SMT130

User Manual V1.0

Revision History

Date	Comments	Engineer	Version
24/05/04	Initial SMT130 Draft	TJW	1.0

Table of Contents

Table of Tables

Table of Figures

Table Of Abbreviations

BAR	Base Address Region
DMA	Direct Memory Access
EPLD	Electrically Programmable Logic Device
PCI	Peripheral Component Interconnect
SDB	Sundance Digital Bus
SRAM	Static Random Access Memory
TBC	Test Bus Controller
TIM	Texas Instruments Module

Table 1 : Table of Abbreviations

1 Introduction

The SMT130 is a single site module carrier board that provides access to a single TIM module over the PCI bus in a PCI-104 stack system.

An on-board JTAG controller allows systems to be debugged using Code Composer Studio. There is also a JTAG input and JTAG output connector on the SMT130. This allows the debugging of all the TIM modules in the PCI-104 stack from a single JTAG controller.

The main connection to the PCI bus is via the module Global Bus. A single Comport is also mapped to the PCI bus providing support for application boot and data transfer.

A 1 MB of SRAM is mapped on to the Global Bus and can be accessed by the TIM as a global resource or by the PCI Bridge.

A through board header is provided for RESET_IN and RESET_OUT to allow multiple SMT130 carriers in a stack to be synchronised.

The board requires a 5 volt supply only, which is taken from the PCI-104 stack connector or an auxiliary power connector from stand alone applications. The 3.3 volt supply required for supplying power to the TIM module is generated locally.

2 Functional Description

The PCI interface connects to a Quick Logic EPC363 Bridge device. It has a 32-bit 33MHz PCI interface that supports I^2C control, mailbox register access and direct memory reads and writes. The PCI bus is translated to a Local PCI bus, which is connected to the following devices:

- Shared SRAM 1MB
- Control EPLD that manages Comport access
- JTAG controller
- Module Global Bus
- PCI Bridge device

An on-board arbitration unit controls which device, Master Module or PCI Bridge, has access to this local PCI bus resource.

The local PCI bus has a 33MHz clock to control transfers between the various resources. This is available on the CLKIN pin on the Master site and should be selected in preference to the on-board oscillator to allow the DSP to synchronise its accesses to and from the PCI Bridge registers. The PCI Bridge has an input and output FIFO capable of transferring 256 32-bit words of data to and from the DSP at 33MHz, thus bursting a maximum local bus transfer rate of 132MB/s.

The TIM Module can access the SRAM over the PCI local bus at transfer rates up to 100MB/s. The number of wait states required by the Master Module will vary depending on the speed of the module. Maximum access rates use a 20ns strobe cycle.

The JTAG controller is based on the TI 8990 device, and drivers can be supplied for Code Composer Studio (Part Number SMT6012).

3   Setting Up the SMT130

The SMT130 should be set up in the following way.

- Turn the system off and add the SMT130 (With TIM already installed) to the PCI-104 stack. Take care to ensure that all stacking connector pins locate correctly. Secure the SMT130 in the stack with the pillars and screws provided. Adjust rotary switch SW1 to ensure that the SMT130 has a unique stack position. No two cards in the PCI-104 stack must have the same setting. Note only positions 0 to 3 are valid.

- Switch on PC and wait for the OS to boot up.

- Windows 95/98/NT/2000 will detect a new hardware.

- Windows should automatically find the drivers from the CD, if not browse to the CD or if you downloaded from the ftp site to the folder where you unzipped the SMT6300 software.

- 3L application software will detect the SMT130 if the SET TISLINK variable is set to SMT320.

- You can run the SMTBoardInfo application to detect the number of SMT130s in your system and report their stack positions and I/O addresses. This information is required when setting up code composer for the board. SMTBoardInfo is part of the SMT6300 package.

4   Memory Map

All address information is given in bytes :

4.1   PCI Bridge Chip Internal Register (BAR0)

Please see V363EPC Local Bus PCI Bridge User Manual V1.04 (http://www.quicklogic.com/home.asp?PageID=223&sMenuID=114#Docs) for details of internal registers.

Note: Where required, registers from the V^3 datasheet have been included.

4.2   I/O Space Register Assignments (BAR1)

In target mode, the SMT130 is accessed by a host device across the PCI bus. This allows access to the target mode registers. The operating system or BIOS will normally allocate a base address for the target mode registers of each SMT130. Access to each register within the SMT130 is then specified by this base address and the offset shown in the table below.

The I/O address space is decoded as shown in the table below.

Offset	Register(Write)	Register(Read)	Width
0x0	-	-	
0x4	-	-	
0x8	-	-	
0x0C	-	-	
0x10	COMPORT_OUT	COMPORT_IN	32
0x14	CONTROL	STATUS	32
0x18	INT_CONTROL		32
0x1C	-	-	
0x20 to 0x3F	COMPORT Configuration	COMPORT Configuration	
0x24	COM_SWITCH	COM_SWITCH	16
0x40	INTREG	INTREG	16
0x80 to 0xAF	TBC Write	TBC Read	16

Table 2 : I/O address space map

4.3 Memory Space Assignments(BAR2)

Address	Description	Notes
0x0000 0000 – 0x000F FFFF	Shared Memory Bank	1MB SRAM
0x00200090	Comport Data Mirror	Mirror of COMPORT_OUT / COMPORT_IN in I/O Space Register Assignments (BAR1) See Note 2
0x00200094	Comport Status Mirror	Mirror of Control / Status in I/O Space Register Assignments (BAR1) See Note 2
0x00200098	Comport Int_Control Mirror	Mirror of Int_Control in I/O Space Register Assignments (BAR1) See Note 2
0x0020 0000-0x0020 007F	Global Bus	See Note 1
0x0020 0240 – 0x0020 025F	SDB Data Register	Input/Output 16 bit SDB Interface
0x0020 0260 – 0x0020 027F	SDB Control Register	SDB Control/Status

Table 3 : Memory space map

Note 1: In order for the TIM to respond to accesses for this area address line GADD30 and GADD19 of the TIM site connector must be decoded as high and GADD7 and GADD5 must be decoded as low.

Note 2: These mirrors of Addresses in the I/O Space (BAR1) allow increased transfer speeds across the host Comport link (in excess of 10X increase).

5 DSP Resource Memory Map

The Master module on the SMT130 can access the various resources available, including the Shared SRAM and the PCI Bridge. Access to the PCI Bridge allows the DMA engine in the PCI Bridge to be initiated by the DSP, mailbox registers can also be manipulated. The table below illustrates the resources and their corresponding address region when accessed by the Master module.

C60 Address Access	Description	Notes
0xD000 0000 – 0xD00F FFFF	Shared Memory Bank	1Mbyte SRAM
0x1C00 0000 – 0x1C00 00FF	PCI Bridge Registers	PCI Bridge Internal resisters
0x1800 0000 – 0x183F FFFF	Local-to-PCI Aperture 0	PCI Bridge Aperture 0 Space

Table 4 : Memory space map

6 Shared Memory Resource

There is 1Mbyte of SRAM on the SMT130, this shared memory can be accessed by the PCI host and the TIM module. This allows applications to transfer data between the host PC and the DSP at data rates approaching 100MB/s. The address of the shared memory is shown in the memory map.

The PCI Bridge DMA processor sees the shared memory at a different address from that used for normal accesses. For normal memory access the memory base address register offset is 0x0000 0000. For DMA access address line A28 (On hardware interface) must be high, therefore DMA memory access starts at 0x4000 0000 (Not 0x1000 0000 as addressing is in bytes).

7   Comports

The SMT130 gives access to five of the six TIM Comports. Comports 0, 1, 4 and 5 are connected to 14-way surface-mount FMS connectors. This gives external access to the TIM. Note, this external access must be with in the same system stack.

Comport 3 is hard wired to the PCI interface. This gives provision to boot the TIM and configure FPGAs on the TIM.

Comport 2 has no physical connection and therefore can't be used on a SMT130.

