

## CANLY V2.0

Standalone programmable I/O controller for the CAN bus

Datasheet Rev.: 1.0  
Date: 20.10.2006

The CANLY modules operate as I/O expanders for a Controller Area Network (CAN) system, supporting CAN V2.0B active, with bus rates up to 1 Mbit/s.

One mask and two acceptance filters are provided to give maximum flexibility during system design with respect to identifiers that the device will respond to.

The device is pre-programmed in non-volatile memory so that the module defaults to a specific configuration at power-up

Through an RS-232 port, the module can be configured and I/O operations can be performed locally. In addition, the serial interface can output received and transmitted CAN messages

### Functional Features

- Programmable bit rate up to 1 Mb/s
- Meets CAN specifications 2.0A (11 bit ID) and 2.0B (29 bit ID)
- One programmable message acceptance mask
- Two programmable message acceptance filters
- Non-volatile memory for user configuration
- User configuration automatically loaded on power-up
- Field upgradeable firmware through serial port
- Individually selectable transmit-on-pinchange for each input
- Message scheduling capability
- Device configuration can be modified via CAN bus messages

### Hardware Features

- Microchip **MCP2515** CAN Controller
- Atmel **ATmega168** RISC CPU @ 16MHz
- **4 protected digital inputs**, configurable for switch to GND, 12V or 24V voltage input
- **4 digital open drain outputs**,
- one output configurable to a **10-bit PWM output**
- **1 RS-232 Serial Interface**
- An **expansion** connector with SPI, I2C, Interrupt and Reset signals
- **Voltage regulator** with input protection, suitable for vehicle operation
- 3 control LEDs.

### Ordering Information

#### Art.-No. 01.0139

CANLY Standlone Programmable CAN I/O Controller, OEM Board

#### Contact:

#### Elektronik-Atelier Kallen

Steinackerweg 14  
CH-3075 Rüfenacht / Switzerland

Web: [www.avrcard.com](http://www.avrcard.com)

Email: [info@avrcard.com](mailto:info@avrcard.com)

Tel: +41-31-832 1441

Fax: +41-31-832 1442

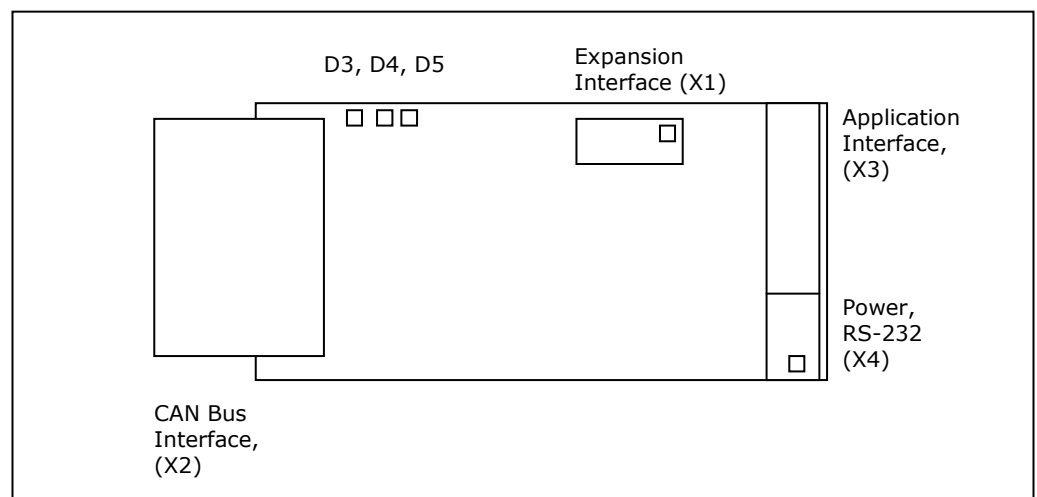
Preliminary

## Specifications

**Table 1 – Technical Specifications**

<b>CAN Bus</b>	
Bit Rate	Up to 1Mbit/s
Controller	Microchip MCP2515 @ 16MHz clock
Transceiver	Philips PCA82C251, for 24V systems
Connector	Standard 9-pin D-Sub connector for CAN bus, as per CiA DS102-1
Termination	On-board 120Ohm selectable
<b>MICROCONTROLLER</b>	
Type	Atmel ATmega168 8-bit RISC, @ 16MHz clock
Memory	16KB flash for program storage 1KB SRAM 512 Bytes EEPROM
<b>I/O</b>	
Inputs	Configurable by hardware for switch to GND, 12V or 24V voltage input Protected with schottky diodes and series resistor
Outputs	25V/0.6A, open drain
Terminal	Screw terminal, 0.1" pitch, max 0.5mm <sup>2</sup> cables
Expansion	5x2-way 0.1" header for SPI, I2C, Reset, Interrupt, +5V, GND
<b>Power Supply</b>	
Requirement	7...12VDC, 70mA, reverse-polarity protected
Source	Dedicated DC supply, or supplied via CAN bus cable
<b>Mechanical</b>	
Size (L x W)	70 x 37 mm
Weight	26 g

**Figure 1 - Connectors**



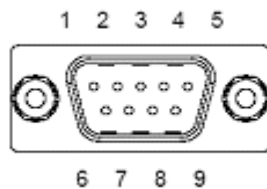
Pin 1 is marked with a square.

