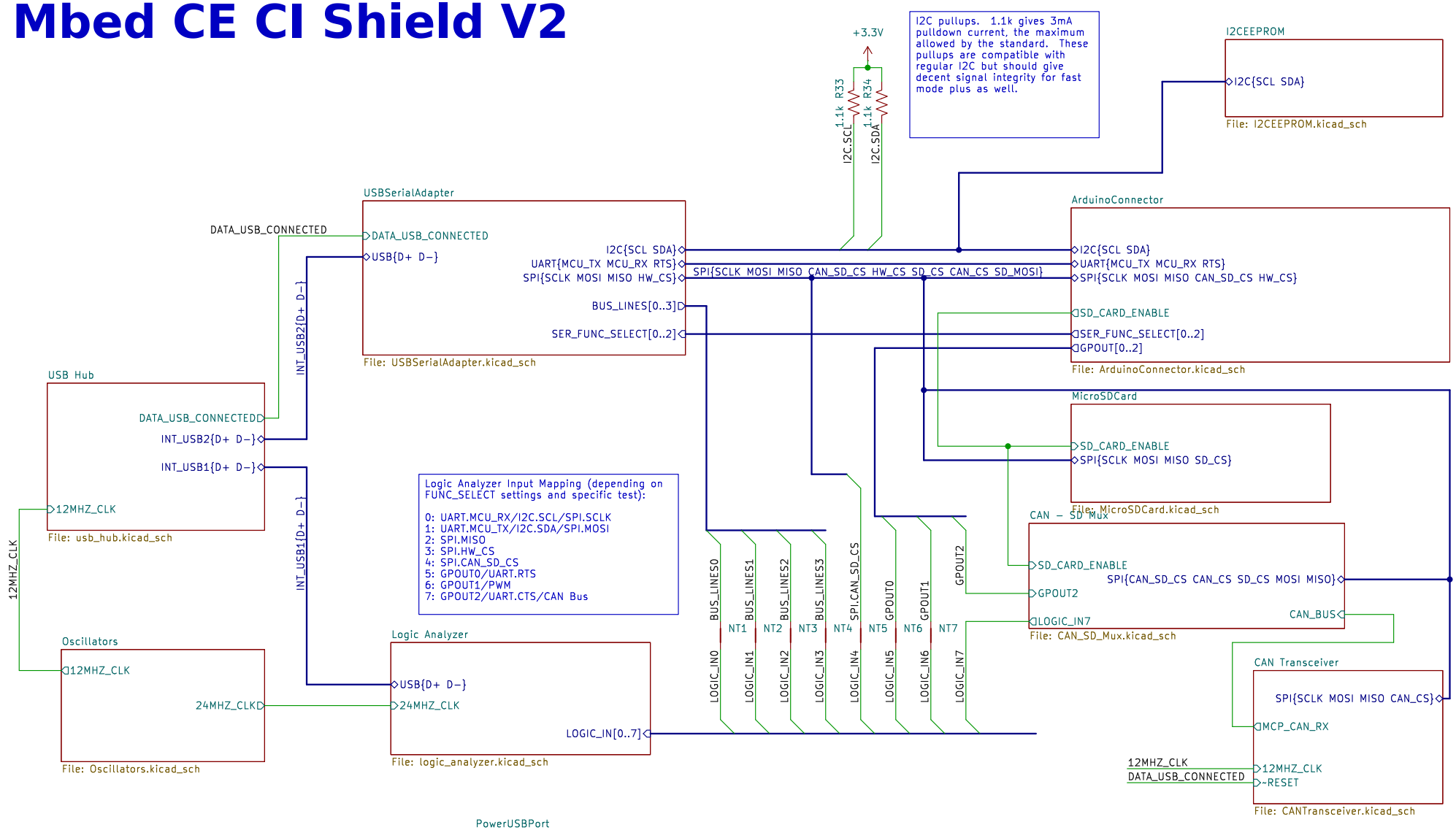
# Mbed CE CI Shield V2



Based on the reference design and review feedback, I assume that needed debouncing is:
– 1x 10uF and 1x 1uF on VREG (which I assume is needed only if the internal regulator is used)
– A 100nF for each VCC_A pin, plus 2x (2.2uF + 1 Ohm) on VCC_A, plus a ferrite bead
– A 100nF for each VCC_D, plus a ferrite bead

U2A
CY7C65632-48AXCT
Power Pins
5V to 3.3V Regulator
VCC
VREG
VCC_A
VCC_D
GND
+3.3V
FB1 300@100MHz
PWR_FLAG
HUB_VDD_A
C33 2.2uF
C34 2.2uF
R48 1
R49 1
C46 100nF
C7 100nF
C3 100nF
C6 100nF
C4 100nF
FB2 300@100MHz
HUB_VDD_D
C10 100nF
C11 100nF

RC timer is used to bring the hub chip's reset line high ~10ms after VBUS from the upstream port is detected, per the reference design.
This circuit also divides the 5V down to 3.3V for use as a general reset signal, and makes sure that the reset isn't high if the power USB is not connected.

UPSTREAM_VBUS
R14 4.7k
D7 BAT54W
C21 2.2uF
R13 9.1k
DATA_USB_CONNECTED
12MHZ_CLK
GANG/SUSPEND
R1 649
R15 10k

U2B
CY7C65632-48AXCT
Support/Misc Pins
SEL48
SEL27
SELFPWR
GANG/SUSPEND
RREF
TEST/I2C_SCL
~RESET
XIN
XOUT

* SEL48 and SEL27 are tied high to select a 12MHz external clock.
* SELFPWR is tied high to indicate that the hub is not powered by the upstream USB port
* TEST/I2C_SCL is pulled down to disable test mode

## External port power switches

U6
BD2221G-LBTR
VIN
VOUT
-EN
GND
-OC
+5V
C14 1u
~PWR_EN1
PREFILT_VBUS1
FB4 300@100MHz
PWR_FLAG
VBUS1
C9 2.2uF
C18 100u
~OVR1
R9 100k

U7
BD2221G-LBTR
+5V
C15 1u
~PWR_EN2
PREFILT_VBUS2
FB5 300@100MHz
PWR_FLAG
VBUS2
C19 2.2uF
C20 100u
~OVR2
R10 100k

## Upstream Port

Upstream port (USB-C, connects to host computer).
This uses the USB connector setup from here:
https://hackaday.com/2022/12/06/usb-c-introduction-for-hackers/
This requests 5V@1.5A for our board.

