5-2015

Scalable Control Architecture for Variable-Rate Turn Compensation

Michael P. Sama
University of Kentucky, michael.sama@uky.edu

Joe D. Luck
University of Nebraska

Timothy S. Stombaugh
University of Kentucky, tim.stombaugh@uky.edu

Digital Object Identifier (DOI)
http://dx.doi.org/10.13031/aea.31.10848

TECHNICAL NOTE:

SCALABLE CONTROL ARCHITECTURE FOR VARIABLE-RATE TURN COMPENSATION

M. P. Sama, J. D. Luck, T. S. Stombaugh

ABSTRACT. *The objective of this study was to determine if a CAN bus could be used to implement variable-rate turn compensation in a manner that is scalable by encoding application rates for an entire implement into a single data message. A variable-rate turn compensation test fixture was developed that used a CAN bus to communicate application rates to 16 individual nodes using a 2-byte data message (80-bit extended identifier CAN messages). The system assumed that the physical structure of an implement was linear and that the control nodes were equally spaced. Application rates for the outer-most nodes were broadcasted and the remaining nodes calculated their application rate using a linear interpolation method. Node locations were determined using a 4-bit binary thumbwheel switch located at each control node, allowing all nodes to run an identical program. Servo-controlled gauges were used to visualize node application rate across the test fixture. A joystick interface was developed to simulate vehicle movements and desired application rates. The system transmitted Bluetooth serial messages at a rate of 20 Hz, which were received by the test fixture and converted to CAN messages before being broadcasted to the control nodes. Two USB to CAN interfaces were connected to the CAN bus to insert additional traffic and measure bandwidth utilization. Due to the minimal amount of bandwidth required (<1%) to transmit variable-rate control messages, the system functioned properly when the CAN bus was heavily loaded with traffic up to 99% of the available bandwidth of 250 kbps. The variable-rate turn compensation test fixture demonstrated that a CAN bus is a suitable protocol for communicating variable-rate data. The scalable encoding technique developed in this study resulted in a single message required to update all nodes, regardless of the number of nodes in the system. The system has broad applicability in future planting, fertilizing, and chemical application systems where deposition points are evenly spaced along an implement.*

Keywords. *Control, Controller area network, Precision agriculture, Turn compensation, Variable-rate.*

Variable-rate control uses spatial information to adjust the application rate of field inputs. Despite the published benefits of variable-rate control, it is an aspect of precision agriculture that has lagged behind other widely adopted management practices (Woods-DeWitt, 2008); contributing factors include large capital requirements and highly customized systems that only provide a single function. Consider a producer who purchases a GPS-based navigation aid or steering system. The same technology is easily used for different field operations and equipment configurations. In many cases, the technology is brand independent due to standardized protocols and interfaces. On the other hand, most variable-rate controllers are designed for a specific field application and cannot be readily interchanged between implements. The move towards standardized communication and hardware specific to variable-rate technology will allow fewer unique components across product lines and greater interoperability. A well-designed variable-rate control system that is scalable will be able to address application rate requirements for several field operations including, but not limited to, planting, fertilizing, and chemical applications.

Variable-rate technology can be subdivided into two categories: constant-coverage, where the goal is to uniformly apply a material across an area; and variable-coverage, where the application rate is adjusted based upon additional parameters. In a constant-coverage scenario, a centralized controller attempts to regulate the flow rate of a material to compensate for variations in speed and turning rate. The controller receives speed and heading information from an on-board GPS and uses this information to actuate individual or groups of deposition points such that the applied rate to the field surface is constant. In a variable-coverage scenario, the controller receives additional data from on-board sensors or prescription maps that are used to adjust the desired application rate with respect to position.

Submitted for review in July 2014 as manuscript number 10848; approved as a Technical Note for publication by the Machinery Systems Community of ASABE in December 2014.

The authors are **Michael P. Sama, ASABE Member**, Assistant Professor, Department of Biosystems and Agricultural Engineering, University of Kentucky, Lexington, Kentucky; **Joe D. Luck, ASABE Member**, Assistant Professor, Department of Biological Systems Engineering, University of Nebraska-Lincoln, Lincoln, Nebraska; **Timothy S. Stombaugh, ASABE Member**, Extension Professor, Department of Biosystems and Agricultural Engineering, University of Kentucky, Lexington, Kentucky. **Corresponding author:** Michael P. Sama, Department of Biosystems and Agricultural Engineering, 119 C.E. Barnhart Bldg., Lexington, KY 40546-0276; phone: 859-218-4325; e-mail: michael.sama@uky.edu.

A key limitation of existing variable-rate technology from a producer's standpoint is lack of verification that the system is functioning. For example, a common method for calculating the average application rate in a fixed-rate scenario is to measure the volume of fluid used over a known area. However, the average application rate is not always indicative of the actual distribution of rate of applied product. Local application errors from vehicle dynamics tend to average out over a large dataset. This may lead a producer to falsely conclude that the correct application rate was applied over the entire implement throughout the field.

Implement dynamics are a major cause of local errors in application rates and uniformity. Research has shown that implement acceleration and height affects performance (Jeon et al., 2004). Turning movements also influence local applications rates. The velocity at the individual application points varies as an implement turns. Luck et al. (2011) reported that failure to compensate for turning movements could result in off-rate errors of more than 10% from the target application occurring in as much as 6.5% to 23.8% of the total field area. The variability in the amount of off-rate error was due to differences in the amount of turning required to cover an individual field. The percentage of a field predicted to have an off-rate application was directly proportional to the percentage of field area covered while turning.

The control architecture of existing sprayer application equipment commonly involves the use of a single flow meter and/or pressure sensor to calculate the total flow of material from a reservoir. The overall flow is adjusted to match the desired rate of the entire system. An example system currently available for variable-rate control utilizes solenoids to rapidly turn individual nozzles on and off. The timing of this operation is controlled by parameters including vehicle speed, system fluid pressure, and system flow rate. Although these types of systems use real-time inputs, they should not be misconstrued as a closed-loop control of flow rate at the nozzle level. The control signal is merely calculated from a calibration curve that relates overall pressure and flow rate to individual nozzle flow rate. Many factors such as variations in fluid viscosity and normal wear on the system can potentially cause deviations between desired and actual flow rates. Research has also shown that turning individual valves on or off interferes with the pressure, and thus the flow rate, of surrounding nozzles (Sharda et al., 2010). Other common sources of application error are latency in processing data from the GPS receiver (Anglund and Ayers, 2003) and valve actuation delays (Sharda et al., 2011) which reduce the ability to adjust the system flow rate in response to pressure fluctuations caused by individual nozzle control.

A closed-loop system uses sensory feedback as part of a decision-making process to determine the state of an output. If the dynamics of the system to be controlled are well defined, then mathematical techniques to compensate for variability can be applied. Measuring pressure and flow rate at individual nozzles, as opposed to upstream in the system, will reduce application error due to pressure fluctuations and latency.

