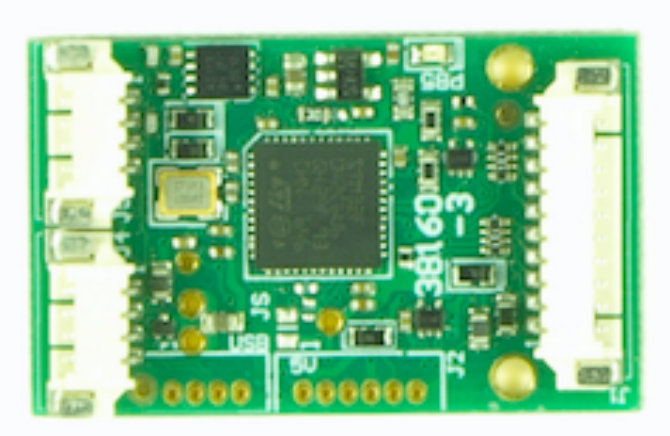
CAN2UART module

Technical Reference Manual



Preliminary - Version 1.0

May 2019

Auvidea GmbH
Kellerberg 3
D-86920 Denklingen

Tel: +49 8243 7714 622
info@auvidea.com
www.auvidea.com

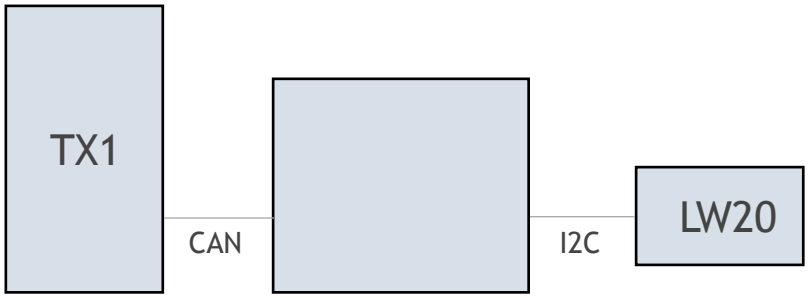
Copyright Notice

Introduction

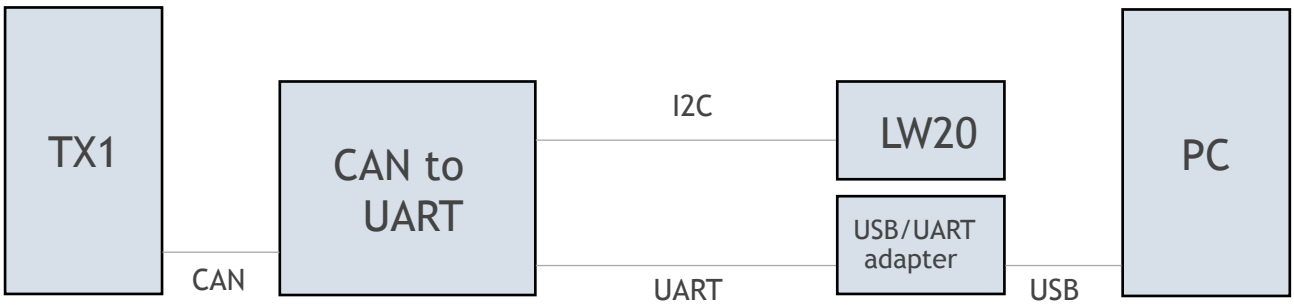
CAN2UART module

The use case of the CAN2UART module is to run the LW20 on the CAN-bus. LW20 only supports native I2C and UART.

The CAN-bus must be terminated with a point to point connection with the Jetson TX1.

