WS 3112
PCI SHARC Cluster

WIESE
signal
verarbeitung

1. Contents

2. Tables and Pictures

3. Manual Revision History

Version	Date	Name	Comments
1.0	06.08.96	kl	First edition (Preliminary)
1.1	20.08.96	kl	added tables
1.2	10.09.96	kl	added software remarks
1.3	11.12.96	kl	added mastermode comments
1.4	06.02.97	kl	added special operating conditions
1.5	18.07.97	te	reformated, contact information updated
1.6	28.10.97	te	added description for /SBTS control in the EPLD register (chapter 6.10)

4. Glossary

There are some terms used in this manual, that perhaps have to be explained:

Term	Definition
PCI	a new bus-standard for different computer-architectures, introduced for high speed processing. On the DSP board there is a standard PCI device, similar to a bridge, that isolates the onboard DSP bus from the host system PCI bus.
DSP	a special CPU-architechture, called digital signal processor, optimized for high speed signal processing as speech, sound, graphics and imaging applications
SHARC	A special DSP-architecture, named „super harvard architecture computer“
ADSP - 2106x SHARC	High performance 32 bit digital signal processor with the 21000 family core
Host	The slot, that has been used for the DSP board, belongs to a computer system. This computer is called the „host“ system.
Local Bus, Local RAM etc.	The term „local“ is related to the DSP board, and means devices on the board, that are all isolated from the PCI bus by a PCI bridge.
Slave Mode	In the slave mode the DSP board is the slave, and the host controls data transfers to or from the board.
Master Mode	In the master mode the DSP board initiates a data transfer by itself, independent of the host CPU. In reality, and this is true for all modern operating systems, it is important, that the host first supplies memory block start addresses, before the DSP board can execute master mode transfers.
Demand Mode DMA	The DMA demand mode is requested and granted by two hardware wires, if enabled in this mode (handshake mode).
Chaining Mode DMA	The DMA Chaining mode uses descriptor-tables to setup the next DMA-Block immediate after finishing the previous one, endless block transfers (in a loop) are possible. Chaining DMA is good for copying data blocks to/from fragmented memory.
Bursts	Bursts are transfered data blocks with a leading address. After the leading address there are only data cycles with an implicit address increment, so the transfer efficiency is better if using long bursts.
EPLD	Electric Programmable Logic Device, pure hardware logic.
DWORD	Double Word, 32 bits wide. Basic data type for the DSP.
XDWORD	Extended Double Word, dependent on SHARC mode, 40 bits or 48 bits wide.

5. Getting Started

Please read the manual carefully before you install the board in your computer. We are sure that you may find there some information you don' have up to now!

5.1 Host Requirements

- **Empty PCI slot**
- **Power:** 5 V, ca. 2A (WS 3112, no SHARC IO-Packs installed)
- **Ambient temperature:** max. 45 °C
- **PCI BIOS**: compatible to PCI Rev. 2.0 or higher
- **Free address space**: min. 65 Mbytes on the PCI bus (64 Mbytes continuous). It is not necessary to have RAM at this locations but the WS 3112 Board only needs a ***free memory window*** of this size
- **CPU:** i386 or higher
- **DRAM**: min. 4 Mbytes
- **Hard disk**: free storage ca. 5 Mbytes for driver and testing software
- **Operating System:** MSDOS 5.0 or higher, MS Windows 3.1x, Windows 95 or Windows NT (depending on the software drivers you are using)

5.2 Unpacking

The WS 3112 Board is a mechanic and ***electrostatic sensible*** CMOS-Device. Before unpacking the board, be sure to discharge yourself at the computers case. Don't ***touch*** the PCI connector. Don't use ***mechanical force*** to install the board into your computer. Don't ***torse*** the board in any direction!

5.3 Dipswitch and Jumper Settings

On the board you can find one dipswitch and some connectors. ***The factoy setting is suitable in most of the cases***. Because of the PCI concept you don't need to worry about memory addresses, I/O addresses or interrupts. The PCI BIOS does the job (normally) for you. The connectors and the installed jumpers are only for testing purpose, and ***need not to be changed***.

In the following **picture 1** you can find the position of the in **table 1** (see below) described components.

