CAN RX INTEGRATION KIT FOR PAC PROJECT GUIDE

Form 1983-1700606—June 2017

OPTO 22
Automation made simple.

43044 Business Park Drive • Temecula • CA 92590-3614
Phone: 800-321-OPTO (6786) or 951-695-3000
Fax: 800-832-OPTO (6786) or 951-695-2712
www.opto22.com

Product Support Services
800-TEK-OPTO (835-6786) or 951-695-3080
Fax: 951-695-3017
Email: support@opto22.com
Web: support.opto22.com

CAN RX Integration Kit for PAC Project Guide
Form 1983-1700606—June 2017

The information in this manual has been checked carefully and is believed to be accurate; however, Opto 22 assumes no responsibility for possible inaccuracies or omissions. Specifications are subject to change without notice.

Opto 22 warrants all of its products to be free from defects in material or workmanship for 30 months from the manufacturing date code. This warranty is limited to the original cost of the unit only and does not cover installation, labor, or any other contingent costs. Opto 22 I/O modules and solid-state relays with date codes of 1/96 or newer are guaranteed for life. This lifetime warranty excludes reed relay, SNAP serial communication modules, SNAP PID modules, and modules that contain mechanical contacts or switches. Opto 22 does not warrant any product, components, or parts not manufactured by Opto 22; for these items, the warranty from the original manufacturer applies. Refer to Opto 22 form 1042 for complete warranty information.

Opto 22
Automation Made Simple.

Table of Contents

Getting Started

Introduction

The Opto 22 CAN Integration Kit for PAC Project™ - Receive Only (part number PAC-INT-CAN-RX) provides a sample PAC Control strategy to enable your Opto 22 PAC system equipped with one or more SNAP-SCM-CAN2B modules to *receive* data from devices on a Controller Area Network (CAN) protocol network.

The sample strategy in the PAC-INT-CAN-RX integration kit runs on an Opto 22 SNAP PAC controller, and includes a variety of Parameter Group Numbers (PGNs) for use with J1939, NMEA 2000, ISO 11783, and other communication standards. (A PGN is the part of the CAN data packet header that identifies how the data is assigned.) Depending on your CAN's implementation, you may be able to use the strategy as-is; if not, you can modify or add to it to work with your system.

NOTE: *If you also want to transmit data to devices on a CAN protocol network, do not use this integration kit. Instead, use the free Opto 22 CAN Integration Kit for PAC Project - Transmit and Receive, part number PAC-INT-CAN-RXTX, available on our website. To transmit as well as receive, your SNAP-SCM-CAN2B modules must have module firmware version R2.0b or higher. Modules with firmware R1.0d and lower can receive data only.*

The example strategy includes the following:

- CAN_Comm chart, which receives data from one to four Opto 22 SNAP-SCM-CAN2B modules. The module can receive all CAN data, or filters can be used to filter the PGNs received.
- Data_Extract chart, which converts the PGN data to variables for each PGN at each address
- Get_CAN_Data subroutine, which you can add to your strategy

This guide assumes that you understand how to use PAC Control™, and how to use and configure an Opto 22 SNAP PAC controller. It also assumes that you understand how to use the CAN protocol and the PGNs required for your system.

What is Required

You will need the following things:

- A PC with PAC Project R9.3a or newer (Basic or Pro)
- A PAC Control strategy that includes a SNAP PAC I/O unit with SNAP-SCM-CAN2B serial communication modules with module firmware R1.0d or lower, connected to devices on a

CAN network. The I/O unit must use a SNAP PAC R-Series controller or EB-series brain with firmware version R9.3a or newer.

- An Opto 22 SNAP PAC R-series or S-series controller

For More Information

For more information, see the following guides:

- Form 1537, *SNAP-SCM-CAN2B Communication Module Data Sheet*
- Form 1704, the *PAC Manager User's Guide*
- Form 1592, the *SNAP PAC S-series User's Guide*
- Form 1595, the *SNAP PAC R-series User's Guide*
- Form 1700, the *PAC Control User's Guide*
- For information on the PGNs required for your device, see the manufacturer's documentation.

