ECM ENGINE CONTROL AND MONITORING

# appsCAN Module

**(accelerator pedal position simulator module)**

***and***

# gpioCAN Module

**(general purpose input/output module)**

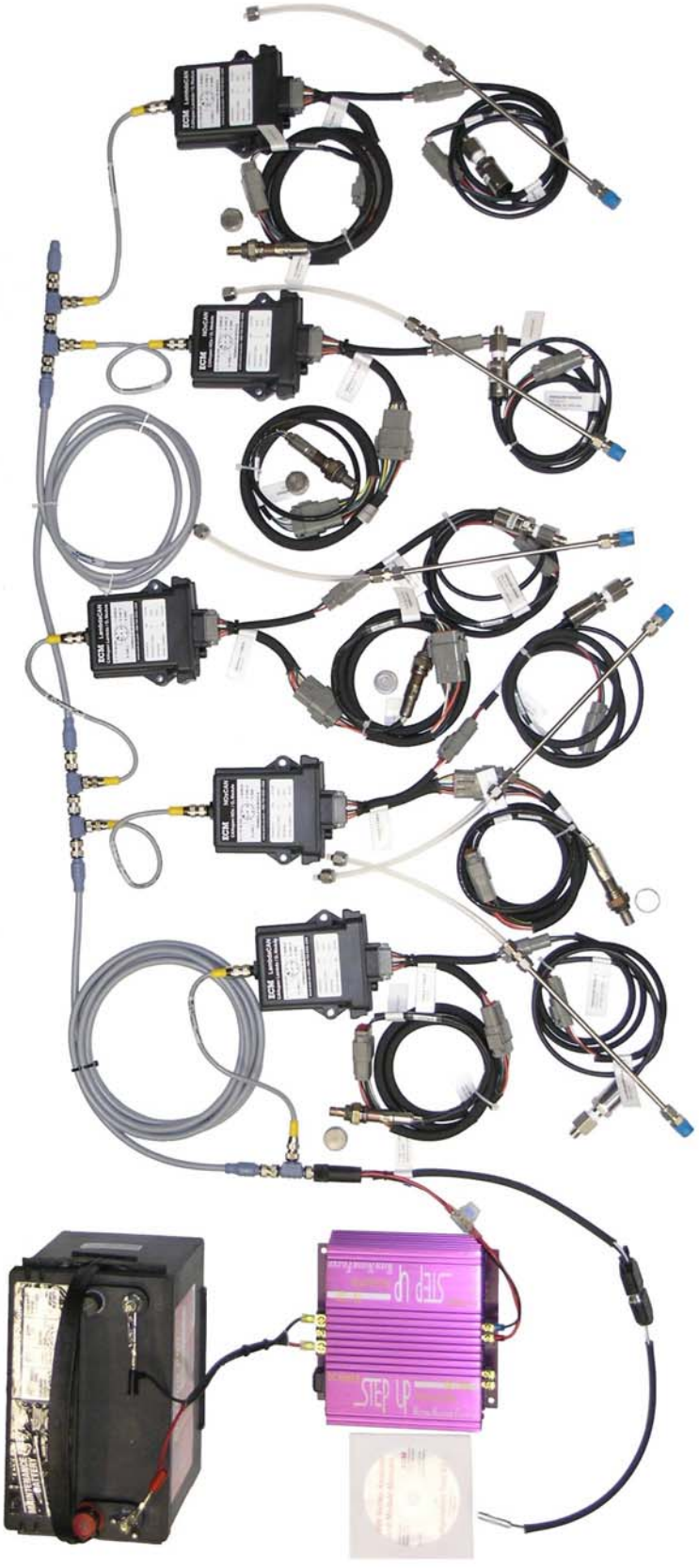
# Instruction Manual

11-22-2009

Information and specifications subject to change without notice.

Printed in the United States of America.

# Table of Contents




Complex measurement and control systems can be easily built with LambdaCAN, NOxCAN, NOxCANg, and appsCAN/gpioCAN modules.

# Introduction

The appsCAN module was originally conceived as a device to simulate the analog and digital signals coming from a drive-by-wire throttle pedal. However, during the product design process it was realized that with a few enhancements, the appsCAN module could be used as a general-purpose control and monitoring device. These enhancements were adopted and the appsCAN module is also sold as gpioCAN. In all but name, the two modules are identical. In this manual, the module will be referred to as the appsCAN module.

## Application as an Accelerator Pedal Position Simulator Module

Most vehicles sold today do not have a physical connection between the accelerator pedal and the engine's throttle. Instead, command signals from the accelerator pedal are sent to the engine controller and the engine's throttle is controlled by an electric motor. When engines or vehicles are tested in dynamometer cells, it is no longer necessary to have a robot or actuator pushing on the accelerator pedal if these signals can be simulated.

The appsCAN module simulates accelerator pedal position signals based on CAN commands sent to it from the dynamometer controller. Up to four analog outputs and four PWM outputs can be simulated. Analog outputs can be absolute (based on an internal voltage reference) or ratiometric (based on external measured reference voltages.

For fail-safe reasons, all accelerator pedals output at least two signals. Coordination of these signals must be maintained to avoid triggering an engine fault code. The appsCAN module supports synchronous signal operation which maintains coordination of these signals.

## Application as General-Purpose Control and Monitoring Device

With four analog inputs, four analog outputs, and four PWM (i.e. low-side driver) outputs, the gpioCAN module can serve as a control and monitoring system. Multiple gpioCAN devices can be linked together to form powerful distributed control and monitoring systems.

# Analog I/O, Digital Outputs, and Other I/O

## Analog I/O

### ◆ Analog Outputs (AO1, AO2, AO3, AO4)

Notes:

1. All analog outputs can be operated in either voltage mode (0 to 5V) or ratiometric mode (programmed as a percentage of an input reference voltage, range 0 to 15V).
2. The programming of the mode of operation and the default (on power up) voltage or ratio is performed in the "Configure Analog Outputs" task of the ECM Configuration Tool [on CD].
3. For example: If analog output AO1 is to be operated in voltage mode, parameter AO1V would be sent to the module (in an RPDO) with the desired voltage value.
4. For example: If analog output AO1 is to be operated in ratiometric mode, AO1 would be programmable as a percentage of whatever input reference voltage VRF1 was (via the parameter AO1%). Both AO1 and VRF1 use GNDa1 as ground. AO1% is sent in an RPDO to the module.
5. The below table gives details of the analog outputs, their voltage references, and their grounds.
6. All voltage outputs are 12 bit, all voltage inputs (references) are 16 bit.
7. All analog outputs and voltage references share a common ground.
8. Max current 10mA/channel. Output Impedance 1Ω.
9. C/T# below means Connector/Terminal Number. The left (gray) terminal is "L", the right (black) is "R". See Figure 1 below for connector and terminal numbering.

| Name | C/T# | Color | **Ref Name** | **Ref C/T#** | **Ref Color** | Gnd Name | Gnd C/T# | Gnd Color |
|---|---|---|---|---|---|---|---|---|
| AO1 | L7 | Orange | VRF1 | L9 | Purple | GNDa1 | L8 | Black |
| AO2 | L10 | Yellow | VRF2 | L12 | Gray | GNDa2 | L11 | Black |
| AO3 | L6 | Green | VRF3 | L4 | White | GNDa3 | L5 | Black |
| AO4 | L3 | Blue | VRF4 | L1 | Tan | GNDa4 | L2 | Black |

**Left (Gray) Connector**

**7 8 9 10 11 12**
**6 5 4 3 2 1**

**Right (Black) Connector**

**1 2 3 4 5 6**
**12 11 10 9 8 7**

Figure 1: Terminal Numbering on Connectors
(Look at Module with Product Label Up)

## ◆ Analog Inputs (VRF1, VRF2, VRF3, VRF4)

Notes:

1. If an input reference voltage is not used for ratiometric mode, it is available as a general-purpose analog input.
2. To be used as a general-purpose analog input, a channel must have its "Enable Ratiometric" box not checked in the "Configure Analog Outputs" task of the Configuration Tool.
3. The below table gives details of the analog inputs and their grounds.
4. Range 0 to 15V, 16 bit.
5. All analog inputs share a common ground.
6. C/T# below means Connector/Terminal Number. The left (gray) terminal is "L", the right (black) is "R". See Figure 1 below for connector and terminal numbering.

| Input Name | Input C/T# | Input Color | **Gnd Name** | **Gnd C/T#** | **Gnd Color** |
|---|---|---|---|---|---|
| VRF1 | L9 | Purple | GNDa1 | L8 | Black |
| VRF2 | L12 | Gray | GNDa2 | L11 | Black |
| VRF3 | L4 | White | GNDa3 | L5 | Black |
| VRF4 | L1 | Tan | GNDa4 | L2 | Black |

**Left (Gray) Connector**

**7 8 9 10 11 12**
**6 5 4 3 2 1**

**Right (Black) Connector**

**1 2 3 4 5 6**
**12 11 10 9 8 7**

Figure 1: Terminal Numbering on Connectors
(Look at Module with Product Label Up)

## Digital Outputs (includes Pulse Output)

### ◆ Digital Outputs (PWM1, PWM2, PMW3, PWM4)

Notes:

1. Digital outputs are programmable in 8 or 16-bit PWM resolution mode. In 8-bit mode, all four PWMs can be programmed. In 16-bit PWM resolution mode, only PWM2 and PWM4 can be programmed.
2. Digital outputs can be programmed (in "Configure PWM Outputs" of Configuration Tool) for:
   i. 8-bit or 16-bit mode. In 8-bit mode worst case resolution: 0.4% Duty Cycle, 4 Hz.
   In 16-bit mode worst case resolution: 0.008% Duty Cycle, 0.1 Hz.
   ii. Pull-up (to 5V), or open-collector output
   iii. Polarity Active Low (i.e. 100% duty cycle activates low-side driver and pulls output to ground)
   iv. Pulse Mode operation of the output. In this mode, a single pulse of programmable width and delay (from clicking on "Start"). Once pulse mode is activated, the delay, width, and triggering of the pulse are programmable in the "Configure & Start Pulse Signal" task in the Configuration Tool.
   v. The default duty cycle and frequency on power up.
3. PWMs are programmable for 0 to 100% Duty Cycle and 1.5 Hz to 1000 Hz
4. Programmable frequency FRQA is for PWM1 and PWM2. FRQB is for PWM3, PWM4.
5. Maximum voltage for low-side driving: 28 VDC
6. Maximum current sinking: 500 mA/channel
7. All digital outputs share a common ground
8. C/T# below means Connector/Terminal Number. The left (gray) terminal is "L", the right (black) is "R". See Figure 1 below for connector and terminal numbering.

| Output Name | Output C/T# | Output Color | **Gnd Name** | **Gnd C/T#** | **Gnd Color** |
|---|---|---|---|---|---|
| PWM1 | R1 | Orange | GNDd1 | R2 | Black |
| PWM2 | R3 | Yellow | GNDd2 | R4 | Black |
| PWM3 | R5 | Green | GNDd3 | R6 | Black |
| PWM4 | R12 | Blue | GNDd4 | R11 | Black |

**Left (Gray) Connector**

**7 8 9 10 11 12**
**6 5 4 3 2 1**

**Right (Black) Connector**

**1 2 3 4 5 6**
**12 11 10 9 8 7**

Figure 1: Terminal Numbering on Connectors
(Look at Module with Product Label Up)

## Other I/O

### ◆ Other I/O (AIN1, 5Vref, VEXC, GNDin)

Notes:

1. AIN1 is an additional 0 to 5V analog input. 16 bits resolution.
2. 5Vref is a 5V reference voltage. 30 mA max.
3. VEXC is an output voltage source (approximately 9.3V) suitable for powering transducers. 100 mA max.
4. AIN1, 5Vref, and VEXC all share GNDin as a ground.
5. C/T# below means Connector/Terminal Number. The left (gray) terminal is "L", the right (black) is "R". See Figure 1 below for connector and terminal numbering.

| Name | C/T# | Wire Color | **Gnd Name** | **Gnd C/T#** | **Gnd Color** |
|---|---|---|---|---|---|
| AIN1 | R10 | Purple | GNDin | R9 | Black |
| 5Vref | R8 | Pink | GNDin | R9 | Black |
| VEXC | R7 | Red | GNDin | R9 | Black |

**Left (Gray) Connector**

**7 8 9 10 11 12**
**6 5 4 3 2 1**

**Right (Black) Connector**

**1 2 3 4 5 6**
**12 11 10 9 8 7**

Figure 1: Terminal Numbering on Connectors
(Look at Module with Product Label Up)

# Programming Outputs and Reading Inputs

There are two ways to program the outputs and to read the inputs of an appsCAN module: via ECM's Configuration Tool and via CAN messages sent by another program.

## Using ECM's Configuration Tool

The Configuration Tool runs on your PC and uses a CAN communication device to communicate with one or more modules. While the tool is being used with modules, just ECM modules set to stand-alone mode (see Appendix B) should be connected to the CAN bus. appsCAN is fixed to operate in stand-alone mode only. This chapter focuses on using the Configuration Tool with the appsCAN module.

The Configuration Tool supports four CAN communication devices: Kvaser, ETAS, Peak USB to CAN adapters, and the VectorCAN CAN adapter card. Driver software for one of these adapters must be installed prior to using the Configuration Tool. This software will be supplied with the adapter or be available on-line. The Configuration Tool is delivered on a CD.

Once the adapter's driver and the Configuration Tool software are installed, and with the appsCAN module(s) powered and connected to the CAN adapter, start the Configuration Tool software. Click on the "Modules" tab, select the CAN adapter, and click on the "START" button.

The software will identify all modules on the bus and display them in the "Module" field. If this does not happen, make sure that the CAN bus is properly terminated (i.e. resistors). Open the Module field to see all the modules on the bus. If a module is not listed, one reason could be that its Node ID is the same as another module. To resolve this, remove all modules except the "missing" one from the CAN bus, STOP then START the software, and change that module's Node ID. Another reason that a module is not listed could be that the module is in EIB mode instead of stand-alone mode. All modules must be in stand-alone mode.

To configure one of the modules (ex. change its Node ID) or to look at that module's data, you have to select that module in the "Module" field.