Picture 1: **Top View of WS 3112 Board with Connectors, Jumpers, DIP-Switch**

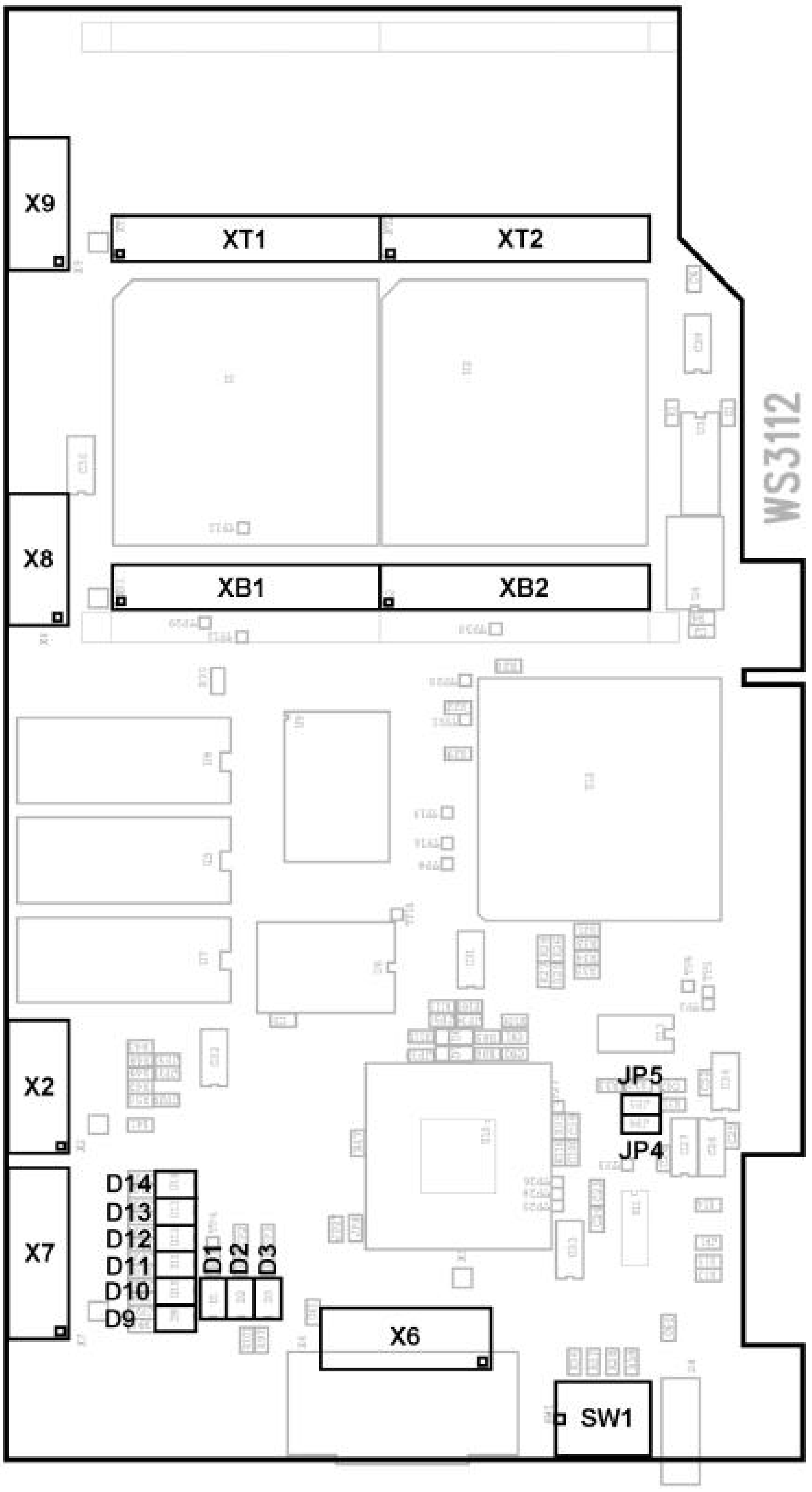


Table 1**: Description of Connectors and Jumper settings**

Device	**Position or Type**	**Default Factory Setting**	**Non-Standard Setting**	**Described in Chapter**	**Comment**
SW 1	DIP			6.10	DIP-Switch: **Bootmode and RPBA** (ON = 0, OFF = 1), selected by default: Host Boot Mode (see table 5, 6)
	SW 1-1	ON			BMS
	SW 1-2	ON			LBOOT
	SW 1-3	ON			EBOOT
	SW 1-4	ON			RPBA
JP 4	Jumper				Jumper: **PCI resource allocation**
	JP 4-1/2	OFF			PCI default resources
	JP 4-1/2		ON		do ***not*** use
JP 5	Jumper			6.12	Jumper: **EEPROM write mode**
	JP 5-1/2	OFF			EEPROM write disabled
	JP 5-1/2		ON		EEPROM write enabled
LED 1	green			6.13	- 12 V power OK
LED 2	green			6.13	+ 12 V power OK
LED 3	green			6.13	+ 5 V power OK
LED 9	red			6.13	Host access to an unknown local device
LED 10	red			6.13	The DIP-Switch settings for boot mode are not valid, board control register defined boot mode
LED 11	red			6.13	Accesses to/from host and/or SHARC
LED 12	red			6.13	Direct master write FIFO is almost full
LED 13	red			6.13	Pending interrupt on PCI or SHARC side
LED 14	red			6.13	Pending DMA request (PCI-Controller or SHARC)
XB1	Molex			6.14	Connector: SHARC 1 IO-Pack Bus
XB2	Molex			6.14	Connector: SHARC 2 IO-Pack Bus
XT1	Molex			6.14	Connector: SHARC 1 IO-Pack Links
XT2	Molex			6.14	Connector: SHARC 2 IO-Pack Links
X6	Header 2x7			7	Connector: SHARC ICE (JTAG), if connected to ICE: remove Jumpers, otherwise don’t remove Jumpers!
	X 6-1/2	no JP			
	X 6-3/4	no JP			
	X 6-5/6	no JP			
	X 6-7/8	JP set			Jumper Set (TCK)
	X 6-9/10	JP set			Jumper Set (/TRST)
	X 6-11/12	no JP			
	X 6-13/14	no JP			
X2		none	do not use!		Connector: reserved
X7		none	do not use!		Connector: reserved
X8		none	do not use!		Connector: reserved
X9		none	do not use!		Connector: reserved

5.4 Installing the DSP Board

- Be sure the computer’s power is turned off, and it’s power cable is disconnected.
- Remove the cover of your computer. Refer to the documentation delivered with the computer
- Ensure that the dipswitch and jumper settings are as described (equal to the factory setup). No special setup to your computer’s requirements is necessary, because of the PCI automatic resource allocation concept.
- Select an empty expansion PCI slot for the WS 3112 board. Do not use a wrong slot (ISA etc.). Try to find a slot where no collision with other devices occurs, e.g. CPU, and where cooling from fan is OK.
- Remove the expansion slot cover
- Insert carefully the WS 3112 board, and fasten the retaining bracket with the screw from the slot cover
- Close the cover of your computer

5.5 Installing the Software

The installation runs under Windows. Use the program manager to start the file „A:SETUP.EXE“. After installation please read at first the file „README.TXT“ or „README.WRI“, where you can find additional information on using the software package.

5.6 Testing the DSP Board

If you have done all the steps before please verify that the software works with the WS3112 board. Please read the file „README.WRI“ for information how to start the test software!

6. Technical Description

The WS 3112 PCI SHARC cluster board is a standard PC AT PCI board with 2 ADSP 2106x. The features are:

1. 2 SHARC ADSP 21060 or ADSP 21062 share one local multiprocessor bus. The features of the ADSP 2106x family are:

   - 32-Bit IEEE and extended 40-Bit IEEE Floating-Point-Unit, Multiplier, ALU, Shifter
   - 16 Data Register Files (2 Sets, 40 Bit)
   - Data Address Generators (DAG1, DAG2)
   - Program Sequencer with Instruction Cache (32 x 48 Bit)
   - Interval Timer
   - Dual-Ported SRAM (21060: 4 Mbit, 21062: 2 MBit)
   - External Port for Interfacing to Off-Chip Memory & Periphals
   - Host Port & Multiprocessor Interface
   - DMA Controller (10 Channels)
   - 2 Serial Ports (max. 40 Mbit/s, Word Length 3 ... 32 Bits)
   - 6 Link Ports (4 Bits wide, max. 240 Mbyte/s)
   - IEEE 1149.1 JTAG Test Access Port

2. Up to 256K x 48 Bit static memory on board. The memory is connected to the multiprocessor bus and can be accessed by every DSP and the host.

3. PCI Controller with 32 bit host interface to the local DSP bus. The controller seperates the PCI bus from the local DSP bus.

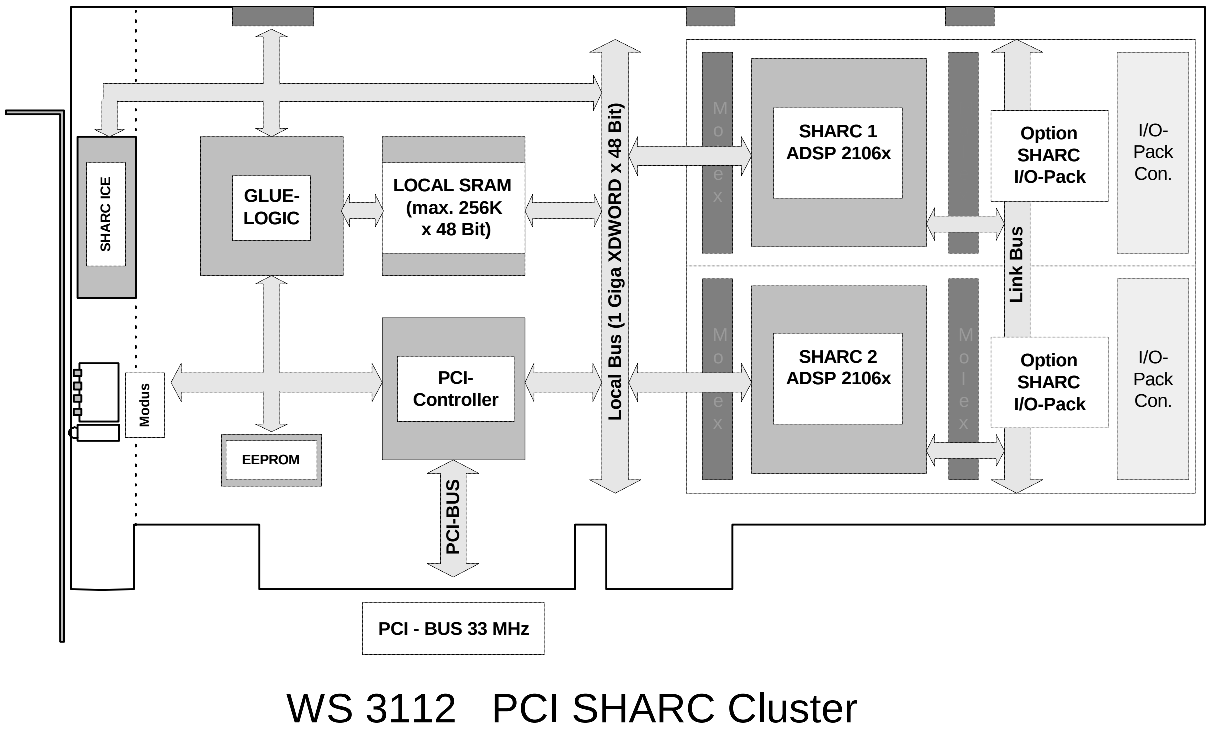
   - FIFO’s in both directions, each 4 DWORDs long
   - 2 DMA Channels for Demand Mode and Chaining Mode Transfers in Master- and Slavemode
   - Master Mode Data Transfer
   - Slave Mode Data Transfer
   - Burst Data Tranfers
   - 8 Mailbox-Register 32 bit bidirectional
   - 2 Doorbell-Register 32 bit

4. All I/O resources of each SHARC (as links, sports etc.) and part of the local parallel SHARC bus are supplied to MOLEX-Connectors (see the appendix for details). These connectors are featuring plug-in modules named „**SHARC IO-Packs**“, that are also available from Fa. Wiese Signalverarbeitung GmbH. **SHARC IO-Pack** is an open interface standard for interfacing user-dependent hardware to the board.

5. For software debugging a JTAG connector is implemented. The EZ-ICE debugger from AD can be connected.

6.1 Block - Diagram

Picture 2**: Block-Diagram of the WS 3112 Board**



In the block diagram above you can identify the different onboard devices of the WS 3112 DSP board. There are two complex devices: the PCI-Controller and the SHARC signal processors. The ***register set description*** of these devices ***is not part of this manual***, so please refer to the literature list in chapter 12.

The local parallel multiprocessor bus is isolated from the PCI bus by the PCI-Controller. All local devices are connected to the multiprocessor bus, with the exception of the EEPROM, that belongs to the PCI-Controller. The glue logic (EPLD) controls the interaction between the devices.

All I/O-signals of the SHARC's are accessible through the SHARC IO-Pack connectors. Fa. Wiese offers SHARC IO-Pack moduls to interface the board to external data sources and/or sinks.

6.2 Local Bus (SHARC Bus)

The SHARC's are internally organized with DWORD or XDWORD data (eXtended Double WORD data: an address-increment of one accesses the next 32, 40, or 48 bit data), and the SHARC addressbus is a 32-bit wide parallel bus in principle, so the ***local address bus*** is hardwired to the PCI-controller with an ***address-translation or shift of two*** (PCI **LA2** to SHARC **SA0**, PCI **LA3** to SHARC **SA1**, etc).