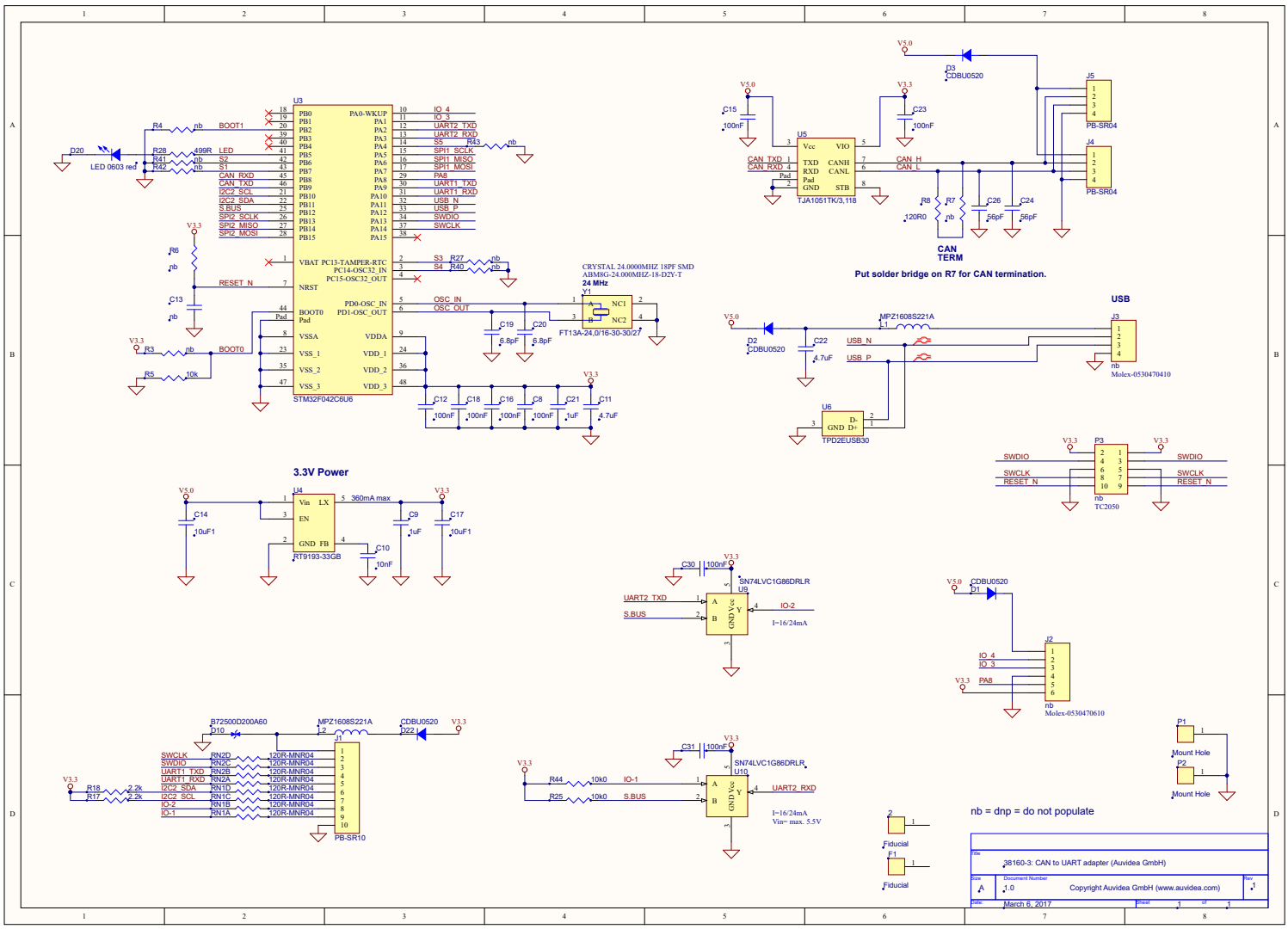


Optional for debugging over the console



Technical details

Schematics



Technical details

- micro controller: STM32F042C6U6
- CAN transceiver: TJA1051TK/3,118
- size: 18 x 27.5 mm
- order code: 38160

Firmware Releases

Release v1.0

- First release

Release v1.1

- Implemented control of basic functions via CAN messages

Release v1.2

- Changes to I2C, and alternative functioning of I2C as serial bus for the LW20

Release v1.3

- Implemented CAN to I2C protocol and list_i2c and read function for I2C via CAN, removed ADC functionality