There are three things you can do with the Configuration Tool:

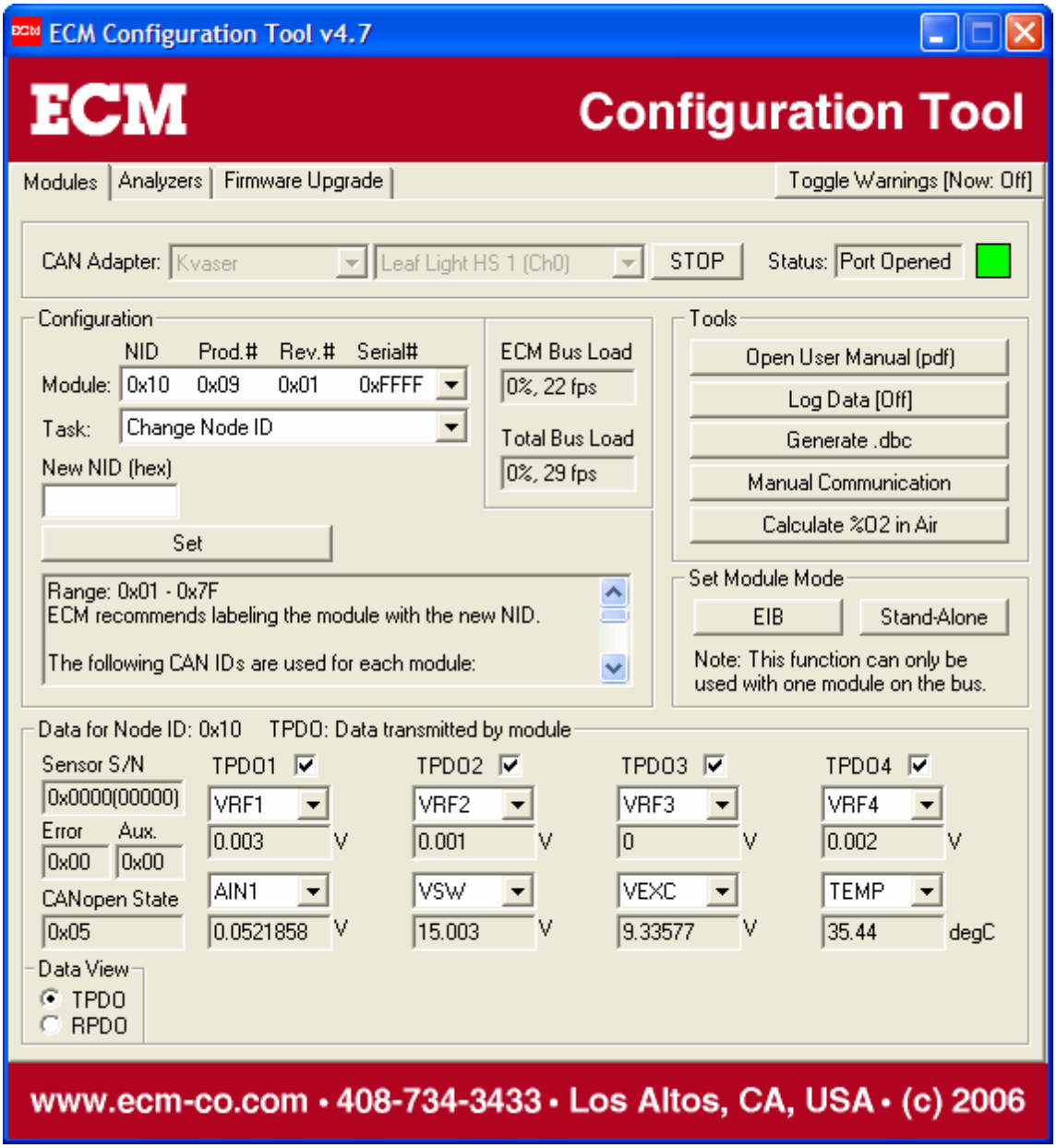
1. Configure a module. This includes programming the module's outputs.
2. Look at data coming from that module in real-time and optionally logging it.
3. Produce a .dbc file to be used by your data acquisition program.

Alternatively, 1. and 2. (above) can be performed by direct CAN communication with the module. For information on how to do this and further detailed information about the appsCAN module, refer to the Appendix E.

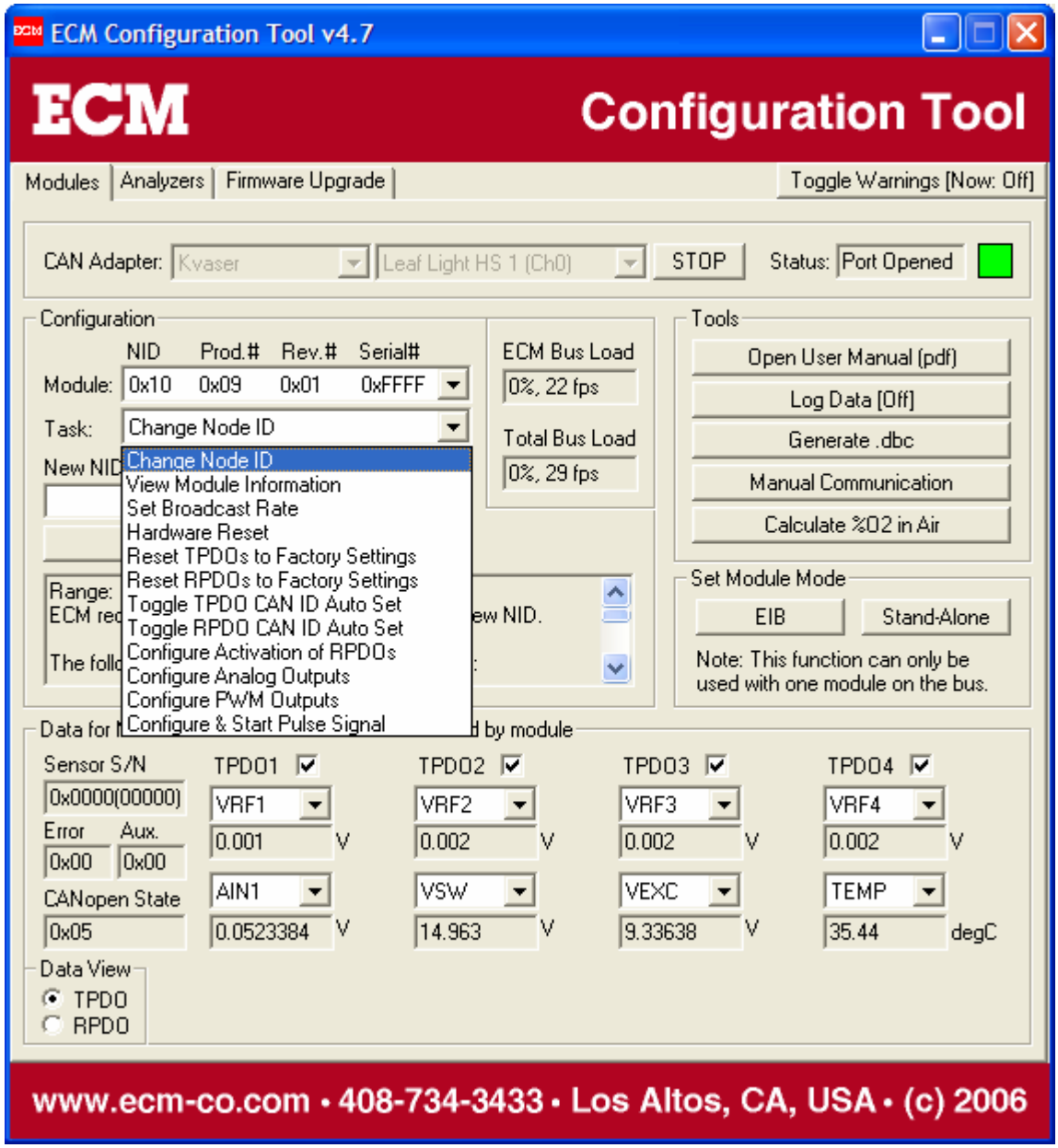
The Configuration Tool for an appsCAN module appears as shown below. On the bottom one-third of the screen either data from the module (TPDOs) or data to be sent to the module (RPDOs) is seen. To toggle between showing TPDOs and RPDOs, click on the "Data View" buttons at the bottom of the screen. Each TPDO or RPDO has two parameters. You can activate 0, 1, 2, 3, or 4 TPDOs and RPDOs by checking the box beside the TPDO or RPDO. Minimize the number of activated TPDOs and RPDOs to minimize the bus load.

TPDOs and RPDOs are selected from the pull-down menus. Once a TPDO is activated, data appears there in real-time. RPDOs are only sent to the module after its "Send" button is pressed and (optionally if the SYNC option is activated) by sending a SYNC signal (as an RPDO).

Table 2 contains the names and meanings of the parameters available with appsCAN.



The Configuration Tool with its Task Menu pulled down appears below. A description of what the tasks do follows.



**Change Node ID:** Allowable range 0x01 to 0x7F (hex). When you assign a Node ID (NID), the following CANs **cannot be used** by any other devices on the bus: 0x00, 0x80 + NID, 0x180 + NID, 0x280 + NID, 0x380 + NID, 0x480 + NID, 0x580 + NID, 0x600 + NID, 0x700 + NID, 0x7E4, 0x7E5.

**View Module Information:** Manufacturer's Name, Hardware Ver., Software Ver., Pinouts

**Set Broadcast Rate:** All activated TPDOs are transmitted every "n" milliseconds. "n" can be programmed. 5 ms is the minimum. Data is sent at a baud rate of 500 kbps. Default: 5 ms.

**Reset TPDOs to Factory Settings:** Selects default input parameters.

**Reset RPDOs to Factory Settings:** Selects default output parameters.

**Toggle TPDO CAN ID Auto Set:** For advanced users, TPDO CAN IDs can be configured to an ID in the range 0x181 – 0x57F. However, they are automatically set to defaults again when the module is power cycled or node ID is changed. If you want the module not to reset the CAN IDs after the module is power cycled, then set to [Off]. Default is [On].

**Toggle RPDO CAN ID Auto Set**: For advanced users, RPDO CAN IDs can be configured to an ID in the range 0x181 – 0x57F. However, they are automatically set to defaults again when the module is power cycled or node ID is changed. If you want the module not to reset the CAN IDs after the module is power cycled, then set to [Off]. Default is [On].

**Configure Activation of RPDOs:** Here the module is configured either to act on the RPDOs immediately after they are sent to the module or upon receiving a SYNC signal after they are sent. The advantage of having a SYNC signal is that you can first load several RPDOs into the module and then have them be activated by the module at the same time. There are two SYNC messages that can be transmitted: One just SYNCs the one module selected. The other SYNC's all the modules on the bus that have been configured to be SYNC'd. To SYNC just one module, send a SYNC message through one of the RPDOs. To SYNC all modules that have been set up to be SYNC'd, click on the "Send Global Sync" button at the bottom of the screen. Default is [When Sent] (i.e. not SYNC'd)

**Configure Analog Outputs:** The analog outputs can be programmed as absolute voltages from 0 to 5V or ratiometrically as a percentage of a 0 to 15V reference input. There are four analog outputs and four input reference voltages. Each analog output has a specific input reference voltage assigned to it. For example, analog output 1 (AO1) has VRF1. In ratiometric mode, if VRF1 is at 8.2V and AO1% is programmed at 40%, AO1 will be 3.28V (i.e. 8.2 x 0.4). If ratiometric mode is not enabled, the voltages of the analog outputs are programmed as absolute voltage values via the RPDO parameters AO1V,…AO4V. When the module is first powered up and has not received commands for the analog outputs, there are default values that the four analog outputs are set to. These are programmable in the "Configure Analog Outputs" task.

**Configure PWM Outputs:** PWM resolution can be chosen as 8-bit or 16-bit. As 8-bit, all four PWMs can be controlled. As 16-bit, only PWM2 and PWM4 can be controlled. PWMs are programmable for pull-up (to 5V) (or low-side driver), polarity, pulse mode, and default (on power up) duty cycle and frequency. For information on pulse mode, see "Configure & Start Pulse Signal".

**Configure & Start Pulse Signal:** Each PWM output can be programmed to output a single pulse signal. The pulse signal has a programmable delay (after pressing "Start" button) and pulse width. Before using this feature, the PWM output must be set to "Pulse Mode" in "Configure PWM Outputs".

Table 1: Task List for appsCAN Module

# Data Sent to (RPDO) and from (TPDO) Module

Data sent to (i.e. commands) appsCAN modules is packaged as RPDOs (Receive Process Data Object). Data sent from appsCAN modules is packaged as TPDOs (Transmit Process Data Object). Each RPDO and TPDO contains two parameters and each module can receive up to four RPDOs and send up to four TPDOs. All selected TPDOs will be sent at the broadcast rate. For example, if the broadcast rate is 5 ms and four TPDOs were selected to be sent, then eight pieces of data would be transmitted every 5 ms. To avoid slowing down the effective data rate on the CAN bus, select the number of TPDOs to be sent and the broadcast rate sparingly. For the case of multiple modules sending multiple TPDOs on the same CAN bus, the minimum (i.e. fastest) broadcast rate is given by:

**Minimum Broadcast rate (ms) = The total number of TPDOs for all modules x 0.3125**

For example, if there are eight modules, each sending two TPDOs, the minimum broadcast rate is 5 ms.

RPDO data is sent to the module when the "Send" button is pressed. However, if the "Activation of RPDOs" is set up to be in SYNC mode, a SYNC command must be sent to the module for it to act on the RPDOs sent.

The data sent or received from appsCAN modules is selected in the "Data" area of the Configuration Tool. Click in the "Data View" area to select TPDOs or RPDOs. Activate the number of TPDOs and RPDOs by clicking in its box to put in a check mark. Select the data contained in each TPDO and RPDO using the pull-down windows. The list of available parameters for the appsCAN module is given in Table 2.

| Parameter Name Displayed | Full Parameter Name | Parameter Description |
|---|---|---|
| VSW | Vsw (V) | Supply voltage measured at the module |
| TEMP | Circuit Board Temp (°C) | Temperature of the module circuit board |
| ERFL | Error bit flags (bits) | Module error flags (unsigned long format) |
| ERCd | ECM CANOpen Error Code | ECM CANOpen Error Code |
| VRF1 | Voltage Reference 1 (V) | Voltage measured at reference input 1 |
| VRF2 | Voltage Reference 2 (V) | Voltage measured at reference input 2 |
| VRF3 | Voltage Reference 3 (V) | Voltage measured at reference input 3 |
| VRF4 | Voltage Reference 4 (V) | Voltage measured at reference input 4 |
| AIN1 | Analog Input 1 (V) | Voltage measured at analog input 1 |
| VEXC | Voltage Excitation (V) | Voltage supplied for sensor excitation |
| PWM1 | PWM Duty Cycle for Ch 1 (%) | Commanded Duty Cycle on PWM channel 1 |
| PWM2 | PWM Duty Cycle for Ch 2 (%) | Commanded Duty Cycle on PWM channel 2 |
| PWM3 | PWM Duty Cycle for Ch 3 (%) | Commanded Duty Cycle on PWM channel 3 |
| PWM4 | PWM Duty Cycle for Ch 4 (%) | Commanded Duty Cycle on PWM channel 4 |
| FRQA | Frequency A (Hz) | Commanded PWM frequency for PWM1 and PWM2 |
| FRQB | Frequency B (Hz) | Commanded PWM frequency for PWM3 and PWM4 |
| AO1V | Analog Output 1 (V) | Analog output commanded to channel 1 |
| AO2V | Analog Output 2 (V) | Analog output commanded to channel 2 |
| AO3V | Analog Output 3 (V) | Analog output commanded to channel 3 |
| AO4V | Analog Output 4 (V) | Analog output commanded to channel 4 |
| AO1% | Analog Output 1 (%) | % of VRF1 commanded to analog output channel 1 |
| AO2% | Analog Output 2 (%) | % of VRF2 commanded to analog output channel 2 |
| AO3% | Analog Output 3 (%) | % of VRF3 commanded to analog output channel 3 |
| AO4% | Analog Output 4 (%) | % of VRF4 commanded to analog output channel 4 |
| SYNC | Synchronize | Use to synchronize application of RPDOs in module |
| NULL | Null | Dummy filler parameter “Nothing” |

Table 2: appsCAN Parameter List

# Producing a .dbc File

A .dbc file describes to the device communicating from one or more appsCAN what is in the data packages. For each module, the packages will contain data for the parameters selected in the activated TPDOs, RPDOs, and an error code. The Configuration Software has a tool called "Generate .dbc…" that will generate a .dbc file for all the modules on a CAN bus. Make sure that each module is configured as desired and that all modules are on the bus before the "Generate .dbc…" button is pushed. Data package information from all the modules is stored in the one .dbc file produced.