U2C
CY7C65632-48AXCT
Upstream Port
D-
D+
UPSTREAM_D-
UPSTREAM_D+
UPSTREAM_VBUS
R2 5.1k
R3 5.1k
U3
USBLC6-2SC6
US_PREFILT_D-
US_PREFILT_D+
J1
USB4105-GF-A
VBUS
CC1
CC2
D-
D+
SBU1
SBU2
GND
SHIELD
S1

## External Ports

U2D
CY7C65632-48AXCT
Downstream Port 1
DD-[1]
DD+[1]
AMBER[1]/SPI_CS
GREEN[1]/SPI_SK/FIXED_PORT1
~OVR[1]
~PWR[1]/I2C_SDA
D1-
D1+
AMBER1
GREEN1
~OVR1
~PWR_EN1
U4
USBLC6-2SC6
VBUS1
VBUS2

U2E
CY7C65632-48AXCT
Downstream Port 2
DD-[2]
DD+[2]
AMBER[2]/SPI_MOSI/PWR_PIN_POL
GREEN[2]/SPI_MISO/FIXED_PORT2
~OVR[2]
~PWR[2]
D2-
D2+
AMBER2
GREEN2
~OVR2
~PWR_EN2
U5
USBLC6-2SC6
PREFILT_D1+
PREFILT_D1-
PREFILT_D2+
PREFILT_D2-
J2
UJ2-ADH-3-TH
GND1
Shield

## Internal Ports

Internal ports. These ports are used to provide USB to devices inside the CI shield. Due to this, they lack ESD protection or their own power supply.

The FIXED_PORT pins are strapped to high to indicate to the hub that these devices are non-removable.

The SET_PORT_NUM[2:1] pins are strapped to 00 to indicate that all 4 ports are in use.

The OVR pins are tied to high to disable overcurrent detection.

U2F
CY7C65632-48AXCT
Downstream Port 3
DD-[3]
DD+[3]
AMBER[3]/SET_PORT_NUM2
GREEN[3]/FIXED_PORT3
~OVR[3]
~PWR[3]
INT_USB1.D-
INT_USB1.D+
INT_USB1{D+ D-}
R4 100k
R7 100k
R8 100k
INTERNAL_PORT_OVR

U2G
CY7C65632-48AXCT
Downstream Port 3
DD-[4]
DD+[4]
AMBER[4]/SET_PORT_NUM1
GREEN[4]/FIXED_PORT4
~OVR[4]
~PWR[4]
INT_USB2.D-
INT_USB2.D+
INT_USB2{D+ D-}
R5 100k
R6 100k

Hub Active LED:
Lights up when hub is active (NOT in suspend mode).
100k pulldown is a strapping setting to set that the PWR pins can act independently.

GANG/SUSPEND
D5
R16 100k
R21 300

## Port Indicators

GREEN1
AMBER1
GREEN2
AMBER2
D1
D3
D2
D4
R17 270
R19 300
R18 270
R20 300

Note: For the GREEN pins, we rely on the internal pulldown resistor to strap them as low. This configures port 1 and port 2 as removable.

Also, for AMBER2, we allow this to be pulled low internally to strap the ~PWR[] pins to active-low polarity.

USB hub with 2 internal and 2 external ports.

MBED CE

Sheet: /USB Hub/
File: usb_hub.kicad_sch

**Title: USB Hub**

| Size: A3 | Date: | Rev: |
|---|---|---|
| KiCad E.D.A. 8.0.8 | | Id: 2/11 |

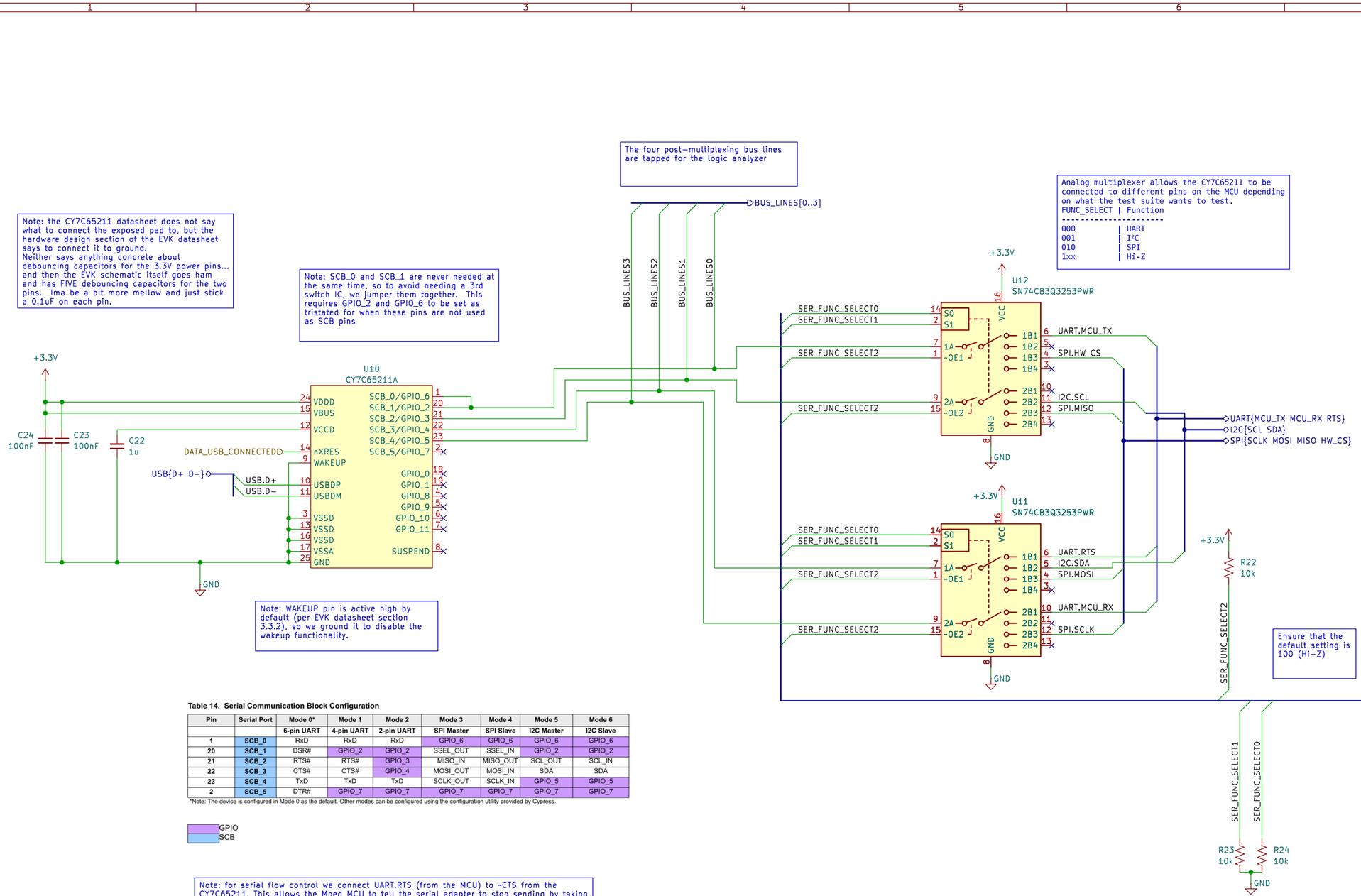
Note: the CY7C65211 datasheet does not say what to connect the exposed pad to, but the hardware design section of the EVK datasheet says to connect it to ground.
Neither says anything concrete about debouncing capacitors for the 3.3V power pins... and then the EVK schematic itself goes ham and has FIVE debouncing capacitors for the two pins. Ima be a bit more mellow and just stick a 0.1uF on each pin.

