MICROGATE ENGINEERING

KECK NEXT GENERATION WAVEFRONT CONTROLLER

Real Time Controller
As Built Design Document

Document :	NGWFC_RTC_ASB_001.doc	
Issue :	1	
Date :	September 2nd, 2007	
Prepared by :	MICROGATE	
	R.Biasi	.....
	D.Pescoller	.....
	M.Andrighettoni	.....
Checked by :		.....
Approved by :		.....
Released by :		.....

September 2nd , 2007

CHANGE RECORDS

ISSUE	DATE	Author	Approved	QA/ QC	SECTION / PARAG. AFFECTED	REASON/INITIATION DOCUMENTS/REMARKS
1	2007.09.02	Microgate			All	First Issue

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 ACRONYMS

AO Adaptive Optics
CCD Charge Coupled Device
CIE Command Interpreter and Executer
COTS Commercial Off-The-Shelf
DDR Double Data Rate
DM Deformable Mirror
DMA Direct Memory Access
DSP Digital Signal Processor
DTT Down Tip Tilt
DTTM Down Tip Tilt Mirror
FC-IP FibreChannel Internet Protocol
FITs Number of Failures in 10^9 hours
FPDP Front Panel Data Port
GPIB General Purpose Interface Bus
HBA Host Adapter Board
HP Width unit for 19" chassis, corresponding to 0.2" (5.08mm)
HV High Voltage
HVA High Voltage Amplifier
HVC High Voltage Control
ICMP Internet Control Message Protocol
IIR Infinite Impulse Response
LFpM Linear Feet per Minute
LAN Local Area Network
LGS Laser Guide Star
LUT Look Up Table
MAC Multiply And Accumulate
mas milliarcseconds
MGAOS Microgate Adaptive Optics real-time System
MIMO Multiple Input Multiple Output
MIL-STD military standard
MMF Multi-Mode Fiber
NDA Non Disclosure Agreement
NFS Network File System
NGS Natural Guide Star
NGWFC Next Generation Wavefront Controller
PCB Printed Circuit Board
PIO Programmable Input Output
PSU Power Supply Unit

RMS	Root-Mean-Square
RTC	Real Time Controller
SAN	Storage Area Network
SAS	Serial Attached SCSI
SCSI	Small Computer System Interface
SFP	Small Form factor Pluggable
SI	The International System of Units
SH	Shack-Hartmann
SRAM	Static Random Access Memory
SDRAM	Synchronous Dynamic Random Access Memory
STRAP	System for Tip-tilt Removal with Avalanche Photo-diodes
TBC	To Be Confirmed
TBD	To Be Defined
TDP	Telemetry Data Packet
TRS	Telemetry Recorder/Server
U	Height unit for 19” chassis, corresponding to 1.75” (44.45mm)
UTT	Uplink Tip Tilt
UTTM	Uplink Tip Tilt Mirror
VME	VersaModule Eurocard
WBS	Work Breakdown Structure
WCP	Wavefront Controller Command Processor
WIF	Wavefront Controller Interface
WFP	Wavefront processor
WFS	Wavefront Sensor

2 APPLICABLE DOCUMENTS

[AD1] CARA/W.M. Keck
NGWFC RTC Requirements – Keck Adaptive optics note #311. Version 1.0, March 11th, 2005

[AD2] CARA/W.M. Keck
NGWFC RTC Tip-Tilt Requirements – Keck Adaptive optics note #329. Version 1.0, May 25th, 2005

[AD3] CARA/W.M. Keck
NGWFC RTC Vendor Statement of Work – Keck Adaptive optics note #310. Version 1.0, March 11th, 2005

[AD4] CARA/W.M. Keck
NGWFC System Design Manual – Keck Adaptive optics note #289. Version 2.0, August 15th, 2005

[AD5] Microgate S.r.l.
Real Time Controller Preliminary Design Review Data Package
Issue 1 – August 22nd, 2005

[AD6] CARA/W.M. Keck
Request for change to the NGWFC RTC: Post PDR updates - Keck Adaptive optics note #354.
Version 1.4, Novemeber 3rd, 2005

[AD7] CARA/W.M. Keck
NGWFC RTC Acceptance Test Plan - Keck Adaptive optics note #374

[AD8] CARA/W.M. Keck
NGWFC Detailed Design Report - Keck Adaptive optics note #371
December 2nd, 2005

[AD9] Microgate S.r.l.
Real Time Controller Detailed Design Review Data Package
Issue 1 – December 2nd, 2005

[AD10] Microgate S.r.l.
NGWFC Test_Report 07/02/06 and 07/11/06,
Issue 1 – July 11th, 2006

[AD11] Microgate S.r.l.
memory map Keck_NGWFC 1_05.xls
Spreadsheet containing the control system memory mapping.
Issue 1.05, September 2007

3 REFERENCE DOCUMENTS

[RD1] R. Biasi, M.Andrighettoni et al. - 'Dedicated flexible electronics for adaptive secondary control', - SPIE Proc. on 'Advancements in Adaptive Optics', 5490, p.1502

[RD2] E-mails exchanged between Microgate and CARA-Keck between April 21st ad May 7th, 2005

[RD3] Department of Defense USA, MIL-HDBK-217 Revision F, Reliability Prediction of Electronic Equipment

[RD4] Keck AO Wavefront Control –Hardware Manual

[RD5] INCITS - FibreChannel – Physical and Signaling Interface – ANSI – INCITS 230-1994

[RD6] M. Rajagopal, R. Bhagwat, W. Rickard - RFC 2625 - IP and ARP over FibreChannel - June 1999

4 INTRODUCTION

This document contains the As Built Data Package of those parts of the NGFWC Real Time Controller that are under responsibility of Microgate.

The document derives directly from the Detailed Design Review report ([AD9]).

Even if the entire document has been thoroughly reviewed, addressing in particular performance analysis and design details, we report hereafter a list of the major additions and modifications with respect to the original document.

- Reset circuitry description . The reset handling circuit was moved from the VME backplane to the MGAOS crate
- Assessment of hardware interfaces
- Chopping functionality
- CLMP loop tuning
- Software design
- WIF-WCP interface
- Command Interpreter and Executer
- TRS database (here we introduced a major change with respect to PDR)
- Results of preliminary tests on TRS performance
- Validation of real-time computation performance estimate through testing of algorithms and C versus assembler comparison
- Assessment of interface test between RTC and TRS including test of IP over FibreChannel communication

An ’Annex’ section has been added at the end of the document, including:

- Low level MGAOS/MVME communication protocol (MGP)
- Final choice of COTS components (jointly with CARA)
- Schematics of all proprietary boards, including component placement and bill of material (these are reported as links to external documents)

5 SYSTEM OVERVIEW

In this section we present a general overview of the NGWFC RTC design.

In Figure 1 we present a global block diagram of the designed architecture.

The main system functional blocks of NGWFC RTC are:

- WaveFront Processor (WFP): it is the hearth of the wavefront controller, performing all real-time computations from pixel acquisition to generation of commands for the DM and DTT/UTT mirrors.
- Wavefront controller Interface (WIF): this is the interface between external commands (from WCP) to the WFP.
- DTT/UTT controllers, implementing closed loop control of the piezoelectric-actuated DTT and UTT mirrors.
- STRAP tip-tilt sensor and controller.
- Telemetry Recorder Server (TRS): it is the main diagnostic storage system, receiving streams of data from WFP, STRAP, WIF and WCP.

All this components can be clearly distinguished in the block diagram of Figure 1, describing also the hardware devices and interfaces where the functions above are actually implemented.

The ***WFP*** is implemented by the Microgate Adaptive Optics real-time System (MGAOS). This is a proprietary hardware architecture that was developed in the frame of other adaptive optics projects, with the aim of providing a well-optimized and scalable high performance computational and control architecture. All MGAOS components are mounted on a single proprietary bus and the whole system is fit into a small case integrated in the VME crate (see Figure 2). The MGAOS is directly interfaced to the Wavefront Sensor Camera (standard AIA interface) and to the DM HVA (FPDP-like interface). The ***DTT/UTT controllers*** are also integrated in the MGAOS, thus a direct interface between MGAOS and DTT/UTT mirrors is provided.

The ***WIF*** is implemented on a standard VME architecture (PowerPC VME 6100 CPU board). The WIF is the interface between WCP and WFP. It also acts as supervisor of all RTC operations. In particular, all setup and configuration of MGAOS are performed by the VME CPU through a private Gbit Ethernet connection. The same interface allows also transferring real-time data from ***STRAP*** to MGAOS.

The ***TRS*** comprehends a storage data server and a large disk array for data storage, capable of storing the system telemetry for longer than five observing nights. It is interfaced to MGAOS through a dedicated FibreChannel link, through which all bulky telemetry data are transferred. STRAP and configuration diagnostics is transferred directly from the VME CPU via a standard Gbit Ethernet LAN. The client workstation can query the TRS by means of the same standard LAN.

An IRIG board installed on the VME crate provides system synchronization to an absolute timing reference.

An external reset line allows to reset the RTC remotely.

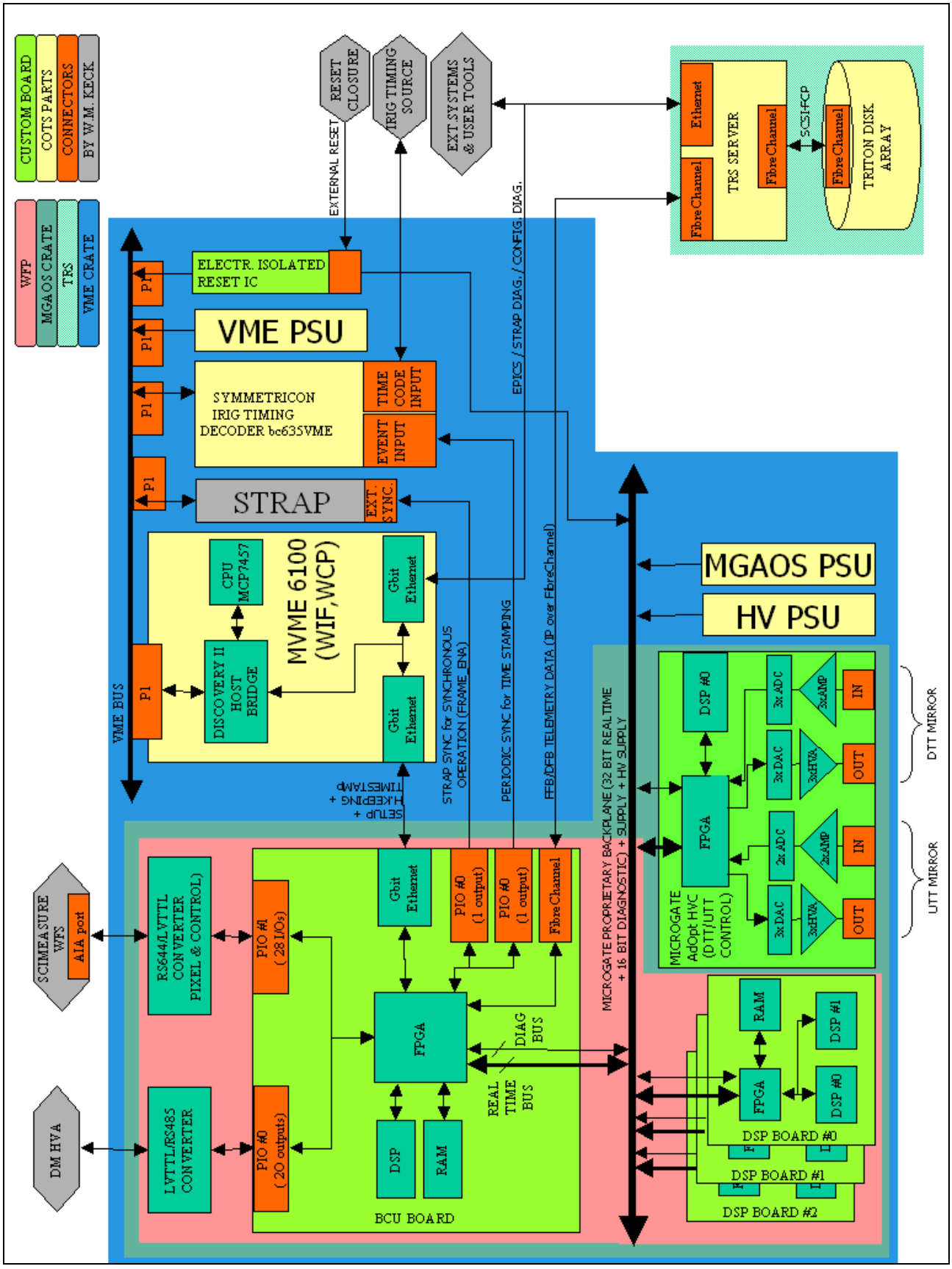


Figure 1 – NGWFC RTC global architecture

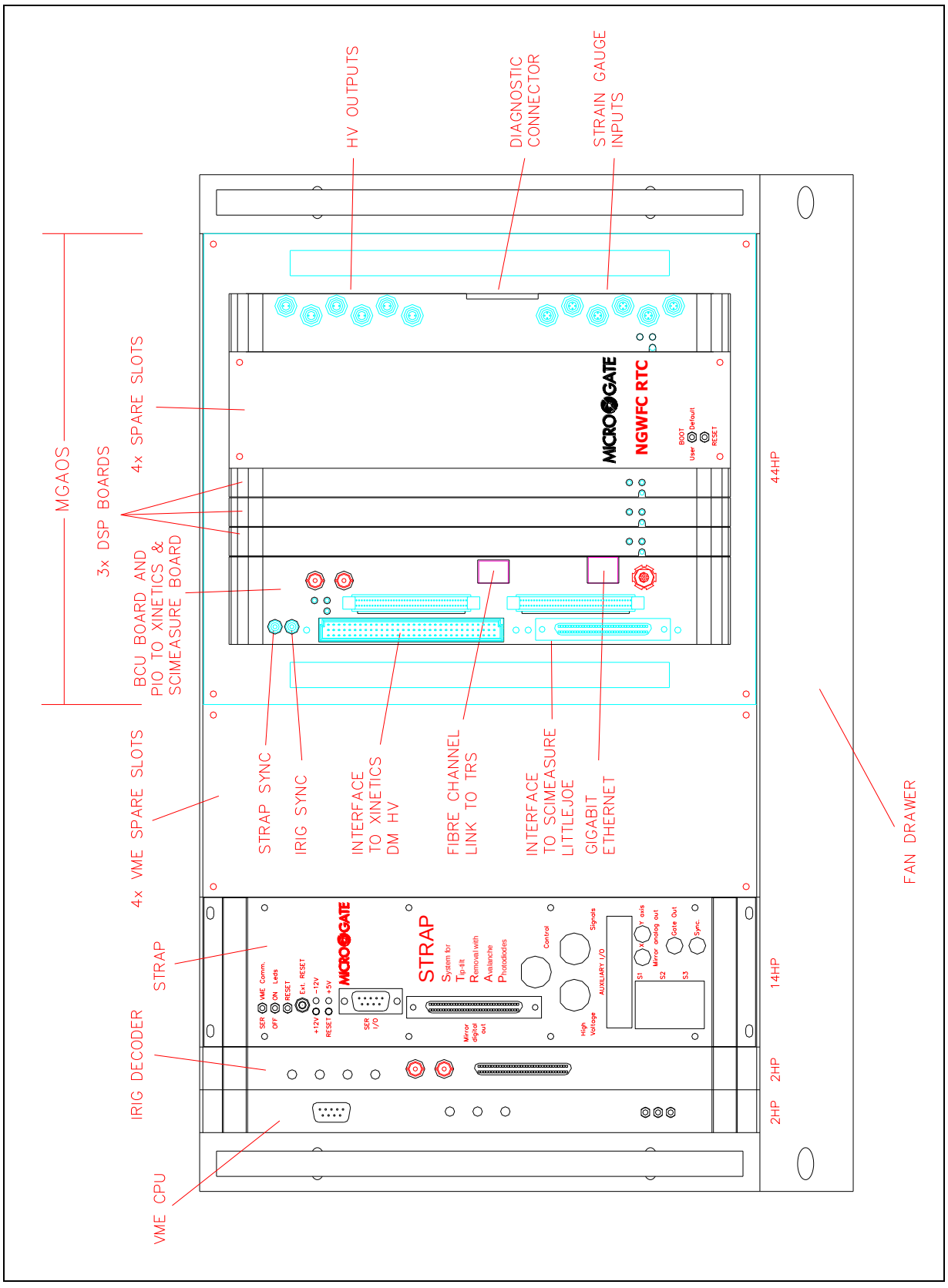


Figure 2 – NGWFC RTC: VME and MGAOS crate mechanical layout

6 HARDWARE DESIGN

In this section we describe the HW design of the NGWFC RTC. To this aim, the main system components are analyzed, namely WIF/WFP hardware, TRS and storage disk array. The subsystem main components, in particular the boards comprehended in the MGAOS system, are described in detail. We analyze also the design/simulation/prototyping results of DTT/UTT HVC board. The design process of this dedicated hardware component has been completed, including experimental tests on a single-channel breadboard. The construction will start immediately after DDR.

Finally, we describe the internal and external system interfaces.

6.1 WIF/WFP Hardware

The WFP is the real-time part of the NGWFC RTC. It acquires the pixels data from the WFS, computes the centroids, generates the residual wavefront and generates the commands for both deformable mirror and DTT/UTT tip-tilt mirrors.

The WIF is the interface between the real-time system and WCP.

Both WIF and WFP hardware are hosted on a single crate, where we can distinguish the following main components:

- Microgate crate (MGAOS), comprehending also the interfaces to WFS, DM drive and DTT/UTT mirrors. All strictly real-time WFP functions are implemented by the MGAOS
- Host VME CPU board
- IRIG timing board
- STRAP
- Power supply system

6.1.1 Microgate Adaptive Optics crate (MGAOS)

The MGAOS implements all real time functions of the WFP. The system is based on the Microgate proprietary ‘AdOpt’ hardware. The proposed configuration comprehends:

- 1 AdOpt BCU board (§6.1.1.1), implementing several functions:
  - input real-time interface to the SciMeasure controller
  - pixel background and flat field compensation
  - centroid computation
  - output real-time interface to the Xinetics controller
  - bus arbiter and master of the data exchange between different processors on the backplane (in particular, transferring centroids to the DSP boards and retrieving DM commands)
  - interface to the VME crate for real-time data transfer of STRAP data and system setup, through a dedicated Gbit Ethernet port
  - telemetry data interface to the TRS through FibreChannel port
- 3 AdOpt DSP boards (§6.1.1.2), each comprehending 2 TigerSharc DSPs (ADSP TS101) with 300 MHz internal clock rate, capable of 0.6 Giga-floating point multiply-and-accumulate per second (sustained performance, each DSP). The DSP boards implement the most extensive computational tasks, namely the residual wavefront computation and the control law servo computation.

- 1 AdOpt HVC board (§6.1.1.3), capable of controlling digitally in closed loop up to six axes based on piezoelectric transducers with strain gauge position feedback. The HVC board is used to drive directly both the DTT and UTT mirrors.
- Passive proprietary backplane (§6.1.1.4), based on B-LVDS technology. This is the backbone of each half-crate. Power supply, diagnostic signals, real time communication and diagnostic communication are distributed by the backplane. The backplane can host up to 12 boards, while the current configuration uses 7 slots.
- Reset Circuitry to handle properly the external reset, the VME reset and the local reset pushbutton (§6.1.1.5)

The design philosophy of the AdOpt components is aimed to realize a very efficient real-time parallel computer, with particular focus on inter-processor communication.

In the following sections we describe in more detail the architecture of the main MGAOS components.

6.1.1.1 AdOpt BCU board

The BCU (Basic Computational Unit) board is a general purpose board, capable of different tasks within an adaptive optics system. It is mainly dedicated to route and manage the communication within a single Microgate AdOpt crate or among several crates or subsystems.

In the NGWFC RTC application, the BCU acts as sequencer of the different computational tasks, it manages the real-time data exchange among the MGAOS DSPs and handles the telemetry buffering and interface.

The on-board DSP handles the pixel background, flat field compensation and centroids computation tasks.

In the following section we give an overview of the main board components. A block diagram of the BCU is presented in Figure 3.

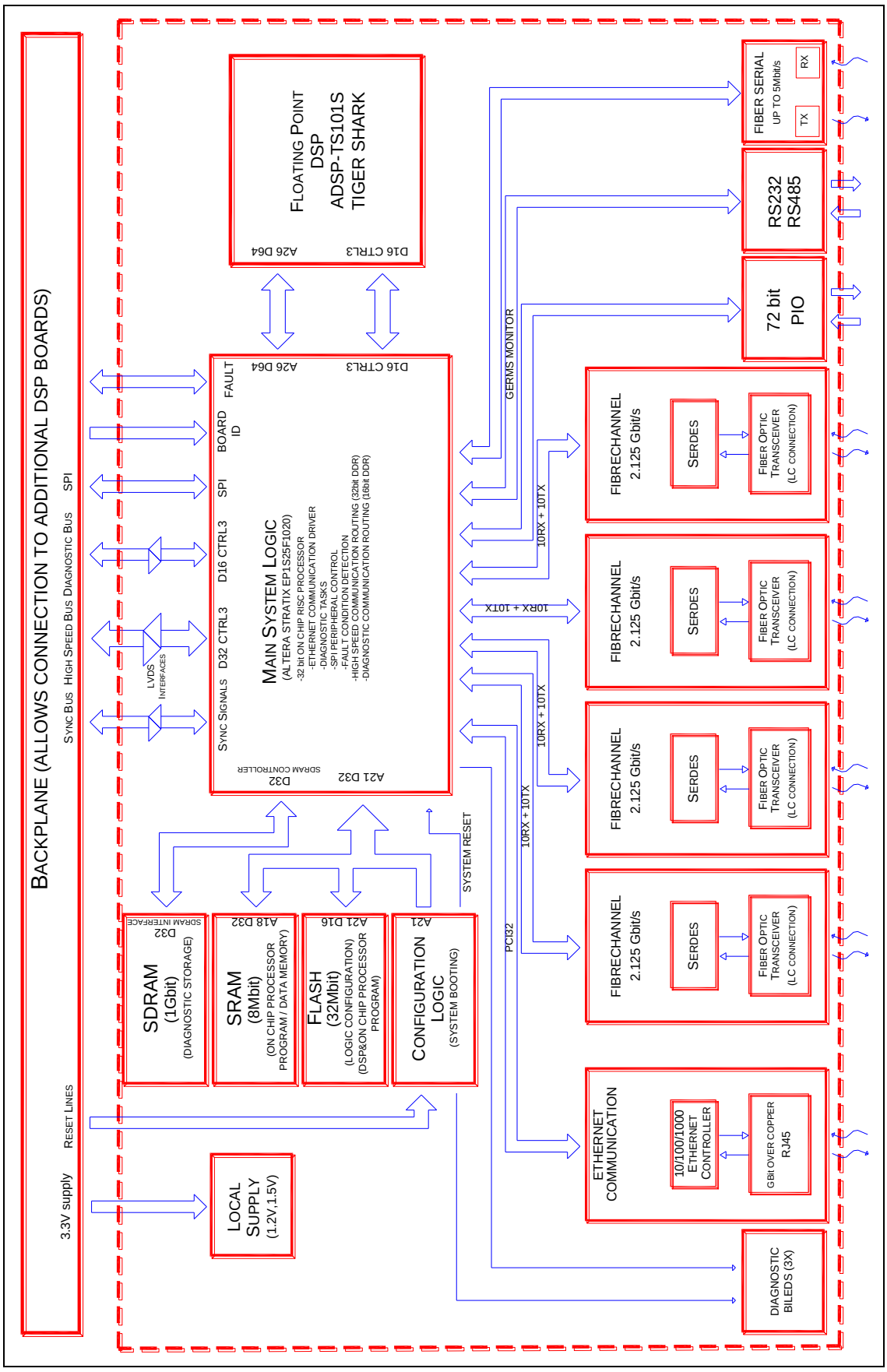


Figure 3 – BCU board block scheme

6.1.1.1.1 Main system logic

The Main System Logic is realized with an Altera Startix EP1S25F1020 programmable logic. It performs the following main tasks:

- interface between all on board devices;
- interface with the backplane buses;
- interface to the PIO (see 6.1.1.1.11). This ports provides a versatile interface to different devices. In our application it is used to read frames from the WFS and to send commands to the DM HVA
- on chip 'soft' processor (Altera Nios II), which performs:
- driver of the Ethernet diagnostic link
- acquisition of all local diagnostic devices (board temperatures)
- transfers telemetry data from DSPs to SDRAM and then to TRS

6.1.1.1.2 Main real time computational unit

The computational unit is based on one ADSP-TS101S DSPs.

The DSPs communicate with the main logic by two shared buses and some direct lines:

- the high speed bus has 64 data bits at ~60MHz of frequency with a typical throughput of 4Gbit per second;
- the diagnostic bus uses two 8 data bits parallel serial lines (called link ports) in DDR mode at ~60MHz of frequency, with a typical throughput of 2Gbit per second;
- additional ports are used by the system logic to synchronize the DSP operation with the external tasks

Within the MGAOS, the BCU DSP performs pixel background, flat field compensation and centroid computation.

6.1.1.1.3 Reconfiguration logic

The reconfiguration logic is realized with an Altera EPM3128. It is a non volatile programmable logic which has the task of downloading the configuration of the main logic and the main logic on-chip processor code from the non volatile Flash.

A safety mechanism is provided in order to guarantee that the system can be always recovered from any wrong configuration. The configuration process runs according to the following steps:

- after reset, the configuration logic starts reading the 'user' configuration from the Flash memory
- if the user configuration is not recognized by the logic as a valid configuration (validation mechanisms are factory embedded in the logic configuration loading circuitry), the configuration logic automatically loads the default configuration. This configuration is stored into a protected area of the Flash memory. Thus, it is not possible to damage accidentally the default configuration through the Ethernet connection.
- If the user configuration is recognized by the system logic as a valid configuration, but it does not work properly (as it might happen frequently during system debug), it is possible to force the default configuration by means of a dedicated hardware signal, which is available to the user (bus_boot_select)

The configuration logic also manages all the reset signals to all the devices that could be reset: Flash, main logic, DSP and Ethernet chip.

6.1.1.1.4 Non volatile storage memory (FLASH)

The non volatile memory is necessary at system booting to configure the system. The memory used is a typical flash memory with a programmable write locked area with a total size of 32Mbit.

The main logic configuration and the on-chip program code are duplicated into two different areas: the **default** and **user** area.

The default configuration and program are placed in the locked area to avoid accidental overwriting. This configuration permits to restart the system if the user configuration or program code have been corrupted or they don't work properly.

The selection of the booting mode is done either automatically, if the user mode configuration fails, or by the external hardware signal bus_boot_select.

The DSP program code area contains the DSP program. It can be automatically downloaded at system booting. In the MGAOS application, however, it is foreseen to upload the DSP code from an external database at every system startup. This task is performed by the VME CPU.

The configuration parameters is a fixed area which contains all the software parameters that define the system board configuration after start up. They are programmable using the diagnostic interface (Ethernet) and are maintained until new configuration parameters are written.

The flash memory is organized as follow:

Start address(byte)	size(byte)	locked	Description
0x000000	0x180000	yes	main logic default configuration
0x180000	0x040000	yes	on-chip default program code
0x1C0000	0x040000	no	on-chip user program code
0x200000	0x180000	no	main logic user configuration
0x380000	0x070000	no	not used area
0x3F0000	0x00A000	no	configuration parameters
0x3FA000	0x006000	no	not used area

Table 1 – Flash memory mapping

The connection between the main logic and the flash it realized by a 16bit data bus at 60 MHz of frequency with a typical throughput of 1Gbit per second. This bus is shared with the SRAM bus because, usually, the flash device is used just at start up.

6.1.1.1.5 SRAM memory

The 8Mbit SRAM memory is dedicated only to the main logic on-chip program and data memory. The organization of the memory depends on the on-chip program code.

The connection between the main logic and the SRAM is realized by a 32bit data bus at 60 MHz of frequency with a typical throughput of 2Gbit per second. This bus is shared with the FLASH device because during ordinary operation the FLASH is never used and then the bus is free to be used with the SRAM.

6.1.1.1.6 SDRAM memory

A bulk memory is provided on the BCU board for diagnostic data storage. The bulk memory available is 1Gbit. The bulk memory is directly connected to the main system logic. It can be accessed directly by the embedded diagnostic processor and, via system logic, by the DSP. The SDRAM is connected to system FPGA by means of a 32 bit bus at 60 MHz with to Gbit/s bandwidth.

6.1.1.1.7 High speed backplane bus

The high speed backplane bus has been designed to allow the connection between the BCU board and all the control boards. The backplane bus is based on a strict master - slave architecture where the BCU board is the master and starts the transaction sending to the all control boards a request. Depending on the request all DSP and HVC boards react and can either receive the data sent by the BCU board (data write to the DSP board), or they can take the control of the bus if data have to be sent back to the BCU board.

The high speed backplane bus controller is performed by the main logic which redirects all the data from or to this bus to the high speed communication link interface.

The bus is physically based on BLVDS devices. This architecture guaranties signal integrity with high density and high speed busses.

The high speed backplane bus is a 32bit data bus at 60MHz in DDR mode with a typical throughput of 4Gbit per second.

6.1.1.1.8 Diagnostic backplane bus

The diagnostic backplane bus has been designed with a very similar architecture of the high speed backplane bus. It is controlled by the main logic which redirects all the data from or to this bus to the diagnostic communication link.

The diagnostic backplane bus is a 16bit data bus at 60MHz in DDR mode with a typical throughput of 2Gbit per second with a double speed with respect to the Ethernet diagnostic communication link. This choice has been taken to have a large margin between the internal bandwidth and the external one.

6.1.1.1.9 High speed communication links

The high speed communication allows to exchange data between different BCU boards or between BCU and external devices. In the NGWFC RTC, one high speed communication channel is used to transfer telemetry data from the BCU board to the TRS.

Every high speed communication channel is based on a full-duplex fiber optic link operating at 2.125 Gbit/s, according to the 2 Gbit/s FibreChannel physical layer standard.

Up to four high speed communication modules can be installed on the board. The number and the logic function of each channel can be configured according to the particular need.

The modules are based on the Agilent HFBR 5923 optical transceiver module (or equivalent). The module is compatible with industry standard LC-duplex fiber optic connectors, and is compatible with both 62.5-125µm and 50-125µm Multi Mode Fibers. The maximum specified link length is 150m with 62.5-125µm fiber and 300m with 50-125µm MMF.

6.1.1.1.10 Diagnostic communication link

The diagnostic communication is intended to be used for different purposes:

- transferring the diagnostic information (like acquired circular buffers and statistics);
- sending commands and acquiring system status;
- perform system maintenance.

The diagnostic communication link is based on a standard Ethernet interface, operating at 10/100/1000 Mbit/s. The physical interface is based on the standard copper link, with standard RJ 45 connector.

6.1.1.1.11 Expansion programmable input/output ports

The BCU/Communication boards comprehends a flexible expansion port (referenced as PIO – Programmable Input Output port) that can be used for interfacing different devices. The expansion ports comprehends 72 reconfigurable digital I/O lines, that can be configured according to different hardware interfacing standards (LVTTL, SSTL-2, SSTL-3, LVDS, …). The I/O lines are directly connected to the system logic (FPGA), and are available on the BCU front panel, on two 80 pins high density ribbon connectors (type 3M 810 series).

Within the MGAOS, the both PIO ports are used:

- PIO #0 implements the interface to the DM HVA. A simple interface board to convert the signal levels from LVTTL (PIO) to RS485 (DM HVA).
- PIO #1 implements the interface to the WFS. A simple interface board to convert the signal levels from RS644 (SciMeasure Little Joe AIA interface) to LVTTL (PIO).
- Two additional signals on PIO #0 are used to generate the synchronization signals for STRAP and IRIG board.

In both cases, the interfaces greatly benefit from the direct FPGA connection. In fact the FPGA handles directly the data transfers, thus offloading completely the DSP. The transfer between logic and DSP occurs by means of a true DMA process.

6.1.1.1.12 Serial links

Two standard serial communication links are available on the BCU board:

- one link is used to control and check the main logic functionalities and on-chip program flow during the system software development. This link is not used during normal operation of the system.
- the second link is available on front panel and can be used as general purpose serial interface. The serial port can be externally configured (by a configuration pin) according to RS232 or RS485 standard by means of a configuration pin. The maximum communication speed is up to 1 Mbit/s. There is no use foreseen for this port in the MGAOS.

6.1.1.1.13 ‘Slow’ fiber input/output

An additional fiber link is available on the BCU board. This port can be used for different purposes:

- asynchronous serial interface with speed up to 5 Mbit/s
- bi-directional transmission of logic signals, e.g. for synchronization purposes.

The fiber optic interface has ST standard connectors. The transmitter/receiver sections are based on Agilent HFBR 1412/2412 devices. One important feature is that the optical interface is ‘static’, i.e. there is no lower speed limit for the information or signals to be transmitted.

In the present design, there is no use foreseen for the ‘slow’ fiber interface.

6.1.1.1.14 Power supply

The BCU board requires a single 3.3V @ 3A (typical) supply. Internally, additional supply voltages are required for board operation, in particular 1.2V and 1.5V at relatively high currents for the DSP and FPGA cores respectively. The high accuracy, low impedance requirements on these power rails can be more easily satisfied if a distributed supply concept is adopted, where these low voltages are generated on-board by means of high efficiency, fast switching DC-DC converters.

The internal supply distribution scheme is resumed in Table 2.

Supply of:	Voltage	Current	Generated by:
DSP core	1.2V	1.2A	On-board switching DC-DC converter, derived from 3.3V
FPGA core	1.5V	0.8A	On-board switching DC-DC converter, derived from 3.3V
Logic, BLVDS	3.3V	0.85A	3.3V (VCC)
Gigabit Ethernet module	3.3V	1.2A	3.3V (VCC)
FibreChannel modules	3.3V	0.8A	3.3V (VCC)

Table 2 – BCU/Communication board supply distribution

6.1.1.1.15 Direct backplane bus signals

From the backplane bus there are some directs signals controlling some hardware configurations or time critical events. The following table summarize these signals:

Name	Direction	Description
bus_power_fault	in/out	typically configured as an input to verify if an external power fault condition happens. If a fault condition happens inside the board the signal is driven as an output by the main logic to notify the fault condition to the rest of the system
bus_sys_rst_n	input	performs a global reset of the board
bus_dsp_rst_n	output	generates a global reset to all DSP and HVC boards (can be SW generated internally by SW)
bus_fpga_clr_n	input	performs a soft reset of the board logic
bus_driver_enable	Input/output	Used in HVC board to force unconditional disabling of high voltage drives (emergency situation). The signal can be monitored by the BCU and forced to disable drives under software control.
bus_boot_select	input	selects the default or user main logic configuration and on-chip program code at the board booting
bus_slot_id[0 .. 3]	input	three static bits and one “three state” bit to identify the slot position from 0 to 31 where the control board is inserted

Table 3 – Direct backplane bus signals

6.1.1.1.16 High speed communication description

The high speed communication is used to transfer the real time data between different parts of the MGAOS. The typical communication paths are shown on Figure 4.

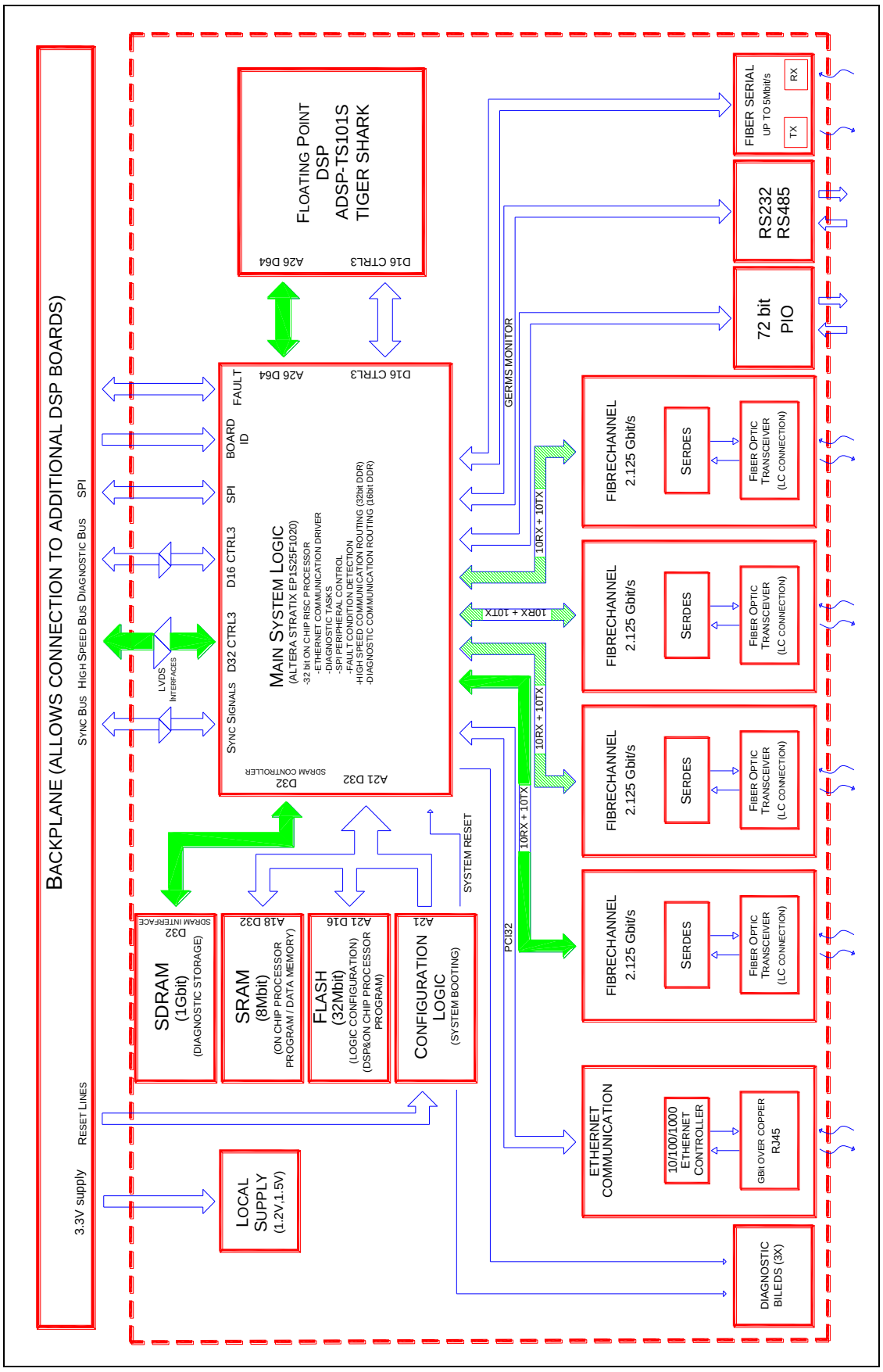


Figure 4 – High speed communication paths

Data can be sent or to received from the high speed communication inputs or exchanged among DSPs. In the system logic, data packets are processed and, according to the message destination, are dispatched to the final receiver. This might be the local DSP, other DSPs within the same crate, another receiver on a different BCU board (not in the current application) or an external device (in our application, the TRS HBA. In this implementation data are read from BCU SDRAM).

The communication from/to the high speed interface modules goes through independent busses (10 bit wide, separated reception and transmission).

When data are sent to or received from the local DSP, the synchronous communication mode on the 64 bit parallel bus of the DSP is used. Data are transferred through a DMA channel on the DSP, therefore the data flow does not interfere with the DSP processing activity. Latencies can occur only if another DMA process attempts to access simultaneously to the same memory location in the DSP internal memory, which is an extremely rare condition. It is important to notice that the DSP parallel bus is used only for real time communication (this is true also for the DSP and HVC control boards).

If the data are destined to other DSPs on the same crate, the logic transfers the data through the 32bit bus on the backplane. Similarly as above, also this bus is dedicated only to real time data transfer.

The communication speed on both the DSP parallel bus and on the 32bit backplane bus is of 4Gbit/s.

6.1.1.1.17 Diagnostic communication description

Diagnostic communication is destined to transfer commands and large amounts of diagnostic data with less stringent speed requirements. Diagnostic communication passes through the Ethernet port, which is connected by a dedicated PCI bus (32 bit) to the on-chip diagnostic processor in the system logic.

Diagnostic data might be transferred from/to the local DSP. To this aim, a dedicated interface, based on the DSP high speed synchronous communication ports (LinkPort) has been foreseen. The transfer rate is up to 2Gbit per second. This link does not interfere with the DSP bus activity, that supports the high speed communication, and can be also handled by DMA processes.

When diagnostic data need to be transferred to other DSPs on the same crate, a dedicated 16bit parallel bus is available on the backplane. This bus is similar to the high speed communication one (but 16bit wide), and has a throughput of 2Gbit per second.

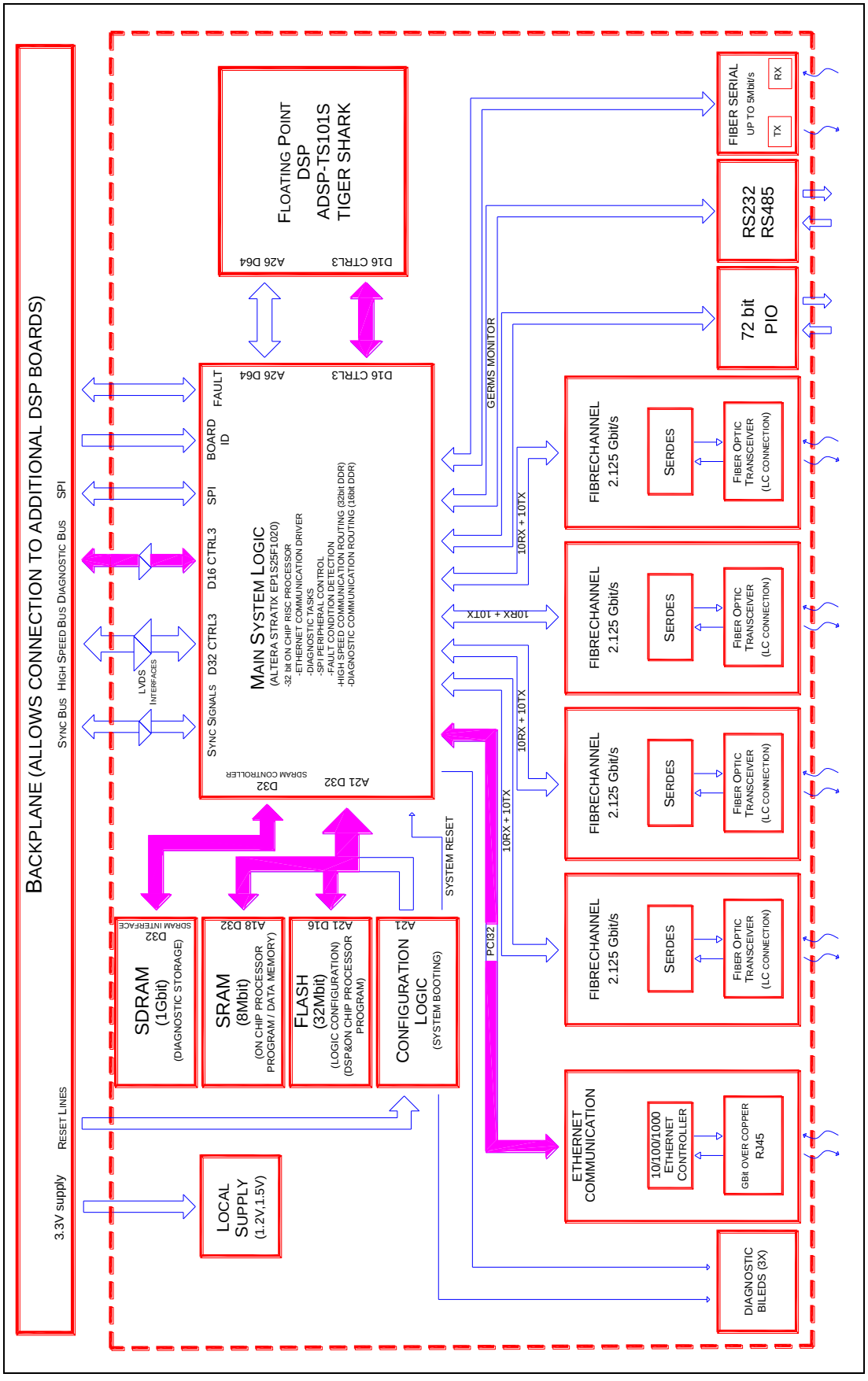


Figure 5 – Diagnostic communication paths

6.1.1.1.18 Boards layout

The BCU board PCB size is 196x91 mm. The limited available space forced to select components with very small SMD packages. Most of the components are mounted on the upper side of the board. The use of the bottom side is limited to decoupling capacitors and some analog components.

