Paper IV

ATLAS DCS

SCT Power Supply PVSS Software

Document Version: 1.0
Document ID: ATLAS-IS-ON-0001
Document Date: 08.05.2007
Document Status: In Work

Authors and Institutes:

Bjarte Mohn (University of Bergen) , Peter W. Phillips (Rutherford Appleton Laboratory) and Ewa Stanecka (Henryk Niewodniczanski Inst. Nucl. Physics, PAN)

Table of Contents

1. About this Document

This document is intended to give an overview of the present version of the ATLAS SCT Power Supply software and its PVSS implementation in particular. Focus is upon the user interfaces, the way they work and how the detector can be operated using them. At the same time the user interfaces are graphical interfaces to an underlying structure of scripts and libraries, and thus some details about them and how they work are also given. No details of the actual code are given, but for the interested reader the code should be relatively easy to read as it is well commented and documented.

Useful Links

SCTPowerSupply: https://twiki.cern.ch/twiki/bin/view/Atlas/SCTPowerSupply

SCTDCS: https://twiki.cern.ch/twiki/bin/view/Atlas/SCTDCS

DCS documents: https://twiki.cern.ch/twiki/bin/view/Atlas/DcsDocu

2. Hardware

The fundamental power supply unit of control in terms of hardware is the power supply crate which contains 12 LV cards, 6 HV cards, 1 interlock card and 1 crate controller card. The Crate Controller (CC) card is the interface between the power supply cards and the PVSS application. It handles multiplexed commands from the PVSS project and translates it into sequences of single commands and propagates them to the cards. The interplay between the CC and the PVSS project is the core of the SCT power supply and thus the following section is gives a brief overview of the CC and the command protocol used between the CC and PVSS.

2.1. Crate Controller firmware

The crate controller works with channel states where each LV or HV channel is assigned with a state. During operation the CC will report the state of each channel upon each readout of the crate and the list of possible states are given in Table 1 – "in state" list.

States are also used when voltages are ramped up and down, thus we talk about a state transition when a channel is commanded from one state to another. The commandable states is a subset of the "in state" list (also listed in Table 1) where ON, OFF and STANDBY are the main states. The voltages and settings for these states are preloaded to the CC's EEPROM and when a request to move into one of these states is received the values are loaded from the EEPROM to the CC's RAM and subsequently sent to the channel.

In the EEPROM the CC also stores information about which group (in practice the cooling loop) a channel belongs to. This information enables sending commands to groups of channels which makes it much faster to apply a state transition to a group of channels as compared to sending sequential commands to each channel in the group. It also makes the system more reliable since it reduces the risk for missed commands simply by sending fewer commands.

Finally the CC can also disable (enable) a channel. When disabled a channel will not respond to any commands until it has been enabled again. This feature allows a module to be temporarily taken out of the mapping in the event of problems.

The format of the set command sent from the PVSS project to the CC is shown in Table 1.

Command G|H|L|nnnnnn|iiii|ooo					
G	H	L	nnnnnn	iiii	ooo
Group 0 1	HV 0 1	LV 0 1	0 – 64 Channel or Group Number	In state 0 OFF 1 ON 2 STANDBY 3 MANUAL 4 MASK OFF 5 MASK ON 6 HARD RESET 7 DISABLE/ENABLE 9 INTERLOCKED A TRIP HW B TRIP SW C LVCARD LATCH D NO MATCH E UNKNOWN F ANY	Out state 0 OFF 1 ON 2 STANDBY 3 MANUAL 4 MASK OFF 5 MASK ON 6 HARD RESET 7 DISABLE/ENABLE

Table 1: Format of the set command sent to the Crate Controller.

3. System Requirements

In order to run the system properly the following software and versions needs to be installed properly:

- Latest version of the CrateController firmware.
- HV card firmware version 3.4 and LV card firmware version 5.7 (The CC firmware is however backwards compatible with all previous versions)
- PVSS, release 3.6.

4. Installing and Setting up the Project

Project installation assumes that PVSS already has been properly installed on the computer. Download the latest version of the project from the SCTPowerSupply wikipage (link given above) and unzip it where the installation should be made. Start the PVSS II Project Administration panel and choose “Register new project” from the top menu. Browse to the folder with the unzipped project and click “OK”. This completes the installation.

The next step is the project setup which defines the number of crates connected to the computer, what their hardware and software addresses are and which BUS they are connected to. To do this, start the project and wait for the appearance of the HVLV panel. This panel will be thoroughly discussed below and thus we here only look at its

setup functionality. Shown at position 4 of Figure 5.1 there is a button “Create DP” which will bring up the project’s main setup panel (file.pnl). Figure 4.1 shows the setup panel (after the setup table has been filled) and below a brief description is given of the setup sequence:

- First the setup table is filled with the information about the crates as shown on Figure 4.1. Remember to set the crates’ hardware addresses correct!
- Create the OPC server config file by clicking the appropriate button. This will write an OPC server config file to <project_path>/panels/OPCsettings.cfg. Copy file to OPC server directory and restart the projects OPC server.
- Create datapoints for crate, cards and channels.
- Set the addresses for the datapoints.
- Set conversion
- Create alerts. This will create the alerts for all datapoints which have alerts associated with them (including summary alerts), but only fill them with dummy values.

