LAWICEL
www.can232.com

CAN232 Version 3 Manual

February 2010
CAN232 Manual Version 3.0

LAWICEL AB
Box 3
SE-282 21 Tyringe
SWEDEN

Embedded Tools, Starterkits & Consulting
AVR • 8051 • PICmicro • HC08 • HC11 • HC12 • MSP430 • 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
support@can232.com
www.can232.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 AB** assumes no responsibility for any inaccuracies. **LAWICEL AB** 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 AB** reserves the right to change the information contained herin without prior notification.

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

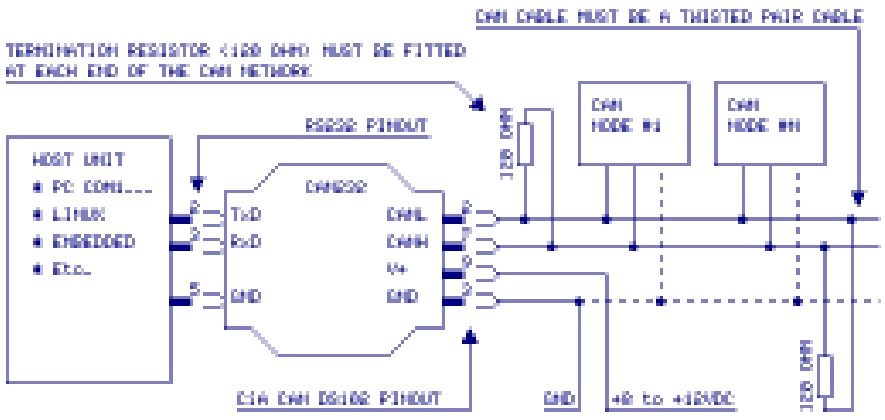
LAWICEL AB
Box 3, Industrigatan 6 2nd Floor
S-282 21 Tyringe
SWEDEN
Phone: +46 451 59877
FAX: +46 451 59878
http://www.can232.com/
support@can232.com

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/XP/Vista/Windows 7, Mac or Linux and ”talk” with the unit in standard ASCII format. It could also be used together with Embedded single board computers that needs a simple CAN connectivity without changing the excisting hardware. The CAN232 handles both the 11bit ID format (standard) as well as the 29bit ID format (extended), 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 ATMEGA162 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.). This document describes the third version of the CAN232 which can be updated via a bootloader (version V1324 or higher of the CAN232). The original versions number 1 and 2 of the CAN232 manual can be found at www.can232.com. Some commands in this manual will not work in the old original version plus the performance (speeds and buffers etc.) described here is only valid for the new version 3.

1.1 Installation

The RS232 side of the dongle (DB9 female) could be 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 6 to 16VDC. 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 4 LEDs (red, yellow & two green) will blink rapidly some times depening on what RS232 speed it is set up to. If the RS232 is set to 57,600baud (default when delivered) all four 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 (or your favourite 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. Finally, make sure you have hardware and software handshaking off and that do LF (ASCII 10) is added on outgoing CR (ASCII 13). 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 (e.g. V1335). 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 on the CAN connector side indicates that a CAN frame is succesfully sent or received into the CAN232 unit and the green LED on the RS232 side blinks for received RS232 data (new in version 3). Note that you must at least have 2 nodes (CAN232 works as one node if it is not set into "listen only mode") 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. The CAN232 can also be tested with the sample programs at www.can232.com.

Example for testing CAN232

V[CR]	(should reply version, e.g. V1324[CR])
S4[CR]	(set up CAN speed to 125Kbit)
O[CR]	(open the CAN channel, the Yellow LED should be activated)
t1001AA[CR]	(sends ID=0x100 hex with DLC=1 and data 0xAA, one byte)

1.3 CAN232 limitations

There are of course limitations of how many CAN frames the CAN232 can send & receive. Current version (V13nn) is tested with a throughput of sending 500 standard 11bit frames with 8 databytes at 125kbit CAN bitrate and 115,200 baudrate of the RS232. The bottle neck is of course the RS232 side and the microcontroller not being able to handle more frames per second. 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 transmit FIFO can handle 8 frames (standard or extended) while the receive FIFO can handle 32 frames (standard or extended). Furthermore the CAN232 has only a small RS232 buffer, so it can only handle one or tow command at a time, meaning before sending the next command to it, you must wait for an answer from 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 ASYNC LIB (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.

