1310 Manual

MANUAL

MODEL

1310

VEHICLE SYSTEM CONTROLLER with VCL

1310 Manual, p/n 36488001
Rev. A: August 2007

CURTIS INSTRUMENTS, INC.

200 Kisco Avenue
Mt. Kisco, New York 10509 USA
Tel. 914.666.2971
Fax 914.666.2188

www.curtisinstruments.com

CONTENTS

1 OVERVIEW

Curtis 1310 provides unprecedented flexibility and ease of use in a full featured programmable Vehicle System Controller. Containing FLASH memory, a powerful microcontroller and a wide range of inputs and outputs, the 1310 can be custom programmed to provide application specific vehicle functions, from the most complex to the most unique. Custom software for the 1310 is developed with the powerful yet easy to learn Curtis VCL (Vehicle Control Language).

The Curtis 1310 Vehicle System Controller integrates and and expands systems through its industry standard CAN bus communication port. The Curtis 1310 seamlessly works in conjunction with Curtis CAN based SepEx and AC motor controllers such as the, 1243, 1244, 1234, 1236 and 1238.

Model 1310 can be applied to electric vehicles, non-electric vehicles or stationary control systems.

Fig. 1. *The Curtis 1310 Vehicle System Controller*

Features include

- ✔ The powerful, user-friendly programming language, VCL (Vehicle Control Language), developed by Curtis, allows custom software to be quickly and easily developed by an OEM for unique applications.
- ✔ CAN bus port allows customized vehicle systems and control.
- ✔ FLASH memory allows easy field upgrades and customization on the assembly line.
- ✔ CAN Open compatible communication protocol provides control and feedback to Curtis CAN-based Motor Controllers, as well as many other CAN based products.
- ✔ Extended Software functions of VCL simplify the integration of OEM requirements (BDI, Hour Meters, PID, RAMP, POT, CAN etc.).
- ✔ Comprehensive Input and Output Selection
- ✔ Two analog outputs (0 to 10 volts at up to 20 mA).
- ✔ Serial Port for communication with the Curtis Programmer or Curtis Model 840 "Spyglass" display.
- ✔ Two quadrature encoder inputs.
- ✔ Up to 22 Digital Switch Inputs and up to 16 Output Channels (up to 3 amps sink per channel) are available to a maximum input/output combination of 22 channels.
- ✔ Two proportional valve control outputs are available (16 output model only).
- ✔ Four software-configurable analog input channels available for any combination of 2- and 3-wire pot inputs or 0 to 5 volt inputs.
- ✔ Real-Time Clock with battery back-up (option).
- ✔ Built-in coil flyback diodes.
- ✔ Software and hardware watchdog circuits ensure proper software operation.
- ✔ Rugged aluminum housing

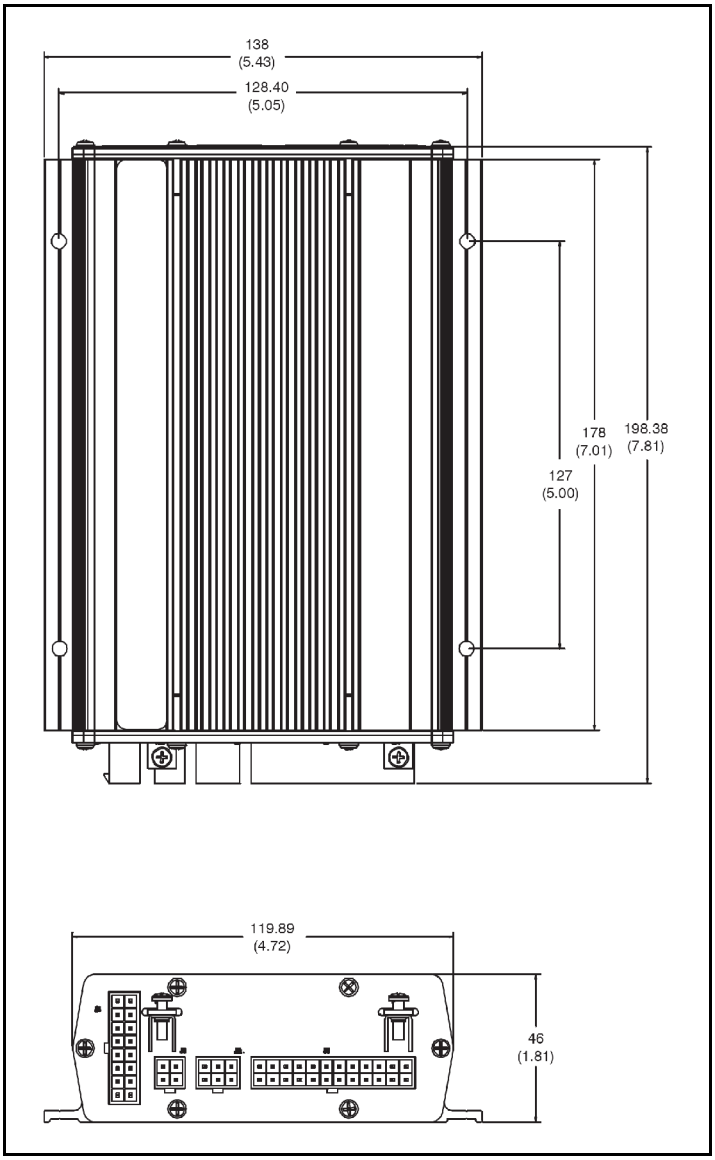
2

INSTALLATION AND WIRING

MOUNTING THE CONTROLLER

The Curtis 1310 mounts securely to the vehicle or system using 4 #8 or M4 screws. Care should be taken to prevent water from splashing or resting on the connector area. If possible, the connector area should be mounted downward and guarded from water and dust born contaminants which can degrade the electrical connections.

Fig 2. *Controller Mounting*



CAUTION

Working on electrical systems is potentially dangerous. You should protect yourself against uncontrolled operation, high current arcs, and outgassing from lead acid batteries:

UNCONTROLLED OPERATION — Some conditions could cause the system to run out of control. Disconnect motors, open valves and jack up the vehicle to get the drive wheels off the ground before attempting any work on vehicle control circuitry or software.

HIGH CURRENT ARCS — Batteries can supply very high power, and arcing can occur if they are short circuited. Always open the battery circuit before working on the system electrical circuit. Wear safety glasses, and use properly insulated tools to prevent shorts.

LEAD ACID BATTERIES — Charging or discharging generates hydrogen gas, which can build up in and around the batteries. Follow the battery manufacturer's safety recommendations. Wear safety glasses.

You will need to take steps during the design and development of your end product to ensure that its EMC performance complies with applicable regulations; see Appendix B for suggestions on managing EMC.

The Curtis 1310 Vehicle System Controller contains **ESD-sensitive components**. Use appropriate precautions in connecting, disconnecting, and handling the controller. See installation suggestions in Appendix B for protecting the controller from ESD damage.

HIGH CURRENT CONNECTIONS

The Curtis 1310 has several options from supply power to the controller. Since the 1310 Vehicle System Controller has many outputs, it is possible to draw a significant load from the battery. The B- high power tab must be used as the controller ground reference if more than 2 amps current is expected in the total system. Likewise, if the system could draw more than 2 amps of current from the B+, the B+ high power tab must be used to power the controller. If the driven loads are inductive, the load's power must be connected to B+ high power tab and the B+ high power tab must be connected to the battery (as shown in the standard wiring diagram).

When using the high power connections tabs, be careful not to bend or break the tab while tightening the bolt. For best results, use a pressure washer (convex side up) under the bolt head. This will help prevent the joint from loosening over time.

To help prevent overheating the joint, insure the that wire cable gage is sufficient to carry the continuous and maximum loads that will be seen by the 1310.

Table 1. High Current Connections

NAME	DESCRIPTION
B+	Battery positive connection tab Internally connected to J1-24. See table 1
B-	Battery Negative connection tab

LOW CURRENT CONNECTIONS

All low current (logic) connections are made through Molex Mini Fit Jr connectors. J1 is a 24 pin connector and contains most of the standard inputs and outputs. J2 is 6 pin connector dedicated to the CAN bus. J3 is a 4 pin connector dedicated to the Curtis serial bus port, used with the 1311 and 1314 programmers and the 840 Spyglass gage. J4 is a 16 pin connector for the analog input/outputs and encoder connections.

Low current wiring recommendations

Encoders

All four encoder wires should be bundled from the encoder to the controller connector. These can often be run with the rest of the low current wiring harness. The encoder cables should not be run near battery or motor cables. In applications where this is necessary, shielded cable should be used with the ground shield connected to the I/O ground at only the controller side. In extreme applications, common mode filters (e.g. Ferrite beads) could be used.

CAN bus

It is recommended that the CAN wires be run as a twisted pair. However, many applications at 125 kBaud are run without twisting, simply using two lines bundled in with the rest of the low current wiring. CAN wiring should be kept away from the high current cables and cross it at right angles when necessary.

All other low current wiring

The remaining low current wiring should be run according to standard practices. Running low current wiring next to the high current wiring should always be avoided.

Notes on the following tables

The proceeding tables are grouped by connector. They define the pin, signal name and basic function (description) of that signal. Often special VCL functions can be used to access or setup or use of these signals. The VCL Functions column notes these when appropriate. Each signal has a predetermined variable name or set of variable names that allow the VCL access to the value or control over the signal. These names are in the VCL References column

Partial Option Models

Model 1310-5210 is not "fully stuffed". This model has Outputs 9 through 16 and Inputs 1 though 13, 16, and 19 through 22 available. Outputs 14 and 15 have over 200kΩ output impedance.

Table 2. J1 Connections- Input/Outputs				
PIN	NAME	DESCRIPTION	RELATED VCL	
			FUNCTIONS	REFERENCES
1	Input/Output 1	A digital input with an open collector high frequency PWM output. This output also provides output current feedback. Signal is pulled to B- when output is on.	Put_PWM Automate_PWM Get_ADC	SW_1 SW_1_UP SW_1_Down PWM1 ADC15_Output
2	Input/Output 2	A digital input with an open collector high frequency PWM output. This output also provides output current feedback. Signal is pulled to B- when output is on.	Put_PWM Automate_PWM Get_ADC	SW_2 SW_2_UP SW_2_Down PWM2 ADC16_Output
3	Input/Output 3	A switch to B+ digital input with an open collector high frequency PWM output. Signal is pulled to B- when output is on.	Put_PWM Automate_PWM	SW_3 SW_3_UP SW_3_Down PWM3
4	Input/Output 4	A switch to B+ digital input with an open collector high frequency PWM output. Signal is pulled to B- when output is on.	Put_PWM Automate_PWM	SW_4 SW_4_UP SW_4_Down PWM4
5	Input/Output 5	A switch to B+ digital input with an open collector low frequency PWM output. Signal is pulled to B- when output is on.	Put_PWM Automate_PWM	SW_5 SW_5_UP SW_5_Down PWM5
6	Input/Output 6	A switch to B+ digital input with an open collector low frequency PWM output. Signal is pulled to B- when output is on.	Put_PWM Automate_PWM	SW_6 SW_6_UP SW_6_Down PWM6
7	Input/Output 7	A switch to B+ digital input with an open collector low frequency PWM output. Signal is pulled to B- when output is on.	Put_PWM Automate_PWM	SW_7 SW_7_UP SW_7_Down PWM7
8	Input/Output 8	A switch to B+ digital input with an open collector low frequency PWM output. Signal is pulled to B- when output is on.	Put_PWM Automate_PWM	SW_8 SW_8_UP SW_8_Down PWM8
9	Input/Output 9	A switch to B+ digital input with an open collector low frequency PWM output. Signal is pulled to B- when output is on.	Put_PWM Automate_PWM	SW_9 SW_9_UP SW_9_Down PWM9
10	Input/Output 10	A switch to B+ digital input with an open collector low frequency PWM output. Signal is pulled to B- when output is on.	Put_PWM Automate_PWM	SW_10 SW_10_UP SW_10_Down PWM10
11	Input/Output 11	A switch to B+ digital input with an open collector low frequency PWM output. Signal is pulled to B- when output is on.	Put_PWM Automate_PWM	SW_11 SW_11_UP SW_11_Down PWM11