The address lines **SA23** ... **SA19** and **SA10** ... **SA8** are used in an EPLD to decode the accesses from host and local side. See the address mapping tables below for additional information.

The ***local data bus*** is 48 bit wide (**SD0** ... **SD47**), and represents the SHARC data path widths (32 bit data on **SD16** ... **SD47**, 40 and 48 bit data on **SD0** ... **SD47**. The data lines **SD16** ... **SD47** are connected to the PCI-Controller data lines **LD0** ... **LD31**. So the normal data width between SHARC and PCI transfers is 32 bits. There are some special packing modes defined via SHARC register commands to transfer other data to/from PCI host.

The ***local SRAM*** is directly connected to the local bus and is 48 bits wide, so every SRAM data line is connected to the corresponding SHARC data line. To access the SRAM directly from host, the SRAM has been logically splitted into two pages, a 32 bits wide page connected to **SD16** ... **SD47**, and a 16 bits wide page connected to **SD0** ... **SD15**. This paging is only visible from the host side, the SHARC's can see all the time the full 48 bits wide SRAM.

6.3 Host Access (Slave Mode)

The host accesses the local onboard resources through a PCI-Controller and some additional onboard glue logic, called host interface. This interface shares the local bus with the SHARC's and other local devices. The host interface has ***highest priority*** in the local system, and so the host has a transparent access to all local bus resources (SHARC's, SRAM, PIGGY-Backs). Transparent means, that at each time the host can arbitrate the local bus and can make data transfers for downloading programs or for debugging or application-dependent purposes.

6.3.1 Host Address Space and Mapping Scheme

Each WS 3112 Board uses a local ***64 Mbyte*** address space window in the 4 Gbyte PCI address space. The local resources are memory-mapped with an individual offset to a base address. The base addresses and offsets are ***different*** from point-of-view, that means, the ***host*** can see the local devices at another address space than the ***local processors*** (the SHARC's)!

The ***address-space and/or mapping*** on the host side ***is not hardwired***. The DSP board has an address space requirement, that is supplied to special PCI-registers. At poweron time the host system BIOS reads the requirements, and the ***BIOS does the mapping*** to a base-address, that is dependent to the requirements of the whole host system, and so the resulting base-address for the DSP-Board is ***system-, slot-, and board-dependent***. To get the correct base address of the board, it is necessary to read the PCI register of the onboard bridge. The DDL software driver, that has been delivered with every board, gives you an easy way to access the board (see chapter 9).

From PCI point-of-view there is a ***hardwired alignment to DWORD***. A DWORD access with an unaligned address from PCI side (that means ***slave*** mode, ***not*** master mode) would result always in correct data. On the other hand, WORD or BYTE accesses are problematic, because each unaligned write access (not DWORD aligned) accesses the same SHARC address.

In **slave mode** only DWORD **accesses** should be used, and the best way is to switch on DWORD or QWORD **alignment** in the compiler, if you want to write your own driver. In **master mode** it is absolute necessary to transfer always DWORD aligned data. Another good and time saving solution is to use our available **software driver package**, that manages all those problems, please refer to chapter 9 of this manual!

Table 2**: Address Map for Board Devices from Host (PCI) side Point of View**

PCI Startaddress	**PCI Endaddress**	**Comment (Host Slave Accesses to...)**
xx 00 0000	xx 07 FFFC	SHARC IOP, internal (access not allowed)
xx 08 0000	xx 0F FFFC	SHARC normal word address (access not allowed)
xx 10 0000	xx 1F FFFC	SHARC short word address (access not allowed)
xx 20 0000	xx 3F FFFC	SHARC 1 Multi-Space
xx 40 0000	xx 5F FFFC	SHARC 2 Multi-Space
xx 60 0000	xx 7F FFFC	reserved (SHARC 3)
xx 80 0000	xx 9F FFFC	reserved (SHARC 4)
xx A0 0000	xx BF FFFC	reserved (SHARC 5)
xx C0 0000	xx DF FFFC	reserved (SHARC 6)
xx E0 0000	xx FF FFFC	SHARC Broadcast (only writes)
x1 00 0000	x1 00 03FC	reserved
x1 00 0400	x1 00 07FC	PIGGY-BACK SHARC 1
x1 00 0800	x1 00 0BFC	PIGGY-BACK SHARC 2
x1 00 0C00	x1 00 0FFC	reserved (PIGGY-BACK SHARC 3)
x1 00 1000	x1 00 13FC	reserved (PIGGY-BACK SHARC 4)
x1 00 1400	x1 00 17FC	reserved (PIGGY-BACK SHARC 5)
x1 00 1800	x1 00 1BFC	reserved (PIGGY-BACK SHARC 6)
x1 00 1C00	x1 00 1FFC	PIGGY-BACK Broadcast (only writes)
x1 20 0000	x1 3F FFFC	EPLD Board Control Register
x1 40 0000	x1 5F FFFC	PLX (PCI 9060) Local Runtime DMA Registers
x1 60 0000	x1 7F FFFC	reserved
x1 80 0000	x1 BF FFFC	Ext. SRAM 32 Bit DWORD (D16 ... D48)
x1 C0 0000	x1 FF FFFC	Ext. SRAM 16 Bit extended WORD (D00 ... D15)
x2 00 0000	x3 FF FFFC	reserved (for SRAM-Extention)

One WS 3112 board occupies ***64 Mbyte*** in the 4 Gbyte PCI address space, so the most significant 6 bits of a 32 bit address are not used for decoding. The maximum usable number of WS 3112 DSP boards is greater ***60*** for a host system.

6.4 SHARC Access (Master Mode and Local Mode)

The SHARC accesses to the PCI side are so called ***'master mode'*** accesses. The master mode enables the local processors (SHARC's) to ***directly*** access resources as a master on the PCI bus, without intervention or interruption of the host CPU; this mode is a fast method to transfer results to the host system. There is only one restriction in modern operating systems: the local processor has to request first e.g. a memory area on the PCI side, before starting transfers.

6.4.1 Local Address Space and Mapping Scheme

From the SHARC point of view there exists ***another*** mapping scheme than from PCI side. The SHARC's can access up to 1 Giga XDWORDs or 4 GBytes (XDWORD means „extended DWORD“, a value greater than 32 bit and lower than 64 bits, e.g. 48 bit) of their address space of 4 Giga XDWORD words, because of only using **SA0** ... **SA29** of the address lines.

Because of the PCI-controller is the only device, that is connected to the local addressbus with an ***address-translation of two*** (PCI **LA2** to SHARC **SA0**, PCI **LA3** to SHARC **SA1**, etc), only the registers of the PCI-Controller and all addresses „on the other side“ (master mode to the PCI-Bus) are affected by this address-translation.

All other local devices (SHARC's, SRAM, SHARC IO-Packs) are connected directly (with ***no address-shift***) to the local addressbus, and so no address-translation takes place.

For the SHARC cluster the (hardwired) mapping of Analog Devices is valid (see the following ***local addressmap*** and the AD 2106x manual). All other local devices are accessed via **/MS0** ... **/MS3 lines**. The mapping of these memory select lines is ***dependent*** on a SHARC register (SYSCON), so this register must be loaded with the correct mapping values before addressing other devices from SHARC side. The value loaded to SYSCON is only a default value and for this mapping value the addressmap below is valid, but if you use this mapping then there are no restrictions in addressing devices. The ***default value*** in the DDL library is 15 for MSIZE, and the resulting mapping space is 256 Mega XDWORDs for each bank.

When the master mode is used and an access to host side is made (by using the **/MS2** line), it is important to use an address window, that matches two PCI-Controller registers („Local Range Register for Direct Master to PCI“ and „Local Bus Base Address Register for Direct Master to PCI Memory“). The PCI Controller translates the **/MS2 -** access to another address, that has been loaded into a third PCI Controller register („PCI Base Address (Rename) Register for Direct Master to PCI“). Have a look for detailed information at the PCI 9060 manual from PLX (see literatur list).

Our DDL (Software Driver Library) helps you doing the job (doing all the hardware setup procedures and supplying data transfer procedures), please refer to the Software Driver Description in chapter 9. The usable default master mode window (DDL, host side) is 64 Mbytes, and may be remapped anywhere in the host address space.

