EUROPEAN SOUTHERN OBSERVATORY

Organisation Européenne pour des Recherches Astronomiques dans l'Hemisphère Austral

Europäische Organisation für astronomische Forschung in der südlichen Hemisphäre

ESO - EUROPEAN SOUTHERN OBSERVATORY
Karl-Schwarzschild-Straße 2 · D-85748 Garching bei München

VLT INSTRUMENTATION

PULPO-2 Manual

VLT-TRE-ESO-13630-3490
Issue 2.0
Date: 23/08/2006

Prepared by: Claudio Cumani
Christoph Geimer
Approved by: Dietrich Baade
Released by: Alan Moorwood

Change Record

Issue/Rev	Date	Section/Page affected	Reason
1.0	07/12/2004	All	First issue
2.0	23/08/2006	6.2	Error codes updated
		6.4	AE: command description improved
		6.4	CD: close shutter delay returned in microseconds
		6.4	ES: command added
		6.4	LC: number of elements in log file has been increased
		6.4	LF: command added
		6.4	OD: open shutter delay returned in microseconds
		6.4	PE: format of reply corrected
		6.4	SS: command syntax enlarged
		6.4	VI: command added
		6.4	VS: format of reply corrected
		6.5	ES: command added
		6.5	LF: command added
		6.4	SS: command syntax enlarged
		6.5	VI: command added
		6.7	Description of logging files refers also to the new LF command
		6.8	Description of Self Recovery include new command syntax
		8	"Maintenance and Troubleshooting" chapter added
		ANNEX B	Also Vacuum Sensor is used for Self Recovery

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Purpose

This document provides a description of the functionalities and the command interface of the PULPO-2 ESO standard multifunctional module.

It is intended to provide the ESO hardware and software engineers who intend to maintain or use the PULPO-2 controller with all the necessary information to:

- understand the PULPO-2 hardware setting
- operate PULPO-2 through the software command interface

Pre-requisites: knowledge of CCD camera systems operations (cooling, CCD operation conditions, shutter) is required, therefore PULPO-2 must be used by experienced engineers only.

1.2 Applicable documents

The following documents, of the exact issue shown, form a part of this document to the extent specified herein.

Reference	Document Number	Issue	Date	Title
[AD 01]	VLT-PRO-ESO-10000-0228	1.0	10/03/1993	VLT Software Programming Standards
[AD 02]	VLT-PLA-ESO-00000-0006	2.0	21/05/1992	VLT Software Management Plan

1.3 Reference documents

The following documents are referenced in this document.

Reference	Document Number	Issue	Date	Title
[RD 01]	INS-01/0044			Pulpo upgrade statement of work
[RD 02]	INS-02/0018			Alternatives for Pulpo upgrade
[RD 03]	VLT-CRE-ESO-13640-0003			Pulpo change request

1.4 Abbreviations and acronyms

This document employs several abbreviations and acronyms to refer concisely to an item, after it has been introduced. The following list is aimed to help the reader in recalling the extended meaning of each short expression:

BIOS	Basic Input/Output System
CCD	Charge-Coupled Devices
CPU	Central Processing Unit
DAC	Digital to Analog Converter
DMA	Direct Memory Access
DN	Data Number
DRAM	Dynamic Random Access Memory
ESO	European Southern Observatory

FFS	Flash File System
FIERA	Fast Imager Electronic Readout assembly
GPIO	General-Purpose I/O
I/O	Input/Output
IDE	Integrated Device (or Drive) Electronics
ISA	Industry Standard Architecture
LCU	Local Control Unit
LCD	Liquid Crystal Display
LED	Light-Emitting Diode
MOSFET	Metal Oxide Semiconductor Field-Effect Transistor
MTBF	Mean Time Between Failures
NAND	Negated AND (logic)
NOR	Negated OR (logic)
OS	Operating System
PC	Printed Circuit
PCMCIA	Personal Computer Memory Card International Association (also: People Can't Memorize Computer Industry Acronyms)
PID	Proportional/Integral/Derivative controller
PLD	Programmable Logic Device
PWM	Pulse Width Modulator
RAM	Random Access Memory
RTC	Real Time Clock
SLCU	Sparc-based Local Control Unit
SMD	Surface Mounted Device
UART	Universal Asynchronous Receiver Transmitter
VLT	Very Large Telescope
XTAL	Crystal (frequency reference)

1.5 Glossary

No special definition is introduced in this manual.

1.6 Stylistic conventions

The following styles are used:

bold
: in the text, for commands, filenames, pre/suffixes as they have to be typed.

italic
: in the text, for parts that have to be substituted with the real content before typing.

teletype
: for examples.

<name>
: in the examples, for parts that have to be substituted with the real content before typing.

bold and *italic* are also used to highlight words.

1.7 Naming conventions

This implementation follows the naming conventions as outlined in [AD 01].

1.8 Acknowledgments

Special thanks to Rolf Gerdes, who contributed to the development of the PULPO-2 project and to the first draft of some of the chapters of this manual.

2 OVERVIEW

PULPO-2 is an ESO standard multifunctional module designed for use with cooling systems and shutters of ESO CCD detector systems. It is able to:

- Read up to 29 temperature sensors, PT100 type,
- Control 8 heaters circuits of up to 7 Watts each,
- Read out of a vacuum sensor,
- Interface in a flexible way to a large variety of shutters
- Centralize alarms from vacuum and temperature sensors,
- Display parameters for easy and quick parameter check-out,
- Log sensor values on internal flash disk for later recovery,
- Provide a serial and an ethernet interface.

The standard PULPO-2 consists of a CPU board, a peripheral board, the user interface (display and buttons), the housing and the cabling inside (figure 1).
The CPU board controls the display, the input keyboard, the serial link and the Peripheral board.
The peripheral board is the interface to the detector head, sensing the temperature and controlling the heaters and the shutter.
With a different housing up to 3 peripheral boards could be used.
The interface between the CPU board and the Peripheral board is via an ISA bus.

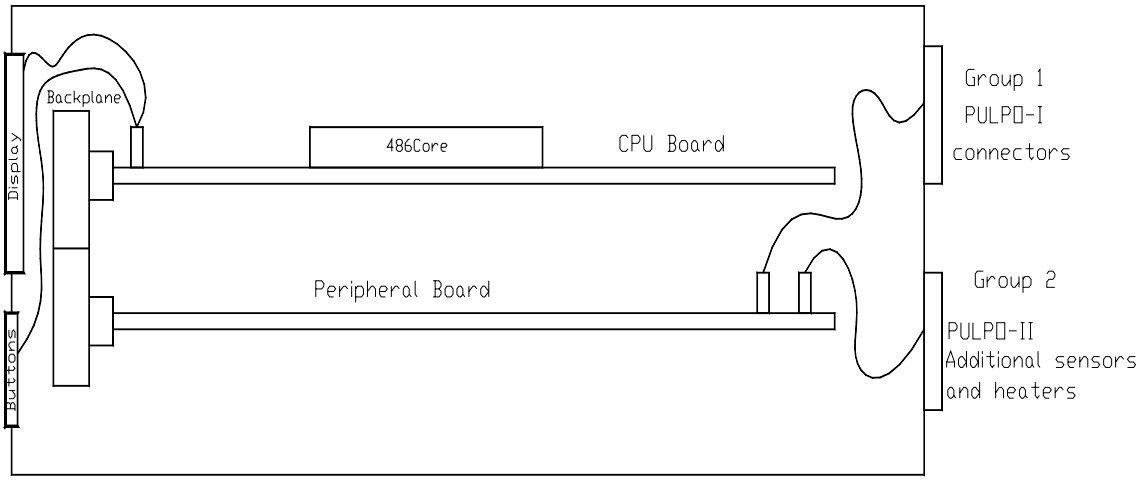


Fig. 1:Overview of PULPO-II

The CPU board is based on a credit-card-size embedded PC mounted on a board containing the power supply and the connectors to the display, input keyboard and the serial communication to the SLCU. This embedded PC is called 486Core in this document.

As can be seen on figure 1, the connectors on the Peripheral board to interface to the Detector Head are split in two groups. The first group of connectors offers the same interface as the current PULPO and the second group presents the additional sensors and heater control lines. By doing this, the current PULPO can be replaced by the new one without changing the cabling between PULPO and the dewar. This idea will help us in maintaining the instruments already deployed in case we need to substitute the current PULPO by an upgraded one.

3 HOUSING

The standard housing consists of off-the-shelf aluminum profiles (kind of profiles from the company MTS Systemtechnik) and aluminum covers. This housing combines a good trade-off among easiness, robustness and accessibility to the internal parts.

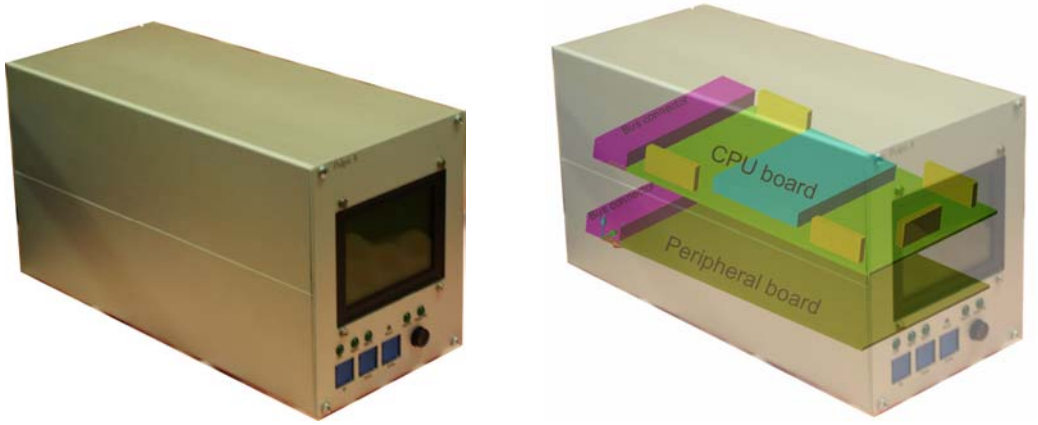


Fig 1 - PULPO-II

With a different - wider - housing, up to 3 peripheral boards could be used.

4 CPU BOARD

4.1 Functional overview

The PULPO-2 CPU board offers the following functionalities:

- Non-volatile FLASH Disk for OS, configuration and user data ,
- ISA-Bus with 16-bit Data, 16-bit address and control lines,
- Power supply for on-board peripherals and the PULPO-2 Peripheral board,
- Up to 3 optoisolated serial ports and one serial fiber connection to the SLCU,
- Ethernet connection for on-site debugging and fast log data download,
- Interface to the user through the front panel keyboard and display.

The PULPO-2 CPU board has a standard Eurocard size (160x100 mm).
The core component is a tiny "486Core" PC card module (75x54mm) from Compulab[1], which hosts a complete PC with a 486 processor, on-board memory, FLASH disk and all standard PC peripherals, e.g. serial ports, parallel port, IDE Interface, keyboard controller, PCMCIA controller, Ethernet controller and external interface (I/O ports, VL bus and ISA bus). For a description of the 486Core see the dedicated chapter below.

The backplane connector is a 96-pin DIN41612. This connector provides power and ISA bus signals to the Peripheral board.

The CPU board also contains the power supply for the 486Core and the Peripheral board. It is supplied with 24V from the FIERA power supply. The required voltages for the 486Core (5V/3.3V) and the peripheral board (+5V, +3.3V,+15V,-15V,+24V) are generated on the CPU board by means of DC/DC converters.

The control connection to the user (display and keyboard) is also performed through the CPU board. The display is a Varitronix MGLS12864T-HT-LED03 with a display size of 128x64 pixel and LED background illumination. It features an on-board Toshiba controller (T6963), 8K-ROM with predefined character sizes of 8x8 or 8x6 pixels, and 8K RAM for user defined characters. It can operate in either text mode, graphic mode or mixed mode.
The user input can be performed via buttons on the front panel. These buttons are read by the 486Core via the GPIO (General Purpose I/O) lines, so there is no need for additional circuitry.

The connection to the LCD display is via the parallel port interface due to the ease of use and availability of the software device driver to control it.