Comport Number	SMT130 Connector reference
0	J6
1	J3
4	J4
5	J13

Table 5 : Comport connector reference

8   Comport to PCI Interface

The Comport interface is memory mapped to the PCI Bridge as illustrated in table 2 : I/O address space map. The Comport uses the Control and Data registers to detect the state of the input and output FIFO. The following section describes the bit definitions for these registers.

8.1   Comport Registers (Offset 0x10, BAR1)

The host is connected to the first TIM site using Comport 3. This port is bi-directional and will automatically switch direction to meet a request from either the host or the DSP. Both input and output registers are 32 bits wide. Data can only be written to COMPORT_OUT when STATUS[OBF] is 0. Data received from the DSP is stored in COMPORT_IN and STATUS[IBF] is set to 1. Reading COMPORT_IN will clear STATUS[IBF] and allow another word to be received from the DSP.

8.2   Control Register (Offset 0x14, BAR1)

The CONTROL register can only be written. It contains flags, which control the boot modes of the first TIM site.

7-5	4	3	2	1	0
	notNMI	IIOF2	IIOF1	IIOF0	RESET

RESET	Write a 1 to this bit to assert the reset signal to the TIM module on the SMT130.
IIOF0 IIOF1 IIOF2	These bits connect to the corresponding pins on the TIM site 1. Writing 0 causes the corresponding IIOF line to go low.
NotNMI	A 0 written to this bit will assert the active low NMI to the TIM.

Table 6 : Control Register

Note 1. On PCI system reset, RESET is asserted to the TIM site.

Note 2. The TIM reset signal on the STM130 is connected to a stacking connector that can common up the TIM reset signals. If there is more than one SMT130 card in a system stack, the TIM reset signal can be connected together if the reset enable jumper is inserted. This maybe required where all TIMs in a system are to be reset at the same time.

8.3 Status Register (Offset 0x14, BAR1 , Read-Only)

31:22	21	20	19	18	17	16	15:12	11	10	9	8
	CONFIG_L	TBC RDY	0	MASTER	IBF	OBF		IM2	IM1	IM0	INTD

7	6	5	4	3	2	1	0
C40 INT	TBC INT	IBF INT	OBE INT	C40 IE	TBC IE	IBF IE	OBE IE

OBE IE	Set if Comport output buffer empty interrupts enabled.
IBF IE	Set if Comport input buffer full interrupts enabled
TBC IE	Set if JTAG interrupts enabled
C40 IE	Set if interrupt from TIM DSP enabled
OBE INT	Set if the Comport output buffer becomes empty. Cleared by writing a 1 to the corresponding bit in the interrupt control register.
IBF INT	Set if the Comport input buffer receives a word. Cleared by writing a 1to the corresponding bit in the interrupt control register
TBC INT	Set when the TBC asserts its interrupt. Cleared by removing the source of the interrupt in the TBC.
C40 INT	Set when the TIM DSP sets its host interrupt bit. Cleared by writing a 1 to the corresponding bit in the interrupt control register.
INTD	The logical OR of bits 7—4 in this register gated with each one's enable bit.
OBF	Set when a word is written to the Comport output register. Cleared when the word has been transmitted to the DSP.
IM0	Interrupt mask 0. Returns Interrupt Control Register Bit 8.
IM1	Interrupt mask 1. Returns Interrupt Control Register Bit 9.
IM2	Interrupt mask 2. Returns Interrupt Control Register Bit 10.
IBF	Set when a word is in the Comport input register.
MASTER	Set when the SMT130 Bridge owns the Comport interface token.
TBC RDY	Reflects the current state of the TBC RDY pin. This bit is active high and therefore and inversion of the TBC pin.

CONFIG_L	Reflects the state of the TIMs' CONFIG signal. Active low.

Table 7 : Status Register

INTD is the input interrupt into the PCI Bridge from the SMT130, this can be routed to either INTA, INTB, or INTC using the PCI Interrupt Configuration Register (offset 0x4C, BAR0)

8.4 Interrupt Control Register (Offset 0x18, BAR1)

This write-only register controls the generation of interrupts on the PCI bus. Each interrupt source has an associated enable and clear flag. This register can be written with the contents of bits 7:0 of the Status Register.

10	9
DSP-PC IIOF2 En	DSP-PC IIOF1 En

8	7	6	5	4	3	2	1	0
DSP-PC IIOF0 En	CLEAR C40 INT	0	CLEAR IBF INT	CLEAR OBE INT	C40 IE	TBC IE	IBF IE	OBE IE

DSP-PC IIOF2 En	Enables DSP-PC interrupts on IIOF2
DSP-PC IIOF1 En	Enables DSP-PC interrupts on IIOF1
DSP-PC IIOF0 En	Enables DSP-PC interrupts on IIOF0
IBF IE	Comport Input Buffer Full Interrupt Enable. Allows an interrupt to be generated when the host Comport input register is loaded with data from the C40.
OBE IE	Comport Output Buffer Empty Interrupt. Allows an interrupt to be generated when the host Comport register has transmitted its contents.
TBC IE	Test Bus Controller Interrupt Enable. Interrupts from the Texas JTAG controller are enabled when set.
C40 IE	C40 Interrupt Enable. Allows a programmed interrupt to be generated by the C40 when set.

CLEAR OBE INT	Write a one to this bit to clear the interrupt resulting from a Comport output event.
CLEAR IBF INT	Write a one to this bit to clear the interrupt event resulting from Comport input.
CLEAR C40 INT	Write a one to this bit to clear down the C40 INT event.

Table 8 : Interrupt Control Register

The JTAG controller which generates TBC INT must be cleared of all interrupt sources in order to clear the interrupt.

9 JTAG Controller

The SMT130 has an on board Test Bus Controller (TBC). The TBC is controlled from the PCI bus giving access to the on site TIM and/or any number of external TIMs. The TBC is a SN74ACT8990 from Texas Instruments. Please refer to the Texas Instruments data sheet for details of this controller. The TBC is accessed in I/O space at the Base address + 0x80.
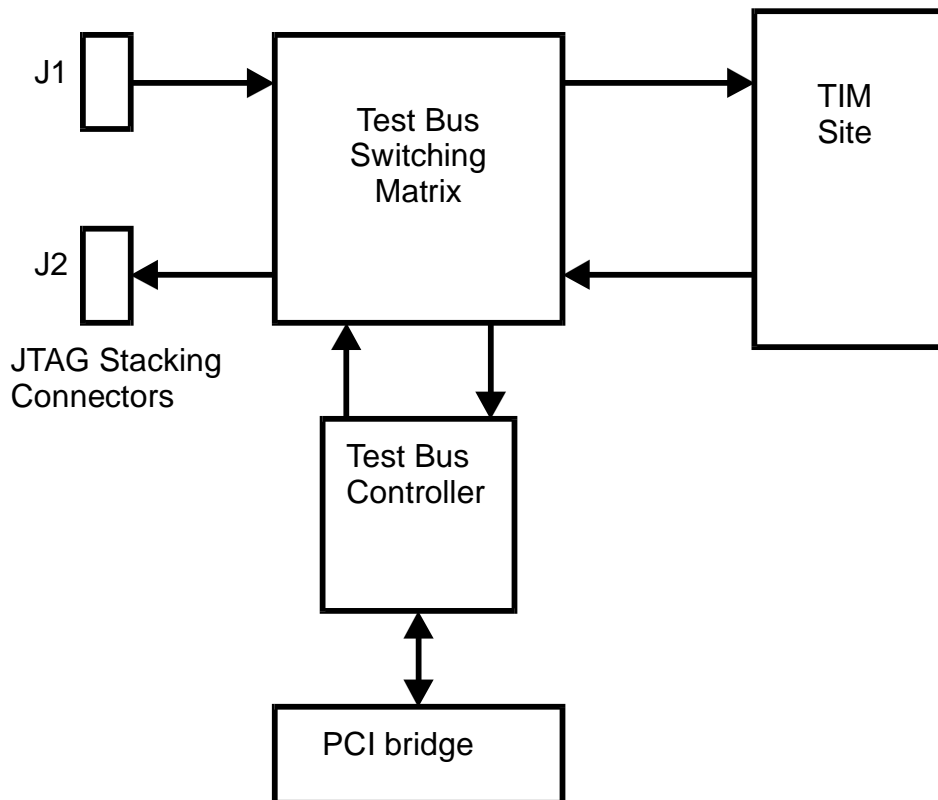
Figure 1 : TBC Data Routing

The SMT130 can operate in two TBC modes; Master mode and Slave mode. In Master mode, the Test Bus Controller on the SMT130 drives the JTAG scan chain through the TIM site on the SMT130. If the site is not populated with a TIM then the modules SENSE signal is used to enable a tri-state buffer connecting the TDI and TDO (JTAG Data In and Data Out) on the site, maintaining the integrity of the JTAG data path. This switching is automatic. The External JTAG Connector J2 is intended to connect to a second SMT130 in the PCI-104 stack. When this is the case the SMT130 automatically detects the device and routes the test data accordingly.

