LHCb 2005-xxx
DAQ
February 24, 2005

SPECS :

a Serial Protocol for the Experiment Control System of LHCb

Version 3.5

Dominique Breton, Daniel Charlet

Laboratoire de l'Accélérateur Linéaire - Orsay

ABSTRACT

This document describes the SPECS protocol and its implementation. The SPECS is a 10Mbit/s serial bus, designed to be simple, cheap and reliable. It is mainly used to download and read back the configuration of the electronics located on the detector. It provides I2C and JTAG interfaces for the users, and is delivered with a software patch.

Contact : charlet@lal.in2p3.fr, breton@lal.in2p3.fr

1 INTRODUCTION 5

2 REQUIREMENTS AND SOLUTIONS FOR LHCB 5

2.1 General presentation 5

2.2 Radiation environment 6

2.3 Data rate 6

2.4 Safety of the information 6

2.5 Physical link 7

3 THE SPECS PROTOCOL 8

3.1 General description 8

3.2 SPECS Frames 9
3.2.1 SPECS Data Frame 9
3.2.2 SPECS Interrupt Frame 11

3.3 Communications 11
3.3.1 Parallel mode Write access 11
3.3.2 Parallel mode read access 12
3.3.3 SPECS serial mode access 13
3.3.4 Interrupt 13

4 SPECS MASTER 14

4.1 SPECS multi-master board 14

4.2 SPECS MASTER DESCRIPTION 15
4.2.1 Firmware description 15
4.2.2 PCI frames 16
4.2.3 Master registers 16

5 SPECS SLAVE 19

5.1 SPECS slave chip 19

5.2 SPECS slave mezzanine general description 21
5.2.1 List of the features of the mezzanine board: 23

5.3 Mezzanine physical implementation 25

5.4 SPECS slave functional description 25
5.4.1 I2C bus 25
5.4.2 JTAG bus 26
5.4.3 Parallel bus 26
5.4.4 Configurable I/O (Static register) 26
5.4.5 DCU Adc 26

5.4.6 Channel B decoding 27
5.4.7 Serial PROM (under development) 27

5.5 Slave registers 27
5.5.1 Test Register 28
5.5.2 Output pins Control Register 28
5.5.3 Output pins Configuration Register 28
5.5.4 Control Register 28
5.5.5 Status Register --- *Under development* --- 29
5.5.6 Serial Bus Selection Register 29
5.5.7 Interrupt Register --- *Under development* --- 29
5.5.8 Date Register 30
5.5.9 User defined Interrupt Register --- *Under development* --- 30
5.5.10 Channel B Register 30

6 SPECS USE CASES 31

6.1 Calorimeter implementation 31

6.2 Multi Mezzanine implementation 32

6.3 How to make long distance I2C to local I2C 34

6.4 Technical Specifications 34
6.4.1 How we make JTAG and I2C from SPECS 34
6.4.2 SPECS connector 37
6.4.3 JTAG connector implementation 38
6.4.4 I2C connector implementation 39
6.4.5 RJ45 connector 40
6.4.6 Mezzanine pinout 40

7 SOFTWARE 43

8 CONCLUSION 43

9 ANNEXE 44

9.1 I2C and JTAG driver cabling scheme 44

9.2 Long distance chaining cabling 45

9.3 Physical Dimensions 46

10 REFERENCES 47

1 Introduction

This document describes the SPECS protocol, a 10 Mbit/s serial link designed for the configuration of remote electronics elements. SPECS is a single master multi-slave bus, designed to allow a simple, fast and cheap means of communication between electronics systems (*see* Figure 1). It is also designed to be efficiently protected against errors, and to be flexible.

Firstly, the requirements and the architecture of the SPECS system are presented. Then the protocol is described in detail. Section 4 describes the SPECS Master and the following section describes in detail the SPECS Slave, its various functions and registers. Finally, several use cases are presented. The last three sections are the most relevant for the user, since they explain how to communicate with the available SPECS interfaces.

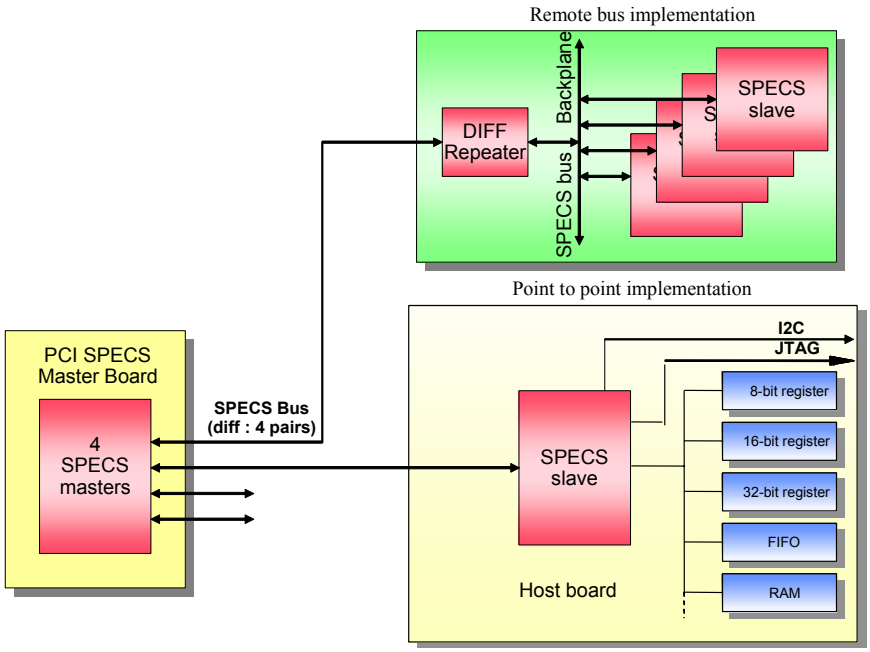


Figure 1: Possible implementations of the SPECS Protocol

2 Requirements and solutions for LHCb

2.1 General presentation

The LHCb sub-detector electronics is organized around the detector in crates or on individual boards, which require a configuration access to load or read different types of information. These operations will be driven from a computer located in the control room, roughly 60 meters distant from the various front-end crates. Therefore, the experiment requires a configuration bus, able to communicate properly on a 100 meter line, with a unique master, and up to 32 slaves.

In such a configuration, the master card would be located in the control room, which is not exposed to radiation, and the slave would be implemented close to the detector electronics boards.

In order to minimize the cost, the volume of cables, and the power consumption of the bus, it is appropriate to use a serial bus, composed of only a few lines, requiring only a few low value resistors for its adaptation, and implying very little remote electronics. Moreover, the fact of having two signals in each direction simplifies the receiver which does not have to extract the clock from a unique mixed line and to encode it back. In addition, the fact of avoiding I2C-like acknowledges ensures much higher data rates, especially at long distance.

2.2 Radiation environment

The radiation environment depends on the specific sub-detector. The total radiation dose is for example low for the calorimeters or the outer-tracker, and much higher for the vertex detector (Velo). For such a variable environment, it seems reasonable to develop a radiation tolerant circuit, to place the SPECS interface chips far enough from the high radiation dose zones, and to communicate with the rad-hard electronics via an intermediate bus (I2C or JTAG) that would be delivered by the SPECS slave.

However, even for the lower radiation zones, hardness to the single event upsets (SEU), single event transients (SET) and single event latch-ups (SEL) are required. This point has to be considered carefully, and the design of the slave chip should ensure that the registers and state machines are protected by appropriate redundancy. The single event latch-ups however are sensitive to the technology used. Therefore tests have been performed in order to choose the most adequate components (see [1], [2]).

2.3 Data rate

The volume of data to be transferred depends on the design of the various sub detector electronics. Let us consider a conservative example.

A frequently used RAM contains 50 to 500 kB of data. If a board houses 5 RAMs to download, we reach roughly 3 MB of data. If one crate, handled by one configuration bus, contains 16 such boards, the whole crate configuration then requires the transfer of 50 MB.

A reasonable delay to initialize a full detector, or read it back, supposing that all the configuration busses can operate simultaneously, is of the order of one minute. Therefore, the effective, or mean data rate has to be close to 1 MB/s, which is actually the bandwidth chosen for the SPECS.

This estimate is rather conservative. It probably over-estimates the amount of data to be transferred. In addition it does not take into account the gain of time expected by broadcast accesses, as many of the boards to configure will be identical.

2.4 Safety of the information

On long cables, at a 10 Mbit/s rate, in a noisy environment, errors may occur during the transfer of data. Protection against transfer errors is necessary to ensure proper behaviour.

This protection can consist of simply detecting errors, or of detecting and correcting them. In the LHCb environment, the frequency of errors expected should be very low. If the configuration bus enables the slaves to send interrupts to the master to mention an error after having received a message, the master can repeat the last transfer to correct the problem. In this case, the option of error correction implemented in the slave is not needed. Moreover, it would cost more redundancy bits in the transferred frames, and therefore would reduce the effective data transfer rate.

The error detection can itself be divided into 2 categories. The error can affect the address or other parameters that qualify the data (SPECS header). In this case, the message can be interpreted by the wrong slave, sent to the wrong memory with the wrong mode. The problem is that there is no way to know for sure after the transfer what happened, and which memory was affected. It thus may be necessary to reload the whole detector to ensure that the error is corrected.

If the error only affects data, the targeted memory is known. It can be reloaded straight away.

For these reasons, the SPECS bus has no error correction, but 2 error detection systems:

- Four bits of redundancy follow the header of the frame, which contains the address of the slave and other parameters. If an error is detected, the slave ignores the whole message, and sends an error interrupt to the master straight away.
- One byte of redundancy is sent at the end of the frame, and allows detection of a data error. If an error occurs, data is loaded, but the concerned slave mentions it to the master, sending an error interrupt straight away.

The error detection system is not yet implemented.

2.5 Physical link

The physical link has to be simple, easy to handle, and robust. An optical link is not mandatory for a 10 Mbit/s rate if the distance is not too large.[i]

The protocol requires 4 or 8 copper links (4 lines or 4 pairs) according to the chosen technology. The proposed default technology is the BLVDS (or LVDM), but PECL, BTL or GTL/GTL+ could also be used.

The BLVDS has excellent qualities for such an implementation. Sent on twisted pair cables, it can propagate fast signals (more than 150 MHz) along relatively long distances thanks to its 10 mA output current. Moreover, it can support a large number of slaves on the same line. PECL works well too. It has been successfully tested at 100 meters with pole-zero cancellation. These tests will be pursued to select the most adequate technology depending on the distance.

The 8-pin RJ45 connector was chosen for the interconnections. It is a cheap and compact standard, which can be associated with a cheap family of relatively good cables (cat 5 Ethernet type). These associated cables are twisted pairs surrounded by a shield. The shield may be connected to the system ground in only one point, or preferentially to all the interconnected electronics systems.

This solution complies with the requirements described in the note [3].

[i] Tests performed in the lab have shown that on a distance of 100 m the shape of the signal is satisfactory.

3 The SPECS Protocol

3.1 General description

The SPECS is a single master, multi slave serial bus. It communicates between a unique master and up to 32 slaves (limit fixed by the address range), at a 10 MHz rate, thanks to 4 different unidirectional lines. These lines called MS_SDA, MS__SCL, SM_SDA, SM_SCL are the data and clock lines, from master to slave, and from slave to master respectively. They can be implemented in differential mode (8 wires) or in unipolar mode (4 wires).

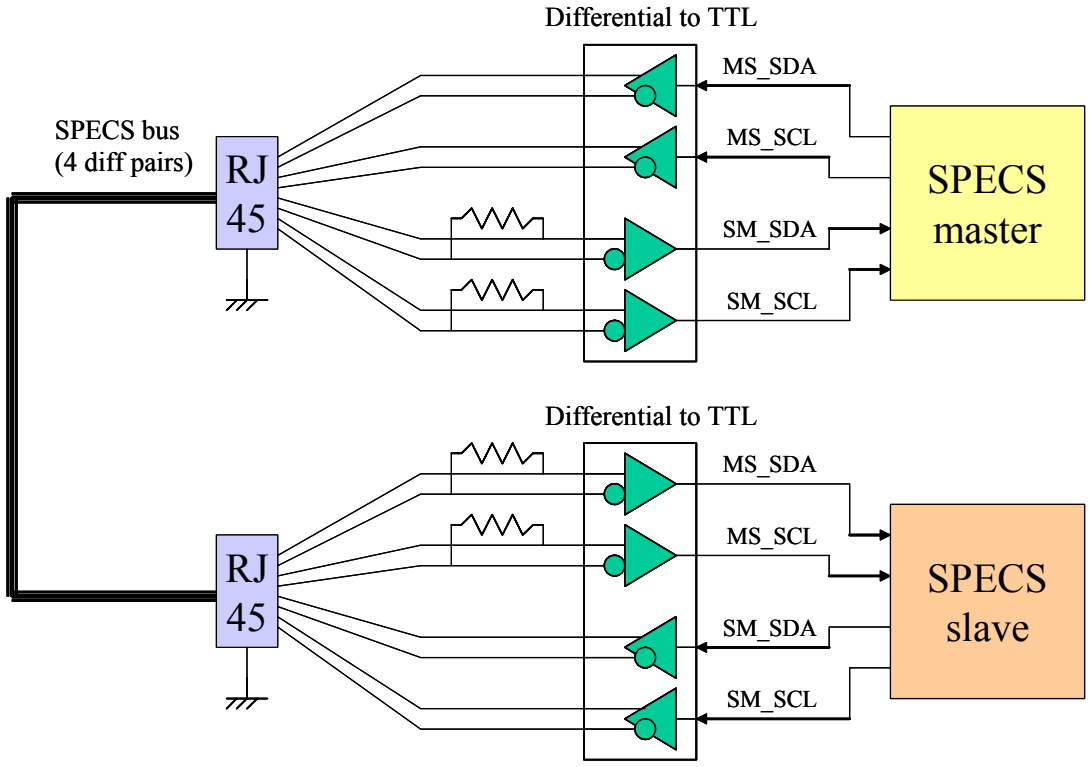


Figure 2 : Point to point Connection

The SPECS can be implemented in two different ways :

- Point to point connection, shown in Figure 2
- Remote bus implementation shown in Figure 3

The first implementation allows the user to connect a master to a single slave. It is for instance very useful for a test bench, or if every slave requires the full data bandwidth of the bus. If the number of slaves to access is important and the mean bandwidth small, it might be more appropriate to use the remote bus implementation. The latter offers the possibility to drive about 32 slaves at a long distance with a single master. It implies however the use of a remote differential repeater to ensure the signal integrity. This solution was chosen for the calorimeter electronics of LHCb, through distribution of the bus on the crate backplanes.

