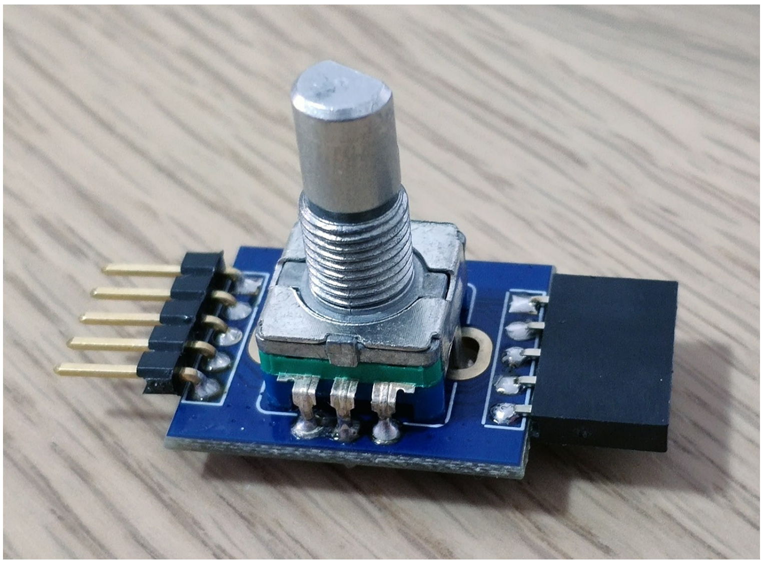


I2C EncoderMini

V1.1

Revision History

Revision	Date	Author(s)	Description
1.2	29.08.20	Simone	Bugs list updated
1.1	17.06.20	Simone	Current consumption update
1.0	22.11.19	Simone	First draft version

Contents

1. Device Overview

The I2C EncoderMini is a small board where you can connect a classical mechanical encoder on I^{2}C bus. It's possible to connect up to 127 boards in cascade and read all of them with the same I^{2}C bus.
The I2C EncoderMini has a series of 8 bit registers where it is possible to make some configuration and four 32 bit of registers.
These 32 bit registers store *counter value*, *increment steps*, *maximum* and *minimum thresholds*. Every time when encoder rotates at least one step, the *counter value* increases or decreases according to the rotation direction by the value of the *increment steps* register .
When the *counter value* is outside of the limit set by the *thresholds registers*, the counter value can be wrapped or can stuck on the threshold value reached.
It's support also the rotary encoder with a push button and it's possible to detect when it's pushed, released, double pushed or a long push.
The I2C EncoderMini also has an open-drain interrupt pin. It is set to logic low every time an interrupt occurs, the source of interrupt can be customized.