[1] See www.compulab.co.il

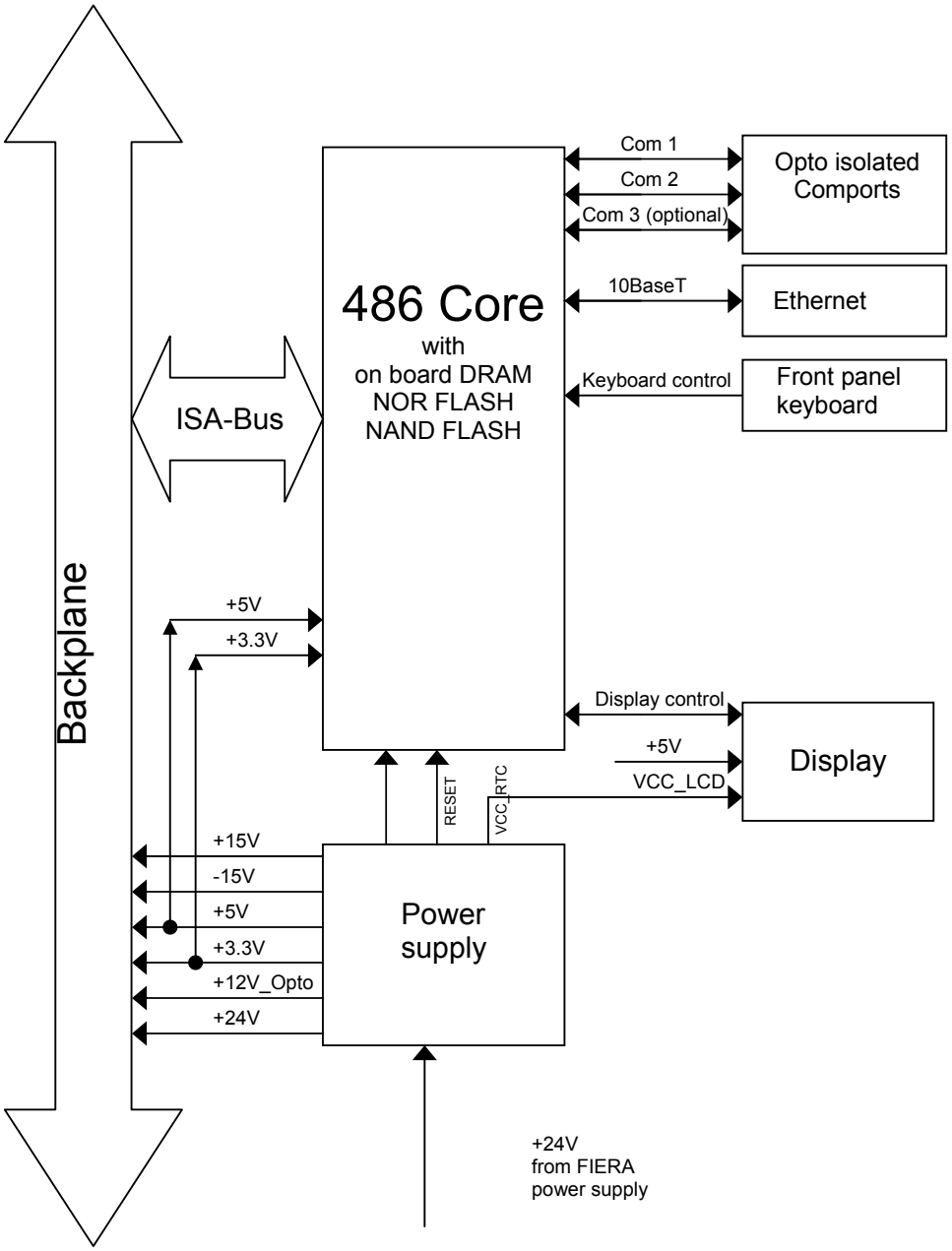


Fig.2: Overview of the PULPO-II CPU board

4.2 Power supply

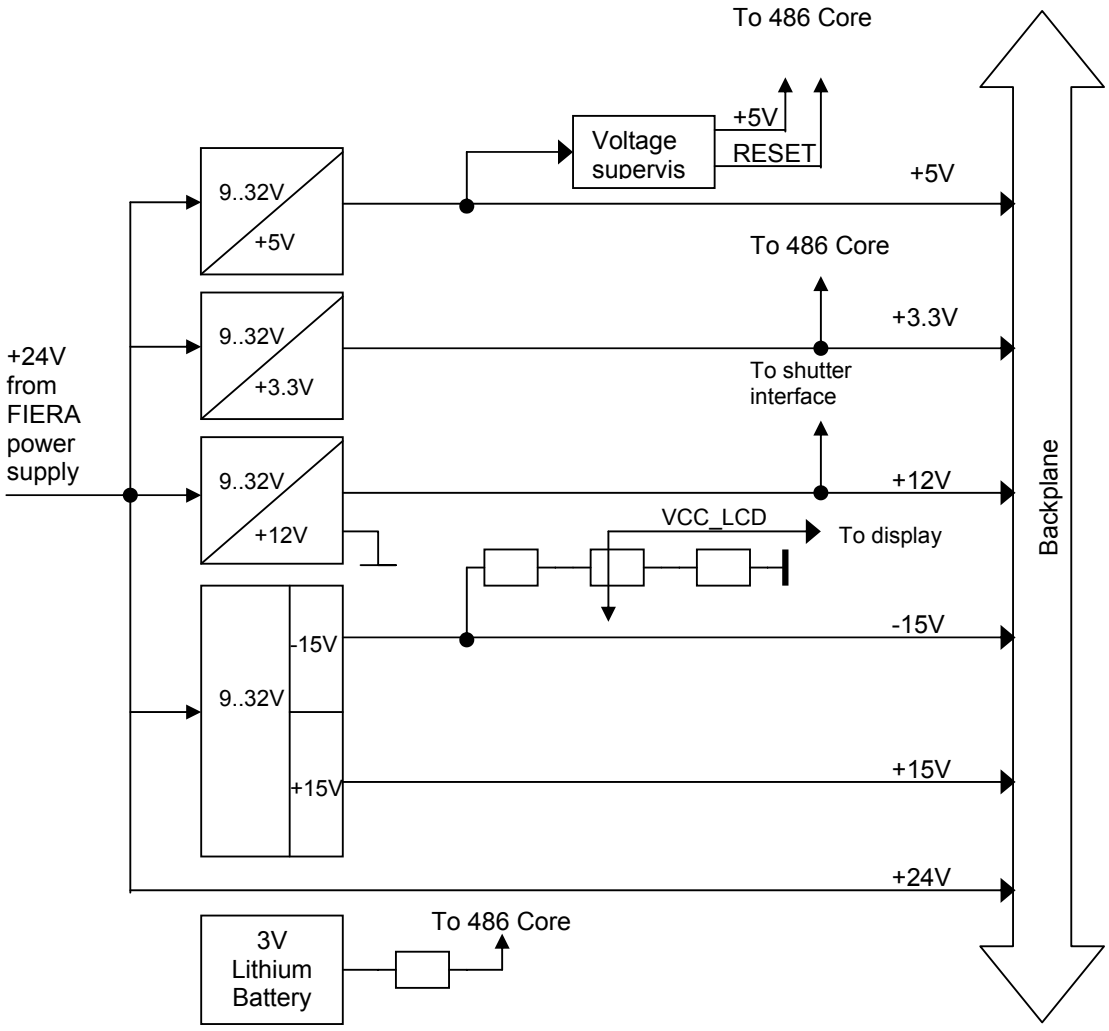


Fig 3: Block diagram of the power supply

All required voltages are generated on the CPU board. The main power supply for the board is a standard FIERA 24V Kniel power supply mounted on the power supply of the Detector Head. All other voltages are derived from this voltage using Deutronic DC/DC converters.

The maximum currents can be seen in the table below.

Voltage	max.Current	Used on :
+5V	2A	CPU Board +peripheral board
+3.3V	2A	CPU Board +peripheral board
+12V	1.6A	CPU Board +peripheral board
+15V	100mA	peripheral board
-15V	100mA	peripheral board + Display

Each voltage has auto resetting SMD fuses. Standard bypassing plus additional PI filters are used to smoothen the supply voltages. A voltage supervisor, MAX 814, on the 5V supply is used to generate a reset on power-up and in case of a power failure.The MAX814 also provides the option of a manual reset via a button.The Peripheral board is reset via the RSTDRV signal on the ISA bus and generated by the 486Core.

The display requires an adjustable contrast voltage (from -9 to -12V). Due to the low current (<4mA) it can be derived from the -15V by means of a simple voltage divider.

The +12V is used to provide power to the opto I/O's on the CPU board and the peripheral board. On the CPU board it supplies the power to the optocouplers for the serial ports and on the peripheral board it provides power to the shutter interface and the user I/O lines.

A 3V lithium battery is used to keep the RTC (real time clock) running in case of a power failure.

Note: *The battery only serves as a backup source for the RTC, when PULPO-2 is off. All configuration data (set points, heater assignment, etc.) is stored in non-volatile FLASH memory, and will be kept even if the battery fails.*

4.3 The 486Core

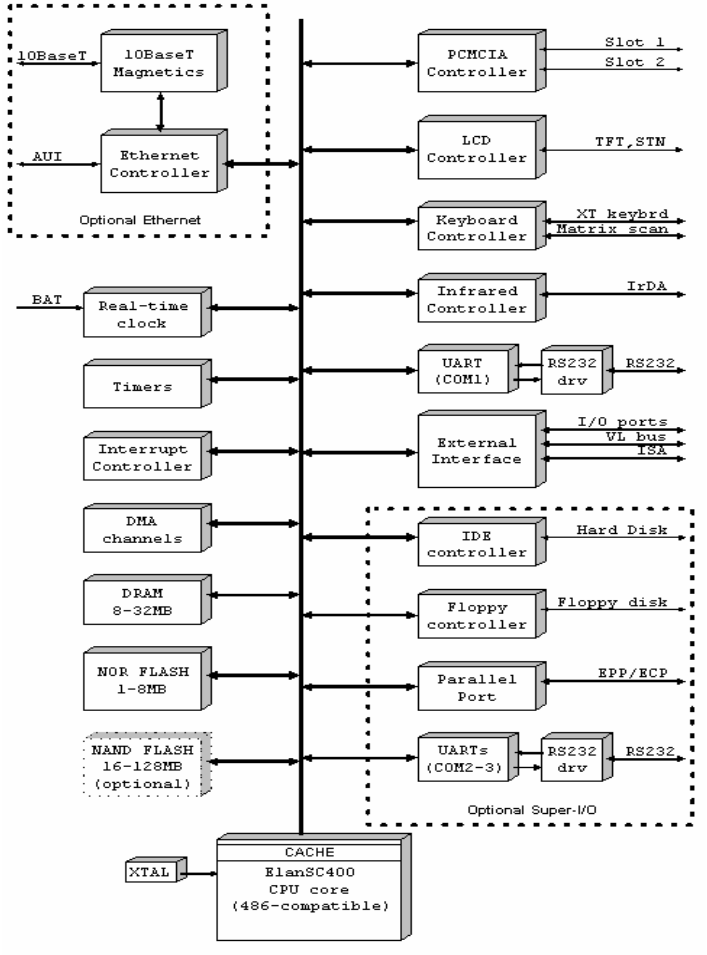


Fig. 4: Block diagram of the 486Core

The 486Core module is a tiny PC Card module (75 x 54 mm). It can be equipped with different sizes of FLASH memory, RAM and different configuration depending on the target application. Table 1 summarises the most important features.

Feature	Description	CPU Board	Remarks
Processor	486 running at 33, 66, or 100Mhz	486 at 66Mhz	
DRAM	8-32 MB	32MB	
Program NOR FLASH	1-8MB	4MB	
NAND Flash Disk	1-136MB	16MB	
LCD Graphics controller		Parallel port device driver or raw control of the parallel port lines	Might be used for LCD. Alternatively the parallel port or GPIO can be used.
PCMCIA controller	Available	No	Control lines can be used for other purposes
PC/AT systemlogic and essential peripherals	Floppy Disk controller, Parallel port, 32 GPIO's, IDE, Keyboard controller	Parallel port, GPIO's.	Either keyboard controller or GPIO's will be used for keyboard
Power consumption	dual supply 3.3V/5V 30mW/Mhz		
MTBF	100.000 hours		
Network interface	Ethernet 10BaseT with transformer and EMI Filter		

Table 1: 486 Core features

About 300 KB of the NOR-FLASH is used for BIOS, the remaining part is used for disk emulation. Flash ROM guarantees a minimum of 1,000,000 write operations per sector.

For applications requiring large non-volatile on-board storage, the 486CORE can be assembled with additional NAND Flash. NAND Flash is a block device optimized for block read and write operations rather than random access. It is used to emulate FFS (Flash File System, any piece of software that allows flash to be used similarly to a disk drive). It is seen by the operating system as a regular hard disk. NAND Flash is available with size of 16, 32, 64 or 128 Mbytes. The 486CORE is designed for upward NAND Flash compatibility with future and larger devices. NAND Flash

guarantees a minimum of 1,000,000 block write operations and 10-year data retention.

Most of the control lines on the 486Core are multiplexed between different functions. There is always a trade-off between them e.g. enabling the keyboard controller limits the memory usage to 16MB.
Three GPIO's (General Purpose I/O) would be needed for the input buttons and one for the control of display illumination.

The 486Core offers 6 GPIO's which are not multiplexed, thus ensuring that the full functionality will be maintained. More I/O lines can be made available by disabling the PCMCIA slot. There are no plans to use the PCMCIA interface for the CPU board.

5 PERIPHERAL BOARD

5.1 Functions, board format and connectors

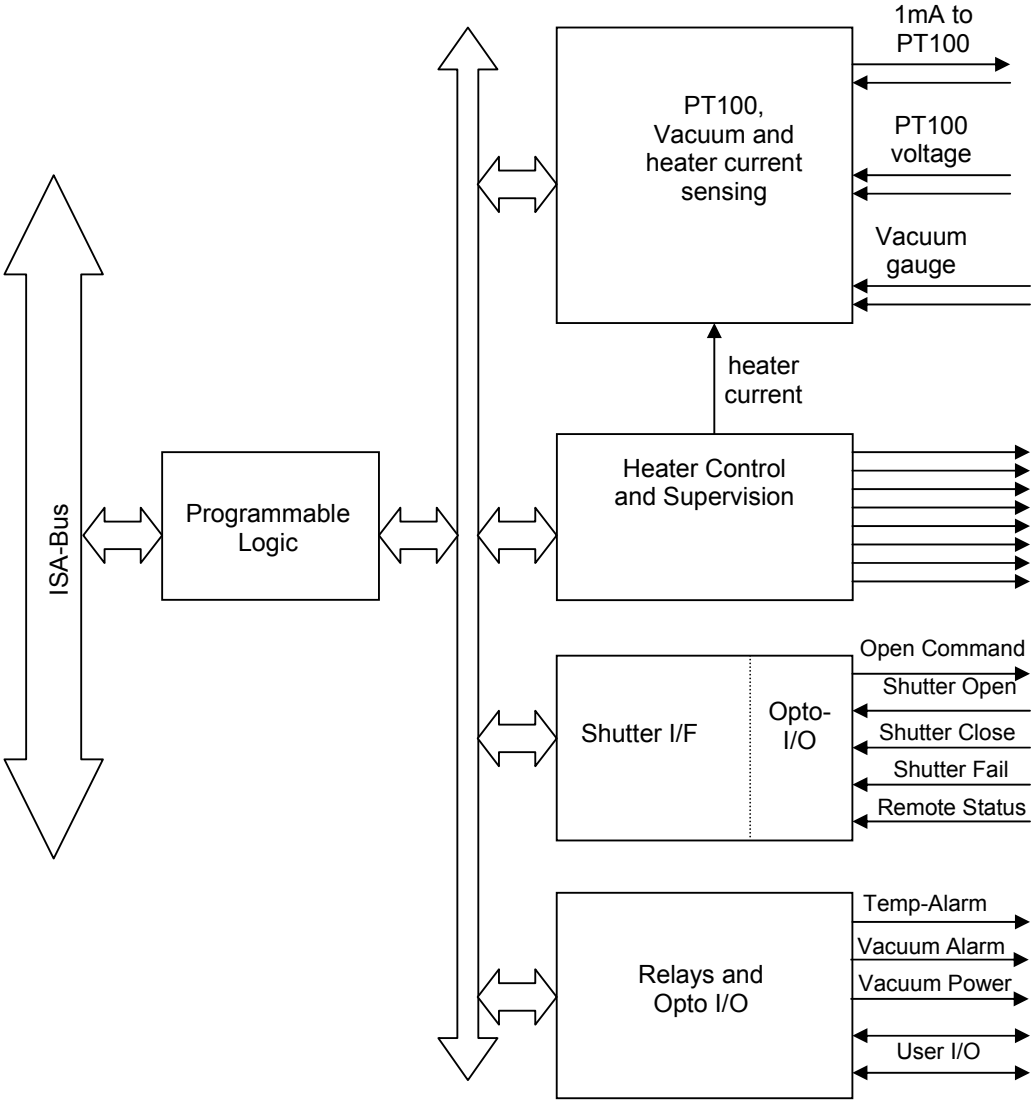


Fig. 5: Overview of PULPO-II peripheral board

The PULPO-2 peripheral board offers the following functionalities:

- Possibility to connect up to 32 sensors:
  - one vacuum sensor (type Balzers or Edwards)
  - one reference sensor (connected to an internal, 100 Ω precision calibration resistor)
  - one heater current sensor (to show the current used by the heaters)
  - up to 29 PT100 temperature sensors
- Heater control of up to 8 heaters (the overall current cannot exceed 3A)
- Safety options for heater control: Software triggered watchdog (max. 1 sec. trigger interval), overcurrent sensing in hardware, current sensing in software
- opto-coupled shutter interface with 1ms resolution, 61ns accuracy, hardware controlled exposure time
- 2 relays for alarm signaling (vacuum and temperature)
- 1 relay to switch vacuum power supply
- 2 additional spare optocoupled outputs and 2 inputs (on the shutter connector)
- current output for LED driving

The board has the standard 3U-Eurocard format (160 x 100 mm). A 96-pin DIN 41612 VG-connector is used for the connection to the CPU-module and for supplying power to the peripheral board.

The power supply is +24VDC for the heaters (@ 3A), +/-15VDC (@ 150 mA) for the analog sensing, +5VDC (@ 350 mA) for the digital electronics, +12V (@100mA) for optocouplers.

The ground for +24V is connected to analog ground directly at the VG-connector. The analog ground will be connected to digital ground at the ADC.

Flat ribbon cables connect PT100s, heaters, etc. internally to the PULPO housing. The Samtec "FFSD" series with a pitch of 1.27 mm are used. Two pins per connector are employed for the heater signals to distribute the current.

5.2 Bus interface

The bus interface is an ISA (Industry Standard Architecture) interface exclusively using 16-bit I/O-cycles. The following bus signals are used to decode valid accesses from the CPU:

Signal	Input/Output	Full Signal Name	Usage
AEN	Input	Address Enable	inactive for valid access
BALE	Input	Buffered Address Latch Enable	Active at start of cycle for valid access
Clk	Input	Clock	14.385 MHz System Clock
SA0..15	Input	System Address	16-bit address
SD0..15	Bidirectional	System Data	16-bit data
/IOR	Input	I/O Read	Read cycle
/IOW	Input	I/O Write	Write cycle
I/OCS16	Tristate Output	I/O chip select 16	Indicate 16bit I/O cycle possible
/0WS	Tristate Output	0 wait state	Indicate zero wait state cycle
I/OCHRDY	Tristate Output	I/O Channel Ready	Indicate cycle completed
RESETDRV	Input	Reset	Power up or system reset
IRQ5, IRQ13	Tristate Output	Interrupt Request	IRQ5 used for optional interrupt signalling

The signal assignment on the backplane follows the one that is specified for the PULPO-2 CPU-board.

Zero wait-state, 16-bit I/O cycles are implemented on the ISA-interface. No other accesses are supported. The registers are normally polled, but interrupts are available for certain bits. IRQ5 (normally used) and IRQ13 (supplementary) are connected to the PLD (Programmable Logic Device). The interrupt is generated, when a certain condition is encountered (e.g. watchdog not triggered) and the corresponding interrupt enable bit is set (watchdog interrupt enable). The interrupt is released, when the bit is polled (i.e. read cycle to watchdog status bit).

The board identification number can be set using two solder jumpers, i.e. there are four possible identification numbers (the identification number 00 should not be used, since this combination results in I/O-port addresses of 0-31, which are already in use by the CPU).

The system clock is 14.385 Mhz. The timing of the shutter is derived from a local oscillator and is therefore independent of the system clock.

There will not be any termination or “quiet state biasing” of signals. This has to be done on the CPU board (e.g. pulling AEN low or pulling /0WS high).