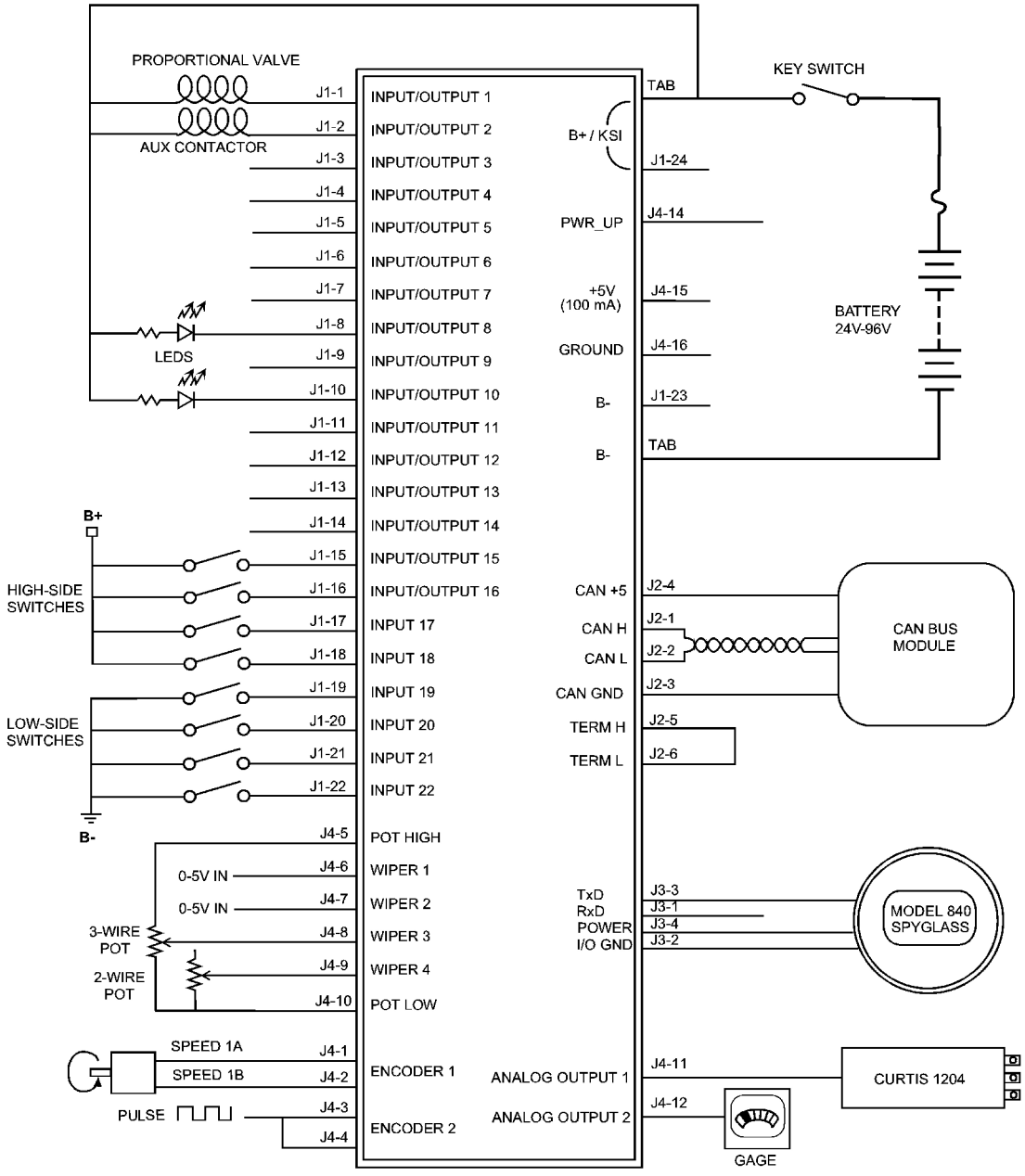
Table 2. J1 Connections- Input/Outputs				
12	Input/Output 12	A switch to B+ digital input with an open collector low frequency PWM output. Signal is pulled to B- when output is on.	Put_PWM Automate_PWM	SW_12 SW_12_UP SW_12_Down PWM12
13	Input/Output 13	A switch to B+ digital input with an open collector low frequency PWM output. Signal is pulled to B- when output is on.	Put_PWM Automate_PWM	SW_13 SW_13_UP SW_13_Down PWM13
14	Input/Output 14	A switch to B+ digital input with an open collector low frequency PWM output. Signal is pulled to B- when output is on.	Put_PWM Automate_PWM	SW_14 SW_14_UP SW_14_Down PWM14
15	Input/Output 15	A switch to B+ digital input with an open collector low frequency PWM output. Signal is pulled to B- when output is on.	Put_PWM Automate_PWM	SW_15 SW_15_UP SW_15_Down PWM15
16	Input/Output 16	A switch to B+ digital input with an open collector low frequency PWM output. Signal is pulled to B- when output is on.	Put_PWM Automate_PWM	SW_16 SW_16_UP SW_16_Down PWM16
17	Input 17	A switch to B+ digital input (pulled low to B-). Switch this pin to B+ to read as ON.		SW_17 SW_17_UP SW_17_Down
18	Input 18	A switch to B+ digital input (pulled low to B-). Switch this pin to B+ to read as ON.		SW_18 SW_18_UP SW_18_Down
19	Input 19	A switch to ground digital input (pulled high to +15v). Switch this pin to B- to read as OFF.		SW_19 SW_19_UP SW_19_Down
20	Input 20	A switch to ground digital input (pulled high to +15v). Switch this pin to B- to read as OFF.		SW_20 SW_20_UP SW_20_Down
21	Input 21	A switch to ground digital input (pulled high to +15v). Switch this pin to B- to read as OFF.		SW_21 SW_21_UP SW_21_Down
22	Input 22	A switch to ground digital input (pulled high to +15v). Switch this pin to B- to read as OFF.		SW_22 SW_22_UP SW_22_Down
23	B-	Can be used as a low power (<2 amp) ground reference or for the switch inputs 12 – 22 B- reference.		
24	B+	Can be used to power the system (<2 amps) or for B+ reference for switches, etc.	Setup_BDI	ADC13_Output KSI_Filtered KSI_Raw

Table 3. J2 Connections- CAN Bus				
PIN	NAME	DESCRIPTION	RELATED VCL	
			FUNCTIONS	REFERENCES
1	CAN Hi	Positive CAN Bus rail	Setup_CAN Setup_Mailbox Send_Mailbox etc...	
2	CAN Lo	Negative CAN bus rail	Setup_CAN Setup_Mailbox Send_Mailbox etc...	
3	GND	Ground reference		
4	+5V	+5v for remote module(s)		
5	Term H	Connect Term H to Term L to create an end-of-bus termination (adds a 120Ω resistor across CAN Hi and CAN Lo)		
6	Term L	See above		

Table 4. J3 Connections- Serial Port				
PIN	NAME	DESCRIPTION	RELATED VCL	
			FUNCTIONS	REFERENCES
1	RxD	Serial receive line for programmer and spyglass communications	Setup_Serial Put_Spy_Message	
2	GND	Communications ground		
3	TxD	Serial Transmit line for programmer and spyglass communications	Setup_Serial Put_Spy_Message	
4	PWR	+12v Power The output current of this pin and +5volts (J4-15) is combined and monitored at ADC12 .		ADC12_Output

Table 5. J4 Connections- Specialty I/O				
PIN	NAME	DESCRIPTION	RELATED VCL	
			FUNCTIONS	REFERENCES
1	Encoder 1A	Pulse count input or Channel A for encoders	Setup_Encoder Get_Enc_Count Get_Enc_Vel Get_Enc_Dir Get_Enc_Error	ENC1 ENC1_Count ENC1_Dir ENC1_Vel ENC_Error SW_23
2	Encoder 1B	Channel B for encoders		SW_24
3	Encoder 2A	Pulse count input or Channel A for encoders	Setup_Encoder Get_Enc_Count Get_Enc_Vel Get_Enc_Dir Get_Enc_Error	ENC2 ENC2_Count ENC2_Dir ENC2_Vel ENC_Error SW_25
4	Encoder 2B	Channel B for encoders		SW_26
5	Pot High	The high voltage reference for the 4 potentiometer inputs	Get_Pot Get_ADC	POT_High ADC1 ADC1_Output
6	Wiper 1	A generic 0-5 volt input which can also be setup for as a potentiometer wiper input	Get_Pot Setup_Pot Setup_Pot_Filtered Get_ADC	POT1_Output ADC2_Output
7	Wiper 2	A generic 0-5 volt input which can also be setup for as a potentiometer wiper input	Get_Pot Setup_Pot Setup_Pot_Filtered Get_ADC	POT2_Output ADC3_Output
8	Wiper 3	A generic 0-5 volt input which can also be setup for as a potentiometer wiper input	Get_Pot Setup_Pot Setup_Pot_Filtered Get_ADC	POT3_Output ADC4_Output
9	Wiper 4	A generic 0-5 volt input which can also be setup for as a potentiometer wiper input	Get_Pot Setup_Pot Setup_Pot_Filtered Get_ADC	POT4_Output ADC5_Output
10	Pot Low	Low voltage reference for the 4 potentiometer inputs	Get_Pot Get_ADC	POT_Low ADC6_Output
11	Analog Output 1	0-10 volt analog output	Put_DAC Automate_DAC	DAC1
2	Analog Output 2	0-10 volt analog output	Put_DAC Automate_DAC	DAC2
13	Not Used			
14	PWR_UP	B+ input can be used to power up the 1310.	Get_ADC	ADC7_Output
15	+5V	+5 volts to power sensors. Can supply up to 200ma. The output current is monitored at ADC11.	Get_ADC	ADC11_Output
16	GND	Ground reference		

CONTROLLER WIRING: BASIC CONFIGURATION



This wiring diagram shown is a generalized diagram. Its purpose is to show a variety of basic uses for the various 1310 Inputs and Outputs. It also provides a standard (although not the only) power and battery connections. The following paragraphs walk though the diagram.

Power Connection

The Battery is connected to to 1310 power tabs though a fuse and a key switch. The power tabs are used because there are inductive loads on the system (Aux Contactor and Proportional Value coils) and the current could exceed 3amps. The fuse is required to protect the wiring as the 1310 could draw significant power if there were a short or failure in the unit.

The key switch is used to "start" the system. Both the B+ High Power Tab and the B+ signal at J1-24 are used as the Key Switch Input. When the key switch is closed, the B+/ KSI input goes high, the 1310 power supply brings up the 1310 and the BDI functions are enabled.

Outputs

The system shown has 2 high power outputs and 2 LEDs, that run of key switch power.