**Table 2 – CAN Bus Interface Pinout**

X2: CAN Interface		
No.	Function	
1		
2	CANL	
3	CAN GND	
4		
5		
6	AUX GND	DC Input 0V
7	CANH	
8		
9	VCAN+	DC Input +

CAN GND should be connected to shield of bus cable only.

The CANly module can be powered either from X4 or from X2. On X2, the positive voltage is applied to VCAN+, negative to AUX GND

**Figure 1 – View to front of CAN connector****Table 3 – Expansion Interface Pinout**

X1: Expansion		
No.	Function	Alt. Function
1	+5V	VCC
2	/CS	Connected to PB2
3	SCK	SPI Bus
4	MOSI	
5	MISO	
6	GND	Ground
7	/RESET	CPU Reset In/Out
8	/INT	Interrupt input (PD2)
9	SCL	I2C Bus
10	SDA	

**Table 4 – Application Interface Pinout**

X3: Application Interface		
No.	Function	Configuration
1	Input 0	0...12V
2	Input 1	0...12V
3	Input 2	0...12V
4	Input 3	0...12V
5	Output 0	Open Drain, 25V/0.6A, PWM
6	Output 1	Open Drain, 25V/0.6A
7	Output 2	Open Drain, 25V/0.6A
8	Output 3	Open Drain, 25V/0.6A

**Table 5 -Power Supply, RS-232 Interface Pinout**

X4: Power/RS-232		
No.	Function	
1	DC Input +	7...12VDC
2	GND	Ground
3	RS-232 TXD	Serial Interface
4	RS-232 RXD	

**Table 6 – Jumper Settings**

Jumpers			
No.	Des.	ON	OFF
J1	TERM	120 Ohm Resistor between CANH and CANL	N.C.

**Table 7 - LEDs**

LEDs		
No.	Des.	Function
D3	PWR	CANly board is powered
D4	RX	Receive data
D5	TX	Transmit data

## CAN Bus Operation

---

### Power-Up Sequence

The following sections describe the events/actions of the CANLY during normal power-up and operation.

The CANLY goes through a sequence of events at power-on reset (POR) in order to load the programmed configuration and insure that errors are not introduced on the bus.

### Operating Mode

The CANLY initially powers up in Configuration mode. While in this mode, the CANLY will be prevented from sending or receiving messages via the CAN interface. The peripherals are disabled while in this mode.

### Self-Configuration

Once the CANLY is out of reset, it will perform a self-configuration by transferring the contents of the EEPROM array to the corresponding locations within the SRAM array.

Once the self-configuration cycle has successfully completed, the CANLY switches to Normal mode. In this mode, it is ready to send/receive messages via the CAN interface. In addition, the command interface on the RS-232 serial line becomes active. At this point the peripherals are operational, if enabled.

### Scheduled Transmissions

Once the CANLY has gone on bus it will transmit the On Bus message once, regardless of whether scheduled transmissions are enabled or not. This message notifies the network of the CANLY's presence. The On Bus message will (if enabled by STCON.STEN) repeat at a frequency determined by the STCON register.

## CAN Bus Timing

The configuration registers CNF1, CNF2 and CNF3 control the bit timing for the CAN bus interface. For a detailed description and explanation of the CAN bit timing, please refer to the MCP2515 datasheet.

Below the factory default values of CNF1...3 registers are detailed:

Assuming

$F_{osc} = 16\text{MHz}$  and a CAN baudrate of  $125\text{kbit/s}$ ,

with

$BRP = 8$

The resulting Time Quantum (TQ) is

$TQ = 1\mu\text{s}$

The CAN bit time is divided into 8TQ by

$PHSEG1 = 3$

$PHSEG2 = 3$

$PRSEG = 1$

(plus 1 TQ for SYNCH)

The CAN Configuration Registers are then assembled as follows:

$CNF1 = SJW1 + (BRP - 1)$

$CNF2 = (BTLMODE\_SET + (PHSEG1 - 1) * 8 + (PRSEG - 1))$

$CNF3 = (SOF\_DISABLE + WAKFIL\_DISABLE + (PHSEG2 - 1))$

## Transmit Message ID's

The CANLY module contains three separate, user-configurable transmit message ID's: TXID0, TXID1 and TXID2. The data length code is predefined for each of the various output messages, with the data that is transmitted coming directly from the contents of the device's peripheral registers.

### Transmit Message ID0 (TXID0)

TXID0 contains the identifier that is used when transmitting the On Bus message. The On Bus message will be transmitted

- a) once after power-up
- b) at predefined intervals, if enabled (STCON.STEN = 1)

The CAN message will send GPIO data.

### Transmit Message ID1 (TXID1)

TXID1 contains the identifier that is used when the CANLY sends the Command Acknowledge message. This message is sent when the CANLY receives an Input Message and processes the instruction (and OPTREG2.CAEN = 1). This message is used as a hand shake for the node requesting the modification of the CANLY. There is no data associated with this message.

