E U R O P E A N S O U T H E R N O B S E R V A T O R Y

Organisation Européenne pour des Recherches Astronomiques dans l'Hémisphère Austral

Europäische Organisation für astronomische Forschung in der südlichen Hemisphäre

LA SILLA OBSERVATORY

HARPS

OS/ICS

Design Description

Doc.No. 3M6-TRE-HAR-33110-0003

Issue 0.2

Date 09/05/01

	Name	Date	Signature
Prepared	J.Guzman	09/05/01	
Approved	D.Mégevand	dd/mm/yy	
Released	M.Mayor	dd/mm/yy	

Change Record

Issue/Rev.	Date	Section/Page affected	Reason/Initiation/Document/Remarks
0.1	21/03/01	All	First preparation
0.2	09/05/01	All	Comments from A.Longinotti & D.Megevand added. Special device FITS configuration keywords added.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Purpose

This document contains the detailed design of HARPS ICS and it is intended to provide all the necessary information for the implementation and for the preparation of the test procedures.

Juan Carlos Guzman and Silvia Baeza from ESO La Silla, are involved in the development of this software module. The Work Package Manager from ESO La Silla is Ueli Weilenmann.

1.2 Scope

This document covers only the instrument software part related to OS and ICS. The MS is covered by the document “HARPS - MS Design Description“ [8]. The instrument software related with observations and calibration templates are covered by [9].

1.3 Applicable Documents

The following documents, of the exact issue shown, form a part of this document to the extent specified herein. In the event of conflict between the documents referenced herein and the contents of this document, the contents of this document shall be considered as a superseding requirement.

[1] VLT-SPE-ESO-17212-0001, 2.0 12/04/1995 --- VLT Software - VLT INS Software Specification

[2] VLT-SPE-ESO-17240-0385, 2.2/prep2 05/05/1998 --- VLT Software - VLT INS Common Software Specification

[3] VLT-PRO-ESO-10000-0228, 1.0 10/03/1993 --- VLT Software Programming Standards

[4] 3M6-TRE-HAR-33110-0002, 1.2 22/02/2001 --- HARPS OS/ICS Software Users Requirements and Design Report

1.4 Reference Documents

The following documents are referenced in this document.

[5] 3M6-PLA-HAR-33100-0005, 1.2 05/03/2001 --- HARPS Operation, Calibration and Maintenance Plan

[6] 3M6-TRE-HAR-33107-0001, 1.0 28/02/2001 --- HARPS Control Electronics Design Report

[7] 3M6-TRE-HAR-33110-0001, 1.3 02/03/2001 --- HARPS DFS Software User Requirements and Design Report

[8] 3M6-TRE-HAR-33110-0007, 0.1(in prep.) --- HARPS MS Design Description

[9] 3M6-TRE-HAR-33110-0005, 0.1 09/05/01 --- HARPS DFS Design Description

[10] GEN-SPE-ESO-00000-0266, 1.0 10/05/93 --- ESO Graphical User Interface Common conventions

[11] VLT-MAN-ESO-17240-0934, 2.3 01/03/2001 --- VLT INS Common Software - Base ICS User Manual

[12] VLT-MAN-ESO-17240-1973, 2.0 03/01/2001 --- VLT INS Software - Template Instrument User Manual

[13] VLT-MAN-ESO -17240-2265, 1.0 03/01/2001 --- VLT Software - BOSS User Manual

[14] VLT-MAN-ESO-17240-2153, 1.3 01/03/2001 --- VLT INS Common Software - Startup Tool User Manual

[15] VLT-MAN-ESO-17240-2325, 1.0 10/10/2000 --- VLT INS Common Software - Configuration Tool User Manual

[16]VLT-MAN-ESO-17240-1913, 1.4 01/03/2001 --- VLT Software - Installation Tool For VLT SW Packages User and Maintenance Manual

[17]VLT-SPE-ESO-13730-1817, 1.1 08/09/2000 --- VLT Software - FLAMES/GIRAFFE ICS Design Description

[18]VLT-MAN-ESO-17210-0669, 1.6 02/10/98 --- VLT Software - Motor Engineering Interface User Manual

[19]P.Ward, S.Mellor, Yourdon Press,1985 --- Struct. Development for Real-Time Systems

[20]J. Rumbaugh et. al., Prentice Hall,1991 --- Object-Oriented Modelling and Design

[21]PREMA Semiconductor GmbH--- Precision Thermometer PTM 3040 User Manual

[22]Lake Shore Cryotronics, Inc. 28/07/00--- Temperature Controller Model 331 User's Manual

1.5 Abbreviations and Acronyms

The following abbreviations and acronyms are used in this document::

CASE	Computer Aided Software Engineering
CCS	Central Control Software
FDDI	Fiber Distributed Data Interface
DBMS	Database Management System
HOS	High Level Operating Software
HW	Hardware
ICS	Instrument Control Software
IEEE	Institute of Electrical and Electronics Engineers
I/O	Input/Output
ISO	International Standardisation Organisation
LAN	Local Area Network
LCC	LCU Common Software
LCU	Local Control Unit
MIDAS	Munich Data Analysis System
N/A	Not Applicable
OSI	Open Systems Interconnection
ROS	Remote Operating Software
SCCP	Software Configuration Control Plan
SRS	Software Requirements Specification
SW	Software
TBC	To Be Confirmed
TBD	To Be Defined
TCS	Telescope Control Software
UIF	(Portable) User Interface (Toolkit)
VLT	Very Large Telescope
WAN	Wide Area Network
WS	Workstation

ICS	Instrument Control System
DCS	Detector Control System
OS	Observation Software
BOB	Broker for Observation Blocks
OB	Observation Block
MS	Maintenance Software

1.6 Glossary

No special definition is intruduced in this manual.

1.7 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.

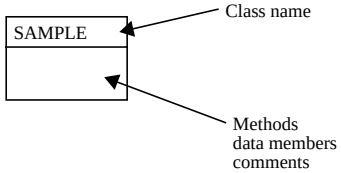
bold and *italic* are also used to highlight words.

1.7.1 Data Flow and Processor Model Diagrams

Data Flow and processor Model Diagrams are based on **De Marco/Yourdon** notation for real-time systems [19].

1.7.2 Class Diagrams

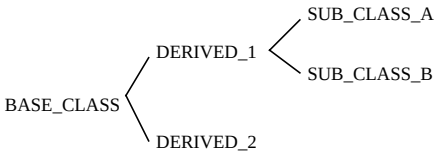
Class diagrams are based on OMT notation [20].
In order to simplify drawing of class symbols, a class is represented by a box, divided in two sections:



Lines connecting classes have to be interpreted as follows:

a ——— b generic association: a is associated to b
a ———● b multiple association: b (one or more) is used by a
a ——▷— b inheritance: a is a subclass of b
a ◇——● b multiple aggregation: b (one or more) is part of a

Class hierarchies are also represented as tree diagrams with the base class on the left and derived classes on the right, connected by lines:



1.8 Naming Conventions

This implementation follows the naming conventions as outlined in [3].

2 OVERVIEW

2.1 Introduction

The HARPS instrument is a High Accuracy Radial velocity for Planetary Search instrument dedicated to the search of extra-solar planets using precise radial velocity (RV) measurement. The instrument will be installed on the Cassegrain focus of the 3.60 m. telescope at La Silla Observatory.
HARPS OS is responsible for the coordination of all control activities related to a “single” exposure for its subsystems involved, namely HARPS instrument and CCD detector control system (FIERA). The corresponding control commands for such an exposure are received in an automatic way (e.g. via the *Sequencer* or the *Broker for Observation Blocks*, in short BOB) or interactively via a GUI.
An overview diagram of the environment in which HARPS OS is operating is shown in Figure 2-1.

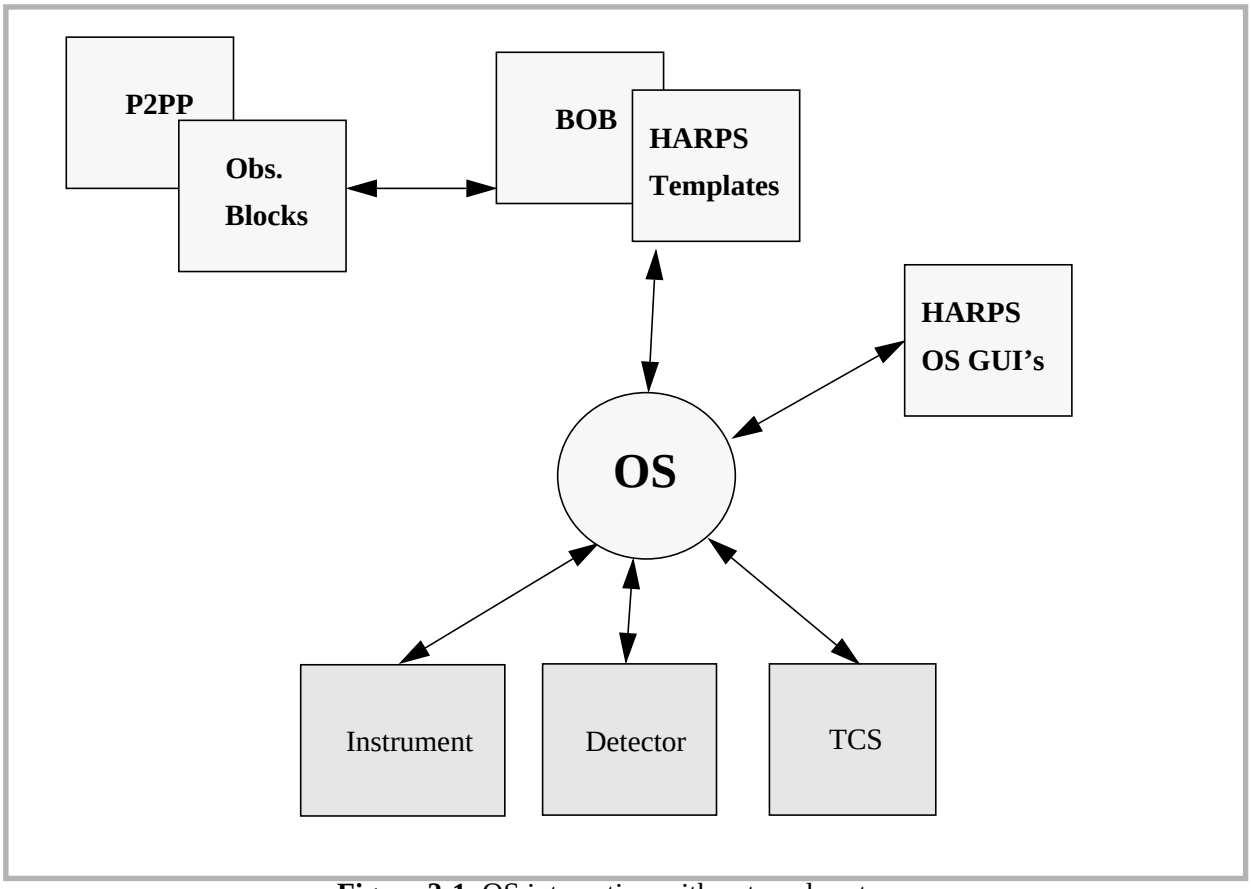


Figure 2-1: OS interaction with external systems

During normal operation at the telescope, the execution of exposures is triggered by Templates. The Templates are driven by BOB which receives Observation Blocks (OBs) from the Phase II Preparation (P2PP) or Scheduling software. The Templates send commands like setting up the instruments and detectors, presetting the telescope, starting an exposure, etc. to the HARPS OS which coordinates with its subsystems the execution. The HARPS OS GUI displays the global status of HARPS and also contains a few buttons to abort or end exposure, or change the exposure time of a running exposure because these buttons are not available in BOB.

The HARPS Instrument Control Software (ICS) is responsible for controlling and monitoring the various components of the HARPS instrument such as atmospheric dispersion corrector, carriage units, calibration unit, instrument environmental conditions, status and alarms.

Furthermore, it interacts with external systems such as OS, GUI's and Templates as shown in Figure 2-2.

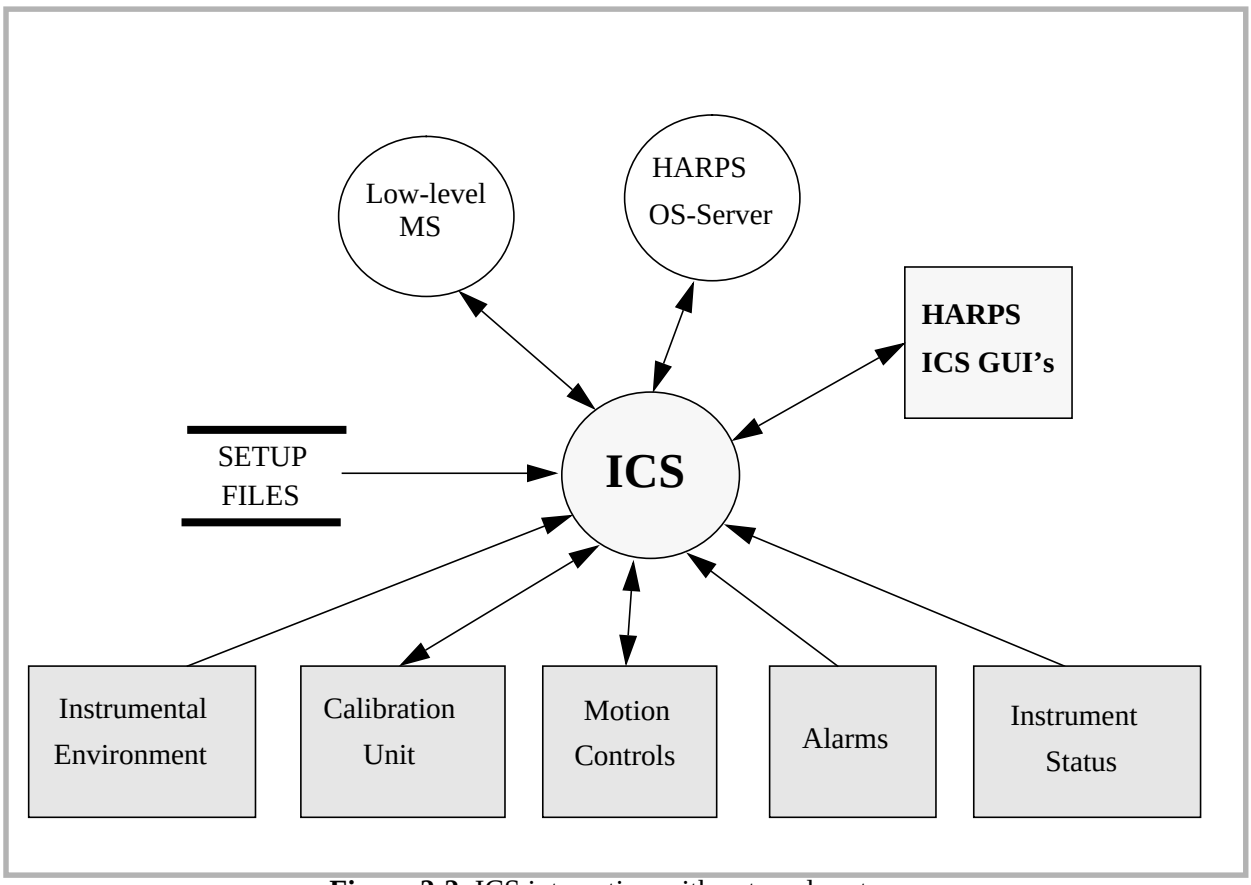


Figure 2-2: ICS interaction with external systems.

The instrument functions are driven by standard ESO hardware such as motor controllers and digital I/O boards (see [6]). The hardware is controlled by LCU software of the HARPS ICS which is based on ESO standard software (LCC, MCM, ICS base module [11]). The workstation software of the HARPS ICS controls and monitors the LCU software as shown in Figure 2-2 which gives an overview of the ICS process structure.
Each hardware function has a corresponding software device. The individual software devices understand the ICS standard commands which control the HW device.
The LCU server process communicates with the server process on the WS. It receives commands from the WS Server and distributes them to the individual software devices for execution.
The WS Server takes care of handling requests from outside. This could either be a user command from a GUI or a process such as the HARPS OS-Server which coordinates all subsystems of the instrument (HARPS instrument and FIERA).
The ICS WS Server checks the validity of commands and handles setup files as well as setup keywords. Commands are either forwarded to the LCU Server (normal mode) or to the Simulator process on the WS when ICS is set to simulation mode. Simulation is usually used for software development and maintenance.
The WS database is updated frequently via the scan system from the LCU local database, with values reflecting the status of the ICS hardware. These values are displayed in GUI panels visualizing the current status. Special events, error conditions or alarms are reported to the WS Server.