Programs importing the .dbc file and applying it to the CAN data transmitted by the modules will see data, etc identified as follows:

Data: **name_nid[units]**

where: name = parameter name. See Table 2.
nid = node id of module in hex
units = units of parameter

for example: VRF1_0X01[%] which is the Voltage Reference #1 voltage measured by module with nid 0X01

Error code: **ECM_Error_Code_nid**

where nid = node id of module hex
error code is in hex and given in Table 3

for example: ECM_Error_Code_0x11

Auxiliary: **ECM_Auxiliary _time[sec]**

where: time = decrementing countdown to module activation in hex

for example: ECM_Auxiliary_0X12[sec]

| ECM ERROR CODE | LED ACTION | DESCRIPTION OF ERRORS |
|---|---|---|
| 0x0000 | Grn ON | All OK, (green led constantly on) |
| 0x0002 | Grn/Both/Red 2s | Power on reset/ Init hardware |
| | | |
| 0x00A1 | N/A | Invalid software state |
| 0x00B1 | N/A | CAN overrun |
| 0x00B2 | N/A | CAN passive mode |
| 0x00B3 | N/A | CAN heartbeat error |
| 0x00B4 | N/A | CAN recover bus off |
| 0x00B5 | N/A | CAN Tx CanId collision |
| 0x00B6 | N/A | Serial overrun |
| 0x00B7 | N/A | CAN overrun Lss |
| 0x00B8 | N/A | CAN overrun Sdo |
| 0x00B9 | N/A | CAN overrun Rx |
| 0x00BA | N/A | CAN overrun ECT5 |
| 0x00FF | Both ON | Module powering down within 500ms |

Table 3: appsCAN Error Codes List

# Using the dashCAN Display

The dashCAN display (see cover and below) is a small (105 mm x 63 mm x 63 mm), two-channel remote display for CAN networks containing appsCAN, LambdaCAN and NOxCAN(g) modules. dashCAN comes with a two meter cable and a "T" (P/N 09-05). Simply attach dashCAN to the CAN bus and any two parameters being transmitted from modules can be displayed. dashCAN can display parameters from the same module or two different modules. Multiple dashCAN displays can be attached to the CAN bus.

dashCAN has two modes of operation: RUN (when measurements are displayed) and SYS (where dashCAN is set-up). The SYS key toggles between the modes.

While in RUN mode:

i. If the ↑ button is pressed, the displays will show the serial numbers of the modules assigned to the displays.
ii. If the ↓ button is pressed, the displays will show the parameter names assigned to the displays. See Table 2.
iii. If the ENT button is pressed, the displays will show the units of the parameters. "PCTG" is %. "dIM" means dimensionless (ex. for AFR, FAR, PHI, Lambda).

In RUN mode, four things other than data can be displayed:

i. "ERR" and "####" where "###" is an error code. See Table 4.
ii. "…." which means that a module has not been assigned to that display.
iii. "----" which means that dashCAN has an internal problem.
iv. "XXXX" which means that dashCAN is not receiving any data from the module assigned to that display.

When first entering SYS mode, either "MOd" will be on the upper display or "LOCK" will be on the lower display. If "MOd" is displayed, the ↑ and ↓ keys will roll through the setup options (see Table 5). First the options for the upper channel are shown on the upper display, followed by identical options for the lower channel on the lower display, ending with the global CONF (Configuration) setup. Pressing the ENT key will select the displayed setup option and allow its programming.

If "LOCK" is displayed, the dashCAN has been locked and its setup cannot be changed until it is unlocked. Appendix D describes how to LOCK and unlock dashCAN.

| Setup Option | Level 1 | Function |
|---|---|---|
| MOd | | Select module s/n. Default is NONE. |
| RATE | | Set parameter averaging rate. Range 0.001 to 1.000 Default is 1.000 |
| dISP | | Select parameter. Note: Parameters available are those programmed using Configuration Software. |
| CONF | LEdS<br>LOCK | Set display intensity. Default is 3333.<br>Lock and Unlock Display for Programming |

MOd, RATE, and dISP appear on the upper display for the upper channel and on the lower display for the lower channel. CONF just appears on the lower display and is for global dashCAN setup. All entries must be followed by pressing the ENT key.

Table 4: Menu Tree for dashCAN

## MOd (Module) Setup Option

In MOd setup, the serial number of the module assigned to the upper or lower channel is entered. The serial number is written on a label on the module. The module assigned to the upper channel will send information to the upper display and the module assigned to the lower channel will send information to the lower display. The same module can be assigned to both channels or different modules can be assigned to each channel.

After entering MOd (i.e. press ENT when "MOd" is displayed), the serial numbers of the available modules will be displayed. Select using ↑ and ↓ followed by the ENT key.

## RATE Setup Option

Data is transmitted from modules at the broadcast rate and the programmed averaging that was programmed using the Configuration Software. This transmitted data can then be further averaged before being displayed on the displays. Separate averaging can be programmed for the upper display and the lower display.

The averaging is programmed with values from 0.001 (heavy averaging) to 1.000 (no averaging). The default is 1.000. The averaging is performed as follows:

$$\mathbf{DisplayedValue_{t+1} = \alpha \times Parameter_{t+1} + (1 - \alpha) \times DisplayedValue_t}$$

where:

$DisplayedValue_{t+1}$ = the new displayed value

$\alpha$ = The user-programmable averaging.
Range: 0.001 (heavy averaging) to 1.000 (no averaging).

$Parameter_{t+1}$ = the latest value transmitted by the module

$DisplayedValue_t$ = the previous displayed value

The selected display averaging does not affect the module's CAN transmission rate or averaging.

## dISP (Display) Setup Option

In dISP setup, the parameters to be displayed are selected. Only parameters selected to be transmitted by the Configuration Software can be displayed.

Here is an example of setting the parameter to be displayed on the upper display:

1. Press the SYS key until "MOd" is displayed.
2. Press the ↓ key until "dISP" is on the top display. Then press the ENT key.
3. Press the ↓ key until desired parameter name is displayed. See Tables 2 and 3. Then press the ENT key.
4. Press SYS to return to RUN mode.

## CONF (Configure) Setup Option

CONF setup appears at the end of the setup list on the lower display. To enter CONF, press the SYS key until "MOd" appears on the upper display, press the ↓ key until "CONF" appears on the bottom display, and then press the ENT key. CONF is for global dashCAN setup.

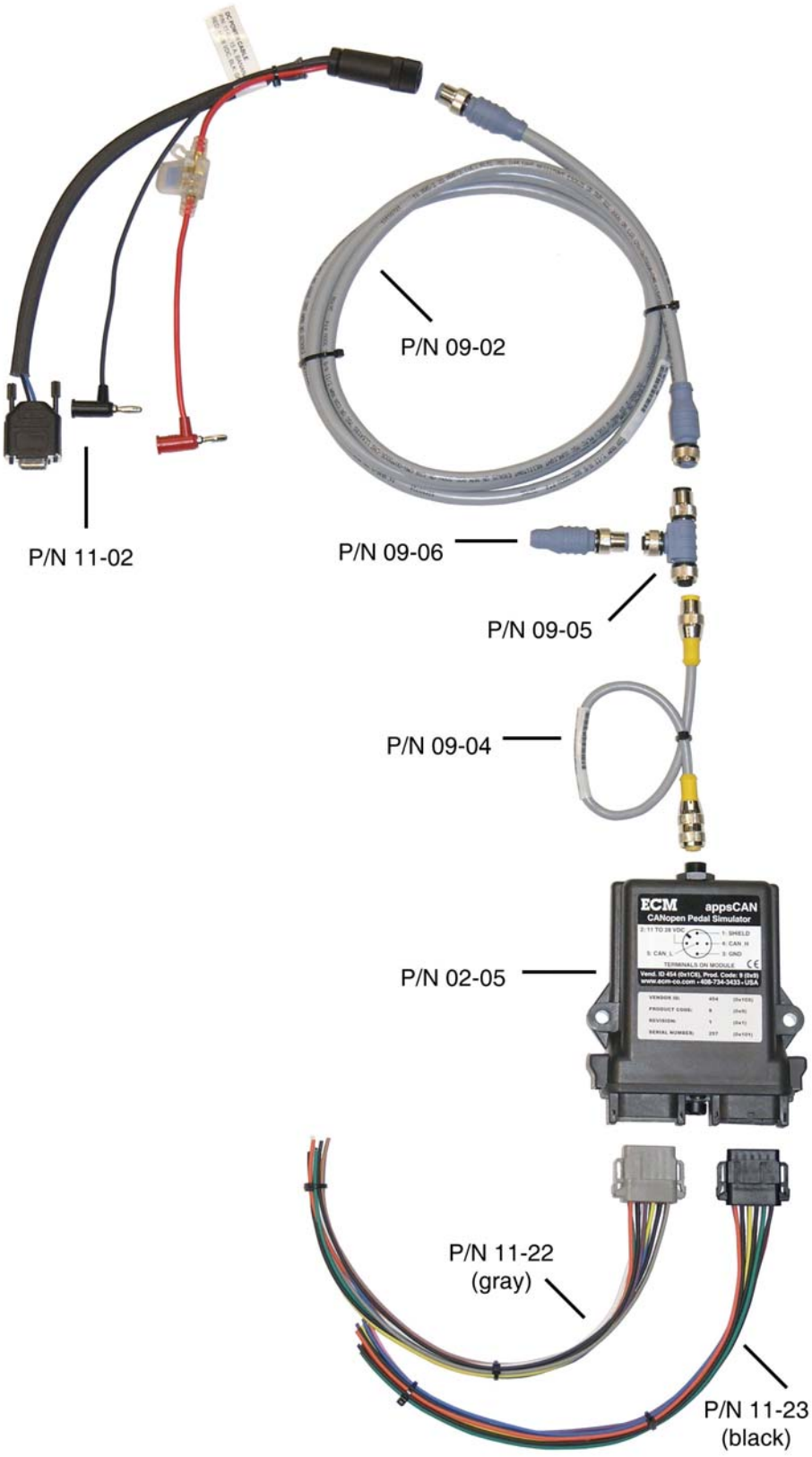
### ◆ LEdS

The display intensity is programmable. Press the ENT key when "LEdS" appears on the lower display, press the ↑ or ↓ keys until the display intensity is suitable, press ENT, and press SYS to return to RUN mode.

### ◆ LOCK

"LOCK" locks the MOd, RATE, dISP, and LEdS setup. This stops unauthorized modification of the display. Refer to Appendix E for more information.

# Appendix A: appsCAN Kit Contents



The appsCAN Kit consists of:

| | Description | P/N | Quantity |
|---|---|---|---|
| 1. | appsCAN Control Module | 02-05 | 1 |
| 2. | Left (gray) connector with 300mm pigtails | 11-22 | 1 |
| 3. | Right (black) connector with 300mm pigtails | 11-23 | 1 |
| 4. | Flexi-Eurofast Cable | 09-04 (0.3m) | 1 |
| 5. | Eurofast "T" | 09-05 | 1 |
| 6. | Eurofast Terminating Resistor | 09-06 | 1 |
| 7. | 2m Eurofast 12mm Cable | 09-02 | 1 |
| 8. | DC Power Cable, DB9F, Banana | 11-02 | 1 |
| 9. | appsCAN Manual and Configuration software CD | 13-01 | 1 |

Optional Cable Components:

| | | | |
|---|---|---|---|
| 1. | Connector Kit<br>2 connectors, 2 locks, 24 terminals, 12 plugs | 11-24 | 1 |

Note: Left (gray) connector is Deutsch P/N DTM06-12SA
Right (black) connector is Deutsch P/N DTM06-12SB
Lock for connectors is Deutsch P/N WM12S (2 required)
Terminals are Deutsch P/N 0462-201-2031
Plugs are Deutsch P/N 0413-204-2005
Crimper is Deutsch P/N HDT-48-00

All connectors, terminals, plugs, and crimper are available from Deutsch, Inc.
In the U.S.A., Deutsch products available from Ladd 1-800-223-1236.

Optional Power Supply:

| | | | |
|---|---|---|---|
| 1. | AC/DC Power Supply, Universal 24VDC @ 4.2A<br>(requires P/N 11-17 Deutsch DTM3M to DB9F) | 04-01 | 1 |

Optional CAN Adapter:

| | | | |
|---|---|---|---|
| 1. | Kvaser Leaf Light, USB to CAN Adapter | 13-02 | 1 |

# Appendix B: Module Stand-alone Mode and EIB Mode

CAN data from LambdaCAN and NOxCAN(g) modules can either be taken directly from the modules themselves or from the CAN port of display heads connected to the modules. When data is taken directly from one or more modules, each module must in Stand-alone mode. When data is taken from one or more display heads of an EGR 5210, Lambda 5220, or EGR 5230 analyzer, each module must be in EIB mode.

Therefore, the module must be properly configured in Stand-alone mode or EIB mode depending on how it will be used. When LambdaCAN and NOxCAN(g) modules are sold alone, they are delivered in Stand-alone mode. When LambdaCAN and NOxCAN(g) modules are sold as part of a NOx 5210, Lambda 5220, or EGR 5230 analyzer, they are delivered in EIB mode.