Release v1.4

- Fixed behavior for can command messages for the LW20

Release v1.5

- Implemented I2C timeout, fix I2C write bugs

Release v1.6

- Implemented CAN addressing via jumpers, added "jumper" UART command to read jumper status

Release v1.7

- Implemented "status" function for can message, changed default loopback mode to normal

Release v1.8

- Fixed bug which prevented UART commands via CAN from working

Release v1.9

- Fixed a bug preventing can loopback mode from working sometimes

Terminal software for UART console

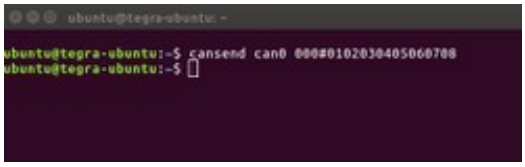
Windows 7

PuTTY 0.29_RC2

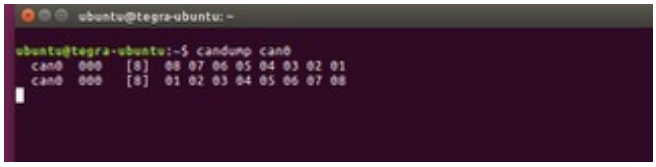
Basic Commands

By default the CAN2UART module is in the normal operation mode. You can switch to the loopback mode for debugging purposes, which will send back any CAN messages it receives as visible on the screenshot below:

Window 1:



Window 2:



You can switch to the other modes either by

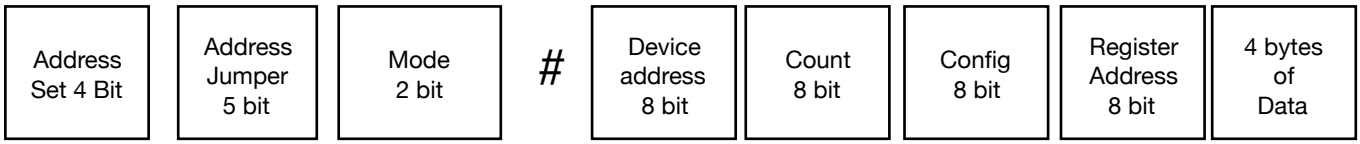
UART-Commands:

```
Canmode 0                              - for can loopback mode
Canmode 1                              - for normal operation mode (default mode)
```

(CAN is described in detail below)

CAN functionality

A message via CAN to the CAN2UART module is divided up into multiple pieces as follows:



Address set

This is a future advancement. All four bits are set to 0000 by default

Address jumper

„Address Jumper“ is the CAN address of the module, set by the jumper solder pads. There are 5 different solder pads to set the solder jumper, allowing 32 total jumper addresses, ranging from 0x00 (default without any pads soldered to 0x7C with all pads soldered.)

	S5	S4	S3	S2	S1
00000	-	-	-	-	-
00001	-	-	-	-	x
00010	-	-	-	x	-
00011	-	-	-	x	x
00100	-	-	x	-	-
00101	-	-	x	-	x
00110	-	-	x	x	-
00111	-	-	x	x	x
01000	-	x	-	-	-
01001	-	x	-	-	x
01010	-	x	-	x	-
01011	-	x	-	x	x
01100	-	x	x	-	-
01101	-	x	x	-	x
01110	-	x	x	x	-
01111	-	x	x	x	x

	S5	S4	S3	S2	S1
10000	x	-	-	-	-
10001	x	-	-	-	x
10010	x	-	-	x	-
10011	x	-	-	x	x
10100	x	-	x	-	-
10101	x	-	x	-	x
10110	x	-	x	x	-
10111	x	-	x	x	x
11000	x	x	-	-	-
11001	x	x	-	-	x
11010	x	x	-	x	-
11011	x	x	-	x	x
11100	x	x	x	-	-
11101	x	x	x	-	x
11110	x	x	x	x	-
11111	x	x	x	x	x

