PME Orchestra User Manual

Table of Contents

Introduction 4
Term Definitions 4
What a Typical System Looks Like 5
Quick Start 7
    Creating a New Project 7
    Setting Up Project 11
    Coding Within Orchestra 2 14
Data Items 20
    Input 20
    Switch to Battery (STB) 20
    Switch to Ground (STG) 20
    Voltage to Digital (VTD) 22
    Resistance to Digital (RTD) 23
    Frequency (Freq) 24
    Pulse Width Modulation (PWM) 25
    Pulse Counter (Count) 26
    Internal 27
    Encoder 27
    Output 28
    Digital 29
    PWM 30
    Current Controlled (Single Wire) 32
    Current Controlled (Dual Wire) 34
    Frequency 36
    Variable 38
    State Machine 40
    Time Counter 41
    EEPROM 42
    CAN Receive 43
    Constant 45
    COM Bridge 45
Comparison Blocks 47
    Equals (A==B) 48
    Less Than (A<B) 48
    Greater Than (A>B) 48
    Not Equal (A<>B) 49
    Less Than or Equal To (A<=B) 49

Greater Than or Equal To (A>=B) 49
Greater Than, Less Than (B<A<C) 50
Greater Than or Equal To, Less Than or Equal To (B<=A<=C) 50
Less Than OR Greater Than ((A<B) or (A>C)) 50
Less Than or Equal To OR Greater Than or Equal To ((A<=B) or (A>=C)) 51
AND Equal To ((A && B) = C) 51
Operator Blocks 52
Set 53
Dec 53
Inc 54
Percent 54
Add 55
Sub 55
Mult 56
Div 56
PID>T 57
PID<T 57
PIDspd 58
Ramp 58
JOYabv 59
JOYblw 59
SETbit 60
CLRbit 61
Lshift 61
Rshift 62
5 Volt 62
CAN Tx 63
Sort 63
Wt AVG 64
SaveET 64
Arranger 66
Label 68
Image 69
Needle 70
Progress Bar 73
Time/Date 75
Video 76
Curved Progress Bar 77
Table 80
Downloader 82
Firmware Download 82
Application Download 84
Display Download 85
Downloader Wizard Packet 87
Downloading from a Packet 87
Creating a Packet 89
Application Configurator 92
Download To Module 92
Upload From Module 93
Convert Chart 94

Presto 95
- Material 95
- Setup 96
- Compiling a Presto Project in Orchestra 97
- The C Project 98
- Compiling and Debugging a C Project 100
- Reloading the Boot Code 100
- Software Notes 100

Appendix 101

Introduction

Orchestra is an integrated development environment that contains the following tools:

– Arranger
– Composer
– Conductor
– Downloader
– Application Configurator

To assist the User in creating an application using off the shelf CANLink Modules regardless of software experience. Applications can be written exclusively by Hydro Electronic Devices (WAG), the original equipment manufacturer (OEM), or a combination of both. The applications also have some flexibility in terms of how they are written. Based on the requirements of the application and the customer, Orchestra allows the software to be written with rungs using ladder logic or coded within C or C++.

Term_Definitions

Application	Software created by the User to control module specific functions. This software is downloaded to the master module.
CAN	Please reference Bosch 2.0 A and B Controller Area Network Specification.
CANLink Module	WAG product utilizing the CANLink protocol.
CANLink Protocol	WAG proprietary J1939 compatible CAN protocol.
Display	Programmable piece of hardware that can give a visual representation of the Application
WAG	Wandfluh AG
I/O	A module’s means of interface to the physical world – short hand for inputs and outputs.
Module	A combination of hardware and software. This includes inputs, outputs, displays, modem, etc.
Master Module	The module to which the application software resides
I/O Module	Client module that has no application software and interfaces its I/O to the master module
OEM	Original Equipment Manufacturer
User	Person operating Orchestra

What a Typical System Looks Like

A typical system consists of a number of CANLink Modules that work together to control some aspect(s) of a vehicle. That control is done through the use of particular communication protocols between the modules. Currently CAN, CANOpen, J1939 and J1708, are used to communicate between the modules, engines, transmissions, etc.; while RS232 and USB are used to communicate with external PCs. Not every module supports all of the protocols and it is highly advised to consult datasheets to determine what is and is not supported with each module.
The CANLink Modules work together in a way that there is a single master Module that will control the other Modules that are designated as I/O Modules, up to 40 total Modules, to accomplish the goal of the system. Each system must have a Master Module, but not every system needs to have I/O Modules; the master Module can act as both. What is needed will be based on the requirements of the Application.

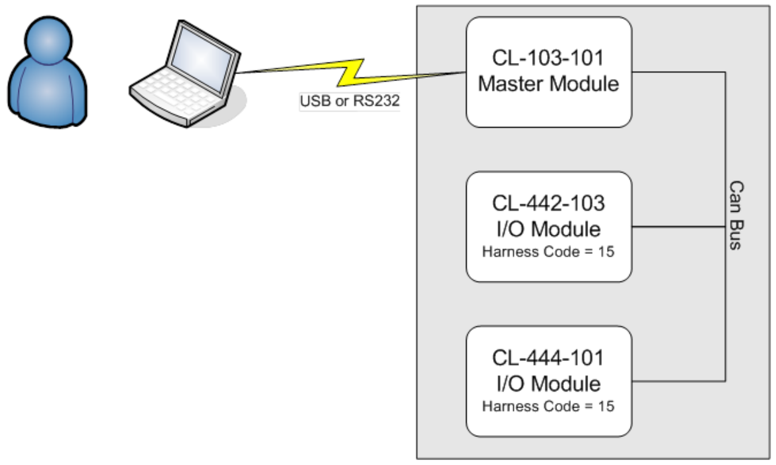


FIGURE 1: SIMPLE SYSTEM EXAMPLE

Each I/O Module has a Module type and harness code, where they type is a unique identifier based on the hardware itself, and the code allows for multiple modules with the same type to exist within the same system. The combination of the two identifies the I/O Module to the master Module. Within a system, two I/O Modules cannot have the same combination of type and code, but two Modules can have the same code as long as the type is different. Orchestra will prevent the addition of Modules that match any existing Modules already in the project.

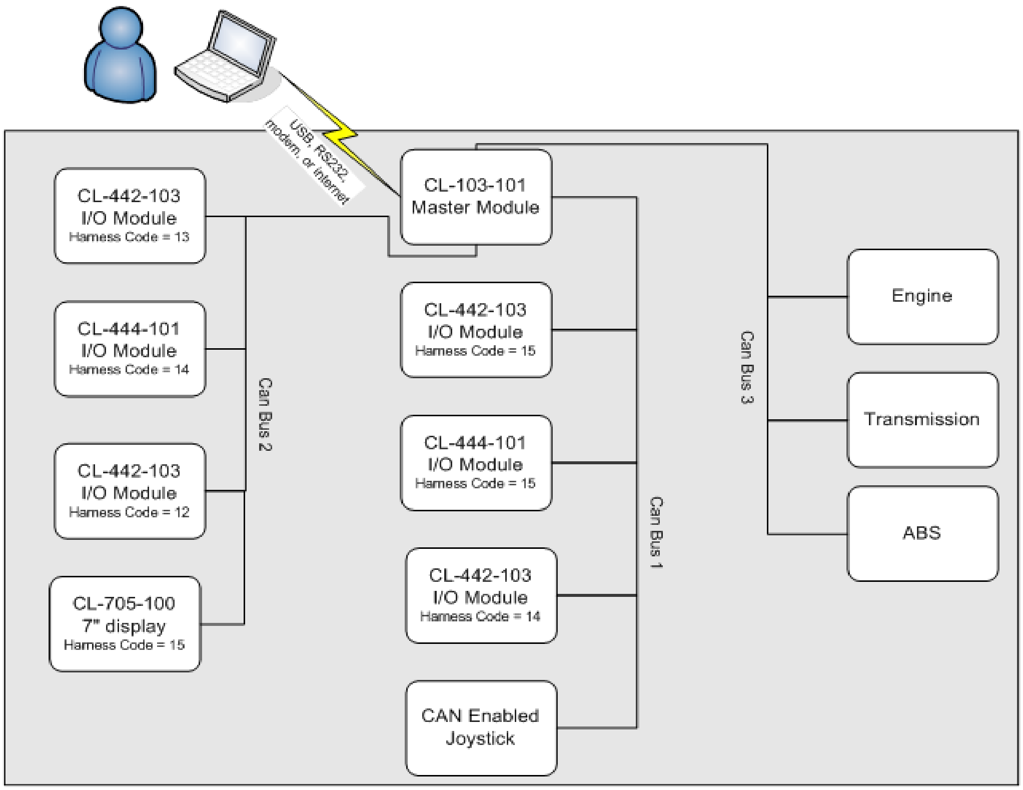

USB, RS232, modem, or internet
CL-103-101 Master Module
CL-442-103 I/O Module Harness Code = 13
CL-444-101 I/O Module Harness Code = 14
CL-442-103 I/O Module Harness Code = 12
CL-705-100 7" display Harness Code = 15
Can Bus 2
CL-442-103 I/O Module Harness Code = 15
CL-444-101 I/O Module Harness Code = 15
CL-442-103 I/O Module Harness Code = 14
CAN Enabled Joystick
Can Bus 1
Can Bus 3
Engine
Transmission
ABS

FIGURE 2: ADVANCED SYSTEM EXAMPLE

Quick Start

This Quick Start guide will walkthrough how to set up a project within Orchestra.

Creating a New Project

To create a project within Orchestra, first, Orchestra needs to be opened by clicking the Orchestra executable within the All Programs pane of the Start explorer.

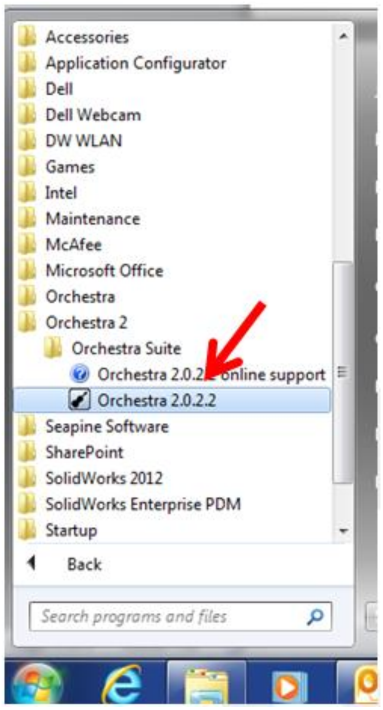


FIGURE 3: OPENING ORCHESTRA

A window like the one below in Figure 4 should be seen.

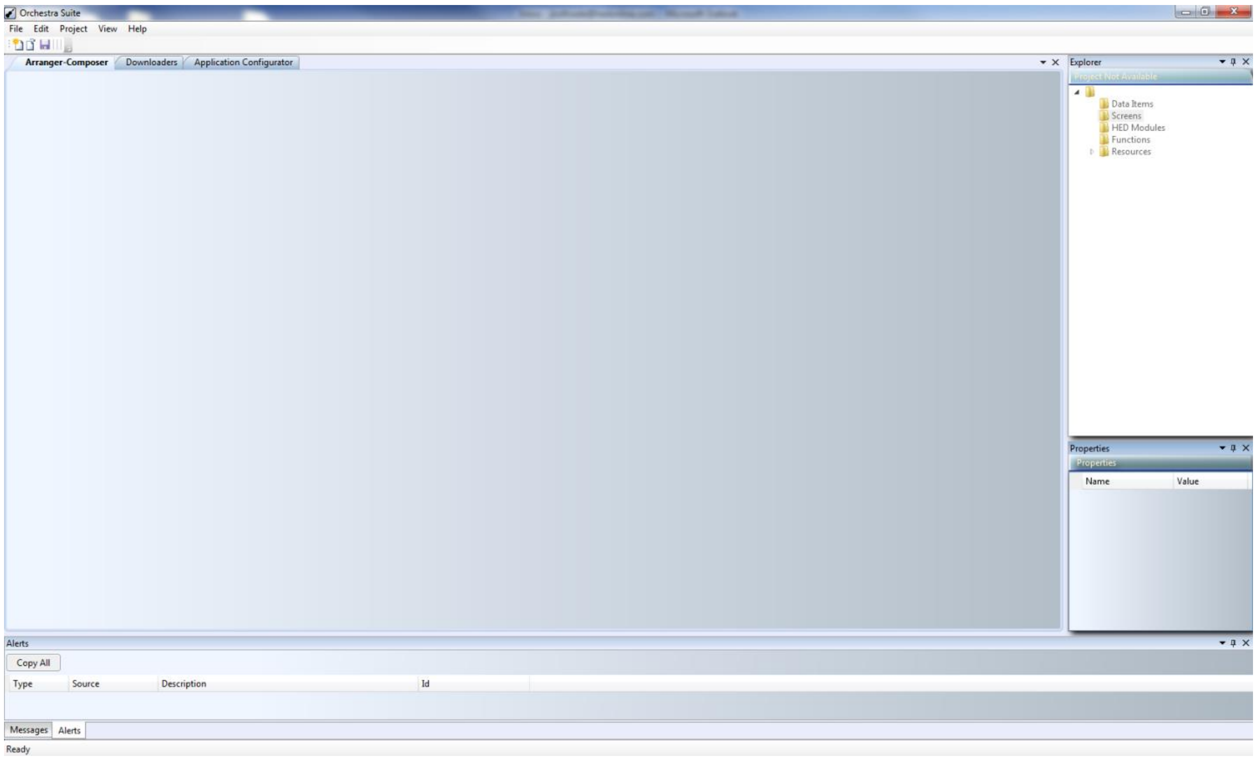


FIGURE 4: ORCHESTRA BEGINNING SCREEN

This is the Composer screen; it has four separate panes; the Designer pane, Explorer pane, Properties pane, and Alerts pane. Along with the four panes, there is the File, Edit, Project, View, and Help drop down menus. To start a project, either click the blank page icon underneath the File menu, or go into the File drop down menu and select New Project.

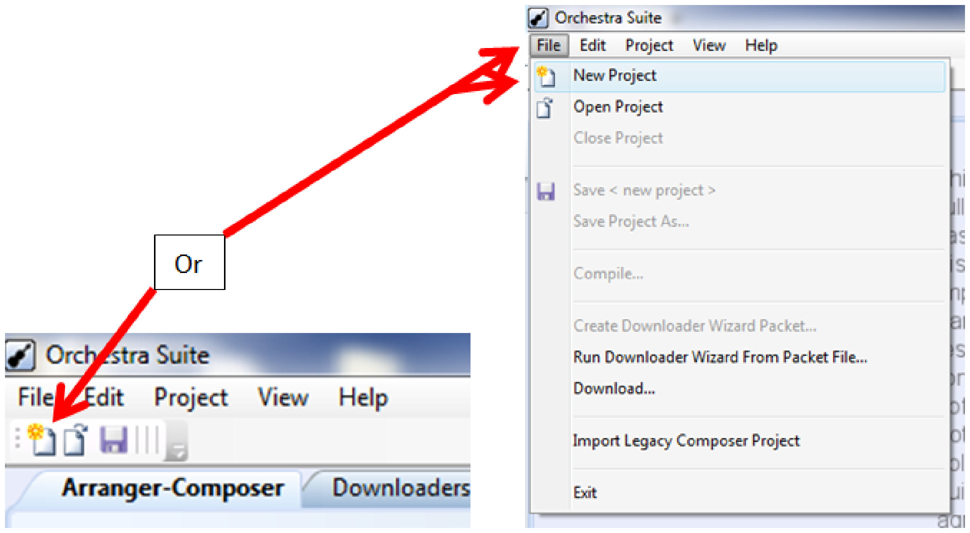
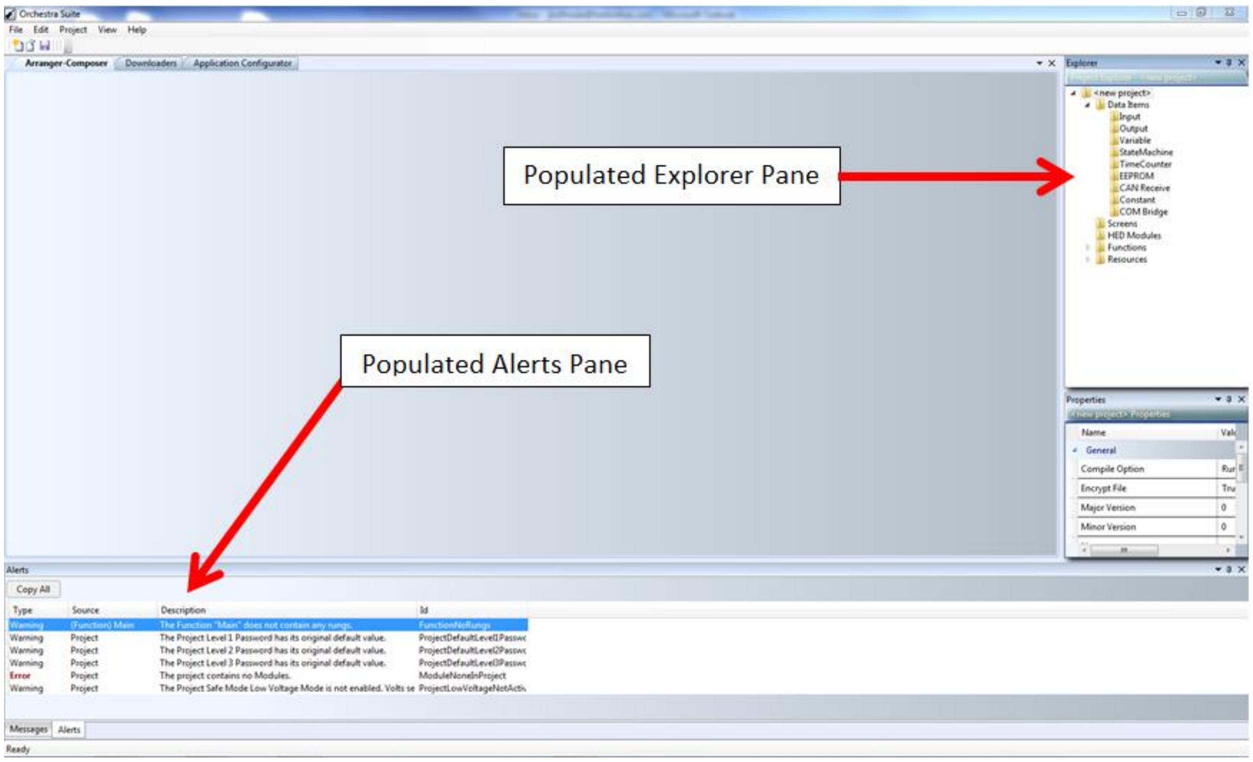

Orchestra Suite
File Edit Project View Help
New Project
Open Project
Close Project
Save < new project >
Save Project As...
Compile...
Create Downloader Wizard Packet...
Run Downloader Wizard From Packet File...
Download...
Import Legacy Composer Project
Exit
Or
Orchestra Suite
File Edit Project View Help
Arranger-Composer
Downloaders

FIGURE 5: CREATING NEW PROJECT

Once the new project is created, the Explorer and Alerts panes will populate. The Explorer pane consists of folders and subfolders that will contain everything the Application will contain in an easily accessible manner. Within the Explorer all of the folders are contained within the project folder. From there the subsequent subfolders are broken down into Data Items, Screens, WAG Modules, Functions, and Resources.

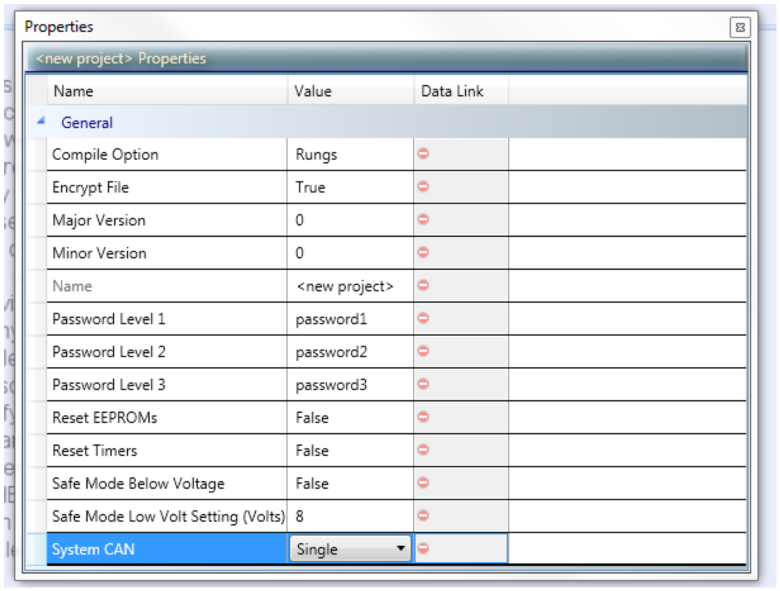


Name	Val
Compile Option	Rur
Encrypt File	Tru
Major Version	0
Minor Version	0

Type	Source	Description	Id
Warning	(Function) Main	The Function "Main" does not contain any rungs.	FunctionNoRungs
Warning	Project	The Project Level 1 Password has its original default value.	ProjectDefaultLevel1Passwc
Warning	Project	The Project Level 2 Password has its original default value.	ProjectDefaultLevel2Passwc
Warning	Project	The Project Level 3 Password has its original default value.	ProjectDefaultLevel3Passwc
Error	Project	The project contains no Modules.	ModuleNoneInProject
Warning	Project	The Project Safe Mode Low Voltage Mode is not enabled. Volts se	ProjectLowVoltageNotActi\

FIGURE 6: EXPLORER AND ALERTS PANES

The Data Items folder comprises all of the data types supported by the Application in a fashion that breaks out each type for quick and easy access by the User. The Screen folder will contain whatever screen(s) created using the Arranger tool. Next, WAG Modules folder is where the modules selected for the project will be stored for reference, editing, and mapping. The Functions folder is where the Application rung logic will be created and edited. Finally, the Resources folder stores all of the Image, Color, and String lists the User may create for the Application.
When a new project is created the Alerts pane will come up with a number of errors and warnings because the settings within the new project have not yet been set by the User and there is also no logic contained within the Function Main. As the settings are filled in and logic is added the warning and errors will go away. This pane is where Orchestra will signal to the User any errors and warnings Orchestra finds with the current Application automatically without having to compile first. The error and warning detection in Orchestra does have limitations regarding what it catches and what the User will be required to catch manually.
The Properties pane will auto-populate with the available properties of whatever is currently highlighted within the Explorer pane. Highlighting the Project folder, all of the associated properties show up for the User to edit as needed. Everything from the compile option to project name to password protection can be changed. The compile option is what determines how the program will be written and has three options; Rungs, Presto, and Presto with Rungs.


Properties

<new project> Properties

Name	Value	Data Link
General		
Compile Option	Rungs	⊖
Encrypt File	True	⊖
Major Version	0	⊖
Minor Version	0	⊖
Name	<new project>	⊖
Password Level 1	password1	⊖
Password Level 2	password2	⊖
Password Level 3	password3	⊖
Reset EEPROMs	False	⊖
Reset Timers	False	⊖
Safe Mode Below Voltage	False	⊖
Safe Mode Low Volt Setting (Volts)	8	⊖
System CAN	Single	⊖

FIGURE 7: PROPERTIES PANE

As a note, the Alerts, Properties, and Explorer panes can be undocked, redocked, and resized as the User sees fit, and if one of those panes was closed it can be accessed again using the View pull down menu. Using the Properties pane in conjunction with the Explorer pane, the User has a quick reference of all the properties associated with anything contained within the Application.

Setting Up Project

To continue from a new project, a Master Module needs to be added to the project. That can be achieved by right clicking on the WAG Modules folder and then left clicking the "Select Module(s)…" option that appears.

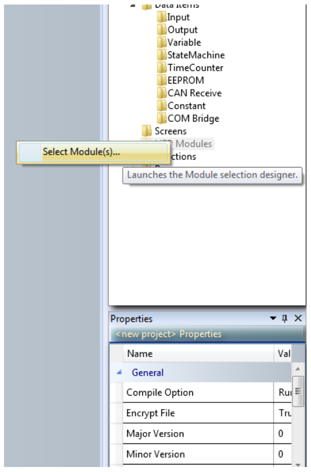
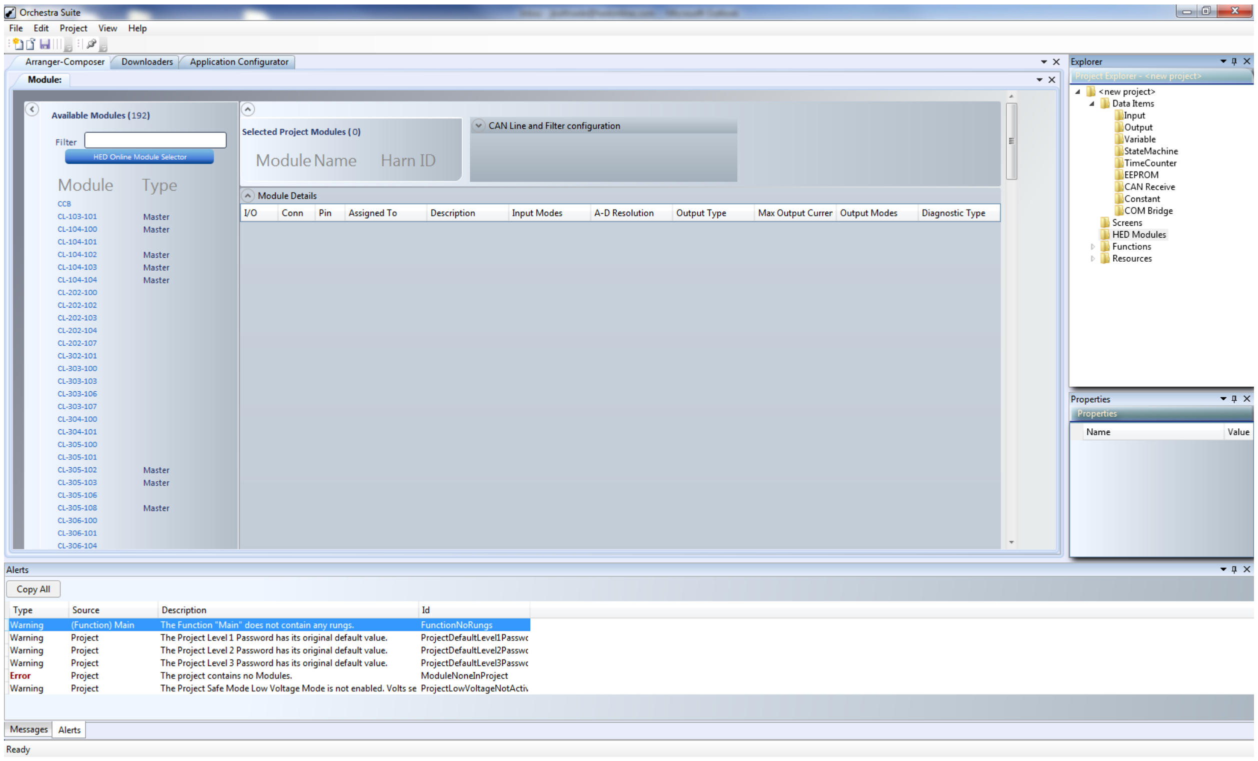


FIGURE 8: SELECTING A MODULE

Clicking that option will populate the Designer pane with an interactive list of all of the Modules supported within Orchestra, as well as, also displaying whether or not that particular Module is a Master Module or not.



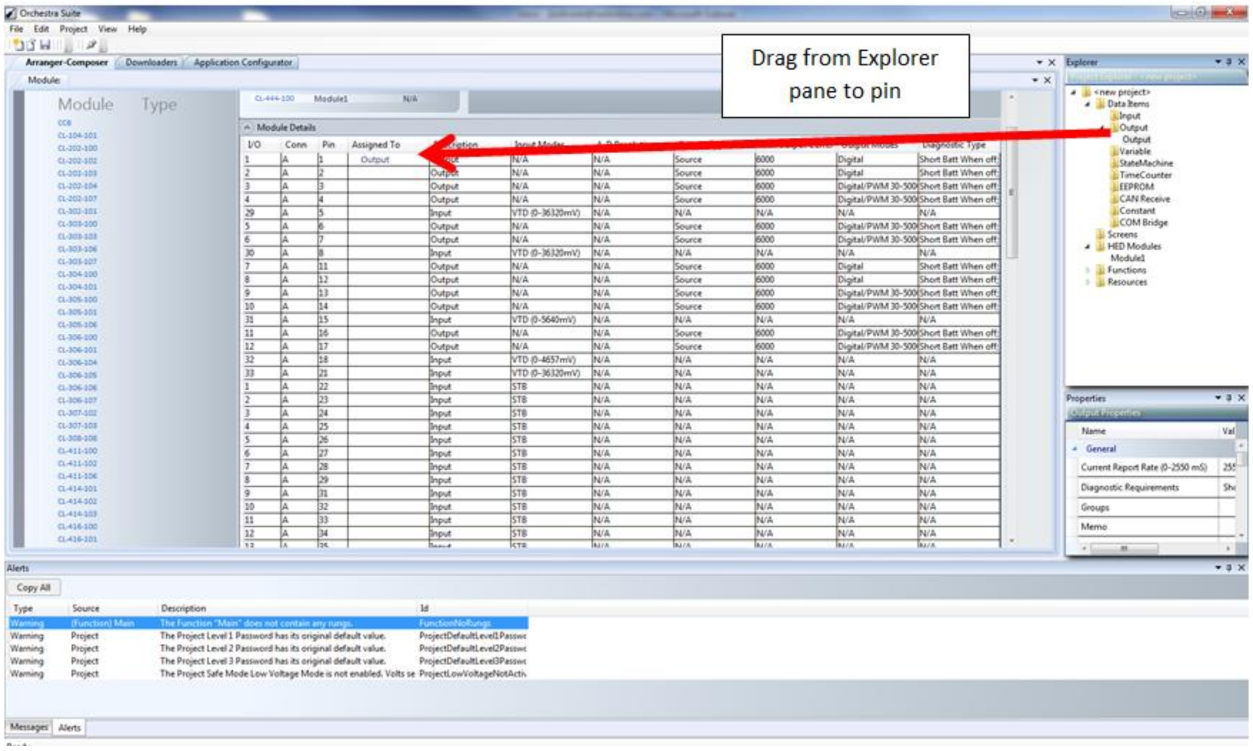
Module	Type
CCB	
CL-103-101	Master
CL-104-100	Master
CL-104-101	
CL-104-102	Master
CL-104-103	Master
CL-104-104	Master
CL-202-100	
CL-202-102	
CL-202-103	
CL-202-104	
CL-202-107	
CL-302-101	
CL-303-100	
CL-303-103	
CL-303-106	
CL-303-107	
CL-304-100	
CL-304-101	
CL-305-100	
CL-305-101	
CL-305-102	Master
CL-305-103	Master
CL-305-106	
CL-305-108	Master
CL-306-100	
CL-306-101	
CL-306-104	

I/O	Conn	Pin	Assigned To	Description	Input Modes	A-D Resolution	Output Type	Max Output Curren	Output Modes	Diagnostic Type

Type	Source	Description	Id
Warning	(Function) Main	The Function "Main" does not contain any rungs.	FunctionNoRungs
Warning	Project	The Project Level 1 Password has its original default value.	ProjectDefaultLevel1Passwc
Warning	Project	The Project Level 2 Password has its original default value.	ProjectDefaultLevel2Passwc
Warning	Project	The Project Level 3 Password has its original default value.	ProjectDefaultLevel3Passwc
Error	Project	The project contains no Modules.	ModuleNoneInProject
Warning	Project	The Project Safe Mode Low Voltage Mode is not enabled. Volts se	ProjectLowVoltageNotActiv

FIGURE 9: MODULE SELECTION LIST

Once a Master Module has been selected it gets added to the WAG Modules folder, and Orchestra will automatically remove all other Master Modules from the list, since only one Master Module can exist in each Application. Also, once a module is selected, the Module Details section will fill with all of the detailed pin assignments. From there the User can drag and drop the proper Data Items to a pin to assign and link that Data Item with that pin on the Module. To do that a Data Item would have to first be created using the Explorer Pane by right clicking on the particular Data Item desired and then selecting the add option. Once the add option is selected the Data Item appears as a sub item in its respective subfolder. Edit the name and other properties by highlighting it and using the Properties Pane.

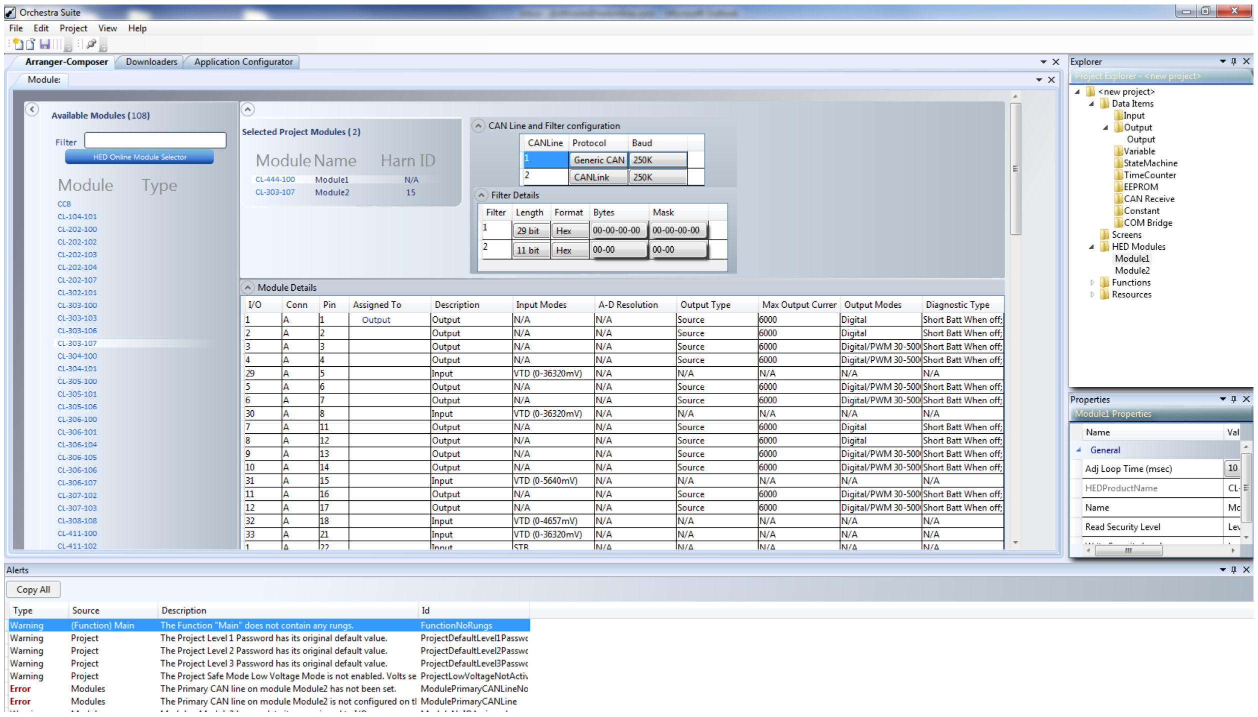


I/O	Conn	Pin	Assigned To	Description	Input Modes	[illegible]	[illegible]	[illegible]	Output Modes	Diagnostic Type
1	A	1	Output	Output	N/A	N/A	Source	6000	Digital	Short Batt When off
2	A	2		Output	N/A	N/A	Source	6000	Digital	Short Batt When off
3	A	3		Output	N/A	N/A	Source	6000	Digital/PWM 30-500	Short Batt When off
4	A	4		Output	N/A	N/A	Source	6000	Digital/PWM 30-500	Short Batt When off
29	A	5		Input	VTD (0-36320mV)	N/A	N/A	N/A	N/A	N/A
5	A	6		Output	N/A	N/A	Source	6000	Digital/PWM 30-500	Short Batt When off
6	A	7		Output	N/A	N/A	Source	6000	Digital/PWM 30-500	Short Batt When off
30	A	8		Input	VTD (0-36320mV)	N/A	N/A	N/A	N/A	N/A
7	A	11		Output	N/A	N/A	Source	6000	Digital	Short Batt When off
8	A	12		Output	N/A	N/A	Source	6000	Digital	Short Batt When off
9	A	13		Output	N/A	N/A	Source	6000	Digital/PWM 30-500	Short Batt When off
10	A	14		Output	N/A	N/A	Source	6000	Digital/PWM 30-500	Short Batt When off
31	A	15		Input	VTD (0-5640mV)	N/A	N/A	N/A	N/A	N/A
11	A	16		Output	N/A	N/A	Source	6000	Digital/PWM 30-500	Short Batt When off
12	A	17		Output	N/A	N/A	Source	6000	Digital/PWM 30-500	Short Batt When off
32	A	18		Input	VTD (0-4657mV)	N/A	N/A	N/A	N/A	N/A
33	A	21		Input	VTD (0-36320mV)	N/A	N/A	N/A	N/A	N/A
1	A	22		Input	STB	N/A	N/A	N/A	N/A	N/A
2	A	23		Input	STB	N/A	N/A	N/A	N/A	N/A
3	A	24		Input	STB	N/A	N/A	N/A	N/A	N/A
4	A	25		Input	STB	N/A	N/A	N/A	N/A	N/A
5	A	26		Input	STB	N/A	N/A	N/A	N/A	N/A
6	A	27		Input	STB	N/A	N/A	N/A	N/A	N/A
7	A	28		Input	STB	N/A	N/A	N/A	N/A	N/A
8	A	29		Input	STB	N/A	N/A	N/A	N/A	N/A
9	A	31		Input	STB	N/A	N/A	N/A	N/A	N/A
10	A	32		Input	STB	N/A	N/A	N/A	N/A	N/A
11	A	33		Input	STB	N/A	N/A	N/A	N/A	N/A
12	A	34		Input	STB	N/A	N/A	N/A	N/A	N/A
13	A	35		Input	STB	N/A	N/A	N/A	N/A	N/A

Type	Source	Description	Id
Warning	(Function) Main	The Function "Main" does not contain any rungs.	FunctionNoRungs
Warning	Project	The Project Level 1 Password has its original default value.	ProjectDefaultLevel1Passw
Warning	Project	The Project Level 2 Password has its original default value.	ProjectDefaultLevel2Passw
Warning	Project	The Project Level 3 Password has its original default value.	ProjectDefaultLevel3Passw
Warning	Project	The Project Safe Mode Low Voltage Mode is not enabled. Volts se	ProjectLowVoltageNotActi

FIGURE 10: DATA ITEM LINKING

Orchestra will prevent the User from assigning a Data Item to a pin that cannot support that Data Item, i.e. assigning a Switch to Battery (STB) defined Input to a Voltage to Digital (VTD) Input pin.
Each subsequent Module added to the project is added as an I/O Module, and, like the Master Module, will show up within the WAG Modules folder. Double clicking a Module in the folder will bring up the Module Details in the Designer pane where pin assignments can be made. Also within the Properties pane the loop time, name, and read/write security settings can be adjusted. Once all of the Modules are selected the User should determine and assign what kind of protocol the Modules will use as well as the baud rate, which CAN lines the I/O Modules will use, and the harness code of each I/O Module using the corresponding drop down menus. All of that can be done using the CAN Line and Filter configuration menu next to the Selected Project Modules.