The components placing has been structured in functional zones. With reference to Figure 6, we can distinguish the following areas:

- Logic and computation: DSPs, FPGA, memories, configuration logic
- link communication front-end;
- Bus interface: B-LVDS drivers;
- Power supply: on-board switching and linear regulators;

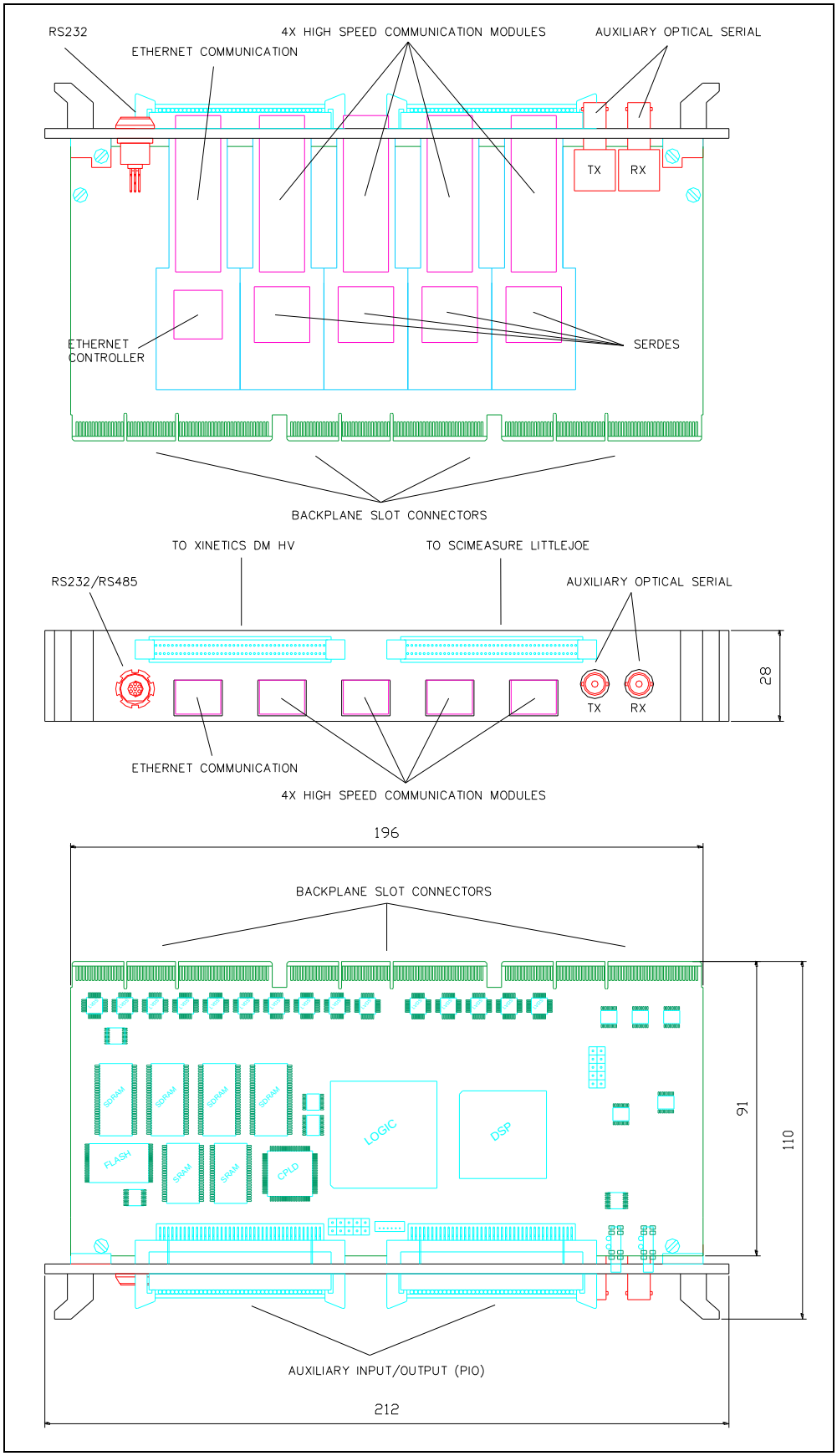


Figure 6 – BCU board components placing, with evidence of the different functional zones.

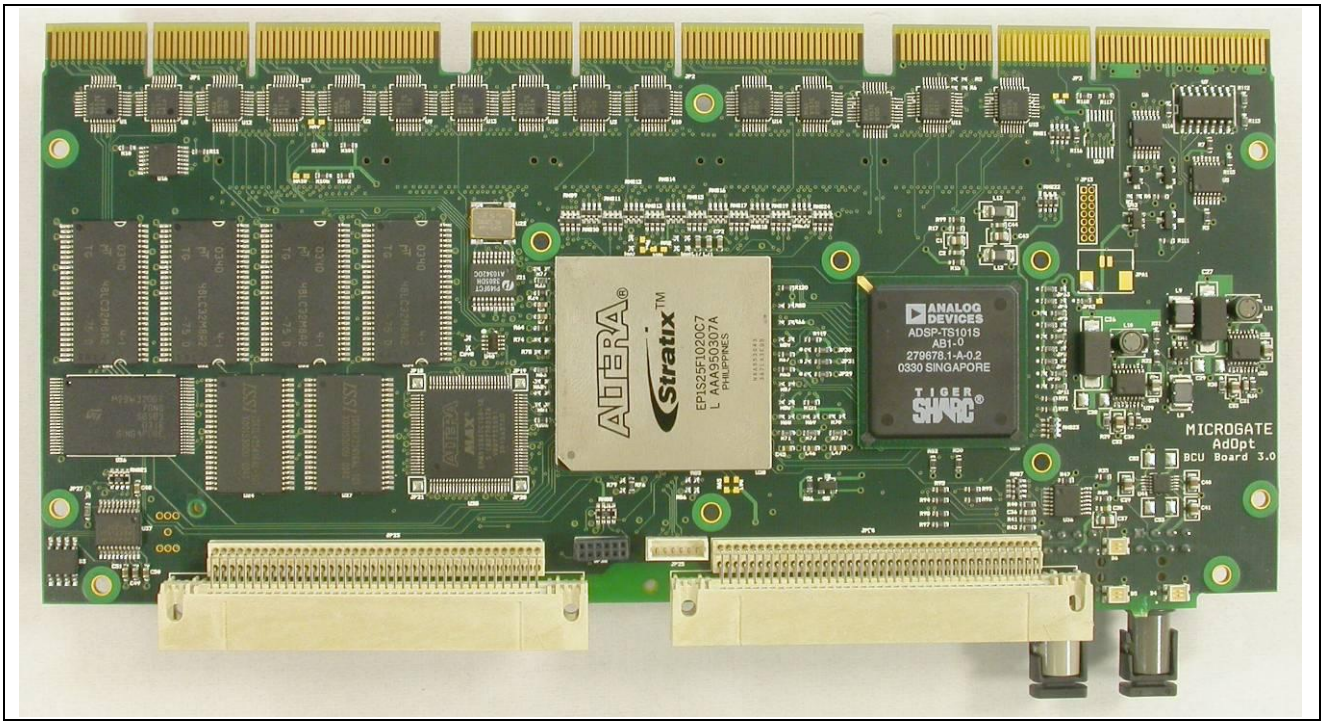


Figure 7 – BCU board.

6.1.1.2 AdOpt DSP BOARD

The *DSP board* is a very important component within the MGAOS. It performs the most heavy real time computations, namely residual wavefront computation and the control law servo computation. The total computational load is distributed in a parallel way among the 3 DSP boards foreseen within the system.

In other adaptive optics applications, the DSP board is directly interfaced to analog actuators and sensors. The analog section of the DSP board is not mounted on the boards dedicated to real-time computation only.

As an exception, the analog part is used for the DSP board that is part of the HVC board, as described in §6.1.1.3.

6.1.1.2.1 DSP board block scheme

The block scheme of the DSP boards is presented in Figure 8.

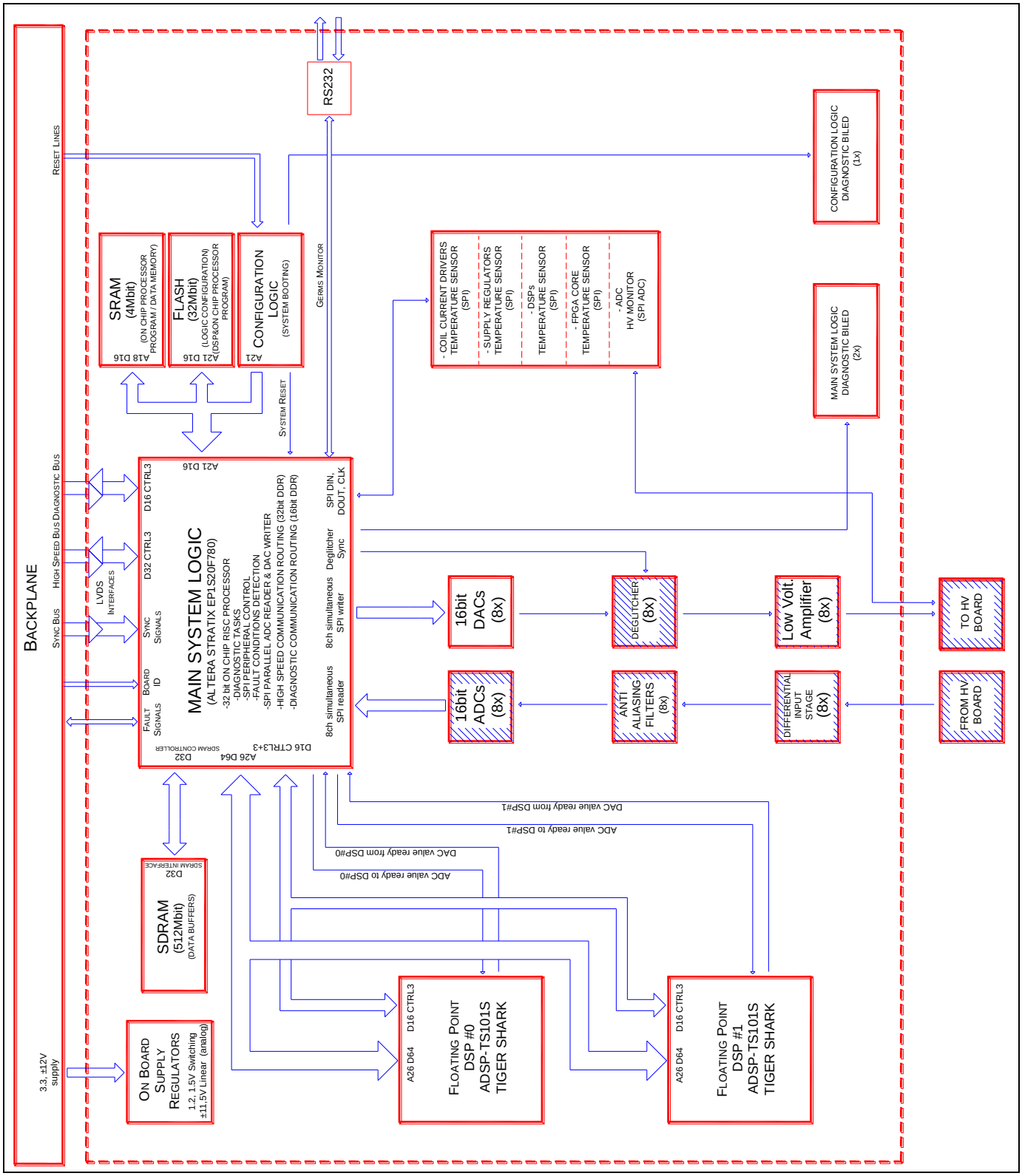


Figure 8 – DSP boards block scheme. The shaded parts refer only to the DSP board mounted on the HVC board.

As shown in Figure 8 the DSP board architecture is designed around the Main System Logic. All data transfers within the board pass through the main system logic which creates the correct interface between the devices.

6.1.1.2.2 Main System Logic

The Main System Logic is realized with an Altera Startix EP1S20F780 programmable logic. It performs the following main tasks:

- the interface between all the on board devices;
- the interface with the backplane buses;
- the on chip processor, which performs the acquisition of all local diagnostic devices, the first level fault analysis and, if necessary, takes the actions correlated;

6.1.1.2.3 Computational devices

The computational part is realized with two ADSP-TS101S DSPs, where all real-time processing is performed.

The DSPs communicate with the main logic by two shared buses and some direct lines:

- the high speed bus has 64 data bits at 60MHz of frequency with a typical throughput of 4Gbit per second; The bus is shared among the two DSPs and the main system logic.
- the diagnostic bus uses two 8 data bits parallel serial lines (called link ports) in DDR mode at 60MHz of frequency, with a typical throughput of 2Gbit per second;
- dedicated hardware flags, directly connected between DSPs and main logic, to provide a proper synchronization and detection of completion of the computational steps

6.1.1.2.4 Configuration Logic

The configuration logic has identical function as in the BCU board. Refer to §6.1.1.1.3.

6.1.1.2.5 Non volatile storage memory (FLASH)

The configuration logic has identical function as in the BCU board. Refer to §6.1.1.1.4.

In the HVC board, the FLASH memory contains also the calibration parameters (offset and gain) for each analog input/output channel.

6.1.1.2.6 SRAM memory

The 4Mbit SRAM memory is dedicated only to the main logic on-chip program and data memory. The organization of the memory depends to the on-chip program code.

The connection between the main logic and the SRAM is realized by a 16bit data bus at 60 MHz of frequency with a typical throughput of 1Gbit per second. This bus is shared with the FLASH device because during ordinary operation the FLASH is never used and then the bus is free to be used with the SRAM.

6.1.1.2.7 SDRAM memory

The 512Mbit SDRAM is used to storage real time data for diagnostic purposes. The main logic manages the bi-directional transfer of diagnostic data between SDRAM and DSPs' memory. Transfer from/to DSPs' memory is performed using a true DMA channel, therefore there is no software overhead by the DSP.

The data flow from the DSPs to the SDRAM is routed through the diagnostic channel, i.e. 16 bit 60MHz DDR LinkPort on DSP and 32 bit, 60MHz synchronous bus on SDRAM. The typical throughput is 2Gbit per second.

6.1.1.2.8 High speed backplane bus

The same considerations as in §6.1.1.1.7 apply to the high speed bus interface with the backplane. The DSP board is slave in the high speed communication. Internally, real time data are routed directly to/from both DSPs using the 64 bit DSP bus. Data transfers are handled by logic only and DSP memory is accessed in DMA, without requiring any software overhead. The high speed backplane bus is a 32bit data bus at 60MHz in DDR mode with a typical throughput of 4Gbit per second.

6.1.1.2.9 Diagnostic backplane bus

The same considerations as in §6.1.1.1.8 apply to the diagnostic bus interface with the backplane.

6.1.1.2.10 The diagnostic serial monitor

A standard RS232 serial link allows to control and check the main logic functionalities and on-chip program flow during software development. This link is not available to the user and it is not foreseen to use it for ordinary operation of the board.

6.1.1.2.11 Direct backplane bus signals

From the backplane bus there are some directs signals which controls some hardware configurations or time critical events. The following table summarize these signals:

Name	Direction	Description
bus_power_fault	in/out	typically configured as an input to verify if an external power fault condition happens. If a fault condition happens inside the board the signal is driven as an output by the main logic to notify the fault condition to the rest of the system
bus_sys_rst_n	input	performs a global reset of the board
bus_dsp_rst_n	input	performs a global reset of the board, similar to the bus_sys_rst_n from the control board point of view
bus_fpga_clr_n	input	performs a soft reset of the board
bus_boot_select	input	selects the default or user main logic configuration and on-chip program code at the board booting
bus_driver_enable	input	Used in HVC board to force unconditional disabling of high voltage drives (emergency situation)
bus_slot_id[0 .. 3]	input	Three static bits and one “three state” bit to identify the slot position from 0 to 31 where the control board is inserted

Table 4 – Direct backplane bus signals

6.1.1.2.12 Power supply

The DSP board requires a single 3.3V @ 1.5A typ. Other voltages are required for board operation, in particular 1.2V and 1.5V at relatively high currents for the DSP and FPGA cores respectively. The high accuracy, low impedance requirements on these power rails can be more easily satisfied if a distributed supply concept is adopted, where these low voltages are generated on-board by means of high efficiency, fast switching DC-DC converters.

The internal supply distribution scheme is resumed in Table 5.

Supply of:	Voltage	Current	Generated by:
DSP core	1.2V	1.2*2=2.4A	On-board switching DC-DC converter, derived from 3.3V
FPGA core	1.5V	0.6A	On-board switching DC-DC converter, derived from 3.3V
Logic, BLVDS	3.3V	0.65A	3.3V (VCC)

Table 5 – DSP board supply distribution

6.1.1.2.13 Mechanical configuration

The DSP board PCB size is 196x91 mm. The limited available space forced to select components with very small SMD packages. Most of the components are mounted on the upper side of the board. The use of the bottom side is limited to decoupling capacitors and some analog components.

The components placing has been structured in functional zones. With reference to Figure 9, we can distinguish the following areas:

- Logic and computation: DSPs, FPGA, memories, configuration logic
- Analog input: capacitive sensors input amplifier and ADCs (HVC board only)
- Analog output: DACs and low voltage amplifiers (HVC board only)
- Bus interface: B-LVDS drivers
- Power supply: on-board switching and linear regulators

The adopted scheme allowed to obtain a very efficient routing, in order to limit cross-talk problems.

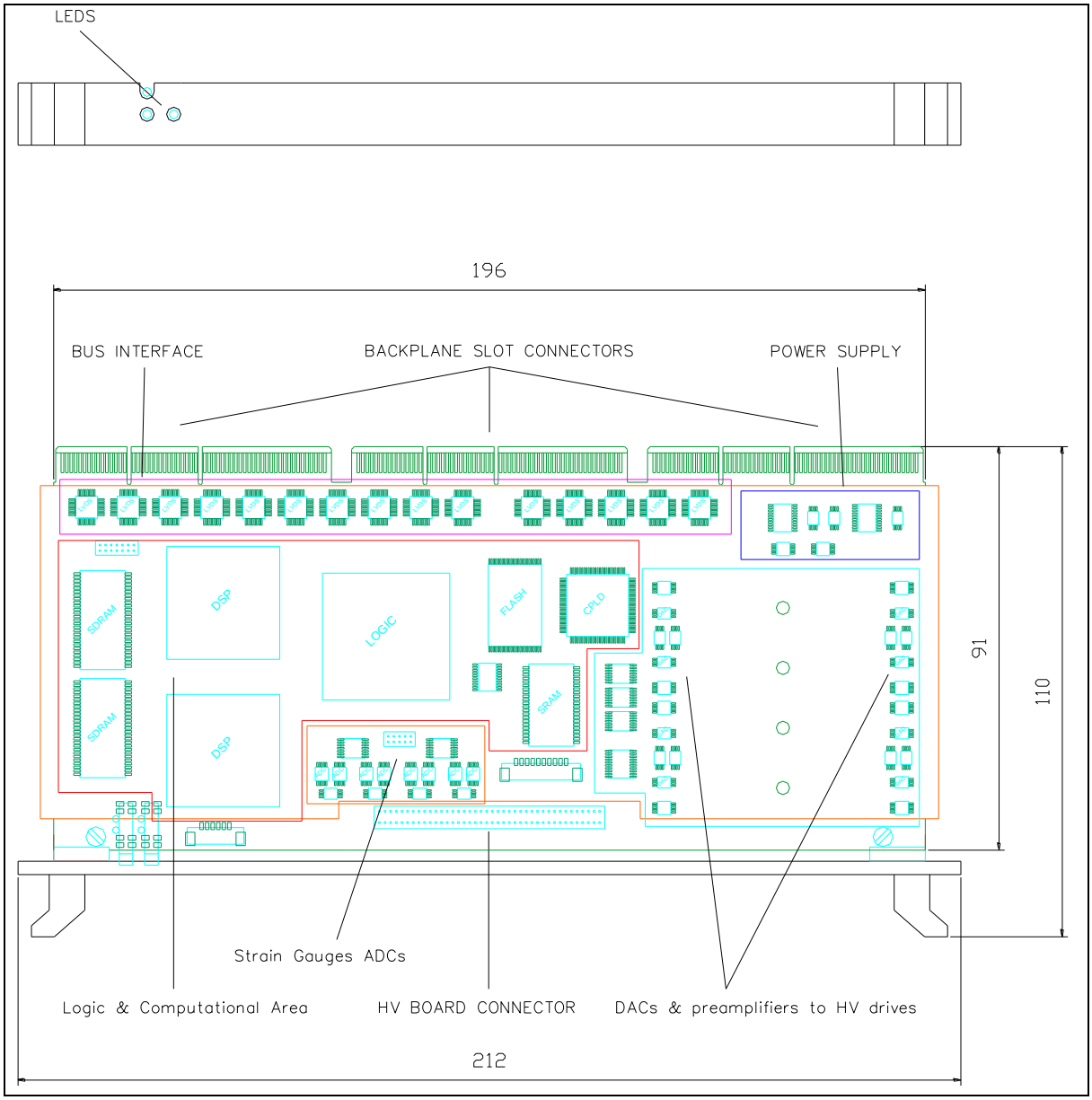


Figure 9 – DSP board components placing, with evidence of the different functional zones.

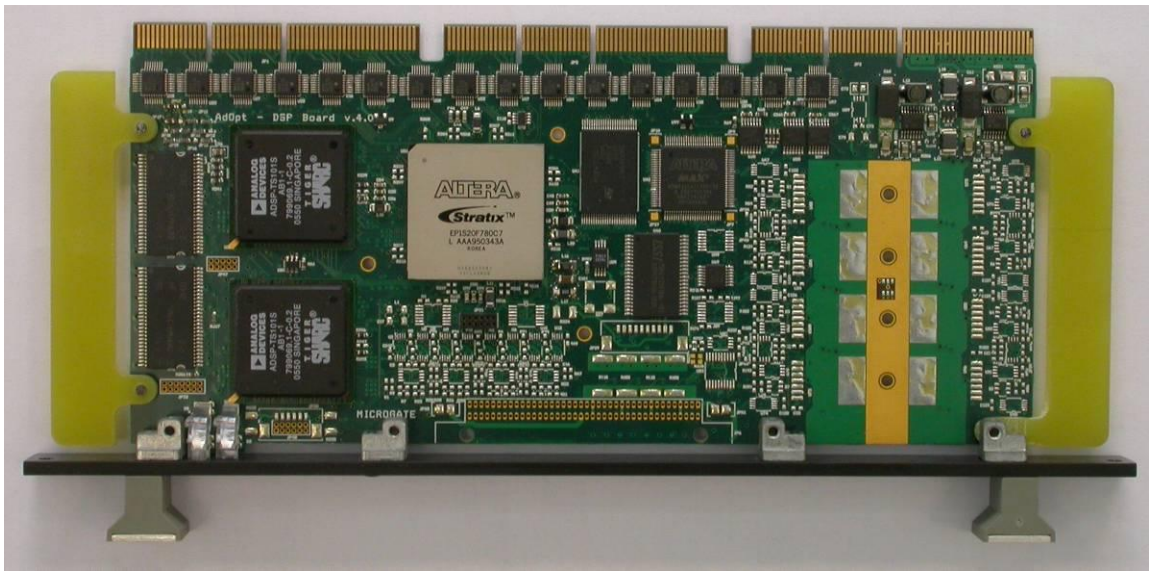


Figure 10 – DSP board V3.0. The board in this picture is completely mounted, including the analog parts used only on the HVC board.

6.1.1.3 AdOpt HVC board

The purpose of the HVC board is to drive and control in open and closed loop the Tip-Tilt mirror units (DTT and UTT).

Within the HVC board, we can distinguish two main subsystems:

- ***digital and low voltage analog control board***. This part is based on the AdOpt DSP board (see §6.1.1.2), with very little modifications, mainly change of some components value. This board has been specifically developed for controlling voice coil motor force actuators with a capacitive sensor position feedback. This design is very well consolidated and similar boards have been already manufactured in significant quantities.
- ***High voltage analog drives and strain gauges conditioning circuitry***, placed on a second board directly interfaced (as daughter-board) to the AdOpt DSP board.

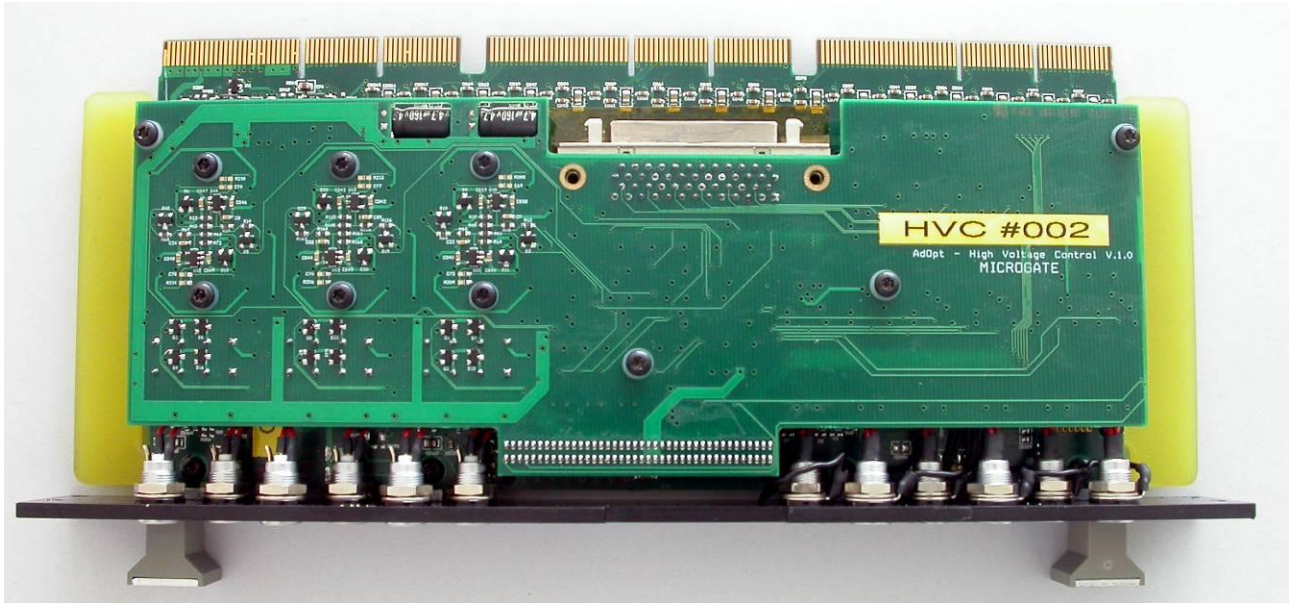


Figure 11 - HVC driver board

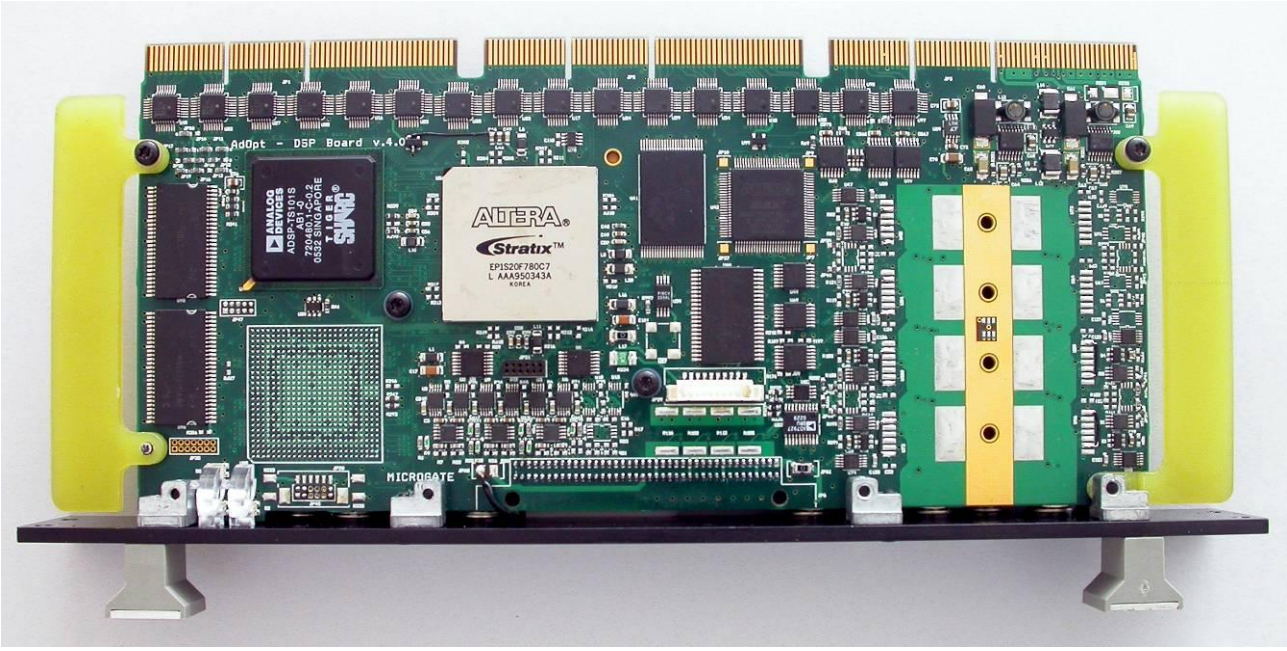


Figure 12 - HVC "mother" board

The AdOpt HVC board has 6 independent channels.

The DTT mirror requires three driving channels and three strain gage inputs, while the UTT mirror requires just two driving channels and two strain gage inputs. One third high voltage output is required to bias the 'hot' common point of the UTT mirror. In the proposed configuration, this mirror channel is driven by an additional high voltage driving channel. In this way, it is possible to control under software a 'smooth' power up and power down of the mirror (see §7.1.2.4).

6.1.1.3.1 DSP board analog input section

The input section of the AdOpt DSP board comprehends an input operational amplifier configured as quasi-differential stage and providing first order, anti-aliasing low pass filter. This stage drives a 16 bit ADC (type: AD7685). This ADC has been chosen for the following parameters:

- maximum conversion rate of 250 Ksamples/s
- very low power consumption, typ. 20mW
- 3.3V SPI serial interface
- very limited size (3x3mm)
- signal to noise ratio: 89dB

The ADC output is directly interfaced to the FPGA that generates the proper sampling signals and reads the eight ADCs simultaneously. Acquired data are directly transferred to the DSP memory without software overhead. At the end of the process, the DSP is notified using a dedicated hardware line. In this way, the readout time amounts to 800ns, while the total delay from 'start of sampling' to 'data available to the DSPs' amounts to 3µs.

6.1.1.3.1.1 DSP board analog output section

The analog reference for the piezoelectric actuators high voltage drives is generated by a 16 bit DAC type Maxim MAX5442.

This DAC has the following relevant characteristics:

- Settling time = 1µs to ½ LSB
- very low power consumption, typ. 60mW
- 3.3V SPI serial interface
- very limited size (3x3mm)
- thermal stability: 0.05 ppm/°C
- signal to noise ratio: 92dB
- digital feedthrough: 0.2nV/s
- seamless interface to generate bipolar output

The final drives reference output is buffered by a high speed, low noise, precision operational amplifier (AD8512) and connected to the high voltage drive (placed on the daughter-board, see §6.1.1.3.1.2).

With respect to the digital interface, the DAC input is directly interfaced to the FPGA that loads the new values on all eight DACs simultaneously. The data are automatically taken from the DSP internal memory. In this way, the update (excluding settling time) lasts just ~800ns.

6.1.1.3.1.2 High voltage drives daughter-board

The high voltage daughter-board comprehends the following sub-circuits:

- six high voltage drives
- six differential analog inputs acting as signal conditioner for strain gauge bridges
- strain gauges supply circuit
- high voltage ‘safety’ circuits
- local and external diagnostic circuits

The design of the high voltage drives has been based on the specifications reported on Table 6:

	DTT	**UTT**
Mirror type	Custom mirror based on 3x PI 841.60 piezoelectric actuators	PI S330.10 tit-tilt mirror
Capacitance (each actuator)	10.8 µF	3.6 µF
Dynamic Operating Current Coefficient	15 µA hz^{-1} μm^{-1} (actuator) 2.7 µA hz^{-1} μrad^{-1} (actuator)	0.22 µA hz^{-1} μrad^{-1}
Voltage range	0-100V	0-100V
Stroke (0-100V)	90 µm (actuator) 0.738 mrad (mirror)	± 1 mrad
Self resonant frequency	1 kHz	2.4 kHz
Dynamic	Actual mirror commands time-history provided by Keck, based on STRAP diagnostic.	

Table 6 – Tip-tilt mirror specifications

The high voltage drive is based on a linear amplifier. This configuration has been preferred to a switching typology to reduce the electromagnetic noise generated by the drive. This is particularly critical in closed loop operation, with the high sensitivity strain-gauge amplifiers placed very close to the high voltage amplifiers. Moreover, as shown in Table 10, the dissipation is very reasonable even using a linear device.

The main component of the high voltage stage, namely the high voltage operational has been selected considering various aspects, in particular:

- Voltage range
- Output current
- Dynamic response
- Quiescent current
- Package

In order to get an estimate of the **voltage and current requirements**, we have simulated the theoretical voltages and currents required to drive the DTT mirror to compensate a real turbulence pattern (generated by real acquisition at Keck), acquired in 'very bad' seeing conditions. The DTT mirror has been chosen to define the design boundaries, because of the higher actuators capacitance.

The gains used in the simulation are the actual ones currently set at the telescope.

The results are presented in Figure 13 and Table 7.

As it can be noticed, the output voltages obtained from the simulation do clearly exceed the voltage range of the drive currently in use (0-100V). To overcome this limitation, the voltage range for the new design has been specified to be significantly higher than now.

The minimum requirements for the drives have been derived according to the following criteria:

- Voltage: imposed by maximum specification for the actuators $\Rightarrow$ **-20/+120V** for both DTT and UTT mirrors
- Current: greater than maximum between 6σ and min-max $\Rightarrow$ **> 39.8mA**

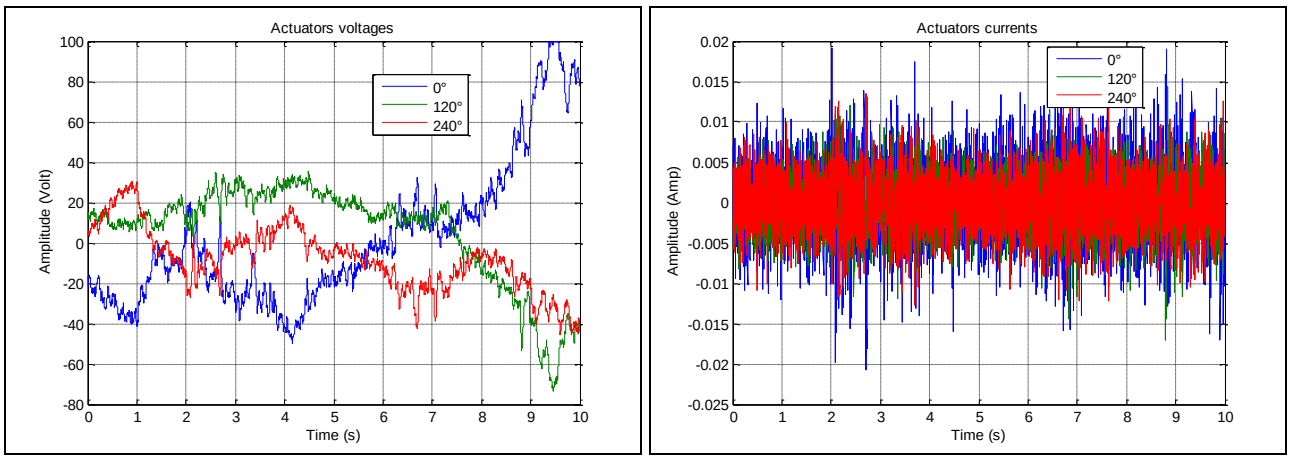


Figure 13 – DTT mirror actuators driving voltages and currents for typical 'bad seeing' compensation.

Parameter	***Act.1 (0°)***	***Act. 2 (60°)***	***Act. 3 (120°)***
Voltage min-max (V)	149.7	108.8	75.9
Voltage RMS (V)	35.6	24.3	15.7
Max slew rate (V/ms)	1.91	1.58	1.25
Current min-max (mA)	39.8	31.5	26.2
Current RMS (mA)	4.45	3.21	3.28

Table 7 – DTT mirror actuators driving voltages, slew rate and currents. RMS and min-max values for typical 'bad seeing' compensation.

With respect to the **dynamic requirements**, we can distinguish between full-power and small signal dynamics.

Full power requirements are related to possible drive saturation due to excessive current and/or slew-rate limitations. The maximum **slew rate** for the studied turbulence simulation is of **1.91 V/ms**.

The **small signal** (i.e. below any saturation) dynamic response requirement can be evaluated considering the closed loop bandwidth specification of 200Hz (see req. 248 and 282 in [AD6]).

Based on the considerations above, a suitable high voltage operational amplifier is the Apex type PA243DF.

The most relevant characteristics of these device are reported on Table 8:

Maximum supply voltage	350 V
Peak output current	120 mA
Continuous output current	60 mA
Quiescent supply current	2.2 mA
Output voltage swing	$\lvert\pm V_{supply}\rvert$ - 10
Slew rate	30 V/µs
Gain Bandwidth Product	3 MHz
Noise, 0-10kHz	50 µV RMS
Package (dual channel amplifier)	24 pin PSOP (14.5 x 16mm footprint)

Table 8 – Apex PA243 amplifier main characteristics

The static and dynamic performances are well suited for the application, with relevant margins. Low quiescent current and small package are also important factor, allowing us to design a very compact multi-channel control board.

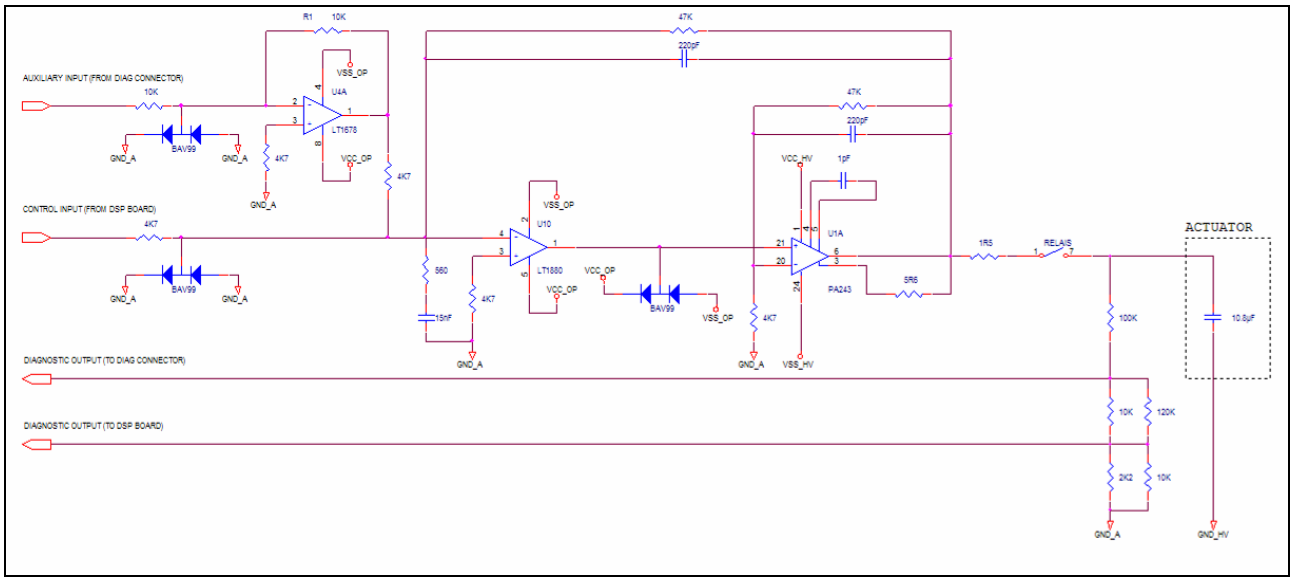


Figure 14 – High voltage drive output stage

The high voltage drive output stage is configured as composite amplifier (see Figure 14). The high voltage op-amp is driven by a precision, low noise, high speed operational amplifier (AD8512). The stabilization of the drive is obtained by means of load isolation through output resistance, dominant pole compensation on feedback network and noise gain compensation on the inverting input. All these parameters have been optimized for the DTT mirror, because its actuators are more critical to drive (10.8µF with respect to 3.6µF on of UTT mirror actuators). The gain of the composite output stage has been set to 12, in order to match both the output range of the DAC driver and the input characteristics of the existing design (a summing input allows to inject analog signals to the high voltage drive for testing purposes, see Figure 14).

The analog performances obtained by design are reported in §6.1.1.3.4.

The *strain gauge input stage* comprehends a precision, low noise differential amplifier (AD8221). The gain of this stage is 50. This first stage is connected to the input amplifier on the DSP board having a gain of 20, therefore the total gain is 1000. It comprehends also the anti-aliasing low-pass filter (dual pole at 26.5 kHz, matched with the control loop rate of 70 kHz), and common mode/differential mode filters on the strain gage inputs, to minimize the effects of noise pick-up on the transmission line.

For safety reasons, all high voltage outputs foresee a relay that allows disconnecting physically the output from the actuator. All these devices are normally open and the high voltage circuits are automatically disconnected in case of system malfunctioning (e.g. intervention of control board watchdog).

The high voltage outputs are connected to a diagnostic ADC (8 channel, 12 bit resolution) on the DSP board. These acquisition channels are not used for real-time control, they are just provided for diagnostic purposes.

The two additional channels monitor continuously the high voltage supply rails.

Additionally, analog monitor and test points are provided on all high voltage channels. These include:

- High voltage drive output voltage divided by 10 (-2 ~ 12V).
- Input signals to a summing junction of the high voltage amplifier. A gain of 10 is applied to the input signal (input range: -2 ~ 12V)
- Strain gauge signals after differential initial amplification (gain 50mV/µm).

6.1.1.3.2 Power dissipation

The HVC board requires five power supplies: 3.3V for logic, ±12V for analog circuitry and –35,+135V for high voltage drives. Other voltages are required for the operation of the DSP board, in particular 1.2V and 1.5V at relatively high currents for the DSP and FPGA cores respectively. As for the standard DSP board, these additional power rails are generated on-board. It shall be noticed that just one DSP is present on the DSP board of HVC.

The high voltage current has been estimated as follows:

- Quiescent current of HV amplifiers: 2.2mA x 6ch = 13.2mA
- Typical RMS current (bad seeing condition): 3.6mA x 6ch = 21.6mA
- Total: 34.8mA

The internal supply distribution scheme is resumed in Table 5.

Supply of:	**Voltage**	**Current**	**Generated by:**
DSP core	1.2V	1.2A	On-board switching DC-DC converter, derived from 3.3V
FPGA core	1.5V	0.6A	On-board switching DC-DC converter, derived from 3.3V
Logic, BLVDS	3.3V	0.65A	3.3V (VCC)
Analog	±11.7V	0.02A	On-board linear regulators, from 12V
High Voltage drives	-35 +135V	0.0348A	Dedicated power supply for high voltage drives.

Table 9 – HVC board supply distribution

6.1.1.3.3 Mechanical configuration

The HVC board is configured as a stack-up of two PCBs. An AdOpt DSP board, slightly modified for the scope, is connected to a second PCB where the high voltage amplifiers and the strain gauge signal conditioning amplifiers are mounted.

The mechanical layout is shown in Figure 15. The board occupies two slots of the MGAOS crate (28mm).

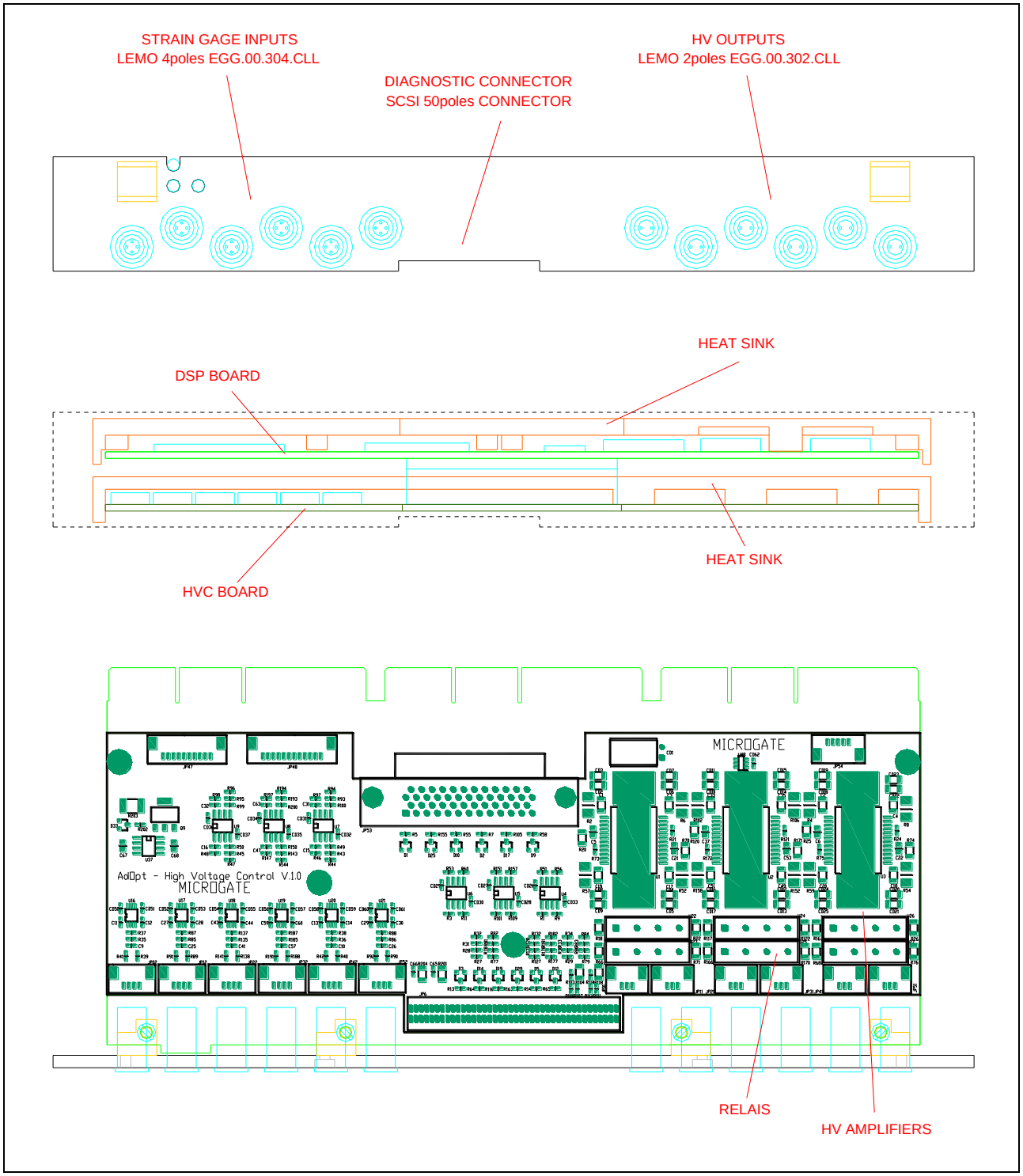


Figure 15 – HVC board mechanical layout

6.1.1.3.4 Performance estimate through numerical simulation

The performance of the HVC board has been estimated using numerical simulation.

