gridconnect

CAN USB-232
User Manual

Revision C November 27 , 2012

Document Part Number: GC_CAN_USB232_UM

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
04/19/2012	A	GR	Preliminary Release
10/15/12	B	GR	Updates to drawings
11/27/12	C	GR/DQ	New Features. V01.02

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

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

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

In the virtual circuit mode, the CAN USB-232 creates a transparent 'virtual' link between itself and another CAN USB-232, providing the ability for applications to use a CAN network as a point-to-point link, supporting full-duplex exchange of arbitrary stream data.

The CAN USB-232 operating mode and all other configuration parameters are accessed through the 'config' button. Pressing the button causes the CAN USB-232 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 to assist in the measurement of bus signal propagation time and termination characteristics.

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

There are two hardware platforms, both of which execute the same firmware, with one providing a CAN-RS232 interface and the other a CAN-USB interface. The CAN-RS232 version is shown below.

The new units are based on an ARM Cortex-M3 32-bit microcontroller, providing 100 MIPS of processing power with an extremely efficient instructions set. The performance increase over the legacy microcontroller is approximately 25 times faster.

All device interfaces (UART, CAN, USB) 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 USB interface has been re-designed based on a parallel-port version of the current USB-to-UART interface chip, providing 100% USB interface speed while using the same USB driver software as the earlier CAN-USB unit. The application interface remains identical, providing 100% backward compatibility at the USB level while adding a major performance boost.

Both versions support isolated and non-isolated CAN configurations. This option must be selected at the time of purchase.

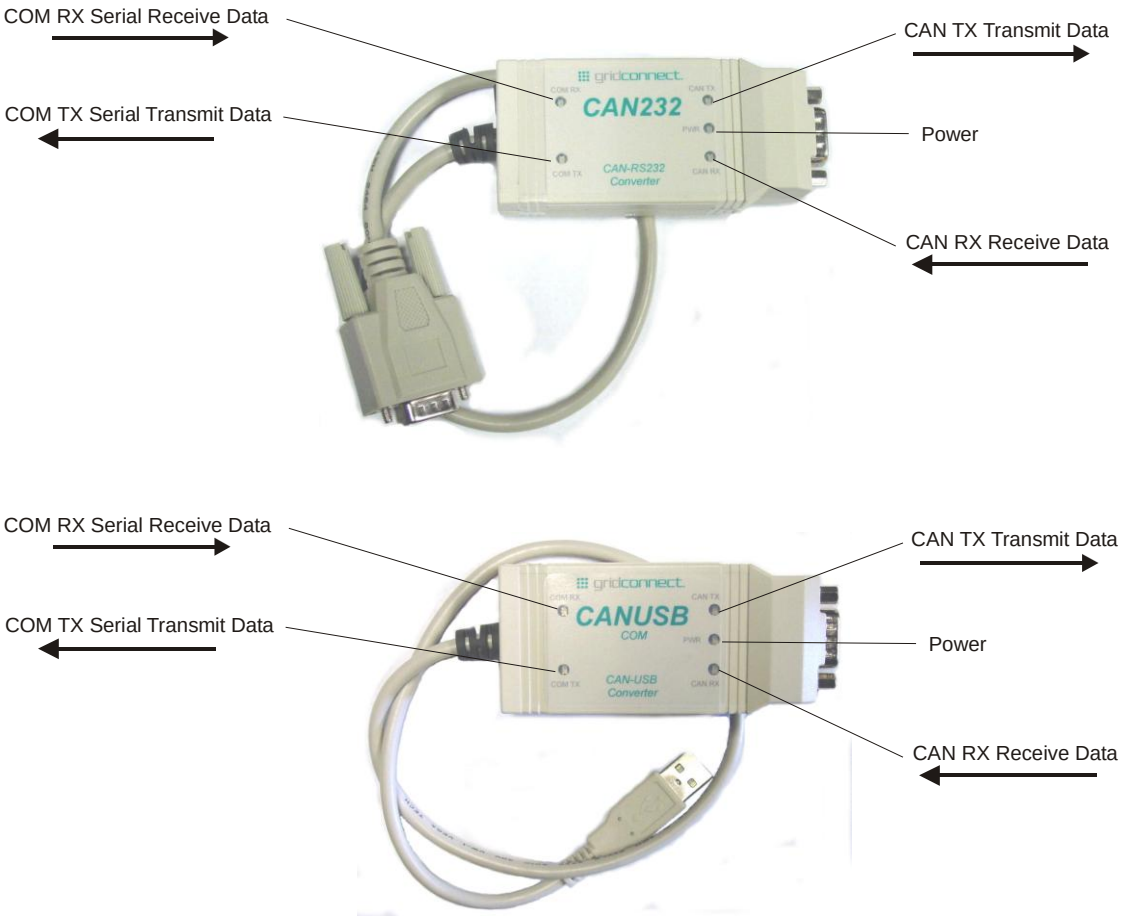
With the additional microcontroller FLASH memory available, the new units support a user-accessible boot-loader, allowing users to download firmware updates by themselves.

By using the time-tested and very simple XMODEM protocol widely available in terminal emulation programs, end-users can download firmware directly through the COM-RS232 or COM-USB port to update the device.

To protect against unauthorized duplication of hardware, the firmware is encrypted with a 256-bit AES cipher. The bootloader decrypts and verifies authenticity of the firmware to ensure only an authorized version is being downloaded.

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 the same firmware, with one providing a CAN-RS232 interface and the other a CAN-USB interface. The CAN-RS232 platform has the option of DTE (male) or DCE(female) on the RS232 serial cable, and the option of a Male or Female DB9 CAN connector.

1.4 CAN Network

Before trying to send commands to the CAN USB-232, make sure the unit is 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 will attempt to transmit the message indefinitely (as per CAN 2.0A/B specifications).

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 ground in order to work properly.

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

1.5 Additional Documentation

This manual provides basic information about the CAN USB-232 installation and operation. For specific details about configuration and operation, please see the following manuals.

The following documents are available on the product CD.

Title	Description	File Name
CAN-USB/232 User Manual	This manual in PDF format.	CAN_USB232_UM.pdf
AN-160	**COM Port Assignment**	AN_160_COMPort_Assigment_User_Guide.pdf
AN-134	**Drivers for MAC OS**	AN_134_FTDI_Drivers_Installation_Guide_for_MAC_OSX.pdf
AN-104	**Drivers for Window XP**	AN_104_FTDI_Drivers_Installation_Guide_for_WindowsXP(FT_000093).pdf See Note 1.
AN-119	**Drivers for Windows 7**	AN_119_FTDI_Drivers_Installation_Guide_for_Windows7.pdf See Note 1.
Windows CE 4.2-5.2	**Installation Guide**	Windows_CE_Installation_Guide.pdf See Note 2.
Windows CE 6.0	**Installation Guide**	Windows_CE_Installation_Guide.pdf See Note 2.
Help Tips	**Additional Help Information**	help-tips.pdf

Note 1: Includes the following versions of the Windows operating system: Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2.

Note 2: Includes the following versions of Windows CE 4.2-5.2 based operating systems: Windows Mobile 2003, Windows Mobile 2003 SE, Windows Mobile 5, Windows Mobile 6, Windows Mobile 6.1 ,Windows Mobile 6.5

2. Getting Started

2.1 Driver Installation

Note: Install the USB drivers ONLY if you are using a CAN-USB. The files are not needed for CAN-232.

1. Insert the product CD into your CD-ROM drive. The CD will automatically start and display the main window.

   If the CD does not launch automatically:

   a) Locate your CD Drive. Example: CD-RW Drive (D:)

   b) Double-click on **autorun.exe** to start the CD browser.

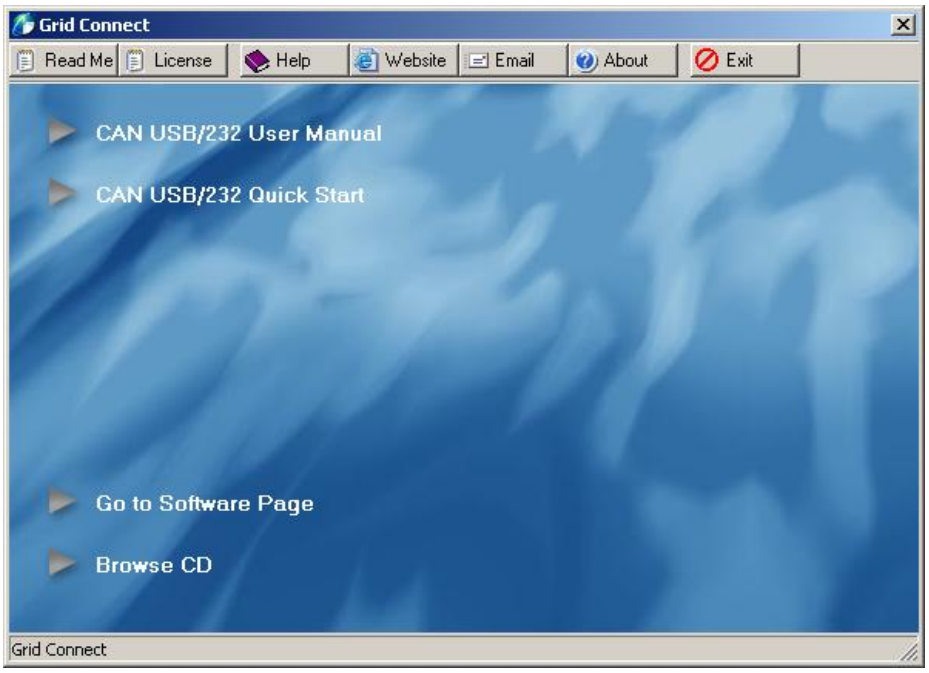
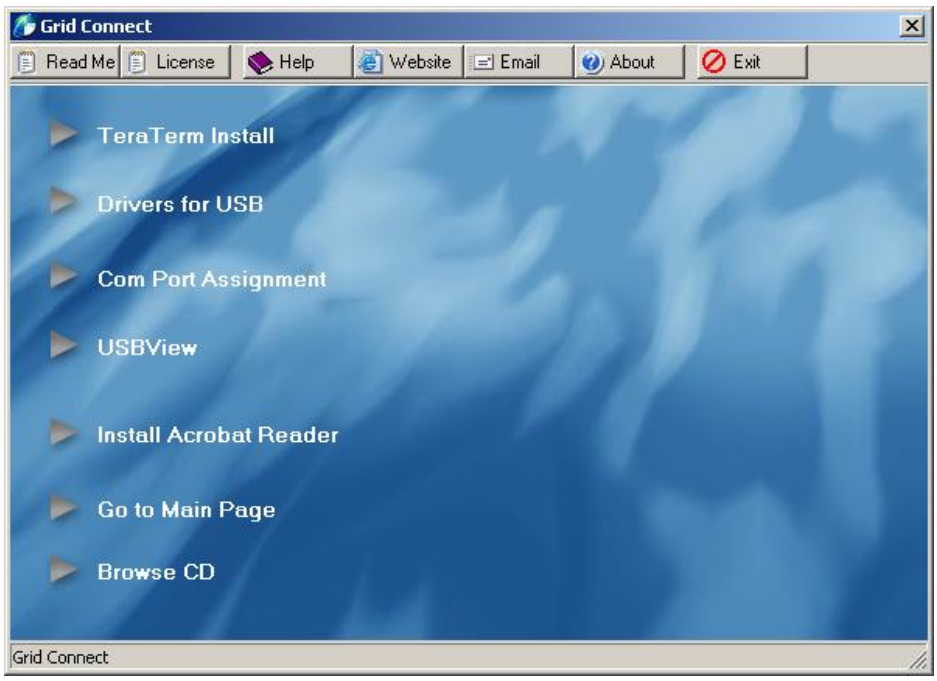


Figure 1 - CD Main Window

1. Click on an item for more information.
2. Click the **Browse CD** button to view the contents of the CD.
3. Click the **Go to Software Page** button to view software options.



2.1.1 TeraTerm Install

Click this button to install TeraTerm terminal emulation software. Very useful for viewing the configuration menus.

2.1.2 USB Drivers

Drivers for Linux, MAC OS and Windows can be found on individual folders on the software CD. Please read the application note if available.

2.1.2.1 Linux

There are two files available for Linux., one in the x64 (64-bit) folder and another in the x86 (32-bit) folder.

2.1.2.2 MAC OS X

Please read the AN_134_FTDI_Drivers_Installation_Guide_for_MAC_OSX.pdf file in the Mac OS X folder on the software CD.

2.1.2.3 Windows

The drivers for Windows includes the following versions of the Windows operating system: Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2.

Please read the Windows application note:
AN_104_FTDI_Drivers_Installation_Guide_for_WindowsXP(FT_000093).pdf.

Please read the Windows 7 application note:
AN_119_FTDI_Drivers_Installation_Guide_for_Windows7.pdf.

The application notes and the driver files are contained on the software CD in the Windows folder.

Once connected to a USB port, the CANUSB will appear as a COM port in the Device Manager. The CANUSB will always use the lowest available COM port for operation. For instance, if COM ports 1 thru 3 are in use by other peripherals and applications, the CANUSB will use COM 4.

The CANUSB functions identically to a COM port from the reference point of both the host application and the serial device, and it can support serial device control requests defined in the Microsoft Win32® Communications API.

The Virtual COM Port (VCP) device drivers allow the CANUSB device to appear to the PC's application software as an additional COM port (in addition to any existing hardware COM ports). Application software running on the PC accesses the CANUSB device as it would access a standard hardware COM port. However, actual data transfer between the PC and the CANUSB device is performed over the USB. Therefore, existing COM port applications may be used to transfer data via the USB to the CANUSB-based device without modifying the application.

2.1.2.4 Windows CE 4.2-5.2

The drivers for Windows CE includes the following versions of Windows CE 4.2-5.2 based operating systems: Windows Mobile 2003, Windows Mobile 2003 SE, Windows Mobile 5, Windows Mobile 6, Windows Mobile 6.1 ,Windows Mobile 6.5

Please read the application note Windows_CE_Installation_Guide.pdf. The purpose of this document is to provide users of FTDI chips with a simple procedure for installing drivers for their devices on PDAs and targets running Windows CE 4.2 and later. The application note and drivers files are contained on the software CD in the Windows CE 4.2-5.2 folder.

