CAN RX/TX Integration Kit for PAC Project

Introduction

The SNAP-SCM-CAN2B is a high-speed serial communications module that can transmit to and receive data from devices on a Controller Area Network (CAN).

The Opto 22 CAN Integration Kit for PAC Project™ (part number PAC-INT-CAN-RXTX) provides sample PAC Control subroutines and charts so that your Opto 22 SNAP PAC System equipped with one or more SNAP-SCM-CAN2B modules can communicate with devices on the CAN network. When transmit functionality is enabled, all transmit and receive data between the module and the SNAP PAC brain or controller are ASCII encoded and frame delimited for robust communications.

The subroutines in the integration kit make it easier to communicate with devices wired to the CAN2B module by showing you the PAC Control commands to use and the format of data in them. The charts illustrate how to call the subroutines.

The sample subroutines run as part of the PAC Control strategy on an Opto 22 SNAP PAC controller. Using PAC Control, you can easily add to or modify the subroutines and charts to match your system's requirements.

The CAN Transport Protocol provides direct access to a SNAP-SCM-CAN2B which implements the CAN transport and physical layers. Access to higher layer protocols—such as J1939, NMEA 2000, CANopen, and DeviceNet—needs to be implemented by the customer.

Requirements

To use this integration kit, you should already know how to use PAC Control and how to use and configure a SNAP PAC controller and I/O units. You also need to understand how to use the CAN protocol.

You will need the following:

- A PC with PAC Project R9.2a or newer (Basic or Pro)
- A PAC Control strategy that includes a SNAP PAC I/O unit with one or more SNAP-SCM-CAN2B serial communication modules with firmware R2.0b or higher, 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 R9.2a or higher.
- An Opto 22 SNAP PAC R-series or S-series controller

Backward Compatibility

This is a new integration kit for SNAP-SCM-CAN2B modules with firmware R2.0b or higher. With this firmware the module can both send and receive CAN data; with earlier firmware the module only receives data.

You can update an existing module to the new firmware following steps in the *PAC Manager User's Guide,* form 1704.

For modules with firmware R1.0d and lower, a different integration kit (part number PAC-INT-CAN-RX, formerly PAC-INT-CAN) is available on the Opto 22 website. It provides sample charts for receiving data only.

About Subroutines

In PAC Control, a subroutine is like a custom command created to perform a sequence of steps. The same sequence can be performed whenever it's needed, just by calling the subroutine from within a chart in your strategy.

If you're not already familiar with subroutines, see Chapter 12 in the *PAC Control User's Guide* for important information on using them.

For Help

This technical note assumes that you understand how to use PAC Control™ and how to use and configure an Opto 22 SNAP PAC controller and I/O units. It also assumes that you understand how to use the CAN protocol.

If you have trouble installing or using the integration kit, you can call, fax, or email Opto 22 Product Support. Product Support is free.

Phone: 800-TEK-OPTO
(800-835-6786 toll-free in the U.S. and Canada)
951-695-3080
Monday through Friday,
7 a.m. to 5 p.m. Pacific Time

NOTE: Email messages and phone calls to Opto 22 Product Support are grouped together and answered in the order received.

Fax: 951-695-3017

Email: support@opto22.com

Opto 22 website: www.opto22.com

When calling for technical support, be prepared to provide the following information about your system to the Product Support engineer:

- Software and version being used
- Firmware versions for brains and controllers
- PC configuration (type of processor, speed, memory, and operating system)
- A complete description of your hardware and operating systems, including:
  - type of power supply
  - types of I/O units installed
  - third-party devices installed (for example, barcode readers)
- Specific error messages seen.

Related Documents

The following guides may also be helpful:

For this information...	See...	Form #
Install and wire SNAP-SCM-CAN2B modules	*SNAP-SCM-CAN2B Communication Module Data Sheet*	1537
Configure SNAP-SCM-CAN2B modules	*SNAP Serial Communication Module User's Guide*	1191
Install and use SNAP PAC S-series controller	*SNAP PAC S-series User's Guide*	1592
Install and use SNAP PAC R-series controller	*SNAP PAC R-series User's Guide*	1595
Develop a control strategy, configure modules, and communicate using commands	*PAC Control User's Guide*	1700
Test modules; alternative configuration tool	*PAC Manager User's Guide*	1704

Setting Up the Integration Kit on Your System

Getting Started

1. To get the free integration kit, go to opto22.com and search on PAC-INT-CAN-RXTX. Download the file and save it to the PC where your PAC Control strategy is located.