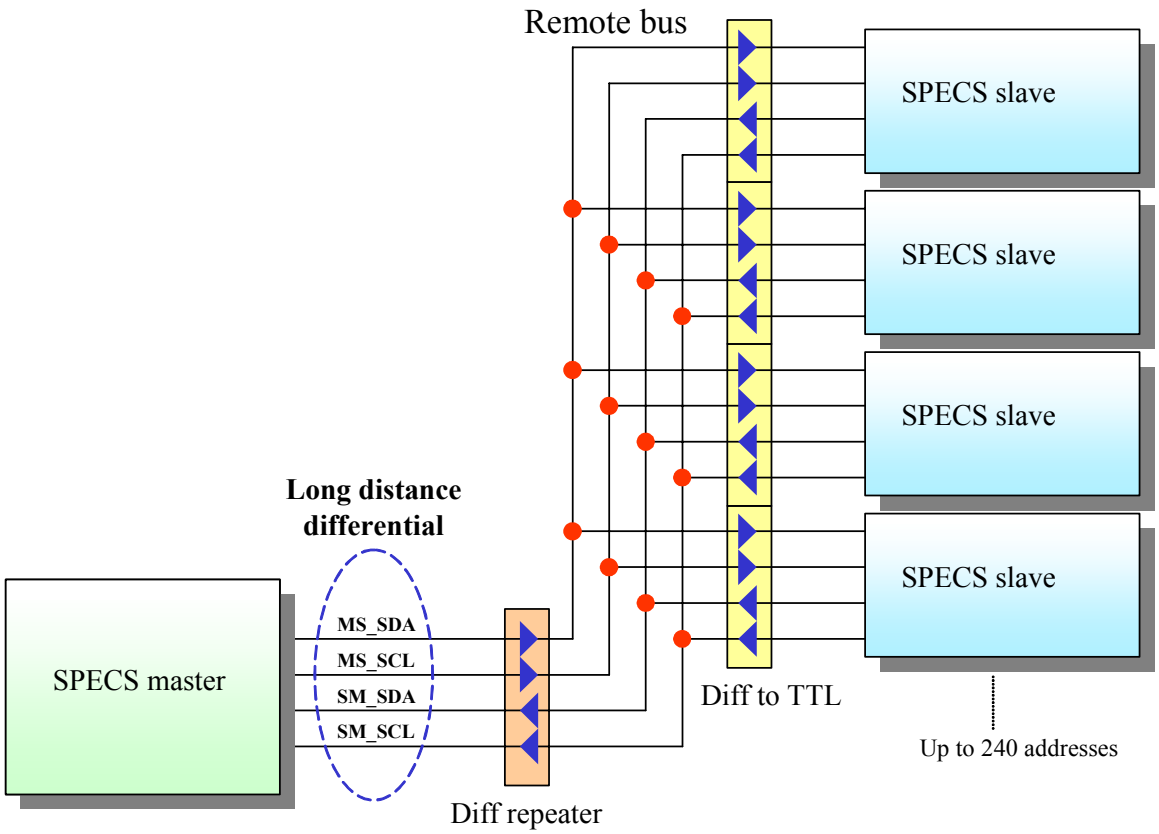


Figure 3 : Remote Bus Implementation

3.2 SPECS Frames

Data transfer is organized in frames of variable length. The format is the same for transfers from the master to the slaves or in the other direction, except for the interrupt commands which have a specific frame. The frames start and stop with a specific transition of the data line when the clock line is at a high level ("1"). These transitions are called 'start condition' and 'stop condition' (Figure 4), and are identical to the ones used in I2C.

The clock lines are active only during a data transfer and remain quiet during an idle phase. A frame is composed of a variable number of words, the latter being separated from each other by a 'missing clock cycle'. This makes the debugging by oscilloscope very easy, since the packets are clearly identified. The words have a fixed length of 9 bits and, due to the missing clock cycle, a word needs 10 cycles to be transferred. Therefore, with the 10 MHz clock frequency, the data transfer rate is 1 MB/s.

3.2.1 SPECS Data Frame

The standard data transfer frame is shown in Figure 4

Each word has a specific role in the protocol. The order of the bits transferred is always least to most significant bit (2^0 to 2^8). The 9^{th} bit (bit 2^8) tags the last data word of the frame, and is set to zero otherwise. The 8 remaining bits are used to transfer different types of data.

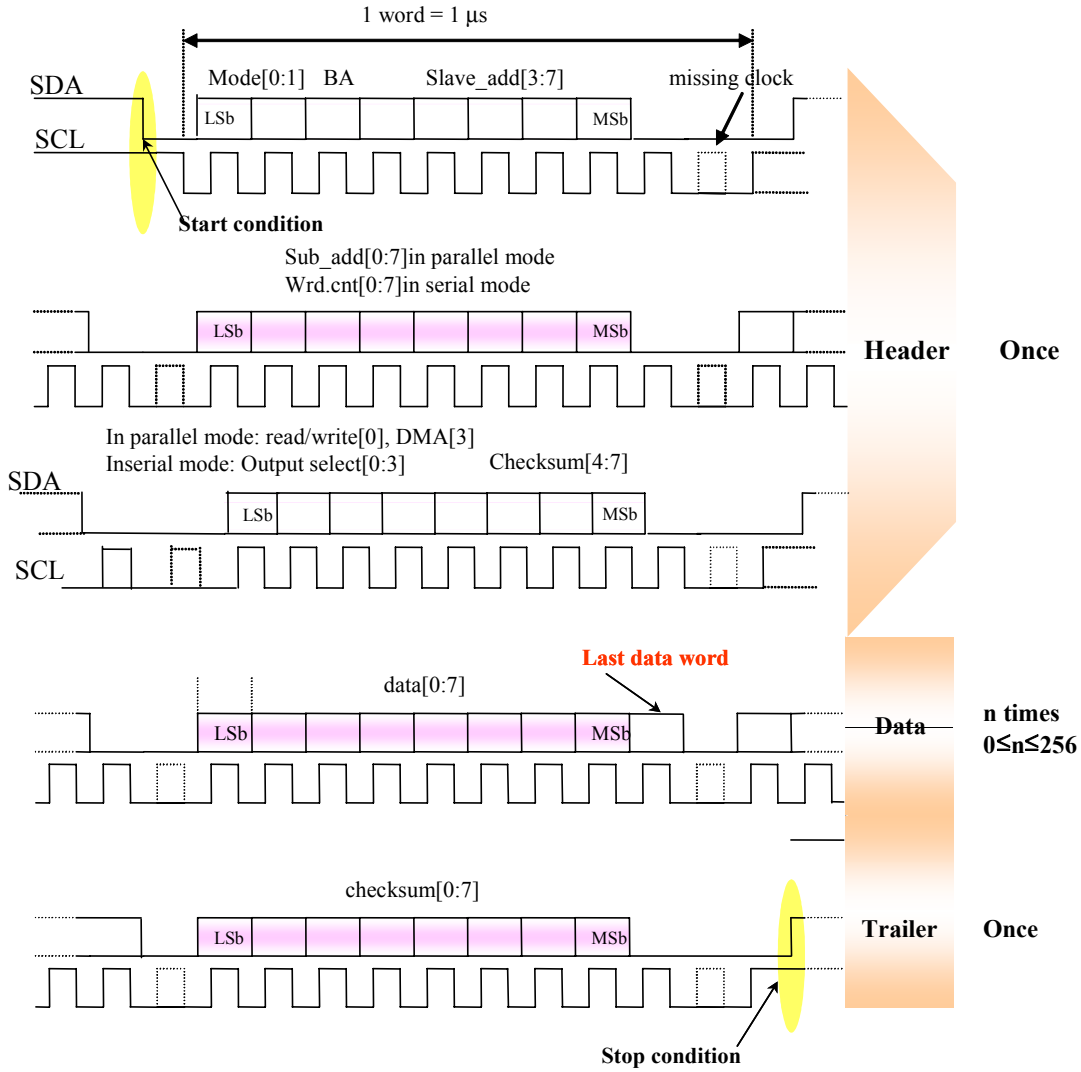


Figure 4 : The Standard SPECS Data Frame

The frame is composed of a fixed size header, a variable size data block, and a fixed size trailer.

- The header consists of 3 8-bit words:
    1. The first word contains 2 bits for the mode (I2C, JTAG, parallel bus, register), one bit for the broadcast address if appropriate, and 5 bits for the slave address;
    2. The second word gives the slave sub-address in parallel mode transfers and the word- count (number of words to transfer) in serial mode, i.e. I2C and JTAG ;
    3. The third word has a different contents depending on the mode:
        - in parallel mode it contains a read/write bit[0], a DMA bit [3] and the 4-bit header checksum. Bits [1] and [2] are not used;
        - in serial mode the first four bits give the output select (internal/external I2C and JTAG addresses) and the 4 most significant bits, the header checksum;

More details on the meaning of these bits can be found in Section 4.2.2. The header checksum is computed as the 'xor' of the bits of the header, taken 4 by 4.

- The data block contains a series of 0 to 256 words, according to the transfer to be performed. A more precise description follows in the next section.

- Finally, the trailer contains the checksum of the data block. The calculation of the checksum is :

$$trailer\_checksum[7:0] = \sum_{i=0}^{number\_of\_data\_words-1} data[i][7:0] \quad \text{where the sum operator is 'Xor'}$$

3.2.2 SPECS Interrupt Frame

The interrupt frame has to be as short as possible in order to reduce the probability of conflict, while providing the minimum relevant information to the master. The interrupt frame consists of a single word frame including only the address of the sender. (see also 3.3.4

3.3 Communications

The master always takes control of the bus to perform the reauired operations**.** The two possible operations for the master are the write and read accesses.

A slave normally does not take control of the bus without being requested to do so by the master. After a write command, it executes the operation, but a priori does not reply anything. No answer means that the transfer was successful. After a read command, the slave takes control of the bus to provide the requested data.

In case of a suspicious transfer, the slave sends an interrupt message to the master, both in read and write mode. The interrupt is sent after a known delay following the master's command.

The slave can also take control of the bus without any occurrence of a command from the master. This is the case when a 'user interrupt' is requested from the slave's host board. Then, totally asynchronously, the slave takes control of the bus to send the user interrupt to the master.

Finally, to prevent any device on the bus from remaining in an abnormal state after a wrong transfer, the master has a timeout counter. Since no transfer can be longer than 3+256+1 words (header size + maximum data size + trailer size), which corresponds to 260 µs, any continuous activity lasting more than about 300 µs will reset the state machines of all the devices

3.3.1 Parallel mode Write access

The frame corresponding to a parallel mode write access is shown in Figure 5. The frames are all transferred from the master to one slave, or more if the address corresponds to a broadcast address. The amount of data varies from 1 to 256 bytes. The latter value was chosen in order to reasonably limit the length of the block mode for transfer reliability reasons.

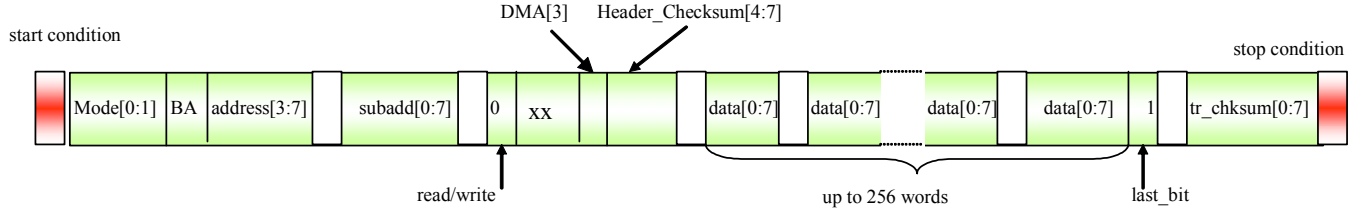


Figure 5: n-byte Write Command (Master)

This transfer does not require an answer from the slave, except in the case of a header or trailer checksum error.

3.3.2 Parallel mode read access

A special register exists in the slave for read operations. The master sends a frame to specify the number of data to read (Figure 6). This number is stored in an 8-bit internal register of the slave called word count. The word count register is a special feature of the slave, it has no address and is not readable. It can only be written by the master, and this is done when the read/write bit of the master frame is set to '1'. As we want to fully cover the 1 to 256 word range, and as the value we can encode over 8 bits are "0" to "255", during the read operation of the slave, the word count is decremented until it reaches '-1 = FFh'. Therefore, the number of words that will be read is "wordcount[7:0] + 1" covering the range [1;256], and the value "0" will involve the readout of one single byte.

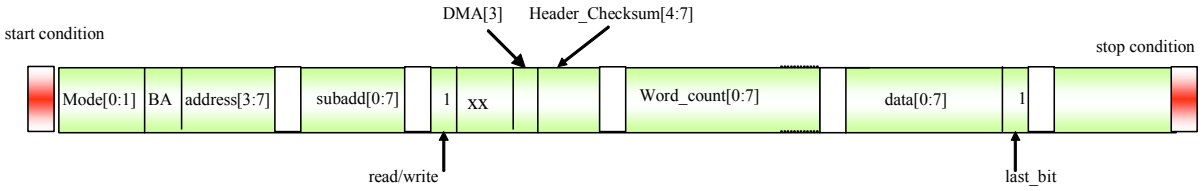


Figure 6 : Standard Read Command (Master)

After a read request, the slave sends a frame with the same header as the one sent by the master, followed by the requested data (Figure 7). Thus the master knows which slave has sent the frame, and what sub-address was read, so it can crosscheck both.

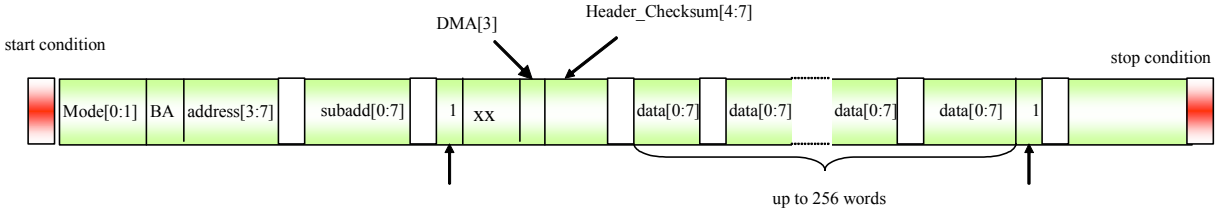


Figure 7 : Standard answer after a Read Command (Slave)

3.3.3 SPECS serial mode access

Roughly speaking, in I2C and JTAG each bit (and its clock) are generated by a SPECS byte, as described in section 6.4.1. I2C and JTAG both need a clock in write and read mode. Therefore the SPECS master needs to generate a frame even during read cycles. This is illustrated in Figure 8 For the case of an I2C transfer.

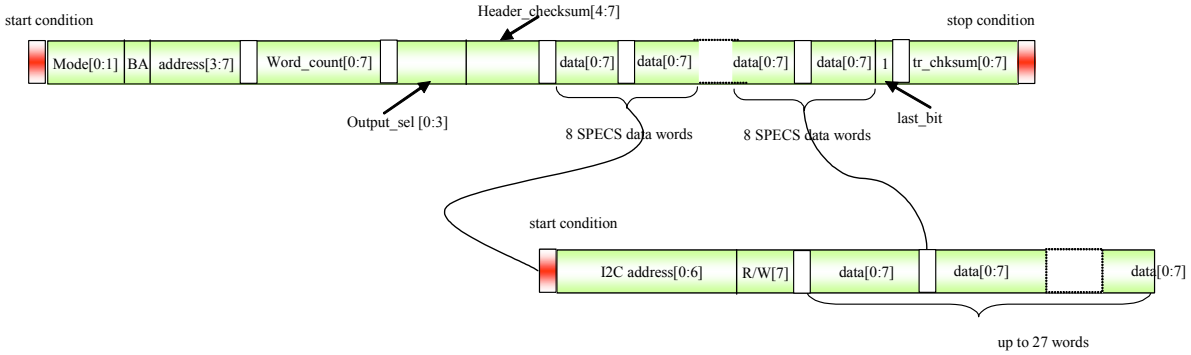


Figure 8 : Serial Mode Access (I2C and JTAG)

3.3.4 Interrupt

The frame corresponding to an interrupt is shown in Figure 9. It is simple and allows the master to target back directly the concerned slave.

After having received an interrupt order, the master will always read the status register of the sender to learn about its origin. If the address does not correspond to any known slave, or after a broadcast access, all slaves will be scanned. Indeed, we cannot ensure that two slaves will never start sending an interrupt at the same time, but the master will then at least be able to recognize the start condition and know that something happened on the other side of the line.

The interrupt order is also available for the user, to request specific attention of the master. Therefore an internal register memorizes the type of interrupt that occurred. Thus, when the master scans the slaves consequently to an interrupt order, it will be able to distinguish between the different types of interrupts, and to react thereto properly.

The interrupt mechanism has not yet been implemented.

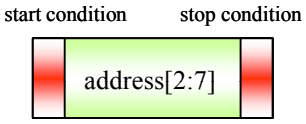


Figure 9 : Interrupt Frame

4 SPECS Master

The technical implementations described below are under development. For further information, please contact Daniel Charlet (charlet@lal.in2p3.fr).

4.1 SPECS multi-master board

The final SPECS master board will host 4 SPECS masters. The present version is based on only one master, as shown in Figure 10.