Table 3: **Address Map for Board Devices from SHARC Point of View (with MSIZE = 15)**

SHARC Startaddress	**SHARC Endaddress**	**/MSx-Line**	**Comment (Local or Master Accesses to...)**
00 00 0000	00 01 FFFF		SHARC IOP, internal (access not allowed)
00 02 0000	00 03 FFFF		SHARC normal word address (access not allowed)
00 04 0000	00 07 FFFF		SHARC short word address (access not allowed)
00 08 0000	00 0F FFFF		SHARC 1 Multi-Space
00 10 0000	00 17 FFFF		SHARC 2 Multi-Space
00 18 0000	00 1F FFFF		reserved (SHARC 3)
00 20 0000	00 27 FFFF		reserved (SHARC 4)
00 28 0000	00 2F FFFF		reserved (SHARC 5)
00 30 0000	00 37 FFFF		reserved (SHARC 6)
00 38 0000	00 3F FFFF		SHARC Broadcast (only writes)
00 40 0000	10 3F FFFF	/MS0	Ext. SRAM (48 Bit wide), Startaddress is fix
10 40 0000	10 40 00FF	/MS1	SHARC I/O-Pack 1 (Slot SHARC 1)
10 40 0100	10 40 01FF	/MS1	SHARC I/O-Pack 2 (Slot SHARC 2)
10 40 0200	10 40 02FF	/MS1	reserved
10 40 0300	10 40 03FF	/MS1	reserved
10 40 0400	10 40 07FF	/MS1	PLX (PCI 9060) Local Runtime Registers
20 40 0000	30 3F FFFF	/MS2	PCI Addresses (1 M XDWORDs or 4 Mbytes)
30 40 0000	40 3F FFFF	/MS3	reserved (for SRAM Extention)

Note: all above address values, that are using a /MSx line, are ***only valid*** with the default value of 15 loaded to **MSIZE** of the **SYSCON** register (Bit 15=1, Bit 14=1, Bit 13=1, Bit 12=1). With this default mapping value the size of SHARC BANK 0 ... BANK 3 is each 256 M XDWORDs. The offset of 0x00400000 for /MS0 ... /MS3 is a fixed value that cannot be changed (only SHARC addressmap dependent).

6.5 Message Handling

In a multiprocessor system with different buses it is a good idea to handle requests to 'the other' system with a message concept. The PCI-Controller acts like a bridge beetween the two buses and in the controller there are eight multipurpose 32 bit wide so called mailbox-registers available for exchanging informations between the two processor-systems. The mailbox-register have the same functionality as true dual-port-RAMs, with the benefit of accessing the registers from each side ***without the need of gaining control over the „other" system bus***.

Each SHARC has also a set of message-registers, so they may also be used for message-passing. The only difference to the preceding paragraph is, that first the PCI host must gain control to the local bus before accessing a SHARC message register.

6.6 Interrupt Handling

The ***interrupt handling*** is devided in two groups. On the PCI side it is possible to generate a host interrupt and on the local side (on the DSP board) it is possible to generate a SHARC interrupt. So called **'doorbell'-register** (32 bits wide) reside in the PCI-Controller to handle interrupt services in both directions.

6.6.1 PCI Interrupts

At poweron time the BIOS supplies to the WS 3112 board an PCI-Interrupt channel (**INT A**). To this channel you normally must define a hardware interrupt (often IRQ10 ... IRQ15, typically some of them you can select in the BIOS) in the ***setup of the host BIOS***, and you must enable ***the corresponding PCI slot*** for IRQ handling in the host BIOS.

If you have done this, it is possible to generate a PCI interrupt by writing from SHARC to the ***local doorbell interrupt register*** of the PCI-Controller. The host reads this register during interrupt service, resets the corresponding bit and handles the interrupt.

Another way to generate a PCI interrupt is to use the FLAG 0 feature (please refer to SHARC Flags).

6.6.2 SHARC Interrupts

On the local side of the DSP-Board exists a similar mechanism. The host may generate a SHARC interrupt (**IRQ0**) by writing a value to the ***PCI doorbell interrupt register***. The SHARC reads this register during interrupt service, resets the corresponding bit and handles the interrupt.

/SHA_IRQ0: connected to the local side of the PCI doorbell register of the PCI-Controller

/SHA_IRQ1: not used, routed to all SHARC IO-Packs connectors

/SHA_IRQ2: not used, routed to all SHARC IO-Packs connectors

6.7 DMA

There are two DMA-Devices available on the DSP Board. The PCI-Controller has two DMA channels (hardware and software controlled), and each SHARC has ten DMA channels, with two of them hardware (Demand mode DMA) and/or software controlled.

The DMA hardware request lines **/SHA_DMAR1** of the SHARC's and the line **/PLX_DREQ0** of the PCI-Controller are tied together, and routed to the SHARC I/O-Connector **XB n**, and the same for the lines **/SHA_DMAR2** and **/PLX_DREQ1**.

The DMA hardware grant lines **/SHA_DMAG1** of the SHARC's are 'wired or', and routed to the SHARC I/O-Connector **XB n**, and the same is valid for the lines **/SHA_DMAG2**.

The SHARC IO-Packs may request a SHARC or PCI-Controller DMA transfer by asserting one of the two DMA request lines, if enabled in the corresponding device before.

Under a modern operating system the main memory is typically fragmented by a MMU into blocks. The usage of ***chaining DMA mode*** is a good way to efficiently read or write a large data block to or from host memory, because the fragmented memory structure may be eliminated by the chaining method. Twodimensional DMA may be implemented by setting up each chain link to a picture line.

6.7.1 PCI - Controller DMA

The two DMA - channels of the PCI-Controller may be used for both directions, slave mode or master mode transfers. The channels are indepent from software controlled transfers, have their own FIFO's and both have the feature of chaining mode DMA.

6.7.2 SHARC DMA

The ten SHARC DMA-Channels are grouped into 'internal' and 'external' channels. Two of the four 'external' channels may be used in demand mode and four in software controlled mode (also chaining mode), and are capable of transferring data to or from PCI side.

6.8 SHARC Flags

FLAG0: the lines of all SHARC's are ***'ored' together*** and routed to the interrupt input pin of the PCI-Controller. So it is possible to generate an interrupt on PCI side by using the FLAGs as outputs and assert one of the Flags **LOW** (deasserted = no interrupt request = **HIGH**). After poweron the flag lines of the SHARC's are defined as inputs, so, before enabling the PLX local interrupt input line it is necessary to define the FLAG0 lines as outputs, and to set them to high values.

This method to generate an interrupt is supported because of compatibility to our VME product line, but needs an additional message-passing to identify the number of the SHARC, that has requested the interrupt.

FLAG1: the lines of all SHARC's are ***tied together*** and used as inputs. With these flags it is possible to read the state of the ***direct master mode write FIFO*** of the PCI-Controller. If the line is asserted (LOW) then the FIFO has been filled up to a (preprogrammable) count, and signals that the PCI bus has ***not*** read out the FIFO data, and so, in other words, the flag line represents the state of the PCI bus, if it is available or not (**PCI-Ready**).

FLAG2: not used, this line is routed to the corresponding **SHARC I/O-Pack Connector**

FLAG3: not used, this line is routed to the corresponding **SHARC I/O-Pack Connector**

6.9 SHARC Links

The link ports of the DSPs support fast communication between the DSPs and with external data sources/sinks up to [1]**40 Mbyte/s** for each link. To keep all options of communication paths, all 6 links of all 6 DSPs are connected to the SHARC I/O-Pack Connectors.

On the board there are two link buses for onboard link interconnections. The **links 1 ... 4** may be jumpered to one of the buses, and **link 5** of the first SHARC may be jumpered to **link 0** of the next SHARC. This jumpering option has to be configured ***before delivering*** the board, so this option is only usable if ordered.

Link interconnections over long distances require differential signal distribution. The maximum line length depends on the type of differential buffers and the maximum frequency which is desired. For long distance purpose there is available a **SHARC I/O-Pack 9001** with high speed differential drivers and receivers.

To boot from link, please refer to the following chapter and the ADSP 2106x manual.

[1] Refer to the latest SHARC anomaly list from Analog Devices

6.10 Board Control Register (EPLD)

The WS 3112 board has an EPLD on board that is controlling the hardware. An 8 bit register is integrated nn this device for defining basic operation modes.

Because of some hardware restrictions there is a special setup procedure. The lower 8 bits of the register ***address*** represent the ***data*** that has to be loaded into the register. Each write operation ***changes all 8 bits*** of the register. The upper addressbits **SA23** ... **SA19** are used to select the register (see address map table below).

Example: if you want to set bit 2 of the register and to reset all other bits of the register, then you have ***first*** to 'OR' a hex value of 0x 04 (register ***data***) and a hex value of 0x 0120 0000 (register ***address***), ***second*** to add this value to the PCI base address of the board, (e.g. 0x F400 0000) and ***finally*** to write this value (e.g. 0x F520 0004) as a physical address to the PCI bus. The ***address*** you write with this operation ***contains all information***, the ***data is don't care***.

