CAN232

RS232 to Controller Area Network Adapter
Application Documentation

Version 1.3
2/9/00

Table Of Contents

Introduction

The CAN232 from Embedded Micro Software is a very economical way to connect a computer with a serial port to a Controller Area Network (CAN). The CAN232 is designed to be easy to use and easy to design applications to interface to a CAN network. The CAN232 connects to the computer's serial port.

Typical Applications

- Send and receive CAN messages
- Develop powerful CAN applications using PC software
- Compact design is ideal for portable applications
- Simulate network traffic to test module applications

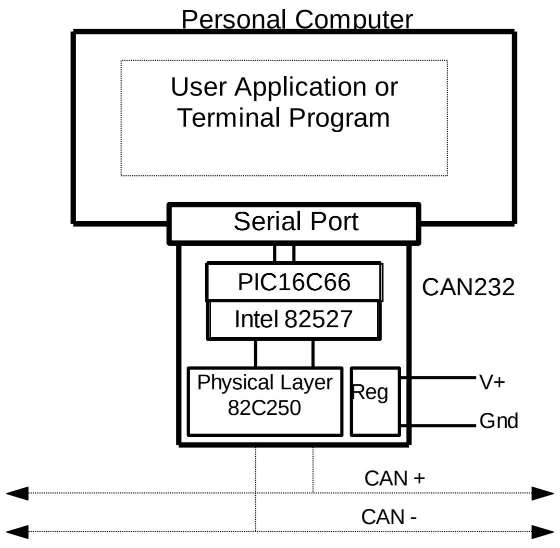
Design Intentions

The CAN232 is intended to be a low-cost alternative to the PCMCIA, PC Bus Cards, and serial interface cards used to connect a PC to the CAN network. In order to keep the cost low while maintaining a high degree of functionality, the CAN232 relies on a PIC16C66 processor from MicroChip.

The software included is general in nature and is intended to aid in getting your application up and running quickly.

CAN232 Overview

The following block diagram shows the hardware and software model.



Hardware Overview

The CAN232 uses an Intel 82527 CAN controller to connect to the network. The 82527 provides;

- Support for CAN Specification 2.0
  - Standard 11 bit message identifier data and remote frames
  - Extended 29 bit message identifier data and remote frames
- Global Identifier Masks (Acceptance filters)
- 14 Message Objects for transmit or receive
- 1 Receive Message Object with programmable mask
- Programmable Bit Rate

The MicroChip PIC16C66 provides the control between the serial port and the Intel 82527.

The physical layer is a Phillips 82C250.

Software Overview

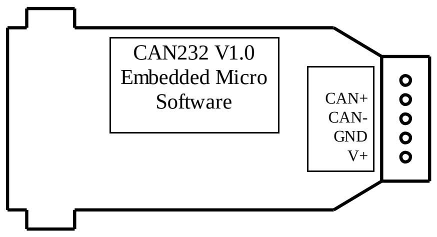
The CAN232 provides a user interface consisting of simple ASCII commands. Using terminal programs such as the TERMINAL.EXE program under Windows, a user can access the CAN network with simple text commands. Binary commands are also included for optimized interfacing to computer program control such as applications written using Visual Basic.

Installation

The CAN232 interfaces to the serial port using a DB25 connector.

The following external connections are required;

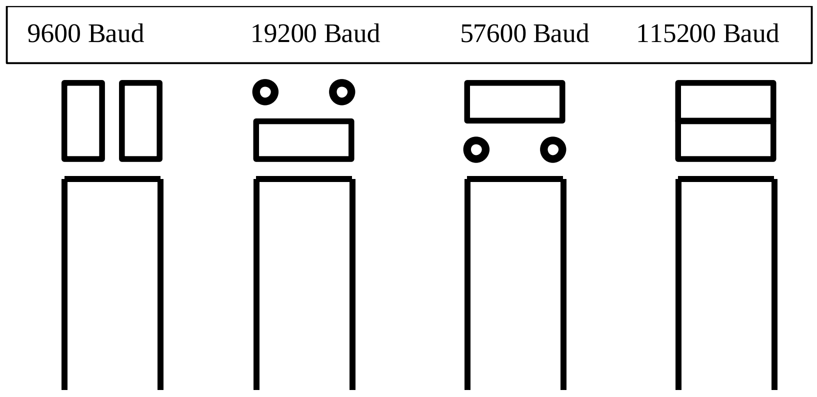
1. Connect the CAN network to CAN+ and CAN- on the green terminal block.
2. Connect a DC voltage (6 to 24 volts) to V+ and GND on the green terminal block.
3. Verify polarity is correct (CAN232 is protected for reverse polarity and overvoltage).
4. Connect the CAN232 to the serial port.
5. Turn on power.
6. Test the installation by running a terminal program at 9600 baud, no parity, 8 bits, and 1 stop bit. Enter the character V and the CAN232 software version is displayed.
7. If CAN232 is not found, verify all connections. Make sure power is connected correctly and the correct communication port is selected.