CAN Line and Filter configuration

CANLine	Protocol	Baud
1	Generic CAN	250K
2	CANLink	250K

Filter Details

Filter	Length	Format	Bytes	Mask
1	29 bit	Hex	00-00-00-00	00-00-00-00
2	11 bit	Hex	00-00	00-00

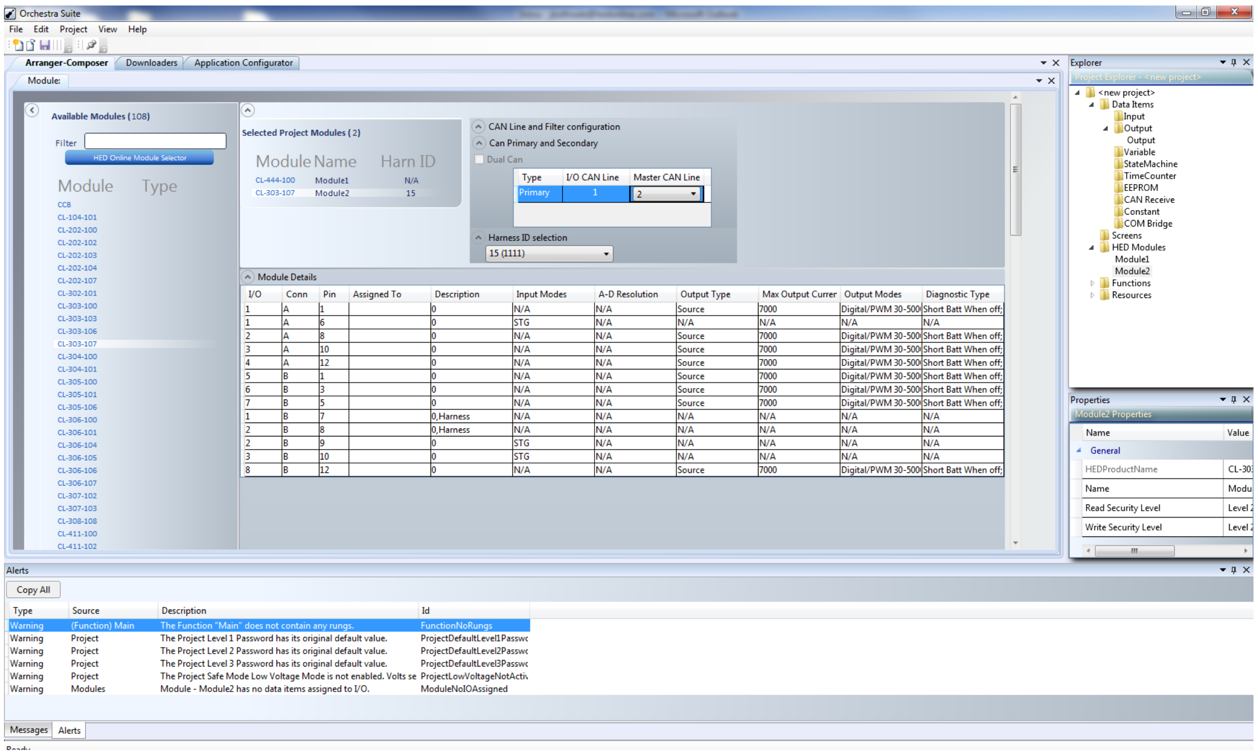
Module Details

I/O	Conn	Pin	Assigned To	Description	Input Modes	A-D Resolution	Output Type	Max Output Currer	Output Modes	Diagnostic Type
1	A	1	Output	Output	N/A	N/A	Source	6000	Digital	Short Batt When off;
2	A	2		Output	N/A	N/A	Source	6000	Digital	Short Batt When off;
3	A	3		Output	N/A	N/A	Source	6000	Digital/PWM 30-500	Short Batt When off;
4	A	4		Output	N/A	N/A	Source	6000	Digital/PWM 30-500	Short Batt When off;
29	A	5		Input	VTD (0-36320mV)	N/A	N/A	N/A	N/A	N/A
5	A	6		Output	N/A	N/A	Source	6000	Digital/PWM 30-500	Short Batt When off;
6	A	7		Output	N/A	N/A	Source	6000	Digital/PWM 30-500	Short Batt When off;
30	A	8		Input	VTD (0-36320mV)	N/A	N/A	N/A	N/A	N/A
7	A	11		Output	N/A	N/A	Source	6000	Digital	Short Batt When off;
8	A	12		Output	N/A	N/A	Source	6000	Digital	Short Batt When off;
9	A	13		Output	N/A	N/A	Source	6000	Digital/PWM 30-500	Short Batt When off;
10	A	14		Output	N/A	N/A	Source	6000	Digital/PWM 30-500	Short Batt When off;
31	A	15		Input	VTD (0-5640mV)	N/A	N/A	N/A	N/A	N/A
11	A	16		Output	N/A	N/A	Source	6000	Digital/PWM 30-500	Short Batt When off;
12	A	17		Output	N/A	N/A	Source	6000	Digital/PWM 30-500	Short Batt When off;
32	A	18		Input	VTD (0-4657mV)	N/A	N/A	N/A	N/A	N/A
33	A	21		Input	VTD (0-36320mV)	N/A	N/A	N/A	N/A	N/A
1	A	22		Input	STB	N/A	N/A	N/A	N/A	N/A

Alerts

Type	Source	Description	Id
Warning	(Function) Main	The Function "Main" does not contain any rungs.	FunctionNoRungs
Warning	Project	The Project Level 1 Password has its original default value.	ProjectDefaultLevel1Passwc
Warning	Project	The Project Level 2 Password has its original default value.	ProjectDefaultLevel2Passwc
Warning	Project	The Project Level 3 Password has its original default value.	ProjectDefaultLevel3Passwc
Warning	Project	The Project Safe Mode Low Voltage Mode is not enabled. Volts se	ProjectLowVoltageNotActiv
Error	Modules	The Primary CAN line on module Module2 has not been set.	ModulePrimaryCANLineNo
Error	Modules	The Primary CAN line on module Module2 is not configured on tl	ModulePrimaryCANLine

FIGURE 11: CAN LINE CONFIGURATION



CAN Line and Filter configuration

Can Primary and Secondary

Type	I/O CAN Line	Master CAN Line
Primary	1	2

Harness ID selection: 15 (1111)

Module Details

I/O	Conn	Pin	Assigned To	Description	Input Modes	A-D Resolution	Output Type	Max Output Currer	Output Modes	Diagnostic Type
1	A	1		0	N/A	N/A	Source	7000	Digital/PWM 30-500	Short Batt When off;
1	A	6		0	STG	N/A	N/A	N/A	N/A	N/A
2	A	8		0	N/A	N/A	Source	7000	Digital/PWM 30-500	Short Batt When off;
3	A	10		0	N/A	N/A	Source	7000	Digital/PWM 30-500	Short Batt When off;
4	A	12		0	N/A	N/A	Source	7000	Digital/PWM 30-500	Short Batt When off;
5	B	1		0	N/A	N/A	Source	7000	Digital/PWM 30-500	Short Batt When off;
6	B	3		0	N/A	N/A	Source	7000	Digital/PWM 30-500	Short Batt When off;
7	B	5		0	N/A	N/A	Source	7000	Digital/PWM 30-500	Short Batt When off;
1	B	7		0,Harness	N/A	N/A	N/A	N/A	N/A	N/A
2	B	8		0,Harness	N/A	N/A	N/A	N/A	N/A	N/A
2	B	9		0	STG	N/A	N/A	N/A	N/A	N/A
3	B	10		0	STG	N/A	N/A	N/A	N/A	N/A
8	B	12		0	N/A	N/A	Source	7000	Digital/PWM 30-500	Short Batt When off;

Alerts

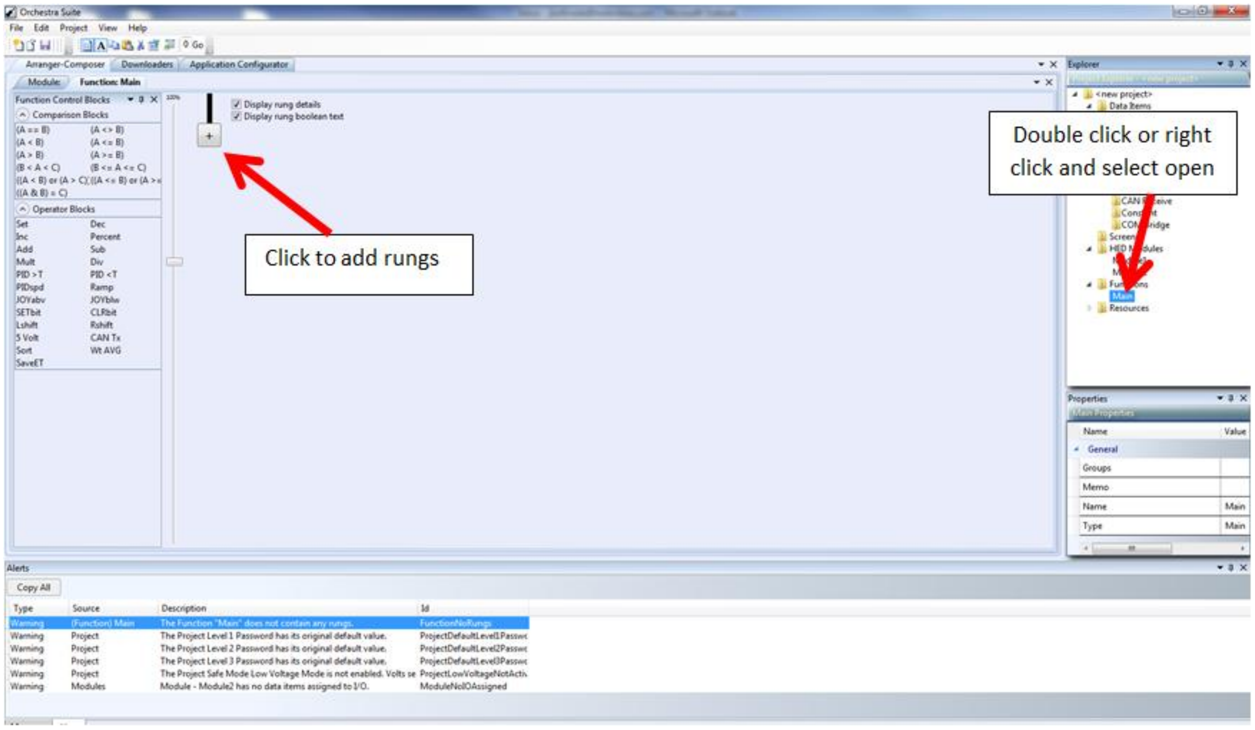
Type	Source	Description	Id
Warning	(Function) Main	The Function "Main" does not contain any rungs.	FunctionNoRungs
Warning	Project	The Project Level 1 Password has its original default value.	ProjectDefaultLevel1Passwc
Warning	Project	The Project Level 2 Password has its original default value.	ProjectDefaultLevel2Passwc
Warning	Project	The Project Level 3 Password has its original default value.	ProjectDefaultLevel3Passwc
Warning	Project	The Project Safe Mode Low Voltage Mode is not enabled. Volts se	ProjectLowVoltageNotActiv
Warning	Modules	Module - Module2 has no data items assigned to I/O.	ModuleNoIOAssigned

FIGURE 12: I/O MODULE CAN LINE SELECTION

The protocols that can be selected by clicking on the protocol are CANLink, Generic CAN, or Not Defined. If there are more than one CAN lines available and only one is desired, the User can choose which line to use and then assign the other as Not Defined. After the protocol is selected the baud rate can be changed by clicking on the number and selecting which rate to use. If CANLink is chosen as the protocol the only baud rate available to use is 250Kbps while a wide range of rates can be chosen for Generic CAN from 20K, 50K, 100K, 125K, 250K, 500K, and 1M. When Generic CAN is selected, another option appears to set up a filter for each channel if so desired.

Coding Within Orchestra 2

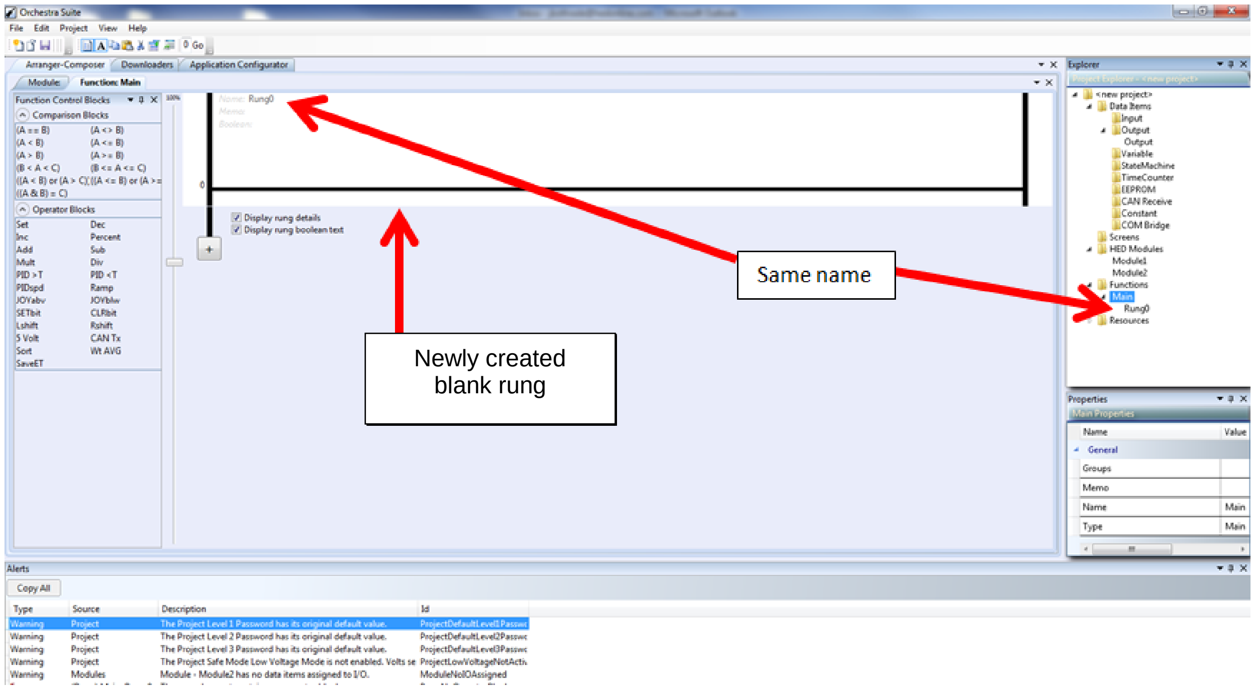
To code within Orchestra, Rungs or Presto with Rungs must be selected as the compile option. Having one of those options selected allows for the Functions folder to be available. The Functions folder is where the Main program will be stored. Double clicking Main will bring up a Function tab within the Designer pane where the ladder logic will be created. Adding rungs is as simple as clicking the large "+" button.



Type	Source	Description	Id
Warning	(Function) Main	The Function "Main" does not contain any rungs.	FunctionNoRungs
Warning	Project	The Project Level 1 Password has its original default value.	ProjectDefaultLevel1Passw
Warning	Project	The Project Level 2 Password has its original default value.	ProjectDefaultLevel2Passw
Warning	Project	The Project Level 3 Password has its original default value.	ProjectDefaultLevel3Passw
Warning	Project	The Project Safe Mode Low Voltage Mode is not enabled. Volts se	ProjectLowVoltageNotActi
Warning	Modules	Module - Module2 has no data items assigned to I/O.	ModuleNoIOAssigned

FIGURE 13: MAIN FUNCTION TAB

When adding rungs, a blank rung is added to the screen with a generic name to it that can be renamed within the Properties pane. That name will appear under Main in the Functions folder which allows for the User to jump to specific rungs at will when needed. After rungs have been added, the User can right click on a rung to bring up options that allows the rung to be copied, cut, pasted, and other rungs to be inserted above or below.



Type	Source	Description	Id
Warning	Project	The Project Level 1 Password has its original default value.	ProjectDefaultLevel1Passw
Warning	Project	The Project Level 2 Password has its original default value.	ProjectDefaultLevel2Passw
Warning	Project	The Project Level 3 Password has its original default value.	ProjectDefaultLevel3Passw
Warning	Project	The Project Safe Mode Low Voltage Mode is not enabled. Volts se	ProjectLowVoltageNotActiv
Warning	Modules	Module - Module2 has no data items assigned to I/O.	ModuleNoIOAssigned

FIGURE 14: BLANK RUNG

Programming ladder logic in Orchestra is as simple as clicking and dragging the Comparison Block(s) and Operator Block(s) to the rung. The Comparison Blocks are grouped together into logic blocks. Each logic block can have up to five Comparison Blocks in each logic block, and each rung can support up to five logic blocks for a max of 25 blocks pending configuration. An easy way to determine supported configurations is to click the small downward arrow box to the left of the logic block. The drop down will display all supported logic variations with that number of comparison blocks within the logic block, and rearrange the blocks automatically once selected.

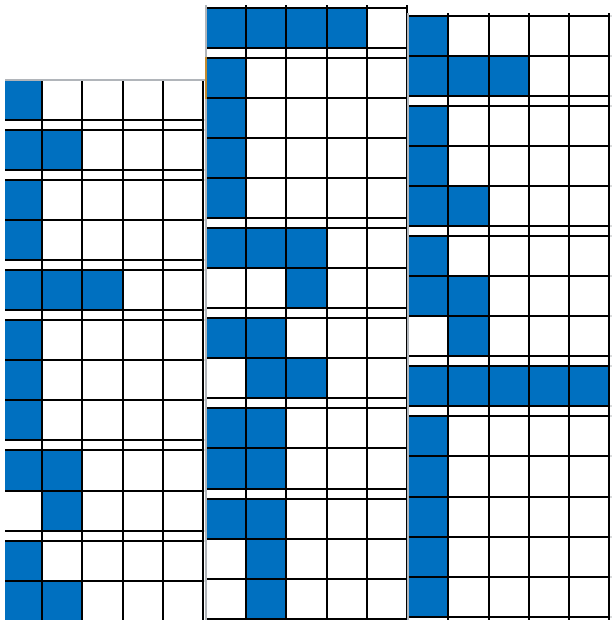
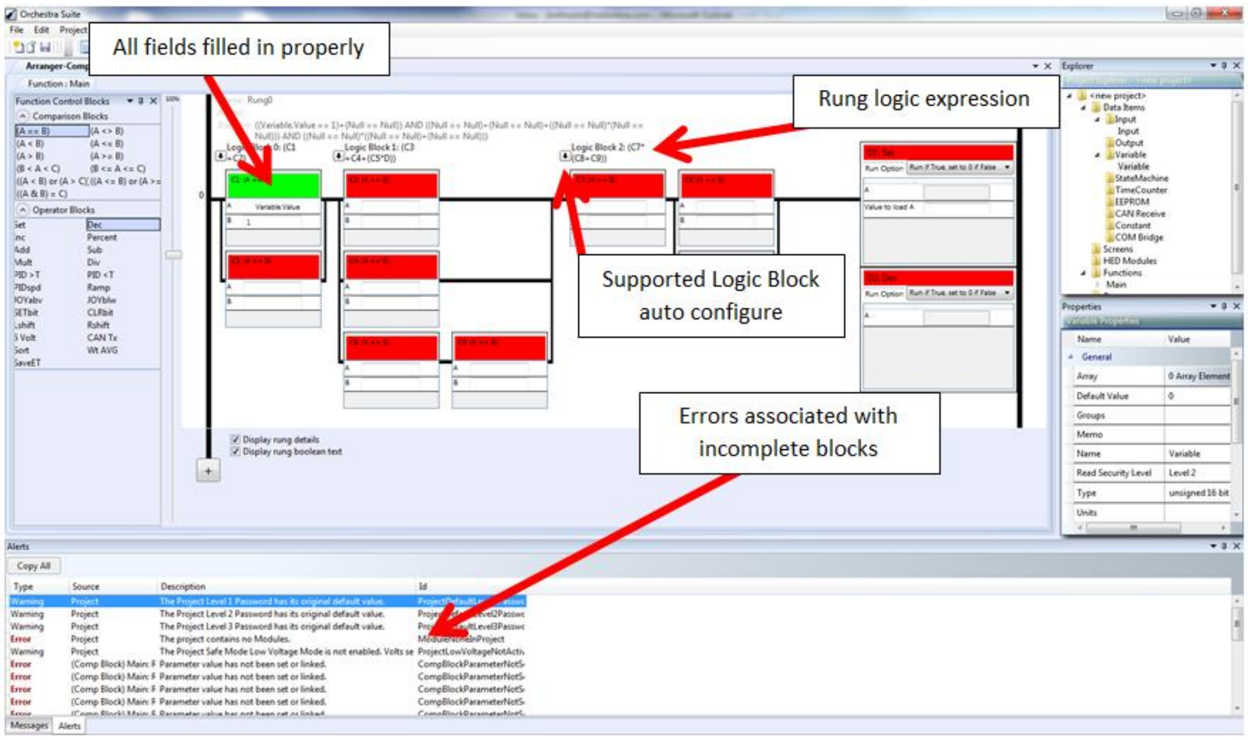


FIGURE 15: SUPPORTED COMPARISON BLOCK CONFIGURATIONS

The rung can also support up to a max of 25 Operator Blocks. Each block has a color band on it, either red or green. The block will remain red and an error will appear in the Alerts pane as long as the block does not have all of the require fields filled in properly. Once each field is filled in properly the block will turn green and the errors will disappear from the Alerts pane.



All fields filled in properly
Rung logic expression
Supported Logic Block auto configure
Errors associated with incomplete blocks

Type	Source	Description	Id
Warning	Project	The Project Level 1 Password has its original default value.	ProjectDefaultL
Warning	Project	The Project Level 2 Password has its original default value.	Proje evel2Passw
Warning	Project	The Project Level 3 Password has its original default value.	Pro ultLevel3Passw
Error	Project	The project contains no Modules.	ModuleNoneInProject
Warning	Project	The Project Safe Mode Low Voltage Mode is not enabled. Volts se	ProjectLowVoltageNotActiv
Error	(Comp Block) Main: F	Parameter value has not been set or linked.	CompBlockParameterNotS
Error	(Comp Block) Main: F	Parameter value has not been set or linked.	CompBlockParameterNotS
Error	(Comp Block) Main: F	Parameter value has not been set or linked.	CompBlockParameterNotS
Error	(Comp Block) Main: F	Parameter value has not been set or linked.	CompBlockParameterNotS

FIGURE 16: RUNG EXAMPLE

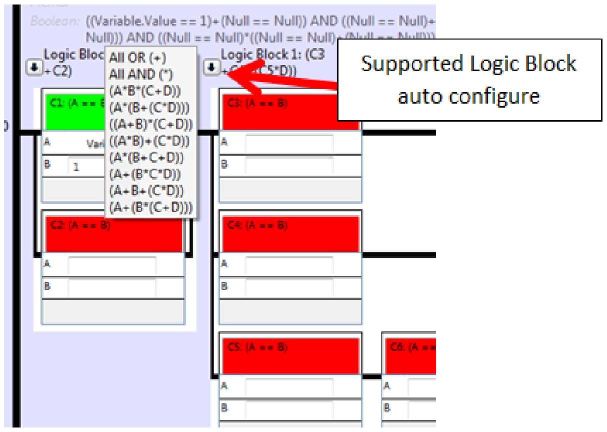
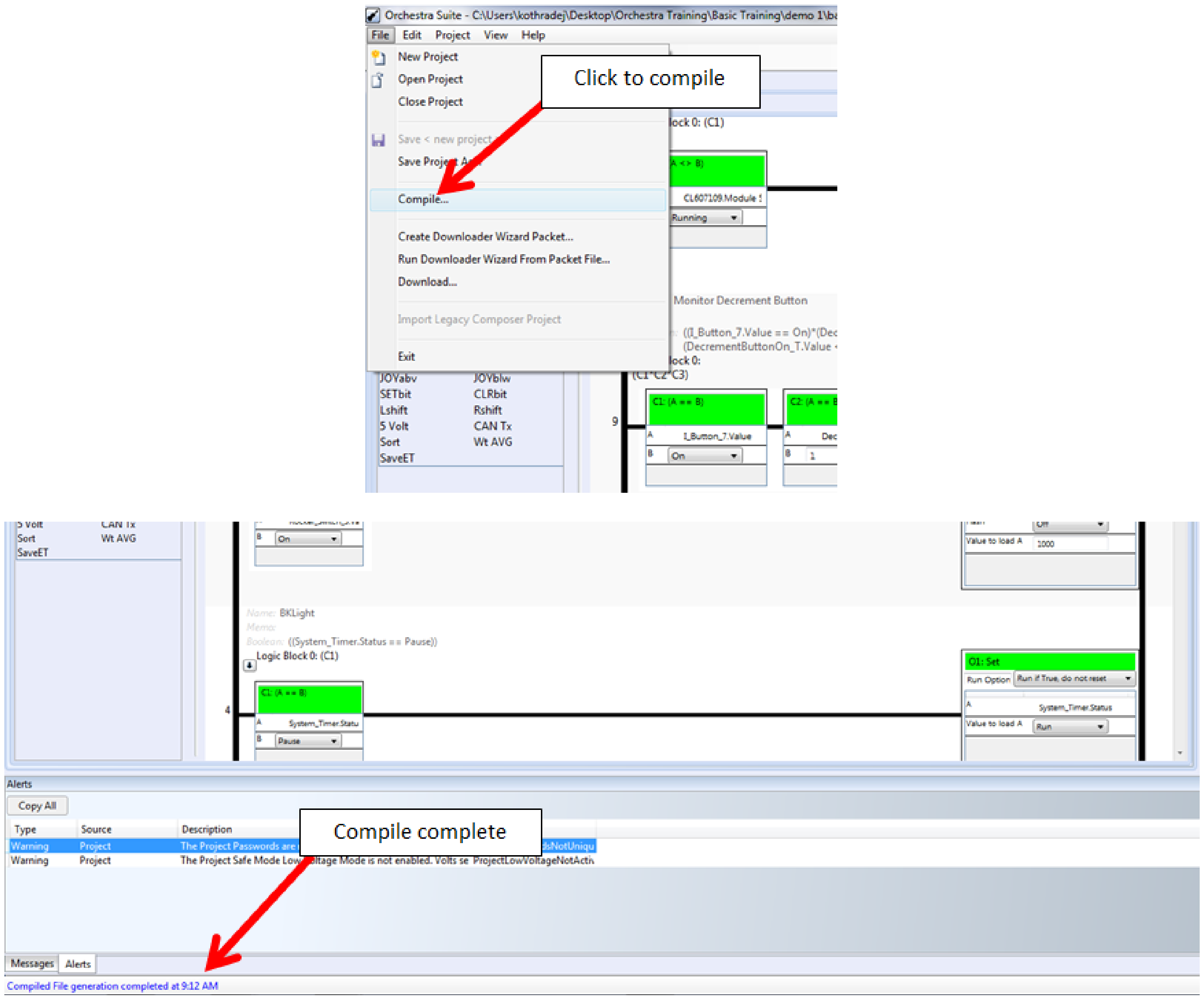


FIGURE 17: SUPPORTED LOGIC BLOCK AUTO CONFIGURE EXAMPLE

How the rung works is based on simple AND/OR logic performed on the Comparison Block(s). Based on the outcome of the logic compares the Operator Block(s) will perform their specific action. The program performs from left to right, top to bottom; meaning the left most Comparison Block will be performed first and the top most Operator Block will operate first. Once Application is written it needs to be compiled before it can be downloaded to the system. To compile the project first needs to be saved by clicking on the File drop down menu and selecting Save. After the Application is saved use the File drop down menu and click the "Compile…" option. In the bottom left hand corner of the screen Orchestra will give the status of the compile.

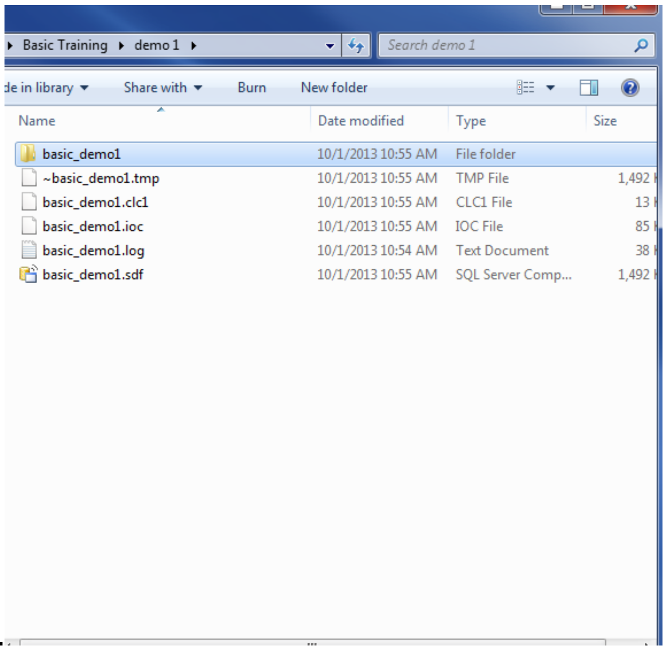


Type	Source	Description
Warning	Project	The Project Passwords are … dsNotUniqu
Warning	Project	The Project Safe Mode Low … ltage Mode is not enabled. Volts se ProjectLowVoltageNotActiv

```
Compiled File generation completed at 9:12 AM
```

FIGURE 18: COMPILING

From the compile a number of files are created within the save directory that the project was saved in.



Name	Date modified	Type	Size
basic_demo1	10/1/2013 10:55 AM	File folder	
~basic_demo1.tmp	10/1/2013 10:55 AM	TMP File	1,492
basic_demo1.clc1	10/1/2013 10:55 AM	CLC1 File	13
basic_demo1.ioc	10/1/2013 10:55 AM	IOC File	85
basic_demo1.log	10/1/2013 10:54 AM	Text Document	38
basic_demo1.sdf	10/1/2013 10:55 AM	SQL Server Comp...	1,492

FIGURE 19: GENERATED FILES

The file folder that bears the same name as your project contains auto-generated files for the Application that includes any fonts, images, display objects, etc. associated with the project. The .clc1 file is a file that pertains to a rung only project, and is what is downloaded to the Module using the Downloader tool. Conductor uses the .ioc file to debug, calibrate, troubleshoot, etc. The .log file contains a record of each time the project was compiled, and the .sdf file is what is needed for Orchestra to open and edit the project. Also, note that when these files are generated the .ioc and .clc1, that is loaded into the Module, must match if the User would like to use Conductor with that particular software on the Module.

Data Items

Input

The Input Data Item holds the incoming information or value from the pin of the Module that it was assigned to. Right clicking the Input subfolder and selecting “Add Input” creates a new Input within the subfolder, and the properties can be viewed and edited within the Properties Pane. There are three groups of properties; General, Digital/Analog, and Scaling. Depending on the Mode of the Input, adjusted using the Input Mode property to define it, the second group will change to Digital or Analog. In doing so, the properties contained under that group will change.
NOTE: PROPERTIES WITHIN ALL DATA ITEMS ARE LINKED AT COMPILE TIME AND CANNOT BE CHANGED IN REAL TIME AS THE CODE IS RUNNING. ANY CHANGES MADE WILL NOT BE SEEN UNTIL THE CODE IS RECOMPILED.

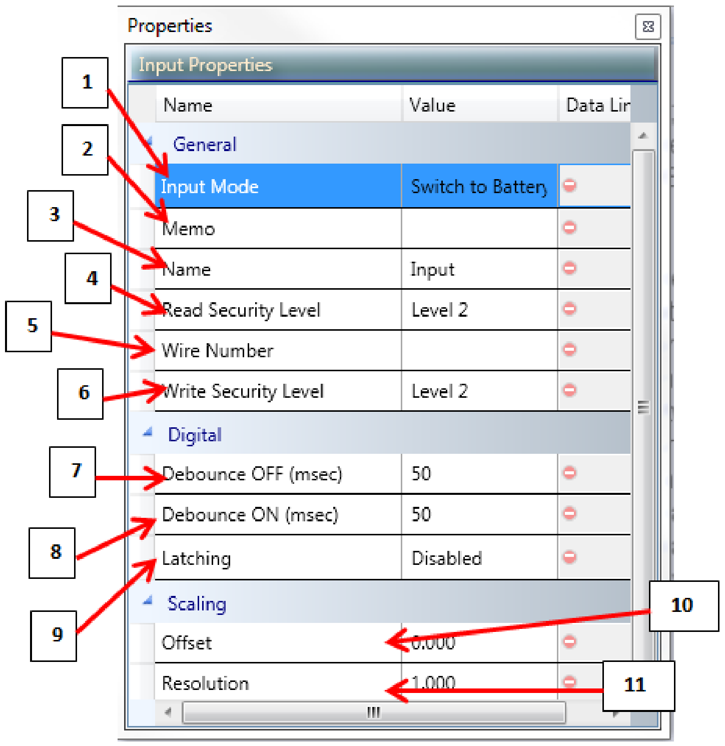
Switch to Battery (STB)

This is one of two digital modes that the Input can be classified as by adjusting the Input Mode property. In this mode the Input is inactive (OFF) when voltage on the pin is below 3.5V, and active (ON) the voltage rises above 7V. There is also a third “unknown” state in which the Input is considered neither active nor open. This unknown state is seen when the Module first initializes, if a communication fault is present, or if the Module is missing. In the event of the Module first powering up, the Input will stay in the unknown state for however long the debounce time within the properties is set to.

Switch to Ground (STG)

STG is the second of the digital modes of the Input Data Item. This mode has the Input inactive when voltage on the pin is above 3.5V and active when the voltage drops below 1.5V. STG has the same digital properties as STB which act the same way; see Switch to Battery for properties.

1. Input Mode
2. Memo
3. Name
4. Read Security Level
5. Wire Number
6. Write Security Level
7. Debounce OFF (msec)
8. Debounce ON (msec)
9. Latching
10. Offset
11. Resolution



Properties

Input Properties

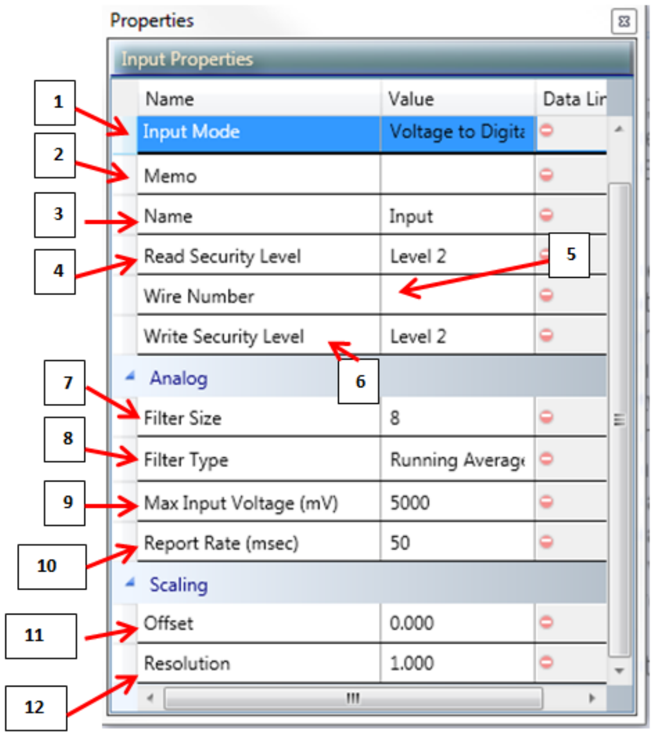
Name	Value	Data Lir
General		
Input Mode	Switch to Battery	⊖
Memo		⊖
Name	Input	⊖
Read Security Level	Level 2	⊖
Wire Number		⊖
Write Security Level	Level 2	⊖
Digital		
Debounce OFF (msec)	50	⊖
Debounce ON (msec)	50	⊖
Latching	Disabled	⊖
Scaling		
Offset	0.000	⊖
Resolution	1.000	⊖

FIGURE 20: STB/STG INPUT PROPERTIES

Voltage to Digital (VTD)

The first analog mode is VTD, and in this mode the Input reads in a voltage in millivolts (mV).

1. Input Mode
2. Memo
3. Name
4. Read Security Level
5. Wire Number
6. Write Security Level
7. Filter Size
8. Filter Type
9. Max Input Voltage (mV)
10. Report Rate (msec)
11. Offset
12. Resolution



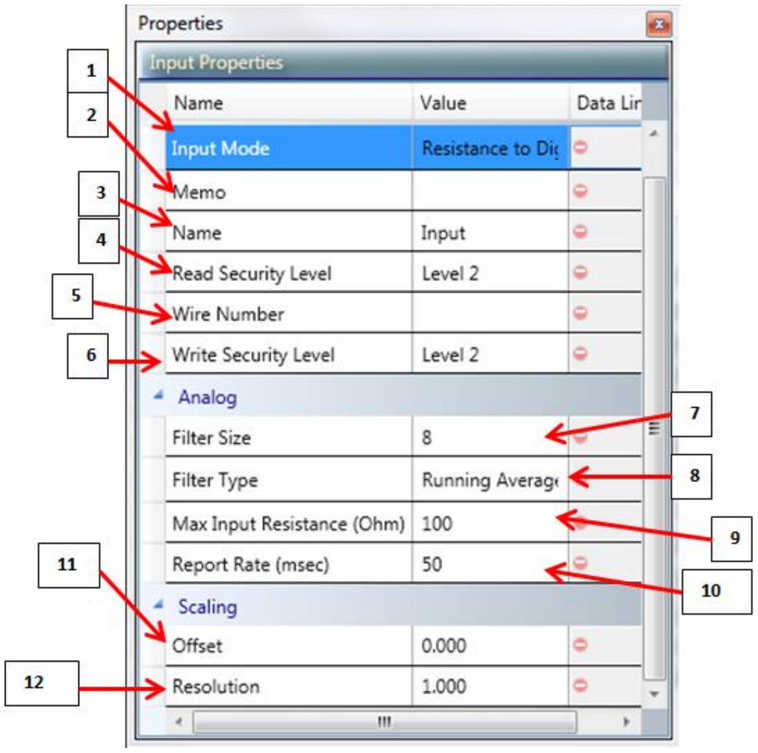
Name	Value	Data Lir
Input Mode	Voltage to Digit	
Memo		
Name	Input	
Read Security Level	Level 2	
Wire Number		
Write Security Level	Level 2	
Analog		
Filter Size	8	
Filter Type	Running Averag	
Max Input Voltage (mV)	5000	
Report Rate (msec)	50	
Scaling		
Offset	0.000	
Resolution	1.000	

FIGURE 21: VTD INPUT PROPERTIES

Resistance to Digital (RTD)

RTD will read in the resistance to ground in Ohms. This Input Mode properties works similarly to the VTD mode with the exception being the User defines the Max Input Resistance, in Ohms, instead of the Max Input Voltage. The rest of the properties work exactly the same.

1. Input Mode
2. Memo
3. Name
4. Read Security Level
5. Wire Number
6. Write Security Level
7. Filter Size
8. Filter Type
9. Max Input Resistance (Ω)
10. Report Rate (msec)
11. Offset
12. Resolution



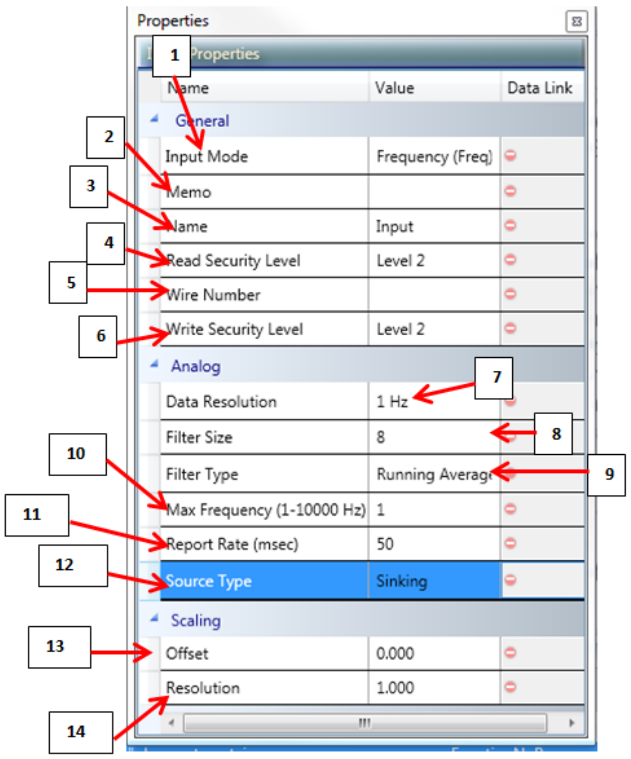
Name	Value	Data Lir
Input Mode	Resistance to Di(	
Memo		
Name	Input	
Read Security Level	Level 2	
Wire Number		
Write Security Level	Level 2	
Analog		
Filter Size	8	
Filter Type	Running Averag(	
Max Input Resistance (Ohm)	100	
Report Rate (msec)	50	
Scaling		
Offset	0.000	
Resolution	1.000	

FIGURE 22: RTD INPUT PROPERTIES

Frequency (Freq)

This mode reports the frequency measured on the Input Signal in Hertz (Hz).

1. Input Mode
2. Memo
3. Name
4. Read Security Level
5. Wire Number
6. Write Security Level
7. Data Resolution
8. Filter Size
9. Filter Type
10. Max Frequency (1-10000Hz)
11. Report Rate (msec)
12. Source Type
13. Offset
14. Resolution



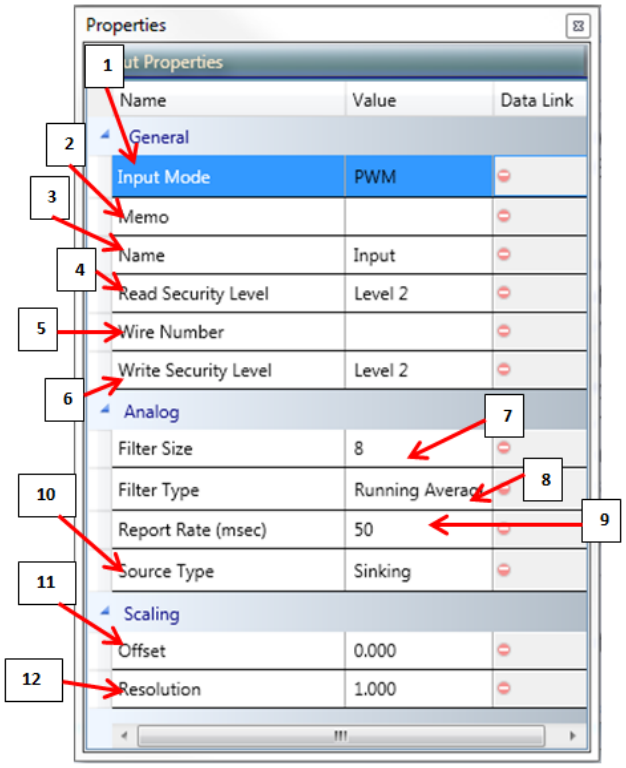
Name	Value	Data Link
General		
Input Mode	Frequency (Freq)	
Memo		
Name	Input	
Read Security Level	Level 2	
Wire Number		
Write Security Level	Level 2	
Analog		
Data Resolution	1 Hz	
Filter Size	8	
Filter Type	Running Average	
Max Frequency (1-10000 Hz)	1	
Report Rate (msec)	50	
Source Type	Sinking	
Scaling		
Offset	0.000	
Resolution	1.000	

FIGURE 23: FREQ INPUT PROPERTIES

Pulse Width Modulation (PWM)

PWM is a modulation technique used to control the width of pulses, usually in terms of controlling the power to an electrical device. The PWM Input Mode measures the duty cycle in 0.1% increments, meaning 0 to 100% is equivalent to values of 0 to 1000 in the Data Item.