Using the PWM outputs to drive the LEDs allows the the brightness of the LEDs to be varied. The frequency is too high for the human eye to see any flickering. Note that a dropping resistor must be used because even low duty cycle PWM applies full battery voltage is short bursts, and this will destroy the LED without a dropping resistor limiting the current. Note the internal impedance to ground of the driver will cause leakage current to flow through the LEDs even when the output driver is off. Refer to Digital Output Specifications following when calculating this leakage current. This leakage current can be enough (> 2 ma) to light high efficiency LEDs. Model 1310-5210 provides two output drivers (Outputs 14 and 15) that do not have leakage current issues and may therefore be the best suited for driving LEDs.

The first power output drives a proportional valve coil. Outputs 1 and 2 are special in that they have internal current feedback lines. VCL can use this signal in a PID loop to regulate current, which is necessary to properly control the position (and flow) in a proportional valve. Outputs 1 through 4 also run at a higher frequency and thus can provide a smoother current (less ripple).

The second power output drives a basic contactor coil. It is connected to output 2, which has a current feedback signal. In this case, the VCL can use the current feedback signal to ensure that the coil is connected and drawing the proper current when on. In this way, enhanced fault diagnostic can be performed.

Switch Inputs

All of the Outputs can be used as active high inputs ("on" when connected to B+). There are 4 special inputs that are active low ("on" when connected to B-). If an Output is being used as an input (such as is the case on Input/Output 15) the VCL must take care not to turn on that output or a direct short to B+ could be established through the switch and the internal FET driver.

Analog Inputs

Three types of analog inputs are used. The first two inputs use a 0-5 volt input. The next is a 3-wire connection for a potentiometer using both Pot High and Pot Low and the third is a 2-wire potentiometer or rheostat.

Note that in all cases, the VCL code must be written to provide the necessary wiring and potentiometer fault checking. To accomplish this, the 1310 provides the measured voltage readings of Pot High and Pot Low connections. Monitoring these values will can indicate if there is a shorted to B+ or B-. Using the Pot High and Pot Low connections for the potentiometer or rheostat will also provide a small lower and (when using Pot High) upper bound to the analog input. Knowing this, proper range checking in VCL can be performed for additional fault diagnostics.

Encoder and Pulse Inputs.

The 1310 has 2 quadrature encoder inputs. Using A and B channels with a quadrature encoder allows velocity, position (count) and direction detection. Tyeing the A and B channels together, as shown on encoder input 2, allows the input to measure a single pulse train. In this configuration, the 1310 will count up (ENC_Count mode) or measure speed magnitude (Enc_Velocity mode). In both cases, the ENC2_DIR variable is not valid.

Power for the encoder can be derived form the +5 Volt output and Ground pins found on J4. The +5 volt output has an output current measurement. VCL can use this value to determine if the encoder and/or any other sensors are connected and drawing the proper current. This can be used to provide additional fault diagnostics.

Note: If the encoder inputs are setup in velocity mode, the direction flag will not be accurate below a low speed threshold. The direction bit may stay in the last direction and may not return to 0 when the speed is at zero. The VCL code must be written to read the velocity variable and double check the direction bit in this case.

Analog Outputs

The 2 analog outputs can be used to interface to analog throttle input motor controllers or other devices, Here, Analog Output 1 is used to control the 1204 Motor Controller throttle input. Note that most throttle inputs are 0-5 volts while the 1310 can provide up to 10 volts.

Analog Output 2 is being used to drive a Curtis gage (enGage 2 or simple voltmeter). VCL code can use this output to display a wide range of data, from the state of the battery charge, potion of the potentiometer wipers or speed of the encoder.

INPUT/OUTPUT SIGNAL SPECIFICATIONS

The input/output signals wired to the J1 through J4 connectors can be grouped by type as follows; their electrical characteristics are discussed below.

- Digital inputs
- Digital outputs
- Analog inputs
- Analog outputs
- Power
- Communications ports

Digital Inputs

These signal lines can be used as digital (ON/OFF) inputs. Normally, the ON signal is made by a connection direct to B+ and OFF is direct to B-. Inputs 1 through 18 will pull low (OFF) if no connection is made. Inputs 19 through 26 will pull high (ON) if no connection is made.

Inputs 1 through 18 are associated with driver outputs. Inputs 19 through 26 are low voltage "TTL" level inputs and can be used when connecting to other low voltage (5v) logic circuits or sensors. The encoder channels are normally used for pulse count inputs from quadrature (2 channel) encoders, but they may also be used as 5v logic level digital inputs. Take careful note of their much lower voltage range.

DIGITAL INPUT SPECIFICATIONS					
SIGNAL NAME	PIN	Logic Thresholds	Input Impedance	Voltage Range	ESD Tolerance
Input/Output 1-16	J1-1 thru 22	*24-48V models:* Low = 7.5v High = 15.8v *48-96V models:* Low = 14.1v High = 29.7v	*24-48V models:* about 5.4 kΩ *48-96V models:* about 22kΩ *see note below	*24-48V models:* -0.5 to 64v *48-96V models:* -0.5 to 124v	± 8 kV (air discharge)
Input 17-18	J1-17 thru 18	*24-48V models:* Low = 7.5v High = 15.8v *48-96V models:* Low = 14.1v High = 29.7v	*24-48V models:* about 5.4 kΩ *48-96V models:* about 22kΩ	*24-48V models:* -0.5 to 64v *48-96V models:* -0.5 to 124v	± 8 kV (air discharge)
Input 19-22	J1-19 thru 22	*All models* High = 3.8v Low = 1.8v	*All models* about 4.5kΩ	*24-48V models:* -0.5 to 64v *48-96V models:* -0.5 to 124v	± 8 kV (air discharge)
Encoder Inputs Inputs 23-26	J4-1 thru 4	Rising edge = 3.0v Falling Edge = 2.0v	*All models* about 4.7kΩ	*All models* -0.5 to 5.5 v	± 8 kV (air discharge)

Tolerance of above values; ±5%

* Outputs 14 and 15 on Model 1310-5210 have over 200kΩ output impedance

Digital Outputs

These signal lines can be used as digital (ON/OFF) or Pulse Width Modulated (PWM) outputs. Each driver is active low, meaning the output will pull low (to B-) when commanded ON. The PWM is at a fixed frequency (9.7kHz for Outputs 1-4 and 400Hz for Outputs 5-16) but can vary duty cycle from 0% (off = 0) to 100% (on = 32767). Digital Outputs 1 and 2 are special as these have current feedback signals (internal) that can be used by VCL to create current sources or check the output load etc.

If the 1310 Digital outputs are connected to inductive loads, the B+ tab must be connected to the battery source. This connection provides a path for the internal freewheel diodes to clamp the turn-off spike. Failure to make this connection with inductive loads can cause permanent damage to the Curtis 1310 as well as propagate failures of other electronics in the system due to the high voltage spike caused when an inductive load turns off without a freewheel path.

DIGITAL OUTPUT SPECIFICATIONS					
SIGNAL NAME	PIN	PWM & Frequency	Output Current	Protected Range	ESD Tolerance
Input/Output 1-16	J1-1 thru 22	0-100% at Outputs 1-4 ~9.7Khz Outputs 5-16 ~400 Hz	Sink 3amps	*24-48V models:* -0.5 to 64v *48-96V models:* -0.5 to 124v	± 8 kV (air discharge)

Analog Inputs

The 1310 provide four analog inputs. These inputs can easily be configured for use with potentiometers. VCL allows each input to be independently set up as a voltage input or as a 2-wire or 3-wire resistance input. Voltage inputs can be connected directly to the Wiper input (with B_ or GND for the return line). Rheostats (2-wire) are connected between the Pot Wiper and Pot Low and a 3-wire potentiometer has the resistance element connected between the Pot High and Pot Low signals and the wiper connected to the Wiper signal. The corresponding VCL setup must be used to allow the 1310 to properly detect and scale the signal.

Although designed to be used with potentiometers, Pot High and Pot Low signals are monitored by analog pins in the 1310 and thus have a limited use as analog inputs. Note that these pins have a low input impedance (~680Ω) which could be damaged by moderate voltages from a low impedance source.

ANALOG INPUT SPECIFICATIONS					
SIGNAL NAME	PIN	Operating Voltage	Input Impedance	Protected Voltage	ESD Tolerance
Wiper 1-4	J4-6 thru 9	0 - 5v	5kΩ	-1v to 30v	± 8 kV (air discharge)
Pot High	J4-5	0 - 5v	680Ω to +5V	-1v to 12v	± 8 kV (air discharge)
Pot Low	J4-10	0 - 5v	680Ω to ground	-1v to 8v	± 8 kV (air discharge)

Analog Outputs

Two signals provide low power analog outputs. These outputs are generated from filtered PWM signals and have about 1% ripple. The settling time (within 2% of final output) is about 30ms for a 0–10V step. The Analog Outputs are protected against shorts to B+ or B-.

ANALOG OUTPUT SPECIFICATIONS					
SIGNAL NAME	PIN	Output Voltage	Output Impedance	Protected Voltage	ESD Tolerance
Analog Output 1-2	J4-11 & 12	0-10v	about 33 kΩ	*-1v to 124v*	± 8 kV (air discharge)

CAUTION Important Note: During a software download (FLASH), the DAC output voltages will float up and can reach as high as 10volts. Make sure that your vehicle system is safe and can tolerate this event.

Note: If the battery system droops below 20volts, the DAC outputs will not reach the 10 volt output specification and will start dropping as the battery voltage drops below 20volts.

Power

These signals provide power for the various sensors and communication system that might be connected to the 1310. The PWR signal is normally only used for powering the 1311 handheld programmer or the 840 Spyglass gage, but can be used to power other small sensors or electronics. These three power supply signals are current limited. The limited supply current can be split between these power pins as long as the total does not exceed 200ma.

POWER SPECIFICATIONS					
SIGNAL NAME	PIN	Output Voltage	Output Current	Protected Voltage	ESD Tolerance
+5 Volts	J4-15	5V ± 5%	200ma max*		± 8 kV (air discharge)
CAN +5v	J2-4	5V ± 5%	200ma Max*		± 8 kV (air discharge)
PWR (Serial Port)	J3-4	13.75V± 5%	200ma Max*		± 8 kV (air discharge)

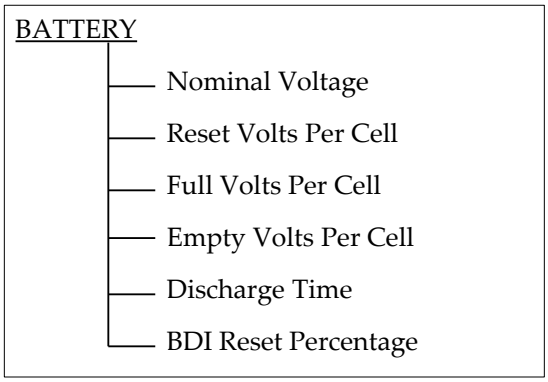
* combined current of +5Volts, CAN +5V and PWR can not exceed 200ma.

