Toxic Detection PCB - Schematic design notes

Revision

Revision	Date	Author	Comment
v4.0	2026 Feb 12	Paul Capgras	First draft : Pre-Layout

Related Documents

- Product Brief.
- All components datasheets, reference manuals, evaluation boards available in `doc/components/`.
- Alphasense sensors' datasheets.

Purpose

The purpose of *Toxic Detection PCB - Schematic design notes* is to document the design choices and provide details on the project.

Table of Content

1. Features
2. Architecture
3. Component choice
4. Schematic
    1. Microcontroller - STM32H755ZIT3
        1. STM32 - Power Supply
        2. STM32 - Booting and programming
        3. STM32 - Clocks
        4. STM32 - Debug
        5. STM32 - Flash
    2. Analog Front End
    3. LDOs
    4. Power Gating
    5. WiFi module - ESP32-WROOM-32E-N16
        1. ESP32 - Booting and programming
        2. ESP32 - Debug
        3. STM32 - ESP32 communication
        4. ESP32 - Pin assignments
    6. GNSS module and antenna - SAM-M10Q
    7. MicroSD card
    8. Debug Features
    9. Shematic Revision 4.0
5. Layout

---

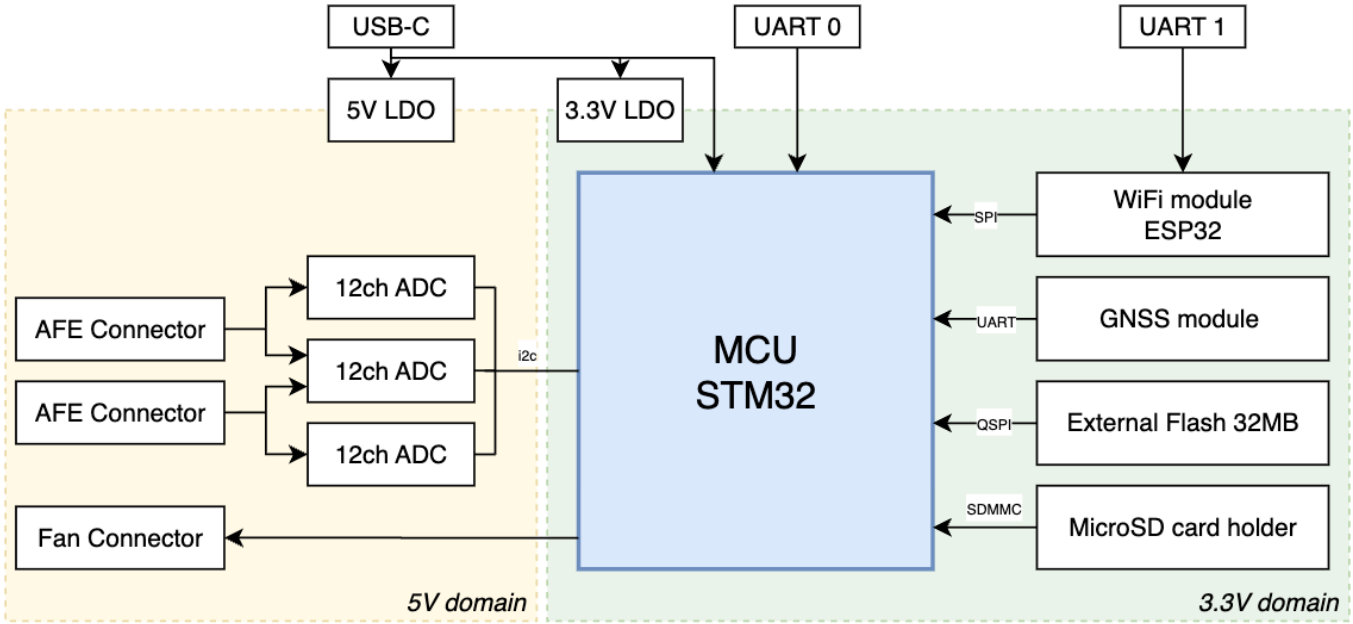
1. Features

- Toxic-Detection-PCB supports 2 sets of 4 chemical sensors from Alphasense.
- Toxic-Detection-PCB integrates an high performance microcontroller (STM32H755).
- Toxic-Detection-PCB can communicate via WiFi, Bluetooth v4.2 and BLE (ESP32).
- Toxic-Detection-PCB supports concurrent reception of four GNSS (GPS, GLONASS, Galileo, and BeiDou).
- Toxic-Detection-PCB is powered through USB-C connector, or via 5V UARTs.
- Toxic-Detection-PCB is reprogrammable: ESP32 is programmable with UART, STM32 is programmable with UART and USB 2.0. Note: Programming thanks to usb can be disable.
- Toxic-Detection-PCB provide power for a 5V-fan.
- STM32 and ESP32 can be debugged via JTAG.
- STM32 have access to a 2MB internal flash, 32MB external flash (QSPI), MicroSD card (SDMMC 4 lines).
- Toxic-Detection-PCB embedded protection against short ciruits (1A fuse), return currents, ESD (on UARTs, USB, SD card).
- Toxic-Detection-PCB can turnoff power for each Analog Front End and for the fan.
- Toxic-Detection-PCB includes 1 red and 2 green LEDs. The red led indicates the board is powered and there are no short circuit detected. The 1rst green led is controlled by the ESP32, the 2nd by the STM32. It is up to the user to define their meaning.
- Unused GPIOs of STM32, ESP32 and channels of ADCs are routed to headers.

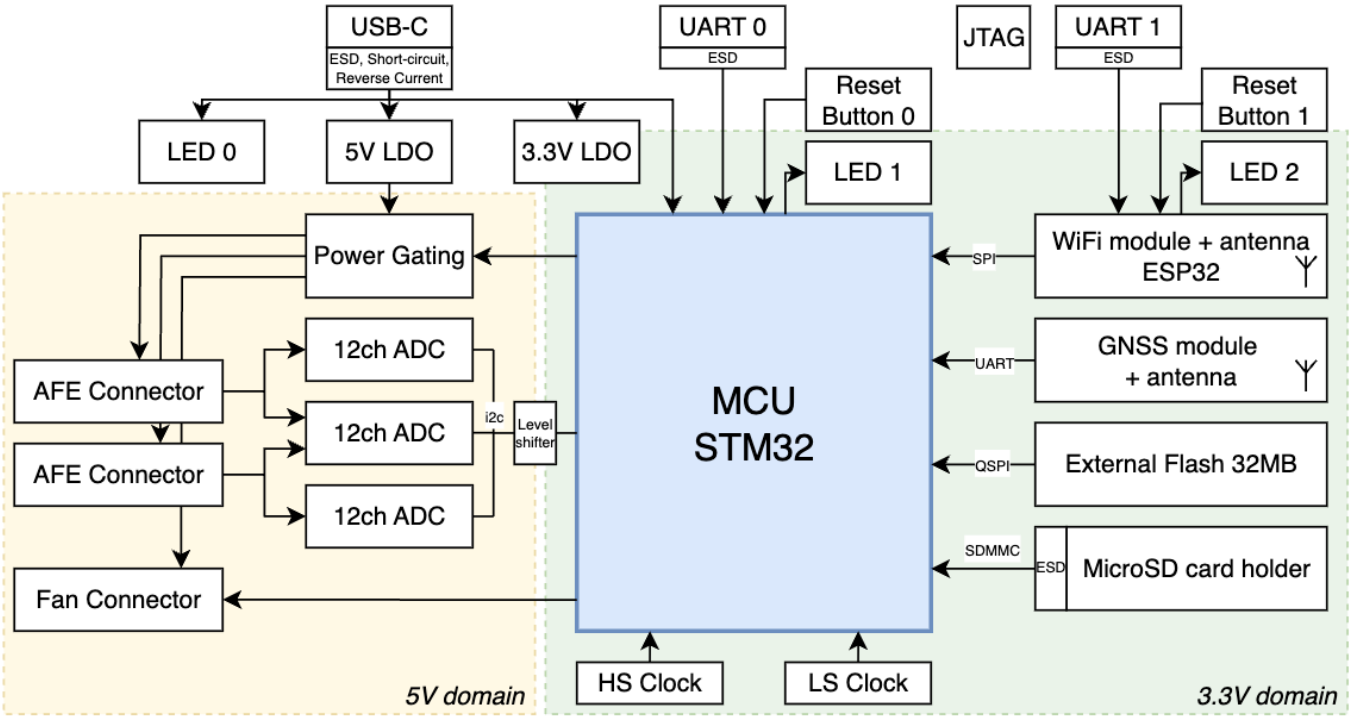
---

2. Architecture

Simplified architecture diagram:



Complete architecture diagram:



---

3. Component choice

Many components are based on the previous board design: Revision 3.3 (April 2024) and Revision 3.3.1 (August 2024).

- MCU: STM32H755ZIT3 is chosen as the Machine Learning applications have been tested on the Nucleo-h755zi development board which contains the STM32H755ZIT3.
- WIFI: ESP32-WROOM-32E is picked because it is easy to use.
- GNSS: SAM-M10Q is selected because it includes both GNSS module and antenna.

---

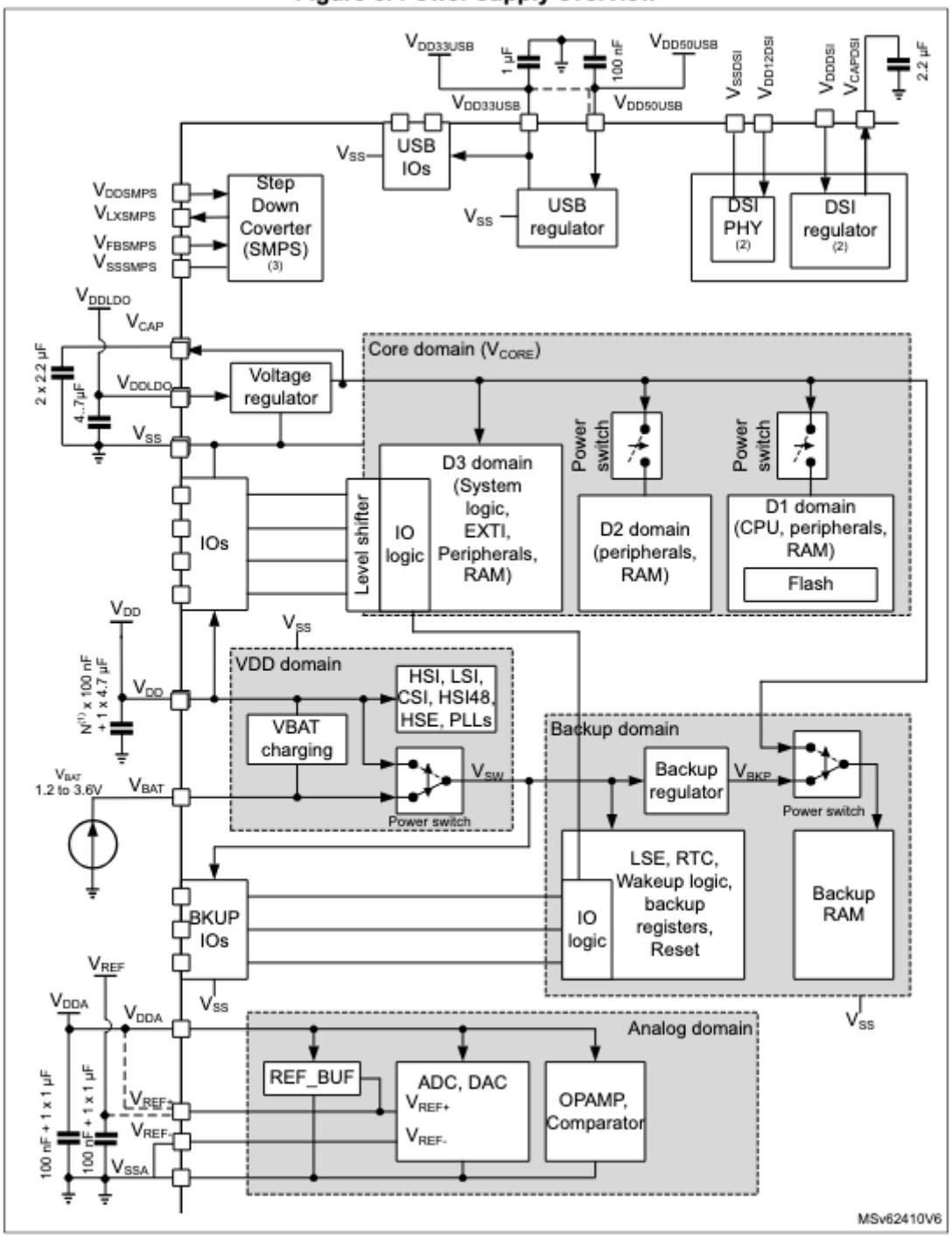
4. Schematic

4.1 Microcontroller - STM32H755ZIT3

4.1.1 STM32 - Power supply

According to the STM 32 power supply figure:

Figure 3. Power supply overview

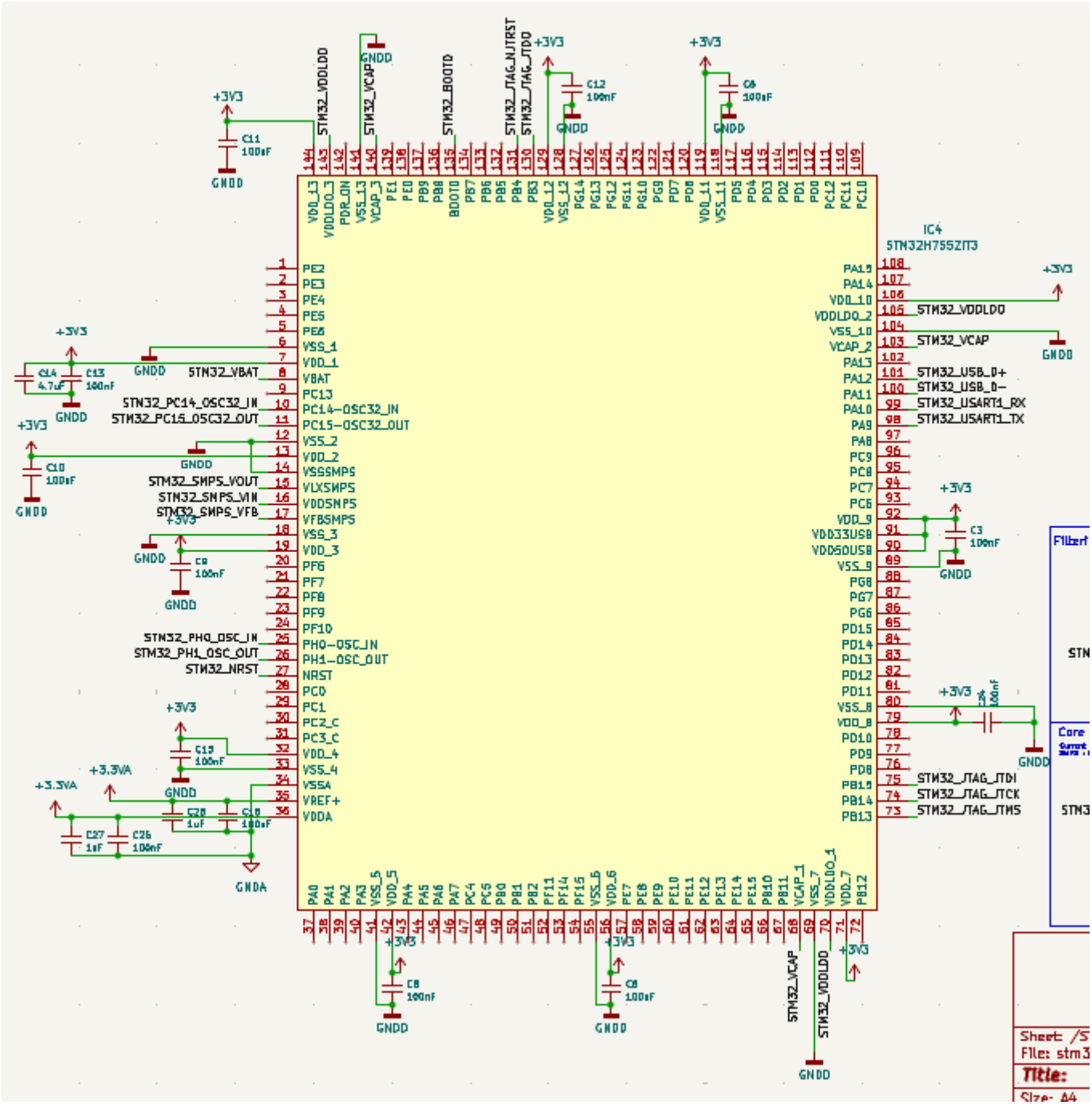


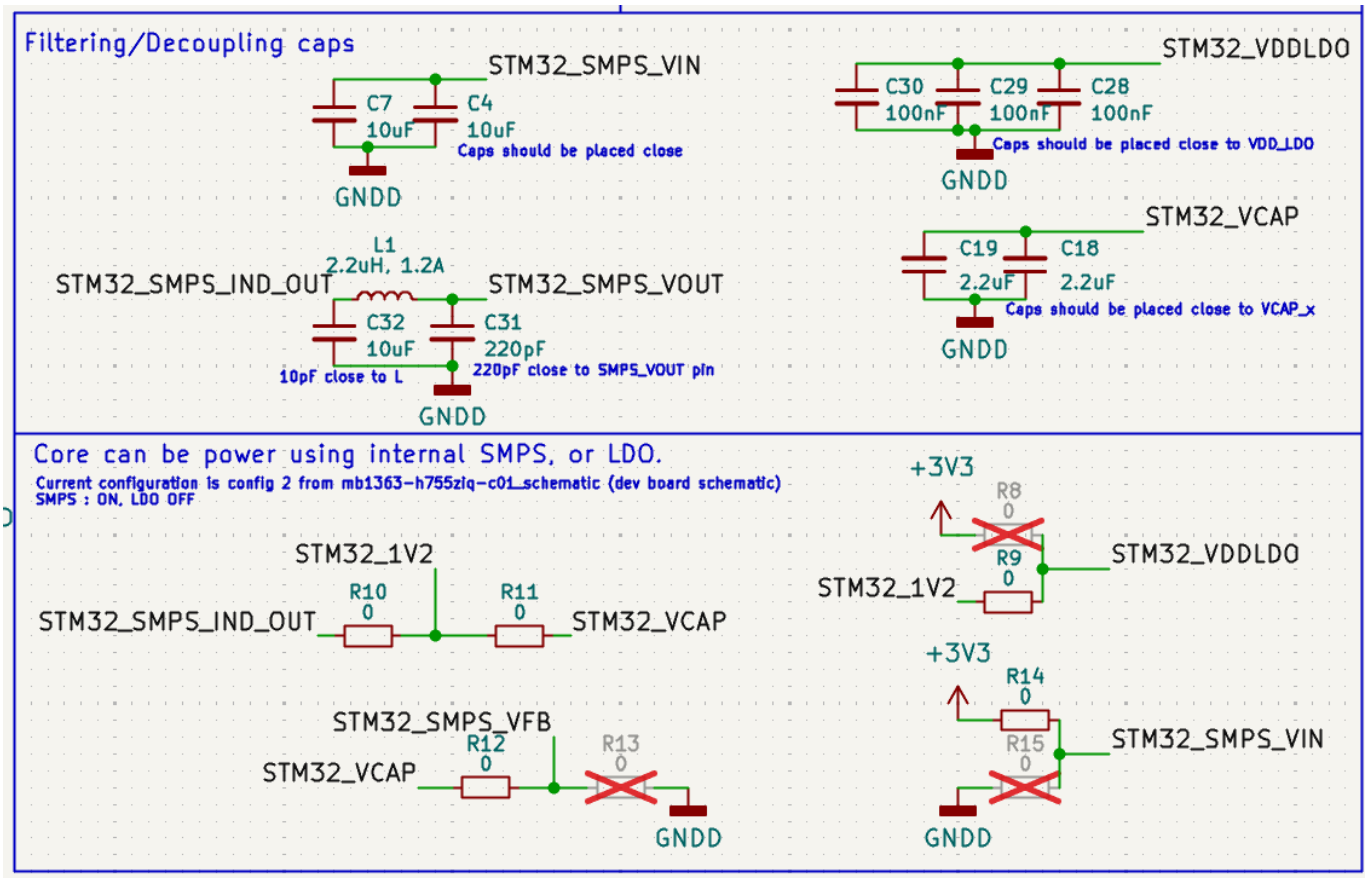
1. N corresponds to the number of V_{DD} pins available on the package.
2. The internal DSI regulator and PHY are available only on STM32H7x7I/G microcontrollers.
3. The SMPS is available only on STM32H7x5I/G and STM32H7x7I/G microcontrollers.

There are multiple key power domain:

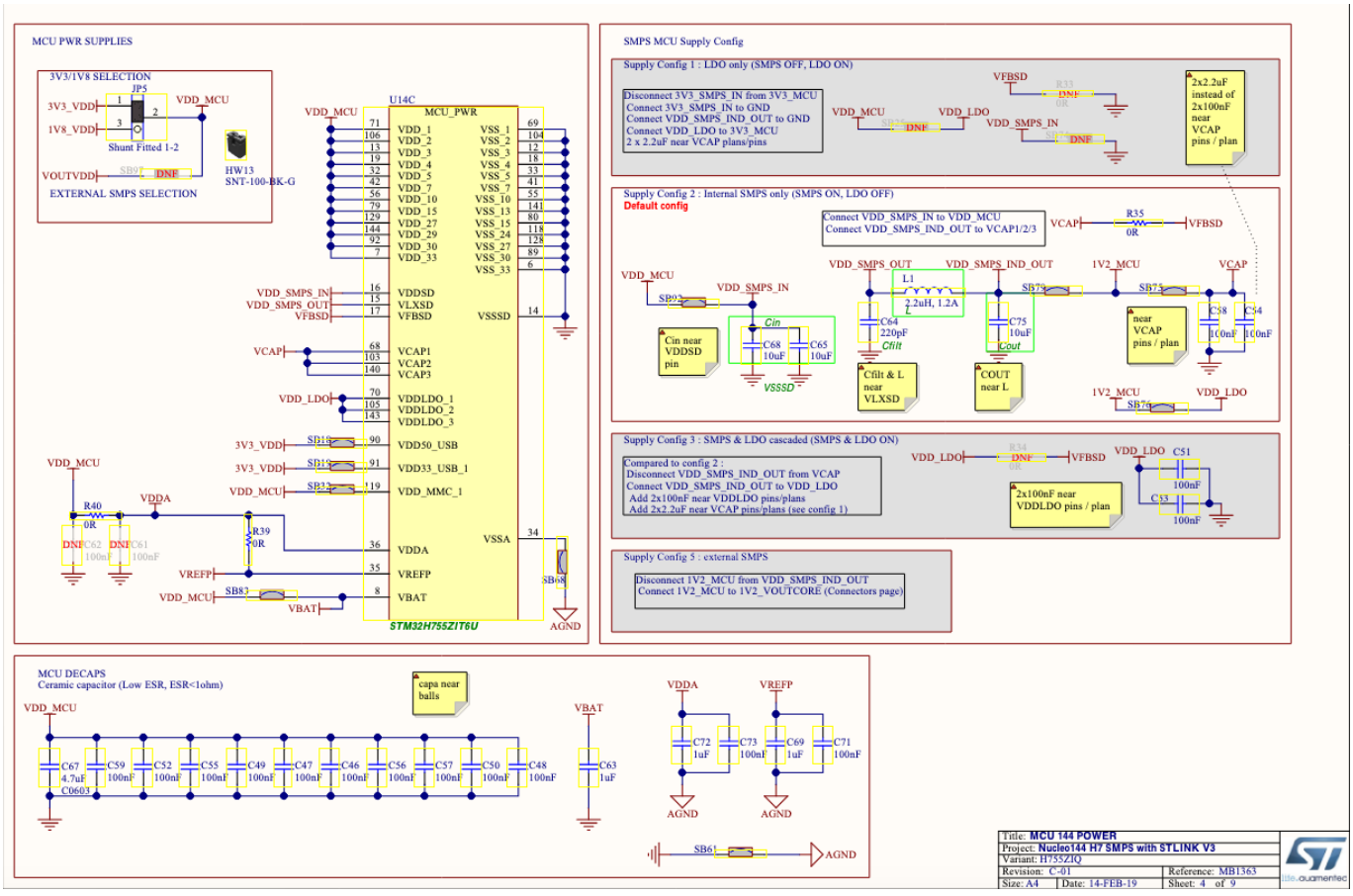
1. VDD domain (clocks, Batterie charging, IOs ring) : It is powered by VDD (3.3V).
2. Analog domain : it is powered by (3.3VA).
3. Core domain : it can be powered by internal SMPS or internal LDO. Default configuration is SMPS (3.3V input, Vcore at 1.2V).
4. USB power domain: not used in this board.

