MEMORANDUM

To: VSCADA Team
From: Dyno Team
Subject: Dyno to VSCADA interface design
Date: 3/5/2015

Introduction:

The purpose of this memo is to detail how to interface the VSCADA system to acquire data from the dynamometer and motor controller as well as how VSCADA can control the dynamometer and the throttle of the motor controller.

Curtis Motor Controller:

The Curtis software is proprietary and closed source. The creates several limitations for interfacing with the motor controller. The motor controller has to be programmed. Curtis provides two solutions for this. One being a Windows program and the other being a handheld programmer. The handheld programmer only offers limited capability and is more expensive. For this reason the Windows software was purchased but this presents another problem. The VSCADA team runs linux and therefore cannot run the programming software. Since the motor controller must be programmed in Windows, a Windows computer must be used to program the controller. With that understanding, the parameters being programmed by the Windows software should simply be set once and not be changed during operation. Knowing the limits of the car this should pose as a minimal inconvenience as the motor controller should seldom need reprogrammed.

Data Acquisition

Data acquisition from the motor controller is handled using the operating system on the motor controller. The data is communicated over CAN from the motor controller to the VSCADA computer. The CAN data is available at addresses 0x601 and 0x0602. Each address is eight bytes. A chart outlining the data is available below and was taken from the manual for the motor controller.

Generic CAN Messages from Controller

	ADDRESS ID					
	CAN ADDRESS 0x601	Units	Scale	CAN ADDRESS 0x602	Units	Scale
Byte0	Motor RPM high byte	RPM	1	Stator Frequency high byte	Hz	1
Byte1	Motor RPM low byte			Stator Frequency low byte		
Byte2	Motor Temp	Deg C	-40 to 200	Controller Fault Primary		
Byte3	Controller Temp			Controller Fault Secondary		
Byte4	RMS Current high byte	Amps	0.1	Throttle Input	%	1
Byte5	RMS Current low byte			Brake Input		
Byte6	Capacitor Voltage high byte	Volts	0.1	System Bits*		
Byte7	Capacitor Voltage low byte			Not used		

*** System bits configuration**

Bit	Logic
0	Econo bit
1	Regen bit
2	Reverse bit
3	Brake Light Bit

VSCADA Control

VSCADA only needs to control the throttle input to the motor controller. The throttle input on the motor controller is a potentiometer that handles voltages from 0 to 5 volts. Pin 16 is the where the voltage needs to be applied. Pin 7 is the ground. The Dyno team is using a microcontroller to output a computer controllable throttle voltage input to the motor controller. A similar approach could be used by the VSCADA team.

Huff Dynamometer:

The dynamometer provided by Huff Industries also created a number of limitations. The provided software works, but is written in visual basic which is designed to only work with Windows OS. This creates a problem for data acquisition and control. The VSCADA computer runs linux and adding a windows computer to run the software creates two systems that now have to work together. In order to have only one system, the software provided by Huff is being scrapped altogether and open source software from Measurement Computing, the company who makes the data acquisition chip inside the Huff data acquisition box, is being used to give the VSCADA computer data access and control of the Huff dynamometer.

Data Acquisition

Data acquisition is being handled through the FlexTest software provided by Measurement Computing. The software establishes a serial connection between the PIC chip on the USB-7204 board and the VSCADA computer via USB. The data can then be acquired by making a request to the PIC chip and a reply will be returned. The format of the request is outline in the manual for the FlexTest. Data can also be acquired by writing code to make the requests over the serial connection. This is a more ideal situation but requires some overhead. Measurement Computing suggests using Mono, which is a C# interpreter, to make the requests using the format outlined in the manual. This has been brought up an a potential instable and inextensible situation. While it should work for the short term, a more long term and maintainable solution would be to learn what the C# methods are doing and write them using custom code. This would provided for maintainability and extensibility for the future of the project.

VSCADA Control

Control of the dynamometer is done much the same as the data acquisition. The command and response method outlined in data acquisition also works for controlling channels connected to the PIC chip. It is as simple as send the right command to the right channel to change or set values. A full diagram of the channels is provided below. The VSCADA team will also have the ability to control the dynamometer valve. The range of values to set to control the opening of the valve is 0 to 5 volts.

