CAN232

Manual

PRELIMINARY INFORMATION

Preliminary
CAN232 Manual

November 2001
Version 0.9C

LAWICEL HB
Klubbgatan 3
SE-282 32 Tyringe
SWEDEN

Embedded Tools, Starterkits & Consulting
AVR • 8051 • PICmicro • HC08 • HC11 • HC12 • SX • C166
CAN (Controller Area Network) • CANopen • USB • Compilers
Single Board Computers • Programmers • Software

Phone: +46 (0)451 598 77
Fax: +46 (0)451 598 78
info@lawicel.com
www.lawicel.com

In this manual are descriptions for copyrighted products that are not explicity indicated as such. The absence of the copyright © symbol does not infer that a product is not protected. Additionally, registered patents and trademarks are similarly not expessly indicated in this manual.

The information in this document has been very carefully checked and is believed to be reliable. However, **LAWICEL** assumes no responsibility for any inaccuracies. **LAWICEL** gives no guarantee nor accepts any liability what so ever for consequential damages resulting from the use of this manual or it's associated product. **LAWICEL** reserves the right to change the information contained herin without prior notification.

Further, **LAWICEL** offers no guarantee nor accepts any liability for damages for improper usage or improper installation of the hardware described herein. Finally **LAWICEL** reserves the right to change the hardware or design without prior notification and accepts no liability for doing so.

1st edition June 2001 (Preliminary).

2nd edition September 2001 (Preliminary).

3rd edition November 2001 (Preliminary).

LAWICEL HB
Klubbgatan 3
S-282 32 Tyringe
SWEDEN
Phone: +46 451 59877
FAX: +46 451 59878
http://www.lawicel.com
info@lawicel.com

1.0 Introduction

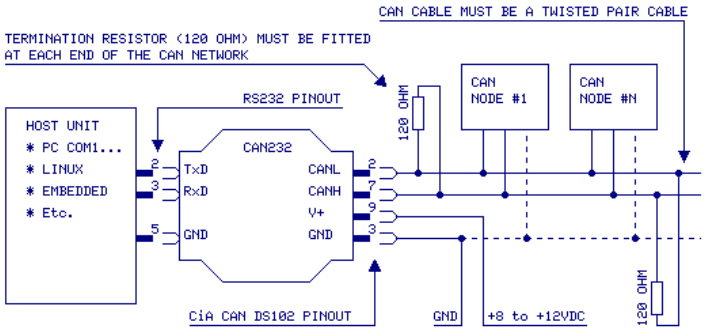
The **LAWICEL CAN232** is a low cost and easy to use dongle, that could be used together with any O/S without drivers since it is an RS232 to CAN gateway. Simply connect it to any PC running DOS, Windows95/98/ME, NT4/2000 or Linux and ”talk” with the unit in standard ASCII format. It could also be used together with Embedded computers that needs a simple CAN connectivity without changing the excisting hardware. The CAN232 handles both the 11bit ID format as well as the 29bit ID format, built in FIFO queues, extended info/ error information and simple power up through a few commands. The CAN232 is only 68mm long, 31mm wide and 16mm thick using the latest technology of small SMD parts on both sides of the board, the power behind is an Atmel AVR and the Philips SJA1000 CAN controller and that makes it very flexible in the way of handling small bursts of CAN frames at a high bus speed. The CAN232 can be customized with your needs as a CAN to RS232 unit (i.e. convert excisting RS232 products to CAN that are too expensive to replace or to extend an RS232 network longer than the normal length is for RS232 etc.).

Bottom side with Atmel 8515 AVR.

Top side (in enclosure) with Philips SJA1000.

1.1 Installation

The RS232 side of the dongle (DB9 female) is inserted directly into a PC’s COM serial port or via a cable to the Host system (such as an embedded system etc.). The CAN side of the dongle (DB9 male) has the same pinout as the standard CAN in Automation (CiA) DS102 profile and the CAN232 dongle must be powered via the CAN side with 8 to 12VDC. The dongle is ESD protected so reversing the power will not damage the CAN232, instead the power supply will be short circuited to protect the CAN232 dongle. The CAN232 dongle needs about 40-100mA depending on how much the CAN network is loaded (i.e. numbers of nodes etc.). Below is a simple schematic showing how to connect the CAN232.



