PCI6713–MFIO

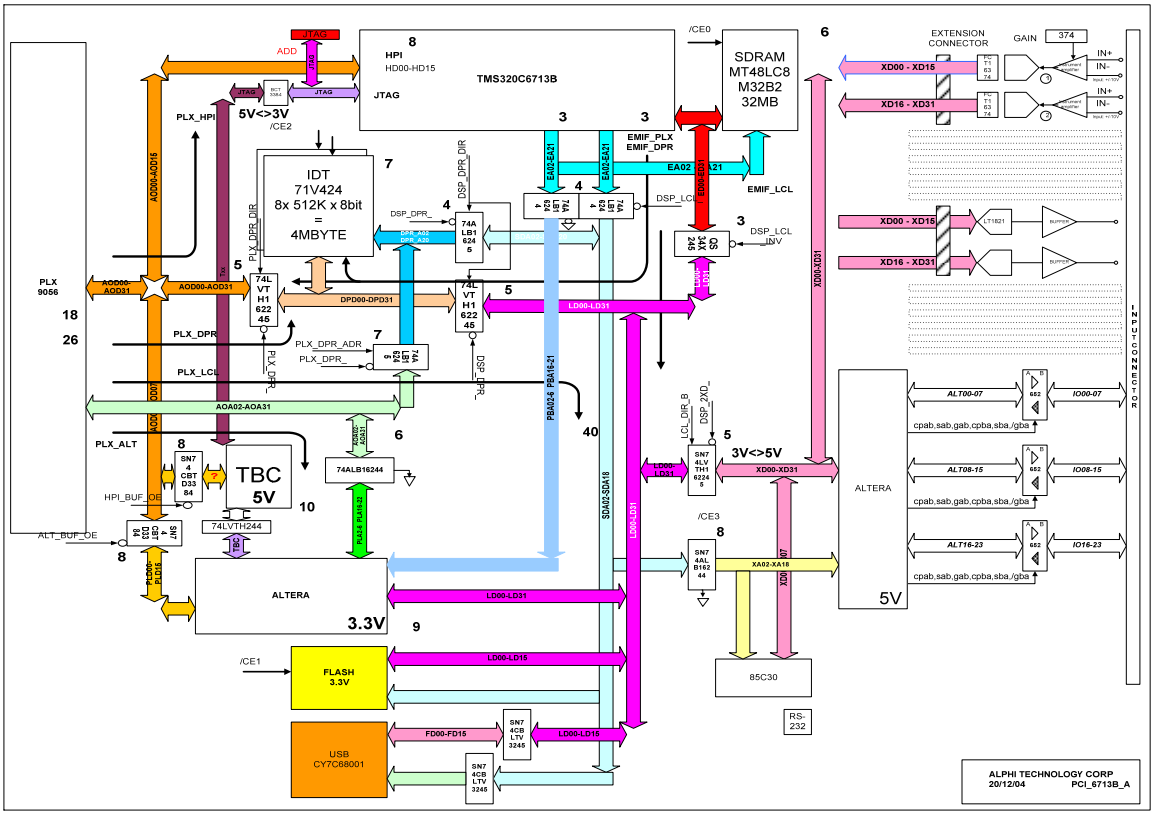
PCI DSP Data Acquisition System

HARDWARE REFERENCE MANUAL

Revision 1.4.1
February 13, 2006

ALPHI TECHNOLOGY CORPORATION
6202 S. Maple Avenue #120
Tempe, AZ 85283 USA
Tel : (480) 838 - 2428
Fax: (480) 838 - 4477
www.alphitech.com
support@alphitech.com

1 BLOCK DIAGRAM



2 ARCHITECTURE

The main processor is a TMS320C6713B with a system clock up to 300MHZ.
Other board resources include:

- Up to 32Mbytes of SDRAM
- 4 Mbytes of SRAM dual-ported between PLX-controlled PCI accesses and the TMS320C6713B
- 512KB Flash memory for the bootstrap program
- USB controller
- RS232C controller
- Add-on connector for A/D, D/A and I/O plug-in module
- Cyclone FPGA for decoding

2.1 THE 6713B BOOT PROCESS

The TMS320C6713/13B device has two boot modes: from the HPI or from external asynchronous FLASH EEPROM. On the TMS320C6713B, boot mode is determined during the device reset.

Refer to the TI application note spra512 for details on booting through the HPI.

When using the FLASH to boot, the first 1K of the FLASH are copied, starting to the local address 0x00000000 of the processor. After the end of the transfer, the CPU starts executing at the address 0. The small program thus loaded has the responsibility of doing at least a minimum configuration of 6713 registers and of copying a bigger program that will start executing and actually boot the board.

The board interface is configured as Big-Endian with the 8-bit data on the ED[31:24] side of the bus. HPI is always enabled

SW3_HD3	BOOT CONFIGURATION
No jumper	CE1 width 32-bit, HPI boot/Emulation boot
jumper	CE1 width 16-bit, Asynchronous external ROM boot with default timings

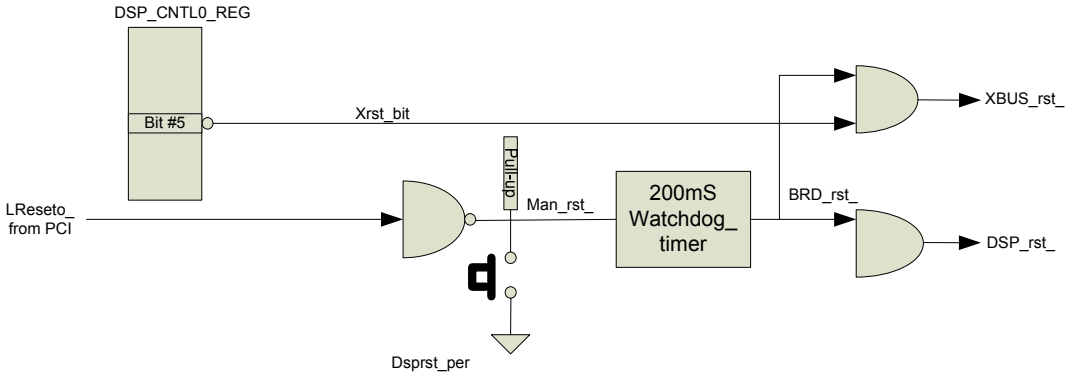
SW3[2..0] are used as factory test points.

2.2 RESET SCHEME

The TMS6713B-MFIO board has multiple source of reset. Upon Power-on a watch-dog timer provide a 200ms minimum delay where the DSP is maintained on reset. This gives each programmable logic chip enough time to get its program loaded.
The board can be reset by:

-Power-on reset
-PCI reset line /LRESETO from PLX
-Push-button

All these methods use the watch-dog timer for delay.


DSP_CNTL0_REG
Bit #5
Xrst_bit
XBUS_rst_
Pull-up
LReseto_
from PCI
Man_rst_
200mS
Watchdog_
timer
BRD_rst_
DSP_rst_
Dsprst_per

3 6713 Local Peripherals

3.1 6713 Internal registers

3.1.1 EMIF programming

The board uses the 6713 EMIF module (External Memory Interface) to generate on-board timings.

Using the EMIF, the memory space is divided as:

MEMORY SPACE	SETTING	RESOURCE	SIZE	ADDRESS RANGE
CE0	0xffffbf33	SDRAM	8-32MB	0x80000000-0x81FFFFFF
CE1	0x12328620	ADD_ON + PERIPHERALS	4MB	0x90000000-0x903FFFFF
CE2	0x00c10320	DUAL-PORTED SRAM	4MB	0xA0000000-0xA31FFFFF
CE3	0x12328620	FLASH (16-bit wide)	4MB	0xB0000000-0xB03FFFFF

The default EMIF configuration in accordance with the PLL settings has been selected to give the maximum performance without sacrificing signal integrity. We recommend not changing it, as an incorrect setting will prevent the board from functioning properly.

Default EMIF configuration:

REGISTER	ADDRESS	SETTING
EMIF_GCTL	0x01800000	0x00000078
EMIF_CE0	0x01800008	0xffffbf33
EMIF_CE1	0x01800004	0x12328620
EMIF_CE2	0x01800010	0x00c10320
EMIF_CE3	0x01800014	0x12328620
EMIF_SDRAMCTL	0x01800018	0x46216000
EMIF_SDRAMTIM	0x0180001C	0x00000618
EMIF_SDRAMEXT	0x01800020	0x00054529

3.1.2 PLL programming

The outside 50MHZ clock provided to the DSP is multiplied by the programmable internal PLL (x12) generating a 600 MHZ internal clock. This clock is then divided by two to provide SYSCLK or CLKOUT2 (300 MHZ).
The outside bus timings are generated by the TMS320C6713B EMIF using CLKOUT3 (the 600MHZ clock divided by 6 by the PLL # 2) which is a 100 MHZ clock.
In any case, the EMIF clock cannot exceed 100MHZ.
We strongly recommend using the default timings.

```
/*----------------------------------------------------------------*/
/* C6713 PLL SUPPORT                                              */
/*----------------------------------------------------------------*/
#define PLL_BASE_ADDR           0x01b7c000
#define PLL_PID                 ( PLL_BASE_ADDR + 0x000 )
#define PLL_CSR                 ( PLL_BASE_ADDR + 0x100 )
#define PLL_MULT                ( PLL_BASE_ADDR + 0x110 )
#define PLL_DIV0                ( PLL_BASE_ADDR + 0x114 )
#define PLL_DIV1                ( PLL_BASE_ADDR + 0x118 )
#define PLL_DIV2                ( PLL_BASE_ADDR + 0x11C )
#define PLL_DIV3                ( PLL_BASE_ADDR + 0x120 )
#define PLL_OSCDIV1             ( PLL_BASE_ADDR + 0x124 )

#define CSR_PLLEN               0x00000001
#define CSR_PLLPWRDN            0x00000002
#define CSR_PLLRST              0x00000008
#define CSR_PLLSTABLE           0x00000040
#define DIV_ENABLE              0x00008000

/*----------------------------------------------------------------*/
/* init_pll()                                                     */
/* Pll Initialization                                             */
/*----------------------------------------------------------------*/
void init_pll()
{
   /*-------------------------------------------------------------*/
   /* When PLLEN is off DSP is running with CLKIN clock           */
   /* source, currently 50MHz or 20ns clk rate.                   */
   /*-------------------------------------------------------------*/
   *(int *)PLL_CSR  &= ~CSR_PLLEN;

   /* Reset the pll.  PLL takes 125ns to reset. */
   *(int *)PLL_CSR  |= CSR_PLLRST;

   /*-------------------------------------------------------------*/
   /* PLLOUT = CLKIN/(DIV0+1) * PLLM                              */
   /* 450    = 50/1 * 9                                           */
   /*-------------------------------------------------------------*/
   *(int *)PLL_DIV0    = DIV_ENABLE + 0;
   *(int *)PLL_MULT    = 12;
   *(int *)PLL_OSCDIV1 = DIV_ENABLE + 4;

   /*-------------------------------------------------------------*/
   /* Program in reverse order.                                   */
   /* DSP requires that pheriheral clocks be less then            */
   /* 1/2 the CPU clock at all times.                             */
   /*-------------------------------------------------------------*/
   *(int *)PLL_DIV3    = DIV_ENABLE + 5;          // 100 MHz EMIF
   *(int *)PLL_DIV2    = DIV_ENABLE + 3;
   *(int *)PLL_DIV1    = DIV_ENABLE + 1;
   *(int *)PLL_CSR     &= ~CSR_PLLRST;

   /*-------------------------------------------------------------*/
   /* Now enable pll path and we are off and running at           */
   /* 300MHz with 100 MHz SDRAM.                                  */
   /*-------------------------------------------------------------*/
   *(int *)PLL_CSR |= CSR_PLLEN;
}
```

3.2 CE0 space

The SDRAM is mapped in the CE0 space. Depending on your board options, up to 32 megabytes are available (4 megabytes is standard). The SDRAM is connected to the 6713B bus. The data and address buses are isolated to the local bus by buffers.

