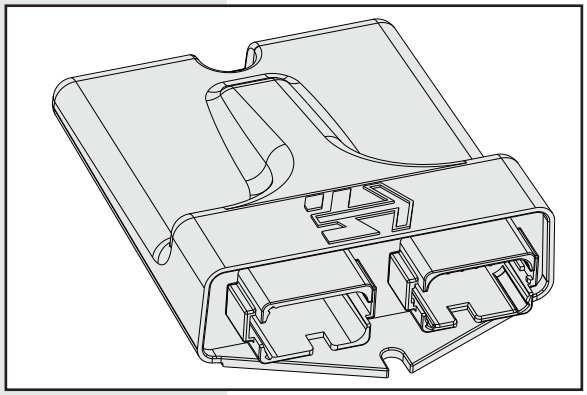
User Manual

ECDR-0506A Electronic Configurable Valve Driver



HydraForce

7709069 • UM_ECDR-0506A • 05/16 •

CONTENTS

INTRODUCTION

Overview

This instruction details the steps required to connect and configure the ECDR-0506A configurable driver using the HydraForce communication adapter and a laptop computer. For help or additional information, email HydraForce at electronic.support@hydraforce.com

Description

The ECDR-0506A is a microprocessor-based valve driver designed for use in hydraulic proportional valve applications. It is configurable to drive coils with an input from voltage, current, resistance, or frequency signal. You can apply complex logic programming with an easy to learn and use diagrammatic programming interface. HF-Impulse, a web accessible configuration tool, is available as a free download at www.hydraforce.com/electronics.

Operation

The driver accepts inputs from commonly available analog or digital operator interface devices (joystick, potentiometer, sensors, master controller, etc.) The diagram interface allows you to configure the unit for complex control of hydraulic systems.

Error Detection Features

This driver has built-in error detection for defined conditions. The driver turns off affected outputs when it detects a fault. When the fault is corrected and the power is cycled, the driver returns to normal operation. The ECDR-0506A monitors the following conditions for error:

- Supply voltage below 8.5 volts.
- Supply voltage above 33 volts.
- Coil resistance below acceptable range, 2 ohms.
- Coil resistance above acceptable range, 100 ohms.

References

Refer to *ECDR-0506A Technical Reference* for information on hardware design and function.

Refer to CAN and CANopen documentation from *CAN in Automation* (CiA) for further information on communication standards.

Tools/Materials You Need

You Supply:

- ❑ Computer with Windows 7 or newer
- ❑ Power supply, 9–32 volts dc

HydraForce Supplies:

Software (download from www.hydraforce.com/electronics)

- ❑ HF-Impulse version 1.7.0 or newer
- ❑ Drivers for CAN/USB converter
- ❑ ECDR-0506A firmware

Hardware

- ❑ CAN/USB converter: 4000371
- ❑ Test harness: 4000307 (connector X1), 4000308 (connector X2)

Documents

- ❑ ECDR-0506A User Manual (this document)
- ❑ ECDR-0506A Technical Reference

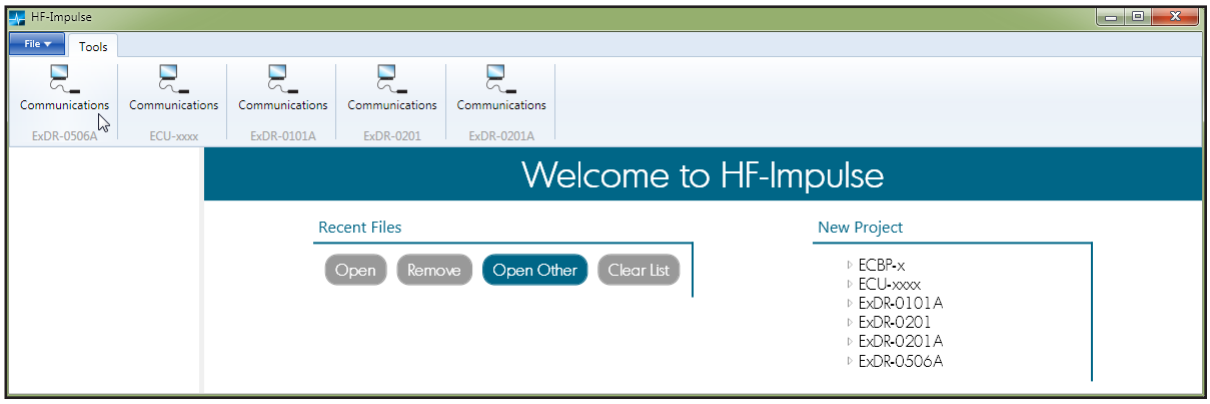
WELCOME SCREEN

Overview

HF-Impulse is the software you use for communicating with and configuring the ECDR-0506A, as well as other HydraForce products. If you have not already, you may download this software from the HydraForce Electronics Portal at www.hydraforce.com/electronics. After downloading, unzip the file and run **`HfImpulse.Installer.msi`** to install.

Note: *www.hydraforce.com/electronics is a secure portal. Register if you do not already have a user ID and password. Registration is handled automatically if you use a company e-mail address.*

When you open HF-Impulse, you first see the *Welcome* screen. This screen allows you to choose the product with which you want to interact.



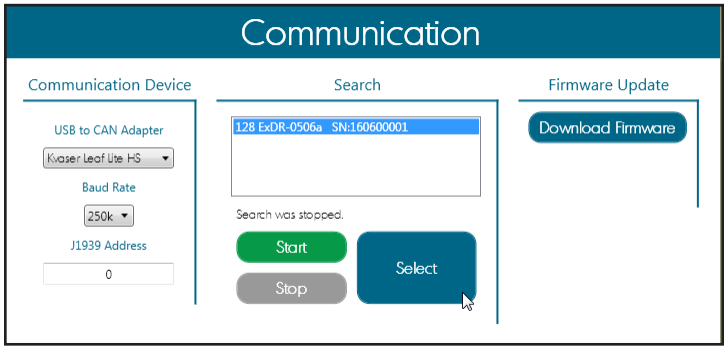
Connecting

Note: *Connecting to the ECDR-0506A requires a wiring harness (HF 4000307, 4000308). The CAN network must have at least one 120 ohm terminating resistor across the CANH and CANL lines. This terminator is included in the 4000307 harness. For more information on connecting, including building a proper test harness, refer to the ECDR-0506A Technical Reference.*

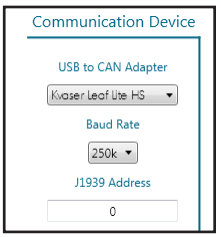
1. Connect the harness to the ECDR-0506A.
2. Connect the harness to 9–32 Vdc power supply.
3. Connect the harness to the CAN/USB converter (Kvaser or PEAK).
4. Connect the converter to the appropriate USB port on your computer.
5. Make sure the correct drivers are installed for the CAN/USB converter.

Establishing Communications

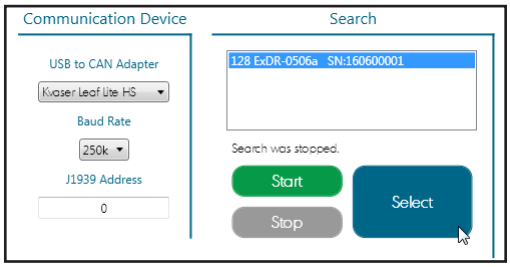
6. From the *Welcome* screen tools bar, select **`Communications`** for the ExDR-0506A.



7. From the *Communication* screen select the appropriate *USB to CAN Adapter* under *Communication Device*. You may also change the *Baud Rate* or enter an alternative *ExDR-0506A J1939* address. The default is 128.



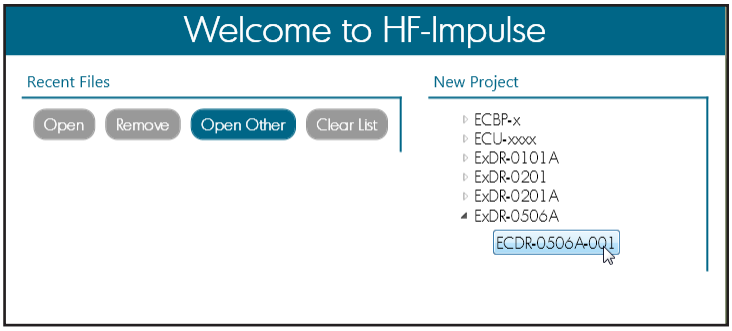
8. Click the **`Start`** button to begin searching for the device. You may click **`Stop`** to cancel the search. When the search is complete, select the device from the choices and click the **`Select`** button. HF-Impulse should find the device at address 128. If there is a communication error, check your electrical connections or check your CAN/USB device drivers.



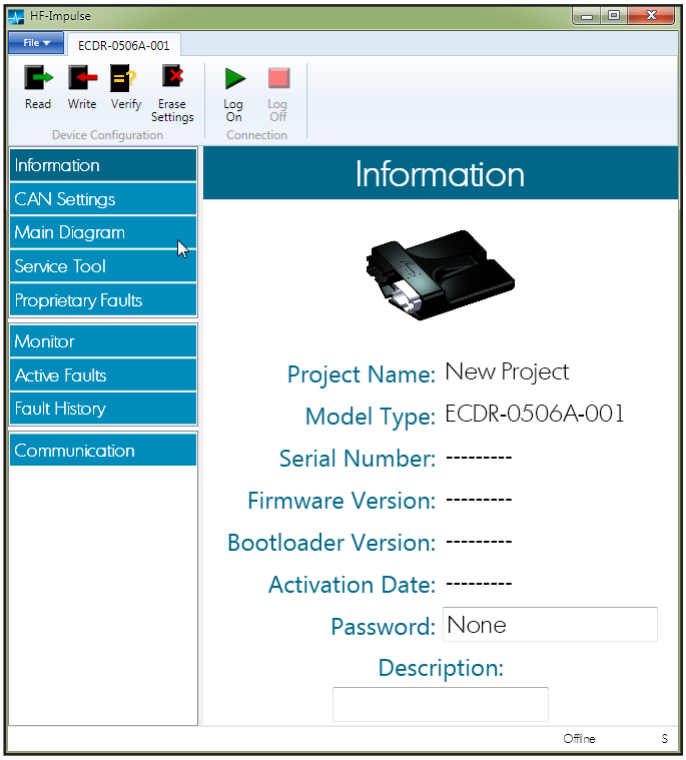
Creating A New Project

1. From the *Welcome Screen* in HF-Impulse, there is a column to the right for *New Projects*. Select the **`ECDR-0506A`** from this group. You may also choose to open an existing project from the **`File|Open`** menu or making a choice from the *Recent Files* area.

   Alternatively, from the *File Menu* you can select **`New|ExDR-0506A|ECDR-0506A-001`** to begin a new project.



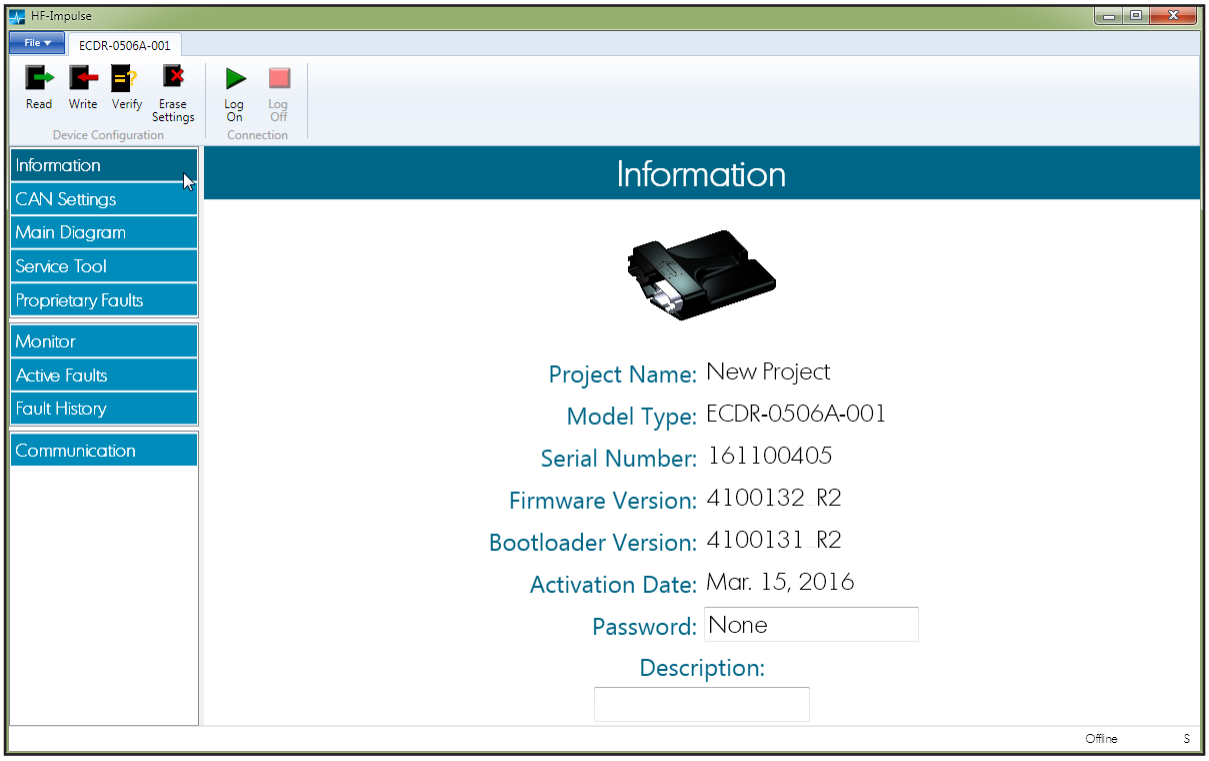
2. HF-Impulse creates a new project and displays the information page. From this screen click **`Main Diagram`** to create the configuration. Refer to *Main Diagram Screen* on page 9 for further instructions.



INFORMATION SCREEN

Overview

The *Information* screen displays details about the connected ECDR-0506A. These details are blank when starting a new project that you have not yet written onto a device.



Details

Project Name

This is the name assigned to the project when it was saved. The default name is "New Project".

Model Type

This is the model type. The ECDR-0506A-001 is an Electronic Configurable valve DRiver.

Serial Number

This is is the serial number of the connected device.

Firmware Version

This is the number and revision of the current firmware version. It consists of a 6 digit number that identifies the model type, then the version preceded by an underscore.

4100132_R**

The _R** represents the version number. Prototype versions are letters _RA through _RZ. Production released versions are numbers _R1 through _R99. You can update the firmware using the *Firmware Update* utility on the *Communication* screen (see *Firmware Update* on page 54).

Bootloader Version

This is the number and revision of the current bootloader version. It consists of a 6 digit number that identifies the model type, then the version preceded by an underscore. The bootloader is installed at the time of manufacture and cannot be updated.

4100132_R**

The _R** represents the version number. Prototype versions are letters _RA through _RZ. Production released versions are numbers _R1 through _R99.

Activation Date

This is the date the device was first configured by the user. A new ECDR-0506A is shipped with a null value. Once a controller is first configured, the *Activation Date* never changes.

Description

The *Description* is saved with the *.icf configuration or service tool file but is not saved in the device itself.

Password

You can configure your ECDR-0506A with a *Password* to prevent unauthorized access to the configuration. This locks access to the configuration file (*.icf) and disables the ability to read from the ECDR-0506A.

Setting a Password

1. On the Information screen, type a *Password* into the field.
2. Click **`Write`** to write the configuration to the ECDR-0506A.
3. Save the configuration file (*.icf) when prompted.
4. A success message is displayed after the write operation is complete.

Note: *Once the password is set you must to save the configuration file (*.icf) before a write is permitted. To further edit the configuration in the ECDR-0506A you must have the *.icf file and password to continue.*

Changing or Removing a Password

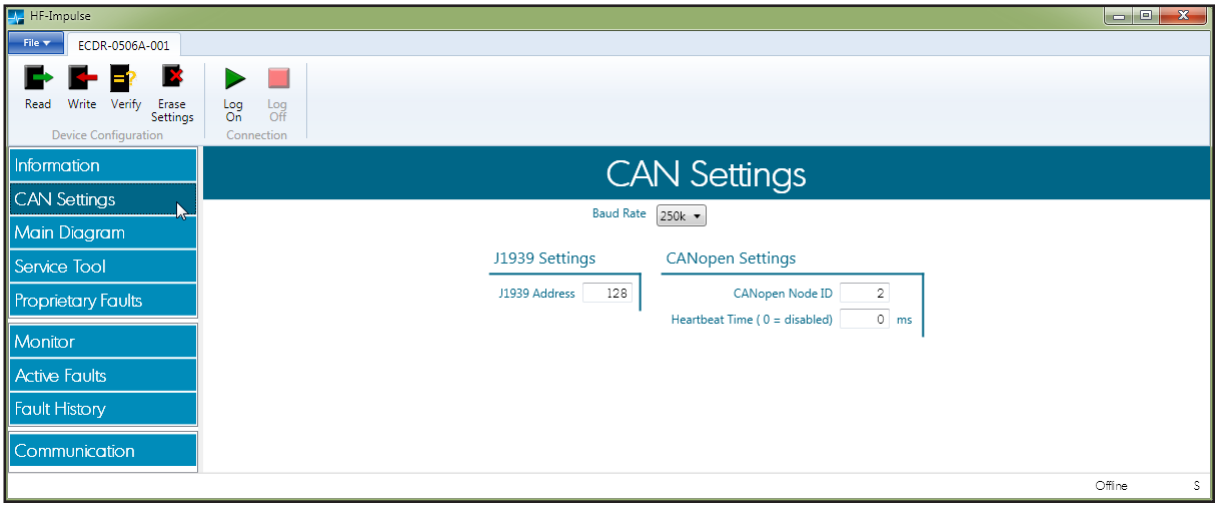
1. First you must load the configuration file (*.icf) into HF-Impulse. You cannot change or reset a password without this.
2. Type the new password into the *Password* field on the *Information* screen. Type the word *None* to remove password protection.
3. Click **`Write`** to write the configuration to the ECDR-0506A.

