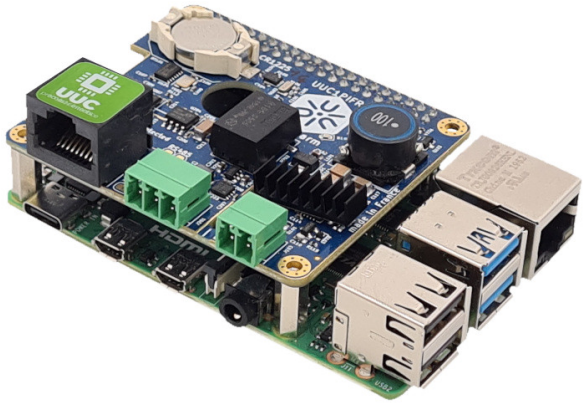
HARDWARE DATASHEET

Raspberry HAT with CAN, RS485 and Real-Time Clock

Description

UUC1PI product line is an HAT (Hardware Attached on Top) module for Raspberry Pi.
It provides a Real Time Clock, a RS485 communication, a CAN communication and in option an improved power system for the Pi.



Features

- Compatible with Raspberry Pi 3/4 (A/B)
- Real time clock
- CAN and RS-485 communication
- Compatibility with CAN Open protocol
- Optional 6.5-55V power supply

Interfaces

- CAN Fd bus up to 8 Mbds (FR-I version)
- CAN bus up to 1 Mbds (CR version)
- RS485 / RS422 interface (up to 50 Mbds) for protocols like Modbus, Profibus or DMX512...
- 1 kV isolation between power-side and interface-side (FR-I version only)

Reference	Raspberry Pi version	Input voltage	RTC	RS485	CAN	Isolated
UUC1PIFR-I	Raspberry Pi 3/4 (A/B)	6.5 V - 55 V	1	1	1 (FD)	1000V
UUC1PICR					1	-

Contents

Chapter 1

Specifications

1 Technical data

Electrical	
Power supply voltage	6.5 V - 55 V
ESD protection	30 kV
Interfaces	
CAN Fd (FR-I version)	max 8 Mbit/s
CAN (CR version)	max 1 Mbit/s
RS-485	max 50 Mbit/s
Isolation (FR-I version only)	1 kV
Physical	
Operating temperature	0°C - 85°C
Dimensions (L x W)	65.1 mm x 56.1 mm
Mounting	4 mounting holes for M2.5 screws

Figure 1.1: Technical data table

2 Electrical

UUCPI have 4 connectors.

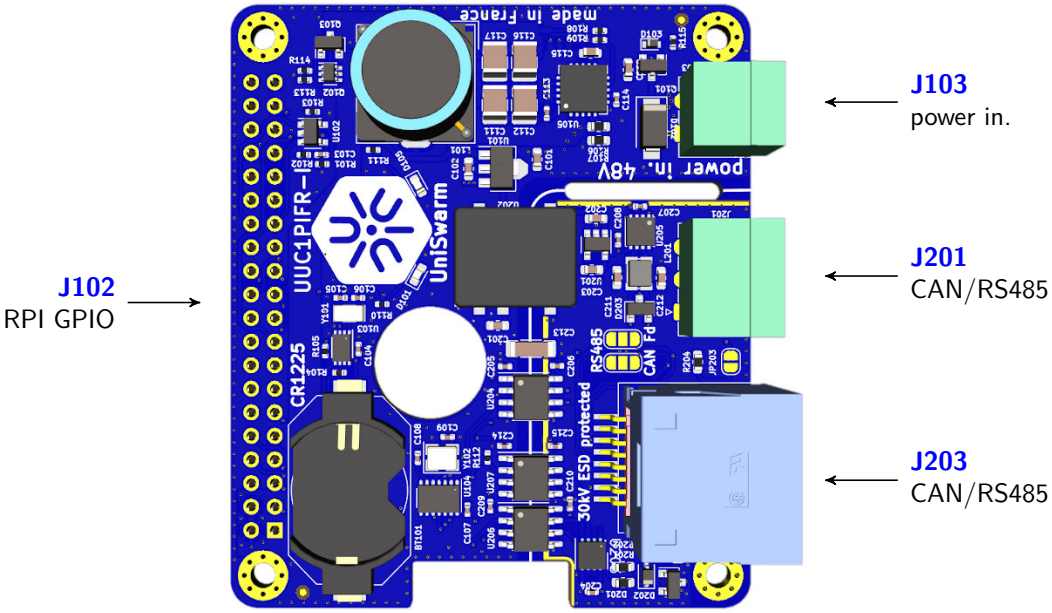


Figure 1.2: UUC1PI connectors

2.1 Power input

There are two different possibilities to power the UUC.

