Version 1.2b

SPAC Master VME64 boards
User Guide

M. Dhellot, M. Escalier, B. Laforge,
O. Le Dortz*, D. Martin, J.M. Parraud

LPNHE Paris

* Contact Person, Olivier.LeDortz@lpnhep.in2p3.fr

1 Introduction

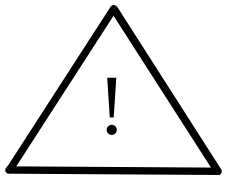
The front-end electronics of the ATLAS liquid argon calorimeters will be driven through an optical serial link between read-out crates and the front-end crates. This link known as SPAC (Serial Protocol for the ATLAS calorimeters) will be used to load, to update or to check the various registers and memories of the various front-end boards. Each serial network is consisted of one master and multiple slaves. Detailed information about the protocol and the serial slave ASICs can be found in [1].

Several versions of master prototypes were already produced, especially for lab test purposes [2]. We present here the specifications of the new SPAC Master VME64 boards, implementing the VME64x interface used in the LARG Read Out Crate environment [3].

The SPAC Master VME64 6U module, designed for lab test purposes can drive, through optical fibres two individual SPAC networks (one of these networks can also be driven in LVDS format). It is designed to be inserted into a “legacy” VME crate, or into a 6U slot of VME64x 9U after a **small hardware operation** on the board (see section 2.3 page 6).

The SPAC Master VME64 9U module is the production board for the ATLAS experiment, designed for a VME64x 9U crate. It can drive optically four individual SPAC networks.

CAUTION



- Always leave caps on optical fibres connectors when unused
- Check the hardware configuration section (page 6) before inserting a 6U board into a VME64 crate (with 5-row backplane connectors)
- When inserting a 6U module into the 6U subrack of a VME64x 9U crate, remove the plastic body of the front panel upper extractor before inserting the board. Also mind the crate IEEE 1101.10 EMC springs on the neighbour boards or crate structure when removing the board.

2 Hardware Description

Figure 1 illustrates the main elements of the SPAC Master VME64 boards. They are essentially consisted of two or three programmable devices:

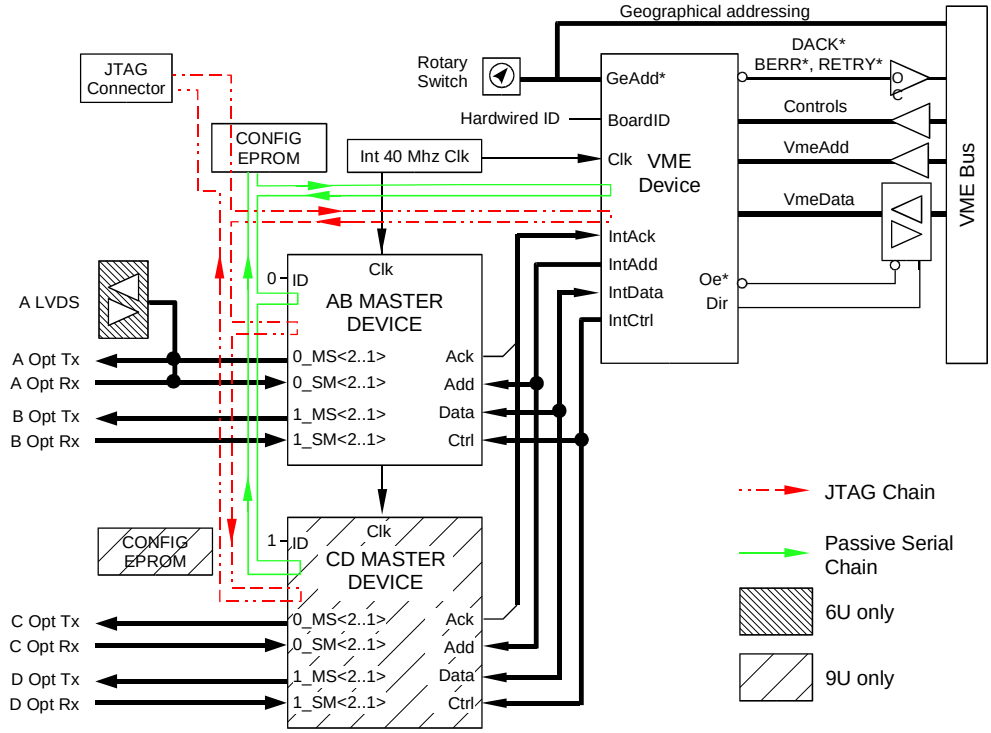


Figure 1: SPAC Master VME64 modules functional diagram

- A *VME* device, housed in an Altera ACEX EP1K50 device, that handles the VME decoding and encoding tasks and implements the CR/CSR features, based on a generic VME64x interface developed for the ROD group by Annie Léger (University of Geneva),

- one or two *MASTER* devices (AB and CD), housed in Altera ACEX EP1K50[1], each handling two SPAC optical networks (or blocks). Each block is made of four optical fibres adapted to be linked to front-end crate controllers. The A block of the 6U board can alternatively drive a LVDS network.

- The above mentioned devices are configured at the board power-up with one or two EPROMs (Altera EPC2), through a passive serial daisy chain. The EPROMs and/or the programmable devices can also be reprogrammed by an Altera download cable, via a JTAG daisy chain. The capability of reprogramming the EPROMs by VME (via the VME device driving the JTAG chain) is also considered in the future.

2.1 Master device

The goal of the master devices (Figure 2) is to emit command frames to (and receive reply frames from) SPAC slaves. The SPAC frame to send on the serial bus has first to be stored into the *Emitter FIFO* through VME. The transmission of the frame on the serial output is initiated by sending a dedicated VME command to the master device. When serial frames are received from a slave on the serial inputs, they are stored into the *Receiver FIFO*, that can be read back through VME. While the *Emitter FIFO* can only store a unique frame at a time, the *Receiver FIFO* can contain several frames. The *Emitter* and *Receiver FIFOs* are

1 For the production phase, ACEX EP1K50 might eventually be replaced by EP1K30

512-element deep.