2.1.2.5 Windows CE 6.0

Please read the application note Windows_CE_Installation_Guide.pdf. The purpose of this document is to provide users of FTDI chips with a simple procedure for installing drivers for their devices on PDAs and targets running Windows CE 6.0 and later. The application note and drivers files are contained on the software CD in the Windows CE 6.0 folder.

2.1.3 COM Port Assignment

COMPort_Assignment is a utility which is used for assigning the COM Port numbers of FTDI devices. The utility is available as a free download from the Utilities page of the FTDI website. Please read the AN_160_COMPort_Assigment_User_Guide.pdf for installation and operating instructions. The application note and the Reassign COMNo Utility.zip file are contained on the software CD.

2.1.4 USBView

USBView is a free utility from Microsoft that displays the USB connection tree and shows the USB devices that are connected to it together with their configuration data. This is very useful for debugging USB enumeration errors. USBView runs under Windows 98, ME, 2000 and XP.

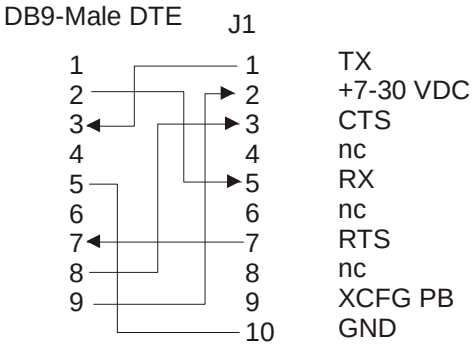
2.2 Hardware Installation

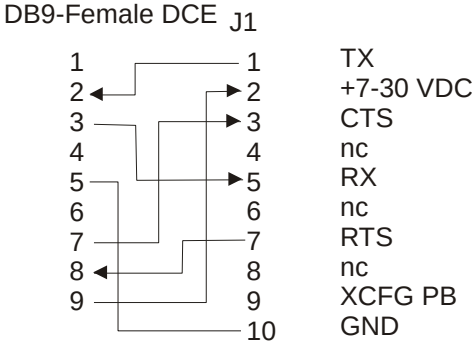
2.2.1 RS232 Serial Cable

The following drawing shows the connection diagram for the DB-9 Male and DB-9 Female cables to the solder pads on the circuit board at J1.

Pin 9 on the DB-9 connector is shown with +7-30 VDC going to Pin 2 of J1. This an optional method of supplying power to the device. It is not required if you use a different power source. The factory default configuration has no internal connection to J1-pin 2. For more details about power, see Power Options on page 2-10.

The signal on J1-pin 9 is named XCFG PB, which is the External Configuration Push Button input. This signal is NOT connected to the DB9-Male cable at the factory. If you need the push button input, contact the sales department to get a quote for a custom cable.





2.2.2 USB Cable Wiring

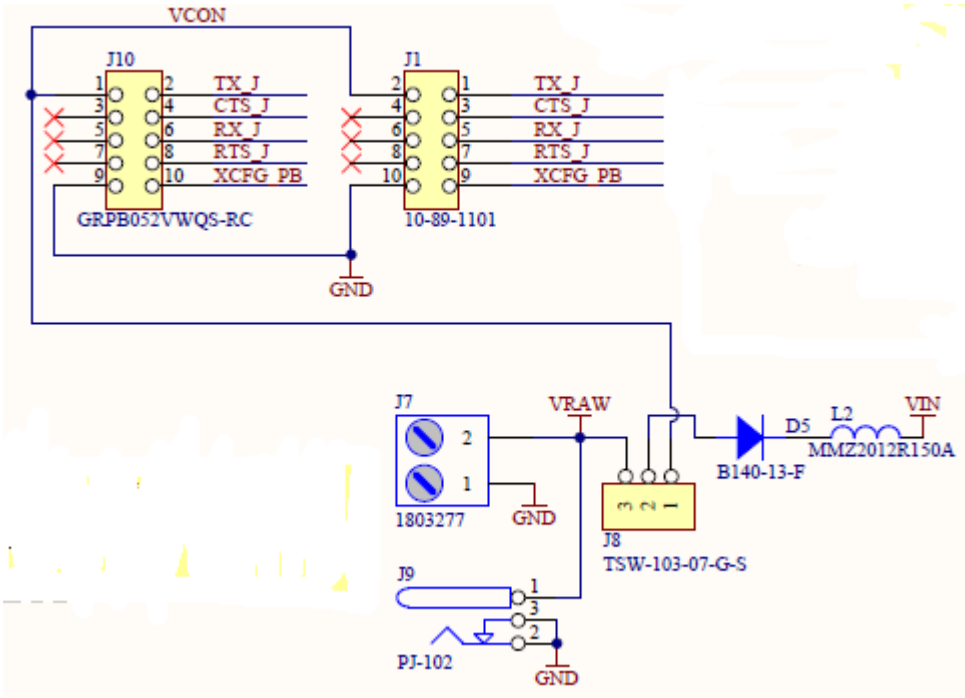
The following drawing shows the connection diagram for the USB cable to the solder pads on the circuit board at J1.

USB Type A	USB-232 J1
1 VBUS (Red)	1 VIN
2 D- (White)	2 USBM
3 D+ (Green)	3 USBP
4 GND (Black)	4 GND

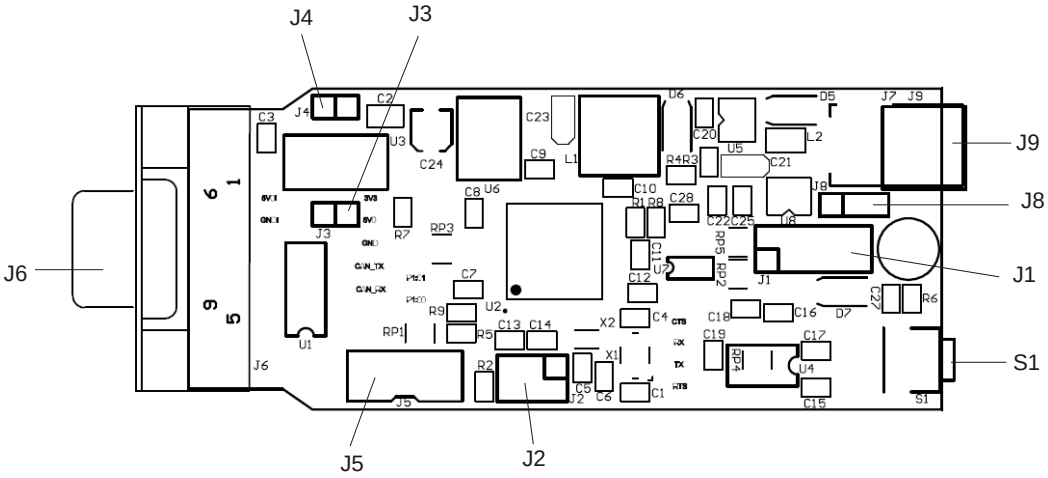
2.2.3 Power Options

For the CAN-232 model, power (+7-30 VDC) can be supplied through a Phoenix terminal block (J7), through a barrel jack (J9), through the serial cable (usually pin 9) to J1-Pin 2, or through the OEM option connector J10-pin 1. The factory default is the Barrel Jack J9.

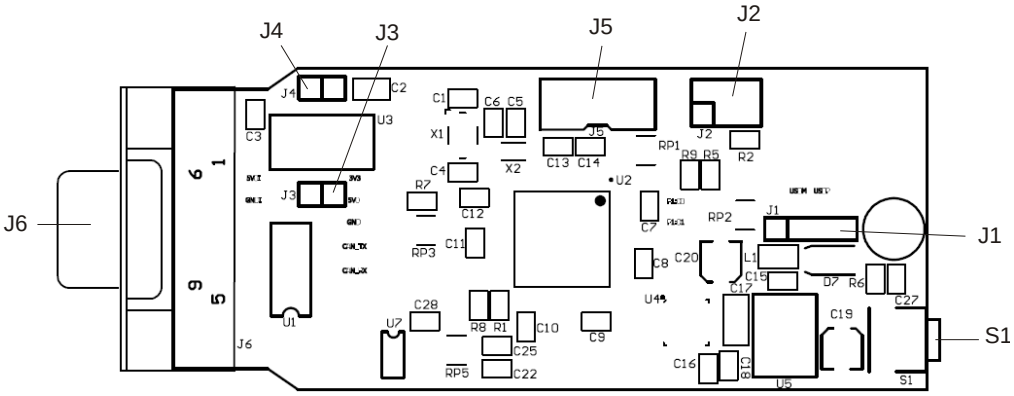
The Jumper J8 should be in position 3-2 for Phoenix or barrel jack power. Jumper J8 should be in position 1-2 for serial cable or OEM (J10) supplied power. Factory default setting is a jumper in position 3-2.



2.2.4 Item Locator CAN-232

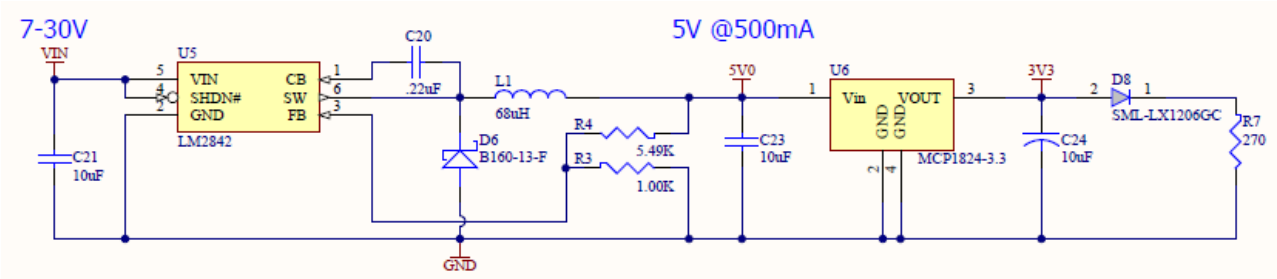


2.2.5 Item Locator CAN-USB

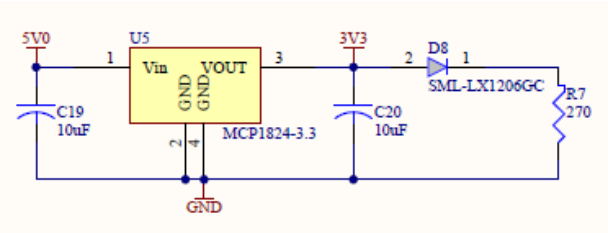


2.2.6 Input Power

The input power range is from 7-30VDC for the CAN-232 model. The 5V regulator U5 supplies power to the U6, the 3.3V regulator. D8 is the Power LED.



For the CAN-USB version, power is supplied by the USB connection. D8 is the Power LED.



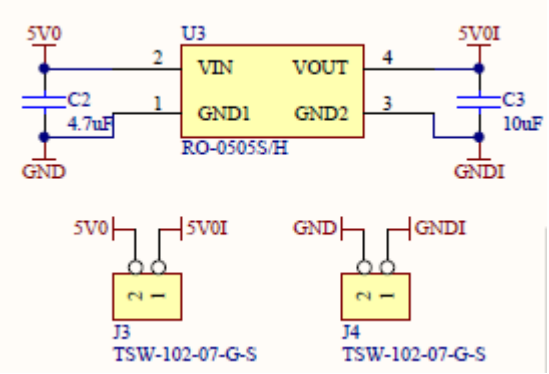
2.2.7 Isolated Power Option

Both the CAN-USB and CAN-232 can be configured for the Isolation option. Non-isolated units do not have U3 installed. Jumpers J3 and J4 are used to supply +5 and ground to the output circuits.

For isolation, U3 is installed and jumpers J3 and J4 are removed. U3 is a DC-DC converter that isolates the two power circuits.

The DC/DC converter is typically used in general purpose power isolation and voltage matching applications, and feature a full industrial operating temperature range of -40°C to +85°C without derating. The device is 2kVDC rated in a UL94V-0 package.

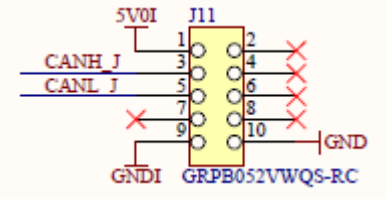
Note: The isolated option must be selected at the time the unit is ordered. This is not a field upgrade option.



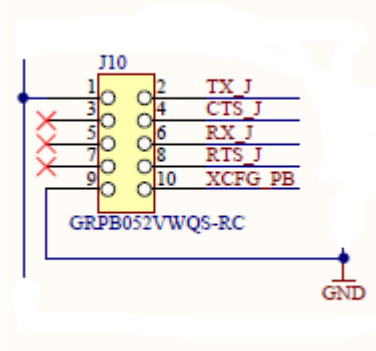
2.2.8 OEM Options

The OEM version comes without a USB or 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 circuit board.

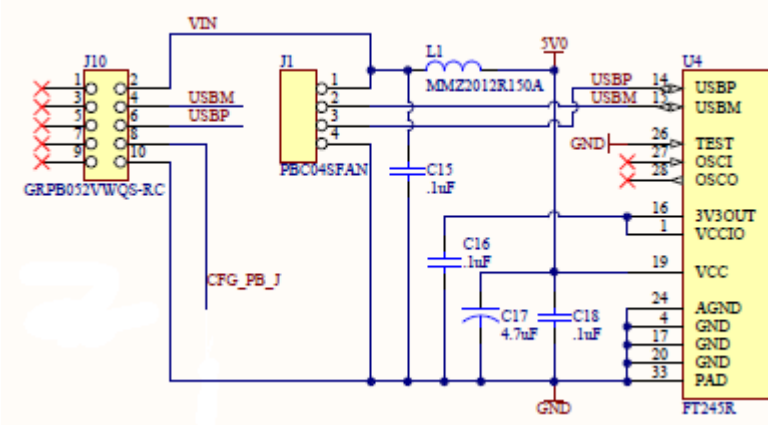
The CAN signals are brought out to the OEM connector at J11. The OEM connector is a 10-pin header soldered to the bottom side of the board.



The RS232 signals are brought out to the OEM connector at J10. Power can be supplied through pin 1 and Ground is tied to pin 9. J8 must also be jumpered between pins 1 and 2. (See Power Options for J8)



The USB version is slightly different. Power is tied to pin 2 and ground to pin 10.

