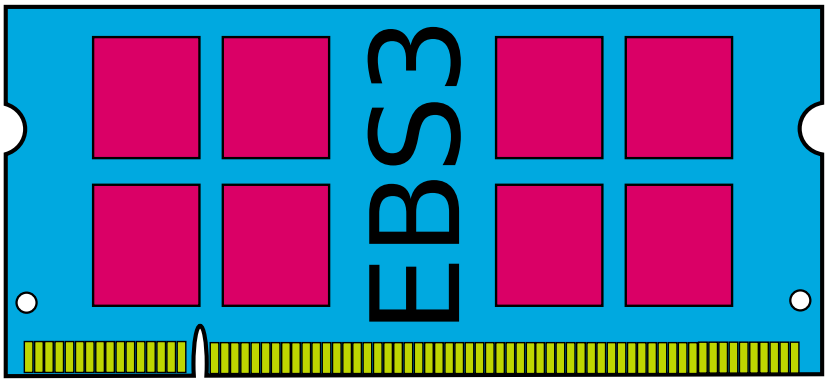


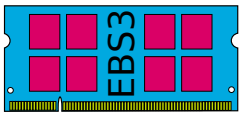
FPGA-EBS3 Electronic

Technical documentation

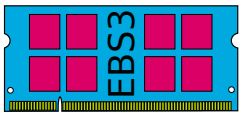
Hes·so VALAIS WALLIS

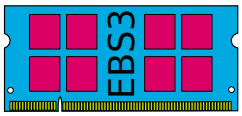
School of Engineering

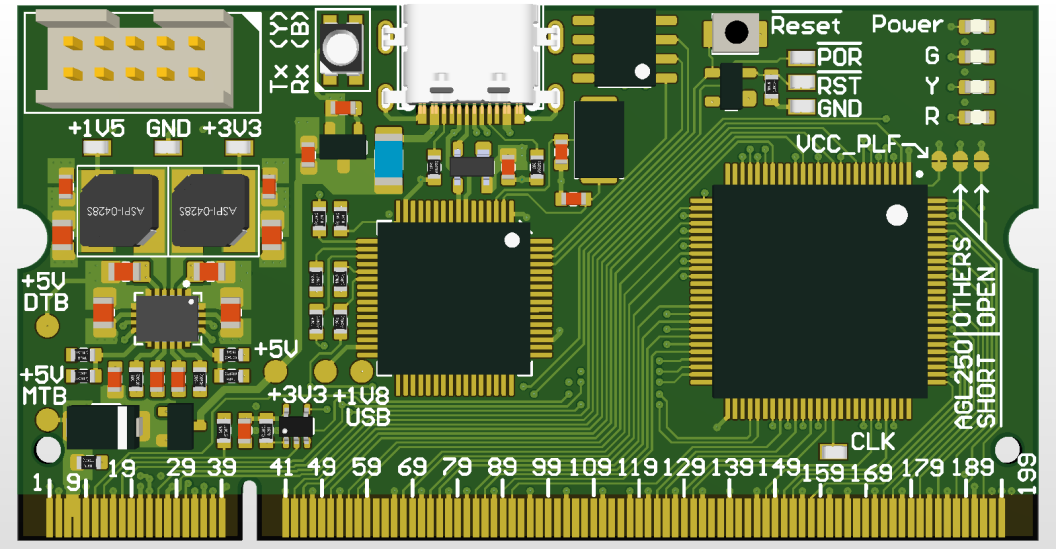
Author: Amand Axel, Silvan Zahno
Date: March 20, 2023
Version: v1.0



Contents




EBS3



Kart Daughterboard (AGLN250)

Hes·so VALAIS WALLIS

School of Engineering

Author: Amand Axel, Silvan Zahno
Date: March 20, 2023
Version: v1.0

1 Overview

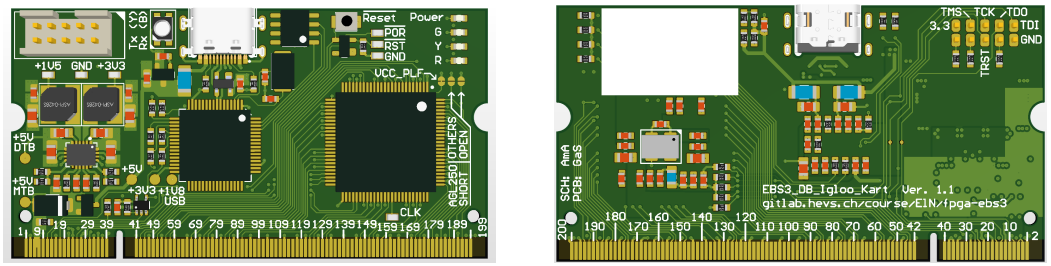


Figure 1: PCB board

The boards embeds the following functionalities:

- USB or MoBo +5V power input, creating internal voltages and a +3.3V / 3A user rail.
- USB connectivity, with one JTAG channel and one UART (virtual COM port) with TX/Rx LEDs.
- External JTAG programmer, automatic JTAG path extender.
- Power-on-reset chip, with an extra reset-button (either on the board or from a MoBo).
- Three user LEDs + 1 power LED indicator.

Technical files

The schematic of the board is given under I Schematic.

Rooting is available under II Rooting *(open the page with Inkscape for layers)*.

The bill of material is given under III BOM.

The pinout of the SODIMM-200 connector is available under IV Pinout.

2 Specifications

2.1 Overview

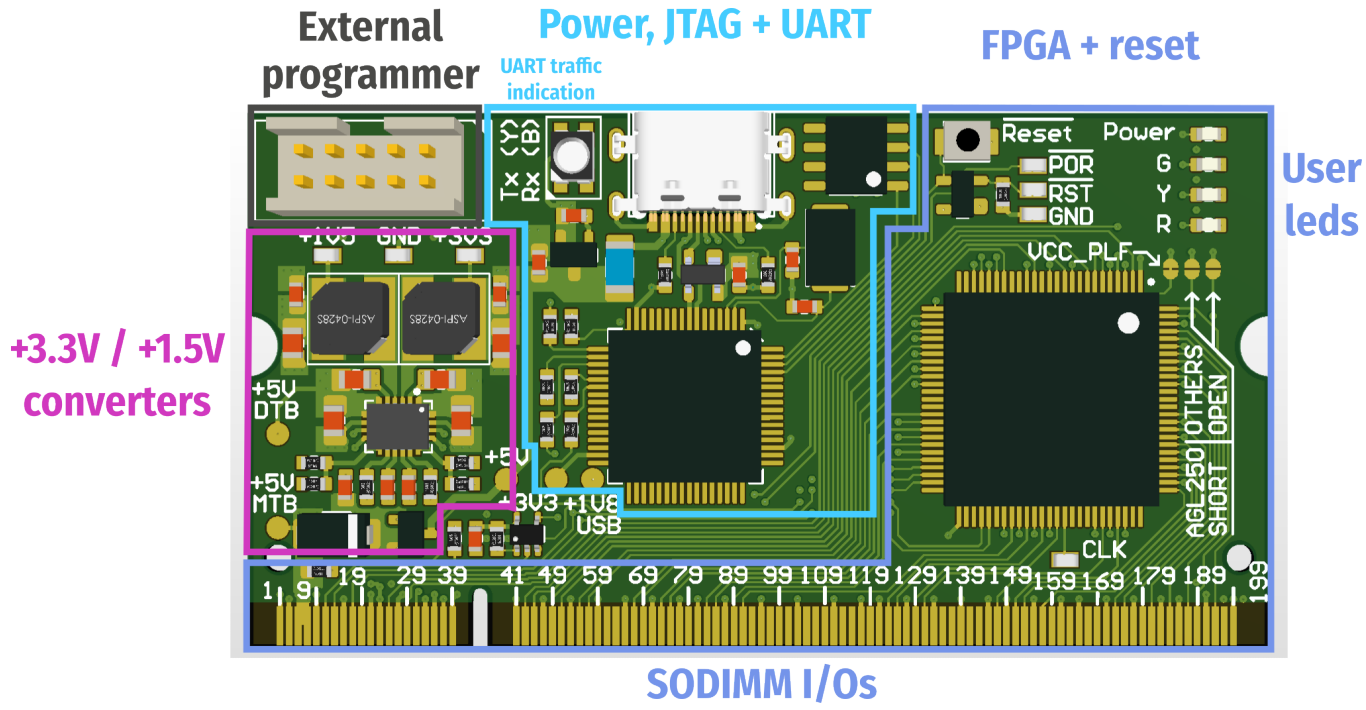


Figure 2: Card overview

2.2 Supply

The board can be powered either through USB *(+5V, no PD)* or from the motherboard (through dedicated SODIMM pins).

The MoBo rail is prioritized over the USB one:

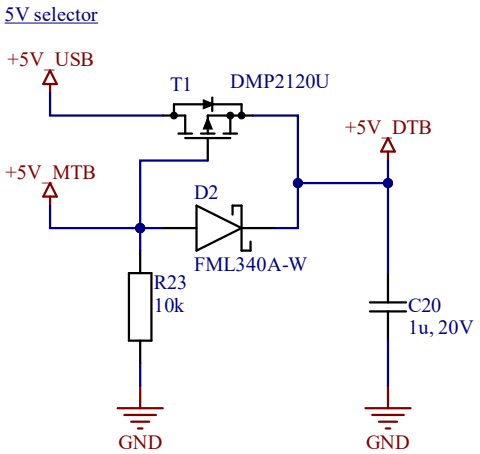


Figure 3: +5V selector - Circuit

Three cases:

- **+5V_USB**, no +5V_MTB : the transistors gate is low => it conducts fully, the diode blocks the path back to +5V_MTB.

- no +5V_USB, **+5V_MTB** : the diode conducts with a slight loss, while the transistors diode blocks the path back to +5V_USB.
- **+5V_USB**, **+5V_MTB** : the transistors gate is high => it blocks, and the internal diode blocks too (+5V on both sides). *Danger here would be for a too high voltage on the USB rail which would then take precedence over the MTB rail if* $U_{usb} > U_{mtb} - U_{d2} + U_{dt1}$.

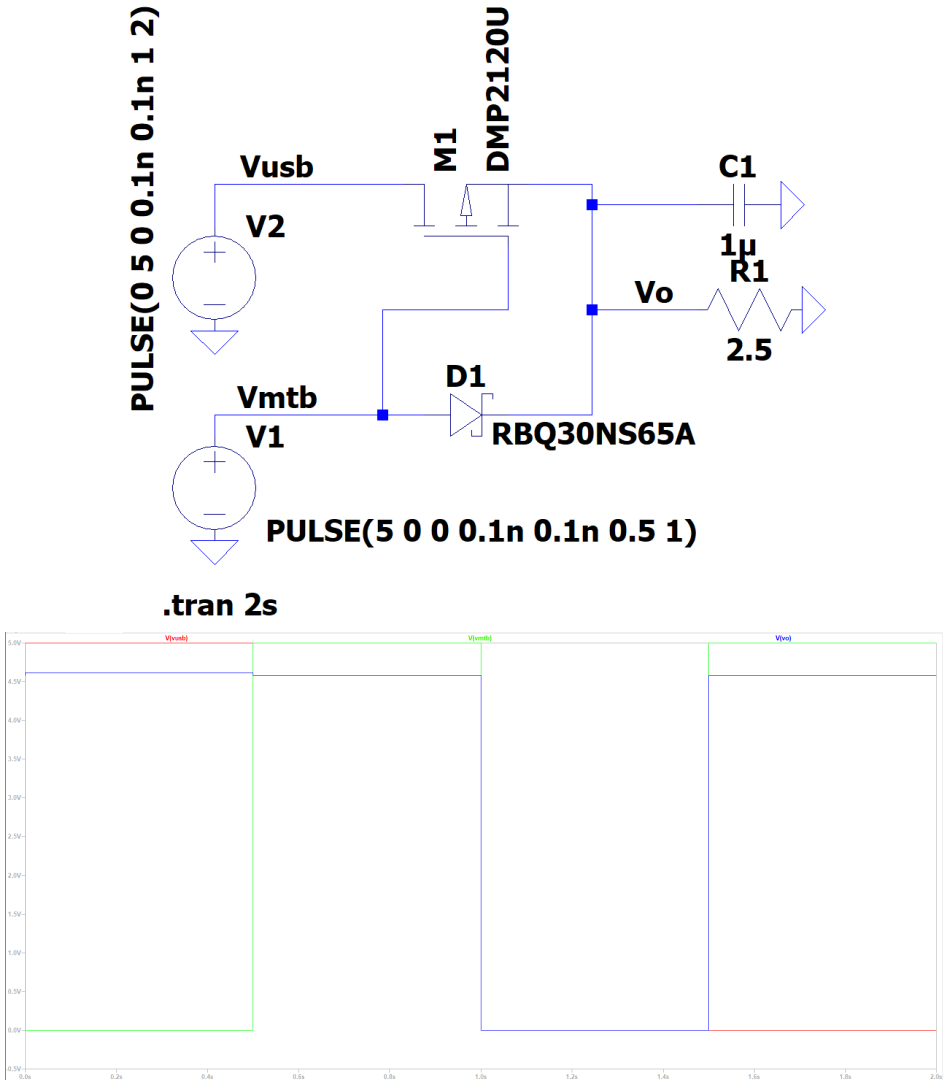


Figure 4: +5V selector - Simulation

Internally, a +1.5V and a +3.3V rails are created with the MIC23158YML buck converter:

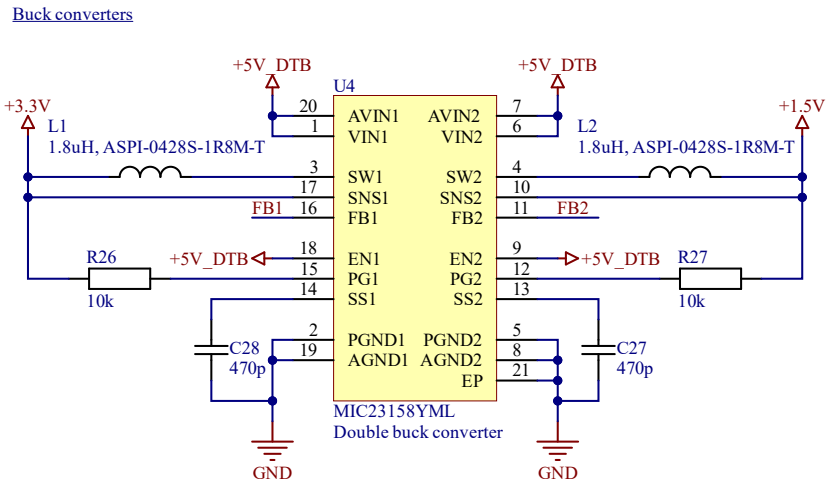


Figure 5: Buck converter - Circuit

The two feedback pins determine the output voltage value:

$$V_{out} = V_{ref} * (1 + \frac{R_h}{R1}), \text{ with } V_{ref} = 0.62[V].$$

For both rails:

+3.3V $0.62 * (1 + \frac{309k}{71.5k}) = 3.299[V]$

+1.5V $0.62 * (1 + \frac{316k}{221k}) = 1.507[V]$

Only the +3.3V is available further through dedicated SODIMM pins.

The blue LED LD5 indicates voltage on the +3.3V rail.

It simply tells the user some voltage is present on the rail, NOT that its voltage is correct.

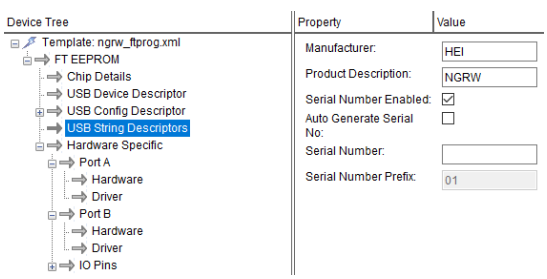
The rail is limited to 3A. *No protection against overvoltage or overcurrent exists.*

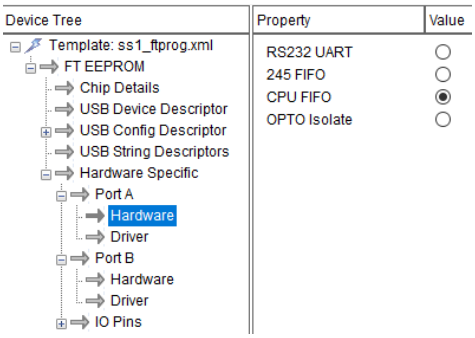
2.3 USB

Thanks to a FTDI2232H chip, the USB connector (over the USB 2.0 norm) can also be used to:

- Flash the FPGA through JTAG (channel A)
- Communicate through UART (Virtual COM port on the PC side, channel B)

The chip is configured through FTDI's program named **FT_Prog**, which saves it into the 93LC56B EEPROM:





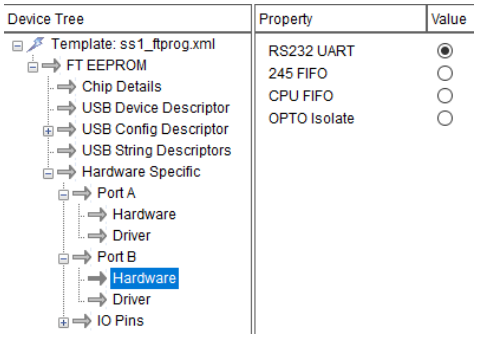


Figure 6: FT_Prog configuration

The UART (channel B) consists only of Rx and Tx (no flow controls). The bicolor LED LD1 indicates data transfer: blue means the board is receiving, while yellow is for data sent to host.

The chip needs to be configured first for the UART to work !

2.4 JTAG

The JTAG protocol is used to flash the FPGA.

It is done either through the USB or with a dedicated programmer (such as the FlashPro4 from Microsemi) on connector J2 with the following pining (As seen from top):

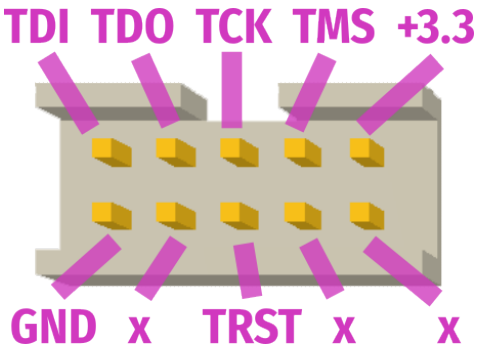


Figure 7: External programmer pining

Both usages are explained under Libero SoC (Microsemi).

The NC7SV126P5X buffer (U7) is used to extend the JTAG path through any other chip who may be present on the motherboard. By setting the signal **JTAG_SHORT_SEL** to '0', the output is no more linked to TDI such that the motherboard is responsible to close the JTAG data path.

2.5 FPGA

The FPGA can be either an AGLN125 or an AGLN250 from Microsemi.

When using the AGLN250, pins 46, 99 and 100 of the chip (pins 89, 194 and 195 of the SODIMM respectively) are power pins. DO NOT inject power back into them.

The FPGA is clocked with a 10 MHz quartz. One PLL block is internally available.

Reset The APX811-31U is used as power-on-reset monitor, holding the reset signal low while either the power is bad (< 3.08 V) or the user presses button S1. Also, one can reset the FPGA by setting the dedicated **nRESET_IN** signal low from the motherboard:

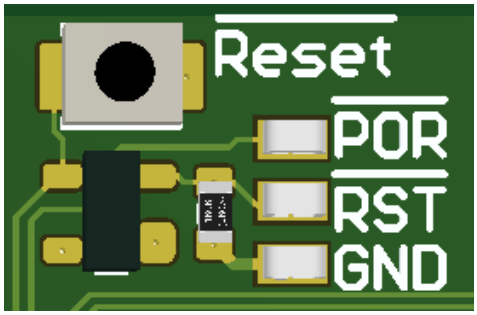


Figure 8: Power-on-reset

Chip selection Two jumpers allow to adapt the board for either an AGLN125 or an AGLN250 FPGA. They are labeled on the board as:

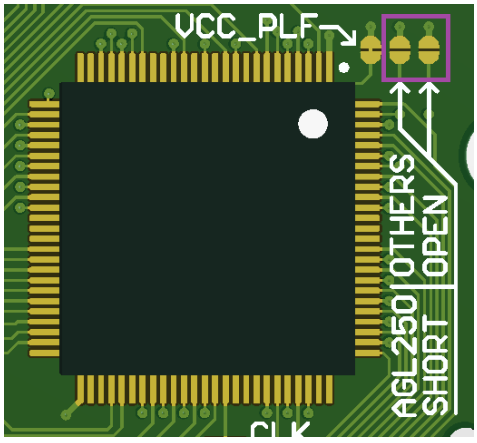


Figure 9: FPGA chip selection jumpers

They are open for the AGLN125 chip, because those are linked to I/Os pins.