The SMT130 is configured in Slave mode if the TBC and JTAG chain from another SMT130 is being used. In this case the JTAG master is connected to the SMT130 on connector J1. When stacking SMT130 carriers in a PCI-104 system, the

JTAG chain can be automatically connected through the stack by inserting a JTAG coupling connector. If this coupling connector is inserted into the stack, the Lower SMT130 controls the JTAG chain. All other TBCs in the PCI-104 stack will be disabled. Any software used to access the TBC must target ONLY the lowest SMT130 in the stack. If the TJAG coupling connectors are omitted from the stack, each TBC operates independently.

10 Using the SMT130 External/Internal JTAG with TI Tools.

For details on using the SMT130 with T.I. Code Composer range, see the SMT6012 documentation.

The SMT6012 is Sundance's driver for the T.I. Code Composer range of products and can be purchased separately. The SMT6012 is free of charge when the SMT130 is bought with the Code Composer software from Sundance.

11 Firmware Upgrades

Much of the SMT130's control interface is achieved using CPLD's. Sometime customers require slightly different interface protocols, which can be catered for by a firmware upgrade. To upgrade firmware, Xilinx JTAG programming software is required together with a lead to connect to the SMT130's header. The image below shows the location of pin 1 of the CPLD JTAG connector J21. This connector is a 2x3 2mm pin header.

Pin 1 – J21
CPLD JTAG connector
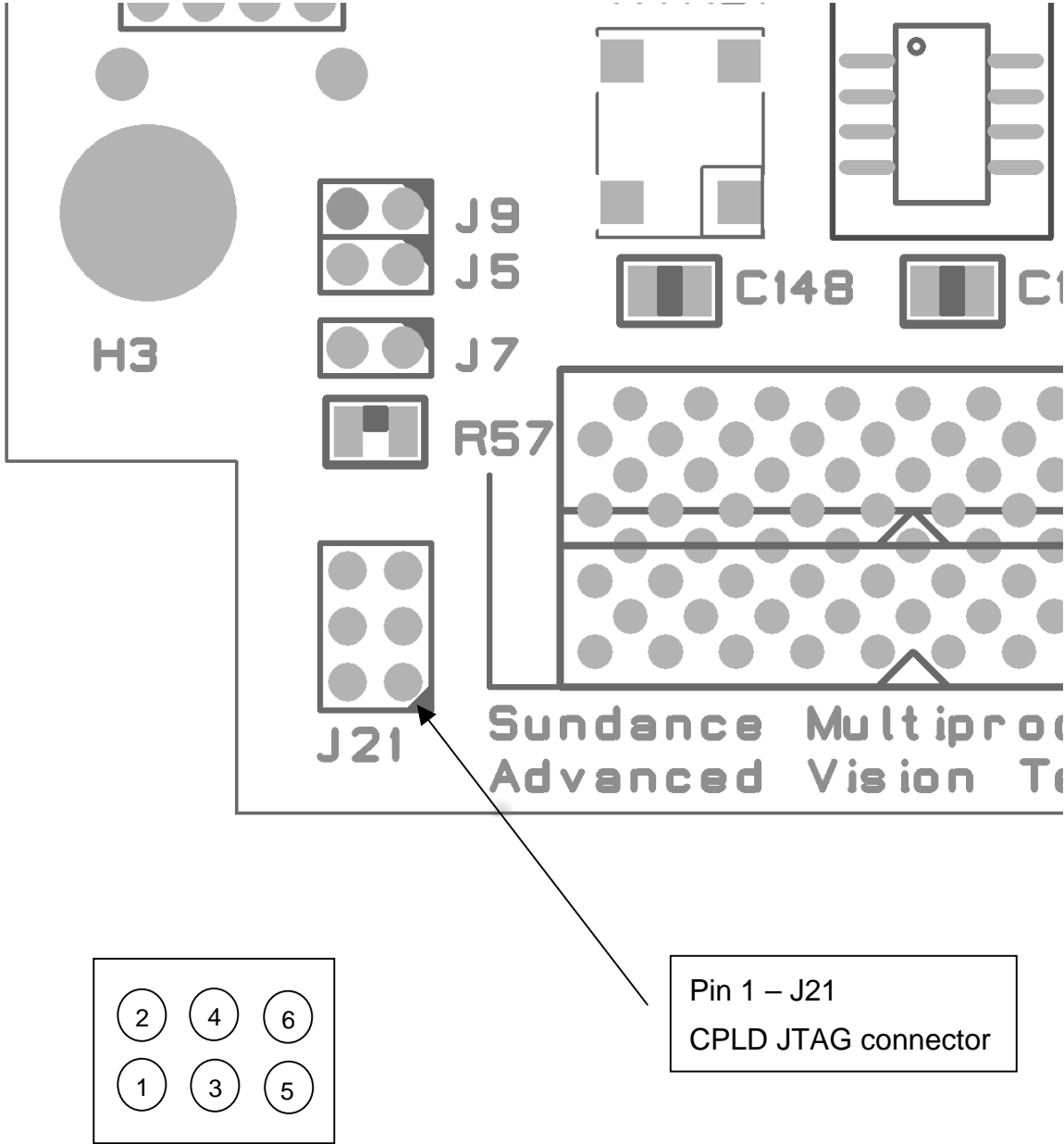
Figure 2 : JTAG header pin numbers

Pin Number	Function
1	Vcc (5v)
2	Gnd
3	TCK
4	TDO
5	TDI
6	TMS

Table 9 : JTAG Header pin function

12 Global/Local Bus Transfers, DSP <-> PCI.

The traditional global bus interface on C6x DSP modules interfaces to the SMT130 via a local bus. This allows Global bus transfers on the DSP to be converted into local bus accesses. This allows direct DSP accesses to the PCI Bridge chip.