Distributed Control for Variable-Rate Applications

Distributed control systems are common in agricultural vehicles and some implements; however, they are not as commonly used in variable-rate technology due to bandwidth requirements from the large number of control nodes. A distributed control system uses a network, or shared communication bus, to communicate information between devices. Each device, referred to in this study as a node, has some computing capacity as opposed to a simple actuator or sensor connected to a single centralized controller that performs all computational tasks. There are several advantages to a distributed approach but the most visible effect is the reduction in the complexity of wiring. Large commercial sprayers can have boom lengths exceeding 30 m. Running control and sensor wiring to each nozzle can be cost prohibitive and exacerbate maintenance issues. Rough estimates indicate that a fully instrumented implement would require a minimum of four wires for every distribution point. The bundle of wiring returning to a centralized controller would exceed several inches in diameter. As a result, commercially available nozzle controllers typically have fewer outputs than the number of nozzles and virtually no sensory feedback. Instead of controlling nozzles individually, they are controlled in groups of nozzles referred to as sections.

One widely adopted network protocol is the controller area network (CAN). It was originally designed for the automotive industry (Bosch, 2001) and successfully adapted to the agricultural equipment industry (ISO, 2007). CAN (Ver. 2.0b) has a maximum specified bit-rate of 1 Mbps and is typically operated at 250 kbps per ISOBUS specifications. Sensors can be located close to the device processing their output which reduces susceptibility to noise. The cost of a distributed control system is proportional to the number of nodes. Since each individual controller is a fraction of the cost of a larger centralized controller, the financial risk associated with node failure is greatly reduced. Actuation latency due to complex processing can also be reduced as each node only computes a single output parameter.

Objectives

The overall objective of this study was to determine if a CAN bus could be used to implement variable-rate turn compensation in a manner that is scalable by encoding application rates for an entire implement into a single data message. Individual objectives were as follows:

1. Fabricate controller nodes capable of communicating over a CAN bus and generating PWM signals as variable-rate outputs.
2. Develop an algorithm for communicating and processing application rates through an encoded CAN message.
3. Fabricate a variable-rate test fixture and evaluate the system's performance.

Materials and Methods

CAN Controller Hardware

Sixteen individual CAN controllers were fabricated to serve as control nodes on a variable-rate turn compensation system. The controller was based on a design used by Sama et al. (2013) to synchronize serial data streams with GNSS time. Major components relevant to this study were a microprocessor (dsPIC30F4013, Microchip Technology Inc., Chandler, Ariz.), CAN transceiver (MCP2551, Microchip Technology Inc., Chandler, Ariz.), custom printed circuit boards (PCB), voltage regulators, light-emitting diodes (LEDs), resistors, capacitors, and connectors. The PCB layouts (figs. 8-13) and complete component lists (table 1) are included in the Appendix. Two PCBs were used in each node to limit the overall dimensions to $7.6 \times 7.6 \times 2.9$ cm ($3 \times 3 \times 1$-$1/8$ in.) (fig. 1).

The controllers were assembled in-house using a reflow soldering oven. Solder paste (Indium 8.9HF; Indium Corporation of America; Utica, N.Y.) was silk-screened on the PCBs using a 0.127 mm (0.005 in.) thick stencil and components were individually placed by hand. The reflow oven was configured to roughly follow the reflow soldering profile defined by the solder paste manufacturer. Oven temperature started at 150°C and linearly reached 180°C over 180 s. After reaching 180°C, the temperature was increased linearly to 237°C over 150 s. Cooling between the final soldering temperature and room temperature (25°C) was approximately 2°C s^{-1}.

The controllers were configured to use four active-low digital inputs with pull-up resistors for identifying the controller location, one digital output for generating pulse-width modulation (PWM) signals based on an internal timer, and two serial pins for CAN-H and CAN-L signals. Additionally, 5 VDC and ground were made available to power external components.

CAN Controller Software

A C program (MPLAB C30 C Compiler v3.31; Microchip; Chandler, Ariz.) was written using the Microchip Integrated Development Environment (IDE) (MPLAB IDE v8.87; Microchip; Chandler, Ariz.). The program configured the microcontroller in the following sequence:

1. Define the oscillator speed and pin aliases
2. Include C libraries for the dsPIC30F4013, delay functions, and CAN
3. Configure digital I/O port directions
4. Configure and enable the CAN and timer hardware peripherals
5. Wait for CAN messages to be received and process the messages into PWM outputs based on the controller location along the CAN bus

The complete program can be found in the Appendix.

The status of four digital inputs (RE0-RE3) were used to identify the position of each controller as an integer location along the CAN bus between 0 and 15. A linear interpolation method was devised that used two bytes of data in a single CAN message to calculate the control rate at every node along the bus. For this study, control rates of 0 to 100 were used to represent the minimum and maximum output rate, respectively. However, the control rates could be easily scaled over a 0 to 255 range to fully utilize the 8-bit data byte structure of the CAN protocol without increasing message length. The control rate was configured to set the position of a PWM servo (900-00005, Parallax Inc., Rocklin, Calif.), which provided a visual indication of the control rate in real-time. The PWM servo had a physical rotation range of 0° to 180° that corresponded to pulse widths of 2300 to 500 µs, respectively. The mapping used to convert control rates to servo output pulse width is show in equation 1.

$$P_w = 2300 - (18)C \qquad (1)$$

where

P_w = pulse width (µs),

C = control rate (integer 0-100).

Linear interpolation based on controller location was incorporated into equation 1 using two steps. The first step calculated the integer location of each controller based

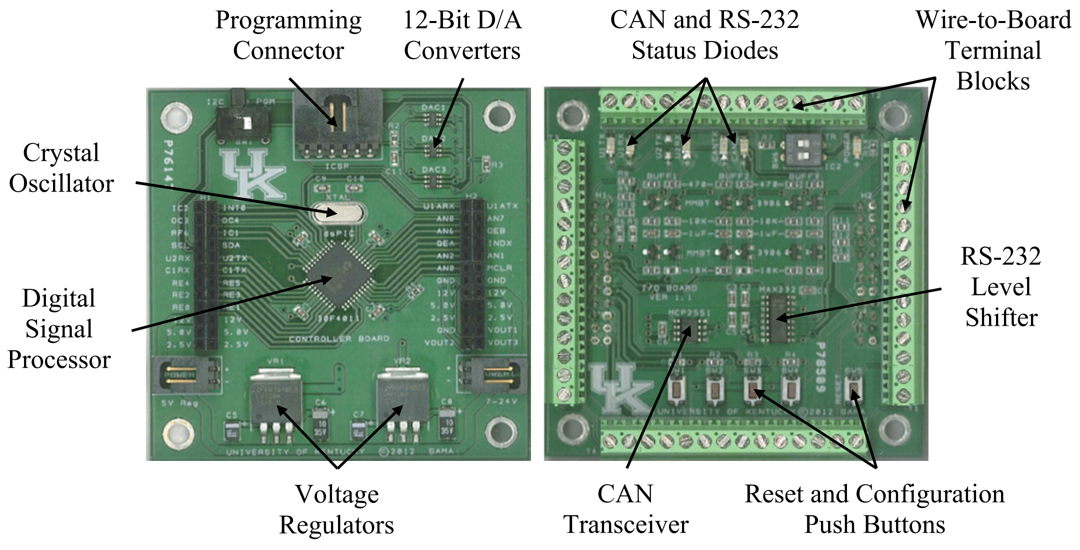


Figure 1. Bottom (left) and top (right) CAN Controller PCBs with components (Sama et al., 2013).

