Ciemat Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas

TIM User Manual

Version 2.0

J.M. Cela, C. Fernández, C. Willmott.

Electrónica y Automática.
CIEMAT.

October 22th, 2006

.

INDEX

1 Introduction 3
2 VME Interface 6
2.1.1 Control & Status. Register $00 6
2.1.2 Interruptions. Register $02 6
2.1.3 TTCrx and QPLL Control and Status. Register $40 7
2.1.4 Individually Addressed Command. Register $42 7
2.1.5 Broadcast data (BRCST_STR1). Register $44 7
2.1.6 Bunch Counter. Register $46 8
2.1.7 Event Counter Low. Register $48 8
2.1.8 Event Counter High. Register $4A 8
2.1.9 QPLL Control & Status. Register $4C 8
2.1.10 DOUT for DQ = "0001" or "0101". Register $4E. 8
2.1.11 DOUT for DQ = "0010" or "0110". Register $50. 8
2.1.12 DOUT for DQ = "0011" or "0111". Register $52. 9
2.1.13 DOUT for DQ = "0100" or "1000". Register $54. 9
2.1.14 DOUT for DQ = "1001". Register $56. 9
2.1.15 DOUT for DQ = "1010". Register $58. 9
3 Voltage, Current and Temperature sensors. 10
3.1 I2C interface through the PCA9564 11
3.1.1 I2C (sensor) Status. Register $80 11
3.1.2 I2C (sensor) Data. Register $82 11
3.1.3 I2C (sensor) Address. Register $84 11
3.1.4 I2C (sensor) Control. Register $86 11
3.1.5 I2C (TTCrx) Status. Register $88 11
3.1.6 I2C (TTCrx) Data. Register $8A 11
3.1.7 I2C (TTCrx) Own address. Register $8C 11
3.1.8 I2C (TTCrx) Control. Register $8E 12
3.1.9 Write_PCA(out p_add, out p_data) 12
3.1.10 Read_PCA (out p_add, in p_data) 13
3.2 1-Wire Interface to the DS2438 through the DS2482. 14
3.2.1 Brief introduction to DS2482 and DS2438 ICs. 14
3.2.2 Useful basic routines 16
3.2.3 Detailed procedure to obtain sensor information. 18
4 Interface to the TTCrx 21
5 Configuration and operation 22
6 References 23

1 Introduction

The TIM board is a 9U VME board used for interface between the TTC system and the Sector Collector crate, that is, the Sector Collector boards and the ROS-25 boards.

It is made up of different boards:

1) A TTCrq board, which contains the optical receiver for the TTC optical link and a TTCrx ASIC that decodes the TTC signals.

2) A TIM-TTC board that receives the signals from the TTCrx and stores the necessary information that can be read from VME accesses. It also allows configuration of the TTCrx through an I2C interface.

3) A TIM-VME board which mainly handles the VME access.

4) A TIM-LED board for visualization of the TIM board status.

In the following pages a picture of the board with its main elements is shown, and it is also presented the front-panel of the TIM with the meaning of the different leds.