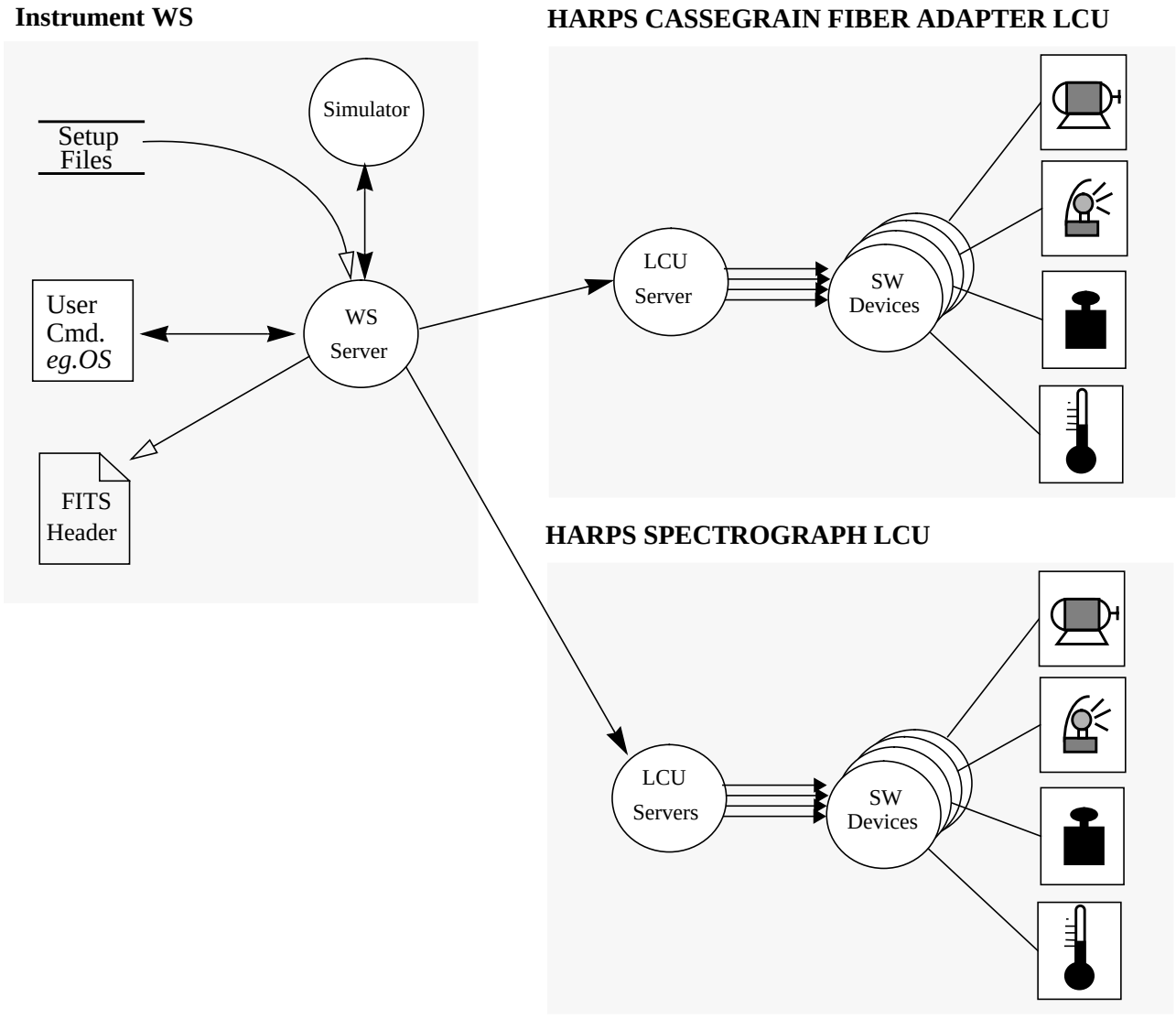


Figure 2-3: ICS process structure overview.

The rest of the manual is organized in the following chapters:

- Chapter 3 describes the architecture of the HARPS OS and ICS sub-systems.
- Chapter 4 describes the data produced and maintained by OS and ICS.
- Chapter 5 describes the features of OS and ICS (i.e., states and substates, commands, start-up and shutdown).
- Chapter 6 describes the OS GUI's.
- Chapter 7 describes the development and test factors.
- Chapter 8 describes the verification matrix.
- Chapter 9 is used for references (dictionaries and configuration files).

2.2 Hardware Environment

The WS platform consists of:

- HP Workstation, model B2000

The HARPS Cassegrain Fiber Adapter (HCFA) LCU platform consists of (see [6]):

- 1 CPU Motorola MVME2604-4331
- 1 Transition Module MVME712M
- 2 Motor controller MACCON MAC4-INC
- 2 Servo Amplifier ESO VME-4SA
- 1 Digital I/O board ACROMAG AVME-9481
- 1 Triple Voltage power supply KNIEL 358-006-02
- 2 Single Voltage power supply KNIEL 311-053-02
- 1 ISER - serial ports interface

The HARPS Spectrograph LCU platform consists of (see [6]):

- 1 CPU Motorola MVME2604-4331
- 1 Transition Module MVME712M
- 1 Motor controller MACCON MAC4-INC
- 1 Servo Amplifier ESO VME-4SA
- 1 Digital I/O board ACROMAG AVME-9481
- 1 Triple Voltage power supply KNIEL 358-006-02
- 2 Single Voltage power supply KNIEL 311-053-02
- 1 ISER - serial ports interface

2.3 Software Environment

The WS Software is developed on and for a Unix environment. HARPS OS will be based on BOSS (see [13]), which is a generic OS on its turn based on the CCS/evh toolkit. The GUIs will be built using the VLT Panel Editor.

The major pieces of software installed and running on the HARPS Instrument Workstation (IWS) are:

- UNIX operating system (HP-UX 11.0).
- VLT Common Software (VCS), including the Central Control Software (CCS).
- HARPS Instrument Control Software (ICS).
- HARPS Detector Control Software (DCS): FIERA.
- HARPS Observation Software (OS).
- VLT On-line Archive Client (volac, vcsolac).
- Real-Time Display (RTD).
- Broker for Observation Blocks (BOB).
- Optionally TCS in simulation mode.

2.3.1 CCS Environments

Three environments are foreseen for HARPS instrument software. Those environments are listed in Table 2-1.

Table 2-1: HARPS CCS environments

Name	Platform	Type	Purpose
wharps	WS	QSEMU	Main HARPS CCSLite environment. Processes running on this are: - OS, DCS WS and ICS WS part; - BOB and templates; - Archiving process (volac & vcsolac).
lhaics1	LCU	LCC	Contains the ICS LCU frontend and device control processes for HARPS CFA functions.
lhaics2	LCU	LCC	Contains the ICS LCU frontend and device control processes for HARPS Spectrograph functions.

2.4 Software Modules

The architecture of the HARPS OS is based on BOSS (see [13]) which is independent of any specific instrument. The architecture of the HARPS ICS is based on a generic standard instrument package (see [12]) which is independent of any specific instrument.
The software modules are listed in Table 2-2

Table 2-2: HARPS software modules

Module-Name	Platform	System	Description
hains	WS	BOTH	HARPS software installation module
hamcfg	WS+LCU	BOTH	All configuration files for the HARPS instrument.
dicHARPS	WS	BOTH	HARPS dictionary.
hao	WS	OS	The HARPS OS specific module
haopan	WS	OS	HARPS OS GUI's (panels and tcl library)
hai	WS	ICS	WS Server (frontend) and Simulator process
haipan	WS	ICS	ICS GUI's (panels and tcl library)
hailtc	LCU	ICS	Lakeshore Temperature Controller special device
haiprt	LCU	ICS	PREMA Thermometer special device

2.5 Standards

The standards defined in the "VLT Programming Standards Specification" and the guidelines described in the "LCU Common Software - User Manual" and "Guidelines for VLT Applications" for the development of LCU applications shall be applied.

3 ARCHITECTURE

3.1 Standards

ICS WS front-end and LCU servers are based on Base ICS. The design and implementation will follow the procedure described in [11].
OS is based on Base OS Stub (see [13]). The standard BOSS implementation fulfill the user requirements for HARPS OS system.

3.2 HARPS ICS Workstation Part

The architecture of ICS software on the workstation will be build using object oriented design principles and Base ICS. The WS processes that are part of HARPS ICS (`haiControl` and `haiSimControl`) have the same general architecture, based on the event handling toolkit. The communication of processes with each other is handled in an event-driven way by instances of the evhHANDLER class. This object receives the incoming messages (commands, events, signals and replies) and dispatches them to the destination objects, invoking the properly installed callbacks.

3.2.1 ICS Simulator

The ICS Simulator is a simulation engine which main purpose is to simulate the ICS LCU in order to test and evaluate WS software. Simulation of the ICS LCU basically means accept commands and provide replies as if it were the LCU and provide the same data in the OLDB which normally is provided from the LCU via the scan system.
This kind of simulation mode is useful mainly during the development phase especially when no LCU is available.
When the operational mode is set to simulation at workstation level the `haiControl` process redirects commands which are normally sent to the LCU to the process `haiSimControl`. The replies are formatted from reply strings stored in tables in the simulation branch of the database.
The simulation of database values is done by changing the WS database values that normally are scanned from the LCU.
The `haiSimControl` process is based on `icbSimControl` (see [11]). There is no special feature of the simulation process foreseen for HARPS.

3.2.2 ICS WS Server

The ICS WS Server `haiControl` is the main controlling process of the HARPS ICS. Any access to ICS has to go through this process, so it is the front-end for any ICS command from outside systems such as OS, MS as well as the ICS stand-alone user interface.
The `haiControl` is based on `ic0Control` and the main tasks are:

- Forward command to LCUs.
- For software device commands, involving FITS keywords, split this list and forward them to each LCU only the sub-set under its responsibility.
- On request, collect status information from the LCUs and store it into a partial FITS header file.
- Perform the calculation of the required statistics values of sensors and store it in the FITS file.
- Handle alarms and exceptions

Commands for performing an action on the HW functions are forwarded to the ICS LCU but they are validated in terms of parameters and state validity, thus only valid commands are forwarded to the ICS LCU.

3.2.3 Special classes for HARPS

The figure 1 shows the class diagram for haiControl process. Since the ICS WS front-end will be implemented based on ic0Control, only two classes will be defined: haiSTATISTIC and haiINS_SENSOR.

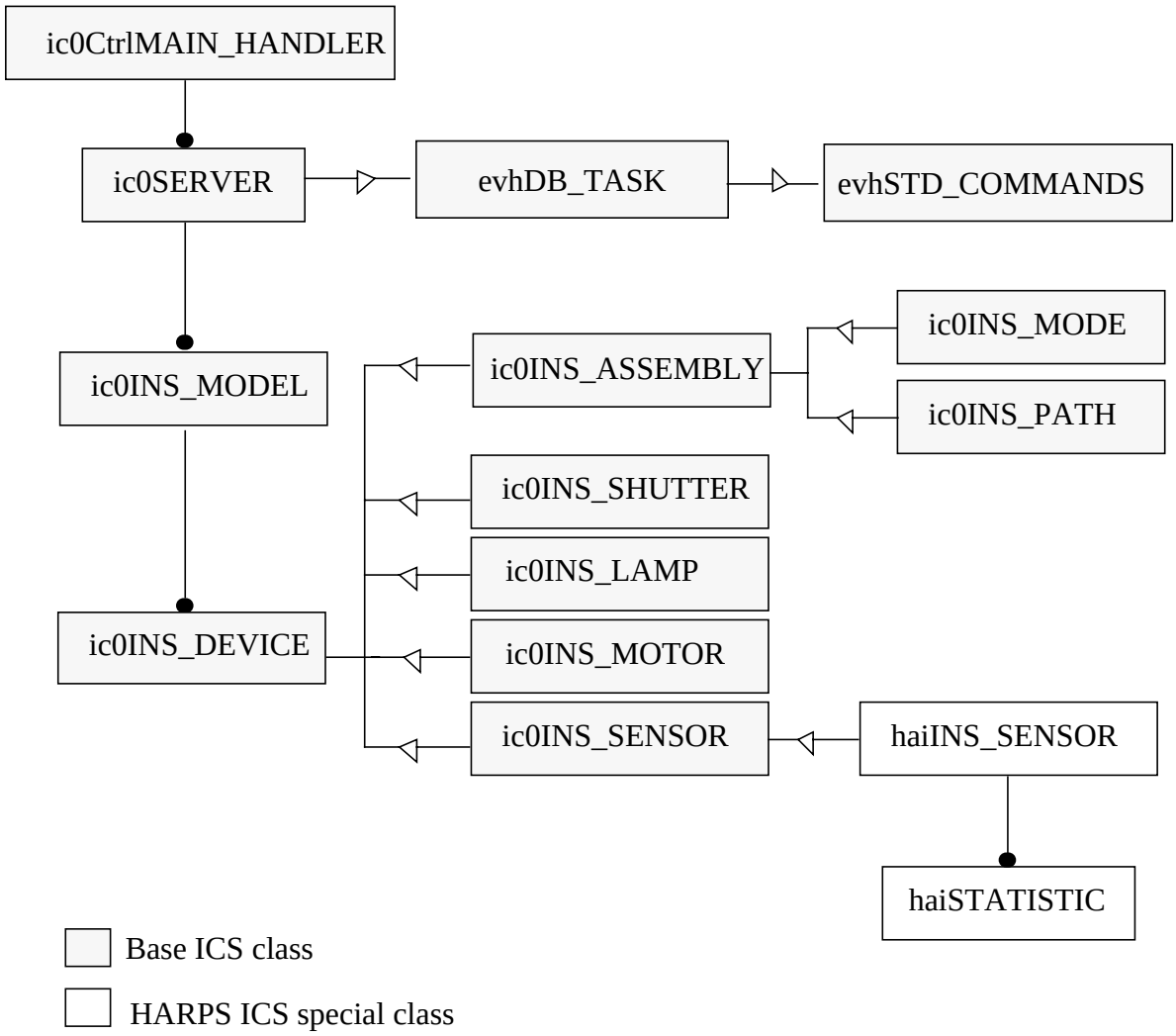


Figure 3-1: HARPS ICS WS classes

- ***haiSTATISTIC***. This class implements the calculation of the statistics values (minimum, maximum, mean and first derivative) for HARPS sensors. The methods are described in the following section.
- ***haiINS_SENSOR***. This is a sub-class of ic0INS_SENSOR, which will just overload the DevTrigger method to start the calculation of the statistics values for HARPS sensor.

3.2.4 Special methods for HARPS classes

3.2.4.1 haiSTATISTIC methods

haiSTATISTIC::Start

reads device configuration, in particular the period of value update,
initialize statistics values (min., max., mean, etc.) on database,
set timer object to period,
attach calculation callback to timer event,
exit.

haiSTATISTIC::Stop

detach calculation callback from timer event,
exit

haiSTATISTIC::Calculation_callback

record time and value,
if value < min then
    min = value,
if value > max then
    max = value,
calculate and register mean value,
calculate and record first derivative,
exit.

3.2.4.2 haiINS_SENSOR methods

Following the implementation of a “special action associated to a particular keyword” listed in [11] (section 7.5.4), the method DevTrigger corresponding to haiINS_SENSOR class will be overload.
This method will be triggered when EXPSTART/EXPEND command is received by WS front-end server. The main purpose of the overloaded method is the calculation of the statistics for all the HARPS sensors values.

hainsINS_SENSOR::DevTrigger

if (command equal EXPSTART) then,
    create haiSTATISTIC instance object, named statistic,
    run method start on statistic object
else if (command equal EXPEND) then,
    run method stop on statistic object,
    delete statistic object.
exit

3.3 HARPS ICS LCU Part

The HARPS ICS LCU software will be implemented with the software supplied by the Base ICS modules. The main architecture for HARPS is described here.

3.3.1 HARPS Base ICS standard devices

The HARPS components are listed below. Each component has a corresponding software device to control the hardware. A component is identified with an acronym or with its related FITS prefix e.g. `INS.ADCi` for the atmospheric dispersion corrector component.

Table 3-1: HARPS Base ICS standard devices

Acronym	Description	Category	FITS prefix	ICB class	LCU
mirr	Calibration mirror & fiber viewer shield	Motor linear	INS.MIRR1	icbMOT_MIRROR	HCFA
adc1	ADC prism1	Motor circular	INS.ADC1	icbMOT_ADC	HCFA
adc2	ADC prism2	Motor circular	INS.ADC2	icbMOT_ADC	HCFA
adcs	ADC slide	Motor linear	INS.ADCS	icbMOT_OPTI	HCFA
fhsel	Fiber heads selector	Motor linear	INS.OPTI2	icbMOT_OPTI	HCFA
dcvr	Fiber heads protector (dust cover)	Motor linear	INS.OPTI3	icbMOT_OPTI	HCFA
iodc	Iodine cell carriage	Motor linear	INS.OPTI4	icbMOT_OPTI	HCFA
filt	Neutral density wheel	Motor circular	INS.POS1	icbMOT_POS	HCFA
cfas	HCFA body temperature sensor	Serial	INS.TEMP1	icbSEN_ADAM	HCFA
iods	Iodine cell temperature sensor	Serial	INS.TEMP2	icbSEN_CN76000	HCFA
casel	Calibration Lamp A selector	Motor linear	INS.OPTI5	icbMOT_OPTI	HARPS
cbsel	Calibration Lamp B selector	Motor linear	INS.OPTI6	icbMOT_OPTI	HARPS
lamp1 to lamp5	Calibration Lamps	Signal	INS.LAMPi	icbLAMP	HARPS
fflamp	Flat field led inside vacuum vessel	Signal	INS.LAMP6	icbLAMP	HARPS
shcla	Calibration Lamp A shutter	Signal	INS.SHUT1	icbSHUTTER	HARPS
shclb	Calibration Lamp B shutter	Signal	INS.SHUT2	icbSHUTTER	HARPS
expm	Exposure meter	Serial	INS.DET2	icbSEN_EXPM	HARPS

Acronym	Description	Category	FITS prefix	ICB class	LCU
prss	Vacuum vessel pressure sensor	Serial	INS.SENS1	icbSEN_ADAM	HARPS
lnlev	LN2 level sensor	Serial	INS.SENS2	icbSEN_ADAM	HARPS

3.3.2 HARPS special devices

The table 2 list all the HARPS special devices not-supported by Base ICS. The implementation effort will be directed to control this specific hardware, since Base ICS provides the control of the standard devices.

Table 3-2: HARPS special devices

Acronym	Description	Hardware	FITS prefix	ICS LCU server	LCU
tmcc	CCD Temperature control	Lakeshore 331S	INS.SENSOR3	hailtcServer	HARPS
tmcv	Vacuum vessel temperature control (Enclosure box temp. control)	Lakeshore 331S	INS.SENSOR4	hailtcServer	HARPS
rvvs	HARPS room and vacuum vessel temperature sensors (14 sensors)	PREMA 3040	INS.SENSOR5 (INS.TEMP3 to INS.TEMP17)	haiprtServer	HARPS
spgs	Spectrograph temperature sensors (13)	PREMA 3040	INS.SENSOR6 (INS.TEMP18 to INS.TEMP31)	haiprtServer	HARPS

3.3.3 Lakeshore Model 331 Temperature Controller (hailtc module)

This ICS special device controls the Lakeshore Model 331 temperature controllers connected to one single serial port. Each controller allows to connect 2 sensor (diode/resistor/thermocouple) inputs and heater outputs. Each controller can report two values: current value and set value.
In Table 3-3 are listed the FITS configuration keywords for this special device, some of them are the same as standard ICS sensors, in that case the reference to the section of the Base ICS document is shown.