Of course, these masters cannot be connected together, the purpose of the board being to deliver 4 independent busses. The SPECS master is implemented on a standard 32-bit 33 MHz PCI board, which can be plugged into a PC. A photograph of the board is shown on Figure 11

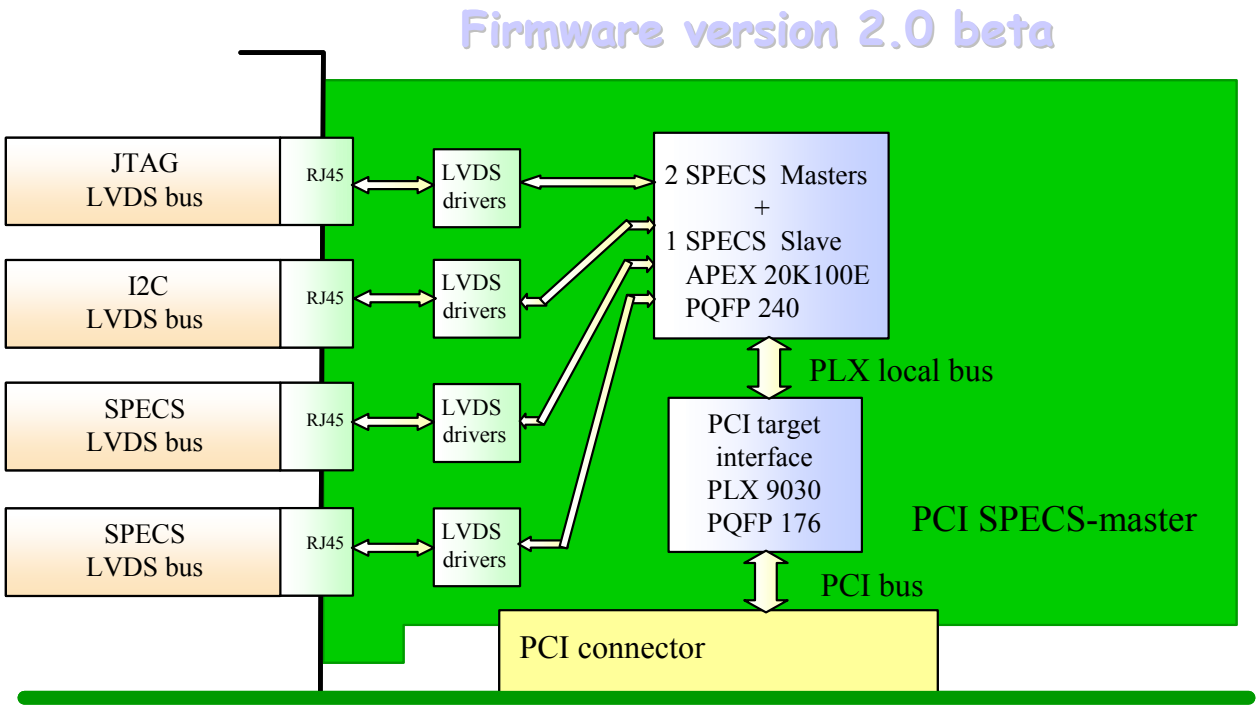


Figure 10 : Prototype of the SPECS Master Board

In order to offer an easy setup for the test benches, the board also includes a SPECS slave, which allows the user to benefit from both JTAG and I2C output capability.

As can be seen in Figure 10, the heart of the system is integrated within the Altera APEX FPGA

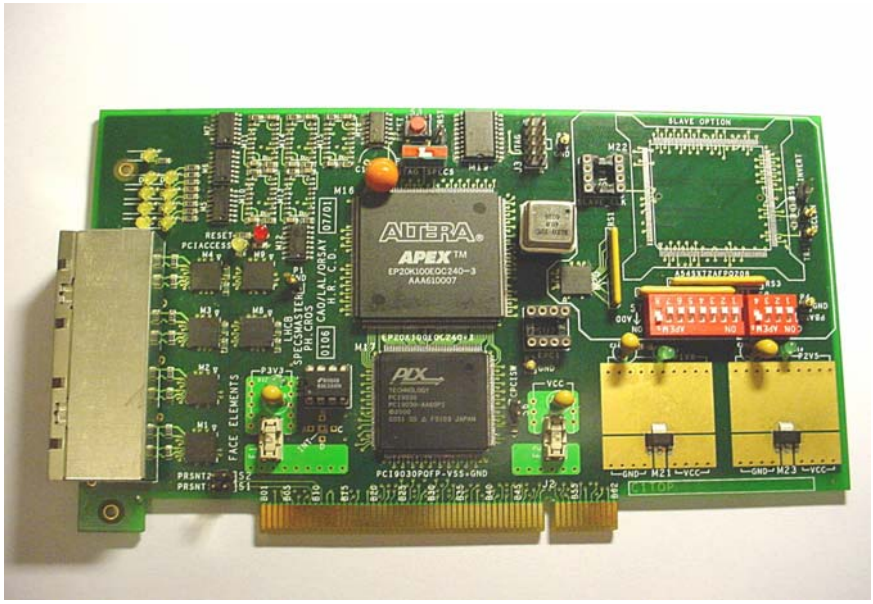


Figure 11 : Photograph of a SPECS Master Prototype board

4.2 SPECS MASTER DESCRIPTION

4.2.1 Firmware description

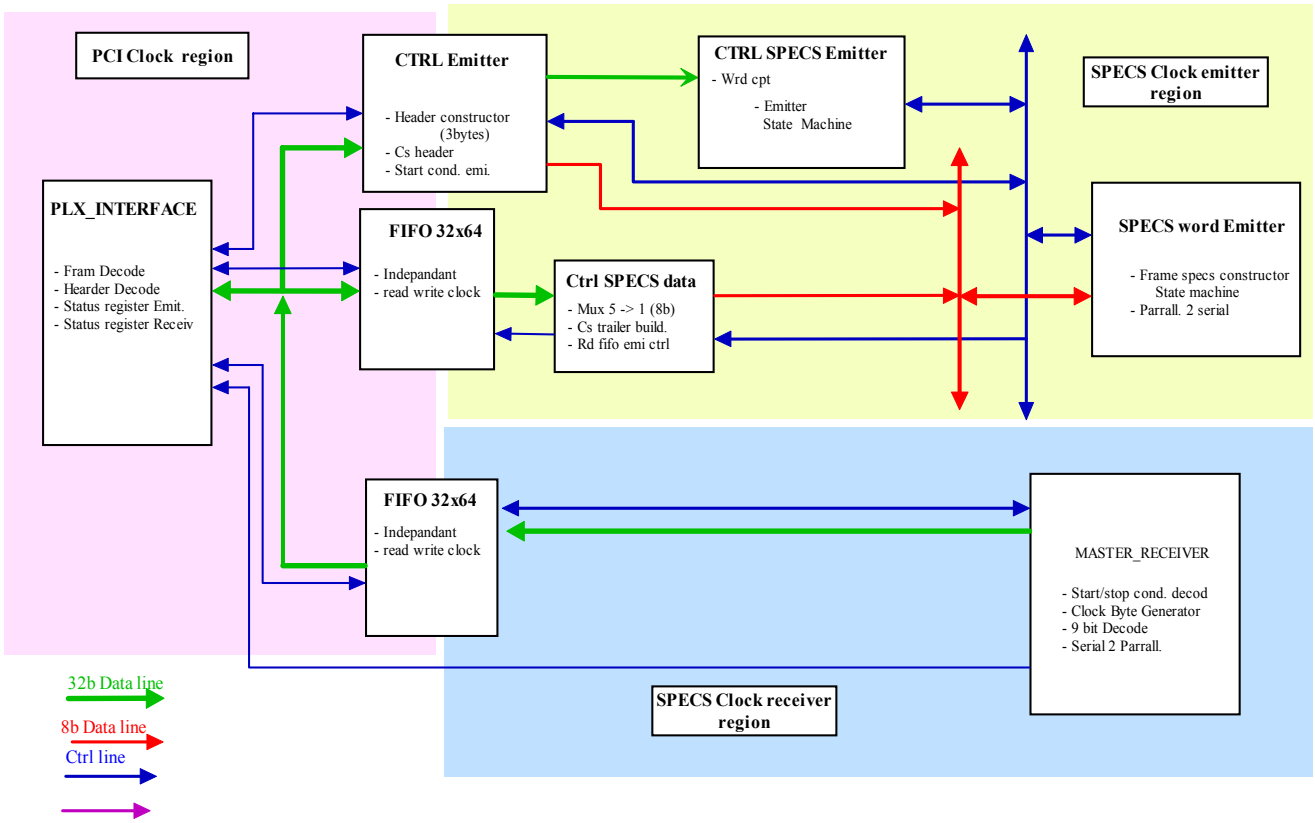


Figure 12 : Block diagram. of the FPGA The corresponding firmware is written in Verilog.

4.2.2 PCI frames

The tables below show the PCI SPECS frames. The write frame is computed by the software library and the read frame is decoded by the software library depending on the requested access mode.

Table 1 – PCI write frame

	2^{31} 2^{28}	2^{27} 2^{24}	2^{23} 2^{20}	2^{19} 2^{16}	2^{15} 2^{12}	2^{11} 2^{8}	2^{7} 2^{4}	2^{3} 2^{0}
Frame separator	7	6	5	4	3	2	1	0
Header Parallel mode	SPECS data word count		Slave address[5:0] Mode[1:0]		Sub-address (reg_mode)		xxxx	DMA/ xx R/W
Header Jtag or I2C mode	SPECS data word count		Slave address[5:0] Mode[1:0]		JTAG/I2C write word count		xxxx	Output Select
Data	Specs data 0		Specs data 1		Specs data 2		Specs data 3	
Data // bus access read	Read wordcount		x		x		x	

Table 2 – PCI read frame

	2^{31} 2^{28}	2^{27} 2^{24}	2^{23} 2^{20}	2^{19} 2^{16}	2^{15} 2^{12}	2^{11} 2^{8}	2^{7} 2^{4}	2^{3} 2^{0}
Frame separator	7	6	5	4	3	2	1	0
Header Parallel mode	SPECS data word count		Slave address[5:0] Mode[1:0]		Sub-address (reg_mode)		Specs data 0	
Header Jtag or I2C mode	SPECS data word count		Slave address[5:0] Mode[1:0]		Read word count		Specs data 0	
Data	Specs data 1		Specs data 2		Specs data n		Specs data n+1	
Last Data	Checksum[7:0]							

4.2.3 Master registers

Some of these registers are not yet implemented (and are identified as "under development") or are currently implemented only for one master.

The following registers are available for each specs master. They are listed in the following table and more details are given in the next sections.

Table 3 – SPECS Master Registers

Register	**Name**	**Address**	**Access**
Status Register	STATUS_REG	'h800 (to 'h803)	READ
Control Register	CONTROL_REG	'h800 (to 'h803)	WRITE
Slave interrupt Register	SPECS_SLAVE_INT		READ

Mask slave interrupt Register	MASK_SPECS_SLAVE_INT		READ/WRITE
PLX Interrupt Register	PLX_INTERRUPT		READ/WRITE
Mask PLX Interrupt Register	MSK_PLX_INTERRUPT		READ/WRITE
Test and date Register	TEST_REG	‘h808	READ/WRITE

4.2.3.1 Status Register (STATUS_REG):

This 5 bit register gives the status of each master.

Table 4 – SPECS_Bus_stat0 bit allocations (STATUS_REG of Master 0)

Bit	Name	Function
7	Checksum_error	If 1, error on the checksum
6	Rdempty_emi	If 1, read emitter FIFO empty
5	Wrfull_emi	If 1, write emitter FIFO full
4	Ena_wr	If 1, write is allowed
3		Always 0
2	Rdempty_rec	If 1, read receiver FIFO empty
1	Wrfull_rec	If 1, write receiver FIFO full
0	Ena_rd	If 1, read is allowed

4.2.3.2 Control Register (CONTROL_REG):

This 6 bit register controls each master.

Table 5 - SPECS_Bus_ctrl0 bit allocations (CONTROL_REG of Master 0)

Bit	Name	Function
4	Rst_rdwr	Set to 1 to reset the enable write bit of the Status Register. (*Under development*)
3	Rst_prog	Set to 1 to perform a software reset of the master. (*Under development*)
[2:0]	Clock_division_factor	Set to 1, 2, 4, 8 or 16 to obtain a clock division factor of 1, 2, 4, 8 or 16 and a I2C or JTAG clock of frequency 1024 kHz, 512 kHz, 256 kHz, 128 kHz or 64 kHz.

4.2.3.3 Slave interrupt Register (SPECS_SLAVE_INT) *--- Under development ---*

This 6 bit register returns the address of the slave which is emitting the interruption.

Table 6 - SPECS_Bus_int0 bit allocations (SPECS_SLAVE_INT of Master 0)

Bit	Name	Function
[5:0]	Slave_adr	Address of the slave

4.2.3.4 Slave mask interrupt Register (MASK_SPECS_SLAVE_INT) --- *Under development* -

This 6 bit register masks the different bit of the Slave interrupt Register.

Table 7 - MSKSPECS_Bus_int0 bit allocations (MASK_SPECS_SLAVE_INT of Master 0)

Bit	Name	Function
[5:0]	MSK_Slave_adr	Mask the address of the slave

4.2.3.5 PLX interrupt Register (PLX _INTERRUPT) --- *Under development* ---

This 2 bit register manages the different interrupts and generates a hardware interrupt on the PLX chip.

Table 8 - PLX_Bus_int0 bit allocations (PLX_INTERRUPT for Master 0)

Bit	Name	Function
1 (Read)	Slave_bus_interrupt	If 1, an interrupt occurred during the SPECS transfer (or of all SPECS_SLAVE_INT bits)
0 (Read)	Master_transfer_interrupt	If 1, the reading of the SPECS slave has ended. (Corresponding to the Ena_rd bit of the Status Register)
1 (Write)	Msk_slave_bus_interrupt_ackno	Set to 1 to acknowledge Slave_bus_interrupt
0 (Write)	Msk_master_transfer_interrupt_ackno	Set to 1 to acknowledge Master_transfer_interrupt

4.2.3.6 Mask PLX interrupt register (MSK_PLX_INTERRUPT) --- *Under development* ---

This register masks the different interrupt sources.

Table 9 - MSKPLX_Bus_int0 bit allocations (MSK_PLX_INTERRUPT of Master 0)

Bit	Name	Function
1 (Write)	Msk_slave_bus_interrupt	Mask of the Slave bus interrupt
0 (Write)	Msk_master_transfer_interrupt	Mask of the Master transfer interrupt
1 (Read)	Msk_slave_bus_interrupt_ackno	
0 (Read)	Msk_master_transfer_interrupt_ackno	

4.2.3.7 Test and date register (TEST_REG)

The MSB of the register contain the date of the design of the firmware of the Specs Master and the LSB allow read/write tests on the PCI interface.

Table 10 - Test and date register bit allocations

Bit	Name	Function
[31:16]	Date_prom	Date of the firmware design in MMDD format (Read only)
[15:0]	Test_reg	Test register (Read and write)

5 SPECS Slave

5.1 SPECS slave chip

The slave is designed as a portable VERILOG code and is eventually physically integrated in an ACTEL anti-fuse PGA. With this technology, the slave will be SEL immune, and will also be made SEU immune, provided the internal registers will appropriately be protected by triple voting techniques, and state machines will use one-hot state. Moreover, to ensure a high reliability of the decoding of the SPECS commands, there will be no state machine in the SPECS receiver part. In addition, all commands (and in particular all resets) will be generated without using any local clock but only the SPECS lines. This may for instance allow the user to reset the TTCrx through SPECS if necessary. The chip will also be reasonably radiation tolerant, up to 10 krad (with some safety margin, the ACTEL chip having been tested up to 40 krad) over the lifetime of the experiment. It can then be placed at most locations where we foresee to have electronics, except in or near the Velo tank. The chip is an ACTEL APA150.