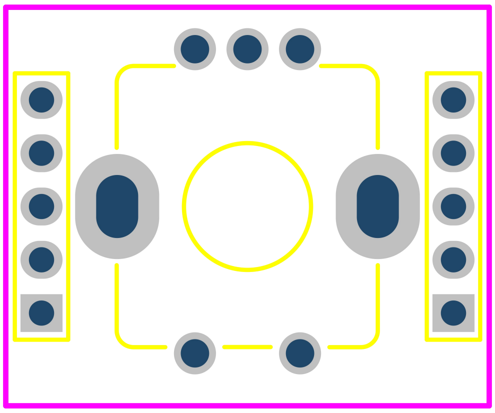


Figure 1.1: Top view of the board

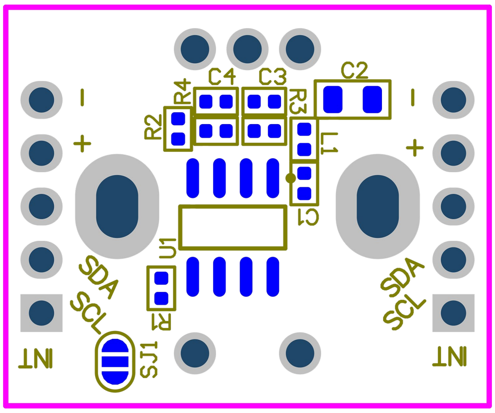


Figure 1.2: Bottom view of the board

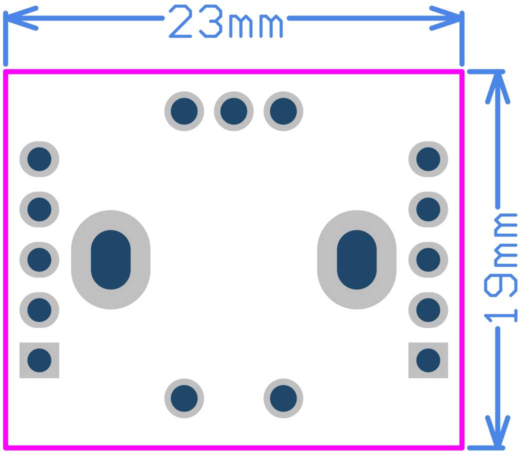


Figure 1.3: Dimensions of the board

1.1 Electrical characteristics

Parameter	Symbol	Min	Max
Supply voltage	V_{DD}	3V	5V
I^2C input-low level	V_{IL}	0	0.3 * V_{DD}
I^2C input-high level	V_{IH}	0.8 * V_{DD}	V_{DD}
I^2C clock input frequency	f_{SCL}		400kHz
Encoder frequency	f_{ENC}		100Hz
Supply current @5V	I_{DD}		5mA
Supply current @3.3V	I_{DD}		3mA
Interrupt pull-up resistor	R_{INT}	15kΩ	120kΩ

1.2 Connection

Figure 1.4 shows the pin-out of the I2C EncoderMini.

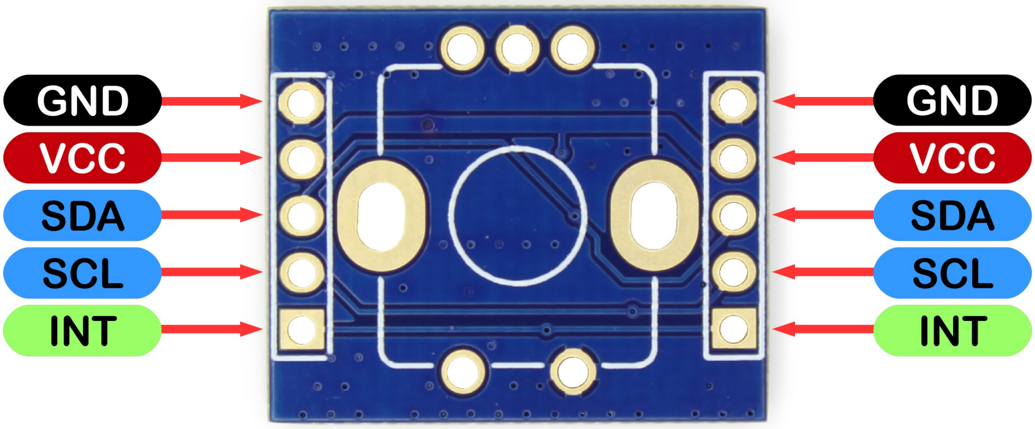


Figure 1.4: Pin-out of the board

There are two 5 pin headers on the right and left sides of the I2C EncoderMini. The pin-out i the following:

Pin	I/O Type	Function
GND	Power	Ground reference for logic
Vcc	Power	Positive supply for logic
SDA	I/O	I^2C data
SCL	I	I^2C clock
INT	OD	Open-drain interrupt output

1.3 I^2C interface

The I2C EncoderMini is a I^2C slave. The address is configurable in software by writing the register I2CADDRESS, the default address is 0x20. In order to avoid to change the address accidentally, the I2CADDRESS register should be written 3 times consecutively with the same value. If this procedure it's not followed the address change is ignored.
The I2C EncoderMini has 4.7kΩ I^2C pull-up resistors, by default they are not enabled. It's possible to enabled them by soldering th e jumper SJ1.
The I2C EncoderMini has the *auto increment* feature. This means that after writing or reading a register, the internal address pointer is automatically incremented by one. This is useful in case of reading or writing consecutive register.