upon the status of the active-low digital inputs (eq. 2). The second step calculated the servo pulse width based on the integer location and the control rate specified for the outer-most control nodes (eq. 3).

$$n = 8(1-B_3) + 4(1-B_2) + 2(1-B_1) + (1-B_0) \quad (2)$$

where

n = integer address

B_0, B_1, B_2, B_3 = digital input bits (integer 0 or 1)

$$P_w = \frac{(2300-(18)C_1)(N-1-n) + (2300-(18)C_2)(n)}{N-1} \quad (3)$$

where

C_1 = left-most node control rate (integer 0 to 100),

C_2 = right-most node control rate (integer 0 to 100),

N = number of control nodes.

The CAN co-processor filter and mask settings were configured to accept extended identifier messages with an ID of 0 into receiver buffer 0 and ignore all other messages. CAN filters and masks directed bus traffic into receive buffers without the need to sort out messages using a software routine. The receive buffer was continually polled to determine if a CAN message had been received. When a new message had been received, the program read the first two bytes which contained the left-most and right-most control rates. The dsPIC30F4013 used 16-bit registers for CAN receive buffers which combined the first two data-bytes into a single 16-bit register. The 16-bit register was decoded directly into base-10 integer numbers using the modulus and division operators (eq. 4). The modulus operation retrieved the first 8-bits of the 16-bit register (R_0) as a base-10 integer (C_1) by returning the remainder of the division between the 16-bit register and 256_{10}. The division operator retrieved the second 8-bits as a base-10 integer (C_2) by truncating the first 8-bits of the 16-bit register. Therefore, R_0 register values between 0_{10} and 25700_{10} resulted in control rates for C_1 and C_2 between 0_{10} and 100_{10}. For example, the 16-bit register $R_0 = 12345_{10}$ would result in $C_1 = 57_{10}$ and $C_2 = 48_{10}$.

$$C_1 = R_0 \ \%256 \quad C_2 = R_0 \ /256 \quad (4)$$

where

R_0 = 16-bit CAN receive buffer register 0 (integer 0 to 65535).

Test Fixture

Individual controllers were mounted on a test fixture constructed from aluminum t-slotted framing and 18-gauge mild steel sheet metal (fig. 2). Sheet metal components were cut on a computer-numeric-controlled (CNC) plasma table and included mounting holes for components and access holes for wiring. Four-pin Deustch DT style connectors (AT06-4S, AT04-4P, Amphenol, Wallingford, Conn.) and 18-AWG stranded wire were used to construct a wiring harness which provided power and CAN signals to each controller. A 120 W 12 VDC power supply, supplied with 120 VAC, was used to power the test fixture. A power entry module in series with a rocker switch allowed power to be physically disconnected and selectively enabled.

Visual gauges were fabricated from 0.32 cm (1/8 in.) acrylic plastic on a CNC milling machine. The servos were mounted behind the gauge and a dial indicator, fabricated on a 3D printer from ABS-like plastic, was attached to each servo (fig. 3). Each servo was powered by the 5 VDC regulator on the corresponding controller and received PWM signals at a rate of 50 Hz.

A four-bit binary thumbwheel switch (CH688, Cherry, Pleasant Prairie, Wis.) was mounted above each controller to set the node location along the test fixture. The switch was connected to four digital inputs (RE0-RE3) and ground. A switch state of 0 set all digital inputs at logic high through pull-up resistors while a switch state of 15 set all digital inputs to a logic low by grounding the digital inputs.

A Bluetooth-to-CAN interface (CANblue II, IXXAT, Chicago, Ill.) was connect to the CAN bus to provide a serial interface between a PC and the test fixture. The interface used the virtual com port driver, which allowed software to be written using Visual Studio 2010 (Microsoft; Redmond, Wash.) without using custom interface drivers.

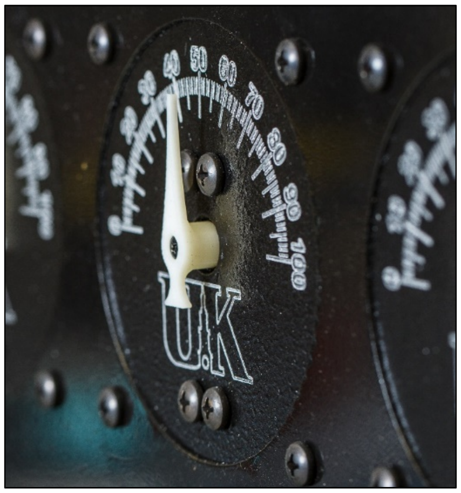


Figure 3. Visual gauge representation of application rate.

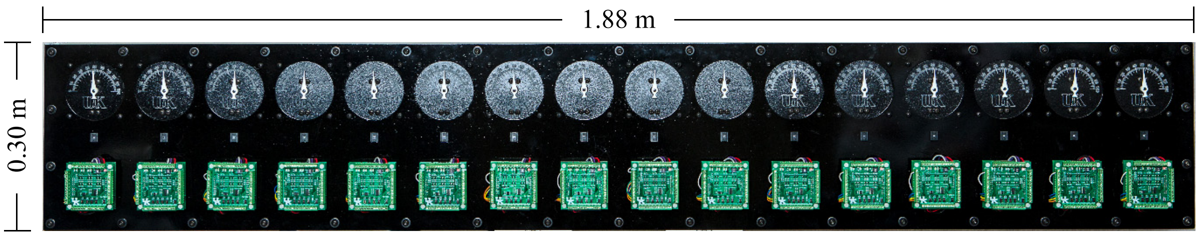


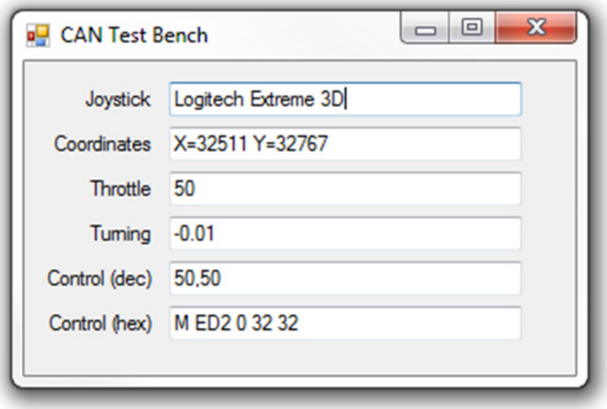
Figure 2. Test fixture.

