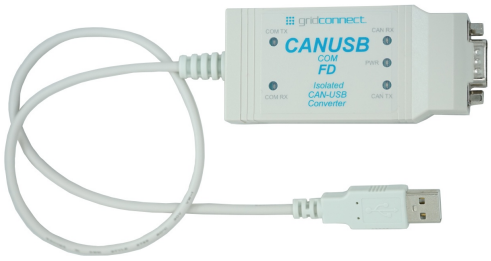
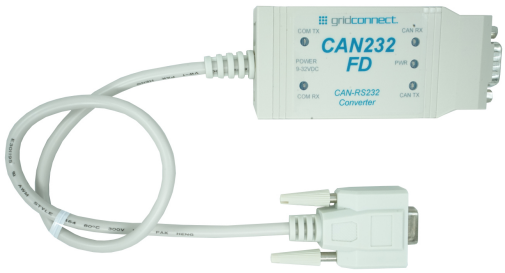
gridconnect™

CAN USB-232 FD
User Manual





Revision D May 18, 2021

Copyright and Trademark

Grid Connect

1630 W. Diehl Rd.
Naperville, IL 60563, USA
Phone: 630.245.1445

Technical Support

Phone: 630.245.1445
Fax: 630.245.1717
On-line: www.gridconnect.com

Disclaimer and Revisions

Operation of this equipment in a residential area is likely to cause interference in which case the user, at his or her own expense, will be required to take whatever measures may be required to correct the interference.

> ***Attention:*** *This product has been designed to comply with the limits for a Class B digital device pursuant to Part 15 of FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses, and can radiate radio frequency energy, and if not installed and used in accordance with this guide, may cause harmful interference to radio communications.*

Changes or modifications to this device not explicitly approved by Grid Connect will void the user's authority to operate this device.

The information in this guide may change without notice. The manufacturer assumes no responsibility for any errors that may appear in this guide.

Date	Rev.	Author	Comments
10/30/2019	A	EDL	Preliminary Release
01/13/2020	B	EDL	Updated pictures and diagrams
09/25/2020	C	EDL	Added the CANUSB FD model
05/18/2021	D	EDL	Updated pictures

Warranty

Grid Connect warrants each product to be free from defects in material and workmanship for a period of ONE YEAR after the date of shipment. During this period, if a customer is unable to resolve a product problem with Grid Connect Technical Support, a Return Material Authorization (RMA) will be issued. Following receipt of a RMA number, the customer shall return the product to Grid Connect, freight prepaid. Upon verification of warranty, Grid Connect will -- at its option -- repair or replace the product and return it to the customer freight prepaid. If the product is not under warranty, the customer may have Grid Connect repair the unit on a fee basis or return it. No services are handled at the customer's site under this warranty. This warranty is voided if the customer uses the product in an unauthorized or improper way, or in an environment for which it was not designed.

Grid Connect warrants the media containing software and technical information to be free from defects and warrants that the software will operate substantially for a period of 60 DAYS after the date of shipment.

In no event will Grid Connect be responsible to the user in contract, in tort (including negligence), strict liability or otherwise for any special, indirect, incidental or consequential damage or loss of equipment, plant or power system, cost of capital, loss of profits or revenues, cost of replacement power, additional expenses in the use of existing software, hardware, equipment or facilities, or claims against the user by its employees or customers resulting from the use of the information, recommendations, descriptions and safety notations supplied by Grid Connect. Grid Connect liability is limited (at its election) to:

1) refund of buyer's purchase price for such affected products (without interest)

2) repair or replacement of such products, provided that the buyer follows the above procedures.

There are no understandings, agreements, representations or warranties, expressed or implied, including warranties of merchantability or fitness for a particular purpose, other than those specifically set out above or by any existing contract between the parties. The contents of this document shall not become part of or modify any prior or existing agreement, commitment or relationship.

Table of Contents

1. Overview

1.1 Introduction

The CAN USB-232 FD enables the use of an RS232 or USB COM port to send and receive CAN messages on an ISO11898 CAN or CAN FD network.

The CAN USB-232 FD can be configured to operate in either the serial command mode or the serial tunnel mode, providing the user with a choice as to the type of functionality desired.

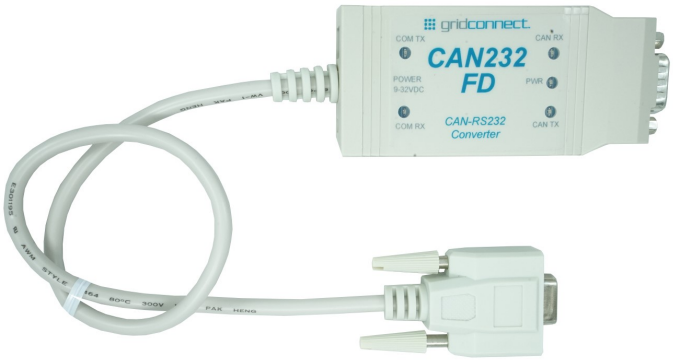
In the command mode, the CAN USB-232 FD is capable of sending and receiving arbitrary CAN messages through the serial port, providing a complete CAN-to-232 interface. This mode also supports message filtering to limit the range of CAN identifiers and the number of CAN messages that will be received.

In Tunnel mode, the CAN USB-232 FD establishes a full-duplex virtual serial link between itself and an application on another CAN node. A serial device can transparently send and receive serial stream data on the CAN USB-232 serial line and the CAN USB-232 will route the data over CAN to the configured target device. In previous versions of the product this feature was referred to as virtual circuit (VC) mode.

The CAN USB-232 FD operating mode and all other configuration parameters are accessed through the 'config' button. Pressing the button causes the CAN USB-232 FD to enter configuration mode where it then prompts the user to modify configuration parameters. In addition to parameter configuration, the configuration mode also provides diagnostic tools.

The CAN USB-232 FD represents an evolution of the earlier CAN USB-232 product line, providing a performance increase and enhanced features, while retaining much backward compatibility.

The CAN232 FD version is shown below.



The new units are based on an ARM Cortex-M0+ 32-bit microcontroller. All device interfaces (UART, USB, CAN) have built-in hardware buffers that allow for high-speed data flow without data loss. Hardware buffering, coupled with a fast and efficient CPU, allows each interface to operate at its maximum rated limit.

The CANUSB FD interface has been designed with a USB to parallel FIFO bidirectional data transfer interface. The USB driver software provides the host with a serial COM port interface, but without dependence on matching serial baud rate and data format. The parallel data transfer also provides higher performance than serial line data transfers.

The CANUSB FD version is shown below.

The new CAN USB-232 FD model is now equipped with an isolated CAN interface standard.

1.2 Hardware Description

The following drawing shows the location and function of the LEDs.

1.3 Model Description

There are two hardware platforms, both of which execute similar firmware, with one providing a CAN-RS232 interface and the other a CAN-USB interface. The CAN232 FD platform has the option of DTE (male) or DCE (female) on the RS232 serial cable. Both platforms have a Male DB9 CAN connector.

1.4 CAN Network

Before trying to send commands to the CAN USB-232 FD, make sure the unit is configured for and attached to a valid CAN network with at least one other node attached and running. Without another node on the network, the CAN USB-232 FD will attempt to transmit the message indefinitely (as per CAN 2.0A/B specifications).

It is very important that all nodes on a CAN FD network have the same sample point as this dictates when a bit rate switch occurs.

Make sure there are terminating resistors on the network. A terminating resistance of 120 ohms is appropriate for ISO11898 drivers. Attempts to communicate over the CAN network without terminating resistors can lead to erratic behavior and many long hours of trouble shooting.

CAN ISO11898 specifies a three wire bus: CAN_H, CAN_L, and GROUND. Failure to provide a common ground between network nodes will create some weird behavior. At best, the system will appear to communicate correctly until either the weather changes, you touch something, or the date changes. CAN_H and CAN_L form a differential pair and each one must be referenced to a common ground in order to work properly.

PIN	Function
2	CAN-L
3	CAN-Ground
7	CAN-H

2. Getting Started

2.1 Hardware Requirements

You will need an RS232 serial port to communicate with the CAN232 FD model.

You will need an open USB port to power and communicate with the CANUSB FD model.

2.2 Software Installation

2.2.1 TeraTerm Install

You will need terminal emulation software in order to configure the CAN USB-232 FD device over the COM port interface. For Windows platforms we recommend the Tera Term terminal emulator. You can download the software using the following link: Tera Term Install.