We strongly recommend that you use the new "AUTO POLL/SEND" feature of the new CAN232 instead of using the old **P** and **A** commands. See **X** command for more info on how to turn on this new feature and disable the old functions (this behaviour is set in EEPROM and will be remembered on next power up, so there is no need to set this each time you configure it up). This new feature also changes the reply back from **t** and **T** commands, so instead of just replying [CR] back when sending a frame it responds z[CR] or Z[CR] depending on command. The reason for this is only to make it simple to parse commands from the CAN232 (i.e. to recognize that a transmit command was successful inbetween frames that are being sent out from the CAN232).

Always start each session (when your program starts) with sending 2-3 [CR] to empty any prior comand or queued character in the CAN232 (many times are power up there could be false characters in the queue or old ones that was from a previous session), 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. Incomming frames from the CAN bus will be sent out at ones on the RS232 or queued in the FIFO if the RS232 is full. Then once in a while send the **F** command to see if there are any errors (e.g. each 100-500mS 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 or application within your program that there is a communication error (e.g. a damaged RS232 tranceiver or power failure).

However the www.can232.com website offers many sample programs as well as a freeware RS232 LIB for Delphi. These programs are free to use or alter to suit your needs.

1.5 Version Information

The version number of CAN232 consists of 2 versions, one for the hardware and one for the software. These two version numbers are combined into one unique version string with 5 characters starting with a V, then 2 characters for hardware and finally 2 characters for software. E.g. version V1324 indicates that it is hardware version 1.3 and software version 2.4. If we update the hardware we will increase version number of the 2 first characters and if we add or change commands or correct bugs the software version number will increase. To see if your CAN232 supports the commands in this manual, check which version number you have by sending the **V** command to the CAN232 (see under commands how it works). Each command in this manual also list a version number and it indicates which version the command works in (i.e. we add commands continiously and the version information indicatates then which version these commands are availible from).

2.0 Availible CAN232 ASCII Commands:

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

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 in normal mode (sending & receiving).
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, yellow LED is turned ON.

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

L[CR]

Open the CAN channel in listen only mode (receiving).
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).

Note: It's not possible to send CAN frames (t, T, r & R)

Example: `L[CR]`
Open the channel, yellow LED is blinking.

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, yellow LED is turned OFF.

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

tiiildd...[CR]

Transmit a standard (11bit) CAN frame.
This command is only active if the CAN232 is open in normal mode.

iii	Identifier in hex (000-7FF)
l	Data length (0-8)
dd	Byte value in hex (00-FF). Numbers of dd pairs must 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: If Auto Poll is disabled (default) the CAN232 replies CR (Ascii 13) for OK or BELL (Ascii 7) for ERROR.
If Auto Poll is enabled (see X command) the CAN232 replies z[CR] for OK or BELL (Ascii 7) for ERROR.

Tiiiiiiiildd...[CR] Transmit an extended (29bit) CAN frame.
This command is only active if the CAN232 is open in normal mode.

iiiiiiii	Identifier in hex (00000000-1FFFFFFF)
l	Data length (0-8)
dd	Byte value in hex (00-FF). Numbers of dd pairs must match the data length.

Example 1: `t00000100211 33[CR]`
Sends a 29bit CAN frame with ID=0x100, 2 bytes with the value 0x11 and 0x33.