Table 4: **Board Control Register, Bit Description**

Bit #	**Control Function**
7	0: DSP Modes (Bits 0..3) disabled, controlled via DIP-SWITCH 1: DSP Modes (Bits 0..3) enabled, controlled via Control Register
6	always 0, for factory testing purpose
5	0: Enable SHARC /SBTS 1: Disable SHARC /SBTS
4	always 0, for factory testing purpose
3	0: DSP Rotating Bus Priority (RPBA) OFF 1: DSP Rotating Bus Priority (RPBA) ON
2	DSP Boot Option EBOOT (see table 5)
1	DSP Boot Option LBOOT (see table 5)
0	DSP Boot Option BMS (see table 5)
	Poweron reset condition: all bits are set to „0“, that means e.g., the poweron boot mode is defined by the DIP-Switch setting.

After poweron the onboard hardware resets all bits of the board control register. The boot condition is then defined by the DIP switch SW 1 (default setting). If bit 7 is set to „0“, all DSPs boot as selected by the ***DIP switch SW1***. To find the correct DIP-Switch-Setting, please translate the DIP-Switch position: the position ON to „0“, and the position OFF to „1“. For information on the ***default DIP-Switch setting*** please refer to **table 1** in chapter 5.3.
The **bits 4 and 6** of the board control register are for testing (factory) purpose only. The default setting of these bits is 0. Please don't change them to another value.
Bit 5 enables/disables the SHARC /SBTS input signal. Normally a timeout mechanism cares for suspending a SHARC DSP from using the SHARC bus after 600ns. If in an application a DSP locks the bus longer than this time interval while a PCI host access is pending, the DSP is forced to release the bus imediately to let the host access pass.
During download of large DSP programs or initialisation of large memory segments by the DSP, it is necessary to temporarely disable this timeout mechanism. If the driver software for WS3112 is used you don´t have to care about this function.
However, if your application needs a DSP bus lock for more than 600ns, you can temporarely disable this timeout by setting bit 5 to "1".
If **bit 7** is set to „1“ all DSPs are booting in the mode selected by bits 0..2 of the ***control register***. The boot modes for the board control register bits 0...2 are:

Table 5**: Boot Operating Modes**

BOOT (Bit 2)	**LBOOT (Bit 1)**	**BMS (Bit 0)**	**Booting Mode**
0	0	0	Host processor booting (Default)
0	0	1	No booting, processor executes from external memory
0	1	0	Link port booting
1	0	Output	not supported by hardware

Table 6**: DIP-Switch setting**

SW4	**SW3**	**SW2**	**SW1**	**Booting Mode, Priority Mode (ON = 0, OFF = 1)**
ON	ON	ON	ON	factory setting
x	OFF	ON	x	not supported by hardware
x	ON	ON	ON	Host processor booting (Default)
x	ON	OFF	ON	Link port booting
x	ON	ON	OFF	No booting, processor executes from external memory
ON	x	x	x	DSP Rotating Bus Priority (RPBA) OFF
OFF	x	x	x	DSP Rotating Bus Priority (RPBA) ON

Host booting is the ***standard boot mode***. It is supported by the WS 3112 Device Driver Kit. In this mode it is possible to load files with the extention „***.LDR**“, that are generated by the AD 2106x Assembler, to the selected SHARC.

6.10.1 Board Reset Condition

After poweron the board is in the reset state. All bits of the board control register have been cleared by hardware. As described above, the DIP-Switch setting defines the boot mode of the SHARC’s. Please check, that the DIP-Switch setting is the default setup (refer to table 1) for downloading files from host to the SHARC’s.

All SHARC’s are in reset state and forced to this state by hardware. To ***recover from this reset state***, it is necessary to write to a PCI-Controller register, that is called „Init Control Register“ (offset 06ch from a base address, that you can read from the „PCI Base Address Register for I/O Access to Runtime Registers“ (Standard PCI Configuration Register with offset 014h of the PCI config space, to this register the host BIOS supplies an address space during poweron)). The bits you have to change in that register are: „0“ to bit 16 (General Purpose Output, connected to the reset line of all SHARC’s), „0“ to bit 30 (PCI Adapter Software Reset), „1“ to bit 31 (Local Init Done).

For detailed information on register setup of the PCI Controller please refer to chapter 12, or use our delivered software driver, that does this job for you.

6.11 Local SRAM

The ***local SRAM*** is 48 bits wide, so every SRAM data line is connected to the corresponding SHARC data line. To access the SRAM directly from host, the SRAM has been logically splitted in ***two pages***, a ***32 bits wide page*** connected to **SD16** ... **SD47**, and a ***16 bits wide page*** connected to **SD0** ... **SD15**. This paging is only visible from the host side, the SHARC's can see every time the full 48 bits wide SRAM. The size of the SRAM may be 64 K DWORDs (384 Kbytes) or 256 K DWORDs (1.5 Mbytes).

6.12 Local EEPROM

The ***local EEPROM*** contains data for initializing the PCI-Board, a reserved area for fabrication information, and a little area for user-supplied data (32 bits). If you want to use this little space, then you must use our software driver, that supports algorithms for writing and reading data. Normally the EEPROM is read-only. To activate a EEPROM write cycle, there has to be set first a jumper on the DSP-Board.

6.13 Status LED's

On the DSP-Board there are 9 status-LED's (if they are mounted on the board). Three are green (LED 1 ... 3), and six are red (LED 4 ... 9). The green one's are signaling the state of the powerlines, and the red one's are signaling logical states during board operation, and are defined in the table below. It is possible, that the red ones glow only a little bit, this is dependent on the duration of the signaled access.

The green LED's have to be ON all the time. After poweron, all the red LED's should be OFF, and this is the normal state of the LED's 9 to 14, only LED 11 should be ON (more or less light) if the host accesses the board (slave mode) or if one of the SHARC accesses the host (master mode).

Table 7: **State of signaling LED's**

LED No.	**Color**	**Definition (State if on)**
LED 1	green	- 12 V power OK
LED 2	green	+ 12 V power OK
LED 3	green	+ 5 V power OK
LED 9	red	host access to an unknown local device
LED 10	red	the DIP-SWICH is not valid, switched to soft boot mode select
LED 11	red	accesses to/from host and/or SHARC
LED 12	red	direct master write FIFO is almost full
LED 13	red	pending interrupt on PCI or SHARC side
LED 14	red	pending DMA request (PCI-Controller or SHARC

6.14 SHARC IO-Packs

The ***modular and open concept*** of add-on modules called „**SHARC IO-Packs**“ offers additional functionality to the DSP-Boards. It is supported by Fa. Wiese with different modules. Please contact the sales office for detailed informations or look at **http://www.wiese.de**

The SHARC I/O-Pack Connectors (Molex) are designed for functional extentions of the DSP-Board. All important signals (from each SHARC) are routed to two corresponding connectors, **XTn** and **XBn** (‘**n**’ is the SHARC processor number).

The user has access to all SHARC IO-ressources: **LINK** ports, to the **SPORT** channels, **IRQ1** and **IRQ2**, **FLAG2**, **FLAG3**, and to most signals of the parallel local SHARC bus. Please refer to the detailed connector-description below (table 8 and 9).