Note: *The password is masked on the Information screen. You may want to record the password in a secure location. If either configuration file or password is lost, then you must* **`erase`** *the ECDR-0506A or* **`write`** *a new configuration to affect changes.*

CAN SETTINGS SCREEN

Overview

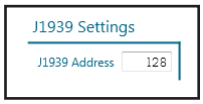
On the *CAN Settings* screen you can set parameters for communicating on the CANopen or SAE J1939 CAN network. Here you can change the J1939 Address or CANopen Node ID. If you alter the addressing or *Baud Rate* and *Write* to the ECDR-0506A, you can use the *Communication* screen to re-establish your connection with the device (see *Communication Screen* on page 53).



Baud Rate

- The *Baud Rate* represents the speed of data transfer on the CAN bus.
- Choices are 250 kbps or 500 kbps.

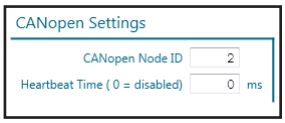
J1939 Settings



J1939 Address

- The address on the SAE J1939 CAN network for the ECDR-0506A
- Value: 1 – 247

CAN Open Settings



CANopen Node ID

- The address on the CANopen network for the ECDR-0506A
- Value: 0 – 127

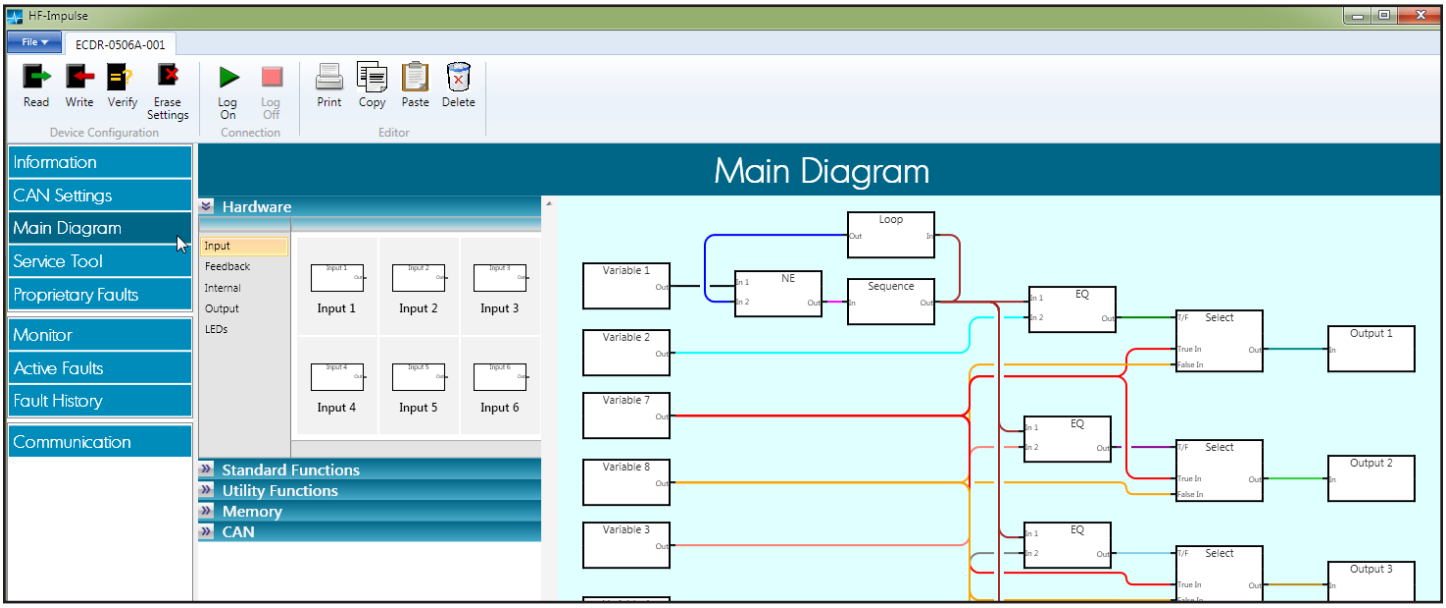
Heartbeat Time

- The interval for the heartbeat message on the CANopen network in milliseconds
- The ECDR-0506A transmits a heartbeat to the bus at this interval. If the setting is zero, it does not transmit a heartbeat.
- Value: 0 – 10 000

MAIN DIAGRAM SCREEN

Overview

The diagram editor allows you to make complex logic for controlling the ECDR-0506A. You can drag control blocks from the *Gallery* and add them to the diagram. You connect the blocks to complete a path from input to output. Each block has internal settings you can adjust. The settings menu also has a text field which allows you to name the blocks.

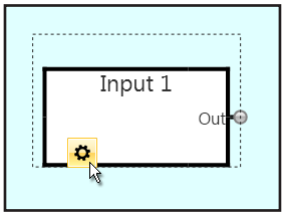


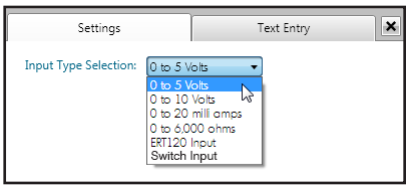
Function Blocks

You drag blocks from the *Gallery* onto the diagram to add them to your project. Deleted blocks return to the gallery and you can reuse them.

Block Settings

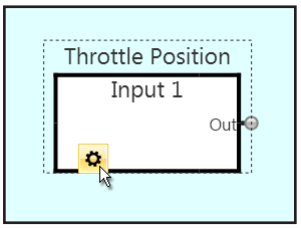
Each block type has a unique settings menu. The settings icon appears when you click the block. Click this icon to view the settings. Each menu has two tabs, *Settings* and *Text Entry*. The *Settings* tab has entries specific to each block type. This document details these entries in subsequent sections.





Block Commenting

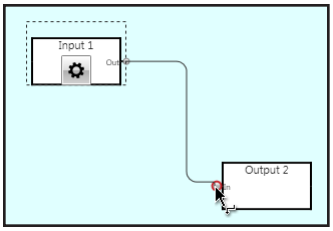
The *Text Entry* tab allows you to add comments to the block. This text appears above the block in the diagram. This can be a useful reminder for the intent of the block.



Block Connections

To connect blocks in the diagram:

1. Click a block. Its connection points appear as a small gray circles.
2. Click and drag a connection point to the next block. The software draws a connector. As the pointer approaches another block, the connection points appear.
3. Release the mouse button to connect.



Connection Restrictions

There are several restrictions when connecting the blocks.

- A single block's input connections cannot be connected to its own output.
- A block's output cannot be connected to another block's output.
- A block's input cannot be connected to another blocks input.

Connections flow from left to right. There is validation to prevent connecting out of sequence. If you wish to send data back to the left of the diagram, use the *Loop* block (see *Loop Blocks* on page 26).

Connection Values

Each connection represents the passing of a value from one block to the other. The values are all unsigned integers of 0 to 65535. Some blocks have binary logical states. The logical states are as follows:

- A zero value represents logical false.
- A non-zero value represents logical true.

Unused Connections

Unused input connections default to zero. The noted exception to this rule is the *EN* input on ramp blocks. An unconnected enable input on a ramp block defaults to 1. Good programming practice demands you initialize all variables, therefore you should make all connections on each block. If zero is desired, connect a variable block.

Editing

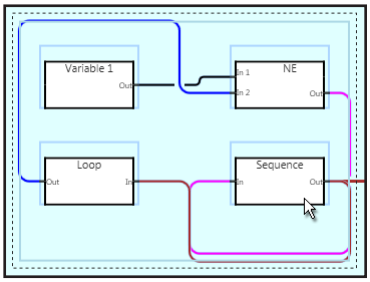
Selecting and Moving

You may click blocks and connectors on the diagram to select them. Hold the **CTRL** key to select multiples, or drag a marquis enclosing/touching all the items you wish to select.

You can move your selected items around the diagram. Click and drag to relocate. The connectors stay attached and redraw automatically if you move one end of the connection. When saving, the location of the blocks is stored in the *.icf file.

Note: *To conserve memory, locations of the diagram items are not stored on the device itself. When performing a Read operation from the device, HF-Impulse interprets the program and draws the Main Diagram according to its ruleset. Comment blocks are also not saved on the device.*

Grouping/Ungrouping



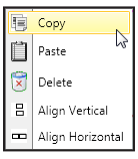
With multiple items selected, press the shortcut key sequence **CTRL+G** to group those items. Moving any item in a group moves all of the items together. While a group is selected, Press **CTRL+U** to ungroup the items.

Zooming/Panning the Viewport

You may hold the **CTRL** key and rotate the mouse wheel to zoom the viewport. When items are outside the viewport the diagram scroll bars appear. You may also use the bars to scroll the view.

You may hold the **CTRL** key while clicking and dragging the mouse to pan the viewport. When items are outside the viewport the diagram scroll bars appear.

Copy/Paste, Delete, Align



To copy diagram items, use one of these methods:

- Click the **Copy** button on the ribbon.
- Right click your selection and choose **Copy** from the context menu.
- Use the shortcut key **CTRL+C**.

Note: *These items cannot be copied/pasted: hardware group blocks (input, output, feedback, internal, and LED blocks), macro blocks, and comment blocks.*

To paste items (duplicating them) do this:

- Click the **Paste** button on the ribbon.
- Right click the diagram and choose **Paste** from the context menu.
- Use the shortcut key **CTRL+V**.

To delete items from the diagram:

- Press the **delete** key on the keyboard.
- Click the **Delete** button on the ribbon.
- Right click your selection and choose **Delete** from the context menu.

Note: *Be sure of your selection and action before pressing the* **delete** *key. Deletions are permanent. There is no undo function.*

To align multiple items:

- Select multiple blocks.
- Right click the selection and choose **Align Vertical** or **Align Horizontal** from the context menu.

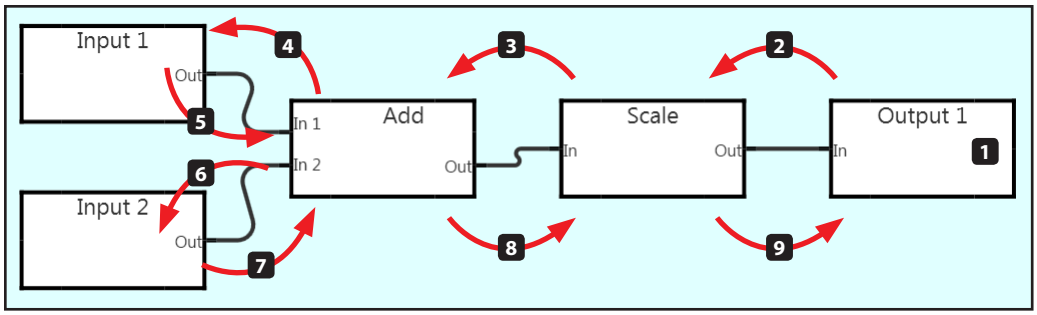
Shortcut Keys

Here is a list of the available shortcut keys you can use while editing the diagram:

- **CTRL+A:** Select All
- **CTRL+C**: Copy
- **CTRL+V**: Paste
- **CTRL+D**: Duplicate (copy and paste in a single operation)
- **CTRL+G**: Group
- **CTRL+U**: Ungroup
- **CTRL+ARROW**: Nudge (move the selection a small amount in any direction)
- **CTRL+SHFT-ARROW**: Super Nudge (move the selection a large amount in any direction)
- **CTRL+Mouse drag**: Pan the view
- **CTRL+Mouse wheel**: Zoom in/out

Diagram Processing

The ECDR-0506A processes the diagram following a specific object order. It resolves the diagram recursively from right to left: starting with output values and working back through the inputs. Each complete evaluation of all the blocks in the diagram is a scan. Some blocks, such as the loop and write blocks, lag by one scan. That is to say that the blocks they affect will not change status until the following scan.



Example

1. The ECDR-0506A scans the diagram and finds the first output (*Output 1*).
2. *Output 1* looks for an input from the *Scale* block.
3. The *Scale* block looks for an input from the *Add* block.
4. The *Add* block looks for its first operand from *Input 1*.
5. *Input 1* has a value, it is the value of the connected input.
6. The *Add* block then looks for its second operand from *Input 2*.
7. The *Add* block, having satisfied its inputs, performs the addition operation and sets its output.
8. The *Scale* block now has an input and performs the scale operation and sets its output.
9. The *Output* block now has an input and sets the ECDR-0506A output to the this value.
10. The ECDR then looks for the next block according to the processing order.
11. Processing repeats recursively until the entire diagram is processed.

Processing Order

Blocks are evaluated in this order:

- Output
- LED
- Monitor
- CAN Tx
- Loop
- Write
- Fault

Device Configuration

After configuring the diagram, you can transfer the parameters to the ECDR-0506A. Controls for transferring parameters include *Read*, *Write*, and *Erase*. These features function when communication is established with the ECDR-0506A. See *Connecting on page 4* for details on connecting to the driver.

Read

The *Read* function allows you to read the configuration from the ECDR-0506A. The diagram is displayed on the canvas. Reading does not retain any of the block position data or block text. HF-Impulse systematically places the blocks in a set pattern. The block position and block text are saved in a configuration (*.icf) file. Therefore it can be better to open a saved configuration than to read from the ECDR-0506A. Also, if you set a password and write it to the unit, the read function becomes disabled. To edit you need the saved configuration file (*.icf).

Write

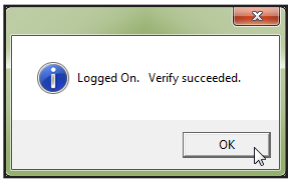
The *Write* function transfers all parameters and diagram logic to the ECDR-0506A. This transfer can take several seconds to perform.

Erase

The *Erase Settings* function sets all the ECDR-0506A parameters and diagram logic, including *Retain* block values and *Fault History*, back to initial values. This operation is not reversible, however if you saved the configuration (*.icf) file you can restore from this file.

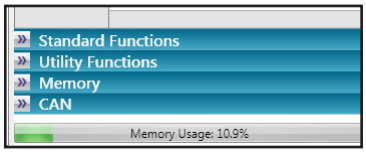
Verify

The *Verify* function compares the current HF-Impulse configuration with the loaded configuration on the connected device. A message displays the result of the comparison.



Memory Usage

The diagram editor includes a memory usage indicator. It is located below the block gallery. It gives an estimated percent of memory used and a progress bar provides visual indication. If memory usage reaches 100%, adding additional blocks results in an error message.

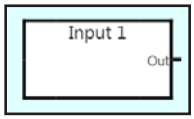


HARDWARE BLOCKS

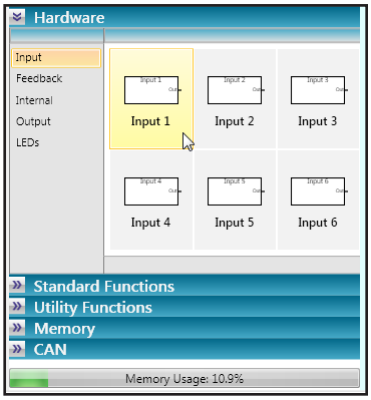
Hardware blocks relate to the physical hardware on the ECDR-0506A. They carry the status of inputs, feedbacks, and other internal parts of the circuit. They drive the outputs and LEDs. Blocks are grouped into these categories:

- Input
- Feedback
- Internal
- Output
- LEDs

Input Blocks



There are 6 *Input* blocks. Each block represents an external input to the ECDR-0506A. The inputs have two types, analog and digital. The input blocks only have an out connection.

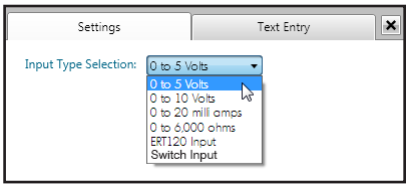


External Inputs

These blocks correspond to the external inputs of the ECDR-0506A.