3.3 CE1 space

CE1 space is share between the boot flash, all peripherals including the ADD_ON module. A Cyclone FPGA is dedicated to 48 digital I/O pin buffered by a group of 6 transceivers. Each transceiver can be set as output or inputs. More specification can be found on the CYCLONE I/O module.

PERIPHERALS	SIZE	ADDRESS RANGE
FLASH	2MB	0x90000000 - 0x901F FFFF
ADD_ON (ADDA)	1MB-8KB	0x90200000 - 0x902F 7FFF
PLX_REG	1KB	0x902F E000 - 0x902F E3FF
SCC8530	1KB	0x902F E400 - 0x902F E7FF
CYCLONE REG	1KB	0x902F E800 - 0x902F EBFF
USB CONTROLLER	1KB	0x902F EC00 - 0x902F EFFF
CYCLONE_IO	1KB	0x902F F000 - 0x902F F3FF
PCI_BUS	1MB	0x9030 0000 - 0x903FFFFF

3.3.1 FLASH EPROM

A 256Kx16 Flash memory M29F400DB is visible at the base address of CE1 (0x0x90000000) space for use during the boot process. It is also visible in CE3 (0xB0000000). The address space CE1 is 32-bit wide once the board has booted, so when accessing the FLASH, only the lower 16 bits of each 32-bit word contains actual data.

By contrast, the CE3 space is 16-bit wide, matching the FLASH actual geometry and the data is continuous.

3.3.2 Cyclone Registers

The cyclone logic has a set of locals registers used to set-up some local parameters or to communicate with the PLX bus.

3.3.2.1 DSP_CNTL0_ADR 0x902F E800

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
XCNTL1_O	XCNTL0_O	XRESET	NMIEN	x	FLASH _PAGE	!/LED_ O[1]	!/LED_ O[0]

/LED_O[1..0] : This register controls 2 LEDs. The reset position is for the LEDs to be "off". A "1" in the position corresponding to the LED will set it "on".

FLASH_PAGE: Currently unused, should be 0.

NMIEN: TMS320C6713B allows receiving the NMI interrupt from multiple sources. The default is 0: "disabled".

XRESET: A "1" will put the extended bus in a reset mode until it is cleared. The default is "no reset".

XCNTL[1..0]: Two controls lines activated by DSP for extended Xbus use. Defaults "0".

3.3.2.2 DSP_STAT0_ADR 0x902F E804

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
XSTAT1	XSTAT0	/USB_RDY	DSPNMI	RY_BY_FLASH	8530_IRQ	/LINTO	USERO

XSTAT[1..0]: Two status lines for extended Xbus use. The default at reset is "0".

/USB_RDY Ready signal from USB

DSPNMI: Status of the PCI-generated bit to activate an NMI interrupt to the processor

RY_BY_FLASH Status of the RY_BY pin from flash memory for use in its programming.

8530_IRQ Status of interrupt from SCC8530: when the value is "0", the interrupt is active.

/LINTO Status of interrupt line activated by the PCI bus through the PLX 9056.

USERO Status of USERO line activated by the PCI bus through the PLX 9056.

3.3.2.3 DSP_CNTL1_ADR 0x902F E808

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
0	INT_8530_EN	INT_VSB_EN	LINTO_DIR	USB_DIR	LOCKOUT 9056	WAKE_USB	/USB_RST

LINTO_DIR A "1" remove the direct connection of LINTO interrupt to the EXT_NMI interrupt.

USB_DIR A "1" remove the direct connection of USB interrupt to the EXT_INT[7] interrupt.

LOCKOUT 9056 Control bit from DSP to solve a possible bottleneck arbitration scheme

WAKE_USB Set to "1" upon reset.

/USB_RST Set to "1" upon reset

INT_VSB_EN When set to"1" enables the USB interrupt as a source of the XINT_INT6 interrupt.

INT_8530_EN When set to"1" enables the 8530 interrupt as a source of the XINT_INT6 -interrupt.

3.3.2.4 DSP_STAT1_ADR 0x902F E80C

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
/USB_ INT	USB_CABLE	FLAGC	FLAGB	FLAGA	S_USER 2	S_USER 1	S_USER 0

S_USER[2..0]: Jumper user status

FLAGA: Indicate the internal flag status of the selected FIFO. When "1" the FIFO is EMPTY.

FLAGB: Indicate the internal flag status of the selected FIFO. When "1" the FIFO is FULL.

FLAGC: Indicate the internal flag status of the selected FIFO. When "1" the FIFO is programmable.

USB_CABLE: When "1" USB cable is present.

/USB_INT Status of the USB interrupt line

3.3.2.5 DSP_CNTL2_ADR 0x902F E810

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
D_DSP7	D_DSP6	D_DSP5	D_DSP4	D_DSP3	D_DSP2	D_DSP1	D_DSP0

This register can be used as an inter-processor communication register with the PLX 9056.

3.3.2.6 DSP_STAT2_ADR 0x902F E814

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
P_CNT7	P_CNT6	P_CNT5	P_CNT4	P_CNT3	P_CNT2	P_CNT1	P_CNT0

This register can be used as an inter-processor communication register with the PLX 9056. Its value reflects what was written in the PCI_CNTL1 register.

3.3.2.7 DSP_IPINT_ADR 0x902F E820-E85C

The TMS320C6713B supports 16 prioritized interrupts. In the present configuration, 5 interrupts are used by external hardware. Each interrupt can be programmed for edge/level and polarity. A set of multiplexer allocated the different source of interrupt to these 5 interrupts. See Interrupt registers details in the paragraph 3.3.3.

3.3.2.8 DSP_LINTI_INTI 0x902F E840

When DSP writes at this location, an interrupt is send to the HOST through the line /LINTI.
The multiplexer must be set to “0”. It is set to “0” upon reset.

3.3.2.9 DSP_LINTI_RSTI 0x902F E844

When the interrupt /LINTI is asserted, either the DSP or the PCI must write to this register. PCI_LINTI_RST located at Base + 0x10. The interrupt is also cleared the board reset.

3.3.2.10 DSP_CYC_REV_ADR 0x902F E860

Unique 8 bits identifiers factory programmed within the Cyclone FPGA

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
0	0	0	0	0	0	0	1

3.3.2.10.1 *DSP_PCB_REV_ADR[] 0x902F E864*

Unique 8 bits identifiers factory programmed within the Cyclone FPGA

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
0	0	0	0	0	0	0	1

3.3.2.10.2 *DSP_PLX_REV_ADR[] 0x902F E868*

Unique 8 bits identifiers factory programmed within the Cyclone FPGA

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
1	0	1	0	0	1	0	1

3.3.2.10.3 *DSP_DSP_REV_ADR[] 0x902F E86C*

Unique 8 bits identifiers factory programmed within the Cyclone FPGA

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
1	0	1	0	1	0	1	0

3.3.3 Interrupt Registers

0x902F E820- 0x902F E85C

3.3.3.1 DEFAULTS INTERRUPT

External interrupt	Name source	Function
XINT[4]	/FF	FIFO FULL ADDA module (programmable)
XINT[5]	MATCH,COS	I/O module.
XINT[6]	/USB_INT, /INT85	USB, SCC8530
XINT[7]	/LINTO	HOST through PLX
NMI	None	

NOTE: THE INTERRUPT NEEDS TO BE PROGRAMMED FOR NEGATIVE EDGE DETECTION AT TMS320C6713B LEVEL

3.3.4 Interrupt to Host

The source for /LINTI is unique and selectable as EXT_INT[6..4]

SOURCE	CODE	Function
DSP_LINTI_INT	0000	DSP write generate an interrupt to pci

3.3.4.1 /RD_INT_STATUS

This register provides the status of the interrupt lines going to the 6713B.

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
/LSERR	/LINTO	/USERO	/USB_INT	XINT[7]	XINT[6]	XINT[5]	XINT[4]

ED15	ED14	ED13	ED12	ED11	ED10	ED9	ED8
X	X	X	X	AUX_2	AUX_1	AUX_0	/INT85

3.3.5 PLX 9056 Registers

Base address: 0x902FFE000

The TMS320C6713B can access the PLX configurations registers at the following addresses.

NAME	PCI AD	LOCAL	FUNCTION
PCIIDR	00h	00h	**PCI Configuration ID**
PCICR	04h	04h	**PCI Command**
PCISR	06h	06h	**PCI Status**
PCIREV	08h	08h	**PCI Revision ID**
PCICCR	09-0Bh	09-0Bh	**PCI Class Code**
PCICLSR	0Ch	0Ch	**PCI Cache Line Size**
PCILTR	0Dh	0Dh	**PCI Bus Latency Timer**
PCIHTR	0Eh	0Eh	**PCI Header Type**
PCIBISTR	0Fh	0Fh	**PCI Built-In Self-Test (BIST**
PCIBAR0	10h	10h	**PCI Base Address for Memory Accesses to Local, Runtime, DMA, and Messaging Queue Registers**
PCIBAR1	14h	14h	**PCI Base Address for I/O Accesses to Local, Runtime, DMA, and Messaging Queue Registers**
PCIBAR2	18h	18h	**PCI Base Addr. for Accesses to Local Address Space 0**
PCIBAR3	1Ch	1Ch	**PCI Base Addr. for Accesses to Local Address Space 1**
PCIBAR4	20h	20h	**PCI Base Address**
PCIBAR5	24h	24h	**PCI Base Address**
PCICIS	28h	28h	**PCI Cardbus Information Structure Pointer**
PCISVID	2Ch	2Ch	**PCI Subsystem Vendor ID**
PCISID	2Eh	2Eh	**PCI Subsystem ID**
PCIERBAR	30h	30h	**PCI Base Address for Local Expansion ROM**
CAP_PTR	34h	34h	**New Capability Pointer**
PCIILR	3Ch	3Ch	**PCI Interrupt Line**
PCIIPR	3Dh	3Dh	**PCI Interrupt Pin**
PCIMGR	3Eh	3Eh	**PCI Minimum Grant**
PCIMLR3	3Fh	3Fh	**PCI Maximum Latency.**
PMCAPID	40h	180h	**Power Management Capability ID .**
PMNEXT	41h	181h	**Power Management Next Capability Pointer**
PMC	42h	182h	**Power Management Capabilities .**
PMCSR	44h	184h	**Power Management Control/Status**
PMCSR_BSE	46h	186h	**PMCSR Bridge Support Extensions**
PMDATA	47h	187h	**Power Management Data**
HS_CNTL	48h	188h	**Hot Swap Control**
HS_NEXT	49h	189h	**Hot Swap Next Capability Pointer .**
HS_CSR	4Ah	18Ah	**Hot Swap Control/Status .**
PVPDID	4Ch	18Ch	**PCI Vital Product Identification .**
PVPD_NEXT	4Dh	18Dh	**PCI Vital Product Data Next Capability Pointer**
PVPDAD	4Eh	18Eh	**PCI Vital Product Data Address**
PVPDATA	50h	190h	**PCI VPD Data .**
LAS0RR	00h	80h	**Direct Slave Local Address Space 0 Range**
LAS0BA	04h	84h	**Direct Slave Local Address Space 0 Local Base Address (Remap**
MARBR	08h or ACh	88h or 12Ch	**Mode/DMA Arbitration**

