WS 2126
VME SHARC Cluster

WIESE signal verarbeitung

1. Manual Revision History

Version	Date	Name	Comments
1.0	24.09.96	te	First edition
1.1	09.01.97	te	Description of SHARC IO-Pack interface added
1.2	24.02.98	te	Description of SBTS control added
1.3	27.04.98	te	Description of VME Blocktransfer BLT32 added
1.4	30.06.00	te	Description of VME Master Mode corrected
1.5	20.07.00	te	Revised, SBTS - and Master Mode description extended
1.6	10.10.01	te	DSP Timeout acknowledge for Master Mode added FIFO status flags corrected
1.7	09.01.02	te	Generally revised Procedure to write to the Master Control Register has changed

2. Contents

3. Glossary

- SHARC - The ***S***uper ***H***arvard ***AR***chitecture ***C***omputer is a high performance 32 bit IEEE floating point digital signal processor. AD21062 and AD21060 are two members of the SHARC family.
The DSPs work at maximum frequency of 40 MHz. All instructions are executet within one cycle.
The AD21062 contains 2 Mbits SRAM of internal dual ported memory whereas the memory size is 4 Mbit in case of the AD21060.
Six link ports for I/O - or interprocessor communication at a data rate of 40 MByte each are provided.
Refer to the ADSP-2106x SHARC User´s Manual📖1 for more details.

- SHARC IO-Pack - A piggy back concept which realizes a very flexible I/O configuration to the SHARC DSPs.
Contact us for more information about SHARC IO-Packs and their availability.

- A24, A32, D16, D32 - Terms which are used when talking about address areas and data size during VME transfers. A24 means that only A1 .. A23 are used for decoding the slave address. The address bits A24 .. A31 are don`t care. In A32 all address lines are used for decoding. To indicate which address mode is used, a set of six address modifier lines (AM5..AM0) present a code during the address phase of a VMEbus cycle.
On WS2126 only aligned transfers are allowed. i.e. a D32 transfer must specify address 0, 4, 8, $C, ... In D16 mode only even adresses are allowed.
Please refer to the latest 📖2 VME specification for more details.

- VME master - In most cases a board which initiates transfer cycles over the VME bus between itself and a slave board. Please refer to the latest VME specification for more details.

- VME slave - In most cases a board which detects transfer cycles and responds to them if selected. Please refer to the latest VME specification for more details.

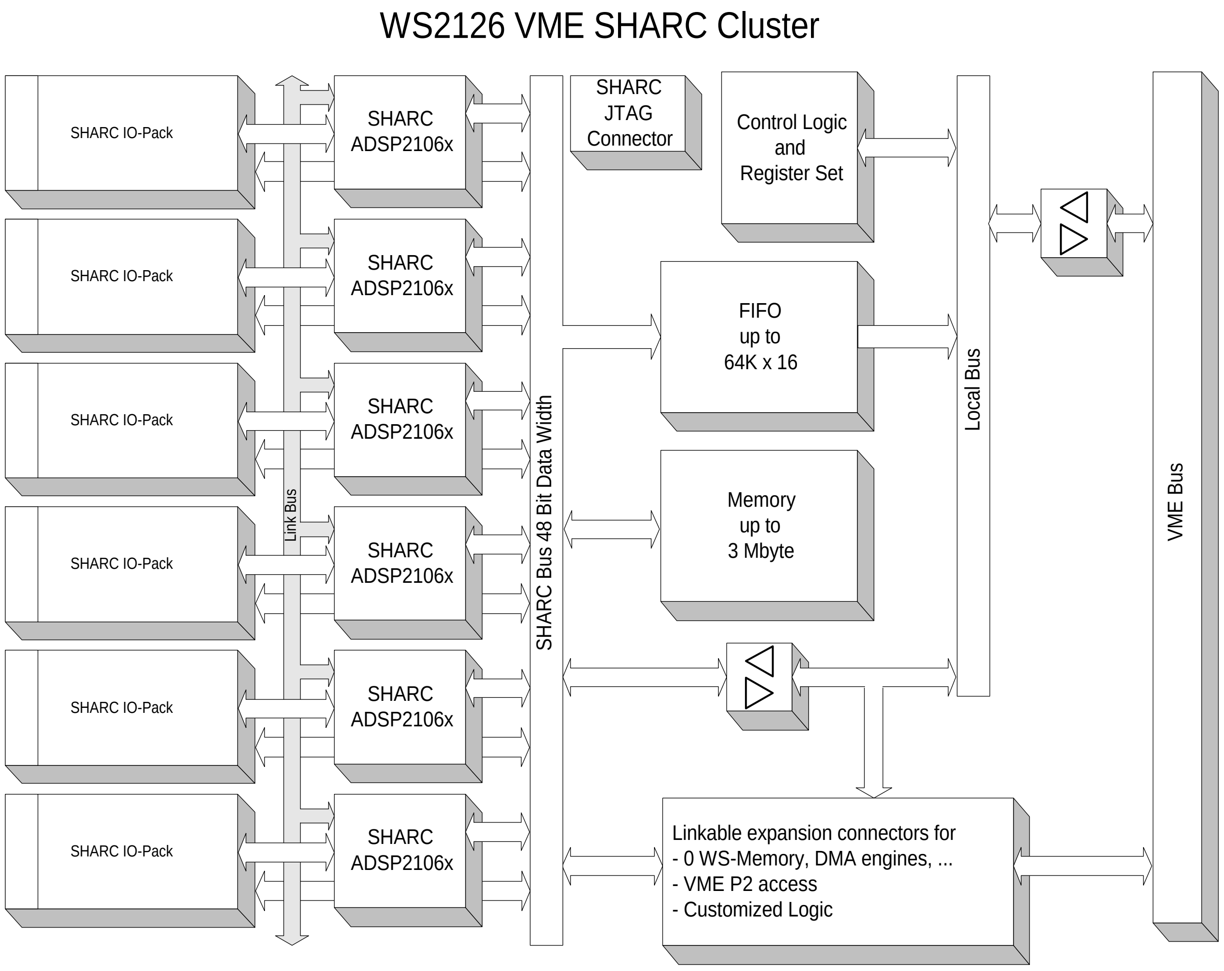
- Host access - The access with the highest priority to the SHARC DSPs is the host access. All interprocessor communication on the DSP bus stops if a host requests control of it. The VME master access to the SHARC DSPs acts as host access. Refer to the ADSP-2106x SHARC User´s Manual for more details.

4. General Description

The WS2126 VME SHARC cluster is a double height VME master/slave board with 6 ADSP 2106x. The features are:

- Six SHARC ADSP 21060 or ADSP 21062 share one local multiprocessor bus.
- Up to 512K * 48 bit static memory on board. The memory is connected to the multiprocessor bus and can be accessed by every DSP and the VME bus.
- VME A32/D16 and D32 host interface to the multiprocessor DSP bus.
- Each DSP can perform master cycles on the VME bus.
- A local D16 bus leads to a configuration-/ FIFO window without touching the DSP multiprocessor bus. The 16 bit wide FIFO is connected between the DSP bus and the local bus to support DSP data to the VME bus without intervention of the DSP bus. FIFO size can be choosen from 2K to 32K * 16 Bit.
- Programable VME IRQ level. Each DSP can request an interrupt on the VME bus.
- All links of all DSPs are routable to the front end connectors (6 links for every DSP). To increase the interprocessor communication paths a link bus runs the full length of the VME card. Each of its three links can be jumpered to any DSP. This allows link interconnections between DSPs on the same board without external wireing.
- Each DSP has a dedicated slot for one SHARC IO-Pack submodule. Customized input logic such as ADCs, DACs and buffer functions realize a very flexible interface concept to the SHARC DSPs.
- DSP bus expansion connectors and a local connector to VME P2 rows a + c offer memory expansion with 0 waitstates, customized piggy backs with specialized functions such as DMA devices, signal distribution to other VME boards ...
- For software debugging a JTAG connector is implemented. The EZ-ICE debugger from ADI can be connected.

5. Blockdiagram



6. Getting Started

6.1 Hardware Configuration

The first thing to do before inserting the module into a VME slot is to adjust the A24 address in the VME space with the rotary hex switches and to verify whether the board pins in the VME P1 and P2 connectors are intact.
If SHARC IO-Packs are installed on the WS2126, please verify that all piggy backs are well situated and fixed in their sockets.

An important thing to check is whether the VME backplane in the actual slot has any kind of signals on P2 rows a and c which may collide with user defined signals on WS2126 which use those connector too.
Ensure a proper jumper configuration of your backplane: Some VME backplanes have jumpers to close the daisy chains for /IACKIN-/IACKOUT and /BGxIN-/BGxOUT if the slot in question is not used.