Input	Connector X1 (gray)	Connector X2 (black)	Input type
1	X1.10		Analog
2	X1.11		Analog
3		X2.8	Analog
4		X2.10	Analog
5		X2.6	Digital
6		X2.7	Digital
GND	X1.2	X2.9, X2.11	Signal ground

Settings

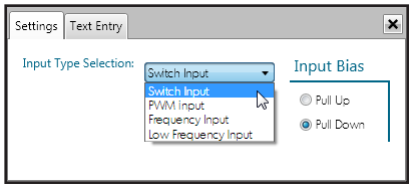


Analog Input Type
Inputs 1 through 4 have the following input types.

- Voltage measurement of 0 to 5 volts
- Voltage measurement of 0 to 10 volts
- Resistance measurement of 0 to 6000 ohms
- Amperage measurement of 0 to 20 milliamps
- Temperature sensor ERT120 manufactured by HydraForce
- Switch input—specify input bias:

  - *Pull Up* – An internal resistor pulls the input pin voltage up to approximately ½ the supply v oltage. This is intended for sinking devices or switches wired to ground.
  - *Pull Down* – An internal resistor pulls the input pin voltage down to ground. This is intended for sourcing devices or switches wired to supply positive.
  - *Pull Center* – This bias choice is only available with switch type analog inputs. It provides three-state switch selection. Input is pulled to 3.3 volts. This is intended for use with double throw switches that switch to ground or supply positive.

Digital Input Type
Inputs 5 and 6 can select these input types.



- Switch Input
- PWM Input
- Frequency Input
- Low Frequency Input (input 5 only)

There are two *Input Bias* choices:

- *Pull Up* – An internal resistor pulls the input pin voltage up to approximately ½ the supply voltage. This is intended for sinking devices or switches wired to ground.
- *Pull Down* – An internal resistor pulls the input pin voltage down to ground. This is intended for sourcing devices or switches wired to supply positive.

Out Connection

The out connection of the input blocks are unsigned integers. The range depends on the input type selected.

Analog Input Type (Inputs 1 through 4)
- 0-5 volt: range is 0 to 5000
- 0-10 volt: range is 0 to 10 000
- 0-20 milliamp: range is 0 to 2000
- 0-6000 ohms: range is 0 to 6000
- ERT120 Input (temperature sensor): range is 0 to 150 °C
- Switch Input, pull up: range is 0 or 1
- Switch Input, pull down: range is 0 or 1
- Switch Input, pull center: range is 0, 1, or 2

The switch input, pull to center bias has three states:
- 0 out, the input is connected to ground
- 1 out, the input is floating (open).
- 2 out, the input is switched to supply voltage.

Digital Input Type (Input 5)
- Switch Input, pull up: range is 0 or 2
- Switch Input, pull down: range is 0 or 2
- PWM Input: range is 0 to 1000 (0 to 100.0% duty cycle)
- Frequency Input: range is 60 to 10 000 hertz
- Low Frequency Input: range is 4 to 2000 hertz

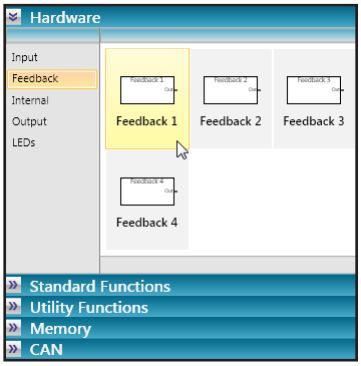
Digital Input Type (Input 6)
- Switch Input, pull up: range is 0 or 2
- Switch Input, pull down: range is 0 or 2
- PWM Input: range is 0 to 1000 (0 to 100.0% duty cycle)
- Frequency Input: range is 4 to 10 000 hertz

Note: *For both Frequency Input and Low Frequency Input, values below minimum are interpreted as zero.*

The switch input, pull up or pull down bias, can have two states.
- 0 out, the input is switched to ground or pulled down.
- 2 out, the input is switched to supply voltage or pulled up.

Feedback Blocks

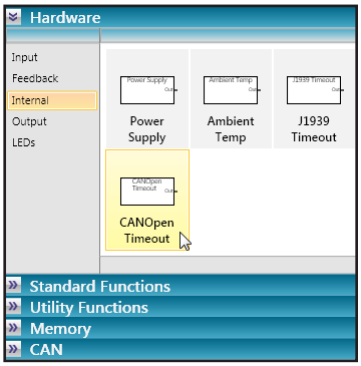
There are four *Feedback* blocks. Each one corresponds to an output block (1 through 4). The value of these blocks represents the current flowing through the output as measured. The output range is 0 to 2000 milliamps. The feedback block has no settings.



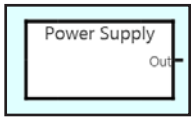
Internal Blocks

There are four *Internal* blocks. These blocks contain internal measurements of conditions on the driver. These include:

- Power Supply
- Ambient Temp
- J1939 Timeout
- CANopen Timeout



Power Supply Block



This block reads the incoming power supply voltage.

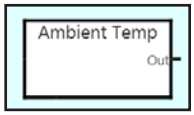
Out Connection

- This is the power supply voltage in millivolts.
- Value: 0 – 65535

Settings

This block has no settings.

Ambient Temp Block



This block reads the internal circuit board temperature.

Out Connection

- Range: 0 to 311
- Unit: degrees Fahrenheit

Settings

This block has no settings.

J1939 Timeout Block

This block monitors the J1939 Rx blocks. The output indicates if any blocks are in a timeout state. You can use this block to take action when an expected PDU message fails to arrive.

Out Connection

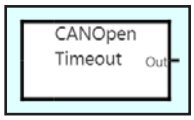
- Range: 0 to 1023
- Bitwise indication of J1939 Rx blocks that are in timeout state.

Bit	0	1	2	3	4	5	6	7	8	9
J1939 block	1	2	3	4	5	6	7	8	9	10
Value (dec)	1	2	4	8	16	32	64	128	256	512

Settings

This block has no settings.

CANopen Timeout Block



This block monitors the CANopen Rx blocks. The output indicates if any blocks are in a timeout state. You can use this block to take action when an expected PDO message fails to arrive.

Out Connection

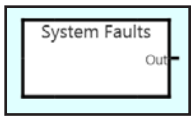
- Range: 0 to 1023
- Bitwise indication of CANopen Rx blocks that are in timeout state.

Bit	0	1	2	3	4	5	6	7	8	9
CANopen block	1	2	3	4	5	6	7	8	9	10
Value (dec)	1	2	4	8	16	32	64	128	256	512

Settings

This block has no settings.

System Faults



This block returns a value representing the type of system fault if one is active. If there is no active fault, the output of the block is zero.

Out Connection

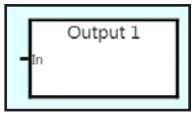
- Range: 0 to 4095
- Bitwise indication of all active system faults.

Bit	Dec.	System Fault
0	1	Power supply, level too high or too low
1	2	Output 1 fault
2	4	Output 2 fault
3	8	Output 3 fault
4	16	Output 4 fault
5	32	Output 5 fault
6	64	Outputs 1 and 2, DRV8301 controller fault
7	128	Outputs 3, 4 and 5, DRV8303 controller fault
8	256	Input 1, 20 mA current measurement, voltage high
9	512	Input 2, 20 mA current measurement, voltage high
10	1024	Input 3, 20 mA current measurement, voltage high
11	2048	Input 4, 20 mA current measurement, voltage high

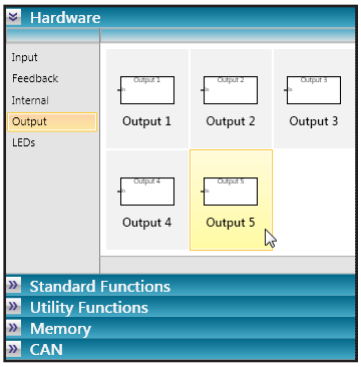
Settings

This block has no settings.

Output Blocks



The *Output* blocks are specific to the electrical pins on the ECDR-0506A. The *In* connection to the output blocks represents the command current in milliamps, or the duty cycle in 0.1% increments.



Outputs

These blocks correspond to the outputs of the ECDR-0506A.

Output	Connector X1 (gray)	Connector X2 (black)	Pin type
1	X1.3		Output
	X1.4		Return
2	X1.5		Output
	X1.6		Return
3		X2.12	Output
		X2.1	Return
4		X2.2	Output
		X2.3	Return
5		X2.4	Output
		X2.5	Return

Closed Loop Outputs (Output 1 through 4)

When the closed loop outputs are set to current control, the device uses feedback to maintain the current requested. The incoming value can be 0 to 2000 milliamps. Any value over 2000 is limited to 2000. Each output in the closed loop type has a corresponding feedback block that monitors the output current. Settings for this group include PWM frequency, and custom PI settings: proportional and integral.

Open Loop Output (Output 5)

When the open loop output is set to current control, it calculates the current according to Ohm's law. You must provide the coil resistance. The resistance of a coil varies with temperature so you should consider the need for accuracy in your application when using this output. The setting also has an entry for output frequency. You can enter the output frequency for optimum valve response.

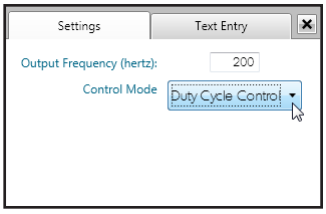
In Connection

Current Control
Using *Current Control*, *In* connection values are limited to 0 to 2000, any values over 2000 are interpreted as 2000. If you wish an output to behave as an on/off control, set the value to 2000 using a variable block. This applies all possible current to the output.

Duty Cycle Control
In *Duty Cycle Control*, *In* connection values are limited to 0 to 1000, any values over 1000 are interpreted as 1000. This value represents duty cycle per cent times 10; example: 155 = 15.5%.

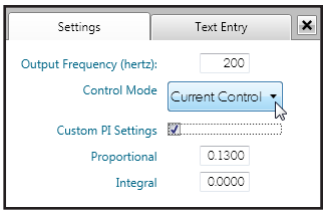
Settings

Output 1 – 4 (Closed Loop Settings)
Duty Cycle Control



- *Output Frequency* is the frequency of the PWM output in hertz.
  - Value: 40 – 400. Default is 200 Hz

Current Control

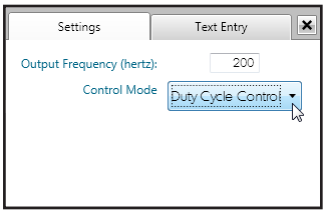


- *Output Frequency (hertz)* is the frequency of the PWM output in hertz.
  - Value: 40 – 400. Default is 200 Hz.
- *PI Settings*: Check this box to modify the PI control formula.
  - *Proportional* is the proportional gain the PI formula uses.
    - Value: 0 – 1
  - *Integral* is the integral gain the PI formula uses
    - Value: 0 – 1

Note: *The PI control formula does not use derivative gain. Generally hydraulic proportional valves act quickly and do not need derivative action to achieve responsive control. You can find a wealth of information on PID control theory in scholarly works and on the internet.*

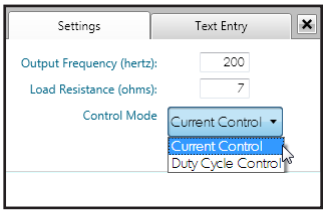
Warning
Unintended machine movement hazard
You must perform thorough testing and account for all possible operating and loading conditions when tuning a PI control. Selecting inappropriate gain values, it is possible to introduce issues such as overshoot and oscillation to the system.

Output 5 (Open Loop Settings)
Duty Cycle Control



- *Output Frequency (hertz)* is the frequency of the PWM output.
  - Value: 40 – 400. Default is 200 Hz.

Current Control



- *Output Frequency (hertz)* is the frequency of the PWM output.
  - Value: 40 – 400. Default is 200 Hz.
- *Load Resistance (ohms)* is the estimated coil resistance. The driver uses this value to calculate the output current based on measured voltage. See *Open Loop Output (Output 5) on page 18* for information on operation of this control mode.
  - Value: 4 – 100 ohms

Control Mode

The control mode allows you to choose between controlling current or controlling duty cycle.

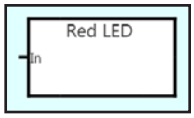
Current Control regulates the output to the command current which is from the *In* connection value of 0 to 2000 milliamps.

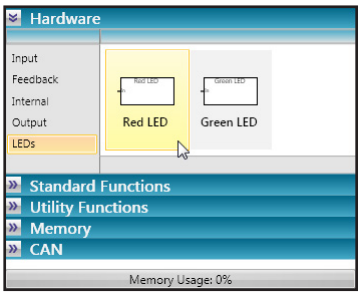
Duty Cycle Control commands the PWM output to a fixed duty cycle. The *In* connection value is 0 to1000 in increments of 0.1% duty cycle. 1000 is equal to 100% duty cycle, or full supply voltage to the output.

Caution
Potential circuit damage
When using duty cycle control without current feedback, a coil or load of low resistance can cause the output current to be too high. Take care to ensure outputs never operate above 2 amps. Estimate the maximum current, dividing the supply voltage by the coil resistance (Ohm's law). Driving the outputs too high results in the outputs shutting down. The ECDR-0506A has hardware protection to prevent damage, but repeated overcurrent shutdowns can eventually damage the device.

Note: *Output 5 can operate at 4 amps if used as an on/off control, or switched between 0 and 100% duty cycle.*

LED Blocks





There are blocks in the gallery to set the display of the *Red LED* or *Green LED* on the driver. The corresponding LED illuminates with a non-zero value at the *In* connection.

The normal behavior The normal behavior of the on-board LED is to pulse green while the ECDR-0506A is on and operating. When the unit detects a fault, the LED pulses red.

***Note:** . When the LED blocks are in use, this behavior is suspended and the on-board LEDs display as the program logic directs.*

In Connection

- Value: 0 – 100
- Sets the display brightness in increments of 1%.

Settings

This block has no settings.

STANDARD FUNCTION BLOCKS

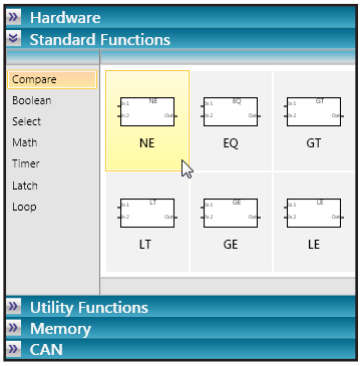
Overview

Standard function blocks are operations you can use to program the driver. The operation, inputs, outputs, and settings of the various functions are detailed here. The blocks are grouped into several categories. These are:

- Compare
- Boolean
- Select
- Math
- Timer
- Latch
- Loop

Compare Blocks

The compare blocks make logical comparisons of two values. There are six types. You may use these as many times in the main diagram as necessary.



Types of Comparison

- NE — Not Equal (≠)
  - Result is true when the comparators are not equal.
- EQ — EQual (=)
  - Result is true when the comparators are equal.
- GT — Greater Than (>)
  - Result is true when *In 1* is greater than *In 2*.
- LT — Less Than (<)
  - Result is true when *In 1* is less than *In 2*.
- GE — Greater or Equal (≥)
  - Result is true when *In 1* is greater than or equal to *In 2*.
- LE — Less or Equal (≤)
  - Result is true when *In 1* is less than or equal to *In 2*.

In 1 Connection

- The first of two comparators
- Value: 0 – 65535

In 2 Connection

- The second of two comparators
- Value: 0 – 65535

Out Connection

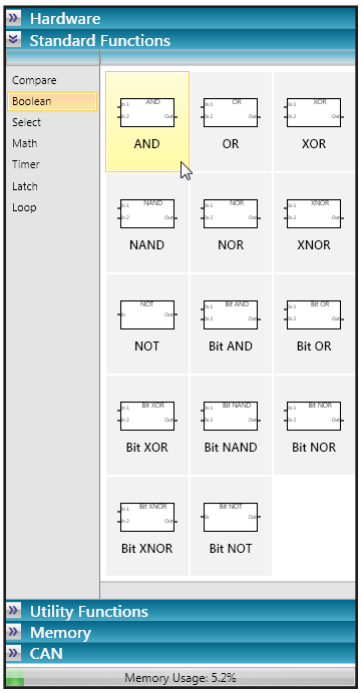
- The result of the comparison
- Zero if the comparison is false
- One if the comparison is true

Settings

These blocks have no settings.

Boolean (Logic) Blocks

The *Boolean* logic blocks perform boolean operations on input values. There are 14 blocks that each perform a unique type of operation: 7 boolean operators and 7 bitwise boolean operators. Boolean operators operate on true/false input (zero/non-zero), while bitwise operators operate on 16 bit input (0 – 65535). The bitwise blocks perform an operation on each binary bit of the input and express the result in the output.



Boolean Operators

You can select from 7 different boolean operators. The result of each of these operations is either 0 (false) or 1 (true). The operators are:

- AND – Result is true when both inputs are true.
- OR – Result is true when either input is true.
- XOR – Result is true when one input is true and one input is false.
- NAND – Result is true when any input is false.
- NOR – Result is true when both inputs are false.
- XNOR – Result is true when both inputs are false. Result is true when both inputs are true.
- NOT –Result is true when input is false. Result is false when input is true.