On the AGLN250, they must both be short-circuited since they are power pins and need to be connected to the correct sources.

VCC_PLF jumper A jumper named **VCC_PLF** (can be seen figure 9) ties the analog PLL power supply to the ground to reduce the chip's power consumption. If opened, it must be manually wired to a filtered supply to work.

3 SW Configuration

The board is used with Libero Libero SoC (Microsemi).

3.1 Board configuration

The default configuration (constraints file) for the board is available under V Constraints file.

- Bank voltages are set with:
  `set_iobank Bank1 -vcci 3.3 -fixed yes` (sets bank to 3.3V)
- I/Os are declared as:

```
set_io Rx_BLE -pinname 3 -fixed yes -DIRECTION Input -RES_PULL Up
set_io {PMOD1[1]} -pinname 8 -fixed yes -DIRECTION Output
set_io SDA_battery -pinname 5 -fixed yes -DIRECTION InOut
```

  - set_io: io name in the VHDL file, use {Name[IDX]} for vectors
  - pinname: the IO number
  - DIRECTION: Input, Output or InOut
  - RES_PULL: Up, Down, or omit the argument for none

3.2 Test project

The tester is planned for 10 MHz clocks.

A HDL Designer test project is available with the following:

- LEDs will light one after the other at 2 Hz.

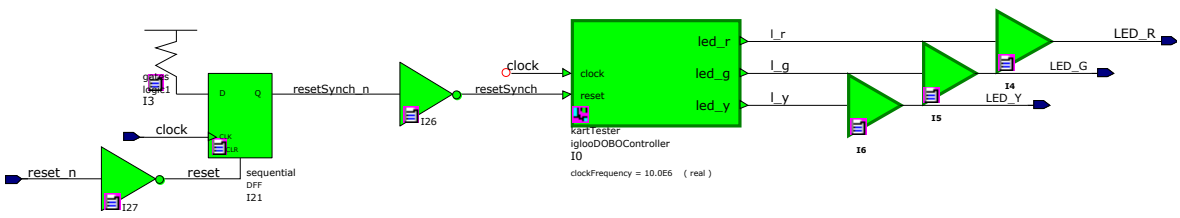
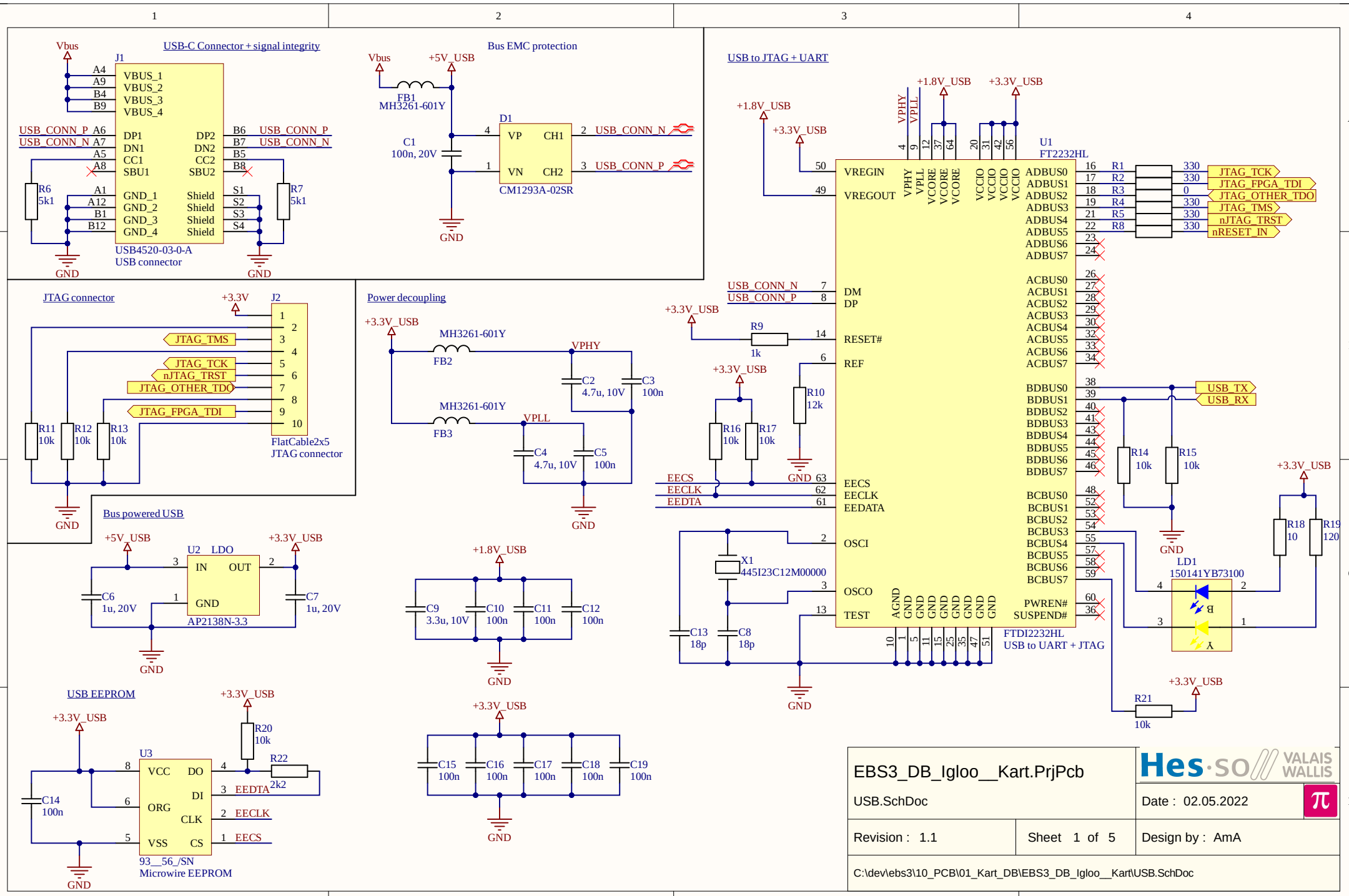


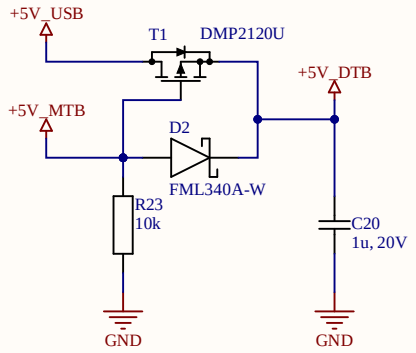
Figure 10: HDL Designer test program

For more complete tests, see Kart Motherboard (Battery, PMOD, BLE).

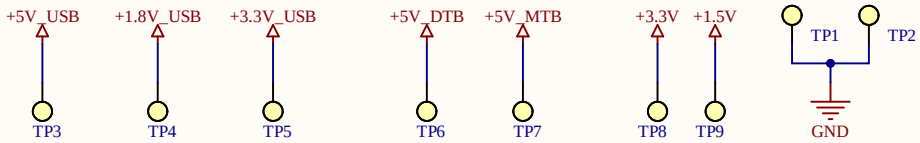
I Schematic


USB-C Connector + signal integrity
USB4520-03-0-A
USB connector
Bus EMC protection
FB1
MH3261-601Y
C1
100n, 20V
D1
CM1293A-02SR
USB to JTAG + UART
U1
FT2232HL
FTDI2232HL
USB to UART + JTAG
JTAG connector
FlatCable2x5
JTAG connector
Power decoupling
Bus powered USB
U2 LDO
AP2138N-3.3
USB EEPROM
U3
93__56_/SN
Microwire EEPROM
X1
445I23C12M00000
LD1
150141YB73100
EBS3_DB_Igloo__Kart.PrjPcb
USB.SchDoc
Date : 02.05.2022
Revision : 1.1
Sheet 1 of 5
Design by : AmA
C:\dev\ebs3\10_PCB\01_Kart_DB\EBS3_DB_Igloo__Kart\USB.SchDoc

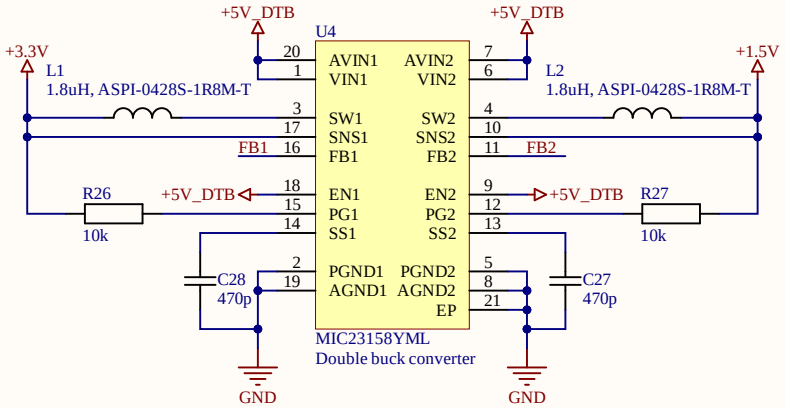
5V selector

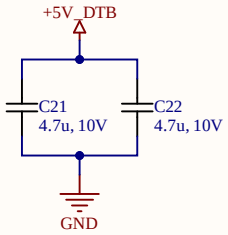


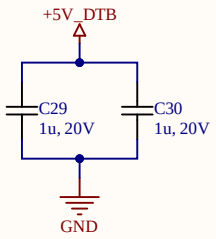
Test points



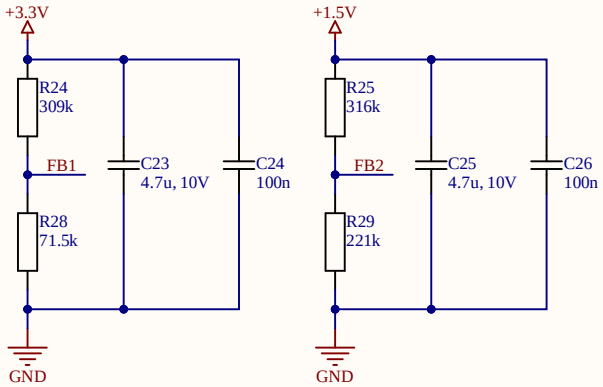
Buck converters





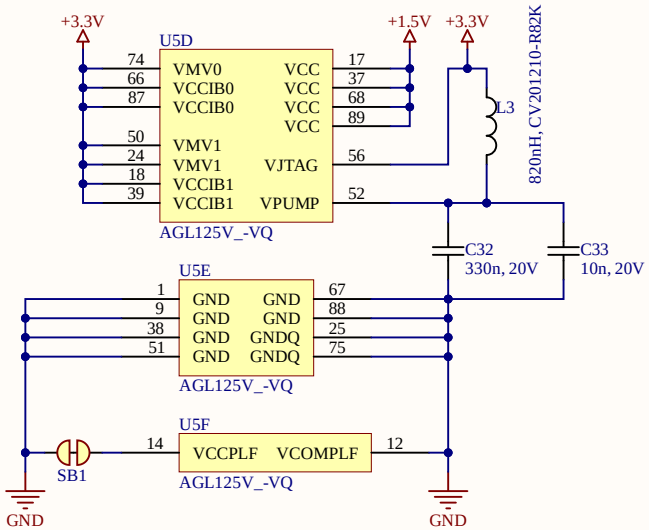


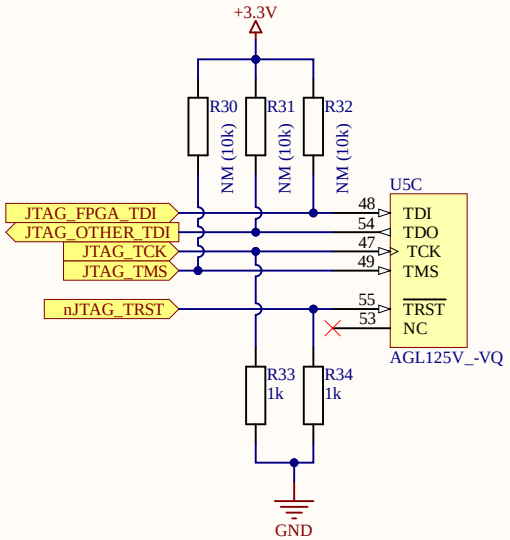
Output voltage definition



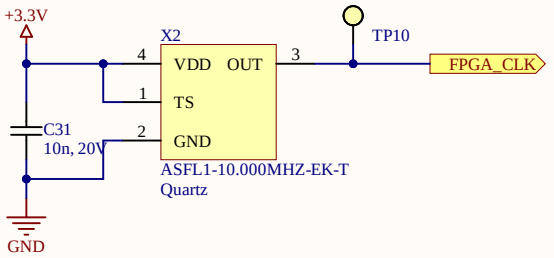
EBS3_DB_Igloo__Kart.PrjPcb		Hes·so VALAIS WALLIS
Power.SchDoc		Date : 02.05.2022
Revision : 1.1	Sheet 2 of 5	Design by : AmA
C:\dev\ebs3\10_PCB\01_Kart_DB\EBS3_DB_Igloo__Kart\Power.SchDoc		

FPGA Power

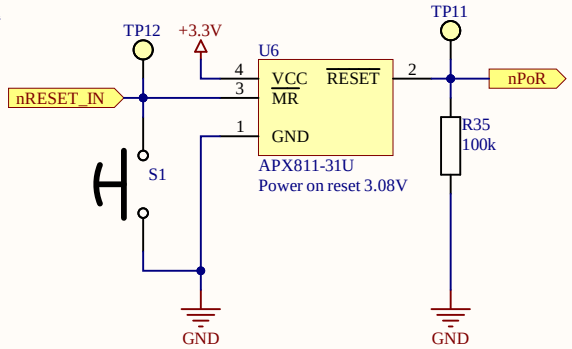




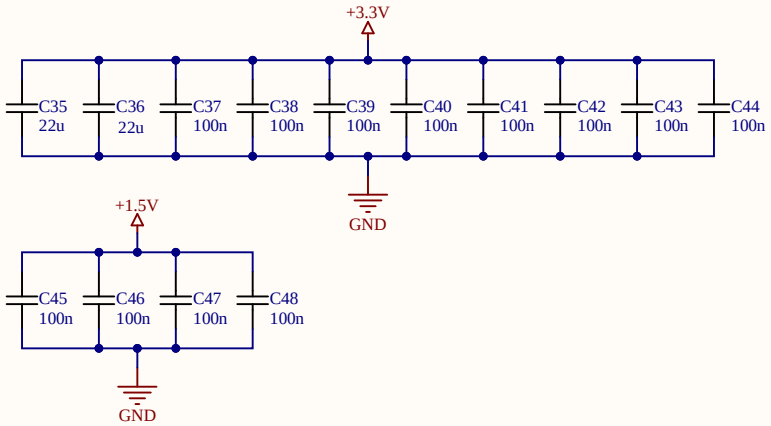
Clock



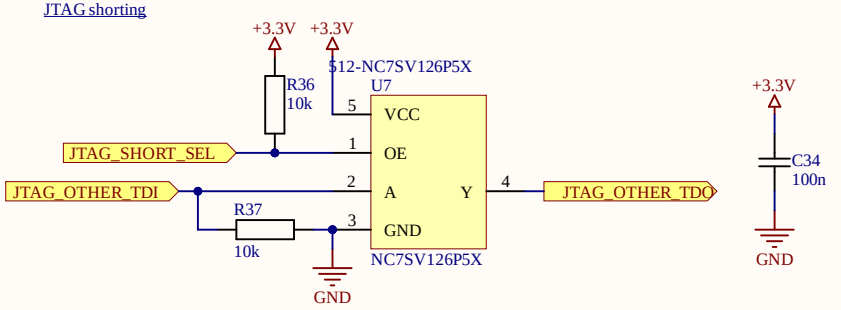
PoR



Decoupling

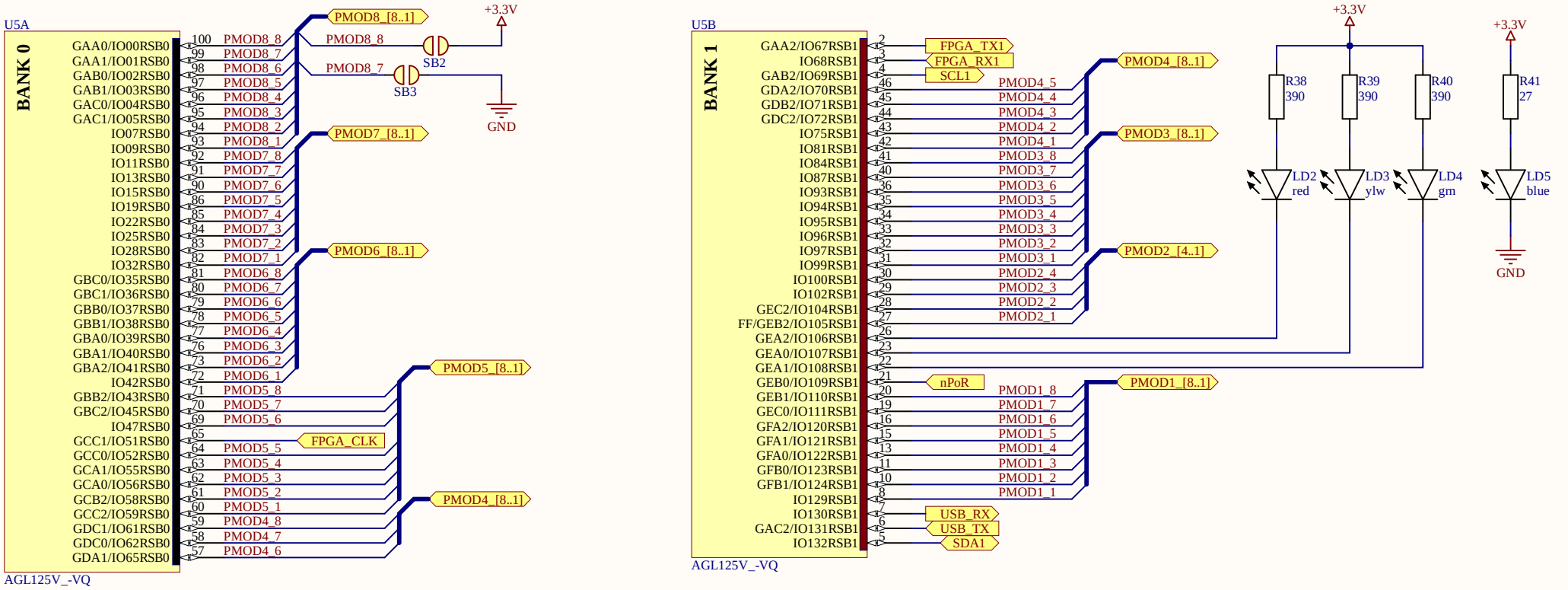


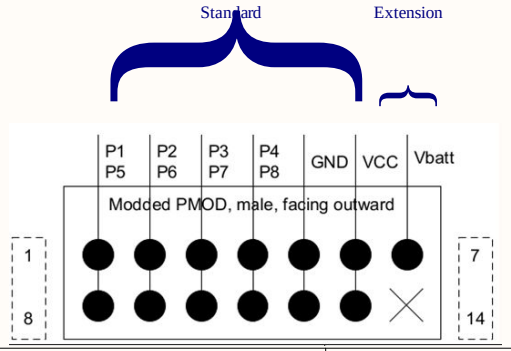
JTAG shorting



EBS3_DB_Igloo__Kart.PrjPcb		Hes·SO VALAIS WALLIS
FPGA_AlimPeriph.SchDoc		Date : 02.05.2022
Revision : 1.1	Sheet 3 of 5	Design by : AmA
C:\dev\ebs3\10_PCB\01_Kart_DB\EBS3_DB_Igloo__Kart\FPGA_AlimPeriph.SchDoc		