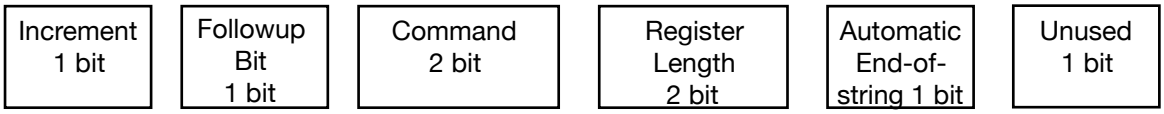
Mode

„Mode“ is 2 bits, and can be the following:

```
0x02                                    - for I2C mode
0x00                                    - for MCU commands
```

Config

In I2C mode, the config byte is divided into following bits:



Increment: not implemented yet
Followup Bit: not implemented yet

Command

```
0x00                                    - list I2C devices on UART and CAN
0x01                                    - read I2C (not implemented yet)
0x10                                    - write I2C
0x11                                    - read I2C in raw / serial mode
```

Register length: Length of the register to read/write
Automatic EOS: Automatically determines the end of a received message

There are more functions planned for the future.

The I2C address has to be entered in a 8 bit Standart, and is automatically converted to a 7 bit Standart by the microcontroller, e.g. the address „0x66“ gets converted to „0x33“ since the possible range of ID assignments is halved.

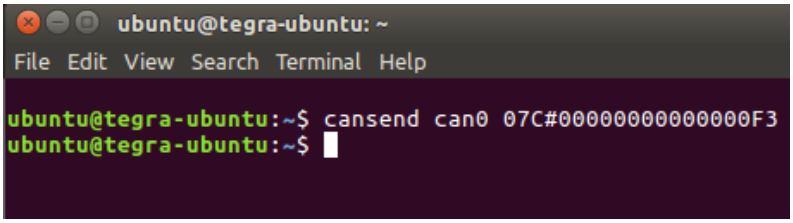
7 bit address		R/W
01100110		
0110	0110	66
011	011	33

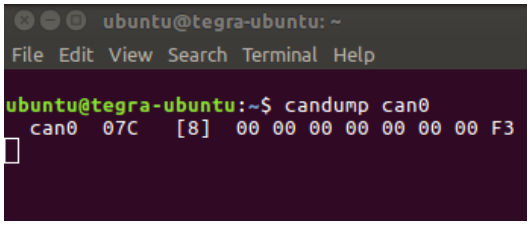
CAN functionality

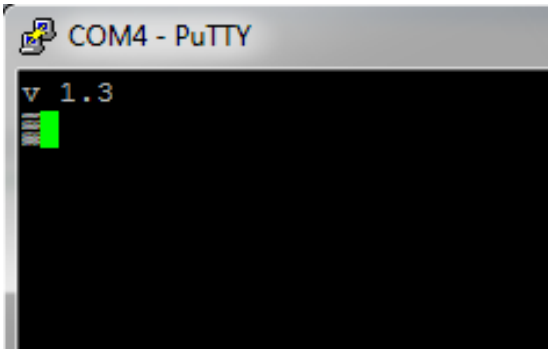
Multiple commands can be sent via can to trigger specific functions on the MCU as follows:

```
0x00000000000000F1                    - turns the LED on constantly
0x00000000000000F2                    - turns the LED off constantly
0x00000000000000F3                    - prints the current version on a console
                                        connected via UART
0x00000000000000F4                    - turns on the normal CAN operation mode
0x00000000000000F5                    - turns on the CAN loopback mode
```

result:

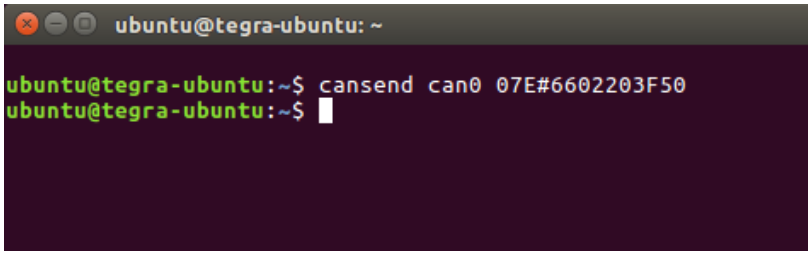






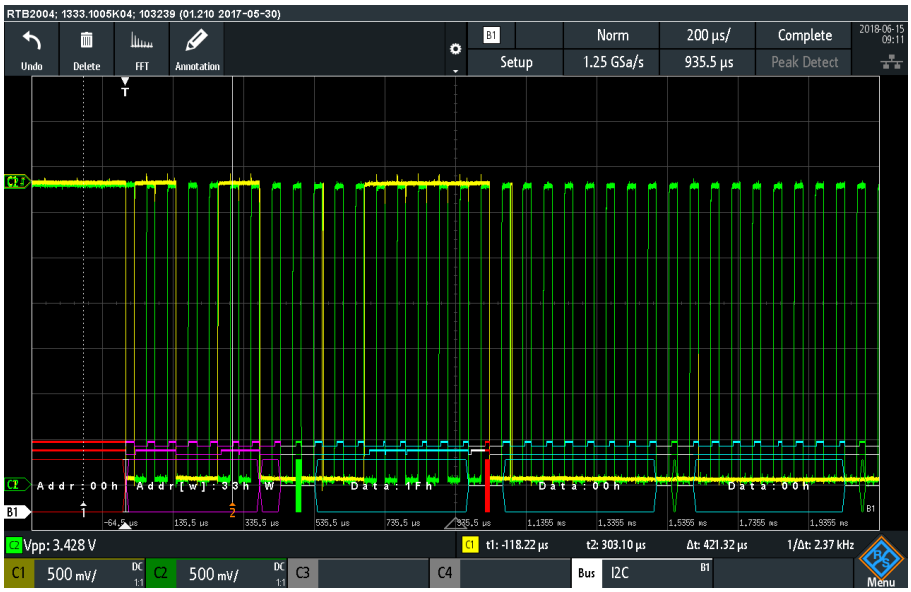
In the screenshots the address is „07C“ which means in this scenario all jumpers were soldered.

In the screenshots below a CAN message is sent to the CAN2UART module which instructs it to send a I2C message:




```
07E                                    - 0x7C for ID jumper + 0x02 for I2C mode
66                                     - 0x66 for device address
02                                     - Message count
20                                     - I2C send command
3F50                                   - Data to be sent
```

I2C Message visible on an oscilloscope



Debugging

You can send I2C messages on the CAN2UART module with the „w“ command. You can also use the „wa“ command to send ASCII chars for a serial implementation of I2C, like with the LW20.
Usage:

`w and wa` send messages on the CAN2UART module

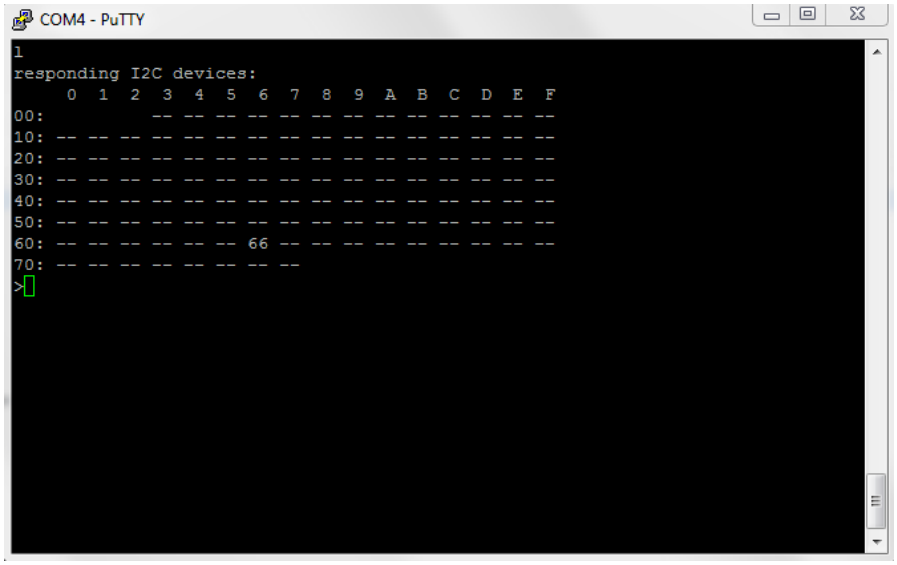
```
w [address] [register] [messages]              - e.g. „w 0x66 0x02 0xFF 0xFF“
```