### Transmit Message ID2 (TXID2)

Transmit ID2 contains the identifier that is used when transmitting digital input edge detection messages.

## Message Filtering

### Acceptance Mask

The acceptance mask is used to define which bits in the CAN ID are to be compared against the programmable filters. Individual bits within the mask correspond to bits in the CAN ID that, in turn, correspond to bits in the acceptance filters. Any bit in the mask that is set to a '1' will cause the corresponding CAN ID bit to be compared against the associated filter bit. Any bit in the mask that is set to a '0' is not compared and effectively sets the associated CAN ID bit to 'don't care'.

To insure proper operation of the information request and input messages, some mask bits (as configured in the mask registers) may be ignored as explained:

**Message with a standard ID** - the three least significant bits of a standard identifier (RXMSIDL.SID2:SID0) are 'don't care' for the mask registers and effectively become '0'.

**Message with an extended ID** - the three least significant bits of the standard identifier (RXMSIDL.SID2:SID0) are configurable and the three least significant bits of the extended identifier (RXMEID0.EID2:EID0) are always 'don't cares' and effectively becomes '0'.

### Acceptance Filters

There are two separate acceptance filters defined for the CANLY: RXF0 and RXF1. Each bit in the filters corresponds to a bit in the CAN ID. Every bit in the CAN ID, for which the corresponding Mask bit is set, must match the associated filter bit in order for the message to be accepted. Messages that fail to meet the mask/filter criteria are ignored.

**Note:** The EXIDE bit in the Mask register (RXMSIDL) can be used to mask the IDE bit in the corresponding Receive buffer register (RXBnSIDL).

When changing the filter settings, the CAN controller will enter configuration mode and after completion of the register modification, will resume normal operation. Please note that the CAN controller enters configuration mode only when there are no pending CAN messages to be transmitted.

## Pre-Defined CAN Messages

### Information Request Messages

Information Request Messages (IRM) are messages that the CANLY receives and then responds to by transmitting an output message containing the requested data. IRMs are implemented as a Remote Transfer Request (RTR) message.

When a node in the CAN bus system wants information from the CANLY, it has to send a remote frame on the bus. The identifier for the remote frame must be such that it will be accepted through the CANLY's mask/filter process.

Information Request messages (IRM) must not only meet the mask and filter criteria but must also have the RTR bit of the CAN ID set (since the filter registers do not contain an explicit RTR bit). If a message passes the mask/filter process and the RTR bit is a '0', that message will be ignored.

Once the CANLY has received a remote frame, it will determine the function to be performed based upon the three LSB's of the received remote frame.

**Table 8 – Information Request Messages, Overview**

IRM Type	SID (bits)										RTR	DLC	
	10	9	8	7	6	5	4	3	2	1			0
Read Control Registers	x	x	x	x	x	x	x	*	0	0	1	Y	0
Read Config Registers	x	x	x	x	x	x	x	*	0	1	0	Y	0
Read CAN Error Registers	x	x	x	x	x	x	x	*	0	1	1	Y	0
Read PWM1 Config Reg	x	x	x	x	x	x	x	*	1	0	0	Y	0

### Output Messages

The data frame sent in response to the information request message is defined as an output message. It will have the same identifier as the IRM.

Bit 3 of a standard or extended identifier of the output message will differ from the received information request message in that the value equals '1' for an IRM and equals '0' for the resulting output message.

Output messages contain the requested data in the data field.

**Table 9 – Control Register Output Message**

Bit Field	Parameter	Description
SID	SID of IRM with bit 3 reset	
DLC	7	
DATA[0]	(reserved)	
DATA[1]	(reserved)	
DATA[2]	(reserved)	
DATA[3]	<b>OPTREG2</b>	See p. 15
DATA[4]	<b>STCON</b>	See p. 15
DATA[5]	<b>IOINTEN</b>	See p. 13
DATA[6]	<b>IOINTPO</b>	See p. 13

**Table 10 – Configuration Register Output Message**

Bit Field	Parameter	Description
SID	SID of IRM with bit 3 reset	
DLC	5	
DATA[0]	(reserved)	
DATA[1]	(reserved)	
DATA[2]	<b>CNF1</b>	See p. 14
DATA[3]	<b>CNF2</b>	See p. 14
DATA[4]	<b>CNF3</b>	See p. 14

**Table 11 – CAN Error Register Output Message**

Bit Field	Parameter	Description
SID	SID of IRM with bit 3 reset	
DLC	3	
DATA[0]	<b>CANSTAT</b>	CAN Status Register
DATA[1]	<b>TEC</b>	Transmit Error Counter
DATA[2]	<b>REC</b>	Receive Error Counter

For a detailed description of these registers, please refer to the MCP2515 datasheet.

