LHCb 2003-004
DAQ
January 20, 2003

SPECS :

the Serial Protocol for the Experiment Control System of LHCb

Version 2.0

Dominique Breton, Daniel Charlet

Laboratoire de l'Accélérateur Linéaire - Orsay

ABSTRACT

This document attempts to describe the SPECS bus. It's a 10Mbit/s serial bus, designed to be simple, cheap and reliable. It's 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

This document presents the SPECS protocol, a 10 Mbit/s serial link defined to be suited for the general 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 most electronics systems (*see fig 1*). It is also designed to be efficiently protected against errors, and to be flexible.

This document presents the necessity and requirements for a configuration bus in LHCb, and the architecture of the SPECS system. Then the protocol is presented in detail. Finally, the technical implementations are described in the third part. This last part may be the most important one for the user, since it explains how to communicate with the available SPECS interfaces.

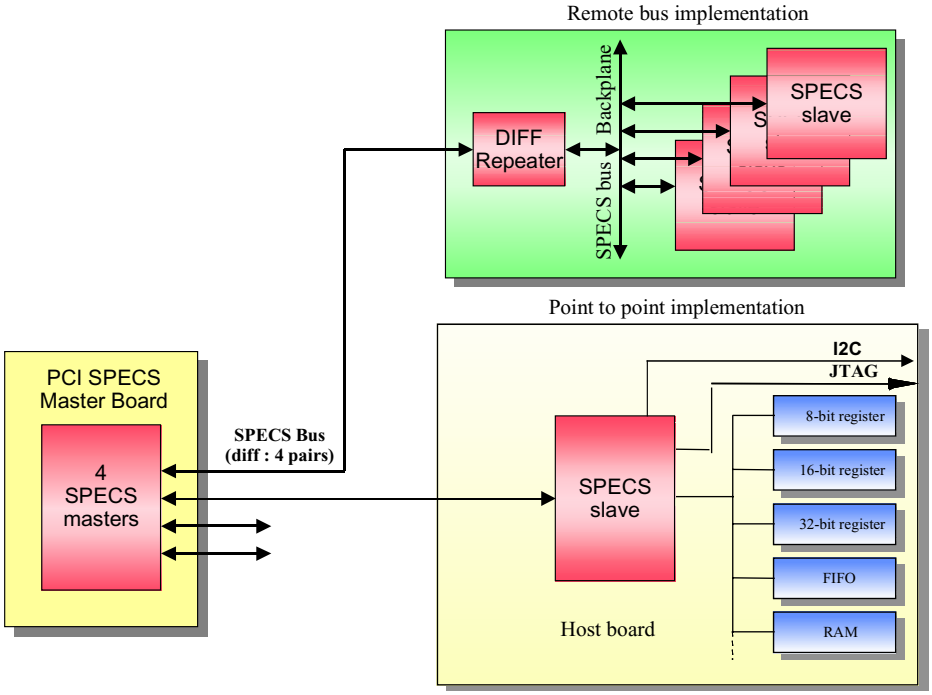

Figure 1

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 about 20 slaves (i.e. one crate). A crate can then be configured by only one bus.

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

In order to minimize the cost, the volume of cables, and also the power consumption of the bus, it is wise 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 parallel, 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 calorimeter or the outer-tracker, and much higher for the 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.

2.3 Data rate

The volume of data to be transferred depends of course on the choices of the front-end board's 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, let's say, 1 minute. Therefore, the effective, or mean data rate has to be close to 1 MB/s, which is actually the bandwidth of the SPECS.

This estimation is rather pessimistic. It probably over-estimates the amount of data to be transferred. It doesn't consider either the gain of time expected by broadcast accesses, as many of the boards to configure will be identical.

2.4 Safety of the information

Along a 100-meter cable, at a 10 Mbit/s rate, in a noisy environment, errors may occur during the transfer of data. The protection against transfer errors is necessary to ensure proper behavior.

This protection can consist in simply detecting errors, or in 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 mention an error to the master 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. In this case, the message can be interpreted by the wrong slave, sent to the wrong memory, the wrong way. 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.

Moreover, the interrupt order is also available for the user to send a read request to 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 interruptions, and to react thereto properly.

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 isn't too large. The protocol would require 2 fibers otherwise, one for each direction.

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 be used too.

The BLVDS has excellent qualities for such an implementation. Sent on twisted pairs cables, it can propagate fast signals (more than 150 MHz) along relatively long distances thanks to its 10mA 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. Those 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 (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. The issue is an EMC question : can the electronics systems be connected to the same ground via the SPECS bus ? Each configuration probably needs to be studied independently. Galvanic isolation elements might thus be necessary.

3 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 240 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, respectively from master to slave, and from slave to master. They can be implemented in differential mode (8 wires) or in unipolar mode (4 wires).

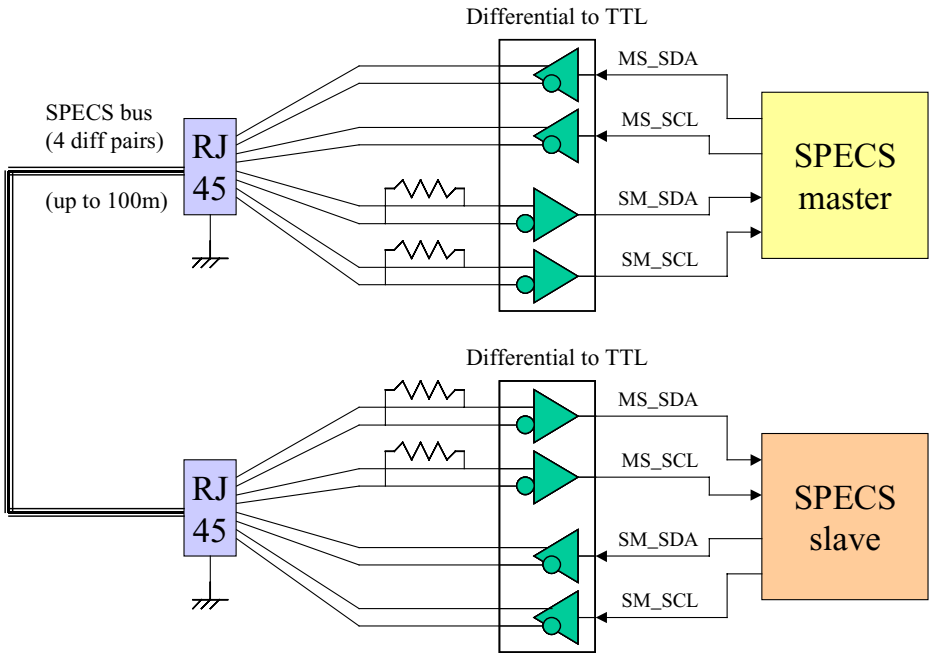

Figure 2a

The SPECS can be implemented in mainly two different ways :

- Point to point connection (Fig. 2a)
- Remote bus implementation (Fig. 2b)

The first one allows the user to connect a master to a single slave. It's 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 smaller, it might be better to use the remote bus implementation. The latter offers the possibility to drive about 20 slaves at a long distance with a single master. It implies however the use of a remote differential repeater to ensure the signal integrity. That's the solution which was chosen for the calorimeter electronics of LHCb, through distribution of the bus on the crate backplanes.

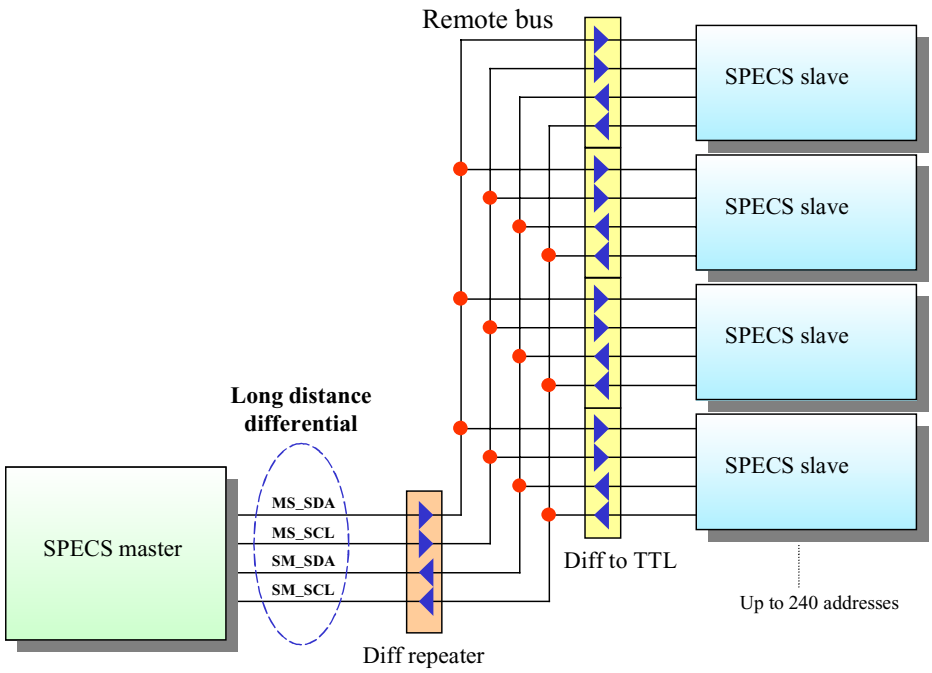

Figure 2b

3.2 SPECS frames

Data transfer is organized in frames of variable length, of which the format is the same for transfers from the master to the slaves or in the other direction, except for the interruption commands. These 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' (*Fig. 3*), and are actually identical to the I2C bus' ones.