The serial frames are encoded in Manchester code. The master commands are sent by the master on the MS line, while the answers from the slaves are written on the SM line. But, for reliability purposes, the SM and MS lines are duplicated. At any time, the master writes its serial frames on either MS1 or MS2, while the slaves send their replies on both SM1 and SM2 lines. The MS line on which the master will send the frames, and the SM line on which it will read the incoming frames, are selectable through a VME configuration register (SelectMS and SelectSM signals in Figure 2).

In SPAC3 protocol, the master can also send on the MS lines a carrier when no frame is being sent, i.e. in idle state (MS1Carrier and MS2Carrier signals in Figure 2).

The version of SPAC protocol that the transmitter and the receiver should handle is specified by a configuration signal (SPAC2_3 signal in Figure 2). By default, the master devices are configured in SPAC3 mode (protocol handled by the production batch of slave ASICs), but the SPAC2 protocol is still maintained for compatibility with the first ASIC prototype (SPAC2Slave).

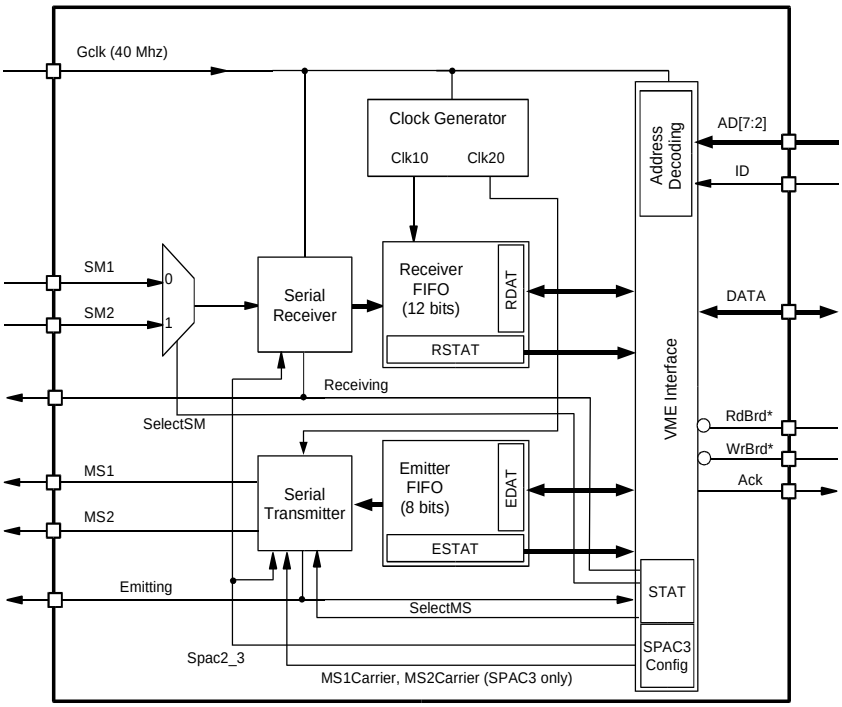


Figure 2: Master device functional schematic

2.2 SPAC network connections

The optical links are designed to drive a front-end crate controller. Table 1 indicates how to connect the master to the controller.

on SPAC Master	on FEC Controller	Wavelength
Tx*i*1 fiber	MS1 fiber (ORx1)	1300 nm
Tx*i*2 fiber	MS2 fiber (ORx2)	
Rx*i*1 fiber	SM1 fiber (OTx1)	850 nm
Rx*i*2 fiber	SM2 fiber (OTx2)	

Table 1: Optical connections between the master and the controller
(*i*=0 to 3 or A to D)

Let us notice that, as the radiation qualified transceivers and receivers of the controllers do not operate at the same wavelength, the optical components of the master are also of different types.

- LVDS connector (A block of the 6U board only)

The LVDS connector (Table 2 is a HE10 type connector, compatible with the SPAC bus. One or several FEBs, Calibration or Tower builder boards can be connected directly to this board for lab tests.

Signal Name	Pin Number		Signal Name
GND	1	2	GND
SM2+	3	4	SM2-
SM1+	5	6	SM1-
MS2+	7	8	MS2-
MS1+	9	10	MS1-
GND	11	12	GND
RESERVED	13	14	GND

Table 2: LVDS connector pin-out

2.3 Hardware configuration of the SPAC Master VME64 6U board[2]

- The selection between optical and LVDS connection for the A block is determined by the position of the S4 switch
- The programmable devices need 3.3 and 2.5 volts power supplies. The 3.3 volts supply is either derived from the VME 5 volts supply or directly connected to the 3.3 volts supply of a VME64 baseplane.

- Board in a standard VME crate (3-row baseplane connectors)
  - The VME base address of the board is defined by the rotary switch RD1 (Figure 3, ref 1)
  - Put F3 fuse to enable the board 3.3 volts power supply to be derived from the VME 5 volts supply

- Board in a VME64 crate (5-row baseplane connectors and 3.3 volts supply available)
  - The VME base address of the board is determined by its slot position (through the VME GEA[4:0]* lines). To avoid interference with the geographical addressing mechanism, RD1 (Figure 3, ref 1) must be left to position 0.
  - **Check that both fuses F2 and F3 are not cabled at the same time**. The two following configurations are possible:
    - F3 removed and F2 cabled: board 3.3v supply from to the VME 3.3 volts (optimal for VME64x backplane, but will not work in a standard VME)
    - F3 cabled and F2 removed: board 3.3v supply derived from VME 5 volts supply (compatible with a standard VME crate)

2.4 Front panel of the SPAC Master VME64 6U board

- 5, 3.3, 2.5 power supplies LED indicators
- RST button manual board reset
- VME LED the board is accessed by VME
- DTACK LED the board issued a VME acknowledge

2 see the appendix for information on the 9U board

- IRQ LED the board issued a VME interrupt request
- TX01/TX02 optical transmitters of the A master (MS1/MS2 lines)
- RX01/RX02 optical receivers of the A master (SM1/SM2 lines)
- TX11/TX12 optical transmitters of the B master (MS1/MS2 lines)
- RX11/RX12 optical receivers of the B master (SM1/SM2 lines)
- Em0/Em1 LED The A master / B master is emitting a frame on its active MS link
- Rec0/Rec1 The A master / B master is receiving a frame on its active SM link
- Det0/Det1 The A master / B master is receiving a carrier on its active SM link
- LVDS0 LVDS connector of the A master

