TileCal ROD Motherboard Software Library

User's Manual

B. Salvachúa, *J. Castelo, V. Castillo, C. Cuenca, A. Ferrer, E. Fullana, E. Higón, C. Iglesias, A. Munar, J. Poveda, A. Ruiz-Martínez, C. Solans, J. Valls*

IFIC
(CSIC-Universidad de Valencia)

Valencia - SPAIN

Abstract

This note describes the software library and an associated standalone application program to handle the TileCal ROD VME motherboard. The library uses the CMT packages vme_rcc and rcc_error, from the ATLAS Online Data Flow to handle the standard crate controller, VP-110 from Concurrent Technologies, and the custom bit3_rcc CMT package to handle an alternative crate controller, the BIT-3 from SBS$_{TM}$ Technologies.

The ROD library defines several C++ classes which can be used in either standalone applications to control and debug the RODs or with the TDAQ online software integration of the back-end hardware for the TileCal detector.

The library also includes special auxiliary classes to handle additional back-end boards related to the ROD operation like the TBM or the ROD injectors.

Index

Tables

Figures

1 Introduction

The trigger and data acquisition (TDAQ) back-end hardware of the ATLAS Hadronic Tile Calorimeter (TileCal) [1] is organized in custom 9U VME boards, called Read-Out Drivers (RODs) [2]. There will be a total of 32 ROD boards associated to the whole readout of the detector. Each ROD board receives a total of 8 input optical links with data from the front-end electronics of the calorimeter modules. Up to 45 channels are sent per link, each channel matching a photomultiplier output signal.

The ROD system constitutes the link between the front-end hardware of the calorimeter and the standard general data acquisition. Data are collected from the front-end system and processed inside the ROD. For this purpose each ROD can handle up to 4 Processing Units (PUs) [3] which contain fixed-point Digital Signal Processors (DSPs). Other functionalities of the ROD are to receive the Timing Trigger and Control (TTC) signals from the Central Trigger Processor (CTP) [4], and to synchronize them with the ones coming from the front-end electronics. The ROD must read, process and format about 9856 channels at the expected maximum trigger rate of 100 KHz (Level 1 Trigger rate).

In the final configuration 4 crates are reserved for the ROD system organization, each one holding up to 8 motherboards. This organization follows the detector Regions of Interest (RoI). The calorimeter is divided in 4 RoI, 2 extended barrels and 2 central barrels. Each RoI constitutes a partition from the TDAQ point of view.

2 ROD Motherboard VME Address Space

The VME address space for the ROD motherboard is described in Table 1, where the 5-bit geographical address of the ROD modules (bits 24-28) corresponds to their slot position on the crate. Bits 7 to 10 are associated to the different functional blocks of the ROD such as the TTC FPGA, PUs OC FPGAs, etc. And bits 2 to 6 are reserved for the internal register mapping of those blocks.

Figure 1 shows a picture of the ROD motherboard indicating the different fundamental blocks of the board.

VMEADD	31	31	29	28 …… 24	23 …… 11	10	9	8	7	6 …… 2	1	0
LOCAL/BUSY				GEOGRAPH.ADD		0	0	0	0	INTERNAL ADDRESS		
BOOT				GEOGRAPH.ADD		0	0	0	1	INTERNAL ADDRESS		
TTC				GEOGRAPH.ADD		0	0	1	0	INTERNAL ADDRESS		
IRQ				GEOGRAPH.ADD		0	0	1	1	INTERNAL ADDRESS		
PU1				GEOGRAPH.ADD		0	1	0	0	INTERNAL ADDRESS		
PU2				GEOGRAPH.ADD		0	1	0	1	INTERNAL ADDRESS		
PU3				GEOGRAPH.ADD		0	1	1	0	INTERNAL ADDRESS		
PU4				GEOGRAPH.ADD		0	1	1	1	INTERNAL ADDRESS		
OC1				GEOGRAPH.ADD		1	0	0	0	INTERNAL ADDRESS		
OC2				GEOGRAPH.ADD		1	0	0	1	INTERNAL ADDRESS		
OC3				GEOGRAPH.ADD		1	0	1	0	INTERNAL ADDRESS		
OC4				GEOGRAPH.ADD		1	0	1	1	INTERNAL ADDRESS		
STAGING1				GEOGRAPH.ADD		1	1	0	0	INTERNAL ADDRESS		
STAGING2				GEOGRAPH.ADD		1	1	0	1	INTERNAL ADDRESS		
STAGING3				GEOGRAPH.ADD		1	1	1	0	INTERNAL ADDRESS		
STAGING4				GEOGRAPH.ADD		1	1	1	1	INTERNAL ADDRESS		

Table 1 : VME address schema.

2.1 Booting of the crate VP-110 controller

The standard VME controller for the ATLAS DAQ is the VP-110/01 [6], VME Pentium Single Board Computer (SBC) from Concurrent Technologies, which is a diskless PC and needs to be booted from the network.

Instructions for the booting are stored in the computer ROM of the SBC. Each SBC has two Ethernet interfaces with two different MAC addresses. An IP address should be assigned to any of those interfaces. So that, when the SBC is powered, it automatically sends a broadcast message to any server of the network requesting the booting of the file system.

The server is a PC connected to the network which has two services or daemons running, ***dhcpd*** and ***tftpd***. The first one, ***dhcpd***, answers the initial booting request from the SBC while the second one, ***tfpd***, initiates the upload of the boot image file to the SBC. The boot image is a binary file which, among other instructions, contains the location directory in the server of the root file system which the SBC will upload.

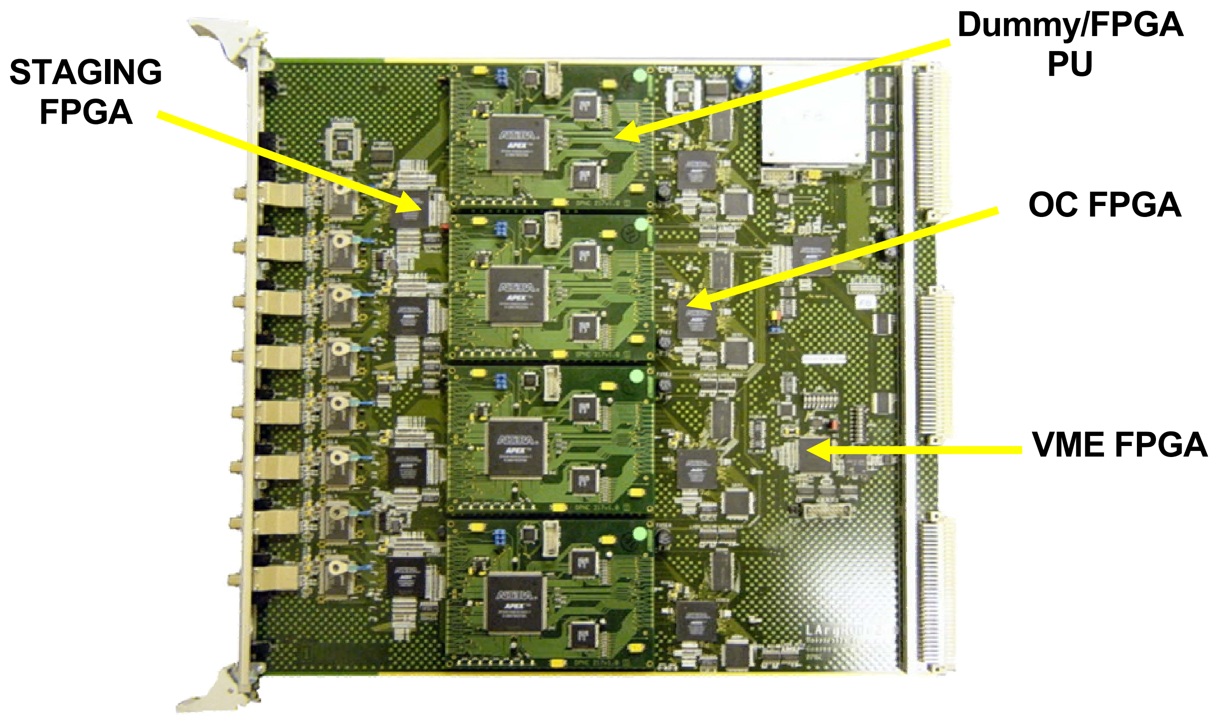

Figure 1: Picture of the ROD motherboard showing the location of the different functional blocks.

3 Software organization

The ROD motherboard software library is part of the TicalOnline software package. The library is written in C++ and it is organized as a standard ATLAS CMT online software package. The different versions of the software can be found in the directory TileVmeROD. The present version of the package corresponds to v5r0p0. This version uses the Data Flow version 01-00-00 and Online 01-00-00. From the directory TileVmeROD the users have access to a CMT like structure of directories (Figure 2) where header, source and binary files are stored.

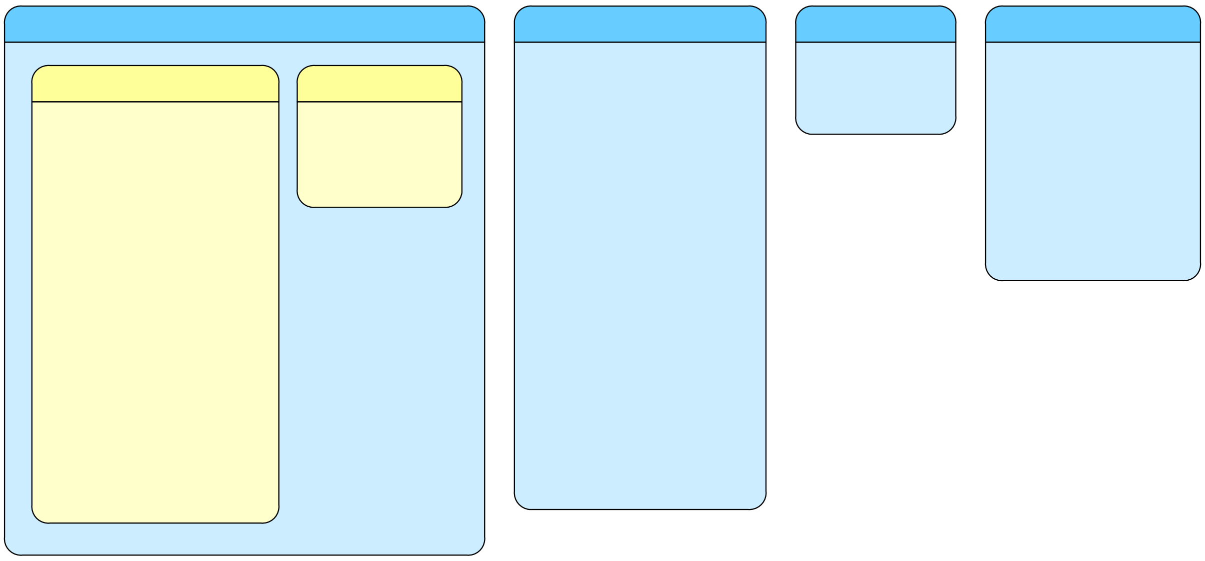

Figure 2: Directory structure of the TileVmeROD CMT package.

The latest version of the package can be download from the TDAQ ATLAS software repository under the tree [atlasdaq]/detectors/Tile. Detailed information about the software can be found in http://ific.uv.es/tical/rod/online_tdaq/index.html .

To download the software package follow the steps described below:

- Configure the environments variables:
  - *export CVSROOT=:ext::atdaq-sw.cern.ch:/atdaqcvs*
  - *export CVS_RSH=ssh*
- Retrieve version vXrYpZ of the package type:
  - *cvs -co -r vXrYpZ -d PackageName vXrYpZ/detectors/Tile/PackageName*

An online description of the TileVmeROD library, generated with the tool doxygen [5], is available under the link http://ific.uv.es/~belen/WWW_tilecal/TileVmeROD/index.html .

The software is based on a hierarchy of classes (see Figure 3). The VME and VMEMasterMap classes are responsible for the VME access either through the VP-110 [6] crate controller, from Concurrent Technologies, or through the BIT-3 [7] crate controller, from SBS$_{TM}$ Technologies. Although the standard of ATLAS is the VP-110, we had incorporated the option to use the BIT-3 crate controller for tests on the ROD system. The library uses the following standard CMT packages:

- vme_rcc [8],
- rcc_error,

when the VP-110 is used and a custom CMT package:

- bit3_rcc

when the BIT-3 controller is used.