6.2 Software Configuration

The configuration of WS2126 is done in VME A24 area which is specified by the rotary hex switches. If i.e. an address $64 is adjusted with the rotary switches the value $64 0000 is the board base address in A24 mode.
Since the DSPs and the global memory are only accessable in A32 mode, the base address for this mode must be defined.
To define the A32 base address, configure the address offset register ($64 0008). The upper 7 bits of the 16 bit data word which is written to this register are interpreted as the module A32 base address.

<u>Sequence:</u> A32 Slave Setup

1. The 16 bit value which was written to the address offset register is $3800. The resulting A32 base address in this case is $3800 0000.
   Note that only the upper <u>7 bits</u> are relevant for the address setting. Writing $3900 to the offset register will also result in a base address of $3**<u>8</u>**00 0000 !
2. The next step is to allow VME masters to access WS2126. After the A32 base address is defined, bit 15 in the control register ($64 0004) must be set to enable A32 access to the DSPs and the memory.
3. One of the six DSPs acts as DSP bus master and submits a host bus grant if the host requests for it. The DSP bus master is only active, if all DSPs are not in the reset state. Therfore the last thing to do is push all DSPs into the run state. This is done by setting bit 3 in the reset register ($64 0000).
4. Now a VME master has full access to the WS2126.

7. VME Interface

7.1 VME Slave Interface

The VME slave interface responds to A24/D16 (AM = $39, $3D) and A32 / D16 / D32 (AM = $09, $0D) single cycles. The D32 access is long word aligned only. Read Modify Write cycles (RMW) are not allowed.
The A24 address area is used for board setup and VME access to the FIFOs. All access to the DSPs and memory is done in A32 mode.

The base address in A24 mode is adjustable by setting the two hex switches on the board. Both hex switches define the 8 upper most address bits of the board VME address:

```
                    23        Address bits          0
VME address A24:     SSSS SSSS xxxx xxxx xxxx xxx-
```

Example: A desired board address of "$70 xxxx" can be set by switching the upper hex switch to "7" and the lower hex switch to "0".

The base address in A32 mode is determined by the value in the register "address offset". This 16 bit wide register contains the upper 7 bits of the VME board address.

```
                    31                Address bits              0
VME address A32:     AAAA AAAx xxxx xxxx xxxx xxxx xxxx xxx-
```

7.1.1 VME Address Map

The final address is calculated by adding the board address to the address named in the table.

Address (Hexadecimal)	Mode	Read/Write	Description
400.. 7fc	A32	R/W	SHARC IO-Pack 1
800.. bfc	A32	R/W	SHARC IO-Pack 2
c00.. ffc	A32	R/W	SHARC IO-Pack 3
1000.. 13fc	A32	R/W	SHARC IO-Pack 4
1400.. 17fc	A32	R/W	SHARC IO-Pack 5
1800.. 1bfc	A32	R/W	SHARC IO-Pack 6
20 0000.. 3f fffc	A32	R/W	SHARC1
40 0000.. 5f fffc	A32	R/W	SHARC2
60 0000.. 7f fffc	A32	R/W	SHARC3
80 0000.. 9f fffc	A32	R/W	SHARC4
a0 0000.. bf fffc	A32	R/W	SHARC5
c0 0000.. df fffc	A32	R/W	SHARC6
e0 0000.. ff fffc	A32	W	Broadcast Area SHARC 1..6
100 0000.. 17f fffc	A32	R/W	Memory: VME D[31..0] -> Mem. D[47..16]
180 0000.. 1ff fffc	A32	R/W	Memory: VME D[15..0] -> Mem. D[15..0]
0	A24	W	Reset Register
4	A24	R/W	Control Register
8	A24	R/W	Address offset register for A32 access
c	A24	R	Fifo Data
10	A24	R	Fifo Status
14	A24	w	Extended Control Register

Attention: ***After power up all DSPs are in the reset state ! The first action to do is to write to the DSP Reset Register in order to start all DSPs to be rady for booting. Memory access is possible only after starting the DSPs.***

STOP

7.1.2 VME Slave Registers

7.1.2.1 Reset Register

Offset address A24: **$ 0 write only**. (In case of A24 base is $64 0000: Register address is $64 0000)

Bit #	Write Function
7	"1": Reset all SHARC DSPs
6	"1": Reset logic on DSP bus expansion connectors
5	"1": Reset onboard FIFOs
4	"1": Board reset for all devices incl. VME offset register and control register
3	"1": Run all SHARC DSPs so that they are ready for booting
2	"1": Release reset for logic on DSP bus expansion connectors
1	"1": Release reset for onboard FIFOs
0	"X": don`t care

After power on all devices are in the reset state (Default setting).
The DSPs are set into "Run" modus by writing a "1" to bit 3 of the Reset Register. Other devices are not affected.
If a board reset (bit 4) is programmed, the reset action is done immediately and it is not necessary to reset bit 4 again.
The Reset Register is write only.

<u>Example:</u> FIFO Reset

Writing a $ 20 into the Reset Register causes the FIFOs step into the reset state. Other devices are not affected. To recover from the reset state, write a $ 2 into the reset register.

7.1.2.2 Control Register

Offset address A24: **$ 4**. (In case of A24 base is $64 0000: Register address is $64 0004)

Bit #	Read / Write Function	
15	"0": Disable slave A32 access	"1": Enable A32 slave access
14	"0": Disable VME Bus Lock via SHARC	"1": Enable VME Bus Lock via SHARC
13	"0": DSP boot condition selected by DIP-switch	"1": DSP boot condition selected by register bits
12	"0": VME IRQs disabled	"1": VME IRQs enabled
11	"0": Fixed DSP Bus Priority	"1": Rotating DSP Bus Priority RPBA
10	DSP Boot Option EBOOT	
9	DSP Boot Option LBOOT	
8	DSP Boot Option BMS	
7 .. 3	Board ID to be set by user	
2 .. 0	VME IRQ Level	

After power on the boot condition is defined by the DIP switches (Default setting). Please refer to chapter **13.1** for the switch locations.
If bit 13 is set to "0" all DSPs boot as selected by the DIP switches. If bit 13 is set to "1" all DSPs boot as selected by bits 8..10 in the control register. The boot modes are:

EBOOT	LBOOT	BMS	Booting Mode
1	0	Output	not supported by hardware
0	0	1	Host processor
0	1	1	Link port
0	0	0	No booting, processor executes from external memory

7.1.2.3 VME Offset Register

Offset address A24: **$ 8.** (In case of A24 base is $64 0000: Register address is $64 0008)

Bit #	**Write Function**	**Read Function**	**Read Function After Reset**
15	no	Status: "1": DSP VME master Buserror occured "0": No VME Buserror during DSP master mode	"x"
14	no	"n2"	"x"
13	no	"n1"	"x"
12	no	Status: "1": DSP VME master Buserror flag enabled "0": DSP VME master Buserror flag disabled	"x"
11	no	"r3"	"x"
10	no	"r2"	"x"
9	no	"r1"	"x"
8	no	"r0"	"x"
7	A31 VME Offset	A31 VME Offset	"0"
6	A30 VME Offset	A30 VME Offset	"0"
5	A29 VME Offset	A29 VME Offset	"0"
4	A28 VME Offset	A28 VME Offset	"1"
3	A27 VME Offset	A27 VME Offset	"0"
2	A26 VME Offset	A26 VME Offset	"0"
1	A25 VME Offset	A25 VME Offset	"1"
0	"x"	"x"	"0"

After power on the board is not accessable in A32 mode. Only after a valid offset address is written into the offset address register and the "A32 Enable"-bit is set, accesses in A32 mode are allowed.
The value in this register plus the specific address in the VME address map is the final VME address for onboard locations.
Only after reset / power on the register value represents a board ID which is $ xx12.
If an address is written to the offset register the lower seven bit of the board ID are overwritten (of course ☺).
In case of DSP master cycles on the VME bus, it may happen that the DSP tries to access locations on the VME bus which are not available. In this case the bus timer of the system (slot 0 -) controller terminates the cycle by pulling /BERR low (Buserror). The DSP then terminates the cycle with unrelieable data.
To indicate whether such a buserror occured, **bit 15** is set to "1". This status is latched so any buserror will set bit 15. Under normal conditions bit 15 is cleared (no buserrors occured).
To control the DSP master VME busserror flag, bit 12 is used: A "1" enables the VME master buserror latch, a "0" disables this feature. Please see description in 7.1.2.4.

