Mbed CE CI Shield V2

Weak connection between MISO and MOSI on the SPI bus. This means that if 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.

I2C pullups. 1.1k gives 3mA pulldown current, the maximum allowed by the standard. These pullups are compatible with regular I2C but should give decent signal integrity for fast mode plus as well.

+3.3V

SPI.MISO — SPI.MOSI
10k R32

I2C.SCL 1.1k R33
I2C.SDA 1.1k R34

I2CEEPROM
I2C{SCL SDA}
File: I2CEEPROM.kicad_sch

USBSerialAdapter
DATA_USB_CONNECTED
USB{D+ D-}

I2C{SCL SDA}
UART{MCU_TX MCU_RX RTS CTS}
SPI{SCLK MOSI MISO SD_CS HW_CS}

BUS_LINES[0..3]

SER_FUNC_SELECT[0..2]

File: USBSerialAdapter.kicad_sch

INT_USB1{D+ D-}

ArduinoConnector
I2C{SCL SDA}
UART{MCU_TX MCU_RX RTS CTS}
SPI{SCLK MOSI MISO SD_CS HW_CS}

SD_CARD_ENABLE
SER_FUNC_SELECT[0..2]
GPOUT[0..2]

File: ArduinoConnector.kicad_sch

USB Hub
DATA_USB_CONNECTED
INT_USB2{D+ D-}
INT_USB1{D+ D-}

File: usb_hub.kicad_sch

MicroSDCard
SD_CARD_ENABLE
SPI{SCLK MOSI MISO SD_CS HW_CS}

File: MicroSDCard.kicad_sch

INT_USB2{D+ D-}

Logic Analyzer Input Mapping (depending on FUNC_SELECT settings and specific test):

0: UART.MCU_RX/I2C.SCL/SPI.SCLK
1: UART.MCU_TX/I2C.SDA/SPI.MOSI
2: UART.RTS/SPI.MISO
3: UART.CTS/SPI.HW_CS
4: SPI.SD_CS
5: GPOUT0
6: GPOUT1/PWM
7: GPOUT2

BUS_LINES0   NT1
BUS_LINES1   NT2
BUS_LINES2   NT3
BUS_LINES3   NT4
SPI.SD_CS   NT5
GPOUT0   NT6
GPOUT1   NT7
GPOUT2   NT8

LOGIC_IN0
LOGIC_IN1
LOGIC_IN2
LOGIC_IN3
LOGIC_IN4
LOGIC_IN5
LOGIC_IN6
LOGIC_IN7

Logic Analyzer
USB{D+ D-}

LOGIC_IN[0..7]

File: logic_analyzer.kicad_sch

PowerUSBPort

File: PowerUSBPort.kicad_sch

LG1

MBED CE

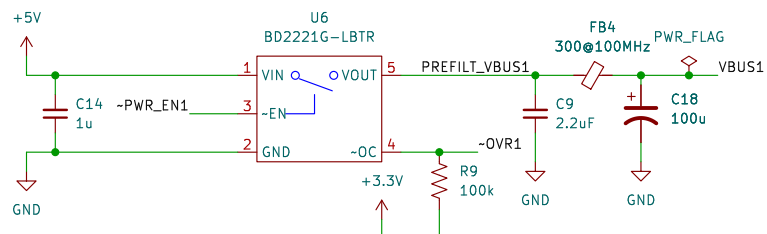MBED CE

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
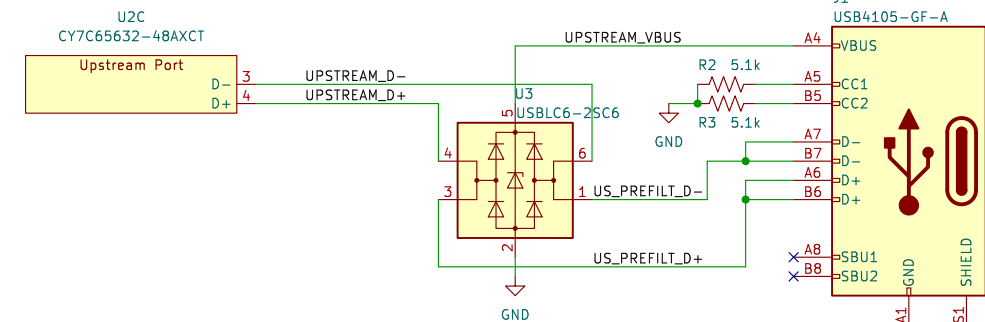– A 100nF for each VCC_D pin, plus a ferrite bead