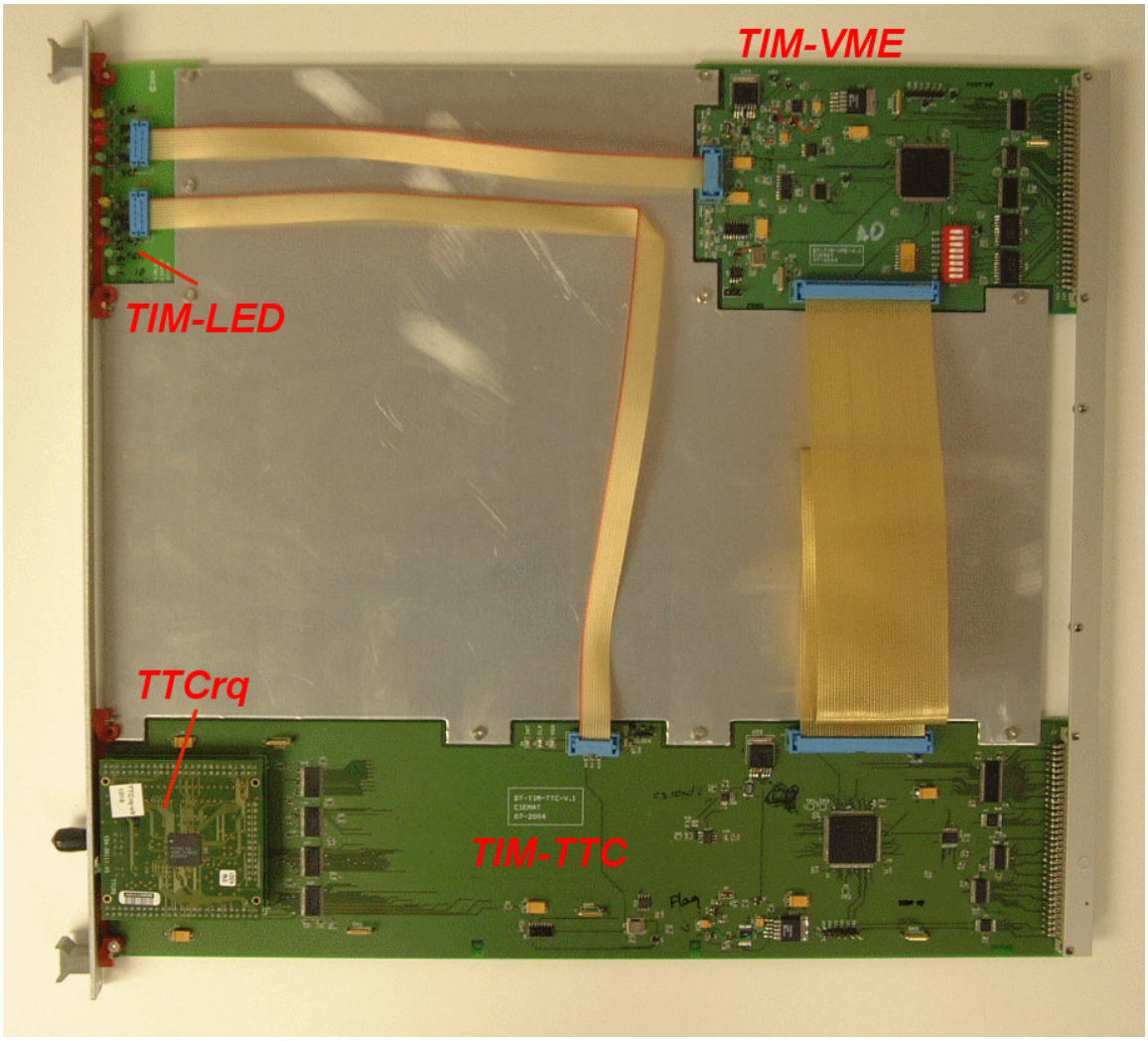
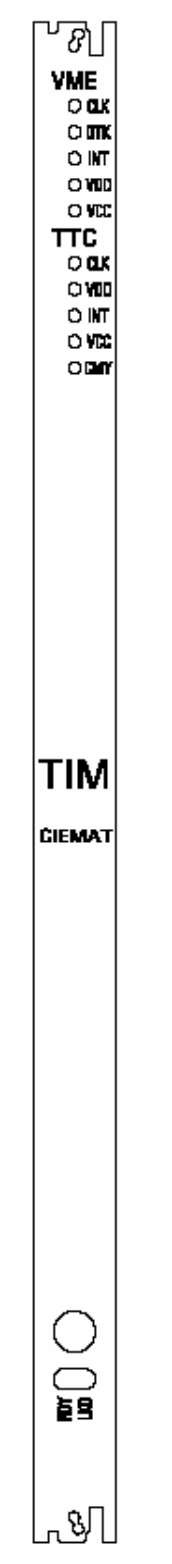


Figure 1: TIM board with its main components.



VME

CLK: ON = clock is received from TIM-TTC.
DTK: ON = VME data transfer acknowledge.
INT: ON = interruption requested.
VDD: ON = 3.3V power on.
VCC: ON = 5V power on.

TTC

CLK: ON = clock is received from the TTCrq board.
VDD: ON = 3.3V power on.
INT: ON = interruption requested. It will only generate a VME interruption if they have been enabled.
VCC: ON = 5V power on.
CMY: spare.

TTC optical fibre input.

RDY: ON = TTCrx ready.
LKD: ON = QPLL locked.