Hereafter we report the simulation procedure and related results:

- *Simulation of single actuator high voltage drive electronics.* The electronic design has been preliminarily optimized and simulated using PSPICE analog simulator. The actuator has been

represented as a capacitance. The simulation results are reported on Figure 16 and Figure 17. The obtained bandwidth is of 13 kHz.

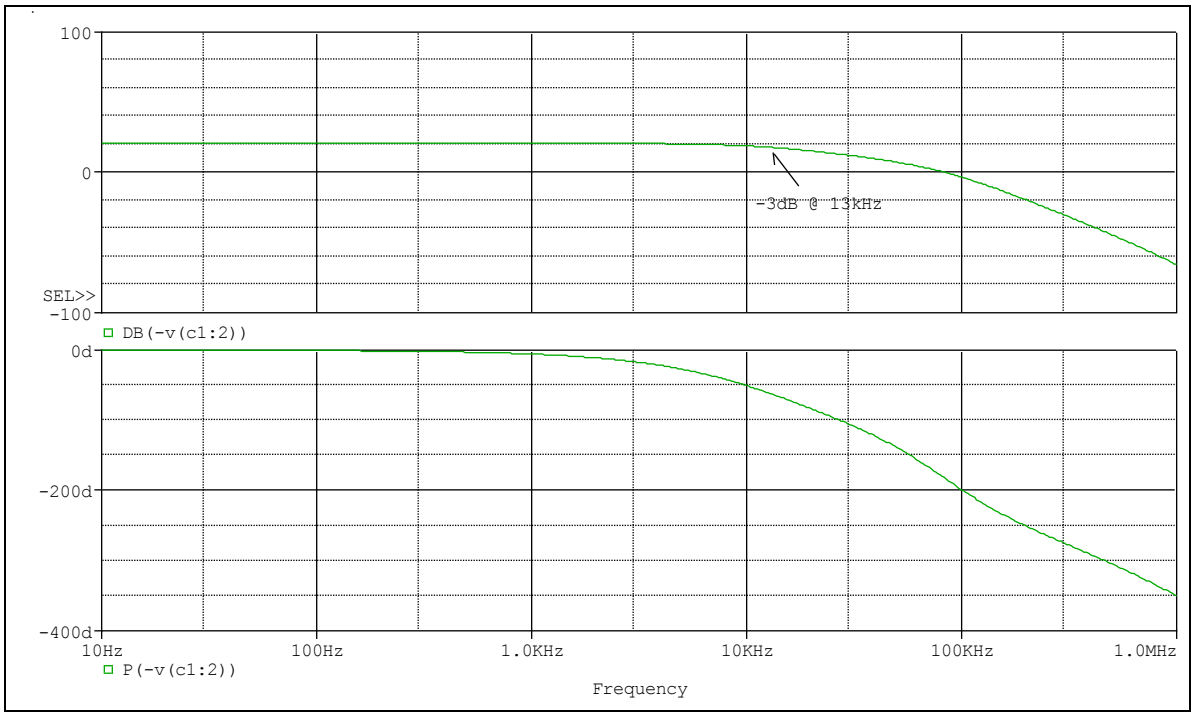


Figure 16 – Bode plot of actuator drive electronics. DTT mirror actuator (10.8µF).

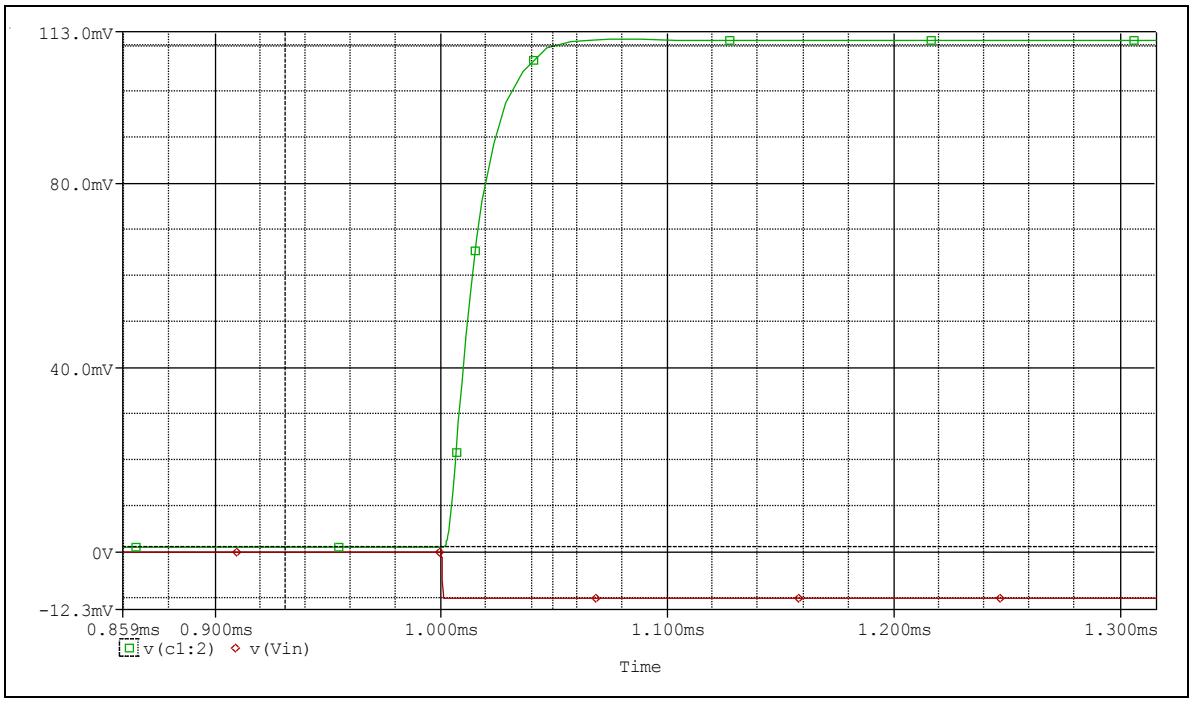


Figure 17 – Step response of actuator drive electronics. DTT mirror actuator (10.8µF).

- *Drive and actuator transfer function.* From the PSPICE simulation, we have derived by a least square algorithm the corresponding continuous transfer function, taking into account the drive dynamics and the capacitive load of the actuator. The transfer function refers to the electrical response only, not including the mechanical system response. The system is well fit by a dominant single pole at 12.7kHz and a complex poles pair at 72kHz.
- *Determination of mirror transfer function.* The transfer function from actuator voltage input to angular displacement has been obtained from the plots reported on page 2-12 of [RD4]. The actual data acquired during the experimental measurement of the transfer function were not available, therefore the transfer function has been manually inserted considering two eigenfrequencies, the first at 700Hz with a damping factor of 0.6, the second at 1kHz with a damping factor of 0.09. For a preliminary evaluation, this 'manual' way of matching the transfer function has proved to be more accurate than a least square method based on the actual transfer function, probably due to the limited number of input points derived from the mirror bode plot.

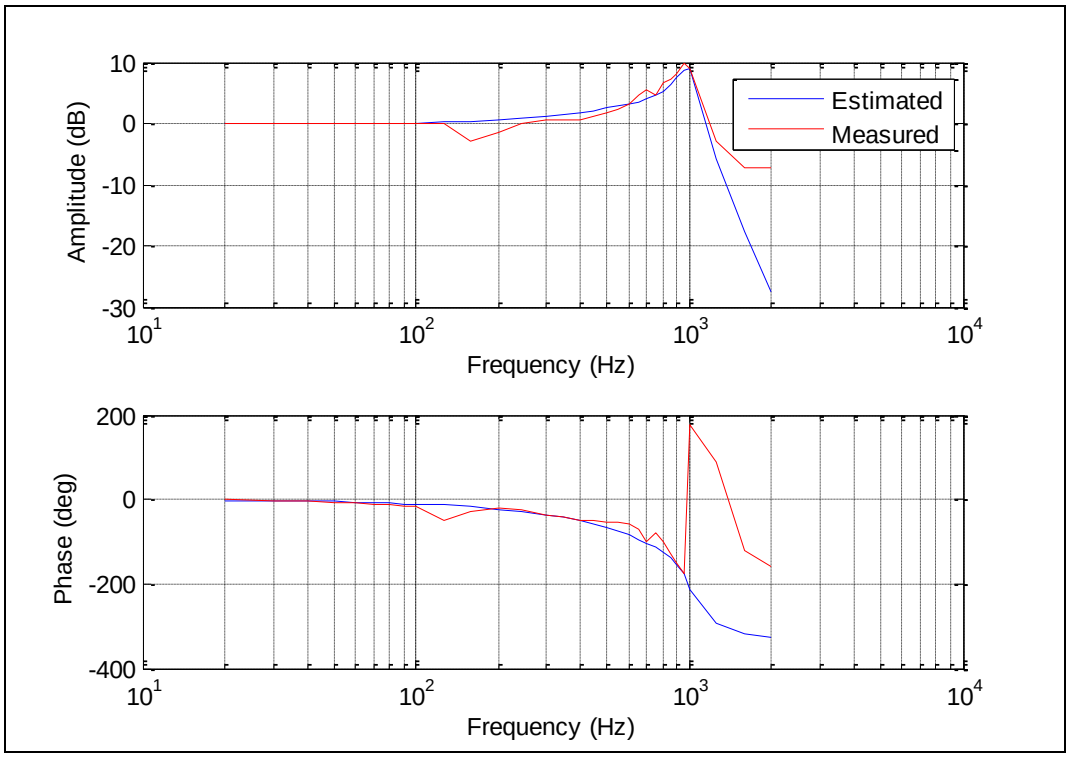


Figure 18 – Estimated vs. actual mirror transfer function.

- *Optimization of dynamic controller.* The open and closed loop transfer functions of the controlled systems have been computed. As compensator, we have initially used a simple PID controller. The control coefficients have been preliminarily optimized by iterative functional minimization. This procedure simulates the procedure we intend to use for automatic coefficient optimization on the real system. The minimization functional is the Integral Square Time^2 Error, applied to a step response. We obtained a **bandwidth of 550Hz** (see Figure 19), with a good phase margin of 65 deg and a quite poor gain margin of 11dB (see Figure 20).

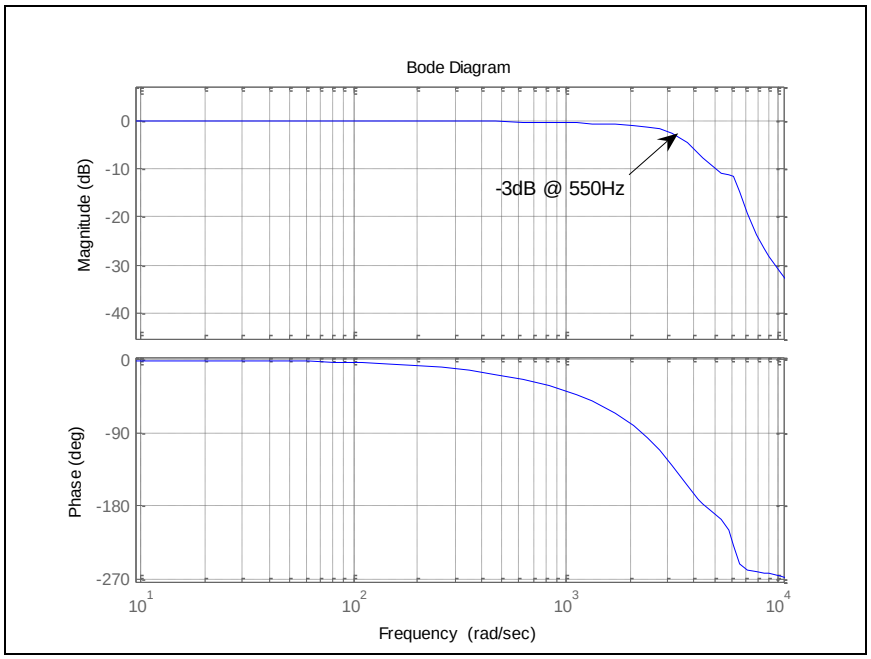


Figure 19 – DTT control. Closed loop transfer function with gain and phase margins.

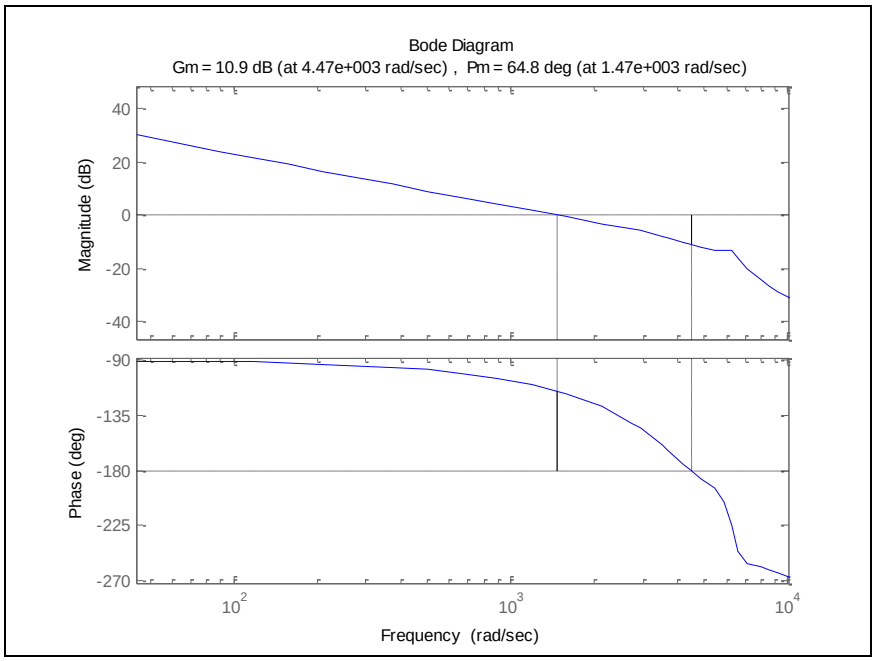


Figure 20 – DTT control. Error transfer function with gain and phase margins.

- *Simulation of system considering non-linear effects.* The closed loop behavior of the system has been simulated using Simulink. At this point, some other non-linear elements have been added to the simulation scheme:

- Digital quantization. It has been assumed that the full stroke motion and full stroke output signals are both divided in 2^{16} steps.
- Computational delay. The computational delay, including ADC sampling, conversion, reading and DAC writing and settling is 7.4µs. This value is extrapolated from actual data measured on similar systems.
- Noise. The noise term generated by the strain gage amplifier is relevant, mainly due to the poor sensitivity of the strain gage installed on the actuator (25 µV/µm with 5V bridge supply, data measured on P840.61 DTT actuator). This forces to amplify the strain gage signal by a factor of 1000, therefore the voltage noise of the first amplifier becomes relevant. Neglecting the effect of external noise sources, like pick-up along the connection to the strain gage, and taking into account the noise sources on amplifiers and resistors (flicker, burst, avalanche), the integrated noise up to 70kHz (CLMP rate), expressed as corresponding on-sky angle, amounts to 13.1 mas RMS. Limiting the noise integration to the specified closed loop bandwidth of 200Hz, the noise is 1.23 mas RMS on-sky.
  Another contribution comes from the high voltage drive noise. At simulation level, this contribution is totally negligible.

The simulation block diagram is presented in Figure 21. Figure 22 reports the results of 1 second of 'bad seeing' activity of the actuator at 0 deg (input data are the same of Figure 13). The 'following error' RMS is **2.8 mas**.

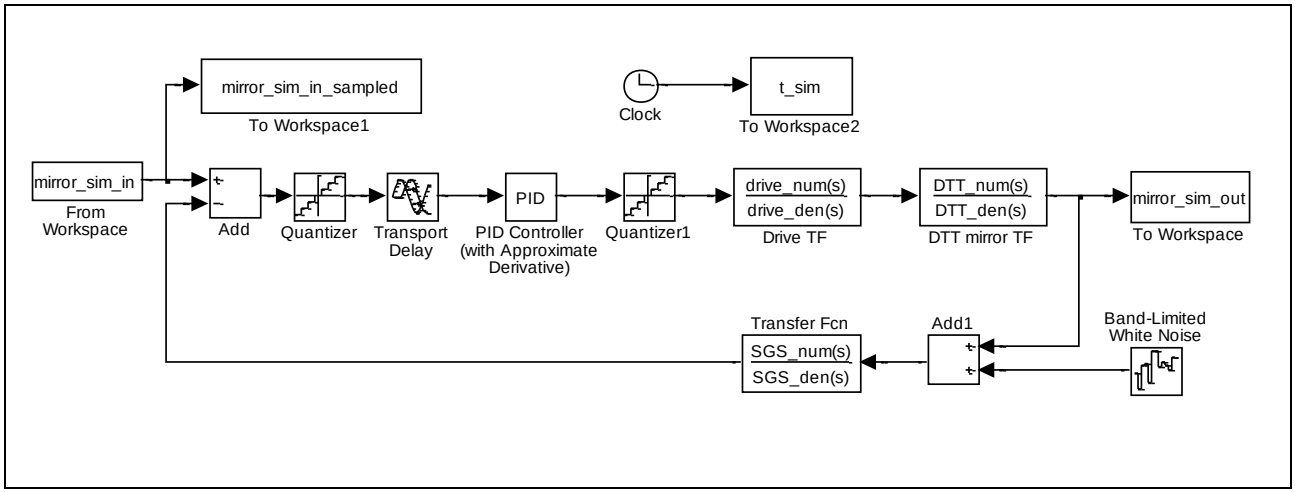


Figure 21 – Simulation block diagram.

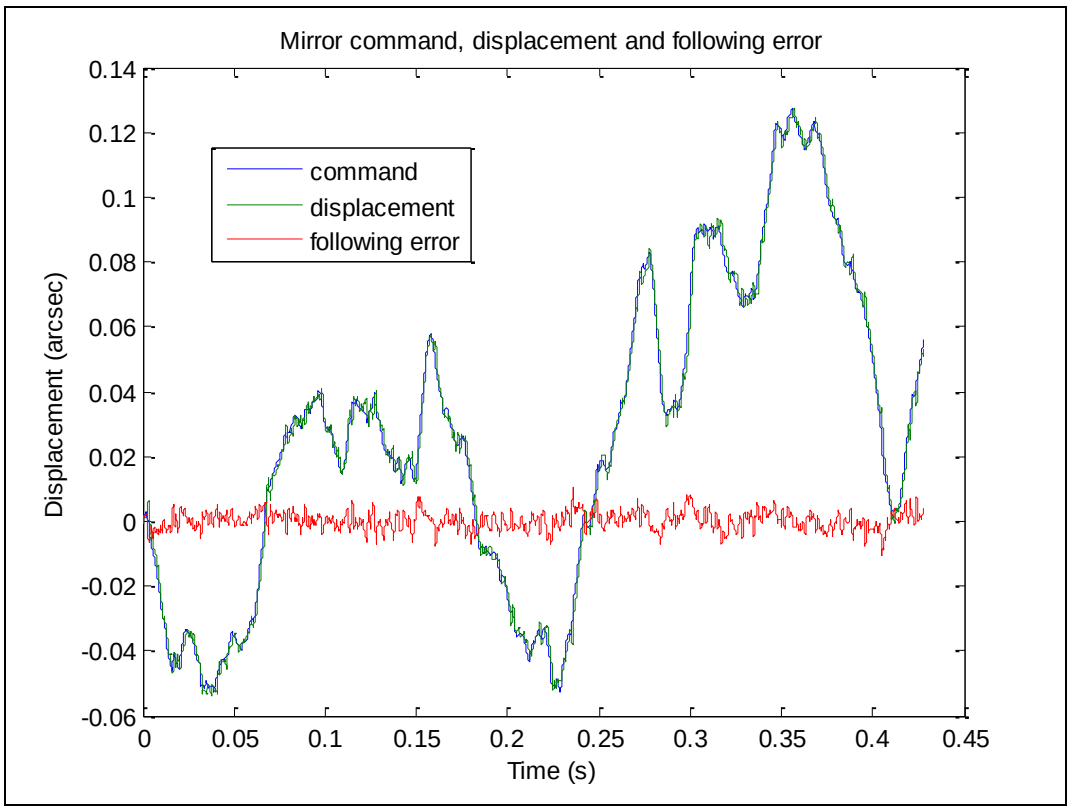


Figure 22 – Simulation of a 'bad seeing' input to actuator at 0 deg.

- *Pspice simulation of 'actual' actuator commands to check for saturation.* The simulation data, taken at drive input, has been used as stimulus for the electronic simulation, to verify that no saturation due to current or voltage limit occurs.

Comments to simulation results.

- The simulation considers the activity of a single actuator, as if the three actuators would be ideally decoupled. This is probably an acceptable simplification, considering the intrinsic high rigidity of the actuators. Their self-resonance is at 6 kHz, while the mirror main resonance is at 1 kHz. This is a significant separation.
- The simulation does not take into account the electrical crosstalk between actuator and strain gauge sensors.

6.1.1.3.5 Experimental tests

In order to validate the above simulation tests and to prove the validity of the described design concept, we have conducted a test campaign on a bread-boarded prototype of the actuator drive.

It shall be noticed that the tests are now superseded by the tests performed on the final unit with the final DTT and UTT mirrors (see [AD10]). Nevertheless the initial tests are reported here to document the development process.

6.1.1.3.5.1 Test setup

The test setup comprehends the following main components:

- Breadboard of a single channel high voltage drive amplifier. The prototype implements the above described circuit, apart from the 'safety' components (output relays, protection diodes) and monitor

features. The high voltage drive supply is set to –35/+135V, as in the final configuration, i.e. we tested also the 'overdrive' capability of the actuator.

- Breadboard of first stage of the strain gage signal conditioner (differential amplifiers).
- Microgate AdOpt DSP board, properly configured to operate as digital part (real-time control) and low voltage signal processor of the HVC board.
- PI840.61 DTT actuator installed on a dedicated mechanical fixture where the pre-loaded actuator drives a mass tailored to reduce the self-resonance frequency down to 1kHz, in order to roughly emulate the dynamic behavior of the DTT mirror.

To maximize the efficiency and reduce the effort to complete the test setup, we used the AdOpt components installed on the internally developed testbench usually dedicated to automated testing of AdOpt systems.

In terms of software, the CLMP has been implemented in an almost final configuration. For the tests we have made extensive use of the '*dynamic buffers*', natively available on the AdOpt DSP boards, allowing us to upload the desired real-time command sequences and to read back the position sampled at full control speed (70kHz).

All tests have been sequenced by means of Matlab scripts, using the MGP communication protocol.

Here following the list of tests and measurements performed:

- Measurement of actuator histeresys (clearly in open loop) over different voltage ranges.
- Open and closed loop transfer function.
- Control loop optimization using automated script.
- Open loop and closed loop step response.
- Response to the same time-history used in simulation (see Figure 22), both in open and closed loop.

6.1.1.3.5.2 Test description and results

Actuator histeresys

It is well known that piezoelectric actuators exhibit a significant histeresys. We have tested the actuator to verify the entity of such behavior at different voltage spans (30/60V, 10/90V, -20/+115V) and different speeds ($5Vs^{-1}$, $18Vs^{-1}$ and 85V s^{-1}). By evidence the test has been conducted operating the actuator in open loop.

Some of the results are presented in Figure 23 to Figure 26. The histeresys amplitude is remarkable, up to 17µm for the same applied voltage on the –20/+115V test. We noticed a very weak dependence on the slew rate, namely the effect tends to increase reducing the speed. This is also confirmed by the large deviation of the first cycle if the actuator is left on the initial position for long time, as in Figure 26, where the test has been executed after driving the actuator at -20V for 5 minutes.

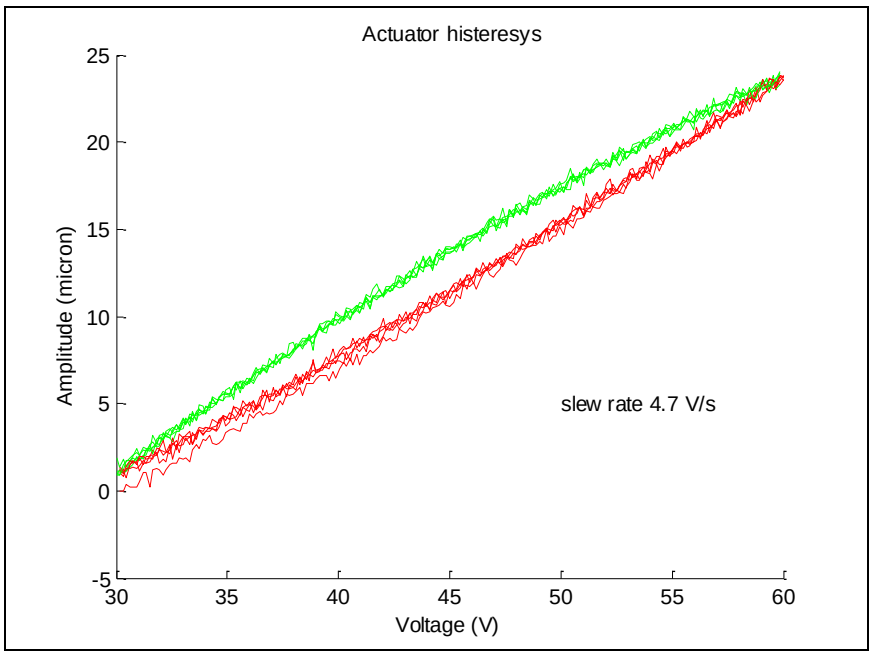


Figure 23 – Actuator histeresys. Voltage span: 30/60V, slew rate 4.7Vs^{-1}

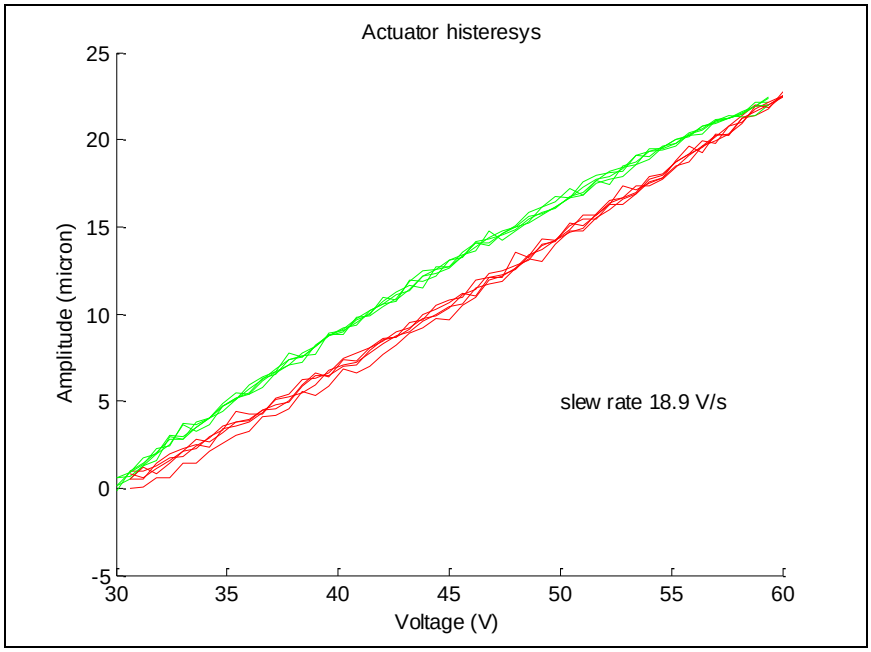


Figure 24 – Actuator histeresys. Voltage span: 30/60V, slew rate 18.9Vs^{-1}

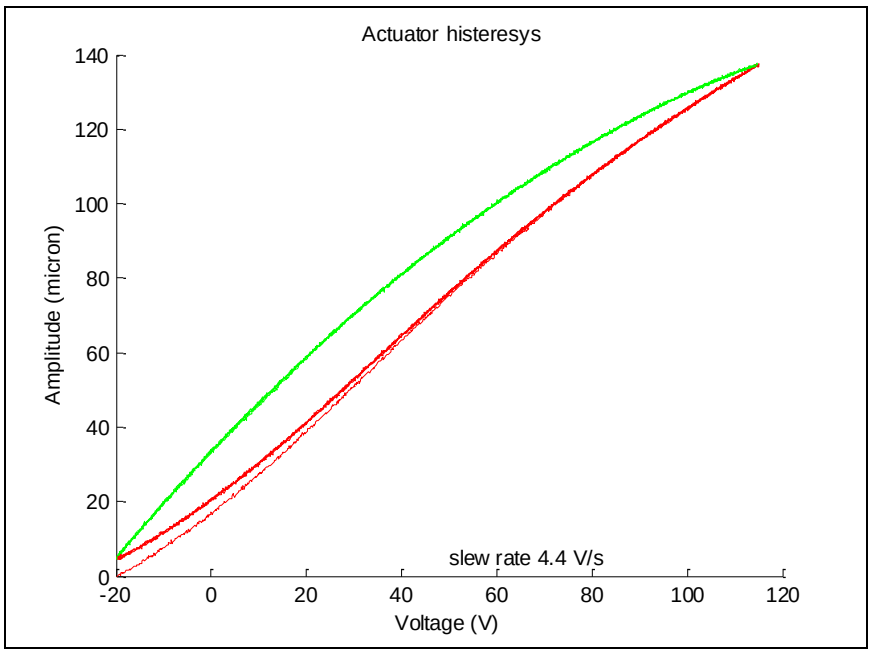


Figure 25 – Actuator histeresys. Voltage span: -20/+115V, slew rate 4.4Vs^{-1}

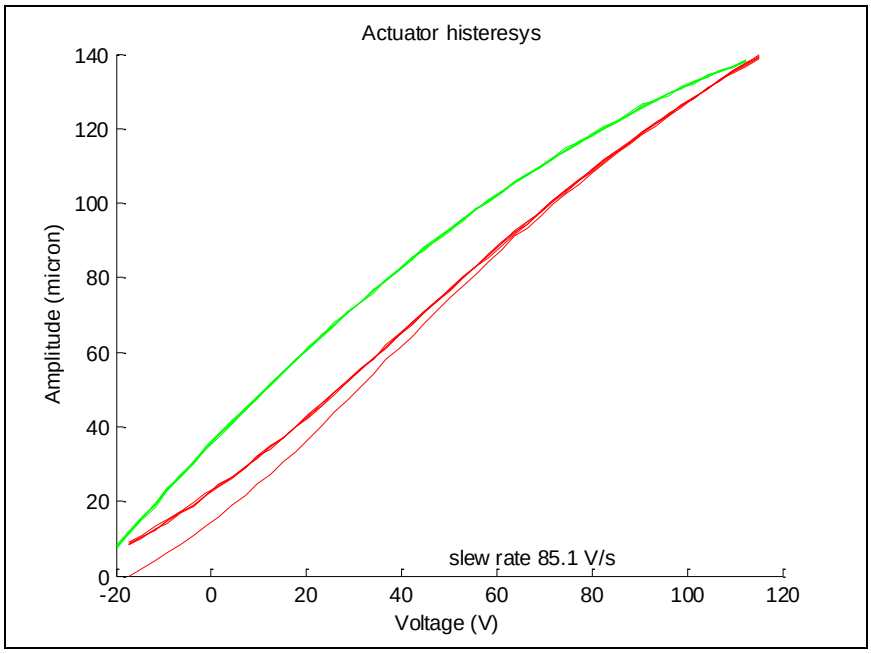


Figure 26 – Actuator histeresys. Voltage span: -20/+115V, slew rate 85.1Vs^{-1}

Step response

We report here the results to a step command with 2µm amplitude. The position output is filtered numerically with a 5th order butterworth filter at 4 kHz. The rationale behind this post-processing is in the fact the position acquired at 70kHz is not the *actual* actuator position; it is affected by the noise of the strain gage and related amplifier. Filtering at 4kHz is still representative of the actuator dynamics, being the mechanical resonance at 1kHz, but reduces the artifacts introduced by the acquisition chain.

The open loop response puts to evidence once more the large static error due to histeresys. The error is particularly evident on a small amplitude step, while on larger movements the actuator tends to behave more linearly, thus reducing the relative error.

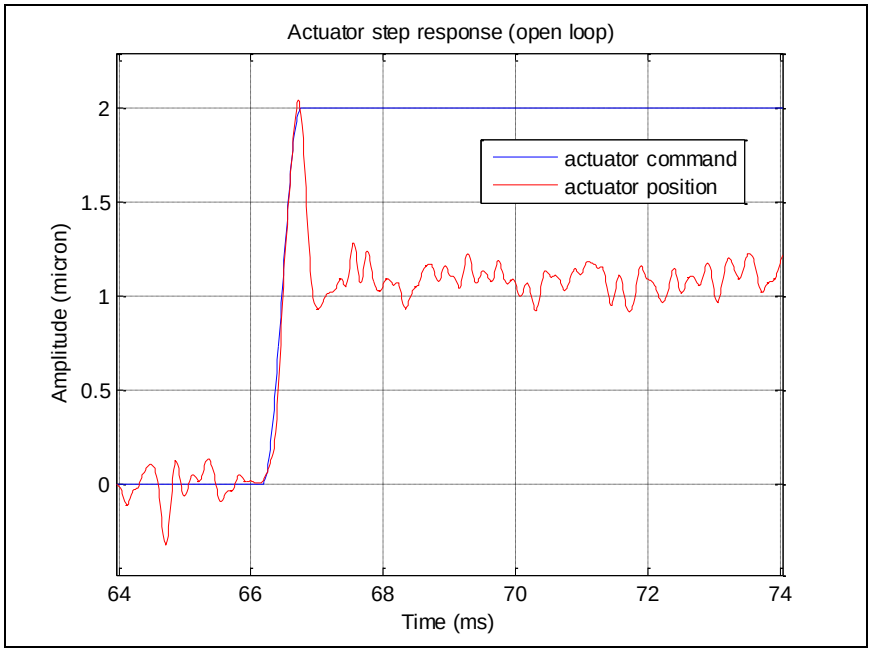


Figure 27 – Open loop step response, 2µm commanded step.

The closed loop response shows that the static error goes to zero thanks to the integral part of the dynamic controller. The effect of the histeresys is evident from the large difference between actuator command and actuator position.

Extrapolating the bandwidth from the step response, we measure a characteristic time of ~0.7ms (an accurate measurement is not possible due to the crosstalk disturbance), which corresponds to ~230Hz bandwidth. Even in this non-optimal condition, it is within specification.

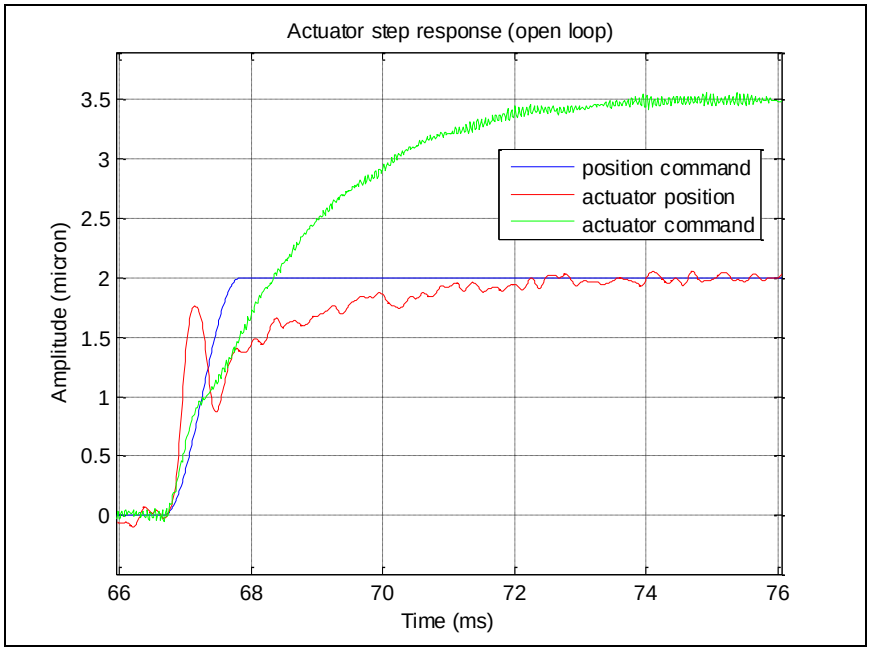


Figure 28 – Closed loop step response, 2µm commanded step.

Time-history response.

The time-history test gives the most realistic perception of the drive behavior in real environment. To this aim, we used the same time-history used for simulation (see §6.1.1.3.4 and Figure 22). It comes from real data acquired during observation in 'bad' seeing conditions.

The test has been performed over 0.43s of observing time. The time history is input at 1kHz, using Zero-Order Hold (ZOH) approximation to oversample it to the control loop rate of 70kHz.

The open loop results are presented in Figure 29, while Figure 30 reports the closed loop ones. The following error is reduced by a factor 5.5.

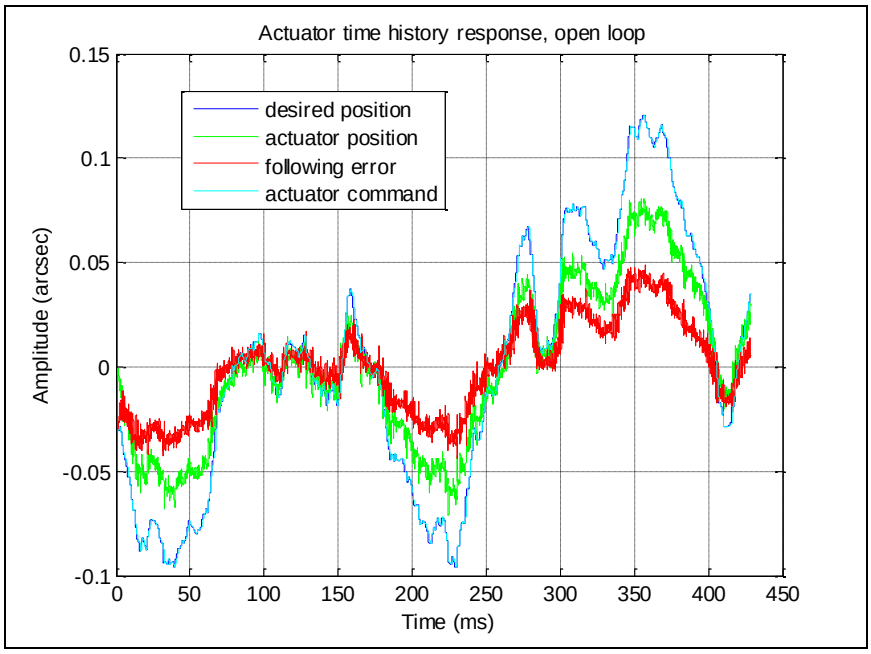


Figure 29 – Time history test in open loop. Following error: 1.69µm RMS corresponding to 21.2 mas RMS on-sky.

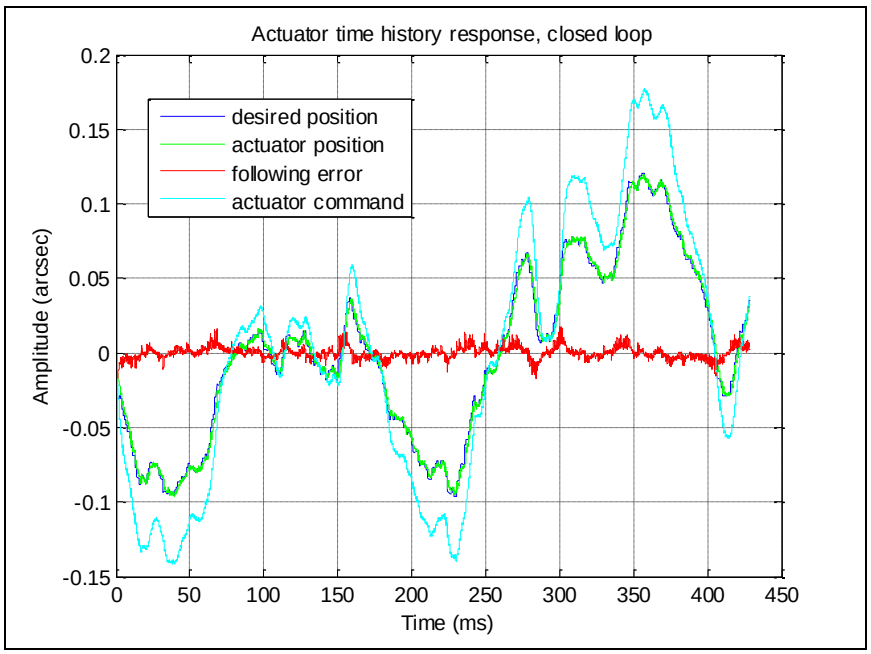


Figure 30 – Time history test in closed loop. Following error: 0.31µm RMS corresponding to 3.86 mas RMS on-sky.

At a first impression, the open loop response seems to be simply affected by a gain error. In reality this is not true, in fact the gain is the correct one but, as already pointed out in the step test and confirmed by the histeresys analysis, on small movements the histeresys affects the gain significantly.

Second, one might argue that the open loop results are not representative of the real behavior, in fact the outer optical loop would compensate for a large amount of the following error. This is true, but it should be also considered that the optical loop bandwidth might be poor, especially in low flux and bad seeing conditions, thus reducing the possibility of effectively correcting the error. This just a qualitative comment; to quantify this aspect, the reported results should be used as input for the error budget and/or optical loop simulation.

General comment on experimental results.

The presented results allow us to conclude that the adopted design is suitable to reach the desired performance. There is some concern on the actuator noise due to the signal conditioning chain, but it is actually difficult to do better, significant improvements on the circuit are not easy because it is already quite well optimized in terms of noise. We will investigate with PI (the actuator manufacturer) if the strain gage can be supplied with higher voltage (we are currently using 5V, most like 10V should be also acceptable), thus increasing its output gain and allowing to reduce the amplifier gain.

Nevertheless, the following error is already acceptable (even with the broken actuator) and it is more limited by dynamics rather than by noise.

6.1.1.3.6 CLMP loop tuning

The CLMP loop is based on a 4th order digital feed-back filter (see Figure 41). The coefficients of the control loop can be changed using the SET_DTT_CLMP_SERVO_COEFFS WIF command.

In order to facilitate the servo tuning an automatic tuning mechanism has been developed. The auto-tuning mechanism is based on both Matlab and the internal diagnostic buffers. A Matlab script tries to minimize a

given cost function by changing certain parameters, normally the PID parameters. The optimization procedure can be divided in the following steps:

1. Generation of a new set of parameters (PID)
2. Loading of the new control law derived from parameters of step 1 into the CLMP loop
3. Executing of a test using the diagnostic buffers, for instance execute a step response
4. Evaluation of the result and computing of a cost based on the error. For instance the cost could be the raise time, or ISE (integral square error), ISTE (integral square error per time), ISTTE (integral square error per square time). Go to step 1.

The minimization function is the Matlab fminsearch routine which is a multidimensional unconstrained nonlinear minimization routine, please refer to Matlab online help for more details.