Upstream Port

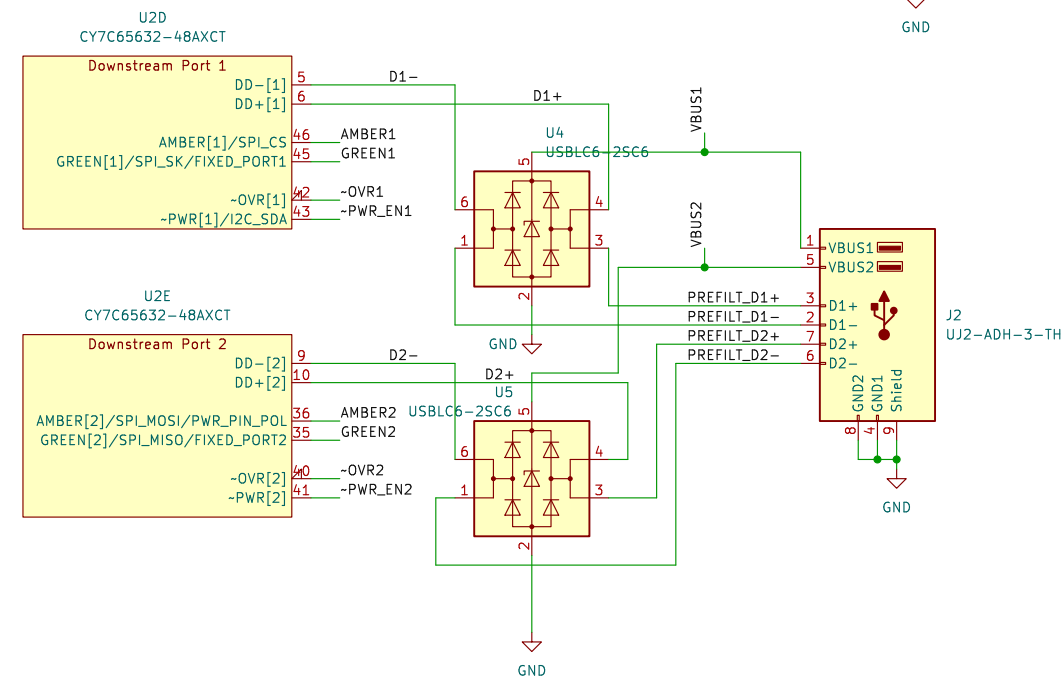Upstream port (USB-C, connects to host computer).
This uses the USB connector setup from here:
https://hackaday.com/2022/12/06/usb-c-introduction-for-hackers/
This requests 5V@1.5A for our board.