Returns: If Auto Poll is disabled (default) the CAN232 replies CR (Ascii 13) for OK or BELL (Ascii 7) for ERROR.
If Auto Poll is enabled (see X command) the CAN232 replies Z[CR] for OK or BELL (Ascii 7) for ERROR.

riiil[CR] Transmit an standard RTR (11bit) CAN frame.
This command is only active if the CAN232 is open in normal mode.

iii	Identifier in hex (000-7FF)
l	Data length (0-8)

Example 1: `r1002[CR]`
Sends an 11bit RTR CAN frame with ID=0x100 and DLC set to two (2 bytes).

Returns: If Auto Poll is disabled (default) the CAN232 replies CR (Ascii 13) for OK or BELL (Ascii 7) for ERROR.
If Auto Poll is enabled (see X command) the CAN232 replies z[CR] for OK or BELL (Ascii 7) for ERROR.

Riiiiiiiil[CR] Transmit an extended RTR (29bit) CAN frame.
This command is only active if the CAN232 is open in normal mode.

iiiiiiii	Identifier in hex (00000000-1FFFFFFF)
l	Data length (0-8)

Example 1: `t000001002[CR]`
Sends an 11bit RTR CAN frame with ID=0x100 and DLC set to two (2 bytes).

Returns: If Auto Poll is disabled (default) the CAN232 replies CR (Ascii 13) for OK or BELL (Ascii 7) for ERROR.
If Auto Poll is enabled (see X command) the CAN232 replies z[CR] 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.
NOTE: This command is disabled in the new AUTO POLL/SEND feature from version V1220. It will then reply BELL if used.

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). If the TIME STAMP is enabled, it will reply back the time in milliseconds as well after the last data byte (before the CR). For more information, see the Z command.

A[CR]

Polls incomming FIFO for CAN frames (all pending frames)
This command is only active if the CAN channel is open.
NOTE: This command is disabled in the new AUTO POLL/SEND feature from version V1220. It will then reply BELL if used.

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). If the TIME STAMP is enabled, it will reply back the time in milliseconds as well after the last data byte (before the CR). For more information, see the Z command.

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]`

Continues on next page...

F[CR] Continued from previous page

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!

Xn[CR]

Sets Auto Poll/Send ON/OFF for received frames. This command is only active if the CAN channel is closed. The value will be saved in EEPROM and remembered next time the CAN232 is powered up. It is set to OFF by default, to be compatible with old programs written for CAN232. Setting it to ON, will disable the P and A command plus change the reply back from using the t and T command (see these commands for more information on the reply). We strongly recommend that you set this feature and upgrade from the old polling mechanism. By doing this, you will save bandwith and increase number of CAN frames that can be sent to the CAN232. When this feature is set, CAN frames will be sent out on the RS232 as soon as the CAN channel is opened.

Example 1: `X0[CR]`
Turn OFF the Auto Poll/Send feature (default).

Example 2: `X1[CR]`
Turn ON the Auto Poll/Send feature.

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

Wn[CR]

Filter mode setting. By default CAN232 works in dual filter mode (0) and is backwards compatible with previous CAN232 versions. It is now possible to put this into single filter mode and the setting is remembered on next startup since it is saved in EEPROM. Command can only be sent if CAN232 is initiated but not open.

Example 1: `X0[CR]`
Set dual filter mode (default).

Example 2: `X1[CR]`
Set single filter mode.

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

Mxxxxxxxx[CR] Sets Acceptance Code Register (ACn Register of SJA1000). This command is only active if the CAN channel is initiated and not opened.

xxxxxxxx Acceptance Code in hex with LSB first, AC0, AC1, AC2 & AC3. For more info, see Philips SJA1000 datasheet.

Example: `M00000000[CR]`
Set Acceptance Code to 0x00000000
This is default when power on, i.e. receive all frames.

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

mxxxxxxxx[CR] Sets Acceptance Mask Register (AMn Register of SJA1000). This command is only active if the CAN channel is initiated and not opened.

xxxxxxxx Acceptance Mask in hex with LSB first, AM0, AM1, AM2 & AM3. For more info, see Philips SJA1000 datasheet.

Example: `mFFFFFFFF[CR]`
Set Acceptance Mask to 0xFFFFFFFF
This is default when power on, i.e. receive all frames.