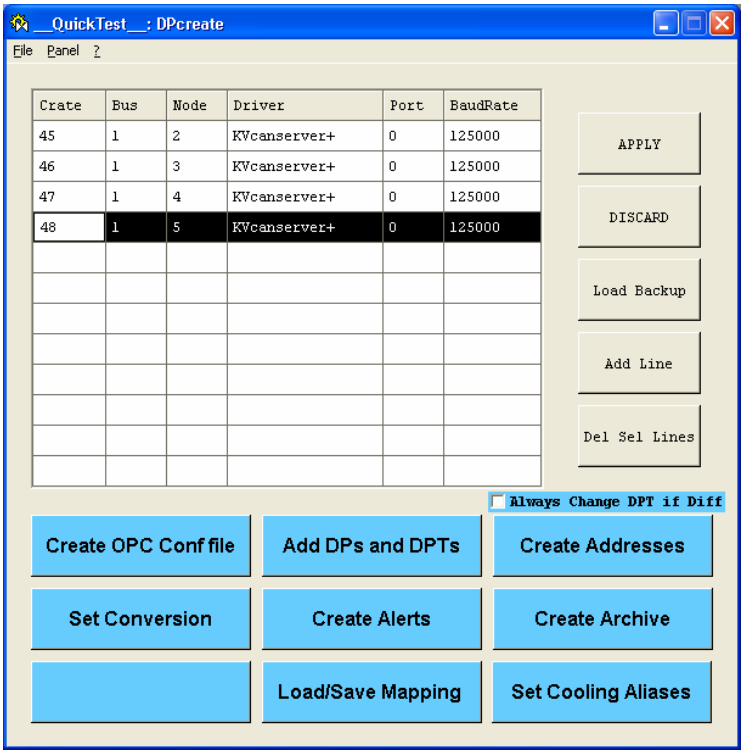

Figure 4.1: The project setup panel.

The last setup step is to “load/save mapping” to the datapoints, i.e. to set aliases to the channel datapoints so that each channel is associated with module on the detector. The alias is used for geographical mapping of the data in the project and thus utmost important. The mapping functionality in the project reads the mapping from a mapping file, Cnf_Map.txt, which is generated by the DAQ powerdump utility. Thus, before the mapping can be applied to the system the DAQ has to produce the mapping

for the setup in question. The output mapping file and configuration files should be saved to the following folder: *C:\SCT_DCS\Configs\Mops*

While the steps above complete the software setup on the computer there is one more step needed before any power can be applied to the modules. The crate controllers need to be configured, i.e. the configurations databases should be uploaded to them. This final step is not done from the setup panel, but from a command line panel and is discussed below in Section 0.

5. The HVLV Panel

The HVLV, or MoPS diagnostic, panel is the main user interface panel of the Power Supply PVSS project. It consists of several sub panels that are located in the panels/subpanels folder of the project and it allows the user to change between these in a convenient way. Figure 5.1 shows a typical appearance of the HVLV panel. The numbers mark where panels of similar functionality are grouped and below a description of the different panels are given:

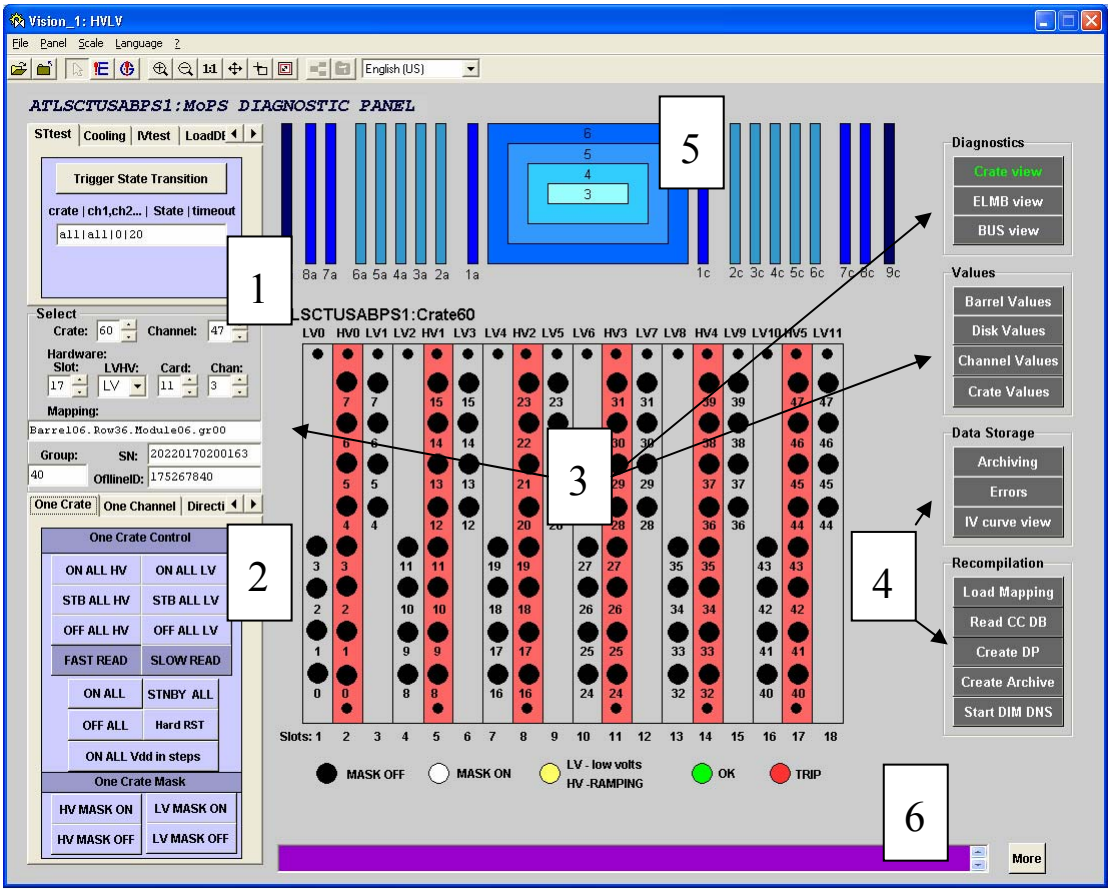

Figure 5.1: The HVLV panel is the main user interface of the MoPS project.

Command Line panels [1]

Location: <*projectpath*>/panels/subpanels/cmd_line_panels

Functionality: The command line panels take a string input and write this string to the corresponding command datapoint of the project. The command is then picked up by a script connected to this datapoint as explained below in Section 6 and executed.

Not all panels in group 1 take string input; some only monitor the status of other subsystems, like the environmental project or cooling project. Below a short description is given of each panel:

STtest (*testRST.pnl*): Takes string input of format 'crate|channel|state|timeout' and writes string to the datapoint for state transitions. The command/string is picked up by the ChangeState script and executed. See Section 6 for details about valid commands.

LoadDB (LoadDB.pnl): Loads channel configuration, i.e. parameter set points for the different run states OFF, STANDBY and ON to the CrateController's memory. String input 'crate|channel|state' specifies which state(s) should be loaded for which channel(s). The panel supports reading the configuration from ASCII text files but it is recommended to use the Oracle database.

HardReset (TestHR.pnl): Command line panel which takes the input string 'crate|channel|timeout'. String command is written to the datapoint for hard reset commands and channels are reset by the hardReset script. See Section 6 for details about the hard reset command sent to the channel(s).

Mask (mask.pnl): Panel to perform masking transitions according to the inputted string. String format is 'crate|channel|state|timeout' where state can be given either as a hex or decimal number. Action is handled by the ChangeMask script and more details about valid string formats are given in Section 6.

ChangeParam (TestParam.pnl): Using this panel it is possible to change a single parameter at the time for either a single channel, but also groups of channels. The input string is 'crate|channel|parameter|value|timeout' where 'parameter' is the parameter to be changed and 'value' is the new set point value. The input string is written to the datapoint for parameter changes and action is handled by the ChangeParam script. See section 6 for details about which parameters that can be changed using this interface.

5.1. Command Panels [2]

Location: <*projectpath*>/panels/subpanels/command_panels

Functionality: The command panels have similar functionality to the command line panels but they offer buttons instead of strings inputs for the most common commands. Figure 5.2 gives an overview of the command panels used in the project. Below a short introduction to each panel is given:

Channel/Card Control (Local.pnl): Shown at position 1 on Figure 5.2 the Channel/Card control panel offers single channel or card control where the channel or card can be selected from the select panel as discussed in Section 5.2. The panel is divided in two parts where the left column involve HV transitions and right column LV transitions. When pressed, the yellow buttons write the configuration settings for the relevant channel's (or channels' if for a card) to the CrateController's RAM. Once written to the RAM transitions can be made using the orange buttons which sends the

command to the CrateController to apply the RAM settings to channel/card. This panel is intended to be used together with the ChannelValues panel (as discussed in Section 5.2) and together they provide the user with a complete control of a channel/card including manual transitions.

One Crate Control (Global.pnl): Shown at position 2 on Figure 5.2 the panel for one crate control features buttons for all predefined state transitions for all channels in the selected crate. Once pressed each button will send a command string to the ChangeState script, which handles the transition and all channels in the crate that has a valid mapping and that is masked on will respond to a state transitions. For masking commands only channels with valid mapping will respond. Care has to be taken when masking off the crate as masking off LV would take action even if power is on for channels in the crate. Masking off HV would give a pop up asking the user for a confirmation if there are channels with HV on.