Power Pins

+3.3V

FB1
300@100MHz
PWR_FLAG

U2A
CY7C65632-48AXCT

47 VCC
48 VREG
1   VCC_A
7   VCC_A
12 VCC_A
16 VCC_A
19 VCC_A
34 VCC_D
38 VCC_D
2   GND
8   GND
13 GND
20 GND

5V to 3.3V Regulator

HUB_VDD_A

C3 100nF C4 100nF C5 1u
C7 100nF C6 100nF C2 100nF C8 1u

GND

+3.3V

FB2
300@100MHz

HUB_VDD_D
C10 100nF C11 100nF

PWR_FLAG
GND

External port power switches

+5V

U6
BD2221G-LBTR

1 VIN VOUT 5
3 -EN
2 GND -OC 4

C14 -PWR_EN1
1u

+3.3V

PREFILT_VBUS1

FB4
300@100MHz
PWR_FLAG

C9
2.2uF

C18
100u

VBUS1

-OVR1

R9
100k

GND   GND

+5V

U7
BD2221G-LBTR

1 VIN VOUT 5
3 -EN
2 GND -OC 4

C15 -PWR_EN2
1u

+3.3V

PREFILT_VBUS2

FB5
300@100MHz
PWR_FLAG

C19
2.2uF

C20
100u

VBUS2

-OVR2

R10
100k

GND   GND

U2C
CY7C65632-48AXCT

Upstream Port

3 D-
4 D+

UPSTREAM_D-
UPSTREAM_D+

UPSTREAM_VBUS

U3
USBLC6-2SC6

US_PREFILT_D-
US_PREFILT_D+

GND

R2 5.1k
R3 5.1k

J1
USB4105-GF-A

A4 VBUS
A5 CC1
B5 CC2
A7 D-
B7 D-
A6 D+
B6 D+
A8 SBU1
B8 SBU2
A1 GND
S1 SHIELD

GND

External Ports

U2D
CY7C65632-48AXCT

Downstream Port 1

5   DD-[1]
6   DD+[1]
46 AMBER[1]/SPI_CS
45 GREEN[1]/SPI_SK/FIXED_PORT1
42 -OVR[1]
43 -PWR[1]/I2C_SDA

D1-
D1+

AMBER1
GREEN1
-OVR1
-PWR_EN1

U4
USBLC6-2SC6

VBUS1

VBUS2

PREFILT_D1+
PREFILT_D1-
PREFILT_D2+
PREFILT_D2-

GND

U2E
CY7C65632-48AXCT

Downstream Port 2

9   DD-[2]
10 DD+[2]
36 AMBER[2]/SPI_MOSI/PWR_PIN_POL
35 GREEN[2]/SPI_MISO/FIXED_PORT2
40 -OVR[2]
41 -PWR[2]

D2-
D2+

AMBER2
GREEN2
-OVR2
-PWR_EN2

U5
USBLC6-2SC6

GND

J2
UJ2-ADH-3-TH

1 VBUS1
5 VBUS2
2 D1+
3 D1-
7 D2+
6 D2-
8 GND1
4 GND1
9 Shield

GND

UPSTREAM_VBUS

RC timer is used to bring the hub chip's reset line high ~10ms after VBUS from the upstream port is detected. Circuit copied from reference design.

+3.3V

R13
10k

DATA_USB_CONNECTED

C21
1u

R14
47k

GND

Support/Misc Pins

+3.3V

U2B
CY7C65632-48AXCT

25 SEL48
44 SEL27
37 SELFPWR
39 GANG/SUSPEND
11 RREF
27 TEST/I2C_SCL
26 -RESET
14 XIN
15 XOUT

GANG/SUSPEND
R1
649
R15
10k

GND

* 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
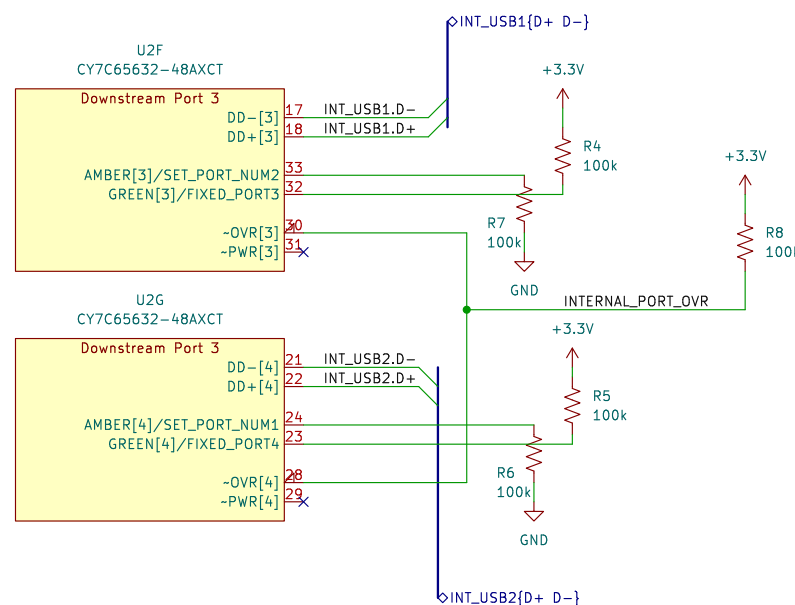* TEST/I2C_SCL is pulled down to disable test mode

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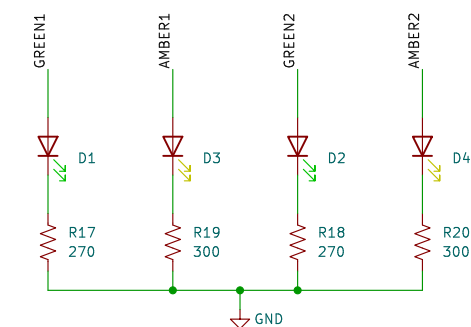
The FIXED_PORT pins are strapped to high to indicate to the hub that these devices are non-removable.