Schematic:


IC4
STM32H755ZIT3
STM32_VDDLDO
STM32_VCAP
STM32_BOOT0
STM32_JTAG_NJTRST
STM32_JTAG_JTDO
STM32_VBAT
STM32_PC14_OSC32_IN
STM32_PC15_OSC32_OUT
STM32_SMPS_VOUT
STM32_SMPS_VIN
STM32_SMPS_VFB
STM32_PH0_OSC_IN
STM32_PH1_OSC_OUT
STM32_NRST
STM32_USB_D+
STM32_USB_D-
STM32_USART1_RX
STM32_USART1_TX
STM32_JTAG_JTDI
STM32_JTAG_JTCK
STM32_JTAG_JTMS
+3V3
+3.3VA
GNDD
GNDA
Title:



Reference:



4.1.2 STM32 - Booting and programming

This section is based on AN2606.

Series	Device	Supported serial peripherals	Bootloader ID		Bootloader (protocol) version
			ID	Memory location	
H5	STM32H503xx	USART1/USART2/USART3 I2C2 I3C1 SPI1/SPI2/SPI3 USB DFU FDCAN1	0xE2	0x0BF8FFFE	USART (V4.0) I2C (V2.0) I3C (V1.0) SPI (V2.0) USB (V3.0) FDCAN (V2.0)
	STM32H562xx/63xx/73xx	USART1/USART2/USART3 I2C3/I2C4 I3C1 SPI1/SPI2/SPI3 USB DFU FDCAN2	0xE4	0x0BF9FAFE	USART (V4.0) I2C (V2.0) I3C (V1.0) SPI (V2.0) USB (V3.0) FDCAN (V2.0)
	STM32H523xx/33xx	USART1/USART2/USART3 I2C1/I2C3 I3C1 SPI1/SPI2/SPI3 USB DFU FDCAN2	0xE2	0x0BF8FFFE	USART (V4.0) I2C (V2.0) I3C (V1.0) SPI (V2.0) USB (V3.0) FDCAN (V2.0)
H7	STM32H72xxx/73xxx	USART1/USART2/USART3 I2C1/I2C2/I2C3 DFU (USB device FS) SPI1/SPI2/SPI3/SPI4 FDCAN1	0x93	0x1FF1E7FE	USART (V3.1) I2C (V1.2) DFU (V2.2) SPI (V1.1) FDCAN (V1.1)
	STM32H74xxx/75xxx	USART1/USART2/USART3 I2C1/I2C2/I2C3 DFU (USB device FS) SPI1/SPI2/SPI3/SPI4 FDCAN1	0x92	0x1FF1E7FE	USART (V3.1) I2C (V1.1) DFU (V2.2) SPI (V1.1) FDCAN (V1.1)
	STM32H7A3xx/7B3xx/7B0xx	USART1/USART2/USART3 I2C1/I2C2/I2C3 DFU (USB device FS) SPI1/SPI2/SPI3 FDCAN1	0x92	0x1FF13FFE	USART (V3.1) I2C (V1.2) DFU (V2.2) SPI (V1.2) FDCAN (V1.1)

It is should be possible to boot with UART, I2C, USB, SPI. Let's setup UART and USB for this board.

1. USB:

DFU	USB	Enabled	USB FS configured in forced device mode. USB FS interrupt vector is enabled and used for USB DFU communications.
	USB_DM pin	Input/output	PA11: USB DM line. Used in alternate push-pull, no pull mode.
	USB_DP pin		PA12: USB DP line. Used in alternate push-pull, no pull mode. No external pull-up resistor is required

2. UART:

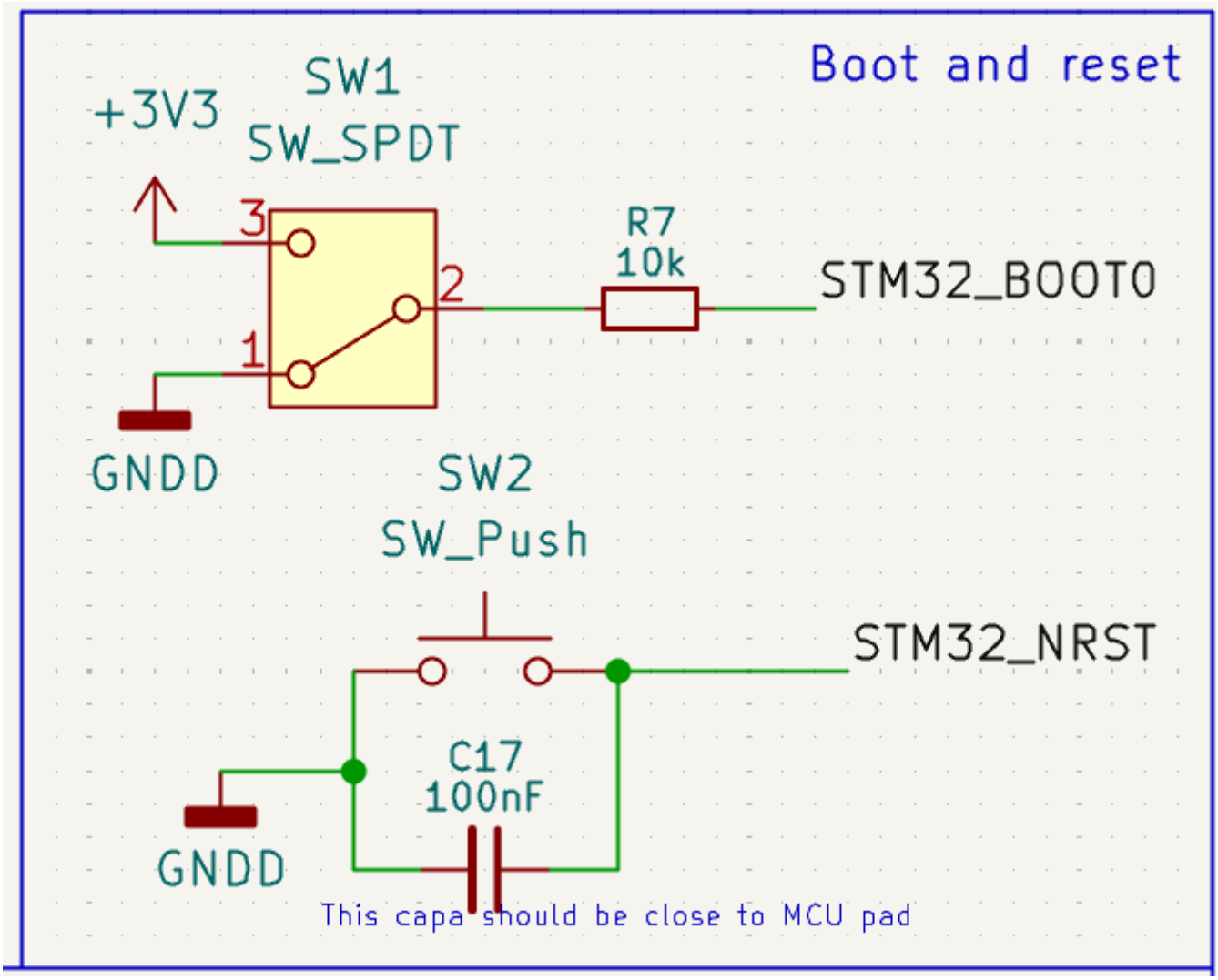
			fixed on the bootloader with 0x91 version.
USART1 (on PA9/PA10)	USART1	Enabled	Once initialized, the configuration is 8-bit, even parity, and one stop bit
	USART1_RX pin	Input	PA10 pin: USART1 in reception mode. Used in alternate push-pull, pull-up mode.
	USART1_TX pin	Output	PA9 pin: USART1 in transmission mode. Used in alternate push-pull, pull-up mode. Set as input until USART1 is detected on the bootloader version 0x91.
	USART1	Enabled	Once initialized, the configuration is 8-bit, even parity,

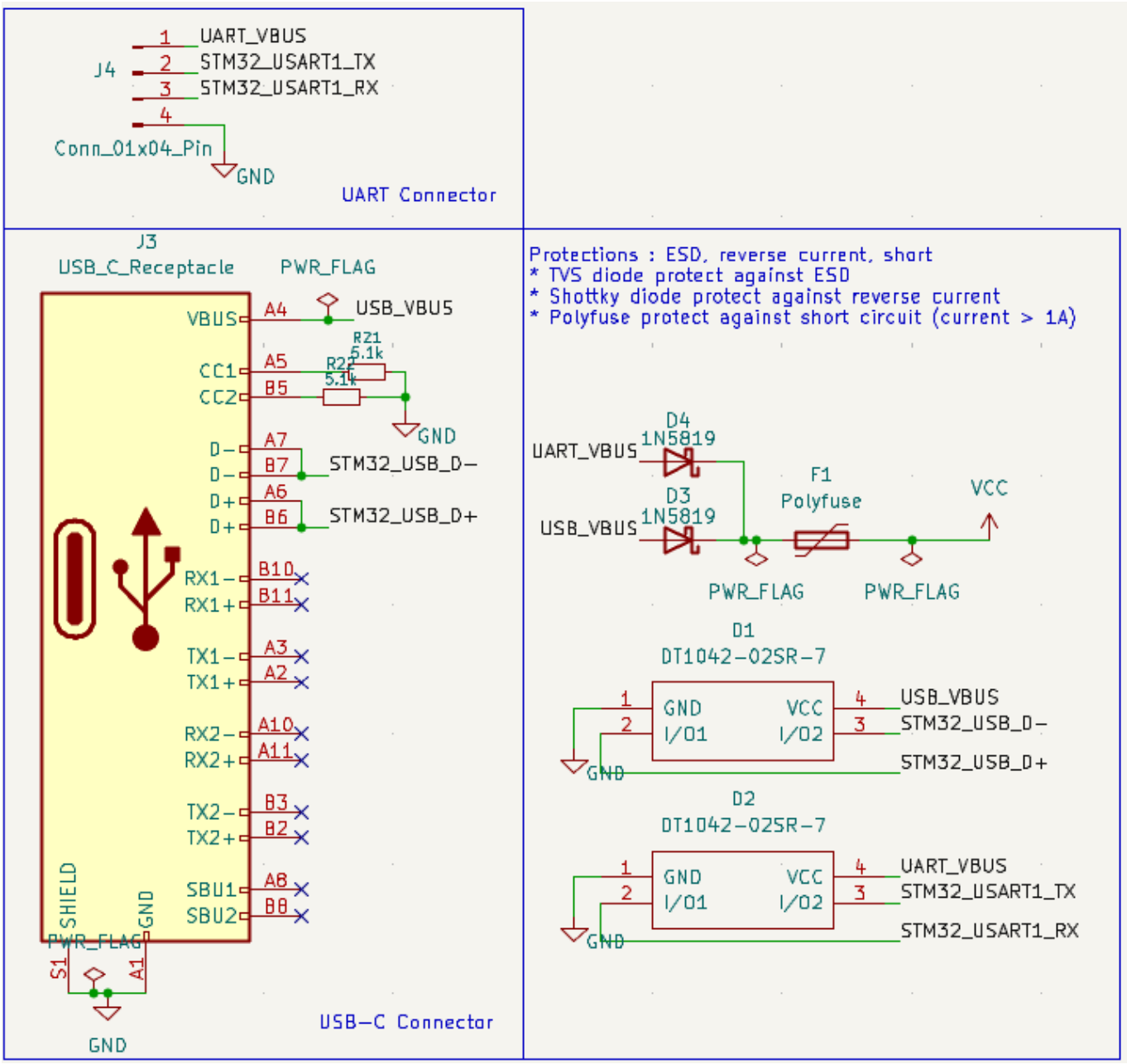
3. SPI:

Table 123. STM32H74xxx/75xxx configuration in system memory boot mode (continued)

Bootloader	Feature/Peripheral	State	Comment
SPI1	SPI1	Enabled	The SPI1 configuration is: – Slave mode – Full Duplex – 8-bit MSB – Speed up to 8 MHz – Polarity: CPOL low, CPHA low, NSS hardware.
	SPI1_MOSI pin	Input	PA7 pin: slave data input line, used in push-pull, no pull mode.
	SPI1_MISO pin	Output	PA6 pin: slave data output line, used in push-pull, no pull mode.
	SPI1_SCK pin	Input	PA5 pin: slave clock line, used in push-pull, no pull mode.
	SPI1_NSS pin	Input	PA4 pin: slave chip select pin used in push-pull, no pull mode.