In Figure 31 the result of an automatic tuning procedure is plotted. The green line shows the position of actuator at the end of the procedure and the red is the one at the beginning of the procedure, the improvement is clearly notable. In the same way the blue and red line at the bottom shows the error respectively at the end and the beginning of the auto-tuning procedure

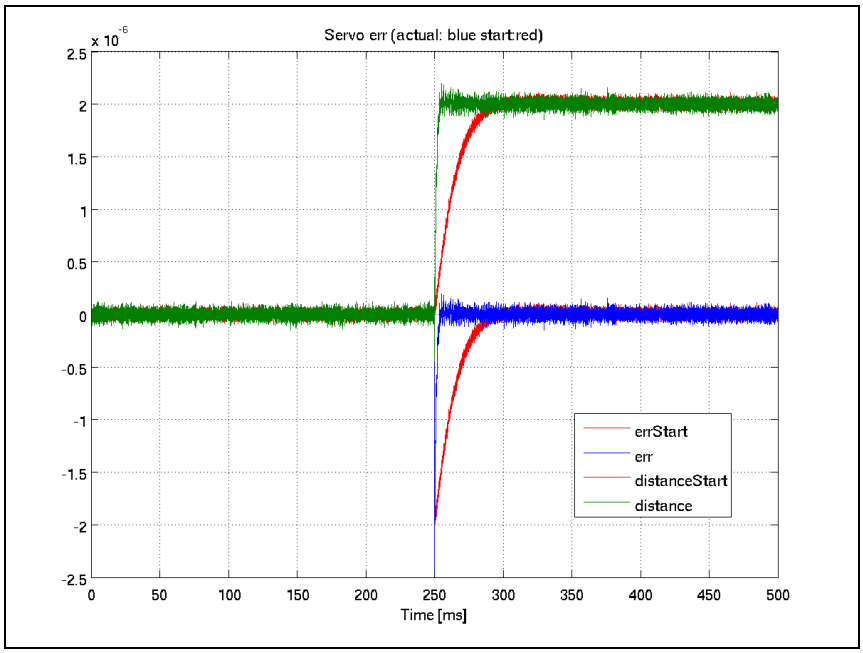


Figure 31 – Step response auto-tuning

6.1.1.3.7 CLMP loop servo oscillation detection

The auto-tuning is a fully automated procedure. While the cost function "drives" the Matlab fminsearch in the appropriate direction, it might be possible that the new control law derived from the automatic generated set of parameters is unstable. In order to deal with these potential dangerous situation an internal oscillation detection mechanism has been implemented on the HVC board. The oscillation detection mechanism works as follows: The error is continuously monitored for a given time, like a running window. In this window of time the variance is computed and if its value is a above a predefined limit for a certain period of time the system interprets this as an oscillation and automatically opens the control loop and brings the actuator to its default bias position. In Figure 32 an oscillation detection is reported.

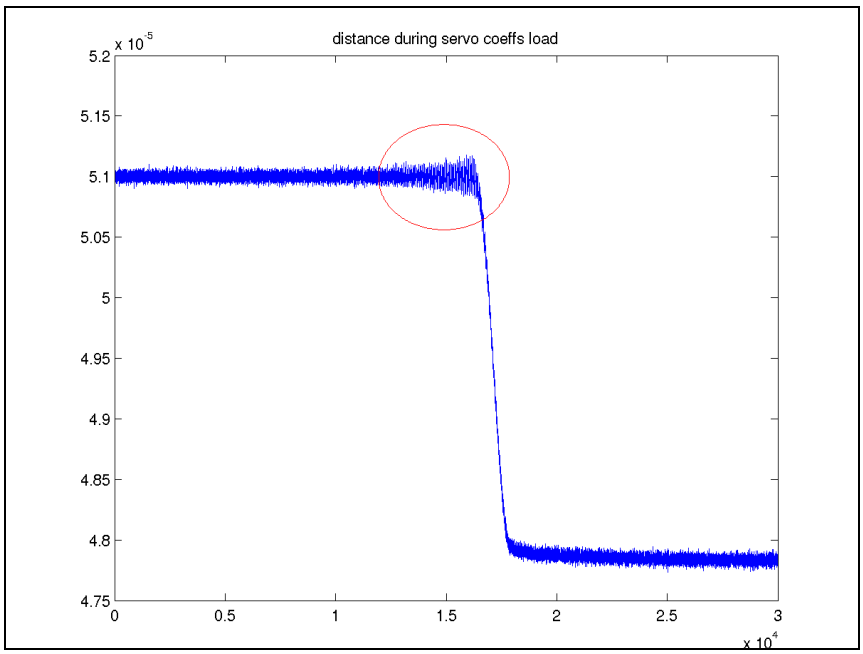


Figure 32 – Oscillation detection

6.1.1.4 Backplane

The system backplane is a passive board with the card-edge connectors for the AdOpt boards. There are different versions of backplane, with up to 17 available slots. For the NGWFC MGAOS we have foreseen a 12 slots backplane, that assures significant expansion possibility and allows at same time to fit all components (MGAOS and VME components) on a single 19" crate.

The backplane is completely passive. It comprehends two busses, 16 bit wide for diagnostic and 32 bits wide for real-time. Both busses operate at ~60 MHz, DDR. Considering the high communication speed and in particular the heavy load on the bus (the loaded bus impedance is down to 23Ω), we have selected B-LVDS technology. Thanks to this design choice, the backplane works properly regardless of the number and position of installed boards. Another advantage related to the use of B-LVDS technology is the reduction of EMI-RFI emission and sensitivity to external disturbance.

Each slot on the backplane is uniquely identified by hardware signals. In this way, it is not required to configure the AdOpt boards in any way. The correct addressing of the board is performed at system startup by an automatic procedure that recognizes the position of the board with respect to the other installed boards.

The backplane distributes also all power supplies (3.3V, ±12V, High Voltage) and some hardware flag dedicated to system operation (see Table 3 and Table 4).

While designing the system, great importance has been given to the reliability of the backplane, in particular considering the large number of connection points. To this aim, all connections are doubled on the two sides of the card edge connector. High quality, gold plated connectors are used and the gold plating thickness on the board terminals is checked during PCB production. To reduce the mechanical stress on the backplane during boards insertion and extraction, the backplane PCB is 3.2mm thick.

6.1.1.5 Reset circuitry

- The reset circuitry allows to handle properly the different reset sources, in particular:
- external reset input (see §6.3.2.3): this signal resets both the VME unit and the MGAOS. The reset input signal galvanically isolated from the system and referred to ground;

- VME reset (either by SW or through the pushbutton on the CPU board): also this signal resets both the VME unit and the MGAOS
- reset pushbutton on MGAOS unit: this resets only the MGAOS unit

6.1.1.6 WFS to BCU and BCU to DM HVA interface boards

The WFS and the DM HVA are interfaced to the PIO ports on BCU (see §6.1.1.1.11) by means of interface boards. These simply provide level shifting (from RS644 to LVTTL for WFS AIA interface and from LVTTL to RS485 for DM HVA interface) and proper connectors according to the existing WFS and DM HVA interfaces (see §6.3.2.3). The BCU to DM HVA interface board also provides the interface for STRAP synchronous operation and to IRIG board for system time-stamping synchronization .

The WFS to BCU interface comprehends also the serial line used for SciMeasure CCD controller configuration. The BCU acts as bridge between WIF (where the CCD configuration actually occurs) and the Little Joe controller.

Mechanically both boards are mounted on a single slot MGAOS front panel, see Figure 2.

6.1.2 VME components

The VME crate is based on a Motorola Power-PC single board computer.

The other main components on the VME crate are:

- Symmetricom ttm635VME timing board
- STRAP (provided by Keck)
- Power supply unit

The MGAOS is also integrated into the VME crate and is described separately in 6.1.1.

6.1.2.1 CPU board

The CPU board is the high performance MVME6100-0173 Power-PC VME single board computer from Motorola.

The key features of the MVME6100 board in the selected configuration are reported hereafter:

- Processor: MPC7457 PowerPC® processor running at 1.267 GHz,
- Co-Processor: 128-bit AltiVec™ coprocessor for parallel processing
- RAM Memory: 1GB DDR ECC memory + 512KB L2 cache + 3MB L3 cache
- Permanent memory: 128 MB Flash
- On board expansions: two PMC expansion slot
- VME bus interface: 2eSST VMEbus protocol with 320MB/s transfer rate across the VMEbus
- Ethernet: two Gigabit ports over copper
- Serial: two 16550 compatible ports

6.1.2.2 IRIG decoder

To synchronize the telemetry time-stamping clock with an external timing source the RTC is equipped with the Symmetricom ttm635VME timing board

The main characteristics and functionalities of this board are the following:

- Time on demand (days through 0.1 microseconds) with zero latency. This feature is implemented with hardware registers which latch the current time upon host request.
- Event logging (days through 0.1 microseconds). This feature is implemented with a second set of hardware registers. Time is captured on a positive or negative input edge.
- Six operational modes are supported. Modes are distinguished by the reference source.

Mode	Source Of Synchronization
0	Timecode – IRIG-A IRIG-B XR3 2137 NASA 36
1	Free running - on board VCXO used as reference.
2	1 PPS - accepts input one pulse per second.
3	RTC - uses battery backed on board real time clock IC.
5	GPS (optional) - double wide configuration including GPS receiver.
6	GPS (optional) - uses GPS receiver/antenna (receiver in antenna).

- Provides an output clock synchronized to the selected reference; programmable 1, 5, or 10MHz TTL.
- All modes of operation are supplemented by flywheel operation. For example, if synchronization source is lost, the TFP will continue to function at the last known reference rate.
- Generates synchronized IRIG B timecode. Modulated and DC level shift formats are produced simultaneously. Also generates IRIG H DC level shift.
- Programmable frequency output (periodics) is provided. The output frequency is 10,000,000/ (n1 * n2), where 1<n1<65536 & 1<n2<65536.
- A time coincidence strobe output is provided. Programmable from hours through milliseconds. This strobe also has an each second mode programmable to milliseconds.
- Five maskable interrupt sources are supported. IRQ levels one through seven are programmable.

Int.	Source Of Interrupt
0	External event input has occurred.
1	Periodic output has occurred.
2	Time coincidence strobe has occurred.
3	One second epoch (1PPS output) has occurred.
4	Output data packet is available.

- Time-of-day, hours, minutes, and seconds are displayed on front panel LED's.
- Most inputs and outputs are accessible via the P2 connector.

6.1.3 Power dissipation

The total power dissipation is reported on Table 10. The reported values refer to

Device	Dissipation
MGAOS BCU	12.5 W
MGAOS DSP (3x)	20.4 W
MGAOS HVC	13.7 W
VME CPU BOARD	42 W

VME IRIG BOARD	8.5 W
STRAP	19.5 W
Total device power	*116.6*
Power conversion efficiency	75%
Total dissipated power	**155 W**

Table 10 – System crate power dissipation summary

The typical power supplied by the MGAOS and VME PSU is 108W, while the HV PSU supplies just 8.6W.

6.1.4 Power supply system

The RTC power supply comes from two units, one supplying the VME and MGAOS system the other supplies the HVC.

6.1.4.1 VME and MGAOS power supply

The MGAOS requires +3.3V, ±12V and high voltage supply. VME requires +5V and ±12V. All these power rails (with exception of the high voltage) are available on quad-output PSUs typically used for Compact PCI systems. Therefore, even if the typical VME PSUs are triple output, we decided to use a Compact PCI PSUto reduce the number of parts and allow simultaneous power up of the different subsystems.

The PSU (Schroff CPCI-AC-6U-400) supplies the following components

- Motorola CPU board
- Timing board
- STRAP
- MGAOS

This power supply unit has an overall load capability at sea level of 400 W. Applying the de-rating for high altitude operation, the maximum output power is reduced to

400 x 0.8 = 320 W

The voltages and maximal currents are listed in the following table

Voltage	available current (max)	MGAOS	Motorola CPU	Irig Timing board	STRAP	TOT current
+5V	50A		10.2A	1.5A	5A	16.7A
+3.3V	80A	10A				7°
+12V	7.5A	0.060 A		0.050A	1.5A	1.61A
-12V	1.5A	0.060 A		0.030A	1A	1.09A
					TOT Power (max)	**149W**

Table 11 – Voltages and currents of RTC power supply rails.

The power supply is well oversized with respect to the actual power demand.

6.1.4.2 HVC supply

The HVC power supply requirements are listed hereafter:

Requirement	Value	Comment
Output voltage, positive rail	>132V	120V maximum actuator voltage + 12V for power

		amplifier headroom
Output voltage, negative rail	<-32V	-20V maximum negative actuator voltage - 12V for power amplifier headroom
Output current	> 120mA x 6 = 720mA	Specification based on maximum amplifier peak current. It exceeds significantly the expected maximum current on each channel. Conservative assumption.
Availability of over-current protection	-	-

Table 12 – High voltage power supply unit requirements.

The selected power supply is an Agilent N6700B modular system equipped with three N6736B modules @60V each.

The main characteristics of this power supply are reported in Table 13.

Specification	**Value**
Input voltage	86~264 VAC, 50~60 Hz
Power	400 W
Output voltage	freely adjustable, (depending on module equipment)
Module characteristics (N6736B)	100V, 0.5A 50W

Table 13 – Agilent N6700B specifications.

This power supply unit can be remotely controlled through a LAN, GPIB and USB. This power supply will be controlled by a GPIB interface provided by Keck, similar to the one that already controls the HVA power supply.

6.1.5 Mechanical aspects and cooling

The MGAOS is contained in a dedicated chassis that fits into a standard 6U, 19”crate (see Figure 2). The chassis appears as a ‘sub-crate’ with 12 slots, 14mm spacing. The width of the chassis is 223.52mm, i.e. 44 TE. The chassis does not make use of standard VME backplane. All power and control connections required for system operation are on standard DIN 41612 connector on the sub-crate rear panel.

6.1.5.1 Cooling

The system is air-cooled and forced air is provided by a standard fan-drawer, that also cools the VME components installed on the main crate.

Considering the low power dissipation (see Table 10), and also on base of previous experience, a forced air flow of 100 LFpM. The required air flow has to be increased according to pressure reduction due to altitude. Therefore a flow of 100/0.8 = 125 LFpM shall be obtained. This can be easily achieved by a standard 19” fan drawer. We plan to install two 3-fans, 1U drawers, one on the front side of the crate (cooling VME boards and MGAOS), the other on the rear side, cooling the PSUs.

We plan to measure the actual air speed on components in order to verify experimentally that the above specification is actually met.

6.2 TRS Hardware

The TRS Hardware is based on two main components:

- Storage Server
- Disk Array

The storage server is linked to the MGAOS trough a FibreChannel link and to the LAN through Ethernet.

The Disk Array is connected only to the Storage Server through a FibreChannel link. The interface between the Server and the Disk Array supports both Arbitrated Loop and Fabric Switching topologies; it also supports FC-4 Layer SCSI-FCP and FC-0 Layer Hot Pluggable SFP. Thus an upgrade to a SAN system is straightforward.

6.2.1 Storage Server

The storage server is an SUN X4100 Model. This is a high performance 19" 1U rack type server with these main characteristics:

- CPU: Dual AMD Opteron Model 252 Processors
- 2-GB Memory
- Disk: 2x36-GB 10000 RPM SAS Disk Drive
- Redundant Fans
- 2 Power Supply Units
- 4 10/100/1000 Ethernet Ports
- FibreChannel Device: 2 SANblade Qlogic 2340

Just the operating system and the query/storage software reside on the server. All telemetry data is stored on the disk array.

6.2.2 Disk array

For the disk array system the SurfRAID TRITON 16FA has been chosen. This is a SATA RAID controller with FibreChannel interface. It is equipped with 16 Hitachi 400GB SATA disks for a total raw storage capacity of 6 TB. This is slightly more then the real storage needs, but this allows to configure the system with some redundancy. In fact the TRITON 16 FA supports RAID levels 0, 1, 3, 5, 6 and 0+1.

The basic idea of RAID is to combine multiple disk drives into an array of disk drives to obtain performance, capacity and reliability that exceeds that of a single large drive. The array of drives appears to the host computer as a single logical drive. Each RAID level provides disk fault-tolerance with different compromises in features and performance. Here is a brief description of the various RAID levels:

- RAID 0 is defined as a group of striped disk drives without parity or data redundancy. RAID0 deliver the best data storage efficiency and performance of any array type. The disadvantage is that if one drive in a RAID 0 array fails, the entire array fails.
- RAID 1, also known as disk mirroring, is simply a pair of disk drives that store duplicate data but appear to the host as a single drive. Each write operation must go to both drives of a mirrored pair so write performance is not increased in respect to a single drive. However, each individual drive can perform simultaneous independent read operations doubling in fact the read performance.
- RAID 3 stripes data across groups of drives, but one drive in the group is dedicated to storing parity information. Records typically span all drives, which optimizes the disk transfer rate. Because each I/O request accesses every drive in the array, RAID3 arrays can satisfy only one I/O request at a time. RAID 3 delivers the best performance for single-user, single-tasking environments with long records.
- RAID 5. In this case parity information is distributed across all the drives. Since there is no dedicated parity drive, all drives contain data and parity, read operations can be overlapped on every drive in the array. Write operations will typically access one data drive and one parity drive, however, because different records store their parity on different drives, write operations can usually be overlapped.

- RAID 6 is similar to RAID 5 in that data protection is achieved by writing parity information to the physical drives in the array. With RAID6, however, two sets of parity data are used. The main advantages of RAID 6 is high data availability, any two drives can fail without loss of critical data.

Considering that the storage is not shared but is used only by one single host (the TRS server) RAID 3 is probably the best choice achieving best performance with acceptable reliability. To increase availability, two drives has been configured as hot spare drives, in case of failure this will automatically replace the damaged disk. The damaged disk can then be hot swapped without interrupting the disk array operation

Using RAID 3 with 2 hot swappable disks the total available capacity amounts to 5.6 TB. The actual storage capability in terms of observing time clearly depends on the observing mode and the amount of configuration data, which is not predictable. Under realistic assumptions we can predict a storage depth of more then five nights (~50 hours) operating at 2kHz frame rate with CCID56.

Another remarkable feature of the TRITON 16FA is that it comes with an environment controller which is capable of accurately monitoring the internal environment such as power supplies, fans, temperatures and voltages. Any malfunctioning can be automatically emailed. Maintenance and configuration of the disk array can be accomplished over the front panel, via a VT100 terminal attached to its serial line, or over Ethernet by telnet or http (web interface) protocol.

The TritonF16 disk array is housed into a standard 19”, 3U cage

6.3 HW Interfaces

In this section we describe the hardware interfaces, distinguishing between *internal* and *external* interfaces. As *internal interfaces* we refer to the connection points connecting internal subsystems of the NGWFC RTC that are delivered by Microgate.

Conversely, *external interfaces* are all connections among Microgate-furnished items and external systems.

Moreover, the hardware interfaces are divided into *electrical* and *data*.

6.3.1 Internal HW interfaces

6.3.1.1 Power and logic interfaces

Interface description	Node A	Node B	Voltage/power rating/level	Connector
MGAOS 'sub-crate' supply and control connector.	MGAOS 'sub-crate'	MGAOS and HV PSU, VME reset board	See Table 15.	DIN 41612 type H7/F24 connector (31 poles)
STRAP synchronization	MGAOS BCU board (PIO #0 port)	STRAP Ext.Sync. input	TTL level. Triggers on falling edge	SMB
Periodic synchronization of RTC	MGAOS BCU board (PIO #0 port)	IRIG board	TTL level. Triggers on raising edge	SMB on MGAOS, BNC on Symmetricon IRIG board

Table 14 – Internal HW interfaces. Power and logic.

	Pin	Description	Connects to	Specification
Upper (P1) connector	H4,H6	Logic supply	VME+MGAOS PSU	3.3 V, 10 A
	H14	Logic supply sense+	VME+MGAOS PSU	
	H18, H20	Logic ground return	VME+MGAOS PSU	-
	H32	Logic supply sense-	VME+MGAOS PSU	
	H28	Analog supply, positive rail	VME+MGAOS PSU	+12 V, 0.2 A
	H10	Analog ground return	VME+MGAOS PSU	
	H16	Analog supply, negative rail	VME+MGAOS PSU	-12 V, 0.2 A
	H24	High voltage supply, positive rail	HV PSU	132 V, 0.72 A
	H12	High voltage ground return	HV PSU	
	H26	High voltage supply, negative rail	HV PSU	-32 V, 0.72 A
	H30, H8, H22	Not used		
Lower (P2) connector	H6	VME reset	VME reset board	TTL-compatible, open collector, active low
	H8	external reset	External input	TTL-compatible, optically isolated, active low (switch to GND to reset)
	H16	MGAOS booting mode	Available, but not connected	TTL-compatible, high=user, low=default
	H24	Logic ground		
	H4,H10,H12,H14,H18, H20,H22,H26,H28,H30	Not used		
	H32	Chassis ground		

Table 15 – MGAOS supply and control connector pinout and specifications. Non listed pins are reserved.

6.3.1.2 Communication interfaces

Interface description	Node A	Node B	Connector	Cable
Gbit Ethernet interface between MGAOS BCU and VME CPU board	MGAOS BCU	VME CPU	Standard RJ45 connector	UTP 5E
FibreChannel interface between MGAOS BCU and TRS server	MGAOS BCU	TRS server	2x Dual LC-type connector	2 multimode fibers 62.5-125µm or 50-125µm (**TBD**), 850nm (2.125 Gbit/s, TX/RX)
FibreChannel interface between TRS server and Disk Array	TRS server	Disk Array	2x Dual LC-type connector	2 multimode fibers 62.5-125µm or 50-125µm (**TBD**), 850nm (2.125 Gbit/s, TX/RX)

Table 16 – Internal HW interfaces: communication.

6.3.2 External HW interfaces

6.3.2.1 Power interfaces

Interface description	From	To	Voltage and power rating	Connector
Crate main supply (unique for VME, MGAOS and HVC PSU)	RTC crate	Main supply	86-264 Vac, 47~63 Hz, max 400 VA	Standard 3-way socket, with EMI-RFI filter.
TRS server main supply	TRS server	Main supply	86-264 Vac, 47~63 Hz, max **TBD** VA	Standard 3-way socket, with EMI-RFI filter.
Disk Array main supply	Disk Array	Main supply	86-264 Vac, 47~63 Hz, max **TBD** VA	Standard 3-way socket, with EMI-RFI filter.

Table 17 – External HW interfaces: power.

6.3.2.2 Communication interfaces

Interface description	Node A	Node B	Connector	Cable
Gbit Ethernet interface on VME CPU	VME CPU	Ethernet LAN	Standard RJ45 connector	UTP 5E
Gbit Ethernet interface on TRS	TRS	Ethernet LAN	Standard RJ45 connector	UTP 5E

Table 18 – External HW interfaces: data.

6.3.2.3 Logic and analog interfaces

Interface description	Node A	Node B	Logic level	Connector	Cable
External reset and boot selector	Telescope control system (TBC by Keck)	VME crate	5V to 24V. Active high. Can be isolated from system ground.	3 poles Neutrik female panel connector	Three poles, section AWG 26 or bigger
Interface between SciMeasure Little Joe WFS and MGAOS	SciMeasure Little Joe AIA port	RS644/LVTTL converter on MGAOS	RS644 differential signals	SCSI-type 68 pin connector	Standard SCSI cable (twisted pairs)
Interface between MGAOS and Xinetics HVA	LVTTL /RS485 converter on MGAOS	Xinetics High Voltage Amplifier	RS485 differential signals	96 pole DIN 41612 connector (identical to current interface)	96 poles flat cable (already in use at Keck)
Actuator interface between HVC board and DTT mirror	HVC board	DTT mirror actuators	High voltage control signals and strain-gauge signals	See Table 20.	
Actuator interface between HVC board and UTT mirror	HVC board	UTT mirror	High voltage control signals and strain-gauge signals	See Table 21.	

Table 19 – External HW interfaces: logic.

	HVC board		Mirror	
Description	**Connector/Socket**	**Pin**	**Connector/Plug**	**Pin**
HV actuator #A	LEMO ERN.00.250.CTL	1	LEMO FFS.00.250.CTA	1
HV GND, actuator #A		5		2
HV, actuator #B	LEMO ERN.00.250.CTL	2	LEMO FFS.00.250.CTA	1
HV GND, actuator #B		5		2
HV, actuator #C	LEMO ERN.00.250.CTL	3	LEMO FFS.00.250.CTA	1
HV GND, actuator #C		5		2
Strain gauge supply, act. #A	LEMO EGG.00.304.CLL	1	LEMO FFA.0S.304 male	1
Strain gauge GND, act. #A		4		4
Strain gauge +, act. #A		3		3
Strain gauge -, act. #A		2		2
Strain gauge supply, act. #B	LEMO EGG.00.304.CLL	1	LEMO FFA.0S.304 male	1
Strain gauge GND, act. #B		4		4
Strain gauge +, act. #B		3		3
Strain gauge -, act. #B		2		2
Strain gauge supply, act. #C	LEMO EGG.00.304.CLL	1	LEMO FFA.0S.304 male	1
Strain gauge GND, act. #C		4		4
Strain gauge +, act. #C		3		3
Strain gauge -, act. #C		2		2

Table 20 – DTT mirror interface. The strain gage connection on the HVC board uses a different connector due to space constrains. To connect the actuator, we will use a dedicated cable adaptor with 1-to-1 connection to ‘00’ and ‘0S’ connectors, or, preferably, the connector on the actuator end will be replaced with a ‘00’ series.

	HVC board		Mirror	
Description	**Connector/Socket**	**Pin**	**Connector/Plug**	**Pin**
HV actuator, X axis	LEMO ERN.00.250.CTL	1	LEMO FFS.00.250.CTA	1
HV GND, X axis		5		2
HV, actuator, Y axis	LEMO ERN.00.250.CTL	2	LEMO FFS.00.250.CTA	1
HV GND, Y axis		5		2
HV, common reference	LEMO ERN.00.250.CTL	3	LEMO FFS.00.250.CTA	1
HV GND, common reference		5		2
Strain gauge supply, X axis	LEMO EGG.00.304.CLL	1	LEMO FFA.0S.304 male	1
Strain gauge GND, X axis		4		4
Strain gauge +, X axis		3		3
Strain gauge -, X axis		2		2
Strain gauge supply, Y axis	LEMO EGG.00.304.CLL	1	LEMO FFA.0S.304 male	1
Strain gauge GND, Y axis		4		4
Strain gauge +, Y axis		3		3
Strain gauge -, Y axis		2		2

Table 21 – UTT mirror interface. The strain gage connection on the HVC board uses a different connector due to space constrains. To connect the actuator, we will use a dedicated cable adaptor with 1-to-1 connection to '00' and '0S' connectors, or, preferably, the connector on the actuator end will be replaced with a '00' series.

Interface description	**From**	**To**	**Signal level**	**Connector**	**Pinout**
HVC input summing junction signals	Any external device (for maintenance and/or development purposes)	HVC board	-1.67V to 10V. The input signal is summed to the regular actuator command and amplified 12x. Input impedance >1MΩ.	SCSI-type 50 pin connector, female, on HVC analog high voltage board	Pins 2,6,10,14,18,22: input channels Pins 5,9,13,17,21 and all pins from 26 to 50: GND
HVC diagnostic output signals	HVC board	Any external device (for maintenance and/or development purposes)	-2V to 10V. The output signal is 1/10 of the actuator output. Output dynamic impedance <10Ω.	SCSI-type 50 pin connector, female, on HVC analog high voltage board	Pins 3,7,11,15,19,23: output channels Pins 5,9,13,17,21 and all pins from 26 to 50: GND
Strain gauges diagnostic output signals	HVC board	Any external device (for maintenance and/or development purposes)	Gain: 50mV/µm. Output dynamic impedance <10Ω.	SCSI-type 50 pin connector, female, on HVC analog high voltage board	Pins 4,8,12,16,20,24: output channels Pins 5,9,13,17,21 and all pins from 26 to 50: GND

Table 22 – HVC board diagnostic interface. This interface is not used during standard operation, therefore it is routed only to an internal connector on HVC analog board. It is possible however to bring the signals externally by means of a ribbon connector.

6.4 Reliability

This chapter reports the preliminary reliability analysis performed on NGWFC RTC. The reliability is based on prediction according to standard methods for what concerns the proprietary electronics. For COTS components, the data reported by the manufacturer have been used.

6.4.1 MGAOS system reliability prediction

This chapter reports the preliminary reliability analysis performed on MGAOS electronics.

The reliability prediction has been computed according to the recommendation of [RD3], Notice 2, Parts Count Method. This method has been used to predict the reliability of each board, and then the partial result have been combined to compute the reliability of the complete control system.

The following conditions have been assumed:

- Operating environment: *ground, benign*. MGAOS will operate in a well protected environment, at moderate temperature and low humidity. There is no significant mechanical stress on the system.
- In most of the cases, the actual reliability factor published by the manufacturer of each electronic component has been used. The values reported for 55°C temperature and 90% confidence have been used. When actual production data were not available (e.g. resistors, connectors), the computation has been based on typical values recommended by [RD3] for the component type.
- The reliability data of the selected high voltage OP-AMP (Apex PA243) are not available. Data referring to amplifier from the same manufacturer with comparable electrical characteristics range from FITs ≅ 400 for MIL-STD screened versions to FITs ≅ 4500 for industrial versions. However one should take into consideration that the parts of which reliability data are available are all hybrids, while the PA243 device is monolithic. It is reasonable to expect a better reliability on the monolithic part, therefore a value of FITs=500 has been preliminarily assumed for this component.

Qty	Part Number	FITs, single part	FITs, total	Contribution
3	AdOpt DSP board	981	2943	33.3%
1	AdOpt BCU Board	503	503	5.7%
1	AdOpt Backplane Board	154	154	1.7%
1	AdOpt HVC board	4895	4895	55.4%
1	WFS AIA to BCU interface board	147	147	1.7%
1	BCU to DM HVA interface board	147	147	1.7%
160	Flat cables single connections	0.26	42	0.5%
		FITs	8830.6	

Computed MTBF (hours)	113,243

Table 23 – MGAOS electronics reliability prediction

6.4.2 Complete VME crate reliability prediction

The partial results obtained for the MGAOS system has been combined with the data of the COTS VME components using the same approach.

The assumptions have been taken:

- Reliability data for the Symmetricom IRIG timing board are not available. The same reliability of the CPU board has been assumed.
- STRAP reliability has not been taken into account, considering that this device is a pre-existing component, not being strictly part of the RTC system. In this perspective, it has been considered similarly to the WFS camera or to the DM HVA.

Qty	Part Number	FITs, single part	FITs, total	Contribution
1	MGAOS	8,831	8,831	24.6%
1	VME CPU board	5,605	5,605	15.6%
1	Symmetricom IRIG board	5,605	5,605	15.6%
1	VME/MGAOS PSU	3,584	3,584	10.0%
1	HV PSU	12,325	12,325	34.3%
		FITs	35,950	

Computed MTBF (hours)	27,816

Table 24 – Complete RTC crate reliability prediction

6.4.3 TRS system reliability

The declared MTBF of the Triton disk array is 300,000 hours, under normal operating conditions (ground, benign). There are no data available for the TRS server.

The actual disk array reliability might be an issue due to the reduced atmospheric pressure at site altitude. This might reduce significantly the lifetime of the disks heads. The disks chosen for this application have fluid-dynamic bearings, instead of air ones. This should guarantee an increased reliability under low pressure operating conditions.

6.4.4 Spares

The RTC electronics spare list has been computed considering a lifetime of 10 years, 12 hours/day operating time, and on base of the reliability data reported in Table 23 and Table 24. We have considered two system operating simultaneously. The number of spares has been estimated imposing a 90% confidence level. The reported numbers do not take into account the availability of the third spare system, which is part of the deliverables.

Qty	Part Number	MTBF (single board)	Spare parts, 90% confidence (total for 2 systems)
6	AdOpt DSP board	1,019,368	1
2	AdOpt BCU Board	1,988,072	1
2	AdOpt Backplane Board	6,493,506	0
2	AdOpt HVC board	204,290	1
2	WFS AIA to BCU interface board	6,802,721	1
2	BCU to DM HVA interface board	6,802,721	1
2	VME CPU board	178,412	1
2	Symmetricom IRIG board	178,412	1
2	VME/MGAOS PSU	279,018	1
2	HV PSU	81,136	2

Table 25 – List of spares for 10 years of operation of two systems, 90% confidence.

All spares are considered as ‘line replaceable units’, i.e. all components can be simply replaced in short time (typ < 30 min), without requiring any particular operation to be performed.

7 SOFTWARE DESIGN

A very clear overview of the software blocks and related data flow is given in [AD4], §5.2. From the implementation point of view can distinguish three different software branches.

- MGAOS resident software, where all strictly real-time computations are performed (WFP, DTT/UTT controller)
- MVME6100 resident software, implementing the interface between WCP and RTC and housekeeping functions (WIF). The WCP is also implemented on the same CPU, but is under Keck responsibility
- TRS software, managing the telemetry database storage and query (TRS)

In the following sections we analyze implementation and performances of the above mentioned main software categories.

7.1 MGAOS Software

The MGAOS resident software is responsible of almost all WFP computation. The tasks computed by the DSP software are the following:

- Centroids computation
- Residual Wavefront computation
- Control law servo computation
- Real-time telemetry acquisition
- HVC real time control software

7.1.1 MGAOS data flow

Before entering in the reconstructor algorithm details, it is important to describe the data flux and the interaction between the various RTC components.

The following figure describes the various parts of the MGAOS and the interaction with the external devices:

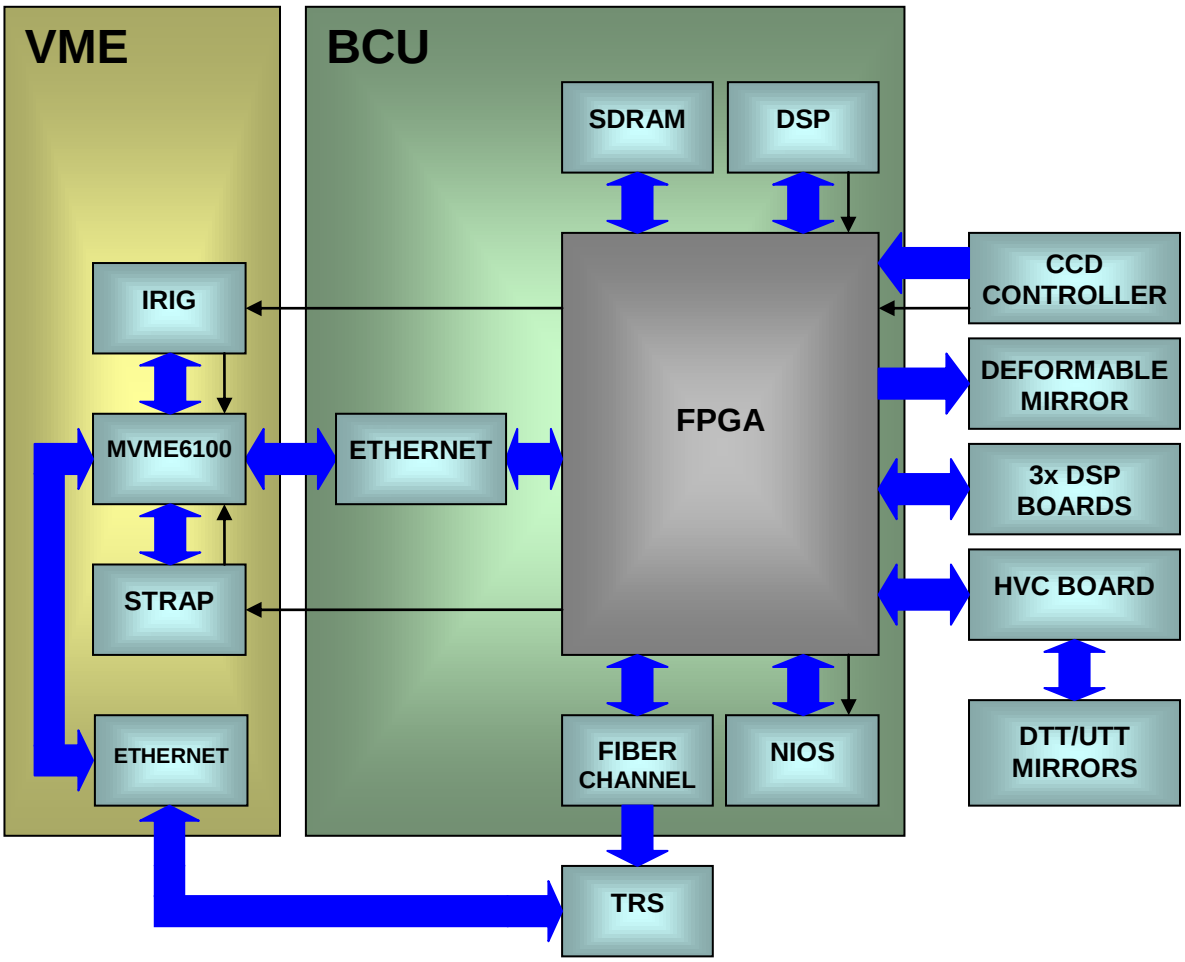


Figure 33 - RTC main scheme

The complete description of components and interconnections could be found in 5 and 6, here after there is a brief description to summarize the components involved in the NGWFC. The system is structured in three main components, the VME rack, the MGAOS crate and the TRS subsystem.

- The VME rack includes the MVME6100, IRIG and STRAP boards;
- The MGAOS crate includes the BCU board which is the arbiter of the real time data flux, three DSP boards to compute the residual wavefront matrix multiplier and one HVC board to compute the DTT and UTT mirror commands and to drive the high voltage tip-tilt mirror actuators.
- The TRS subsystem includes a server and a disk array for telemetry data storing.

The BCU board is the node of the real time system and it is directly connected to almost of peripherals. This board includes:

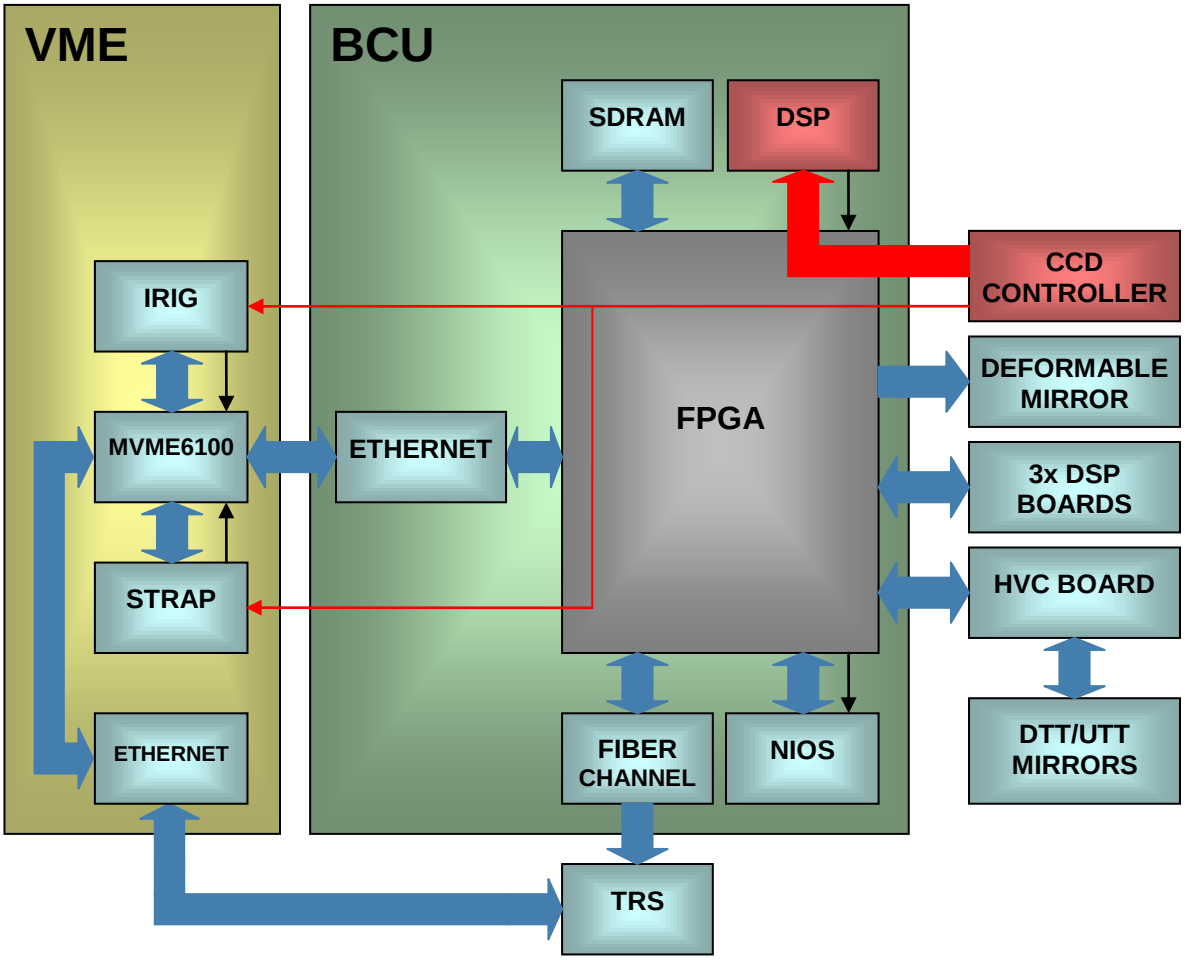
- a FPGA logic which controls all the peripherals and implements the various interfaces;
- the SDRAM used to temporary store the telemetry data before to send to the TRS;
- a DSP which computes the first part of reconstructor algorithm and manages the real time data flux;

- a NIOS controller which manages the diagnostic acquisitions and checks, the Ethernet and Fiber Channel interpreter and some real time tasks;
- the Fiber Channel interface;
- the Ethernet interface;
- the CCD controller interface;
- the Xinetics High Voltage driver interface;
- the DSP and HVC boards interface (two buses at 2Gbit/sec and 4Gbit/sec of bandwidth for diagnostic and real time data transfer respectively).

The real time data flux can be divided in a sequence of steps, some steps begin at the end of the previous step instead others are executed in parallel. Each step takes a fixed time or depends to the system configuration. The following paragraph 7.1.1.1 describes the operations performed by each step and the paragraph 7.1.1.2 the duration of each step.

7.1.1.1 Real time steps and processing time

STEP 1:



The real time sequence begins when a start of frame signal followed by a new CCD image arrives from the CCD controller. The acquired data is reorganized according to the LUT and downloaded to the DSP of the BCU board and the centroid computation starts. The start of frame signal is sent also to the external interrupt lines of IRIG and STRAP systems.

The first step duration depends on the CCD type and setting used. Starting from the table 7-1 of [AD1] the following table indicates in the various CCD conditions the real centroid computation time (T3 in the Figure 34).

CCD	Bin	Raw CCD size	Binned CCD size	Subaperture size (after binning)	Subaperture spacing (pixels)	Max frame rate (Hz)	Centroid computational time (µs)
CCD-39	1	80x80	80x80	2x2	4	2000	131
CCD-39	2	80x80	40x40	2x2	2	2000	131
CCD-39	1	80x80	80x80	4x4	4	2000	221
CCiD-56	1	160x160	160x160	2x2	8	1000	131
CCiD-56	2	160x160	80x80	2x2	4	2000	131
CCiD-56	4	160x160	40x40	2x2	2	2000	131
CCiD-56	1	160x160	160x160	4x4	8	1000	221
CCiD-56	2	160x160	80x80	4x4	4	2000	221
CCiD-56	1	160x160	160x160	8x8	8	1000	582

Table 26 - Centroid computational time using the subaperture flux

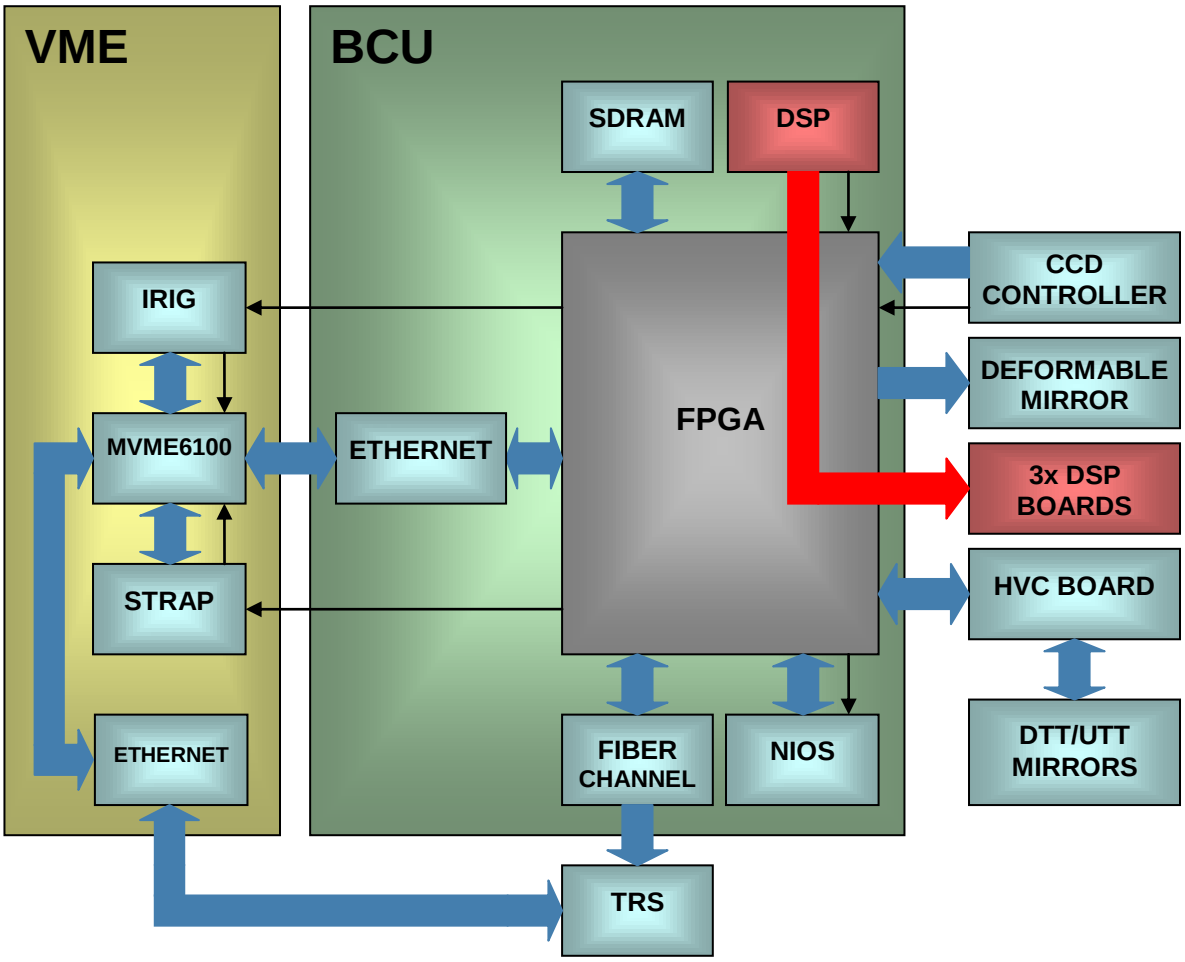
In case of denominator free centroiding the computational time is lower, see the following table.