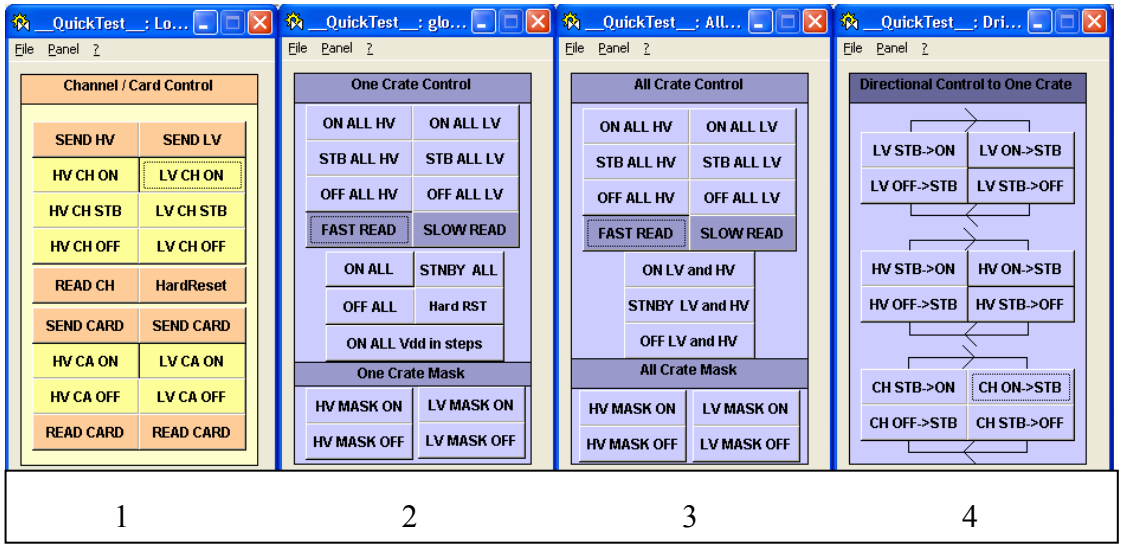

Figure 5.2: Overview of the command panels

All Crate Control (AllGlobal.pnl): Depicted at position 3 in Figure 5.2 is the all crate control panel from which it is possible to control all crates connected to the system. From a barrel LCS commands sent using this panel would thus possibly affect all 4 SCT barrels. Commands issued this way on an end cap LCS would take affect on both end cap A and C since several crates are shared between the two end caps.

Directional Control to One Crate (Directional.pnl): As seen at position 4 in Figure 5.2 the directional command panel enables to send commands from a given state to the neighboring state in a circular state transition scheme with OFF at the bottom and ON as the highest level. This is implemented for HV and LV independently or together. The panel is useful for recovering modules (channels) that are stuck in the "wrong" state without interfering with any of the "working" modules (channels).

5.2. Display panel [3]

Location: *<projectpath>/panels/subpanels/display_panels*

Functionality: The center part of the HVLV panel is used for displaying the panels referred to as display panels. These panels display all monitored parameters of the system and using mapping through datapoint aliases the panels can present the information to the user not only in terms of hardware units (like the crate view as seen on Figure 5.1), but also mapped to structures like barrel or end cap cooling loops. From the buttons given in the upper half of the right column of the HVLV panel the user can change between the different displaying modes. Below a short description is given of each panel:

Crate View (glmon.pnl): The Crate View as seen on Figure 5.1 is a representation of the physical crate showing both the LV and HV cards of the crate. Each LV card has 4 channels, each HV card has 8 and their LEDs represent the different channels. In addition LV card has one LED at the top which shows if the 4 channels of the card is interlocked (the interlock granularity corresponds to groups of 4 channels, i.e. one LV card). The HV card thus has two interlock LEDs to show if the lower or higher group of 4 channels have been interlocked. The channel LEDs show the present state of the channel. It is however important to note that the green "OK" color does not necessarily correspond to a channel ON state is defined in the configuration. For HV the OK state requires HV > 19V while LV "OK" is fulfilled by Vcc > 3.2V and Vdd > 3.7V.

