Stepper Motor Control

System Design

Revision:	1.3
Status:	RELEASED
Repository:	https://internal.cosylab.com/svn/acc/projects/GSI/
Project:	FAIR-GenericSlits
Folder:	trunk/doc/
File:	DES-FAIR-StepperMotorControl-SystemDesign.odt
Owner:	Gasper Jansa
Last modification:	July 30, 2015
Created:	October 17 2012

Document History

Revision	*Date*	*Author*	*Section*	*Modification*
0.1	10/17/12	gjansa	All	Created.
0.2	11/06/12	gjansa	6.2.2	Added slits class definition.
0.3	11/09/12	gjansa	6	Updated FESA properties.
1.0	08/15/13	gjansa	All	Review and released.
1.1	09/19/13	gjansa	6.2	Updated properties
1.2	11/14/13	gjansa	6.2	Updated properties (ssi encoder resolution description)
1.3	11/17/28	ggaspersic	6.2	Added move to position and named positions property. Updated the configuration property with sampling.

Confidentiality

This document is classified as a **`public document`**. As such, it or parts thereof are openly accessible to anyone listed in the Audience section, either in electronic or in any other form.

Scope

This document describes the design of the stepper motor control software for FAIR.

Audience

All Cosylab and GSI/FAIR members involved in the project and users of the system.

References

[1] M. Levicnik, M-Box with PDC Motion control, Implementation, 4.1
[2] Turbo PMAC/PMAC2, Software Reference Manual, 3Ax-01.937-xSxx May 24, 2004

Table of Contents

1 Overview....4
2 PMAC software....6
2.1 Settings and configuration....6
2.2 Motion programs....6
2.3 PLC programs....6
3 System Driver....8
3.1 Access control....9
4 Local control server and GUI....11
4.1 Local control GUI....12
5 Access control monitor....15
6 FESA device server....16
6.1 General Design....16
6.1.1 Class composition....16
6.1.2 Data Types....16
6.1.3 Data Synchronization and Notifications....16
6.1.4 Error Handling....17
6.1.5 Initialization....17
6.1.6 Access control....17
6.2 Class Properties....17
6.2.1 Motor class....17
6.2.2 Slit class....22
7 Configuration files....30

Table of Figures

Figure 1: The architecture of motion control software....4
Figure 2: The architecture of the system driver....8

OVERVIEW

This design document describes motion control front-end software to be used in FAIR project. In addition to software running on front-end local control GUI application is also included which is used for motor commissioning.

MicroIOC-M-Box-PMAC, which is microIOC- based product and hosts powerful PMAC motor controller, is used as front-end controller [1].

The software architecture is shown on Figure 1.

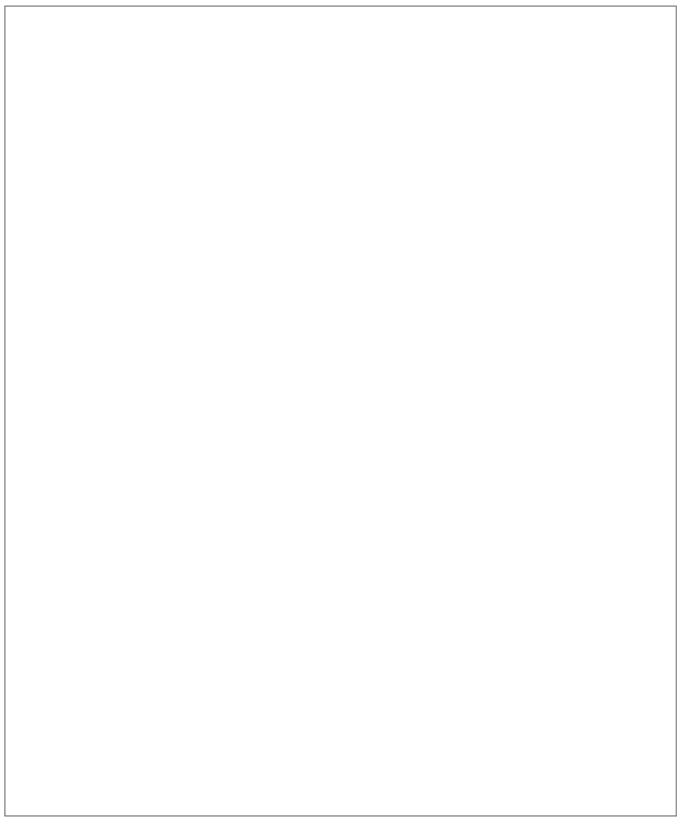

Figure 1: The architecture of motion control software

As shown on the figure above the software comprises of the following components:

- **PMAC software**
  - Low level software used to control single or pair of motors which resides on PMAC controller inside M-Box.
- **System driver**
  - C++ system driver that communicates with PMAC via TCP/IP
  - Communicates with any other additional device (e.g. encoders) via serial connection.