The resources in the PCI Bridge chip are illustrated in the figure below.
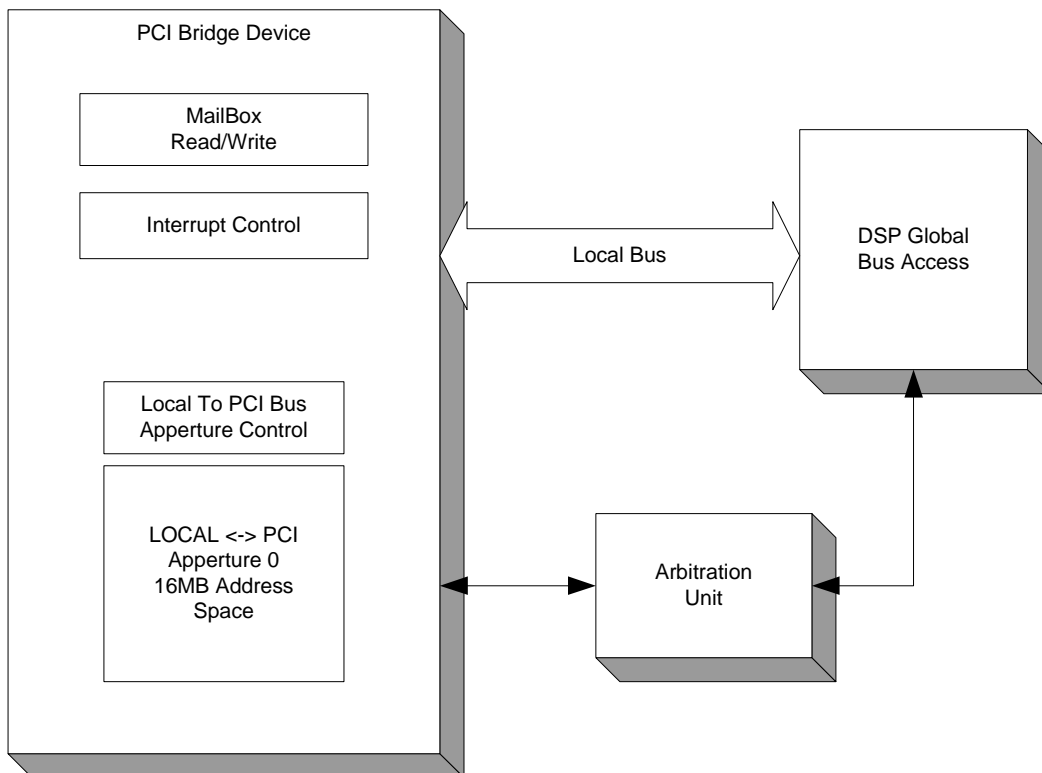
Figure 3 : Local Bus to DSP Connectivity

12.1 Mailbox Accesses

The mailbox registers can be used if small amounts of data or commands between the PCI bus and the DSP, via the local bus, need to be transferred. Rather than sending Comport data, and therefore require the DSP to be checking its Comport for commands, a mailbox write by the PCI bus can be initialized to generate an interrupt on an DSP IIOF line indicating, to the DSP, that data is available.   The PCI Bridge device provides 16 8-bit mailbox registers, which may be used to transmit and receive data between the DSP and Host.

The mailbox registers are accessed from the DSP through the Local-to-Internal Register (LB_IO_BASE) aperture. As illustrated in section 5, table 4 of this document this region is accessed by the DSP via a global bus access to the PCI Bridge Registers (Address : 0x1C00 0000).

The mailbox registers themselves are on byte boundaries with offsets 0xC0 -> 0xCF, from the LB_IO_BASE. As DSP global bus accesses are carried out on WORD (32-bit) boundaries a write access over the global bus to 0x1C00 0000 + 0xC0 will write to the first 4 mailbox registers in the PCI Bridge device.

The mailbox registers are accessed from the PCI bus through the PCI-to-Internal Register (PCI_IO_BASE) aperture. This is accessed via the PCI Bridge Chip Internal Register (BAR0), byte offset 0xC0 -> 0xCF.

12.1.1 Doorbell Interrupts

Each of the 16 mailbox registers can generate four different interrupt requests called doorbell interrupts. Each of these requests can be independently masked for each mailbox register. The four doorbell interrupt types are:

• DSP interrupt request on read from PCI side
• DSP interrupt request on write from PCI side
• PCI interrupt request on read from DSP side
• PCI interrupt request on write from DSP side

The PCI read and DSP read interrupts are OR'd together and latched in the mailbox read interrupt status register (MAIL_RD_STAT). Similarly, the PCI write and DSP write interrupts are OR'd together and latched in the mailbox write interrupt status register (MAIL_WR_STAT). All of the interrupt request outputs from the status registers are OR'd together to form a single mailbox unit interrupt request and routed to both the Local and PCI Interrupt Control Units.
When a block of mailbox registers are accessed simultaneously, for example when 4 mailbox registers are read as a word quantity, then each register affected will request a separate interrupt if programmed to do so.
See section 14 for further information on Interrupts.

12.2 DSP Interrupt Control

Interrupts can be enabled from a number of different sources i.e. DSP-> Host and Host -> DSP. See section 14 for a description of these functions.

12.3 DSP To Local Aperture 0 control and Accessing

The quickest way to transfer information from the DSP to PCI Bus or vice versa is to use the Local-to-PCI Aperture 0 in the PCI Bridge device. A DSP unit may be required to transfer large amounts of acquired data to the PC host for data storage or post-processing etc. Allowing the DSP to take control of the PCI bus means that the HOST only requires to transfer data, from an internal allocated region of memory, after the transfer has been completed by the DSP. Alerting the Host that data has been transferred can be accomplished in a number of ways i.e. writing to the mailbox register, which can then generate an interrupt.

As shown in Table 4, section 5. The Local-to-PCI Aperture 0 is mapped as a region of addressable space from 0x1800 0000 – 0x183F FFFF (words).

There are a number of registers in initialise before data can be read or written via this address space.

- Unlock the PCI Bridge System register. This requires a write to

- Write the upper 8 bits of your destination address (in bytes) to the upper 8 bits of the 32-bit Local Bus to PCI Map 0 register (LB_MAP0_RES, offset in bytes 0x5c).

- Convert you lower 24-bit address to a word aligned value.

- Write/Read data from Local-to-PCI Aperture 0 .

The diagram below illustrates this procedure.

In the example below the *WritetoPCIregisters(offset,data)* function writes data over the DSP's Global bus, at a base address of 0x1800 0000 (words), the first parameter passed to this function in the offset address in words, and the second is the data to be written.

Receive Target address via Com-Port Link

i.e. 0xFE12 3000

↓

Unlock the system register in PCI Bridge

#define LB_CFG_SYSTEM   (0x78>>2) (divide by 4 to word align offset address of PCI bridge register)

WritePCIRegisters((unsigned int)LB_CFG_SYSTEM,0xA05F);   **// Unlock Code**

WritePCIRegisters((unsigned int)LB_CFG_SYSTEM,0x22008800);   **// Restore Value**

↓

Write Lower 24 bits of byte TargetAddress to the LB_MAP0_RES register

#define LB_MAP0_RES   (0x5C>>2)

#define LB_MAP0_MEMORY_RW   0x00060000

#define LB_MAP0_ADDRESS_MASK 0xFF000000

WritePCIRegisters((unsigned int)LB_MAP0_RES,(unsigned int)(TargetAddress & LB_MAP0_ADDRESS_MASK) | LB_MAP0_MEMORY_RW);

// The `LB_MAP0_MEMORY_RW must be Or'ed with the data to ensure the other` register in // the 32-bit word has its correct value.

↓

Write WORD Aligned data to Local Aperture space 0.

*WritePCIApperture0((**ByteToWord32**(((TargetAddress) & ~LB_MAP0_ADDRESS_MASK))),Length,buffer);*

Where

*void WritePCIApperture0(unsigned int address,unsigned long Length,unsigned int *buffer)*

{

unsigned int Index;