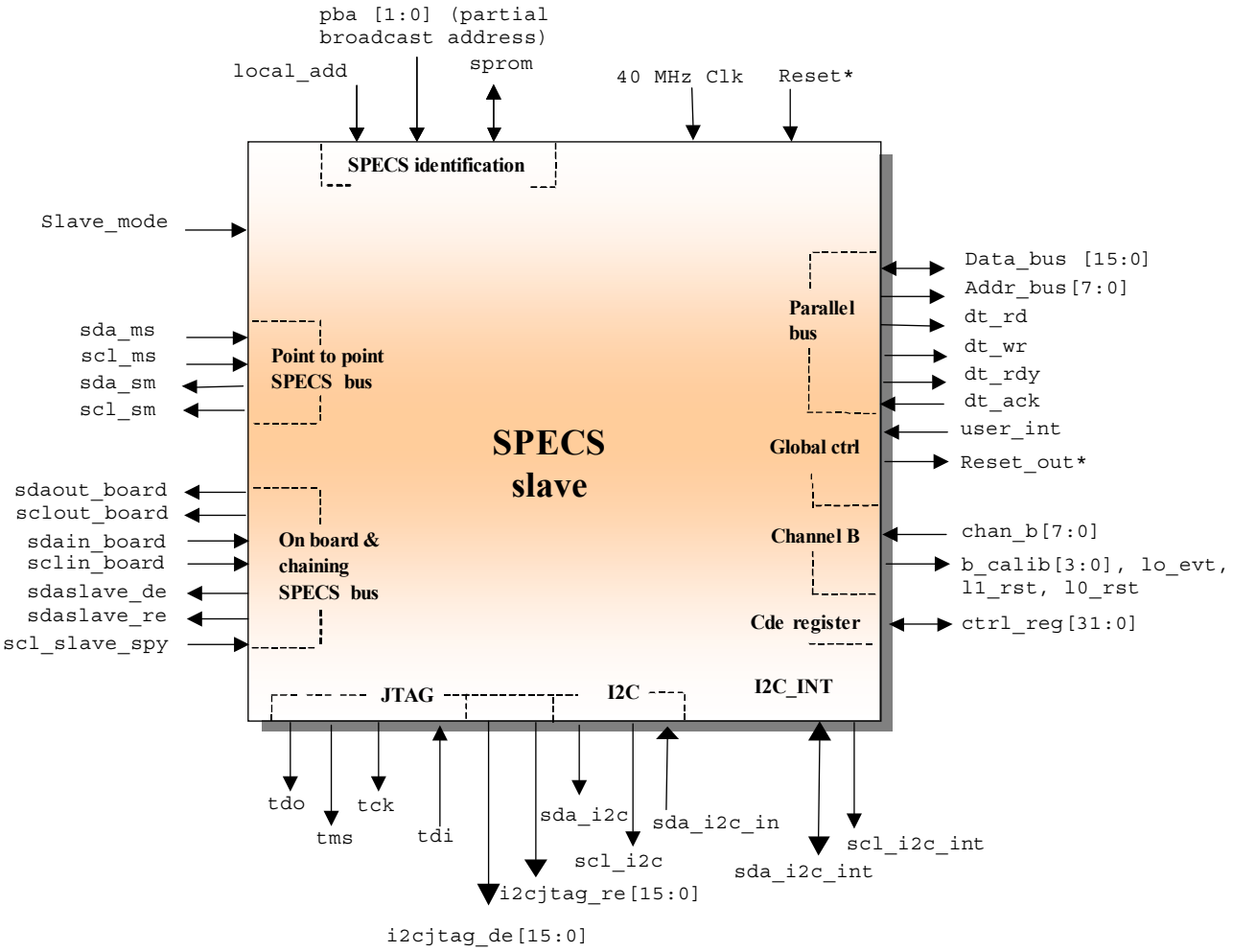


Figure 13 : Synopsis of the SPECS Slave chip interfaces

The synopsis of the chip's I/Os is shown in **Error! Reference source not found.**. It offers 7 different interfaces:

- Point to point SPECS slave interface, composed of 4 unidirectional signals `sda_ms`, `scl_ms`, `sda_sm`, and `scl_sm` that have to be connected to the differential LVDS drivers.
- Multi drop SPECS bus, composed of 4 unidirectional signals `sdain_board`, `sclin_board`, `sdaout_board`, and `sclout_board` that have to be connected to the other SPECS slaves. The `scl_slave_spy` signal allows the slave to check the state of the SCL line before taking control of it. The Receive Enable (RE) and Driver Enable (DE) lines (`sda_slave_re` and `sda_slave_de`) are the enable for the `sda_sm` and `scl_sm` LVDS drivers that follow the DS92LV010 logic.
- Local bus master interface delivering an easy to use parallel bus. This interface is composed of a bi-directional 16-bit data bus (`data_bus[15:0]`), for write and read operations. An 8-bit sub-address bus (`addr_bus[7:0]`), a write* pulse (`dt_wr`) and a read* enable signal (`dt_rd`) are provided with the data bus.
- 32 configuration I/O lines (`ctrl_reg[31:0]`): these I/O lines can be individually configured in input or output. This functionality can be done without any clock.
- Reset I/O (`rst_reg`): dedicated output pin for reset.
- User Interrupt input (`user_inter`): propagates an interrupt into the SPECS bus.
- JTAG master interface: provides the usual `tdi`, `tms`, `tck`, and `tdo` signals. The 16 Receive Enable (`i2cjtag_re`) and 16 Driver Enable (`i2cjtag_de`) busses allow the slave chip to drive up to 16 independent JTAG busses. In this case, each bus is controled by drivers, using the unique JTAG bus. The logic respects the DS92LV010 command logic to enable these drivers.
- Long distance I2C master bus: composed of three signals, `sda_i2c`, `scl_i2c`, and `sda_i2c_in`. Similarly to JTAG, the 16 Receive Enable (`i2cjtag_re`) and 16 Driver Enable (`i2cjtag_de`) busses allow the slave chip to drive up to 16 independent I2C links. The same command is used for I2C and JTAG busses so that users can mix both I2C and JTAG up to 15 busses, configured by the "output select" value.
- Internal I2C: this bus (`sda_i2c_int` and `scl_i2c_int`) is used to communicate with the on-board I2C slaves (Output select = 'hF).
- Channel B decoding: 4 commands are decoded from the channel-B: L0 front-end reset (`l0_rst`), L1 front-end reset (`l1_rst`), L1 event-ID reset (`lo_evt`) and Calibration pulse type 0 (`b_calib[3:0]`). The last one can be delayed by a programmable delay (1 to 255 with 25 ns steps).
- Serial Prom interface: reads back a serial prom (`sprom_d0`, `sprom_clk`, `sprom_oerst`, and `sprom_ce0`). This Prom contains registers to access the identification of the board, the serial and revision number and allows to implement user defined registers and test registers.

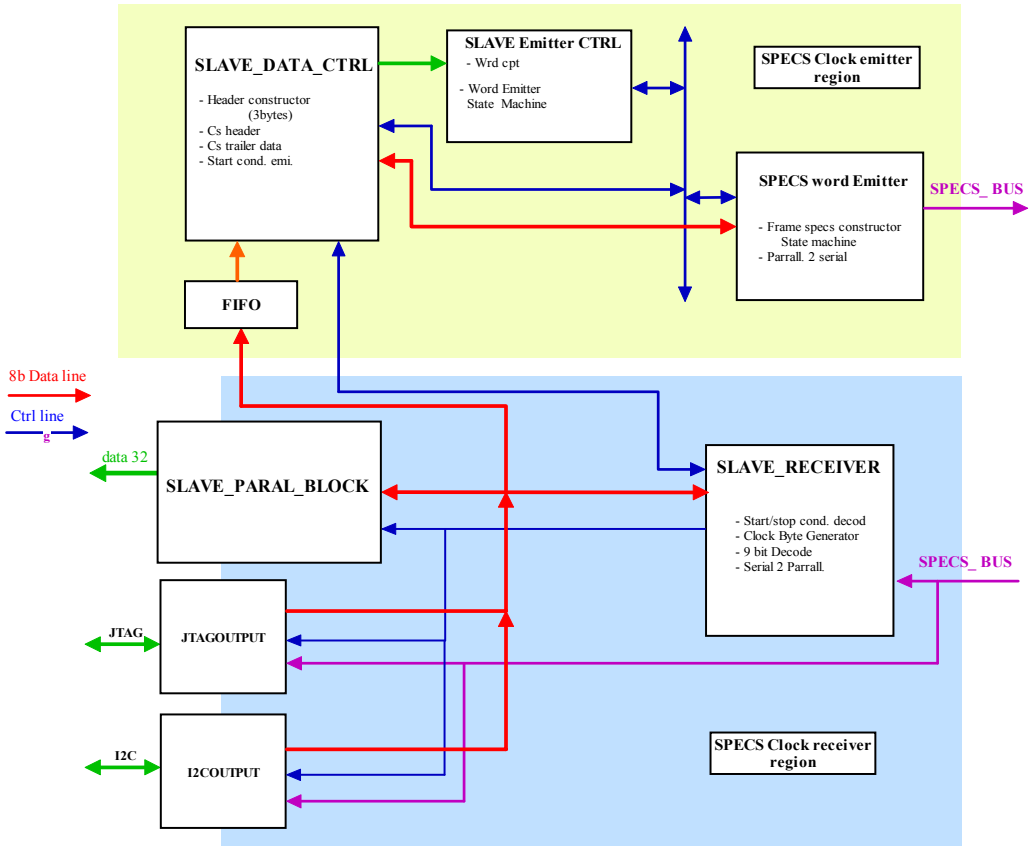


Figure 14 : Internal block diagram of the slave chip upon which the Verilog code is based

5.2 SPECS slave mezzanine general description

An intermediate mezzanine board can be provided by LAL-Orsay to house the SPECS slave, and to provide all the described functionalities of the SPECS Slave chip using 2 SMC connectors. The mezzanine board also provides most of the necessary service functions for the sub-detector front-end electronics. The goal is to avoid putting unnecessary electronics in the radiation sensitive area.

The mezzanine can be configured in master mode or in slave mode. In master mode, the mezzanine delivers 2 SPECS busses:

- 1 point to point long distance bus, this bus is implemented with RJ45 connectors on the mezzanine and allows a direct connection to the master by an Ethernet cable,

- 1 multi-drop SPECS bus, on the SMC connector. On this bus one can connect up to 31 slaves. To connect further mezzanines, one has to respect the signal integrity on this bus (see chapter 6.2).

In slave mode, the mezzanine controls the SPECS bus on the SMC connector. On this bus, up to 31 slaves can be connected (with the same rules as before, see chapter 6.2).

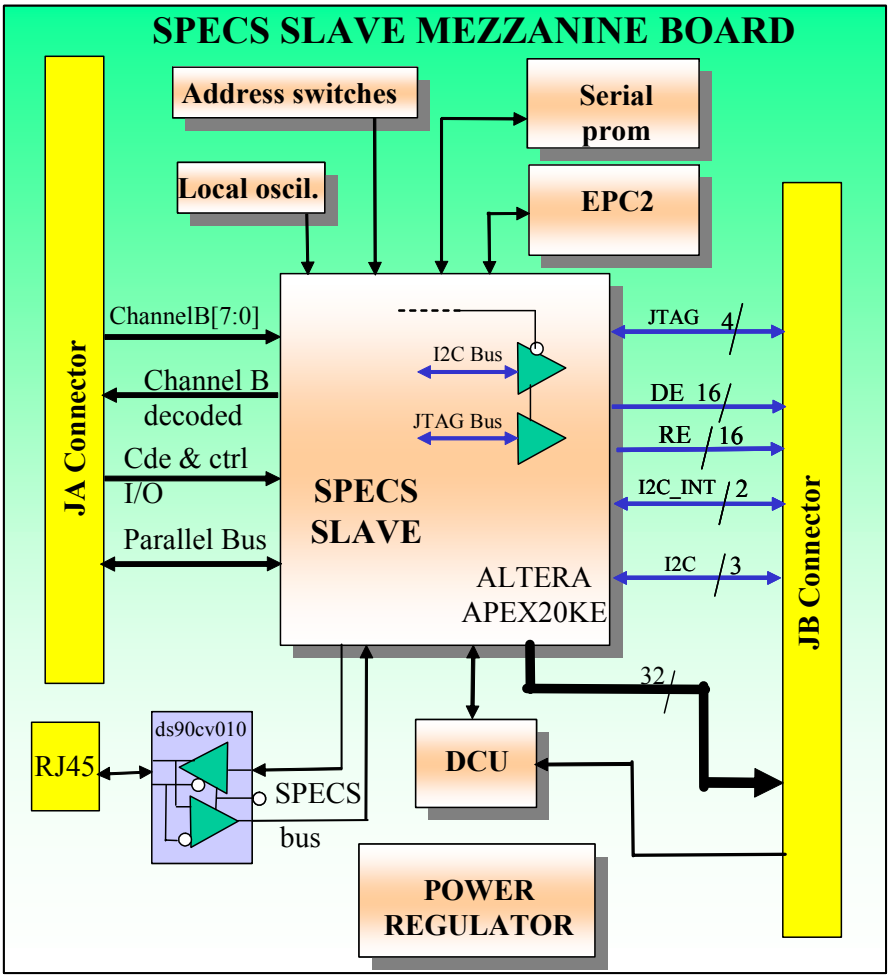


Figure 15 : SPECS Slave Mezzanine Board

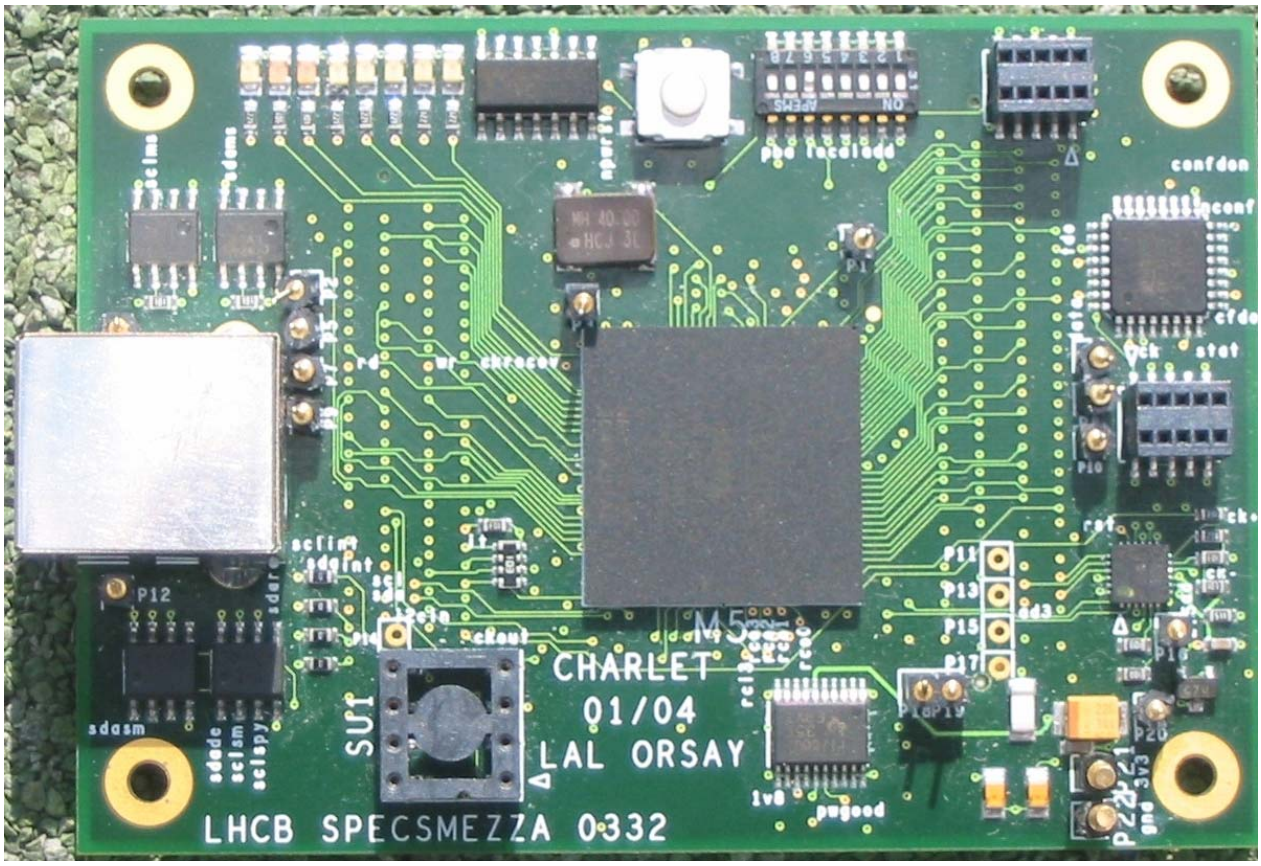


Figure 16 : Photograph of a SPECS prototype Mezzanine Board

5.2.1 List of the features of the mezzanine board:

- One long distance point to point differential SPECS interface (coming from the SPECS master).
- One unipolar SPECS local interface for multi-load bus applications.
- One output for long distance I2C.
- One JTAG bus.
- 16 JTAG or 15 I2C chip-select control bits for external drivers.
- 16 JTAG or 16 I2C direction control bits for external drivers.
- One local I2C bus.
- One parallel bus offering 16 data bits and 8 address bits.