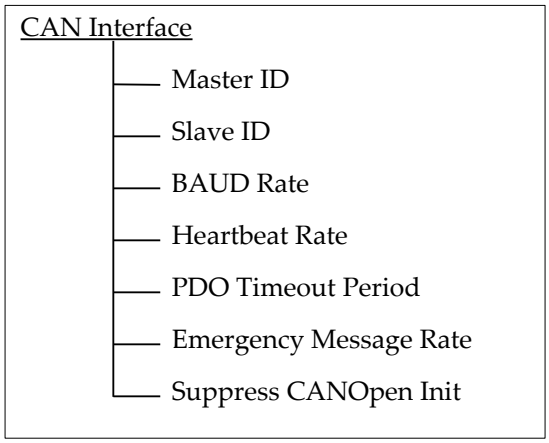
3 PROGRAMMABLE PARAMETERS

The Curtis 1310 Vehicle System Controller is designed a universal programmable control block and has only a few standard parameters that can be programmed using a Curtis 1311 handheld programmer. Using VCL, many custom parameters and menus can be added to meet the needs of the application. Refer to section 5 of the VCL Programmers Guide for detailed information on setting up parameter lists and menus that can be read by the Curtis 1311 handheld or Curtis 1314 PC based programmers. For basic information on Curtis 1311 handheld programmer operation, see Appendix C.

PROGRAM MENU

The following sub menus are found under the PROGRAM Menu;





Individual parameters are presented as follows in the menu charts:

Parameter name as it appears in the programmer display ⇓	Allow able range in the programmer's units ⇓	Description of the parameter's function and, w here applicable, suggestions for setting it ⇓
Nominal Voltage *Nominal_Voltage*	**0 - 48 volts** *0 - 3072*	Defines the nominal battery pack voltage
⇑ *Parameter name in VCL*	⇑ *range in VCL units*	

Battery Discharge Indicator algorithm

The 1310 contains a sophisticated battery state of charge algorithm. Setup properly, this algorithm can track the remaining battery charge (in percent) using only a voltage reading from the B+ Power tab (or J1-24). To achieve any accuracy, it is critical to set the BDI parameters correctly for the vehicle, battery and normal duty cycle of the application. Note that many of the parameters are in volts per cell. A normal 24 volt battery has 12 cells.

The remaining battery capacity is automatically updated into the value named;

BDI_Percentage

There are 6 parameters that set up the tracking algorithm;

BDI PARAMETERS		
Nominal Voltage *Nominal_Voltage*	24 – 48v 1536 - 3072	Must be set to the systems nominal battery voltage. This is used by the 1310 to determine the number of cells in the pack for the BDI parameters below.
BDI Reset Volts Per Cell *BDI_Reset_Volts_Per_Cell*	0.00 – 3.00v 0 - 3000	At power on, The BDI is reset to 100% if the battery is measured above this setting. Typical value is 2.09v. Reset Volts must be set above Full Volts.
BDI Full Volts Per Cell *BDI_Full_Volts_Per_Cell*	0.00 – 3.00v 0 - 3000	The BDI will output a 100%, full, while the average voltage per cell reading is above this setting.
BDI Empty Volts Per Cell *BDI_Empty_Volts_Per_Cell*	0.00 – 3.00v 0 - 3000	The BDI will output a 0%, empty, when the average voltage per cell reading is below this setting.
BDI Discharge Time *BDI_Discharge_Time*	0 – 600 min 0 - 600	The time it will take the BDI to go from 100% to 0% at maximum current draw.
BDI Reset Percent *BDI_Reset_Percent*	0 - 100% 0 - 100	If the previous BDI % value was above this point at power up, the BDI will NOT reset, even if the battery is measured above the Reset Volts level. This will prevent a slightly discharged battery from "floating" and reseting the BDI at every power up.

CAN Open Interface

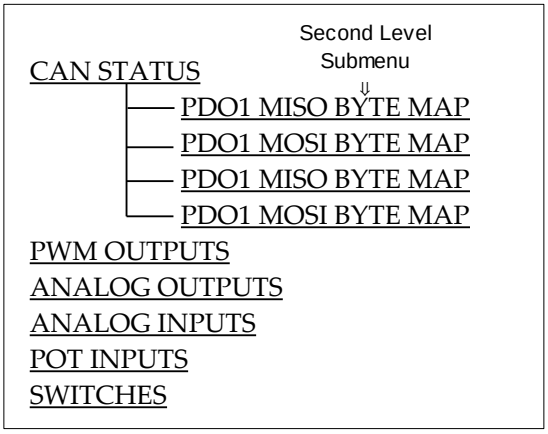
The 1310 can be easily interfaced to other CAN Open modules. These parameters work with VCL to setup the basic CAN Open IDs and rates. Refer to the VCL Common Functions Manual section G on setting up CAN Open PDO, SDO and other CAN related functions.

CAN Open Interface		
Master ID *CAN_Master_ID*	0 - 3 0 - 3	CAN ID for incoming messages to a CANOpen slave system
Slave ID *CAN_Slave_ID*	0 - 31 0 - 31	CAN ID for outgoing messages from a CANOPen slave system
BAUD Rate *CAN_BAUD_Rate*	0 - 2 0 - 2	Sets the CAN BAUD rate 0 = 125Kbps, 1 = 250Kbps 2 = 500kbps
Heartbeat Rate *Heartbeat_Rate*	16 – 200 mS 4 - 50	Sets the rate at which the CANOpen Heatbeat messages are sent by the system
PDO Timeout Period *PDO_Timeout_Period*	0 – 200 ms 0 - 50	Sets the PDO timeout period for the CANopen Slave system. After the slave controller has sent a PDO MISO, it will declare a PDO Timeout Fault if the master controller has not sent a reply PDO MOSI message within theset time. Either PDO1 MOSI or PDO2 MOSI will reset the timer. Setting the PDO Timeout Period = 0 will disable this fault check.
Emergency Message Rate *Emergency_Message_Rate*	16 – 200 mS 4 - 50	Sets the minimum rate between CAN emergency messages from the CANopen Slave system. This prevents quickly changing fault states from generating so many emergency messages that they flood the CAN bus.
Suppress CANOpen Init *Suppress_CANOpen_Init*	0 - 1 0 - 1	When Suppress CANopen Init is set = 1, at Power On the initialization of the CANopen system is suppressed. Typically this is done so that the VCL program can make changes to the CANopen system before enabling it (by setting the variable Suppress_CANopen_Init = 0 and running the Setup_CAN() function)

4 INTERNAL DATA

MONITOR MENU

The 1311 Handheld or 1314 PC programmer provides access to many interal variables that are continuously read and updated. The values are displayed under the MONITOR menu. The variables are further organized into submenus as depicted below;



PWM OUTPUTS SUB MENU		
DISPLAY	VARIABLE RANGE	DESCRIPTION
Channel # *PWM#_Output*	0–32767 *0–32767*	PWM output value (32767 = 100%) of one of the 16 PWM signals. Where # = 1 through 16

ANALOG OUTPUTS SUB MENU		
DISPLAY	VARIABLE RANGE	DESCRIPTION
Channel # *DAC#_Output*	0–32767 *0–32767*	Analog output value (32767 = 10v) of one of the 2 DAC channels. Where # = 1 or 2

ANALOG INPUTS SUB MENU		
DISPLAY	VARIABLE RANGE	DESCRIPTION
KSI Filtered *KSI Filterd*	0–100.0 v *0–10000*	Filtered and calibrated value of the B+ / KSI input signals. 1 volt = 100 counts
KSI Raw *KSI_Raw*	0–1023 *0–1023*	Unfiltered (raw) value of the B+ / KSI input signals ~ 1 volt = 9.5 counts (uncalibrated)
Channel # *ADC#_Input*	0–1023 *0–1023*	Analog input value (~1023 = 5v) of one of the 16 ADC channels. Where # = 1 through 16

POT INPUTS SUB MENU		
DISPLAY	VARIABLE RANGE	DESCRIPTION
POT # *POT#_Input*	0–32767 *0–32767*	Analog value of the Pot Wiper input signal, where # = 1 through 4.
POT High *Pot_High*	0–32767 *0–32767*	Analog reading of the Pot High signal.
Pot Low *Pot_Low*	0–32767 *0–32767*	Analog reading of the Pot Low signal.

SWITCHES SUB MENU		
DISPLAY	VARIABLE RANGE	DESCRIPTION
Switch # *SW_#*	On/Off *0–255*	Switch state, where # = 1 through 26. where # 1 - 22 are the 22 inputs (J1-1 through 22) and 23-26 are the 4 encoder channels (J4-1 through 4)

CAN STATUS SUB MENU		
DISPLAY	VARIABLE RANGE	DESCRIPTION
CAN NMT State *CAN_NMT_State*	0 - 127	CAN Network state. 0=initialization, 4=stopped, 5=operational, 127=pre-operational.
PDO 1 MISO COB ID	0 - 65355	Communication Object ID for the PDO 1 Master In Slave Out message.
PDO 1 MISO Byte Map	*Sub Menu**	Submenu for the PDO 1 MISO data. *Byte Map submenu has 8 data bytes, which show the current data on the CAN bus for this PDO
PDO 1 MOSI COB ID	0 - 65355	Communication Object ID for the PDO 1 Master Out Slave In message.
PDO 1 MOSI Byte Map	*Sub Menu**	Submenu for the PDO 1 MOSI data. *Byte Map submenu has 8 data bytes, which show the current data on the CAN bus for this PDO
PDO 2 MISO COB ID	0 - 65355	Communication Object ID for the PDO 2 Master In Slave Out message.
PDO 2 MISO Byte Map	*Sub Menu**	Submenu for the PDO 2 MISO data. *Byte Map submenu has 8 data bytes, which show the current data on the CAN bus for this PDO
PDO 2 MOSI COB ID	0 - 65355	Communication Object ID for the PDO 2 Master Out Slave In message.
PDO 1 MOSI Byte Map	*Sub Menu**	Submenu for the PDO 2 MOSI data. *Byte Map submenu has 8 data bytes, which show the current data on the CAN bus for this PDO

CONTROLLER INFORMATION MENU

This menu provides ID and version numbers for your controller hardware and software.

CONTROLLER INFORMATION MENU		
DISPLAY	VARIABLE RANGE	DESCRIPTION
Model Number *Model_Number*	0–4294967295 *0–4294967295*	Model Number. For example, if you have a controller with the model number 1310-4501, the *Model_Number* variable will have a value of 13104501.
Serial Number *Serial_Number*	0–4294967295 *0–4294967295*	Serial Number. For example, if the serial number printed on your controller is 05045L.11493, the *Serial_Number* variable will have the value of 11493.
Mfg Date Code *Manuf_Date*	0–32767 *0–32767*	Controller date of manufacture, with the last three digits being reserved for the day. For example, if the serial number printed on your controller is 05045L.11493, the *Manuf_Date* variable will have the value of 5045 (45th day of 2005).
Hardware Version *Hardware_Ver*	0–32.767 *0–32767*	The hardware version number uniquely describes the version of electronic assemblies used in the controller.
OS Version *OS_Ver*	0–32767 *0–32767*	Version number of the operating system software that is loaded into the controller. This variable specifies the major version number of the controller's operating system
Build Number *Build_Number*	0–32767 *0–32767*	Build number of the operating system software that is loaded into the controller. This variable specifies the minor version number of the controller's operating system.
SM Version *SM_Ver*	0–327.67 *0–32767*	Version number of the Start Manager software that is loaded into the controller.
Param Block Version *Param_Blk_Ver*	0–327.67 *0–32767*	Version number of the parameter block that is loaded into the controller.
VCL App Version *VCL_App_Ver*	0–327.67 *0–32767*	Version number of the VCL application software that is loaded into the controller. This value is set in the VCL program by assigning a value to the *VCL_App_Ver* variable.

6 VEHICLE CONTROL LANGUAGE

INTRODUCTION

Curtis 1310 Vehicle System Controller is similar to a programmable logic controllers with application specific functions generally found in the vehicle control industry. Key to the flexibility and application of the Curtis 1310 is a proprietary software language; VCL (Vehicle Control Language). VCL software provides a fast and easy way to implement unique and complex vehicle control functions.

