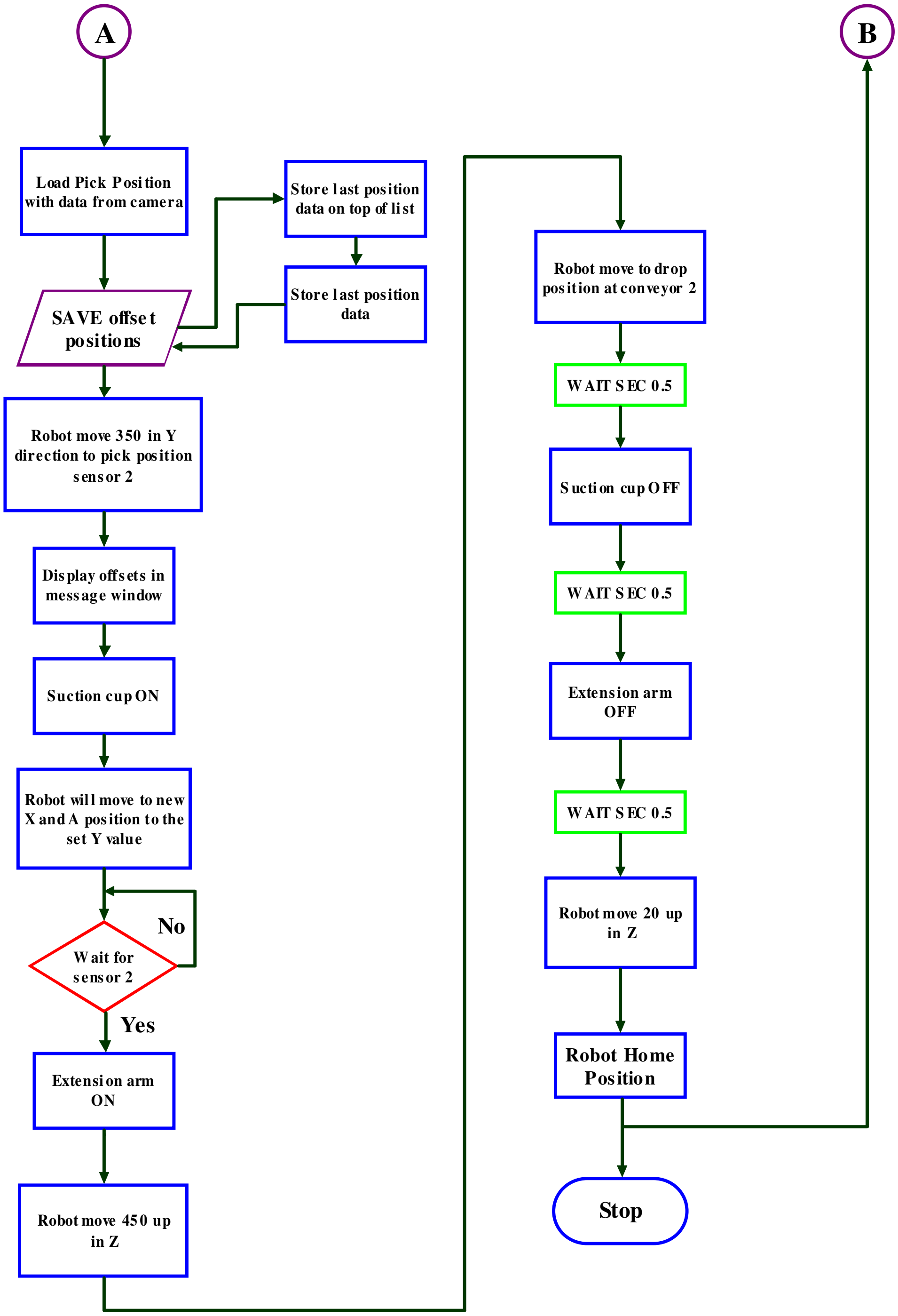

A
Load Pick Position with data from camera
Store last position data on top of list
Store last position data
SAVE offset positions
Robot move 350 in Y direction to pick position sensor 2
Display offsets in message window
Suction cup ON
Robot will move to new X and A position to the set Y value
No
Wait for sensor 2
Yes
Extension arm ON
Robot move 450 up in Z
Robot move to drop position at conveyor 2
WAIT SEC 0.5
Suction cup OFF
WAIT SEC 0.5
Extension arm OFF
WAIT SEC 0.5
Robot move 20 up in Z
Robot Home Position
Stop
B

Figure 5-22 Test 4 flow chart 2

5.4.3 Benchmark Tests

For the bench mark test a total of 25 parts were passed along conveyor 1, from which nine parts had to be picked and placed on conveyor 2, while the other 16 parts continued along conveyor 1 and had to be dumped in a "parts of no interest" bin. The test run was conducted at the first speed of conveyor 1 of 26 Hz at 8 m/min. The test speed was selected by pressing the soft key on the system SCADA. Initial test runs were made at the slowest speed of conveyor 1 of 4 m/min before the benchmark tests were conducted. The results of the benchmark test are shown in Table 4.

Table 4 Benchmark test

Number of runs made	Number of parts to be inspected	Number of good parts	Number of bad parts	Number of good parts picked and placed	Percentage accuracy of good parts selection
1	25	9	16	9	100%
2	25	9	16	9	100%
3	25	9	16	9	100%
4	25	9	16	8	96%
5	25	9	16	9	100%
6	25	9	16	9	100%
7	25	9	16	9	100%
8	25	9	16	9	100%
9	25	9	16	9	100%
10	25	9	16	9	100%

Total	Total	Total	Total	Total	Average
10	250	90	160	89	99%

The benchmark test indicated 99% accuracy. The only possible reason for the miss pick in run 4 was that pick sensor 2 had to be slightly adjusted.

5.4.4 Summary

All the immediate problems at hand were dealt with and the benchmark test proved the effectiveness of the two programs. Based on the results of the benchmark test, it can be concluded that test 4 had a very high accuracy and that all the intended conditions were met.

5.5 TEST 5

5.5.1 Vision Program V1.2

Vision program V1.1 was kept and only the new pattern that was to be searched was trained for the sensor to find. The unique complex pattern was replaced by a circular pattern pasted onto the top of the object to be picked. The field of view of the camera was increased by raising the inspection position on the Z axis. This meant that the camera had a full field of view of the conveyor and could inspect and locate patterns anywhere on the conveyor on the inspection position.

5.5.2 Robot Program V1.4

The three different speed options of the conveyor were now addressed. The robot was now required to move at the same speed as the conveyor and pick the patterns correctly. The operator selected the three speed options by pressing a soft key on the SCADA of the system. This indicated to the robot the speed at which the conveyor was moving.

The robot was also required to start in auto external mode. This was done by hard-wiring 8 bits from the PLC’s outputs to the robot's I/O module and 5 bits from the robot's outputs to the PLC. After programming both sides of the system, the robot could now be started and the program selected externally by the PLC.

The V1.4 robot program was an improvement on the V1.3. The robot’s inspection position for the camera was raised to ensure that the camera had a full field of view of the conveyor. Once the pattern is detected, the robot should move downwards linearly to the pick position.

An additional function was added: once the robot had inspected the pattern and moved to the pick location, a timer was started. The timer ensures that the robot does not wait endlessly for the pattern to arrive at the pick location – thus if the part is removed from the conveyor before arriving at the pick location, the robot's timer will time out after 5 seconds and move back to the inspection position.

The processing speed of the KRL program was reviewed and optimised to ensure that all motion and processing actions are completed in the fastest time possible.

Refer to the KRL flow chart for test 5 in Figure 5-23 and Figure 5-24. The full KRL program is shown in Appendix H.

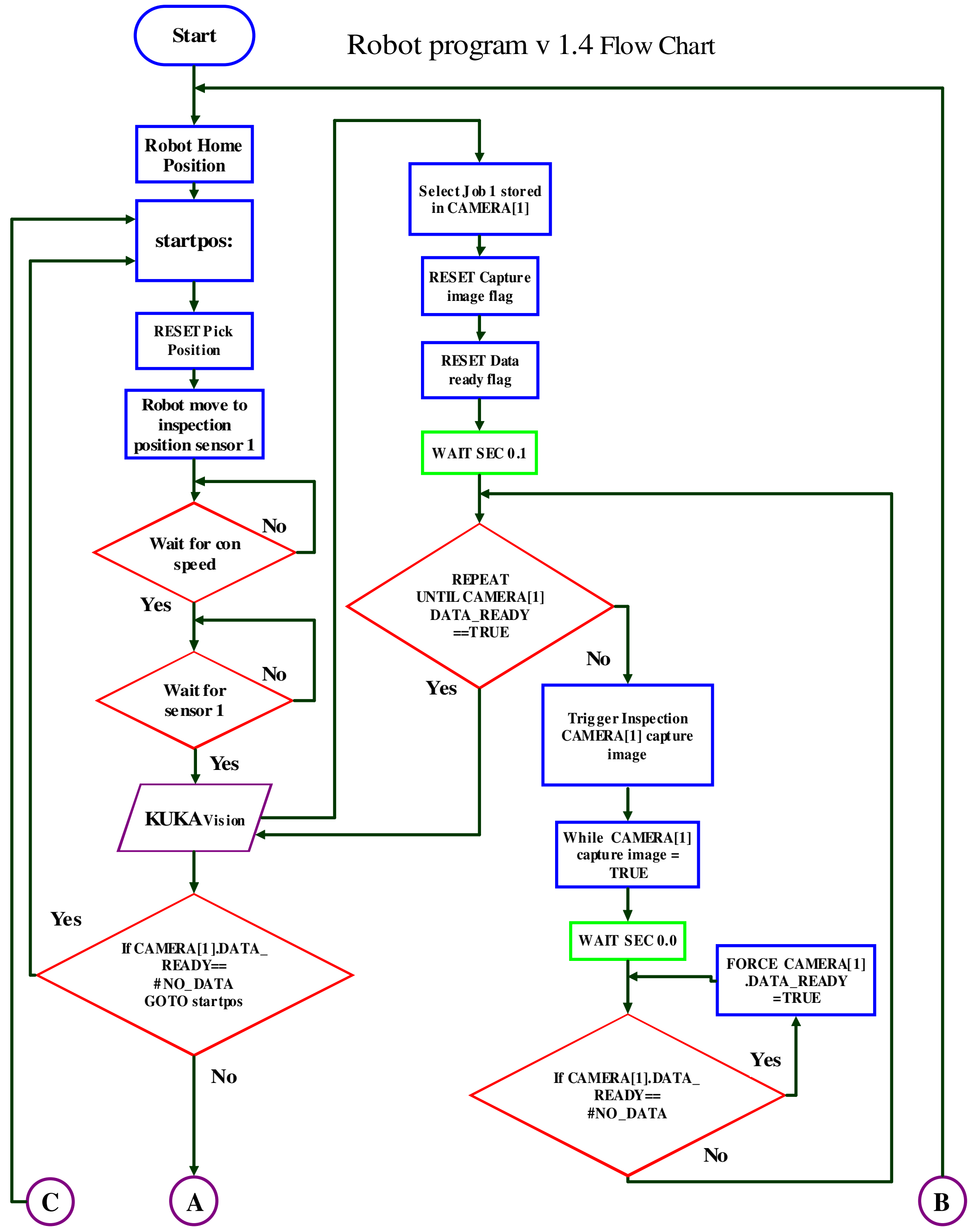

Robot program v 1.4 Flow Chart
Start
Robot Home Position
startpos:
RESET Pick Position
Robot move to inspection position sensor 1
Wait for con speed
No
Yes
Wait for sensor 1
No
Yes
KUKA Vision
Yes
If CAMERA[1].DATA_ READY== #NO_DATA GOTO startpos
No
C
A
Select Job 1 stored in CAMERA[1]
RESET Capture image flag
RESET Data ready flag
WAIT SEC 0.1
REPEAT UNTIL CAMERA[1] DATA_READY ==TRUE
No
Yes
Trigger Inspection CAMERA[1] capture image
While CAMERA[1] capture image = TRUE
WAIT SEC 0.0
FORCE CAMERA[1] .DATA_READY =TRUE
Yes
If CAMERA[1].DATA_ READY== #NO_DATA
No
B

Figure 5-23 Test 5 flow chart 1

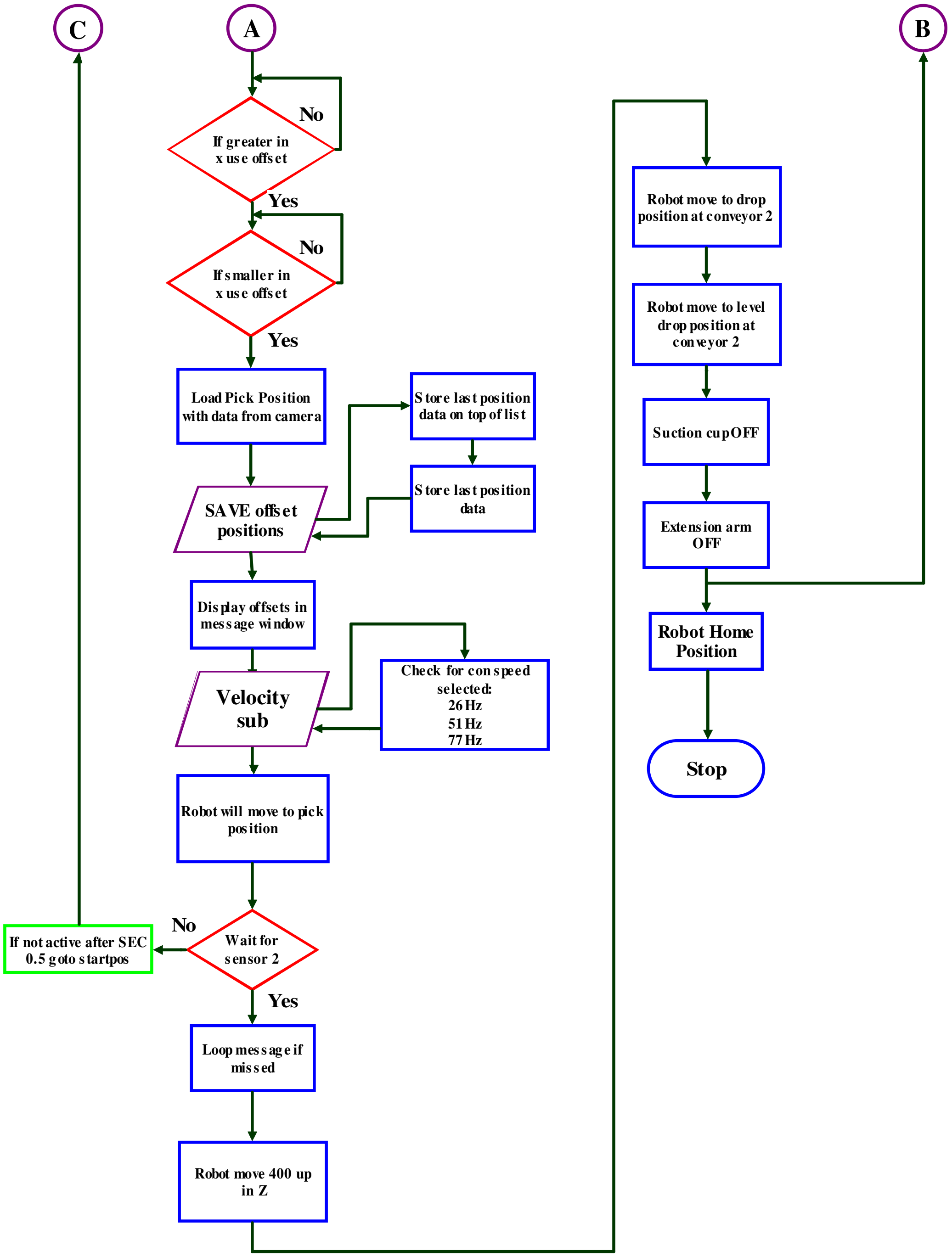

C
A
B
If greater in x use offset
No
Yes
If smaller in x use offset
No
Yes
Load Pick Position with data from camera
Store last position data on top of list
Store last position data
SAVE offset positions
Display offsets in message window
Velocity sub
Check for con speed selected: 26 Hz 51 Hz 77 Hz
Robot will move to pick position
Wait for sensor 2
No
If not active after SEC 0.5 goto startpos
Yes
Loop message if missed
Robot move 400 up in Z
Robot move to drop position at conveyor 2
Robot move to level drop position at conveyor 2
Suction cup OFF
Extension arm OFF
Robot Home Position
Stop

Figure 5-24 Test 5 flow chart 2

5.5.3 Benchmark Testing

To be able to establish how well the V1.4 robot program functioned, benchmark tests at all three speeds were carried out.

The first speed of the conveyor was 8 metres per minute or 26 Hz as indicated on the SCADA. The benchmark test results in Table 5 were excellent as all the patterns were selectively picked from the number of parts to be inspected.

Table 5 Benchmark test at 26 Hz

Number of runs made	Number of parts To be inspected	Number of good parts	Number of bad parts	Number of good parts picked and placed	Percentage accuracy of Good parts picked
1	30	10	20	10	100%
2	30	10	20	10	100%
3	30	10	20	10	100%
4	30	10	20	10	100%
5	30	10	20	10	100%
6	30	10	20	10	100%
7	30	10	20	10	100%
8	30	10	20	10	100%
9	30	10	20	10	100%
10	30	10	20	10	100%

Total	Total	Total	Total	Total	Average
10	300	100	200	100	100%

The second speed of the conveyor was 12 metres per minute or 51 Hz as indicated on the SCADA. The benchmark test results in Table 6 were excellent as all the patterns were selectively picked from the number of parts to be inspected at this speed as well.

Table 6 Benchmark test at 51 Hz

Number of runs made	**Number of parts to be inspected**	**Number of good parts**	**Number of bad parts**	**Number of good parts picked and placed**	**Percentage accuracy of good parts picked**
1	30	10	20	10	100%
2	30	10	20	10	100%
3	30	10	20	10	100%
4	30	10	20	10	100%
5	30	10	20	10	100%
6	30	10	20	10	100%
7	30	10	20	10	100%
8	30	10	20	10	100%
9	30	10	20	10	100%
10	30	10	20	10	100%

Total	Total	Total	Total	Total	Average
10	300	100	200	100	100%

The third speed of the conveyor was 12 metres per minute or 77 Hz as indicated on the SCADA. The benchmark test results in Table **7** were excellent, and only one pattern was missed from the number of parts inspected, thus indicating that the new program functioned very effectively.

Table 7 Benchmark test at 77 Hz

Number of runs made	**Number of parts to be inspected**	**Number of good parts**	**Number of bad parts**	**Number of good parts picked and placed**	**Percentage accuracy of good parts picked**
1	30	10	20	9	96%
2	30	10	20	10	100%
3	30	10	20	10	100%
4	30	10	20	10	100%
5	30	10	20	10	100%
6	30	10	20	10	100%
7	30	10	20	10	100%
8	30	10	20	10	100%
9	30	10	20	10	100%
10	30	10	20	10	100%

Total	Total	Total	Total	Total	Average
10	300	100	200	99	99%

The benchmark tests indicated that the VGR had an accuracy of 99% for all the tests combined.

5.5.4 Conclusion

After all the new requirements had been implemented and the benchmark testing had been completed, it was established that the VGR was fully functional at an accuracy rating of 99%. From the functionality of the VGR with the added functions, it can be concluded that all the conditions set were successfully met in the final test.

6 Conclusion

6.1 Adding Functionality to the Robot and its Environment

- A mounting bracket for the camera head was manufactured and fixed to the gripper, which in turn was mounted on the flange of the robot.
- Two suction cup extension devices to pick objects were manufactured and fixed to the gripper. The first was too flimsy, so a second improved design was made.
- The pneumatic expansion module to control all external pneumatic devices was fixed to the robot just below the In-Sight 5400R module.
- A mounting plate was cut and bent to fix the In-Sight 5400R module to the robot.
- The breakout board for the In-Sight 5400R module was mounted in the KUKA cabinet and the 24 V dc power was drawn from a power supply in the cabinet.
- Two sensors used to trigger the robot as well as inputs received from the PLC and the pneumatic expansion modules were hardwired to the BECKHOFF I/O module located in the KUKA cabinet.
- Air was supplied at a pressure of 6 psi and regulated by a pneumatic filter to add functionality to the pneumatic devices.