1. Input Mode
2. Memo
3. Name
4. Read Security Level
5. Wire Number
6. Write Security Level
7. Filter Size
8. Filter Type
9. Report Rate (msec)
10. Source Type
11. Offset
12. Resolution



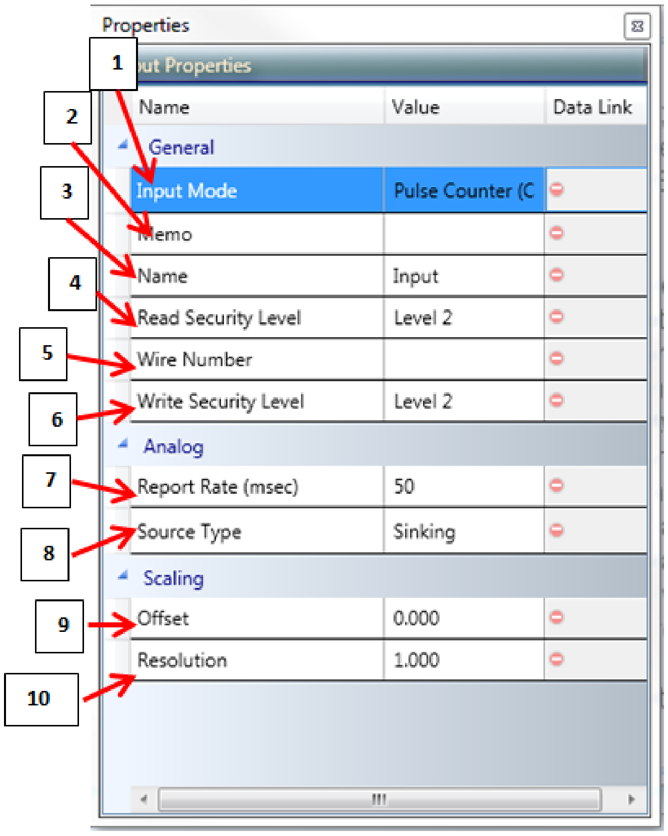
Name	Value	Data Link
General		
Input Mode	PWM	⊖
Memo		⊖
Name	Input	⊖
Read Security Level	Level 2	⊖
Wire Number		⊖
Write Security Level	Level 2	⊖
Analog		
Filter Size	8	⊖
Filter Type	Running Averag	⊖
Report Rate (msec)	50	⊖
Source Type	Sinking	⊖
Scaling		
Offset	0.000	⊖
Resolution	1.000	⊖

FIGURE 24: PWM INPUT PROPERTIES

Pulse Counter (Count)

This mode counts the number of pulses read by the Input.

1. Input Mode
2. Memo
3. Name
4. Read Security Level
5. Wire Number
6. Write Security Level
7. Report Rate (msec)
8. Source Type
9. Offset
10. Resolution



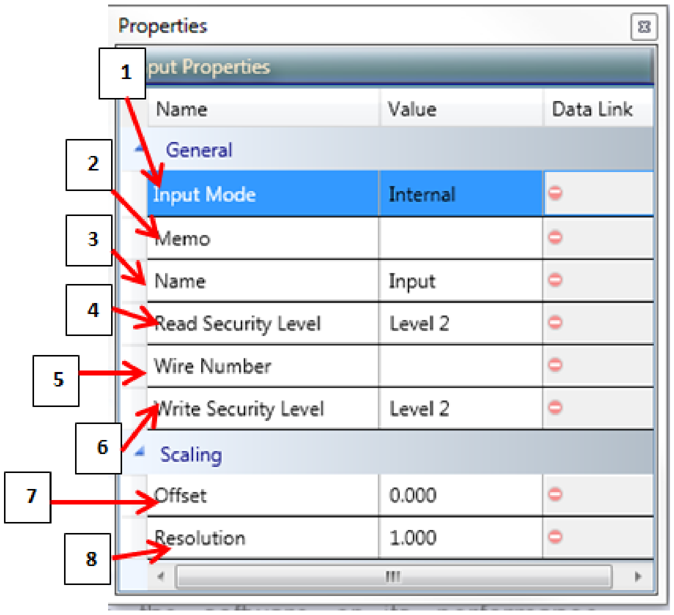
Name	Value	Data Link
General		
Input Mode	Pulse Counter (C	
Memo		
Name	Input	
Read Security Level	Level 2	
Wire Number		
Write Security Level	Level 2	
Analog		
Report Rate (msec)	50	
Source Type	Sinking	
Scaling		
Offset	0.000	
Resolution	1.000	

FIGURE 25: COUNTER INPUT PROPERTIES

Internal

The Internal Input Mode is used to monitor a signal internal to the Module such as a Real Time Clock (RTC), accelerometer, etc.

1. Input Mode
2. Memo
3. Name
4. Read Security Level
5. Wire Number
6. Write Security Level
7. Offset
8. Resolution



Name	Value	Data Link
General		
Input Mode	Internal	
Memo		
Name	Input	
Read Security Level	Level 2	
Wire Number		
Write Security Level	Level 2	
Scaling		
Offset	0.000	
Resolution	1.000	

FIGURE 26: INTERNAL INPUT PROPERTIES

Encoder

The Encoder is actually a combination of the Frequency and Pulse Counter Inputs. For modules that support the Encoder Input, there are Encoder A and Encoder B pins that must be used as a pair. It does not matter the order but one pin must be a Frequency Input and the other a Pulse Counter. The Pulse Counter will essentially become the direction indicator where spinning clockwise will increment the value up to 1000 and spinning counterclockwise will decrement the value. The Frequency Input will indicate how fast the position is changing. For properties please refer to the Frequency and Pulse Counter Input sections.

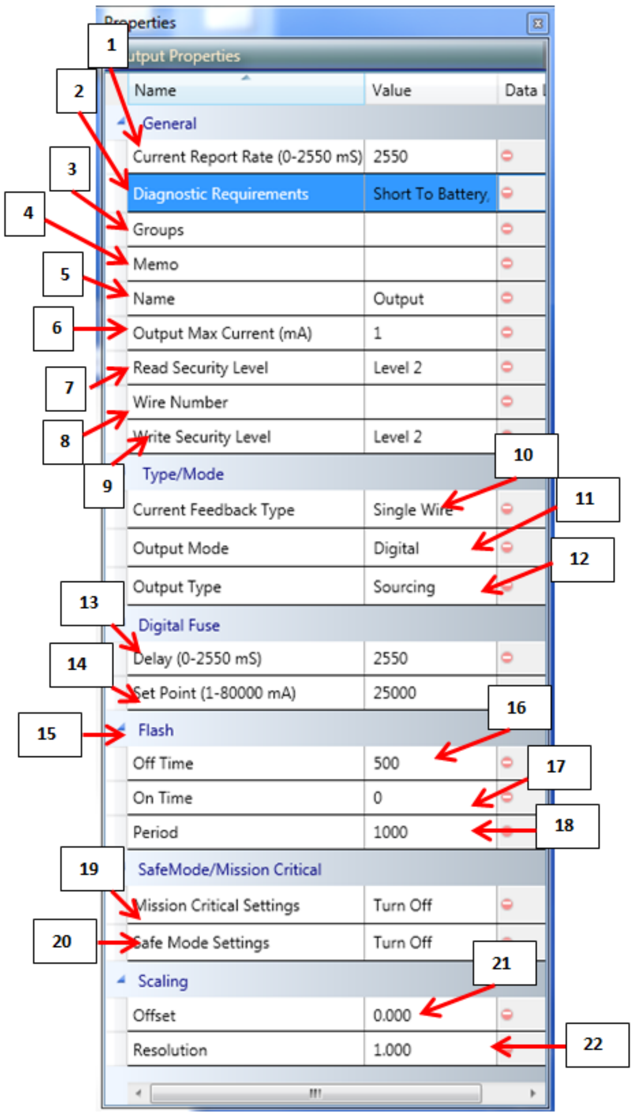
Output

An Output within Orchestra is the end result from using and performing some kind of conditioning or manipulation of another Data Item. The Output will hold a value that the Module will use to perform an action based on how the Application was written. Like the Input Data Item, the Output Data Item has a number of properties and modes that will define the Output such that the Application can react and produce expected results.

Digital

In this mode the Output can be set to be either On or Off.

1. Current Report Rate
2. Diagnostic Requirements
3. Groups
4. Memo
5. Name
6. Output Max Current
7. Read Security Level
8. Wire Number
9. Write Security Level
10. Current Feedback Type
11. Output Mode
12. Output Type
13. Delay (0-2550 mS)
14. Set Point
15. Flash
16. Off Time
17. On Time
18. Period
19. Mission Critical Settings
20. Safe Mode Settings
21. Offset
22. Resolution



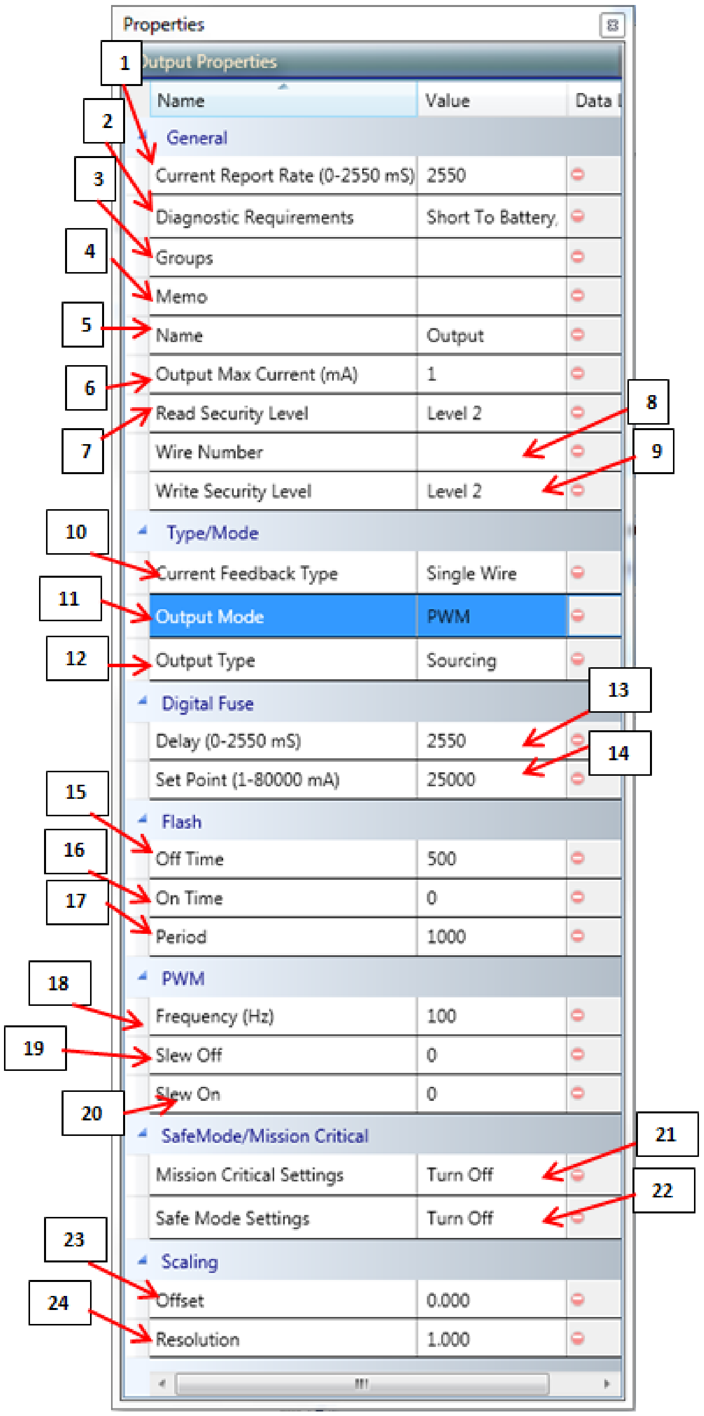
Name	Value	Data
General		
Current Report Rate (0-2550 mS)	2550	
Diagnostic Requirements	Short To Battery,	
Groups		
Memo		
Name	Output	
Output Max Current (mA)	1	
Read Security Level	Level 2	
Wire Number		
Write Security Level	Level 2	
Type/Mode		
Current Feedback Type	Single Wire	
Output Mode	Digital	
Output Type	Sourcing	
Digital Fuse		
Delay (0-2550 mS)	2550	
Set Point (1-80000 mA)	25000	
Flash		
Off Time	500	
On Time	0	
Period	1000	
SafeMode/Mission Critical		
Mission Critical Settings	Turn Off	
Safe Mode Settings	Turn Off	
Scaling		
Offset	0.000	
Resolution	1.000	

FIGURE 27: DIGITAL OUTPUT PROPERTIES

PWM

This mode sets the Output to produce a PWM signal at a frequency defined by the Frequency property between 40 and 5000Hz. Currently because of firmware limitations setting the frequency property below 40Hz will produce undesired results.

1. Current Report Rate
2. Diagnostic Requirements
3. Groups
4. Memo
5. Name
6. Output Max Current
7. Read Security Level
8. Wire Number
9. Write Security Level
10. Current Feedback Type
11. Output Mode
12. Output Type
13. Delay (0-2550 mS)
14. Set Point
15. FlashOff Time
16. On Time
17. Period
18. Frequency (Hz)
19. Slew Off
20. Slew On
21. Mission Critical Settings
22. Safe Mode Settings
23. Offset
24. Resolution



Properties

Output Properties

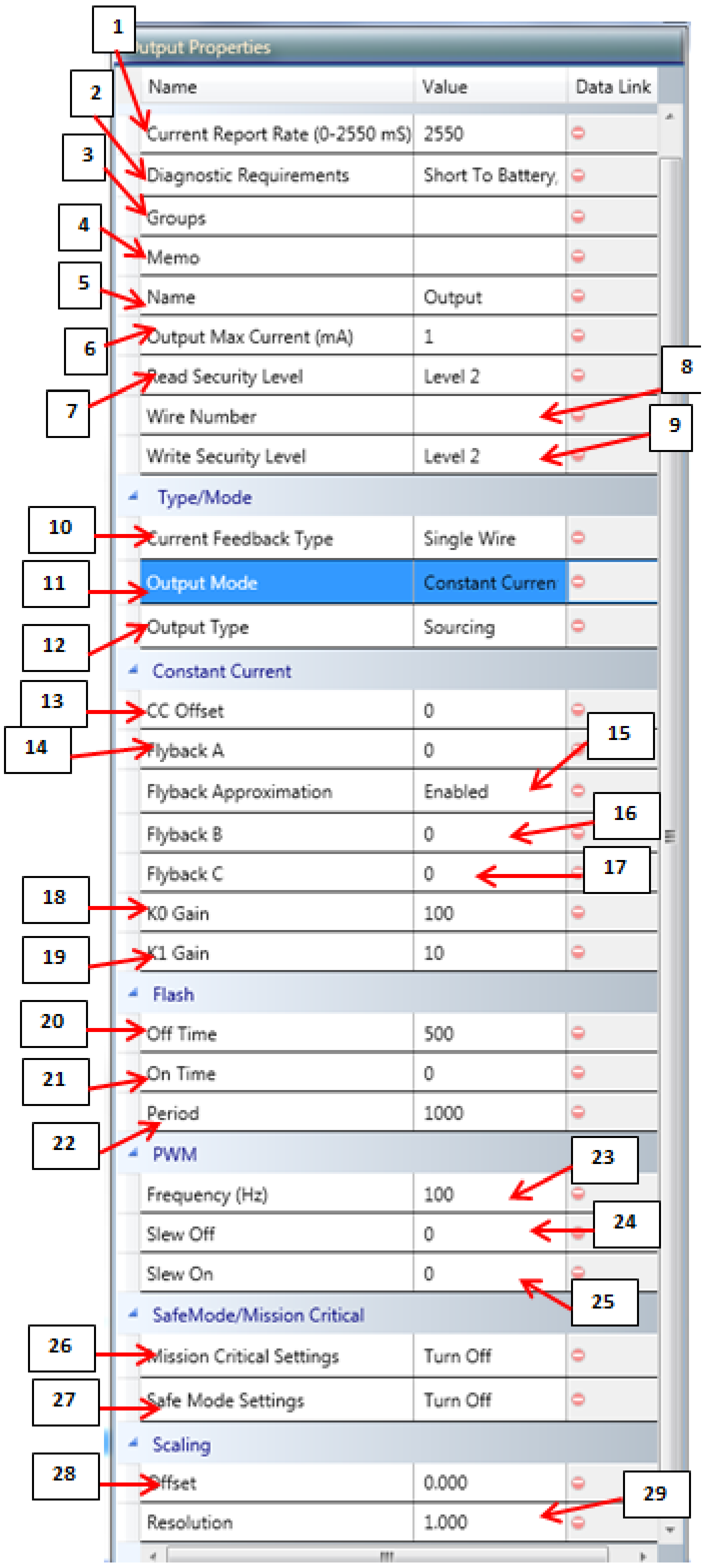
Name	Value
General	
Current Report Rate (0-2550 mS)	2550
Diagnostic Requirements	Short To Battery,
Groups	
Memo	
Name	Output
Output Max Current (mA)	1
Read Security Level	Level 2
Wire Number	
Write Security Level	Level 2
Type/Mode	
Current Feedback Type	Single Wire
Output Mode	PWM
Output Type	Sourcing
Digital Fuse	
Delay (0-2550 mS)	2550
Set Point (1-80000 mA)	25000
Flash	
Off Time	500
On Time	0
Period	1000
PWM	
Frequency (Hz)	100
Slew Off	0
Slew On	0
SafeMode/Mission Critical	
Mission Critical Settings	Turn Off
Safe Mode Settings	Turn Off
Scaling	
Offset	0.000
Resolution	1.000

FIGURE 28: PWM OUTPUT PROPERTIES

Current Controlled (Single Wire)

In this mode the User sets a current the Output should drive and the module will adjust the duty cycle until the current matches the requested value. This is essentially a PWM Output that has a closed loop control to maintain a specific current. The Single Wire version of this Output does not have a feedback line for the current to return, and in order to perform a closed loop current control for the Output additional properties are needed to approximate the return.

1. Current Report Rate
2. Diagnostic Requirements
3. Groups
4. Memo
5. Name
6. Output Max Current
7. Read Security Level
8. Wire Number
9. Write Security Level
10. Current Feedback Type
11. Output Mode
12. Output Type
13. CC Offset
14. Flyback A
15. Flyback Approximation
16. Flyback B
17. Flyback C
18. K0 Gain
19. K1 Gain
20. FlashOff Time
21. On Time
22. Period
23. Frequency (Hz)
24. Slew Off
25. Slew On
26. Mission Critical Settings
27. Safe Mode Settings
28. Offset
29. Resolution



Output Properties

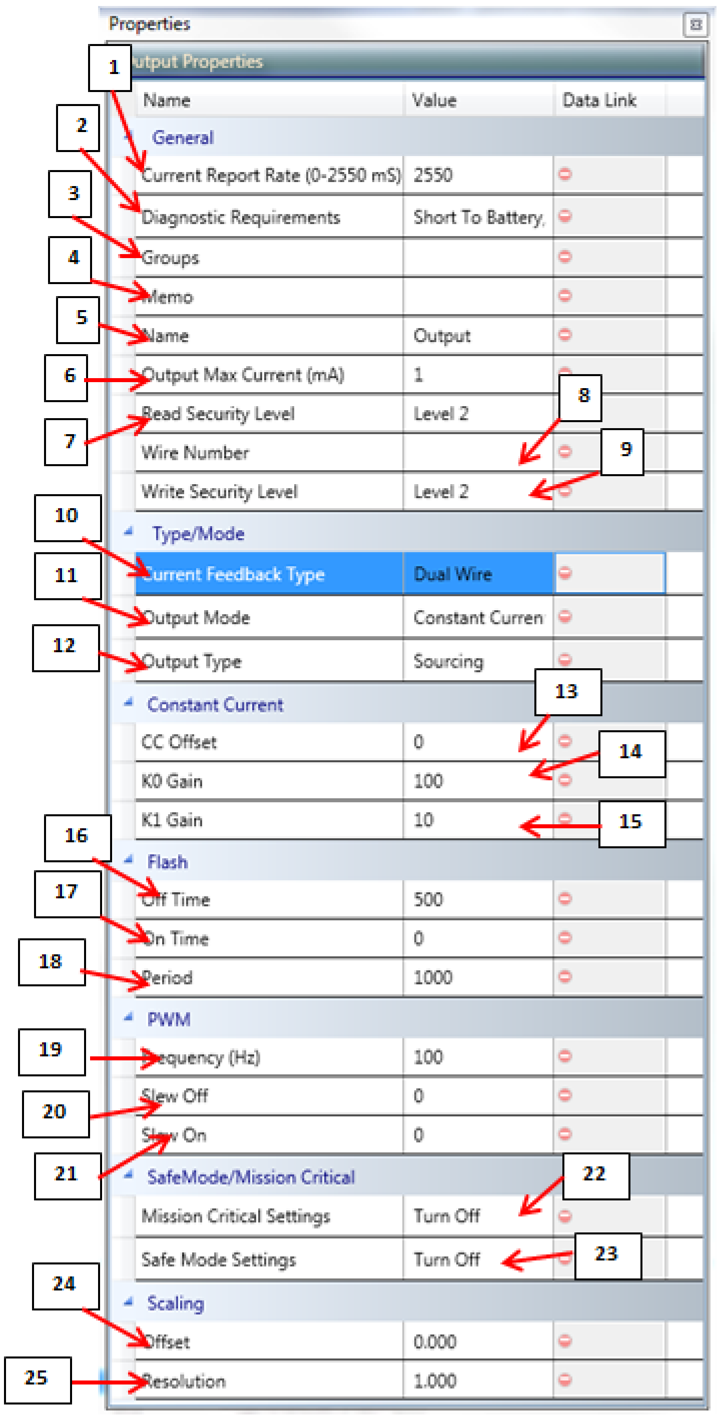
Name	Value	Data Link
Current Report Rate (0-2550 mS)	2550	⊖
Diagnostic Requirements	Short To Battery,	⊖
Groups		⊖
Memo		⊖
Name	Output	⊖
Output Max Current (mA)	1	⊖
Read Security Level	Level 2	⊖
Wire Number		⊖
Write Security Level	Level 2	⊖
Type/Mode		
Current Feedback Type	Single Wire	⊖
Output Mode	Constant Curren	⊖
Output Type	Sourcing	⊖
Constant Current		
CC Offset	0	⊖
Flyback A	0	⊖
Flyback Approximation	Enabled	⊖
Flyback B	0	⊖
Flyback C	0	⊖
K0 Gain	100	⊖
K1 Gain	10	⊖
Flash		
Off Time	500	⊖
On Time	0	⊖
Period	1000	⊖
PWM		
Frequency (Hz)	100	⊖
Slew Off	0	⊖
Slew On	0	⊖
SafeMode/Mission Critical		
Mission Critical Settings	Turn Off	⊖
Safe Mode Settings	Turn Off	⊖
Scaling		
Offset	0.000	⊖
Resolution	1.000	⊖

FIGURE 29: CURRENT CONTROLLED SINGLE WIRE OUTPUT PROPERTIES

Current Controlled (Dual Wire)

In this mode the User sets a current the Output should drive and the module will adjust the duty cycle until the current matches the requested value. This is essentially a PWM Output that has a closed loop control to maintain a specific current. The Dual Wire version of this Output has a feedback line for the current to return, and does not need the extra properties to approximate the return since it is measured directly.

1. Current Report Rate
2. Diagnostic Requirements
3. Groups
4. Memo
5. Name
6. Output Max Current
7. Read Security Level
8. Wire Number
9. Write Security Level
10. Current Feedback Type
11. Output Mode
12. Output Type
13. CC Offset
14. K0 Gain
15. K1 Gain
16. FlashOff Time
17. On Time
18. Period
19. Frequency (Hz)
20. Slew Off
21. Slew On
22. Mission Critical Settings
23. Safe Mode Settings
24. Offset
25. Resolution



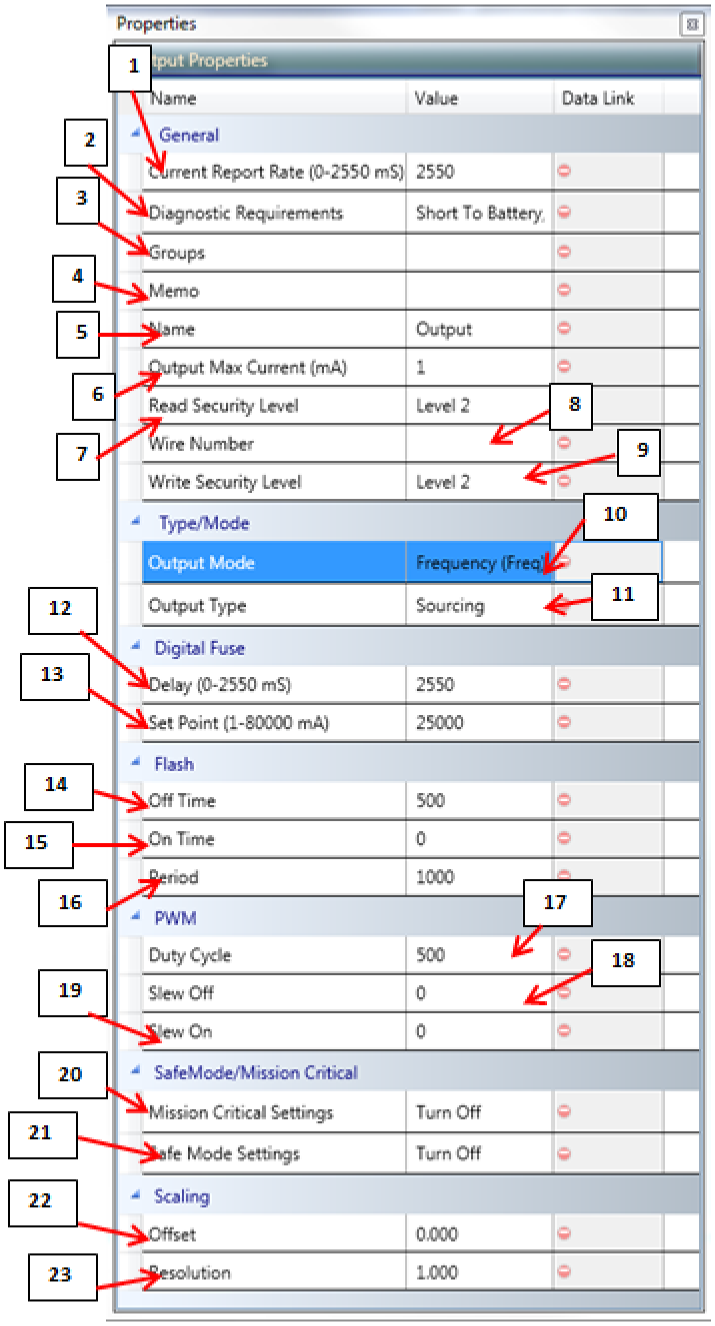
Name	Value	Data Link
General		
Current Report Rate (0-2550 mS)	2550	⊖
Diagnostic Requirements	Short To Battery,	⊖
Groups		⊖
Memo		⊖
Name	Output	⊖
Output Max Current (mA)	1	⊖
Read Security Level	Level 2	⊖
Wire Number		⊖
Write Security Level	Level 2	⊖
Type/Mode		
Current Feedback Type	Dual Wire	⊖
Output Mode	Constant Curren	⊖
Output Type	Sourcing	⊖
Constant Current		
CC Offset	0	⊖
K0 Gain	100	⊖
K1 Gain	10	⊖
Flash		
Off Time	500	⊖
On Time	0	⊖
Period	1000	⊖
PWM		
Frequency (Hz)	100	⊖
Slew Off	0	⊖
Slew On	0	⊖
SafeMode/Mission Critical		
Mission Critical Settings	Turn Off	⊖
Safe Mode Settings	Turn Off	⊖
Scaling		
Offset	0.000	⊖
Resolution	1.000	⊖

FIGURE 30: CURRENT CONTROLLED DUAL WIRE OUTPUT PROPERTIES

Frequency

The Output has the ability to produce a variable frequency (duty cycle constant). The duty cycle for the frequency can be adjusted using the Duty Cycle property to a value between 100 and 900 corresponding to 10% and 90%. Like the PWM Output, the Slew On and Off rates can be adjusted to control how fast the frequency will ramp up the set point and back down to zero.

1. Current Report Rate
2. Diagnostic Requirements
3. Groups
4. Memo
5. Name
6. Output Max Current
7. Read Security Level
8. Wire Number
9. Write Security Level
10. Output Mode
11. Output Type
12. Delay (0-2550 mS)
13. Set Point (1-80000 mA)
14. FlashOff Time
15. On Time
16. Period
17. Frequency (Hz)
18. Slew Off
19. Slew On
20. Mission Critical Settings
21. Safe Mode Settings
22. Offset
23. Resolution



Name	Value	Data Link
General		
Current Report Rate (0-2550 mS)	2550	⊖
Diagnostic Requirements	Short To Battery,	⊖
Groups		⊖
Memo		⊖
Name	Output	⊖
Output Max Current (mA)	1	⊖
Read Security Level	Level 2	
Wire Number		⊖
Write Security Level	Level 2	⊖
Type/Mode		
Output Mode	Frequency (Freq)	⊖
Output Type	Sourcing	
Digital Fuse		
Delay (0-2550 mS)	2550	⊖
Set Point (1-80000 mA)	25000	⊖
Flash		
Off Time	500	⊖
On Time	0	⊖
Period	1000	⊖
PWM		
Duty Cycle	500	⊖
Slew Off	0	⊖
Slew On	0	⊖
SafeMode/Mission Critical		
Mission Critical Settings	Turn Off	⊖
Safe Mode Settings	Turn Off	⊖
Scaling		
Offset	0.000	⊖
Resolution	1.000	⊖

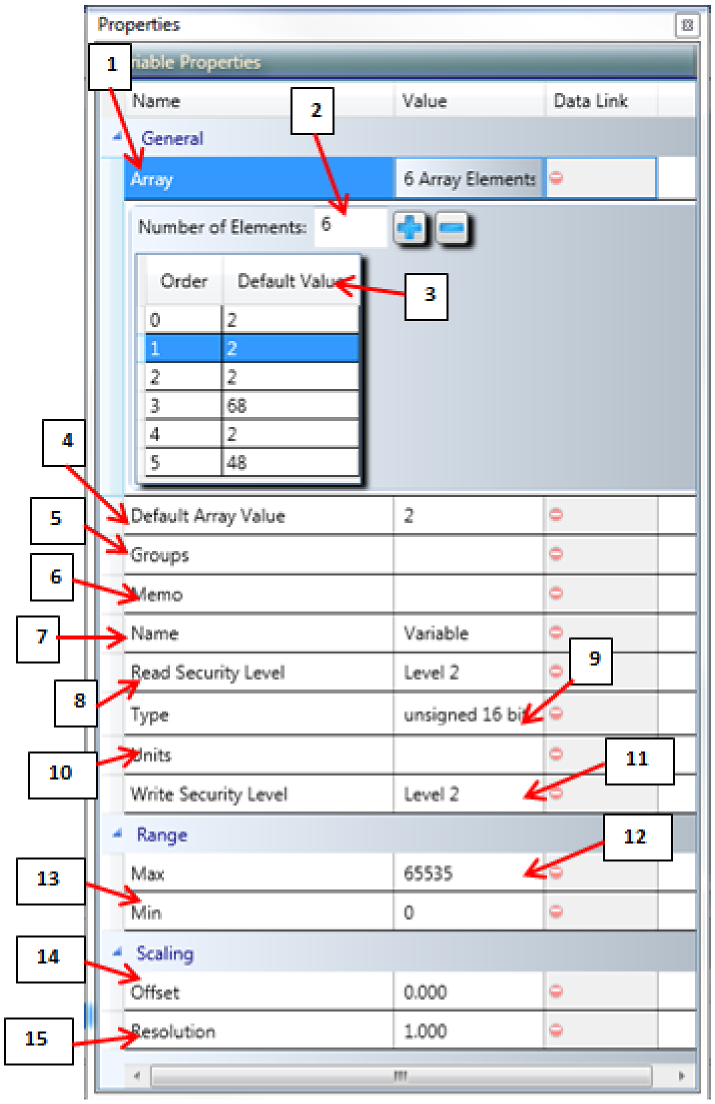
FIGURE 31: FREQUENCY OUTPUT PROPERTIES

As a note, the PWM and Constant Current have two extra Output Types to choose from, PVG and EDC respectively. Those two settings are hardware specific settings that are used when WAG electronics are interfacing with a specific competitor’s hydraulic equipment.

Variable

The Variable Data Item is used to save temporary and calculated values as a single value or as an array with multiple values. The first thing that should be done after creating a Variable is defining the Max and Min values within the Range portion of the Property pane; with the Max value being limited by the Type of Variable. The Variable can be defined as an unsigned 16 bit (0-65,535), unsigned 32 bit (0-4,294,967,295, and an alarm.
After defining the Max and Min values, the size of the array can be chosen by left clicking on the Array property and either typing in the size of the array desired or by clicking the "+" button. An array with size zero will be a Variable that can hold a single value. As the array size increases, the User can define the Default Values of each element in the array using the small table that appears below the "Number of Elements" in the Property Pane. The Default Array Value property below the Array property sets a global default for the array, so each additional element added will be initialized to that global value.

1. Array
2. Number of Elements
3. Default Value
4. Default Array Value
5. Groups
6. Memo
7. Name
8. Read Security Level
9. Type
10. Units
11. Write Security Level
12. Max
13. Min
14. Offset
15. Resolution



Properties

iable Properties

Name	Value	Data Link
General		
Array	6 Array Elements	
Number of Elements: 6		

Order	Default Value
0	2
1	2
2	2
3	68
4	2
5	48

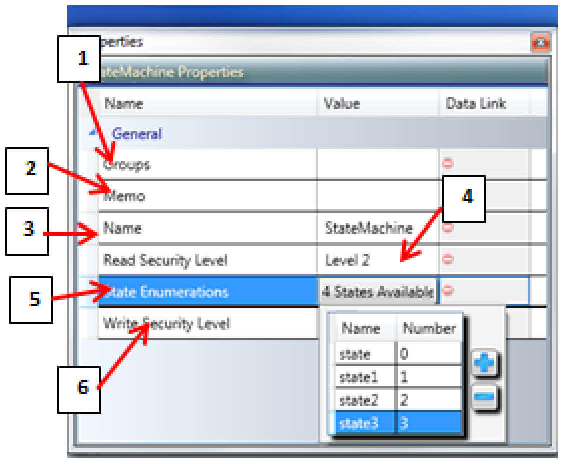
Name	Value	Data Link
Default Array Value	2	
Groups		
Memo		
Name	Variable	
Read Security Level	Level 2	
Type	unsigned 16 bit	
Units		
Write Security Level	Level 2	
Range		
Max	65535	
Min	0	
Scaling		
Offset	0.000	
Resolution	1.000	

FIGURE 32: VARIABLE PROPERTIES

State Machine

The State Machine Data Item is a special type of Variable the largest difference being that the value(s) of the State Machine only get updated at the end of each loop. A State Enumerations property allows the User to define the number of states available, as well as, define the names and numbers for those states. Each state must have a unique number and name, Orchestra will not allow the User to enter duplicate states. To add a state, left click the State Enumerations value and then click the "+" button to increase the number of states and the "-" button to decrease the number of states. Editing the state name and number is done by just left clicking on the number or name; once it is highlighted the User can type in a new name or number assignment.

1. Groups
2. Memo
3. Name
4. Read Security Level
5. State Enumerations
6. Write Security Level



Name	Value	Data Link
Groups		
Memo		
Name	StateMachine	
Read Security Level	Level 2	
State Enumerations	4 States Available	
Write Security Level		

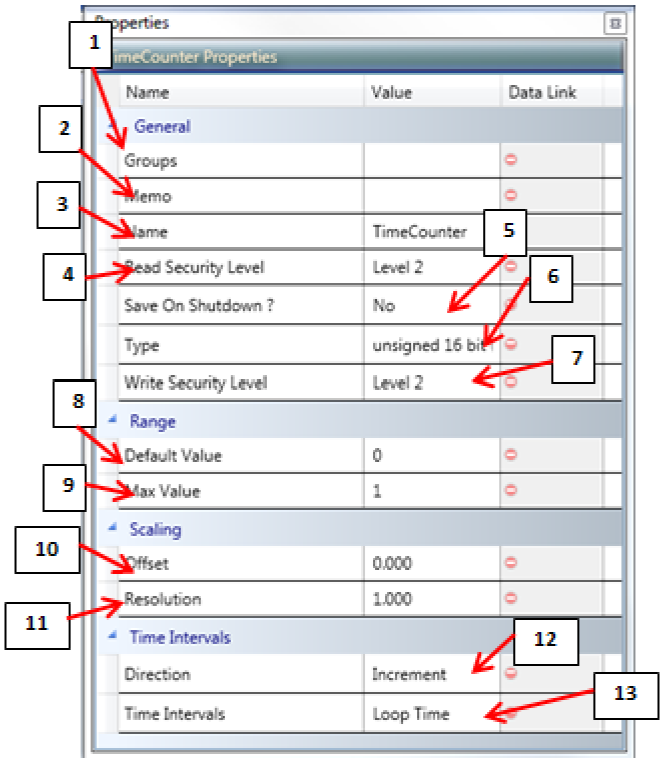
Name	Number
state	0
state1	1
state2	2
state3	3

FIGURE 33: STATE MACHINE PROPERTIES

Time Counter

Time Counters are Data Items that increment or decrement a set number of times within a defined time frame. Note that there is no Min Value to set, it is always zero. The counter will decrement or increment once per Time Interval so the User should keep in mind that the total amount of time it will take the counter to fully increment or decrement is a product of the Time Interval and the value that the User sets. Also, in order to properly implement the timer within Rung Logic the Time Counter's sub state must be set to Run since the default state of the timer is Paused, this is also true within C code if using an Orchestra Time Counter.

1. Groups
2. Memo
3. Name
4. Read Security Level
5. Save On Shutdown?
6. Type
7. Write Security Level
8. Default Value
9. Max Value
10. Offset
11. Resolution
12. Direction
13. Time Interval



Properties

TimeCounter Properties

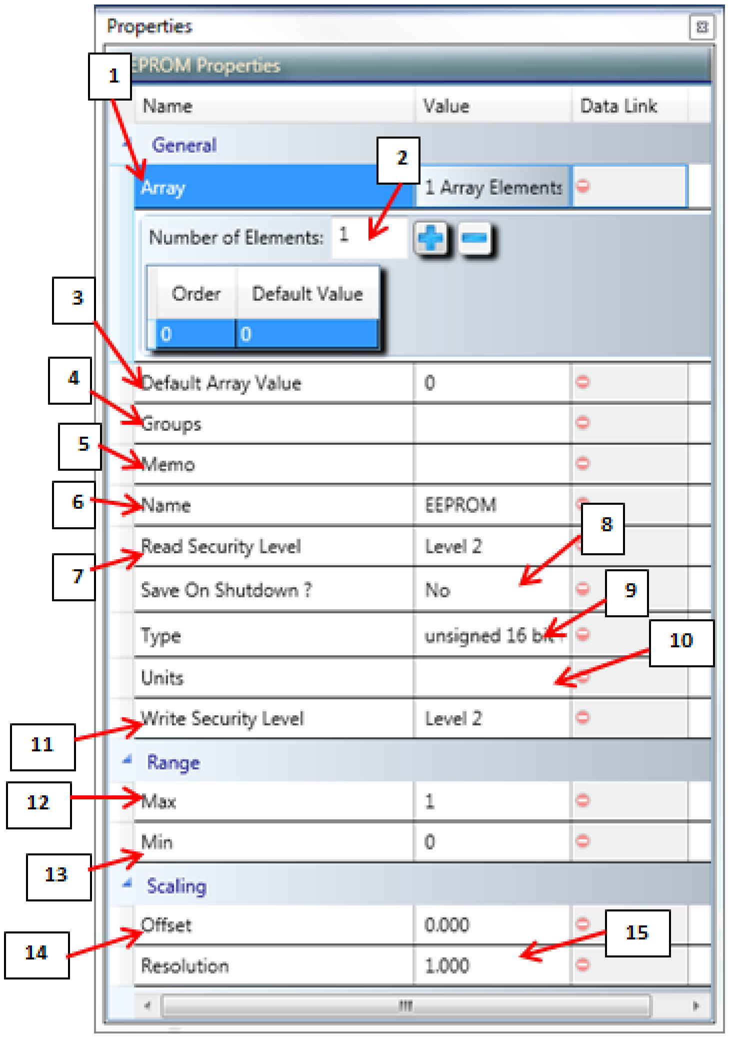
Name	Value	Data Link
General		
Groups		
Memo		
Name	TimeCounter	
Read Security Level	Level 2	
Save On Shutdown ?	No	
Type	unsigned 16 bit	
Write Security Level	Level 2	
Range		
Default Value	0	
Max Value	1	
Scaling		
Offset	0.000	
Resolution	1.000	
Time Intervals		
Direction	Increment	
Time Intervals	Loop Time	

FIGURE 34: TIME COUNTER PROPERTIES

EEPROM

The EEPROM Data Item is very similar to a Timer and a Variable, property wise. This Data Item is useful for allowing the Customer or End User some flexibility in their Application to adjust the values of other Data Items by using the EEPROM. The values from EEPROM get read and placed into variables before any rungs within Orchestra are processed. When writing to the EEPROM the program will actually write to the variable location, in order to save the EEPROM values the option to save on shutdown within the Properties pane must be changed to Yes.

1. Array
2. Number of Elements
3. Default Value
4. Default Array Value
5. Groups
6. Memo
7. Name
8. Read Security Level
9. Type
10. Units
11. Write Security Level
12. Max
13. Min
14. Offset
15. Resolution

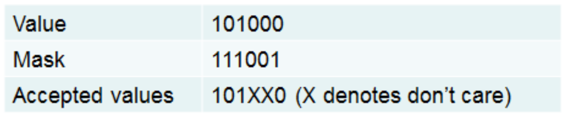


Name	Value	Data Link
General		
Array	1 Array Elements	
Number of Elements: 1		
Order	Default Value	
0	0	
Default Array Value	0	
Groups		
Memo		
Name	EEPROM	
Read Security Level	Level 2	
Save On Shutdown ?	No	
Type	unsigned 16 bit	
Units		
Write Security Level	Level 2	
Range		
Max	1	
Min	0	
Scaling		
Offset	0.000	
Resolution	1.000	

FIGURE 35: EEPROM PROPERTIES

CAN Receive

CAN Receive is used to read in a piece of data from received CAN messages off of the CAN bus and then sets a status to 1 (Received) from a 0 (Clear) each time that message is received. Up to 32 bits can be read per CAN Receive, so to read in an entire message it may take multiple CAN Receives.