The CRATE, ROD and AUX classes are those which describe our system. For instance, in the CRATE class it is defined the number of ROD modules in a crate and the number of PUs in each ROD module. Through the CRATE class a VME initialization and mapping is done. This class contains an array of pointers to the classes ROD and AUX. And it is inside the classes ROD and AUX where the users can read and write in the module registers.

The AUX class is reserved to access the registers of additional modules. This is the case of the Trigger and Busy Module (TBM) or the pre-RODs.

Through the ROD class users can read and write in the ROD module registers associated to each functional block. For that purpose this class has pointers to the following classes:

- **LOCAL class**: access to the ROD motherboard LOCAL registers.
- **BUSY class**: access to the ROD motherboard BUSY registers.
- **TTC class**: access to the ROD motherboard TTC FPGA registers.
- **OC class**: access to the ROD motherboard Output Controller FPGA registers.
- **STAGING class**: access to the ROD motherboard Staging FPGA registers.
- **FPGA_PU class**: access to the ROD FPGA/Dummy PU (Processing Unit) control.
- **DSP_PU class**: access to the ROD PU (Processing Unit) control.

Declarations of the classes and its members are done in header files, while definitions are done in source files. Each class has an associated header and a source file with the same name as the class preceded by the ROD prefix.

Figure 3 shows the hierarchy of classes of the ROD library. In the next sections a more detailed description of each class, its members and its methods is presented.

All necessary headers for the TileVmeROD library are included in a file called RODIncludes.h. This file includes also the file RODDefinitions.h where the offsets of the registers, error codes and other variables are defined. A set of important variables of the RODDefinitions.h file and their description is shown in Table 2.

The error code variables will be explained in Section 4.

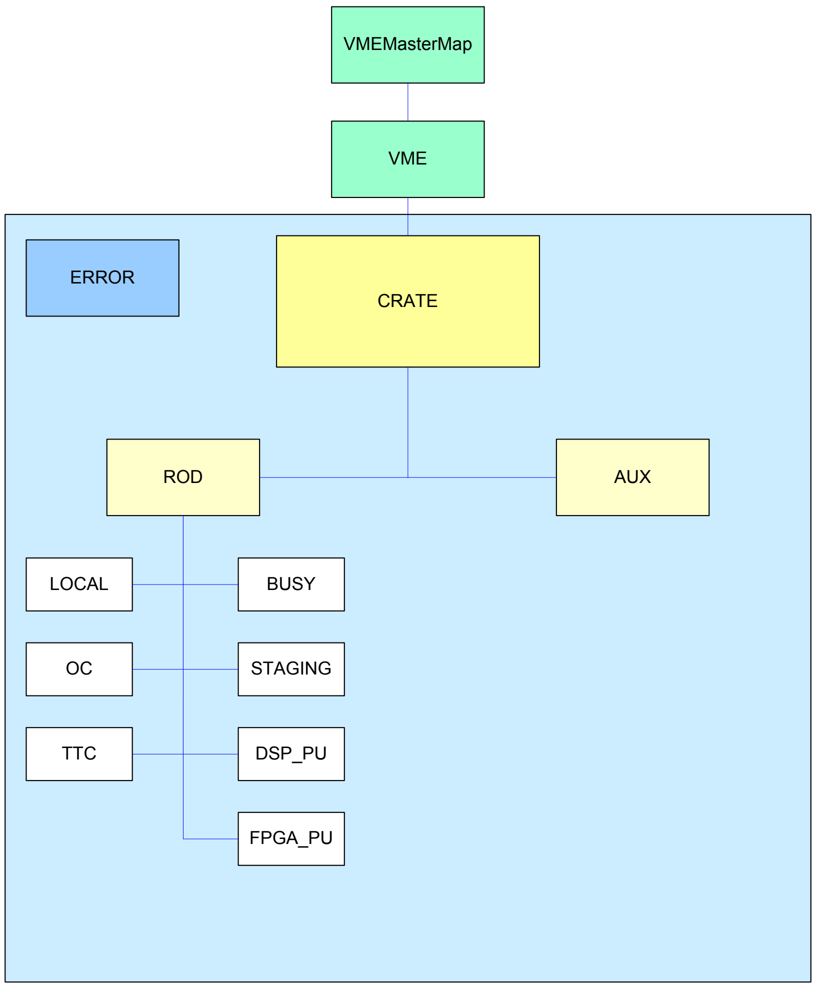

Figure 3: ROD library classes' hierarchy.

NAME	VALUE	DESCRIPTION
WIN_SIZE	0x800	Size in bytes for the ROD mapping.
FAST	1	Fast VME mode of reading.
SAFE	0	Safe VME mode of reading.
BIT3	20	Crate Controller from SBS Techonologies.
VP110	30	Crate Controller from Concurrent Techonologies.
MAXNOSTAG	4	Maximum number of Stagging FPGAs in a ROD module.
MAXNOOCS	4	Maximum number of Output Controllers in a ROD module.
MAXNOPUS	4	Maximum number of Processing Units in a ROD module.
MIN_SLOT	4	First slot number that can accept a ROD or an AUX module.
MAX_SLOT	20	Last slot number that can accept a ROD or an AUX module.
MAXNORODS	16	Maximum number of ROD modules in a VME crate.
MAXNOAUXS	5	Maximum number of AUXiliary modules in a VME crate.
MAXTEMPID	7	Maximum temperature identifier.

Table 2: Description of some variables defined in RODDefinitions.h.

4 Error Return Codes

Most of the methods in the TileVmeROD library return an error corresponding either to the VME error code from the rcc_vme package, from the bit3_rcc package or from the TileVmeROD library itself. The methods to handle with these errors are defined in the ERROR class. Table 3 shows a description of the possible errors associated to the ROD library. Other possible error values may come from the vme_rcc package and the bit3_rcc package during the access to the VME bus.

ERROR NAME	VALUE	DESCRIPTION
ROD_SUCCES	VME_SUCCES	All ok.
ROD_FAIL	-1	Something failed.
ROD_NO_KNOW	60	Error not known.
ROD_FAIL_NORODS	61	Incorrect number of ROD modules.
ROD_FAIL_NOAUXS	62	Incorrect number of auxiliary modules.
ROD_FAIL_NOPUS	63	Incorrect number of Processing Units in a ROD module.
ROD_FAIL_ALLOC	64	Failed allocating memory space.
ROD_NO_MODULE	65	ROD module not found in the slot number.
ROD_FAIL_OC	66	Incorrect Output Controller identifier.
ROD_FAIL_STAG	67	Incorrect Staging FPGA identifier.
ROD_FAIL_FPGA_PU	68	Incorrect Dummy/FPGA Processing Unit.
ROD_ERROR_S_F	69	Incorrect mode, nor SAFE nor FAST.
ROD_STAG_TEMP	70	Incorrect temperature identifier. Up to 8 temperatures can be read, one per G-Link. From '0' to MAXTEMPID.
INFPGA_FILE_ERR	71	Error opening the Input FPGA code.
PU_FAIL_MODE	72	Incorrect PU boot mode.
INFPGA_FAIL_LOAD	73	Failed loading Input FPGA code.
DSP_FILE_ERR	74	Error opening the DSP code.
DSP_FAIL_LOAD	75	Failed loading DSP code.
DSP_HPI_UNALIGNED_DATA	76	HPI DSP address unaligned.

Table 3: Error code description.

5 Application Example

To access the registers of a ROD module a simple application can be built following these few steps:

1. In order to get all the definitions and includes one should include the file RODIncludes.h. The classes are defined in a namespace, RODTile, which should be used in the application:

   ```
   #include "TileVmeROD/RODIncludes.h"

   using namespace RODTile;
   ```

2. Create a CRATE instance (or a pointer), and an ERROR instance (or a pointer):

   ```
   CRATE * MyCrate = new CRATE;

   ERROR * Error = new ERROR;

   ROD_Error ret;
   ```

3. Init the CRATE object by calling the method CrateInit( ) from the CRATE class:

   ```
   int NORODS = 1;

   int NOAUXS = 0;

   int DeviceName = VP110; // or BIT3

   ret = MyCrate->CrateInit(NORODS,NOAUXS,DeviceName);

   if (ret != ROD_SUCCESS) {

          Error->RODErrorPrint(ret);

          Return (ROD_FAIL);

   }
   ```

4. In this application the user defines a single ROD module, to access the registers of this ROD module the user has first to initialize it through the following code:

   ```
   Int NOPUS = 2;        // Number of PU in a ROD module

   Int SLOT = 7;         // Slot where the ROD is connected

   ret = MyCrate->pRod[0]->RODInit(NOPUS, SLOT);

   if (ret != ROD_SUCCESS) {

          Error->RODErrorPrint(ret);

          Return (ROD_FAIL);

   }
   ```

   The index in the pRod[ ] pointer indicates the user is accessing the first ROD module initialized (in this case, there is only one).

5. Once the ROD module is initialized the user can access all the registers of the ROD:

   ```
   u_int value, data;

   int mode;

   data = 0xfa12eb34;

   mode = SAFE;          // or FAST

   ret = MyCrate->pRod[0]->pStaging[3]->WriteCommande(data,mode);

   if (ret != ROD_SUCCESS) {
   ```

```
               Error->RODErrorPrint(ret);
               Return (ROD_FAIL);
        }
        ret = MyCrate->pRod[0]->pFpga_pu[0]->ReadStatus(&value,mode);
        if (ret != ROD_SUCCESS) {
               Error->RODErrorPrint(ret);
               Return (ROD_FAIL);
        }
        ret = MyCrate->pRod[0]->pFpga_pu[0]->PrintStatus(value);
        if (ret != ROD_SUCCESS) {
               Error->RODErrorPrint(ret);
               Return (ROD_FAIL);
        }
        data = 0xbabebabe;
        ret = MyCrate->pRod[0]->pTtc->WriteControl(data,mode);
        if (ret != ROD_SUCCESS) {
               Error->RODErrorPrint(ret);
               Return (ROD_FAIL);
        }
```

6. At the end of the application the VME bus should be closed and all created pointers deleted. In order to do it the user should call the CrateShutDown( ) method from the CRATE class:

```
MyCrate->CrateShutDown();
delete MyCrate;
delete Error;
```

There are many applications already built using the ROD library. For example, **menuROD** is a program which allows the ROD registers to be accessed through a simple menu list.

A GUI Panel, XTestROD, has being also developed using the TileVmeROD library. XTestROD [9] is being used for the ATLAS TileCal ROD characterization and system tests.

6 Future Upgrades

Class AUX was created to access any new VME module installed in the ROD crate. However, the library is being continually updated due to new developments on the electronics and firmware upgrades that can change the VME mapping. The near future plan is to implement two new classes: the class PreROD to handle the PreROD modules and the class TBM to handle the Trigger and Busy Module (TBM).

The PreROD is a 9U VME module which has two functionalities. The first one is to work as a ROD injector, the card can be configured to send known or random data to the ROD modules. The second one is to work as an optical multiplexer, the signal in TileCal is redundant, and each superdrawer sends two fibres with the same information to the ROD system. When the PreRODs are configured as an optical multiplexer they check the data and send the uncorrupted data to the ROD modules.

The TBM is a 9U VME module which is placed in each ROD crate and is responsible for handling input TTC signals and the output busy signals of the 8 ROD motherboards in the ROD crate.

7 Class CRATE

FILES

RODCrate.h

RODCrate.cc

PUBLIC VARIABLES

NAME	DESCRIPTION
int NoRODModules	Number of ROD modules inside a crate.
int NoAUXModules	Number of Auxiliary modules inside a crate.
ROD* pRod[MAXNORODS]	Array of pointers to an object of the ROD class.
AUX* pAux[MAXNOAUXS]	Array of pointers to an object of the AUX class.
VME* pVme	Pointer to an object of the VME class.

PUBLIC METHODS

7.1 CRATE ( )

Description

Constructor of the class, it does nothing.

7.2 ~CRATE ( )

Description

Destructor of the class, it does nothing.

7.3 ROD_Error CrateInit (int NoROD, int NoAUX, int DeviceName)

Description

This method initializes the crate by allocating the required memory for the pRod[ ] and pAux[ ] pointers. It also initializes the variable NoRODModules and NoAUXModules and opens the VMEbus library/driver. It must be called at the beginning of any application to allow VME access.