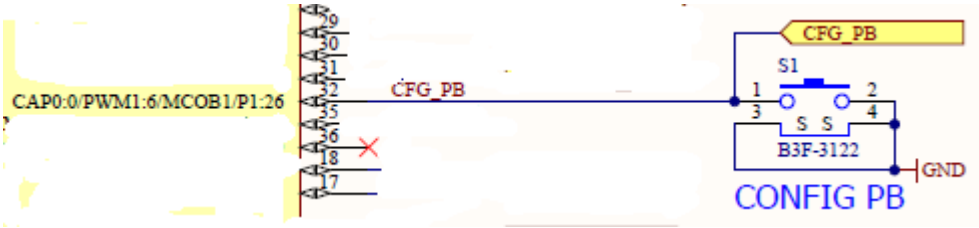


2.2.9 Configuration Push Button

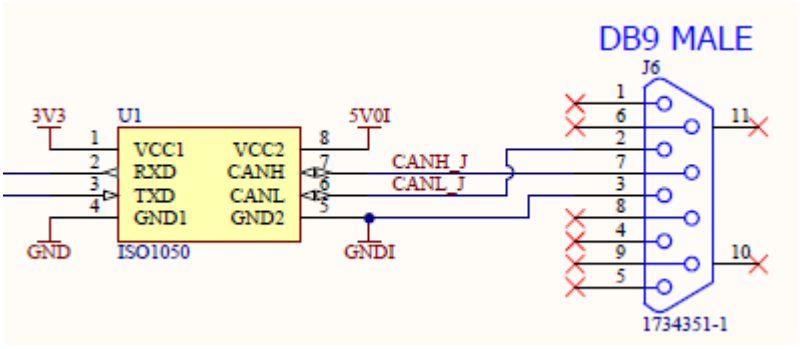
Pressing the recessed push button for **less than 3 seconds** will put the processor into configuration mode.

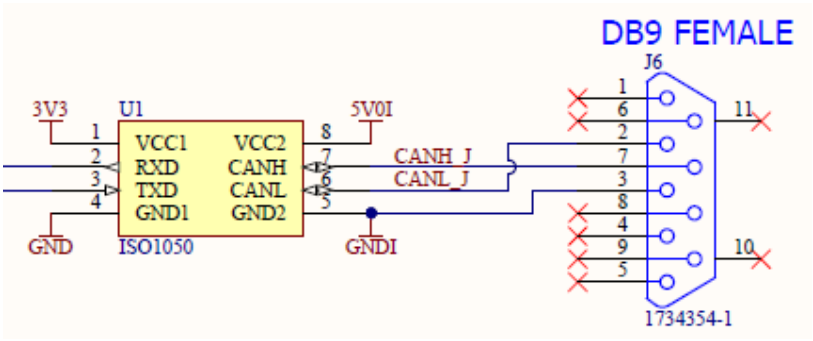
Pressing the recessed push button for **MORE than 3 seconds** will reset the unit to factory defaults.

See the Configuration chapter for detailed operation.



2.2.10 CAN Driver





The ISO1050 is a galvanically isolated CAN transceiver that meets or exceeds the specifications of the ISO 11898 standard. The device has the logic input and output buffers separated by a silicon oxide (SiO2) insulation barrier that provides galvanic isolation of up to 5000 VRMS for ISO1050DW and 2500 VRMS for ISO1050DUB and ISO1050LDW. Used in conjunction with isolated power supplies, the device prevents noise currents on a data bus or other circuits from entering the local ground and interfering with or damaging sensitive circuitry.

As a CAN transceiver, the device provides differential transmit capability to the bus and differential receive capability to a CAN controller at signaling rates up to 1 megabit per second (Mbps). Designed for operation in especially harsh environments, the device features cross-wire, overvoltage and loss of ground protection from –27 V to 40 V and over-temperature shut-down, as well as –12 V to 12 V common-mode range.

The ISO1050 is characterized for operation over the ambient temperature range of –55°C to 105°C.

- 2500-VRMS Isolation (ISO1050DUB and ISO1050LDW)
- 5000-VRMS Isolation (ISO1050DW)
- Failsafe Outputs
- Meets or Exceeds ISO 11898 requirements
- Bus-Fault Protection of –27 V to 40 V
- Dominant Time-Out Function
- IEC 60747-5-2 (VDE 0884, Rev. 2) & IEC 61010-1 Approved
- UL 1577 Double Protection Approved
- IEC 60601-1 (Medical) and CSA Approved
- 5 KVRMS Reinforced Insulation per TUV
- Approved for EN/UL/CSA 60950-1 (ISO1050DW)
- Typical 25-Year Life at Rated Working Voltage

DOMINANT TIME-OUT

A dominant time-out circuit in the ISO1050 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

If the bus-side power supply Vcc2 is lower than about 2.7V, the power shutdown circuits in the ISO1050 will disable the transceiver to prevent spurious transitions due to an unstable supply. If Vcc1 is still active when this occurs, the receiver output will go to a failsafe HIGH value in about 6 microseconds.

THERMAL SHUTDOWN

The ISO1050 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

In the CAN standard ISO 11898-2 the driver differential output is specified with a 60Ω load (must be greater than 1.5V) and with a fully-loaded bus (must be greater than 1.2V). The ISO1050 is specified to meet the 1.5V requirement with a 60Ω load, and 1.4V with a 45Ω load. The differential input resistance of the ISO1050 is a minimum of 30KΩ. If the 167 transceivers are in parallel on a bus, this is equivalent to a 180Ω differential load. That transceiver load of 180Ω in parallel with the 60Ω (two 120Ω termination resistors) gives a total 45Ω. Therefore, the ISO1050 supports over 167 transceivers on a single bus segment, with margin to the 1.2V CAN requirement.

2.2.11 TTL Option

This CAN-232 model option allows a user to connect TTL level signals to the board through the serial port cable. U4, the TTL to RS232 converter, is removed and the resistor pack, RP4 is installed. U8 is also installed for protection. This option is by special order and NOT a field option.

2.3 Entering Configuration Mode

2.3.1 RS232 Interface:

To view the Configuration menu, first connect the CAN-232 serial cable to a PC running a monitor program, such as Terra Term. Set the COM port parameters to 115200, 8, 1, None. The cable should be a null-modem since the CAN-232 is wired as DTE.

2.3.2 USB Interface:

Use Device Manager to determine which port the CAN-USB module is assigned. Device Manager will show the device as a USB Serial Port. To view the Configuration menu, first connect the CAN-USB cable to a PC running a monitor program, such as Terra Term. Set the COM port parameters to 115200, 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 #0#. Type help or ? to display the prompt messages.

Whenever the CONFIG button is pressed, the COM port settings are automatically set to 115200 N 8 1 F- (for RS232 units). This ensures that an application can smoothly change from RUN to CONFIG and back to RUN mode without needing to change COM port settings.

Note : For USB devices, the COM port settings have no effect as the actual baud rate is the full-USB speed at all times.

```
                                    C A N - V C C M - P R O
                Application Ver : 01.02     Bootloader Ver : 01.01     Hardware Ver : A
                              B R I E F   C O M M A N D   S U M M A R Y
 (00) : get mode
 (01) : set mode vc | cm

 (02) : get com port
 (03) : set com port baud parity bits stop flow

 (04) : get can port
 (05) : set can port baud %sample-point
 (06) : set can port baud %sample-point tol
 (07) : set can port baud %sample-point ?
 (08) : set can port baud %sample-point tol ?

 (09) : set can port (CLK_DIV BRG TSEG1 TSEG2 SJW)

 (10) : get can cm
 (11) : set can cm filter=ON | OFF
 (12) : set can cm mode=NORMAL | MONITOR
 (13) : set can cm ifmt=ASCII | BINARY
 (14) : set can cm ofmt=ASCII | BINARY
 (15) : set can cm eol=CR | LF | CRLF | LFCR | NONE

 (16) : get can vc
 (17) : set can vc tx=id-type id-val
 (18) : set can vc rx=id-type id-val
 (19) : set can vc fs=code
 (20) : set can vc fw=code
 (21) : set can vc ts=timeout
 (22) : set can vc tw=timeout
 (23) : set can vc ww=timeout
 (24) : set can vc wmsg=can-msg

 (25) : get can timeout
 (26) : set can timeout can_tx_busy=timeout

 (27) : get can filter
 (28) : set can filter std id | ext id | std id-min id-max | ext id-min id-max
 (29) : del can filter #
 (30) : del can filter # #
 (31) : del can filter all

 (32) : get cfgcmd cm
 (33) : set cfgcmd cm enable=YES | NO

 (34) : get cfgcmd vc
 (35) : set cfgcmd vc enable=YES | NO
 (36) : set cfgcmd vc tbeg=timeout
 (37) : set cfgcmd vc tend=timeout
 (38) : set cfgcmd vc tid=timeout
 (39) : set cfgcmd vc seq="sequence"

 (40) : profile #
 (41) : profile copy # #
 (42) : profile note
 (43) : profile note #
 (44) : profile note all
 (45) : profile note="text"
 (46) : profile note #="text"

 (47) : default
 (48) : default #
 (49) : default all

 (50) : get sernum
 (51) : get version

 (52) : test
 (53) : test bps

 (54) : exit
```

```
*------------------------------------------------------------------------------------------*
|                                        N O T E S                                          |
*------------------------------------------------------------------------------------------*
| - View command detail by typing 'help #' where '#' is a number from the above list.       |
|   Ex 1 : help 10                                                                           |
| - Most commands can be entered in angle brackets to avoid echo of the individual characters.  When |
|   angled, no editing or input key echo is performed.  Commands will execute on the '>' terminator. |
|   No CRLF will be issued and input prompting will be suspened until a CRLF sequence is seem.  |
|   Ex 2 : <set com port 115200 N 8 1 F->                                                    |
| - Normal input has limited line editing capability:                                        |
|   ESC    : Deletes entire input line.                                                       |
|   BSPACE : Deletes last entered character.                                                  |
|   Note   : In angled mode, editing is disabled.                                              |
| - Profiles are numbered from 0 to 9.                                                        |
*------------------------------------------------------------------------------------------*
```

2.4 Mode Options

2.4.1 Get Mode (00)

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

In virtual circuit mode (VC), the CAN USB-232 establishes a full-duplex, virtual circuit between itself and another CAN USB-232 or application device. By providing a virtual circuit over the CAN network, applications can exchange data in a network-transparent fashion, using existing CAN network cabling as a data link.

```
*------------------------------------------------------------------------------------------*
| Command : get mode                                                                          |
*------------------------------------------------------------------------------------------*
| Desc : Gets the current operating mode (CM or VC).                                         |
| Ex 1 : get mode                                                                             |
| Resp : <A:CM> or <A:VC>                                                                     |
*------------------------------------------------------------------------------------------*
```

2.4.2 Set Mode (01)

```
*------------------------------------------------------------------------------------------*
| Command : set mode                                                                          |
*------------------------------------------------------------------------------------------*
| Desc : Sets the current operating mode (CM or VC).                                         |
| Ex 1 : set mode cm                                                                          |
| Ex 2 : set mode vc                                                                          |
| Resp : <A:CM> or <A:VC>                                                                     |
*------------------------------------------------------------------------------------------*
```

2.5 COM Options

2.5.1 Get Com Port (02)

```
*------------------------------------------------------------------------------------------*
| Desc : Gets the current COM port settings                                                  |
| Ex 1 : get com port                                                                         |
| Resp : <A:115200 N 8 1 F- -0.27%>                                                           |
|               |    | | | |    +-> baud rate % tolerance                                     |
|               |    | | | +------> HW flow control (F+ = ON , F- = OFF)                       |
|               |    | | +--------> stop bits (1 or 2)                                         |
|               |    | +----------> data bits (7 or 8)                                         |
|               |    +------------> parity (N = none, O = odd , E = even , 0 = force-0 , 1 = force-1 |
|               +-----------------> baudrate (300 to 1000000)                                 |
*------------------------------------------------------------------------------------------*
```

2.5.2 Set Com Port (03)

Configuration of the COM port is simple and consists of specifying the serial baudrate, parity, the number of data bits, stop bits, Hardware flow control. The response will echo the selected settings and the baud rate tolerance percent.

```
Command : set com port
Desc : Sets the current COM port settings
Ex 1 : set com port 115200 N 8 1 F-
Resp : <A:115200 N 8 1 F- -0.27%>
                    |  |  |  |    |
                    |  |  |  |    +-> baud rate % tolerance
                    |  |  |  +------> HW flow control (F+ = ON , F- = OFF)
                    |  |  +---------> stop bits (1 or 2)
                    |  +------------> data bits (7 or 8)
                    +---------------> parity (N = none, O = odd , E = even , 0 = force-0 , 1 = force-1
                 +------------------> baudrate (300 to 1000000)
```

2.6 CAN Port Options

2.6.1 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 must be correctly specified for the specific characteristics of the network. The CAN USB-232 provides a means of configuring these parameters so as to support a wide array of network scenarios.

Note that the CAN bit rate settings can be specified as a bit-rate, sampling point, and tolerance. The code will find the best fit solution.

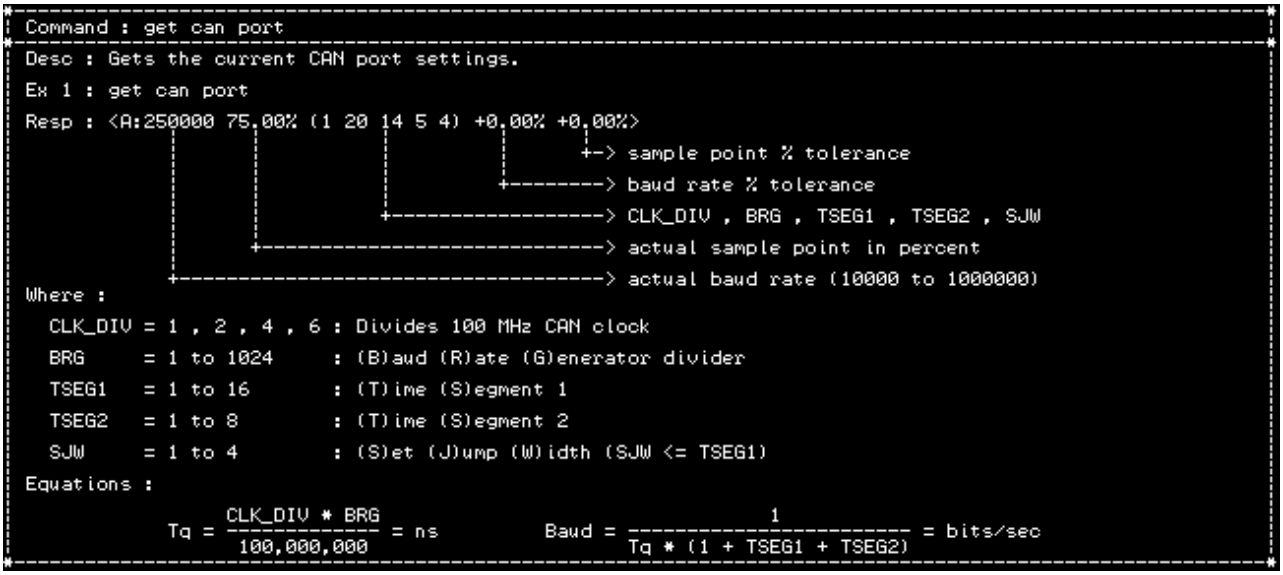
2.6.2 CAN Bus State During Configuration

While the CAN USB-232 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 while still connected to an active CAN network and not adversely affect network operation.