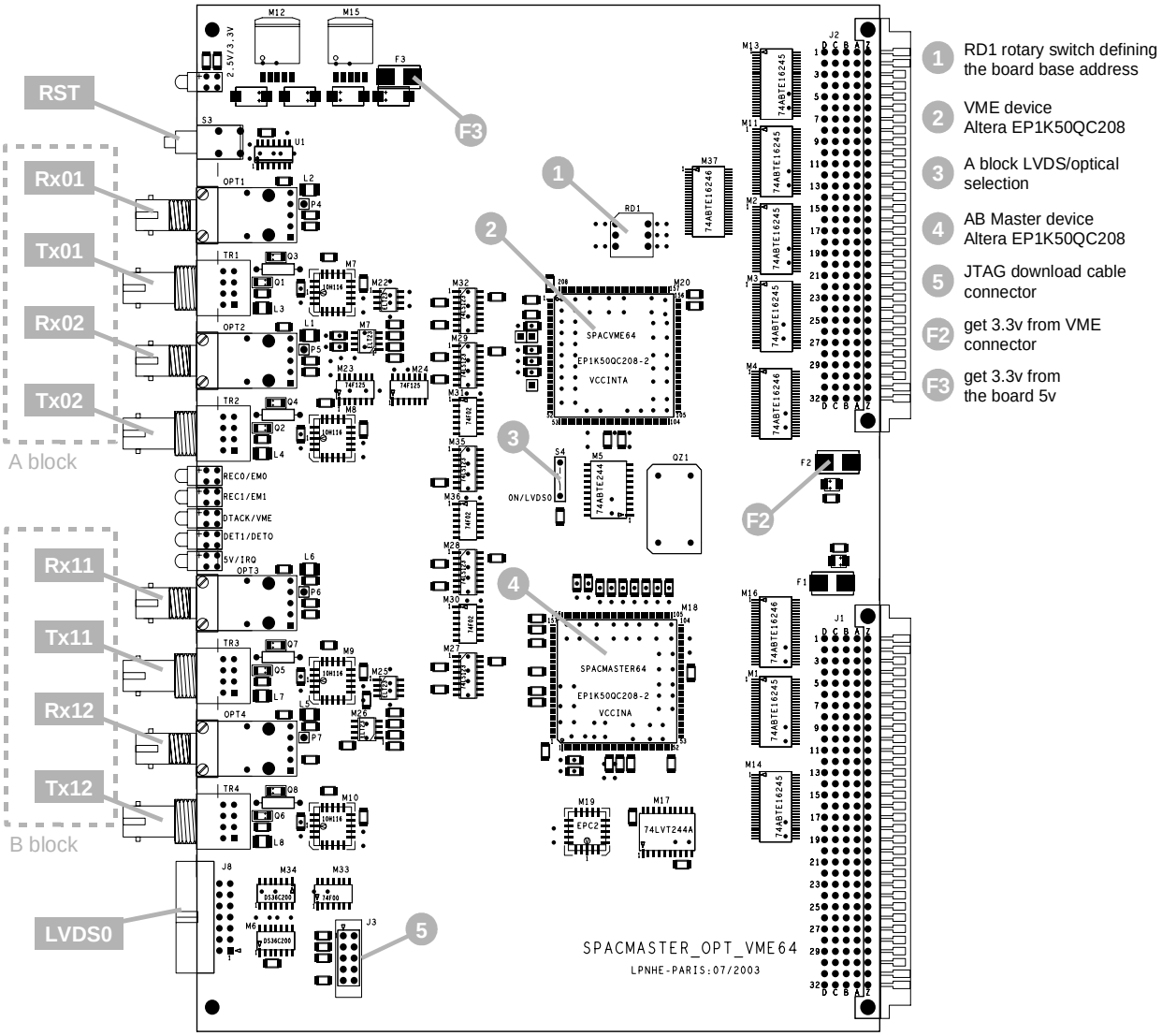


Figure 3: Components of the SPAC Master 6U board

3 VME interface

The SPAC Master VME64 boards handle A32/D16-D32 transfers and CR/CSR transfers. In a VME64 crate, the address ranges handled by the boards are determined by the VME lines GEA[4:0]*. When a 6U board is inserted in a standard VME crate, its rotary switch RD1 is used to replace the geographical addresses lines GEA[4:1]* (GEA0* remains at its default value 1).

3.1 CR/CSR space

The VME device implements the CR/CSR capabilities of the board (D08/D16/D32 access, with address modifier 0x2F). The CR/CSR space is a 0x80000 byte memory space, the base address of which is determined by the VME slot in which the board is plugged in such that:

CrBase[23:19] = GEA[4:0] (other bits to 0)

and, in a legacy VME crate, by the value of the rotary switch RD1, such that:

CrBase[23:20] = RD1[3:0] (other bits to 0)

Examples:

- The board is in slot 11 decimal of a VME64 crate (and RD1 is set to 0): the CR/CSR occupies an address space of 0x80000 bytes starting at 0x580000 (range 0x580000-0x5FFFFF)
- The board is in a standard VME crate, RD1 is set to C: the CR/CSR occupies an address space of 0x80000 bytes starting at 0xC00000 (range 0xC00000-0xC7FFFF)

3.2 A32 user space

Some internal registers handled by the VME device and the registers of the two or four master blocks are controlled via A32 VME accesses (address modifiers 0x09, 0x0d and, for block transfers 0x0b and 0x0f). The board occupies a page of 0x1000000 bytes, starting at the base address A32Base. At startup, A32Base is determined by the geographical addressing of the board and/or the rotary switch RD1 position. However, this base address can be changed from its default value by an access to the CR/CSR space. The default value of the base address is such that:

A32Base[28:24] = GEA[4:0] (other bits to 0)

and, in a legacy VME crate, by the value of the rotary switch RD1:

A32Base[28:25] = RD1[3:0] (other bits to 0)

Examples:

- The board is in slot 11 decimal of a VME64 crate (and RD1 is set to 0): the A32 space starts at the base address 0x0B000000 (range 0x0B000000-0x0BFFFFFF)
- The board is in a standard VME crate, RD1 is set to C: the A32 space starts at the base address 0x18000000 (range 0x18000000-0x18FFFFFF)

A list of the accessible register is given in Table 3.