Input values

NAME	DESCRIPTION
int NoROD	Number of ROD modules inside a Crate.
int NoAUX	Number of Auxiliary modules inside a Crate.
Int DeviceName	The device name can be set to **VP110** or to **BIT3.**

Return values

It returns ROD_SUCCESS on success and the corresponding error code on failure. Possible errors in this method are: ROD_FAIL_NORODS, ROD_FAIL_NOAUXS, ROD_FAIL_ALLOC or the error code returned from the vme_rcc and bit3_rcc libraries.

7.4 ROD_Error CrateShutDown ( )

Description

It must be called before ending any application. CrateShutDown( ) frees the memory of several internal pointers, pRod[ ] and pAux[ ], and closes the VME library/driver.

Return values

It returns ROD_SUCCESS on success and the corresponding error code on failure.

8 Class ROD

FILES

RODModule.h

RODModule.cc

PUBLIC VARIABLES

NAME	DESCRIPTION
int npus	Number of Processing Units inside a ROD (up to 4).
int slot	Crate slot number where a ROD motherboard is placed.
VMEMasterMap * pVmm	Pointer to an object of the VMEMasterMap class.
VME* p_My_Vme	Pointer to an object of the VME class.
u_int base_address	Base Address of a ROD board. It is built when the RODInit( ), method of the ROD class, is called.
u_int window_size	The window size is initialized when the RODInit( ), method of the ROD class, is called.
u_int add_modif	The address modifier is initialized when the RODInit( ), method of the ROD class, is called.
u_int options	Set to '0'.
LOCAL* pLocal	Pointer to an object of the LOCAL class.
BUSY* pBusy	Pointer to an object of the BUSY class.
OC* pOc[MAXNOOCS]	Array of pointers to an object of the OC class.
TTC* pTtc	Pointer to an object of the TTC class.
STAGING* pStaging[MAXNOSTAG]	Pointer to an object of the STAGING class.
FPGA_PU* pFpga_pu[MAXNOPUS]	Pointer to an object of the FPGA_PU class.

PUBLIC METHODS

8.1 ROD ( )

Description

Constructor of the class, it does nothing.

8.2 ~ROD ( )

Description

Destructor of the class. It frees the memory space reserved for the pointers pLocal, pBusy, pOc[ ], pTtc[ ], pStaging[ ] and pFpga_pu[ ] defined as public variables. It also unmaps VME by calling to the MasterUnmap( ) method of the VME class.

8.3 ROD_Error RODInit (int NoPu, int Slot)

Description

This method must be called before accessing any register of the ROD motherboard. It initializes the variables of the class, performs a VME map of the ROD board by calling to the MasterMap( ) method of the VME class, and reserves the necessary memory space for the declared pointers. RODInit( ) method also checks if a ROD module is in the slot by writing the hexadecimal word 0xbabebabe on the ROD Base Address.

Input values

NAME	DESCRIPTION
int NoPu	Number of PU modules inside a ROD module (up to 4).
int Slot	Crate slot number where a ROD motherboard is installed.

Return values

If no errors occur RODInit( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

9 Class AUX

FILES

AUXModule.h

AUXModule.cc

PUBLIC VARIABLES

NAME	DESCRIPTION
int slot	Crate slot number where an auxiliary module is placed.
VMEMaterMap* pVmm	Pointer to an object of the VMEMasterMap class.
u_int base_address	Base Address of an Auxiliary module. It is initialized when running the AUXInit( ) method of this class (see below).
u_int window_size	The windows size is initialized when the AUXInit( ) method is called.
u_int add_modif	The address modifier is initialized when the AUXInit( ) method is called.
u_int options	Set to '0'.

PUBLIC METHODS

9.1 AUX ( )

Description

Constructor of the class, it does nothing.

9.2 ~AUX ( )

Description

Destructor of the class. It does the master unmaping of the auxiliary module done in AUXInit( ).

9.3 ROD_Error AUXInit (int Slot)

Description

This method must be called before accessing any register in the AUX VME module. It initializes the variables of the class, and does a VME map of the auxiliary board by calling to the MasterMap( ) method of the VME class.

Input values

NAME	DESCRIPTION
int Slot	Crate slot number where an auxiliary VME card is installed.

Return values

If no errors occur RODInit( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

10Class LOCAL

FILES

RODLocal.h

RODLocal.cc

PUBLIC VARIABLES

NAME	DESCRIPTION
VMEMasterMap* p_My_Vmm	Pointer to an object of the VMEMasterMap class.

PUBLIC METHODS

10.1 LOCAL ( )

Description

Constructor of the class, it reserves memory space for private pointers.

10.2 ~LOCAL ( )

Description

Destructor of the class. It frees the memory space reserved for private pointers.

10.3 ROD_Error WriteBaseAdd (u_int data, int mode)

Description

This method writes an unsigned integer word on the Local Base Address register of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
u_int data	32-bit word to be written in the Local Base Address register of the ROD board.

Return values

If no errors occur WriteBaseAdd( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

10.4 ROD_Error ReadBaseAdd(u_int* value, int mode)

Description

This method reads in the Local Base Address register of the ROD module and keeps the value read in the pointer value. It returns an unsigned integer word with the corresponding error based on the VME error code from rcc_vme package or bit3_rcc package.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the Local Base Address register of ROD module.

Return values

If no errors occur ReadBaseAdd( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

10.5 void PrintBaseAdd (u_int value)

Description

This method prints on the standard output the value read from the Local Base Address register of the ROD module.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Local Base Address register.

10.6 ROD_Error GeneralReset(int mode)

Description

This method sends a General Reset to the ROD board.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur GeneralReset( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

10.7 ROD_Error ReadStatus (u_int* value, int mode)

Description

This method reads in the Local Status register of the ROD board.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the Local Status register of the ROD module.

Return values

If no errors occur ReadStatus( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

10.8 void PrintStatus (u_int status)

Description

This method decodes and prints to the standard output the information read in the Local Status register of the ROD cards.

Input values

NAME	DESCRIPTION
u_int status	Word read from the Local Status register.

10.9 ROD_Error ReadBoardId (u_int* value, int mode)

Description

This method reads in the Local Board Identifier register of the ROD.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the Local Board Identifier register of the ROD module.

Return values

If no errors occur ReadBoardId( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

10.10 void PrintBoardId (u_int identif)

Description

This method decodes and prints to the standard output the information read from the Local Board Identifier register of the ROD module.

Input values

NAME	DESCRIPTION
u_int identif	Word read from the Local Board Identifier register of the ROD module.

10.11 ROD_Error ForceBusy (u_int data, int mode)

Description

This method writes in the Local Force Busy register of the ROD.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
u_int data	32-bit word with the information to be written in the Local Force Busy register: bit 0 = 1 → Busy Forced to 1 (default during installation) bit 1 = 1 → Data bus extern chip VME (vmeconfig) bit 2 = 1 → Quick answer 2 clock (vmeconfig) bit 3 = 1 → Quick answer 4 clock (vmeconfig)

Return values

If no errors occur ForceBusy( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

10.12 ROD_Error ReadVmeVersion (u_int* value, int mode)

Description

This method reads the VME FPGA version of the ROD.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word with the information of the VME FPGA version of the ROD module.

Return values

If no errors occur ReadVmeVersion( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

10.13 void PrintVmeVersion (u_int value)

Description

This method decodes and prints to the standard output the VME FPGA version of the ROD module.

Input values

NAME	DESCRIPTION
u_int value	Word with the information of the VME FPGA version of the ROD module.

11 Class OC

FILES

RODOc.h

RODOc.cc

PUBLIC VARIABLES

NAME	DESCRIPTION
int id	Output Controller FPGA identifier. The ***id*** can be 0, 1, 2 or 3. It must be set in order to access the registers of the corresponding Output Controller FPGA.
VMEMasterMap* p_My_Vmm	Pointer to an object of the VMEMasterMap class.

The following methods will be applied to the corresponding Output Controller FPGA, selected by the ***id*** public variable. If the user does not set the ***id*** variable, the method will return ROD_FAIL_OC.

PUBLIC METHODS

11.1 OC ( )

Description

Constructor of the class, it reserves memory space for private pointers.

11.2 ~OC ( )

Description

Destructor of the class. It frees the memory space reserved for private pointers.

11.3 ROD_Error WriteDummy (u_int data, int mode)

Description

This method writes in the Dummy register of the selected Output Controller FPGA.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written in the Dummy register of the selected Output Controller FPGA. Depending on the setting of the variable class member ***id*** it will access to the corresponding Output Controller FPGA.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteDummy( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

11.4 ROD_Error ReadDummy (u_int* value, int mode)

Description

This Method reads in a Dummy register of the selected Output Controller FPGA.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from a Dummy register of the selected Output Controller FPGA.

Return values

If no errors occur ReadDummy( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

11.5 void PrintDummy (u_int value)

Description

This method decodes and prints to the standard output the information read in the Dummy register of the selected Output Controller FPGA.

Input values

NAME	DESCRIPTION
u_int identif	Word read from a Dummy register of the selected Output Controller FPGA.

11.6 ROD_Error Reset (u_int data, int mode)

Description

This method resets the selected Output Controller FPGA.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written in the Reset register of the selected Output Controller FPGA. Depending on the setting of the variable class member ***id*** it will access to the corresponding Output Controller.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur Reset( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

11.7 ROD_Error ReadStatus (u_int* value, int mode)

Description

This method reads in the Status register of the selected Output Controller FPGA.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the Status register of the selected Output Controller FPGA.

Return values

If no errors occur ReadStatus( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

11.8 void PrintStatus (u_int value)

Description

This method decodes and prints to the standard output the information read in the Status register of the selected Output Controller FPGA.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Status register of the selected Output Controller FPGA.

11.9 ROD_Error WriteConfig (u_int data, int mode)

Description

This method writes in the Configuration register of the selected Output Controller FPGA.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
u_int data	32-bit word to be written in the Configuration register of the selected Output Controller FPGA. Depending on the setting of the variable class member ***id*** it will access to the corresponding Output Controller FPGA.

Return values

If no errors occur WriteConfig( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

11.10 ROD_Error ReadConfig (u_int* value, int mode)

Description

This method reads in the Configuration register of the selected Output Controller FPGA.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the Configuration register of the selected Output Controller FPGA.

Return values

If no errors occur ReadConfig( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

11.11 void PrintConfig (u_int value)

Description

This method decodes and prints to the standard output the information read in the Configuration register of selected Output Controller FPGA.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Configuration register of the selected Output Controller FPGA.

11.12 ROD_Error ReadSdram (u_int* value, int mode)

Description

ReadSdram( ) reads the Output Controller SDram register.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	32-bit word read from the SDram register of the selected Output Controller FPGA.

Return values

If no errors occur ReadSdram( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

11.13 ROD_Error ReadVersion(u_int* value, int mode)

Description

This method reads in the Version register of the selected Output Controller FPGA.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the Version register of the selected Output Controller FPGA.

Return values

If no errors occur ReadVersion( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

11.14 void PrintVersion (u_int value)

Description

This method decodes and prints to the standard output the information read in the Version register of the selected Output Controller FPGA.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Version register of the selected Output Controller FPGA.

12 Class BUSY

FILES

RODBusy.h

RODBusy.cc

PUBLIC VARIABLES

NAME	DESCRIPTION
VMEMasterMap* p_My_Vmm	Pointer to an object of the VMEMasterMap class.

PUBLIC METHODS

12.1 BUSY ( )

Description

Constructor of the class, it reserves memory space for private pointers.

12.2 ~BUSY ( )

Description

Destructor of the class. It frees the memory space reserved for private pointers.

12.3 ROD_Error ReadStatus (u_int* value, int mode)

Description

This method reads in the Busy Status register of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the Busy Status register of the ROD module.

Return values

If no errors occur ReadStatus( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

12.4 void PrintStatus (u_int status)

Description

This method decodes and prints to the standard output the information read in the Busy Status register of the ROD module.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Busy Status register of the ROD module.

12.5 ROD_Error WriteResetControl (u_int data, int mode)

Description

This method writes in the Busy Reset and Control register of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
u_int data	32-bit word to be written in the Busy Reset and Control register of the ROD module. bit 0 = 1 → Reset fifo bit 1 = 1 → Reset busy duration counter bit 2 = 1 → Reset Sreq counter bit 4 = 1 → Reset timing: reset interval, duration and clock divider counters bit 8 = 1 → Write busy duration counter to fifo bit 16 = 1 → Send sreq

Return values

If no errors occur ReadBaseAdd( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

12.6 ROD_Error ReadMisc (u_int* value, int mode)

Description

This method reads the Busy Miscellaneous register of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the Busy Miscellaneous register of the ROD module.