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, for 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 MByte/s.

The standard data transfer frames are shown in fig 3a, while the format of the interruption is shown in fig 3b.

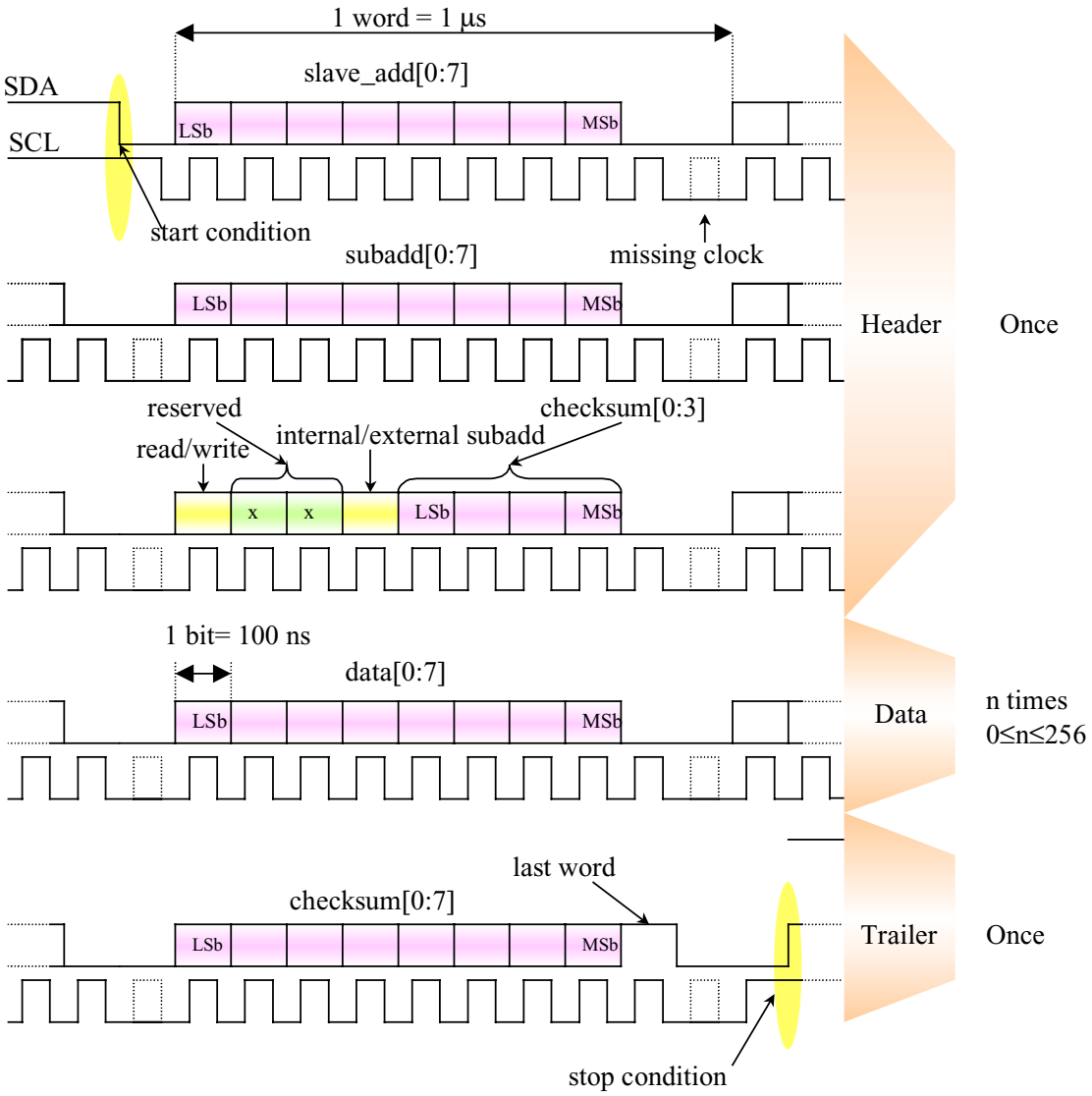

Figure 3a

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 9th bit (bit 2^8) tags the last word of the frame, and is set to zero otherwise. The 8 remaining bits are used to transfer different types of data.

The frame is composed of a fixed size header, of a variable size data block, and of a fixed size trailer. The header contains 3 words : the 8-bit address of the slave, the 8-bit sub-address of the element to access, and a control word, mentioning the read/write bit, the internal/external sub-address bit, and the 4-bit header checksum. These bits are described more accurately in *Fig 4*, *Fig 5,* and *Fig 6*.

As shown on fig 3b, the format of the interrupt is a little different. It indeed has to be as short as possible to reduce the probability of conflict and anyway to bring back the minimum information to the master. Thus it consists in a standard frame but including only the address of the sender, thus one single word.

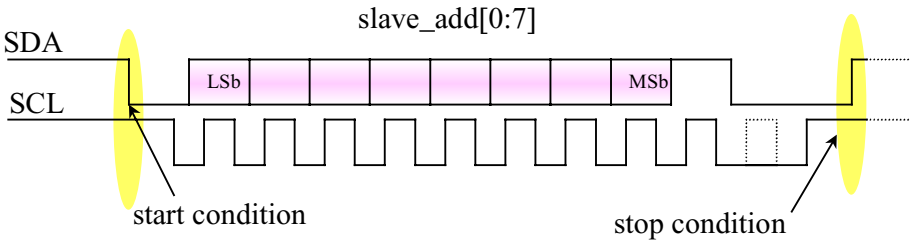

Figure 3b

After having received an interrupt order, the master will always go read the status register of the sender to learn about its origin. If the address doesn't correspond to any known slave, or after a broadcast access, they will all be scanned. We cannot indeed 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 thus to know that something happened on the other side of the line.