FPGA IOs



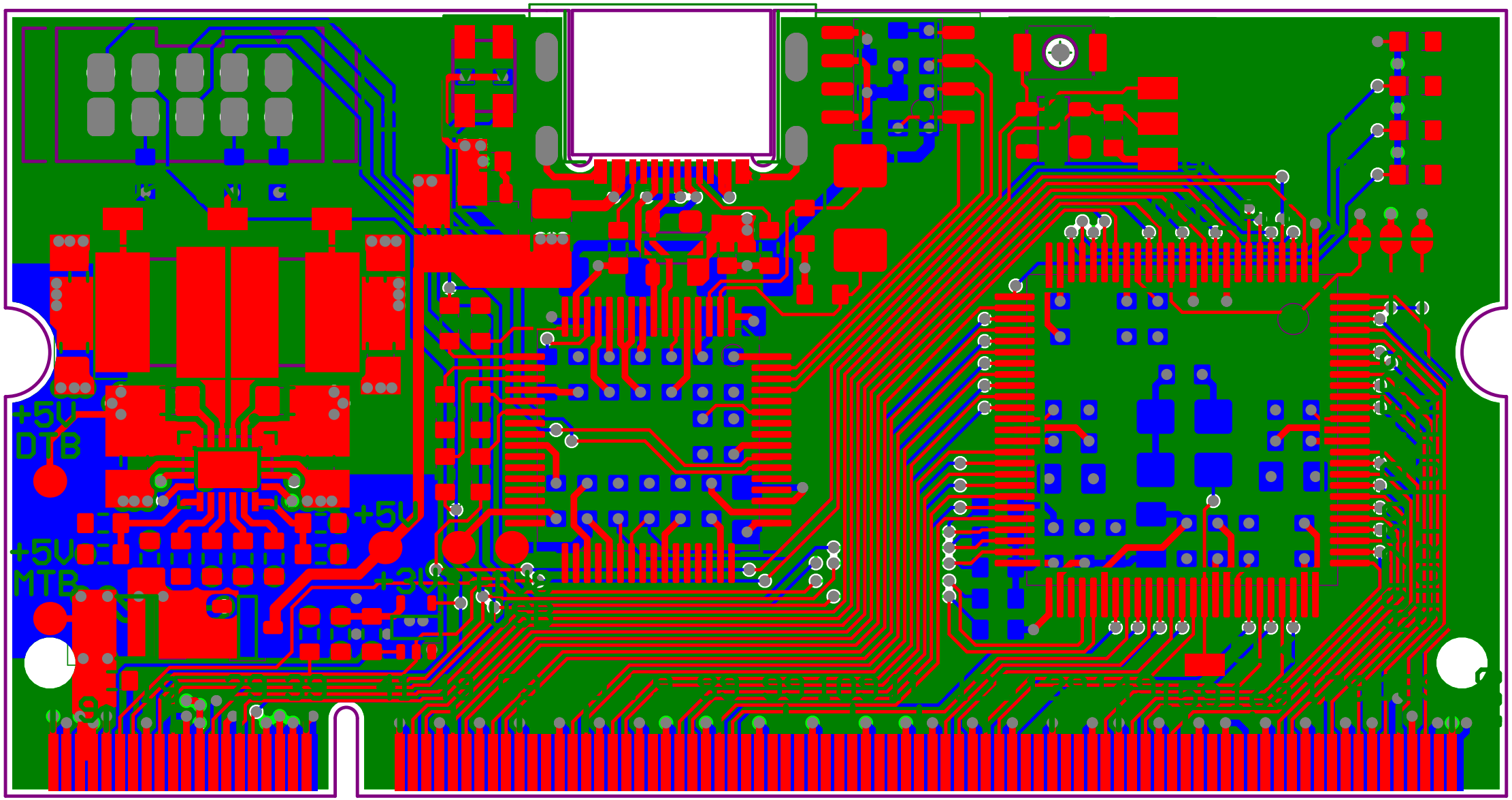


EBS3_DB_Igloo__Kart.PrjPcb		Hes·so VALAIS WALLIS
FPGA_IO.SchDoc		Date : 02.05.2022
Revision : 1.1	Sheet 4 of 5	Design by : AmA
C:\dev\ebs3\10_PCB\01_Kart_DB\EBS3_DB_Igloo__Kart\FPGA_IO.SchDoc		

SODIMM-200 connections

EBS3_DB_Igloo__Kart.PrjPcb			Hes·SO VALAIS WALLIS
SODIMM200.SchDoc			Date : 02.05.2022
Revision : 1.1	Sheet 5 of 5		Design by : AmA
C:\dev\ebs3\10_PCB\01_Kart_DB\EBS3_DB_Igloo__Kart\SODIMM200.SchDoc			

II Rooting


+5V
DIB
+5V
MTB
+5V
+3V3

III BOM

Name	Description	Designator	Quantity	Footprint
1mm PCB Thickness	DDR1 & DDR2 Board 0.6mm pitch 200 Positions	B1	1	DDR2 SODIMM Socket 0.6mm pitch 200 Positions 2.5V
100n, 20V	Capacitor	C1	1	CAPACITOR 1608 REFLOW
4.7u, 10V	Capacitor	C2, C4, C21, C22, C23, C25	6	CAPACITOR 2012 REFLOW
100n	Capacitor	C3, C5, C10, C11, C12, C14, C15, C16, C17, C18, C19, C24, C26, C34, C37, C38, C39, C40, C41, C42, C43, C44, C45, C46, C47, C48	26	CAPACITOR 1608 REFLOW
1u, 20V	Capacitor	C6, C7, C20, C29, C30	5	CAPACITOR 1608 REFLOW, CAPACITOR 2012 REFLOW
18p	Capacitor	C8, C13	2	CAPACITOR 1608 REFLOW
3.3u, 10V	Capacitor	C9	1	CAPACITOR 2012 REFLOW
470p	Capacitor	C27, C28	2	CAPACITOR 1608 REFLOW
10n, 20V	Capacitor	C31, C33	2	CAPACITOR 1608 REFLOW
330n, 20V	Capacitor	C32	1	CAPACITOR 1608 REFLOW
22u	Capacitor	C35, C36	2	CAPACITOR 2012 REFLOW
CM1293A-02SR	ESD Protection Array 2 Channels	D1	1	SOT143 P1.9 C1.4X3.0 H1.1
FML340A-W	Diode Schottky generic 2 Leads	D2	1	DIODE DO-214AC (SMA)
MH3261-601Y	Inductor	FB1, FB2, FB3	3	INDUCTOR 3216 REFLOW
USB4520-03-0-A	Connector	J1	1	USB-C-SMD GCT USB4520030A
FlatCable2x5	FlatCable2x5	J2	1	FlatCable2x5 2MM Straight TH FCI 98414-G06-10ULF
1.8uH, ASPI-0428S-1R8M-T	Inductor	L1, L2	2	ASPI-0428S
820nH, CV201210-R82K	Inductor	L3	1	INDUCTOR 2012 REFLOW
150141YB73100	3.5x2.8mm SMD Bi-color TOP LED	LD1	1	LED WE 150141YB73100
red	LED generic	LD2	1	LED 1608 REFLOW
ylw	LED generic	LD3	1	LED 1608 REFLOW
grn	LED generic	LD4	1	LED 1608 REFLOW
blue	LED generic	LD5	1	LED 1608 REFLOW
330	Resistor	R1, R2, R4, R5, R8	5	RESISTOR 1608 REFLOW
0	Resistor	R3	1	RESISTOR 1608 REFLOW
5k1	Resistor	R6, R7	2	RESISTOR 1608 REFLOW
1k	Resistor	R9, R33, R34	3	RESISTOR 1608 REFLOW
12k	Resistor	R10	1	RESISTOR 1608 REFLOW
10k	Resistor	R11, R12, R13, R14, R15, R16, R17, R20, R21, R23, R26, R27, R36, R37	14	RESISTOR 1608 REFLOW
10	Resistor	R18	1	RESISTOR 1608 REFLOW
120	Resistor	R19	1	RESISTOR 1608 REFLOW
2k2	Resistor	R22	1	RESISTOR 1608 REFLOW
309k	Resistor	R24	1	RESISTOR 1608 REFLOW
316k	Resistor	R25	1	RESISTOR 1608 REFLOW
71.5k	Resistor	R28	1	RESISTOR 1608 REFLOW
221k	Resistor	R29	1	RESISTOR 1608 REFLOW
NM (10k)	Resistor	R30, R31, R32	3	RESISTOR 1608 REFLOW
100k	Resistor	R35	1	RESISTOR 1608 REFLOW
390	Resistor	R38, R39, R40	3	RESISTOR 1608 REFLOW
27	Resistor	R41	1	RESISTOR 1608 REFLOW
	Push Button, 2 pins	S1	1	Push Button C2.5X3.0 OMRON B3U-1000P(M)-B
DMP2120U	Generic MOS-P	T1	1	SOT23 P0.95 C1.6x3.0 H1.3
ARWIN S2761-46R	Test Point	TP1, TP2, TP8, TP9, TP10, TP11, TP12	7	Test Point ARWIN S2761-46R
FT2232HL	Dual High Speed USB to Multipurpose UART/FIFO IC	U1	1	LQFP64 P0.5 C10.0x10.0 H1.6
AP2138N-3.3	Generic Positive Voltage Regulator	U2	1	SOT23 P0.95 C1.6x3.0 H1.3
93__56_/SN	2K bit low voltage serial EEPROMs, Without ORG Pin	U3	1	SO8 P1.27 C4.0X5.0 H1.75
MIC23158YML	3 MHz PWM Dual 2A Buck Regulator with HyperLight Load and Power Good	U4	1	QFN20 P0.5 T2.65x1.65 C4.0x3.0 H0.9
AGL125V_-VQ	IGLOO Low Power Flash FPGAs	U5	1	VQ100 P0.5 C14x14 H1.2
APX811-31U	4-Pin Microprocessor Supervisor With Manual Reset	U6	1	SOT143 P1.9 C1.4X3.0 H1.1
NC7SV126P5X	single 3–State buffer	U7	1	SC70-5 P0.65 C1.4x2.2 H1.0
445I23C12M00000	Crystal C5.0x3.2 CTS 445	X1	1	CRYSTAL C5.0x3.2 CTS 445
ASFL1-10.000MHZ-EK-T	Oscillator C5.0x3.2 Abracon ASFL1	X2	1	Oscillator C5.0x3.2 Abracon ASFL1

IV Pinout

Connector name on the schematic :	B1

Product Name Igloo AGLN250	Product Name Igloo AGLN125	Standard	Pin Numbers		Standard	Product Name Igloo AGLN125	Product Name Igloo AGLN250
Pin Type	Pin Type	Pin Type	Top	Bottom	Pin Type	Pin Type	Pin Type
+3.3V	+3.3V	+3.3V	1	2	GND	GND	GND
GND	GND	GND	3	4	+5V MTB	+5V MTB	+5V MTB
+5V MTB	+5V MTB	+5V MTB	5	6	+5V MTB	+5V MTB	+5V MTB
+5V MTB	+5V MTB	+5V MTB	7	8	GND	GND	GND
GND	GND	GND	9	10	JTAG_SEL	JTAG_SEL	JTAG_SEL
POR/	POR/	POR/	11	12	GND	GND	GND
TDO	TDO	TDO	13	14	RESET/	RESET/	RESET/
GND	GND	GND	15	16	TDI	TDI	TDI
TMS	TMS	TMS	17	18	GND	GND	GND
TRST	TRST	TRST	19	20	IO0 / CTS1		
GND	GND	GND	21	22	+5V DTB	+5V DTB	+5V DTB
+5V DTB	+5V DTB	+5V DTB	23	24	GND	GND	GND
SCL1	SCL1	IO1 / SCL1	25	26	TCK	TCK	TCK
GND	GND	GND	27	28	GND	GND	GND
Tx1	Tx1	IO2 / Tx1	29	30	IO3 / SDA1	SDA1	SDA1
Rx1	Rx1	IO4 / RX1	31	32	IO5 / RTS1		
+5V MTB	+5V MTB	+5V MTB	33	34	+5V MTB	+5V MTB	+5V MTB
+5V MTB	+5V MTB	+5V MTB	35	36	+5V MTB	+5V MTB	+5V MTB
+5V MTB	+5V MTB	+5V MTB	37	38	+5V MTB	+5V MTB	+5V MTB
GND	GND	GND	39	40	GND	GND	GND
GND	GND	GND	41	42	GND	GND	GND
P1_1	P1_1	IO6	43	44	IO7	P1_2	P1_2
P1_3	P1_3	IO8	45	46	IO9	P1_4	P1_4
GND	GND	IO10	47	48	IO110	GND	GND
P1_5	P1_5	IO12 / SCLK1	49	50	IO13 / MOSI1 / QS1_0	P1_6	P1_6
P1_7	P1_7	IO14 / MISO1 / QS1_1	51	52	IO15 / QS1_2	P1_8	P1_8
GND	GND	GND	53	54	GND	GND	GND
P2_1	P2_1	IO16 / QS1_3	55	56	IO17	P2_2	P2_2
P2_3	P2_3	IO18	57	58	IO19	P2_4	P2_4
GND	GND	IO20	59	60	IO21	GND	GND
P2_5	P2_5	IO22	61	62	IO23	P2_6	P2_6
P2_7	P2_7	IO24	63	64	IO25	P2_8	P2_8
GND	GND	GND	65	66	GND	GND	GND
P3_1	P3_1	IO26	67	68	IO27	P3_2	P3_2
P3_3	P3_3	IO28	69	70	IO29	P3_4	P3_4
GND	GND	IO30	71	72	IO31	GND	GND
P3_5	P3_5	IO32	73	74	IO33	P3_6	P3_6
P3_7	P3_7	IO34	75	76	IO35	P3_8	P3_8
GND	GND	GND	77	78	GND	GND	GND
		DIFF1-A / CANL	79	80	DIFF1-B / CANH		
+3.3V	+3.3V	+3.3V	81	82	+3.3V	+3.3V	+3.3V
P4_1	P4_1	IO36	83	84	IO37	P4_2	P4_2
P4_3	P4_3	IO38	85	86	IO39	P4_4	P4_4
+3.3V	+3.3V	+3.3V	87	88	+3.3V	+3.3V	+3.3V
P8_6	P8_6	IO40	89	90	IO41	P4_6	P4_6
P4_7	P4_7	IO42	91	92	IO43	P4_8	P4_8
		IO44	93	94	IO45		
+3.3V	+3.3V	+3.3V	95	96	+3.3V	+3.3V	+3.3V
		IO46	97	98	IO47		
		IO48	99	100	IO49		
		IO50	101	102	IO51		
+3.3V	+3.3V	+3.3V	103	104	+3.3V	+3.3V	+3.3V
		IO52	105	106	IO53		
		IO54	107	108	IO55		
		IO56	109	110	IO57		
+3.3V	+3.3V	+3.3V	111	112	+3.3V	+3.3V	+3.3V
		IO58	113	114	IO59		
		IO60	115	116	IO61		
+3.3V	+3.3V	+3.3V	117	118	+3.3V	+3.3V	+3.3V
		DIFF2-A	119	120	DIFF2-B		
GND	GND	GND	121	122	GND	GND	GND
P5_1	P5_1	IO62 / MDC	123	124	IO63 / MDIO	P5_2	P5_2
P5_3	P5_3	IO64 / INTR	125	126	IO65 / nRESET	P5_4	P5_4
GND	GND	IO66 / CRS	127	128	IO67 / COL	GND	GND
P5_5	P5_5	IO68 / TX_CLK	129	130	IO69 / GTX_CLK	P5_6	P5_6
P5_7	P5_7	IO70 / TX_EN	131	132	GND	GND	GND
GND	GND	GND	133	134	IO71 / TX_ER	P5_8	P5_8
		IO72 / TXD0	135	136	IO73 / TXD1		
		IO74 / TXD2	137	138	GND	GND	GND
GND	GND	GND	139	140	IO75 / TXD3		
		IO76 / TXD4	141	142	IO77 / TXD5		
		IO78 / TXD6	143	144	GND	GND	GND
GND	GND	GND	145	146	IO79 / TXD7		
P6_1	P6_1	IO80 / RX_CLK	147	148	IO81 / RX_DV	P6_2	P6_2
GND	GND	IO82 / RX_ER	149	150	IO83 / RXD0	GND	GND
P6_3	P6_3	IO84 / RXD1	151	152	IO85 / RXD2	P6_4	P6_4
P6_5	P6_5	IO86 / RXD3	153	154	IO87 / RXD4	P6_6	P6_6
GND	GND	GND	155	156	GND	GND	GND
P6_7	P6_7	IO88 / RXD5	157	158	IO89 / RXD6	P6_8	P6_8
		IO90 / RXD7	159	160	IO91		
GND	GND	GND	161	162	GND	GND	GND
		DIFF3-A / SDA2	163	164	DIFF3-B /SCL2		
GND	GND	GND	165	166	DIFF4-A / Tx2		
		DIFF4-B / Rx2	167	168	GND	GND	GND
		DIFF5-A /RTS2	169	170	DIFF5-B / CTS2		
GND	GND	GND	171	172	GND	GND	GND
P7_1	P7_1	IO92 / SCLK2	173	174	IO93 / MOSI2 / QS2_0	P7_2	P7_2
P7_3	P7_3	IO94 / MISO2 / QS2_1	175	176	IO95 / QS2_2	P7_4	P7_4
GND	GND	GND	177	178	GND	GND	GND
P7_5	P7_5	IO96 / QS2_3	179	180	IO97	P7_6	P7_6
P7_7	P7_7	IO98	181	182	IO99	P7_8	P7_8
GND	GND	IO100	183	184	IO101	GND	GND
P8_1	P8_1	IO102	185	186	IO103	P8_2	P8_2
GND	GND	GND	187	188	DIFF6-A	P8_3	P8_3
P8_4	P8_4	DIFF6-B	189	190	GND	GND	GND
P8_5	P8_5	IO104	191	192	IO105	P4_5	GND
GND	GND	IO106	193	194	IO107	P8_7	GND
+3.3V	P8_8	IO108	195	196	IO109	GND	GND
		IO110	197	198	IO111		
+3.3V	+3.3V	+3.3V	199	200	GND	GND	GND

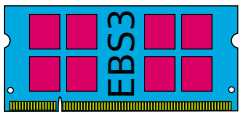
FREE PINS (no NC) : 0
FREE PINS (NC comprised) : 42

Signals names as seen from the daughterboard if not specified otherwise (e.g. Tx is the output of FPGA's UART block)

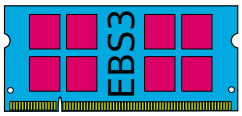
GND	Common ground
+3.3V	Daughterboard 3.3V output (max 1.5 [A])
+5V DTB	Daughterboard 5V output (LOW CURRENT CONSUMPTION IF RUNNING ON USB !)
+5V MTB	Motherboard 5V output
Txx	JTAG signals /JTAG_SEL : 1 = no extra chip, 0 = extra JTAG line used)
RESET/	Daughterboard reset input (active low)
POR/	Daughterboard PowerOnReset signal output (active low)
DIFFx-A/B	Differential pair I/O preferred pins (can be used as standard I/O)
IOx	Input / output pin
TX / Rx	3.3V EUSART signals, any I/O for complementary (CTR, DTS ...)
SCL / SDA	3.3V I2C signals
SCLK / MOSI / MISO	3.3V SPI signals, any I/O as slave select
QS	SPI dual/quad interface
CANL / CANH	3.3V CAN signals (REQUIRES A TRANSCEIVER ON THE MOTHERBOARD !)
TXD, RXD ...	Gigabit ethernet signals ((G)MII, MDC/MDIO, interrupt)

Jedec 4.20.11-1 link :
- Power pins, except pin 200 (added as GND), 199 and 1 as standard +3.3V
- 69, 83, 120 and 163 (test pins) as standard I/Os
- 30, 32, 164 and 166 (clock outputs) as standard I/Os
- Fixed JTAG signals, + RESET/ and POR/

