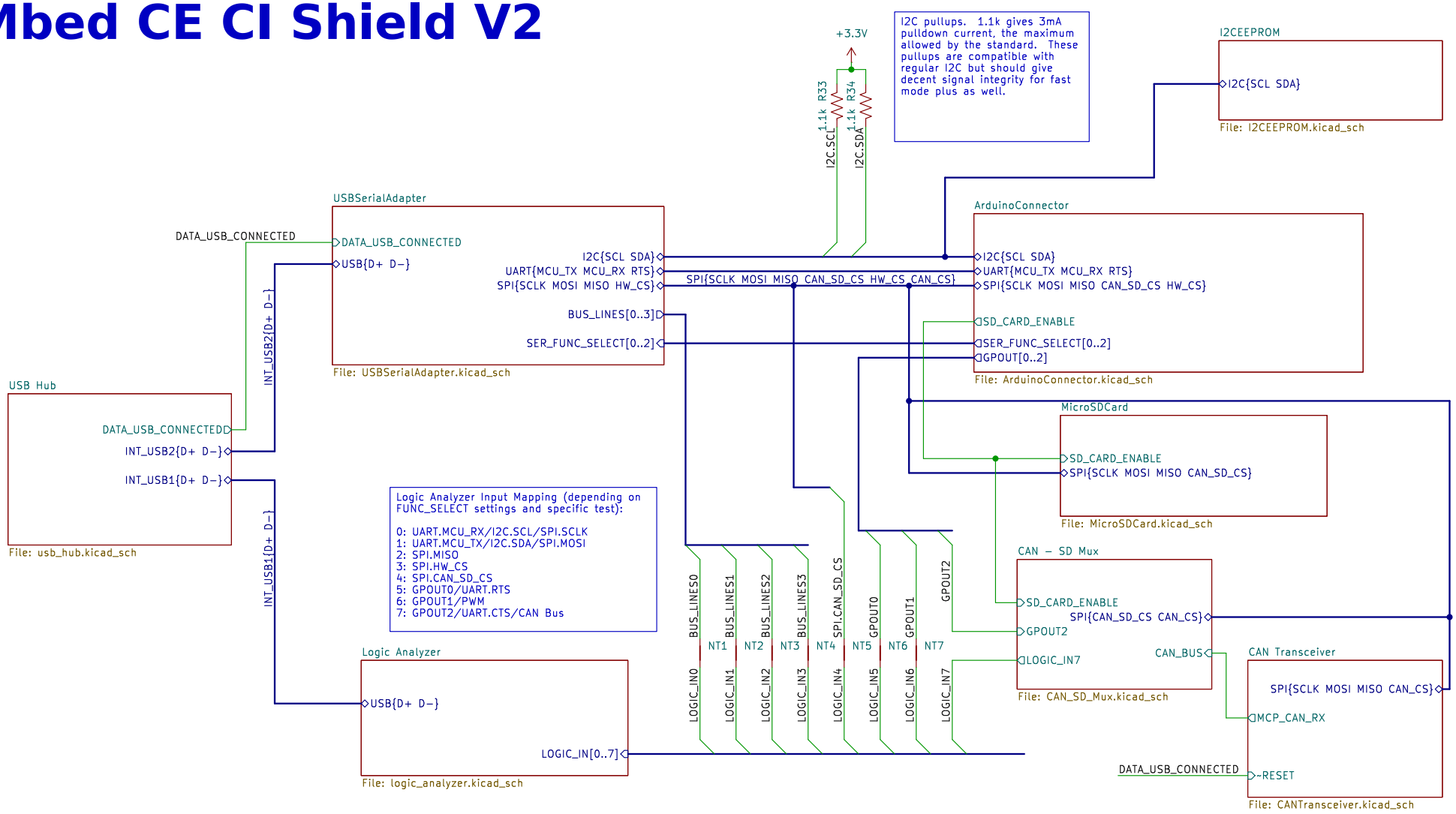
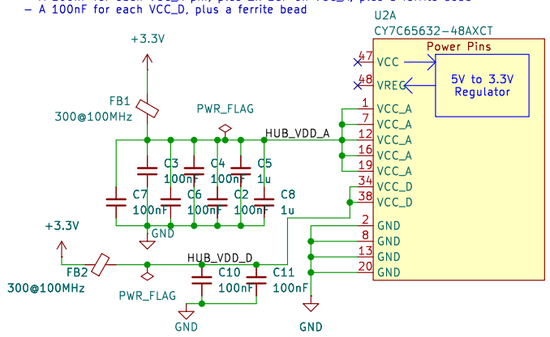
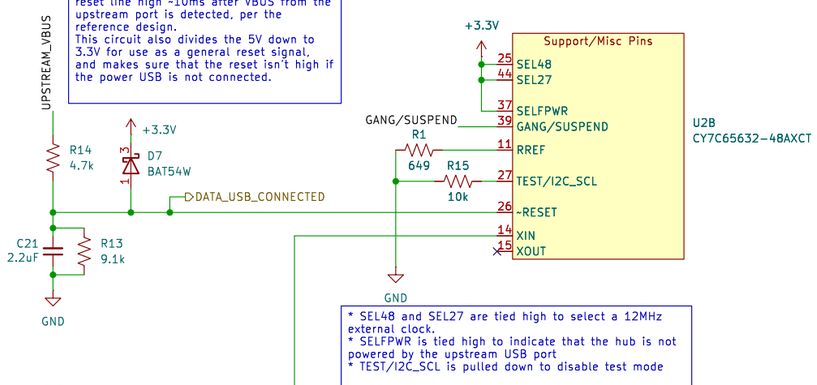
Mbed CE CI Shield V2