6.2 Utilising Vision for Component Detection and Inspection

The initial commissioning stages of the COGNEX In-Sight 5400R module consisted of self-training. The reason for this was that first the operation of the vision system and how to write the program to be used for detection and inspection had to be clearly understood. To aid with the initial stages of inspection, a frame was fabricated from aluminium. This aided as a simulated environment on which the vision system was tested (Figure 6-1 Test frame). Once the vision system had been successfully commissioned on the test frame and was well understood, it could be integrated into the robot's environment.

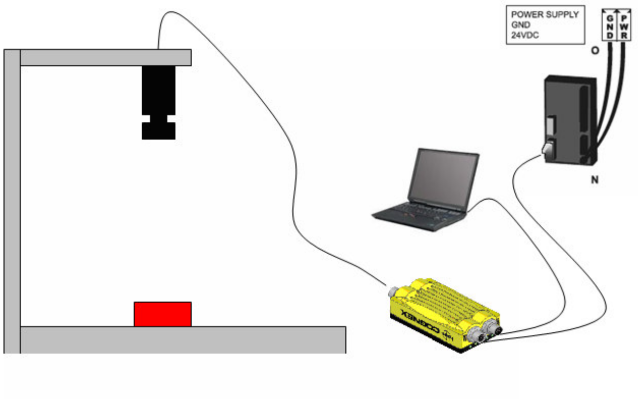


Figure 6-1 Test frame for vision system

6.3 Interfacing Vision with the Robot System

The interfacing KUKA Vision software was installed and commissioned so that data received from the In-Sight camera could be converted into KUKA Robot Logic (KRL). This was done so that the robot could understand and use the data received from the In-Sight camera. The interfacing software required the camera's specified Internet protocol (IP) address to have access to the In-Sight camera and receive data from it. The cell location from where to read the data in the In-Sight Explorer software running in the In-Sight module had to be specified as well. The data received from the cell relayed the location information of the object to the robot. This information was then used to guide the robot's tool centre point (TCP) to the part to be picked.

6.4 Calibration of Vision and Robot system

6.4.1 Vision System Calibration

The vision system had to be calibrated to avoid radial distortion caused by the lens when parts were inspected and distortion possibly caused by the mounting configurations of the camera head being at a slight angle. To calibrate the camera head a grid of dots was used with no fiducial to indicate the origin, the positive X and the positive Y direction for the camera. The reason for this was that the orientation directions were manually indicated so that the robot's base calibration coincided with that of the camera. After calibrating the vision system, real-world measurements could be made of parts inspected. Once the vision system orientation directions had been specified and calibrated, the next step was to calibrate the robot's base to the same orientation directions as that of the vision system. The grid of dots used for calibration should not be removed as the robot's base is calibrated according to the same grid.

6.4.2 Robot Calibration

It is important first to train the tool to be used for the pick application so that the robot will know its location within its real-world co-ordinates. For the pick-and-place application a suction cup mounted on an extension arm was used. A brass spike was machined so that it could be fitted in place of the suction cup. This gave a true indication of the exact centre point of the suction cup, and was used to calibrate the TCP to the centre point of the base spike located anywhere in space. A 4-point method was used to train the TCP from four different directions in relation to the base spike, as explained in section 4.3.1. Once trained, the TCP is named and saved. The robot's new TCP was successfully trained.

Once the TCP had been trained the base of the TCP had to be calibrated. Using the grid of dots used to calibrate the vision system, the robot tool with the brass spike still in place of the suction cup was driven to the origin, and this was used as the vision system's origin on the grid of dots. A 3-point method was used to calibrate the base. The pattern of the base calibration was in the shape of an L.

Once the TCP and base of the new tool had been calibrated and coincided with the calibration co-ordinates of the vision system, the programming of the robot was started. It is important to note that when programming motions for the robot, the trained TCP and calibrated base were used so that the robot would know the location orientation co-ordinates of the new tool in the real world. The brass spike was removed and the suction cup replaced.

6.5 Using Vision Results for Stationary Inspection Pick

Once the vision system calibration and robot calibration had been completed and the interfacing software successfully integrated, an inspection was attempted. A simple program for the robot and vision system was used to determine whether the two different software components were functioning correctly via the interfacing software.

The inspection worked – the location data of the part was displayed on the KUKA Vision results tab. This indicated that a stationary pick could be attempted.

6.6 Using VGR for Stationary Pick

The original program for the vision system was used and functionality was added to the robot program so that the tool to be used for the pick was capable of motion. The robot successfully inspected and picked the object from stationary conveyor 1 and placed it at its designated location on conveyor 2. A pick could then be attempted from a moving conveyor.

6.7 Using VGR to pick objects from a moving conveyor

The vision program was not altered, but the robot program was developed further. The robot could now move to a pick location from the inspection position to pick the object from the moving conveyor 1. This motion to the pick location allowed the tool to be orientated so that it could pick the object. The pick from conveyor 1 while it was in motion and the place on conveyor 2 were successful, and this paved the way for a more refined program for both systems to be written and problem areas to be addressed.

6.8 Using VGR to selectively pick trained objects from a moving conveyor

The vision program was altered as a different pattern was placed on the blocks to be inspected. The change was made when it was discovered that the old pattern was being detected in the slats of conveyor 1. The new pattern was of a unique design to limit the chances of it being found anywhere else except on the blocks to be picked.

The KRL program would only fully execute the KRL vision subprogram if all conditions were true. When a condition was found to be false, the subprogram would go into a continuous loop. An additional piece of KRL programming was added to the subprogram to deal with the false condition. When the condition is false the KLR is forced to exit the vision subprogram and run, and if the statement to recheck whether the condition is false, the program will go from here to the start position, which will restart the inspection cycle when the camera is triggered again.

The robot was now programmed to let parts of no interest pass and continue along the conveyor, and only picked parts of interest from the conveyor.

A benchmark test was run to determine the accuracy and effectiveness of the two new programs, and the test results gave a 99% average. This proved that the VGR was very effective in the environment in which it functioned.

6.9 Final VGR program

From the final VGR program three different conveyor speeds could be selected. These conveyor speeds were hardwired from the PLC to the robot to ensure that the robot's linear motion speed would coincide with the speed at which the conveyor was running. The three different conveyor speed options could be selected from the system's SCADA by pressing one of three speed soft keys.

The robot was programmed to start in auto external mode. This was done by hardwiring 8 bits from the PLC’s outputs to the robot's I/O module and 5 bits from the robot's outputs to the PLC. After programming both parts of the system, the robot could be started and the program selected externally via the PLC.

The KRL program was optimised with additional functionality and improvements were made in processing time.

Based on the final benchmark tests, it can be concluded that all the conditions stipulated in the hypothesis and additional functionality were met. The conclusion is supported by the successful results.

6.10 Future Research

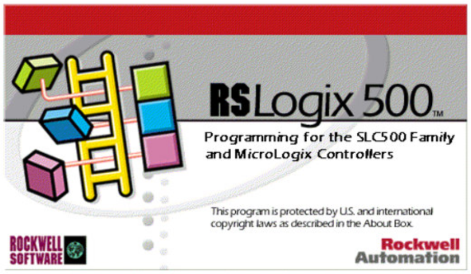
Future research should be focused on the development, improvement and utilisation of the following:

- Different lighting techniques and types should be tested to determine which work best when illuminating objects on the glossy surface of a conveyor.
- A conveyor with a matt surface would eliminate reflection of light into the camera lens. Similarly, a conveyor system utilising a belt rather than slats should also be considered.
- Different tools can be used to pick different objects which would make the robot's end effecter multi-functional and far more versatile.
- The robot's speed can be synchronised with that of the conveyor by using a very accurate conveyor package supplied by KUKA, but this is very costly.
- The vision system used is very flexible. It can also be used to inspect barcodes, to measure objects accurately and to check for consistency in size and shape. It can also be used for Optical Character Recognition.
- Guidelines can be established to ensure that all factors influencing the implementation and operation of a VGR system are considered before implementation.
- The VGR can be commissioned to function in a bin pick-and-place environment.

It took many months of research before a final decision was made on the different hardware and software combinations for the working VGR. Various tools were changed, modified and redesigned to improve on previous designs. Programs for the vision and robot systems were written and rewritten until a functioning base line was established. Problems were considered as the norm but they could always be solved. A working VGR has now been commissioned and all the conditions set have been met, thus opening the way for a variety of applications for this system.

7 Appendix A

RSLogix500 Project Report

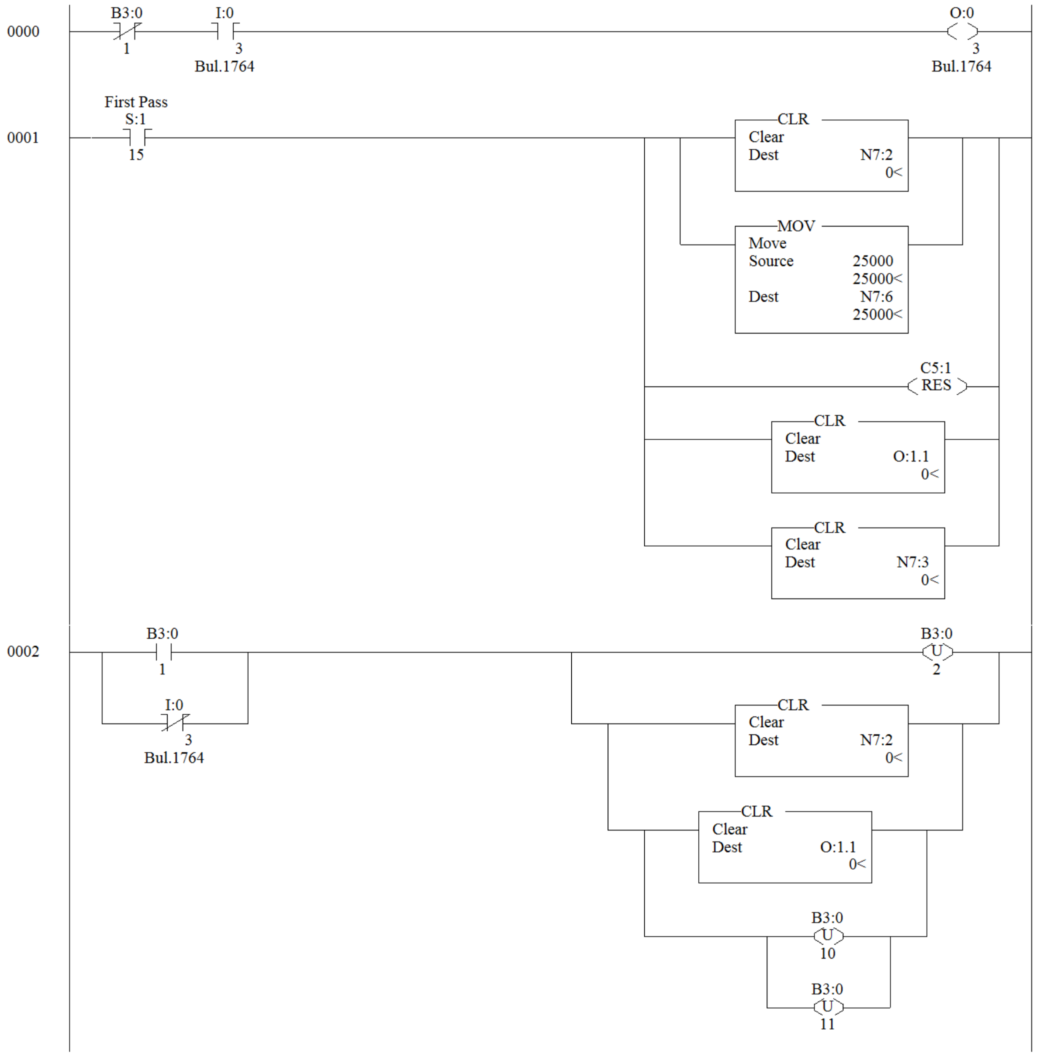


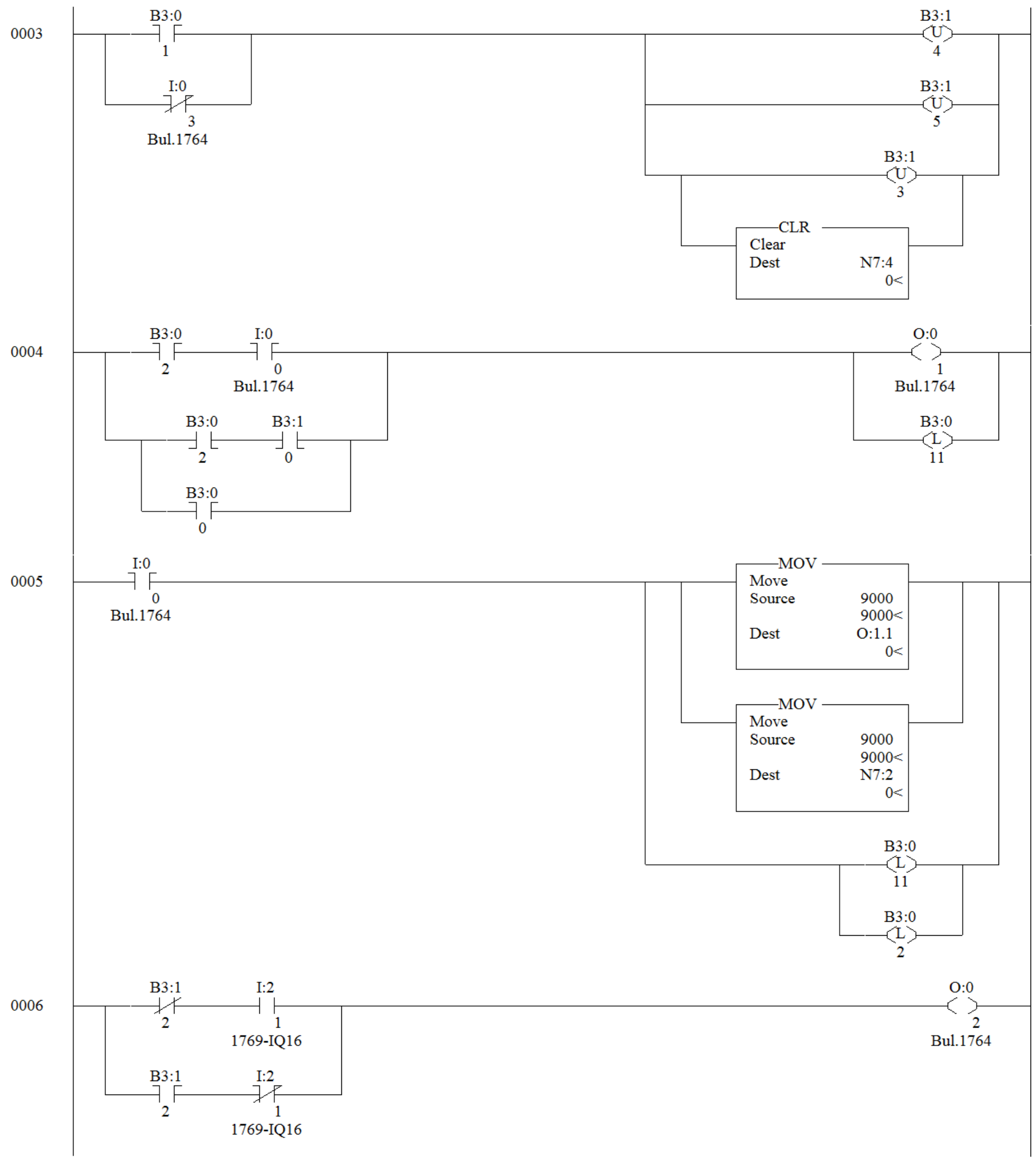
RSLogix 500™

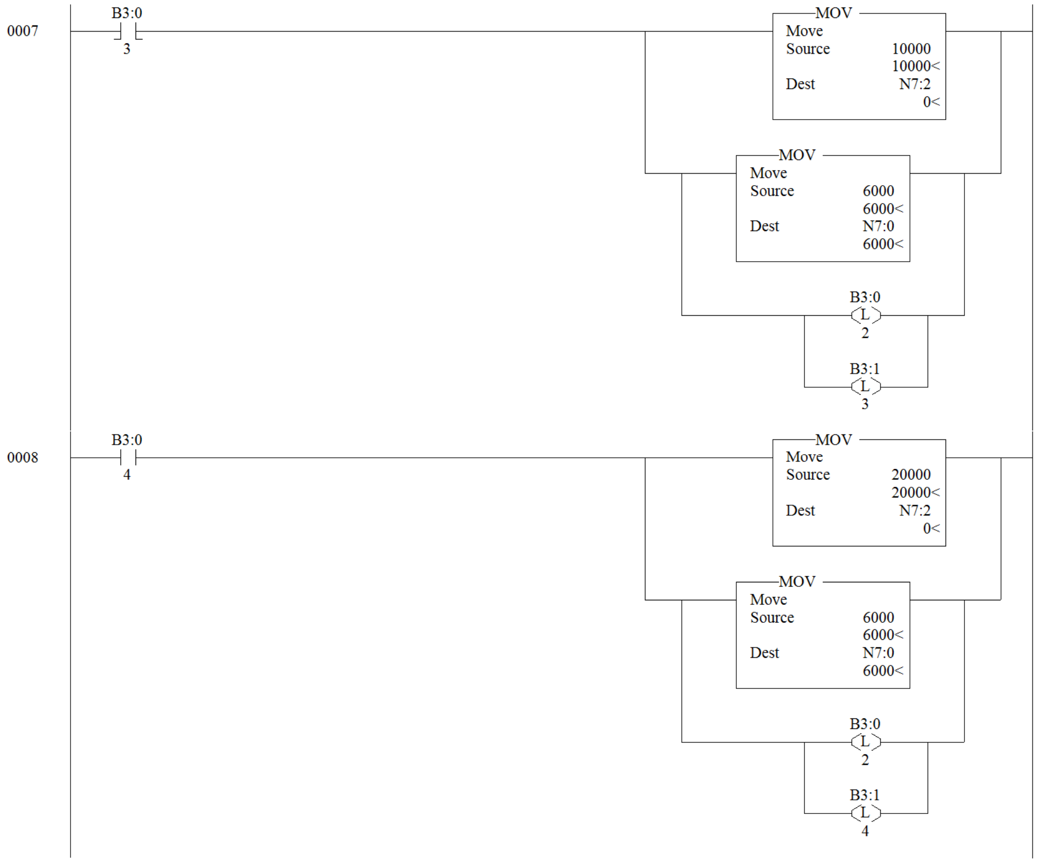
Programming for the SLC500 Family and MicroLogix Controllers

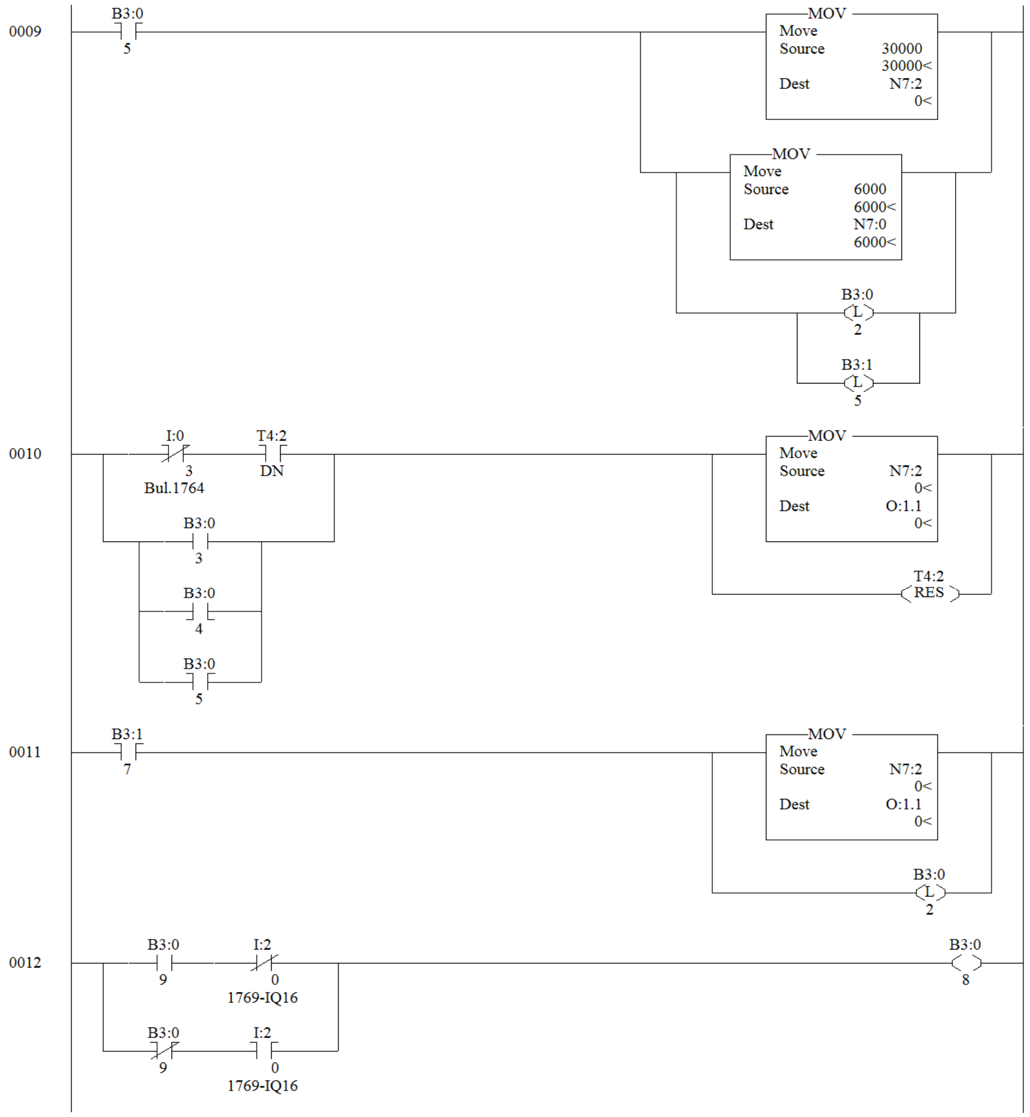
This program is protected by U.S. and international copyright laws as described in the About Box.