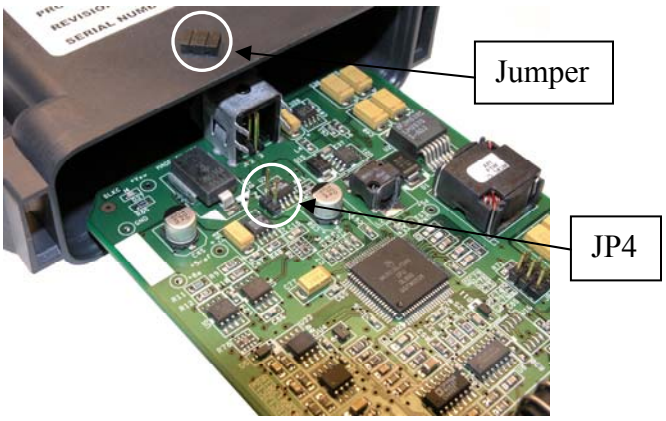
To convert from one mode to the other requires software reprogramming of the module followed by the installation (to set to Stand-alone) or removal (to set to EIB) of a jumper inside the module.

◆ **To convert a module from EIB to Stand-alone Mode**

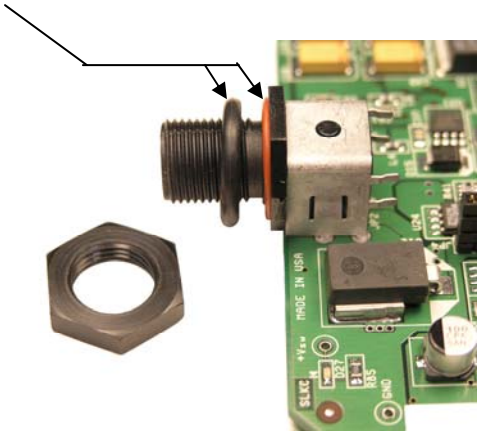
1. Take the nut off the end of the module. Use an 18mm socket without the wrench.


2. Release the two tangs at each side of the module.

3. Slide the PCB out. Install a jumper on JP4.




4. Make sure both O-rings are on the threaded connector.



5. Slide the PCB into the enclosure until the two tangs “click”.

6. Put the nut on and tighten ONLY ½ turn from where it is seated. If this nut is tightened too much, the connector will crack and the enclosure will not be sealed.

7. Connect the module to a power supply and a PC (via a CAN communication adapter) using the cabling shown. A sensor does not have to be connected to the module. Note that only one module is connected and a display head is not involved.

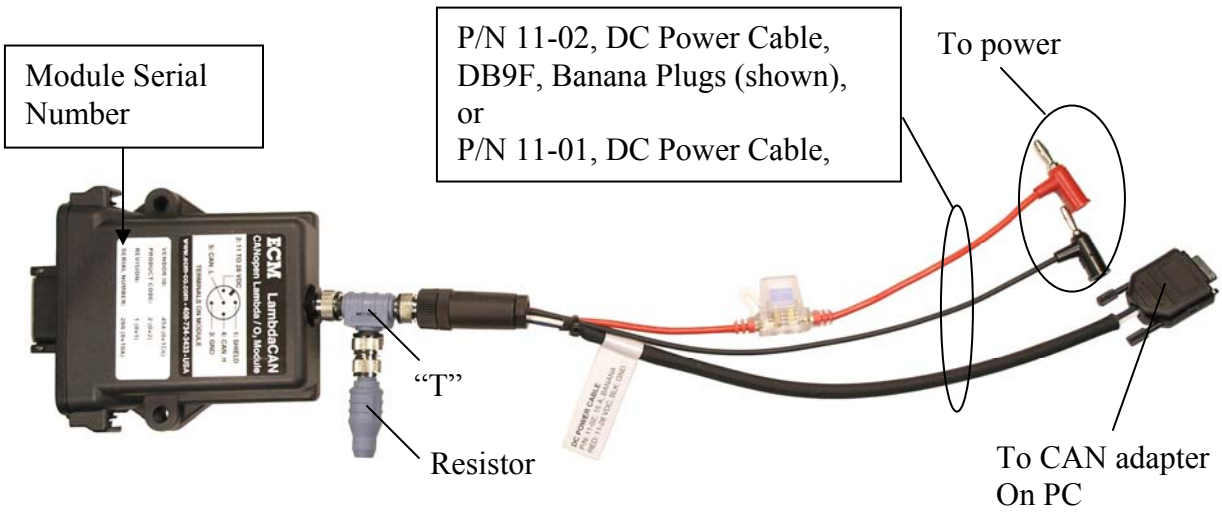
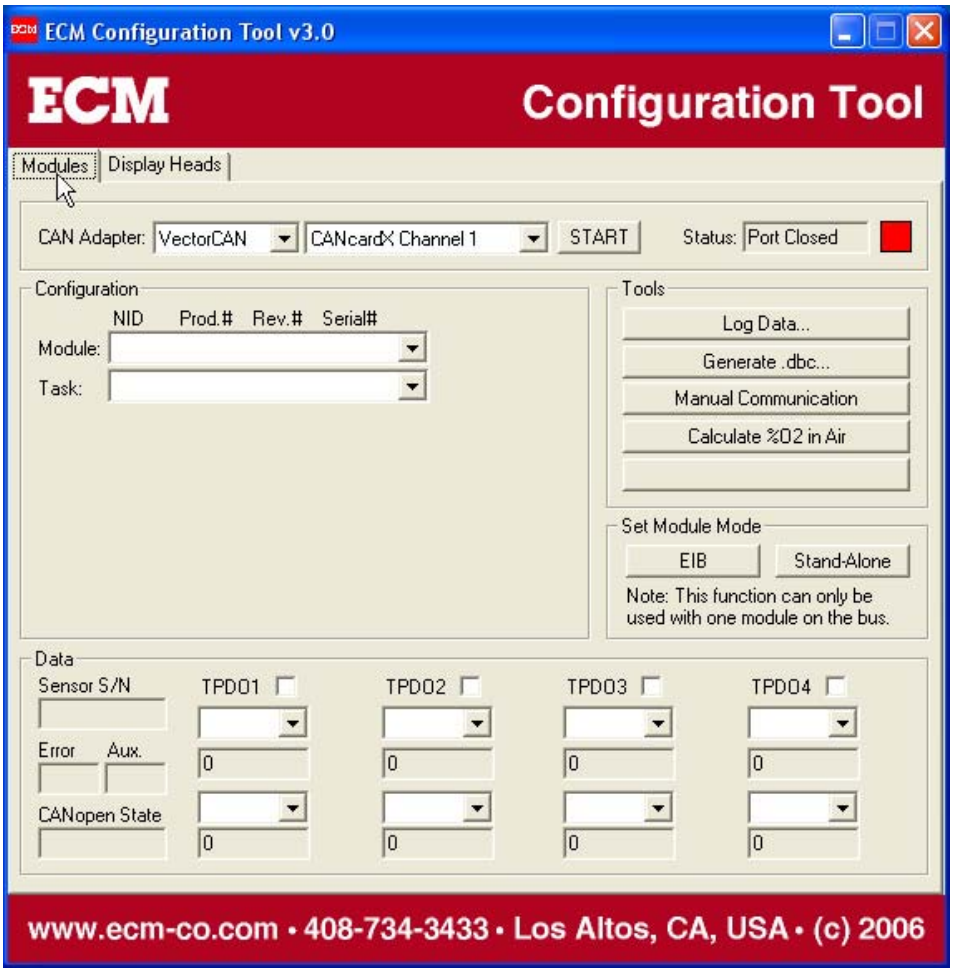



Figure A1: Module prepared for Reprogramming

8. Start the Configuration Tool (software). Click on the "Module" tab. Select the CAN adapter being used. Then start the communication.



9. Click on the “Set to Stand-Alone Mode”. Wait for “Done” Message.
   Stop communication and exit program. The module is in Stand-alone mode.

◆ **To convert a module from Stand-alone Mode to EIB Mode**

1. Use the Configuration Tool (software) to “Set to EIB Mode”.
2. Remove the jumper on JP4 in the lambda module.

# Appendix C: Setting Up ETAS INCA for ECM Modules

**Hardware Setup: Using ETAS ES591.1**

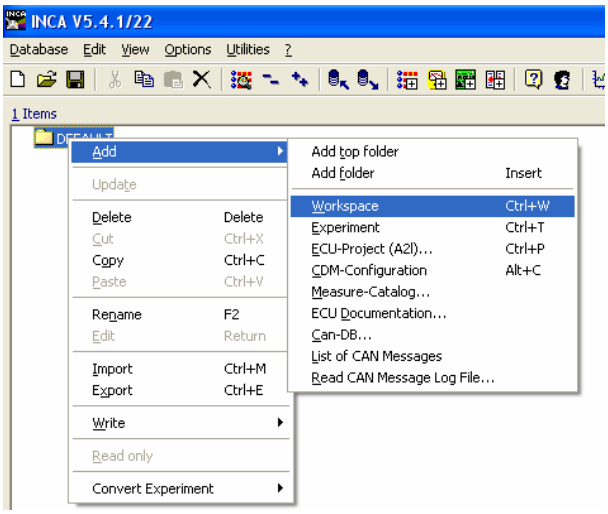
1. Connect the power port to a power source between 6V and 32V.
2. Connect the Ethernet port directly to the Ethernet port on your PC. This port does not use an internet/intranet connection like a router.
3. Connect either the CAN1 or CAN2 port to a CAN network (i.e. ECM modules or display heads).

**Software Setup: Using ETAS INCA V5.4.1, Hotfix 22, GM Install**

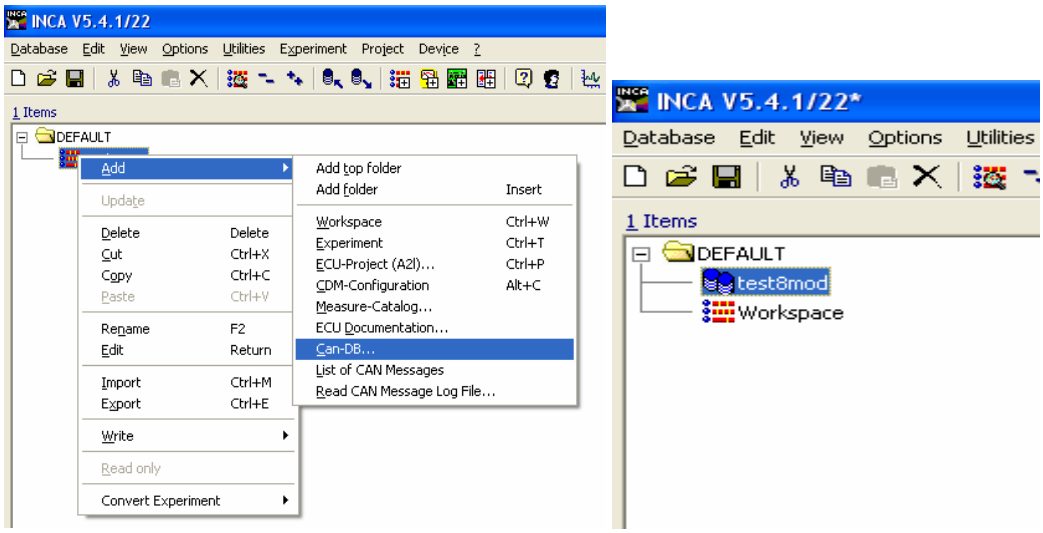
1. Double click the INCA V5.4 icon to open the software.

2. **Create a new Database**. In the Database menu, select New. Give your database a name (i.e. a folder name). In INCA, a Database means the current working directory. Each project is created in a unique directory. When INCA is opened, it will default to the last Database that was used.

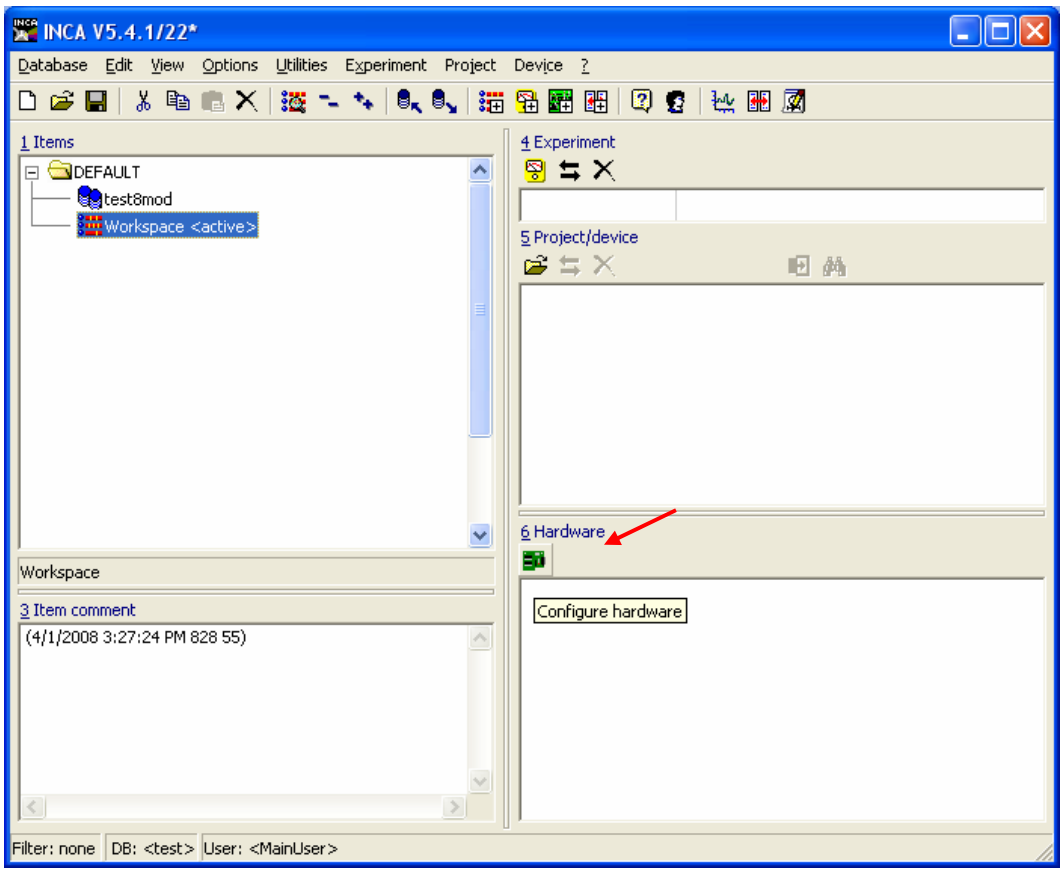
3. **Add a new Workspace**. Right click on the "DEFAULT" folder icon, select Add > Workspace. You can rename it to whatever you want.