Table 3-3: LakeShore Model 331 Temp. Controller Config. Keywords

Section	Keyword	Type	Description
8.10.3 [11]	INS.SENSORi.DEVNAME	Design	Acronym of the ICS software device.
	INS.SENSORi.DEVDESC	Normal	Description of the ICS software device.
	INS.SENSORi.LCUID	Design	LCU number of the LCU managing the special device (1..n).
	INS.SENSORi.SWSIM	Normal	Device simulation flag (T/F).
8.13.1.1 [11]	INS.SENSORi.DEVTYPE	Design	Device type (value=”LKSH331S”).
8.13.1.2 [11]	INS.SENSORi.PORT	Normal	Hardware device (e.g. “/iser0”).
8.13.1.3 [11]	INS.SENSORi.NUM	Normal	Number of managed sensor inputs.
	INS.SENSORi.NAMEi	Normal	Sensor value name (acronym).
	INS.SENSORi.DESCi	Normal	Sensor value description.
	INS.SENSORi.HEADERi	Normal	If T, report sensor value in the image FITS header.
	INS.SENSORi.FITSi	Normal	Sensor value FITS keyword.
	INS.SENSORi.SENUNITi	Normal	Sensor value unit, e.g. “K”.
3.5.3.1	INS.SENSORi.INPUTi	Design	Input channel used by managed sensor (A or B).
3.5.3.2	INS.SENSORi.SETPi	Normal	Temperature setpoint.

3.3.3.1 Input channel

The input channel specify where the sensor are connected to the controller. The values could be A or B.

3.3.3.2 Temperature setpoint

The temperature setpoint set the reference temperature value to be controlled. The value should be in INS.SENSORi.SENUNITi units.

3.3.3.3 Serial port

The serial port to access the Lakeshore 331S temperature controllers is preconfigured as follows:
Baudrate=9600; CharBits=1 Start, 7 Data, 1 Parity, 1 Stop; Parity=Odd; Transmission mode=Half Duplex.

3.3.4 PREMA Thermometer (haiprt module)

This ICS special device access the PREMA thermometer connected to one serial port. Each PREMA thermometer is allowed to read up to 18 platinum sensors or 34 thermocouples.
In Table 3-4 are listed the FITS configuration keywords for this special device, some of them are the same as standard ICS sensors, in that case the reference to the section of the Base ICS document is shown.

Table 3-4: PREMA Thermometer Configuration Keywords

Sec.	Keyword	Type	Description
8.10.3 [11]	INS.SENSORi.DEVNAME	Design	Acronym of the ICS software device.
	INS.SENSORi.DEVDESC	Normal	Description of the ICS software device.
	INS.SENSORi.LCUID	Design	LCU number of the LCU managing the special device (1..n).
	INS.SENSORi.SWSIM	Normal	Device simulation flag (T/F).
8.13.1.1 [11]	INS.SENSORi.DEVTYPE	Design	Device type (value=”LKSH331S”).
8.13.1.2 [11]	INS.SENSORi.PORT	Normal	Hardware device (e.g. “/iser0”).
8.13.1.3 [11]	INS.SENSORi.NUM	Normal	Number of managed sensor inputs.
	INS.SENSORi.NAMEi	Normal	Sensor value name (acronym).
	INS.SENSORi.DESCi	Normal	Sensor value description.
	INS.SENSORi.HEADERi	Normal	If T, report sensor value in the image FITS header.
	INS.SENSORi.FITSi	Normal	Sensor value FITS keyword.
	INS.SENSORi.SENUNITi	Normal	Sensor value unit, e.g. “K”.
3.5.4.1	INS.SENSORi.SENADDRi	Normal	Sensor channel address (xyy).

3.3.4.1 Sensor channel address

The channel address must have the format ‘xyy’, where ‘x’ identifies the type of the sensor, and ‘y’ the channel number to be retrieved:

x = T: Thermocouple selected.

x = R: Resistor thermometer sensor selected.

yy = A: Front channel A.

yy = B: Front channel B.

yy = AB: X-B activated with channel A.

yy = 01 to 32: rear channels 01 to 32.

yy = AZ: Auto-zero channel.

Special channels are entered as following:

xyy = CJ: Cold junction channel.

xyy = D01 to D32: X-B activated with rear channels 01 to 32.

3.3.4.2 Serial port

The serial port to access the PREMA thermometer is preconfigured as follows:
Baudrate=9600; CharBits=8 Data, No parity, 1 Stop.

3.3.5 ICS LCU Server

The ICS LCU server `icblcuServer` is the frontend of ICS at the LCU. It is responsible for handling commands received from outside the LCU. These are normally sent from the ICS WS frontend process.
The ICS LCU server is an instance of the LCC command interpreter. which translates commands received from the WS into corresponding commands to the addressed software devices and collect their replies. A typical command is SETUP which contains several keyword/value pairs for different software devices.

According with Table 3-2, HARPS will have special devices not supported by Base ICS. Each of this special device will have their own server, handling the specific actions. The name of the LCU server are shown in Table 3-2.

3.4 HARPS OS Architecture

The HARPS OS is based on the Base OS Stub (BOSS) - see [13].

3.4.1 Instrument modes

The various observing and calibration modes are defined in and are summarized in the following tables. The modes can be easily expanded or modified with the OS-server configuration file (ASCII file, see 9.4).

Table 3-5: HARPS observation modes

Mode	Description	Setup
SIM_WAVE_REF	Simultaneous wavelength reference	INS.MIRR1.NAME FIBB INS.OPTI4.NAME OUT INS.OPTI6.NAME THAR INS.LAMP1.ST TRUE INS.SHUT2.ST TRUE
SPEC_I2CELL	Spectroscopy with Iodine cell	INS.MIRR1.NAME NONE INS.OPTI4.NAME IN
SPEC_OBJECT	Spectroscopy with object only	INS.MIRR1.NAME NONE INS.OPTI4.NAME OUT
SPEC_OBJ_SKY	Spectroscopy with object and sky	INS.MIRR1.NAME FIBB INS.OPTI4.NAME OUT INS.OPTI6.NAME FIBERFP INS.LAMP5.ST TRUE INS.SHUT2.ST TRUE

Table 3-6: HARPS calibration modes

Mode	Description	Setup
DARK	Dark and bias	INS.MIRR1.NAME NONE INS.OPTI4.NAME OUT INS.OPTI3.NAME IN

Mode	Description	Setup
WAVE_CAL	Wavelength calibration	INS.MIRR1.NAME BOTH INS.OPTI4.NAME OUT INS.OPTI5.NAME THAR INS.OPTI6.NAME THAR INS.LAMP1.ST TRUE INS.SHUT1.ST TRUE INS.SHUT2.ST TRUE
IOD_SPEC	Iodine spectrum	INS.MIRR1.NAME NONE INS.OPTI4.NAME IN
LOC_ORDER_A	Location orders fiber A	INS.MIRR1.NAME FIBA INS.OPTI4.NAME OUT INS.OPTI5.NAME WHITE INS.LAMP2.ST TRUE INS.SHUT1.ST TRUE
LOC_ORDER_B	Location orders fiber B	INS.MIRR1.NAME FIBB INS.OPTI4.NAME OUT INS.OPTI6.NAME WHITE INS.LAMP2.ST TRUE INS.SHUT2.ST TRUE
FLAT_FIELD	Flat-field	INS.MIRR1.NAME BOTH INS.OPTI4.NAME OUT INS.OPTI5.NAME WHITE INS.OPTI6.NAME WHITE INS.LAMP2.ST TRUE INS.SHUT2.ST TRUE

From the tables above, we conclude that there are two instrument modes:

- **SPECTROSCOPY WITH BOTH FIBERS.** This instrument mode is related with observations using both fibers. The common setup for this instrument mode is the following:

  INS.OPTI2.NAME HARPS

  INS.OPTI4.NAME OUT

  Under this instrument mode, several observation and calibration modes could be used by corresponding templates: Flat-field, Location orders fiber A or B, simultaneous wavelength reference and spectroscopy with object and sky.

- **SPECTROSCOPY WITH SINGLE FIBER.** This instrument mode use just one fiber. The common setup is the following:

  INS.OPTI2.NAME HARPS

  INS.MIRR1.NAME NONE

  Spectroscopy with iodine cell and object only; and iodine spectrum calibration mode could be group under this instrument mode.

3.4.2 OS classes

The haoSERVER class contains the main class implementing most of the behavior of the OS. It is based on xxoSERVER (belonging to the template instrument OS, xxo), which on its turn is derived from bossSERV-

ER. This already provides default behavior for:

- handling of the standard commands
- handling of incoming Setup files and setup keywords, checking of individual keywords (correctness of values) and of the whole keywords (completeness and consistency) and forwarding them to the appropriate sub-system,
- handling of Application Configuration File to configure and set-up,
- preferences for the OS-Server process,
- handling of instrument modes,
- handling of states and sub-states, state transitions, as well as state alignment according to the sub-systems,
- handling of sub-system communication: sending synchronous and asynchronous commands, synchronization of replies.

BOSS provides hooks for the pre-and/or post-processing for various commands.

3.4.3 Overloaded BOSS methods

No overloaded BOSS methods are foreseen for HARPS.

3.4.4 HARPS OS Server configuration file

The HARPS OS configuration part is included on HARPS Instrument configuration file (hamcfgINS.cfg, see 9.4). This file is read during start-up of the central HARPS OS server process (haoControl, adapted from the template instrument OS server xxoControl).

4 DATA DESCRIPTION

4.1 ICS Dictionary

Since HARPS instrument software is based on the Template Instrument (XXXX, see [12]), in the dicHARPS module (based on dicXXXX module) there are three files which contains a set of keywords for subsystems definition. For ICS the file dicHARPS_ICS.txt contains the ICS keyword set to generate the ICS dictionary named ESO-VLT-DIC.HARPS_ICS, after run the make under sources directory.
The dicHARPS_ICS.txt file is shown in 9.2.

4.2 OS Dictionary

Currently there are no special keywords defined for the HARPS OS. HARPS OS will load (and “understand”) the keywords of the dictionaries provided by BOSS, HARPS-ICS and FIERA (see 9.3).

4.3 HARPS INS Database

The On-line Database shall be designed according to [11]. An example of the structure is shown in Figure 4-1.

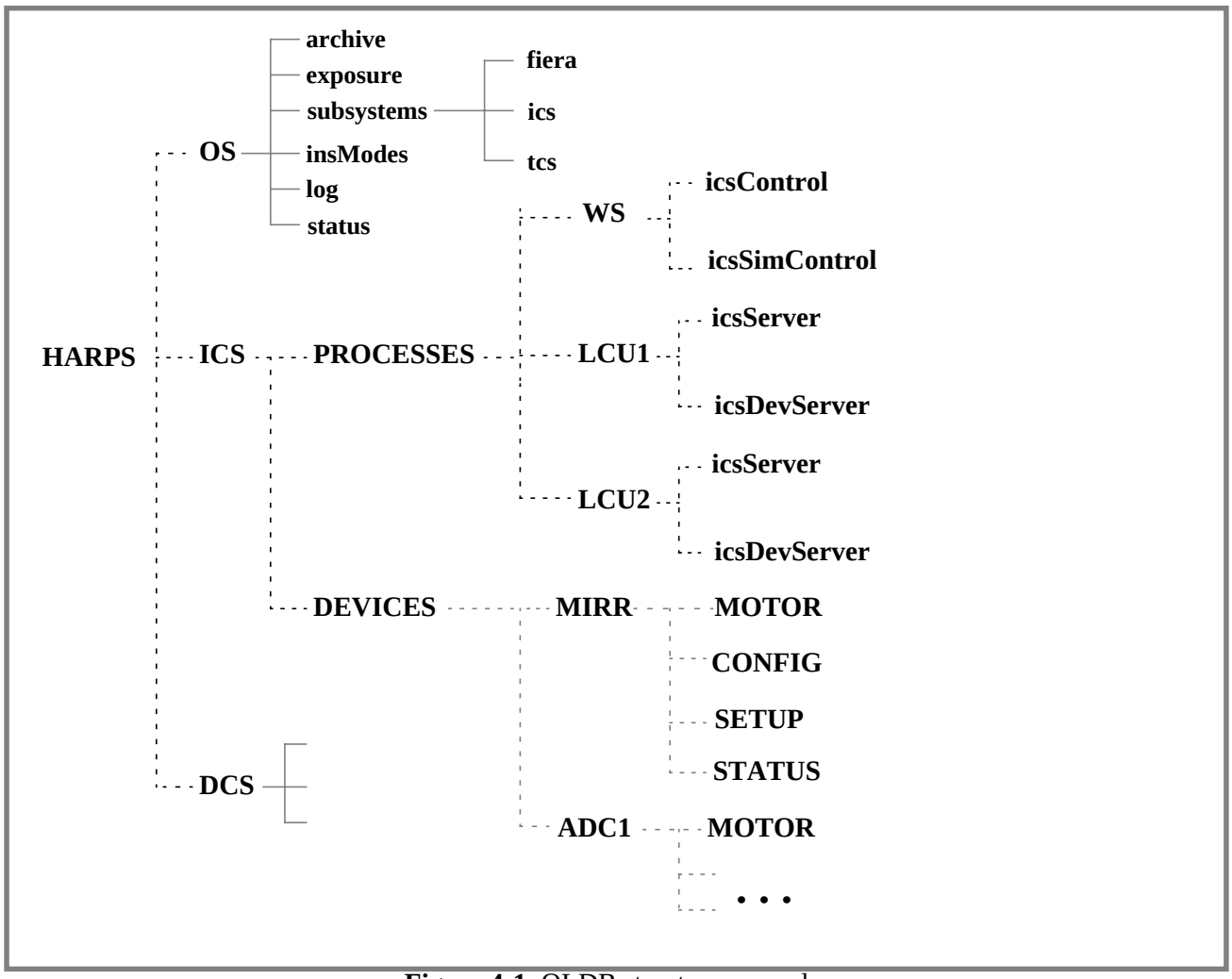


Figure 4-1: OLDB structure example

4.3.1 Database backup

When the control software start, reads the configuration files to update the configuration database values. In case of the motorized functions, the database values are stored in different files (*.dbcfg) stored in hamcfg module. This file is read when the LCU booted, updating the database values. This files are under configuration control and should be modified just by the software developer or maintenance engineer. There is one file for each motorized function

4.4 Instrument configuration file

The up-to-date list of named positions for all HARPS devices, as well as all the OS parameters are defined in the Instrument Configuration File (hamcfgINS.cfg, see 9.4). This file is read directly by the control software during transition from OFF state to LOADED state. This file should be modified just by the software developer or maintenance engineer, it is not intended to by modified neither by the operator nor by the astronomer.

If you need to change the configuration of ICS, you need to change the .cfg file and run icbConfigSet.

The specification on how create configuration files are detailed in [11] for ICS part and [13] for OS part.

4.5 ICS Fits header

When an observation is started (actually before OS sends to DCS the command to start it), OS sends the command EXPSTRT to ICS in order to inform it to perform all actions associated to the beginning of an observation, in particular storage of the current status of the devices (temperatures) and start of statistics calculation, for later saving in a FITS file.
At the end of the observation (after OS has been informed by DCS that the exposure has been finished), OS sends the command EXPEND, followed by ‘STATUS -header -dumpFits’. The ICS FITS header contains all keywords of the ICS Dictionary (see 9.2) which Class is ***header***.
A preliminary FITS header example was shown in [4].

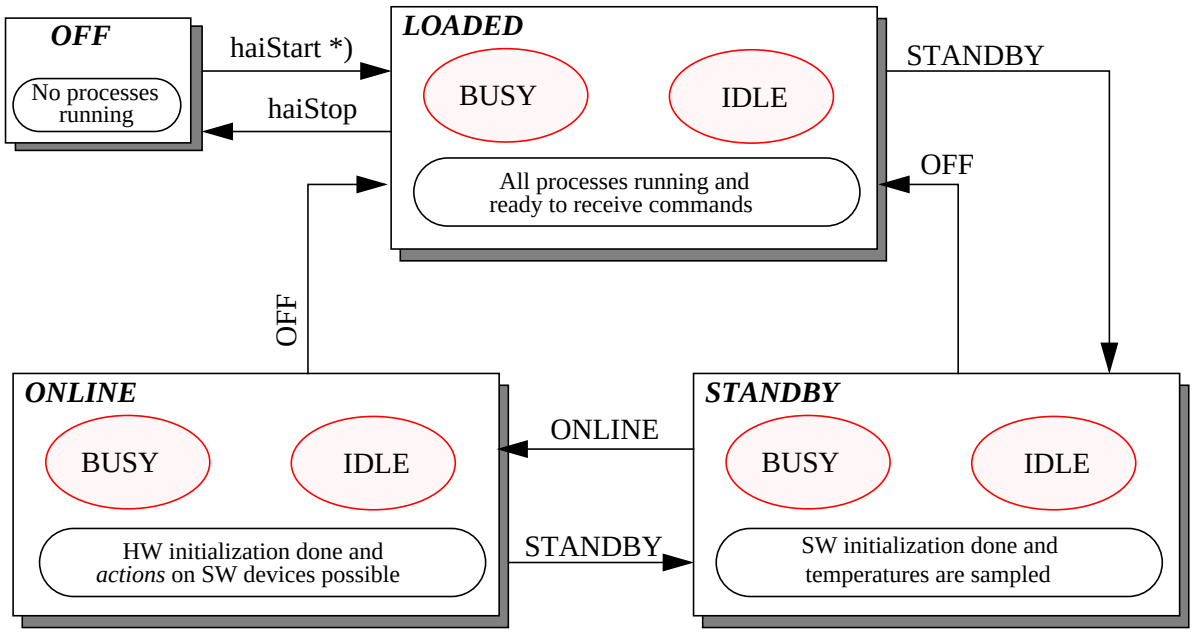
5 FEATURES

5.1 States and Modes

Each HARPS component has its own state, mode and status. The WS Server reports a global operational state based on the states of the individual components as described in the following table.
The operational state can be changed according to Figure 5-1 which shows the ICS states and state transitions:

Table 5-1: Operational states

STATE	Description
OFF	The instrument control software is not running. Instrument functions are powered off After an LCU reboot the system is in this state
LOADED	The instrument control software is running and the database is loaded. No hardware access. Instrument functions are not initialized. Functions maybe under handset control
STANDBY	All instrument devices are in a steady stand-by status, in particular: • Motors are powered off. • Lamps are off • Sensors are monitored
ONLINE	All devices are initialised.