CCD	Bin	Raw CCD size	Binned CCD size	Subaperture size (after binning)	Subaperture spacing (pixels)	Max frame rate (Hz)	Centroid computational time (µs)
CCD-39	1	80x80	80x80	2x2	4	2000	90
CCD-39	2	80x80	40x40	2x2	2	2000	90
CCD-39	1	80x80	80x80	4x4	4	2000	181
CCiD-56	1	160x160	160x160	2x2	8	1000	90
CCiD-56	2	160x160	80x80	2x2	4	2000	90
CCiD-56	4	160x160	40x40	2x2	2	2000	90
CCiD-56	1	160x160	160x160	4x4	8	1000	181
CCiD-56	2	160x160	80x80	4x4	4	2000	181
CCiD-56	1	160x160	160x160	8x8	8	1000	541

Table 27 - Centroid computational time using the denominator free centroiding

As mentioned in section 7.1.3 the centroid computation can be executed in parallel to the CCD pixels download; in this case if the centroid computational time is slower than the pixel read out the required time between the last pixel downloaded and the last centroids computed is absolutely negligible. As indicated in Table 26 and Table 27 for all the CCD type and configuration required the real computational time is significantly slower respect to the pixels download which is typically the 90% of the frame rate.

STEP 2:



When the centroid computation is finished the new centroid vector is sent to the DSPs of the DSP boards and the residual wavefront vector and control law servo loop computation start.

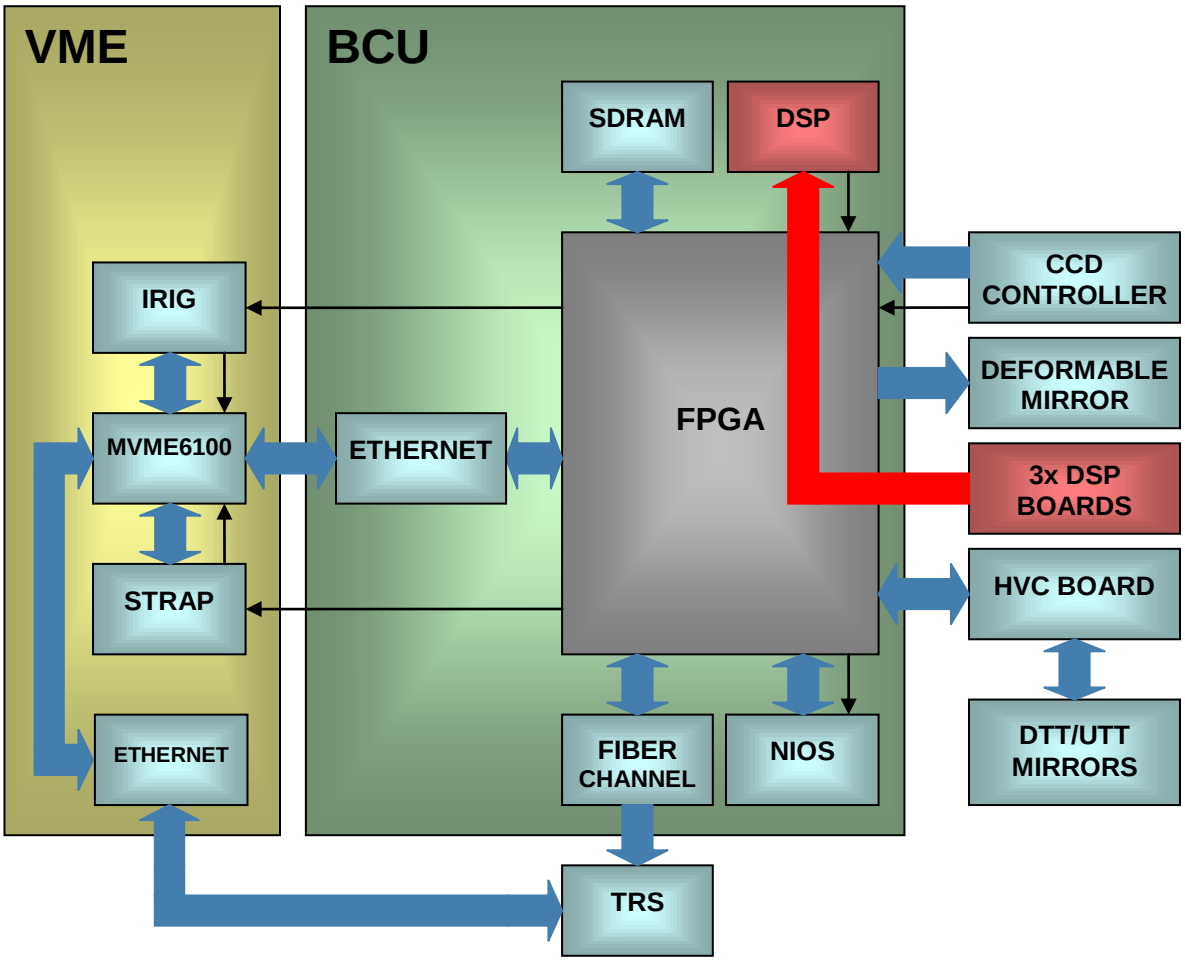
For the current MGAOS configuration (number of DSP boards and DSPs' speed) this step takes:

vector download (µs)	computation (µs)	total (µs)
7	82	89
T4 in the Figure 34	T5 in the Figure 34	

At the end of the real time critical part the DSP boards pre-compute the control law servo loop of the next step, this phase takes:

pre-computation (µs)
15
T6 in the Figure 34

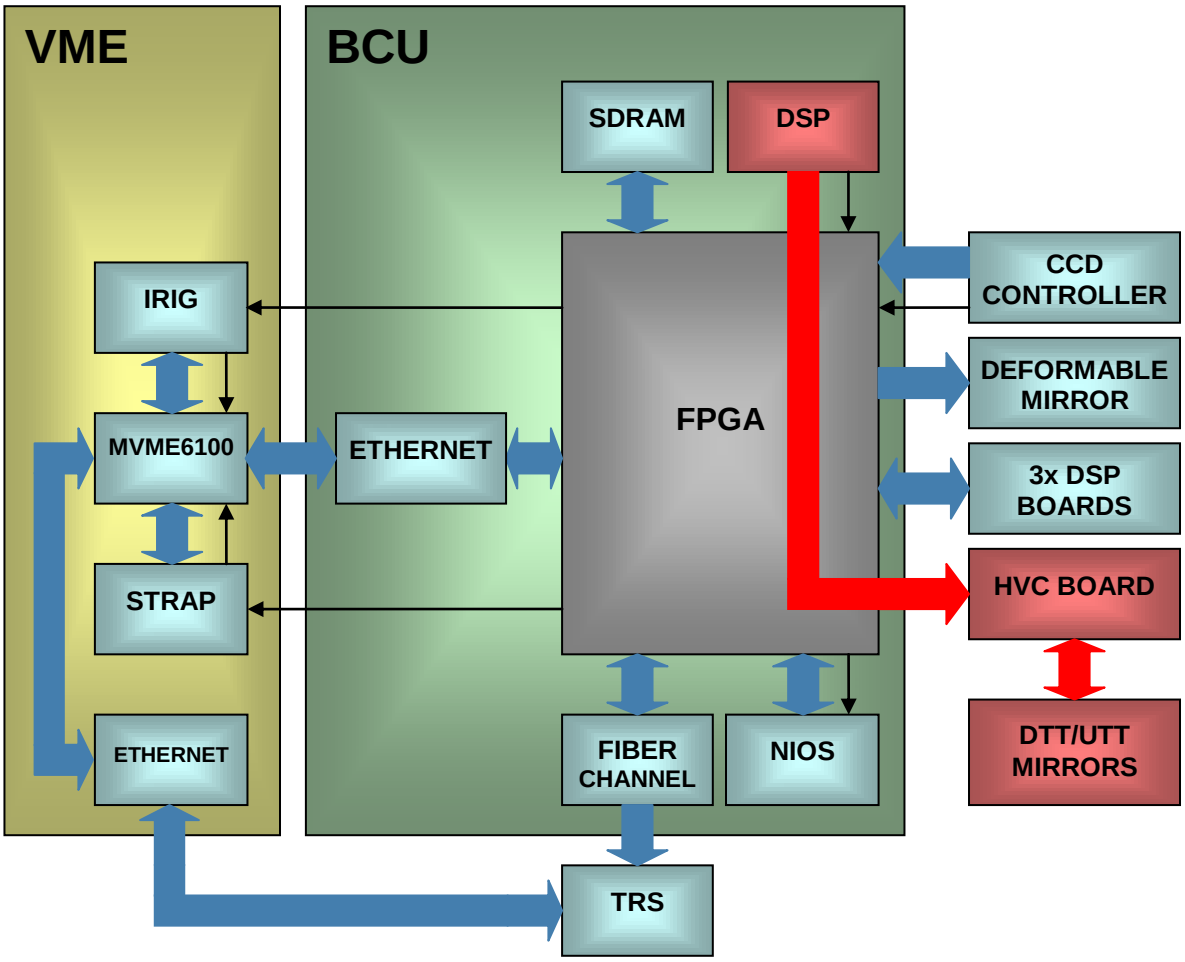
STEP 3:



The DSP of BCU board checks until the new residual wavefront vector is ready; then it reads the new data from the DSPs of the DSP boards.

This step considers just the read back of the new residual wavefront vector and takes 8 µs (T7 in the Figure 34).

STEP 4:

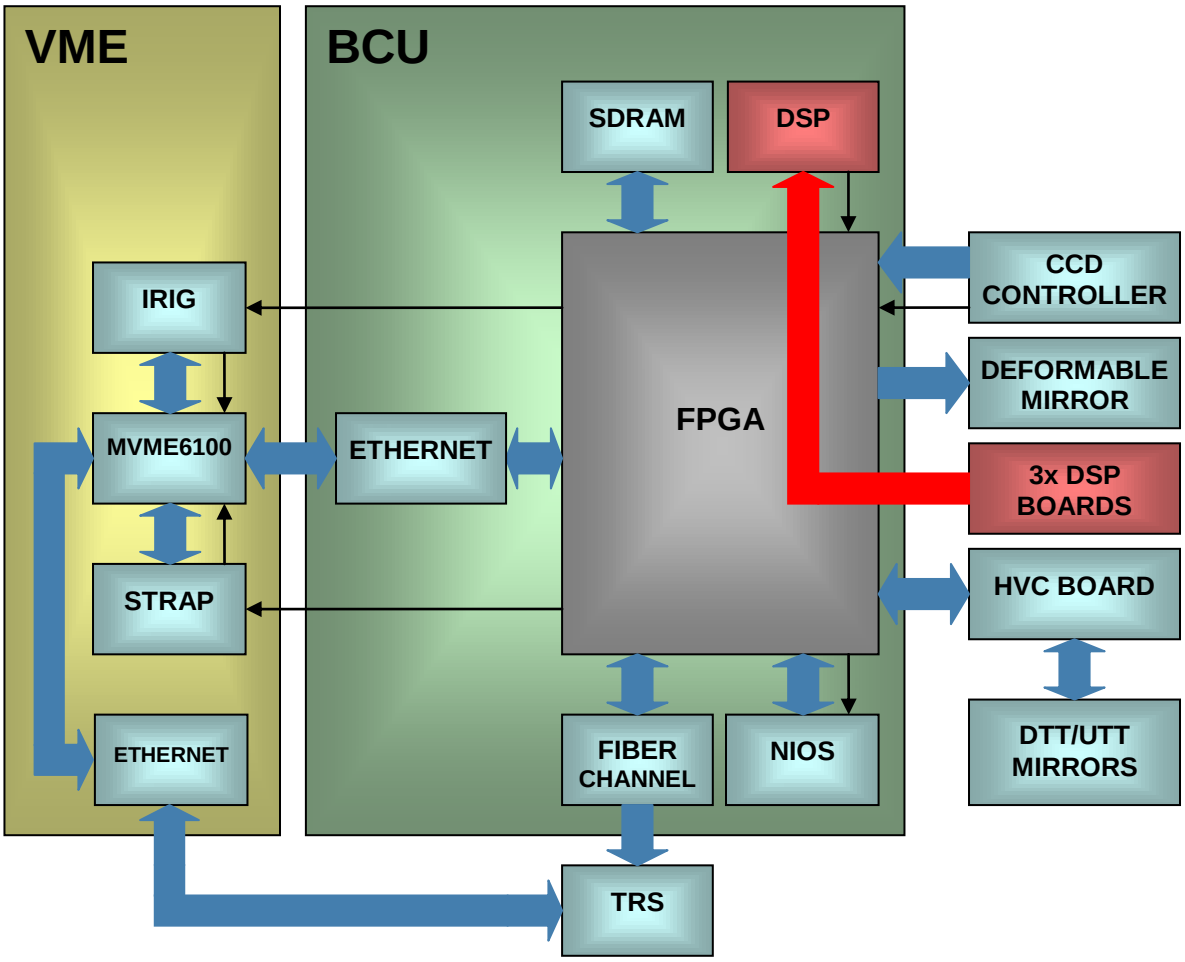


Depending on the system configuration the tip-tilt elements of the residual wavefront vector are sent to the DSP of HVC board and used to update the DTT or UTT mirror position. Before to be applied the new data is converted (gain, rotation or splitting) by the DSP of HVC board and then it is passed to the CLMP loop which controls the DTT and UTT actuators output. Then the code organizes the telemetry data to be read by the DSP of the BCU board.

The timing of this step is the follow:

Mirror	data transfer (µs)	commands computation (µs)
DTT	1.5 µs	3 µs
UTT	1.5 µs	7 µs
	T8 in the Figure 34	T9 in the Figure 34

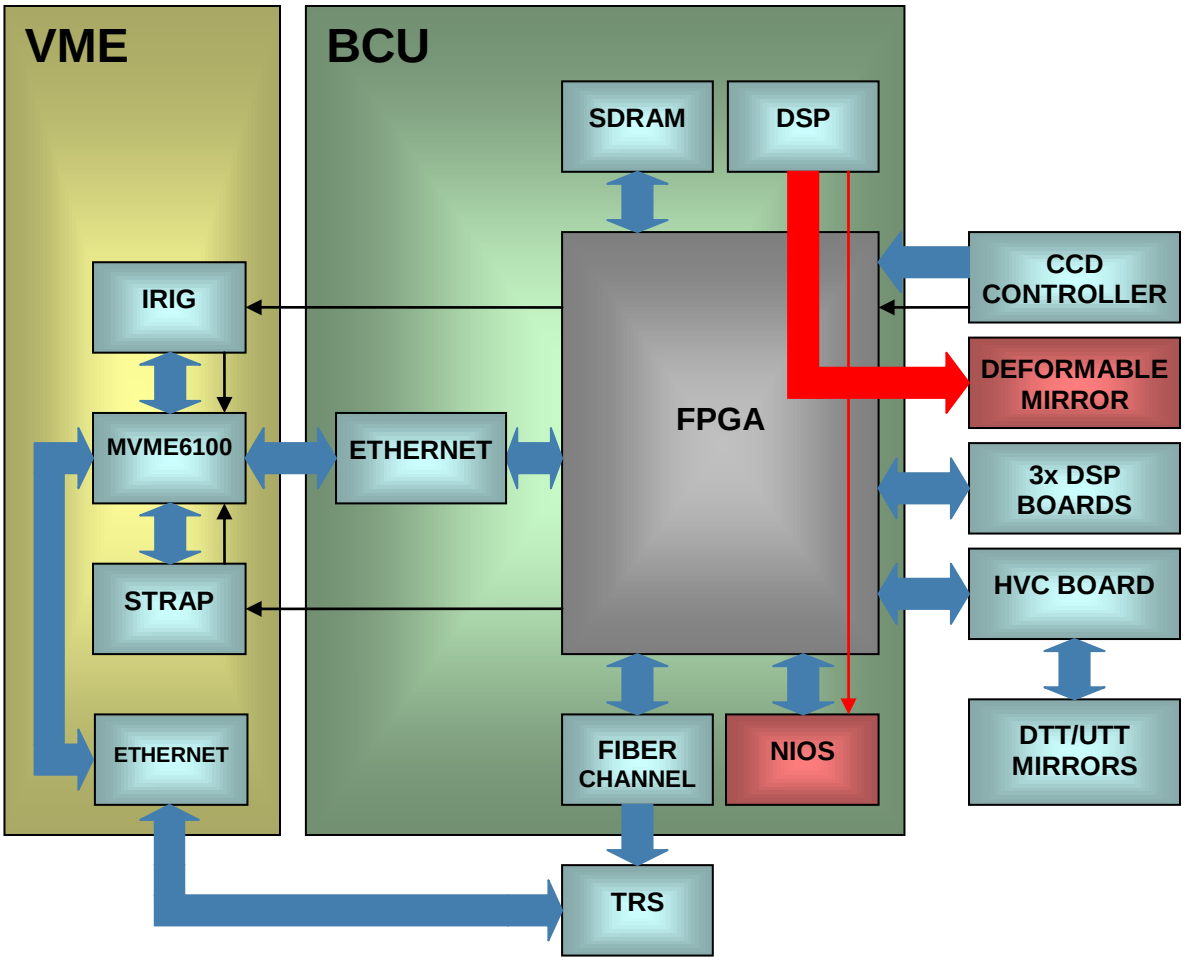
STEP 5:



In this step the DSP of the BCU board reads the new DM commands vector from the DSPs of the DSP boards.

This step takes 8 µs (T10 in the Figure 34).

STEP 6:



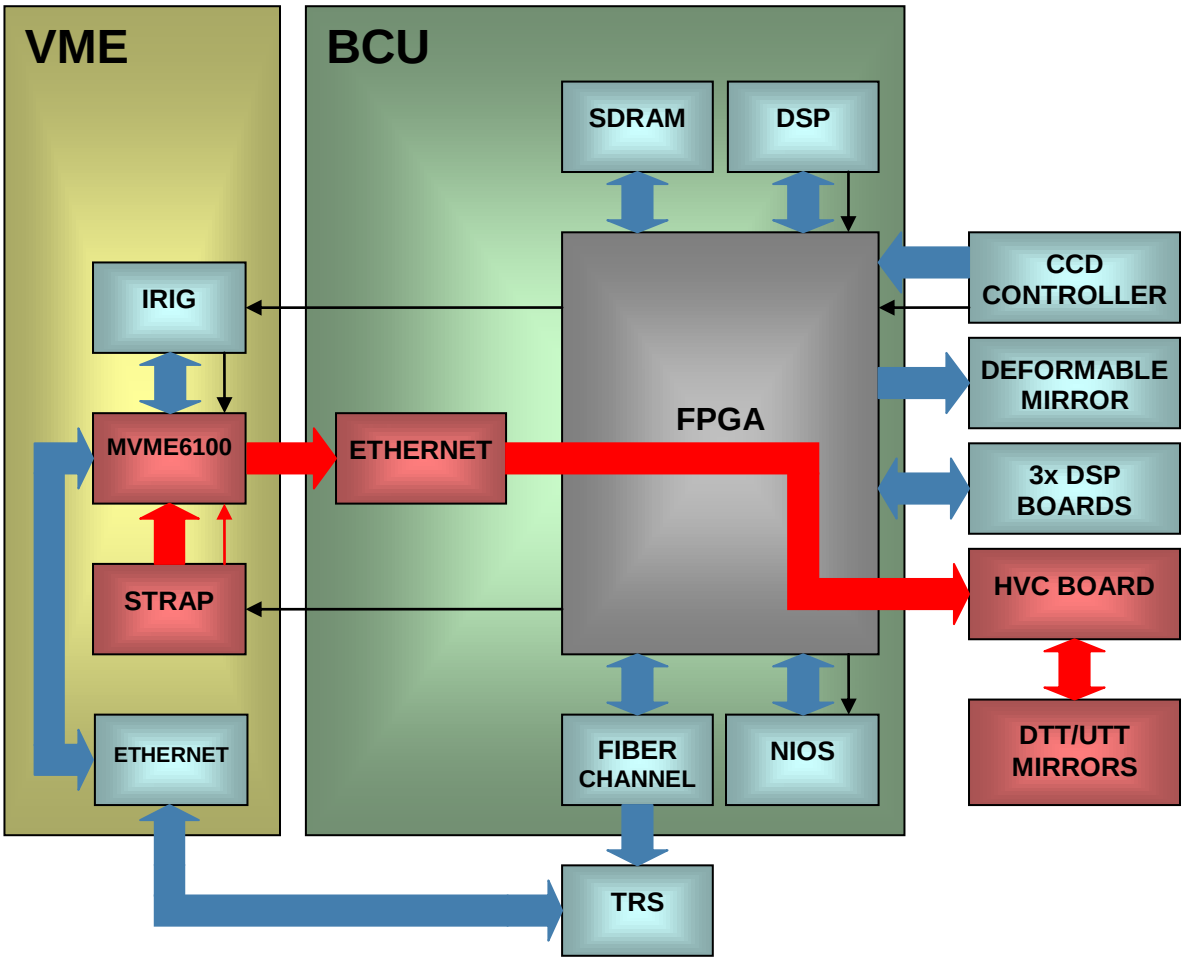
Once read, the command vector is converted and reorganized before to be sent to the deformable mirror. When the vector is ready the DSP generates an interrupt to the NIOS controller and at this point the DSP can continue with the real time process.

The interrupt routine initializes a DMA transaction between the DSP memory and the deformable mirror logic interface. This interface is able to store all the commands in a local FIFO and then they are sent slower according to the Xinetics high voltage controller interface, in this way the NIOS controller can continue with the main functions and do not have to wait the end of the deformable mirror update.

The following table shows the main timing of this step

Action	**Sub-step (µs)**		**Total time (µs)**	**Note**
Conversion and reorganization	34	T11 in the Figure 34	34	
interrupt generation	0		34	at this point the DSP can continue with the real time process
NIOS interrupt routine (DMA initialization and interrupt line release)	12	T12 in the Figure 34	46	at this point the NIOS controller code continue with the main functions
end of DMA transaction	3		49	
end of commands update to DM	36		85	

STEP 7:



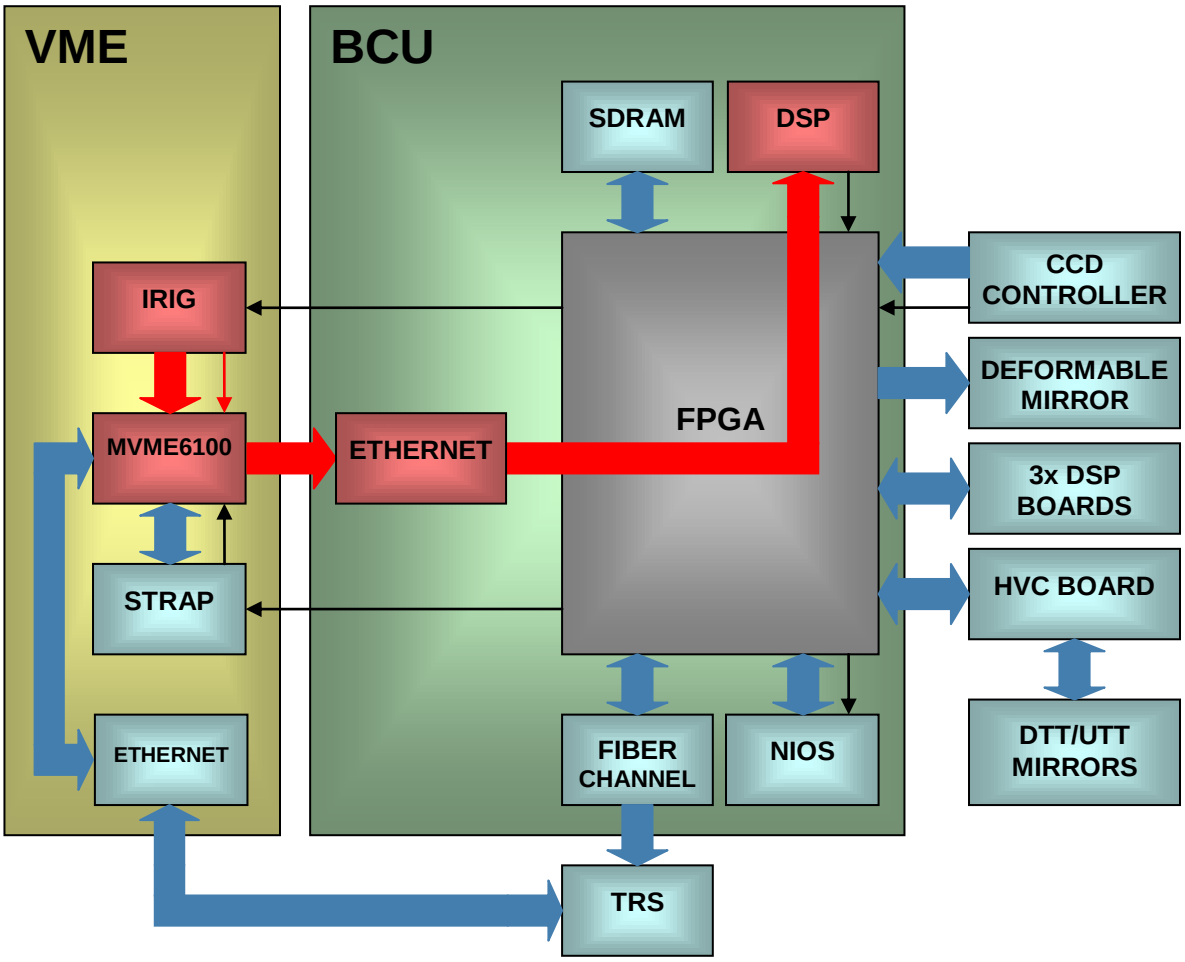
If enabled, when the STRAP subsystem receives the external interrupt notification from the CCD controller, it performs the quad-cell acquisition and generates a VME interrupt to the MVME6100 board. This board reads the new quad cell counts and writes them to the DSP of HVC board. The HVC board computes the new centroids and related DTT mirror commands which are passed to the CLMP loop.

This step ends the real time critical part. The rest of operations prepare and save the telemetry data, they are not real time critical but they should be executed before the beginning of the following frame.

From the timing point of view this step is quasi asynchronous respect to the previous steps, in fact this step starts with the start of frame signal and uses different resources respect to the previous steps.

All this phase could have a small jitter because the MVME6100 to DSP of HVC board data transfer is performed by the Ethernet link and a small jitter of ~50 µs has been measured. For this reason this step takes 150 to 200 µs (T13 + T14 in the Figure 34).

STEP 8:



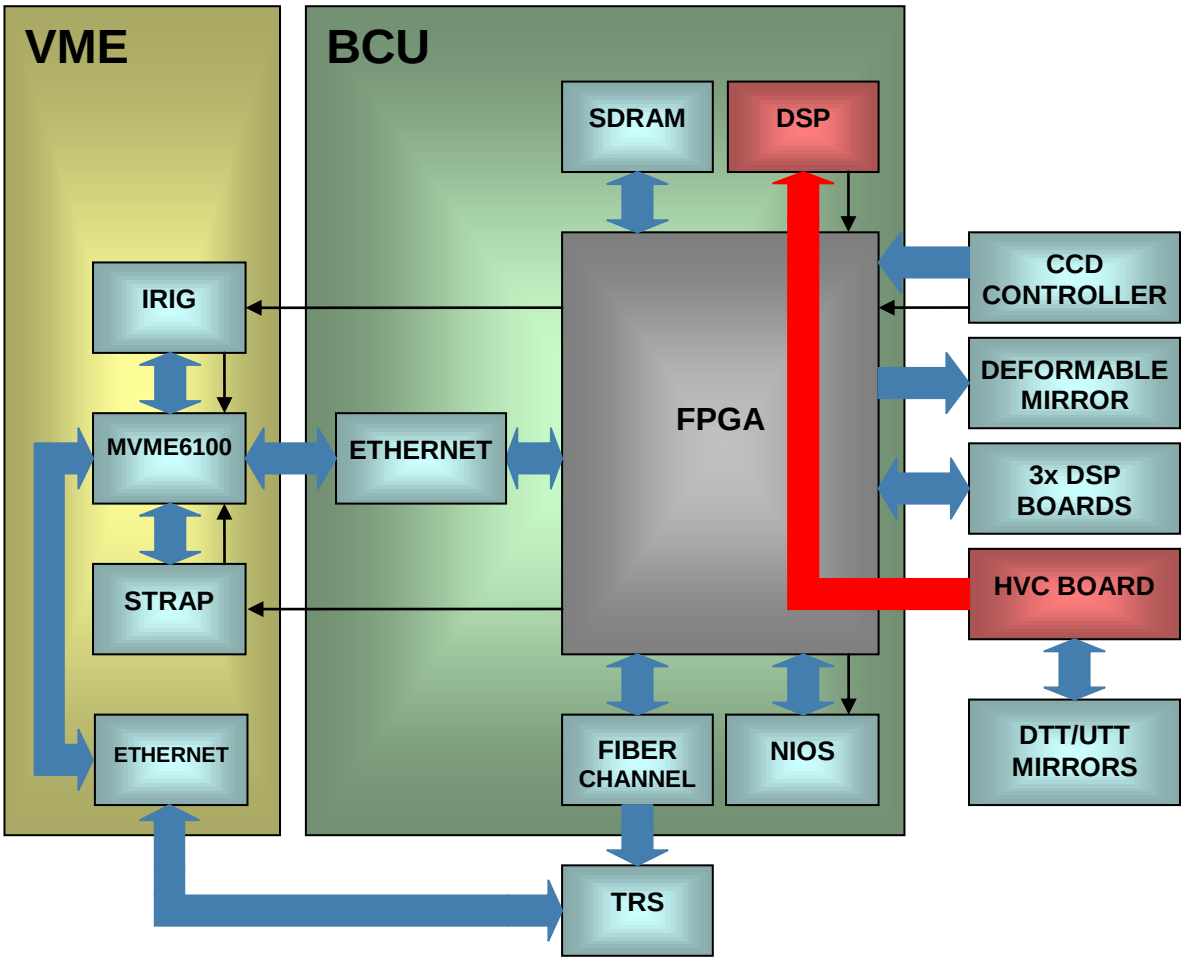
In parallel to the steps 2 to 5 when the IRIG board receives the external interrupt (step 1), it latches the absolute time and generates a VME interrupt to MVME6100 board. This board reads the time stamp and writes the real time packet to the DSP of BCU board. This packet includes:

- time stamp (64 bits);
- real time flags (32 bits);

Similarly to the previous step, this step could have a small jitter (~ 50 µs) because the data transfer between the MVME6100 and the DSP of BCU board is performed by the Ethernet link. In this case the timing is not critical because it is sufficient that new real time packet arrives to the MGAOS system before the step 9 begins. The WIF has been projected to send the two commands (STRAP and real time packet) always in this order because the STRAP packet is more time critical (in order to reduce the DTT mirror update phase lag) respect to the real time packet.

This step takes 190 to 240 µs (T15 in the Figure 34).

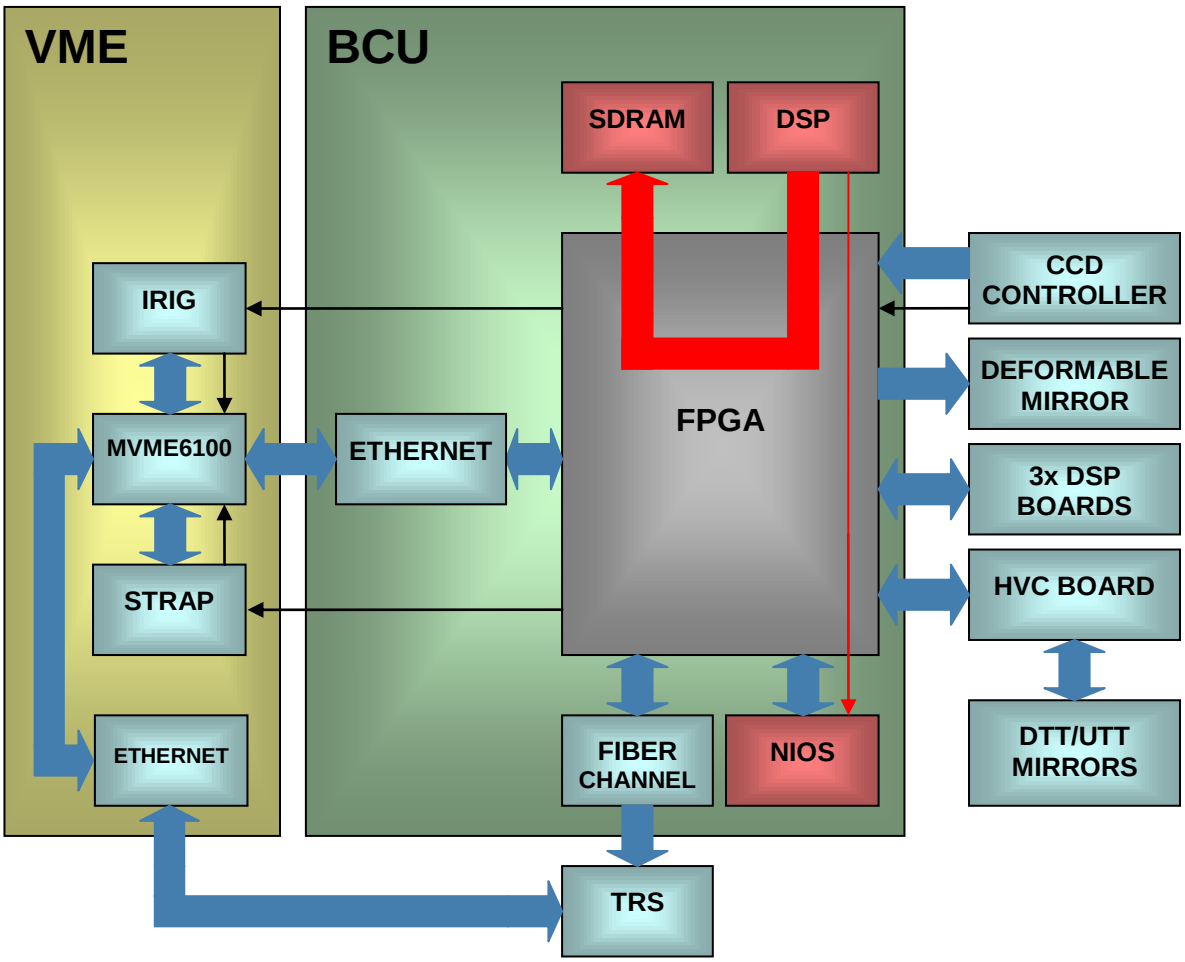
STEP 9:



The DSP of BCU board waits until the DSP of the HVC board has finished the step 4 and has prepared the telemetry data, then it reads the new data. The step 5 takes more time of step 4 so this step starts at the end of step 5.

This step takes a fixed time of 4 µs (T16 in the Figure 34).

STEP 10:



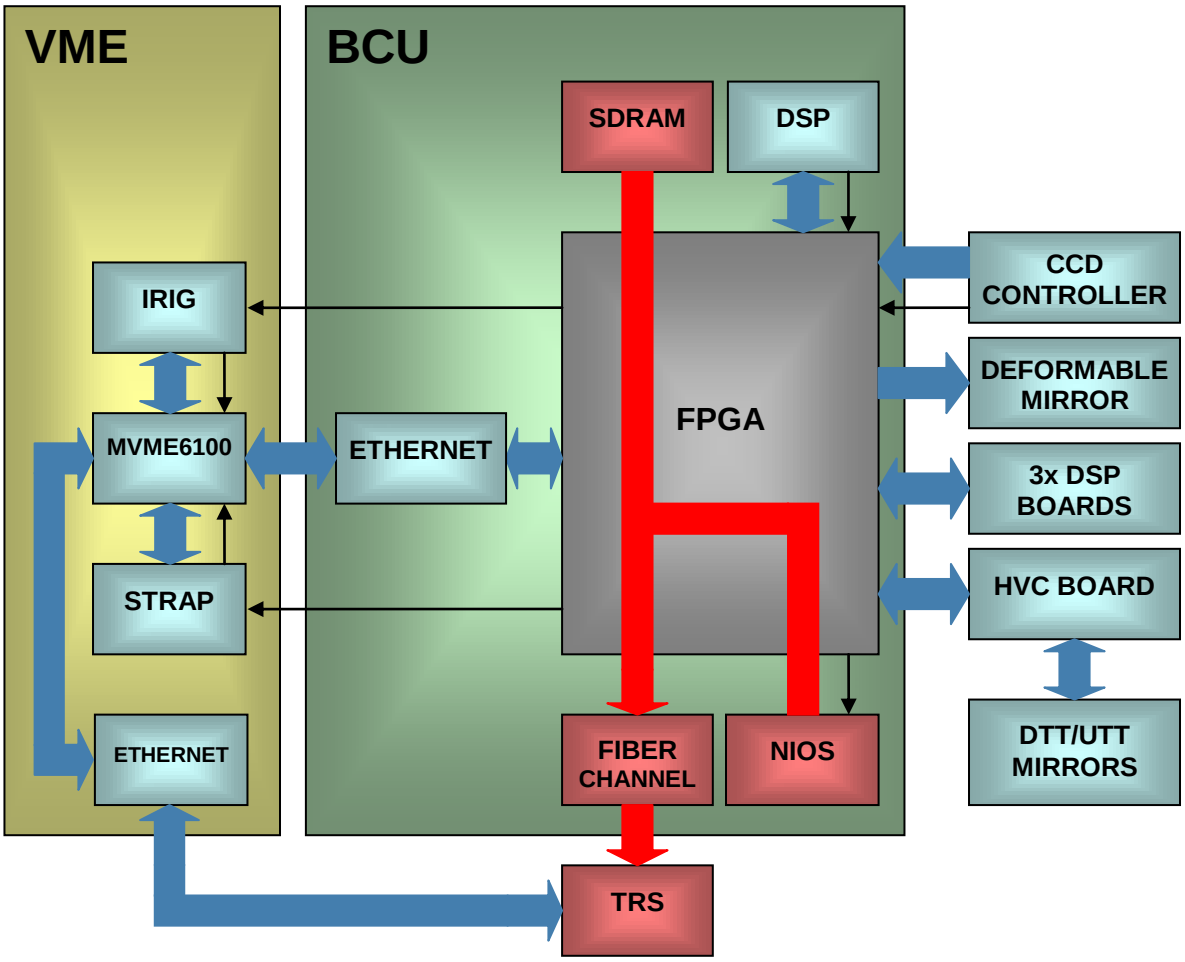
When the steps 8 and 9 are completed the DSP of BCU board is able to prepare the telemetry data records (DFB and FFB). Then it generates an interrupt to the NIOS controller. The NIOS interrupt routine initializes a DMA between the DSP memory and SDRAM in order to save the new telemetry records. The FFB record is stored at every CCD frame instead the DFB record is stored at 100Hz.

The SDRAM is used as a circular buffer to compensate the jitter between the uploaded data and the downloaded data to the TRS throw the Fiber Channel link.

The following table shows the main timing of this step

Action	Sub-step (µs)		Total time (µs)	Note
Telemetry records organization	62	T17 in the Figure 34	62	
interrupt generation	0		62	at this point the DSP has finished the real time loop and it is ready for a new one
NIOS interrupt routine (DMA initialization and interrupt line release)	15	T18 in the Figure 34	77	at this point the NIOS controller code continue with the main functions
end of DMA transaction (DFB only)	29		106	
end of DMA transaction (DFB & FFB)	83		160	

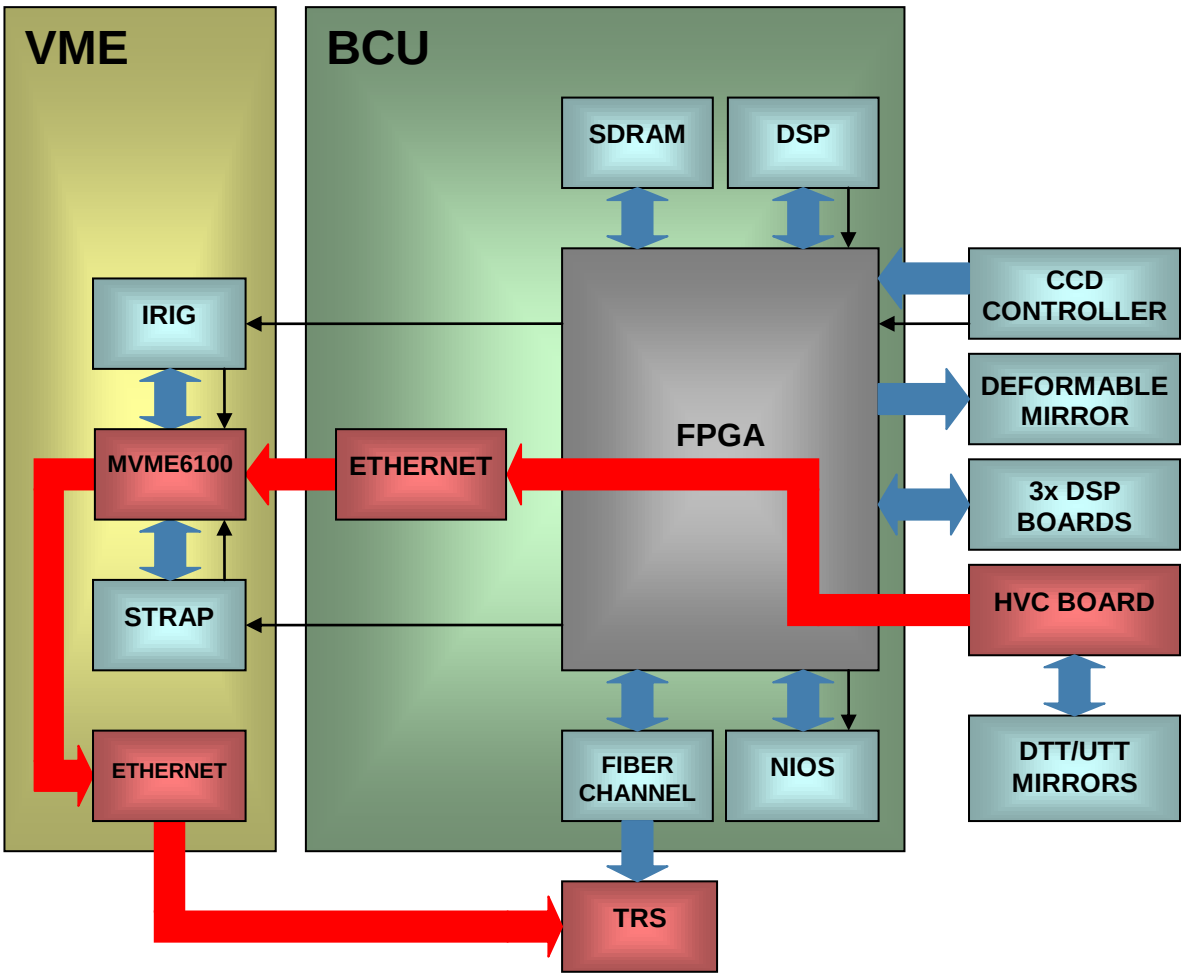
STEP 11:



The NIOS controller checks continuously if new telemetry data is available in the SDRAM buffer. When new data arrives a sequence of UDP/IP over Fiber Channel packets are prepared by the NIOS and sent to the TRS.

This step is managed by the NIOS controller and it is performed at lower priority respect to the real time part so it could be interrupted by the various higher priority tasks running in the NIOS controller. For this reason it is better to indicate the maximum sustained bandwidth of this link, which is 31 MB/s (T19 in the Figure 34), higher than the minimum requested bandwidth as indicated in Table 37.

STEP 12:



When STRAP is used the DTT telemetry data is temporary stored in the DSP memory of HVC board. Then the MVME6100 board reads this data by the Ethernet link and redirects it to the TRS through the public Ethernet link.