2.2.2 USB Driver Install

USB driver installation is not required for the CAN232 FD model. The USB driver for the CANUSB FD model will likely install automatically when connecting the device to the host USB port. If needed the royalty free VIRTUAL COM PORT (VCP) DRIVERS for Windows, Mac OS, OS-X and Linux are all available to download for free from FTDI website (www.ftdichip.com). Various 3rd party drivers are also available for other operating systems - see FTDI website (www.ftdichip.com) for details.

For driver installation, please refer to http://www.ftdichip.com/Documents/InstallGuides.htm

2.3 Entering Configuration Mode

2.3.1 RS232 Interface:

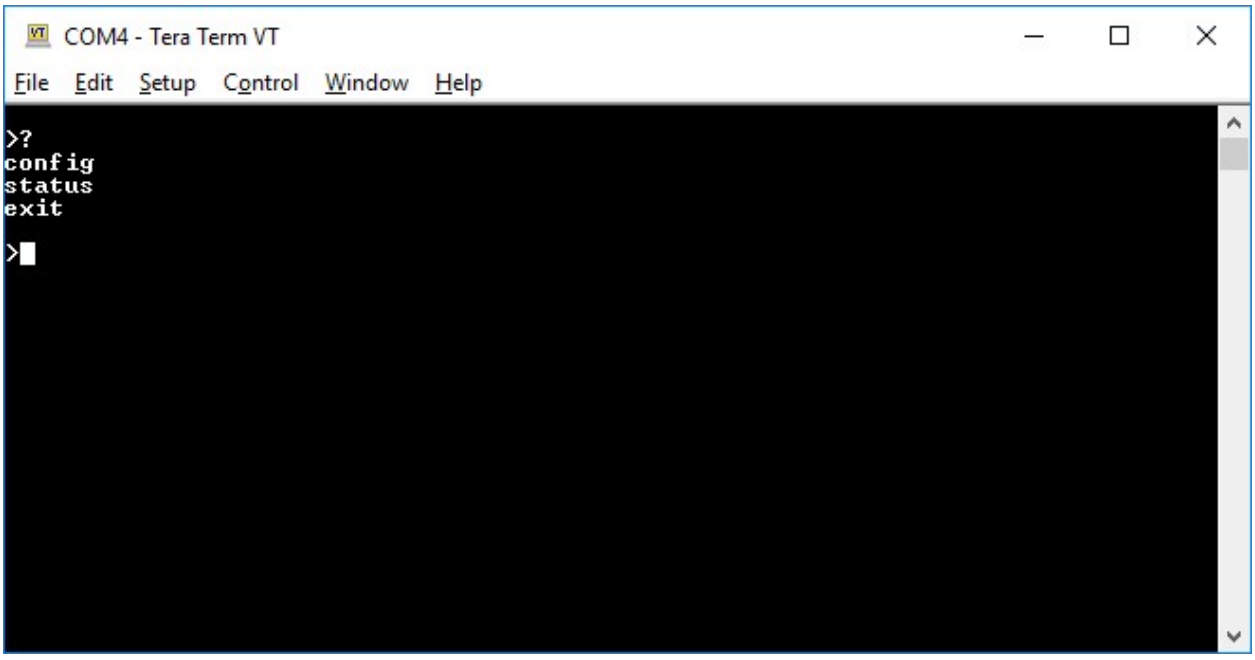
To view the Configuration menu, first connect the CAN232 FD serial cable to a PC running a monitor program, such as Tera Term. Set the COM port parameters to 115200, 8, 1, None. The cable should be a null-modem if the CAN232 FD is wired as DTE.

2.3.2 USB Interface:

Use Device Manager to determine which COM port the CANUSB FD module was assigned. Device Manager will show the device as a USB Serial Port. To view the Configuration menu, first connect the CANUSB FD cable to a PC running a monitor program, such as Tera Term. Set the connection to the assigned COM port number and serial parameter to any baud, 8, 1, None.

2.3.3 Config Button

Press the Config button (for less than 3 seconds) located next to the cable. The config prompt should appear as **'>'**. Type **'help'** or **?** to display the possible commands and settings for each level. Also see Section 4 Configuration Mode for navigation help and descriptions of all configuration options.



Whenever the CONFIG button is pressed, the COM port settings are automatically set to 115200, 8, N, 1 with no flow control in case the user forgets the configured settings.

2.3.4 Serial Command

When using Command mode it's possible to enter configuration mode using the proper command. In this case the serial parameters are left at their configured settings. This ensures that an application can smoothly change from RUN to CONFIG and back to RUN mode without needing to change COM port settings.

2.4 Hardware Installation

2.4.1 Configuration Push Button

Pressing the recessed push button for **less than 3 seconds** will put the processor into configuration mode.

Pressing the recessed push button for **MORE than 3 seconds** will reset the unit to factory defaults.

See the Configuration Mode chapter for detailed operation.

2.4.2 RS232 Serial Cable

The table below lists the serial line data and control signals for the CAN232 FD. The RS232 interface is a 9-pin D-style connector. Male connectors are wired as DTE and female connectors are wired as DCE.

CAN232 FD Signal	Direction (J8)	DTE DB-9 Male Pin #	DCE DB-9 Female Pin #
Data Out (TXD)	Out (3)	3	2
Data In (RXD)	In (4)	2	3
Ground	(1)	5	5
CTS	In (6)	8	7
RTS	Out (5)	7	8
No Connection	(8)	6	4
No Connection	(9)	4	6
+9-32Vdc	In (10)	9	9
No Connection	(7)	1	1
Frame Ground	(2)	Shield	Shield

Units can be specially ordered with TTL level signals, which are connected in the same manner as the RS232 versions.

Pin 9 on the DB-9 connector is shown with +9-32 VDC going to Pin 10 of J8. This is an optional method of supplying power to the device. It is not required if you use a different power source. For more details about power, see Power Options below.

2.4.3 Power Options

For the CAN-232 model, power (+9-32 VDC) can be supplied through a Phoenix terminal block, through a barrel jack, through the serial cable (usually pin 9), or through the OEM option connector on the circuit board. The factory default configuration is the Barrel Jack.

For the CAN-USB version, power is supplied by the USB connection.

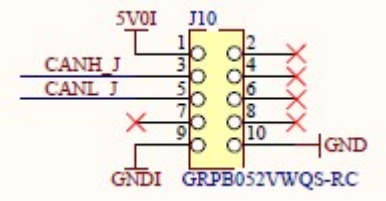
2.4.4 Input Power

The input power range is from 9-32VDC for the CAN-232 model.

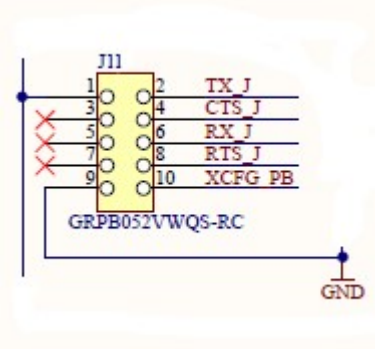
2.4.5 OEM Options

The OEM version comes without a RS232 cable. It is designed to be part of an embedded product. Instead of a cable connector, a 10-pin header is used to connect the signals to an OEM's circuit board.

The CAN signals are brought out to the OEM connector at J10. The OEM connector is a 10-pin header soldered to the bottom side of the board.



The RS232 signals are brought out to the OEM connector at J11. Power can be supplied through pin 1 and Ground is tied to pin 9.



2.4.6 CAN Driver

The TCAN1042 is a CAN transceiver that meets or exceeds the specifications of the ISO11898-2 (2016) High Speed CAN (Controller Area Network) physical layer standard. The device is designed for use in CAN FD networks up to 5 Mbps (megabits per second). Additionally, the device includes many protection features to enhance device and network robustness.

The TCAN1042 is characterized for operation over the ambient temperature range of –40°C to 125°C.

- Failsafe Outputs
- Meets or Exceeds ISO 11898-2 requirements
- Bus Fault Protection: ±58 V
- Dominant Time-Out Function
- IEC ESD Protection up to ±15 kV

DOMINANT TIME-OUT

A dominant time-out circuit in the TCAN1042 prevents the driver from blocking network communications if a local controller fault occurs. The time-out circuit is triggered by a falling edge on TXD. If no rising edge occurs on TXD before the time-out of the circuits expires, the driver is disabled to prevent the local node from continuously transmitting a Dominant bit. If a rising edge occurs on TXD, commanding a Recessive bit, the timer will be reset and the driver will be re-enabled. The time-out value is set so that normal CAN communication will not cause the Dominant time-out circuit to expire.

FAILSAFE

The supply terminals have undervoltage detection that places the device in protected mode. This protects the bus during an undervoltage event on either of the supply terminals.