NOTE: The CAN232 does not contain a termination resistor between CAN+ and CAN-. One can be added across the connector terminals if needed.

RS232 Baud Rate Setting

The CAN232 can be configured to operate at 4 different baud rates. The 2 jumpers are located next to pin 1 on the PIC microcontroller. The default setting (no jumpers installed) is 9600 baud, 8 data bit, no parity, and 1 stop bit.



CAN232 Commands - ASCII

The CAN232 processes commands to configure the network, send messages, and monitor messages. Each are explained in detail in the following sections.

OFFLINE

The offline command, F, disables communications. This command does not require the carriage return to terminate the command.

ONLINE

The online command, O, enables communications. This command does not require the carriage return to terminate the command.

RESET

The reset command, X, resets the 82527 CAN chip on the CAN232.

BAUD RATE

The baud rate command is used to view or set the baud rate.

- B[cr] displays current baud rate
- B0[cr] = 1 mbps
- B1[cr] = 889 kbps
- B2[cr] = 800 kbps
- B3[cr] = 727 kbps
- B4[cr] = 666 kbps
- B5[cr] = 500 kbps (default value)
- B6[cr] = 250 kbps
- B7[cr] = 167 kbps
- B8[cr] = 125 kbps
- B9[cr] = 100 kbps
- B10[cr] = 83.3 kbps

- B11[cr] = 62.5 kbps
- B12[cr] = 50 kbps
- B13[cr] = 41.67 kbps
- B14[cr] = 33 kbps
- B15[cr] = 31.25 kbps
- B16[cr] = 25 kbps
- B17[cr] = 20 kbps
- B18[cr] = 10 kbps

CONFIGURE MESSAGE OBJECT

The configure message object command sets the direction, type, length, and id for a message.

Cobj,dir,type,len,id[cr]

- **obj** specifies one of 10 message objects from 0 to 9
- **dir** specifies the direction, either R for receive or T for transmit
- **type** specifies either S for standard 11 bit identifiers or E for extended 29 bit identifiers
- **len** specifies the message length for 0 to 8 data bytes
- **id** specifies the 11 or 29 bit identifier

For example:

C1,R,S,8,0x100[cr] specifies that message object #1 is a receive message, standard 11 bit ID, 8 data bytes and id=0x100(hex).
C2,T,S,5,50[cr] specifies message #2 is a transmit message, standard 11 bit ID, 5 data bytes, and id is 50 (decimal).

SET MESSAGE DATA

Message data is set using this command.

For example:

S1,0,0,55,0,0xaa,0,0,1[cr] sets all 8 data bytes for message object #1.
S1,0,2[cr] sets only the first 2 data bytes for message object #1. The remaining 6 are left unchanged.
Note: To set individual bytes use the Set Message Data Byte command.

DISPLAY MESSAGE DATA

Message data is displayed using this command.

For example:

D1[cr] displays all 8 data bytes for message object #1

SEND/RECEIVE MESSAGE

The ASCII characters 0 through 9 can be used to send or receive a message. If the message has been configured to transmit then the message will be sent. If the message has been configured to receive then CAN232 will display if the message is new and will also display the data bytes.

PUT INDIVIDUAL MESSSAGE DATA BYTE

The P command allows writing an individual data byte in a message.

Pobj,b,d[cr] where **obj** is message object 1 to 10 (0 representing 10), **b** is the data byte location in the message from 0 to 7, and **d** is value to write to the data byte.

For example:

P1,0,10[cr] writes decimal 10 to the first data byte of message 1
P9,7,0xa0[cr] writes hexadecimal a0 to the last data byte of message 9

GET INDIVIDUAL MESSAGE DATA BYTE

The G command allows reading an individual data byte in a message.

Gobj,b[cr] where **obj** is message object 1 to 10 (0 representing 10), and **b** is the data byte location in the message from 0 to 7. The value is returned as a decimal value terminated with a carriage return.

For example:

G1,0[cr] reads the first data byte of message 1
G9,7[cr] reads the last data byte of message 9

MESSAGE SEND TIME

Message objects can be configured to be transmitted periodically.

Qobj,b[cr] where **obj** is message object 1 to 10 (0 representing 10), and **b** is the number of milliseconds from 1 to 255. A value of 0 disables the feature.
Q[cr] returns a list of all the values.