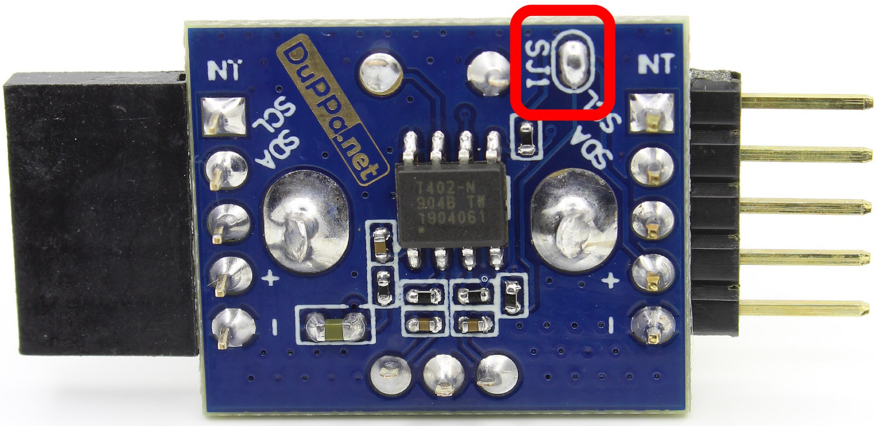


Figure 1.5: Pull-up resistors location

1.4 Rotary encoder

On the I2C EncoderMini, it is possible to solder a mechanical rotary encoder with or without dents and with any type of steps.

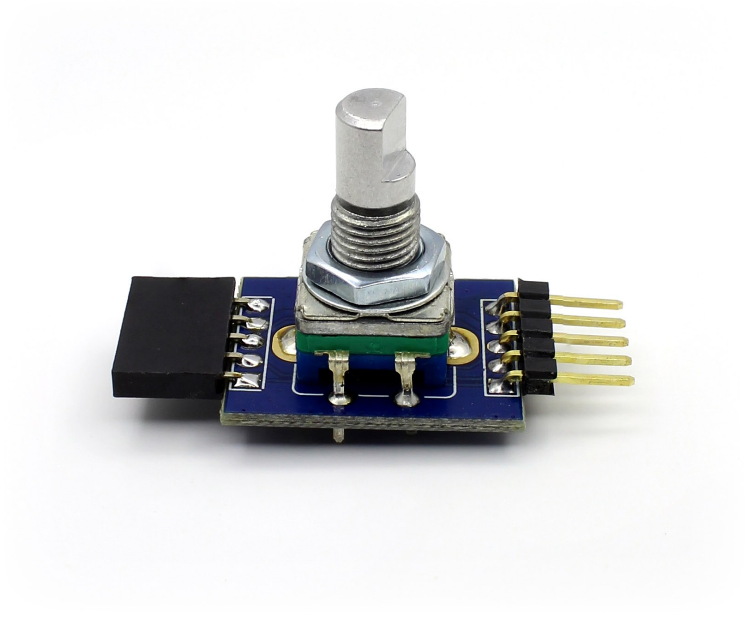


Figure 1.6: EC11 encoder with 20mm shaft

It's possible to configure the I2C EncoderMini with the **GCONF** register, it is possible to configure several parameters of the encoder.
it's possible to set the polarity of the encoder, and also the way of reading the encoder: X1, X2 and X4.
In X1 mode the counting happens only on the falling edge of the channel A, The B channel is used for the direction. Most of the encoder have this type of encoding.
In X2 mode the it's used both the falling and rising edge of the channel A, in this way the resolution is the double respect the X1. Few encoder have this type of encoding, or can be used for the encoder without dent. The last mode, the X4 mode they are used both the falling and rising edge of the channel B and A, so the total resolution is 4 times more of the X1 mode.
For reading the rotary encoder movement, there are 4 32bit registers: **CVAL**, **CMAX**, **CMIN** and **ISTEP**. All of these 4 registers work as 32bit int.
The counter limits are the following:

- **32bit INT:** from $-2.147.483.648$ to $+2.147.483.647$

It's not necessary to read all the 4 byte, you can read only the first 8 bit or the first 16 bit. For example, if you want to count between 0 and 10, you can read only the first byte of the **CVAL** register. In this way you can save I2C transactions.

Every time the encoder moves one step, the value of the **CVAL** register is increased or decreased of the value of **ISTEP**. The direction of the rotation decides if **ISTEP** is added or subtracted from **CVAL**.

CMAX and **CMIN** are used for setting a minimum and maximum thresholds of **CVAL**. In the **GCONF** register, there is **WRAPE** bit. This bit is used to enable or disable a wrap functionality of **CVAL** when it exceeds from the thresholds.

For example, if i configure the I2C EncoderMini as following:

- **CVAL**= 0
- **CMAX** = 5
- **CMIN** = -5
- **ISTEP**= 1

I will have **CVAL** is incremented of 1 at each rotation step of the encoder. The maximum value that **CVAL** can reach will be 5 while the minimum is -5. In the figure 1.7 shows the value of **CVAL**.

As showed in the figure 1.8, when **WRAPE** is set to 1 when **CVAL** reaches the value of 5, at the next increment **CVAL** it will be wrapped to -5.

Every time when the encoder is rotated one step and when **CVAL** touch the thresholds, an interrupt is generated and is possible to read in the register **ESTATUS**.

The I2C EncoderMini support also the rotary encoder with the push button. When the push button is pressed an interrupt is generated at the rising and falling edge. Inn this way, it is possible to check when the push button is pressed or released.

There is also possibility to read a fast double push by setting a window time in the register **DPPERIOD**. When a double push is made inside of the **DPPERIOD** window, an interrupt is generated.

If the **DPPERIOD** is 0, the double push function is disabled.

All the above interrupt are possible to read in the register **ESTATUS**, and can be also disabled with the register **INTCONFIG**.

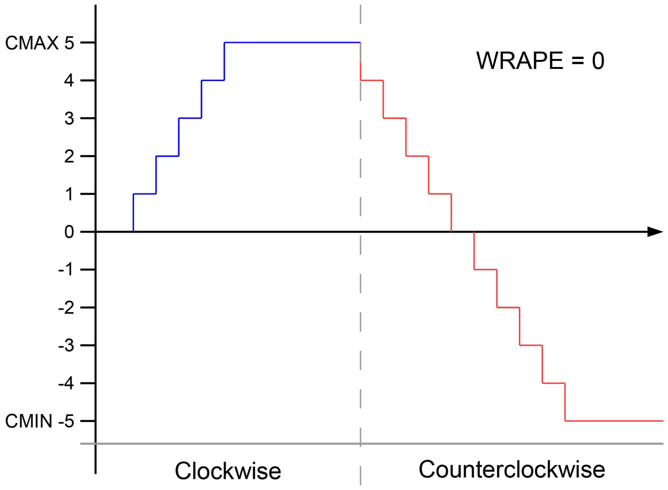


Figure 1.7: Blue and red line are the **CVAL** values when the encoder is rotate and the **WRAPE** is disabled

