Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas

ROS-25 User Manual

Version 1.2

J. M. Cela, C. Fernández, C. Willmott.

Electrónica y Automática.
CIEMAT.

PRELIMINARY

August 11th, 2006.

INDEX

1 INTRODUCTION

In this manual it is explained the basic functionality of the Read-Out Server (ROS) board. This board performs the second level of read-out in the data acquisition chain of the muon drift tube system of CMS. Its main task is the management of digitalized data that comes from the Read-Out Boards (ROB), located inside the Minicrates, attached to the drift tube chambers. Each ROS board can handle up to 25 input channels (up to 25 ROBs), storing data and performing a data merging event by event for further optical transmission to next level of the DAQ chain, the DDU.

The ROS board is a 9U VME board, 400 mm depth with 8 RJ-45 input connectors and one optical output. Apart from transmission of received data through the optical output, the ROS has a spy memory where a programmed amount of events can be stored for further reading through a VME interface.

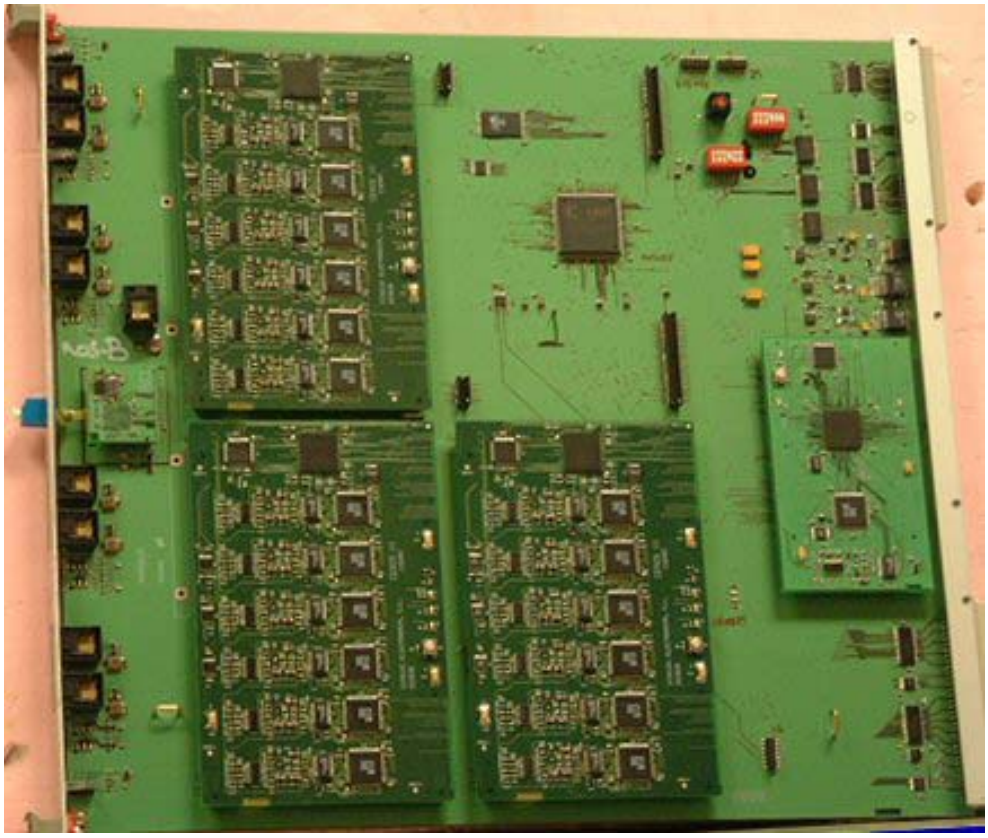

Figure 1: Picture of a Read-Out Server board.

2 ROS-25 CONNECTIONS

ROS boards will be located in two crates (TOP and BOTTOM) of the DT & RPC TRIGGER RACK (X2J22, X2J512, X2J02, X2V12, X2V22), that is, the outer rack of the lowest near balcony of each wheel. There will be 6 ROS per crate, named from ROS1 to ROS12, one per sector.

In principle, each wheel is subdivided in an upper part (sectors 1 to 6) that are connected to the ROSs of the TOP crate and a bottom part (sectors 7 to 12) connected to the ROSs of the BOTTOM crate.

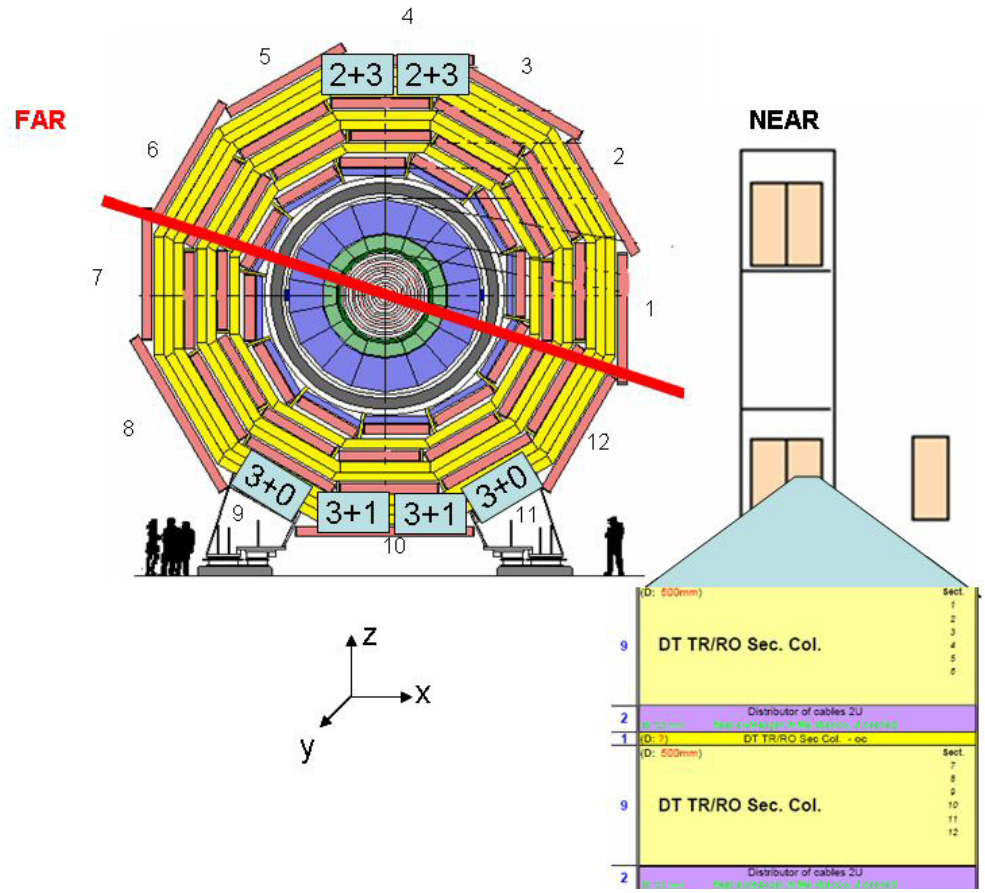

Figure 2: View of the two Sector Collectors crates for one CMS wheel.

Minicrate to Read-Out Server (ROS) links have been distributed in order to minimize the required number of connectors on the ROS while maintaining a unique ROS board format for every sector. The ROS board has been designed with 25 channels to read in average one full sector, and each ROS has 8 RJ-45 connectors with some of its pairs operational according to the following tables:

Connector	Number of links	Cable	Coming from
RJ1	3	MB1-A	MB1 (ROB 0 to 2)
RJ2	3	MB1-B	MB1 (ROB 3* to 5)
RJ3	3	MB2-A	MB2 (ROB 0 to 2)
RJ4	3	MB2-B	MB2 (ROB 3 to 5)
RJ5	4	MB3-A	MB3 (ROB 0 to 3)
RJ6	3	MB3-B	MB3 (ROB 4 to 6)
RJ7	3		Depends on the sector
RJ8	3		Depends on the sector

Note: * ROB 3 of MB1 is a ROB-32.

At sectors 1, 2, 3, 5, 6, 7, 8 and 12, connectors RJ7 and RJ8 are related to MB4(ROB 0 to 2) and MB4(ROB 3 to 5) respectively, but as sectors 4 and 10 have two chambers MB4 each, the extra channels have been accommodated in the ROS boards reading sectors 9 and 11 that have fewer channels. A patch panel is attached to the Sector Collector crate to allow interconnection. Final scheme for connectors RJ7 and RJ8 is presented in the following table:

	Connector	Number of links	Cables		Coming from
ROS for sectors (1,2,3,5,6,7 8 and 12)	RJ7	3		MB4(s)-1	3 from MB4 (ROB 0 to 2)
	RJ8	3		MB4(s)-2	3 from MB4 (ROB 3 to 5)
ROS 4	RJ7	3		MB4(4)3-2	3 from MB4(4)3 (ROB 2 to 4)
	RJ8	3		MB4(4)5-2	3 from MB4(4)5 (ROB 2 to 4)
ROS 9	RJ7	3		MB4(9)-1	3 from MB4(9) (ROB 0 to 2)
	RJ8	3	MBX	MB4(4)5-1 + MB4(10)9-2	2 from MB4(4)5 (ROB 0 and 1) + 1 from MB4(10)9 (ROB 3)
ROS 10	RJ7	3		MB4(10)11-1	3 from MB4(10)11 (ROB 0 to 2)
	RJ8	3		MB4(10)9-1	3 from MB4(10)9 (ROB 0 to 2)
ROS 11	RJ7	3		MB4(11)-1	3 from MB4(11) (ROB 0 to 2)
	RJ8	3	MBY	MB4(4)3-1 + MB4(10)11-2	2 from MB4(4)3 (ROB 0 and 1) + 1 from MB4(10)11 (ROB 3)

Here, MB4(4)5 stands for Minicrate MB4 from sector 4 that is closest to sector 5, and MB(4)3, the closest to sector 3. Idem for MB4(10)9 and MB4(10)11.

In the following diagram the front panel of the ROS-25 is shown. **Note that connectors are not in consecutive order.**

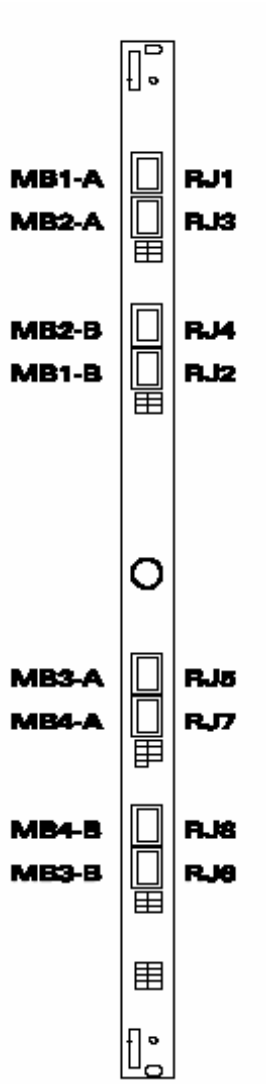

Figure 3: Scheme of the ROS-25 front panel.

3 SECTOR COLLECTOR CRATE

The preliminary layout of each crate is presented in figure 4. Inside each crate there will be: 6 ROS boards, 6 Sector Collector boards, a TIM board [5] a 3U VME controller and 3 TTC [3] boards for splitting and fan-out of the TTC optical signal to the Minicrates and also to the Sector Collector crate. And finally, the crate will also include a patch panel for grouping ROB-ROS channels and the connectors for the 5V power supply of the Sector Collector crate.