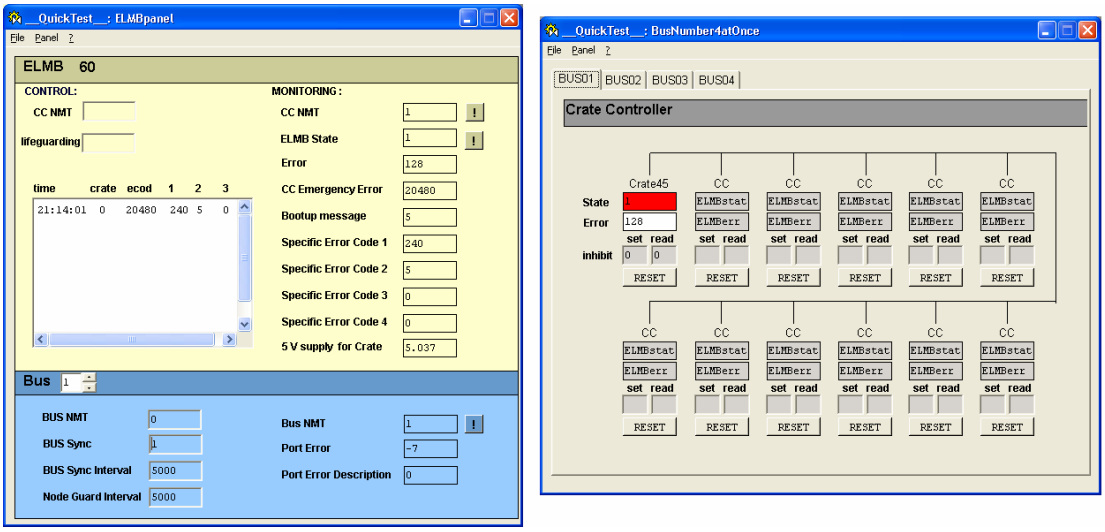

Figure 5.3: ELMB view showed on the left hand side, on the right the BUS view

ELMB panel (ELMB.pnl): The ELMB panel (Figure 5.3, left) allows for monitoring and control of the global CC parameters and values related to the CANbus. The upper, (yellow) part of the panel displays the information for the selected node. The node can be started, stopped or reset by means of the NMT commands (CC NMT), according to CANopen communication layer (See CANopen specification). On the right side of the panel the actual NMT state of the CC is displayed (CC NMT), together with a toggling heart beat (ELMB State), boot-up message counter and the 5V power supply readout. In addition the error codes from emergency messages are shown on the panel. An explanation of the different emergency messages generated by the **CC** application in the event of an error is given in Appendix A:.

BUS view (BUS_four.pnl): On each BUS a maximum of 11 crates can be operated and the BUS view panel (Figure 5.3, right) shows which crates and their corresponding status that is connected to each BUS. From the Bus view panel the user can perform 'software' reset of the selected CC node by clicking 'Reset' button. This action will send the sequence of NMT commands to restart node.

When there is more than 4 CC nodes on the bus, the high rate of readout data transmission causes a 'traffic jam' on the CAN bus. To prevent this situation, each CC transmit has an inhibit time parameter. This is a counter with a 100 ms resolution, which will inhibit the next readout transmission for a period of time. In the 'Inhibit' text box user can specify the inhibit time for each CC.

.

Figure 5.4: To the right the Barrel view panel, to the left the Disk view panel.

Barrel View (BarrelView.pnl): The barrel view panel (Figure 5.4, left) depicts the barrel as seen from the side and shows 4 rows of modules at the same time. In the "Next" field the user can choose which barrel to study and which row to start from. From a drop down menu it is possible to choose any module parameter (HV,LV and temperature), and the "View" switch can be used to rotate the barrel by 180° i.e. to view the barrel with the "A" side to the left or right. The "Now" field shows the present settings and the "Save Me!" button will take a snapshot of the present displayed data and write them to a tab separated text file. In the "Action" field of the barrel view panel it is possible to mask on HV/LV for the displayed modules or ramp HV to a given input value.

Disk View (DiskView.pnl): The disk view panel (Figure 5.4, right) has the same functionality for the disks as the barrel view has for the barrel. From the "Select-O-Matic" one can choose which disk, quadrant and parameter to look at, and the data is displayed in the "Disk-O-Meter". Using the masking buttons modules can be masked on and through the "Ramp HV" button a HV value can be set.

5.3. Setup and Storage [4]

At the time of writing the way data (voltages, currents, etc) is stored is being reimplemented in the project as the final system will use an ORACLE database for storing the DCS conditions. Thus no details are given in the present version of this manual.

Listed under "Recompilation" are several useful buttons for project setup and change of mapping (as discussed above in Section 4), read back CC database and finally a button to start the DIM DNS server needed for DAQ-DCS communication.

5.4. Detector Panels [5]

Location: <*projectpath*>/panels/subpanels/detector_panels

Functionality: The detector panels are meant to give a complete overview of the HV and LV state and status of all modules connected to either a disk or a barrel. The state is retrieved from the crate controller which checks online channel parameters with database entries (see Section 2.1 for more details). Status is derived from each channel's summary alert. To ease maintenance and development, all code for decoding of state and status is gathered in the DetectorPanelFunctions.ctl library which is included by each panel.

Barrel X or Disk Y (barrelX.pnl / diskY.pnl): Clicking on any of the barrels or disks as seen on top of Figure 5.1 would make a pop up appear for this barrel/disk showing all modules mounted on it. The module's LV state is represented by a square and on top of that a circular LED shows the HV state. An explanation of the colour is given to the right. In the event of an alarm the status colour will override the state information to show that there is a problem with either LV or HV. More information about the alarm can be found by a double-click on the LV field of the channel which would give a pop up panel with channel details.

5.5. Message logging [6]

Along the bottom edge of the HVLV panel there is a log field for important messages to the user. The different scripts and panels prints information about errors or results upon completion here and the user can browse the history by clicking the "more" button.

6. Command Handling using Scripts and Libraries

In the Power Supply project all command handling, i.e. all steps from a button is clicked or command string is inputted until requested command has been executed, are handled by scripts and libraries to ensure a stable and safe operation of the system. Scripts are run by a PVSS Control Manager and connect a callback function to one or several datapoints using the *dpConnect* function of PVSS. Once connected, any change of the datapoint(s) would make the callback function run.