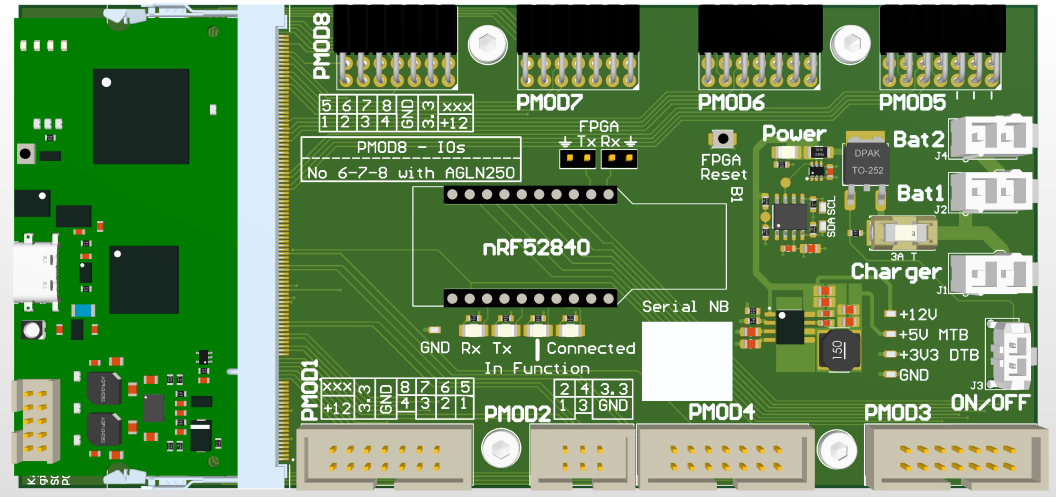
V Constraints file




```
# Actel Physical design constraints file
# Family: IGLOO , Die: AGLN250V5 , Package: 100 VQG

#-----------------------------------------------------------------------------
# Clocks and reset
#
set_io  clock           -pinname 65     -fixed yes      -DIRECTION Input
set_io  reset_n         -pinname 21     -fixed yes      -DIRECTION Input

#-----------------------------------------------------------------------------
# I/O - DOBO
#
set_io  LED_R           -pinname 26     -fixed yes      -DIRECTION Output
set_io  LED_Y           -pinname 23     -fixed yes      -DIRECTION Output
set_io  LED_G           -pinname 22     -fixed yes      -DIRECTION Output

#-----------------------------------------------------------------------------
# UART (USB)
#
set_io  Tx_USB           -pinname 7      -fixed yes      -DIRECTION Output
set_io  Rx_USB           -pinname 6      -fixed yes      -DIRECTION Input   -RES_PULL Up

##############################################################################
# IO banks setting
#
set_iobank Bank1         -vcci 3.3       -fixed yes
set_iobank Bank0         -vcci 3.3       -fixed yes

##############################################################################
# MOBO specific
#
set_io  SDA_battery     -pinname 5      -fixed yes      -DIRECTION InOut
set_io  SCL_battery     -pinname 4      -fixed yes      -DIRECTION Output

set_io  Tx_BLE          -pinname 2      -fixed yes      -DIRECTION Output
set_io  Rx_BLE          -pinname 3      -fixed yes      -DIRECTION Input   -RES_PULL Up

set_io  {PMOD1[1]}      -pinname 8      -fixed yes      -DIRECTION Output
set_io  {PMOD1[2]}      -pinname 10     -fixed yes      -DIRECTION Output
set_io  {PMOD1[3]}      -pinname 11     -fixed yes      -DIRECTION Output
set_io  {PMOD1[4]}      -pinname 13     -fixed yes      -DIRECTION Output
set_io  {PMOD1[5]}      -pinname 15     -fixed yes      -DIRECTION Output
set_io  {PMOD1[6]}      -pinname 16     -fixed yes      -DIRECTION Output
set_io  {PMOD1[7]}      -pinname 19     -fixed yes      -DIRECTION Output
set_io  {PMOD1[8]}      -pinname 20     -fixed yes      -DIRECTION Output

set_io  {PMOD2[1]}      -pinname 27     -fixed yes      -DIRECTION Output
set_io  {PMOD2[2]}      -pinname 28     -fixed yes      -DIRECTION Output
set_io  {PMOD2[3]}      -pinname 29     -fixed yes      -DIRECTION Output
set_io  {PMOD2[4]}      -pinname 30     -fixed yes      -DIRECTION Output

set_io  {PMOD3[1]}      -pinname 31     -fixed yes      -DIRECTION Output
set_io  {PMOD3[2]}      -pinname 32     -fixed yes      -DIRECTION Output
set_io  {PMOD3[3]}      -pinname 33     -fixed yes      -DIRECTION Output
set_io  {PMOD3[4]}      -pinname 34     -fixed yes      -DIRECTION Output
set_io  {PMOD3[5]}      -pinname 35     -fixed yes      -DIRECTION Output
set_io  {PMOD3[6]}      -pinname 36     -fixed yes      -DIRECTION Output
set_io  {PMOD3[7]}      -pinname 40     -fixed yes      -DIRECTION Output
set_io  {PMOD3[8]}      -pinname 41     -fixed yes      -DIRECTION Output
```




```
set_io  {PMOD4[1]}      -pinname 42     -fixed yes      -DIRECTION Output
set_io  {PMOD4[2]}      -pinname 43     -fixed yes      -DIRECTION Output
set_io  {PMOD4[3]}      -pinname 44     -fixed yes      -DIRECTION Output
set_io  {PMOD4[4]}      -pinname 45     -fixed yes      -DIRECTION Output
# CANNOT BE USED ON AGLN250 ! #set_io  PM4_5 -pinname 46  -fixed yes -DIRECTION Input
set_io  {PMOD4[5]}      -pinname 57     -fixed yes      -DIRECTION Output
set_io  {PMOD4[6]}      -pinname 58     -fixed yes      -DIRECTION Output
set_io  {PMOD4[7]}      -pinname 59     -fixed yes      -DIRECTION Output

set_io  {PMOD5[1]}      -pinname 60     -fixed yes      -DIRECTION Output
set_io  {PMOD5[2]}      -pinname 61     -fixed yes      -DIRECTION Output
set_io  {PMOD5[3]}      -pinname 62     -fixed yes      -DIRECTION Output
set_io  {PMOD5[4]}      -pinname 63     -fixed yes      -DIRECTION Output
set_io  {PMOD5[5]}      -pinname 64     -fixed yes      -DIRECTION Output
set_io  {PMOD5[6]}      -pinname 69     -fixed yes      -DIRECTION Output
set_io  {PMOD5[7]}      -pinname 70     -fixed yes      -DIRECTION Output
set_io  {PMOD5[8]}      -pinname 71     -fixed yes      -DIRECTION Output

set_io  {PMOD6[1]}      -pinname 72     -fixed yes      -DIRECTION Output
set_io  {PMOD6[2]}      -pinname 73     -fixed yes      -DIRECTION Output
set_io  {PMOD6[3]}      -pinname 76     -fixed yes      -DIRECTION Output
set_io  {PMOD6[4]}      -pinname 77     -fixed yes      -DIRECTION Output
set_io  {PMOD6[5]}      -pinname 78     -fixed yes      -DIRECTION Output
set_io  {PMOD6[6]}      -pinname 79     -fixed yes      -DIRECTION Output
set_io  {PMOD6[7]}      -pinname 80     -fixed yes      -DIRECTION Output
set_io  {PMOD6[8]}      -pinname 81     -fixed yes      -DIRECTION Output

set_io  {PMOD7[1]}      -pinname 82     -fixed yes      -DIRECTION Output
set_io  {PMOD7[2]}      -pinname 83     -fixed yes      -DIRECTION Output
set_io  {PMOD7[3]}      -pinname 84     -fixed yes      -DIRECTION Output
set_io  {PMOD7[4]}      -pinname 85     -fixed yes      -DIRECTION Output
set_io  {PMOD7[5]}      -pinname 86     -fixed yes      -DIRECTION Output
set_io  {PMOD7[6]}      -pinname 90     -fixed yes      -DIRECTION Output
set_io  {PMOD7[7]}      -pinname 91     -fixed yes      -DIRECTION Output
set_io  {PMOD7[8]}      -pinname 92     -fixed yes      -DIRECTION Output

set_io  {PMOD8[1]}      -pinname 93     -fixed yes      -DIRECTION Output
set_io  {PMOD8[2]}      -pinname 94     -fixed yes      -DIRECTION Output
set_io  {PMOD8[3]}      -pinname 95     -fixed yes      -DIRECTION Output
set_io  {PMOD8[4]}      -pinname 96     -fixed yes      -DIRECTION Output
set_io  {PMOD8[5]}      -pinname 97     -fixed yes      -DIRECTION Output
set_io  {PMOD8[6]}      -pinname 98     -fixed yes      -DIRECTION Output
# CANNOT BE USED ON AGLN250 ! #set_io  PM8_7 -pinname 99 -fixed yes -DIRECTION Input
# CANNOT BE USED ON AGLN250 ! #set_io  PM8_8 -pinname 100 -fixed yes -DIRECTION Input
```



Kart Motherboard (Battery, PMOD, BLE)

Hes·so VALAIS WALLIS

School of Engineering

Author: Amand Axel, Silvan Zahno
Date: March 20, 2023
Version: v1.0

1 Overview

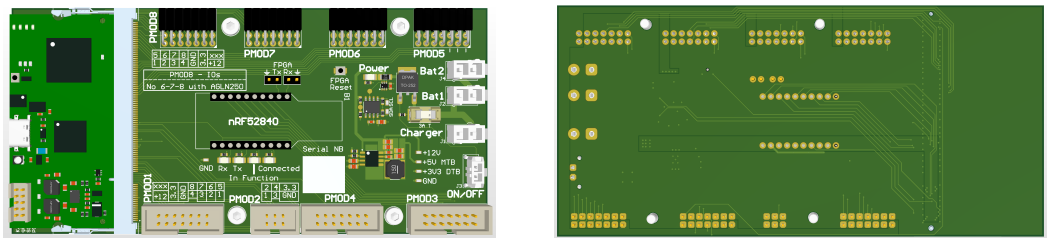


Figure 1: PCB board

The boards embeds the following functionalities:

- Battery connectors for 2 * 6V packs (Tamiya side-connector type), 3A fuse-protected
- Charger input connector (Tamiya side-connector type)
- Switch input to power the board on
- IIC DAC for battery voltage and current readings
- Four standard expanded dual PMOD connectors (PMOD + +12V pin)
- Three flat-cable expanded dual PMOD connectors (PMOD + +12V pin)
- One flat-cable single PMOD connector
- Connector for a nRF52840 BLE-USB dongle, with two headers for UART sniffing and four state LEDs
- Extra reset button for the FPGA

Technical files

The schematic of the board is given under I Schematic.

Rooting is available under II Rooting *(open the page with Inkscape for layers)*.

The bill of material is given under III BOM.

2 Specifications

2.1 Overview

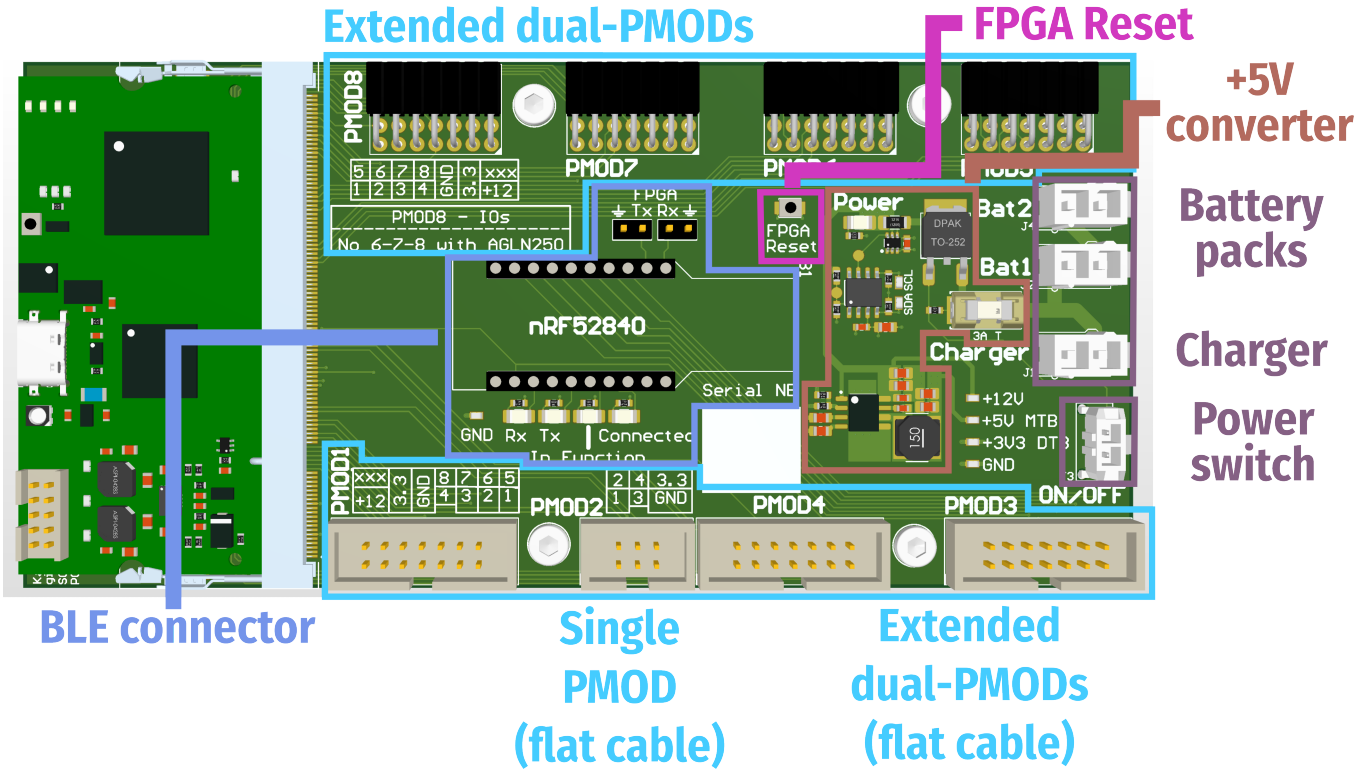


Figure 2: Card overview

2.2 Supply

The board is powered thanks to two Ni-Cd +6V packs connected on J2 and J4:

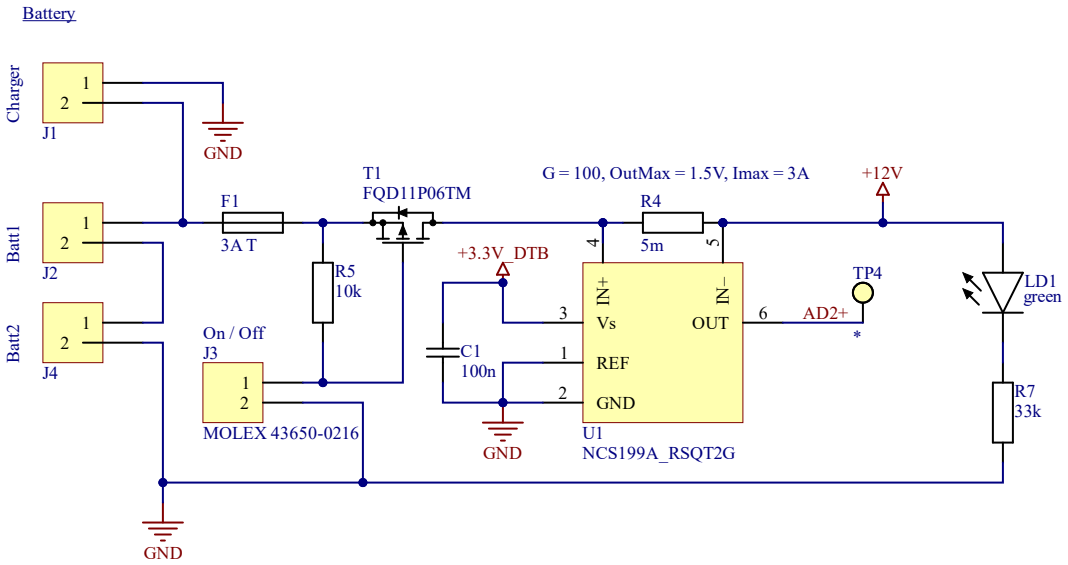


Figure 3: Battery power

If powered through the SODIMM card, any extension card requiring +12V and the nRF52840 dongle will not be working.

The circuit is protected with a 3A slow fuse and is switched on and off with a button on J3. By default (no switch or it being open) the transistor does not conduct.

The circuit contains no reverse voltage protection. Be sure the batteries are oriented correctly before connecting them.

Connector J1 allows for an external charger to be plugged in and recharge the two Ni-Cd packs directly.

Be sure to power off the rest of the circuit. Charging may induce greater voltage than the pack itself. Never exceed a charge rate of 1C (gives roughly 80% of the battery capacity), and prefer a rate of C/10.

The polarities of the batteries and charger connectors are the following:

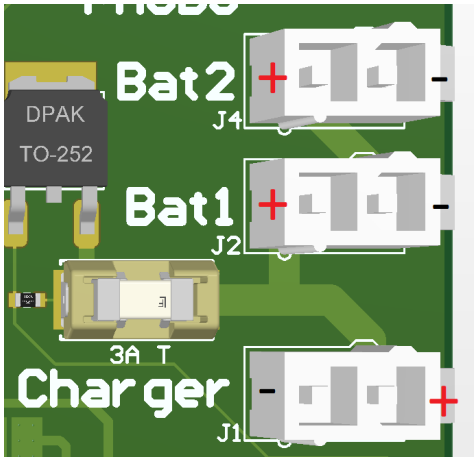


Figure 4: Batteries and charger polarities

The voltage is then stepped down to power both the daughterboard and the nRF52840 dongle with +5V through the BD9D321EFJ buck converter. V_{out} is set through R8, R9 and L1, the values being given in the datasheet:

$$V_{in} = +12[V], V_{out} = +5[V] => R8 = 123.3[k], R9 = 22[k], L1 = 3.3[uH].$$

2.3 Battery status

The battery voltage and current are read through the MCP3426A0 ADC converter. Since the chip can read differential signals from $\pm V_{ref} = \pm 2.048[V]$ and both our values are positive only, the measuring scale will miss 1 bit (MSB always 0 = always positive). The input must never be greater than V_{ref}.

In the tester 3 SW Configuration, the convertion rate is set to 60 samples per second (14 bits, i.e. 13 useful bits) with a gain of *1.

The voltage is read through a resistor bridge with a gain of $G = \frac{R6}{R6+R3} = \frac{10k}{68k+10k} = 0.128$. With two Ni-Cd packs of 5 cells each the maximum voltage is $V_{ch1_{max}} = V_{12V_{max}} * G = 2*5*1.3*0.128 = 1.66[V]$.

The current is read on a 5 mΩ shunt resistor with the NCS199A2R shunt-monitor (see U1 on 2.2 Supply). Its internal gain is 100*, which gives a maximal output for 3 A of $V_{ad2} = I_{max} * R_{shunt} * G = 3 * 5m * 100 = 1.5[V]$. The maximal voltage on the resistor is $V_{shunt_{max}} = R_{shunt} * I_{max} = 5m * 3 = 15[mV]$ with a dissipated power of $P_{shunt_{max}} = R_{shunt} * I^2_{max} = 5m * 3 * 3 = 45[mW]$.

2.4 PMOD connectors

The PMOD connector present on the board are slightly modified to include an extra +12V pin.

The female socket, front faced, contains the following pins:

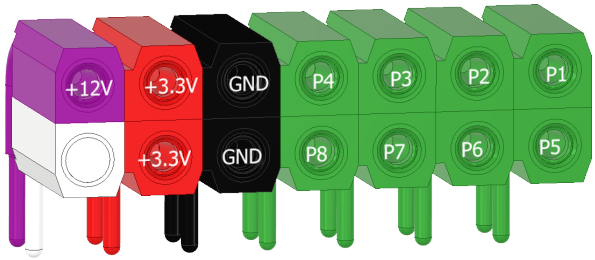


Figure 5: Modified PMODs pining

The flat cable versions have the following pinings (as seen from top):

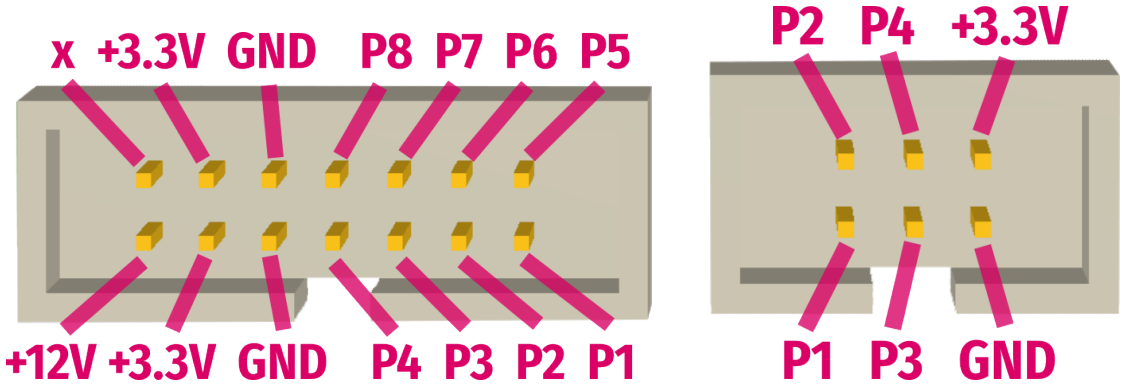


Figure 6: Modified flat-cable PMODs pining

Extra precaution must be taken not to plug extension boards in the wrong pins/direction !

