A1222/TABOR MOTHERBOARD TECHNICAL REFERENCE MANUAL VERSION 1.0.3 AMIGAONE 1222 TOPAZ

Issue	Author	Approved	Date
1.00	James Felix	Marcin Jankowski	24/03/2015
1.01	R.T.Dickinson	R.T.Dickinson	01/04/2015
1.02	R.T.Dickinson/J. Krueger	Changes under review	07/01/2019
1.03	R.T.Dickinson	Changes under review	18/02/2020

1 CONTENTS

FIGURES

TABLES

2 INTRODUCTION

The Tabor motherboard combines a high performance Freescale QorIQ CPU with powerful and flexible I/O features to deliver a desktop platform for AmigaOS users.

This manual contains hardware and software reference information to assist with installation, configuration and low level programming of Tabor.

2.1 TECHNICAL SUPPORT

For technical support, please contact your reseller.

2.2 ABBREVIATIONS

Acronym	Description
PCIe	PCI Express
PSU	Power Supply Unit
CPU	Central Processing Unit
Hot-Plug	Remove or insert connection/cable whilst power is on
RTC	Real time clock
OD	Open Drain
PU	Pulled-Up
PD	Pulled-Down
RO	Read only
RW	Read write
BCD	Binary-coded decimal
ACPI	Advanced Configuration and Power Interface
SD	Secure Digital
ASCII	American Standard Code for Information Interchange
MS	Most significant
LS	Least significant
MSB	Most significant bit
LSB	Least significant bit
DIU	Display Interface Unit
GM bus	GPIO CPLD Memory Bus
LB	Local Bus
DDC	Display Data Channel

A '#' suffix denotes an active-low signal when used throughout the document.

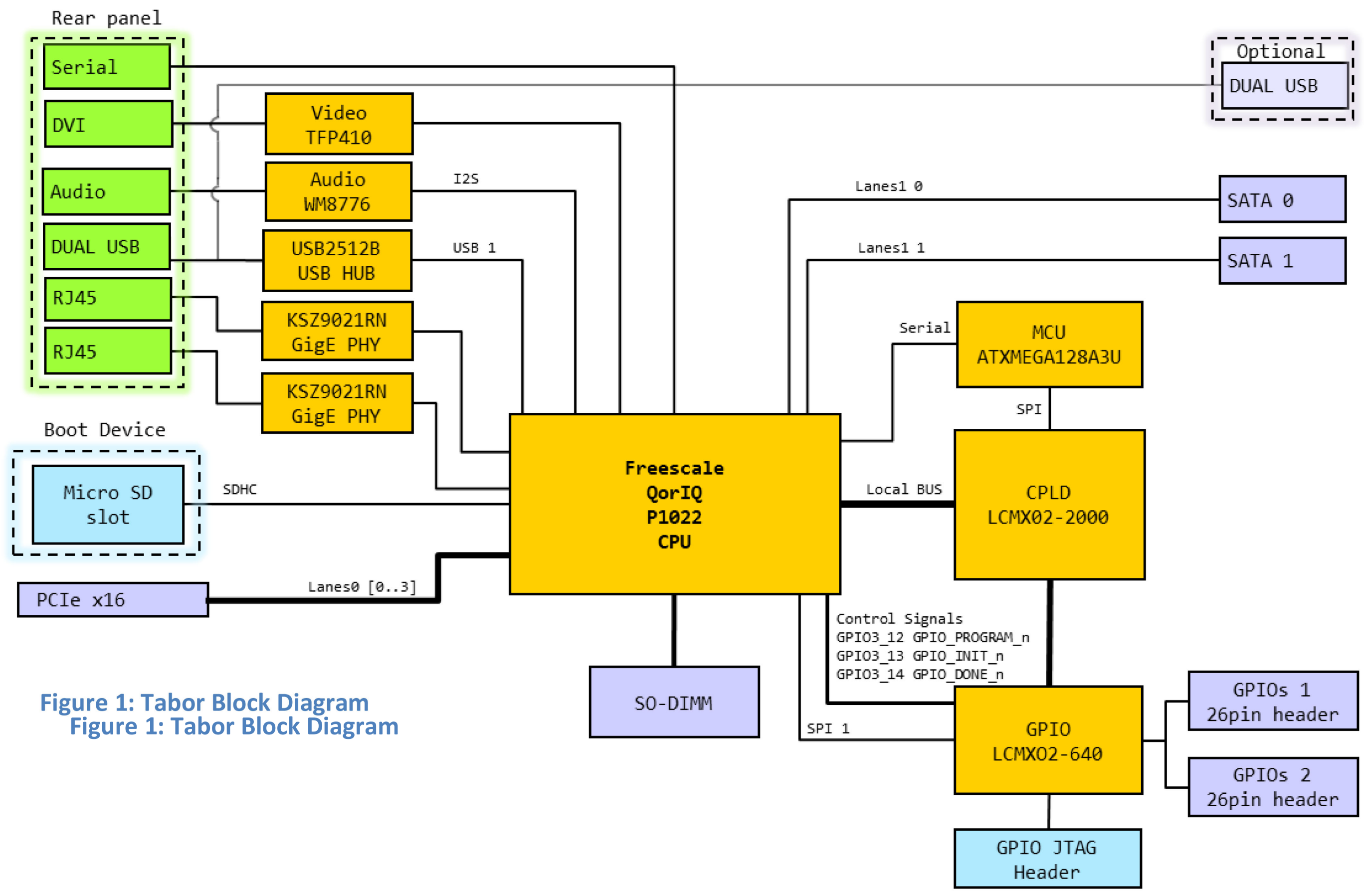


Figure 1: Tabor Block Diagram

Figure 1: Tabor Block Diagram

3 ARCHITECTURE: TABOR'S ARCHITECTURE IS SHOWN IN FIGURE 1 ABOVE:

3.1 CPU

The CPU on Tabor is a Freescale QorIQ Power Architecture P1022 series processor.

3.1.1 P1022

This CPU combines two 1.2 GHz 32-bit e500v2 cores with a 256KB L2 cache, a single DDR3 memory controller (800MT/s) and 6 SerDes channels.

The Power Architecture e500v2 cores adhere to Power ISA v.2.03, for more information on the e500v2 check the Freescale website.

3.2 MAIN MEMORY

The P1022 has one memory controller which is connected to a standard DDR3 SO-DIMM slot.

For further details, see section 5.

3.3 Ethernet Phy

The two Micrel KSZ9021RN Gigabit Ethernet PHYs use the RGMII protocol.

The PHYs have two LEDs to indicate the link status as shown in Table 1. When there is activity on the port whichever LED is/are on will blink.

Speed	LED1 (right)	LED2 (left)
1000 link	Off	On
100 link	On	Off
10 link	On	On

Table 1: Ethernet link speed

3.4 AUDIO

Tabor's audio is provided by a Wolfson WM8776 I^{2}S codec. This is connected to J6 and provides Line-in (blue), Line-out (green) and microphone in (pink) connections.

3.5 DVI

The CPU has an on chip DIU; the output of this is buffered by a Texas Instruments TFP410 driver. This provides a DVI-D output on J7.

For further details on the DIU see the P1022 Reference Manual.

3.6 MAIN CPLD

The Main CPLD provides glue logic and control registers. It also provides a fast mailbox and data interface between the CPU and the GPIO CPLD.

The logic in this CPLD is fixed.

For further details on the Main CPLD, see section 6.16 and for the GPIO CPLD see section7.57.

3.7 GPIO CPLD

A Lattice LCMXO2-640 CPLD device is provided to support general purpose inputs and outputs. This has a high speed interface to the Main CPLD, and a separate SPI interface to the CPU.

The GPIO CPLD is reconfigurable and can contain custom user logic. It can be programmed either by the CPU or using the JTAG header (H3).

For further details on the GPIO CPLD see section6.17.

3.8 BOOT SD CARD

The Tabor motherboard is booted from a micro SD card fitted in P8. A valid BIOS is required in the first 10Mb, for more information see section 9.

4 CPU

This section provides programmer visible details of CPU hardware implementation.

4.1 SERDES LANES

The SerDes1 lanes are connected as shown in Table 2 below:

Lane	Connection
0	PCIe slot 1 lane 0
1	PCIe slot 1 lane 1
2	PCIe slot 1 lane 2
3	PCIe slot 1 lane 3

Table 2: CPU SerDes1 Lane Assignments

The SerDes2 lanes are connected as shown in Table 3Table 2 below:

Lane	Connection
0	SATA1
1	SATA2

Table 3: CPU SerDes2 Lane Assignments

4.2 UARTs

The CPU provides two UARTs, one for external RS232 communication and one for MCU supervisor interface.