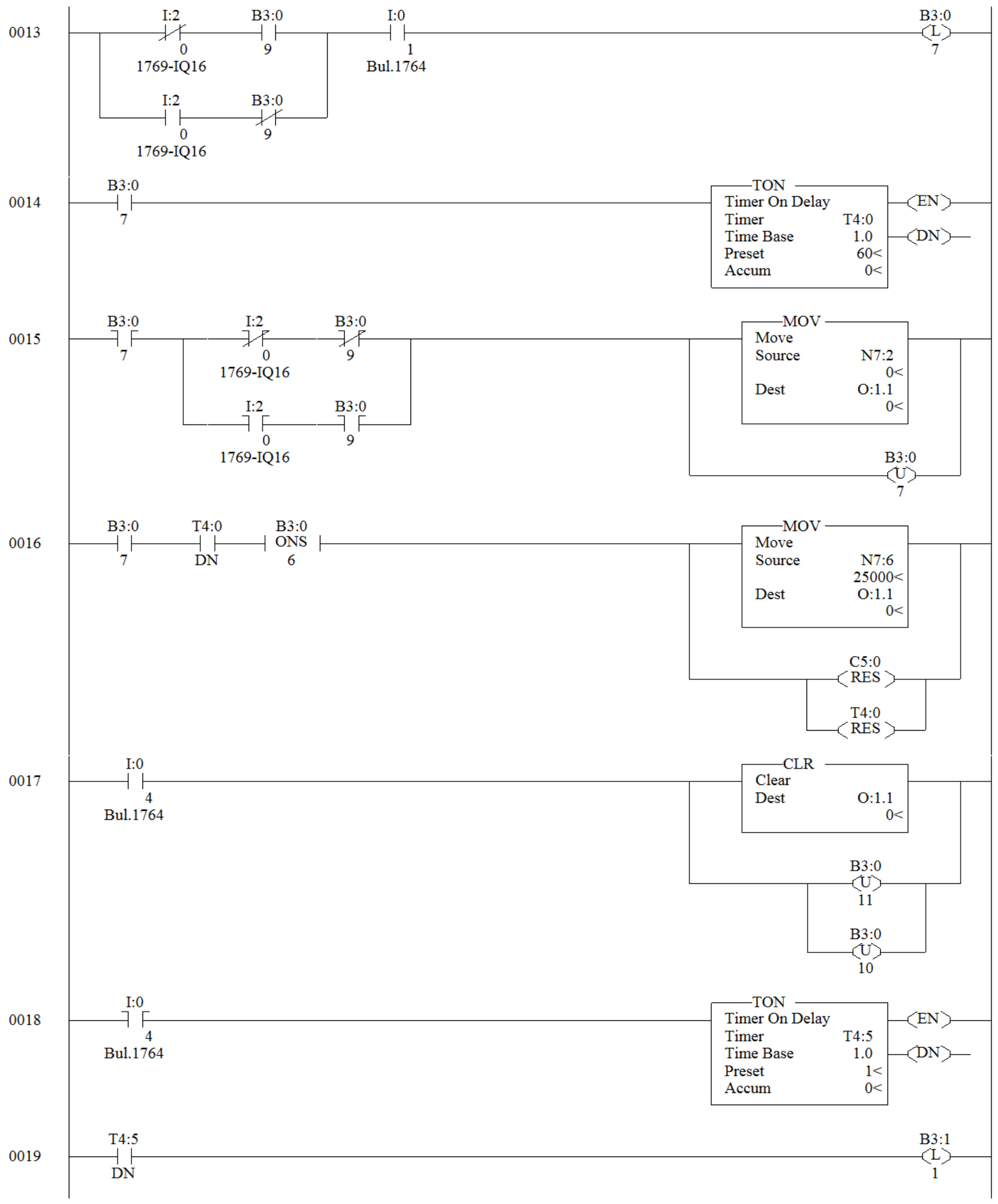
```
Processor Type: Bul.1764     Micrologix 1500 LSP Series C

Processor Name: ML1500

Total Memory Used: 850 Instruction Words Used - 199 Data Table Words Used

Total Memory Left: 4846 Instruction Words Left

Program Files: 3

Data Files: 9

Program ID: ce08
```

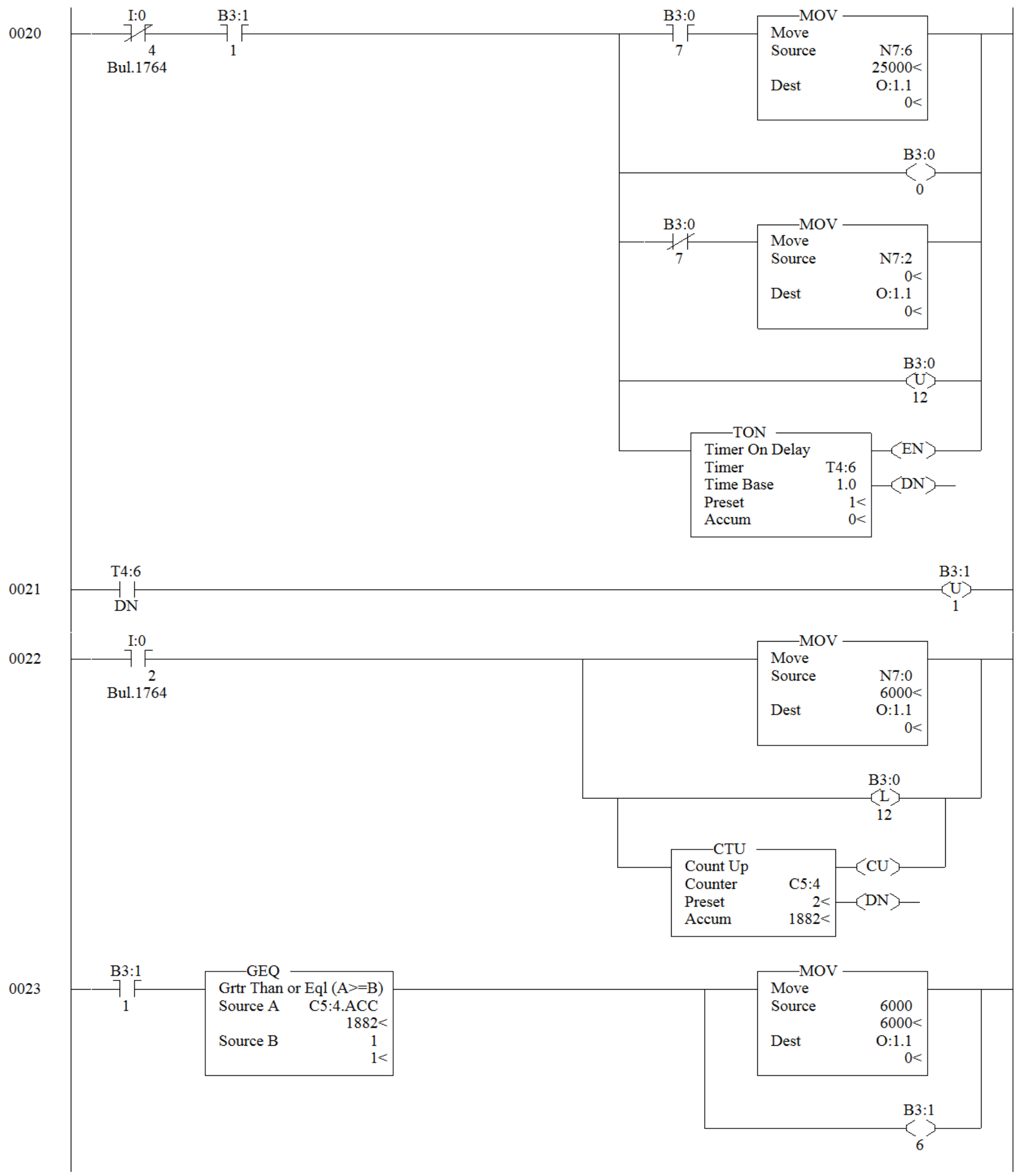

0000
B3:0
1
I:0
3
Bul.1764
O:0
3
Bul.1764
0001
First Pass
S:1
15
CLR
Clear
Dest N7:2
0<
MOV
Move
Source 25000
25000<
Dest N7:6
25000<
C5:1
RES
CLR
Clear
Dest O:1.1
0<
CLR
Clear
Dest N7:3
0<
0002
B3:0
1
I:0
3
Bul.1764
B3:0
U
2
CLR
Clear
Dest N7:2
0<
CLR
Clear
Dest O:1.1
0<
B3:0
U
10
B3:0
U
11

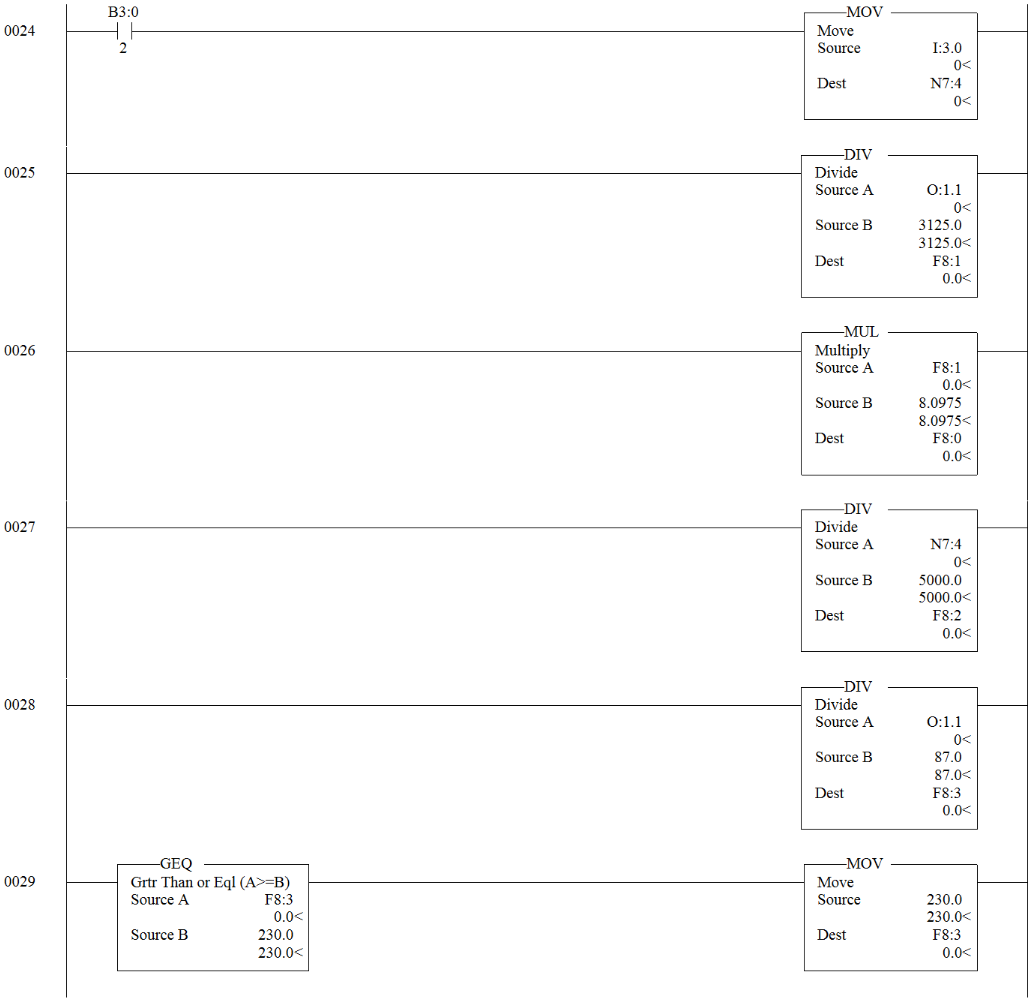

0003
B3:0
1
I:0
3
Bul.1764
B3:1
U
4
B3:1
U
5
B3:1
U
3
CLR
Clear
Dest N7:4
0<
0004
B3:0
2
I:0
0
Bul.1764
B3:0
2
B3:1
0
B3:0
0
O:0
1
Bul.1764
B3:0
L
11
0005
I:0
0
Bul.1764
MOV
Move
Source 9000
9000<
Dest O:1.1
0<
MOV
Move
Source 9000
9000<
Dest N7:2
0<
B3:0
L
11
B3:0
L
2
0006
B3:1
2
I:2
1
1769-IQ16
B3:1
2
I:2
1
1769-IQ16
O:0
2
Bul.1764


0007
B3:0
3
MOV
Move
Source 10000
10000<
Dest N7:2
0<
MOV
Move
Source 6000
6000<
Dest N7:0
6000<
B3:0
L
2
B3:1
L
3
0008
B3:0
4
MOV
Move
Source 20000
20000<
Dest N7:2
0<
MOV
Move
Source 6000
6000<
Dest N7:0
6000<
B3:0
L
2
B3:1
L
4

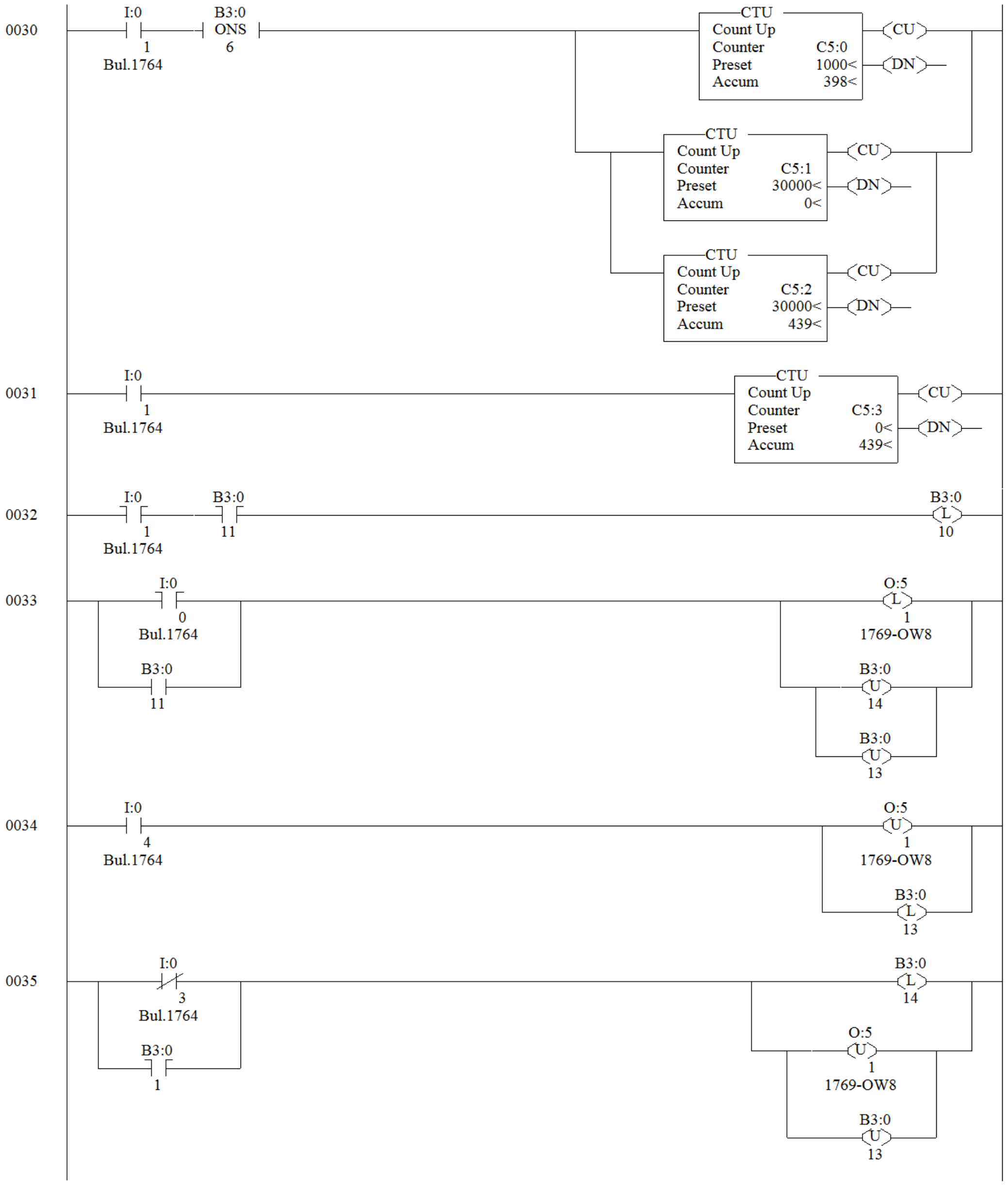

0009
B3:0
5
MOV
Move
Source 30000
30000<
Dest N7:2
0<
MOV
Move
Source 6000
6000<
Dest N7:0
6000<
B3:0
L
2
B3:1
L
5
0010
I:0
3
Bul.1764
T4:2
DN
B3:0
3
B3:0
4
B3:0
5
MOV
Move
Source N7:2
0<
Dest O:1.1
0<
T4:2
RES
0011
B3:1
7
MOV
Move
Source N7:2
0<
Dest O:1.1
0<
B3:0
L
2
0012
B3:0
9
I:2
0
1769-IQ16
B3:0
9
I:2
0
1769-IQ16
B3:0
8