7.1.2.4 FIFO Status

Offset address A24: **$ 10.** (In case of A24 base is $64 0000: Register address is $64 0010)

Bit #	**Read Function**
2	"0": FIFO full
1	"0": FIFO half full
0	"0": FIFO empty

After power on the FIFOs are held in reset state: Bit 2 and bit 1 are set to "1" and bit 0 is cleared.

7.1.2.5 Extended Control Register

Offset address A24: **$ 14 write only.** (In case of A24 base is $64 0000: Register address is $64 0014)

Bit #	Write Function	
0	"0": SBTS signal enabled	"1": SBTS signal disabled
1	"0": VME master buserror flag disabled	"1": VME master buserror flag disabled

Bit 0: During downloading very large DSP programs, it may be necessary to switch off the SBTS signal. Please refer to the SHARC Users manual regarding the SBTS function. After the program download SBTS has to be enabled again !
Bit 1: In some cases of soft- and/or hardware debugging, a DSP which is working as a VME master gets no valid /DTACK response from a VME slave. In order to prevent the whole system from a 'hang up', the bus timer in the system controller submitts a buserror signal (/BERR). The VME cycle in question will be terminated and a status bit " DSP VME master timeout" (bit 15 in the VME Offset Register) will be set.
The error status is latched and can only be cleared by resetting bit 1 in the Extended Control Register.
This feature should only be used in critical applications, because if the VME master timeout is enabled, you normally have to monitor the error status after every VME master cycle.

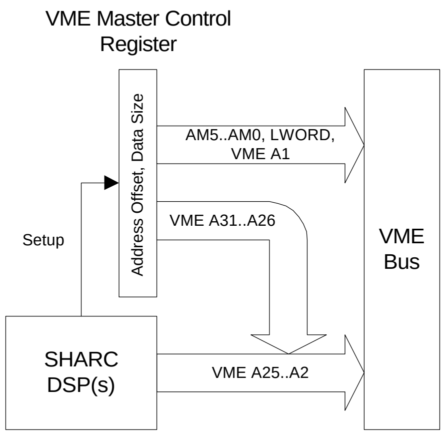
Procedure to enable the VME master buserror flag:

1) Set bit 1 to "1" in the Extended Control Register.
   The hardware is now able to detect and flag buserrors during DSP-VME master access.
2) Verify, that bit 12 in the VME Offset Register is set to "1". Because you can't read back the contents of the Extended Control Register, bit 12 in the VME Offset Register displays the status: VME master buserror detection is enabled.
3) Verify, that bit 15 in the VME Offset Register is cleared to "0": No buserror has occured up to now.
4) Perform a VME bus access by a DSP.
5) Check bit 15 in the VME Offset Register. If it's status is "0" then no VME buserror has occurred up to now and you can continue to execute VME cycles. If a "1" in bit 15 indicates an error, (one of) the last VME cycle(s) caused a VME timeout. Checking the status of the VME Offset Register automatically clears bit 15 to "0" afterwards.

7.2 VME Master Interface

The VME master interface is able to generate A16/A24/A32 cycles with D16 and D32 data width. The address modifier code is free programmable. The master interface does not include system controller functions such as bus arbiters and VME interrupt handler functions.
Each DSP on WS2126 can perform VME cycles by accessing its MS2 predecoded external memory area. To be able to generate addresses for the whole VME address area (A31..A1) an offset register is implemented which holds the upper 6 VME address lines (A31..A26) and control signals for data size on the VME bus. The lower VME address lines are controlled directly by the SHARC DSP.

A mapping scheme shows the address paths:



Sequence: VME master access from a DSP

1. Set ***MemSize*** in the DSP ***SYSCON*** register to a value that ensures an address window for ***MS2*** which is large enough to address the desired VME bus address area.
2. Write a data word to the master control register. This data word (16 bit wide) includes the upper 6 VME address bits (offset window), the AM code, LWORD and VME A1.
3. The DSP address for the master access in this offset window is calculated by shifting right twice the desired VME address (divide by 4). With this address value the DSP can perform VME master cycles.

7.2.1 Master Control Register

DSP address: $ 40 1c00 + ***MemSize*** (write only)

Bit #	Write Function	Read Function
15	VME A31	X
14	VME A30	X
13	VME A29	X
12	VME A28	X
11	VME A27	X
10	VME A26	X
9	BLT32 see **7.2.2.3**	X
8	VME Bus Lock see **7.2.2.2**	X
7	/LWORD (Active low ! To assign /LWORD, set this bit to "0")	X
6	VME A1	X
5	Address Modifier AM5	X
4	Address Modifier AM4	X
3	Address Modifier AM3	X
2	Address Modifier AM2	X
1	Address Modifier AM1	X
0	Address Modifier AM0	"1": Buserror during master cycle(s)

7.2.2 DSP Master Cycles

On the VME bus all WS2126 master cycles are single cycles but in fact we have to distinguish between core driven DSP cycles and DMA cycles from the DSP´s point of view.
Core driven cycles are directly controlled by the DSP core´s program sequence. DMA cycles are controlled by on chip machines which can perform internal and external DSP cycles without disturbing the core´s program sequenceing.
The hardware interface handles DMA master cycles in a different way from core master cycles and the following two sections describe how to handle both master modes.
Please ensure, that the SBTS Control Register is set to "Enable" while doing any master cycles.

7.2.2.1 Core Driven Master Cycles

While executing core driven VME master cycles there are no programming restrictions. Every time the DSP does access to the master window, the VME bus is requested for one cycle and released afterwards.

<u>**Example Master Setup (core driven)**</u>

In this AD-Assembler example, a VME master setup is programmed. This is not a complete source file but will give you an idea how to prepare VME master mode for core driven cycles.

Contents of the H - file:

```
/* Sharc Extern Banksize Exponent: 2^28 = 256 MB */
#define SHC_EXT_BANKSIZE_EXP    28

/* Sharc Extern Banksize in words */
#define SHC_EXT_BANKSIZE        (0x01 << SHC_EXT_BANKSIZE_EXP)

/* MSIZE in SYSCON-Reg: SHC_EXT_BANKSIZE_EXP-13 ShiftLeft 12 */
#define SHC_MSIZE               (SHC_EXT_BANKSIZE_EXP-13)<<12

/* Addresses of external Banks */
#define SHC_EXT_BANK0   0x00400000
#define SHC_EXT_BANK1   SHC_EXT_BANK0+SHC_EXT_BANKSIZE
#define SHC_EXT_BANK2   SHC_EXT_BANK1+SHC_EXT_BANKSIZE
#define SHC_EXT_BANK3   SHC_EXT_BANK2+SHC_EXT_BANKSIZE

#define MASTERRAM_BASE  SHC_EXT_BANK2
#define VME_MASTER_BASE  SHC_EXT_BANK2 + 0xc00000   /* IMPORTANT ! See remark below */
```

The DSP`s master window is bank 2 which starts at an address defined by (2 x MSIZE) + 0x40 0000. In the example below the window starts at 0x2040 0000. Due to the fact that the lower 26 DSP address lines are mapped directly to the VME port (DSP A0 = VME A2 !!), the lowest address for a VME master cycle would be 0x XX40 0000. To avoid this an 0xc0 0000 offset added to the master base address lifts the DSP address to 0x0100 0000 and "hides" the upper address byte behind the upper VME address bits which are specified in the master control register.
In the example below, the base address to access VMEbus hardware is: 0x2100 0000. This is the base address of a 64 Mbyte window. If you want the DSPs to access higher address locations on the VMEbus you have to increase the address value in the master control register.

If you set the bank size to maximum, the bank boundaries are:

SHC_EXT_BANK0:	0x0040 0000
SHC_EXT_BANK1:	0x1040 0000
SHC_EXT_BANK2:	0x2040 0000
SHC_EXT_BANK3:	0x3040 0000
VME_MASTER_BASE:	0x2100 0000 (DSP A24 becomes VME A26 and is hidden behind latched address in the master control register)

Contents of the assembler source file VME_MST.ASM:

```
        r0 = 0x00008009;                      Define Offset Master Window 0x 80xx xxxx
                                              VME bus lock = 0 (inactive)
                                              /LWORD = 0 (active)
                                              VME A1 = 0
                                              AM code = 9

        dm(SHC_EXT_BANK1 + 0x700) = r0;       write value to Master Control Register

        do check_flag1 until FLAG1_IN;        wait until hardware has stored address into latches
        nop;
check_flag1:
        nop;
        nop;
        r0 = dm(VME_MASTER_BASE + 0);         read data from VME address 0x8000 0000
```

It is important, to wait for FLAG1 is "1" again after a value was written to the master control register because the address value is stored in an external address latch.