   The file is an archived strategy containing charts and subroutines.

2. Extract the file you downloaded to a folder you choose. Double-click `PAC-INT-CAN-RXTX-R1.0b.idb` to open the sample strategy in PAC Control.

   NOTE: the last part of the .idb filename is the release number, which may change.

3. When PAC Control asks you where the subroutines are located, navigate to the `Subs` subfolder inside the folder where you extracted the files.
4. Examine the integration kit charts and subroutines to see how they are used.

Including Subroutines in Your Strategy

To use a sample subroutine in your strategy, you include it in the strategy and then add it as a command so it can be called from a chart, as shown in the sample charts.

1. Choose the subroutines you need to use from the kit; see details starting on page 4.
2. With your strategy open in Configure mode, double-click the Subroutines Included folder on the Strategy Tree or select Configure > Subroutine Included.
3. Click Add. Navigate to the directory containing the subroutine and double-click the subroutine's name. Then click OK.

The new subroutine appears in the Strategy Tree in the Subroutines Included folder. You use a subroutine just like a PAC Control command: by adding the subroutine instruction to a block in a chart.

4. In your strategy, open the chart that will use the subroutine and double-click the block that will call the subroutine. Add the subroutine:
   - If it is an OptoScript block, list parameters (arguments) in order within parentheses. Be sure to use the subroutine's OptoScript name, using underscores instead of spaces. The Put Status In return value can be consumed by a variable (as shown below) or used in a mathematical expression or a control structure.

     ```
     nStatus = Variable_Increase_Notification
         ( bCondition, nValue, Output1 );
     ```

     For more information on OptoScript, see Chapter 11 in the *PAC Control User's Guide*.
   - If it is not an OptoScript block, enter the subroutine name or choose it from the dropdown list. Complete the parameters (Type, Name) based on the subroutine information starting on page 4.
5. When finished, click OK to return to the chart.

   Your chart is now set up to call the subroutine.

PAC Control Sample Subroutines

The following sample subroutines are included in the integration kit. To modify them, see "Building a CAN Transport Frame" on page 13.

IMPORTANT: *First use O22Can2BModuleCtrlStat to enable transmit functionality with ASCII encoding and frame delimiting. Then use other subroutines as needed for your system. (If you later send a new configuration with O22Can2BConfigMasksFiltersBaud, then resend the "k" command with O22Can2BModuleCtrlStat.)*

- O22Can2BModuleCtrlStat—Use transmit mode or read module status.
- O22Can2BConfigMasksFiltersBaud—Configure receive filtering. Use this subroutine before receiving data from a device on the CAN bus.
- O22Can2BParseModuleCtrlStatResp—Parse the frame received from the O22Can2BModuleCtrlStat subroutine.
- O22Can2BSend—Send data to a device on the CAN bus.
- O22Can2BRecv—Receive data from a device on the CAN bus.
- O22Can2BSendRecv—Send and receive data from a device on the CAN bus.
- O22Can2BParseCanTransportFrame—Parse the frame received from a O22Can2BRecv or O22Can2BSendRecv subroutine.

Form 2151-170607 • Opto 22 • 43044 Business Park Drive • Temecula, CA 92590-3614 • **www.opto22.com**
SALES 800-321-6786 • 951-695-3000 • FAX 951-695-3095 • sales@opto22.com • **SUPPORT** 800-835-6786 • 951-695-3080 • FAX 951-695-3017 • support@opto22.com

O22Can2BModuleCtrlStat

Description

CAN module control and status subroutine. Use this subroutine to request the SNAP-SCM-CAN2B module to perform one of two actions:

- Place the CAN module into transmit mode—uses the "k" command
- Read module status registers and other information—uses the "S" command

CAUTION: *Before you use this subroutine to read status ("S" command), first use O22Can2BRecv to store any data received from the CAN bus. Otherwise, any data stored in the module receive buffer that holds received CAN Frames from the bus will be discarded.*