The QPLL is only locked at the LHC frequency. Note that this is not the frequency that the present TTCex module generates.

Figure 2: TIM front pannel.

2 VME Interface

The VME interface to the TIM board is performed through standard A16 access 16 bit words. The address selection is done through switch S1.

In what follows it is explained the content of the different registers and it is also indicated to which board (TIMVME or TIMTTC) corresponds each of the registers. The default value corresponds to the expected proper value read after a reset.

2.1.1 Control & Status. Register $00

(TIMVME). Default value = 0x04

0	enable interrupt	R/W	If set to 1, VME interruptions are allowed (interruption level has to be different from 0). After an acknowledge, it will have to be enabled again. When enabled, an interruption will be generated when there is a TIMTTC board error.
1	interrupt requested	R	=1 VME interruption has been requested.
2	ttc_on	R	=1 means that the TIMTTC boad is connected and powered.
3	Was_off	R/W	=1 means that at some point the TIMTTC board has been disconnected. It can be reseted by writing here a 0 or with a global board reset.
4	TIMTTC board error	R	=1 if the TIMTTC board detects an error in one of this signals: single error and double error from the TTCrx, QPLL unlock, QPLL SEU error or TTCrx not ready.
5	I2C int line	R	"Int" line from the PCA9564 device.
6	Reset I2C device	W	=1 resets the PCA9564 chip present at the TIMVME board (for sensor reading). The PCA9564 chip is also reseted with a global board reset.
7	Local_clk	R	=1 internal clock from the board is being used (no clock received from the TIMTTC)
8	Local_clk_reg	R/W	=1 at any time the clock from the TIMTTC board has been disconnected.
15	Global TIM reset	W	=1 performs a reset of all the registers of TIMVME, TIMTTC and the 2 PCA9564 modules. It does not reset the TTCrq board (nor the TTCrx nor the QPLL).

A VME sys-reset will force also a global TIM reset.

2.1.2 Interruptions. Register $02

(TIMVME). Default value = 0x100

0-7	Interrupt Vector	R/W	Any value
8-10	Interrup Level (1-7)	R/W	Any value from 1 to 7.

The TIMTTC board will flag an error signal to the TIMVME in any of the following circumstances:

- A single error or double error is received in the TTC data stream.
- The QPLL is unlocked.
- The QPLL has a SEU error.
- The TTCrx is not ready.

Whenever any of these situations take place, the INT led at the TIMTTC will turn on. Besides, if interruptions are enabled (bit 0 of register $00 to 1), these situations will also generate a VME interruption with the level written at register $02. In such a case, led INT at TIMVME will also turn on, and bit 1 of register $00 will be set to 1 until the interruption is acknowledged.

2.1.3 TTCrx and QPLL Control and Status. Register $40

(TIMTTC)

0	TTCrx ready	R	=1, the TTCrx is connected and ready. It is ready although the QPLL is not locked.
1	QPLL locked	R	=1 the QPLL is locked. If = 0, the clock at the TIM_TTC and at the backplane will not be synchronaised with the TTC signals (L1A, event ID,etc) and they will be wrongly detected.
2	QPLL_unlocked_registered flag	R/W	=1 if at any point the QPLL has been unlocked.
3	sinerr_str	R	=1 Single error in the TTCrx (See TTCrx manual).
4	Dberr_str	R	=1 Double error in the TTCrx. (See TTCrx manual).
5	sinerr_str_reg	R/W	Registered values of the previous.
6	Dberr_str_reg	R/W	Registered values of the previous.
9	TTCrx_not_ready_registered flag	R/W	If at any time the TTCrx is not ready, it is registered in this bit. After a TTCrx reset, this bit should be erased by writing a 0.
14	Resets I2C device for TTCrx access.	W	It resets the PCA9564 device at TIM-TTC. This device is also reset with a global board reset.
15	TTCrx reset	W	When writing a 1, a reset is sent to the TTCrq board. (Reset is 51 μs long)

2.1.4 Individually Addressed Command. Register $42

(TIMTTC). Default value = 0x00

0-7	Dout(7:0)	R/W
8-15	Subaddress(7:0)	R/W

Through the TTC system, individually addressed commands (IAC) can be sent to each of the subsystems. In this register those IAC commands will be stored, although no action is taken at the moment. Note that in order to allow these commands to exit the TTCrx and therefore, be registered here, bit 5 (“Enable Parallel output bit”) has to be enabled at the TTCrx control register.