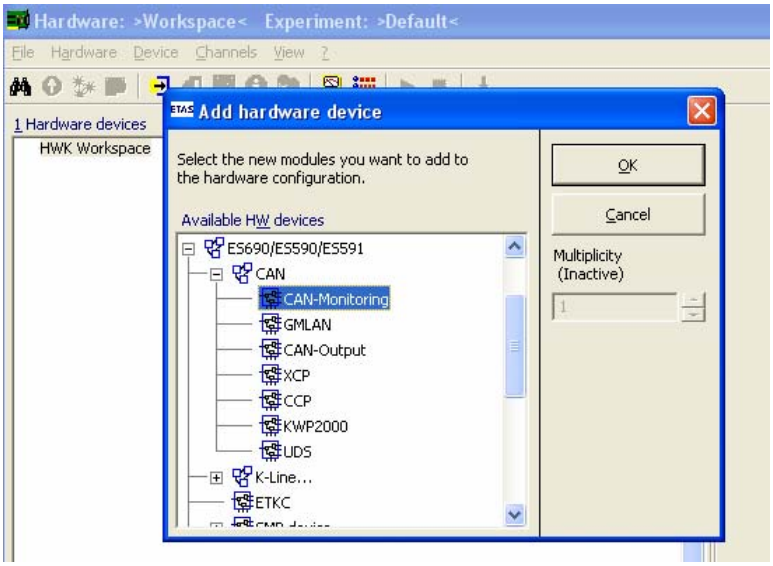

4. **Add a new dbc file for your project.** Right click on the workspace you created in step 3, select Add > Can-DB. Browse to your dbc file and click open. In this example, we are using a file named test8mod.dbc. An INCA log window will pop up. You can ignore this.



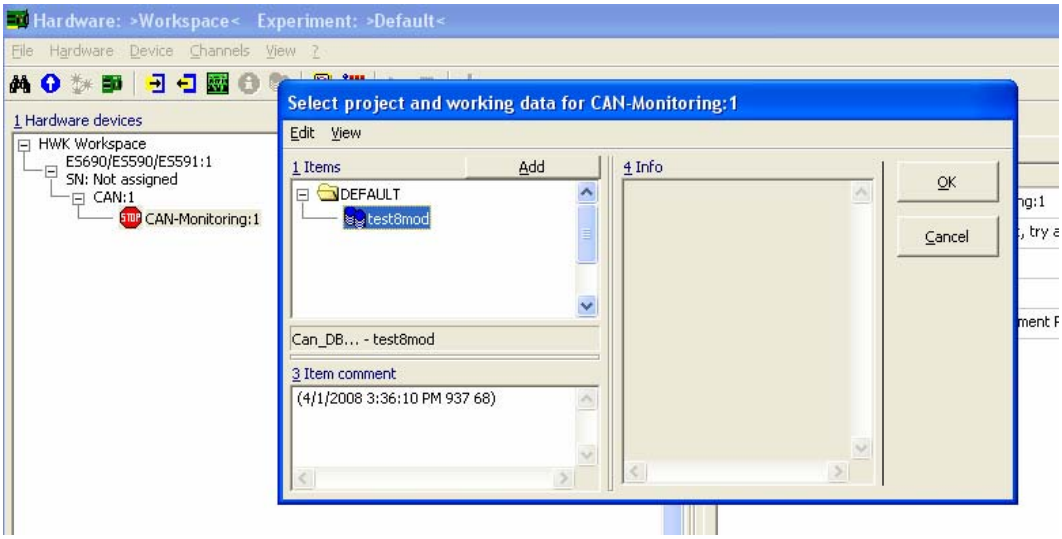

5. **Configure the hardware.** Click on the icon for the workspace you created in step 3. Open the Hardware Configuration icon under the section text "6. Hardware". A hardware configuration window will open.



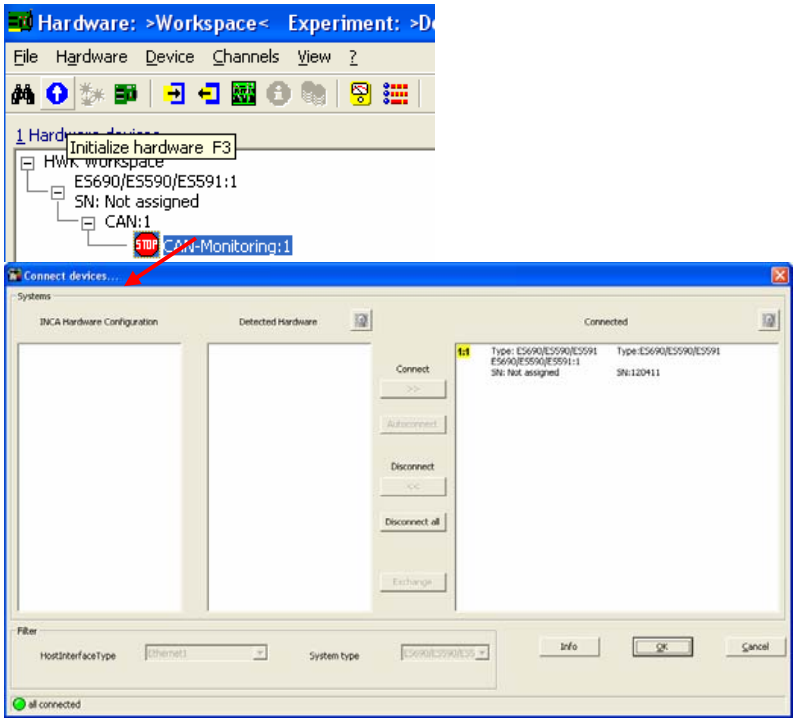
6. **Select the hardware.** In the hardware configuration window, right click the "HWK Workspace" listed under the section text "1. Hardware Devices", and select Insert. Select the ETAS device you wish to use. In this example, we are using an ETAS ES591.1. Expand the selection tree by clicking the "+" next to the hardware device model. Expand the CAN selection and select CAN-Monitoring. Click OK.



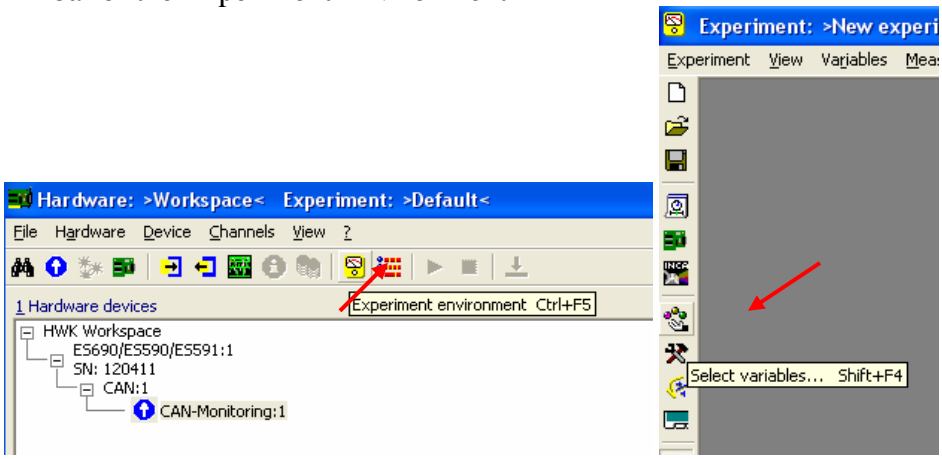

7. **Associate the dbc.** When you clicked OK in the last step, another window will pop up that will allow you to select a dbc that you have added to your workspace from step 4. Expand the selection tree, select your dbc file, and click OK.



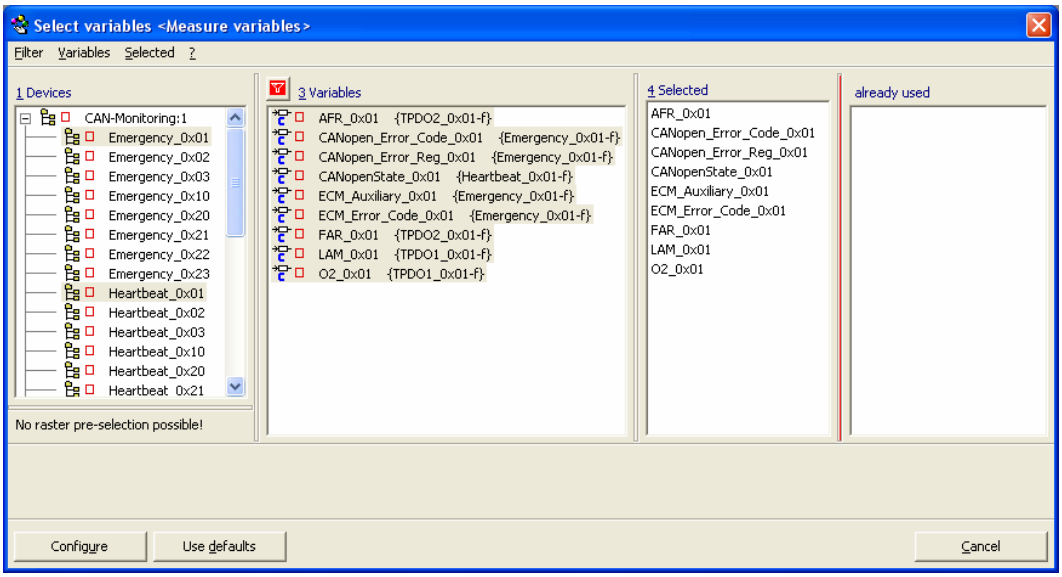
8. **Initialize hardware.** The hardware is currently stopped, as indicated by the red stop sign icon next to the selected hardware. You must initialize it before you can use it to collect data. Click on the Initialize Hardware button on the upper tool bar and wait for the hardware to complete its initialization. Another window will pop up to confirm the device to connect to. Click OK.



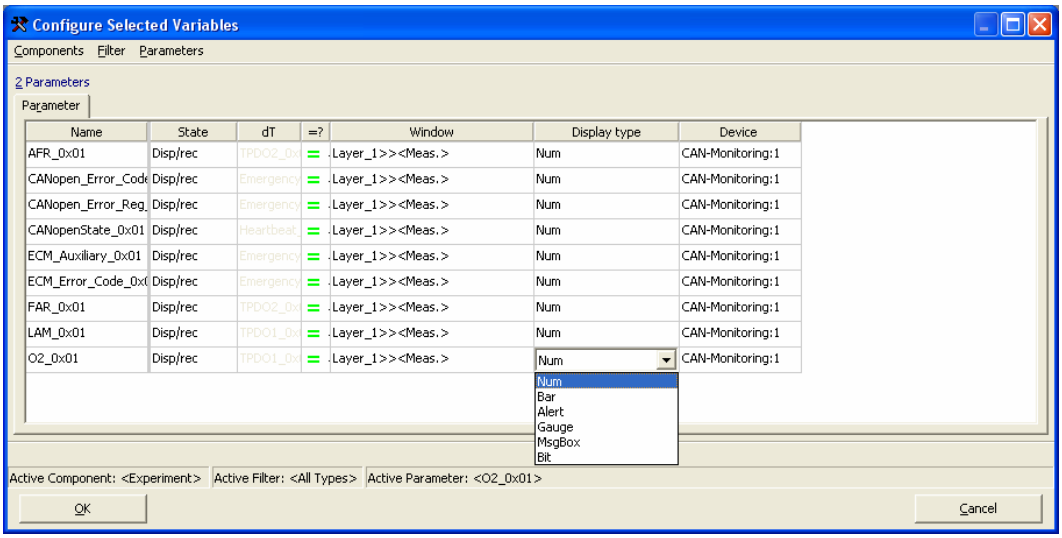

9. **Open an Experiment Environment.** Click on the Experiment Environment button on the upper tool bar to open an Experiment Environment. The Experiment Environment is where you can setup the monitoring of the CAN bus. By default, the Experiment Environment will be blank. You must select the variables from the dbc file that you wish to monitor. Click on the Select Variables icon in the left hand tool bar of the Experiment Environment.



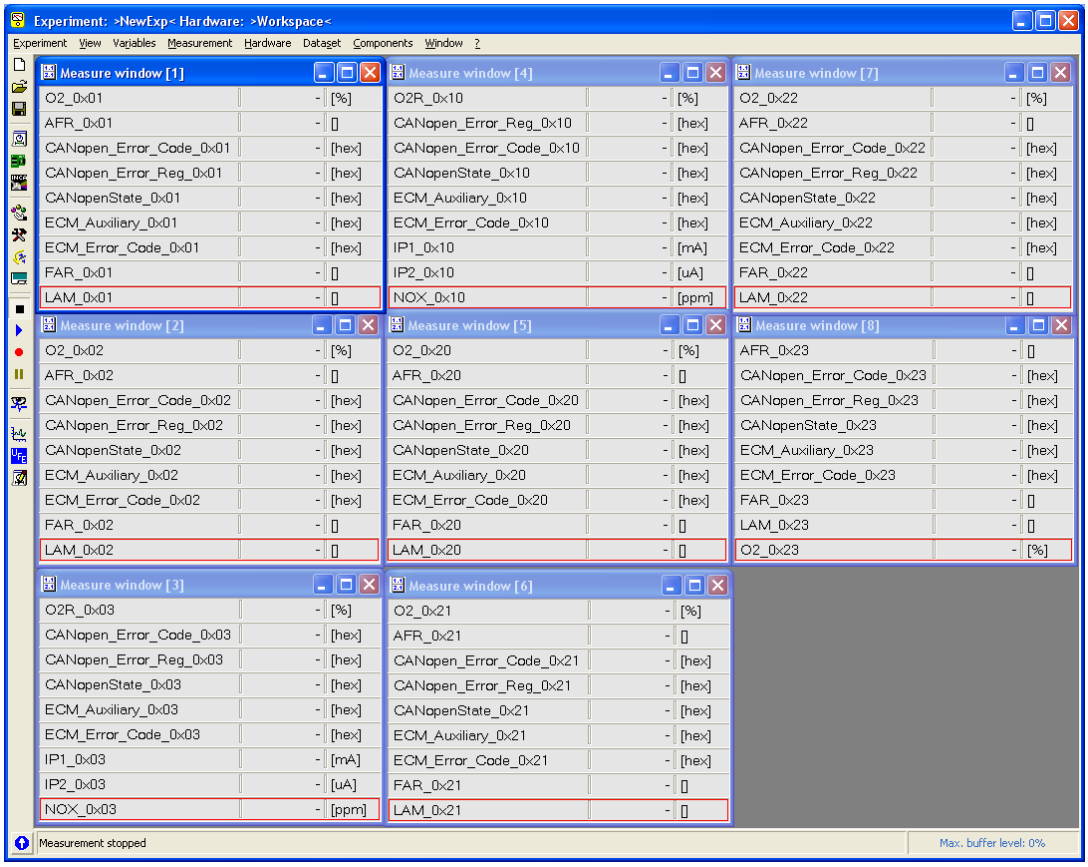
10. **Select and Configure Variables.** Select the variables that you wish to monitor in the Experiment Environment. These variables names are based on the data found in the dbc file. Click Configure.