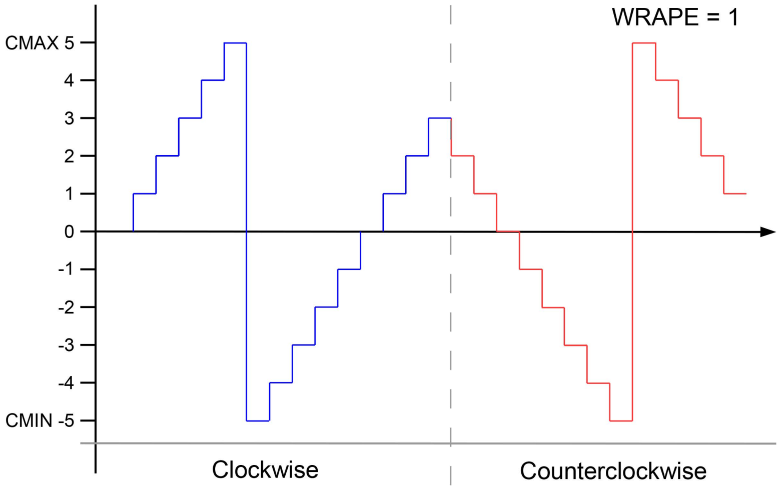


Figure 1.8: Blue and red line are the **CVAL** values when the encoder is rotate and the **WRAPE** is enabled

1.5 EEPROM

The I2C EncoderMini has 256 bytes of EEPROM.
This memory is divided in two banks of 127 bytes. The memory area is between 0x81 and 0xFF address. To use the EEPROM, user only needs to perform reading or a writing in these address areas.
The writing time takes 4 - 5ms to be executed. Wait this time before sending other commands.

1.6 Interrupt

The I2C EncoderMini has multiple interrupt source previously described. When an interrupt is generated, the **INT** pin is tied low. By reading the register **ESTATUS**, the interrupts are cleared and the **INT** pin returns high.
The INT pin is open-drain output. Hence it requires an external pull-up resistor, or internal pull-up resistor can be enabled by setting the bit **IPUD** to 1.
In a chain of I2C EncoderMini all the **INT** pins can be connected together, like the pin of the I^{2}C. When an interrupt occurs, user has to scan the boards in the chain to find who generates the interrupt.
With the register **INTCONF**, it is possible to enable or disable interrupt. When an interrupt is disabled, the corresponding bit is set, but the **INT** pin is not affected.

2. Registers

In this section, the internal registers of I2C EncoderMini is described.

Address range	Name	Description	Dimension	Default value
0x00	GCONF	General Configuration	1 Byte	0
0x01	INTCONF	INT pin Configuration	1 Byte	0
0x02	ESTATUS	Encoder Status	1 Byte	0
0x03 - 0x06	CVAL	Counter Value	4 Byte	0
0x07 - 0x0A	CMAX	Counter Max value	4 Byte	0
0x0B - 0x0E	CMIN	Counter Min value	4 Byte	0
0x0F - 0x12	ISTEP	Increment step value	4 Byte	0
0x13	DPPERIOD	Double push period	1 Byte	0
0x70	IDCODE	Unique number of the device	1 Byte	0x39
0x71	VERSION	HW and FW version	1 Byte	0x10
0x72	I2CADDRESS	Set a custom I^2Caddress	1 Byte	0
0x81 - 0xFF	EEPROM	EEPROM memory	127 Byte	0

2.1 Configuration

2.1.1 General Configuration

GCONF Address: **0x00**							
7	6	5	4	3	2	1	0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RESET	-	-	RMOD		IPUD	DIRE	WRAPE

❖ **WRAPE** Enable counter wrap.

1: Wrap enable. When the counter value reaches the **CMAX**+1, restart to the **CMIN** and vice versa

0: Wrap disable. When the counter value reaches the **CMAX** or **CMIN**, the counter stops to increasing or decreasing

❖ **DIRE** Direction of the encoder when increment.

1: Rotate left side to increase the value counter

0: Rotate right side to increase the value counter

❖ **IPUD** Interrupt Pull-UP disable.