The UART 0 signals are available on a DB9 connector, J9.

UART 1 is connected to the MCU; this provides access to temperature and voltage readings. For further details on the MCU's supervisor interface see section 8.1.

4.3 GPIOS

The CPU provides 87 general purposes I/Os (GPIOs) on 3 banks (1-3). 13 of these are used. For details on how these are wired, see Table 4.

GPIO line	Function	Direction	Signal Name	Notes
GPIO2_20	DVI SDA	bidirectional	SDA_CPU_DVI	PU,OD
GPIO2_21	DVI SCL	bidirectional	SCL_CPU_DVI	PU,OD
GPIO2_31	VGA_BIOS_ENABLE#	In	GPIO2_31	0 = jumper fitted, PU
GPIO3_9	Jumper 1	In	GPIO3_9	0 = jumper fitted, PU
GPIO3_10	CPU LED	Out	CPU_LED	1 = LED on, PU
GPIO3_11	HDD ACTIVITY LED	Out	HDD_LED	1 = LED on, PU
GPIO3_12	GPIO_PROGRAM#	Out	GPIO_PROGRAM_n	PU
GPIO3_13	GPIO_INIT#	Out	GPIO_INIT_n	PU
GPIO3_14	GPIO_DONE#	In	GPIO_DONE_n	PU
GPIO3_15	HARD_RESET#	Out	CPU_MCU_GPIO0	OD, PU
GPIO3_16	POWER_OFF#	Out	CPU_MCU_GPIO1	OD, PU
GPIO3_17	-	In	CPU_MCU_GPIO2	PU

Table 4: CPU usable GPIOs

4.4 External Interrupts

P1022 External Interrupt	Connection	Setup
IRQ0#	Unused	interrupt input, level sensitive, active low
IRQ1#	CPLD	interrupt input, level sensitive, active low
IRQ2#	Ethernet PHY #1	interrupt input, level sensitive, active low
IRQ3#	Ethernet PHY #2	interrupt input, level sensitive, active low
IRQ4#	GPIO	interrupt input, level sensitive, active low
IRQ5#	MCU	interrupt input, level sensitive, active low

Table 5: CPU External interrupts

4.5 I^{2}C Controllers

The CPU has a hardware drive I^{2}C controller that is used for communicating with the Audio Codec, DVI controller, DDR3 SO-DIMM, RTC and MAC address EEPROMs. The addresses for these devices on the I^{2}C bus are given in

Error: Reference source not found.

Device	Address
Audio codec	0x1A
DVI driver	0x38
MAC address 2	0x50
DDR3 SPD	0x51
System Configuration Data EEPROM	0x53
MAC address 1	0x57
Unique ID	0x5B
RTC SRAM	0x6F
PCIe Clock IC	0x6E

Table 6: CPU Hardware I^{2}C Device Addresses

There is also a software driven I^{2}C controller for the DDC of the DVI monitor.

4.6 SERIAL TERMINAL

For serial communications, on a PC it is recommended to use TeraTerm. The serial port control must be configured as follows:

- 115200 Baud
- 8 bit data
- No Parity
- 1 Stop bit
- No Flow Control

5 DDR3 SO-DIMM

Tabor uses a standard 1.5V DDR3 SO-DIMM.

The board has been qualified with unbuffered non-ECC SO-DIMMs. For the latest information on recommended SO-DIMM module types, please contact your reseller.

5.1 SIZE

The total physical maximum size of memory that the memory controllers can address is 64GB, however the practical memory size limit will depend on software and SO-DIMM support.

5.2 SPEED

The maximum speed supported by the memory controllers is DDR3-800. Faster memory may be fitted but this speed limit will apply.

5.3 SERIAL PRESENCE DETECT

The Serial Presence Detect (SPD) address of the SO-DIMM socket is 0x51, hex as shown in section .

5.4 TESTED SO-DIMMS

Tabor has been tested with different SO-DIMMs as shown in Table 7.

Manufacture	Part Number	Speed	Size
Kingston	KVR800D3S8S6/2G	800MHz	2 GB
Kingston	KVR13S9S6/2	1.333GHz	2 GB

Table 7: Tested SO-DIMMs

6 MAIN CPLD

The Main CPLD is connected to the CPU via the local bus and allows for a high speed interface between the CPU and the GPIO CPLD. There are other registers available which include the CPU fan speed.

6.1 CPU COMMS

The interface for the CPU to the Main CPLD is an 8-bit multiplexed address/data bus. The registers and RAM within the Main CPLD are accessed indirectly. There is a 16 bit index register that is used to store the address of the register or the RAM to be accessed, and in the case of the RAM this index is auto-incremented during data transfers to allow efficient block transfers.

The distinction between data and index space is made using bit 7 of the address, and the distinction between upper and lower byte of the index register is made with bit 0 of the address. The Main CPLD Local bus address space is given in Table 8 below.

LB Address (hex)	Description
0x00	MS Byte index register addressed
0x01	LS Byte index register addressed
0x80	data space addressed

Table 8: Main CPLD Local bus address space

For register accesses, it is only necessary to program the MS Byte index register with bit 7 = 0 to indicate a register address and the lower bits to select which register, this is because the LS Byte index register is *'don't care'* for register accesses. For the RAM it is necessary to program the MS Byte index register with bit 7 = 1 to indicate a RAM address with the MS byte start address bits, and also program the LS Byte register with the low order start address bits as both MS Byte and LS Byte index registers are used to address the 8K dual port RAM.

Example reading SIG2 register with U-Boot commands:

Command	Description
`mw.b e0000000 1`	programs MS Byte order index with SIG2 address
`md.b e0000080 1`	reads SIG2 register from data space

Example writing 2 bytes of data to dual port RAM at start address 0x0240 with U-Boot commands:

Command	Description
`mw.b e0000000 82`	programs MS Byte index with MS byte RAM address, and bit 7 = 1
`mw.b e0000001 40`	programs LS Byte index with LS byte RAM address
`mw.b e0000080 BE`	writes 0xBE data to address 0x0240 in the RAM, index auto-increments
`mw.b e0000080 EF`	writes 0xEF data to address 0x0241 in the RAM, index auto-increments

```
There is a mailbox interface to the GPIO CPLD, for more details see section 7.5.
```

6.1.1 MAIN CPLD LOCAL BUS REGISTER MEMORY MAP

The register memory map of the Main CPLD LB as seen by the CPU is shown below in Table 9.

Address (hex)	Register name	Description	Read/ Write
0x00xx	SIG1	signature value 1 (0xDE)	RO
0x01xx	SIG2	signature value 2 (0xAD)	RO
0x02xx	HWREV	Hardware revision	RO
0x03xx	CPLDREV	CPLD version	RO
0x04xx	MBC2X	CPU to GPIO CPLD mailbox	RW
0x05xx	MBX2C	GPIO CPLD to CPU mailbox	RW
0x06xx	INT_STATUS	Mail box interrupt status (INT_MBC2G, INT_MBG2C)[1:0]	RO
0x07xx	MBC2G_RO	CPU to GPIO CPLD mailbox, read only	RO
0x08xx	MBG2C_RO	GPIO CPLD to CPU mailbox, read only	RO
0x09xx	GPIO_OUT	GPIO CPLD loopback out pin (LSB)	RW
0x0Axx	GPIO_IN	GPIO CPLD loopback in pin (LSB)	RO
0x10xx	FAN_TACHO_UB[1]	CPU Fan speed in Revolution per seconds, upper byte	RO
0x11xx	FAN_TACHO_LB[1]	CPU Fan speed in Revolution per seconds, lower byte	RO
0x12xx	FAN1_TACHO_UB[1]	CASE Fan1 speed in Revolution per seconds, upper byte	RO
0x13xx	FAN1_TACHO_LB[1]	CASE Fan1 speed in Revolution per seconds, lower byte	RO
0x14xx	FAN2_TACHO_UB[1]	CASE Fan2 speed in Revolution per seconds, upper byte	RO
0x15xx	FAN2_TACHO_LB[1]	CASE Fan2 speed in Revolution per seconds, lower byte	RO
0x18xx	ETH2STATE	Number of Ethernet PHYs fitted, [0x1=2 0x0=1]	RO
0x19xx	-	Reserved for internal use	-
0x20xx	DATE_UUB	CPLD build date bits[31:24], see Error: Reference source not found for build format	RO
0x21xx	DATE_ULB	CPLD build date bits[23:16], see Error: Reference source not found for build format	RO
0x22xx	DATE_LUB	CPLD build date bits[15:8], see Error: Reference source not found for build format	RO
0x23xx	DATE_LLB	CPLD build date bits[7:0], see Error: Reference source not found for build format	RO
0x24xx	TIME_UUB	CPLD build time bits[31:24], see Error: Reference source not found for time format	RO
0x25xx	TIME_ULB	CPLD build time bits[23:16], see Error: Reference source not found for time format	RO
0x26xx	TIME_LUB	CPLD build time bits[15:8], see Error: Reference source not found for time format	RO
0x27xx	TIME_LLB	CPLD build time bits[7:0], see Error: Reference source not found for time format	RO
0x5Cxx	SCR1	Scratch register	RW
0x6Axx	SCR2	Scratch register	RW
0x8000-0xFFFF	RAM	Dual port RAM, 8 bits wide, 8kbytes	RW