VCL is a programming language that will feel very familiar to anyone who has worked with BASIC, Pascal, or C. VCL code is written on any code writing program (such as Code Warrior or UltraEdit) or non-formating text program (such as Windows Notepad). VCL software is complied, managed and downloaded into the 1310 using the Curtis developed WinVCL PC program.

The common portions of the VCL programming language are described in the VCL Programmer's Guide and the VCL Common Functions Manual. These two manuals include more detailed information about VCL than is included here and should be your starting point if you are not yet familiar with VCL. A third manual, the WINVCL Users Guide should also be reviewed prior to starting your VCL programming.

This section describes aspects and functions of VCL that are unique to the 1310 product, and provides a basic summary overview of VCL.

Summary of VCL Basics

- VCL is not case-sensitive:
  put_pwm(), Put_PWM(), and PUT_PWM() are identical.
- Spaces in variable and constant names are not allowed in VCL;
  use underscores in place of spaces. For example: SW_1 is the VCL name for the switch input #1
- Functions are followed by parentheses;
  for example:GET_ADC() is a function. ADC7_Output is a variable.
- Logical statements must be inside parentheses;
  examples: IF (setpoint >50)
  ELSE IF ((setpoint <20) & (temperature >100)).
- Comments are preceded by semicolons.
  example: **;** This is a comment

VARIABLE TYPES & QUANTITIES

VCL provides dedicated space in which to store custom variables. There are four types of variables, based on their type of storage:

- Volatile Memory (RAM)
- Automatic Non-Volatile memory (EEPROM)
- Non-Volatile Block Memory (EEPROM)
- Parameter Non-Volatile Memory (EEPROM)

Volatile Memory (RAM) variables are stored only while power is on; they are lost at power down. This is the typical form of memory used to hold temporary calculations, counters and other variable that are only needed while running. The generic VCL name for these variables are User1 – 120. They can be initialized on power-up by explicit VCL assignments (i.e., User1 = 12) otherwise they will be reset to a value of 0.

Automatic Non-Volatile memory (EEPROM) are labeled NVUser1–15 in VCL. These 15 variables are stored cyclically while running and at power-down. They can be recalled by using the VCL NVM_NVUser_Restore function. Thus, they are automatically saved and can then be recalled at the next power-on cycle to restore their previous values. See the section on non-volatile memory access in the VCL Common Functions manual for more information.

Non-Volatile Block Memory (EEPROM) are 38 blocks of 15 variables (total of 570 variables), which are stored and recalled using the functions NVM_Block_Read and NVM_Block_Write. The 38 blocks are called NVM3–NVM40. The read and write functions will retrieve and store RAM (such as User20) variable(s) from and into the EEPROM blocks. See the section on non-volatile memory access in the VCL Common Functions manual for more information.

Parameter Non-Volatile Memory (EEPROM) variables are a special type of EEPROM variable that are only to be used to create OEM defined 1311 parameters. These 1311 parameters can be defined as a 16-bit word by using the P_User variables or they can be defined as single bit by using the P_UserBit variables. These value of these variables are modified and stored by using the the 1311 or 1314 programmer interface (i.e., when a 1311 user changes a parameter setting using the 1311). They can be read or written to in the VCL code. <u>It is very important to note that writing to parameters in VCL will not be stored in EEPROM</u> or read by the 1311 or 1314 programmer. At the next power down, the data change made by VCL will be lost.

TYPE	QUANTITY	RANGE
RAM	120 variables	User1 – User120
NVUser EEPROM	15 variables	NVUser1 – NVUser15
Block EEPROM	38 blocks	NVM3 – NVM40 (15 variables each)
Parameters EEPROM	100 16 bit variables 160 single bit variables	P_User1 – P_User100 P_UserBit1 – P_UserBit10

VCL RUNTIME RATES

VCL is an interpreted language. Each line of VCL code is converted into an array of pseudo-code by the WINCL complier which can then be flash loaded into the controller. The controller interprets these pseudo-codes one line at a time while the system is running (powered). The table below lists the rate that the VCL interpreter runs each of the various functions (Service Rate) and lists how many of each function is available to the VCL software (Instances):

FUNCTION	FUNCTION FULL NAME	INSTANCES	SERVICE RATE
ABS	Absolute Value	2	4 ms
ADC	Analog to Digital Converter Input	2	1 ms
CAN	CAN Communications	2	4 ms
CPY	Copy	8	4 ms
DAC	Digital to Analog Converter Output	2	4 ms
DLY	Delay	16	1 ms
FLT	Filter	4	1 ms
LIM	Limit	4	4 ms
MAP	Map	4	4 ms
MTD	Multiply then Divide	4	4 ms
NVM	Non-Volatile Memory	38	2 ms
PID	Proportional Integral Derivative	2	4 ms
POT	Potentiometer Input	4	4 ms
PWM	Pulse Width Modulated Output	16	4 ms
RMP	Ramp	4	1 ms
SCL	Scaling	4	4 ms
SEL	Selector 2-position switch	8	4 ms
SEL_4P	Selector 4-position switch	8	32 ms
SW	Switch Inputs (all)	1	4 ms
RTC	Real Time Clock	1	4 ms
TMR	Timers (hourmeters)	3	1 ms

SPECIFIC VCL FUNCTIONS & EXTENSIONS

The VCL functions described in the VCL Common Functions Manual are available on 1310 controllers. In addition, the 1310 Vehicle System Controller has the following additional or expanded functions:

- Pot Wiper Inputs
- Analog Inputs
- Analog (DAC) outputs
- Digital (PWM) outputs
- Encoder Inputs
- Real Time Clock

Pot Wiper Inputs

Setup_POT(2)

This function sets the type of input that will be connected to the 4 analog inputs. The system uses the illustration of a potentiometer and thus are called Wiper inputs. The inputs can be set up as one-wire, two-wire or three-wire, referring to the number of wires connected from the pot to the 1310. A 1 wire pot, is a voltage input (0-5v), a 2 wire pot uses wiper and pot low (rheostat) and a 3-wire pot connects with Pot High, Wiper and Pot Low signals. Note this functions does not enable any fault checks. Fault checks must be done by custom VCL code.

Data Values

Pot#_Output Variable that is automatically updated with the value of the wiper input.

Parameters

Pot_ID Identifies which wiper input is used
- Can be POT1 through POT4

Type Identifies which type of pot is connected
- Can be ONE_WIRE, TWO_WIRE or THREE_WIRE

Returns

0 – Setup did not execute.
1 – Setup successful.

Error Codes

Bad_ID = incorrect pot ID was used.
Param_Range = the Type value is not within range.

Examples

```
Setup_Pot(POT1,TWO_WIRE)
        ; refer to wiring diagram pins J4-6 and J4-7
Setup_Pot(POT3, THREE_WIRE)
        ; refer to wiring diagram pins J4-8 , J4-5, J4-10
Setup_Pot(POT4, TWO_WIRE)
        ; refer to wiring diagram pins J4-9, J4-10
```

Analog Inputs

Get_ADC(1)

This function retrieves the present input value of the selected ADC (Analog to Digital Converter) channel. Though there are only 4 dedicated analog inputs (called Wiper 1- 4), the 1310 actually monitors 12 channels of analog values. Many of these analog signals are internal to the 1310 hardware but can be used in VCL. It is NOT necessary to use the Get_ADC() function, as the ADC channels are constantly monitored and automatically placed in the corresponding ADC#_Output variable, but the VCL programmer my wish to use this function to insure that the most recent value is read or for clarity in the program. Note that all ADC channels are read as 10 bit and therefore have a range of 0 to 1023.

Data Values

ADC#_Output Variable that is updated with the value of the ADC channel.

Parameters

ADC# Identifies which ADC channel is to be read
Can be ADC1 through ADC16

- ADC1 = Common Pot High
- ADC2 = Pot 1 wiper input
- ADC3 = Pot 2 wiper input
- ADC4 = Pot 3 wiper input
- ADC5 = Pot 4 wiper input
- ADC6 = Common Pot Low
- ADC7 = Pwr_Up Input (~9.5 counts/volt)
- ADC8, 9 & 10 = Not connected
- ADC11 = +5 volt output current monitor
- ADC12 = +5 & +12 volt combined current (~4.21 counts/ma)
- ADC13 = B+/ KSI Input (~9.5 counts/volt)**
- ADC14 = Not Connected
- ADC15 = PWM 1 drive current (~310 counts/amp)
- ADC16 = PWM 2 drive current (~310 counts/amp)

Returns

0 to 1023 = the value of ADC channel

Error Codes

Bad_ID = ADC channel is out of range (>16)

Examples

```
User1 = Get_ADC(ADC11)  ; Put the current reading of +5v into User1
```

** It is recommended to use the variables KSI_Filtered (instead of ADC_13) since it is factory calibrated for 100 counts per volt. ADC_13 is uncalibrated.

Analog Outputs

Put_DAC(2)

This function outputs an analog voltage on selected DAC (Digital to Analog Converter) channel. A constant or a variable may be used as the output Value.

Parameters

DAC# Identifies which DAC channel is to be read

- can be DAC1 or DAC2

Value The digital value of the output voltage

- The scale is 0-32767 = 0.0 to 10.0 volts.

Returns

0 – new value did not go out.
1 – new value output on DAC.

Error Codes

Bad_ID = incorrect DAC ID was used.
Auto_Run = Trying to access a Automated DAC is not allowed.

Examples

```
Put_DAC(DAC1, 16383) ; Output 5 volts on DAC1 signal
```

or

```
User1=16383
Put_PWM(PWM1, User1) ; a 50% pulse wave is output on Output 1
```

Automate_DAC(2)

This function is used automatically update the DAC output voltage. This function only need be called once. After this function is called, the DAC output it will run continuously. Note that in this function, the output variable must be a variable.

Parameters

DAC# Identifies which DAC channel is to be read

- can be DAC1 or DAC2

Variable The variable that hold the desired output voltage

- The scale is 0-32767 = 0.0 to 10.0 volts.

Returns

0 – Setup did not execute.
1 – Setup successful.

Error Codes

Bad_ID = an incorrect DAC ID was used.
PT_Range = the Variable used is not acceptable

Examples

```
        User1 = 0
        Automate_DAC (DAC1, User1)
Loop:   User1 = User1 +1 ;This will create a 0 to 10 volt ramp wave on DAC1
        If User1 = 32767
          {
          User1 = 0
           }
        ; add some delay or code here
        Goto Loop
```

Digital Outputs

Put_PWM(2)

This function outputs an Pulse Width Modulated voltage on selected Output pin. This function is used to control the state of the active low output FET drivers. A value of 0 is Off (the output is open). A value of 32767 (the output is fully on, closed to ground). Intermediate values provide a pulse train. A value of 16383 provide a 50% outputs (square wave). This can be useful to provide "average" voltages and regulate current in an inductive load. A constant or a variable may be used as the output Value

Parameters

PWM# Identifies which DAC channel is to be read

- can be PWM1 through PWM16

Value The digital value of the output voltage

- The scale is 0-32767 = 0 to 100% ON (switched to ground).

Returns

0 – new value did not go out.
1 – new value output.

Error Codes

Bad_ID = incorrect PWM ID was used.
Auto_Run = Trying to access a Automated PWM is not allowed.

Examples

```
Put_PWM(PWM1, 16383)  ; a 50% pulse wave is output on Output 1
```

or

```
User1=16383
Put_PWM(PWM1, User1)  ; a 50% pulse wave is output on Output 1
```

Automate_PWM(2)

This function is used automatically update the PWM output. This function only need be called once. After this function is called, the PWM output it will run continuously. Note that in this function, the output variable must be a variable.