Two USB-to-CAN interfaces (Leaf Light HS; Kvaser; Mölndal, Sweden) were connected to a PC and used to generate CAN bus traffic and record bus utilization (fig. 4).

INTERFACE SOFTWARE

An interface program was written to allow a user to simulate the motion of an agricultural vehicle using a joystick and transmit an encoded application rate based on the joystick position (fig. 5). The program used the DirectX DirectInput (Version 1.0.2902.0; Microsoft; Redmond, Wash.) plug-in to sample the joystick position and convert position to application rates.

The program was configured to connect to the virtual serial communication port assigned to the Bluetooth-to-CAN interface. The user display contained six labels and associated text boxes for displaying data. The "Joystick" label displayed the joystick model to indicate which joystick was used as a control input. The program automatically selected the first available joystick if any were connected to the computer. The "Coordinates" label displayed the current position of the x and y axis as a 16-bit integer. The joystick position output for each axis ranged from 0 to 65535 and was oriented in the first quadrant of a Cartesian coordinate system. The "Throttle" label displayed a simulated throttle position based on the y-axis of the joystick and ranged from 0 to 100. The "Turning" label displayed a simulated turning rate that was based on the x-axis of the joystick and ranged from -1.00 to 1.00. The "Control (dec)" label was a base-10 integer representation of the desired application rate at both ends of the CAN test fixture with values ranging from 0 to 100. The "Control (hex)" label was a CAN message formatted for the Bluetooth-to-CAN interface and contained two data bytes in hexadecimal format ranging from 0×00 to 0×64. The resulting CAN message was capture from the CAN HIGH



Joystick	Logitech Extreme 3D
Coordinates	X=32511 Y=32767
Throttle	50
Turning	-0.01
Control (dec)	50,50
Control (hex)	M ED2 0 32 32

Figure 5. CAN test bench interface software.

signal using a digital oscilloscope (DPO 4034, Tektronix, Beaverton, Ore.) to visualize the message format (fig. 6).

The example CAN message shown in figure 6 demonstrates the protocol structure of the CAN message including the extended identifier (Id), number of bytes of data, individual data bytes (D), and the checksum (CRC) for error checking, in order from left to right.

CAN BUS TESTING

Two instances of CanKing (Version 5.00.229, Kvaser, Mölndal, Sweden) were run to generate CAN bus traffic from the USB-to-CAN interfaces. Dummy messages, or messages not intended for the test fixture, were transmitted from both interfaces in addition to the 20 Hz CAN data used to control the test fixture. The dummy message had extended identifiers of 0×0001 or 0×0002 and eight data bytes of 0×FF resulting in 128-bit length CAN messages. The CanKing program was capable of transmitting the dummy message at a fixed interval. The interval was

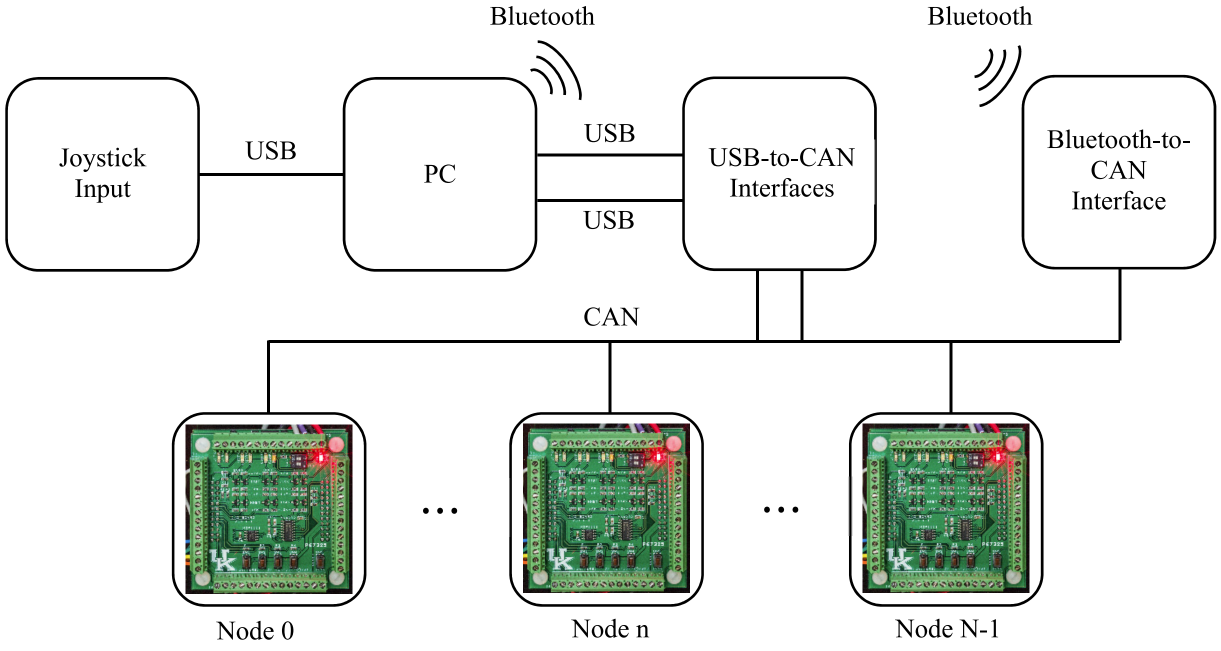


Figure 4. Test fixture communication schematic.

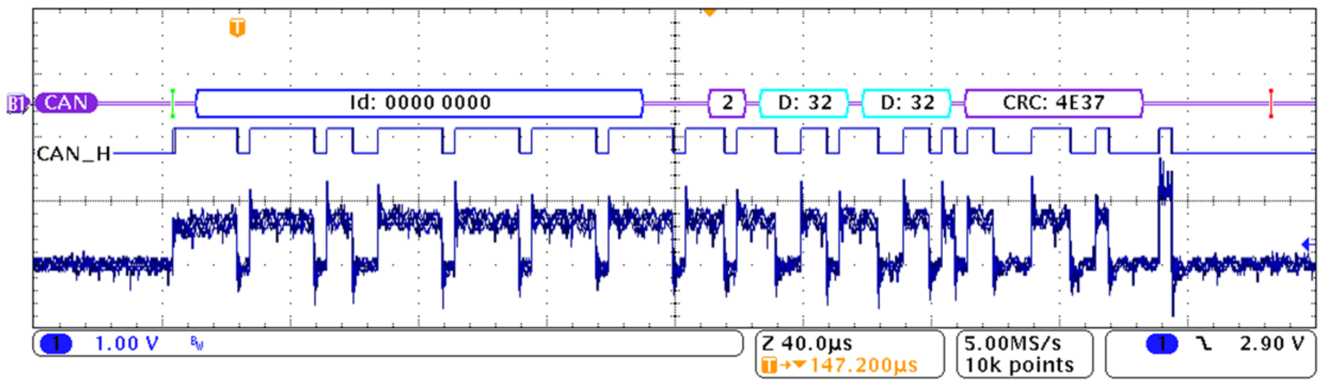


Figure 6. Two-byte extended ID CAN message.

controllable between 1 and 1000 ms in increments of 1 ms, resulting in a theoretical maximum data message rate of 2000 Hz. Note that this data rate exceeded the bandwidth limitations of a 250 kbps CAN bus when transmitting 8-byte extended identifier messages. Therefore, the expected maximum data message rate was less than 2000 Hz.