Table 9: Main CPLD LB register memory map

Notes:

1. **The FAN_TACHO registers should be read until two consecutive readings are the same, this eliminates asynchronous sampling errors.**
2. **Main CPLD Build Format**

6.1.2 MAIN CPLD BUILD FORMAT

The format of the Main CPLD build time and date are stored in a 32-bit value using BCD, one for build date and one for build time. The date is store as YYYYMMDD and the time is store as 00HHMMSS.

Address (hex)	Register name	Register value
0x20xx	DATE_UUB	Build year upper byte
0x21xx	DATE_ULB	Build year lower byte
0x22xx	DATE_LUB	Build month
0x23xx	DATE_LLB	Build day
0x24xx	TIME_UUB	00
0x25xx	TIME_ULB	Build time hours
0x26xx	TIME_LUB	Build time days
0x27xx	TIME_LLB	Build time minutes

Table 10: Main CPLD Build information registers

7 GPIO CPLD

Tabor has a user programmable GPIO CPLD connected to the Main CPLD and the CPU.

There are two GPIO headers (P16, P17) both with 17 GPIOs available on each header. These headers have a similar pinout to the Raspberry Pi GPIO header. The GPIO headers have both 5V and 3.3V power rails for powering external devices. There is a 0.5A current limit on both power rails, if more current is required use the ATX supply for power. The GPIO CPLD IO pins can be accessed either by the Main CPLD by becoming a master on the high speed Main CPLD to GPIO CPLD bus, or directly by the CPU via SPI. This section provides essential details of Tabor's GPIO CPLD and should be read in conjunction with relevant Lattice documentation.

7.1 BLOCK DIAGRAM

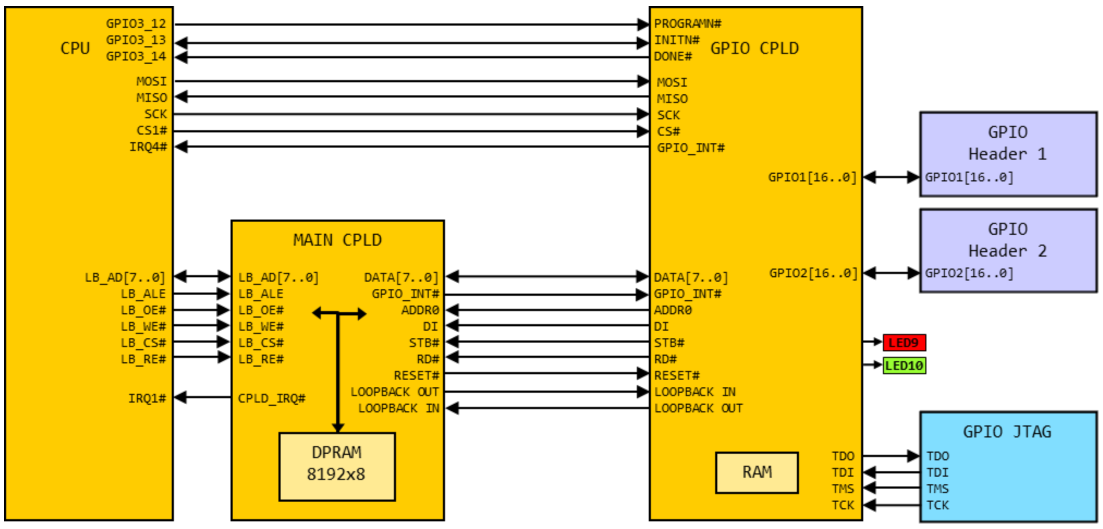


Figure 2: GPIO CPLD Block Diagram

7.2 HEADERS AND LEDS

shows how the GPIO CPLD is connected to the CPU, Main CPLD, GPIO headers and the JTAG header.

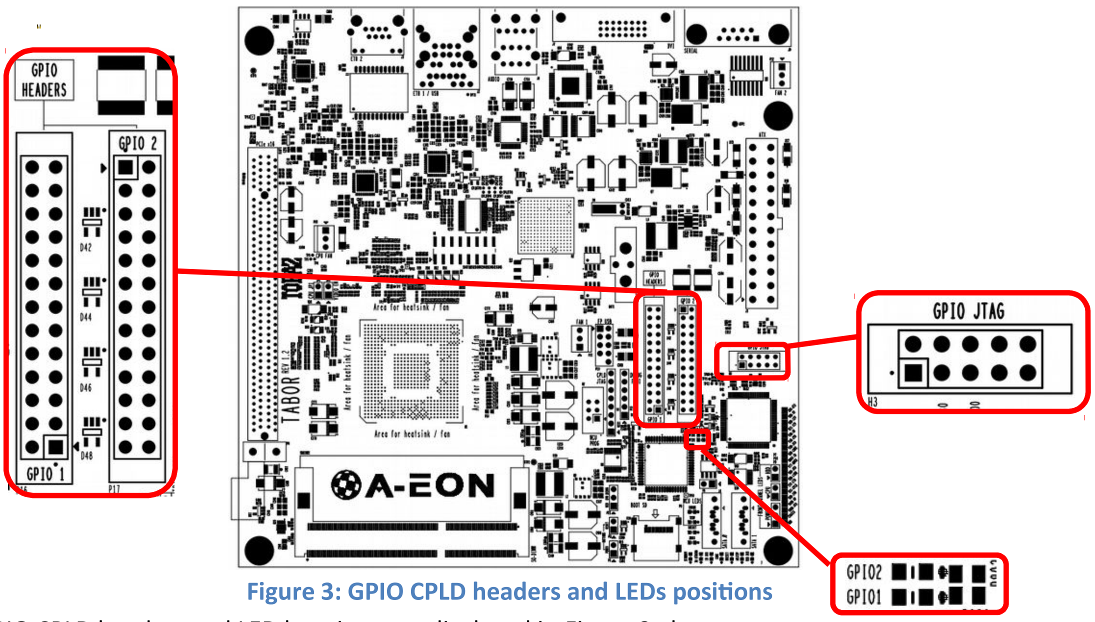


Figure 3: GPIO CPLD headers and LEDs positions

The GPIO CPLD headers and LED locations are displayed in Figure 3 above.

7.3 DEVICE TYPE

The GPIO CPLD is a Lattice LCMX02-640 CPLD in a 100-pin QFN package. It can be programmed in two different ways, either via a JTAG header (H3) for programming from an external source, or using the CPU's SPI bus along with some CPU GPIOs.

7.4 PROGRAMMING

To program the GPIO CPLD, you will first need to write firmware (usually in VHDL or Verilog) with which it can be programmed. This firmware can be simulated and compiled to a JEDEC programming file with Lattice's free Diamond software. The example code is written in VHDL and compiled using Diamond 3.1.

The JTAG header for the GPIO CPLD programming is H3 and the pinout is shown in Table 11.

Pin	Signal	Signal	Pin
1	TCK	GROUND	2
3	TDO	VGPIO-3V3V	4
5	TMS	-	6
7	-	-	8
9	TDI	GROUND	10

Table 11: GPIO CPLD JTAG header pinout

The other way of programming the GPIO CPLD is via SPI, from the CPU. Three CPU GPIOs will need to be toggled correctly to perform this process. The CPU GPIOs used with programming the GPIO CPLD are listed in Table 12.

Signal	Description	I/O in GPIO CPLD	GPIO CPLD pin
GPIO3_12	Pulse low to enable program mode.	Input	PROGRAMN#
GPIO3_13	Start configuration cycle, pulse low and error detection	bidirectional	INITN#
GPIO3_14	GPIO finish reconfiguration indication	output	DONE#

Table 12: CPU GPIO CPLD configuration pins

For more information, please see the Lattice MachXO2 Programming and Configuration Usage Guide, TN1204.