THERMAL SHUTDOWN

The TCAN1042 has an internal thermal shutdown circuit that turns off the driver outputs when the internal temperature becomes too high for normal operation. This shutdown circuit prevents catastrophic failure due to short-circuit faults on the bus lines. If the device cools sufficiently after thermal shutdown, it will automatically re-enable, and may again rise in temperature if the bus fault is still present. Prolonged operation with thermal shutdown conditions may affect device reliability.

BUS LOADING

The TCAN1042 family is specified to meet the 1.5 V requirement with a 50Ω load, incorporating the worst case including parallel transceivers. The differential input resistance of the TCAN1042 family is a minimum of 30 kΩ. If 100 TCAN1042 family transceivers are in parallel on a bus, this is equivalent to a 300Ω differential load worst case. That transceiver load of 300 Ω in parallel with the 60Ω gives an equivalent loading of 50 Ω. Therefore, the TCAN1042 family theoretically supports up to 100 transceivers on a single bus segment. However, for CAN network design margin must be given for signal loss across the system and cabling, parasitic loadings, network imbalances, ground offsets and signal integrity thus a practical maximum number of nodes is typically much lower.

2.4.7 TTL Option

This CAN-232 model option allows a user to connect TTL level signals to the board through the serial interface. This option is by special order and NOT a field option.

3. Operation

The following sections describe the operation of the CAN USB-232 FD.

3.1 CAN Command Mode

In the command mode, the CAN USB-232 FD is capable of sending and receiving arbitrary CAN messages via the use of ASCII or Binary formatted message strings (Figure 3).

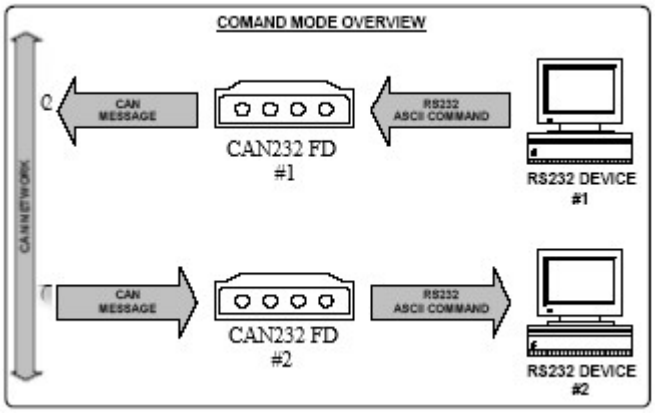


Figure 3

When the CAN USB-232 FD receives a valid command message on the serial line, it converts it to a CAN message and transmits it out over the CAN network.

Conversely, when a CAN message is received by the CAN USB-232 FD, it converts it to a command message and transmits it out of the serial port.

The CAN USB-232 FD supports ten (10) CAN receive message filters each for standard IDs and Extended IDs consisting of individual IDs, ID ranges or classic ID masks. By using the masks to specify which bits of an identifier are to be compared to the filter value, the CAN USB-232 FD is capable of selecting an arbitrary sub-set of the total possible CAN messages and rejecting all others. Thus, only desired messages will be received and the total required bandwidth of the serial link is kept to a minimum.

3.1.1 ASCII Message String Syntax

Message strings are formatted as human-readable ASCII sequences that are easy to enter and read. Each message string is of variable length, depending on the number of data bytes included in the message.

In order to facilitate human CAN network monitoring, there is an option to append a CR/LF sequence to each output ASCII message string. Doing so makes it much easier to watch the incoming messages on a terminal where each message is on a separate line. See Appending CR/LF To Received Command Strings on page 3-13.

Messages starting with '|' will generate a self-receive of the transmitted message. If the message is transmitted on CAN successfully, it will be received back on the serial line as if it had been sent from some other node.

Terminating with '!' instead of ';' indicates the intent is a one-shot transmit. This command option is maintained in the firmware for backward compatibility with previous versions. A one-shot message is only

accepted when the **mode** option under **command** settings in configuration is set to **one-shot**. In this mode, no attempts to automatically retry on error will happen. This is used in time-triggered CAN protocols.

All message string characters are in upper case only. Lower case characters will be interpreted as a syntax error and the message will be discarded.

Identifier and data fields are treated as base-16 digits (hexadecimal).

The syntax of both transmit and receive message strings are identical.

There are three types of message strings:

- Normal CAN messages

- Remote Transmission Request CAN messages (RTR)

- Report CAN error status messages (optional)

As per the CAN 2.0A/B specification, RTR messages do not contain data. To support RTR messages, the syntax is modified to include an explicit single-digit length. The length field is treated as a base-10 digit and must be a single digit between 0 and 8. The CAN FD specification does not support Remote frames.

3.1.1.1 Normal CAN Message

A normal CAN message consists of the identifier type (11-bit or 29-bit), identifier, CAN type, and data bytes and is encoded as follows:

Normal CAN Message Syntax

`: <S | X> <IDENTIFIER> <N | F | H> <DATA-0> <DATA-1> … <DATA-n> ;`

The first character,':', is for synchronization and allows the CAN USB-232 FD parser to detect the beginning of a command string.

The following character is either 'S' for standard 11-bit, or 'X' for extended 29-bit identifier type.

The 'IDENTIFIER' field consists of three or eight hexadecimal digits, indicating the value of the identifier. Note that if a 29-bit value is entered and an 11-bit value was specified, the command will be treated as invalid and ignored.

The character 'N' indicates that the message is a normal (non-RTR) CAN 2.0A/B transmission. The character 'F' or 'H' indicates that the message is a normal CAN FD transmission without or with the higher data baud rate switch (BRS) respectively.

Each 'DATA-n' field is a pair of hexadecimal digits defining the data byte to be sent. If no data is to be sent (length = zero), then the data bytes are omitted. CAN FD messages can be up to 64 data bytes in length where only lengths 0 – 8, 12, 16, 20, 24, 32, 48 and 64 are valid based on the standard.

Each data byte specified must be a hexadecimal pair in order to eliminate ambiguity.

The terminating character ';' signals the end of the message.

Examples

Example #1

```
:S123N12345678;
```

This message string indicates a 11-bit identifier whose value is $123, is normal, and has four data bytes: $12 $34 $56 and $78.

Example #2

```
:XF00DN;
```

This message string indicates a 29-bit identifier whose value is $F00D, is normal, and has zero data bytes.

Example #3

```
:X12345678H0102030405060708090A0B0C;
```

This message string indicates a 29-bit identifier whose value is $12345678, is normal CAN FD with BRS, and has twelve data bytes: $01 $02 $03 $04 $05 $06 $07 $08 $09 $0A $0B and $0C.

3.1.1.2 RTR CAN Message

A RTR CAN message consists of the type (11-bit or 29-bit), identifier, CAN type, length, does not have any data bytes, and is encoded as follows:

RTR CAN Message Syntax

```
: <S | X> <IDENTIFIER> <R> <LENGTH> ;
```

The first character ,':', is for synchronization and allows the CAN USB-232 FD parser to detect the beginning of a command string.

The following character is either 'S' for standard 11-bit, or 'X' for extended 29-bit identifier type.

The 'IDENTIFIER' field consists of from one to eight hexadecimal digits, indicating the value of the identifier. Note that if a 29-bit value is entered and an 11-bit value was specified, the command will be treated as invalid and ignored.

The character 'R' indicates that the message is a RTR transmission. The CAN FD specification does not support Remote frames.

The 'LENGTH' field is a single ASCII decimal character from '0' to '8' that specifies the length of the data being requested.

The terminating character ';' signals the end of the message.

Examples

Example #1

```
:S123R8;
```

This message string indicates a 11-bit identifier whose value is $123, is RTR, and has length 8.

Example #2

```
:XF00DR0;
```

This message string indicates a 29-bit identifier whose value is $F00D, is RTR, and has length 0.

3.1.1.3 Report Error Status Message

A Report Error Status message indicates changes in the current CAN bus error status detected by the CAN controller. This message is only transmitted if the **report err** feature is enabled under the **command** settings. By default this feature is disabled.

Report CAN Error Status Message Syntax

: <E> <A | W | P | B> ;

The first character ‚:', is for synchronization and allows the receiver's parser to detect the beginning of a command string.

The character 'E' indicates that this is a Report Error Status message.

The following character represents the new CAN error status.

'A' – Active (actively flagging errors on the bus, CAN operation is normal)
'W' – Warning (error count above the warning threshold)
'P' – Passive (only passively flagging errors on the bus, too many errors detected)
'B' – Bus Off (no longer transmitting on the CAN bus)

The terminating character ';' signals the end of the message.