Parameters

PWM# Identifies which PWM channel is to be read

- can be PWM1 through PWM16

Variable The variable that hold the desired output voltage

- The scale is 0-32767 = 0 to 100% ON.

Returns

0 – Setup did not execute.
1 – Setup successful.

Error Codes

Bad_ID = an incorrect PWM ID was used.
PT_Range = the Variable used is not acceptable

Examples

```
        User1 = 0
        Automate_PWM (PWM1, User1)
Loop:   User1 = User1 +1 ;This will create a 0 to 100% PWM ramp Output 1
        If User1 = 32767
          {
          User1 = 0
           }
        ; add some delay or code here
        Goto Loop
```

Encoder Inputs

There are two quadrature encoder inputs. These inputs can detect direction, position and velocity from a 2 channel pulse train, 90 degree offset form each other. To use the encoder input, it must be first setup for use as a position counter or as a velocity measurement. After they are set up, the VCL uses special functions to retrieve the count or velocity (in RPM) of the incoming pulse train. The setup also allows enabling error detection.

Setup_Encoder(4)

The VCL must set up the encoder channel before it can be used. The Setup_Encoder function sets the mode, conversion factor and error detection parameters for the encoder VCL functions.

Parameters

ENC# Identifies which Encoder channel is to be read

- can be ENC1 or ENC2

Mode Set the encoder to Velocity or Count mode

- use the predefined constant ENC_COUNT or ENC_VELOCITY

Max_PPR Sets the Pulses Per Revolution or the maximum pulse count error

- If the Mode is ENC_VELOCITY, the value sets the pluses that are seen in one revolution of the encoder. Used to convert to RPM
- If the Mode is ENC_COUNT, the value sets the number of counts on channel A that can be tolerated without a count on channel B before an error is declared. A setting of 0 disables error checking.

Max_Time Used only in ENC_COUNT mode

- Sets the time limit that the count error (in Max_PPR) must be reached within for an error to be declared. Setting this to 0 means there is no time limit. Time limit= Max_Time x 16milliseconds

Returns

0 – Setup did not execute.
1 – Setup successful.

Error Codes

Bad_ID = an incorrect ENC ID was used.

Examples

```
Setup_ENC(ENC1, ENC_VELOCITY , 64,0)
; Velocity mode where 64 pulse = one revolution.
; Note the last parameter is don't care, but MUST be filled with something!


Setup_ENC(ENC2, ENC_COUNT, 10, 10)
; Count mode .
; If we have more that 10 counts on one channel without the other in under 160mS = Error!
```

Get_Encoder_Count(1)

This function retrieves the current position (count) of the encoder. Note that this function is not required to get the current count as it is continuously updated in the *ENC#_Count* variable

Data Values

ENC#_Count Variable that is updated with the value of the encoder count
can be 1 or 2

Parameters

ENC# Identifies which Encoder channel is to be read

- can be ENC1 or ENC2

Returns

N – Encoder count (0 - 32767)

Error Codes

Bad_ID = an incorrect ENC ID was used.

Examples

```
User1 = Get_Encoder_Count(ENC1)
```
or
```
User1 = ENC1_Count
```

Get_Encoder_Vel(1)

This function retrieves the current velocity (PRM) of the encoder. The output is the speed in revolutions Per Minute as scaled by the Setup_Encoder() function Note that this function is not required to get the current velocity as it is continuously updated in the *ENC#_VEL* variable.

Data Values

ENC#_Vel Variable that is updated with the encoder velocity in RPM
can be 1 or 2

Parameters

ENC# Identifies which Encoder channel is to be read

- can be ENC1 or ENC2

Returns

N – Encoder velocity (0 - 32767)

Error Codes

Bad_ID = an incorrect ENC ID was used.

Examples

```
User1 = Get_Encoder_Vel(ENC1)
```
or
```
User1 = ENC1_Vel
```

Get_Encoder_Dir(1)

This function retrieves the current direction (CW or CCW) of the encoder. Note that this function is not required to get the current direction as it is continuously updated in the *ENC#_Dir* variable. However, at low velocity, the direction indicated may not be accurate in the *ENC#_DIR* variable. You must check the *ENC#_VEL* variable to verify if there is no velocity.

Data Values

ENC#_Dir Variable that is updated with the direction of the encoder # can be 1 or 2

Parameters

ENC# Identifies which Encoder channel is to be read

- can be ENC1 or ENC2

Returns

Encoder direction (0 = stopped, -1 = reverse, 1 = forward)

Error Codes

Bad_ID = an incorrect ENC ID was used.

Examples

```
User1 = Get_Encoder_Dir(ENC1)
or
User1 = ENC1_Dir
```

Get_Encoder_Error(1)

This function retrieves the current error status of the encoder. Note that this function is not required to get the current status as it is continuously updated in the *ENC#_Error* variable.

Data Values

ENC#_Vel Variable that is updated with the value of the ADC channel. # can be 1 or 2

Parameters

ENC# Identifies which Encoder channel is to be read

- can be ENC1 or ENC2

Returns

Status (0 = OK, 1 = Channel A error or 2 = Channel B error)

Error Codes

Bad_ID = an incorrect ENC ID was used.

Examples

```
User1 = Get_Encoder_Error(ENC1)
or
User1 = ENC1_Error
```

Real Time Clock (RTC)

The 1310 Vehicle System Controller contains a battery backed-up real time clock. The clock keeps accurate date and time for use by VCL (time stamping errors, clock display, timed events etc). When first used, the RTC may need to be updated. If so, the RTC-Needs_Update variable will be set. The Setup_RTC can be used to set the day and time.

If the RTC stops working, the battery may need to be replaced (should last several years). Open the end cap (the one with the LED status window) by removing the 6 screws. Slide out the button Lithium battery. Replace the battery and end cap. You will need to run a VCL program to update the RTC time and day parameters. See the example following.

After it is set, the RTC will continuously updates the following variables;

- *Hours_24* Actual Hour in 24h format (0...23)
- *Hours_12* Actual Hour in 12h format (1...12)
- *Am_Pm* 0 => AM, 1 => PM
- *Minutes* Actual Minute (0...59)
- *Seconds* Actual Seconds (0...59)
- *Day* Actual Day (1...31)
- *Month* Actual Month (1...12)
- *Year* Year (00...99)
- *Day_of_Week* Actual Day of week (SUNDAY...SATURDAY)*

A few other variables of importance when using the RTC

- *RTC_Needs_Update* Set to 1 when the RTC has to be updated
- *RTC_Disabled* Set to 1 when RTC is disabled

* When setting up the RTC or reading the Day_of_Week variable, VCL must use the predefined constants:

SATURDAY
SUNDAY
MONDAY
TUESDAY
WEDNESDAY
THURSDAY
FRIDAY

Setup_RTC(7)

This function sets up the Real Time Clock date and time. Note that the 24 hour format must be used to set the time.

Parameters

Hours	Actual Hours in 24h format (0...23)
Minutes	Actual Minutes (0...59)
Seconds	Actual Seconds (0...59)
Day	Actual Day of Month (1...31)
Month	Actual Month (1...12)
Year	Actual Year (00...99)
DayofWeek	Actual Day of Week (SUNDAY...SATURDAY)

Returns

0 = Setup failed
1 = Setup correctly executed

Error Codes

PARAM_RANGE = a parameter is out of range

Examples

```
If (RTC_Needs_Update = 1)
   {
   Setup_RTC(14,35,00,6,11,7,MONDAY) ; June 11, 2007 at 2:35pm
   }
```

Hold_RTC(0) and Release_RTC(0)

These functions stop and start the updating of the Date and Time variables. Note that the internal clock continues to run independent of these functions, only the VCL variables are not updated after a HOLD_RTC is called. This can be useful when writing the time out to a display or using them to time stamp a fault into EEPROM, as the time will not tick forward unexpectedly during the process.

Use the Release_RTC to allow the automatic update of the the correct time and date VCL variables to continue.

Hold_RTC() will set the RTC_Disabled variable (=1) and Release_RTC() will clear it (=0)

These functions have no parameters, no errors and return nothing.

Examples

```
Hold_RTC()
Release_RTC()
```

UNIQUE I/O & VCL USAGE

The Curtis 1310 Vehicle System Controller is designed to be extremely flexible and as as such, there is really no "standard configuration" or "standard wiring". Because of its wide ranging application and large array of inputs and outputs, many features and uses of the 1310 may not be readily apparent. This section will cover the unique features and uses of several of the I/O and associated VCL. The sections will provide examples that illustrate some of the key uses of the 1310 Inputs and Outputs. Examples cover such concepts as;

- Switch input usage with rising and falling edge detection (interfacing to push buttons)
- Output 1 & 2 used as current source drivers (interfacing to proportioning valves)
- Using arrays of text in VCL (advanced interfacing to the Spyglass)
- Single channel pulse/frequency counters
- Sensor fault detection (using the measured voltage and load on +5 and +12V)

I/O CONTROL WITH VCL

Digital Inputs

The 1310 controller can have at total of 26 digital inputs (see the model selection chart in Appendix D). Eighteen are switch-to-B+ inputs (SW_1 through SW_18) and 6 are switch-to-ground (SW-19 though SW-26). SW_1 through SW_18 will be sensed ON when switched to B+ and OFF when left open. SW_19 through SW_22 will be sensed ON when left open and OFF when switched to ground. SW_23 though SW_26 are associated with the encoders and are sensed ON when closed to ground and OFF when left open.

To address a digital input in a VCL program, use the desired input label (SW_1 through SW_22). You must use the predefined constants ON or OFF in the code when determining a switch state; using true/false or 1/0 will give erroneous results.

```
if (SW_1 = ON)
  {
  ;put code here to run when switch 1 is On
  }
if (SW_16 = OFF)
  {
  ;put code here to run when switch 16 is Off
  }
```

All switch inputs are automatically debounced by the VCL operating system. This prevents noisy contacts or contact bounce from causing erroneous events in your VCL code. The debounce time can be varied from 0 to 32 milliseconds in 4ms steps, using this function:

```
Setup_Switches(5); 20 milliseconds
```

If this line is not in the VCL code, the debounce time is set at 16 ms.

The previous example "polls" the switch inputs at the the time the statement VCL is run. If there is a need to read fast inputs, the VCL will need to poll these inputs very often as to not miss a correct reading. Sometimes it is not possible to run the VCL fast enough. Big programs or push buttons cause the switch state to be easily missed.

SW_#_UP and SW_#_DOWN variables (where # = 1 though 24) give VCL a better way to catch fast transients on the inputs. The "up" and "down" terms use the illustration of push button, which is pushed down to turn something "on". The 1310 samples the switch states 250 times per second. Any input that has changed state from "off " to "on" will set the corresponding SW_#_DOWN variable. Any switch that has changed state from "on " to "off" will set the corresponding SW_#_UP variable. It is vital to note that once the bit is set, it is not cleared by the corresponding variable. An input going off (released) will not clear the SW_#_DOWN bit and likewise and switch being pushed down (on) will not clear the SW_#_UP bit. It is this feature that allows the VCL code to run less often and still detect input changes, even after the event occurs,. Once the VCL code has detected the change, it can clear the bit to allow the next detection. The example below illustrates a push button interface.

```
if (SW_1_UP = ON)
  {
  ;put code here to run when switch 1 is OFF (up & released)
  SW_1_UP = OFF  ; clear the bit so we can detect the button release
  }
if (SW_1_DOWN = ON)
  {
  ;put code here to run when switch 1 is ON (down & pressed)
  SW_1_DOWN = OFF  ; clear the bit so we can detect the next button press
  }
```

Note that these bits are always checked to be ON, even for the switch off state. Think of it not as the state of the input, but as the transition of the input. In the first line, the VCL checks to see if it is true that the button went "off" (up). Normally, software is written to clear it (OFF) after reading it so that the next state can be caught. Also note that it is entirely possible that both the SW_#_UP and SW_#_DOWN bits are set. This simply means that the input went both on and off within the time it took for the VCL code to return to these lines of code.