The scripts are started upon system (project) boot up by a dedicated control manager which reads the list of scripts to be started from a text file. The text file (DDC-DCS_script.lst) is stored in the project's script folder and must be given as argument to the manager upon startup.

6.1. Scripts

Location: <*projectpath*>/*scripts*

Table 2 gives an overview of the scripts used in the project, what their main functionalities are and which libraries they depend on for command execution. Each script is dpConnected to a parameter datapoint in which the panels (or the DAQ) write the parameters of the command (See Chapter 8 for a list of which datapoint belongs to which script). The parameters are given as a string and the format of the string depends on the script in question. Table 3 shows format and valid input for each script.

For fast state transitions, and general response time of the system, the main scripts change* and hardReset use multithreading where one "command-thread" is started for each crate targeted by the command. Threaded actions are possible for most actions as the larger fraction of time is spent waiting for hardware to respond, and not by the computer's CPU working. The extra load on the computer is thus minimal and the threaded approach enables a system response time that is almost independent of the scope of the command.

Script	Main functionality	Includes Library
ChangeState.ctl	Performs HV & LV state transitions: Predefined (configuration), manual state transitions or power cycle. Can not set manual parameters.	CCListCreator.ctl PsSet2.ctl
ChangeMask.ctl	Masking of channels for both HV & LV together or independently of each other.	CCListCreator.ctl PsMask2.ctl
ChangeParam.ctl	Can write manual settings to CC RAM for chosen variables (some values)	
ChangeClock.ctl	Set clock select line for the module	CCListCreator.ctl Cooling_Enviroment.ctl
hardReset.ctl	Performs hard reset of the channel	CCListCreator.ctl PsSet2.ctl Cooling_Enviroment.ctl
Initialisation.ctl	dpConnects to cooling loop status (cooling project) and automatically shuts down the PS channels of a loop in the event of a cooling failure. Monitors ELMB communication.	PsSet2.ctl sctMaps.ctl
LoadMapping.ctl	Loads the channel mapping, i.e. alias to both PVSS datapoints and CC memory.	CCListCreator.ctl sctMaps.ctl
LoadConfiguration.ctl	Write configurations (OFF, STB or ON) to the CC memory (EEPROM).	CCListCreator.ctl

Table 2: Functionality and companion libraries for the scripts used by the project.

In Figure 6.1 a schematic overview of the changeState script is given where the script is triggered by an input string. First the script retrieves the list of the crates targeted by the transition (either single crate (integer) or if string ‘all’ is given, the list of crates is determined by the channel argument). Following this the script starts one thread per crate which handles the rest of the state transition. The key component of each thread is the PsFsmCommandSender from the PsSet2.ctl library which handles alarms and the execution of the command followed by an acknowledgement from the CC. Note that the alarms are updated upon each state transition and that the appropriate alarms thresholds are applied according to the requested state.

Script	Command crate|channel|state/param|value			
	Crate	Channel	state/param	Value
ChangeState	‘X’ ’all’	‘X’ an integer e.g. 2 or 45 ’X,Y,Z,’ list of integer ’plB304’ PVSS loop number Alias(using wildcards (*) where needed): ’BarrelXXRowYYModule±Z’ ’EndcapXA/C.QuadrantYY.Ring0Z.ModuleXX’ ’BarrelXX.LoopYY’ Row counts from 0, Endcap disk 1–9, Quadrant 0–3, Ring 1– 3 , Loop counts from 1.	‘0’ INITIAL ’1’ ON ’2’ STARTING ’4’ LV_STB ’5’ HV_STB ’6’ LV_ON ’7’ HV_ON ’8’ LV_ON ’9’ HV_OFF ’88’ LV_MAN ’99’ HV_MAN	-
ChangeMask	Same	Same	‘1’ HV&LVON ’2’ HV&LVOFF Hex format: ’OxHL’ Where H & L ’0’ Mask OFF ’1’ Mask ON ’2’ Do nothing	-
ChangeParam	Same	Same	LVchVCSV LVchPINV LVchCLKS HVchVOLT	0 – 6 0 – 10 0 or 1 0 – 400
hardReset	Same	Same	-	-

Table 3: Valid script input arguments. See FSM chapter for more details about the changeState 'states'. Fields left empty do not apply.