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.

+3.3V

C1
100nF

EN Vdd 4
OUT 3

12MHZ_CLK

GND

X1
12.000MHz

12MHz CMOS Oscillator

GANG/SUSPEND

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.

D5
R16
100k
R21
300

GND

U2F
CY7C65632-48AXCT

Downstream Port 3

17 DD-[3]
18 DD+[3]
33 AMBER[3]/SET_PORT_NUM2
32 GREEN[3]/FIXED_PORT3
30 -OVR[3]
31 -PWR[3]

INT_USB1{D+ D-}

INT_USB1.D-
INT_USB1.D+

+3.3V

R4
100k

R7
100k

GND

+3.3V

R8
100k

INTERNAL_PORT_OVR

U2G
CY7C65632-48AXCT

Downstream Port 3

21 DD-[4]
22 DD+[4]
24 AMBER[4]/SET_PORT_NUM1
23 GREEN[4]/FIXED_PORT4
28 -OVR[4]
29 -PWR[4]

INT_USB2.D-
INT_USB2.D+

+3.3V

R5
100k

R6
100k

GND

INT_USB2{D+ D-}

Port Indicators

GREEN1   AMBER1   GREEN2   AMBER2

D1   D3   D2   D4

R17 270   R19 300   R18 270   R20 300

GND

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.

MBED CE

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: 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

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

FX2LP Logic Analyzer — Schematic

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

+3.3V

FB8 300@100MHz   PWR_FLAG

FX2_VDDA

+3.3V

C42 2.2uF   C43 100nF   C46 100nF

C44 100nF   C47 100nF   C48 100nF   C49 100nF   C50 100nF   C51 100nF   GND

C40 100nF

X2 24.000MHz

GND

EN Vdd OUT   4 / 1 / 3 / 2
GND

GND

U1 CY7C68013A-56PVX

AVCC 10   VCC 6

LOGIC_IN[0..7]

Pin	Signal	Net
25	PB0/FD0	LOGIC_IN0
26	PB1/FD1	LOGIC_IN1
27	PB2/FD2	LOGIC_IN2
28	PB3/FD3	LOGIC_IN3
29	PB4/FD4	LOGIC_IN4
30	PB5/FD5	LOGIC_IN5
31	PB6/FD6	LOGIC_IN6
32	PB7/FD7	LOGIC_IN7

PD0/FD8 52
PD1/FD9 53
PD2/FD10 54
PD3/FD11 55
PD4/FD12 56
PD5/FD13 1
PD6/FD14 2
PD7/FD15 3

11 XTALOUT

12 XTALIN

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.

+3.3V

R40 100k

+3.3V

R39 10k

FX2_RESET

49 RESET
51 WAKEUP

C45 100nF
GND

RDY0/SLRD 8
RDY1/SLWR 9

USB{D+ D-}

USB.D+ 15   D+
USB.D- 16   D-

IFCLK 20
CLKOUT 5

CTL0/FLAGA 36
CTL1/FLAGB 37
CTL2/FLAGC 38

+3.3V

C41 100nF
GND

+3.3V

U16 24FC64-I/SN

+3.3V

R38 2.2k   R37 2.2k

FX2_EE_SDA   23 SDA
FX2_EE_SCL   22 SCL

PA0/INT0 40
PA1/INT1 41
PA2/SLOE 42
PA3/WU2 43
PA4/FIOADDR0 44
PA5/FIOADDR1 45
PA6/PKTEND 46
PA7/FLAGD/SLCS 47

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.