2.1.5 Broadcast data (BRCST_STR1). Register $44

(TIMTTC). Default value = 0x00

0-1	00	R
2-7	Broadcast command	R

Bits7-2 of each TTC broadcast command are stored in this register. The strobe signal corresponds to BRCST1, that is, synchronous with ckdesv1 from TTCrx.

Bits 5 to 2 of the broadcast command, synchronous to ckdesv1, exit (25ns width pulse) through the command lines (cmd) to the TIMBUS backplane.

Broadcast bit 2	cmd(0)
Broadcast bit 3	cmd(1)
Broadcast bit 4	cmd(2)
Broadcast bit 5	cmd(3)

2.1.6 Bunch Counter. Register $46

(TIMTTC) Default value = 0x00

0-11	Bunch counter	R

It stores the Bunch counter value sent from the TTCrx after each L1A.

2.1.7 Event Counter Low. Register $48

(TIMTTC). Default value = 0xFFF

0-11	Event counter low	R

It stores the 12 least significant bits of the Event counter value sent from the TTCrx after each L1A. After an Event counter reset, the first L1A is signalled with an Event counter value of 0.

2.1.8 Event Counter High. Register $4A

(TIMTTC). Default value = 0xFFF

0-11	Event counter high	R

It stores the 12 most significant bits of the Event counter value sent from the TTCrx after each L1A. After an Event counter reset, the first L1A is signalled with an Event counter value of 0.

2.1.9 QPLL Control & Status. Register $4C

(TIMTTC). Default value = 0x01

0	QPLL locked	R	= 1 the QPLL is locked (Same as register $40).
1	QPLL_unlocked_registered flag	R	= 1, at any time it has been unlocked. (Same as register $40).
2	QPLL error (SEU)	R	1 = an SEU error from the QPLL
3-10	SEU counter	R	It counts the amount of SEU errors from the QPLL. This counter is reseted by writing a 1 in the Reset SEU counter and also by a global TIM reset.
14	Reset SEU counter	W	
15	Reset QPLL	W	When writing a 1 only the QPLL at the TTCrq is reseted, but not the TTCrx device.

The QPLL is only locked at the exact LHC frequency. Note that this is not the one that provides the TTCex module.

2.1.10 DOUT for DQ = “0001” or “0101”. Register $4E.

(TIMTTC). Default value = 0x00

0-7	Dout (7:0)	R
8-11	DQ (Data Qualifier)	R

2.1.11 DOUT for DQ = “0010” or “0110”. Register $50.

(TIMTTC). Default value = 0x00

0-7	Dout (7:0)	R
8-11	DQ (Data Qualifier)	R

2.1.12 DOUT for DQ = “0011” or “0111”. Register $52.

(TIMTTC). Default value = 0x00

0-7	Dout (7:0)	R
8-11	DQ (Data Qualifier)	R

2.1.13 DOUT for DQ = “0100” or “1000”. Register $54.

(TIMTTC). Default value = 0x00

0-7	Dout (7:0)	R
8-11	DQ (Data Qualifier)	R

2.1.14 DOUT for DQ = “1001”. Register $56.

(TIMTTC). Default value = 0x90

0-7	Dout (7:0)	R
8-11	DQ (Data Qualifier)	R

2.1.15 DOUT for DQ = “1010”. Register $58.

(TIMTTC). Default value = 0xA0

0-7	Dout (7:0)	R
8-11	DQ (Data Qualifier)	R

These 6 registers are used to store the data generated at the TTCrx after an ERDUMP or CRDUMP command. These data are related to the internal registers of the TTCrx. (See page 22 on the TTCrx user manual).

3 Voltage, Current and Temperature sensors.

At the TIM board, there are some sensors that provide the voltage, current and temperature values. At TIMVME there are two sensors, one of them to provide the 3.3V current consumption and another one for the 5V current consumption. At TIMTTC there is one sensor and only 5V current can be read.

These sensors are 1-wire DS2438 devices, that can be accessed through an I2C to 1-wire interface (DS2482 device). The DS2482 can interface with three different 1-wire devices, according to the selected channel. In the TIMVME board, the mapping of the channels is the following: channel 0 and channel 1 are connected to the DS2438 sensors at TIMVME and channel 2 corresponds to the sensor located at TIMTTC.