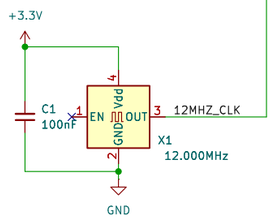
The reference design seems to recommend:
– 1x 10uF and 1x 1uF on VREG (which I assume is needed only if the internal regulator is used)
– A 100nF for each VCC_A pin, plus 2x 1uF on VCC_A, plus a ferrite bead
– A 100nF for each VCC_D, plus a ferrite bead



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.

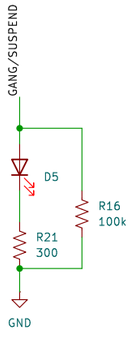


* SEL48 and SEL27 are tied high to select a 12MHz external clock.
* SELFPWR is tied high to indicate that the hub is not powered by the upstream USB port
* TEST/I2C_SCL is pulled down to disable test mode

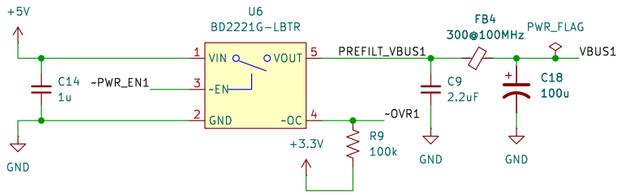


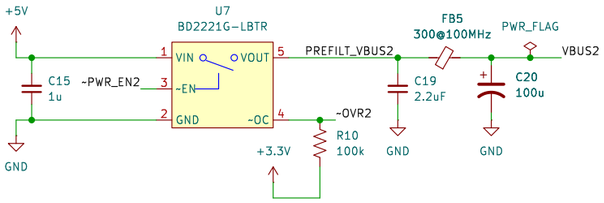
12MHz CMOS Oscillator

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.



External port power switches





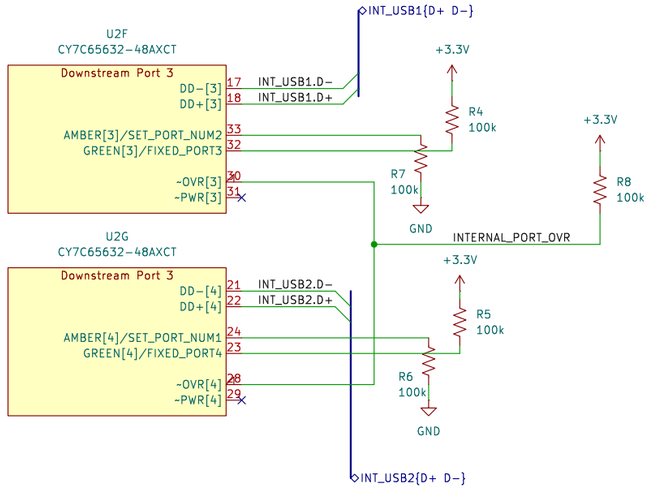
Internal Ports

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.

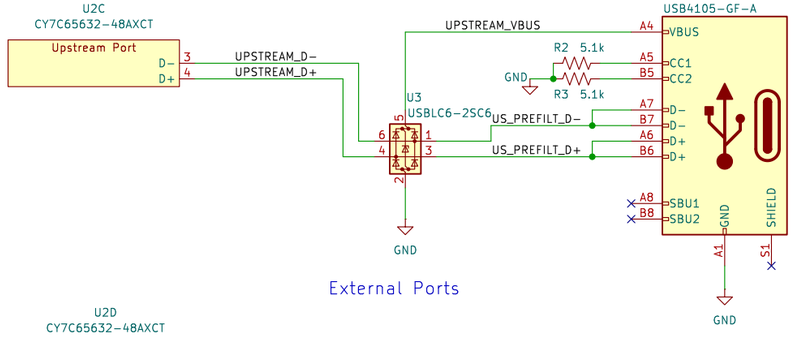
The SET_PORT_NUM[2:1] pins are strapped to 00 to indicate that all 4 ports are in use.