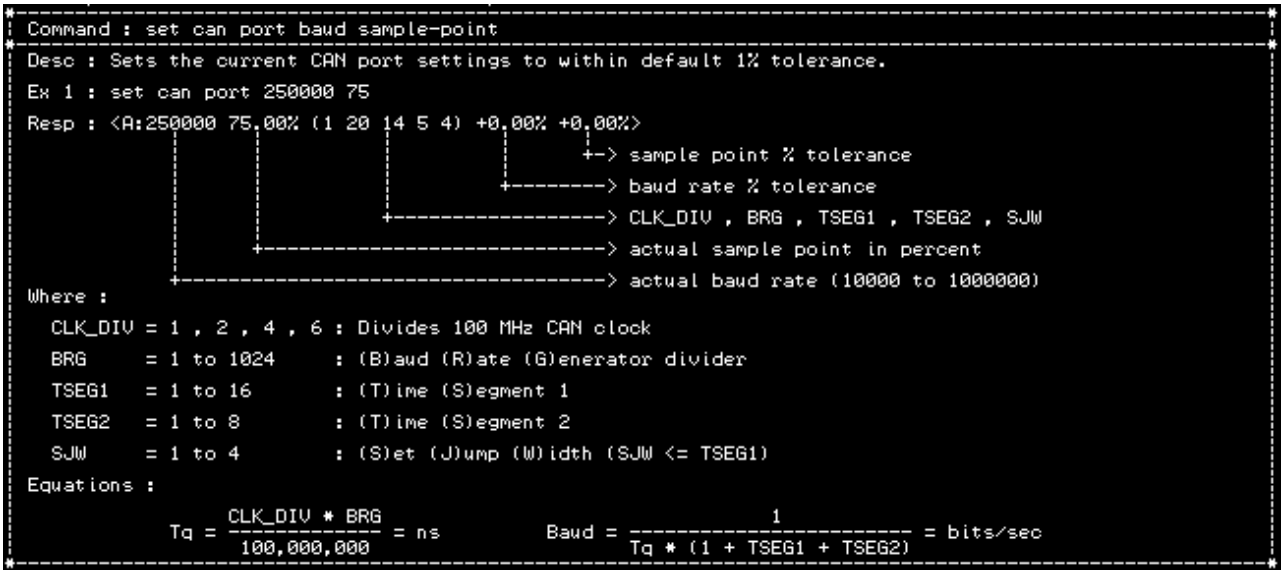
Once the configuration state is exited, the CAN USB-232 will activate the CAN controller and begin immediately interacting with the CAN network according to the CAN 2.0A/B specification.

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

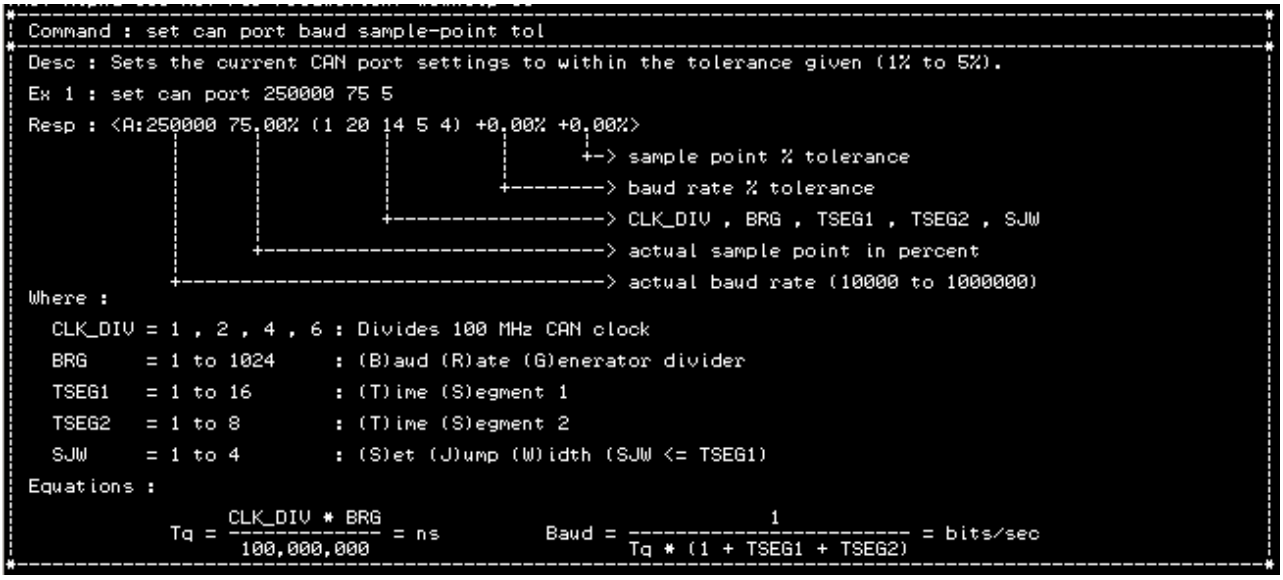
2.6.3 Get CAN Port (04)



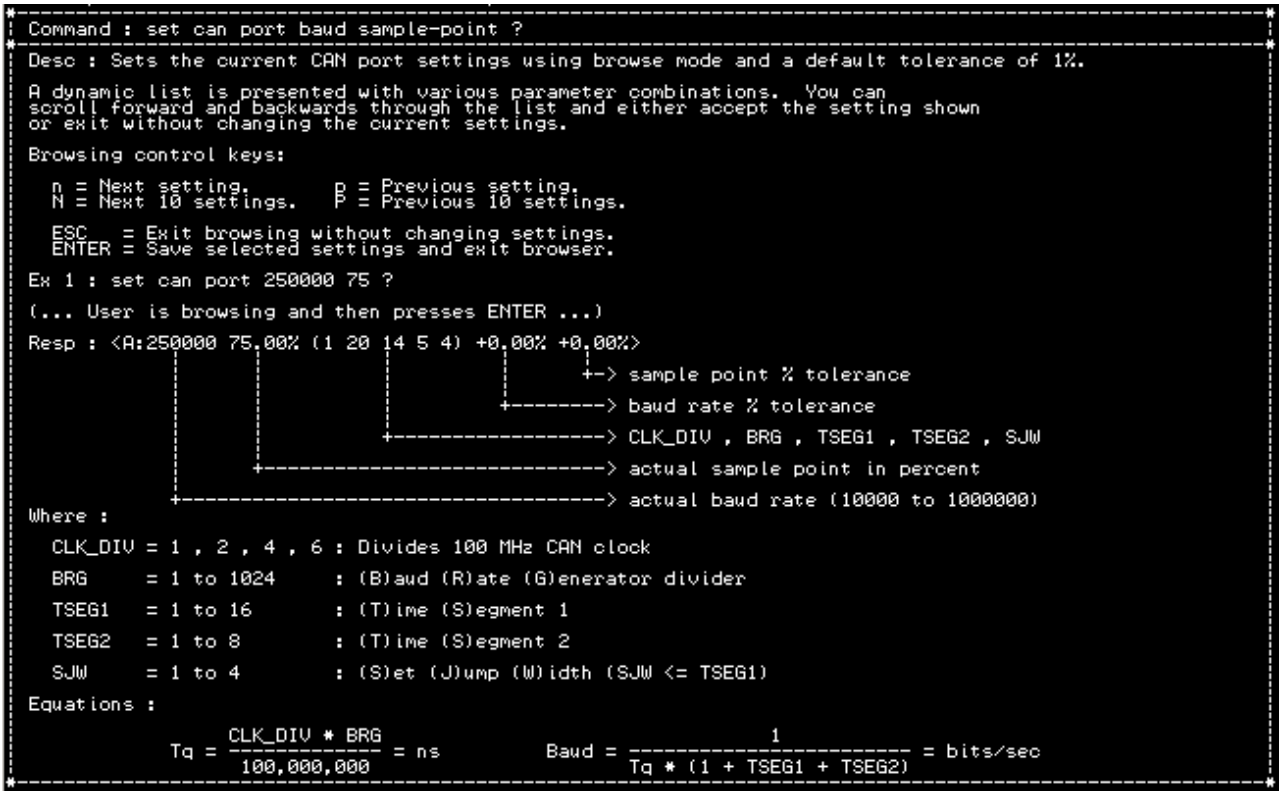
2.6.4 Set CAN Port Baud Sample Point (05)



2.6.5 Set CAN Port Baud Sample Point Tolerance (06)



2.6.6 Set CAN Port Baud Sample Point Browse (07)



2.6.7 Set CAN Port Baud Sample Point Tolerance (08)

```
Command : set can port baud sample-point tol ?

Desc : Sets the current CAN port settings using browse mode and within the given tolerance (1% to 5%).

A dynamic list is presented with various parameter combinations.  You can
scroll forward and backwards through the list and either accept the setting shown
or exit without changing the current settings.

Browsing control keys:

  n = Next setting.       p = Previous setting.
  N = Next 10 settings.   P = Previous 10 settings.

  ESC   = Exit browsing without changing settings.
  ENTER = Save selected settings and exit browser.

Ex 1 : set can port 250000 75 5 ?

(... User is browsing and then presses ENTER ...)

Resp : <A:250000 75.00% (1 20 14 5 4) +0.00% +0.00%>
            |      |      |      |      |      |
            |      |      |      |      |      +-> sample point % tolerance
            |      |      |      |      +--------> baud rate % tolerance
            |      |      |      +---------------> CLK_DIV , BRG , TSEG1 , TSEG2 , SJW
            |      |      +----------------------> actual sample point in percent
            |      +-----------------------------> actual baud rate (10000 to 1000000)
Where :
  CLK_DIV = 1 , 2 , 4 , 6 : Divides 100 MHz CAN clock
  BRG     = 1 to 1024     : (B)aud (R)ate (G)enerator divider
  TSEG1   = 1 to 16       : (T)ime (S)egment 1
  TSEG2   = 1 to 8        : (T)ime (S)egment 2
  SJW     = 1 to 4        : (S)et (J)ump (W)idth (SJW <= TSEG1)
Equations :
                CLK_DIV * BRG                                  1
           Tq = ------------- = ns          Baud = ------------------------ = bits/sec
                100,000,000                       Tq * (1 + TSEG1 + TSEG2)
```

2.6.8 Set CAN Port (CLK_DIV BRG TSEG1 TSEG2 SJW) (09)

```
Command : set can port (CLK_DIV BRG TSEG1 TSEG2 SJW)

Desc : Sets the current CAN port settings using exact HW parameters.

Ex 1 : set can port (1 20 14 5 4)

Resp : <A:250000 75.00% (1 20 14 5 4) +0.00% +0.00%>
            |      |      |      |      |      |
            |      |      |      |      |      +-> sample point % tolerance
            |      |      |      |      +--------> baud rate % tolerance
            |      |      |      +---------------> CLK_DIV , BRG , TSEG1 , TSEG2 , SJW
            |      |      +----------------------> actual sample point in percent
            |      +-----------------------------> actual baud rate (10000 to 1000000)
Where :
  CLK_DIV = 1 , 2 , 4 , 6 : Divides 100 MHz CAN clock
  BRG     = 1 to 1024     : (B)aud (R)ate (G)enerator divider
  TSEG1   = 1 to 16       : (T)ime (S)egment 1
  TSEG2   = 1 to 8        : (T)ime (S)egment 2
  SJW     = 1 to 4        : (S)et (J)ump (W)idth (SJW <= TSEG1)
Equations :
                CLK_DIV * BRG                                  1
           Tq = ------------- = ns          Baud = ------------------------ = bits/sec
                100,000,000                       Tq * (1 + TSEG1 + TSEG2)
```

The ‘TSEG1’ parameter determines the amount of time to wait before the hardware attempts to sample the bit. The value can range from 1 to 16.

The ‘TSEG2’ parameter determines the amount of time remaining before the end of the bit. The value can range from 1 to 8.

The SJW parameter controls the maximum allowable adjustment to the sampling point. This allows for a CAN node to adjust its sampling point when it determines that the sample point would be too early or too late. By setting the SJW field to a large number, the CAN USB-232 can work with other nodes with a large variation is bus oscillator tolerances. The value can range from 1 to 4.

2.7 CAN Command Mode

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

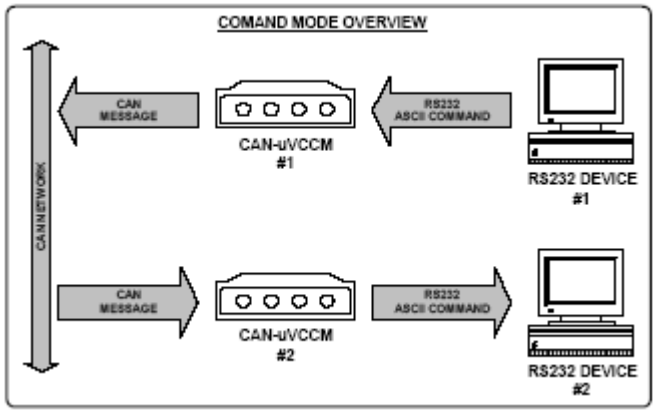


Figure 3

When the CAN USB-232 receives a valid ASCII message string, 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, it converts it to an ASCII message string and transmits it out of the serial port.

The CAN USB-232 supports ten (10) CAN receive message filters consisting of STD, EXT, STD-RANGE, or EXT-RANGE. By using the masks to specify which bits of an identifier are to be compared to the filter value, the CAN USB-232 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.

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 Set CAN Command Mode Message Output Termination (15) on page 2-30.