Return values

If no errors occur ReadMisc( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

12.7 void PrintMisc (u_int value)

Description

This method decodes and prints to the standard output the information read in the Busy Miscellaneous register of the ROD module.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Busy Miscellaneous register of the ROD module.

12.8 ROD_Error WriteMisc (u_int data, int mode)

Description

This method writes in the Busy Miscellaneous register of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
u_int data	32-bit word to be written in the Busy Miscellaneous register. bit 0 = 0 → Busy source VME bit 0 = 1 → Busy source PU bit 1 = 0 → Manual mode bit 1 = 1 → Normal mode bit 9 = 1 → Enable busy bit 12 = 1 → Enable sreq bit 23:16 → Mask busy (pu4_busy2 down to pu1_bus1)

Return values

If no errors occur WriteMisc( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

12.9 ROD_Error ReadIntervFifo (u_int* value, int mode)

Description

This method reads in the Busy Interval Fifo register of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the Busy Interval Fifo register of ROD module.

Return values

If no errors occur ReadIntervFifo( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

12.10 ROD_Error WriteIntervFifo (u_int data, int mode)

Description

This method writes in the Busy Interval Fifo register of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
u_int data	32-bit word to be written in the Busy Interval register of ROD module.

Return values

If no errors occur WriteIntervFifo( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

12.11 ROD_Error ReadSreqMax (u_int* value, int mode)

Description

This method reads in the Busy Sreq Maximum register of the ROD module. This register contains the maximum number of busies before send a Sreq.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the Busy Sreq Maximum register of ROD module.

Return values

If no errors occur ReadSreqMax( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

12.12 ROD_Error WriteSreqMax (u_int data, int mode)

Description

This method writes in the Busy Sreq Maximum register of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
u_int data	32-bit word to be written in the Busy Sreq maximum register of ROD module. Number of busy counts before send a IRQ.

Return values

If no errors occur WriteSreqMax( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

12.13 ROD_Error ReadDivClock (u_int* value, int mode)

Description

This method reads in the Busy Divider Clock register.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the Busy Divider Clock register of the ROD module.

Return values

If no errors occur ReadDivClock( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

12.14 ROD_Error WriteDivClock (u_int data, int mode)

Description

This method writes in the Busy Divider Clock register of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
u_int data	32-bit word to be written in the Busy Divider Clock register of the ROD module. bit 7:0 → busy count each (100 nS + 25nS × (n - 1))

Return values

If no errors occur WriteDivClock( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

12.15 ROD_Error ReadFifo (u_int* value, int mode)

Description

This method reads in the Busy Fifo register of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the Busy Fifo register of the ROD module.

Return values

If no errors occur ReadFifo( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

12.16 void PrintFifo (u_int value)

Description

This method decodes and prints to the standard output the information read in the Busy Fifo register of the ROD module.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Busy Fifo register of the ROD module.

12.17 ROD_Error ReadDurationBusy (u_int* value, int mode)

Description

This method reads in the Busy Duration Busy register of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the Busy Duration register of ROD module.

Return values

If no errors occur ReadDurationBusy( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

12.18 void PrintDurationBusy (u_int value)

Description

This method decodes and prints to the standard output the information read in the Busy Duration register of the ROD module.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Busy Duration register of ROD module.

12.19 ROD_Error ReadSreqCounter (u_int* value, int mode)

Description

This method reads in the Busy Sreq Counter register of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the Busy Sreq Counter register of ROD module.

Return values

If no errors occur ReadDurationBusy( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

12.20 void PrintSreqCounter (u_int value)

Description

This method decodes and prints to the standard output the information read in the Busy Sreq Counter register of the ROD module.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Busy Sreq Counter register of ROD module.

12.21 ROD_Error SendBusy (int mode)

Description

This method writes in the Busy Send Busy register of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur SendBusy( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

13 Class TTC

FILES

RODTtc.h

RODTtc.cc

PUBLIC VARIABLES

NAME	DESCRIPTION
VMEMasterMap* p_My_Vmm	Pointer to an object of the VMEMasterMap class.

PUBLIC METHODS

13.1 TTC ( )

Description

Constructor of the class, it reserves memory space for private pointers.

13.2 ~TTC ( )

Description

Destructor of the class. It frees the memory space reserved for private pointers.

13.3 ROD_Error WriteDummy (u_int data, int mode)

Description

This method writes in the TTC Dummy register of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
u_int data	32-bit word to be written in the TTC Dummy register of ROD modules.

Return values

If no errors occur WriteDummy( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

13.4 ROD_Error ReadDummy (u_int* value, int mode)

Description

This method reads in the TTC Dummy register of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the TTC Dummy register of ROD module.

Return values

If no errors occur ReadDummy( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

13.5 void PrintDummy (u_int value)

Description

This method decodes and prints to the standard output the information read in the TTC Dummy register of the ROD module.

Input values

NAME	DESCRIPTION
u_int value	Word read from the TTC Dummy register of the ROD module.

13.6 ROD_Error WriteControl (u_int data, int mode)

Description

This method writes in the TTC Control register of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
u_int data	32-bit word to be written in the TTC Control ID register of ROD modules. bit 2:0 → Mode , Only one bit can be set to 1: bit 0 = 1 → Mode VME bit 1 = 1 → Mode Local bit 2 = 1 → Mode TTC bit 3 = 0 → Local clock selection bit 3 = 1 → TTCrx clock selection bit 4 = 1 → L1A mode double pulse bit 7:5 → Pulse bit 5 = 1 → Flush bit 6 = 1 → Clear Status bit 7 = 1 → TTC reset

Return values

If no errors occur WriteControl( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

13.7 ROD_Error ReadControl (u_int* value, int mode)

Description

This method reads in the TTC Control register of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the TTC Control register of ROD module.

Return values

If no errors occur ReadControl( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

13.8 void PrintControl (u_int value)

Description

This method decodes and prints to the standard output the information read in the TTC Control register of the ROD module.

Input values

NAME	DESCRIPTION
u_int value	Word read from the TTC Control register of ROD module.

13.9 ROD_Error WriteBCID (u_int data, int mode)

Description

This method writes in the TTC Bunch Crossing ID register of ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
u_int data	32-bit word to be written in the TTC Bunch Crossing ID register of the ROD module. bit 11:0 → TTC Bunch Crossing ID

Return values

If no errors occur WriteBCID( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

13.10 ROD_Error WriteEVID (u_int data, int mode)

Description

This method writes in the TTC Event ID register of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
u_int data	32-bit word to be written in the TTC Event ID register of ROD module. bit 31:0 → TTC Event ID

Return values

If no errors occur WriteEVID( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

13.11 ROD_Error WriteTtype (u_int data, int mode)

Description

This method writes in the TTC Type register of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
u_int data	32-bit word to be written in the TTC Type register of the ROD module. bit 7:0 → TTC type

Return values

If no errors occur WriteTtype( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

13.12 ROD_Error ReadStatus (u_int* value, int mode)

Description

This method reads in the TTC Status register of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the TTC Status register of the ROD module.

Return values

If no errors occur ReadStatus( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

13.13 void PrintStatus (u_int value)

Description

This method decodes and prints to the standard output the information read in the TTC Status register of the ROD module.

Input values

NAME	DESCRIPTION
u_int value	Word read from the TTC Status register of the ROD module.

13.14 ROD_Error WriteTtypeSubAddress (u_int data, int mode)

Description

This method writes in the TTC Type Sub Address register of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
u_int data	32-bit word to be written in the TTC Type Sub Address register of ROD modules.

Return values

If no errors occur WriteTtypeSubAddress( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

13.15 ROD_Error ReadTtypeSubAddress (u_int* value, int mode)

Description

This method reads in the TTC Type Sub Address register of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the TTC Ttype Sub Address register of ROD module.

Return values

If no errors occur ReadTtypeSubAddress( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

13.16 void PrintTtypeSubAddress (u_int value)

Description

This method decodes and prints to the standard output the information read in the TTC Ttype Sub Address register of the ROD module.

Input values

NAME	DESCRIPTION
u_int value	Word read from the TTC Ttype Sub Address register of the ROD module.

13.17 ROD_Error ReadVersion (u_int* value, int mode)

Description

This method reads in the TTC Version register of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the TTC Version register of ROD module.

Return values

If no errors occur ReadVersion( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

13.18 void PrintVersion (u_int value)

Description

This method decodes and prints to the standard output the information read in the TTC Version register of the ROD module.

Input values

NAME	DESCRIPTION
u_int value	Word read from the TTC Version register of the ROD module.

13.19 ROD_Error SendTTCEvent (u_int TTYPE, u_int EVID, u_int BCID, int mode)

Description

This method writes in the TTC Type, Event ID and Bunch Crossing registers of the ROD module.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
u_int BCID	32-bit word to be written in the TTC Bunch Crossing ID register.
u_int EVID	32-bit word to be written in the TTC Event ID register.
u_int TTYPE	32-bit word to be written in the TTC Type register.

Return values

If no errors occur SendTTCEvent( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

14 Class STAGING

FILES

RODStag.h

RODStag.cc

PUBLIC VARIABLES

NAME	DESCRIPTION
int id	Staging FPGA identifier. The ***id*** can be 0, 1, 2, or 3. It must be set in order to access the registers of the corresponding staging FPGA.
VMEMasterMap* p_My_Vmm	Pointer to an object of the VMEMasterMap class.

The following methods will be applied to the corresponding staging FPGA which is selected by the ***id*** public variable. If the user does not set the ***id*** variable, the method will return the error code ROD_FAIL_STAG.

PUBLIC METHODS

14.1 STAGING ( )

Description

Constructor of the class, it reserves memory space for private pointers.

14.2 ~STAGING ( )

Description

Destructor of the class. It frees the memory space reserved for private pointers.

14.3 ROD_Error WriteDummy (u_int data, int mode)

Description

This method writes in a Dummy register of the selected Staging FPGA.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written in a Dummy register of the selected Staging FPGA. Depending on the setting of the variable class member ***id*** it will access to the corresponding Staging FPGA.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteDummy( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

14.4 ROD_Error ReadDummy (u_int* value, int mode)

Description

This method reads in the Dummy register of the selected Staging FPGA.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from a Dummy register of the selected Staging FPGA.

Return values

If no errors occur ReadDummy( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

14.5 void PrintDummy (u_int value)

Description

This method decodes and prints to the standard output the information read in the Dummy register of the selected Staging FPGA.

Input values

NAME	DESCRIPTION
u_int value	Word read from a Dummy register of the selected Staging FPGA.

14.6 ROD_Error WriteSetReset (u_int data, int mode)

Description

This method set and reset the selected Staging FPGA.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to set and/or reset the selected Staging FPGA. Depending on the setting of the variable class member ***id*** it will access to the corresponding Staging FPGA.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteSetReset( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

14.7 ROD_Error ReadStatus (u_int* value, int mode)

Description

This method reads in the Status register of the selected Staging FPGA.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from a Status register of the selected Staging FPGA.

Return values

If no errors occur ReadStatus( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

14.8 void PrintStatus (u_int value)

Description

This method decodes and prints to the standard output the information read in the Status register of the selected Staging FPGA.

Input values

NAME	DESCRIPTION
u_int value	Word read from a Status register of the selected Staging FPGA.

14.9 ROD_Error WriteConfig1 (u_int data, int mode)

Description

This method writes in the Configuration register 1 of the selected Staging FPGA.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written in the Configuration register 1 of the selected Staging FPGA. Depending on the setting of the variable class member ***id*** it will access to the corresponding Staging FPGA.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteConfig1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

14.10 ROD_Error ReadConfig1 (u_int* value, int mode)

Description

This method reads in Configuration register 1 of the selected Staging FPGA.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the Configuration register 1 of the selected Staging FPGA.

Return values

If no errors occur ReadConfig1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

14.11 void PrintConfig1 (u_int value)

Description

This method decodes and prints to the standard output the information read in the Configuration register 1 of the selected Staging FPGA.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Configuration register 1 of the selected Staging FPGA.

14.12 ROD_Error WriteConfig2 (u_int data, int mode)

Description

This method writes in the Configuration register 2 of the selected Staging FPGA.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written in the Configuration register 2 of the selected Staging FPGA. Depending on the setting of the variable class member ***id*** it will access to the corresponding Staging FPGA.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteConfig2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

14.13 ROD_Error ReadConfig2 (u_int* value, int mode)

Description

This method reads in the Configuration register 2 of the selected Staging FPGA.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the Configuration register 2 of the selected Staging FPGA.