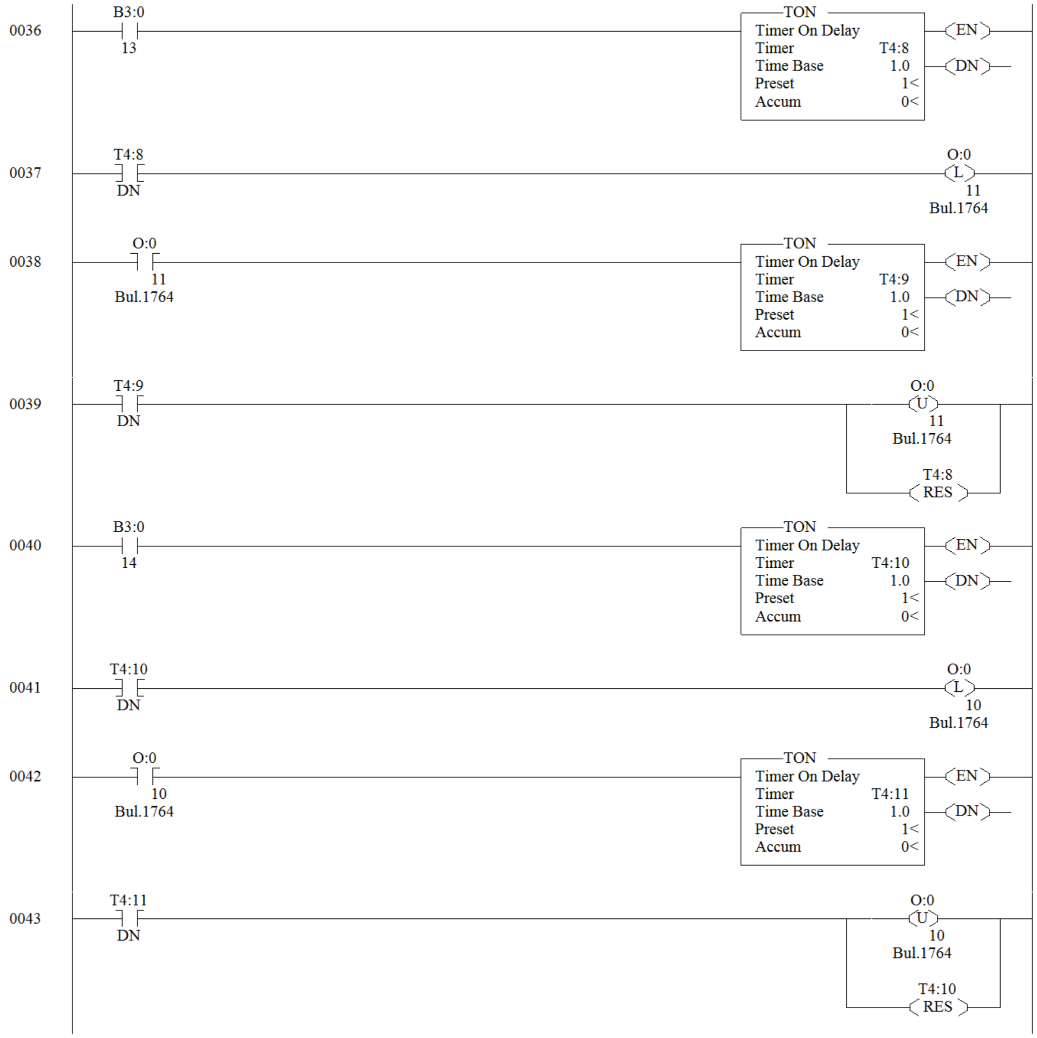

0013
I:2
0
1769-IQ16
B3:0
9
I:0
1
Bul.1764
B3:0
L
7
I:2
0
1769-IQ16
B3:0
9
0014
B3:0
7
TON
Timer On Delay
Timer T4:0
Time Base 1.0
Preset 60<
Accum 0<
EN
DN
0015
B3:0
7
I:2
0
1769-IQ16
B3:0
9
I:2
0
1769-IQ16
B3:0
9
MOV
Move
Source N7:2
0<
Dest O:1.1
0<
B3:0
U
7
0016
B3:0
7
T4:0
DN
B3:0
ONS
6
MOV
Move
Source N7:6
25000<
Dest O:1.1
0<
C5:0
RES
T4:0
RES
0017
I:0
4
Bul.1764
CLR
Clear
Dest O:1.1
0<
B3:0
U
11
B3:0
U
10
0018
I:0
4
Bul.1764
TON
Timer On Delay
Timer T4:5
Time Base 1.0
Preset 1<
Accum 0<
EN
DN
0019
T4:5
DN
B3:1
L
1

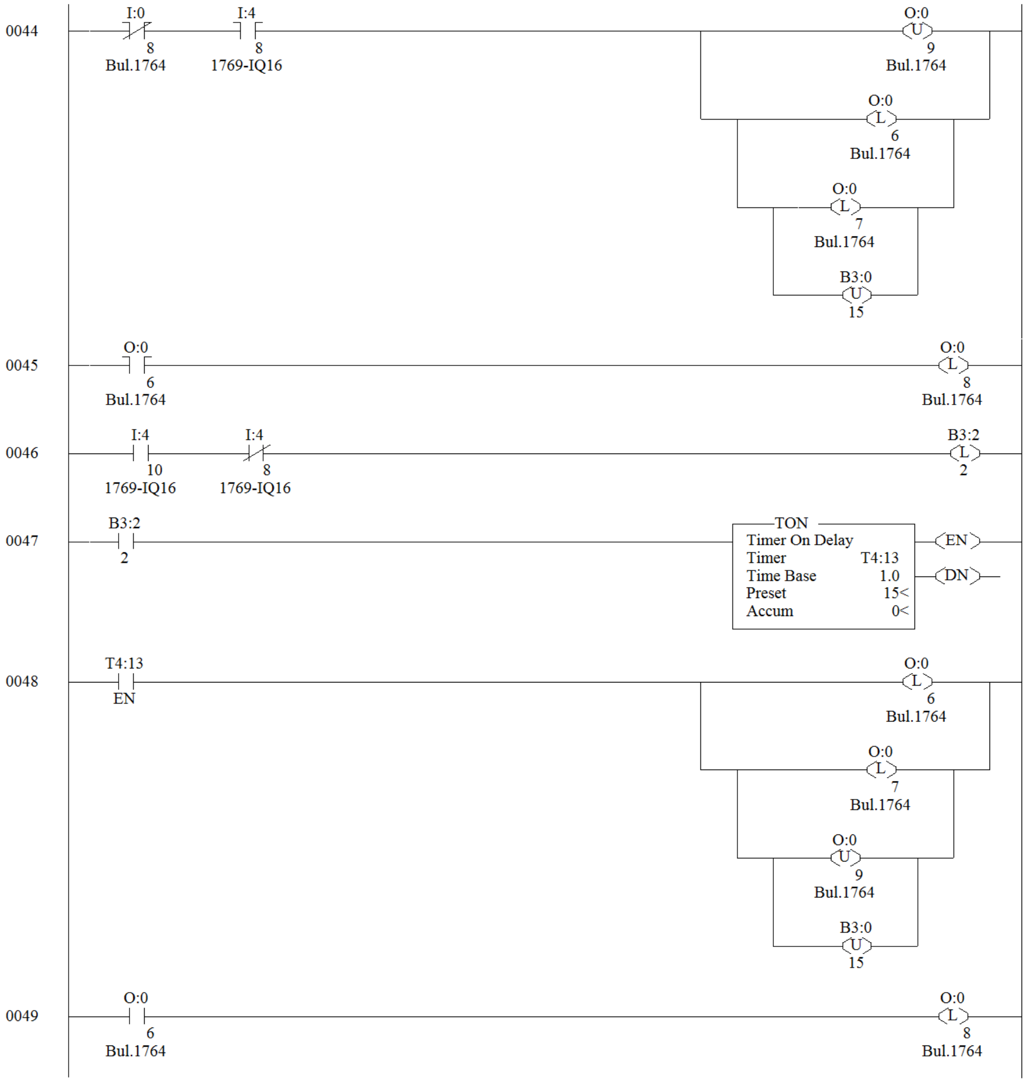

0020
I:0
4
Bul.1764
B3:1
1
B3:0
7
MOV
Move
Source N7:6
25000<
Dest O:1.1
0<
B3:0
0
B3:0
7
MOV
Move
Source N7:2
0<
Dest O:1.1
0<
B3:0
U
12
TON
Timer On Delay
Timer T4:6
Time Base 1.0
Preset 1<
Accum 0<
EN
DN
0021
T4:6
DN
B3:1
U
1
0022
I:0
2
Bul.1764
MOV
Move
Source N7:0
6000<
Dest O:1.1
0<
B3:0
L
12
CTU
Count Up
Counter C5:4
Preset 2<
Accum 1882<
CU
DN
0023
B3:1
1
GEQ
Grtr Than or Eql (A>=B)
Source A C5:4.ACC
1882<
Source B 1
1<
MOV
Move
Source 6000
6000<
Dest O:1.1
0<
B3:1
6

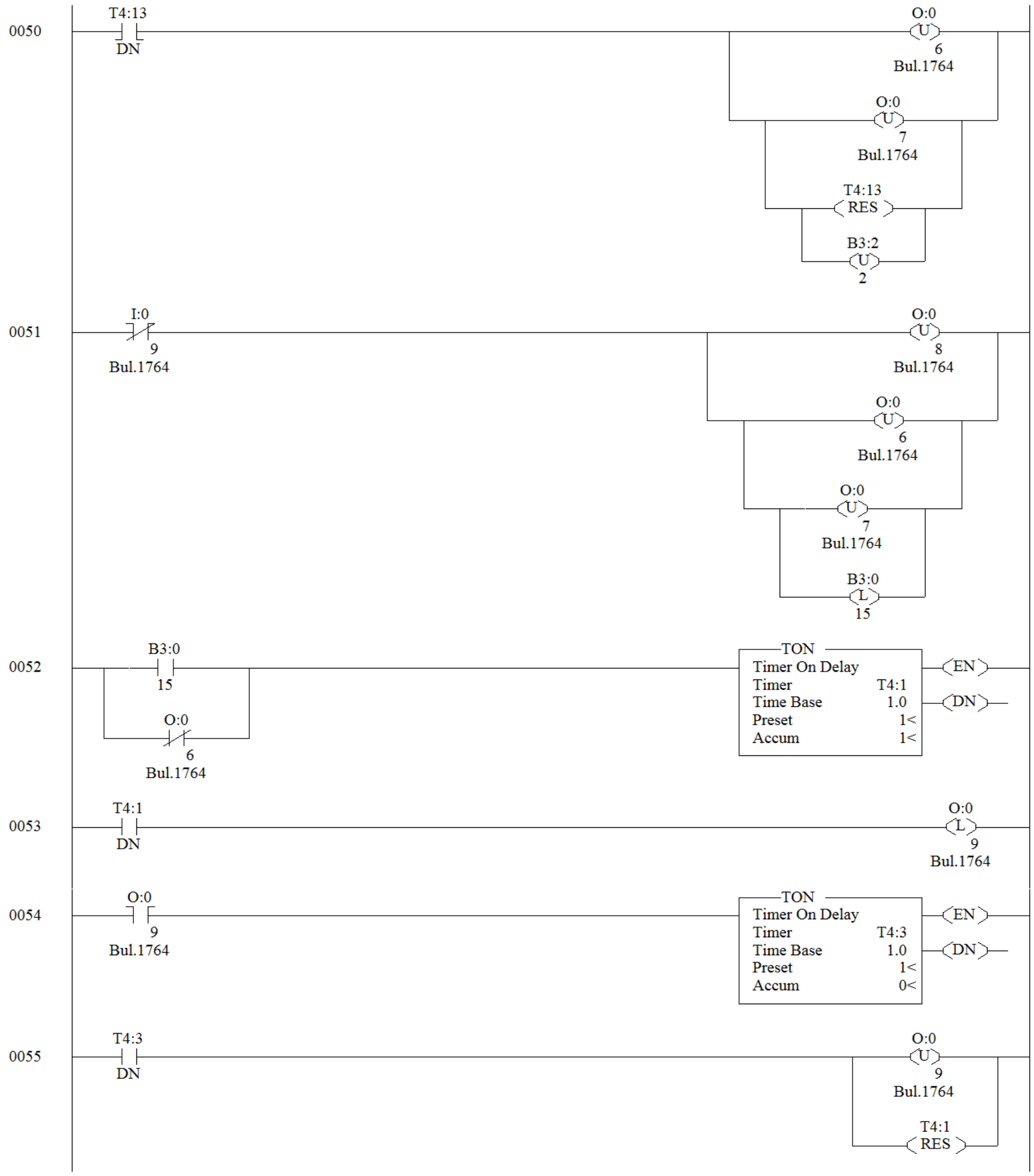


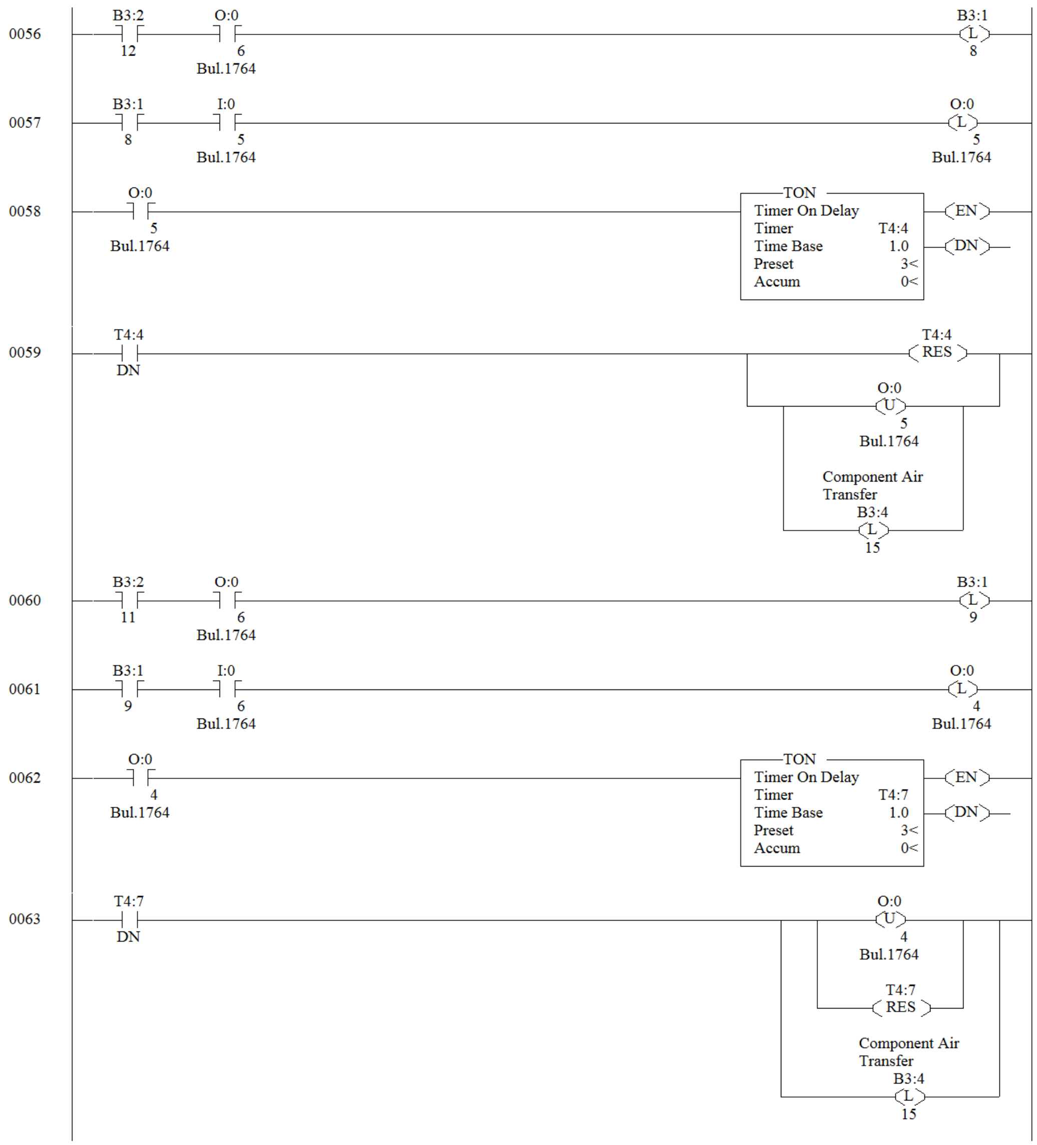
```
0024  B3:0 ] [ 2                         MOV Move         Source I:3.0  0<
                                                          Dest   N7:4   0<

0025                                     DIV Divide       Source A O:1.1    0<
                                                          Source B 3125.0   3125.0<
                                                          Dest     F8:1     0.0<

0026                                     MUL Multiply     Source A F8:1     0.0<
                                                          Source B 8.0975   8.0975<
                                                          Dest     F8:0     0.0<

0027                                     DIV Divide       Source A N7:4     0<
                                                          Source B 5000.0   5000.0<
                                                          Dest     F8:2     0.0<

0028                                     DIV Divide       Source A O:1.1    0<
                                                          Source B 87.0     87.0<
                                                          Dest     F8:3     0.0<

0029  GEQ Grtr Than or Eql (A>=B)        MOV Move         Source 230.0  230.0<
      Source A F8:3   0.0<                                Dest   F8:3   0.0<
      Source B 230.0  230.0<
```

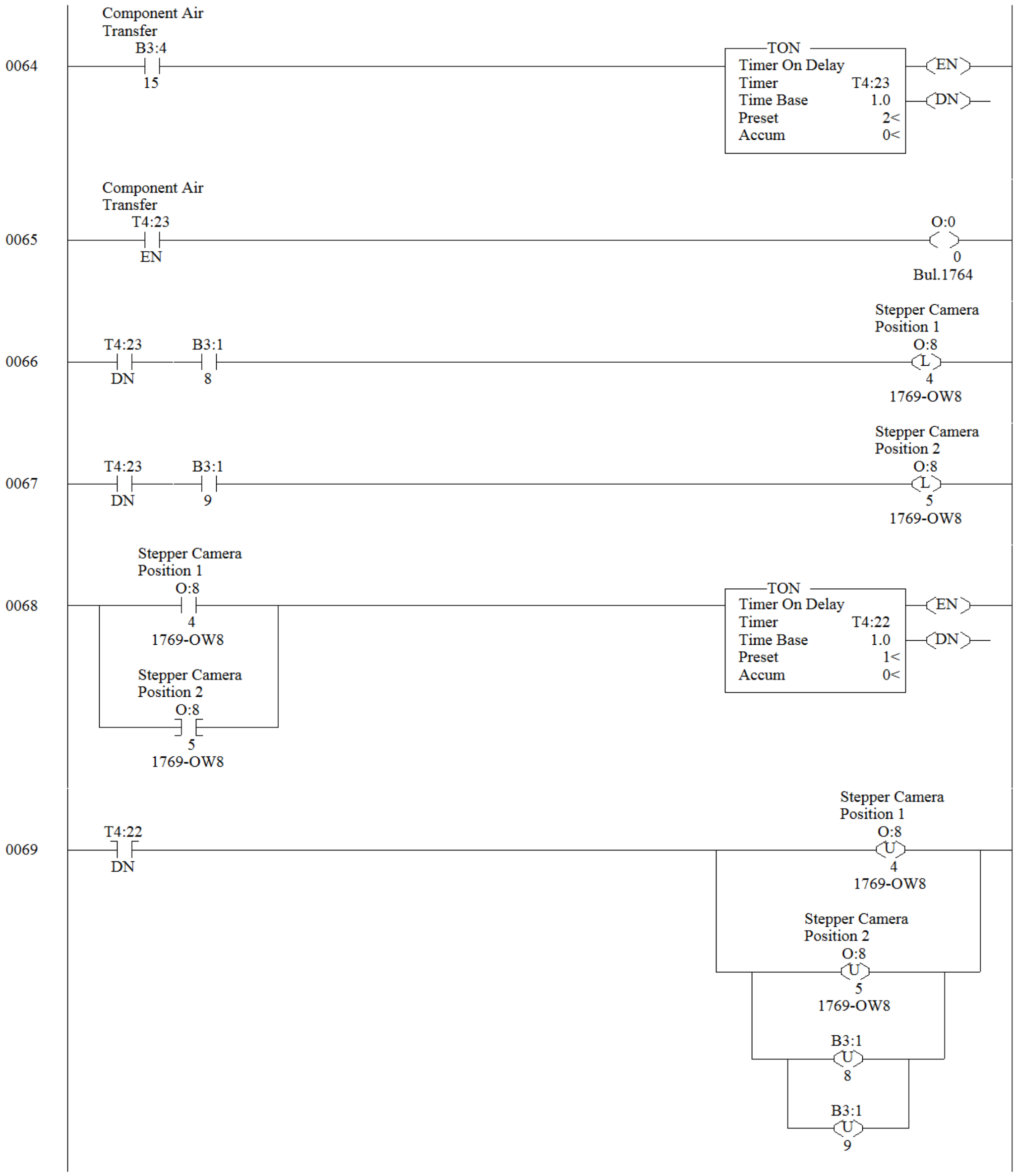

0030
I:0
1
Bul.1764
B3:0
ONS
6
CTU
Count Up
Counter C5:0
Preset 1000<
Accum 398<
CU
DN
CTU
Count Up
Counter C5:1
Preset 30000<
Accum 0<
CU
DN
CTU
Count Up
Counter C5:2
Preset 30000<
Accum 439<
CU
DN
0031
I:0
1
Bul.1764
CTU
Count Up
Counter C5:3
Preset 0<
Accum 439<
CU
DN
0032
I:0
1
Bul.1764
B3:0
11
B3:0
L
10
0033
I:0
0
Bul.1764
B3:0
11
O:5
L
1
1769-OW8
B3:0
U
14
B3:0
U
13
0034
I:0
4
Bul.1764
O:5
U
1
1769-OW8
B3:0
L
13
0035
I:0
3
Bul.1764
B3:0
1
B3:0
L
14
O:5
U
1
1769-OW8
B3:0
U
13