- Transforms requests from upper level software to PMAC commands
- Handles local/remote control connection

- **Local control server**
  - Local control server used for motor configuration
- **Local control GUI**
  - Local control GUI running for motor configuration
- **Access control monitor**
  - Used to implement access control fall-back and shared memory cleanup in case any of the application crashes unexpectedly. In addition to that it is also used to control LEDs on front panel of the M-Box and for software minimum distance check for paired motors. In case minimum distance is violated it stops the motors.
- **LCD control**
  - Used for controlling motors locally from the LCD on the front panel of M-Box.
- **FESA device server**
  - FESA classes which model motor and pair of motors (slit)
- **Configuration files**
  - Files for system configuration

Each of these components is described in the following chapters.

1 PMAC SOFTWARE

PMAC software is divided into:

- settings and configuration
- motion programs
- PLC programs

Short description of the above is described in the following chapters.

1.1 SETTINGS AND CONFIGURATION

These are files with simple commands which get executed only when they are downloaded to PMAC. They will be used to:

- store the permanent motor configuration (e.g. PID parameters, channel addresses, ...)
- default values for motor settings (e.g. velocity, acceleration, movement range...)

1.2 MOTION PROGRAMS

When a motion program is being executed, the PMAC works through the program one move at a time, performing all the calculations up to that move. Motion programs are executed in coordinate systems, which motors are assigned to. There shall be :

- One motion program for each motor (8 motion programs)
- One motion program for each motor pair for slit positioning (4 motion programs)
- One motion program for each motor pair for executing homing procedure (4 motion programs) This depends on the type of feedback axis is using (incremental encoders require homing).

1.3 PLC PROGRAMS

PLC programs are special PMAC programs that operate asynchronously and with rapid repetition. They have the same logical constructs as the motion programs. There shall be:

- One PLC program that performs initialization upon PMAC power up. Used for I/O initialization, internal variables initialization, enabling all the motors, applying brake on the motors (if applicable)...
- One PLC for each motor that is executed upon specific event (command from upper layers of software e.g. FESA device server). They are used to stop the motion and apply brake (if applicable) and/or kill the motor.
- PLC that monitors the execution of motion programs and performs certain actions like changing the assignment of the motors to the correct coordinate systems.

2 SYSTEM DRIVER

System driver offers routines to operate single motor or pair of 2 motors. In addition to that it provides routines for system inspection (e.g. PMAC status, motor status,...) and routines to operate any other device used in the system (e.g potentiometers, SSI encoders).

The architecture of the driver is depicted on the figure below.

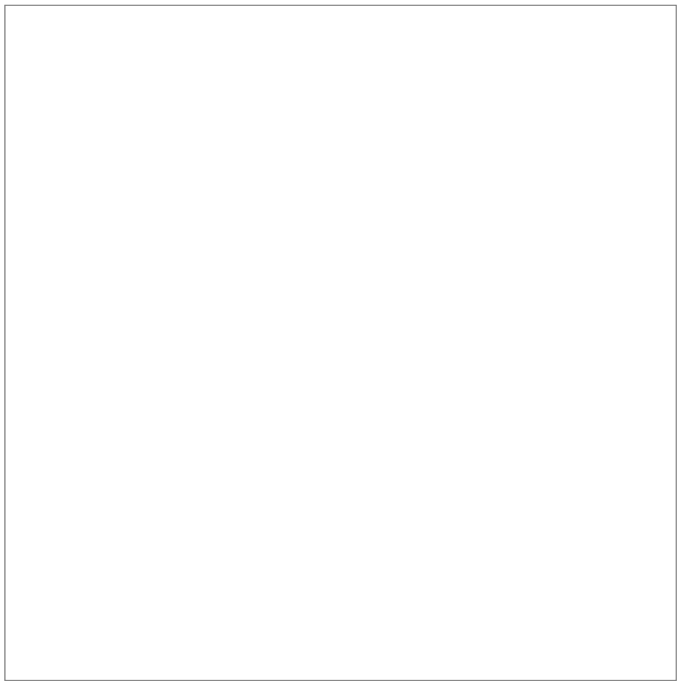

Figure 2: The architecture of the system driver

The architecture of the driver comprises:

- local/remote/lcd control logic controls access rights of the local and remote control. See Access controlfor details.
- Single motor monitor and control
- Control and monitor of pair of motors (slit)
- Readout of the encoders (potentiometer or SSI encoder)

Driver is written in C++ programming language. All API routines are synchronous meaning that they block until completed. This does not mean that the routine blocks until the motor operation is completed, e.g. moving of the motor, it means the routine will block until the execution of move command is completed. All API routines throw exception on error with message describing the problem. Driver uses 2 configuration files. For more on configuration see Configuration files.