Results and Discussion

Response Time

Extended identifier data messages were broadcasted from the interface software to the CAN test bench at a data message rate of 20 Hz. The transmission time, or the time that it took to transmit the 80-bit CAN message, was 320 µs. Given that all nodes received the broadcasted message at nearly identical times, there was no difference in response time between individual nodes. If the same data structure had been used but with individual identifiers for each node, the time delay between when the first node and the last node received a new data message would have been 5.12 ms. The delay between the first and last node will increase linearly with the number of nodes used in the control system. If, for instance, 100 nodes were used to control 100 individual nozzles on a sprayer, the minimum delay between the first and last node would be 32 ms. On a self-propelled sprayer travelling at 10 m/s, 32 ms of delay would result in 0.32 m of position error at the last node solely due to the time it takes for all preceding messages to be sent. The estimated position error increases when adding the fact that boom tips may move much faster than the vehicle speed due to turning movements.

Bus Loading

Each USB-to-CAN interface was capable of individually generating CAN messages at 1000 Hz. The maximum combined CAN bus message rate was determined to be approximately 1650 Hz and was a result of the CAN baud rate and message format. The USB-to-CAN interfaces were not capable of loading the bus beyond 100% or intentionally generating bus collisions. As a result, The relationship between data update rate and percentage of bus utilization was determined to be linear over the entire range of bus utilization when only three devices were transmitting on the bus (fig. 7). Application rate CAN messages were successfully transmitted, regardless of bandwidth usage, as long as bus collisions were not present. An actual CAN bus on an agricultural vehicle or implement may have substantially more nodes attempting to transmit at the same time which could result in arbitration, a process where nodes determine message priority, as well as collisions between nodes transmitting data with the same priority level. However, typical CAN bus implementations are rarely designed to use 100% of the available bandwidth. The worst-case scenario for interference in transmitting application rate CAN messages will be a short delay between when the data is available and when it is transmitted over the bus.

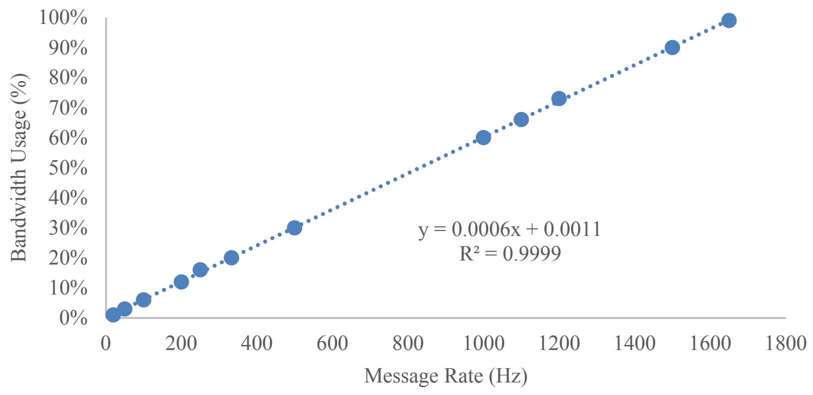


Figure 7. Data update rate vs. bandwidth usage.

CONCLUSION

Sixteen CAN-based control nodes were designed and fabricated as part of a test fixture for evaluating a variable-rate turn compensation control scheme. A control scheme was developed to allow individual nodes to simultaneously calculate their application rate from a single CAN message containing two data bytes. The data bytes in each CAN message corresponded to the application rates at the outer-most control nodes. Nodes calculated their respective application rate using linear interpolation. The system required two assumptions, (1) that the target implement was a linear structure and (2) that application nodes were evenly spaced.

The user interface software developed to test the control method was capable of transmitting 20 Hz control data, regardless of CAN bus utilization. Results showed that the control data were successfully transmitted to the control nodes at bandwidth utilization rates up to 99% when bus traffic was generated at a constant interval. Not all data are transmitted at the same interval on a typical CAN bus so the bandwidth utilization can vary over time. Additional testing should be conducted to determine the effect of bus collisions and un-equally spaced data rates.

The test fixture successfully demonstrated that the CAN bus is a suitable protocol for implementing a large number of control nodes when using an encoded data message to broadcast application rate to all nodes, simultaneously. This methodology addressed a bottleneck of current systems by eliminating bandwidth concerns for applications like modern high clearance sprayers with implement widths in excess of 40 m and over 100 dispersion points. Existing manufacturers of CAN-based variable-rate systems that operate over standardized protocols, such as ISOBUS, will be able to implement the encoding and decoding method outlined in this study through simple firmware updates to their systems. The substantial reduction in bandwidth required for control will allow more flexibility in CAN bus utilization for other tasks such as data acquisition.

In this study, the total number of nodes was hard-coded into the controller software. Future work will involve developing initialization routines to improve node identification and to make the system more scalable by self-identifying the total number of nodes every time the system is powered on.

ACKNOWLEDGEMENTS

The authors would like to thank John T. Evans for his assistance in the fabrication of the CAN controllers used in this study. This material is based upon work supported by the National Institute of Food and Agriculture, U.S. Department of Agriculture, under Agreements 2010-34628-21691 and KAES 14-02. Any opinions, findings, or conclusions expressed in this publication are those of the author(s) and do not necessarily reflect the views of the U.S. Department of Agriculture or the Kentucky Agricultural Experiment Station. The information reported in this paper (No. 15-05-054) is part of a project of the Kentucky Agricultural Experiment Station and is published with approval of the Director.

REFERENCES

Anglund, E. A., & Ayers, P. D. (2003). Field evaluation of response times for a variable rate (pressure-based and injection) liquid chemical applicator. *Appl. Eng. Agric., 19*(3), 273-282. http://dx.doi.org/10.13031/2013.13659.

Bosch. (2001). D-70442: CAN Specification. Stuttgart, Germany: Bosch Gmbh.