Bitwise Operators

The bitwise operators (prefix Bit) perform boolean operations on each binary bit of the 16 bit inputs. See examples below:

Bit AND

Connection	Decimal value	Binary value
In1	5	0110
In2	15	1111
Out	5	0110

Bit OR

Connection	Decimal value	Binary value
In1	5	0110
In2	15	1111
Out	15	1111

- Bit AND – Bitwise result is true when both input bits are true.
- Bit OR – Bitwise result is true when either input bits is true.
- Bit XOR – Bitwise result is true when one input bit is true and one is false.
- Bit NAND – Bitwise result is true when any input bit is false.
- Bit NOR – Bitwise result is true when both input bits are false.
- Bit XNOR – Bitwise result is true when both input bits are false. Result is true when both are true.
- Bit NOT –Bitwise result is true when input bit is false. Bitwise result is false when input bit is true.

In 1 Connection

- The first of two operands
- Value: 0 – 65535

In 2 Connection (if present)

- The second of two operands
- Value: 0 – 65535

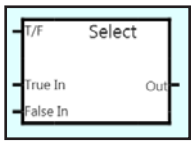
Out Connection

- The result of the operation
- Value: 0 – 65535
- In the case of binary results, zero is false and one is true

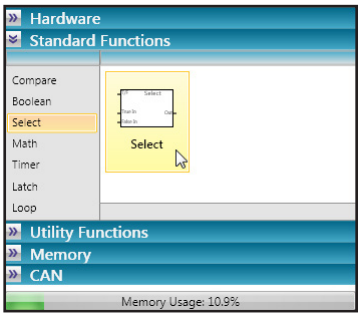
Settings

These blocks have no settings.

Select Block



The *Select* block selects between two input values based on the value of a third logic input. *Out* is the value of *True In* when *T/F* is non-zero, or it is the value of *False In* when *T/F* is zero.



T/F Connection

- The block uses this input to select the output value.
- Value: 0 – 65535
- Zero value is false, non-zero value is true.

True In Connection

- This is the value passed when T/F is true.
- Value: 0 – 65535

False In Connection

- This is the value passed when T/F is false.
- Value: 0 – 65535

Out Connection

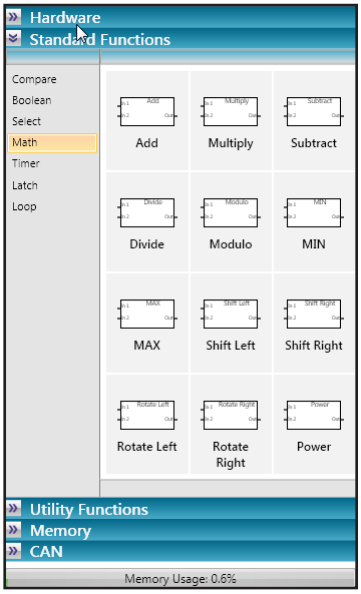
- This is the result of the select operation.
- Value: 0 – 65535

Settings

This block has no settings.

Math Blocks

The *Math* blocks can perform various arithmetic operations on two input values. All operands and results are unsigned 16 bit integers. You can choose to limit the result or allow it to rollover. The available math operations are:



- Add – Addition: *In 1* + *In 2*
- Subtract – Subtraction: *In 1* - *In 2*
- Multiply – Multiplication *In 1* x *In 2*
- Divide – Division: *In 1* ÷ *In 2 (result rounded to nearest integer)*
- Modulo – Modulus of *In 1* ÷ *In 2*
- Max – Comparison: greater of *In 1* or *In 2*
- Min – Comparison: lesser of *In 1* or *In 2*
- Shift Left – This is a bitwise operation on *In 1*. The result is 16 bits of *In 1* are shifted left by the count value of *In 2*. Zero is added to the right bit and the left bit is discarded. The *Limit Output* setting has no function.
- Shift Right – This is a bitwise operation on *In 1*. The result is 16 bits of *In 1* are shifted right by the count value of *In 2*. Zero is added to the left bit and the right bit is discarded. The *Limit Output* setting has no function.
- Rotate Left – This is a bitwise operation on *In 1*. The result is 16 bits of *In 1* are shifted left by the count value of *In 2*. The leftmost bits are rotated to the right side. The *Limit Output* setting has no function.
- Rotate Right – This is a bitwise operation on *In 1*. The result is 16 bits of *In 1* are shifted right by the count value of *In 2*. The rightmost bits are rotated to the left side. The *Limit Output* setting has no function.
- Power – Exponentiation: *In 1* ^ *In 2*

In 1 Connection

- The first operand
- Value: 0 – 65535

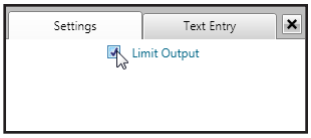
In 2 Connection

- The second operand
- Value: 0 – 65535

Out Connection

- The result of the arithmetic operation
- Value: 0 – 65535

Settings



Limit Output
Because the output is a 16 bit unsigned integer, the value is limited to 65535. The *Limit Output* setting prevents the result from rolling over. When the setting **is checked** the following rules apply:

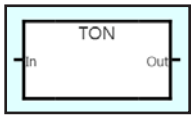
- Operations that exceed 65535 will result in a value of 65535.
- Operations that are less than zero will result in a value of zero.

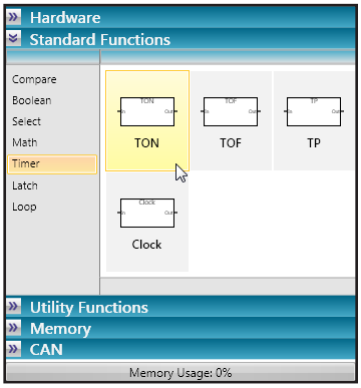
When the setting **is not checked**, these rules apply:

- Operations that exceed 65535 will result in rollover. The *Out* value is the expected value minus 65534. For example: the result of 65530+10 is 4.
- Operations that are less than zero will result in rollover, the *Out* value will be the expected value plus 65535. For example: the result of 4-10 is 65530.

Limit Output has no function for *Shift* and *Rotate* blocks.

Timer Blocks

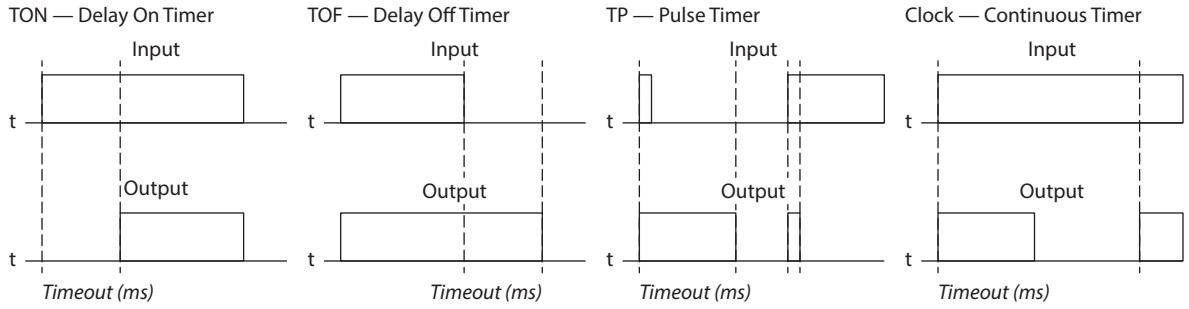




The *Timer* blocks provide delay on, delay off, and pulse timing. You can set the blocks to output either the state or the elapsed time. Timeout setting establishes the period or interval of each timer. Types of timer blocks are:

- TON – Delay on timer
  - Trigger on rising state change
  - Output state changes to true after a set time elapses.
- TOF – Delay off timer
  - Trigger on falling state change
  - Output state changes to false after a set time elapses.
- TP – Pulse timer
  - Output sends a pulse of set duration on the rising state change (zero to non-zero) of the input.
- Clock – Continuous clock
  - While input is true (non-zero) the output toggles from 0 to 1 at 50% duty cycle.

This diagram illustrates the input/output relationship of the various timers:



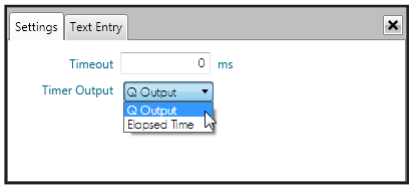
In Connection

- Starts the timer.
- Value: 0 – 65535. Zero if false and non-zero is true.

Out Connection

- The output of the timer can indicate status or the elapsed time.
- Status (*Q Output*) is true/false (1/0)
- *Elapsed Time* output is 0 – 65535 milliseconds. The output rolls over from 65535 to zero and continues counting up.

Settings



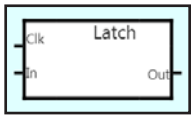
Timeout
The timing value is in milliseconds. You can set the value in increments of 1 millisecond, but in operation the processor can only respond within 5 to 10 milliseconds. The max value can be 32 bits or 4,294,967,295 milliseconds

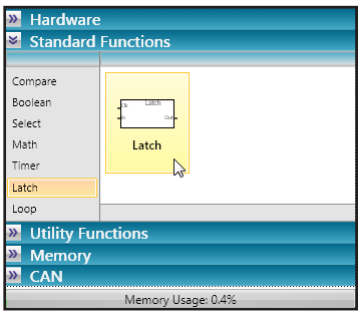
The output of the *Clock* block is a continuous timing running output. The duty cycle is 50% and the interval for each cycle is the *Timeout* setting.

Timer Output
You can select the *Out* value for the timing block. There are two types:

- *Q Output* – This is a boolean representation of the timer state. Zero is false and 1 is true.
- *Elapsed Time* – This is the elapsed time in milliseconds. When this setting is selected the output ranges from 0 to 65535 milliseconds. The output rolls over from 65535 to zero and continues counting up.

Latch Blocks





The *Latch* block holds a value. It only stores on the rising edge of the input.

Clk Connection

- Rising state change (zero to non-zero) triggers the latch.
- Value: 0 – 65535. Zero is false, non-zero is true.

In Connection

- Value to be stored
- Value: 0 – 65535

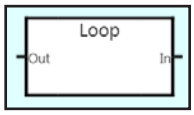
Out Connection

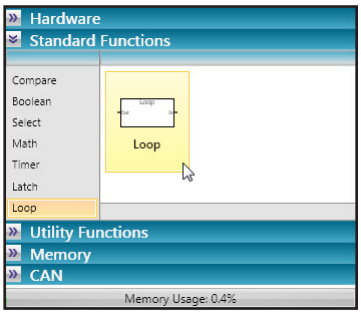
- The stored value: the value of *In* at the last time the latch was triggered.
- Value: 0 – 65535

Settings

This block has no settings.

Loop Blocks





The *Loop* block allows you to send values from right to left. Normally it is not allowed to connect an *Out* back to the left of the diagram. The values are always one scan behind when using this feature, meaning the *Out* value is always delayed (see *Diagram Processing* on page 12).

In Connection

- The value to be looped
- Value: 0 – 65535

Out Connection

- Output is the value of the *In* connection.
- Value: 0 – 65535

Settings

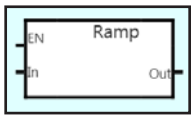
This block has no settings.

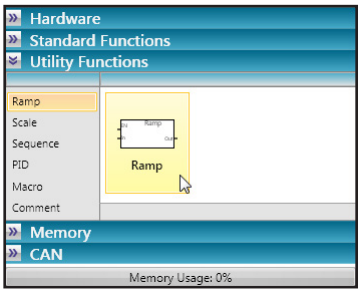
UTILITY FUNCTION BLOCKS

This set of blocks contains useful tools for manipulating the outputs, closed-loop control, reusable macros, and other useful utilities. The groups of blocks in this set include:

- Ramp
- Scale
- Sequence
- PID
- Service Control
- Macro
- Comment

Ramp Blocks





The *Ramp* blocks limit the rate of change of a value. They have the ability to smooth rapidly changing signals to avoid abrupt operation of the valves in the system. The ramp block has an enable input that allows ramping to be switched on and off. You can set the ramp up and ramp down rates independently. The ramp rates are expressed in units per second.

EN Connection

- Non-zero value enables ramping.
- Value: 0 – 65535. Zero is false, non-zero is true.

Note: *if the EN input is unconnected, the default value is 1 or True.*

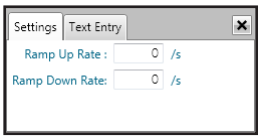
In Connection

- The input value
- Value: 0 – 65535

Out Connection

- The output value
- Output is only ramped when *En* is true.
- Value: 0 – 65535

Settings



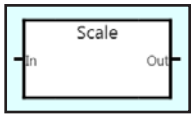
Ramp Up Rate

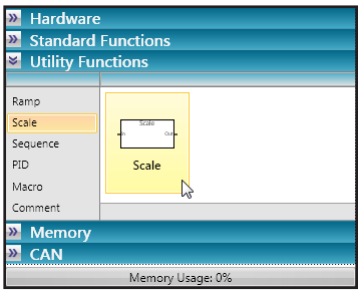
- This is the rate limit for rising signals. It applies whenever the *In* value is increasing.
- Units: units/second
- Example: given a setting of 500, and *In* changing abruptly from 0 to 1000, *Out* rises from 0 to 1000 over a period of 2 seconds (1000 ÷ 500).

Ramp Down Rate

- This is the rate limit for falling signals. It applies whenever the *In* value is decreasing.
- Units: units/second
- Example: given a setting of 500, and *In* changing abruptly from 1000 to 0, *Out* falls from 1000 to 0 over a period of 2 seconds (1000 ÷ 500).

Scale Blocks





The *Scale* block lets you set a ratio between input and output. You can create a multi-sloped scale curve with up to 3 breakpoints. You can use this block to create a deadband at the signal start and endpoints.

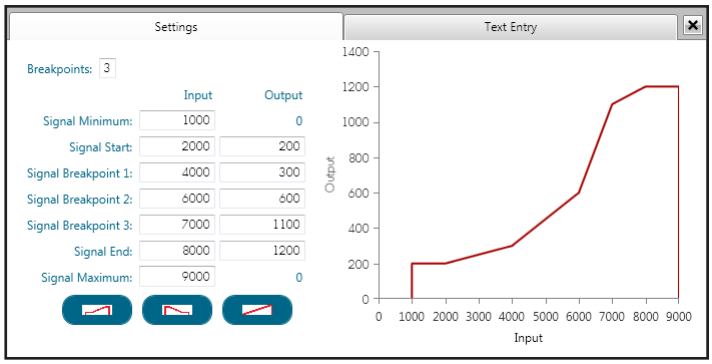
In Connection

- The input for the scale block
- Value: 0 – 65535

Out Connection

- The output of the scale block
- Value: 0 – 65535

Settings



Breakpoints
Sets the number of scale breakpoints. The limit is three.

Signal Minimum
This sets the lower signal limit. The area between *Signal Minimum* and *Signal Start* forms the lower deadband.

Signal Start
This is the coordinate (input, output) of the beginning of the scale curve.

Signal Breakpoints (1, 2, 3)
These are the coordinates (input, output) of the curve breakpoints. These allow you to form a multi-slope curve.

Signal End
This is the coordinate (input, output) of the end of the scale curve.

Signal Maximum
This is the maximum input signal. The area between the *Signal End* and *Signal Maximum* forms the upper deadband.

Preset Buttons
You can use the preset buttons to quickly create preset scaling profiles. You can then modify these to suit your application. The three choices include:

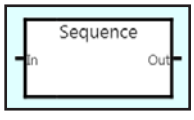
- Single slope – Creates a single rising slope with neutral deadband at signal start

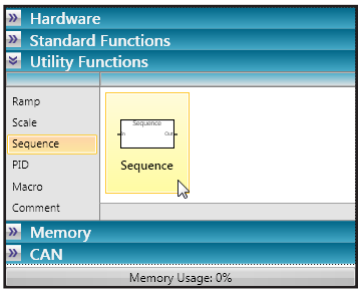
- Single slope reverse – Creates a single falling slope with neutral deadband at signal end

- Linear scale – Creates a linear rising slope with no deadband

Note: *Values cannot be entered until the connections are connected to another block.*

Sequence Blocks





The *Sequence* block is a six-period timing sequence. You can use this block to shape the lead edge of an output for applications like hot-shot or soft clutch control. Out values can step or ramp over time from one period value to the other. After the sequence period is completed, the output remains steady at the last value in the sequence and is held until the input goes to zero.

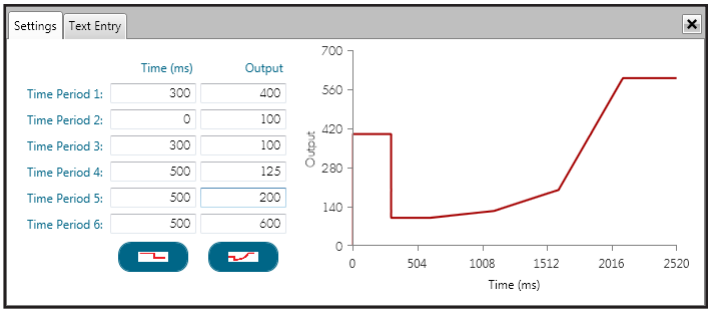
In Connection

- This is the trigger for the timing sequence. The sequence begins when the *In* value changes to non-zero (true).
- Value: 0 – 65535. Zero is false, non-zero is true.