- One decoder for the channel B of the TTCrx:
    - L0 front-end reset,
    - L1 front-end reset,
    - L1 event ID reset,
    - Calibration pulse type 0, which may be delayed with a programmable counter.
- One 32-bit static register to control or read back the local environment. The bits [31:24] can be individually configured either in output or in input mode. In input mode, they may be configured as an interrupt vector, each of them being then able to generate a SPECS interrupt. The bits [23:0] can be individually programmed in input or output mode. This register does not need any external clock to be written by the SPECS. After a hardware reset, all I/Os are configured in input mode by default for safety reasons.
- One reset signal. This output can be triggered without the need of any clock on the board.
- One local 40MHz oscillator. It is also provided as an output of the mezzanine and can be enabled by the software. In case of LHC clock failure, *it automatically replaces the LHC clock to allow the read-back over the SPECS bus.(Under development)*
- One PROM which will allow the ECS system to obtain some information about the front-end element housing the mezzanine. It will be mounted on a socket.
- One DCU chip with 6 ADC channels of 12 bits resolution.
- 6 switches to fix the SPECS slave address and 1 switch to allow the broadcast access

5.3 Mezzanine physical implementation

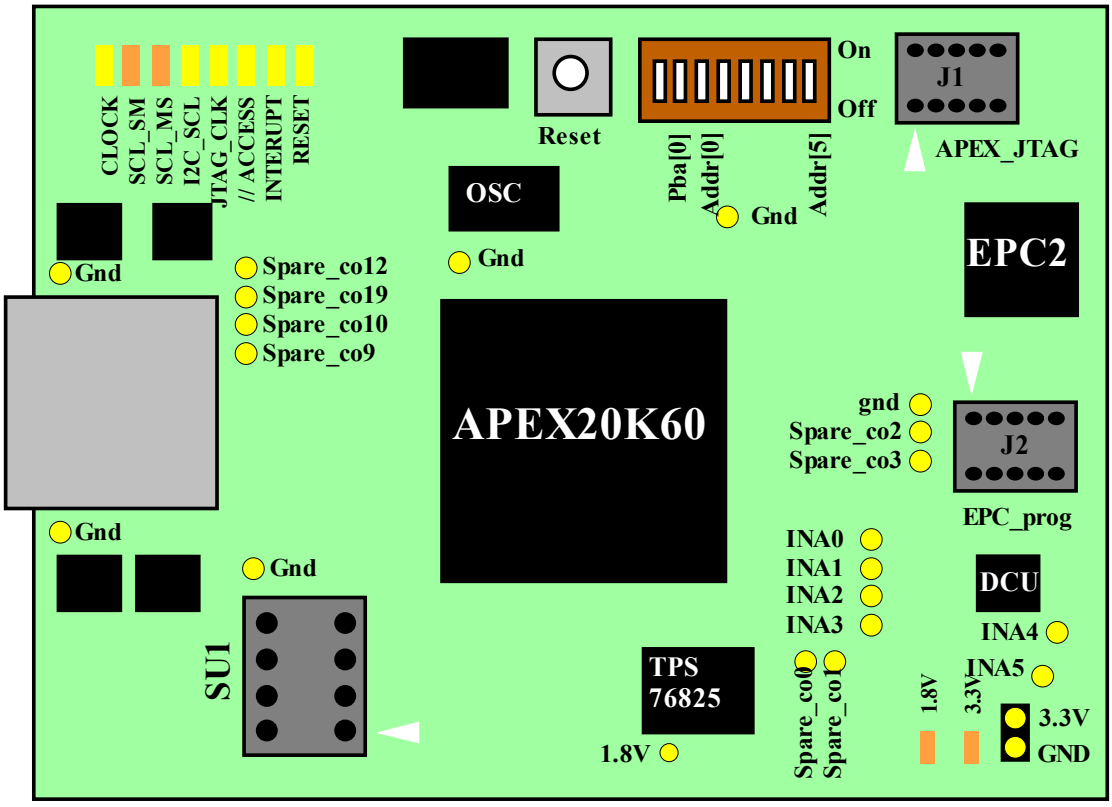


Figure 17 : Schematics of the physical implementation of the mezzanine

Note that t*he LED functions can vary depending on the mezzanine version*.

5.4 SPECS slave functional description

All commands coming from the master are sent without the need of any external clock. For reception, all mezzanine logics use the clock of the SPECS bus. For slave to master commands, the mezzanine uses the local oscillator or the master board clock for the logic.

The different functions are selected by the SPECS header frame. The number of SPECS data is limited to 255 but in serial access (I2C or JTAG) there is no limitation on the data transfer size.

The SPECS clock can be changed in the master (using the CONTROL_REG register), then the timing for slave access depends on the selected frequency.

5.4.1 I2C bus

One can implement on this board up to 15 long distance I2C outputs and one short distance I2C output for onboard applications, if necessary. The SPECS system is seen in this case as a I2C provider, and its internal features can be completely ignored. The use of the long distance differential I2C provided by the SPECS slave is mandatory to communicate with boards situated in high radiation areas, like the Velo detector boards, since it is not possible to put the SPECS FPGA at these locations.

1) The long distance I2C bus is based on 3 unidirectional signals : `scl_i2c`, `sda_i2c`, and `sda_i2c_in`. These signals must be buffered by drivers. For long distance communication, LVDS drivers seem to be the best choice. On the receiver part, the equivalent of an open collector must be implemented (see chapter 6.3). The frequency of the I2C signals can be selected by a register in the master and decreased for slow I2C slaves (1 MHz, 500 kHz, 250 kHz, 125 kHz, or 65 kHz).

To implement additional busses, 15 selection signals and 15 direction signals (`i2cjtag_de` and `i2cjtag_re`) are provided. These signals can be directly connected to a DS92LV010 transceiver without any extra logic and they are dynamically driven by the SPECS slave frame. The `i2cjtag_de` signals (but not the `i2cjtag_re` signals) can alternatively be commanded statically (e.g. for external multiplexer commands) by a 16 bit internal register. This mode has to be selected in the MEZZA_CTRL register.

2) The internal I2C bus: for onboard I2C slaves, an I2C bus is implemented. It is based on 2 signals: `scl_i2c_int` and `sda_i2c_int`. It is selected with OUTPUT_SELECT = ‘hF. The DCU chip on the mezzanine is connected to this bus. This implies that a 2.5 V logic level is used instead of the usual 3.3 V level.

5.4.2 JTAG bus

One can implement on this board up to 16 JTAG outputs if necessary. The SPECS system is seen in this case as a JTAG provider, and its internal features can be completely ignored. The use of the long distance differential JTAG provided by the SPECS slave is mandatory for boards situated in high radiation areas, like the RICH detector boards, since it is not possible to put the SPECS FPGA at these locations. The SPECS JTAG is composed of 4 signals: `tdo`, `tdi`, `tck`, and `tms`. For long distance, differential drivers can be implemented. The bus selection is done using the same signals than the I2C bus (`i2cjtag_de` and `i2cjtag_re`).

5.4.3 Parallel bus

A 16 bit data bus with 8 addresses, a read signal, and write signal are delivered on the mezzanine. 2 access modes are possible:

1) Address mode: in this mode, the address is send by the master in the header, followed by the 2 data bytes. Then one SPECS frame is needed for a 16 bit data access.

2) DMA mode: in this mode, the address is sent by the master in the header, followed by up to 255 data bytes. The address is automatically incremented by one unit after each 16 bit data access. The timing of bus signals in write access depends on the SPECS clock frequency and, for read access, depends on the mezzanine clock (40 MHz). The logic level is LVTTL.

5.4.4 Configurable I/O (Static register)

On the mezzanine, 32 user configurable I/O are implemented. They can be read or controlled by two 16 bit registers, REGOUT_MSB and REGOUT_LSB. These registers can be configured either in output or in input mode by 2 internal registers, CONFREGOUT_LSB and CONFREGOUT_MSB. By default (at power on or after Reset), the registers are configured in input mode to avoid any conflict with the signal. *8 bits of REGOUT can be configured as a vector interrupt (Under development).* The logic is LVTTL.

5.4.5 DCU Adc

A DCU2 chip is implemented on the mezzanine. It can be accessed by the internal I2C bus (OUTPUT_SEL = ‘hF). The address of the chip is fixed to 0 but the user can change this address

using the JB 12, 14, 16, and 18 pins. The 6 input channels are linked to the JB 5, 6, 7, 8, 9, and 13 pins. One 10 Ω resistor is implemented on each input.

5.4.6 Channel B decoding

All Channel-B[7:2] and the 2 broadcast strobe signals Channel-B[1:0] are connected to the SPECS slave to allow decoding and synchronization. Currently 4 commands are decoded:

1. L0_FRONT-END-RST: A synchronous pulse with SRTB1 is delivered.
2. L1_FRONT-END-RST:A synchronous pulse with SRTB1 is delivered.
3. L1_EVENT_ID_RST: A synchronous pulse with SRTB1 is delivered.
4. CALIB[0]: This signal can be delayed with a programmable delay up to 255 ns, writing the value of the delay in the mezzanine register CHANB_DELAY. The clock of the counter is the selected mezzanine clock. A pulse of one clock cycle is generated.

5.4.7 Serial PROM (under development)

A serial PROM is implemented on the mezzanine to integrate specific registers on the mezzanine, these registers are only in read access. The PROM is plugged on a DIL 8 socket what allows the user to change the register values. The PROM is equivalent to the XC17XX PROM of XILINX. A logic is implemented on the slave to access this register after a master read command.

5.5 Slave registers

The registers contained in the slaves are summarized in the table below. They are described in more details in the following sections.

Table 11 - Slave registers

Register	Name	Address	Access
Test Register (Internal slave Only)	TEST_MSB	‘h0	READ/WRITE
	TEST_LSB	‘h1	READ/WRITE
Output pins control Register	REGOUT_MSB	‘h0	READ/WRITE
	REGOUT_LSB	‘h1	READ/WRITE
Output pins configuration Register	CONFREGOUT_MSB	‘h2	READ/WRITE
	CONFREGOUT_LSB	‘h3	READ/WRITE
Control Register	MEZZA_CTRL	‘h4	READ/WRITE
Status Register	MEZZA_STAT	‘h5	READ
Serial Bus Selection Register	BUS_SEL	‘h6	READ/WRITE
Interrupt Register	INT_VECT	‘h7	READ/WRITE
Date Register	DATE_PROM	‘h8	READ
Masks of Interrupt Register	DEFINT_VECT	‘h9	READ/WRITE
Channel B Register	CHANB_DELAY	‘hA	READ/WRITE

5.5.1 Test Register

These 16 bit registers are implemented only in the internal slave. They allow reading and writing tests.

Table 12 - Test Register bit allocations

Bit	Name	Function
[15:0]	TEST_MSB	Tests
[15:0]	TEST_LSB	Tests

5.5.2 Output pins Control Register

These 16 bit registers allow to control the 32 output pins of the slave, when they are configured in output mode using the CONFREGOUT_MSB and CONFREGOUT_LSB. When they are configured in input mode by the same registers, the status of the 32 input pins is read using these Output pins Control Registers.

Table 13 - Output pins Control Register bit allocations

Bit	Name	Function
[15:0]	REGOUT_MSB (Pins 16 to 31)	Write in Bit #i to control output pin #i or read in Bit #i to read the status of input pin #i. The read/write mode is modified using the CONFREGOUT registers.
[15:0]	REGOUT_LSB (Pins 0 to 15)	

5.5.3 Output pins Configuration Register

These 16 bit registers configure the 32 output pins in input or output mode. They can also be used for read/write tests.

Table 14 - Output pins Configuration Register bit allocations

Bit	Name	Function
[15:0]	CONFREGOUT_MSB (Pins 16 to 31)	Set bit #i to 1 to configure pin #i in output mode (it is configured in input mode – value 0 -- by default)
[15:0]	CONFREGOUT_LSB (Pins 0 to 15)	

5.5.4 Control Register

This 4 bit register allows to configure the mezzanine. (NB: All mezzanine registers can be controlled without any clock, even the Cmd_oscillator register)

Table 15 - Control Register bit allocations

Bit	Name	Function
3	Stadyn_mode	Set to 1 to use static mode, 0 for dynamic mode for the control of `i2cjtag_de` lines.
2	Cmd_oscillator	Source of the mezzanine clock: if 0, local oscillator; if 1, use external clock and disable local oscillator.
1	Output_external_reset	Control the RST_REG pin of the mezzanine, or-ing with the global reset.
0	Global_software_reset	Global reset of the mezzanine, or-ing with the switch reset.

5.5.5 Status Register --- *Under development* ---

This register gives the status of the mezzanine.

Table 16 - Status Register bit allocations

Bit	Name	Function
0	Mastslave	If 0, Mezzanine is in Master mode, if 1, in Slave mode.

5.5.6 Serial Bus Selection Register

This 16 bit register controls the `i2cjtag_de` lines (only `i2cjtag_de` lines. `i2cjtag_re` lines cannot be controlled in static mode.)

Table 17 - Serial Bus Selection Register bit allocations

Bit	Name	Function
[15:0]	Bus_sel	In static mode (set by the Stadyn_mode bit of the control register), control `i2cjtag_de` lines.

5.5.7 Interrupt Register --- *Under development* ---

This register gives the type of the interrupt generated by the slave. The OR of all interrupt bits generate a slave interrupt. The register is cleared before each reading access.

Table 18 - Interrupt Register bit allocations

Bit	Name	Function
[15:8]	Int_vector	Status of the `ctrl_reg[31:23]` input/output pins, configured in interrupt mode.
2	User_int	Status of the `user_inter` input/output pin.
1	I2C_ack	No I2C acknowledge.
0	Check_err	Error on the header or the trailer checksum.

5.5.8 Date Register

This register gives the date of the creation of the PROM of the Mezzanine.

Table 19 - Date Register bit allocations

Bit	Name	Function
[3:0]	PROM_Date	Date of the PROM in format MMDD.

5.5.9 User defined Interrupt Register --- *Under development* ---

This register configures the Slave Interrupt register.

Table 20 - Mask Slave Interrupt Register bit allocations

Bit	Name	Function
[15:8]	Int_vector	
2	User_int	
1	I2C_ack	
0	Check_err	

5.5.10 Channel B Register

This register allows to program the Channel B.

Table 21 - Channel B Register bit allocations

Bit	Name	Function
[7:0]	ChanB_delay	Delay Calibration pulse type 0 of Channel B with 25 ns steps.

6 SPECS Use Cases

6.1 Calorimeter implementation

As described in this note, the SPECS protocol is mono-master multi-slave. This permits the implementation of many slaves on the same bus. For the Calorimeter electronics, the SPECS bus will actually be distributed on the remote crate backplane by the so-called Crate Controller board (CROC), which also takes care of the TTC signals for all the boards in the crate (see Figure 18). This allows us to ensure a perfect integrity of the signals. This dedicated backplane already exists, and is potentially available for use by other detectors.

In this implementation, a mezzanine mounted on the CROC board (configured in master mode) forwards the SPECS bus towards all the other boards in the crate. These house their own SPECS interface, in this case integrated in an ACTEL FPGA which is shared with other logics.

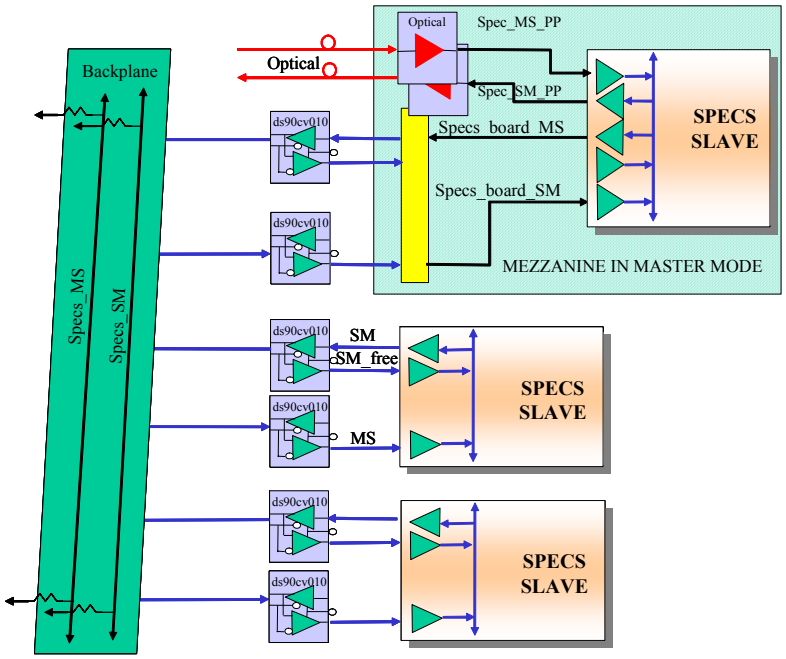


Figure 18 : Calorimeter implementation of the SPECS bus

6.2 Multi Mezzanine implementation

On the SPECS mezzanine, we provided 2 SPECS interfaces : one point to point for long distance interconnection, one for local bus connection. Thanks to this feature, we can handle several mezzanines on the same SPECS bus. Over a backplane and for short distance, this can be done by interconnecting the lines `sdaout_board`, `sclout_board`, `sdain_board`, `sclin_board`, and `scl_slave_spy` of the different mezzanines (Figure 19). If the number of mezzanines exceeds two, it is mandatory to adapt the bus at both ends (~470 Ω). In this case, tests have to be performed in order to see if bus drivers are necessary. Another important point is to configure the main mezzanine in master mode, this being done thanks to pull down the master slave pin JA48 to (0 for master).