ISO. (2007). 11783-1: Tractors and machinery for agriculture and forestry—Serial control and communications data network—Part 1: General standard for mobile data communication. ISO/DIS.

Jeon, H. Y., Womac, A. R., & Gunn, J. (2004). Sprayer boom dynamic effects on application uniformity. *Trans. ASAE, 47*(3), 647-658. http://dx.doi.org/10.13031/2013.16094.

Luck, J. D., Pitla, S. K., Zandonadi, R. S., Sama, M. P., & Shearer, S. A. (2011). Estimating off-rate pesticide application errors resulting from agricultural sprayer turning movements. *Precision Agric., 12*(4), 534-545. http://dx.doi.org/10.1007/s11119-010-9199-9.

Sama, M. P., Stombaugh, T. S., & Lumpp, J. E. (2013). A hardware method for time-stamping asynchronous serial data streams relative to GNSS time. *Computers Elec. Agric., 97*, 56-60. http://dx.doi.org/10.1016/j.compag.2013.07.003.

Sharda, A., Fulton, J. P., McDonald, T. P., & Brodbeck, C. J. (2011). Real-time nozzle flow uniformity when using automatic section control on agricultural sprayers. *Computers Elec. Agric., 79*(2), 169-179. http://dx.doi.org/10.1016/j.compag.2011.09.006.

Sharda, A., Fulton, J. P., McDonald, T. P., Zech, W. C., Darr, M. J., & Brodbeck, C. J. (2010). Real-time pressure and flow dynamics due to boom section and individual nozzle control on agricultural sprayers. *Trans. ASABE, 53*(5), 1363-1371. http://dx.doi.org/10.13031/2013.34891.

Woods-DeWitt, M. T. (2008). Precision agriculture in the corn belt region of Ohio 2007: A Double hurdle model estimating farmer characteristics resulting in adoption and satisfaction.MS thesis. The Ohio State University.

APPENDIX

BOARD LAYOUTS (SCALE = 1:1)

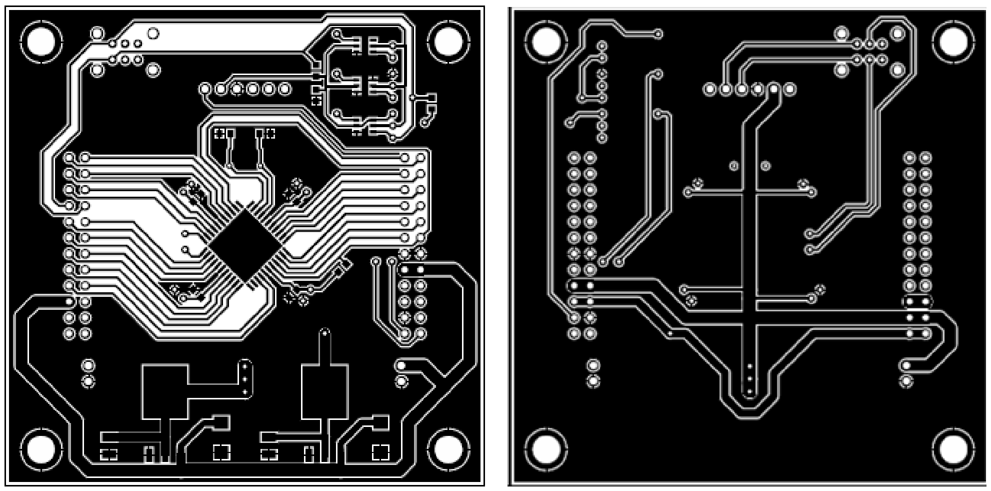


Figure 8. Controller board top (left) and bottom (right) copper.

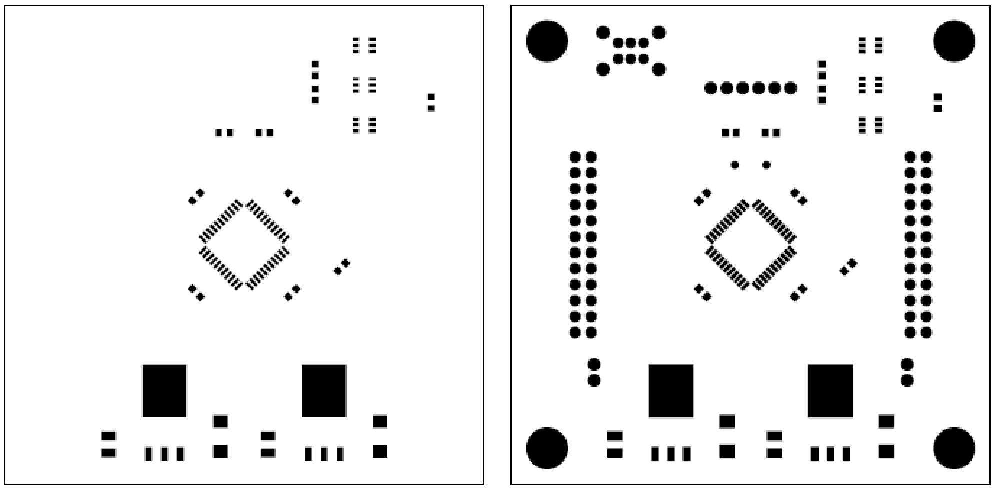


Figure 9. Controller board paste mask (left) and solder mask (right).

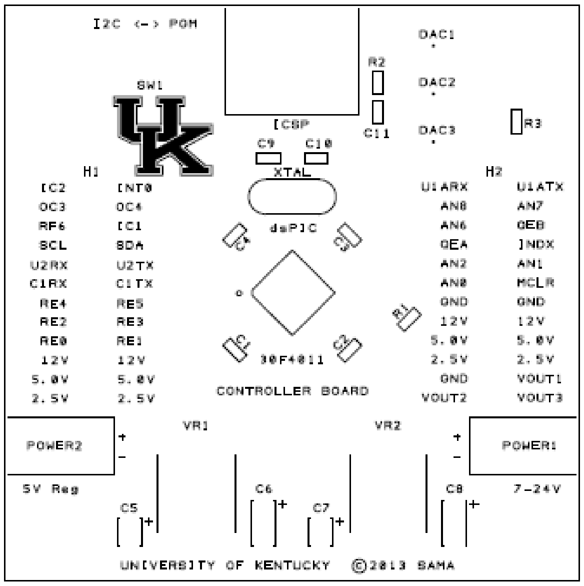


Figure 10. Controller board silk screen.