Note: SCB_0 and SCB_1 are never needed at the same time, so to avoid needing a 3rd switch IC, we jumper them together. This requires GPIO_2 and GPIO_6 to be set as tristated for when these pins are not used as SCB pins

The four post-multiplexing bus lines are tapped for the logic analyzer

Analog multiplexer allows the CY7C65211 to be connected to different pins on the MCU depending on what the test suite wants to test.

| FUNC_SELECT | Function |
|---|---|
| 000 | UART |
| 001 | I²C |
| 010 | SPI |
| 1xx | Hi-Z |




Note: WAKEUP pin is active high by default (per EVK datasheet section 3.3.2), so we ground it to disable the wakeup functionality.

Ensure that the default setting is 100 (Hi-Z)

**Table 14. Serial Communication Block Configuration**

| Pin | Serial Port | Mode 0* | Mode 1 | Mode 2 | Mode 3 | Mode 4 | Mode 5 | Mode 6 |
|---|---|---|---|---|---|---|---|---|
| | | 6-pin UART | 4-pin UART | 2-pin UART | SPI Master | SPI Slave | I2C Master | I2C Slave |
| 1 | SCB_0 | RxD | RxD | RxD | GPIO_6 | GPIO_6 | GPIO_6 | GPIO_6 |
| 20 | SCB_1 | DSR# | GPIO_2 | GPIO_2 | SSEL_OUT | SSEL_IN | GPIO_2 | GPIO_2 |
| 21 | SCB_2 | RTS# | RTS# | GPIO_3 | MISO_IN | MISO_OUT | SCL_OUT | SCL_IN |
| 22 | SCB_3 | CTS# | CTS# | GPIO_4 | MOSI_OUT | MOSI_IN | SDA | SDA |
| 23 | SCB_4 | TxD | TxD | TxD | SCLK_OUT | SCLK_IN | GPIO_5 | GPIO_5 |
| 2 | SCB_5 | DTR# | GPIO_7 | GPIO_7 | GPIO_7 | GPIO_7 | GPIO_7 | GPIO_7 |

*Note: The device is configured in Mode 0 as the default. Other modes can be configured using the configuration utility provided by Cypress.

GPIO
SCB

Note: for serial flow control we connect UART.RTS (from the MCU) to ~CTS from the CY7C65211. This allows the Mbed MCU to tell the serial adapter to stop sending by taking its RTS pin high (deasserted).
As for UART.CTS from the MCU, that is looped back to another GPIO on the ArduinoConnector sheet. We can use that GPIO to control whether the UART peripheral is seeing CTS asserted or deasserted.

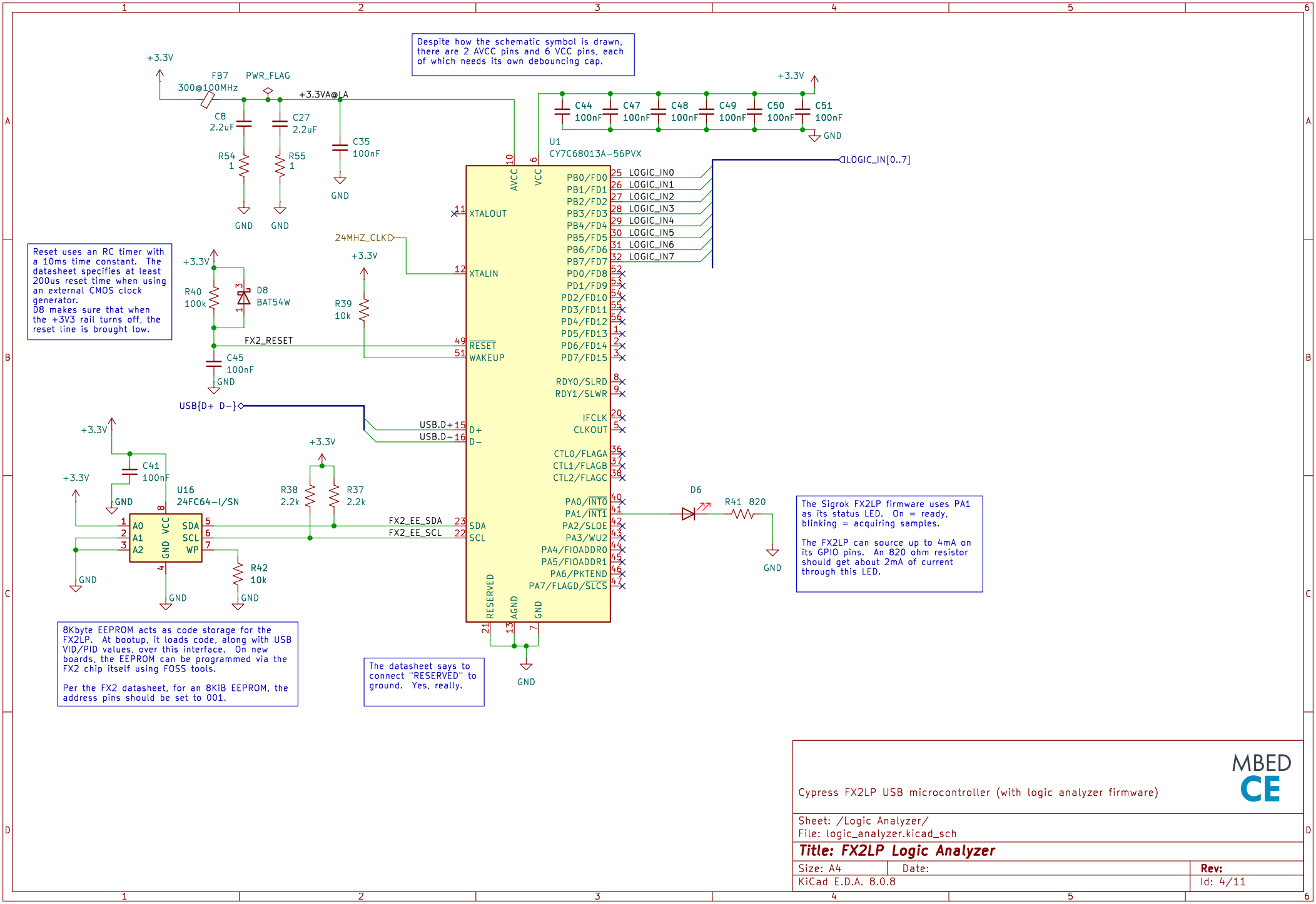
CY7C65211-based USB to UART/SPI/I2C converter