The A, B, C and D master blocks are completely independent and could be seen as four individual VME «sub-boards», of address space starting at A32Base+0x200, +0x240, +0x280 and +0x2C0, except for the V_RST register that affects all the blocks.

Compared with the previous prototype of master boards, the addresses of the different registers and the address modifiers used to access them have changed, but their functions are more or less identical. However, the CSR *registers now became* STAT *registers, which are read only. The bits that could be written previously in the* CSR *registers must now be written in the* CONF *registers. Especially, in the software libraries distributed for the previous boards, the* RunEnable *signal was set or unset (SPACRun procedure) by writing the* CSR *register. To guarantee a good operation of the board,* RunEnable *must now be accessed through the* CONF *register.*

On the other hand, all the master registers can now be accessed in D16 or D32 mode.

The value of the A32 base address initial value, with respect to the geographical addressing or the rotary switch might change in a future version of the VME device.

Name	Address	Mode	Description
Registers handled by the VME device, D32 only			
V_DUM	+000	R/W	a dummy 32 bits test register
V_RST	+004	W only	a dummy write at this address resets the whole board
V_STAT	+008	R only	Reserved
V_BID	+00C	R only	8 bits hardwired board ID (also available in CR space)
V_CTL	+010	W only	Reserved
V_VER	+03C	R only	16-bit-word coding the VME interface version
V_IRQ	+180	-	Reserved
Registers handled by the A Master block, D16 or D32			
A_ID	+200	R only	A Master Identifier
A_STAT	+208	R only	A Master Status word
A_CONF	+210	R/W	A Master Configuration register
A_RDAT	+220	R/W	A Master Receiver FIFO content
A_RSTAT	+224	R only	A Master Receiver FIFO status
A_RELOAD	+230	W only	a dummy write at this address reloads the A Master Emitter FIFO
A_EDAT	+238	R/W	A Master Emitter FIFO content
A_ESTAT	+23C	R only	A Master Emitter FIFO status
In the same way, the registers of the B,C and D master block are identical to the A ones, with an offset +040 between blocks. For example, with the ID registers:			
B_ID	+240	R only	B Master Identifier
C_ID	+280	R only	C Master Identifier
D_ID	+2C0	R only	D Master Identifier

Table 3: A32 VME registers of the SPAC Master VME64 boards
(the addresses are the offsets from the board A32 base address)

3.3 Registers description

The following tables make a summary of the different fields of the A32 registers. By convention, active low signals are named <SignalName>B.

- Master Identifier: *_ID (+200, +240, +280, +2C0)

Returns a 16-bit value, indicating the version of the master block PROM. Up to now, the only available version is `0x1010`. This register is kept for historical reasons, and compatibility with the previous master boards. It is now recommended to use the ad-hoc registers of the CR/CSR configuration ROM (Vendor ID, Board Id, Revision ID).

- Master Status: *_STAT (+208, +248, +288, +2C8)

Bit(s)	Name	Mode	Default	Description
0	RunEnable	R only	0	See CONF register
1	EmitFullB	R only	1	The emitter FIFO is not full
2	EmitEmptyB	R only	0	The emitter FIFO is not empty
3	RecFullB	R only	1	The receiver FIFO is not full
4	RecEmptyB	R only	0	The receiver FIFO is not empty
5	ReadyToSend	R only	0	The emitter FIFO contains a full frame, ready to be sent
6	HasReceived	R only	0	The receiver FIFO has received a full frame
7	Interrupt	R only	0	Received an interrupt from a slave
8	SoftError	R only	0	Received an undecodable frame
9	ChkErr	R only	0	Received a frame with a bad checksum
10	Emitting	R only	0	The master is busy and emitting a frame
11	Receiving	R only	0	The master is busy and receiving a frame
12	SerialBusy	R only	0	The master is emitting or receiving a frame
13	SelectMS	R only	0	see CONF register
14	SelectSM	R only	0	
15	EmitForce	R only	0	see bit 12 of CONF register

Table 4: Master STAT register

The Interrupt, SoftError and ChkErr bits go high when an error is detected, and remain active until the board is reset or the master block receives a CLR VME command (see CONF register, ClrModule bit).

- Master Configuration: *_CONF (+210, +250, +290, +2D0)

Bit(s)	Name	Mode	Default	Description
0	RunEnable	R/W	0	Enables to send frames on the serial outputs
1	ClrAlarms	W only	0	1 = Clear the STAT Alarm bits (Interrupt, SoftError, ChkErr)
4	Spac2_3	R/W	0	0 = Generate and receive SPAC3 frames 1 = Generate and receive SPAC2 frames
5	CarDet	R only	0	1 = A carrier is detected on the active SM input
6	InvMS1	R/W	0	1 = send the MS1 (MS2) signal inverted
7	InvMS2			
8	MS1Carrier	R/W	0	0 = write 0 on MS1 (MS2) line in idle state 1 = write a 5 MHz carrier on the MS1 (MS2) line in idle state set to 0 if bit Spac2_3=1
9	MS2Carrier			
10	InvSM1	R/W	0	1 = invert the received SM1 (SM2) signal before frame detection
11	InvSM2			
12	EmitForce	R/W	0	Enables the emission of frames even when SerialBusy=1
13	SelectMS	R/W	0	0 = Emit the MS frame on the MS1 output, leave MS2 idle
14	SelectSM	R/W	0	0 = Watch at the incoming frame on SM1 input, ignore SM2 input
15	ClrModule	W only	0	1 = Clear the master block (STAT alarm bits, FIFOs, etc)

Table 5: Master Configuration register

This configuration register enables to set-up the protocol used by the serial emitter and the serial receiver of the master. By default, the master sends SPAC3 frames on one of the MS lines and stores SPAC3 frames from one of the SM lines. When the bit SPAC2_3 is set, the master handles the SPAC2 protocol.

The EmitForce bit should not be set, in a standard use of the master. This feature is only proposed for debugging purposes.

A frame is emitted on the line as soon as:
RunEnable=1 and ReadyToSend=1 and (SerialBusy=0 or EmitForce=1)

InvMS[2:1], and InvSM[2:1] are dedicated to correct pin swapping of some optical transceiver samples. By default, and when using the LVDS serial link, those bits should be kept to 0.