NAME	PCI AD	LOCAL	FUNCTION
BIGEND	0Ch	8Ch	**Big/Little Endean Descriptor.**
LMISC1	0Dh	8Dh	**Local Miscellaneous Control**
PROT_AREA	0Eh	8Eh	**Serial EEPROM Write-Protected Address Boundary**
LMISC2	0Fh	8Fh	**Local Miscellaneous Control 2**
EROMRR	10h	90h	**Direct Slave Expansion ROM Range**
EROMBA	14h	94h	**Direct Slave Expansion ROM Local Base Address Re BREQo Control**
LBRD0	18h	98h	**Local Address Space 0/Expansion ROM Bus Region Descriptor**
DMRR	1Ch	9Ch	**Local Range for Direct Master-to-PCI**
DMLBAM	20h	A0h	**Local Base Address for Direct Master-to-PCI Memory**
DMLBAI	24h	A4h	**Local Base Address for Direct Master-to-PCI I/O Configuration**
DMPBAM	28h	A8h	**PCI Base Address (Remap for Direct Master-to-PCI Memory)**
DMCFGA	2Ch	Ach	**PCI Configuration Address for Direct Master-to-PCI I/O Configuration**
LAS1RR	F0h	170h	**Direct Slave Local Address Space 1 Range**
LAS1BA	F4h	174h	**Direct Slave Local Address Space 1 Local Base Address (Remap**
LBRD1	F8h	178h	**Local Address Space 1 Bus Region Descriptor**
DMDAC	FCh	17Ch	**Direct Master PCI Dual Address Cycles Upper Address**
PCIARB	100h	1A0h	**PCI Arbiter Control**
PABTADR	104h	1A4h	**PCI Abort Address**
MBOX0	40h or 78h	C0h	**Mailbox 0**
MBOX1	44h or 7Ch	C4h	**Mailbox 1**
MBOX2	48h	8Ch	**Mailbox 2**
MBOX3	4Ch	CCh	**Mailbox 3**
MBOX4	50h	D0h	**Mailbox 4**
MBOX5	54h	D4h	**Mailbox 5**
MBOX6	58h	D8h	**Mailbox 6**
MBOX7	5Ch	DCh	**Mailbox 7**
P2LDBELL	60h	E0h	**PCI-to-Local Doorbell**
L2PDBELL	64h	E4h	**Local-to-PCI Doorbell**
INTCSR	68h	E8h	**Interrupt Control/Status**
CNTRL	6Ch	ECh	**Serial EEPROM Control, PCI Command Codes,User I/O Control, and Init Control**
PCIHIDR	70h	F0h	**PCI Hardwired Configuration ID**
PCIHREV	74h	F4h	**sPCI Hardwired Revision ID**

NAME	PCI AD	LOCAL	FUNCTION
DMAMODE0	80h	100h	**DMA Channel 0 Mode**
DMAPADR0	84h	104h	**when DMAMODE0[20]=0**
Or	88h	108h	**when DMAMODE0[20]=1 DMA Channel 0 PCI Address**
DMALADR0	88h	108h	**when DMAMODE0[20]=0**
Or	8Ch	0Ch	**when DMAMODE0[20]=1 DMA Channel 0 Local Address**
DMASIZ0	8Ch	10Ch	**when DMAMODE0[20]=0**
Or	84h	4h	**when DMAMODE0[20]=1 DMA Channel 0 Transfer Siz Bytes**
DMADPR0	90h	110h	**DMA Channel 0 Descriptor Pointer**
DMAMODE	14h	114h	**DMA Channel 1 Mode**
DMAPADR1	98h	118h	**when DMAMODE1[20]=0**
Or	9Ch	11Ch	**when DMAMODE1[20]=1 DMA Channel 1 PCI Address**
DMALADR1	9Ch	11Ch	**when DMAMODE1[20]=0**
Or	A0h	120h	**when DMAMODE1[20]=1 DMA Channel 1 Local Address.**
DMASIZ1	A0h	120h	**when DMAMODE1[20]=0**
Or	98h	118h	**when DMAMODE1[20]=1 DMA Channel 1 Transfer Size Bytes**
DMADPR1	A4h	124h	**DMA Channel 1 Descriptor Pointer**
DMACSR0	A8h	128h	**DMA Channel 0 Command/Status**
DMACSR1	A9h	129h	**DMA Channel 1 Command/Status.**
DMAARB	Ach	12Ch	**DMA Arbitration**
DMATHR	B0h	130h	**DMA Threshold**
DMADAC0	B4h	134h	**DMA Channel 0 PCI Dual Address Cycles Upper Address**
DMADAC1	B8h	138h	**DMA Channel 1 PCI Dual Address Cycle Upper Address**
OPQIS	30h	B0h	**Outbound Post Queue Interrupt Status**
OPQIM	34h	B4h	**Outbound Post Queue Interrupt Mask.**
IQP	40h		**Inbound Queue Port.**
OQP	44h		**Outbound Queue Port**
MQCR	C0h	140h	**Messaging Queue Configuration**
QBAR	C4h	144h	**Queue Base Address**
IFHPR	C8h	148h	**Inbound Free Head Pointer**
IFTPR	CCh	14Ch	**Inbound Free Tail Pointer**
IPHPR	D0h	150h	**Inbound Post Head Pointer**
IPTPR	D4h	154h	**Inbound Post Tail Pointer**
OFHPR	D8h	158h	**Outbound Free Head Pointer**
OFTPR	DCh	15Ch	**Outbound Free Tail Pointer**
OPHPR	E0h	160h	**Outbound Post Head Pointer**
OPTPR	E4h	164h	**Outbound Post Tail Pointer**
QSR	E8h	168h	**Queue Status/Control**

Detailed information can be found into the PLX 9056 data sheet.

3.3.6 RS-232C Port

Base address: 0x902FFE400
Port A Control Register: 0x902FFE408
Port A Data Register: 0x902FFE40C

The board uses the port A of a SCC85C230 to provide a serial interface. It use a 7.372800 MHz local clock oscillator. The port B has no outside connection.
The interrupt from the serial controller is routed to the Cyclone to be shared with others interrupts source.

3.3.7 USB Port

Base address: 0x902C0000
The USB port use the CY7C68001 (SX20, high-speed USB interface from Cypress. It uses a 16 bit bus configuration. IFCLK is 45 MHZ.
The device will boot from its own EEPROM.
The base address of the CY7C68001 is 0x902FFEC00
The SX2 uses two interfaces to the TMS320C6713B.

- FIFO interface through witch EP2, 4, 6, 8 data flows.
- Command interface, witch is use to set up the SX2, read status, load descriptors, and access Endpoint 0.

Base Address	Selection
0x902C0000	**FIFO2**
0x902C4000	**FIFO4**
0x902C8000	**FIFO6**
0x902CC000	**FIFO8**
0x902D0000	**COMMAND**
0x902D4000	**Reserved**
0x902D8000	**Reserved**
0x902DC000	**Reserved**

The access by the TMS320C6713B to the FIFO is made asynchronously.
Flag status can be read in the TMS320C6713B_status register.
The USP LEDs are directly connected to the USB chip to provide visual display.
The interrupt from the USB controller is routed to the Cyclone to be share with other peripherals interrupt.

3.4 CE2 space

The CE2 space is dedicated to a 4Mbytes SRAM dual ported between the TMS320C6713B and the PLX 9056. Hardware arbitration is provided. It is recommended to used USERo signal from the PLX (host) and the 9056 lock bit on the TMS320C6713B to avoid bus lock contention.

3.5 CE3 space

The CE3 space allows access to the boot FLASH as a 16-bit space. This allows to for instance running a program stored into it, or in general to access the data continuously. By contrast when accessing the same FLASH at the address 0x90000000, only 2 bytes are valid in each 4 bytes.

4 PCI-Slave Address Map

4.1 PLX Address Map

4.2 Local Resources

There are 4 base address regions available on the PLX PCI9056. The **PCI-6713MFC** uses all 4 regions.

- BAR0: Memory access the PLX operation registers.
- BAR1: I/O access the PLX operation registers.
- BAR2: Pass-thru for the dual ported ram. The pass-thru is established at PCI reset when the DSP boots and enables it.
- BAR3: Pass-thru for the HPI (offset 0x20000) and cyclone (offset 0x00000) accesses

RESOURCE	REGION	SPACE	ADDRESS RANGE
PLX registers	BAR0	AS0	0x000-0x1ff
Dual-Ported SRAM	BAR2	AS2	0x000000-0x3fffff
Cyclone registers	BAR3	AS15	0x00000
DSP HPI registers	BAR3	AS15	0x20000-0x2000f

4.2.1 Cyclone Registers

The HOST can access several registers located inside the CYCLONE FPGA.
These registers are used mainly to monitor the behavior of the board or to control and/or generate interrupt to the DSP. Details are provided above.

4.2.1.1 PCI_CNTL0 Base + 0x0

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
DSPNMI	USERI_O	HINTEN	0	0		DSPRST	0

DSPNMI A “1” generates an NMI interrupt to the TMS320C6713B. NMIEN bit must be selected.

HINTEN Host interrupt enable. This bit, when set to “1”, allows the LINT1 signal to generate an interrupt on the PCI. The default is “disabled”

DSPRST When set to “1” by PCI will maintain the TMS320C6713B on reset mode. It needs to be set back to “0” to remove the DSP reset.

4.2.1.2 PCI_STAT0 Base + 0x4

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
0	N/A_	0	/LINTI	XCNTL1	XCNTL0	/LED_1_	/LED_0_

/LINTI Status of the interrupt request line LINTI to the PCI bus. When this bit is "0", an interrupt is requested. HINTEN must be enabled for this line to actually generate an interrupt on the PCI bus.

DSP_HINT Status of TMS320C6713B generated HPI interrupt

XCNTL1, XCNTL0 Status of two controls lines activated by DSP for extended Xbus use. At reset, it defaults to "0".

/LED_0 Status of the LEDs controlled by the DSP. At reset, the LEDS are "off". A "1" will turn the corresponding LED "on".

/LED_1 Status of the LEDs controlled by the DSP. At reset, the LEDS are "off". A "1" will turn the corresponding LED "on".

4.2.1.3 PCI_CNTL1 Base + 0x8

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
P_CNT7	P_CNT6	P_CNT5	P_CNT4	P_CNT3	P_CNT2	P_CNT1	P_CNT0

The value written by the host in the PCI_CNTL1 register can be read by the DSP in the DSP_STAT2 register.

4.2.1.4 PCI_STAT1 Base + 0xC

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
D_DSP7	D_DSP6	D_DSP5	D_DSP4	D_DSP3	D_DSP2	D_DSP1	D_DSP0

The value written in the *DSP_CNTL2* register by the DSP can be read by the host in this register.

4.2.1.5 PCI_LINTI_RST Base + 0x10

A write at this address by the host will de-assert the line */LINTI* that generates an interrupt on the PCI bus.

4.2.1.6 PCI_DSP_INT Base + 0x14

Not used

4.2.2 DSP HPI Registers

The HOST through the PLX 9056 can access to all the resource of the board through the Host Port Interface bus or H.P.I. of the DSP. It is a 16 bit bus with only four (4) registers locations. As the TMS320C6713B is a 32bit machine, every access consists of two consecutive 16 bit half words.
The two DSP signals *HCNTL1* and *HCNTL0* that select the type of access are automatically generated by the board logic from the address used.