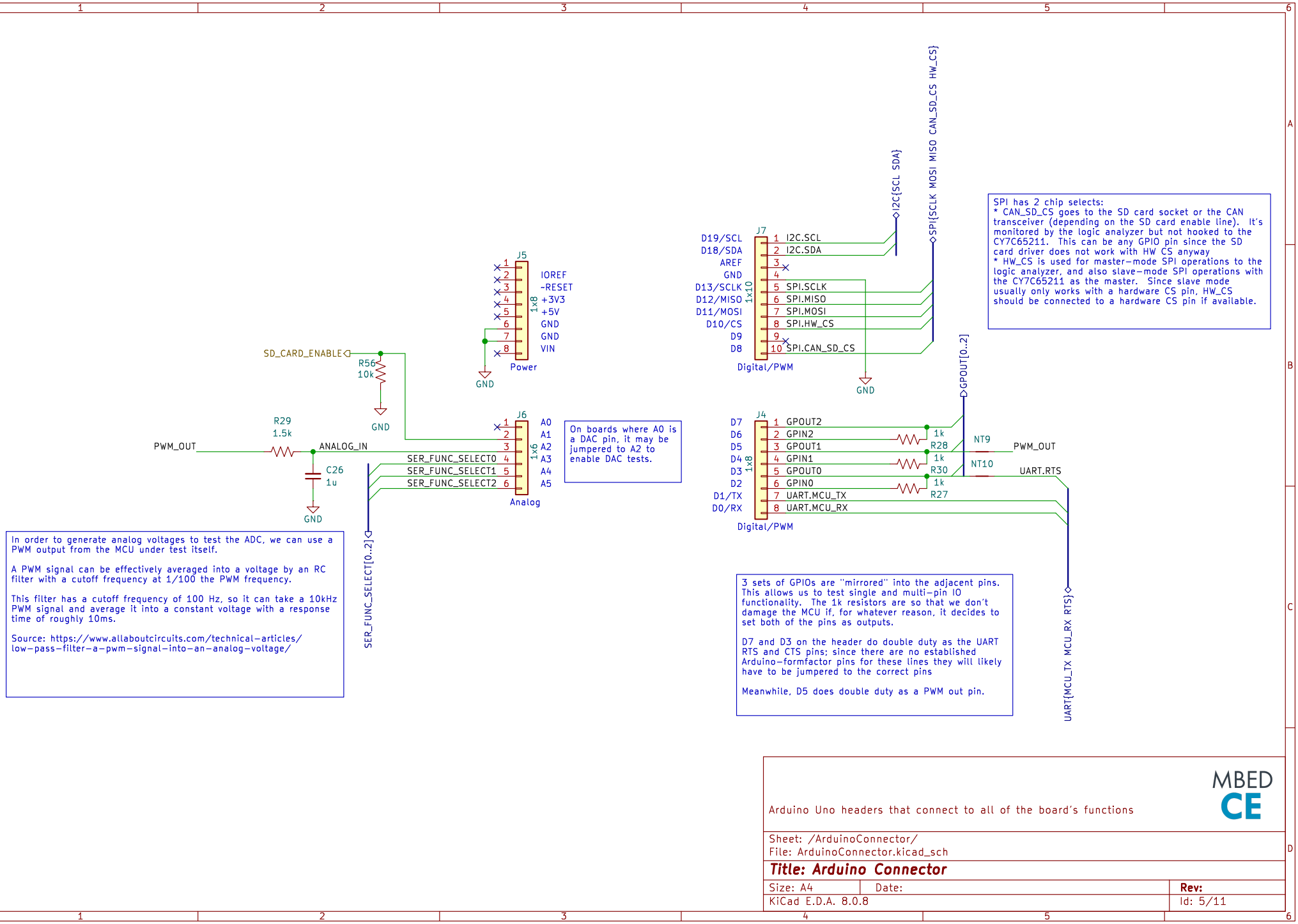
MBED CE

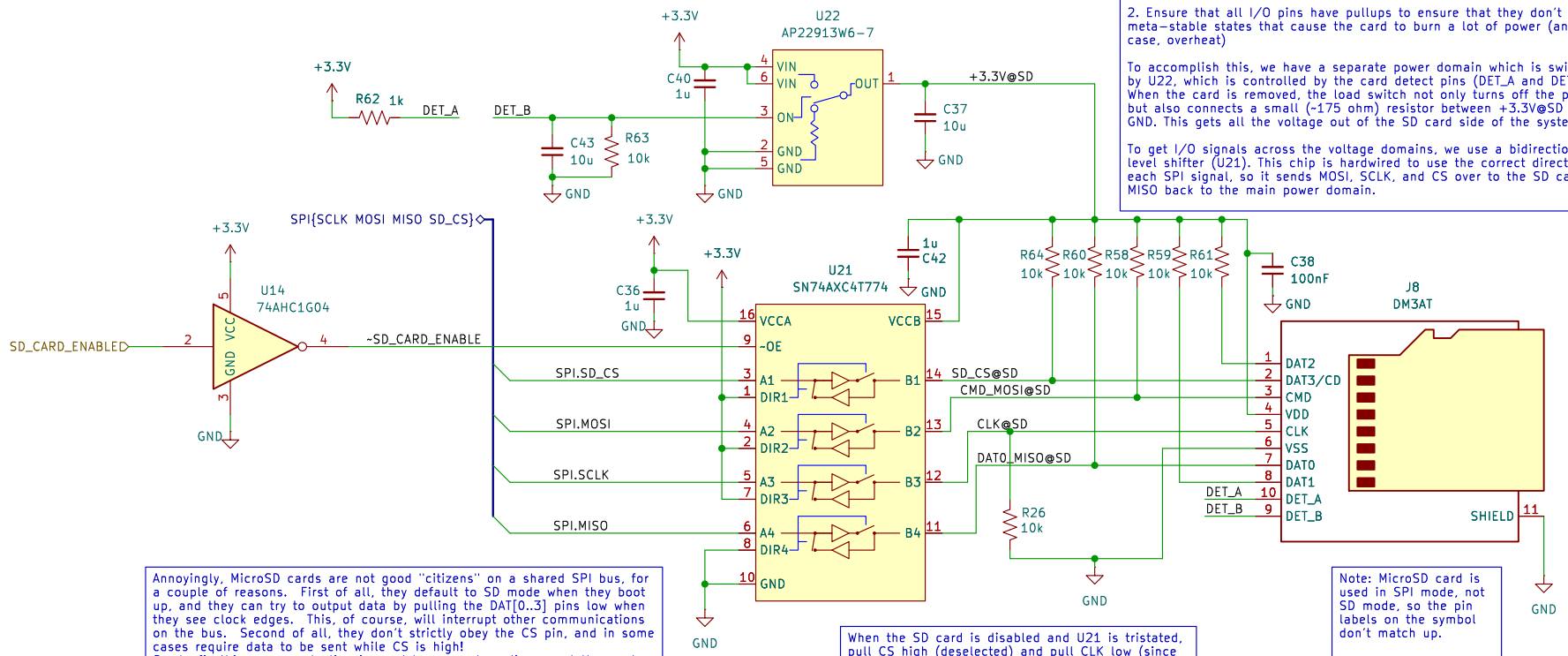
Sheet: /USBSerialAdapter/
File: USBSerialAdapter.kicad_sch

**Title:**

| Size: A3 | Date: | **Rev:** |
|---|---|---|
| KiCad E.D.A. 8.0.8 | | Id: 3/11 |


Despite how the schematic symbol is drawn, there are 2 AVCC pins and 6 VCC pins, each of which needs its own debouncing cap.
+3.3V
FB7
300@100MHz
PWR_FLAG
+3.3VA@LA
C8
2.2uF
C27
2.2uF
C35
100nF
R54
1
R55
1
GND
GND
GND
C44
100nF
C47
100nF
C48
100nF
C49
100nF
C50
100nF
C51
100nF
+3.3V
GND
U1
CY7C68013A-56PVX
AVCC
VCC
XTALOUT
XTALIN
RESET
WAKEUP
D+
D-
RDY0/SLRD
RDY1/SLWR
IFCLK
CLKOUT
CTL0/FLAGA
CTL1/FLAGB
CTL2/FLAGC
PA0/INT0
PA1/INT1
PA2/SLOE
PA3/WU2
PA4/FIOADDR0
PA5/FIOADDR1
PA6/PKTEND
PA7/FLAGD/SLCS
SDA
SCL
RESERVED
AGND
GND
PB0/FD0
PB1/FD1
PB2/FD2
PB3/FD3
PB4/FD4
PB5/FD5
PB6/FD6
PB7/FD7
PD0/FD8
PD1/FD9
PD2/FD10
PD3/FD11
PD4/FD12
PD5/FD13
PD6/FD14
PD7/FD15
LOGIC_IN0
LOGIC_IN1
LOGIC_IN2
LOGIC_IN3
LOGIC_IN4
LOGIC_IN5
LOGIC_IN6
LOGIC_IN7
LOGIC_IN[0..7]
24MHZ_CLK
+3.3V
R39
10k
Reset uses an RC timer with a 10ms time constant. The datasheet specifies at least 200us reset time when using an external CMOS clock generator.