*) provided by module stoo

Figure 5-1: ICS states

Table 5-2: State transition commands

From \ To	OFF	LOADED	STANDBY	ONLINE
OFF	---	via start-up script	---	---
LOADED	shutdown script	---	STANDBY	ONLINE
STANDBY	shutdown script	OFF	---	ONLINE
ONLINE	shutdown script	OFF	STANDBY	---

5.1.1 Substate

Each device has a substate reported in the database. The substate reflects the current activity or the status of a device and can be, for example:

- **IDLE**: ICS is ready to accept and execute a command.
- **BUSY**: A command is being executed. Apart from STOP all other commands will be rejected until the current command was executed.
- **ERROR**: Requires to set the faulty device again to ONLINE (after repair) before a command can be sent to ICS.

5.1.2 Software devices modes

The mode of a software device can be modified updating the instrument configuration file or calling the corresponding command as described below (see [11]):

- **handset**: reports when a device is in handset mode (only applicable to motorized functions).
- **simulation**: Reports the simulation mode of the device. It is set with commands SIMULAT and STOPSIM.
- **enabled**: this mode are not used any more, that means all the functions are enabled and available at any time. If you want to avoid any hardware setup, put the function in simulation mode.

5.2 Commands

5.2.1 List of ICS commands

ICS accepts the standard commands listed in and some additional commands used mainly to configure or calibrate the components. The STATUS command can return not only FITS values, but also additional information that can be useful during maintenance procedures.

The state column lists the device states where a command is accepted by the ICS device manager (L = Loaded, S = Standby, O = On-line).
The Operations/Maintenance (O/M) column indicates if the command is accepted during normal operations (O) or only for maintenance (M).

Table 5-3: Commands accepted by ICS

Name	Arguments	Description	State	O/M
STANDBY	-function	Switch a device to Stand-by state	All	O/M
ONLINE	-function	Switch a device to On-line state	All	O/M
OFF	-function	Switch a device to Loaded state	All	O/M
EXIT	-function	Switch a device to Off state	All	O/M
STATE	-function	Return the state of a device	All	O/M
SIMULAT	-function	Start simulation of a device	All	M
STOPSIM	-function	Stop simulation of a device	All	M
STATUS	-function -header	Return the status of a device	S, O	O/M
STOP	-function	Stop the current movements of a device	All	O/M
SETUP	-function -noMove -check	setup the instrument	O	O/M
DEBUG	-level -log -verbose -timeout	Set the process debug level, logging and verbose mode.	All	M
VERSION	-	Return the process version	All	M
EXPSTRT	-path	Exposure start	O	O/M
EXPEND	-path	Exposure end	O	O/M

The default for many common INS commands is *-function all*. The meaning of this are well explained in [11].

5.2.2 The STATE and STATUS commands

The STATE command returns the global state of the ICS in the LCU. The global state is derived from the states of the individual devices as follows:

- if any of the devices is in state Loaded, then the global state is Loaded;
- if all devices are in state On-line, then the global state is On-line;
- otherwise the global state is Stand-by.

The STATUS command returns the values of the names specified in the `-function` parameter. The accepted names are (the case is ignored):

- All FITS keywords known to ICS (LCU part).
- Function or device names.

If the `-header` option is specified, then the FITS keywords reported in an image header are returned.

5.2.3 OS commands

HARPS OS will inherit all the BOSS commands. These contain all standard OS commands (see [13]). There is no special commands foreseen for HARPS.

5.3 Simulation

The ICS distinguishes between three level of simulation as described in the following table.

Table 5-4: ICS simulation levels

MODE	Description
Software Simulation	The control over the whole hardware is disabled The LCU sw is simulated at Workstation level
Hardware Simulation	The control over some parts of the hardware is disabled The missing parts of the hardware are simulated at LCU level
Normal	The control over the whole hardware is enabled

The normal level used to perform observations is “Normal”. “Hardware Simulation” is used when a particular function needs to be simulated (e.g. set to a fixed position). The level “Software Simulation” can be used when no hardware is available e.g. for software development and tests.
The setting of the simulation mode can be done ONLY before starting up ICS by setting the keyword INS.CON.OPMODE in the file hamcfgSTART.cfg (module hamcfg). It is not possible (and normally not necessary) to change the global simulation level while the system is running.
The simulation level is always indicated in the FITS header files generated by ICS.
The simulation level of each device can be changed on-line by sending the commands SIMULAT and STOPSIM with -function <device_FITS_keyword> argument.

5.4 Errors

5.4.1 ICS Errors

Errors are unfortunately unavoidable so the ICS software needs to define a scheme to handle possible errors. In general errors in ICS are treated via the CCS error system and any error is always logged.
Errors in the ICS LCU are reported to the WS in order to inform the WS Server process. LCU errors are also logged via the same logging system used by the WS.

Errors are defined in the following categories:

- FATAL - no recovery possibly (or unlikely).
- SERIOUS - error recovery likely (but not always).
- WARNING - error recovery always possible.

As ICS is a distributed system which consists of a WS and a LCU part, errors can occur independently and do not necessarily have the same effect. As an example: a fatal error on the ICS LCU which requires a reboot does not neccessarily mean that the ICS WS part has to be restarted.

In the following the different categories are mapped to a list of possible areas where the errors could occur:

Table 5-5: Classification of errors

General System Errors	Fatal	WS: System crash, network failure, Rtap failures. LCU: System locked.
HW Errors	Fatal	LCU: HW Initialization or board does not respond.
Communication Errors	Fatal	WS: Message system failure. LCU: Queue server failure.
Database Errors	Fatal	WS: WS database in-accessible. LCU:Local database failures.
Access Errors	Serious	WS: Missing files, unset environment, invalid path.
System Management Errors	Serious	WS: Disk space full e.g. logging data LCU: Memory leaks.
User Errors	Warning	WS: invalid commands, invalid parameters etc. LCU: direct command sent from other processes.

5.4.2 OS Errors

The various subsystems can report various errors back to HARPS OS, indicating various levels of failure from opto-mechanical or electronic devices. The error classification was detailed in the above section.
HARPS OS will report and log all these subsystem-errors, and in the case of fatal or serious errors, propagate them further up (e.g. to BOB) or in stand-alone mode prompt the operator with an error panel.
On top of the subsystem errors, HARPS OS will have its own set of errors/warning.

5.5 Alarms

The ICS is in charge of reporting alarm and warning signals that are connected to the LCU HW. These digital I/O signals are scanned to the OLDB of the INS WS where the CCS alarm system is used to handle the alarm conditions.

The following severity level of alarms are defined in CCS:

- FATAL
- SERIOUS
- WARNING

Possible alarms are:

- Interlock active
- Iodine cell-fan off
- Motor Drives failure
- Calibration Lamp Failure

Any alarm is logged when it occurs and also when it is acknowledged by the operator.

The actions which need to be performed upon arrival of alarm as well as the decision if the operation can continue after the alarm is acknowledged is to be decided by the Operator.

5.6 Logging

The following set of logs are generated:

Table 5-6: Log types

Error Logs	Error logging is done based on the CCS Error System.
Automatic Logs	Automatic logging provided by the CCS Logging System is used for periodic sampling data.
Event Logging	Logging of abnormal events and alarms.
Operations Logs and Configuration Logs	Any setup and configuration of the instrument.

5.7 Software Installation

The standard installation tool for VLT SW packages shall be used (see [16]).

5.8 Start-up and Shutdown

The standard start-up/shutdown procedure shall be used. The standard start-up tool for VLT Instrument packages shall be used (see [14]).

6 INTERFACES

6.1 HARPS ICS panels

6.1.1 ICS control panel

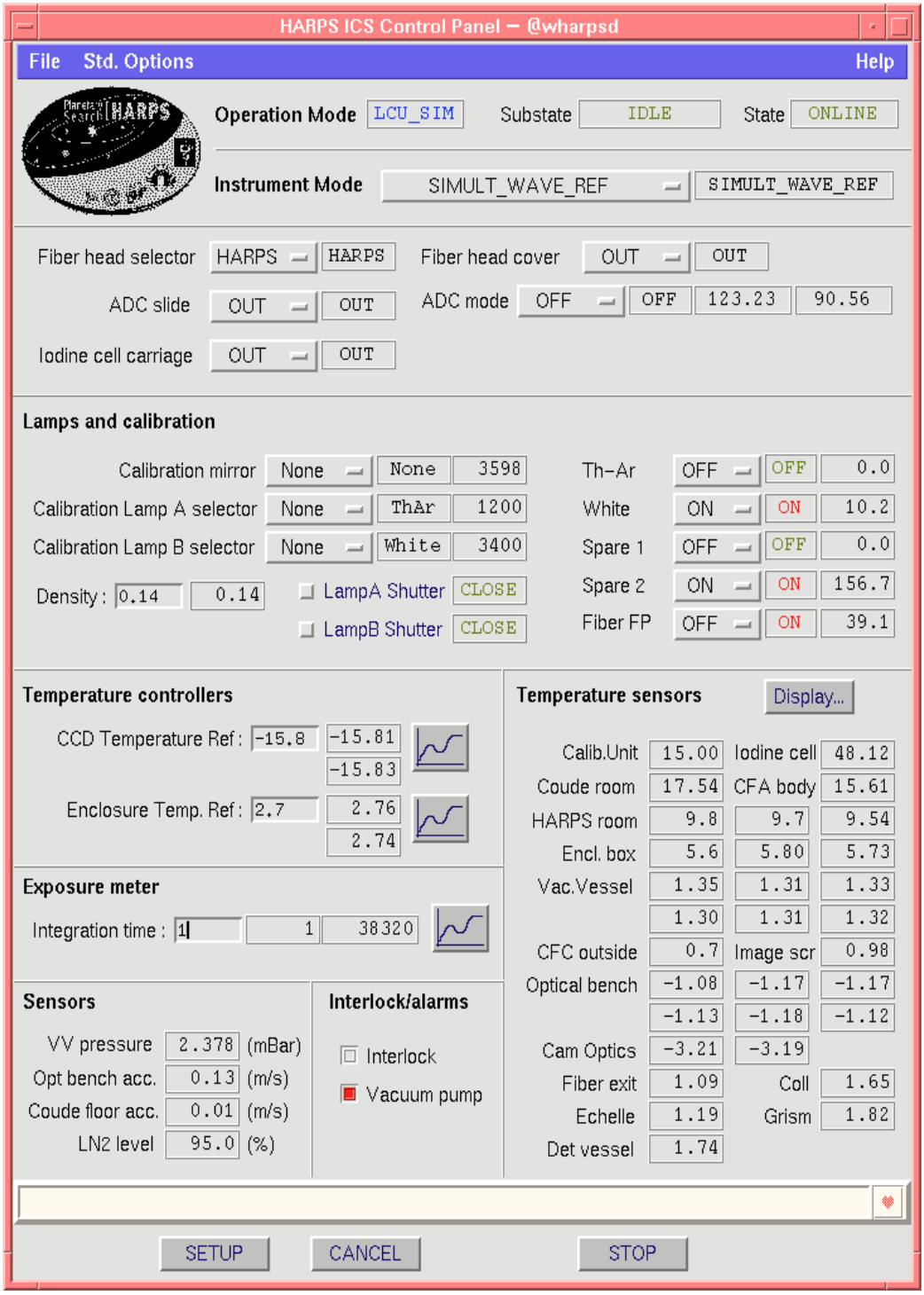


Figure 6-1: HARPS ICS control panel

The control panel allows direct operations with the HARPS ICS without involving any OS interaction and is only used for maintenance and test purposes. Its basic functionality is:

- Setup all instrument functions
- Monitor functions status

6.1.2 Motor control

The Figure 6-2 shows an example of a motor device control panel taken from SOFI instrument software. The same interface will be used to control a single motor as an alternative of the Motor Engineering Interface (see [18]).

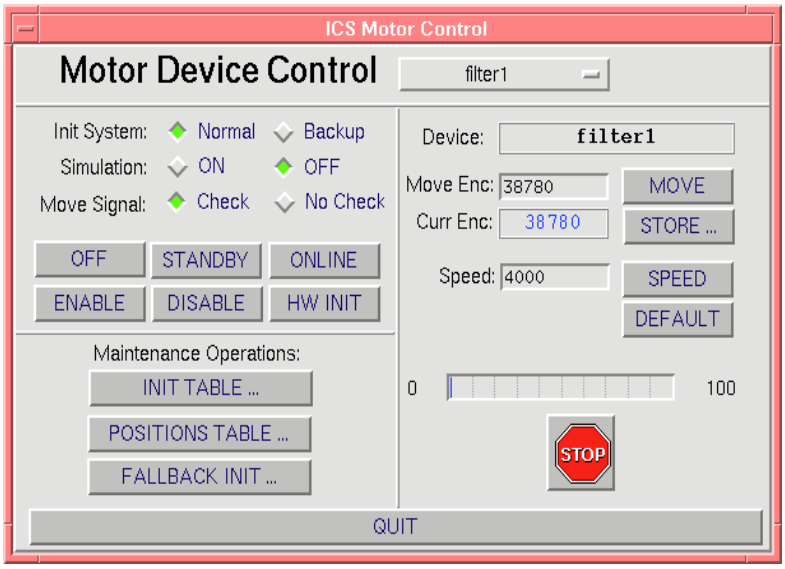


Figure 6-2: ICS motor device control panel example

6.1.3 Exposure meter

The Figure 6-3 shows the exposure meter control panel used in the CES instrument software. This panel will be the base for the HARPS exposure meter panel.

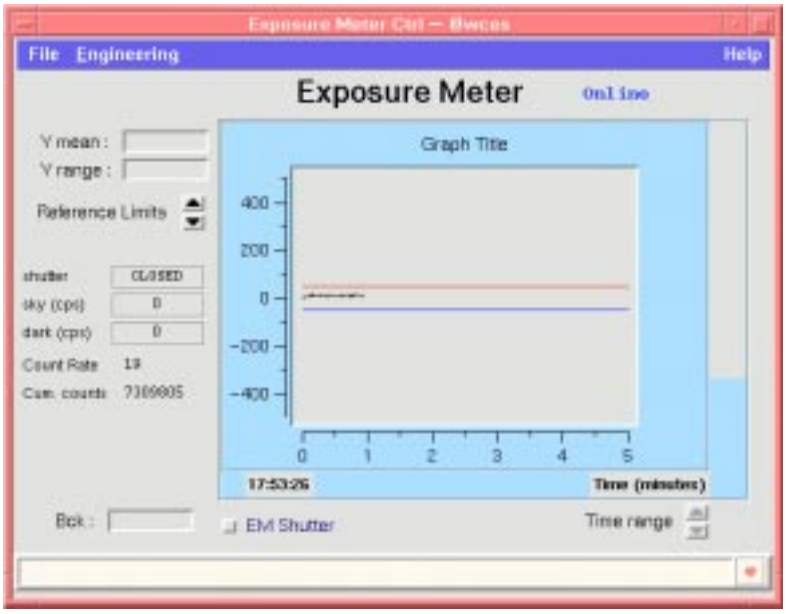


Figure 6-3: Exposure meter control panel example

6.1.4 Sensors plots

For all the sensor values, it is possible to show them in a graphically way using the sampPlot tool provided by VLT Software.