1.2 Testing the CAN232

Test the CAN232 by installing it to a PC's COM port and power it up according to instructions on previous page under section 1.1. When the CAN232 gets power the 2 LEDs (red & green) will blink rapidly some times depening on what RS232 speed it is set up to (this works only on version V1013 and later, V1012 had only one long blink and prior to that no blink at all). If the RS232 is set to 57,600baud (default when delivered) both LEDs will blink 3 times (the higher RS232 speed the less it blinks, see the **U** command for more info). Then start Windows Terminal software and set it up to e.g. 57600baud, 8 databits, no parity, 1 stop bit (if the CAN232 is set to 57,600baud), also set local echo on so you can see what you type and set the check flag so that it appends a line feed when it receive and end of line. Then make sure you are connected and press >ENTER< and it will make a new line, then press V and >ENTER< and it will print/reply Vhhss, where hh is the hardware version and ss is the software version. Now you know you have full contact with the CAN232 unit and can set it up with a CAN speed and open the CAN port, send and receive frames. Note that the green LED indicates that a CAN frame is succesfully sent or received into the CAN232 unit. Note that you must at least have 2 nodes to send/receive CAN frames and that the CAN cable network is terminated at both ends with 120 ohms over the CANL and CANH lines plus that a twisted pair CAN cable is used. The CAN232 is set to accept all frames, so no need to set filters etc. for testing.

1.3 CAN232 limitations

There are of course limitations of how many frames the CAN232 can send & receive. Current version (V1013) is tested with a throughput of sending 300 extended frames at 125kbit CAN and 57,600 baud of the RS232. The bottle neck is of course the RS232 side and the microcontroller not being able to handle more frames per second. Even if the RS232 is increased to the double (115,200 baud) it isn't possible to send 600 frames, it may increase to 350-400 frames. So the CAN232 is aimed for low speed CAN networks and works very well with CAN speeds at 125kbit or less but of course it is usable up to 1 Mbit (but the bus load may not be high at these speeds or e.g. the filter has to be set to accept some of the frames). The CAN232 has software CAN FIFO queues for both sending and reception. These FIFO's can handle each 8 frames (standard or extended). Furthermore the CAN232 has only a small RS232 buffer, so it can only handle one command at a time, meaning before sending the next command to it, you must wait for an answer

rom the CAN232 unit (OK which is [CR] or Error which is [BELL]).

1.4 Driver Design Guide

The CAN232 doesn’t come with a driver. Since many commercial development tools provide an RS232 driver (such as Visual Basic, Deplhi etc.) it is simple to write a simple program to ”talk” to the CAN232 unit. The best way is to make a thread that handles all the communication to the CAN232 and puts all messages in FIFO queues or mail boxes depending on your application. Start with sending 2-3 [CR] to empty any prior comand, then check the CAN version with **V** command (to be sure that you have communication with the unit at correct speed) then set up the CAN speed with **s** or **S** command, then open the CAN port with **O**, then the CAN232 is in operation for both sending and receiving CAN frames. Send frames with the **t** ot **T** command and wait for a response back to see it it was placed in the CAN FIFO transmission queue or the queue was full. When there is no frame to send to the CAN232, just send the **P** or **A** command to poll frames (if there are any pending frames to be polled). Then once in a while send the **F** command to see if there are any errors (e.g. after 10-20 **P** commands or if you get an error back from the CAN232). If you get to many errors back after sending commands to the unit, send 2-3 [CR] to empty the buffer, then issue the commands again, if this continue allert the user within your program that there is a communication error (e.g. a damaged RS232 tranceiver or power failure).

2.0 Availible CAN232 ASCII Commands:

Note: All commands to the CAN232 must end with [CR] (Ascii=13) and they are CASE sensetive.

Sn[CR]

Setup with standard CAN bit-rates where n is 0-8.
This command is only active if the CAN channel is closed.

S0	Setup 10Kbit
S1	Setup 20Kbit
S2	Setup 50Kbit
S3	Setup 100Kbit
S4	Setup 125Kbit
S5	Setup 250Kbit
S6	Setup 500Kbit
S7	Setup 800Kbit
S8	Setup 1Mbit

Example: `S4[CR]`
Setup CAN to 125Kbit.

Returns: CR (Ascii 13) for OK or BELL (Ascii 7) for ERROR.

sxxyy[CR]

Setup with BTR0/BTR1 CAN bit-rates where xx and yy is a hex value. This command is only active if the CAN channel is closed.

xx	BTR0 value in hex
yy	BTR1 value in hex

Example: `s031C[CR]`
Setup CAN with BTR0=0x03 & BTR1=0x1C which equals to 125Kbit.

Returns: CR (Ascii 13) for OK or BELL (Ascii 7) for ERROR.

O[CR]

Open the CAN channel.
This command is only active if the CAN channel is closed and has been set up prior with either the S or s command (i.e. initiated).

Example: `O[CR]`
Open the channel

Returns: CR (Ascii 13) for OK or BELL (Ascii 7) for ERROR.

C[CR]

Close the CAN channel.
This command is only active if the CAN channel is open.

Example: `C[CR]`
Close the channel

Returns: CR (Ascii 13) for OK or BELL (Ascii 7) for ERROR.

tiiildd...[CR]

Transmitt a standard (11bit) CAN frame.
This command is only active if the CAN channel is open.

iii	Identifier in hex (000-7FF)
l	Data length (0-8)
dd	Byte value in hex (00-FF). Numbers of dd pairs muct match the data length, otherwise an error occur.

