USB–I2C (I^{2}C Interface)

Description

The USB–I2C module provides an interface between your PC and your devices on I^{2}C (inter-integrated circuit) bus[1]. The module (hereinafter the adapter) is self powered from the USB host and can supply up to 70 mA at 3.3 V or 5 V for external circuitry from a standard 100 mA USB port. The adapter acts as an I^{2}C master only.

After connecting the adapter to the computer USB port an extra (virtual) serial port is created. Application software accesses the adapter in the same way as it would access a standard serial port. No special drivers or libraries are needed, with the exception for older Windows system, see drivers section.

I^{2}C bus voltage:	**3.3 V** and **5 V** tolerant
I^{2}C bus frequency:	from **10 kHz** to **10 MHz**
USB standard:	**USB 2.0 Full Speed** (12 MHz)

The adapter supports clock stretching up to one second.

Connections

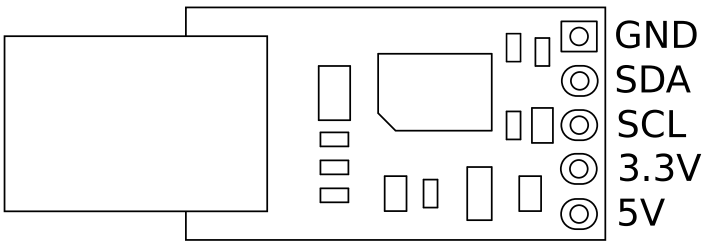

Figure 1: Pins (upper, populated side shown)

The GND pin must be connected to the 0 V (ground) on your I^{2}C device.

The SDA and SCL pins compose the I^{2}C bus. They should be connected directly to the SCL and SDA pins on your I^{2}C device. The pull-up resistor are needed to 3.3 V or 5 V, depending on your I^{2}C device. Sometimes they are already populated on PCB with your device.

The 3.3 V and 5 V supply from the adapter can supply up to 70 mA to external devices. If your I^{2}C device requires more than this, or has its own supply, then leave these pins unconnected. Do not apply power of your own supply to these pins.

[1]Also known as TWI, IIC and I2C bus.

Protocol

On top of serial communication a protocol is defined which allows user to perform operations and send/receive the data on the I²C bus. The protocol defines messages which are sent from the PC to the adapter and which get answered by a reply from the adapter.

To put the adapter into defined (reset) state a RESET message consisting of <ESC> character (0x1B) is used.

Having this convention would disallow the <ESC> character in data (payload). Therefore an another character is used to escape the <ESC> character. This character is the „\“ character (0x5C). To send a literal <ESC> value the sequence 0x5C 0xB1 has to be sent (0xB1 is the <ESC> character 0x1B with its nibbles exchanged). To send a literal „\“ character the sequence 0x5C 0xC5 has to get sent. For all other combinations of 0x5C with another character an error is generated (except for 0x5C 0x1B which is a reset).

Protocol messages and usage

To set an address of target I²C device, a ADDRESS message is used. This message starts with `A` character followed by the address. The address must be less than 128 otherwise an error is generated.

If the adapter needs to send some data to the I²C device, a WLENGTH message is used. This message starts with `W` character followed by the number of bytes to be sent. The number must be less or equal to 64. The bytes that are going to be sent are transfered with WDATA message starting with `w` character followed by the data.

If the adapter needs to receive some data from I²C device, a RLENGTH message is used. This message starts with `R` character followed by the number of bytes that the adapter would receive from the I²C device. The number must be less or equal to 64. The received data are read from the adapter by RDATA command; character `r`. Reading is usually done after START message.

The adapter will return <ESC> character when the given parameter (address, length) is invalid (out of range).

The transfer from adapter to I²C device and from I²C device to the adapter is started by START message. After the message the adapter sends address of target I²C device followed by the data from buffer and restarts I²C bus if RLENGTH is non-zero by sending address once again and reads data until it reads RLENGTH bytes or receive NACK condition. START message consists of S character. After starting the transfer a timeout timer of one second is started. The timer stops when the adapter has sent/received all the data, otherwise it expires and the adapter is put into definited reset state.

Since the transfer operation take some time a STATUS message could be used. This message consists of character `s`. The adapter returns 0xFF if it is still busy or 0x00 if it is idle – ready for accepting new commands.

The last (transmission) status (error) could be obtained by ERROR message which uses character E. The device replies with error code. This response could be: `N` – no error; `U` – unknown escape sequence; `C` – unknown message (command); `L` – invalid length; `A` – invalid address; `T` – timer expired and; `a` – NACK received. After an error sending a RESET message to the adapter is recommended.

By default I^{2}C bus is clocked at 100 kHz (standard mode), but it could be clocked to 400 MHz (fast mode). To change this setting a TIMING messsage is needed. This message consists of character `t` followed by four bytes: 0x13 0x0F 0x42 0xB0 for 100 kHz and 0x09 0x03 0x33 0x50 for 400 kHz. To apply change of frequency a REINIT message is used. This is accomplished by sending character `i` to the adapter. Note that a RESET message does not change the frequency.

Each adapter features an unique (32-bit) serial number in hexadecimal notation that is returned to a PC after SERIAL message: character `n`.

Example

To send bytes 0x03 0x05 0x1B to a I^{2}C device with address 0x27, the following bytes should be sent: `0x1B A 0x27 W 0x03 w 0x03 0x05 0x5C 0xB1 S`. Since we have to escape third data byte four bytes are sent after `w` character.

To send the byte 0xE3 to a I^{2}C device with address 0x40 and receive 3 bytes, the following bytes should be sent: `0x1B A 0x40 W 0x01 w 0xE3 R 0x03 S`, then wait some time for data transmission and continue with `r` to get received bytes.

Drivers

On Linux based operating system, a serial driver is usually provided by the kernel: USB Modem (CDC ACM) support: `cdc-acm.c` and, after connecting the USB–I2C module, the serial device usually appears under `/dev/ttyACM0` device file.

On Windows 10 the system loads generic communication port driver. Windows 7 and 8 need a simple (one text file) driver. Since the adapter features an unique serial number the number of COM port on the system stays the same.

System requirements:

- a free USB port
- Linux, or Windows 7, 8, 10

Document release: Thursday 31st May, 2018

Document last update: Sunday 25th November, 2018

KEL