1: Disable

0: Enable

❖ **RMOD** Reading Mode.

10: X4 mode

01: X2 mode

00: X1 mode

❖ **RST** Reset of the I2C EncoderMini

1: Reset of the I2C EncoderMini. The RESET command takes 400us to be executed.

0: No reset

2.2 Interrupt output Configuration

This register is used for enable or disable the interrupt source selectively. When an interrupt event occurs, the **INT** pin goes low and the event is stored in the status register.

INTCONF Address: **0x01**							
7	6	5	4	3	2	1	0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IRMIN	IRMAX	IRDEC	IRINC	IPUSHL	IPUSHD	IPUSHP	IPUSHR

❖ **IPUSHR** Push button release bit

1: Interrupt enabled when the push button is released.

0: Interrupt disabled

❖ **IPUSHP** Push button press bit

1: Interrupt enabled when the push button is pressed.

0: Interrupt disabled

❖ **IPUSHD** Push button double press

1: Interrupt enabled when the push button is double pressed.

0: Interrupt disabled

❖ **IPUSHL** Push button long press

1: Interrupt enabled when the push button pressed for long time.

0: Interrupt disabled

❖ **IRINC** Rotary encoder direction of increase

1: Interrupt enabled when the encoder is rotated in the direction of increase

0: Interrupt disabled

❖ **IRDEC** Rotary encoder direction of decrease

1: Interrupt enabled when the encoder is rotated in the direction of decrease

0: Interrupt disabled

❖ **IRMAX** **CVAL** reaches **CMAX** bit

1: Interrupt enabled when **CVAL** reaches **CMAX**

0: Interrupt disabled

❖ **IRMIN** **CVAL** reaches **CMIN** bit

1: Interrupt enabled when **CVAL** reaches **CMIN**

0: Interrupt disabled

2.3 Status

2.3.1 Encoder Status

This register if only readable, when is read is automatically cleared.

ESTATUS Address: **0x02**							
7	6	5	4	3	2	1	0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
RMIN	RMAX	RDEC	RINC	PUSHL	PUSHD	PUSHP	PUSHR

❑ **PUSHR** Status of the push button of the encoder

1: Push button is released

0: Push button is not released

❑ **PUSHP** Status of the push button of the encoder

1: Push button is pressed

0: Push button is not pressed

❑ **PUSHD** Status of the push button of the encoder

1: Push button is double pressed

0: Push button is not double pressed

❑ **PUSHL** Status of the push button of the encoder

1: Push button is long pressed

0: Push button is not long pressed

❑ **RINC** Rotary encoder is rotated in the increase direction

1: Encoder is rotated

0: Encoder is not rotated

❑ **RDEC** Rotary encoder is rotated in the decrease direction

1: Encoder is rotated

0: Encoder is not rotated

❑ **RMAX** Status of the counter value

1: **CVAL** reaches the **CMAX** value

0: **CVAL** is below the **CMAX** value

❑ **RMIN** Status of the counter value

1: **CVAL** reaches the **CMIN** value

0: **CVAL** is above the **CMIN** value

2.4 Encoder registers

2.4.1 Counter Value

CVAL Address: **0x03**							
31	30	29	28	27	26	25	24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CVAL BYTE 4 <31 - 24>							
Address: **0x04**							
23	22	21	20	19	18	17	16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CVAL BYTE 3 <23 - 16>							
Address: **0x05**							
15	14	13	12	11	10	9	8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CVAL BYTE 2 <15 - 8>							
Address: **0x06**							
7	6	5	4	3	2	1	0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CVAL BYTE 1 <7 - 0>							

2.4.2 Counter Max

CMAX Address: **0x07**							
31	30	29	28	27	26	25	24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CMAX BYTE 4 <31 - 24>							
Address: **0x08**							
23	22	21	20	19	18	17	16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CMAX BYTE 3 <23 - 16>							
Address: **0x09**							
15	14	13	12	11	10	9	8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CMAX BYTE 2 <15 - 8>							
Address: **0x0A**							
7	6	5	4	3	2	1	0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CMAX BYTE 1 <7 - 0>							