2.5 Bluetooth dongle

Pins B1 are meant to host a nRF52840 BLE-USB dongle from Nordic Semiconductor:

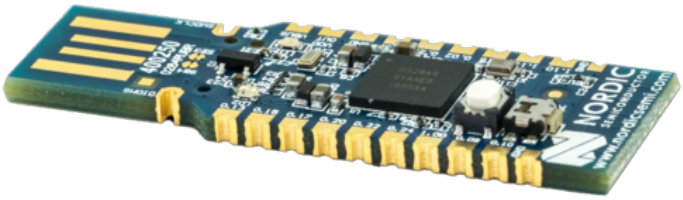


Figure 7: nRF52840 dongle

The dongle is powered by +5 V and creates its own +3.3 V rail which is used for the rest of the I/Os.

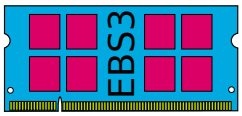
It has four dedicated LEDs which can be used to display its current status.

The module communicates with the FPGA through a dedicated UART line and can also access pins of the PMOD3 connector (either to control an external module itself, interact with the FPGA, add flow control to the UART line ...).

Finally, he also gets the **nPOR** signal as input to detect system reset.

2.6 Reset

The button labeled **FPGA Reset** allows to reset the FPGA by setting the **nRESET_IN** signal low.



3 SW Configuration

The board is used with Libero Libero SoC (Microsemi).

3.1 Board configuration

The default configuration (constraints file) for the board is available under IV Constraints file.

- Bank voltages are set with:
  `set_iobank Bank1 -vcci 3.3 -fixed yes` (sets bank to 3.3V)
- I/Os are declared as:
```
set_io Rx_BLE -pinname 3 -fixed yes -DIRECTION Input -RES_PULL Up
set_io {PMOD1[1]} -pinname 8 -fixed yes -DIRECTION Output
set_io SDA_battery -pinname 5 -fixed yes -DIRECTION InOut
```
  - set_io: io name in the VHDL file, use {Name[IDX]} for vectors
  - pinname: the IO number
  - DIRECTION: Input, Output or InOut
  - RES_PULL: Up, Down, or omit the argument for none

3.2 Test project

The tester is planned for 10 MHz clocks.

A HDL Designer test project is available with the following:

- PMODs I/Os (except PMOD3 pins 2-4-6-7) will blink at either 10, 2 or 1 Hz.
- The ADC is setup then periodically read (each refresh (1 s) toggles the embedded yellow LED, the green LED indicates an idle state (waiting for the second to pass) and the red one is on when current is not zero).
- The FTDI UART Tx signal is redirected to the BLE device, while the BLE Tx is AND with the previously read measurement (both should not emit simultaneously to avoid scrambled data).
- The SCL and SDA in-out internal signals are redirected to PMOD3 pins 2-4-6-7 for debug measurements.

The blinking PMOD part is the following:

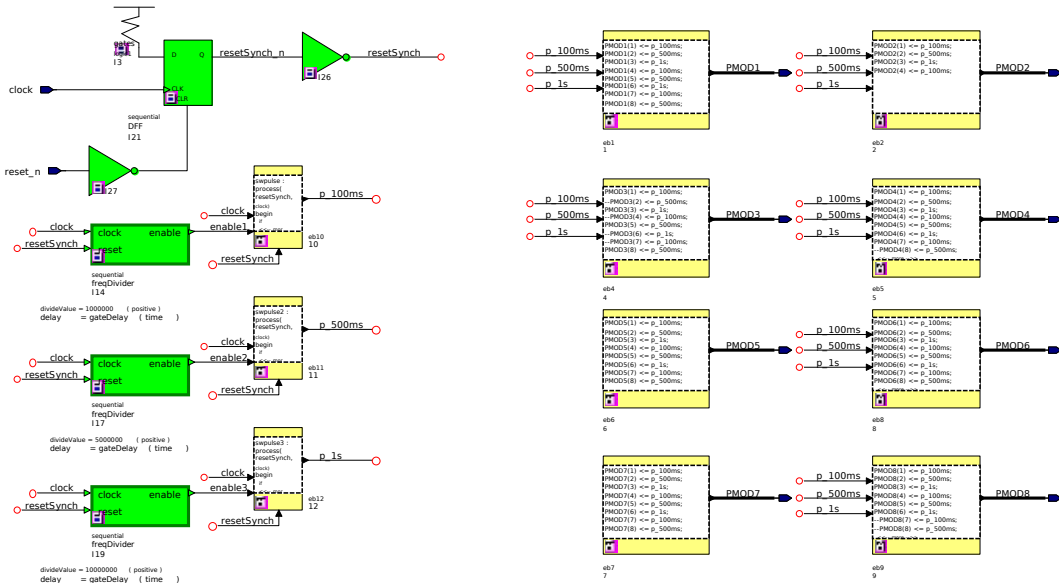


Figure 8: Tester - blinkers

The ADC measurements and UART handling are made in the following:

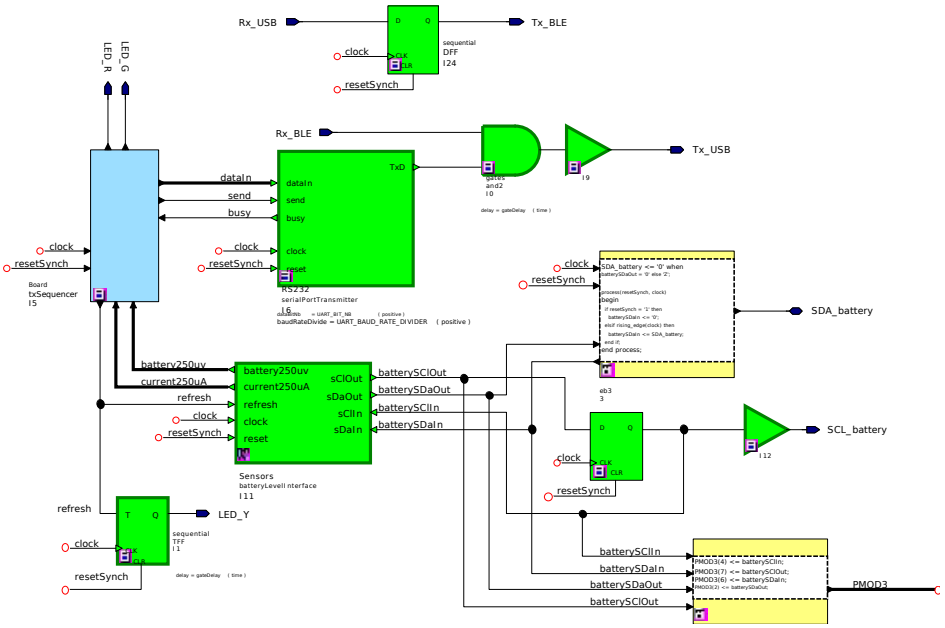


Figure 9: Tester - ADC and UART

The **batteryLevelInterface** is composed as:

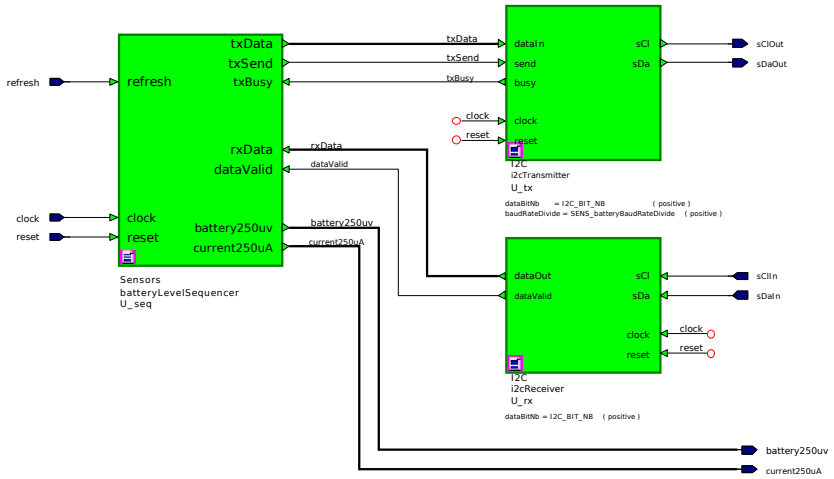


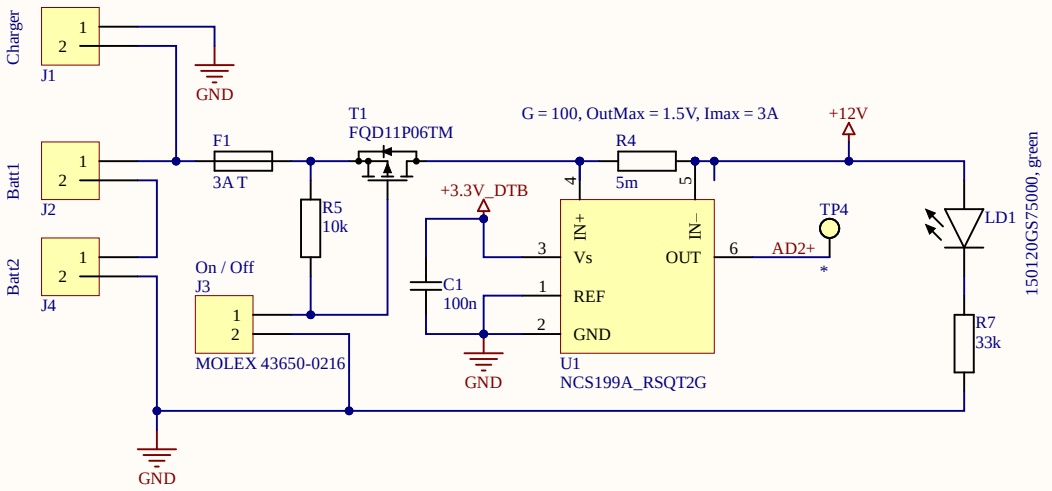
Figure 10: Tester - blinkers

Generic IIC transmitter and receivers are read from the **batteryLevelSequencer** block to setup and read from the ADC. The related code is given in appendix V Battery Level Interface - VHDL: after the **refresh** signal blinks, the **sendI2cCommands** process setup the read channel, reads, redo the process for the second channel, while the **readFSM** and **storeData** processes register the data to internal registers.

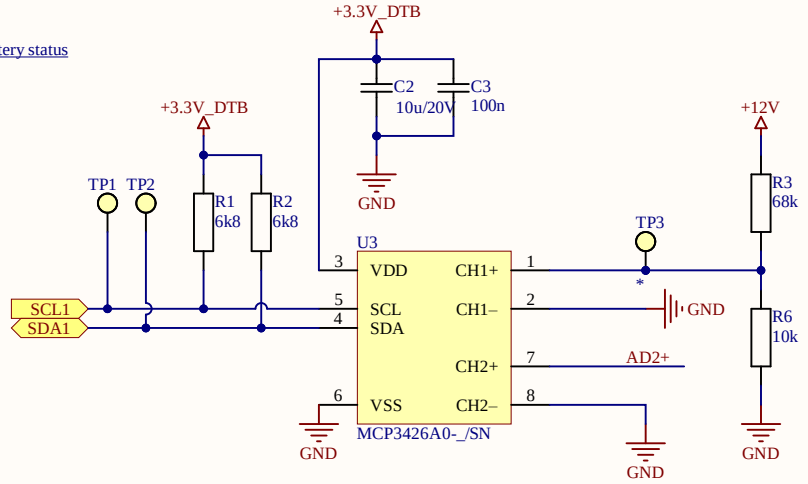
The Tx sequencer code is given in appendix VI Tx Sequencer - VHDL: it waits 1 second, toggle the refresh and wait another 100 ms, then send " **Voltage :** " followed by the ASCII-converted value of the voltage, same then for the current and loop as is indefinitely.

I Schematic

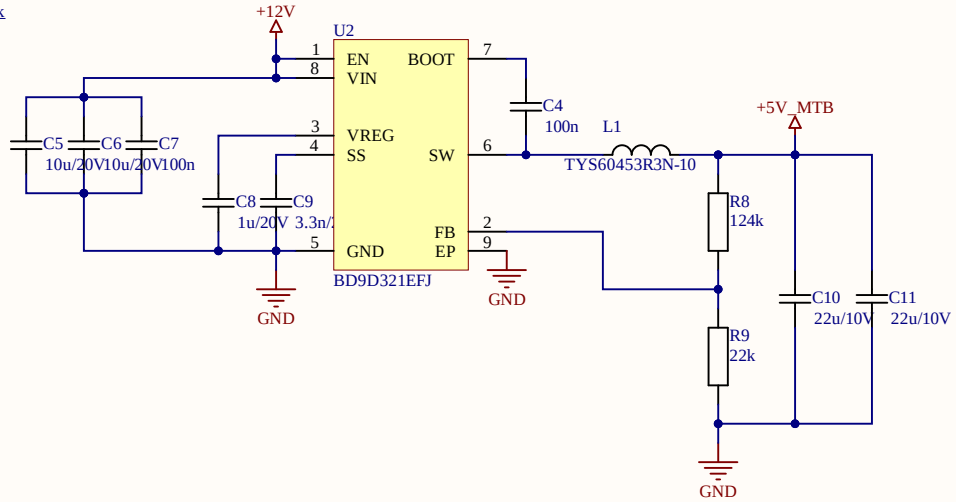
Battery



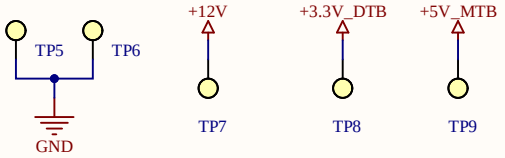
Battery status



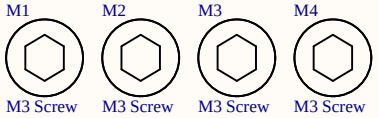
Buck



Testpoints

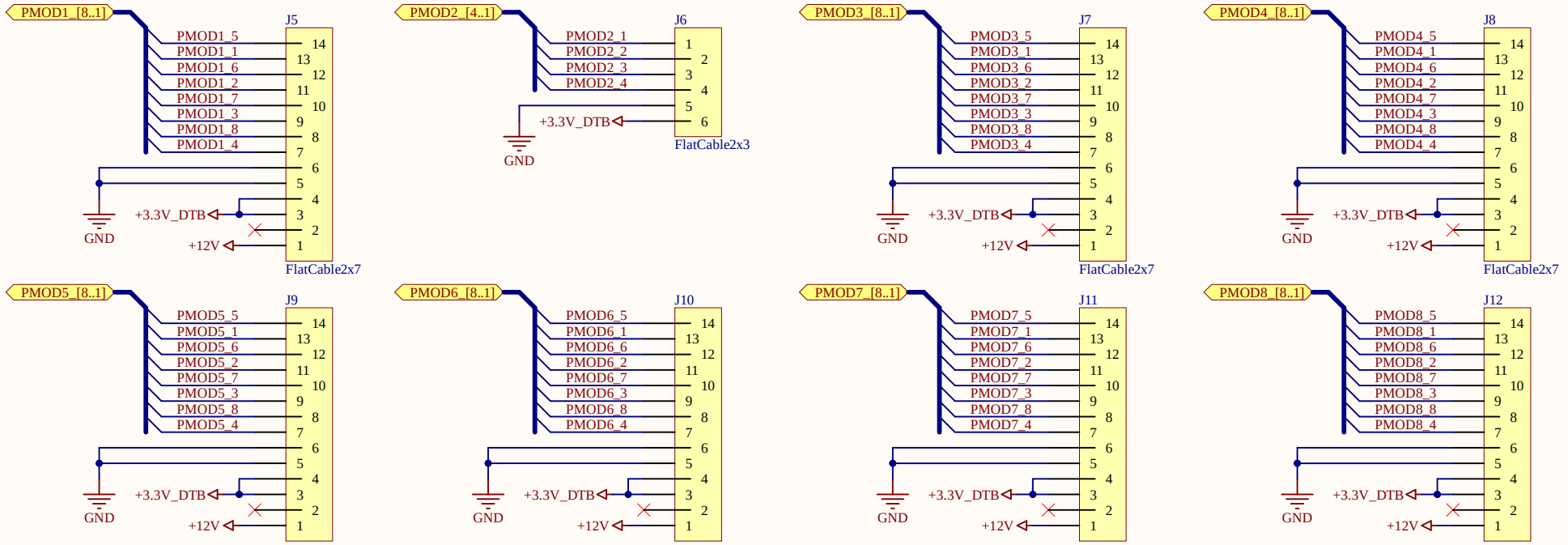


Other

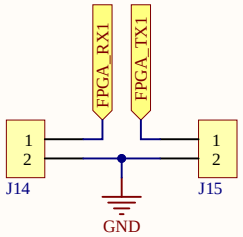


Kart_Mobo.PrjPcb		Hes·so VALAIS WALLIS
Power.SchDoc		Date : 14.06.2022
Revision : 1.1	Sheet 1 of 3	Design by : AmA
C:\dev\ebs3\10_PCB\02_Kart_Mobo\Kart_Mobo\Power.SchDoc		

PMODs



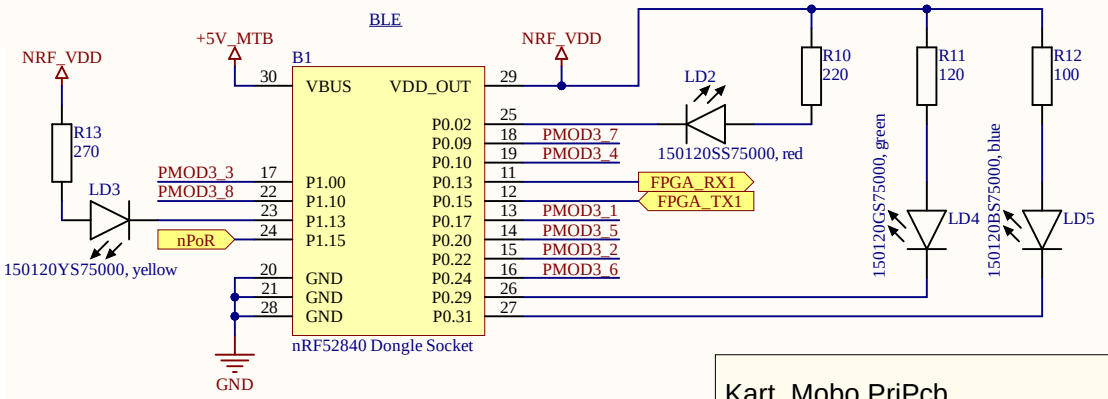
UART sniffer



nRF IO descr.
Low freq = max 10kHz

P0.02 (25) : low freq.
P0.09 (18) : low freq.
P0.10 (19) : low freq.
P0.13 (11) : IO
P0.15 (12) : IO
P0.17 (13) : IO
P0.20 (14) : IO
P0.22 (15) : IO
P0.24 (16) : IO
P0.29 (26) : low freq.
P0.31 (27) : low freq.
P1.00 (17) : IO, SWO
P1.10 (22) : low freq.
P1.13 (23) : low freq.
P1.15 (24) : low freq.