6.2 HARPS OS panels

6.2.1 OS control

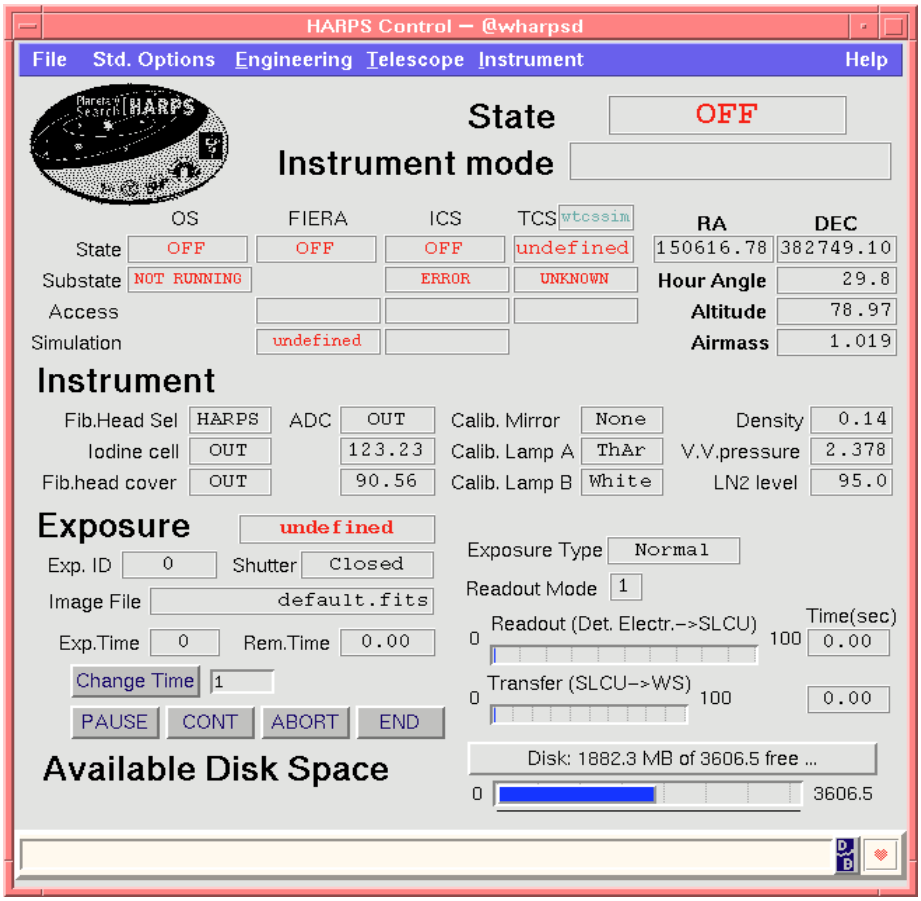


Figure 6-4: HARPS OS control panel

6.2.2 OS engineering

Figure 6-5: HARPS OS engineering panel

7 DEVELOPMENT AND TEST FACTORS

7.1 Planning

In Table 7-1 are listed all the task involved in the development of the OS/ICS system. In the resource column appears the acronym of every resource. Here is the list of them:

OGE Observatory of Geneva

LSO La Silla Observatory

DME Denis Mégevand

FPE Francesco Pepe

UWE Ueli Weilenmann

DGO Domingo Goyak

WEC Wolfgang Eckert

DSO Danuta Sosnowska

JGU Juan Carlos Guzman

SBA Silvia Baeza

Table 7-1: Task list planning

Id	Task description	Start	Finish	Duration	Prerequisite	Responsible
1	Build OS/ICS prototype: - OS/ICS dictionary - INS config file - INS integration module - ICS control panel - OS control panel - Test prototype	03/03/01	06/16/01	60d		JGU
2	OS/ICS prototype installation at OGE	28/06/01	29/06/01	2d	1	JGU, DME, DSO
3	LCU hardware delivery at LSO software laboratory	05/07/01	05/07/01	milestone		DGO
4	Test & update prototype with delivery hardware	05/07/01	11/07/01	8d	3	JGU
5	Additional hardware delivery at LSO software laboratory	20/07/01	20/07/01	milestone		DGO, FPE
6	Implementation of Lakeshore temperature controller special device: - Integration to ICS - Write control code - Test - Update users manual	20/07/01	21/08/01	20d	4	JGU

Id	Task description	Start	Finish	Duration	Prerequisite	Responsible
7	Implementation of PREMA thermometer special device: - Integration to ICS - Write control code - Test - Update users manual	21/08/01	20/09/01	20d	4	JGU
8	Test & update INS package including special devices	21/09/01	24/09/01	4d	6,7	JGU
9	HCFA + sensors hardware ready for software test	30/09/01	30/09/01	milestone	5	UWE, WEC, DGO, AGI
10	Test of OS/ICS in normal mode and DCS in simulation mode	01/10/01	31/10/01	20d	9	JGU, SBA
11	Write OS/ICS Users Manual	05/07/01	31/10/01	80d	2	JGU
12	Write OS/ICS Acceptance Test plan document	21/09/01	31/10/01	25d	8	JGU
13	Write OS/ICS Maintenance Manual	01/11/01	01/12/01	20d	12	JGU
14	Delivery HARPS to OGE	27/01/02	01/01/02	5d	10	LSO
15	INS Software integration at OGE	14/01/02	15/02/02	20d	14	DME, JGU, SBA
16	INS Software PAE	18/02/02	05/07/02	80d	15	OGE
17	Delivery HARPS to LSO	06/07/02	06/08/02	20d	16	OGE, LSO
18	INS Software Preliminary Acceptance Test LSO	06/08/02	27/08/02	15d	17	OGE, LSO
19	First commissioning at 3.60m telescope	28/08/02	13/09/02	10d	18	OGE, LSO
20	Second commissioning at 3.60m telescope	4/11/02	20/11/02	10d	19	OGE,LSO
21	Delivery HARPS to normal operations	21/11/02	21/11/02	milestone	20	OGE, LSO

7.2 Development requirements

According with Table 7-1, there are 3 hardware critical milestone.
The component list corresponding to LCU hardware delivery at LSO software laboratory (task 3) are:

- HP B2000 Instrument workstation with HP-UX 11.0 plus VLTSW MAR2001 CCSLite release.

- 2 LCUs (Cassegrain Fiber Adapter and Spectrograph) with all the boards listed in section 2.2.

The hardware needed for task 5 (Additional hardware delivery at LSO software laboratory) are:

- 1 motor function (mechanical part + motor + tacho + encoder + connections to LCU)
- 1 ADAM controller
- 1 Lakeshore 331S temperature controller
- 1 PREMA thermometer
- 1 OMEGA CN76000 temperature controller
- Exposure meter
- Connections to serial board

The hardware needed for task 9 (HCFA + sensors hardware ready for software test) are:

- Cassegrain Fiber Adapter complete unit with connections of motor, tacho, encoders and signals to LCU.
- 1 sensor for each thermometer and temperature controller.

7.3 Test strategy

The test strategy will be describe in a future document “HARPS OS/ICS Acceptance Test Plan”.

8 FUNCTION TRACEABILITY MATRIX

Detailed Design Document		User Requirements and Design Report	
solution description	sections	requirement reference	sections
Base ICS commands and ICS self-test procedure	5.2.1 4.6 of [11]	[ICS_SR1] General operation requirements	2.3.2.1
SETUP and STATUS standard base ICS commands	5.2.1	[ICS_SR2] Instrument setup and status information	2.3.2.2 4.1.2.1
EXPSTRT & EXPEND standard base ICS commands	3.2.4.2 5.2.1	[ICS_SR3] Actions at exposure start and end	2.3.2.3 4.1.2.1
ICS fits header data	4.5	[ICS_SR4] ICS FITS header keywords	2.3.2.4
HARPS motorized functions are standard base ICS devices	3.3.1 8.12 of [11]	[ICS_SR5] Control of motorized functions	2.3.2.5 4.1.3
ADC is a standard base ICS device	3.3.1 8.12.1 of [11]	[ICS_SR6] Special control of ADC	2.3.2.6 4.1.2.1 4.1.3
Exposure meter is a standard base ICS	3.3.1	[ICS_SR7] Exposure meter software device control	2.3.2.7 4.1.3.1
Sensors controlled by standard base ICS and special devices modules	3.3	[ICS_SR8] Configuration and monitoring of all sensor’s values	2.3.2.8 4.1.3.3
hailtc special device module	3.3.3	[ICS_SR9] Control of external temperature controller	2.3.2.9 4.1.3.2
Calibration lamps and shutters are standard base ICS devices	3.3.1 8.11 of [11]	[ICS_SR10] & [ICS_SR11] Control of calibration lamps & shutters	2.3.2.10 2.3.2.11 4.1.3
ICS logging system	5.6	[ICS_SR12] Logging	2.3.2.12 5.5
ICS error and alarms system	5.4 and 5.5	[ICS_SR13] Errors and alarms	2.3.2.13
Standard base ICS SETUP command implementation	5.2.1	[ICS_SR14] Safety	2.3.2.14 4.1.2.1
ICS control panel	6.1.1	[ICS_SR15] UIF capable to set-up all devices and show the result after completion of the set-up	2.3.5.1

Detailed Design Document		User Requirements and Design Report	
solution description	sections	requirement reference	sections
ICS control panel	6.1.1	[ICS_SR16] UIF showing all sensor's values	2.3.5.1
Exposure meter panel	6.1.3	[ICS_SR17] UIF showing the status of exposure meter	2.3.5.1
Standard base OS commands	5.2.3 9.4 of [13]	[OS_SR1] Sequence of actions of a single exposure	2.4.1.1 4.2.1
Setup guide of base OS module	3.4.2 8 of [13]	[OS_SR2] & [OS_SR3] Set-up requirements	2.4.1.2
ICS will handle this feature (e.g. ADC function)	8.12.1 of [11]	[OS_SR4] Function movements	2.4.1.4
VLT CCS scan OLDB system		[OS_SR5] Update requests	2.4.1.5
Exposure handling by base OS	3.4.2 9.6 of [13]	[OS_SR6] Exposure handling	2.4.1.6
This is done by ICS configuration file	4.4 8.9 of [11]	[OS_SR7] Simulation mode	2.4.1.7
HARPS OS control panel	6.2.1	[OS_SR8] Status monitoring	2.4.1.8
HARPS OS control panel	6.2.1	[OS_SR9] UIF showing status and relevant devices	2.4.4.1
This is done by BOB and templates. Some of buttons are implemented in HARPS OS control panel	6.2.1	[OS_SR10] UIF capable of make instrument, detector and telescope set-up.	2.4.4.2

9 REFERENCE

9.1 Command Definition Tables (CDTs)

The standard CDT's provided by icb are used.

9.2 HARPS ICS Dictionary file (dicHARPS_ICS.txt)

```
#*******************************************************************************
# E.S.O. - VLT project
#
# "@(#) $Id: dicHARPS_ICS.txt,v 1.3 2001/05/10 02:20:09 vltsccm Exp $"
#
# HARPS_ICS dictionary
#
# who       when      what
# --------  --------  ----------------------------------------------
# jguzman 2001-05-10 neutral density wheel FITS keywords modified
# jguzman 2001-05-09 Lakeshore special device specific keywords added
# jguzman 2001-05-08 move exposure meter as standard device
# jguzman 2001-03-05 Adapted to HARPS based on dicXXXX v.1.15
# alongino  15/01/01  VLTSW20010016: compliance with DICB
# rschmutz  10/12/00  INS.ADCi.* keywords added.
# alongino  06/06/00  added INS.SENSORi ID and NAME. Same for TEMPi
# alongino  04/02/00  INS.ROTi and INS.GRISi keywords added.
# rschmutz  01/02/00  INS.FOCUi keywords added.
# alongino  05/01/00  added keywords to address devices (for cmds like SIM)
# rschmutz  17/12/99  specific HARPS sensor device SWSIM,INIT keywords added.
# rschmutz  21/08/99  special device YYYY keywords added.
# rsc/tph   19/08/99  DROT -> DROTi.
# rschmutz  08/08/99  created
#      (copied icbDictionary.txt and removed the
#       unneeded keywords; CCC1 and DIS1 keywords added.).
#

#*******************************************************************************
#
# This dictionary contains all keywords used by HARPS ICS at runtime
# to execute and understand commands with -function option, such
# as SETUP and STATUS.
#
#*******************************************************************************

#
# Notes
# 1) The letters within () indicate the Class that keyword belongs to:
#    c = config
#    h = header
#    l = conf-log
#    p = private
#    s = setup
```

```
#    o = ops-log
# 2) The value format for keywords of Type logical is set to 'T'.
#    It means that the actual value is of type %c, with possible
#    values 'T' or 'F'

DICTIONARY HARPS_ICS "HARPS" "HARPS Instrument ICS keywords"
REVISION"$Revision: 1.3 $"
CONTEXT"Instrument"

#
# 2.2 ICS setup and header
#
INS.ID%30sInstrument ID (chl).
Combines the ESO identification of the instrument
and the software version of the control software.
Format: NAME/HW-REV/SW-VERSION
INS.DID%30sData dictionary for INS (ch).
Name and version of ESO DID to which INS keywords
comply to.
INS.OPER%30sInstrument Operator (hs).
Initials and family name of instrument operator or
service observer.
INS.SWSIM%15sSoftware simulation (chs).
Possible values:
NORMAL:  no simulation.
LCU_SIM: LCU is simulated by WS simulator.
HW_SIM:  Hardware is simulated on LCU.
INS.MODE%10sInstrument mode used (hs).
Acronym of instrument mode used. This acronym identifies
the optical path within the instrument. Examples:
BLUE, RED, DIC1, DIC2
INS.PATH%10sOptical path used (h).
The optical path within the instrument used for
this exposure.

#
# The following configuration parameters describe the individual instrument
# devices.  Each device type has its own set of configuration parameters.
#
# Defaults:
#
# - The <subsystem>.TYPE keyword specifies the implementation to be used
#   for the device.  The default corresponds to the implementation supplied
#   by Base ICS.
#
# - If the <subsystem>.PREFIX keyword is not supplied, then the 'FITS prefix'
#   needed by the ic0 devices corresponds to the <subsystem>.
#

#
# 3. Device configuration
#
# 3.1 Calibration mirror
```

```
# 3.2 ADC
# 3.3 Optical functions:
#     - Fiber heads selector
#     - Dust cover
#     - Iodine cell carriage
#     - Calibration lamp A selector
#     - Calibration lamp B selector
# 3.4 Neutral density wheel
# 3.5 Calibration lamps
# 3.6 Calibration lamps shutters
# 3.7 Analog sensors
# 3.8 Temperature sensors
#
# 3.9 Special function
#
# 3.9.1 Exposure meter
# 3.9.2 Temperature controllers
#

#
# 3.1 Calibration mirror
#
#   Config. FITS keywords:
#DEVNAME, DEVDESC, LCUID, AVAIL, ENABLED, SWSIM,
#POSNUM, POSIDi, IDi, NAMEi, FOCUSi.
#   Setup and header FITS keywords:
#NO, ID.
#   Maint. keywords (reserved):
#ENC, ENCREL, LIMIT, TURN, SPEED.
#   Conf. log FITS keywords:
#INIT, MOVE, STOP.
#

# To address the device in commands such as SIM, ONLINE etc.
INS.MIRRi%10sMirror device

# 3.1.1 Mirror config. FITS keywords
#
INS.MIRRi.SWSIMTIf T, function is software simulated (cho).

INS.MIRRi.IDi%10sMirror unique ID (chls).
Name and hardware identification of mirror.
Format: NAME/SER-NO
INS.MIRRi.NAMEi%15sMirror name (chls).
Name for the mirror.
INS.MIRRi.TYPEi%19sElement type (cl).
Kind of optical element described.

# 3.1.2 Mirror setup and header FITS keywords
#
INS.MIRRi.NO%dMirror slide position (hs).
Index of the position of the mirror slide
INS.MIRRi.ENC%dMirror slide absolute position [Enc] (s).
```

```
INS.MIRRi.ENCREL%dMirror slide relative position [Enc] (s).
INS.MIRRi.LIMIT%10sMove to the limit (s).
Move to the lower or upper limit.
INS.MIRRi.TURN%10sTurn right/left (s).
Turn the wheel to the right or to the left.
INS.MIRRi.SPEED%.1fMoving speed (s).
Speed (default units) the motor is moved (see LIMIT).

# 3.1.3 Mirror operational logs.
#
INS.MIRRi.INIT%sHardware initialisation (o).
INS.MIRRi.MOVE%sMotion execution (o).
INS.MIRRi.STOP%sMotion stop (o).

#
# 3.2 ADC
#
#   Config. FITS keywords:
#DEVNAME, DEVDESC, LCUID, AVAIL, ENABLED, SWSIM,
#PERIOD, REFENC, MOTSIGN, FORMULA, MINELEV,
#POSOFFST, COFFSET, PSLOPE, POFFSET, TSLOPE,
#TOFFSET, AFACTOR, DROTFACT.
#   Setup and header FITS keywords:
#MODE, RA, DEC, TEMP, SENS1.
#   Maint. keywords (reserved):
#POSANG, ENC, ENCREL, TURN, SPEED.
#   Conf. log FITS keywords:
#INIT, MOVE, STOP.
#

# 3.2.1 ADC config. FITS keywords
#
INS.ADCi.DEVNAME%19sName of the ICS device (C).
INS.ADCi.DEVDESC%19s    Description of the ICS device (c).
INS.ADCi.LCUID%dId. of the LCU managing the device (c).
INS.ADCi.AVAIL%cDevice availability flag (c).
INS.ADCi.ENABLED%cFlag for enabling and disabling (c).
INS.ADCi.SWSIM%cIf T, function is software simulated (cho).

INS.ADCi.PERIOD%dUpdate frequency [msec] (c).
INS.ADCi.REFENC%dMotor reference position [Enc] (c).
INS.ADCi.MOTSIGN        %dMotor direction sign (c).
INS.ADCi.FORMULA%dADC formula selector (c).
Two ADC formulas are available (1 and 2).
See the user manual for details.
INS.ADCi.MINELEV%.2fMin. allowed elevation [deg] (c).
INS.ADCi.POSOFFST%.2fPosition offset [deg] (c).
INS.ADCi.COFFSET%.4fC offset [arcsec] (c).
INS.ADCi.PSLOPE%.4fPressure slope [arcsec/mbar] (c).
INS.ADCi.POFFSET%.0fPressure offset [mbar] (c).
INS.ADCi.TSLOPE%.4fTemperature slope [arcsec/C] (c).
INS.ADCi.TOFFSET%.1fTemperature offset [C] (c).
INS.ADCi.AFACTOR%.3fA factor [1/arcsec] (c).
```

INS.ADCi.DROTFACT%.0fDerotator factor (c).

3.2.2 ADC setup and header FITS keywords
#
INS.ADCi.MODE%4sADC mode (hs).
Possible values: AUTO, OFF.
INS.ADCi.RA%.6fTelescope right ascension [deg] (hs).
INS.ADCi.DEC%.5fTelescope desclination [deg] (hs).
INS.ADCi.POSANG%.4fPosition angle [deg] (hs).
INS.ADCi.TEMP%.1fTemperature [C] (hs).
INS.ADCi.SENS1%.1fPressure [mbar] (hs).
INS.ADCi.ENC%dADC absolute position [Enc] (s).
INS.ADCi.ENCREL%dADC relative position [Enc] (s).
INS.ADCi.TURN%10sTurn right/left (s).
Turn the wheel to the right or to the left.
INS.ADCi.SPEED%.1fTurning speed (s).
Speed (default units) the wheel is turned (see TURN).