D8 makes sure that when the +3V3 rail turns off, the reset line is brought low.
+3.3V
R40
100k
D8
BAT54W
FX2_RESET
C45
100nF
GND
USB{D+ D-}
USB.D+
USB.D-
+3.3V
C41
100nF
GND
+3.3V
U16
24FC64-I/SN
A0
A1
A2
GND
VCC
SDA
SCL
WP
R38
2.2k
R37
2.2k
FX2_EE_SDA
FX2_EE_SCL
R42
10k
GND
GND
GND
D6
R41 820
GND
The Sigrok FX2LP firmware uses PA1 as its status LED. On = ready, blinking = acquiring samples.
The FX2LP can source up to 4mA on its GPIO pins. An 820 ohm resistor should get about 2mA of current through this LED.
8Kbyte EEPROM acts as code storage for the FX2LP. At bootup, it loads code, along with USB VID/PID values, over this interface. On new boards, the EEPROM can be programmed via the FX2 chip itself using FOSS tools.
Per the FX2 datasheet, for an 8KiB EEPROM, the address pins should be set to 001.
The datasheet says to connect "RESERVED" to ground. Yes, really.
GND
MBED
CE
Cypress FX2LP USB microcontroller (with logic analyzer firmware)
Sheet: /Logic Analyzer/
File: logic_analyzer.kicad_sch
Title: FX2LP Logic Analyzer
Size: A4
Date:
Rev:
KiCad E.D.A. 8.0.8
Id: 4/11