2.7.1 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 identifier value and the number of data bytes included in the message.

Messages starting with '|' will generate a self-receive of the transmitted message. If the message passes filtering, it will be received as if it had been sent from some other node. Terminating with '!' instead of ';' issues a on-shot transmit. Ie. No attempts to automatically retry on error will happen. This is used in time-triggered CAN protocols.

Terminating with '!' instead of ';' issues a on-shot transmit. 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 length field is treated as a base-10 digit and must be a single digit between 0 and 8.

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

There are two types of message strings:

- Normal CAN messages

- Request-To-Transmit CAN messages (RTR)

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.

2.7.2 Normal CAN Message

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

Normal CAN Message Syntax

: <S | X> <IDENTIFIER> <N> <DATA-0> <DATA-1> … <DATA-7> ;

The first character ,':', is for synchronization and allows the CAN USB-232 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 'N' indicates that the message is a normal (non-RTR) transmission.

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.

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

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

2.7.2.1 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.

2.7.3 RTR CAN Message

A RTR CAN message consists of the type (11-bit or 29-bit), identifier, 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 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 'LENGTH' field is a single ASCII decimal character from '0' to '8' that specifies the length of the message.

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

2.7.3.1 Examples

Example #1

```
:S123R8;
```

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

Example #2

```
:XF00DR0;
```

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

2.7.4 Appending CR/LF To Received Command Strings

When using the CAN USB-232 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 Set CAN Command Mode Message Output Termination (15) on page 2-30.

2.7.5 Binary Formatted Messages

The tables below show the format for Binary message format.

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	ONE-SHOT	SELF-RCEV	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	ONE-SHOT	SELF-RCEV	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	ONE-SHOT	SELF-RCEV	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								

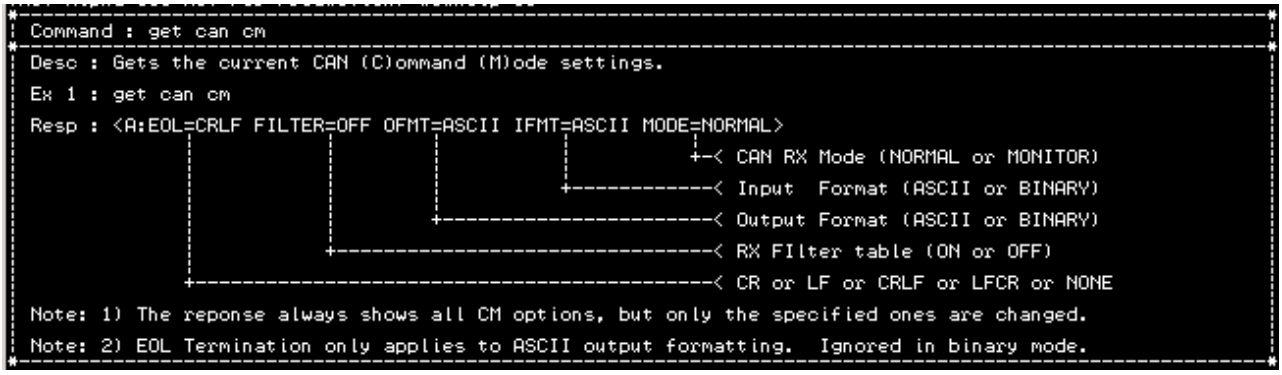
EXT-RTR Message Format

	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	ONE-SHOT	SELF-RCEV	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)								

DLE Sequences Defined

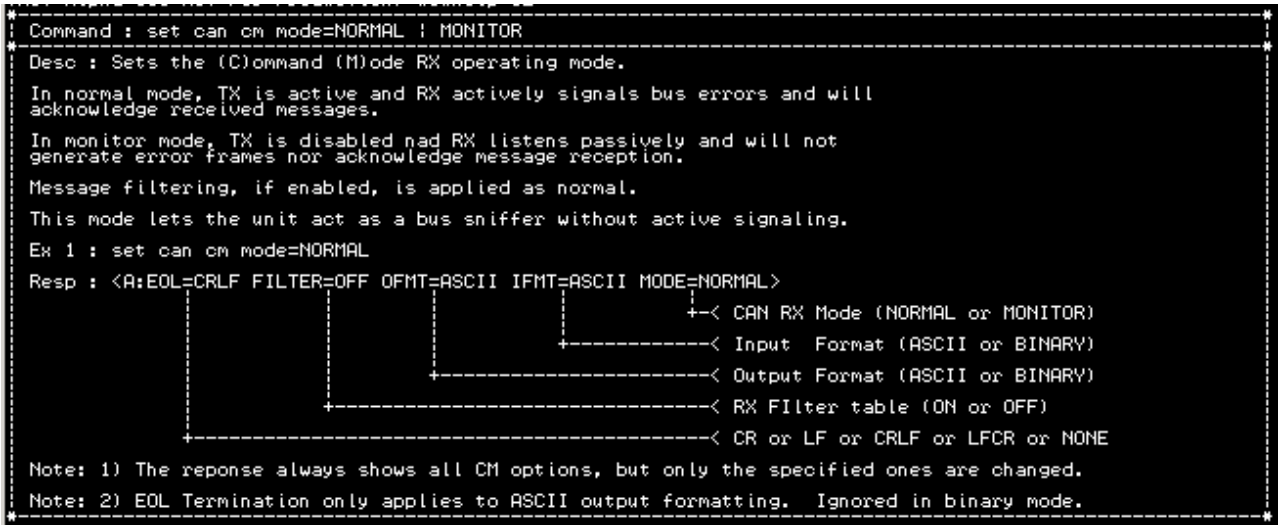
Name	Byte-0	Byte-1	Type	Comment
SYNC	$FF	$00	FRAME	Marks start of message
DATA	$FF	$01	DATA	Defines data value of $FF
DLE RE-SYNC	$FF	$FF	ERROR	Restarts DLE sequence parsing

2.7.6 Get CAN Command Mode Settings (10)

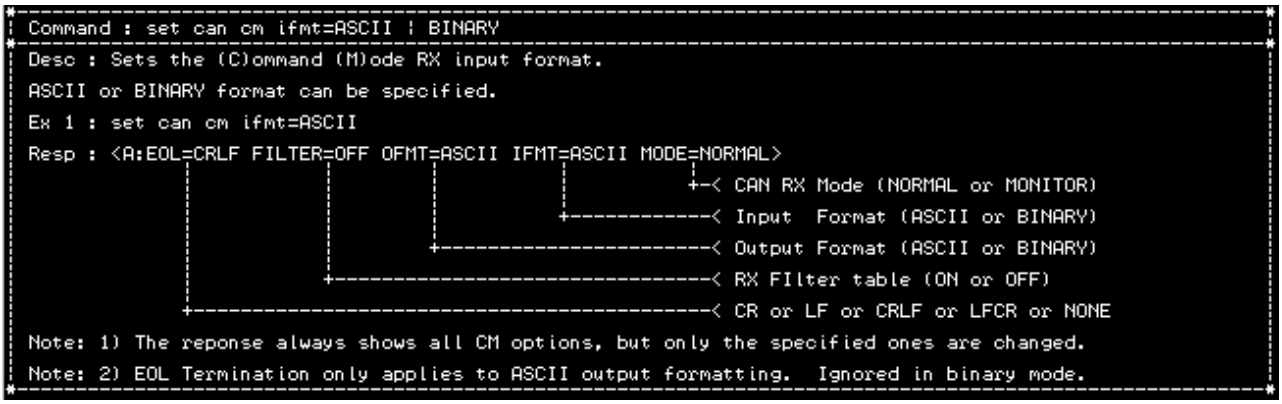


2.7.7 Set CAN Command Mode Filter On/Off (11)

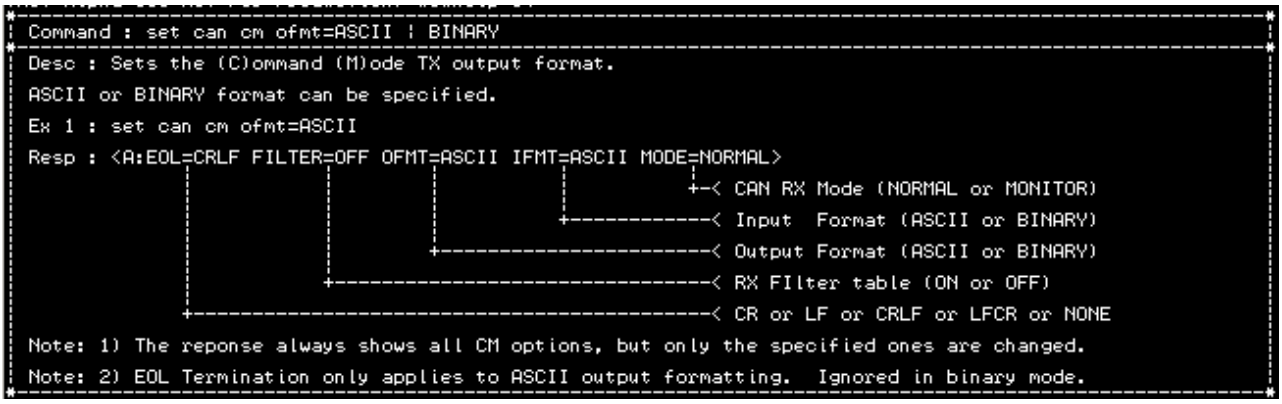
2.7.8 Set CAN Command Mode RX Operating Mode (12)



2.7.9 Set CAN Command Mode RX Input Format (13)



2.7.10 Set CAN Command Mode TX Output Format (14)



2.7.11 Set CAN Command Mode Message Output Termination (15)

```
Command : set can cm eol=CR | LF | CRLF | LFCR | NONE

Desc : Sets the (C)ommand (M)ode message output termination.

If output formatting is in ASCII, then this option determines
how to terminate the message.

Ex 1 : set can cm eol=CRLF

Resp : <A:EOL=CRLF FILTER=OFF OFMT=ASCII IFMT=ASCII MODE=NORMAL>
                                          +-< CAN RX Mode (NORMAL or MONITOR)
                                    +-----< Input  Format (ASCII or BINARY)
                              +-----------< Output Format (ASCII or BINARY)
                        +-----------------< RX FIlter table (ON or OFF)
                  +-----------------------< CR or LF or CRLF or LFCR or NONE

Note: 1) The reponse always shows all CM options, but only the specified ones are changed.
Note: 2) EOL Termination only applies to ASCII output formatting.  Ignored in binary mode.
```

2.8 CAN Virtual Command Mode

In virtual circuit mode, the CAN USB-232 establishes a full-duplex, virtual circuit between itself and another CAN USB-232 or application device. By providing a virtual circuit over the CAN network, applications can exchange serial stream data in a network-transparent fashion, using existing CAN network cabling as a data link (Figure 4).

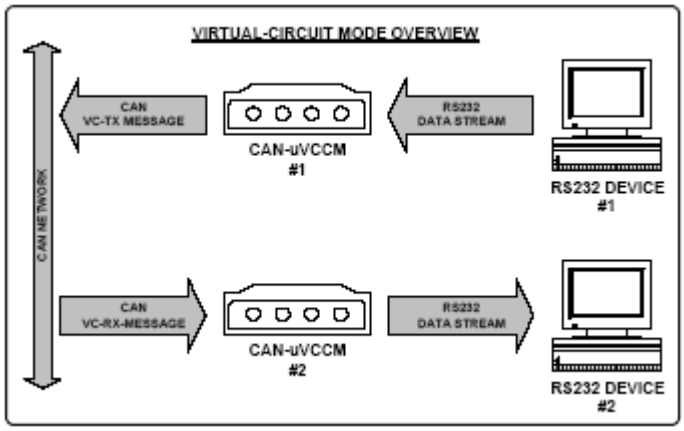


Figure 4

The CAN USB-232 takes data bytes coming into the serial port and groups them into CAN messages for transmission. When a remote CAN USB-232 or application target receives these messages, it extracts the data bytes and recreates the original data stream on its serial output port. This operation is fully transparent to the connected application devices.

The virtual circuit requires two, user-configurable, CAN identifiers in order to identify the source & destination devices for the circuit. Since virtually every CAN protocol provides a sub-set of unused or 'user-specific' identifiers, establishing a virtual circuit on a CAN network in conjunction with an existing protocol is supported and very easy to do.

The CAN USB-232 only sends CAN messages when eight bytes of data have 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. To deal with this in a transparent fashion, the CAN USB-232 provides a timeout feature that will automatically force a transmission of the last accumulated byte(s) after a specified maximum waiting time.

The CAN USB-232 can also be configured to force immediate transmission upon detection of specific user-configured bytes in the data stream (CR or LF, for example).

It also supports defining and sending of WAKE messages to support CAN networks that go into SLEEP mode. See more detail in help options under 'set can vc'

2.8.1 The 'XMIT' and 'RCEV' Identifiers

When configuring the CAN USB-232 for Virtual Circuit operation, two identifiers must be specified in order for the virtual circuit to work correctly. These two identifiers are referred to as the 'XMIT' and 'RCEV' identifiers.

2.8.2 'XMIT' Identifier

The 'XMIT' identifier is used when the CAN USB-232 has stream data to send over the CAN network.

A CAN message is formatted to include this identifier, the number of bytes to be sent, and the actual stream data that was received from the serial COM port. The CAN message is then sent over the CAN network.

This identifier signals that the CAN message which contains it also contains serial port stream data. Other CAN USB-232 units can be configured to look for this identifier and will then extract the stream data from the CAN message.

The 'XMIT' identifier can be either standard 11-bit or extended 29-bit.