Warning: When configuring the CPLD in Lattice Diamond make sure sysConfig SLAVE_SPI_PORT is enabled, otherwise you will have to use the JTAG header to reprogram the GPIO CPLD.

7.5 HIGH SPEED COMMUNICATION

As shown in Figure 2, data communication between the GPIO CPLD and the CPU is done via the Main CPLD which acts as a temporary data storage device and therefore allows high speed burst transfers between the CPU and the GPIO CPLD.

There is an 8-bit data and control bus interface between the Main CPLD and the GPIO CPLD known as the **GMBus** (with the GPIO CPLD being the master of that bus, see Table 13: GPIO CPLD to Main CPLD pin connections), and the Main CPLD provides a set of registers that are accessible by the GPIO CPLD.

Similarly the CPU is also connected to the Main CPLD via its own 8-bit local bus interface known as the **Local Bus** (with the CPU the master of that bus), and the Main CPLD provides a set of registers that are accessible by the CPU.

Each interface also has its own mailbox interrupt that can be triggered by a write to a register from the other interface, and cleared by reading a register from its own interface. Therefore the GPIO CPLD can interrupt the CPU, and vice versa. Also each interface can write data to the shared dual port RAM. The combination of these features allows efficient interrupt driven data transfer between the CPU and the GPIO CPLD using a shared dual port RAM mailbox system.

There are two spare GPIO_CONTROL connections, GPIO_CONTROL[6] and GPIO_CONTROL[7], which have been implemented as an input and output respectively. The default GPIO CPLD image currently loops this signal back to the Main CPLD with an inversion to allow testing.

Signal	Description	I/O (GPIO CPLD)	GPIO Pin (GPIO CPLD)
DATA[7..0]	8 bit data	bidirectional	GPIO_DATA[7..0]
MBC2G_INT#	Mailbox CPU to GPIO CPLD interrupt	input	GPIO_CONTROL[0]
GM_ADDR0	0 = MS byte index, 1 = LS byte index	output	GPIO_CONTROL[1]
GM_DI	0 = index, 1 = data	output	GPIO_CONTROL[2]
GM_STB#	Read or write strobe	output	GPIO_CONTROL[3]
GM_RD#	Read access	output	GPIO_CONTROL[4]
LB_RESET#	Reset from Main CPLD	input	GPIO_CONTROL[5]
INTERNAL_LOOPBACK	GPIO OUT from Main CPLD	input	GPIO_CONTROL[6]
INTERNAL_LOOPBACK#	Invert of GPIO OUT from Main CPLD	output	GPIO_CONTROL[7]

Table 13: GPIO CPLD to Main CPLD pin connections

The memory map of the Main CPLD as seen by the GPIO CPLD is shown in Table 14Table 14: Main CPLD Memory Map Seen by GPIO CPLD below.

Address (hex)	Register name	Description	Read/ Write
0x00xx	SIG1	signature value 1 (0xDE)	RO
0x01xx	SIG2	signature value 2 (0xAD)	RO
0x02xx	PCB_VER	Hardware revision	RO
0x03xx	PLD_VER	CPLD version	RO
0x04xx	MBC2G	CPU to GPIO CPLD mailbox	RW
0x05xx	MBG2C	GPIO CPLD to CPU mailbox	RW
0x06xx	INT_STATUS	Mail box interrupt status (INT_MBC2G, INT_MBG2C)[1:0]	RO
0x07xx	MBC2G_RO	CPU to GPIO CPLD mailbox, read only	RO
0x08xx	MBG2C_RO	GPIO CPLD to CPU mailbox, read only	RO
0x5Cxx	SCR1	Scratch register	RO
0x6Axx	SCR2	Scratch register	RO
0x8000-0xFFFF	RAM	Dual port RAM, 8 bits wide, 8kbytes	RW

Table 14: Main CPLD Memory Map Seen by GPIO CPLD

Writing a vector byte to the MBC2G mailbox register from the CPU side (CPU to GPIO CPLD) will automatically assert the MBC2G_INT# interrupt to the GPIO CPLD. Reading the MBC2G register from the GPIO CPLD side will return the vector value that was written to it from the other side, and automatically de-assert the MBC2G_INT# interrupt.

Similarly writing a vector byte to the MBG2C mailbox register from the GPIO CPLD side (GPIO CPLD to CPU) will automatically assert the MBG2C_INT# interrupt to the CPU. Reading the MBG2C register from the CPU side will return the vector value that was written to it from the other side, and automatically de-assert the MBC2G_INT# interrupt.

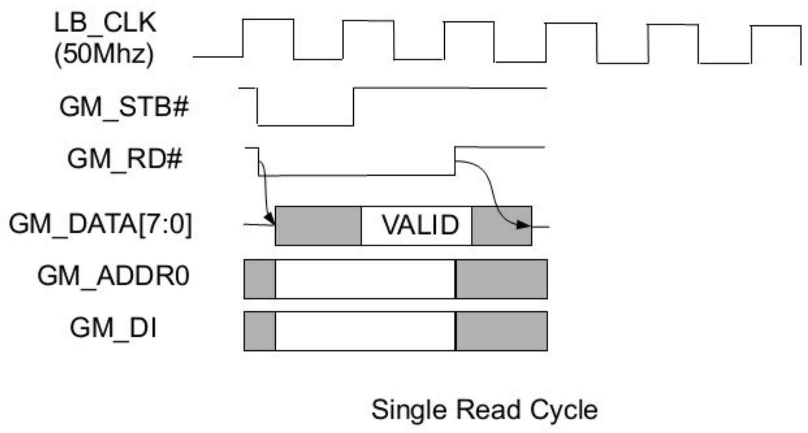
It is possible to read the MBG2C or MBC2G registers without clearing the respective interrupts by reading the MBG2C_RO or MBC2G_RO registers instead. This allows the vector byte to be passed to the interrupted device, and a decision made by that device on how to handle the interrupt based on that vector value, without clearing the respective interrupt which normally would indicate that the interrupt had been serviced or operation completed. When the operation has completed the normal MBG2C or MBC2G registers can be read to clear the interrupt.

Note that it is also possible from one interface to assert its own mailbox interrupt, and clear it. This is useful during testing, as it can simulate the operation of the other interface. For example if the CPU were to fill the SRAM with some test data that simulated what the GPIO CPLD would send, then write to the MBG2C register to assert its own MBG2C_INT# signal. The whole interrupt routine can be fully tested and simulated without the GPIO CPLD needing to be involved.

7.5.1 TIMING DIAGRAMS

Figure 4Figure 4: GM Bus, single read and writes shows single read and write timing diagrams on the GM Bus. Note these are 3 clock cycles long and that there is always a single clock 'recovery' cycle at the end of a read or a write, this gives time for whichever device is driving the GM_DATA bus to tri-state it ready for the next access. The position of the GM_STB# is different for reads and writes as there is a one clock pipeline delay for reads. The GM_RD# signal is used to indicate a read access, and is additionally used by the Main CPLD as an output enable for the GM_DATA bus.

GM_DI differentiates between index or data access (0 = index, 1 = data), and for index accesses GM_ADDR0 differentiates between high or low order index register (0 = high order, 1 = low order).