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

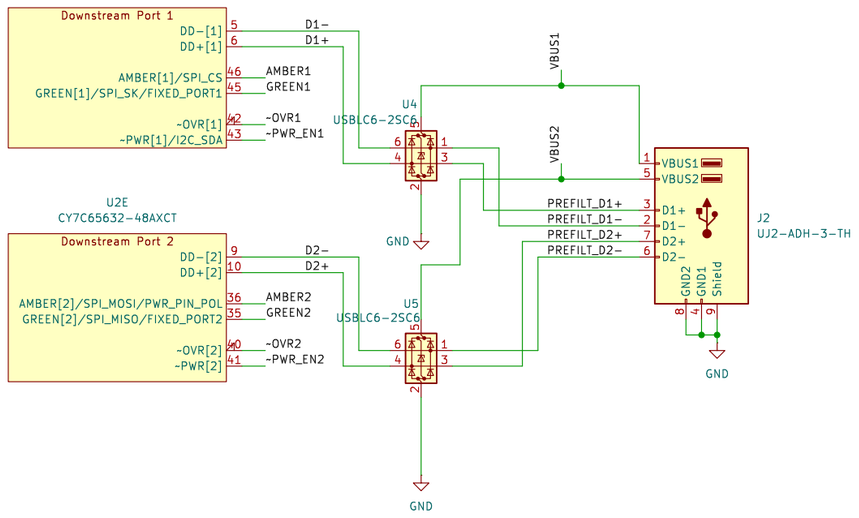


Upstream Port

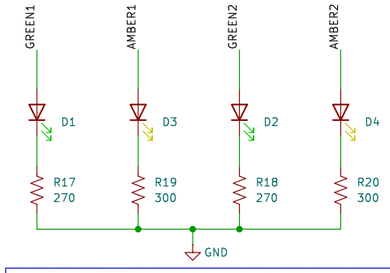
Upstream port (USB-C, connects to host computer).
This uses the USB connector setup from here:
https://hackaday.com/2022/12/06/usb-c-introduction-for-hackers/
This requests 5V@1.5A for our board.



External Ports



Port Indicators



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:
KiCad E.D.A. 8.0.8	Id: 2/10

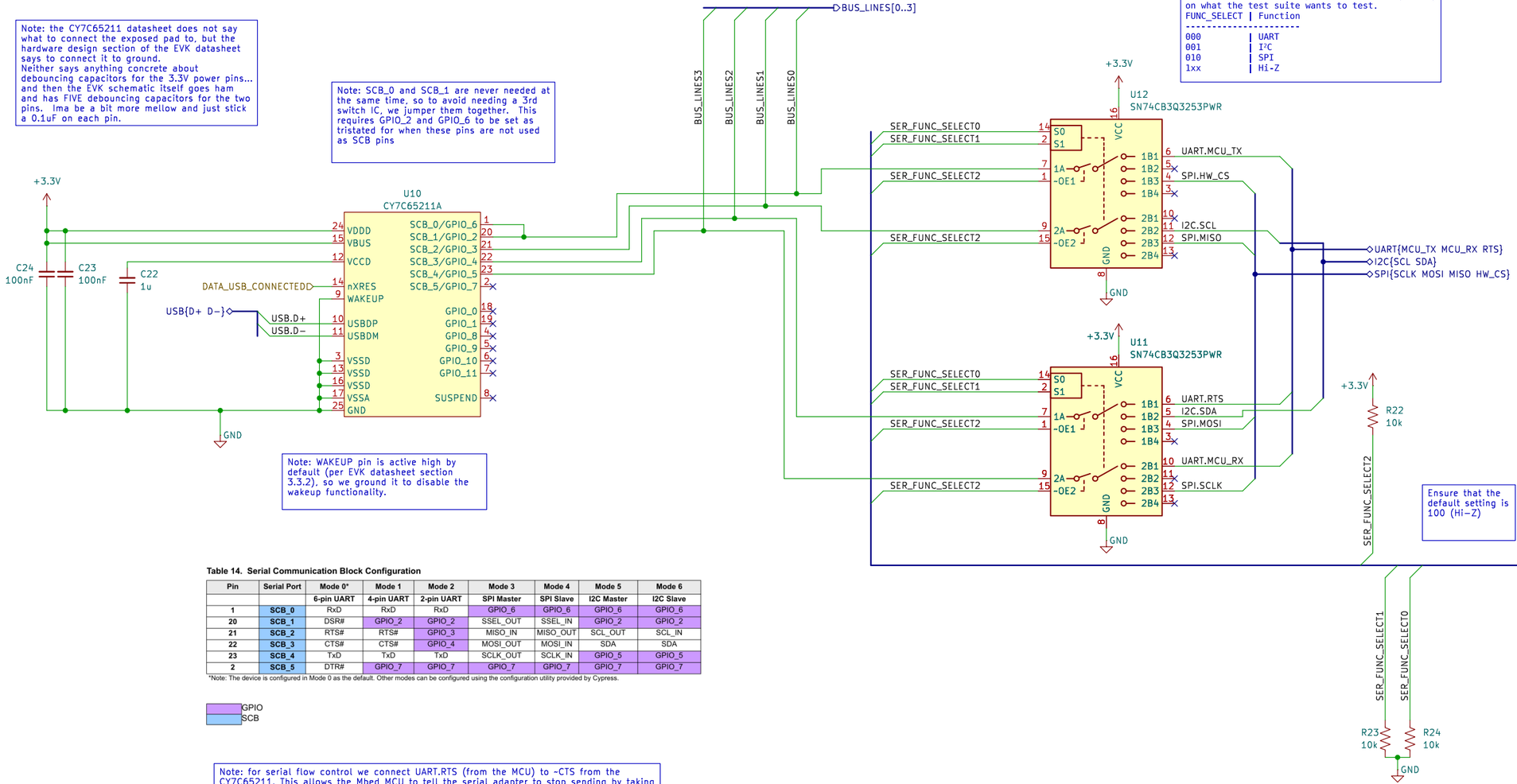
Note: the CY7C65211 datasheet does not say what to connect the exposed pad to, but the hardware design section of the EVK datasheet says to connect it to ground.
Neither says anything concrete about debouncing capacitors for the 3.3V power pins... and then the EVK schematic itself goes ham and has FIVE 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.