5.3 Sensors

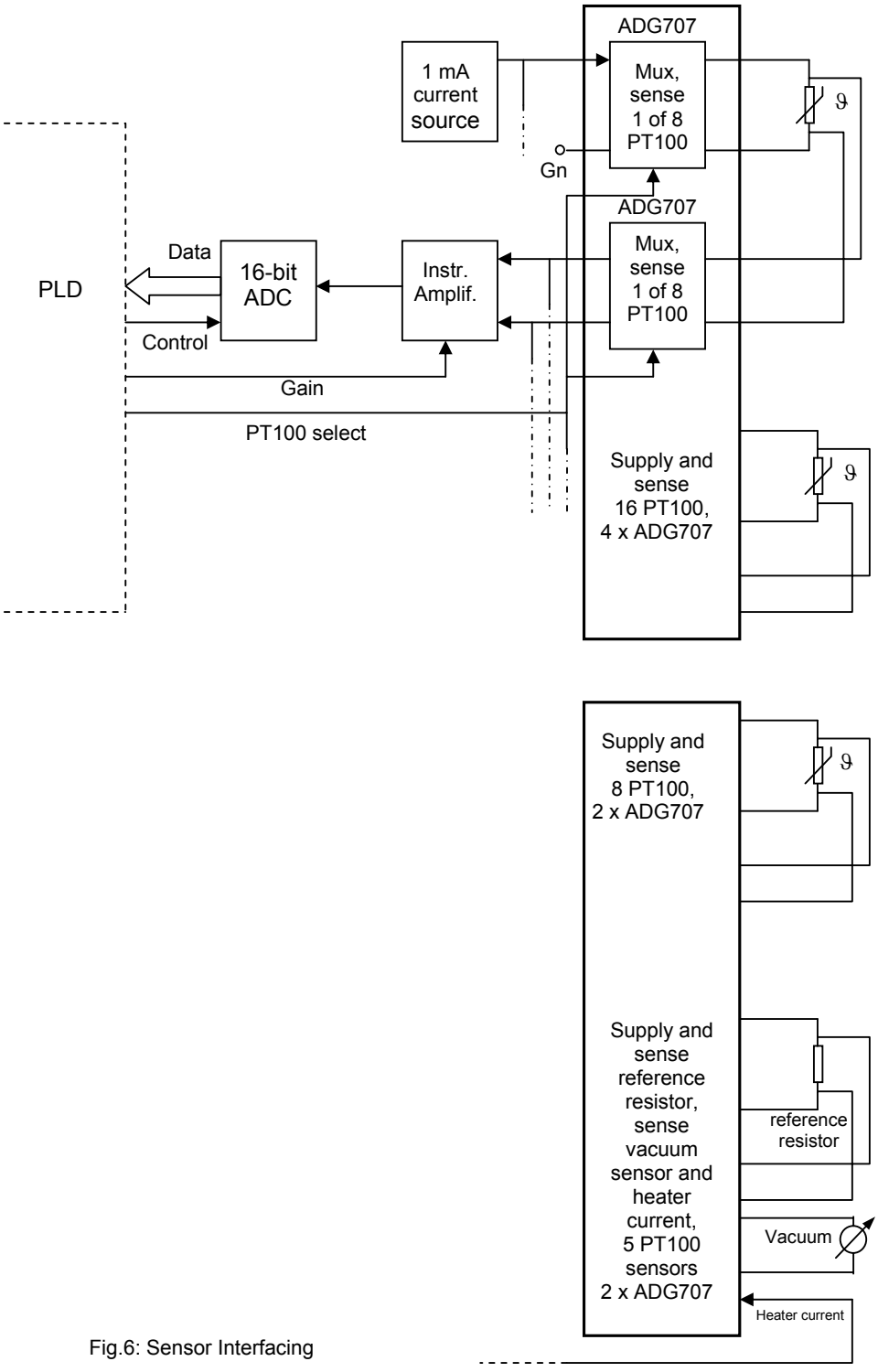


Fig.6: Sensor Interfacing

As shown in Fig.6, four ADG707-differential multiplexers connect one of up to 29 PT100 sensors to a 0,5 mA current source. The other terminal of the selected PT100 is simultaneously switched to ground. Four additional ADG707-differential multiplexers connect one of the sense conductors of the selected PT100, the reference resistor or the vacuum gauge or the heater current sense output to an analog signal chain. The gain in the signal chain can be switched to accommodate the different output levels of the PT100s and the vacuum gauge.

The analog chain uses the PGA205 and the INA118 instrumentation amplifiers. An AD7663 analog-to-digital converter is used (parallel 16-bit output). The input range of the AD7663 is 0..10V. The AD7663 is supplied with +5VDC. The +5V is derived from the +15V analog supply voltage by means of a linear regulator.

The heater current sensing is effected using a current sense amplifier, which converts the current to an equivalent voltage. This voltage is supplied to a multiplexer for sensing.

The ADC range and resolution are sufficient to detect whether or not a PT100-connection is broken. If the sensed voltage exceeds the nominal range of temperatures, one of the wires that are used to connect the PT100 must be broken.

In the following table the sensor identification number and description is summarized:

Sensor number	Description
1...5	internal PT100 Temperature sensor
6	external PT100 connector (FIERA box)
7	reference sensor (connected to internal, 100 Ω precision calibration resistor)
8	vacuum sensor
9	heater current sensor
10...32	internal PT100 Temperature sensor

Summary of sensor identification numbers and description

External multiplexers can be connected to the ADG707-differential multiplexers, in this way the number of connected PT100 sensors can be increased.
So far, this possibility has been used only in the case of OmegaCam.
In the following table the OmegaCam identification number and description is summarized:

Sensor number	Description
1...4	to external multiplexers
5	internal PT100 Temperature sensor
6	external PT100 connector
7	reference sensor (connected to internal, 100 Ω precision calibration resistor)
8	vacuum sensor
9	heater current sensor
10...32	internal PT100 Temperature sensor
111...118	external PT100 via external multiplexer 1
121...128	external PT100 via external multiplexer 1
131...138	external PT100 via external multiplexer 1
211...218	external PT100 via external multiplexer 2
221...228	external PT100 via external multiplexer 2
231...238	external PT100 via external multiplexer 2
311...318	external PT100 via external multiplexer 3
321...328	external PT100 via external multiplexer 3
331...338	external PT100 via external multiplexer 3
411...418	external PT100 via external multiplexer 4
421...428	external PT100 via external multiplexer 4
431...438	external PT100 via external multiplexer 4

Summary of sensor identification numbers and description in the OmegaCam

5.4 Heater control

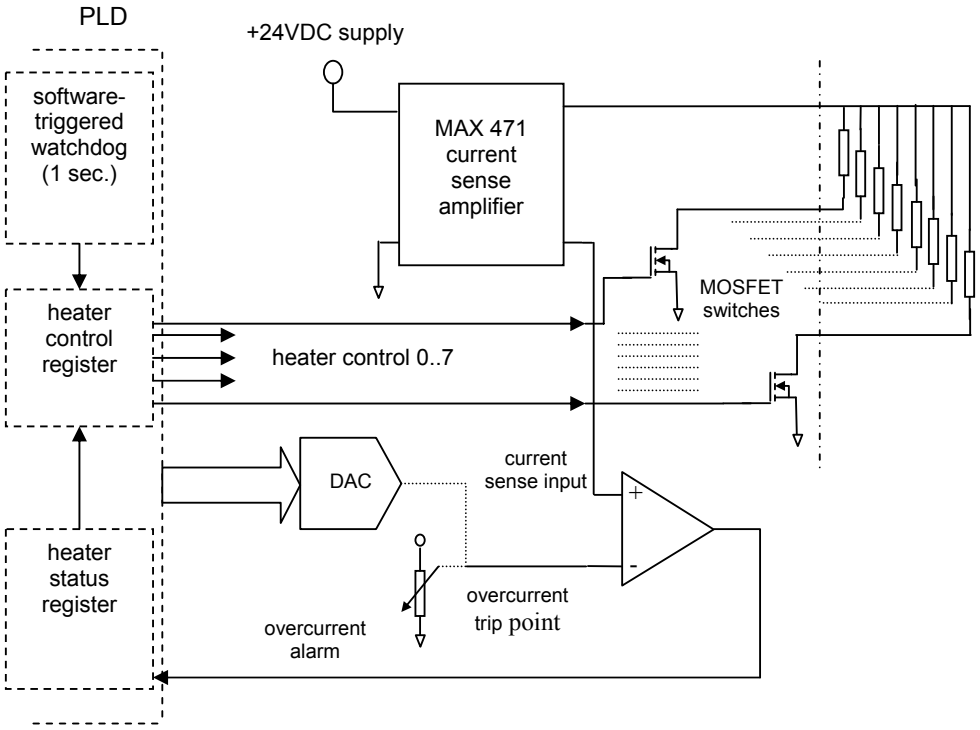


Fig. 7: heater control

With the PULPO-2 Peripheral board it is possible to connect up to 8 heaters. These heaters are driven with 24V in PWM-Mode. The overall current for all heaters must not exceed 3A. A MOSFET switch is connected in series to each heater and switches the current (320 mA) through the heater.

The heaters can be activated individually. The total heater current is sensed and the overcurrent trip point is set for the total heater current.
The overcurrent trip point can be set by software (via the second channel of the DAC for the bias LED) or alternatively using a potentiometer. A 2-channel 12-bit serial DAC (MAX532) is used for the circuit to drive a bias LED with a constant current.

Safety mechanisms to avoid overcurrent damaging the CCD:

- Software-triggered watchdog (1 second interval for triggering) in the PLD
- Current Sensing and possibility to read-back current value by software (see section 4.2)

- Comparison of heater current with trip point. This is implemented in hardware (see Fig. 7), whereby the trip point can be fixed (potentiometer) or set by software (DAC). The selection between hardware and software settings is made by means of solder jumper on the peripheral board.

5.5 Shutter control

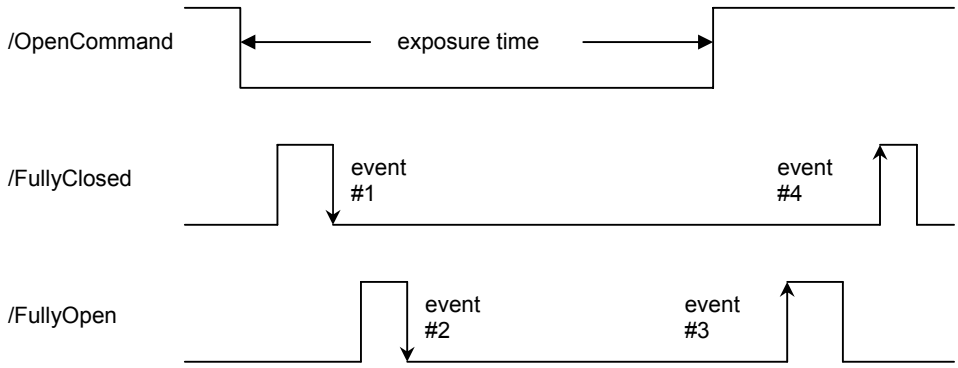


Fig. 8: shutter control and status signals

The shutter timing of Fig.8 is one possible example. Due to a programmable shutter interface different shutter configurations can be implemented. The programmability of the interface allows for different signal polarities, and allows for the measurement of the delays of the shutter mechanics with respect to the exact time, at which a start or end of an exposure is commanded.

The exposure time is controlled by the PLD on the PULPO-2 peripheral board. The timing control is based on the local 16.000 MHz clock and hence accurate to within +/- 62.5 ns. The local clock is divided down to a 1ms-clock and this slow clock is then used for the exposure time counting. The resolution of the exposure time setting is therefore 1ms. Because a 24-bit counter is used, a maximum of 4.6 hours (16.777 s) exposure time can be set.

The shutter control on the PULPO-2 peripheral board uses the following shutter signals:

Signal	Input/Output	Function
/OpenCommand	Output	shutter opens when signal is active (active low or high programmable)
/FullyClosed	Input	pulses inactive, when shutter leaves and enters fully closed status (acc. to configuration of Fig. 8)
/FullyOpen	Input	pulses inactive, when shutter leaves and enters fully open status (acc. to configuration of Fig. 8)
ShutterFail	Input	shutter failed
RemoteStatus	Input	shutter dependent

All shutter I/F-signals are opto-isolated to avoid ground loops.

The exposure command (/OpenCommand = 0) is issued to the shutter when the StartExposure-bit in the shutter control register is set. A counter is initialized with the combined value of the shutter exposure time low word and high word registers and downcounting is started. Moreover, the counters for the Event1- and Event2-delays are reset and start incrementing. The event1 counter is stopped, when event1 occurs. Event1 is defined as a transition of the /FullyClosed-bit. Whether the leading or trailing edge of the /FullyClosed-bit is taken as event1, is defined by the Event1ActiveEdge-bit in the shutter status/control register (see chapter “Software Interface”). The event2-delay is similarly counted (see Fig.8).
The exposure is finished (/OpenCommand = 1) when the shutter control register count reaches zero. Event3 and event4-counters are reset and start incrementing at this point. The counters are stopped on the occurrence of event3 and event4, respectively. The event-counters comprise 16-bits, which allows for delays up to 65 seconds.

5.6 Other functions

Other functions on the PULPO-2 board are four alarm outputs (two temperature alarms and two vacuum alarms) as well as two user-defined outputs and inputs. The alarm signals are used to close two normally-open relays. The user defined outputs and inputs are opto-isolated.

An additional relay can be used to switch the power of the vacuum sensor.

The Peripheral board also controls two LEDs on the front panel. One LED lights up when one heater is on and the other when the shutter is open.

5.7 Register interface

There are 14 registers on the PULPO-2 peripheral board, which are accessed using the following addresses:

Register	Binary address, SA15..0	Access type
Sensor Control/Status Register	ID1 ID0 X X X X X X X X X 1 0 0 0	r/w
Sensor Data Register	ID1 ID0 X X X X X X X X X 0 1 0 1	ro
Heater Control Register	ID1 ID0 X X X X X X X X X 1 1 0 1	wo
Heater Supervision Register	ID1 ID0 X X X X X X X X X 1 0 0 1	r/w
Serial DAC Control/Status Reg	ID1 ID0 X X X X X X X X X 1 1 0 0	wo
Serial DAC Data Register	ID1 ID0 X X X X X X X X X 1 0 1 1	r/w
Shutter Control/Status Register	ID1 ID0 X X X X X X X X X 0 1 1 0	r/w
Shutter Exposure Time High Word Register	ID1 ID0 X X X X X X X X X 0 1 1 1	r/w
Shutter Exposure Time Low Word Register	ID1 ID0 X X X X X X X X X 0 0 0 0	r/w
Event1 counter	ID1 ID0 X X X X X X X X X 0 0 0 1	ro
Event2 counter	ID1 ID0 X X X X X X X X X 0 0 1 0	ro
Event3 counter	ID1 ID0 X X X X X X X X X 0 0 1 1	ro
Event4 counter	ID1 ID0 X X X X X X X X X 0 1 0 0	ro
Misc I/O Register	ID1 ID0 X X X X X X X X X 1 0 1 0	r/w

ID1..0: board identification (jumper setting)
X: don´t care
r/w: read/write-access
ro: read only access
wo: write only access

Sensor Control/Status Register

bit	write access	read access
15..8	not used	
7	ADC start of conversion	ADC Busy
6	PGA gain control	
5	enable sensor/mux	
4..0	sensor channel	

Bits 15..8 are not used, i.e. a read cycle of the sensor control/status register will yield meaningless results in these bit positions. The bits written to these bit positions will not be registered.
Writing a 1 to bit 7 will start a conversion of the ADC. A read cycle will yield the ADC status in bit 7. Bit 6 is used to set the gain of the instrumentation amplifier / programmable gain amplifier (PGA). One of 32 sensor channels can be selected with bits 4..0 and the selected channel is connected to the instrumentation amplifier by setting bit 5 of the register.

The channel assignment of the sensors is as follows:

sensor channel (bits 4..0 of sensor control/status register)	sensor
1...5	PT100 in cryostat
6	PT100 in FIERA
7	100 Ω reference resistor
8	vacuum sensor
9	heater current
10...32	optional PT100 sensors

Sensor Data Register

bit	write access	read access
15...0	no result	last converted sensor value

Heater Control Register

bit	write access	read access
15...9	unused	
8	heater enable	not used
7	heater select 7	not used
6	heater select 6	not used
5	heater select 5	not used
4	heater select 4	not used
3	heater select 3	not used
2	heater select 2	not used
1	heater select 1	not used
0	heater select 0	not used

Every heater can be individually enabled or disabled independent of all other heaters. A heater is switched on, if the heater select bit for the specific heater is set and the heater enable bit is set as well.

Heater Supervision Register

bit	write access	read access
15...5	not used	
4	watchdog interrupt enable	not used
3	overcurrent interrupt enable	not used
2	watchdog trigger	not used
1	watchdog reset	watchdog status
0	overcurrent reset	overcurrent status

The overcurrent status bit is set as soon as the overcurrent condition is encountered. The bit can only be reset by a global board reset (RESETDRV-Line of the ISA-Bus) or writing a 1 to the overcurrent reset bit.
The same applies for the watchdog status bit. This bit is set, when the watchdog trigger bit is not written to in regular intervals of <1 second. Once set, the bit can only be reset by writing a 1 to the watchdog reset bit.
Both status bits can be used to generate interrupts.