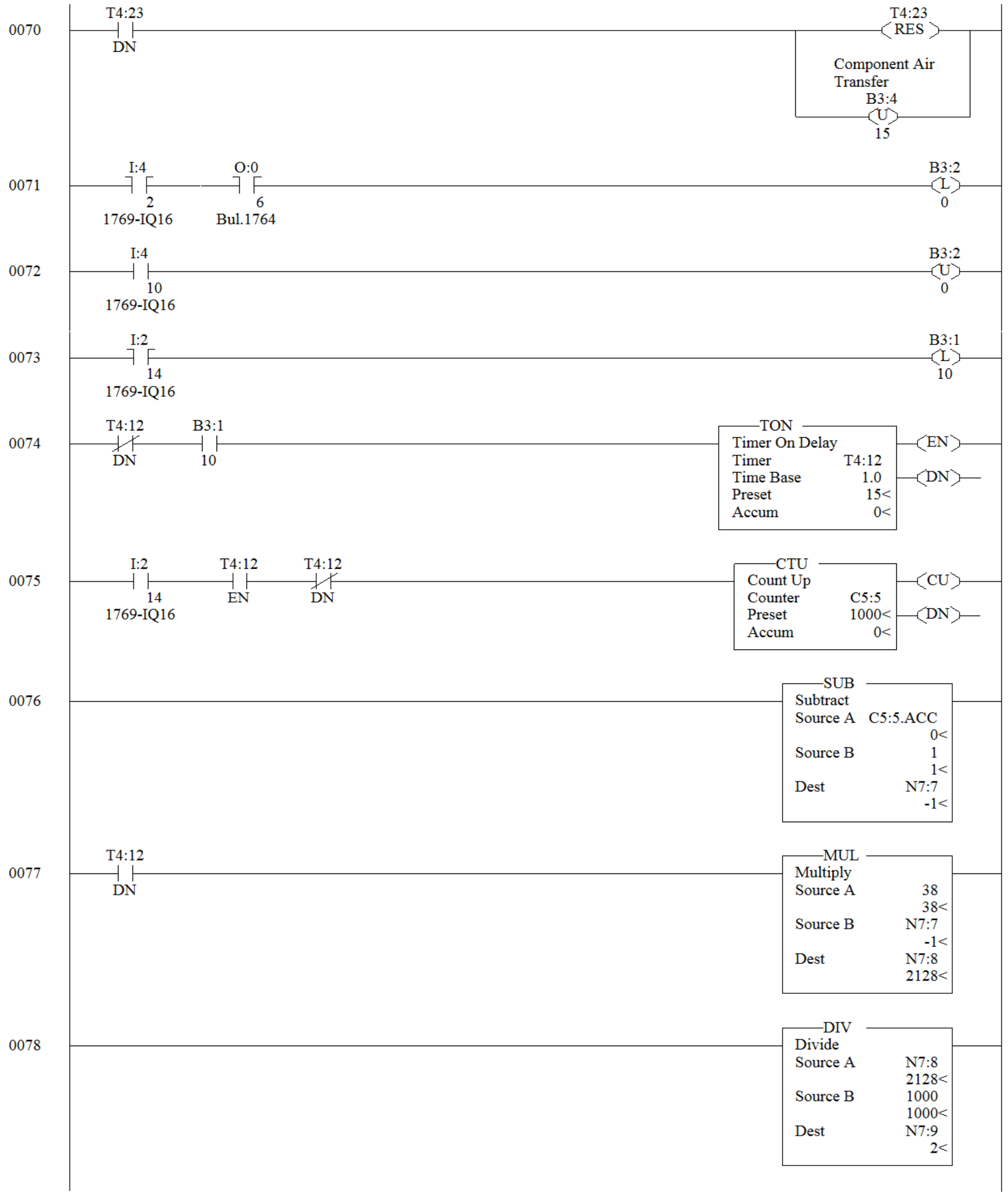

0036
B3:0
13
TON
Timer On Delay
Timer T4:8
Time Base 1.0
Preset 1<
Accum 0<
EN
DN
0037
T4:8
DN
O:0
L
11
Bul.1764
0038
O:0
11
Bul.1764
TON
Timer On Delay
Timer T4:9
Time Base 1.0
Preset 1<
Accum 0<
EN
DN
0039
T4:9
DN
O:0
U
11
Bul.1764
T4:8
RES
0040
B3:0
14
TON
Timer On Delay
Timer T4:10
Time Base 1.0
Preset 1<
Accum 0<
EN
DN
0041
T4:10
DN
O:0
L
10
Bul.1764
0042
O:0
10
Bul.1764
TON
Timer On Delay
Timer T4:11
Time Base 1.0
Preset 1<
Accum 0<
EN
DN
0043
T4:11
DN
O:0
U
10
Bul.1764
T4:10
RES

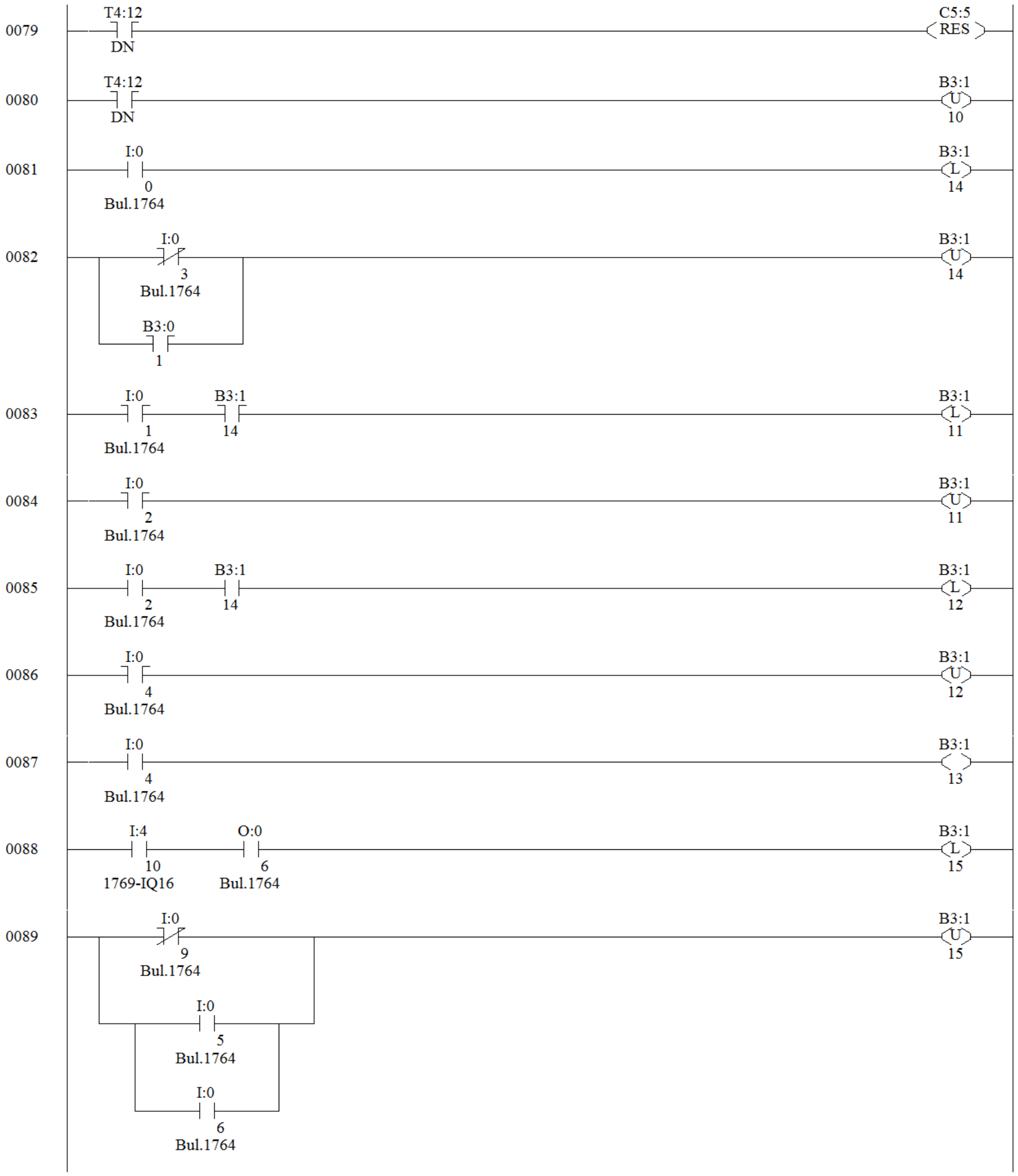

0044
I:0
8
Bul.1764
I:4
8
1769-IQ16
O:0
U
9
Bul.1764
O:0
L
6
Bul.1764
O:0
L
7
Bul.1764
B3:0
U
15
0045
O:0
6
Bul.1764
O:0
L
8
Bul.1764
0046
I:4
10
1769-IQ16
I:4
8
1769-IQ16
B3:2
L
2
0047
B3:2
2
TON
Timer On Delay
Timer T4:13
Time Base 1.0
Preset 15<
Accum 0<
EN
DN
0048
T4:13
EN
O:0
L
6
Bul.1764
O:0
L
7
Bul.1764
O:0
U
9
Bul.1764
B3:0
U
15
0049
O:0
6
Bul.1764
O:0
L
8
Bul.1764

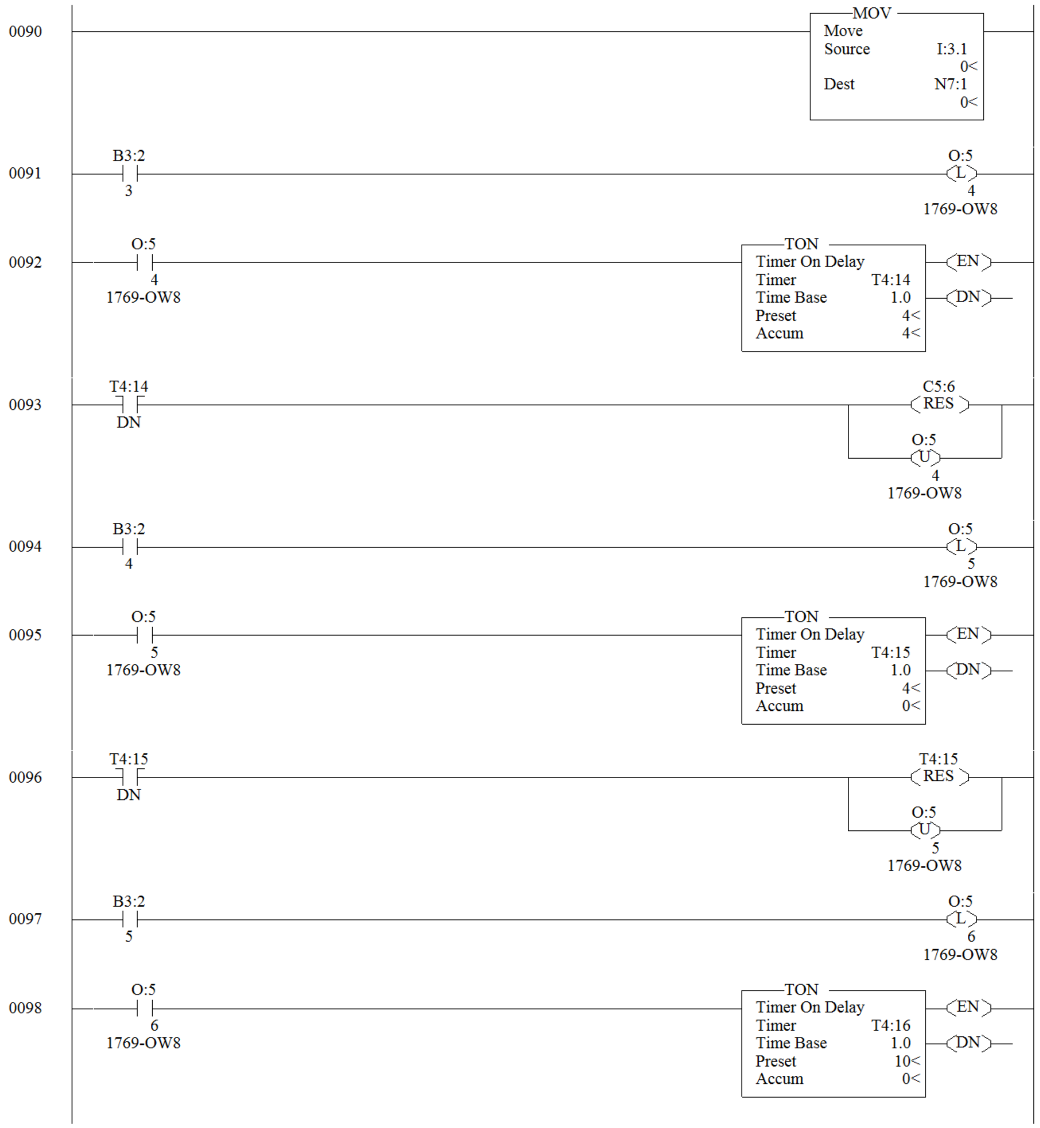

0050
T4:13
DN
O:0
U
6
Bul.1764
O:0
U
7
Bul.1764
T4:13
RES
B3:2
U
2
0051
I:0
9
Bul.1764
O:0
U
8
Bul.1764
O:0
U
6
Bul.1764
O:0
U
7
Bul.1764
B3:0
L
15
0052
B3:0
15
O:0
6
Bul.1764
TON
Timer On Delay
Timer T4:1
Time Base 1.0
Preset 1<
Accum 1<
EN
DN
0053
T4:1
DN
O:0
L
9
Bul.1764
0054
O:0
9
Bul.1764
TON
Timer On Delay
Timer T4:3
Time Base 1.0
Preset 1<
Accum 0<
EN
DN
0055
T4:3
DN
O:0
U
9
Bul.1764
T4:1
RES

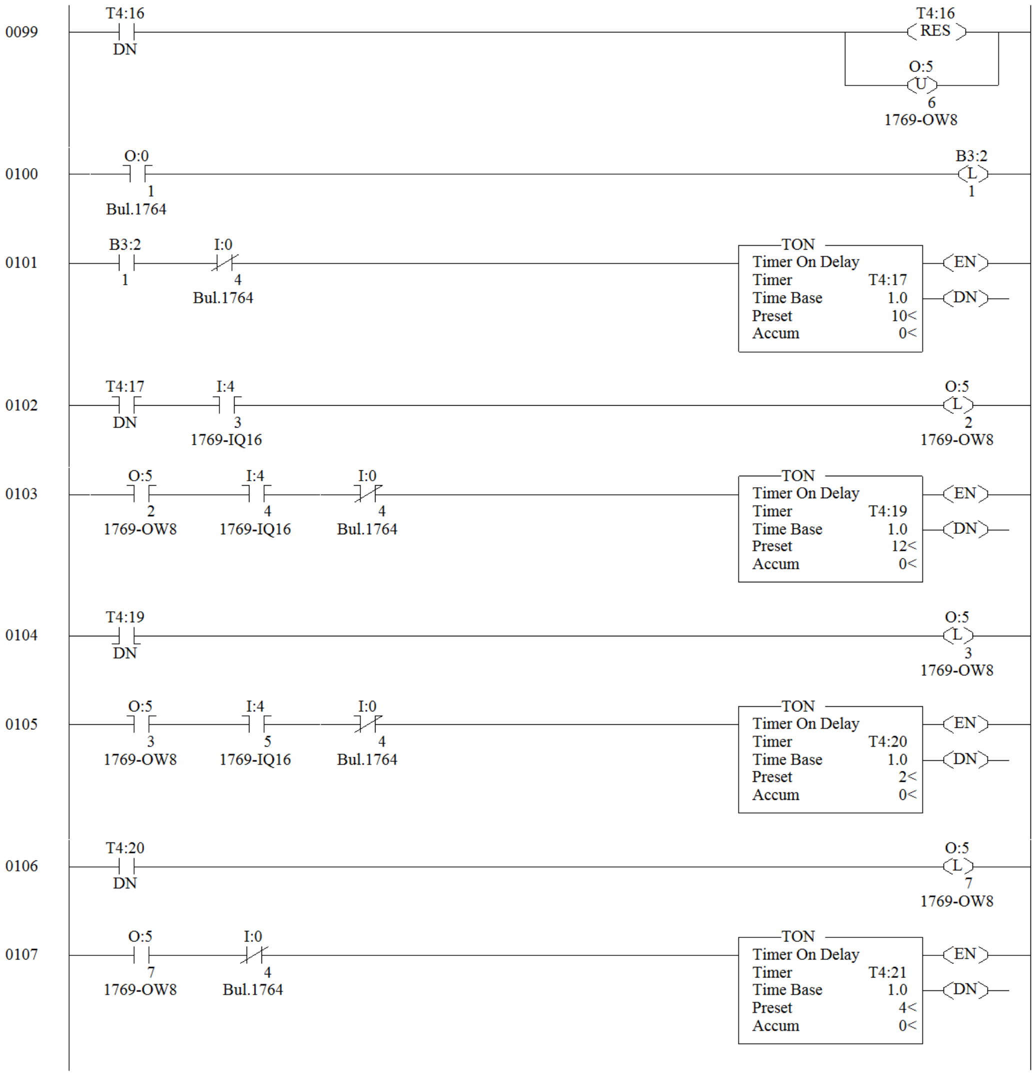

0056
B3:2
12
O:0
6
Bul.1764
B3:1
L
8
0057
B3:1
8
I:0
5
Bul.1764
O:0
L
5
Bul.1764
0058
O:0
5
Bul.1764
TON
Timer On Delay
Timer T4:4
Time Base 1.0
Preset 3<
Accum 0<
EN
DN
0059
T4:4
DN
T4:4
RES
O:0
U
5
Bul.1764
Component Air
Transfer
B3:4
L
15
0060
B3:2
11
O:0
6
Bul.1764
B3:1
L
9
0061
B3:1
9
I:0
6
Bul.1764
O:0
L
4
Bul.1764
0062
O:0
4
Bul.1764
TON
Timer On Delay
Timer T4:7
Time Base 1.0
Preset 3<
Accum 0<
EN
DN
0063
T4:7
DN
O:0
U
4
Bul.1764
T4:7
RES
Component Air
Transfer
B3:4
L
15