In order to optimize the Ethernet bandwidth the DTT telemetry data is accumulated to the DSP of HVC memory with the same concept of the circular buffer of the SDRAM and read by the MVME6100 only when a sufficient quantity of data is available. Also in this case it is better to indicate the maximum sustained bandwidth of this link, which is 7 MB/sec (T20 in the Figure 34), higher than the minimum requested bandwidth as indicated in Table 38.

.

7.1.1.2 NGWFC timing diagram

Following the real time flow and timing described in 7.1.1.1, the Figure 34 shows the connection and the dependency between the various steps.

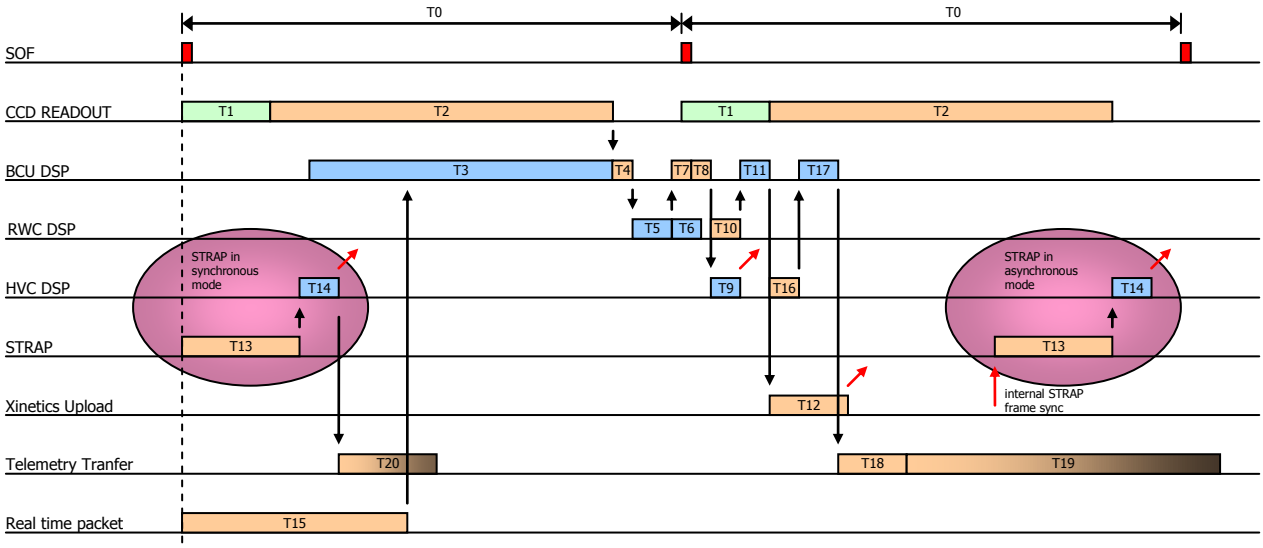


Figure 34 - Real time sequence

Step	Time	Note
	T0	The frame rate depends on the CCD type and configuration
	T1	The pre-readout pixels time depends on the CCD type and configuration
	T2	The readout pixels time depends on the CCD type and configuration
Step #1	T3	See step 1 description
Step #2	T4	This phase refers only to the centroid vector download
	T5	This phase refers to the residual wavefront vector and the last tap servo loop computation
	T6	This phase refers to the servo loop pre-computation for the next step
Step #3	T7	See step 3 description
Step #4	T8	This phase refers to the tip-tilt elements of the residual wavefront vector download to the HVC board
	T9	This phase refers to the tip-tilt mirror commands computation (DTT or UTT depending on the system configuration)
Step #5	T10	See step 5 description
Step #6	T11	This phase refers to the DM mirror commands conversion and reorganization
Step #6	T12	This phase refers to all the NIOS controller and Xinetics interface actions
Step #7	T13	This phase refers to the STRAP APD counts acquisition, MVME6100 real time reading of the new data via VME bus and sending of data to the HVC board via Ethernet real time packet
	T14	This phase refers to the tip-tilt centroids and DTT mirror commands computation and the update of the CLMP loop
Step #8	T15	See step 8 description
Step #9	T16	See step 9 description
Step #10	T17	This phase refers to the telemetry records preparation and NIOS interrupt generation
	T18	This phase refers to the NIOS controller interrupt service routine and DMA data transfer
Step #11	T19	See step 11 description
Step #12	T20	See step 12 description

Table 28 - Timing description

7.1.2 MGOAS DSP codes description

Regarding the real time computation it is completely performed by the DSPs of BCU, DSP and HVC board.

In particular:

- the DSP of BCU board implements the centroid computation and the telemetry record organization and manages the real time data flux between the MGAOS boards. The code running in this DPS is called CentroidCalculator
- the DSPs of DSP boards implement the parallel computation of residual wavefront vector and command servo loop filter. The code running in these DPSs is called ResidualWavefrontCalculator
- the DSP of HVC board implements the computation of APD centroid and tip-tilt mirror commands of DTT and UTT mirrors. This board also controls in open or close loop the high voltage drives for DTT and UTT mirror actuators. The code running in this DPS is called HVCController

7.1.2.1 CentroidCalculator code description

The following flow chart describes the flux of the CentroidCalculator code.

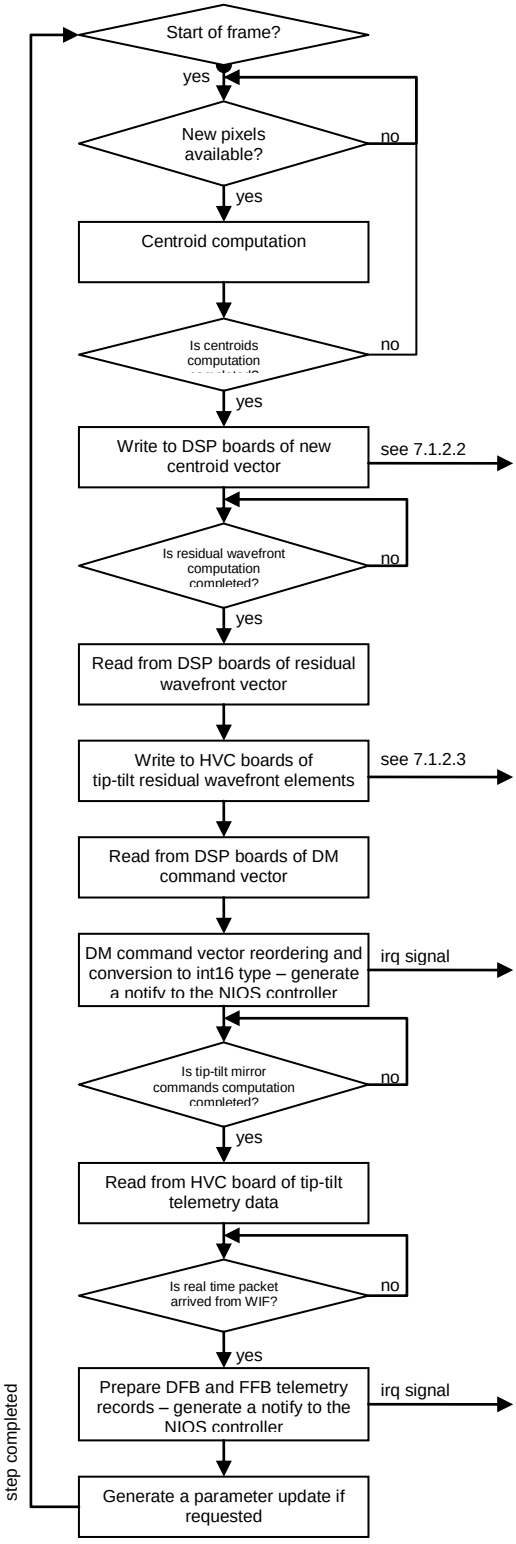


Figure 35 - flow chart of DSP code of BCU board

7.1.2.2 ResidualWavefrontCalculator code description

The following flow chart describes the flux of the ResidualWavefrontCalculator code.

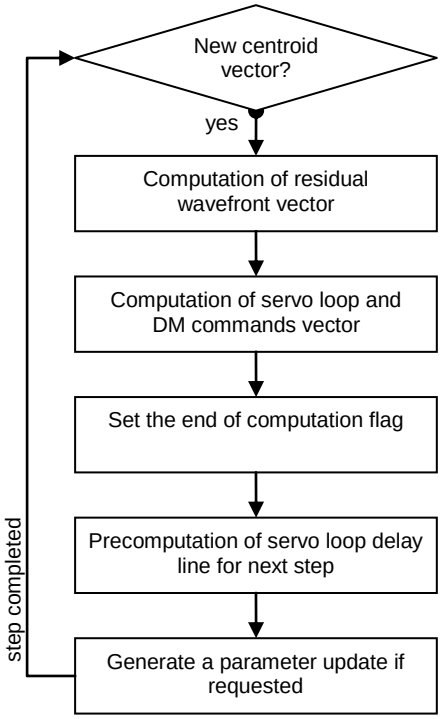


Figure 36 - flow chart of DSP code of DSP boards

7.1.2.3 HVCController code description

The following flow charts describe the flux of the HVCController code. For clarity the flow charts has been divided in four charts:

- Main scheme
- APD centroid computation
- DTT mirror commands computation
- UTT mirror commands computation
- Local actuator servo control loop (CLMP loop)

Main scheme

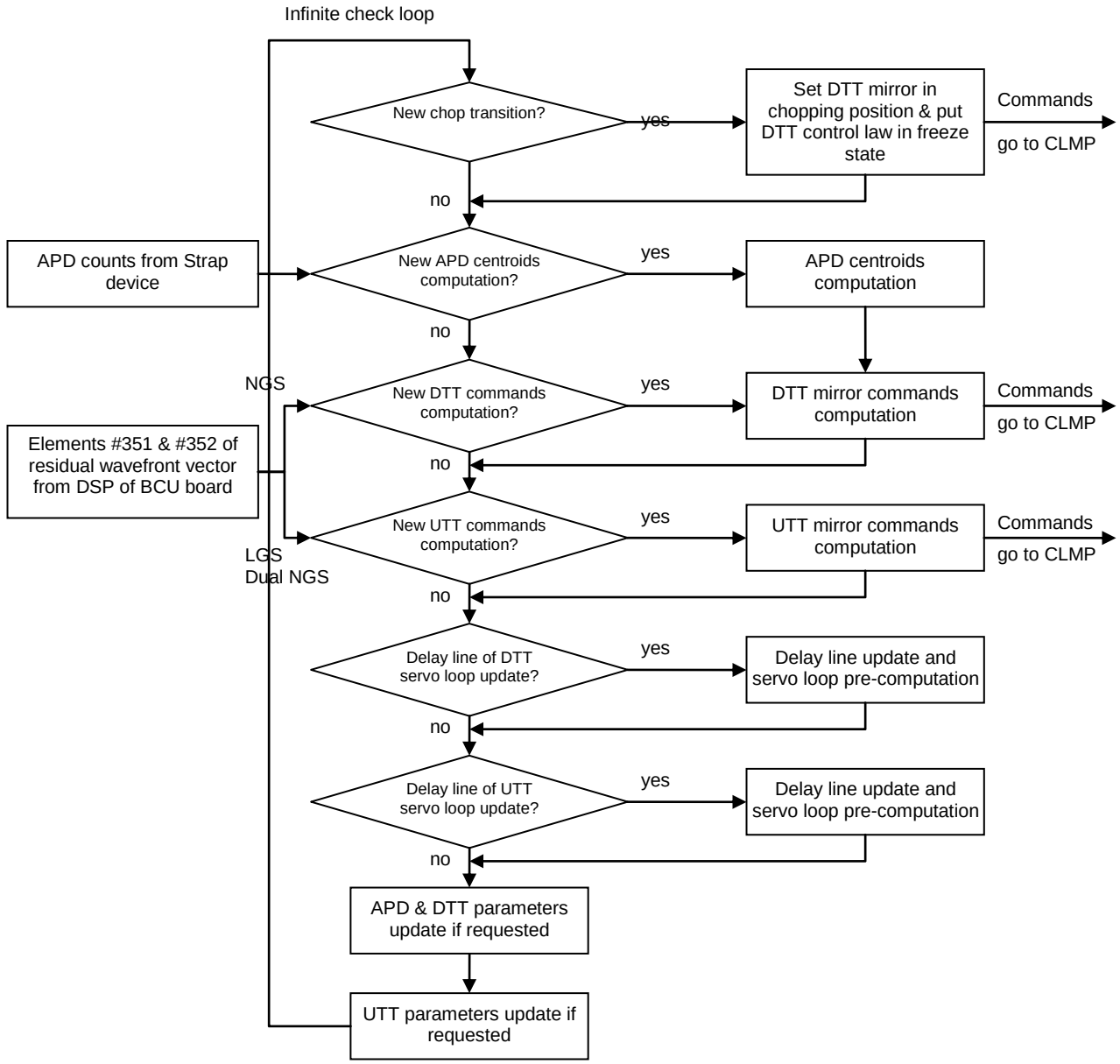


Figure 37 - main scheme of DSP code of HVC board

APD centroids computation

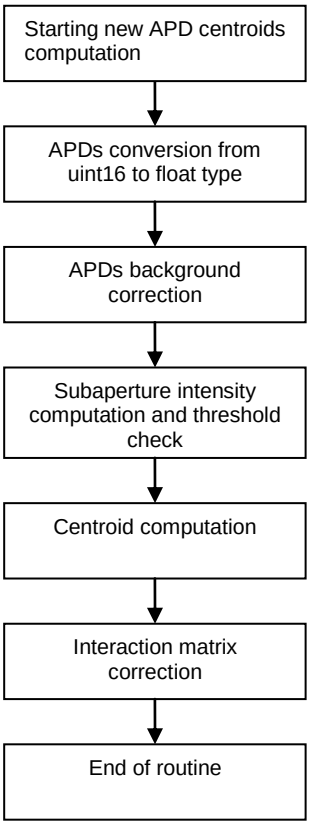


Figure 38 - flow chart of APD centroids computation

DTT mirror commands computation

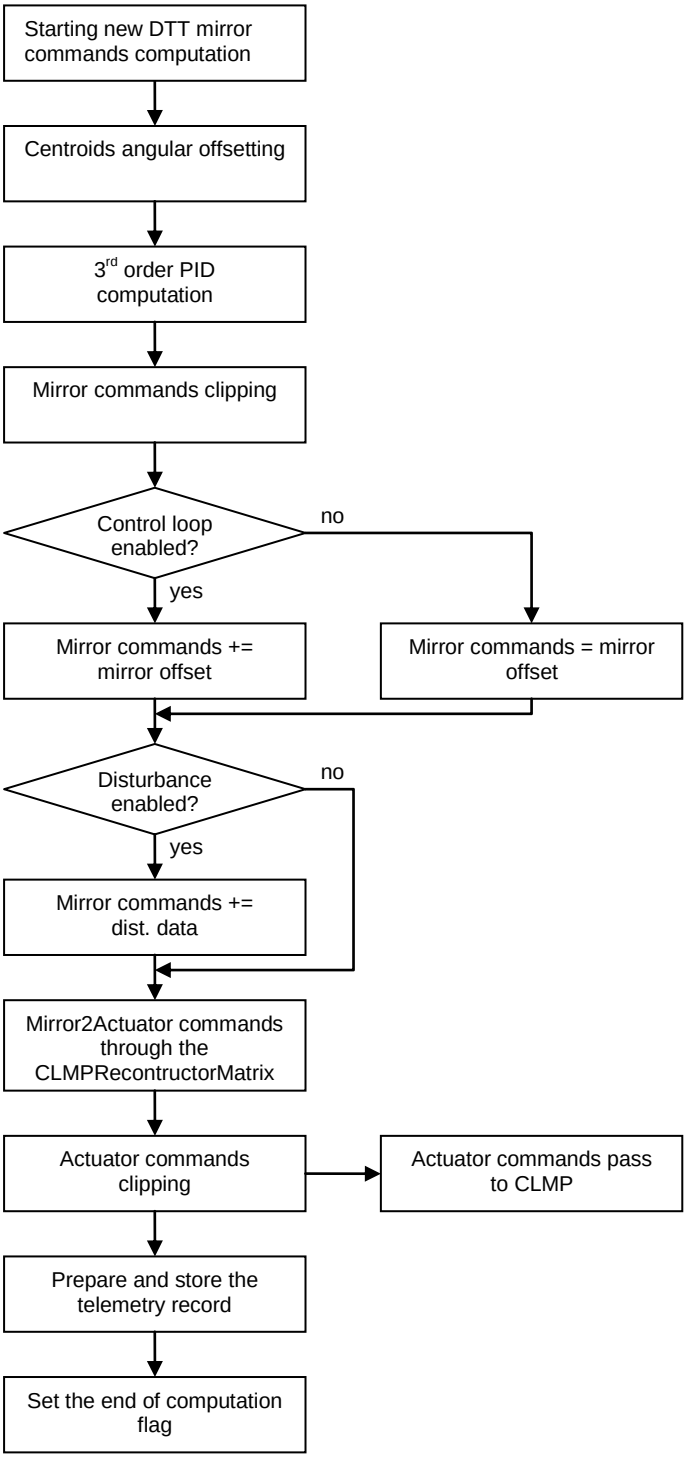


Figure 39 - flow chart of DTT mirror commands computation

UTT mirror commands computation

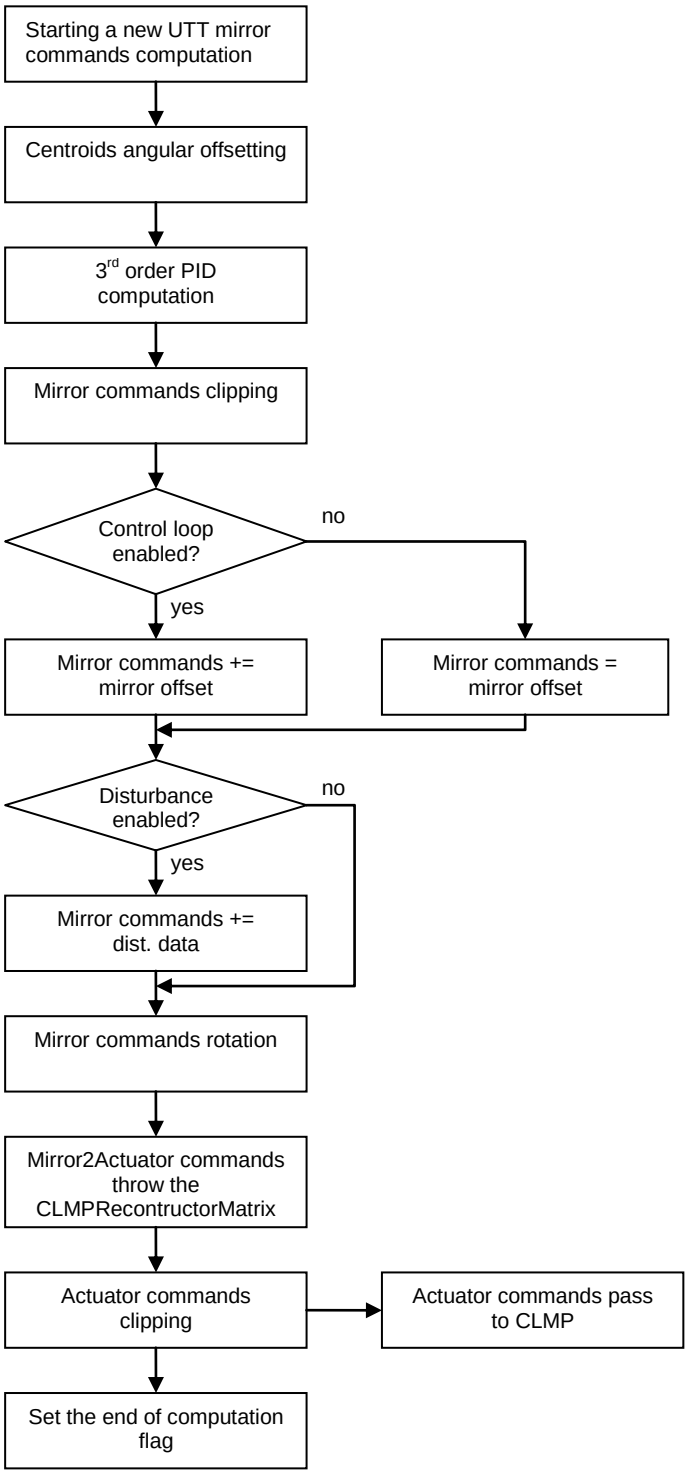


Figure 40 - flow chart of UTT mirror commands computation

Local actuator servo control loop

The local actuator servo control loop is an interrupt service routine @ 70 KHz and its scope is to control the actuators of the tip-tilt mirrors in position.

The routine implements a digital filter with 3 taps IIR filter on the position error and 3 taps IIR filter on the absolute position. This is typically used to implement a derivative control acting as 'electronic damper'. The filter is completely independent for each mirror actuator.

The DTT mirror uses three channels, while the UTT mirror makes use of two channels and a third channel is driven at constant voltage, which definable through WIF. During operation this voltage is typically set at mid output range.

The output is converted to voltage and sent to the HV driver and then to the corresponding tip-tilt mirror actuator.

The feedback actuator position is acquired by the strain gauge available on the tip-tilt mirror.

Input and output gains and offsets on each actuator allow considering strain gauge and actuator calibration, so that the CLMP control filter has unity gain.

Tuning of servo-loop is possible by means of automated tuning scripts, based on Matlab routines. A direct communication interface between Matlab and MGAOS is available. Matlab scripts command the HVC board to perform dynamic response tests. To do this, time-sequences can be uploaded to and downloaded from the HVC board. The sequences are executed and acquired at the same speed of the local servo-loop (~70 KHz). The tuning algorithm is based on function-cost minimization algorithms, as explained in 6.1.1.3.4.

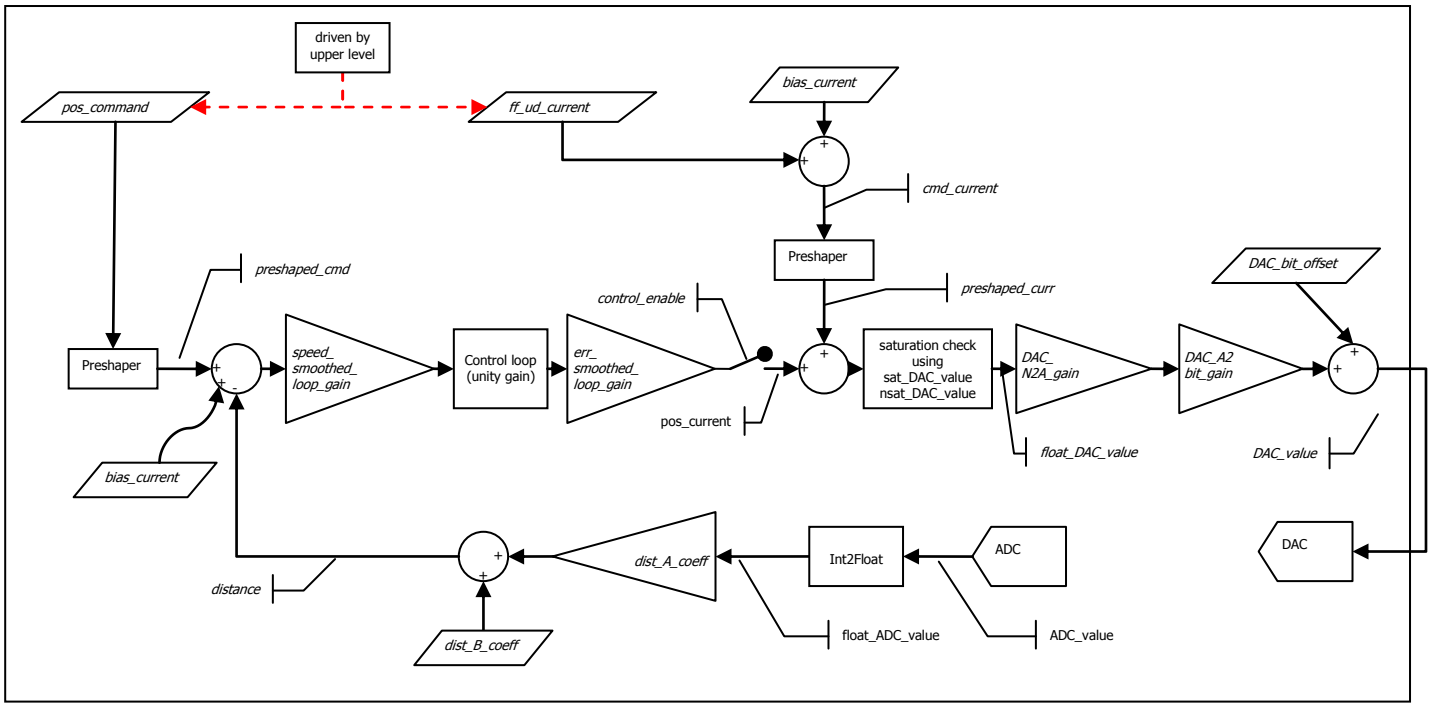


Figure 41 - CLMP loop flow chart

7.1.2.4 DTT/UTT power up and shutdown

In order to avoid excessive mechanical stress to the mirrors and/or undefined output states during mirror power up, 'smooth' power up and shutdown routines for both DTT and UTT (wifStartD/UTT and wifStopD/UTT) are available.

The power-up procedure is described hereafter:

- all HV relays open
- switch on HV supply
- command all HV outputs to 0V
- close (now safely) HV output relays
- check actual outputs with embedded diagnostics
- rise smoothly (under SW control) the voltages. In particular, the UTT mirror will be driven so that the common voltage is always kept at 2x command voltages while the high voltage is rising. This will bring smoothly the mirror to mid range position.

Once the mirror is powered up the control loop can be closed with the SET_D/UTT_CLMP_LOOP WIF command

Before calling the shutdown procedure the CLMP cotnrol loop must be opened then the wifStopD/UTT routine will do the following steps:

- decrease smoothly (under SW control) the voltages. In particular, the UTT mirror will be driven so that the common voltage is always kept at 2x command voltages while the high voltage is going to zero. This will bring smoothly the mirror to off position
- check actual outputs with embedded diagnostics (shall be all zero)
- open HV relays
- switch off HV supply

The wifStartD/UTT and wifStopD/UTT shall always be called when the system is in standby

7.1.2.5 Chopping

The NGWFC ha also chopping capabilities with the following characteristics:

- Chopping is performed by the DTT mirror.
- Maximum chop amplitude: ±800 mas, corresponding to the full throw of the mirror.
- Typical chop amplitude ±300 mas.
- Maximum chop frequency: 10 Hz. This is the number of times per second the chopper is in the on-star phase.
- Chop modes:
    a. **single-sided**. The mirror chops away from the star and then chops on-star again.
    b. **2-sided (symmetric)**. The mirror chops away from the star in one direction, then chops on-star again, then chops away in the opposite direction, then chops on-star again and the cycle is repeated.
    c. **4-sided (X-Y)**. The 4 sided is identical to the double sided chop shown above except that after completing one entire cycle, the chop angle is rotated by 90 degrees so the next double

sided chop cycle is orthogonal to the previous. After chopping at 90 degrees, the next chop cycle is back at the original angle.

- Synchronization: The chop phases is synchronized with the IRIG time so that the interferometer instruments will know precisely when to look for the beam on their detectors.
- Chop angle: The chopper can chop at any angle within the motion limits in X-Y space, as determined by the user.

The adaptive optics control loop remains active during the 'chop-away' and 'on-star' phase. but during the transitions the system will 'freeze' it. This means that the input to the digital filter is not updated (even if the WFS continues running) and similarly the outputs to the mirrors (DM/UTT/DTT) is not influenced by the adaptive optics loop. In this way, when DTT reaches a stable position (either 'chop-away' or 'on-star'), the control can restart the operation. Restarting of the normal adaptive optics operation can occur with an user-definable time delay w.r.t. the chopping transition. This allows to take into account the cycles required by the WFS in order to generate 'correct' slopes (should be typically 2). The unfreezeDelay does not comprehend the mirror settling time, the WIF and MGAOS SW will take into account this delay automatically.

Once the 'chop-away' position is reached, the loop can be closed but it is mandatory to correct the RTR inputs with a 'centroid offset', otherwise the control loop would tend to restore the 'on-star' position. This centroid offset acts as the 'centroid origin', but is implemented in a totally independent way in order to keep the current command set unaltered.

On Table 29 we report a list of the WIF commands required by the chopping functionality.

COMMAND	PARAMETERS	DESCRIPTION
SET_CHOP_PARAMETERS	uint32 chopmode; (0,1 or 2) uint32 chopFreq; uint32 unfreezeDelay; double gain[2]	Sets the chopping parameters: - chopmode, 0 = single-sided, 1 = 2-sided, 2 = 4 sided; - chopfreq is the chopping frequency in Hz; - unfreezeDelay is the time after DTT is settle to wait before enabling input (expressed in number of frames); - gain converts the centroid offset to mirror commands;
GET_CHOP_PARAMETERS	same parameters as the SET command	
SET_CHOPPING	uint32 chopEnable (0 or 1)	Enable (1) / disables (0) chopping
GET_CHOPPING	same parameters as the SET command	
SET_CHOP_CENTROID_OFFSET	single chopCentrOffset[2]	Sets the chopping centroid offset X and Y. This is done with a separated command because this will be send in near real time
GET_CHOP_CENTROID_OFFSET	same parameters as the SET command	

Table 29 – WIF commands required for chopping operation.

The chop state machine is built into the WIF on the MVME. The IRIG board internal counter will be used as time reference. The various steps of the state machine is illustrated in Figure 42.

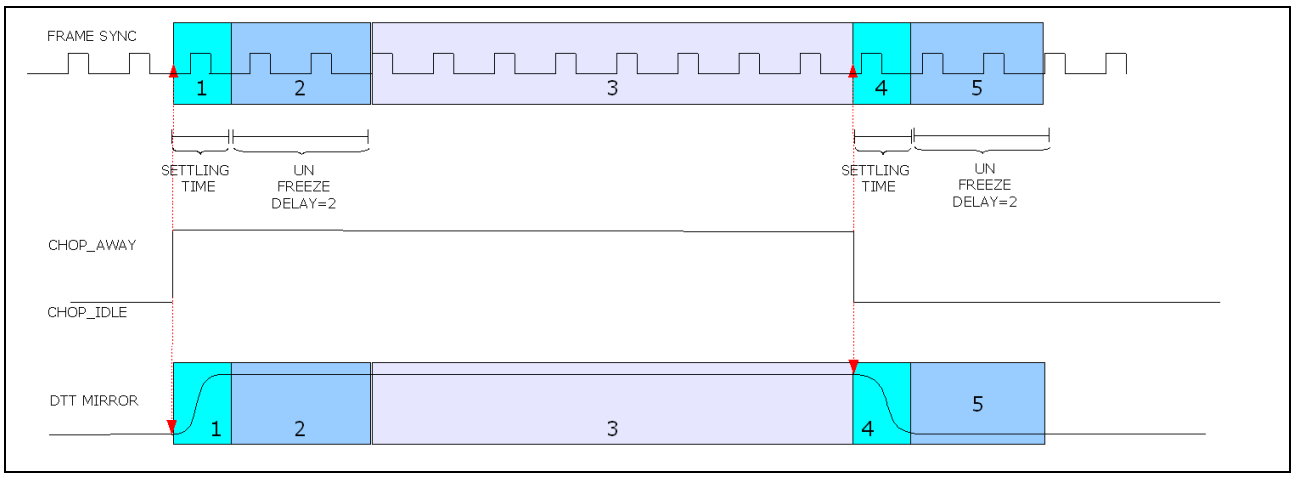


Figure 42 – Chopping state machine

Steps for the CHOP-IDLE -> CHOP-AWAY transition:

1. Motion phase
    a. The MVME will send a message to the HVC board containing the desired chop mirror and centroid offsets, the desired preshaper speed and a flag indicating the CHOP-AWAY state.
    b. In this situation the HVC will ignore position updates from the BCU and will command the DTT to the desired CHOP-AWAY position. This movement will be done in a save and clean way, using the preshaper set value, that will be to a slower settling time for chopping (long stroke movement). After motion completion the preshaper will be set back automatically to the original fast settling time, as required by field stabilization.
    c. The MVME will signalize to the MGAOS to freeze the DTT optical loop, while the DM optical loop will not be disabled at any time. This command is sent as an additional flag within the real-time communication between MVME and BCU.
    d. The MVME will wait for DTT mirror to settle. This time interval corresponds to the preshaper settling time, therefore the MVME can measure this time internally without implying any communication with MGAOS.
2. Frozen loop.
   When the mirror is in position it will wait for unfreezeDelay frames before signalizing to the MGOAS to unfreeze the optical loop (this is done using a flag in the timestamp real time packet).
3. Close loop operation during CHOP-AWAY phase.
   Now the DTT is in close loop in the CHOP-AWAY position, with a mirror chopping offset applied to DTT and a centroid chopping offset in the optical loop in order to annul the offset due to the chopping.

•

Steps for the CHOP-AWAY -> CHOP-IDLE transition:

4. Motion phase
    a. The MVME will send a message to the HVC to return to CHOP-IDLE state
    b. In this situation the HVC will ignore position updates from the BCU and will command the DTT to the original position. (mirror offsets = 0) This movement will be done in a safe and clean way, using the preshaper set value, which will be set to a slower settling time for chopping (long stroke movement).
    c. The MVME will signalize to the MGAOS to freeze again the DTT optical loop.

5. Frozen loop.
   The MVME will wait for DTT mirror to settle. When the mirror is in position it will wait for unfreezeDelay frames before signalizing to the MGOAS to unfreeze the optical loop (this is done using a flag in the timestamp real time packet).
6. Now the DTT is in close loop again in the CHOP-IDLE position, no mirror centroid offset and no centroid chopping offset.

7.1.2.6 MGAOS variables description and mapping

First of all a brief description of the internal memory organization of TigerShark DSP (TS101) used for this project should be explained. The memory of this DSP is organized into three banks of 64K dword each. The banks are not continuous so it is not possible to allocate a buffer between two banks. The following table summarizes the DSP's memory organization

Name	Start address (dword)	Size (dword)
Bank #0	0x00000000	0x00010000
Bank #1	0x00080000	0x00010000
Bank #2	0x00100000	0x00010000
Bank #3 (internal registers)	0x00180000	0x00000100

Table 30 - DSP's memory organization

As main rule, for this project, the DSP memory has been organized putting the DSP code and the single variables in the bank #0. The banks #1 and 2 has been used to store the vector parameters and input/output data vector. In this document the total variables memory mapping table is not included, the variables description and mapping is summarized in the "memory map Keck_NGWFC x_yy.xls" excel file where x_yy refers to the current release.

The following paragraphs report the total amount of memory used.

7.1.2.7 BCU DSP memory requirement

As indicated in [AD11], in this DSP runs the CentroidCalculator code.

This code uses the following quantity of DSP memory:

	Total memory (dword)	Allocated memory (dword)	Note
Bank #0	2970	16384	Code area
	28940	49152	Variables area
Bank #1	57944	61440	Variables area
	###	4096	Compiler stack (#1)
Bank #2	57944	61440	Variables area
	###	4096	Compiler stack (#2)

Table 31 – BCU DSP memory requirements

7.1.2.8 DSP, 3 boards memory requirement

As indicated in [AD11], in these DSPs run the ResidualWavefrontCalculator code.

For each DSP this code uses the following quantity of memory:

	Total memory (dword)	Allocated memory (dword)	Note
Bank #0	946	16384	Code area
	800	49152	Variables area
Bank #1	36900	61440	Variables area
	###	4096	Compiler stack (#1)
Bank #2	37512	61440	Variables area
	###	4096	Compiler stack (#2)

Table 32 – DSP Boards #0, #1 and #2 memory requirements

7.1.2.9 HVC board memory requirement

As indicated in [AD11], in this DSP runs the HVCController code.

This code uses the following quantity of DSP memory:

	Total memory (dword)	Allocated memory (dword)	Note
Bank #0	4322	16384	Code area
	21092	49152	Variables area
Bank #1	33184	61440	Variables area
	###	4096	Compiler stack (#1)
Bank #2	3936	61440	Variables area
	###	4096	Compiler stack (#2)

Table 33 – HVC DSP memory requirements

7.1.3 Centroids computation

Centroids computation is performed by the DSP of AdOpt BCU board. The pixels are read sequentially from the CCD and stored in local DSP RAM following a user defined look up table. This look up table depends on the selected wavefront operating mode. The goal of this look-up table is to group pixels which form one sub-aperture. Pixels are ordered to achieve high DSP performance in the next computation steps.

To explain the principle of operation, a possible organization of the user-definable LUT is given in Table 34.

The BCU FPGA reads the pixels from the SciMeasure Little Joe. While the pixels are acquired, they are automatically transferred to BCU DSP memory by means of a DMA process and stored into the appropriate memory location, optimized for guaranteeing the maximum DSP computational throughput. Every 8 pixels, the number of centroids that can be computed with the already acquired pixels is transferred to a fixed DSP memory location.

Gain and offset compensation and centroid computation are executed by a background routine. The number of *computable centroids* is checked vs. the number of already computed centroids, and, if the former is greater than the latter, the DSP computes a set of centroids and returns to the previous check.

This implementation allows obtaining a very efficient code execution on the DSP. Setting properly the ‘number of computable centroids’ in the LUT one can control pipelined operation. Moreover, by simply modifying the LUT, the system can be easily adapted to different wavefront sensor configurations.

Memory address of 1st acquired pixel

Memory address of 2nd acquired pixel
Memory address of 3rd acquired pixel
Memory address of 4th acquired pixel
Memory address of 5th acquired pixel
Memory address of 6th acquired pixel
Memory address of 7th acquired pixel
Memory address of 8th acquired pixel
Number of *computable centroids* at this point
Memory address of 9th acquired pixel
Memory address of 10th acquired pixel
Memory address of 11th acquired pixel
Memory address of 12th acquired pixel
Memory address of 13th acquired pixel
Memory address of 14th acquired pixel
Memory address of 15th acquired pixel
Memory address of 16th acquired pixel
Number of *computable centroids* at this point
…

Table 34 – Pixel LUT content

The centroids are transferred to the DSP boards for residual wavefront computation. Data transfer is sequenced by the BCU board and the data are transmitted through the real-time bus by means of DMA processes without DSP software overhead.

It is possible to select between normal operation with sub-aperture flux normalization of centroids and denominator-free centroiding.

7.1.4 Residual Wavefront computation

This is the dominant computation in the entire WFP. Residual wavefront is computed as:

$$\{W\}=[R]\{C\}$$

R is a 608 x 352 matrix. The upper operation takes about 214 floating point kMACs. To achieve better performance and speed, this work is parallelized among the three DSP boards for a total amount of 6 DSPs. Each one is responsible of the computation of a sixth of the W vector.

In order to maximize the computational throughput of the DSP core and to exploit its hardware architecture, the total number of MAC operations performed by each DSP should be a multiple of four. In our configuration we have 6 DSPs, since the requested vector size is 352, this number is rounded up to 360 elements, so that each DSP performs a 608 x 60 matrix multiplication.

The R matrix is divided in 6 sub matrices: R_1, R_2, R_3, R_4, R_5, R_6.

$\{R_1\}$ takes rows 1 to 60 of {R},

$\{R_2\}$ takes rows 61 to 120 of {R},

$\{R_3\}$ takes rows 121 to 180 of {R},

$\{R_4\}$ takes rows 181 to 240 of {R},

$\{R_5\}$ takes rows 241 to 300 of {R},

$\{R_6\}$ takes rows 301 to 352 of {R}, and the remaining rows are zeroed.

The 6 resulting operations are:

$\{W_1\}=[R_1]\{C\}$ assigned to DSP #1

$\{W_2\}=[R_2]\{C\}$ assigned to DSP #2

$\{W_3\}=[R_3]\{C\}$ assigned to DSP #3

$\{W_4\}=[R_5]\{C\}$ assigned to DSP #4

$\{W_5\}=[R_5]\{C\}$ assigned to DSP #5

$\{W_6\}=[R_6]\{C\}$ assigned to DSP #6

Each W_i is a 60 element vector, together they form the full 352 element W vector. From W_6 vector the last eight elements are simply ignored.

7.1.5 Control law servo computation

Like the residual wavefront also the control law computation is split among DSP boards 1, 2 and 3.

Each DSP is responsible of one sixth of the full command vector computation which is computed as:

$$C_i[n] = - b_1C_i[n-1] - b_2C_i[n-2] - b_3C_i[n-3] + a_0W_i[n] + a_1W_i[n-1] + a_2W_i[n-2] + a_3W_i[n-3]$$

where the input *W* is the residual wavefront vector and the output *C* is the output vector for the deformable mirror. The index *i* refers to each DSP (1 to 6). Also in this case the last eight elements of C_6 sub-vector are simply ignored. The *a* and *b* values are the control law coefficients. The indexes *n, n-1, n-2* and *n-3* refer the current value and the three previous values of each vector.

Actuator commands are clamped to a maximum admissible value, and the occurrence of threshold exceeding is recorded by incrementing a dedicated counter. If the control loop is closed the computed command is the new position that will be sent to the mirror otherwise, if the control loop is open, only the DM origin vector (nominal flat position) will be sent to the mirror.

Once the output commands for the DM are available, they are normalized, converted to the appropriate unsigned 16-bit DAC values and transferred to the BCU in a single read operation through the real-time backplane. Finally, from the BCU PIO port, the DM commands are transmitted to the Xinetics HVA. The BCU FPGA implements the real-time FPDP-like interface to the HVA. The RMS wavefront error is also computed by the DSP of BCU.

7.1.6 HVC real-time control software

The HVC board controls the DTT/UTT mirror outputs.

The use of this board depends on system setup according to the following table:

Mode	DTT Mirror	UTT Mirror
NGS	Used, driven by WFP	Not used
LGS	Used, driven by STRAP	Used, driven by WFP
Dual NGS	Used, driven by STRAP	Used, driven by WFP

Table 35 -Tip-tilt operating modes. The computation step refer to the algorithm description in §7.1.6.1.

The HVC software can be divided into two main tasks:

- tip-tilt mirror correction before applying the command. This task waits for a tip-tilt mirror command update, when it arrives it performs the two different pre-correction algorithms for the DTT and UTT mirrors as described later and sends the results to the local mirror servo control loop.
- local mirror servo control loop. This interrupt control routine is an high speed digital servo loop (running at ~70kHz) which controls the HV outputs in order to obtain the desired tip-tilt mirror movement requested from the first task. According to the high performances required by this routine, it has been decided to implement it using assembler code.

7.1.6.1 DTT mirror correction

The DTT mirror command source is indicated in Table 35. The first four steps of the algorithm are executed only if the input data arrives from STRAP, which sends the four APD counts. In NGS mode data arrive from the WFP (elements #349 and #350 of the residual wavefront vector); in this case the algorithm begins directly from the step #5.

Hereafter the sequence of computational steps:

Step #1: sky background correction:

- $\tilde{A}_i = A_i - S_i$ where $i = 1,2,3,4$

Step #2: total flux computation and threshold check:

- $tot\_flux = \sum \tilde{A}_i$ - if $tot\_flux < flux\_threshold$ then $tot\_flux = flux\_threshold$

Step #3: APD centroids computation:

- $C_{Ax} = \dfrac{A_1 - A_2 - A_3 + A_4}{tot\_flux}$
- $C_{Ay} = \dfrac{-A_1 - A_2 + A_3 + A_4}{tot\_flux}$

Step #4: APD centroids interaction matrix correction:

- $\begin{bmatrix} C_{ix} \\ C_{iy} \end{bmatrix} = \begin{bmatrix} m_{1,1} & m_{2,1} \\ m_{1,2} & m_{2,2} \end{bmatrix} \begin{bmatrix} C_{Ax} \\ C_{Ay} \end{bmatrix}$

Step #5: angular offset correction:

- $C_x = C_{ix} - C_{ox}$
- $C_y = C_{iy} - C_{oy}$

Step #6: control law servo computation in the discrete domain (3/3 order) applied to each centroid element:

- $m[n] = -b_1 m[n-1] - b_2 m[n-2] - b_3 m[n-3] + a_0 c[n] + a_1 c[n-1] + a_2 c[n-2] + a_3 c[n-3].$
  where: c is the generic input centroid element and m is the generic output element of the filter, which is the raw mirror output command;

Step #7: mirror output command offsetting:

- $\tilde{M} = M + M_{off}$

Step #7: disturbance history optional correction:

- $M_f = \tilde{M} + M_d[i]$

Step #8: DTT mirror 3 axis splitting:

- $$\begin{bmatrix} M_0 \\ M_1 \\ M_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\sin(\frac{\pi}{3}) & -\sin(\frac{\pi}{6}) \\ \sin(\frac{\pi}{3}) & -\sin(\frac{\pi}{6}) \end{bmatrix} \begin{bmatrix} M_{fx} \\ M_{fy} \end{bmatrix}$$

After the correction is computed the new three actuator mirror commands are passed to local actuator servo control loop to perform the desired tip-tilt mirror movement.

7.1.6.2 UTT mirror correction

The UTT mirror is used only in LGS mode.

The tip-tilt inputs are computed by the WFP (elements #349 and #350 of the residual wavefront vector).