Serial DAC Control Register

bit	write access	read access
15...2	not used	
1	Chip Select DAC	not used
0	Load DAC	not used

The Chip Select DAC and the Load DAC bits connect to the CSDAC and LdDAC inputs of the serial DAC.
Provided the jumper on the board is set to programmable overcurrent, the overcurrent trip point can be set in steps of 0.73 mA within a range of 0 mA to 3 A, i.e. one increment of the overcurrent trip point register (one DN, data number) corresponds to 0.73 mA. The overall current cannot exceed.
The bias LED current is also determined by the programming of the serial DAC, second channel.

Serial DAC Data Register

bit	write access	read access
15...1	not used	
0	serial data in	serial data out

The SClk signal is automatically generated with the proper timing. For details concerning the sequence of data bits, see data sheet MAX532.

Shutter Status/Control Register

bit	write access	read access
15	StartExposure	not used
14	AbortExposure	not used
13	PauseExposure	not used
12	EndofExposure Interrupt Enable	EndofExposure
11	DarkExposure	
10	OpenCommandActHi	
9	RemoteStatus Interrupt Enable	RemoteStatus Line
8	ShutterFail Interrupt Enable	ShutterFail Line
7	Event4 is a rising edge	Event4 counter overrun
6	Event3 is a rising edge	Event3 counter overrun
5	Event2 is a rising edge	Event2 counter overrun
4	Event1 is a rising edge	Event1 counter overrun
3	Event4 Interrupt Enable	Event4 encountered
2	Event3 Interrupt Enable	Event3 encountered
1	Event2 Interrupt Enable	Event2 encountered
0	Event1 Interrupt Enable	Event1 encountered

The status of the RemoteStatus and ShutterFail-signals is directly reflected in bits 9 and 8 of the shutter status register. The readable bits 0..7 are used to signal an overrun of the event-counters and the occurrence of an event. Bit 12 signals that the exposure time counter has counted down to zero. These bits 0..7 and 12 are reset, when a shutter exposure is started.
In order to determine which interrupt source generated an interrupt, the status register has to be read. If the interrupt is generated by a pulse on the RemoteStatus or ShutterFail-line, the external hardware has to ensure that these lines are asserted for a long enough period of time (interrupt latency + saftey margin), because otherwise the interrupt source cannot be found out by software.
Bits 7-4 determine whether the respective event is considered to be triggered on a falling (when bit=0) or a rising edge (bit=1). Each event can trigger an interrupt, if the corresponding Event Interrupt Enable bit is set. The interrupt source can be determined from the “Event encountered”-bits of the Shutter Status Register. The “Event encountered” bits are reset, when the next exposure is started by setting the StartExposure bit in the ShutterCtrlReg.

An exposure is started synchronous with an internal 1ms clock, immediately after the StartExposure-bit is set. The exposure can be aborted. Aborting an exposure has the same effect as if the shutter exposure time counter reaches zero, i.e. the /OpenCommand-signal is reset. The only difference is that the EndofExposure-bit in the Shutter Status/Control-Register is not set.
The setting of the PauseExposure-bit simply causes the shutter exposure time counter to stop decrementing and keep the current value, until the PauseExposure-bit is reset again.
The polarity of the OpenCommand-output can be set using the OpenCommandActHi-bit. If the DarkExposure-bit is set, the OpenCommand-line is kept in its inactive state for the duration of the exposure, i.e. the shutter is not opened.

Shutter Exposure Time High Word Register

bit	write access	read access
15...8	not used	
7...0	bits 23..16 of exposure time	

The lowest 8 bits are used to set the highest byte of the shutter exposure time counter.

Shutter Exposure Time Low Word Register

bit	write access	read access
15...0	bits 15..0 of exposure time	

Whenever the StartExposure-bit in the shutter control/status register is set, the 24-bit shutter exposure time counter is loaded with the composite contents of bits 7..0 from the shutter exposure time high word register and bits 15..0 of the shutter exposure time low word register. The /OpenCommand-signal is immediately asserted and the shutter exposure time counter starts decrementing. The /OpenCommand is asserted for the composite content of the exposure time registers + 1 in milliseconds.
Moreover, this register is available to test read/write accesses with 16 bit data (i.e. to test whether the board is present, test whether all lines are operative, etc.).

Eventx Counter Register (x=1..4)

bit	write access	read access
15...12	not used	
11...0	not used	event counter value

Misc I/O-Register

bit	write access	read access
15	MuxPower1	not used
14	MuxPower0	not used
13...8	MuxChannel[5..0]	not used
7...6	not used	
5	LEDEnable	not used
4	TemperatureAlarm	not used
3	VacuumAlarm	not used
2	VaccuumPower	not used
1	OptoOut1	OptoIn1
0	OptoOut0	OptoIn0

Bit 5 is used to enable two LED outputs. One LED output can be connected to an LED to signal that at least one of the heaters is powered on. The other LED output signals that the shutter OpenCommand is asserted.

Bits 4 and 3 are used to switch a relay in case that an overtemperature or loss of vacuum is detected. The relay can be used to trigger an alarm.
Bit 2 is available to switch the power supply to the vacuum sensor.
Bits 1 and 0 are user-defined inputs/outputs.

As an option bits 15 and 14 can be used to enable +15V/+5V and -15V to a multiplexer in the detector head. Digital data bits 13..8 can be fed to the detector head to select a multiplexer channel in the detector head.

6 SOFTWARE

The PULPO-2 microprocessor is a 486CORE, with 16MB flash disk and 32MB DRAM.
The operating system is Linux EM i486.
The code - completely written in C - follows the VLTSW standards defined in [AD 01].
The code is archived within the CMM "pulpo2" module.
For detailed information about the software structure, compilation, debugging and installation, refer to the README file in the "pulpo2" module.

6.1 Status bytes

PULPO-2 has 34 status bytes which are used for maintaining information on PULPO-2 operation. They are accessible using the command ***SB,n*** where n can be any number from 1 to 34.

STATUS BYTE 1	BIT	FUNCTION
LEDEnab	0	Set if front panel status LEDs enabled
BiasLedOn	1	Set if Bias LED enabled
HeaterOvercurrent	2	Set if Overcurrent
Watchdog	3	Set if Watchdog
TunerActive	4	Set if Tuner Active
AlarmsGlobalEnabled	5	Set if Global Alarm enabled
AlarmsGlobalTEnabled	6	Set if Temperature Global Alarm enabled
Unused	7	Not Used

STATUS BYTE 2	BIT	FUNCTION
ShOpen	0	Set if shutter open
ExpPaused	1	Set if exposure paused by CCD controller (PE,1)
ExpActive	2	Set if exposure active
ShutterNotConnected	3	Set if no shutter detected at start-up shutter test
StatusOpenAvailable	4	Set if OPEN status available at start-up shutter test
StatusCloseAvailable	5	Set if CLOSE status available at start-up shutter test
StatusRemoteAvailable	6	Set if REMOTE status available at start-up shutter test
ShutterError	7	Set if Shutter error detected during start-up or normal exp

STATUS BYTE 3	BIT	FUNCTION
AlarmS1Enab	0	Set if Alarm enabled for Sensor 1
AlarmS2Enab	1	Set if Alarm enabled for Sensor 2
AlarmS3Enab	2	Set if Alarm enabled for Sensor 3
AlarmS4Enab	3	Set if Alarm enabled for Sensor 4
AlarmS5Enab	4	Set if Alarm enabled for Sensor 5
AlarmS6Enab	5	Set if Alarm enabled for Sensor 6
AlarmS7Enab	6	Set if Alarm enabled for Sensor 7
AlarmS8Enab	7	Set if Alarm enabled for Sensor 8

STATUS BYTE 4	BIT	FUNCTION
AlarmS9Enab	0	Set if Alarm enabled for Sensor 9
AlarmS10Enab	1	Set if Alarm enabled for Sensor 10
AlarmS11Enab	2	Set if Alarm enabled for Sensor 11
AlarmS12Enab	3	Set if Alarm enabled for Sensor 12
AlarmS13Enab	4	Set if Alarm enabled for Sensor 13
AlarmS14Enab	5	Set if Alarm enabled for Sensor 14
AlarmS15Enab	6	Set if Alarm enabled for Sensor 15
AlarmS16Enab	7	Set if Alarm enabled for Sensor 16

STATUS BYTE 5	BIT	FUNCTION
AlarmS17Enab	0	Set if Alarm enabled for Sensor 17
AlarmS18Enab	1	Set if Alarm enabled for Sensor 18
AlarmS19Enab	2	Set if Alarm enabled for Sensor 19
AlarmS20Enab	3	Set if Alarm enabled for Sensor 20
AlarmS21Enab	4	Set if Alarm enabled for Sensor 21
AlarmS22Enab	5	Set if Alarm enabled for Sensor 22
AlarmS23Enab	6	Set if Alarm enabled for Sensor 23
AlarmS24Enab	7	Set if Alarm enabled for Sensor 24

STATUS BYTE 6	BIT	FUNCTION
AlarmS25Enab	0	Set if Alarm enabled for Sensor 25
AlarmS26Enab	1	Set if Alarm enabled for Sensor 26
AlarmS27Enab	2	Set if Alarm enabled for Sensor 27
AlarmS28Enab	3	Set if Alarm enabled for Sensor 28
AlarmS29Enab	4	Set if Alarm enabled for Sensor 29
AlarmS30Enab	5	Set if Alarm enabled for Sensor 30
AlarmS31Enab	6	Set if Alarm enabled for Sensor 31
AlarmS32Enab	7	Set if Alarm enabled for Sensor 32

STATUS BYTE 7	BIT	FUNCTION
AlarmS111Enab	0	Set if Alarm enabled for Sensor 111
AlarmS112Enab	1	Set if Alarm enabled for Sensor 112
AlarmS113Enab	2	Set if Alarm enabled for Sensor 113
AlarmS114Enab	3	Set if Alarm enabled for Sensor 114
AlarmS115Enab	4	Set if Alarm enabled for Sensor 115
AlarmS116Enab	5	Set if Alarm enabled for Sensor 116
AlarmS117Enab	6	Set if Alarm enabled for Sensor 117
AlarmS118Enab	7	Set if Alarm enabled for Sensor 118

STATUS BYTE 8	BIT	FUNCTION
AlarmS121Enab	0	Set if Alarm enabled for Sensor 121
AlarmS122Enab	1	Set if Alarm enabled for Sensor 122
AlarmS123Enab	2	Set if Alarm enabled for Sensor 123
AlarmS124Enab	3	Set if Alarm enabled for Sensor 124
AlarmS125Enab	4	Set if Alarm enabled for Sensor 125
AlarmS126Enab	5	Set if Alarm enabled for Sensor 126
AlarmS127Enab	6	Set if Alarm enabled for Sensor 127
AlarmS128Enab	7	Set if Alarm enabled for Sensor 128

STATUS BYTE 9	BIT	FUNCTION
AlarmS131Enab	0	Set if Alarm enabled for Sensor 131
AlarmS132Enab	1	Set if Alarm enabled for Sensor 132
AlarmS133Enab	2	Set if Alarm enabled for Sensor 133
AlarmS134Enab	3	Set if Alarm enabled for Sensor 134
AlarmS135Enab	4	Set if Alarm enabled for Sensor 135
AlarmS136Enab	5	Set if Alarm enabled for Sensor 136
AlarmS137Enab	6	Set if Alarm enabled for Sensor 137
AlarmS138Enab	7	Set if Alarm enabled for Sensor 138

STATUS BYTE 10	BIT	FUNCTION
AlarmS211Enab	0	Set if Alarm enabled for Sensor 211
AlarmS212Enab	1	Set if Alarm enabled for Sensor 212
AlarmS213Enab	2	Set if Alarm enabled for Sensor 213
AlarmS214Enab	3	Set if Alarm enabled for Sensor 214
AlarmS215Enab	4	Set if Alarm enabled for Sensor 215
AlarmS216Enab	5	Set if Alarm enabled for Sensor 216
AlarmS217Enab	6	Set if Alarm enabled for Sensor 217
AlarmS218Enab	7	Set if Alarm enabled for Sensor 218

STATUS BYTE 11	BIT	FUNCTION
AlarmS221Enab	0	Set if Alarm enabled for Sensor 221
AlarmS222Enab	1	Set if Alarm enabled for Sensor 222
AlarmS223Enab	2	Set if Alarm enabled for Sensor 223
AlarmS224Enab	3	Set if Alarm enabled for Sensor 224
AlarmS225Enab	4	Set if Alarm enabled for Sensor 225
AlarmS226Enab	5	Set if Alarm enabled for Sensor 226
AlarmS227Enab	6	Set if Alarm enabled for Sensor 227
AlarmS228Enab	7	Set if Alarm enabled for Sensor 228

STATUS BYTE 12	BIT	FUNCTION
AlarmS231Enab	0	Set if Alarm enabled for Sensor 231
AlarmS232Enab	1	Set if Alarm enabled for Sensor 232
AlarmS233Enab	2	Set if Alarm enabled for Sensor 233
AlarmS234Enab	3	Set if Alarm enabled for Sensor 234
AlarmS235Enab	4	Set if Alarm enabled for Sensor 235
AlarmS236Enab	5	Set if Alarm enabled for Sensor 236
AlarmS237Enab	6	Set if Alarm enabled for Sensor 237
AlarmS238Enab	7	Set if Alarm enabled for Sensor 238

STATUS BYTE 13	BIT	FUNCTION
AlarmS311Enab	0	Set if Alarm enabled for Sensor 311
AlarmS312Enab	1	Set if Alarm enabled for Sensor 312
AlarmS313Enab	2	Set if Alarm enabled for Sensor 313
AlarmS314Enab	3	Set if Alarm enabled for Sensor 314
AlarmS315Enab	4	Set if Alarm enabled for Sensor 315
AlarmS316Enab	5	Set if Alarm enabled for Sensor 316
AlarmS317Enab	6	Set if Alarm enabled for Sensor 317
AlarmS318Enab	7	Set if Alarm enabled for Sensor 318

STATUS BYTE 14	BIT	FUNCTION
AlarmS321Enab	0	Set if Alarm enabled for Sensor 321
AlarmS322Enab	1	Set if Alarm enabled for Sensor 322
AlarmS323Enab	2	Set if Alarm enabled for Sensor 323
AlarmS324Enab	3	Set if Alarm enabled for Sensor 324
AlarmS325Enab	4	Set if Alarm enabled for Sensor 325
AlarmS326Enab	5	Set if Alarm enabled for Sensor 326
AlarmS327Enab	6	Set if Alarm enabled for Sensor 327
AlarmS328Enab	7	Set if Alarm enabled for Sensor 328

STATUS BYTE 15	BIT	FUNCTION
AlarmS331Enab	0	Set if Alarm enabled for Sensor 331
AlarmS332Enab	1	Set if Alarm enabled for Sensor 332
AlarmS333Enab	2	Set if Alarm enabled for Sensor 333
AlarmS334Enab	3	Set if Alarm enabled for Sensor 334
AlarmS335Enab	4	Set if Alarm enabled for Sensor 335
AlarmS336Enab	5	Set if Alarm enabled for Sensor 336
AlarmS337Enab	6	Set if Alarm enabled for Sensor 337
AlarmS338Enab	7	Set if Alarm enabled for Sensor 338

STATUS BYTE 16	BIT	FUNCTION
AlarmS411Enab	0	Set if Alarm enabled for Sensor 411
AlarmS412Enab	1	Set if Alarm enabled for Sensor 412
AlarmS413Enab	2	Set if Alarm enabled for Sensor 413
AlarmS414Enab	3	Set if Alarm enabled for Sensor 414
AlarmS415Enab	4	Set if Alarm enabled for Sensor 415
AlarmS416Enab	5	Set if Alarm enabled for Sensor 416
AlarmS417Enab	6	Set if Alarm enabled for Sensor 417
AlarmS418Enab	7	Set if Alarm enabled for Sensor 418

STATUS BYTE 17	BIT	FUNCTION
AlarmS421Enab	0	Set if Alarm enabled for Sensor 421
AlarmS422Enab	1	Set if Alarm enabled for Sensor 422
AlarmS423Enab	2	Set if Alarm enabled for Sensor 423
AlarmS424Enab	3	Set if Alarm enabled for Sensor 424
AlarmS425Enab	4	Set if Alarm enabled for Sensor 425
AlarmS426Enab	5	Set if Alarm enabled for Sensor 426
AlarmS427Enab	6	Set if Alarm enabled for Sensor 427
AlarmS428Enab	7	Set if Alarm enabled for Sensor 428