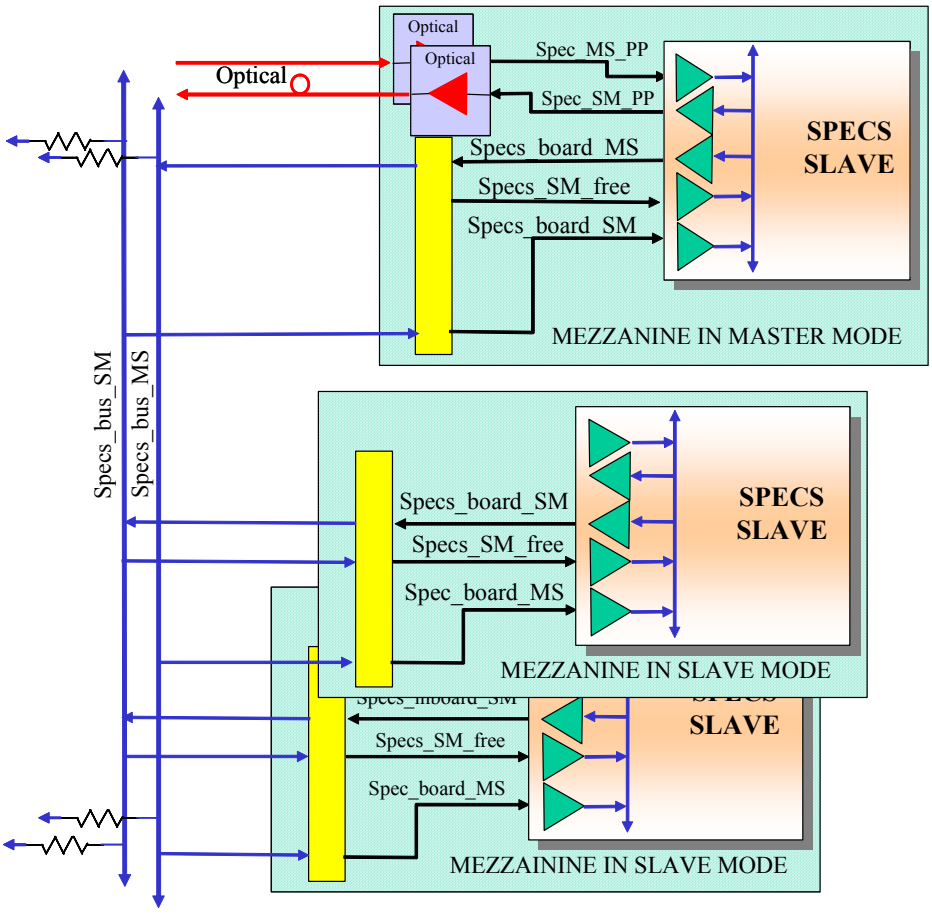


Figure 19 : Multi Mezzanine Implementation

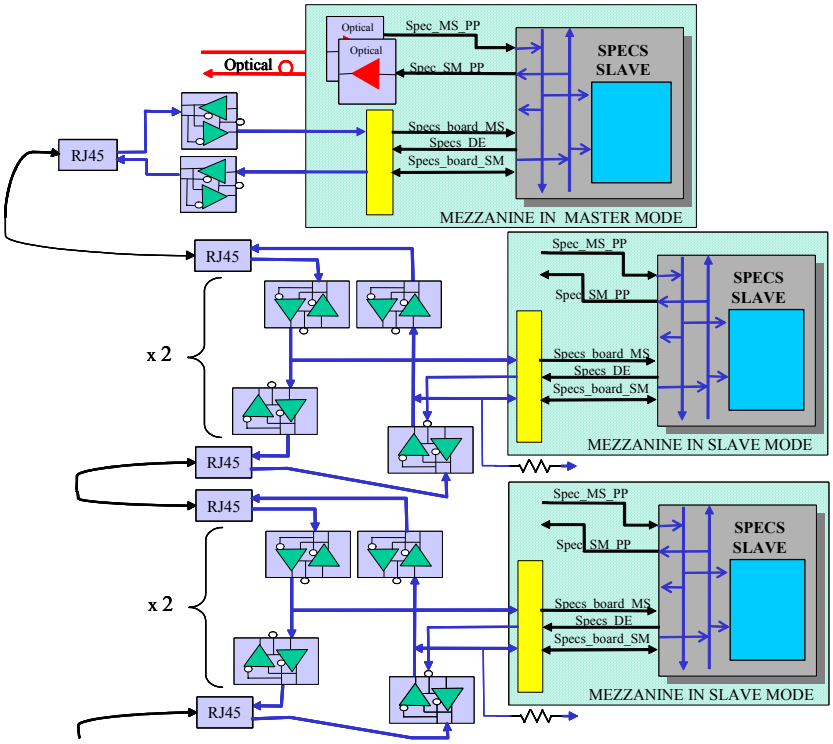


Figure 20 : Chaining of mezzanines on a cable

Figure 20 shows how mezzanines can be chained by cable, this structure needs 8 drivers for each slave (DS92LV010). With this logic, the link between slaves is point to point, there is no limit for cable length (up to 40 m) and also for the number of mezzanines chained (up to 32). To reduce the number of drivers, the DS92LV040 driver has been validated and can be used.

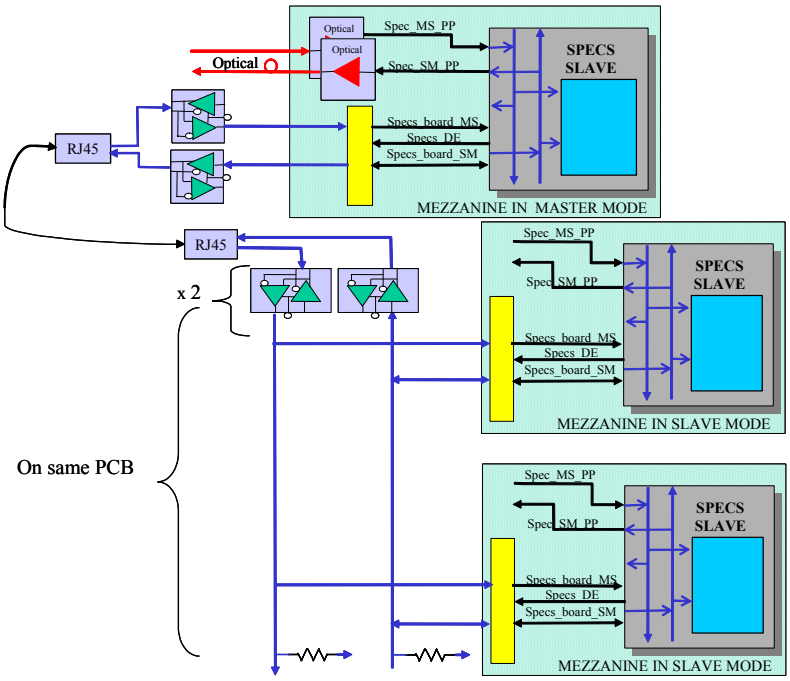


Figure 21

A combination of the 2 previously described solutions can be used: a long distance connection between the mezzanine in “master mode” and the other slaves, which are implemented on a same PCB (Figure 21).

6.3 How to make long distance I2C to local I2C

The I2C slaves on the user’s board need 2 I2C links for local I2C use that have to be extracted from the 3 I2C links needed for long distance I2C. A simple interface based on 4 LVDS transceivers is described in Figure 22.

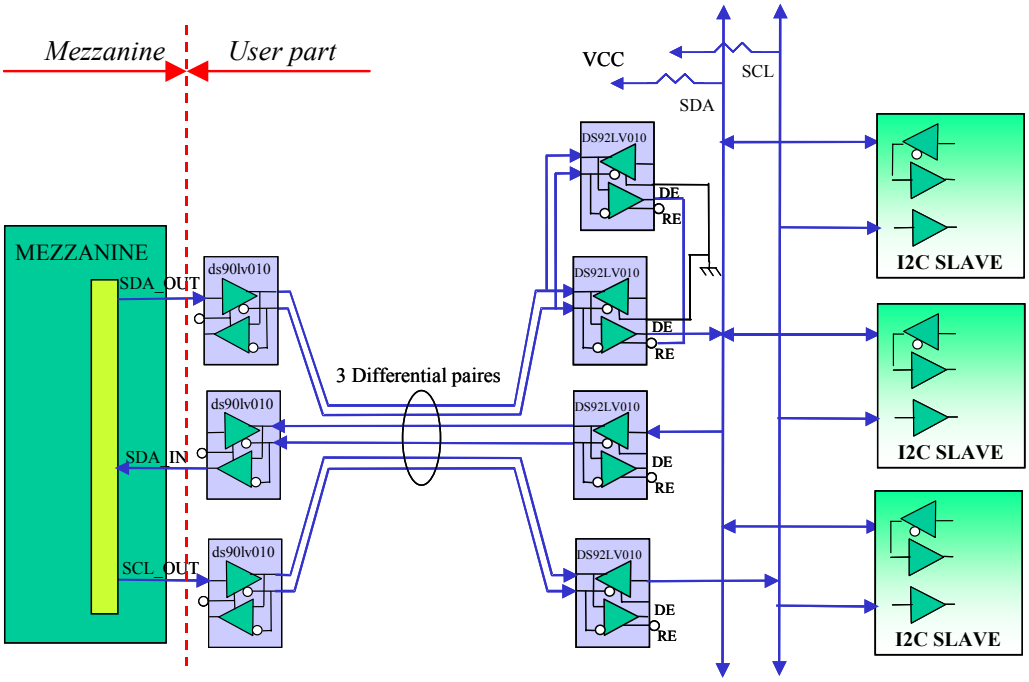


Figure 22 : Long distance to local I2C

The chips on the remote board may have to be resistant to radiation. The DS92LV010 LVDS Transceiver has been tested up to 25 krad [2]. The 10H116D receiver has already been validated by ATLAS at a 100 krad level and for unipolar logic, F125 has also been validated by ATLAS and can also be used.

6.4 Technical Specifications

6.4.1 How we make JTAG and I2C from SPECS

The goal here is to realize JTAG and I2C interfaces in the simplest possible way, in order to limit as much as possible the amount of electronics in the radiation sensitive area. We actually take benefit of the fact that the SPECS bus can carry 8 Mbit of data per second whereas JTAG and I2C are usually limited to 1Mbit/s. This factor 8 allows to transfer 1 bit of data on the JTAG or I2C bus

from 1 byte of SPECS data, making use of all 8 bits of the SPECS data to produce the different necessary control signals. This is described on the following figures.

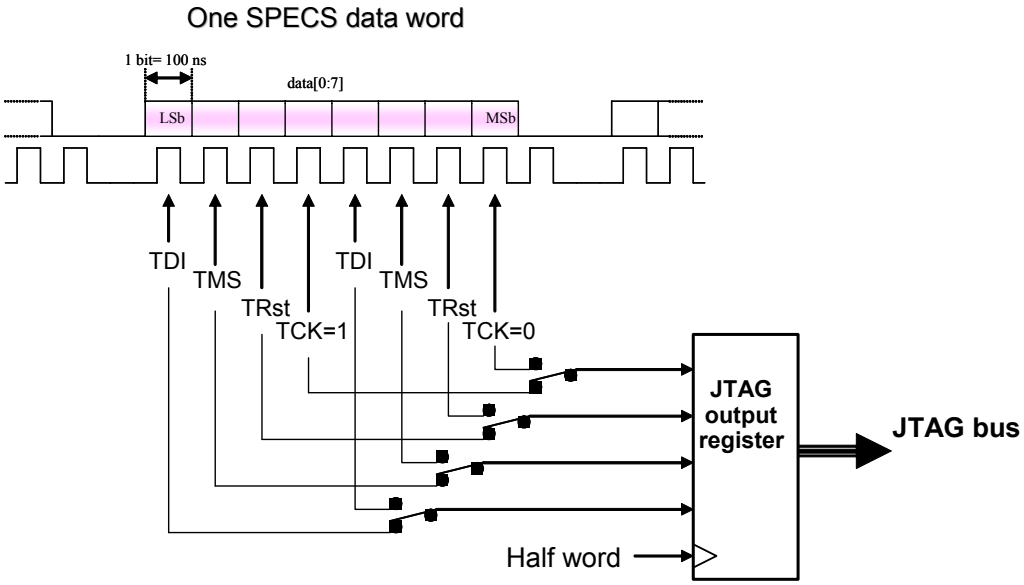


Figure 23 : SPECS to JTAG Interface

In JTAG (Figure 23), as there is a half period of clock phase relation between lines, we can produce the 4 lines corresponding to one bit transfer directly from the SPECS byte. The maximum SPECS frame length being 256 words, the transfers will be limited to 256 bits at a time, but they can be accumulated one after the other.

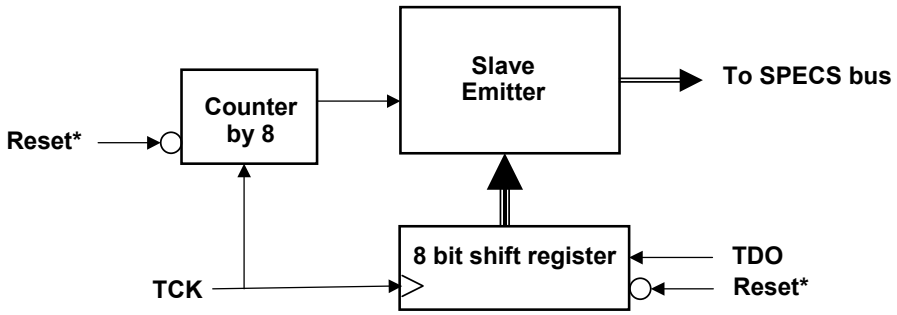


Figure 24

The way we deal with the TDO return line is shown on Figure 24. Every time a whole byte has been transferred on TDI, another byte is sent back from TDO by the SPECS slave towards the master in the standard read operation format.

Figure 25 displays the way we deal with I2C. In this case, there are only two signals but there is a quarter of clock period phase relation between them. Thus we divide the SPECS byte into 4 phases. We need a SPECS byte for each of the start and stop conditions, and 8 SPECS bytes for the I2C slave address. So 27 bytes of data will be available in one single SPECS transfer, but longer chains can also be transferred if they are spread over successive SPECS frames sent towards the same bus. The I2C state machine can indeed be asked to restart from its last position (no reset) by a special command.