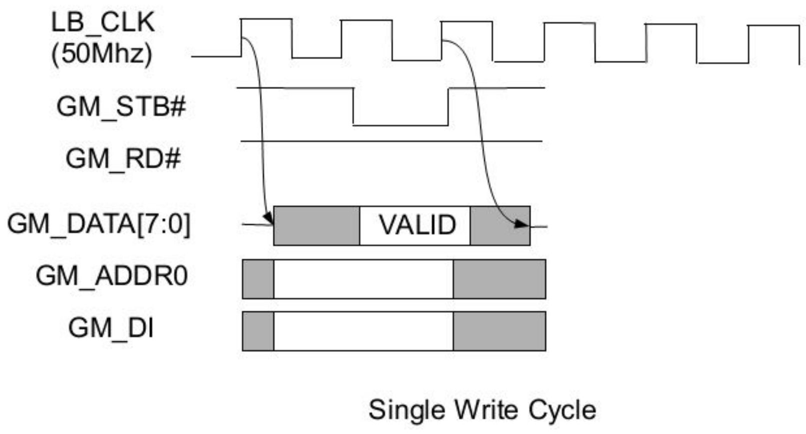
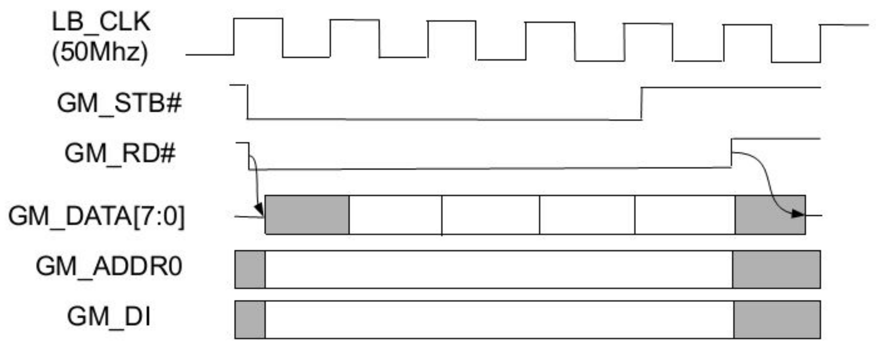
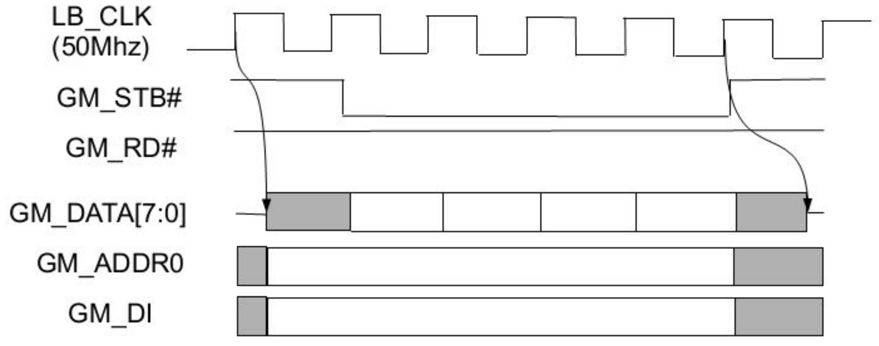


Figure 4: GM Bus, single read and writes



GM Bus Burst Read Cycle



GM Bus Burst Write Cycle

Figure 5: GM Bus, burst read and writes shows burst read and write cycles on the GM Bus. This interface is clocked at 50 MHz, and this allows data to be transferred between the GPIO CPLD and the Main CPLD at 50MB/sec. Figure 5: GM Bus, burst read and writes

7.6 SPI Communication

There is an SPI interface directly between the CPU and the GPIO CPLD.

The default firmware in the GPIO CPLD has an SPI slave with 4 registers defined. The addresses of these registers are shown in Table 15Table 15: GPIO SPI register map below.

Address (hex)	**Register name**	**Description**	**Read/Write**
0x00	SIG1	signature value 1 (0xDE)	RO
0x01	SIG2	signature value 2 (0xAD)	RO
0x02	Scratch	Scratch register	RW
0x03	LED	GPIO LED control	RW

Table 15: GPIO SPI register map

The Scratch register is 8 bits wide and can be written and read back via SPI. The bottom 2 bits of the LED register (LED[1:0]) are active high LED on control signals which are mapped to GPIO_LED[2:1] respectively, see Figure 3.

The format of the SPI accesses for a read and write cycles is shown in Figure 6Figure 6: SPI read and write format below.

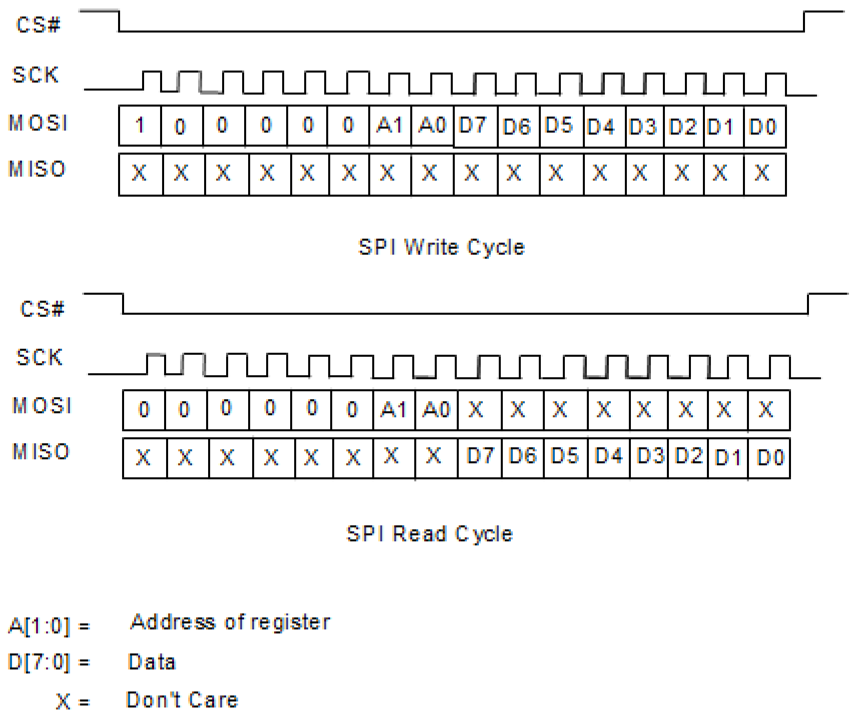

CS#
SCK
MOSI
1 0 0 0 0 0 A1 A0 D7 D6 D5 D4 D3 D2 D1 D0
MISO
X X X X X X X X X X X X X X X X
SPI Write Cycle
CS#
SCK
MOSI
0 0 0 0 0 0 A1 A0 X X X X X X X X
MISO
X X X X X X X X D7 D6 D5 D4 D3 D2 D1 D0
SPI Read Cycle
A[1:0] = Address of register
D[7:0] = Data
X = Don't Care

Figure 6: SPI read and write format

7.7 GPIO Headers

The two GPIO headers have an identical pinout and all the pins have ESD protection. The pin out of the headers is shown in Table 16.

Pin	Signal	Signal	Pin
1	VGPIO-3V3	VGPIO-5V	2
3	GPIO 0	VGPIO-5V	4
5	GPIO 1	GROUND	6
7	GPIO 2	GPIO 3	8
9	GROUND	GPIO 4	10
11	GPIO 5	GPIO 6	12
13	GPIO 7	GROUND	14
15	GPIO 9	GPIO 8	16
17	VGPIO-3V3	GPIO 12	18
19	GPIO 10	GROUND	20
21	GPIO 11	GPIO 14	22
23	GPIO 13	GPIO 15	24
25	GROUND	GPIO 16	26

Table 16: GPIO headers pinout

Warning: The orientation of the GPIO2 header is 180 degrees rotated with respect to the GPIO1 header. See Figure 3**Figure 3: GPIO CPLD headers and LEDs positions.**

7.8 LEDS

A pair of simple LEDs is provided for diagnostic purposes. These GPIO_LED2 (green) and GPIO_LED1 (red) and are connected to pins 98 and 99 of the GPIO CPLD respectively. The signals that drive these LEDs are active low to illuminate the LEDs.

7.9 DEBUG PINS

There are 8 debug pins which are connected to test points, these have ESD protection and the test points are dotted around the GPIO CPLD on both sides of the PCB. The test point connections of the debug pins are shown in Table 17 below.

Test Point	GPIO Signal
TP40	Debug0
TP41	Debug1
TP42	Debug2
TP43	Debug3
TP44	Debug4
TP45	Debug5
TP46	Debug6
TP47	Debug7

Table 17: GPIO debug pin test points

7.10 GPIO CPLD PIN OUT

The pinout of the GPIO CPLD is given below in Table 18.

Pin Name	Pins(LSB to MSB)
GPIO1[0..16]	7,13,3,9,8,12,4,10,35,39,29,37,47,40,78,42,28
GPIO2[0..16]	83,43,45,41,18,16,27,14,20,24,2,15,1,17,21,19,38
GPIO_DATA[0..7]	75,74,71,70,69,68,67,66
GPIO_CONTROL[0..7]	65,64,60,59,58,57,54,53
LB_CLK	63
GPIO_INT#	25
SPI_MISO	32
SPI_MOSI	49
SPI_SCK	31
SPI_CS#	48
DEBUG[0..7]	97,96,51,88,87,86,85,84
GPIO_LED[1..2]	99,98

Table 18: GPIO CPLD pinout

8 MCU

The MCU is a supervisor for the Tabor motherboard and provides voltage and temperature monitoring for the CPU.