BLE



Kart_Mobo.PrjPcb		Hes·so VALAIS WALLIS
IOs.SchDoc		Date : 14.06.2022
Revision : 1.1	Sheet 2 of 3	Design by : AmA
C:\dev\ebs3\10_PCB\02_Kart_Mobo\Kart_Mobo\IOs.SchDoc		

SODIMM-200 connections

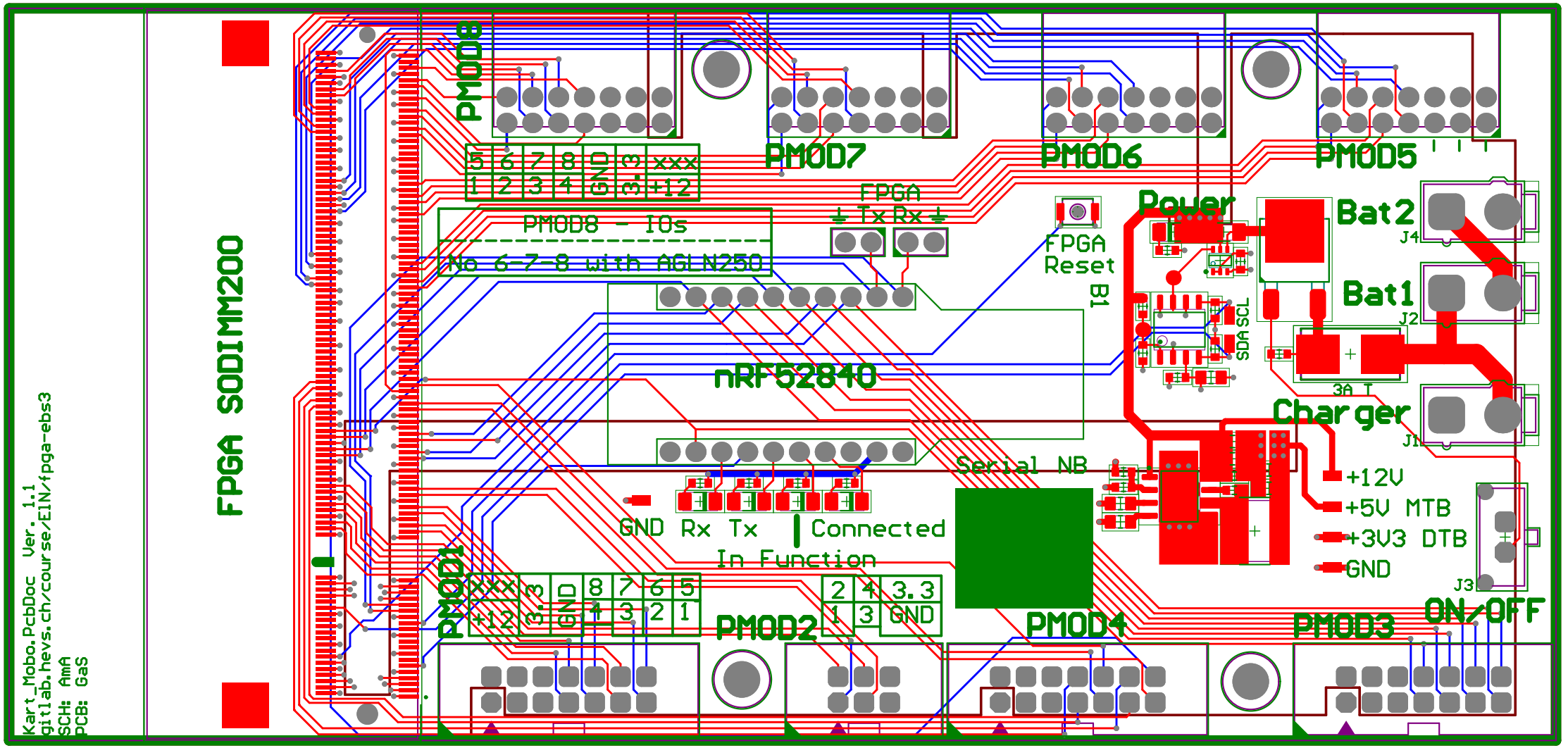
SODIMM200 Male (Board)

J16
1473149-1

Not using extra JTAG

Kart_Mobo.PrjPcb		Hes·so VALAIS WALLIS
SODIMM200.SchDoc		Date : 14.06.2022
Revision : 1.1	Sheet 3 of 3	Design by : AmA
C:\dev\ebs3\10_PCB\02_Kart_Mobo\Kart_Mobo\SODIMM200.SchDoc		

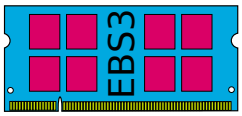
II Rooting


FPGA SODIMM200
Kart_Mobo.PcbDoc Ver. 1.1
gitlab.hevs.ch/course/EIN/fpga-ebs3
SCH: AmA
PCB: GaS
PMOD8
PMOD7
PMOD6
PMOD5
PMOD8 - IOs
No 6-7-8 with AGLN250
FPGA
Tx Rx
FPGA Reset
B1
Power
Bat2
J4
Bat1
J2
SDA SCL
3A T
Charger
J1
nRF52840
Serial NB
GND Rx Tx
Connected
In Function
+12V
+5V MTB
+3V3 DTB
GND
J3
ON/OFF
PMOD1
PMOD2
PMOD4
PMOD3

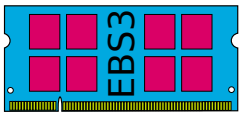
III BOM

Name	Description	Designator	Quantity	Footprint
nRF52840 Dongle Socket	nRF52840 Dongle Socket	B1	1	nRF52840 Dongle socket
100n	Capacitor	C1, C3, C4, C7	4	CAPACITOR 1608 REFLOW
10u/20V	Capacitor	C2, C5, C6	3	CAPACITOR 2012 REFLOW
1u/20V	Capacitor	C8	1	CAPACITOR 2012 REFLOW
3.3n/20V	Capacitor	C9	1	CAPACITOR 2012 REFLOW
22u/10V	Capacitor	C10, C11	2	CAPACITOR 2012 REFLOW
3A T	LittleFuse series 0154 C10X5.0 H3.8	F1	1	LittleFuse series 0154 C10X5.0 H3.8
	Mini-Fit Jr. Header, Vertical, 2 Circuits	J1, J2, J4	3	MOLEX 39-28-8020
MOLEX 43650-0216	Micro-Fit 3.0 Vertical Header, 3.00mm Pitch, Single Row, 2 Circuits	J3	1	MOLEX 43650-0216
FlatCable2x7	FlatCable2x7	J5, J7, J8	3	FlatCable2x7 2.54MM Straight TH
FlatCable2x3	FlatCable2x3	J6	1	FlatCable2x3 2.54MM Straight TH
Socket2x7	Receptacle connector	J9, J10, J11, J12	4	Socket2X7 P2.54 RIGHT ANGLE
	Plug connector	J14, J15	2	PinHeader1x2 P2.54
1473149-1	DDR1 & DDR2 SODIMM Socket 0.6mm pitch 200 Positions	J16	1	SO-DIMM TE 1473149-1
TYS60453R3N-10	Inductor	L1	1	TAIYO YUDEN NR6028T
150120GS75000, green	LED generic	LD1, LD4	2	LED 3216 REFLOW
150120SS75000, red	LED generic	LD2	1	LED 3216 REFLOW
150120YS75000, yellow	LED generic	LD3	1	LED 3216 REFLOW
150120BS75000, blue	LED generic	LD5	1	LED 3216 REFLOW
M3 Screw	Screw Hex Socket Unconnected BN610	M1, M2, M3, M4	4	Screw Hex Socket M3X5 Unconnected
6k8	Resistor	R1, R2	2	RESISTOR 1608 REFLOW
68k	Resistor	R3	1	RESISTOR 1608 REFLOW
5m	Resistor	R4	1	RESISTOR 3216 REFLOW
10k	Resistor	R5, R6	2	RESISTOR 1608 REFLOW
33k	Resistor	R7	1	RESISTOR 1608 REFLOW
124k	Resistor	R8	1	RESISTOR 1608 REFLOW
22k	Resistor	R9	1	RESISTOR 1608 REFLOW
220	Resistor	R10	1	RESISTOR 1608 REFLOW
120	Resistor	R11	1	RESISTOR 1608 REFLOW
100	Resistor	R12	1	RESISTOR 1608 REFLOW
270	Resistor	R13	1	RESISTOR 1608 REFLOW
	Push Button, 2 pins	S1	1	Push Button C2.5X3.0 OMRON B3U-1000P(M)-B
FQD11P06TM	Generic MOS-P	T1	1	TO252AA-2 DPAK P2.29 C6.2x6.7 H2.4
	Test Point	TP1, TP2, TP5, TP6, TP8, TP9	6	Test Point ARWIN S2761-46R
S2761-46R	Test Point	TP7	1	Test Point ARWIN S2761-46R
NCS199A_RSQT2G	NCS199A1R	U1	1	SC70-6 P0.65 C1.4x2.2 H1.0
BD9D321EFJ	Integrated MOSFET Buck DC/DC Converter	U2	1	BD9D321EFJ
MCP3426A0-_/SN	16-Bit, Multi-Channel ΔΣ Analog-to-Digital Converter	U3	1	SO8 P1.27 C4.0X5.0 H1.75

IV Constraints file

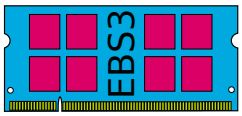



```
# Actel Physical design constraints file
# Family: IGLOO , Die: AGLN125, AGLN250, AGL600

#--------------------------------------------------------------------------
# Clocks and reset
#
set_io  clock           -pinname 65     -fixed yes      -DIRECTION Input
set_io  reset_n         -pinname 21     -fixed yes      -DIRECTION Input

#--------------------------------------------------------------------------
# I/O - DOBO
#
set_io  LED_R           -pinname 26     -fixed yes      -DIRECTION Output
set_io  LED_Y           -pinname 23     -fixed yes      -DIRECTION Output
set_io  LED_G           -pinname 22     -fixed yes      -DIRECTION Output

#--------------------------------------------------------------------------
# UART (USB)
#
set_io  Tx_USB           -pinname 7      -fixed yes      -DIRECTION Output
set_io  Rx_USB           -pinname 6      -fixed yes      -DIRECTION Input   -RES_PULL Up

##############################################################################
# IO banks setting
#
set_iobank Bank1         -vcci 3.3       -fixed yes
set_iobank Bank0         -vcci 3.3       -fixed yes

##############################################################################
# MOBO specific
#
set_io  SDA_battery     -pinname 5      -fixed yes      -DIRECTION InOut
set_io  SCL_battery     -pinname 4      -fixed yes      -DIRECTION Output

set_io  Tx_BLE          -pinname 2      -fixed yes      -DIRECTION Output
set_io  Rx_BLE          -pinname 3      -fixed yes      -DIRECTION Input   -RES_PULL Up

set_io  {PMOD1[1]}      -pinname 8      -fixed yes      -DIRECTION Output
set_io  {PMOD1[2]}      -pinname 10     -fixed yes      -DIRECTION Output
set_io  {PMOD1[3]}      -pinname 11     -fixed yes      -DIRECTION Output
set_io  {PMOD1[4]}      -pinname 13     -fixed yes      -DIRECTION Output
set_io  {PMOD1[5]}      -pinname 15     -fixed yes      -DIRECTION Output
set_io  {PMOD1[6]}      -pinname 16     -fixed yes      -DIRECTION Output
set_io  {PMOD1[7]}      -pinname 19     -fixed yes      -DIRECTION Output
set_io  {PMOD1[8]}      -pinname 20     -fixed yes      -DIRECTION Output

set_io  {PMOD2[1]}      -pinname 27     -fixed yes      -DIRECTION Output
set_io  {PMOD2[2]}      -pinname 28     -fixed yes      -DIRECTION Output
set_io  {PMOD2[3]}      -pinname 29     -fixed yes      -DIRECTION Output
set_io  {PMOD2[4]}      -pinname 30     -fixed yes      -DIRECTION Output

set_io  {PMOD3[1]}      -pinname 31     -fixed yes      -DIRECTION Output
set_io  {PMOD3[2]}      -pinname 32     -fixed yes      -DIRECTION Output
set_io  {PMOD3[3]}      -pinname 33     -fixed yes      -DIRECTION Output
set_io  {PMOD3[4]}      -pinname 34     -fixed yes      -DIRECTION Output
set_io  {PMOD3[5]}      -pinname 35     -fixed yes      -DIRECTION Output
set_io  {PMOD3[6]}      -pinname 36     -fixed yes      -DIRECTION Output
set_io  {PMOD3[7]}      -pinname 40     -fixed yes      -DIRECTION Output
set_io  {PMOD3[8]}      -pinname 41     -fixed yes      -DIRECTION Output
```

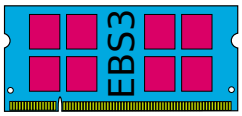



```
set_io  {PMOD4[1]}      -pinname 42     -fixed yes      -DIRECTION Output
set_io  {PMOD4[2]}      -pinname 43     -fixed yes      -DIRECTION Output
set_io  {PMOD4[3]}      -pinname 44     -fixed yes      -DIRECTION Output
set_io  {PMOD4[4]}      -pinname 45     -fixed yes      -DIRECTION Output
# CANNOT BE USED ON AGLN250 ! AGLN125 and AGL600 OK ! #set_io  PM4_5 -pinname 46  -fixed yes
set_io  {PMOD4[5]}      -pinname 57     -fixed yes      -DIRECTION Output
set_io  {PMOD4[6]}      -pinname 58     -fixed yes      -DIRECTION Output
set_io  {PMOD4[7]}      -pinname 59     -fixed yes      -DIRECTION Output

set_io  {PMOD5[1]}      -pinname 60     -fixed yes      -DIRECTION Output
set_io  {PMOD5[2]}      -pinname 61     -fixed yes      -DIRECTION Output
set_io  {PMOD5[3]}      -pinname 62     -fixed yes      -DIRECTION Output
set_io  {PMOD5[4]}      -pinname 63     -fixed yes      -DIRECTION Output
set_io  {PMOD5[5]}      -pinname 64     -fixed yes      -DIRECTION Output
set_io  {PMOD5[6]}      -pinname 69     -fixed yes      -DIRECTION Output
set_io  {PMOD5[7]}      -pinname 70     -fixed yes      -DIRECTION Output
set_io  {PMOD5[8]}      -pinname 71     -fixed yes      -DIRECTION Output

set_io  {PMOD6[1]}      -pinname 72     -fixed yes      -DIRECTION Output
set_io  {PMOD6[2]}      -pinname 73     -fixed yes      -DIRECTION Output
set_io  {PMOD6[3]}      -pinname 76     -fixed yes      -DIRECTION Output
set_io  {PMOD6[4]}      -pinname 77     -fixed yes      -DIRECTION Output
set_io  {PMOD6[5]}      -pinname 78     -fixed yes      -DIRECTION Output
set_io  {PMOD6[6]}      -pinname 79     -fixed yes      -DIRECTION Output
set_io  {PMOD6[7]}      -pinname 80     -fixed yes      -DIRECTION Output
set_io  {PMOD6[8]}      -pinname 81     -fixed yes      -DIRECTION Output

set_io  {PMOD7[1]}      -pinname 82     -fixed yes      -DIRECTION Output
set_io  {PMOD7[2]}      -pinname 83     -fixed yes      -DIRECTION Output
set_io  {PMOD7[3]}      -pinname 84     -fixed yes      -DIRECTION Output
set_io  {PMOD7[4]}      -pinname 85     -fixed yes      -DIRECTION Output
set_io  {PMOD7[5]}      -pinname 86     -fixed yes      -DIRECTION Output
set_io  {PMOD7[6]}      -pinname 90     -fixed yes      -DIRECTION Output
set_io  {PMOD7[7]}      -pinname 91     -fixed yes      -DIRECTION Output
set_io  {PMOD7[8]}      -pinname 92     -fixed yes      -DIRECTION Output

set_io  {PMOD8[1]}      -pinname 93     -fixed yes      -DIRECTION Output
set_io  {PMOD8[2]}      -pinname 94     -fixed yes      -DIRECTION Output
set_io  {PMOD8[3]}      -pinname 95     -fixed yes      -DIRECTION Output
set_io  {PMOD8[4]}      -pinname 96     -fixed yes      -DIRECTION Output
set_io  {PMOD8[5]}      -pinname 97     -fixed yes      -DIRECTION Output
set_io  {PMOD8[6]}      -pinname 98     -fixed yes      -DIRECTION Output
# CANNOT BE USED ON AGLN250 ! AGLN125 and AGL600 OK ! #set_io  PM8_7 -pinname 99 -fixed yes -
# CANNOT BE USED ON AGLN250 ! AGLN125 and AGL600 OK ! #set_io  PM8_8 -pinname 100 -fixed yes
```

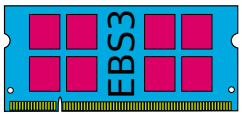
V Battery Level Interface - VHDL



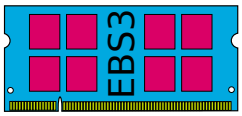

```
--
-- VHDL Architecture Sensors.batteryLevelSequencer.rtl
--
-- Created:
--          by - axel.amand.UNKNOWN (WE7860)
--          at - 17:19:58 19.05.2022
--
-- using Mentor Graphics HDL Designer(TM) 2019.2 (Build 5)
--

library Common;
  use Common.CommonLib.all;

library Kart;
  use Kart.Kart.all;

ARCHITECTURE rtl OF batteryLevelSequencer IS

  constant sequenceLength : positive := 2*4 + 2*6 + 2*1;
  signal sequenceCounter: unsigned(requiredBitNb(sequenceLength)-1 downto 0);

  constant i2cWordBitNb: positive := 8;
  subtype i2cWordType is std_ulogic_vector(i2cWordBitNb-1 downto 0);
  constant i2cStart                     : i2cWordType := X"00";
  constant i2cStop                      : i2cWordType := X"FF";
  constant i2cRead                      : i2cWordType := X"FF";
  -- MCP3426
  constant mcpAddress              : i2cWordType := "11010000";

  -- Configuration register
  constant rdy : std_ulogic := '1';
    -- channel
  constant channel1 : std_ulogic_vector(1 downto 0) := "00";
  constant channel2 : std_ulogic_vector(1 downto 0) := "01";
    -- conversion mode, 1 = continuous, 0 = one-shot
  constant conversionMode : std_ulogic := '0';
    -- SR : 00 = 240 SPS (12bits), 01 = 60 SPS (14 bits), 10 = 15 SPS (16 bits)
    -- Check datasheet for how data is handled and modify storeData process
  constant sampleRate : std_ulogic_vector(1 downto 0) := "01";
    -- 00 = x1, 01 = x2, 10 = x4, 11 = x8
  constant gain : std_ulogic_vector(1 downto 0) := "00";

  -- Voltage (2+1B) and current (2+1B)
  constant readLength : positive := 4; -- 2+1B + null state

  signal startStop: std_ulogic;
  signal i2cWord: i2cWordType;
  signal ack: std_ulogic;


  signal readCounter: unsigned(requiredBitNb(readLength)-1 downto 0);
  signal p_volt_high, p_curr_high, p_volt_low, p_curr_low :
    unsigned(i2cWordBitNb-1 downto 0) := (others=>'0');
  signal p_volt, p_curr : unsigned(2*i2cWordBitNb-1 downto 0);

  -- Manages bad reads
  signal p_redo_volt, p_orv, p_redo_curr, p_orc : std_ulogic;
  signal p_ok : std_ulogic;
  signal p_retries : unsigned(requiredBitNb(SENS_BATT_READ_RETRIES)-1 downto 0);
```



