eScada

v24.3.0

Drivers

eScada.Drivers.USBTin

eScada.Drivers.USBTin

(USBtin - USB to CAN interface based on MCP2515)

OS availability
Windows, Linux, RaspBian

Atomic data type
Byte oriented protocol.

Hardware and documentation reference
www.fischl.de/usbtin/
Document number DS21801E for MCP2515 specification (Microchip Technology Inc.)

Parameters available in every section

Channel:	none	
Device:	COM Port	Serial port name depending on OS type. e.g. Linux: /dev/ttyS0, /dev/ttyUSB0 e.g. Windows: COM1, COM3
	CAN Bus Baud rate	CAN Bus communication baud-rate, standard or user defined
	CAN Bus interface mode	Active (default, receive and transmit), Listen Only, Loop Back
	Mask RXM0	RXB0 Buffer mask
	Acceptance filter RXF0 to RXF1	RXB0 Buffer acceptance filters
	Mask RXM1	RXB1 Buffer mask
	Acceptance filter RXF2 to RXF5	RXB1 Buffer acceptance filters
	Reconnect timeout [ms]	Waiting time before a reconnection after COMM break-down
	Receiving Frames Interval [ms]	Timeout interval used to wait for bus frames.
Group:	none	
Tag:	none	

Useful Linux commands

COM List:	`dmesg | grep tty`
COM rights:	`sudo chmod a+rw /dev/ttyUSB0`
COM user info:	`ls -l /dev/ttyUSB0`
COM add user:	sudo adduser *username* dialout - (dialout is the default group)

Further information is contained in the Troubleshooting section at www.fischl.de/usbtin/

CAN Bus frames supported
- Receive/Transmit standard (11 bit) frame.
Identifier in hexadecimal format (000-7FF)
Data length (0-8)

- Receive/Transmit extended (29 bit) frame.
Identifier in hexadecimal format (0000000-1FFFFFFF)
Data length (0-8)

- Receive/Transmit standard RTR (11 bit) frame.
Identifier in hexadecimal format (000-7FF)
Data length (0-8)

- Receive/Transmit extended RTR (29 bit) frame.
Identifier in hexadecimal format (0000000-1FFFFFFF)
Data length (0-8)

The RTR frame, is a frame requesting the transmission of a specific CAN identifier.

Addressing

The tag address can be specified as follow:

FS.hhh	Receive/Transmit standard (11 bit) frame
FE.hhhhhhhh	Receive/Transmit extended (29 bit) frame
RS.hhh	Receive/Transmit standard RTR (11 bit) frame
RE.hhhhhhhh	Receive/Transmit extended RTR (29 bit) frame

<u>CAN ID identifier</u>

hhh	CAN ID identifier in hexadecimal format (000-7FF)
hhhhhhhh	CAN ID identifier in hexadecimal format (0000000-1FFFFFFF)

During tag declaration, only unsigned byte can be specified as data type; it is the default data type.

- Receiving frames
They must be declared with the attribute 'Read only' specified as TRUE
The content of these frames will be filled up as soon as their CAN ID is received.

- Sending frames
They must be declared with the attribute 'Read only' specified as FALSE

<u>Cycling mode</u>

This mode can be implemented specifying the polling attribute with a value greater than ZERO.
These kind of frames will be sent even every time a value is changed.
In case of RTR frames, their content will be filled up as soon as their CAN ID is received.

<u>On change mode</u>

This mode can be implemented specifying the polling attribute with a value of ZERO.
These kind of frames will be sent only every time a value is changed.

Consecutive items

CAN Bus protocol allows from 0 to 8 items (bytes), each frame.

Message acceptance filters and masks

The MCP2515 offers two filter chains.
Each chain consists of one mask and a set of filters.

```
        RXM0          RXM1
          |             |
        RXF0          RXF2
        RXF1          RXF3
                      RXF4
                      RXF5
```

Bit-mask:
0 = accept (accept regardless of filter)
1 = check (accept only if RXM or RXF matches)

mask examples 29bit:
mask = 1FFFFFFF Check whole extended id
mask = 1FFFFF00 Check extended id except last 8 bits

mask examples 11bit:

mask = 7FF, (byte)00, (byte)00	check whole id, data bytes are irrelevant
mask = 7F0, (byte)00, (byte)00	check whole id except last 4 bits, data bytes are irrelevant
mask = 7Ff0, (byte)FF, (byte)00	check whole id except last 4 bits, check first data byte, second is irrelevant

More information on document number DS21801E for MCP2515 specification, pages 32, 33

Filter syntax for standard 11bit frames

```
S.7FF.FF.0
| |    |   |Second data byte (optional)
| |    |
| |    | First data byte (optional)
| |
| |Mask (required)
|
|Type of filter (required S=11bit)
```

Filter syntax for extended 29bit frames

```
E.7FFFFFFF
| |
| |Mask (required)
|
|Type of filter (required E=29bit)
```

Remarks: all numeric values must be expressed in hexadecimal format.

Example 1:
How to set a filter in order to get data only for 70B standard identifier and the first byte of data is 5

```
RXM0  →     S.7FF.FF
RXF0  →     S.70B.5
```

Example 2:
How to set a filter in order to get data only for 18F00F0B extended identifier

```
RXM0  →     E.1FFFFFFF
RXF0  →     E.18F00F0B
```

Example 3:
How to set a filter in order to get data for 18B and 70B standard identifiers

```
RXM0  →     S.7FF
RXF0  →     S.18B
RXF1  →     S.70B
```

Device tag information

Read time [ms]	This is the time between two received frames.
Write time [ms]	This is the time between two sent frames.

How to get values from frames

The values contained in CAN bus frames can be evaluated or written mainly using the following derived variables:

HexToValue, ConvertTo, SplitBits, Function, ValueToHex, SetItem, Copy