Digital Outputs

All 16 outputs on the 1310 are Pulse Width Modulated active low FET drivers. They are not just turned "on" or "off" but must be set to a duty cycle between 0% and 100%. Setting the PWM value to 0 will turn the output off completely (open output) while a setting of 32767 will set it completely on (always pulled to B-). A setting of 16383 provides nearly 50% duty cycle. The Put_PWM and Automate_PWM functions are used for all digital outputs. The variable PWM#_Output (where # is 1 through 16) can be used by VCL to read the present state of any output driver.

Each outputs also has an associated input. This input goes on and off with the PWM and senses the actual state of the FET driver and wiring. Using the input function on an output can allow the wiring of a circuit to be fault checked. Using or basic wiring configuration, the Aux Contactor on Output 2 can be checked for proper connection before and after engaging it.

```
If (PWM2_Output = 0)  ; check if the Aux Contactor is open (off)
  {
  if (SW_2 = ON) ; check if the input is high
    {
    ;The PWM is off and B+ is getting to the pin, so the coil must be connected
    Put_PWM(PWM2, 32767)  ; close the Aux Contactor
    }
  Else ; there is a fault!
    {
  ; the input sense was low, so the coil must be disconnected or open.
  ; put your fault detect code HERE....
  }
}
If (PWM2_Output = 32767)  ; check if the Aux Contactor is closed (on)
  {
  if (SW_2 = ON) ; the input should be low, check if it is high
    {
    ; the PWM is full on but B+ is getting to the pin, so the driver is bad
    ; put your bad FET driver code HERE
    }
  }
```

Because, in the basic wiring configuration, the Aux Contactor is on Output 2, the VCL code can check the the coil, once turned on, draws a reasonable current. Looking at Table to in section 2, the raw value of this current reading is put into ADC16_Output. The example below extends the fault checking to include this feature.

```
If (PWM2_Output = 32767)  ; check if the Aux Contactor is closed (on)
  {
  If (ADC16_Output < 1000) ; the coil is drawing less than the minimum raw current reading
     {
     ; put your low current coil fault detect code HERE....
     }
  }
```

Note that all ADC#_Output values are raw 10 bit value. The VCL programmer must experientially determine the reasonable values for this reading. In the case of ADC15_Output and ADC16_output, a full scale reading (1024) is equal to about 3.33 amps.

The current feedback signal in Output 1 and 2 can also be used to create a current controlled output. This type of output is useful for accurately positioning a flow proportional valve. In the basic wiring diagram, Output1 is wired to such a valve coil.

In order to create a constant current with a PWM output, a PID loop is added, using the current measurement as the feedback (see section 14 of the Common Function manual for more detail). The PID controller automatically regulates the PWM output such that he current reading matches the current command. To keep a valve from sticking in one position, a small amount of "jitter" as added to the command.

```
        ; Setup code
        Kp = 16767 ; Proportional gain
        Ki = 2          ; Integral gain
        Kd = 0        ; Derivative gain
        Command equals User1
        Command = 300 ; about 1 amp
        Automate_PID(PID1, @Command, @ADC16_Output,@Kp,@Ki,@Kd,1,1)
        Automate_PWM(PWM1, PID1_Output)
        ; Main Loop
Main:   If Command = 300
          {
          Command = 305  ; adding some jitter to the command
          }
        Else
          {
          Command = 300
          }
        ;  Add some more code or force a delay HERE
        Goto Main
```

Encoder Inputs

The encoder inputs can also be used as digital inputs. Pulling any of these pins down to ground will cause the input to turn "off". Leaving it open will read "on" as internally these 4 encoder inputs are pulled high to 5v. Care must be taken not connect these inputs to any voltage above 5.5v or the 1310 may be damaged.

```
if (SW_23 = ON)
  {
  ;put code here to run when encoder 1 channel A is On (J4-1)
  }
if (SW_24 = OFF)
  {
  ;put code here to run when encoder 1 channel B is Off (J4-2)
  }
```

These inputs also have the edge triggered variables SW_#_up and SW_#_Down

```
if (SW_23_Up = ON)
  {
  ;put code here to run when encoder 1 channel A goes from On to Off
  SW_23_Up = OFF ; clear the bit so VCL can read it next time around.
  }
```

The encoder channels can also be used to read a single pulse train. In the Basic Wiring Diagram, this can be seen on J4-3, encoder 2 channel A. When using the Setup_Encoder, it is important to turn off any fault checking. The normal ENC#_Count and ENC#_Vel variable will be valid. Note that ENC#_Dir and ENC#_Error have no meaning in a single pulse train measurement.

```
Setup_ENC(ENC2, ENC_COUNT, 0, 0)
      ; Count mode with error checking turned off
```

Normally, the encoder will be powered off the +5 volt supply on J4-15 (ground is J4-16). The current leaving this pin is measured and placed in the variable ADC11_Output. The voltage at this pin is placed in ADC11_Output. Checking the actual values against a known nominal value will allow the VCL to catch a disconnected encoder/sensor (current is too low on ADC11) or a short/excessive current (current is too high on ADC11).

```
if (ADC11_Output < 10)
  {
  ;Error! encoder is disconnected, current draw is too low
  }
if (ADC11_Output > 1000)
  {
  ;Error! There is a short dragging down the supply or too much current draw.
  }
```

If the sensor or encoder needs +12V power, that is available at J3-4 and the output current is sensed at ADC-12.

Arrays

Strings are handled in a unique way in VCL. All the string definitions are taken in order they appear in VCL and concatenated together into one large string array that is attached to the end of the VCL program. The array of strings is then indirectly address through there index into the array. If we know the first message in the array, we can index off it to find the next. In this way, a one dimensional array of strings can be made and addresses. This can be useful in creating messages for the Spyglass.

The following example creates a 5 string array and outputs a new message every 500 ms, to the Spyglass depending on a user variable.

```
                Display_Offset          equals      User1
                MSG_01                  string      "HELLO"
                MSG_02                  string      "ARRAYS"
                MSG_03                  string      "WITHIN"
                MSG_04                  string      "VCL"
                MSG_05                  string      "STRINGS"
                Display_Offset = 0
Main:
                Put_Spy_Text(MSG_01 + Display_Offset)
                setup_delay(DLY2, 500)
                while (DLY2_Output <> 0) {} ; 500 msec
                Display_Offset = Display_Offset + 1
                If ( Display_Offset = 5)
                   {
                   Display_Offset = 0 ; reset the offset if it is past the last message
                   }
                goto Main
```

7 DIAGNOSTICS AND TROUBLESHOOTING

The following errors will be returned if VCL encounters a runtime error while running one of its internal library functions. The error code consists of the Module ID that the error occurred in and a Returned Error Value. The Module ID can be found in the variable Last_VCL_Error_Module The Error Code is held in the variable Last_VCL_Error.

When and error occurs in the VCL runtime library, the ID and value are also automatically sent out over the serial port in Spy Glass format. The format is EV *Module ID – Value*. i.e. EV55-01 means Error VCL DAC Module Bad ID.

Note:All these faults will be flashed on the 1310 status LEDs as a Code "68". (1 red flash followed by 6 yellow flashed, followed by 2 red flasshes and lastly followed by 8 yellow flashes).

VCL Module	Value	Description
MOD_CAN	1	Routines for CAN Bus
MOD_MAP	2	Routines for Mapping
MOD_VCL	3	Run-Time Interpreter
MOD_RMP	4	Routines for Ramping
MOD_SEL	5	Routines for Input Selectors
MOD_DLY	6	Routines for Delays
MOD_LIM	7	Routines for Limits
MOD_SCL	8	Routines for Scaling
MOD_CPY	9	Routines for Copying Objects
MOD_NVM	10	Routines for Non_Volatile Memory Access
MOD_PID	12	Routines for PID Control
MOD_TMR	13	Routines for Timers
MOD_ABS	14	Routines for Absolute Value Functions
MOD_ASP	15	Routines for Asynchronous Serial I/O
MOD_SYS	16	Routines for System Level Functions
MOD_FLT	17	Routines For Filtering
MOD_PAR	18	Routines for Non_Volatile Parameter Access
MOD_SPY	19	Routines for Spyglass
MOD_MTD	21	Routines for Multiply then Divide fucntion
MOD_ESP	23	Routines for Extended Serial Protocol Processing
MOD_ADC	50	Routines for Analog Inputs
MOD_POT	51	Routines for Pot Inputs
MOD_ENC	52	Routines to Read Encoder Inputs
MOD_PWM	53	Routines for PWM Outputs
MOD_SWI	54	Routines for Binary Switch Inputs
MOD_DAC	55	Routines for Analog Outputs
MOD_RTC	56	Routines for Real Time Clock

Returned Error	Value	Description
BAD_ID	1	Bad Index (device ID)
PT_RANGE	2	Variable Table Access out of range
AUTO_RUN	3	Attempt to access element running automatically
UNIT_UNDEF	4	Element accessed before being setup
PARAM_RANGE	5	Parameter is out of range
UNIT_BUSY	6	Unit is already busy
NO_FLASH_BLOCK	7	Could not find the specificed flash block
FLASH_CHECKSUM	8	Flash block checksum is bad
TASK_OVR_RUN	9	Task queue overrun
NO_INIT	10	Tried to execute before initialization
CMD_END	11	Found end of table command
BAD_ENDIF	12	Bad End If location
CALL_OVF	13	Too many call statements
CALL_UNF	14	Too many return statements
OP_BAD	15	Undefined op-code
OP_UNKN	16	Unknown op-code
FUNC_IND	17	Function index is out of range
VAR_IND	18	Variable index is out of range
EVAL_OVF	19	Evaluation stack overflow
EVAL_UNF	20	Evaluation stack underflow
RESULT_OVF	21	Result stack overflow
RESULT_UNF	22	Result stack underflow
LOW_IPMS	23	Not enough instructions per millisecond
BAD_BAUD	30	Bad CAN Baud Rate
BAD_MO_ID	31	Bad Mailbox ID
BAD_MO_LEN	32	Bad Mailbox Length parameter
MO_INACTIVE	33	Attempt to access an inactive mailbox
MO_SENDING	34	Resending a message before old one sent
MO15_XMIT	35	Attempt to set mailbox to transmit
TYPE_UNDEF	40	Accessed a limit block with an undefined type
TYPE_SETUP	41	Accessed a limit block with a bad type specifier
TYPE_RUN	42	Bad type specifier in limit block while running
BAD_EEADD	43	EEPROM Address is out of range
BAD_DEFAULTS	44	Parameter Block Default Values are not up to date
PARAM_RO	45	Tried to Write to a Read-Only Parameter
BAD_P_TYPE	50	Pot Type out of Range
ENC_PHASE	51	Encoder phase error

8 MAINTENANCE

The Real Time Clock battery is the only user serviceable parts in Curtis 1310 Vehicel Syste,m Controller. This battery is accessed from the rear panel (with the label and status LEDs.)
No attempt should be made to open the front panel, remove the PCB or otherwise modify the controller. Doing so may damage the controller and will void the warranty. Carefully follow the procedure below to replace the RTC battery.