For example, set message 2 to send every 50 milliseconds then display all the message send times.

```
Q2,50[cr]
Q[cr]

Message send times
1: 0
2: 50
3: 0
4: 0
5: 0
6: 0
7: 0
8: 0
9: 0
10: 0
```

SET BIT TIME REGISTERS

The SET BIT TIME REGISTERS command allows a user to setup the registers that control the baud rate to values other than the standard values set using the BAUD RATE command. Refer to the 82527 Bit Timing Definitions data sheet for more information. This command is for advanced users and will not be needed in most cases.

Tsjw,brp,spl,tseg1,tseg2 where **sjw** is the (Re)Synchronization Jump Width, **brp** is the Baud Rate Prescaler, **spl** is the Sampling Mode, **tseg1** is Time Segment 1, and **tseg2** is Time Segment 2.

READ REGISTER

The READ REGISTER command allows a user to read the value of one of the 82527 registers. Refer to the 82527 data sheet for further information.

WRITE REGISTER

The WRITE REGISTER command allows a user to write a value to one of the 82527 registers. Refer to the 82527 data sheet for further information.

Wb,d[cr] writes the 1 byte value (range 0 to 255 or 0 to 0xff)
in **d** to register **b** (0 to 254 or 0 to 0xfe). Do not write to register 255 (0xff).

VERSION

This command displays the version information, terminated with a carriage.

CAN232 Commands - Binary

The CAN232 allows commands to be sent to it as binary values to facilitate interfacing with computer programs.

Read Register - E0

This command reads one of the 82527 registers.

Format of command from host to CAN232;

1.	Command(hex)	E0
2.	Register	xx

Data returned from CAN232;

1. Value of register xx (0 to 255)

Write Register - E1

This command writes one of the 82527 registers.

Format of command from host to CAN232;

1.	Command(hex)	E1
2.	Register	xx
3.	Value	yy

Data returned from CAN232;

1. Value of register xx (0 to 255)

Read Message Data Byte - E2

This command reads one of the message objects data bytes.

Format of command from host to CAN232;

1.	Command(hex)		E2
2.	Message # (1 to 10)	mn	
3.	Data Byte (0 to 7)	db	

Data returned from CAN232;

1. Value of data byte (0 to 255)

Write Message Data Byte - E3

This command writes one of the message objects data bytes.

Format of command from host to CAN232;

1.	Command(hex)		E3
2.	Message # (1 to 10)	mn	
3.	Data Byte (0 to 7)	db	

Data returned from CAN232;

1. Value of data byte written (0 to 255)

Set Baud Rate - E7

This command sets the baud rate. See above table in ASCII command for setting baud rate for values.

Format of command from host to CAN232;

1.	Command(hex)		E7
2.	Baud Index (0 to 18)	br	

Data returned from CAN232;

1. Baud Index selected

Get Baud Rate - E8

This command gets the baud rate index. See above table in ASCII command for baud rate for values.

Format of command from host to CAN232;

1.	Command(hex)		E8

Data returned from CAN232;

1. Baud Index selected

Configure Message Object - E9

This command configures the message object.

Format of command from host to CAN232;

1.	Command(hex)		E9
2.	Message # (1 - 10)	mn	
3.	Direction	di	1=transmit, 0=receive
4.	Identifier Type	it	0=standard 11bit, 1=extended 29 bit
5.	Message Length	ml	0 to 8
6.	Message ID	id0	most significant byte
7.	Message ID	id1	upper middle significant byte
8.	Message ID	id2	lower middle significant byte
9.	Message ID	id3	least significant byte

Data returned from CAN232;

1. Command E9 is returned

Read Message Object - EA

This command reads the message configuration.

Format of command from host to CAN232;

1.	Command(hex)		EA

Data returned from CAN232;

1.	Direction	di	1=transmit, 0=receive
2.	Identifier Type	it	0=standard 11bit, 1=extended 29 bit
3.	Message Length	ml	0 to 8
4.	Message ID	id0	most significant byte
5.	Message ID	id1	upper middle significant byte
6.	Message ID	id2	lower middle significant byte
7.	Message ID	id3	least significant byte

Set Message Send Time - EB

This command sets the message send time for a message object.

Format of command from host to CAN232;

1.	Command(hex)		EB
2.	Message # (1 to 10)	mn	
3.	Most significant byte of send time	mt	
4.	Least significant byte of send time	lt	

Data returned from CAN232;

1 if ok
0 if error

Read Message Send Time - EC

This command reads the message send time for a message object.

Format of command from host to CAN232;

1.	Command(hex)	EC
2.	Message # (1 to 10)	mn

Data returned from CAN232;

1. Most significant byte of send time
2. Least significant byte of send time

Send/Receive Message Send Time - D1 to DA

These commands will send a message or return the data received.

Format of command from host to CAN232;

1.	Command(hex)	D1 to DA

Data returned from CAN232;

1. Least significant byte of send time