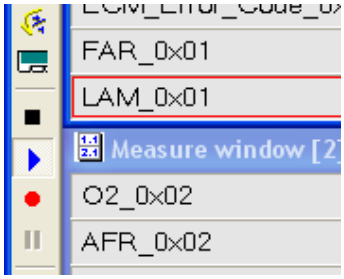

11. Another window will pop up to configure each selected variable. You can configure, for each variable, whether to record or simply display the data, how the data will be displayed (graphs, charts, gauges, numeric, etc.). When complete, click OK. We have left all configurations at default for this example.




12. A new sub-window will be added to the Experiment Environment. You do not need to select all the variables you want to monitor all at once. You can click on the Select Variables icon again at a later time to add more variables. Each set of variables you add will be placed in a new sub-window unless it is configured to join an existing sub-window. In this example, we have created a sub-window for each of the eight modules in the dbc file.




13. **Start CAN monitoring.** Right now there is no data displayed. That is because the CAN monitoring is stopped. To begin CAN monitoring, click on the Start Visualization icon (blue triangle) on the left hand tool bar. To stop CAN monitoring, click the Stop Measuring icon (black square) on the left hand tool bar. To begin recording the data, click on the Start Recording icon (red circle) on the left hand tool bar.



# Appendix D: LOCKing and unLOCKing dashCAN

When dashCAN is locked, its setup cannot be modified.

◆ **To LOCK dashCAN**

1. Press SYS until "MOd" is displayed.
2. Press ↓ until "CONF" is displayed. Then press ENT.
3. Press ↓ until "LOCK" is displayed. Then press ENT.
4. "50" will be displayed. Press ↑ until "60" is displayed. Then press ENT. dashCAN is now LOCKed.

◆ **To unLOCK dashCAN**

1. Press SYS until "LOCK" is displayed. Then press ENT.
2. "50" will be displayed. Press ↑ until "60" is displayed. Then press ENT. dashCAN is now unLOCKed.

If an unauthorized person learns that 60 is the key number, contact ECM.

# Appendix E: General Information

Voltage Input: 11 to 25 VDC @ 250 mA

Terminal Assignments on Eurofast Connector:

1. Shield, 2. 11~28 VDC In, 3. Power Ground, 4. CAN_H, 5. CAN_L

Environmental: IP67, -55 to 125 °C, 100% humidity, module is sealed

Dimensions and Weight: 120 mm x 37 mm x 143 mm, 4 ¾" x 1 ½" x 5 ¾", (W x H x D)
244 gm, 8.7 oz

# Appendix F: Programming appsCAN via CAN Messages

## 1.0 Connecting the appsCAN module

1. Power and CAN connections to the module are made using the Eurofast 12mm connector on the module. The power input requirement is 11 to 28VDC at 250mA. Multiple modules can be connected together. All modules are configured to broadcast CAN messages at the CAN bit rate of 500kbps. The maximum distance between any two nodes on the CAN bus at this baud rate is 100m. Each end of the CAN bus must have a terminating resistor of 121 Ohms.
2. Configuration software (ECM Configuration Tool) for the module is located on the CD. This software allows the setup, configuration, monitoring, and recording of data using supported CAN adapters.
3. You can lengthen the power wires on the DC Power Cable (P/N: 11-01 or 11-02) but use large gauge wire and make sure that the voltage at the power terminals of the supplied harness is at least 11V. You can lengthen the CAN communication wires using Eurofast 12mm cable. Eurofast 12mm cable was designed specifically for CAN communication and along with additional "Tees", allows you to easily build long and reliable CAN networks.
4. The appsCAN broadcasts several messages on the CAN bus using the CANopen protocol. Each message has an identifying number known as the CAN identifier (CANid). Since multiple modules can be placed on the same CAN bus, each module on the bus also has an identifying number known as the node identifier (NID). The allowable range for the NID is 0x01 to 0x7F. When connecting other non-ECM devices on the same CAN bus, ensure that the following CANids are not used:

| Message type | CANid (hex) |
|---|---|
| NMT | 0x00 |
| Emergency | 0x80 + NID |
| TPDO1 | 0x180 + NID |
| RPDO1 | 0x200 + NID |
| TPDO2 | 0x280 + NID |
| RPDO2 | 0x300 + NID |
| TPDO3 | 0x380 + NID |
| RPDO3 | 0x400 + NID |
| TPDO4 | 0x480 + NID |
| RPDO4 | 0x500 + NID |
| SDO Tx | 0x580 + NID |
| SDO Rx | 0x600 + NID |
| Heartbeat | 0x700 + NID |
| LSS | 0x7E4, 0x7E5 |

Note this list applies to EACH ECM module on the CAN bus.

# 2.0 Getting Information from the appsCAN Module

As soon as power is attached to the appsCAN module, it will perform a POWER ON/RESET sequence during which the bi-color LED will display a 2 second GREEN/BOTH/RED pattern. After the POWER ON/RESET sequence is finished, the bi-color LED will display GREEN continuously.

Approximately 5 seconds after power is applied, the unit will start broadcasting CAN messages at a CAN baud rate of 500kbps. All CAN messages have an identifier (CANid) that is related to the Node ID (NID) of the particular module. As shipped, the Node ID is pre-assigned and is written on a label above the LED. The NID can be changed using the supplied configuration software.

## 2.1 CANopen Message Types

**i) HEARTBEAT** (Broadcast rate = 0.5sec, DLC=1)

| CAN id | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x700+NID | value | | | | | | | |

value = NMT STATE (see Section 8.0 for list of NMT States)

**ii) ERROR** (Broadcast rate = 0.250sec, DLC=6)

| CAN id | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x80+NID | 0x00 | 0xFF | 0x81 | lo byte | 0x00 | 0x00 | | |

lo byte = ECM Error Code (0x00 = Data valid, see Table 3 in "Producing a .dbc File" section)

**iii) TRANSMIT PROCESS DATA OBJECT [TPDO]** (Broadcast rate = 0.005sec, DLC=8)

| **TPDO1** CAN id | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x180+NID | VRF1 (V) | | | | AIN1 (V) | | | |
| **TPDO2** CAN id | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
| 0x280+NID | VRF2 (V) | | | | VSW (V) | | | |
| **TPDO3** CAN id | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
| 0x380+NID | VRF3 (V) | | | | VEXC (V) | | | |
| **TPDO4** CAN id | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
| 0x480+NID | VRF4 (V) | | | | TEMP (degC) | | | |

The table above shows the default TPDO assignments. Note that all TPDOs are enabled (see Sections 6.8 Enable TPDO, 6.9 Disable TPDO, and 6.10 TPDO Mapping). Table 2 in "Data

Sent to (RPDO) and from (TPDO) Module" section contains a list of all available TPDO parameters.

Each module can transmit up to four TRANSMIT PROCESS DATA OBJECTS (TPDO) at the programmed TPDO broadcast rate (see Section 6.7 to determine minimum broadcast rate). A TPDO contains two data values; each corresponds to a measured parameter (e.g. VSW, AIN1, VRF1, etc). These data values are referred to as PROCESS DATA OBJECTS (PDO). Each PDO is a single precision 32 bit floating point number that conforms to the IEEE-754 standard. All TPDO data is transmitted on the CAN bus least significant byte first (Intel format).

The NID, TPDO Broadcast rate and TPDO mapping can be changed by the user.

Example: The following data was transmitted by the module with NID = 0x10 on TPDO1 and contains 2 PDOs, VRF1 and AIN1.

| **TPDO1** CAN id | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x190 | 0xA0 | 0x1A | 0x4B | 0x41 | 0x79 | 0x58 | 0xC0 | 0x3F |

VRF1 = 0x414B1AA0 = 12.694
AIN1 = 0x3FC05879= 1.5027

Configuring which PDOs are transmitted in a particular TPDO is also known as TPDO MAPPING and can be set by the user (see Section 6.10 TPDO Mapping).

**iv) RECEIVE PROCESS DATA OBJECT [RPDO]** (DLC=8)

| **RPDO1** CAN id | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x200+NID | AO1V (V) | | | | PWM1 (%) | | | |
| **RPDO2** CAN id | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
| 0x300+NID | AO2V (V) | | | | PWM2 (%) | | | |
| **RPDO3** CAN id | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
| 0x400+NID | AO3V (V) | | | | PWM3 (%) | | | |
| **RPDO4** CAN id | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
| 0x500+NID | AO4V (V) | | | | PWM4 (%) | | | |

The table above shows the default RPDO assignments. Note that all RPDOs are enabled (see Sections 6.11 Enable RPDO, 6.12 Disable RPDO, and 6.13 RPDO Mapping). Table 2 in "Data Sent to (RPDO) and from (TPDO) Module" section contains a list of all available PDO parameters.

Each module can receive up to four RECEIVE PROCESS DATA OBJECTS (RPDO). An RPDO contains two data values; each corresponds to a commanded parameter (e.g. AO1V, AO1%, PWM1, etc). These data values are referred to as PROCESS DATA OBJECTS (PDO). Each PDO is a single precision 32 bit floating point number that conforms to the

IEEE-754 standard. All RPDO data is transmitted on the CAN bus least significant byte first (Intel format).

The NID and RPDO mapping can be changed by the user.

Example: The following data was sent to the module with NID = 0x10 on RPDO1 and contains 2 PDOs, AO1V and PWM1.

| **TPDO1** CAN id | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x210 | 0xA0 | 0x1A | 0x4B | 0x41 | 0x79 | 0x58 | 0xC0 | 0x3F |

AO1V = 0x40900000 = 4.50
PWM1 = 0x42960000 = 75

Configuring how each PDO in a particular RPDO is interpreted by the module is also known as RPDO Mapping and can be set by the user (see Section 6.13 RPDO Mapping).

## 3.0 Writing to the appsCAN Module (SDO Write)

Configuration of the appsCAN module is performed by writing to the Object Dictionary (OD) and by issuing ECM CANopen OS Commands (OS Command). Both of these actions are implemented using a Service Data Object Expedited Write (SDO Write). The format is as follows:

| **SDO Write Tx** CAN id | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | Size | OD lo | OD hi | Sub | Data0 | Data1 | Data2 | Data3 |

Size = 0x2F (1 byte write)
0x2B (2 byte write)
0x23 (4 byte write)

OD lo = low byte of OD address
OD hi = hi byte of OD address
Sub = Subindex of OD address
Data0 always contains the Least Significant Byte (LSB) of the data to be written to the OD.

A SDO Write will generate the following reply:

| **SDO Write Rx** CAN id | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x580+NID | 0x60 | OD lo | OD hi | Sub | | | | |

Example: Write a 2 byte integer = 0x204 to OD address 0x5017 subindex 0 in the module with NID = 0x10.

| **SDO Write Tx** CAN id | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x610 | 0x2B | 0x17 | 0x50 | 0x00 | 0x04 | 0x02 | | |

The module will reply as follows:

| **SDO Write Rx** CAN id | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x590 | 0x60 | 0x17 | 0x50 | 0x00 | | | | |

## 4.0 Reading from the appsCAN Module (SDO Read)

During configuration, it may be necessary to read certain locations in the Object Dictionary (OD). This can be done with a Service Data Object (SDO) Read. The format for a (SDO Read) is as follows:

| **SDO Read Tx** CAN id | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x40 | OD lo | OD hi | Sub | | | | |

OD lo = low byte of OD address
OD hi = hi byte of OD address
Sub = Subindex of OD address

A SDO Read will generate the following reply:

| **SDO Read Rx** CAN id | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x580+NID | Size | OD lo | OD hi | Sub | Data0 | Data1 | Data2 | Data3 |

Size = 0x4F (1 byte response)
0x4B (2 byte response)
0x43 (4 byte response)

OD lo = low byte of OD address
OD hi = hi byte of OD address
Sub = Subindex of OD address
Data0 always contains the Least Significant Byte (LSB) of the data present at the OD address.

Example: Read OD address 0x5008 subindex 0x32 in the module with NID = 0x10.

| **SDO Write Tx** CAN id | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x610 | 0x40 | 0x08 | 0x50 | 0x32 | | | | |

The module will reply as follows:

| **SDO Write Rx** CAN id | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x590 | 0x4B | 0x08 | 0x50 | 0x32 | 0xBC | 0x02 | | |

OD address 0x5008, subindex 0x32 of the module with NID = 0x10 contains the 2 byte value 0x2BC.

## 5.0 Identifying the appsCAN Module

Each appsCAN module can be uniquely identified by reading the following four parameters in the OD:

i) Vendor ID (0x000001C6) located at OD address 0x1018, subindex 0x01 (4 byte integer/unsigned 32)
ii) Product Code (appsCAN = 0x00000009) located at OD address 0x1018 subindex 0x02 (4 byte integer/unsigned 32)
iii) Revision Number located at OD address 0x1018, subindex 0x03 (4 byte integer/unsigned 32)
iv) Serial Number located at OD address 0x1018, subindex 0x04 (4 byte integer/unsigned 32)

Furthermore, the hardware and software revision number can be found at the following locations:

i) Hardware Revision is located at OD address 0x1009, subindex 0x00 (4 byte string)
ii) Software Revision is located at OD address 0x100A, subindex 0x00 (4 byte string)

# 6.0 Commands to the appsCAN Module

## 6.1. Configure Activation of RPDOs

Send the following OS Command to configure the module to act on RPDOs immediately after they are sent.

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x2F | 0x23 | 0x10 | 0x01 | 0x33 | | | |

Send the following OS Command to configure the module to act on RPDOs only after also receiving to SYNC message (see "SYNC Messages" section).

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x2F | 0x23 | 0x10 | 0x01 | 0x34 | | | |

**SYNC Messages**
There are two different types of SYNC messages that can be transmitted. One applies only to the module selected. The other is a global SYNC which synchronizes the RPDOs of all modules on the bus configured to require a SYNC message.

To send a SYNC message to only one module, configure any one of the RPDOs for that module to be the "SYNC" parameter, and transmit a non-zero value. Since RPDOs are sent as a pair of parameters, the 2nd parameter you choose will also be applied. If you do not wish to send a 2nd parameter, you can configure it as the "NULL" parameter. See Section 6.13 RPDO Mapping for details on how to map and send RPDOs.

To send a global SYNC message, send the following message. Note that no data is necessary in the data bytes.