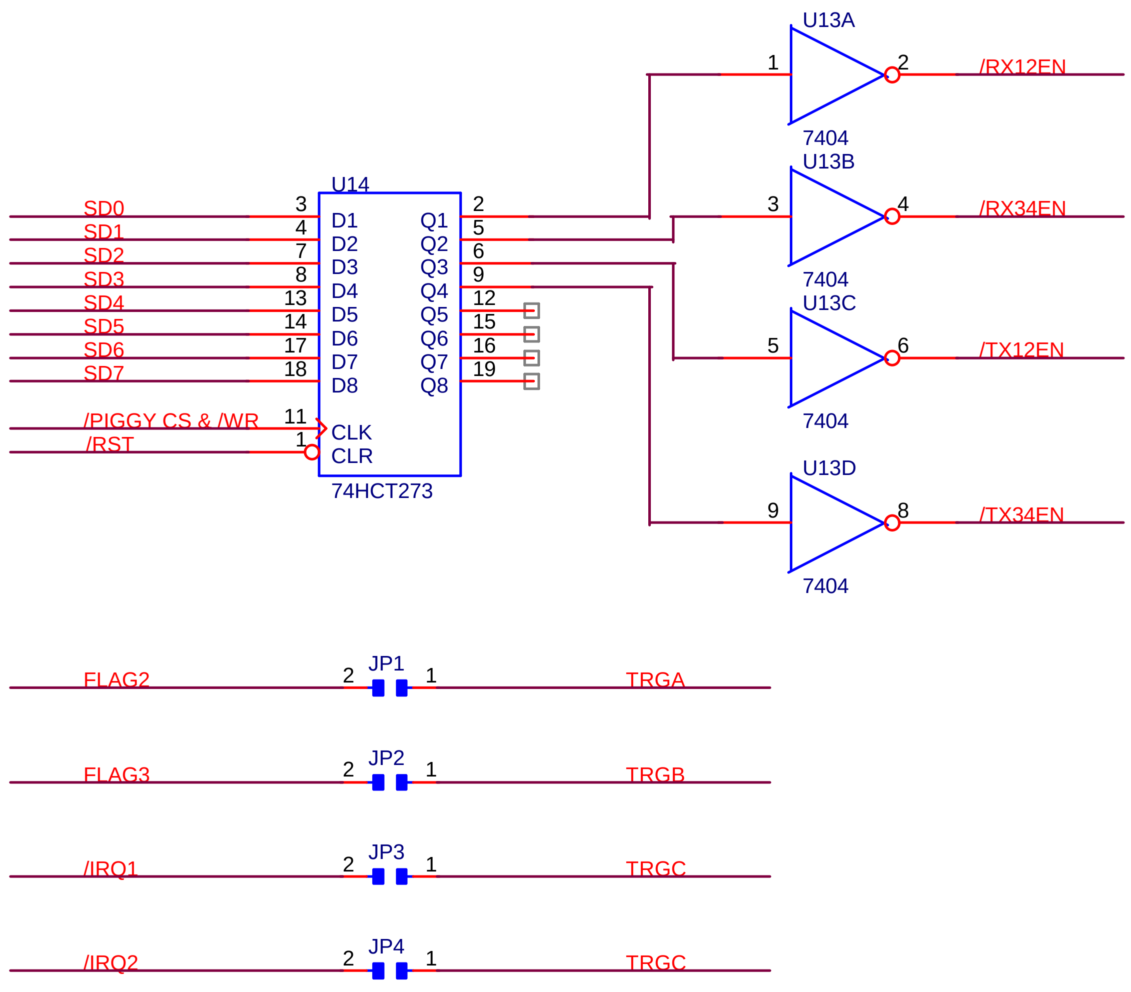
It is possible to access on the piggyback 32 bits of the data bus (**SD16** ... **SD47**), six address lines (**SA0** ... **SA5**) for accessing a set of up to 64 registers (dword) with a pre-select signal (**/PIGGY_CS**).
The signal **/PIGGY_CS** is generated by the glue-logic (EPLD) on board, and follows the address mapping defined in the chapter ‘Host Access’ and ‘SHARC Access’.

With the use of the signals **ADRCLK**, **/RD**, **/WR**, and **/ACK** it is possible to interface synchronous and asynchronous devices to the (synchronous) SHARC bus, and to generate waitstates if necessary (local bus clock is 40 MHz). The lines **/DMARx** and **/DMAGx** are for requesting hardware-controlled DMA channels.

On the board there are some multipurpose lines: three trigger lines **TRGA ... TRGC** and two additional lines called **BUS1** and **BUS2 .** They are all bus lines, seperated from each other, and only connected to all SHARC I/O-Pack Connectors.

In the tables below you can find every signal with a symbloc name and an explanation. The symbolics are the same as used by Analog Devices, so please refer to the SHARC manual, where you also can find the timing specification for the bus signals.

Picture: **Example of interfacing user-defined logic to SHARC IO-Pack connectors**



The drawing above is a little ***example for interfacing*** some user-specific logic to the SHARC multiprocessor bus. One 8-bit-register is used to store some output data on the piggyback module (e.g. for controlling some LED's). Two flag lines and two interrupt lines are routed over jumper to use interconnection lines between different piggy-backs.

In the example there is no need for an address-decoder on the piggyback. In this case decoding it is very simple: you only use the predecoded /PIGGY_CS signal and /WR for accessing the register. The /PIGGY_CS signal has a different offset address on every piggyback, and the usable address space for every piggyback is SA0 ... SA5 (64 DWORDs). Please refer to the address mapping table for information on the address space. The piggybacks are accessible from both sides, the host (PCI) and the SHARC's side. The asserted period of /PIGGY_CS signal is, depending on host or SHARC access, between ca. 50 ns and ca. 100 ns.

Connecting the /RST signal to the clear input of the latches clears the register during poweron. The inverters beyond the register are supporting negative-logic for e.g. chip-selects.

With the example of setting jumpers JP 1 ... JP 4 on the piggyback it is possible to configure different routings, e.g. to use either /IRQ1 or /IRQ2 as a bused signal line between piggybacks.

Table 8: **SHARC I/O-Pack Connector XT 1,2**

Pin-Nr.	Signal	Description XT1, XT2
1	FLAG3	SHARC Flag 3 Pin
2	TRGC	Trigger Bus C for SHARC I/O-Pack
3	FLAG2	SHARC Flag 2 Pin
4	DGND	Reserved for future use
5	TIMEXP	Timer expired
6	/IRQ1	Interrupt Request Line 1
7	/IRQ2	Interrupt Request Line 2
8	DT1	Data Transmit (SPORT 1)
9	/RST	Board Hardware Reset
10	TCLK1	Transmit Clock (SPORT 1)
11	TFS1	Transmit Frame Sync (SPORT 1)
12	DR1	Data Receive (SPORT 1)
13	RCLK1	Receive Clock (SPORT 1)
14	RFS1	Receive Frame Sync (SPORT 1)
15	DGND	
16	L5ACK	Link Port 5 Acknowledge
17	L5CLK	Link Port 5 Clock
18	L5D0	Link Port 5 Data D0
19	L5D1	Link Port 5 Data D1
20	L5D2	Link Port 5 Data D2
21	L5D3	Link Port 5 Data D3
22	+5VD	
23	DGND	
24	+5VD	
25	L4ACK	Link Port 4 Acknowledge
26	L4CLK	Link Port 4 Clock
27	L4D0	Link Port 4 Data D0
28	L4D1	Link Port 4 Data D1
29	L4D2	Link Port 4 Data D2
30	L4D3	Link Port 4 Data D3
31	DGND	
32	L3ACK	Link Port 3 Acknowledge
33	L3CLK	Link Port 3 Clock
34	L3D0	Link Port 3 Data D0
35	L3D1	Link Port 3 Data D1
36	L3D2	Link Port 3 Data D2
37	L3D3	Link Port 3 Data D3
38	+5VD	
39	N.C.	Reserved for future use
40	L2ACK	Link Port 2 Acknowledge
41	L2CLK	Link Port 2 Clock
42	L2D0	Link Port 2 Data D0
43	L2D1	Link Port 2 Data D1
44	L2D2	Link Port 2 Data D2
45	L2D3	Link Port 2 Data D3
46	+5VD	
47	DGND	
48	DGND	
49	L1ACK	Link Port 1 Acknowledge
50	L1CLK	Link Port 1 Clock
51	L1D0	Link Port 1 Data D0
52	L1D1	Link Port 1 Data D1
53	L1D2	Link Port 1 Data D2
54	L1D3	Link Port 1 Data D3
55	+5VD	
56	L0ACK	Link Port 0 Acknowledge
57	L0CLK	Link Port 0 Clock
58	L0D0	Link Port 0 Data D0
59	L0D1	Link Port 0 Data D1
60	L0D2	Link Port 0 Data D2
61	L0D3	Link Port 0 Data D3
62	DGND	
63	TRGB	Trigger Bus B for SHARC I/O-Pack
64	TRGA	Trigger Bus A for SHARC I/O-Pack

Pin-Nr.	Signal	Description XB1, XB2
1	SA0	SHARC address bus (PCI: A2)
2	SA3	SHARC address bus (PCI: A5)
3	SA1	SHARC address bus (PCI: A3)
4	SA4	SHARC address bus (PCI: A6)
5	SA2	SHARC address bus (PCI: A4)
6	SA5	SHARC address bus (PCI: A7)
7	-12V	
8	+5VD	
9	DGND	
10	DGND	
11	/DMAR1	DMA Request 1 (Channel 7)
12	/DMAR2	DMA Request 2 (Channel 8)
13	DGND	
14	ACK	Memory Acknowledge
15	SD17	SHARC data bus (PCI: D1)
16	SD32	SHARC data bus (PCI: D16)
17	SD16	SHARC data bus (PCI: D0)
18	DGND	
19	+12V	
20	BUS1	Bussignal 1
21	DT0	Data Transmit (SPORT 0)
22	/PIGGY_CS	SHARC I/O-Pack Chipselect
23	TFS0	Transmit Frame Sync (SPORT 0)
24	TCLK0	Transmit Clock (SPORT 0)
25	RCLK0	Receive Clock (SPORT 0)
26	DR0	Data Receive (SPORT 0)
27	BUS2	Bussignal 2
28	RFS0	Receive Frame Sync (SPORT 0)
29	+5VD	
30	SD18	SHARC data bus (PCI: D2)
31	/RD	Memory Read Strobe
32	ADRCLK	Synch. Address Clock
33	SD39	SHARC data bus (PCI: D23)
34	/WR	Memory Write Strobe
35	SD37	SHARC data bus (PCI: D21)
36	SD38	SHARC data bus (PCI: D22)
37	SD35	SHARC data bus (PCI: D19)
38	SD36	SHARC data bus (PCI: D20)
39	SD34	SHARC data bus (PCI: D18)
40	SCLK	System Clock (40 MHz)
41	SD33	SHARC data bus (PCI: D17)
42	/DMAG2	DMA Grant 2 (Channel 8)
43	SD20	SHARC data bus (PCI: D4)
44	/DMAG1	DMA Grant 1 (Channel 7)
45	SD30	SHARC data bus (PCI: D14)
46	SD31	SHARC data bus (PCI: D15)
47	SD28	SHARC data bus (PCI: D12)
48	SD29	SHARC data bus (PCI: D13)
49	SD26	SHARC data bus (PCI: D10)
50	SD27	SHARC data bus (PCI: D11)
51	SD24	SHARC data bus (PCI: D8)
52	SD25	SHARC data bus (PCI: D9)
53	SD47	SHARC data bus (PCI: D31)
54	SD23	SHARC data bus (PCI: D7)
55	SD22	SHARC data bus (PCI: D6)
56	SD46	SHARC data bus (PCI: D30)
57	SD21	SHARC data bus (PCI: D5)
58	SD45	SHARC data bus (PCI: D29)
59	SD19	SHARC data bus (PCI: D3)
60	SD44	SHARC data bus (PCI: D28)
61	SD42	SHARC data bus (PCI: D26)
62	SD43	SHARC data bus (PCI: D27)
63	SD41	SHARC data bus (PCI: D25)
64	SD40	SHARC data bus (PCI: D24)