7.2.2.2 DMA Master Cycles

During DMA cycles the SHARC is not always able to handle external requests for the SHARC bus. The workaround for this case is to request at first for the VME bus and if the bus is granted the DMA is started. This method also is described in the SHARC´s user manual.

Sequence:

Before performing master DMA cycles, bit 14 in the Control Register (A24 mode) has to be set to "1" and the SHARC has to request for the VME bus and verify the VME bus grant:

a) Allow any SHARC to lock the VMEbus	:	Set bit 14 in the Control Register (A24 mode).
b) Request for and lock the VME bus	:	Set bit 8 in the Master Control Register.
c) Verify VME bus grant	:	Read back value of DSP Flag 1 "1": VME bus is not yet granted "0": VME bus is granted and locked to WS2126

Now the DMA sequence can run without being disturbed by requests for the DSP bus (the VME bus is locked to WS2126 !). After that the VME bus lock is released by clearing bit 8 in the Master Control Register.

Important: If the VME bus is locked to WS2126 no other VME masters are allowed to get bus mastership until WS2126 has released the VME bus. If you need to perform SHARC DMAs on the VME bus, keep the bus lock times as short as possible !

Example Master Setup (SHARC DMA driven)

In this AD-Assembler example, a VME master setup is programmed. This is not a complete source file but will give you an idea how to prepare VME master mode for SHARC DMA driven cycles.

Contents of the H - file: See section "Core driven Master cycles"

Contents of the assembler source file VME_MST.ASM:

```
        /* request for VME bus */

        r0 = 0x00008109;                    Define Offset Master Window 0x 80xx xxxx
                                            VME bus lock = 1 (active)
                                            /LWORD = 0 (active)
                                            VME A1 = 0
                                            AM code = 9

        dm(SHC_EXT_BANK1 + 0x700) = r0;     write value to Master Control Register

        do got_bus until not FLAG1_IN;      wait for VME bus grant
        nop;
got_bus:
                . . .
        [ continue with SHARC master DMA cycles ]
                . . .
        /* release VME bus */

        r0 = 0x00008009;                    Define Offset Master Window 0x 80xx xxxx
                                            VME bus lock = 0 (inactive)
                                            /LWORD = 0 (active)
                                            VME A1 = 0
                                            AM code = 9

        dm(SHC_EXT_BANK1 + 0x700) = r0;     write value to Master Control Register

        do check_flag1 until FLAG1_IN;      wait until hardware has stored address into latches
        nop;
check_flag1:
        nop;
        nop;
```

7.2.2.3 VME Blocktransfer Master Cycles (BLT32)

WS2126 is capable of generating 32 bit aligned VME blocktransfer cycles (BLT32). This mode is simillar to the 7.2.2.2 Master DMA Cycles. The difference is that the Master Control Register is setup with address modifiers which correspond to specified VME blocktransfers. E.g.: 0x0B, 0x0F, 0x3B, 0x3F .
If VME blocktransfers are programmed, the programmer has to care for the transfer boundaries as defined in the VME specification: **The maximum length of one data block is 256 byte which are 64 longwords (32 bit)**. There is no hardware which watches the block length **-** if larger blocks should be transfered, the DSP software has to devide the amount of data into several blocks with maximum length of 64 longwords each.
Because the SHARC DSP is able to perform "chained DMA transfers", you can predefine data chaines which meet this restrictions very easy.
To prepare the board logic for executing blocktransfer cycles on the VME bus, bit 14 in the Control Register and bit 9 in the Master Control Register has to be set. After the blocktransfer is done, bit 9 in the Master Control Register should be cleared. *As long as bit 9 is set, all DSP cycles which access the VME address window are transfered to VME blocktransfer cycles !*
-> It is a good idea, to set the address modifier to standard or extended access after the BLT32 is done !

Sequence:

At first, bit 14 in the Control Register (A24 mode) has to be set in order to enable the DSPs to lock the VME bus.
Before performing master VME blocktransfer cycles, the SHARC has to prepare the board logic for this mode and request for the VME bus and verify the VME bus grant:

a) Allow any SHARC to lock the VMEbus	:	Set bit 14 in the Control Register.
b) Set BLT32 mode	:	Set bit 9 in the Master Control Register.
c) Request for and lock the VME bus	:	Set bit 8 in the Master Control Register.
d) Verify VME bus grant	:	Read back value of DSP Flag 1 "1": VME bus is not yet granted "0": VME bus is granted and locked to WS2126

Now the (chaining) DMA sequence can run without being disturbed by requests for the DSP bus (the VME bus is locked to WS2126 !). After that the VME bus lock is released by clearing bit 8 in the Master Control Register.

Important: **If the VME bus is locked to WS2126 no other VME masters are allowed to get bus mastership until WS2126 has released the VME bus. If you need to perform SHARC DMAs on the VME bus, keep the bus lock times as short as possible !**

Example BLT32 Master Setup (SHARC DMA driven)

```
        /* Set mode to VME blocktransfer and request for VME bus */

        r0 = 0x0000830f;                    Define Offset Master Window 0x 80xx xxxx
                                            BLT32 = 1 (active)
                                            VME bus lock = 1 (active)
                                            /LWORD = 0 (active)
                                            VME A1 = 0
                                            AM code = 0x0f ( Extended superv. BLT )

        dm(SHC_EXT_BANK1 + 0x700) = r0;     write value to Master Control Register
        do got_bus until not FLAG1_IN;      wait for VME bus grant
        nop;
got_bus:
                  . . .
        [ continue with SHARC master DMA cycles ]
                  . . .
        /* release VME bus and set mode to VME single cycles */

        r0 = 0x00008009;                    Define Offset Master Window 0x 80xx xxxx
                                            BLT32 = 0 (inactive)
                                            VME bus lock = 0 (inactive)
                                            /LWORD = 0 (active)
                                            VME A1 = 0
                                            AM code = 9 ( Extended nonpriv. data acc. )

        dm(SHC_EXT_BANK1 + 0x700) = r0;     write value to Master Control Register

        do check2_flag1 until FLAG1_IN;     wait until hardware has stored address into latches
        nop;
check2_flag1:
        nop;
        nop;
```

8. DSP Interface

The DSPs, the external memory and the input path to the FIFOs are connected to the DSP bus. If one DSP tries to access another DSP, it arbitrates for the DSP bus and moves the data. Depending on bits 11 and 13 of the control register, the DSPs are set to a fixed or a rotating bus priority scheme.
If a VME master accesses the DSPs or the external memory, it does a *host access* with the highest priority. In this case all DSPs recognize that a host request is pending and will release the DSP bus.
A host access can perform:
- Read from / write to the external memory
- Boot the DSPs by downloading programs into the DSPs (in host booting mode)
- Read from / write to the DSPs

8.1 External DSP Address Map

The DSP can access :
- the external memory
- the FIFOs (write only)
- other DSPs
- SHARC IO-Packs
- the VME bus (Master Mode)

Address	MS-Line	Description
$ 8 0000	-	Internal memory of DSP1
$ 10 0000	-	Internal memory of DSP2
$ 18 0000	-	Internal memory of DSP3
$ 20 0000	-	Internal memory of DSP4
$ 28 0000	-	Internal memory of DSP5
$ 30 0000	-	Internal memory of DSP6
$ 38 0000	-	Broadcast to all DSPs (write only)
$ 40 0000	MS0	External Memory (read / write)
$ 40 0000 + **[*MemSize*]**	MS1	FIFOs (write only)
$ 40 0000 + **[*MemSize*]**	MS1	FIFO Status (read only) ; **Status bits see chapter 6.2**
$ 40 0100 + **[*MemSize*]**	MS1	SHARC IO-Pack 1 (read / write)
$ 40 0200 + **[*MemSize*]**	MS1	SHARC IO-Pack 2 (read / write)
$ 40 0300 + **[*MemSize*]**	MS1	SHARC IO-Pack 3 (read / write)
$ 40 0400 + **[*MemSize*]**	MS1	SHARC IO-Pack 4 (read / write)
$ 40 0500 + **[*MemSize*]**	MS1	SHARC IO-Pack 5 (read / write)
$ 40 0600 + **[*MemSize*]**	MS1	SHARC IO-Pack 6 (read / write)
$ 40 0700 + **[*MemSize*]**	MS1	Master Control Register (write only)
$ 40 0000 + 2 x **[*MemSize*]**	MS2	Master Mode: VME address window (read / write)

MemSize is a constant which is written to the DSP’s SYSCON register (Bits 12 to 15). **[*MemSize*]** defines the window size for each predefined MSx signal. ***MemSize*** should be set to a value which is greater or equal than the amount of external memory which is installed on the board (MS0 window).
In case of using WS2126 as a VME master ***MemSize*** should be as large as the required VME address area but not smaller than the amount of the onboard memory (MS0).
The next free address behind the MS0 memory window accesses the FIFOs.
See **9.3** for an example on accessing memory and FIFO locations and refer to the SHARC User Manual for detailed information about the DSP memory map.