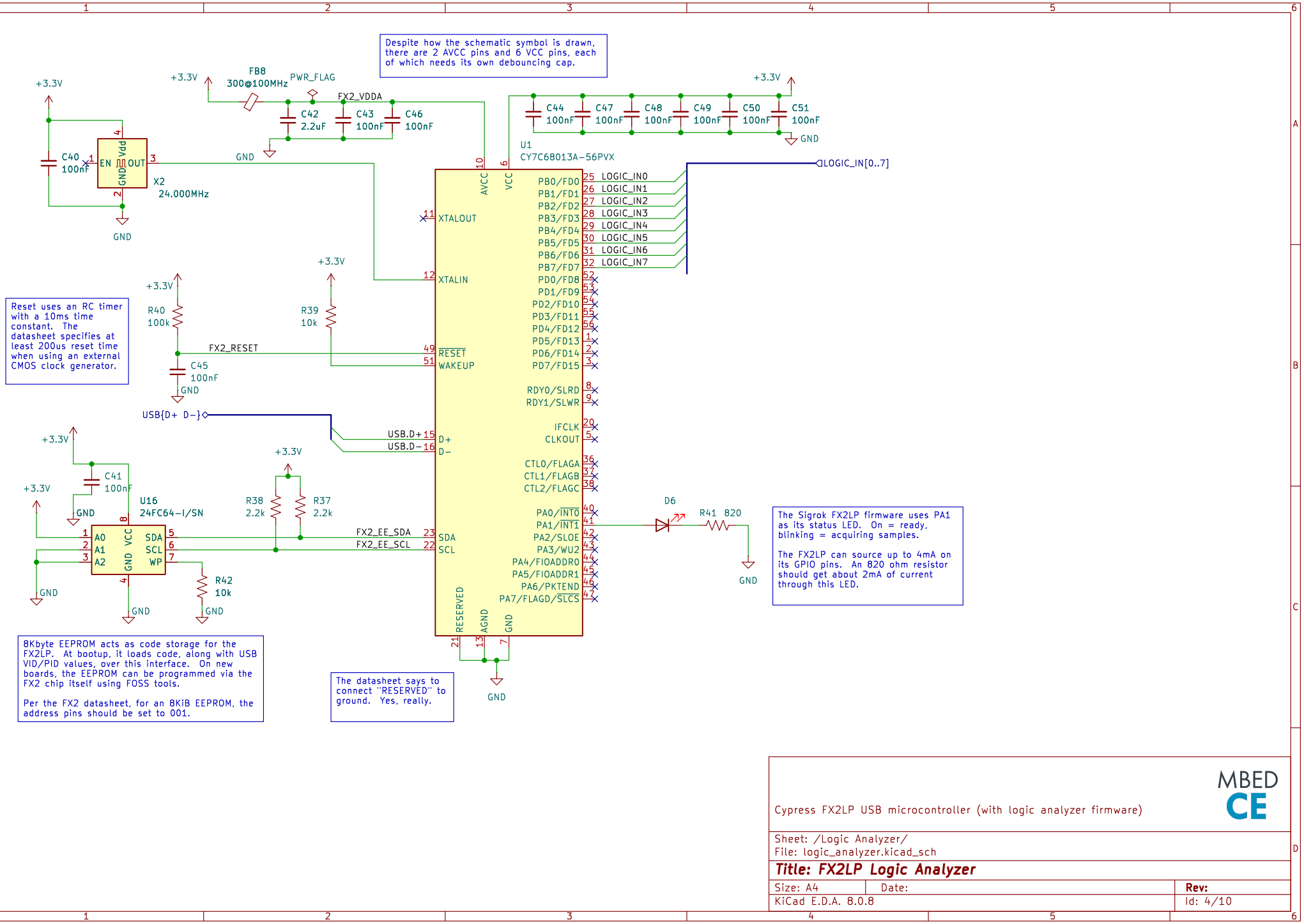
CY7C65211-based USB to UART/SPI/I2C converter

MBED CE

Sheet: /USBSerialAdapter/
File: USBSerialAdapter.kicad_sch

Title:

Size: A3	Date:	**Rev:**
KiCad E.D.A. 8.0.8		Id: 3/10

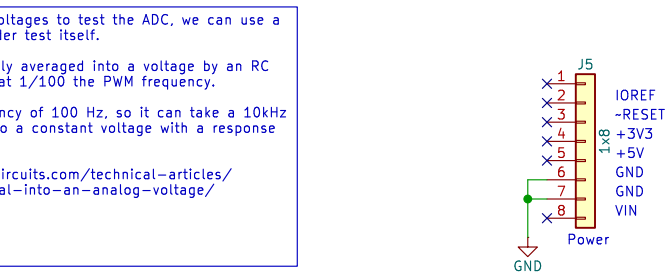

Despite how the schematic symbol is drawn, there are 2 AVCC pins and 6 VCC pins, each of which needs its own debouncing cap.
+3.3V
FB8
300@100MHz
PWR_FLAG
FX2_VDDA
C42
2.2uF
C43
100nF
C46
100nF
C44
100nF
C47
100nF
C48
100nF
C49
100nF
C50
100nF
C51
100nF
GND
U1
CY7C68013A-56PVX
C40
100nF
X2
24.000MHz
LOGIC_IN[0..7]
XTALOUT
XTALIN
AVCC
VCC
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
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.
R40
100k
R39
10k
C45
100nF
FX2_RESET
RESET
WAKEUP
USB{D+ D-}
USB.D+
USB.D-
D+
D-
RDY0/SLRD
RDY1/SLWR
IFCLK
CLKOUT
CTL0/FLAGA
CTL1/FLAGB
CTL2/FLAGC
C41
100nF
U16
24FC64-I/SN
R38
2.2k
R37
2.2k
FX2_EE_SDA
FX2_EE_SCL
SDA
SCL
PA0/INT0
PA1/INT1
PA2/SLOE
PA3/WU2
PA4/FIOADDR0
PA5/FIOADDR1
PA6/PKTEND
PA7/FLAGD/SLCS
D6
R41 820
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.
R42
10k
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.
RESERVED
AGND
The datasheet says to connect "RESERVED" to ground. Yes, really.
Cypress FX2LP USB microcontroller (with logic analyzer firmware)
MBED CE
Sheet: /Logic Analyzer/
File: logic_analyzer.kicad_sch
Title: FX2LP Logic Analyzer
Size: A4
Date:
Rev:
KiCad E.D.A. 8.0.8
Id: 4/10

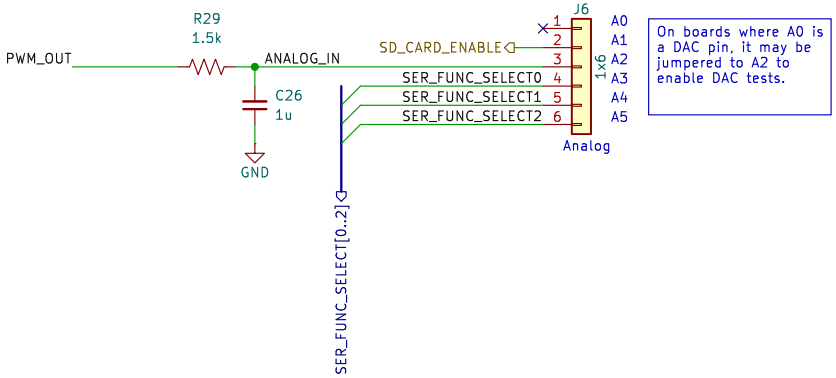
In order to generate analog voltages to test the ADC, we can use a PWM output from the MCU under test itself.

A PWM signal can be effectively averaged into a voltage by an RC filter with a cutoff frequency at 1/100 the PWM frequency.

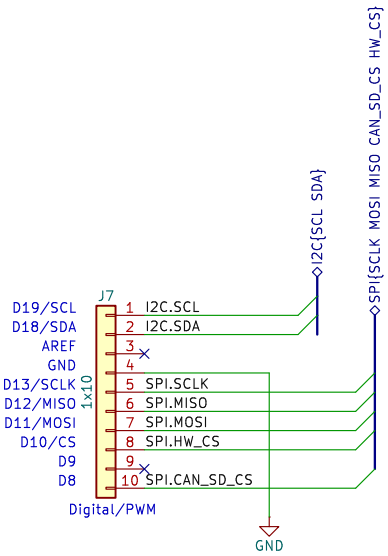
This filter has a cutoff frequency of 100 Hz, so it can take a 10kHz PWM signal and average it into a constant voltage with a response time of roughly 10ms.

Source: https://www.allaboutcircuits.com/technical-articles/low-pass-filter-a-pwm-signal-into-an-analog-voltage/

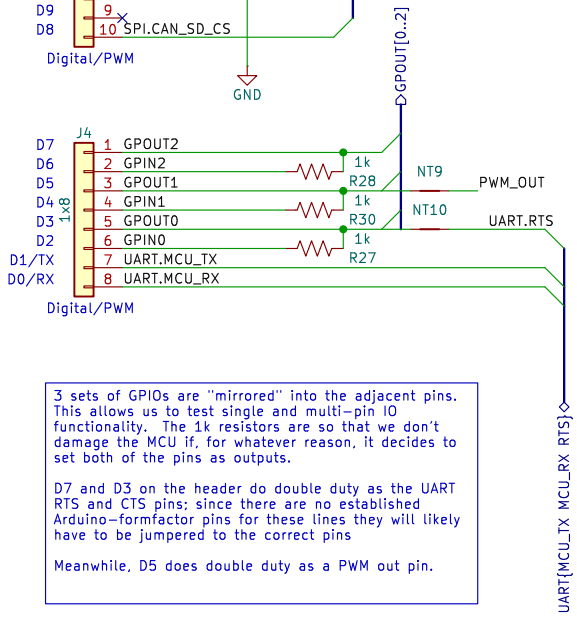




On boards where A0 is a DAC pin, it may be jumpered to A2 to enable DAC tests.



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.



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.