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

Setting Acceptance Code and Mask registers

The Acceptance Code Register and the Acceptance Mask Register works together and they can filter out 2 groups of messages. For more information on how this work, look in the SJA1000 datasheet. In 11bit ID´s it is possible to filter out a single ID this way, but in 29bit ID's it is only possible to filter out a group of ID's. The example below will set a filter to only receive all 11bit ID's from 0x300 to 0x3FF.

Commands	*Comments*
M00006000[CR]	AC0=0x00, AC1=0x00, AC2=0x60 & AC3=0x00
m00001FF0[CR]	AM0=0x00, AM1=0x00, AM2=0x1F & AM3=0xF0

The first command tells the filter 2 to match 2 bits and if they are not set (in this case it corresponds to 0x3nn, the 3). The second command tells the nn to be don't care, so it could be from 0 to FF, though not so easy to read, since they are not placed nice in a row in memory. Filter 1 s turned off (uses AM0, AM1 & half lower AM3). The last byte in the mask could also be 0xE0 instead of 0xF0, then we filter out the RTR bit as well and you wont accept RTR frames.

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]`

N[CR]

Get Serial number of the CAN232.
This command is only active always.

Example: `N[CR]`
Get Serial number

Returns: N and a 4 bytes value for serial number plus CR (Ascii 13) for OK. E.g. `NA123[CR]`
Note that the serial number can have both numerical and alfa numerical values in it. The serial number is also printed on the CAN232 for a quick reference, but could e.g. be used in a program to identify a CAN232 so the program know that it is set up in the correct way (for parameters saved in EEPROM).

Zn[CR]

Sets Time Stamp ON/OFF for received frames only.
This command is only active if the CAN channel is closed.
The value will be saved in EEPROM and remembered next time the CAN232 is powered up. This command shouldn't be used more than when you want to change this behaviour. It is set to OFF by default, to be compatible with old programs written for CAN232.
Setting it to ON, will add 4 bytes sent out from CAN232 with the A and P command or when the Auto Poll/Send feature is enabled.
When using Time Stamp each message gets a time in milliseconds when it was received into the CAN232, this can be used for real time applications for e.g. knowing time inbetween messages etc.
Note however by using this feature you will decrease bandwith on the RS232, since it adds 4 bytes to each message being sent.

If the Time Stamp is OFF, the incomming frames looks like this:
`t10021133[CR]` *(a standard frame with ID=0x100 & 2 bytes)*

If the Time Stamp is ON, the incomming frames looks like this:
`t100211334D67[CR]` *(a standard frame with ID=0x100 & 2 bytes)*
Note the last 4 bytes 0x4D67, which is a Time Stamp for this specific message in milliseconds (and of course in hex). The timer in the CAN232 starts at zero 0x0000 and goes up to 0xEA5F before it loop arround and get's back to 0x0000. This corresponds to exact 60,000mS (i.e. 1 minute which will be more than enough in most systems).

Example 1: `Z0[CR]`
Turn OFF the Time Stamp feature (default).

Example 2: `Z1[CR]`
Turn ON the Time Stamp feature.

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

Qn[CR]

Auto Startup feature (from power on). Command works only when CAN channel is open. Use this function when you have set up CAN speed and filters and you want the CAN232 to boot up with these settings automatically on every power on. Perfect for logging etc. or when no master is availible to set up the CAN232.

Note: Auto Send is only possible (see X command), so CAN frames are sent out automatically on RS232 when received on CAN side. No polling is allowed.

Example 1: `Q0[CR]`
Turns OFF the Auto Startup feature (default). On next power up, the CAN232 works normally waiting for commands for setup etc.

Example 2: `Q1[CR]`
Turn ON the Auto Startup feature in normal mode. Filters etc. are save and used on next power up.

Example 3: `Q2[CR]`
Turn ON the Auto Startup feature in listen only mode. Filters etc. are save and used on next power up. This dissables t, T, r and R commands!

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