Title: **AVR controlled USB<>CAN Bus Interface**
Issue: F
Author: Michael Wolf
Printed: 04/03/2015
A black bar on the left side marks changes since last issue.

Intro

The purpose of this project is a simple and easy to use USB <> CAN bus interface.
Heart of the circuit is an Atmel AVR ATmega162 microcontroller which controls the Communication between FTDI FT245BM USB to parallel converter and Philips SJA1000 Stand-alone CAN bus controller.
The SJA1000 is memory mapped into memory space of the AVR, which reduces data and address line handling and increases data transfer speed.
An FT245BM handles all USB communication and provides data transfer from and to PC application. An extra EEPROM can store the multi device template for the FT245BM.
A Philips PCA82C250/251 works as CAN controller interface.
Firmware is written in C, compatible to latest compiler version WinAVR.

Since version v1.07 the firmware is compatible to the CANDIP hardware.
See the CANDIP homepage for more details about the required hardware CANDIP/M162.

Hardware description

IC1

CAN controller interface
Note:
Use a PCA82C251 if 24V compatibility is required.

IC2

CAN controller

IC3

Main microcontroller
The AVR is clocked from the SJA1000 with same crystal frequency (16MHz)
Note:
The pin out shown in schematic is for ATmega162 in SMD TQFP44 package!

IC4

Voltage converter for power supply via CAN bus connector
The circuit is also powered via USB via D3.

IC5

USB communication controller

IC6

EEPROM for multi device template
A template is needed if more than one of the same device is connected to a PC.

JP1

Jumper for CAN bus termination

D1

Red LED for error indication
Blinks if an error was detected by SJA1000.
Continuous light if a bus error was detected.

D2

Green LED for CAN traffic indication, both Rx and Tx.

D3

Power supply via USB connection. This diode must be a low drop version!
The diode is just a safety feature but reduces the voltage for the main circuit.
I run my own circuit without diode. The power supply is then shared between USB and 78M05 (if externally powered).

K1

CAN bus connector
Either 10 pin header or 9 pin Sub-D male connector
Pin out in accordance with CiA Draft Standard 102-2

K2

USB type B connector

K3

ISP header for in circuit programming of the AVR.
Compatible to Atmel STK500 ISP header.
Pin out:

1 - MISO	4 - MOSI
2 - +5V	5 - Reset
3 - SCK	6 - GND

Firmware configuration and description

In file main.h:

- Hardware version
- Software version
- Device serial number
- Control commands

In file sja1000.h:

- Pin connection for SJA1000
- SJA1000 modes of operation
- Fixed bit rate values for BTR0 and BTR1

In file usb.h:

- Pins for USB FT245BM connection

The current firmware uses NO FIFO so far for Rx and Tx messages. Some more practice tests are necessary to check if FIFOs are needed in further versions.

ATmega162 Fuse setting

The ATmega162 controller will run with external clock source enabled. This controller is shipped with internal RC oscillator enabled at 1MHz. After changing the fuses the AVR is only accessible via ISP if the SJA1000 is connected!

Fuse Settings in AVR-Studio
Checked Boxes:

- ☑ SPI Enabled; (SPIEN=0)
- ☑ Preserve EEPROM memory;(EESAVE=0)
- ☑ Boot flash section size = 128 words;(BOOTSZ=11)
- ☑ Brown-out detection level at Vcc=2.7V;(BODLEVEL=101)
- ☑ Ext. Clock; Start-up time: 6CK + 4.1ms;(CKSEL=0000;SUT=01)

All other boxes unchecked.

Fuse Settings in PonyProg2000
Checked Boxes:

- ☑ SPIEN
- ☑ EESAVE
- ☑ BOD1LEVEL
- ☑ SUT1
- ☑ CKSEL3
- ☑ CKSEL2
- ☑ CKSEL1
- ☑ CKSEL0

All other boxes unchecked.

Use our favourite AVR programmer to download the firmware into the microcontroller.

These fuse settings are also valid for the CANDIP hardware.

USB Driver installation

A virtual COM port (VCP) driver is required for used FT245BM USB to parallel converter.
The driver can be found on FTDI website at: http://www.ftdichip.com/

This device works also with DirectDLL D2XX drivers from FTDI. The D2XX driver is required for use with CAN-Control software.

Device control commands

The device can be controlled with simple ASCII commands from any terminal tool (using the VCP driver).

Each command consists of one character for command identification and, depending upon command, several data bytes.