2.4.3 Counter Min

CMIN Address: **0x0B**							
31	30	29	28	27	26	25	24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CMIN BYTE 4 <15 - 8>							
Address: **0x0C**							
23	22	21	20	19	18	17	16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CMIN BYTE 3 <7 - 0>							
Address: **0x0D**							
15	14	13	12	11	10	9	8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CMIN BYTE 2 <15 - 8>							
Address: **0x0E**							
7	6	5	4	3	2	1	0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CMIN BYTE 1 <7 - 0>							

2.4.4 Increment step

ISTEP Address: **0x0F**							
31	30	29	28	27	26	25	24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ISTEP BYTE 4 <15 - 8>							
Address: **0x10**							
23	22	21	20	19	18	17	16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ISTEP BYTE 3 <7 - 0>							
Address: **0x11**							
15	14	13	12	11	10	9	8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ISTEP BYTE 2 <15 - 8>							
Address: **0x12**							
7	6	5	4	3	2	1	0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ISTEP BYTE 1 <7 - 0>							

2.5 Timing registers

This register are used for changing some timing parameter

2.5.1 Push button timing

This register is used for setting the double push and the long push timeout of the rotary encoder switch. The value is in ms × 10. When this register is 0 this function is disabled.

DPPERIOD Address: **0x13**							
7	6	5	4	3	2	1	0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PDPUSH <7 - 0>							

2.6 I2C EncoderMini unique code

This register contains an unique code that it's used to identify the I2C EncoderMini. This register is only readable and not writable.

IDCODE Address: **0x70**							
7	6	5	4	3	2	1	0
R-0	R-0	R-1	R-1	R-1	R-0	R-0	R-1
IDCODE = 0x39							

2.7 I2C EncoderMini version

This register contains the version of the I2C EncoderMini. This value will change in case of a new hardware or firmware release. This register is only readable and not writable.

VERSION Address: **0x71**							
7	6	5	4	3	2	1	0
R-0	R-0	R-0	R-1	R-0	R-0	R-0	R-0
Version = 0x10							

2.8 I^{2}C address

This register is used for set the I^{2}C address. The address is store in the EEPROM of the I2C EncoderMiniİn order to correctly set the address, this register must be written 3 consecutive times.

VERSION Address: **0x72**							
7	6	5	4	3	2	1	0
-	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
-	I^{2}C address <6 - 0>						

3. Reference

This project is open source, the HW and the FW as well as some example can be found on GitHub:
`https://github.com/Fattoresaimon/I2CEncoderMini`