2.8.3 'RCEV' Identifier

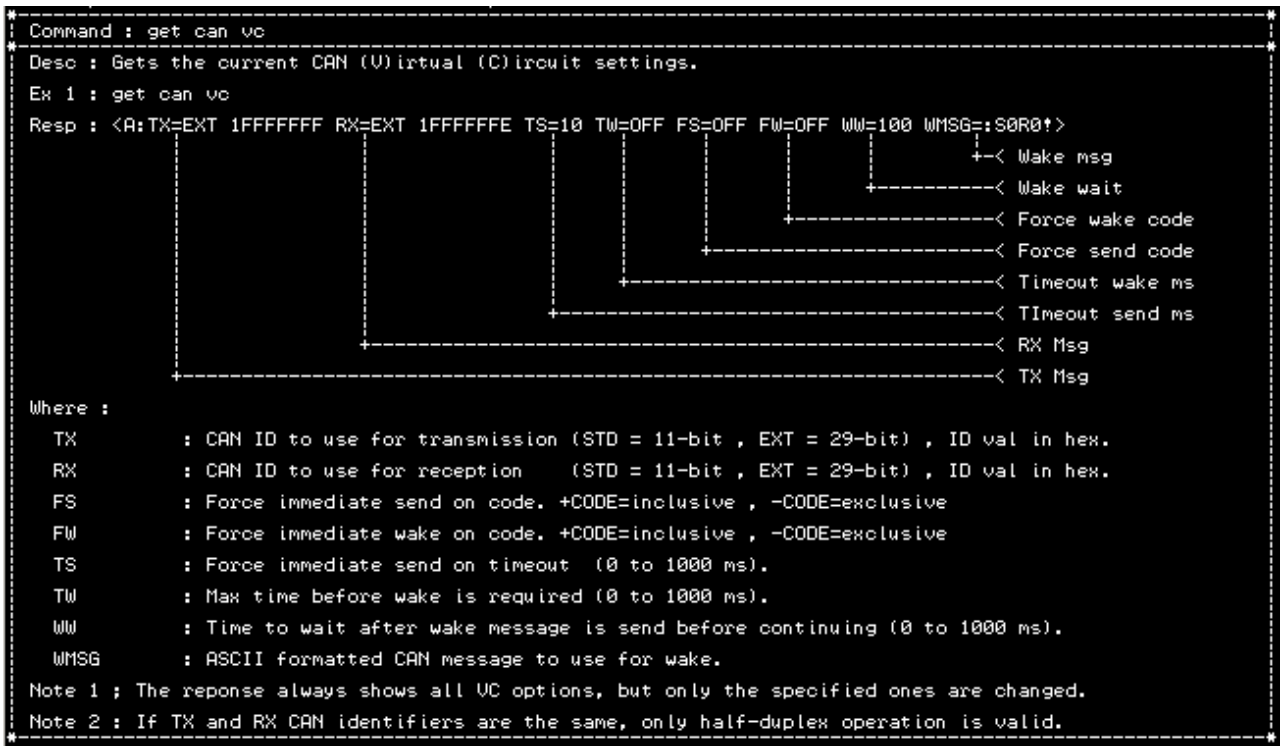
The 'RCEV' identifier is used to specify which CAN message should be received by the CAN USB-232 when waiting for incoming stream data. The CAN USB-232 will discard all other messages.

When a CAN message containing this identifier is received, the data in the message is assumed to be stream data from another CAN USB-232. This data is then read and output directly to the serial COM port of the receiving CAN USB-232 unit, providing the completion of the virtual circuit.

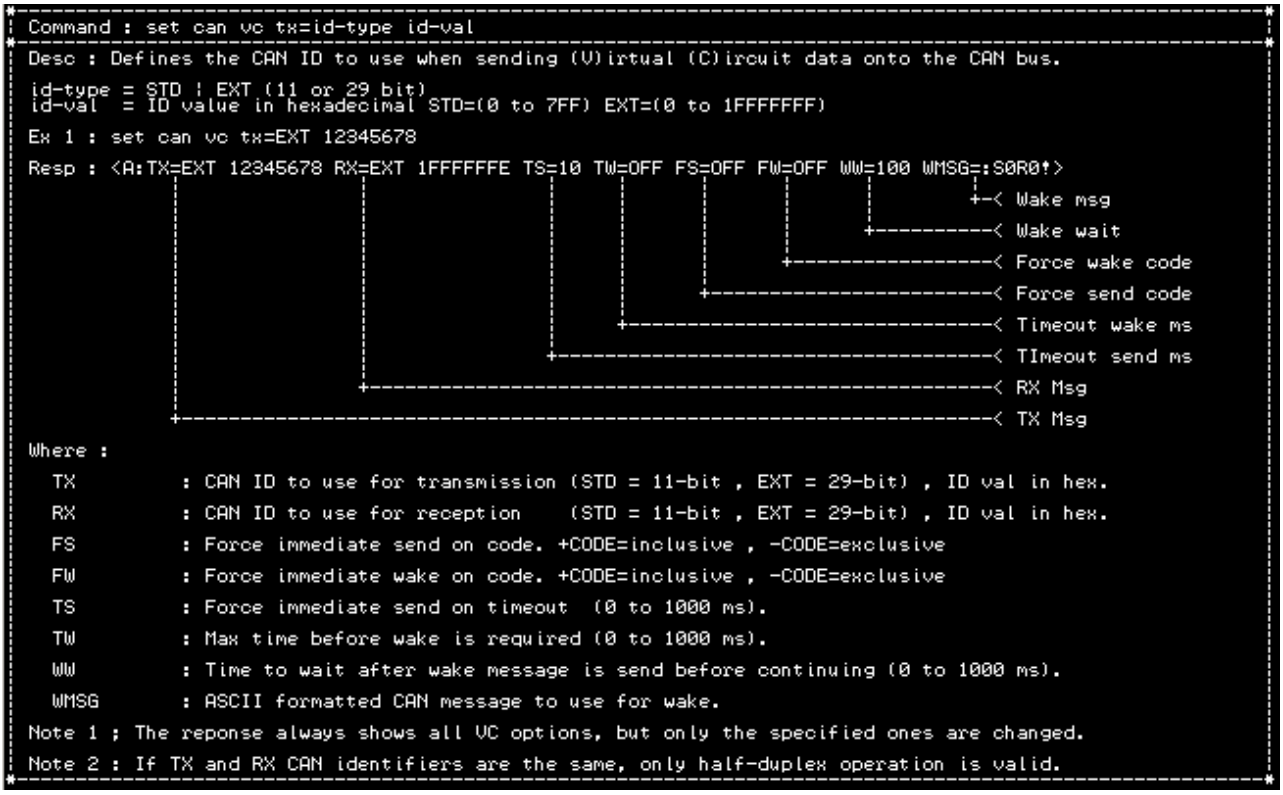
If the received message for any reason contains a length of zero, or is an RTR message, it is discarded and not stream data is generated.

The 'RCEV' identifier can be either standard 11-bit or extended 29-bit.

2.8.4 Get CAN Virtual Circuit Setting (16)



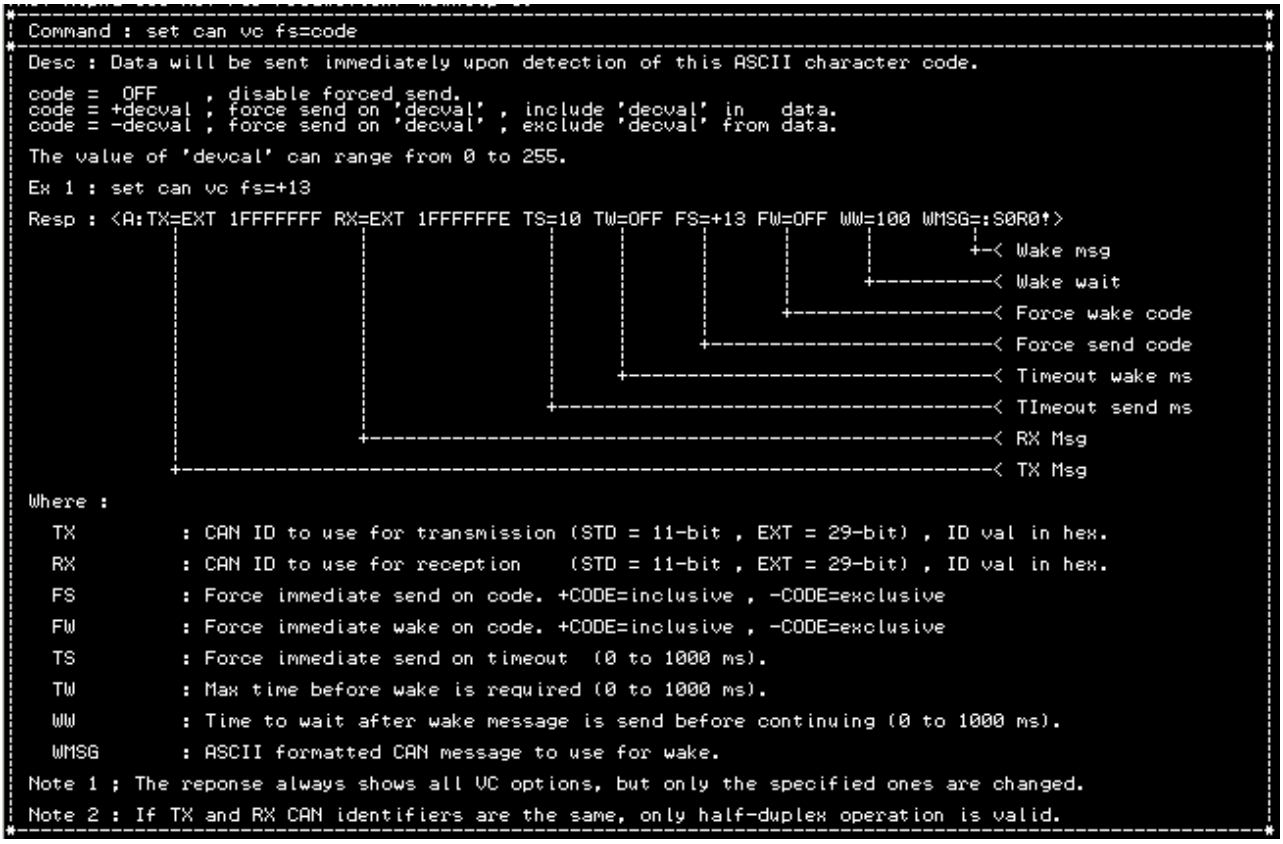
2.8.5 Set CAN Virtual Circuit TX ID (17)



2.8.6 Set CAN Virtual Circuit RX ID (18)

```
Command : set can vc rx=id-type id-val

Desc : Defines the CAN ID to use when receiving (V)irtual (C)ircuit data from the CAN bus.

id-type = STD | EXT (11 or 29 bit)
id-val  = ID value in hexadecimal STD=(0 to 7FF) EXT=(0 to 1FFFFFFF)

Ex 1 : set can vc rx=EXT 12345678

Resp : <A:TX=EXT 1FFFFFFF RX=EXT 12345678 TS=10 TW=OFF FS=OFF FW=OFF WW=100 WMSG=:S0R0↑>
                                                                    +-< Wake msg
                                                               +----------< Wake wait
                                                          +-----------------< Force wake code
                                                     +------------------------< Force send code
                                                +-------------------------------< Timeout wake ms
                                           +------------------------------------< TImeout send ms
                                      +-------------------------------------------< RX Msg
                                 +-------------------------------------------------< TX Msg

Where :
  TX         : CAN ID to use for transmission (STD = 11-bit , EXT = 29-bit) , ID val in hex.
  RX         : CAN ID to use for reception    (STD = 11-bit , EXT = 29-bit) , ID val in hex.
  FS         : Force immediate send on code. +CODE=inclusive , -CODE=exclusive
  FW         : Force immediate wake on code. +CODE=inclusive , -CODE=exclusive
  TS         : Force immediate send on timeout  (0 to 1000 ms).
  TW         : Max time before wake is required (0 to 1000 ms).
  WW         : Time to wait after wake message is send before continuing (0 to 1000 ms).
  WMSG       : ASCII formatted CAN message to use for wake.
Note 1 ; The reponse always shows all VC options, but only the specified ones are changed.
Note 2 : If TX and RX CAN identifiers are the same, only half-duplex operation is valid.
```

2.8.7 Set CAN Virtual Circuit Forced Send Code (19)



```
Command : set can vc fs=code

Desc : Data will be sent immediately upon detection of this ASCII character code.

code =  OFF    , disable forced send.
code = +decval , force send on 'decval' , include 'decval' in   data.
code = -decval , force send on 'decval' , exclude 'decval' from data.

The value of 'devcal' can range from 0 to 255.

Ex 1 : set can vc fs=+13

Resp : <A:TX=EXT 1FFFFFFF RX=EXT 1FFFFFFE TS=10 TW=OFF FS=+13 FW=OFF WW=100 WMSG=:S0R0!>
                                                                        +-< Wake msg
                                                                  +-----------< Wake wait
                                                            +-----------------< Force wake code
                                                      +-------------------------< Force send code
                                                +-------------------------------< Timeout wake ms
                                          +-------------------------------------< TImeout send ms
                                    +-----------------------------------------------< RX Msg
                              +-------------------------------------------------------< TX Msg

Where :
  TX         : CAN ID to use for transmission (STD = 11-bit , EXT = 29-bit) , ID val in hex.
  RX         : CAN ID to use for reception    (STD = 11-bit , EXT = 29-bit) , ID val in hex.
  FS         : Force immediate send on code. +CODE=inclusive , -CODE=exclusive
  FW         : Force immediate wake on code. +CODE=inclusive , -CODE=exclusive
  TS         : Force immediate send on timeout  (0 to 1000 ms).
  TW         : Max time before wake is required (0 to 1000 ms).
  WW         : Time to wait after wake message is send before continuing (0 to 1000 ms).
  WMSG       : ASCII formatted CAN message to use for wake.
Note 1 ; The reponse always shows all VC options, but only the specified ones are changed.
Note 2 : If TX and RX CAN identifiers are the same, only half-duplex operation is valid.
```

2.8.8 Set CAN Virtual Circuit Forced Wake Code (20)

```
Command : set can vc fw=code

Desc : Wake message will be sent immediately upon detection of this ASCII character code.

code =  OFF    , disable forced wake.
code = +decval , force wake on 'decval' , include 'decval' in   data.
code = -decval , force wake on 'decval' , exclude 'decval' from data.

The value of 'devcal' can range from 0 to 255.

Ex 1 : set can vc fw=-13

Resp : <A:TX=EXT 1FFFFFFF RX=EXT 1FFFFFFE TS=10 TW=OFF FS=OFF FW=-13 WW=100 WMSG=:S0R0!>
                                                                      +-< Wake msg
                                                                +-----< Wake wait
                                                          +-----------< Force wake code
                                                    +-----------------< Force send code
                                              +-----------------------< Timeout wake ms
                                        +-----------------------------< TImeout send ms
                                  +-----------------------------------< RX Msg
                            +-----------------------------------------< TX Msg

Where :
  TX         : CAN ID to use for transmission (STD = 11-bit , EXT = 29-bit) , ID val in hex.
  RX         : CAN ID to use for reception    (STD = 11-bit , EXT = 29-bit) , ID val in hex.
  FS         : Force immediate send on code. +CODE=inclusive , -CODE=exclusive
  FW         : Force immediate wake on code. +CODE=inclusive , -CODE=exclusive
  TS         : Force immediate send on timeout  (0 to 1000 ms).
  TW         : Max time before wake is required (0 to 1000 ms).
  WW         : Time to wait after wake message is send before continuing (0 to 1000 ms).
  WMSG       : ASCII formatted CAN message to use for wake.
Note 1 ; The reponse always shows all VC options, but only the specified ones are changed.
Note 2 : If TX and RX CAN identifiers are the same, only half-duplex operation is valid.
```