Each command must end with a carriage return [CR] ASCII code 13.
All command identifiers are case sensitive.

All other characters for data bytes are NOT case sensitive, but must be valid hex characters (0-9 A-F a-f).

The firmware checks also required command length against given data.
Any detected error will be indicated with return of character [BEL] ASCII code 7.

The following control commands are implemented:

A[CR]

Read last stored Arbitration Lost Capture register content from last AL interrupt.

Return: Axx[CR]

xx = One byte hexadecimal with register content.(00-FF)

C[CR]

This command switches the CAN controller from operational in reset mode. The controller is no longer involved in bus activities.
Command is only active if controller was set to operational mode with command “O” before.

Return: [CR] or [BEL]

E[CR]

Read last stored Error Capture register content from last bus error interrupt.

Return: Exx[CR]

xx = One byte hexadecimal register content.(00-FF)

F[CR]

This command reads the error flags from CAN controller.

Return: Fxx[CR] or [BEL]

xx = one byte hexadecimal with error flags

Bit 0 Not used
Bit 1 Not used
Bit 2 Error warning
Bit 3 Data overrun
Bit 4 Not used
Bit 5 Error passive
Bit 6 Arbitration Lost
Bit 7 Bus error

The red error indication LED will blink if an error interrupt was triggered from SJA1000.
A bus error will generate a constant red LED light.
Command “F” and command “S” will clear a bus error indication.

For detailed error description see the SJA1000 datasheet.

Gxx[CR]

Read register content from SJA1000 controller.

xx = Register to read (00-7F)

Return: Gdd[CR]

dd = Data in register

L[CR]

This command will switch the CAN controller in Listen Only mode. No channel open command (“O”) is required after issuing “L”.
Use the close channel command “C” to return to reset mode, for re-init of SJA1000 send a set bit rate command “s” or “S”.

Return: [CR]

Mxxxxxxxx[CR]

Set acceptance code register of SJA1000. This command works only if controller is setup with command “S” and in reset mode.

xxxxxxxx = Acceptance Code in hexadecimal, order ACR0 ACR1 ACR2 ACR3

Default value after power-up is 0x00000000 to receive all frames.

Return: [CR] or [BEL]

mxxxxxxxx[CR]

Set acceptance mask register of SJA1000. This command works only if controller is setup with command “S” and in reset mode.

xxxxxxxx = Acceptance Mask in hexadecimal, order AMR0 AMR1 AMR2 AMR3

Default value after power-up is 0xFFFFFFFF to receive all frames.

Return [CR] or [BEL]

The acceptance filter is defined by the Acceptance Code Registers ACRn and the Acceptance Mask Registers AMRn. The bit patterns of messages to be received are defined within the acceptance code registers. The corresponding acceptance mask registers allow to define certain bit positions to be ‘don’t care’.

This device uses dual filter configuration.
For details of ACR and AMR usage see the SJA1000 datasheet.

N[CR]

Read serial number from device.

Return: Nxxxx[CR]

xxxx = Serial number in alphanumeric characters.

O[CR]

This command switches the CAN controller from reset in operational mode. The controller is then involved in bus activities. It works only if the initiated with “S” or “s” command before, or controller was set to reset mode with command “C”.

Return: [CR] or [BEL]

riiiL [CR]

This command transmits a standard remote 11 Bit CAN frame. It works only if controller is in operational mode after command “O”.

iii Identifier in hexadecimal (000-7FF)
L Data length code (0-8)

Return: [CR] or [BEL]

RiiiiiiiiL [CR]

This command transmits an extended remote 29 Bit CAN frame. It works only if controller is in operational mode after command “O”.

iiiiiiii Identifier in hexadecimal (00000000-1FFFFFFF)
L Data length code (0-8)

Return: [CR] or [BEL]

Sn[CR]

This command will set the CAN controller to a predefined standard bit rate.
It works only after power up or if controller is in reset mode after command “C”.
The following bit rates are available:

S0	10Kbps	S5	250Kbps
S1	20Kbps	S6	500Kbps
S2	50Kbps	S7	800Kbps
S3	100Kbps	S8	1Mbps
S4	125Kbps	S9	95.238kbps
Sa	8.333Kbps	Sb	47.619kbps
Sc	33.333Kbps	Sd	5kbps

Return: [CR] or [BEL]

sxxyy[CR]

This command will set user defined values for the SJA1000 bit rate register BTR0 and BTR1. It works only after power up or if controller is in reset mode after command “C”.

xx = hexadecimal value for BTR0 (00-FF)
yy = hexadecimal value for BTR1 (00-FF)

Return: [CR] or [BEL]

tiiiLDDDDDDDDDDDDDDD[CR]

This command transmits a standard 11 Bit CAN frame. It works only if controller is in operational mode after command “O”.

iii Identifier in hexadecimal (000-7FF)
L Data length code (0-8)
DD Data byte value in hexadecimal (00-FF). Number of given data bytes will be checked against given data length code.