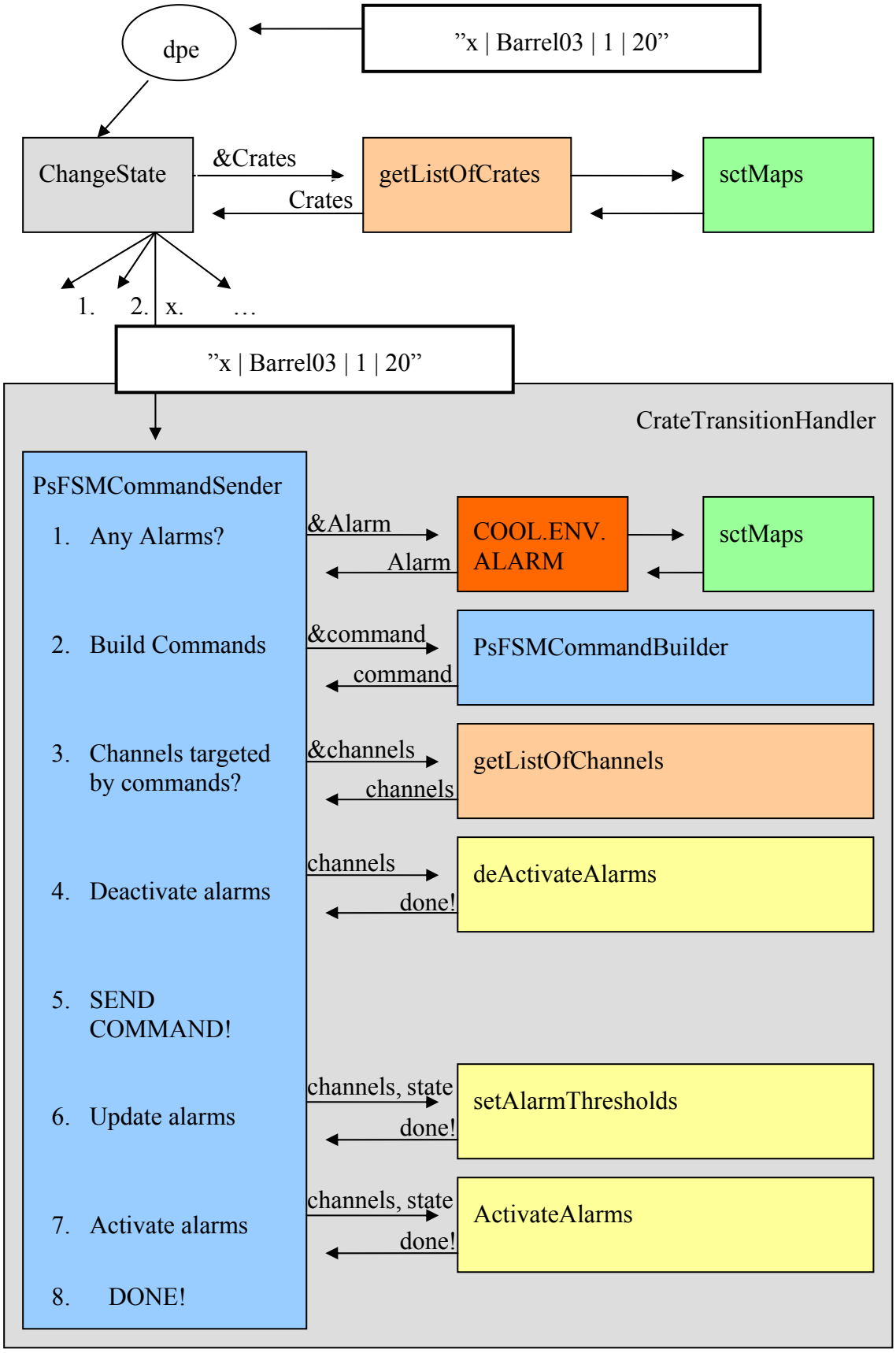

Figure 6.1: Schematic overview of a state transition command.

Scripts run by the SCS

All scripts running mentioned in Table 2 also run on the SCS but in a special mode as there is no PS hardware connected to the SCS. Each of these scripts has a command distribution functionality which detects if the script is run by the SCS or LCS. If ran by the SCS the script will enable command distribution rather than command execution, writing the command argument (parameters) to the LCS(´s) command datapoints, thus causing their scripts to execute the callback function and process the command.

Distribution of commands from the SCS is important functionality for integration of the 8 LCS systems and communication with the DAQ as the DAQ only communicates with the SCS (see also Chapter 8).

6.2. Libraries

Location: <*projectpath*>*/scripts/libs*

The libraries of the project can be divided in two groups. There are *executive libraries* which contain functions for hardware state transitions, and there are *support libraries* which contains support functions for scripts, other libraries or panels. Table 5 shows the libraries used in the project and their main functions.

The PsSet2 library is particularly important as its functions handle all hardware state transitions (see also Figure 6.1). Its implementation of the PsFsmCommandSender function features full control over a channel's (module's) LV and HV supply according to the command protocol as outlined in Section 2.1. The design of the library is tailored to the implementation of the Finite State Machine (FSM), and thus the PsFsmCommandSender works with state strings rather then numbers as the command protocol. While most command/state strings are straight forward to understand some require an explanation and Table 4 shows the corresponding CC command protocol number for different FSM commanded states.

State	HV	LV
FSM commanded states		
'OFF'	4 (MASK OFF)	4 (MASK OFF)
'INITIAL'	0 (OFF)	0 (OFF)
'STARTING'	2 (STB)	2 (STB)
'STANDBY'	1 (STB)	2 (ON)
'ON'	1 (ON)	1 (ON)
'POWERCYCLE'	1021 (ON,OFF, STB,ON)	-
'HARDRESET'	6 (HARDRESET)	-

Table 4: Translation between FSM commanded states and the Crate Controller states.