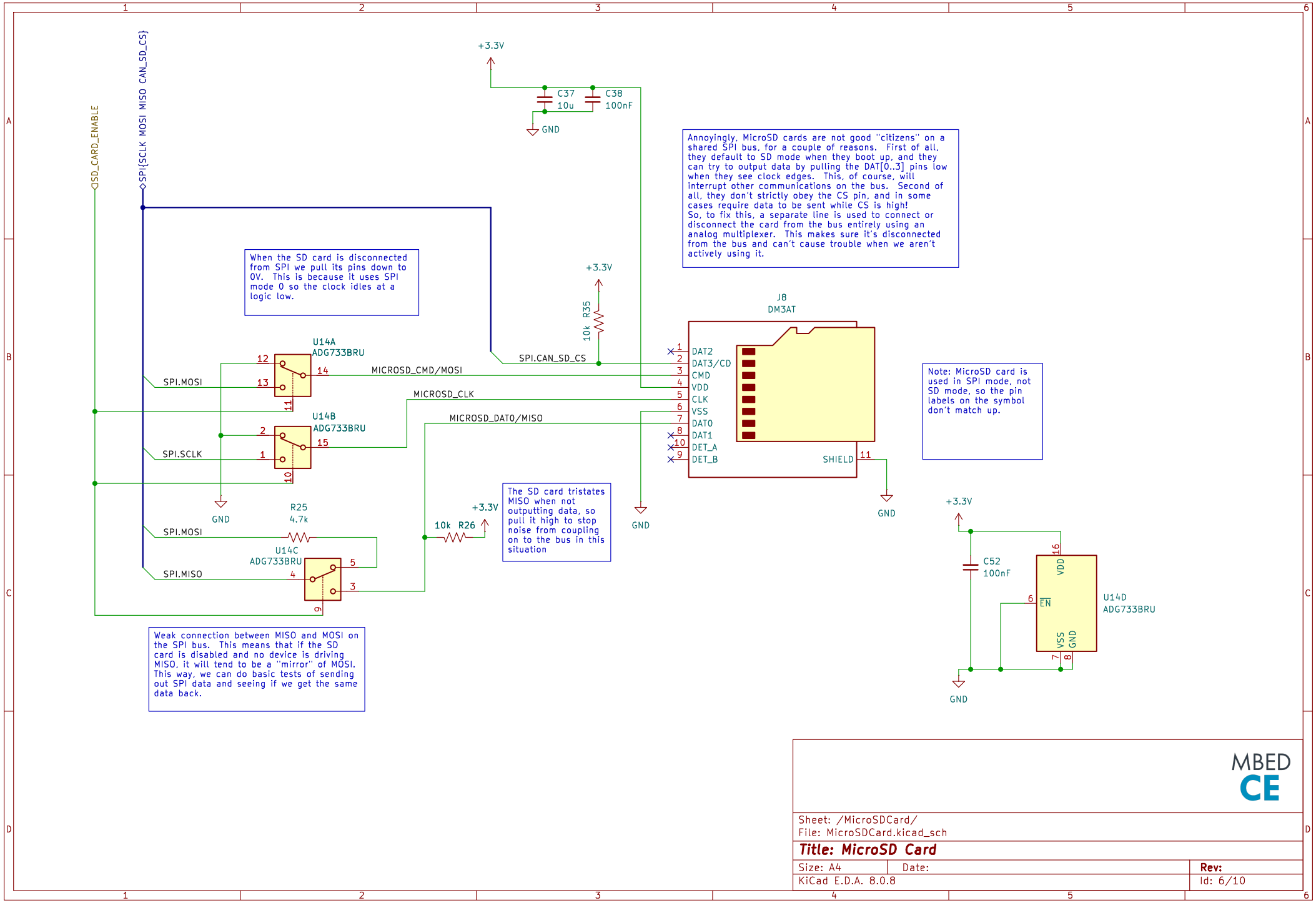
Arduino Uno headers that connect to all of the board's functions