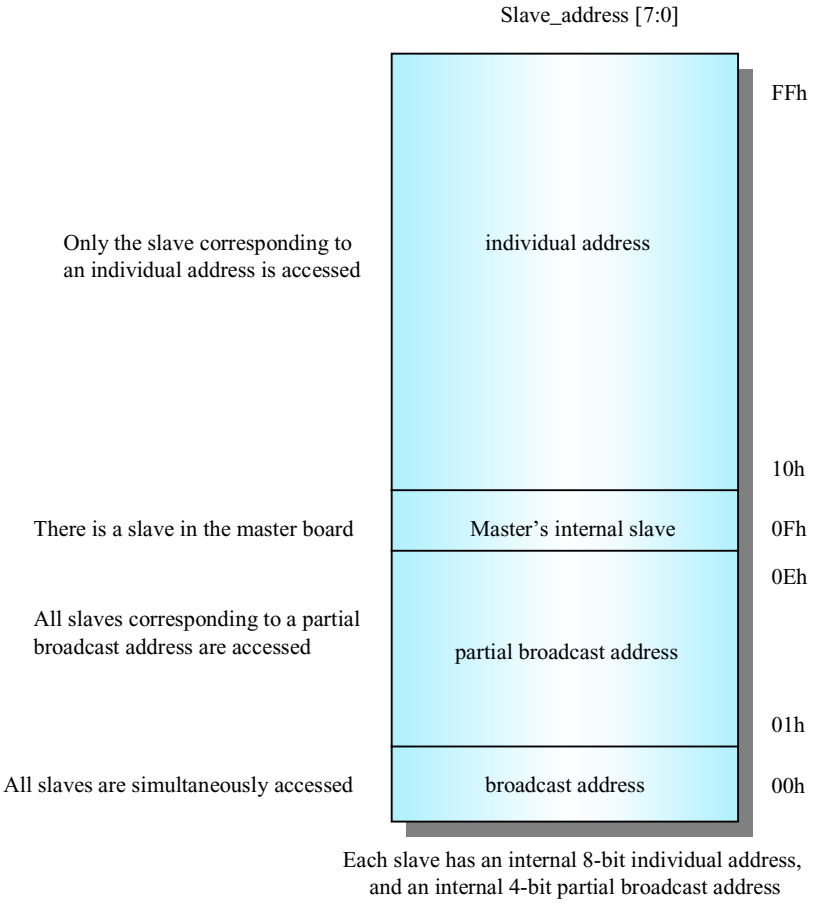

Figure 4

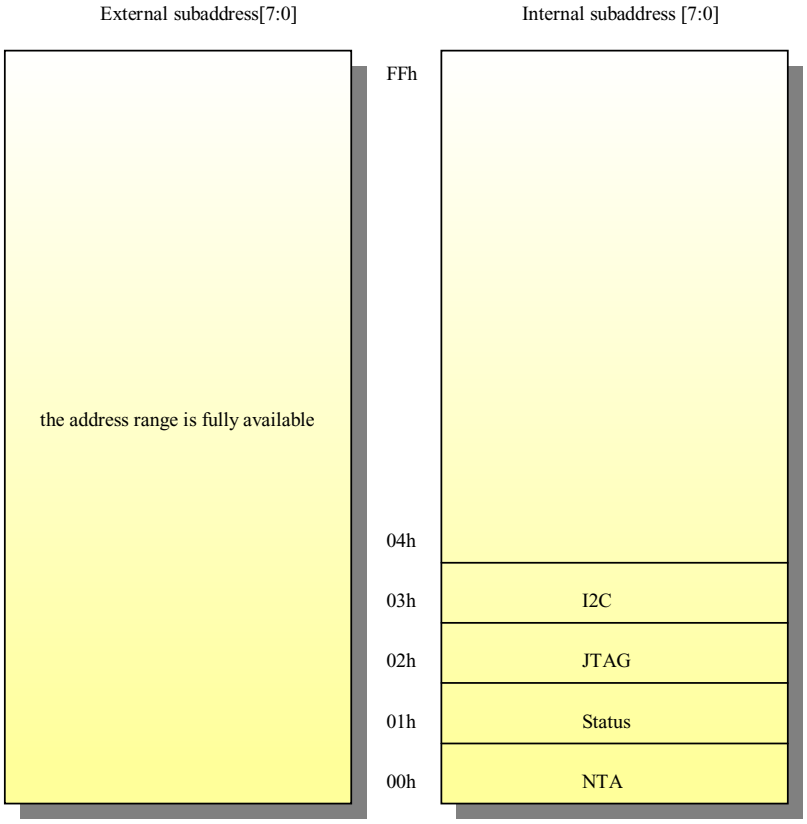

Figure 5

read/write	1 : read request, 0 : write request
internal/external subaddress	1 : internal subaddress, 0 : external subaddress
header checksum[3:0]	head_checksum[3:0] = Xor (word1[3:0], word1[7:4], word2[3:0], word2[7:4], word3[3:0], word3[7:4])

Figure 6

The data block contains a succession of 0 to 256 words, according to the transfer to realize. A more precise description follows in the next section.

Then the trailer contains the checksum of the data block. The calculation of the checksum is :

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

Being the last word, *trailer_checksum*[8] = 1, which corresponds to the last word flag.

3.3 Communications

3.3.1 Arbitration

Various kinds of accesses are possible. The two possible operations for the master are the write and read accesses.

A slave normally doesn't 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 doesn't reply anything to the master. No answer indeed means that the transfer was successful.

After a read command, it takes control of the bus to provide the requested data. Its answer goes to the bus after a known delay (the various delays are described in the state machines below (*Fig 7, 8 and 9*)).

However, in case of a suspicious transfer, the slave sends an interrupt message back to the master, after a read access as well as after a write access. The interrupt is sent after a known delay after the master's command.

For the JTAG and I2C interfaces, a special functionality, the asynchronous read sending, allows the slave to send a frame to the master after a certain delay, which is determined by the secondary bus transfer, and limited to a known number of cycles.

The slave can also take control of the bus without any occurrence of a command from the master. This case occurs 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.

In any case, before the master or a slave can take the control of the bus, they check that no transfer on the SM (slave to master) line is being performed. If any activity is detected, the action is delayed until the end of the current transfer. Then, when the SM bus becomes idle, the slaves wait for 1 or 2 clock cycles, and the master waits for 3 clock cycles before they can take control of the bus. This gives the right of way to the slaves, which may have to reply something before the master sends another command (*Fig 7 and 9*). Moreover, the waiting time for the interrupt order is shorter than for the other commands, which gives it priority.

Finally, to prevent any device on the bus from remaining in an abnormal state after a wrong transfer, the master and the slaves have 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 (master and slaves) (*Fig 8*).

Master's emission state machine

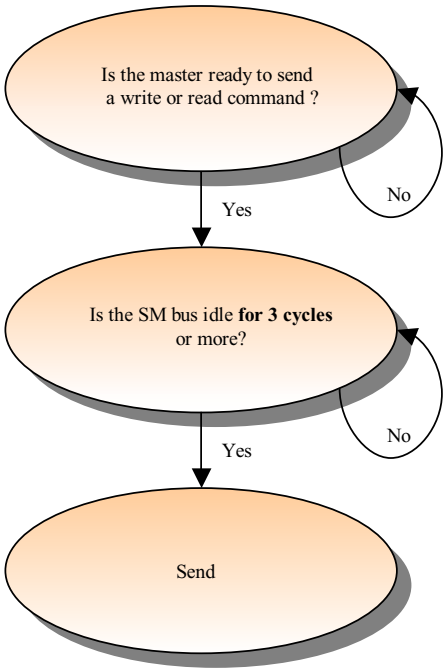

Figure 7

Master and slave timeout

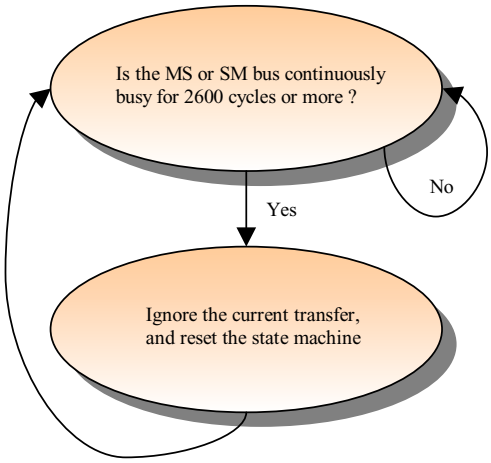

Figure 8

Slave's emission state machine

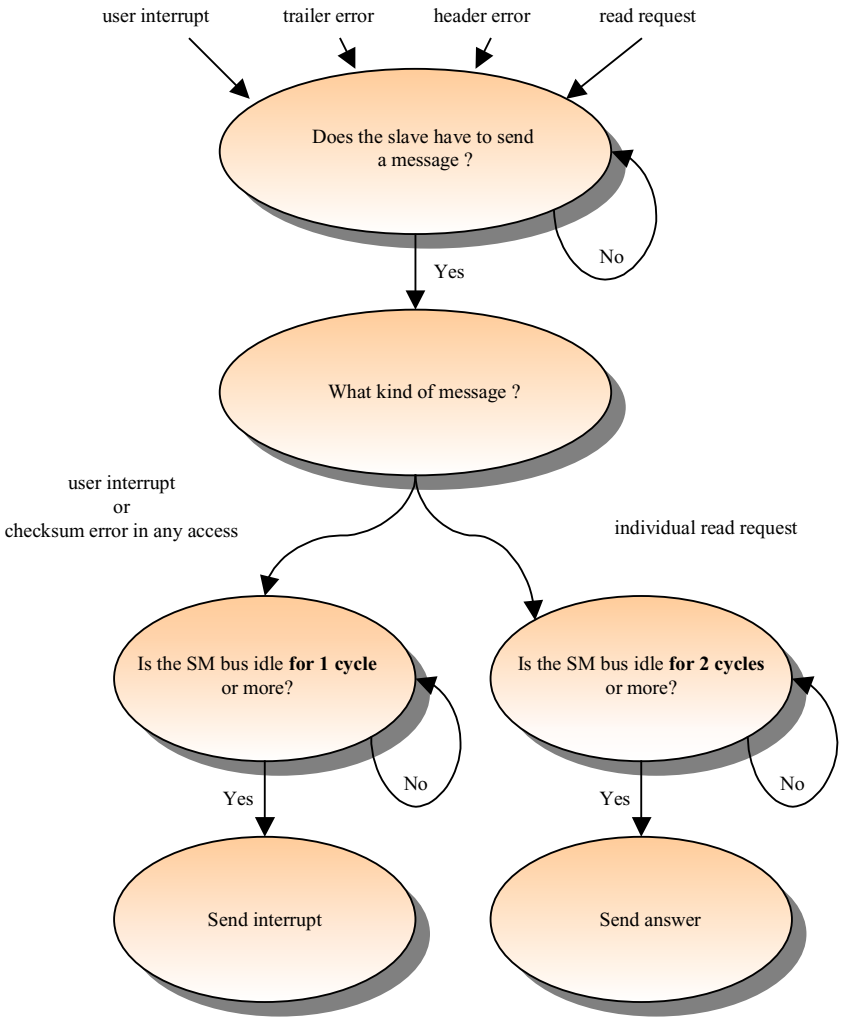

Figure 9

3.3.2 Write access

Below is shown the frame corresponding to a write access (*Fig 10*). Those 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.

n-byte write command (master)

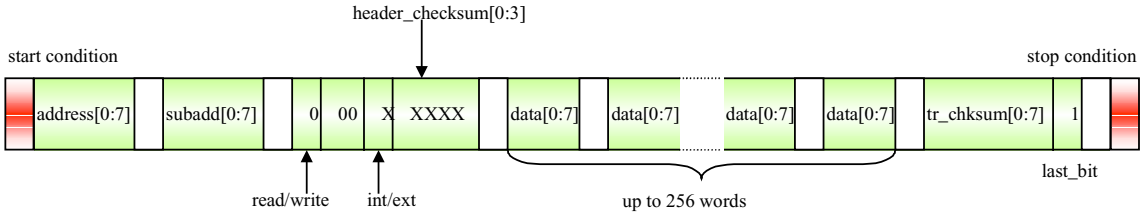

Figure 10

This frame doesn't expect an answer from the slave, except in case of a header or trailer checksum error.