3.1.1.4 Appending CR/LF To Received Command Strings

When using the CAN USB-232 FD to view received CAN messages directly on a terminal, the user can set a configuration parameter to append a <CR> <LF> sequence to each message string generated. This makes it easier for the user to see each received message as each message will be on a separate line. See the **eol** option under the **command** settings.

3.1.1.5 Timestamp

Outgoing serial command messages can have a 16-bit time-stamp appended to them with a 1-ms resolution. In ASCII mode, the timestamp is appended to the data block in uppercase HEX ASCII in the same way that the data field is presented (i.e.: two ASCII HEX digits to represent a byte). The '@' character is inserted between the last DATA digit and the first TIMESTAMP digit to differentiate between the two fields.

Zero-length messages appear as they would normally, except the prefixed timestamp is added.

Ex 1:S12N12@F00F: This is a 11-bit identifier with 1 data byte and timestamp of $F00F ms.

Ex 2:X13N@2EDF: This is a 29-bit identifier with 0 data bytes and timestamp of $2EDF ms.

Ex 3:S14R5@15E5: This is a 11-bit identifier with RTR=5 and timestamp of $15E5 ms.

3.1.2 Binary Formatted Messages

Transmitting CAN messages over the serial line in binary is more efficient because the data can be transmitted as-is without converting each byte to two ASCII characters in hex. However, it's also a little more complicated to frame the message since any binary value that marks the start of a message may also appear in the data. This binary protocol makes use of DLE sequences and byte stuffing to uniquely frame messages.

A binary message always begins with the 2-byte DLE SYNC sequence $FF $00. If this sequence happened to occur in the middle of the binary message, it would incorrectly signal to the receiver that a new message

was starting. To ensure the SYNC sequence does not occur in the middle of the message, the transmitter must scan the message for the binary value $FF and replace it with the 2-byte DATA sequence $FF $01. The $01 byte is the stuff byte. Now if a ..$FF $00.. pair occurs inside the message then it will be translated into $FF $01 $00 by the byte stuffing. The receiver must convert the DLE DATA sequence back to a single byte $FF by removing the $01 byte.

The following table defines all the valid DLE sequences implemented in the protocol. Any other DLE sequence received should be considered an error and will cause the message to be discarded.

DLE Sequences Defined

Name	Byte-0	Byte-1	Type	Comment
SYNC	$FF	$00	FRAME	Marks start of message
DATA	$FF	$01	DATA	Defines data value of $FF
CONFIG	$FF	$02	COMMAND	Configuration mode entry command
REPORT	$FF	$03	STATUS	Report CAN error status change
DLE RE-SYNC	$FF	$FF	ERROR	Restarts DLE sequence parsing

After the DLE SYNC sequence to start the message, a CAN message header format is defined, followed by the CAN data values. Note any of the header or data bytes might contain the $FF value and must be scanned to perform the byte stuffing.

3.1.2.1 CAN 2.0A/B Messages

The tables below show the Binary message format for CAN 2.0A/B messages.

Note: The ONE-SHOT bit in earlier versions has been replaced with the FD bit. Set the **mode** option to **one-shot** under the **command** settings in configuration to continue using the one-shot functionality. Also, when FD=0, then BIT-6 is interpreted as the RTR flag.

STD-NORMAL Message Format

	BIT-7	BIT-6	BIT-5	BIT-4	BIT-3	BIT-2	BIT-1	BIT-0	
Byte-0	1	1	1	1	1	1	1	1	SYNC
Byte-1	0	0	0	0	0	0	0	0	
Byte-2	EXT=0	RTR=0	FD=0	SELF-RCV	LENGTH				TYPE
Byte-3	0	0	0	0	0	ID (10 -8)			ID
Byte-4	ID (7-0)								
Byte-5	DATA-0								DATA
Byte-6	DATA-1								
Byte-7	DATA-2								
Byte-8	DATA-3								
Byte-9	DATA-4								
Byte-10	DATA-5								
Byte-11	DATA-6								
Byte-12	DATA-7								

STD-RTR Message Format

	BIT-7	BIT-6	BIT-5	BIT-4	BIT-3	BIT-2	BIT-1	BIT-0	
Byte-0	1	1	1	1	1	1	1	1	SYNC
Byte-1	0	0	0	0	0	0	0	0	
Byte-2	EXT=0	RTR=1	FD=0	SELF-RCV	LENGTH				TYPE
Byte-3	0	0	0	0	0	ID (10 -8)			ID
Byte-4	ID (7-0)								

EXT-NORMAL Message Format

	BIT-7	BIT-6	BIT-5	BIT-4	BIT-3	BIT-2	BIT-1	BIT-0	
Byte-0	1	1	1	1	1	1	1	1	SYNC
Byte-1	0	0	0	0	0	0	0	0	
Byte-2	EXT=1	RTR=0	FD=0	SELF-RCV	LENGTH				TYPE
Byte-3	0	0	0	ID (28-24)					ID
Byte-4	ID (23-16)								
Byte-5	ID (15-8)								
Byte-6	ID (7-0)								
Byte-7	DATA-0								DATA
Byte-8	DATA-1								
Byte-9	DATA-2								
Byte-10	DATA-3								
Byte-11	DATA-4								
Byte-12	DATA-5								
Byte-13	DATA-6								
Byte-14	DATA-7								

EXT-RTR Message Format

	BIT-7	BIT-6	BIT-5	BIT-4	BIT-3	BIT-2	BIT-1	BIT-0	
Byte-0	1	1	1	1	1	1	1	1	SYNC
Byte-1	0	0	0	0	0	0	0	0	
Byte-2	EXT=1	RTR=1	FD=0	SELF-RCV	LENGTH				TYPE
Byte-3	0	0	0	ID (28-24)					ID
Byte-4	ID (23-16)								
Byte-5	ID (15-8)								
Byte-6	ID (7-0)								

3.1.2.2 CAN FD Messages

The tables below show the Binary message format for CAN FD. The **FD** feature must be enabled under **can** settings to support these commands.

The Length field is now a Length Code for FD data lengths up to 64 bytes. When FD=1, then BIT-6 is interpreted as the BRS flag. The BRS flag indicates whether Bit Rate Switching is active for the message and data transmission is at the higher data rate.

STD-NORMAL FD Message Format

	BIT-7	BIT-6	BIT-5	BIT-4	BIT-3	BIT-2	BIT-1	BIT-0	
Byte-0	1	1	1	1	1	1	1	1	SYNC
Byte-1	0	0	0	0	0	0	0	0	
Byte-2	EXT=0	BRS	FD=1	SELF-RCV	LENGTH CODE				TYPE
Byte-3	0	0	0	0	0	ID (10 -8)			ID
Byte-4	ID (7-0)								
Byte-5	DATA-0								DATA
Byte-6	DATA-1								
Byte-7	DATA-2								
Byte-8	DATA-3								
Byte-9	DATA-4								
Byte-10	DATA-5								
:	:								
Byte-n	DATA-N								

EXT-NORMAL FD Message Format

	BIT-7	BIT-6	BIT-5	BIT-4	BIT-3	BIT-2	BIT-1	BIT-0	
Byte-0	1	1	1	1	1	1	1	1	SYNC
Byte-1	0	0	0	0	0	0	0	0	
Byte-2	EXT=1	BRS	FD=1	SELF-RCV	LENGTH CODE				TYPE
Byte-3	0	0	0	ID (28-24)					ID
Byte-4	ID (23-16)								
Byte-5	ID (15-8)								
Byte-6	ID (7-0)								
Byte-7	DATA-0								DATA
Byte-8	DATA-1								
Byte-9	DATA-2								
Byte-10	DATA-3								
Byte-11	DATA-4								
Byte-12	DATA-5								
:	:								
Byte-n	DATA-N								

Length Code Defined

Length Code	Length
$00-$08	0-8
$09	12
$0A	16
$0B	20
$0C	24
$0D	32
$0E	48
$0F	64

3.1.2.3 Report Error Status Message

A Report Error Status message indicates changes in the current CAN bus error status detected by the CAN controller. This message is only transmitted if the **report err** feature is enabled under the **command** settings. By default this feature is disabled.

Report Error Status Message Format

	BIT-7	BIT-6	BIT-5	BIT-4	BIT-3	BIT-2	BIT-1	BIT-0	
Byte-0	1	1	1	1	1	1	1	1	SYNC
Byte-1	0	0	0	0	0	0	0	0	
Byte-2	1	1	1	1	1	1	1	1	REPORT
Byte-3	0	0	0	0	0	0	1	1	
Byte-4	ERROR STATUS CODE (7-0)								STATUS

The following character represents the new CAN error status code.