Schematic:


Boot and reset
SW1
SW_SPDT
+3V3
3
2
1
R7
10k
STM32_BOOT0
GNDD
SW2
SW_Push
STM32_NRST
C17
100nF
GNDD
This capa should be close to MCU pad

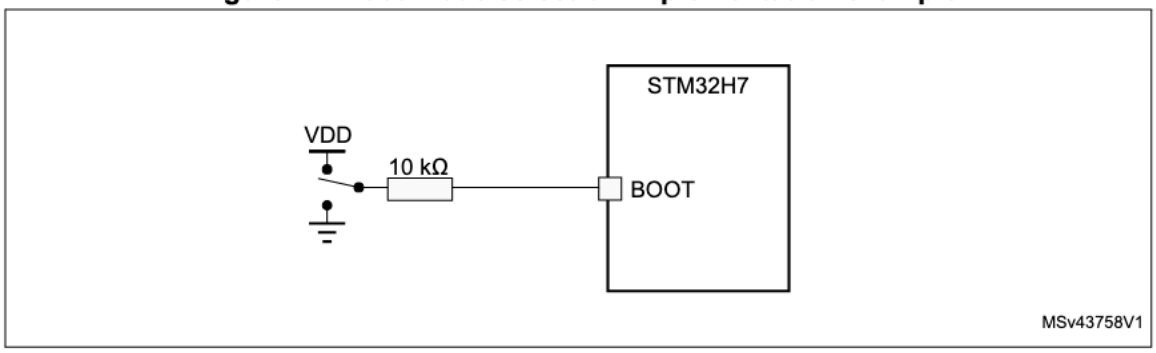


References:

5.2 Boot pin connection

Figure 14 shows the external connection required to select the boot memory of STM32H74xI/G and STM32H75xI/G microcontrollers.

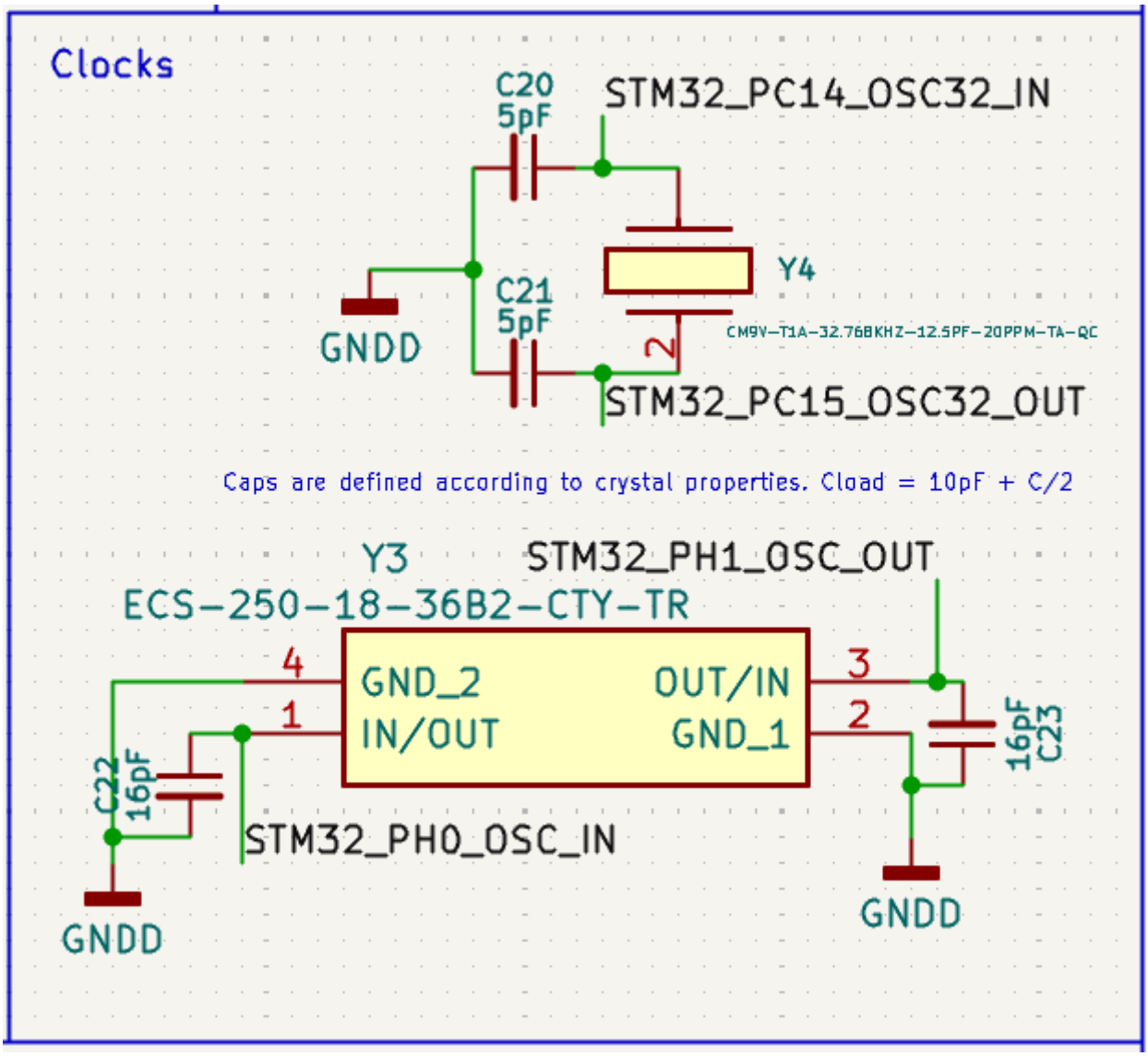
Figure 14. Boot mode selection implementation example



1. Resistor values are given only as a typical example.

4.1.3 STM32 - Clocks

Schematic:



References: It is based on AN4938.

4.1 HSE oscillator clock

The high-speed external clock signal (HSE) can be generated from two possible clock sources:

- HSE external user clock (see *Figure 10*).
- HSE external crystal/ceramic resonator (see *Figure 11*).

The resonator and the load capacitors have to be placed as close as possible to the oscillator pins to minimize the output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected resonator.

Figure 10. HSE external clock

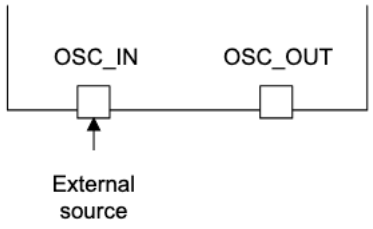
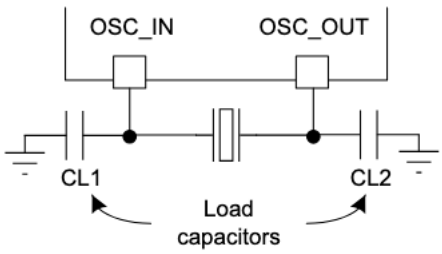


Figure 11. HSE crystal/ceramic resonators



Section 4.1.2 of :

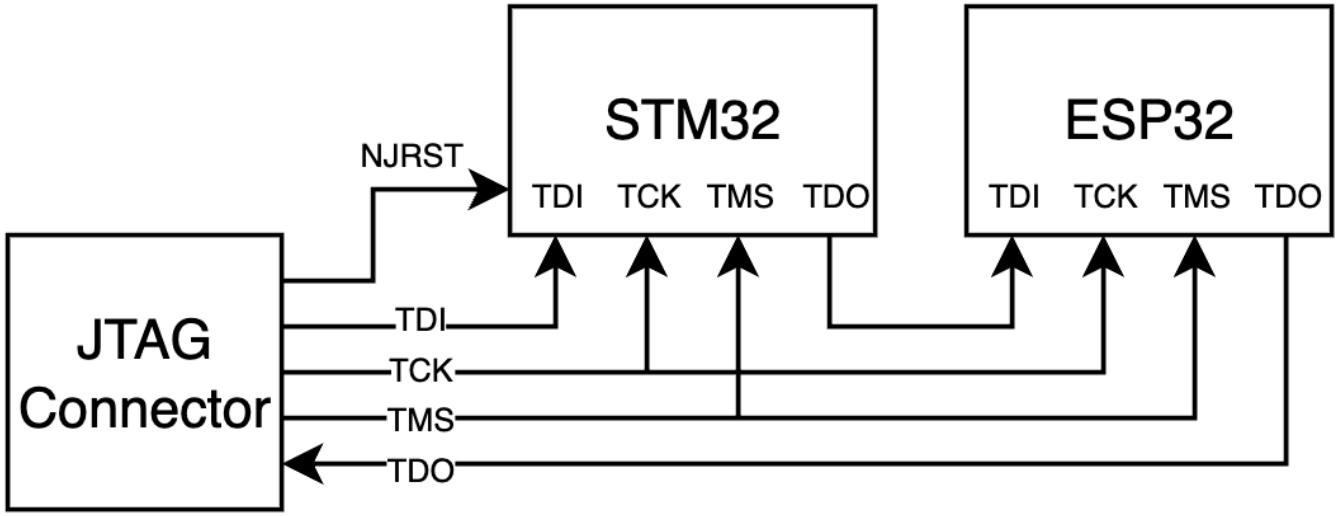
```
The external oscillator frequency ranges from 4 to 48 MHz. The external
oscillator has the advantage of producing a very accurate main clock. The
associated hardware configuration is shown in Figure 11. Using a 25 MHz
oscillator frequency is a good choice to get accurate Ethernet, USB OTG
high-speed peripheral, I2S, and SAI.
The resonator and the load capacitors have to be connected as close as
possible to the oscillator pins to minimize the output distortion and
startup stabilization time. The load capacitance values must be adjusted
according to the selected oscillator.
For CL1 and CL2 it is recommended to use high-quality ceramic capacitors
in the 5 pF to 25 pF range (typical), designed for high-frequency
applications and selected to meet the requirements of the crystal or
resonator. CL1 and CL2 are usually the same value. The crystal
manufacturer typically specifies a load capacitance that is the series
combination of CL1 and CL2. The PCB and MCU pin capacitances must be
included when sizing CL1 and CL2 (10 pF can be used as a rough estimate of
the combined pin and board capacitance).
```

Datasheet for the 25MHz clock (HSE): ECX_2236B2. The Load capacitance of this clock is 18pF (its in the name, do not refer to datasheet). So the Capacitors are 16pF to have `18 = 10 + 16/2`

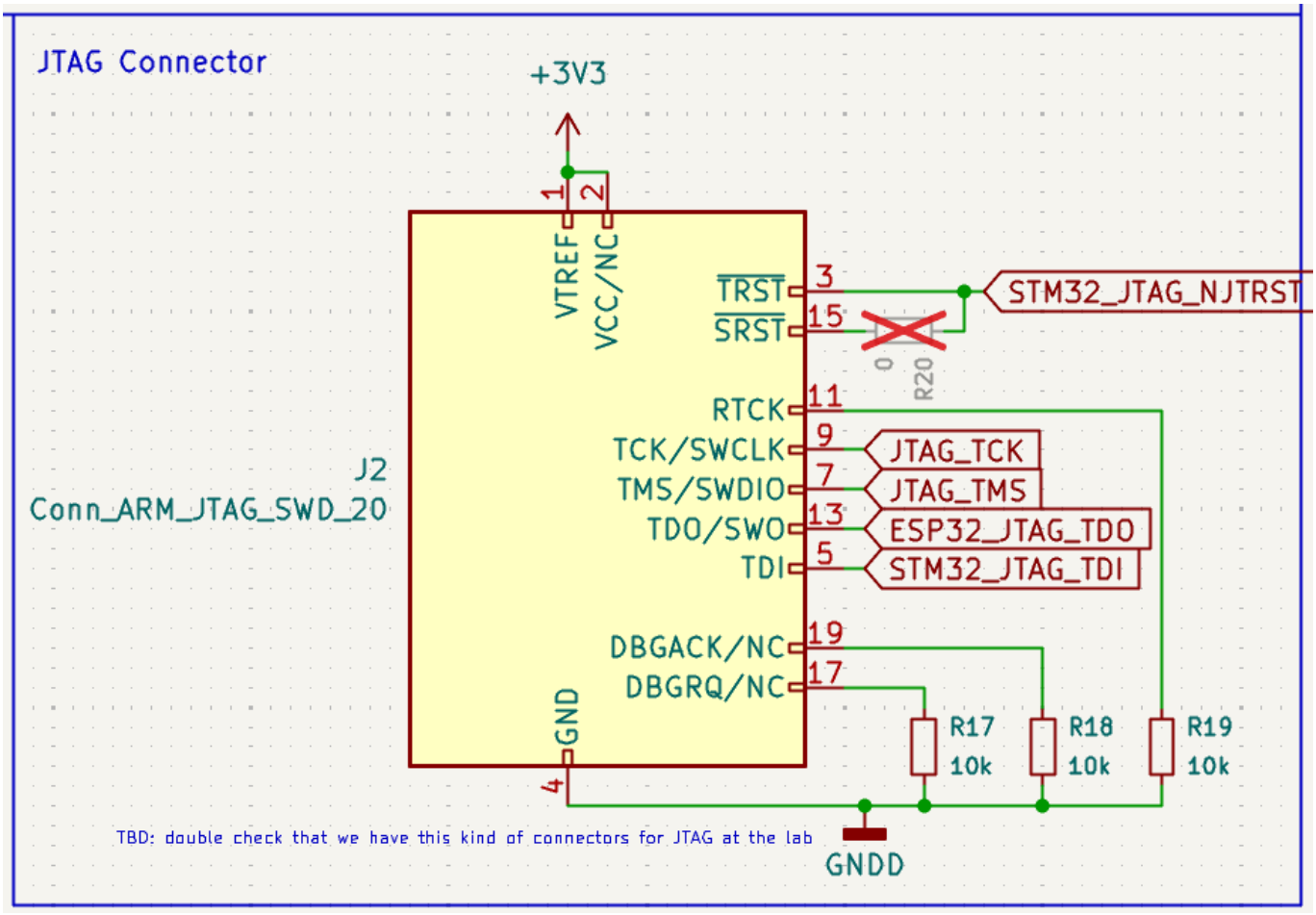
Datasheet for the 32.768kHz clock(OSC32): CM9V-T1A. The load capacitance of this clock is 12.5pF. So the capacitors are 5pF to have `12.5 = 10 + 5/2`