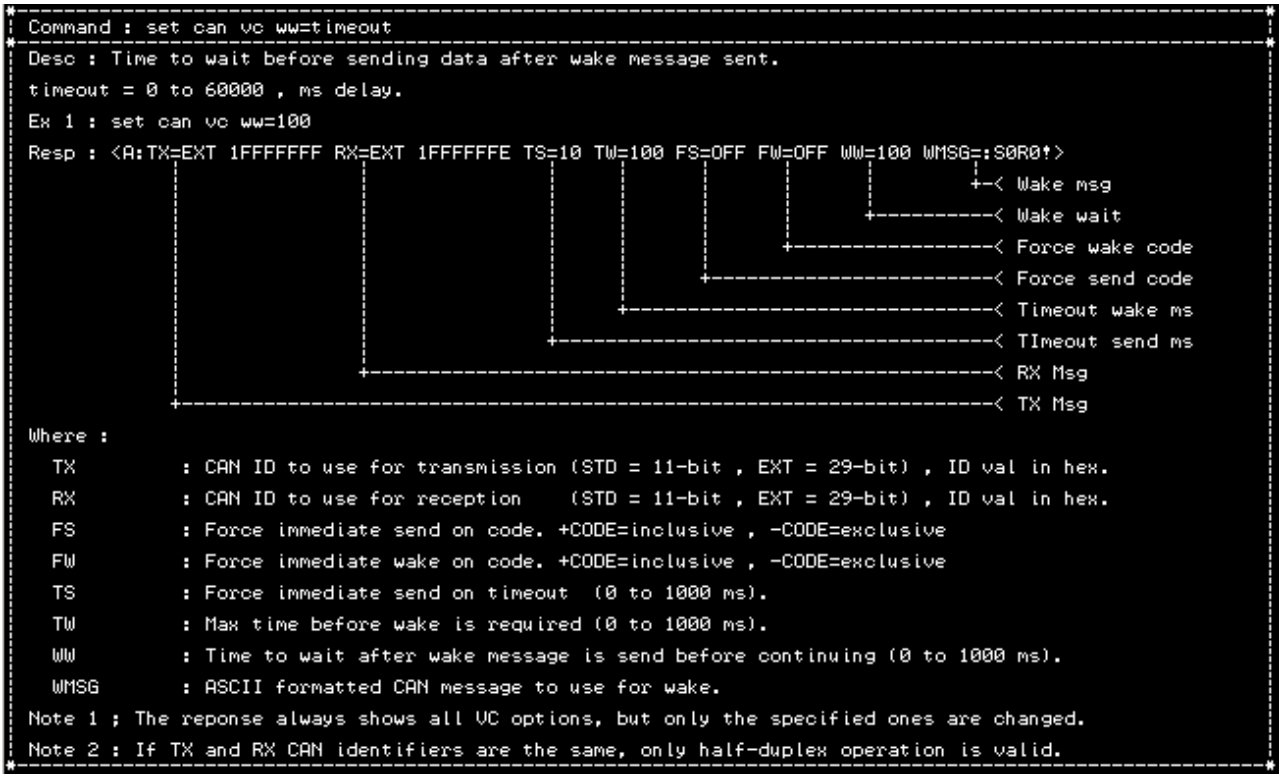
2.8.9 Set CAN Virtual Circuit Timeout Send (21)

```
Command : set can vc ts=timeout
Desc : Partial data packet will be sent if this timeout occurs.

timeout = OFF       , disabled.
timeout = 0 to 1000 , ms to wait before send.

Ex 1 : set can vc ts=10

Resp : <A:TX=EXT 1FFFFFFF RX=EXT 1FFFFFFE TS=10 TW=OFF FS=OFF FW=OFF WW=100 WMSG=:S0R0!>
                                                                        +-< Wake msg
                                                                  +-----< Wake wait
                                                            +-----------< Force wake code
                                                      +-----------------< Force send code
                                                +-----------------------< Timeout wake ms
                                          +-----------------------------< TImeout send ms
                                    +-----------------------------------< RX Msg
                              +-----------------------------------------< TX Msg

Where :
  TX         : CAN ID to use for transmission (STD = 11-bit , EXT = 29-bit) , ID val in hex.
  RX         : CAN ID to use for reception    (STD = 11-bit , EXT = 29-bit) , ID val in hex.
  FS         : Force immediate send on code. +CODE=inclusive , -CODE=exclusive
  FW         : Force immediate wake on code. +CODE=inclusive , -CODE=exclusive
  TS         : Force immediate send on timeout  (0 to 1000 ms).
  TW         : Max time before wake is required (0 to 1000 ms).
  WW         : Time to wait after wake message is send before continuing (0 to 1000 ms).
  WMSG       : ASCII formatted CAN message to use for wake.
Note 1 ; The reponse always shows all VC options, but only the specified ones are changed.
Note 2 : If TX and RX CAN identifiers are the same, only half-duplex operation is valid.
```

2.8.10 Set CAN Virtual Circuit Wake Timeout (22)

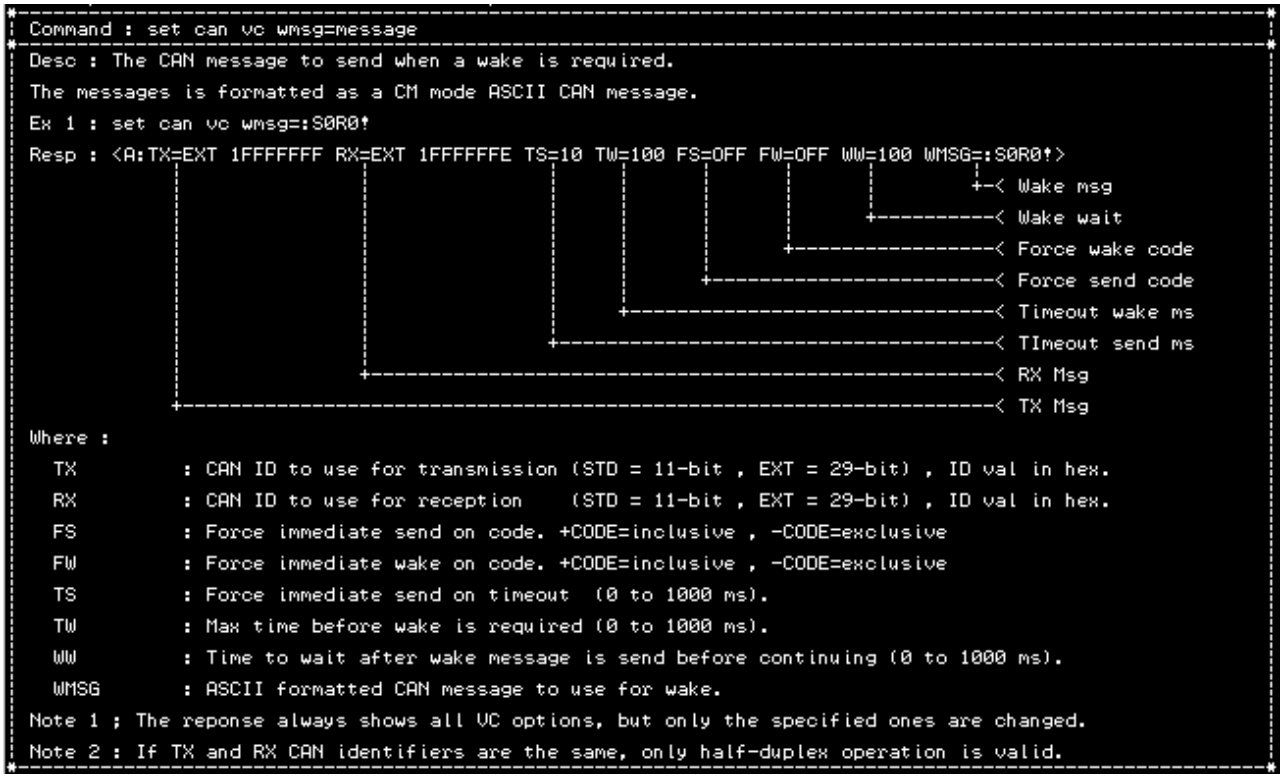
```
Command : set can vc tw=timeout

Desc : Maximum CAN bus idle time before wake is required.

timeout = OFF       , disabled.
timeout = 0 to 60000 , ms window.

Ex 1 : set can vc tw=100

Resp : <A:TX=EXT 1FFFFFFF RX=EXT 1FFFFFFE TS=10 TW=100 FS=OFF FW=OFF WW=100 WMSG=:S0R0!>
                                                                  +-< Wake msg
                                                            +-----< Wake wait
                                                      +-----------< Force wake code
                                                +-----------------< Force send code
                                          +-----------------------< Timeout wake ms
                                    +-----------------------------< TImeout send ms
                              +-----------------------------------< RX Msg
                        +-----------------------------------------< TX Msg

Where :
  TX         : CAN ID to use for transmission (STD = 11-bit , EXT = 29-bit) , ID val in hex.
  RX         : CAN ID to use for reception    (STD = 11-bit , EXT = 29-bit) , ID val in hex.
  FS         : Force immediate send on code. +CODE=inclusive , -CODE=exclusive
  FW         : Force immediate wake on code. +CODE=inclusive , -CODE=exclusive
  TS         : Force immediate send on timeout  (0 to 1000 ms).
  TW         : Max time before wake is required (0 to 1000 ms).
  WW         : Time to wait after wake message is send before continuing (0 to 1000 ms).
  WMSG       : ASCII formatted CAN message to use for wake.
Note 1 ; The reponse always shows all VC options, but only the specified ones are changed.
Note 2 : If TX and RX CAN identifiers are the same, only half-duplex operation is valid.
```

2.8.11 Set CAN Virtual Circuit Wait after Wakeup (23)



```
Command : set can vc ww=timeout
Desc : Time to wait before sending data after wake message sent.
timeout = 0 to 60000 , ms delay.
Ex 1 : set can vc ww=100
Resp : <A:TX=EXT 1FFFFFFF RX=EXT 1FFFFFFE TS=10 TW=100 FS=OFF FW=OFF WW=100 WMSG=:S0R0↑>
                                                                  +-< Wake msg
                                                        +---------< Wake wait
                                                  +-----------------< Force wake code
                                            +-------------------------< Force send code
                                      +-------------------------------< Timeout wake ms
                                +-------------------------------------< TImeout send ms
                          +-----------------------------------------------< RX Msg
                    +-------------------------------------------------------< TX Msg
Where :
  TX         : CAN ID to use for transmission (STD = 11-bit , EXT = 29-bit) , ID val in hex.
  RX         : CAN ID to use for reception    (STD = 11-bit , EXT = 29-bit) , ID val in hex.
  FS         : Force immediate send on code. +CODE=inclusive , -CODE=exclusive
  FW         : Force immediate wake on code. +CODE=inclusive , -CODE=exclusive
  TS         : Force immediate send on timeout  (0 to 1000 ms).
  TW         : Max time before wake is required (0 to 1000 ms).
  WW         : Time to wait after wake message is send before continuing (0 to 1000 ms).
  WMSG       : ASCII formatted CAN message to use for wake.
Note 1 ; The reponse always shows all VC options, but only the specified ones are changed.
Note 2 : If TX and RX CAN identifiers are the same, only half-duplex operation is valid.
```

2.8.12 Set CAN Virtual Circuit Wakeup Message (24)



When in VC mode and enabling force-wake and/or force-tx codes, these codes will be overidden by the CONFIG sequence check. For example, if either of the force-wake or force-tx codes is enabled and the CONFIG command is enabled, should the config command sequence match one of the force codes, it will be treated as a config command character.

To avoid this, either change the force codes or change the config command sequence.

2.9 CAN Timeout Commands

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. 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 prevent this, a timeout parameter can be specified that will ensure additional input parsing can occur at the expense of some CAN messages. This will allow a CONFIG command sequence to be recognized even when the CAN bus is overloaded or faulty.

The command to set/get the timeout follows:

get can timeout	: Gets the timeout setting.
set can timeout can_tx_busy=off	: Disables timeout, will wait indefinitely (default)
set can timeout can_tx_busy=1000	: Sets the timeout (10 to 10000 ms)

2.9.1 Get can timeout (25)

```
Command : get can timeout
Desc : Gets the current CAN timeout settings.
Ex 1 : get can timeout
Resp : <A:can_tx_busy=off>
Note: 1) Timeout can range from 10 to 10000 ms.  Use 'off' to disable.
```

2.9.2 Set can timeout can_tx_busy=timeout (26)

```
Command : set can timeout can_tx_busy=timeout
Desc : Sets the current CAN timeout parameter 'can_tx_busy'.
Ex 1 : set can timeout can_tx_busy=off
Resp : <A:can_tx_busy=off>
Ex 2 : set can timeout can_tx_busy=1000
Resp : <A:can_tx_busy=1000>
Note: 1) Timeout can range from 10 to 10000 ms.  Use 'off' to disable.
```

2.10 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.
- Definitions can be either a single ID or an ID range.
- Standard (11-bit) and Extended (29-bit) IDs can be defined.
- Definitions must not conflict nor overlap.
- ID values are in hexadecimal and do not need a '$' or '0x' prefix.
- The list can be enabled or disabled.
- 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 empty filter list that is enabled will receive nothing, but still permit transmission (making the unit effectively TX only).
- Multiple filter definitions can be specified in a single command line.
- The maximum number of filter definitions possible is ten (10) per profile.

EX-1: set can filter std 100 ext 200 std 1 10 ext 300 3ff

This will define four filters as follows:

- 11-bit : { 0x100 , 0x001 to 0x010 }
- 29-bit : (0x200 , 0x300 to 0x3FF }

All other IDs received are ignored.