Executive Library	Main Functionality	Main Functions
PsSet2.ctl	Contains functions for executing state transitions. All state changes in the project (including FSM) are handled by this library.	PsFsmCommandBuilder PsFsmCommandSender ultimateRecovery
PsMask2.ctl	Masks channels on and off. Companion library to the ChangeMask script.	PsMask2
Support Libraries	**Main Functionality**	**Main Functions**
AlarmHandler.ctl	Deactivates Alarms, applies new thresholds and reactivates the alarm during state transitions.	ChannelAlarmSwitcher SetAlarmThresholds
AlarmHelper.ctl	Support library to the AlarmHandler. Generates lists of datapoints and reads alarm thresholds from configuration.	Activate, DeActivate getChannelAlarmList getChannelNewThresholds
CCListCreator.ctl	Crucial support library for most scripts and libraries. Decodes an input string (alias) to a list of crates or channels.	getListOfCrates getListOfChannels getListofChannelInState
ChannelStateAndColor.ctl	Decodes the channel state integer (reported by the CC) into a state string and matching color.	GetState
Cooling_Enviroment.ctl	Checks that no alarms are in progress for the part of the detector targeted by a command.	CoolingEnviromentalAlarms
DetectorPanelFunctions.ctl	Accompanying library to the detector panels. See Section 5.4.	displayPopup UpdateColors
sctMaps.ctl	Contains a hard coded mapping to convert from ELMB to PVSS to COOLING loop numbers.	sctMaps_PVSSLoopFromAlias sctMaps_elmbLoopFromAlias sctMaps_coolLoopFromAlias
SCT_FSM_Generator.ctl	Support library to the FSM generator panel. Builds the FSM structure for a LCS.	FSMcreator_GroupInit FSMcreator_LoopInit

Table 5: Overview of the different libraries in the project, their functionality and most important functions. For a complete list of functions consult the code.

7. System Safety and Distributed Systems

Upon start up the initialisation script will run and dpConnect the PsFsmCommandSender function to the cooling loop datapoints of the cooling project. The initialisation script will start a dpConnect for every cooling loop present on the system. If the cooling datapoint enters an error mode the PsFsmCommandSender will be called with the argument to shut down the modules on the loop in question.

A necessity for this mechanism to work is that the Power Supply project and the Cooling project are connected to each other as distributed systems. This is be default the case but can be changed in the project config file.

8. DAQ – DCS Communication (DDC)

The DAQ – DCS communication is provided via a Distributed Information Management (DIM) system which is enabled through the PVSS-DIM toolkit. Using this toolkit the project publishes the readout data via a DNS server from which the DAQ system can subscribe to the data. The following data is published for each channel (module):

HVchVolt.Recv, HVchCurr.Recv, MOch_Tm0.Recv, MOch_Tm1.Recv, LVch_Vcc.Recv, LVch_Icc.Recv, LVch_Vdd.Recv, LVch_Idd.Recv, LVchPINI.Recv, State

The other way around the project can also retrieve commands sent from the DAQ by commands published to the DNS server. This is a crucial feature since the foreseen mode of operation for the SCT detector once in place inside ATLAS is that the DAQ will be in charge of the power supply. The following commands are published:

ChangeClock, changeMask, ChangeState, changeParam, HardReset, loadMapping, loadConfiguration

When the project is set up for the first time the data has to be published. This can be done manually using the interfaces provided with DIM or there is a tool panel available which will do this automatically. The panel DIMDDC_config.pnl is available in the <projectpath>/panels/subpanels/ToolPanels folder.

Finally also the HVLV Message logging field is also published. Through that the DAQ will receive important information about system changes, like emergency shut downs due to a cooling loop failure.

Appendix A: Crate Controller Emergency Messages

Error Description	CC Emergency Error	Manufacturer-Specific Error Code
CAN communication	8100	Byte 3: 81C91 Interrupt Register content Byte 4: 81C91 Mode/Status Register content Byte 5: error counter
Life Guarding	8130	(CAN-controller has been reinitialized)
RPDO: too few bytes	8210	Byte 3: minimum DLC (Data Length Code) required
Slave processor not responding	5000	Byte 3: 20
CRC error	5000	Byte 3: 30 Byte 4: 1 (Master FLASH), 2 (Slave FLASH)
EEPROM: write error	5000	Byte 3: 41 Byte 4: Parameter block index Byte 5: 0 : writing block info > 0: size of parameter block to write
EEPROM: read error	5000	Byte 3: 42 Byte 4: Parameter block index Byte 5: Error id (1=CRC, 2=length, 4=infoblock)
Irregular reset	5000	Byte 3: F0 Byte 4: microprocessor MCUCSR register contents
No Bootloader	5000	Byte 3: F1
Backplane communication lost	FF00	Byte 3: card number
PS errors/wornings	FF01	Byte 3: channel number (0-47 for LV ; 128+175 for HV) Byte 4: Alarm_Code Byte 5: channel status byte Byte 6: run state (only for LV)
I-V curve error	FF02	Byte 3: IV error code
LV card busy	FF03	Byte 3:40 Byte 4: card number Byte 5: channel number
Channel not set	FF04	Byte 3: card number Byte 4: channel number