Out Connection

- This is the output value. The sequence curve defines how it changes over time.
- Value: 0 – 65535

Settings



Time Periods (1 – 6)
There are six time periods that you can use to shape the lead edge of an output signal. You can set the duration (*Timer Block*) and the *Output* value for each period.

Time (ms)

- This is the length of the period in milliseconds.
- Value: 0 – 65535

Output

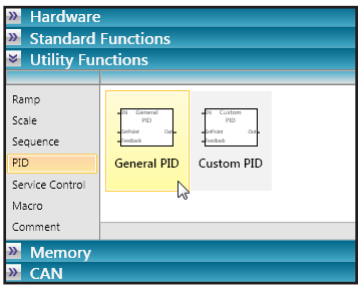
- This is the output value for the time period.
- Value: 0 – 3000

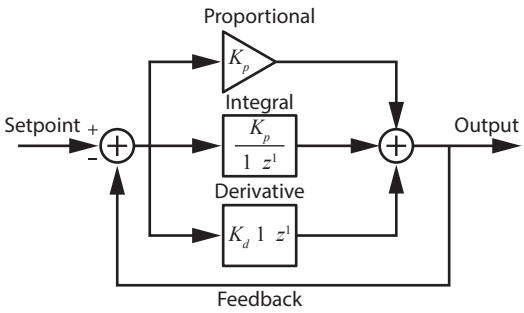
Preset Buttons
You can use the preset buttons to quickly create preset sequence profiles. You can then modify these to suit your application. The two choices include:

- Hotshot Profile – Creates a basic hotshot profile. Output is high for a period to overcome inertia or friction in the actuator, then drops to a lower level to maintain the state.
- Clutch Profile – Creates a basic clutch filling profile. Output starts out high to fill the clutch, then drops and rises smoothly to full engagement.

PID Blocks

The *PID* blocks allow you to create closed-loop control following the Proportional Integral Derivative (PID) theory. These blocks have three inputs: enable, set point, and feedback. Settings allow you to modify variables of the control formula affecting how it operates. The block varies the output to make the feedback equal to the setpoint. The underlying mathematics are discussed below.





There are two types of PID block:

- General PID
  - This is a simple PID control that allows you to set a gain coefficient for each of the terms.
- Custom PID
  - This is a more sophisticated PID control that includes the additional features of *Windup Guard* and *Integral Window*.
  - With this control, the coefficient for the three terms is inverse. This means higher numbers decrease the action of the individual terms on the overall output.

EN Connection

- This is the enable input for the block.
- Zero value initializes the PID and output goes to zero.
- Non-zero value enables the block to operate.
- Value: 0 – 65535. Zero is false, non-zero is true.

SetPoint Connection

- This is the desired condition.
- Value: 0 – 65535

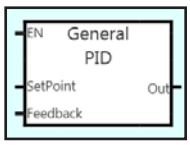
Feedback Connection

- This is the measured condition. Typically this is connected to one of the six inputs.
- Value: 0 – 65535

Out Connection

- This is the block output. The block manipulates the output to drive the feedback equal to the setpoint.
- Value: 0 – 65535

General PID Block



The *General PID* block is a Proportional Integral Derivative (PID) controller. It is a generic feedback control loop mechanism widely used in industrial control systems. A PID controller is the most commonly used type of feedback controller. You can manipulate the mathematical formula driving the PID output using several variables. These are the *Proportional Gain (K_p)*, *Integral Gain (K_i)*, *Derivative Gain (K_d)*.

Algorithm

$$y[n] = y[n-1] + A0 * x[n] + A1 * x[n-1] + A2 * x[n-2]$$

$$A0 = K_p + K_i + K_d$$
$$A1 = (-K_p) - (2 * K_d)$$
$$A2 = K_d$$

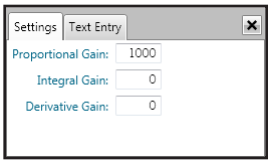
Where:

- K_p is proportional constant
- K_i is integral constant
- K_d is derivative constant
- $y[n]$ is an array of output values
- $x[n]$ is an array of error states

Operation

The PID controller calculates an *error* value as the difference between the measured output and the reference (feedback) input. The controller attempts to minimize the error by adjusting the process control inputs. The proportional value determines the reaction to the current error, the integral value determines the reaction based on the sum of recent errors, and the derivative value determines the reaction based on the rate at which the error has been changing.

Settings



Proportional Gain (K_p)

- This is the proportional constant. It weights the proportional action of the PID control.
- The proportional term is the reaction to the magnitude of the current error.
- Value: 0 – 65535

Integral Gain (K_i)

- This is the integral constant. It weights the integral action of the PID control.
- The integral term represents the reaction to the sum of previous errors.
- Value: 0 – 65535

Derivative Gain (K_d)

- This is the derivative constant. It weights the derivative action of the PID control.
- The derivative term represents the reaction to the rate of change of the error.
- Value: 0 – 65535

Warning
Unwanted machine movement hazard
Hydraulic proportional valves are generally responsive enough that derivative action is not required and is more likely to yield unwanted behavior. Unless you are controlling something other than a hydraulic proportional valve with a longer period (≥15 minutes to reach setpoint) leave the Derivative Time setting at zero.

Simple PID Tuning Procedure for the General PID Block.

This procedure helps guide you to find a quick response from your PID control system. We suggest more tuning after following this procedure, as well as reading up on PID control and PID tuning. If you feel uncomfortable with any of this terminology please consult an expert, textbook, or reliable online resources.

Warning
Unwanted machine movement hazard
Always tune your control in a laboratory environment to ensure reasonably stable and predictable operation before installing on a machine.

1. Set all the values to zero to begin.
2. Set the *Proportional Gain* value. Smaller values retard the system's response to error. Larger values speed the system's response, but too large of a value can cause overshoot and oscillating output. Start with fair value, possibly 1000 – 5000.

Note: *Proportional only control will produce a steady state error, or offset from the setpoint. Adding an integral term corrects this offset.*

3. Download the program to the ECDR-0506A and use the *Monitor Screen* to test the performance (see *Monitor Screen on page 50*). The desired output should have just a little overshoot and then come to a steady state output. If there is no overshoot, double the *Proportional Gain* value until you have very little overshoot. If the output is oscillating and overshoot is too high, halve the *Proportional Gain* value until you have very little overshoot. Once you find an output with little overshoot, stop this part and make note of *Proportional Gain* value.
4. Now set the *Integral Gain*. Possible starting values are 100 to 1000. If there is no overshoot and it takes to too long to reach steady state, double the *Integral Gain* value until there is some overshoot and no oscillation in the output. If the output is oscillating, halve *Integral Gain* until these oscillations stop.

   Download the program to the ECDR-0506A and retest. Once you have very little or no overshoot, and the output is reaching the setpoint reliably, you may stop.

Note: *If you have followed all these steps correctly, you should have a decent controller at this point. You may choose to leave the Derivative Gain at zero if performance is satisfactory.*

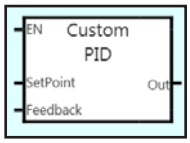
Warning

Unwanted machine movement hazard

Hydraulic proportional valves are generally responsive enough that derivative action is not required and is more likely to yield unwanted behavior. Unless you are controlling something other than a hydraulic proportional valve with a longer period (≥15 minutes to reach setpoint) leave the Derivative Gain setting at zero. If you need to add derivative action to your control block, ensure you have an expert understanding of the concepts and the behavior of the controlled system.

5. Lastly, if needed, set the *Derivative Gain*. Possible starting values are 100 to 5000. If the output surges or needs additional damping, double the *Derivative Gain* value. If oscillations occur, halve the *Derivative Gain* value. When you have the desired control, you may stop.
6. Download the program to the ECDR-0506A and retest.

Custom PID Block

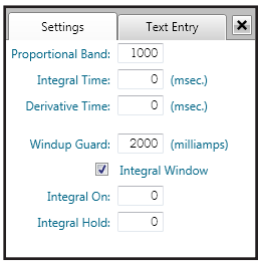


The *Custom PID* block is a PID controller with additional control features. You can manipulate the mathematical formula driving the PID output using several variables. These are the *Proportional Band*, *Integral Time*, *Derivative Time, Windup Guard,* and *Integral Window*.

Operation

The PID controller calculates an *error* value as the difference between the measured output and the reference (feedback) input. The controller attempts to minimize the error by adjusting the process control inputs. The proportional value determines the reaction to the current error, the integral value determines the reaction based on the sum of recent errors, and the derivative value determines the reaction based on the rate at which the error has been changing. Windup guard and integral window set limits on integral calculation to control windup and limit unwanted oscillation.

Settings



Proportional Band (K_p)

- In the PID control formula, the proportional term produces an output proportional to the current error value. The error value is defined as:
  $e = SetPointValue — Feedback$
- This parameter amplifies the error value of the output to determine the proportional term. With *Proportional Band* $K_P \in [1, 10\,000]$, we define the proportional term:

  $K_p \in [1, 10000], P = 2000 * \frac{e}{K_p}$

  $P = 0, K_p = 0$
- Value: 0 – 65535

Integral Time (K_i)

- With the integral term, the output is proportional to the magnitude and the duration of the error. With *Integral Time* $K_i \in [1, 10\,000]$, we define the Integral term:

  $K_i \in [1, 10000], I = I + \left(e * \frac{dt}{K_i}\right)$

  $I = 0, K_i = 0$
- Value: 0 – 65535 milliseconds

Derivative Time (K_d)

- With the derivative term, the output is proportional to the rate of change of the error signal. To calculate this term we must save the previous error signal after each calculation so that it may be used in the next calculation. With *Derivative Time* $K_d \in [1, 10\,000]$, we define the derivative term:

  $K_d \in [1, 10000], D = (D + de) - \left(\frac{D}{K_d}\right)$

  $D = 0, K_d = 0$

- Value: 0 – 65535 milliseconds

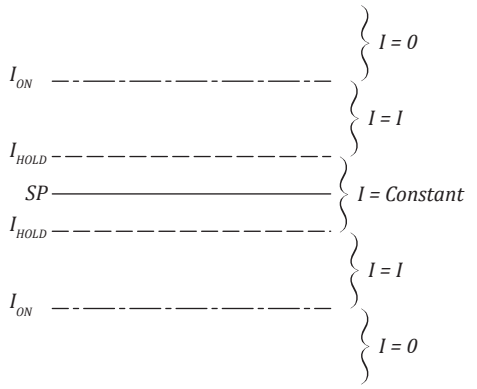
Warning
Unwanted machine movement hazard
Hydraulic proportional valves are generally responsive enough that derivative action is not required and is more likely to yield unwanted behavior. Unless you are controlling something other than a hydraulic proportional valve with a longer period (≥15 minutes to reach setpoint) leave the Derivative Time setting at zero.

Windup Guard

- A large change in setpoint can cause integral windup. The integral term accumulates a significant error (winds up) and can drive the output beyond the controllable area. It can take some time for this error to correct (unwind).
- This value limits the maximum output to prevent integral windup. Generally this limit is set to the maximum current of the controlled coil.
  $-Windup \le P, D \le Windup$
  $0 \le I \le Windup$
- After bounding our terms with windup control, we now sum up our terms and bound the output by the *Windup Guard* value.
  $0 \le (Output = P + I + D) \le Windup$
- Value: 0 – 65535

Integral Window

- These parameters set operating bands, or window, for the integral term. Checking the *Integral Window* box enables control over when the integral term is used in calculating the output.
  $I = I, IntegralHold < |e| < IntegralOn$
- *Integral On* sets the range for integral calculation. It is an absolute value. When the error is greater than this value, the integral term is set at zero.
  $I = 0, |e| > IntegralOn$
- *Integral Hold* sets the range to suspend integral calculation. It is an absolute value. It must be less than the *Integral On* value. When the error is less than this value, the integral term is kept constant at the previous value.
  $I = Constant, |e| < IntegralHold$



Integral Window settings create operating bands for the integral term.

Simple PID Tuning Procedure for the Custom PID Block.

This procedure helps guide you to find a quick response from your PID control system. We suggest more tuning after following this procedure, as well as reading up on PID control and PID tuning. If you feel uncomfortable with any of this terminology please consult an expert, textbook, or reliable online resources.

Warning
Unwanted machine movement hazard
Always tune your control in a laboratory environment to ensure reasonably stable and predictable operation before installing on a machine.

1. Set all the values to zero to begin.
2. Set the *Proportional Band* value. Larger values retard the system's response to error. Smaller values cause the system to respond faster, but too small of a value can cause overshoot and oscillating output. Start with fair value, possibly 1000 – 5000.

Note: *Proportional only control will produce a steady state error, or offset from the setpoint. Adding an integral term corrects this offset.*

3. Set the *Windup Guard* value. Set this value to the maximum output of your controlled device (coil, etc.)
4. Download the program to the ECDR-0506A and use the *Monitor Screen* to test the performance (see *Monitor Screen on page 50*). The desired output should have just a little overshoot and then come to a steady state output. If there is no overshoot, halve the *Proportional Band* value until you have very little overshoot. If the output is oscillating and overshoot is too high, double the *Proportional Band* value until you have very little overshoot. Once you find an output with little overshoot, stop this part and make note of *Proportional Band* value.
5. Now set the *Integral Time*. You can use the same values for *Proportional Band* and *Windup Guard*. Possible starting values are 100 to 1000 milliseconds. If there is no overshoot and it takes to too long to reach steady state, halve the *Integral Time* value until there is some overshoot and no oscillation in the output. If the output is oscillating, double *Integral Time* until these oscillations stop.

   Download the program to the ECDR-0506A and retest. Once you have very little or no overshoot, and the output is reaching the setpoint reliably, you may stop.
6. If desired, set the integral window. Check the *Integral Window* box in the settings window to show the settings. Set the *Integral On* value to the maximum error value you want for integral calculation. Absolute error values above this value result in an integral term of zero ($\boldsymbol{I = 0}$). Set the *Integral Hold* to the value you want to suspend integral calculation. Absolute error values less than this value result in a constant integral term ($\boldsymbol{I = Constant}$).

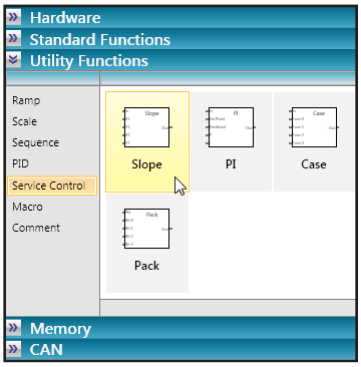
Note: *If you have followed all these steps correctly, you should have a decent controller at this point. You may choose to leave the Derivative Time at zero if performance is satisfactory.*

Warning
Unwanted machine movement hazard
Hydraulic proportional valves are generally responsive enough that derivative action is not required and is more likely to yield unwanted behavior. Unless you are controlling something other than a hydraulic proportional valve with a longer period (≥15 minutes to reach setpoint) leave the Derivative Time setting at zero. If you need to add derivative action to your control block, ensure you have an expert understanding of the concepts and the behavior of the controlled system.

7. Lastly, if needed, set the *Derivative Time*. You can use the same values for *Proportional Band, Integral Time,* and *Windup Guard*. Possible starting values are 100 to 5000 milliseconds. If the output surges or needs additional damping, halve the *Derivative Time* value. If oscillations occur, double the *Derivative Time* value. When you have the desired control, you may stop.
8. Download the program to the ECDR-0506A and retest.

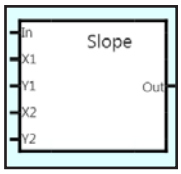
Service Control

The service control group includes some blocks that duplicate other functions such as scale and PID control. Using these blocks you can set up your configuration using retain blocks to set the operating parameters, because although the service tool does not allow changing the configuration or block settings, it does allow changing the value stored in retain blocks. If you are planning a deployment of the service tool for your configured drivers, and you want technicians to tune parameters in the field, consider using these blocks.



- Slope—similar to Scale block
- PI—similar to PID block
- Case—similar to the select block with four input values
- Pack—assembles 4 bits into a nybble

Slope Block



The slope block performs a similar function as the scale bock with limitations. You can specify two x/y pairs to create a slope that the block uses to scale the input. The formula for calculation is:

Out = In(Y2-Y1/X2-X1)

In Connection
- The input to the block
- Value: 0–65535

X1 Connection
- The X position of the first coordinate
- Value: 0–65535

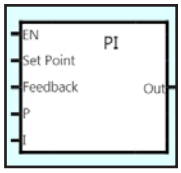
Y1 Connection
- The Y position of the first coordinate
- Value: 0–65535

X2 Connection
- The X position of the second coordinate
- Value: 0–65535

Y2 Connection
- The y position of the second coordinate
- Value: 0–65535

Out Connection
- The output of the block
- Value: 0–65535
- The calculation is performed as a floating point operation and converted to an unsigned integer.
- The result does not rollover.

Settings
This block has no settings.