STATUS BYTE 18	BIT	FUNCTION
AlarmS431Enab	0	Set if Alarm enabled for Sensor 431
AlarmS432Enab	1	Set if Alarm enabled for Sensor 432
AlarmS433Enab	2	Set if Alarm enabled for Sensor 433
AlarmS434Enab	3	Set if Alarm enabled for Sensor 434
AlarmS435Enab	4	Set if Alarm enabled for Sensor 435
AlarmS436Enab	5	Set if Alarm enabled for Sensor 436
AlarmS437Enab	6	Set if Alarm enabled for Sensor 437
AlarmS438Enab	7	Set if Alarm enabled for Sensor 438

STATUS BYTE 19	BIT	FUNCTION
AlarmS1Trigger	0	Set if Alarm triggered for Sensor 1
AlarmS2Trigger	1	Set if Alarm triggered for Sensor 2
AlarmS3Trigger	2	Set if Alarm triggered for Sensor 3
AlarmS4Trigger	3	Set if Alarm triggered for Sensor 4
AlarmS5Trigger	4	Set if Alarm triggered for Sensor 5
AlarmS6Trigger	5	Set if Alarm triggered for Sensor 6
AlarmS7Trigger	6	Set if Alarm triggered for Sensor 7
AlarmS8Trigger	7	Set if Alarm triggered for Sensor 8

STATUS BYTE 20	BIT	FUNCTION
AlarmS9Trigger	0	Set if Alarm triggered for Sensor 9
AlarmS10Trigger	1	Set if Alarm triggered for Sensor 10
AlarmS11Trigger	2	Set if Alarm triggered for Sensor 11
AlarmS12Trigger	3	Set if Alarm triggered for Sensor 12
AlarmS13Trigger	4	Set if Alarm triggered for Sensor 13
AlarmS14Trigger	5	Set if Alarm triggered for Sensor 14
AlarmS15Trigger	6	Set if Alarm triggered for Sensor 15
AlarmS16Trigger	7	Set if Alarm triggered for Sensor 16

STATUS BYTE 21	BIT	FUNCTION
AlarmS17Trigger	0	Set if Alarm triggered for Sensor 17
AlarmS18Trigger	1	Set if Alarm triggered for Sensor 18
AlarmS19Trigger	2	Set if Alarm triggered for Sensor 19
AlarmS20Trigger	3	Set if Alarm triggered for Sensor 20
AlarmS21Trigger	4	Set if Alarm triggered for Sensor 21
AlarmS22Trigger	5	Set if Alarm triggered for Sensor 22
AlarmS23Trigger	6	Set if Alarm triggered for Sensor 23
AlarmS24Trigger	7	Set if Alarm triggered for Sensor 24

STATUS BYTE 22	BIT	FUNCTION
AlarmS25Trigger	0	Set if Alarm triggered for Sensor 25
AlarmS26Trigger	1	Set if Alarm triggered for Sensor 26
AlarmS27Trigger	2	Set if Alarm triggered for Sensor 27
AlarmS28Trigger	3	Set if Alarm triggered for Sensor 28
AlarmS29Trigger	4	Set if Alarm triggered for Sensor 29
AlarmS30Trigger	5	Set if Alarm triggered for Sensor 30
AlarmS31Trigger	6	Set if Alarm triggered for Sensor 31
AlarmS32Trigger	7	Set if Alarm triggered for Sensor 32

STATUS BYTE 23	BIT	FUNCTION
AlarmS111Trigger	0	Set if Alarm triggered for Sensor 111
AlarmS112Trigger	1	Set if Alarm triggered for Sensor 112
AlarmS113Trigger	2	Set if Alarm triggered for Sensor 113
AlarmS114Trigger	3	Set if Alarm triggered for Sensor 114
AlarmS115Trigger	4	Set if Alarm triggered for Sensor 115
AlarmS116Trigger	5	Set if Alarm triggered for Sensor 116
AlarmS117Trigger	6	Set if Alarm triggered for Sensor 117
AlarmS118Trigger	7	Set if Alarm triggered for Sensor 118

STATUS BYTE 24	BIT	FUNCTION
AlarmS121Trigger	0	Set if Alarm triggered for Sensor 121
AlarmS122Trigger	1	Set if Alarm triggered for Sensor 122
AlarmS123Trigger	2	Set if Alarm triggered for Sensor 123
AlarmS124Trigger	3	Set if Alarm triggered for Sensor 124
AlarmS125Trigger	4	Set if Alarm triggered for Sensor 125
AlarmS126Trigger	5	Set if Alarm triggered for Sensor 126
AlarmS127Trigger	6	Set if Alarm triggered for Sensor 127
AlarmS128Trigger	7	Set if Alarm triggered for Sensor 128

STATUS BYTE 25	BIT	FUNCTION
AlarmS131Trigger	0	Set if Alarm triggered for Sensor 131
AlarmS132Trigger	1	Set if Alarm triggered for Sensor 132
AlarmS133Trigger	2	Set if Alarm triggered for Sensor 133
AlarmS134Trigger	3	Set if Alarm triggered for Sensor 134
AlarmS135Trigger	4	Set if Alarm triggered for Sensor 135
AlarmS136Trigger	5	Set if Alarm triggered for Sensor 136
AlarmS137Trigger	6	Set if Alarm triggered for Sensor 137
AlarmS138Trigger	7	Set if Alarm triggered for Sensor 138

STATUS BYTE 26	BIT	FUNCTION
AlarmS211Trigger	0	Set if Alarm triggered for Sensor 211
AlarmS212Trigger	1	Set if Alarm triggered for Sensor 212
AlarmS213Trigger	2	Set if Alarm triggered for Sensor 213
AlarmS214Trigger	3	Set if Alarm triggered for Sensor 214
AlarmS215Trigger	4	Set if Alarm triggered for Sensor 215
AlarmS216Trigger	5	Set if Alarm triggered for Sensor 216
AlarmS217Trigger	6	Set if Alarm triggered for Sensor 217
AlarmS218Trigger	7	Set if Alarm triggered for Sensor 218

STATUS BYTE 27	BIT	FUNCTION
AlarmS221Trigger	0	Set if Alarm triggered for Sensor 221
AlarmS222Trigger	1	Set if Alarm triggered for Sensor 222
AlarmS223Trigger	2	Set if Alarm triggered for Sensor 223
AlarmS224Trigger	3	Set if Alarm triggered for Sensor 224
AlarmS225Trigger	4	Set if Alarm triggered for Sensor 225
AlarmS226Trigger	5	Set if Alarm triggered for Sensor 226
AlarmS227Trigger	6	Set if Alarm triggered for Sensor 227
AlarmS228Trigger	7	Set if Alarm triggered for Sensor 228

STATUS BYTE 28	BIT	FUNCTION
AlarmS231Trigger	0	Set if Alarm triggered for Sensor 231
AlarmS232Trigger	1	Set if Alarm triggered for Sensor 232
AlarmS233Trigger	2	Set if Alarm triggered for Sensor 233
AlarmS234Trigger	3	Set if Alarm triggered for Sensor 234
AlarmS235Trigger	4	Set if Alarm triggered for Sensor 235
AlarmS236Trigger	5	Set if Alarm triggered for Sensor 236
AlarmS237Trigger	6	Set if Alarm triggered for Sensor 237
AlarmS238Trigger	7	Set if Alarm triggered for Sensor 238

STATUS BYTE 29	BIT	FUNCTION
AlarmS311Trigger	0	Set if Alarm triggered for Sensor 311
AlarmS312Trigger	1	Set if Alarm triggered for Sensor 312
AlarmS313Trigger	2	Set if Alarm triggered for Sensor 313
AlarmS314Trigger	3	Set if Alarm triggered for Sensor 314
AlarmS315Trigger	4	Set if Alarm triggered for Sensor 315
AlarmS316Trigger	5	Set if Alarm triggered for Sensor 316
AlarmS317Trigger	6	Set if Alarm triggered for Sensor 317
AlarmS318Trigger	7	Set if Alarm triggered for Sensor 318

STATUS BYTE 30	BIT	FUNCTION
AlarmS321Trigger	0	Set if Alarm triggered for Sensor 321
AlarmS322Trigger	1	Set if Alarm triggered for Sensor 322
AlarmS323Trigger	2	Set if Alarm triggered for Sensor 323
AlarmS324Trigger	3	Set if Alarm triggered for Sensor 324
AlarmS325Trigger	4	Set if Alarm triggered for Sensor 325
AlarmS326Trigger	5	Set if Alarm triggered for Sensor 326
AlarmS327Trigger	6	Set if Alarm triggered for Sensor 327
AlarmS328Trigger	7	Set if Alarm triggered for Sensor 328

STATUS BYTE 31	BIT	FUNCTION
AlarmS331Trigger	0	Set if Alarm triggered for Sensor 331
AlarmS332Trigger	1	Set if Alarm triggered for Sensor 332
AlarmS333Trigger	2	Set if Alarm triggered for Sensor 333
AlarmS334Trigger	3	Set if Alarm triggered for Sensor 334
AlarmS335Trigger	4	Set if Alarm triggered for Sensor 335
AlarmS336Trigger	5	Set if Alarm triggered for Sensor 336
AlarmS337Trigger	6	Set if Alarm triggered for Sensor 337
AlarmS338Trigger	7	Set if Alarm triggered for Sensor 338

STATUS BYTE 32	BIT	FUNCTION
AlarmS411Trigger	0	Set if Alarm triggered for Sensor 411
AlarmS412Trigger	1	Set if Alarm triggered for Sensor 412
AlarmS413Trigger	2	Set if Alarm triggered for Sensor 413
AlarmS414Trigger	3	Set if Alarm triggered for Sensor 414
AlarmS415Trigger	4	Set if Alarm triggered for Sensor 415
AlarmS416Trigger	5	Set if Alarm triggered for Sensor 416
AlarmS417Trigger	6	Set if Alarm triggered for Sensor 417
AlarmS418Trigger	7	Set if Alarm triggered for Sensor 418

STATUS BYTE 33	BIT	FUNCTION
AlarmS421Trigger	0	Set if Alarm triggered for Sensor 421
AlarmS422Trigger	1	Set if Alarm triggered for Sensor 422
AlarmS423Trigger	2	Set if Alarm triggered for Sensor 423
AlarmS424Trigger	3	Set if Alarm triggered for Sensor 424
AlarmS425Trigger	4	Set if Alarm triggered for Sensor 425
AlarmS426Trigger	5	Set if Alarm triggered for Sensor 426
AlarmS427Trigger	6	Set if Alarm triggered for Sensor 427
AlarmS428Trigger	7	Set if Alarm triggered for Sensor 428

STATUS BYTE 34	BIT	FUNCTION
AlarmS431Trigger	0	Set if Alarm triggered for Sensor 431
AlarmS432Trigger	1	Set if Alarm triggered for Sensor 432
AlarmS433Trigger	2	Set if Alarm triggered for Sensor 433
AlarmS434Trigger	3	Set if Alarm triggered for Sensor 434
AlarmS435Trigger	4	Set if Alarm triggered for Sensor 435
AlarmS436Trigger	5	Set if Alarm triggered for Sensor 436
AlarmS437Trigger	6	Set if Alarm triggered for Sensor 437
AlarmS438Trigger	7	Set if Alarm triggered for Sensor 438

6.2 Command reply and errors

Any command reply has one of the following forms:

OK
OK,<parameter>
ERR,<error code>

The error codes are defined in the `pulpolErrors.h` file:

```
/*
 *  The error number from 0 to 25 are kept with the same definition as
 *  PULPO1 for backward compatibility
 */
#define SUCCESS                                 0  /*                          */
#define ERR_UNDEFINED_CMD                       1  /*                          */
#define ERR_BAD_PARAMETER                       2  /*                          */
#define ERR_OUT_OF_RANGE                        3  /*                          */
#define ERR_SENSOR_NOT_CONNECTED                4  /*                          */
#define ERR_NO_EXPOSURE_ACTIVE                  5  /*                          */
#define ERR_EXP_ALREADY_RUNNING                 6  /*                          */
#define ERR_EXP_ALREADY_PAUSED                  7  /*                          */
#define ERR_EXP_UNDEFINED                       8  /*                          */
#define ERR_RTC_TIMEOUT                         9  /*                          */
#define ERR_VAC_GAUGE_DEFECTIVE                 10 /*                          */
#define ERR_LOGGING_OFF                         11 /*                          */
#define ERR_NO_SENSOR_ASSOCIATED                12 /*                          */
#define ERR_NO_SHUTTER_STATUS_AVAILABLE         13 /*                          */
#define ERR_RTC_ERROR                           14 /*                          */
#define ERR_NO_LOG_ACTIVITY                     15 /*                          */
#define ERR_SHUTTER_OPEN_TIMEOUT                16 /*                          */
#define ERR_SHUTTER_CLOSE_TIMEOUT               17 /*                          */
#define ERR_VAC_PWR_OFF                         18 /*                          */
#define ERR_SHUTTER_ALREADY_CLOSED              19 /*                          */
#define ERR_NO_SHUTTER_BOARD_CONNECTED          20 /*                          */
#define ERR_INSTRUMENT_SHUTTER_UNDEFINED        21 /*                          */
#define ERR_SHUTTER_INSTRUMENT_INCONSISTENCY    22 /*                          */
#define ERR_NO_INTEGER_VALUE                    23 /*                          */
#define ERR_OUT_OF_TIME                         24 /*                          */
#define ERR_PAUSED_NOT_ALLOWED_DURING_DARK      25 /*                          */
#define ERR_NOT_YET_IMPLEMENTED                 26 /*                          */

#define ERR_GENERAL                40 /* General error                         */

/*
 * p2periph errors:
 */
#define ERR_READ_REG               41 /* Error when reading physical register */
#define ERR_WRITE_REG              42 /* Error when writing physical register */
#define ERR_ADC_TIMEOUT            43 /* Timeout during A/D conversion        */
#define ERR_CURRENT_OUT_OF_RANGE   44 /* Value for current out of range       */
#define ERR_HEATER_WRONG_ID        45 /* Heater ID wrong                      */
#define ERR_XTIME_OUT_OF_RANGE     46 /* Exposure time out of range           */
#define ERR_ENABLE_IO              47 /* Error when enabling I/O access       */
#define ERR_DISABLE_IO             48 /* Error when disabling I/O access      */
#define ERR_SENSOR_VAL             49 /* Wrong sensor id                      */
#define ERR_INITREG_MAINLOOP       50 /* Error initialising registers         */
#define ERR_ENABLE_LED             51 /* Error when enabling LED              */
#define ERR_DISABLE_LED            52 /* Error when disabling LED             */
#define ERR_ARM_SHUTTER            53 /* Error when arming shutter            */
#define ERR_OPENING_SHUTTER        54 /* Error when opening shutter           */
#define ERR_CLOSING_SHUTTER        55 /* Error when closing shutter           */
#define ERR_READ_EXPTIME           56 /* Error while reading exp time         */
#define ERR_RESET_HEATERWD         57 /* Error while reseting Heater watchdog */
#define ERR_RESET_VACALARM         58 /* Error while reseting VacAlarm        */
#define ERR_RESET_TEMPALARM        59 /* Error while reseting TempAlarm       */
```

```
#define ERR_SET_VACALARM           60 /* Error while seting VacAlarm          */
#define ERR_SET_TEMPALARM          61 /* Error while seting TempAlarm         */
#define ERR_TURNOFF_HEATER         62 /* Error while turning off heater       */
#define ERR_TURNON_HEATER          63 /* Error while turning on heater        */
#define ERR_CREATING_SEMAPHORE     64 /* Error while creating semaphore       */
#define ERR_INITIALIZING_SEMAPHORE 65 /* Error while init semaphore           */
#define ERR_LOCKING_SEMAPHORE      66 /* Error while locking semaphore        */
#define ERR_UNLOCKING_SEMAPHORE    67 /* Error while unlocking semaphore      */
#define ERR_RESET_HEATEROC         68 /* Error while reseting Heater watchdog */
#define ERR_EXTMUX_ENABLE          69 /* Error en/disabling External Mux      */
#define ERR_WRONG_EXTMUX           70 /* Wrong External Mux configuration     */
#define ERR_READ_SENSOR            71 /* Error reading Sensor                 */
#define ERR_READ_EXT_SENSOR        72 /* Error reading Sensor on ExtMux       */
#define ERR_ACTIVATE_HEATER        73 /* Error activating heater              */
#define ERR_SHUTTER_STATUS         74 /* Error checking shutter status        */
#define ERR_LOCK                   75 /* Error locking programme              */
#define ERR_SET_ALARMS             76 /* Error setting all alarms             */
#define ERR_RESISTANCE             77 /* Error reading heater resistances     */
#define ERR_BROKEN_SENSOR          78 /* PT100 is broken                      */
#define ERR_MSGINIT                79 /* Error initialising msg system        */
#define ERR_LOADING_CONF           80 /* Error loading configuration file     */
#define ERR_HEATER_ON              81 /* Heater on, sensor can't be disabled  */
#define ERR_SENSOR_NOT_ENABLED     82 /* Disabled sensor can't be read        */

/* Logging errors: */
#define ERR_LOGINIT               100 /* Error initialising Logging           */
#define ERR_LOGFILE_OPEN          101 /* Error opening log file               */
#define ERR_LOGFILE_FTRUNCATE     102 /* Error truncating log file            */
#define ERR_LOGFILE_MMAP          103 /* Error mmapping memory for log file   */
#define ERR_LOGFILE_WRITE         104 /* Error writing file                   */


/*
 * p2serial errors:
 */
#define ERR_READING_BUFFER        200 /* Error while reading serial buffer    */
#define ERR_WRITING_BUFFER        201 /* Error while writing serial buffer    */
#define ERR_OPEN_SERIAL           202 /* Error while opening serial port      */

/*
 * p2uif errors:
 */
#define ERR_CHECKING_KEYS         300 /* Error when checking keys             */
#define ERR_INIT_KEYPAD           301 /* Error when initializing keypad       */
#define ERR_GET_KEYPAD_STATE      302 /* Error when getting keypad state      */
#define ERR_RESET_KEYPAD_STATE    303 /* Error when resetting keypad state    */
#define ERR_DISPLAY_SENSORS       304 /* Error when displaying sensors        */
#define ERR_DISPLAY_MENU1         305 /* Error when displaying menu1          */
#define ERR_DISPLAY_MENUPID       306 /* Error when displaying menu PID       */
#define ERR_DISPLAY_MENUALARMS    307 /* Error when displaying menu Alarms    */
#define ERR_DISPLAY_MENUMISC      308 /* Error when displaying menu Misc      */

#define ERR_SHOW_STATE_SENSORS    309 /* Error when displaying state sensors  */
#define ERR_MAIN_MENU             310 /* Error when displaying main menu      */
#define ERR_MENU_HEATER_CTRL      311 /* Error when displaying Heater Ctrl    */
#define ERR_MENU_ALARMS           312 /* Error when displaying Menu Alarms    */
#define ERR_MENU_MISC             313 /* Error when displaying Menu Misc      */

/*
 * PID automatas errors:
 */
#define ERR_PID                   400
```