3.3.3 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 back (*Fig 11*). This frame contains this number which is stored into the 8-bit internal slave register called 'wordcount[7:0]'. This register is a special feature of the slave, because it has no address and is not readable. It can only be written by the master, and this is done if 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", then 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.

Standard read command (master)

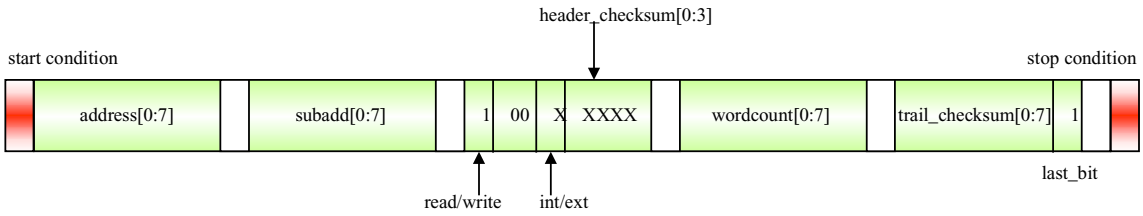

Figure 11

After a read request, the slave answers a frame with the same header as the one from the frame sent by the master, followed by the requested data (*Fig 12*). So, the master knows which slave has sent the frame, and what sub-address was read, so it can crosscheck the both.

Standard answer after a read command (slave)

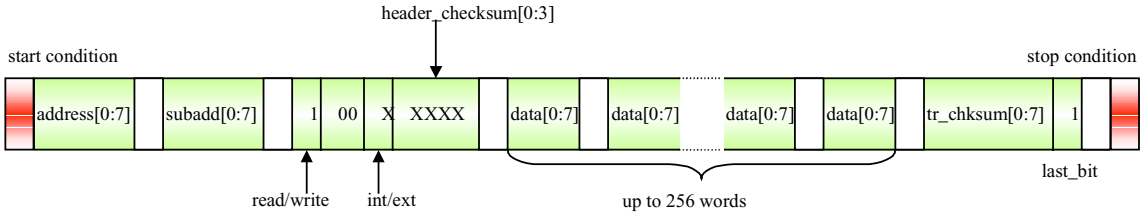

Figure 12

3.3.4 Interrupt

Whether it flags a header checksum error, a trailer checksum error, or an user interrupt, the interrupt frame has the following format (*Fig 13*) : it's simple and allows the master to target back directly the concerned slave.

Interrupt frame (slave)

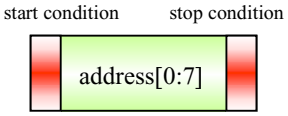

Figure 13

4 Technical implementations

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 SPECS master board actually contains four SPECS masters (*Fig. 14a*). Of course, these masters cannot be connected together, the purpose of the board being to deliver 4 independent busses. It is a standard 32-bit 33 MHz PCI board, which can be plugged into a PC or MacIntosh. A picture of the board is shown on fig 14b.

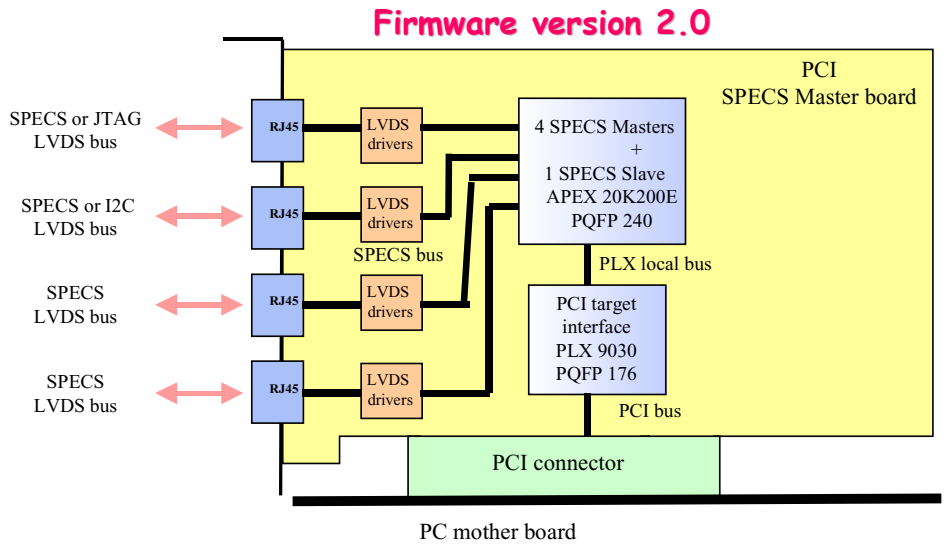

Figure 14a

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. To replace SPECS by JTAG on the first output, and/or replace SPECS by I2C on the second one, jumpers are available on the board. This operation is transparent for the software. The address of this slave is reserved : 0Fh.

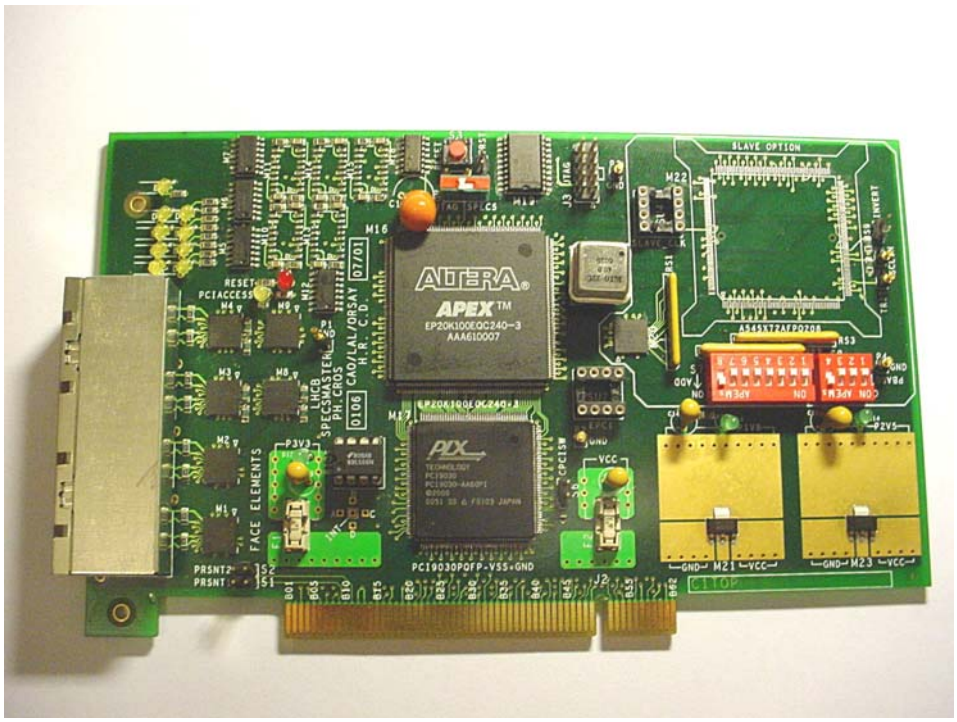

Figure 14b

As it appears on fig 14a, the heart of the system is integrated within the Altera APEX FPGA. Fig 15 displays its block diagram. The corresponding firmware is written in Verilog.

MASTER BLOCK DIAGRAM

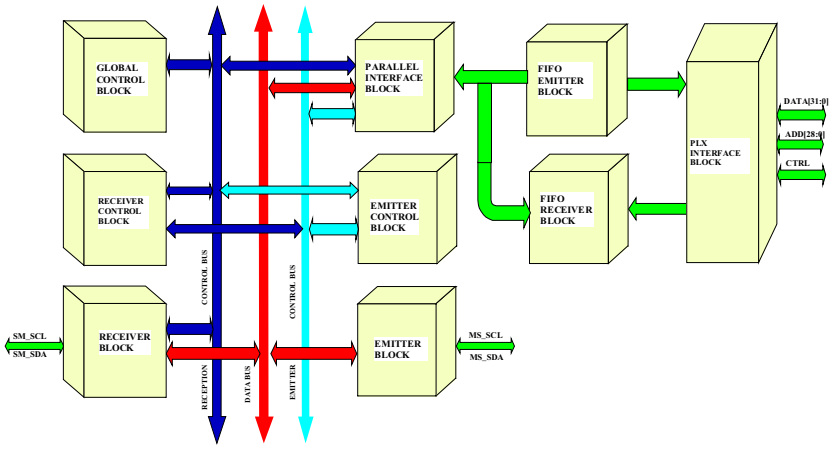

Figure 15

4.2 SPECS slave chip

The physical version of the SPECS slave is an FPGA that has to be implemented on a board. To prevent the configuration of the chip (firmware) from being altered by SEUs, the FPGA uses an anti-fuse technology. The design is studied to provide an efficient protection against SEUs, and uses redundancy and one-hot-state machines.