8.2 Internal DSP Address Map

The internal address map is defined by Analog Devices in the ADSP-2106x User`s Manual

9. Board Resources

Due to the fact that the DSPs work with 32 or 48bit wide data, the smallest addressable location is one longword = 32 bit. From VME point of view, the DSPs and the memory are addressable only 32 bit aligned and therefore the DSP address line A0 corresponds to the VME address line A2. This address shift by two should be kept in mind when talking about absolute DSP addresses in SHARC data manuals.

Example: The address of SHARC 4 within the DSP multiprocessor space is $ 20 0000. From VME point of view (address shift by two) the device address is $ 80 0000.

9.1 Memory

The memory on WS2126 is 48 bit wide and up to 512k deep (3 MByte). One word read/write from the DSPs always accesses one 48 bit word in the memory. For address information see chapter **7.**1 .
Due to the D32 boundary, VME can access the memory within ***two*** memory banks:
Bank 1: Start offset address is $100 0000. VME data D0..31 appear on the SHARC bus and in the memory as SHARC data D16..47 .
Bank 2: Start offset address is $180 0000. VME data D0..15 appear on the SHARC bus and in the memory as SHARC data D0..15 .

9.2 FIFOs

The FIFO on WS2126 is 16 bit wide and up to 64k deep. It can be written only from the DSPs. The FIFO status can de deternined by reading the status register (VME bus and FIFO). FIFO full, half full and empty are connected to this register. For address information see chapter **7.1** .
The FIFO can be written by the DSPs and read only by the VME bus. For address information see chapter **5.1** .

9.3 Example DSP Access To FIFO & Memory

In this AD-Assembler example, the size of the external memory is 64 k x 48. This is not a complete source file but will give you in idea how to access memory and the FIFOs.

Contents of the archive file: SHARC.ACH

```
!_________External RAM__________
!Bank 0: 64k external RAM
.segment /dm /ram /begin = 0x00400000 /end = 0x0040ffff /width=32 extdata1;

!_________FIFO__________
!Bank 1: FIFO
.segment /dm /ram /begin = 0x00410000 /end = 0x00410fff /width=16 /unint dm_fifo;
```

Contents of the assembler source file MEM.ASM

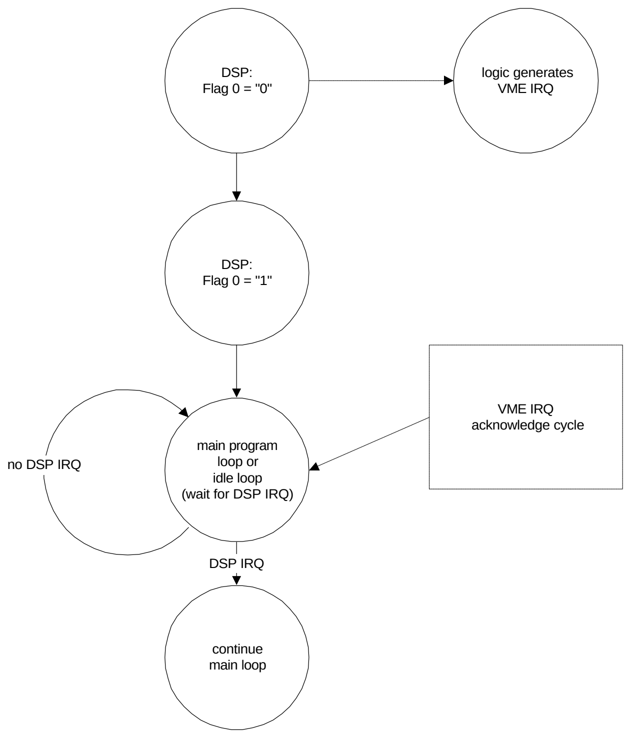
```
.segment/extdata1 ;        /* fifo segment */
.var mem1;
.endseg

.segment/dm_fifo ;         /* fifo segment */
.var fifo;
.endseg

/*  SYSCON:  $ xxxx 3xxx    memory size 64 k x 48 written into SYSCON register */

r0 = dm(SYSCON);           /* get status of SYSCON */
r0 = bclr r0 by 3;         /* set external data access to 32 bit */
r0 = bclr r0 by 9;         /* set bank 1 to 32 bit data */
r0 = bset r0 by 8;         /* set bank 0 to 48 bit data */
r0 = bset r0 by 12;        /* msize 64k = 2 power 16;  msize = 16-13 => msize = 3 [Bits 15..12] */
r0 = bset r0 by 13;
r0 = bclr r0 by 14;
r0 = bclr r0 by 15;
dm(SYSCON) = r0;           /* writing new value to SYSCON */

/* write data to fifo */

r3 = 0x12345678            /* simple test data */
r3 = dm(fifo)              /* write into fifo */

/* write increment pattern to memory */

i0 = extdata1              /* set up index register */
m0 = 1;                    /* increment value */
r3 = 0xffffffff            /* start value for data */

LCNTR = r2, DO loop UNTIL LCE;
  dm(i0,m0) = r3;  /* write pattern */
loop: r3 = r3 + 1;         /* increment write pattern */
```

10. VME Interrupts

10.1 General Concept

Any DSP on WS2126 can request an interrupt on the VME bus by pulling down its flag 0 for one or more cycles. This action causes an interrupt on the VME bus with the programmed IRQ level.
During interrupt service on the VME bus the control logic on WS2126 generates an internal interrupt for the DSP (IRQ0) which had initiated the IRQ on VME side. The DSP interrupt therefore acknowledges a successful VME interrupt service.

Flow chart: DSP requests interrupt service on VME



To enable the interrupt feature on WS2126 the following actions have to be done:
a) Define a VME interrupt level to request for (bits 2..0 in the control register; bit 2: MSB, bit 0: LSB).
b) Define a board ID that is single in the VME rack. This is important if more than one WS2126 reside in the same VME rack.
c) Enable the IRQs by setting bit 12 in the control register.

10.2 Status ID During VME Interrupts

Data Bit #	Function
7	Board ID, Bit4
6	Board ID, Bit3
5	Board ID, Bit2
4	Board ID, Bit1
3	Board ID, Bit0
2	DSP ID, Bit2
1	DSP ID, Bit1
0	DSP ID, Bit0

10.3 Example

Some fragmentary DSP assembler code for requesting interrupts on VME

Setup procedures:

```
/* prepare flag0 for output */
bit set astat FLG0;
bit set mode2 FLGO0;

bit set mode2 IRQ0E;      /* make IRQ0 edge sensitive */
bit set imask IRQ0I;      /* enable IRQ0 */
bit set mode1 IRPTEN      /* global IRQ enable */
```

Main program starts an IRQ on VME:

```
...
bit clr astat FLG0;               /* set interrupt request */
...
```

DSP Interrupt service routine:

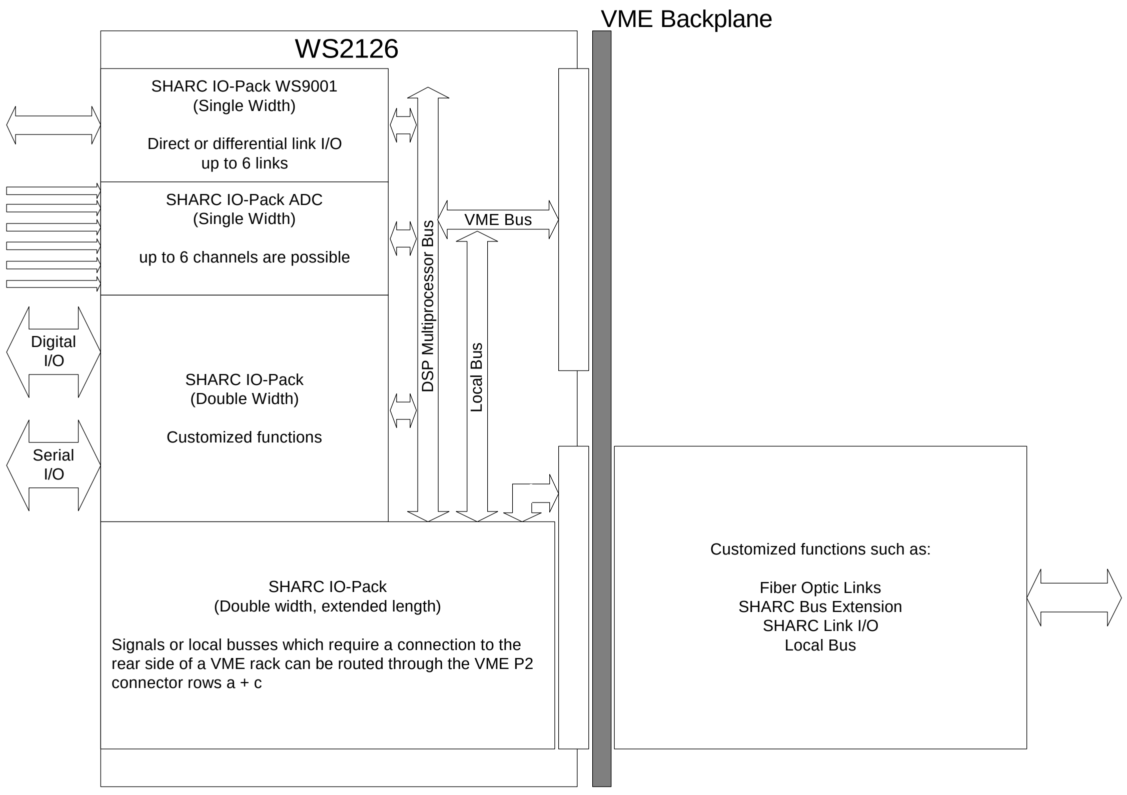
```
irq0_service:
  bit set astat FLG0;     /* clear interrupt request */
  ...

rti;
```

11. SHARC IO-Packs

11.1 General Concept

In general the link ports of the DSPs support fast communication between the DSPs and external data sources/sinks up to 40 Mbyte/s for each link *). To keep all options of communication paths, all 6 links of all 6 DSPs are routed to the SHARC IO-Pack front end connector.
A link bus with three links is routed to each DSP by bus jumpers. This allows DSP intercommunication without offboard hardware connections.
A very flexible I/O concept has been realized by introducing the SHARC IO-Packs on WS2126. Each SHARC DSP now may "carry" a dedicated I/O interface. The range of I/O options may vary from simple link interconnections up to ADCs/DACs I/Os and fiber optic link interfaces.
In customized applications a SHARC IO-Pack also can cover more than one slot on WS2126. Due to the fact that the DSP bus connectors, the local bus connector and the P2 extension connector can be used by a SHARC IO-Pack configuration, the number of I/O alternatives is numerous.
Depending on the kind of SHARC IO-Pack the SHARC signals which are routed to the front connectors may vary. Please refer to the specification of the SHARC IO-Pack which is in use.

The picture below picks up a few I/O options to give you an idea how to use SHARC IO-Packs.



*) The maximum speed is documented in the latest ADSP-2106x data sheet from Analog Devices.

11.2 SHARC IO-Pack Interface

All SHARC IO-Packs are memory mapped into the VME board address space and into the DSP external address area. For VME address info please refer to chapter **7.1.1** and for the DSP address table look at chapter **8.1.**
The SHARC IO-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 I/O ressources: **LINK** ports, **SPORT** channels, **IRQ1** and **IRQ2**, **FLAG2**, **FLAG3**, and to most signals of the parallel local SHARC bus. Please refer to the detailed connector-description in chapter **12.1** .
32 bit of the data bus (**SD16** ... **SD47**), six address lines (**SA0** ... **SA5**) and a select signal (**/PIGGY_CS**) are routet to each SHARC IO-Pack.. The signal **/PIGGY_CS** is generated by the glue logic on board and follows the address mapping defined in **7.1.1** and **8.1** .
With the use of the signals **ADRCLK**, **/RD**, **/WR**, and **/ACK** it is possible to interface synchronous and asynchronous devices to the SHARC bus, and to generate waitstates if necessary. The lines **/DMARx** and **/DMAGx** interface hardware-controlled DMA channels.
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 the SHARC IO-Pack connectors.

For more detailed description of the SHARC IO-Pack interface please refer to:

📖[3] SHARC IO-Pack Interface
Hardware Manual
Wiese Signalverarbeitung GmbH

and

ADSP-2106x SHARC User´s Manual
Analog Devices

12. Connectors & Switches

12.1 SHARC IO-Pack Front & Rear Connectors

Pin-Nr.	Signal	Description (Front)
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 (Rear)
1	SA0	SHARC address bus (VME: A2)
2	SA3	SHARC address bus (VME: A5)
3	SA1	SHARC address bus (VME: A3)
4	SA4	SHARC address bus (VME: A6)
5	SA2	SHARC address bus (VME: A4)
6	SA5	SHARC address bus (VME: 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 (VME: D1)
16	SD32	SHARC data bus (VME: D16)
17	SD16	SHARC data bus (VME: 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 (VME: D2)
31	/RD	Memory Read Strobe
32	ADRCLK	Synch. Address Clock
33	SD39	SHARC data bus (VME: D23)
34	/WR	Memory Write Strobe
35	SD37	SHARC data bus (VME: D21)
36	SD38	SHARC data bus (VME: D22)
37	SD35	SHARC data bus (VME: D19)
38	SD36	SHARC data bus (VME: D20)
39	SD34	SHARC data bus (VME: D18)
40	SCLK	System Clock (40 MHz)
41	SD33	SHARC data bus (VME: D17)
42	/DMAG2	DMA Grant 2 (Channel 8)
43	SD20	SHARC data bus (VME: D4)
44	/DMAG1	DMA Grant 1 (Channel 7)
45	SD30	SHARC data bus (VME: D14)
46	SD31	SHARC data bus (VME: D15)
47	SD28	SHARC data bus (VME: D12)
48	SD29	SHARC data bus (VME: D13)
49	SD26	SHARC data bus (VME: D10)
50	SD27	SHARC data bus (VME: D11)
51	SD24	SHARC data bus (VME: D8)
52	SD25	SHARC data bus (VME: D9)
53	SD47	SHARC data bus (VME: D31)
54	SD23	SHARC data bus (VME: D7)
55	SD22	SHARC data bus (VME: D6)
56	SD46	SHARC data bus (VME: D30)
57	SD21	SHARC data bus (VME: D5)
58	SD45	SHARC data bus (VME: D29)
59	SD19	SHARC data bus (VME: D3)
60	SD44	SHARC data bus (VME: D28)
61	SD42	SHARC data bus (VME: D26)
62	SD43	SHARC data bus (VME: D27)
63	SD41	SHARC data bus (VME: D25)
64	SD40	SHARC data bus (VME: D24)

12.2 Local Bus

Pin-Nr.	Signal	Description
1	D31	
2	LOC_CLK	Local Clock
3	D30	
4	GND	
5	D29	
6	/LOC_WR	Local Write
7	D28	
8	/LOC_CS	Local Select
9	D27	
10	/CTLRST	Local Reset
11	D26	
12	/Fifo Full	Fifo Full
13	D25	
14	/Fifo Half	Fifo Half Full
15	D24	
16	/Fifo Empty	Fifo Empty
17	D23	
18	LOC_A22	
19	D22	
20	LOC_A21	
21	D21	
22	LOC_A20	
23	D20	
24	LOC_A19	
25	D19	
26	LOC_A18	
27	D18	
28	LOC_A17	
29	D17	
30	LOC_A16	
31	D16	
32	LOC_A15	
33	D15	
34	LOC_A14	
35	D14	
36	LOC_A13	
37	D13	
38	LOC_A12	
39	D12	
40	LOC_A11	
41	D11	
42	LOC_A10	
43	D10	
44	LOC_A9	
45	D9	
46	LOC_A8	
47	D8	
48	LOC_A7	
49	D7	
50	LOC_A6	
51	D6	
52	LOC_A5	
53	D5	
54	LOC_A4	
55	D4	
56	LOC_A3	
57	D3	
58	LOC_A2	
59	D2	
60	LOC_A1	
61	D1	
62	LOC_A0	
63	D0	
64	* free *	