Using the Charts and Subroutine

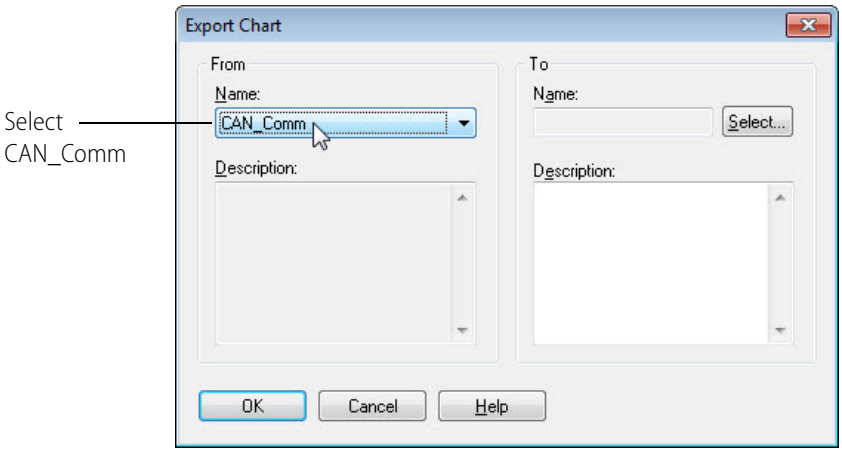
In order to use the example strategy charts, you must export these charts from the example strategy, and then import the charts into your own strategy as described in the following sections:

- "Exporting the Charts" (see below)
- "Importing the Charts" on page 3

To use the subroutine, see "Including the Subroutine" on page 3.

Exporting the Charts

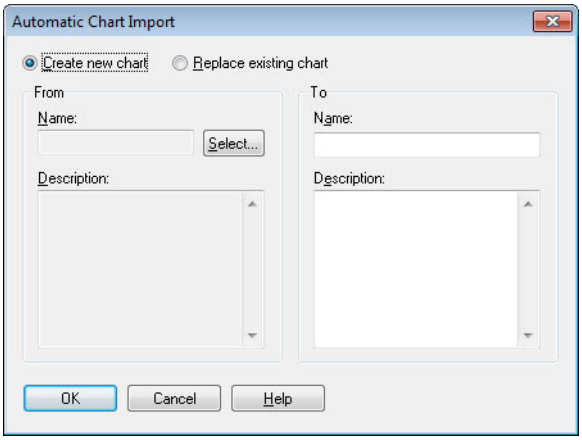
1. Open the zip file, and extract the contents of the zip file to a directory on your hard drive.
2. Start PAC Control, and open the strategy file (PACCAN.idb) you just extracted to your hard drive.
3. Select Chart > Export to open the Export Chart dialog box.



4. In the From combo box, select CAN_Comm.
5. Under To, click Select to open the Select Destination dialog box, and then browse to an appropriate directory such as your strategy's directory.
6. Name the export file *CAN_Comm_export*, and then click Save to close the Select Destination dialog box.
7. Click OK to export the file and exit the Export Chart dialog box.
8. Repeat steps 3-7 to export the Data_Extract chart. Name the export file *Data_Extract_export*.

Importing the Charts

1. Open the strategy you want to use with the CAN2B module.
2. Select Chart > Import to open the Automatic Chart Import dialog box.



3. With "Create new chart" selected, click Select to open the Select File to Import dialog box.
4. Browse to the directory that contains the export file, CAN_Comm_export.cxf.
5. Select the export file, and then click Open.

   The Select File to Import dialog box closes.
6. Under To in the Automatic Chart Import dialog box, enter the name of the chart, *CAN_Comm*. Then click OK.
7. Repeat steps 2-6 to import the Data_Extract_export.cxf file. Name the chart *Data_Extract*.

Including the Subroutine

1. Make sure the strategy is open in Config mode.
2. Select Configure > Subroutines Included to open the Subroutine Files dialog box.
3. Click Add and navigate to the Get_CAN_Data.isb subroutine file from the folder PACCAN\Subs.
4. Click OK.