The chip can either be an ACTEL A54SX32A or A54SX72A, or even an AX125 (the choice will be made soon). The package is a 208-pin PQFP for the SX or a 256-pin FBGA for the AX.

The synopsis of the chip's I/Os is shown below *(Fig 16a)*. It offers 4 different interfaces :

- The SPECS slave interface, composed of the 4 unidirectional signals MS_SDA, MS_SCL, SM_SDA, SM_SCL that have to be connected to the differential LVDS drivers. The SerialOutEnable line is the enable for the SM_SDA and SM_SCL LVDS drivers.
- The local bus master interface delivers an easy to use parallel bus. This interface is composed of a bi-directional 8-bit data bus, for write and read operations. An 8-bit sub-address bus, a write* pulse and a read* enable signal are provided with the data bus. The ChipSelect* 8-bit bus is the decoding of subadd[2:0], and can replace the sub-address bus for simple architectures requiring 8 addresses or less. The NTA (Next Transfer Address) is the output of a 16-bit counter. Each read* or write* pulse increments the counter, so the NTA bus is dedicated to address RAMs, with an automatically incremented RAM address. The 4-bit ByteNumber bus delivers four positive enable

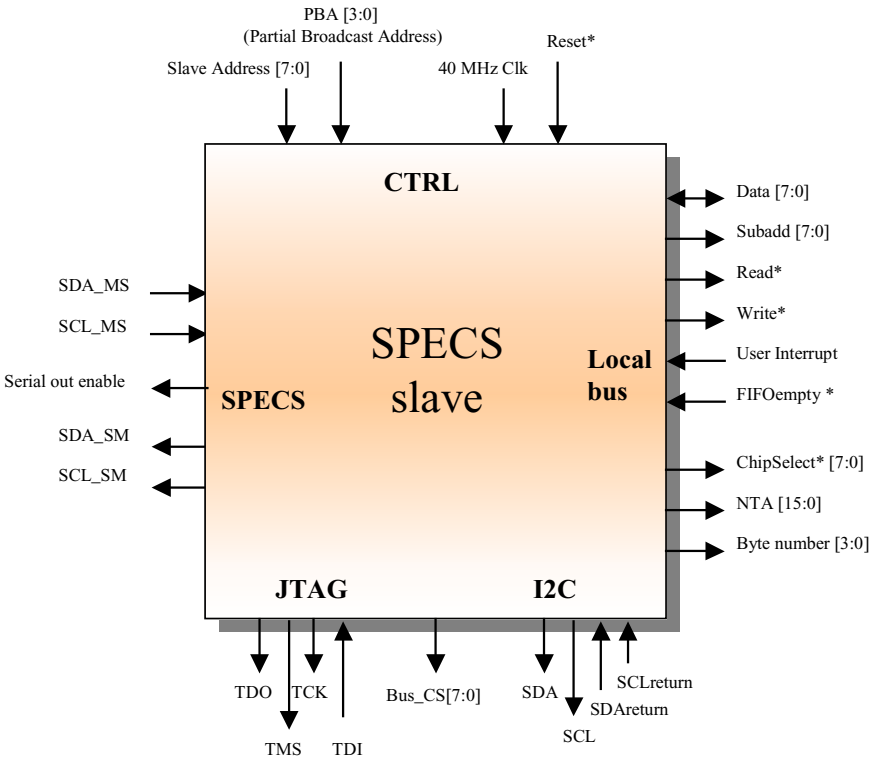

Figure 16a

SLAVE BLOCK DIAGRAM

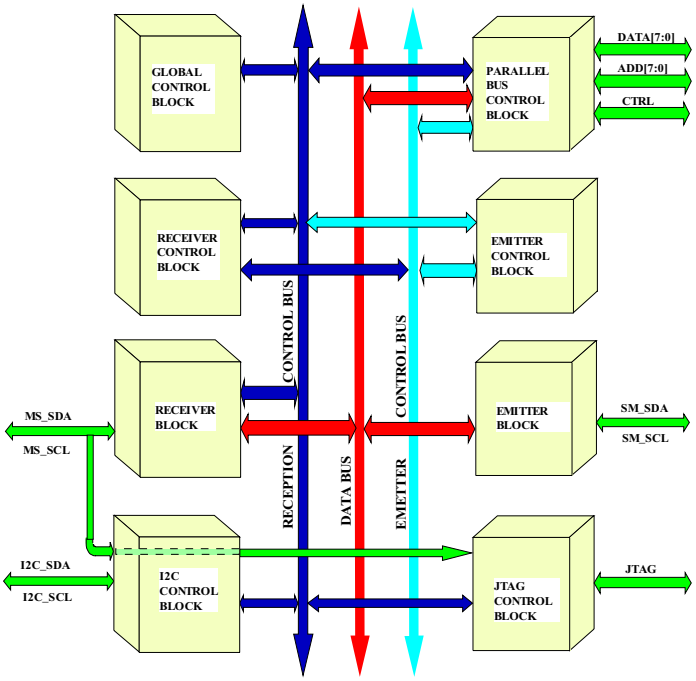

Figure 16b

signals to write into registers up to 32-bits wide, thus allowing them to be written byte by byte from an unique SPECS frame. The UserInterrupt input propagates an interrupt into the SPECS bus.

- The JTAG master interface provides the usual TDI, TMS, TCK, TDO signals. The Bus_Chipselect 8-bit bus allows the slave chip to drive up to 8 independent JTAG busses. In this case, each bus is driven via drivers from the unique JTAG bus, the 8 Bus_Chipselect signals being used to enable the drivers.
- The I2C master bus is composed of three signals, MS_SDA, SM_SDA, MS_SCL. As well as for JTAG, the Bus_Chipselect 8-bit bus allows the slave chip to drive up to 8 independent I2C links.

Fig 16b displays the internal block diagram of the slave chip, upon which the Verilog code is based.

4.3 How we make JTAG and I2C from SPECS

The goal here was 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 took benefit of the fact that the SPECS bus can carry 8Mbit/s of data per second whereas JTAG and I2C are usually limited to 1Mbit/s. This factor 8 allowed us to transfer 1 bit of data on either JTAG and I2C from 1 byte on the SPECS, while making use of all 8 bits to produce the different necessary control signals. This is described on the following figures (17a to 17d).

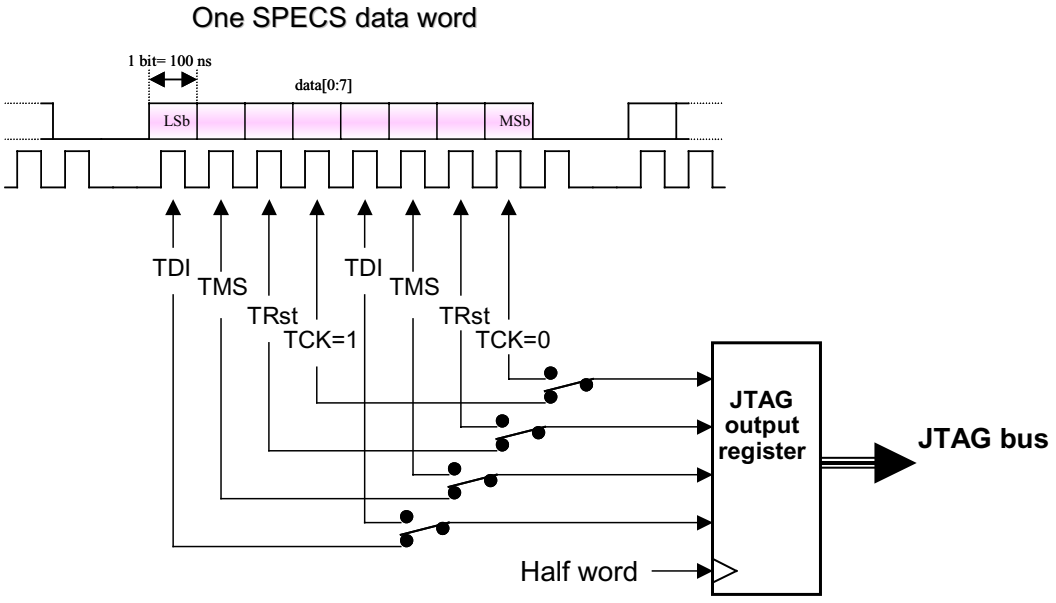

Figure 17a

In JTAG (fig 17a), 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 accumulate one after the other.