3.2.3 ADC operational logs.
#
INS.ADCi.INIT%sHardware initialisation (o).
INS.ADCi.MOVE%sMotion execution (o).
INS.ADCi.STOP%sMotion stop (o).

3.2.4 ADC Slide
INS.ADCS%5sADC slide device (p).
Used in connection with ENABLE/DISABLE commands
INS.ADCS.ID%30sADC slide identification (l).
Hardware identification of the ADC slide.
INS.ADCS.NAME%10sADC slide position (hs).
Position of the ADC slide. Possible values:
IN, OUT
INS.ADCS.NO%dADC slide position (hs).
Index of the position of the ADC slide
INS.ADCS.ENC%dADC slide absolute encoder position (ps).
INS.ADCS.ENCREL%dADC slide relative encoder position (ps).
INS.ADCS.INIT%sHardware initialisation (o).
INS.ADCS.MOVE%sMotion execution (o).
INS.ADCS.STOP%sMotion stop (o).
INS.ADCS.UNFORSEEN%sUnforeseen event (o).
INS.ADCS.SWSIM%cIf T, function is software simulated (ho).
Function software simulation.
INS.ADCS.LIMIT%10sMove to the limit (ps).
Move to the lower or upper limit.
INS.ADCS.SPEED%.1fMoving speed (ps).
Speed (default units) the motor is moved (see LIMIT).
INS.ADCS.SAMPLE%.3fMotor status sampling rate [Hz] (ps).
Rate at which MCM must monitor the motor status
A value 0 means “take the default value from MCM OLDB”
INS.ADCS.CNCT%cConnect/Disconnect motor power (ps).
Connect (T) or disconnect (F) the motor from the
amplifier
INS.ADCS.TYPE%10sType of component (l).

```
#
# 3.3 Optical functions
#
#   Config. FITS keywords:
#DEVNAME, DEVDESC, LCUID, AVAIL, ENABLED, SWSIM,
#CHGABLE, POSNUM, POSIDi, IDi, NAMEi, TYPEi, FOCUSi, ALIGNi.
#   Setup and header FITS keywords:
#NO, ID, NAME, TYPE.
#   Maint. keywords (reserved):
#ENC, ENCREL, LIMIT, TURN, SPEED.
#   Conf. log FITS keywords:
#INIT, MOVE, STOP, CHANGE.
#

# To address the device in commands such as SIM, ONLINE etc.
INS.OPTIi%10sGeneric Optical device

# 3.3.1 Optical function config. FITS keywords
#
INS.OPTIi.SWSIMTIf T, function is software simulated (cho).

# Each element position registers only the IDi of the mounted element.
# A corresponding ID entry is searched to retrieve the rest of the
# element data.
#
# A list of elements that can be mounted in this function is described
# with entries IDi, NAMEi, etc. (one keyword set describes one element,
# see section 3.6 of the INS Common Software Spec. manual,
# VLT-SPE-ESO-17240-0385).


INS.OPTIi.IDi%10sOPTIi unique ID (chls).
Identification of element.
INS.OPTIi.NAMEi%15sOPTIi name (chl).
Name of element.
INS.OPTIi.TYPEi%10sOPTIi element (hl).
Kind of optical element described.
INS.OPTIi.ALIGNi%dOPTIi element alignment [Enc] (cs).
Alignment value of element.

# 3.3.2 Optical function setup and header FITS keywords
#
INS.OPTIi.NO%dOPTIi slot number (hls).
When OPTIi can mount more than one element, like
filters on a wheel, this keyword gives the number
of the slot positioned in the optical axis.
INS.OPTIi.ENC%dComponent absolute position [Enc] (s).
INS.OPTIi.ENCREL%dComponent relative position [Enc] (s).
INS.OPTIi.LIMIT%10sMove to the limit (s).
Move to the lower or upper limit.
INS.OPTIi.TURN%10sTurn right/left (s).
Turn the wheel to the right or to the left.
```

INS.OPTIi.SPEED%.1fMoving speed (s).
Speed (default units) the motor is moved (see LIMIT).

3.3.3 Optical function operational logs.
#
INS.OPTIi.INIT%sHardware initialisation (o).
INS.OPTIi.MOVE%sMotion execution (o).
INS.OPTIi.STOP%sMotion stop (o).
INS.OPTIi.CHANGE%sElement changed (o).
Element has been physically replaced.
Maintenance operation.

#
3.4 Neutral density wheel (filter)
#
Config. FITS keywords:
#DEVNAME, DEVDESC, LCUID, AVAIL, ENABLED, SWSIM,
#FRML, POSMIN, POSMAX, TOLERANC, OFFSET, RAMP.
Setup and header FITS keywords:
#POS.
Maint. keywords (reserved):
#ENC, ENCREL, LIMIT, TURN, SPEED.
Conf. log FITS keywords:
#INIT, MOVE, STOP.
#

To address the device in commands such as SIM, ONLINE etc.
INS.ROTi%10sGeneric rotating device

3.4.1 Position function config. FITS keywords
#
INS.ROTi.SWSIMTIf T, function is software simulated (cho).
INS.ROTi.TYPE%10sROTi element (hl).
Kind of optical element described.

3.4.2 Position function setup and header FITS keywords
#
INS.ROTi.POS%.1fPosition [deg] (hs).

INS.ROTi.ENC%dAbsolute position [Enc] (hs).
INS.ROTi.ENCREL %dRelative position [Enc] (s).
INS.ROTi.TURN%10sTurn right/left (s).
Turn the wheel to the right or to the left.
INS.ROTi.SPEED%.1fMoving speed (s).
Speed (default units) the motor is moved (see TURN).

3.4.3 Position function operational logs.
#
INS.ROTi.INIT%sHardware initialisation (o).
INS.ROTi.MOVE%sMotion execution (o).
INS.ROTi.STOP%sMotion stop (o).

```
#
# 3.5 Lamp
#
#   Assembly of lamp, shutter and wheel:
#SHUTTER, POSDEV, POSID.
#   Config. FITS keywords:
#DEVNAME, DEVDESC, LCUID, AVAIL, ENABLED, SWSIM,
#ID, NAME, WARMUP, CHGABLE, TIMEOUT,
#SIGUSED, SIGDEV, SIGBITi, SIGLOWi.
#   Setup and header FITS keywords:
#ID, NAME, ST, WAIT, SWSIM.
#   Operational log FITS keywords:
#INIT, START, TIME, STOP, CHANGE.
#   Maint. keywords (reserved):
#STAT, ONTIME, STBYTIME, READY.
#

# To address the device in commands such as SIM, ONLINE etc.
INS.LAMPi%10sLamp device

# 3.5.0 Assembly of lamp, shutter and wheel.
#
# A shutter may be placed in front of a lamp.
# A lamp may be mounted on a wheel to select several lamps.
#
# In this cases, when the keyword INS.LAMPi.NAME is used in a SETUP command,
# then the associated wheel is moved to the corresponding position and the
# associated shutter is opened.
# Other lamps are turned off and associated shutters are closed, unless
# more INS.LAMPi.NAME keywords are specified in the _same_ SETUP command.
#
# To only turn on/off a lamp use the keyword INS.LAMP.ST.
#

# 3.5.1 Lamp config. FITS keywords
#
INS.LAMPi.SWSIMTIf T, function is software simulated (cho).
INS.LAMPi.ID%10sLamp ID (chl).
Hardware identification of lamp bulb unit.
                        This ID is unique for the observatory.
INS.LAMPi.NAME%15sLamp name (chls).
Name for lamp unit.
INS.LAMPi.WAIT%dMax. time [sec] to wait for lamp warm-up (cs).
ICS must check if the lamp is warm. If not, it must
wait maximum n sec before replying. The reply is
FAILURE if after this time the lamp is NOT warm yet.

# Digital signals:

# 3.5.2 Lamp setup and header FITS keywords
#
INS.LAMPi.STTLamp activated (hs).
```

```
# 3.5.3 Lamp operational logs.
#
INS.LAMPi.INIT%sLamp hardware initialized (o).
INS.LAMPi.START%sLamp turned on (o).
INS.LAMPi.TIME%.1fTime [sec] lamp was on (o).
INS.LAMPi.STOP%sLamp turned off (o).
INS.LAMPi.CHANGE%sLamp changed (o).
Lamp has been physically replaced.
Maintenance operation.

#
# 3.6 Shutter
#
#   Config. FITS keywords:
#DEVNAME, DEVDESC, LCUID, AVAIL, ENABLED, SWSIM,
#ID, NAME, TIMEOUT
#SIGUSED, SIGDEV, SIGBITi, SIGLOWi.
#   Setup and header FITS keywords:
#ID, NAME, ST, SWSIM.
#   Operational log FITS keywords:
#INIT, OPEN, TIME, CLOSE.
#   Maint. keywords (reserved):
#STAT, OPENTIME.
#

# To address the device in commands such as SIM, ONLINE etc.
INS.SHUTi%10sShutter device

# 3.6.1 Shutter config. FITS keywords
#
INS.SHUTi.SWSIMTIf T, function is software simulated (cho).
INS.SHUTi.ID%10sShutter ID (chl).
Hardware identification of shutter unit.
                    This ID is unique for the observatory.
INS.SHUTi.NAME%15sShutter name (chl).
Name for shutter unit.

# Digital signals:

# 3.6.2 Shutter setup and header FITS keywords
#
INS.SHUTi.STTShutter open (hs).

# 3.6.3 Shutter operational logs.
#
INS.SHUTi.INIT%sShutter initialized (o).
INS.SHUTi.OPEN%sShutter opened (o).
INS.SHUTi.TIME%.1fTime [sec] shutter was open (o).
INS.SHUTi.CLOSE%sShutter closed (o).

#
# 3.7 Exposure meter
```

```
#
INS.DETi%5sDetector device (p).
Used in connection with ENABLE/DISABLE commands
INS.DETi.TYPE%10sType of detector (hl).
Possible value: PMT
INS.DETi.NAME%10sdetector name (hl).
ESO name of detector.
INS.DETi.FILE%sdetector output file name (hs).
INS.DETi.UIT%.3fUser defined Integration time [sec] (hs).
INS.DETi.CTMIN%.0fMinimum count during exposure (h).
INS.DETi.CTMAX%.0fMaximum count during exposure (h).
INS.DETi.CTLAST%.0fLast counts read (p).
INS.DETi.CTTOT%.0fTotal counts during exposure (h).
INS.DETi.CTMEAN%.1fAverage counts during exposure (h).
Mean value of counts during exposure
INS.DETi.TMMEAN%.2fNormalised mean exposure time (h).
Normalised mean exposure time
Formula: sum(i * count(i))/(ncounts * sum(count(i))
Result in the range (0,1)
INS.DETi.CTRMS%.2fRMS of counts during exposure (h).
Root Mean Square value of counts during exposure
INS.DETi.OFFDRK%.0fAverage dark background counts (hs).
INS.DETi.OFFSKY%.0fAverage sky background counts (hs).
INS.DETi.SWSIM%cIf T, function is software simulated (ho).
Function software simulation.
INS.DETi.INIT%sHardware initialisation (o).


#
# 3.8 Sensors
#
# Each sensor device INS.SENSORi manages a set of related sensors values
# INS.SENSi.  The index SENSi of each sensor value must be unique within
# the instrument (to be able to report each sensor value with a different
# FITS keyword).
#
# There is no relation between sensor device indexes (SENSORi) and sensor
# value indexes (SENSi), e.g. one sensor device INS.SENSOR3 can manage
# e.g. the sensor values INS.SENS12 to INS.SENS16.
#
# The sensor values managed by one sensor device are usually connected
# to a single harware port (PORT).  The hardware port of analog sensors
# is normally a serial port, the hardware ‘port’ of a digital sensor
# device is an ACRO board.
#

#
# 3.8.1 Analog sensors
#
#   Config. FITS keywords:
#DEVNAME, DEVDESC, DEVTYPE, LCUID, AVAIL, ENABLED, SWSIM,
#PORT, NUM, NAMEi, DESCi, HEADERi, FITSi,
#SENADDRi, SENTYPEi, SENREFi, SENRAMPi, SENUNITi.
```

```
#   Setup and header FITS keywords:
#INS.SENSi.ST, INS.SENSi.STAT, INS.SENSi.VAL, INS.SENSi.REF.
#   Maint. keywords (reserved):
#-
#   Conf. log FITS keywords:
#INIT.
#

#
# 3.8.2 Digital sensors (DEVTYPE=DIGITAL)
#
#   Config. FITS keywords:
#DEVNAME, DEVDESC, DEVTYPE, LCUID, AVAIL, ENABLED, SWSIM,
#PORT, NUM, NAMEi, DESCi, HEADERi, FITSi,
#SIGBITi, SIGWIDTHi, SIGLOWi, SIGVALi, SIGSIMi.
#   Setup and header FITS keywords:
#INS.SENSi.ST, INS.SENSi.STAT, INS.SENSi.VAL.
#   Maint. keywords (reserved):
#-
#   Conf. log FITS keywords:
#INIT.
#

# To address the device in commands such as SIM, ONLINE etc.
INS.SENSORi%10sSensor device

# 3.8.[12].1 Sensors common config. FITS keywords
#
INS.SENSORi.SWSIMTIf T, function is software simulated (cho).
INS.SENSORi.ID          %10s    Sensor device unique id (co).
                        ESO identification number for the
                        sensor device.
INS.SENSORi.NAME        %10s    Sensor device common name (co).
                        Name for the sensor device.

# 3.8.1.1 Analog sensors config. FITS keywords
#

# 3.8.2.1 Digital sensors config. FITS keywords
#

# 3.8.[12].2 Sensors header FITS keywords
#
INS.SENSi.ID            %10s    sensor ID (ho).
INS.SENSi.NAME          %30s    sensor common name (ho).
INS.SENSi.STTSensor boolean value (ho).
INS.SENSi.STAT%sSensor string value (ho).
INS.SENSi.VAL%fSensor numeric value (ho).
INS.SENSi.REF%fSensor reference value (ho).
INS.TEMPi.ID            %10s    Temperature sensor ID (ho).
INS.TEMPi.NAME          %30s    Temperature sensor name (ho).
INS.TEMPi.VAL%fTemperature Sensor numeric value (ho).
INS.TEMPi.REF%fTemperature Sensor reference value (ho).
```

```
# 3.8.[12].3 Sensors statistical values
#           Non-standard Base ICS keywords
INS.SENSi.START%.1fsensor value at start (ho).
sensor value of the instrument measured at the start
of the exposure.
INS.SENSi.SAMP%.1fSensirs sampling time [sec] (hos).
Sampling time for temperature.
Special value: -1 = no sampling
INS.SENSi.STOP%.1fsensor value at end (ho).
sensor value of the instrument measured at the end
of the exposure.
INS.SENSi.MIN%.1fMinimum sensor value (ho).
Minimum sensor value during the exposure.
INS.SENSi.MAX%.1fMaximum sensor value (ho).
Maximum sensor value during the exposure.
INS.SENSi.MEAN%.1fAverage sensor vlaue (ho).
Mean Sensor value during the exposure.
INS.SENSi.SIGMA%.1fFirst derivative of sensor values (ho).
INS.TEMPi.START%.1fTemperature at start [C] (ho).
Temperature of the instrument measured at the start
of the exposure.
INS.TEMPi.SAMP%.1fTemperature sampling time [sec] (hos).
Sampling time for temperature.
Special value: -1 = no temperature sampling
INS.TEMPi.STOP%.1fTemperature at end [C] (ho).
Temperature of the instrument measured at the end
of the exposure.
INS.TEMPi.MIN%.1fMinimum temperature [C] (ho).
Minimum Temperature during the exposure.
INS.TEMPi.MAX%.1fMaximum temperature [C] (ho).
Maximum Temperature during the exposure.
INS.TEMPi.MEAN%.1fAverage temperature [C] (ho).
Mean Temperature during the exposure.
INS.TEMPi.SIGMA%.1fFirst derivative of temp. values (ho).


# 3.8.[12].3 Sensors operational logs.
#
INS.SENSORi.INIT%sHardware initialisation (o).


#
# 4. HARPS specific
#

#
# 4.1 Special devices (preliminary)
#
# A special device is an instrument specific device not directly supported by Base ICS.
# The FITS keywords associated to a special device must be stored in a device (type)
# specific dictionary.  This dictionary has to be stored in the instrument specific
# $INS_ROOT and listed in the file $INS_ROOT/SYSTEM/COMMON/CONFIG_FILES/INS.dic, so
```

```
# that it is known before the instrument configuration itself is loaded.
#
# When the special device is registered with the Base ICS configuration UIF, then
# a new INS.CON.DEVICEi parameter is added to the instrument configuration. This
# INS.CON.DEVICEi parameter contains the device FITS prefix (e.g. “INS.TBDROT1”).
# The device specific dictionary should contain keywords starting with this prefix
# (e.g. type = “INS.TBDROTi”).
#
# Special devices can use a type name already known to Base ICS, if the DEVTYPE config.
# is specified.  Otherwise an instrument specific type should be used (e.g. “INS.TBDROTi”)
# and add an entry ‘INS.TBDROT1.PREFIX “INS.DROT1”’ to specify the prefix used for setup,
# header and log FITS keywords.
#
# The Base ICS utilities expect to find the following keywords associated to the
# special device (specified in the device type dictionary and stored in the instrument
# configuration by the configuration UIF associated to the device):
#
# INS.<type>.DEVNAME%19sName of the ICS device (C).
# INS.<type>.DEVDESC%19s    Description of the ICS device (c).
# INS.<type>.LCUID%dId. of the LCU managing the device (c).
#LCU, where the device process is running.
# INS.<type>.AVAILTDevice availability flag (c).
# INS.<type>.ENABLEDTFlag for enabling and disabling (c).
# INS.<type>.SWSIMTIf T, function is software simulated (c).
#
# INS.<type>.DEVTYPE%30sICS device OLDB type (C).
#The DEVTYPE contains the name of the OLDB class used to store
#the device data.
# INS.<type>.DEVPROC%30sICS device process (C).
#The DEVPROC contains the name of the LCU process that receives
#the device commands.
# INS.<type>.PREFIX%15sDevice FITS prefix (c).
#FITS prefix “INS.<name>” associated to the device.
#All FITS keywords starting with this prefix are sent to the
#device process.
# INS.<type>.DICT%sDevice dictionary (c).
#The device dictionary must contain all device specific FITS
#keywords: configuration, setup, header and log FITS keywords.
# INS.<type>.CONUIF%sConfiguration user interface (c).
#Name of the configuration user interface used to
#configure the special device.
#

# Lakeshore specific FITS keywords
INS.SENSORi.INPUTi%cInput channel used by managed sensor. (c)
INS.SENSORi.SETPi%fTemperature setpoint. (c)



#___oOo___
```