'A' – Active (actively flagging errors on the bus, CAN operation is normal)
'W' – Warning (error count above the warning threshold)
'P' – Passive (only passively flagging errors on the bus, too many errors detected)
'B' – Bus Off (no longer transmitting on the CAN bus)

3.1.2.4 Timestamp

Outgoing serial command messages can have a 16-bit time-stamp appended to them with a 1-ms resolution. In binary mode, the received CAN messages are extended by appending the time-stamp as two binary bytes (MSB first).

The encoding of the two bytes is the same as for the DATA field (i.e., If any of the time-stamp bytes is a $FF byte, it is escaped).

The time-stamp field is always two logical bytes, but can be expanded up to four in the case where both MSB and LSB bytes are $FF.

NORMAL Timestamped Message Format

	BIT-7	BIT-6	BIT-5	BIT-4	BIT-3	BIT-2	BIT-1	BIT-0	
Byte-0	1	1	1	1	1	1	1	1	SYNC
Byte-1	0	0	0	0	0	0	0	0	
:	:								TYPE ID DATA
Byte-n	DATA-N								
Byte-n+1	TIMESTAMP (15-8)								TIME STAMP
Byte-n+2	TIMESTAMP (7-0)								

3.2 Using Message Filters

In command mode, the CAN USB-232 FD supports message filtering through the use of identifier ranges or masks and filter values, providing the ability to receive only a sub-set of all the possible CAN identifiers. This can greatly reduce the required serial data bandwidth on a busy network by only allowing certain messages to be received. Required serial bandwidth can be further reduced by using limiters.

To use the filters, they must be enabled, their type must be specified, and appropriate ID values must be assigned. This configuration is done while in the configuration mode.

3.2.1 Defining Filter Entries

- CAN messages can be filtered according to a message filter list.
- The list defines which identifiers will be received and which will be ignored.
- Each filter in the list can be enabled or disabled separately.
- Definitions can be either a single ID, a pair of IDs, an ID range or an ID mask and match value.
- A filter can be inclusive (allow to pass) or exclusive (discard).
- Standard (11-bit) and Extended (29-bit) IDs can be defined.
- Definitions must not conflict nor overlap.
- Filtering can be enabled or disabled under the **Command** settings.
- When disabled, all CAN messages are received regardless of the definitions in the list.
- When enabled, only messages which match filter definitions in the list will be passed through.
- An additional limiter can be set for messages that pass through the filter by dividing the number of messages (eg. 1 out of 20) or keeping a minimum frequency (eg. 1 every 100 ms).
- An empty filter list that is enabled will receive nothing, but still permit transmission (making the unit effectively TX only).
- The maximum number of filter definitions possible is ten (10) each for Standard and Extended IDs.

3.2.2 How Filtering Works

Whenever the CAN USB-232 FD receives a CAN message from the network, it checks to see if any filters are enabled. If none of the filters are enabled, it assumes no filtering should be performed and outputs the command message format to the serial port. If any filters are enabled, the CAN USB-232 FD will attempt to match the received message to each enabled filter in order. At the first successful match, the message is accepted (if inclusive) or discarded (if exclusive). If accepted the message is checked against any limiters then converted to command message format and output to the serial port. If no matches were successful, the message is discarded.

Note: The output format depends on mode : ASCII or BINARY

3.3 Tunnel Mode

Tunnel mode is used to transparently transfer serial data over the CAN bus. In previous versions of the product this feature was referred to as virtual circuit (VC) mode. In Tunnel mode, the CAN232 FD establishes a full-duplex virtual serial link between itself and an application on another CAN node. A serial device can transparently send and receive serial stream data on the CAN232 serial line and the CAN232 will route the data over CAN to the configured target device.

A pair of CAN232 FD devices can be utilized to create a virtual serial link. By providing a virtual serial link over the CAN network, an application can exchange data with a serial device in a network-transparent fashion, using existing CAN network cabling as a data link (Figure 4).

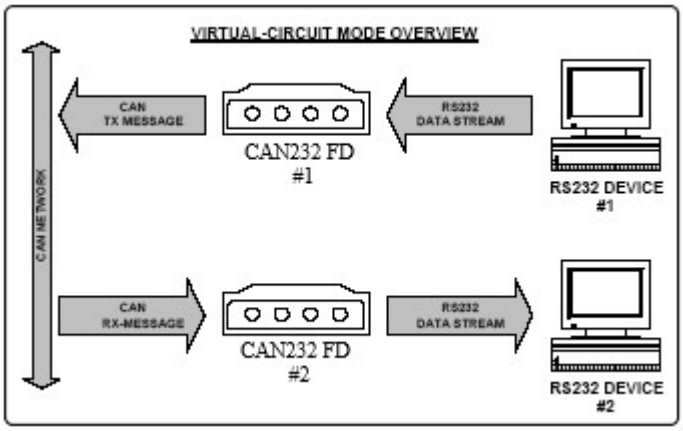


Figure 4

The Tunnel requires two, user-configurable, CAN identifiers for the transmit ID in the serial to CAN direction and the receive ID in the CAN to serial direction. The IDs can be either standard 11-bit or extended 29-bit. Since virtually every CAN protocol provides a sub-set of unused or 'user-specific' identifiers, establishing IDs for the Tunnel on a CAN network in conjunction with an existing protocol is very likely supported and easy to do.

The receive ID is used to specify which CAN message should be received by the CAN232 FD when waiting for incoming stream data. The CAN232 FD will filter out all other messages. When a CAN message containing this identifier is received, the data in the message is assumed to be stream data from another application on the CAN bus or perhaps from another CAN232 FD. This data is then read and output directly to the serial COM port.

The CAN232 FD takes data bytes coming into its serial port and groups them into CAN messages for transmission. When a remote CAN application target receives these messages, it extracts the data bytes and processes them as needed based on the serial protocol being tunneled. This operation is fully transparent to the connected serial device.

The CAN232 FD only sends CAN messages when a full CAN packet of data has been received. This leads to a case where the last part of a data stream is not sent if it is less than eight bytes long. The maximum data length can also be configured for CAN FD implementations, default 32. To deal with leftover data in a transparent fashion, the CAN232 FD provides a timeout feature that will automatically force a transmission of the last accumulated byte(s) after a specified maximum waiting time.

The CAN232 FD can also be configured to force immediate transmission upon detection of a specific user-configured byte in the data stream (CR or LF, for example).

4. Configuration Mode

Configuration mode provides a command line interface organized into multiple levels or groups that can be navigated by the user. Appropriate commands are provided at each level. The current level is always displayed as part of the prompt (eg. “config com>”). The two main levels of the CLI are for status (status>) and configuration (config>). Under the status level the user can view unit information such as serial number and firmware version as well as a log of error and debug events.

CAN Bus State During Configuration

While the CAN USB-232 FD is in the configuration mode, the CAN controller is in the bus-off state. It does not interact with the bus and is effectively ‘invisible’. Thus, it is possible to perform configuration of the CAN USB-232 FD while still connected to an active CAN network and not adversely affect network operation.

Once the configuration state is exited, the CAN USB-232 FD will activate the CAN controller and begin immediately interacting with the CAN network according to the CAN ISO 11898-2 specification.

It is the users responsibility to ensure that the configuration settings are appropriate for the network to which the CAN USB-232 FD is connected so that the CAN USB-232 FD operates correctly and does not cause erratic network operation.

4.1 General Commands

All commands are entered in lower case

```
>config
config>?
com
can
command
filters
tunnel
save
exit
```

4.1.1 help or ?

Displays a list of available commands for the given level.

4.1.2 show

Displays the configuration settings for all parameters on the given level.

4.1.3 exit

Exits the level and returns to the next level up.

4.2 root level

The root level is the entry point into configuration mode. Under the root level you can navigate to the config and status levels. There are also tools for backing up and restoring the configuration settings.

```
>?
config
status
export config
import config
exit
```

4.2.1 config

Move to the **config** level.

4.2.2 status

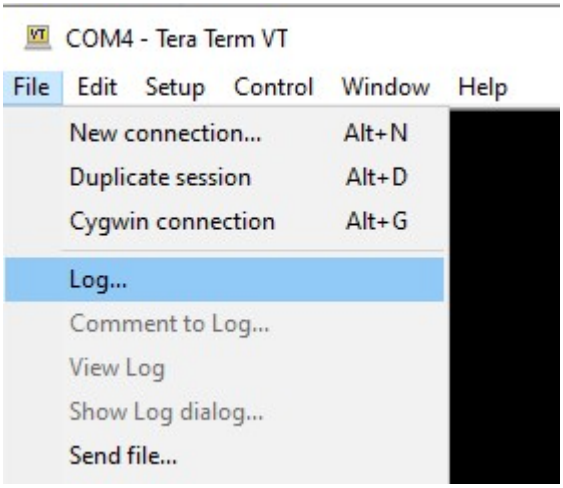
Move to the **status** level.