PI Block (Closed Loop Control)

This block is a closed-loop proportional/integral control without derivative action. This block has inputs for the proportional and integral gain instead of settings. This allows you to develop an adaptive control that changes its behavior in response to conditions during runtime. You can also connect the P and I inputs to retain blocks to allow tuning after deployment with the service tool.

Operation
For details on operation see *General PID Block* on page 30. The derivative gain for this control is always set to zero.

EN Connection
- This is the enable input for the block.
- Zero value initializes the PID and output goes to zero.
- Non-zero value enables the block to operate.
- Value: 0 – 65535. Zero is false, non-zero is true.

SetPoint Connection
- This is the desired condition.
- Value: 0 – 65535

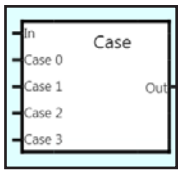
Feedback Connection
- This is the measured condition. Typically this is connected to one of the six inputs.
- Value: 0 – 65535

P Connection—Proportional Gain (K_p)
- This is the proportional constant. It weights the proportional action of the PI control.
- The proportional term is the reaction to the magnitude of the current error.
- Value: 0 – 65535

I Connection—Integral Gain (K_i)
- This is the integral constant. It weights the integral action of the PI control.
- The integral term represents the reaction to the sum of previous errors.
- Value: 0 – 65535

Out Connection
- This is the block output. The block manipulates the output to drive the feedback equal to the setpoint.
- Value: 0 – 65535

Settings
This block has no settings.



Case

This is a logic block that passes one of four values to the output based on the 2-bit value of the input. Although the input range is 16 bits, the control only acts on the first two. All higher bits are ignored. See below for a truth table defining operation.

In value (dec)	In value (binary)	Out value
0	00	Case 0 connection
1	01	Case 1 connection
2	10	Case 2 connection
3	11	Case 3 connection

In Connection
- The input value. This value determines which value to send to the output.
- Value: 0–65535. Only the first two bits are used for evaluation.

Case 0 Connection
- The value passed to the output when the In connection value is 0 (binary 00)
- Value: 0–65535

Case 1 Connection
- The value passed to the output when the In connection value is 1 (binary 01)
- Value: 0–65535

Case 2 Connection
- The value passed to the output when the In connection value is 2 (binary 10)
- Value: 0–65535

Case 3 Connection
- The value passed to the output when the In connection value is 3 (binary 11)
- Value: 0–65535

Out Connection
- The output value
- Value: 0–65535

Settings
This block has no settings.

Pack

The pack block assembles bits into a nybble (half byte). Four inputs define the bits. The output is an assembly of the four packed bits. You can cascade multiple pack blocks to form a byte or word.

EN Connection
- Non-zero (true) value enables the block. Zero (false) value drives the output to zero.
- Value: 0–65535. Zero is false, non-zero is true.

Bit 0 Connection
- Bit 0 of the nybble
- Value: 0–65535. Zero is false, non-zero is true.

Bit 1 Connection
- Bit 1 of the nybble
- Value: 0–65535. Zero is false, non-zero is true.

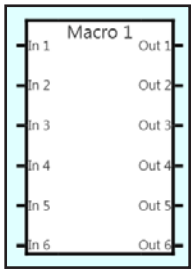
Bit 2 Connection
- Bit 2 of the nybble
- Value: 0–65535. Zero is false, non-zero is true.

Bit 3 Connection
- Bit 3 of the nybble
- Value: 0–65535. Zero is false, non-zero is true.

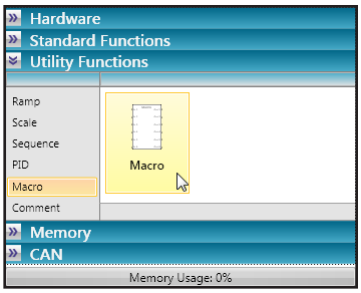
Out Connection
- The resultant assembled nybble
- Value 0–15 (decimal)

Settings
This block has no settings.

Macro Blocks

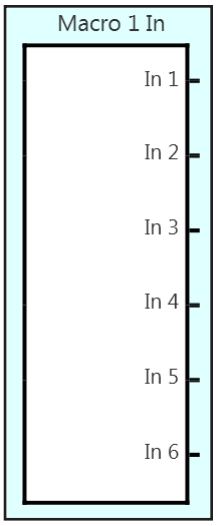


The *Macro* block allows you to organize your logic. You can package complex diagrams inside a macro to make the main diagram cleaner and easier to read. The programming in the macro is the same diagram programming as the main diagram, except the set of hardware blocks are not available in the macro diagram. The macro block also provides import and export features. This is useful for creating libraries of macros, so you can easily reuse logic across different projects. There are a maximum of 10 macro blocks available.



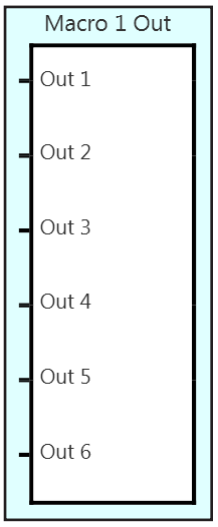
In Connection

There are 6 *In* connections on the macro block. You do not have to connect all inputs in order to function. The 6 inputs correspond to the inputs on the macro block in the main diagram. These connections pass values from the main diagram.



Out Connection

There are 6 *Out* connections on the macro block. You do not have to connect all outputs in order to function. The 6 outputs correspond to the outputs on the macro block in the main diagram. These connections pass values to the main diagram.

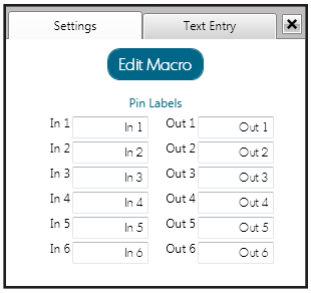


Settings

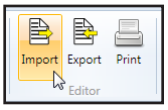
Pin Labels
You may enter customized input and output labels. Type text into the desired box to modify the labeling.

Edit Macro

Click the **`Edit Macro`** button in the settings window to bring up the macro diagram. On the macro diagram you may build a program using the function blocks from the gallery. The hardware blocks (inputs, outputs, feedback, etc.) are not available on the macro diagram. To use any of these you must connect them to an input or an output of your macro. Click **`Main Diagram`** on the left side of the screen to return to the editing the *Main Diagram*.



Importing and Exporting Macros

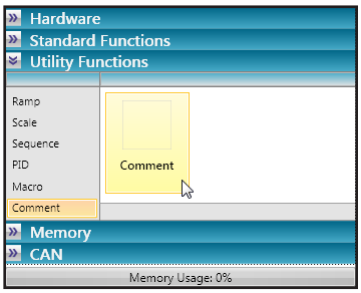


The macro blocks are useful for reusing your code in multiple programs. Using the import/export functions you can create a library of macros for different purposes. When in the *Macro Diagram*:

- Click **`Import`** to retrieve a previously saved macro.
- Click **`Export`** to save the current macro.

Comment Block

The *Comment* block allows you to document the diagram and to segregate areas. The comment block always remains behind all other blocks and connectors in the diagram. You can enter text to explain operation of the diagram. You can also resize and move the comment block to surround groups of blocks. You also have the option of color coding comment blocks.



Adding Comments

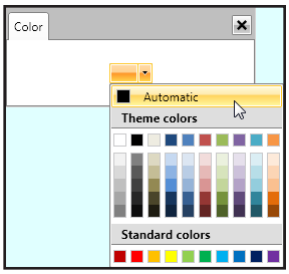
Simply type text into the block to add comments to your diagram.

Moving and Resizing

Click the open area of the block to select it. You can then drag the block anywhere on the diagram. You can also drag the corner sizing handles to change the size of the block.

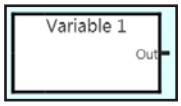
Settings

Click the color tab in the settings window to set the background color of the block. Use the Windows color picker to select a custom background color.

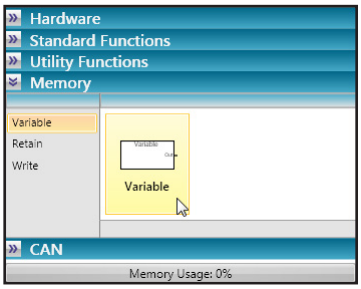


MEMORY BLOCKS

Variable Blocks



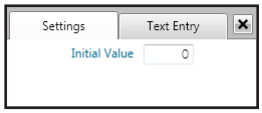
The *Variable* blocks can store a constant or variable value. There are 127 blocks available in the *Main Diagram*. They are numbered sequentially as you drag them from the gallery. You can change the stored value using the *Write* block (see *Write Blocks on page 41*).



Out Connection

- This provides the stored value to another block.
- Value: 0 – 65535

Settings

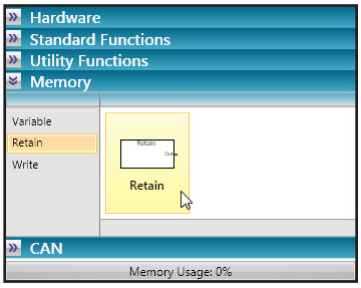


Initial Value

- This is the value at initialization.
- The *Write* block can change the stored value during operation, but it returns to the *Initial Value* on power up.
- Value: 0 – 65535

Retain Blocks

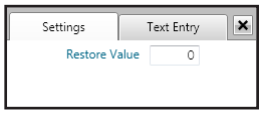
The *Retain* blocks can store a constant or a changeable value, and retain it through a power cycle. Each block represents a distinct variable. There are 127 blocks available in the *Main Diagram*. They are numbered sequentially (indexed) as you drag them from the gallery. The value is initialized with the *Restore* value on power up. The *Write* block can change the block value. Once this occurs, the restore value is ignored on further power cycles. If you click the **`Erase Settings`** button and reload the program, the value will revert to the *Restore* setting.



Out Connection

- This provides the stored value to another block.
- Value: 0 – 65535

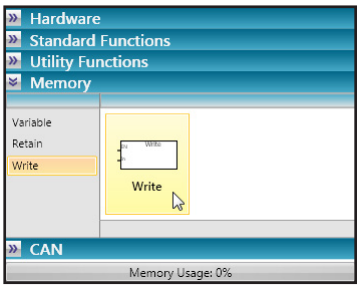
Settings



Restore Value

- This is the value at initialization.
- The *Write* block can change the stored value during operation. Once this happens, the *Restore* value is ignored on all subsequent power cycles.
- If you *Erase Settings* and reload the program, the block will re initialize with the *Restore* value.
- Value: 0 – 65535

Write Blocks



The *Write* block allows you to change the value of *Variable* and *Retain* blocks. The settings specify the target block. The diagram can have multiple write blocks. Multiple write blocks can write to the same variable. The input sets the value to write. The enable input allows the write operation to occur.

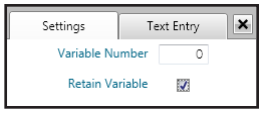
EN Connection

- Non-zero value enables the write operation. The block stores the value of the *In* connection to the target block *on the next scan*.
- Value: 0 – 65535. Zero is false, non-zero is true.

In Connection

- This is the value to write.
- Value: 0 – 65535

Settings



Variable Number

- This is the index of the variable you are addressing with the write operation.
- Value: 1 – 127

Retain Variable

- Check this box to address *Retain* blocks instead of *Variable* blocks.

Operation

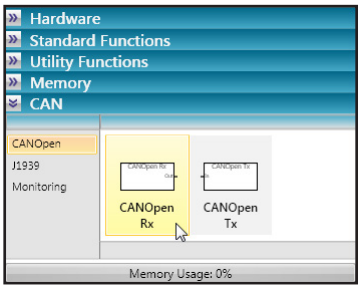
When a true value is seen at the *EN* input, the block updates the stored value of the targeted *Variable* or *Retain* block with the *In* value on the next scan.

With Variable Blocks
The stored value is updated. The *Variable* block returns to its *Initial Value* on the next power cycle.

With Retain Blocks
The stored value is updated. The write operation sets a flag in the *Retain* block that causes the driver to retain the stored value through a power cycle. Only erasing and reloading the program can reset this internal flag.

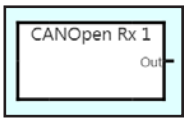
CAN BLOCKS

CANopen Blocks



The *CANopen Rx* and *CANopen Tx* blocks allow you to receive and transmit messages on the connected CANopen bus. The CANopen type is intended to work with a PDO (Process Data Object) message from another device. You can specify the COB-ID, start bit, data length, and timeout for the message.

Note: *Details of the CANopen communication standard are beyond the scope of this document. Please refer to CANopen documentation from CAN in Automation (CiA) for further information on the communication standard.*



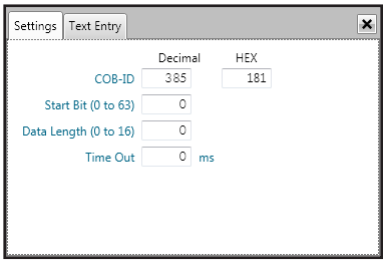
CANopen Rx Block

This block receives a PDO message from the CANopen network . You can set COB ID values from 385 to 1407 decimal. For convenience the COB ID is displayed in both decimal and hex.

Out Connection

- This is the value of the received 16-bit message.
- Value: 0 – 65535
- If you specify a *Data Length* shorter than 16 bits, the data is truncated.

Settings



COB-ID

- This is the Communication Object IDentifier for the device whose message you wish to receive.

Start Bit

- The start bit for the message
- Value: 0 – 63

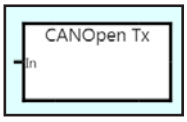
Data Length

- This is the length of the data retrieved from the CAN message, in bits. The output range of the block changes based on this value. You can compute the range converting from binary: ***0*** through (2^n-1) where n = ***Data Length***.
- Value: 0 – 16

Timeout

- The message timeout in milliseconds
- If the incoming message does not arrive, a flag is set in the *CANopen Timeout* block (see *CANopen Timeout Block* on page 17).
- Value: 0 – 65535

Note: *If the timeout value is set to zero, then the Out value holds to the last value received. Also, the CANopen timeout block does not report a timeout.*



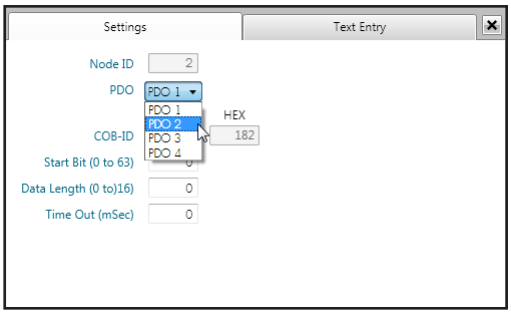
CANopen Tx

This block transmits a message on the CANopen network as a Process Data Object (PDO). You can set COB ID values from 385 to 1407 decimal. For convenience the COB ID is displayed in both decimal and hex.

In Connection

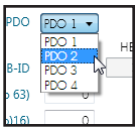
- This is the message to transmit.
- Value: 0 – 65535
- If you specify a *Data Length* shorter than 16 bits, the data is truncated.

Settings



COB-ID

- This is the Communication Object IDentifier for the device to which you wish to transmit a message.
- The value range depends on the *Node ID* selected in *CAN Open Settings*. See *CAN Open Settings on page 36*.



PDO (available when a *Node ID* is specified)

- The Process Data Object
- Select from the list.

Note: *If you select a Node ID on the CAN Settings screen, then your COB ID value is calculated depending on which PDO you chose. The fields: Node ID and COB-ID will not accept values unless Node ID is zero.*

Start Bit

- The start bit for the message
- Value: 0 – 63

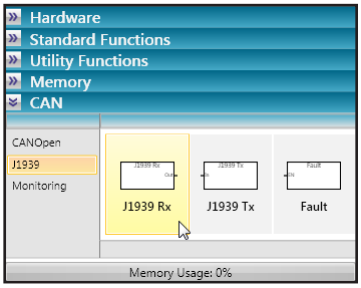
Data Length

- This is the length of the data retrieved from the CAN message, in bits. The output range of the block changes based on this value. You can compute the range converting from binary: ***0*** through (2^n-1) where ***n = Data Length***
- Value: 0 – 16

Timeout

- The message transmission interval in milliseconds
- The ECDR-0506A periodically transmits the message at this rate.
- Value: 0 – 65535

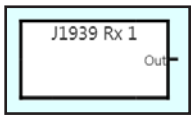
SAE J1939 Blocks



The *SAE J1939* blocks allow you to receive and transmit messages on the connected SAE J1939 CAN bus. These blocks are intended to work with a PDU (Protocol Data Unit) message from/to another device. You can specify PDU-1 or PDU-2 type, Parameter Group Number (PGN), Destination Address (DA), Source Address (SA) filter, start bit, data length, and timeout for the message.