Hereafter the sequence of computational steps:

Step #1: angular offset correction:

- $C_x = C_{ix} - C_{ox}$
- $C_y = C_{iy} - C_{oy}$

Step #2: control law servo computation in the discrete domain (3/3 order) applied to each centroid element:

- $m[n] = -b_1 m[n-1] - b_2 m[n-2] - b_3 m[n-3] + a_0 c[n] + a_1 c[n-1] + a_2 c[n-2] + a_3 c[n-3].$
  where: c is the generic input centroid element and m is the generic output element of the filter, which is the mirror output command;

Step #3: mirror output command offsetting:

- $\tilde{M} = M + M_{off}$

Step #4: disturbance history optional correction:

- $M_f = \tilde{M} + M_d[i]$

Step #5: UTT axis rotation:

- $$\begin{bmatrix} M_0 \\ M_1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} M_{fx} \\ M_{fy} \end{bmatrix}$$

After computing the correction, the new actuator mirror commands are passed to the local actuator servo control loop to perform the desired tip-tilt mirror movement.

7.1.7 Telemetry data

The scope of the telemetry data is to monitor the WFC system performance and status, to allow offloading of large adaptive optics corrections to the telescope control system and to reconstruct the system history by means of post processing.

For this reason two different kinds of telemetry data are available on the NGWFC, called averaged telemetry and full data telemetry:

- The averaged telemetry computes the average of the most important data with a user defined rate (typically 1 second). When available, the WCP reads these data to monitor the WFC system during operation.
- The full data telemetry saves most of the real-time data stream to a big data repository (TRS). The complete system configuration and all applied configuration changes are also saved by TRS. Therefore, it is always possible to reconstruct the past system functioning. The TRS can save about 50 hours of system operation at maximum frame rate.

7.1.7.1 Averaged telemetry computation

The averaged telemetry data can be distinguished in two groups: *continuous telemetry* data stream and *on demand* telemetry data stream. This telemetry data can be retrieved through the WIF interface (§7.2.2).

We report hereafter a list of all the telemetry streams computed online.

Continuous data stream is computed accumulating samples until a user definable period has expired, then the average is calculated and the result saved to be read by WIF. Automatically a new data accumulation begins until the following period expires and so on. Some continuous data streams are performed by the DSP of the AdOpt BCU board and others by MVME CPU board.

The following continuous time averaged values are computed by the AdOpt BCU board:

- *Tip-Tilt mirror command* (DTT or UTT according to the tip-tilt operating mode adopted) derived from the high voltage output command of the AdOpt HVC board. *DM Zernike focus* derived from the DM position vector.
- *WFS centroid vector* derived from the first 349 elements of the residual wavefront vector.
- *WFS Zernike tip-tilt* derived from the elements #349 and #350 of the residual wavefront vector.
- *WFS Zernike focus* derived from the element #351 of the residual wavefront vector.
- *WFS subaperture intensity values* derived from the 304 subaperture intensity values.

The following continuous time averaged values are computed by a dedicated real-time task on MVME CPU board. They are used when the DTT mirror command is calculated through the STRAP sensor device:

- *DTT mirror command* derived from the high voltage output command of the AdOpt HVC board.
- *APDs sensor centroids* derived from the acquired four APD counts.

On demand data stream is computed similarly to continuous data stream but the accumulation starts by means of a WIF command. A single accumulation and average computation is performed until a new request arrives.

This telemetry is computed by the AdOpt BCU board, the following output values are available on demand:

- *CCD raw frames*

7.1.7.2 Full data telemetry management

The full data telemetry is stored to TRS for off-line analysis and data retrieval. The telemetry data have been divided in four streams, according to the different update rates. This subdivision corresponds also to four different tables allocated on the TRS. The four streams are reported on Table 36:

Telemetry stream name	**Source**	**Rate**	**Transfer over:**
Full Frame rate telemetry Buffer (FFB)	MGAOS	Full frame rate	FibreChannel from MGAOS to TRS
Decimated Frame rate telemetry Buffer (DFB)	MGAOS	Decimated frame rate (up to 100 Hz)	FibreChannel from MGAOS to TRS
STRAP diagnostic	STRAP	Full STRAP acquisition rate (may be synchronous or not to frame rate, see Table 35)	VME backplane to VME CPU, then Ethernet from VME CPU to TRS
Configuration	VME CPU	On change	Ethernet from VME CPU to TRS

Table 36 - Telemetry buffer types

The details of FFB and DFB are reported in this section, while *STRAP diagnostic* and *Configuration* streams are reported in §7.1.7.2.2 and §7.2.4 respectively.

FFB and DFB data are first buffered and prepared in the WFP, taking into account that TRS is not a real time machine, and also in order to optimize communication between WFP and TRS.

In Table 37 all fields of FFB record are listed:

Section	**Field Name**	**Type**	**Size (byte)**
Header	Telemetry Buffer Type	Uint16	2
	Record Length	Uint16	2
	Timestamp	Uint64	8
	Frame Counter	Uint32	4
Data	Subaperture intensity vector	Uint32	304 x 4
	Offset centroid vector	Float32	608 x 4
	Residual wavefront error vector	Float32	349 x 4
	Residual TT & Focus	Float32	3 x 4
	DM commands vector	Float32	349 x 4
	DM commands clipped flag	Uint32	1 x 4
	Residual RMS	Float32	1 x 4
	Conf-id	Uint32	1 x 4
	Tip-Tilt mirror command data (DTT or UTT)	Float32	2 x 4
	CLMP strain gauge (DTT or UTT)	Float32	3 x 4
	Tip-Tilt commands clipped flag	Uint32	1 x 4
	Not used area	Uint32	2 x 4
Footer	Telemetry Buffer Type	Uint16	2
	Record Length	Uint16	2
	Timestamp	Uint64	8
	Frame Counter	Uint32	4
Total size			6528
Throughput @ 2Khz (CCD-39)			12.5 MB/s
Throughput @ 1Khz (CCiD-56)			6.2 MB/s

Table 37 - FFB record description

The Dummy field is an implementation detail which allows a 16 byte alignment of the various records. At each real-time computational cycle, all data for telemetry are transferred from the DSP memory of all AdOpt DSP board to the BCU and here these data are stored into a transfer queue located on SDRAM.

The DFB is running at reduced frame rate (up to 100 Hz) and retrieves the Raw CCD pixels from BCU DSP Ram. Also this transfer queue resides in BCU SDRAM. The single records of this buffer are listed in Table 38:

Section	**Field Name**	**Type**	**Size (byte) CCD39**	**Size (byte) CCiD59**
Header	Telemetry Buffer Type	Uint16	2	2
	Record Length	Uint16	2	2
	Timestamp	Uint64	8	8
	Frame Counter	Uint32	4	4
Data	Raw CCD data (from 1600 to 25600 pixels, after binning)	Uint16	Max. 12800 (80x80x16 bit)	Max. 51200 (160x160x16 bit)
Footer	Telemetry Buffer Type	Uint16	2	2
	Record Length	Uint16	2	2
	Timestamp	Uint64	8	8
	Frame Counter	Uint32	4	4
Total size			12832	51232
Throughput @ 100Hz			1.2 MB/s	4.9 MB/s

Table 38 - DFB record description

At each new frame, the BCU sequences the transfer of data from its local SDRAM to the TRS by means of the FibreChannel link. As already mentioned above, a strict time determinism on the TRS interface is not required, simply it has to be guaranteed that the mean data rate of ~13 MB/s is achieved. This rate is the highest throughput needed and is given when the CCD-39 is used at 2kHz (0.6 MB/s + 12.4MB/s).

7.1.7.2.1 WFP–TRS data transfer over FibreChannel

The communication between WFP–TRS goes over FibreChannel using UDP/IP protocol.

This is a standard protocol supported by many FibreChannel devices, and is also supported by the Qlogic HBA device installed on the TRS. An overview of the implementation details is given in Table 39.

The basic idea of UDP/IP over FibreChannel is to encapsulate standard UDP/IP packets into the data field of FibreChannel frames (see also [RD6]).

Standard	Name	Size (bytes)	Description
FibreChannel	SOF	4	Start of FC Frame
FibreChannel	Frame Header	24	FC Frame Header
FibreChannel	Network Header	16	Network header (is part of FC Frame payload)
FibreChannel	LCC/SNAP Header	8	LCC/SNAP Header
IP	IP Header	20	IP Header
UDP	UDP Header	8	UDP Header
TDP	TDP Header: Fiber Channel packet identifier	2	Fiber Channel packet identifier, this word identifies the sequence of Fiber Channel packets of the same Telemetry record. It starts always from zero.
TDP	TDP Header: Telemetry record identifier	2	Telemetry record identifier, this word identifies the telemetry record
TDP	TDP Header: telemetry record size	2	Total telemetry record size (in byte)
TDP	TDP Header: telemetry packet start address	2	It indicates the start address of the Fiber Channel packet respect to the entire Telemetry record (in byte). It starts always from zero.
TDP	Data…	…	Fiber Channel packet data. It contains a piece of the Telemetry record.
FibreChannel	CRC	4	Check packet word
FibreChannel	EOF	4	End of Frame

Table 39 - FC-UDP/IP encapsulation structure

The data transmission overhead of this protocol is about 6% for the FFB record. The overhead has been computed taking into account all protocol specific fields and all dummy bytes needed for alignment purposes.

Figure 43 shows how the telemetry record is reconstructed. Each telemetry record is composed by 1 or more telemetry data packets (TDP). The TDP header contains information how to rebuild the telemetry record.

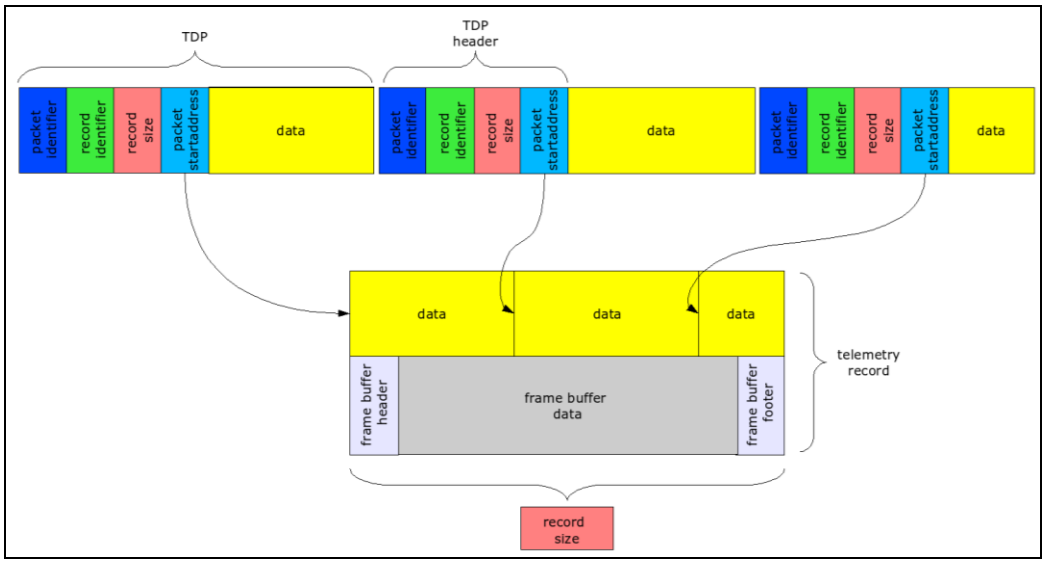


Figure 43 – telemetry frame reconstruction

7.1.7.2.2 MVME 6100 – TRS data transfer

The communication between MVME6100 and TRS goes over standard TCP/IP protocol using the public Ethernet interface. The data transfer rates on this communication link are quite small. We can distinguish three streams:

- RTC configuration
- DTT/STRAP data
- non-RTC telemetry

All data will be transferred using a single TCP/IP socket connecting WIF and TRS. Table 40 shows the base data packet of the WIF-TRS link.

Non-RTC telemetry is a record capability provided by the TRS for configuration data referring to devices outside of the NGWFC such as OBS, SC, NIRC2. This data are sent to the TRS by the WCP by means of the *SEND_TRS_NON_RTC_DATA* WIF command.

	MG_DF	Header Telemetry Buffer Type	uint16	Buffer Type to distinguish diagnostic buffer records (STRAP, CONFIG, NON-RTC)
	MG_DF	Header Record Length	uint32	
	MG_DF	Data…	…	

Table 40 - TCP-IP data structure for STRAP and Configuration streams (payload only shown here)

7.1.8 On-the-fly parameter swapping

For those parameters which needs to be changed on-fly the MGAOS reserves two memory locations: a used parameter area and a setup parameter area. The MVME6100 always writes in the setup area, then when it is done it sets an update flag triggering the MGAOS to copy the setup area into the used area. When the MGAOS has finished it will clear the update flag. Polling on the update flag the MVME6100 knows when the MGAOS has finished to copy. The copy is always done synchronously with the WFS frame.

When STRAP is used there are two update flags one for the part driven by the WFS and the other for the part driven by STRAP the mechanism how the on fly parameter update works are identical for both.

A dedicated WIF command UPDATE_PARAMETERS is available to control this on-fly update. In this way it is possible to change more then one parameter at a time.

7.2 MVME6100 Software

The software running on the MVME6100 is based on the VxWorks operating system and accomplishes the following main tasks:

- System initialization
- WIF interface
- STRAP and Timing support task
- System monitoring
- System configuration storage to TRS

The WCP software resides also on the MVME6100 but is not described in this document.

7.2.1 System start-up

At system start up the MVME6100 downloads the VxWorks operating system from a workstation host through Ethernet. When the VxWorks operating system is ready a start-up scripts are launched:

- the first part of the script will initialize the WIF and MGAOS software which have been developed by Microgate
- the second part will initialize all the WCP software which is developed by Keck

The first script will initialize:

- TRS communication
- IRIG timing board
- WFP: code is uploaded to the DSPs
- MVME to MGAOS communication
- WCP/WIF interface

After initialization is complete the RTC will be in the standby state with default parameters already loaded. The WIF is ready to accept commands from the WCP. At this point the WCP can change parameters or put the system in the normal state. Remember that the DTT and UTT mirrors must be started (wifStartDTT, wifStartUTT routines) in standby mode; the same, when shutting down the system it must first be set in standby mode and then the wifStopDTT and wifStopUTT routines can be called.

The system configuration is recorded on the TRS and contains all parameters which can be changed by the WIF interface. It is possible at any time to generate a binary configuration file using one of the following two methods:

- through the WIF interface using the *SAVE_CONFIGURATION* command
- directly on the TRS using the ctl program as follow: *ctl save filename*.

The above commands will generate a binary file which can be loaded by the RTC:

The structure of the configuration binary file is as follow:

```
NUMBER OF PARAMETERS
PARAMETER ID  PARAMETER SIZE  PARAMETER DATA
PARAMETER ID  PARAMETER SIZE  PARAMETER DATA
PARAMETER ID  PARAMETER SIZE  PARAMETER DATA
…
PARAMETER ID  PARAMETER SIZE  PARAMETER DATA
```

The first element is the number of parameters contained in the configuration file. It is a 4 byte integer (big endian). Then all the parameters are listed in sequence. For each parameter we have:

- parameter ID: same id as the related SET WIF command (4 bytes integer, big endian).
- parameter size: in bytes (4 bytes integer, big endian).
- parameter data: a byte sequence. The parameter data has the same format as the WIF command interface.

There is no newline or separation bytes between each parameter. The single parameter data can be distinguished by looking at the parameter data size.

Besides this WIF parameter configuration file there is a second configuration file (wifMGAOSBaseParameter.ini) for basic MGAOS configuration. The purpose of this configuration file is to initialize such parameters, which are not changeable through the WIF interface. These parameters will not require any change during operation. The structure of the file is strictly related to the mgp protocol. Each line

represents a mgp command to be executed. The file is processed during MGAOS initialization by the MVME.

Parameters like DTT/UTT, DAC/ADC offset and gain calibration are stored on the HVC board itself. In this way it is possible to replace the MGAOS hardware without changing the configuration files.

7.2.2 WIF/WCP interface

The Wavefront controller Interface (WIF) interfaces the WCP with the RTC system. It accepts commands from WCP, which is implemented on the same CPU. Commands are transferred by means of a simple VxWorks read/write driver. The WIF is a software interface that is used to transfer all control parameters to the WFP and to read back its status information. The interaction is achieved by means of a data exchange structure that relies on a real-time semaphore to synchronize the operation between WIF and WCP. The communication supports transfer of both scalars and more complex data structures (e.g. matrixes).

Hereafter we report the communication structure:

```
struct wifStruct
 {
   int semID
   int cmdIndex
   int status
   void *p_data
 }
```

- `semID` is a VxWorks semaphore identifier which is used for synchronization.
- `cmdIndex` is the command ID, refer to the WCP/WIF command list to see all available commands and details.
- `status` is the command status register; this is needed for command tracking and return information .i.e. RUNNING, SUCCES or FAILED.
- `p_data` is a pointer to a memory area. The definition of this area is related to the specific command. For instance for the Reconstruction Matrix setup command this is a pointer to a sequence of floats representing the matrix coefficients.

For communication to the MGAOS the WIF software bases on the Command Interpreter and Executor (CIE). Its main purpose is to translate high level WCP/WIF commands to the low level communication primitives according to MGP (Microgate UDP/IP Protocol) (see 7.2.2.2). A single WIF/WCP command may translate into a single MGP command or into a sequence of commands. In the latter case, the CIE generates and controls autonomously the execution of the MGP sequence. All commands to the MGAOS are sent only by the CIE, which has full control of the dedicated MGAOS-MVME6100 Ethernet link, thus the CIE can act as arbiter between real time data transfers (STRAP counts and IRIG timing information), and WIF/WCP commands.

During “operational mode” (default mode) the WIF will only accept WIF/WCP protocol compliant commands. For low level debugging purposes the WIF can be switched into “debugging mode” using the SET_DEBUG_MGP WIF command. In this mode the WIF will accept also direct MGP commands from external Ethernet, and in this sense it will just act as a bridge for MGP commands. This allows the user to use other SW instruments like Microgate Engpanel or Microgate Matlab scripts, or other instruments which use the MGP protocol to communicate and control the MGAOS.

This feature is extensively used for system development and testing, together with a dedicated tool that allows to generate WIF/WCP commands directly from an external Ethernet interface, without requiring the WCP itself. These aspects are treated in detail in [AD7].

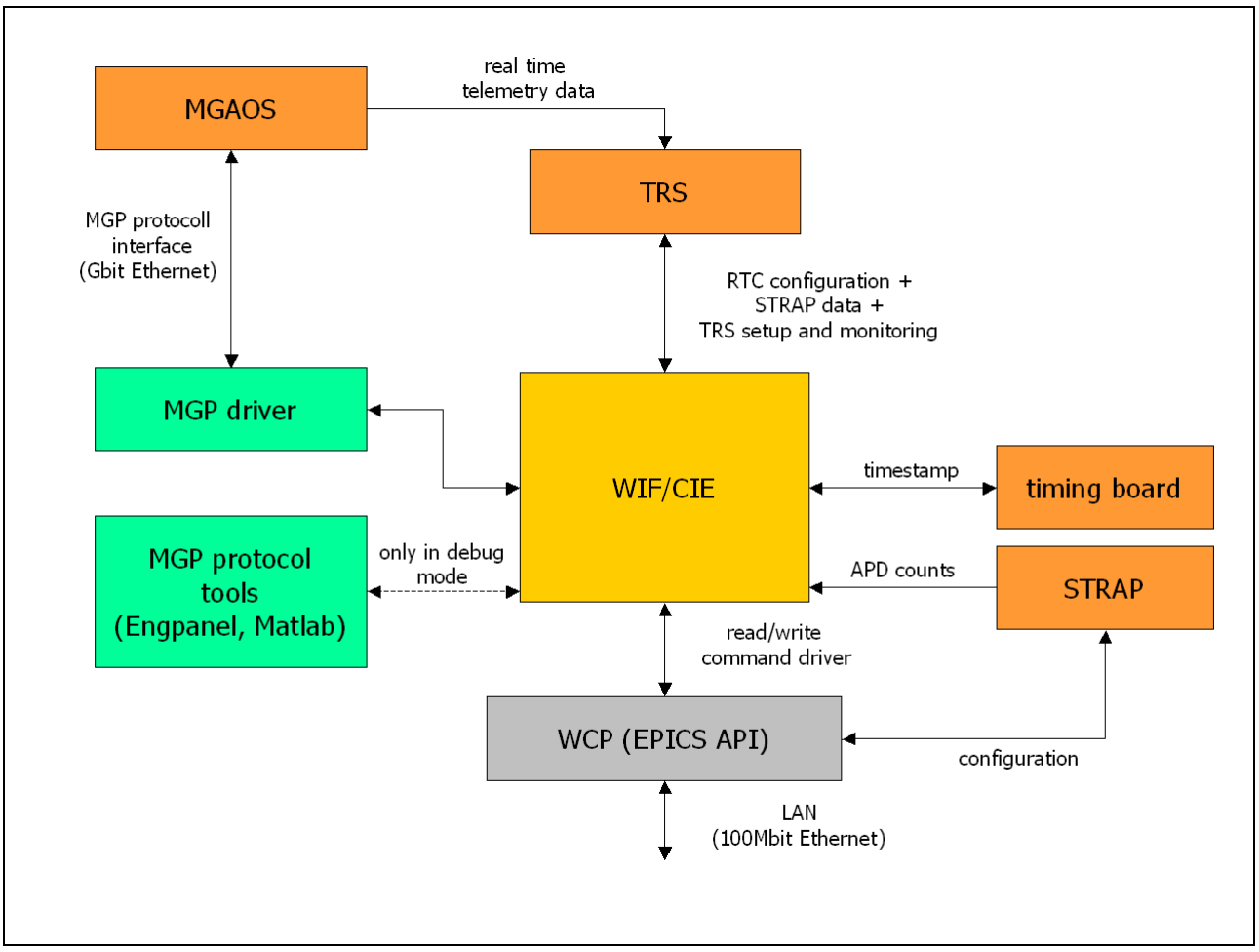


Figure 44 - MVME 6100 software block diagram

7.2.2.1.1 CIE command translation

As already said in the previous chapter, the main job of the CIE is sequencing WIF commands. The core of the CIE is a command table, *wif_cmdList* containing information on how to execute/perform a WIF command. The most important information in this table is a pointer to a function which will execute the command. Other meta information is also contained in this table. Doing so, some WIF commands can be grouped and handled by a single routine, just the meta information differs by each command.

7.2.2.1.2 Heartbeat/Watchdog

A watchdog has been implemented on the MGAOS to assure safety in case of comunication problems. The SET_WATCHDOG WIF command is used to control this feature. The watchdog can be:

- disabled with parameter 0
- enabled with parameter 1
- cleared with parameter 2

Once the watchdog has been enabled the WCP must periodically clear it (each 3 seconds). This watchdog mechanism assures that the master, i.e. the WCP, is always online and operation is normal. If there is some

communication problem with the WCP, the watchdog will trigger a mirror software shutdown procedure, and after few seconds (equal to mirror shutdown time) it will also reset itself.

The same mechanism is safe also when the MGAOS itself is failing. In fact, if the MVME does not receive correct replies, it will stop clearing the watchdog thus causing a reset of MGAOS.

The way the watchdog is implemented is crucial for correct operation of the above described mechanism.

The watchdog is realized in hardware (BCU logic) and before resetting the whole MGAOS it sends a hardware signal on the MGAOS backplane. This signal will start the safe shutdown procedures on the DSP and HVC boards.

7.2.2.2 MGP protocol

MGP protocol is a set of commands to operate the MGAOS system through Ethernet. The MGP commands are designed to allow a low level access to the MGAOS system. Read and write operations can be performed from/to each MGAOS component, i.e. SRAM, SDRAM, DSP RAM and FLASH, of each board. Instead of sending real operational commands, MGAOS operation is controlled by changing the status of specific memory locations, checked by the MGAOS software.

The single commands are specified by the MGP-protocol whose packets are a subset of the IP/UDP protocol. According to the standard Ethernet protocol, the structure of the Ethernet packet used for the MGAOS communication is the following:

Standard	Name	Size (bit)	Fixed value		Description
Ethernet	Destination address	48	0x----------		Destination MAC address
Ethernet	Source address	48	0x----------		Source MAC address
Ethernet	Type	16	0x0800		Ethernet protocol used: 0x0800 for IP protocol
Ethernet	Data…				
IP	Version	4	0x4	0x45	IP protocol version: the system uses IPv4
IP	IHL	4	0x5		IP Header length in dwords
IP	Type of service	8	0x00		don't care
IP	Total length	16	0x----		total length of IP&UDP packets in bytes
IP	Identification	16	0x----		the communication board sets always to 0x0000
IP	Flags	3	0x0000		bit 0: 0 = must be zero bit 1: 1 = don't fragment bit 2: 0 = last fragment
IP	Fragment offset	13			0 = first & last fragment
IP	Time to live	8	0x80		typical value
IP	Protocol	8	0x11		IP protocol used: 0x11 for the UDP protocol
IP	Header checksum	16	0x----		
IP	Source address	32	0x--------		Source IP address
IP	Destination address	32	0x--------		Destination IP address
IP	Data…				
UDP	Source port	16	0x2710		TCP/UDP port used: 0x2710 for Microgate UDP
UDP	Destination port	16	0x2710		TCP/UDP port used: 0x2710 for Microgate UDP
UDP	Length	16			UDP packet length in bytes
UDP	Checksum	16	0x----		
UDP	Data…				See the following description
MGP	dummy word	16	0x0000		
MGP	header dword #0	32	0x--------		
MGP	header dword #1	32	0x--------		
MGP	header dword #2	32	0x--------		
MGP	Data…				
Ethernet	CRC	32			

Table 41 – MGP record structure

The "data…" field of the UDP/MGP protocol is composed by a header and a data buffer in according to the command specified.

The command reply is obtained by an asynchronous Ethernet command sent by the communication board to the host that has sent the command. Each command has a command identifier which is copied to the reply command to match the command sent to the reply received.

The following table describes the UDP/MGP packet:

	Name	Size (bit)	Description
dummy word		16	Necessary to align the rest of MGP data to 32 bit
header dword #0	first crate	4	don't care
	first DSP	8	first DSP of the crate where the command should be actuated
	last crate	4	don't care
	last DSP	8	last DSP of the crate where the command should be actuated
	command	8	see the command table
header dword #1	length	16	packet length in dword
	flags	8	some commands can require additional flags
	command identifier	8	a 8 bit identifier of the packet. The same value is used on the reply packet to match the sent command to the reply
header dword #2	start address	32	start address where to write or to read the data
Data…	…		if necessary

Table 42 - MGP packet structure

The first DSP and last DSP are used to define from which board to which board the command should be performed. It reduces a lot the Ethernet communication loading in fact it is possible, for example, write the same buffer or different buffers or read different portion of memory of several devices with only one Ethernet command.

The limitations are:

- only one device can be accessed by a single command;
- all AdOpt DSP boards addressed by a broadcast command must be in consecutive sequence;
- 'special' boards, like BCU, are accessed individually and not in conjunction with DSP boards.

The size of a single Ethernet packet is limited by the Ethernet standard protocol. For the Fast Ethernet, maximum 100Mbit of speed, the maximum acceptable size is 1512 bytes, meanwhile for the Gigabit Ethernet it is possible go on up to 65536 bytes. Of course the best size is the right tradeoff between size, which reduces the start up delay, and latency of a single packet which is stopped until the previous one is not finished.

The maximum size currently accepted by a communication boards is 4096 bytes, which requires enabling the "Jumbo frames". Of course it is possible to use a smaller packet size, for example if the system is connected to a Fast Ethernet system.

The basic MGP-commands available are listed in following table:

Command name	Value	Description
MGP_OP_WRSAME_DSP	0	Writes the same buffer to the DSPs
MGP_OP_WRSEQ_DSP	1	Writes different buffers to the DSPs
MGP_OP_RDSEQ_DSP	2	Reads a portion of memory from the DSPs
RESERVED CMD	4	This command is not used in MGAOS
RESERVED CMD	5	This command is not used in MGAOS
RESERVED CMD	6	This command is not used in MGAOS
MGP_OP_RESET_DEVICES	10	Resets independently all the devices on the MGOAS crate
RESERVED CMD	11	This command is not used in MGAOS
MGP_OP_LOCK_FLASH	128	Locks a portion of the flash
MGP_OP_UNLOCK_FLASH	129	Unlocks a portion of the flash
MGP_OP_CLEAR_FLASH	130	Clears a portion of the flash
MGP_OP_WRITE_FLASH	131	Writes the same buffer to the flash
MGP_OP_RDSEQ_FLASH	132	Reads a portion of memory from the flash
MGP_OP_WR_SCIMEASURE_RAM	135	Writes to the AIA lookup table
RESERVED CMD	136	This command is not used in MGAOS
RESERVED CMD	137	This command is not used in MGAOS
MGP_OP_CLEAR_SDRAM	140	Clears a portion of the sdram
MGP_OP_WRSAME_SDRAM	141	Writes the same buffer to the sdram
MGP_OP_WRSEQ_SDRAM	142	Writes different buffers to the sdram
MGP_OP_RDSEQ_SDRAM	143	Reads a portion of memory from the sdram
MGP_OP_CLEAR_SRAM	145	Clears a portion of the sram
MGP_OP_WRSAME_SRAM	146	Writes the same buffer to the sram
MGP_OP_WRSEQ_SRAM	147	Writes different buffers to the sram
MGP_OP_RDSEQ_SRAM	148	Reads a portion of memory from the sram
RESERVED CMD	150	This command is not used in MGAOS
RESERVED CMD	151	This command is not used in MGAOS
RESERVED CMD	152	This command is not used in MGAOS
MGP_OP_WRITE_CCDI	155	Writes to the CCD module interface registers
MGP_OP_READ_CCDI	156	Reads from the CCD module interface registers
MGP_OP_CMD_SUCCESS	200	Used on the reply packet if the command has been processed with success
MGP_OP_CMD_FAULT	201	Used on the reply packet if a fault condition happened, Typically when the input parameters of the command was wronged
MGP_OP_CMD_TIMEOUT	202	Used on the reply packet if an internal code execution timeout has happened
MGP_OP_CMD_WARNING	203	Used to notify at the supervisor that the MGAOS has detected a warning condition
MGP_OP_CMD_DIAGNOSTIC	204	Used on the reply of a diagnostic packet
RESERVED CMD	240	This command is not used in MGAOS
RESERVED CMD	241	This command is not used in MGAOS
RESERVED CMD	242	This command is not used in MGAOS
RESERVED CMD	243	This command is not used in MGAOS
MGP_OP_CMD_NULL	255	Null command

Table 43 - MGP commands list

The complete format description of all the MGP commands is available in 8.1.

There are also some generic flags used to give some additional information on the command:

Command name	Bit	Description
MGP_FL_WANTREPLY	1	Enable the creation of a reply command from the MGAOS. To be set also when a read command is sent.
MGP_FL_ASQUADWORD	2	Used in the write to DSP memory commands. It enables a special write mode without break during the data transfer
RESERVED FLAG	3	This flag is not used in MGAOS
RESERVED FLAG	4	This flag is not used in MGAOS
RESERVED FLAG	5	This flag is not used in MGAOS

Table 44 - MGP flags list

7.2.3 System monitoring

The MGAOS boards have on-board diagnostics that allows keeping under control several 'system-health' parameters. Hereafter we provide a list of these parameters organized by board:

- BCU board
  a. FPGA temperature (measured directly on chip)
  b. Local power supply temperature
- DSP board
  a. FPGA temperature (measured directly on chip)
  b. DSP temperature
  c. Local power supply temperature
- HVC board
  a. FPGA temperature (measured directly on chip)
  b. DSP temperature
  c. Local power supply temperature
  d. High voltage drives temperature
  e. Actual voltage on each actuator (measured independently with respect to commanded voltage)
  f. Actual voltage on positive and negative high voltage power rails

All these parameters are acquired by each AdOpt board (BCU, DSP, HVC) and updated every second. External access to these parameters is obtained through the 'GET_STATUS' WIF command.

Beside the MGAOS status the GET_STATUS WIF command provides information about the TRS. This is necessary to ensure that data are registered properly, because MGAOS-TRS communication is only a one way communication. The TRS will check data integrity and if one data-frame is corrupted this will only be marked, and a data transmission failure counter will be incremented, but it will not signal the problem directly to MGAOS. The mgpDriver status information is also contained in the GET_STATUS WIF command.

7.2.4 System configuration storage to TRS

In order to well interpret the diagnostic data stored on TRS it is very important to know the exact operation mode/configuration for each data frame. Obviously it is not possible to save at each real time data frame the corresponding system configuration, because this would lead to saving a huge amount of redundant data. To be remembered that system configuration includes CCD-background, reconstruction matrix, control servo parameters, and so on.

To achieve this task reducing at a minimum the stored data the following scheme has been conceived.

At system startup the full configuration set is written to the TRS. Each different configuration, is stamped with a unique id, which is simply an incremental counter. The full configuration set will be uploaded to TRS only once, at start-up, afterwards, when the configuration changes only the single changed parameter will be registered. For each change the configuration-ID will be incremented.

The configuration storage and change is made by the MVME6100, but it is the MGAOS which really uses the configuration parameter set. MVME6100 doesn't know the exact frame/time when MGAOS uses a new configuration set. This is the rule of the conf-ID field in the FFB (see Table 37), it will be saved together with the other time-stamped real time data. So each data frame is associated with one unique configuration ID. Knowing this ID, it is possible to reconstruct the full configuration/setup of the system.

7.2.5 MVME6100 software structure

The two main blocks of the MVME6100 software are the low level mgp driver and the upper level wif software.

Figure 45 illustrates the internal organization of the mgp driver. The entry point for the mgp driver is the *mgpTransaction* routine. The mgpTransaction routine takes care of proper mgp packets generation and synchronization with the MGAOS. For instance, you can simply write a certain amount of DSP memory, no matter for alignment or size, this will be handled by the mgpTransaction, which will generate as many mgp packets as needed, and will wait until the last packet has got a reply from the MGAOS. In order to optimize performance and to avoid a lot of memory allocation and release, the packets are preallocated in the *mgp_packetFreeMsgQ*. The *mgpTransaction* will take a message from this queue, fill it with proper data and then it will send it to the *mgp_packetToSendMsgQ* queue.

The *mgpSend* task takes packets from the mgp_packetToSendMsgQ and sends this to the *MGAOS* using the UDP protocol. Once the MGAOS has processed the udp packet it will send a response.

The incoming udp packets are processed by the *mgpReceive* task. Note that the two tasks mgpSend and mgpReceive are totally independent, this allows to achieve a better Ethernet throughput. In order to correlate response packets to the previously sent packets the *mgp_packetTrackTable* is used. Each time a packet is sent an entry in this table is set. The mgpReceive task will check this table when it receives a mgpPacket. If the received UDP packet was the last packet of a transaction a semaphore to the mgpTransaction routine is given. Once the packet has finished his cycle it will be put back in the mgp_packetFreeMsgQ queue. It may be possible that some mgp packets do not get a response, may be UDP packet went lost, or the MGAOS is down. The *mgpTimeoutCheck* continuously checks the mgp_packetTrackTable for timeouted packets.

The *mgpDebugBridge* task listens to the public network interface for external clients like the Engpanel or Matlab library. Like the mgpTransaction routine it sends the packets to the mgp_packetToSendMsgQ and waits for reply coming from the *mgp_packetDebugToSendMsgQ* queue. The mgpReceive task looking into the mgp_packetTrackTable will distinguish between internal and external packets, in the latter case it will simply send the reply to the mgp_packetDebugToSendMsgQ. If debug mode is not enabled the mgpDebugBridge task will simply ignore external requests.

Normally packets to be send are always put at the bottom of the message queue, the real time packets however are set at the top.

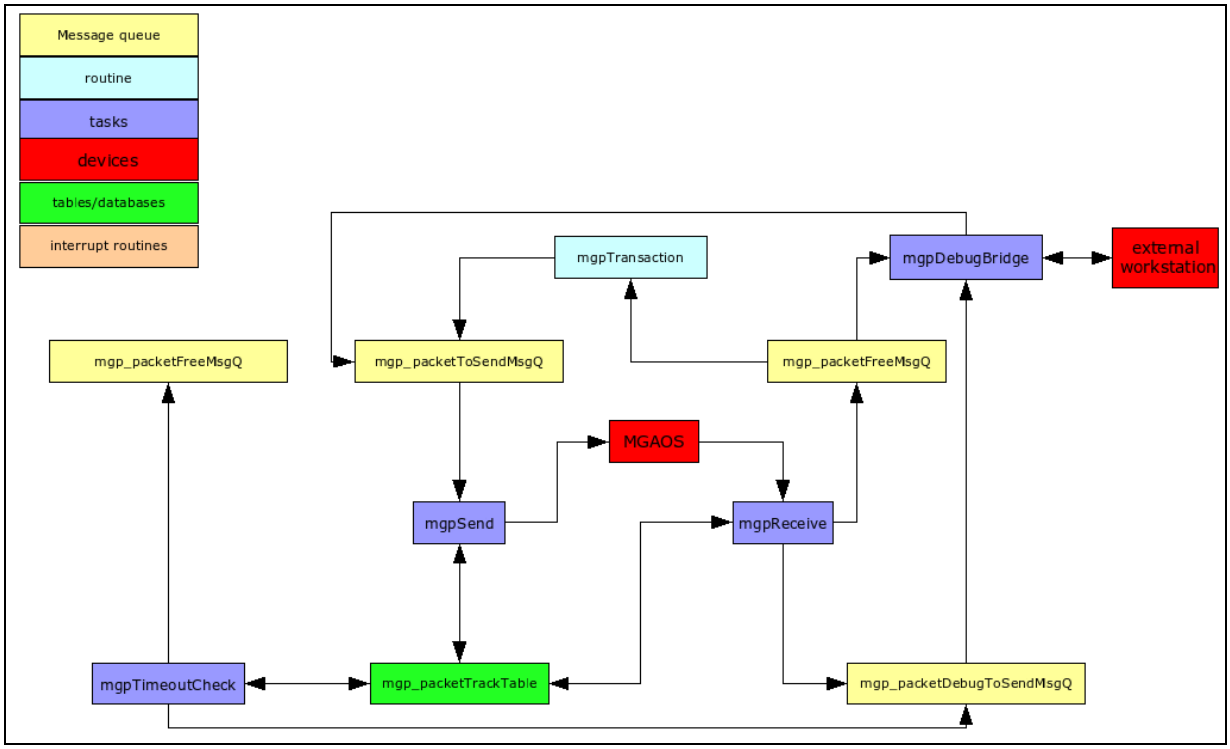


Figure 45 - MVME 6100 mgp driver scheme

Figure 46 illustrates the organization of the MVME6100 upper level software. *EPICS* commuicates with the WIF with a simple read/write driver. The *wifCmd* function is the entry routine for the WIF interface. The wifCmd routine looks into the *wif_cmdList* table for instructions how to process the received command. All commands are related to a dedicated function in this table. Generally a command can be performed by making some read/writes operations on the MGAOS or better by making some mgpTransactions. It will be the WIF/CIE which decides the priority, a specific mgpTransaction has to be performed, by supplying proper parameters to the mgpTransaction routine. The mgpTransaction routine is the entry for the mgpDriver software layer, at this point the WIF command will be handled by the mgpDriver.

Parameter storage to TRS: If the command has been performed successfully and the command is a parameter change which needs to be registered to the TRS the wifCmd will send the data into the *wif_trsPacketMsgQ*. (the wif_cmdList table contains a flag for each command indicating if it has to be stored to TRS or not). The *wifTRSSend* task takes data from the wif_trsPacketMsgQ and sends it to the TRS.

The *wifTRSStatus* task checks the TRS status at certain intervals, the information can be retrieved with the GET_STAUS WIF command.

The real time operation of the MVME is driven by the *MGAOS* which sends at each frame a synch signal to both *STRAP* and the *Irig board.*

The Irig board will latch the current time and signal to the MVME by means of a VME interrupt. The *wifTimestampIsr* interrupt routine reads data from the Irig board and signals to the *wifTimestampMGAOS* task trough a semaphore that a new frame has arrived. The wifTimestampMGAOS task will send the WIF_MGAOSREALTIMEPACKET to the MGAOS at each frame. This real time packet is described in Table 45.

field	type	Description
timestamp	uint64	is the WFS frame timestamp
avgDemTelemetryActStart	1 bit	this flag signalize to MGAOS to start on Demand average telemetry accumulation

chopUnFreeze	1 bit	this flag signalize to MGAOS to leave the DTT control loop in the freeze state because the system is on chopping transition phase
avgDemTelemetryActStop	1 bit	this flag signalize to MGAOS to stop on Demand average telemetry accumulation
avgContTelemetryAct	1 bit	this flag signalize to MGAOS to start a new continuous average telemetry accumulation; when the MGAOS receives this flag it will copy the current accumulation to a temporary location and start immediately with the new accumulation. The MVME will read from the temporary location
saveDFB	1 bit	this flag indicates to the MGAOS that CCD data of the current frame must be stored to the TRS
dataReady	1 bit	this flag notifies to the MGAOS that a new real time packet is available. The MGAOS will clear this flag once used.
counter	8 bit	this counter is used to synchronize with MGAOS

Table 45 - WIF_MGAOSREALTIMEPACKET

The wifTimestampMGAOS task will signalize to the *wifAvgDem* and the *wifAvgCont* tasks when a telemetry set is ready. This two tasks will read telemetry accumulators from the MGAOS and compute the average.

The STRAP real time part works in a similar way. Here we have two cases: asynchronous and synchronous operation. In synchronous mode the external STRAP interrupt is enabled. In this case there is no need for additional timestamping because it is the same as the WFS. However when STRAP operates asynchronously, i.e. an internal timer is used, the *wifSTRAPRtIsr* interrupt routine needs to read the timestamp from the Irig board. For this purpose, the internal software latch capabilities of the Irig board are used. In both cases the APD counts are read from the STRAP device. The transfer to the MGAOS of the APD counts is performed by the *wifSTRAPMGAOS* task. The wifSTRAPMGAOS task will send the WIF_STRAPREALTIMEPACKET to the MGAOS at each STRAP frame. This real time packet is described in Table 46.

field	**type**	**Description**
APDCounts	4xint16	this are the 4 STRAP APD counts
timestamp	uint64	is the STRAP frame timestamp. If STRAP operates in synchronous mode it will be the same as the WFS timestamp.
avgContTelemetryAct	1 bit	this flag signalize to MGAOS to start a new continuous average telemetry accumulation (STRAP part); when the MGAOS receives this flag it will copy the current accumulation to a temporary location and start immediately with the new accumulation. The MVME will read from the temporary location
dataReady	1 bit	this flag notifies to the MGAOS that a new real time packet is available. The MGAOS will clear this flag once used.

Table 46 - WIF_STRAPREALTIMEPACKET

To control DTT and UTT proper startup and shutdown, four routines are available namely *wifStartUTT*, *wifStopUTT*, *wifStartDTT* and *wifStopDTT*.

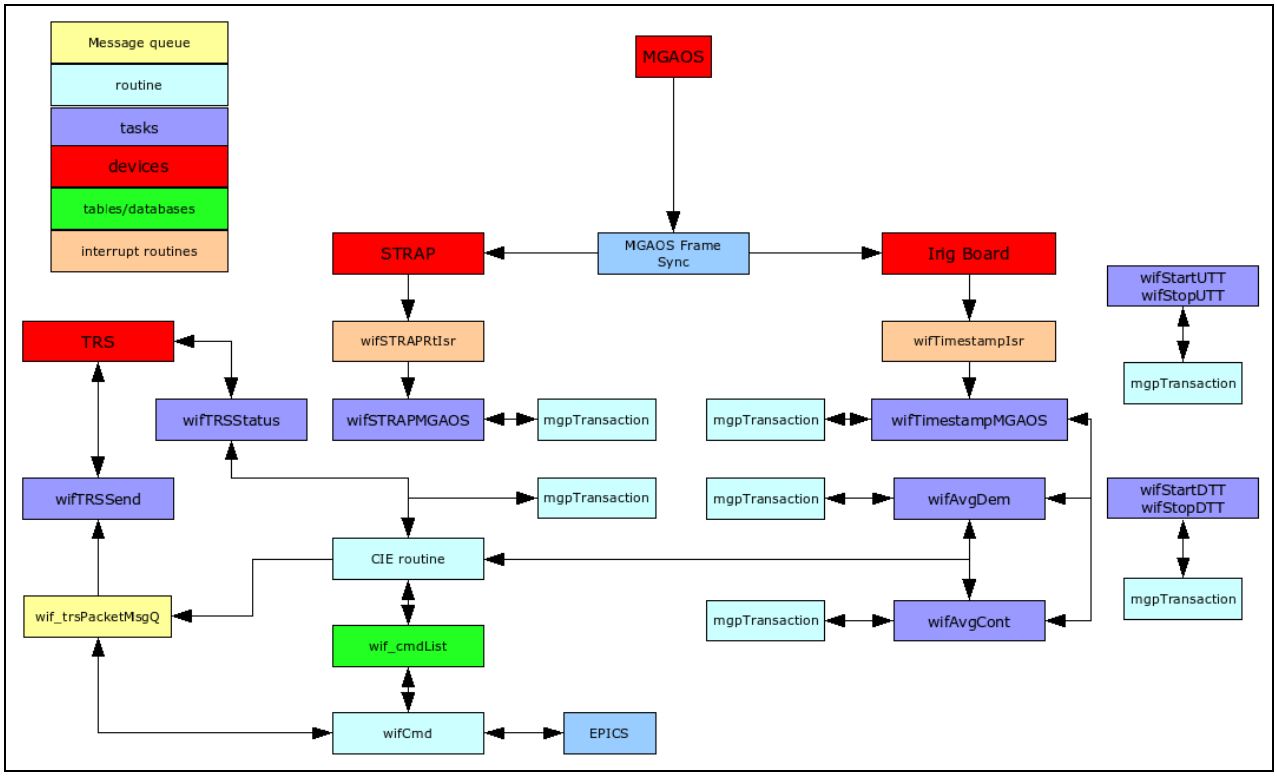


Figure 46 - MVME 6100 wif software scheme

7.3 TRS

The TRS is responsible for data storage and queries accomplishments. The TRS is based on two main blocks: the server running Solaris 10 and the disk array. The software running on the disk array system is not described in this document and can be treated as a black-box provided by the disk manufacturer. It is worth noticing here that the server sees the disk array just like a normal single big SCSI-disk.