A0 1   GND VCC 8
A1 2
A2 3   SDA 5
SCL 6
WP 7

GND

R42 10k

GND   GND

RESERVED 21   AGND 13   GND 7

GND

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.

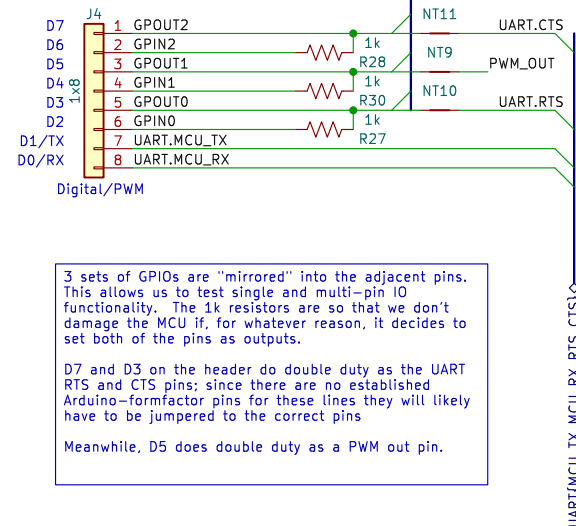**The datasheet says to connect "RESERVED" to ground. Yes, really.**
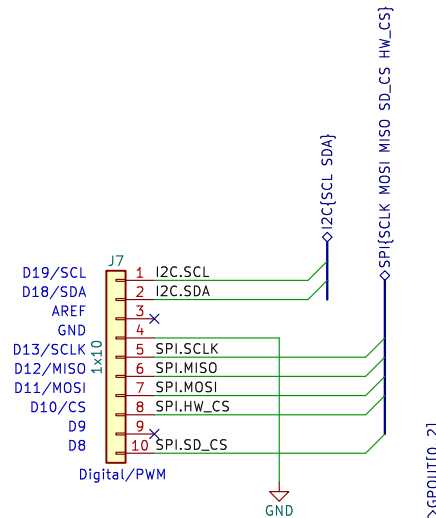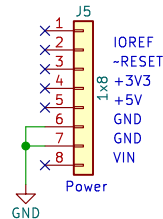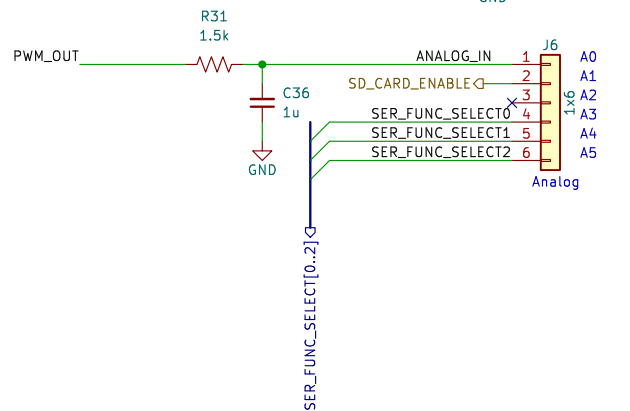
MBED CE

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/

SPI has 2 chip selects:
* SD_CS goes to the SD card socket. It's monitored by the logic analyzer but not hooked to the FT232H. 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 FT232H 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.

J5

1	IOREF	
2	~RESET	
3	+3V3	
4	+5V	
5	GND	
6	GND	
7	VIN	

1x8

Power

GND

J7

I2C{SCL SDA}

SPI{SCLK MOSI MISO SD_CS HW_CS}

D19/SCL	1	I2C.SCL
D18/SDA	2	I2C.SDA
AREF	3	
GND	4	
D13/SCLK	5	SPI.SCLK
D12/MISO	6	SPI.MISO
D11/MOSI	7	SPI.MOSI
D10/CS	8	SPI.HW_CS
D9	9	
D8	10	SPI.SD_CS

1x10

Digital/PWM

GND

R31
1.5k

PWM_OUT

ANALOG_IN

SD_CARD_ENABLE

SER_FUNC_SELECT0
SER_FUNC_SELECT1
SER_FUNC_SELECT2

J6

1	A0	
2	A1	
3	A2	
4	A3	
5	A4	
6	A5	

1x6

Analog

C36
1u

GND

SER_FUNC_SELECT[0..2]