Parameters

Type	Name	Details
String Literal OR String Variable	Command_Letter	k = Enable module transmit along with ASCII encoding and frame delimiting for both transmit and receive. S = Read CAN module status registers. See "CAN Module Status Reply" on page 16 for return string details.
String Variable	Response	A string length of 35 is ample space to store the value. Before using the Response variable check Status for success. This subroutine removes frame delimiting before returning the value.
Communication Handle	Comm_Handle	The communication handle must be of the format `tcp:ip address:port` For example: `tcp:127.0.0.1:22500` is the local controller's default serial port for the module at position 0.
Integer 32 Variable	Status	0 = Success -1 = Invalid Command_Letter -2 = Failure to open communication -3 = Improper Response format returned from module **Or** the status code for the PAC Control command TransmitString(), GetNumCharsWaiting(), or ReceiveNChars()

O22Can2BConfigMasksFiltersBaud

Use this subroutine first, before other subroutines, to set baud rate and configure receive masks and filters in the module. Alternatively, you can configure these in PAC Manager's Inspect window (Tools > Inspect > Communications > CAN Modules). Any changes made in your PAC Control strategy override the PAC Manager configuration.

This subroutine does not send or receive any CAN frames. It sends module configuration data to the SNAP PAC brain on the rack with the module.

For more on setting masks and filters for received CAN frames, see "Configuring CAN Modules" in the *SNAP Serial Communication Module User's Guide* (form 1191).

NOTE: Whenever you send a new configuration with O22Can2BConfigMasksFiltersBaud, also resend the "k" command with O22Can2BModuleCtrlStat.

Parameters

Type	Name	Details
Generic OptoMMP Device	OptoMMP	Configures an I/O unit as a Generic OptoMMP Device that is located on the same rack as the SNAP-SCM-CAN2B module
Integer 32 Literal OR Integer 32 Variable	Position	Position of the module on the rack. Values can be 0–15, inclusive
Integer 32 Literal OR Integer 32 Variable	Baud	Sets the baud rate of the CAN bus channel to a value in bps (bits per second). Valid values are 10000, 20000, 50000, 100000, 125000, 250000, 500000, and 1000000. Defaults to 250000 on powerup or reset of brain.
Integer 32 Literal OR Integer 32 Variable	Mask_0	Has higher priority than Mask_1. Is associated with Filters 0–1. Defaults to 0 on powerup or reset of brain.
Integer 32 Literal OR Integer 32 Variable	Filter_0	Has higher priority than Filter_1. Defaults to 0 on powerup or reset of brain.
Integer 32 Literal OR Integer 32 Variable	Filter_1	Has higher priority than Filter_2. Defaults to 0 on powerup or reset of brain.
Integer 32 Literal OR Integer 32 Variable	Mask_1	Has lower priority than Mask_0. Is associated with Filters 2–5. Defaults to 0 on powerup or reset of brain.
Integer 32 Literal OR Integer 32 Variable	Filter_2	Has higher priority than Filter_3. Defaults to 0 on powerup or reset of brain.
Integer 32 Literal OR Integer 32 Variable	Filter_3	Has higher priority than Filter_4. Defaults to 0 on powerup or reset of brain.
Integer 32 Literal OR Integer 32 Variable	Filter_4	Has higher priority than Filter_5. Defaults to 0 on powerup or reset of brain.
Integer 32 Literal OR Integer 32 Variable	Filter_5	Is lowest priority filter. Defaults to 0 on powerup or reset of brain.
Integer 32 Variable	Status	0 = Success -1 = Cannot communicate with OptoMMP device (brain) -2 = Position parameter invalid -3 = Baud parameter invalid -4 = Invalid mask and/or filter configuration

O22Can2BParseModuleCtrlStatResp

Description

Use this subroutine to parse the frame you receive after sending a command "S" with the subroutine O22Can2BModuleCtrlStat. This subroutine parses the frame for Baud, CAN Controller Flags, Transmit Error Counter, Receive Error Counter, and CAN Module Flags.

NOTE: To parse a received frame from command type "t", "T", "e", or "E" use the O22Can2BParseCanTransportFrame subroutine instead.

Parameters

See "CAN Module Status Reply" on page 16 for more details.

Type	Name	Details
String Variable OR String Literal	Frame	The frame without the frame delimiters
Integer 32 Variable	Baud	Returns the current baud value in bits per second, for example: `10000`
Integer 32 Variable	CAN Bus Error Flags	Returns the current CAN Controller Flags on the SNAP-SCM-CAN2B module. See "CAN Module Status Reply" on page 16.
Integer 32 Variable	CAN TEC	Returns the CAN Transmit Error Counter (the number of failed transmit attempts)
Integer 32 Variable	CAN REC	Returns the CAN Receive Error Counter (the number of failed receives)
Integer 32 Variable	CAN Transport Error Flags	Returns the current CAN Module (module firmware) Flags.
Integer 32 Variable	Status	0 = Success -1 = Empty frame -2 = Invalid frame type (not "S") No action is taken if you're trying to parse a "k" frame.