BAR3 offset	HCNTL1	HCNTL0	Description
0x20000	0	0	Host reads from or writes to the HPI **control** register (HPIC).
0x20004	0	1	Host reads from or writes to the HPI **address** register (HPIA).
0x20008	1	0	Host reads from or writes to the HPI **data** register (HPID) in *auto increment* mode. The HPI address register (HPIA) is post incremented by a word address (four bytes addresses)
0x2000C	1	1	Host reads from or writes to the HPI **data** register (HPID) in *fixed* address mode. The HPI address register (HPIA) is not affected.

```
typedef struct hpi_t {
        long ctrl;
        long addr;
        long data;
        long autoinc;
} hpi_t;
hpi_t *hpi;

ULONG readHPI(ULONG addr)
{
        int i = 10000;

        hpi->addr = addr;
        hpi->ctrl = 0x00110011;
        while ((hpi->ctrl & 8)&& (i--)) ;               // wait for ready
        if (i == 0) printf("\nTimeout accessing the HPI and waiting for ready\n");
        return hpi->data;
}

void writeHPI(ULONG addr, ULONG data)
{
        int i = 10000;

        hpi->addr = addr & 0xfffffffc;                  // force a 32 bit boundary
        hpi->data = data;
        hpi->ctrl = 0x00110011;
        while ((hpi->ctrl & 8)&& (i--)) ;               // wait for ready
        if (i == 0) printf("\nTimeout accessing the HPI and waiting for ready\n");
}
```

4.2.3 Dual-Ported SRAM

The CE2 space is dedicated to a 4Mbytes SRAM dual ported between the TMS320C6713B and the PLX 9056. Hardware arbitration is provided

5 PARALLEL INPUT/OUTPUT MODULE

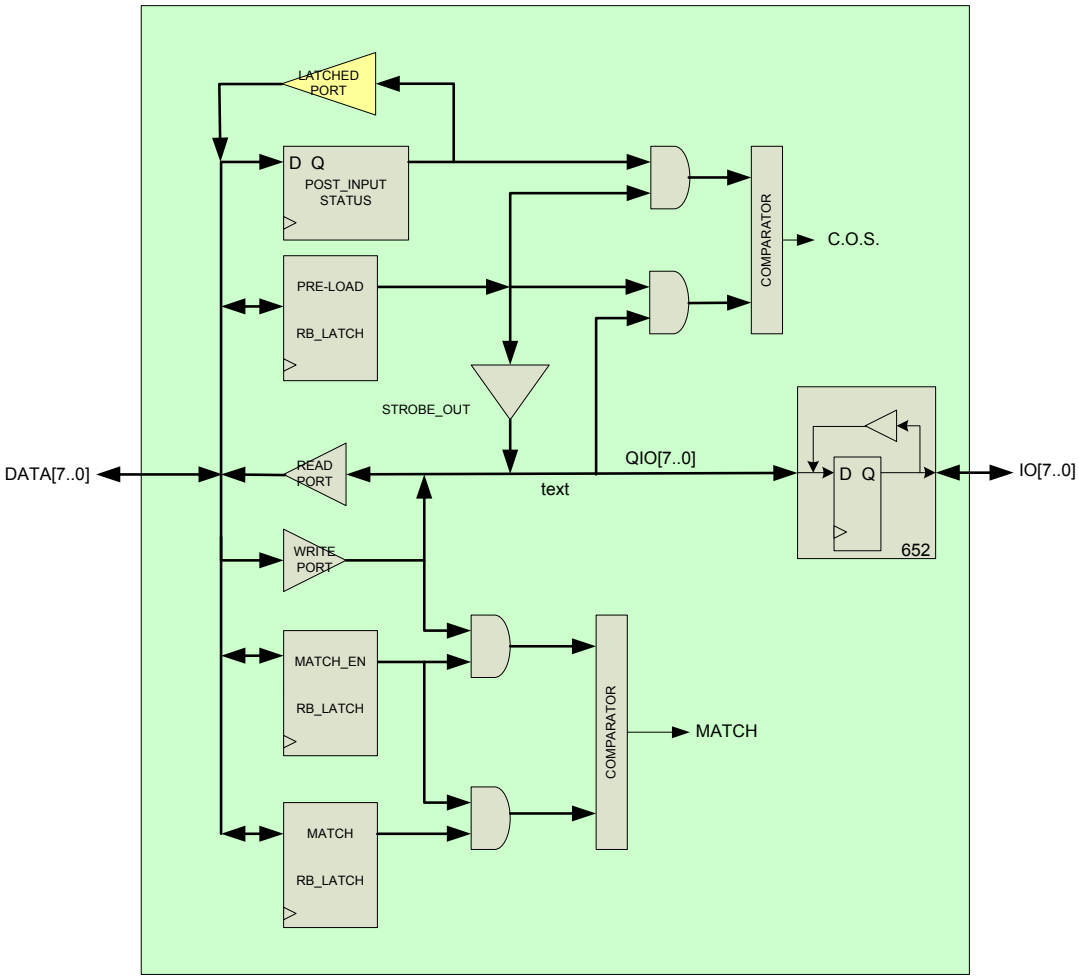
5.1 Functional description

The IO module logic is embedded within an ALTERA “CYCLONE” Embedded Programmable Logic Family.
The addresses of the I/O module are between 0x902F F000 and 0x902F F3FF.

Key Features are:

- 48 I/O Pin selectable as input or output by group of 8
- TTL signals
- Direct read-back outputs
- Pre-load output register with strobe signal use as sync
- Change of state detection since last Input read with interrupts generation
- Match input detection with pre-load match word (by group of 8) with interrupt generation
- 100 MHz CLK
- Interrupts on Change of State
- Custom functions possible using ALTERA tools

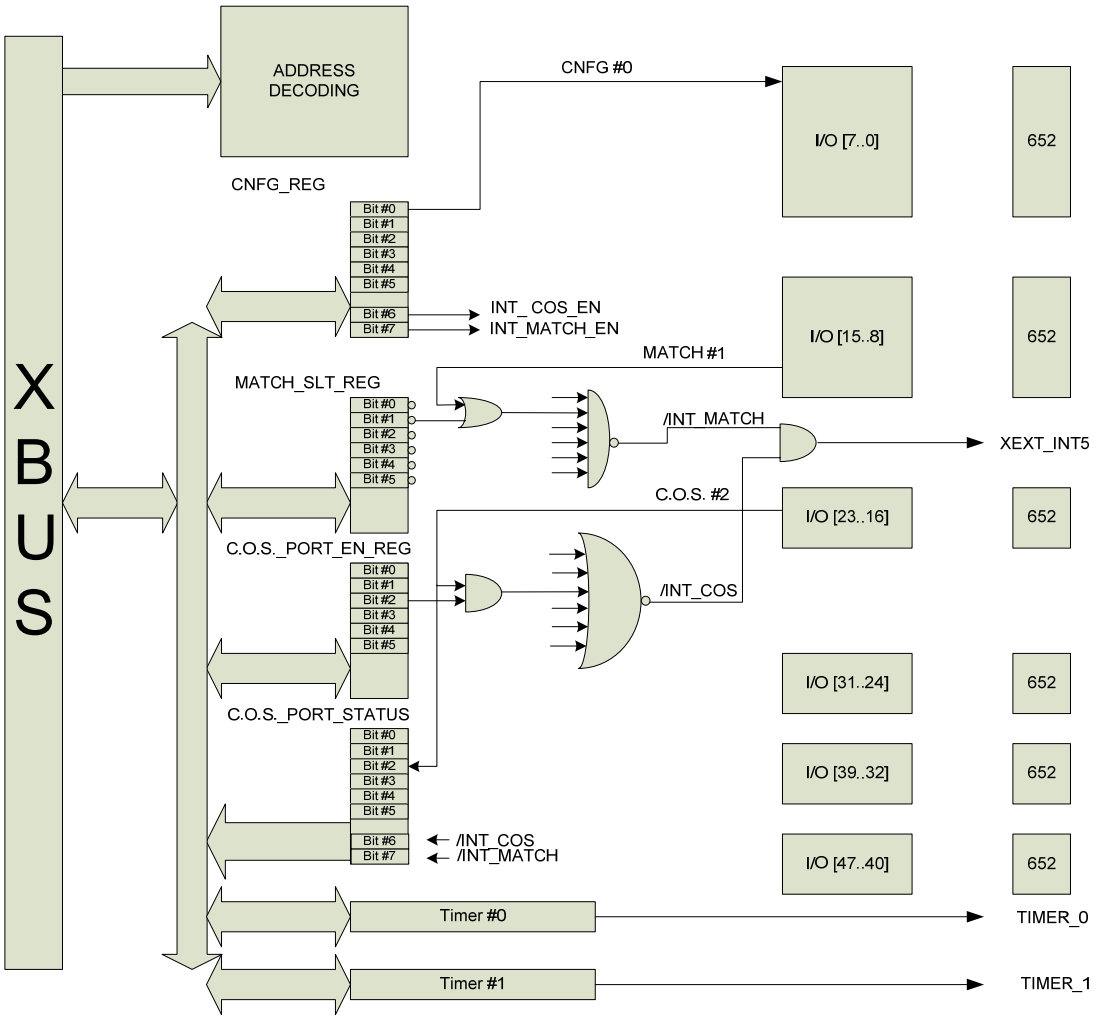
Functional description port [5..0] :



5.2 Address Map

		INPUT PORT	OUTPUT PORT		
$00	R	C.O.S. STATUS		4..0	
$04	RW	INPUT/OUTPUT PORT CONFIGURATION		4..0 4..0	
$08	R	INPUT PORT STATUS	OUTPUT PORT DATA LATCHED BY 652	31..0	IO[31..0]
$08	W	NO EFFECT	NEW DATA UPDATE LATCHED BY 652	31..0	
$0C	R	INPUT PORT STATUS	OUTPUT PORT DATA LATCHED BY 652	15..0	IO[47..32]
$0C	W	NO EFFECT	NEW DATA UPDATE LATCHED BY 652	15..0	
$10	R	READ-BACK	PRE-LOAD OUTPUT DATA THAT WILL BE LATCHED BY 652 AND TRANSFERED UPON STROBE	31..0	IO[31..0]
$10	W	BIT SELECTION OF EACH PORT [3..0] FOR C.O.S. DETECTION		31..0	
$14	R	READ-BACK	PRE-LOAD OUTPUT DATA THAT WILL BE LATCHED BY 652 AND TRANSFERED UPON STROBE	15..0	IO[47..32]
$14	W	BIT SELECTION OF EACH PORT [5..4] FOR C.O.S. DETECTION		15..0	
$18	R	READ-BACK	NO EFFECT	31..0	IO[31..0]
$18	W	BIT VALUE EACH PORT [3..0] FOR MATCH		31..0	
$1C	R	READ-BACK	NO EFFECT	15..0	IO[47..32]
$1C	W	BIT VALUE EACH PORT [5..4] FOR MATCH		15..0	
$20	R	READ-BACK	NO EFFECT	31..0	IO[31..0]
$20	W	BIT MASK ENABLE FOR MATCH_BIT		31..0	
$24	R	READ-BACK	NO EFFECT	15..0	IO[47..32]
$24	W	BIT MASK ENABLE FOR MATCH_BIT ENABLE		15..0	
$28	R	READ_BACK.	NO EFFECT	4..0	
$28	W	PORT [5..0] SELECTION FOR INTERRUPT GENERATED BY C.O.S.		4..0	
$2C	R	READ_BACK.	NO EFFECT	15..0	IO[47..32]
$2C	W	PORT [5..0] SELECTION FOR INTERRUPT GENERATED BY MATCH		15..0	
$30	W	NOT USED	STROBE PULSE SEND PRE-LOADED DATA TO IO SELECTED AS OUTPUT IN A SYNCRO MODE		IO[47..0]
$34	R	READ-BACK OF THE LATCHED INPUT DATA USE FOR C.O.S. COMPARATOR	STROBE PULSE SEND PRE-LOADED DATA TO IO SELECTED AS OUTPUT IN A SYNCRO MODE		IO[47..0]
$38	R	READ-BACK OF THE LATCHED INPUT DATA USE FOR C.O.S. COMPARATOR	STROBE PULSE SEND PRE-LOADED DATA TO IO SELECTED AS OUTPUT IN A SYNCRO MODE		IO[47..0]
$40	R	TIMER #0 READ_BACK.	NO EFFECT		31..0
$40	W	TIMER #0 COUNTER			
$42	R	TIMER #0 READ_BACK.	NO EFFECT		47..32
$42	W	TIMER #0 COUNTER			

XBUS


X
B
U
S
ADDRESS
DECODING
CNFG #0
I/O [7..0]
652
CNFG_REG
Bit #0
Bit #1
Bit #2
Bit #3
Bit #4
Bit #5
Bit #6
Bit #7
INT_ COS_EN
INT_MATCH_EN
I/O [15..8]
652
MATCH #1
MATCH_SLT_REG
/INT_MATCH
XEXT_INT5
C.O.S. #2
I/O [23..16]
652
C.O.S._PORT_EN_REG
/INT_COS
I/O [31..24]
652
C.O.S._PORT_STATUS
I/O [39..32]
652
/INT_COS
/INT_MATCH
I/O [47..40]
652
Timer #0
TIMER_0
Timer #1
TIMER_1

5.2.1 REG_COS (Read Only)

Address: 0x902F F000

```
volatile char *Pio_Reg_Conf = (char *)0x902ff004;
```

This register signals if a change of state (C.O.S.) change has occurred since the last read of the corresponding input ports has been made. Each bit corresponds to a group of 8 inputs. A "1" means a C.O.S. has occurred.