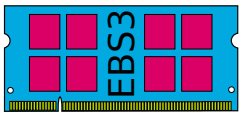

```
60
61    -- Manages timeout before read (measure time)
62    signal p_waitTimeout, p_timed_out : std_ulogic;
63    constant CNT_TARGET : positive :=
64      positive(CLOCK_FREQUENCY * real(SENS_BATT_READ_TMOUT_MS) / 1000.0);
65    signal p_waitTimeout_cnt : unsigned(requiredBitNb(CNT_TARGET) - 1 downto 0);
66
67  BEGIN
68
69    p_timed_out <= '1' when
70      p_waitTimeout_cnt >= to_unsigned(CNT_TARGET, p_waitTimeout_cnt'length)
71      else '0';
72
73    --============================================================================
74                                                              -- count send sequence
75    countSequence: process(reset, clock)
76    begin
77      if reset = '1' then
78        sequenceCounter <= (others => '0');
79        p_retries <= (others=>'0');
80        p_orv <= '0';
81        p_orc <= '0';
82        p_volt <= (others=>'0');
83        p_curr <= (others=>'0');
84        p_waitTimeout_cnt <= (others=>'0');
85      elsif rising_edge(clock) then
86        if sequenceCounter = 0 then
87          p_retries <= (others=>'0');
88          if refresh = '1' then
89            sequenceCounter <= to_unsigned(1,
    ↪  sequenceCounter'length);--sequenceCounter + 1;
90          end if;
91        else
92          if p_waitTimeout = '0' then
93            p_waitTimeout_cnt <= (others=>'0');
94          else
95            p_waitTimeout_cnt <= p_waitTimeout_cnt + 1;
96          end if;
97
98          if txBusy = '0' then
99            p_orv <= p_redo_volt;
100           p_orc <= p_redo_curr;
101           -- Check for retries
102           if p_redo_volt = '1' and p_orv = '0' then
103               p_retries <= p_retries + 1;
104               sequenceCounter <= to_unsigned(6, sequenceCounter'length);
105               if p_retries >= to_unsigned(SENS_BATT_READ_RETRIES,
    ↪  p_retries'length) then
106                 sequenceCounter <= (others => '0');
107               end if;
108           elsif p_redo_curr = '1' and p_orc = '0' then
109               p_retries <= p_retries + 1;
110               sequenceCounter <= to_unsigned(17, sequenceCounter'length);
111               if p_retries >= to_unsigned(SENS_BATT_READ_RETRIES,
    ↪  p_retries'length) then
112                 sequenceCounter <= (others => '0');
113               end if;
114           elsif sequenceCounter < sequenceLength then
115             if p_waitTimeout = '0' or
```




```
116                 (p_waitTimeout = '1' and p_timed_out = '1') then
117                 if sequenceCounter = 12 then
118                   p_volt <= "00" & p_volt_high(i2cWordBitNb-3 downto 0) &
    ↪  p_volt_low;
119                   p_retries <= (others=>'0');
120                 end if;
121                 sequenceCounter <= sequenceCounter + 1;
122               end if;
123             else
124               p_curr <= "00" & p_curr_high(i2cWordBitNb-3 downto 0) & p_curr_low;
125               sequenceCounter <= (others => '0');
126             end if;
127           end if;
128         end if;
129       end if;
130     end process countSequence;
131
132     --============================================================================
133                                                                 -- send I2C commands
134     sendI2cCommands: process(sequenceCounter)
135     begin
136       startStop <= '0';
137       i2cWord <= (others => '0');
138       ack <= '1';
139       p_waitTimeout <= '0';
140       case to_integer(sequenceCounter) is
141           -- Setup channel 1
142         when  1 => startStop <= '1'; i2cWord <= i2cStart;
143         when  2 => i2cWord <= mcpAddress;
144         when  3 => i2cWord <= rdy & channel1 & conversionMode & sampleRate & gain;
145         when  4 => startStop <= '0'; i2cWord <= i2cStop;
146           -- Wait
147         when  5 => p_waitTimeout <= '1';
148           -- Read channel 1
149         when  6 => startStop <= '1'; i2cWord <= i2cStart;
150         when  7 => i2cWord <= mcpAddress or X"01";
151         when  8 => i2cWord <= i2cRead; ack <= '0';
152         when  9 => i2cWord <= i2cRead; ack <= '0';
153         when  10 => i2cWord <= i2cRead; ack <= '0';
154         when  11 => startStop <= '0'; i2cWord <= i2cStop;
155           -- Setup channel 2
156         when  12 => startStop <= '1'; i2cWord <= i2cStart;
157         when  13 => i2cWord <= mcpAddress;
158         when  14 => i2cWord <= rdy & channel2 & conversionMode & sampleRate & gain;
159         when  15 => startStop <= '0'; i2cWord <= i2cStop;
160           -- Wait
161         when  16 => p_waitTimeout <= '1';
162           -- Read channel 2
163         when  17 => startStop <= '1'; i2cWord <= i2cStart;
164         when  18 => i2cWord <= mcpAddress or X"01";
165         when  19 => i2cWord <= i2cRead; ack <= '0';
166         when  20 => i2cWord <= i2cRead; ack <= '0';
167         when  21 => i2cWord <= i2cRead; ack <= '0';
168         when  22 => startStop <= '0'; i2cWord <= i2cStop;
169         when others => null;
170       end case;
171     end process sendI2cCommands;
172
173     txData <= startStop & ack & i2cWord;
```



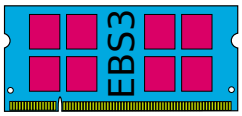

```
174     txSend <= '1' when (sequenceCounter > 0) and (txBusy = '0')
175       else '0';
176
177     --============================================================================
178                                                             -- read data from I2C
179     ----------------------------------------------------------------------------
180                                           -- read counter increments with input data
181     readFSM: process(reset, clock)
182     begin
183       if reset = '1' then
184         readCounter <= (others => '0');
185       elsif rising_edge(clock) then
186         -- start condition clears counter
187         if (dataValid = '1') and (rxData(rxData'high) = '1') and
188             (rxData(rxData'high-1) = '0') then
189           readCounter <= (others => '0');
190         -- chip address starts counting
191         elsif readCounter = 0 then
192           if (dataValid = '1') and (rxData(rxData'high-2 downto 0) =
193               (mcpAddress or X"01")) then
194             readCounter <= readCounter + 1;
195           end if;
196         else
197           if dataValid = '1' then
198             -- acknowledge defines when to stop counter
199             if rxData(rxData'high-1) = '0' then
200               readCounter <= readCounter + 1;
201             else
202               readCounter <= (others => '0');
203             end if;
204           end if;
205         end if;
206       end if;
207     end process readFSM;
208
209     ----------------------------------------------------------------------------
210                                         -- store all high and then all low data bytes
211     storeData: process(reset, clock)
212     begin
213       if reset = '1' then
214         p_volt_high <= (others=>'0');
215         p_curr_high <= (others=>'0');
216         p_redo_volt <= '0';
217         p_redo_curr <= '0';
218       elsif rising_edge(clock) then
219         if dataValid = '1' then
220           p_redo_volt <= '0';
221           p_redo_curr <= '0';
222           case to_integer(readCounter) is
223             when 1 =>
224               if sequenceCounter < to_unsigned(12, sequenceCounter'length) then
225                 p_volt_high <= unsigned(rxData(i2cWordBitNb-1 downto 0));
226               else
227                 p_curr_high <= unsigned(rxData(i2cWordBitNb-1 downto 0));
228               end if;
229             when 2 =>
230               if sequenceCounter < to_unsigned(12, sequenceCounter'length) then
231                 p_volt_low <= unsigned(rxData(i2cWordBitNb-1 downto 0));
232               else
```




```
233               p_curr_low <= unsigned(rxData(i2cWordBitNb-1 downto 0));
234             end if;
235           when 3 =>
236             if sequenceCounter < to_unsigned(12, sequenceCounter'length) then
237               p_redo_volt <= rxData(i2cWordBitNb-1);
238             else
239               p_redo_curr <= rxData(i2cWordBitNb-1);
240             end if;
241           when others => null;
242         end case;
243       end if;
244     end if;
245   end process storeData;
246
247   battery250uv <= dataRegisterType(p_volt);
248   current250uA <= dataRegisterType(p_curr);
249
250 END ARCHITECTURE rtl;
```

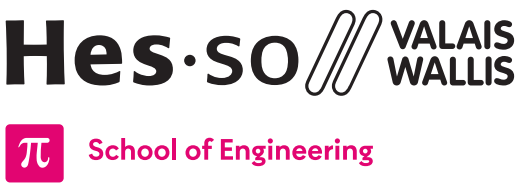
VI Tx Sequencer - VHDL

```
Library Kart;
  Use Kart.Kart.all;

ARCHITECTURE rtl OF txSequencer IS

signal p_refresh, p_send : std_ulogic;
signal p_data : std_uLogic_vector(UART_BIT_NB-1 DOWNTO 0);
signal p_cnter : unsigned(23 downto 0);
constant V_HEADER : string := "Voltage : ";
constant C_HEADER : string := "    Current : ";
constant CH0 : std_uLogic_vector(UART_BIT_NB-1 DOWNTO 0) :=
  std_ulogic_vector(to_unsigned(48, UART_BIT_NB));
constant CH1 : std_uLogic_vector(UART_BIT_NB-1 DOWNTO 0) :=
  std_ulogic_vector(to_unsigned(49, UART_BIT_NB));


type send_state is (idle, refreshs, sendVoltHead, sendVolt,
  sendCurrHead, sendCurr);
signal p_state : send_state;

BEGIN

  refresh <= p_refresh;
  send <= p_send;
  dataIn <= p_data;

  LED_G <= '0' when p_state = idle else '1';
  LED_R <= '1' when unsigned(current250uA) = 0 else '0';

  sendSM : process(resetSynch, clock)
  begin
    if resetSynch = '1' then
      p_refresh <= '0';
      p_send <= '0';
      p_data <= (others=>'0');
      p_state <= idle;
      p_cnter <= (others=>'0');
    elsif rising_edge(clock) then
      p_refresh <= '0';
      p_send <= '0';
      case p_state is
        when idle =>
          if p_cnter >= to_unsigned(10000000, p_cnter'length) then -- each second
            p_state <= refreshs;
            p_refresh <= '1';
            p_cnter <= (others=>'0');
          else
            p_cnter <= p_cnter + 1;
          end if;

        when refreshs => -- after sending refresh, wait for data to read
          if p_cnter >= to_unsigned(1000000, p_cnter'length) then -- arbitrary
  ↪  100ms
            p_state <= sendVoltHead;
            p_cnter <= (others=>'0');
          else
            p_cnter <= p_cnter + 1;
          end if;

```




```
59          when sendVoltHead =>
60            if p_cnter >= to_unsigned(V_HEADER'length, p_cnter'length) then
61              p_cnter <= (others=>'0');
62              p_state <= sendVolt;
63            elsif busy = '0' then
64              p_data <=
    ↪  std_ulogic_vector(to_unsigned(character'pos(V_HEADER(to_integer(p_cnter +
    ↪  1))),UART_BIT_NB));
65              p_send <= '1';
66              p_cnter <= p_cnter + 1;
67            end if;
68
69          when sendVolt =>
70            if p_cnter >= to_unsigned(battery250uv'length, p_cnter'length) then
71              p_cnter <= (others=>'0');
72              p_state <= sendCurrHead;
73            elsif busy = '0' then
74              p_data <= CH1 when
    ↪  battery250uv(to_integer(battery250uv'length-p_cnter-1)) = '1' else CH0;
75              p_send <= '1';
76              p_cnter <= p_cnter + 1;
77            end if;
78
79          when sendCurrHead =>
80            if p_cnter >= to_unsigned(C_HEADER'length, p_cnter'length) then
81              p_cnter <= (others=>'0');
82              p_state <= sendCurr;
83            elsif busy = '0' then
84              p_data <=
    ↪  std_ulogic_vector(to_unsigned(character'pos(C_HEADER(to_integer(p_cnter +
    ↪  1))),UART_BIT_NB));
85              p_send <= '1';
86              p_cnter <= p_cnter + 1;
87            end if;
88
89          when sendCurr =>
90            if p_cnter >= to_unsigned(current250uA'length, p_cnter'length) then
91              p_cnter <= (others=>'0');
92              p_state <= idle;
93            elsif busy = '0' then
94              p_data <= CH1 when
    ↪  current250uA(to_integer(current250uA'length-p_cnter-1)) = '1' else CH0;
95              p_send <= '1';
96              p_cnter <= p_cnter + 1;
97            end if;
98
99          when others => null;
100       end case;
101     end if;
102   end process sendSM;
103
104 END ARCHITECTURE rtl;
```

Libero SoC (Microsemi)



Author: Amand Axel, Silvan Zahno
Date: March 20, 2023
Version: v1.0

1 Overview

Libero SoC is a design software from Microchip (former Actel) for FPGAs from the following families:

- RTG4 Rad-Tolerant
- SmartFusion 2 and SmartFusion
- Igloo 2 and Igloo
- ProASIC 3
- Fusion

1.1 Licensing

A floating license is granted through a node server by connecting to HEI's VPN or locally by using the **secure-hevs** Wi-Fi or an ethernet cable.

The server includes 25 silver seats (users licenses) and 1 platinum seat (permanent license).

In order to correctly obtain said license, the server must be added as an environment variable:

Windows
- Right-click on **Computer** → **Properties** → **Advanced** → **Environment Variables**
- Select or add the system variable **LM_LICENSE_FILE**
- Append it **1702@actellm.hevs.ch**

Unix
- Open a shell and type the following:
  - `export LM_LICENSE_FILE=$LM_LICENSE_FILE:1702@actellm.hevs.ch`

1.2 Version

In order to get a compatible license, Libero's version must be the v11.8 SP3 which can be downloaded from Microchip's software archive. Both the base version and the service pack must be installed.

1.3 Running Libero

Synthesis

Libero SoC can be launched as a standalone or from one of its ***.prjx** project file to complete the synthesis process 2 Synthesis.

By using a correctly configured HDL Designer project one can launch Libero directly from it.

The dedicated HDL tasks are found under the

Prefs/hds_user/vXXX/tasks/libero_project_navigator.tsk folder. Libraries available in the **Libs** folder, pulled from the DiD-libs repository. Scripts are located in the **Scripts** folder, pulled from the DiD-scripts repository.

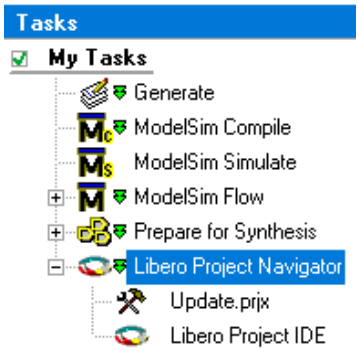


Figure 1: HDL Designer - Libero shortcut

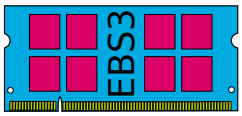
Figure 2: HDL Designer - Update .prjx (left) and Libero IDE (right) configurations

Flash

The FPGA's flash is done through the FlashPro software included with Libero through the generated ***.pdb** bitfile thanks to a dedicated programmer such as the FlashPro4.

It can be launched as a standalone or directly from within Libero 3 Flashing.

FlashPro can also be used to generate ***.svf** files which can then be used with OpenOCD to flash the FPGA by its USB port.



2 Synthesis

2.1 Create project

- Launch Libero SoC
- Click on **Project** → **New Project**
- **Proj. Details**: enter project's name, location, and set HDL type to **VHDL**
- **Dev. Selection**: select the correct chip. Example for the Kart board:
  - **Family** : Igloo
  - **Die** : AGLN250V5
  - **Package**: 100 VQFP
  - **Range** : COM
  - Gives the par number AGNL250V5-VQ100(1.5V)
- **Dev. Settings**: set I/O technology to **LVCMOS 3.3V**
- **HDL Sources**: link the vhdl file to synthesis (when created through HDL Designer, select the **Board/concat/projectName.vhd** (not the concatenated one))
- **Constraints**: link the constraints file (for school projects, the file is located under **Board-/concat/projectName.pdc**)
- Finish

2.2 Prepare project

The project window will then open. On the list located on the left, locate the **Compile** → **Constraints** → **yourConstraintsFile.pdc** → right-click → **Mark as used**:

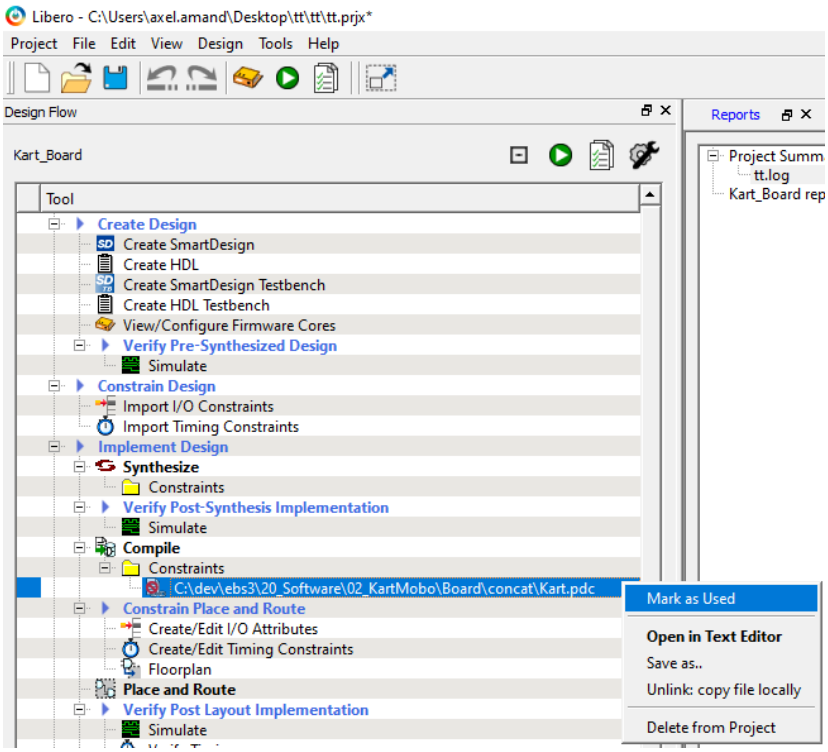


Figure 3: Use constraints

Right click on **Synthesize** → **Open Interactively**. In the newly opened window, on the left, set the correct clock frequency and exit while saving:

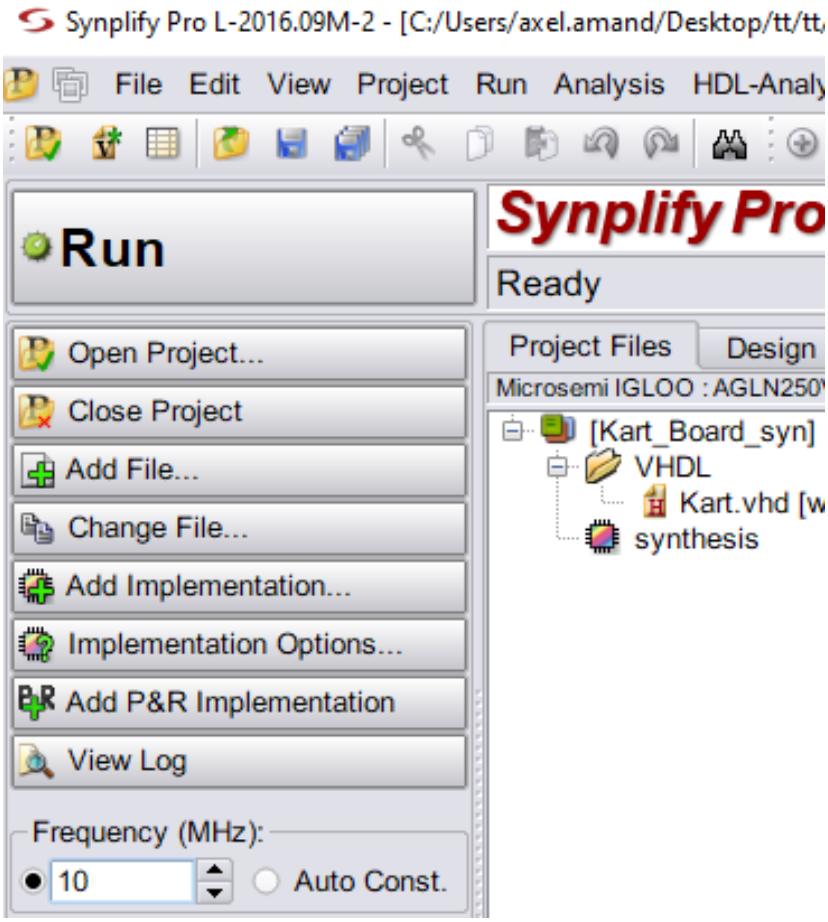


Figure 4: Setup clock

This step is required for the program to estimate if the implementation reaches the correct timings.