- From the Pi via the GPIO port : The UUC is powered by the Pi through the 5V GPIO Pin. This method can be use for every UUC, it just needs to be plug on the Pi.
- From an external power supply via the UUC external power in : The Pi is powered by the HAT through the 5V GPIO Pin. This method can only be used by UUCPI with the optional power supply. It accepts voltage between 6.5V and 55V and is protected against polarity reverse.

To know if your UUC has a power supply part you have to look if it has a J103 connector.

2.1.1 Connector J103, power input

Pins	Name	Description
1	+V	Power in +
2	GND	Ground, power in -

Figure 1.3: J103 pins

2.1.2 Recommended connector references

Screw connection :

- Phoenix® : MC 1,5/2-ST-3,5

Push-in spring connection :

- Phoenix® : FK-MCP 1,5/ 2-ST-3,5

2.2 Buses

Both buses (RS485 and CAN Fd) have 30 kV Electrostatic Discharge (ESD) protection and high quality filters for noisy environment.

A full 1kV isolation is present between bus-side and power-side to prevent damage and avoid noise from propagating through the bus.

Each bus has a 120 Ohm terminaison resistor that can be welded thanks to a dedicated jumper (JP204 for RS-485 and JP203 for CAN).

Both buses are link to the RJ45 connector and a jumper (JP201) can select which one is on the J201 connector.

Speed of both buses can be set by software. CAN Fd Bus can reach 8 Mbps. CAN Bus can reach 1 Mbps. RS-485 can reach 50 Mbps.

2.2.1 Recommended connector references

Standard straight RJ45 cable.

2.2.2 Connector J203, CAN Fd / RS485

Pins	Name	Description
1	CAN H	CAN Fd dominant
2	CAN L	CAN Fd recessive
3	GND	Ground, connected to 7
4	RS485 B	RS485 B side
5	RS485 A	RS485 A side
6	-	Unused, not connected
7	GND	Ground, connected to 3
8	-	Unused, not connected

Figure 1.4: J203 pins

2.2.3 Connector J201, CAN Fd / RS485

Pins	Name	Description
1	GND	Ground
2	CAN H/ 485A	CAN Fd dominant or RS485 A side
3	CAN L/ 485B	CAN Fd recessive or RS485 B side

Figure 1.5: J201 pins

2.2.4 Recommended connector references

Screw connection :

- Phoenix® : MC 1,5/3-ST-3,5

Push-in spring connection :

- Phoenix® : FK-MCP 1,5/ 3-ST-3,5

2.3 Raspberry Pi connection

The UUC uses the Raspberry Pi connectivity and has the J104 connector on the back of the board that must be connected with the 40 GPIO port of the Raspberry PI.

This port is compatible with Pi Zero/Zero W/ Zero WH/2B/3B/3B+/4

The UUC board uses some of the GPIO:

- One of the SPI buses is used for the communication between the Raspberry Pi and the CAN module.
- The I2C bus is used for the communication between the Raspberry Pi and an EEPROM memory.
- The UART bus is used for the communcation between the Raspberry Pi and the RS485 module.
- One GPIO is used for the led.

2.3.1 Connector J102, Raspberry PI GPIO

The following figure resumes all the GPIO used by the UUC.

Pins	Name	Description
1	3v3	Power 3,3v
2	5v	Power
3	GPIO 2	I2C1 SDA
4	5v	Power
5	GPIO 3	I2C1 SCL
6	GND	Ground
7	GPIO 4	-
8	GPIO 14	UART TX
9	GND	Ground
10	GPIO 14	UART RX
11	GPIO 17	RS485-DE
12	GPIO 18	-
13	GPIO 27	-
14	GND	Ground
15	GPIO 22	-
16	GPIO 23	-
17	3v3	Power
18	GPIO 24	-
19	GPIO 10	SPI0 MOSI
20	GND	Ground