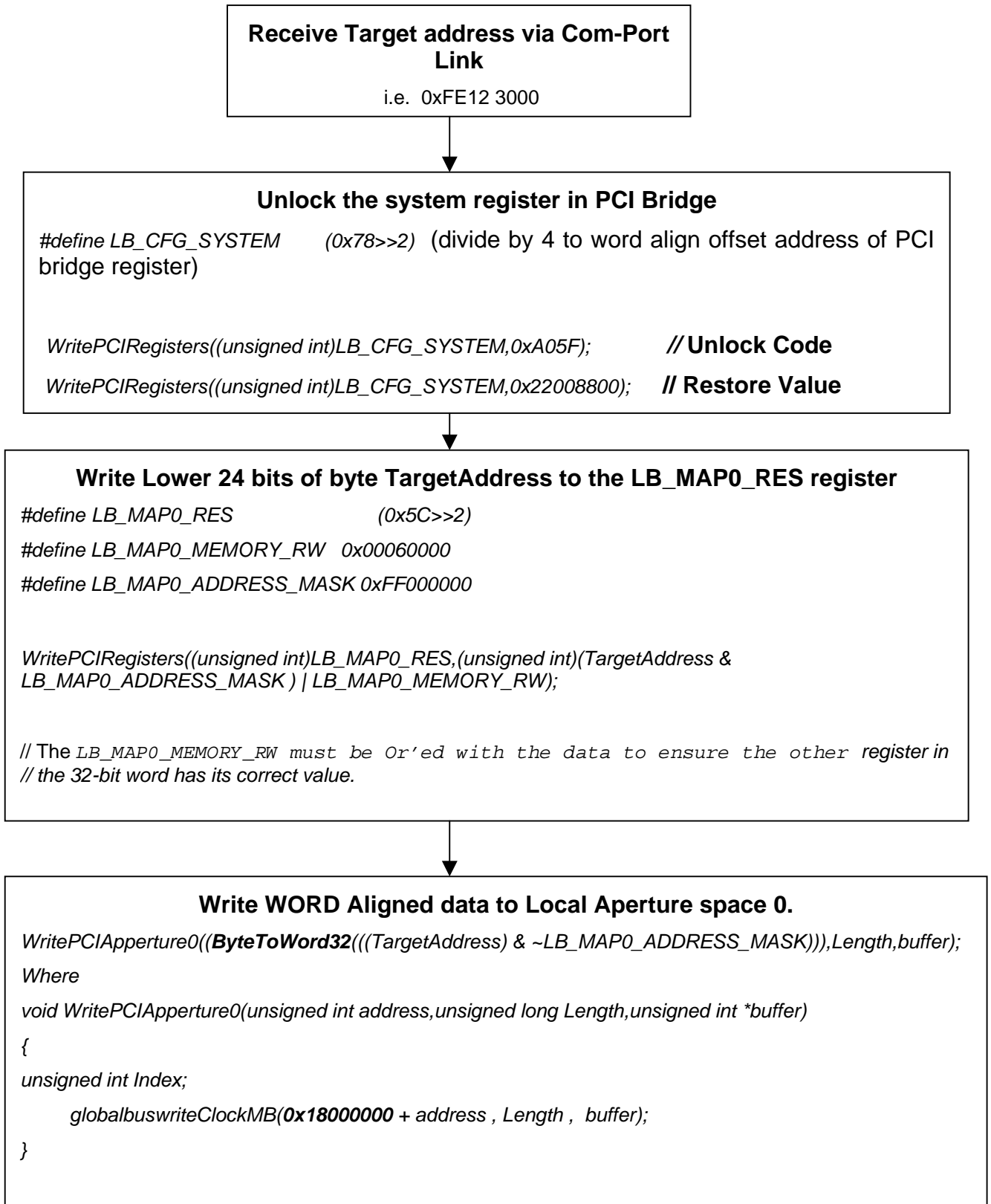*   globalbuswriteClockMB(**0x18000000** + address , Length , buffer);*

}

Figure 4 : DSP Transfer via the Local Aperture 0

12.3.1 Global bus access protocol

In Figure 6, the WritePCIApperture function calls a function C6xGlobalWriteClockMB().

This function enables the DSP's global bus to transfer *Length* words from the DSP's internal memory map pointed to by *buffer.* The function puts the Global bust into burst mode. This interface allows a synchronous stream of data to be written to the 256 WORD input FIFO of the Local To PCI aperture 0. For more information on setting this mode from the DSP can be found in the *SMT335 Users Guide.* This section concentrates on the burst mode interface and arbitration mechanism for the DSP to PCI Bridge aperture access.

The Global bus interface of the DSP uses the following signals to interface to the local bus of the SMT130.

DSP Signals.

AE*, DE*, CE0*

AE*/DE*   are active low address/Data enable signal driven by the SMT130, when the DSP has ownership of the Bus this signal is driven low by the SMT130 allowing the DSP to drive the Address pins and Data pins.

CE0* is the Tri-state control for the DSP's global bus control pins. This is permanently tied low by the SMT130 as the control signal is always enable.

STRB1*

Data strobe signal from the DSP's global bus. This is driven low when the DSP is carrying out an access cycle.   The DSP waits for the RDY1* to be driven low by the SMT130 to indicate transfer has been completed. This interface is carried out in synchronous burst mode. This requires the DSP to indicate when the burst transfer is completed. This is accomplished by the DSP by pulling STAT0 low.

RDY1*

Active low transfer acknowledgement. This is driven by the SMT130 to indicate that the current transfer has been completed.

STAT0..3

DSP Status line. When all of the signals are logic '1' then the DSP Global bus interface is in an idle state. When any of these signals is driven low the DSP requesting ownership of the SMT130's local bus. STAT0 has a special status and indicates the last data packet transfer when driven low by the DSP.

A0..A30

DSP's global Bus address lines

D0..D31

DSP's global Bus data lines

IIOF0, IIOF1 & IIOF2

DSP's Interrupt signals. These are open collector signals on the SMT130. They can be driven by the DSP to generate an interrupt to the host, or they can be driven by the host to interrupt the DSP.

In the timing diagram below all signals change relative to the rising LCLK signal. This signal is the H1 clock signal of the DSP when using the DSP global bus in synchronous mode (see SMT335 User Guide).
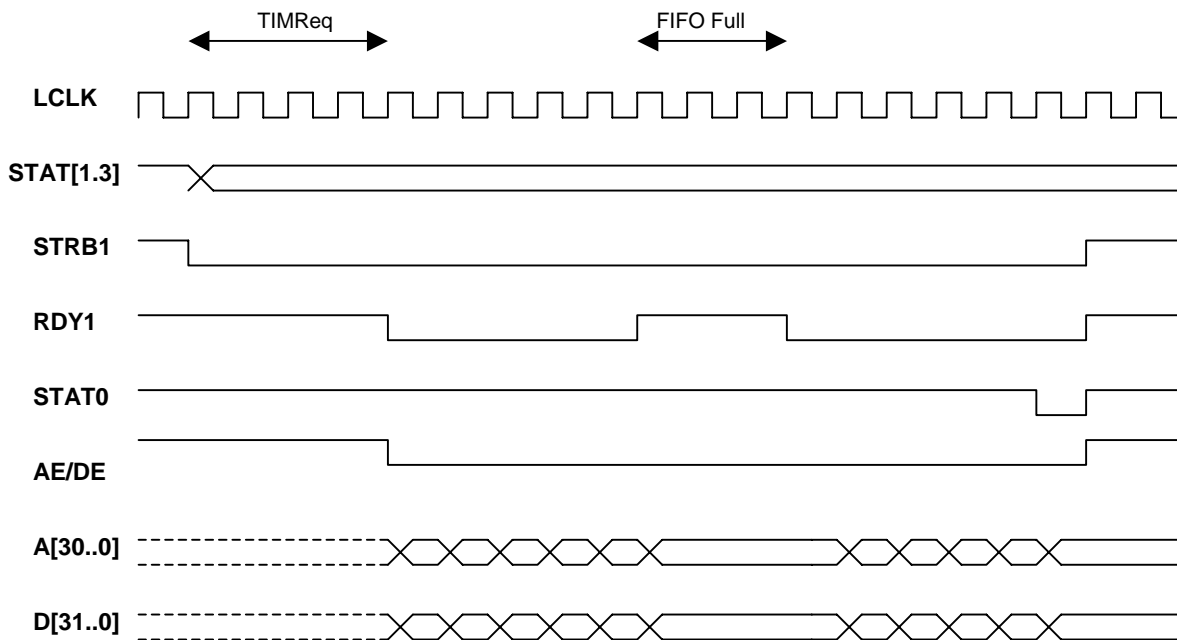

Figure 5 : Timing diagram for DSP local bus access

* LCLK Period =30nS, frequency is 33MHz.