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x80 | | | | | | | | |

## 6.2. Configure Analog Outputs (AO1, AO2, AO3, AO4)

**Select Output Type**
Each of the four analog outputs can be configured as absolute voltages from 0 to 5V or ratiometrically as a percentage of a 0 to 15V reference input. These settings are controlled by an 8-bit register that can be accessed by performing an SDO read or write to OD address 0x5023, subindex 0x00.

| **bit 7** | **bit 6** | **bit 5** | **bit 4** | **bit 3** | **bit 2** | **bit 1** | **bit 0** |
|---|---|---|---|---|---|---|---|
| Reserved | Reserved | Reserved | Reserved | Aout4 | Aout3 | Aout2 | Aout1 |

Aout#: 0 = configure channel as absolute voltage
1 = configure channel as ratio of the reference associated with this output (i.e. Aout1 uses the voltage reference Vref1).

Example: Configure all analog channels as a ratiometric output of the reference.

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x2F | 0x23 | 0x50 | 0x00 | 0x0F | | | |

**Setting Default Output Values**

Default analog output values are stored as single precision floating point numbers (IEEE-754) in the module. These values are applied to the outputs on startup before any commands are received. There are a set of default values for both absolute voltage and ratiometric settings. The appropriate value will be used depending on the configuration selected for the particular output. They can be accessed by performing an SDO read or write to the following Object Dictionary addresses.

| | OD Address | OD Subindex | Factory Default |
|---|---|---|---|
| AO1V | 0x5025 | 0x00 | 0V |
| AO2V | 0x5025 | 0x01 | 0V |
| AO3V | 0x5025 | 0x02 | 0V |
| AO4V | 0x5025 | 0x03 | 0V |
| AO1% | 0x5025 | 0x04 | 0% |
| AO2% | 0x5025 | 0x05 | 0% |
| AO3% | 0x5025 | 0x06 | 0% |
| AO4% | 0x5025 | 0x07 | 0% |

## 6.3. Configure PWM Resolution

The PWM frequency and duty cycle resolution can be set to either 8-bit or 16-bit. In 8-bit mode, all four PWM channels can be used. In 16-bit mode, only PWM2 and PWM4 can be used, PWM1 and PWM3 are disabled. In 8-bit mode, worst case resolution is 0.4% Duty Cycle, 4 Hz. In 16-bit mode, worst case resolution is 0.008% Duty Cycle, 0.1 Hz. The worst case occurs at highest frequencies.

Send the following OS Command to configure the module's PWM outputs as 8-bit mode.

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x2F | 0x23 | 0x10 | 0x01 | 0x35 | | | |

Send the following OS Command to configure the module's PWM outputs as 16-bit mode.

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x2F | 0x23 | 0x10 | 0x01 | 0x36 | | | |

## 6.4. Configure PWM Outputs (PWM1, PWM2, PWM3, PWM4)

**Select Output Type**

There are three possible settings for each PWM output: pull-up resister, polarity, and pulse mode. These settings are controlled by a 16-bit register that can be accessed by performing an SDO read or write to OD address 0x5024, subindex 0x00.

| **bit 15** | **bit 14** | **Bit 13** | **bit 12** | **bit 11** | **bit 10** | **bit 9** | **bit 8** |
|---|---|---|---|---|---|---|---|
| Pulse4 | Pulse3 | Pulse2 | Pulse1 | Reserved | Reserved | Reserved | Reserved |
| **bit 7** | **bit 6** | **bit 5** | **bit 4** | **bit 3** | **bit 2** | **bit 1** | **bit 0** |
| Polarity4 | Polarity3 | Polarity2 | Polarity1 | Pull-up4 | Pull-up3 | Pull-up2 | Pull-up1 |

Pulse#: 0 = configure channel as a normal PWM.
1 = configure channel as a one-shot pulse output.
(see Section 6.5 Configure Pulse Mode Outputs for full details)

Polarity#: 0 = configure as active high. 100% duty cycle deactivates low-side driver and output is held high, either by the pull-up resistor or an external load.
1 = configure as active low. 100% duty cycle actives low-side driver and pulls output to ground.

Pull-up#: 0 = configure as low-side driver. Pull-up resister is disabled.
1 = configure as output with 1K pull-up resister to 5V.

Example: Configure all PWM channels as active high outputs with pull-up resisters enabled.

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x2B | 0x24 | 0x50 | 0x00 | 0x0F | 0x00 | | |

**Set Default Frequency and Duty Cycle**

Default frequency and duty cycle values are stored as single precision floating point numbers (IEEE-754) in the module. These values are applied to the outputs on startup before any commands are received. They can be accessed by performing an SDO read or write to the following Object Dictionary addresses. Note that FRQA applies to PWM channel 1 & 2, and FRQB applies to PWM channel 3 & 4.

| | OD Address | OD Subindex | Factory Default |
|---|---|---|---|
| FRQA (1&2) | 0x5027 | 0x00 | 100Hz |
| FRQB (3&4) | 0x5027 | 0x01 | 100Hz |
| PWM1 | 0x5026 | 0x00 | 0% |
| PWM2 | 0x5026 | 0x01 | 0% |
| PWM3 | 0x5026 | 0x02 | 0% |
| PWM4 | 0x5026 | 0x03 | 0% |

## 6.5. Configure Pulse Mode Outputs

These configurations only apply if the PWM output is configured as Pulse Mode (see Section 6.4 Configure PWM Outputs). Polarity and pull-up resister enables still apply in pulse mode. There are three registers required to initiate a pulse signal: delay, pulse width, and status.

The configurations for pulse mode are located at the following Object Dictionary entries.

| | OD Address |
|---|---|
| PWM1 | 0x5028 |
| PWM2 | 0x5029 |
| PWM3 | 0x502A |
| PWM4 | 0x502B |

| | OD Subindex |
|---|---|
| Status | 0x00 |
| Delay | 0x01 |
| Pulse Width | 0x02 |

Status: 0 = pulse has not started and is free for use.
1 = pulse is in use and is in the "delay" phase. Do not write to related registers.
2 = pulse is in use and is in the "on" phase. Do not write to related registers.
This register also serves as the start command. Write 1 to this location to start a pulse. A new pulse can only be started when the status is 0. Writing any other value has no effect.

Delay: Specifies in milliseconds how long to wait after receiving the start command to begin the pulse. This value is a 16-bit integer with a valid range between 1 and 60000.

Pulse Width: Specifies in milliseconds how long to hold the pulse on. This value is a 16-bit integer with a valid range between 1 and 60000.

The procedure to start a pulse is as follows:

i) Check that a pulse has not already started by reading the status to make sure it is 0.

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x40 | 0x28~0x2B | 0x50 | 0x00 | | | | |

ii) Perform an SDO write to OD address 0x5028~B, subindex 0x01 with delay value.

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x2B | 0x28~0x2B | 0x50 | 0x01 | delay lo | delay hi | | |

iii) Perform an SDO write to OD address 0x5028~B, subindex 0x02 with pulse width value.

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x2B | 0x28~0x2B | 0x50 | 0x02 | width lo | width hi | | |

iv) Perform an SDO write to OD address 0x5028~B, subindex 0x00 with the value 1.

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x2B | 0x28~0x2B | 0x50 | 0x00 | 0x01 | | | |

Example: Configure PWM3 of module with NID = 0x05 for pulse mode and start a pulse of 250ms after a delay of 1000ms.

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x605 | 0x2B | 0x24 | 0x50 | 0x00 | 0x00 | 0x40 | | |
| 0x605 | 0x2B | 0x2A | 0x50 | 0x01 | 0xFA | 0x00 | | |
| 0x605 | 0x2B | 0x2A | 0x50 | 0x02 | 0xE8 | 0x03 | | |
| 0x605 | 0x2B | 0x2A | 0x50 | 0x00 | 0x01 | | | |

## 6.6. Changing the NID

The Node ID (NID) can be programmed from 0x01 to 0x7F (1 to 127). To change the NID, several messages must be sent to the appsCAN module. This must be followed by a reset of the module (that can be performed three different ways; see the following).

Start by sending the following message to place the module into pre-operational mode.

| CAN id | byte 0 | byte 1 |
|---|---|---|
| 0x00 | 0x80 | NID |

The next message(s) place the module(s) into LSS (Layer Select Services) configuration mode. If there is only one CANopen module on the CAN bus this process requires only one message. If there are several CANopen modules on the same CAN bus the specific module must be identified using Product Code, Revision Number and Serial Number, (these can be found on a white label placed on the top of the plastic enclosure).

MULTIPLE MODULES ON BUS

| CAN id | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 |
|---|---|---|---|---|---|
| 0x7E5 | 0x04 | 0x00 | | | |
| 0x7E5 | 0x40 | 0xC6 | 0x01 | 0x00 | 0x00 |
| 0x7E5 | 0x41 | Product Code | | | |
| 0x7E5 | 0x42 | Revision Number | | | |
| 0x7E5 | 0x43 | Serial Number | | | |

SINGLE MODULE ON BUS

| CAN id | byte 0 | byte 1 |
|---|---|---|
| 0x7E5 | 0x04 | 0x01 |

The module will reply with byte 0 = 0x44 on CAN id 0x7E4 if it enters LSS configuration mode successfully.

The next message sent contains the new NID as an unsigned hexadecimal character.

| CAN id | byte 0 | byte 1 |
|---|---|---|
| 0x7E5 | 0x11 | new NID |

The module will reply with byte 0 = 0x11 and byte 1 = 0x00 on CAN id 0x7E4 indicating a successful NID change.

The last message sent takes the module out of configuration mode.

| CAN id | byte 0 | byte 1 |
|---|---|---|
| 0x7E5 | 0x04 | 0x00 |

After the NID has been successfully changed, the module enters pre-operational mode and does not broadcast data. The module can be returned to broadcast mode 1 of 3 ways:

i) Power-cycle the module by disconnecting and reconnecting the power.
ii) A second method is to send a command instructing the module to perform a hard reset (similar to power-cycling the module but software controlled).

| CAN id | byte 0 | byte 1 |
|---|---|---|
| 0x00 | 0x81 | NID |

iii) A third method is to send a command instructing the module to reset the CAN interface only.

| CAN id | byte 0 | byte 1 |
|---|---|---|
| 0x00 | 0x82 | NID |

Example: Change the NID for the following module with **multiple modules** on the CAN bus.

CURRENT NID = 0x10 (16)
PRODUCT CODE = 0x09 (9)
REVISION NUMBER = 0x01 (1)
SERIAL NUMBER = 0x192 (402)
NEW NID = 0x1A (26)

MESSAGE SENT

| CAN id | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 |
|---|---|---|---|---|---|
| 0x00 | 0x80 | 0x10 | | | |
| 0x7E5 | 0x04 | 0x00 | | | |
| 0x7E5 | 0x40 | 0xC6 | 0x01 | 0x00 | 0x00 |
| 0x7E5 | 0x41 | 0x09 | 0x00 | 0x00 | 0x00 |
| 0x7E5 | 0x42 | 0x01 | 0x00 | 0x00 | 0x00 |
| 0x7E5 | 0x43 | 0x92 | 0x01 | 0x00 | 0x00 |
| 0x7E5 | 0x11 | 0x1A | | | |
| 0x7E5 | 0x04 | 0x00 | | | |
| 0x00 | 0x82 | 0x1A | | | |

MODULE REPLY

| CAN id | byte 0 | byte 1 |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| 0x7E4 | 0x44 | |
| 0x7E4 | 0x11 | 0x00 |
| | | |
| | | |

Example: Change the NID for the **only CANopen module** on the CAN bus.
CURRENT NID = 0x10 (16)

NEW NID = 0x1A (26)

MESSAGE SENT

| CAN id | byte 0 | byte 1 |
|---|---|---|
| 0x00 | 0x80 | 0x10 |
| 0x7E5 | 0x40 | 0x01 |
| 0x7E5 | 0x11 | 0x1A |
| 0x7E5 | 0x04 | 0x00 |
| 0x00 | 0x82 | 0x1A |

MODULE REPLY

| CAN id | byte 0 | byte 1 |
|---|---|---|
| | | |
| 0x7E4 | 0x44 | |
| 0x7E4 | 0x11 | 0x00 |
| | | |
| | | |

## 6.7. Changing the TPDO Broadcast Rate

The data broadcast rate can be programmed from 5 ms to 65535 ms and applies to all TPDOs that have been enabled (see Section 6.8 Enable TPDOs). It is an unsigned 16bit integer (2 bytes) written least significant byte (LSB) first (Intel format) to OD address 0x1800, subindex 0x05. The format of the SDO Write to the appsCAN module is as follows:

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x2B | 0x00 | 0x18 | 0x05 | broadcast rate lo | broadcast rate hi | | |

Example: Set TPDO broadcast rate to 500 ms (0x01F4) for the module with NID = 0x0F (15).

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x60F | 0x2B | 0x00 | 0x18 | 0x05 | 0xF4 | 0x01 | | |

There is a minimum broadcast rate that is dependent on the number of modules transmitting on the CAN bus and how many TPDOs have been enabled for each module. If the broadcast rate is too fast the ECM Configuration Tool will not be able to identify or configure any of the modules. The formula for calculating the minimum broadcast rate is as follows:

Minimum Broadcast Rate (ms) > Total number of TPDOs for all modules x 0.3125

Example: There are 8 modules on the CAN bus.

NID 0x01 has 3 TPDOs enabled
NID 0x02 has 1 TPDOs enabled
NID 0x03 has 4 TPDOs enabled
NID 0x04 has 2 TPDOs enabled
NID 0x05 has 4 TPDOs enabled
NID 0x06 has 4 TPDOs enabled
NID 0x07 has 4 TPDOs enabled
NID 0x08 has 4 TPDOs enabled

Minimum Broadcast Rate (ms) = (3 + 1 + 4 + 2 + 4 + 4 + 4 + 4) x 0.3125 = 8.125ms. Since the broadcast rate is valid only in increments of 1ms, round 8.125ms up to the next integer value; 9ms. Therefore no module can have a TPDO broadcast rate less than 9ms.

## 6.8. Enable Transmit Process Data Object (TPDO)

There are four TPDOs; each can be individually enabled to transmit the mapped PDO data at the broadcast rate. The following OD addresses are required to enable each TPDO.

| TPDO | EnableOD Address | Transmit CANid |
|---|---|---|
| TPDO1 | 0x1800 | 0x180 + NID |
| TPDO2 | 0x1801 | 0x280 + NID |
| TPDO3 | 0x1802 | 0x380 + NID |
| TPDO4 | 0x1803 | 0x480 + NID |

To enable a TPDO, perform a SDO Write to the Enable OD Address for that particular TPDO as follows:

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x23 | EnableOD Address lo | EnableOD Address hi | 0x01 | Transmit CANid lo | Transmit CANid hi | 0x00 | 0x40 |

Example: Enable TPDO4 for the module with NID = 0x20, (EnableOD Address = 0x1803, Transmit CANid = 0x480 + 0x20 = 0x4A0).