Value	101000
Mask	111001
Accepted values	101XX0 (X denotes don't care)

FIGURE 36: MASK EXAMPLE

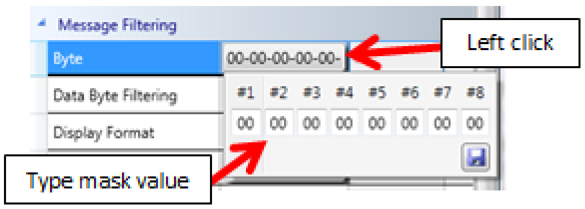
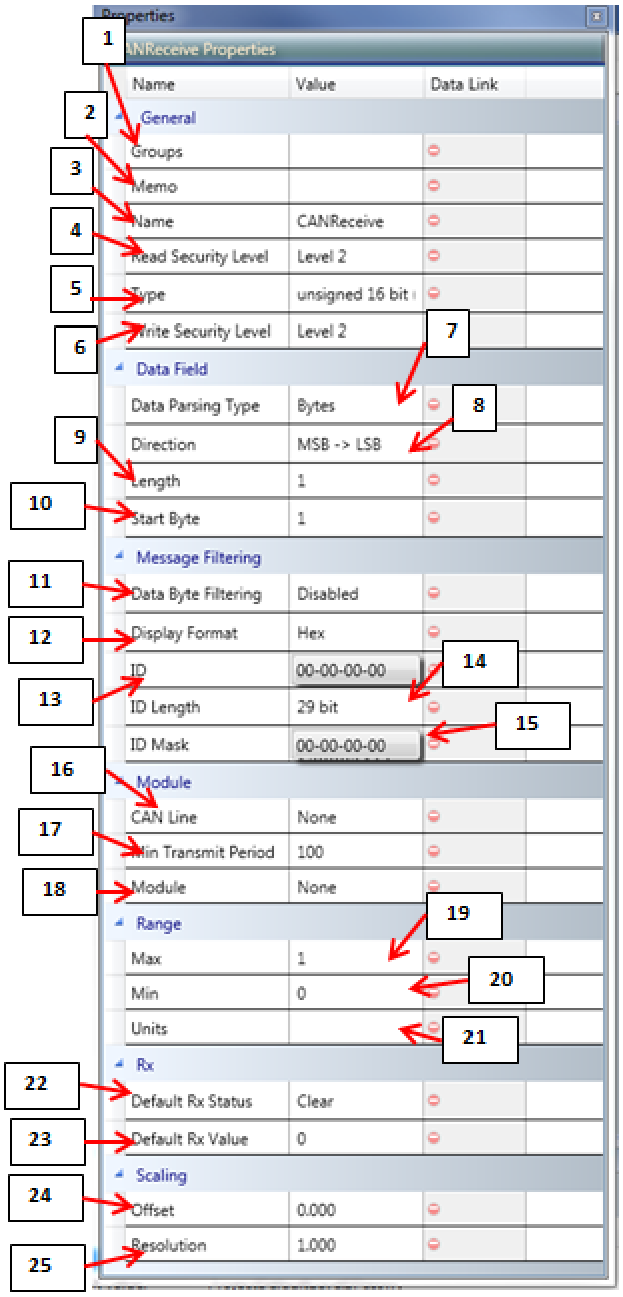


FIGURE 37: MASK BIT SELECTION

1. Groups
2. Memo
3. Name
4. Read Security Level
5. Type
6. Write Security Level
7. Data Parsing Type
8. Direction CAN
9. Length
10. Start Byte
11. Data Byte Filtering
12. Display Format
13. ID
14. ID Length
15. ID Mask
16. CAN Line
17. Min Transmit Period
18. Module
19. Max
20. Min
21. Units
22. Default Rx Status
23. Default Rx Value
24. Offset
25. Resolution



Properties

ANReceive Properties

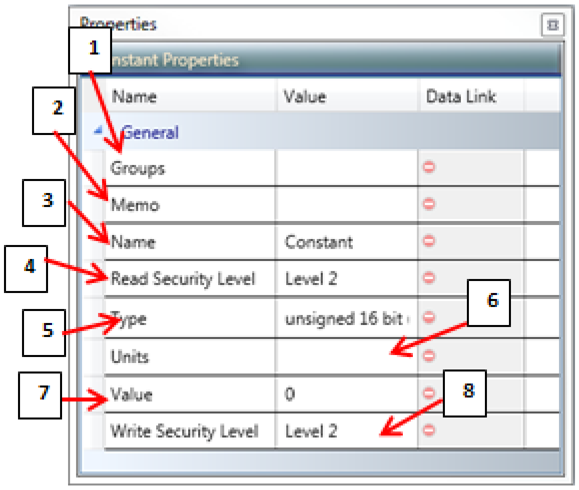
Name	Value	Data Link
General		
Groups		
Memo		
Name	CANReceive	
Read Security Level	Level 2	
Type	unsigned 16 bit	
Write Security Level	Level 2	
Data Field		
Data Parsing Type	Bytes	
Direction	MSB -> LSB	
Length	1	
Start Byte	1	
Message Filtering		
Data Byte Filtering	Disabled	
Display Format	Hex	
ID	00-00-00-00	
ID Length	29 bit	
ID Mask	00-00-00-00	
Module		
CAN Line	None	
Min Transmit Period	100	
Module	None	
Range		
Max	1	
Min	0	
Units		
Rx		
Default Rx Status	Clear	
Default Rx Value	0	
Scaling		
Offset	0.000	
Resolution	1.000	

FIGURE 38: CAN RECEIVE PROPERTIES

Constant

The Constant Data Item is a static value defined by the User through the Properties.

1. Groups
2. Memo
3. Name
4. Read Security Level
5. Type
6. Units
7. Value
8. Write Security Level



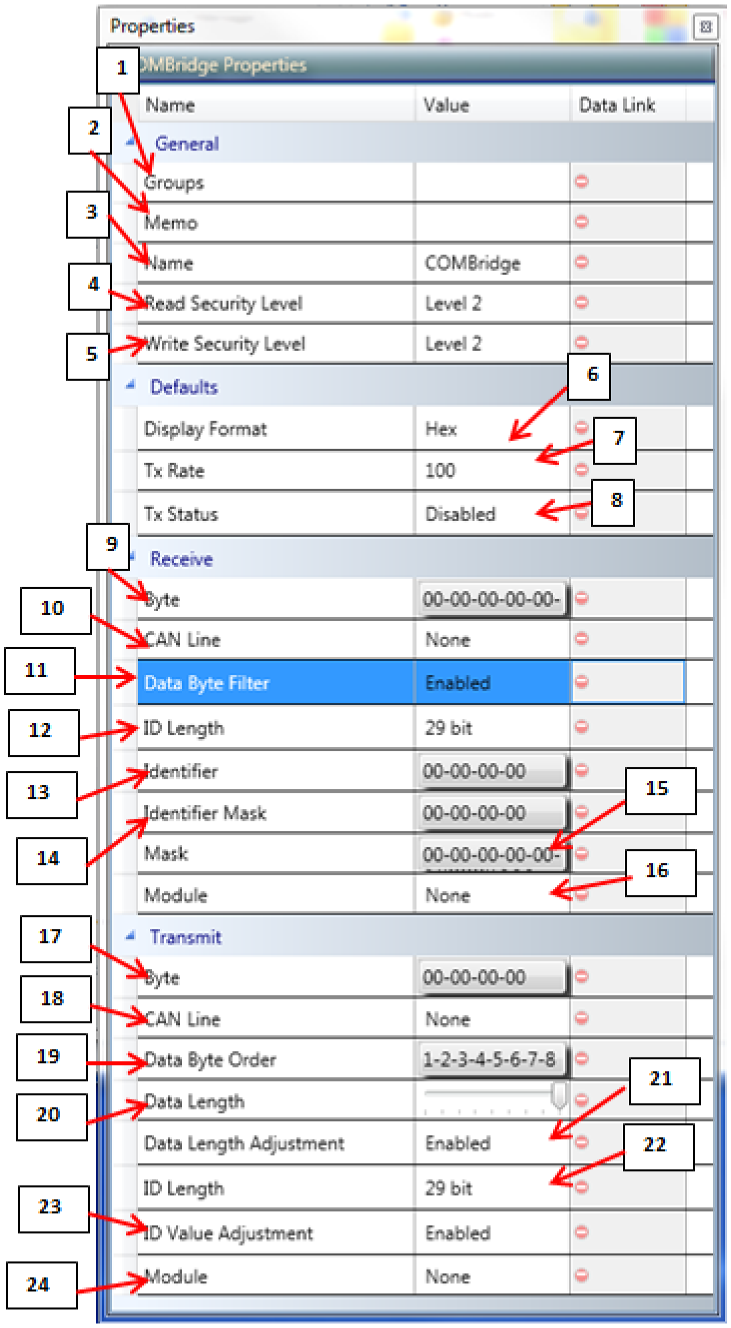
Name	Value	Data Link
General		
Groups		⊖
Memo		⊖
Name	Constant	⊖
Read Security Level	Level 2	⊖
Type	unsigned 16 bit	⊖
Units		⊖
Value	0	⊖
Write Security Level	Level 2	⊖

FIGURE 39: CONSTANT PROPERTIES

COM Bridge

A COM Bridge Data Item is used to pass some or all CAN messages based on an Identifier. This is useful for acting as a CAN filter, a CAN Bus extender for those longer than 40 meters, or even as a way to reorder the messages coming through. The COM Bridge is limited to passing only messages with a similar Identifier, so if there are multiple messages with different Identifiers a COM Bridge would have to be created for each of those messages that are desired.

1. Groups
2. Memo
3. Name
4. Read Security Level
5. Write Security Level
6. Display Format
7. Tx Rate
8. Tx Status
9. Byte
10. CAN Line
11. Data Byte Filtering
12. ID Length
13. Identifier
14. Identifier Mask
15. Mask
16. Module
17. Byte
18. CAN Line
19. Data Byte Order
20. Data Length
21. Data Length Adjustment
22. ID Length
23. ID Value Adjustment
24. Module
26. CAN Line
27. Min Transmit Period
28. Module
29. Max
30. Min
31. Units
32. Default Rx Status
33. Default Rx Value
34. Offset



OMBridge Properties

Name	Value	Data Link
General		
Groups		⊖
Memo		⊖
Name	COMBridge	⊖
Read Security Level	Level 2	⊖
Write Security Level	Level 2	⊖
Defaults		
Display Format	Hex	⊖
Tx Rate	100	⊖
Tx Status	Disabled	⊖
Receive		
Byte	00-00-00-00-00-	⊖
CAN Line	None	⊖
Data Byte Filter	Enabled	⊖
ID Length	29 bit	⊖
Identifier	00-00-00-00	⊖
Identifier Mask	00-00-00-00	⊖
Mask	00-00-00-00-00-	⊖
Module	None	⊖
Transmit		
Byte	00-00-00-00	⊖
CAN Line	None	⊖
Data Byte Order	1-2-3-4-5-6-7-8	⊖
Data Length		⊖
Data Length Adjustment	Enabled	⊖
ID Length	29 bit	⊖
ID Value Adjustment	Enabled	⊖
Module	None	⊖

FIGURE 40: COM BRIDGE PROPERTIES

Comparison Blocks

Orchestra uses two kinds of blocks in its rung logic, Comparison Blocks and Operator Blocks. The Comparison blocks are what are used to create the logic for the rung program, while the Operator Blocks perform some kind of action whether or not the Comparison Block logic is true or false. These Comparison Blocks have two or three data fields, “A”, “B”, or “C”, that the User can fill in. The User can either fill those fields in with their own numbers or use Data Items, so Data Items and numbers can be compared to other Data Items or numbers.
As discussed within the Data Items section, a number of Data Items have some extra properties, besides just a value of a number, such as states, indexes within arrays, statuses, etc. that define the Data Item. To tell what property of the Data Item is being used in the comparison Orchestra attaches a “.Value” or “.Status” extension to the Data Item name within the Comparison Block.
Those properties can be used for comparisons within Comparison Blocks, and to access them for the block to use, take the mouse pointer and hover over the Data Item after it has been linked to a data field within the block, and a small downward pointing arrow should appear to the left of the Data Item name. Clicking that arrow will drop down a small menu for the User to choose from. Most items only have “Value” within that drop down, but if it has anything else such as a “Status” it will be shown within that menu. Data Items that have the ability to become arrays will have a second drop down menu that will allow the User to select a specific element within the array to do the comparison on, and the selected element will be indicated within the Data Item name in the Comparison Block. All Data Items will have a red “X” next to those menus that will remove the selected Data Item from the Comparison Block.
The User should ensure that the values used within the Comparison Blocks fall within the range of the Data Item or defined value, otherwise the block may never become true and could potentially prevent that particular rung from ever executing. As a note, purposely creating a condition in which the Comparison Block is never true can be a simple way of commenting out a rung within the program, and can be useful for troubleshooting.

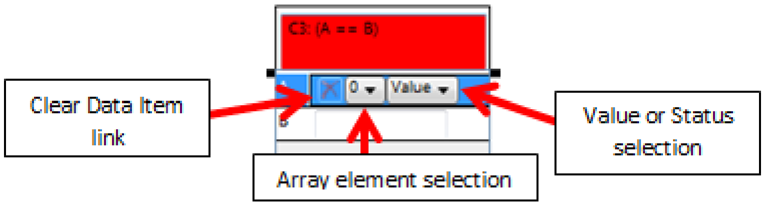


FIGURE 41: COMPARISON BLOCK PROPERTIES EXAMPLE

Equals (A==B)

The Equals Comparison Block performs a logic check to determine whether or not the two values, “A” and “B”, in the block are equal or not.

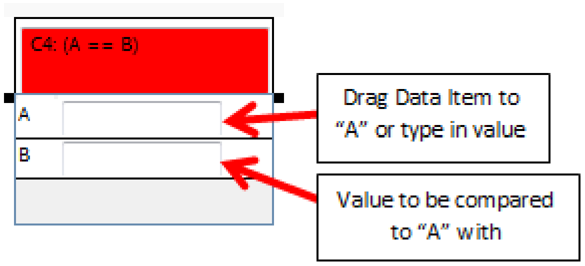

C4: (A == B)
A
B
Drag Data Item to “A” or type in value
Value to be compared to “A” with

FIGURE 42: EQUALS COMPARISON BLOCK

Less Than (A<B)

This block compares the value of what is in “A” to what is in “B”, and returns a true if the current value or status of the “A” is less than the compared value or status in “B”.

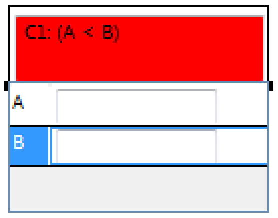


FIGURE 43: LESS THAN COMPARISON BLOCK

Greater Than (A>B)

This block is true if the value of the Data Item in “A” is greater than the defined value of “B”.

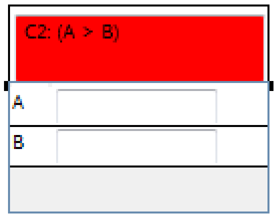


FIGURE 44: GREATER THAN COMPARISON BLOCK

Not Equal (A<>B)

As long as the item in “A” does not equal the value in “B”, this block will be true.

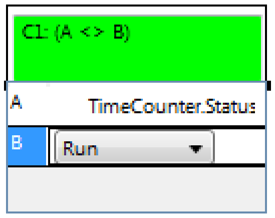


FIGURE 45: NOT EQUAL COMPARISON BLOCK

Less Than or Equal To (A<=B)

This block works the same way as the Less Than block except that the value to cause the block to return a true can include the value in “B”. The value of A must be greater than “B” for this block to be false.

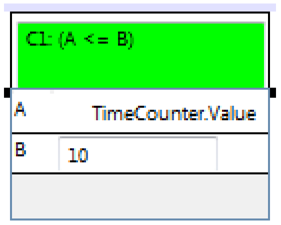


FIGURE 46: LESS THAN OR EQUAL TO COMPARISON BLOCK

Greater Than or Equal To (A>=B)

This block works just like the Greater Than block, but the range of values that causes the block to return as true includes the value designated in “B”. Any value less than the value in “B” will cause this block to return a false.

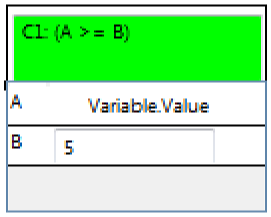


FIGURE 47: GREATER THAN OR EQUAL TO COMPARISON BLOCK

Greater Than, Less Than (B<A<C)

The Greater Than, Less Than block checks to see if the value of “A” falls in between the values set by “B” and “C”, and returns true if it does.

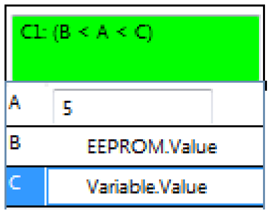


FIGURE 48: GREATER THAN, LESS THAN COMPARISON BLOCK

Greater Than or Equal To, Less Than or Equal To (B<=A<=C)

This block works by comparing the value in “A” to the other two values in “B” and “C”, and returns a true if it falls between them or equals either of them.

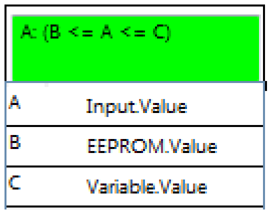


FIGURE 49: GREATER THAN OR EQUAL TO, LESS THAN OR EQUAL TO COMPARISON BLOCK

Less Than OR Greater Than ((A<B) or (A>C))

The value of “A” is compared to both the values of “B” and “C”, and if either “A” is less than “B” or “A” is greater than “C” this block returns true.

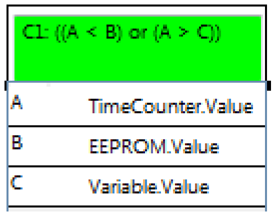


FIGURE 50: LESS THAN OR GREATER THAN COMPARISON BLOCK

Less Than or Equal To OR Greater Than or Equal To ((A<=B) or (A>=C))

If "A" is either greater than or equal to the value in "C" or is less than or equal to the value in "B", this block returns a true.

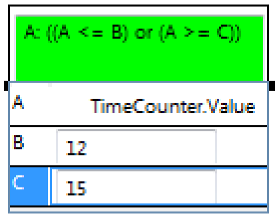


FIGURE 51: LESS THAN OR EQUAL TO OR GREATER THAN OR EQUAL TO COMPARISON BLOCK

AND Equal To ((A && B) = C)

This block performs a bit-wise AND operation between the value in "A" and the value in "B" then compares the result to "C". If the result is equal to the value of "C" the block returns a true.

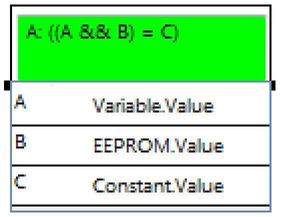


FIGURE 52: AND EQUAL TO COMPARISON BLOCK

Operator Blocks

Operator Blocks appear on the right hand side of the rung and perform a specific action if all of the logic from the Comparison Blocks is true. The Operator Blocks can accept a combination of Data Items, predefined values from drop down menus, and User defined values within the data fields where permissible. Some areas can only accept Data Items, those areas can be identified by being unable to click on them and have a cursor appear to input User values. Data Items also function in the Operator Blocks similarly to the way they function within the Comparison Blocks in that particular properties or portions of the Data Item can be selected to perform the operation on.
Each Operator Block, with the exception of three, has a Run Option field at the top of the block that contains four possible options; Run if True set to 0 if False, Run if True do not reset, Run if False set to 0 if True, and Run if False do not reset. The three that do not have that option will be explained within their respective sections following. The Run if True set to 0 if False option will allow that Operator Block to only execute if the rung logic is true and if the rung logic is false that Operator Block will produce a value of zero. Run if True do not reset will execute the operation if the rung is true and whatever the outcome of the operation is, that value will be held until that Data Item or User defined value is operated upon again. The Run if False set to 0 if True and Run if False do not reset work in the same manner as their Run if True counterparts.

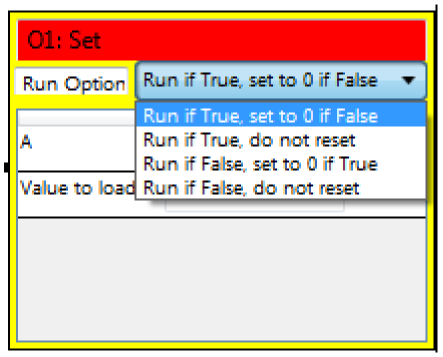
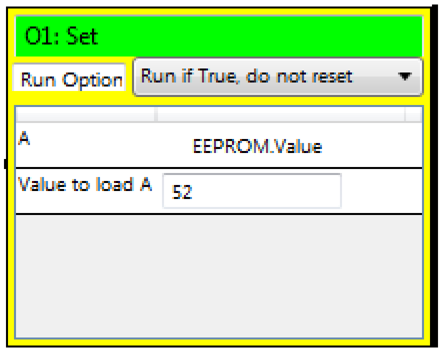

O1: Set
Run Option
Run if True, set to 0 if False
Run if True, set to 0 if False
Run if True, do not reset
Run if False, set to 0 if True
Run if False, do not reset
A
Value to load

FIGURE 53: OPERATOR BLOCK RUN OPTIONS

Set

The Set Operator Block requires a Data Item in "A" while the "Value to load A" can accept a Data Item or a User defined value. If the logic in the run is true and depending on what the run option is for the Set block, the value of the Data Item in "A" will change to the "Value to load A".



Run Option: Run if True, do not reset

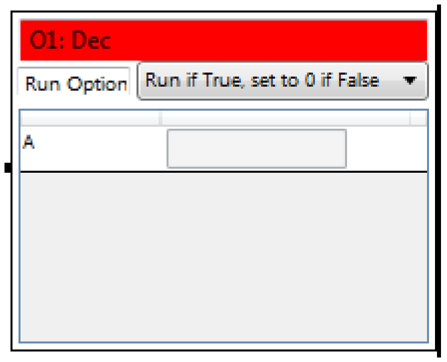
A: EEPROM.Value

Value to load A: 52

FIGURE 54: SET OPERATOR BLOCK

Dec

The Data Item that gets assigned to "A" in the Dec block will have its value decremented by one whenever the rung logic satisfies the run option selected for the Operator Block. The decrement will only occur once for each time the rung is true, so to decrement multiple times the rung must transition from true to false and back to true.



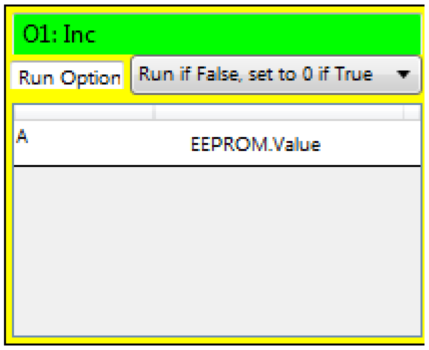
Run Option: Run if True, set to 0 if False

A

FIGURE 55: DEC OPERATOR BLOCK

Inc

The Inc block will increment the value of the Data Item in “A” by one if the rung logic fulfills the run option requirement chosen. Similar to the Dec block, this operation will only occur once for each transition to a true state from a false state.



Run Option: Run if False, set to 0 if True

A	EEPROM.Value

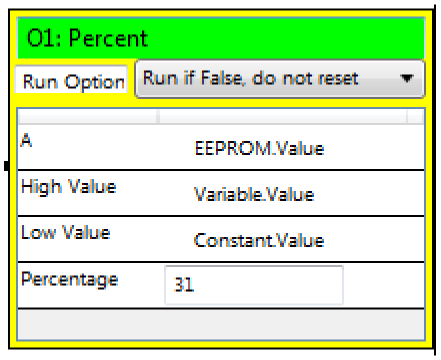
FIGURE 56: INC OPERATOR BLOCK

Percent

The Percent Operator Block will return a value based on a User or Data Item specified percentage over a specific range of values whenever the run option for the block is met, using the following formula:

$$A = (High\,Value - Low\,Value) \times Percentage + Low\,Value$$

EQUATION 1: PERCENT OPERATOR BLOCK EQUATION



Run Option: Run if False, do not reset

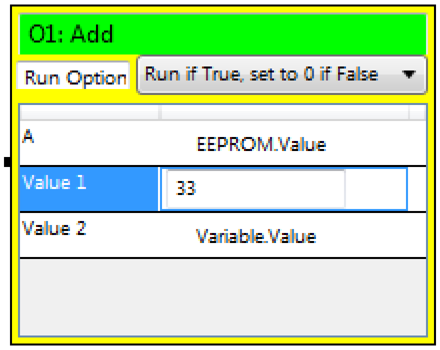
A	EEPROM.Value
High Value	Variable.Value
Low Value	Constant.Value
Percentage	31

FIGURE 57: PERCENT OPERATOR BLOCK

Note that within Orchestra percentages are from 0 to 1000 where 1000 is equal to 100%.

Add

The Add block will take two values and add them together and place the result in the Data Item specified in “A” each time the rung meets the run option criteria. If the criteria is true each time the program loops the addition will take place.



A	EEPROM.Value
Value 1	33
Value 2	Variable.Value

FIGURE 58: ADD OPERATOR BLOCK

$$A = (Value\,1 + Value\,2)$$

EQUATION 2: ADD OPERATOR BLOCK EQUATION

Sub

The Sub block will subtract “Value 2” from “Value 1” and place the result in “A” whenever the run option criterion is met. Just like the Add block, if this rung is held true then each time the program loops a subtraction will occur.

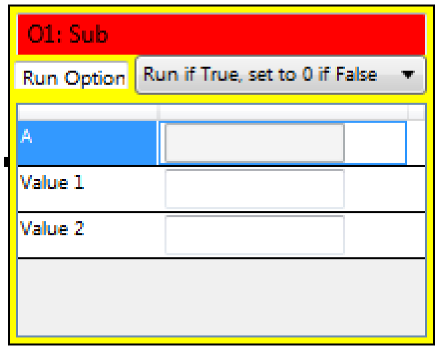


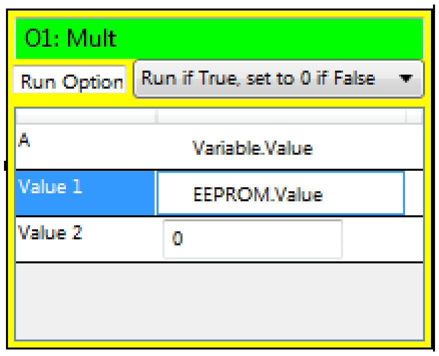
FIGURE 59: SUB OPERATOR BLOCK

$$A = (Value\,1 - Value\,2)$$

EQUATION 3: SUB OPERATOR BLOCK EQUATION

Mult

Mult will multiply “Value 1” and “Value 2” together, and place the resulting value in the Data Item assigned to “A” whenever the rung logic and the run option of the block coincide. This block will continue to execute each time the program loops as long as the rung is held true.


O1: Mult
Run Option: Run if True, set to 0 if False

A	Variable.Value
Value 1	EEPROM.Value
Value 2	0

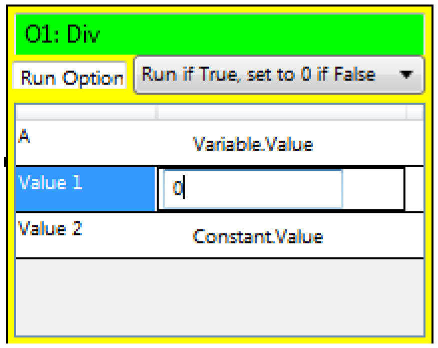
FIGURE 60: MULT OPERATOR BLOCK

$$A = (Value\,1 \times Value\,2)$$

EQUATION 4: MULT OPERATOR BLOCK EQUATION

Div

This block will divide the value in “Value 1” by the value in “Value 2” then place the resulting value in “A” if the run option is met. This block will continue to execute each time the program loops if the rung is held true.


O1: Div
Run Option: Run if True, set to 0 if False

A	Variable.Value
Value 1	0
Value 2	Constant.Value

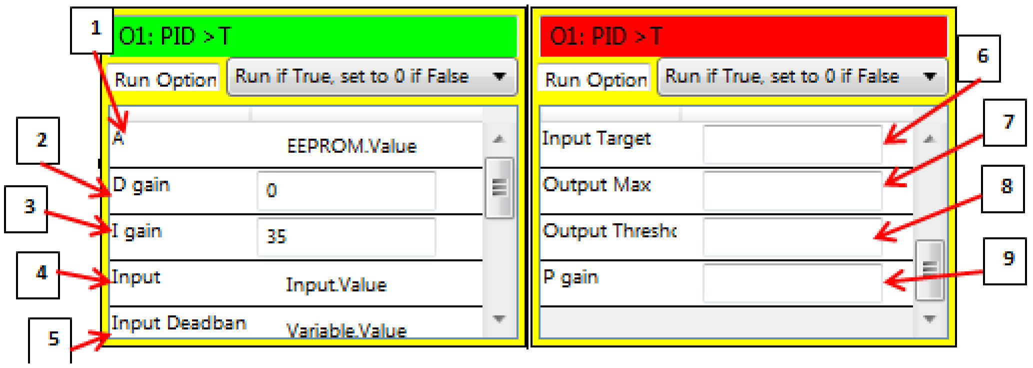
FIGURE 61: DIV OPERATOR BLOCK

$$A = (Value\,1 \div Value\,2)$$

EQUATION 5: DIV OPERATOR BLOCK EQUATION

PID>T

This block is used to provide some closed loop control for an output using the error correction from a PID operation as long as target value is less than the input value, i.e. the lowering of a robotic arm on a refuse vehicle to its resting position after being raised.


1
2
3
4
5
6
7
8
9
O1: PID > T
Run Option
Run if True, set to 0 if False
O1: PID > T
Run Option
Run if True, set to 0 if False

A	EEPROM.Value
D gain	0
I gain	35
Input	Input.Value
Input Deadban	Variable.Value

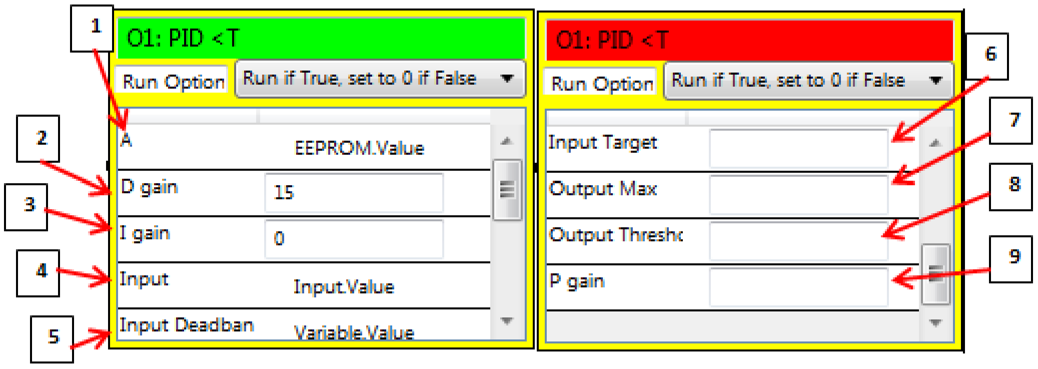
Input Target	
Output Max	
Output Threshc	
P gain	

FIGURE 62: PID>T OPERATOR BLOCK

1. A
2. D gain
3. I gain
4. Input
5. Input Deadband
6. Input Target
7. Output Max
8. Output Threshold
9. P gain

PID<T

This block is used to provide some closed loop control for an output using the error correction from a PID operation as long as the target value is greater than the input value, i.e. the raising of a robotic arm on a refuse vehicle to some point above its resting position.
As a note, in most cases the PID<T and PID>T blocks are used in conjunction to control an output such as described raising and lowering of a robotic arm in a smooth and controlled manner.


1
2
3
4
5
6
7
8
9
O1: PID < T
Run Option
Run if True, set to 0 if False
O1: PID < T
Run Option
Run if True, set to 0 if False

A	EEPROM.Value
D gain	15
I gain	0
Input	Input.Value
Input Deadban	Variable.Value

Input Target	
Output Max	
Output Threshc	
P gain	

FIGURE 63: PID<T OPERATOR BLOCK

1. A
2. D gain
3. I gain
4. Input
5. Input Deadband
6. Input Target
7. Output Max
8. Output Threshold
9. P gain

PIDspd

This block is used to provide some closed loop control for an output using the error correction from a PID operation as long as the target value is not equal to the input value, i.e. cruise control within a vehicle. Based on the outcome of the operation, an Output will be driven accordingly to maintain a target value.

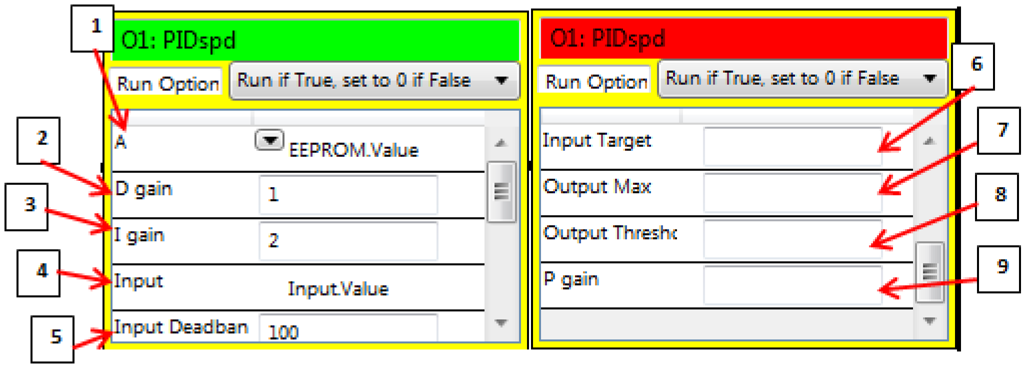


FIGURE 64: PIDSPD OPERATOR BLOCK

1. A
2. D gain
3. I gain
4. Input
5. Input Deadband
6. Input Target
7. Output Max
8. Output Threshold
9. P gain

Ramp

The Ramp Operator Block will gradually change the value passed to “A” from the value entered into the “Start Value” to the “End Value”. How gradually it changes is controlled by the value assigned to the “Ramp” data field. Each loop through the program will change the value in “A” by the “Ramp” amount up to the “End Value” as long as the run option criterion is met.

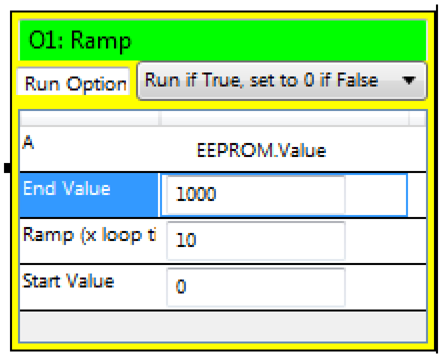


FIGURE 65: RAMP OPERATOR BLOCK

JOYabv

The JOYabv block works to convert an Input value into a linearly proportional Output value when the Input value is above the defined Input Center.

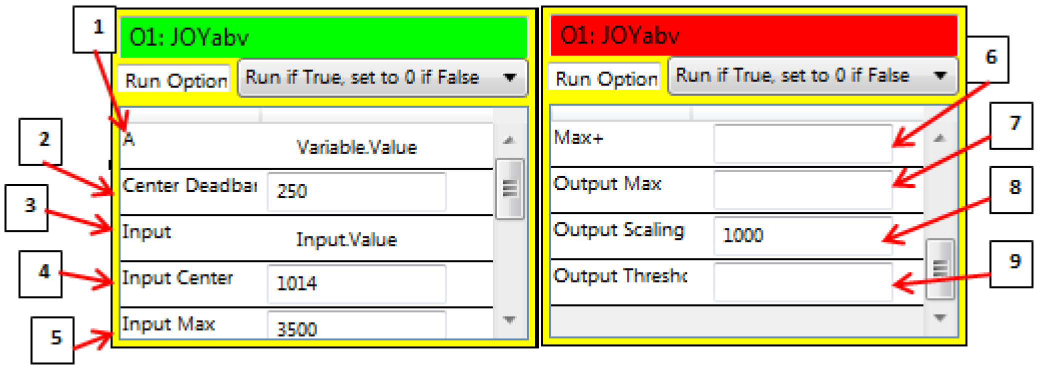


FIGURE 66: JOYabv OPERATOR BLOCK

1. A
2. Center Deadband
3. Input
4. Input Center
5. Input Max
6. Max+
7. Output Max
8. Output Scaling
9. Output Threshold

JOYblw

The JOYblw block works to convert an Input value into a linearly proportional Output value when the Input value is below the defined Input Center.

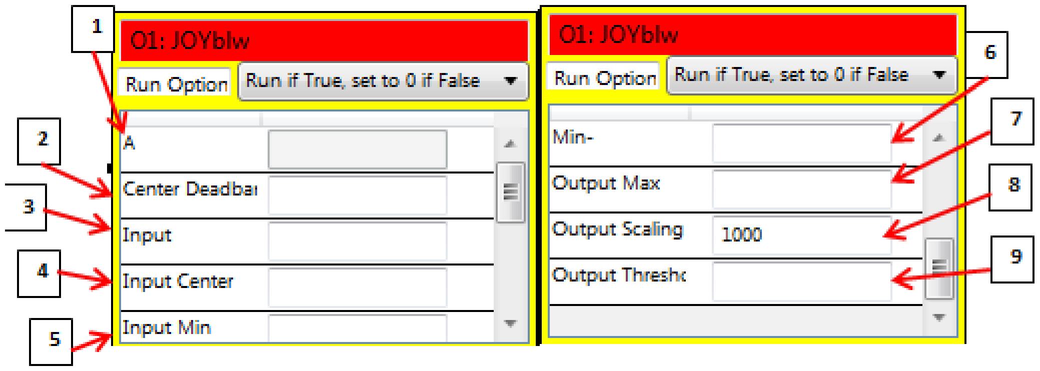


FIGURE 67: JOYblw OPERATOR BLOCK

1. A
2. Center Deadband
3. Input
4. Input Center
5. Input Min
6. Min-
7. Output Max
8. Output Scaling
9. Output Threshold

The JOYabv and JOYblw are usually used in conjunction in a setting that would need an output driven linearly by an input value. An example of this would be a joystick, as the position input of the joystick increases the output increases proportionally to possibly open something (JOYabv), while the decreasing of the joystick position past the center point would increase an output (JOYblw) that could possibly close what was opened.

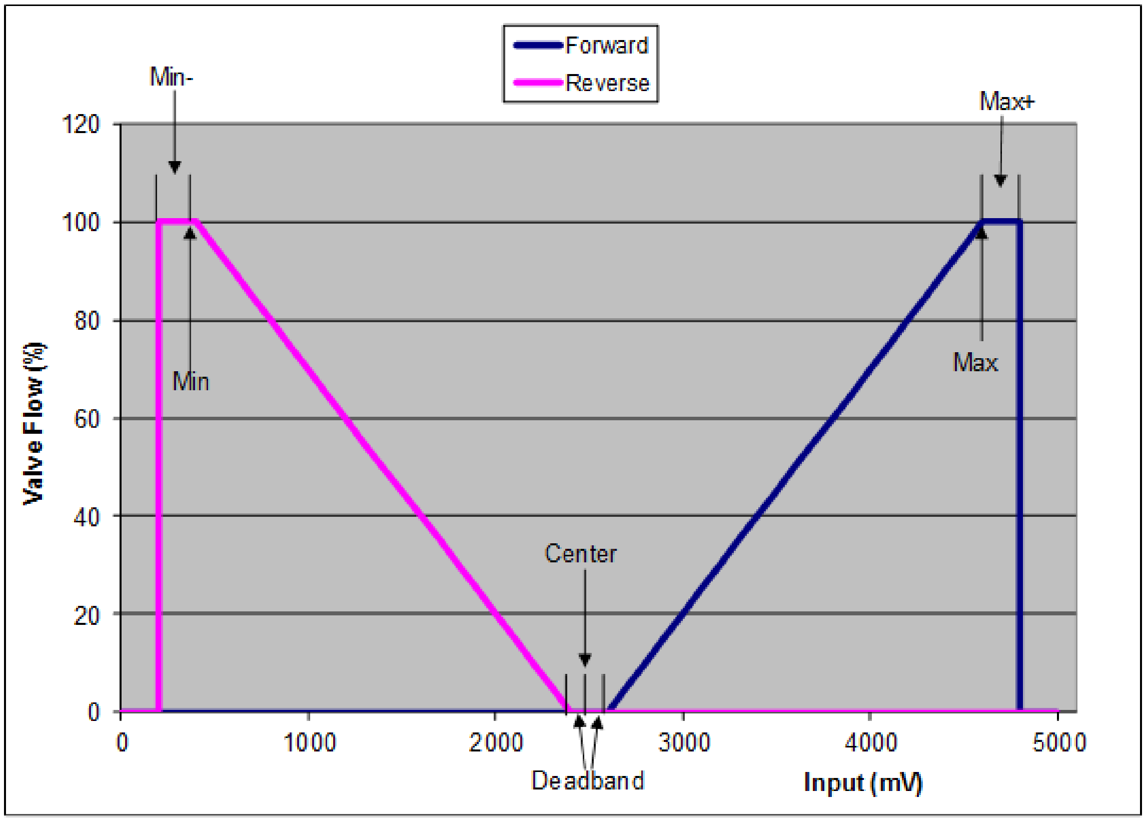
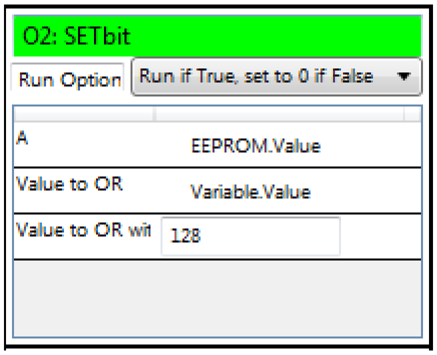


FIGURE 68: GRAPHICAL REFERENCE OF JOYABV AND JOYBLW PROPERTIES

SETbit

The SETbit block performs a bitwise OR on the value in the "Value to OR" field with a value in the "Value to OR with". The resulting value is placed in "A" if the rung logic satisfies the run option of the block.