When using the optical link master block to drive a full front-end crate through a controller board, it is recommended to have two modes of operation:

- *disable the transmission of carrier on the MS lines during data taking and when the serial links are idle (bits MS1Carrier and MS2Carrier cleared).*
- *initially enable the transmission of carrier on both links before accessing slaves on the serial link (bits MS1Carrier and MS2Carrier set).*

- FIFO Status: *_ESTAT (+23C, +27C, +2BC, +2FC) and *_RSTAT (+224, +264, +2A4, +2E4)

Bit(s)	Name	Mode	Default	Description
[11:0]	Nbytes	R only	0	Number of bytes stored into the FIFO
12	Ready	R only	0	The FIFO contains a full frame ESTAT(Ready) = STAT(ReadyToSend) RSTAT(Ready) = STAT(HasReceived)
13	FullB	R only	1	The FIFO is not full
14	EmptyB	R only	0	The FIFO is not empty
15	Aempty	R only	0	The FIFO is almost empty, i.e. Nbytes=1

Table 6: FIFO Status Registers

*_ESTAT are the status registers of the emission FIFOs, *_RSTAT are those of the reception FIFOs.

- Emitter FIFO Content: *_EDAT (+238, +278, +2B8, +2F8)

Bit(s)	Name	Mode	Default	Description		
[7:0]	Data	R/W	0	The sequence of bytes to send as a frame		
				1st byte	7	Direction bit (1=Master to Slave)
					[6:0]	Slave address[6:0]
				2nd byte	7	Read/Write
					[6:0]	SubAddress[6:0]
				Next bytes	Data bytes (low order first)	
8	Eof	R/W	0	1= end of the frame		
12	Invalid	R only	0	The FIFO content is not available, because the master is busy		

Table 7: Emitter FIFO Content register

The emitter FIFO is 8-bit wide and can only contain one frame. When writing the data bytes of the frame into the FIFO, write the last word with its EOF bit to '1'. This has the effect to set the ReadyToSend flag (see STAT register), and even to immediately send the frame on the line, if all the conditions required to emit the frame are fulfilled (if the RunEnable of the CONF register is already set).

The emitter FIFO can also be reloaded with the last frame written, through the RELOAD VME command.

- Receiver FIFO Content: *_RDAT (+220, +260, +2A0, +2E0)

Bit(s)	Name	Mode	Default	Description		
[7:0]	Data	R/W	-	The sequence of bytes to send as a frame		
				1st byte	7	Direction bit (1=Master to Slave)
					[6:0]	Slave address[6:0]
				2nd byte	7	Read/Write
					[6:0]	SubAddress[6:0]
				next bytes	Data bytes (low order first)	
				Last byte	Checksum byte	
8	Eof	R/W	-	1= end of the frame, the Data byte is the (data) checksum byte		
9	ChkOK	R only	1	Data or Partial checksum calculation is correct		
10	Interrupt	R only	0	An «Interrupt» error was received		
11	SoftErr	R only	0	A «SoftErr» error was received		
12	Invalid	R only	0	The FIFO content is not available, because the master is busy		
13	FullB	R only	1	The FIFO is not full		
14	EmptyB	R only	0	The FIFO is not empty		
15	Aempty	R only	0	The FIFO is almost empty, i.e. the data byte is the last one received from the serial inputs		

Table 8: Receiver FIFO Content register

When several frames are stored into the receiver FIFO, the Eof bit is the frame delimiter. If the master is configured in SPAC3 (i.e. SPAC2_3=0 in SPAC3 Configuration register), the partial checksum byte is not written into the receiver FIFO. In case of error in the partial checksum, the data bytes written into the FIFO have the CHKOK bit to zero.

Accessing the emitter or the receiver FIFO contents when the master is busy is not allowed. When reading a FIFO while the master is busy, the returned data is not guaranteed, and the InValid bit is high.

3.4 Operation examples

The following tables make a summary of the different fields of the A32 registers. By convention, active low signals are named <SignalName>B.

We present, in this chapter, some simplified procedure examples to read and write SPAC resources through the master board.

- Writing a resource into a SPAC remote slave

To make the A Master send a write frame to the serial bus, the following procedure can be applied:

1. (optional) Board reset — `Write XX @ Base+004`
   or A Master Configuration (SPAC2/SPAC3, Clear ...) — `Write xy @ Base+210`
2. Write the slave address into the emitter FIFO — `Write 1 & SlaveAdd[6:0] @ Base+238`
   Direction bit at 1
3. Write the resource sub-address into the emitter FIFO — `Write 0 & SubAdd[6:0] @ Base+238`
   Read/Write at 0
4. Write the n data bytes into the emitter FIFO — `Write 0 & Data[7:0] @ Base+238`
   last word with Eof bit at 1 — `Write 1 & Data[7:0] @ Base+238`
5. (optional) write the A_CONF register with RunEnable=1 — `Write 01 @ Base+210`

Steps 2 to 4 can be combined into a unique VME block write if possible. If the RunEnable bit of the A_CONF was set before writing the emitter FIFO, step 5 is not required.

- Reading a resource from a SPAC remote slave

  To make the A Master send a read frame to the serial bus, the following procedure can be applied:

  1. (optional) Board reset — `Write XX @ Base+004`
     or A Master Configuration (SPAC2/SPAC3, Clear ...) — `Write xy @ Base+210`
  2. Write the slave address into the emitter FIFO — `Write 1 & SlaveAdd[6:0] @ Base+238`
     Direction bit at 1
  3. Write the resource sub-address into the emitter FIFO — `Write 1 & SubAdd[6:0] @ Base+238`
     Read/Write at 1
  4. Write the resource size into the emitter FIFO — `Write @ Base+238 :`
     if size < 256 — `Write 1 & Size[7:0]`
     if size > 255 — `Write 0 & Size[7:0]` `Write 1 & Size[15:8]`
  5. (optional) write the A_CONF register with RunEnable=1 — `Write 01 @ Base+210`

  Steps 2 to 4 can be combined into a unique VME block write if possible. If the RunEnable bit of the A_CONF was set before writing the emitter FIFO, step 5 is not required.