O22Can2BSend

Description

Use this subroutine to place a single CAN frame on the bus.

Only Command_Letters "t", "T", "e", or "E" may be used. (See "CAN Transport Commands" on page 15.) The module receives the command and formats the fields into a proper CAN frame for transmission on the bus.

CAUTION: *Any previously received data stored in the module receive buffer will be discarded. If you need to store the received data while sending a frame, use the O22Can2BSendRecv subroutine instead.*

Parameters

Type	Name	Details
String Literal OR String Variable	Command_Letter	Can be passed as a String Variable or String Literal: t = Standard Data T = Standard Remote e = Extended Data E = Extended Remote
Integer 32 Literal OR Integer 32 Variable	Arbitration	If the Command_Letter is `t` or `T` then the lower 11 bits of the value are used (Standard Frame type). If the Command_Letter is `e` or `E` then the lower 29 bits of the value are used (Extended Frame type).
Integer 32 Literal OR Integer 32 Variable	Data_Length	Value must be between 0 and 8, inclusive.
Integer 64 Literal OR Integer 64 Variable	Data	The data field of the CAN Frame. If passing an Integer 64 Literal, don't forget the "i64" at the end of the decimal or hexadecimal value, for example:`123456789i64` or `0x123456789ABCDEFi64` so the PAC Control compiler doesn't truncate the value to an Integer 32.
Communication Handle	Comm_Handle	The communication handle must be of the format `tcp:ip address:port` For example: `tcp:127.0.0.1:22500` is the local controller's default serial port for the module at position 0.
Integer 32 Variable	Status	0 = Success -1 = Invalid Command_Letter or Data_Length -2 = Failure to open communication **Or** the status code for the PAC Control command TransmitString()

O22Can2BRecv

Description

Use this subroutine to receive single or multiple CAN frames from the SNAP-SCM-CAN2B module. No data is transmitted on the CAN bus.

The subroutine strips out CAN transport frame delimiting and then places values in the string table stRecv_Frame starting at index 0. A numeric table (ntDuplicate_Frame) registers duplicate frames: if a duplicate frame is received, the value of the corresponding index in ntDuplicate_Frame is incremented.

The subroutine waits up to 1 second to allow partial frames to be completely received before exiting.

The maximum encoded frame length is 29 characters, so make sure that the stRecv_Frame element width is a minimum of 29. Also, match the lengths of the stRecv_Frame and ntDuplicate_Frame tables.

Parameters

Type	Name	Details
Communication Handle	Comm_Handle	The communication handle must be of the format `tcp:ip address:port` For example: `tcp:127.0.0.1:22500` is the local controller's default serial port for the module at position 0.
String Table	stRecv_Frame	Received frames are placed into this string table starting with element 0. Set table element width to a minimum of 29 and table length to a maximum of 910.
Integer 32 Table	ntDuplicate_Frame	Duplicate frames received are indicated in this table. Set table length to match the stRecv_Frame string table (maximum 910). If duplicate frames are received, the corresponding element in this numeric table is incremented.
Integer 32 Variable	Status	0 = Success -1 = Failure to open communication **Or** the status code for the PAC Control command GetNumCharsWaiting() or ReceiveNChars()

O22Can2BSendRecv

Description

Use this subroutine to first receive CAN frames from the SNAP-SCM-CAN2B module and then send a single CAN frame on the bus. You can receive a single frame or multiple frames.

Only Command_Letters "t", "T", "e", or "E" may be used. (See "CAN Transport Commands" on page 15.) For sending, the module formats the fields into a proper CAN frame for transmission on the bus after frames are received.

For received frames, the subroutine strips out CAN transport frame delimiting and then places values in the string table stRecv_Frame starting at index 0. A numeric table (ntDuplicate_Frame) registers duplicate frames: if a duplicate frame is received, the value of the corresponding index in ntDuplicate_Frame is incremented.

The subroutine waits up to 1 second to allow partial frames to be completely received before exiting.

The maximum encoded frame length is 29 characters, so make sure that the stRecv_Frame element width is a minimum of 29. Also, match the lengths of the stRecv_Frame and ntDuplicate_Frame tables.