BD07	BD06	BD05	BD04	BD03	BD02	BD01	BD00
/INT_ MATCH	/INT_ COS	COS_ PORT #5	COS_ PORT #4	COS_ PORT #3	COS_ PORT #2	COS_ PORT #1	COS_ PORT #0

/INT_COS: Status of interrupt line from an enabled COS event.
/INT_MATCH: Status of interrupt line from an enabled MATCH event.

5.2.2 REG_CONF (Read and Write)

Address: 0x902F F004

This register is used to program each group of ports as Input or Output.
Upon reset all the ports are set as input.
A "1" in the corresponding bit select a group of 8 to be an output.

BD07	BD06	BD05	BD04	BD03	BD02	BD01	BD00
0	0	PORT #5	PORT #4	PORT #3	PORT #2	PORT #1	PORT #0

5.2.3 DIGITAL I/O PORTS (Read and Write)

There are 48 bits of digital I/O lines. They are configurable as output or input by group of 8, in a 6-byte array. The REG_CONF register specifies if a given 8-bit group is an input or an output.

The base address of the array is: 0x902F F008. From a hardware point of view, the registers are accessible on a 32-bit bus, and it is somewhat more efficient to access them several at a time.

```
volatile char *Pio_Port0 = (char *)0x902ff008;
volatile char *Pio_Port1 = (char *)0x902ff009;
volatile char *Pio_Port2 = (char *)0x902ff00a;
volatile char *Pio_Port3 = (char *)0x902ff00b;
volatile char *Pio_Port4 = (char *)0x902ff00c;
volatile char *Pio_Port5 = (char *)0x902ff00d;
```

5.2.3.1 Group selected as OUTPUT

5.2.3.1.1 *READ*

If the corresponding port is selected as an output, a read to the location will return the status of the data previously latched.

5.2.3.1.2 *WRITE*

A write to this location will latch the data into the register and the outputs will be updated.

5.2.3.2 Group selected as INPUT

5.2.3.2.1 *READ*

A read to the location will provide real time information from the port.

5.2.3.2.2 *WRITE*

A write to the location will have no effect.

5.2.4 PIO_LOAD_OUT / PIO_COS_MASK (Read and Write)

Address: 0x902F F010 to 0x902F F015

```
// when the I/O is used as an input port
volatile char *Pio_CosMask0 = (char *)0x902ff010;
volatile char *Pio_CosMask1 = (char *)0x902ff011;
volatile char *Pio_CosMask2 = (char *)0x902ff012;
volatile char *Pio_CosMask3 = (char *)0x902ff013;
volatile char *Pio_CosMask4 = (char *)0x902ff014;
volatile char *Pio_CosMask5 = (char *)0x902ff014;

// when the I/O is used as an output port
volatile char *Pio_LoadOut0 = (char *)0x902ff010;
volatile char *Pio_LoadOut1 = (char *)0x902ff011;
volatile char *Pio_LoadOut2 = (char *)0x902ff012;
volatile char *Pio_LoadOut3 = (char *)0x902ff013;
volatile char *Pio_LoadOut4 = (char *)0x902ff014;
volatile char *Pio_LoadOut5 = (char *)0x902ff014;
```

5.2.4.1 Input

If the corresponding register is set as an INPUT, the PR_COS_MASK register is used as a Bit-Mask to enable a state of change of the matching input.

5.2.4.2 Output

When the port is used as an OUTPUT, each register PR_LOAD_OUT [x] can be pre-loaded separately. When a write occurs in the PIO_STROBE register, all the pre-loaded values are transferred simultaneously do the output registers.

5.2.5 PORTEN ((Read and Write))

Address: 0x902F F028

As with the previous register each bit has a different function depending if the PORT is defined as Input or Output.

BD07	BD06	BD05	BD04	BD03	BD02	BD01	BD00
COSR_EN		PACT #5	PACT #4	PACT #3	PACT #2	PACT #1	PACT #0

BD07: COSR_EN
A “0” will disable all the C.O.S. logic.

5.2.5.1 Input

If COSR_EN is “1”, a “1” in one in a (PACT) bit enables the C.O.S. for the corresponding 8-bit port.
Upon reset this register is set to “0”.

5.2.5.2 Output

When an 8-bit PORT is defined as an output port, a “0” in the corresponding (PACT) bit forces all the outputs to “0”, independently of what was written in the corresponding port register.
Upon reset this register is set to “0”.

5.2.6 PIO_STROBE (Write only)

Address: 0x902F F02C

A write to this location will transfer the content of the PRLDOUT [0..5] Registers to the corresponding outputs (if programmed as output).

5.2.7 RB_LATCH [5..0] (Read only)

Address: 0x902F F034

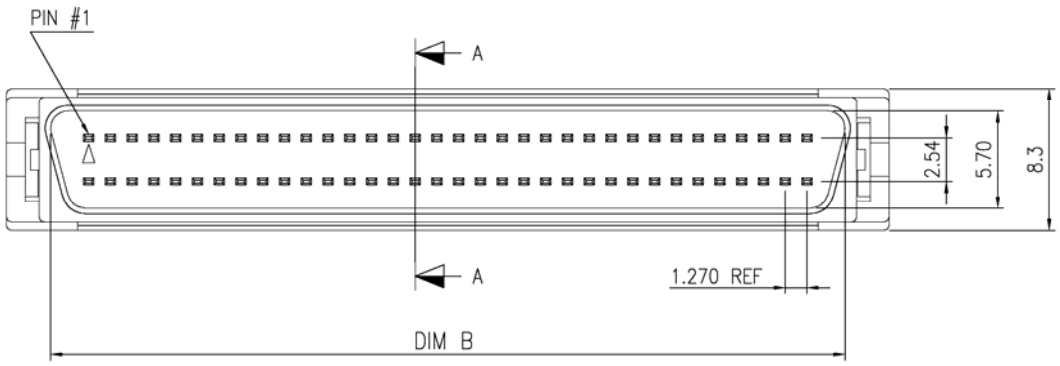
This register latches the input data when a read is made to the input port. The latched data are then used as reference for a change of state (C.O.S.) that can occurs since the last read of the inputs ports has been made. Each byte corresponds to each group selected as input.

5.2.8 TIMER1, TIMER2 Internal use only

Address: 0x902F F040, 0x902F F040

These two 32-bit timers are used for internal testing only. The values can only be written in 16-bit accesses, so that programming any of them requires 2 accesses.

5.3 I/O Port



PIN	I/O	Cyclone line	PIN	I/O	Cyclone line
Pin 1	I/O 00	U9	Pin 2	I/O 01	R9
Pin 3	I/O 02	V9	Pin 4	I/O 03	T9
Pin 5	I/O 04	R7	Pin 6	I/O 05	V8
Pin 7	I/O 06	T8	Pin 8	I/O 07	R8
Pin 9	I/O 08	*C12*	Pin 10	I/O 09	*D13*
Pin 11	I/O 10	*B12*	Pin 12	I/O 11	*A13*
Pin 13	I/O 12	*B13*	Pin 14	I/O 13	*C13*
Pin 15	I/O 14	*A12*	Pin 16	I/O 15	*D12*
Pin 17	I/O 16	E7	Pin 18	I/O 17	C7
Pin 19	I/O 18	B9	Pin 20	I/O 19	E8
Pin 21	I/O 20	B8	Pin 22	I/O 21	Dir16
Pin 23	I/O 22	D8	Pin 24	I/O 23	C8
Pin 25	I/O 24	*A7*	Pin 26	I/O 25	*D6*
Pin 27	I/O 26	*C6*	Pin 28	I/O 27	*E6*
Pin 29	I/O 28	*B6*	Pin 30	I/O 29	*B5*
Pin 31	I/O 30	*D7*	Pin 32	I/O 31	*B7*
Pin 33	I/O 32	D10	Pin 34	I/O 33	C9
Pin 35	I/O 34	D9	Pin 36	I/O 35	E10
Pin 37	I/O 36	A10	Pin 38	I/O 37	A9
Pin 39	I/O 38	C10	Pin 40	I/O 39	B10
Pin 41	I/O 40	*H14*	Pin 42	I/O 41	*H13*
Pin 43	I/O 42	*H17*	Pin 44	I/O 43	*H15*
Pin 45	I/O 44	*J13*	Pin 46	I/O 45	*H18*
Pin 47	I/O 46	*H16*	Pin 48	I/O 47	*J14*
Pin 49	GND		Pin 50	GND	

Table 1 I/O connector

6 MAIN BOARD CONNECTORS

6.1 Ps mode

Pin	Signal name	Description
1	DCLK	Clock Signal
2	GND	Ground
3	CONF_DONE	Configuration done
4	VCC	Power supply
5	nCONFIG	Configuration control
6	NC	No connect
7	nSTATUS	Configuration status
8	NC	No connect
9	DATA0	Data to Device
10	GND	Ground

6.2 Cyclone

6.2.1 USB Byteblaster

The two Cyclones can be programmed using the ByteBlaster.
Connector P1 is used for the I/O CYCLONE.
Connector P2 is used for the DSP CYCLONE.

J2 Pin	Signal
Pin 1	DCLK
Pin2	GND
Pin3	DONE
Pin4	+3.3V
Pin5	nCONFIG
Pin6	nCE
Pin7	Data0
Pin8	IO nCSO
Pin9	IO ASDO
Pin10	GND

Table 6-2 ByteBlaster pod

The ByteBlaster programs the serial EEPROM EPCS1 or EPCS4, then the Programmed data are transferred to the CYCLONE using Asynchronous Serial Mode.

7 AD/DA DAUGHTER BOARD

Base address: 0x9020 0000 - 0x902F 7FFF
The daughter board main features are:

A/D

- Sixteen channels of simultaneous 16 bit A/D acquisition, operating at a maximum rate of 100 KHz (option 250 KHz, 500 KHz and1MHz).
- Inputs level : +/- 10V ,+/-5V, +/-2.5V, 0-10V, 0-5V, 0-2.5V
- Differential mode.
- Input impedance: 1MΩ
- Up to 64K of A/D samples stored in on-board FIFO memory.
- Sampling clock selected from one of the following sources:
- Internal divider
- External clock.
- Internal timer
- Trigger event selected from one of the following sources
- External trigger.
- Continuous streaming acquisition is also possible at lower throughput.

D/A

- Sixteen channels of D/A converter with a 2 µS settling time to 1 LSB.
- The D/A converters are multiplying converters with +/-10 V output using the +10V internal Reference or an external signal reference.