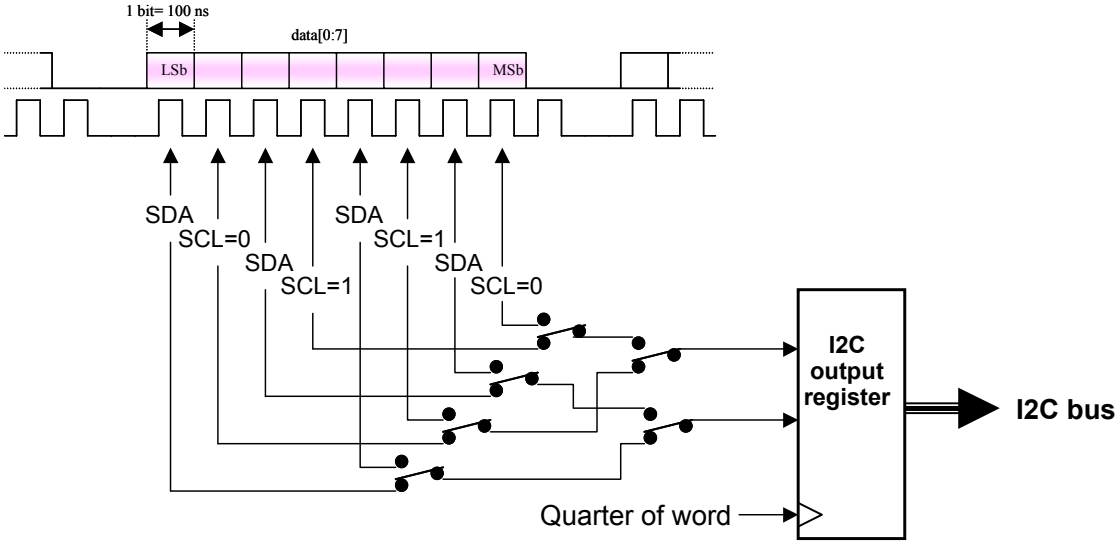


Figure 25 : SPECS to I2C Interface

The main difficulty with I2C is the acknowledge signal. In order to limit the number of transfers on the SPECS bus, it is not sent back to the master. However it is checked at the slave level. The slave warns the master only the acknowledge is missing, sending an interrupt order. Then the master will stop sending data and will finish the frame with a stop condition.

Concerning the data readout through I2C, it is done almost the same way as for JTAG, as shown on Figure 26

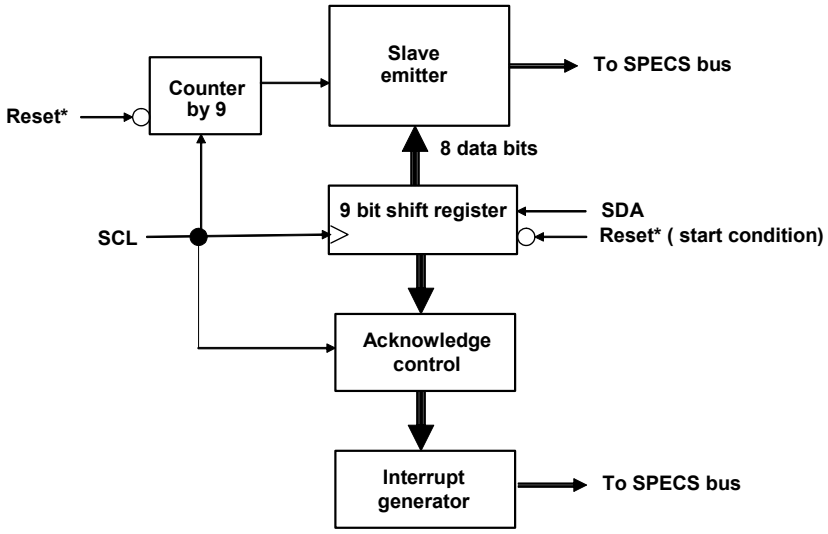


Figure 26

The SPECS frames are prepared by a dedicated software library. The user will give it the necessary information for the transfer and it will produce the frames which will be stored in the SPECS master board.

6.4.2 SPECS connector

The SPECS connector delivers 2 differential output pairs and 2 differential input pairs in LVDS levels *(*Figure 27*)*. It follows the EIA/TIA T568B standard for Ethernet. Standard Ethernet cables can then be used.

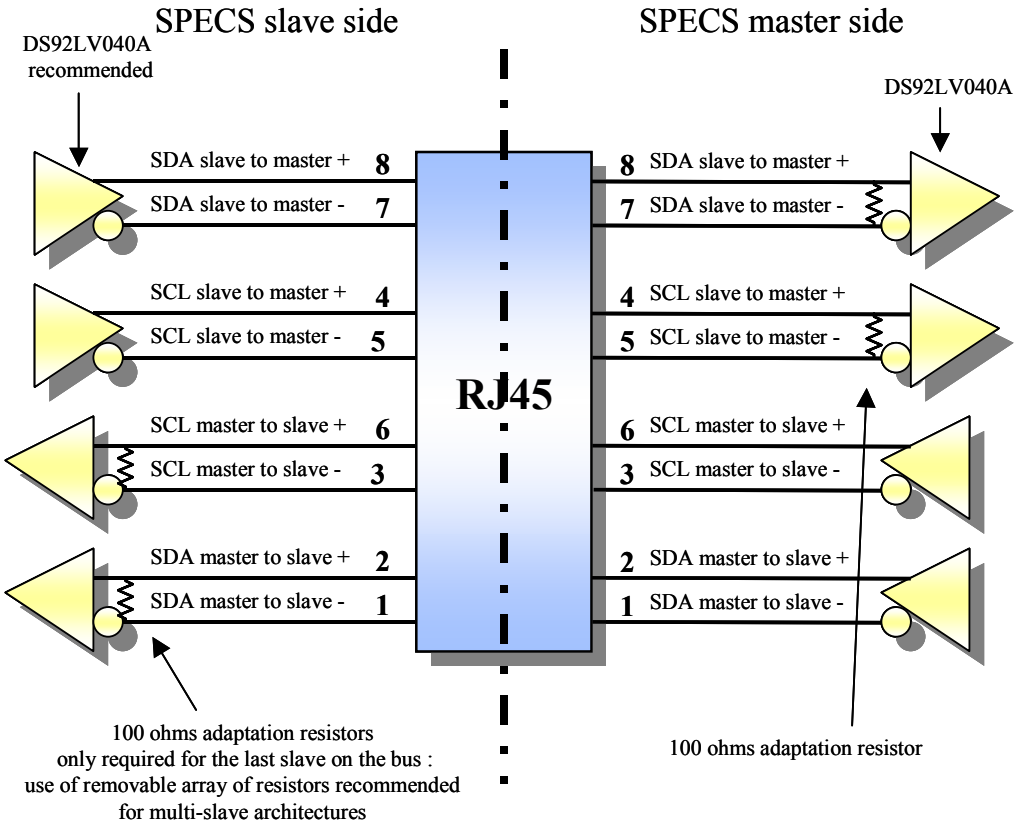


Figure 27 : SPECS Connector

6.4.3 JTAG connector implementation

A JTAG connector providing the TDI, TDO, TMS, and TCK signals of the JTAG bus in differential mode (LVDS) can be implemented following Figure 28. For this connector, the SPECS slave is also the JTAG master, and is thus represented on the right side of the figure.

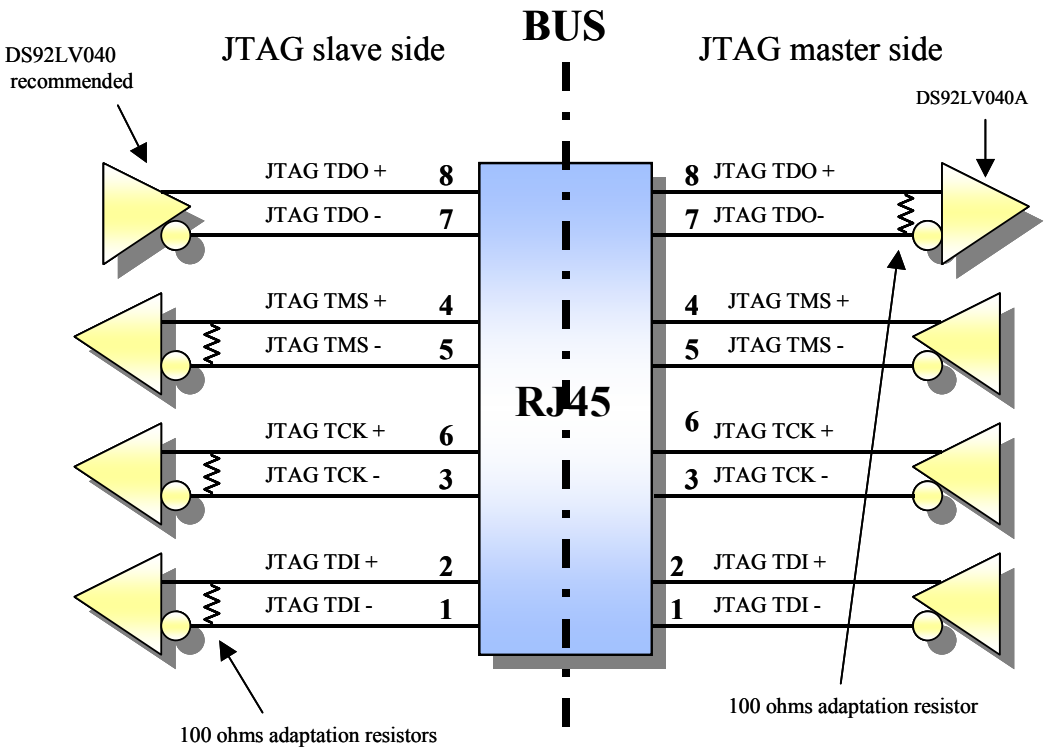


Figure 28 : JTAC Connector

6.4.4 I2C connector implementation

An I2C connector providing the SDA and SCL signals in unidirectional and differential mode can be implemented following Figure 29 . 4 pins are necessary for the SDA lines and all signals are in LVDS technology. The SPECS slave is also the I2C master, and is thus represented on the right side of the figure.

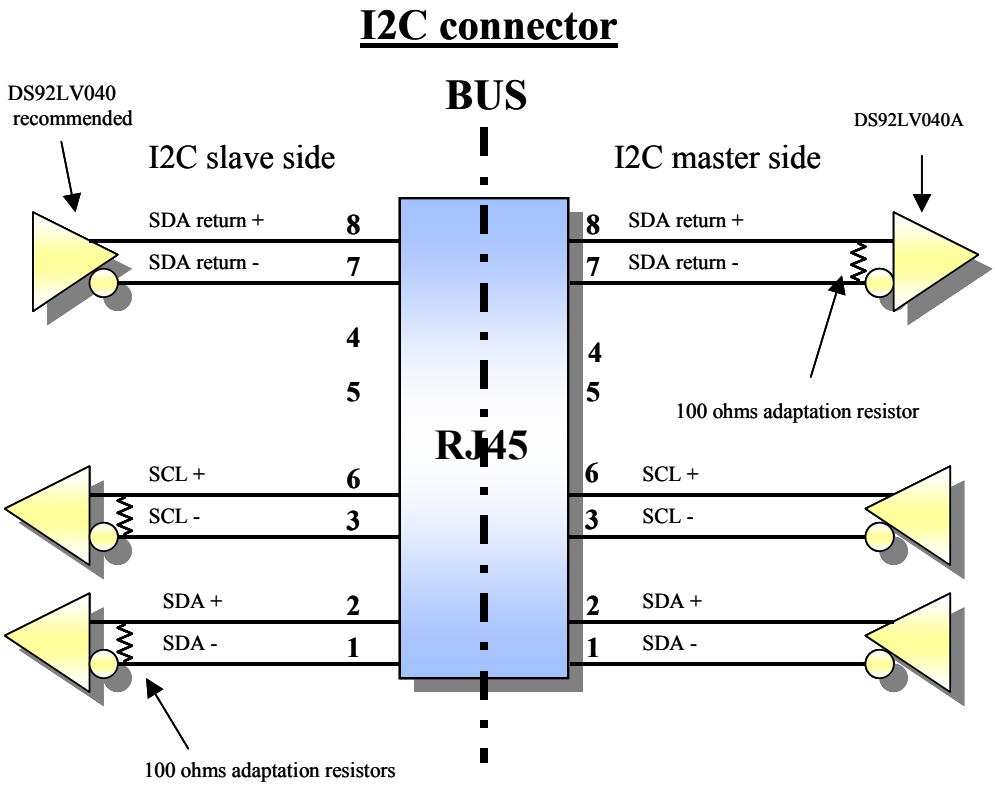


Figure 29 : I2C Connector

6.4.5 RJ45 connector

The pin numbers of the RJ45 connector are described in Figure 30. The connector on the SPECS master board is a FCI 95678-004-00 (4 connectors per part, the FCI 95678-001-00 is the equivalent single connector version).

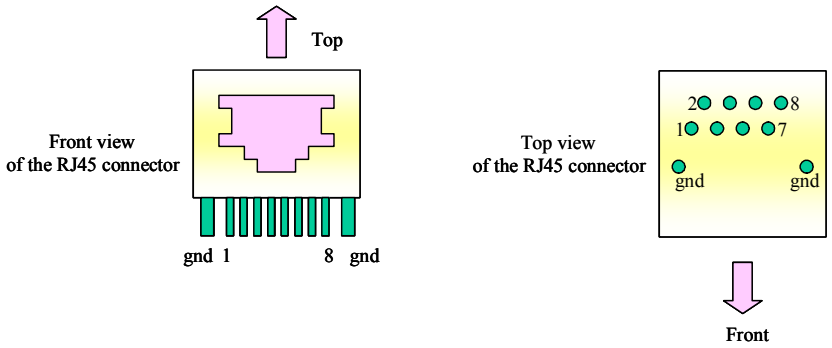


Figure 30 : RJ45 Connector

Pin Name	Pin Description	Logic	Dir	Pin Number
sda_ms	SPECS Bus: data from master +/-	LVDS	I	**RJ45** 2/1
scl_ms	SPECS Bus: clock from master +/-	LVDS	I	**RJ45** 6/3
sda_sm	SPECS Bus: data from slave +/-	LVDS	O	**RJ45** 8/7
scl_sm	SPECS Bus: clock from slave +/-	LVDS	O	**RJ45** 5/4

Table 22 – RJ45 Connector

6.4.6 Mezzanine pinout

PIN NAME	PIN DESCRIPTION	Levels	Pull up	Dir	pin number
SDAOUT_BOARD	SPECS bus : SPECS data output data for multi-mezzanine on board (with pull-up at each end of the bus)	LVTTL	Up user	O	**JA** 47
SCLOUT_BOARD	SPECS bus : SPECS clock output for multi-mezzanine on board (with pull-up at each end of the bus)	LVTTL	Up user	O	**JA** 49
SDAIN_BOARD	SPECS bus : SPECS data input data for multi-mezzanine on board (with pull-up at each end of the bus)	LVTTL	up	I	**JA** 43
SCLOUT_BOARD_DE	SPECS bus : command driver for for multi_mezzanine on board	LVTTL		O	**JA 55**
SCLOUT_BOARD_SPY	SPECS bus : for spying the SCLOUT_BOARD line activity	LVTTL	up	I	**JA** 53
SCLIN_BOARD	SPECS bus : SPECS clock input for multi-mezzanine on board (with pull-up at each end of the bus)	LVTTL	Up	I	**JA** 45

SDA_ I2C_ INT	On-board I2C data bus (with pull-up at each end of the bus)	LCMOS2. 5V	up	I/O/ T	**JA** 37
SCL_ I2C_ INT	On-board I2C clock bus (with pull-up at each end of the bus)	LCMOS2. 5V	up	O	**JA** 39
SDA_ I2C	I2C bus : long distance I2C data output	LVTTL	up	O	**JA** 33
SCL_ I2C	I2C bus : long distance I2C clock output	LVTTL	up	O	**JA** 35
SDA_ I2CIN	I2C bus : long distance I2C data input	LVTTL	up	I	**JA** 25
SCL_ I2CIN	I2C bus : long distance I2C data input	LVTTL	up	I	**JA** 27
RESET_REG*	Reset output. This output does not need any clock to be triggered	LVTTL		O	**JA** 44
USER_INTER	User interrupt. Generates an interrupt towards the SPECS bus	LVTTL	up	I	**JA** 46
MASTERSLAVE	Selection of mezzanine mode master = 0/slave = 1	LVTTL		I	**JA 48**
SPARE{2}	FPGA spare pin	LVTTL		I/O	**JA** 52
SPARE_con{12,11,10,9}	Spare connector pin.	LVTTL		I/O	**JA** 23, 19, 17, 15
SPECS_CLOCKIN	Clock from motherboard	LVTTL		I	**JA** 26
SPECS_CLOCKOUT	40 MHz mezzanine oscillator. This clock can be disabled through SPECS control register	LVTTL		O	**JA** 22

Note: User in the pull-up imply to user to integrate resistor on his mother board