9.3 HARPS OS Dictionary file (dicHARPS_OS.txt)

```
#*******************************************************************************
# E.S.O. - VLT project
#
# "@(#) $Id: dicHARPS_OS.txt,v 1.1 2001/03/06 13:57:39 vltsccm Exp $"
#
# HARPS_OS dictionary
#
# who       when      what
# --------  --------  ----------------------------------------------
# jguzman 2001-03-05 Adapted to HARPS using dicXXXX v.1.15
# alongino  01/09/00  created (copied from xxodic)
#

#*******************************************************************************
#
# This dictionary contains all keywords used by HARPS OS at runtime
# to execute and understand commands with -function option, such
# as SETUP and STATUS.
#
#*******************************************************************************

DICTIONARY HARPS_OS "HARPS" "Template Instrument OS keywords"
REVISION"$Revision: 1.1 $"
CONTEXT"Instrument"

# Max. comment length ----------------------|
#                                    V
# ----------------------------------------+-----------------
# General




# --- oOo ---
```

9.4 HARPS Instrument configuration file

```
#*******************************************************************************
# E.S.O. - VLT project
#
# "@(#) $Id: hamcfgINS.cfg,v 1.2 2001/05/10 02:27:32 vltsccm Exp $"
#
# who       when      what
# --------  --------  ----------------------------------------------
# jguzman 2001-05-10 Neutral density wheel definition modified
# jguzman 2001-05-08 add exp. meter and vv analog sensors as standard devices
```

```
# jguzman 2001-03-05 Adapted for HARPS from xxmcfg version 1.30
# alongino  24/01/01  removed ACCESS keywords (in hamcfgSTART.cfg). Dedicated IRACE .cfg file
# epozna    12/01/01  VLTSW20010007: added OS config keywords (moved from haoControl.cfg)
# alongino  03/01/01  INS.CON.OPMODE removed (set in hamcfgSTART.cfg)
# rschmutz  11/12/00  INS.GRATi.TEMPREF,TEMPRAMPi added.
# rschmutz  10/12/00  ADC1 device added.
# rschmutz  07/12/00  IRCCD keywords added.
# alongino  08/11/00  removed all keys already defined in hamcfgSTART.cfg
# alongino  16/10/00  adapted to hao 1.19
# rschmutz  12/10/00  adapted to osb 1.5.
# alongino  13/06/00  added INS.CON.CONFIGSET as example for special devices
# rschmutz  07/06/00  startup config. added (shared with OS).
# alongino  18/05/00  SENSORi.NUM set to correct value
# alongino  04/02/00  GRIS and ROT devices added.
# rschmutz  01/02/00  FOCU function added.
# alongino  25/01/00  added ASSEMBLY and TEST part
# rschmutz  09/12/99  one device/type added.
# rsc/tph   19/08/99  INS.CON.ONLINE added.
# rschmutz  08/08/99  device configurations added.
# rschmutz  30/10/98  created
#
#*************************************************************************
# The following table summarizes the type of devices available and the
# corresponding name for HARPS instrument
#=========================================================================
=============================================================
# Description          | Positions  | Motor Axis    | Units| short-FITS | setup parameters  | HARPS |   icb class     |
Remarks
#=========================================================================
=============================================================
# Calibration mirror    | discrete   | linear        | none | INS.MIRR1  |NAME             | mirr | icbMOT_MIRROT
|
# ADC prism1           | continuous | circular      | deg  | INS.ADC1   |MODE             | adc1 | icbMOT_ADC      |
# ADC prism2           | continuous | circular      | deg  | INS.ADC2   |MODE             | adc2 | icbMOT_ADC      |
# ADC slide            | IN/OUT     | linear        | none | INS.OPTI1  |NAME             | adcs | icbMOT_OPTI     |
# Fiber heads selector  | HARPS/CES  | linear        | none | INS.OPTI2  |NAME             | fhsel| icbMOT_OPTI
|
# Dust cover           | IN/OUT     | linear        | none | INS.OPTI3  |NAME             | dcvr | icbMOT_OPTI     |
# Iodine cell carriage  | IN/OUT     | linear        | none | INS.OPTI4  |NAME             | iodc | icbMOT_OPTI     |
# Cal Lamp A selector   | discrete   | linear        | none | INS.OPTI5  |NAME             | casel| icbMOT_OPTI     |
# Cal Lamp B selector   | discrete   | linear        | none | INS.OPTI6  |NAME             | cbsel| icbMOT_OPTI     |
# Neutral density wheel | discrete   | circular      | none | INS.POS1   |NAME             | filt | icbMOT_FILTER   |
# Cal Lamps            | ON/OFF     | N/A           | N/A  | INS.LAMPi  |ST               | lampi| icbLAMP         |
# FF led vacuum vessel  | ON/OFF     | N/A           | N/A  | INS.LAMP6  |ST               | ffla | icbLAMP         |
# Cal Lamp A shutter    | OPEN/CLOSED| N/A           | N/A  | INS.SHUT1  |ST               | shcla| icbSHUTTER
|
# Cal Lamp B shutter    | OPEN/CLOSED| N/A           | N/A  | INS.SHUT2  |ST               | shclb| icbSHUTTER
|
# HCFA temp sensor      | continuous | N/A           | C deg| INS.SENSOR1|none             | cfas | icbSEN_ADAM
| ADAM 4017 (RS-485)
# Iodine temp sensor      | continuous | N/A               | C deg|  INS.SENSOR2|none                   | iods |
icbSEN_CN76000   | OMEGA CN76000 (RS-485)
# Exposure meter        | any        | N/A           | any  | INS.DET2   |UIT              | expm | none            |
```

```
# Vv analog sensors     | continuous | N/A          | any  | INS.SENSOR3|none              | vvas | icbSEN_ADAM
| ADAM 4017 (RS-485)
#=============================================================================
============================================================
PAF.HDR.START;                # Start of PAF Header
PAF.TYPE        “Configuration”; # Type of PAF
PAF.ID          “       “;   # ID for PAF
PAF.NAME        “       “;   # Name of PAF
PAF.DESC        “       “;   # Short description of PAF
PAF.CRTE.NAME   “       “;   # Name of creator
PAF.CRTE.DAYTIM “       “;   # Civil Time for creation
PAF.LCHG.NAME   “       “;   # Name of person/appl. changing
PAF.LCHG.DAYTIM “       “;   # Timestamp of last change
PAF.CHCK.NAME   “       “;   # Name of appl. checking
PAF.HDR.END;                  # End of PAF Header

#
# 1.0 Startup config.
#
START.CON.TYPE          “INS”; # Application type
##START.CON.TCLFILE “hainsStoo”; # Startup user extensions

#
# 1.1 General instrument config.
#
INS.CON.ID              “HARPS”; # Instrument identifier
INS.ID  “HARPS/$Revision: 1.2 $”; # Instrument identifier
INS.CON.PREFIX          “ha”; # Name prefix for modules and servers

#
# 1.2 Environments and LCUs
#
INS.CON.WSENV         “wharps”; # Workstation RTAP environment
INS.CON.LCUNUM      2; # Number of instrument LCUs
INS.CON.LCUENV1      “lhaics1”; # LCU 1 environment
INS.CON.LCUENV2      “lhaics2”; # LCU 2 environment

#
# 2.1 ICS general config.
#
# INS.CON.OPMODE removed (set in hamcfgSTART.cfg)
INS.CON.ONLINE               F; # If T, sensors are placed online during booting.
INS.CON.CONFIGSET “haiConfigSet”; # User script called by icbConfigSet
#INS.CON.CMDTOUT         15000; # Timeout for other cmds. [msec]
#INS.CON.MOVETOUT       120000; # Timeout for movement cmds. [msec]
#INS.CON.MOVETDIF          1; # Interval between parallel movements [msec]
#INS.CON.MONTIME         60000; # Device monitoring period [msec]
#INS.CON.MOVEMAX           100; # Max. number of devices moving in parallel

# Device FITS prefix used in the config file
INS.CON.DEVNUM              22; # Number of ICS devices
INS.CON.DEVICE1     “INS.MIRR1”; # Calibration mirror
INS.CON.DEVICE2      “INS.ADC1”; # ADC prism 1
```

```
INS.CON.DEVICE3     “INS.ADC2”; # ADC prism 2
INS.CON.DEVICE4    “INS.OPTI1”; # ADC slide
INS.CON.DEVICE5    “INS.OPTI2”; # Fiber heads selector
INS.CON.DEVICE6    “INS.OPTI3”; # Fiber heads protector
INS.CON.DEVICE7    “INS.OPTI4”; # Iodine cell carriage
INS.CON.DEVICE8    “INS.OPTI5”; # Calibration Lamp A selector
INS.CON.DEVICE9    “INS.OPTI6”; # Calibration Lamp B selector
INS.CON.DEVICE10    “INS.POS1”; # Neutral density wheel
INS.CON.DEVICE11    “INS.LAMP1”; # Calibration Lamp 1
INS.CON.DEVICE12    “INS.LAMP2”; # Calibration Lamp 2
INS.CON.DEVICE13    “INS.LAMP3”; # Calibration Lamp 3
INS.CON.DEVICE14    “INS.LAMP4”; # Calibration Lamp 4
INS.CON.DEVICE15    “INS.LAMP5”; # Calibration Lamp 5
INS.CON.DEVICE16    “INS.LAMP6”; # Calibration Lamp 6
INS.CON.DEVICE17    “INS.SHUT1”; # Calibration Lamp A shutter
INS.CON.DEVICE18    “INS.SHUT2”; # Calibration Lamp B shutter
INS.CON.DEVICE19     “INS.DET2”; # Exposure meter
INS.CON.DEVICE20  “INS.SENSOR1”; # HCFA body temperature sensor (ADAM)
INS.CON.DEVICE21  “INS.SENSOR2”; # Iodine cell temperature controller (CN76000)
INS.CON.DEVICE22  “INS.SENSOR3”; # Vacuum vessel pressure & LN2 level sensors (ADAM)
# special devices
#INS.CON.DEVICE23  “INS.SENSOR4”; # CCD temp. controller (Lakeshore)
#INS.CON.DEVICE24  “INS.SENSOR5”; # Vacuum vessel temp. controller (Lakeshore)
#INS.CON.DEVICE25  “INS.SENSOR6”; # HARPS room & external temp. sensors (PREMA)
#INS.CON.DEVICE26  “INS.SENSOR7”; # Spectrograph temp. sensors (PREMA)

#
# Calibration light projection mirror
#
INS.MIRR1.DEVNAME      “mirr”; # Name of the ICS device
INS.MIRR1.DEVDESC “Calib. mirror carriage”; # Description of the ICS device
INS.MIRR1.LCUID            1; # Id. of the LCU managing the device
INS.MIRR1.SWSIM            T; # If T, function is software simulated

INS.MIRR1.POSNUM           4;
INS.MIRR1.POSID1      “FIBA”;
INS.MIRR1.POSID2      “BOTH”;
INS.MIRR1.POSID3      “FIBB”;
INS.MIRR1.POSID4      “NONE”;

INS.MIRR1.ID1         “FIBA”;
INS.MIRR1.NAME1       “FIBA”;
INS.MIRR1.ID2         “BOTH”;
INS.MIRR1.NAME2       “BOTH”;
INS.MIRR1.ID3         “FIBB”;
INS.MIRR1.NAME3       “FIBB”;
INS.MIRR1.ID4         “NONE”;
INS.MIRR1.NAME4       “NONE”;

#
# ADC
#
INS.ADC1.DEVNAME       “adc1”; # Name of the ICS device
```

```
INS.ADC1.DEVDESC “ADC prism 1”; # Description of the ICS device
INS.ADC1.LCUID          1; # Id. of the LCU managing the device
INS.ADC1.SWSIM          T; # If T, function is software simulated

INS.ADC1.PERIOD      15000;
INS.ADC1.REFENC          0;
INS.ADC1.MOTSIGN         1;
INS.ADC1.FORMULA         1;
INS.ADC1.MINELEV     27.64;
INS.ADC1.POSOFFST       90;
INS.ADC1.COFFSET    1.7387;
INS.ADC1.PSLOPE     0.0023;
INS.ADC1.POFFSET       743;
INS.ADC1.TSLOPE    -0.0061;
INS.ADC1.TOFFSET        12;
INS.ADC1.AFACTOR      3.32;
INS.ADC1.DROTFACT        0; # use 2 or -2 if derotator is present

INS.ADC2.DEVNAME     “adc2”; # Name of the ICS device
INS.ADC2.DEVDESC “ADC prism 2”; # Description of the ICS device
INS.ADC2.LCUID          1; # Id. of the LCU managing the device
INS.ADC2.SWSIM          T; # If T, function is software simulated

INS.ADC2.PERIOD      15000;
INS.ADC2.REFENC          0;
INS.ADC2.MOTSIGN         1;
INS.ADC2.FORMULA         1;
INS.ADC2.MINELEV     27.64;
INS.ADC2.POSOFFST       90;
INS.ADC2.COFFSET    1.7387;
INS.ADC2.PSLOPE     0.0023;
INS.ADC2.POFFSET       743;
INS.ADC2.TSLOPE    -0.0061;
INS.ADC2.TOFFSET        12;
INS.ADC2.AFACTOR      3.32;
INS.ADC2.DROTFACT        0; # use 2 or -2 if derotator is present

#
# ADC carriage or slide
#
INS.OPTI1.DEVNAME     “adcs”; # Name of the ICS device
INS.OPTI1.DEVDESC “ADC slide”; # Description of the ICS device
INS.OPTI1.PREFIX   “INS.ADCS”; # FITS prefix
INS.OPTI1.LCUID         1; # Id. of the LCU managing the device
INS.OPTI1.SWSIM         T; # If T, function is software simulated

INS.OPTI1.POSNUM    2;
INS.OPTI1.POSID1      “IN”;
INS.OPTI1.POSID2      “OUT”;

INS.OPTI1.ID1        “IN”;
INS.OPTI1.NAME1       “IN”;
INS.OPTI1.TYPE1      “ADC”; # Element type
```

```
INS.OPTI1.ID2          “OUT”;
INS.OPTI1.NAME2        “OUT”;
INS.OPTI1.TYPE2        “ADC”; # Element type

#
# Fiber heads selector
#
INS.OPTI2.DEVNAME      “fhsel”; # Name of the ICS device
INS.OPTI2.DEVDESC “Fiber heads selector”; # Description of the ICS device
INS.OPTI2.LCUID            1; # Id. of the LCU managing the device
INS.OPTI2.SWSIM            T; # If T, function is software simulated

INS.OPTI2.POSNUM    2;
INS.OPTI2.POSID1     “HARPS”;
INS.OPTI2.POSID2       “CES”;

INS.OPTI2.ID1        “HARPS”;
INS.OPTI2.NAME1      “HARPS”;
INS.OPTI2.TYPE1      “FIBER”; # Element type
INS.OPTI2.ID2          “CES”;
INS.OPTI2.NAME2        “CES”;
INS.OPTI2.TYPE2      “FIBER”; # Element type

#
# Dust cover
#
INS.OPTI3.DEVNAME      “dcvr”; # Name of the ICS device
INS.OPTI3.DEVDESC “Dust cover”; # Description of the ICS device
INS.OPTI3.LCUID            1; # Id. of the LCU managing the device
INS.OPTI3.SWSIM            T; # If T, function is software simulated

INS.OPTI3.POSNUM    2;
INS.OPTI3.POSID1     “IN”;
INS.OPTI3.POSID2     “OUT”;

INS.OPTI3.ID1        “IN”;
INS.OPTI3.NAME1      “IN”;
INS.OPTI3.TYPE1      “COVER”; # Element type
INS.OPTI3.ID2        “OUT”;
INS.OPTI3.NAME2      “OUT”;
INS.OPTI3.TYPE2      “COVER”; # Element type

#
# Iodine cell carriage
#
INS.OPTI4.DEVNAME      “iodc”; # Name of the ICS device
INS.OPTI4.DEVDESC “Iodine cell carriage”; # Description of the ICS device
INS.OPTI4.LCUID            1; # Id. of the LCU managing the device
INS.OPTI4.SWSIM            T; # If T, function is software simulated

INS.OPTI4.POSNUM    2;
INS.OPTI4.POSID1     “IN”;
INS.OPTI4.POSID2     “OUT”;
```

```
INS.OPTI4.ID1       “IN”;
INS.OPTI4.NAME1     “IN”;
INS.OPTI4.TYPE1     “IN”; # Element type
INS.OPTI4.ID2       “OUT”;
INS.OPTI4.NAME2     “OUT”;
INS.OPTI4.TYPE2     “OUT”; # Element type

#
# Calibration lamp A selector
#
INS.OPTI5.DEVNAME     “casel”; # Name of the ICS device
INS.OPTI5.DEVDESC “Cal Lamp A selector”; # Description of the ICS device