The subroutine appears in the Subroutines Included folder and is ready to be used in your strategy.

Configuring the Charts

This chapter includes the following topics:

Configuring Communications in CAN_Comm Chart

The CAN_Comm chart has several PGNs included for you to use. See "Example PGNs Included" on page 11.

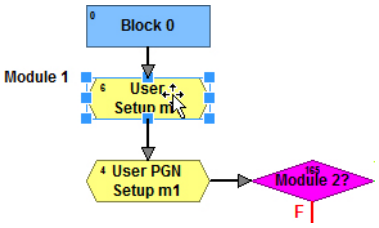
Adding or modifying a PGN is a four-step process, explained in the following pages:

- Adjust the PGN filters in the User Setup block of the CAN_Comm chart. See the next section, "Configuring User Setup m1 Block (Module 1)".
- Add the variables and set the parameters for the PGN in the User PGN Setup block of the chart CAN_Comm. See page 7.
- Add the PGN in the User Get Pointers block of the CAN_Comm chart. See page 9.
- Add the data conversion for the PGN in the Data_Extract chart. See page 10.

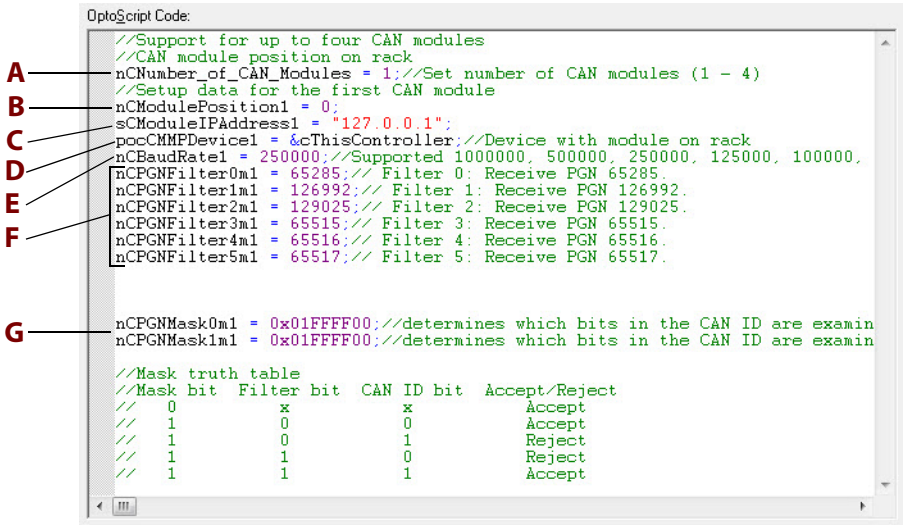
The SNAP-SCM-CAN2B module is configured in the CAN_Comm chart.

Configuring User Setup m1 Block (Module 1)

1. With your strategy open in PAC Control, open the CAN_Comm chart.
2. Double-click on the User Setup m1 block to open the OptoScript dialog.



This opens the OptoScript dialog box.



3. Enter the following information in the OptoScript Code:

- **A** Enter the number of CAN modules (1 – 4). This is the current limit of this chart; it can be modified for additional modules.
- **B** Enter the position of the module in the rack (0 -15).
- **C** Enter the IP address of the brain or controller for module. Use 127.0.0.1 for local rack. The port number is calculated by the strategy.
- **D** Add the I/O unit to the pointer variable. This is used to write to the memory map area of the brain or controller.
- **E** Enter the baud rate for the module: 1000000, 500000, 250000, 125000, 100000, 50000, 20000, or 10000 bps
- **F** For the filters, enter the PGN numbers you want the module to pass to the receive buffer. In the example it will receive PGNs 65285, 126992, 129025, 65515, 65516, 65517. All others will not be received. If no filter is used, all CAN packets will be received.
- **G** The mask in the example is setup to work with the filters. Mask 0 affects filter 0 and 1. Mask 1 affects filter 2, 3, 4 and 5. See also, "Filters and Data Masks Example" on page 7.