4.2.3 export config

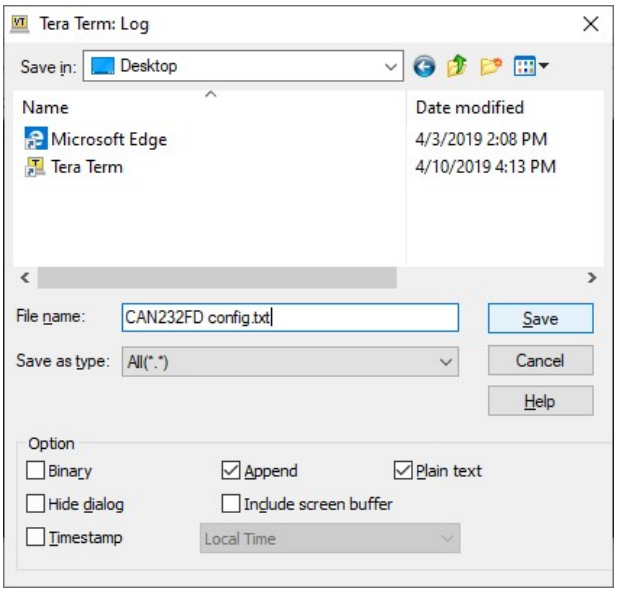
Export the current configuration to the serial line. The serial output can be captured, logged or copy/pasted to a configuration file.

Here is an example of using Tera Term to log the configuration output to a file.

First, configure the CAN USB-232 FD to the desired settings and return to the root level.
Second, in Tera Term click on File then Log…



Pick the location and file name to save the configuration file and click Save.

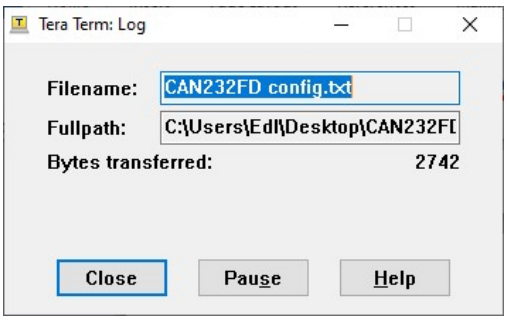


Enter the **export config** command at the prompt. The configuration output will look similar to the following:

```
>export config
config
{
com
{

baud : 115200
data bits : 8
parity : none
stop : 1
flow : none
mode : command
startup delay : 0
}

can
{

baud : 250000
sample point : 80
timeout : 0
FD : disable
FDbaud : 2000000
       .
       .
       .
type : range
reject : no
limiter : none
scale : 0
}

}

tunnel
{

rxid size : std
rxid : 0
txid size : std
txid : 0
txFD : disable
lenFD : 32
trigger : 0
timer : 20
}

}

>
```

Note: The configuration is stored as simple text and it can be edited, but it is recommended that edits to the configuration be done on the device for error checking and then use the **export config** command and log to a new file.

Next, Click Close in the Tera Term: Log window.



4.2.4 import config

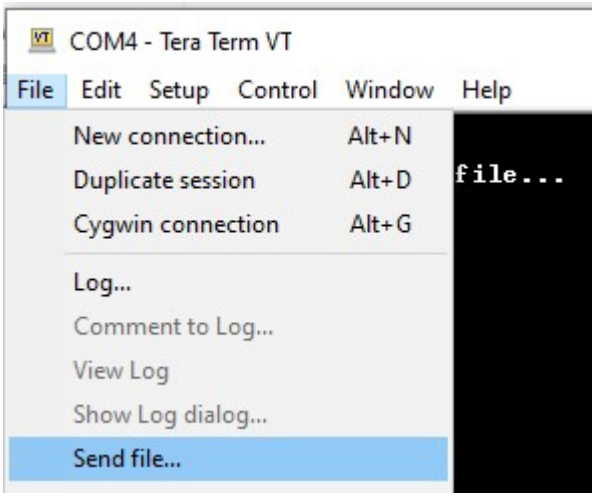
Writes new configuration settings received over the serial line from a configuration file created using the **export config** command.

Here is an example of using Tera Term to send the configuration file to the CAN USB-232 FD.

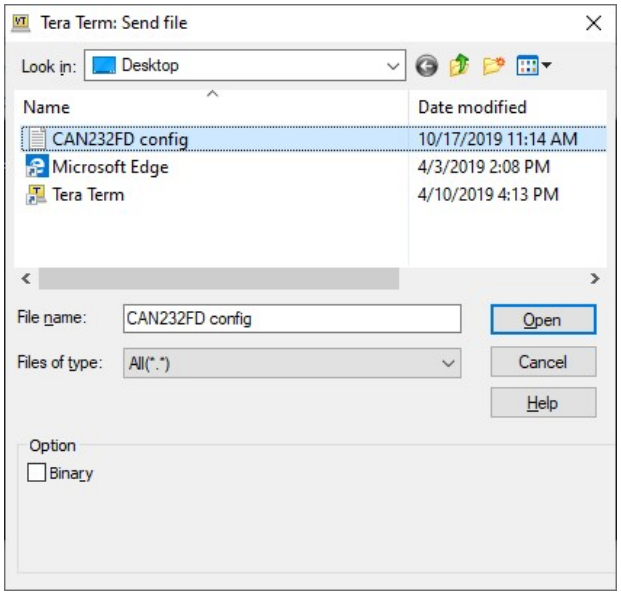
First, enter the import config command.

```
>import config
I14: Send the configuration file...
```

Second, in Tera Term click on File then Send file…



Pick the location and file name of the configuration file and click Open.



The transfer should complete quickly without errors and exit at the **config** level where you can check the new settings and use the **save** command to save the settings in the device.

```
>import config
I14: Send the configuration file...

config>
```

4.2.5 exit

Exits the root level and leaves configuration mode. The device restarts with the last saved configuration settings.

4.3 status level

Under the status level you can view device information and diagnostic data.

```
status>?
show all
log
test
exit
```

4.3.1 show all

This will show all the CAN USB-232 FD status information which includes factory settings, general configuration and a CAN bus health status.

```
status>show all
Device Name : CAN232
Serial Number : 000002
FW Version : 01.00
COM 115200 baud, 8, none, 1, none, command
CAN 250000 bps

CAN Status : Active
Bus Off events : 0
Passive events : 0
Warning events : 0
Overrun events : 0
CAN Rx Packets : 0
CAN Rx Overflow: 0
CAN Rx Skipped : 0
CAN Tx Packets : 0
CAN Tx Timeouts: 0

status>
```

4.3.2 log

This will display the debug log generated during the previous device operation.

```
status>log
CAN: command mode
UART bad idfmt
CAN Error_warning
CAN Error_passive

status>
```

4.3.3 test

To confirm that all LED indicators are working, this diagnostic causes the CAN-TX/RX and COM-TX/RX LED indicators to cycle continuously.

To terminate the diagnostic, press any key.

4.4 config level

Under the config level you can view and change settings for the operation and behavior of the device.

```
>config
config>?
com
can
command
filters
tunnel
save
exit
```

4.4.1 com

Move to the **com** level to view or make adjustments to the serial com settings.

4.4.2 can

Move to the **can** level to view or make adjustments to the CAN settings.

4.4.3 command

Move to the **command** level to view or make adjustments to the command mode settings.

4.4.4 filters

Move to the **filters** level to view or make adjustments to the filter settings.

4.4.5 tunnel

Move to the **tunnel** level to view or make adjustments to the tunnel mode settings.

4.4.6 save

Saves the current configuration settings to non-volatile memory. A warning message is displayed if you exit the config level without saving your changes.

4.5 config com

The com level contains the serial communication settings.

```
config com>?
show
baud <value>
data bits <value>
parity <value>
stop <value>
flow <value>
mode <value>
startup delay <value>
exit
```

4.5.1 baud <value>

Sets the baud rate to be used for serial communication. Any baud rate from **1200** to **1000000** (1 Mbps) can be chosen, but not all baud rates can be matched exactly by the device. All standard serial baud rates are supported. The default baud rate is **115200**.

4.5.2 data bits <value>

Set the number of data bits for serial communication. Values of **7** or **8** are valid.