6.15 Special Operating Conditions

In the following section we discuss some special operating conditions, that need additional attention in proper handling.

The WS 3112 board is capable of performing a 'backoff' sequence on it's local side, if a 'deadlock' condition has been detected. Deadlock is a situation, where the local bus of the WS 3112 board is used by e.g. the SHARC's, and a PCI master tries to get the local bus with a 'bus request' signal, but receives no 'bus grant' signal. There are two types of deadlock, a 'partial deadlock' (a PCI master trys to access the DSP board, and at the same time the DSP board trys to access ***another*** PCI device), that could be handled with PCI protocol, and a 'full deadlock' (a PCI master trys to access the DSP board, and at the same time the DSP board trys to access this ***same*** PCI device), that could be handled with 'backoff' (see below).

Under 'normal' conditions this problem is handled by the ***prirorization scheme*** of the local bus. The host has highest priority in requesting the bus, and so the host gets 'always' the local bus, if he wants to have it **-** but there are ***two exceptions***: the local bus has been 'locked' by one of the SHARC's, or a SHARC master access to the PCI has not finished (e.g. is waiting for 'write FIFO not full').

In these two cases a deadlock condition may occur and could be handled with the help of a special SHARC input signal command: 'SBTS' (Suspend Bus Tristate). With 'SBTS' the SHARC's are ***forced to grant*** the bus to the host, so long as the host requires the bus. When the host frees the bus, the signal 'SBTS' is released (deasserted), and the SHARC transfers could be finished. This capability is called ***'backoff'***. The WS 3112 board handles the 'SBTS' signal in this way.

Unfortunately there is again an ***exception***: if one of the SHARC's ***DMA channels*** is used to transfer data in mastermode to e.g. host memory, then the 'SBTS' backoff ***does not work***: a full deadlock may occur (SBTS don't works during DMA transfers, please refer to the AD SHARC manual, chapter 8.8.2.3). The full deadlock arises ***only under the following condition:*** there is ***running a SHARC DMA*** transfers between SHARC memory and host memory, ***and at the same time*** a quick block transfer from host memory to SHARC memory. Since source and destination are the same for both transfers, there is a full deadlock situation, that could not be handled with 'SBTS'. Block ***reads*** at the same time the DMA is running ***are no problem***, and also not ***'slow' writes***, where the PLX FIFO is buffering the incoming data until the actual DMA transfer has been finished.

For this problem there is a ***workaround***: you may use a software solution for avoiding this deadlock. Before the SHARC DMA is started, an information bit in one of the PLX register must be used to signal: „DMA transfer is running, don't write block data at this time to the local side of the DSP board". This bit must be resetted after the end of DMA transfer and polled by host, when he wants to start a block transfer to local resources.

A better way of avoiding the problem is to use one of the ***PLX DMA*** channels, that are not affected by this problem, but have the same functionality. With the PLX DMA's there are ***additional advantages***: DMA transfers from host to local side are more effective than SHARC DMA transfers, because the PLX DMA fetches the data from PCI bus to internal FIFOs ***before*** requesting the local bus. In the other direction there is a similar benefit. The host access has ***higher priority*** than a running DMA, so the host gets control of the local bus ***at all time*** he wants. In mastermode (PLX DMA reads or writes) no mapping scheme restriction exists when transferring data to or from host, the full 4 GByte address space is available the whole time.

The ***design structure of the host buses*** (on your PC motherboard) is another cause of perhaps unexpected behaviour. The PCI bus is capable of performing a 'backoff' when accessing a device, that means release the bus, and repeat a retry at another time. If 'behind' the PCI bus there is another system bus, e.g. 'ISA' bus (as in most modern PC motherboard designs), then the behaviour of handling an access is depending on the so called ‘legacy’ bus: on ***ISA buses*** e.g. an access must be completed within a given time, there is no ‘backoff’ protocol defined. So, if you transfer data between e.g. the WS 3112 board and a device located at the ISA bus, then a backoff of the PCI bus does not work in all cases and the backoff mechanism of the WS 3112 board is needed.

The next point is to discuss the ***RESET features*** of the board. There are different ways to perform a reset. First, after poweron, the local side of the WS 3112 board (PLX buslogic, EPLD logic and SHARC’s reset pins) is held in reset state, the so called ***board reset***, until the DDL driver starts (if you don’t use the DDL driver: reset release is performed by writing a ‘0’ to the general purpose output bit of the PLX user I/O control register). When the DDL driver finishes, there is also a reset performed to ensure, that none of the SHARC’s could perform e.g. some write operations in mastermode to host memory any longer.

If you push the host reset button, the same condition arises: a ***board reset*** is performed (this board reset ***does not*** include the setup of the PCI register on the board, so if you want to be sure, that they are set correctly by the system BIOS, you must ***switch off and on*** the computer!).

In earlier hardware revisions of the ADSP 2106x it was allowed to reset a DSP by writing to a bit in the SYSCON register **- This reset method is no longer supported by Analog Devices !** In case you have questions to this point, please look at the anomaly list of the SHARC DSP in use or contact us.

7. JTAG Interface

A JTAG interface for debugging the DSP software and status analizing is implemented. The EZ-ICE from Analog Devices is supported.
When you are unpacking the board, then you can find on connector X6 two jumpers (in position X6 - 7/8 and X6 - 9/10). They are ***necessary*** if you use the board ***without*** the EZ-ICE, otherwise you must remove the jumpers. Please refer to chapter 5.3 for the default setting.

8. Connectors

On the board you can find different connectors. For identifying the name of the connectors and their position please refer to picture 1 (top view of the board) and to chapter 5.3.

The connectors **XT1,2** and **XB 1,2** are for use in conjunction with the SHARC IO-Packs, please refer to chapter 6.14.

The connector **X6** is for use with the EZ-ICE from Analog Devices, please refer to chapter 7.

The other connectors, named **X2,** and **X7 to X9,** are for future use or for factory testing and setup procedures. **Don’t use them** or connect any jumpers to one of these connectors!

9. Software Driver Package „WS 3112 Device Driver Library (DDL)“

With your DSP board you have got a software package, called **D**evice **D**river **L**ibrary (DDL, „Slave“ version) with a standard set of function calls. This standard set includes function calls for using all board devices in ***slave mode***, which means, that the host controls all transfers.

The enhanced („Standard“) version of the device driver library, that is available for the WS 3112 board from Fa. Wiese is downward compatible to the slave version, includes the standard set of slave functions, and ***additional functionality*** for controlling master mode, interrupts, dma transfers and multiple boards.

The use of the driver package is the best way to have a quick and troublefree start with the DSP board. The user is free of understanding the PCI mapping and operating system dependent address translation procedures and alignments, because the driver automatically uses all mapping infor-mations, address-translations, alinments, register setups, and other system requirements. Instead of low-level programming you can use high-level calls to all important features of the WS 3112 DSP board. Both driver packages are using modern, object-oriented methods for accessing all hardware components on board.

Dependent on your problem, on your operating system and your prefered language, you can get different versions of the driver package from Fa. Wiese Signalverarbeitung GmbH (the drivers are only usable on hosts with INTEL 386 or higher CPU’s):

- „Slave“ and „Standard“ DLL, with ANSI-C interface, running under Windows 3.1x or Windows 95
- „Slave“ and „Standard“ DLL, with PASCAL interface, running under Windows 3.1x or Windows 95
- „Slave“ and „Standard“ LIB, with ANSI-C interface, running under MSDOS 5.0 or higher (not compatible to the EMM386.EXE driver!)