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x620 | 0x23 | 0x03 | 0x18 | 0x01 | 0xA0 | 0x04 | 0x00 | 0x40 |

## 6.9. Disable Transmit Process Data Object (TPDO)

The following OD addresses are required to disable each TPDO.

| TPDO | EnableOD Address | Transmit CANid |
|---|---|---|
| TPDO1 | 0x1800 | 0x180 + NID |
| TPDO2 | 0x1801 | 0x280 + NID |
| TPDO3 | 0x1802 | 0x380 + NID |
| TPDO4 | 0x1803 | 0x480 + NID |

To disable a TPDO, perform a SDO Write to the Enable OD Address for that particular TPDO as follows:

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x23 | EnableOD Address lo | EnableOD Address hi | 0x01 | Transmit CANid lo | Transmit CANid hi | 0x00 | 0xC0 |

Example: Enable TPDO1 for the module with NID = 0x10, (EnableOD Address = 0x1800, Transmit CANid = 0x180 + 0x10 = 0x190).

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x620 | 0x23 | 0x00 | 0x18 | 0x01 | 0x90 | 0x01 | 0x00 | 0xC0 |

## 6.10. Transmit Process Data Object Mapping (TPDO MAPPING)

Each TPDO transmits two PROCESS DATA OBJECTS (PDOs). Which PDOs are transmitted by the module in a particular TPDO can be configured by the user.

Configuring a TPDO is a 4 step process:

i) Write a 0 to the TPDO Configuration OD Address, subindex 0x00.
ii) Enter the OD address of the 1st PDO.
(see Table 2 in "Data Sent to and from Module")
iii) Enter the OD address of the 2nd PDO.
iv) Enter the number of PDOs in the TPDO.

Also, the following information is required to successfully map a TPDO.

| TPDO | ConfigOD Address | EnableOD Address | Transmit CANid |
|---|---|---|---|
| TPDO1 | 0x1A00 | 0x1800 | 0x180 + NID |
| TPDO2 | 0x1A01 | 0x1801 | 0x280 + NID |
| TPDO3 | 0x1A02 | 0x1802 | 0x380 + NID |
| TPDO4 | 0x1A03 | 0x1803 | 0x480 + NID |

Write a 0 to the TPDO Configuration OD Address, subindex 0x00 by performing a SDO Write as follows:

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x2F | ConfigOD Address lo | ConfigOD Address hi | 0x00 | 0x00 | | | |

Configure the 1st PDO by performing a SDO Write follows:

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x23 | ConfigOD Address lo | ConfigOD Address hi | 0x01 | 0x20 | 0x00 | PDO OD Address lo | PDO OD Address hi |

Configure the 2nd PDO by performing a SDO Write follows:

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x23 | ConfigOD Address lo | ConfigOD Address hi | 0x02 | 0x20 | 0x00 | PDO OD Address lo | PDO OD Address hi |

Enter the number of PDOs in the TPDO by performing a SDO Write as follows:

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x2F | ConfigOD Address lo | ConfigOD Address hi | 0x00 | 0x02 | | | |

Example: Map the PDO for AIN1 (V) and VRF3 (V) to TPDO2 for the module with NID = 0x02. (AIN1 PDO OD Address = 0x2027, VRF3 PDO OD Address = 0x2025, ConfigOD Address for TPDO2 = 0x1A01)

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x602 | 0x2F | 0x01 | 0x1A | 0x00 | 0x00 | | | |
| 0x602 | 0x23 | 0x01 | 0x1A | 0x01 | 0x20 | 0x00 | 0x27 | 0x20 |
| 0x602 | 0x23 | 0x01 | 0x1A | 0x02 | 0x20 | 0x00 | 0x25 | 0x20 |
| 0x602 | 0x2F | 0x01 | 0x1A | 0x00 | 0x02 | | | |

### 6.11. Enable Receive Process Data Object (RPDO)

There are four RPDOs; each can be individually enabled to be received by the module as the mapped PDO data. The module will ignore sent RPDOs if it is disabled. The following OD addresses are required to enable each RPDO.

| RPDO | EnableOD Address | Receive CANid |
|---|---|---|
| RPDO1 | 0x1400 | 0x200 + NID |
| RPDO2 | 0x1401 | 0x300 + NID |
| RPDO3 | 0x1402 | 0x400 + NID |
| RPDO4 | 0x1403 | 0x500 + NID |

To enable a RPDO, perform a SDO Write to the Enable OD Address for that particular RPDO as follows:

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x23 | EnableOD Address lo | EnableOD Address hi | 0x01 | Receive CANid lo | Receive CANid hi | 0x00 | 0x40 |

Example: Enable RPDO4 for the module with NID = 0x20, (EnableOD Address = 0x1403, Receive CANid = 0x500 + 0x20 = 0x520).

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x620 | 0x23 | 0x03 | 0x14 | 0x01 | 0x20 | 0x05 | 0x00 | 0x40 |

## 6.12. Disable Receive Process Data Object (RPDO)

The following OD addresses are required to disable each RPDO.

| RPDO | EnableOD Address | Receive CANid |
|---|---|---|
| RPDO1 | 0x1400 | 0x200 + NID |
| RPDO2 | 0x1401 | 0x300 + NID |
| RPDO3 | 0x1402 | 0x400 + NID |
| RPDO4 | 0x1403 | 0x500 + NID |

To disable a RPDO, perform a SDO Write to the Enable OD Address for that particular RPDO as follows:

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x23 | EnableOD Address lo | EnableOD Address hi | 0x01 | Receive CANid lo | Receive CANid hi | 0x00 | 0xC0 |

Example: Enable RPDO1 for the module with NID = 0x10, (EnableOD Address = 0x1400, Receive CANid = 0x200 + 0x10 = 0x210).

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x620 | 0x23 | 0x00 | 0x14 | 0x01 | 0x10 | 0x02 | 0x00 | 0xC0 |

## 6.13. Receive Process Data Object Mapping (RPDO MAPPING)

Each RPDO is sent to the module with two PROCESS DATA OBJECTS (PDOs). Which PDOs that the module interprets them as can be configured by the user.

Configuring a RPDO is a 4 step process:

i) Write a 0 to the RPDO Configuration OD Address, subindex 0x00.
ii) Enter the OD address of the 1st PDO.
(see Table 2 in "Data Sent to and from Module")
iii) Enter the OD address of the 2nd PDO.
iv) Enter the number of PDOs in the RPDO.

Also, the following information is required to successfully map a RPDO.

| RPDO | ConfigOD Address | EnableOD Address | Receive CANid |
|---|---|---|---|
| RPDO1 | 0x1600 | 0x1400 | 0x200 + NID |
| RPDO2 | 0x1601 | 0x1401 | 0x300 + NID |
| RPDO3 | 0x1602 | 0x1402 | 0x400 + NID |
| RPDO4 | 0x1603 | 0x1403 | 0x500 + NID |

Write a 0 to the RPDO Configuration OD Address, subindex 0x00 by performing a SDO Write as follows:

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x2F | ConfigOD Address lo | ConfigOD Address hi | 0x00 | 0x00 | | | |

Configure the 1st PDO by performing a SDO Write follows:

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x23 | ConfigOD Address lo | ConfigOD Address hi | 0x01 | 0x20 | 0x00 | PDO OD Address lo | PDO OD Address hi |

Configure the 2nd PDO by performing a SDO Write follows:

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x23 | ConfigOD Address lo | ConfigOD Address hi | 0x02 | 0x20 | 0x00 | PDO OD Address lo | PDO OD Address hi |

Enter the number of PDOs in the RPDO by performing a SDO Write as follows:

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x2F | ConfigOD Address lo | ConfigOD Address hi | 0x00 | 0x02 | | | |

Example: Map the PDO for FRQA (Hz) and PWM1 (%) to RPDO2 for the module with NID = 0x02. (FRQA PDO OD Address = 0x202D, PWM1 PDO OD Address = 0x2029, ConfigOD Address for RPDO2 = 0x1601)

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x602 | 0x2F | 0x01 | 0x16 | 0x00 | 0x00 | | | |
| 0x602 | 0x23 | 0x01 | 0x16 | 0x01 | 0x20 | 0x00 | 0x2D | 0x20 |
| 0x602 | 0x23 | 0x01 | 0x16 | 0x02 | 0x20 | 0x00 | 0x29 | 0x20 |
| 0x602 | 0x2F | 0x01 | 0x16 | 0x00 | 0x02 | | | |

### 6.14. Factory Reset

Parameters that are stored in non-volatile memory (EEprom) can be reset to a standard configuration by issuing the ECM OS Command 0xDF (see Appendix B).

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x2F | 0x23 | 0x10 | 0x01 | 0xDF | | | |

Issuing this command sets configuration and module parameters as follows:

1. Analog and PWM output configurations and default values return to factory settings.
2. Module output synchronization is disabled. RPDOs applied immediately when sent.
3. Expert mode disabled.
4. TPDOs are reset to factory default.
5. RPDOs are reset to factory default.
6. TPDO Broadcast rate set to 5ms (see section 8.6)

## 7.0 ECM CANopen OS Commands

A user-specific CANopen OS Command to the appsCAN module is sent using an SDO expedited write message in the following form. These commands apply only to the appsCAN module and are listed on the following page:

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x2F | 0x23 | 0x10 | 0x01 | Command | | | |

Issuing a SDO Read of OD address 0x1023, subindex 0x02 will indicate the status of the command.

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x40 | 0x23 | 0x10 | 0x02 | | | | |

The module will reply as follows:

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x580+NID | 0x4F | 0x23 | 0x10 | 0x02 | Status | | | |

The values that may be returned are listed below.

| Status | |
|---|---|
| 0x00 | Last command completed. No error occurred. No reply. |
| 0x01 | Last command completed. No error occurred. The reply can now be read. |
| 0x02 | Last command completed. Error occurred. No reply. |
| 0x03 | Last command completed. Error occurred. The reply can now be read. |
| 0x04 - FE | Reserved |
| 0xFF | Command is executing. |

If there is a reply it can read using an SDO Read of OD address 0x1023, subindex 0x03.

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x600+NID | 0x40 | 0x23 | 0x10 | 0x03 | | | | |

The reply value will be located in byte 4 of the response to the SDO Read.

| CANid | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x580+NID | 0x4F | 0x23 | 0x10 | 0x03 | Reply | | | |

The reply values and what they indicate are listed on the following page. Commands that are written in *ITALICS* are valid only in Expert Mode.

| COMMAND | VALUE | REPLY | DESCRIPTION |
|---|---|---|---|
| ResetAllFilters | 0x15 | | Resets alpha for recursive average to factory values |
| | | 0x00 | defAlphaOK |
| ExpertModeDisable | 0x16 | None | This command removes the unit from expert mode. |
| NoUpdateCanIdsOnNewNodeId | 0x17 | None | |
| UpdateCanIdsOnNewNodeId | 0x18 | None | |
| ResetTPDOs | 0x1F | None | Set all TPDOs as delivered from factory. |
| DisableTPDOCOBReset | 0x22 | None | Do not allow CAN id's to be set by CANopen spec.<br>This affects TPDO's 1-4, Heartbeat, Error messages. |
| EnableTPDOCOBReset | 0x23 | None | CAN id's set by CANopen spec based on module's NID. |
| ResetRPDOs | 0x30 | None | Set all RPDOs as delivered from factory. |
| DisableRPDOCOBReset | 0x31 | None | Do not allow CAN id's to be set by CANopen spec.<br>This affects RPDO's 1-4. |
| EnableRPDOCOBReset | 0x32 | None | RPDO CAN id's set by CANopen spec based on module's NID. |
| DisableSyncMode | 0x33 | None | Disable updating outputs synchronously to a sync message. |
| EnableSyncMode | 0x34 | None | Enable updating outputs synchronously to a sync message. |
| 8bitPWM | 0x35 | None | Use four 8-bit PWMs. |
| 16bitPWM | 0x36 | None | Use two 16-bit PWMs. |
| FactoryReset | 0xDF | None | Set all EE values to std configuration. |
| ExpertModeEnable | 0xE0 | None | This command enables expert mode. (ulng password required @ 0x52FF) |
| *UseCalibratedValues* | 0xE5 | None | Use calibrated ADC/DAC values |
| *UseUnCalibratedValues* | 0xE6 | None | Use raw uncalibrated ADC/DAC |
| *UseProgrammedFilter* | 0xE7 | None | Use programmed recursive average filter constants (std) |
| *UseCalibrationFilter* | 0xE8 | None | Use slow recursive average filter for calibration |

## 8.0 Heartbeat

A Heartbeat message is transmitted every 0.5 seconds by the appsCAN module. During normal operation the module is in operational mode (NMT state = 0x05).

| CAN id | byte 0 | byte 1 | byte 2 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 |
|---|---|---|---|---|---|---|---|---|
| 0x700+NID | NMT state | | | | | | | |

| NMT state | |
|---|---|
| 0x00 | Boot-up |
| 0x04 | Stopped |
| 0x05 | Operational |
| 0x7F | Pre-operational |

ECM ENGINE CONTROL AND MONITORING

586 Weddell Drive, Suite 2
Sunnyvale, CA 94089-2134
USA
Phone: (408) 734-3433
FAX: (408) 734-3432
Email: sales@ecm-co.com
Web: www.ecm-co.com

# EC DECLARATION OF CONFORMITY

We declare under our sole responsibility that the products:

**AFM1540 Lambda Module**
**AFM1600 Lambda and $O_2$ Analyzer**
**DIS1000 Display head**
**EGR 4830 Analyzer**
**NOx 5210 NOx Analyzer**
**Lambda 5220 Lambda Analyzer**
**EGR 5230 EGR Analyzer**
**LambdaCAN Lambda Module**
**LambdaCANc Lambda Module**
**NOxCAN NOx Module**
**NOxCANg NOx Module**
**NOx1000 NOx Module**
**dashCAN**
**DashCANc**
**appsCAN**
**gpioCAN**
**SIM300**
**SIM400**
**BTU200**

To which this declaration relates are in conformity with the essential requirements of the following standards:
**EN61326: 1997/A2: 2001 (Class A & Annex A)**
**EN61010-1: 2001 (Electrical Safety)**

And therefore conform to the requirements of the following directives:
**89/336/EEC Electromagnetic Compatibility (EMC)**
**72/23/EEC Low Voltage Directive (LVD)**

Ronald S. Patrick
Vice President Sales
November 20, 2009

ECM ENGINE CONTROL AND MONITORING
Los Altos, CA 94023-0040 • USA • (408) 734-3433 • Fax: (408) 734-3432 • www.ecm-co.com