8.1 SUPERVISOR INTERFACE

The supervisor interface is like an ACPI and is connected over serial port 1 to the CPU. The connection should be setup using:

- 38400 Baud
- 8 bit data
- No Parity
- 1 Stop bit
- No Flow Control

Each packet has a start and end character. The CPU can tell the MCU to turn off the power, get temperatures, get voltages and read the CPU fan speed.

Commands can be pipelined as the serial interface is interrupt driven, and responses contain the command it is responding to.

All MCU to CPU messages start with '$' and end with a new line character, ASCII 0x0A.
All CPU to MCU messages start with '#' and end with a new line character, ASCII 0x0A.

8.1.1 POWER BUTTON

When the power button is pressed, a 1ms low pulse is generated on IRQ4# to the CPU. The CPU can use the interrupt to cleanly shut down the OS and the power supplies via the ACPI like serial interface.

Holding the power button for greater than 5 seconds will force the power off.

8.1.2 SHUT POWER DOWN

To shut down the power of the Tabor motherboard from the CPU, the '**s**' command is used.
Example:

CPU command	#s
MCU Returns	$s

8.1.3 TEMPERATURES READINGS

To read the temperature readings, the '**t**' command is used.

Returns the temperatures in the following format:
$t<sign>HH...<sign>HH

Where HH is the ASCII hex value of the temperature (8 bit signed value), and <sign> is either '+' or '-'. HH is in the range -128°C to 128°C.

There are three temperatures available to read on the Tabor motherboard. The temperatures given are returned in this order:

1. PCB temperature
2. CPU temperature
3. PCB temperature 2

Example:

CPU command	`#t`
MCU Returns	`$t+20+35+1C`

Represents
+32°C for the PCB temperature
+56°C for the CPU temperature
+28°C for the PCB temperature 2

8.1.4 VOLTAGES

To read all the measured voltages, the '**v**' command is used.

Returns the voltages in the following format:
$vXXYY...XXYY

Where XX represents the whole number of volts as ASCII hex, YY represents the number of 10mV units as ASCII hex.

The voltages are sent in the order:

1. CPLD 3.3V
2. 3.3V
3. 2.5V
4. Audio 5.0V
5. Audio 3.3V
6. ATX 3.3V
7. CPLD 2.5V
8. 1.8V
9. Ethernet 1.2V
10. Core 1.05V
11. DDR3 IO 1.5V

Example for the above default values:

CPU command	`#v`
MCU Returns	`$v03210324023205030323032002340151011601050132`

Examples:

CPLD 3.3V :	**0x03 0x21 = 3.33V**
1.8V :	**0x01 0x51 = 1.81V**
DDR3 IO 1.5V :	**0x01 0x32 = 1.50V**

8.1.5 CPU FAN SPEED

To read the CPU fan speed, the '**f**' command is used. This returns the fan speed (in RPM) and the fan duty cycle (0 -> 255, where 0 is off and 255 is full on).

Returns the fan status in the following format:
$fXXYYYY

Where XX is the ASCII hex value of the PWM (0x00 to 0xFE) YYYY is the ASCII hex value of the RPM, MSB first.

Example:

CPU command	#f
MCU Returns	$fFE2A6C

Represents
Fan PWM, 0xFE = 254 decimal
Fan RPM, 0x2A6C= 10860 decimal

8.2 DEBUG SERIAL TERMINAL

The MCU also provides serial debug interface for status reporting of the read voltages and temperature rails. This uses a 6 pin FTDI USB-TTL cable pinout (P15), see Figure 7 for location.

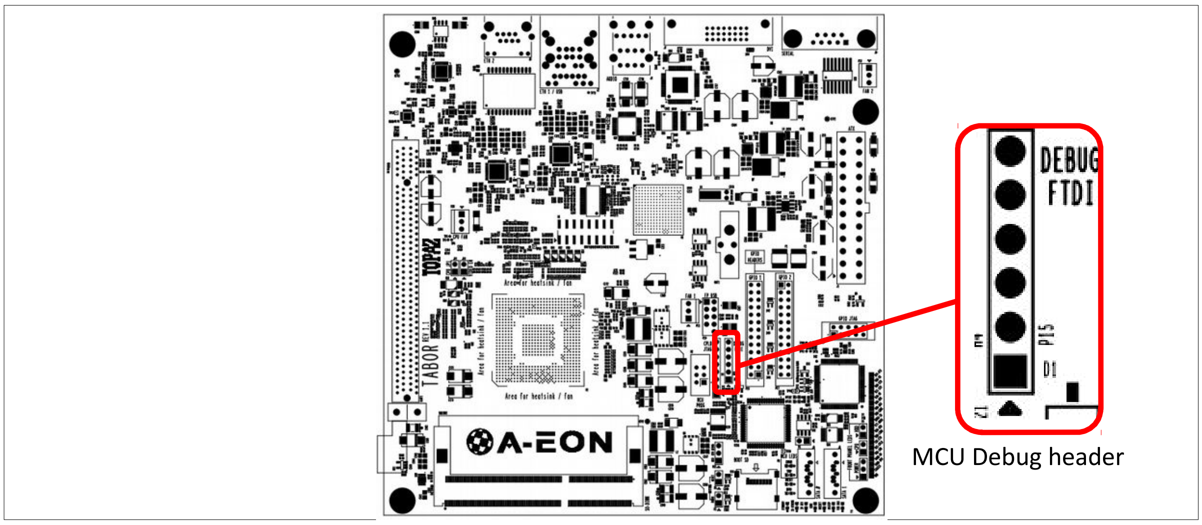


Figure 7: MCU Debug header

The pinout of the Debug serial terminal is give below in Table 19.

Pin	MCU Connection	MCU Direction
1	Ground	-
4	MCU RX	In
5	MCU TX	Out

Table 19: MCU serial pinout

Notes:
Pins 2-4 and 6 are unconnected.

To set up a serial communications on a PC, it is recommended to use TeraTerm. The serial port must be configured as follows:

- 38400 Baud
- 8 bit data
- No Parity
- 1 Stop bit
- No Flow Control

9 BOOT

This section contains specific Tabor boot information on the micro SD card and U-Boot.

9.1 MICRO SD CARD

The micro SD card contains all the U-Boot data. This is required to boot the system to U-Boot. The first 10MB or 0x5000 blocks contain the U-Boot with all the boot loader and environment settings. The boot loader data should not be edited when accessing the SD card otherwise the system will cease to boot. The structure of the boot loader blocks of the micro SD card is shown in Table 20 below.

Function	Start block	Block count	End block
Pre-boot U-Boot	0x0000	256	0x00FF
Main U-Boot	0x0100	1792	0x07FF
Boot images	0x0800	10240	0x2FFF
Environment	0x3000	16	0x3010
OS4BootLoader	0x4000	4096	0x4FFFF

Table 20: SD boot loader blocks

9.2 MAIN U-BOOT

Tabor uses a standard version of U-Boot configured for the Tabor hardware. For further reference of the U-Boot commands check out U-Boot website.

There are specific environment settings which need to be configured for the system work correctly, these are shown in Table 21. These should not be edited as they will affect the functionality of Tabor Motherboard.

Environment name	Environment value
`baudrate`	`115200`
`consoledev`	`ttyS0`
`hwconfig`	`esdhc;audclk:12;usb1:dr_mode=host,phy_type=ulpi`
`stdin`	`serial,usbkbd`
`stdout`	`serial,vga`
`video-mode`	`fslfb:800x600-32@60,monitor=dvi`

Table 21: U-Boot critical settings

There are specific environment settings for booting Amiga OS and these are listed in Table 22 below.

Environment name	Environment value
`aosautoboot`	`if run aosusbboot; then echo OK; else sata init; if run aossata0boot; then echo OK;` `echo if run aossata1boot; then echo OK; else run aosnetboot; fi; fi; fi`
`bootmenu_0`	`OS4 Auto Boot=run aosautoboot`
`bootmenu_1`	`USB Boot=run aosusbboot`
`bootmenu_2`	`Net Boot=run aosnetboot`
`bootmenu_3`	`SATA 0 Boot=sata init; run aossata0boot`
`bootmenu_4`	`SATA 1 Boot=sata init; run aossata1boot`

Table 22: U-Boot Amiga OS boot setting

10 SWITCHES, JUMPERS AND LEDS

10.1 SWITCHES

Headers are provided for front panel power and reset switches/buttons of the momentary, normally open type.