Return values

If no errors occur ReadConfig2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

14.14 void PrintConfig2 (u_int value)

Description

This method decodes and prints to the standard output the information read in the Configuration register 2 of the selected Staging FPGA.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Configuration register 2 of the selected Staging FPGA.

14.15 ROD_Error WriteStartAdd (u_int data, int mode)

Description

This method writes the initial address where data is sent to the RAM of the selected Staging FPGA. This version only allows 512 words, so, address should be written in a word of 9 bits (8:0).

Input values

NAME	DESCRIPTION
u_int data	9 bit addressed can be written for setting the initial address of the RAM of the selected Staging FPGA. Depending on the setting of the variable class member ***id*** it will access to the corresponding Staging FPGA.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteStartAdd( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

14.16 ROD_Error ReadStartAdd (u_int* value, int mode)

Description

Method to read the initial address of the RAM of the selected Staging FPGA.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	32-bit word with the information of the initial address of the RAM of the selected Staging FPGA.

Return values

If no errors occur ReadStartAdd( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

14.17 void PrintStartAdd (u_int value)

Description

This method prints to the standard output the initial address of the RAM of the selected Staging FPGA.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Start Address register of the selected Staging FPGA.

14.18 ROD_Error WriteDataRam (u_int data, int mode)

Description

This method writes a word in the memory RAM of the selected Staging FPGA.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written in memory RAM of the selected Staging FPGA. Depending on the setting of the variable class member ***id*** it will access to the corresponding Staging FPGA.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteDataRam( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

14.19 ROD_Error ReadDataRam (u_int* value, int mode)

Description

This method reads in the memory RAM of the selected Staging FPGA.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from memory RAM of the selected Staging FPGA.

Return values

If no errors occur ReadDataRam( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

14.20 void PrintDataRam (u_int value)

Description

This method decodes and prints to the standard output the information read in the memory RAM of the selected Staging FPGA.

Input values

NAME	DESCRIPTION
u_int value	Word read from the memory RAM of the selected Staging FPGA.

14.21 ROD_Error StartTransmission (int mode)

Description

This method starts the transmission in the selected Staging FPGA.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur StartTransmission( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

14.22 ROD_Error WriteLinkConfig (u_int data, int mode)

Description

This method writes the link configuration of the selected Staging FPGA.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written in the Link Configuration of the selected Staging FPGA. Depending on the setting of the variable class member ***id*** it will access to the corresponding Staging FPGA.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteLinkConfig( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

14.23 ROD_Error ReadLinkConfig (u_int* value, int mode)

Description

This method reads the link configuration of the selected Staging FPGA.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from memory RAM of the selected Staging FPGA.

Return values

If no errors occur ReadLinkConfig( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

14.24 void PrintLinkConfig (u_int value)

Description

This method decodes and prints to the standard output the information read in Link Configuration register of the selected Staging FPGA.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Link Configuration register of the selected Staging FPGA.

14.25 ROD_Error ReadTemperature (int num, int* value, int mode)

Description

This method reads the Temperature of the selected G-Link. With the input parameter ***num***, the user can read up to 8 temperature values from the 8 G-links of the ROD module. Never mind through which Staging FPGA the user reads the temperature, because this method always accesses the same registers.

Input values

NAME	DESCRIPTION
int num	G-Link temperature identifier. This parameter can take values from '0' to '7' for reading the temperature of the eight G-links of the ROD module.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word for reading the Temperature of the selected G-Link.

Return values

If no errors occur ReadTemperature( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

14.26 void PrintTemperature (u_int value)

Description

This method decodes and prints to the standard output the information of the Temperature of the selected G-Link.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Temperature register of the selected G-Link. When it is decoded the user will read the maximum, the minimum and the actual temperature.

14.27 Void ConvertTemperature (u_int input_value, double* voltage, double* resistance, double* temperature)

Description

This method converts the G-Link temperature in ADC counts to temperature in Celsius degrees.

Input values

NAME	DESCRIPTION
u_int input_value	G-Link temperature in ADC counts.

Output values

NAME	DESCRIPTION
double * voltage	Pointer to the measured voltage.
double * resistance	Pointer to the measured resistance.
double * temperature	Pointer to the temperature in Celsius degrees.

14.28 ROD_Error ReadVersion (int* value, int mode)

Description

This method reads the Version of the selected Staging FPGA.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read for reading the version of the selected Staging FPGA.

Return values

If no errors occur ReadVersion( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

14.29 void PrintVersion (u_int value)

Description

This method decodes and prints to the standard output the information of the Version of the selected Staging FPGA.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Version register of the selected Staging FPGA.

15 Class FPGA_PU

FPGA PU DESCRIPTION

The FPGA PU does not apply any online reconstruction algorithm. A picture of the FPGA PU is shown in Figure 4, the core for dataflow management is implemented in the main chip (Altera APEX20kE FPGA) of the card, while two external FIFOs have the task of data buffering from the Processing Unit to the Output FPGA.

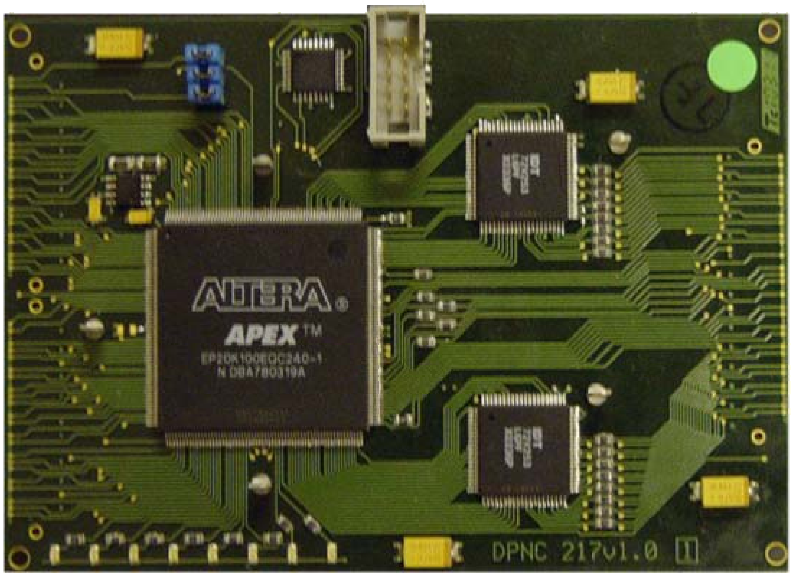

Figure 4: Picture of the Dummy/FPGA PU.

FILES

RODFpga_pu.h

RODFpga_pu.cc

PUBLIC VARIABLES

NAME	DESCRIPTION
int id	Dummy/FPGA Processing Unit Identifier. The ***id*** can be 0, 1, 2, or 3. It must be set in order to access to the registers of the corresponding Dummy/FPGA Processing Unit.
VMEMasterMap* p_My_Vmm	Pointer to an object of the VMEMasterMap classs.

The following methods will be applied to the corresponding FPGA Processing Unit which selected by the ***id*** public variable. If the user does not set the ***id*** variable, the method will return the error code ROD_FAIL_FPGA_PU.

PUBLIC METHODS

15.1 FPGA_PU ( )

Description

Constructor of the class, it reserves memory space for private pointers.

15.2 ~FPGA_PU ( )

Description

Destructor of the class. It frees the memory space reserved for private pointers.

15.3 ROD_Error WriteDummy (u_int data, int mode)

Description

This method writes in the Dummy register of the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written in the Dummy register of the selected Dummy/FPGA Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding Dummy/FPGA PU.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteDummy( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.4 ROD_Error ReadDummy (u_int* value, int mode)

Description

This method reads in the Dummy register of the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the Dummy register of the selected FPGA Processing Unit.

Return values

If no errors occur ReadDummy( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.5 void PrintDummy (u_int value)

Description

This method decodes and prints to the standard output the information read in the Dummy register of the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word read from a Dummy register of the selected DUMMY/FPGA Processing Unit.

15.6 ROD_Error Reset (u_int data, int mode)

Description

This method sends a reset to the selected DUMMY/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written in a Reset register of the selected Dummy/FPGA Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding Dummy/FPGA Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur Reset( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.7 ROD_Error ReadStatus (u_int* value, int mode)

Description

This method reads in the Status register of the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from a Status register of the selected Dummy/FPGA Processing Unit.

Return values

If no errors occur ReadStatus( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.8 void PrintStatus (u_int value)

Description

This method decodes and prints to the standard output the information read in the Status register of the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word read from a Status register of the selected Dummy/FPGA Processing Unit.

15.9 ROD_Error WriteConfig (u_int data, int mode)

Description

This method writes in a Configuration register of the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written in a Configuration register of the selected Dummy/FPGA Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding Dummy/FPGA Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteConfig( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.10 ROD_Error ReadConfig (u_int* value, int mode)

Description

This method read in the Configuration register of the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from a Configuration register of the selected Dummy/FPGA Processing Unit.

Return values

If no errors occur ReadConfig( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.11 void PrintConfig (u_int value)

Description

This method decodes and prints to the standard output the information read in the Configuration register of the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word read from a Configuration register of the selected Dummy/FPGA Processing Unit.

15.12 ROD_Error ReadFifo (u_int* value, int mode)

Description

This method reads the Fifo of the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the Fifo of the selected Dummy/FPGA Processing Unit.

Return values

If no errors occur ReadFifo( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.13 void PrintFifo (u_int value)

Description

This method decodes and prints to the standard output the information read from the Fifo of the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Fifo of the selected Dummy/FPGA Processing Unit.

15.14 ROD_Error WriteSetResetBusy (u_int data, int mode)

Description

This method writes in the Pulse and Set/Reset Busy register of the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written in the Pulse and Set/Reset Busy register of the selected Dummy/FPGA Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding Dummy/FPGA Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteSetResetBusy( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.15 ROD_Error ReadSetResetBusy (u_int* value, int mode)

Description

This method reads in the Pulse and Set/Reset Busy register of the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read from the Pulse and Set/Reset Busy register of the selected Dummy/FPGA Processing Unit.

Return values

If no errors occur ReadSetResetBusy( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.16 void PrintSetResetBusy (u_int value)

Description

This method decodes and prints to the standard output the information read in the Pulse and Set/Reset register of the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Pulse and Set/Reset Busy register of the selected Dummy/FPGA Processing Unit.

15.17 ROD_Error WriteFormatVersionNumber (u_int data, int mode)

Description

This method writes the Format Version Number for the data in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written to set the Format Version Number of the data in the selected Dummy/FPGA Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding Dummy/FPGA Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteFormatVersionNumber( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.18 ROD_Error ReadFormatVersionNumber (u_int* value, int mode)

Description

This method read the Format Version Number of the data in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word with the Format Version Number the information of the data in the selected Dummy/FPGA Processing Unit.

Return values

If no errors occur ReadFormatVersionNumber( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.19 void PrintFormatVersionNumber (u_int value)

Description

This method decodes and prints to the standard output the Format Version Number of the data in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word with the Format Version Number information of the data in the selected Dummy/FPGA Processing Unit.

15.20 ROD_Error WriteSouceID _FEB1 (u_int data, int mode)

Description

This method writes the Source Identifier for the data in the FEB 1 in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written to set the Source Identifier of the data in the FEB 1 in the selected Dummy/FPGA Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding Dummy/FPGA Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteSourceID_FEB1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.21 ROD_Error ReadSourceID_FEB1 (u_int* value, int mode)

Description

This method reads the Source Identifier of the data in the FEB 1 in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word with the Source Identifier information of the data in the FEB 1 in the selected Dummy/FPGA Processing Unit.

Return values

If no errors occur ReadSourceID_FEB1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.22 void PrintSourceID_FEB1 (u_int value)

Description

This method decodes and prints to the standard output the Source Identifier of the data in the FEB 1 in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word with the Source Identifier information of the data in the FEB 1 in the selected Dummy/FPGA Processing Unit.

15.23 ROD_Error WriteSouceID _FEB2 (u_int data, int mode)

Description

This method writes the Source Identifier for the data in the FEB 2 in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written to set the Source Identifier of the data in the FEB 2 in the selected Dummy/FPGA Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding Dummy/FPGA Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteSourceID_FEB2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.24 ROD_Error ReadSourceID_FEB2 (u_int* value, int mode)

Description

This method reads the Source Identifier of the data in the FEB 2 in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word with the Source Identifier information of the data in the FEB 2 in the selected Dummy/FPGA Processing Unit.