4.1.4 STM32 - Debug

JTAG interface:



Schematic:



References:

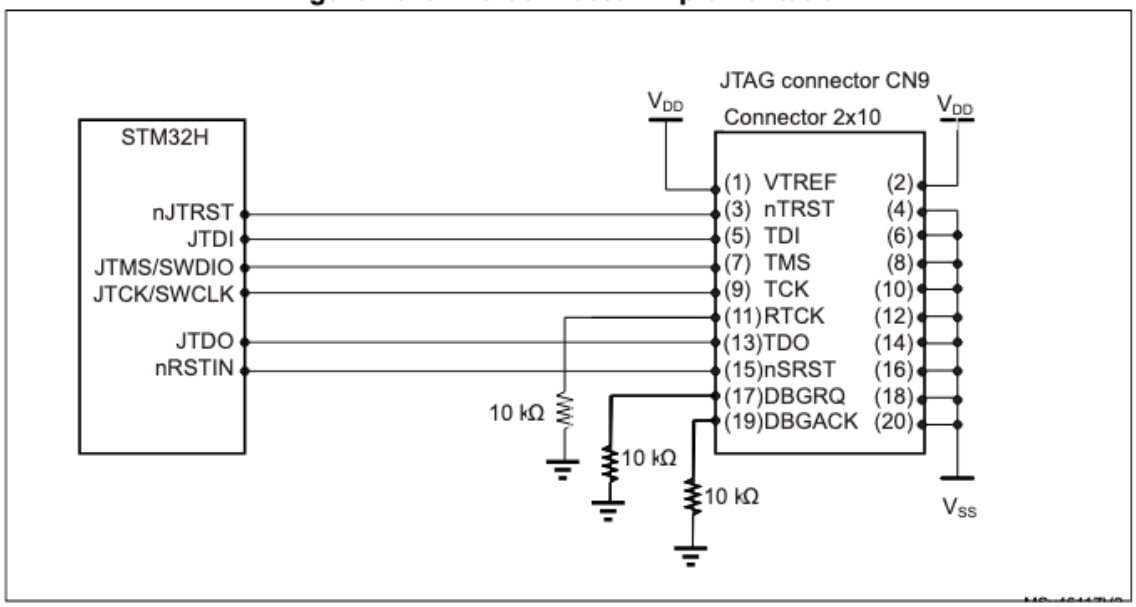
Table 6. SWJ debug port pins

SWJ-DP pin name	JTAG debug port		SW debug port		Pin assignment
	Type	Description	Type	Debug assignment	
JTMS/SWDIO	I	JTAG test mode Selection	IO	Serial wire data input/output	PA13
JTCK/SWCLK	I	JTAG test clock	I	Serial wire clock	PA14
JTDI	I	JTAG test data input	-	-	PA15
JTDO/TRACESWO	O	JTAG test data output	-	TRACESWO if asynchronous trace is enabled	PB3
NJTRST	I	JTAG test nReset	-	-	PB4

6.3.4 SWJ debug port connection with standard JTAG connector

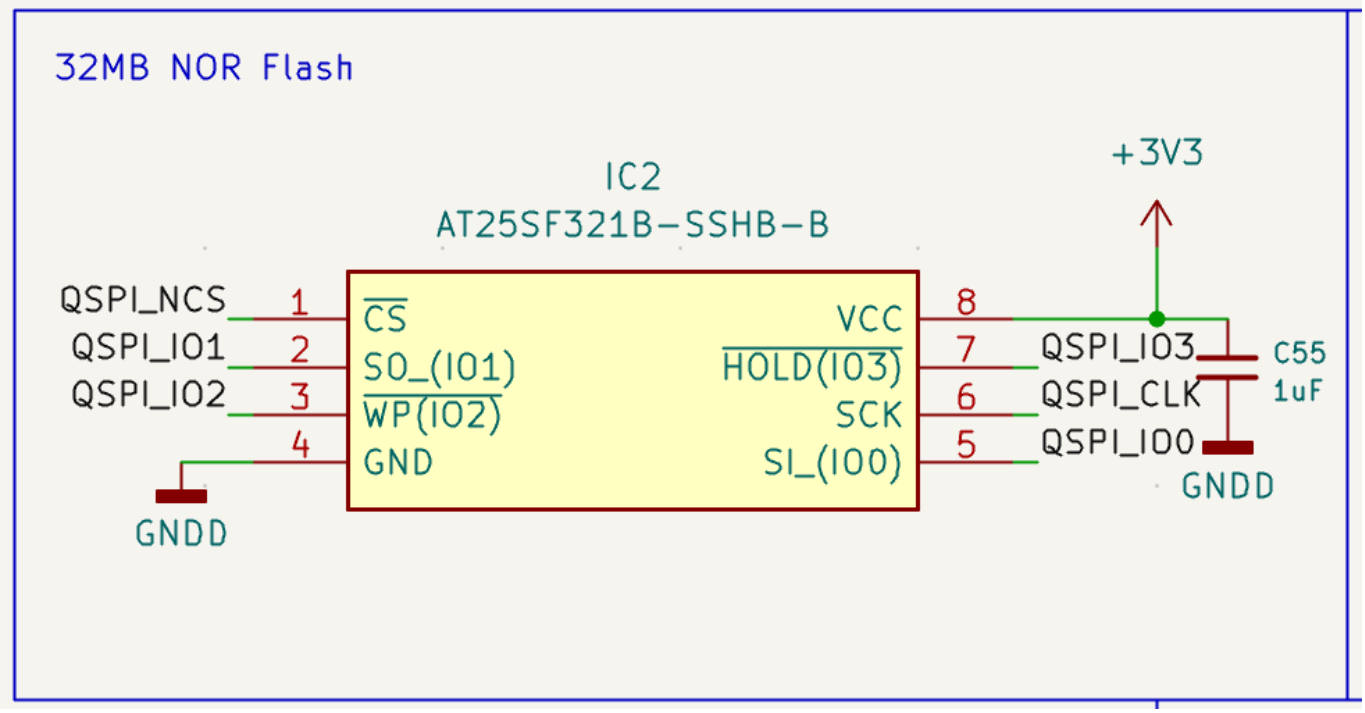
Figure 16 shows the connection between STM32H74xI/G and STM32H75xI/G devices and a standard JTAG connector.

Figure 16. JTAG connector implementation



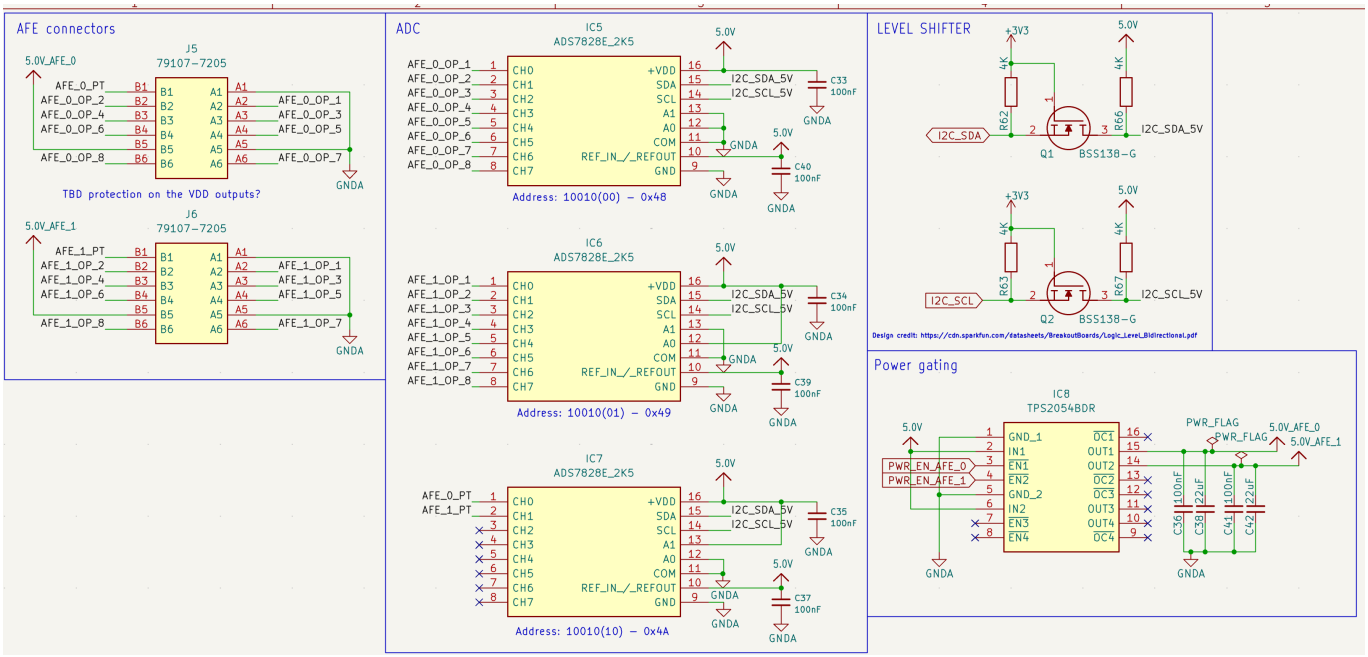
4.1.5 STM32 - Flash

An external 32MB NOR flash is added.

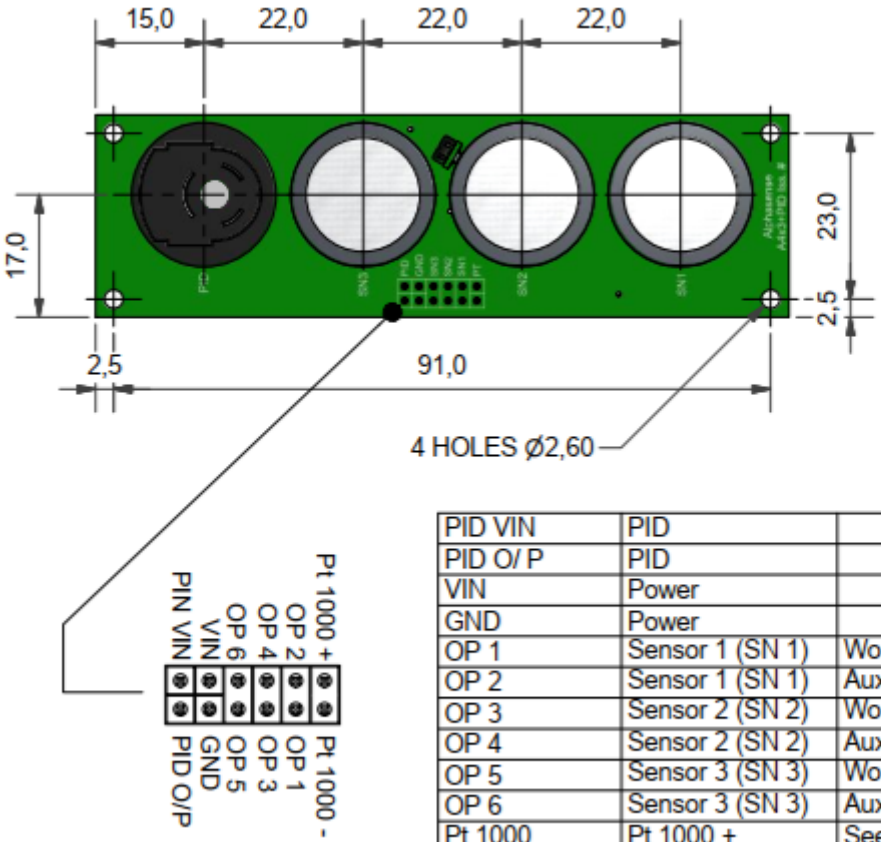


4.2 Analog Front End

Schematic:



References: This design is based on the previous board revision (3.3), on AFE Datasheet and components

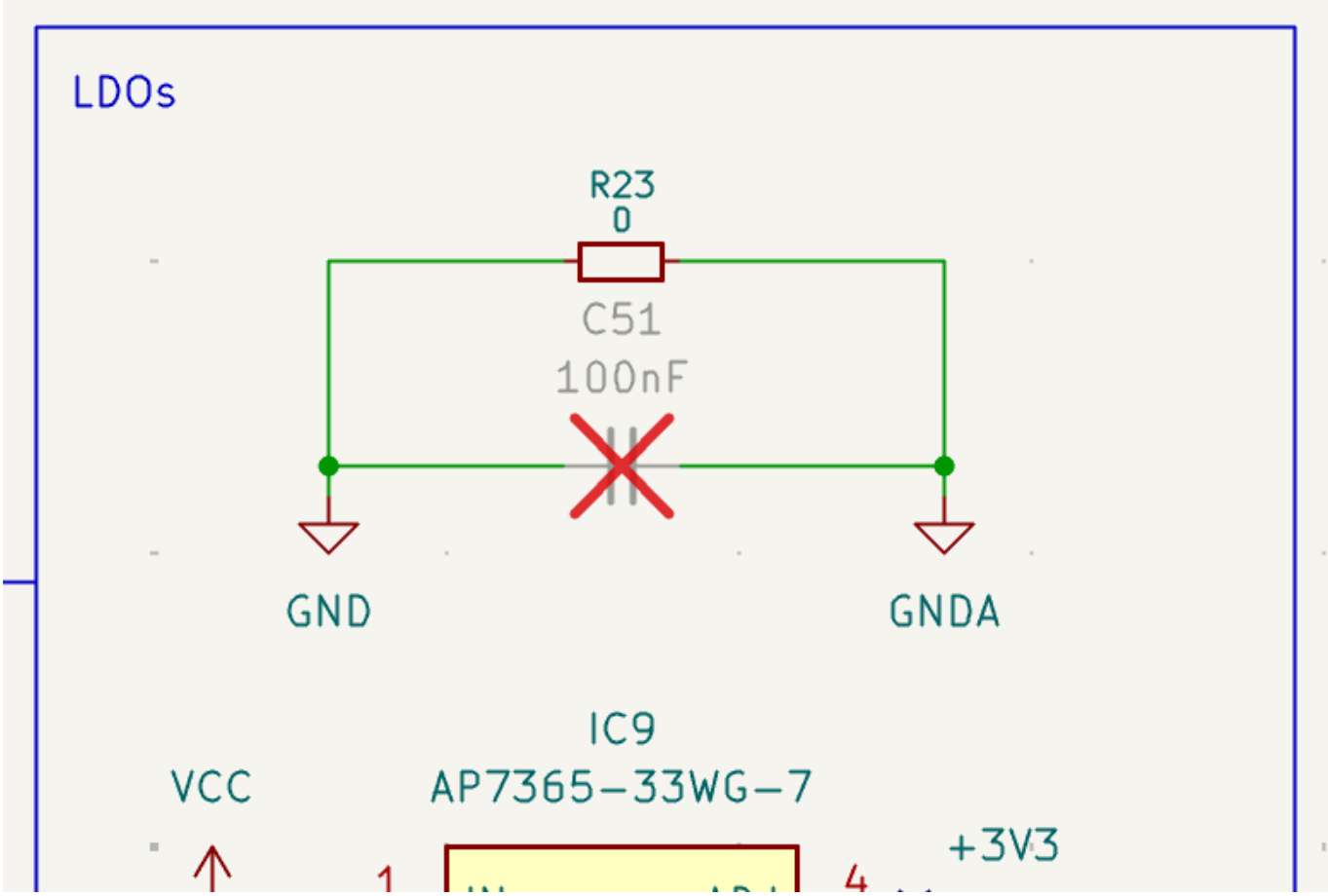


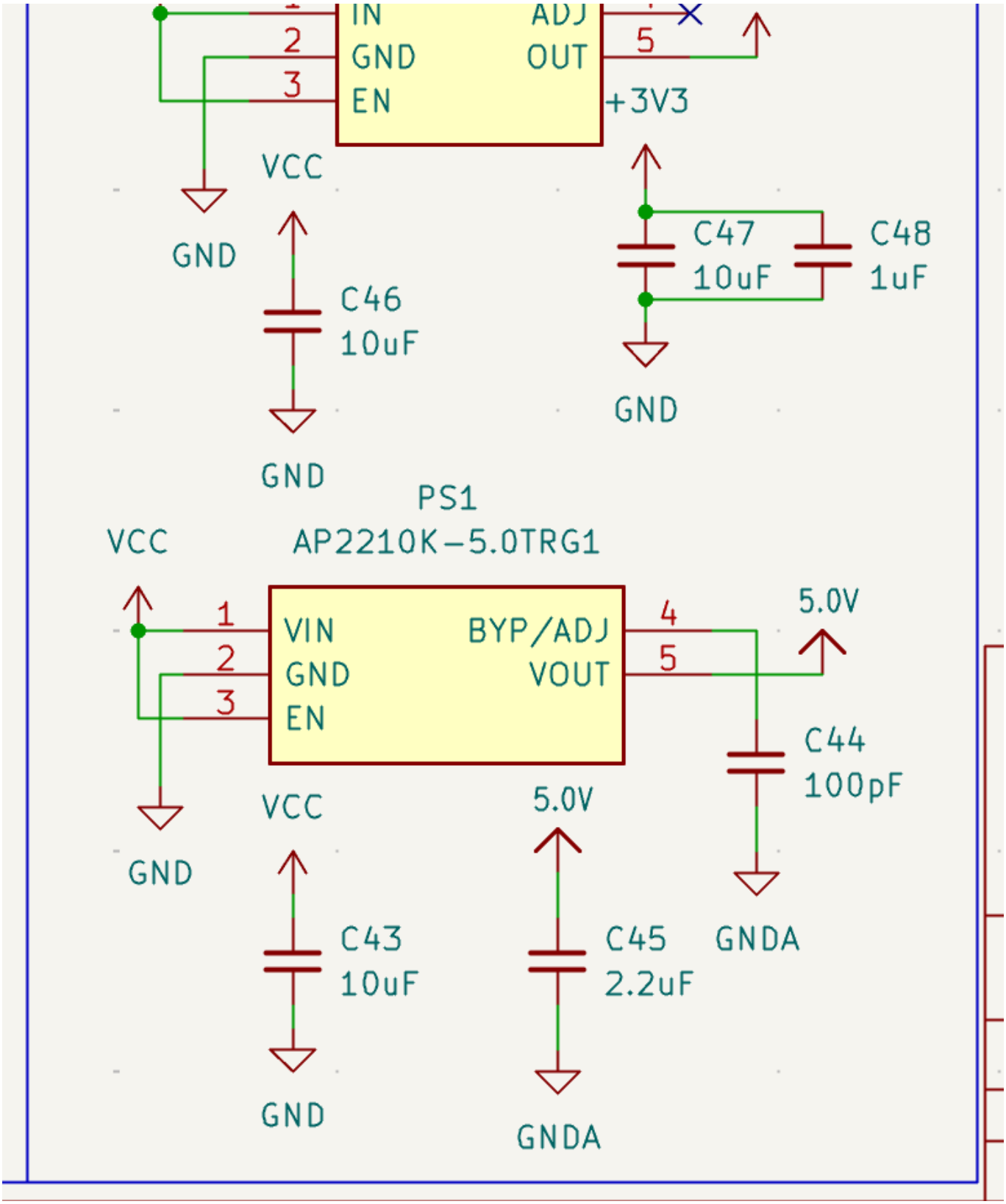
PID VIN	PID	
PID O/ P	PID	
VIN	Power	
GND	Power	
OP 1	Sensor 1 (SN 1)	Working electrode
OP 2	Sensor 1 (SN 1)	Auxillary electrode
OP 3	Sensor 2 (SN 2)	Working electrode
OP 4	Sensor 2 (SN 2)	Auxillary electrode
OP 5	Sensor 3 (SN 3)	Working electrode
OP 6	Sensor 3 (SN 3)	Auxillary electrode
Pt 1000	Pt 1000 +	See Notes
Pt 1000	Pt 1000 -	See Notes

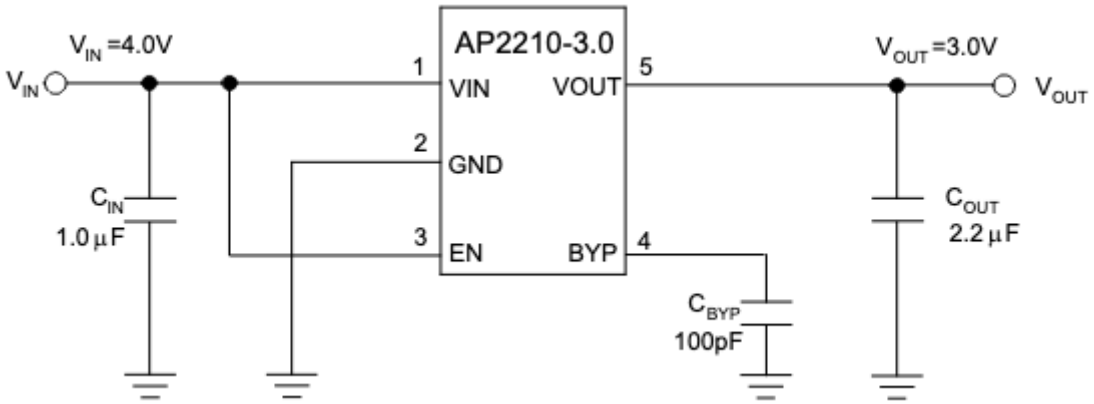
datasheets.

4.3 LDOs

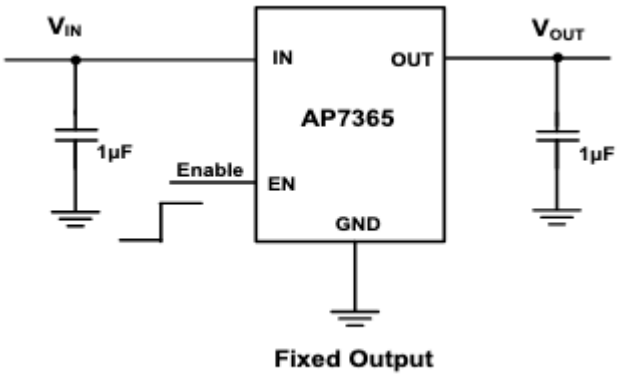
Schematic:




IN
GND
EN
ADJ
OUT
2
3
5
+3V3
VCC
GND
C46
10uF
GND
C47
10uF
C48
1uF
GND
PS1
AP2210K-5.0TRG1
VCC
1
2
3
VIN
GND
EN
BYP/ADJ
VOUT
4
5
5.0V
C44
100pF
GND
VCC
C43
10uF
GND
5.0V
C45
2.2uF
GNDA
GNDA



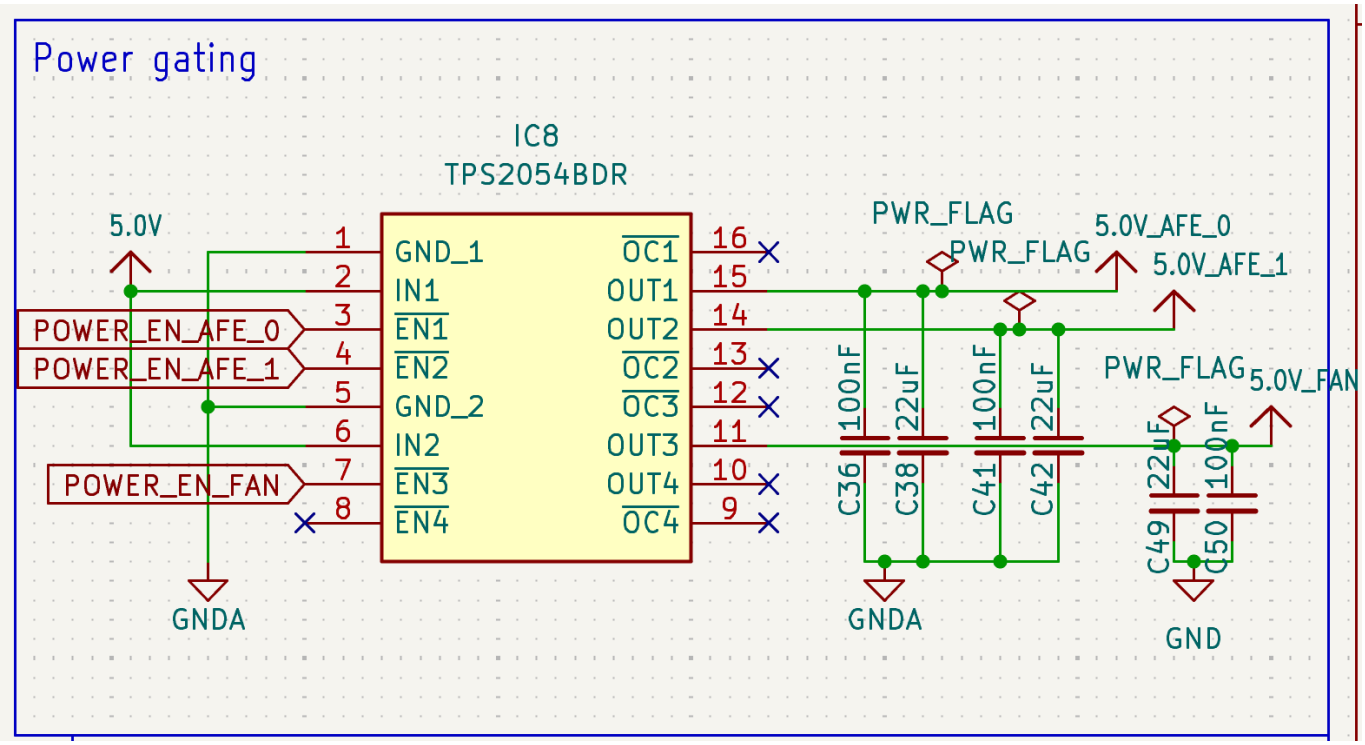
References:



Note Datasheet LDO 3.3V says that pin 4 should be unconnected when using a fix LDO.