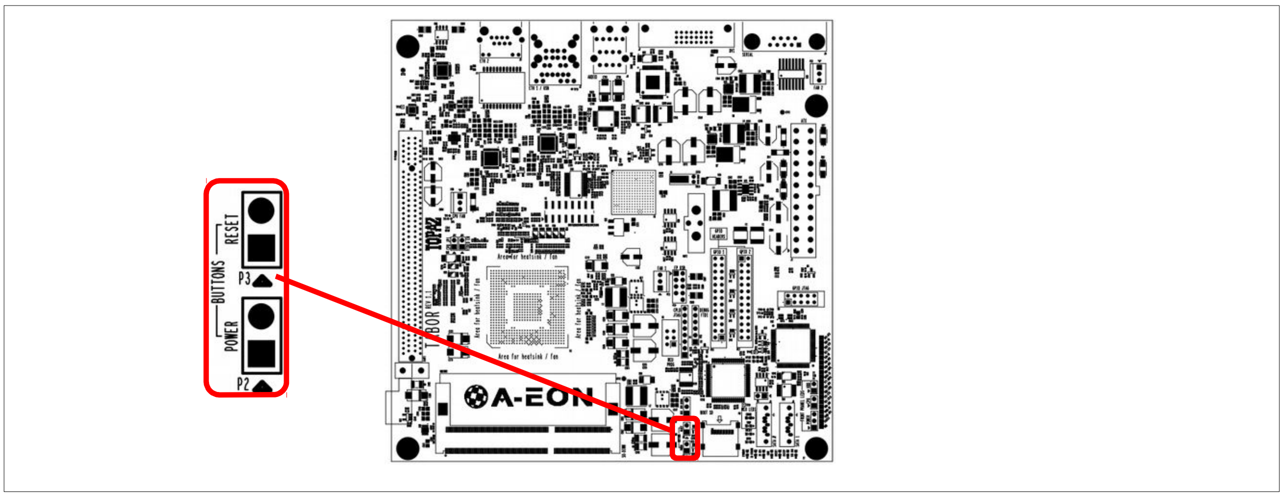


Figure 8: Front Panel switch headers locations

P2 (labelled POWER) is for the power button. P3 (RESET) is for the reset button. For both headers pin 1 is grounded and pin 2 is pulled up to 3.3V.

10.2 JUMPERS

Jumpers are provided to select boot configuration options, there position on the motherboard as shown in Figure 9.

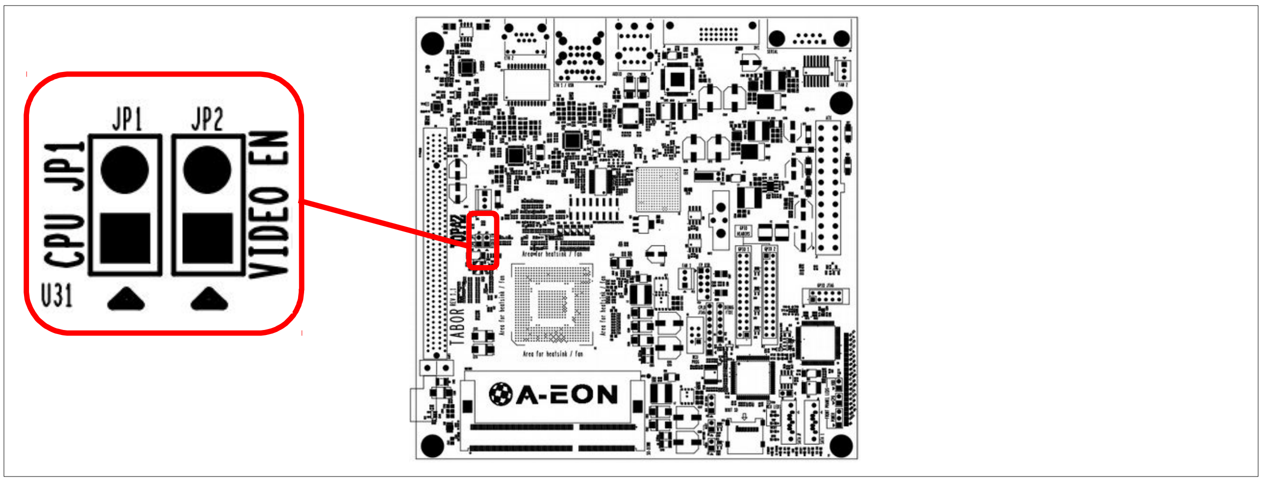


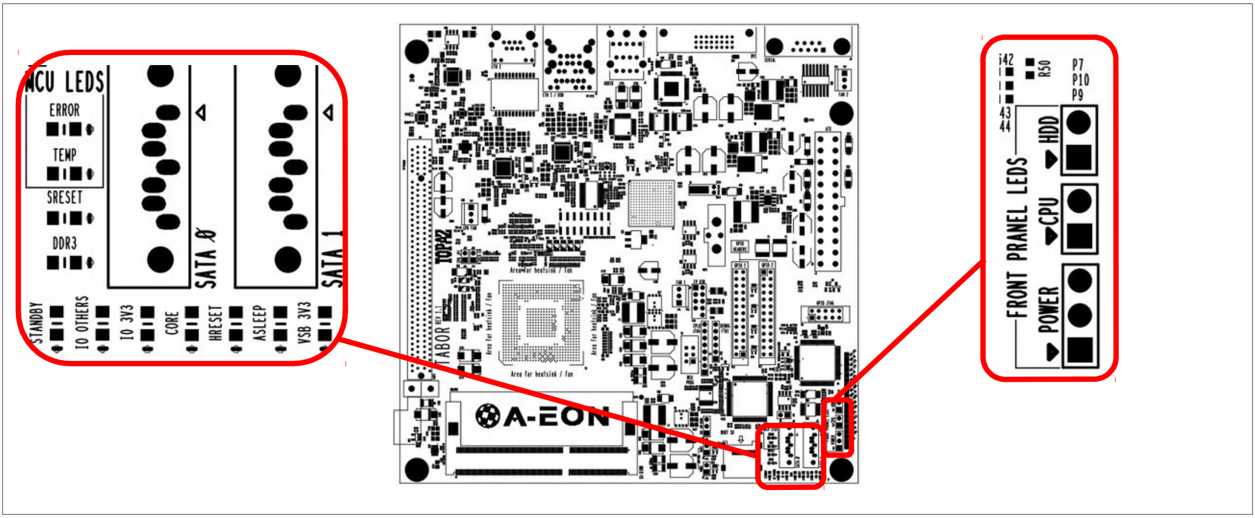
Figure 9: Tabor jumpers locations

Jumper	Description
JP1	Controls CPU GPIO0, 1 = Jumper unfitted, 0 = Jumper fitted.
JP2	Controls VGA enable, 1 = U-Boot serial (jumper unfitted), 0 = U-Boot VGA if available (jumper fitted).

Table 23: Jumpers descriptions

10.3 LEDS

Figure 10: Tabor LEDs locations



Tabor provides 11 on-board LEDs and headers for 3 off board LEDs. Their location on the motherboard is shown in Figure 10.

Reference	Type	Description
VSB 3V3 (LED1)	Green	Standby power indicator
ASLEEP (LED6)	Red	ASLEEP state indicator (CPLD controlled)
HRESET (LED4)	Red	HRESET state indicator (CPLD controlled)
CORE (LED2)	Red	CPU Core voltage power good (CPLD controlled)
IO 3V3 (LED11)	Red	3.3V voltage rail power good (CPLD controlled)
IO OTHERS (LED12)	Red	IO voltages power good (CPLD controlled)
STANDBY (LED13)	Red	STANDBY state indicator (CPLD controlled)
DDR3 (LED3)	Red	DDR3 voltages power good (CPLD controlled)
SRESET (LED5)	Red	SRESET state indicator (CPLD controlled)
TEMP (LED7)	Red	MCU temperature Warning LED
ERROR (LED8)	Red	MCU Error LED
POWER (P9)	0.1" header	Power on (pins 1 & 2 = anode, pin 3 = cathode)
HDD (P10)	0.1" header	SATA activity, CPU GPIO5 (pin 1 = anode, pin 2 = cathode)
CPU (P7)	0.1" header	CPU GPIO4 (pin 1 = anode, pin 2 = cathode)