NOTE: The filters and data masks can also be configured in PAC Manager. However, the settings in your strategy will override the settings in PAC Manager for each module. For more information on using PAC Manager to set the filters and data masks, see form 1191, the SNAP Serial Communication Module User's Guide.

Filters and Data Masks Example

Use the following table to determine how CAN packets are accepted or rejected. For example, if you want to receive CAN ID 0x00**FF05**00 and only care about the part in **bold**, then set your Mask to 0x00FFFF00 and your Filter to 0x00FF0500.

Data Mask Bit	Filter Bit	CAN ID Bit	Accept/Reject
0	X*	X*	Accept
1	0	0	Accept
1	0	1	Reject
1	1	0	Reject
1	1	1	Accept

* X = Don't Care

In the following examples, the bits we care about are determined by the Data Mask.

Packet Accepted Example.

This packet is accepted because the Filter bits match the CAN Packet bits.

```
Data Mask  = 0000 0000 1111 1111 1111 1111 0000 0000
Filter     = XXXX XXXX 1111 1111 0000 0101 XXXX XXXX
CAN Packet = XXXX XXXX 1111 1111 0000 0101 XXXX XXXX
```

Packet Rejected Example.

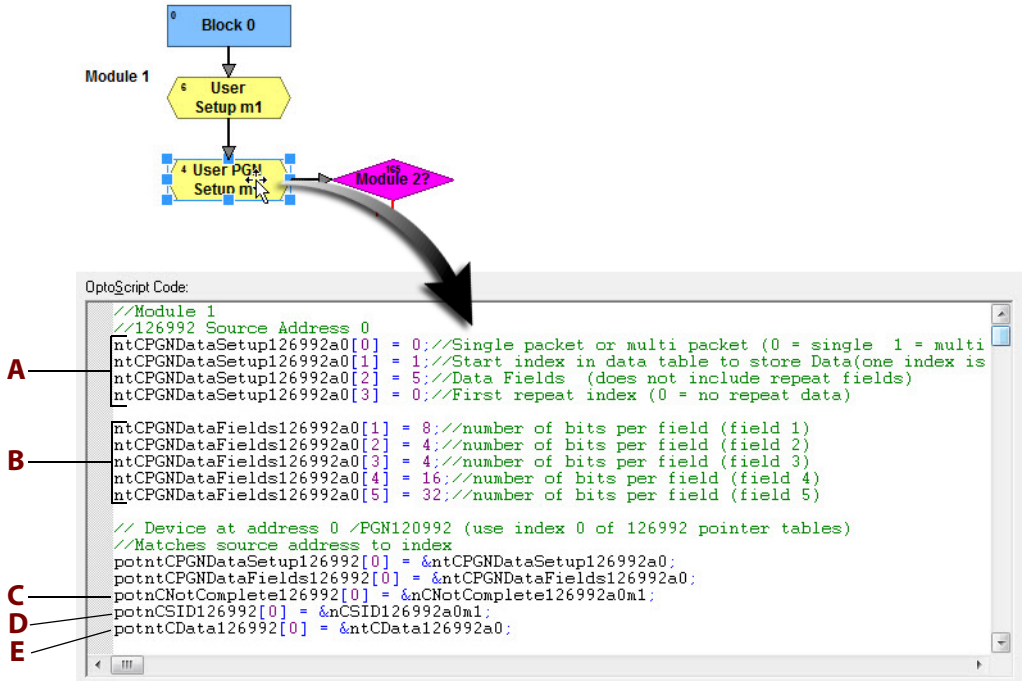
This packet is rejected because the Filter bits do not match the CAN Packet bits.

```
Data Mask  = 0000 0000 1111 1111 1111 1111 0000 0000
Filter     = XXXX XXXX 1111 1111 0000 0101 XXXX XXXX
CAN Packet = XXXX XXXX 1111 1111 0001 0111 XXXX XXXX
```

Configuring User PGN Setup m1 Block (Module 1)

Each PGN at each address uses three integer 32 tables and two integer 32 variables.