Pins	Name	Description
21	GPIO 9	SPI0 MISO
22	GPIO 25	-
23	GPIO 11	SPI0 SCLK
24	GPIO 8	SPI0 CEO
25	GND	Ground
26	GPIO 7	-
27	GPIO 0	EEPROM SDA
28	GPIO 1	EEPROM SCL
29	GPIO 5	-
30	GND	Ground
31	GPIO 6	-
32	GPIO 12	-
33	GPIO 13	-
34	GND	Ground
35	GPIO 19	-
36	GPIO 16	-
37	GPIO 26	LED
38	GPIO 20	-
39	GND	Ground
40	GPIO 21	-

Figure 1.6: J203 pins, Raspberry Pi GPIO

More information about the Raspberry Pi GPIO can be found on the official site (https://www.raspberrypi.org/documentation/usage/gpio/)

2.4 Real Time Clock (RTC)

The UUC can provide a Real Time Clock and Calendar with the following informations :

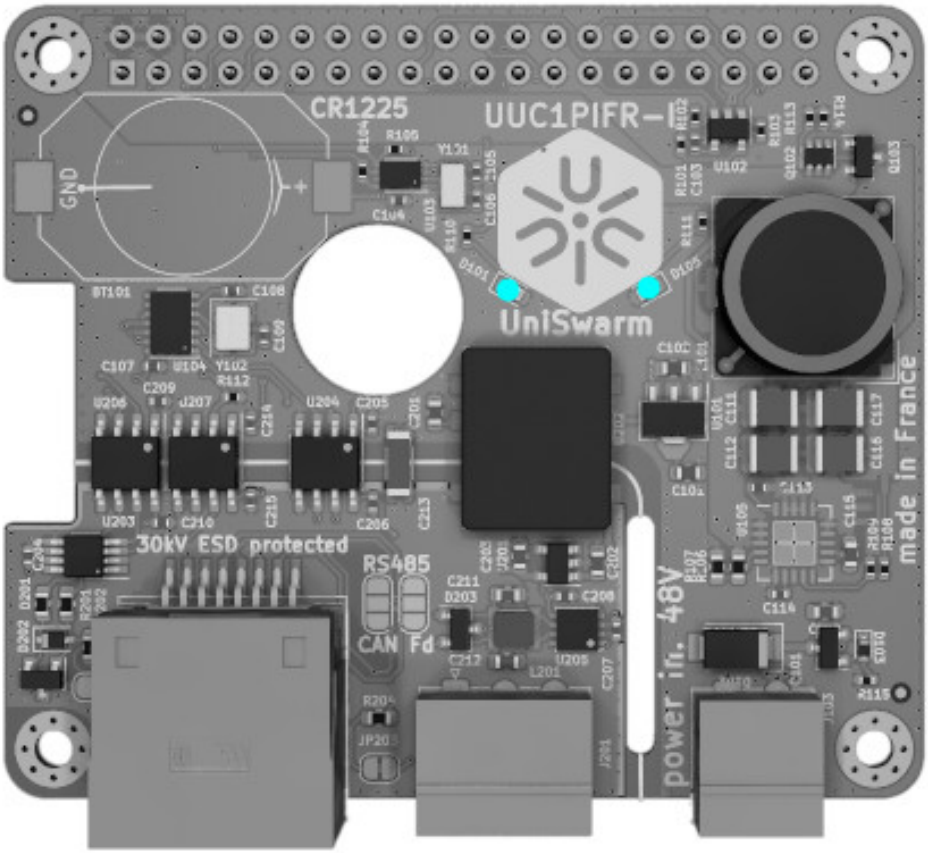
- Seconds
- Minutes
- Hours
- Day of week
- Days
- Months
- Years

The module can be configured using the I2C communication. Thanks to a 3V CR1225 battery, the RTC can keep time even in case of main power failure.