12.3 DSP Bus Data & Address Connector

Pin-Nr.	Signal	Description (Data)
1	SHARC D1	
2	SHARC D0	
3	SHARC D3	
4	SHARC D2	
5	SHARC D5	
6	SHARC D4	
7	SHARC D7	
8	SHARC D6	
9	SHARC D9	
10	SHARC D8	
11	SHARC D11	
12	SHARC D10	
13	SHARC D13	
14	SHARC D12	
15	SHARC D15	
16	SHARC D14	
17	SHARC D17	
18	SHARC D16	
19	SHARC D19	
20	SHARC D18	
21	SHARC D21	
22	SHARC D20	
23	SHARC D23	
24	SHARC D22	
25	SHARC D25	
26	SHARC D24	DSP Data
27	SHARC D27	
28	SHARC D26	
29	SHARC D29	
30	SHARC D28	
31	SHARC D31	
32	SHARC D30	
33	SHARC D33	
34	SHARC D32	
35	SHARC D35	
36	SHARC D34	
37	SHARC D37	
38	SHARC D36	
39	SHARC D39	
40	SHARC D38	
41	SHARC D41	
42	SHARC D40	
43	SHARC D43	
44	SHARC D42	
45	SHARC D45	
46	SHARC D44	
47	SHARC D47	
48	SHARC D46	
49	+5V	
50	+5V	
51	* free *	
52	/CTLIRQ	Expansion Interrupt Request
53	* free *	
54	/CTLIGR	Expansion Interrupt Grant
55	TD_END	JTAG
56	/CTLCE	Expansion Select
57	BTDI	JTAG
58	/CTLRQ	Expansion Bus Request
59	/BTRST	JTAG
60	/CTLGR	Expansion Bus Grant
61	BTCK	JTAG
62	* free *	
63	BTMS	JTAG
64	* free *	

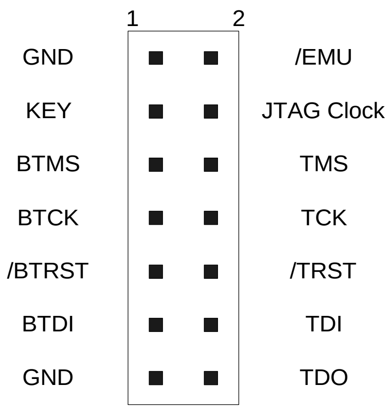
Pin-Nr.	Signal	Description (Address)
1	SHARC A1	
2	SHARC A0	
3	SHARC A3	
4	SHARC A2	
5	SHARC A5	
6	SHARC A4	
7	SHARC A7	
8	SHARC A6	
9	SHARC A9	
10	SHARC A8	
11	SHARC A11	
12	SHARC A10	
13	SHARC A13	
14	SHARC A12	
15	SHARC A15	
16	SHARC A14	
17	SHARC A17	
18	SHARC A16	DSP Addresses
19	SHARC A19	
20	SHARC A18	
21	SHARC A21	
22	SHARC A20	
23	SHARC A23	
24	SHARC A22	
25	SHARC A25	
26	SHARC A24	
27	SHARC A27	
28	SHARC A26	
29	SHARC A29	
30	SHARC A28	
31	SHARC A31	
32	SHARC A30	
33	+5V	
34	/MS0	Memory Select 0
35	/MS3	Memory Select 3
36	/MS2	Memory Select 2
37	/BMS	Boot Memory Select
38	GND	
39	+5V	
40	SACK	Acknowledge
41	PAGE	DRAM Page Signal
42	GND	
43	+5V	
44	/SRD	DSP Read Signal
45	SACLK	Synchr. Address Clock
46	GND	
47	+5V	
48	/SWR	DSP Write Signal
49	/SW	Synchr. Write
50	GND	
51	+5V	
52	REDY	Host Bus Acknowledge
53	CTL_CLK	Derivated System Clock
54	GND	
55	+5V	
56	/DMAR1	DMA Request To SHARCs 1
57	/DMAG1	DMA Grant From SHARCs 1
58	GND	
59	+5V	
60	/DMAR2	DMA Request To SHARCs 2
61	/DMAG2	DMA Grant From SHARCs 2
62	GND	
63	+5V	
64	/CTLRST	Reset

12.4 P2 Utility Bus

The rows *a* and *c* on the P2 connector are not used and directly routed to X7 (64 pins). User defined logic or special I/O requirements can be implemented by connecting onboard devices to external building blocks.

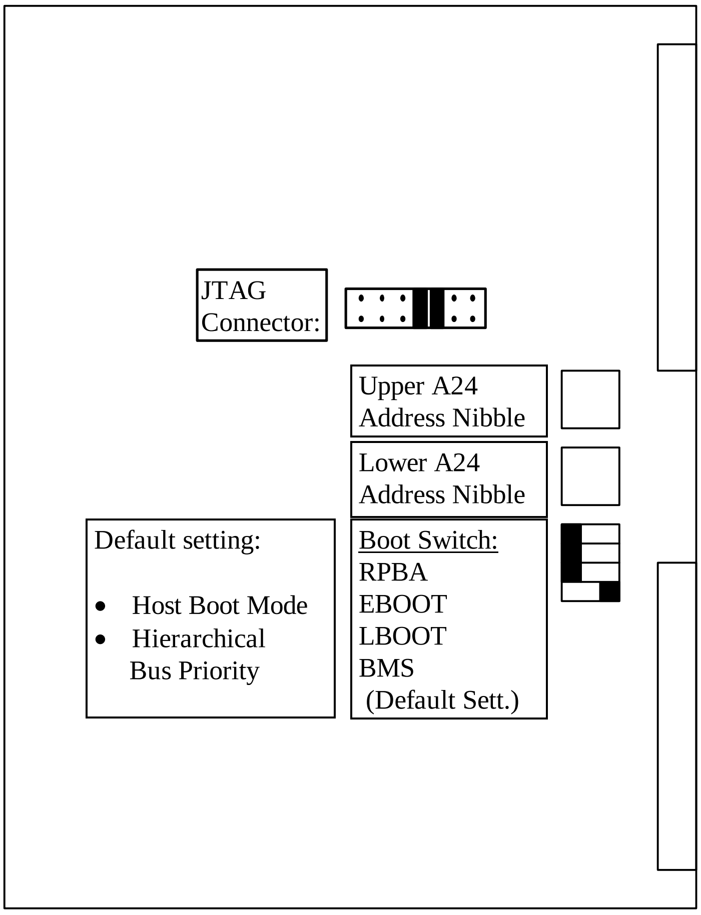
12.5 JTAG

A JTAG interface for debugging the DSP software and status analizing is implemented. The EZ-ICE from Analog Devices is supported.
The pin location "KEY" has no pin !