1. Double-click on the User PGN Setup m1 block to open the OptoScript dialog.



2. Enter the necessary information as described below.

A The PGN Data Setup table (integer 32 table).

Index 0 = 0 for single packet and 1 for multi packet.

Index 1 = The start index for the data storage table.

Index 2 = Number of data fields. This does not include repeat fields.

Index 3 = First repeat field. Use this when data fields repeat. 0 = no repeat fields.

B The PGN Data Fields table (integer 32 table)

Index 0 = Not used

Index 1 = Bits of first data field

Index 2 = Bits of second data field

Configure indexes as needed, one per data field.

C The Not Complete variable is used by the strategy to track incomplete data fields. (Integer 32)

D The SID variable is used by the strategy to track multiple packets for a PGN. (Integer 32)

E The data table is the table the data will be stored in. (integer 32 table)

Each PGN uses 5 pointer tables. The source address determines the index used.

Example Setup for PGN 65285 Source Address 1

```
ntCPGNDataSetup65285a1[0] = 0;//Single packet or multi packet (0 = single 1 = multi packet)
ntCPGNDataSetup65285a1[1] = 1;//Start index in data table to store Data (one index is used for each data field)
ntCPGNDataSetup65285a1[2] = 5;//Data Fields (does not include repeat fields)
ntCPGNDataSetup65285a1[3] = 0;//First repeat index (0 = no repeat data)

ntCPGNDataFields65285a1[1] = 8;//number of bits per field (field 1)
ntCPGNDataFields65285a1[2] = 16;//number of bits per field (field 2)
ntCPGNDataFields65285a1[3] = 16;//number of bits per field (field 3)
ntCPGNDataFields65285a1[4] = 16;//number of bits per field (field 4)
ntCPGNDataFields65285a1[5] = 8;//number of bits per field (field 5)

// Device at address 1 /PGN65285 (use index 1 of 65285 pointer tables)
//Matches source address to index
potntCPGNDataSetup65285[1] = &ntCPGNDataSetup65285a1;
potntCPGNDataFields65285[1] = &ntCPGNDataFields65285a1;
potnCNotComplete65285[1] = &nCNotComplete65285a1m1;
potnCSID65285[1] = &nCSID65285a1m1;
potntCData65285[1] = &ntCData65285a1;
```

In User Get Pointers m1 (Module 1)

This block gets the pointer variables for each PGN used by modules. There are five pointer tables per PGN which are loaded in the User Block. If you add a new PGN, you need to duplicate what is done for the existing PGNs. You can either remove the unused PGNs or just leave them.

To open the OptoScript dialog box, double-click block 100, User Get Pointers.

Use the same setup for modules 2, 3 and 4 if those modules are used. The data fields for each PGN are received to tables with no conversion.

Example Setup for PGN 65285 Source Address 1

```
 if (not potntCPGNDataSetup65285[nCtSourceAddressRecm1] == null) then
       pontCPGNDataSetup = potntCPGNDataSetup65285[nCtSourceAddressRecm1];//Get Pointer table for this address
       pontCRGNDataFields = potntCPGNDataFields65285[nCtSourceAddressRecm1];//Get Pointer table for this address
       ponCNotComplete = potnCNotComplete65285[nCtSourceAddressRecm1];//Get Pointer table for this address
       ponCSID = potnCSID65285[nCtSourceAddressRecm1];//Get Pointer table for this address
       pontCData = potntCData65285[nCtSourceAddressRecm1];//Get Pointer table for this address
       SetVariableTrue(nCPassFilter);//Used by strategy to check if PGN and address are configured
else
       SetVariableFalse(nCPassFilter);// Used by strategy to check if PGN and address are configured
endif
```

Logging received data

To enable logging of received data, in Debug mode set the module's log variable to true (1). The data will be logged to the log's string table until the table is full. Then it will set the log variable to false. The data is in hexadecimal.