Note: *Details of the SAE J1939 CAN communication standard are beyond the scope of this document. Please refer to documentation from SAE for further information on the communication standard.*

SAE J1939 Rx



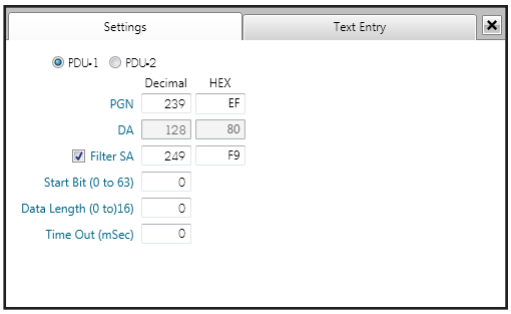
The *J1939 Rx* block receives a PDU message from the CAN network. You can use PDU-1 to receive messages from a specific device or PDU-2 to receive broadcast messages. The output from the block is an unsigned integer from 0 to 65535. This value is truncated if you choose a data length shorter than 16 bits.

There are 10 J1939 Rx blocks allowed in the *Main Diagram*. They are sequentially numbered as you drag them from the gallery.

Out Connection

- This is the value of the received message.
- Value: 0 – 65535
- If you specify a *Data Length* shorter than 16 bits, the data is truncated.

Settings



PDU

- The Protocol Data Unit type
- Select PDU-1 or PDU-2

PGN

- The Parameter Group Number
- Value:
  - 0 – 232 (0 – E8) for PDU-1
  - 240 – 65535 (00E0 – FFFF) for PDU-2

DA (PDU-1 only)

- The Destination Address of the ECDR-0506A
- This value is set in CAN Settings. See *CAN Settings Screen on page 8*.

Filter SA

- Check the box to filter for a specific source address.
- Enter the SA in decimal or HEX format.
- Value: 0 – 254 (0 – F9)

Start Bit

- The start bit for the message
- Value: 0 – 63

Data Length

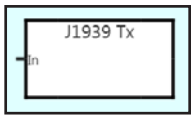
- This is the length of the data retrieved from the CAN message, in bits. The output range of the block changes based on this value. You can compute the range converting from binary: ***0*** through (2^n-1) where ***n = Data Length***.
- Value: 0 – 16

Timeout

- The message timeout in milliseconds
- If the incoming message does not arrive, a flag is set in the *J1939 Timeout* block (see *J1939 Timeout Block* on page 16).
- Value: 0 – 65535

Note: *If the timeout value is set to zero, then the Out value holds to the last value received. Also, the SAE J1939 timeout block does not report a timeout.*

SAE J1939 Tx

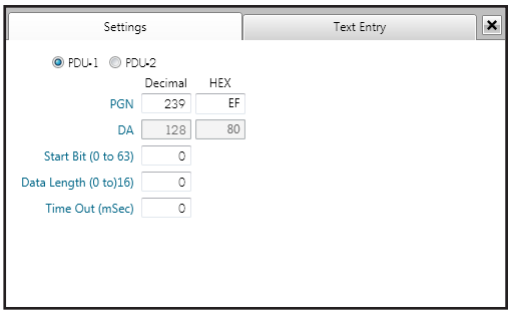


The *J1939 Tx* block transmits a PDU message to the CAN network. You can use PDU-1 or PDU-2 to transmit messages on the network. The input from the block is an unsigned integer from 0 to 65535. This value is truncated if you choose a data length shorter than 16 bits.

In Connection

- This is the value of the transmitted message.
- Value: 0 – 65535
- If you specify a *Data Length* shorter than 16 bits, the message is truncated.

Settings



PDU

- The Protocol Data Unit type
- Select PDU-1 or PDU-2

PGN

- The Parameter Group Number
- Value:
  - 0 – 239 (0 – EF) for PDU-1
  - 240 – 65535 (00E0 – FFFF) for PDU-2

DA (PDU-1 only)

- The Destination Address of the ECDR-0506A
- This value is set in CAN Settings. See *CAN Settings Screen on page 8*.

Start Bit

- The start bit for the message
- Value: 0 – 63

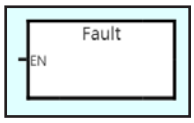
Data Length

- This is the length of the data inserted into the CAN message, in bits. The output range of the block changes based on this value. You can compute the range converting from binary: ***0*** through (2^n-1) where ***n = Data Length***
- Value: 0 – 16

Timeout

- The interval for sending the message in milliseconds
- Value: 0 – 65535

Fault Block

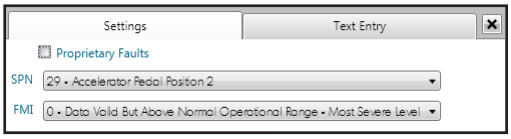


The *Fault* block allows you to use the internal SAE J1939 communication for Diagnostic Messages: DM1, DM2, and DM3. These features allow you to make your own fault indications for active faults. The ECDR-0506A sends a DM1 message once a second. When the *Fault* block is enabled, the Suspect Parameter Number (SPN) and Failure Mode Identifier (FMI) of the block are included in the outgoing DM1 message. When the condition clears, the fault is recorded in fault history. Making a DM2 request, other devices can access this information. The ECDR-0506A responds to a DM2 request and sends any instances of the fault blocks that have been triggered. The ECDR-0506A supports DM3 requests also. Once a DM3 request is received, all fault history is cleared from the ECDR-0506A.

EN Connection

- This enables a fault to be stored.
- Value: 0 – 65535. Zero value is false, non-zero value is true.

Settings



SPN

- This is the Suspect Parameter Number of the fault.
- You can select from a list of defined codes.

FMI

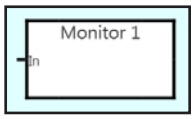
- This is the Failure Mode Identifier of the fault.
- You can select from a list of defined codes.

Note: *Descriptions for SPNs and FMIs, as defined by SAE (as well as any user-defined descriptions), are referenced from the file* **`C:\Program Files (x86)\Hydraforce\HF-Impulse\Modules\J1939Data.xml`**. *If you are comfortable editing the XML, you may add new descriptions to this file. Use caution: if you make any markup errors in this file, you may impact HF-Impulse's ability to display SPN and FMI descriptions.*

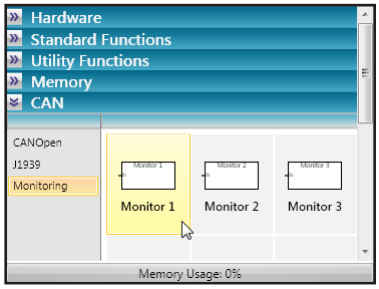
Operation

When the *EN* connection is false, no fault is transmitted or stored. When the EN connection is true, the SPN and FMI, as defined in the block settings, are sent to the active fault control for DM1 transmission. When the EN connection falls to zero, the SPN and FMI are stored to fault history where another device can retrieve them via a DM2 or DM3 request. The ECDR-0506A transmits the fault history in response to a DM2 request. A DM3 request clears the history after transmission.

Monitor Blocks



The *Monitor* blocks are a debugging tool which allows you to monitor specific areas of the diagram. The numbers correspond to the check boxes on the *Monitor* screen (see *Monitor Screen* on page 50). HF-Impulse plots the values over time when monitoring operation of the ECDR-0506A.



In Connection

- This is the value to be monitored.
- Value: 0 – 65535

Settings

This block has no settings.

SERVICE TOOL

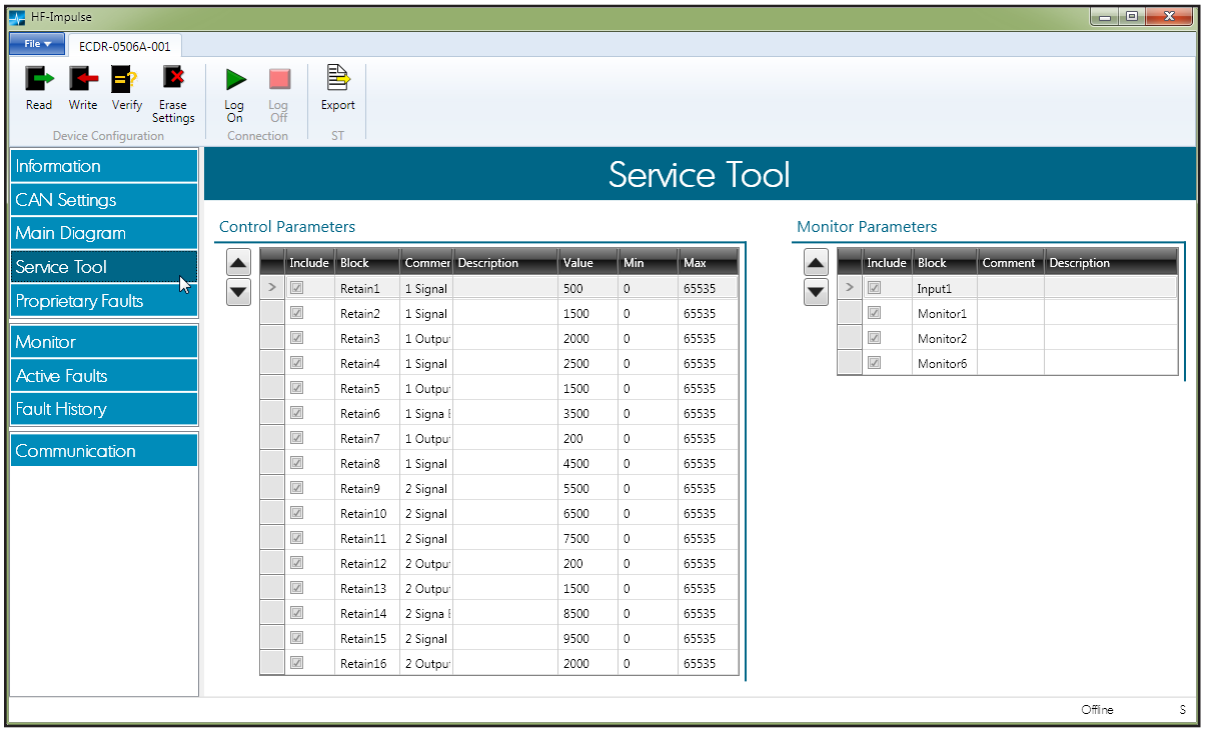
Overview

The *Service Tool* screen allows you to set up parameters that technicians can use to make adjustments to the configuration after deployment. You can make a service tool configuration file (*.icf) that allows access to these parameters while protecting the configuration from being edited. This is useful for tuning machines in a factory setting, or for troubleshooting and repair in the field.

The service tool configuration allows access to these items on the monitor screen:

- Input values
- Output values
- Monitor blocks

On the *Service Tool* screen, technicians have access to alter values stored in *Retain* blocks. You can set *Min* and *Max* limits to restrict the range of these values. You can select which parameters are included in the service tool configuration. The tables display labels (*Settings*|*Text Entry*) from the retain block, and you can also provide helpful descriptions.



Control Parameters

This table displays all the retain blocks from the main diagram.

- Include—check this box to include this parameter in the service tool configuration file.
- Block—the sequence number of the retain block on the diagram
- Comment—the block label as entered into the *Text Entry* tab in the block's settings pane
- Description—enter descriptive reference text for the technician.

Note: *The block sequence number and comment label will not appear in the service tool configuration, therefore it is important to make a descriptive entry here for the technician's reference.*

- Value—the current value of the retain block
- Min—enter the minimum value allowed.

- Max—enter the maximum value allowed.

Note: *The Min/Max values represent a validation range for entry using the service tool configuration. This does not limit the actual value of the block at initialization or during operation.*

- Up/Down arrows—use these arrows to modify the order of the list.

Monitor Parameters

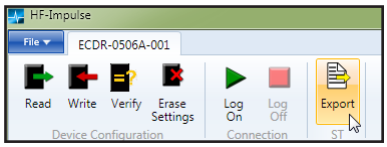
This table displays the *Input* blocks, *Output* blocks, and *Monitor* blocks on the main diagram.

- Include—check this box to include this parameter in the service tool configuration file.
- Block—the sequence number of the block on the diagram
- Comment—the block label as entered into the *Text Entry* tab in the block's settings pane
- Description—enter descriptive reference text for the technician.

Note: *The block sequence number and comment label will not appear in the service tool configuration, therefore it is important to make a descriptive entry here for the technician's reference.*

- Up/Down arrows—use these arrows to modify the order of the list.

Exporting the Service Tool Configuration



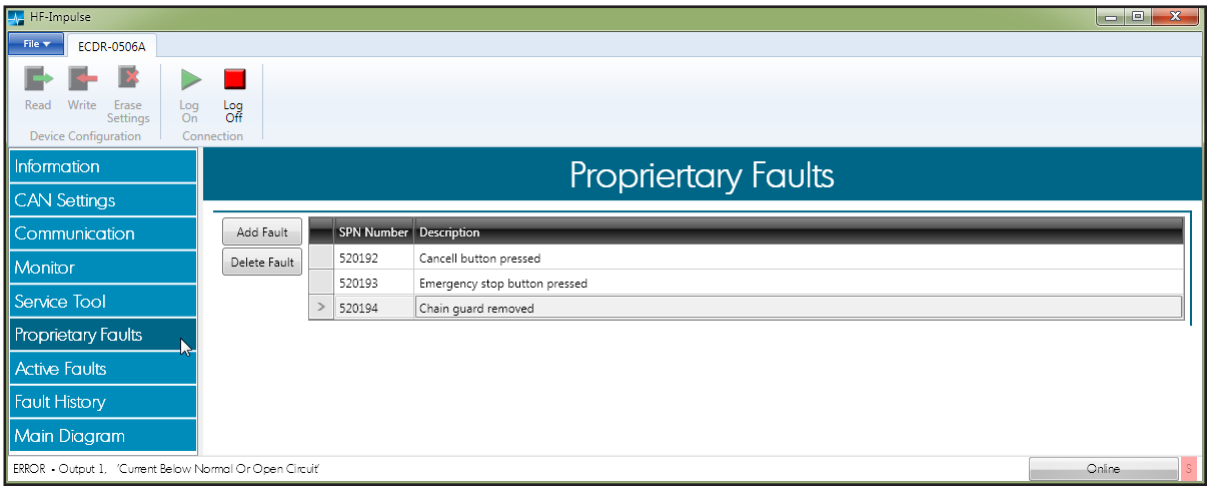
When you are ready to create a service tool configuration file, do the following:

1. Verify all the parameters to be included are checked and they have appropriate descriptions and validation range values.
2. Check the order of the listed parameters and make any necessary adjustments.
3. Click the **`Export`** button.
4. Save the *.icf file to an appropriate location.

PROPRIETARY FAULTS SCREEN

Overview

The *Proprietary Faults* are text descriptions for custom faults. In the *Main Diagram* you can use the *Fault Blocks* to set custom faults. The fault system broadcasts the faults using SAE J1939 DM1 and DM2 messages. SAE J1939 allows proprietary or custom fault messages for Suspect Parameter Number (SPN) values of 520192 through 524287. HF-Impulse has built-in descriptions for standard SAE J1939 SPNs. The proprietary fault descriptions appear in the *Active Faults* and *Fault History* screen when one of the proprietary faults is received from the ECDR-0506A.



Details

SPN

- This is the Suspect Parameter Number of the fault.
- The ECDR-0506A transmits this number on the CAN bus via Diagnostic Message DM1, or when requested by another device via DM2 or DM3.

Description

- This is the description of the fault.
- Fault descriptions are kept for convenience with the configuration. They are not part of the diagnostic messaging features of SAE J1939.

Adding and Deleting Proprietary Faults

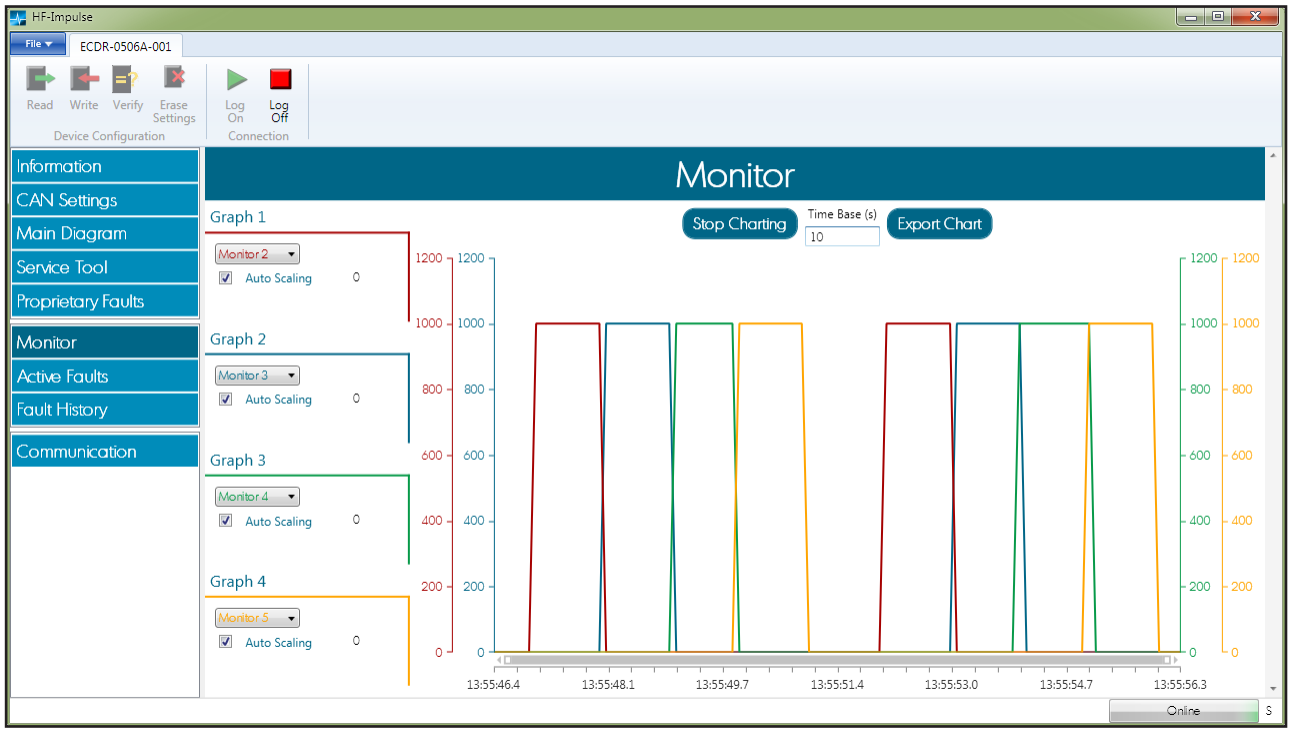
To add a proprietary fault, do this:

1. Click **`Add Fault`**.
2. Type the desired custom *SPN* in the table. Custom SPNs must be in the range of 520192 to 524287.
3. Type a *Description* for the fault.
4. Select a fault and click **`Delete Fault`** to remove it from the list.