2.1 ACCESS CONTROL

Since more than one process is used to control motors on M-Box access control has to be implemented. The job of the access control is to:

- prevent concurrent access
- grant access rights for certain operation depending on the nature of the caller

The following processes will use be used to manipulate motors on M-Box:

- FESA classes
- Local control server
- M-Box front panel LCD application

Access control is implemented using shared memory. A single object is created in shared memory which holds the following information:

- current mode which can be one of:
  - Remote control
    - Used by FESA class to move motors.
    - Motor configuration is not possible from FESA class.
    - Local control and LCD application have read only access.
  - Local control
    - Used by local control to move motors
    - Motor configuration from local control is not possible
    - FESA class and LCD application have read only access
    - Falls back to remote control after certain time of inactivity
  - Local configuration
    - Used by local control to modify configuration and move motors
    - LCD application and FESA class have read only access
    - No fall back
    - Local control has to explicitly switch from local configuration

    - LCD control
        - Used by LCD application to move motors
        - Motor configuration from LCD application is not possible
        - Local control and FESA class have read only access
        - Falls back to remote control after certain time of inactivity
- state of the semaphore used to prevent concurrent access to the system driver shared resources
- locked counter, used to clean up shared memory in case 'user' of the shared memory object crashed. See Access control monitor for details.
- Heart beat counter, used to implement fallback from local control and LCD control mode to remote mode (used if local control mode operator forgets to change the mode or LCD user forgets to exit the application). Local configuration mode does not have a fallback mode as this can be potentiality dangerous if the configuration of the motors was changed but not verified.
- motor and motor pair configuration
- motor and motor pair readback values are cached in shared memory (e.g. status, position readback)

3 LOCAL CONTROL SERVER AND GUI

Maintenance persons need separate maintenance access. The access is needed to:

- Configure motor parameters
- Have access to all diagnostic data
- Operate individual motors
- Operate pair of motors

Maintenance access to devices installed in the accelerator will be needed:

- In case of trouble shooting (problems in accelerator operation)
- In case of reconfiguring in shutdown periods

Maintenance access is independent of control system infrastructure (control system middleware, central data bases, …) and is possible via direct network connection to M-Box.

Local control consists of two parts as shown on figure below:

- Local control server
  - C++ Ethernet socket server implementation which uses system driver to manipulate motors.
- Local control GUI
  - Java GUI which can be run from remote computer. It communicates with local control server via Ethernet. Described in details below.

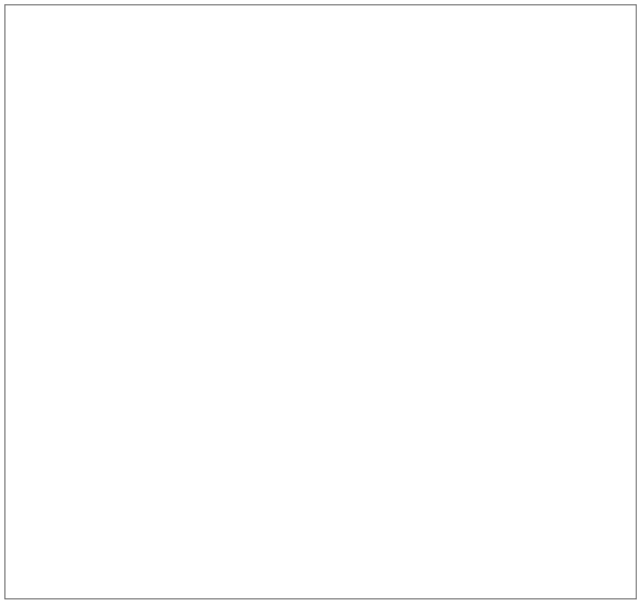

Figure 1: Layout of the local control

3.1 LOCAL CONTROL GUI

Local control GUI is basically divided into two main screens:

- Motor and motor pair configuration screen
- Motor and motor pair control screen

Motor and motor pair configuration screen can be seen on the figure below.