  To read back the slave reply frames, several procedures are possible:

  1. (optional) wait until the A master is not busy — `Read @ Base+208, bit 12=0 ?`
  2. Read the receiver FIFO content as long as — `Read @ Base+220`
     it is not Empty — `bit 14=0 ?`
     or until it is Almost Empty — `bit 15=1 ?`

  Another possibility:

  1. (optional) wait until the A master is not busy — `Read @ Base+208, bit 12=0 ?`
  2. Read the receiver FIFO status — `Read @ Base+224`
     get Nbytes, the number of bytes stored — `nbytes[11:0]`
  3. Read Nbytes times the receiver FIFO content — `Read Nbytes times @ Base+220`
     (block mode possible).

  The data block read back from the receiver FIFO are one or several frames, that then have to be decoded (removing the slave address, sub-address and checksum words).

4 Software

A set of routines, under the form of a C library (spac2lib) is provided to the user to drive the SPAC2 Master modules and access the slaves of a SPAC network [4]. A set of LabVIEW tools for lab tests can also be supplied «as is» under request (libraries for LabVIEW 6 or higher, or LabVIEW runtimes).

The C library consists in a set of functions, separated in several modules:

- `SPAC2.*`: low and high level routines to drive a SPAC Master module
- `SPAC2Utils.*`: some extra debugging routines to analyse the masters VME registers
- `SPAC2Messages.*`: (platform dependent) error and text messages interface module. `SPAC2Messages.c` can be adapted to your environment.
- `SPACSlave.*`: SPAC slave definitions.
- `SPACVME.*`: (platform dependent) low level VME access routines. `SPACVME.c` can be adapted to your VME environment.

- `SPACTypes.h`: miscellaneous integer type definitions.
- `FecController.*`: routines to drive the front-end crate controller via the Optical SPACMaster
- `SPACMaster6uTest.c`: example of a main program adapted to the SPAC Master VME64 6U

The following routines return 0 in case of error, unless indicated.

4.1 Initialisation routines

- `short` **`SPAC6uDeclareBoard`**`(SPAC6uBoard** Card, u_long WhichBaseAddress)`

  Allocates a `SPAC6uBoard` structure that stores the specifications of the 6U board to be used. After completion, the fields `A` and `B` of the allocated `SPAC6uBoard` structure are the `SPAC2MasterBlock` structures of the master blocks. `WhichBaseAddress` is the logical base address of the VME board (see `SPACMaster6uTest.c` for an example when your VME hardware setup requires mapping procedures).

- `short` **`SPACBoardReset`** `(SPAC2Board *Card)`

  Global Reset of the whole module (RESET VME function). `SPAC2Board` is a structure storing the specifications of a SPAC2/SPAC3 Master. However, the `SPACBoardReset` routine can also be applied to a `SPAC6uBoard` structure.

- `short` **`SPACReset`** `(SPAC2MasterBlock *Master)`

  Reset of one master block (A or B).

4.2 Master configuration routines

- `u_short` **`SPACIdentifier`** `(SPAC2MasterBlock *Master)`
- `u_short` **`SPACMasterStatus`** `(SPAC2MasterBlock *Master)`
- `u_short` **`SPACMasterConfig`** `(SPAC2MasterBlock *Master)`

  resp. read the ID, STAT and CONF VME register of the Master block `Master` and return it.

- `short` **`SPACSetConfig`** `(SPAC2MasterBlock *Master, u_short Conf)`

  write `Conf` into the CONF VME register of the Master block `Master` (see table 5). The `Conf` parameter can be built as a combination of the following constants defined in `SPAC2.h`:

Constant Name	**Value**
SPAC2_RUN_ENABLE_CONFIG_MASK	(0x0001)
SPAC2_CLEAR_ALARMS_MASK	(0x0002)
SPAC2_SPAC_VERSION_MASK	(0x0010)
SPAC2_INVMS1_MASK	(0x0040)
SPAC2_INVMS2_MASK	(0x0080)
SPAC2_INVSM1_MASK	(0x0400)
SPAC2_INVSM2_MASK	(0x0800)
SPAC2_MS1_CARRIER_MASK	(0x0100)
SPAC2_MS2_CARRIER_MASK	(0x0200)
SPAC2_EMITFORCE_CONFIG_MASK	(0x1000)
SPAC2_SELECTMS_CONFIG_MASK	(0x2000)
SPAC2_SELECTSM_CONFIG_MASK	(0x4000)
SPAC2_CLEAR_MODULE_MASK	(0x8000)

The bits of the configuration register (except EmitForce) can also be changed individually, by the following dedicated routines. The parameter `on_off` can be assigned the constant values `SPAC_ON` or `SPAC_OFF`:

- `short` **`SPACRun`** `(SPAC2MasterBlock *Master, char on_off)`
- `short` **`SPACClearAlarms`** `(SPAC2MasterBlock *Master)`
- `short` **`ChangeSpacVersion`** `(SPAC2MasterBlock *Master, int vers)`

  `vers` (1,2 or 3) defines the SPAC protocol version to be used. An error is returned if the requested version is not supported by the `SPAC2MasterBlock` master block.

- `short` **`ChangeInvMS1`** `(SPAC2MasterBlock *Master, char on_off)`

- short **ChangeInvMS2** (SPAC2MasterBlock *Master, char on_off)
- short **ChangeInvSM1** (SPAC2MasterBlock *Master, char on_off)
- short **ChangeInvSM2** (SPAC2MasterBlock *Master, char on_off)
- short **ChangeMS1Carrier** (SPAC2MasterBlock *Master, char on_off)
- short **ChangeMS2Carrier** (SPAC2MasterBlock *Master, char on_off)
- short **ChangeTransferModeMS** (SPAC2MasterBlock *Master, char ms2_ms1)
- short **ChangeTransferModeSM** (SPAC2MasterBlock *Master, char sm2_sm1)

  ms2_ms1 and sm2_sm1: 0 for link1, 1 for link2.

4.3 SPAC slaves access routines

- short **SPACBoardScan** (SPAC2Board * Card)

  is a utility routine that allows to detect all the SPAC slaves connected to the A and B blocks of the SPAC2Master module Card. It returns the number of detected slaves.