MONITOR SCREEN

Overview

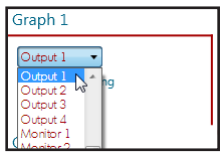
When you *Log On* to an operating ECDR-0506A, HF-Impulse can monitor and log, in real time, the status of outputs 1–4, inputs 1–6, and any of the 20 *Monitor* blocks that you have placed in the diagram. This is helpful when tuning, troubleshooting or testing the ECDR-0506A. You can plot up to 4 data streams at once..



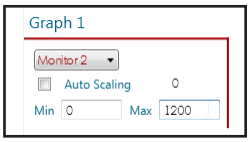
Plotting Data

To plot data during operation, do the following:

1. Connect to your ECDR-0506A (see *Connecting* on page 4).
2. Click **`Read`** to retrieve the program from the connected device, or open a configuration (*.icf) file.



3. Click the drop-down and select a channel to monitor for each graph (up to four).
4. Check **`Auto Scaling`** (default) to allow HF-Impulse to select a vertical scale for the plot. Uncheck this box and enter *Min* and *Max* values to set your own scale.



5. Click **`Log On`** to begin monitoring operation.
6. Enter a value for *Time Base (s)* to change the period of data displayed in the chart area. The default is 10. This is also the length of the data log.
7. Click **`Stop Charting`** to pause monitoring.
8. When you are finished monitoring, click **`Log Off`**.

Exporting Data

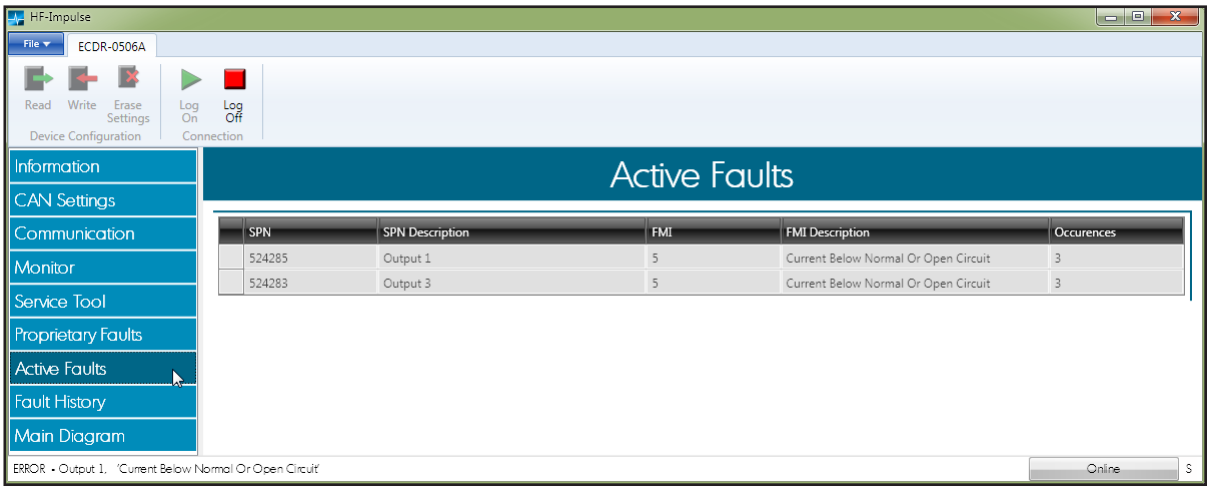
Once you have logged some operating data, you can save it to your hard disk in *.CSV format. Click **`Export Chart`** and browse for a location to save the logged data. *Time Base (s)* determines the length of the recorded data stream.

ACTIVE FAULTS SCREEN

Overview

The fault monitoring system is based on SAE J1939 Diagnostic Messaging protocol (DM1). Each fault contains a Suspect Priority Number (SPN), a Failure Mode Identifier (FMI), and an occurrence count. The *Active Faults* page displays any faults that currently exist. If the ECDR-0506A transmits an empty DM1 packet then the *Active Faults* page displays "No Faults" in the first SPN description column.

If there is no detected SAE J1939 DM1 then the message "No Communication" is displayed in the first SPN description column. The "No Communication" message is also displayed if the SAE J1939 address does not match the address of the ECDR-0506A. The SAE J1939 address is manually set on the *Communication* screen (see *Communication Screen* on page 53).



Details

SPN

- This is the Suspect Parameter Number of the fault.
- The ECDR-0506A transmits this number on the CAN bus via Diagnostic Message DM1, or when requested by another device via DM2 or DM3.

SPN Description

- This is the description of the fault.
- Standard SPN descriptions are referenced from an internal table within HF-Impulse. The SAE J1939 standard describes these.
- Proprietary SPN descriptions are referenced from the configuration file. You can read them on the *Proprietary Faults* screen.

FMI

- This is the Failure Mode Identifier for the displayed fault.

FMI Description

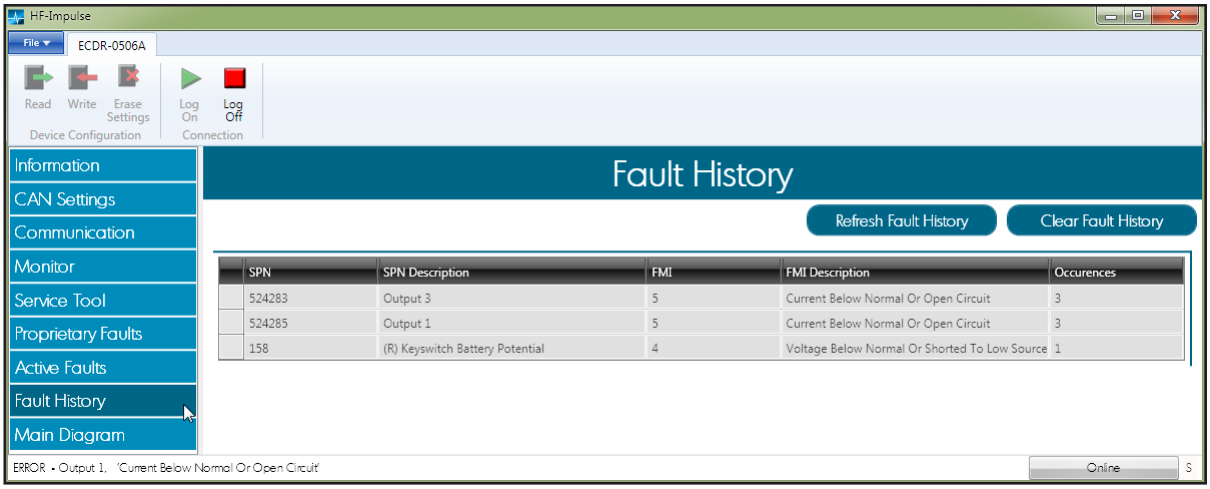
- This is the description of the failure mode.
- Standard FMI descriptions are referenced from an internal table within HF-Impulse. The SAE J1939 standard describes these.
- Proprietary faults do not have an associated FMI.

Occurrences

- This is the number of times the fault has occured since the fault history was last cleared.

FAULT HISTORY SCREEN

On the *Fault History* screen, HF-Impulse uses an SAE J1939 DM2 or DM3 request to poll the fault history from the ECDR-0506A. When *Log On* begins, HF-Impulse automatically sends a DM2 request to the ECDR-0506A for fault history information. You can refresh or clear the fault history using the buttons on this screen. These features only function when HF-Impulse is logged on to the ECDR-0506A.



Details

SPN

- This is the Suspect Parameter Number of the fault.
- The ECDR-0506A transmits this number on the CAN bus via Diagnostic Message DM1, or when requested by another device via DM2 or DM3.

SPN Description

- This is the description of the fault.
- Standard SPN descriptions are referenced from an internal table within HF-Impulse. The SAE J1939 standard describes these.
- Proprietary SPN descriptions are referenced from the configuration file. You can read them on the *Proprietary Faults* screen.

FMI

- This is the Failure Mode Identifier for the displayed fault.

FMI Description

- This is the description of the failure mode.
- Standard FMI descriptions are referenced from an internal table within HF-Impulse. The SAE J1939 standard describes these.
- Proprietary faults do not have an associated FMI.

Occurrences

- This is the number of times the fault has occured since the fault history was last cleared.

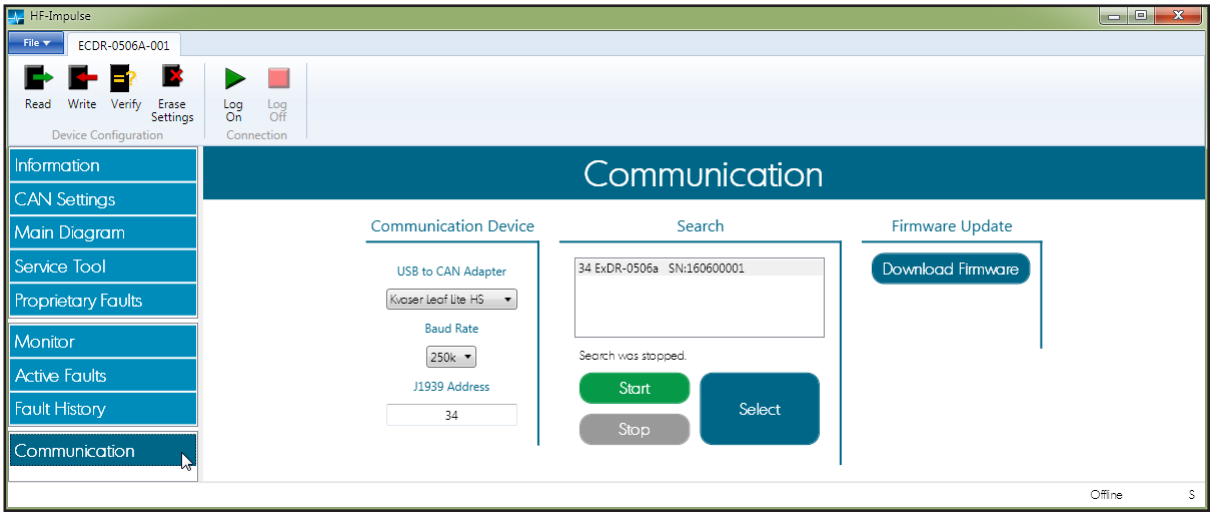
Reading and Clearing the Fault History

1. Click the **`Log On`** button to open a connection to the ECDR-0506A.
2. Click **`Refresh Fault History`** to send a DM2 transmission and update the fault history.
3. Click **`Clear Fault History`** to send a DM3 transmission and update and then clear the fault history.

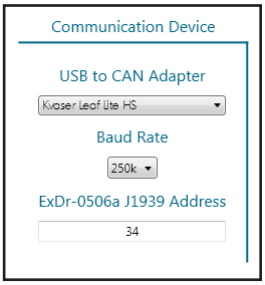
COMMUNICATION SCREEN

Overview

This screen allows you to set up communications with the ECDR-0506A. On the *Communication Screen* you can select the USB to CAN adapter, search for an ECDR-0506A on the network, or download firmware.



Communication Device



Connecting to the Device

HF-Impulse requires a USB/CAN adapter to connect with the ECDR-0506A. The software works with adapters made by Kvaser and Peak. You can purchase the Kvaser adapter from HydraForce, or from another source. You must install drivers for the adapter to operate. These are available from the manufacturer or you can download them from the electronics portal at www.hydraforce.com/electronics.

Note: *A wiring harness is required to connect the ECDR-0506A. The CAN network requires at least one 120 ohm terminating resistor across the CANH and CANL lines. For more information on connecting, including building a proper test harness, refer to the ECDR-0506A Technical Reference.*

USB to CAN Adapter

Select an adapter from the list:

- Kvaser Leaf Lite HS
- PEAK PCAN-USB

Baud Rate

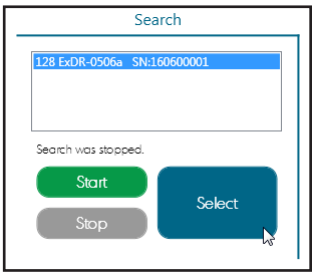
- This is the baud rate for communication.
- Choices: 250 kbps or 500 kbps. Default is 250 kbps

ExDR-0506a J1939 Address

- This is the address of the ECDR-0506A on the SAE J1939 CAN network.
- Enter the address if known.
- This field will be populated automatically when you select a device using *ExDR-0506a Search*.

Note: *These are the settings for communication from HF-Impulse to the ECDR-0506A. To change communications settings on the ECDR-0506A device, go to the CAN Settings screen (see CAN Settings Screen on page 8). If you change these settings, you must write the change to the device. The baud rate and J1939 address settings must match for communication to occur.*

ECDR-0506A Search

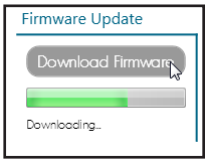


The *Search* function uses the *address claimed request* to look for all SAE J1939 devices on the network. The request is made for all addresses from 1 to 240. All discovered devices are displayed with a model description and serial number. Any other devices discovered on the bus are listed with *unknown* as the model.

To perform a model search:

1. Click **`Start`** to begin searching.
2. Click **`Stop`** at any time to end the search.
3. After HF-Impulse finds your device, click it in the search window and click **`Select`**.

Firmware Update



The ECDR-0506A models are shipped with firmware installed. You can use the *Firmware Update* utility to load the latest firmware or change the model of the hardware. Updating the firmware does not overwrite the program or parameter settings on the device.

To update the firmware:

1. Obtain the latest firmware from the HydraForce electronics portal. Save it to your hard disk.
2. Check the *Communication Device* settings to ensure you are connected to the device.
3. Click **`Download Firmware`**.
4. Browse to the location of the firmware file.
5. Click **`Open`** to begin the download.
6. The progress bar displays the download progress. A success message is displayed when the download is complete.

NOTES

HydraForce Inc.
500 Barclay Blvd.
Lincolnshire, IL 60069
Phone: +1 847 793 2300
Fax: +1 847 793 0086
Member: National Fluid Power Association
ISO 9001

HydraForce Hydraulics Ltd.
Advanced Manufacturing Hub
250 Aston Hall Road
Birmingham B67 FE England
Phone: +44 121 333 1800
Fax: +44 121 333 1810
Member: British Fluid Power Association and
Verband Deutscher Maschinen-und Anlagenbau e.V. (VDMA)
ISO 9001 & ISO 14001

HydraForce Hydraulic
Systems (Changzhou) Co. Ltd.
388 W. Huanghe Road, Building 15A
GDH Changzhau Airport Industrial Park Xinbel District
Changzhou, China 213022
Phone: +86 519 6988 1200
Fax: +86 519 6988 1205
ISO 9001

GLOBAL SALES OFFICES

HydraForce India LLC
Vatika Business Centre
Suite No. 22, Level 5, C Wing
Techpark One, Airport Road
Yerwada, Pune 411006
Maharashtra, India
Tel: +91 020 40111304
Fax: +91 020 40111105
Email: nileshr@hydraforce.com

HydraForce Hydraulics Ltd.
Prager Ring 4-12
D-66482 Zweibrücken, Germany
Tel: +49 (0) 6332 79 2350
Fax: +49 (0) 6332 79 2359
Member: Verband Deutscher Maschinen-und
Anlagenbau e.V. (VDMA)
Email: sales-germany@hydraforce.com

HydraForce Korea LLC
A-506 Bupyeong Woorim Lions Valley, 283
Bupyeong-daero, Bupyeong-gu,
Incheon, Korea 403-911
Tel: +82 32 623 5818
Fax: +82 32 623 5819
Email: jong-seongl@hydraforce.com

7709069 • UM_ECDR-0506A • 05/16 •