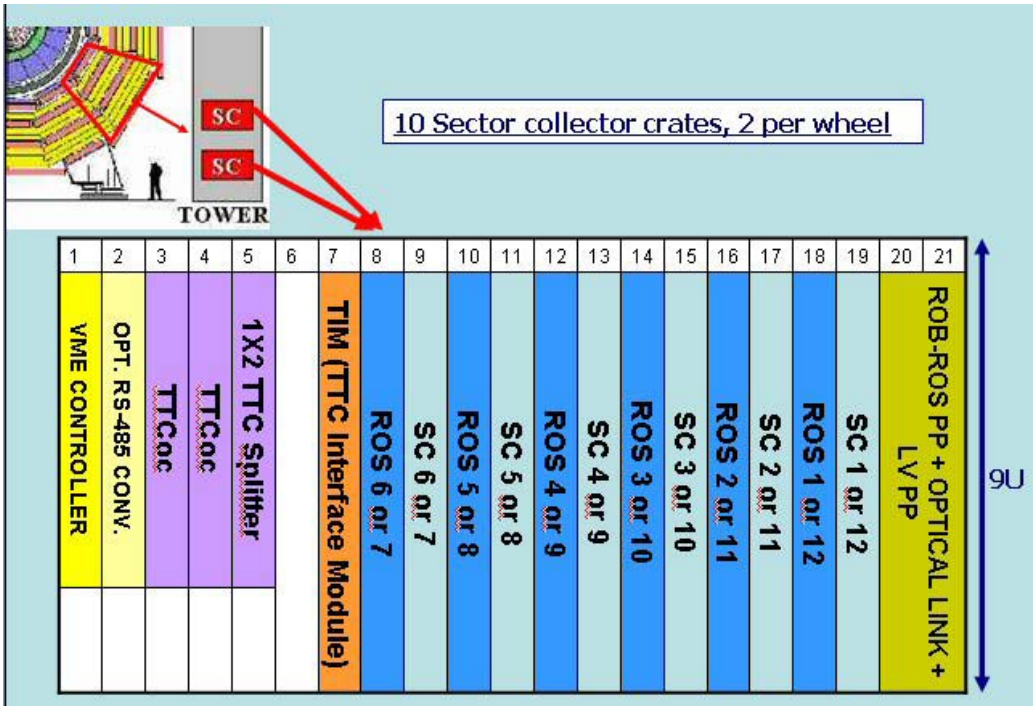

Figure 4: Diagram of the Sector Collector crate and the different boards at each slot.

The TIM module is a 9U board used for receiving the optical TTC signal and retransmitting the decoded information to the ROS and the Sector Collector boards through a 3U and 13 slots backplane located in the bottom part of the crate.

Through the TIM backplane the following TTC signals are retransmitted:

- L1A
- Bunch counter reset
- Event counter reset
- Bunch counter value at the corresponding L1A (12 bits).
- Event counter value at the corresponding L1A (24 bits).
- Other B-Go commands for the Sector Collector board.

It has to be noted that by default the ROS will read the Bunch and Event counter values from the TIM backplane and not from an internal counter. These values are decoded at the TTCrx device at each TIM board.

In order to obtain proper values of these two counters at the ROS and SC, the TTCrx has to be operated in trigger mode “11” (as it is by default), that is, there cannot be two consecutive triggers separated in less than 75 ns, or in other words, the maximum L1A frequency is 13.33MHz. The following picture clarifies this requirement, however, for more information you can have a look at the TTCrx user manual [6].

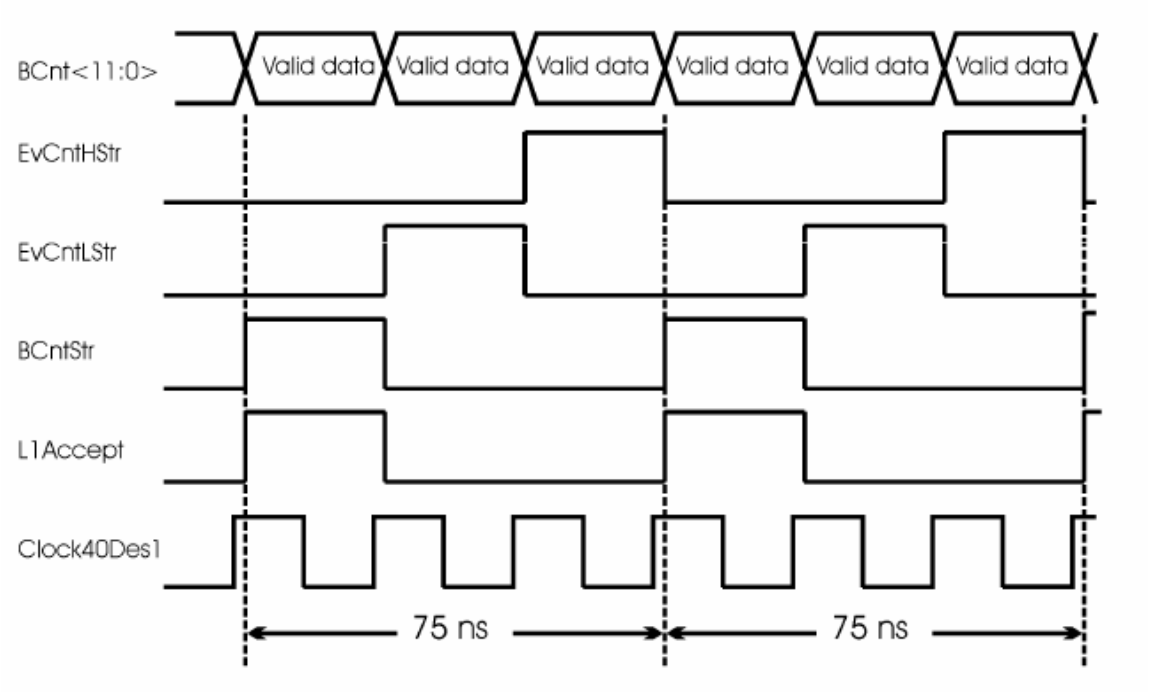

Figure 5: Trigger mode “11” at the TTCrx.

4 OVERVIEW OF THE ROS-25 DESIGN

At the ROS, the 25 channels are grouped in blocks of 6 channels, controlled by an FPGA device that performs the read-out of these channels and checks whether the channels are locked, there are parity errors, etc. Each of these groups of 6 channels is called a CEROS, and there are four of these functional blocks in a ROS as can be seen in figure 2.

There is also a fifth CEROS functional block (CEROS 4) that handles only 1 input channel, number 25th. It is identical to the other four CEROS but only for 1 channel.

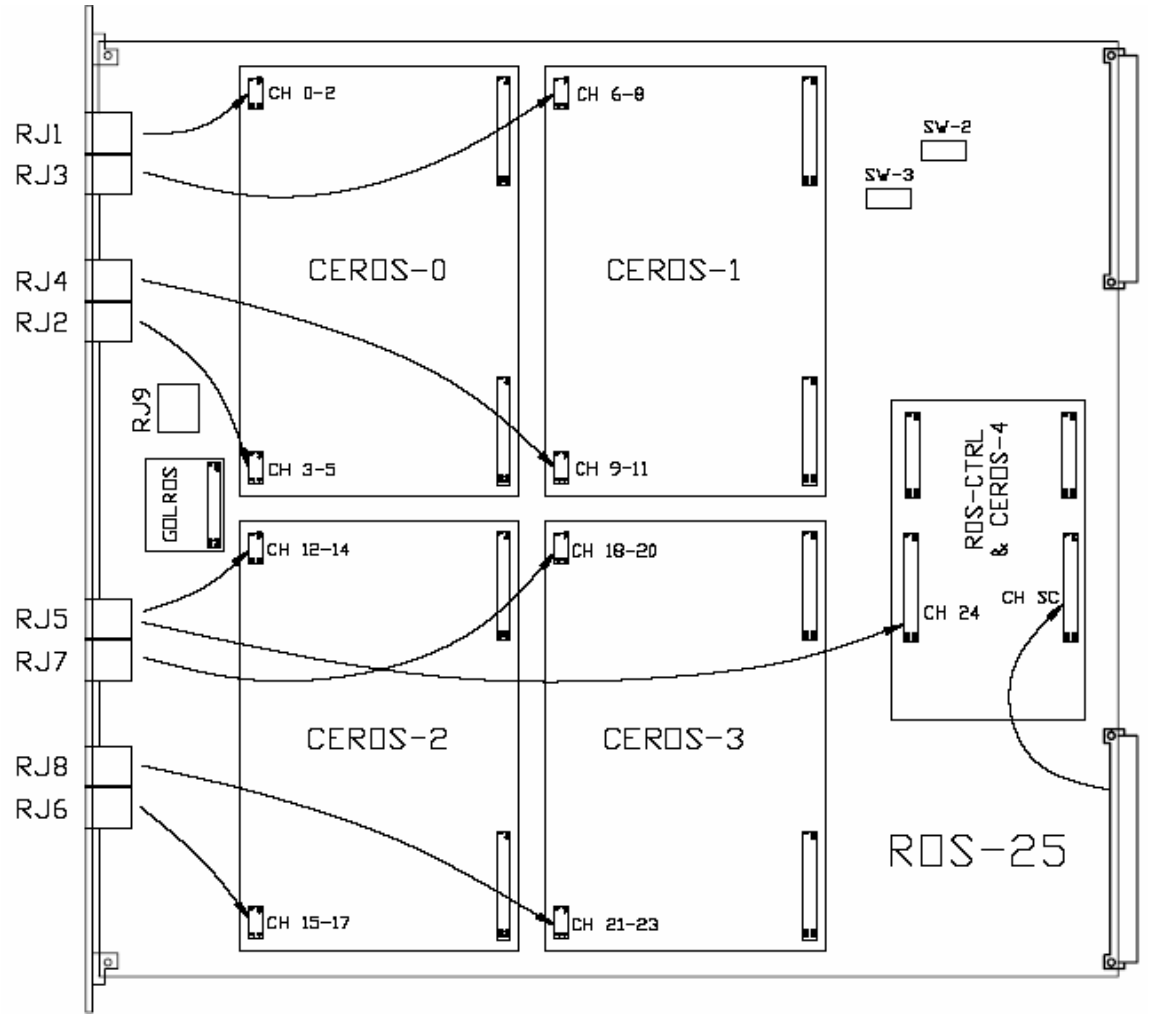

Figure 5: Diagram of the ROS-25 with its main piggy boards.

Another functional module in the ROS is called ROSCTRL. This module contains: first of all, the "Sector Collector channel", and second, the control of the whole read-out functionality.

In certain modes of operation, it may be desired to read the Sector Collector trigger data within the DAQ data flow. Therefore, each Sector Collector board will send, at each event, their

data to its contiguous ROS in the crate, and these data will be treated by the ROS as if it was another input channel

Finally, we can find also the GOLROS module, the last functional block in the ROS which includes the GOL serializer and the VCSEL optical transmitter.

Related to how the read-out is performed, the ROS can be programmed in different modes of operation:

- **Normal operation mode:**
  The 25 input channels at the ROS receive data from the ROBs and send it to the DDU through an optical link.

- **Spy mode:**
  Besides sending the data to the DDU, the ROS can store a programmable number of words or of events in a spy memory. This memory can be accessed from a VME interface in normal or block transfer mode.

- **Transmission test mode (GOL_TEST):**
  Through a VME interface, data can be written in the internal ROS memory and then sent through the optical transmitter to the DDU. The number of words sent and also the average bandwidth can be selected.

- **Straight FIFO read-out:**
  Basically, this mode of operation has been implemented for debugging purposes, where data are read through VME directly from the input FIFOs. In this way, the format of the data will be the same that comes from the HPTDC, as the ROS is almost transparent in this operation mode.

  These different operation modes are schematically represented in the following figure:

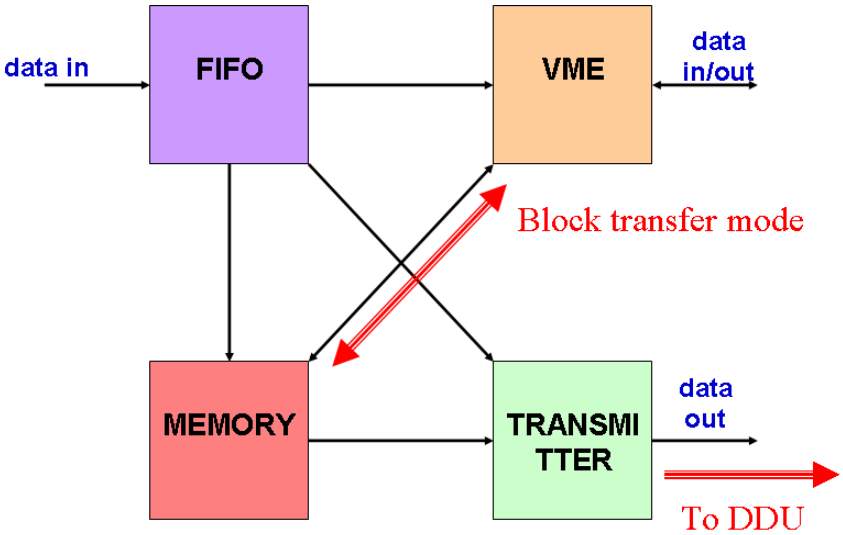

Figure 6: ROS-25 functional diagram.