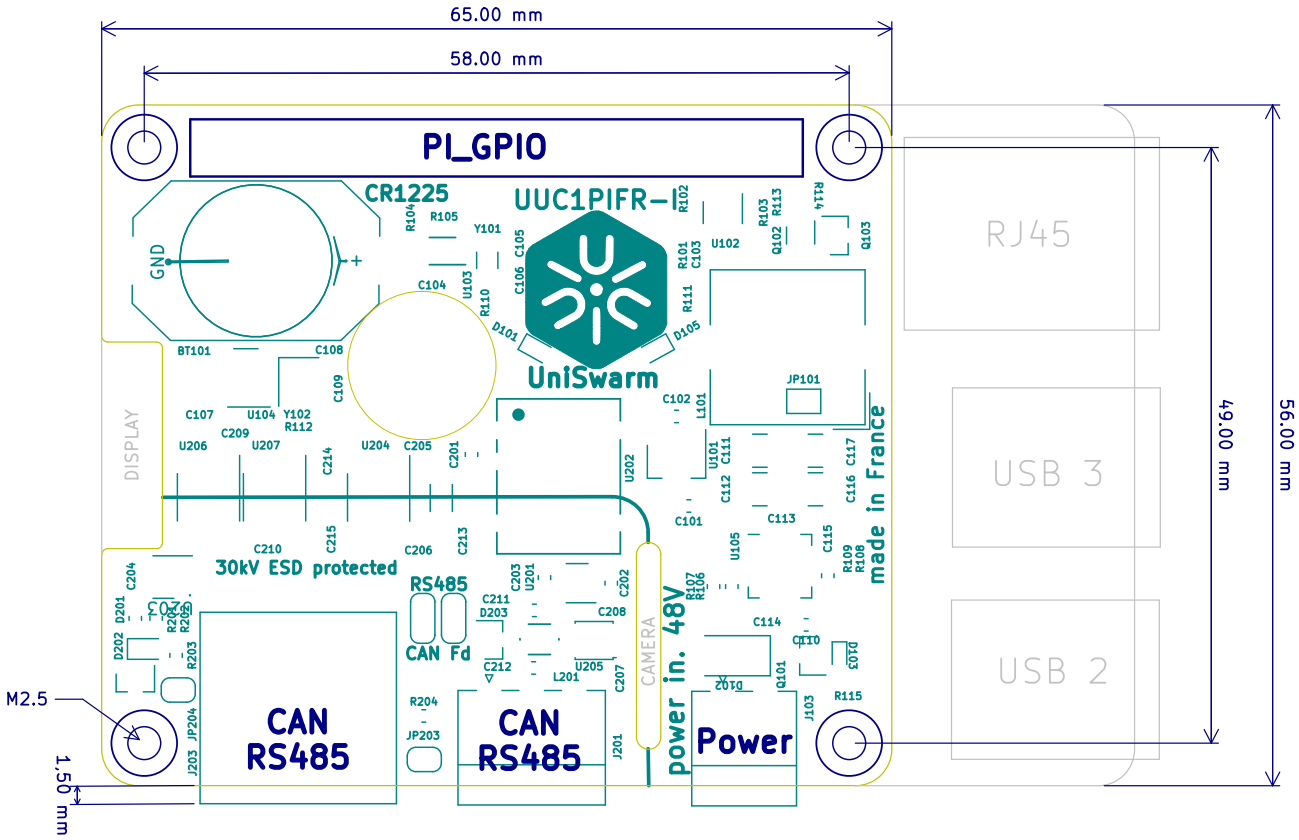
2.5 Leds

There are 2 leds on the UUC board:

- The first one indicates that the card is powered. (D105)
- The second one can be activated with the Pi GPIO. (D101)



3 Drawings



Maximum height (UUC only) : 23.00 mm
Maximum height with Pi 4 : 32.00 mm

4 Command lines

4.1 LED

This first section can allow you to verify that the connection with your device is correct by executing simple instructions that light up the LED.

For Raspberry Pi OS :

Configure LED pin as output :

```
raspi-gpio set 26 op
```

LED On :

```
raspi-gpio set 26 op pn dh
```

LED Off :

```
raspi-gpio set 26 dl
```

For Ubuntu :

Configure LED pin as output :

```
gpio -g mode 26 output
```

LED On :

```
gpio -g write 26 1
```

LED Off :

```
gpio -g write 26 0
```

4.2 CAN Bus

This section will allow you to configure the CAN and to communicate using it.
First of all be sure to write this line :

```
sudo apt install can-utils
```

If you use an ubuntu OS, add :

```
sudo apt install wiring-pi
```

4.2.1 Enable CAN

You need to write the following lines to enable CAN (according to the OS used).