4. Issues

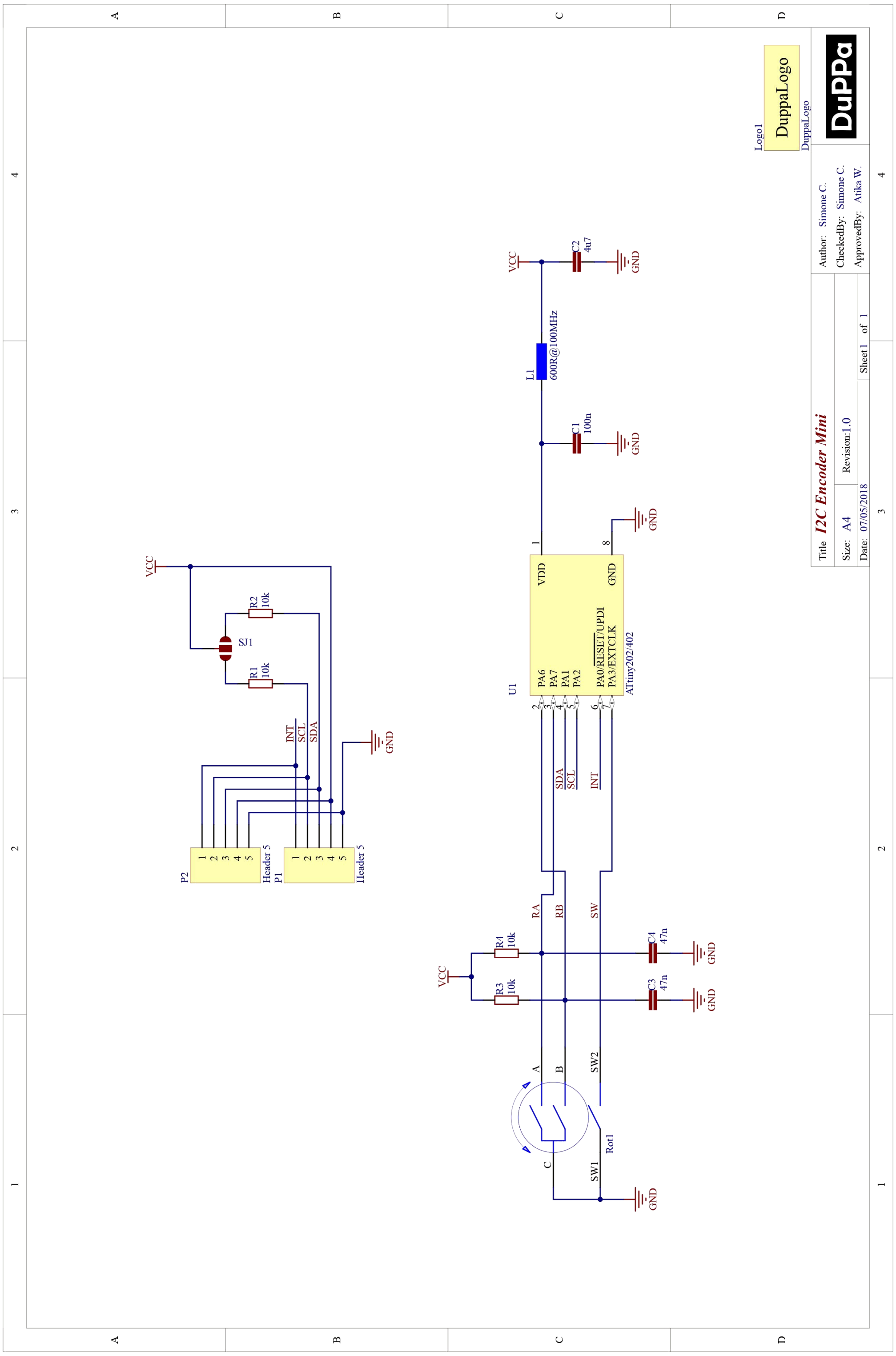
In this section, there are listed the known bugs of the I2C EncoderMini

4.1 Bug #1

Cause: The interrupt when the push button is released doesn't work if the push button is released outside of the double push period. Also it never appear if the DPPERIOD register is set to 0.

Workaround: The only way is the update the FW to the version V1.1

5. Schematic


Title I2C Encoder Mini
Size: A4
Revision:1.0
Date: 07/05/2018
Sheet1 of 1
Author: Simone C.
CheckedBy: Simone C.
ApprovedBy: Atika W.
Logo1
DuppaLogo
DuPPa
U1
ATtiny202/402
VDD
GND
PA6
PA7
PA1
PA2
PA0/RESET/UPDI
PA3/EXTCLK
VCC
GND
C1 100n
C2 4u7
L1 600R@100MHz
C3 47n
C4 47n
R1 10k
R2 10k
R3 10k
R4 10k
SJ1
P1 Header 5
P2 Header 5
INT
SCL
SDA
RA
RB
SW
Rot1
SW1
SW2
A
B
C