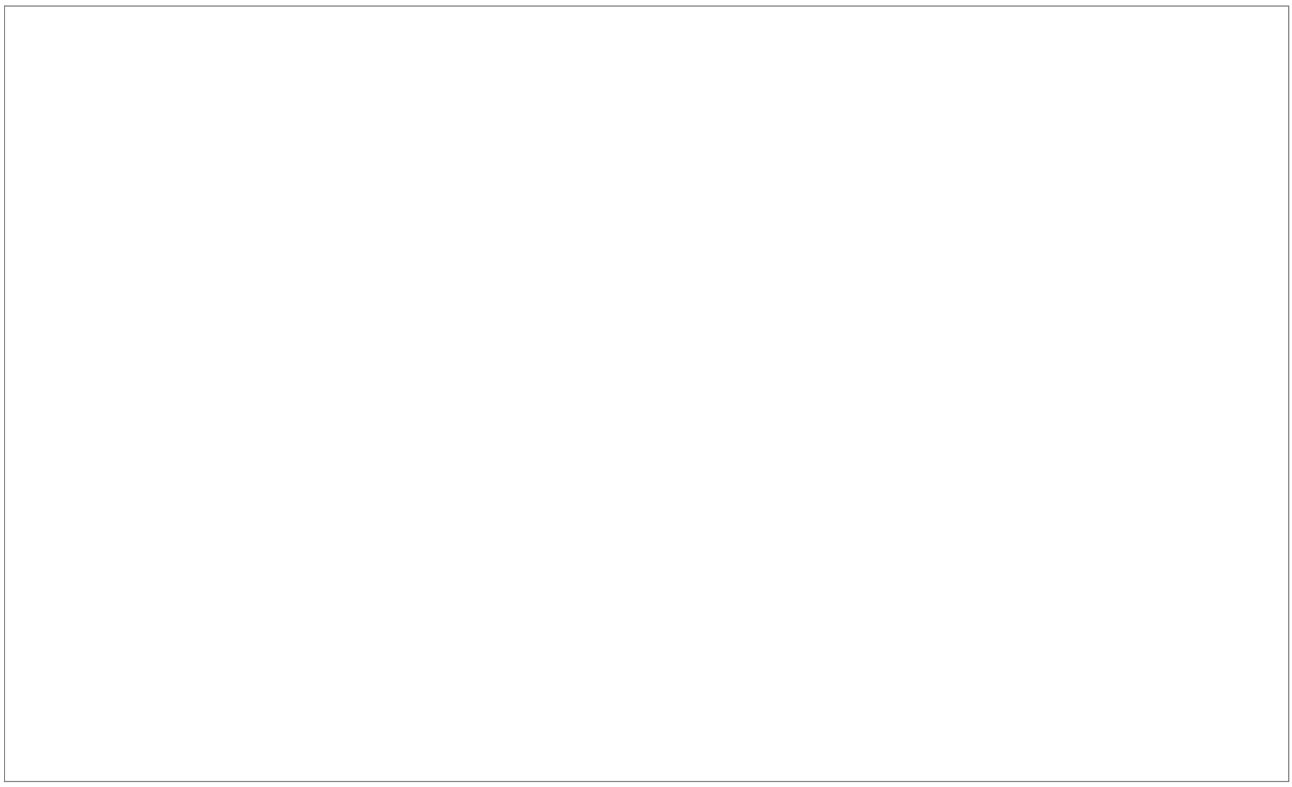

Figure 6: Motor and motor pair configuration screen

Motor and motor pair configuration screen offers modification of the following parameters for each motor :

- nomenclature of the motor
- mounting orientation
- part of motor pair or not
- horizontal or vertical installation
- encoder type selection
- middle switch enabled or disabled

- limit switch enabled/disabled
- set driver direction CW/CCW (linked to PMAC variable I7mn8, see [2])
- driving frequency (velocity)
- pulse width (linked to PMAC variables I7m03 and I7m04, see [2])
- pulse polarity (linked to PMAC variable I7mn7, see [2])
- acceleration time
- engineering unit to steps conversion (EGU/encoder steps)
- offset
- positive/negative limits
- position factor
- minimum spacing if middle switch disabled

Configuration screen also offers properties to be saved/load to/from files on the PC where local control GUI is running or directly to/from M-Box.

Motor and motor pair control screen is shown on the figure below:

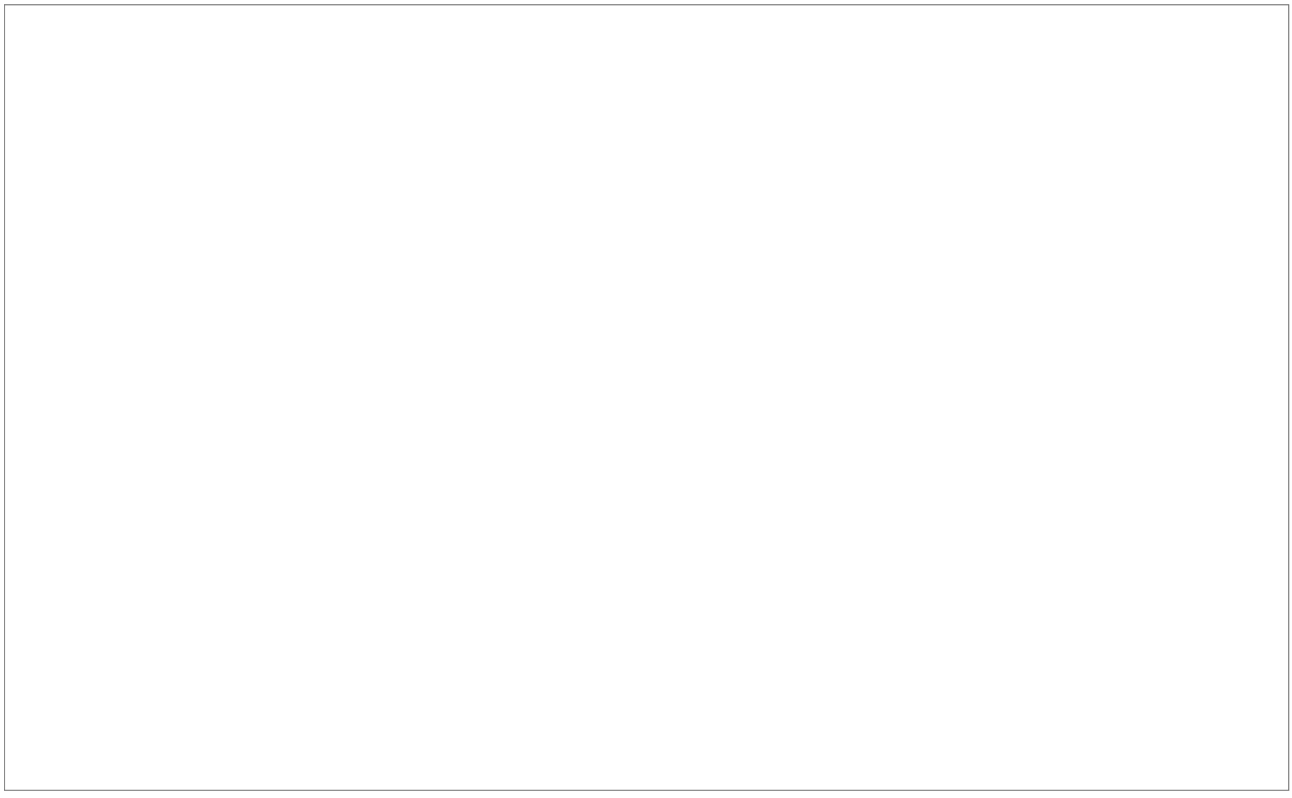

Figure 10: Motor and motor pair control screen

As shown on the figure above motor and motor pair screen is divided into two sections. Upper side of the screen comprises:

- Motor and motor pair status
- General status
- Graphical presentation of motor positions

Lower part comprises motor and motor pair controls.

4 ACCESS CONTROL MONITOR

Access control monitor is an application which resides on M-Box and monitors the state of the shared memory used for access control. It performs the following tasks:

- monitors the lock counter of shared object. If object is locked for 10 seconds and the counter does not increase in this time (which would indicate that process locking the object crashed) then object is unlocked so that it can be used by other processes.
- Monitors the heart beat counter of local control or LCD control mode. When local control (or LCD control) is active it will increase this counter. If the counter will not increase for a certain time the monitor application will change the mode to remote control. The time-out for this callback to occur is configurable via configuration file.

In addition to controlling access to the resources access control monitor is used also for the following two tasks:

- controlling the LEDs on the front end of the M-Box
- software minimum distance monitoring for pair of motors which don't have hardware middle switch

Code base for this application is part of the system driver described in chapter 3.

5 FESA DEVICE SERVER

FESA class mirrors the functionality supported in the system driver i.e. FESA class offers control of single or pair of motors.

GENERAL DESIGN

Class composition

Two FESA classes are provided. A FESA class to manipulate a single motor and a FESA class to manipulate pair of motors. FESA deployment mechanism is used to deploy them as single deployment unit.

5.1.1 Data Types

doubleing point values are of double type. Step unit values are of 32-bit integer type. Information property items are numeric for numerical items and strings for the rest.

5.1.2 Data Synchronization and Notifications

Since the system driver does not support callbacks for notifications of data change, and FESA properties interface assumes that properties can be monitored for changes, the notification mechanism is implemented in the FESA class.

Notification mechanism uses polling of the data on the system driver at regular intervals. It is assumed that FESA does not handle very frequent updates well, so the polling is expected to be done at a 1s interval. Due to the low update rate, there is no distinction between properties and all are updated at this same interval.