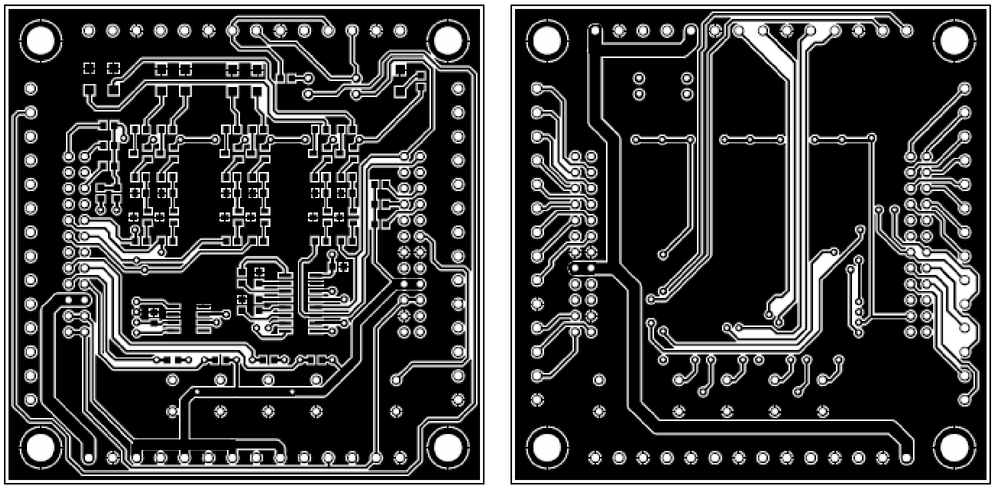


Figure 11. I/O interface board top (left) and bottom (right) copper.

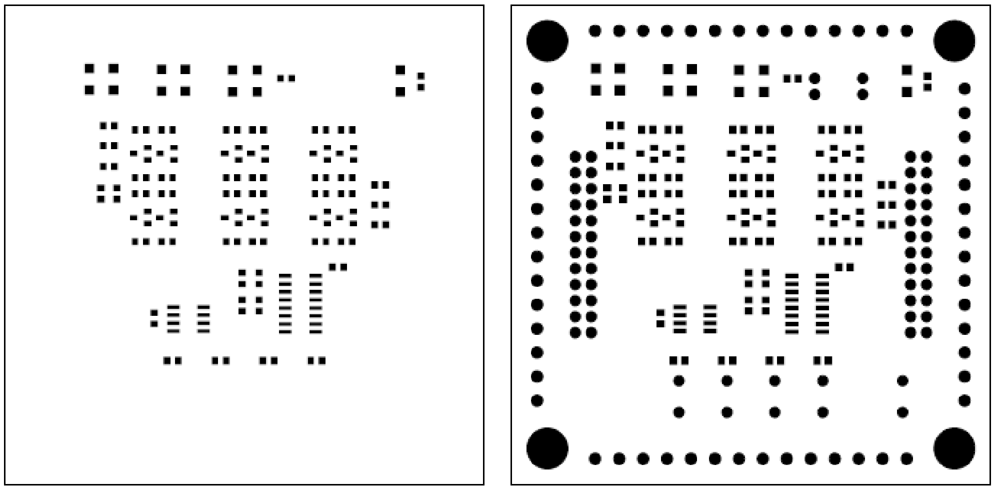


Figure 12. I/O interface board paste mask (left) and solder mask (right).

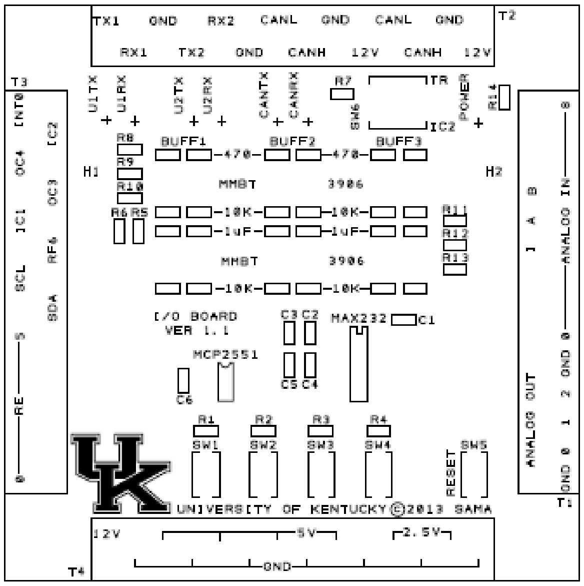


Figure 13. I/O interface board silk screen.

Component List

Table 1. CAN Node Component List.

Component	Part Number	Manufacturer	PCB Label
Controller Board			
Microcontroller	DSPIC30F4011-30I/PT	Microchip Technology	dsPIC30F4011
12-Bit Digital-to-Analog Converter	DAC121C081CIMM/NOPB	National Semiconductor	DAC1, DAC2, DAC3
5V Regulator	LM1084IS-5.0/NOPB	National Seminconductor	VR1
2.5V Regulator	LM1086CS-2.5/NOPB	National Seminconductor	VR2
24 Pin Receptical	1-534206-2	TE Connectivity	H1, H2
6 Pin Right-Angle Connector	70553-0005	Molex	ICSP
2 Pin Right-Angle Connector	70553-0001	Molex	POWER1, POWER2
15 MHZ Crystal Oscillator	HC49US-15.000MABJ-UB	Citizens Finetech Miyota	XTAL
2312 Capacitor 10 uF Tantalum	F931V106MCC	Nichicon	C6, C8
1210 Capacitor 10 uF Tantalum	F931A106MBA	Nichicon	C5, C7
0603 Capacitor 0.1 uF	C1608X7R1C104K	TDK Corporation	C1, C2, C3, C4, C11
0603 Capacitor 22 pF	06035A220JAT2A	AVX Corporation	C9, C10
0603 Resistor 10 kOhm	CRCW060310K0FKEA	Vishay Dale	R1
0603 Resistor 4.7 kOhm	CRCW06034K70FKEAHP	Vishay Dale	R2, R3
DPDT Side PCB Switch	MMS22R	TE Connectivity	SW1
I/O Interface Board			
Component	Part Number	Manufacturer	PCB Label
Tactile SPST Switch	EVQPJJ04T	Panasonic	SW1, SW2, SW3, SW4, SW5
10 Position 0.150” Terminal Block	1-284392-0	TE Connectivity	T1 ,T2, T3, T4
2 Position 0.150” Terminal Block	284392-2	TE Connectivity	T1 ,T2, T3, T4
2 Position SPST DIP Switch	ADE0204	TE Connectivity	SW6
Yellow 1206 LED	LTST-C150KSKT	Lite-On Inc.	U1RX, U2RX, CANRX
Red 1206 LED	LTST-C150KRKT	Lite-On Inc.	POWER
Green 1206 LED	LTST-C150KGKT	Lite-On Inc.	U1TX, U2TX, CANTX
PNP Transistor	MMBT3906	Fairchild Semiconductor	MMBT, 3906
Dual RS232 Level Shifter	MAX232DR	Texas Instruments	MAX232
CAN Transceiver	MCP2551-I/SN	Microchip Technology	MCP2551
0603 Capacitor 0.1 uF	C1608X7R1C104K	TDK Corporation	C6
0603 Capacitor 1uF	C1608Y5V1E105Z	TDK Corporation	C1, C2, C3, C4, C5, 1uF
0603 Resistor 10 kOhm	CRCW060310K0FKEA	Vishay Dale	R1 - R6, R8 - R13, 10K
0603 Resistor 470 Ohm	CRCW0603470RJNEAHP	Vishay Dale	470
0603 Resistor 120 Ohm	CRCW0603120RFKEA	Vishay Dale	R7
24 POS Pin Header (80 Pins)	4-103186-0	TE Connectivity	H1, H2