5 ROS-25 VME INTERFACE

The ROS-25 supports different access modes:

Access type	Data Width	Minimum address space needed	
Short non-privileged A16	Word (16 bits)	1024 words	Access to all of the ROS-25 registers
Standard A24 non-privileged	Word (16 bits)	256 kwords	Access to the memory.
Standard A24 standard non-privileged block transfer	Word (16 bits)	256 kwords	Access to the memory.

The A16 base address is selected with switch SW3, (A15-A10), and the A24 address is selected with switch SW2, (A23–A20). The address mapping for the registers of the different functional blocks is what follows:

A16 Base Address (A15 – A10)	Functional block
Base Address + 0x00	CEROS0
Base Address + 0x80	CEROS1
Base Address + 0x100	CEROS2
Base Address + 0x180	CEROS3
Base Address + 0x200	CEROS4
Base Address + 0x280	ROSCTRL
Base Address + 0x300	MEMCTRL
Base Address + 0x380	ROSVME

5.1 Memory access : A24 access, word width

It can be standard or non-privileged block data transfer.

Address from 0x00000 to 0x7FFFE (256kwords = 262144 words)

0-15	Memory Data	R/W

5.2 Registers access: A16 access, word width.

5.2.1 ROSVME

All the registers from ROSVME are not reset by a global reset but by a local reset, except for the registers belonging to the PCA9564 device.

Control y status (ROSVME + 0x00)

Default value = 0x7D

bits	description	acc	
0	ROS waiting	R	1 = ROS has finished processing an event.
1	Memory Done	R	1 = programmed limit achieved
2-6	FPGA's programmed	R	1 = corresponding FPGA properly programmed
7	Registers loaded	R	TBI
8	Enable EVCNT_reset to perform a global reset in the ROS-25	R/W	
9	Select GOL clock	R/W	0 = ckdesv1, 1 = ckdesv2
10	Select ROS-25 clock	R/W	0 = ckdesv1, 1 = ckdesv2
11	Load PAF values in all the FIFOs	W	
12	Partial CEROS FIFO reset (PRS)	W	
13	Master CEROS FIFO reset (MRS)	W	
14	Local reset	W	Resets only those registers from the ROSVME functional block (except for the PCA9564 chip)
15	Global board reset	W	Resets the rest of the devices in the ROS board.

A VME sys-reset will force also a local reset and a global board reset.

Bits 9 and 10 select among the two output clocks from the TTCrx device with their corresponding programmed de-skew. As the output of the ckdesv2 signal is disabled at the TTCrx device by default and for the ROS-25 prototypes it is not recommended to use ckdesv2, we do not recommend writing a 1 in any of those two bits. (The last does not apply to the ROS-25 prototype at Torino).

VME FLASH (ROSVME + 0x02) To be implemented

bits	description	acc
9-15	VME FLASH page for loading the registers	R/W

SPAF / SFF (ROSVME + 0x04)

Default value: 0x00

bits	description	acc	
0-4	SPAF	R	1 = Almost Full flag active in any of the FIFOs of each CEROS.
5-9	SFF	R	1 = Full flag active in any of the FIFOs of each of the CEROS.

Interruptions (ROSVME + 0x06)

Default = 0x 20

bits	Description	acc	
0	enable SPAF interrupt	R/W	If any of the FIFOs signals the almost full flag (PAF), an interruption will take place.
1	enable SFF interrupt	R/W	If any of the FIFOs gets full at any time, an interruption will take place.
2	enable memory done interrupt (rising edge)	R/W	An interruption will occur when the ROS has achieved the programmed condition (number of words/events to be transmitted/stored on memory).
3	enable ROS waiting interrupt (rising edge)	R/W	An interruption will occur when the ROS finishes processing an event. To be used only in normal or spy mode.
4	interrupt requested	R	= 1 An interruption from the ROS has occurred.
5-7	interrupt level	R/W	
8-15	interrupt vector	R/W	

FPGA Control (ROSVME + 0x08)

Default value = 0x1000

bits	description	acc	
0-4	FPGAs INIT signal	R/W	If the INIT bit is set to one, whenever that FPGA is reprogrammed it will be standing in erase mode until the INIT bit is released.
5-9	Reprogram the FPGAs	W	When writing a 1 here, the FPGAs are reprogrammed
10-12	Select clock frequency for loading FPGAs from their program memories	R/W	
13	Enables reprogramming FPGAs with an event reset signal.	R/W	
14	Enables reprogramming FPGAs with a global reset signal.	R/W	

Note: FPGA load clock frequency:

- 000 – 1,25 MHz
- 001 – 2.5 MHz
- 010 – 5 MHz
- 011 – 10 MHz
- 100 – 20 MHz
- Others => 1,25 MHz

JTAG Control (ROSVME + 0x0A)

Default value = 0x7

bits	description	acc
0-2	Selection of the device to be configured through connector J1	R/W

0 Configure CEROS 0
1 Configure CEROS 1
2 Configure CEROS 2
3 Configure CEROS 3
4 Configure ROSCTRL
5 Configure ROSMEM
6 Configure devices through VME
7 Device selected from corresponding position of switch SW1.

JTAG Register 1 (ROSVME + 0x0C) To be implemented

bits	description	acc
0-3	Enables JTAG access to each CEROS FLASH	R/W
4	Enables JTAG access to ROSCTRL FLASH	R/W
5	Enables JTAG access to ROSMEM	R/W
6	flash_tck	R/W
7	flash_tms	R/W
8	flash_tdi	R/W
9-14	flash_tdo	R

JTAG Register 2 (ROSVME + 0x0E) To be implemented

bits	description	acc
0-3	Number of bits to send by JTAG	R/W
4-6	JTAG clock frequency	R/W

JTAG Data (ROSVME + 0x10) To be implemented

bits	description	acc
0-15	Sequence of bits to send (TDI)	W
0-15	Parity of the sequence of bits received (TDO)	R

I2C & 1-wire & GOL & QPLL (ROSVME + 0x12)

bits	description	acc
0	Enable I2C access to GOL	R/W
1	Enable I2C access to I,V, T sensors	R/W
2	GOL power off	R/W
3	GOL ready	R
4	QPLL error registered	R
5	QPLL locked	R
6	QPLL unlocked registered	R
7	GOL not ready registered	R/W
8	QPLL reset	W

Bits 0 and 1 can not be enabled at the same time, so if ever a 1 is written on both at the same time, only the access to the I, V, T sensors will be enabled.

PCA STATUS (ROSVME + 0x20)

bits	description	acc	default
0-7	Status	R	0xF8
0-7	Time-out	W	0xFF

PCA DATA (ROSVME + 0x22)

Default = 0x00

bits	description	acc
0-7	Data	R/W

PCA ADDRESS (ROSVME + 0x24)

Default = 0x00

bits	description	acc
0-7	Own address	R/W

PCA CONTROL (ROSVME + 0x26)

Default = 0x00

bits	description	acc	Description
7	AA	R/W	Assert acknowledge flag
6	ENSIO	R/W	SIO enable bit
5	STA	R/W	Start flag
4	STO	R/W	Stop flag
3	SI	R/W	Serial interrupt flag
2-0	CR	R/W	Serial clock rate

CR: Serial clock frequency

000	330 kHz
001	288 kHz
010	217 kHz
011	146 kHz
100	88 kHz
101	59 kHz
110	44 kHz
111	36 kHz

5.2.2 ROSMEM

Control y status (ROSMEM + 0x00)

Default value = 0x00

bits	description	acc
0	Enables transfer from memory to GOL. Mode GOL_TEST. If only this bit is set to 1, an unlimited transfer takes places. It can be stopped again by writing a 0 into this bit.	R/W
1	Enables storing FIFOs data in ROS memory (SPY MODE). Any of bits 2, 3 or 4 should also be selected. The storage can also be forced to stop by writing a 0 again into this bit, in such a case, if limit by number of events is selected, the ROS will stop to write on the memory as soon as the actual event is completed.	R/W
2	Limits transfer by number of words. For GOL_TEST or SPY MODE.	R/W
3	Limits transfer by number of events. Only for SPY MODE.	R/W
4	Enables transfer until memory full. For mode GOL_TEST or SPY MODE. 262143 words will be sent or written into the memory.	R/W
5	Enables repeat cycles of number of words for sending data to the transmitter. Only for mode GOL_TEST.	R/W
6	Selects random transmission enable for GOL_TEST mode of operation.	R/W
7-10	Enables corresponding LFSR register for random transmission enable. Only for mode GOL_TEST.	R/W

The transmitter enable signal that loads the 16 bit words data into the GOL transmitter is generated by default in GOL_TEST mode as a 20 MHz clock, obtaining therefore a bandwidth of 320 Mbps.

If bit 6 is set to 1, then, a random transmitter enable signal can be chosen. This random signal is obtained as the output of some pseudo-random shift registers (LFSR). Bits 7 to 10 enable each a LFSR register, and the transmitter enable signal is obtained as the logical AND of each of the enabled LFSR.

As these LFSR run at 40 MHz, the average frequency is also 20 MHz. By enabling more LFSRs the average bandwidth can be reduced to something more similar to the expected value during normal ROS operation mode: (~100 Mbps)

1 LFSR enabled: 20 MHz => 320 Mbps
2 LFSR enabled: 10 MHz => 160 Mbps
3 LFSR enabled: 5 MHz => 80 Mbps
4 LFSR enabled: 2,5 MHz => 40 Mbps

Memory pointer low (ROSMEM + 0x02)

Default value = 0x00

bits	description	acc
0-15	Memory address pointer (bits 0-15)	R/W

Memory pointer high (ROSMEM + 0x04)

Default value = 0x00

bits	Description	acc
0-1	Memory address pointer (bits 16-17)	R/W
2	Resets the memory pointer	W

The memory pointer gives the number of 16 bit words stored in the memory.

Max word (16 bits) count low (ROSMEM + 0x06)

Default value = 0x00

bits	Description	acc
0-15	Maximum number of words (bits 0-15)	R/W

Max word (16 bits) count high (ROSMEM + 0x08)

Default value = 0x00

bits	Description	acc
0-1	Maximum number of 16 bits words (bits 16-17)	R/W
2	Resets max word count register	W

Max event count (ROSMEM + 0x0A)

Default value = 0x00

bits	Description	acc
0-15	Maximum number of events	R/W

Cycle count (ROSMEM + 0x0C)

Default value = 0x00

bits	description	acc
0-15	Number of cycles to repeat the transmission	R/W

If number of cycles is set to 0, then 65536 cycles (0x10000) will be performed.

5.2.3 CEROS 0-4

CEROSX is replaced by the base address of the corresponding CEROS. Note that CEROS 4 only has one channel (related information is located in the least significant bit).