To use an already existing FESA functionality, a custom software event generator is included in the RT part of the FESA class design to perform the notifications. On the event, acquisition data is read from the system driver and relevant properties are notified. The acquisition data that is updated in such way is:

- motor position
- all status information
- reference voltage
- raw potentiometer voltage
- raw SSI encoder readback

The following properties are notified:

- Status

- Acquisition
- Diagnostics
- Configuration

On the client property get request, the acquisition values are filled from acquisition fields whereas configuration values are read directly from the system driver. This is done with the assumption that these reads the system driver are not a bottleneck.

On the client property set request, the value is set on the system driver regardless if the value is the same as old value, and no checking with re-reading is performed. This assumes that the system driver throws an exception in every situation in which the value could not be set exactly.

5.1.3 Error Handling

FESA class checks the error state and catches exceptions generated by the system driver, and re-throws them as FESA exceptions.

5.1.4 Initialization

During FESA class initialization, the system driver is first initialized from configuration files and then property values are set from the driver. For motor parameters initialization FESA instantiation XML files are not used. These files are used only for FESA specific initialization.

5.1.5 Access control

As described in chapter Access controlaccess control is implemented on the system driver level. When system is in use by local control FESA class will be denied any manipulation of the motors (e.g. change of setpoints, moving the motors etc.), only reading from the system driver is possible.

5.2 CLASS PROPERTIES

5.2.1 Motor class

Property name	***Property type***	***Value Items name***	***Value Item type***	***unit***	***description***
Status	status				
		mode	GSI-mode-field		Always on
		control	GSI-control-field	/	REMOTE/LOCAL_ CONTROL/ LOCAL_CONFIG

					URATION/LCD_CONTROL
		status	GSI-status-field	/	OK/ERROR
		detailed-status	GSI-detailed-status-field	/	0 – inner hw end limit set 1 – outer hw end limit set 2 – inner sw end limit set 3 – outer sw end limit set 4 – moving 5 – break 6 – fatal following error 7 - amplifier fault 8 - overheat 9 – axis interlock 10 - potentiometer reference error 11- middle switch (only used if motor is paired) 12 – position tolerance
		pmacMotorStatus1	int32	/	Pmac motor status 1. See [1]
		pmacMotorStatus2	int32	/	Pmac motor status 2. See [1]
		pmacStatus1	int32	/	Pmac general status 1. See [1]
		pmacStatus2	int32	/	Pmac general status 2. See [1]
		pmacStatus3	int32	/	Pmac general status 3. See [1]
Power	power				Always on. Returns error if try to switch it off.
Init	init				Initialize (from XML and configuration files)

					Motor is not moved. Set position and readback are consistent.
Reset	reset				Same as init.
Version	version				
		driver_version	string	/	Version of driver
		pmac_version	string	/	Version of pmac software
		cpld_version	string	/	Version of CPLD firmware
Setting	setting				
		position	double	meter	Absolute position setpoint
Acquisition	acquisition				
		position	double	meter	Absolute position readback
		position_status	AQN_STATUS	/	Set to DIFFERENT_FROM_SETTING when deviation from set value greater position_tolAbs And to BUSY when motor is moving.
		position_tolAbs	double	meter	Absolute tolerance of deviation. This is the same as position tolerance parameter.
		positionSet	double	meter	Absolute position setpoint
		resolution	double	meter	Resolution precision of readback
		motorStatus	bitset	/	0 – inner hw end limit set 1 – outer hw end limit set 2 – inner sw end limit set

					3 – outer sw end limit set 4 – moving 5 – break 6 – fatal following error 7 - amplifier fault 8 - overheat 9 – axis interlock 10 - potentiometer reference error 11- middle switch (only used if motor is paired) 12 – position tolerance
PositionRelative	setting	positionVariation	double	meter	Move relative to current value.
MoveSteps	setting	steps	long	/	Stepwise relative movement
ToEndPosition	command	endPosition	long	0/1	move to inner/outer end position (0: out, 1: in)
StopMotor	command	/	/	/	/
Configuration	acquisition				
		encoderType	string	potentiometer ssi	
		mounting	string	left/down right/up	
		installation	string	vertical horizontal	
		partOfPair	string	yes no	
		settingResolution	double	meter	minimum possible movement (1 step)
		maxPosition	double	meter	
		minPosition	double	meter	
		offset	double	meter	