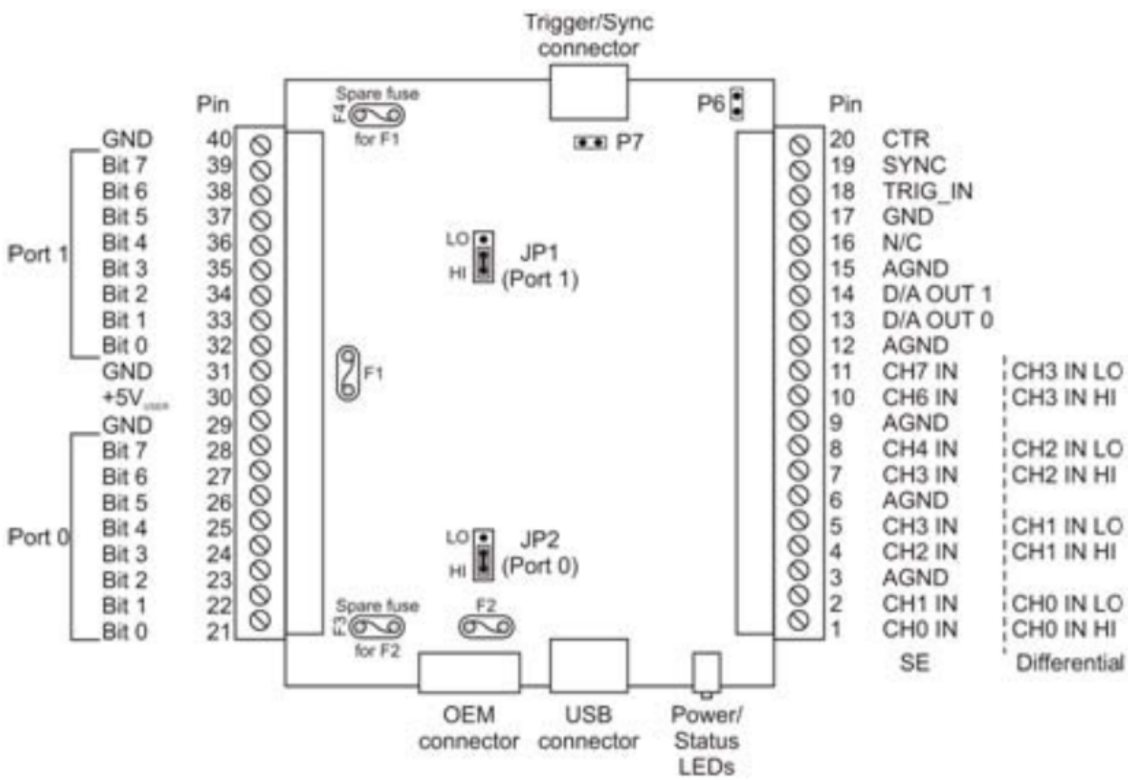

Figure 4. Screw terminal pinout

Component	Supported Property/Command	Set/Get	Supported Values
AI		Get	Number of analog input channels
	CAL	Set/Get	ENABLE, DISABLE
	CHMODE	Set/Get	SE, DIFF
	SCALE	Set/Get	ENABLE, DISABLE
AI{ch}	OFFSET	Get	4-byte floating point numeric
	RANGE	Set/Get	BIP20V, BIP10V, BIP5V, BIP4V, BIP2PT5V, BIP2V, BIP1PT25V, BIP1V
	SLOPE	Get	4-byte floating point numeric
	VALUE	Get	Unsigned integer numeric
	VALUE/{format}	Get	RAW, VOLTS
AISCAN	BUFOVERWRITE	Set/Get	ENABLE, DISABLE
	BUFSIZE	Set	Set/Get
	CAL	Set/Get	ENABLE, DISABLE
	COUNT	Get	
	EXTPACER	Set/Get	ENABLE/MASTER, ENABLE/SLAVE, ENABLE/GSLAVE
	HIGHCHAN	Set/Get	0 to 7 single-ended, 0 to 3 differential
	INDEX	Get	
	LOWCHAN	Set/Get	0 to 7 single-ended, 0 to 3 differential (must be ≤ HIGHCHAN)
	QUEUE	Set/Get	ENABLE, DISABLE, RESET
	RANGE{ch}	Set/Get	BIP20V, BIP10V, BIP5V, BIP4V, BIP2PT5V, BIP2V, BIP1PT25V, BIP1V
	RANGE{element/ch}	Set	Element: 0 to 15 Channel: 0 to 7 single-ended, 0 to 3 differential Range: see the range values above.
	RATE	Set/Get	0.596 Hz to 50,000 Hz (1 channel)
	SAMPLES	Set/Get	0 to N (0 = continuous scan; N = 32-bit)
	SCALE	Set/Get	ENABLE, DISABLE
	START		
	STATUS	Get	IDLE, RUNNING, OVERRUN
	STOP		
	TRIG	Set/Get	ENABLE, DISABLE
	XFRMODE	Set/Get	BLOCKIO, SINGLEIO
AITRIG	Type	Set/Get	EDGE/RISING, EDGE/FALLING
	REARM	Set/Get	ENABLE, DISABLE

Component	Supported Property/Command	Set/Get	Supported Values
AO		Get	Number of analog output channels
	CAL	Set/Get	ENABLE, DISABLE
	SCALE	Set/Get	ENABLE, DISABLE
AO{ch}	RANGE	Get	UNI4.096V
	VALUE	Set	0 to 4095
AOSCAN	BUFSIZE	Set/Get	
	COUNT	Get	
	HIGHCHAN	Set/Get	0 to 1
	INDEX	Get	
	LOWCHAN	Set/Get	0 to 1
	RANGE	Get	UNI4.096V
	RATE	Set/Get	1 kHz to 10 kHz (1 channel)
	SAMPLES	Set/Get	0 to N (0 = continuous scan; N = 32-bit)
	SCALE	Set/Get	ENABLE, DISABLE
	START		
	STATUS	Get	IDLE, RUNNING, UNDERRUN
	STOP		
CTR		Get	Number of counter channels
CTR{ch}	START		
	STOP		
	VALUE	Get	0 to 4,294,967,295
		Set	0
DEV	FLASHLED/{n}	Set	0 to 255
	FWV	Get	MM.mm (M = major; m = minor)
	ID	Set/Get	Up to 56 characters
	MFGCAL	Get	yyyy-mm-dd HH:MM:SS
	MFGCAL{item}	Get	YEAR as yyyy; 20xx MONTH as mm; 01 to 12 DAY as dd; 01 to 31 HOUR as HH; 01 to 23 MINUTE as MM; 01 to 59 SECOND as SS; 01 to 59
	MFGSER	Get	Up to 8 hexadecimal digits
DIO		Get	Number of digital ports
DIO{port}	DIR	Set/Get	IN, OUT (port-configurable)
	VALUE	Set/Get	0 to 255
DIO{port/bit}	VALUE	Set/Get	0 or 1