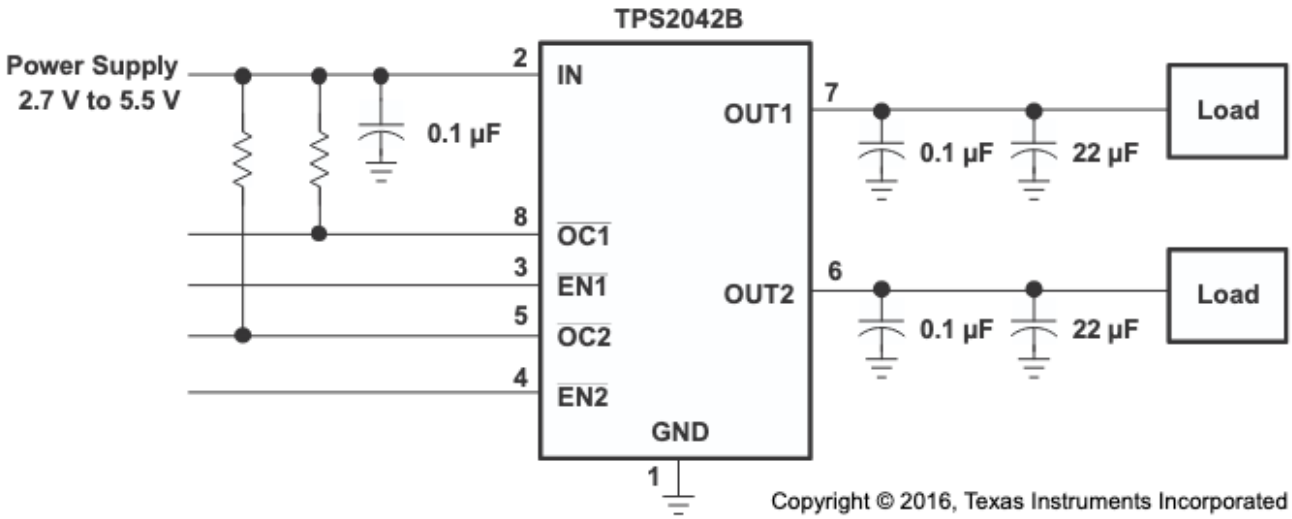
4.4 Power Gating

Schematic:



Note: The enable signal is considered high if the voltage is above 2.7V so there is no need for a level shifter here as the STM32 is producing a 3.3V level.

References:

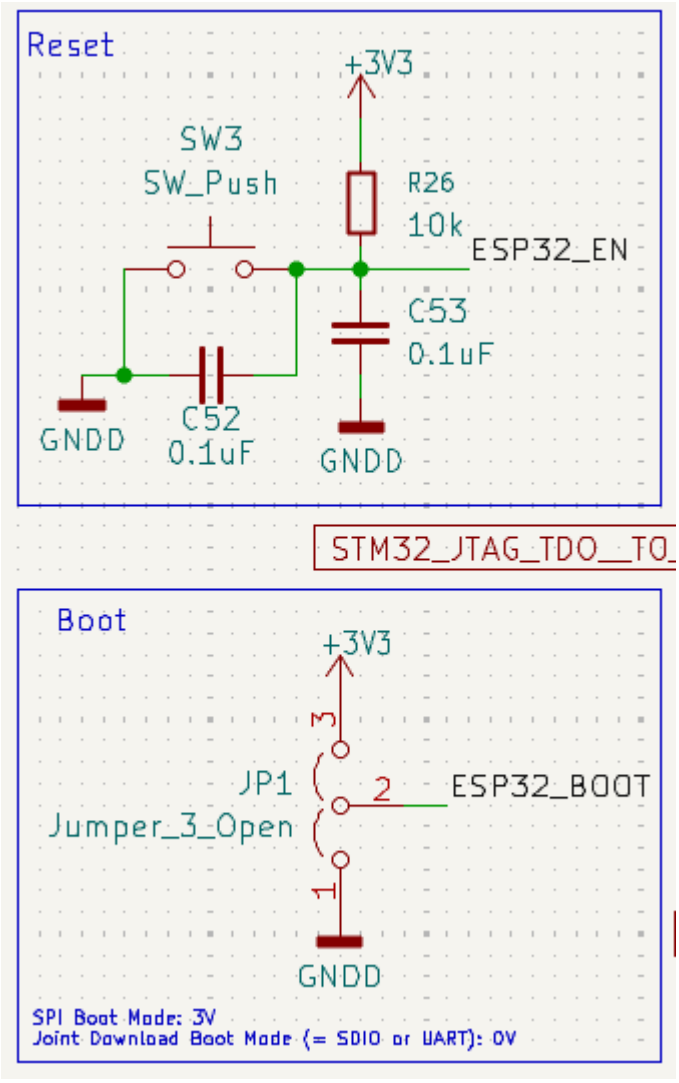


Typical Application Schematic

4.5 WiFi Module : ESP32-WROOM-32E-N16

4.5.1 ESP32 - Booting and programming

Schematic:



To program the ESP32:

- First, connect ESP32_BOOT to GND using the jumper.
- Then program the 16MB flash using UART.
- Finally, put the jumper back to 3.3v to boot from the flash that has been programmed.

References:

- DevKitC V4.

4.5.2 ESP32 - Debug

Debug can be performed via JTAG. See STM32 debug section.

4.5.3 STM32 - ESP32 communication

To communicate with the ESP from the STM32, decision is made to use SPI because it is fast and the ESP32 has SPI slave available natively.

References: The General Purpose SPI can behave in slave mode. For them any GPIO is possible, however for best performance, it is better to use parallel QSPI pins. Since HSPI pins are used by JTAG, decision is

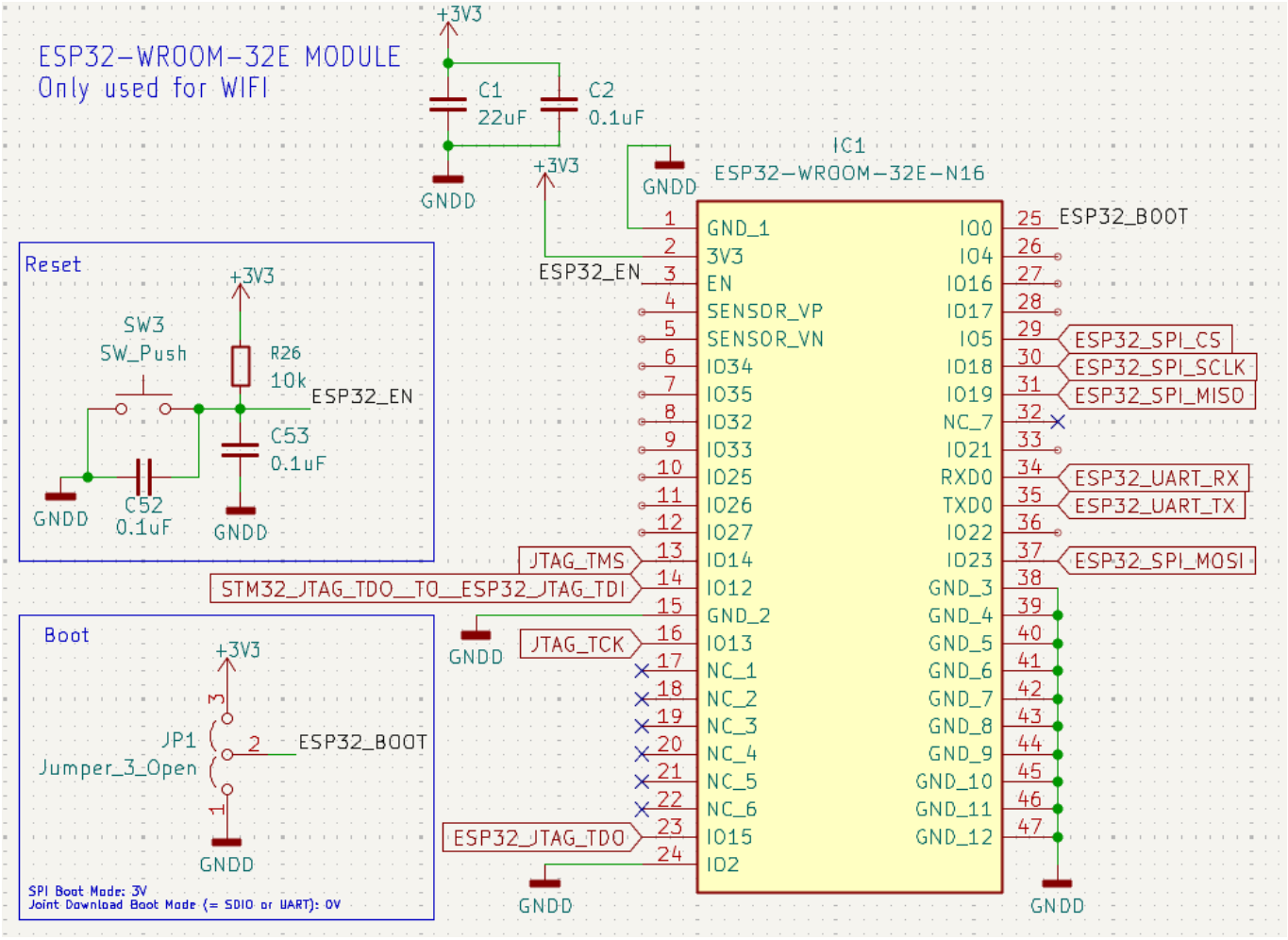
made to use the VSPI pins.