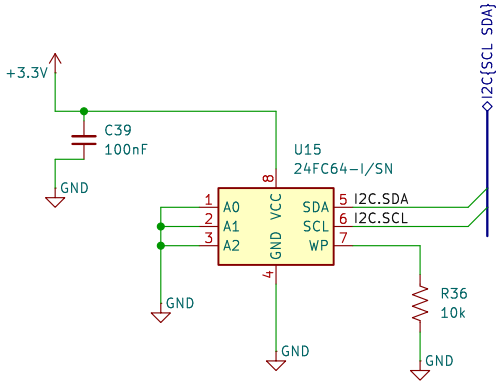
MBED CE

Sheet: /ArduinoConnector/
File: ArduinoConnector.kicad_sch

Title: Arduino Connector

Size: A4	Date:	Rev:
KiCad E.D.A. 8.0.8		Id: 5/10


SD_CARD_ENABLE
SPI{SCLK MOSI MISO CAN_SD_CS}
+3.3V
C37
10u
C38
100nF
GND
Annoyingly, MicroSD cards are not good "citizens" on a shared SPI bus, for a couple of reasons. First of all, they default to SD mode when they boot up, and they can try to output data by pulling the DAT[0..3] pins low when they see clock edges. This, of course, will interrupt other communications on the bus. Second of all, they don't strictly obey the CS pin, and in some cases require data to be sent while CS is high! So, to fix this, a separate line is used to connect or disconnect the card from the bus entirely using an analog multiplexer. This makes sure it's disconnected from the bus and can't cause trouble when we aren't actively using it.
When the SD card is disconnected from SPI we pull its pins down to 0V. This is because it uses SPI mode 0 so the clock idles at a logic low.
+3.3V
R35
10k
J8
DM3AT
U14A
ADG733BRU
SPI.CAN_SD_CS
MICROSD_CMD/MOSI
MICROSD_CLK
MICROSD_DAT0/MISO
SPI.MOSI
SPI.SCLK
SPI.MISO
U14B
ADG733BRU
U14C
ADG733BRU
R25
4.7k
R26
10k
+3.3V
GND
DAT2
DAT3/CD
CMD
VDD
CLK
VSS
DAT0
DAT1
DET_A
DET_B
SHIELD
Note: MicroSD card is used in SPI mode, not SD mode, so the pin labels on the symbol don't match up.
The SD card tristates MISO when not outputting data, so pull it high to stop noise from coupling on to the bus in this situation
Weak connection between MISO and MOSI on the SPI bus. This means that if the SD card is disabled and no device is driving MISO, it will tend to be a "mirror" of MOSI. This way, we can do basic tests of sending out SPI data and seeing if we get the same data back.
C52
100nF
U14D
ADG733BRU
VDD
EN
VSS
GND
MBED
CE
Sheet: /MicroSDCard/
File: MicroSDCard.kicad_sch
Title: MicroSD Card
Size: A4
Date:
Rev:
KiCad E.D.A. 8.0.8
Id: 6/10



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

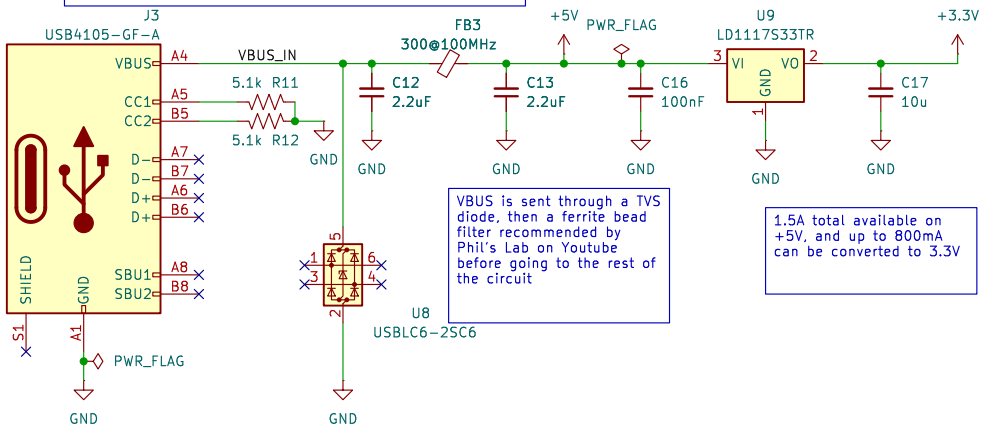
Sheet: /I2CEEPROM/
File: I2CEEPROM.kicad_sch

Title: I2C EEPROM

Size: A5	Date:	**Rev:**
KiCad E.D.A. 8.0.8		Id: 7/10

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.



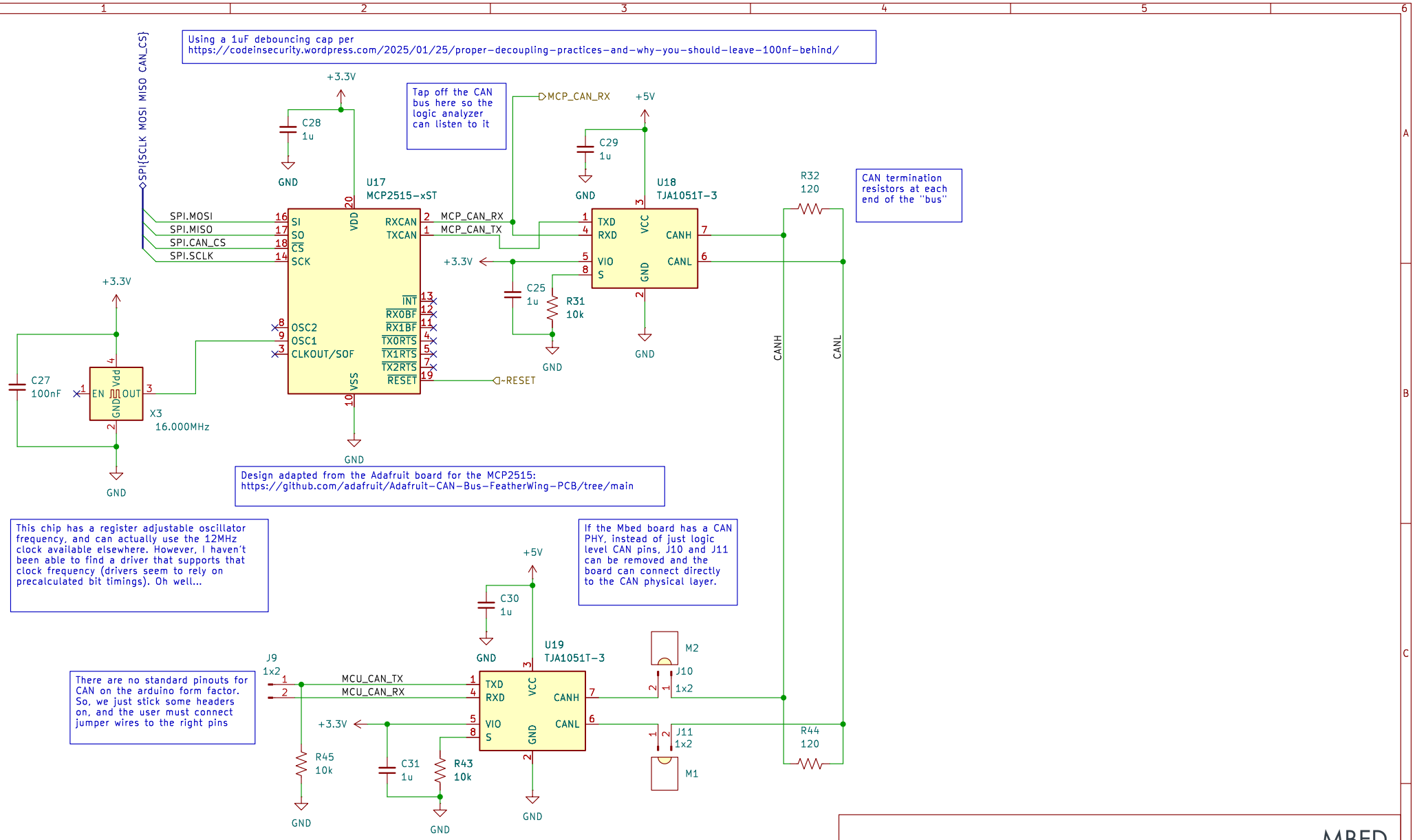
Power supply for the board using a second (power only) USB port.
Intended to connect to a multi port USB power brick.