2.11 Using Message Filters

In command mode, the CAN USB-232 supports message filtering through the use of identifier 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 data bandwidth on a busy network by only allowing certain messages to be received.

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.

2.11.1 How Filtering Works

Whenever the CAN USB-232 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 ASCII message string of the message to the serial port. If any filters are enabled, the CAN USB-232 will attempt to match the received message to each enabled filter. At the first successful match, the message is converted to an ASCII message string 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

2.11.2 Setting Up Message Filters

To use message filtering, each filter used must be enabled and then configured as to the type of filtering that is desired. All of these parameters are configured during the configuration process.

A filter is configured in the following order:

1) Enable the filter

2) Specify if standard or extended and if single ID or ID range.

Once all configuration parameters have been saved and the CAN USB-232 enters the normal operating mode, the new filter settings will become active and will be applied to each received CAN message.

2.11.3 Get CAN Filter (27)

- The list can be viewed with the ‘get can filter’ command.
- The list is sorted in ascending order.
- Each entry is assigned an index number starting from zero.
- The index number is used to identify an entry for deletion.

```
Command : get can filter

Desc : Returns a list of defined CAN filters.

A numbered list of filter definitions is returned.
If the command was angled, the list is returned on one line and delimited by angle brackets.
Otherwise each filter definition is shown on a separate line to make it easier to read.

Ex 1 : get can filter

Resp : <A:NONE>

Ex 2 : get can filter

Resp :
<A:
#0=STD 000
#1=STD 001 0FF
#2=EXT 00000000
#3=EXT 00000001 000000FF
>

Ex 3 : <get can filter>

Resp : <A:#0=STD 000 #1=STD 001 0FF #2=EXT 00000000 #3=EXT 00000001 000000FF>

Note 1 : All identifier values are in hexadecimal.

Note 2 : The decimal number between the '#' and '=' is the index into the filter list.
```

2.11.4 Set CAN Filter (28)

Format: set can filter std id xx

```
                    |    |  |____ filter bits
                    |    |
                    |    |___ CAN ID or (id-min id-max)
                    |
                    |____ std or ext
```

Note: Once you set the CAN filter, make sure you set the command mode Filter to ON.

```
Command : set can filter

Desc : Defines a new CAN filter definition.

Will create a new filter and add it to the filter table.  If the new filter conflicts
with any existing fitlers, an error response is returned and the filter is discarded.

Ex 1 : set can filter std 0 std 1 ff ext 0 ext 1 ff

Resp :
<A:
#0=STD 000
#1=STD 001 0FF
#2=EXT 00000000
#3=EXT 00000001 000000FF
>

Note 1 : All identifier values are in hexadecimal.

Note 2 : The decimal number between the '#' and '=' is the index into the filter list.

Note 3 : Multiple filter definitions can be defined in a single command (see above example).
```

2.11.5 Delete CAN Filter (29)

- Either a single entry or and entry range can be specified.
- Entries are identified by their index number.
- Indexes remaining are re-sorted so there are no gaps.

EX-1: del can filter 1

```
Command : del can filter #

Desc : Deletes a CAN filter definition from the filter table.

The decimal index given identifies the filter to be deleted.
If the index is out of range or the table is empty, an error is returned.

The filter identified by the index is deleted and all filters after that are
renumbered using the given index as the starting point.

Ex 1 : del can filter 1

Before :
<A:
#0=STD 000
#1=STD 001 0FF
#2=EXT 00000000
#3=EXT 00000001 000000FF
>

After :
<A:
#0=STD 000
#1=EXT 00000000
#2=EXT 00000001 000000FF
>

Note 1 : All identifier values are in hexadecimal.

Note 2 : The decimal number between the '#' and '=' is the index into the filter list.
```

2.11.6 Delete CAN Filter Range (30)

```
Command : del can filter # #

Desc : Deletes a CAN filter definition range from the filter table.

The decimal indices given identify the range of filters to be deleted.
The first index is the starting point and the second index is the end point.

The filters defined in the range are delted and all filters after that are
renumbered using the first index as the starting point.

If either index is out of range of the table, then an error is returned.

Ex 1 : del can filter 1 2

Before :
<A:
#0=STD 000
#1=STD 001 0FF
#2=EXT 00000000
#3=EXT 00000001 000000FF
>

After :
<A:
#0=STD 000
#1=EXT 00000001 000000FF
>

Note 1 : All identifier values are in hexadecimal.

Note 2 : The decimal number between the '#' and '=' is the index into the filter list.
```

2.11.7 Delete CAN Filter All (31)

```
Command : del can filter all

Desc : Deletes all CAN filter definitions.

Ex 1 : del can filter all

Resp : <A:All filters deleted>
```

2.12 Config Entry Commands

It is now possible to enter configuration mode remotely through the COM port in either CM or VC mode.

In CM mode : 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'

In VC mode : 1-second idle time, followed by three '!' characters, followed by 1-second idle time.

The exact sequence can be specified by the user.

The 'set/get cfgcmd' configuration command will allow you to enable/disable the CONFIG command and set additional parameters described below.

CM Mode:

get cfgcmd cm : Returns the current state of the CM mode config command

set cfgcmd cm enabled=yes | no : Enables or disables the command in CM mode (default is enabled)

VC Mode:

get cfgcmd vc : Returns the current state of the VC mode config command

set cfgcmd vc enabled=yes | now : Enables or disables the command in VC mode (default is disabled)

set cfgcmd vc tbeg=timeout : Minimum IDLE time to wait before matching "!!!" sequence : 1000 to 10000 ms (default 1000)

set cfgcmd vc tend=timeout : Minimum IDLE time to wait after matching "!!!" sequence : 1000 to 10000 ms (default 1000)

set cfgcmd vc tid=timeout : Maximum IDLE time between sequence characters : 10 to 1000 ms (default 1000)

set cfgcmd vc seq="text" : Defines the command sequence text to match : 1 to 10 characters long (default "!!!")

Note : For the text specified in the 'seq' parameter, arbitrary ASCII values can be entered with the backslash escape sequence followed by three decimal digits from 0 to 255.

Common escape sequences are defined :

\r : CR

\n : LF

\t : TAB

\" : " (double quote)

\\ : BACKSLASH

Ex 1 : "\000" : Binary 0x00

Ex 2 : "\r\n" " CR followed by LF

2.12.1 Get cfgcmd cm (32)

```
Command : get cfgcmd cm
Desc : Gets the current CM-Mode config command settings.
Ex 1 : get cfgcmd cm
Resp : <A:enabled=yes>
```

2.12.2 Set cfgcmd cm enable=YES | NO (33)

```
Command : set cfgcmd cm enabled=YES ! NO
Desc : Enables or disables CM-Mode config command.
Ex 1 : set cfgcmd cm enabled=no
Resp : <A:enabled=no>
```

2.12.3 Get cfgcmd vc (34)

```
Command : get cfgcmd vc
Desc : Gets the current VC-Mode config command settings.
Ex 1 : get cfgcmd vc
Resp : <A:enabled=no tbeg=1000 tend=1000 tid=1000 seq="!!!">
```

2.12.4 set cfgcmd vc enable=YES | NO (35)

```
Command : set cfgcmd vc enabled=YES | NO
Desc : Enables or disables VC-Mode config command.
Ex 1 : set cfgcmd vc enabled=no
Resp : <A:enabled=no tbeg=1000 tend=1000 tid=1000 seq="!!!">
Note: 1) Response always includes all parameters.
```

2.12.5 set cfgcmd vc tbeg=timeout (36)

```
Command : set cfgcmd vc tbeg=timeout
Desc : Sets min idle time (ms) before VC-Mode config command can be recognized.
Ex 1 : set cfgcmd vc tbeg=1000
Resp : <A:enabled=no tbeg=1000 tend=1000 tid=1000 seq="!!!">
Note: 1) Response always includes all parameters.
Note: 2) Timeout is from 1000 to 10000 ms.
```

2.12.6 set cfgcmd vc tend=timeout (37)

```
Command : set cfgcmd vc tend=timeout
Desc : Sets min idle time (ms) after VC-Mode config command can be recognized.
Ex 1 : set cfgcmd vc tend=1000
Resp : <A:enabled=no tbeg=1000 tend=1000 tid=1000 seq="!!!">
Note: 1) Response always includes all parameters.
Note: 2) Timeout is from 1000 to 10000 ms.
```

2.12.7 set cfgcmd vc tid=timeout (38)

```
Command : set cfgcmd vc tid=timeout
Desc : Sets max time (ms) allowed between VC-Mode config command sequence bytes.
Ex 1 : set cfgcmd vc tid=1000
Resp : <A:enabled=no tbeg=1000 tend=1000 tid=1000 seq="!!!">
Note: 1) Response always includes all parameters.
Note: 2) Timeout is from 10 to 1000 ms.
```

2.12.8 set cfgcmd vc seq=”sequence” (39)

```
Command : set cfgcmd vc seq="text"
Desc : Sets the VC-Mode config command sequence bytes.
Ex 1 : set cfgcmd vc seq="!!!"
Resp : <A:enabled=no tbeg=1000 tend=1000 tid=1000 seq="!!!">
Note: 1) Response always includes all parameters.
Note: 2) Use "\nnn" to enter 3-digit decimal ASCII values (000 to 255).
Note: 3) For '<' , '>' , and '"' : Use ASCII values.
Note: 4) Use \t for TAB, \r for CR , \n for LF , \" for QUOTE , \\ for BACK-SLASH
Note: 5) Command sequence length constraints : 1 <= length <= 10
```

Whenever the CONFIG button is pressed, the COM port settings are automatically set to 115200 N 8 1 F- (for RS232 units). However, if CONFIG mode is entered via a command sequence through the COM port, then the COM port settings used are the configured ones.

This ensures that an application can smoothly change from RUN to CONFIG and back to RUN mode without needing to change COM port settings.

Note : For USB devices, the COM port settings have no effect as the actual baud rate is the full-USB speed at all times.

2.13 Profiles

Users can now assign short text notes to each profile to aid in remembering the settings.

The command to set/get profile notes follows:

profile note : Returns the note for the currently active profile

profile note # : Returns the nore for the specified profile number (0 to 9)

profile note all : Returns a list of all profile notes.

Note : When 'all' is specified, the return format depends on whether the command was entered in EDIT or ANGLED mode. If in edit mode, then it is returned as a list with each profile note on a separate line. In angled mode, it is returned as a compance, single line between angle brackets.

profile note="Text" : Sets the profile note for the current profile to "Text" (default "")

profile note #="Text" : Sets the specified profile note to "Text" (0 to 9)

2.13.1 Set Active Profile Number (40)

```
Command : profile #
Desc : Sets the active profile.
There are ten profiles numbered from 0 to 9.  This command determines which profile
will be active in CM or VC operating mode.
Ex 1 : profile 1
Resp : <A:Active profile 1>
```

2.13.2 Copy Profile (41)

```
Command : profile copy # #
Desc : Copies source profile to dest profile.
The first number is the source profile and the second number is the destination profile.
Ex 1 : profile copy 1 2
Resp : <A:Profile 1 copied to 2>
```

2.13.3 Set Active Profile to Default (47)

```
Command : default
Desc : Sets the active profile to default settings.
Ex 1 : default
Resp : <A:Current profile (1) set to defaults>
```

2.13.4 Set Specific Profile to Default (48)

```
Command : default #
Desc : Sets the spcified profile to default settings.
Ex 1 : default 2
Resp : <A:Profile (2) set to defaults>
```

2.13.5 Set All Profiles to Default (49)

```
Command : default all
Desc : Sets all profiles to default settings.
Ex 1 : default all
Resp : <A:All profiles set to defaults>
```

2.13.6 CAN Port SERNUM Interaction

The SERNUM command provides a means for the user to query the CAN USB-232 for its unique 128-bit serial number. The GET SERNUM command only works while in CONFIG mode and always returns the serial number over the COM port. There is never any serial number sent out of the CAN port.

2.13.7 Get Serial Number (50)

```
Command : get sernum
Desc : Returns a 128-bit hex device serial number.
Ex 1 : get sernum
Resp : <A:16 16 02 1E 53 56 12 F0 4E BB 35 4D F5 00 00 00>
```

2.13.8 Get Version Number (51)

```
Command : get version
Desc : Returns hardware, bootloader, and application version information.
Ex 1 : get version
Resp : HW=A BOOT=1.00 APP=1.00>
```

2.14 Test Options

2.14.1 Cycling LED Indicators

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

To terminate the diagnostic, press any key or the config button.

2.14.2 Generating CAN Square Wave

The CAN USB-232 can generate a square wave output onto the CAN bus arbitrary frequencies. It is useful for measuring propagation delay through cabling with the aid of an oscilloscope and can be used to evaluate bus termination.

The square wave frequency can be in the range of 10,000 to 1,000,000 Hz.

2.14.3 Test LEDs (52)

```
Command : test
Desc : Cycles CAN and COM TX & RX LED indicators.
Note 1 : This command can not be executed in angle mode and does not return a status.
```

2.14.4 Test LEDs and CAN Bus (53)

```
Command : test bps
Desc : Cycles CAN and COM TX & RX LED indicators and generates 'bps' HZ square wave on CAN bus.
Frequency range of bps: 10,000 to 1,000,000 HZ
Note 1 : This command can not be executed in angle mode and does not return a status.
```

2.14.5 Exit (54)

```
Command : exit
Desc : Exit configuration mode.
Will exit configuration mode and begin executing according to the configuration
settings present in the active profile.
Ex 1 : exit
Resp : <A>
```

2.15 XMODEM Download

You can do a download from XMODEM using the following steps:

1) Enter config mode

2) Type ‘get sernum’ to get the device 128-bit serial number

3) Type ‘firmware download ## ## ## ## ## ## ## ## ## ## ## ## ## ## ## ## ##’ where ‘##’ … is the serial number from step 2. This must be done with care, otherwise it will not work.

At this point, there should be a prompt telling you to start the XMODEM transfer.

Do the following:

1) File menu <Alt-F>

2) Transfer <T>

3) Xmodem <X>

4) Send <S>

5) Enter the path to the IMAGE file and press enter.

You should see a progress bar indicating how the download is proceeding.

When it is done, the device will start up in CM mode (as per defaults).

To confirm operation, press the CONFIG button normally (ie: less than three seconds) and you will be in CONFIG mode.

2.16 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.

Grid Connect technical support: (630) 245-1445.

Our phone lines are open from 8:00AM - 4:30 PM Central Time Monday through Friday excluding holidays.