The DSP initiates a global bus R/W by asserting the STRB1 low and STAT[1:3} change (see TIM Spec for details of STAT[1..3] details). Once the arbitration unit detects this, it waits for the last cycle of the Local bus to be completed by the PCI Bridge, before allowing the DSP to become Bus Master. Once the DSP is Master the arbitration unit drives AE and DE low to enable the DSP's address and data lines. RDY1 is driven low by the arbitor to indicate to the DSP, on the next rising LCLK, that the data packet has been transferred. If the input FIFO (256 Words Deep) becomes

full, the arbitration logic de-asserts the RDY1 signal to indicate a hold-off state. Once the data has been transferred from the FIFO to the PCI bus RDY1 is re-asserted to continue the transfer. The end of the burst access is indicated by asserting STAT0 low. If RDY1 is not active then STAT0 should remain asserted until ready is asserted and the final data transaction has been completed.

Situations can arise where a deadlock condition car arrive, i.e. the PCI bus is trying to read from the SMT130 resources while the DSP is reading from the PCI Bus. If this situation arises the arbitration unit gives the PCI Bridge device priority and services the HOST PCI access before giving bus ownership back to the DSP.

When running code composer applications to debug the DSP a reduction in the speed of the debugger will be noticed. The DSP has priority when accessing the local bus and any other accesses will only occur under the following conditions.

- Burst access is finished

- Deadlock condition occurs which releases DSP ownership of the Bus.

For multi-threaded applications the length of the DSP burst can be lowered to allow PCI bus R/W cycles to snatch cycles from the DSP.

13 Interrupts
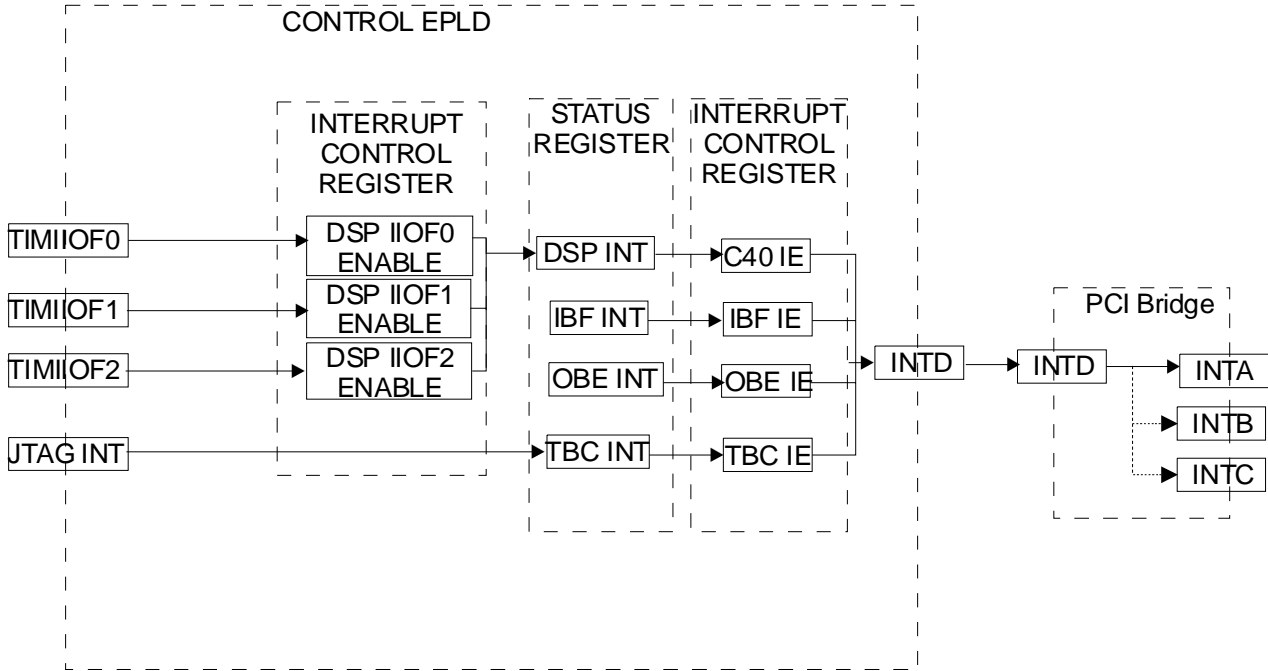
13.1 SMT130-To-PCI Interrupts

Figure 6 : SMT130 to PCI Interrupts

Interrupts can also be caused by the SMT130 by writing or reading the mailbox registers in the PCI Bridge.

13.2 PCI-To-SMT130 Interrupts
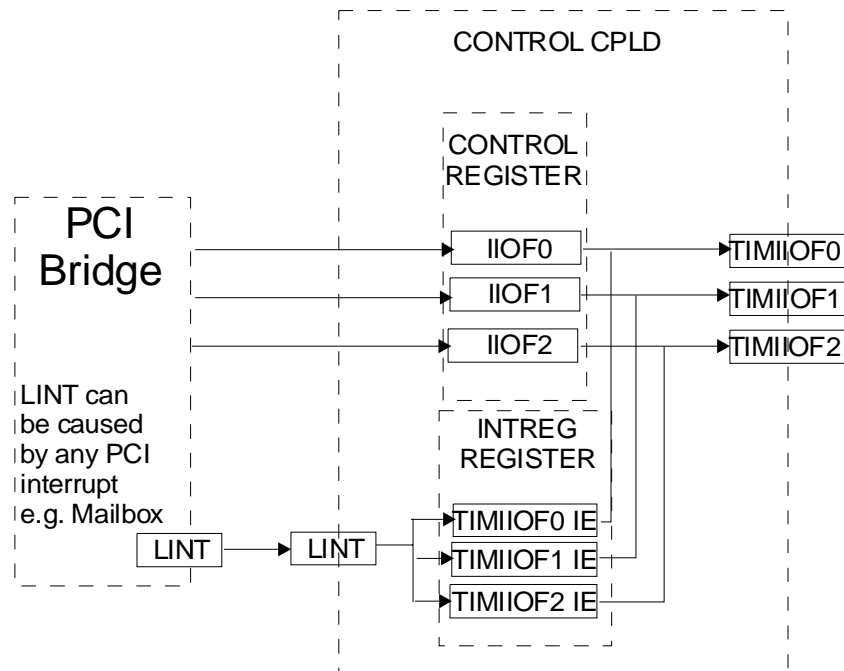
Figure 7 : PCI to SMT130 Interrupts

13.3 Interrupt Registers

The following register are used to control PCI-To-DSP and DSP-To-PCI interrupts:

Note that Control Register (Offset 0x14, BAR1) and Interrupt Control Register (Offset 0x18, BAR1) are also used to control interrupts.

13.3.1 PCI Interrupt Configuration Register(Offset 0x4C, BAR0)

Bits	Name	Description
31	MAILBOX	Mailbox Interrupt Enable: Enables a PCI interrupt from the mailbox unit
30	LOCAL	Local Bus Direct Interrupt Enable: Enables direct local bus to PCI interrupts
29	MASTER_PI	PCI Master Local Interrupt Enable: (see V3 datasheet)
28	SLAVE_PI	PCI Slave Local Interrupt Enable: (see V3 datasheet)
27	OUT-POST	I2O Outbound Post List Not Empty: (see V3 datasheet)