Return values

If no errors occur ReadSourceID_FEB2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.25 void PrintSourceID_FEB2 (u_int value)

Description

This method decodes and prints to the standard output the Source Identifier of the data in the FEB 2 in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word with the Source Identifier information of the data in the FEB 2 in the selected Dummy/FPGA Processing Unit.

15.26 ROD_Error WriteRunNumber (u_int data, int mode)

Description

This method writes the Run Number for the data in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written to set the Run Number of the data in the selected Dummy/FPGA Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding Dummy/FPGA Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteRunNumber( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.27 ROD_Error ReadRunNumber (u_int* value, int mode)

Description

The method reads the Run Number of the data in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word read the Run Number of the data in the selected Dummy/FPGA Processing Unit.

Return values

If no errors occur ReadRunNumber( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.28 void PrintRunNumber (u_int value)

Description

This method decodes and prints to the standard output the Run Number of the data in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word with the Run Number information of the data in the selected Dummy/FPGA Processing Unit.

15.29 ROD_Error WriteExtendedL1ID (u_int data, int mode)

Description

This method writes the Extended L1ID (Level 1 Identifier) for the data in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written to set the Extended L1ID (Level 1 Identifier) of the data in the selected Dummy/FPGA Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding Dummy/FPGA Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteExtendedL1ID( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.30 ROD_Error ReadExtendedL1ID (u_int* value, int mode)

Description

This method reads the Extended L1ID (Level 1 Identifier) of the data in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word with the Extended L1ID (Level 1 Identifier) information of the data in the selected Dummy/FPGA Processing Unit.

Return values

If no errors occur ReadExtededL1ID( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.31 void PrintExtendedL1ID (u_int value)

Description

This method decodes and prints to the standard output the Extended L1ID (Level 1 Identifier) of the data in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word with the Extended L1ID (Level 1 Identifier) information of the data in the selected Dummy/FPGA Processing Unit.

15.32 ROD_Error WriteBCID_L1_Ttype (u_int data, int mode)

Description

This method writes the BCID (Bunch Crossing Identifier) and L1 Ttype (Level 1 Trigger type) for the data in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written to set the BCID and L1 Ttype of the data in the selected Dummy/FPGA Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding Dummy/FPGA Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteBCID_L1_Ttype( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.33 ROD_Error ReadBCID_L1_Ttype (u_int* value, int mode)

Description

This method reads the BCID and L1 Ttype of the data in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word with the BCID and L1 Ttype information of the data in the selected Dummy/FPGA Processing Unit.

Return values

If no errors occur ReadBCID_L1_Ttype( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.34 void PrintBCID_L1_Ttype (u_int value)

Description

This method decodes and prints to the standard output the BCID and L1 Ttype of the data in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word with the BCID and L1 Ttype information of the data in the selected Dummy/FPGA Processing Unit.

15.35 ROD_Error WriteDetectorEventType (u_int data, int mode)

Description

This method writes the Detector Event Type for the data in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written to set the Detector Event Type of the data in the selected Dummy/FPGA Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding Dummy/FPGA Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteDetectorEventType( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.36 ROD_Error ReadDetectorEventType (u_int* value, int mode)

Description

This method reads the Detector Event Type of the data in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word with the Detector Event Type information of the data in the selected Dummy/FPGA Processing Unit.

Return values

If no errors occur ReadDetectorEventType( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.37 void PrintDetectorEventType (u_int value)

Description

This method decodes and prints to the standard output the Detector Event Type of the data in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word with the Detector Event Type information of the data in the selected Dummy/FPGA Processing Unit.

15.38 ROD_Error WriteWordsPerEvent (u_int data, int mode)

Description

This method writes the number of words sent per event in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written to set the number of words sent per event in the selected Dummy/FPGA Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding Dummy/FPGA Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteWordsPerEvent( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.39 ROD_Error ReadWordsPerEvent (u_int* value, int mode)

Description

This method reads the number of words sent per event in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word with the number of words sent per event in the selected Dummy/FPGA Processing Unit.

Return values

If no errors occur ReadWordsPerEvent( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.40 void PrintWordsPerEvent (u_int value)

Description

This method decodes and prints to the standard output the number of words sent per event in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word with the number of words sent per event in the selected Dummy/FPGA Processing Unit.

15.41 ROD_Error ReadEvents_FEB1 (u_int* value, int mode)

Description

This method reads the Events from the FEB 1 (Front-End Board) in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word with the Events from the FEB 1 in the selected Dummy/FPGA Processing Unit.

Return values

If no errors occur ReadEvents_FEB1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.42 void PrintEvents_FEB1 (u_int value)

Description

This method decodes and prints to the standard output the Events from the FEB 1 in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word with Events from the FEB 1 in the selected Dummy/FPGA Processing Unit.

15.43 ROD_Error ReadEvents_FEB2 (u_int* value, int mode)

Description

This method reads the Events from the FEB 2 (Front-End Board) in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word with the Events from the FEB 2 in the selected Dummy/FPGA Processing Unit.

Return values

If no errors occur ReadEvents_FEB2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.44 void PrintEvents_FEB2 (u_int value)

Description

This method decodes and prints to the standard output the Events from the FEB 2 in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word with Events from the FEB 2 in the selected Dummy/FPGA Processing Unit.

15.45 ROD_Error ReadLinkErrorCounterGLink1 (u_int* value, int mode)

Description

This method reads the number of times a link error signal was asserted in the G-Link 1 of the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word with the number of link errors asserted in the G-Link 1 of the selected Dummy/FPGA Processing Unit.

Return values

If no errors occur ReadLinkErrorCounterGLink1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.46 void PrintLinkErrorCounterGLink1 (u_int value)

Description

This method decodes and prints to the standard output the number of link errors asserted in the G-Link 1 of the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word with the number of link errors asserted in the G-Link 1 of the selected Dummy/FPGA Processing Unit.

15.47 ROD_Error ReadLinkReadyCounterGLink1 (u_int* value, int mode)

Description

This method reads the number of times a link ready signal was lost in the G-Link 1 of the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word with the number of times a link ready signal was lost in the G-Link 1 of the selected Dummy/FPGA Processing Unit.

Return values

If no errors occur ReadLinkReadyCounterGLink1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.48 void PrintLinkReadyCounterGLink1 (u_int value)

Description

This method decodes and prints to the standard output the number of times a link ready signal was in the G-Link 1 of the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word with the number of times a link ready signal was lost in the G-Link 1 of the selected Dummy/FPGA Processing Unit.

15.49 ROD_Error ReadLinkErrorCounterGLink2 (u_int* value, int mode)

Description

This method reads the number of times a link error signal was asserted in the G-Link 2 of the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word with the number of link errors asserted in the G-Link 2 of the selected Dummy/FPGA Processing Unit.

Return values

If no errors occur ReadLinkErrorCounterGLink2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.50 void PrintLinkErrorCounterGLink2 (u_int value)

Description

This method decodes and prints to the standard output the number of link errors asserted in the G-Link 2 of the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word with the number of link errors asserted in the G-Link 2 of the selected Dummy/FPGA Processing Unit.

15.51 ROD_Error ReadLinkReadyCounterGLink2 (u_int* value, int mode)

Description

This method reads the number of times a link ready signal was lost in the G-Link 2 of the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word with the number of times a link ready signal was lost in the G-Link 2 of the selected Dummy/FPGA Processing Unit.

Return values

If no errors occur ReadLinkReadyCounterGLink2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.52 void PrintLinkReadyCounterGLink2 (u_int value)

Description

This method decodes and prints to the standard output the number of times a link ready signal was in the G-Link 2 of the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word with the number of times a link ready signal was lost in the G-Link 2 of the selected Dummy/FPGA Processing Unit.

15.53 ROD_Error WriteSubFragmentID_FEB1 (u_int data, int mode)

Description

This method writes the Sub fragment Identifier in data from FEB 1 in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written to set the Sub fragment Identifier in data from FEB 1 in the selected Dummy/FPGA Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding Dummy/FPGA Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteSubFragmentID_FEB1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.54 ROD_Error ReadSubFragmentID_FEB1 (u_int* value, int mode)

Description

This method reads the Sub fragment Identifier in data from FEB 1 in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word with the Sub fragment Identifier in data from FEB 1 in the selected Dummy/FPGA Processing Unit.

Return values

If no errors occur ReadSubFragmentID_FEB1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.55 void PrintSubFragmentID_FEB1 (u_int value)

Description

This method decodes and prints to the standard output the Sub fragment Identifier in data from FEB 1 in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word with the Sub fragment Identifier in data from FEB 1 in the selected Dummy/FPGA Processing Unit.

15.56 ROD_Error WriteSubFragmentID_FEB2 (u_int data, int mode)

Description

This method writes the Sub fragment Identifier in data from FEB 2 in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written to set the Sub fragment Identifier in data from FEB 2 in the selected Dummy/FPGA Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding Dummy/FPGA Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteSubFragmentID_FEB2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.57 ROD_Error ReadSubFragmentID_FEB2 (u_int* value, int mode)

Description

This method reads the Sub fragment Identifier in data from FEB 2 in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word with the Sub fragment Identifier in data from FEB 2 in the selected Dummy/FPGA Processing Unit.

Return values

If no errors occur ReadSubFragmentID_FEB2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.58 void PrintSubFragmentID_FEB2 (u_int value)

Description

This method decodes and prints to the standard output the Sub fragment Identifier in data from FEB 2 in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word with the Sub fragment Identifier in data from FEB 2 in the selected Dummy/FPGA Processing Unit.

15.59 ROD_Error WriteVersion (u_int data, int mode)

Description

This method writes the Version in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written to set the Version in the selected Dummy/FPGA Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding Dummy/FPGA Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteVersion( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.60 ROD_Error ReadVersion (u_int* value, int mode)

Description

This method reads the Version in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word with the Version in the selected Dummy/FPGA Processing Unit.

Return values

If no errors occur ReadVersion( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

15.61 void PrintVersion (u_int value)

Description

This method decodes and prints to the standard output the Version in the selected Dummy/FPGA Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word with the Version in the selected Dummy/FPGA Processing Unit.

16 Class DSP_PU

DSP PU DESCRIPTION

The DSP PU does apply online reconstruction algorithms as Optimal Filtering [10]. The DSP PU (Figure 5) is a mezzanine card with the dimensions of 120 × 85 mm. It is composed of two blocks, each one can process up to 48 read-out channels in normal mode and 96 read-out channels in staging mode (with half the number of DSP PUs in the motherboard). Each DSP block is composed of an input FPGA Cyclone EP1C6, a TMS320C6414 DSP from Texas Instruments and an external output FIFO. The DSP PU also contains an output FPGA Cyclone EP1C6 used for VME and TTC interface.

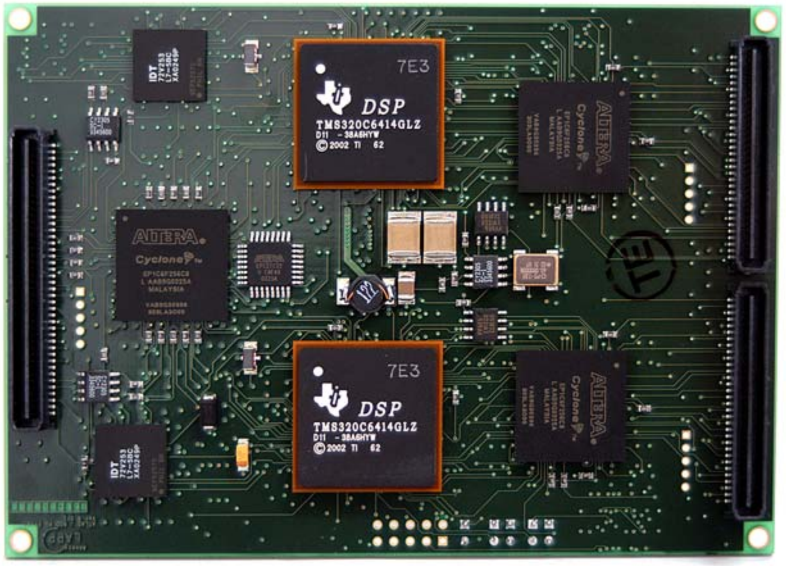

Figure 5: Picture of the DSP PU.

The input FPGAs and the DSPs can be programmed via VME. As soon as the board is powered the EEPROM uploads the code into the output FPGA which allows VME access to the whole board. Now the input FPGA code and the DSP code can be uploaded through the VME interface. The class DSP_PU contains methods for this booting sequence.