O2: SETbit

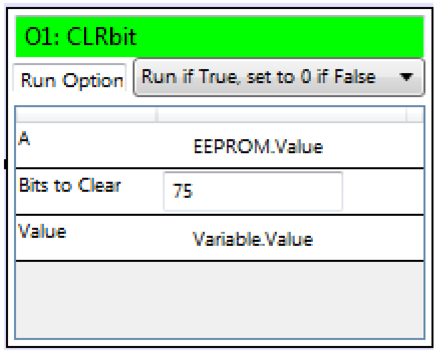
Run Option: Run if True, set to 0 if False

A	EEPROM.Value
Value to OR	Variable.Value
Value to OR wit	128

FIGURE 69: SETBIT OPERATOR BLOCK

CLRbit

This block will use the “Bits to Clear” value as a mask to toggle the selected high bits (1) in the “Value” data field to low (0) if the run option is met, and place the resulting value in “A”.



01: CLRbit

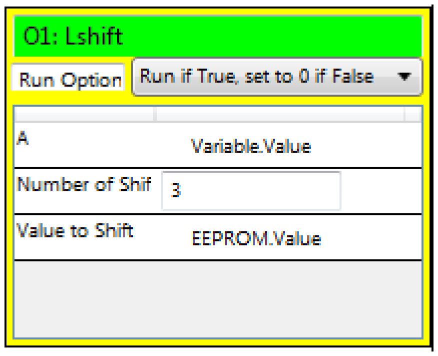
Run Option: Run if True, set to 0 if False

A	EEPROM.Value
Bits to Clear	75
Value	Variable.Value

FIGURE 70: CLRBIT OPERATOR BLOCK

Lshift

The Lshift Operator Block performs a logical shift left on the value in the “Value to Shift” field by the number of times of the value in “Number of Shifts” field and places the result in “A” if the run option is met.



01: Lshift

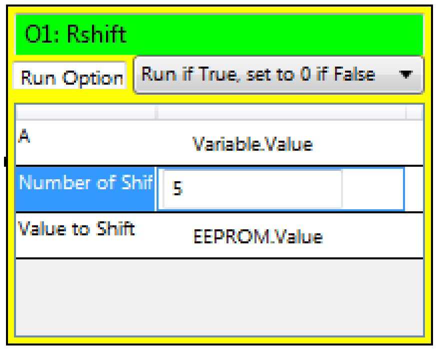
Run Option: Run if True, set to 0 if False

A	Variable.Value
Number of Shif	3
Value to Shift	EEPROM.Value

FIGURE 71: LSHIFT OPERATOR BLOCK

Rshift

The Rshift Operator Block performs a logical shift right on the value in the “Value to Shift” field by the number of times of the value in “Number of Shifts” field and places the result in “A” if the run option is met.



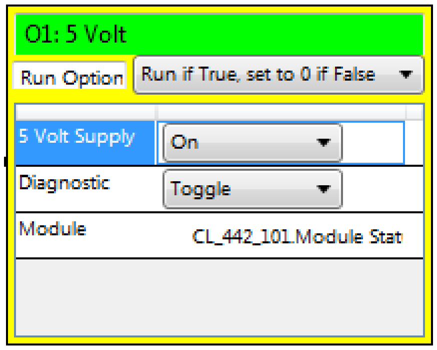
O1: Rshift

Run Option	Run if True, set to 0 if False
A	Variable.Value
Number of Shif	5
Value to Shift	EEPROM.Value

FIGURE 72: RSHIFT OPERATOR BLOCK

5 Volt

This Operator Block will turn the 5 Volt supply within the Module on or off when the run option is met. The User will also have the ability to select whether or not they would like to turn a diagnostic on, off, or toggle the diagnostic on and off. The diagnostic uses a pull up resistor to monitor and report back whether or not the output is shorted to battery, ground, open when closed expected, etc. The toggle feature will turn the diagnostic on for 500ms then off for 500ms and repeat.



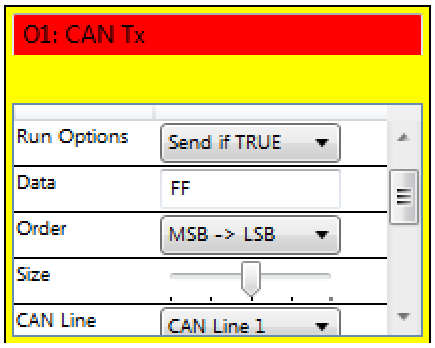
O1: 5 Volt

Run Option	Run if True, set to 0 if False
5 Volt Supply	On
Diagnostic	Toggle
Module	CL_442_101.Module Stat

FIGURE 73: 5 VOLT OPERATOR BLOCK

CAN Tx

The CAN Tx block will transmit a User defined CAN message from the selected Module over the chosen CAN channel whenever the run option is met. The message can have either a hex or decimal format and can be a defined data length of up to 8 bytes. With the addition of each byte, the User can choose the size of the data within the message i.e. a defined length of 3 bytes can have 3 separate 1 byte data values or 1 2 byte and a 1 byte value. The data can have maximum of 1 8 byte value, 2 4 byte values, and so on, down to 8 1 byte values. If the data is greater than 1 byte the User must define the "Order" of the bytes from MSB->LSB or LSB->MSB. The final two fields within the block are the "ID" and "ID Size" fields. The "ID Size" can be 11-bit or 29-bit and defines the size of the identifier for the CAN message that the User specifies.

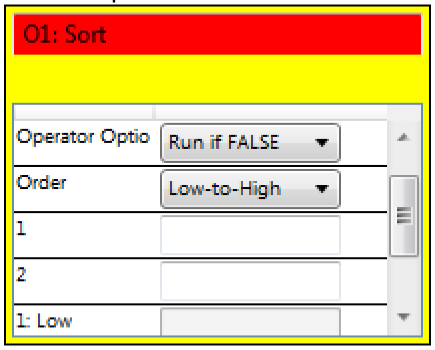

O1: CAN Tx

Run Options	Send if TRUE
Data	FF
Order	MSB -> LSB
Size	
CAN Line	CAN Line 1

FIGURE 74: CAN TX OPERATOR BLOCK

Sort

The Sort block is another one of the few blocks that do not have a "Run Option" and instead has an "Operator Option". That option only allows the block to execute if the rung is either True or False depending on what the User chooses. This block can take up to four values and sort then in either a highest to lowest or lowest to highest order. The sorted order is placed into corresponding Data Items within the respective fields.


O1: Sort

Operator Optio	Run if FALSE
Order	Low-to-High
1	
2	
1: Low	

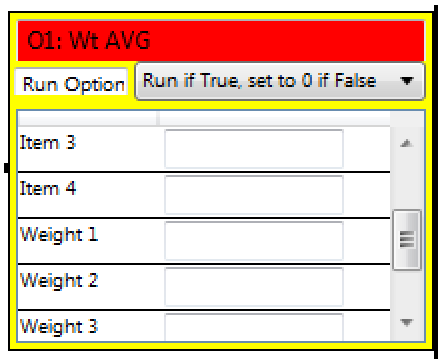
FIGURE 75: SORT OPERATOR BLOCK

Wt AVG

This block computes a weighted average of up to four values and four different weights, then places the average in the Data Item within the “AVG stored here” data field each time the rung logic meets the run option criteria. Each “Item” has a corresponding “Weight” and is utilized in the following way:

$$Wt\,AVG = (Item1 \times Weight1 + Item2 \times Weight2 + Item3 \times Weight3 + Item4 \times Weight4) \div Sum\ of\ weights$$

EQUATION 6: WEIGHTED AVERAGE EQUATION


O1: Wt AVG

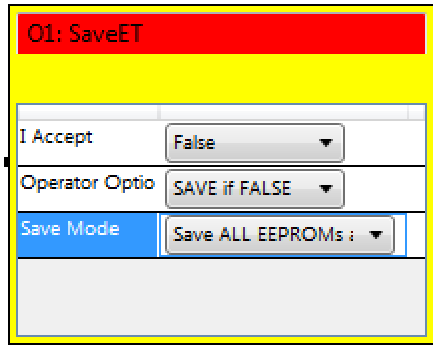
Run Option: Run if True, set to 0 if False

Item 3	
Item 4	
Weight 1	
Weight 2	
Weight 3	

FIGURE 76: WT AVG OPERATOR BLOCK

SaveET

This operator block takes the temporary stored EEPROM and Timer values and stores them to the processor's EEPROM locations. This storing process could cause the system to have an unknown delay in its ability to execute its logic and timing capability (extended loop time).
The system memory is rated up to 10,000 write cycles for EEPROMS and Timers. Execution of this operator block more than 10,000 times may cause unforeseen errors to the data stored and consequently to the system logic. In order to activate this block the User must change the “I Accept” field to True and agree to the terms stated previously.
The SaveET block is the last block without a “Run Option” field, instead there is an “Operator Option” field that allows the User to select a “Save if True” or “Save if False” option that dictates when the Operator Block will run.
EEPROMS and Timers, as explained in their respective Data Item sections, each have an option for saving and when that option is selected, this Operator Block is what performs the actual save. There are five options for saving; Save ALL EEPROMs with feature enabled, Save ALL Timers with feature enabled, Save ALL EEPROMs and Timers with feature enabled, Save single EEPROM, and Save single Timer. If either of the save single options are selected, the User will have to designate which one to save by dragging that Data Item to the “Select” field

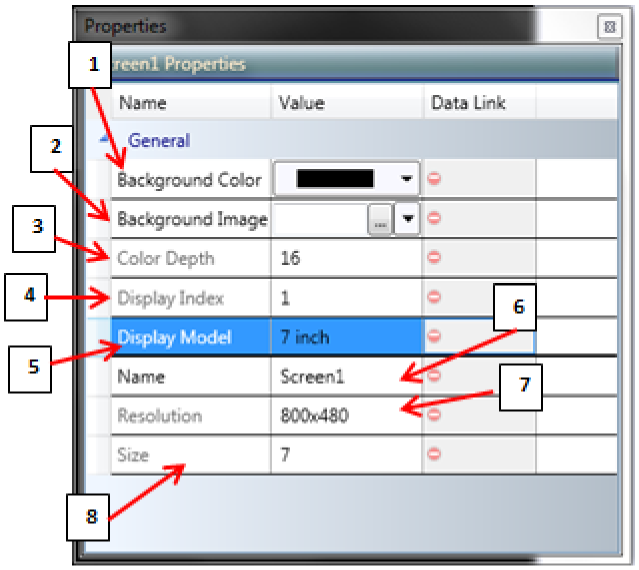

O1: SaveET

I Accept	False
Operator Optio	SAVE if FALSE
Save Mode	Save ALL EEPROMs a

FIGURE 77: SAVEET OPERATOR BLOCK

Arranger

The Arranger Tool is what enables the User to easily create screens to be placed on a display that will convey the desired information to the outside world by simply placing, clicking, and dragging the various widgets available. To begin using Arranger right click the "Screens" folder on the Explorer Panel and create a new Screen. Upon creating a new screen the User will see a black screen with a "Widgets" tool panel to the left of it. Highlighting the newly created screen in the Explorer Panel will switch the properties panel to the properties associated with the screen.

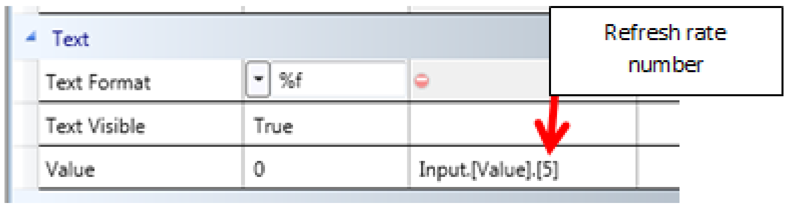

Properties
reen1 Properties

Name	Value	Data Link
General		
Background Color	■	⊖
Background Image		⊖
Color Depth	16	⊖
Display Index	1	⊖
Display Model	7 inch	⊖
Name	Screen1	⊖
Resolution	800x480	⊖
Size	7	⊖

FIGURE 78: SCREEN PROPERTIES

1. Background Color
2. Background Image
3. Color Depth
4. Display Index
5. Display Model
6. Name
7. Resolution
8. Size

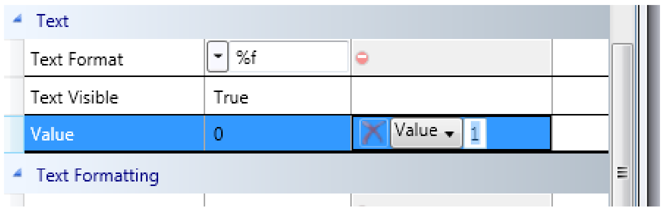
Once the properties are set to their desired values, the User should begin editing the screen by adding Widgets. To add a Widget just left click on the desired Widget from the Widget Panel on the left and it will appear in the far upper left corner of the screen (0, 0).
As a note, when using data links with properties a small number next to the link represents the rate, in 100 ms increments, at which the display will request an update of that link.



Text		
Text Format	%f	⊖
Text Visible	True	
Value	0	Input.[Value].[5]

FIGURE 79: DATA LINK REFRESH RATE

The number can be changed if desired by double clicking on the link, highlighting the number, and typing in a new one.



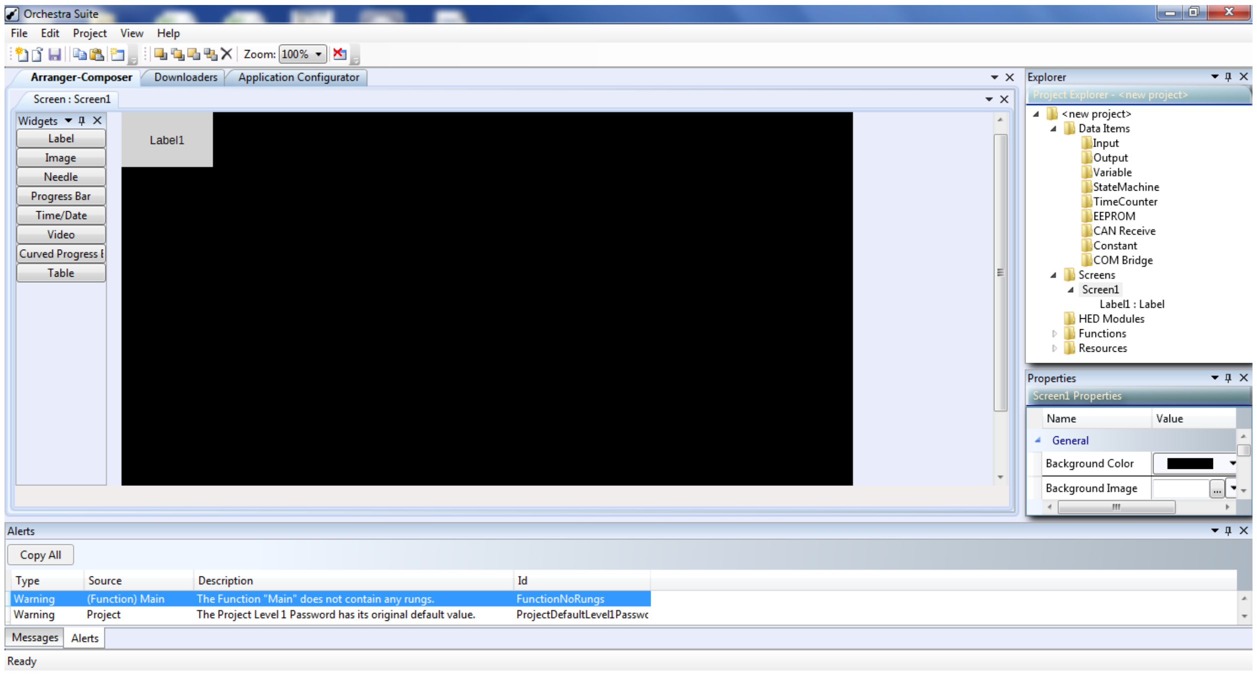
Text

Text Format	%f	
Text Visible	True	
Value	0	Value 1

Text Formatting

FIGURE 80: CHANGING DATA LINK REFRESH RATE

Changing this number can help with improving response time of changing values on the display, but will use more processing power.



Name	Value
General	
Background Color	■
Background Image	

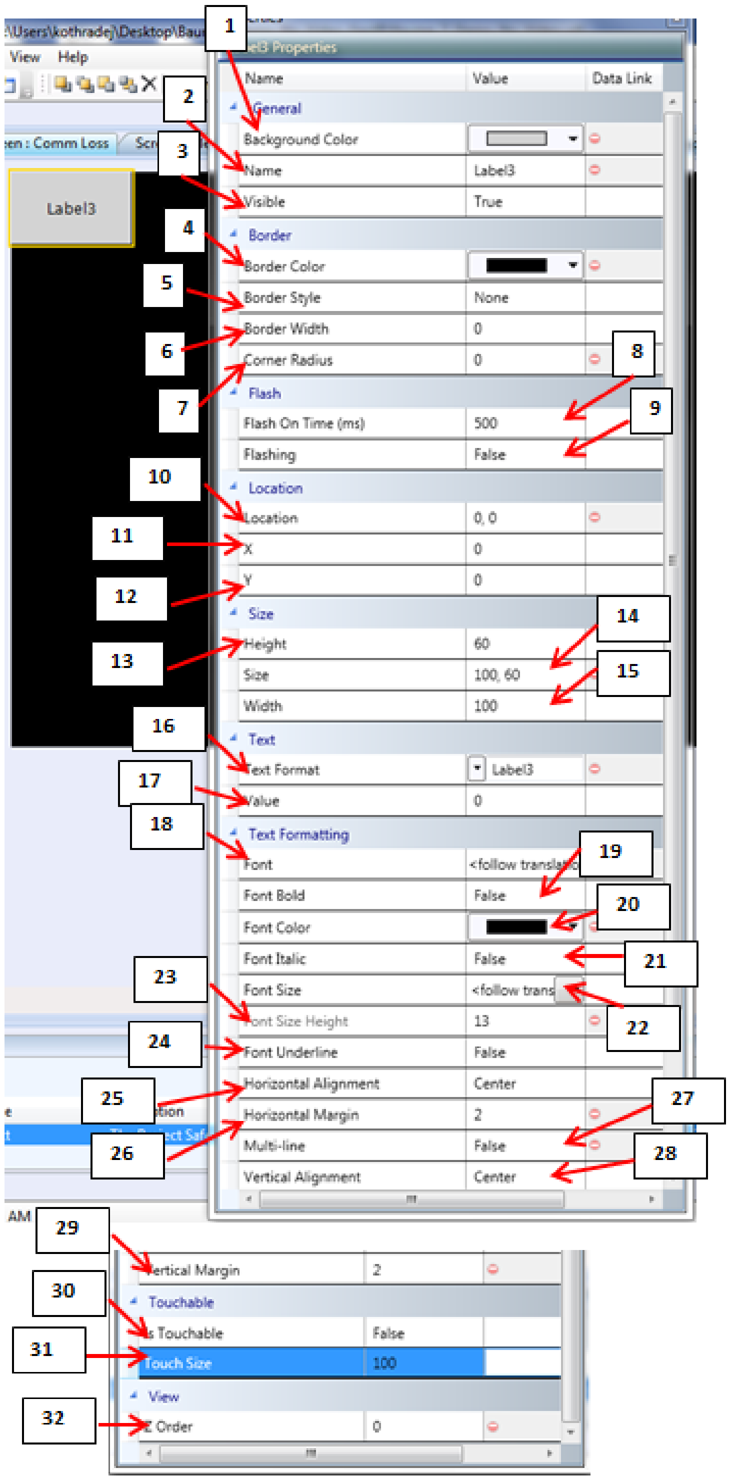
Alerts

Type	Source	Description	Id
Warning	(Function) Main	The Function "Main" does not contain any rungs.	FunctionNoRungs
Warning	Project	The Project Level 1 Password has its original default value.	ProjectDefaultLevel1Passwc

FIGURE 81: ARRANGER PANEL

Label

The Label widget allows the User to display an alpha numeric note or message.



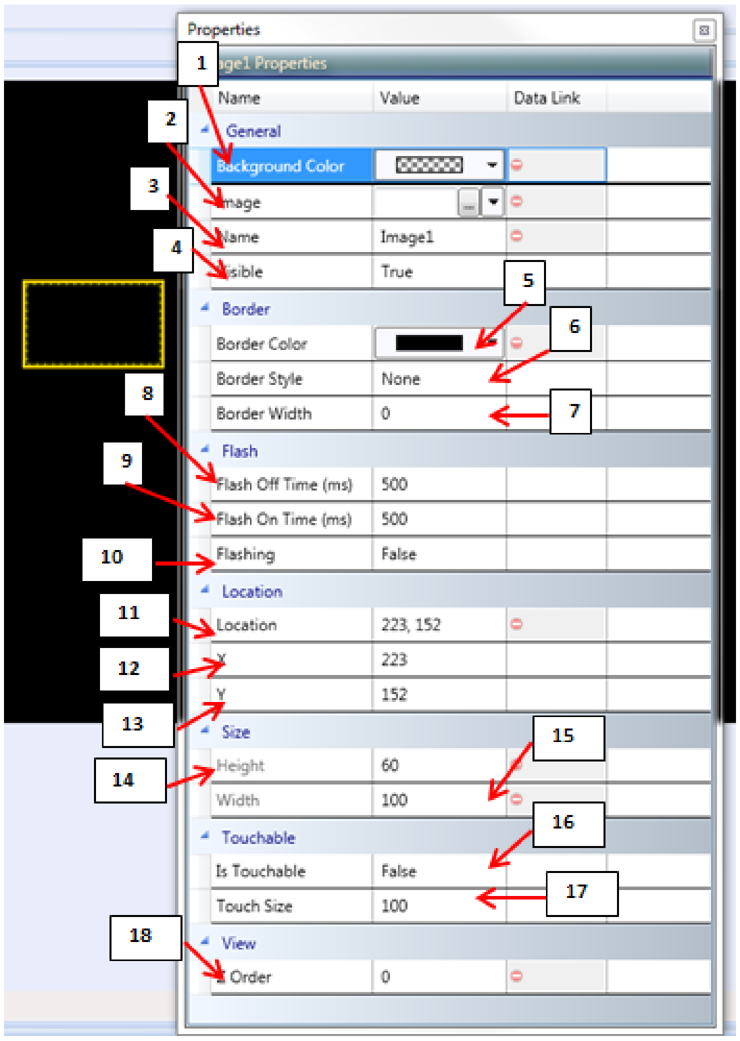
Name	Value	Data Link
General		
Background Color	▭	
Name	Label3	
Visible	True	
Border		
Border Color	■	
Border Style	None	
Border Width	0	
Corner Radius	0	
Flash		
Flash On Time (ms)	500	
Flashing	False	
Location		
Location	0, 0	
X	0	
Y	0	
Size		
Height	60	
Size	100, 60	
Width	100	
Text		
Text Format	Label3	
Value	0	
Text Formatting		
Font	<follow translatio	
Font Bold	False	
Font Color	■	
Font Italic	False	
Font Size	<follow trans	
Font Size Height	13	
Font Underline	False	
Horizontal Alignment	Center	
Horizontal Margin	2	
Multi-line	False	
Vertical Alignment	Center	
ertical Margin	2	
Touchable		
s Touchable	False	
Touch Size	100	
View		
Z Order	0	

FIGURE 82 : LABEL PROPERTIES

1. Background Color
2. Name
3. Visible
4. Border Color
5. Border Style
6. Border Width
7. Corner Radius
8. Flash On Time (ms)
9. Flashing
10. Location
11. X
12. Y
13. Height
14. Size
15. Width
16. Text Format
17. Value
18. Font
19. Font Bold
20. Font Color
21. Font Italic
22. Font Size
23. Font Size Height
24. Font Underline
25. Horizontal Alignment
26. Horizontal Margin
27. Multi-Line
28. Vertical Alignment
29. Vertical Margin
30. Is Touchable
31. Touch Size
32. Z Order

Image

The Image widget allows the User to use an image from a file and place it on the screen.



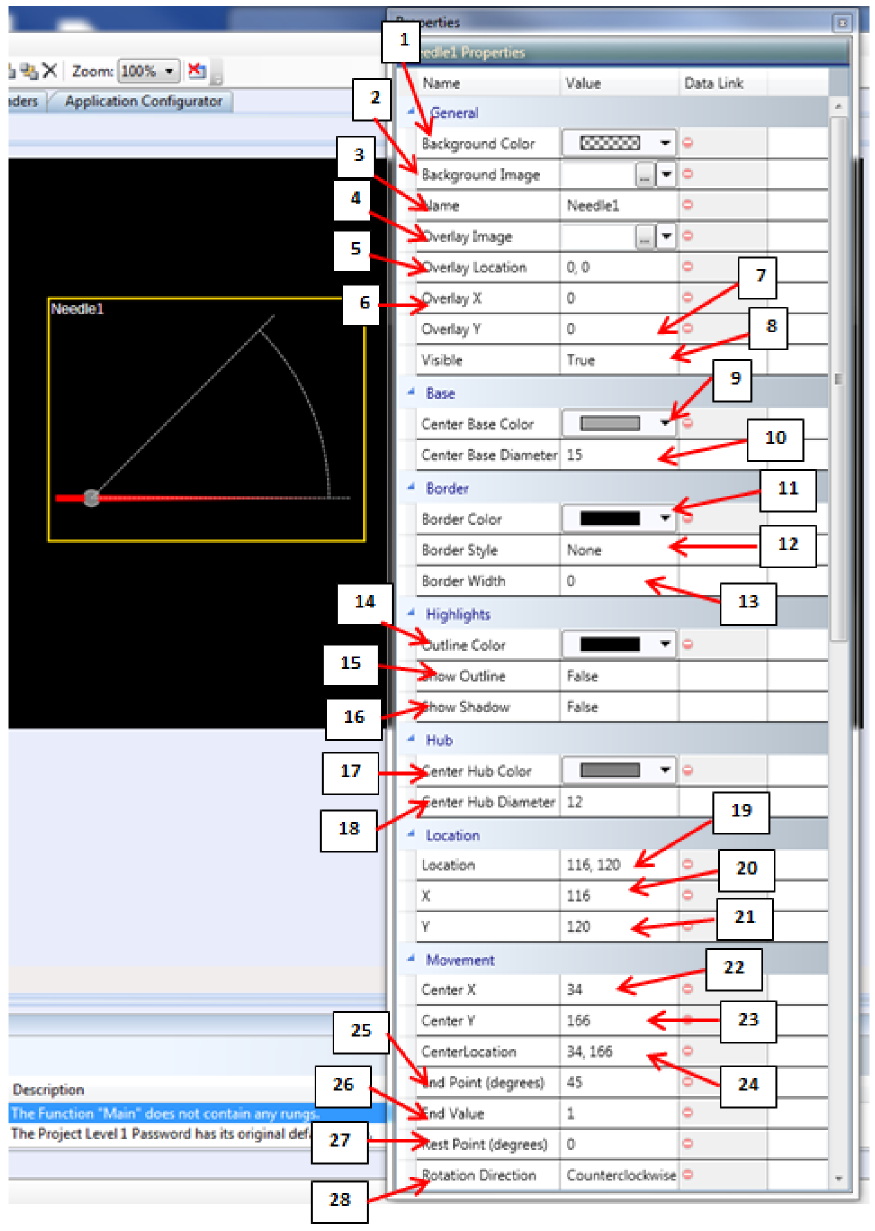
Name	Value	Data Link
General		
Background Color	▨	
Image		
Name	Image1	
Visible	True	
Border		
Border Color	■	
Border Style	None	
Border Width	0	
Flash		
Flash Off Time (ms)	500	
Flash On Time (ms)	500	
Flashing	False	
Location		
Location	223, 152	
X	223	
Y	152	
Size		
Height	60	
Width	100	
Touchable		
Is Touchable	False	
Touch Size	100	
View		
Z Order	0	

FIGURE 83 : IMAGE PROPERTIES

1. Background Color
2. Image
3. Name
4. Visible
5. Border Color
6. Border Style
7. Border Width
8. Flash Off Time (ms)
9. Flash On Time (ms)
10. Flash
11. Location
12. X
13. Y
14. Height
15. Width
16. Is Touchable
17. Touch Size
18. Z Order

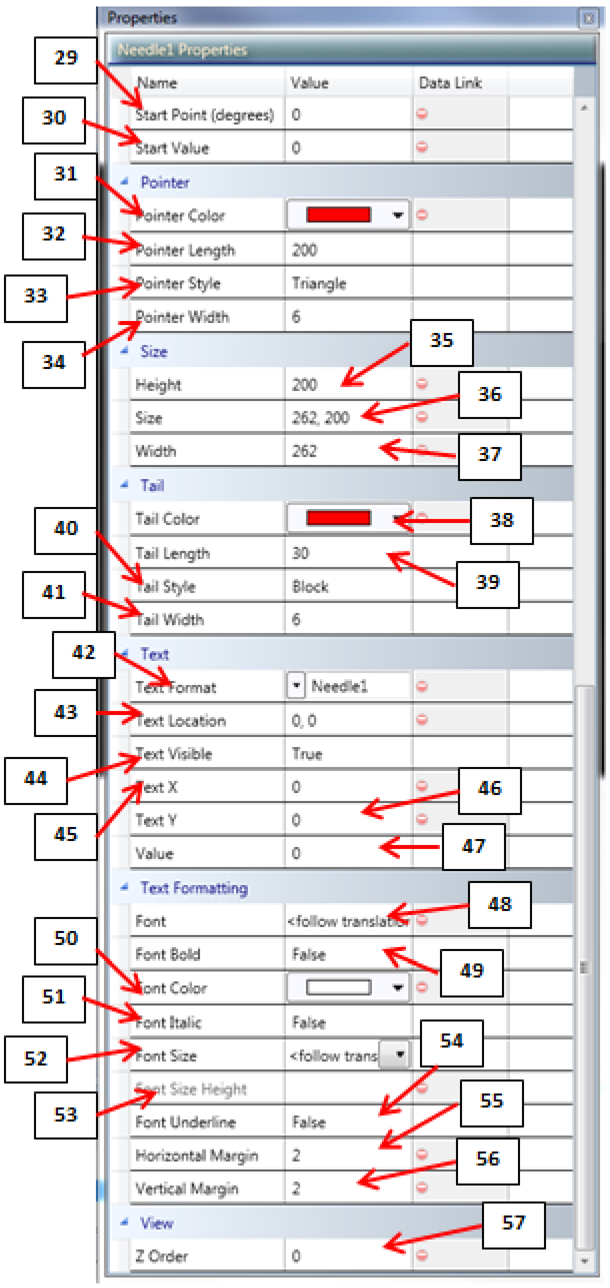
Needle

The Needle widget will place a needle gauge on the screen that will show the progress of the item linked to it.



Name	Value	Data Link
General		
Background Color	▨	
Background Image		
Name	Needle1	
Overlay Image		
Overlay Location	0, 0	
Overlay X	0	
Overlay Y	0	
Visible	True	
Base		
Center Base Color		
Center Base Diameter	15	
Border		
Border Color	■	
Border Style	None	
Border Width	0	
Highlights		
Outline Color	■	
Show Outline	False	
Show Shadow	False	
Hub		
Center Hub Color		
Center Hub Diameter	12	
Location		
Location	116, 120	
X	116	
Y	120	
Movement		
Center X	34	
Center Y	166	
CenterLocation	34, 166	
End Point (degrees)	45	
End Value	1	
Rest Point (degrees)	0	
Rotation Direction	Counterclockwise	

Description
The Function "Main" does not contain any rungs.
The Project Level 1 Password has its original def



Name	Value	Data Link
Start Point (degrees)	0	
Start Value	0	
Pointer		
Pointer Color	🟥	
Pointer Length	200	
Pointer Style	Triangle	
Pointer Width	6	
Size		
Height	200	
Size	262, 200	
Width	262	
Tail		
Tail Color	🟥	
Tail Length	30	
Tail Style	Block	
Tail Width	6	
Text		
Text Format	Needle1	
Text Location	0, 0	
Text Visible	True	
Text X	0	
Text Y	0	
Value	0	
Text Formatting		
Font	<follow translatio	
Font Bold	False	
Font Color	⬜	
Font Italic	False	
Font Size	<follow trans	
Font Size Height		
Font Underline	False	
Horizontal Margin	2	
Vertical Margin	2	
View		
Z Order	0	

1. Background Color
2. Background Image
3. Name
4. Overlay Image
5. Overlay Location
6. Overlay X
7. Overlay Y
8. Visible
9. Center Base Color
10. Center Base Diameter
11. Border Color
12. Border Style
13. Border Width
14. Outline Color
15. Show Outline
16. Show Shadow
17. Center Hub Color
18. Center Hub Diameter
19. FlashingLocation
20. X
21. Y
22. Center X
23. Center Y
24. CenterLocation
25. End Point (degrees)
26. End Value
27. Rest Point (degrees)
28. Rotation Direction
29. Start Point (degrees)
30. Start Value
31. Pointer Color
32. Pointer Length
33. Pointer Style
34. Pointer Width
35. Height
36. Size

37. Width
38. Tail Color
39. Tail Length
40. Tail Style
41. Tail Width
42. Text Format
43. Text Location
44. Text Visible
45. Text X
46. Text Y
47. Value
48. Font
49. Font Bold
50. Font Color
51. Font Italic
52. Font Size
53. Font Size Height
54. Font Underline
55. Horizontal Margin
56. Vertical Margin
57. Z Order

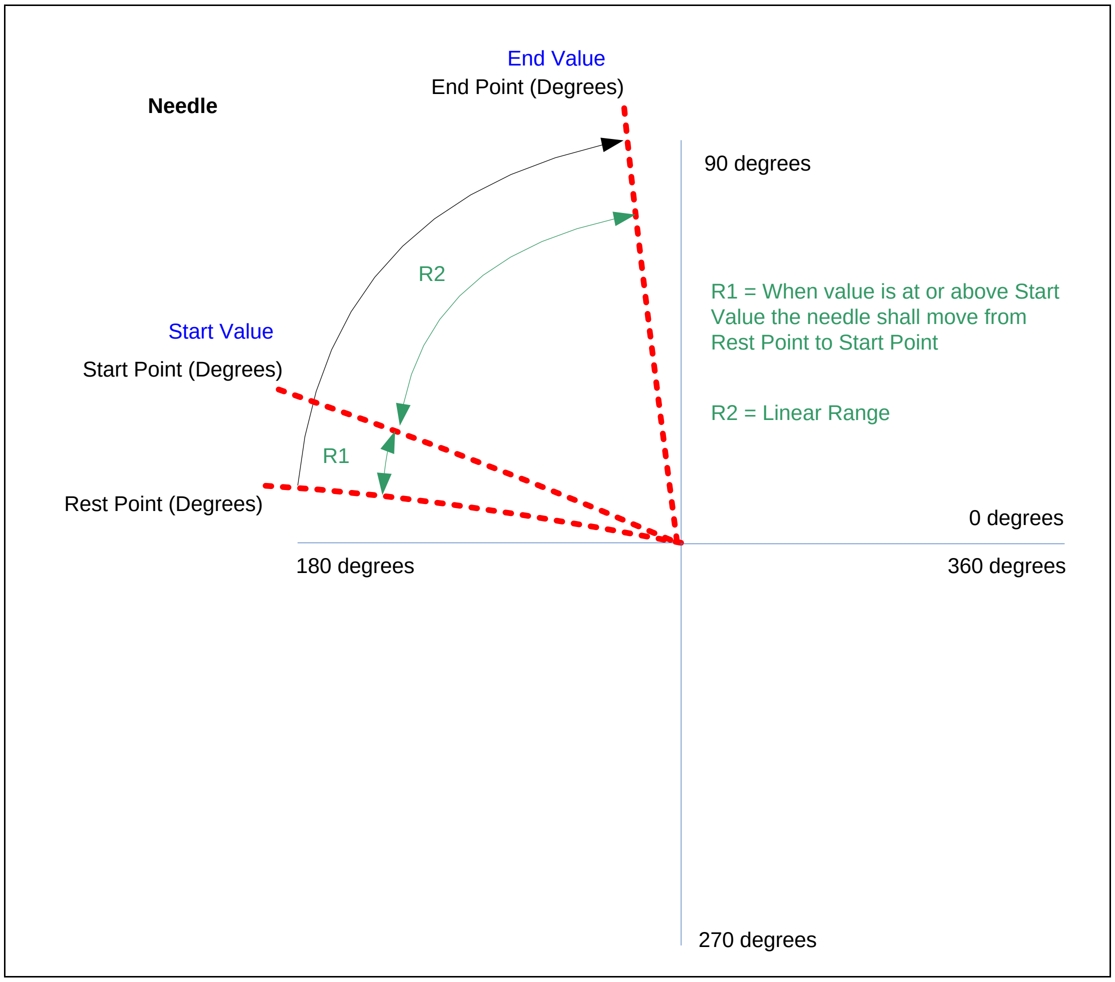
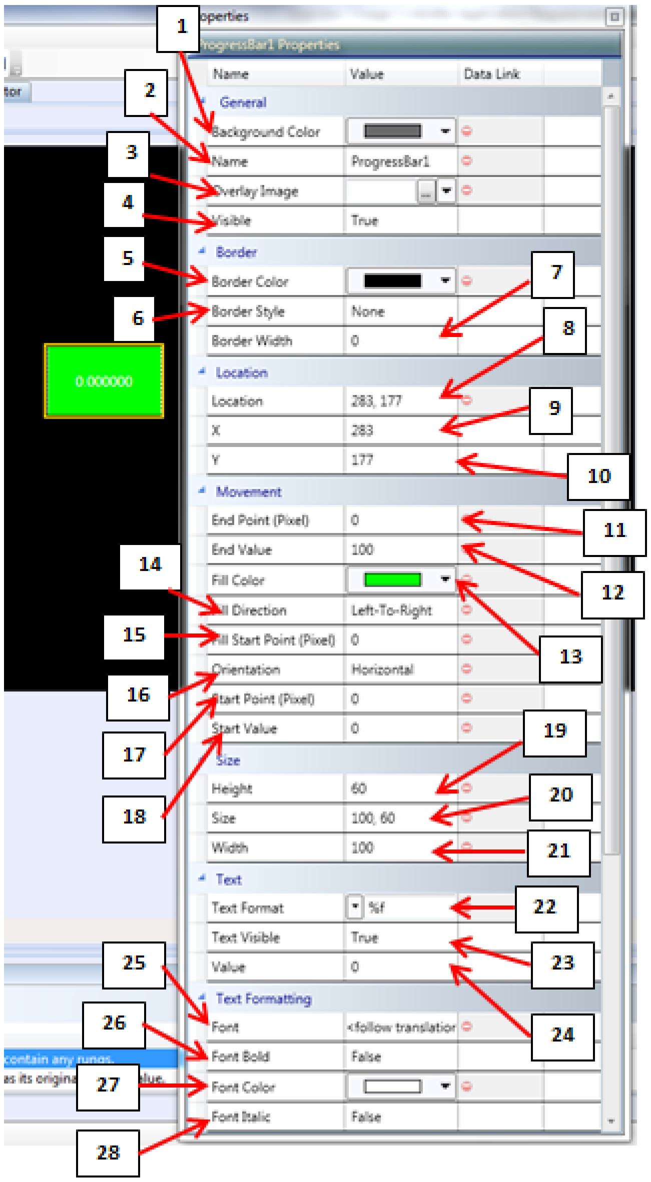


FIGURE 84: NEEDLE WIDGET START AND END POINT EXPLANATION

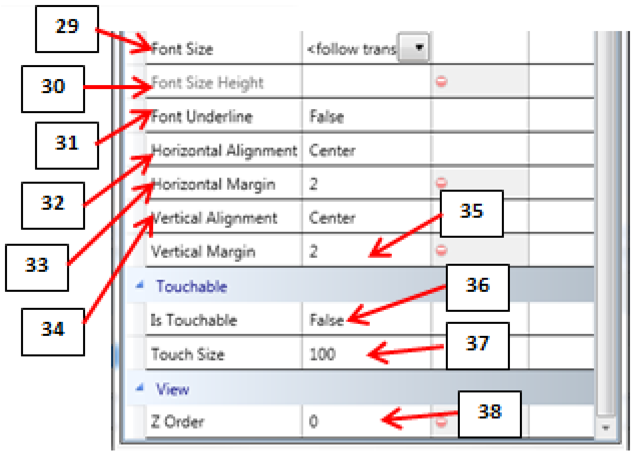
Progress Bar

The Progress Bar widget will place a progress bar on the screen that the User can link an item to, to track that item's value graphically.