(If you want to send ASCII chars to the LW20 this way, use the first char of the string instead of the address and append „0x0A 0x0D“ for carriage return and newline e.g. w 0x66 0x3F 0x50 0x0A 0x0D (sends „?P“ to the LW20))

```
wa [address] [ASCII chars to be sent]          - e.g. „wa 0x66 ?P“
```

`l` list I2C devices

```
l                                              - list I2C devices
```



`r and rr` read from I2C

```
r                                              - read specific registers
rr                                             - serial implementation of I2C on
                                                 the LW20
```

`cansend` send can message

```
cansend (can_id) (can_dlc) (can_data[1])
(can_data[2]) … (can_data[can_dlc])            - send CAN message
```

candumpa only print messages with correct CAN address

```
candumpa                                    - only print messages with correct
                                              CAN address
```

jumper prints the jumper status and the current CAN address

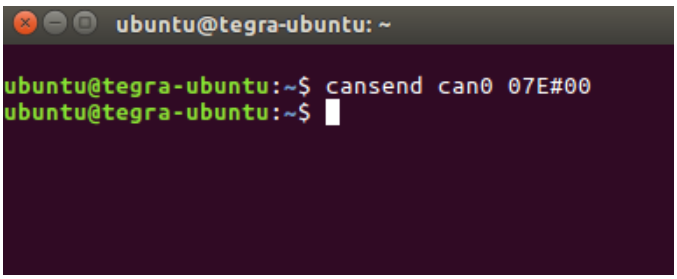
```
jumper                                      - prints the jumper status and the
                                              current CAN address
```

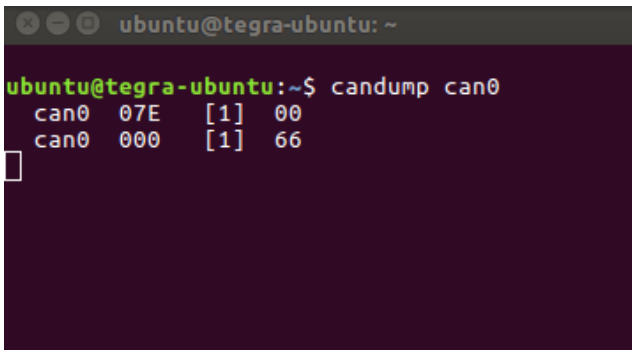
canmode (0/1) set the module in loopback mode or normal mode

```
canmode (0/1)                               - set the module in loopback mode
                                              (0) or normal mode (1)
```

Below are screenshots where I2C messages are sent and received via CAN

Example 1

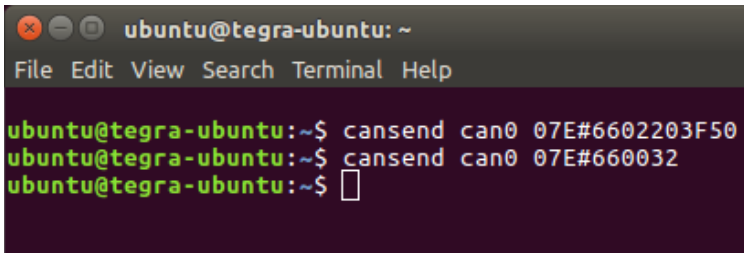


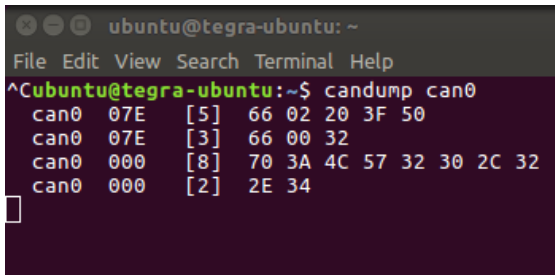



```
07E                                         - 0x07C for jumper address + 0x02 for I2C
                                              mode
```

If the config byte is empty, the CAN2UART module will automatically send a list of all connected I2C devices via CAN, in this case „0x66“, the LW20 module.