FILES

RODDsp_pu.h

RODDsp_pu.cc

PUBLIC VARIABLES

NAME	DESCRIPTION
int id	DSP Processing Unit Identifier. The ***id*** can be 0, 1, 2, or 3. It must be set in order to access to the registers of the corresponding DSP Processing Unit.
VMEMasterMap* p_My_Vmm	Pointer to an object of the VMEMasterMap classs.

The following methods will be applied to the corresponding DSP Processing Unit, selected by the ***id*** public variable. If the user does not set the ***id*** variable, the method will return ROD_FAIL_DSP_PU.

The DSP class is more complex than the other classes. It incorporates thus methods to boot the chips and methods to access the PU registers (as the other classes). The booting can be done by calling the method Config( ), Figure 6 shows an application which configures the PU for the booting.

```
#include "TileVmeROD/RODIncludes.h"
using namespace RODTile;

int usage( ){
    cout<<"Usage: BootPU <InFpgaFile> <InDspFile>"<<endl;
    return (true);
}

int main(int argc, char * argv[]) {
    if (argc != 3) return(usage());
    char * InFpgaFile;
    char * DspFile;

    InFpgaFile = argv[1];
    DspFile    = argv[2];

    CRATE* mycrate = new CRATE;
    ERROR* myerror = new ERROR;
    ROD_Error ret;

    int rods = 1; // CRATE INIT
    int auxs = 0;
    int device_name = VP110;
    int mode = SAFE;
    ret = mycrate->CrateInit(rods, auxs, device_name);
    if (ret != ROD_SUCCESS) {
       myerror->RODErrorPrint(ret);
       return(false);
    }

    int pus  = 1; // INIT ROD
    int slot = 12;
    ret = mycrate->pRod[0]->RODInit(pus, slot);
    if (ret != ROD_SUCCESS) {
       myerror->RODErrorPrint(ret);
       return (false);
    }

    cout<<"\nPU slot (1..4):   "<<endl; // PU select
    int pu_slot=1;
    cin>>pu_slot;
    cout<<"pu_slot ="<<pu_slot<<endl;

    cout<<"PU booting..."<<endl;
    cout<<"  BOOT all PU       -> 0"<<endl;
    cout<<"  BOOT PU up part   -> 1"<<endl;
    cout<<"  BOOT PU down part -> 2"<<endl;
    int FlagPuUpDown;
    cin>>FlagPuUpDown;
    int FlagDspLaunch = 1;

    // Set pDspPu[0] to id number 3 (the last pu in the module)
    // by default this id number is 0 (the first one)
    ret = mycrate->pRod[0]->pDspPu[0]->id = (pu_slot-1);
    ret = mycrate->pRod[0]->pDspPu[0]->Config(mode,FlagPuUpDown,InFpgaFile,
DspFile, FlagDspLaunch);
    if (ret != ROD_SUCCESS){
        myerror->RODErrorPrint(ret);
        mycrate->CrateShutDown();
        return (false);
    }

mycrate->CrateShutDown();
return (true);

}
```

Figure 6: Example of how to boot the DSP PUs.

PUBLIC METHODS

16.1 DSP_PU ( )

Description

Constructor of the class, it reserves memory space for private pointers.

16.2 ~DSP_PU ( )

Description

Destructor of the class. It frees the memory space reserved for private pointers.

16.3 ROD_Error GeneralReset (int mode)

Description

This method resets all the components of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur GeneralReset( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.4 ROD_Error StartSendData (int mode)

Description

This method configures the two DSPs of the selected PU to start send data.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur StartSendData( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.5 ROD_Error Config (int mode, int FlagPuUpDown, char* InFpgaFile, char* DspFile, int FlagDspLaunch)

Description

This method configures the selected DSP Processing Unit. Internally Config( ) calls several methods to boot the components of the processing units, as the DSP processors and the FPGA chips.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
int FlagIPuUpDown	PU_UP(1), PU_DOWN(2) and PU_BROADCAST(0) The whole PU can be booted simultaneously (PU_BROADCAST) or separately, i.e the up part, Input FPGA 1 and DSP 1, (PU_UP) and the down part, Input FPGA 2 and DSP 2, (PU_DOWN).
char* InFpgaFile	Name of the file to be loaded into the FPGA chips.
char* DspFile	Name of the file to be loaded into the DSP chips.
int FlagDspLaunch	DSP_LAUNCH_NOSTART(0) and DSP_LAUNCH_START(1). If FlagDspLaunch is set to DSP_LAUNCH_NOSTART the program is loaded into the DSP but it does not start running, otherwise if it is set to DSP_LAUNCH_START the program starts after loading.

Return values

If no errors occur Config( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.6 ROD_Error ReadDspVersion1 (u_int* value, int mode)

Description

This method reads the Version of the code loaded in the DSP 1 of the selected Processing Unit.

Input values

NAME	DESCRIPTION

int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word with the Version of the code loaded in the DSP 1 of the selected Processing Unit.

Return values

If no errors occur ReadDspVersion1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.7 void PrintDspVersion1 (u_int value)

Description

This method decodes and prints to the standard output the Version of the code loaded in the DSP 1 of the selected Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word with the Version of the code loaded in the DSP 1 of the selected DSP Processing Unit.

16.8 ROD_Error ReadDspVersion2 (u_int* value, int mode)

Description

This method reads the Version of the code loaded in the DSP 2 of the selected Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word with the Version of the code loaded in the DSP 2 of the selected DSP Processing Unit.

Return values

If no errors occur ReadDspVersion2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.9 void PrintDspVersion2 (u_int value)

Description

This method decodes and prints to the standard output the Version of the code loaded in the DSP 2 of the selected Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word with the Version of the code loaded in the DSP 2 of the selected DSP Processing Unit.

16.10 ROD_Error WriteDummy1 (u_int data, int mode)

Description

This method writes in the Dummy register of the Output FPGA of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written in the Dummy register of the Output FPGA of the selected DSP Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding DSP Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteDummy1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.11 ROD_Error ReadDummy1 (u_int* value, int mode)

Description

This method reads in the Dummy register of the Output FPGA of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word in the Dummy register of the Output FPGA of the selected DSP Processing Unit.

Return values

If no errors occur ReadDummy1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.12 void PrintDummy1 (u_int value)

Description

This method decodes and prints to the standard output the word read in a Dummy register of the Output FPGA of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Dummy register of the Output FPGA of the selected DSP Processing Unit.

16.13 ROD_Error WriteDummy2 (u_int data, int mode)

Description

This method writes in the Dummy register of the Output FPGA of the selected DSP Processing Unit

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written in the Dummy register of the Output FPGA of the selected DSP Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding DSP Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteDummy2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.14 ROD_Error ReadDummy2 (u_int* value, int mode)

Description

This method reads in the Dummy register of the Output FPGA of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word in the Dummy register of the Output FPGA of the selected DSP Processing Unit.

Return values

If no errors occur ReadDummy2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.15 void PrintDummy2 (u_int value)

Description

This method decodes and prints to the standard output the word read in the Dummy register of the Output FPGA of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Dummy register of the Output FPGA of the selected DSP Processing Unit.

16.16 ROD_Error WriteControl1 (u_int data, int mode)

Description

This method writes in the Control register of the part 1 of the selected DSP Processing Unit. By writting in this register a reset of the DSP 1, the Input FPGA 1 and FIFO 1 can be done.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written in the Control register 1 of the selected DSP Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding Dummy/FPGA Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteControl1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.17 ROD_Error ReadControl1 (u_int* value, int mode)

Description

This method reads in the Control register 1 of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word in the Control register 1 of the selected DSP Processing Unit.

Return values

If no errors occur ReadControl1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.18 void PrintControl1 (u_int value)

Description

This method decodes and prints to the standard output the word read in the Control register 1 of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Control register 1 of the selected DSP Processing Unit.

16.19 ROD_Error WriteControl2 (u_int data, int mode)

Description

This method writes in the Control register of the part 2 of the selected DSP Processing Unit. By writting in this register a reset of the DSP 2, the Input FPGA 2 and FIFO 2 can be done.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written the Control register 2 of the selected DSP Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding Dummy/FPGA Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteControl2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.20 ROD_Error ReadControl2 (u_int* value, int mode)

Description

This method reads in the Control register 2 of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word in the Control register 2 of the selected DSP Processing Unit.

Return values

If no errors occur ReadControl2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.21 void PrintControl2 (u_int value)

Description

This method decodes and prints to the standard output the word read in the Control register 2 of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Control register 2 of the selected DSP Processing Unit.

16.22 ROD_Error ReadStatus1 (u_int* value, int mode)

Description

This method reads in the Status register 1 of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word in the Status register 1 of the selected DSP Processing Unit.

Return values

If no errors occur ReadStatus1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.23 void PrintStatus2 (u_int value)

Description

This method decodes and prints to the standard output the word read in the Status register 2 of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Status register 2 of the selected DSP Processing Unit.

16.24 ROD_Error ReadStatus2 (u_int* value, int mode)

Description

This method reads in the Status register 2 of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit word in the Status register 2 of the selected DSP Processing Unit.

Return values

If no errors occur ReadStatus2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.25 void PrintStatus2 (u_int value)

Description

This method decodes and prints to the standard output the word read in the Status register 2 of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Status register 2 of the selected DSP Processing Unit.

16.26 ROD_Error WriteSerial1 (u_int data, int mode)

Description

This method writes Serial Data to McBSP2 (DSP1) of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
u_int data	32-bit data word written to the McBSP2 (DSP1) of the selected DSP Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding Dummy/FPGA Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteSerial1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.27 ROD_Error ReadSerial1 (u_int* value, int mode)

Description

This method reads Serial Data from McBSP2 (DSP1) of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit data word from the McBSP2 (DSP1) of the selected DSP Processing Unit.

Return values

If no errors occur ReadSerial1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.28 void PrintSerial1(u_int value)

Description

This method decodes and prints to the standard output the data from the McBSP2 (DSP1) of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word read from the McBSP1 (DSP1) of the selected DSP Processing Unit.

16.29 ROD_Error WriteSerial2 (u_int data, int mode)

Description

This method writes Serial Data to McBSP2 (DSP2) of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
u_int data	32-bit data word written to the McBSP2 (DSP2) of the selected DSP Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding Dummy/FPGA Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteSerial2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.30 ROD_Error ReadSerial2 (u_int* value, int mode)

Description

This method reads Serial Data from McBSP2 (DSP2) of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit data word from the McBSP2 (DSP2) of the selected DSP Processing Unit.

Return values

If no errors occur ReadSerial2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.31 void PrintSerial2(u_int value)

Description

This method decodes and prints to the standard output the data from the McBSP2 (DSP2) of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word read from the McBSP2 (DSP2) of the selected DSP Processing Unit.

16.32 ROD_Error WriteInFpgaConfig1 (u_int data, int mode)

Description

This method configures the Input FPGA 1 of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
u_int data	32-bit word, where only the 16 LSB are useful, for configuring the Input FPGA 1 of the selected DSP Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding Dummy/FPGA Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteInFpgaConfig1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.33 ROD_Error WriteInFpgaConfig2 (u_int data, int mode)

Description

This method configures the Input FPGA 2 of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
u_int data	32-bit word, where only the 16 LSB are useful, for configuring the Input FPGA 2 of the selected DSP Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding Dummy/FPGA Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteInFpgaConfig2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.34 ROD_Error WriteInFpgaProgram1 (u_int data, int mode)

Description

This method allows programming the Input FPGA 1 through VME of the selected DSP Processing Unit. The previous method Config( ) calls this method when booting the Input FPGA 1.

Input values

NAME	DESCRIPTION
u_int data	32-bit word, to program the Input FPGA 1 of the selected DSP Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding Dummy/FPGA Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteInFpgaProgram1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.35 ROD_Error WriteInFpgaProgram2 (u_int data, int mode)

Description

This method allows programming the Input FPGA 2 through VME of the selected DSP Processing Unit. The previous method Config( ) calls this method when booting the Input FPGA 1.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to program the Input FPGA 2 of the selected DSP Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding Dummy/FPGA Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteInFpgaProgram2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.36 ROD_Error ReadInFpgaStatus1 (u_int* value, int mode)

Description

This method reads the Status register of the Input FPGA 1 of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit data word from Status register of the Input FPGA 1 of the selected DSP Processing Unit.

Return values

If no errors occur ReadInFpgaStatus1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.37 void PrintInFpgaStatus1 (u_int value)

Description

This method decodes and prints to the standard output the data from the Status register of the Input FPGA 1 of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Status register of the Input FPGA 1 of the selected DSP Processing Unit.