**Table 12 – PWM Configuration Register Output Message**

Bit Field	Parameter	Description
SID	SID of IRM with bit 3 reset	
DLC	6	
DATA[0]	(reserved)	
DATA[1]	(reserved)	
DATA[2]	<b>T1CON</b>	See p. 13
DATA[3]	(reserved)	
DATA[4]	<b>PWM1DC</b>	See p. 13
DATA[5]	(reserved)	

### Input Messages

Input messages are received and used to change the values of the pre-defined groups of registers. There is also an input message that can change a single register's contents. The primary purpose of input messages are to reconfigure the CANLY parameters (if needed) while in an operating CAN system and are, therefore, optional in system implementation. These messages are in the form of standard (or extended) data frames (per the CAN 2.0B specification) that have identifiers which pass the CANLY's mask/filter process. After passing the mask and filter, the lower three bits of the standard identifier (RXF1SIDL.SID2:SID0) will indicate which register(s) are to be written.

**Table 13 – Write Registers Input Message**

Bit Field	Parameter	Description
SID	Bits 2:0 = 000	
DLC	3	
DATA[0]	<b>Address</b>	Register address as per Table 21 – Register Map, p. 12
DATA[1]	<b>Mask</b>	Bit mask. 0: register bit will not be modified 1: register bit will be modified
DATA[2]	<b>Value</b>	Desired register value

**Table 14 – Write TXIDO Input Message**

Bit Field	Parameter	Description
SID	Bits 2:0 = 001	
DLC	4	
DATA[0]	<b>TXOSIDH</b>	See p. 17
DATA[1]	<b>TXOSIDL</b>	See p. 17
DATA[2]	(reserved)	
DATA[3]	(reserved)	

**Table 15 – Write TXID1 Input Message**

Bit Field	Parameter	Description
SID	Bits 2:0 = 010	
DLC	4	
DATA[0]	<b>TX1SIDH</b>	See p. 17
DATA[1]	<b>TX1SIDL</b>	See p. 17
DATA[2]	(reserved)	
DATA[3]	(reserved)	

**Table 16 – Write TXID2 Input Message**

Bit Field	Parameter	Description
SID	Bits 2:0 = 011	
DLC	4	
DATA[0]	<b>TX2SIDH</b>	See p. 17
DATA[1]	<b>TX2SIDL</b>	See p. 17
DATA[2]	(reserved)	
DATA[3]	(reserved)	

**Table 17 – Write I/O Configuration Input Message**

Bit Field	Parameter	Description
SID	Bits 2:0 = 100	
DLC	5	
DATA[0]	<b>IOINTEN</b>	See p. 13
DATA[1]	<b>IOINTPO</b>	See p. 13
DATA[2]	(reserved)	
DATA[3]	(reserved)	
DATA[4]	(reserved)	

**Table 18 – Write Receive Mask Input Message**

Bit Field	Parameter	Description
SID	Bits 2:0 = 101	
DLC	4	
DATA[0]	<b>RXMSIDH</b>	See p. 15
DATA[1]	<b>RXMSIDL</b>	See p. 15
DATA[2]	<b>RXMEID8</b>	See p. 16
DATA[3]	<b>RXMEIDO</b>	See p. 16

**Table 19 – Write Receive Filter 0 Input Message**

Bit Field	Parameter	Description
SID	Bits 2:0 = 110	
DLC	4	
DATA[0]	<b>RXF0SIDH</b>	See p. 16
DATA[1]	<b>RXF0SIDL</b>	See p. 16
DATA[2]	<b>RXF0EID8</b>	See p. 16
DATA[3]	<b>RXF0EIDO</b>	See p. 16

**Table 20 – Write Receive Filter 1 Input Message**

Bit Field	Parameter	Description
SID	Bits 2:0 = 111	
DLC	4	
DATA[0]	<b>RXF1SIDH</b>	See p. 16
DATA[1]	<b>RXF1SIDL</b>	See p. 16
DATA[2]	<b>RXF1EID8</b>	See p. 16
DATA[3]	<b>RXF1EIDO</b>	See p. 16

## Serial Communication

---

### Setup

#### Connection

The RS-232 serial communication is set up by connecting the TXD, RXD and GND signals (on X4) to a terminal. The communications parameters are:  
19'200bits/s, 8 databits, 1 stopbit, no parity

#### Startup

After power-on, the CANLY module will send an initialization message over the serial port as follows:

```
INI: CANLY V2.1b
>
```

After the `>` prompt appears, the module is ready to receive commands over the serial interface.

### Commands on the Serial Interface

The following commands are available for reading and modifying the CANLY registers. Arguments must be separated from the command mnemonic by one space. Command line must be terminated by a <CR>.

<b>WR:</b>	<b>Write Register</b>
Arguments:	aa mm vv Aa = register address mm = mask bits vv = register value. Values must be given as 2-digit hex numbers, separated by one space
Description:	Writes the byte value to the register with the given address, using the given bit mask. Only those register bits will be modified where the corresponding mask bit is set.
Response:	none
Errors:	ERR: 2 – unknown command ERR: 3 – wrong argument
<b>WA:</b>	<b>Write Address</b>
Arguments:	aa vvvv Aa = register address vvvv = register value. Values must be given as 2 and 4-digit hex numbers, separated by one space
Description:	Writes the word value to the register with the given address (byte order: lo, hi)
Response:	none
Errors:	ERR: 1 – missing argument ERR: 2 – unknown command ERR: 3 – wrong argument
<b>WP:</b>	<b>Write Persistent</b>
Argument:	none
Description:	Writes all registers persistently into the EEPROM.
Response:	Writing EEPROM...