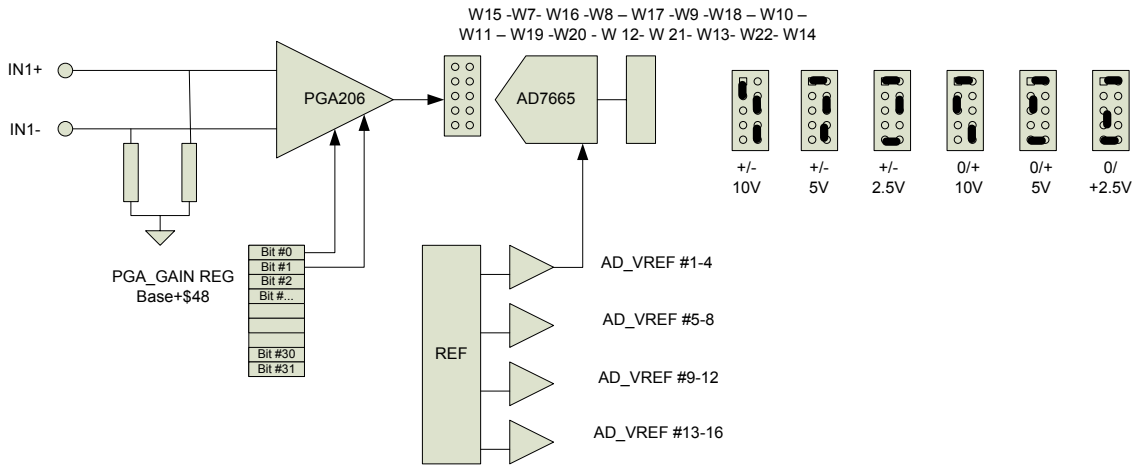
EEPROM
A 1-Mbit EEPROM store calibration parameters starting at address 0x9030 0000.

7.1.1 Analog Inputs

Sixteen differential analog inputs are provided. By default, the inputs are high impedance +/- 10 volt. A set of jumpers allows other inputs configurations

7.1.1.1 A/D Converters

There are Sixteen A/D converters. The A/D converters operate continuously at the selected sampling rate. Results are either stored in the FIFO or directly available.



7.1.2 Analog Outputs

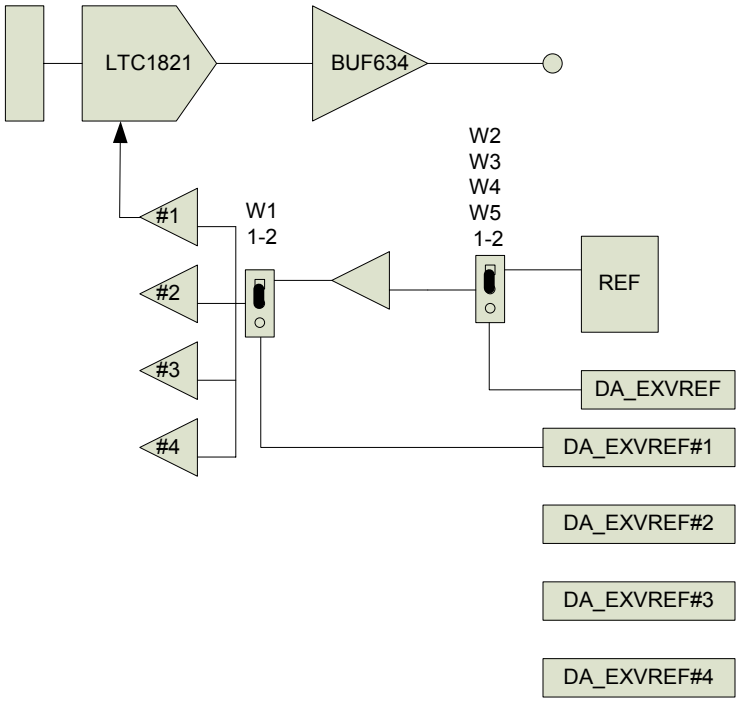
The module has 16 D/A converters with full parallel 16-bit multiplying voltage.
The device operates with +/- 15V provided by a DC/DC converter or from an external source on the front panel connector.
The voltage reference for the D/A converter is provided by either:

- A very high precision reference voltage providing +10V. The reference can be trimmed to achieve better performance.
- An external input signal.

A jumper W1 selects the input global reference. This global reference signal is then buffered. A set of jumper selects by block of 4 D/A either the global reference or an independent reference. Output of the D/A is buffered to provide high current capabilities with +/- 10Volts voltage output. The conversion settles to 1 LSB in 2µS.
The asynchronous /CLR signal resets the LTC1821 to zero scale and resets the LTC1821-1 to mid-scale.

W 1 jumper between	Global Reference
1-2	+10V internal reference
2-3	External signal DA_EXVREF

W 2 W3 W4 W5	Reference
1-2	Global reference signal
2-3	External signal DA_EXREF[1..4]



7.1.3 Daughter Board Registers

The registers are accessed in 32-bit mode. It is possible to do 32-bit accesses to the A/D and D/A registers. This allows writing or reading 2 values in one access. Depending on the program structure, this can provide some optimization by decreasing the number of relatively slow outside accesses.

Address	Name	Type	Size	Name
0x9020 0000	SAMPL0	R/W	15-0	Internal Clock Divisor
0x9020 0004	WR_MASTER_FIFO_RST	W	7-0	A write to this address resets the FIFO and pointer programming
0x9020 0008	WR_PARTIAL_FIFO_RST	W	7-0	A write to this address resets the FIFO READ pointer leaving the PAE and PAF unchanged
0x9020 0014	LED	R/W	7..4	User LEDs
0x9020 0018	STOP_ACQ	W	N/A	A write to this address stops an ongoing acquisition and storage to the FIFO
0x9020 001C	ACQCSR	R/W	7-0	Acquisition Control Register. Controls source sampling clock, source signal end acquisition
0x9020 0020	FIFO_IRQ_EN	R/W	7-0	Source selection for Interrupt XINT4
0x9020 0028		R/W	7-0	Trigger selection
0x9020 002C		R/W	7-0	FIFO Control / Status Register
0x9020 0030		WS	N/A	Control bit #0 for Reset FIFO
0x9020 0034		R	7-0	FIFO pointers Status
0x9020 0038	SINGLE_ACQ	W		A write to this address initiates a single A/D acquisition
0x9020 003C				Not used. Reads internal local FIFO.
0x9020 0040				A/D mode selection and FIFO data pointer selection
0x9020 0048	PGA GAIN			PGA A/D gain selection
0x9020 004C		W	N/A	Start Acquisition
0x9020 0054	DA_EXT	R/W	7-0	D/A external selection trigger type
0x9020 0060	AD_EXT	R/W	7-0	A/D external selection trigger type
0x9020 0070	FIFO_DATA	R	31..0	FIFO data (Read only)
0x9020 0080 – 0x9020 009F	D/A DATA REGISTERS	W	15..0	D/A 0-15
0x9020 00A0 – 0x9020 00BF	A/D DATA REGISTERS	R	31..0	A/D 0-15

Table 7.1 A/D Registers

Address	Name	Type	Size	Name
0x9020 0080	D/A DATA REGISTERS	W	15..0	D/A # 0
0x9020 0082		W	31..16	D/A # 1
0x9020 0084		W	15..0	D/A # 2
0x9020 0086		W	31..16	D/A # 3
0x9020 0088		W	15..0	D/A # 4
0x9020 008A		W	31..16	D/A # 5
0x9020 008C		W	15..0	D/A # 6
0x9020 008E		W	31..16	D/A # 7
0x9020 0090		W	15..0	D/A # 8
0x9020 0092		W	31..16	D/A # 9
0x9020 0094		W	15..0	D/A # 10
0x9020 0096		W	31..16	D/A # 11
0x9020 0098		W	15..0	D/A # 12
0x9020 009A		W	31..16	D/A # 13
0x9020 009C		W	15..0	D/A # 14
0x9020 009E		W	31..16	D/A # 15
0x9020 00A0	A/D DATA REGISTERS	R	15..0	A/D # 0
0x9020 00A2		R	31..16	A/D # 1
0x9020 00A4		R	15..0	AD/ # 2
0x9020 00A6		R	31..16	A/D # 3
0x9020 00A8		R	15..0	A/D # 4
0x9020 00AA		R	31..16	A/D # 5
0x9020 00AC		R	15..0	A/D # 6
0x9020 00AE		R	31..16	A/D # 7
0x9020 00B0		R	15..0	A/D # 8
0x9020 00B2		R	31..16	A/D # 9
0x9020 00B4		R	15..0	A/D # 10
0x9020 00B6		R	31..16	A/D # 11
0x9020 00B8		R	15..0	A/D # 12
0x9020 00BA		R	31..16	A/D # 13
0x9020 00BC		R	15..0	A/D # 14
0x9020 00BE		R	31..16	A/D # 15

Table 7.2 Data Registers

7.1.4 Internal Clock Divisor 0x9020 0000

A 16 bit register serves as a divisor of the EMIF clock ECLKOUT running at 100 MHZ.

$$SamplingRate = \frac{EMIFClockFreq}{N+1}$$

7.1.5 FIFO reset 0x9020 0004 -0x9020 0008

- Writing at the address 0x9020 0004 generates a Master Reset of the FIFO.
- Writing at the address 0x9020 0008 generates a Partial Reset of the FIFO.

The 6713B MFIO module has a 32 bit x 65,536 locations FIFO to store incoming data from the A/D converters.
Two types of reset are possible.
- Master reset. Reset the read/write pointer and output data pointer to zero. Reset the PAE and PAF pointer to defaults offset setting. Upon reset LD line is high.
    PAE is set to 03FF from beginning of the FIFO
    PAF is set to 03FF from End of the FIFO.
- Partial reset. Reset the read/write pointer and output data pointer to zero. Maintain the PAE and PAF programmed setting.

Upon a power-on Reset , the LD line is high which would allows programming the pointer through serial mode that we don't support, a reset sequence is needed to set LD low using bit # 3 of the register at address 0x9020 0040, activate the Master reset then put back LD high. Parallel programming can then be made. FIFO pointers are accessible in a circular mode. The first write will control the PAE pointer. The second write will control the PAF pointer. Same thing applies to a read of the pointer. Read and Write are independent. The line LD will automatically be set to low during the programming. More information can be found on the IDT data sheet IDT72V285.

7.1.6 LEDs 0x9020 0014

Four LEDS of the A/D daughter board are available to the user.

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
/Led 4	/Led 3	/Led 2	/Led1	Not used	Not used	Not used	Not used

7.1.7 Stop acquisition 0x9020 0018

A write at this location will stop storage of data inside the FIFO.

7.1.8 Acquisition control register 0x9020 001C

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
ClockSource1	ClockSource0			StopSource1	StopSource0		

Stop Acquisition Source determines the cause of ending the acquisition. The register is cleared upon reset. FF pointer is the default setting

StopSource1	StopSource0	Signal
0	0	FIFO FF Flag
0	1	FIFO PAF Flag
1	0	FIFO PAE Flag
1	1	FIFO HF Flag

Sampling Clock Source determines where the A/D converters get their start conversion source from.

ClockSource1	ClockSource0	Signal
0	0	DSP write into the Software Acquisition Start register (0x9020 0038)
0	1	DSP timer # 0
1	0	DSP timer # 1
1	1	Local timer

7.1.9 Source XINT4 0x9020 0020

This register allows enabling the signal(s) to generate the interrupt XINT4. It is possible to enable several of the signals, although it does not always make sense.

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
Not Used	Not Used	Not Used	FIFO EF	FIFO PAE	FIFO HF	FIFO PAF	FIFO FF

FIFO FF
This bit, when low (0), indicates that there are 65536 samples stored in the hardware FIFO. It is usually an indication that the acquisition is completed.

FIFO PAF
This bit, when low (0), indicates that there is a programmable number of samples stored in the hardware FIFO.

FIFO HF
This bit, when low (0), indicates that there is a 65536/2 samples number of samples stored in the hardware FIFO.