In the following table it is indicated the values that can be read at each of the DS2438 devices, and in figure 3 it is presented a diagram of the different interconnections.

TIMVME 3.3V current, channel 0
5V voltage at TIMVME (VDD input of the DS2438)
3.3V voltage at TIMVME (VAD input of the DS2438)
3.3V current consumption at TIMVME.
Temperature at TIMVME.
TIMVME 5V current, channel 1
5V voltage at TIMVME (VDD input of the DS2438)
3.3V voltage at TIMVME (VAD input of the DS2438)
5V current consumption at TIMVME.
Temperature at TIMVME.
TIMTTC, channel 2
5V voltage at TIMTTC(VDD input of the DS2438)
3.3V voltage at TIMTTC (VAD input of the DS2438)
5V current consumption at TIMTTC.
Temperature at TIMTTC.

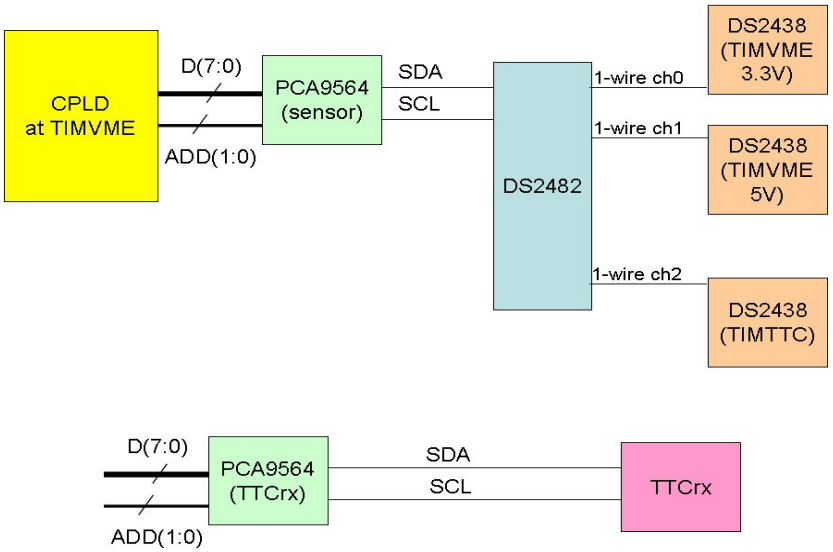


Figure 3: Diagram of the interface to the sensor devices and to the TTCrx.

3.1 I2C interface through the PCA9564

At the TIM board, it is implemented a VME to I2C interface through the device PCA9564. As can be seen in figure 3, there are two PCA9564, one of them for accessing to the DS2438 sensors at TIMVME and TIMTTC for voltage, current and temperature monitoring, and also, a second PCA9564 to access to the TTCrx device, that also has an I2C interface.

As can be seen in [5], the PCA9564 device has 4 registers (Status, Data, Own address and Control), which can be accessed directly as TIM registers. Here we present the offset address at the TIM board of the corresponding registers for the PCA9564 used for reading the sensors and for the PCA9564 used for interface with the TTCrx.

3.1.1 I2C (sensor) Status. Register $80

0-7	Status	R	Default value = 0xF8
0-7	Time-out	W	Default value = 0xFF

3.1.2 I2C (sensor) Data. Register $82

Default value = 0x00

0-7	I2C Data	R/W

3.1.3 I2C (sensor) Address. Register $84

Default value = 0x00

0-7	I2C address	R/W

3.1.4 I2C (sensor) Control. Register $86

Default value = 0x00

0-7	I2C control	R/W

3.1.5 I2C (TTCrx) Status. Register $88

0-7	Status	R	Default value = 0xF8
0-7	Time-out	W	Default value = 0xFF

3.1.6 I2C (TTCrx) Data. Register $8A

Default value = 0x00

0-7	I2C Data	R/W

3.1.7 I2C (TTCrx) Own address. Register $8C

Default value = 0x00

0-7	I2C address	R/W

3.1.8 I2C (TTCrx) Control. Register $8E

Default value = 0x00

0-7	I2C control	R/W

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 (Write 0x40 in TIM_Base_Add + I2C (sensor) 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:

3.1.9 Write_PCA(out p_add, out p_data)

1) **Check PCA9564 status register, it has to be 0xF8.**
   (Read TIM_Base_Add + I2C (sensor) Status).
2) **Send START**
   (Write 0x60 in TIM_Base_Add + I2C (sensor) Control).
3) **Wait until interrupt flag (SI) is asserted.**
   (Read bit 3 of TIM_Base_Add + I2C (sensor) Control).
4) **Check status, it has to be 0x08.**
   (Read TIM_Base_Add + I2C (sensor) Status).
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 TIM_Base_Add + I2C (sensor) Data).
6) **Reset SI flag.**
   (Write 0x40 in TIM_Base_Add + I2C (sensor) Control).
7) **Wait until interrupt flag (SI) is asserted.**
   (Read bit 3 of TIM_Base_Add + I2C (sensor) Control).
8) **Check status, it has to be 0x18.**
   (Read TIM_Base_Add + I2C (sensor) Status).
9) **Send the data byte that you want to write by writing it in PCA_DATA.**
   (Write p_data on TIM_Base_Add + I2C (sensor) Data).
10) **Reset SI flag.**
    (Write 0x40 in TIM_Base_Add + I2C (sensor) Control).
11) **Wait until interrupt flag (SI) is asserted.**
    (Read bit 3 of TIM_Base_Add + I2C (sensor) Control).

11-BIS) If there is more than 1 byte to transmit (an array of bytes), go back to step 9) until all bytes have been sent.

12) **Check status, it has to be 0x28.**
    (Read TIM_Base_Add + I2C (sensor) Status).
13) **Send a STOP condition.**

(Write 0x50 to TIM_Base_Add + I2C (sensor) Control).

14) **Check status, it has to be 0xF8.**
(Read TIM_Base_Add + I2C (sensor) Status).

3.1.10 Read_PCA (out p_add, in p_data)

1) **Check PCA9564 status register, it has to be 0xF8.**
(Read TIM_Base_Add + I2C (sensor) Status).
2) **Send START**
(Write 0x60 in TIM_Base_Add + I2C (sensor) Control).
3) **Wait until interrupt flag (SI) is asserted.**
(Read bit 3 of TIM_Base_Add + I2C (sensor) Control).
4) **Check status, it has to be 0x08.**
(Read TIM_Base_Add + I2C (sensor) Status).
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 TIM_Base_Add + I2C (sensor) Data).
6) **Reset SI flag.**
(Write 0x40 in TIM_Base_Add + I2C (sensor) Control).
7) **Wait until interrupt flag (SI) is asserted.**
(Read bit 3 of TIM_Base_Add + I2C (sensor) Control).
8) **Check status, it has to be 0x40.**
(Read TIM_Base_Add + I2C (sensor) Status).
9) **Reset SI flag.**
(Write 0x40 in TIM_Base_Add + I2C (sensor) Control).
10) **Wait until interrupt flag (SI) is asserted.**
(Read bit 3 of TIM_Base_Add + I2C (sensor) Control).
11) **Read the data received from the device from PCA_DATA.**
(Read p_data from TIM_Base_Add + I2C (sensor) Data).
12) **Check status, it has to be 0x58.**
(Read TIM_Base_Add + I2C (sensor) Status).
13) **Send a STOP condition.**
(Write 0x50 to TIM_Base_Add + I2C (sensor) Control).
14) **Check status, it has to be 0xF8.**
(Read TIM_Base_Add + I2C (sensor) Status).

3.2 1-Wire Interface to the DS2438 through the DS2482.

3.2.1 Brief introduction to DS2482 and DS2438 ICs.

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 various internal registers that has this chip. 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). Executing 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 through the 1-W channel in use. 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.

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. It 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 passed from internal registers before being read by software, 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. It is 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 only 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 should be polled continuously until they are done (bit LOW).

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 "l". When the copy is completed, this bit will reset 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 completed.

- Recall Memory [B8hXXh]: This command recalls the stored values in EEPROM / SRAM page XXh to the scratchpad page XXh. This command must preceed 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 "1" during conversion. When the temperature conversion is done, the TB flag will clear 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 completed.

- 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 completed.

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 [6]). At the TIM 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.

3.2.2 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.