J4

GPOUT[0..2]

D7	1	GPOUT2
D6	2	GPIN2
D5	3	GPOUT1
D4	4	GPIN1
D3	5	GPOUT0
D2	6	GPIN0
D1/TX	7	UART.MCU_TX
D0/RX	8	UART.MCU_RX

1x8

Digital/PWM

NT11   UART.CTS

1k
R28   NT9   PWM_OUT

1k
R30   NT10   UART.RTS

1k
R27

UART{MCU_TX MCU_RX RTS CTS}

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

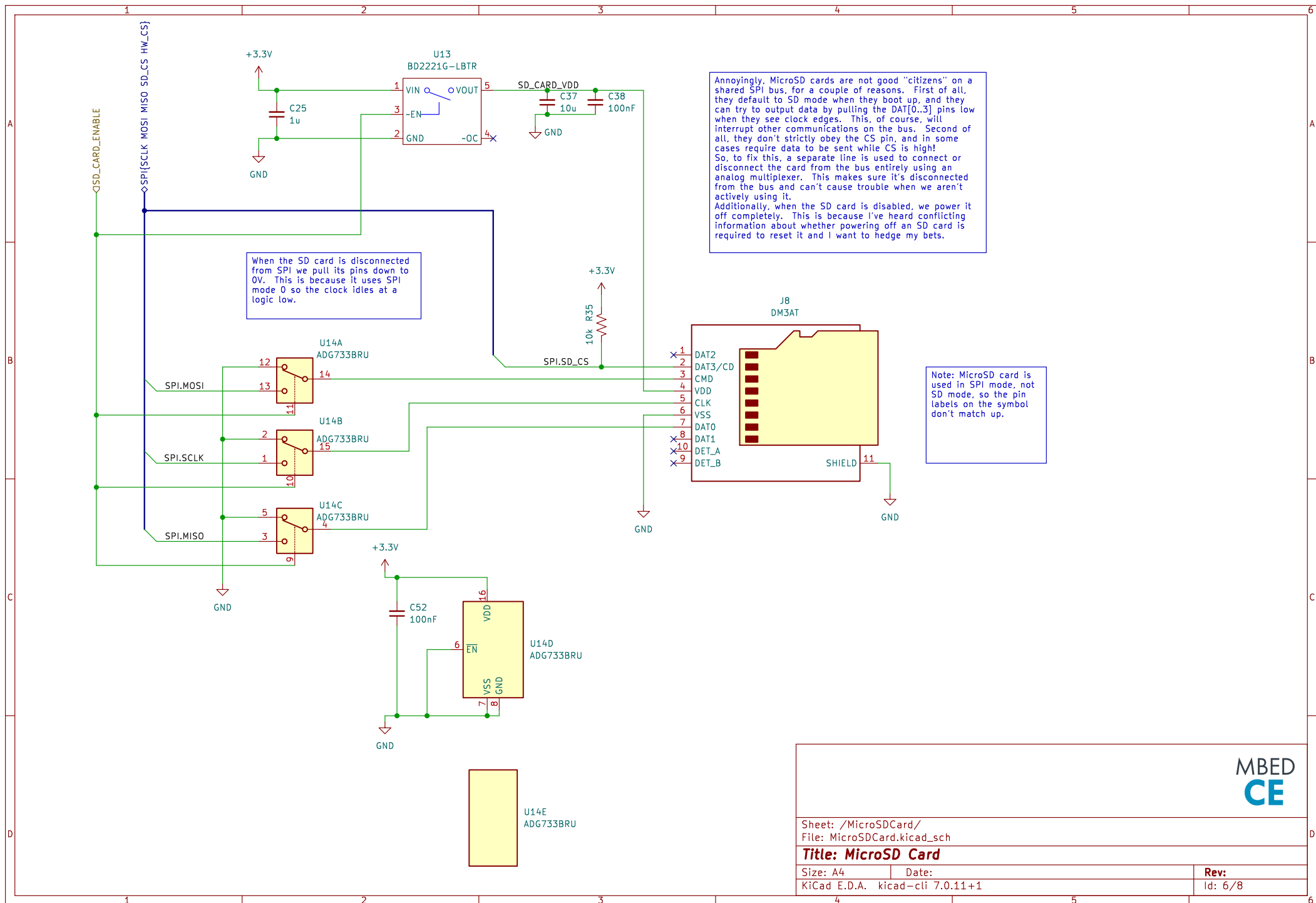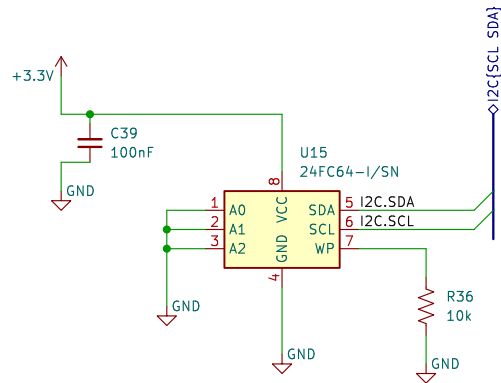Meanwhile, D5 does double duty as a PWM out pin.