Table 2 : JA Connector

PIN NAME	PIN DESCRIPTION	Levels	Pullup / down	Dir	pin number
TRST_SPECS	For future application	LVTTL		O	**JA** 42
TCK_SPECS	JTAG Bus : JTAG Clock	LVTTL	down	O	**JA** 32
TDI _SPECS	JTAG Bus : data output from JTAG master	LVTTL	up	O	**JA** 34
TDO _SPECS	JTAG Bus : data input from JTAG slave	LVTTL	up	I	**JA** 36
TMS _SPECS	JTAG Bus : JTAG command line	LVTTL	up	O	**JA** 38
CHAN_B [7:0]	Channel_B[7:0] from TTCrx	LVTTL		I	**JA** 67, 66, 65, 64, 63, 62, 57, 56
CHAN_B [8]	Channel_A from TTCrx → L0	LVTTL		I	**JA** 68
L1_EVENT-ID RST	from TTCrx	LVTTL		O	**JA** 78
L0_FRONT-END RST	from TTCrx	LVTTL		O	**JA** 76
L1_FRONT-END RST	from TTCrx	LVTTL		O	**JA** 69
B_CALIB[0]	from TTCrx	LVTTL		O	**JA**,72
BUS_R*	Parrallel bus : Read	LVTTL		O	**JA** 83
BUS_W*	Parrallel bus : Write	LVTTL		O	**JA** 84

BUS_DATA[15:0]	Parrallel bus : Data	LVTTL		I/O	**JA** 115, 116, 113, 114, 111, 112, 107, 110, 105, 106, 103, 104, 99, 102, 97, 98
BUS_ADR[7:0]	Parrallel bus : Address	LVTTL		O	**JA** 95, 96, 93, 94, 87, 92, 85, 86
DT_RDY	Parrallel bus : Data ready (for slow components future application)	LVTTL		I	**JA** 82
DT_ACK	Parrallel bus : Data acknowledge (for slow components future application)	LVTTL		O	**JA** 79
VPOS_ANA	5V power supply (for Mezzanine optical version)			I	**JA** 5,9,8,13,14,16
POWER_MEZZANINE	3.3V power supply			I	**J3** 3, 4, 29, 28, 59, 58, 89, 88, 117, 118
GND	Reference gnd			I	**JA** 1, 2, 11, 10, 21, 20, 31, 30, 41, 40, 51, 50, 61, 60, 71, 70, 81, 80, 91, 90, 101, 100, 109, 108, 119, 120, 7, 6, 12, 18, 24, 54

Table 3 : JA Connector

PIN NAME	PIN DESCRIPTION	Level	Dir	Pullup: down	Pin number
REG_EXT[31:0]	User defined control/status register. Every I/O can be configured as input or output through a SPECS control register. This register does not need any clock to be written	LVTTL	I/O		**JB** 116, 115, 114, 113, 112, 109, 108, 107, 106, 105, 104, 103, 102, 99, 98, 97, 96, 95, 94, 93, 92, 87, 86, 85, 84, 83, 82, 79, 78, 77, 76, 75
I2CJTAG_DE[15:0]	I2C & JTAG bus : controls the direction for the external bus drivers	LVTTL	O		**JB** 74, 72, 68, 66, 64, 62, 56, 54, 52, 48, 46, 44, 42, 38, 36, 34
I2CJTAG_RE[15:0]	I2C & JTAG bus : controls the selection for the external bus drivers	LVTTL	O		**JB** 73, 69, 67, 65, 63, 57, 55, 53, 49, 47, 45, 43, 39, 37, 35, 33
I2C_ADDRESS[6:3]	I2C Address for on-board ADC	LCMOS2.5 V	I	puldown	**JB** 18, 16, 14, 12
SPARE{3,2,1,0}	Spare connector pin.	LVTTL	I/O		**JB** 26, 32, 23, 25
InAn[5:0]	Analog input channel (max dynamic range 1.75v)	Analog	I	10 Ohm Series res	**JB** 6, 8, 5, 7, 9, 13
iout10		Analog			**JB** 17
External resistor	Input for thermal sensor	Analog	I		**Do not connect**
POWER_MEZZANINE	3.3V Power supply	3.3V	I		**JB** 3, 4, 29, 28, 59, 58, 89,88,117,118

GND	Reference gnd	0V	I		**JB** 1, 2, 11, 10, 21, 20, 31, 30, 41, 40, 51, 50, 61, 60, 71, 70, 81, 80, 91, 90, 101, 100, 111, 110, 119, 120 15, 19, 27, 24

Table 4 : JB Connector

Mother-board JA& JB connector reference: HIROSE FX8-120S-SV (stacking height 3.2mm)

Web documentation on www.hirose.com

7 Software

Software libraries are available to use the SPECS System. One low level library allows to access the master registers and to prepare and send SPECS frames to communicate with the electronics using the different modes available (I2C, JTAG, DMA Bus, Parallel Bus). One high level library interfaces all commonly used functionalities to write or read data directly to the electronics. The libraries are distributed both for the Windows and the Linux platforms.

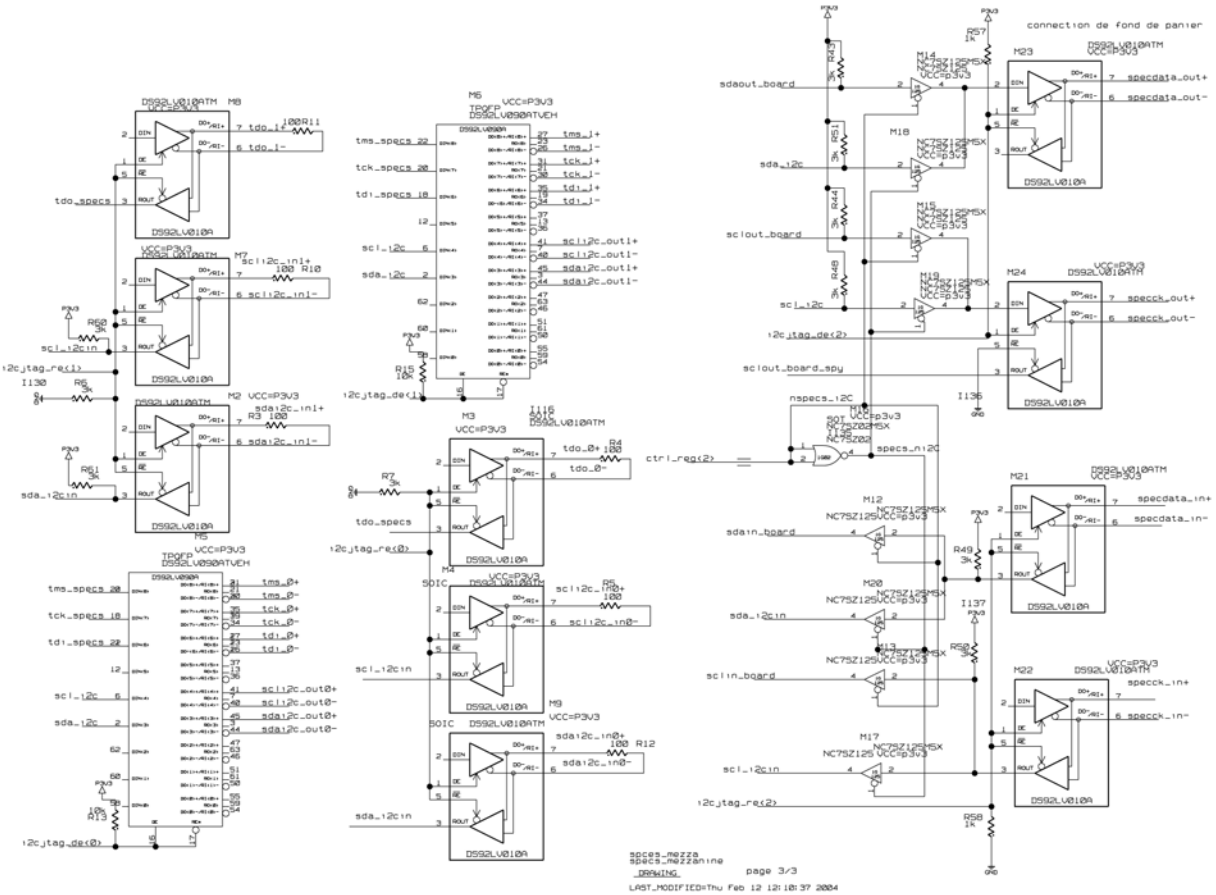
8 Conclusion

In this document, we have presented a detailed description of the SPECS protocol. The technical implementations have also been described.

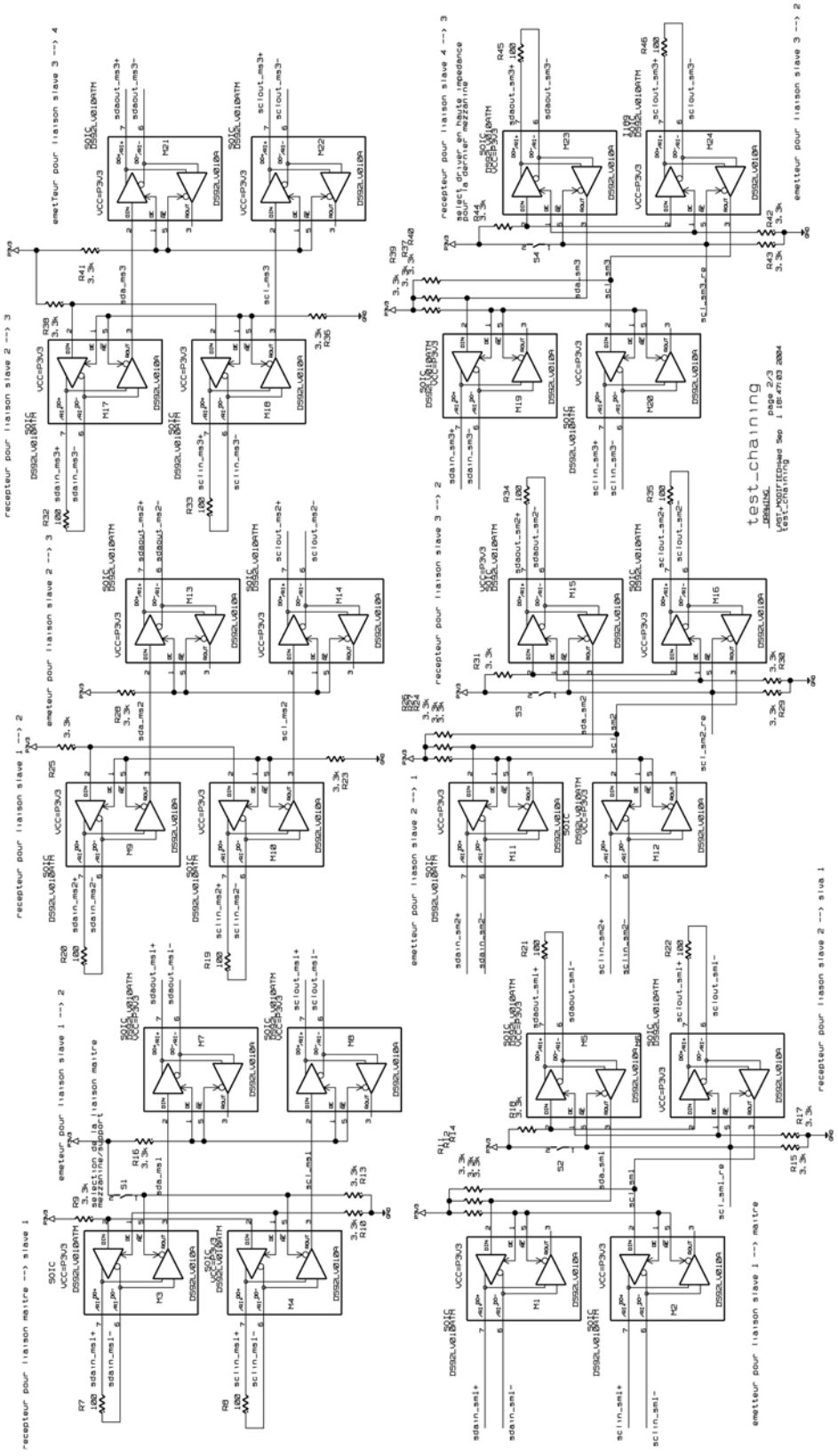
The SPECS protocol and its implementations have been designed in order to meet the requirements of the LHCb front-end electronics. It provides a simple, cheap, powerful and flexible communication link, which can be used in a large field of applications.

9 Annexe

9.1 I2C and JTAG driver cabling scheme

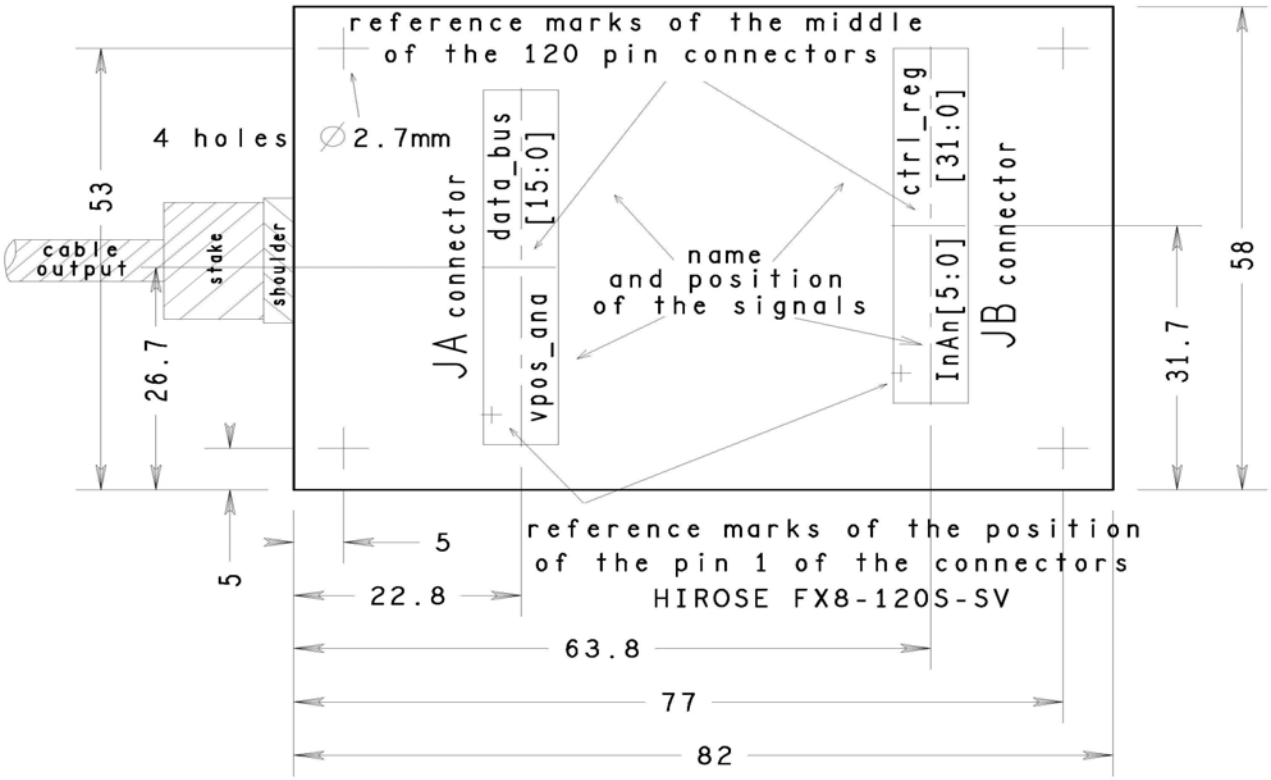


9.2 Long distance chaining cabling



9.3 Physical Dimensions

Drawing of the outline and of the female connectors on the motherboard



10 References

[1] Beigbeder, C; Breton, D; Charlet, D; Lefrançois, J; Machefert, F P; Tocut, V; Truong, K D "LHCb calorimeter front-end electronics radiation dose and single event effects" LHCb-2002-021

[2] Frédéric Machefert "Ganil Irradiation in 2003" LHCb week May 2003, Calorimeter meeting: http://agenda.cern.ch/askArchive.php?base=agenda&categ=a031004&id=a031004s3t1%2Ftransparencies%2F210503.pdf

[3] Vincent Bobillier, Jörgen Christiansen, Raymond Frei "Grounding, Shielding and Power Distribution in LHCb" LHCb 2004-039