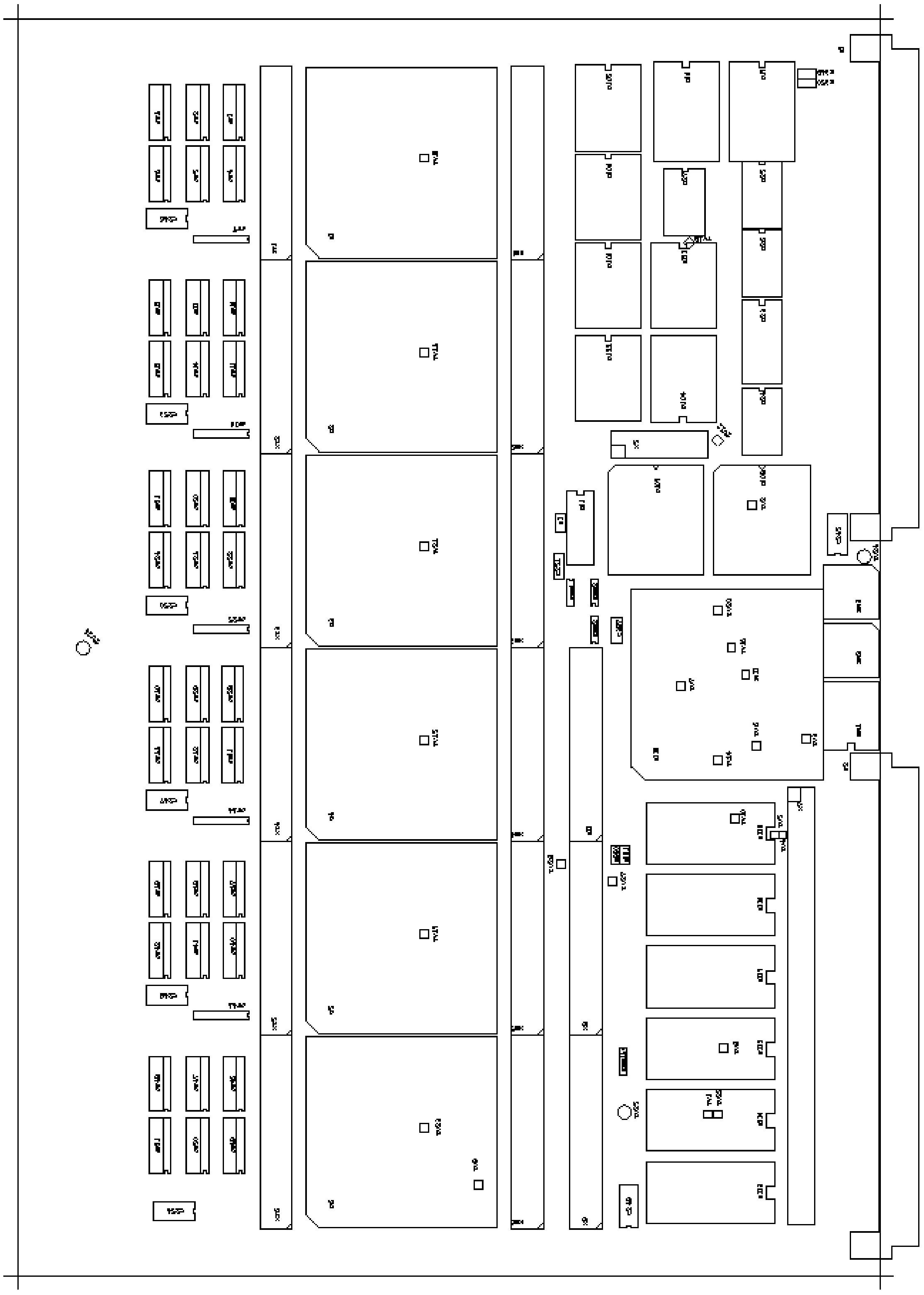


13. Board Layout

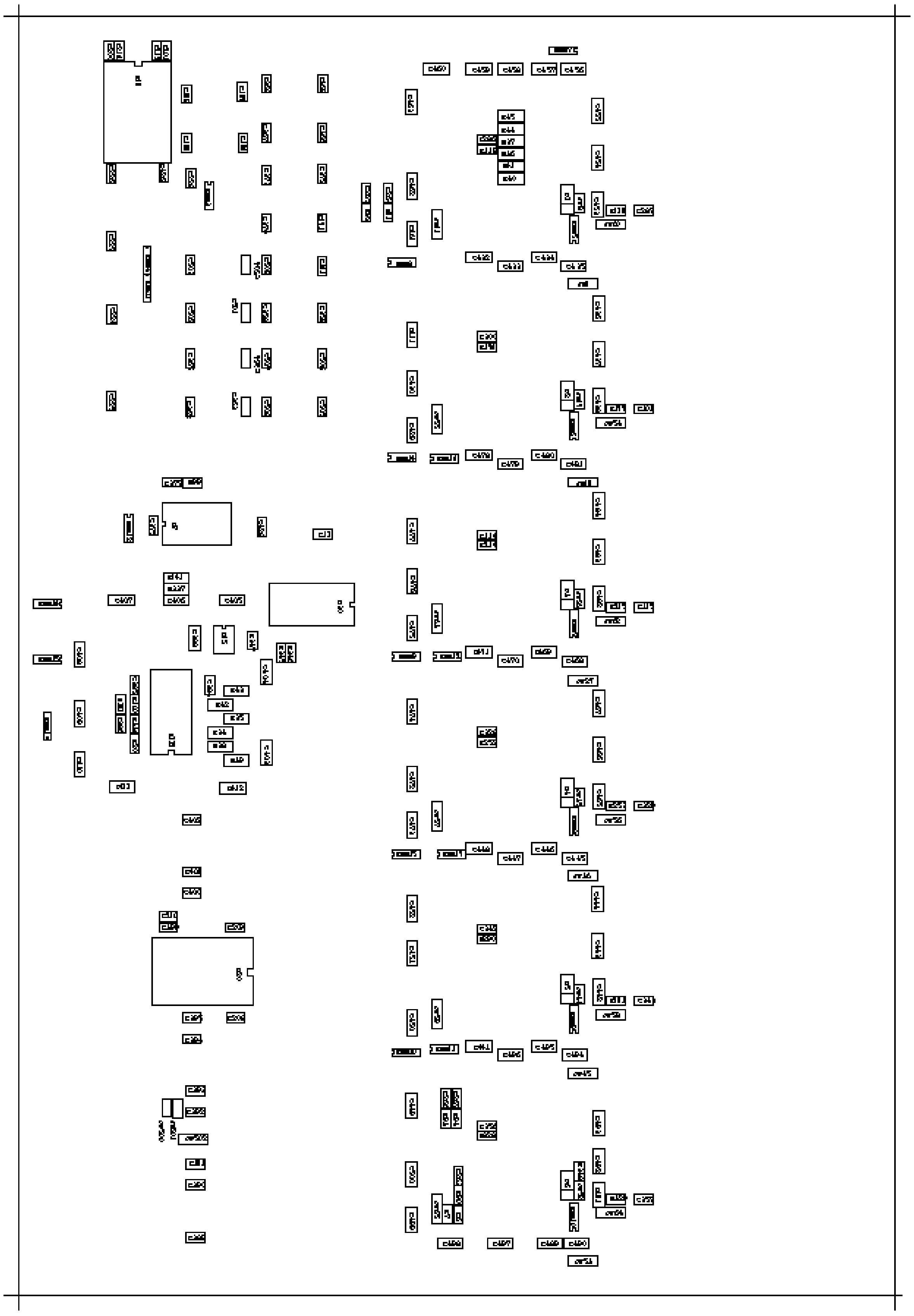
13.1 Switches



13.2 Component Side



13.3 Solder Side



14. Technical Data

14.1 Power Requirements

The WS2126 requires app. 7A at 5V. The value for the current may vary in dependence of the SHARC IO-Packs which are used.

15. Trouble Shooting / Support

Voltages - Are all voltages in the VME rack within it´s specfied limits ?

A24 access - Do the rotary switches correspond to bits A23..A16 of the active VME address ?

A32 access - If an access to WS2126 is not possible, please check whether the "A32 enable"-bit in the Control Register is set.

DSP access - If the access to the DSP(s) does not work, please check if they are in the "Run" state. Another point to verify is that if one DSP is owner of the DSP bus and did a "bus lock". This software command can lock the bus to the current DSP and no other device - the VME host not either - will get the DSP bus granted. The only possibility to change this status is to reset the DSP in question.

DSP host packing mode - If the VME access to the DSP(s) returns unexpected results, verify that the DSP works in the correct "host packing mode".

DSP bus priority - In most cases the DSPs work in a "hierarchical bus priority". Every time the DSP with the highest priority requests for the bus, all other DSPs are forced to stop their own bus activity. In some cases a "rotating priority" scheme is more suitable. The bus priority scheme can be modified by software (Control Register) or by hardware (DIP switch).

VME IRQ - If WS2126 does not generate interrupts, please verify whether the bit "Interrupt Enable" in the Control Register is enabled. An example of how to generate VME interrupts by DSP software is given in chapter **11.3 .** Check also if the IACKIN-IACKOUT daisy chain on the VME backplane is routed to the slot with WS2126. In most cases jumpers are used to route the daisy chain through empty VME slots.

VME Master - If WS2126 is not able to do any master cycles, check if the BGIN3IN-BGIN3OUT daisy chain on the VME backplane is routed to WS2126. In most cases jumpers are used to route the daisy chain through empty VME slots.
If the master cycles result in unexpected addresses, please verify that the DSP´s *MSIZE* and the master control register is set correct.
VME blocktransfers: A maximum amount of 64 longwords can be transfered within one block. If this limit is exceeded, it is possible that the corresponding slave overwrites previous written data because often only 8 bit counters are used to generate internal addresses.
If unexpected data is read during VME master accesses, check if the "VME master timeout" (enable status: bit 12 in the VME Offset Register, error status in bit 15 in the VME Offset Register) is set.

We offer support for all of our products. In case you have questions or you need additional information to a specific subject you can contact us:

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
	sales@wiese.de	Pricing information, delivery times
Web:	**http://www.wiese.de**	Product overview, technical specifications, anouncements, news

List Of Literature

[1] ADSP-2106x SHARC User´s Manual, Analog Devices.
For newest information and SHARC-FAQs look at: http://www.analog.com

[2] The VMEbus Specification, VITA
For the list of publications/specifications look at: http://www.vita.com

[3] SHARC IO-Pack Interface Hardware Manual, Wiese Signalverarbeitung GmbH