Under Development: Standard DLL for Windows NT, with NT Kernel Mode Hardware Driver. For availability of source-code of the driver library or for the need of special designed drivers and application-specific support please contact the Wiese Signalverarbeitung sales office.

9.1 Host Software Development Procedure

The development of software for the SHARC board is typically devided in two different parts, the host part and the SHARC part, and normally the two software-parts have to communicate with each other.

The host is capable of producing code for both parts, with native compiler and assembler for the host CPU, and with cross compiler and assembler for the embedded SHARC system. The driver of our DDL package represents a software interface for the host system. Dependent on the operating system there are different appearances of the interface.

The host development software package is also dependent on the operating system you use. By using the matching tools you can generate executable code. To accessi all resources on the WS 3112 board, it is necessary to link our software driver package with this code, or to use a dynamic link library (DLL) version of our DDL package.

9.1.1 Description of the Windows DLL Interface

The windows DLL interface represents a flat model or representation of the hardware. For the generation of the DLL we translated the internal C++ calls to a standard C DLL interface. The functionality of the class library and the DLL library is equivalent.

The details of the description please take from the readme and header files of the delivered software. There you can find a short description of every public method.

9.1.2 Description of the DOS Interface

The DOS interface represents a flat model or representation of the hardware, and is the same as for Windows. The difference to the Windows DLL is, that the DOS version is a library, that is linkable to your application program during link time, not run time.

The details of the description please take from the header files of the delivered software. There you can find a short description of every public method.

9.1.3 Description of the Class Library Interface

The class library interface consists of a set of classes, that is used to build the driver. The classes follow a hierarchical structure, and are representing in software a picture of the real hardware world.

In the library you can find for every hardware device on the board a corresponding class representation. The board class is the ‘main’ class and contains objects (instances of a set of derived classes). To each class or object there are belonging ‘methods’, that are describing the behaviour of the object.

The objectoriented design is (by definition) well structured, and so perhaps easier to understand than standard software procedures, but, of course, you have to be familiar with the language C++.

The details of the description please take from the header files of the delivered software. There you can find a short description of every method. The class library interface is ***not*** part of the „slave“ or „standard“ DDL versions, it must be ordered seperately!

9.2 SHARC Software Development Procedure

The normal way to develop software for the SHARC DSP’s is to use the host PC resources. With an standard editor and a cross-compiler and -linker package from Analog Devices it is possible to generate SHARC loadable files on a PC. These files can be downloaded to the SHARC’s by using the WS 3112 DDL.

After starting on the SHARC’s the downloaded DSP-code it is possible to use the host slave communication capability to monitor and debug the running SHARC by e.g. inspecting it’s registers and internal memory.

9.2.1 Language Interface

Analog Devices delivers an Assembler and an ANSI C Compiler for use with the 21xxx DSP family. The assembler package is necessary to produce efficient and quick code, and the compiler package you can use to encapsulate your assembler code, or to produce a result in a few minutes. This compiler package has a library extension to the ANSI standard, and includes quick, optimized FFT's, filters etc. The compiler can link high level modules and assembler modules to an loadable file for downloading over the host interface or over a link port. On our DDL floppy you can find an example how to handle the result of a compilation.

9.2.2 Testing your Software

You can test your software with the Analog Devices simulator, that is running on the host, and simulates the SHARC's behaviour like an interpreter. Another way of debugging is to use the emulator of Analog Devices (ICE), that enables testing during runtime via the JTAG port of the board.

The runtime inter-communication or the interfacing between the host system and the independent SHARC software must be done by using a handshake mechanism, for instance message-passing through mailbox register (by polling), doorbell register (by generating interrupts) or DMA channels (by exchanging data structures).

10. Trouble - Shooting

If you have any problems with the WS 3112 board, please follow the steps below ***before*** calling our hotline service:

Please use our delivered testing program for identifying the problem. If you use this file, then there is a log file, that may contain errors and messages. The file 'XXXX.log' appends all errors, so please look to the bottom of that file, or check the date and time to get the latest (newest) error information, or delete this file before starting the driver. The file also contains messages, for instance the revision code of the PCI BIOS you use, and PCI mapping informations. ***Both file informations are necessary to identify the problem you have***. This log file is always generated, even if you use our delivered driver software in your own environment.

Switch off your computer, wait some seconds, and switch it on again (in a normal situation you may use the reset button, because the DSP board uses the reset line to resets all onboard devices). ***Don't use the combination <CTRL><ALT><DELETE>,*** because this software reset generates no ***hardware*** reset, and so no board reset! If the error disappears, then probably you have written some software that accesses registers on the board in a wrong way, or destroys some necessary register contents, e.g. PCI-register.

The next step is to ***reduce your system to a minimal configuration***. Try to boot your system from a diskette that contains no 'autoexec.bat' and no 'config.sys' files, and start our testing software. If there are again errors reported, remove other hardware components, and retry starting the driver. If all this has no success, then please fill out the following sheet before contacting our hotline service!

11. Hotline & Service Information

Please fill out the following table before contacting our Hotline:

Description	Example	Your Configuration
Computer Information (if known)		
Vendor	Compac	
Purchase Date	1995	
Operating System	DOS, Win 95, Win NT	
CPU and Speed	Pentium 133 MHz	
Extern Clock (PCI Bus Clock)	66 MHz	
DRAM Size	32 MByte	
Cache	Piplined Burst 256 K	
Other Hardware Components	Adaptec SCSI xxxx Adapter	
Other Software Drivers		
WS 3112 Driver Information (Contents of File 'XXXX.LOG')		
Version	0.9	
Date of Generation	Sep 05 1996	
No of Boards supported	3	
PCI BIOS Information		
BIOS Version	2.16	
No of PCI Buses	1	
PCI Bus No	1	
PCI Slot No	9	
Vendor ID	10b5	
Device ID	f060	
Revision	03	
Class	06	
Subclass	80	
Mapping Information		
RTR Base Address	fbfe0000	
Local Base Address	f4000000	
Hardware IRQ	12	
PCI INT	A	
Board Version	1.1	
Serial No.	00001	
KJ / KW	9626	
EEProm Setup Date	Jun 25 1996	
Device Test Information		
PCI Bridge	OK	
EEPROM	OK	
EPLD	OK	
SRAM	64K OK	
SHARC 1	OK	
SHARC 2	OK	

Contact:

Post:	Fa. Wiese Signalverarbeitung GmbH Seelandstr. 3 23569 Lübeck - Germany -	
Tel.:	**+49 (0)451 3909 454**	
Fax.:	**+49 (0)451 3909 499**	
E-mail:	**support@wiese.de**	Technical questions
	info@wiese.de	Overview of Wiese products and pointer to other specific product information (all information requested from this site are automatic e-mail responses)
	sales@wiese.de	Pricing information, delivery times
Web:	http://**www.wiese.de**	Product overview, technical specifications, anouncements, news

12. Literature List

For understanding the software handling of the WS 3112 DSP board it is important to own the user manuals of the two complex onboard devices, the PCI-Controller PCI 9060 and the SHARC DSP 2106x. In the manuals you can find a detailed description of the functionality of these devices:

ADSP - 2106x SHARC User's Manual
First Edition (3/95)
Analog Devices, Inc.
Computer Products Division
For newest information and SHARC-FAQs look at: http://www.analog.com

PCI Bus Interface and Clock Distribution Chips / Product Catalog / May 1996
PCI 9060 Bus Master Interface Chip for Adapters and Embedded Systems
PLX Technology, Inc.
Mountain View, CA 94043
For newest information look at: http://www.plx.com

13. Technical Data

Specification	Data
Mechanics	
Length	194 mm (Middle Length)
Height	107 mm (Standard)
Weight	approx. 600 g
Operating Temperature, ambient	0° C ... + 50° C
Storing Temperature	-20° C ... + 70° C
rel. Feuchte	20 % ... 85 %
PCI Bus	
Supply voltage	5 V
Supply current	approx. 2 A
Bus width	32 bits
Clock	33 MHz
Compliance	2.0
PCI Controller	
Controller	PCI 9060 (PLX Technology)
Message Register	8, each 32 bits
Doorbell Register (for interrupt handling)	2, each 32 bits
FIFO	4, each 8 DWORD Depth
DMA Channels	2
DMA Transfer Modes	normal, on demand, chaining mode
Data Transfer Modes	normal, bursting
Controller Operation Modes	Slave Mode, Master Mode
Local Bus Devices	
Clock	40 MHz
DSP	ADSP 21062 (SHARC, Analog Devices)
Address Bus Width	A0 ... A29
Data Bus Width	D0 ... D47
Normal (32 Bit Path)	D16 ... D47
Extended Data Path	D0 ... D15
Local (external SRAM)	64 K or 256 K DWORD (48 Bits)
Local SRAM Waitstates	1

14. Appendix

There is no appendix up to now...