- For Module 1, set variable nCLogDatam1 to true. The data will be logged to the string table stCLogm1 until the table is full. Then it will set the variable nCLogDatam1 to false.
- For Module 2, set variable nCLogDatam2 to true. The data will be logged to the string table stCLogm2 until the table is full. Then it will set the variable nCLogDatam2 to false.
- For Module 3, set variable nCLogDatam3 to true. The data will be logged to the string table stCLogm3 until the table is full. Then it will set the variable nCLogDatam3 to false.
- For Module 4, set variable nCLogDatam4 to true. The data will be logged to the string table stCLogm4 until the table is full. Then it will set the variable nCLogDatam4 to false.

Log example: 54 FF 05 01 08 8C 80 FF 76 74 FF FF FF

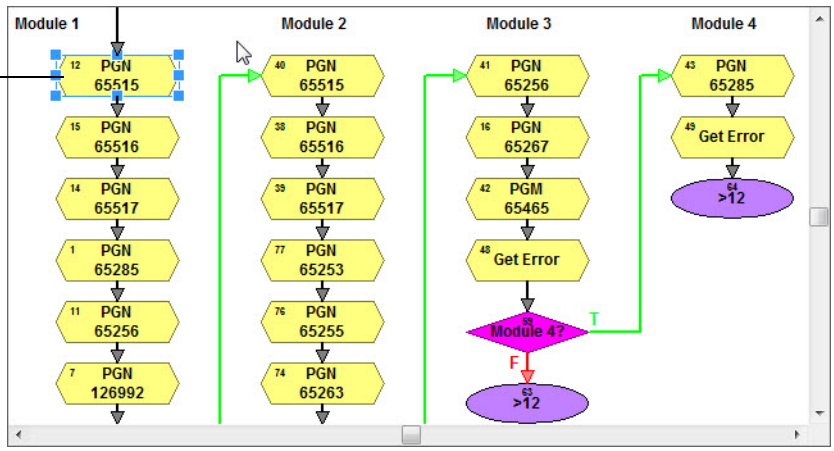
Data_Extract Chart

The Data_Extract chart converts the data from the PGN receive tables to variables for each module. The chart includes a variety of PGN conversions that you can use. In order to match your system, you will need to modify the chart's PGN conversions or create some new ones. For a list of the PGNs included in this chart, see "Example PGNs Included" on page 11.

Extracted data can be viewed in PAC Display. For more information, see form 1702, the *PAC Display User's Guide*.

Data_Extract Chart

PGN Conversion OptoScript block



Example PGNs Included

The following example PGNs are provided:

PGN	Name
65515	Spreader Status
65516	Position
65517	Transmission Status
65256	Direction/Speed
65285	Temperature
126992	System Time and Data
129025	Position
129026	Course Over Ground (COG) and Speed Over Ground (SOG)
129029	Global Navigation Satellite System (GNSS) Position
129539	GNSS Dilution of Precision components (DOPs)
129540	GNSS Satellites (Sats) in View

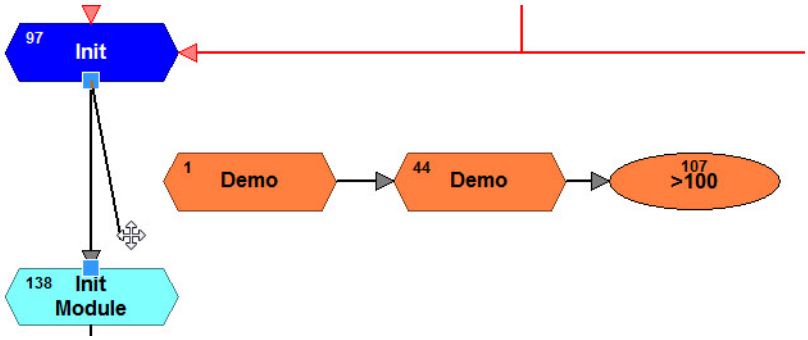
Using the Demo Blocks

Demo blocks (1, 44, 100 and 90) are provided in the CAN_Comm chart to simulate or test data without using a SNAP-SCM-CAN2B module. In case you don't have access to a module, this allows you to test your strategy.