J5
IOREF
~RESET
+3V3
+5V
GND
GND
VIN
Power
SD_CARD_ENABLE
R56
10k
GND
R29
1.5k
PWM_OUT
ANALOG_IN
C26
1u
GND
J6
A0
A1
A2
A3
A4
A5
Analog
SER_FUNC_SELECT0
SER_FUNC_SELECT1
SER_FUNC_SELECT2
SER_FUNC_SELECT[0..2]
On boards where A0 is a DAC pin, it may be jumpered to A2 to enable DAC tests.
In order to generate analog voltages to test the ADC, we can use a PWM output from the MCU under test itself.
A PWM signal can be effectively averaged into a voltage by an RC filter with a cutoff frequency at 1/100 the PWM frequency.
This filter has a cutoff frequency of 100 Hz, so it can take a 10kHz PWM signal and average it into a constant voltage with a response time of roughly 10ms.
Source: https://www.allaboutcircuits.com/technical-articles/low-pass-filter-a-pwm-signal-into-an-analog-voltage/
I2C{SCL SDA}
SPI{SCLK MOSI MISO CAN_SD_CS HW_CS}
J7
D19/SCL
D18/SDA
AREF
GND
D13/SCLK
D12/MISO
D11/MOSI
D10/CS
D9
D8
I2C.SCL
I2C.SDA
SPI.SCLK
SPI.MISO
SPI.MOSI
SPI.HW_CS
SPI.CAN_SD_CS
Digital/PWM
GND
SPI has 2 chip selects:
* CAN_SD_CS goes to the SD card socket or the CAN transceiver (depending on the SD card enable line). It's monitored by the logic analyzer but not hooked to the CY7C65211. This can be any GPIO pin since the SD card driver does not work with HW CS anyway
* HW_CS is used for master-mode SPI operations to the logic analyzer, and also slave-mode SPI operations with the CY7C65211 as the master. Since slave mode usually only works with a hardware CS pin, HW_CS should be connected to a hardware CS pin if available.
GPOUT[0..2]
J4
D7
D6
D5
D4
D3
D2
D1/TX
D0/RX
GPOUT2
GPIN2
GPOUT1
GPIN1
GPOUT0
GPIN0
UART.MCU_TX
UART.MCU_RX
Digital/PWM
1k
R28
1k
R30
1k
R27
NT9
NT10
PWM_OUT
UART.RTS
UART{MCU_TX MCU_RX RTS}
3 sets of GPIOs are "mirrored" into the adjacent pins. This allows us to test single and multi-pin IO functionality. The 1k resistors are so that we don't damage the MCU if, for whatever reason, it decides to set both of the pins as outputs.
D7 and D3 on the header do double duty as the UART RTS and CTS pins; since there are no established Arduino-formfactor pins for these lines they will likely have to be jumpered to the correct pins
Meanwhile, D5 does double duty as a PWM out pin.
MBED
CE
Arduino Uno headers that connect to all of the board's functions
Sheet: /ArduinoConnector/
File: ArduinoConnector.kicad_sch
Title: Arduino Connector
Size: A4
Date:
Rev:
KiCad E.D.A. 8.0.8
Id: 5/11

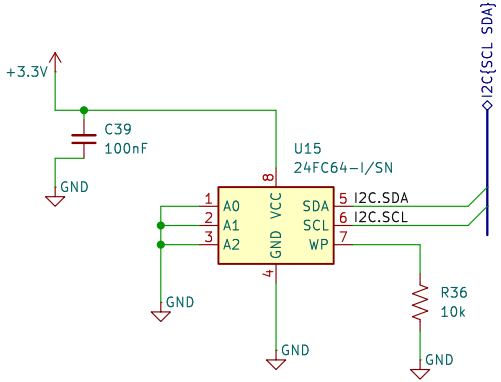



MBED CE

Sheet: /MicroSDCard/
File: MicroSDCard.kicad_sch

**Title: MicroSD Card**

| Size: A4 | Date: | Rev: |
|---|---|---|
| KiCad E.D.A. 8.0.8 | | Id: 6/11 |




This I2C EEPROM provides something to test an MCU's I2C implementation against. There are huge numbers of I2C EEPROMs out there, but I chose this specific one because it supports all three bus speeds in common use: 100kHz, 400kHz, and 1MHz (Fast Mode Plus). Additionally, it's cheap, seems reasonably available, and worked on the previous revision of the board.

If this specific part becomes hard to find, there should be other pin-compatible options, as this form factor of EEPROM is sort of a de-facto standard

EEPROM for the board to talk to over I2C

MBED CE

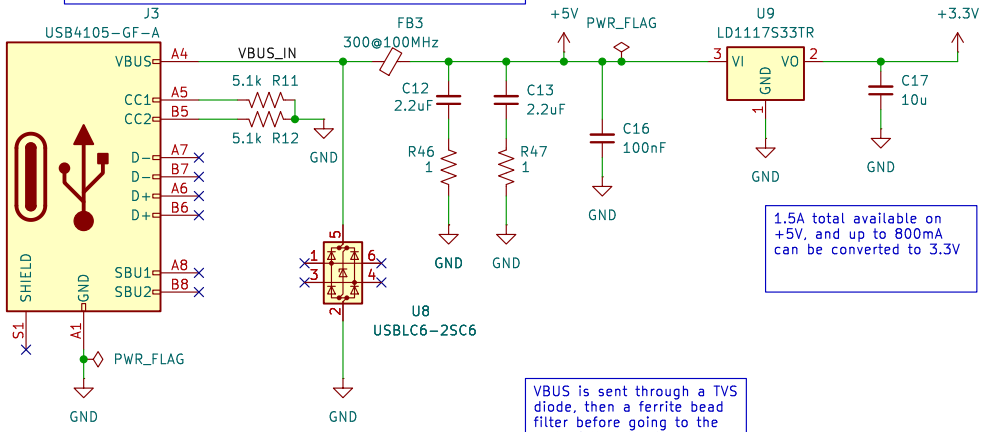
Sheet: /I2CEEPROM/
File: I2CEEPROM.kicad_sch

**Title: I2C EEPROM**

| Size: A5 | Date: | **Rev:** |
|---|---|---|
| KiCad E.D.A. 8.0.8 | | Id: 7/11 |

# Power Input USB Port

This port supplies power to the board but not data. A separate port is used to avoid demanding too much power from the host, which might be a single-board computer without too much power to share with USB devices.