3.2.2.1 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:

3.2.2.2 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:

3.2.2.3 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) throw 1-W channel, and the page number of our 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 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.

3.2.3 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 (0xF0 for TIMVME sensor, 0xE1 for TIMTTC 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, [0xC3,0xF0]) ← for TIMVME 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 throw 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.

Therefore, 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.

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.

$$T = (-1)^{bit15\_of\_byte2}\left(\frac{byte\_1}{256} + byte\_2\right)^{o} C$$

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

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

At TIMVME channel 0 R_{sens} = 0.033 Ω, for TIMVME channel 1 R_{sens} = 0.150 Ω and for TIMTTC channel 2 R_{sens} = 0.150 Ω.

4 Interface to the TTCrx

The internal configuration registers of the TTCrx can also be read or written through an I2C interface. As explained in the TTCrx User Manual [2], the TTCrx occupies two consecutive positions in the 7-bit wide I2C address space:

TTCrx I2C access register name	TTCrx I2C address
I2C_pointer	ID_I2C<5:0>*2
I2C_data	ID_I2C<5:0>*2 + 1

Where, ID_I2C<5:0> correspond to the 6 least significant bits of the TTCrx address (ID<14:0>). The ID value is set at power up through some resistors at the TTCrq module, a label with its corresponding value has been attached to each TTCrq board.

The value written in the I2C_pointer will point to the corresponding TTCrx internal register. There are 29 TTCrx internal registers; their description can be seen in the TTCrx User Manual [2].

The procedure to read or write from a TTCrx internal register can be summarized as follows:

Write to a TTCrx internal register.

1) Write TTCrx internal register address to TTCrx I2C_pointer:
   Write_PCA(p_add = ID<5:0>*2, p_data = TTCrx internal register address)
2) Write desired data (TTCrx data) to TTCrx I2C_data:
   Write_PCA(p_add = ID<5:0>*2+1, p_data = TTCrx data)

Read from a TTCrx internal register.

1) Write TTCrx internal register address to TTCrx I2C_pointer:
   Write_PCA(p_add = ID<5:0>*2, p_data = TTCrx internal register address)

2) Read desired data (TTCrx data) to TTCrx I2C_data:
   Read_PCA(p_add = ID<5:0>*2+1, p_data = TTCrx data)

The routines Read_PCA and Write_PCA are the same described in 3.1.5 and 3.1.6, replacing the VME address offsets of I2C (sensor) with the ones corresponding to I2C (TTCrx).

5 Configuration and operation

After power up it is recommended to execute a reset of the board (write 0x8000 at register $00) and a reset of the TTCrq (write 0x8000 at $40).

The TTCrq reset will make that the QPLL_unlocked_registered and TTCrx_not_ready_registered flags are set to 1 due to the initialization procedure of the TTCrx and QPLL devices. Therefore, these two flags should be erased after ~800ms of a TTCrq reset, to allow detection of further problems (write a 0 at bit 2 and bit 9 of register $40).

Note that the TTCrq reset will reset the contents of the event and bunch counters, therefore, it is not recommended to perform a TTCrq reset unless a subsequent TTC event and bunch counters reset is going to be performed in the system.

After a reset, the actions to be performed can include programming interruptions, monitoring the sensors at the board or configuring any parameter of the TTCrx device. Otherwise, the TIM board is ready to start operation.

The main status information that should be checked before and/or during operation is:

- That the TIMTTC board is ON and has not been disconnected (bits 2 and 3 of register $00).
- Insure proper reception of TTC clock, i.e., no local clock is being used at the TIMVME or has ever been used (bits 7 and 8 of register $00 always to 0).
- The TTCrx is ready (bit 0 of $40 to 1) and has not gone into a not-ready status (bit 9 of $40 to 0). Also, no TTC transmission errors have occurred (bits 3 to 6 of $40 to 0).
- Insure proper locking of the QPLL (bit 1 of $40 to 1 and bit 2 of $40 to 0), and SEU status (bits 3 to 10 of register $4C to 0).

6 References

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

[2] TTCrx reference manual. http://ttc.web.cern.ch/TTC/TTCrx_manual3.11.pdf

[3] QPLL ASIC. http://proj-qpll.web.cern.ch/proj-qpll/

[4] TTCrq. http://ttc.web.cern.ch/TTC/TTCrqSpec.pdf

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

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

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