Example 2





First message, send „?P“ via I2C:
07E = 0x07C for jumper address + 0x02 for I2C mode
66 = I2C device address

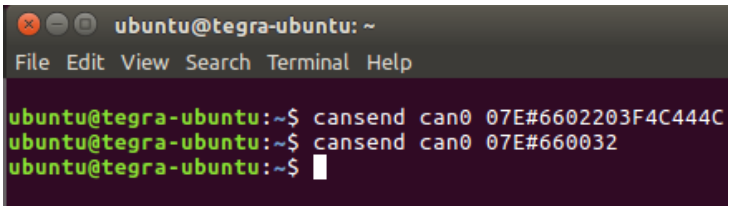
```
07E                                     – 0x07C for jumper address + 0x02 for I2C
                                          mode
66                                      – I2C device address
02                                      – Char count
20                                      – I2C write configuration
3F50                                    – hex= „?P“ ASCII
```

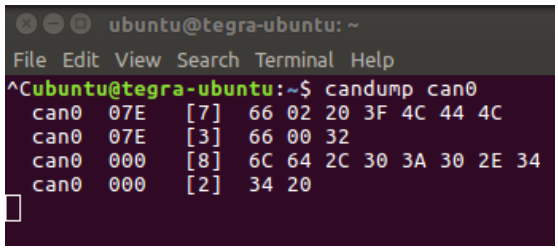
Second message, receive I2C message:

```
07E                                     – 0x07C for jumper address + 0x02 for I2C
                                          mode
66                                      – I2C device address
00                                      – count (find 0x00 bit is 1, so no manual
                                          count needed)
32                                      – 0x30 for I2C readraw + 0x02 to
                                          automatically set correct count
```

The CAN2UART module sends back „70 3A 4C 57 32 30 2C 32 2E 34“ hex, which is „p:LW20,2.4“ in ASCII.

Example 3





First message, send „?LDL“ via I2C:

```
07E                                     - 0x07C for jumper address + 0x02 for I2C
                                          mode
66                                      - I2C device address
02                                      - Char count
20                                      - I2C write configuration
3F4C444C                                - hex = „?LDL“ ASCII
```

Second message, receive I2C message:

```
07E                                     - 0x07C for jumper address + 0x02 for I2C
                                          mode
66                                      - I2C device address
00                                      - count (find 0x00 bit is 1, so no manual
                                          count needed)
32                                      - 0x30 for I2C readraw + 0x02 to
                                          automatically set correct count
```

The CAN2UART module sends back „`6C 64 2C 30 3A 30 2E 34 34 20`“ hex, which is „`ld,0:0.44`“ in ASCII.

	Register Access	Text String
I2C write	x	x
I2C read	normal	raw

Appendix

To configure CAN on the Jetson if not configured already, enter
„`sudo ip link can0 type can bitrate 1000000`“ into the console on the Jetson.
Use „`cansend can0 [address]#[values]`“ to send messages via can.
Use „`candump can0`“ to display any messages by any device on the console.

You can also use these commands on the CAN2UART module with the following commands:

```
cansend [address] [bytes to be sent] [data]
candump
```

Changelog

0.4 First release
0.5 Added I2C CAN message screenshots and description, replaced „cansend“ command with „w“ command
0.6 Changed format
1.0 Fixed instructions to accommodate the code

FAQ

to be added

Disclaimer

Thank you for reading this manual. If you have found any typos or errors in this document or any bugs or issues in the software or API, please let us know.

The Auvidea Team