26	-	Reserved
25	DMA1	DMA Channel 1 interrupt enable
24	DMA0	DMA Channel 0 interrupt enable
23-22	MODE_D	INTD interrupt mode: Determines use of corresponding interrupt pin <table><tr><td>00</td><td>Active low level triggered input</td></tr><tr><td>01</td><td>High-to-low edge triggered input</td></tr><tr><td>10</td><td>Software cleared output. INTD pin is asserted via an interrupt event and cleared through PCI_INT_STAT register</td></tr><tr><td>11</td><td>Reserved</td></tr></table>
21-20	MODE_C	INTC interrupt mode (as MODE_D)
19-18	MODE_B	INTB interrupt mode (as MODE_D)
17-16	MODE_A	INTA interrupt mode (as MODE_D)
15	INTD_TO_LB	1=INTD will request LICU interrupts when the input is active 0=INTD will never request LICU interrupts
14	INTC_TO_D	When set INTC will act as interrupt request for INTD output
13	INTB_TO_D	When set INTB will act as interrupt request for INTD output
12	INTA_TO_D	When set INTA will act as interrupt request for INTD output
11	INTD_TO_C	When set INTD will act as interrupt request for INTC output
10	INTC_TO_LB	1=INTC will request LICU interrupts when the input is active 0=INTC will never request LICU interrupts
9	INTB_TO_C	When set INTB will act as interrupt request for INTC output
8	INTA_TO_C	When set INTA will act as interrupt request for INTC output
7	INTD_TO_B	When set INTD will act as interrupt request for INTB output
6	INTC_TO_B	When set INTC will act as interrupt request for INTB output

5	INTB_TO_LB	1=INTB will request LICU interrupts when the input is active
		0=INTB will never request LICU interrupts
4	INTA_TO_B	When set INTA will act as interrupt request for INTB output
3	INTD_TO_A	When set INTD will act as interrupt request for INTA output
2	INTC_TO_A	When set INTC will act as interrupt request for INTA output
1	INTB_TO_A	When set INTB will act as interrupt request for INTA output
0	INTA_TO_LB	1=INTA will request LICU interrupts when the input is active
		0=INTA will never request LICU interrupts

Table 10 : PCI Interrupt Configuration Register

13.3.2 PCI Interrupt Status Register(Offset 0x48, BAR0)

Bits	Name	Description
31	MAILBOX	Mailbox Interrupt:
		1=Mailbox interrupt request active
		0=No mailbox interrupts pending
		Cleared by clearing MAIL_RD_STAT and MAIL_WR_STAT
30	LOCAL	Local bus direct interrupt:
		1=Local bus master requests a PCI interrupt
		0=No operation
		This bit is set by writing 1 and cleared by writing 0
29-28	-	Reserved
27	OUT_POST	I2O outbound post list not empty: (see V3 datasheet)
26	-	Reserved
25	DMA1	DMA channel 1 interrupt
24	DMA0	DMA channel 0 interrupt
23-15	-	Reserved
14	INTC_TO_D	INTD output from INTC input: when set (1) an interrupt has

		occurred on INTC
13	INTB_TO_D	INTD output from INTB input: when set (1) an interrupt has occurred on INTB
12	INTA_TO_D	INTD output from INTA input: when set (1) an interrupt has occurred on INTA
11	INTD_TO_C	INTC output from INTD input: when set (1) an interrupt has occurred on INTD
10	-	Reserved
9	INTB_TO_C	INTC output from INTB input: when set (1) an interrupt has occurred on INTB
8	INTA_TO_C	INTC output from INTA input: when set (1) an interrupt has occurred on INTA
7	INTD_TO_B	INTB output from INTD input: when set (1) an interrupt has occurred on INTD
6	INTC_TO_B	INTB output from INTC input: when set (1) an interrupt has occurred on INTC
5	-	Reserved
4	INTA_TO_B	INTB output from INTA input: when set (1) an interrupt has occurred on INTA
3	INTD_TO_A	INTA output from INTD input: when set (1) an interrupt has occurred on INTD
2	INTC_TO_A	INTA output from INTC input: when set (1) an interrupt has occurred on INTC
1	INTB_TO_A	INTA output from INTB input: when set (1) an interrupt has occurred on INTB
0	-	Reserved

Table 11 : PCI Interrupt Status Register

13.3.3 Local Bus Interrupt Mask Register(Offset 0x77, BAR0)

Bits	Name	Description
7	MAILBOX	Global mailbox interrupt enable
6	PCI_RD	PCI read error interrupt enable
5	PCI_WR	PCI write error interrupt enable
4	PCI_INT	Global PCI interrupt to local interrupt enable
3	PCI_PERR	PCI parity error interrupt enable
2	I2O_QWR	I2O inbound post queue write interrupt enable

1	DMA1	DMA channel 1 interrupt enable
0	DMA0	DMA channel 0 interrupt enable

Table 12 : Local Bus Interrupt Mask Register

13.3.4 Local Bus Interrupt Status Register(Offset 0x76, BAR0)

Bits	Name	Description
7	MAILBOX	1=interrupt has been requested by one or more of the mailbox registers 0=no mailbox interrupts pending
6	PCI_RD	See V3 datasheet
5	PCI_WR	See V3 datasheet
4	PCI_INT	See V3 datasheet
3	PCI_PERR	See V3 datasheet
2	I2O_QWR	See V3 datasheet
1	DMA1	See V3 datasheet
0	DMA0	See V3 datasheet

Table 13 : Local Bus Interrupt Status Register

13.3.5 PCI Mailbox WRITE/READ Interrupt Control Register(Offset: Write 0xD0, BAR0 Read 0xD2, BAR0)

Bits	Name	Description
15	EN15	Local interrupts on PCI bus writes/reads to mailbox15 enable
14	EN14	Same as above for mailbox 14
13	EN13	Same as above for mailbox 13
12	EN12	Same as above for mailbox 12
11	EN11	Same as above for mailbox 11
10	EN10	Same as above for mailbox 10
9	EN9	Same as above for mailbox 9
8	EN8	Same as above for mailbox 8
7	EN7	Same as above for mailbox 7
6	EN6	Same as above for mailbox 6
5	EN5	Same as above for mailbox 5
4	EN4	Same as above for mailbox 4

3	EN3	Same as above for mailbox 3
2	EN2	Same as above for mailbox 2
1	EN1	Same as above for mailbox 1
0	EN0	Same as above for mailbox 0

Table 14 : PCI Mailbox WRITE/READ Interrupt Control Register

13.3.6 Local Bus Mailbox WRITE/READ Interrupt Control Register(Offset: Write 0xD4, BAR0 Read 0xD6, BAR0)

Bits	Name	Description
15	EN15	PCI interrupts on Local bus writes/reads to mailbox15 enable
14	EN14	Same as above for mailbox 14
13	EN13	Same as above for mailbox 13
12	EN12	Same as above for mailbox 12
11	EN11	Same as above for mailbox 11
10	EN10	Same as above for mailbox 10
9	EN9	Same as above for mailbox 9
8	EN8	Same as above for mailbox 8
7	EN7	Same as above for mailbox 7
6	EN6	Same as above for mailbox 6
5	EN5	Same as above for mailbox 5
4	EN4	Same as above for mailbox 4
3	EN3	Same as above for mailbox 3
2	EN2	Same as above for mailbox 2
1	EN1	Same as above for mailbox 1
0	EN0	Same as above for mailbox 0

Table 15 : Local Bus Mailbox WRITE/READ Interrupt Control Register

13.3.7 Mailbox Write/Read Interrupt Status Register(Offset: Write 0xD8, BAR0 Read 0xDA, BAR0)

Bits	Name	Description
15	EN15	1=Mailbox 15 has requested a PCI or Local write/read interrupt
		0=Mailbox 15 has not requested a PCI or Local write/read interrupt
14	EN14	Same as above for mailbox 14
13	EN13	Same as above for mailbox 13
12	EN12	Same as above for mailbox 12
11	EN11	Same as above for mailbox 11
10	EN10	Same as above for mailbox 10
9	EN9	Same as above for mailbox 9
8	EN8	Same as above for mailbox 8
7	EN7	Same as above for mailbox 7
6	EN6	Same as above for mailbox 6
5	EN5	Same as above for mailbox 5
4	EN4	Same as above for mailbox 4
3	EN3	Same as above for mailbox 3
2	EN2	Same as above for mailbox 2
1	EN1	Same as above for mailbox 1
0	EN0	Same as above for mailbox 0

Register cleared by writing 1, writing 0 has no effect

Table 16 : Mailbox Write/Read Interrupt Status Register

13.3.8 INTREG Register(Offset 0x40, BAR1)

Bits	Name	Description
15	-	Reserved
14	-	Reserved
13	-	Reserved
12	-	Reserved
11	-	Reserved

10	-	Reserved
9	-	Reserved
8	-	Reserved
7	-	Reserved
6	-	Reserved
5	-	Reserved
4	-	Reserved
3	-	Reserved
2	IIOF2EN	PC to DSP TIMIIOF2 interrupt enable
1	IIOF1EN	PC to DSP TIMIIOF1 interrupt enable
0	IIOF0EN	PC to DSP TIMIIOF0 interrupt enable

Table 17 : INTREG Register

13.4 Example

The example below shows how the DSP can cause an interrupt on the PC by writing to mailbox register 0.