4.5.3 parity <value>

Set the parity to the options of **none**, **even**, or **odd**.

4.5.4 stop <value>

Sets the number of stop bits for serial communication. Values of **1** or **2** are valid.

4.5.5 flow <value>

Sets flow control to the options of **none**, **software** or **hardware**. Software flow control uses the XON and XOFF characters for limiting serial communication to prevent buffer overflow and is only recommended for ASCII protocols. Hardware flow control uses the RS232 control signals RTS and CTS for limiting serial communication.

4.5.6 mode <value>

Sets the serial communication mode to the options of **command** or **tunnel**.

In the command mode, the CAN USB-232 FD is capable of sending and receiving arbitrary CAN messages over the serial line via the use of ASCII or binary formatted messages.

In tunnel or virtual circuit mode, the CAN USB-232 FD transparently tunnels serial data over the CAN network between devices creating a virtual serial link. The CAN ID destination for received serial data is specified along with the CAN ID for receiving the serial data packets.

4.5.7 startup delay <value>

Sets the serial communication startup delay. Values of **0** to **60** seconds are valid.

This feature allows the user to specify a time-out from the time the unit is first powered up to when it starts processing CAN messages. This can be useful to eliminate message flooding on the serial line while the host application code gears up.

4.6 config can

The **can** level contains the CAN communication settings.

```
config can>?
show
baud <value>
sample point <value>
timeout <value>
expert
FD <value>
FDbaud <value>
FDexpert
exit
```

4.6.1 baud <value>

Sets the baud rate to be used for classic CAN communication. Any baud rate from **5000** to **1000000** (1 Mbps) can be chosen, but not all baud rates can be matched exactly by the device. All standard CAN baud rates are supported. The default baud rate is **250000**.

Defining the CAN Bit-Time

In order for a CAN network to correctly arbitrate multiple senders and to adapt to variances in system clocks, the low-level bit time must be correctly specified for the specific characteristics of the network. The CAN USB-232 FD provides a means of configuring these parameters so as to support a wide array of network scenarios.

Note that the CAN bit-time settings at this level can only be specified as a baud rate and sample point. The code will find the best fit solution. Move to the **expert** level for fine tune adjustments to the CAN bit sampling.

$$Tq = \frac{clkdiv}{48{,}000{,}000} \qquad baud = \frac{1}{Tq * (1 + tseg1 + tseg2)} = bits/sec$$

4.6.2 sample point <value>

Sets the sample point during the total bit time as a percentage which dictates when the CAN signal is sampled at each bit. A value from **70** to **95**% can be chosen. The default is **75**%. It is very important that all nodes on a CAN FD network have the same sample point as this dictates when a bit rate switch occurs.

4.6.3 timeout <value>

Sets a CAN message transmit timeout in milliseconds. A value of **0** to **10000** ms can be chosen. A value of **0** will disable the timeout.

When a message is transmitted on the CAN bus and loses arbitration to a higher priority message or is transmitted without an acknowledgement, it is resent continuously until successful. It is possible, under heavy CAN bus loads or faulty CAN bus, to place the unit in a state where no more input parsing from the COM port can occur due to the internal buffers being full and the CAN TX not being able to proceed. This will block further CAN messages from being transmitted. Also under this condition, a config command cannot be processed. This is to ensure that no CAN messages are lost while waiting for the CAN bus to become available.

In order to alleviate this potential problem, a timeout parameter can be specified that will cancel a transmit in progress and allow other CAN messages to be transmitted.

4.6.4 expert

Move to the **`expert`** level to view or make fine tune adjustments to the CAN bit sampling.

4.6.5 FD <value>

Set the CAN FD feature to **enable** or **disable**. With the CAN FD feature disabled (default), the CAN USB-232 FD can only send and receive CAN 2.0A/B packets on the bus and makes the device incompatible with CAN FD networks.

4.6.6 FDbaud <value>

Sets the baud rate to be used during the data phase of CAN FD communication when bit rate switching (BRS) is enabled. Any baud rate from **20000** to **4000000** (4 Mbps) can be chosen, but not all baud rates can be matched exactly by the device. All standard CAN baud rates are supported. The default baud rate is **2000000** (2 Mbps).

Defining the Data CAN Bit-Time

In order for a CAN FD network to correctly operate and to adapt to variances in system clocks, the low-level bit time must be correctly specified for the specific characteristics of the network. The CAN USB-232 FD provides a means of configuring these parameters so as to support a wide array of network scenarios.

Note that the FD CAN bit-time settings at this level can only be specified as a baud rate. The code will find the best fit solution. Move to the **`FDexpert`** level for fine tune adjustments to the CAN bit sampling.

4.6.7 FDexpert

Move to the **`FDexpert`** level to view or make fine tune adjustments to the CAN bit sampling during the data phase of CAN FD.

4.7 config can expert

The **can expert** level contains the fine tune parameters for CAN bit timing and sampling.

```
config can expert>?
show
baud <value>
sample point <value>
clkdiv <value>
tseg1 <value>
tseg2 <value>
sjw <value>
exit
```

4.7.1 baud <value>

Sets the baud rate to be used for classic CAN communication. This is the same setting as described for the previous **config can** level.

4.7.2 sample point <value>

Sets the sample point during the total bit time as a percentage which dictates when the CAN signal is sampled at each bit. This is the same setting as described for the previous **config can** level.

4.7.3 clkdiv <value>

Sets the clock divider for the CAN controller's baud rate generator. This value is automatically determined by the device based on baud rate and does not normally need to be adjusted. Any value from **1** to **48** will be accepted. The baud rate and other expert settings will be automatically adjusted accordingly when this parameter is changed.

The CAN controller uses a 16 MHz internal clock for baud rate generation.

4.7.4 tseg1 <value>

Sets the time segment 1 for the total bit time which is the number of clock cycles before the sample point. This value is automatically determined by the device based on baud rate, clkdiv and sample point. It does not normally need to be adjusted. Any value from **1** to **256** will be accepted. The baud rate, sample point and other expert settings will be automatically adjusted accordingly when this parameter is changed.

4.7.5 tseg2 <value>

Sets the time segment 2 for the total bit time which is the number of clock cycles after the sample point. This value is automatically determined by the device based on baud rate, clkdiv and sample point. It does not normally need to be adjusted. Any value from **1** to **128** will be accepted. The baud rate, sample point, sjw and other expert settings will be automatically adjusted accordingly when this parameter is changed.

4.7.6 sjw <value>

Sets the synchronization jump width (sjw) for the CAN receiver. This is the maximum number of Tq adjustment the receiver can make to adjust for oscillator differences between CAN nodes. It does not normally need to be adjusted. Any value from **1** to **tseg2** will be accepted.

4.8 config can FDexpert

The **can FDexpert** level contains the fine tune parameters for CAN FD bit timing and sampling during the data phase.

```
config can FDexpert>?
show
FDbaud <value>
FDsample point <value>
FDclkdiv <value>
FDtseg1 <value>
FDtseg2 <value>
FDsjw <value>
exit
```

4.8.1 FDbaud <value>

Sets the baud rate to be used during the data phase of CAN FD communication when bit rate switching (BRS) is enabled. This is the same setting as described for the previous **config can** level.

4.8.2 FDsample point <value>

Sets the sample point during the total bit time of the data phase as a percentage which dictates when the CAN signal is sampled at each data bit.

4.8.3 FDclkdiv <value>

Sets the clock divider for the CAN controller's baud rate generator. This value is automatically determined by the device based on baud rate and does not normally need to be adjusted. Any value from **1** to **32** will be accepted. The FDbaud rate and other FDexpert settings will be automatically adjusted accordingly when this parameter is changed.

The CAN controller uses a 16 MHz internal clock for baud rate generation.

4.8.4 FDtseg1 <value>

Sets the time segment 1 for the total bit time in the data phase which is the number of clock cycles before the sample point. This value is automatically determined by the device based on FDbaud rate, FDclkdiv and FDsample point. It does not normally need to be adjusted. Any value from **1** to **32** will be accepted. The FDbaud rate, FDsample point and other FDexpert settings will be automatically adjusted accordingly when this parameter is changed.

4.8.5 FDtseg2 <value>

Sets the time segment 2 for the total bit time in the data phase which is the number of clock cycles after the sample point. This value is automatically determined by the device based on baud rate, clkdiv and sample point. It does not normally need to be adjusted. Any value from **1** to **16** will be accepted. The FDbaud rate, FDsample point, FDsjw and other FDexpert settings will be automatically adjusted accordingly when this parameter is changed.