**RD: Read Register**

---

Argument: aa  
 Aa = register address.  
 Value must be given as 2-digit hex number

Description: Reads the value of the register at given address.

Response: REG: aa vv

Errors: ERR: 1 – missing argument  
 ERR: 2 – unknown command  
 ERR: 3 – wrong argument

**UD: Update Flash**

---

Argument: (none)

Description: Invokes firmware upload mode.

Response: Confirm Flash Update (Y|N).  
 Enter Y to confirm, or any other character to abort.  
 For further details on how to upload the firmware, please refer to page 18.

**Debug Output**

When DBGCON.DBGRX bit is set, an output string with the following format will be sent to the RS-232 serial interface when a CAN message is received through the acceptance mask and filters:

```
RCV: SID xxxx RTR DLC: 0
RCV: SID xxxx      DLC: 5 DATA: xx xx xx xx xx
```

When DBGCON.DBGTX bit is set, an output string with the following format will be sent to the RS-232 serial interface when a CAN message is transmitted:

```
TRX: SID xxxx      DLC: 5 DATA: xx xx xx xx xx
```

When DBGCON.DBGAC bit is set, one of the following output strings will be sent to the RS-232 serial interface when an I/O pin configuration changed:

```
ACT: Set outputs to 0000xxxx
ACT: Set PWM duty to xxx
ACT: Set PWM off
ACT: Set Automsg Interval to xxxx ms
ACT: Set Automsg off
```

## Registers

The CANLY allows the user to pre-program registers pertaining to CAN module and device configuration into non-volatile EEPROM memory. In this way, the device is initialized to a default state after power-up. The user registers are transferred to SRAM during the power-up sequence and many of the registers are able to be accessed via the CAN bus once the device establishes a connection with the bus. The registers are summarized in Table 21. A detailed description of each register follows after the table.

**Table 21 – Register Map**

Address	Name	Description	Factory Default	See Page
0x1C	IOINTEN	Enable inputs for Transmit-On-Change feature	0x0F	13
0x1D	IOINTPO	Defines polarity for I/O Transmit-On-Change inputs	0x01	13
0x1E	GPLAT	Output latch	0x00	13
0x21	T1CON	PWM1 Timer Control Register; contains enable bit and DC LSBs	0x00	13
0x25	PWM1DCH	PWM1 Duty Cycle (DC)	0x00	13
0x27	CNF1	CAN module register, configures synchronization jump width and baud rate prescaler	0x74	14
0x28	CNF2	CAN module register, configures propagation segment, phase segment 1, and determines number of sample points	0x90	14
0x29	CNF3	CAN module register, configures phase segment 2	0x02	14
0x2C	STCON	Scheduled transmission control register	0x00	13
0x2D	OPTREG2	Option Register 2	0x00	15
0x30	RXMSIDH	Acceptance Filter Mask, Standard ID MSB	0x0F <sup>1</sup>	15
0x31	RXMSIDL	Acceptance Filter Mask, Standard ID LSB and Extended ID USB	0x00	15
0x32	RXMEID8	Acceptance Filter Mask, Extended ID MSB	0x00	16
0x33	RXMEID0	Acceptance Filter Mask, Extended ID LSB	0x00	16
0x34	RXF0SIDH	Acceptance Filter 0, Standard ID MSB	0x08 <sup>2</sup>	16
0x35	RXF0SIDL	Acceptance Filter 0, Standard ID LSB and Extended ID USB	0x00	16
0x36	RXF0EID8	Acceptance Filter 0, Extended ID MSB	0x00	16
0x37	RXF0EID0	Acceptance Filter 0, Extended ID LSB	0x00	16
0x38	RXF1SIDH	Acceptance Filter 1, Standard ID MSB	0x09 <sup>3</sup>	16
0x39	RXF1SIDL	Acceptance Filter 1, Standard ID LSB and Extended ID USB	0x00	16
0x3A	RXF1EID8	Acceptance Filter 1, Extended ID MSB	0x00	16
0x3B	RXF1EID0	Acceptance Filter 1, Extended ID LSB	0x00	16
0x3C	TXID0SIDH	Transmit Message 0, Standard ID MSB	0x00	17
0x3D	TXID0SIDL	Transmit Message 0, Standard ID LSB	0x3C	17
0x40	TXID1SIDH	Transmit Message 1, Standard ID MSB	0x00	17
0x41	TXID1SIDL	Transmit Message 1, Standard ID LSB	0x3D	17
0x44	TXID2SIDH	Transmit Message 2, Standard ID MSB	0x00	17
0x45	TXID2SIDL	Transmit Message 2, Standard ID LSB	0x3E	17
0x7F	DBGCON	Debug Output Control Register	0xE0	16