INS.OPTI5.LCUID          2; # Id. of the LCU managing the device
INS.OPTI5.SWSIM          T; # If T, function is software simulated

INS.OPTI5.POSNUM   5;
INS.OPTI5.POSID1    “NONE”;
INS.OPTI5.POSID2    “THAR”;
INS.OPTI5.POSID3    “FF”;
INS.OPTI5.POSID4    “SPARE1”;
INS.OPTI5.POSID5    “SPARE2”;

INS.OPTI5.ID1       “NONE”;
INS.OPTI5.NAME1     “NONE”;
INS.OPTI5.TYPE1     “FREE”; # Element type
INS.OPTI5.ID2       “THAR”;
INS.OPTI5.NAME2     “THAR”;
INS.OPTI5.TYPE2     “LAMP”; # Element type
INS.OPTI5.ID3       “FF”;
INS.OPTI5.NAME3     “FF”;
INS.OPTI5.TYPE3     “LAMP”; # Element type
INS.OPTI5.ID4       “SPARE1”;
INS.OPTI5.NAME4     “SPARE1”;
INS.OPTI5.TYPE4     “LAMP”;
INS.OPTI5.ID5       “SPARE2”;
INS.OPTI5.NAME5     “SPARE2”;
INS.OPTI5.TYPE5     “LAMP”;

#
# Calibration lamp B selector
#
INS.OPTI6.DEVNAME     “cbsel”; # Name of the ICS device
INS.OPTI6.DEVDESC “Cal Lamp B selector”; # Description of the ICS device
INS.OPTI6.LCUID          2; # Id. of the LCU managing the device
INS.OPTI6.SWSIM          T; # If T, function is software simulated

INS.OPTI6.POSNUM   5;
INS.OPTI6.POSID1    “NONE”;
INS.OPTI6.POSID2    “THAR”;
INS.OPTI6.POSID3    “FF”;
INS.OPTI6.POSID4    “SPARE1”;
INS.OPTI6.POSID5    “SPARE2”;
```

```
INS.OPTI6.ID1        “NONE”;
INS.OPTI6.NAME1      “NONE”;
INS.OPTI6.TYPE1      “FREE”; # Element type
INS.OPTI6.ID2        “THAR”;
INS.OPTI6.NAME2      “THAR”;
INS.OPTI6.TYPE2      “LAMP”; # Element type
INS.OPTI6.ID3        “FF”;
INS.OPTI6.NAME3      “FF”;
INS.OPTI6.TYPE3      “LAMP”; # Element type
INS.OPTI6.ID4        “SPARE1”;
INS.OPTI6.NAME4      “SPARE1”;
INS.OPTI6.TYPE4      “LAMP”;
INS.OPTI6.ID5        “SPARE2”;
INS.OPTI6.NAME5      “SPARE2”;
INS.OPTI6.TYPE5      “LAMP”;

#
# Neutral density wheel
#
INS.POS1.DEVNAME      “filt”; # Name of the ICS device
INS.POS1.DEVDESC “Neutral density wheel”; # Description of the ICS device
INS.POS1.LCUID            1; # Id. of the LCU managing the device
INS.POS1.SWSIM            T; # If T, function is software simulated
INS.POS1.PREFIX    “INS.ROT1”; # FITS prefix

INS.POS1.UNIT          “deg”; # Unit
INS.POS1.OFFSET        1000; # Reference position
INS.POS1.RAMP           10; # Ramp
INS.POS1.POSMIN         0.0; # Min. focus position
INS.POS1.POSMAX        360.0; # Max. focus position


#
# lamps
#
INS.LAMP1.DEVNAME     “lamp1”; # Name of the ICS device
INS.LAMP1.DEVDESC “ThAr lamp1”; # Description of the ICS device
INS.LAMP1.LCUID           2; # Id. of the LCU managing the device
INS.LAMP1.SWSIM           T; # If T, function is software simulated

INS.LAMP1.ID       “TAL1”;
INS.LAMP1.NAME    “ThAr_Lamp1”;
INS.LAMP1.SIGDEV     “/acro0”;
INS.LAMP1.SIGBIT1         52; # lampSwitchO output signal
INS.LAMP1.SIGBIT4         53; # lampOnI feedback signal
INS.LAMP1.IGFAULT          T;

INS.LAMP2.DEVNAME     “lamp2”; # Name of the ICS device
INS.LAMP2.DEVDESC “ThAr lamp2”; # Description of the ICS device
INS.LAMP2.LCUID           2; # Id. of the LCU managing the device
INS.LAMP2.SWSIM           T; # If T, function is software simulated
```

```
INS.LAMP2.ID       “TAL2”;
INS.LAMP2.NAME    “ThAr_Lamp2”;
INS.LAMP2.SIGDEV     “/acro0”;
INS.LAMP2.SIGBIT1         54;
INS.LAMP2.IGFAULT          T;

INS.LAMP3.DEVNAME      “lamp3”; # Name of the ICS device
INS.LAMP3.DEVDESC   “FF lamp1”; # Description of the ICS device
INS.LAMP3.LCUID            2; # Id. of the LCU managing the device
INS.LAMP3.SWSIM            T; # If T, function is software simulated

INS.LAMP3.ID      “FFL1”;
INS.LAMP3.NAME     “FF_Lamp1”;
INS.LAMP3.SIGDEV     “/acro0”;
INS.LAMP3.SIGBIT1         55;
INS.LAMP3.IGFAULT          T;

INS.LAMP4.DEVNAME      “lamp4”; # Name of the ICS device
INS.LAMP4.DEVDESC   “FF lamp2”; # Description of the ICS device
INS.LAMP4.LCUID            2; # Id. of the LCU managing the device
INS.LAMP4.SWSIM            T; # If T, function is software simulated

INS.LAMP4.ID       “FFL2”;
INS.LAMP4.NAME      “FF_Lamp2”;
INS.LAMP4.SIGDEV     “/acro0”;
INS.LAMP4.SIGBIT1         56;
INS.LAMP4.IGFAULT          T;

INS.LAMP5.DEVNAME      “lamp5”; # Name of the ICS device
INS.LAMP5.DEVDESC   “Gen lamp”; # Description of the ICS device
INS.LAMP5.LCUID            2; # Id. of the LCU managing the device
INS.LAMP5.SWSIM            T; # If T, function is software simulated

INS.LAMP5.ID      “GENL”;
INS.LAMP5.NAME     “Gen_Lamp”;
INS.LAMP5.SIGDEV     “/acro0”;
INS.LAMP5.SIGBIT1         57;
INS.LAMP5.IGFAULT          T;

INS.LAMP6.DEVNAME      “ffla”; # Name of the ICS device
INS.LAMP6.DEVDESC   “FF led vv”; # Description of the ICS device
INS.LAMP6.LCUID            2; # Id. of the LCU managing the device
INS.LAMP6.SWSIM            T; # If T, function is software simulated

INS.LAMP6.ID      “FFLED”;
INS.LAMP6.NAME       “FF_Led”;
INS.LAMP6.SIGDEV     “/acro0”;
INS.LAMP6.SIGBIT1         58;
INS.LAMP6.IGFAULT          T;


#
# Calibration lamp A shutter
```

```
#
INS.SHUT1.DEVNAME       “shcla”; # Name of the ICS device
INS.SHUT1.DEVDESC “Cal lamp A shutter”; # Description of the ICS device
INS.SHUT1.LCUID           2; # Id. of the LCU managing the device
INS.SHUT1.SWSIM           T; # If T, function is software simulated

INS.SHUT1.ID       “SHUA”;
INS.SHUT1.NAME  “CalA_Shutter”;
INS.SHUT1.SIGDEV     “/acro0”;
INS.SHUT1.SIGBIT1         0;
INS.SHUT1.IGFAULT         T;

#
# Calibration lamp B shutter
#
INS.SHUT1.DEVNAME       “shclb”; # Name of the ICS device
INS.SHUT1.DEVDESC “Cal lamp B shutter”; # Description of the ICS device
INS.SHUT1.LCUID           2; # Id. of the LCU managing the device
INS.SHUT1.SWSIM           T; # If T, function is software simulated

INS.SHUT1.ID       “SHUB”;
INS.SHUT1.NAME  “CalB_Shutter”;
INS.SHUT1.SIGDEV     “/acro0”;
INS.SHUT1.SIGBIT1         1;
INS.SHUT1.IGFAULT         T;

#
# Exposure meter
#
INS.DET2.DEVNAME      “expm”; # Name of the ICS device
INS.DET2.DEVDESC “Exposure meter”; # Description of the ICS device
INS.DET2.LCUID          2;
INS.DET2.SWSIM          T;

INS.DET2.PORT       “/tyCo1”;

#
# HCFA body temperature sensor
#
INS.SENSOR1.DEVNAME       “cfas”; # Name of the ICS device
INS.SENSOR1.DEVDESC “CFA temperature sensor”; # Description of the ICS device
INS.SENSOR1.DEVTYPE      “ADAM”; # Device type
INS.SENSOR1.LCUID          1; # Id. of the LCU managing the device
INS.SENSOR1.SWSIM          T; # If T, function is software simulated

INS.SENSOR1.PORT       “/iser0”; # Hardware device
INS.SENSOR1.NUM      1; # Number of managed sensor values

INS.SENSOR1.NAME1       “CFAS”; # Sensor value name
INS.SENSOR1.DESC1 “HCFA body temperature sensor”; # Sensoar value description
INS.SENSOR1.HEADER1         T; # If T, report sensor value in image header
INS.SENSOR1.FITS1 “INS.TEMP1.VAL”; # Sensor value FITS keyword
INS.SENSOR1.SENUNIT1       “C”; # Sensor value unit
```

```
INS.SENSOR1.SENADDR1        01; # Sensor value hardware address
INS.SENSOR1.SENTYPE1   “ON_OFF”; # Sensor value report type
INS.SENSOR1.SENREF1        2.0; # Sensor value reference parameter
INS.SENSOR1.SENRAMP1      -1.0; # Sensor value ramp parameter

#
# Iodine temperature controller
#
INS.SENSOR2.DEVNAME     “iods”; # Name of the ICS device
INS.SENSOR2.DEVDESC “Iodine cell temp. controller”; # Description of the ICS device
INS.SENSOR2.DEVTYPE   “CN76000”; # Device type
INS.SENSOR2.LCUID           1; # Id. of the LCU managing the device
INS.SENSOR2.SWSIM           T; # If T, function is software simulated

INS.SENSOR2.PORT      “/iser1”; # Hardware device
INS.SENSOR2.NUM     1; # Number of managed sensor values

INS.SENSOR2.NAME1       “IODS”; # Sensor value name
INS.SENSOR2.DESC1 “Iodine cell temp. sensor”; # Sensoar value description
INS.SENSOR2.HEADER1         T; # If T, report sensor value in image header
INS.SENSOR2.FITS1 “INS.TEMP2.VAL”; # Sensor value FITS keyword
INS.SENSOR2.SENUNIT1       “C”; # Sensor value unit
INS.SENSOR2.SENADDR1        10; # Sensor value hardware address

#
# Vacuum vessel analog sensors
#
INS.SENSOR3.DEVNAME      “vvas”; # Name of the ICS device
INS.SENSOR3.DEVDESC “V.vessel analog sensors”; # Description of the ICS device
INS.SENSOR3.DEVTYPE     “ADAM”; # Device type
INS.SENSOR3.LCUID           2; # Id. of the LCU managing the device
INS.SENSOR3.SWSIM           T; # If T, function is software simulated

INS.SENSOR3.PORT      “/iser0”; # Hardware device
INS.SENSOR3.NUM     2; # Number of managed sensor values
# Vaccum vessel pressure sensor
INS.SENSOR3.NAME1       “PRSS”; # Sensor value name
INS.SENSOR3.DESC1 “V.vessel pressure sensor”; # Sensoar value description
INS.SENSOR3.HEADER1         T; # If T, report sensor value in image header
INS.SENSOR3.FITS1 “INS.SENS1.VAL”; # Sensor value FITS keyword
INS.SENSOR3.SENUNIT1     “mbar”; # Sensor value unit
INS.SENSOR3.SENADDR1        01; # Sensor value hardware address
INS.SENSOR3.SENTYPE1   “PRESSURE”; # Sensor value report type
INS.SENSOR3.SENREF1        2.0; # Sensor value reference parameter
INS.SENSOR3.SENRAMP1      -1.0; # Sensor value ramp parameter
# LN2 level sensor
INS.SENSOR3.NAME2       “LNLV”;
INS.SENSOR3.DESC2  “LN2 level sensor”;
INS.SENSOR3.HEADER2       “T”;
INS.SENSOR3.FITS2 “INS.SENS2.VAL”;
INS.SENSOR3.SENUNIT2      “%”;
INS.SENSOR3.SENADDR2       02;
INS.SENSOR3.SENTYPE2  “ANALOG”;
```

```
INS.SENSOR3.SENREF2       2.0;
INS.SENSOR3.SERRAMP2      1.0;

#
# Special devices
#



#
# 2.2 ICS Assemblies
#
# accept any value of INS.MODE and do not forward this key to the LCUs
INS.ASSEMBLY1 “INS.MODE”; # Assembly FITS name
INS.ASSEMBLY1.KEY1 “*”
INS.ASSEMBLY1.VAL1 ““

#
# 2.3 ICS self-test utility (ic0SelfTest) commands
#

#*************************************************************************
# 5. OS configuration
#

OCS.CON.RELEASE  “2000-06-16”; # Release date “yyyy-mm-dd”
OCS.CON.ORIGIN         “TEST”; # Origin
OCS.CON.LOGLEVEL            0;

#
# 5. OS Subsystems
#

#
# 5.1 OS telescope control subsystems
#
OCS.TEL.NAME       “3P6”;              # Telescope name (‘UT1’,’UT2’,’UT3’ or ‘UT4’)
OCS.TEL.FOCUS      “CA”;               # Telescope focus
OCS.TEL.DICT         “TCS”;              # dictionary
OCS.TEL.ENVNAME       “wtcssim”;           #
OCS.TEL.PROCNAME      “tifCA” ;           # process name
OCS.TEL.DBSTATE       “<alias>TCS:tcsState.tcsSubstate”;   #
OCS.TEL.KEYWFILT      “TEL.*.*.*.*.*.*”;     # keyword filter
OCS.TEL.TIMEOUT      180000;             # timout in seconds


#
# 5.2 OS instrument control subsystems
#
#    Category indices are always removed before forwarding the FITS keyword.
#
OCS.INS.NUM1;
```

```
# subsystem: ICS
# ----------------------------------------------------------------------------------------
OCS.INS.NAME“ICS”;
OCS.INS.DICT1          “HARPS_ICS”;
OCS.INS.ENVNAME        “wharps”;
OCS.INS.PROCNAME“haiControl”;
OCS.INS.DBSTATE        “<alias>HARPS:ICS:PROCESSES:WS:icsControl.state”;
OCS.INS.KEYWFILT       “INS.*.*.*.*.*.*”;
OCS.INS.TIMEOUT        60000;

OCS.INS.STRTUIFF;


#
# 5.3 OS detector control subsystems
#
#     Category indices are always removed before forwarding the FITS keyword.
#
OCS.DET.NUM1;

# subsystem: DET1 FIERA
# ----------------------------------------------------------------------------------------
OCS.DET1.NAME        “FIERA” ;
OCS.DET1.DICT        “FCDDCS”;
OCS.DET1.ENVNAME     “wharps”;
OCS.DET1.PROCNAME    “fcdconCI_fiera”;
OCS.DET1.KEYWFILT    “DET1.*.*.*.*.*.*”;
OCS.DET1.TIMEOUT     1000;
OCS.DET1.DBSTATE     “<alias>fiera.opState”;
OCS.DET1.DBNEWDT     “<alias>fiera:exposures:exposure_1:transfer.fileNameUnComp”;
OCS.DET1.DBEXPSTS    “<alias>fiera:exposures:exposure_1.expStatus”;

OCS.DET1.TYPE     “FIERA”;
OCS.DET1.CCDNAME     “fiera”;
OCS.DET1.CCDLENV     “whafcd”;
OCS.DET1.DBFILE     “fcdSciTemplate.dbcfg”;

#
# 5.4 OS observation software subsystems
#
OCS.OCS.NUM0;


#
# 5.5 Instrument modes
#
OCS.MODE1.NAME       “SIMULTAN_WAVE_REF”;                    # name of the instrmode
OCS.MODE1.SETUP      “-function INS.OPTI2.NAME OUT “;  # setup the mode
OCS.MODE1.SUBSYST    “FIERA ICS 3P6”;                  # subsystems involved in the given mode
OCS.MODE1.PATH       ““;                       # instrument path (EXPSTRT, EXPEND)

OCS.MODE2.NAME       “IODINE_SPECTROSCOPY”;                      # name of the instrmode
OCS.MODE2.SETUP      “-function INS.OPTI2.NAME IN”;    # setup of the given mode
```

```
OCS.MODE2.SUBSYST    “FIERA ICS 3P6”;                    # subsystems involved in the given mode
OCS.MODE2.PATH       ““;                    # instrument path (EXPSTRT, EXPEND)

OCS.MODE3.NAME       “OBJECT_SPECTROSCOPY”;              # name of the instrmode
OCS.MODE3.SETUP      “-function INS.OPTI2.NAME OUT “;  # setup of  mode
OCS.MODE3.SUBSYST    “FIERA ICS 3P6”;             # subsystems involved in the given mode
OCS.MODE3.PATH       ““;                    # instrument path (EXPSTRT, EXPEND)

OCS.MODE4.NAME       “OBJ_SKY_SPECTROSCOPY”;                        # name of the instrmode
OCS.MODE4.SETUP      “-function INS.OPTI2.NAME OUT”;  # setup of mode
OCS.MODE4.SUBSYST    “FIERA ICS 3P6”;                       # subsystems
OCS.MODE4.PATH       ““;                          # instrument path




#*************************************************************************
# 6. Additional panels
#

START.PANEL1.NAME    “ALARM”; # Panel name
START.PANEL1.EXECMD  “haopanAlarm”; # Command to start the panel
START.PANEL1.DESC    “HARPS Alarms”
START.PANEL1.DEFAULT F;


START.PANEL2.NAME    “OS_ENGINEERING”; # Panel name
START.PANEL2.EXECMD  “haopanEngineering”; # Command to start the panel
START.PANEL2.DESC    “HARPS OS Engineering”
START.PANEL2.DEFAULT F;


#*************************************************************************
# ___oOo___
```