Return: [CR] or [BEL]

TiiiiiiiiLDDDDDDDDDDDDDDD[CR]

This command transmits an extended 29 Bit CAN frame. It works only if controller is in operational mode after command “O”.

iiiiiiii Identifier in hexadecimal (00000000-1FFFFFFF)
L Data length code (0-8)
DD Data byte value in hexadecimal (00-FF). Number of given data bytes will be checked against given data length code.

Return: [CR] or [BEL]

V[CR]

Read hardware and firmware version from device.

Return: Vhhff[CR]

hh = hardware version
ff = firmware version

v[CR]

Read detailed firmware version from device.

Return: vmami[CR]

ma = major version number
mi = minor version number

Wrrdd[CR]

Write SJA1000 register with data.
The data will be written to specified register without any check!

rr = Register number (00-7F)
dd = Data byte (00-FF)

Return: [CR]

Z[CR]

This command will toggle the time stamp setting for receiving frames. Time stamping is disabled by default, but a change of this setting will be stored in EEPROM and remembered for the next time. So this command needs to be issued only if necessary.

If time stamping is enabled for received frames, an incoming frame includes 2 more bytes at the end which is a time stamp in milliseconds.
The time stamp starts at 0x0000 and overflows at 0xEA5F which is equal to 59999ms.
Each increment time stamp indicates 1ms within the 60000ms frame.
The time stamp counter resets if this setting is turned ON.

All incoming frames are sent via USB after successful receiving, optional with time stamp.
No polling is needed.

They will be sent in the following format:

11 Bit ID Frame
tiiiLDDDDDDDDDDDDDDD[ssss][CR]

11 Bit ID Remote Frame
riiiL[ssss][CR]

29 Bit ID Frame
TiiiiiiiiLDDDDDDDDDDDDDDD[ssss][CR]

29 Bit ID RemoteFrame
RiiiiiiiiL[ssss][CR]

r = Identifier for Remote 11 bit frame
R = Idenifier for Remote 29 bit frame
t = Identifier for 11 bit frame
T = Idenifier for 29 bit frame
i = ID bytes (000-7FF) or (00000000-1FFFFFFF)
L = Data length code (0-8)
DD = Data bytes (00-FF)
ssss = Optinal time stamp (0000-EA5F)

Overview about command status (active or invalid) depending on controller mode.
X = active/valid

Command	Operational Mode Channel Open	Reset Mode Channel Closed
A	X	
C	X	
E	X	
F	X	
G	X	X
L		X
M		X
m		X
N	X	X
O		X
r	X	
R	X	
S		X
s		X
t	X	
T	X	
V	X	X
v	X	X
W	X	X
Z	X	X

Additional Notes

You will find schematic in the appendix of this document.

PCB Layout file is available on request. Format is T3001 for Target 3001 v11 layout editor from http://www.ibfriedrich.com .

Any comments, suggestions or critic are welcome.
Feel free to mail me at: Michael@mictronics.de or use my forum at http://www.mictronics.de

Recommended documents:
Philips SJA1000 Datasheet
www.semiconductors.philips.com/acrobat/datasheets/SJA1000_3.pdf

Application Note 97076 – SJA1000 Stand alone CAN controller
www.semiconductors.philips.com/acrobat/applicationnotes/AN97076.pdf

Disclaimer:

By using this circuit or description or other material presented, then you (meaning the reader of this material being yourself, or any other person that you may subsequently pass any material to) explicitly accept the following.
I will accept no liability for loss or damages that may be imposed upon you by any Court of Law, statutory or other body. It is entirely the reader's responsibility to determine the suitability of any design for the intended purpose. The project is presented "as is" and are believed to be without error, however this cannot be guaranteed, and it is reasonable to assume that mistakes or other errors will occur from time to time. If errors are found, please contact me, Michael and describe the error (and its consequences) so that corrections may be made.

License for firmware source code

You can contact me, Michael, by:
WWW: http://www.mictronics.de
E-mail: michael@mictronics.de