<sup>1</sup> Resulting SID: 0x07F8

<sup>2</sup> Resulting SID: 0x0040

<sup>3</sup> Resulting SID: 0x0048

**Register 1: IOINTEN – Enable inputs for Transmit-On-Change**

U-0	U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1
				<b>IN3TXC</b>	<b>IN2TXC</b>	<b>IN1TXC</b>	<b>IN0TXC</b>
Bit 7							Bit 0

bit 3-0 **IN3TXC:IN0TXC**: Transmit-on-change Enable bits

- 1 = Enable Transmit-On-Change
- 0 = Disable Transmit-On-Change

**Register 2: IOINTPO – Enable inputs for Transmit-On-Change**

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-1
				<b>IN3POL</b>	<b>IN2POL</b>	<b>IN1POL</b>	<b>IN0POL</b>
Bit 7							Bit 0

bit 3-0 **IN3POL:IN0POL**: Transmit-on-change Polarity bits

- 1 = Low-to-High Transition On Corresponding Input Pin Generates a transmit message
- 0 = High-to-Low Transition On Corresponding Input Generates transmit message

**Register 3: GPLAT – Output Latch Register**

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
				<b>OUT3</b>	<b>OUT2</b>	<b>OUT1</b>	<b>OUT0</b>
Bit 7							Bit 0

bit 3-0 **OUT3:OUT0**: Digital Output Latch

- 1 = Corresponding output pin is activated
- 0 = Corresponding output pin is deactivated

OUT0 can be activated through GPLAT:OUT0 only when the PWM1 is not active (T1CON:TMR1ON = 0).

**Register 4: T1CON – PWM1 Timer Control Register**

R/W-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
<b>TMR1ON</b>						<b>DC1B1</b>	<b>DC1B0</b>
Bit 7							Bit 0

bit 7 **TMR1ON**: Timer1 On bit

- 1 = Enables Timer1 for the PWM output
- 0 = Disables Timer1 and PWM1, OUT0 reflects the state of GPLAT:OUT0

bit 6-2 **Unimplemented**: Read as '0'

bit 1-0 **DC1B1:DC1B0**: Least Significant PWM1 Duty Cycle bits

**Register 5: PWM1DCH – PWM1 Duty Cycle MSB Register**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
<b>DC1B9</b>	<b>DC1B8</b>	<b>DC1B7</b>	<b>DC1B6</b>	<b>DC1B5</b>	<b>DC1B4</b>	<b>DC1B3</b>	<b>DC1B2</b>
Bit 7							Bit 0

bit 7-0 **DC1B9:DC1B2**: Most Significant PWM1 Duty Cycle bits

**Register 6: CNF1 – CAN Configuration Register 1**

R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-1
<b>SJW1</b>	<b>SJW0</b>	<b>BRP5</b>	<b>BRP4</b>	<b>BRP3</b>	<b>BRP2</b>	<b>BRP1</b>	<b>BRP0</b>
Bit 7							Bit 0

bit 7-6 **SJW1:SJW0**: Synchronized Jump Width bits

11 = Length = 4 x T<sub>Q</sub>

10 = Length = 3 x T<sub>Q</sub>

01 = Length = 2 x T<sub>Q</sub>

00 = Length = 1 x T<sub>Q</sub>

bit 5-0 **BRP5:BRP0**: Baud Rate Prescaler bits

$T_Q = 2 \times (BRP+1) / F_{osc}$

**Register 7: CNF2 – CAN Configuration Register 2**

R/W-1	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0
<b>BTLMOD</b>	<b>SAM</b>	<b>PHSEG12</b>	<b>PHSEG11</b>	<b>PHSEG10</b>	<b>PRSEG2</b>	<b>PRSEG1</b>	<b>PRSEG0</b>
Bit 7							Bit 0

bit 7 **BTLMOD**: Length determination of PHSEG2 bit

1 = Length of Phase\_Seg2 determined by bits 2:0 of CNF3

0 = Length of Phase\_Seg2 is the greater of Phase\_Seg1 or IPT(2T<sub>Q</sub>)

bit 6 **SAM**: Sample of the CAN bus line bit

1 = Bus line is sampled three times at the sample point

0 = Bus line is sampled once at the sample point

bit 5-3 **PHSEG12:PHSEG10**: Phase Buffer Segment1 bits

111 = length = 8 x T<sub>Q</sub>

000 = length = 1 x T<sub>Q</sub>

bit 2-0 **PRSEG2:PRSEG0**: Propagation Time Segment bits

111 = length = 8 x T<sub>Q</sub>

000 = length = 1 x T<sub>Q</sub>

**Register 8: CNF3 – CAN Configuration Register 3**

U-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-1	R/W-0
	<b>WAKFIL</b>				<b>PHSEG22</b>	<b>PHSEG21</b>	<b>PHSEG20</b>
Bit 7							Bit 0

bit 7 **Unimplemented**: (Reads as 0)

bit 6 **WAKFIL**: Wake-up filter bit

1 = Wake-up filter enabled

0 = Wake-up filter disabled

bit 5-3 **Unimplemented**: (Reads as 0)

bit 2-0 **PHSEG22:PHSEG20**: Phase Buffer Segment2 bits

111 = length = 8 x T<sub>Q</sub>

001 = length = 2 x T<sub>Q</sub>

000 = Invalid

**Register 9: STCON – Scheduled Transmission Control Register**

R/W-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
<b>STEN</b>		<b>STBF1</b>	<b>STBF0</b>	<b>STM3</b>	<b>STM2</b>	<b>STM1</b>	<b>STM0</b>
Bit 7							Bit 0