- SPACSlaveTab ***SPACBoardScanTab**((SPAC2MasterBlock *Master, int *nSlaves, int verbose))

  is an alternative of SPACBoardScan, that returns an array of detected slaves instead of printing them. The returned SPACSlaveTab element is an array of SPACSlave pointers (see SPACSlave.h).

The seven following routines write or read in the remote SPAC slave identified by Address[3] a resource of sub-address SubAddress, via the master block defined by SPAC2MasterBlock.

- short **SPACWriteRegister** (SPAC2MasterBlock* Master, u_char Address, u_char SubAddress, u_long Data, u_long Size)
- short **SPACReadRegister** (SPAC2MasterBlock *Master, u_char Address, u_char SubAddress, u_long *Data, u_long Size)

  Size is 1 to 4 and Data is the 8 to 32 bits register.

- u_short **SPACSlaveStatus** (SPAC2MasterBlock *Master, u_char Address)

  read the Status register of the Address SPAC slave and return it.

- short **SPACWriteFIFO** (SPAC2MasterBlock *Master, u_char Address, u_char SubAddress, u_char *Data, u_long Size)
- short **SPACReadFIFO** (SPAC2MasterBlock *Master, u_char Address, u_char SubAddress, u_char* Data, u_long Size)
- short **SPACWriteRAM** (SPAC2MasterBlock *Master, u_char Address, u_char SubAddress, u_char* Data, u_short Size, u_short RAMStartAddress)
- short **SPACReadRAM** (SPAC2MasterBlock *Master, u_char Address, u_char SubAddress, u_char* Data, u_short Size, u_short RAMStartAddress)

  In read mode, the Data buffer that receives the n=Size read bytes has to be allocated before the routine calls. The number Size of bytes to write or read is not limited. If it exceeds the master FIFOs depth, the operation is performed transparently in several packets.
SPACWriteRAM and SPACReadRAM write the slave NTA value to RAMStartAddress before the data block transfer.

The two following routines perform I^2C accesses on the remote SPAC slave identified by Address, via the master block defined by SPAC2MasterBlock. The target I^2C Slave is defined by I2C_Slave_Address. The operation is done through the link I2C_Link (0 or 1) at a rate defined by I2C_Clock_period (0 for 400 ns to 0xF for 6.4us, see I^2C feature of the SPAC ASICs [1]). Size is limited to 15, Data must be allocated before the routine calls.

3 Address lower than 0x10 is reserved to global and local broadcast write operations

- short **SPACWriteI2C** (SPAC2MasterBlock *Master, u_char Address, u_char* Data, u_short Size, u_char I2C_Slave_Address, u_char I2C_Clock_period, u_char I2C_Link)
- short **SPACReadI2C** (SPAC2MasterBlock *Master, u_char Address, u_char* Data, u_short Size, u_char I2C_Slave_Address, u_char I2C_Clock_period, u_char I2C_Link)

The two following routines are dedicated to access, through I^2C, the internal registers of the TTCrx connected to the SPAC slave. The TTC_ID_I2C is derived from the TTCrx individual address, TTCRegAdd is the address of the TTCrx register and TtcRegDat is the register value [5].

- short **SPACWriteTTC** (SPAC2MasterBlock *Master, u_char Address, u_char TtcRegAdd, u_char TtcRegDat, u_char TTC_ID_I2C, u_char I2C_Clock_period, u_char I2C_Link)
- short **SPACReadTTC** (SPAC2MasterBlock *Master, u_char Address, u_char TtcRegAdd, u_char *TtcRegDat, u_char TTC_ID_I2C, u_char I2C_Clock_period, u_char I2C_Link)

4.4 Useful routines defined in SPAC2Utils.c

The following routines are debugging routines that produce a comment about some read VME registers of the SPAC2Master module.

- char ***SPACStatusComment** (u_short Status)
  returns a comment string about the value Status of a master block STAT VME register.

Example:

```
u_short Status;

printf("Master Status = 0x%x",Status=SPACMasterStatus(curmaster));
printf("%s\n",SPACStatusComment(Status));
```

- char ***SPACConfigComment** (u_short Config)
  to comment the CONF VME register values.
- char ***SPACFifoStatComment** (u_short Status)
- char ***SPACRecFifoComment** (u_short Data)
  to comment the ESTAT, RSTAT, and RDAT VME register values.

4.5 Front-End Crate Controller routines

The controller hosts two SPAC slaves, configured in repeater mode, namely the SPAC#1 and the SPAC#2. The SPAC#1 handles the MS1/SM1 lines of the SPAC network and the SPAC#2 the MS2 and SM2 serial lines. Each SPAC ASIC receives its clock from a TTCrx (TTCrx#1 and TTCrx#2). The SPAC ASICs can access their TTCrx through their I2C link 0. The different routines of the FecController.c file are explained in the example program SPACMasterOptTest.c.

- short **FecContDeclareBoard** (FecContBoard** ContBoard, SPACSlavePtr *Slaves, u_short *TTCrxAdd)
  Allocates a FecContBoard structure that stores the specifications of the controller. Slaves is an input array of 2 SPACSlave structure pointers that contain, before the routine calls, the specifications of the SPAC slaves (slave address, local broadcast address and master block through which it is accessible). The input TTCrxAdd parameter is an array of size 2, containing the TTCrx IDs of the TTCrx#1 and the TTCrx#2. An example of call is given below.

Example:

```
short             ok;
SPAC2MasterBlock *CurMaster;
FecContBoard     *FecController;
SPACSlavePtr      FecContSlaves[FECCONTNUMSLAVES];
u_short           FecTTCrxAdd[FECCONTNUMSLAVES];

/* Declare CurMaster before SPACDeclareSlave calls */

ok = SPACDeclareSlave( &(FecContSlaves[0]),0,0x1F,0xF,CurMaster, NULL);
ok = SPACDeclareSlave( &(FecContSlaves[1]),0,0x2F,0xF,CurMaster, NULL);
FecTTCrxAdd[0] = 0x1234;
FecTTCrxAdd[1] = 0x5678;
ok = FecContDeclareBoard (&FecController,FecContSlaves,FecTTCrxAdd);
```