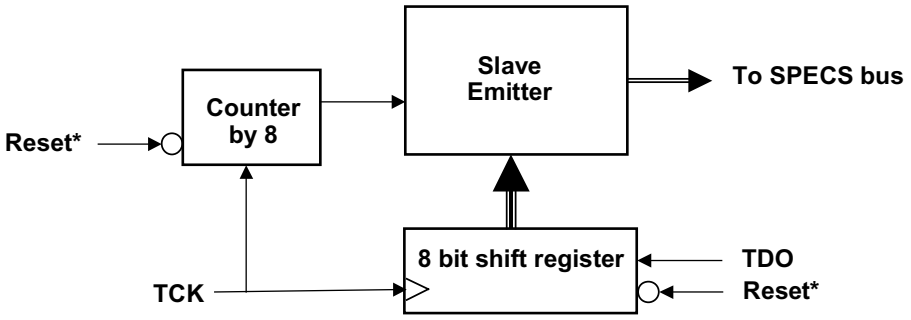

Figure 17b

The way we deal with the TDO return line is shown on fig 17b. 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.

Fig 17c 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'll need a SPECS byte for each of the start and stop conditions, and 8 for the I2C slave address. So 30 bytes of data will be available in one single SPECS transfer, but longer chains can also be transferred if 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.

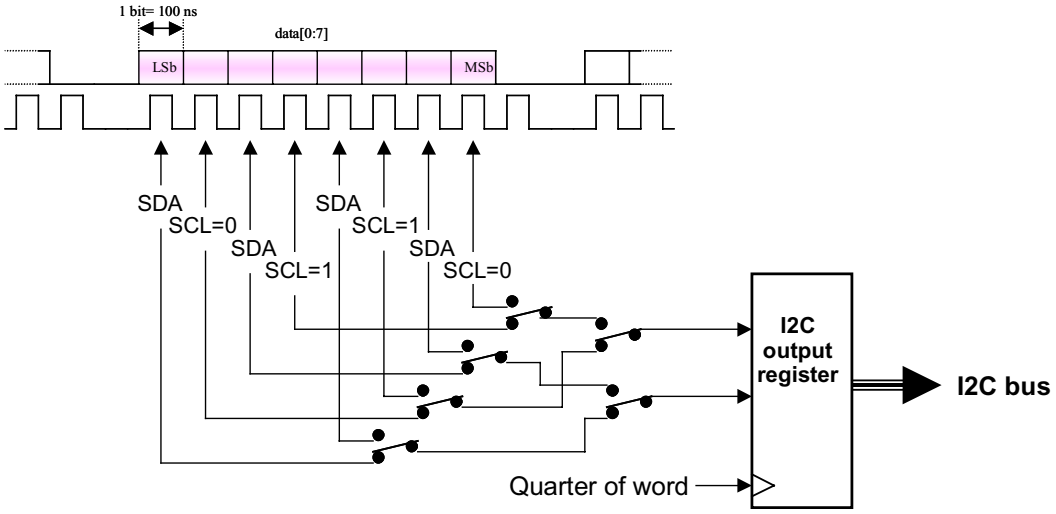

Figure 17c

The main difficulty with I2C is the acknowledge signal. In order to limit the number of transfers on the SPECS bus, we won't send it back to the master. Conversely, we'll check it at the slave level, and warn the master only if it's missing, thanks to the sending of an interrupt order. Consequently, the master will stop sending data and finish the frame with a stop condition.

Concerning the data readout through I2C, it's done almost the same way as for JTAG, as shown on fig 17d.

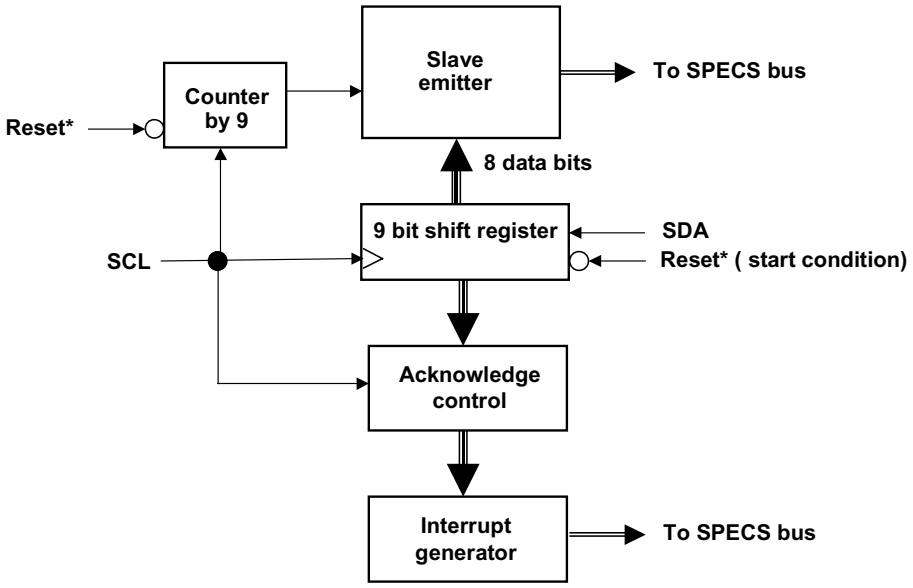

Figure 17d

All the preparation of the frames will be made by dedicated software. The user will give it the information necessary for the transfer and it will produce the frames to be stored in the SPECS master board.

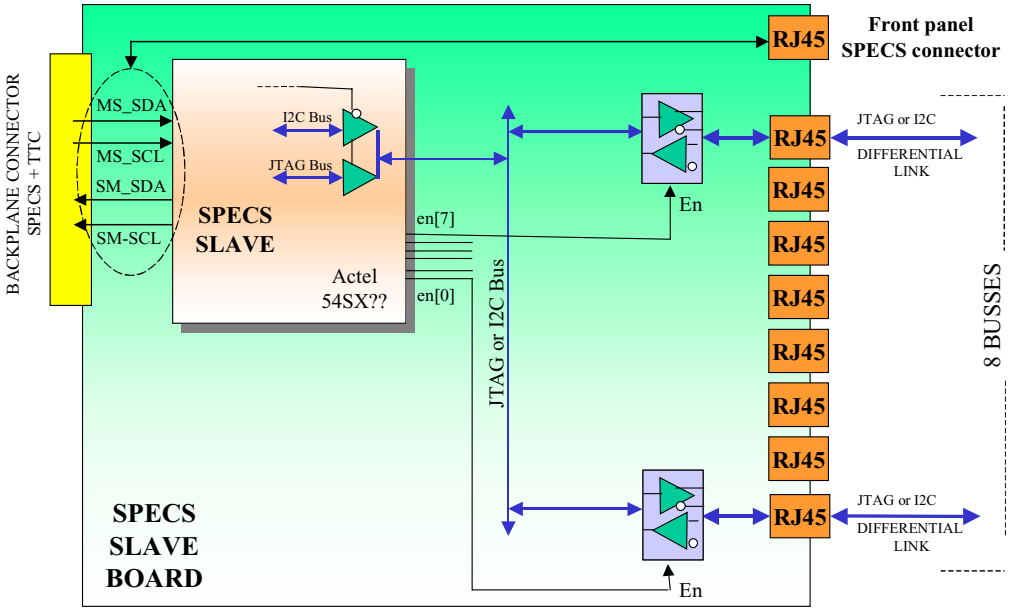

Figure 18a

To allow the users to dispose of numerous JTAG and I2C links, a VME 6U size compatible board will be developed. Its block diagram is shown on fig 18a. One SPECS slave chip will distribute 8 links to differential front panel outputs on RJ45 plugs. Every link type will be configurable by software. The SPECS access will be available either from the backplane or from the

front panel. This choice will depend on the required bandwidth. The quoted backplane is the one developed for the calorimeter crates. Fitting in VME 3U, equipped with 2mm hard metric connectors, it distributes the power supplies, and also the TTC and the SPECS from a board designed therefore. If the front panel SPECS plug is used, the board can run in standalone mode using a local oscillator. It just needs to be supplied.

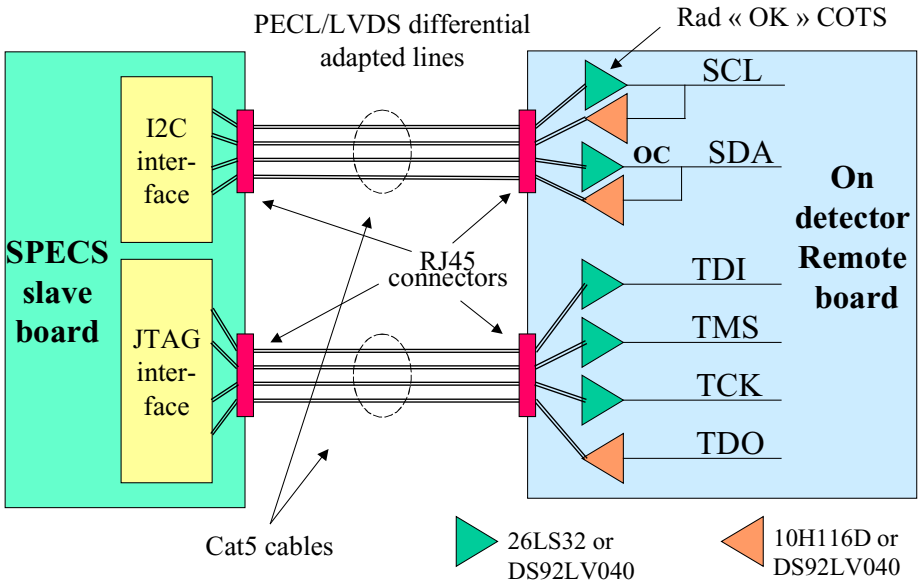

Figure 18b

Fig 18b shows the way the JTAG and I2C long distance differential links should be implemented. Please notice that both I2C signal lines have to be sent back to the SPECS slave board for a proper sampling, independent of the cable length. The chips on the remote board might have to be resistant to radiation. We'll soon test the DS92LV040 therefore. The 10H116D has already been validated by ATLAS at a 100krad level.