6.3 Dimensions of command and reply parameters

The following conventions are used:

- **temperature** values are expressed in **Kelvin** (**K**)
- **pressure** values are expressed in **millibar** (**mB**)
- **current** values are expressed in **milliampere** (**mA**)
- **sensor average** and **deviation** values are expressed in digital units (**ADU**)
- heater **resistances** are expressed in **Ohm** (**Ω**)
- **power consumption** is expressed in **Watt** (**W**)
- **expoure times** are expressed in **second** (**s**)
- **shutter open and close delays** are expressed in **microsecond** (**µs**)

6.4 Commands

Here is the complete list of serial PULPO-2 commands. The commands consist of two alphabetic characters followed by arguments. The end of the command string is always a carriage return (ASCII 0x13). The answer from PULPO is always an OK plus parameters, if any, or an ERR.

In the examples the commands are in `courier`, while the replies are in ***`courier bold italic`***.

> (Start Exposure)

```
Start Exposure. Open the shutter for the time defined with the XT
(set eXposure Time) command, or until an Abort Exposure (<) command
is received. The reply contains the open shutter delay (in µs) or an
error code.

Syntax: >

Example:    XT,10       // set 10 seconds for the exposure time
            OK
            >           // start the exposure
            OK,42       // open shutter delay was 42 ms

Related commands: < (Abort Exposure)
                  PE (Pause Exposure)
                  XT (eXposure Time)
```

< (Abort Exposure)

Close the shutter. The reply contains the close shutter delay (in µs) or an error code.

Syntax: <

```
Example:    XT,10       // set 10 seconds for the exposure time
            OK
            >           // start the exposure
            OK,42       // open shutter delay was 42 ms
            <           // abort exposure
            OK,45       // close shutter delay was 45 ms
```

Related commands: > (Start Exposure)
PE (Pause Exposure)
XT (eXposure Time)

Alarm Enable

Enable alarms connected to the sensors.

```
Syntax: AE,s[,v]
where   s=0                          => switch on/off alarms
        s=1-6                        => temperature sensors
        s=8                          => vacuum sensor
        s=10-32                      => temperature sensors
        s=111-118, 121-128, 131-138 => sensors on ext Mux 1
        s=211-218, 221-228, 231-238 => sensors on ext Mux 2
        s=311-318, 321-328, 331-338 => sensors on ext Mux 3
        s=411-418, 421-428, 431-438 => sensors on ext Mux 4
        v=0,1                        => disable/enable alarm
```

To access a sensor on an external Mux, the command
`EM,1`
must have been issued before.

If the parameter v is not included, the enable status for the given sensor is returned.

If an alarm is triggered, the sequence

```
AE,0,0      // disable all alarms
AE,0,1      // enable all alarms
```

resets the Triggered Alarm bits in the Status Bytes (shown by SA command).
Note: the condition which triggered the alarm must be fixed before issuing the AE,0,1 command, otherwise the alarm is immediately triggered again.

If the Self Recovery was triggered, the sequence

```
AE,0,0      // disable all alarms
AE,0,1      // enable all alarms
```

resets the Heater Set Points to the default values (those defined before the Self recovery was started).
Note: the condition which triggered the Self Recovery must be fixed before issuing the AE,0,1 command, otherwise the alarm is immediately triggered and the Self Recovery is enabled again.

Default values: 0 (alarms disabled)

```
Example:    AE,0,0      // disable all alarms
            OK
            AE,1        // read alarm enable status for sensor 1
            OK,0        // alarm for sensor 1 disabled
```

```
Related commands: LL (Low Limit for alarm)
                  TT (Temperature Trip point)
                  TA (Temperature Alarm)
                  SA (Show Triggered Alarms)
                  SR (Self Recovery)
```

Auto Tuner

Turns the PID Auto Tuner routine ON or OFF. This command determines the PID parameters for heater 1, or 1 and 2 when they are connected in parallel.

```
Syntax: AT[,n]
        n=0: turn Auto Tuner off
        n=1: turn Auto Tuner on

Default value: 1

Example:    AT,1        // start AutoTuner
            OK
            AT,0        // stop AutoTuner
            OK
            AT          // check AutoTuner status
            OK,0        // AutoTuner off
```

NOT YET IMPLEMENTED.

AVerage

Computes the average value and standard deviation of 10 measurements of sensor 's'. The reply values are in ADUs.

```
Syntax: AV,s
where:  s=1-32                      => temperature sensors
        s=8                         => vacuum sensor
        s=9                         => heater current
        s=111-118, 121-128, 131-138 => sensors on ext Mux 1
        s=211-218, 221-228, 231-238 => sensors on ext Mux 2
        s=311-318, 321-328, 331-338 => sensors on ext Mux 3
        s=411-418, 421-428, 431-438 => sensors on ext Mux 4

Example:    AV,1        // read average value of sensor 1
            OK,42350.1,2.5
```

Bias Power

Set the current level (%) to the bias LED. Full power (100%) is approximately 25mA.

```
Syntax: BP[,f]
        with f=0.0->100.0

Default value: 0

Example:    BP,50       // set LED current to 12.5 mA
            OK
            BP          // read Power to Bias LED
            OK,50.0
```

Close Delay

Return the last shutter Opening Delay [in microseconds].

```
Syntax: CD

Example:    CD          // request shutter close delay
            OK,45000    // close shutter delay was 45ms
```

Controller Mode

Set serial link in controller mode (characters are not echoed back).
This is the normal mode for connecting PULPO with the CCD controller.

Syntax: CM

```
Example:    CM    // set serial link in Controller Mode
            OK
```

Related commands: EM (Echo Mode)
TM (Terminal Mode)

Control Sensor

Command Control Sensor

Syntax: CS,h[,s]
where:

h=1-8	=> heater
s=1-32	=> temperature sensors
s=8	=> vacuum sensor
s=9	=> heater current
s=111-118, 121-128, 131-138	=> sensors on ext Mux 1
s=211-218, 221-228, 231-238	=> sensors on ext Mux 2
s=311-318, 321-328, 331-338	=> sensors on ext Mux 3
s=411-418, 421-428, 431-438	=> sensors on ext Mux 4

This command associates the sensor 's' to heater 'h' for the PID controller.
If no parameter 's' is given, the reply contains the sensor which is associated with the heater 'h'.

```
Example:    CS,2,1      // associate heater 2 with sensor 1
            OK
            CS,2        // read sensor associated with heater 2
            OK,1
```

Disable Leds

Disable the front panel LEDs.

Syntax: DL[,n]
without parameter "n", or n=1: disable leds
with n=0: enable leds

This command is intended to avoid light pollution.

```
Example:    DL          //disable LEDs
            OK
            DL,0        //enable LEDs
            OK
```

Dewar Model

Define or read the dewar model.

Syntax: DM[,m] m=0..4

where m=0..4 (Dewar Model connected to PULPO)
- m=0 : all independent heaters
- m=1 : heaters 1 and 2 in parallel, the others independent
- m=2 : heaters 2 and 3 in parallel, the others independent
- m=3 : heaters 1 and 3 in parallel, the others independent
- m=4 : heaters 1, 2 and 3 in parallel, the others independent

Default value: 0

```
Example:    DM,1        // set dewar model 1
            OK
            DM          // read dewar model
            OK,1
```

ECho Mode

Set serial link in Echo Mode (characters received through the serial link are echoed back).
This mode is useful to test the serial link reliability.
This mode is active since a [cr] is detected in the incoming string

Syntax: EC

```
Example:    EC    // set serial link in Echo Mode
            OK
```

Related commands: CM (Controller Mode)
TM (Terminal Mode)

External Muxboards

Enable/Disable External Muxboards.

Syntax: EM[,n]
with n=0 Disable External MuxBoards
n=1 Enable External MuxBoards

with no parameter, it returns the MuxBoard enable status.

Default value: 0

```
Example:    EM,1        // enable external muxboards
            OK
            EM          // read if external muxboards are enabled
            OK,1
```

Note: Command implemented for PULPO-2 (not existing in Pulpo)

Enable/disable Sensor

```
Disable or enable the readout of a Sensor.
By default, at startup all the Sensor readout are enabled.

Syntax: ES,s[,v]
where   s=1-7,10-32                 => temperature sensors
        s=8                         => vacuum sensor
        s=9                         => heater current
        s=111-118, 121-128, 131-138 => sensors on ext Mux 1
        s=211-218, 221-228, 231-238 => sensors on ext Mux 2
        s=311-318, 321-328, 331-338 => sensors on ext Mux 3
        s=411-418, 421-428, 431-438 => sensors on ext Mux 4
with v=0 Disable Sensor readout
     v=1 Enable  Sensor readout

Example:    ES,111,0    // disable readout of Sensor 111
            OK
            ES,111      // read if Sensor 111 readout is enabled
            OK,0
```

Filter Value

```
Read or set the value for the A coefficient in the digital recursive
filter used to calculate the sensors temperature. The low pass filter
has the form of Yi= A*Yi-1 + (1-A)*Xi, were Xi is the sensor readout,
Yi-1 is the previous computed value for the temperature.

Syntax: FV[,f]
        with f = 0.0 -> 1.0

Default value: 0

Example:    FV,0.5      // set filter A coefficient to 0.5
            OK
            FV          // set filter A coefficient
            OK,0.5
```

HEater n ON/OFF

```
Enable or disable the PID routine for heater 'h'.

Syntax: HE,h[,v]
        with h:1->8  v:0->3
        v=0 turn off the PID
        v=1 turns the PID on

Default values: 0 (heaters disabled)

Example:    HE,1,1      // enable heater 1
            OK
            HE,1        // read if heater 1 is enabled or disabled
            OK,1
```

Heater Mode

Define the sampling time for the PID controller.

```
Syntax: HM,h[,m]
        with h:1->8 and m:0->2
        m=0 or 1 : 1 s of duty cycle
        m=2      : 10 s of duty cycle

Default value: 1

Example:    HM,1,1      // set heater 1 mode to 1
            OK
            HM,1        // read heater 1 mode
            OK,1
```

Heater Resistance

Read the resistance value of a heater. Normally the range is between 75 Ohms and 100 Ohms for the old dewars.
The resistance value is calculated during PULPO initialization.

```
Syntax: HR,h
        with h:1->8

Example:    HR,1        //read resistance for heater 1
            OK,75.6     //resistance equal 75.6 Ohms
```

INit

Initialize PULPO-2 (reboot).

```
Syntax: IN

Example:    IN          // reboot PULPO-2
            OK
```

Derivative Constant (kd) for PID controller

Read or set derivative constant for PID controller associated to a heater (see 6.6 for the formula).

```
Syntax: KD,h[,f]
        with h:1->8
             f: 0.0->200.0

Default value: 0

Example:    KD,1,30.5   //set kd for heater 1 equal to 30.5
            OK
            KD,1
            OK,30.5
```

Integral Constant (ki) for PID controller

Read or set integral constant for the PID controller associated to a heater (see 6.6 for the formula).

Syntax: KI,h[,f]
with h:1->8
f: 0.0->1000.0

Default value: 120

Example: KI,1,40.5 //set ki for heater 1 equal to 40.5
OK
KI,1
OK,40.5

Proportional Constant (kp) for PID controller

Read or set the proportional constant for the PID controller associated to a heater (see 6.6 for the formula).

Syntax: KP,h[,f]
with h:1->8
f: 0.0->1000.0

Default value: 37

Example: KP,1,50.5 //set kp for heater 1 equal to 50.5
OK
KP,1
OK,50.5

Log Begin

Start logging.

Syntax: LB

Example: LB // log begin
OK

Related commands: LC (Log Check)
LO (LOg interval)
LR (Log Restart)
LS (Log Stop)

Log Check

Log Check. Return last element written in log file and how many elements are still free.

Syntax: LC

Example: LC
OK,218,64782 // 218 records written, 64782 free

Related commands: LB (Log Begin)
LO (LOg interval)
LR (Log Restart)
LS (Log Stop)

Log Dump

Send logged information through serial link.

Syntax: LD

NOT IMPLEMENTED.
IMPORTANT: due to the dimension of the log file, the only practical way to retrieve it is using ftp, through the ethernet connection.

Log to local File

Enable/disable logging to local file on PULPO-2.

```
Syntax: LF[,v]
where   n=0,1
        n=0 : disable logging to local file
        n=1 : enable logging to local file

Default value: 1

Example:    LF,1        // enable logging to local file
            OK
            LF
            OK,1
```

Low Limit

Read or set the Low Limit 'f' for triggering an alarm on sensor 's'. 'f' is in 'K' for a temperature sensor, or 'mB' for the vacuum sensor.

```
Syntax: LL,s[,f]
where   s=1-6                       => temperature sensors
        s=8                         => vacuum sensor
        s=10-32                     => temperature sensors
        s=111-118, 121-128, 131-138 => sensors on ext Mux 1
        s=211-218, 221-228, 231-238 => sensors on ext Mux 2
        s=311-318, 321-328, 331-338 => sensors on ext Mux 3
        s=411-418, 421-428, 431-438 => sensors on ext Mux 4
        s=1-6                       => temperature sensors

Default values: 1e-09 mB for vacuum sensor
                77 K for temperature sensors

Example:    LL,9,130    // set alarm low limit on sensor 9 to 130 K
            OK
            LL,9
            OK,130

Related commands: TT (Temperature Trip point)
                  AE (Alarm Enable)
                  TA (Temperature Alarm)

Note: Command implemented for PULPO-2 (not existing in Pulpo)
```

LOg set

Define the logging interval [sec].

Syntax: LO,[t]
with t>0

```
Example:    LO,60       // store once per minute
            OK
            LO
            OK,60
```

Default value: 600

Related commands: LB (Log Begin)
LC (Log Check)
LR (Log Restart)
LS (Log Stop)

Log Restart

Restart logging from the first free place in the circular buffer (i.e., without overwriting the previous information).

Syntax: LR

```
Example:    LR          // log restart
            OK
```

Related commands: LB (Log Begin)
LC (Log Check)
LO (LOg interval)
LS (Log Stop)

Log Stop

Stop logging.

Syntax: LS

```
Example:    LS          // log stop
            OK
```

Related commands: LB (Log Begin)
LC (Log Check)
LO (LOg interval)
LR (Log Restart)

Open Delay

Return the last open shutter delay (in microseconds).

Syntax: OD

```
Example:    OD          // read open shutter delay
            OK,42000    // open shutter delay was 42ms
```

Open Shutter

Open the shutter. The shutter remains open until a Shutter Close (SC) or an Abort (<) command is issued.