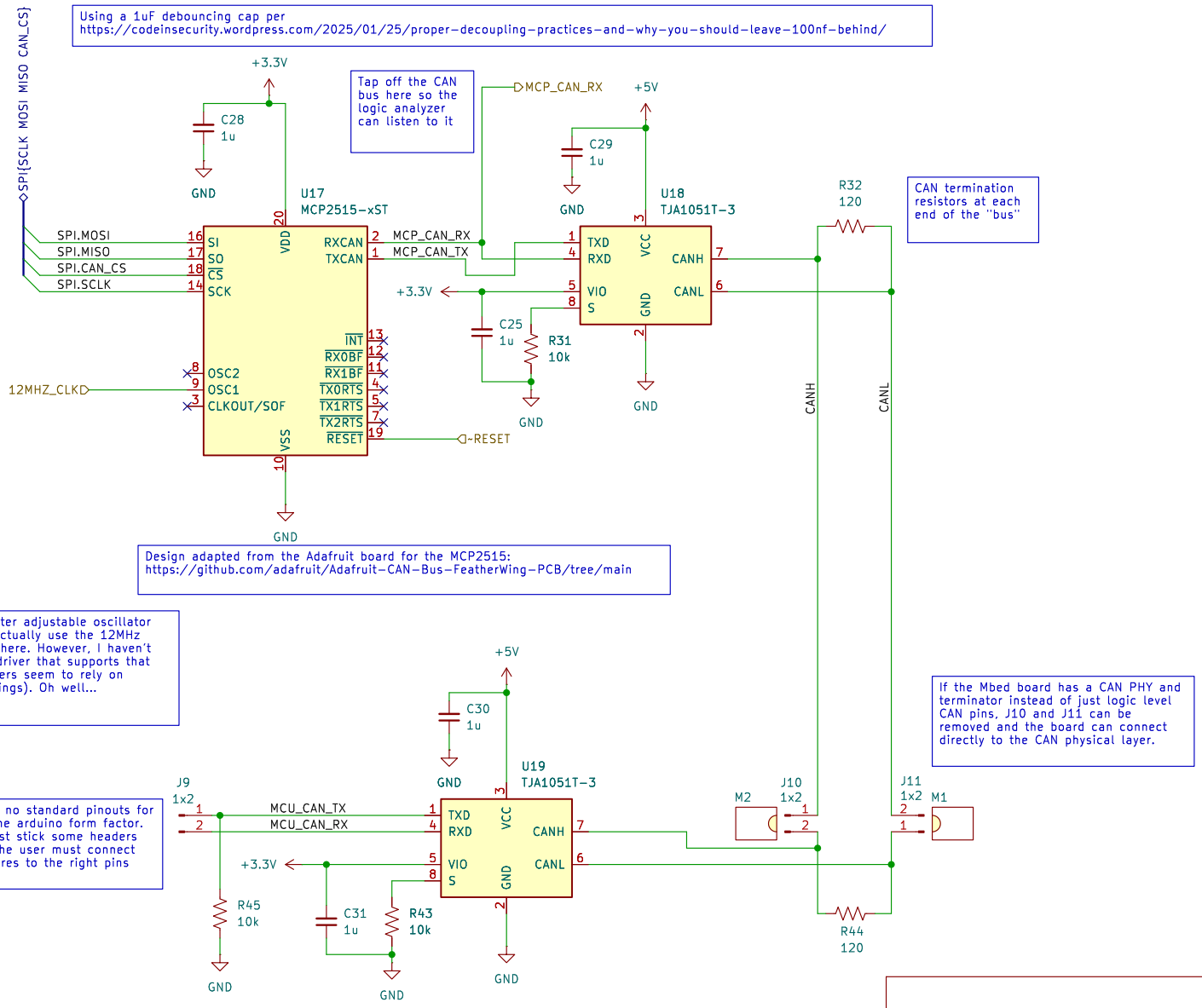

Power supply for the board using a second (power only) USB port.
Intended to connect to a multi port USB power brick.

MBED CE

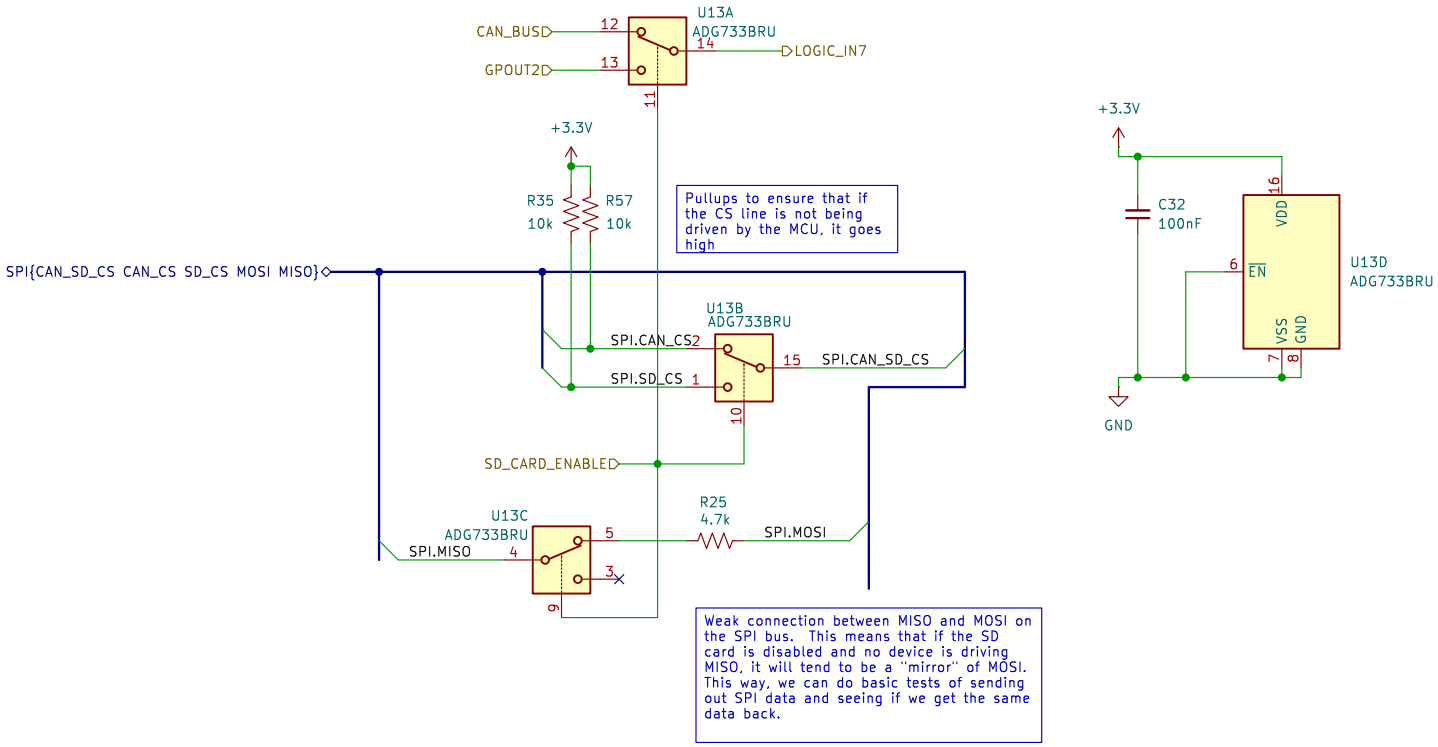
Sheet: /PowerUSBPort/
File: PowerUSBPort.kicad_sch