LOCK / MASK (CEROSX + 0x00)

bits	description	acc	
0-5	Channel UNLOCK	R	1 = channel is unlocked
6-11	MASK Channel	R/W	1 = channel masked
12-14	CEROS identification	R	0 for ceros 0, 1 for ceros 1, 2 for ceros 2, 3 for ceros 3, 4 for ceros 4

DISABLE REGISTER (CEROSX + 0x02)

Default value = 0x84

bits	Description	Acc
0	Disable NOT LOCK to block channel	R/W
1	Disable HAS UNLOCK to block channel	R/W
2	Disable TIMED OUT or max words to block channel	R/W
3	Disable FIFO Full to block channel	R/W
4	Disable send PAF warning word within the data.	R/W
5	Disable send EVID misalignment error word within the data.	R/W
6	Enable send HPTDC data although there are no hits.	R/W
7	Disable EVID misalignment to block channel	R/W
10-15	Channel blocked	R

If for any reason the ROS blocks automatically one of the channels, it will be signalled with a 1 in the Channel blocked field. A channel will appear as blocked either because the user has masked it by writing in register LOCK/MASK or due to any of the following malfunctions:

- The channel is unlocked and the option “Disable not lock to block channel” was set to 0.
- The channel has unlocked at any time (although at the moment is again locked) and the option “Disable has unlock to block channel” was set to 0.
- The channel has timed out and the option “Disable time-out or max words to block channel” was set to 0.
- More than the programmed number of consecutive words have been read out from that channel without finding a ROB trailer and the option “Disable time-out or max words to block channel” was set to 0.
- The FIFO of that channel has been full and the option “Disable FIFO full” was set to 0.
- There has been an Event Identification misalignment in that channel and the option “Disable EVID misalignment to block channel” was set to 0.

TIMEDOUT / HASUNLOCK (CEROSX + 0x04)

bits	Description	acc
0-5	Channel has TIMED OUT	R/W
6-11	Channel HAS UNLOCK	R/W

There are two reasons why a channel can signal a timeout:

A channel gives a timeout when the ROS is waiting to read that channel and its input FIFO is empty longer than the timeout value programmed. Therefore, the programmed timeout should be long enough to cope with the different latency of the L1A signal between the Minicrates and the ROS and include also the propagation time between the cooper links from Minicrates to ROS.

This timeout value is common for each CEROS, i.e. 6 channels except CEROS 4 which only has 1 channels (register TIMEOUT VALUE).

PAF (CEROSX + 0x06)

bits	Description	acc	
0-5	FIFO's PAF	R	= 1 an Almost Full condition has been achieved in the FIFO of the corresponding channel.
6-11	FIFO's PAF REGISTERED	R/W	Registered value of the previous.

EF (CEROSX + 0x08)

bits	description	acc	
0-5	FIFO's EF	R	1 = FIFO is empty
6-11	Event ID misalignment error	R/W	1 = that channel had an event ID misalignment error.

FF and MAXWORDS (CEROSX + 0x0A)

bits	description	acc	
0-5	Maximum number of words reached	R/W	1 = max number of words reached
6-11	FIFO's FF REGISTERED	R/W	1 = FIFO has been full at any time.

If the ROS is reading one channel, and reads more than the programmed maximum number of words and does not find a ROB trailer, it stops reading that channel and flags the maximum number of words reached for that channel.

PAF VALUE (CEROSX + 0x0C)

Default value = 0x200

bits	description	acc	
0-10	FIFO's PAF VALUE	R/W	When the number of empty 16 bits words in a FIFO is smaller than this value, the FIFO will set to 1 the PAF flag. This value is common all channels in a CEROS.
11	RECEIVERS OFF	R/W	= 1 the 6 channels will turn off their receivers, so no data will be received.

The 2kwords input FIFO for each channel has a flag (PAF flag) to signal that the number of empty positions at the FIFO is smaller than a programmed limit, the PAF value.

By default, the PAF value at the FIFOs after a Master Reset (MRS) is 15, that is, when the number of words in the FIFO is larger than 2049-15 = 2034, the PAF flag will be set to 1.

In order to program a new PAF value, it is necessary to write the desired PAF value in this register (at each CEROS) and then write a 1 in bit 11 of register ROSVME+0x00 to download the new value into the FIFOs. During this task, the channels will be unlocked, so before taking data again, the TIMEDOUT/HASUNLOCK register should be erased.

If a Master CEROS FIFO reset (MRS) is performed, the programmed PAF value in the FIFOs will be erased. To empty the data in a FIFO without erasing the programmed PAF value, a Partial CEROS FIFO reset (PRS) should be used instead.

FIFO WORD PARITY (CEROSX + 0x0E)

bits	description	acc	
0-15	FIFO WORD PARITY	R/W	= 1 a parity error has been detected in a word. The position of that word in the data stream is the same as the corresponding position of the parity bit in this register.

This is a shift register where it is stored the parity errors of the 16 bits words read from the FIFO, starting from the least significant bit.

FIFO BYTE PARITY 0 (CEROSX + 0x10)

bits	description	acc
0-15	FIFO BYTE PARITY bits 0-15	R/W

FIFO BYTE PARITY 1 (CEROSX + 0x12)

bits	description	acc
0-15	FIFO BYTE PARITY bits 16-31	R/W

In these two registers, the actual parity and not the parity error is shifted starting by the least significant bit. In this case, each bit corresponds to the parity of a byte and not of a 16 bits word.

FIFO DISPARITY COUNTER (CEROSX + 0x14)

bits	description	acc
0-15	FIFO DISPARITY COUNTER	R/W

Each time a parity error is found in any of the channels of a CEROS, this counter will be incremented.

TIMEOUT VALUE(CEROSX + 0x16)

Default value = 0xFFF (102,4 μs)

bits	description	acc	
0-11	TIME OUT VALUE for all channels of this CEROS	R/W	Number of clock cycles before timeout.

This value corresponds to the number of 40 MHz clock cycles that the ROS will wait when it is reading a channel but its FIFO is empty.

MAXIMUM NUMBER OF WORDS VALUE(CEROSX + 0x18)

Default value = 0x1FF

bits	description	acc	
0-7	Maximum number of consecutive words allowed to be read out from one input channel. (Steps of 256 words.)	R/W	
8	Unlimited read-out	R/W	1 = The number of words to be read from one channel is not limited

FIFO PAF COUNTER 0-5 (CEROSX + 0x20-2A)

bits	description	acc
0-15	FIFO 0-5 PAF counter	R/W

It counts the number of times that the programmed almost full limit has been achieved in a particular channel of that CEROS.

FIFO DATA 0-5 (CEROSX + 0x30-3A)

bits	description	acc
0-15	FIFO 0-5 data	R

TO BE USED ONLY FOR DEBUGING.

By reading at this register, the data from the corresponding channel is taken out from the input FIFO, that is, once read, **it cannot be read again**. When all the data have been read, the Empty FIFO flag will be signalled and the last word read will remain at the FIFO output, so the same value will be read again in following accesses.

Note that data read from this register has the HPTDC data format, and therefore, there is no indication of the number of channel that is being read.

PAE/PAF PROGRAMMED 0-5 (CEROSX + 0x40-4A)

bits	description	acc
0-10	FIFO 0-5 PAE/PAF programmed	R

The PAF value programmed at each FIFO can be read from these registers.

This register **should only be accessed during configuration, not during data acquisition,** otherwise data at the FIFOs can be corrupted.

5.2.4 ROSCTRL

CEROS or SC TIMED OUT (ROSCTRL + 0x00)

Default value = 0x00

bits	description	acc
0-4	CEROS timed out	R/W
5	SECTOR COLLECTOR channel has timed out	R/W
6	SECTOR COLLECTOR channel FIFO full registered.	R/W
7	Bunch counter FIFO full registered.	R/W
8	Event counter HIGH FIFO full registered.	R/W
9	Event counter LOW FIFO full registered.	R/W

The Sector Collector channel is not blocked if at any time its FIFO is full, but some words may be lost. At the moment, the Sector Collector FIFO is 511 words (16 bits) deep, so it should be large enough to cope with the data received from the Sector Collector for 1 event.

MASKS (ROSCTRL + 0x02)

Default value = 0xD40

bits	description	acc
0-4	Mask CEROS	R/W
5	Enable SECTOR COLLECTOR channel	R/W
6	Disable TIMED OUT to mask CEROS or SECTOR COLLECTOR channel	R/W
7	Enable reception of AUTOL1A trigger at ROS	R/W
8	Disable reception of TTC L1A trigger from TIM module.	R/W
9	Select ROS internal Event ID counter	R/W
10	Enable send BUNCH COUNTER debugging word within the data flow.	R/W
11	Enable send BUNCH RESET COUNTER debugging word within the data flow.	R/W
13	Read Internal SECTOR COLLECTOR FIFO (to be used for debugging)	R/W
15	Generate local VME trigger at ROS.	W

Note that the TTC L1A is disabled by default at a reset to allow proper configuration of the ROS although the TTC system is sending L1As.

The bit number 13, for reading internal Sector Collector FIFO should be set only for debugging purposes. If it is set, the ROS_READY-TRIGGER_STROBE handshake will not take place.

Bit number 9, “Select local Event ID counter” should be used either for debugging purposes when there is no TTC or TIM boards present, or when the L1A comes from the Sector Collector board (AutoL1A). In such a case, L1As from the TIM board should be disabled (bit 8 to 0).

TIMEOUT VALUE (ROSCTRL + 0x04)

Default value = 0xFFF (102,4 μs)

bits	description	acc
0-11	TIME OUT VALUE for CEROS and SECTOR COLLECTOR channel	R/W

For the Sector Collector channel, this value represents the number of clock cycles since the ROS_READY signal was set to 1 until a timeout is given because there has been no TRG_STROBE signal from the Sector Collector board.

For the CEROS module, this value should be larger than the maximum number of cycles needed for processing one event in one channel (0x190 ~10 μs).

The programmed value should match the larger of these two timeouts.

BUNCH NUMBER (ROSCTRL + 0x06)

bits	description	acc	
0-11	Read last BUNCH NUMBER	R	
0-11	Write BUNCH NUMBER in FIFO	W	Should be used only for ROS debugging.

EVENT LOW NUMBER (ROSCTRL + 0x08)

bits	description	acc	
0-11	Read last EVENT LOW NUMBER	R	
0-11	Write EVENT LOW NUMBER in FIFO	W	Should be used only for ROS debugging.

EVENT HIGH NUMBER (ROSCTRL + 0x0A)

bits	description	acc	
0-11	Read last EVENT HIGH NUMBER	R	
0-11	Write EVENT HIGH NUMBER in FIFO	W	Should be used only for ROS debugging.

SECTOR COLLECTOR DATA (ROSCTRL + 0x0C)

bits	description	acc	
0-15	Write SECTOR COLLECTOR data in SECTOR COLLECTOR FIFO	W	Should be used only for ROS debugging.
0-8	SECTOR COLLECTOR FIFO occupancy (16 bits words)	R	

BUNCH COUNTER FIFO OCCUPANCY (ROSCTRL + 0x0E)

bits	description	acc
0-7	BUNCH COUNTER FIFO maximum occupancy registered	R

EVENT COUNTER FIFO OCCUPANCY (ROSCTRL + 0x10)

bits	description	acc
0-7	EVENT LOW FIFO maximum occupancy registered	R
8-15	EVENT HIGH FIFO maximum occupancy registered	R

These two registers store the maximum occupancy of the corresponding FIFOs since last global reset.

These FIFOs are 255 words deep, and store the Bunch and Event counter values received from the TIM module through the TIM backplane. Once the ROS has finished processing one event, it will look at these FIFOs to see if there is any event pending and will start processing it. Therefore, if these FIFOs are full at any time, some L1As may be lost.

TRIGGER COUNTER (ROSCTRL + 0x12)

bits	description	acc
0-15	All triggers counter	R/W

This register is an internal counter in the ROS that is incremented at each L1A arrival, no matter which is the source (VME, AUTOL1A from Sector Collector or TTC system). It will be incremented even if that L1A source is not enabled.

ORBIT COUNTER (ROSCTRL + 0x14)

bits	description	acc
0-15	Orbit counter (bits 0-15)	R/W

This register is an internal counter in the ROS that is incremented at each Bunch reset signal, i.e., it will count the number of orbits. This value can be sent within the data flow in the Bunch reset counter debugging word.

EVENT ID COUNTER (ROSCTRL + 0x18)

Default value = 0xFFF

bits	description	acc
0-11	Event ID counter	R/W

This register is an internal counter in the ROS that is incremented when a L1A is received from any of the sources only if they have been enabled, that is:

- Local L1A generated from a VME access (bit 15 of register ROSCTRL+0x02).
- AutoL1A from the SECTOR COLLECTOR (Only if it has been enabled by writing a 1 on bit 7 of register ROSCTRL+0x02).
- L1A from the TTC system. (Only if it has been enabled by writing a 0 on bit 8 of register ROSCTRL+0x02).