4.4 Interconnections

4.4.1 SPECS connector

The SPECS connector delivers 2 differential output pairs and 2 differential input pairs in LVDS levels *(Fig 19)*.

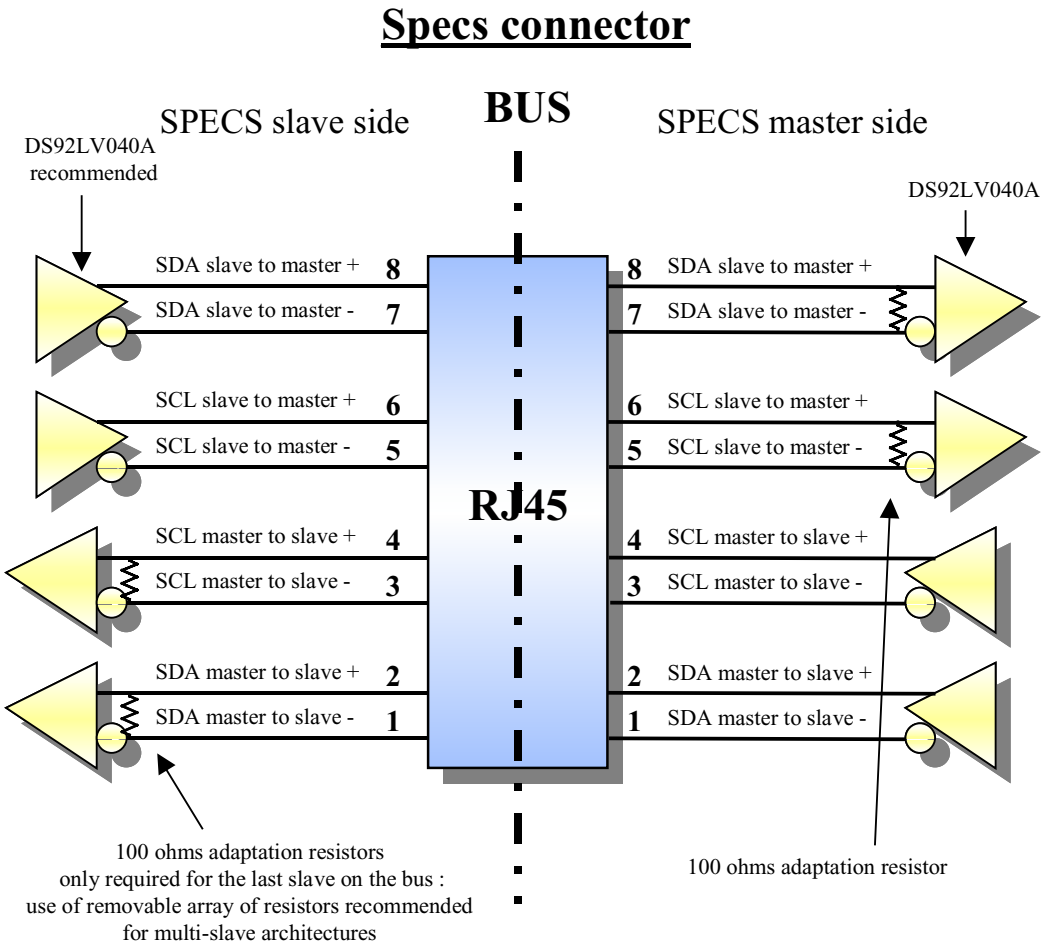

Figure 19

4.4.2 JTAG connector

The JTAG connector provides the TDI, TDO, TMS and TCK signals of the JTAG bus in differential mode (LVDS) *(Fig 20)*. For this connector, the SPECS slave is also the JTAG master, and is thus represented on the right side of *Fig 20*.

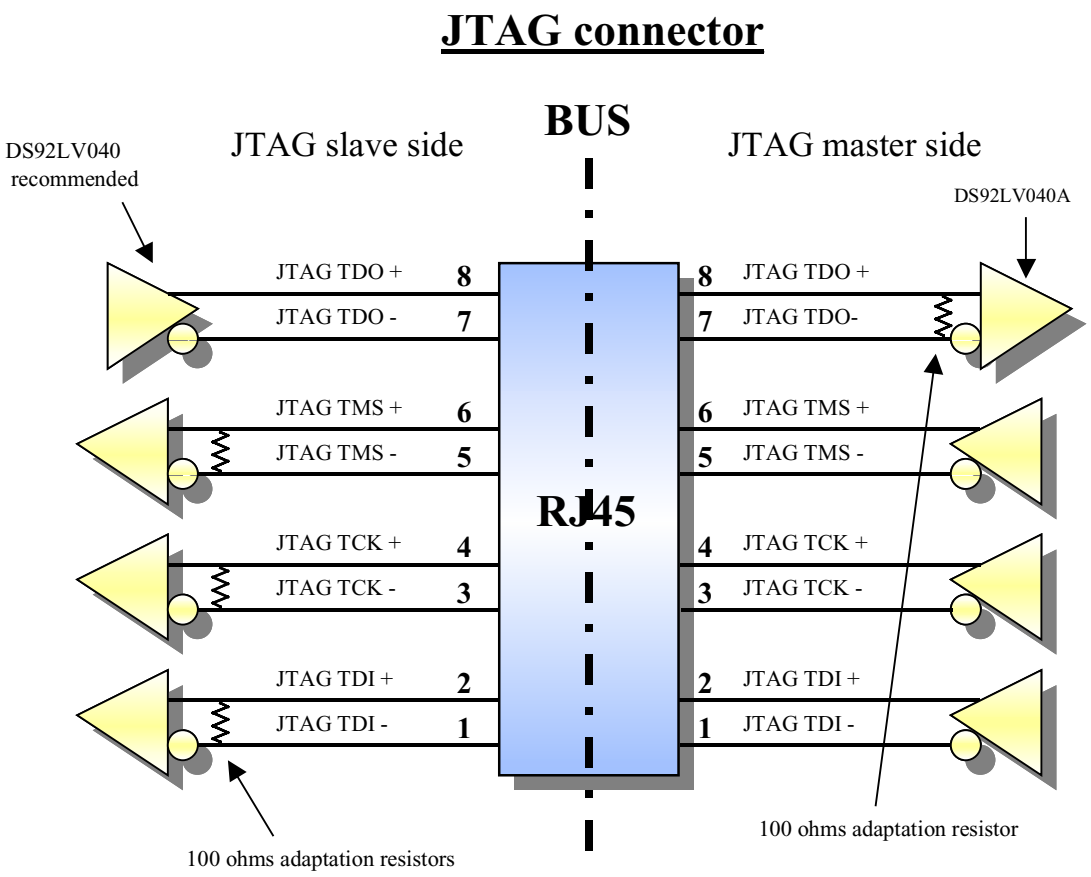

Figure 20

4.4.3 I2C connector

The I2C connector provides the SDA signals in unidirectional and differential mode. Therefore, 4 pins are necessary for the SDA line. The SCL signals are unidirectional and in differential mode. All signals are in LVDS technology *(Fig 19)*. The SPECS slave is also the I2C master, and is thus represented on the right side of *Fig 21*.