		positionFactor	double	/	
		driveDirection	string	clockwise anticlockwise	
		pulseWidth	double	seconds	
		pulsePolarity	string	positive negative	
		accelerationTime	double	seconds	
		limitsEnabled	string	yes no	
		potentiometerLength	double	meter	
		velocity	double	meter/s	
		ssiTurnResolution	long	counts	Number of counts per one encoder turn.
		ssiResolution	double	meter	Resolution of ssi encoder (mm/revolution)
		referenceVoltageTolerance	double	Volt	Maximum allowed difference for referenceVoltage
		positionTolerance	double	meter	Positon tolerance
		positionAverage	long	1-30	Number of samples for position averaging.
Diagnostics	acquisition				
		referenceVoltage	double	Volt	Reference voltage
		positionFactor	double	/	
		offset	double	meter	
		positionVoltage	double	Volt	Raw position value received from potentiometer
		positionSSI	long	counts	Raw position value received from potentiometer
NamedPositions	acquisition				
		names	array of strings	/	Array of names of

					the positions.
		positions	array of doubles	meter	Array of positions that correspond to a name.
		nomenclatures	array of booleans	/	Array of is nomenclature values.
MoveToPosition	setting				
		positionName	string	/	Move to position with specified name.

5.2.2 Slit class

Property name	***Property type***	***Field name***	***Field type***	***unit***	***description***
Status	status				
		mode-item	GSI-mode-field		Always on
		control-item	GSI-control-field	/	REMOTE/LOCAL_CONTROL/ LOCAL_CONFIGURATION/LCD_CONTROL
		status-item	GSI-status-field	/	OK/ERROR
		detailed-status-field	GSI-detailed-status-field	/	0 – m1 inner hw end limit set 1 – m1 outer hw end limit set 2 – m1 inner sw

					end limit set 3 – m1 outer sw end limit set 4 – m1 moving 5 – m1 break 6 – m1 fatal following error 7 - m1 amplifier fault 8 - m1 overheat 9 – m1 axis interlock 10 - m1 potentiometer reference error 11- m1 middle switch (only used if motor is paired) 12 – m1 position tolerance 13 – m2 inner hw end limit set 14 – m2 outer hw end limit set 15 – m2 inner sw end limit set 16 – m2 outer sw end limit set 17 – m2 moving 18 – m2 break 19 – m2 fatal following error 20 - m2 amplifier fault 21 - m2 overheat 22 – m2 axis interlock 23 - m2 potentiometer reference error 24- m2 middle switch (only used if motor is paired) 25 – m2 position tolerance 26 – pair amplifier fault

					27 - pair moving 28- pair following error 29 – pair middle switch status
		pmacMotor1Status1	int32	/	Pmac motor status 1. See [1]
		pmacMotor1Status2	int32	/	Pmac motor status 2. See [1]
		pmacMotor2Status1	int32	/	Pmac motor status 1. See [1]
		pmacMotor2Status2	int32	/	Pmac motor status 2. See [1]
		pmacSlitStatus1	int32	/	Motor pair general status 1. See [1]
		pmacSlitStatus2	int32	/	Motor pair general status 2. See [1]
		pmacSlitStatus3	int32	/	Motor pair general status 3. See [1]
		pmacStatus1	int32	/	Pmac general status 1. See [1]
		pmacStatus2	int32	/	Pmac general status 2. See [1]
		pmacStatus3	int32	/	Pmac general status 3. See [1]
Power	power				Always on. Error if try to switch it off.
Init	init				Initialize (from XML and configuration files) Motor is not moved. Set position and readback are consistent.
Reset	reset				Same as init.
Version	version				

		driver_version	string	/	Version of driver
		pmac_version	string	/	Version of pmac software
		cpld_version	string	/	Version of CPLD firmware
Setting	setting				
		center	double	meter	Slit center
		width	double	meter	Slit width
Acquisition	acquisition				
		center	double	meter	Actual center
		width	double	meter	Actual width
		center_status	AQN_STATUS	/	Set to DIFFERENT_FROM_SETTING when deviation from set value greater position_tolAbs
		center_tolAbs	double	meter	Absolute tolerance of deviation
		width_status	AQN_STATUS	/	Set to DIFFERENT_FROM_SETTING when deviation from set value of one of the motors is greater than position_tolAbs and to BUSY when at least one of motors is moving.
		width_tolAbs	double	meter	Absolute tolerance of deviation
		centerSet	double	meter	Center setpoint
		widthSet	double	meter	Width setpoint
		resolution1	double	meter	Resolution precision of readback
		resolution2	double	meter	Resolution precision of readback

		motor1Status	bitset	/	0 – inner hw end limit set 1 – outer hw end limit set 2 – inner sw end limit set 3 – outer sw end limit set 4 – moving 5 - break 6 – fatal following error 7 - amplifier fault 8 - overheat 9 – axis interlock 10 - potentiometer reference error 11- middle switch
		motor2Status	bitset	/	0 – inner hw end limit set 1 – outer hw end limit set 2 – inner sw end limit set 3 – outer sw end limit set 4 – moving 5 - break 6 – fatal following error 7 - amplifier fault 8 - overheat 9 – axis interlock 10 - potentiometer reference error 11- middle switch
		slitsStatus	bitSet	/	0 – amplifier fault 1 - moving 2- following error 3– middle switch status
CenterRelative	setting	centerVariation	double	meter	Move center