This is the event ID that will be sent in the ROS Event Header word if ROS internal Event counter is selected (bit 9 of register ROSCTRL+0x02). Note that as the Event ID received from the TTC system, the first L1A will be marked with Event ID = 0, so in principle its value will be one less than the number of L1As received.

Also, note that if internal event ID is selected, the L1As will not be stored in a FIFO, therefore, if a L1A is received while the ROS is still processing a previous event, the event ID counter will be incremented, and when the ROS finishes, it will start to process the next event with the event ID value that it reads from the counter, so an event ID misalignment will happen.

6 ROS-25 CONFIGURATION

6.1 Mode of operation: Memory readout.

Reset the ROS-25 board.

Write 0xF000 to register ROSVME+ 0x00.

Power up the GOL.

Write 0x0 to register ROSVME+ 0x12 to power up the 2.5V power supply.
Wait ~800ms.
Write 0x0 to register ROSVME+ 0x12 to erase all the registered flags.

Optional: Mask channels not used.

For example: Mask full ceros 3 (channels from 18 to 23)
Write 0x008 to register ROSCTRL+ 0x02.
Note that this register has other bits that should not be modified unless desired.

Mask channel 16 which corresponds to channel 4 of ceros 2.
Write 0x400 to register CEROS2+ 0x00.

Optional: Write timeout values.

For example:
Write 0x1FF to register CEROS0+ 0x16.
Write 0x1FF to register CEROS1+ 0x16.
Write 0x1FF to register CEROS2+ 0x16.
Write 0x1FF to register CEROS3+ 0x16.
Write 0x1FF to register CEROS4+ 0x16.

Write 0xFFF to register ROSCTRL+ 0x04.

Optional: Write PAF values

Write PAF value in register CEROS0 + 0x0C
Write PAF value in register CEROS1 + 0x0C
Write PAF value in register CEROS2 + 0x0C
Write PAF value in register CEROS3 + 0x0C
Write PAF value in register CEROS4 + 0x0C

Load PAF values: Write a 1 in bit 11 of ROSVME + 0x00.
Note that this register has other bits that should not be modified unless desired.

Reset timeouts and unlocks flags.

Write 0x0000 to register CEROS0+ 0x04.
Write 0x0000 to register CEROS1+ 0x04.
Write 0x0000 to register CEROS2+ 0x04.
Write 0x0000 to register CEROS3+ 0x04.
Write 0x0000 to register CEROS4+ 0x04.

Write 0x0000 to register ROSCTRL+ 0x00.

Optional. Specify different options for disabling channels.

For example: If you want to disable that the ROS-25 sends an error Ev ID word when events are not aligned, you would have to write a 0 in bit 5 of register CEROSX+0x02.

If you want to disable that a channel is blocked when its FIFO is full then you would have to write a 1 on bit 3 of CEROSX+0x02.

Write maximum number of 16 bit words or of events to be written in memory.

Write max number words to registers ROSMEM+ 0x06 and ROSMEM+ 0x08.
Write max number of events to register ROSMEM+ 0x0A.

Reset memory pointer.

Write 0x0004 to register ROSMEM+ 0x04.

Optional: Program interruptions.

For example, program interruption by "memory done" at level 2 with an interruption vector of 0xC4:
Write 0xC444 at ROSVME + 0x06.

Specify options for ROSCTRL and enable L1A:

For example:

Enable sending Bunch ID debug word or bunch reset counter. Write a 1 in bit 10 and 11 of register ROSCTRL+0x02.

If Sector Collector channel wants to be read, bit 5 of ROSCTRL + 0x02 should also bit set to 1.

Furthermore,

At some point before starting the data acquisition a L1A source should be enabled:

- in normal operation mode it will be the TTC L1A, that is bit 7 and bit 8 to 0 in ROSCTRL + 0x02.
- if AUTOL1As are going to be used as trigger signal, bit 7 and bit 8 should be set to 1. In such a case, the bunch and event counters received from the TIM will not have sense, so the Bunch counter and Bunch counter resets words should be disabled (bit 10 and 11 to 0), and also, internal Event counter should be chosen (bit 9 to 1).

Enable transfer mode

This is the last access that should be done before data acquisition starts. For sending data from the FIFOs to the memory, bit 1 of register ROSMEM+0x00 should be set to 1.

If you want to limit the number of words written to memory you would have to set to 1 also bit 2.
Either, if you want to limit the number of events written to memory you would have to set to 1 bit 3.

Start the data acquisition.

Send N L1A through the TTC system.
Wait until Memory Done = 1. (bit 1 of register ROSVME+0x00)
If you want to read the memory contents before Memory Done=1 then you would have to disable the transfer from FIFOs to memory. Write a 0 in bit 1 of register ROSMEM+0x00.
Read memory pointer.
Read from the memory the number of 16 bits words specified by the memory pointer.
Reset memory pointer.
Enable again the transfer mode.

6.2 Check status

In order to check that the readout and the ROS-25 is behaving properly, there are some checking that can also be performed after configuration is done or every once in a while.

- Check that all the FPGAS have been properly loaded. (Read bits 6 to 2 of ROSVME+0x00. If 1, they are ok).
- Check the channel blocked flags. (bits 10 to 15 of CEROSX + 0x02).
- Check the empty FIFO flags.
- Check the event ID misalignment flags. (Register CEROSX + 0x08).
- Check the full FIFO flags. (Register CEROSX + 0x0A). (At least the registered values).
- Check PAF flags. (Register CEROSX + 0x06) (At least the registered values).
- Check if any channel has been unlocked or timedout (register CEROSX+0x04) or the actual lock status (register CEROSX+0x00)
- Check timeouts of the CEROS and of the Sector collector channel.
- Check FIFO full flag of the SECTOR COLLECTOR channel.
- Check FIFO full flags of the Bunch, Event High and Event Low FIFOs. (Register ROSCTRL+0x00)
- Check GOL ready and GOL not ready registered flag and QPLL locked and QPLL unlocked registered flag. (Register ROSVME + 0x12).

7 INTERFACE TO THE VOLTAGE, CURRENT AND TEMPERATURE SENSORS.

At the ROS there are three sensors that allow measuring each of the three voltages used at this board (5 V, 3.3 V and 1.8 V), measure their corresponding currents and performing temperature monitorization.

These sensors (DS2438) are 1-wire devices, which can be accessed through an I2C to 1-wire interface (DS2482). Channel 0 of the DS2482 is connected to DS2450 device for reading the laser transmitter optical power (this is not explained in this version of the manual). The other three channels are connected as indicated in the table:

Channel number	Sensor	VAD	VDD	Current
Channel 1	5 V sensor	5 V	3.3 V	5 V
Channel 2	3.3 V sensor	3.3 V	3.3 V	3.3 V
Channel 3	1.8 V sensor	1.8 V	3.3 V	1.8 V

The VME to I2C interface is implemented through the device PCA9564 [3]. There is only one PCA9564 device to perform I2C access, both to the GOL and to the DS2482 bridge to the sensors. To choose among both I2C buses, a bit has to be enabled on register " I2C & 1-wire & GOL & QPLL (ROSVME+0x12)":

- Write a 1 in bit 0 for enabling access to the GOL.
- Write a 1 in bit 1 for enabling access to the 1-wire sensor.

Setting both bits to 1 is not allowed, having priority the I,V,T 1-wire sensor.

In the following picture, a diagram of how the different devices are interconnected at the ROS is presented:

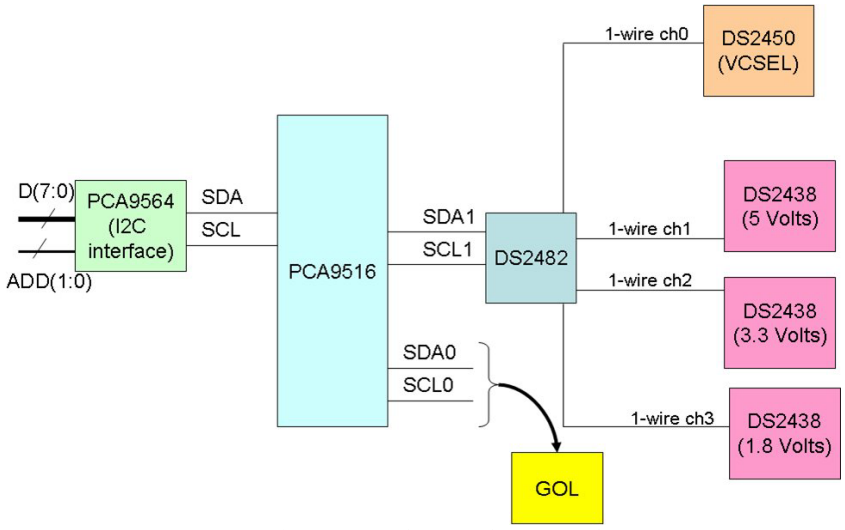

Figure 7: Diagram of the I2C buses at the ROS.

7.1 I2C interface through the PCA9564

As can be seen in its datasheet [3], the PCA9564 device has 4 registers (Status, Data, Own address and Control), which can be accessed directly as ROS registers. Here we present again the corresponding registers at the ROS for the PCA9564 used for reading the sensors and for the PCA9564 used for interface with the GOL. (Remember that ROSVME=ROS Base Address + 0x380).

PCA_STATUS (ROSVME + 0x20)

bits	description	acc	default
0-7	Status	R	0xF8
0-7	Time-out	W	0xFF

PCA_DATA (ROSVME + 0x22)

Default = 0x00

bits	description	acc
0-7	Data	R/W

PCA_ADDRESS (ROSVME + 0x24)

Default = 0x00

bits	description	acc
0-7	Own address	R/W

PCA_CONTROL (ROSVME + 0x26)

Default = 0x00

bits	description	acc	description
7	AA	R/W	Assert acknowledge flag
6	ENSIO	R/W	SIO enable bit
5	STA	R/W	Start flag
4	STO	R/W	Stop flag
3	SI	R/W	Serial interrupt flag
2-0	CR	R/W	Serial clock rate

CR: Serial clock frequency

000	330 kHz
001	288 kHz
010	217 kHz
011	146 kHz
100	88 kHz
101	59 kHz

110	44 kHz
111	36 kHz

The procedure for writing and reading via I2C is described in the PCA9564 datasheet; however, we include here a summary of routines Write_PCA(out p_add, out p_data) and Read_PCA (out p_add, in p_data), where p_add is the I2C address of the device you want to communicate with and p_data is the 7 bit data to send or receive.

The PCA9564 will be used in Master mode in every access; therefore, PCA_ADDRESS register is not needed.

The first step that has to be done is to enable SIO logic in the device at register PCA_CONTROL which requires to wait 10 ms before starting any other access. Unless this bit is disabled by the user at some point, this step needs to be done only once at the beginning. At this point, the I2C bit rate can also be selected on this register.

The next steps are summarized in what follows:

I2C WRITE ACCESS THROUGH THE PCA9564

Write_PCA(out p_add, out p_data)

1) Check PCA9564 status register, it has to be 0xF8. (Read ROS_Base_Add + 0x3A0).
2) Send START (Write 0x60 in ROS_Base_Add + 0x3A6).
3) Wait until interrupt flag (SI) is asserted. (Read bit 3 of ROS_Base_Add + 0x3A6).
4) Check status, it has to be 0x08. (Read ROS_Base_Add + 0x3A0).
5) Send the I2C address of the device you want to access to by writing p_add in bits 7 to 1 and a 0 (write) in bit 0 of PCA_DATA. (Write p_add & '0' in ROS_Base_Add + 0x3A2).
6) Reset SI flag. (Write 0x40 in ROS_Base_Add + 0x 3A6).
7) Wait until interrupt flag (SI) is asserted. (Read bit 3 of ROS_Base_Add + 0x3A6).
8) Check status, it has to be 0x18. (Read ROS_Base_Add + 0x3A0).
9) Send the data byte that you want to write by writing it in PCA_DATA. (Write p_data on ROS_Base_Add + 0x3A2).
10) Reset SI flag. (Write 0x40 in ROS_Base_Add + 0x 3A6).
11) Wait until interrupt flag (SI) is asserted. (Read bit 3 of ROS_Base_Add + 0x3A6).
12) Check status, it has to be 0x28. (Read ROS_Base_Add + 0x3A0).
13) Send a STOP condition. (Write 0x50 to ROS_Base_Add + 0x3A6).
14) Check status, it has to be 0xF8. (Read ROS_Base_Add + 0x3A0).