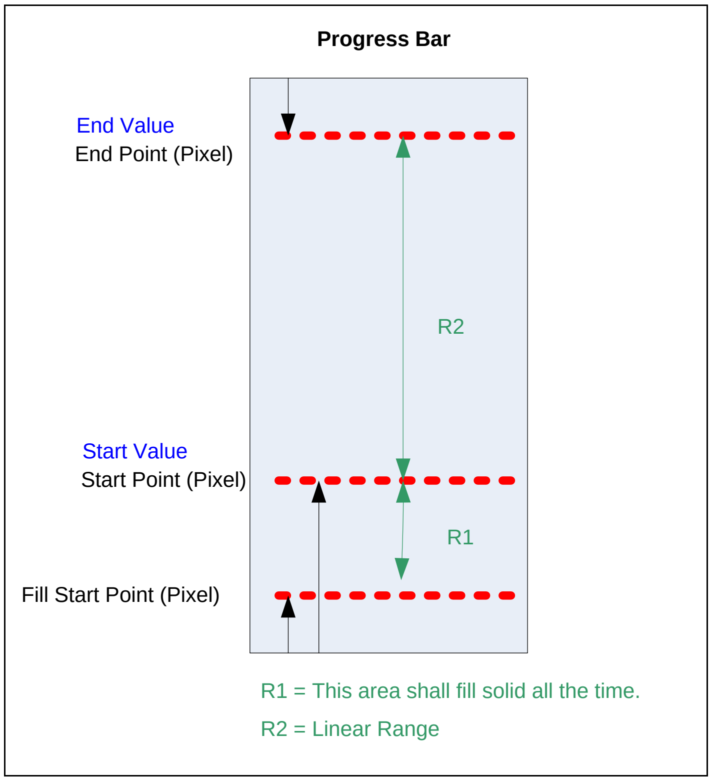
ProgressBar1 Properties

Name	Value	Data Link
General		
Background Color	⬛	
Name	ProgressBar1	
Overlay Image		
Visible	True	
Border		
Border Color	⬛	
Border Style	None	
Border Width	0	
Location		
Location	283, 177	
X	283	
Y	177	
Movement		
End Point (Pixel)	0	
End Value	100	
Fill Color	🟩	
Fill Direction	Left-To-Right	
Fill Start Point (Pixel)	0	
Orientation	Horizontal	
Start Point (Pixel)	0	
Start Value	0	
Size		
Height	60	
Size	100, 60	
Width	100	
Text		
Text Format	%f	
Text Visible	True	
Value	0	
Text Formatting		
Font	<follow translatior	
Font Bold	False	
Font Color	⬜	
Font Italic	False	



Font Size	<follow trans
Font Size Height	
Font Underline	False
Horizontal Alignment	Center
Horizontal Margin	2
Vertical Alignment	Center
Vertical Margin	2
Touchable	
Is Touchable	False
Touch Size	100
View	
Z Order	0

1. Background Color
2. Name
3. Overlay Image
4. Visible
5. Border Color
6. Border Style
7. Border Width
8. FlashingLocation
9. X
10. Y
11. End Point (Pixel)
12. End Value
13. Fill Color
14. Fill Direction
15. Fill Start Point (Pixel)
16. Orientation
17. Start Point (Pixel)
18. Start Value
19. Height
20. Size
21. Width
22. Text Format
23. Text Visible
24. Value
25. Font
26. Font Bold
27. Font Color
28. Font Italic
29. Font Size
30. Font Size Height
31. Font Underline
32. Horizontal Alignment
33. Horizontal Margin
34. Vertical Alignment
35. Vertical Margin
36. Is Touchable
37. Touch Size
38. Z Order

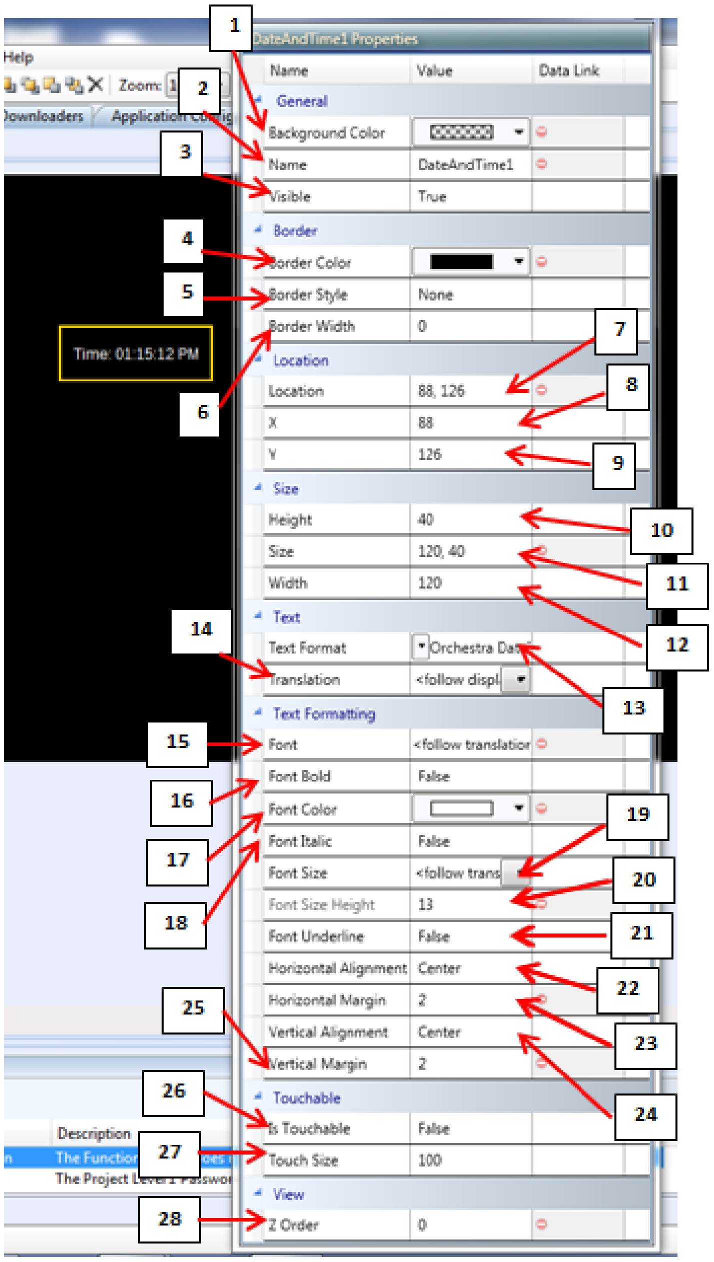


58. Text X

FIGURE 85: PROGRESS BAR START AND END POINT EXPLANATION

Time/Date

The Time/Date widget will place a label that has been preformatted to display the current time.



Name	Value	Data Link
General		
Background Color		
Name	DateAndTime1	
Visible	True	
Border		
Border Color		
Border Style	None	
Border Width	0	
Location		
Location	88, 126	
X	88	
Y	126	
Size		
Height	40	
Size	120, 40	
Width	120	
Text		
Text Format	Orchestra Da	
Translation	<follow displ	
Text Formatting		
Font	<follow translation	
Font Bold	False	
Font Color		
Font Italic	False	
Font Size	<follow trans	
Font Size Height	13	
Font Underline	False	
Horizontal Alignment	Center	
Horizontal Margin	2	
Vertical Alignment	Center	
Vertical Margin	2	
Touchable		
Is Touchable	False	
Touch Size	100	
View		
Z Order	0	

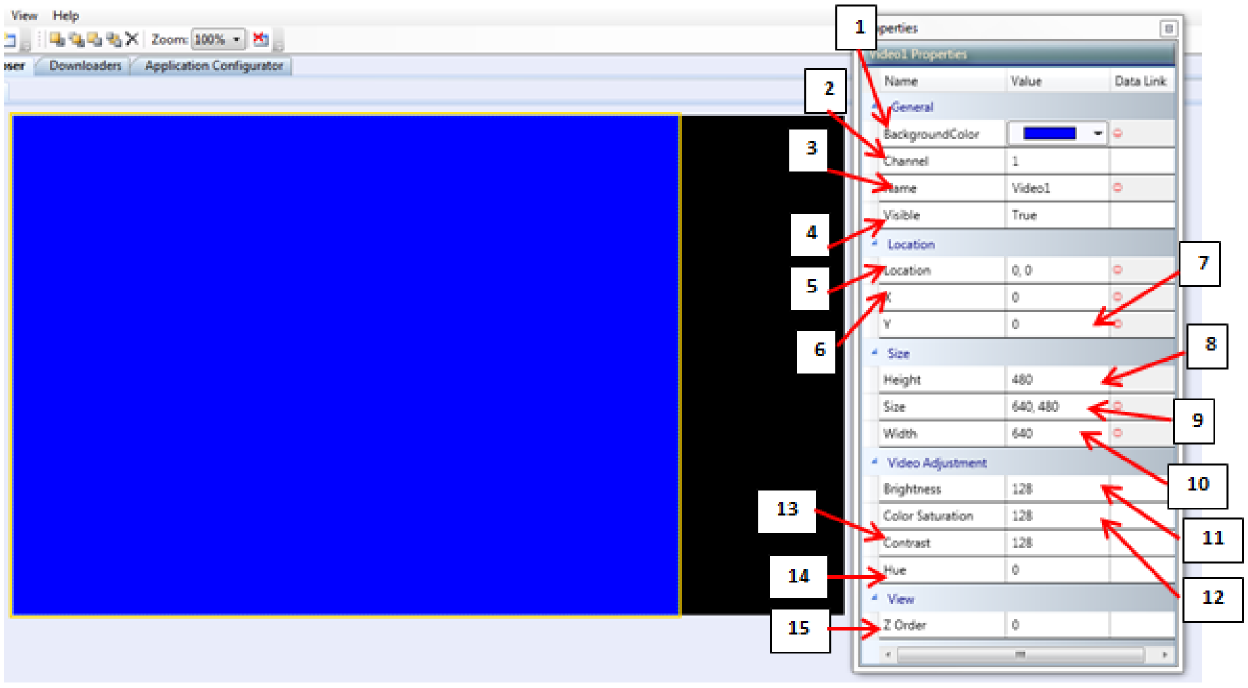
FIGURE 86: DATE AND TIME WIDGET PROPERTIES

1. Background Color
2. Name
3. Visible
4. Border Color
5. Border Style
6. Border Width
7. FlashingLocation
8. X
9. Y
10. Height
11. Size
12. Width
13. Text Format
14. Translation
15. Font
16. Font Bold
17. Font Color
18. Font Italic

19. Font Size
20. Font Size Height
21. Font Underline
22. Horizontal Alignment
23. Horizontal Margin
24. Vertical Alignment
25. Vertical Margin
26. Is Touchable
27. Touch Size
28. Z Order

Video

Video places a widget on the screen that will display a video feed from a camera.



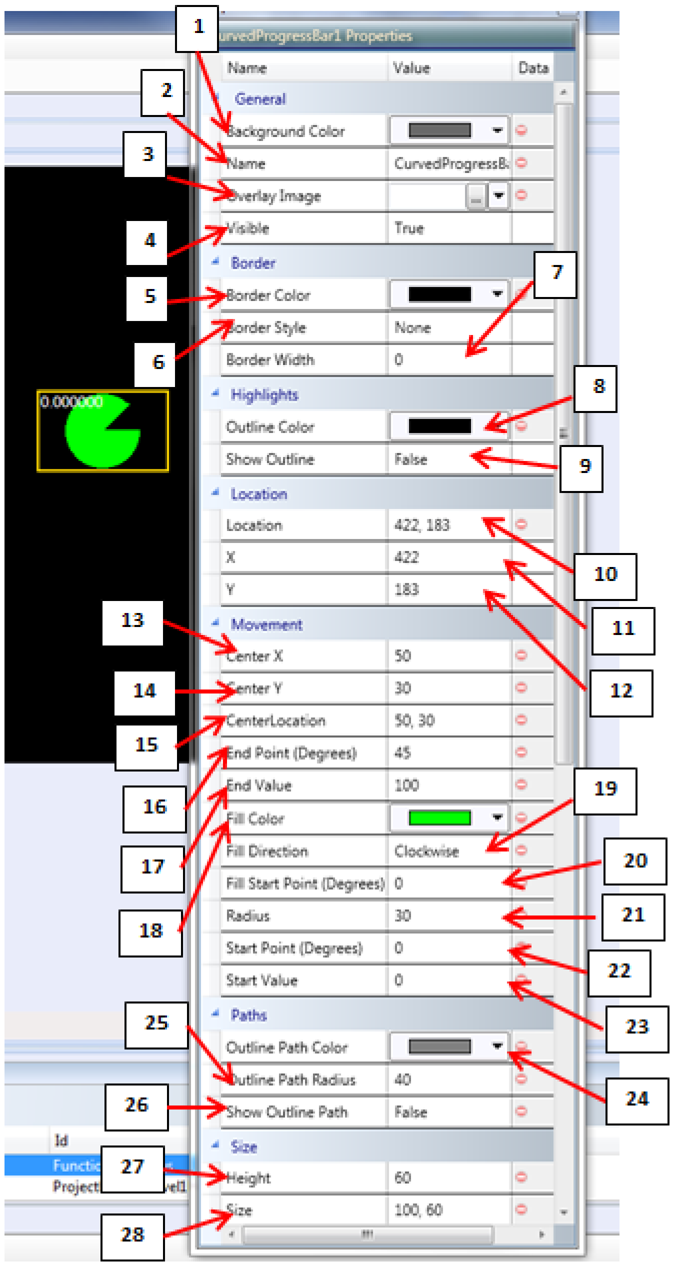
Name	Value	Data Link
General		
BackgroundColor	🟦	
Channel	1	
Name	Video1	
Visible	True	
Location		
Location	0, 0	
X	0	
Y	0	
Size		
Height	480	
Size	640, 480	
Width	640	
Video Adjustment		
Brightness	128	
Color Saturation	128	
Contrast	128	
Hue	0	
View		
Z Order	0	

FIGURE 87: VIDEO WIDGET PROPERTIES

1. Background Color
2. Channel
3. Name
4. Visible
5. FlashingLocation
6. X
7. Y
8. Height
9. Size
10. Width
11. Brightness
12. Color Saturation
13. Contrast
14. Hue
15. Z Order

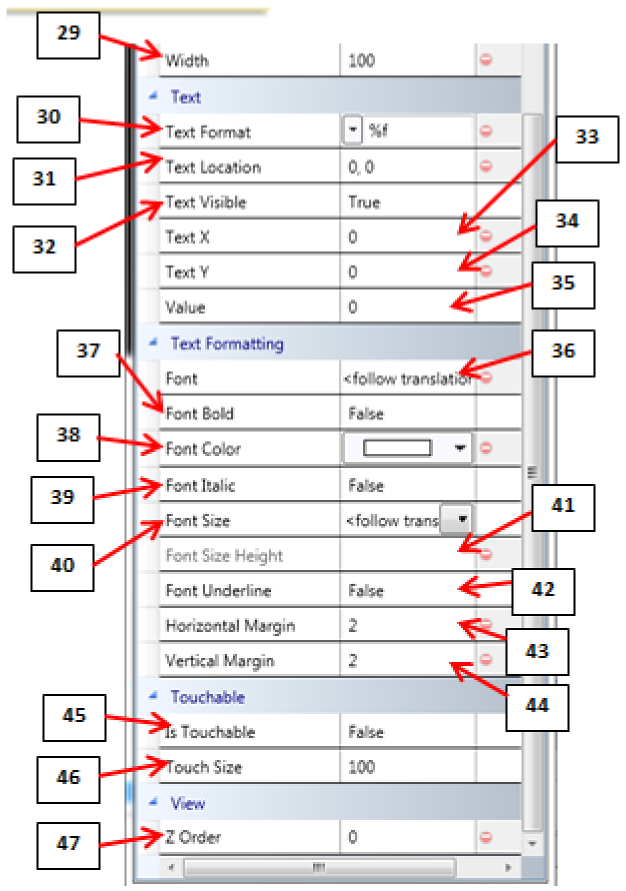
Curved Progress Bar

The Curved Progress Bar works similarly to the Progress bar, but instead it fills in a curved fashion versus a straight fashion.



urvedProgressBar1 Properties

Name	Value
General	
Background Color	▨
Name	CurvedProgressB
Overlay Image	
Visible	True
Border	
Border Color	■
Border Style	None
Border Width	0
Highlights	
Outline Color	■
Show Outline	False
Location	
Location	422, 183
X	422
Y	183
Movement	
Center X	50
Center Y	30
CenterLocation	50, 30
End Point (Degrees)	45
End Value	100
Fill Color	🟩
Fill Direction	Clockwise
Fill Start Point (Degrees)	0
Radius	30
Start Point (Degrees)	0
Start Value	0
Paths	
Outline Path Color	▨
Outline Path Radius	40
Show Outline Path	False
Size	
Height	60
Size	100, 60



Width	100
Text	
Text Format	%f
Text Location	0, 0
Text Visible	True
Text X	0
Text Y	0
Value	0
Text Formatting	
Font	<follow translatio
Font Bold	False
Font Color	▭
Font Italic	False
Font Size	<follow trans
Font Size Height	
Font Underline	False
Horizontal Margin	2
Vertical Margin	2
Touchable	
Is Touchable	False
Touch Size	100
View	
Z Order	0

1. Background Color
2. Name
3. Overlay Image
4. Visible
5. Border Color
6. Border Style
7. Border Width
8. Outline Color
9. Show Outline
10. FlashingLocation
11. X
12. Y
13. Center X
14. Center Y
15. CenterLocation
16. End Point (degrees)
17. End Value
18. Fill Color
19. Fill Direction
20. Fill Start Point (Degrees)
21. Radius
22. Start Point (degrees)
23. Start Value
24. Outline Path Color
25. Outline Path Radius
26. Show Outline Path
27. Height
28. Size
29. Width
30. Text Format
31. Text Location
32. Text Visible
33. Text X
34. Text Y
35. Value
36. Font
37. Font Bold
38. Font Color
39. Font Italic
40. Font Size
41. Font Size Height
42. Font Underline
43. Horizontal Margin
44. Vertical Margin
45. Is Touchable
46. Touch Size
47. Z Order

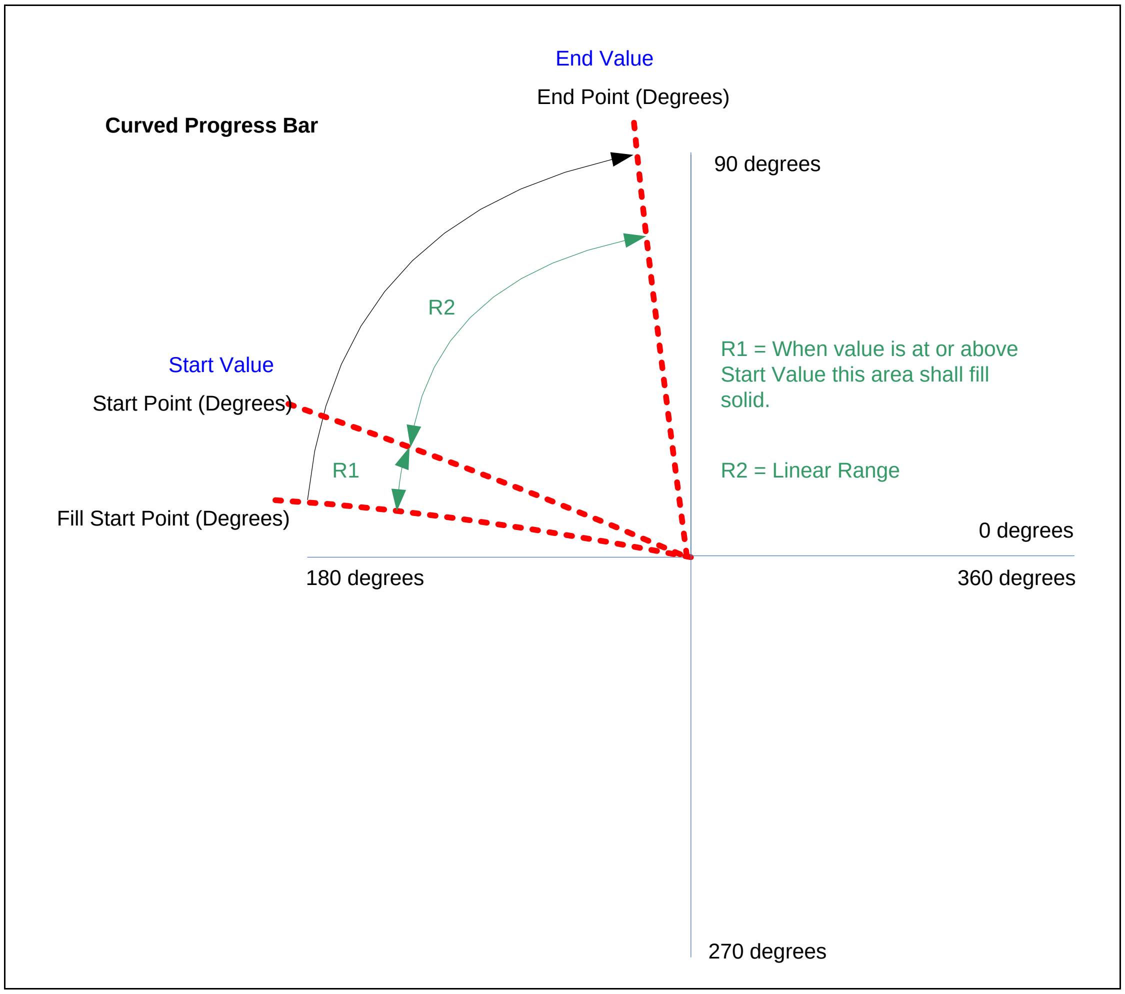
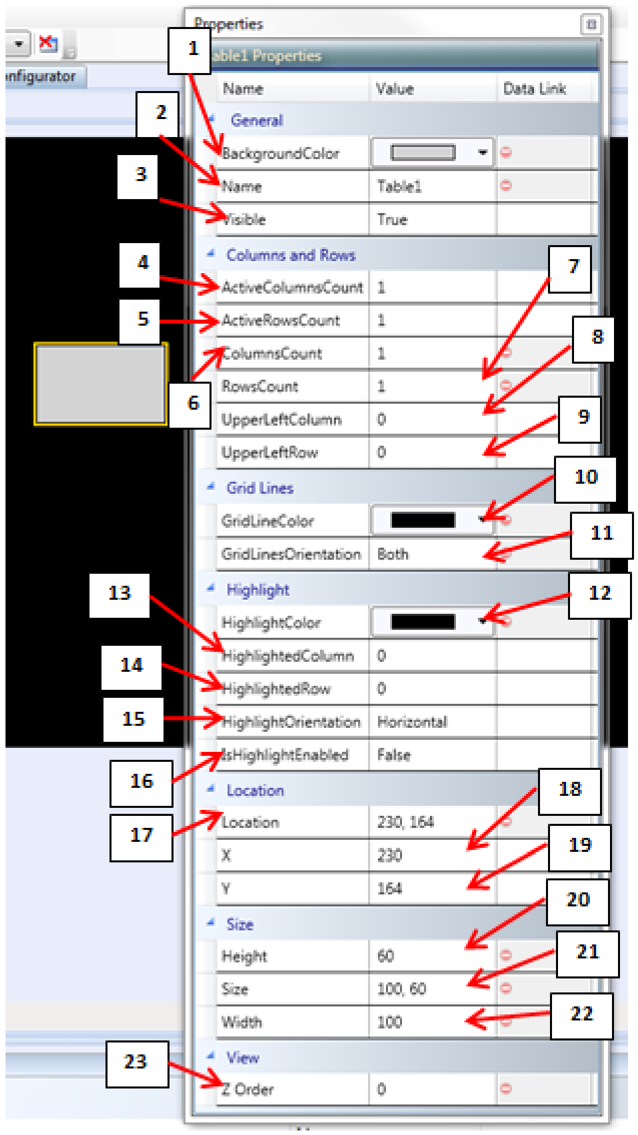


FIGURE 88: CURVED PROGRESS BAR START AND END POINT EXPLANATION

Table

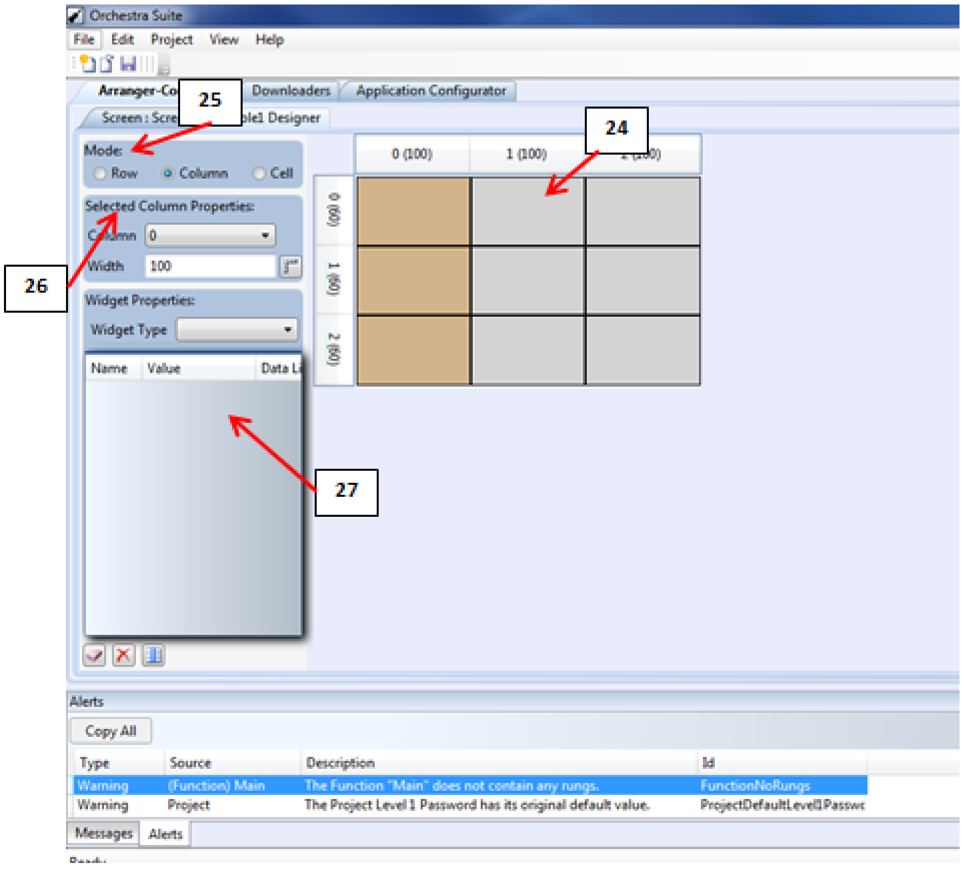
The Table widget will allows for the creation of an Excel-like data table within the screen. What will be shown is a small window, the size of a single cell, into that table that the User will be able to scroll through in order to show what is required. Double clicking on the table widget will bring up the Table Designer tab where the contents of the table can be edited, or like many other things within Orchestra right clicking the item within the Explorer pane and selecting open will accomplish the same task.



able1 Properties

Name	Value	Data Link
General		
BackgroundColor	▭	⊖
Name	Table1	⊖
Visible	True	
Columns and Rows		
ActiveColumnsCount	1	
ActiveRowsCount	1	
ColumnsCount	1	⊖
RowsCount	1	⊖
UpperLeftColumn	0	
UpperLeftRow	0	
Grid Lines		
GridLineColor	■	⊖
GridLinesOrientation	Both	⊖
Highlight		
HighlightColor	■	⊖
HighlightedColumn	0	
HighlightedRow	0	
HighlightOrientation	Horizontal	
IsHighlightEnabled	False	
Location		
Location	230, 164	⊖
X	230	
Y	164	
Size		
Height	60	⊖
Size	100, 60	⊖
Width	100	⊖
View		
Z Order	0	⊖

FIGURE 89: TABLE WIDGET PROPERTIES



Type	Source	Description	Id
Warning	(Function) Main	The Function "Main" does not contain any rungs.	FunctionNoRungs
Warning	Project	The Project Level 1 Password has its original default value.	ProjectDefaultLevel1Passw

1. Background Color
2. Name
3. Visible
4. ActiveColumnsCount
5. ActiveRowsCount
6. ColumnsCount
7. RowsCount
8. UpperLeftColumn
9. UpperLeftRow
10. GridLineColor
11. GridLineOrientation
12. HighlightColor
13. HighlightedColumn
14. HighlightedRow
15. HighlightOrientation
16. IsHighlightEnabled
17. FlashingLocation
18. X
19. Y
20. Height
21. Size
22. Width
23. Z Order
24. Table Area
25. Modes
26. Selected Properties
27. Widget Properties

Downloader

To access the Downloader tool, just left click on the top middle tab that says “Download Project” within the main Orchestra screen.

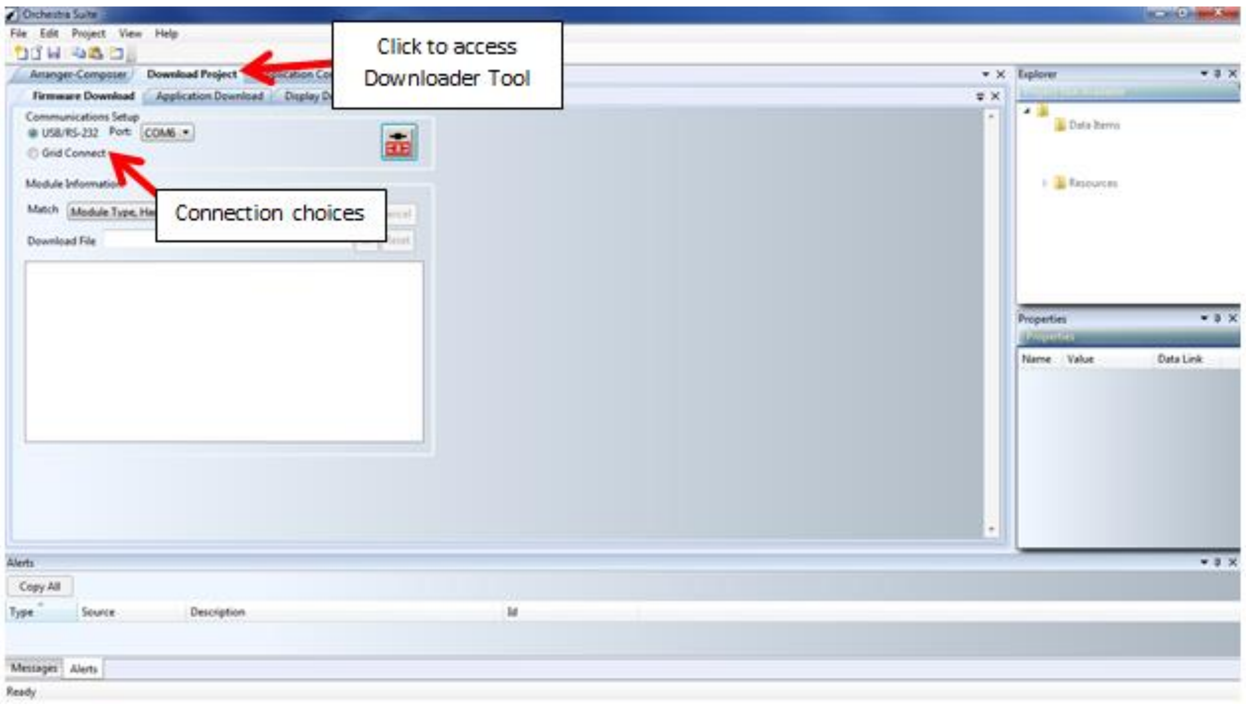


FIGURE 90: DOWNLOADER

Three sub tabs will appear; “Firmware Download”, “Application Download”, and “Display Download”. As their names suggest, they each download specific software to a specific device. Before any downloading can be done, though, the hardware and tools must be connected and configured properly. There are two ways to download to a Module, through a USB/RS-232 connection if the Module has the capability, or through the CAN using a USB to CAN adapter such as Gridconnect. Please note that if the Module is USB capable, Gridconnect can still be used instead if the User wishes to download over CAN.
For the USB connection, a simple USB to USB cable is all that is needed, and after the cable is connected the User must choose the COM port that the Module is connected to via the drop down menu. To connect to the Gridconnect device the CAN line needs to be broken out to a RS-232 connector with the standard CAN pin configuration to plug into the device.
As a note, all of the downloaders can be used whether or not a project is loaded. If no project is loaded then browsing to the correct files to load will be needed.

Firmware Download

To download the firmware first power up the Module(s) then plug in the interface method desired to download with. Once powered and connected, click the “Download Project” tab within Orchestra and then the “Firmware Download” tab. Within that window the User must choose the communication method being used, USB/RS-232 or Grid Connect, then click the connect icon to the right. Please note that if using the USB/RS-232 option, the User needs to also select the port that is being used to communicate to the Module(s).

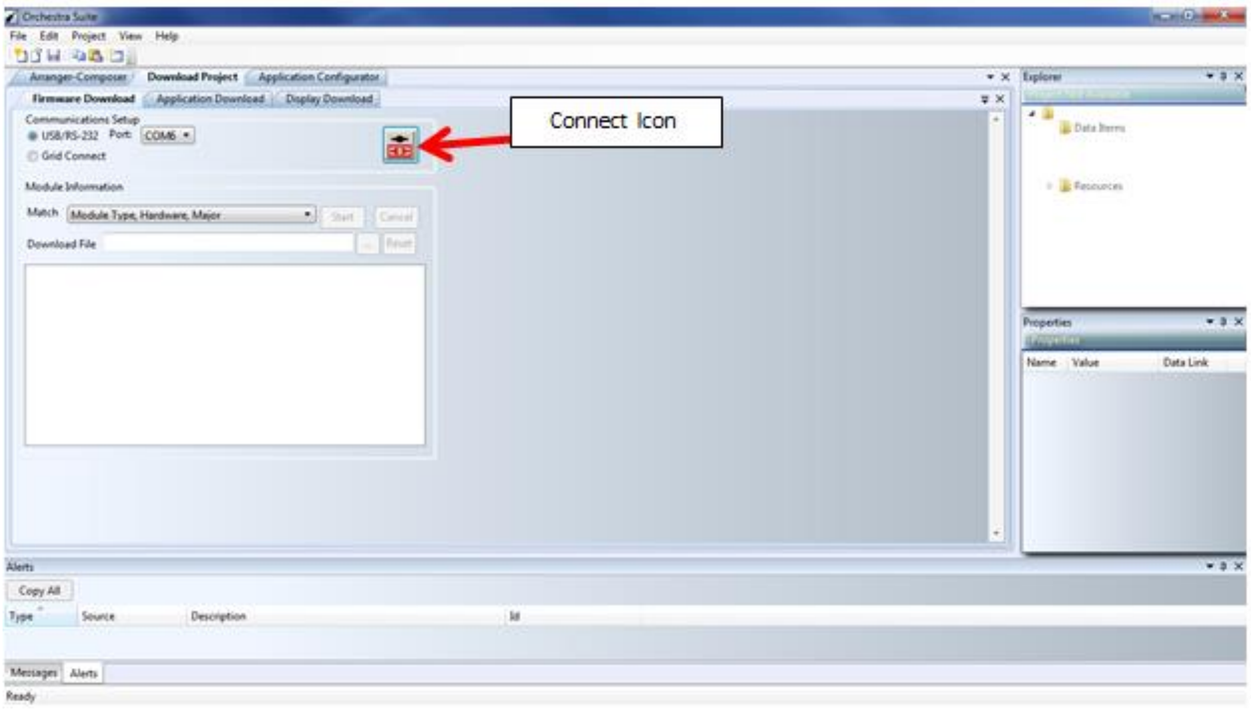


FIGURE 91: FIRMWARE DOWNLOADER

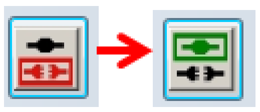
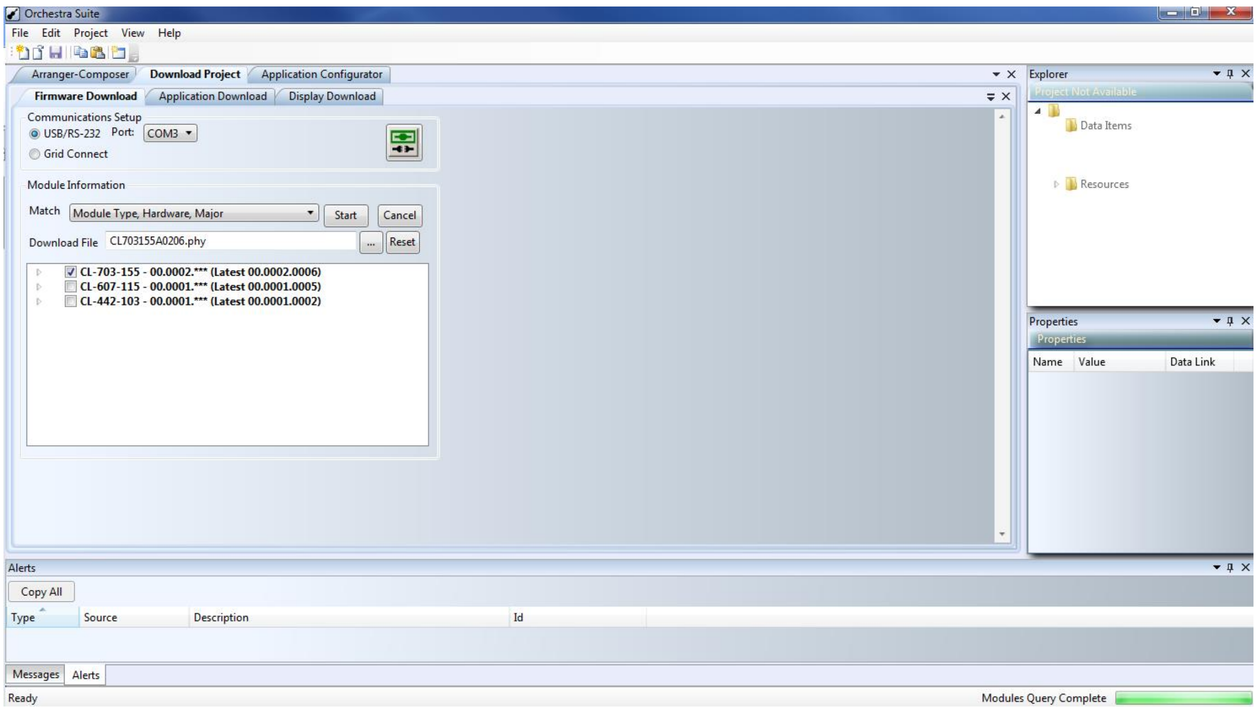


FIGURE 92: DISCONNECTED TO CONNECTED ICON CHANGE

Once connected the icon will turn green and information within the “Module Information” field will be populated with all of the Modules detected within the system. Next to each found Module will be a check box and clicking the check box marks that module for the firmware download. When marked, the “Download File” field will be auto-populated with the latest firmware file that is available with that version of Orchestra. Clicking start will begin the download and Orchestra will notify the User of completion.
For Presto projects the Firmware Downloader will be used to load the Master Module with the application software. Instead of loading the auto-populated firmware after selecting the correct module, use the “…” button to browse to the Presto project’s created .PHY file, select that and then click start to download the software.



Application Download

To download a rungs application power up the Module, then plug in the interface desired to download over, and then click on the “Application Download” to access that downloader. Within that tab, depending if the User has a project loaded or not, there may or may not be a window to select the CLC1 file to download. If there is an open project, that window will not be there and Orchestra will automatically select the open project’s CLC1 file for download. If there is no project open, the User will have to manually select the file to download.

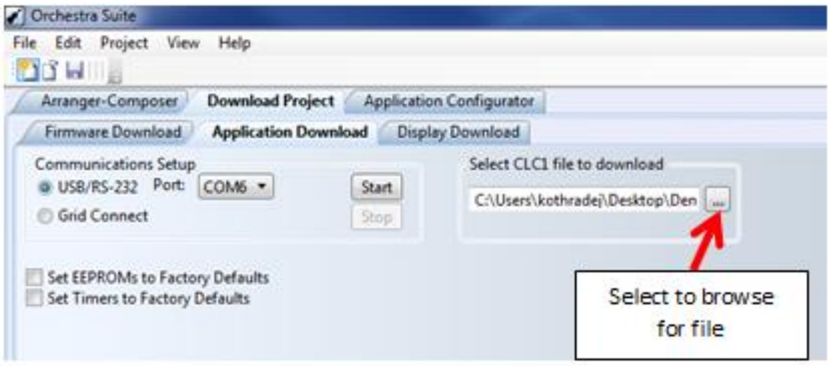
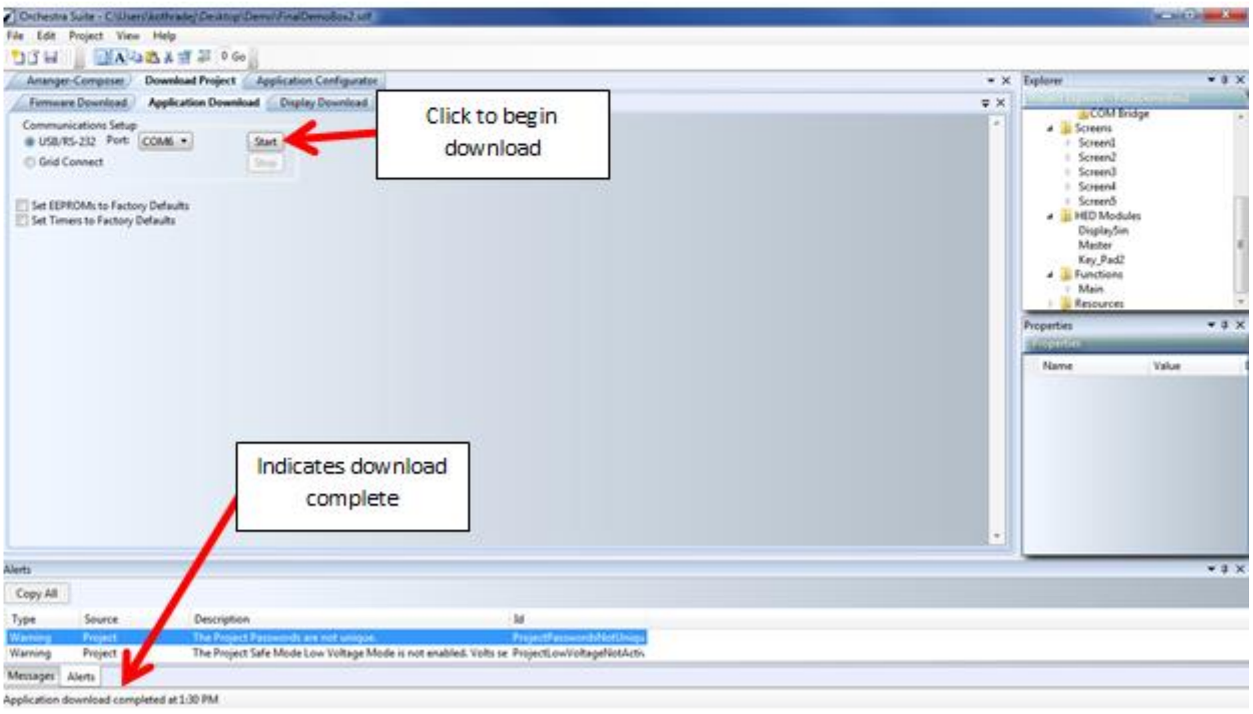


FIGURE 93: APPLICATION DOWNLOADER MANUAL FILE SELECTION

Once the desired CLC1 file is selected, either manually or automatically through Orchestra, choose the communication method that is desired and click the “Start” button. The User can track the download progress via a progress bar in the lower right hand corner of the screen. Once the

download is complete Orchestra will indicate to the User via text in the lower left hand corner of the screen.
There are two check boxes; “Set EEPROMS to Factory Defaults” and “Set Timers to Factory Defaults” that the User can check if they desire to reset the EEPROMs and Timers to their factory default values.