Figure 47 shows how the TRS is organized. On the server we can distinguish two main software blocks:

- PostgreSQL database
- Storage client

The PostgreSQL is a relational database system of public domain, all information and documentation can be found online at www.postgresql.org. The layout of the database (database definition) will be described in the next chapters.

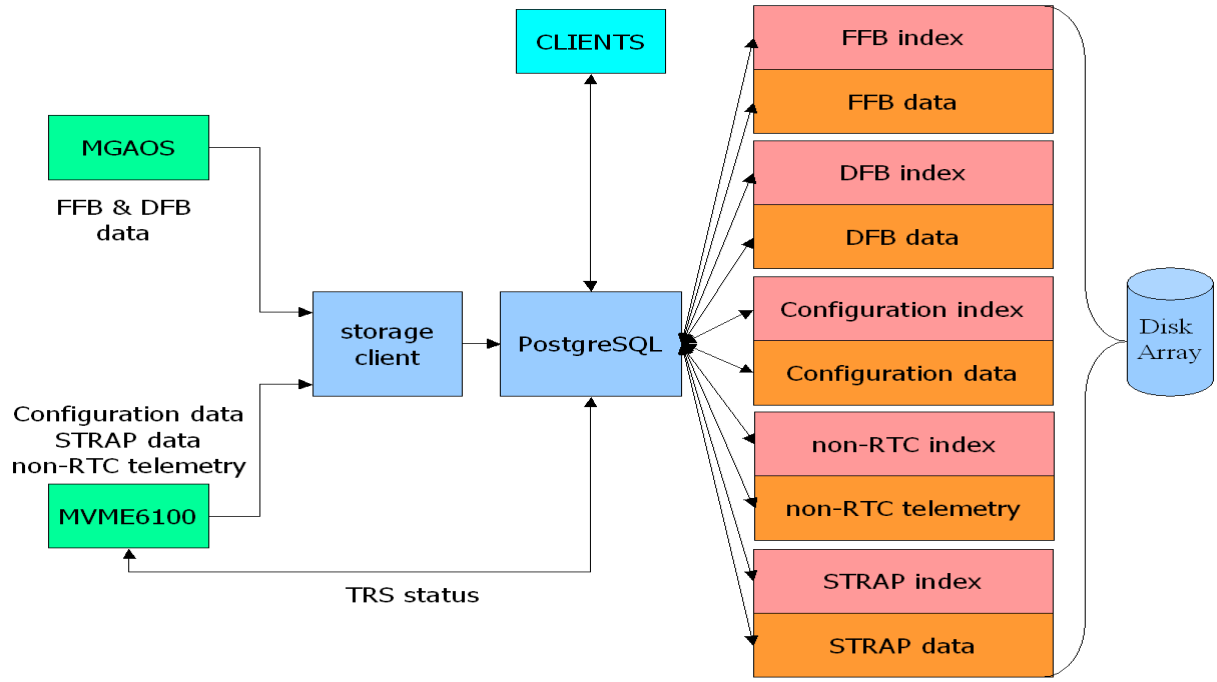


Figure 47 - TRS generic data storage

The storage client acts as a bridge between the RTC and the PostgreSQL database. It accepts data from both the MGAOS and the MVME6100 and sends this data to the PostgreSQL in a proper way. So the RTC will not communicate directly with Postgresql. Query clients whoever will connect directly to the PostgreSQL database running on the TRS over TCP/IP socket. Libraries to create this connection are available for different programming languages. For testing purposes a PostgreSQL shell is also available.

The use of a standard SQL database greatly improves the query server capabilities. Once a client is connected to the database it can submit queries using standard SQL language.

The query capabilities also depend on the internal data organization and on the custom defined PostgreSQL extensions. In a relational database like SQL data are organized in tables where each raw represents a single record of an object and each column represents all entries for a single field. In our case the single telemetry frames will represent a row of the telemetry storage table.

TIMESTAMP	**CENTROID**	**DTT_COMMAND**	**DM_COMMAND**
timestamp_1	centroid_1	dtt_command_1	dm_command_1
timestamp_2	centroid_2	dtt_command_2	dm_command_2
timestamp_3	centroid_3	dtt_command_3	dm_command_3

The smallest object the SQL can operate on is a cell; e.g., it will be possible to get each individual full DM_COMMAND vector, but not a single channel out of this vector. If this is needed, it can be extracted by post-processing the data returned by SQL. This can be done by exploiting the possibility of extending PostgreSQL capabilities with a user defined function, where the channel number is passed as parameter for example:

SELECT channel(DM_COMMAND,3) FROM telemetryTbl

Another important aspect is the records selection:

Te data telemetry frames will be stored in a table where the timestamp will act as an index and record ID. Queries based on index can be performed efficiently while queries where the ID is not specified will take longer to execute. As an example, a query like

SELECT max(rms(CCD)) FROM telemetryTbl WHERE timestamp > 920 and timestamp < 2343

will execute more efficiently than

SELECT max(rms(CCD)) FROM telemetryTbl

In the first case the rms(CCD) is computed only for frames with a given timestamp while in the second one the rms(CCD) is computed for all frames in the database.

Note the queries above, assumes that we have defined the RMS function for a CCD image with PostgreSQL functional extensions while the max of a float (rms(CCD)) is already a standard SQL function.

7.3.1.1 Extending PostgreSQL capability

As already mentioned above one of the most interesting feature of PostgreSQL is the possibility of extending its query capabilities by means of custom functions written in C. In this chapter the intention is to give a practical example of how this extensions works.

Our goal is to create a function, which computes the RMS of the CCD. The first task is to generate the function, hereafter we report the code:

```
1.  #include "postgres.h"
2.  #include "fmgr.h"
3.
4.  PG_FUNCTION_INFO_V1(rmsCCD);
5.  Datum rmsCCD(PG_FUNCTION_ARGS)
6.  {
7.      int nr_pixels=0;
8.      float rms=0;
9.      int rmsSum=0;
10.     int i;
11.     bytea *CCD=PG_GETARG_BYTEA_P(0);
12.     nr_pixels=(VARSIZE(CCD)-VARHDRSZ)/2;
13.     for (i=0;i<nr_pixels;i++)
14.         rmsSum+=(((uint16 *)VARDATA(CCD))[i]*((uint16 *)VARDATA(CCD))[i]);
15.     PG_RETURN_FLOAT4(sqrt(rmsSum/(nr_pixels-1)));
16. }
```

The first two lines are standard includes needed for PostgreSQL function extensions.

Line 4 is a macro which must precede each function definition; the parameter is the name of the function.

Line 5 is the function declaration. Each PostgreSQL extension function has Datum as return type and PG_FUNCITON_ARGS as argument.

Line 7 to 10 are simply some variable definitions needed for the rms computation.

Line 11 declares a pointer to bytea and is initialized with the first argument passed to this function. In the specific case CCD will point to the CCD memory area. Bytea is a PostgreSQL defined type which is used for general data storage and is essentially an array of bytes.

At line 12 we compute the number of pixels, so that the function works with every CCD configuration, regardless of its size. To accomplish this job, the VARSIZE macro is available which returns the total number of bytes of the bytea memory area. From this number we must subtract VARHDRSZ which is used to store the bytea memory size. Finally we have the raw size of memory occupied by the CCD image, since each pixel is an uint16 we divide by 2 and we get the number of pixels.

Line 13 and 14 is simply the rms computation. Here the VARDATA macro returns a pointer to the data of a bytea

Line 15 we return the value which with the PG_RETURN_XXX macro.

Once we have created the function we can create a library. Assuming we saved the above function in a file named rmsCCD.c the following two commands will create the library under Linux platform. For other platforms please refer to PostgreSQL documentation:

```
gcc –fpic –c rmsCCD.c
gcc –shared –o rmsCCD.so rmsCCD.o
```

Note: pic stay's for position independent code.

Now that we have the library we must register the function into PostgreSQL database. For this task the following commands must be executed from a PostgreSQL shell:

```
CREATE FUNCTION rmsCCD(img) RETURNS float
    AS 'FUNC_DIRECTORY/rmsCCD', 'rmsCCD'
    LANGUAGE C STRICT;
```

FUNC_DIRECTORY/rmsCCD is the compiled library filename without the "so" extension. We need to supply also the function name this is why a complied library file could contain more than one function. The last line C STRICT is for automatically null pointer checking. If a NULL data is supplied to the function the result will be empty and there is no need to make some pointer checking within the function itself.

Once the new function is registered we can use it in the SQL language:

```
dbtest=# SELECT rmsCCD(img) FROM dbtest WHERE id=5930920;

rmsCCD
---------
 593.920
(1 row)
```

7.3.2 Database layout

For each telemetry stream a dedicated storage table has been foreseen. Each real time data table will have a timestamp column, which acts as an index, and a column for each stream data field. Hereafter we describe the tables of the database

FFB: in this table the MGAOS FFB stream is stored.

column name	postgresql datatype	system datatype
buffertype	smallint	int16
recordlength	smallint	int16
timestamp	bigint	uint64
framecounter	integer	int
subapintensity	bytea	array of float
offsetcentroid	bytea	array of float
residualwavefront	bytea	array of float
dmcommand	bytea	array of float
dmcommandsclipped	bytea	int
residualrms	bytea	float
conf_id	integer	integer
ttmirrorcmddata	bytea	array of float
ttstraingauge	bytea	array of float
ttcommandsclipped	bytea	array of float

DFB: in this table the MGAOS DFB stream is stored.

column name	postgresql datatype	system datatype
buffertype	smallint	int16
recordlength	smallint	int16
timestamp	bigint	uint64
framecounter	integer	int
rawpixel	bytea	array of int16
rawpixelsize	smallint	int16

STRAP: in this table the STRAP data is stored. Please note that when STRAP is used the DTT data is also stored in this table, and the FFB will contain UTT data.

column name	postgresql datatype	system datatype
timestamp	bigint	uint64
framecounter	integer	int
conf_id	integer	int
apdcounts	bytea	array of float
dttcentroids	bytea	array of float
dttcommands	bytea	array of float
dttstraingage	bytea	array of float
dttcommandsclipped	bytea	array of float

NONRTC: this table is used to provide a general storage capability for system other then the RTC. A dedicated WIF command provides this capability.

column name	postgresql datatype	system datatype
timestamp	bigint	uint64
code	integer	int
data	bytea	free, is not defined

CONFIGURATION: this table contains all the RTC configuration. Each time a RTC parameter is changed it will be saved in this table.

column name	postgresql datatype	system datatype
conf_id	integer	int
par_id	integer	int
par_val	bytea	depends on the specific wif command

The conf_id is related with the conf_id of the FFB in this way it is possible to retrieve the configuration at a given timestamp. The par_id identifies the specific WIF parameter and has the same value as the specific WIF SET command. The par_val contains the value of the parameter and has the same format as the one of the WIF command. (see WIF command table for more details).

In order to allow a more comfortable configuration retrieval the following routines have been developed.

- *get_conf_all(conf_id):* returns all configuration parameters and their values at a given conf_id
  example usage: *select * from get_conf_all(1500);*
- *get_conf(par_id, conf_id):* returns the value of the given configuration parameter at a given conf_id
  example usage: *select * from get_conf(1000, 1500);*
- *ts2conf(ts bigint):* returns the conf id by a given timestamp by searching it in the ffb buffer. If timestamp does not exist it will simply return the timestamp just before... It is useful to use it in conjunction with get_conf and get_conf_all
  example usage:
  *select * from get_conf(1000, ts2conf(13));*
  *select * from get_conf_all(ts2conf(13));*

7.3.2.1 Endianness and array reordering

Many of the TRS tables uses the bytea for data storage. Bytea type is a PostgreSQL internal datatype and is essentially a memory block (sequence of bytes) and is useful for custom data representation. Since this datatype is just a sequence of bytes, the endianness can not be "controlled" by the PostgreSQL and its libraries. So the client software must take care of this depending on which hardware it runs. All data is stored with the same endianness of the MGAOS dsps i.e. Little Endian. The only exception for this are the parameter data (configuration table), where a Big Endian policy has been used, this simplifies the MVME-TRS comunication. Please note that the endianness is only a concern for the bytea types for all the other types it is the PostgreSQL client library itself which takes care of this.

Another important aspect is the array reordering. In order to optimize the real time centroids computation, and to allow computation pipelining, the MGAOS internal pixel ordering is not the same as the CCD read out sequence. The internal ordering is defined by the pixel LUT. All computational steps are done with this internal optimized pixel ordering. The residual wavefront is the last computational step, which depends from the pixel ordering. (The reconstruction matrix has to be compatible with this internal optimized pixel ordering). The real time data is stored to the TRS in the same order they are in the MGAOS. So when data is read from the TRS it has to be unscrambled according to the LUT in use. The TRS data fields interested for this pixel ordering are the following:

- ffb.subapintensity
- ffb.offsetcentroid
- dfb. rawpixel

7.4 SW Interfaces

7.4.1 Internal SW interfaces

Node #1	**Node #2**	**Protocol**	**Description**
MVME6100	MGAOS	MGP protocol (UDP socket port 10000)	WIF-WFP software interface
MVME6100	TRS	MG_DF (TCP port 10001)	STRAP diagnostic + configuration diagnostic
MVME6100	STRAP	Shared Ram, interrupt	STRAP can act as tip/tilt sensor
MVME6100	IRIG	Shared RAM, interrupt	Needed for timestamping
MGAOS	TRS	IP over FibreChannel (UDP port 10000)	DFB, FFB diagnostic

Table 47 - Internal SW interfaces

7.4.2 External SW interfaces

Node #1	**Node #2**	**Protocol**	**Description**
MVME6100	Host workstation	ftp, nfs protocol	Configuration and code download, (bootstrap)
MVME6100	WCP	read/write command driver	WCP-WIF interface
MVME6100	Workstation	Ssh (TCP)	Maintenance purposes
Workstation	TRS	postgres library (TCP)	Diagnostic data evualuation/quering
MVME6100/MGAOS	Workstation	Encapsulated (UDP) MGP protocol	Maintenance
MGAOS	WFS	AIA protocol	WFS interface
MGAOS	Xinetics HVA	Custom FPDP	DM command

Table 48 - External SW interfaces

8 Annex

8.1 MGP commands list

This annex includes the format of all MGP commands.

MGP_OP_WRSAME_DSP		this command allows to write the same buffer to all the DSPs as defined into the header	
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00 - 0xFF
	last crate	4	0x00
	last DSP	8	0x00 - 0xFF
	command	8	MGP_OP_WRSAME_DSP
header dword #1	length	16	0x0000 - 0x0FC0
	flags	8	allowed command flags: MGP_FL_WANTREPLY MGP_FL_ASQUADWORD
	command identifier	8	random value between 0x00 to 0xFF
header dword #2	start address	32	start address where to write the data
Data…		length	data buffer
Reply command if requested			
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00
	last crate	4	0x00
	last DSP	8	0x00
	reply_command	8	possible commands reply: MGP_OP_CMD_SUCCESS MGP_OP_CMD_FAULT MGP_OP_CMD_TIMEOUT
header dword #1	reply_length	16	0x0000
	flags	8	0x00
	command identifier	8	same command identifier of the master command
header dword #2	start address	32	0x00000000
Data…		0	not used

MGP_OP_WRSEQ_DSP		this command allows to write different buffers to all the DSPs as defined into the header	
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00 - 0xFF
	last crate	4	0x00
	last DSP	8	0x00 - 0xFF
	command	8	MGP_OP_WRSEQ_DSP
header dword #1	length	16	0x0000 - 0x0FC0
	flags	8	allowed command flags: MGP_FL_WANTREPLY MGP_FL_ASQUADWORD
	command identifier	8	random value between 0x00 to 0xFF
header dword #2	start address	32	start address where to write the data
Data…		length * (last_dsp-first_dsp+1)	data buffer
Reply command if requested			
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00
	last crate	4	0x00
	last DSP	8	0x00
	reply_command	8	possible commands reply: MGP_OP_CMD_SUCCESS MGP_OP_CMD_FAULT MGP_OP_CMD_TIMEOUT
header dword #1	reply_length	16	0x0000
	flags	8	0x00
	command identifier	8	same command identifier of the master command
header dword #2	start address	32	0x00000000
Data…		0	not used

MGP_OP_RDSEQ_DSP		this command allows to read a portion of all DSPs memory as defined into the header	
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00 - 0xFF
	last crate	4	0x00
	last DSP	8	0x00 - 0xFF
	command	8	MGP_OP_RDSEQ_DSP
header dword #1	length	16	0x0000 - 0x0FC0
	flags	8	allowed command flags: MGP_FL_WANTREPLY
	command identifier	8	random value between 0x00 to 0xFF
header dword #2	start address	32	start address where to read the data
Data…		0	not used
Reply command if requested			
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00
	last crate	4	0x00
	last DSP	8	0x00
	reply_command	8	possible commands reply: MGP_OP_CMD_SUCCESS MGP_OP_CMD_FAULT MGP_OP_CMD_TIMEOUT
header dword #1	reply_length	16	length * (last_dsp-first_dsp+1) if success otherwise 0
	flags	8	0x00
	command identifier	8	same command identifier of the master command
header dword #2	start address	32	0x00000000
Data…		length * (last_dsp-first_dsp+1) or 0	data buffer

MGP_OP_ RESET_DEVICES		this command resets all the devices specified on the command header	
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00 - 0xFF
	last crate	4	0x00
	last DSP	8	0x00 - 0xFF
	command	8	MGP_OP_RESET_DEVICES
header dword #1	length	16	0x0002
	flags	8	0x00
	command identifier	8	random value between 0x00 to 0xFF
header dword #2	start address	32	0x00000000
Data…		32 * 2	see the following table to configure the first dword.
Reply command if requested			
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00
	last crate	4	0x00
	last DSP	8	0x00
	reply_command	8	possible commands reply: MGP_OP_CMD_SUCCESS MGP_OP_CMD_FAULT MGP_OP_CMD_TIMEOUT
header dword #1	reply_length	16	0x0000
	flags	8	0x00
	command identifier	8	same command identifier of the master command
header dword #2	start address	32	0x00000000
Data…		0	not used

BIT (dword #0)	**AdOpt BCU board**	**AdOpt DSP board**	**AdOpt HVC board**
bits 0 – 1	reset all crate	not valid	not valid
bits 2 – 3	reset all boards excluded the BCU board	not valid	not valid
bits 4 – 5	not valid	reset board	reset board
bits 6 – 7	reset FPGA	reset FPGA	reset FPGA
bits 8 – 9	reset FLASH	reset FLASH	reset FLASH
bits 10 – 11	reset DSP	reset DSP #0	reset DSP #0
bits 12 – 13	not valid	reset DSP #1	not valid
bits 14 – 15	reset Ethernet chip	not valid	not valid
bits 16 – 17	not valid	not used	not used
bits 18 – 19	not valid	not used	not used
bits 20 – 21	control system watchdog	not used	not used
bit 22	generate IRIG & STRAP interrupt test	not used	not used
bit 23	generate Fiber Channel packet test	not used	not used
bits 24 – 31	not used	not used	not used

Table 49 - MGP_OP_RESET_DEVICES command: first control DWORD description

The reset operations available are: 00 = do nothing;

01 = to de-assert the signal reset;

10 = generate a sequence of reset if the device was operating;

11 = asserts the signal reset.

For the watchdog control, operations available are: 00 = do nothing;

01 = disable watchdog;

10 = enable watchdog;

11 = refresh watchdog (stay sleeping…).

BIT (dword #1)	AdOpt BCU board	AdOpt DSP board	AdOpt HVC board
bits 0 – 1	not valid	not used	enable/disable high voltage channel #0
bits 2 – 3	not used	not used	enable/disable high voltage channel #1
bits 4 – 5	not used	not used	enable/disable high voltage channel #2
bits 6 – 7	not used	not used	enable/disable high voltage channel #3
bits 8 – 9	not used	not used	enable/disable high voltage channel #4
bits 10 – 11	not used	not used	enable/disable high voltage channel #5
bits 12 – 13	not used	not used	not used
bits 14 – 15	not used	not used	not used
bits 16 – 31	not used	not used	not used

Table 50 - MGP_OP_RESET_DEVICES command: second control DWORD description

For the high voltage channels, operations available are: 00 = do nothing;

01 = disable channel;

10 = enable channel;

11 = do nothing.

MGP_OP_LOCK_FLASH		this command allows to lock a portion of the Flash area to avoid the possibility to overwrite it	
	Name	Size (bit)	Value
header dword #0	first crate	4	0x00
	first DSP	8	0x00 - 0xFF
	last crate	4	0x00
	last DSP	8	0x00 - 0xFF
	command	8	MGP_OP_LOCK_FLASH
header dword #1	length	16	0x0001
	flags	8	allowed command flags: MGP_FL_WANTREPLY
	command identifier	8	random value between 0x00 to 0xFF
header dword #2	start address	32	start address where to start to lock the flash
Data…		32	size of portion of flash to lock
Reply command if requested			
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00
	last crate	4	0x00
	last DSP	8	0x00
	reply_command	8	possible commands reply: MGP_OP_CMD_SUCCESS MGP_OP_CMD_FAULT MGP_OP_CMD_TIMEOUT
header dword #1	reply_length	16	0x0000
	flags	8	0x00
	command identifier	8	same command identifier of the master command
header dword #2	start address	32	0x00000000
Data…		0	not used

MGP_OP_UNLOCK_FLASH		this command allows to unlock the entire Flash, it is not possible to unlock a single portion of Flash	
	Name	Size (bit)	Value
header dword #0	first crate	4	0x00
	first DSP	8	0x00 - 0xFF
	last crate	4	0x00
	last DSP	8	0x00 - 0xFF
	command	8	MGP_OP_UNLOCK_FLASH
header dword #1	length	16	0x0000
	flags	8	allowed command flags: MGP_FL_WANTREPLY
	command identifier	8	random value between 0x00 to 0xFF
header dword #2	start address	32	start address where to start to lock the flash
Data…		0	not used
Reply command if requested			
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00
	last crate	4	0x00
	last DSP	8	0x00
	reply_command	8	possible commands reply: MGP_OP_CMD_SUCCESS MGP_OP_CMD_FAULT MGP_OP_CMD_TIMEOUT
header dword #1	reply_length	16	0x0000
	flags	8	0x00
	command identifier	8	same command identifier of the master command
header dword #2	start address	32	0x00000000
Data…		0	not used

MGP_OP_CLEAR_FLASH		this command allows to clear a portion of the Flash area	
	Name	Size (bit)	Value
header dword #0	first crate	4	0x00
	first DSP	8	0x00 - 0xFF
	last crate	4	0x00
	last DSP	8	0x00 - 0xFF
	command	8	MGP_OP_CLEAR_FLASH
header dword #1	length	16	0x0001
	flags	8	allowed command flags: MGP_FL_WANTREPLY
	command identifier	8	random value between 0x00 to 0xFF
header dword #2	start address	32	start address where to start to clear the flash
Data…		32	size of portion of flash to clear
Reply command if requested			
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00
	last crate	4	0x00
	last DSP	8	0x00
	reply_command	8	possible commands reply: MGP_OP_CMD_SUCCESS MGP_OP_CMD_FAULT MGP_OP_CMD_TIMEOUT
header dword #1	reply_length	16	0x0000
	flags	8	0x00
	command identifier	8	same command identifier of the master command
header dword #2	start address	32	0x00000000
Data…		0	not used

MGP_OP_WRITE_FLASH		this command allows to write the same buffer to all the flash as defined into the header. It is not possible to perform a write sequential to the flash device	
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00 - 0xFF
	last crate	4	0x00
	last DSP	8	0x00 - 0xFF
	command	8	MGP_OP_WRITE_FLASH
header dword #1	length	16	0x0000 - 0x0FC0
	flags	8	allowed command flags: MGP_FL_WANTREPLY
	command identifier	8	random value between 0x00 to 0xFF
header dword #2	start address	32	start address where to write the data
Data…		length	data buffer
Reply command if requested			
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00
	last crate	4	0x00
	last DSP	8	0x00
	reply_command	8	possible commands reply: MGP_OP_CMD_SUCCESS MGP_OP_CMD_FAULT MGP_OP_CMD_TIMEOUT
header dword #1	reply_length	16	0x0000
	flags	8	0x00
	command identifier	8	same command identifier of the master command
header dword #2	start address	32	0x00000000
Data…		0	not used

MGP_OP_RDSEQ_FLASH		this command allows to read a portion of all flash memory as defined into the header	
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00 - 0xFF
	last crate	4	0x00
	last DSP	8	0x00 - 0xFF
	command	8	MGP_OP_RDSEQ_FLASH
header dword #1	length	16	0x0000 - 0x0FC0
	flags	8	allowed command flags: MGP_FL_WANTREPLY
	command identifier	8	random value between 0x00 to 0xFF
header dword #2	start address	32	start address where to read the data
Data…		0	not used
Reply command if requested			
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00
	last crate	4	0x00
	last DSP	8	0x00
	reply_command	8	possible commands reply: MGP_OP_CMD_SUCCESS MGP_OP_CMD_FAULT MGP_OP_CMD_TIMEOUT
header dword #1	reply_length	16	length * (last_dsp-first_dsp+2) / 2 if success otherwise 0
	flags	8	0x00
	command identifier	8	same command identifier of the master command
header dword #2	start address	32	0x00000000
Data…		length * (last_dsp-first_dsp+2) / 2 or 0	data buffer

MGP_OP_WR_SCIMEASURE_RAM		this command allows to write a buffer to the lookup table of the SciMeasure to BCU logic interface.	
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0xFF (allowed on BCU board only)
	last crate	4	0x00
	last DSP	8	0xFF (allowed on BCU board only)
	command	8	MGP_OP_WRITE_FLASH
header dword #1	length	16	0x0000 - 0x0FC0
	flags	8	allowed command flags: MGP_FL_WANTREPLY
	command identifier	8	random value between 0x00 to 0xFF
header dword #2	start address	32	start address where to write the data
Data…		length	data buffer
Reply command if requested			
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00
	last crate	4	0x00
	last DSP	8	0x00
	reply_command	8	possible commands reply: MGP_OP_CMD_SUCCESS MGP_OP_CMD_FAULT MGP_OP_CMD_TIMEOUT
header dword #1	reply_length	16	0x0000
	flags	8	0x00
	command identifier	8	same command identifier of the master command
header dword #2	start address	32	0x00000000
Data…		0	not used

MGP_OP_CLEAR_SDRAM		this command allows to clear a portion of the Sdram area	
	Name	Size (bit)	Value
header dword #0	first crate	4	0x00
	first DSP	8	0x00 - 0xFF
	last crate	4	0x00
	last DSP	8	0x00 - 0xFF
	command	8	MGP_OP_CLEAR_SDRAM
header dword #1	length	16	0x0001
	flags	8	allowed command flags: MGP_FL_WANTREPLY
	command identifier	8	random value between 0x00 to 0xFF
header dword #2	start address	32	start address where to start to clear the Sdram
Data…		32	size of portion of Sdram to clear
Reply command if requested			
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00
	last crate	4	0x00
	last DSP	8	0x00
	reply_command	8	possible commands reply: MGP_OP_CMD_SUCCESS MGP_OP_CMD_FAULT MGP_OP_CMD_TIMEOUT
header dword #1	reply_length	16	0x0000
	flags	8	0x00
	command identifier	8	same command identifier of the master command
header dword #2	start address	32	0x00000000
Data…		0	not used

MGP_OP_WRSAME_SDRAM		this command allows to write the same buffer to all the Sdram as defined into the header.	
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00 - 0xFF
	last crate	4	0x00
	last DSP	8	0x00 - 0xFF
	command	8	MGP_OP_WRSAME_SDRAM
header dword #1	length	16	0x0000 - 0x0FC0
	flags	8	allowed command flags: MGP_FL_WANTREPLY
	command identifier	8	random value between 0x00 to 0xFF
header dword #2	start address	32	start address where to write the data
Data…		length	data buffer
Reply command if requested			
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00
	last crate	4	0x00
	last DSP	8	0x00
	reply_command	8	possible commands reply: MGP_OP_CMD_SUCCESS MGP_OP_CMD_FAULT MGP_OP_CMD_TIMEOUT
header dword #1	reply_length	16	0x0000
	flags	8	0x00
	command identifier	8	same command identifier of the master command
header dword #2	start address	32	0x00000000
Data…		0	not used

MGP_OP_WRSEQ_SDRAM		this command allows to write different buffers to all the Sdram as defined into the header.	
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00 - 0xFF
	last crate	4	0x00
	last DSP	8	0x00 - 0xFF
	command	8	MGP_OP_WRSEQ_SDRAM
header dword #1	length	16	0x0000 - 0x0FC0
	flags	8	allowed command flags: MGP_FL_WANTREPLY
	command identifier	8	random value between 0x00 to 0xFF
header dword #2	start address	32	start address where to write the data
Data…		length * (last_dsp-first_dsp+2) / 2	data buffer
Reply command if requested			
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00
	last crate	4	0x00
	last DSP	8	0x00
	reply_command	8	possible commands reply: MGP_OP_CMD_SUCCESS MGP_OP_CMD_FAULT MGP_OP_CMD_TIMEOUT
header dword #1	reply_length	16	0x0000
	flags	8	0x00
	command identifier	8	same command identifier of the master command
header dword #2	start address	32	0x00000000
Data…		0	not used

MGP_OP_RDSEQ_SDRAM		this command allows to read a portion of all Sdram memory as defined into the header	
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00 - 0xFF
	last crate	4	0x00
	last DSP	8	0x00 - 0xFF
	command	8	MGP_OP_RDSEQ_SDRAM
header dword #1	length	16	0x0000 - 0x0FC0
	flags	8	allowed command flags: MGP_FL_WANTREPLY
	command identifier	8	random value between 0x00 to 0xFF
header dword #2	start address	32	start address where to read the data
Data…		0	not used
Reply command if requested			
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00
	last crate	4	0x00
	last DSP	8	0x00
	reply_command	8	possible commands reply: MGP_OP_CMD_SUCCESS MGP_OP_CMD_FAULT MGP_OP_CMD_TIMEOUT
header dword #1	reply_length	16	length * (last_dsp-first_dsp+2) / 2 if success otherwise 0
	flags	8	0x00
	command identifier	8	same command identifier of the master command
header dword #2	start address	32	0x00000000
Data…		length * (last_dsp-first_dsp+2) / 2 or 0	data buffer

MGP_OP_CLEAR_SRAM		this command allows to clear a portion of the Sram area	
	Name	Size (bit)	Value
header dword #0	first crate	4	0x00
	first DSP	8	0x00 - 0xFF
	last crate	4	0x00
	last DSP	8	0x00 - 0xFF
	command	8	MGP_OP_CLEAR_SRAM
header dword #1	length	16	0x0001
	flags	8	allowed command flags: MGP_FL_WANTREPLY
	command identifier	8	random value between 0x00 to 0xFF
header dword #2	start address	32	start address where to start to clear the Sram
Data…		32	size of portion of Sram to clear
Reply command if requested			
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00
	last crate	4	0x00
	last DSP	8	0x00
	reply_command	8	possible commands reply: MGP_OP_CMD_SUCCESS MGP_OP_CMD_FAULT MGP_OP_CMD_TIMEOUT
header dword #1	reply_length	16	0x0000
	flags	8	0x00
	command identifier	8	same command identifier of the master command
header dword #2	start address	32	0x00000000
Data…		0	not used

MGP_OP_WRSAME_SRAM		this command allows to write the same buffer to all the Sram as defined into the header.	
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00 - 0xFF
	last crate	4	0x00
	last DSP	8	0x00 - 0xFF
	command	8	MGP_OP_WRSAME_SRAM
header dword #1	length	16	0x0000 - 0x0FC0
	flags	8	allowed command flags: MGP_FL_WANTREPLY
	command identifier	8	random value between 0x00 to 0xFF
header dword #2	start address	32	start address where to write the data
Data…		length	data buffer
Reply command if requested			
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00
	last crate	4	0x00
	last DSP	8	0x00
	reply_command	8	possible commands reply: MGP_OP_CMD_SUCCESS MGP_OP_CMD_FAULT MGP_OP_CMD_TIMEOUT
header dword #1	reply_length	16	0x0000
	flags	8	0x00
	command identifier	8	same command identifier of the master command
header dword #2	start address	32	0x00000000
Data…		0	not used

MGP_OP_WRSEQ_SRAM		this command allows to write different buffers to all the Sram as defined into the header.	
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00 - 0xFF
	last crate	4	0x00
	last DSP	8	0x00 - 0xFF
	command	8	MGP_OP_WRSEQ_SRAM
header dword #1	length	16	0x0000 - 0x0FC0
	flags	8	allowed command flags: MGP_FL_WANTREPLY
	command identifier	8	random value between 0x00 to 0xFF
header dword #2	start address	32	start address where to write the data
Data…		length * (last_dsp-first_dsp+2) / 2	data buffer
Reply command if requested			
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00
	last crate	4	0x00
	last DSP	8	0x00
	reply_command	8	possible commands reply: MGP_OP_CMD_SUCCESS MGP_OP_CMD_FAULT MGP_OP_CMD_TIMEOUT
header dword #1	reply_length	16	0x0000
	flags	8	0x00
	command identifier	8	same command identifier of the master command
header dword #2	start address	32	0x00000000
Data…		0	not used

MGP_OP_RDSEQ_SRAM		this command allows to read a portion of all Sram memory as defined into the header	
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00 - 0xFF
	last crate	4	0x00
	last DSP	8	0x00 - 0xFF
	command	8	MGP_OP_RDSEQ_SRAM
header dword #1	length	16	0x0000 - 0x0FC0
	flags	8	allowed command flags: MGP_FL_WANTREPLY
	command identifier	8	random value between 0x00 to 0xFF
header dword #2	start address	32	start address where to read the data
Data…		0	not used
Reply command if requested			
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00
	last crate	4	0x00
	last DSP	8	0x00
	reply_command	8	possible commands reply: MGP_OP_CMD_SUCCESS MGP_OP_CMD_FAULT MGP_OP_CMD_TIMEOUT
header dword #1	reply_length	16	length * (last_dsp-first_dsp+2) / 2 if success otherwise 0
	flags	8	0x00
	command identifier	8	same command identifier of the master command
header dword #2	start address	32	0x00000000
Data…		length * (last_dsp-first_dsp+2) / 2 or 0	data buffer

MGP_OP_WRITE_CCDI		this command writes to the CCD interface module registers	
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00 - 0xFF
	last crate	4	0x00
	last DSP	8	0x00 - 0xFF
	command	8	MGP_OP_WRITE_CCDI
header dword #1	length	16	0x0000 - 0x0007
	flags	8	allowed command flags: MGP_FL_WANTREPLY
	command identifier	8	random value between 0x00 to 0xFF
header dword #2	start address	32	start address where to write data
Data…		length	registers value
Reply command if requested			
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00
	last crate	4	0x00
	last DSP	8	0x00
	reply_command	8	possible commands reply: MGP_OP_CMD_SUCCESS MGP_OP_CMD_FAULT MGP_OP_CMD_TIMEOUT
header dword #1	reply_length	16	0x0000
	flags	8	0x00
	command identifier	8	same command identifier of the master command
header dword #2	start address	32	0x00000000
Data…		0	not used

MGP_OP_READ_CCDI		this command reads from the CCD interface module registers	
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0xFF (only)
	last crate	4	0x00
	last DSP	8	0xFF (only)
	command	8	MGP_OP_READ_CCDI
header dword #1	length	16	0x0000 - 0x0007
	flags	8	allowed command flags: MGP_FL_WANTREPLY
	command identifier	8	random value between 0x00 to 0xFF
header dword #2	start address	32	start address where to read the data
Data…		0	not used
Reply command if requested			
	Name	**Size (bit)**	**Value**
header dword #0	first crate	4	0x00
	first DSP	8	0x00
	last crate	4	0x00
	last DSP	8	0x00
	reply_command	8	possible commands reply: MGP_OP_CMD_SUCCESS MGP_OP_CMD_FAULT MGP_OP_CMD_TIMEOUT
header dword #1	reply_length	16	length if success otherwise 0
	flags	8	0x00
	command identifier	8	same command identifier of the master command
header dword #2	start address	32	0x00000000
Data…		length or 0	data buffer

8.2 PostgreSQL

PostgreSQL is a powerful, open source relational database system. It has more than 15 years of active development and a proven architecture that has earned it a strong reputation for reliability, data integrity, and correctness. It runs on all major operating systems, including Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, SunOS, Tru64), BeOS, and Windows. It is fully ACID compliant, has full support for foreign keys, joins, views, triggers, and stored procedures (in multiple languages). It includes most SQL92 and SQL99 data types, including INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, and TIMESTAMP. It also supports storage of binary large objects, including pictures, sounds, or video. It has native programming interfaces for C/C++, Java, Perl, Python, Ruby, Tcl, ODBC.

An enterprise class database, PostgreSQL boasts sophisticated features such as the Multi-Version Concurrency Control (MVCC), point in time recovery, tablespaces, asynchronous replication, nested transactions (savepoints), online/hot backups, a sophisticated query planner/optimizer, and write ahead log for fault tolerance. It is highly scalable both in sheer quantity of data it can manage and in the number of concurrent users it can accommodate. There are active PostgreSQL systems in production environments that manage in excess of 4 terabytes of data. Some general PostgreSQL limits are included in the table below.

Limit	Value
Maximum Database Size	Unlimited
Maximum Table Size	32 TB
Maximum Row Size	1.6 TB
Maximum Field Size	1 GB
Maximum Rows per Table	Unlimited
Maximum Columns per Table	250 - 1600 depending on column types
Maximum Indexes per Table	Unlimited

SQL implementation conforms to the ANSI-SQL 92/99 standards. It has full support for subqueries (including subselects in the FROM clause), read-committed and serializable transaction isolation levels. And while PostgreSQL has a fully relational system catalog which itself supports multiple schemas per database, its catalog is also accessible through the Information Schema as defined in the SQL standard.

PostgreSQL has won praise from it's users and industry recognition, including the Linux New Media Award for TRS data storage organizationBest Database System and three time winner of the The Linux Journal Editors' Choice Award for best DBMS.

PostgreSQL's source code is available under the most liberal open source license: the BSD license. This license gives you the freedom to use, modify and distribute PostgreSQL in any form you like, open or closed source. Any modifications, enhancements, or changes you make are yours to do with as you please. As such, PostgreSQL is not only a powerful database system capable of running the enterprise, it is a development platform upon which to develop in-house, web, or commercial software products that require a capable RDBMS.

To learn more about PostgreSQL please visit the official PostgreSQL website at www.postgresql.org

8.3 Final COTS selection

Qty	Component	Product	Manufacturer
1	VME CPU board	MVME6100-0173	Motorola
1	IRIG board	TTM635VME-VCXO	Symmetricom
1	VME/MGAOS PSU	CPCI-AC-6U-400	Schroff
3	HV PSU	N6700B + 3 N6736B	Agilent Technologies
1	TRS Server	Fire x4100	Sun
1	TRS Disk Array	SR-TRITON16FA	Partners
2	Fibre Channel Communication	QLA2340-CK	Qlogic
1	VME Crate	VME64 20836	Schroff

Table 51 –COTS product list

8.4 Microgate boards schematics, layout and bill of material

8.4.1 AdOpt BCU 4.0

Schematics

AdOpt-BCU-Main 4.0

AdOpt-BCU-Bus Interface 4.0

AdOpt-BCU-Clock 4.0

AdOpt-BCU-Dsp 4.0

AdOpt-BCU-Fast Link A 4.0

AdOpt-BCU-Fast Link B 4.0

AdOpt-BCU-Fast Link C 4.0

AdOpt-BCU-Fast Link D 4.0

AdOpt-BCU-Flash Sram Cpld 4.0

AdOpt-BCU-Fpga 4.0

AdOpt-BCU-Pci 4.0

AdOpt-BCU-Pio 4.0

AdOpt-BCU-Power 4.0

AdOpt-BCU-Sdram 4.0

AdOpt-BCU-Serial Jtag 4.0

AdOpt-BCU-Thermic 4.0

Layout

AdOpt-BCU-Layout TOP 4.0

AdOpt-BCU-Layout BOT 4.0

Bill of Material

AdOpt-BCU-BOM 4.0

8.4.2 AdOpt Ethernet PCI 1.1

Schematics

AdOpt-PCI 1.1

Layout

AdOpt-PCI-Layout BOT 1.1

Bill of Material

AdOpt-PCI-BOM 1.1

8.4.3 AdOpt Fastlink 2.0

Schematics

AdOpt-Fast Link 2.0

Layout

AdOpt-Fast Link-Layout BOT 2.0

Bill of Material

AdOpt-Fast Link-BOM 2.0

8.4.4 AdOpt DSP Digital Only 3.1

Schematics

AdOpt-DSP-DIGITAL ONLY- Main 3_1

AdOpt-DSP-DIGITAL ONLY- Bus Interface 3_1

AdOpt-DSP-DIGITAL ONLY- Clock 3_1

AdOpt-DSP-DIGITAL ONLY- Flash Sram Cpld 3_1

AdOpt-DSP-DIGITAL ONLY- Diagnostic 3_1

AdOpt-DSP-DIGITAL ONLY- Dsp A 3_1

AdOpt-DSP-DIGITAL ONLY- Dsp B 3_1

AdOpt-DSP-DIGITAL ONLY- Fpga 3_1

AdOpt-DSP-DIGITAL ONLY- Power 3_1

AdOpt-DSP-DIGITAL ONLY- Sdram 3_1

AdOpt-DSP-DIGITAL ONLY- Serial Jtag 3_1

Layout

AdOpt-DSP-DIGITAL ONLY- Layout TOP 3_1

AdOpt-DSP-DIGITAL ONLY- Layout BOT 3_1

Bill of Material

AdOpt-DSP-DIGITAL ONLY- BOM 3_1

8.4.5 AdOpt DSP HVC 3.2

Schematics

AdOpt-DSP-HVC- Main 3_2

AdOpt-DSP-HVC- ADC 3_2

AdOpt-DSP-HVC- Bus Interface 3_2

AdOpt-DSP-HVC- Clock 3_2

AdOpt-DSP-HVC- Coil Connector 3_2

AdOpt-DSP-HVC- Flash Sram Cpld 3_2

AdOpt-DSP-HVC- DAC 3_2

AdOpt-DSP-HVC- Diagnostic 3_2

AdOpt-DSP-HVC- Dsp A 3_2

AdOpt-DSP-HVC- Fpga 3_2

AdOpt-DSP-HVC- Power 3_2

AdOpt-DSP-HVC- Sdram 3_2

AdOpt-DSP-HVC- Serial Jtag 3_2

AdOpt-DSP-HVC- Signal Generator 3_2

Layout

AdOpt-DSP-HVC- Layout TOP 3_2

AdOpt-DSP-HVC- Layout BOT 3_2

Bill of Material

AdOpt-DSP-HVC- BOM 3_2

8.4.6 AdOpt HVC 1.0

Schematics

AdOpt-HVC 1.0

Layout

AdOpt-HVC-Layout TOP 1.0

AdOpt-HVC-Layout BOT 1.0

Bill of Material

AdOpt-HVC-BOM 1.0

8.4.7 AdOpt AIA to PIO 2.0

Schematics

AdOpt-AIA to PIO 2.0

Layout

AdOpt-AIA to PIO- Layout TOP 2_0

AdOpt-AIA to PIO- Layout BOT 2_0

Bill of Material

AdOpt-AIA to PIO-BOM 2.0

8.4.8 AdOpt PIO to DM 1.0

Schematics

AdOpt-PIO to DM 1.0

Layout

AdOpt-PIO to DM- Layout TOP 1_0

Bill of Material

AdOpt-PIO to DM-BOM 2.0

8.4.9 AdOpt Reset Board 1.0

Schematics

AdOpt-Reset Board 1.0

Bill of Material

AdOpt-Reset Board- BOM 1.0

8.4.10 AdOpt Backplane 2.1

Schematics

AdOpt-Backplane- Slot 0 2.1

AdOpt-Backplane- Connectors 2.1

Layout

AdOpt-Backplane- Layout TOP 2_1

AdOpt-Backplane- Layout BOT 2_1

Bill of Material

AdOpt-Backplane- BOM 2.1

8.5 Code reference

CodeIrig.pdf

CodeMGPDriver.pdf

CodeWIF.pdf