I2C READ ACCESS THROUGH THE PCA9564

Read_PCA (out p_add, in p_data)

1) Check PCA9564 status register, it has to be 0xF8. (Read ROS_Base_Add + 0x3A0).
2) Send START (Write 0x60 in ROS_Base_Add + 0x3A6).
3) Wait until interrupt flag (SI) is asserted. (Read bit 3 of ROS_Base_Add + 0x3A6).

4) Check status, it has to be 0x08. (Read ROS_Base_Add + 0x3A0).
5) Send the I2C address of the device you want to access to by writing p_add in bits 7 to 1 and a 1 (read) in bit 0 of PCA_DATA. (Write p_add & '0' in ROS_Base_Add + 0x3A2).
6) Reset SI flag. (Write 0x40 in ROS_Base_Add + 0x 3A6).
7) Wait until interrupt flag (SI) is asserted. (Read bit 3 of ROS_Base_Add + 0x3A6).
8) Check status, it has to be 0x40. (Read ROS_Base_Add + 0x3A0).
9) Reset SI flag. (Write 0x40 in ROS_Base_Add + 0x 3A6).
10) Wait until interrupt flag (SI) is asserted. (Read bit 3 of ROS_Base_Add + 0x3A6).
11) Read the data received from the device from PCA_DATA. (Read p_data from ROS_Base_Add + 0x3A2).
12) Check status, it has to be 0x58. (Read ROS_Base_Add + 0x3A0).
13) Send a STOP condition. (Write 0x50 to ROS_Base_Add + 0x3A6).
14) Check status, it has to be 0xF8. (Read ROS_Base_Add + 0x3A0).

7.2 1-Wire Interface to the DS2438 through the DS2482.

7.2.1 Brief introduction to DS2482.

As we said before, in order to retrieve the information from the DS2438 sensor we will use the DS2482 I2C to 1-Wire (1-W) bridge.

As an I2C slave of the PCA9564, the DS2482 has a slave address set by three pins (AD0, AD1, AD2) that, in our case, are connected to ground. Therefore, its address value (*slave-add*) is 0x18 (see datasheet [7]).

From all the DS2482 functionality, only some commands will be necessary:

- "Channel Select" (command code 0xC3), used to select one among the 8 output channels.

- "Set Read Pointer" (command code 0xE1), needed for selecting one of the different internal registers that this chip has. The command must be followed by the code of the specific register which will be used in further commands. (see datasheet for codes of different registers). The execution of other commands may set the read pointer to a different value; therefore, it will be necessary to reassign its value, using this command, before executing another one.

- "1-Wire Reset" (command code 0xB4), used to start the transaction cycle specified by 1-W protocol. In fact, this command should be sent first, before all other commands sent to the DS2438.

- "1-Wire Read Byte" (command code 0x96), used to read 1 byte from the 1-W channel. Before doing this, previous commands should instruct 1-W slave IC to prepare information that is going to be read.

- "1-Wire Write Byte" (command code 0xA5), needed for writing 1 byte to the 1-W slave. It must be followed by the byte to be written in a unique transaction.

Many operations may modify one or more bits on the internal registers of DS2482, but we only need to pay special attention to two of them: "Status Register" and "Read data register".

The "Status Register" has the "1WS" bit (bit number 0) which indicates the status of the channel that is being used in a transaction. If this bit is HIGH, the last command executed is still being processed, so it's not possible to send nor receive data (even commands) to that channel. It's necessary to wait until this bit turns into LOW state (polling it) before executing new commands.

The "Read data register" is used to retrieve data read after a "1-Wire Read Byte" command.

7.2.2 Brief introduction to DS2438.

As DS2482, DS2438 has many commands and registers, but not all of them are needed for our purposes. Registers are organized in memory pages (9 bytes in length) instead of individually addressable bytes.

This forces to manage/retrieve complete pages before getting specific bytes, even significant bits. Doing this involves the management of an intermediate memory area, called scratchpad area (with corresponding scratchpad pages), where results must be copied from DS2438 internal registers before being read by the DS2482, or where configuration data must be written before being transferred to internal DS2438 registers.

The most important page is page number 0 that contains the status/configuration register, and six additional registers which store the current, temperature and voltage information (two bytes in length for each one) once the appropriated commands have been executed. So, in fact, we will be reading and writing this page every time. Its very recommended to see DS2438 datasheet to see the specific binary data format of current, voltage and temperature because not all register bits are in use.

Due to the importance of status/configuration register, we include some datasheet paragraphs describing those bits useful for us:

> The Status/Configuration Register is a non-volatile read/write byte which defines which features of the DS2438 are enabled and how they will function. The register is formatted as follows:
>
> X ADB NVB TB AD EE CA IAD
> MSb LSb
>
> IAD = Current A/D Control Bit. "1" = the current A/D and the ICA are enabled, and current measurements will be taken at the rate of 36.41 Hz; "0" = the current A/D and the ICA have been disabled. The default value of this bit is a "1" (current A/D and ICA are enabled).
>
> CA = Current Accumulator Configuration. "1" = CCA/DCA is enabled, and data will be stored and can be retrieved from page 7, bytes 4-7; "0" = CCA/DCA is disabled, and page 7 can be used for general EEPROM storage. The default value of this bit is a "1" (current CCA/DCA are enabled).

AD = Voltage A/D Input Select Bit. “1” = the battery input (VDD) is selected as the input for the DS2438 voltage A/D converter; “0” = the general purpose A/D input (VAD) is selected as the voltage A/D input. For either setting, a Convert V command will initialize a voltage A/D conversion. The default value of this bit is a “1” (VDD is the input to the A/D converter).

TB = Temperature Busy Flag. “1” = temperature conversion in progress; “0” = temperature conversion complete.

ADB = A/D Converter Busy Flag. “1” = A/D conversion in progress on battery voltage; “0” = conversion complete, or no measurement being made. An A/D conversion takes approximately 10 ms.

Bits TB and ADB can be either polled continuously until they are done (bit LOW) or just wait time enough to ensure that the process has finished.

The process of getting sensor information involves the whole command set available in the DS2438 so we reproduce here their command codes and their functionality. A more practical explanation of their use will be presented in certain useful routines explained in further paragraphs.

Command list:

- Write Scratchpad [4EhXXh]: This command writes to the scratchpad page XXh of the DS2438. The entire 8-byte scratchpad space may be written, but all writing begins with the byte present at address 0 of the selected scratchpad. After issuing this command, the user must send the page number of the scratchpad to be written; then the user may begin writing data to the DS2438 scratchpad. Writing may be terminated at any point by issuing a reset. Valid page numbers for writing are 00h-07h.

- Read Scratchpad [BEhXXh]: This command reads the contents of the scratchpad page XXh on the DS2438. After issuing this command, the user must send the page number of the scratchpad to be read, and then may begin reading the data, always beginning at address 0 of the selected scratchpad. The user may read up to the end of the scratchpad space (byte 07h), where reserved bits will be read as 1s. Then, it can read the data CRC, and after that, all bits read will be 1s. If not all locations are to be read, the master may issue a reset to terminate reading at any time. Valid page numbers are 00h – 07h.

- Copy Scratchpad [48hXXh]: This command copies the scratchpad page XXh into the EEPROM / SRAM memory page XXh of the DS2438. After issuing this command, the user must write a page number to set which page of memory the scratchpad is to be copied. Valid page numbers are 00h - 07h. During the copy function, the NVB bit in the Status/Configuration register will be set to a “l”. When the copy is completed, this bit will set to “0”. If the bus master issues read time slots following this command, the DS2438 will output “0” on the bus as long as it is busy copying the scratchpad to SRAM/EEPROM; it will return a “1” when the copy process is complete.

- Recall Memory [B8hXXh]: This command recalls the stored values in EEPROM / SRAM page XXh to the scratchpad page XXh. This command must precede a Read SPxx command in order to read any page of memory on the DS2438. Valid page numbers are 00h – 07h.

- Convert T [44h]: This command begins a temperature conversion. No further data is required. The temperature conversion will be performed, setting the TB flag in the Status/Configuration register to a “1” during conversion. When the temperature conversion is done, the TB flag will be set to “0”. If the bus master issues read time slots following this

command, the DS2438 will output “0” on the bus as long as it is busy making a temperature conversion; it will return a “1” when the temperature conversion is complete.

- Convert V [B4h]: This command instructs the DS2438 to initiate a voltage analog-to-digital conversion cycle. This sets the ADB flag (see Status/Configuration register discussion in the Memory Map section). The voltage supply that is measured is defined by the AD bit of the Status/Configuration register. When the A/D conversion is done, the ADB flag is cleared and the current voltage value is placed in the VOLTAGE REGISTER of page 00h. While an A/D conversion is taking place, all other memory functions are still available for use. If the bus master issues read time slots following this command, the DS2438 will output “0” on the bus as long as it is busy making a voltage measurement; it will return a “1” when the conversion is complete.

Besides, DS2438 supports four access methods: *READ-ROM*, *SEARCH-ROM*, *MATCH ROM*, and *SKIP-ROM*. The first three of them are only useful in case that a 1-W channel has 2 or more slaves connected (see [8]). At the ROS board this is not necessary as each DS2438 device has an independent connection to each of the channels of the DS2482 device (only one slave).

All accesses to the DS2438 consist of transactions beginning with a *RESET-PULSE* sent by the DS2482 master, followed by a DS2438 *SKIP-ROM* command, and a sequence of simple DS2438 commands or command+parameter. After sending each byte (command or parameter), a certain time has to be waited until the serial transmission in the 1-Wire has finished. In order to know when the 1-Wire is ready to send more data, polling can be performed to bit 0 of the DS2482 status register. When this bit is 0, the 1-W bus is ready for a new access.

7.2.3 Useful basic routines

In case they are useful for the user, in what follows we have included some routines for basic actions to be performed:

struct error_code DS2482_Wait_1W_stop (timeout):

1. Read system time
2. Read DS2482 status register: ***Read_PCA (slave-add, status-value)***
3. Verify if bit-0 (1-WS) of status-value is low. If true, exit.
4. If not, verify if actual system time minus former system time, exceeds timeout parameter. If true, exit with error.
5. If not, wait (for example) 1 ms, and return to step 2.

Another useful routine is responsible for *RESET-PULSE* and *SKIP-ROM* sequence:

void RP_SkROM ( ):

1. Command the DS2482 to send a Reset Pulse: ***Write_PCA (slave-add, 0xB4).***

2. Wait until command ends: ***DS2482_Wait_1W_stop ( )***

3. Send DS2438 *SKIP-ROM* command (0xCC code). To do this it's necessary to send to the DS2482 a *1-Wire-WRITE-BYTE* command (0xA5 plus byte-to-write):
   Write_PCA (slave-add, [0xA5, 0xCC])

4. Again, wait until command ends: ***DS2482_Wait_1W_stop ( )***

Since DS2438 internal registers are organized in memory pages (9 bytes in length), it is suitable to have a procedure to read a page in one step. The procedure steps are: first, transfer information from DS2438 memory space to its scratchpad area, and then, get data bytes from the scratchpad area. This is presented in the following routine:

array integer DS2438_read_mem_page (page_number)

1. As in other transactions, we begin with a *RESET-PULSE* and *SKIP-ROM* sequence:
   - ***RP_SkROM ( )***
2. Transfer memory-space data page to temporary (scratchpad) area. Achieving this requires to send the appropriate DS2438 command (0xB8) to the 1-W channel, and the page number of interest:
   - ***RP_SkROM ( )***
   - ***Write_PCA (slave-add, 0xB8)***
   - ***DS2482_Wait_1W_stop ( )***
   - ***Write_PCA (slave-add, page-number)***
   - ***DS2482_Wait_1W_stop ( )***
3. Next step is to point at the data bytes (scratchpad page) we want to get (code 0xBE followed by page number).
   - ***RP_SkROM ( )***
   - ***Write_PCA (slave-add, 0xBE)***
   - ***DS2482_Wait_1W_stop ( )***
   - ***Write_PCA (slave-add, page-number)***
   - ***DS2482_Wait_1W_stop ( )***
4. Finally, we should read the corresponding 9 data bytes. This action should be done right after previous step, so it is mandatory not repeating *RESET-PULSE* sequence. It involves

telling DS2482 to read a data byte from a 1-W channel (this reading automatically increments internal DS2438 memory pointer) and then get that information from DS2482 data register:

- Send DS2482 read-one-byte command: ***Write_PCA (slave-add, 0x96)***
- ***DS2482_Wait_1W_stop ( )***
- Point to DS2482 data register: ***Write_PCA (slave-add, [0xE1, 0xE1])***
- Read DS2482 data register: ***Read_PCA (slave-add, data-byte).***
- Repeat previous four steps up to nine times.