Right click on **Compile** → **Open Interactively** → **I/O attribute editor** and check that pins are correctly linked to the internal signals with correct settings:

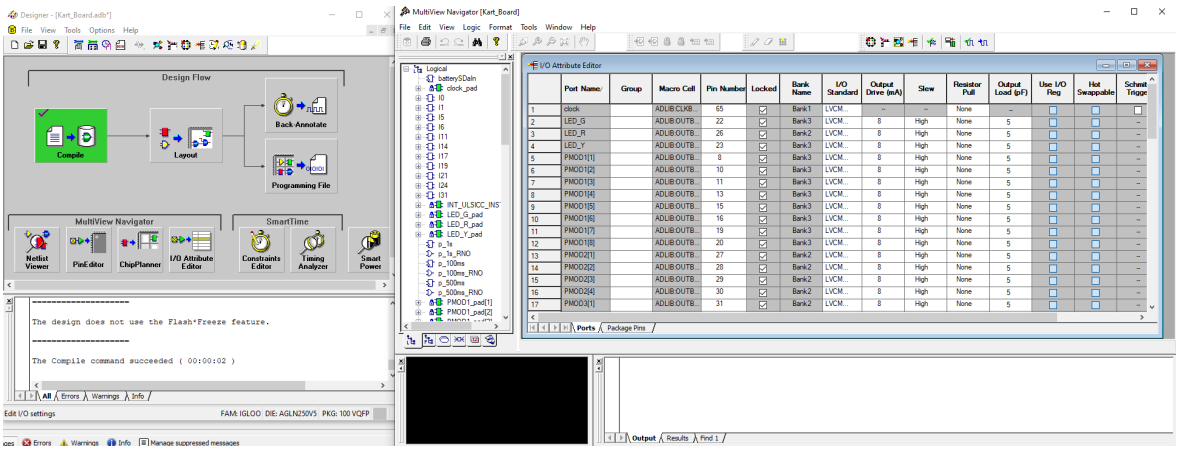


Figure 5: Constraints check

2.3 Synthesize

When the project is ready, the VHDL file must be synthesized, compiled and rooted on the chip.

Either click on **Layout** in the previously opened window (**Compile** → **Open Interactively** → **Layout** → **Ok**) or double click on **Place and Route** in the task list.

The **Message** window on the bottom of Libero can be used to check for errors and warning (both should always be checked). Some parts of the circuit may be pruned, clocks inferred unintentionally, unused signals found ...

In addition, reports can be browsed in the **Reports** tab, notably:

- **Synthesize - prjName.srr**

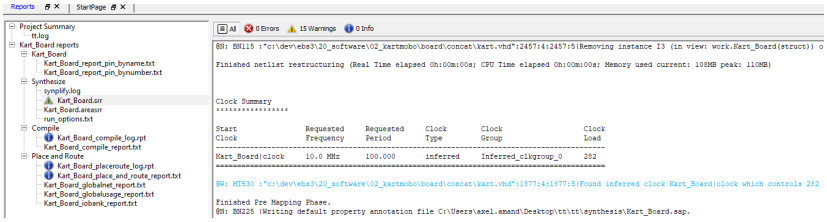


Figure 6: SRR report

  which contains informations on:

  - **Clock summary**: should only show the actual clocks - if some inferred ones are found, there is a design problem in the VHDL code
  - **Performance summary**: shows the worst slacks, if the timings are met and the potential fastest clock usable
  - **Core Cells** and **RAM/ROM** usages: how full the FPGA is

- **Compile - prjName_compile_log.rpt** shows the following:
  - **Compile Report**: more detailed view of used cells, BRAM block, I/Os ...
  - **I/O Technology**: ensure the standard is set to **LVCMOS33**
  - **I/O Placement**: ensure I/Os are all locked (Placed and UnPlaced ones may indicate errors or that the compilation used a pad to root a signal more easily because said pad was not locked even if not used → user must ensure the pad is not rooted to anything on the board or lock it beforehand)
- **Place and Route - prjName_globalnet_report.txt** shows global clocks and reset signals found under **Nets Sharing Loads**. In most cases, only a clock and a reset should be shown.

2.4 Bitfile

If the compilation is successfull, double click on **Generate Programming Data** to generate a ***.pdb** bitfile.

From there, refer to 3 Flashing. FlashPro can be launched directly with a right click on **Program Device** → **Open Interactively**.

2.5 Tips

Locked pins

All I/Os referenced in the VHDL file should be linked to a pad of the chip. If when trying to compile the following window appears, the ***.pdc** constraints file may be missing some I/Os:

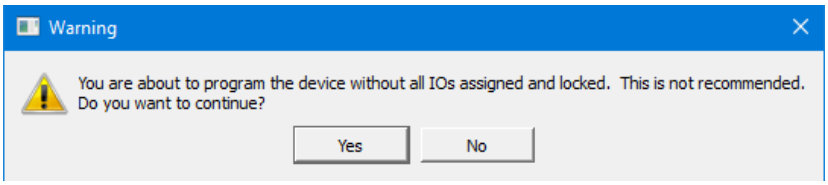


Figure 7: Incomplete PDC file warning

Click No and correct it.

Inferred clocks

Incorrect designs may lead the synthesizer to infer some clocks (e.g. may appear on state machines depending on a signal to trigger each state).

If the warning is ignored it may replace the clock with said signal. Incorrect behavior may result (desynchronized state changed, no state changes at all ...). Here is an example of inferred clock:

```
Clock Summary
*****************

Start                                Requested     Requested     Clock        Clock                Clock
Clock                                Frequency     Period        Type         Group                Load
-----------------------------------------------------------------------------------------------------------
Kart_Board|clock                     10.0 MHz      100.000       inferred     Inferred_clkgroup_0  1328
coilControl|stepdelayed_inferred_clock  10.0 MHz   100.000       inferred     Inferred_clkgroup_1  4
===========================================================================================================
```

Figure 8: Inferred extra clock

Pruned signals

The circuit will likely be optimized by Libero. Unused signals will be pruned and left out, possibly leading it to drop complete functionalities because the signal does not exist anymore. To keep internal signals (e.g. a pad → linked to an internal signal → wired to an empty block; in such case, the signal block is pruned, then the signal too, then Libero will throw an error reading the PDC and trying to link the pad to a now unexisting signal).

To avoid this, attributes can be put on signals which will then tweak the synthesizer.

In pure VHDL, the attributes are put after the **ENTITY** declaration:

```
1 attribute syn_keep : boolean;
2 attribute syn_keep of
3     clock,
4     reset_n,
5     mySig1,
6     mySig2
7 : signal is true;
```

In HDL Designer, the same can be achieved by opening the **symbol** of a block → **Declarations** and writing attributes there.

3 Flashing

3.1 FlashPro

Overview

FlashPro is the official tool supported for the FPGAs listed on 1 Overview and comes with Libero SoC (here it is the v11.8.3.6). *It is required to create compatible programming files for OpenOCD.*

The Express version is available as a standalone.

Usage

Launch FlashPro directly or from within the Libero project with a right click on **Program Device** → **Open Interactively**.

Create a **New Project**, name it, select **Single device** then Ok. Wire a compatible programmer (such as a FlashPro4) and click on **Refresh/Rescan for Programmers** which should show found devices:

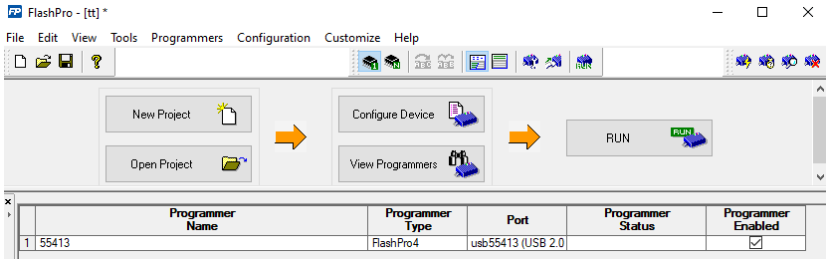


Figure 9: FlashPro found programmers

Under **Configuration** → **Load Programming File** select the previsouly created ***.pdb** file.

Wire the board, connect the programmer, then click on **PROGRAM**. The advancement is shown in column **Programmer Status**.

The programmer does not supply the daughterboard with power. An external power source (motherboard or USB-C) is needed.

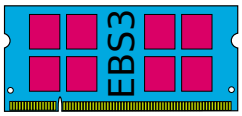
I/Os states normally remain in high-impedance state (with potential pull resistors) while the chip is being programmed. For sensitive applications, disconnect it from the motherboard beforehand.

3.2 OpenOCD

The board can be programmed without any third-parties hardware through OpenOCD thanks to the embedded FT2232HL chip on the daughterboard which offers both an UART and a JTAG interface.

Windows

- Download OpenOCD unofficial binary releases or build it yourself.



- Add the **.exe** to the path:
  - Right-click on **Computer** → **Properties** → **Advanced** → **Environment Variables**
  - Select the system variable **Path**
  - Append it **openOcdLocation/bin**
  - Open a command prompt and type **openocd -v** which should output OpenOCD's version
- Download the scripts to support the custom EBS3 chips
- Copy the **board**, **interface** and **target** folders into the OpenOCD scripts folder (either under **share/openocd/scripts/** or **scripts/**)
- In the **support** folder, run **Zadig** as administrator → select the **USB <-> Serial Converter (Interface 0)** and press **Replace driver**

Without Zadig libusb drivers, the board cannot be found by OpenOCD.

Linux

- Download OpenOCD xpack to **/usr/opt/OpenOCD**:

```
DOWNLOAD_URL='https://github.com/xpack-dev-tools/openocd-xpack/releases/download'
VERSION='0.11.0-5'
curl -L $DOWNLOAD_URL/v$VERSION/xpack-openocd-$VERSION-linux-x64.tar.gz -o /tmp/openocd.tar.gz
mkdir /tmp/OpenOCD
tar -xvzpf /tmp/openocd.tar.gz -C /tmp/OpenOCD/
sudo mv /tmp/OpenOCD/xpack-openocd-* /usr/opt/OpenOCD
/usr/opt/OpenOCD/bin/openocd -v
```

- Add a USB rule for the FTDI chip:

```
lsusb | grep FT2232C
cat /usr/opt/OpenOCD/contrib/60-openocd.rules | grep -A 1 FT2232
sudo cp /usr/opt/OpenOCD/contrib/60-openocd.rules /etc/udev/rules.d/
sudo reboot
```

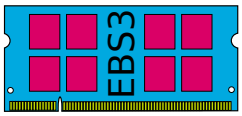
- Download the scripts for the EBS3 boards then:

```
unzip ~/Downloads/ebs3-oocd-scripts.zip -d /tmp
sudo cp -pv /tmp/ebs3-oocd-scripts/openocd/board/*.cfg /usr/opt/OpenOCD/scripts/board/
sudo cp -pv /tmp/ebs3-oocd-scripts/openocd/interface/*.cfg /usr/opt/OpenOCD/scripts/interface/
sudo cp -pv /tmp/ebs3-oocd-scripts/openocd/target/*.cfg /usr/opt/OpenOCD/scripts/target/
```

OpenOCD scripts

If scripts cannot be downloaded, create the following files under openocd **share/openocd/scripts/** or **scripts/** subfolder (reference only, may not be up to date):

- **board/kart.cfg**: refer to appendix I OpenOCD board/kart.cfg file *(may be omitted since*



the board contains no extra JTAG chip)

- **interface/ebs3.cfg**: refer to appendix II OpenOCD interface/ebs3.cfg file (contains the FTDI layout)
- **target/igloo_agln125.cfg**: refer to appendix III OpenOCD target/igloo_agln125.cfg file (if using an AGLN125 chip)
- **target/igloo_agln250.cfg**: refer to appendix IV OpenOCD target/igloo_agln250.cfg file (if using an AGLN250 chip)
- **target/igloo_agl600.cfg**: refer to appendix V OpenOCD target/igloo_agl600.cfg file (if using an AGL600 chip)

3.3 Programming files

With FlashPro project opened, click on **File** → **Export** → **Export Single Device SVF File** and select the desired output folder. Multiple files are generated, from which ***_ARRAY_*** files are not used.

3.4 Flash

Windows

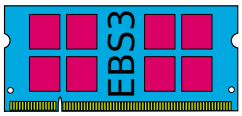
- Open a terminal and **cd** to the ***.svf** files location.
- Type and launch the following command, which should log the results shown below:
  openocd -f target/igloo_agln250.cfg -c init -c "svf -quiet projname_ERASE.svf" -c "svf -quiet projname_PROGRAM.svf" -c "svf -quiet projname_VERIFY.svf" -c shutdown
  - Info : clock speed 4000 kHz
  - Info : JTAG tap: agl250.tap tap/device found: 0x1ba541cf (mfg: 0x0e7 (GateField), part: 0xba54, ver: 0x1)
- While the board is being programmed, only the **erase**, **program** and **verify** steps are logged (you can remove the **-quiet** switch to see which command is actually processed, but with more than a million for the program part, it only slows down the process)
- Once the three steps are completed (around 4 minutes for a 75% filled FPGA), the design is ready to be tested

Linux

Modify and run the following (see the Windows chapter for explanations):

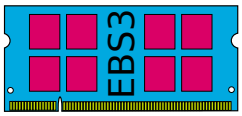
```
OPEN_OCD_DIR='/usr/opt/OpenOCD/bin'
OPEN_OCD_FILES_DIR='/home/path/to/oocd/svffiles'
OPEN_OCD_FILE='projname'
$OPEN_OCD_DIR/openocd                                                \
-f target/igloo_agl250.cfg                                           \
-c init                                                              \
-c "svf -quiet $OPEN_OCD_FILES_DIR/${OPEN_OCD_FILE}_ERASE.svf"   \
-c "svf -quiet $OPEN_OCD_FILES_DIR/${OPEN_OCD_FILE}_PROGRAM.svf" \
-c "svf -quiet $OPEN_OCD_FILES_DIR/${OPEN_OCD_FILE}_VERIFY.svf"  \
-c shutdown
```

I OpenOCD board/kart.cfg file

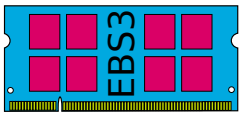



```
#
# OpenOCD board configuration file for the Kart motherboard
# Axam
#

echo "\nLoading Kart board configuration"

# Target FPGA must be included by another arg. call since any SODIMM200 board can
↪  be used

# No special chip, nothing else to do

echo "*** Kart board loading done\n"
```

II OpenOCD interface/ebs3.cfg file

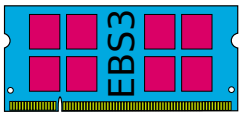



```
#
# EBS3 JTAG interface
# Based on FT2232HL chip
# JTAG wired on channel 0
#
# https://gitlab.hevs.ch/course/ElN/fpga-ebs3
#

if { [info exists SAMPLE_ON_FALLING] } {
  set _SAMPLE_ON_FALLING $SAMPLE_ON_FALLING
} else {
  set _SAMPLE_ON_FALLING 0
}

if { [info exists nSRST_WIRED] } {
  set _nSRST_WIRED $nSRST_WIRED
} else {
  set _nSRST_WIRED 0
}

echo "\nLoading EBS3 JTAG interface"
echo " * Based on FT2232HL, channel 0"

# Adapter configuration
adapter driver ftdi

# Which programmer chip we are looking for and its configuration
ftdi_vid_pid 0x0403 0x6010 ; # vid-pid as stated here :
↪  https://www.ftdichip.com/Support/Documents/TechnicalNotes/TN_100_USB_VID-
↪  PID_Guidelines.pdf
ftdi_channel 0
# May help with sampling problems
if { $_SAMPLE_ON_FALLING } {
        echo " * Sampling on TCK falling edge is activated"
        ftdi_tdo_sample_edge falling ;
} else {
        echo " * Sampling normally on TCK rising edge"
}
# Setup FTDI layout
        # First is data initial value (for selected channel)
                # 0x0038 = 0b 11 1000 => TMS, nTRST and nSRST as 1
        # Second is pin direction (!!! 1 is output, 0 is input)
                # 0x003B = 0b 11 1011 => TDO as input, others as outputs
if { $_nSRST_WIRED } {
        echo " * nSRST is set to wired"
        ftdi_layout_init 0x0038 0x003B
        ftdi_layout_signal nSRST -data 0x0020 -oe 0x0020
        ftdi_layout_signal SRST -nalias nSRST
} else {
        echo " * nSRST is not used"
        ftdi_layout_init 0x0018 0x001B
}
        # Define specific signals
                # -ndata : which pin, n is for inverted signal (active low)
                # -noe : where the output is, i.e. through buffer (if equals data
↪  -> no buffer)
        # NOTE : TRST is JTAG TAP reset, while SRST is the overall system reset
↪  (linked to nPoR signal)
ftdi_layout_signal TCK -data 0x0001
```



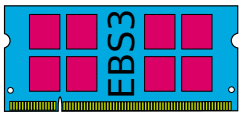

```
ftdi_layout_signal TDO -data 0x0002
ftdi_layout_signal TDI -input 0x0004
ftdi_layout_signal TMS -data 0x0008
ftdi_layout_signal nTRST -data 0x0010 -oe 0x0010
ftdi_layout_signal TRST -nalias nTRST

# JTAG protocol is used
transport select jtag
        # 4 MHz speed, may be lowered in case of errors
adapter speed 4000

        # Transport configuration
if { $_nSRST_WIRED } {
        reset_config trst_and_srst separate trst_push_pull srst_push_pull
        # Minimal time before deasserting nSRST, in ms
        adapter srst pulse_width 400
        # Delay between reset release and new JTAG operation, in ms
        adapter srst delay 50
} else {
        reset_config trst_only separate trst_push_pull
}
 # Minimal time before deasserting nTRST, in ms
jtag_ntrst_assert_width 550
jtag_ntrst_delay 350


echo "*** EBS3 loading done\n"
```

III OpenOCD target/igloo_agln125.cfg file



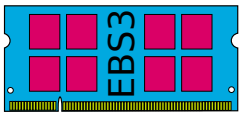

```
#
# OpenOCD configuration file for the FTDI FT2232HL
# Used by the FPGA-EBS3 based projects (SODIMM200 FPGA)
# Axam
#

if { [info exists CHIPNAME] } {
  set _CHIPNAME $CHIPNAME
} else {
  set _CHIPNAME agl125
}

source [find interface/ebs3.cfg]

echo "\nLoading Igloo AGL125 target configuration"

# irlen given by bsd file (instruction register length)
# ignore version ignore bits 28-31 (the X ones)
# ID given by bsd file :
# Version     Part Number             Manuf. ID        LSB
# XXXX        X011 1010 0101 0010     000 1110 0111    1
jtag newtap $_CHIPNAME tap -irlen 8 -ignore-version -expected-id 0x03A521CF
 ↪  -expected-id 0x0BA521CF

echo "*** Igloo AGL125 target loading done\n"
```

IV OpenOCD target/igloo_agln250.cfg file




```
#
# OpenOCD configuration file for the FTDI FT2232HL
# Used by the FPGA-EBS3 based projects (SODIMM200 FPGA)
# Axam
#

if { [info exists CHIPNAME] } {
  set _CHIPNAME $CHIPNAME
} else {
  set _CHIPNAME agl250
}

source [find interface/ebs3.cfg]

echo "\nLoading Igloo AGL250 target configuration"

# irlen given by bsd file (instruction register length)
# ignore version ignore bits 28-31 (the X ones)
# ID given by bsd file :
# Version     Part Number             Manuf. ID        LSB
# XXXX        X01X 1010 0X01 0100     000 1110 0111    1
jtag newtap $_CHIPNAME tap -irlen 8 -ignore-version -expected-id 0x03A141CF
 ↪  -expected-id 0x03A541CF -expected-id 0x0BA141CF -expected-id 0x0BA541CF

echo "*** Igloo AGL250 target loading done\n"
```

V OpenOCD target/igloo_agl600.cfg file




```
#
# OpenOCD configuration file for the FTDI FT2232HL
# Used by the FPGA-EBS3 based projects (SODIMM200 FPGA)
# Axam
#

if { [info exists CHIPNAME] } {
  set _CHIPNAME $CHIPNAME
} else {
  set _CHIPNAME agl600
}

source [find interface/ebs3.cfg]

echo "\nLoading Igloo AGL600 target configuration"

# irlen given by bsd file (instruction register length)
# ignore version ignore bits 28-31 (the X ones)
# ID given by bsd file :
# Version     Part Number             Manuf. ID        LSB
# XXXX        X01X 1010 0X01 0100     000 1110 0111    1
# XXXX        X01X 1011 0010 0110     000 1110 0111    1
jtag newtap $_CHIPNAME tap -irlen 8 -ignore-version -expected-id 0x02B261CF
 ↪  -expected-id 0x03B261CF -expected-id 0x0AB261CF -expected-id 0x0BB261CF

echo "*** Igloo AGL600 target loading done\n"
```