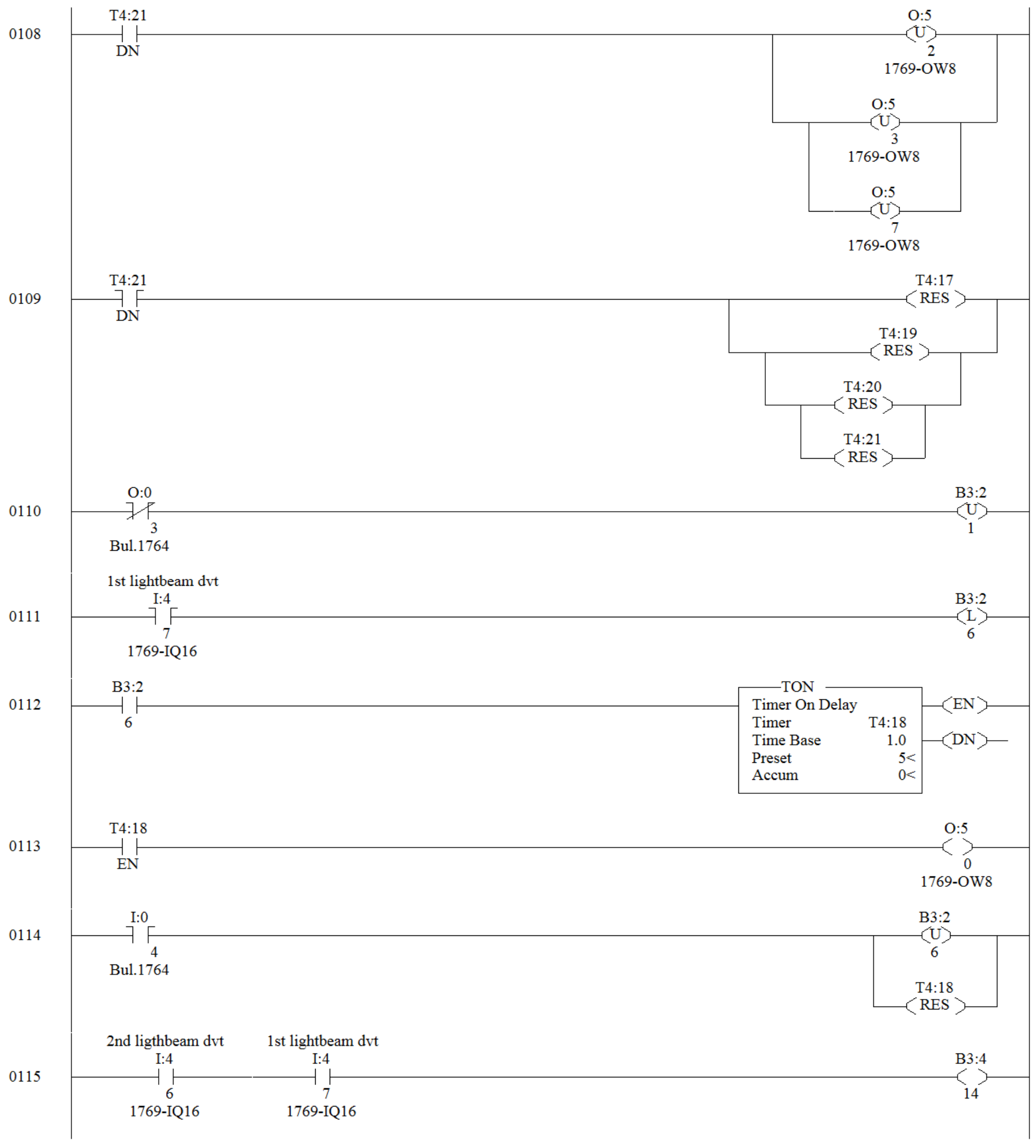

0064
Component Air Transfer
B3:4
15
TON
Timer On Delay
Timer T4:23
Time Base 1.0
Preset 2<
Accum 0<
EN
DN
0065
Component Air Transfer
T4:23
EN
O:0
0
Bul.1764
0066
T4:23
DN
B3:1
8
Stepper Camera Position 1
O:8
L
4
1769-OW8
0067
T4:23
DN
B3:1
9
Stepper Camera Position 2
O:8
L
5
1769-OW8
0068
Stepper Camera Position 1
O:8
4
1769-OW8
Stepper Camera Position 2
O:8
5
1769-OW8
TON
Timer On Delay
Timer T4:22
Time Base 1.0
Preset 1<
Accum 0<
EN
DN
0069
T4:22
DN
Stepper Camera Position 1
O:8
U
4
1769-OW8
Stepper Camera Position 2
O:8
U
5
1769-OW8
B3:1
U
8
B3:1
U
9

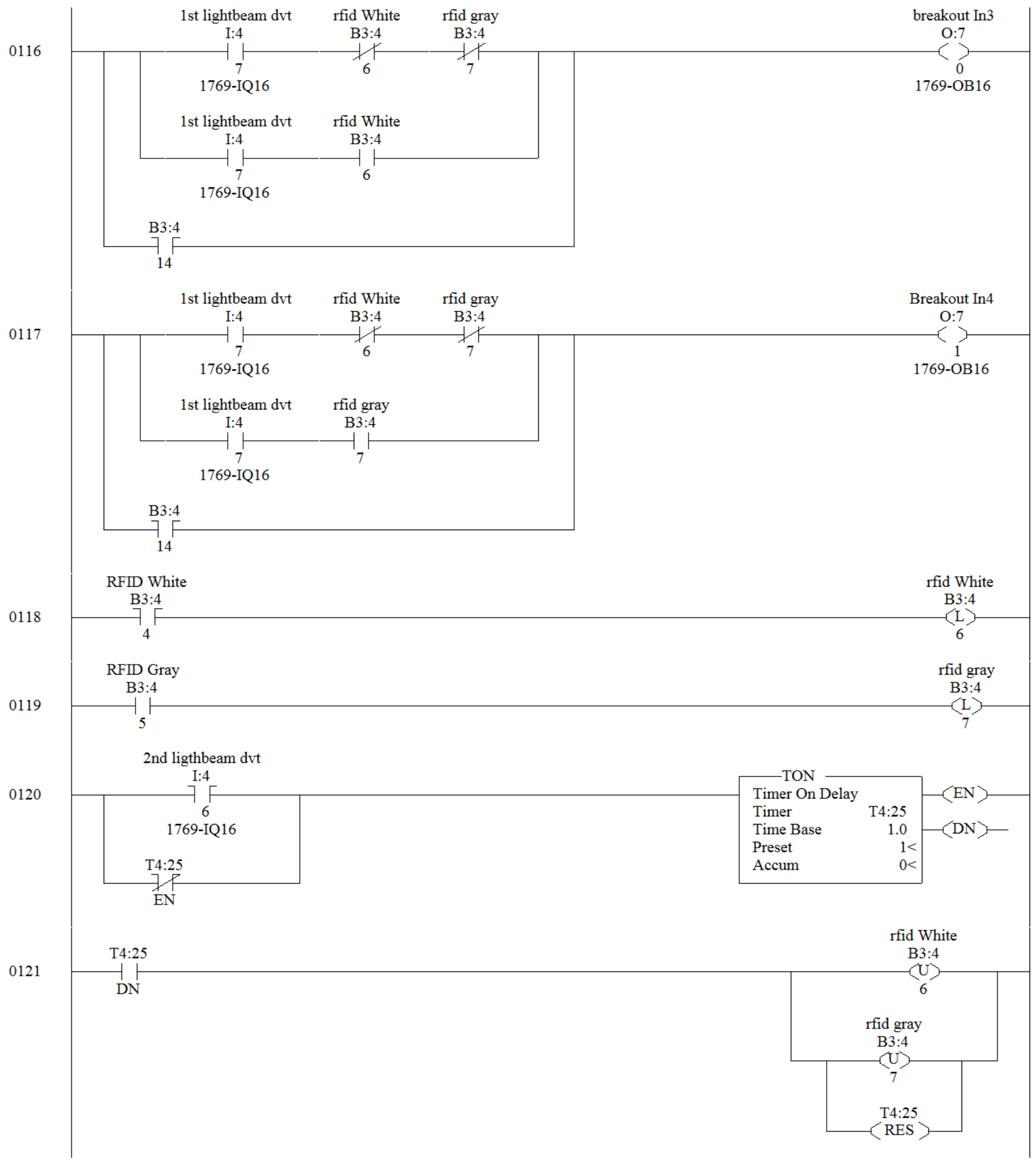

0070
T4:23
DN
T4:23
RES
Component Air
Transfer
B3:4
U
15
0071
I:4
2
1769-IQ16
O:0
6
Bul.1764
B3:2
L
0
0072
I:4
10
1769-IQ16
B3:2
U
0
0073
I:2
14
1769-IQ16
B3:1
L
10
0074
T4:12
DN
B3:1
10
TON
Timer On Delay
Timer T4:12
Time Base 1.0
Preset 15<
Accum 0<
EN
DN
0075
I:2
14
1769-IQ16
T4:12
EN
T4:12
DN
CTU
Count Up
Counter C5:5
Preset 1000<
Accum 0<
CU
DN
0076
SUB
Subtract
Source A C5:5.ACC
0<
Source B 1
1<
Dest N7:7
-1<
0077
T4:12
DN
MUL
Multiply
Source A 38
38<
Source B N7:7
-1<
Dest N7:8
2128<
0078
DIV
Divide
Source A N7:8
2128<
Source B 1000
1000<
Dest N7:9
2<

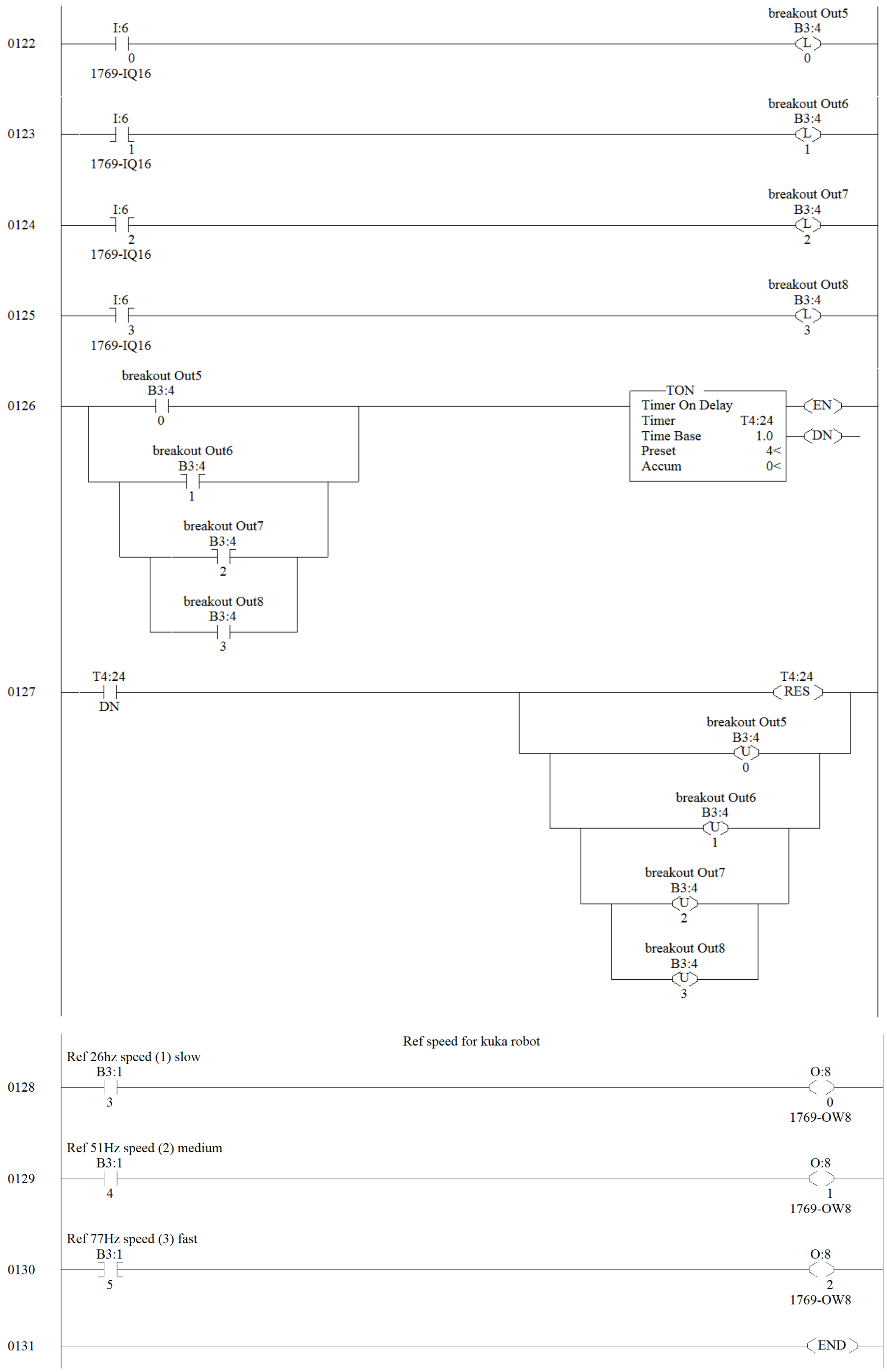

0079
T4:12
DN
C5:5
RES
0080
T4:12
DN
B3:1
U
10
0081
I:0
0
Bul.1764
B3:1
L
14
0082
I:0
3
Bul.1764
B3:0
1
B3:1
U
14
0083
I:0
1
Bul.1764
B3:1
14
B3:1
L
11
0084
I:0
2
Bul.1764
B3:1
U
11
0085
I:0
2
Bul.1764
B3:1
14
B3:1
L
12
0086
I:0
4
Bul.1764
B3:1
U
12
0087
I:0
4
Bul.1764
B3:1
13
0088
I:4
10
1769-IQ16
O:0
6
Bul.1764
B3:1
L
15
0089
I:0
9
Bul.1764
I:0
5
Bul.1764
I:0
6
Bul.1764
B3:1
U
15


0090
MOV
Move
Source I:3.1
0<
Dest N7:1
0<
0091
B3:2
3
O:5
L
4
1769-OW8
0092
O:5
4
1769-OW8
TON
Timer On Delay
Timer T4:14
Time Base 1.0
Preset 4<
Accum 4<
EN
DN
0093
T4:14
DN
C5:6
RES
O:5
U
4
1769-OW8
0094
B3:2
4
O:5
L
5
1769-OW8
0095
O:5
5
1769-OW8
TON
Timer On Delay
Timer T4:15
Time Base 1.0
Preset 4<
Accum 0<
EN
DN
0096
T4:15
DN
T4:15
RES
O:5
U
5
1769-OW8
0097
B3:2
5
O:5
L
6
1769-OW8
0098
O:5
6
1769-OW8
TON
Timer On Delay
Timer T4:16
Time Base 1.0
Preset 10<
Accum 0<
EN
DN

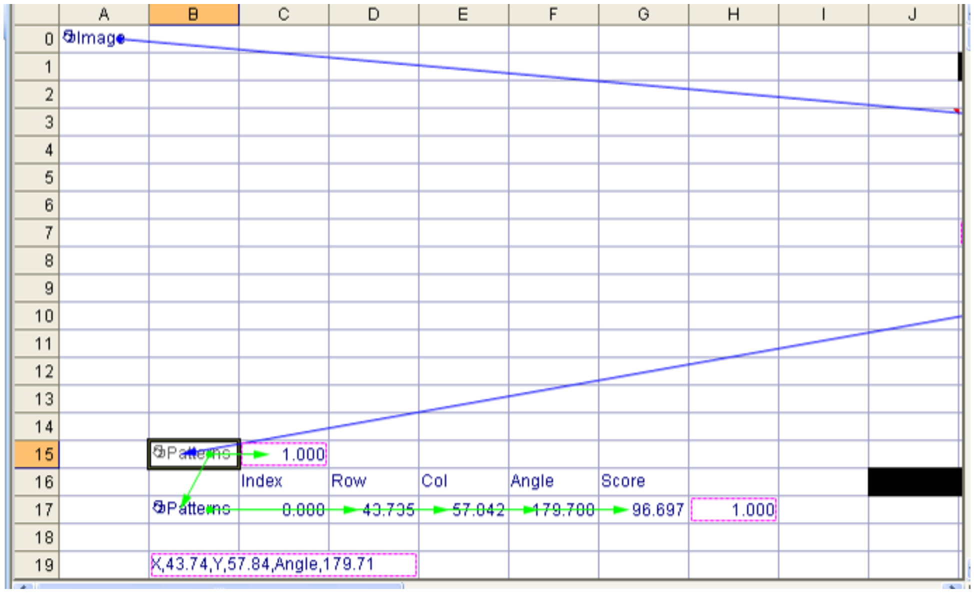

0099
T4:16
DN
T4:16
RES
O:5
U
6
1769-OW8
0100
O:0
1
Bul.1764
B3:2
L
1
0101
B3:2
1
I:0
4
Bul.1764
TON
Timer On Delay
Timer T4:17
Time Base 1.0
Preset 10<
Accum 0<
EN
DN
0102
T4:17
DN
I:4
3
1769-IQ16
O:5
L
2
1769-OW8
0103
O:5
2
1769-OW8
I:4
4
1769-IQ16
I:0
4
Bul.1764
TON
Timer On Delay
Timer T4:19
Time Base 1.0
Preset 12<
Accum 0<
EN
DN
0104
T4:19
DN
O:5
L
3
1769-OW8
0105
O:5
3
1769-OW8
I:4
5
1769-IQ16
I:0
4
Bul.1764
TON
Timer On Delay
Timer T4:20
Time Base 1.0
Preset 2<
Accum 0<
EN
DN
0106
T4:20
DN
O:5
L
7
1769-OW8
0107
O:5
7
1769-OW8
I:0
4
Bul.1764
TON
Timer On Delay
Timer T4:21
Time Base 1.0
Preset 4<
Accum 0<
EN
DN


0108
T4:21
DN
O:5
U
2
1769-OW8
O:5
U
3
1769-OW8
O:5
U
7
1769-OW8
0109
T4:21
DN
T4:17
RES
T4:19
RES
T4:20
RES
T4:21
RES
0110
O:0
3
Bul.1764
B3:2
U
1
0111
1st lightbeam dvt
I:4
7
1769-IQ16
B3:2
L
6
0112
B3:2
6
TON
Timer On Delay
Timer T4:18
Time Base 1.0
Preset 5<
Accum 0<
EN
DN
0113
T4:18
EN
O:5
0
1769-OW8
0114
I:0
4
Bul.1764
B3:2
U
6
T4:18
RES
0115
2nd ligthbeam dvt
I:4
6
1769-IQ16
1st lightbeam dvt
I:4
7
1769-IQ16
B3:4
14


0116
1st lightbeam dvt
I:4
7
1769-IQ16
rfid White
B3:4
6
rfid gray
B3:4
7
breakout In3
O:7
0
1769-OB16
1st lightbeam dvt
I:4
7
1769-IQ16
rfid White
B3:4
6
B3:4
14
0117
1st lightbeam dvt
I:4
7
1769-IQ16
rfid White
B3:4
6
rfid gray
B3:4
7
Breakout In4
O:7
1
1769-OB16
1st lightbeam dvt
I:4
7
1769-IQ16
rfid gray
B3:4
7
B3:4
14
0118
RFID White
B3:4
4
rfid White
B3:4
L
6
0119
RFID Gray
B3:4
5
rfid gray
B3:4
L
7
0120
2nd ligthbeam dvt
I:4
6
1769-IQ16
T4:25
EN
TON
Timer On Delay
Timer T4:25
Time Base 1.0
Preset 1<
Accum 0<
EN
DN
0121
T4:25
DN
rfid White
B3:4
U
6
rfid gray
B3:4
U
7
T4:25
RES

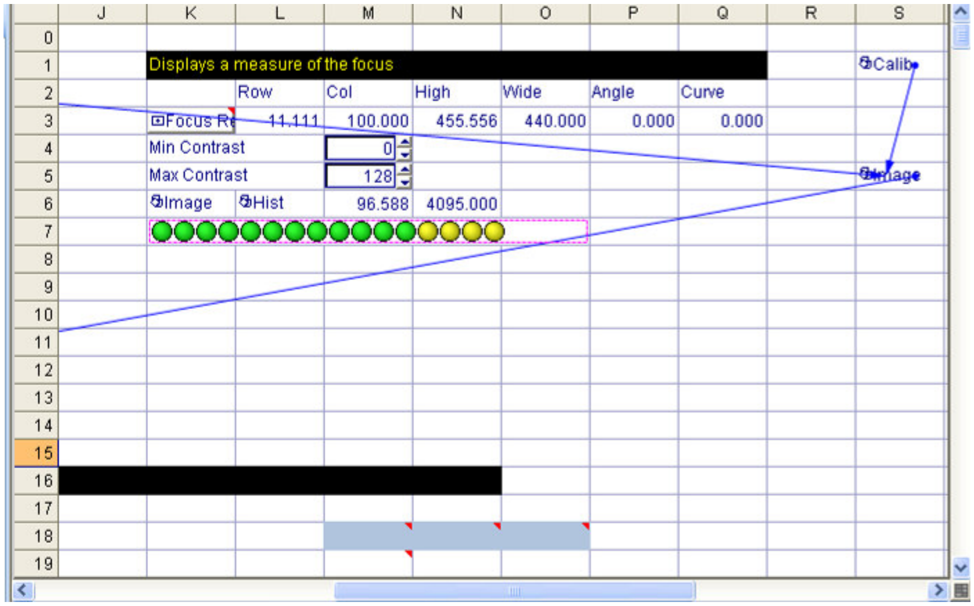

0122
I:6
0
1769-IQ16
breakout Out5
B3:4
L
0
0123
I:6
1
1769-IQ16
breakout Out6
B3:4
L
1
0124
I:6
2
1769-IQ16
breakout Out7
B3:4
L
2
0125
I:6
3
1769-IQ16
breakout Out8
B3:4
L
3
0126
breakout Out5
B3:4
0
breakout Out6
B3:4
1
breakout Out7
B3:4
2
breakout Out8
B3:4
3
TON
Timer On Delay
Timer T4:24
Time Base 1.0
Preset 4<
Accum 0<
EN
DN
0127
T4:24
DN
T4:24
RES
breakout Out5
B3:4
U
0
breakout Out6
B3:4
U
1
breakout Out7
B3:4
U
2
breakout Out8
B3:4
U
3
Ref speed for kuka robot
0128
Ref 26hz speed (1) slow
B3:1
3
O:8
0
1769-OW8
0129
Ref 51Hz speed (2) medium
B3:1
4
O:8
1
1769-OW8
0130
Ref 77Hz speed (3) fast
B3:1
5
O:8
2
1769-OW8
0131
END