Parameters

Type	Name	Details
String Literal OR String Variable	Command_Letter	Can be passed as a String Variable or String Literal: t = Standard Data T = Standard Remote e = Extended Data E = Extended Remote
Integer 32 Literal OR Integer 32 Variable	Arbitration	If the Command_Letter is `t` or `T` then the lower 11 bits of the value are used (Standard Frame type). If the Command_Letter is `e` or `E` then the lower 29 bits of the value are used (Extended Frame type).
Integer 32 Literal OR Integer 32 Variable	Data_Length	Value must be between 0 and 8, inclusive.
Integer 64 Literal OR Integer 64 Variable	Data	The data field of the CAN Frame. If passing an Integer 64 Literal, don't forget the "i64" at the end of the decimal or hexadecimal value, for example:`123456789i64` or `0x123456789ABCDEFi64` so the PAC Control compiler doesn't truncate the value to an Integer 32.
String Table	stRecv_Frame	Received frames are placed into this string table starting with element 0. Set table element width to a minimum of 29 and table length to a maximum of 910.

Type	Name	Details
Integer 32 Table	ntDuplicate_Frame	Duplicate frames received are indicated in this table. Set table length to match the stRecv_Frame string table (maximum 910). If duplicate frames are received, the corresponding element in this numeric table is incremented.
Communication Handle	Comm_Handle	The communication handle must be of the format `tcp:ip address:port` For example: `tcp:127.0.0.1:22500` is the local controller's default serial port for the module at position 0.
Integer 32 Variable	Status	0 = Success -1 = bad Command_Letter and/or Data_Length -2 = Failure to open communication -3 = Failed Communication Open? (Check after receiving frames.) **Or** the status code for the PAC Control command TransmitString(), GetNumCharsWaiting(), or ReceiveNChars()

O22Can2BParseCanTransportFrame

Description

Use this subroutine after receiving frames from O22Can2BRecv or O22Can2BSendRecv to parse the frame for the Arbitration, Data Length, and Data fields.

NOTE: To parse a received frame from a command "S" use O22Can2BParseModuleCtrlStatResp instead.

Parameters

Type	Name	Details
String Variable OR String Literal	Frame	The frame without the frame delimiters
Integer 32 Variable	Arbitration	If the CAN frame type is Standard, the lower 11 bits of the value are valid. If the CAN frame type is Extended, the lower 29 bits of the value are valid.
Integer 32 Variable	Data_Length	Returns the data length value
Integer 64 Variable	Data	Returns the data field value of the CAN frame
Integer 32 Variable	Status	0 = Success -1 = Empty frame -2 = Invalid frame type (not "t", "T", "e", or "E")

Building a CAN Transport Frame

NOTE: Partial, single, or multiple frames can be sent or received in one TCP send/receive call.

CAN Transport Frame Contents

Start-of-Frame Delimiter

Always ASCII > character: signals the beginning of a new CAN frame.

Command Letter

An 8-bit field encoded as 1 ASCII character (non-ASCII hex). See "CAN Transport Commands" on page 15.

A command letter may specify a Standard, Extended, or Remote Frame; flow control; error codes; or even configuration. The command letter specifies what fields are required.

Arbitration

Use either the 11-bit Arbitration field OR the 29-bit Arbitration field:

11-bit Arbitration Field for Standard Frames, Std-Arb—a 16-bit field encoded as 4 ASCII hex characters.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit Field	0	0	0	0	0	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	SID2	SID1	SID0

SIDn = Standard ID bits

29-bit Arbitration Field for Extended Frames, Ext-Arb—a 32-bit field encoded as 8 ASCII HEX characters.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit Field	0	0	0	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	SID2	SID1	SID0	EID17	EID16	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0

SIDn = Standard ID bits; EIDn = Extended ID bits

Data Length Code (DLC)

An 8-bit field encoded as 2 ASCII HEX characters. This length field specifies the number of bytes of data in the frame (not the number of ASCII hex characters used to represent the data). Under the current CAN protocol this is 0 to 8 bytes, inclusive.

Bit	7	6	5	4	3	2	1	0
Bit Fields	0	0	0	0	DLC3	DLC2	DLC1	DLC0

DLCn = Data Length Code bits

Data Field

Each data byte is encoded as 2 ASCII hex characters. For example, a 2-byte payload is 4 ASCII hex characters; an 8-byte payload is 16 ASCII hex characters.