MBED CE

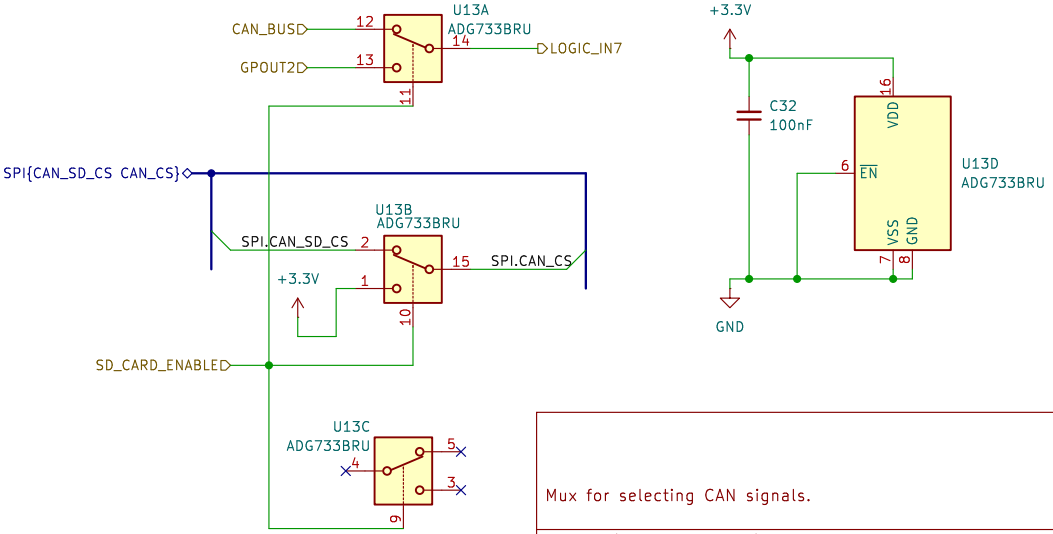
Sheet: /PowerUSBPort/
File: PowerUSBPort.kicad_sch

Title: Power Input USB Port

Size: A4	Date:	Rev:
KiCad E.D.A. 8.0.8		Id: 8/10


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
Tap off the CAN bus here so the logic analyzer can listen to it
MCP_CAN_RX
+5V
C29
1u
GND
U17
MCP2515-xST
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
INT
RX0BF
RX1BF
TX0RTS
TX1RTS
TX2RTS
RESET
~RESET
OSC2
OSC1
CLKOUT/SOF
VSS
C27
100nF
EN
Vdd
OUT
X3
16.000MHz
CANH
CANL
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...
If the Mbed board has a CAN PHY, instead of just logic level CAN pins, J10 and J11 can be removed and the board can connect directly to the CAN physical layer.
C30
1u
U19
TJA1051T-3
J9
1x2
MCU_CAN_TX
MCU_CAN_RX
M2
J10
1x2
J11
1x2
M1
R45
10k
C31
1u
R43
10k
R44
120
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
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: 9/10

This sheet is responsible for:
* Creating CAN_CS based on CAN_SD_CS
* Creating input 7 of the logic analyzer by switching between GPOUT2 and CAN_LISTEN



Mux for selecting CAN signals.

MBED CE

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

Title:

Size: A5	Date:	**Rev:**
KiCad E.D.A. 8.0.8		Id: 10/10