MBED
CE

Sheet: /ArduinoConnector/
File: ArduinoConnector.kicad_sch

Title: Arduino Connector

Size: A4	Date:		Rev:
KiCad E.D.A. kicad-cli 7.0.11+1			Id: 5/8

+3.3V

U13
BD2221G-LBTR

1 VIN — VOUT 5 **SD_CARD_VDD**
3 ~EN
2 GND — ~OC 4

C25
1u

GND

C37
10u

C38
100nF

GND

<SD_CARD_ENABLE

<SPI{SCLK MOSI MISO SD_CS HW_CS}

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.
Additionally, when the SD card is disabled, we power it off completely. This is because I've heard conflicting information about whether powering off an SD card is required to reset it and I want to hedge my bets.

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

10k R35

J8
DM3AT

SPI.SD_CS

1 DAT2
2 DAT3/CD
3 CMD
4 VDD
5 CLK
6 VSS
7 DAT0
8 DAT1
10 DET_A
9 DET_B

SHIELD 11

GND

Note: MicroSD card is used in SPI mode, not SD mode, so the pin labels on the symbol don't match up.

U14A
ADG733BRU

12
13 — 14
11

SPI.MOSI

U14B
ADG733BRU

2
1 — 15
10

SPI.SCLK

U14C
ADG733BRU

5
3 — 4
9

SPI.MISO

GND

GND

+3.3V

C52
100nF

U14D
ADG733BRU

16 VDD
6 EN
7 VSS
8 GND

GND

U14E
ADG733BRU

MBED
CE

Sheet: /MicroSDCard/
File: MicroSDCard.kicad_sch
Title: MicroSD Card
Size: A4 | Date:
KiCad E.D.A. kicad-cli 7.0.11+1 | **Rev:**
Id: 6/8

+3.3V

C39
100nF

GND

U15
24FC64-I/SN

1	A0		SDA	5	I2C.SDA
2	A1	VCC	SCL	6	I2C.SCL
3	A2	GND	WP	7	

I2C{SCL SDA}

GND

GND

GND

R36
10k

GND

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

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.

3.3V LDO

J3
USB4105-GF-A

VBUS	A4
CC1	A5
CC2	B5
D−	A7
D−	B7
D+	A6
D+	B6
SBU1	A8
SBU2	B8

SHIELD S1

GND A1 PWR_FLAG

GND

VBUS_IN

5.1k R11
5.1k R12

GND

C12
2.2uF

GND

FB3
300@100MHz

C13
2.2uF

GND

+5V PWR_FLAG

C16
100nF

GND

U9
LD1117S33TR

3 VI   VO 2
1 GND

GND

+3.3V

C17
10u

GND

U8
USBLC6-2SC6

6   4
1   3
5
2

GND

VBUS is sent through a TVS diode, then a ferrite bead filter recommended by Phil's Lab on Youtube before going to the rest of the circuit

1.5A total available on +5V, and up to 800mA can be converted to 3.3V

MBED
CE

Sheet: /PowerUSBPort/
File: PowerUSBPort.kicad_sch

Title: Power Input USB Port

Size: A4   Date:

KiCad E.D.A. kicad-cli 7.0.11+1

Rev:

Id: 8/8