Click to begin download
Indicates download complete
Start
Set EEPROMs to Factory Defaults
Set Timers to Factory Defaults

Type	Source	Description	Id
Warning	Project	The Project Passwords are not unique.	ProjectPasswordsNotUniqu
Warning	Project	The Project Safe Mode Low Voltage Mode is not enabled. Volts se	ProjectLowVoltageNotActiv

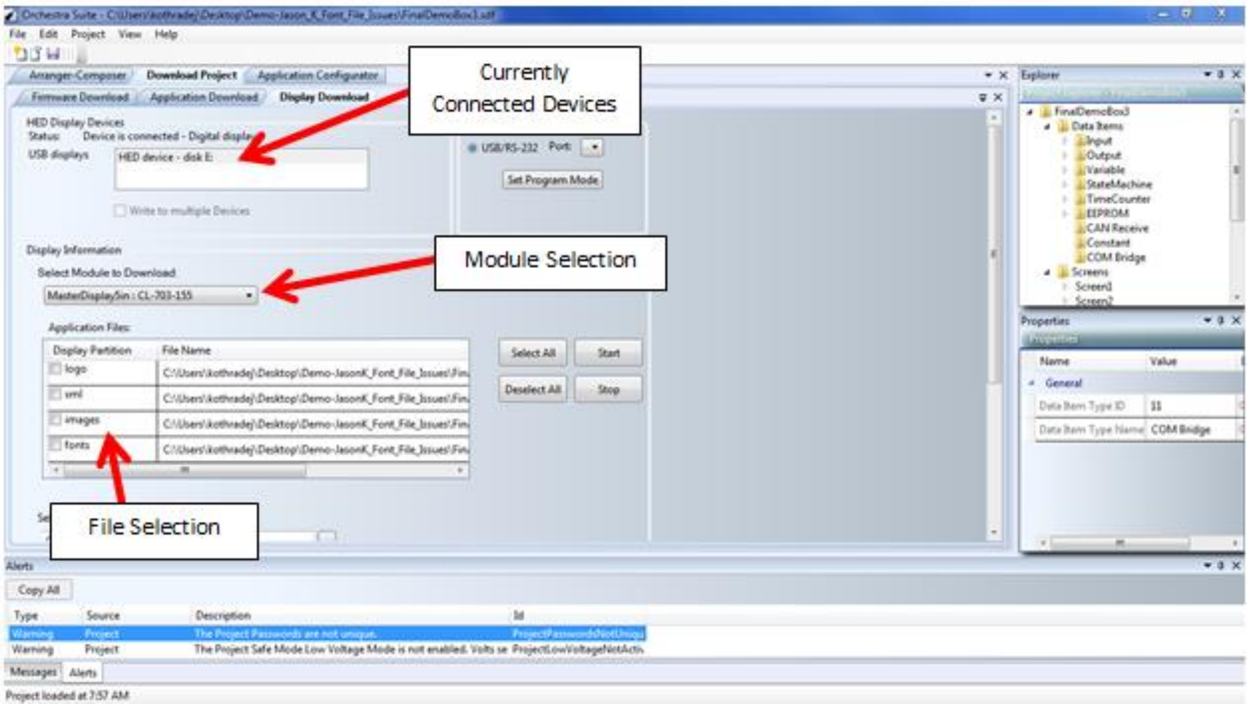
Application download completed at 1:30 PM

FIGURE 94: APPLICATION DOWNLOADER

Display Download

If the display is an I/O Module, it cannot be downloaded to through the Master Module and will require its own USB connection. If the display is a Master Module, there is a built in USB CAN pass through internal to the display that will allow for the display to be programmed with the Application software and the display files using the same connection. There is a specific order in which the USB cable is plugged in, in order to either program the display files or the application files. If the display is a Master and the User desires to flash application software the USB cable must be plugged in after the display has already been powered up. If the User wants to flash the display files the USB must be plugged in prior to powering up the display regardless of being a Master or I/O Module.
For flashing the display files plug the USB in prior to powering up the display, doing so, the User should see a bottom portion of the screen turn black. That band on the bottom alerts the User that the display is ready to be flasWAG. Within the “Display Download” tab the display should appear as a USB device within the “USB displays” field. As a note, the project needs to be loaded to access the rest of the information to properly download the display files.
Under the “Display Information” the User should see a drop down menu that has the option to choose which display to download to if there are multiple displays connected, otherwise the only option will be the single display. Underneath the selection are the “Application Files” that include a logo, xml, images, and fonts selection. Each of those is their own separate files and the paths to

them should be automatically filled. The User can choose which files to load by clicking the check box next to each one. The main file for the display will be the xml file which is the equivalent of the clc1 file that a Master Module needs that holds the actual Application code. Each time a new version of Application software is downloaded the corresponding version of the xml file needs to be downloaded to the display. If there were any changes to the images, fonts, or logo those must be downloaded as well, otherwise they do not have to be and significantly reduces download time if they are not selected for download. Select the files to download and click the start button to begin. Once the files have downloaded and verified, indicated by the bar on the lower right corner of the screen, the USB cable can be unplugged and the display power cycled to resume normal operation.



HED Display Devices
Status: Device is connected - Digital display
USB displays: HED device - disk E:

Select Module to Download: MasterDisplay5in : CL-703-155

Display Partition	File Name
logo	C:\Users\kothrade\Desktop\Demo-JasonK_Font_File_Issues\Fin
xml	C:\Users\kothrade\Desktop\Demo-JasonK_Font_File_Issues\Fin
images	C:\Users\kothrade\Desktop\Demo-JasonK_Font_File_Issues\Fin
fonts	C:\Users\kothrade\Desktop\Demo-JasonK_Font_File_Issues\Fin

Type	Source	Description	Id
Warning	Project	The Project Passwords are not unique.	ProjectPasswordsNotUniqu
Warning	Project	The Project Safe Mode Low Voltage Mode is not enabled. Volts se	ProjectLowVoltageNotActiv

Project loaded at 7:57 AM

FIGURE 95: DISPLAY DOWNLOADER SCREEN

Downloader Wizard Packet

The Downloader Wizard Packet is a way to fully package and download only the files needed for that given project which also provides a form of revision control.

Downloading from a Packet

To download from a packet follow the steps outlined below.

1. Open a new instance of Orchestra
2. Select the file drop down menu and click on "Run Downloader Wizard from Packet File…".

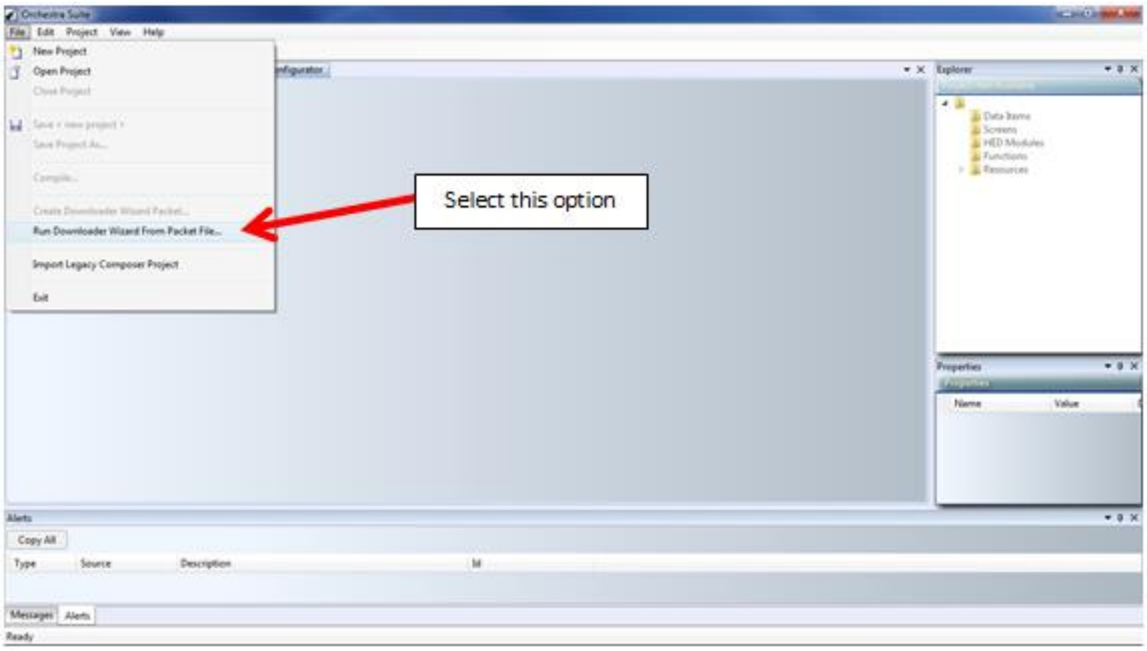
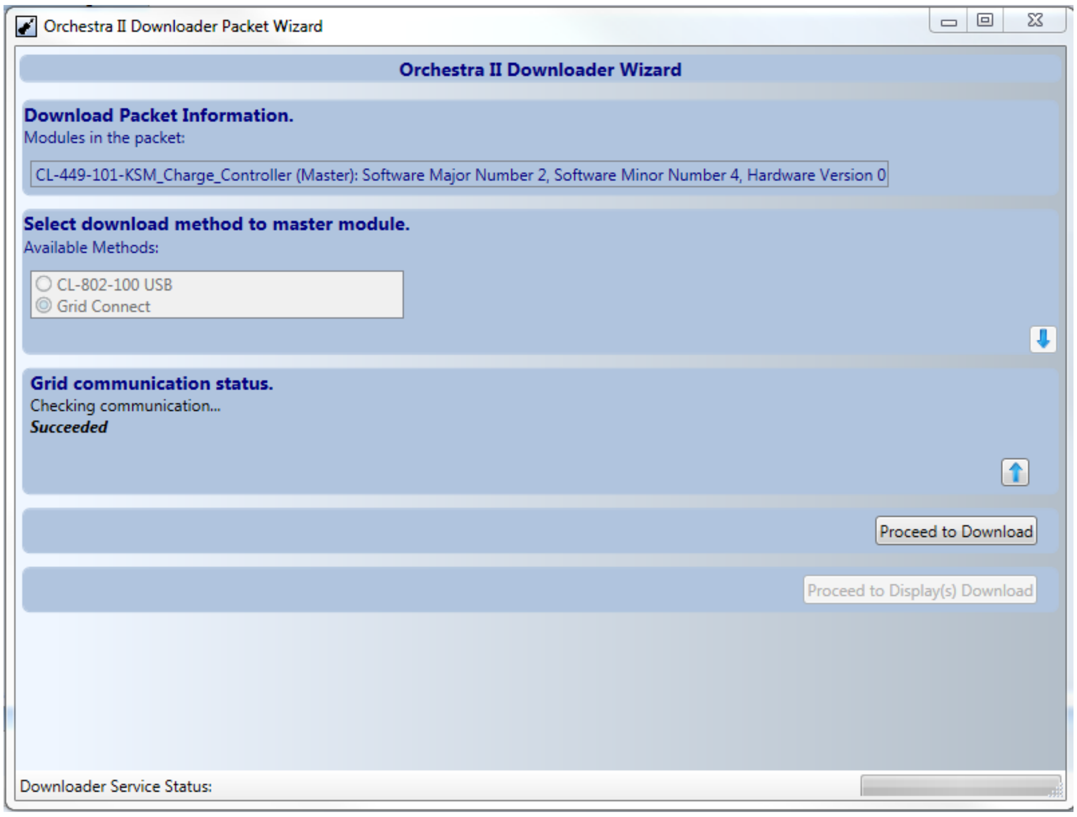


FIGURE 96: BEGIN DOWNLOAD FROM PACKET

3. Navigate to the .dwp file within the computer and select it.
4. The Downloader Packet Wizard will pop up showing the module information that the packet contains. Choose which method of downloading the file is desired from the available options.
5. Click the blue arrow once a selection has been made, and the wizard will check for the communication.


Orchestra II Downloader Packet Wizard

Orchestra II Downloader Wizard

Download Packet Information.
Modules in the packet:

CL-449-101-KSM_Charge_Controller (Master): Software Major Number 2, Software Minor Number 4, Hardware Version 0

Select download method to master module.
Available Methods:

CL-802-100 USB
Grid Connect

Grid communication status.
Checking communication...
Succeeded

Proceed to Download

Proceed to Display(s) Download

Downloader Service Status:

FIGURE 97: DOWNLOADER PACKET WIZARD

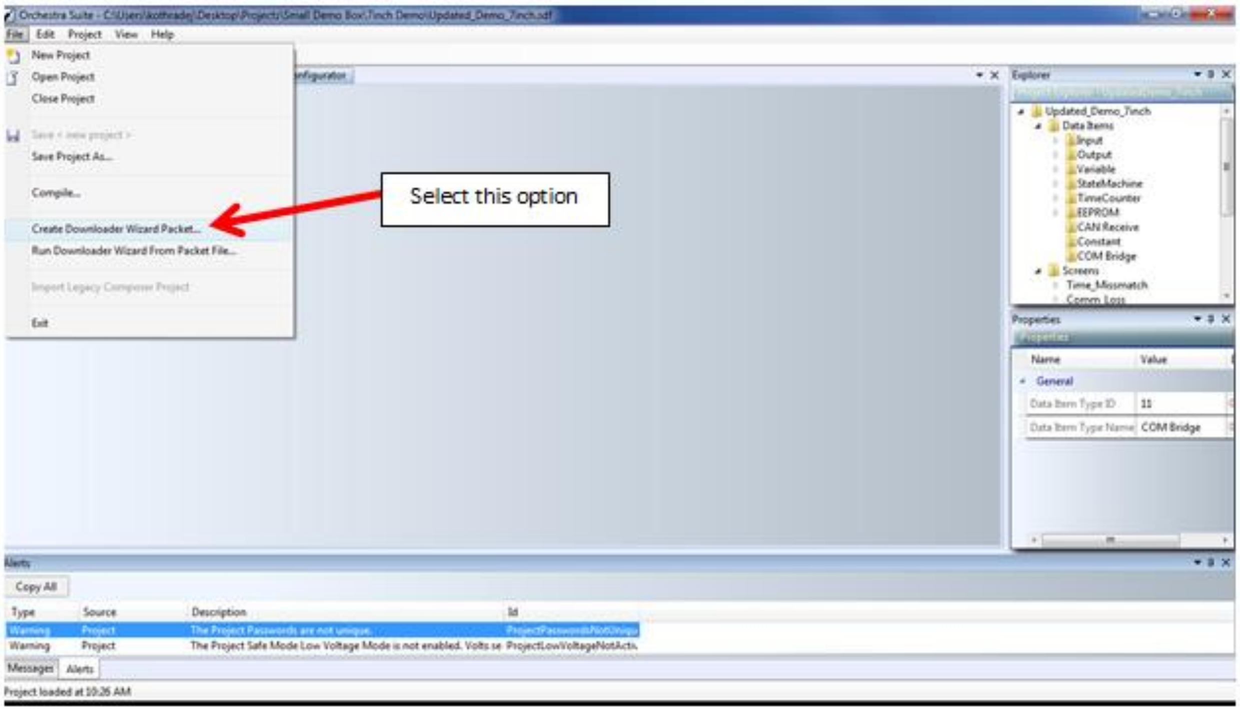
6. Once connected click Proceed to Download and the download process will begin. The progress can be viewed by the status bar in the lower right hand corner.
7. Once the application is loaded, if there are also files to download to a display the User will be prompted to do so.

Creating a Packet

To create a Downloader Wizard Packet follow the steps below.

1. Open the project within Orchestra
2. Go to the file drop down and select “Create Downloader Wizard Packet…”.
3. Select which download methods will be available to choose from when downloading. Please note the first General Notes section gives a small warning about downloading incompatible software and it is advised this be read. The subsequent General Notes tabs give brief overviews of that section.
4. Select the firmware to load into the master module, either the default or a specific file specified by the User. If this is selected when the download wizard is run it will update the firmware of the master module if necessary. Please note that if you do not select the firmware to be downloaded there is a chance that the master module will not support the .clc1 format of the application. It is suggested that you use the default software as this is the latest known firmware for the module and will be compatible with the .clc1 file generated with this version of Orchestra.
5. Select the .clc1 file (if rungs) to download to the module. The default .clc1 file is the one generated from the open Orchestra project.
6. If a display is part of the project select all of the files related to the display that is desired to be included with the download packet. For a brief description of each file please refer to the General Notes drop down within the Select Master Module Display Software pane of the window.
7. Select I/O Module software if it is desired to have the I/O Modules firmware updated. The file to be loaded to the modules can be specifically chosen if the default is unchecked and a new file is selected.
8. Click Save and Quit once everything that is desired to include within the packet is chosen to create the .dwp file and close the Downloader Packet Wizard, or click Save and Download to save the .dwp file and immediately download the packet.

If there are any required fields that are empty or improperly filled there will be an error box giving a brief description of the issue. Note that when creating a .dwp for a Presto project Step 4 is where the file to be loaded is selected instead of the firmware file.

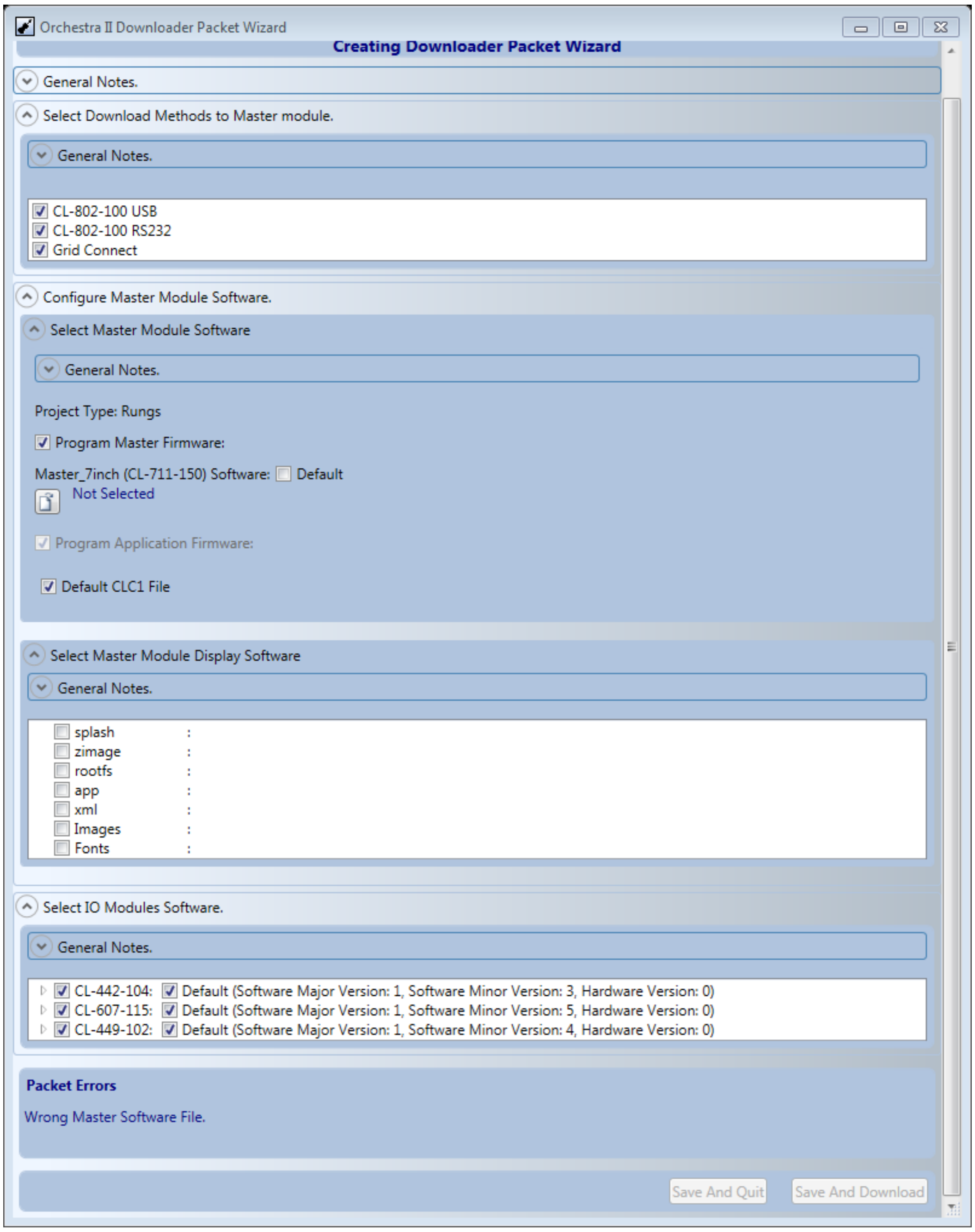


Name	Value
Data Item Type ID	11
Data Item Type Name	COM Bridge

Type	Source	Description	Id
Warning	Project	The Project Passwords are not unique.	ProjectPasswordsNotUniqu
Warning	Project	The Project Safe Mode Low Voltage Mode is not enabled. Volts se	ProjectLowVoltageNotActi

Project loaded at 10:26 AM

FIGURE 98: BEGIN DOWNLOAD PACKET CREATION


Orchestra II Downloader Packet Wizard
Creating Downloader Packet Wizard
General Notes.
Select Download Methods to Master module.
General Notes.

☑ CL-802-100 USB
☑ CL-802-100 RS232
☑ Grid Connect

Configure Master Module Software.
Select Master Module Software
General Notes.

Project Type: Rungs

☑ Program Master Firmware:

Master_7inch (CL-711-150) Software: ☐ Default

Not Selected

☑ Program Application Firmware:

☑ Default CLC1 File

Select Master Module Display Software
General Notes.

☐ splash	:
☐ zimage	:
☐ rootfs	:
☐ app	:
☐ xml	:
☐ Images	:
☐ Fonts	:

Select IO Modules Software.
General Notes.

☑ CL-442-104:	☑ Default (Software Major Version: 1, Software Minor Version: 3, Hardware Version: 0)
☑ CL-607-115:	☑ Default (Software Major Version: 1, Software Minor Version: 5, Hardware Version: 0)
☑ CL-449-102:	☑ Default (Software Major Version: 1, Software Minor Version: 4, Hardware Version: 0)

Packet Errors

Wrong Master Software File.

FIGURE 99: DOWNLOADER PACKET WIZARD PACKET CREATION

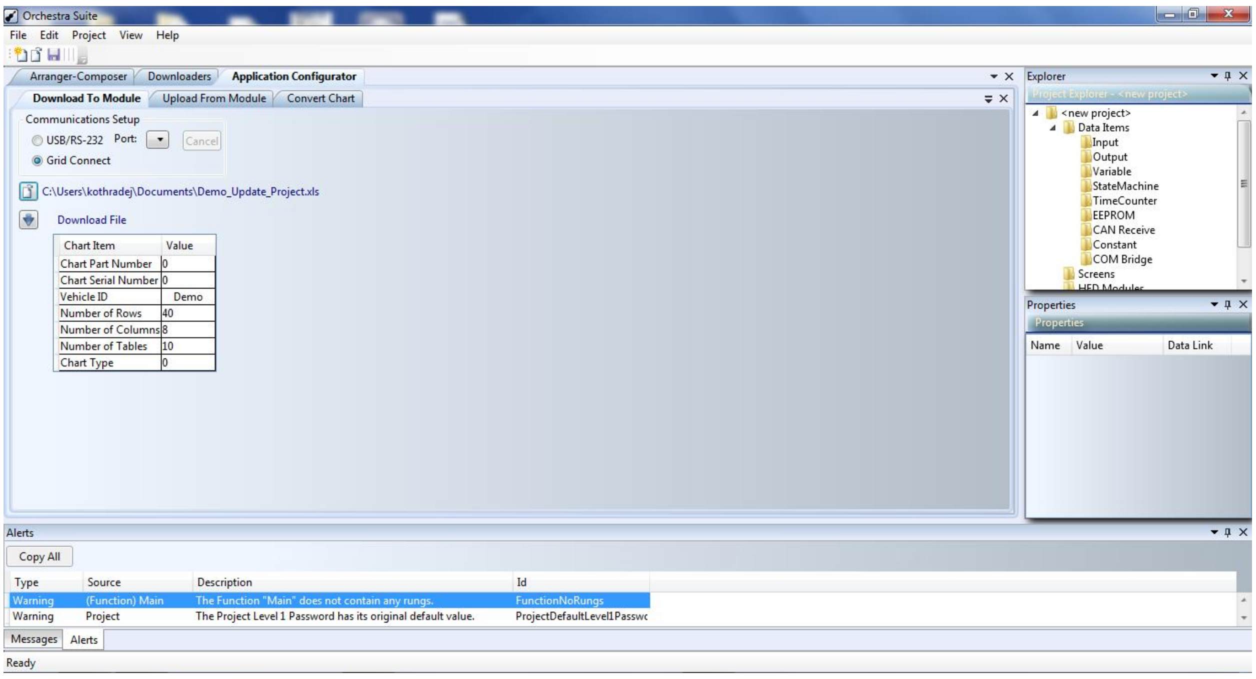
Application Configurator

Application Configurator is a feature within Orchestra that allows the user to download and upload configurable charts to and from modules, respectively. It can also do a conversion to a chart to provide a kind of revision control that can prevent the given chart from being edited.

Download To Module

To download a chart to a module, follow the steps below.

1. Open Orchestra and click the Application Configurator Tab.
2. Within the Application Configurator tab click on the Download To Module tab.
3. Choose the connection method to be used, either through CAN using Gridconnect or through USB.
4. Click the small button below the communication set up to navigate to the chart that is to be downloaded.
5. Once the chart is selected and loaded within Orchestra click the download file button.
6. After the download completes it is safe to disconnect the module.



Communications Setup
USB/RS-232 Port: Cancel
[✓ Grid Connect]

C:\Users\kothradej\Documents\Demo_Update_Project.xls

Download File

Chart Item	Value
Chart Part Number	0
Chart Serial Number	0
Vehicle ID	Demo
Number of Rows	40
Number of Columns	8
Number of Tables	10
Chart Type	0

Alerts

Type	Source	Description	Id
Warning	(Function) Main	The Function "Main" does not contain any rungs.	FunctionNoRungs
Warning	Project	The Project Level 1 Password has its original default value.	ProjectDefaultLevel1Passwc

FIGURE 100: DOWNLOAD TO MODULE

Upload From Module

To upload a chart from a module, follow the steps below.

1. Open Orchestra and click the Application Configurator Tab.
2. Within the Application Configurator tab click on the Upload From Module tab.
3. Choose the connection method to be used, either through CAN using Gridconnect or through USB.
4. Choose the chart type that will be uploaded from the module.
5. Select a file for the chart to be uploaded into, usually an Excel file. After choosing, the Configurator will attempt to upload the chart.

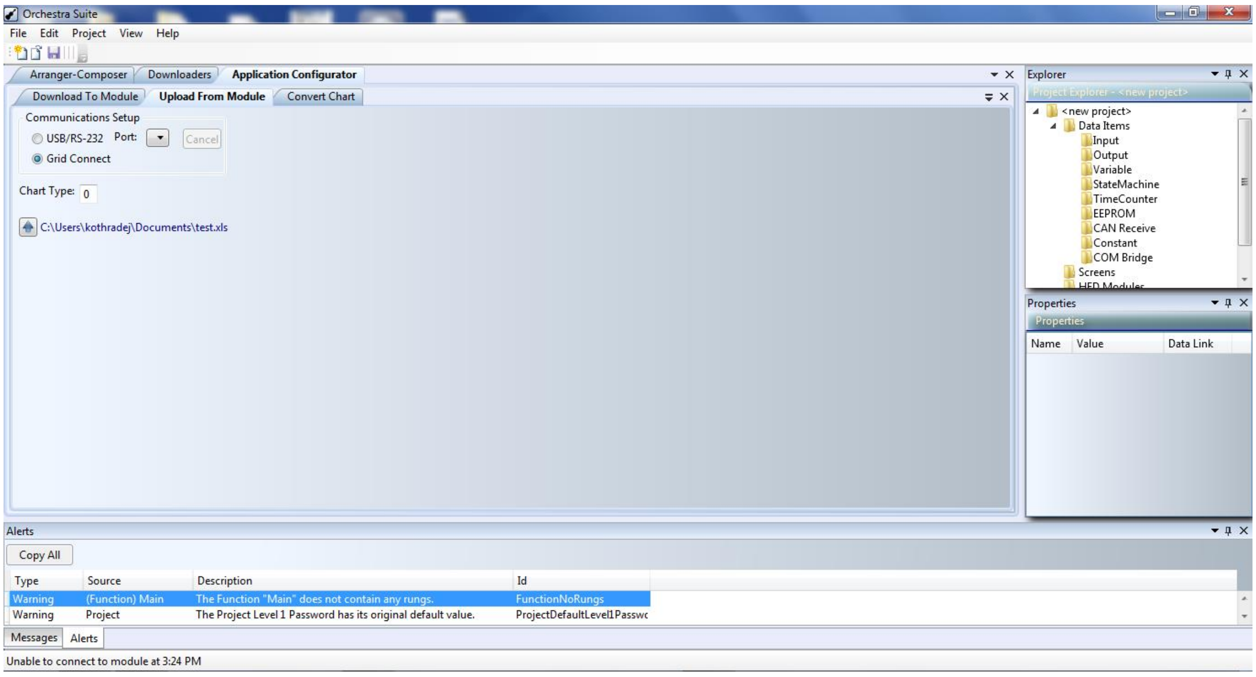


Chart Type: 0

C:\Users\kothradej\Documents\test.xls

Type	Source	Description	Id
Warning	(Function) Main	The Function "Main" does not contain any rungs.	FunctionNoRungs
Warning	Project	The Project Level 1 Password has its original default value.	ProjectDefaultLevel1Passwc

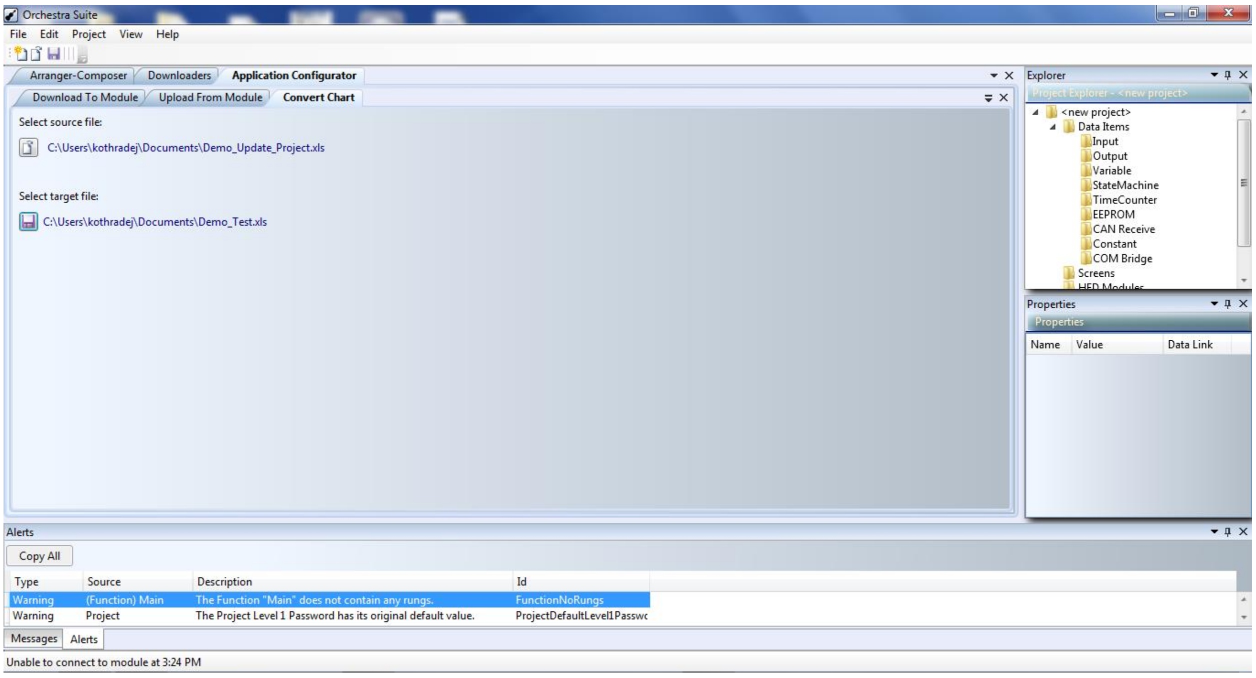
Unable to connect to module at 3:24 PM

FIGURE 101: UPLOAD FROM MODULE

Convert Chart

Converting a chart can be done by following the steps below.

1. Open Orchestra and click the Application Configurator Tab.
2. Within the Application Configurator tab click on the Convert Chart tab.
3. Click the button to browse to the file that is to be converted, and select it. The Configurator will prepare the file to be converted.
4. Once the file to be converted has been chosen and prepared, another button will appear. Click it and browse to the location where the newly created file will be placed, choose the format for the file to be converted to, and give the new file a name.



Select source file:

C:\Users\kothradej\Documents\Demo_Update_Project.xls

Select target file:

C:\Users\kothradej\Documents\Demo_Test.xls

Type	Source	Description	Id
Warning	(Function) Main	The Function "Main" does not contain any rungs.	FunctionNoRungs
Warning	Project	The Project Level 1 Password has its original default value.	ProjectDefaultLevel1Passwc

Unable to connect to module at 3:24 PM

FIGURE 102: CONVERT CHART

Presto

Orchestra's Presto™ enables a user to use all the features Orchestra provides and write their application or part of the application using the C programming language instead of only Orchestra ladder logic. If a Presto and Rungs project is being made, keep in mind that the C code will be executed first and then the Rungs code.
The low level CPU drivers, such as communication protocol, diagnostic tools, I/O management, etc., are all provided as precompiled object code. The user is responsible for setting up a system in Orchestra and using CodeWarrior™ to add user specific code to the C project and compile it. Presto™ is intended for users with C programming experience.

Material

Required

1. WAG Blue Dongle with Presto™
   a. WAG Part Number: CL-012-304
2. Freescale CodeWarrior™ For HCS12(X) Version 4.7 with appropriate license
   a. Not included with Orchestra. Needs to be ordered through Freescale

Optional

1. Wire Harness or Switch Box
   a. Contact your supplier for ordering information
   b. Create your own
2. CL-802-100
   a. Only required if the master module you are using does not have RS232 or USB
   b. Can be used even if the master module does have RS232 or USB
3. P&E Multilink
   a. Used for debugging software through JTAG
   b. If this is not purchased module can still be programmed using Orchestra tools however debugging your software is more difficult
   c. Not included with Orchestra. Needs to be ordered through P&E Micro
   d. Note: opening the module to plug the Multilink in will void the module's warranty

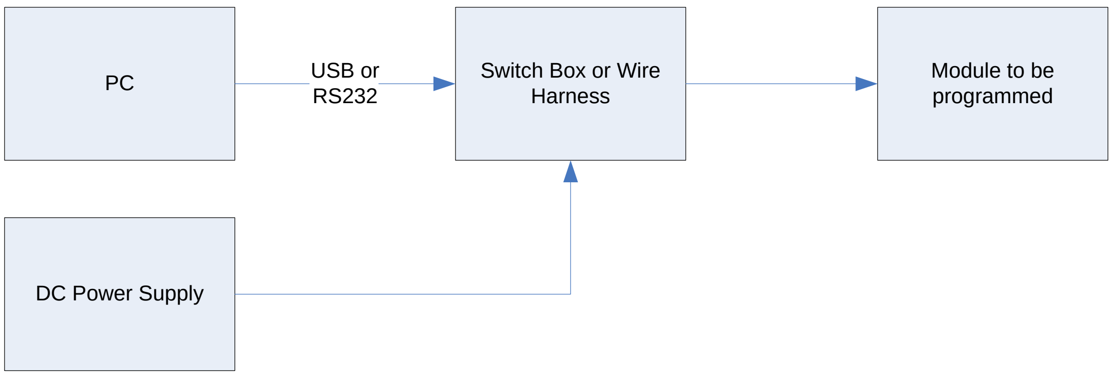


FIGURE 103: CONNECTION WHEN MASTER MODULE HAS RS232 OR USB

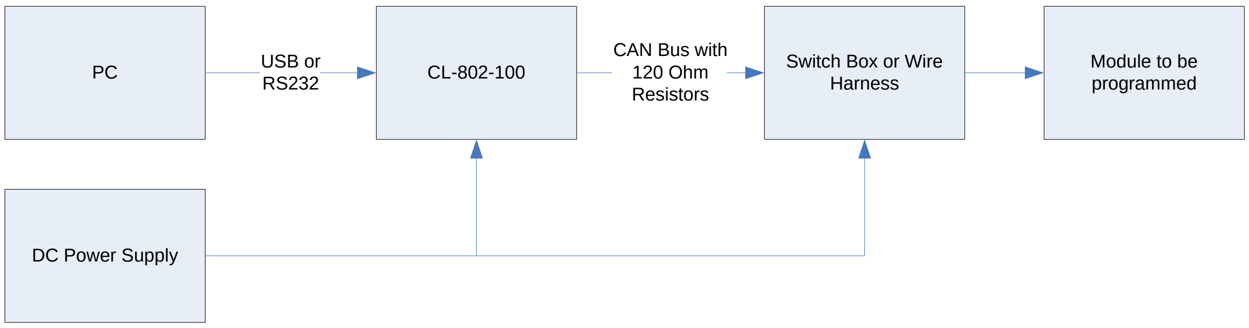
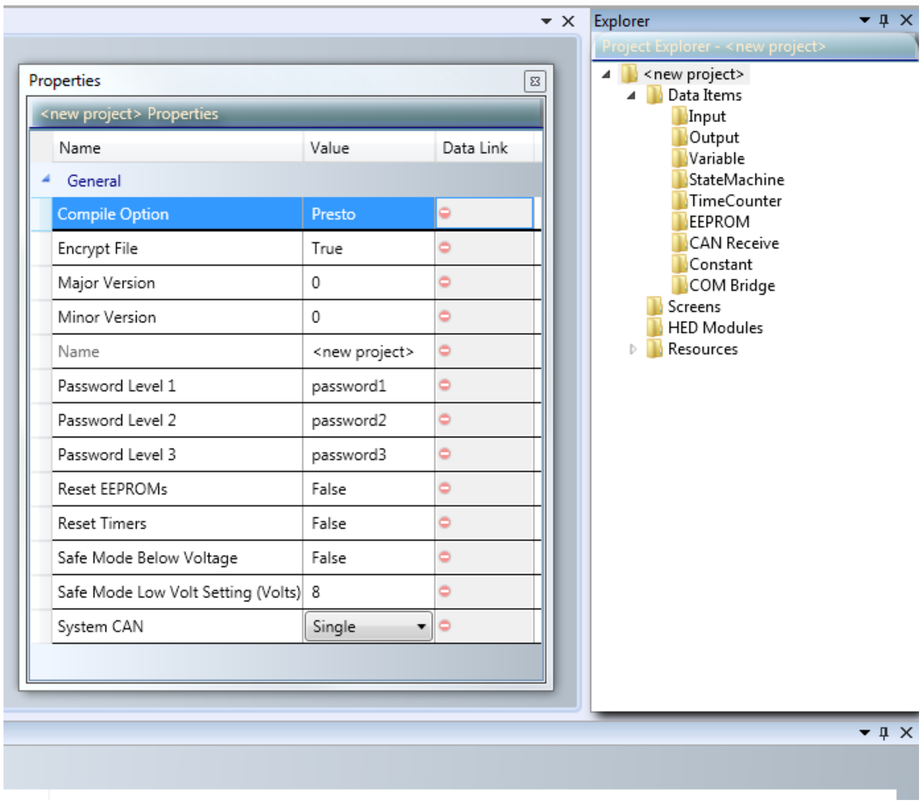


FIGURE 104: CONNECTION WHEN MASTER MODULE DOES NOT HAVE RS232 OR USB

Setup

Using Orchestra to create header files for a C project is almost identical to a standard Orchestra only project. The only difference is the Compile Option has to be changed in the project properties by clicking on the project and adjusting the option through the drop down menu.