bit 7 **STEN:** Scheduled Transmission Enable bit

- 1 = Enabled
- 0 = Disabled

bit 6 **Unimplemented:** (Reads as 0)

bit 5-4 **STBF1:STBF0:** Base Transmission Frequency bits

- 00 = 10ms
- 01 = 160ms
- 10 = 2560ms
- 11 = 40960ms

bit 3-0 **STM3:STM0:** Scheduled Transmission Multiplier bits

- 0000 = 1
- 0001 = 2
- 1110 = 15
- 1111 = 16

**Register 10: OPTREG2 – Option Register 2**

R/W-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
<b>CAEN</b>							
Bit 7							Bit 0

bit 7 **CAEN:** Command Acknowledge Enable bit

- 1 = Enables the command acknowledge message (TXID1)
- 0 = Enables the receive overflow message (TXID1)

bit 6-0 **Unimplemented:** (Reads as 0)

**Register 11: RXMSIDH – Acceptance Filter Mask, Standard ID MSB**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
<b>SID10</b>	<b>SID9</b>	<b>SID8</b>	<b>SID7</b>	<b>SID6</b>	<b>SID5</b>	<b>SID4</b>	<b>SID3</b>
Bit 7							Bit 0

bit 7-0 **SID10:SID3:** Standard Identifier bits

**Register 12: RXMSIDL – Acceptance Filter Mask, Standard ID LSB**

R/W-0	R/W-0	R/W-0	U-0	R/W-0	U-0	R/W-0	R/W-0
<b>SID2</b>	<b>SID1</b>	<b>SID0</b>		<b>EXIDE</b>		<b>EID17</b>	<b>EID16</b>
Bit 7							Bit 0

bit 7-5 **SID2:SID0:** Standard Identifier bits

bit 4 **Unimplemented:** Read as '0'

bit 3 **EXIDE:** Extended Identifier Enable bit

- 1 = Apply filter to RXFnSIDL.EXIDE (filter applies to standard **or** extended message frames depending on filter bit)
- 0 = Do not apply filter to RXFnSIDL.EXIDE (filter will be applied to both standard **and** extended message frames)

bit 2 **Unimplemented:** Read as '0'

bit 1-0 **EID17:EID16:** Extended Identifier bits

**Register 13: RXMEID8 – Acceptance Filter Mask, Extended ID Mid**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
<b>EID15</b>	<b>EID14</b>	<b>EID13</b>	<b>EID12</b>	<b>EID11</b>	<b>EID10</b>	<b>EID9</b>	<b>EID8</b>
Bit 7							Bit 0

bit 7-0 **EID15:EID8**: Extended Identifier bits

**Register 14: RXMEID0 – Acceptance Filter Mask, Extended ID LSB**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
<b>EID7</b>	<b>EID6</b>	<b>EID5</b>	<b>EID4</b>	<b>EID3</b>	<b>EID2</b>	<b>EID1</b>	<b>EID0</b>
Bit 7							Bit 0

bit 7-3 **EID7:EID3**: Extended Identifier bits

bit 2-0 **EID2:EID0**: Extended Identifier bits (always reads as '0')

**Register 15: RXFnSIDH – Acceptance Filter n, Standard ID MSB**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
<b>SID10</b>	<b>SID9</b>	<b>SID8</b>	<b>SID7</b>	<b>SID6</b>	<b>SID5</b>	<b>SID4</b>	<b>SID3</b>
Bit 7							Bit 0

bit 7-0 **SID10:SID3**: Standard Identifier bits

**Register 16: RXFnSIDL – Acceptance Filter n, Standard ID LSB**

R/W-0	R/W-0	R/W-0	U-0	R/W-0	U-0	R/W-0	R/W-0
<b>SID2</b>	<b>SID1</b>	<b>SID0</b>		<b>EXIDE</b>		<b>EID17</b>	<b>EID16</b>
Bit 7							Bit 0

bit 7-5 **SID2:SID0**: Standard Identifier bits

bit 4 **Unimplemented**: Read as '0'

bit 3 **EXIDE**: Extended Identifier Enable bit  
 1 = Filter will apply to extended identifier  
 0 = Filter will apply to standard identifier

bit 2 **Unimplemented**: Read as '0'

bit 1-0 **EID17:EID16**: Extended Identifier bits

**Register 17: RXFnEID8 – Acceptance Filter n, Extended ID Mid**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
<b>EID15</b>	<b>EID14</b>	<b>EID13</b>	<b>EID12</b>	<b>EID11</b>	<b>EID10</b>	<b>EID9</b>	<b>EID8</b>
Bit 7							Bit 0

bit 7-0 **EID15:EID8**: Extended Identifier bits

**Register 18: RXFnEID0 – Acceptance Filter n, Extended ID LSB**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
<b>EID7</b>	<b>EID6</b>	<b>EID5</b>	<b>EID4</b>	<b>EID3</b>	<b>EID2</b>	<b>EID1</b>	<b>EID0</b>
Bit 7							Bit 0

bit 7-3 **EID7:EID0**: Extended Identifier bits (always reads as '0')