Byte Position:								
0 to 8 bytes, inclusive	0	1	2	3	4	5	6	7

End-of-Frame Delimiter

Always ASCII `cr` (carriage return, 13 decimal or 0x0D hex). Signals the end of a CAN frame.

CAN Transport Frame Examples

Send or receive a Standard Data Frame

Arbitration ID = 0x015A

Data Length = 6

Data payload = 0x23456789ABCD

Frame: `>t015A0623456789ABCDcr`

Send or receive a Standard Remote Frame requesting 4 bytes of data

Arbitration ID = 0x015A

Frame: `>T015A04cr`

Send or receive an Extended Data Frame

Arbitration ID = 0x015A36FF

Data Length = 8

Data payload = 0x0123456789ABCDEF

Frame: `>e015A36FF080123456789ABCDEFcr`

Send or receive an Extended Remote Frame requesting 2 bytes of data

Arbitration ID = 0x015A36FF

Frame: `>E015A36FF02cr`

Read CAN Controller Status

Send Frame: `>Scr`

Response Frame: `>S503006900cr`

Which means:

- The current CAN baud is 250k bps
- RXWARN Flag is set (also EWARN Flag is set)
- TEC = 0

- REC = 105
- No CAN Module Error Flags are set

CAN Transport Commands

Command Name	Description	ASCII Char	Format	Expected Reply
Standard Data Frame	Sends or receives a standard data frame	t	>t[Std-Arb][DLC][Data]cr	None
Standard Remote Frame	Sends or receives a standard remote frame	T	>T[Std-Arb][DLC]cr	Standard Data Frame from remote CAN device
Extended Data Frame	Sends or receives an extended data frame	e	>e[Ext-Arb][DLC][Data]cr	None
Extended Remote Frame	Sends or receives an extended remote frame	E	>E[Ext-Arb][DLC]cr	Extended Data Frame from remote CAN device
Read CAN Controller Status	Reads the CAN Controller Error Flags	S	>Scr	>Sabbccddeecr S = Command Letter a = CAN Controller Baud Rate bb = CAN Controller Error Flags cc = Transmit Error Counter, TEC dd = Receive Error Counter, REC ee = CAN Module Error Flags
Enable Module Frame Delimiting	Enables Frame Delimiting for both Tx and Rx	k	>kcr	>kcr if module supports Tx capability along with frame delimiting

CAN Module Status Reply

CAN controller baud rate is encoded as one ASCII hex character:

ASCII	Baud rate
0	10k bps
1	20k bps
2	50k bps
3	100k bps
4	125k bps

ASCII	Baud rate
5	250k bps
6	500k bps
7	reserved
8	1000k bps

CAN bus error flags are shown as two ASCII hex characters:

Bit	7	6	5	4	3	2	1	0
Bit Fields	RX1OVR Receive Buffer 1 overflow Flag	RX0OVR Receive Buffer 0 overflow Flag	TXBO Bus-Off Error Flag Set when TEC reaches 255.	TXEP Transmit Error-Passive Flag Set when TEC is greater than or equal to 128.	RXEP Receive Error-Passive Flag Set when REC is greater than or equal to 128.	TXWARN Set when TEC is greater than or equal to 96.	RXWARN Set when REC is greater than or equal to 96	EWARN Set when TEC or REC is greater than or equal to 96.

CAN transport error flags are shown as two ASCII HEX characters:

Bit	7	6	5	4	3	2	1	0
Bit Fields	0	0	0	CAN CAN Arbitration, Data Length Code, and/or Data framing error.	ASCII CAN Arbitration, Data Length Code, and/or Data Field has Non-ASCII Hex characters.	FRMG Opto Transport for CAN Framing errors (SOF, EOF, Frame Type).	TXFIFOOVR Transmit FIFO overflow Flag SNAP-SCM-CAN2 B Tx Buffer Full.	RXFIFOOVR Receive FIFO overflow Flag SNAP-SCM-CAN2 B Rx Buffer Full.

Form 2151-170607 • Opto 22 • 43044 Business Park Drive • Temecula, CA 92590-3614 • **www.opto22.com**
SALES 800-321-6786 • 951-695-3000 • FAX 951-695-3095 • sales@opto22.com • **SUPPORT** 800-835-6786 • 951-695-3080 • FAX 951-695-3017 • support@opto22.com