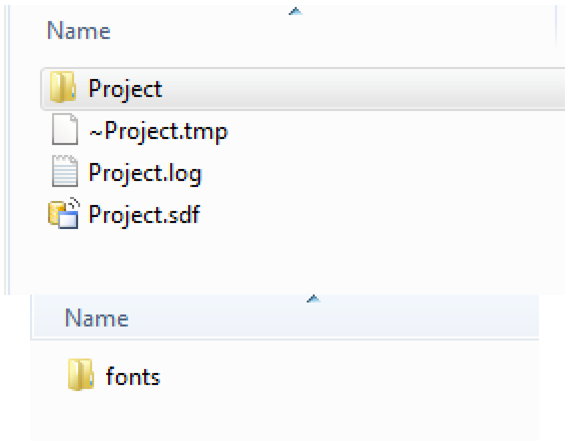


Name	Value	Data Link
General		
Compile Option	Presto	
Encrypt File	True	
Major Version	0	
Minor Version	0	
Name	<new project>	
Password Level 1	password1	
Password Level 2	password2	
Password Level 3	password3	
Reset EEPROMs	False	
Reset Timers	False	
Safe Mode Below Voltage	False	
Safe Mode Low Volt Setting (Volts)	8	
System CAN	Single	

FIGURE 105: SELECTING COMPILE OPTION

Compiling a Presto Project in Orchestra

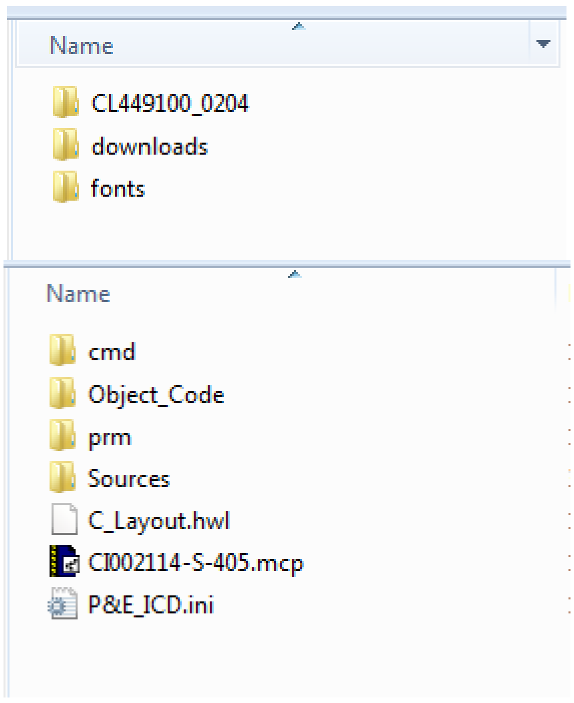
Orchestra's Compiler creates a folder within the project's folder, named after the master module, that all of the files needed to begin writing in CodeWarrior™ including the "Constants.h" and "Constants.c" files. The "Constants.h" file contains the #defines where all the data items are in the IOMap. The "Constants.c" file contains 3 constant arrays that the firmware will use to set up all the modules in the system and run the rungs if activated.



Name
Project
~Project.tmp
Project.log
Project.sdf

Name
fonts

FIGURE 106: PRE-COMPILE



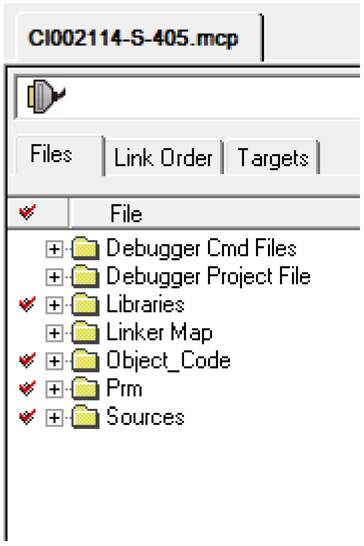
Name
CL449100_0204
downloads
fonts

Name
cmd
Object_Code
prm
Sources
C_Layout.hwl
CI002114-S-405.mcp
P&E_ICD.ini

FIGURE 107: POST COMPILE

The C Project

Opening the project within CodeWarrior™, the starter project and files that have been provided will appear in the pane off to the left of the screen.



Debugger Cmd Files
Debugger Project File
Libraries
Linker Map
Object_Code
Prm
Sources

FIGURE 108: CODEWARRIOR™ PROJECT PANE

1. Debugger Cmd Files
   a. These files are used to setup the P&E debugger.
   b. These files should not need to be edited however there is information available from CodeWarrior™ should they need to be edited.
2. Debugger Project Files
   a. These files are used to setup the P&E debugger.
   b. These files should not need to be edited however there is information available from CodeWarrior™ should they need to be edited.
3. Libraries
   a. These files are for the processor.
   b. These files should not be edited.
4. Linker Map
   a. Contains the memory map after the project has successfully linked.
5. Object_Code
   a. Contains all of the .o files associated with the project.
   b. Files cannot be edited.
6. Prm
   a. Burner.bbl
      i. Command for how to build the s19 and phy files.
      ii. This file should not be edited.
   b. P&E_ICD_linker.prm
      i. This is the linker file for the project. It setups up where all the memory is and where all the different sections of the project go.
      ii. If the processor uses paged memory, 1 page of RAM is reserved for the User application. Therefore this file should not be edited.
7. Sources Folder
   a. Precompiled Folder

   i. Precompiled object files for all the low level firmware.
   ii. Do not edit these files.
b. Constants .h and .c files
   i. The Orchestra Compiler created these files to match the system that was setup in Orchestra.
   ii. Do not edit these files.
c. User_Init.h
   i. Function Prototype for User_Init().
   ii. Add user software when applicable.
d. User_Init.c
   i. Function is called once on startup.
   ii. Add user software when applicable.
e. User_App.h
   i. Function Prototype for User_App().
   ii. Add user software when applicable.
f. User_App.c
   i. Function is called once per loop.
   ii. Add user software when applicable.
g. User_Can_Receive.h
   i. Function Prototype for User_Can_Receive().
   ii. Add user software when applicable.
h. User_Can_Receive.c
   i. Function is called once for every CAN message thadt id not come from a module on the system.
   ii. Add user software when applicable.
i. User_J1708_Receive.h
   i. Function Prototype for User_J1708_Receive().
   ii. Add user software when applicable.
   iii. Note: This file will only be available on modules that have J1708.
j. User_J1708_Receive.c
   i. Function is called once for every J1708 message received.
   ii. Add user software when applicable.
   iii. Note: This file will only be available on modules that have J1708.
k. User_Serial_Receive.h
   i. Function Prototype for User_Serial_Receive().
   ii. Add user software when applicable.
   iii. Note: This file will only be available on modules that have RS232 or USB.
l. User_Serial_Receive.c
   i. Function is called once for every byte received on any of the SCI lines on the module if the WAG packets are disabled.
   ii. Add user software when applicable.
   iii. Note: This file will only be available on modules that have RS232 or USB.
m. User_Serial_Packet_Receive.h
   i. Function Prototype for User_Serial_Packet_Receive().
   ii. Add user software when applicable.
   iii. Note: This file will only be available on modules thta have RS232 or USB on them.
n. User_Serial_Packet_Receive.c
   i. Funcition is called once for every packet received on any of the SCI lines on the module if the WAG packet is enabled.
   ii. Add user software when applicable.
   iii. Packet has to be in the following format [*….] where … is the ACII characters with the user data.
   iv. Note: This file will only be available on modules that have RS232 or USB .

o. Add additional source and header files as necessary.

Compiling and Debugging a C Project

The C project can be compiled and debugged just like a normal CodeWarrior™ project. The CodeWarrior™ documentation has a very thorough explanation of this. Keep in mind that when using the debugger it will erase all the FLASH including the boot code so the WAG downloader will not work correctly. See the following Reloading the Boot Code section on how to reload the boot code.

Reloading the Boot Code

To reload the boot code after a debug session, please refer to PE Micro's website regarding support of using the Prog12z program to download and restore software files to a module. Once the boot code has been reloaded the module will work properly with Orchestra's Downloader. Follow the Downloader section to download software to the module as needed.

Software Notes

Below are some things to consider when writing the application software:

1. Flash
    a. To reserve a section of flash for data logging, remove it from the linker file. This will ensure that the compiler will not place any of the application code there.
        i. Comment out the pages to be used in the paged flash section where it lists all the pages and their range
        ii. Remove or comment out the pages in the DEFAULT_ROM Section.
    b. Use caution when erasing or writing to addresses
        i. Using an address that does not exist will cause the processor to res et
        ii. Erasing an address with application code will cause it to on long function properly
2. Serial
    a. All WAG PC programs connections start at 9600 baud rate. If the application changes the baud rate to something other than 9600 ensure that it restores it to 9600to allow WAG tools to connect.
    b. Changing the baud rate when a WAG tool is connected will cause the tool to timeout and disconnect.
    c. WAG Serial Packet scheme
        i. Start Byte = [
        ii. End Byte =]
        iii. User Packet command = *
        iv. Example: [*123]
    d. printf function
        i. The printf function needs to have the serial line, to send on, defined prior to be used in the application code.
        ii. Set "SciLineForPrintf" to the desired line.

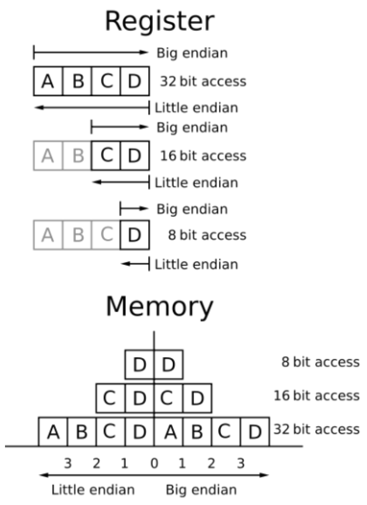
Appendix

- *A* – Target Data Item that will have its value updated based on the outcome of the Operator Block
- *ActiveColumnsCount* – This will limit the number of columns the User will be able to scroll through to only what is active or populated at that time.
- *ActiveRowsCount* – This will limit the number of rows the User will be able to scroll through to only what is active or populated at that time.
- *Array* – Clicking this brings up a drop down menu to edit an array that will be associated with the Variable.
- *Background Color* – Choose the color of the screen background with this option.
- *Background Image* – If an image is desired for the background, clicking the "…" button will allow the User to select an image from a location on their computer.
- *Border Color* – Choose the color of the border surrounding the label.
- *Border Style* – Choose the style of border the label possesses.
- *Border Width* – Determines how thick the border is.
- *Brightness* – This value controls how bright the video will appear in a range of 0 to 255 where 255 is the brightest.
- *Byte* – Appears when Data Byte Filter is Enabled, input what data to filter on.
- *CAN Line* – Select the CAN line in which the message is being received on.
- *CC Offset* – This property sets the value of the duty cycle for the current closed loop control to begin at when going from a command of 0 to a non-zero command.
- *Center Base Color* – This property controls the color of the circular base of the needle that lies below the needle hub.
- *Center Base Diameter* – Change the size of the center base, in pixels.
- *Center Deadband* – The amount of movement that is required from the center position to consider the Input active
- *Center Hub Color* – Property to determine the color of the needle hub.
- *Center Hub Diameter* – Change the size of the hub, in pixels.
- *Center X* – Determines the horizontal position of the center of the needle hub within the widget, in pixels.
- *Center Y* – Determines the vertical position of the center of the needle hub within the widget, in pixels.
- *CenterLocation* – Determines the horizontal and vertical positions that the center of the needle hub resides within the widget.
- *Channel* – This property directs the widget to the channel in which the video feed will be coming from.
- *Color Depth* – Up to 16 colors can be shown.
- *Color Saturation* – This value controls the saturation of the colors within the video in a range of 0 to 255 where 255 is highest saturation level.
- *ColumnsCount* – Defines the number of columns the table will have.
- *Contrast* – This value controls the video contrast in a range of 0 to 255 where 255 is the highest level of contrast.

- *Corner Radius* – Changing this value will affect how rounded the corners of the label appear to be.
- *Current Feedback Type* – Can choose from Single wire or Dual wire to measure the current feedback of the Output.
    a. Single Wire – This mode uses an approximation when determining the current feedback for use in a current controlled application. This method is simpler and less I/O intensive as it only uses a single wire connection on the Module, but it can be inaccurate based on approximations. Use this mode if the current control does not need to be precise.
    b. Dual Wire – This mode uses another I/O connection to receive the current feedback from the device connected to the Output. With a return line for the current, a more precise measurement can be made and Output adjusted for a current controlled application. Use this mode if precision is critical.
- *Current Report Rate (0-2550mS)* – When the Diagnostic Requirement check box for Over current is checked this option appears. This value determines how often, in milliseconds, that the current on the Output is reported for monitoring. Unless it is absolutely necessary, it is generally preferred to keep the rate at 2550mS.
- *D gain* – The derivative gain associated with the PID loop
- *Data Byte Order* – This property allows the User to reorder the bytes to be transmitted from the bytes that were received.
- *Data Byte Filtering* – Enables the ability to filter a message by a specific byte. When this is enabled another field will show in which the User will be able to create a mask to allow only the desired bytes of information through.
- *Data Length* – Appears if the Data Length Adjustment is Enabled, allows User to define the length of the message to be transmitted.
- *Data Length Adjustment* – Enable or disable User defined message data length.
- *Data Parsing Type* – This field allows the User to choose the format of the desired message in either Bits or Bytes.
- *Data Resolution* – Adjust the frequency resolution of the signal from 0.01, 0.05, 0.1, 0.5, 1, and 2Hz.
- *Debounce OFF (msec)* – The amount of time the input must be open before it will switch from the active state to the inactive state.
- *Debounce ON (msec)* – The amount of time the Input must be not open before it will switch from the inactive state to the active state.
- *Default Array Value* – This is the global default value for all elements within the array that have not had a default value manually set already.
- *Default Rx Status* – On power up this is the default status of the CAN receive.
- *Default Rx Value* – On power up this is the default value of the CAN receive.
- *Default Value* – Sets the default value of that element within the array.
- *Delay (0-2550 mS)* – When the Diagnostic Requirement check box for Over Current is checked this option appears. This option lets the User select how long the Output can be in an overcurrent condition before it blows the Digital Fuse and turns the Output to the Device Off.
- *Diagnostic Requirements* – This field has a drop down box of 5 requirements that can be monitored for diagnostics if so desired.

a. Short to Battery – Sets a diagnostic if the Output has detected it is shorted to "Battery".
b. Short to Ground – Sets a diagnostic if the Output has detected it is shorted to "Ground".
c. Open When Off – Sets a diagnostic if the Output has detected an Open condition when commanded Off.
d. Open when On – Sets a diagnostic if the Output has detected an Open condition when commanded On.
e. Over Current – Sets a diagnostic if the Output detects an overcurrent condition. Additional options appear when this diagnostic is selected; Current Report Rate, Digital Fuse Delay, and Digital Fuse Set Point.

- *Direction* – This option allows the User to decide if the Counter will increment (count up from the default to max) or decrement (count down from the max to 0).
- *Direction CAN* – Selects the way in which the message was transmitted and stored into memory; either LSB->MSB (least significant byte to most significant byte) or MSB->LSB. Below is an example of the difference for Bytes A B and C.



a. If the Parsing Type is Bits then there is no Direction field, instead there is a Start Bit field that the User is able to choose at which bit in the message to begin reading.

- *Display Format* – The User is able to choose the display format of the message in either Hex or decimal.
- *Display Index* –
- *Display Model* – This selects the model of display for which the screen will be associated with.
- *End Point (degrees)* – Dictates the angle that the needle will travel when at its max value.
- *End Point (Pixel)* – Define how many pixels from the edge of the widget that the bar will end i.e. a value of 0 will fill the bar completely and a value of 10 will fill the bar until the edge is 10 pixels away from the edge of the widget.
- *End Value* – The value that the needle widget will be when it reaches the end point.
- *Fill Color* – Choose the color that the bar will be to fill the widget.
- *Fill Direction* – Choose the direction that the bar will fill up, in a horizontal orientation it can fill left to right or right to left while in a vertical position it can be filled top to bottom or bottom to top.

- *Fill Start Point (Pixel)* – This is the point that the bar will begin filling from in reference to the start point.
- *Filter Size* – Used only for Running Average, and sets how many samples, taken independent of the Report Rate, to average together in order to obtain a value to report back.
- *Filter Type* – VTD also contains two options to filter the voltage coming from the pin in order to obtain a usable value; Running Average and Min Max Average. The Running Average has another property, Filter Type, associated with it which is explained in point 7. The Min Max Average averages the Min and Max voltages read on the pin, and places that value in the Data Item each time the Input value changes direction.
- *Flash* –

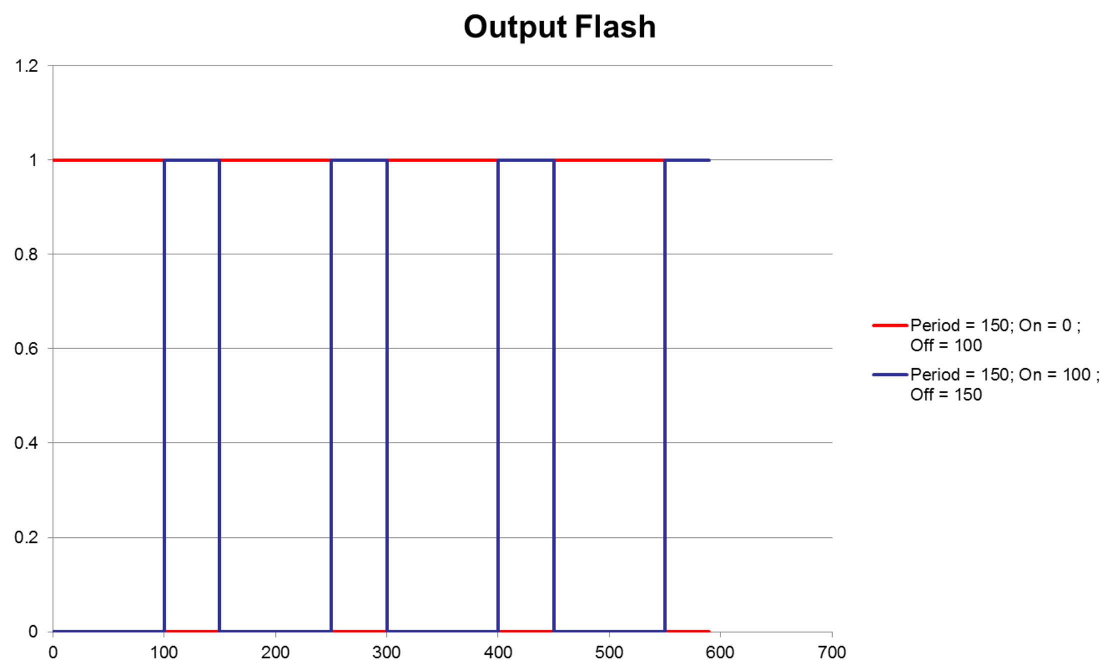
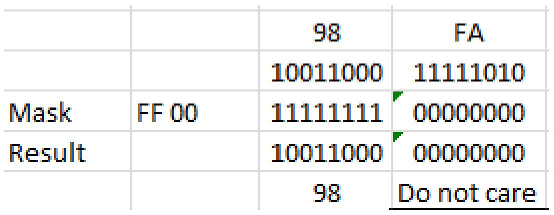


FIGURE 109: FLASH EXAMPLE

- *Flash Off Time (ms)* – The amount of time the image will not be visible when flashing.
- *Flash On Time (ms)* – This number determines how quickly the label will flash when in a flash state.
- *Flashing* – Determines whether or not the label is flashing.
- *Flyback A* – Available only for Single Wire Current Controlled Output. A calculated value, using the Flyback Calculation.xls worksheet, that factors into the approximation of the feedback current when using Single Wire with Flyback Approximation Enabled.
- *Flyback Approximation* – Available only for Single Wire Current Controlled Output. Enabling this allows the use of the Flyback A, B, and C properties to assist in Single Wire current control approximations. This property should be enabled if the Output has an inductive load.
- *Flyback B* – Available only for Single Wire Current Controlled Output. A calculated value, using the Flyback Calculation.xls worksheet, that factors into the approximation of the feedback current when using Single Wire with Flyback Approximation Enabled.

- *Flyback C* – Available only for Single Wire Current Controlled Output. A calculated value, using the Flyback Calculation.xls worksheet, that factors into the approximation of the feedback current when using Single Wire with Flyback Approximation Enabled.
- *Font* – Choose a font from the list of supported fonts.
- *Font Bold* – A true value bolds the font.
- *Font Color* – Choose the color of the font.
- *Font Italic* – A true value italicizes the font.
- *Font Size* – Adjust the size of the font.
- *Font Size Height* – The height of the font in pixels. Adjusted automatically based on the font size property.
- *Font Underline* – A true value underlines the font.
- *Frequency (Hz)* – This field is where to input the frequency desired for the Output between 40 and 5000Hz.
- *GridLineColor* – Choose what color to make the grid lines.
- *GridLineOrientation* – Choose what grid lines are visible; none, horizontal, vertical, both.
- *Groups* – Can assign a group that the Output is associated with to be used in Conductor for easy management and viewing of specific Outputs.
- *Height* – The vertical size, in pixels, of the widget.
- *HighlightColor* – Choose what color indicates that the row(s) and/or column(s) are highlighted.
- *HighlightedColumn* – Determine how many columns to highlight at a time.
- *HighlightedRow* – Determine how many rows to highlight at a time.
- *HighlightOrientation* – Choose the way to scroll through highlighted areas; moving horizontally, vertically, or cell by cell.
- *Horizontal Alignment* – The text can be left, center, or right justified.
- *Horizontal Margin* – Used as spacing between the right and left sides of the font on the respective widget.
- *Hue* – This value controls the video hue in a range of 0 to 255 where 255 is the highest level of hue.
- *I gain* – The integral gain associated with the PID loop
- *Identifier* – This is where the User defines the message ID to look for in order to receive the proper message. The ID can either be defined in Hex or Decimal depending on the display format.
- *ID Length* – Determines the length of the message ID; either 11-bit or 29-bit can be chosen.
- *Identifier Mask* – The User can define a mask to use for the message ID in order to filter for the correct message from the desired location. A mask is a Decimal or Hex, pending on Parsing Type, representation of a binary number, that when compared with the binary version of the message ID, allows specific bits to “fall through”. Those bits are the ones that form the valued portion of the message ID i.e. for a message ID of 98 A3 in Hex and the only portion of the ID that really matters is the 98 a mask of FF 00 should be used.



		98	FA
		10011000	11111010
Mask	FF 00	11111111	00000000
Result		10011000	00000000
		98	Do not care

The converted value from Hex to binary in the example is compared to the binary version of the ID and a bitwise AND operation is done. Anything compared with a 1 is what is desired and anything compared with a 0 is a do not care.

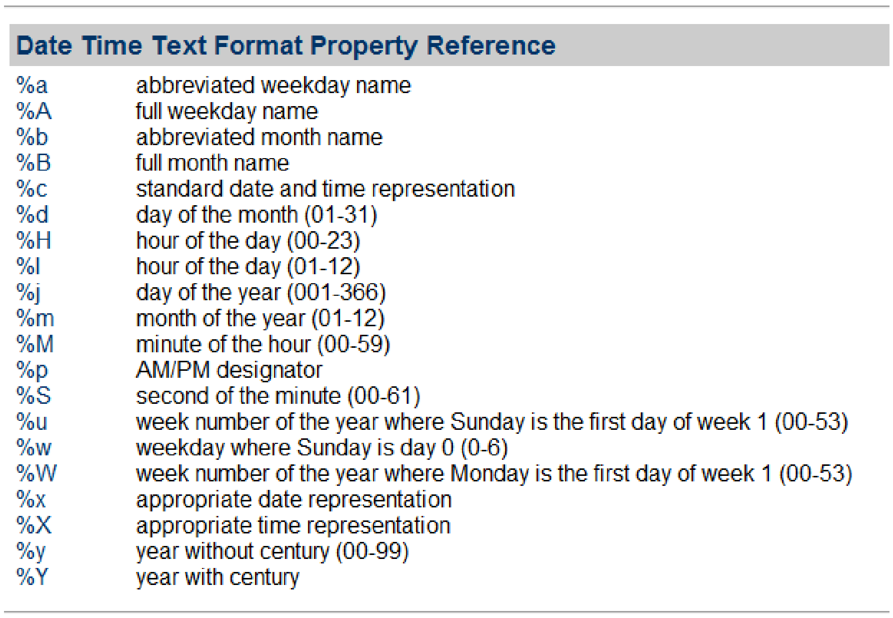
- *ID Value Adjustment* – Enables or disables the ability to choose an ID Length.
- *Image* – Path to and select the desired image to be displayed.
- *Input* – The input signal that will be used as a reference to run the PID loop
- *Input Center* – This is the value of the Input when in a rest position
- *Input Deadband* – A range above and below, depending on PID Operator Block, the Input Target that provides the system some hysteresis
- *Input Max* – This is the maximum value of the Input
- *Input Min* – This is the minimum value of the Input
- *Input Mode* – *Menu* that determines the type of Input.
- *Input Target* – The desired value that the PID operation drives to achieve
- *Is Touchable* – Sets whether or not this item can be interacted with via touchscreen.
- *IsHighlightEnabled* – Enable or disable if effect of highlighting.
- *K0 Gain* – Gain for the current control and can be derived from the equation: $(Propotional\ Gain + (Inegral\ Gain\ \times Time))$ where Time is the loop time.
- *K1 Gain* – This is the proportional gain for the current control.
- *Latching* – Setting this False will set the Input state for ON when the pin is active and OFF when it is inactive, similar to a momentary switch. If set True, when the pin changes from inactive to active, the Input will toggle between reporting ON and OFF and hold that value until the next transition from inactive to active.
- *Length* – Determines the size of the data being read in which is determined by the Parsing Type. For Bytes the Length can be 1 or 2 and for Bits 1 to 16.
- *Location* – Determines where the upper left corner of the widget resides on the screen where 0,0 (pixels) is the upper left most corner of the screen.
- *Mask* – This mask is for filtering specific data within the message to pass through.
- *Max* – This field is used to set what the max value of the Variable can be.
- *Max+* -- The amount allowed for the Input to go above the Input Max before it is considered and error
- *Max Input Frequency* – Set what the max expected frequency will be from 1 to 10000Hz.
- *Max Input Resistance (Ω)* – The Max Input Resistance can be changed with a range of 1Ω to 65535Ω.
- *Max Input Voltage (mV)* – The Max Input voltage can be changed with a range of 1mV to 65535mV, and is the maximum voltage expected to be seen on that pin. While the range can go to 65535mV, the pin may not handle that amount and the use of a data sheet to determine maximum input voltage for that pin is highly recommended.
- *Memo* – Space for an optional internal note for the User to use if desired.
- *Min* – This field is used to set what the min value of the Variable can be.

- *Min-* -- The amount allowed for the Input to go below the Input Min before it is considered and error
- *Min Transmit Period* – This is the amount of time, where 100 is equal to 1 second, that must elapse before the message is transmitted from an I/O module to the Master Module. This property can be used to slow down the message some by only receiving it on given intervals.
- *Mission Critical Settings* – Here is where the Mission Critical Settings for what the Output will do when in a Mission Critical condition.
    a. Turn Off – The Output is turned off when in a Mission Critical condition.
    b. Turn On – The Output is turned on when in a Mission Critical condition.
    c. Turn On and Flash – The Output is turned on and Flash setting is enabled when in a Mission Critical condition.
    d. Maintain Current State – Commands the Output to maintain whatever state it was in when the Mission Critical condition was detected.
- *Module* – This is the module that will receive the CAN message.
- *Modes* – Select what will be edited; a single cell, entire row, or entire column.
- *Multi-Line* – This option if true, allows the label to have multiple lines and will wrap the text to fit within the defined size of the label.
- *Name* – Space to enter a unique name for the Input/Output/Display/Property.
- *Number of Elements* – Defines the number of elements within the array.
- *Off Time* – This determines the length of time, in milliseconds, for the Output to be off, within the Period, when in a Flash condition. This value must be less than the Period value.
- *Offset* – Optional setting used for adjusting the incoming value to the desired units for the telematics service, or linking to a display widget.
- *On Time* – This determines the length of time, in milliseconds, for the Output to be on, within the Period, when in a Flash condition. This value must be less than the Period value.
- *Orientation* – This determines how the bar is placed on the screen, in either a vertical or horizontal fashion.
- *Outline Color* – Choose the color of the needle outline.
- *Outline Path Color* – Choose the color that the outline path will have.
- *Outline Path Radius* – This is the radius that the outline will follow.
- *Output Max* – This is the max value that will be placed within the A term to drive an output based on the result of the PID calculation
- *Output Max Current (mA)* – Define the maximum current for the Output to command, in milliamps. Please make note that not all Outputs on all modules have the same current limitations, please refer to the specific module data sheet to determine what the Output can allow.
- *Output Mode* – This field is where the mode of Output is chosen; Digital, PWM, Constant Current, or Frequency.
- *Output Scaling* – This is a scaling factor for the output value where 1000 is equal to a factor of 1
- *Output Threshold* – This is the minimum value that will be placed within the A term to drive an output based on the result of the PID calculation
- *Output Type* – This field sets whether the Output will be Sourcing, Sinking, or a Servo.
    a. Sourcing – The Output is sourcing the current to the device, connecting the pin to "Battery" when On.

   b. Sinking – The Output is sinking the current from the device, connecting the pin to "Ground" when On.
   c. Servo – This mode allows the Output to both sink and source the current to the device allowing it to be connected to either "Ground" or "Battery" when On and Open when Off.

- *Overlay Image* – Path to and select an image to overlay the needle if desired.
- *Overlay Location* – This is the location the upper left corner of the overlay image will appear where 0,0 is the upper left most pixel of the needle widget.
- *Overlay X* – The horizontal location of the overlay image, in pixels, on the needle widget.
- *Overlay Y* – The vertical location of the overlay image, in pixels, on the needle widget.
- *P gain* – The proportional gain associated with the PID loop
- *Period* – This value is what determines how long the period of the Flash, typically found by adding the Off Time and On Times together.
- *Pointer Color* – This adjusts the pointer color.
- *Pointer Length* – The length of the pointer part of the needle, in pixels.
- *Pointer Style* – Able to choose between a triangle or block style. The triangle will come to a point at the end while the block will have a uniform thickness throughout the entire length.
- *Pointer Width* – Determines how thick the pointer is. For the triangle style the width is what the base starts at before tapering to the point.
- *Radius* – This value will determine the size of the progress bar, in pixels. Increasing this number will increase the size of the circle that the progress bar will fill.
- *Read Security Level* – Determines the level dongle needed to see the value of the Input/Output within Conductor. This can be Level 1, 2, or 3 with 1 needing the highest security clearance.
- *Report Rate (msec)* – The Report Rate controls how often, over CAN, the I/O Module will report the value on the pin, and can be changed from 10ms to 2550ms in increments of 10ms. Note that if a Master Module uses a VTD Input the Report Rate will not be taken into account and the value will be updated every loop.
- *Resolution* – Optional setting used for adjusting the incoming value to the desired units for the telematics service, or linking to a display widget.
- *Rest Point (degrees)* – Dictates the angle at which the needle widget begins.
- *Rotation Direction* – Controls the direction in which the needle will travel from its rest point.
- *RowsCount* – Defines the number of rows the table will have.
- *"Screen" Resolution* – Changing the Display Model will adjust the automatically.
- *Safe Mode Settings* – Here is where the Safe Mode Settings for what the Output will do when in a Safe Mode condition.
   a. Turn Off – The Output is turned off when in a Safe Mode condition.
   b. Turn On – The Output is turned on when in a Safe Mode condition.
   c. Turn On and Flash – The Output is turned on and Flash setting is enabled when in a Safe Mode condition.
   d. Maintain Current State – Commands the Output to maintain whatever state it was in when the Safe Mode condition was detected.
- *Save On Shutdown?* – The default is No, changing this to Yes will save the value stored within the Time Counter when the system is shut down in order to resume at the same point when the system is restarted.

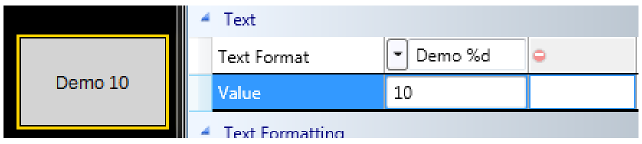
- *Selected Properties* – This allows the User to select a specific cell, or edit the height or width of the selected row or column respectively.
- *Set Point (1-80000 mA)* – When the Diagnostic Requirement check box for Over Current is checked this option appears. This option allows the User to determine the current setting that the Output must exceed before it detects an Over Current event.
- *Show Outline* – Makes the outline visible or not.
- *Show Outline Path* – This property controls whether or not the outline path will be visible.
- *Show Shadow* – Choose whether or not to show a shadow effect of the needle. Please note that depending on the background color the shadow may not show or show well.
- *Size* –Determines how large the widget/display will appear to be, comprised of the height and width 0,0 (pixels) respectively.
- *Slew Off* – This is the amount of time it takes for the Output to go from 100% to 0, in milliseconds with a max time of 1000mS.
- *Slew On* – This is the amount of time it takes for the Output to go from 0 to 100%, in milliseconds with a max time of 1000mS.
- *Source Type* – Choose whether the signal is a sourcing or sinking Input.
- *Start Byte* – Defines the most significant byte, i.e. if LSB->MSB and start byte is 3 it would read in bytes 3 and 2 while MSB->LSB with start byte 3 would read in bytes 3 and 4.
- *Start Point (degrees)* – Determines the angle at which the needle widget starts its travel from as an offset from the Rest Point. May not always equal the rest point.
- *Start Point (Pixel)* – This value will determine how far from the edge the bar starts at.
- *Start Value* – This is the value of the widget when at the start point.
- *State Enumerations* – Clicking on this property brings up a mini menu to add states. Within that mini menu is where the User can also edit the names and numbers of those states that have been added.
- *Table Area* – Rows and columns of the created table.
- *Tail Color* – This adjusts the tail color.
- *Tail Length* – The length of the tail part of the needle, in pixels.
- *Tail Style* – Able to choose between a triangle or block style. The triangle will come to a point at the end while the block will have a uniform thickness throughout the entire length.
- *Tail Width* – Determines how thick the tail is. For the triangle style the width is what the base starts at before tapering to the point.
- *Text Location* – Determines the location of the text within the widget where 0,0 are the X and Y coordinates, in pixels.
- *Text Visible* – This options controls whether or not the text will be seen.
- *Text X* – The horizontal location within the widget that the text will begin, in pixels.
- *Text Y* – The vertical location within the widget that the text will begin, in pixels.
- *Text Format* – This is the information the label will display from either a manual entry or a string list.



Date Time Text Format Property Reference

%a	abbreviated weekday name
%A	full weekday name
%b	abbreviated month name
%B	full month name
%c	standard date and time representation
%d	day of the month (01-31)
%H	hour of the day (00-23)
%I	hour of the day (01-12)
%j	day of the year (001-366)
%m	month of the year (01-12)
%M	minute of the hour (00-59)
%p	AM/PM designator
%S	second of the minute (00-61)
%u	week number of the year where Sunday is the first day of week 1 (00-53)
%w	weekday where Sunday is day 0 (0-6)
%W	week number of the year where Monday is the first day of week 1 (00-53)
%x	appropriate date representation
%X	appropriate time representation
%y	year without century (00-99)
%Y	year with century

FIGURE 110 : SUPPORTED TEXT FORMATTING



Text Format	Demo %d
Value	10

FIGURE 111: LABEL TEXT FORMATTING EXAMPLE

- *Time Interval* – This determines when the Counter will either increment or decrement.
  a. Loop Time – Runs as quickly as the application is executed, ~10mS.
  b. 1 Second – Will run once a second.
  c. 10 Seconds – Executes once every 10 seconds.
  d. 1 Minute – Executes once every 1 minute.
  e. 10 Minutes – Executes once every 10 minutes.
- *Touch Size* – This determines the size of the area that will register a touch for the item when using a touchscreen.
- *Translation* – This option determines what language the text will be displayed in. Leaving it on the “follow display” option allows the language to be changed dynamically by only adjusting the language of the display, otherwise the translation will remain static to what is chosen with this property.
- *Tx Rate* – The rate in which a message is transmitted based on the 10ms loop time.
- *Tx Status* – Determines when a message can be transmitted.
  a. Disabled – This message will not be transmitted, can act as a stop.
  b. Pass Through – The message will be passed through as it is received.
  c. On Report Rate – Message will be transmitted at the interval determined by the Tx Rate property.
- *Type* – This property sets what the size of the Variable can be either a 16-bit unsigned or 32-bit unsigned (65,535 or 4,294,967,295). There is an option for Alarm as well which is

used in a case of the Variable being an alarm to send notice to the network through the telematics Module.

- *Units* – This is an option field to associate a unit description with the Variable to be viewable within Conductor.
- *UpperLeftColumn* – Dictates what column will be the starting column to be used or viewed within the table.
- *UpperLeftRow* – Dictates what row will be the starting row to be used or viewed within the table.
- *Value* – Input variable used within the widget/constant.
- *Vertical Alignment* – The text can be vertically top, center, or bottom justified.
- *Vertical Margin* – Used as spacing between the top and bottom sides of the font on the respective widget.
- *Visible* – Determines if and when the label is visible. Linking this property to a data item can determine when the label becomes visible on the screen based on the data item returning a true of false value.
- *Widget Properties* – Here the User can select what the cell, row, or column will have either an image or label. After selecting image or label, those properties will appear within that window to be edited.
- *Width* – The horizontal size, in pixels, of the widget.
- *Wire Number* – Space for an internal note to document the Input's or Output's wire or harness number if desired.
- *Write Security Level* – Determines the level dongle needed to edit the value of the Input/Output within Conductor. This can be Level 1, 2, or 3 with 1 needing the highest security clearance.
- *X* – The horizontal position, in pixels, on the screen where the label resides.
- *Y* – The vertical position, in pixels, on the screen where the label resides.
- *Z Order* – The lower this number is the lower it will be in the "layer" meaning it will be drawn before higher numbers. The higher numbers will lay over the lower "layers", so the User must be careful not to cover up something that they wish to be visible with a higher z ordered widget.

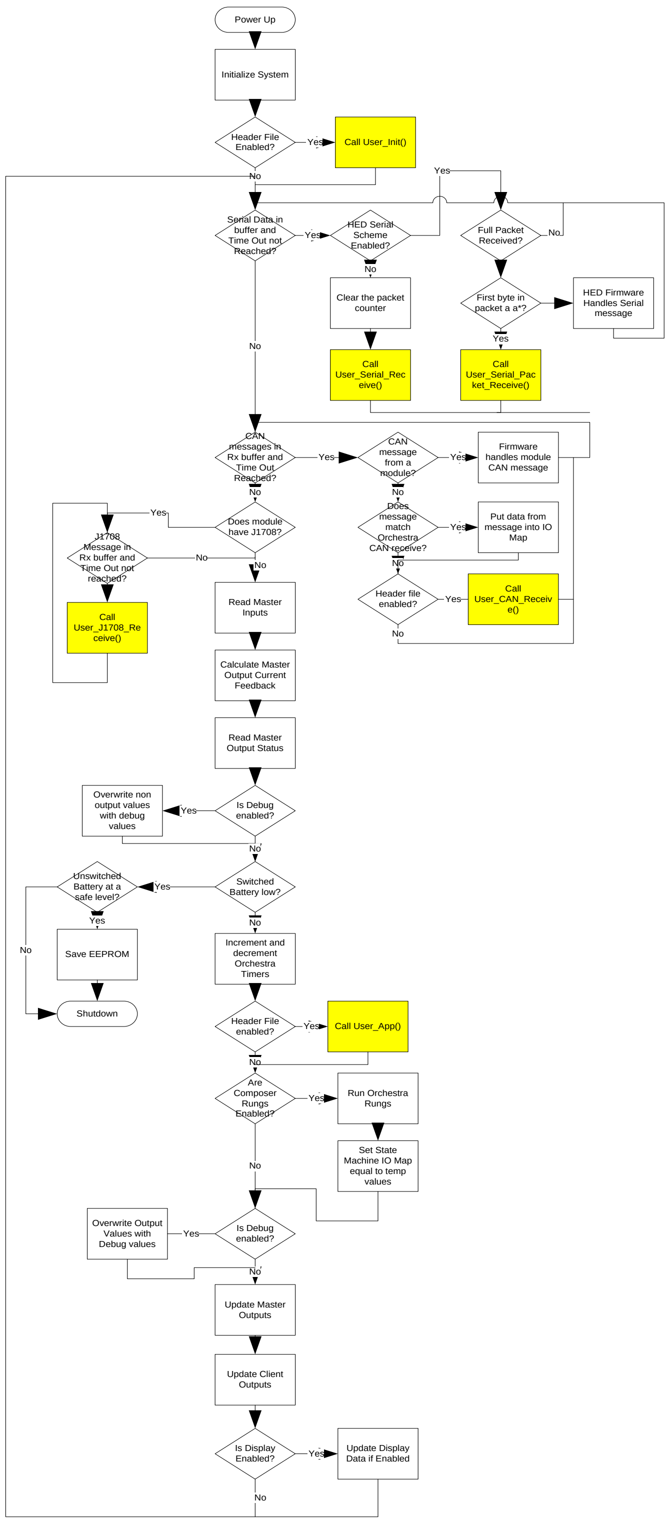

Power Up
Initialize System
Header File Enabled?
Yes
Call User_Init()
No
Yes
Serial Data in buffer and Time Out not Reached?
Yes
HED Serial Scheme Enabled?
No
Clear the packet counter
Call User_Serial_Receive()
Full Packet Received?
No
First byte in packet a a*?
HED Firmware Handles Serial message
Yes
Call User_Serial_Packet_Receive()
No
CAN messages in Rx buffer and Time Out Reached?
Yes
CAN message from a module?
Yes
Firmware handles module CAN message
No
Does message match Orchestra CAN receive?
Yes
Put data from message into IO Map
No
Header file enabled?
Yes
Call User_CAN_Receive()
No
No
Does module have J1708?
Yes
J1708 Message in Rx buffer and Time Out not reached?
No
Call User_J1708_Receive()
No
Read Master Inputs
Calculate Master Output Current Feedback
Read Master Output Status
Is Debug enabled?
Yes
Overwrite non output values with debug values
No
Switched Battery low?
Yes
Unswitched Battery at a safe level?
Yes
Save EEPROM
No
Shutdown
No
Increment and decrement Orchestra Timers
Header File enabled?
Yes
Call User_App()
No
Are Composer Rungs Enabled?
Yes
Run Orchestra Rungs
Set State Machine IO Map equal to temp values
No
Is Debug enabled?
Yes
Overwrite Output Values with Debug values
No
Update Master Outputs
Update Client Outputs
Is Display Enabled?
Yes
Update Display Data if Enabled
No

FIGURE 112: PRESTO FLOW CHART

Revision History			
Rev No.	**Description**	**Date**	**User**
00.01	Initial rough draft release	3/31/2014	J. Kothrade