- `short` **`FecContCheck`** `(FecContBoard* ContBoard)`

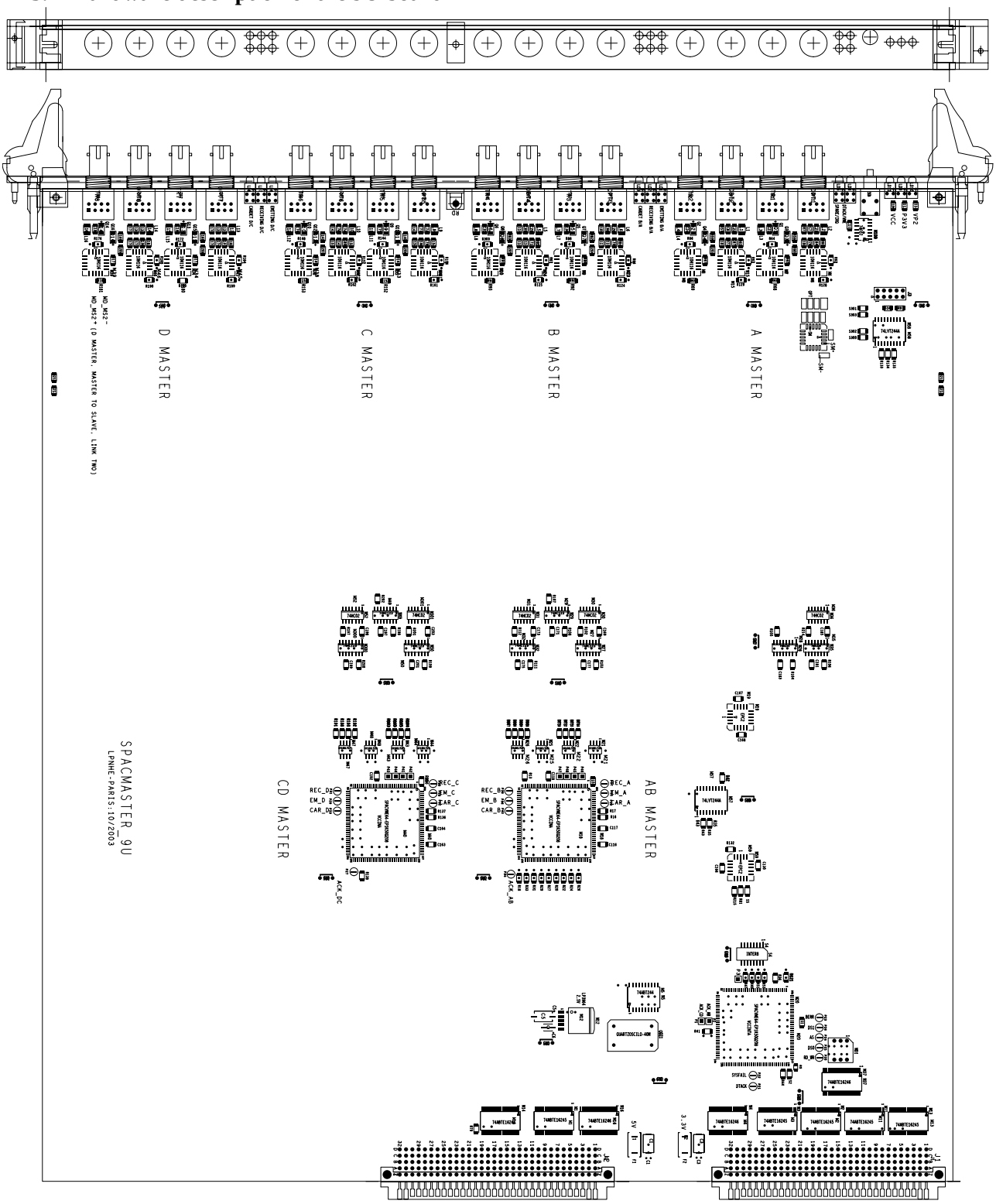
Check that both SPAC slaves of the controller are accessibles, and that all the power levels of the board are OK.

- `short` **`FecContSetQuiet`** `(FecContBoard* ContBoard, int Verbose)`
- `short` **`FecContSetDownload`** `(FecContBoard* ContBoard, int Verbose)`

Configure the SPACMaster and the controller such that the SPAC bus is quiet ("Physics mode") or to be able to access the SPAC slaves of the SPAC bus ("Download mode"). See `SPACMasterOptTest.c` for more detailed explanations.

5 Appendix A: 9U board addendum

5.1 Hardware description of the 9U board



- Power consumption

Power Supply	Consumption	Fuse	Fuse Value
5 V	<3 A	F1	4 A
3.3 V	250 mA	F2	500 mA

- Serial Number

The Board ID byte (or Serial Number) of the board is coded by the 8 switches S4 close to the VME device. It can be read back in the V_BID VME register or the Serial Number defined in the VME64x CR space (which address BEG_SN is defined in the CR space at 0xCB, 0xCF and 0xD3).

- FPGA programming chains

The programmable devices are accessible through the JTAG programming port J3 (upper left edge of the board). The JTAG chain is the following:

Device	Device Type	Description
1	EPC2LC20	PROM of the AB and CD Master devices
2	EP1K50	AB Master device
3	EP1K50	CD Master device
4	EP1K50	VME device
5	EPC2LC20	PROM of the VME device

At startup, both EPROMs configure the devices they are connected to in the passive serial scheme.

The master EPROM (device #1 in the previous table) contains the codes of two EP1K50 master devices (AB master and then CD master). The VME EPROM (device #5 in the previous table) contains the code for the EP1K50 VME device.

6 References

[1] Documentation on the «SPAC2» and «SPAC3» protocols can be found at
http://lpnhe-atlas.in2p3.fr/pubdocs

[2] SPAC2 / SPAC3 Master Boards User Guide, O. Le Dortz et al., September 2002
http://lpnhe-atlas.in2p3.fr/pubdocs/serial/master

[3] ATLAS Readout Driver VMEbus implementation, Recommendations of the Detector Interface Group: http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/DIG/ROD/

[4] Latest version of this note, PROM upgrades and software:
http://lpnhe-atlas.in2p3.fr/pubdocs/serial/master

[5] TTCrx Reference Manual, a Timing Trigger and Control Receiver ASIC for LHC Detectors, RD12 project collaboration, CERN EP-MIC:
http://ttc.web.cern.ch/TTC/TTCmain.html#TTCrx