**Register 19: TXnSIDH – Transmit Message n, Standard ID MSB**

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
					<b>SID10</b>	<b>SID9</b>	<b>SID8</b>
Bit 7					Bit 0		

bit 7-3 **Unimplemented:** (Reads as 0)

bit 2-0 **SID10:SID8:** Standard Identifier bits

**Register 20: TXnSIDL – Transmit Message n, Standard ID LSB**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
<b>SID7</b>	<b>SID6</b>	<b>SID5</b>	<b>SID4</b>	<b>SID3</b>	<b>SID2</b>	<b>SID1</b>	<b>SID0</b>
Bit 7				Bit 0			

bit 7-0 **SID7:SID0:** Standard Identifier bits

**Register 21: DBGCON – Debug Output Control Register**

R/W-1	R/W-1	R/W-1	U-0	U-0	U-0	U-0	U-0
<b>DBGRX</b>	<b>DBGTX</b>	<b>DBGAC</b>					
Bit 7			Bit 0				

bit 7 **DBGRX:** Display Received Messages Enable bit

- 1 = Enables the output on the RS-232 interface on reception of a CAN message
- 0 = Disables the output on the RS-232 interface on reception of a CAN message

bit 6 **DBGTX:** Display Transmitted Messages Enable bit

- 1 = Enables the output on the RS-232 interface on transmission of a CAN message
- 0 = Disables the output on the RS-232 interface on transmission of a CAN message

bit 5 **DBGAC:** Display local I/O actions

- 1 = Enables the output on the RS-232 interface on I/O changes
- 0 = Disables the output on the RS-232 interface on I/O changes

bit 4-0 **Unimplemented:** (Reads as 0)

## Loading of New Firmware to CANLY

### Prerequisites

1. Image file with firmware in Intel-Hex format (\*.hex)
2. Download tool for PC: `flashloader.exe`
3. Serial connection.

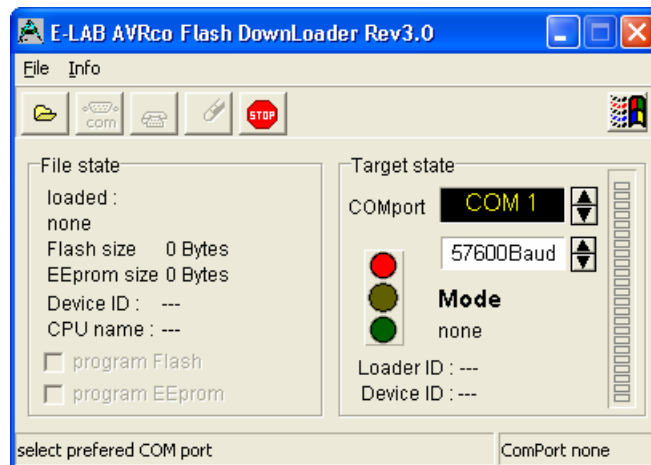
### Procedure

#### Hardware Setup

1. Connect CANLY module via serial cable to COM port of PC.
2. Apply power to the module

#### Prepare PC software

3. Download firmware file (\*.hex) from support website or receive it via email, and store it on PC.
4. Start a terminal program on the PC, set to 19'200bit/s, 8 databits, 1 stopbit, no parity.
5. Enter the command 'UD' and confirm message with 'Y' to invoke the bootloader on the CANLY module.
6. Close the terminal connection in order to release the COM port.
7. Start the download tool `flashloader.exe` on the PC.
8. Set the appropriate COM port and transmission speed to 57'600Baud:



9. Open the file selection dialog in the menu „File – Open File...“, select the firmware imagefile and select „Open“.
10. Click the button „com“. The status display on the right side should now show „Target connected“.
11. Start downloading by clicking on the “Telephone” button. The status display will now show „programming“, and the progress bar fills up. After the download is finished, the status message reads again „Target connected“.
12. Start firmware by clicking on the „Stop“ button.



#### Check for Successful Download.

4. Close the download tool
5. Open the terminal connection again and issue any command.

## Appendix

---

### Notice to Users

The intended use of the CANLY modules is described in this document. Other than the described uses are not permitted or only after consultation with the manufacturer.

CANLY modules are not authorized for use as critical components in life-support devices or systems.

Life-support devices or systems are devices or systems intended for surgical implantation into the body or to sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling and user's manual, can be reasonably expected to result in significant injury.

No complex software or hardware system is perfect. Bugs are always present in a system of any size. In order to prevent danger to life or property, it is the responsibility of the system designer to incorporate redundant protective mechanisms appropriate to the risk involved.

All CANLY modules are 100 percent functionally tested. Additional testing may include visual quality control inspections. Specifications are based on characterization of tested sample units rather than testing over temperature and voltage of each unit. CANLY modules may qualify components to operate within a range of parameters that is different from the manufacturer's recommended range.

### Revision History

Rev. 1.0, Date: 20.10.2006

Initial release.

### Contact

Elektronik-Atelier Kallen  
Steinackerweg 14  
3075 Rüfenacht

Web: [www.avrcard.com](http://www.avrcard.com)

Email: [support@avrcard.com](mailto:support@avrcard.com)

Tel: +41 31 832 1441

Fax: +41 31 832 1442