Syntax: OS

```
Example:    OS          // open shutter
            OK
```

Pause Exposure

Pause an exposure. The reply contains the close or open shutter delay (in microseconds) or an error code.

```
Syntax: PE,n
        with n=0,1

        n=1  pause the active exposure (if normal exposure, the
             shutter is closed)
        n=0  paused exposure is continued

Example:    PE,1        // pause exposure
            OK,45000    // close shutter delay was 45ms
            PE,0        // continue exposure
            OK,45000    // open shutter de where   n=0,1
        n=0 : disable vacuum gauge
        n=1 : enable vacuum gaugelay was 45ms

Related commands: > (Start Exposure)
                  < (Abort Exposure)
                  XT (eXposure Time)
```

set PoWer

Read or set the power delivered to a heater. In case of setting the power, he reply contains the duty cycle (%) and the power consumption (W). 100% duty cycle corresponds to ~7.5 W for a 75 Ohm heater (ESO standard).

```
Syntax: PW,n[,duty]
        with n:1->8 duty:0->100

Default value: 0

Example:    PW,1,100    // set duty cycle for heater 1 to 100%
            OK
            PW,1        // read duty cycle for heater 1
            OK,100,6,5  // 100% duty cycle, 6.5 W
```

Reset Heaters

Perform a test on the heaters , activating them for a short time and measuring the current and the resistance.

```
Syntax: RH

Example:    RH          // reset heaters
            OK
```

Reset OverCurrent

Reset OverCurrent on heaters. Read the Heater Supervision Register and reset Overcurrent if it is set.

Syntax: RO

```
Example:    RO          // reset overcurrent
            OK
```

Reset Shutter

Run a test on the shutter and update the status.

Syntax: RS

```
Example:    RS          // reset shutter
            OK
```

Reset Temperature sensors

Test all the sensor inputs to determine which ones are connected and set the status bytes accordingly.

Syntax: RT

```
Example:    RT          // reset temperature sensors
            OK
```

Reset Vacuum sensor

Check Vacuum Sensor connection. In the reply the connection state is shown ('0' for 'not connected', '1' for 'connected')

Syntax: RV

```
Example:    RV          // reset vacuum sensor
            OK,1        // vacuum sensor is connected
```

Show triggered Alarms

Show triggered Alarms.

Syntax: SA

```
Example:    SA
            OK,S212,S313   // alarm triggered by Sensors 212 and 313
```

Note: Command implemented for PULPO-2 (not existing in Pulpo)

Status Byte

Return a status byte. The staus byte is shown in hexadecimal form

```
Syntax: SB,n
        with n:1->34

Example:    SB,2            // read status byte 2 (shutter)
            OK,A0           // status byte 2 has bits 10100000
```

Shutter Close

Close shutter.

Syntax: SC

```
Example:    SC          // close shutter
            OK
```

Set Date for real time clock

Define or read the date in the real time clock

Syntax: SD[,dd/mm/yy]

```
Example:    SD,01/05/98    // set date to May 1, 1998
            OK
```

SEnd sensor value

Read the value of a sensor.

```
Syntax: SE,s
where   s=1-7,10-32                 => temperature sensors
        s=8                         => vacuum sensor
        s=9                         => heater current
        s=111-118, 121-128, 131-138 => sensors on ext Mux 1
        s=211-218, 221-228, 231-238 => sensors on ext Mux 2
        s=311-318, 321-328, 331-338 => sensors on ext Mux 3
        s=411-418, 421-428, 431-438 => sensors on ext Mux 4
```

To access a sensor on an external Mux, the command
EM,1
must have been issued before.

The reply is the value of sensor 'n' in floating point notation, or an error code if the sensor is out of range or not connected.

Shutter Identifier

Define or check the instrument shutter identifier.

```
Syntax: SI[,n]
where:  n=0,...,12
        n=0   no shutter board is connected
        n=1   shutter without status bit
        n=2   iris shutter
        n=3   iris, open status bit, no close status bit
        n=4   iris, open and close status bits
        n=5   iris, no open status bit, close status bit
        n=6   two blades, open status bit, no close status bit
        n=7   seso shutter
        n=8   emmi shutter
        n=9   Wide Field Imager shutter
        n=10  Efosc2 shutter
        n=11  Fors shutter
        n=12  OmegaCam shutter

Default value: 1

Example:    SI,1        // set shutter identifier to 1
            OK
            SI          // read shutter identifier
            OK,1        // shutter identifier is 1
```

Shutter Logic

Set or read the logic level to open the shutter.

```
Syntax: SL[,n]
where:  n=0,1
        n=0 : shutter open with low TTL level
        n=1 : shutter open with high TTL level

Default value: 0

Example:    SL,1        // set shutter logic to 1
            OK
            SL          // read shutter logic
            OK,1        // shutter logic is 1
```

Set Mode for shutter

Set or read the type of exposure to perform.

```
Syntax: SM[,m] with m:0->4
        m=0 : dark exposure
        m=1 : science exposure
        m=2 : operate bias LED instead of shutter
        m=3 : operate bias LED AND shutter at same time

Default value: 1

Example:    SM,1        // set shutter mode to 1
            OK
            SM          // read shutter mode
            OK,1        // shutter mode is 1
```

Set Point for heater temperature

Set or check the reference temperature (K) for a heater.

Syntax: SP,h[,t]
with h:1->8 and t: 77.0 -> 350.0

Default value: 300

```
Example:    SP,1,330    // set reference for heater 1 to 330 K
            OK
            SP,1        // read reference for heater 1
            OK,330
```

Self Recovery

Enable or disable the Self Recovery.

Syntax: SR[,n]
where n=0,1
n=0 : disable self recovery
n=1 : enable self recovery

Default value: 0

```
Example:    SR,1        // enable self recovery
            OK
            SR          // read self recovery
            OK,1
```

Note: Command implemented for PULPO-2 (not existing in Pulpo)

Self recovery Sensor

Set or read out the Sensor(s) used for Self Recovery.

Syntax: SS[,s[,t]]

Default value: -1

```
Example:    SS,8,19     // use sensors 8 and 19 for self recovery
            OK
            SS          // read sensors used for self recovery
            OK,8,19
            SS,19       // use sensor 19 for self recovery
            OK
            SS          // read sensor used for self recovery
            OK,19
```

Note: Command implemented for PULPO-2 (not existing in Pulpo)

Set Time for real time clock

Set or read the time on the real time clock.

Syntax: ST[,hh:mm:ss]

```
Example:    ST,10:00:00    // set time to 10 o'clock
            OK
            ST             // read time
            OK,10:00:05
```

Self recovery reference Value

Set or read out the Self recovery temperature reference Value (K).

Syntax: SV[,t]
with t: 77.0 -> 350.0

Default value: 273.15

```
Example:    SV,310      // set self recovery temperature to 310 K
            OK
            SV          // read self recovery temperature
            OK,310
```

Note: Command implemented for PULPO-2 (not existing in Pulpo)

Temperature Alarms

Enable or disable Temperature Alarms globally.

Syntax: TA[,v]
where n=0,1
n=0 : disable temperature alarms
n=1 : enable temperature alarms

Default value: 1

```
Example:    TA,1        // enable temperature alarms
            OK
            TA
            OK,1
```

Related commands: LL (Low Limit for alarm)
TT (Temperature Trip point)
AE (Alarm Enable)

Note: Command implemented for PULPO-2 (not existing in Pulpo)

Terminal Mode

Set serial link in Terminal Mode (characters received through the serial link are echoed back).
This mode is meant for working with PULPO connected to a local terminal

Syntax: TM

```
Example:    TM    // set serial link in Terminal Mode
            OK
```

Related commands: CM (Controller Mode)
EM (Echo Mode)

Trip Point for DAC

Set or read the DAC Trip Pont level (mA).

Syntax: TP[,f]
with f=50->1280

Default value: 1000

```
Example:    TP,100      // set DAC Trip Pont to 100 mA
            OK
            TP          // read DAC Trip Pont
            OK,100
```

Temperature Slope

Set or read the maximum temperature variation allowed per minute by the PIDs.

Syntax: TS[,f]
with f: 0.5 -> 10.0 (K/min)

Default value: 5

```
Example:    TS,7        // set max 7 K variation per minute
            OK
            TS          // read max temperature variation per minute
            OK,7
```

Temperature Trip point

Set or check the temperature (high limit) at which the temperature alarm is triggered.

```
Syntax: TT,s[,f]
where   s=1-6                       => temperature sensors
        s=10-32                     => temperature sensors
        s=111-118, 121-128, 131-138 => sensors on ext Mux 1
        s=211-218, 221-228, 231-238 => sensors on ext Mux 2
        s=311-318, 321-328, 331-338 => sensors on ext Mux 3
        s=411-418, 421-428, 431-438 => sensors on ext Mux 4
        s=1-6                       => temperature sensors
```

Default value: 350 K

```
Example:    TT,9,330    // set alarm high limit on sensor 9 to 330 K
            OK
            TT,9
            OK,330
```

Related commands: LL (Low Limit for alarm)
AE (Alarm Enable)
TA (Temperature Alarm)

VAcuum power

Turn the vacuum gauge off or on, or read the present status.

```
Syntax: VA[,v]
where   n=0,1
        n=0 : disable vacuum gauge
        n=1 : enable vacuum gauge

Default value: 1

Example:    VA,1        // enable vacuum gauge
            OK
            VA
            OK,1
```

Vacuum Identifier

Set or read the vacuum identifier.

```
Syntax: VI[,v]
where   n=1,2
        n=1 : Pfeiffer type
        n=2 : Edwards type

Default value: 1

Example:    VI,2        // set vacuum identifier to Edwards type
            OK
            VI
            OK,2
```

Vacuum Limit

Set or read the vacuum limit to trigger the Vacuum Alarm (in mB).

```
Syntax: VL[,f]

Default value: 1e0

Example:    VL,1e-02    // set vacuum alarm high limit to 10^-2 mB
            OK
            VL
            OK,1e-02

Related commands: LL (Low Limit for alarm)
                  AE (Alarm Enable)
```

Version Software

Return the PULPO software version

```
Syntax: VS

Example:    VS
            OK,V4.79
```

eXposure Time

```
Define the time for the next exposure.
If the exposure is active, a + or - sign in front of the value
increments or decrements the remaining time .
One restriction of this command is that you can add or subtract only
integer values but the initial exposure command accepts floating
point values.
The maximum exposure time is 16777 s (~4.6 hrs)

Syntax: XT[,[+/-]f
        with 0<f<16777 floating point number (s)

Default value: 0

Example:    XT,10.5     //define a 10.5 sec exposure
            OK
            >           // start the exposure
            OK
            XT,-3       // decrement the remaining time by 3 sec
            OK
            XT,+20      //increment the remaining time by 20 sec
            OK
            XT          // request remaining time
            OK,26.3     // still 26.3 seconds to go

Related commands: > (Start Exposure)
                  < (Abort Exposure)
                  PE (Pause Exposure)
```

6.5 Commands (short list)

HEATER-TEMPERATURE CONTROL LOOP		
Command	**Parameter Range**	**Description**
AT [,v]	v=0,1	turn PID Auto Tuner [off/on]
AV,s	s=1...32	AVerage value and deviation on sensor n
CS,h [,s]	h=1...8, s=1...438	Connect Sensor heater m [to sensor n]
DM [,v]	v=0...4	Deware Model [all ind. / 1-2 par / 2-3 par / 1-3 par / 1-2-3 par]
EM [,v]	v=0,1	Enable Multiplexer [off/on]
ES,s [,v]	s=1...438, v=0,1	Enable/disable Sensor readout
FV [,f]	0.0 <= f <= 1.0	Filter Value
HE,h [,v]	h=1...8, v=0...3	HEater [off / on / on 90% / on 45%]
HM,h [,v]	h=1...8, v=0...2	Heater Mode [linear / 1 s. duty cycle / 10 s. duty cycle]
HR,h	h=1...8	Heater Resistance
KD,h [,v]	h=1...8, 0<= v <= 200	PID derivative constant
KI,h [,v]	h=1...8, 0<= v <= 1000	PID integral constant
KP,h [,v]	h=1...8, 0<= v <= 1000	PID proportional constant
PW,h [,v]	h=1...8, 0<= v <= 100	percentage of PoWer delivered to heater
RO		Reset Overcurrent on heaters
RH		Reset Heaters
RT		Reset Temperature sensors
SE,s	s=1...438	SEnsor value
SP,h [,t]	h=1...8	Set Point temperature for heater
SR [,v]	v=0,1	Self Recovery enable [off/on]
SS [,s[,t]]	s,t=1...438	define Self recovery Sensor(s)
SV [,t]		Self recovery Value [temperature]
TP [,v]	50.0 <= v <= 1280.0	Trip Point for DAC
TS [,v]	0.5 <= v <= 10	Temperature Slope

VACUUM		
Command	**Parameter Range**	**Description**
RV		Reset Vacuum sensor
VA [,v]	v=0,1	VAcuum power [off/on]
VI [,v]	v=1,2	Vacuum Identifier [Pfeiffer/Edwards]

ALARMS		
Command	**Parameter Range**	**Description**
AE,s [,v]	s=0...438, v=0/1	Alarm Enable [s=0: global, v: off/on]
LL,s [,t]	s=1...438	alarm Lower Limit [temperature]
SA		Show triggered Alarms
TA [,v]	v=0/1	set global Temperature Alarm flag [off/on]
TT,s [,t]		Temperature Trip point - alarm upper limit [temperature]
VL [,p]		Vacuum upper Limit [pressure]

SHUTTER		
Command	**Parameter Range**	**Description**
>		start exposure
<		abort exposure
CD		Close Delay
OD		Open Delay
OS		Open Shutter
PE,v	v=0,1	Pause Exposure [continue/pause]
RS		Reset Shutter
SC		Shutter Close
SI [,id]	id=1...12	Shutter Identifier
SL [,v]	v=0,1	Shutter Logic
SM [,m]	m=0...3	Shutter Mode [dark / science / LED no shut. / LED and shut.]
XT [,t]	0 < t <= 16777.215	eXposure Time [s]

FRONT PANEL LEDS		
Command	**Parameter Range**	**Description**
DL [,v]	v=1/0	Disable Leds [leds off/on]

BIAS LED		
Command	**Parameter Range**	**Description**
BP [,v]	0.0 <= v <= 100.0	Bias Power

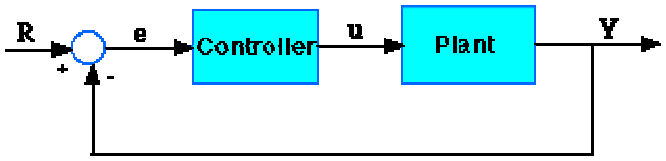
SYSTEM COMMANDS		
Command	**Parameter Range**	**Description**
SB,b	b=1...34	Status Byte
IN		INitialize pulpo (reboot)
SD [,date]	date=dd/mm/yy	Set Date for real time clock
ST [,time]	time=hh:mm:ss	Set Time for real time clock
VS		Version of Software

SYSTEM COMMANDS (Serial Port)		
Command	**Parameter Range**	**Description**
CM		Controller Mode
EC		ECho mode
TM		Terminal Mode

LOGGING		
Command	**Parameter Range**	**Description**
LB		Log Begin – store in log file from the beginning
LC		Log Check – check last element written in log file
LD		Log Dump
LF [,v]	v=0,1	Log to local File - enable/disable logging to Local File
LO [,t]	t>0	LOg interval [sec]
LR		Log Restart – store in log file from the first free place
LS		Log Stop

6.6 PID

PULPO-2 uses a common PID (Proportional, Integral, Derivative) control loop to control the power to the heaters.
Showing our system with the usual diagram:



(in our case, "Plant" is a heater), the transfer function is:

$$K_p + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_p s + K_I}{s}$$

KP, KI, KD can be defined using the KP, KI, KD commands (see the "Commands" chapter for the default values).

6.7 Logging

PULPO-2 keeps trace of performed actions, errors, warnings and modifications in the configuration, by writing the following files in the `/home/chris/pulpo2` directory:

`pulpoSystem.out`: it contains the list of the commands executed by pulpo2 (if Log to Local file enabled, see LF command) and the error and warning messages,

`pulpoConfig.out`: it contains the actual configuration values (heater settings, alarm settings, shutter settings, and so on).

With logging enabled (LB command), the following file is also created:

`pulpoLog.out`: it contains the values of the sensors and the heater powers periodically detected (at time intervals defined by the LO command).

Due to the dimension of the `pulpoLog.out` file, it can be retrieved only via the ethernet connection (e.g., using `ftp`).

6.8 Self Recovery

With Self Recovery enabled, the system is brought to a defined temperature whenever the alarm related to one or two defined sensors is triggered.