Example 1: `t10021133[CR]`
Sends an 11bit CAN frame with ID=0x100, 2 bytes with the value 0x11 and 0x33.

Example 2: `t0200[CR]`
Sends an 11bit CAN frame with ID=0x20 & 0 bytes.

Returns: CR (Ascii 13) for OK or BELL (Ascii 7) for ERROR.

Tiiiiiiiildd...[CR]

Transmitt an extended (29bit) CAN frame.
This command is only active if the CAN channel is open.

iiiiiiii	Identifier in hex (00000000-1FFFFFFF)
l	Data length (0-8)
dd	Byte value in hex (00-FF). Numbers of dd pairs muct match the data length, otherwise an error occur.

Example 1: `t0000010021133[CR]`
Sends a 29bit CAN frame with ID=0x100, 2 bytes with the value 0x11 and 0x33.

Example 2: `t000000200[CR]`
Sends a 29bit CAN frame with ID=0x20 & 0 bytes.

Returns: CR (Ascii 13) for OK or BELL (Ascii 7) for ERROR.

P[CR] Poll incomming FIFO for CAN frames (single poll)
This command is only active if the CAN channel is open.

Example 1: `P[CR]`
Poll one CAN frame from the FIFO queue.

Returns: A CAN frame with same formatting as when sending frames and ends with a CR (Ascii 13) for OK. If there are no pendant frames it returns only CR. If CAN channel isn't open it returns BELL (Ascii 7).

A[CR] Polls incomming FIFO for CAN frames (all pending frames)
This command is only active if the CAN channel is open.

Example 1: `A[CR]`
Polls all CAN frame from the FIFO queue.

Returns: CAN frames with same formatting as when sending frames seperated with a CR (Ascii 13). When all frames are polled it ends with an A and a CR (Ascii 13) for OK. If there are no pending frames it returns only an A and CR. If CAN channel isn't open it returns BELL (Ascii 7).

F[CR] Read Status Flags.
This command is only active if the CAN channel is open.

Example 1: `F[CR]`
Read Status Flags.

Returns: An F with 2 bytes BCD hex value plus CR (Ascii 13) for OK. If CAN channel isn't open it returns BELL (Ascii 7). This command also clear the RED Error LED. See availible errors below. E.g. `F01[CR]`

Bit 0	CAN receive FIFO queue full
Bit 1	CAN transmit FIFO queue full
Bit 2	Error warning (EI), see SJA1000 datasheet
Bit 3	Data Overrun (DOI), see SJA1000 datasheet
Bit 4	Not used.
Bit 5	Error Passive (EPI), see SJA1000 datasheet
Bit 6	Arbitration Lost (ALI), see SJA1000 datasheet *
Bit 7	Bus Error (BEI), see SJA1000 datasheet **

* Arbitration lost doesn't generate a blinking RED light!
** Bus Error generates a constant RED light!

Mxxxxxxxx[CR]

Sets Acceptance Code Register.
This command is only active if the CAN channel is initiated and not opened.

xxxxxxxx — Acceptance Code in hex. For more information, see Philips SJA1000 datasheet.

Example: `M00000000[CR]`
Set Acceptance Code to 0x00000000

Returns: CR (Ascii 13) for OK or BELL (Ascii 7) for ERROR.

mxxxxxxxx[CR]

Sets Acceptance Mask Register.
This command is only active if the CAN channel is initiated and not opened.

xxxxxxxx — Acceptance Mask in hex. For more information, see Philips SJA1000 datasheet.

Example: `mFFFFFFFF[CR]`
Set Acceptance Mask to 0xFFFFFFFF

Returns: CR (Ascii 13) for OK or BELL (Ascii 7) for ERROR.

Un[CR]

Setup UART with a new baud rate where n is 0-6.
This command is only active if the CAN channel is closed.
The value is saved in EEPROM and is remembered next time the CAN232 is powered up.

U0	•	Setup 230400 baud (not guaranteed to work)
U1	••	Setup 115200 baud
U2	•••	Setup 57600 baud (default when delivered)
U3	••••	Setup 38400 baud
U4	•••••	Setup 19200 baud
U5	••••••	Setup 9600 baud
U6	•••••••	Setup 2400 baud

Example: `U1[CR]`
Setup UART to 115200 baud.

Returns: CR (Ascii 13) for OK or BELL (Ascii 7) for ERROR.

The dots above indicates how many times the red and green LED blink when it is powered up. This is a simple way of knowing which RS232 speed it is configured with.

V[CR]

Get Version number of both CAN232 hardware and software
This command is only active always.

Example: `V[CR]`
Get Version numbers

Returns: V and a 2 bytes BCD value for hardware version and a 2 byte BCD value for software version plus CR (Ascii 13) for OK. E.g. `V1013[CR]`