7.2.4 Detailed procedure to obtain sensor information.

Now, we proceed to explain how to read sensor's information, assuming that previous routines have already been developed.

1. First step is to select the DS2438 device to be read. To select the channel at DS2482, we have to send to the DS2482 command 0xC3 plus channel number (0xE1 for 5V sensor, 0xD2 for 3.3V sensor and 0xC3 for 1.8V sensor). Once selected, if we don't change channel number or don't reset DS2482, it remains selected, so latter communications will be done to that channel:

   Write_PCA (slave-add, [0xC3,0xE1]) ← for 5V sensor.

2. Next, it's necessary to read the status/configuration register to modify some of its bits to select or not current conversion at the DS2438 and to select the input pin where we want to read the voltage from. This register is located in byte number 0 from memory page 0:

   DS2438_read_mem_page(0)

3. Set IAD bit (bit-0 DS2438 configuration register) high to perform current conversion. Set VAD bit (bit-3 DS2438 configuration register) high for voltage measurement from VDD pin or low for voltage measurement from DS2438 chip's VAD pin.

4. Send command for writing data in page number 0. This is done by sending 3 bytes:

   - DS2438 command for writing page,
   - page number,
   - data byte to be written.

   Remember that writing a byte to the 1-Wire bus is equivalent to send DS2482 *1-Wire-WRITE-BYTE* command (0xA5 plus byte-to-write).

Actually, the operation performed by this transaction is writing data to a so-called "scratchpad page", a kind of temporary data buffer into DS2438. After that, it's required to transfer that page into the memory space.

So, the overall procedure will be:

(Write to scratchpad page)

- *RP_SkROM ( )*
- *Write_PCA (slave-add, [0xA5,0x4E])*
- *DS2482_Wait_1W_stop ( )*
- *Write_PCA (slave-add, [0xA5,0x00])*
- *DS2482_Wait_1W_stop ( )*
- *Write_PCA (slave-add, [0xA5, value-of-control-register-modified])*

(Transfer page into memory)

- *RP_SkROM ( )*
- *Write_PCA (slave-add, [0xA5,0x48])*
- *DS2482_Wait_1W_stop ( )*
- *Write_PCA (slave-add, [0xA5,0x00])*

5. Next step is to command DS2438 to perform temperature (code 0x44) and voltage/current (code 0xB4) measurements.

   (Convert temperature)

   - *RP_SkROM ( )*
   - *Write_PCA (slave-add, [0xA5,0x44])*

   (Convert voltage/current)

   - *RP_SkROM ( )*
   - *Write_PCA (slave-add, [0xA5,0xB4])*

6. Now it is necessary to wait enough time so voltage and temperature conversion are done. It is possible to poll ADB and TB status bits so, when they are both low, we ensure that conversion has finished, but it is more simple time waiting and save accesses to the device.

7. Then, we retrieve DS2438 memory page number 0 to get bytes 1 (LSB) and 2 (MSB) with temperature information, bytes 3 (LSB) and 4 (MSB) with voltage information, and 5 (LSB) and 6 (MSB) with current information. Remainder bytes are discarded:

   DS2438_read_mem_page(0)

8. Finally, we will convert binary bytes into proper units.

$$T = (-1)^{bit15\_of\_byte\_2}\left(\frac{byte\_1}{256} + byte\_2\right)^{o}C$$

$$V = \frac{(256 \times byte\_4 + byte\_3)}{100} volts$$

$$I = \frac{1.606 \cdot 10^{-4}}{R_{sens}} \cdot (-1)^{bit15\_of\_byte\_6}(256 \times (byte\_6) + byte\_5)A$$

At ROS prototypes $R_{sens} = 0.033\ \Omega$

8 INTERFACE TO THE GOL.

The GOL (Gigabit Optical Transmitter) is an ASIC developed at CERN Microelectronics group [4] that serialises the 16 bit word data and drives an optical laser (VCSEL HFE4190-541) for transmission through optical fibber to the next step in the read-out chain, the DDU boards.

To avoid over-currents on the VCSEL transmitter at start-up, the GOL and the VCSEL are powered off by default. Therefore, when the link is going to be used, they should be turned on by writing a 0 in bit 2 of register "I2C & 1-wire & GOL & QPLL (ROSVME+0x12)".

In that register there is also a flag that indicates the status of the GOL (GOL ready (bit 3) and GOL not ready registered (bit7)). If at any time the GOL has not been ready, the GOL not ready registered flag will keep a 1, although the GOL has recovered from its error and GOL is again ready (bit 3 set to 1). This registered flag should be erased after turning on the GOL in order to reflect further problems.

It is recommended to wait around 800 ms between turning on the GOL and erasing the GOL not ready registered flag, to insure that the GOL is in the ready state. Note that the turning on of the GOL includes the power up time of the 2.5V regulator, the QPLL initialization time and the GOL initialization time.

Besides, the GOL internal registers can be accessed for configuration and monitoring through an I2C interface. This interface has been explained in previous section and the routines Write_PCA and Read_PCA can be used as detailed before.

Note: At the moment an error in the status words read from the PCA9564 has been found if the Read_PCA and Write_PCA procedure is followed. It is due to strange behaviour of the GOL I2C interface that is under study at the moment. However, the procedure as it is described works fine and access to the GOL registers can be done properly, the only difference is that status words received from the PCA9564 are not the ones indicated.

For what concerns to the I2C address of the GOL, it occupies two consecutive positions in the 7-bit wide I2C address space.

I2C access register name	I2C address
I2C_pointer	0
I2C_data	1

Data written in the first address (0) is a pointer to the corresponding GOL internal register. Therefore, at that address it should be written any of the 6 possible values of the GOL internal registers:

GOL internal registers address (I2C_pointer content)	Internal Register	Default content
0	Config 0	0x33
1	Config 1	0x1F
2	Config 2	0x10
3	Config 3	0x20
4	Status 0	0x00
5	Status 1	--

Depending on which value has been written to the I2C_pointer register, the corresponding content of the GOL internal register can be written or read from the I2C_data register.

The main parameter that may be desired to modify at the GOL is the VCSEL bias current, which can be done in register Config 3:

GOL Internal Register Config 3

Bits	Name	Description
<6:0>	LD_current/driver_strength	Defines the bias current for the Laser driver.
<7>	use_conf_regs	When 1, the content of the Configuration register 2 and 3 are used to define the value for the PLL_current and LD_current. If 0, the values are derived from the encoded values on the pads (0x00 for LD_current)

The number in bits Config3<6:0> translates the laser-diode bias current according to:

$$I = 1\ mA + Config3<6:0> \times 0.4\ mA$$

By default, the bias current is 1 mA, and the maximum bias current should be kept below 12mA, **therefore a value over 0x1B should not be programmed at Config3<6:0>**.

Also, could be interesting to read the value of the status registers:

GOL Internal Register Status 0

Bits	Name	Description
<7:0>	loss_of_lock_count	Number of "loss-of-lock"events since last reset.

GOL Internal Register Status 1

Bits	Name	Description
<7:6>	link_control_state_A	Current state of link initialisation logic A.
<5:4>	link_control_state_B	Current state of link initialisation logic B.
<3:2>	link_control_state_C	Current state of link

		initialisation logic C.
<1:0>		Set by hardware to “01”

For SEU robutness, the link initialisation logic is triplicated. The possible status are:

00: Out of lock state
01: Locked state
10 REady state
11 TX_lolc state.

For more information refer to the GOL user manual [4]

8.1 Procedure for accessing the GOL.

According to what explained before, the procedure to read or write from the GOL internal registers can be summarized as follows:

WRITE TO A GOL INTERNAL REGISTER:

1. Enable GOL I2C access at the ROS (write a 1 on bit 0 of ROS_Base_Add + 0x392).
2. Write GOL internal register address to the GOL I2C_pointer:
   Write_PCA (p_add = 0, p_data = GOL internal register address)
3. Write GOL internal register data to the GOL I2C_data:
   Write_PCA (p_add = 1, p_data = GOL internal register data)

READ FROM A GOL INTERNAL REGISTER:

1. Enable GOL I2C access at the ROS (write a 1 on bit 0 of ROS_Base_Add + 0x392).
2. Write GOL internal register address to the GOL I2C_pointer:
   Write_PCA (p_add = 0, p_data = GOL internal register address)
3. Read GOL internal register data from the GOL I2C_data:
   Read_PCA (p_add = 1, p_data = GOL internal register data)

9 ROS-25 DATA FORMATS

When the ROS receives the data from the ROBs, it stores it in one 4 kBytes FIFO per channel, and at the reception of an L1A signal, it starts to read the data from the FIFOs and send it to a common bus to the GOL serialiser.

In principle, ROS will be operated in normal mode, with a TIM module in the same crate and receiving the L1A signal from the TIM through the custom backplane.

In general, at the reception of a L1A, the ROS will send a ROS header (called "Event header") that will include the 24 bits of the ROS event ID. After that, the ROSCTRL module will start a read-out mechanism passing the token to each CEROS using a star topology. After each CEROS has performed the read-out of its channels, they will return the token to the ROSCTRL that will send it to the following CEROS.

It is important to know that at each CEROS, unless this feature is disabled, all the data packets that only contain headers and trailers and not time information or errors signalling will be discarded, in order to reduce the total throughput.

The data coming from the ROBs will be modified at each CEROS, in order to include other necessary information.

9.1 ROS-25 Control words

9.1.1 Event Header from ROS

Event header: event header from ROS

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ROS	0	0	0	1	1	1	1	1	TTC Event Counter																							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

This event header encloses all the data belonging to one event. In this word, it is included the event ID of the event that is being processed, either it comes from the TIM module or from the ROS internal event counter.

9.1.2 Event trailer from ROS

Event trailer: event trailer from ROS

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ROS	0	0	1	1	1	1	1	1	TFF	TXP	ECHO		ECLO		BCO		Event Word count (words of 32 bits)															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Here there is also a word count field for all the 32 bit words of the processed event, including ROS event header and trailer. The other fields in this word are:

- **TFF**: When it is 1, it indicates that any of the L1A FIFOs (Bunch counter, Event counter high or low) has been full at any time. These FIFOs store the number of events that are pending to be processed.

- **TXP**: transmitter parity error. This bit is set to 1 if the number of 16 bits words transmitted to the GOL is odd, i.e., the ROS is not working fine.

- **ECHO**: These two bits are the two higher bits (7 and 6) of the value of the occupancy of the Event Counter High FIFO, which stores the higher 11 bits of the event ID only in the case it has been received from the TIM.

- **ECLO**: It is the same that the ECHO bits but for the FIFO that stores the lower 11 bits of the event counter.

- **BCO**: It is the same that the ECHO bits but for the FIFO that stores the bunch counter.

It is important to take into account that when the L1A is not received from the TIM module, the TFF, ECHO, ECLO and BCO fields have no sense. This also implies, as said before, that the L1A rate should be slow enough to allow the ROS to process each event before receiving a new one, as overlapping triggers are not allowed in this mode.