For Raspberry Pi OS :

```
printf "\n # CAN Fd SPI (MCP2518Fd) \ndtoverlay=mcp251xfd,spi0-0,interrupt=25,oscillator=40000000\n" |
sudo tee -a /boot/config.txt
```

Then, you need to reboot your device by writing the following line :

```
sudo reboot
```

For Ubuntu :

```
printf "\n # CAN Fd SPI (MCP2518Fd) \ndtoverlay=mcp251xfd,spi0-0,interrupt=25,oscillator=40000000\n" |
sudo tee -a /boot/firmware/config.txt
```

Then, you need to reboot your device by writing the following line :

```
sudo reboot
```

4.2.2 Configure CAN

Now you are able to configurate your CAN and use its functions :

Set baud rate

```
sudo ip link set can0 type can bitrate 1000000
```

Enable/Disable CAN

```
sudo ip link set can0 up/down
```

Communicate with the CAN

```
candump can0
cansend can0 080#ABCD1234
```

4.3 Enable I2C

This section will allow you to configure and to enable the I2C.
First of all be sure to write this line :

```
sudo apt install i2c-tools
```

If you use an Ubuntu OS, add :

```
sudo apt install raspi-config
```

Now write the following line to enable the I2C on your Pi:

```
sudo raspi-config nonint do_i2c 0
```

Then, you need to reboot your device by writing the following line :

```
sudo reboot
```

You can check if the I2C is enabled:

```
lsmod
```

You can also check the operation of the I2C and its buses

```
i2cdetect -F 1
i2cdetect -y 1
```

If I2C is enabled, the terminal echoes an i2c-bcm2708 device. Else you can also add it manually.

```
sudo nano /etc/modules
```

write:

```
i2c-bcm2708 i2c-dev
```

4.4 Real Time Clock (RTC)

This section will allow you to enable VBAT and to access to the Hardware clock. The i2c must be enable. (See the Enable I2C section)

Start oscillator:

```
i2cset -y 1 0x6f 0 0x80
```

Enable VBAT:

```
i2cset -y 1 0x6f 3 0x08
```

Hardware clock:
You need to write these lines to access to the hardware clock (according to the OS used). (Take care, you will not be able to enable Vbat or start oscillator after writting this line)

For Raspberry Pi OS :

```
printf "\n# RTC I2C (MCP7940NT) \ndtoverlay=i2c-rtc,mcp7940x\n" | sudo tee -a /boot/config.txt
```

And now you are able to use it with :

```
sudo hwclock -r --verbose
```

For Ubuntu :

```
printf "\n# RTC I2C (MCP7940NT) \ndtoverlay=i2c-rtc,mcp7940x\n" | sudo tee -a /boot/firmware/config.txt
```

And now you are able to use it with :

```
sudo hwclock -r --verbose
```

4.5 RS485

This section will allow you to configure the RS485 and to communicate using it.

4.5.1 Enable RS485

You need to write these lines to enable RS485 communication (according to the OS used).

For Raspberry Pi OS :

```
printf "\n # RS485 \ndtoverlay=sc16is752-spi1,int_pin=24\n" | sudo tee -a /boot/config.txt
```

Then, you need to reboot your device by writing the following line :

```
sudo reboot
```

For Ubuntu :

```
printf "\n # RS485 \ndtoverlay=sc16is752-spi1,int_pin=24\n" | sudo tee -a /boot/firmware/config.txt
```

Then, you need to reboot your device by writing the following line :

```
sudo reboot
```

4.5.2 Configure RS485

After rebooting, the driver of SC16IS752 will be loaded into the system kernel. You can run command ls /dev to check the following devices:

```
gpiochip3
ttySC0
ttySC1
```

4.6 UDTStudio

Install QT5 charts for UDTStudio :

```
sudo apt install git make g++ qtbase5-dev libqt5charts5-dev
```

Build :

```
git clone https://github.com/UniSwarm/UDTStudio.git –recursive
cd UDTStudio/
mkdir build
cd build/
qmake ../src/
make -j4
```

Note: Do not forget to init and update the submodule before building it or cloning it recursively.

UDTStudio uses an environment variable called EDS_PATH to read all EDS_files.
You can export them with the following line :

```
export EDS_PATH="pathToEdsRepertory"
```

Appendix A

Hardware revision history

Version	Date	Change
v1.0.1	2020/08/10	Initial public version
v1.0.2	2021/03/20	Improved DC/DC performance Removed border line mask

Appendix B

Datasheet revision history

Revision	Date	Change
A	2020/10/15	Initial public revision
B	2022/08/31	Improved Installation part Added CR board variant Added command lines for Ubuntu and Raspberry PI OS