General Purpose SPI	HSPIQ_in/_out	Any GPIO Pins	Standard SPI consists of clock, chip-select, MOSI and MISO. These SPIs can be connected to LCD and other external devices. They support the following features: • Both master and slave modes; • Four sub-modes of the SPI transfer format; • Configurable SPI frequency; • Up to 64 bytes of FIFO and DMA.
	HSPID_in/_out		
	HSPICLK_in/_out		
	HSPI_CS0_in/_out		
	HSPI_CS1_out		
	HSPI_CS2_out		
	VSPIQ_in/_out		
	VSPID_in/_out		
	VSPICLK_in/_out		
	VSPI_CS0_in/_out		
	VSPI_CS1_out		
	VSPI_CS2_out		

4 Functional Description

Interface	Signal	Pin	Function
Parallel QSPI	SPIHD	SD_DATA_2	Supports Standard SPI, Dual SPI, and Quad SPI that can be connected to the external flash and SRAM
	SPIWP	SD_DATA_3	
	SPICS0	SD_CMD	
	SPICLK	SD_CLK	
	SPIQ	SD_DATA_0	
	SPID	SD_DATA_1	
	HSPICLK	MTMS	
	HSPICS0	MTDO	
	HSPIQ	MTDI	
	HSPID	MTCK	
	HSPIHD	GPIO4	
	HSPIWP	GPIO2	
	VSPICLK	GPIO18	
	VSPICS0	GPIO5	
	VSPIQ	GPIO19	
	VSPID	GPIO23	
	VSPIHD	GPIO21	
	VSPIWP	GPIO22	

Schematic:



4.5.4 ESP32 - Pin assignments

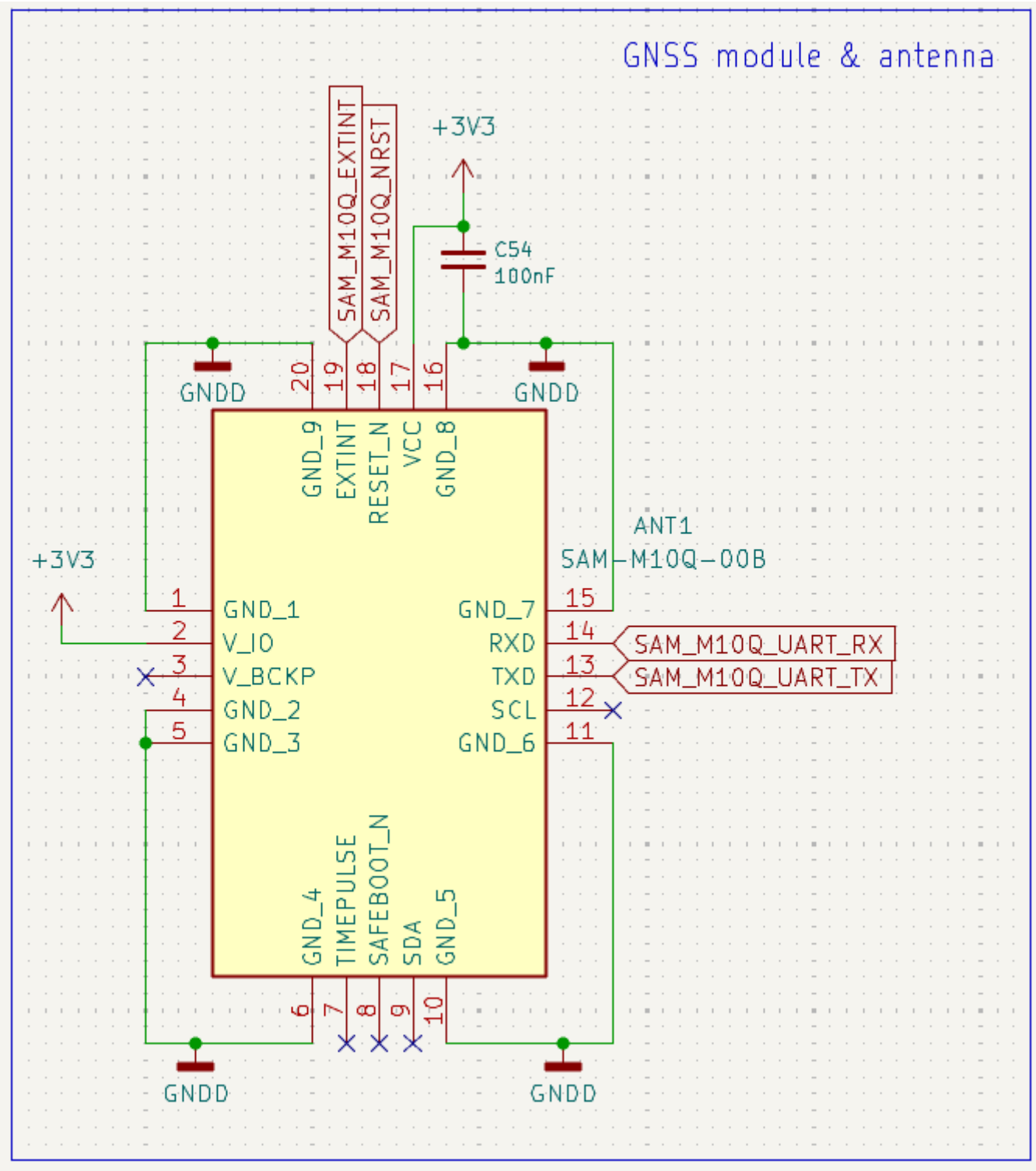
This table contains all the pin assignments. It was updated during layout to make rooting easier.

IO°	Type	Active Mode	Debug/Bringup
IO0	I/O		booting
IO1 TXD0	I/O		esp32_uart_tx
IO2	I/O		booting
IO3 RXD0	I/O		esp32_uart_rx
IO4	I/O		
IO5	I/O	esp32_spi_cs	
IO12	I/O		jtag_tdi
IO13	I/O		jtag_tck
IO14	I/O		jtag_tms
IO15	I/O		jtag_tdo
IO16	I/O	esp32_led_0	

IO°	Type	Active Mode	Debug/Bringup
IO17	I/O		
IO18	I/O	esp32_spi_sclk	
IO19	I/O	esp32_spi_miso	
IO21	I/O		
IO22	I/O		
IO23	I/O	esp32_spi_mosi	
IO25	I/O		
IO26	I/O	stm32_esp32_irq_1	
IO27	I/O	stm32_esp32_irq_2	
IO32	I/O		
IO33	I/O		
IO34	I		
IO35	I		
IO36 SENSOR_VP	I		
IO39 SENSOR_VN	I		

4.6 GNSS module and antenna - SAM-M10Q

Schematic:


GNSS module & antenna
SAM_M10Q_EXTINT
SAM_M10Q_NRST
+3V3
C54
100nF
GNDD
GNDD
20
19
18
17
16
GND_9
EXTINT
RESET_N
VCC
GND_8
ANT1
SAM-M10Q-00B
+3V3
1
2
3
4
5
GND_1
V_IO
V_BCKP
GND_2
GND_3
GND_7
RXD
TXD
SCL
GND_6
15
14
13
12
11
SAM_M10Q_UART_RX
SAM_M10Q_UART_TX
GND_4
TIMEPULSE
SAFEBOOT_N
SDA
GND_5
6
7
8
9
10
GNDD
GNDD

References:

Here are some key features for a SAM-M10Q typical design:

- VCC and V_IO are connected together to a single supply as shown in Figure 28.
- V_BCKP supply is optional. If present, the hardware backup mode is supported. This mode maintains the time and GNSS orbit data in the battery-backed RAM memory if the main supply is switched off.
  If there is no backup supply, time aiding with the UBX-MGA-INI-TIME_UTC message (optionally with a timing signal at the EXTINT pin) and the GNSS orbit data from the AssistNow services or stored on the host controller can be used to reduce the TTFF.
- UART and I2C communication interfaces are available.
- For an absolute minimum design using UART, other PIOs (RESET_N, EXTINT, TIMEPULSE, SDA, SCL, SAFEBOOT_N) can be left open.

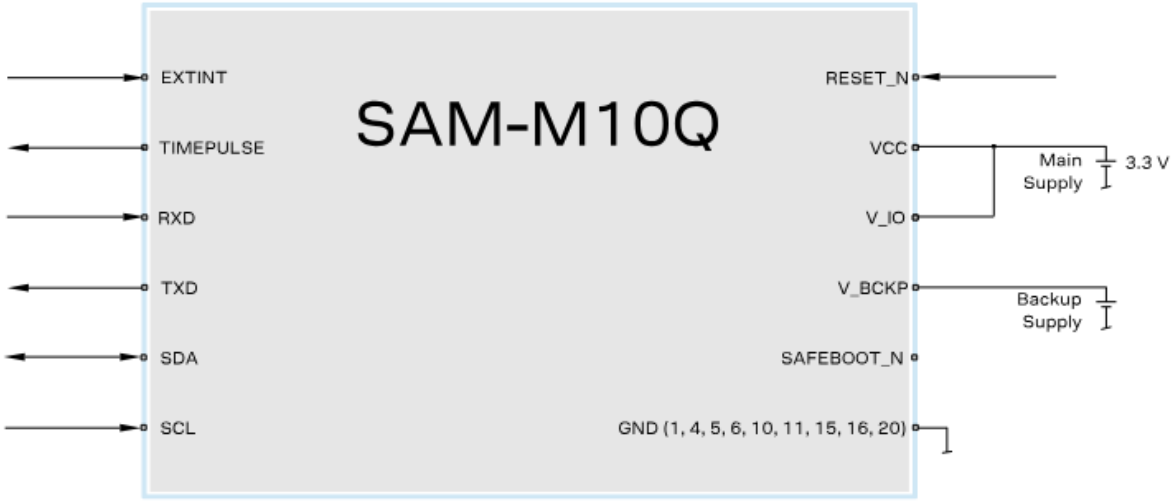
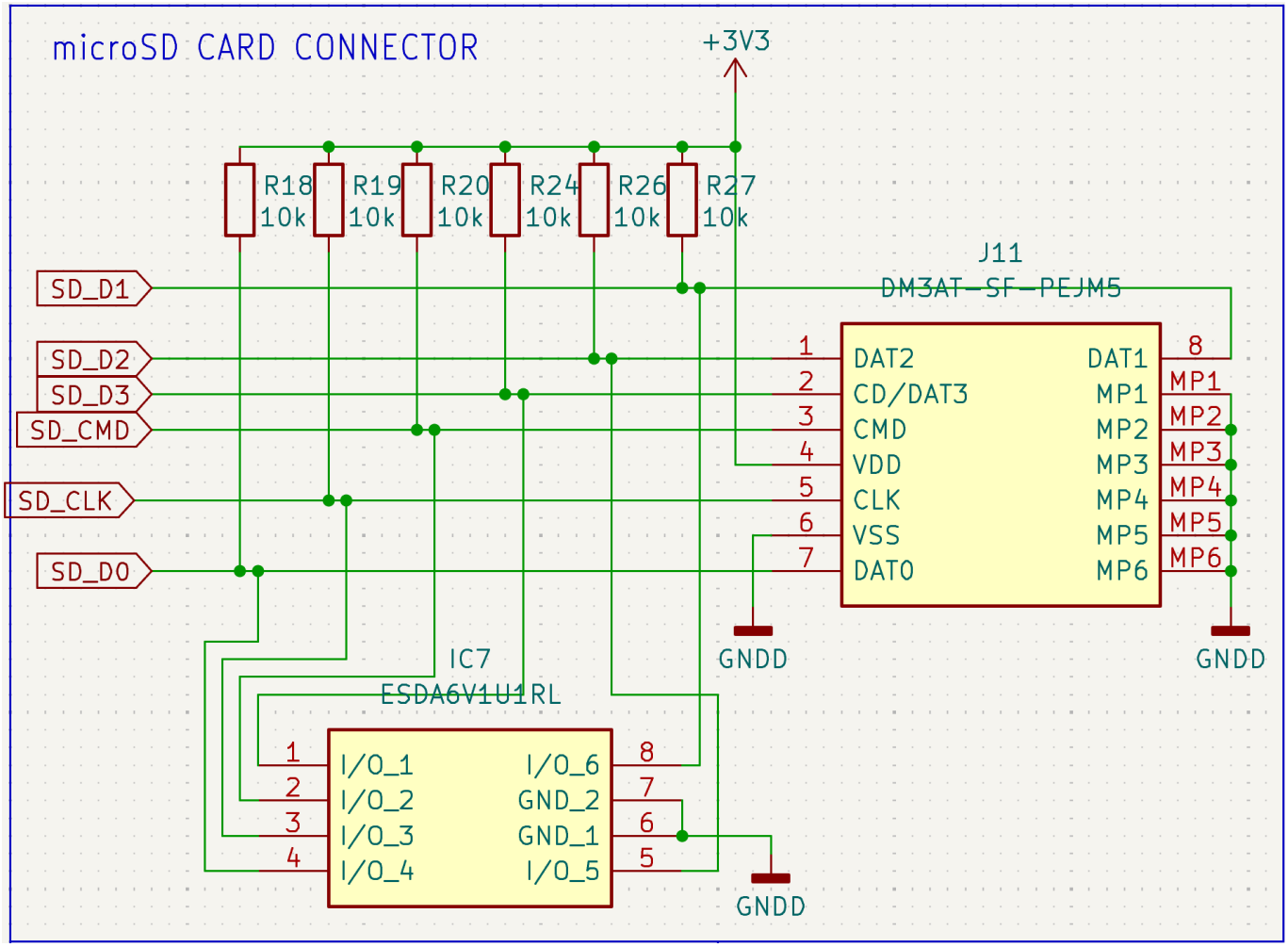


Figure 28: Typical design

4.7 MicroSD card

Schematic:



4.8 Debug features

Debug through JTAG is possible see STM32 Debug, ESP32 Debug.

Test points have been added to:

- monitor SDMMC bus (STM32-MicroSD)
- monitor fuse voltage
- monitor UART (STM32-GNSS module)
- monitor SPI (STM32-ESP32)
- monitor I2C both before and after level shifter (STM32-ADCs)
- monitor QSPI (STM32-Flash)
- monitor quartz

Optional 2.54mm Header have added to:

- monitor/force power
- monitor JTAG transmit JTAG to each chip individually.

4.9 Schematic Revision 4.0

Schematic Revision 4.0

5. Layout