9.1.3 Error flags from ROS

Errors: error flags sent when error condition is detected

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ROS	1	1	0	1	1	1	1	1	Error type			Link/ROB ID					1															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Error types:

	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Link timed out			ROB ID					1										EVID mis	PAF	EF	LK	HU	FF
0	Sector Collector time out			25					1												EF			FF
1	Event number misalignment			ROB ID					1				Event ID at ROS											
2	Channel FIFO almost full			ROB ID					1										EVID mis	PAF	EF	LK	HU	FF
3	Channel FIFO full			ROB ID					1										EVID mis	PAF	EF	LK	HU	FF
3	Sector Collector FIFO full			25					1												EF			FF
4	Ceros timed out			31					1										Ceros ID					
5	Max number words			ROB ID					1										EVID mis	PAF	EF	LK	HU	FF
6																								
	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits 16 to 20 indicate the number of the channel that produced the error and bits 21 to 23 specify the type of error that is being reported, it can be any of the following:

- **Link timed out**: Each FPGA device will wait a programmable time when it tries to read an input FIFO but that channel is still empty. After that time, if the channel FIFO is still empty, an error word will be generated and the channel may be blocked.

As in other error words, all the status flags corresponding to that channel will be included in the error word:

- **PAF**: Indicates that the FIFO is over the almost full limit programmed.
- **EF**: Indicates that the FIFO is empty
- **LK**: Indicates if the channel is locked at that moment.
- **HU**: Indicates that the channel has been unlocked at any moment.
- **FF**: Indicates that the FIFO has been full at any time.
- **EvID mis**: Indicates that at some point the event ID read at this channel did not match the ROS event ID.

If the channel that has timedout or which FIFO is full is the Sector Collector (Channel ID = 25), only EF and FF flags will be included in the error word.

- **Event number misalignment:** In the case that the event ID received from the TIM does not match the event ID that is included in the TDC group headers, the ROS generates an error word. Note that the events are not re-aligned, as the misalignment may come from a wrong number in the event field while the data may really belong to this event. Only a track fitting can really determine to which event the data belongs to.

- **Fifo full:** This word is generated when any channel FIFO is full. If this happens, further words received will be lost.

- **Ceros timed out**: It may happen that a full group of 6 channels (a CEROS) is not working properly. If it keeps the token longer than a programmable time, it may be disabled and an error word will be generated. The timeout controller is reset when the CEROS starts to read a new channel, therefore, the timeout value should be larger than the expected time to process an event in one channel.

- **Max number of words**: This error word is generated when the ROS is reading one channel and reads out more than 200 words from a FIFO without finding a ROB trailer. If this happens, it is considered that either the data stored at that FIFO is not valid or that this channel has too much noise.

9.2 ROS-25 Debugging data

Debugging data

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ROS	1	1	1	1	1	1	1	1	Debug type			31					1															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Debug types:

	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Bunch number			31					1				TTC Bunch Counter											
1	BcntResCntLow			31					1	Bcnt_res Counter Low														
2	BcntResCntHigh			31					1															Bcnt_res Counter High
	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

This information is to be used for debugging the ROS or the full system; in principle, is not necessary to use it in normal mode. These words can be differentiated from the TDC debugging

data because bit 15 is set to 1. Bits 23 to 21 indicate the type of debug word that is going to be sent, the possibilities are:

- **Bunch number**: Bits 11 to 0 will include the Bunch ID received from the TIM through the backplane.

- **Bcnt_ResCnt High and Low**: The Bcnt_res Counter Low field contains bits 14 to 0 of the number of bunch counter resets that have been sent. Bcnt_res Counter High contains bit 15.

9.3 Generated at HPTDC, modified by ROS-25.

Group header: event header from master TDC (one per ROB)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDC	0	0	0	0	TDC ID				Event ID												Bunch ID											
ROS	0	0	0	ROB ID (0-24)					Event ID												Bunch ID											
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Group trailer: event trailer from master TDC (one per ROB)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDC	0	0	0	1	TDC ID				Event ID												Word count (32 bits words)											
ROS	0	0	1	ROB ID (0-24)					Event ID												Word count (32 bits words)											
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

TDC header: event header from TDC (master and slaves)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDC	0	0	1	0	TDC ID				Event ID												Bunch ID											
ROS	0	1	0	0	PC	PAF	TDC ID		Event ID												Bunch ID											
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

TDC trailer: event trailer from TDC (master and slaves)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDC	0	0	1	1	TDC ID				Event ID												Word count (32 bits words)											
ROS	0	1	1	0	PC	PAF	TDC ID		Event ID												Word count (32 bits words)											
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Leading measurement: single edge (normal time measurement)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDC	0	1	0	0	TDC ID				Channel					Leading time																		
ROS	1	0	0	0	PC	PAF	TDC ID		Channel					Leading time																		
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Leading measurement: combined measurement of leading and trailing edge

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDC	0	1	0	0	TDC ID				Channel					Width							Leading time											
ROS	1	0	0	0	PC	PAF	TDC ID		Channel					Width							Leading time											
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Trailing measurement

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDC	0	1	0	1	TDC ID				Channel					Trailing time																		
ROS	1	0	1	0	PC	PAF	TDC ID		Channel					Trailing time																		
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Errors: error flags sent when error condition is detected

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

TDC	0	1	1	0	TDC ID														Error flag													
ROS	1	1	0	0	PC	PAF	TDC ID												Error flag													
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Debugging data: separator

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDC	0	1	1	1	TDC ID				0	0	0	0									Bunch ID											
ROS	1	1	1	0	PC	PAF	TDC ID		0	0	0	0									Bunch ID											
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Debugging data: L1 buffer occupancy

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDC	0	1	1	1	TDC ID				0	0	0	1											GR		L1 occupancy							
ROS	1	1	1	0	PC	PAF	TDC ID		0	0	0	1											GR		L1 occupancy							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Debugging data: trigger and readout FIFO occupancy

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDC	0	1	1	1	TDC ID															Trigger fifo				F	Read-out fifo							
ROS	1	1	1	0	PC	PAF	TDC ID													Trigger fifo				F	Read-out fifo							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

As can be seen, the identification of the TDC Master that was in the HPTDC data has been replaced by the information of the number of link (ROB) that the data belongs to (links from 0 to 24). Moreover, the Sector Collector data is treated as if it was another link, link number 25.

The remaining fields have been left basically as they came from the HPTDC, so we will just make a few comments on the main differences:

- **PC**: (Parity check). The parity, calculated over each byte is transmitted from the ROB within the data flow, and it is stored in the input fifo at the ROS. The FPGA device checks if the parity is correct over the four bytes of each 32 bits word and signals it with a 0 if it is correct. If the bit is 1, this 32 bit word should be discarded as any or more bits might have flipped.

- **PAF**: (Programmable Almost Full). If this bit is set to 1, the FIFO at that particular channel has less empty words than the programmed value. By using this flag, the DAQ system can be informed that a FIFO is almost full.

9.4 Sector Collector Trigger Data

9.4.1 Sector Collector header.

Sector Collector header

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ROS	0	0	0	25					ROS Event ID															SC fifo occupancy (16 bits words)								
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

The SC header includes the least significant 11 bits of the ROS event ID. It also includes information of the occupancy of the SC fifo located at the ROS, where the data received from the contiguous SC board is stored.

9.4.2 Sector Collector trailer

Sector Collector trailer

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ROS	0	0	1	25													Word count (words of 32 bits)															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

The SC trailer includes a 16 bit word count of the number of SC 16 bits words that have been transmitted. The expected number of 16 bit words to be transmitted per event is ~ 80.

9.4.3 Sector Collector data

Sector Collector Data

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ROS	1	0	0	25													SC Data															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Finally, the SC data is transmitted in the 16 least significant bits of the SC data packet.

10 APENDIX A: DATA GENERATED AT THE HPTDC (ROB).

As described before, each ROB has 4 HPTDCs in a token ring, where one of them has been configured as Master. This HPTDC will control the token of the read-out ring and is the one in charge of generating a group header and trailer that will enclose the time information generated by all HPTDCs in the ROB.

This header and trailer will contain information that allows identifying the event number and the bunch crossing the data belongs to. This information comes from internal counters in the HPTDC that are reset by signals from the TTC system, and not directly from TTC devices. These counters are 11 bits long, so their possible values are between 0 and 4095.

Typically, the HPTDCs are configured to provide the time information of the leading edge of the signals within a programmable window. This information corresponds to the leading measurement word [page 22 ref 4] and contains information of the TDC number in the ROB (from 0 to 3) and the channel that received the signal (from 0 to 31). It is important to take into account that the two least significant bits of the leading time field are not used in low resolution mode, so the value should be divided by 4 in order to obtain correct timing information. Therefore, calculating the time value of the measurement can be easily made by applying:

$$\frac{(Leading\_time)}{4} * \frac{25}{32}\, ns$$

When any HPTDC detects an error condition, i.e. buffer overflow, it signals it to the CCB board and also, it is programmed to send an error word within the data flow. The possible error flags are described in page 23 of reference [4].

Although it is not probable to use it in normal operation mode, the HPTDCs can be programmed to provide other information words such as:

- trailing measurement, where it is digitalized the time information of the falling edge of the signal,
- combined measurement of leading and trailing edge, where the obtained information is the leading time and the width of the input pulse,
- Local TDC headers and trailers, where all the HPTDCs will give headers and trailers besides the master header and trailer. In this mode, the throughput of the link is largely increased.
- and other debugging information that show the occupancies of the different FIFOs in the HPTDC.

The first four bits from each 32 bit TDC word identify the type of word generated: header, leading measurement, error word, etc.

A snap of the data flow from a ROB could be something like this:

```
.....
TDC Header: TDC ID=3 Event ID=18 Bunch ID=2186
        Time measurement: TDC ID=0 Channel=4 Time=158ns
        Time measurement: TDC ID=0 Channel=14 Time=152ns
        Time measurement: TDC ID=0 Channel=7 Time=235ns
        Time measurement: TDC ID=0 Channel=22 Time=267ns
        Time measurement: TDC ID=0 Channel=4 Time=523ns
        Time measurement: TDC ID=1 Channel=11 Time=85ns
        Time measurement: TDC ID=1 Channel=25 Time=411ns
        TDC Error: TDC ID=1 Error flag=Internal fatal error
        Time measurement: TDC ID=2 Channel=6 Time=147ns
        Time measurement: TDC ID=3 Channel=6 Time=54ns
        Time measurement: TDC ID=3 Channel=27 Time=81ns
        Time measurement: TDC ID=3 Channel=3 Time=347ns
        Time measurement: TDC ID=3 Channel=19 Time=389ns
TDC Trailer: TDC ID=3 Event ID=18 Wordcount=15

TDC Header: TDC ID=3 Event ID=19 Bunch ID=642
        Time measurement: TDC ID=0 Channel=4 Time=167ns
        Time measurement: TDC ID=0 Channel=14 Time=162ns
        Time measurement: TDC ID=0 Channel=9 Time=255ns
.....
```

It can be seen that after a header, the first data received belong to TDC 0, afterwards TDC 1, etc. Nevertheless, inside each TDC it is not in strict temporal order and not even follows a fixed channel number order. The reason of this due to the way the different read-out FIFOs are organised inside the HPTDC and the arbitration mechanism that is used to perform the trigger matching. For more information, refer to HPTDC manual.

In MB1 Minicrates, due to the little space available, one of the ROBs is a ROB-32, that is, it only has one HPTDC. This HPTDC is configured as master and will provide master headers and trailers, but usually will be configured with a TDC ID = 0.

11 REFERENCES

[1] "High Performance Time to Digital Converter". Version 2.1. J. Christiansen. CERN/EP – MIC. July 2002. Accessible from:
http://micdigital.web.cern.ch/micdigital/hptdc/hptdc_manual_ver2.1.pdf

[2] TTC system. http://ttc.web.cern.ch/TTC/intro.html

[3] PCA9564 datasheet. http://www.semiconductors.philips.com/pip/PCA9564N.html

[4] GOL user manual. http://proj-gol.web.cern.ch/proj-gol/golManuals.htm

[5] TIM User manual. http://wwwae.ciemat.es/cms/DTE/idoc.htm#TIM

[6] TTCrx User Manual. http://ttc.web.cern.ch/TTC/TTCrx_manual3.11.pdf

[7] DS2482-800 datasheet: http://pdfserv.maxim-ic.com/en/ds/DS2482-800-DS2482S-800.pdf

[8] DS2438 datasheet: http://pdfserv.maxim-ic.com/en/ds/DS2438.pdf