FIFO PAE
This bit, when low (0), indicates that there is less than a programmable number of samples stored in the hardware FIFO. In continuous modes, this bit being high indicates that there are N samples ready to be read, where N-1 is the number programmed in the FIFO.

FIFO EF
This bit, when low (0), indicates that there are no samples stored in the hardware FIFO.

7.1.10 Trigger selection 0x9020 0028

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
							Conversion source

When set to "1" Start conversion of the A/D converter is set by a raising pulse (>=40nS) from the *External Event* input signal.
When set to "0", the Start conversion is initiated by a write in the *Software acquisition start Register* (0x9020 0038).
Upon reset, this bit is cleared.

7.1.11 FIFO Status (Read) 0x9020 0034

This register allows for querying the current state of the hardware FIFO flags and a means to program the FIFO.

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
FIFO FF	FIFO PAF	FIFO PAE	FIFO EF	HF	0	Busy	FIFO REG

FIFO REG
When the bit is set to "1", writes and reads of the FIFO will access the FIFO programming pointer registers PAE and PAF in a circular fashion

Busy
"0" means that the conversion has finished. "1" means that all the A/D are done.
This bit represents the status of the global A/D busy signal. When the A/Ds have all finish converting, the signal goes low. It is used when using an A/D direct conversion as in example # 1.

7.1.12 Software acquisition start 0x9020 0038

A write to this location will start an immediate unique acquisition. The *Acquisition Control Register* bits 6 and 7 need to be set to "0".

7.1.13 Direct A/D data read 0x9020 00A0 - 0x9020 00BC

After a single acquisition data from each A/D converter can be read directly from the A/D converter. 0x9020 00AO is for the channel #1 and channel # 2, etc…

7.1.14 A/D mode and FIFO control register 0x9020 0040

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
OB_/2C	Warp Mode	impulse	Byte Swap	FIFO_REG	ADIRECT	Partial FIFO reset	Master FIFO reset

Note: Default setting upon reset: 0x38.

ByteSwap
When this bit is default low. The LSB is output on D[7..0] and the MSB is output on D[15..8].When set high, the LSB is output on D[15..8] and the MSB is output on D[7..0].

IMPULSE
When set to “1” and WARP to “0”, this bit selects a reduced power mode. In that mode the power dissipation is approximately proportional to the sampling.

Warp Mode
When set to “1” and IMPULSE to “0”, this selects the fastest mode, the maximum throughput is achievable, and a minimum conversion rate must be applied in order to guarantee full specified accuracy. When “0”, full accuracy is maintained independently of the minimum conversion rate.

OB/2C
When the OB/2C bit is low, the A/D output is straight binary
The OB-2C bit reflects the inverted position of the OB_2C signal connected to the A/D converter hardware. The on-board logic inverts the bit #7 in such a way that upon reset the A/D are set for the Straight Binary mode

ADIRECT
Upon reset this bit is cleared. Direct data access from the A/D conversion result can be made in the A/D area starting at address 0x902000A0.

7.1.15 PGA Gain 0x9020 0048

This register is used to program each Programmable Gain Amplifier associated with each A/D converter. A set of two bit by PGA gives a possible gain of 1, 2, 4 or 8.
This register reset to 0 upon power on reset, giving a gain of 1 for all the channels.

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
A/D#3		A/D#2		A/D#1		A/D#0	
bit1	bit0	bit1	bit0	bit1	bit0	bit1	bit0

ED15	ED14	ED13	ED12	ED11	ED10	ED9	ED8
A/D#7		A/D#6		A/D#5		A/D#4	
bit1	bit0	bit1	bit0	bit1	bit0	bit1	bit0

ED23	ED22	ED21	ED20	ED19	ED18	ED17	ED16
A/D#11		A/D#10		A/D#9		A/D#8	
bit1	bit0	bit1	bit0	bit1	bit0	bit1	bit0

ED31	ED30	ED29	ED28	ED27	ED26	ED25	ED24
A/D#15		A/D#14		A/D#13		A/D#12	
bit1	bit0	bit1	bit0	bit1	bit0	bit1	bit0

Bit 1	Bit 0	Gain Value
0	0	1
0	1	2
1	0	4
1	1	8

7.1.16 FIFO data 0x9020 0070

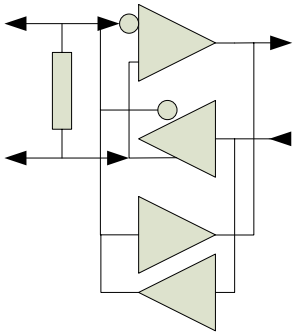
The data can be stored in a 64-kB 32-bit wide hardware FIFO based on the IDT72V285. The data is always stored 16 channels at a time, in eight 32-bit words.
The data then is also retrieved 32-bit, 2 channels, at a time. Only 32-bit reads are possible. There is no intrinsic mechanism signaling the beginning of a 16-channel sequence. It is the responsibility of the program to insure that proper tracking is kept.

7.1.17 External Triggers

Multiple external lines can be used to activate A/D converter, synchronize multiple boards, synchronize combined D/A outputs. Signal logic can be TTL or RS422C.

7.1.17.1 D/A external trigger 0x9020 0054

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
		EN_EXT_LOAD	D/A reset bit	En_Load_DIFF_IN	En_Load_DIFF OUT	En_Load_TTL_IN	En_Load_TTL OUT



EN_EXT_LOAD
When set to "1", a positive signal with a minimum pulse of 40 nS will transfer the content of the first D/A registers to the Output D/A register in a synchronous fashion.

D/A reset bit
When set to "1", it will maintain the D/A in a reset mode.
D/A load signal from the internal logic can be output to the external connector for synchronization with multiple board. Logic can be TTL or RS422 differential.
BY the same way the load signal can input to the board.

7.1.17.2 A/D external trigger 0x9020 0060

ED7	ED6	ED5	ED4	ED3	ED2	ED1	EDO
		En_EVENT _TTL	En_TRIG_ TTL	En_CLK_ DIFF_IN	En_CLK_ DIFF OUT	En_CLK_ TTL_IN	En_CLK_ TTL OUT

The EXT_TRIG signal generates an internal positive pulse when set from low to high. The logic levels can be TTL or RS 422.
EXT_EVENT signal is re-synchronized to the local ECLKOUT clock (100 MHZ) and can be the source of the A/D start conversion if bit # 0 of register 0x9020 0028 is set to "1". The logic levels can be TTL or RS 422.

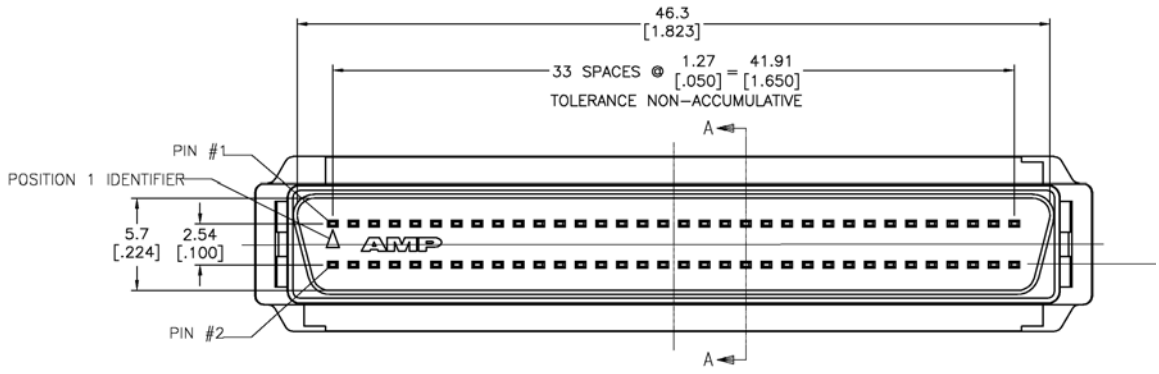
As an output, the EXT_CLOCK signal is the sample clock signal use by the internal A/D converter.
As an input this signal is re-synchronized to the local ECLKOUT clock (100 MHZ) and can be the source of the A/D sample clock.
These three signals are presently inactive.

7.2 JTAG mode

Pin	Signal name	Description
1	TCK	Clock Signal
2	GND	Ground
3	TDO	Data from device
4	VCC	Power supply
5	TMS	JTAG state machine control
6	NC	Power supply
7	NC	No connect
8	NC	No connect
9	TDI	Data to Device
10	GND	Ground

7.2.1 Analog connector P1

Pin are described by function



7.3 A/D signals

New Pin	Signal	
1	**AGND**	Analog Ground
2	**AGND**	Analog Ground
3	***IN1+***	Analog positive Input 1
4	***IN1-***	Analog negative Input 1
5	***IN2+***	Analog positive Input 2
6	***IN2-***	Analog negative Input 2
7	***IN3+***	Analog positive Input 3
8	***IN3-***	Analog negative Input 3
9	***IN4+***	Analog positive Input 4
10	***IN4-***	Analog negative Input 4
11	***IN5+***	Analog positive Input 5
12	***IN5-***	Analog negative Input 5
13	***IN6+***	Analog positive Input 6
14	***IN6-***	Analog negative Input 6
15	***IN7+***	Analog positive Input 7
16	***IN7-***	Analog negative Input 7
17	***IN8+***	Analog positive Input 8
18	***IN8-***	Analog negative Input 8
19	***IN9+***	Analog positive Input 9
20	***IN9-***	Analog negative Input 9
21	***IN10+***	Analog positive Input 10
22	***IN10-***	Analog negative Input 10
23	***IN11+***	Analog positive Input 11
24	***IN11-***	Analog negative Input 11
25	***IN12+***	Analog positive Input 12
26	***IN12-***	Analog negative Input 12
27	***IN13+***	Analog positive Input 13
28	***IN13-***	Analog negative Input 13
29	***IN14+***	Analog positive Input 14
30	***IN14-***	Analog negative Input 14
31	***IN15+***	Analog positive Input 15
32	***IN15-***	Analog negative Input 15
33	***IN16+***	Analog positive Input 16
34	***IN16-***	Analog negative Input 16
35	**AGND**	Analog Ground
36	**AGND**	Analog Ground

7.4 D/A signals

New Pin	Signal	
37	**AGND**	Analog Ground
38	**AGND**	Analog Ground
39	***DA_OUT1***	Analog output signal 1
40	***DA_OUT2***	Analog output signal 2
41	***DA_OUT3***	Analog output signal 3
42	***DA_OUT4***	Analog output signal 4
43	***DA_OUT5***	Analog output signal 5
44	***DA_OUT6***	Analog output signal 6
45	***DA_OUT7***	Analog output signal 7
46	***DA_OUT8***	Analog output signal 8
47	***DA_OUT9***	Analog output signal 9
48	***DA_OUT10***	Analog output signal 10
49	***DA_OUT11***	Analog output signal 11
50	***DA_OUT12***	Analog output signal 12
51	***DA_OUT13***	Analog output signal 13
52	***DA_OUT14***	Analog output signal 14
53	***DA_OUT15***	Analog output signal 15
54	***DA_OUT16***	Analog output signal 16
55	***AGND***	Analog Ground from External Power Supply
56	***AGND***	Analog Ground from External Power Supply
57	***+15V Ext***	External +15V Power supply
58	***+15V Ext***	External +15V Power supply
59	***-15V Ext***	External -15V Power supply
60	***-15V Ext***	External -15V Power supply

7.5 Other

Pin		Signal	
61		DA_XVREF1	Analog input Reference for D/A channel # 1 to #4
62		DA_XVREF2	Analog input Reference for D/A channel # 5 to #8
63		DA_XVREF3	Analog input Reference for D/A channel # 9 to #12
64		DA_XVREF4	Analog input Reference for D/A channel # 13 to #16
65		AGND	Analog Ground
66		GND	Digital Ground
67		DA_XVREF	Analog input GLOBAL Reference for D/A
68		NULL_OUT	Open drain OUTPUT FOR SERVO CONTROL APPLICATION

8 PROGRAMMING EXAMPLE

8.1 A/D ACQUISITION

8.1.1 Introduction

Two mode of acquisition are available to get data from the A/D converter:

-Direct mode. An acquisition is initiated, the DSP wait until the A/D have finished converting the analog signal.