**Title: Power Input USB Port**

| Size: A4 | Date: | Rev: |
|---|---|---|
| KiCad E.D.A. 8.0.8 | | Id: 8/11 |


Using a 1uF debouncing cap per https://codeinsecurity.wordpress.com/2025/01/25/proper-decoupling-practices-and-why-you-should-leave-100nf-behind/
SPI{SCLK MOSI MISO CAN_CS}
+3.3V
C28
1u
GND
U17
MCP2515-xST
Tap off the CAN bus here so the logic analyzer can listen to it
MCP_CAN_RX
+5V
C29
1u
GND
U18
TJA1051T-3
R32
120
CAN termination resistors at each end of the "bus"
SPI.MOSI
SPI.MISO
SPI.CAN_CS
SPI.SCLK
SI
SO
CS
SCK
VDD
RXCAN
TXCAN
MCP_CAN_RX
MCP_CAN_TX
TXD
RXD
VCC
CANH
VIO
S
GND
CANL
+3.3V
C25
1u
R31
10k
GND
GND
CANH
CANL
INT
RX0BF
RX1BF
TX0RTS
TX1RTS
TX2RTS
RESET
~RESET
12MHZ_CLKD
OSC2
OSC1
CLKOUT/SOF
VSS
GND
Design adapted from the Adafruit board for the MCP2515: https://github.com/adafruit/Adafruit-CAN-Bus-FeatherWing-PCB/tree/main
This chip has a register adjustable oscillator frequency, and can actually use the 12MHz clock available elsewhere. However, I haven't been able to find a driver that supports that clock frequency (drivers seem to rely on precalculated bit timings). Oh well...
+5V
C30
1u
GND
U19
TJA1051T-3
If the Mbed board has a CAN PHY and terminator instead of just logic level CAN pins, J10 and J11 can be removed and the board can connect directly to the CAN physical layer.
J9
1x2
J10
1x2
J11
1x2
M2
M1
There are no standard pinouts for CAN on the arduino form factor. So, we just stick some headers on, and the user must connect jumper wires to the right pins
MCU_CAN_TX
MCU_CAN_RX
TXD
RXD
VCC
CANH
VIO
S
GND
CANL
+3.3V
R45
10k
C31
1u
R43
10k
R44
120
GND
GND
GND
Miniature CAN bus with an SPI-CAN adapter and a second CAN PHY for the MCU to use.
MBED CE
Sheet: /CAN Transceiver/
File: CANTransceiver.kicad_sch
Title: CAN Transceiver
Size: A4
Date:
Rev:
KiCad E.D.A. 8.0.8
Id: 9/11

This sheet is responsible for:
* Switching CAN_SD_CS to connect to either the SD card or the CAN transceiver CS line
* Creating input 7 of the logic analyzer by switching between GPOUT2 and CAN_LISTEN
* Switching SPI.MISO to connect to the mirror resistor



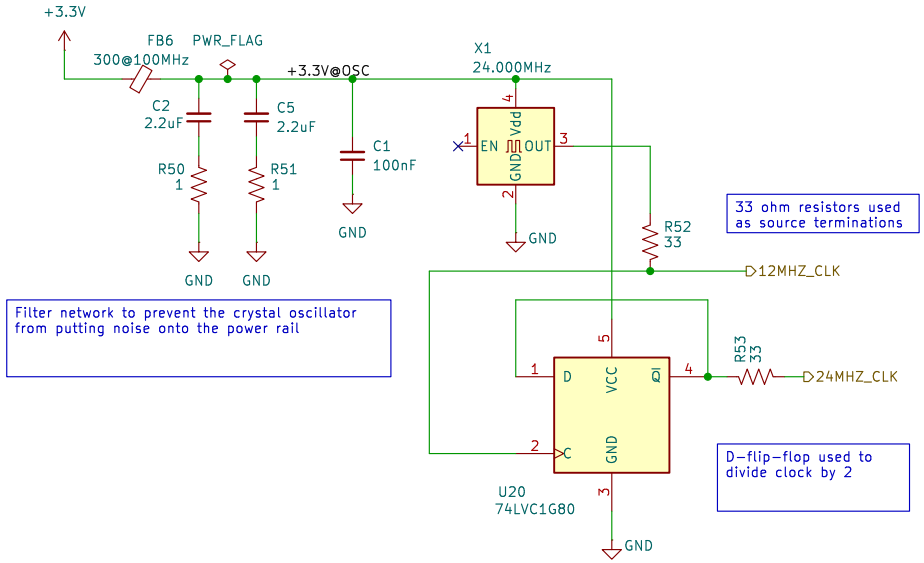

Mux for selecting CAN or SD signals.

MBED CE

Sheet: /CAN – SD Mux/
File: CAN_SD_Mux.kicad_sch

**Title:**

| Size: A4 | Date: | **Rev:** |
|---|---|---|
| KiCad E.D.A. 8.0.8 | | Id: 10/11 |




MBED CE

Sheet: /Oscillators/
File: Oscillators.kicad_sch

**Title:**

| Size: A4 | Date: | **Rev:** |
|---|---|---|
| KiCad E.D.A. 8.0.8 | | Id: 11/11 |