16.38 ROD_Error ReadInFpgaStatus2 (u_int* value, int mode)

Description

This method reads the Status register of the Input FPGA 2 of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit data word from Status register of the Input FPGA 2 of the selected DSP Processing Unit.

Return values

If no errors occur ReadInFpgaStatus2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.39 void PrintInFpgaStatus2 (u_int value)

Description

This method decodes and prints to the standard output the data from the Status register of the Input FPGA 2 of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Status register of the Input FPGA 2 of the selected DSP Processing Unit.

16.40 ROD_Error WriteInFpgaProgramBroadcast1 (u_int data, int mode)

Description

This method allows programming through VME both the Input FPGA 1 and the Input FPGA 2 of the selected DSP Processing Unit. The previous method Config( ) calls this method when booting both Input FPGAs at the same time.

Input values

NAME	DESCRIPTION
u_int data	32-bit word, to program both the Input FPGA 1 and the Input FPGA 2 of the selected DSP Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding DSP Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteInFpgaProgramBroadcast1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.41 ROD_Error WriteInFpgaProgramBroadcast2 (u_int data, int mode)

Description

This method allows programming through VME both the Input FPGA 1 and the Input FPGA 2 of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
u_int data	32-bit word, to program both the Input FPGA 1 and the Input FPGA 2 of the selected DSP Processing Unit. Depending on the setting of the variable class member ***id*** it will access to the corresponding DSP Processing Unit.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteInFpgaProgramBroadcast2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.42 ROD_Error ReadOutFpgaVersion1 (u_int* value, int mode)

Description

This method reads the version of the firmware of the Output FPGA of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit data word from the Version register 1 of the selected DSP Processing Unit.

Return values

If no errors occur ReadOutFpgaVersion1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.43 void PrintOutFpgaVersion1 (u_int value)

Description

This method decodes and prints to the standard output the version of the Output FPGA of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Version register 1 of the selected DSP Processing Unit.

16.44 ROD_Error ReadOutFpgaVersion2 (u_int* value, int mode)

Description

This method reads the version of the firmware of the Output FPGA of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit data word from the Version register 2 of the selected DSP Processing Unit.

Return values

If no errors occur ReadOutFpgaVersion2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.45 void PrintOutFpgaVersion2 (u_int value)

Description

This method decodes and prints to the standard output the version of the Output FPGA of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word read from the Version register 2 of the selected DSP Processing Unit.

16.46 ROD_Error WriteLoadInFpga1 (int mode, char * InFpgaFile)

Description

This method reads the file InFpgaFile and uploads it to the Input FPGA 1 of the selected DSP PU using the method WriteInFpgaProgram1( ).

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
char* InFpgaFile	Name of the file to be uploaded in the Input FPGA 1 of the selected DSP.

Return values

If no errors occur WriteLoadInFpga1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.47 ROD_Error WriteLoadInFpga2 (int mode, char * InFpgaFile)

Description

This method reads the file InFpgaFile and uploads it to the Input FPGA 2 of the selected DSP PU using the method WriteInFpgaProgram2( ).

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
char* InFpgaFile	Name of the file to be uploaded in the Input FPGA 2 of the selected DSP.

Return values

If no errors occur WriteLoadInFpga2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.48 ROD_Error WriteLoadInFpgaBroadcast (int mode, char * InFpgaFile)

Description

This method reads the file InFpgaFile and uploads it to the Input FPGA 1 and the Input FPGA 2 of the selected DSP PU using the method WriteInFpgaProgramBroadcast1( ).

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
char* InFpgaFile	Name of the file to be uploaded in the Input FPGA 1 and the Input FPGA 2 of the selected DSP.

Return values

If no errors occur WriteLoadInFpgaBroadcast( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.49 ROD_Error WriteLoadDsp1 (int mode, char * DspFile, int flagLaunch)

Description

This method uploads the file DspFile into the DSP 1 of the selected DSP PU using the method WriteHPI1(int mode, char * DspFile), and configures the DSP to start running the program if the flagLaunch is set to DSP_LAUNCH_START.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
char* DspFile	Name of the file to be uploaded in the DSP 1 of the selected DSP.

int flagLaunch	Can be set to:
	DSP_LAUNCH_START = 1
	DSP_LAUNCH_NOSTART = 0

Return values

If no errors occur WriteLoadDsp1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.50 ROD_Error WriteHPI1 (int mode, char * DspFile)

Description

This method reads the file DspFile which will be upload into the DSP 1 of the selected DSP PU using the method WriteHPI1(int mode, u_int addr, u_int * DspData, const int dimDspData).

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
char* DspFile	Name of the file to be uploaded in the DSP 1 of the selected DSP.

Return values

If no errors occur WriteWriteHPI1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.51 ROD_Error WriteHPI1 (int mode, u_int addr, u_int * DspData, const int dimDspData)

Description

This method uses the method WriteHPI1(u_int data, int mode) to set the DSP address where the program is written. And then uploads the program, which is accessible through the pointer DspData.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
u_int addr	Address in the DSP's memory space where the program is written.
u_int * DspData	Pointer to the DSP code.
const int dimDspData	Dimension of the pointer DspData.

Return values

If no errors occur WriteWriteHPI1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.52 ROD_Error WriteHPI1 (u_int data, int mode)

Description

This method writes through the Host Port Interface (HPI) of the DSP 1, which is a 16-bit wide bus that can directly access the DSP's memory space

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written through the Host Port Interface of the DSP 1 of the selected DSP PU.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteWriteHPI1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.53 ROD_Error ReadHPI1 (u_int* value, int mode)

Description

This method reads through the Host Port Interface of the DSP 1 of the selected DSP PU.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit data word read from the HPI of the DSP 1 of the selected DSP Processing Unit.

Return values

If no errors occur ReadHPI1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.54 void PrintHPI1 (u_int value)

Description

This method decodes and prints to the standard output the word read from the HPI of the DSP 1 of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word read through the HPI of the DSP 1 of the selected DSP Processing Unit.

16.55 ROD_Error WriteLoadDsp2 (int mode, char * DspFile, int flagLaunch)

Description

This method uploads the file DspFile into the DSP 2 of the selected DSP PU using the method WriteHPI2( ), and configures the DSP to start running the program if the flagLaunch is set to DSP_LAUNCH_START.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
char* DspFile	Name of the file to be uploaded in the DSP 2 of the selected DSP.

int flagLaunch	Can be set to:
	DSP_LAUNCH_START = 1
	DSP_LAUNCH_NOSTART = 0

Return values

If no errors occur WriteLoadDsp2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.56 ROD_Error WriteHPI2 (int mode, char * DspFile)

Description

This method reads the file DspFile which will be upload into the DSP 2 of the selected DSP PU using the method WriteHPI2(int mode, u_int addr, u_int * DspData, const int dimDspData).

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
char* DspFile	Name of the file to be uploaded in the DSP 2 of the selected DSP.

Return values

If no errors occur WriteWriteHPI2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.57 ROD_Error WriteHPI2 (int mode, u_int addr, u_int * DspData, const int dimDspData)

Description

This method uses the method WriteHPI2(u_int data, int mode) to set the DSP address where the program is written. And then uploads the program, which is accessible through the pointer DspData.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
u_int addr	Address in the DSP's memory space where the program is written.
u_int * DspData	Pointer to the DSP code.
const int dimDspData	Dimension of the pointer DspData.

Return values

If no errors occur WriteWriteHPI2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.58 ROD_Error WriteHPI2 (u_int data, int mode)

Description

This method writes through the Host Port Interface of the DSP 2, which is a 16-bit wide bus that can directly access the DSP's memory space

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written through the Host Port Interface of the DSP 2 of the selected DSP PU.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteWriteHPI2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.59 ROD_Error ReadHPI2 (u_int* value, int mode)

Description

This method reads through the Host Port Interface of the DSP 2 of the selected DSP PU.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Output values

NAME	DESCRIPTION
u_int * value	Pointer to a 32-bit data word read from the HPI of the DSP 2 of the selected DSP Processing Unit.

Return values

If no errors occur ReadHPI2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.60 void PrintHPI2 (u_int value)

Description

This method decodes and prints to the standard output the word read from the HPI of the DSP 2 of the selected DSP Processing Unit.

Input values

NAME	DESCRIPTION
u_int value	Word read through the HPI of the DSP 2 of the selected DSP Processing Unit.

16.61 ROD_Error WriteLoadDspBroadcast (int mode, char * DspFile, int flagLaunch)

Description

This method uploads the file DspFile into the DSPs 1 and 2 of the selected DSP PU using the method WriteHPIBroadcast(int mode, char * DspFile), and configures the DSPs to start running the programs if the flagLaunch is set to DSP_LAUNCH_START.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
char* DspFile	Name of the file to be uploaded in the DSPs 1 and 2 of the selected DSP.

int flagLaunch	Can be set to:
	DSP_LAUNCH_START = 1
	DSP_LAUNCH_NOSTART = 0

Return values

If no errors occur WriteLoadDspBroadcast( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.62 ROD_Error WriteHPIBroadcast (int mode, char * DspFile)

Description

This method reads the file DspFile which will be upload into the DSPs 1 and 2 of the selected DSP PU using the method WriteHPIBroadcast(int mode, u_int addr, u_int * DspData, const int dimDspData).

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
char* DspFile	Name of the file to be uploaded in the DSPs 1 and 2 of the selected DSP.

Return values

If no errors occur WriteWriteHPIBroadcast( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.63 ROD_Error WriteHPIBroadcast (int mode, u_int addr, u_int * DspData, const int dimDspData)

Description

This method uses the method WriteHPIBroadcast1(u_int data, int mode) to set the DSP address where the program is written. And then uploads the program, which is accessible through the pointer DspData. The program is uploaded in both DSP 1 and DSP 2 of the selected DSP PU.

Input values

NAME	DESCRIPTION
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.
u_int addr	Address in the DSP's memory space where the program is written.
u_int * DspData	Pointer to the DSP code.
const int dimDspData	Dimension of the pointer DspData.

Return values

If no errors occur WriteWriteHPIBroaccast( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.64 ROD_Error WriteHPIBroadcast1 (u_int data, int mode)

Description

This method writes through the Broadcast HPI register of the DSP 1 of the selected DSP PU.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written through the Broadcast Host Port Interface of the DSP 1 of the selected DSP PU.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteWriteHPIBroadcast1( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

16.65 ROD_Error WriteHPIBroadcast2 (u_int data, int mode)

Description

This method writes through the Broadcast HPI register of the DSP 2 of the selected DSP PU.

Input values

NAME	DESCRIPTION
u_int data	32-bit word to be written through the Broadcast Host Port Interface of the DSP 2 of the selected DSP PU.
int mode	SAFE (0) and FAST (1) The FAST mode does not return any error message.

Return values

If no errors occur WriteWriteHPIBroadcast2( ) returns ROD_SUCCESS otherwise returns the corresponding error code.

References

[1] ATLAS Collaboration. Tile Calorimeter Technical Design Report.
Technical Report, CERN/LHCC 96-42, CERN, 1996.

[2] Tile Calorimeter Read Out Driver. Firmware Developments for the Final Prototype.
J. Castelo *et al.,* September 2004.
Proceedings of LECC 2004.

TileCal ROD Hardware and Software Requirements.
J. Castelo *et al.,* February 2005.
ATLAS Internal Note, ATL-TILECAL-2005-003.

[3] The TMS320C6414 DSP Mezzanine Board.
J. Prast. August 2004.
ATLAS EDMS document ATL-AL-EN-0051.

[4] TTC website: http://ttc.web.cern.ch/TTC/intro.html .

[5] Doxygen website: http://www.doxygen.org/index.html .

[6] Technical Reference Manual for VP 110/01x VME Pentium® III-M Single Board Computer.
Manual Order Code 550 0014 Rev 02, August 2002.

[7] SBS$_{TM}$ Technologies, Model 946, 965, 983, 993 & 1003 Support Software.

[8] VMEbus Application Program Interface.
R. Spiwoks *et al.,* June 2002.
ATLAS Internal Note, ATL-D-ES-0004.

[9] Standalone Software for TileCal ROD Characterization and System Tests.
J. Poveda *et al.,* December 2004.
ATLAS Internal Note, ATL-TILECAL-2004-012.

[10] Optimal Filtering in the ATLAS Hadronic Tile Calorimeter.
E. Fullana et al., January 2005.
ATLAS Internal Note, ATL-TILECAL-2005-001.