Controller Program

```
//////////////////////////////////////////////////
//   Title: CANBasedTurnCompensation.c         //
//   Author: Michael P. Sama                   //
//   Date: May, 2012 - June, 2014              //
//////////////////////////////////////////////////
#define SYSCLK          15000000UL  //Define the system clock speed as 15 MHz.
#define FCY 3750000UL        //Define the instruction clock speed as 3.75 MHz.
//Pin aliases for 4-bit board position thumbwheel switch.
#define   B1   PORTEbits.RE0   //LSB.
#define   B2   PORTEbits.RE1.
#define   B3   PORTEbits.RE2.
#define   B4   PORTEbits.RE3   //MSB.
#include <p30fxxxx.h>   //Base Library for the dsPIC30F4013.
#include <libpic30.h>   //Software Delay Library.
#include <can.h>        //CAN Library.
_FOSC(XT)        //Set the oscillator to external high speed crystal.
_FWDT(WDT_OFF)   //Turn off the watch dog timer.
/* Main Function */.
  int main (void).
{.
   TRISC = 0b101111111111111;   //Configure PORTC I/O directions.
   TRISD = 0b01;                //Configure PORTD I/O directions.
   TRISE = 0b001111;            //Configure PORTE I/O directions.
   TRISF = 0b0101;              //Configure PORTF I/O directions.
     __delay_ms(500);             //Wait 0.5 s.
  /* Configure Timer 2 */.
  T2CONbits.TSIDL = 0;   //Continue timer operation in Idle mode.
  T2CONbits.TGATE = 0;   //Timer gated time accumulation disabled.
  T2CONbits.TCKPS = 1;   //Timer input clock pre-scale bits set to 1:64.
  T2CONbits.T32=0;       //Timer2 and Timer3 form separate 16-bit timers.
  T2CONbits.TCS=0;       //Internal timer clock (FOSC/4).
  T2CONbits.TON=1;       //Start Timer2.
  /* Configure Output Compare 2 */.
  OC2CONbits.OCSIDL = 0;   //Continue in idle mode.
  OC2CONbits.OCFLT = 0;    //Reset any fault conditions.
  OC2CONbits.OCTSEL = 0;   //Timer2 is the clock source for output compare 2.
  OC2CONbits.OCM = 6;      //PWM mode on OC2, fault pin disabled.
  /* Set request for configuration mode */.
  CAN1SetOperationMode(CAN_IDLE_CON  & CAN_CAPTURE_DIS & CAN_MASTERCLOCK_0 &.
                       CAN_REQ_OPERMODE_CONFIG);.
  while(C1CTRLbits.OPMODE !=4);  //Wait for configuration mode.
  /* Configure CAN Registers */.
  C1CTRLbits.CANCKS = 0;     //CAN Master Clock (TQ) is set to 4 FCY.
    C1CFG1bits.SJW = 0;        //Synchronized jump width = 1 x TQ.
    C1CFG1bits.BRP = 2;        //Baud rate pre-scaler = 8.
  C1CFG2bits.WAKFIL = 0;     //CAN bus line filter is not used for wake-up.
  C1CFG2bits.SEG2PHTS = 1;   //SEG2 is freely programmable.
  C1CFG2bits.SEG2PH = 5;     //Phase Segment 2 = 5 x TQ.
  C1CFG2bits.SEG1PH = 8;     //Phase Segment 1 = 8 x TQ.
  C1CFG2bits.SAM = 0;        //Bus line is sampled once at each sample point.
  C1CFG2bits.PRSEG = 1;      //Propagation time segment = 2 x TQ
    /* Load Acceptance Filter Register 0 */.
  CAN1SetFilter(0, CAN_FILTER_SID(0) & CAN_RX_EID_EN,
                CAN_FILTER_EID(0));.
  /* Load Mask Filter Register 0 */.
  CAN1SetMask(0, CAN_MASK_SID(3) & CAN_MATCH_FILTER_TYPE,
              CAN_MASK_EID(0));.
  C1RXM0SID = 0b0001111111111111;  //Include all bits in filter comparison.
  /* Set CAN Receiver Mode */.
  CAN1SetRXMode(0, CAN_RXFUL_CLEAR & CAN_BUF0_DBLBUFFER_EN);.
  /* Set request for Normal mode */.
  CAN1SetOperationMode(CAN_IDLE_CON & CAN_CAPTURE_DIS & CAN_MASTERCLOCK_0 &.
                       CAN_REQ_OPERMODE_NOR);.
  while(C1CTRLbits.OPMODE != 0);  //Wait for normal mode.
  C1RX0CONbits.RXFUL = 0;         //Clear receive buffer 0 flag.
  /* Set the Initial Servo Position and Node Location*/.
    unsigned int SERVO = 2300;  //2300 ms = gauge value 0.
  unsigned int DATA1 = 0;  //Default to..
  unsigned int DATA2 = 0;  //location 0.
  /* Calculate the PWM Parameters From the Desired Servo Pulse Width */.
  OC2RS = (unsigned int) (((unsigned long) SERVO * 468) / 1000);.
  PR2 = OC2RS+9372;   //Set the period register.
  while(1)  // do forever.
  {.
       if(C1RX0CONbits.RXFUL)  //Receiver buffer 0 contains a CAN message.
     {.
        DATA1 = C1RX0B1%256;      //Retrieve the first data byte.
        DATA2 = C1RX0B1/256;      //Retrieve the second data byte.
        C1RX0CONbits.RXFUL = 0;   //Reset the receive buffer 0 flag.
          /* Calculate the Node Location from the Thumbwheel Switch */.
        CAN_ADDR = 8*(1-B4) + 4*(1-B3) + 2*(1-B2) + (1-B1);.
        /* Calculate the Servo Pulse Width */      .
        SERVO = ((2300 - DATA1 * 18)*(15-CAN_ADDR)) / 15 + ((2300 - DATA2 *
                  18)*(CAN_ADDR)) / 15;.
        /* Calculate the PWM Parameters from the Desired Servo Pulse Width */.
        OC2RS = (unsigned int) (((unsigned long) SERVO * 468) / 1000);.
        PR2 = OC2RS+9372;   //Set the period register.
      } .
     }.
     return 0;.
}.
```