The PC must first enable the interrupts, to do this the following register bits must be altered, and an interrupt thread handler needs to be created.

PCI Interrupt Configuration Register(Offset 0x4C, BAR0) – bit 31 must be set.

Local Bus Interrupt Mask Register(Offset 0x77, BAR0) – bit 7 must be set.

PCI Mailbox WRITE/READ Interrupt Control Register(Offset: Write 0xD0, BAR0 Read 0xD2, BAR0) – bit 0 of the write register (0xD4) must be set.*

An Interrupt service routine must be set up, in this the following register will need to be cleared.

Mailbox Write/Read Interrupt Status Register(Offset: Write 0xD8, BAR0 Read 0xDA, BAR0) – bit 0 of the write register (0xD8) must be cleared.**

Local Bus Interrupt Status Register(Offset 0x76, BAR0) – bit 7 must be cleared.

To cause the interrupt the DSP needs to write to the mailbox register in the V^3 chip, this is done by writing to address 0x1C0000C0 (this will write to the first four mailboxes).

*These are two separate registers, one to enable interrupts on reads from the mailbox registers the other to enable interrupts on writes to the mailbox registers.

**These are two separate registers, one shows interrupt status for reads from the mailbox register the other to show interrupt status on writes to the mailbox registers.

14 Stand-alone mode

For the SMT130 to operate in stand-alone mode Jumper J9 (**Figure 8 : Jumper Finder Diagram**) must be installed and the Auxiliary power header (J8) connected. The connector type and pin out is compatible with the floppy drive power connector in on a PC power supply.

15 Performance Figures

Following are the performance figures for the SMT130. Performance figures are stated for the SMT130 with the Rev. **A1** V^3 PCI bridging device fitted. Further performance figures will be issued as faster V^3 PCI bridging devices become available and are fitted to the SMT130.

Transfer type	Speed in Mbytes/second	Comments
Standard read from Global bus	TBA	
Standard write to Global bus	TBA	
DMA read from Global bus	TBA	
DMA write to Global bus	TBA	
DMA read from shared memory	75-100	Will Vary depending on PCI traffic
DMA write to shared memory	75-100	Will Vary depending on PCI traffic
DMA read from SDB	75-100	Will Vary depending on PCI traffic
DMA write to SDB	75-100	Will Vary depending on PCI traffic
DMA read from Comport	TBA	
DMA write to Comport	TBA	

16 Mechanical Dimensions

The SMT130 size is 95.8 mm by 115.5 mm. The SMT130 conforms to the physical specification defined by the PC/104 Consortium. The inter stack height may be violated if the SMT130 is populated with a "tall" TIM. In this case "connector extensions" and longer fixing pillars maybe used.

17 Power consumption

The SMT130 is powered from the PCI stack connector on the host PC controller. The card uses the 5v supply only. The following current consumption figures were measured using a LEM current clamp during a quiescent period. No TIM was fitted to the SMT130 during this test

Current drawn from 5v supply   :   200mA

18 Cables and Connectors

18.1 Comports

The cables used with FMS connectors are not supplied with the SMT130. You can order them separately from Sundance with part number SMT500-FMSxx, where xx is the cable length in centimetres.
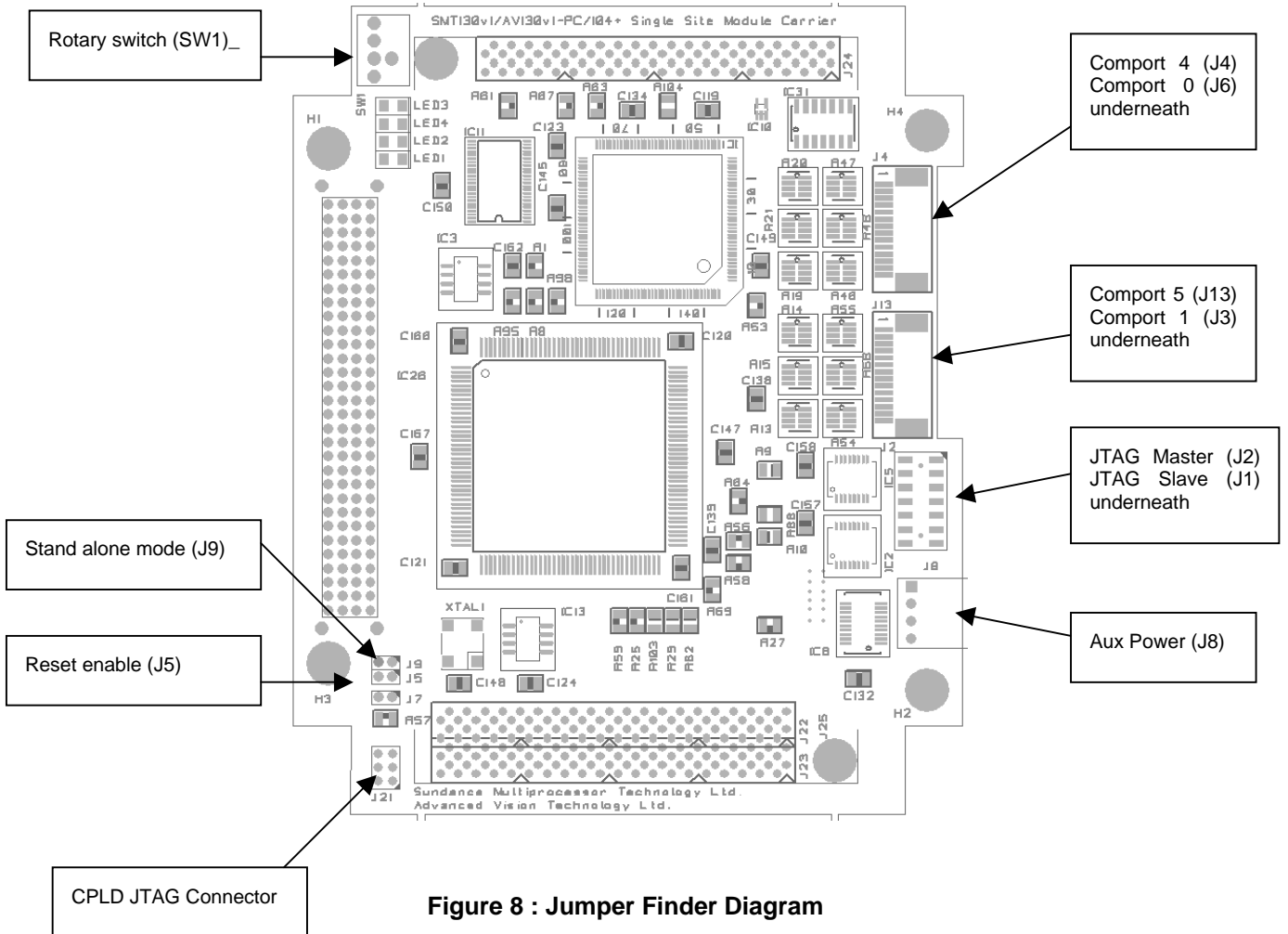
19 Where's that Jumper?

Below is a diagram to help locate the jumpers:

Figure 8 : Jumper Finder Diagram