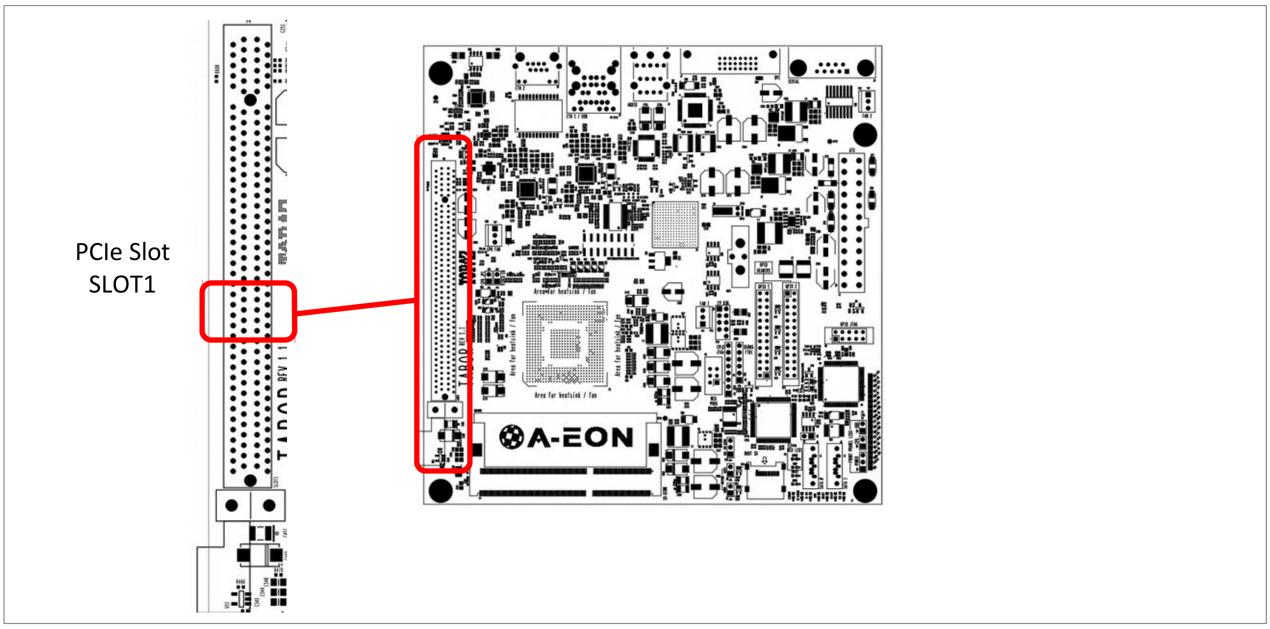
Table 24: LEDs description

Notes:
The LED header drivers are of the constant current (20mA) type and are suitable for driving LEDs with a forward voltage of between 2 and 12V (they will directly drive any standard LED assuming it is rated for 20mA or more). P12 is pinned out as follows: 1,2 = +/anode, 3 = -/cathode. P10 and P13 are pinned out as follows: 1 = +/anode, 2 = -/cathode. Pin 1 is marked by an arrow in each case.

10.4 PCIE SLOT

The pinout of the PCIe slot 1 is shown in Table 25 and the position of the connector for the PCIe slot is shown in Figure 11.

Figure 11: PCIe slot location



pin	row A	row B	pin	row A	row B
1	PRSNT1#	+12V	42	GND	PETn6
2	+12V	+12V	43	PERp6	GND
3	+12V	+12V	44	PERn6	GND
4	GND	GND	45	GND	PETp7
5	TCK	SMCLK	46	GND	PETn7
6	TDI	SMDAT	47	PERp7	GND
7	TDO	GND	48	PERn7	PRSNT2#
8	TMS	+3.3V	49	GND	GND
9	+3.3V	TRST#	50	RSVD	PETp8
10	+3.3V	3.3Vaux	51	GND	PETn8
11	PERST#	WAKE#	52	PERp8	GND
12	GND	RSVD	53	PERn8	GND
13	REFCLK+	GND	54	GND	PETp9
14	REFCLK-	PETp0	55	GND	PETn9
15	GND	PETn0	56	PERp9	GND
16	PERp0	GND	57	PERn9	GND
17	PERn0	PRSNT2#	58	GND	PETp10
18	GND	GND	59	GND	PETn10
19	RSVD	PETp1	60	PERp10	GND
20	GND	PETn1	61	PERn10	GND
21	PERp1	GND	62	GND	PETp11
22	PERn1	GND	63	GND	PETn11
23	GND	PETp2	64	PERp11	GND
24	GND	PETn2	65	PERn11	GND
25	PERp2	GND	66	GND	PETp12
26	PERn2	GND	67	GND	PETn12
27	GND	PETp3	68	PERp12	GND
28	GND	PETn3	69	PERn12	GND
29	PERp3	GND	70	GND	PETp13
30	PERn3	RSVD	71	GND	PETn13
31	GND	PRSNT2#	72	PERp13	GND
32	RSVD	GND	73	PERn13	GND
33	RSVD	PETp4	74	GND	PETp14
34	GND	PETn4	75	GND	PETn14
35	PERp4	GND	76	PERp14	GND
36	PERn4	GND	77	PERn14	GND
37	GND	PETp5	78	GND	PETp15
38	GND	PETn5	79	GND	PETn15
39	PERp5	GND	80	PERp15	GND
40	PERn5	GND	81	PERn15	PRSNT2#
41	GND	PETp6	82	GND	RSVD

Table 25: PCIe x16 Slot Pinout

Note: Slot 1 lanes 4-15 are always no connect.

10.5 PROGRAMMING HEADERS

The locations for the programing headers are located below in Figure 12.

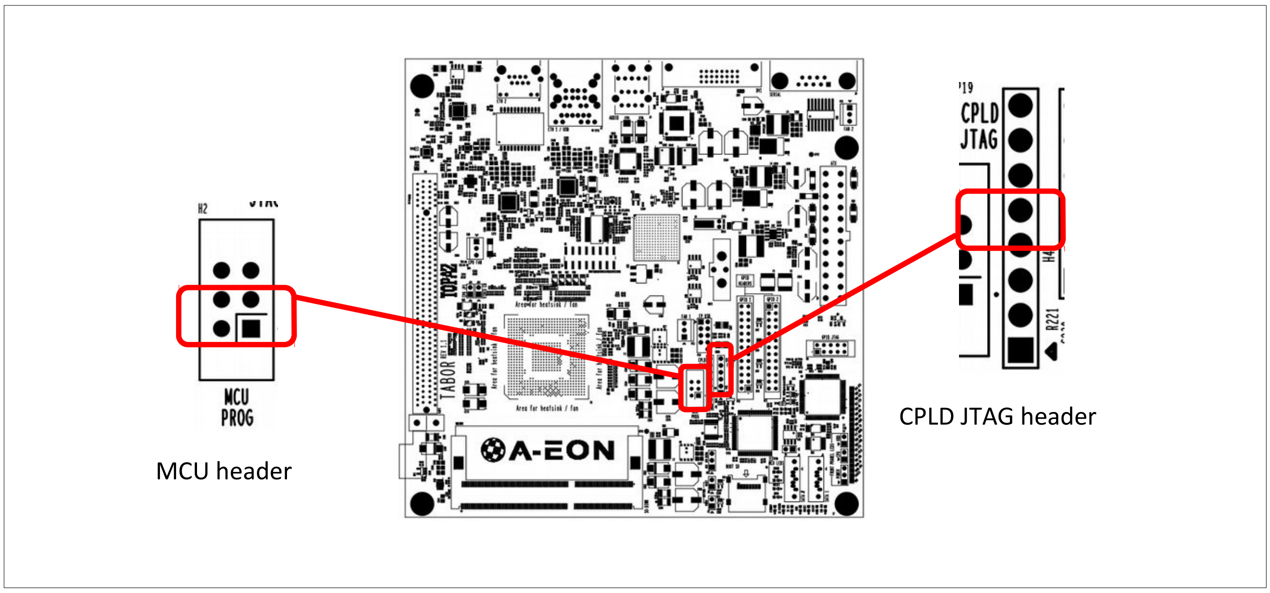


Figure 12: Programming Headers locations

10.5.1 MAIN CPLD

The pinout for H4 (labelled CPLD JTAG) is shown in Table 26Table 26: CPLD JTAG Header below:

Pin	Signal
1	3.3VSB
2	TDO
3	TDI
4	n/c
5	n/c
6	TMS
7	Ground
8	TCK

Table 26: CPLD JTAG Header pinout

10.5.2 MCU

The pinout for H2 (labelled MCU PROG) is shown in Table 27Table 27: MCU Programming Header below:

Pin	Signal
1	PDI DATA
2	3.3VSB
3	n/c
4	n/c
5	PDI CLK
6	Ground

Table 27: MCU Programming Header pinout