It is recommended that the controller and connections be kept clean and dry and that the controller's fault history file be checked and cleared periodically.

CLEANING
Periodically cleaning the controller exterior will help protect it against corrosion and possible electrical control problems created by dirt, grime, and chemicals that are part of the operating environment and that normally exist in battery powered systems.

When working around any battery powered system, proper safety **precautions should be taken.** These include, but are not limited to: proper training, wearing eye protection, and avoiding loose clothing and jewelry.

Use the following cleaning procedure for routine maintenance. Never use a high pressure washer to clean the controller.
1. Remove power by disconnecting the battery.
2. Discharge the capacitors in the controller by connecting a load (such as a contactor coil) across the controller's **B+** and **B** power tabs.
3. Remove any dirt or corrosion from the power and signal connector areas. The controller should be wiped clean with a moist rag. Dry it before reconnecting the battery.
4. Make sure the connections are tight and plugs are seated and latched

Replacing the RTC Battery
It is not likely that you will need to replace the RTC battery as it is designed to last 10 + years. But is the RTC has stopped functioning, the battery may be dead.

1. Remove the 6 Phillips head screws on the rear panel (it has the status LEDs)
2. Carefully slide out the battery, noting the polarity
3. Replace with a identical lithium battery, taking care of the proper polarity
4. Replace rear panel, noting the LED staus is on the lower right.
5. Replace the 6 Phillips screws. You may need to press lightly on the bottom center cover to align the lower screw holes.

APPENDIX A - Design Considerations

ELECTROMAGNETIC COMPATIBILITY (EMC)

Electromagnetic compatibility (EMC) encompasses two areas: emissions and immunity. *Emissions* are radio frequency (RF) energy generated by a product. This energy has the potential to interfere with communications systems such as radio, television, cellular phones, dispatching, aircraft, etc. *Immunity* is the ability of a product to operate normally in the presence of RF energy. EMC is ultimately a system design issue. Part of the EMC performance is designed into or inherent in each component; another part is designed into or inherent in end product characteristics such as shielding, wiring, and layout; and, finally, a portion is a function of the interactions between all these parts. The design techniques presented below can enhance EMC performance in products that use Curtis control products.

Emissions

Signals with high frequency content can produce significant emissions if connected to a large enough radiating area (created by long wires spaced far apart). PWM drivers can contribute to RF emissions. Pulse width modulated square waves with fast rise and fall times are rich in harmonics. (Note: PWM drivers are 100% not contribute to emissions.) The impact of these switching waveforms can be minimized by making the wires from the controller to the load as short as possible and by placing the load drive and return wires near each other.

For applications requiring very low emissions, the solution may involve enclosing the system, interconnect wires and loads together in one shielded box. Emissions can also couple to battery supply leads and circuit wires outside the box, so ferrite beads near the controller may also be required on these unshielded wires in some applications. It is best to keep the noisy signals as far as possible from sensitive wires.

Immunity

Immunity to radiated electric fields can be improved either by reducing overall circuit sensitivity or by keeping undesired signals away from this circuitry. The controller circuitry itself cannot be made less sensitive, since it must accurately detect and process low level signals from sensors such as the throttle potentiometer. Thus immunity is generally achieved by preventing the external RF energy from coupling into sensitive circuitry. This RF energy can get into the controller circuitry via conducted paths and radiated paths. Conducted paths are created by the wires connected to the controller. These wires act as antennas and the amount of RF energy coupled into them is generally proportional to their length. The RF voltages and currents induced in each wire are applied to the controller pin to which the wire is connected.

The Curtis 1310 includes bypass capacitors on the printed circuit board's sensitive input signals to reduce the impact of this RF energy on the internal circuitry. In some applications, additional filtering in the form of ferrite beads may also be required on various wires to achieve desired performance levels. A full metal enclosure can also improve immunity by shielding the 1310 from outside RF energy.

ELECTROSTATIC DISCHARGE (ESD)

Curtis products. Like most modern electronic devices, contain ESD-sensitive components, and it is therefore necessary to protect them from ESD (electrostatic discharge) damage. Most of the product's signal connections have protection for moderate ESD events, but must be protected from damage if higher levels exist in a particular application.

ESD immunity is achieved either by providing sufficient distance between conductors and the ESD source so that a discharge will not occur, or by providing an intentional path for the discharge current such that the circuit is isolated from the electric and magnetic fields produced by the discharge. In general the guidelines presented above for increasing radiated immunity will also provide increased ESD immunity.

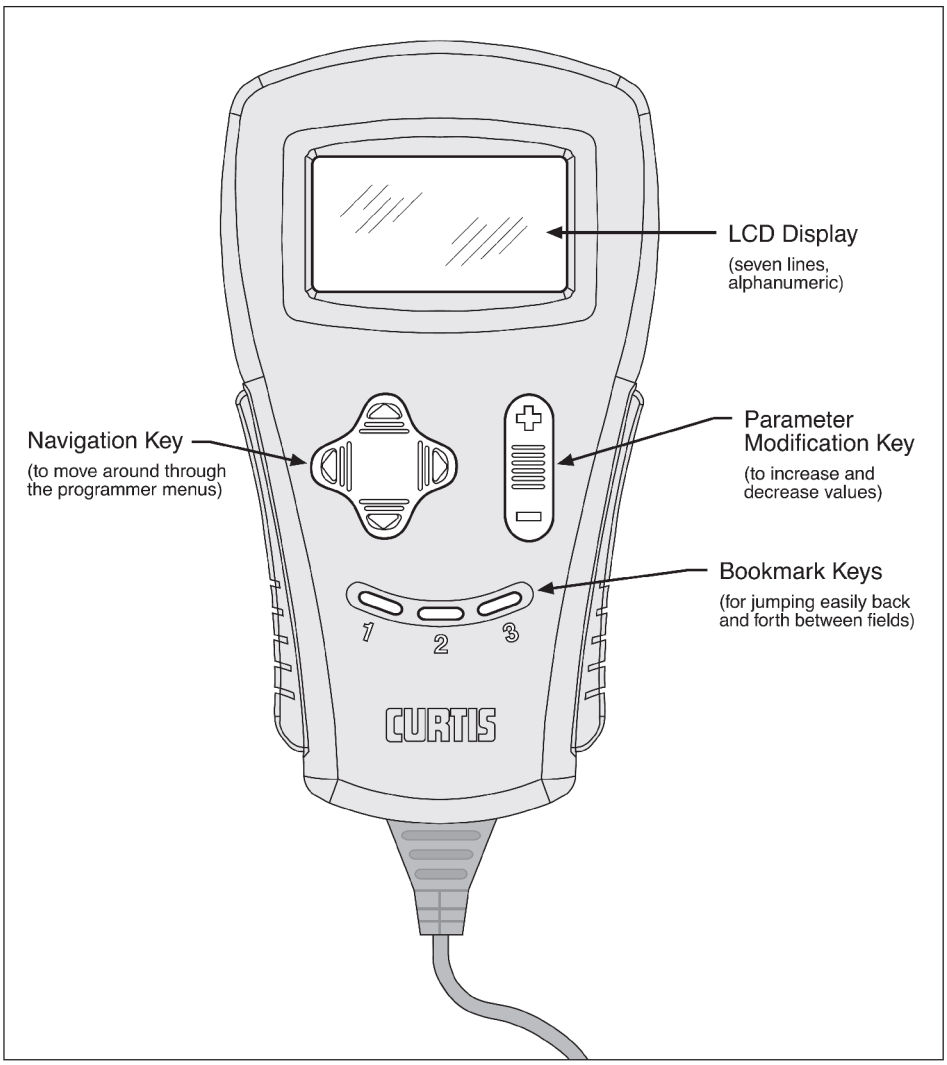
It is usually easier to prevent the discharge from occurring than to divert the current path. A fundamental technique for ESD prevention is to provide adequately thick insulation between all metal conductors and the outside environment so that the voltage gradient does not exceed the threshold required for a discharge to occur. If the current diversion approach is used, all exposed metal components must be grounded. The shielded enclosure, if properly grounded, can be used to divert the discharge current; it should be noted that the location of holes and seams can have a significant impact on ESD suppression. If the enclosure is not grounded, the path of the discharge current becomes more complex and less predictable, especially if holes and seams are involved. Some experimentation may be required to optimize the selection and placement of holes, wires, and grounding paths. Careful attention must be paid to the control panel design so that it can tolerate a static discharge. MOV, transorbs, or other devices can be placed between B- and offending wires, plates, and touch points if ESD shock cannot be otherwise avoided.

APPENDIX B - Programmer

Curtis 1311 HANDHELD PROGRAMMER

The Curtis 1311 handheld programmer provides programming, diagnostic, and test capabilities for the 1234/36/38 controller. The power for operating the programmer is supplied by the host controller via a 4-pin connector wired to the 35-pin low current connector (at pins 7, 25, 28, 29). The unit consists of an LCD display, rocker-type keys for navigating through the display and for modifying parameters (+/-), and three keys that can be used as bookmarks.

Multiple versions of the 1311 programmer are available, each of which can adjust the parameters at its own access level and below. A Dealer programmer, for example, can adjust all the Dealer, Service, and User access parameters, but not the OEM access parameters.



PROGRAMMER OPERATION

The 1311 programmer is easy to use, with self-explanatory functions. After plugging in the programmer, wait a few seconds for it to boot up and gather information from the controller.
For experimenting with settings, the programmer can be left plugged in while the vehicle is driven.

The bookmark keys can make parameter adjustment more convenient. For example, in setting the drive forward throttle parameters, you might set a bookmark at the first of these parameters [Program » Throttle » Forward Offset] and another at the raw throttle readout [Monitor » Inputs » Throttle Pot]; this way you can easily toggle between the readout and the parameters.

PROGRAMMER MENUS

There are six main menus, which in turn lead to nested submenus:

Program — provides access to the individual programmable parameters (see Section 3).

Monitor — presents real-time values during vehicle operation; these include all inputs and outputs, as well as the mapped throttle values and conditioned throttle requests (see Section 4).

Faults — presents diagnostic information, and also a means to clear the fault history file (see Section 7).

Functions — provides access to the controller-cloning commands and to the "reset" command.

Information — displays data about the host controller: model and serial numbers, date of manufacture, hardware and software revisions, and itemization of other devices that may be associated with the controller's operation.

Programmer Setup — displays data about the programmer: model and serial numbers, date of manufacture, and a list of the programmable parameters thatcan be accessed with this particular programmer.

APPENDIX C - Specifications

1310 Specifications

Nominal Input Voltage	24 – 48 volts
Electrical Isolation to case	500 VAC
Storage Ambient Temperature Range	-50 to +90 ºC
Operation Ambient Temperature Range	-40 to +85 ºC
Enclosure Protection Rating	IP42
Weight	0.82 Kg
Dimensions	198 x 138 x 46 (mm) (l x w x h)

Model Number	Voltage	Outputs / Inputs	Options
1310-5210	24-48v	8 output / 18 input*†	RTC,
1310-5310	24-48v	16 output / 22 input*	RTC, PV drivers**

* Encoder inputs can be used for an additional 4 inputs

** PV (Proportional Value) drivers are on outputs 1 and 2 and have current feedback

† Outputs 9 through 16 are available and Inputs 14,15,17 and 18 are <u>not</u> available.