8 Appendix B



	A	B	C	D	E	F	G	H	I	J
0	Image									
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										
11										
12										
13										
14										
15		Patterns	1.000							
16			Index	Row	Col	Angle	Score			
17		Patterns	0.000	43.735	57.042	179.700	96.697	1.000		
18										
19		X,43.74,Y,57.84,Angle,179.71								

Figure 8-1 Dependencies 1



	J	K	L	M	N	O	P	Q	R	S
0										
1		Displays a measure of the focus								Calib
2			Row	Col	High	Wide	Angle	Curve		
3		Focus Re	11.111	100.000	455.556	440.000	0.000	0.000		
4		Min Contrast		0						
5		Max Contrast		128						Image
6		Image	Hist	96.588	4095.000					
7		🟢🟢🟢🟢🟢🟢🟢🟢🟢🟢🟢🟢🟡🟡🟡🟡								
8										
9										
10										
11										
12										
13										
14										
15										
16										
17										
18										
19										

Figure 8-2 Dependencies 2

9 Appendix C

```
&ACCESS RVP
&REL 68
&PARAM TEMPLATE = C:\KRC\Roboter\Template\vorgabe
&PARAM EDITMASK = *
DEF m( )
;FOLD INI
  ;FOLD BASISTECH INI
GLOBAL INTERRUPT DECL 3 WHEN $STOPMESS==TRUE DO IR_STOPM ( )
INTERRUPT ON 3
    BAS (#INITMOV,0 )
  ;ENDFOLD (BASISTECH INI)
  ;FOLD USER INI

  ;ENDFOLD (USER INI)
;ENDFOLD (INI)

; Home position of the robot
;FOLD PTP HOME  Vel= 100 %
DEFAULT;%{PE}%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:HOME,
3:, 5:100, 7:DEFAULT
$BWDSTART = FALSE
PDAT_ACT=PDEFAULT
FDAT_ACT=FHOME
BAS (#PTP_PARAMS,100 )
$H_POS=XHOME
PTP  XHOME
;ENDFOLD

; Move to inspection position over sensor 1
;FOLD PTP P10  Vel= 100 % PDAT9 Tool[3]:suctioncup Base[2]:camera;%{PE}%R
5.2.25,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:P10, 3:, 5:100,
7:PDAT9
$BWDSTART = FALSE
PDAT_ACT=PPDAT9
FDAT_ACT=FP10
BAS(#PTP_PARAMS,100)
PTP XP10
;ENDFOLD

; Wait for sensor 1
WAIT FOR $IN[4]==TRUE

; Select job1 stored in camera[1]
CAMERA[1].PRODUCT=1

; Reset Capture Image Trigger Flag
CAMERA[1].CAPTURE_IMAGE=FALSE
```

```
; Reset Data_Ready Flag
CAMERA[1].DATA_READY=FALSE

WAIT SEC 0.1

; Repeat until data received from camera
REPEAT

; Trigger an inspection in camera 1
CAMERA[1].CAPTURE_IMAGE=TRUE

; Wait for inspection to finish
WHILE (CAMERA[1].CAPTURE_IMAGE==TRUE)
WAIT SEC 0.0
ENDWHILE
UNTIL (CAMERA[1].DATA_READY==TRUE)

; Reset pick position
PICK_POS=$NULLFRAME

;Load pick position with data from the camera
XPICK_POS.X=RESULT[1,1]
XPICK_POS.Y=RESULT[1,2]
XPICK_POS.A=RESULT[1,3]

; Robot move TCP to centre point above pattern
LIN XPICK_POS

; Home position of robot
;FOLD PTP HOME  Vel= 100 %
DEFAULT;%{PE}%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:HOME,
3:, 5:100, 7:DEFAULT
$BWDSTART = FALSE
PDAT_ACT=PDEFAULT
FDAT_ACT=FHOME
BAS (#PTP_PARAMS,100 )
$H_POS=XHOME
PTP  XHOME
;ENDFOLD

END
```

10 Appendix D

```
&ACCESS RVP
&REL 68
&PARAM TEMPLATE = C:\KRC\Roboter\Template\vorgabe
&PARAM EDITMASK = *
DEF m( )
;FOLD INI
  ;FOLD BASISTECH INI
GLOBAL INTERRUPT DECL 3 WHEN $STOPMESS==TRUE DO IR_STOPM ( )
INTERRUPT ON 3
    BAS (#INITMOV,0 )
  ;ENDFOLD (BASISTECH INI)
  ;FOLD USER INI

  ;ENDFOLD (USER INI)
;ENDFOLD (INI)

; Home position of the robot
;FOLD PTP HOME  Vel= 100 %
DEFAULT;%{PE}%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:HOME,
3:, 5:100, 7:DEFAULT
$BWDSTART = FALSE
PDAT_ACT=PDEFAULT
FDAT_ACT=FHOME
BAS (#PTP_PARAMS,100 )
$H_POS=XHOME
PTP  XHOME
;ENDFOLD

; Move to inspection position over sensor 1
;FOLD PTP P10  Vel= 100 % PDAT9 Tool[3]:suctioncup Base[2]:camera;%{PE}%R
5.2.25,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:P10, 3:, 5:100,
7:PDAT9
$BWDSTART = FALSE
PDAT_ACT=PPDAT9
FDAT_ACT=FP10
BAS(#PTP_PARAMS,100)
PTP XP10
;ENDFOLD

; Wait for sensor 1
WAIT FOR $IN[4]==TRUE

; Select job1 stored in camera[1]
CAMERA[1].PRODUCT=1

; Reset Capture Image Trigger Flag
CAMERA[1].CAPTURE_IMAGE=FALSE
```

```
; Reset Data_Ready Flag
CAMERA[1].DATA_READY=FALSE

WAIT SEC 0.1

;Repeat until data received from camera
REPEAT

; Trigger an inspection in camera 1
CAMERA[1].CAPTURE_IMAGE=TRUE

; Wait for inspection to finish
WHILE (CAMERA[1].CAPTURE_IMAGE==TRUE)
WAIT SEC 0.0
ENDWHILE
UNTIL (CAMERA[1].DATA_READY==TRUE)

; Reset pick position
PICK_POS=$NULLFRAME

; Load pick position with data from the camera
XPICK_POS.X=RESULT[1,1]
XPICK_POS.Y=RESULT[1,2]
XPICK_POS.A=RESULT[1,3]

; Robot move TCP to centre point above pattern
LIN XPICK_POS

; Suction cup positive suction ON
;FOLD OUT 2 'Extension cup'  State= TRUE CONT;%{PE}%R
5.2.25,%MKUKATPBASIS,%COUT,%VOUTX,%P 2:2, 3:Extention cup, 5:TRUE,
6:CONTINUE
CONTINUE
$OUT[2]=TRUE
;ENDFOLD

; Wait 2 seconds
;FOLD WAIT Time= 2 sec;%{PE}%R
5.2.25,%MKUKATPBASIS,%CWAIT,%VWAIT,%P 2:2
WAIT SEC 2
;ENDFOLD

; Extension arm OUT pick object
;FOLD OUT 3 'Extension arm'  State= TRUE CONT;%{PE}%R
5.2.25,%MKUKATPBASIS,%COUT,%VOUTX,%P 2:3, 3:Extention arm, 5:TRUE,
6:CONTINUE
CONTINUE
$OUT[3]=TRUE
;ENDFOLD
```

```
; Move to drop location at conveyor 2
;FOLD PTP P12  Vel= 100 % PDAT14 Tool[3]:suctioncup
Base[2]:camera;%{PE}%R 5.2.25,%MKUKATPBASIS,%CMOVE,%VPTP,%P
1:PTP, 2:P12, 3:, 5:100, 7:PDAT14
$BWDSTART = FALSE
PDAT_ACT=PPDAT14
FDAT_ACT=FP12
BAS(#PTP_PARAMS,100)
PTP XP12
;ENDFOLD

; Suction cup positive suction OFF release object
;FOLD OUT 2 'Extension cup'  State= FALSE CONT;%{PE}%R
5.2.25,%MKUKATPBASIS,%COUT,%VOUTX,%P 2:2, 3:Extention cup, 5:FALSE,
6:CONTINUE
CONTINUE
$OUT[2]=FALSE
;ENDFOLD

; Wait 2 seconds
;FOLD WAIT Time= 2 sec;%{PE}%R
5.2.25,%MKUKATPBASIS,%CWAIT,%VWAIT,%P 2:2
WAIT SEC 2
;ENDFOLD

; Extension arm IN
;FOLD OUT 3 'Extension arm'  State= FALSE CONT;%{PE}%R
5.2.25,%MKUKATPBASIS,%COUT,%VOUTX,%P 2:3, 3:Extention arm, 5:FALSE,
6:CONTINUE
CONTINUE
$OUT[3]=FALSE
;ENDFOLD

; Wait 2 seconds
;FOLD WAIT Time= 2 sec;%{PE}%R
5.2.25,%MKUKATPBASIS,%CWAIT,%VWAIT,%P 2:2
WAIT SEC 2
;ENDFOLD

; Home position of robot
;FOLD PTP HOME  Vel= 100 %
DEFAULT;%{PE}%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:HOME,
3:, 5:100, 7:DEFAULT
$BWDSTART = FALSE
PDAT_ACT=PDEFAULT
FDAT_ACT=FHOME
BAS (#PTP_PARAMS,100 )
$H_POS=XHOME
PTP  XHOME
;ENDFOLD END
```

11 Appendix E

```
&ACCESS RVO
&REL 102
&COMMENT Version105
&PARAM TEMPLATE = C:\KRC\Roboter\Template\vorgabe
&PARAM EDITMASK = *
DEF vision_find( )
;FOLD INI
  ;FOLD BASISTECH INI
  GLOBAL INTERRUPT DECL 3 WHEN $STOPMESS==TRUE DO IR_STOPM ()
    INTERRUPT ON 3
    BAS (#INITMOV,0 )
  ;ENDFOLD (BASISTECH INI)
  ;FOLD USER INI
    ; Make your modifications here

  ;ENDFOLD (USER INI)
;ENDFOLD (INI)


LOOP

; Robot home position
;FOLD PTP HOME  Vel= 100 % DEFAULT;%{PE}%R
5.2.26,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:HOME, 3:, 5:100,
7:DEFAULT
$BWDSTART = FALSE
PDAT_ACT=PDEFAULT
FDAT_ACT=FHOME
BAS(#PTP_PARAMS,100)
$H_POS=XHOME
PTP XHOME
;ENDFOLD

; Reset pick position rest PICK_POS
PICKDATA[1].PICKPOS.X=0
PICKDATA[1].PICKPOS.Y=0
PICKDATA[1].PICKPOS.A=0


; Move to inspection position over sensor 1
;FOLD PTP P3  Vel= 100 % PDAT4 Tool[3]:suctioncup Base[2]:camera;%{PE}%R
5.2.25,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:P3, 3:, 5:100,
7:PDAT4
$BWDSTART = FALSE
PDAT_ACT=PPDAT4
FDAT_ACT=FP3
BAS(#PTP_PARAMS,100)
PTP XP3
```

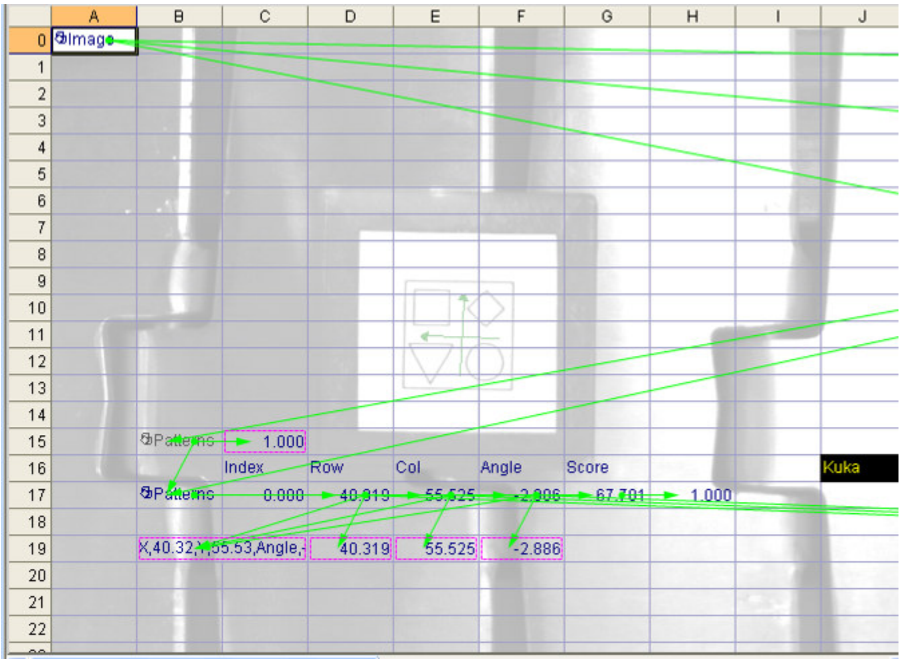
```
;ENDFOLD

:Wait for sensor 1
WAIT FOR $IN[PICKDATA[1].CAM_SENSOR]

; Subprogram for getting KUKA Vision pick offsets
Kuka_Vision ( )

; Load pick position with data from camera
PICKDATA[1].PICKPOS.X=RESULT[1,1]+20   ;Offset from Kuka vision saved to
Pickpos
PICKDATA[1].PICKPOS.Y=RESULT[1,2]   ;Offset from Kuka vision saved to
Pickpos
PICKDATA[1].PICKPOS.A=RESULT[1,3]   ;Offset from Kuka vision saved to
Pickpos

; Sub-program Save offset positions
SaveOffsetPos ( )

; Move TCP to fixed Y offset
PICKDATA[1].PICKPOS.Y=300.710876          ;Fixed y offset

; Fold display offsets in message window
DisplayX=PICKDATA[1].PICKPOS.X
DisplayY=PICKDATA[1].PICKPOS.Y
DisplayA=PICKDATA[1].PICKPOS.A
Msg_Display(DisplayX,DisplayY,DisplayA)
;endfold

; Suction cup ON
;FOLD OUT 2 'Extension cup'  State= TRUE CONT;%{PE}%R
5.2.25,%MKUKATPBASIS,%COUT,%VOUTX,%P 2:2, 3:Extention cup, 5:TRUE,
6:CONTINUE
CONTINUE
$OUT[2]=TRUE
;ENDFOLD

; Robot will move to new X and A position to the set Y value
LIN PICKDATA[1].PICKPOS

; Wait for sensor 2
WAIT FOR $IN[PICKDATA[1].PICK_SENSOR]


; Extension arm ON
;FOLD OUT 3 'Extension arm'  State= TRUE CONT;%{PE}%R
5.2.25,%MKUKATPBASIS,%COUT,%VOUTX,%P 2:3, 3:Extention arm, 5:TRUE,
6:CONTINUE
CONTINUE
$OUT[3]=TRUE
```

```
;ENDFOLD
WAIT SEC PICKDATA[1].RetractDelayTime
;$OUT[PICKDATA[1].ExtendARM]=FALSE

; Move 600 up in Z from conveyor 1
LIN_REL {Z 600}

; Move to drop position at conveyor 2
;FOLD PTP P12  Vel= 100 % PDAT14 Tool[3]:suctioncup
Base[2]:camera;%{PE}%R 5.2.25,%MKUKATPBASIS,%CMOVE,%VPTP,%P
1:PTP, 2:P12, 3:, 5:100, 7:PDAT14
$BWDSTART = FALSE
PDAT_ACT=PPDAT14
FDAT_ACT=FP12
BAS(#PTP_PARAMS,100)
PTP XP12
;ENDFOLD

; Wait 1 second
;FOLD WAIT Time= 1 sec;%{PE}%R
5.2.25,%MKUKATPBASIS,%CWAIT,%VWAIT,%P 2:1
WAIT SEC 1
;ENDFOLD

; Suction cup OFF
;FOLD OUT 2 'Extension cup'  State= FALSE CONT;%{PE}%R
5.2.25,%MKUKATPBASIS,%COUT,%VOUTX,%P 2:2, 3:Extention cup, 5:FALSE,
6:CONTINUE
CONTINUE
$OUT[2]=FALSE
;ENDFOLD

; Wait 1 second
;FOLD WAIT Time= 1 sec;%{PE}%R
5.2.25,%MKUKATPBASIS,%CWAIT,%VWAIT,%P 2:1
WAIT SEC 1
;ENDFOLD

; Extension arm OFF
;FOLD OUT 3 'Extension arm'  State= FALSE CONT;%{PE}%R
5.2.25,%MKUKATPBASIS,%COUT,%VOUTX,%P 2:3, 3:Extention arm, 5:FALSE,
6:CONTINUE
CONTINUE
$OUT[3]=FALSE
;ENDFOLD

; Wait 2 seconds
;FOLD WAIT Time= 2 sec;%{PE}%R
5.2.25,%MKUKATPBASIS,%CWAIT,%VWAIT,%P 2:2
WAIT SEC 2
```

```
;ENDFOLD

; Move 20 up in Z from drop position at conveyor 2
LIN_REL {Z 20}

; Robot home position
;FOLD PTP HOME  Vel= 100 % DEFAULT;%{PE}%R
5.2.26,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:HOME, 3:, 5:100,
7:DEFAULT
$BWDSTART = FALSE
PDAT_ACT=PDEFAULT
FDAT_ACT=FHOME
BAS(#PTP_PARAMS,100)
$H_POS=XHOME
PTP XHOME
;ENDFOLD

ENDLOOP

END


DEF KUKA_VISION ( )

; Select job1 stored in camera[1]
CAMERA[1].PRODUCT=1

; Reset Capture Image Trigger Flag
CAMERA[1].CAPTURE_IMAGE=FALSE

; Reset Data_Ready Flag
CAMERA[1].DATA_READY=FALSE

WAIT SEC 0.1

; Repeat until data received from camera
REPEAT

; Trigger an inspection in camera 1
CAMERA[1].CAPTURE_IMAGE=TRUE

; Wait for inspection to finish
WHILE (CAMERA[1].CAPTURE_IMAGE==TRUE)
WAIT SEC 0.0
ENDWHILE
UNTIL (CAMERA[1].DATA_READY==TRUE)
END

; Display offsets of object found in window
DEF MSG_Display (DisplayX : IN,DisplayY : IN,DisplayA : IN)
```