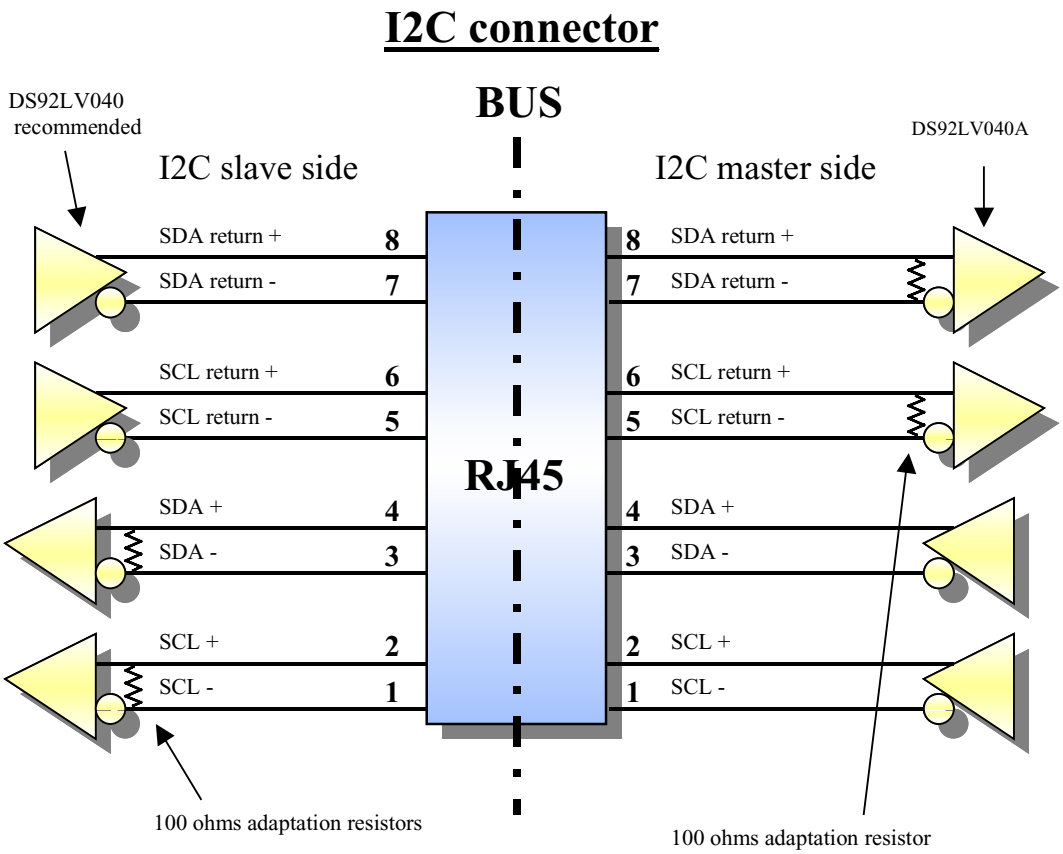

Figure 21

4.4.4 RJ45 connector

The pin numbers of the RJ45 connector (board through mount version shown) are described below *(Fig 22)*.

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).

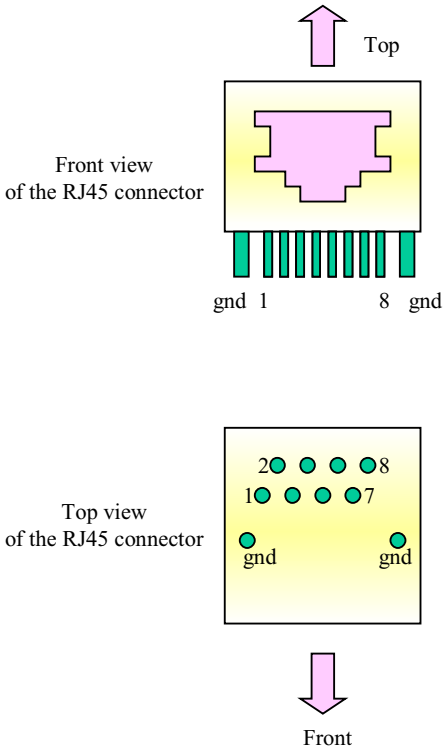

Figure 22

4.5 Software

A software kit is currently being developed in association with the hardware implementations. It will be composed of 5 different libraries organized in 3 levels. This kit should allow a very large variety of software developments, from various platforms and systems, using the C language (*Fig 23*).
The upper level library, SPECS slave, will allow the application programmer to use all the functionalities of the SPECS bus transparently, without having to know the details of the protocol.

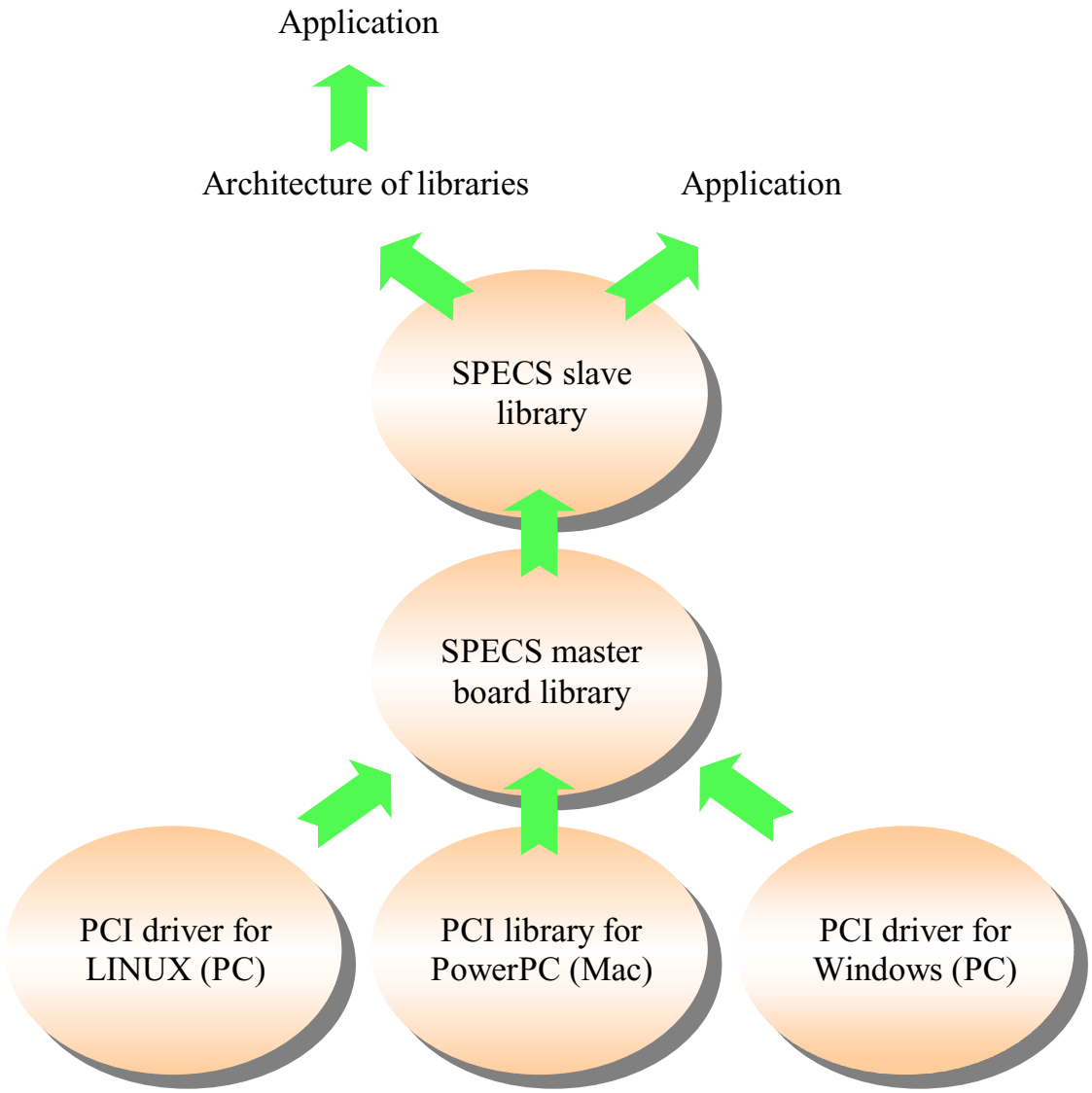

Figure 23

5 Conclusion

In this document, we presented a detailed description of the SPECS protocol. The technical implementations were also briefly 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, adaptable to a large field of applications.

Our plan is to provide the first full configuration system at the beginning of year 2003. Some of the hardware or software developments are not yet fully over. Therefore, this document may be subject to modifications. It is also likely that other hardware developments will be added to the hardware section.

1 INTRODUCTION 3

2 REQUIREMENTS AND SOLUTIONS FOR LHCB 3

2.1 GENERAL PRESENTATION 3
2.2 RADIATION ENVIRONMENT 4
2.3 DATA RATE 4
2.4 SAFETY OF THE INFORMATION 4
2.5 PHYSICAL LINK 5

3 SPECS PROTOCOL 6

3.1 GENERAL DESCRIPTION 6
3.2 SPECS FRAMES 7
3.3 COMMUNICATIONS 11
3.3.1 Arbitration 11
3.3.2 Write access 13
3.3.3 Read access 14
3.3.4 Interrupt 15

4 TECHNICAL IMPLEMENTATIONS 15

4.1 SPECS MULTI-MASTER BOARD 15
4.2 SPECS SLAVE CHIP 17
4.3 HOW ME MAKE JTAG AND I2C FROM SPECS 19
4.4 INTERCONNECTIONS 23
4.4.1 SPECS connector 23
4.4.2 JTAG connector 24
4.4.3 I2C connector 25
4.4.4 RJ45 connector 26
4.5 SOFTWARE 27

5 CONCLUSION 28