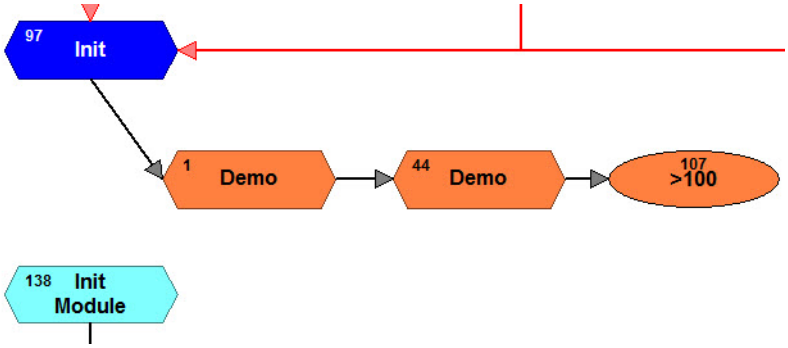
Block 44 moves the data to the header and data string. From this point on it uses the same blocks that the real CAN data uses. If data conversion is set up in the Data_Extract chart the data will be converted as normal.

To use the demo blocks:

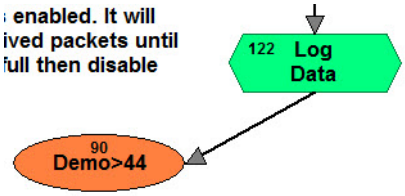
1. Use normal setup for the PGNs you want to test. The demo blocks use the setup for module 1.
2. Disconnect block 138 from block 97.



3. Reconnect block 97 to block 1.



4. Disconnect block 122 from 183, and then reconnect block 122 to block 90.



5. Enter the simulated CAN data in block 1.

Now you are ready to run your strategy using the demo blocks.

Structure of a CAN Packet in PAC Control

In PAC Control, bytes 0–12 in a CAN packet are as follows:

Byte	Description
Byte 0	CAN ID (MSB) Bit 7 is Reserved Bit 6 indicates Standard or Extended CAN ID Bit 5 is Reserved Bits 4-0 is the CAN ID.
Byte 1	CAN ID
Byte 2	CAN ID
Byte 3	CAN ID (LSB)
Byte 4	Data Length: how many CAN Data bytes are valid.
Byte 5	CAN Data (LSB) Data transmitted in Little Endian format. Each field will have to be converted individually to Big Endian.
Byte 6	CAN Data
Byte 7	CAN Data
Byte 8	CAN Data
Byte 9	CAN Data
Byte 10	CAN Data
Byte 11	CAN Data
Byte 12	CAN Data (MSB)

The strategy receives a CAN packet as two strings:

- Bytes 0 – 4 are in the header string.
- Bytes 5 – 12 are in the data string. The data is converted from little Endian to big Endian and stored in an integer 32 table for each PGN at each address.

Examples:

- PGN 65285 address 1 header example: 54 FF 05 01 08
- PGN 65285 address 1 data example: 8C 80 FF 76 74 FF FF FF

Example Data in Table (PGN 65285)

PGN 65285 configured for 5 data fields produces a table similar to following. Position 3 is the needed data: Index 3 / 100 = 298.14 kelvin

Index 1 = 140
Index 2 = 65408
Index 3 = 29814
Index 4 = 65535
Index 5 = 255

The OptoScript for PGN 65285 is as follows:

```
// Convert data to Kelvin and Fahrenheit. PGN65285
fCTemperatureKelvin1 = ntCData65285a1[3];       // Convert the fixed point integer to
fCTemperatureKelvin1 = fCTemperatureKelvin1 / 100;
fCTemperatureFahrenheit1 = ((fCTemperatureKelvin1 - 273) * 1.8) + 32;

fCtTemperatureKelvin2 = ntCData65285a2[3];       // Convert the fixed point integer to
fCtTemperatureKelvin2 = fCTemperatureKelvin1 / 100;
fCTemperatureFahrenheit2 = ((fCTemperatureKelvin1 - 273) * 1.8) + 32;
```

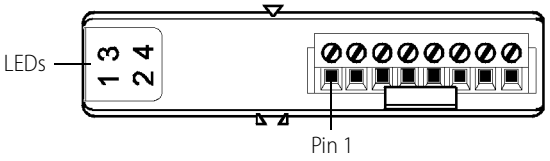
Module 1 Variables

The following table shows the variables for Module 1.