-FIFO mode. An acquisition is initiated. The state machine stores the A/D data automatically within the FIFO at the end of acquisition and then starts the next acquisition.

8.1.2 Direct mode

It is the simplest mode.

-Verify that bit #3 from register 0x90200040 is = "0". This bit make the selection between DIRECT or FIFO mode. In DIRECT mode the A/D data result registers use address lines to be read, while on FIFO mode this registers are controlled by the state machine.

For two' complement set bit #7 to "1". (This bit is inverted after the read-back register to have the A/D in Straight binary mode upon reset).

- Write for example 0x1 (any data) at address 0x90200038 to start acquisition.
- Monitor busy signal BUSYNOT until going low by reading the FIFO status register bit # 1 at address 0x90200034.
- A/D data are now available at address 0x902000A0 to 0x902000AC.

The conversion results can be read individually. They can also be read 2 at a time by doing a 32-bit access.

```
volatile unsigned short *FifoStatus          = ((volatile unsigned short *)0x90200034);
volatile unsigned short *StrtConvReg         = ((volatile unsigned short *)0x90200038);
volatile unsigned int *AdCntrl               = ((volatile unsigned int *)0x90200040);
volatile unsigned short *AdConverter         = ((volatile unsigned short *)0x902000a0);

static void StartAcquisition()
{
        *AdCntrl = 0;           // direct mode
        *StrtConvReg = 1;       // start the acquisition

        // wait for the data to be ready
        while ((*FifoStatus & 2) == 2);
}

void AdDispAll()
{
        int index;

        StartAcquisition();
        for (index = 1; index<16; index++){
                printf("channel #%d: %x\n", index, AdConverter[index]);
        }
}
```

8.1.3 FIFO mode

The FIFO fills up with the A/D data and stops storing the data as soon as it is full. Data now needs to be read by the D.S.P. In this 2nd case, the position of the FIFO pointer to stop acquisition can be selected, and/or the logic can work under interrupt. Data can be acquired in a continuous manner if the DSP can read the FIFO data as fast or faster as it is stored into the FIFO

- Monitor FF signal by reading register 0x90200034 bit # 7 until low.
- Read data from FIFO data register at address 0x90200070 until FIFO empty. Check the FIFO register 0x90200034 bit # 4. (EF)

Continuous acquisition will require programming of the FIFO pointers PAE and PAF with a software program that will "throttle the reading of the data from the FIFO using these two pointers. See example Test Program provided.

```
volatile unsigned short *CycloneTimer          = ((volatile unsigned short *)0x90200000);
volatile unsigned int *MasterFifoReset         = ((volatile unsigned int *)0x90200004);
volatile unsigned int *ClockSourceSel          = ((volatile unsigned int *)0x9020001c);
volatile unsigned short *FifoStatus            = ((volatile unsigned short *)0x90200034);
volatile unsigned short *StrtConvReg           = ((volatile unsigned short *)0x90200038);
volatile unsigned int *AdCntrl                 = ((volatile unsigned int *)0x90200040);  // bit 3: FIFO register
volatile unsigned int *PgaGain                 = ((volatile unsigned int *)0x90200048);
volatile unsigned int *FifoData                = ((volatile unsigned int *)0x90200070);

static void InitAdTimer(int t)
// t: period in nanosecond
{
        *CycloneTimer = (unsigned short) (t / 10) -1;                   //
        Discard(*CycloneTimer);                                 // read back the value to start the count
}

static void StartFifo()
{
        *brdControlRegister = *brdControlRegister | 0x20;       // Reset the analog board
        *brdControlRegister = *brdControlRegister & 0xffdf;     // Reset the analog board
        *AdCntrl = 8;                                           // allow accessing FIFO pointers in parallel
        *MasterFifoReset = 0;                                   // any write to this address does a master FIFO reset
        *AdCntrl = 4;                                           // ready to access the FIFO data
        *ClockSourceSel = 0xc0;                                 // select the local timer as trigger source
        *PgaGain = 0;                                           // make sure that the gain is set to 1
        InitAdTimer(2000);                                      // set the timer for 2 microseconds
}

boolean CheckFifoFull()
{
        short i;

        i = *FifoStatus;
        return ((i & 0x80)==0);
}

boolean CheckFifoEmpty()
{
        short i;

        i = *FifoStatus;
        i = i & 0x10;
        return (i==0);
}
```

```
int ReadFifo(unsigned short *s)
// s is an array of 16 values, for each of the channel
// returns 0 for no error
// returns 1 for underflow
{
        int i;
        int *values;            // we are going to access the data 32-bit at a time

        values = (int *)s;
        if (CheckFifoEmpty()) return 1;

        for (i = 0; i<8; i++)
                values[i] = *FifoData;

        return 0;
}

int main()
{
        int i;
        unsigned short ADdata[15];

        StartFifo();
        *StrtConvReg = 0;               // any write will start the A/D conversion
        while (!CheckFifoFull())
                delay(1); // wait for the FIFO to be full
        for (i = 0; i < 0x4000; i++) {
                ReadFifo( ADdata );
                // some processing here
        }
}
```

8.2 I/O MODULE

8.2.1 I/O interrupt generation on Change Of State (COS)

Test configuration: Port # 0, 2, 3 as input, Port #1, 4, 5 as output.
Port # 0 is connected to port # 1.
We will use bit #0 of port 1 to generate an interrupt INT# 5 to the DSP.

-Configure Port: 0x32 at address 0x902FF04
-Read Port # 0 to clear any interrupts
-Enable C.O.S. of port # 0 writing 0x01 at address 0x902FF028
-Enable Mask bit associate with Port # 0 writing 0x01 at address 0x902FF010.
-Set output of port #1 all to "0" by writing 0x00 at address 0x902FF009
-Read Port #0 will latch the intended status of the actual input.

The latched value can be read also at address 0x902FF034.

-Enable C.O.S. interrupt by writing 0x72 at address 0x902FF004
-Now write a "1" into Port #1 at address 0x902FF009

An interrupt should be generating. C.O.S. status can be seen at address 0x902FF000 where Bit #0 should be low.

Reading Port #0 should remove the interrupt and store the actual status of the input. Verify the latched data at address 0x902FF034.
Writing a "0" into Port #1 will generate another interrupt…

8.2.2 Event capture

This snippet of code programs the parallel interface for event capture. In this example in test mode, we will use a loop-back cable

```
int IO_Event_Test ()
{
        char c;
        volatile char c2;
        unsigned short s;

        *Pio_Reg_Conf = 0x32;
        c = *Pio_Port0;
        *Pio_CosMask0 = 0;
        *Pio_CosSlt = 0;                        // AEQB
        *Pio_EvntCtrl |= 2;                     // reset FIFO
        *Pio_EvntCtrl = 0;
        *Pio_EvntRst = 0;                       // general reset
        *Pio_EvntTsub = 0x20c49c;               // set the capture window to 40 microseconds

        printf("Event count: %d, fifo pointer: %d\n", *Pio_EvntCount, *Pio_EvntFifoPtr);
        *Pio_CosCtrl = 1;                       // set the masks for the event
        *Pio_CosMask0 = 1;

        c = 1;
        *Pio_Port1 = c;
        c2 = *Pio_Port0;
        *Pio_EvntCount = 0;                     // start the time step

#if DEBUG
        // use the loop-back cable to have the DSP send dummy data, generating events
        *Pio_Port4 = 0x55;
        *Pio_Port5 = 0xaa;
        *Pio_Port1 = 0x0;
        c2 = *Pio_Port0;
        *Pio_Port4 = 0xaa;
        *Pio_Port5 = 0x55;
        *Pio_Port1 = 0x1;
        c2 = *Pio_Port0;
        *Pio_Port4 = 0x33;
        *Pio_Port5 = 0xcc;
        *Pio_Port1 = 0xa0;
        c2 = *Pio_Port0;
#endif

        while (*Pio_EvntFifoPtr){
                s = *Pio_EvntCntr;
                printf ("event left: %d, count: %d (0x%04x), data: 0x%08x 0x%04x\n",
                        *Pio_EvntFifoPtr, s, s, *Pio_EvntFifoL, *Pio_EvntFifoH);
        }
        printf("Event count: %d, fifo pointer: %d\n", *Pio_EvntCount, *Pio_EvntFifoPtr);

        return COMMAND_SUCCEEDED;
}
```

8.3 PCI ACCESS

8.3.1 Resetting the board in a Windows XP environment

```
HRESULT PlxDevice::Reset()
{
        HRESULT result;

        if (!m_pPlxRegisters)
                return ALPHI_E_BAD_CARD_CONFIGURATION;

        m_ProgramStartingAddress = 0;

        m_pPlxRegisters->cntrl = 0xc0017676;            // assert the board local reset

        DWORD Expiration = QueryTimeInMs() + scm_resetdelay;
        DWORD Expiration2 = QueryTimeInMs() + scm_resetdelay/2;

        while (QueryTimeInMs() <= Expiration2);

        GetMboxStatus(0xffffffff);

        // Since the DSP is not running yet, it has not set
        // the MBOX empty bits yet.
        m_pPlxRegisters->p2ldbell = 0xf0;

        while (QueryTimeInMs() <= Expiration);

        // Turn off PLX related interrupts on the DSP.
        m_pPlxRegisters->intcsr &= ~0x10000;
        // Remove the reset, turning off access from HOST
        m_pPlxRegisters->cntrl  = 0x17676;                      // remove the local reset

        if (m_DeviceCaps.ProcessorType == None)
                return ALPHI_S_OK;

        Expiration = QueryTimeInMs() + scm_startupdelay;

        // wait for DSP to clear the status bits
        while (m_pPlxRegisters->p2ldbell && (QueryTimeInMs() <= Expiration));

        if (m_pPlxRegisters->p2ldbell) {
                printf("The board did not clear up the doorbell register\n");
                return ALPHI_E_DEVICE_FAILED_RESET;
        }

        // Reset the empty bits
        m_pPlxRegisters->p2ldbell = 0xf0;

        if (FAILED(ReadMbox(2, true))){
                printf("Failed to read the mailbox\n");
                return ALPHI_E_DEVICE_FAILED_RESET;
        }

        if ((result = IsKernelRunning()) == ALPHI_E_DSP_NOT_RUNNING_KERNEL) {
                printf("The kernel test failed!\n");
                return ALPHI_E_DSP_NOT_RUNNING_KERNEL;
        }
        else return result;
}
```

8.3.2 Accessing the local peripheral using HPI access

The 6713 HPI interface allows accessing any address on the board, through the processor itself. This snippet of code gives an example of using the interface. It can be much improved by using the auto-increment access for instance.

```
typedef struct hpi_t {
        long ctrl;
        long addr;
        long data;
        long autoinc;
} hpi_t;
hpi_t *hpi;

ULONG readHPI(ULONG addr)
{
        int i = DELAY;

        hpi->addr = addr;
        hpi->ctrl = 0x00110011;
        while ((hpi->ctrl & 8)&& (i--)) ;                    // wait for ready
        if (i == 0) printf("\nTimeout accessing the HPI and waiting for ready\n");
        return hpi->data;
}

void writeHPI(ULONG addr, ULONG data)
{
        int i = DELAY;

        hpi->addr = addr & 0xfffffffc;                       // force a 32 bit boundary
        hpi->data = data;
        hpi->ctrl = 0x00110011;
        while ((hpi->ctrl & 8)&& (i--)) ;                    // wait for ready
        if (i == 0) printf("\nTimeout accessing the HPI and waiting for ready\n");
}
```