- Define one or two sensors whose triggered alarm should start the Self Recovery:

  ```
  SS,s
  ```

  ```
  SS,s,t
  ```

- Define the reference temperature which will be defined for all the active heaters, if the alarm for sensor "s" or sensor "t" (defined by SS,s,t) is triggered:

  ```
  SV,t
  ```

- Enable the Self Recovery:

  ```
  SR,1
  ```

With the above commands, all the enabled heaters will bring the system to the temperature `t`, whenever the sensor `s` triggers an alarm.

7 DIFFERENCIES WITH PULPO

7.1 Incompatibilities

With respect to the previous multifunctional module for the ESO cooling systems and shutters - Pulpo - PULPO-2 has the following incompatibilities:

- no LN_2 sensor and alarm
- DB command - DeBug mode - not available
- LD command - Log Dump - not implemented: due to the dimension of the log file, it can be retrieved only through the ethernet connection (e.g., via ftp)
- LO command has only one parameter (time interval)
- XD command - eXposure Delay - not available

7.2 New functionalities

With respect to the previous multifunctional module for the ESO cooling systems and shutters - Pulpo - PULPO-2 has the following new functionalities:

- enhanced number of temperature sensors and heaters (in particular SE, CS, HE commands enhanced)
- possibility to connect more heaters through external multiplexers (new EM command)
- self recovery (new SS, SV, SR commands)
- low level alarms implemented (new LL command)
- possibility to easily check the triggered alarms (new SA command)
- global temperature alarm flag (new TA command)

8 MAINTENANCE AND TROUBLESHOOTING

8.1 PULPO-2 software installation

To install a new version of the software on a PULPO-2 unit, the unit must be connected to the network.

a. retrieve the PULPO-2 software from the CMM archive

```
cmmCopy pulpo2
```

The PULPO-2 software is the `p2Server` program in the `pulpo2/p2/bin` directory

b. Ftp on the PULPO-2 unit as `root` and install the `p2Server` program in the directory

```
/home/chris/pulpo2
```

c. Set the appropriate permission mode

```
chmod 755 p2Server
```

d. Reboot PULPO-2 (see 8.3)

8.2 Sending commands to PULPO-2 from a SLCU terminal

Sending a command to a PULPO-2 unit from the terminal of a Sparc Local Control Unit (SLCU) workstation (running Solaris) can be performed

e. directly via the serial port, using the program `tip` (terminal interface processor):

```
tip hardwire
```

and then issuing the commands with the syntax explained in 6.4
To exit from the program `tip` use the keyboard combination `~.` (tilde dot).

f. using the Pulpo Server program:

msgSend -n "" fcdpServer <NUM> <CMD>[,<PAR>]

where:

- `<NUM>` is the PULPO-2 Id, as defined in the file: `$INS_ROOT/SYSTEM/COMMON/CONFIGFILES/$CCDNAME/pulpo.cfg`
- `<CMD>` and `<PAR>` are commands and parameters explained in 6.4

g. using the Pulpo Engineering Panel:

```
fcdpms &
```

and then issuing the commands with the syntax explained in 6.4

8.3 PULPO-2 reboot

Rebooting a PULPO-2 unit can be performed as follows:

a. the most drastic way to reboot a PULPO-2 unit (the only one, when the unit is not reachable via the network or via the SLCU serial port) is to power off and on the unit (e.g., disconnecting the power cable)

b. if PULPO-2 is connected to the network and answering to a `ping`, log on the unit as `root` and then run

```
reboot
```

c. if PULPO-2 is reachable via the serial port of a SLCU, issue the `IN` command from a SLCU terminal (see 8.1)

ANNEX A. Cable connector pinout

Pinout of SCSI 2 Connector: Sensors 1-5,10-18, Heater 1 – 4

Pin	Signal	Pin	Signal
1	Sensor 1 current +	35	Sensor 1 current -
2	Sensor 1 sense +	36	Sensor 1 sense -
3	Sensor 2 current +	37	Sensor 2 current -
4	Sensor 2 sense +	38	Sensor 2 sense -
5	Sensor 3 current +	39	Sensor 3 current -
6	Sensor 3 sense +	40	Sensor 3 sense -
7	Sensor 4 current +	41	Sensor 4 current -
8	Sensor 4 sense +	42	Sensor 4 sense -
9	Sensor 5 current +	43	Sensor 5 current -
10	Sensor 5 sense +	44	Sensor 5 sense -
11	Sensor 10 current +	45	Sensor 10 current -
12	Sensor 10 sense +	46	Sensor 10 sense -
13	Sensor 11 current +	47	Sensor 11 current -
14	Sensor 11 sense +	48	Sensor 11 sense -
15	Sensor 12 current +	49	Sensor 12 current -
16	Sensor 12 sense +	50	Sensor 12 sense -
17	Sensor 13 current +	51	Sensor 13 current -
18	Sensor 13 sense +	52	Sensor 13 sense -
19	Sensor 14 current +	53	Sensor 14 current -
20	Sensor 14 sense +	54	Sensor 14 sense -
21	Sensor 15 current +	55	Sensor 15 current -
22	Sensor 15 sense +	56	Sensor 15 sense -
23	Sensor 16 current +	57	Sensor 16 current -
24	Sensor 16 sense +	58	Sensor 16 sense -
25	Sensor 17 current +	59	Sensor 17 current -
26	Sensor 17 sense +	60	Sensor 17 sense -
27	Sensor 18 current +	61	Sensor 18 current -
28	Sensor 18 sense +	62	Sensor 18 sense -
29	N.C	63	N.C
30	N.C	64	N.C
31	Heater 4	65	Heater Power (24V)
32	Heater 3	66	Heater Power (24V)
33	Heater 2	67	Heater Power (24V)
34	Heater 1	68	Heater Power (24V)

***IMPORTANT:* This connector shares sensors 1-5 and Heater 1-3 with the 37pin SUB-D connector. Use only the DB connector *OR* the SCSI for these controls.**

Pinout of SCSI 1 Connector: Sensors 19 – 32 , Heater 5 – 8

Pin	Signal	Pin	Signal
1	Sensor 19 current +	35	Sensor 19 current -
2	Sensor 19 sense +	36	Sensor 19 sense -
3	Sensor 20 current +	37	Sensor 20 current -
4	Sensor 20 sense +	38	Sensor 20 sense -
5	Sensor 21 current +	39	Sensor 21 current -
6	Sensor 21 sense +	40	Sensor 21 sense -
7	Sensor 22 current +	41	Sensor 22 current -
8	Sensor 22 sense +	42	Sensor 22 sense -
9	Sensor 23 current +	43	Sensor 23 current -
10	Sensor 23 sense +	44	Sensor 23 sense -
11	Sensor 24 current +	45	Sensor 24 current -
12	Sensor 24 sense +	46	Sensor 24 sense -
13	Sensor 25 current +	47	Sensor 25 current -
14	Sensor 25 sense +	48	Sensor 25 sense -
15	Sensor 26 current +	49	Sensor 26 current -
16	Sensor 26 sense +	50	Sensor 26 sense -
17	Sensor 27 current +	51	Sensor 27 current -
18	Sensor 27 sense +	52	Sensor 27 sense -
19	Sensor 28 current +	53	Sensor 28 current -
20	Sensor 28 sense +	54	Sensor 28 sense -
21	Sensor 29 current +	55	Sensor 29 current -
22	Sensor 29 sense +	56	Sensor 29 sense -
23	Sensor 30 current +	57	Sensor 30 current -
24	Sensor 30 sense +	58	Sensor 30 sense -
25	Sensor 31 current +	59	Sensor 31 current -
26	Sensor 31 sense +	60	Sensor 31 sense -
27	Sensor 32 current +	61	Sensor 32 current -
28	Sensor 32 sense +	62	Sensor 32 sense -
29	N.C	63	N.C
30	N.C	64	N.C
31	Heater 8	65	Heater Power (24V)
32	Heater 7	66	Heater Power (24V)
33	Heater 6	67	Heater Power (24V)
34	Heater 5	68	Heater Power (24V)

Pinout of 37pin SUB-D connector: Sensor 1-5, Heater 1-3

This connector can be used to replace a PULPO1 unit with PULPO 2. It has the same pinout as the DB 37 on PULPO1.

Pin	Signal	Pin	Signal
1	Heater 1	20	Sensor 1 current +
2	Heater Power (24V)	21	Sensor 2 current +
3	Sensor 1 current –	22	Sensor 3 current +
4	Sensor 2 sense +	23	Sensor 3 sense +
5	Sensor 2 current -	24	Sensor 3 current -
6	Sensor 4 current +	25	NC
7	Sensor 4 sense +	26	NC
8	Sensor 4 sense -	27	Sensor 5 current -
9	Sensor 4 current -	28	NC
10	Sensor 5 current +	29	Heater 3
11	Sensor 5 sense +	30	Heater power (24V)
12	Sensor 5 sense -	31	Sensor 1 sense -
13	NC	32	Sensor 2 sense -
14	NC	33	Sensor 3 sense -
15	NC	34	NC
16	NC	35	NC
17	NC	36	NC
18	Heater 2	37	NC
19	Sensor 1 sense +		

Pinout for external sensor: Sensor 6 (normally used for FIERA box)

Pin	Signal	Pin	Signal
1	Sensor 6 current +	3	Sensor 6 sense -
2	Sensor 6 sense +	4	Sensor 6 current -

Pinout of the 24V power connector

Pin	Signal	Pin	Signal
1	+24V	3	Sense -
2	Sense +	4	24V_GND

Alarm outputs:

There are 2 alarm outputs for temperature alarm and 2 outputs for vacuum alarm. Each output offers a normally open relay contact. The connectors are 2 pin Lemo FFA.0S.302.CLAC42.

Pinout of the vacuum gauge connector

Pin	Signal	Pin	Signal
1	NC	4	VAC -*
2	GND	5	NC
3	VAC +*	6	VAC Power (24V)**

* 0-10V output of the vacuum gauge
** only if power to the vacuum gauge is enabled (VA command)

Pinout of the RS232 connector

Pin	Signal	Pin	Signal
1	NC	6	NC
2	Receive	7	NC
3	Transmit	8	NC
4	NC	9	NC
5	GND		

This pinout requires a Nullmodem cable for the connection to a terminal.

Pinout of the shutter connector (15pin High Density)

Pin	Signal	Pin	Signal
1	Fully OPEN status input	9	OptoOut 1 (spare open collector)
2	Fully CLOSE status input	10	OptoOut 2 (spare open collector)
3	REMOTE status input (for SESO)	11	Common Emitter for 8,9,10
4	FAIL status input	12	+5V for Bias LED
5	OptoGND	13	BIAS LED -
6	OptoGND	14	OptoIN 0 (spare)
7	OptoPower (+12V)	15	OptoIN 1 (spare)
8	OPEN COMMAND (open collector output)		

Pinout of the Digital I/O connector

Pin	Signal	Pin	Signal
1	Mux control 3 (Enable 0)	6	Mux control 5 (Enable 2)
2	Mux control 2 (Address 2)	7	Mux Power -15V
3	Mux control 1 (Address 1)	8	Mux Power +15V/5V *
4	Mux control 0 (Address 0)	9	AGND
5	Mux control 4 (Enable 1)		

* The voltage on this pin depends on the jumper setting on the board. It can be set to +15V or +5V. Values in brackets indicate usage of signals for the Omega Multiplexer boards.

ANNEX B. Practical Example (OmegaCam)

1. Reading the Sensor values:

To get the value of a sensor use:
SE,<Sensor number>

The reply will be one of the following:
OK,<Value> e.g.: SE,7 -> OK,273.1
ER,2 (Sensor doesn't exist) e.g.: SE,439 -> ER,2
ER,4 (Sensor not connected) e.g.: SE,5 -> ER,4
The sensor assignment is the following:
S1-5 PT100 connected via the DB37 or SCSI 1
S6 Fiera Box (LEMO connector)
S7 internal calibration resistor (always 273.1K)
S8 Vacuum sensor in mB
S9 Heater current in mA
S10-18 PT 100 via SCSI 1
S19-32 PT 100 via SCSI 2

If the multiplexer boards are connected (and enabled):
S111-118,S121-128,S131-138 for MUX board 1
S211-218,S221-228,S231-238 For MUX board 2
S311-318,S321-328,S331-338 For MUX board 3
S411-418,S421-428,S431-438 For MUX board 4

2. MUX boards:

Enable/Disable the Multiplexer boards with:
EM,<1-Enable or 0-disable>

For Omegacam:
EM,1 Check with EM should return OK,1

3. Heater setup

3.1 Control sensor assignment

Read or set the control sensor for a heater with:

CS,<Heater number>,<Sensor Number>

For Omegacam :
CS,1,217 Check with CS,1 should reply OK,217
CS,2,314 Check with CS,2 should reply OK,314
CS,3,114 Check with CS,3 should reply OK,114
CS,4,317 Check with CS,4 should reply OK,317

3.2 Setpoint4

Read or set the setpoint for each heater with:
SP,<Heater number>,<*Setpoint in Kelvin(!!)*>

For OmegaCam:

SP,1,153	Check with SP,1 should return OK,153.0
SP,2,153	Check with SP,2 should return OK,153.0
SP,3,153	Check with SP,3 should return OK,153.0
SP,4,153	Check with SP,4 should return OK,153.0

Note: The default setpoint for warmup is 310

3.3 Enable/Disable heaters4

Enable or disable the heaters with:
HE,<Heater number>,<0-disable or 1-enable>

For Omegacam:

HE,1,1	Check with HE,1 should return OK,1
HE,2,1	Check with HE,2 should return OK,1
HE,3,1	Check with HE,3 should return OK,1
HE,4,1	Check with HE,4 should return OK,1

4. Alarm setup

4.1 Vacuum alarm setpoint

Enter the setpoint with
VL,<Setpoint> Note: The format must be exponential

For OmegaCam:

VL,1.0e-03	Check with VL should return OK,1.0e-03

4.2 Enable/Disable the vacuum alarm

The vacuum sensor is connected as Sensor 8. To enable the vacuum alarm use:
AE,8,<1-enable or 0-disable>

For Omegacam:

AE,8,1	Check with AE,8 should return OK,1

4.3 Temperature alarm

To enter the high setpoint for a temperature alarm use:
TT,<Sensor number>,<Setpoint in Kelvin(!!)>

For Omegacam:

TT,217,180	Check with TT,217 should return OK,180
TT,114,180	Check with TT,114 should return OK,180
TT,314,180	Check with TT,314 should return OK,180
TT,317,180	Check with TT,217 should return OK,180
TT,19,155	Check with TT,19 should return OK,155

To enter the low setpoint for a temperature alarm use:
LL,<Sensor number>,<Setpoint in Kelvin(!!)>

For Omegacam:

LL,217,80	Check with LL,217 should return OK,80
LL,114,80	Check with LL,114 should return OK,80
LL,314,80	Check with LL,314 should return OK,80
LL,317,80	Check with LL,217 should return OK,80
LL,19,80	Check with LL,19 should return OK,80

4.4 Enable/Disable the temperature alarm

To enable/disable the temperature alarm use:
AE,<Sensor number>,<1-Enable or 0-Disable>

For OmegaCam:

AE,217,1	Check with AE,217 should return OK,1
AE,114,1	Check with AE,114 should return OK,1
AE,314,1	Check with AE,314 should return OK,1
AE,19,1	Check with AE,19 should return OK,1

4.5 Temperature Global Alarm

***IMPORTANT:* PULPO has a global temperature alarm which enables/disables all temperature alarms. As long as the global temperature alarm is not enabled no temperature alarm will be triggered even if all temperature alarms are enabled.**

To enable/disable the global temperature Alarm use:
TA,<1-Enable or 0-Disable>

For OmegaCam:

TA,1	Check with TA should return OK,1

4.6 Global Alarm

***IMPORTANT:* PULPO has a global alarm which enables/disables all alarms. As long as the global alarm is not enabled no alarm will be triggered even if vacuum alarm or temperature alarms are enabled.**

To enable/disable the global Alarm use:
AE,0,<1-Enable or 0-Disable>

For OmegaCam:

AE,0,1	Check with AE,0 should return OK,1

4.7 If an Alarm is triggered

To disable the alarms use:
AE,0,0
To check which alarms have been triggered use:
SA
To reenable the alarms use:
AE,0,1

For OmegaCam:

AE,0,0	Check with AE,0 should return OK,0
SA	Should return the triggered alarms
AE,0,1	Check with AE,0 should return OK,1

4.8 Enable/Disable Self Recovery

To enable/disable the Self Recovery use:
SS,<Sensor number>
SV,<Temperature>
SR,<1-Enable or 0-Disable>

For OmegaCam:

SS,8,19	Check with SS should return OK,8,19
SV,320	Check with SV should return OK,320
SR,1	Check with SR should return OK,1