Variable	Description
nCNumber_of_CAN_Modules	number of CAN modules
nCModulePosition1	position on rack
sCModuleIPAddress1	IP address of I/O unit
pocCMMPDevice1	I/O unit with module
nCBaudRate1	module baud rate
nCPGNFilter0m1	PRN filter (highest priority)
nCPGNFilter1m1	PRN filter
nCPGNFilter2m1	PRN filter
nCPGNFilter3m1	PRN filter
nCPGNFilter4m1	PRN filter
nCPGNFilter5m1	PRN filter
nCPGNMask0m1	Mask for filter 0 and 1
nCPGNMask1m1	Mask for filter 2,3,4 and 5
nCPGNRecm1	PRN received
nCtSourceAddressRecm1	Source address
nCPortOpenStatus1	Open port status
nCCHRAtPortm1	Characters in buffer
nCModuleErrorCode1	module error codes
nCLogDatam1	1 = enable data logging

SNAP-SCM-CAN2B Module Pinouts and LEDs

The following illustration shows the location of pin 1 and the LEDs.



Module Pins

Pinout for the connector on the top of the SNAP-SCM-CAN2B module. Pins 1-4 are in parallel to pins 5-8. V+ is not used by the module.

Pin	Use
1,5	V +
2,6	CAN +
3,7	CAN –
4,8	GND

Module LEDs

LED	Type	Indicates
1	CAN Bus Activity	Communication activity with the CAN module. This LED illuminates while the module is being configured and when CAN data is received.
2	STATUS	2 blinks: SNAP-SCM-CAN2B firmware has started. 5 blinks: firmware error 8 blinks: CAN module error
3	POWER	Power is applied to the module.
4	ERROR	Error on the CAN bus. (For details, see next section.)

Error LED Operation. The #4 LED indicates an error on the CAN bus. Error codes include:

Error Codes	Description
0	Error—Active State. The SNAP-SCM-CAN2B has received less than 96 errors.
-1	Error—Active State. The SNAP-SCM-CAN2B has received 96 or more errors but less than 128 errors.
-2	Receiver Overflow. A CAN packet was dropped. This happens when the SNAP-SCM-CAN2B can't keep up with the traffic on the CAN bus. This means the internal buffer on the CAN2B module is full. This can happen if the strategy isn't reading the data fast enough, or too many serial modules are on the rack. To resolve it, you can: • Configure the Data Masks and Filters to receive fewer CAN packets. • Reduce the number of serial modules on the rack. • Increase how frequently the strategy reads the module.
-3	Error—Passive State. The SNAP-SCM-CAN2B has received 128 or more errors but less than 255 errors.

To clear the error, you can use PAC Manager or PAC Control to read the CAN module configuration from the memory map. (For the memory map addresses required to access the error codes, see form 1465, *OptoMMP Protocol Guide*, Appendix A: SNAP-SCM-CAN2B Serial Module Configuration-Read/Write.)

If your device has **SNAP-SCM-CAN2B Firmware R2.0b or higher**, the Error LED also illuminates when any bits are set in either the CAN bus error flags register or the CAN transport error flags register. For information on error flags, see the CAN RX/TX Integration Kit for PAC Project Technical Note, "CAN Module Status Reply."

In addition to clearing the error by reading from the memory map, with firmware R2.0b, you can also clear the error by issuing an "S" command from the O22Can2BModuleCtrlStat subroutine (in the CAN Integration Kit for PAC Project). For more information, see form 2151, the CAN RX/TX Integration Kit for PAC Project Technical Note, "PAC Control Sample Subroutines."