4.8.6 FDsjw <value>

Sets the synchronization jump width (SJW) for the CAN FD receiver in the data phase. This is the maximum number of Tq adjustment the receiver can make to adjust for oscillator differences between CAN nodes. It does not normally need to be adjusted. Any value from **1** to **tseg2** will be accepted.

4.9 config command

The **command** level contains the Command mode parameters.

```
config command>?
show
filter <value>
mode <value>
format <value>
timestamp <value>
eol <value>
config cmd <value>
report err <value>
exit
```

4.9.1 filter <value>

Sets whether incoming CAN message filtering is **on** or **off**.

4.9.2 mode <value>

Sets whether the command's CAN interface mode is **normal**, **monitor** or **one-shot**. A mode of **normal** allows transmit and receive of CAN packets with retries. A mode of **monitor** only allows receive of CAN packets and prevents any disturbing of the CAN bus. In **one-shot** mode no attempts to automatically retry any transmissions due to error or lost arbitration will occur. This is used in time-triggered CAN protocols.

4.9.3 format <value>

Sets the serial command message format between **ascii** or **binary**. The ASCII format uses only printable characters that can also be easily read by humans on a terminal display. All data is converted to ASCII hex. The binary format requires fewer bytes on the serial line and is more efficient for a software application.

4.9.4 timestamp <value>

Outgoing serial command messages can have a 16-bit time-stamp appended to them with a 1-ms resolution. Values of **off** or **on** are valid.

4.9.5 eol <value>

Configuration parameter to append an end-of-line sequence to each ASCII command message string generated. Values of **none**, **cr**, **lf**, **crlf** and **lfcr** are valid.

4.9.6 config cmd <value>

It is possible to enter configuration mode through the COM port in Command mode by enabling this option. Valid settings are **enable** or **disable**.

ASCII mode command = :CONFIG;

BINARY mode command = FF 00 FF 02 43 4F 4E 46 49 47
Note: The last six bytes of the BINARY command are ASCII 'C' 'O' 'N' 'F' 'I' 'G'

4.10 config tunnel

The **tunnel** level contains the Tunnel mode parameters.

```
config tunnel>?
show
rxid size <value>
rxid <value>
txid size <value>
txid <value>
txFD <value>
lenFD <value>
trigger <value>
timer <value>
exit
```

4.10.1 rxid size <value>

Sets the receive ID size between **std** (11-bit) or **ext** (29-bit).

4.10.2 rxid <value>

Sets the receive ID for CAN messages containing the serial stream data. Values between 0 and 1FFFFFFF in hex can be specified.

4.10.3 txid size <value>

Sets the transmit ID size between **std** (11-bit) or **ext** (29-bit).

4.10.4 txid <value>

Sets the transmit ID for CAN messages containing the serial stream data. Values between 0 and 1FFFFFFF in hex can be specified.

4.10.5 txFD <value>

Sets whether the serial stream data is sent with CAN FD protocol. Valid settings are **enable** or **disable**. If disabled, CAN 2.0A/B messages are transmitted with 8 serial bytes per message maximum. If enabled, CAN FD messages with bit rate switching (BRS) are transmitted and the **lenFD** parameter sets the maximum data length.

4.10.6 lenFD <value>

Sets the maximum data length when transmitting serial stream data in CAN FD packets. Valid settings are **8**, **12**, **16**, **20**, **24**, **32**, **48**, or **64** bytes. These are the specific data length codes (DLC) supported by CAN FD.

When a CAN FD transmit is initiated by the **trigger** byte or the **timer** then the largest possible data length code, based on the remaining data length, is transmitted in successive packets until the initiated serial data is completely transferred.

4.10.7 trigger <value>

Sets the trigger byte value that will force an immediate CAN transmit when matched on the serial input. Values between **00** and **FF** in hex can be specified. A value of **00** disables the trigger.

4.10.8 timer <value>

Sets the timer value in milliseconds that will force a CAN transmit of any remaining serial data. Values between **0** and **1000** ms can be specified. A value of **0** disables the timer.

4.11 config filters

The **filters** level contains the CAN ID filters used in command mode.

```
config filters>?
show all
std filter <instance>
ext filter <instance>
exit
```

4.11.1 show all

This will show all the filters configured in a condense list for easier viewing.

```
config filters>show all
Standard Filters
01: + 000 - 7FF

Extended Filters
01: + 00000000 - 1FFFFFFF

config filters>
```

This shows the default filter settings which are two fully inclusive identifier ranges.

A + marks the filter as enabled (inclusive) and a – marks the filter as enabled (exclusive). A space (no + or -) means the filter is disabled.

4.11.2 std filter <instance>

Move to the standard identifier filter instance detail view. The instance must be from 1 – 10.

4.11.3 ext filter <instance>

Move to the extended identifier filter instance detail view. The instance must be from 1 – 10.

4.11.4 Setting Up Message Filters

To use message filtering, each filter used must be enabled and configured as to the type of filtering that is desired. Once all configuration parameters have been saved and the CAN USB-232 FD enters the normal operating mode, the new filter settings will become active and will be applied to each received CAN message.

4.12 config filters std #n

The **filters std #n** level contains the detail parameters for standard ID filter #n.

```
config filters std #1>?
show
enable <value>
sid1 <value>
sid2 <value>
type <value>
reject <value>
limiter <value>
scale <value>
exit
```

4.12.1 enable <value>

Sets the enable value for this filter to **no** or **yes**.

4.12.2 sid1 <value>

Sets the sid1 value for this filter. Values between 0 and 7FF in hex can be specified.

4.12.3 sid2 <value>

Sets the sid2 value for this filter. Values between 0 and 7FF in hex can be specified.

4.12.4 type <value>

Sets the type for this filter between **range**, **dual** and **classic**.

- range: sid1=< ID <= sid2
- dual: ID = sid1 or ID = sid2
- classic: (Active/1 bits in sid1) = (ID bits in sid2)

Note to create a single ID filter use type range or dual and set sid1=sid2=ID or use type classic and set sid1=7FF and sid2=ID.

4.12.5 reject <value>

Sets the reject option for this filter to **no** or **yes** for inclusive or exclusive filtering respectively.

4.12.6 limiter <value>

Sets the limiter type for this filter between **none**, **divide** and **frequency**. The **none** selection is the default and disables the limiter so all messages pass. The divide setting will only pass 1 message out of every **scale** messages. The frequency setting will only pass 1 message every **scale** milliseconds.

4.12.7 scale <value>

Sets the scale value for this limiter. Values between 0 and 10000 can be specified.

4.13 config filters ext #n

The **filters ext #n** level contains the detail parameters for extended ID filter #n.

```
config filters ext #1>?
show
enable <value>
eid1 <value>
eid2 <value>
type <value>
reject <value>
limiter <value>
scale <value>
exit
```

4.13.1 enable <value>

Sets the enable value for this filter to **no** or **yes**.

4.13.2 eid1 <value>

Sets the eid1 value for this filter. Values between 0 and 1FFFFFFF in hex can be specified.

4.13.3 eid2 <value>

Sets the eid2 value for this filter. Values between 0 and 1FFFFFFF in hex can be specified.

4.13.4 type <value>

Sets the type for this filter between **range**, **dual** and **classic**.

- range: eid1=< ID <= eid2
- dual: ID = eid1 or ID = eid2
- classic: (Active/1 bits in eid1) = (ID bits in eid2)

Note to create a single ID filter use type range or dual and set eid1=eid2=ID or use type classic and set eid1=1FFFFFFF and eid2=ID.

4.13.5 reject <value>

Sets the reject option for this filter to **no** or **yes** for inclusive or exclusive filtering respectively.

4.13.6 limiter <value>

Sets the limiter type for this filter between **none**, **divide** and **frequency**. The **none** selection is the default and disables the limiter so all messages pass. The divide setting will only pass 1 message out of every **scale** messages. The frequency setting will only pass 1 message every **scale** milliseconds.

4.13.7 scale <value>

Sets the scale value for this limiter. Values between 0 and 10000 can be specified.

5. Technical Support

If you are experiencing a problem, please read the user manual and other technical document supplied on the product CD. If you are unable to solve the problem, please contact technical support.

We are located in a far western suburb of Chicago and are on Central Standard Time. Our location allows us to best service all of North America.

Grid Connect, Inc.
1630 W. Diehl Road
Naperville, Illinois 60563 USA

+1 (800) 975-GRID (4743) USA Toll Free
+1 (630) 245-1445 Phone
+1 (630) 245-1717 Fax

Hours of Operation

Monday - Friday
Business Hours: 7:30 a.m. - 5:00 p.m. CST
Tech Support Hours: 8 a.m. - 5:30 p.m. CST