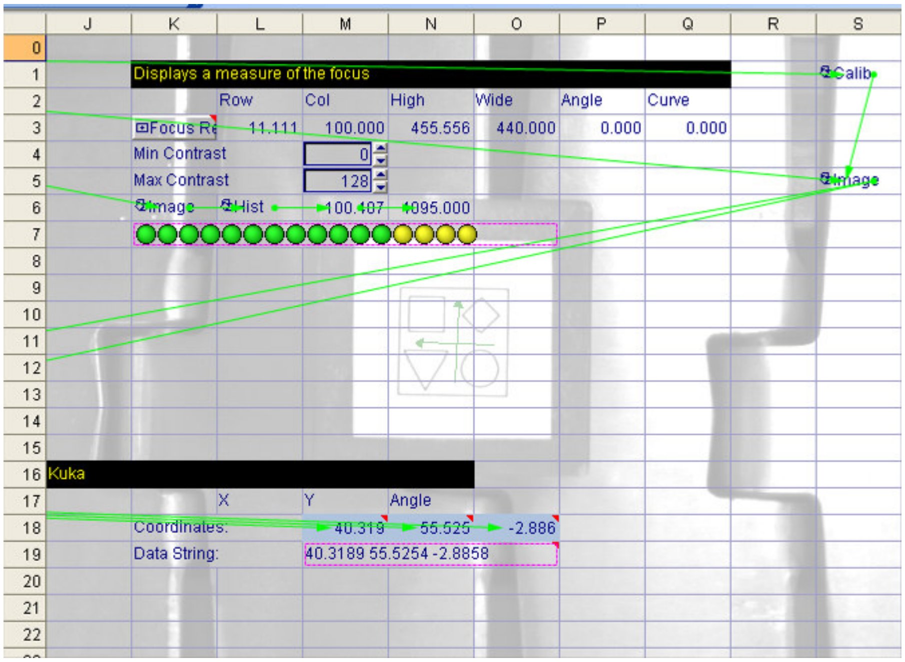
```
decl Real DisplayX
decl Real DisplayY
decl Real DisplayA
;int pointer
DECL STATE_T STATE
        $MSG_T=EMPTY_MSG
        POINTER = 0
VISION_MSG={MSG_T: VALID TRUE,RELEASE TRUE,TYP
#NOTIFY,MODUL[] "VISION",KEY[] " ",PARAM_TYP #VALUE,PARAM[] "
",DLG_FORMAT[] " ",ANSWER 0}
swrite (VISION_MSG.KEY[],STATE,POINTER,"X = %f,Y = %f,A = %f
",DisplayX,DisplayY,DisplayA)
$MSG_T=VISION_MSG
END

DEF SAVEOFFSETPOS ( )
INT I
; Store last pos data on top of the list
FOR I=50 TO 2 STEP -1
SAVEDOFFSET[I]=SAVEDOFFSET[I-1]
ENDFOR
; Store last pos data
SAVEDOFFSET[1]=PICKDATA[1].PICKPOS
END
```

12 Appendix F



	A	B	C	D	E	F	G	H	I	J
0	Image									
15		Patterns	1.000							
16			Index	Row	Col	Angle	Score			Kuka
17		Patterns	0.000	40.319	55.525	-2.886	67.701	1.000		
19		X,40.32,Y,55.53,Angle,-		40.319	55.525	-2.886				

Figure 12-1 Vision program V1.1with dependencies 1



	J	K	L	M	N	O	P	Q	R	S
1		Displays a measure of the focus								Calib
2			Row	Col	High	Wide	Angle	Curve		
3		Focus Re	11.111	100.000	455.556	440.000	0.000	0.000		
4		Min Contrast		0						
5		Max Contrast		128						Image
6		Image	Hist	100.407	4095.000					
7		●●●●●●●●●●●●●●●●								
16	Kuka									
17			X	Y	Angle					
18		Coordinates:		40.319	55.525	-2.886				
19		Data String:		40.3189 55.5254 -2.8858						

Figure 12-2 Vision programV1.1 with dependencies 2

13 Appendix G

```
&ACCESS RVO
&REL 102
&COMMENT Version105
&PARAM TEMPLATE = C:\KRC\Roboter\Template\vorgabe
&PARAM EDITMASK = *
DEF vision_find( )
;FOLD INI
  ;FOLD BASISTECH INI
  GLOBAL INTERRUPT DECL 3 WHEN $STOPMESS==TRUE DO IR_STOPM ()
    INTERRUPT ON 3
    BAS (#INITMOV,0 )
  ;ENDFOLD (BASISTECH INI)
  ;FOLD USER INI

  ;ENDFOLD (USER INI)
;ENDFOLD (INI)

LOOP

; Robot home position
;FOLD PTP HOME  Vel= 100 % DEFAULT;%{PE}%R
5.2.26,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:HOME, 3:, 5:100,
7:DEFAULT
$BWDSTART = FALSE
PDAT_ACT=PDEFAULT
FDAT_ACT=FHOME
BAS(#PTP_PARAMS,100)
$H_POS=XHOME
PTP XHOME
;ENDFOLD

; After camera data = no data, reset and start here
startpos:

; Reset pick position rest PICK_POS
PICKDATA[1].PICKPOS.X=0
PICKDATA[1].PICKPOS.Y=0
PICKDATA[1].PICKPOS.A=0

; Move to inspection position over sensor 1
;FOLD PTP P3  Vel= 100 % PDAT4 Tool[3]:suctioncup Base[2]:camera;%{PE}%R
5.2.25,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:P3, 3:, 5:100,
7:PDAT4
$BWDSTART = FALSE
PDAT_ACT=PPDAT4
FDAT_ACT=FP3
BAS(#PTP_PARAMS,100)
```

```
PTP XP3
;ENDFOLD

:Wait for sensor 1
WAIT FOR $IN[PICKDATA[1].CAM_SENSOR]

; Subprogram for getting KUKA Vision pick offsets
Kuka_Vision ( )

; If camera data = no data go to startpos
 IF CAMERA[1].DATA_READY==#NO_DATA THEN
GOTO startpos
ENDIF

; Load pick position with data from camera
PICKDATA[1].PICKPOS.X=RESULT[1,1]+20   ;Offset from Kuka vision saved to
Pickpos
PICKDATA[1].PICKPOS.Y=RESULT[1,2]   ;Offset from Kuka vision saved to
Pickpos
PICKDATA[1].PICKPOS.A=RESULT[1,3]   ;Offset from Kuka vision saved to
Pickpos

; Subprogram Save offset positions
SaveOffsetPos ( )

; Move TCP to fixed Y offset
PICKDATA[1].PICKPOS.Y=350.710876          ;Fixed Yoffset


; Fold Display Offsets in message window
DisplayX=PICKDATA[1].PICKPOS.X
DisplayY=PICKDATA[1].PICKPOS.Y
DisplayA=PICKDATA[1].PICKPOS.A
Msg_Display(DisplayX,DisplayY,DisplayA)
;endfold

; Suction cup ON
;FOLD OUT 2 'Extension cup'  State= TRUE CONT;%{PE}%R
5.2.25,%MKUKATPBASIS,%COUT,%VOUTX,%P 2:2, 3:Extention cup, 5:TRUE,
6:CONTINUE
CONTINUE
$OUT[2]=TRUE
;ENDFOLD

; Robot will move to new X and A position to the set Y value
LIN PICKDATA[1].PICKPOS

; Wait for sensor 2
WAIT FOR $IN[PICKDATA[1].PICK_SENSOR]
```

```
; Extension arm ON
;FOLD OUT 3 'Extension arm'  State= TRUE CONT;%{PE}%R
5.2.25,%MKUKATPBASIS,%COUT,%VOUTX,%P 2:3, 3:Extention arm, 5:TRUE,
6:CONTINUE
CONTINUE
$OUT[3]=TRUE
;ENDFOLD
WAIT SEC PICKDATA[1].RetractDelayTime
;$OUT[PICKDATA[1].ExtendARM]=FALSE

; Move 450 up in Z from conveyor 1
LIN_REL {Z 450}

; Move to drop position at conveyor 2
;FOLD PTP P12  Vel= 100 % PDAT14 Tool[3]:suctioncup
Base[2]:camera;%{PE}%R 5.2.25,%MKUKATPBASIS,%CMOVE,%VPTP,%P
1:PTP, 2:P12, 3:, 5:100, 7:PDAT14
$BWDSTART = FALSE
PDAT_ACT=PPDAT14
FDAT_ACT=FP12
BAS(#PTP_PARAMS,100)
PTP XP12
;ENDFOLD

; Wait 0.5 second
;FOLD WAIT Time= 0.5 sec;%{PE}%R
5.2.25,%MKUKATPBASIS,%CWAIT,%VWAIT,%P 2:1
WAIT SEC 0.5
;ENDFOLD

; Suction cup OFF
;FOLD OUT 2 'Extension cup'  State= FALSE CONT;%{PE}%R
5.2.25,%MKUKATPBASIS,%COUT,%VOUTX,%P 2:2, 3:Extention cup, 5:FALSE,
6:CONTINUE
CONTINUE
$OUT[2]=FALSE
;ENDFOLD

; Wait 0.5 second
;FOLD WAIT Time= 0.5 sec;%{PE}%R
5.2.25,%MKUKATPBASIS,%CWAIT,%VWAIT,%P 2:1
WAIT SEC 0.5
;ENDFOLD

; Extension arm OFF
;FOLD OUT 3 'Extension arm'  State= FALSE CONT;%{PE}%R
5.2.25,%MKUKATPBASIS,%COUT,%VOUTX,%P 2:3, 3:Extention arm, 5:FALSE,
6:CONTINUE
CONTINUE
$OUT[3]=FALSE
```

```
;ENDFOLD

; Wait 0.5 seconds
;FOLD WAIT Time= 0.5 sec;%{PE}%R
5.2.25,%MKUKATPBASIS,%CWAIT,%VWAIT,%P 2:2
WAIT SEC 0.5
;ENDFOLD

; Move 20 up in Z from drop position at conveyor 2
LIN_REL {Z 20}

; Robot home position
;FOLD PTP HOME  Vel= 100 % DEFAULT;%{PE}%R
5.2.26,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:HOME, 3:, 5:100,
7:DEFAULT
$BWDSTART = FALSE
PDAT_ACT=PDEFAULT
FDAT_ACT=FHOME
BAS(#PTP_PARAMS,100)
$H_POS=XHOME
PTP XHOME
;ENDFOLD

ENDLOOP

END

DEF KUKA_VISION ( )

; Select job1 stored in camera[1]
CAMERA[1].PRODUCT=1

; Reset Capture Image Trigger Flag
CAMERA[1].CAPTURE_IMAGE=FALSE

; Reset Data_Ready Flag
CAMERA[1].DATA_READY=FALSE

WAIT SEC 0.1

; Repeat until data received from camera
REPEAT

; Trigger an inspection in camera 1
CAMERA[1].CAPTURE_IMAGE=TRUE

; Wait for inspection to finish
WHILE (CAMERA[1].CAPTURE_IMAGE==TRUE)
WAIT SEC 0.0
ENDWHILE
```

```
; Has data been written to robot?
CONTINUE
IF CAMERA[1].DATA_READY==#NO_DATA THEN
CAMERA[1].DATA_READY=TRUE
ENDIF

UNTIL (CAMERA[1].DATA_READY==TRUE)
END

; Display offsets of object found in window
DEF MSG_Display (DisplayX : IN,DisplayY : IN,DisplayA : IN)
decl Real DisplayX
decl Real DisplayY
decl Real DisplayA
;int pointer
DECL STATE_T STATE
        $MSG_T=EMPTY_MSG
        POINTER = 0
VISION_MSG={MSG_T: VALID TRUE,RELEASE TRUE,TYP
#NOTIFY,MODUL[] "VISION",KEY[] " ",PARAM_TYP #VALUE,PARAM[] "
",DLG_FORMAT[] " ",ANSWER 0}
swrite (VISION_MSG.KEY[],STATE,POINTER,"X = %f,Y = %f,A = %f
",DisplayX,DisplayY,DisplayA)
$MSG_T=VISION_MSG
END

DEF SAVEOFFSETPOS ( )
INT I
; Store last pos data on top of the list
FOR I=50 TO 2 STEP -1
SAVEDOFFSET[I]=SAVEDOFFSET[I-1]
ENDFOR
; Store last pos data
SAVEDOFFSET[1]=PICKDATA[1].PICKPOS
END
```

14 Appendix H

```
&ACCESS RVP
&REL 414
&COMMENT Version106
&PARAM TEMPLATE = C:\KRC\Roboter\Template\vorgabe
&PARAM EDITMASK = *
DEF vision_find( )
;FOLD INI
  ;FOLD BASISTECH INI
    GLOBAL INTERRUPT DECL 3 WHEN $STOPMESS==TRUE DO
     IR_STOPM ()
    INTERRUPT ON 3
    BAS (#INITMOV,0 )
  ;ENDFOLD (BASISTECH INI)
  ;FOLD USER INI

  ;ENDFOLD (USER INI)
;ENDFOLD (INI)
;Fold IO Breakdown
;OUT1 CLOSE GRIPPER
;OUT2 SUCTION CUP
;OUT3  GRP EXTENSION

;IN1 CONV SLOW
;IN2 CONV MEDIUM
;IN3 CONV FAST
;IN4 CAM SENSOR
;IN5 PICK SENSOR
;endfold

; Robot home position
;FOLD PTP HOME  Vel= 100 % DEFAULT;%{PE}%R
5.2.25,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:HOME, 3:, 5:100,
7:DEFAULT
$BWDSTART = FALSE
PDAT_ACT=PDEFAULT
FDAT_ACT=FHOME
BAS(#PTP_PARAMS,100)
$H_POS=XHOME
PTP XHOME
;ENDFOLD

LOOP

; When auto selected run robot at 100
;wait for $Auto
$ov_pro=100
startpos:
```

```
; Reset pick position rest PICK_POS
PICKDATA[1].PICKPOS.X=0
PICKDATA[1].PICKPOS.Y=0
;PICKDATA[1].PICKPOS.A=0


;Move to inspection position
;FOLD PTP P3  Vel= 100 % PDAT4 Tool[3]:suctioncup Base[2]:camera;%{PE}%R
5.2.25,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:P3, 3:, 5:100,
7:PDAT4
$BWDSTART = FALSE
PDAT_ACT=PPDAT4
FDAT_ACT=FP3
BAS(#PTP_PARAMS,100)
PTP XP3
;ENDFOLD

; Wait for selected speed from SCADA for conveyor
WAIT FOR $IN[PICKDATA[1].CONVEYOR_BIT_26HZ] EXOR
$IN[PICKDATA[1].CONVEYOR_BIT_51HZ] EXOR
$IN[PICKDATA[1].CONVEYOR_BIT_77HZ]

:Wait for sensor 1
WAIT FOR $IN[PICKDATA[1].CAM_SENSOR]

; Subprogram for getting KUKA Vision pick offsets
Kuka_Vision ( )

; If camera data = no data go to startpos
IF CAMERA[1].STATE==#NO_DATA THEN
goto startpos
endif

; If greater in + X use offset
IF RESULT[1,1] > 60 THEN
PICKDATA[1].PICKPOS.X=RESULT[1,1] + xoffset   ;+15Offset from Kuka vision
saved to Pickpos
ENDIF

; If smaller in X use offset
IF RESULT[1,1] < 60 THEN
PICKDATA[1].PICKPOS.X=RESULT[1,1] + xoffsetmin   ;+15Offset from Kuka
vision saved to Pickpos
ENDIF

; Load pick position with data from camera
PICKDATA[1].PICKPOS.Y=RESULT[1,2] + yoffset  ;Offset from Kuka vision
saved to Pickpos
```

```
;PICKDATA[1].PICKPOS.A=RESULT[1,3] + zoffset  ;Offset from Kuka vision
saved to Pickpos


; Subprogram Save offset positions
SaveOffsetPos ( )


;Fold Display Offsets in message window
DisplayX=PICKDATA[1].PICKPOS.X
DisplayY=PICKDATA[1].PICKPOS.Y
DisplayA=PICKDATA[1].PICKPOS.A
Msg_Display(DisplayX,DisplayY,DisplayA)
;endfold

: Velocity subprogram
SETVEL ( )
$OUT[PICKDATA[1].SUCTION_ON]=TRUE

; Robot moves to pick position
LIN PICKDATA[1].PICKPOS
$ADVANCE = 0
$TIMER[1]=0
$TIMER_STOP[1]=FALSE

; Wait for pick sensor if not active after 5 sec go to start position
WAIT FOR $IN[PICKDATA[1].PICK_SENSOR] OR ($TIMER[1] > 5000)
IF $TIMER[1] > 5000 THEN
GOTO STARTPOS

; Loop message if missed part pick
$LOOP_MSG [ ]= "MISSED PART PICK "
ENDIF
$OUT[PICKDATA[1].ExtendARM]=TRUE
WAIT SEC delaytime
$VEL.CP = 2.0

; Move up in Z 400
LIN_REL {Z 400}

; Move to drop pos
;FOLD PTP P12  Vel= 100 % PDAT14 Tool[3]:suctioncup
Base[2]:camera;%{PE}%R 5.2.25,%MKUKATPBASIS,%CMOVE,%VPTP,%P
1:PTP, 2:P12, 3:, 5:100, 7:PDAT14
$BWDSTART = FALSE
PDAT_ACT=PPDAT14
FDAT_ACT=FP12
BAS(#PTP_PARAMS,100)
PTP XP12
;ENDFOLD
```

```
; Move to level part drop
;FOLD PTP P5  Vel= 100 % PDAT17 Tool[3]:suctioncup
Base[2]:camera_base1;%{PE}%R
5.2.25,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:P5, 3:, 5:100,
7:PDAT17
$BWDSTART = FALSE
PDAT_ACT=PPDAT17
FDAT_ACT=FP5
BAS(#PTP_PARAMS,100)
PTP XP5
;ENDFOLD

; Suction cup off
$OUT[PICKDATA[1].SUCTION_ON]=FALSE
wait sec 1
; Extension arm in
$OUT[PICKDATA[1].ExtendARM]=FALSE


ENDLOOP

; Robot home position
;FOLD PTP HOME  Vel= 100 % DEFAULT;%{PE}%R
5.2.25,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:HOME, 3:, 5:100,
7:DEFAULT
$BWDSTART = FALSE
PDAT_ACT=PDEFAULT
FDAT_ACT=FHOME
BAS(#PTP_PARAMS,100)
$H_POS=XHOME
PTP XHOME
;ENDFOLD
END


DEF KUKA_VISION ( )

; Select job1 stored in camera[1]
CAMERA[1].PRODUCT=1

; Reset Capture Image Trigger Flag
CAMERA[1].CAPTURE_IMAGE=FALSE

; Reset Data_Ready Flag
CAMERA[1].DATA_READY=FALSE

WAIT SEC 0.1
```

```
; Repeat until data received from camera
REPEAT

; Trigger an inspection in camera 1
CAMERA[1].CAPTURE_IMAGE=TRUE

; Wait for inspection to finish
WHILE (CAMERA[1].CAPTURE_IMAGE==TRUE)
WAIT SEC 0.0
ENDWHILE

; Has the data been written to robot?
CONTINUE
IF CAMERA[1].STATE==#NO_DATA
CAMERA[1].DATA_READY=TRUE
ENDIF

UNTIL (CAMERA[1].DATA_READY==TRUE)
END

; Display offsets of object found in window
DEF MSG_Display (DisplayX : IN,DisplayY : IN,DisplayA : IN)
decl Real DisplayX
decl Real DisplayY
decl Real DisplayA


; int pointer
DECL STATE_T STATE
        $MSG_T=EMPTY_MSG
        POINTER = 0
VISION_MSG={MSG_T: VALID TRUE,RELEASE TRUE,TYP
#NOTIFY,MODUL[] "VISION",KEY[] " ",PARAM_TYP #VALUE,PARAM[] "
",DLG_FORMAT[] " ",ANSWER 0}
swrite (VISION_MSG.KEY[],STATE,POINTER,"X = %f,Y = %f,A = %f
",DisplayX,DisplayY,DisplayA)
$MSG_T=VISION_MSG
END


DEF SAVEOFFSETPOS ( )
INT I
; Store last pos data on top of the list
FOR I=50 TO 2 STEP -1
SAVEDOFFSET[I]=SAVEDOFFSET[I-1]
ENDFOR
; Store last pos data
SAVEDOFFSET[1]=PICKDATA[1].PICKPOS
END
```

```
; Check to see what speed has been selected from SCADA
DEF SETVEL ( )

; First speed from SCADA slow
IF $IN[PICKDATA[1].CONVEYOR_BIT_26HZ] THEN
PICKDATA[1].PICKPOS.Y=PICKDATA[1].Y_26HZ
$VEL.CP=PICKDATA[1].CONVEYOR_VEL_26HZ
ENDIF

; Second speed from SCADA medium
IF $IN[PICKDATA[1].CONVEYOR_BIT_51HZ] THEN
PICKDATA[1].PICKPOS.Y=PICKDATA[1].Y_51HZ
$VEL.CP=PICKDATA[1].CONVEYOR_VEL_51HZ
ENDIF

; Third speed from SCADA fast
IF $IN[PICKDATA[1].CONVEYOR_BIT_77HZ] THEN
PICKDATA[1].PICKPOS.Y=PICKDATA[1].Y_77HZ
$VEL.CP=PICKDATA[1].CONVEYOR_VEL_77HZ
ENDIF
```