					relative to current set value
WidthRelative	setting	widthVariation	double	meter	Move width relative to current set value
ToEndPosition	command	endPosition	long	0/1	move to inner/outer end position (0: out, 1: in)
StopMotor	command	/	/	/	Stops both motors
Configuration	read-only				
		encoderType1	string	potentiometer ssi	
		mounting1	string	left/down right/up	
		installation1	string	vertical horizontal	
		partOfPair1	string	yes no	
		settingResolution1	double	meter	minimum possible movement (1 step)
		maxPosition1	double	meter	
		minPosition1	double	meter	
		offset1	double	meter	
		positionFactor1	double	/	
		driveDirection1	string	clockwise anticlockwise	
		pulseWidth1	double	seconds	
		pulsePolarity1	string	positive negative	
		accelerationTime1	double	seconds	
		limitsEnabled1	string	yes no	
		potentiometerLength1	double	meter	
		velocity1	double	meter/s	

		ssiTurnResolution1	long	counts	Number of counts per one encoder turn.
		ssiResolution1	double	meter	Resolution of ssi encoder (mm/revolution)
		referenceVoltageTolerance1	double	Volt	Maximum allowed difference for referenceVoltage
		positionTolerance1	double	meter	Positon tolerance
		positionAverage1	long	1-30	Number of samples for position averaging.
		encoderType2	string	potentiometer ssi	
		mounting2	string	left/down right/up	
		installation2	string	vertical horizontal	
		partOfPair2	string	yes no	
		settingResolution2	double	meter	minimum possible movement (1 step)
		maxPosition2	double	meter	
		minPosition2	double	meter	
		offset2	double	meter	
		positionFactor2	double	/	
		driveDirection2	string	clockwise anticlockwise	
		pulseWidth2	double	seconds	
		pulsePolarity2	string	positive negative	
		accelerationTime2	double	seconds	
		limitsEnabled2	string	yes no	

		potentiometerLength2	double	meter	
		velocity2	double	meter/s	
		ssiTurnResolution2	long	counts	Number of counts per one encoder turn.
		ssiResolution2	double	meter	Resolution of ssi encoder (mm/revolution)
		referenceVoltageTolerance2	double	Volt	Maximum allowed difference for referenceVoltage
		positionTolerance2	double	meter	Position tolerance
		positionAverage2	long	1-30	Number of samples for position averaging.
		middleSwitch	string	enabled/disabled	If disabled then sw check is enabled
		minimumSpacing	double	meter	Minimum distance if sw middle switch enabled
		middleSwitchPolarity	string	positive negative	Polarity for hw middle switch.
Diagnostics	acquisition				
		referenceVoltage1	double	Volt	Reference voltage
		positionFactor1	double	/	
		offset1	double	meter	
		positionSSI1	long	counts	Raw position value received from potentiometer
		positionVoltage1	double	Volt	Raw position value received from potentiometer
		referenceVoltage2	double	Volt	Reference voltage
		positionVoltage2	double	Volt	Raw position value received from potentiometer

		positionSSI2	long	counts	Raw position value received from potentiometer
		positionFactor2	double	/	
		offset2	double	meter	
NamedPositions	acquisition				
		names1	array of strings	/	Array of names of the positions.
		positions1	array of doubles	meter	Array of positions that correspond to a name.
		nomenclatures1	array of booleans	/	Array of is nomenclature values.
		names2	array of strings	/	Array of names of the positions.
		positions2	array of doubles	meter	Array of positions that correspond to a name.
		nomenclatures2	array of booleans	/	Array of is nomenclature values.

6 CONFIGURATION FILES

Systems uses several configuration files during initialization and for persistence. The following configuration files are used in the system:

- System data file
  - This file contains system configuration (e.g. number and type of the motors and motor pairs). It is used to generate PMAC software (e.g. genericsSlits.pmac) and is used by system driver to create internal structures (e.g. number of motors in the system). Here number of motors does not mean number of connected motors but number of motors supported by the motion controller i.e. 8. This file should not be changed by the users.
- PMAC properties file
  - This file contains PMAC configuration setup and commands that needs to be executed to drive single motor or pair of motors, to stop single motor or pair of motors etc. This file needs to be changed only if basic system reconfiguration is done e.g. reconfiguration of motor channels on the PMAC level. It is used by the system driver.
- Motor properties file
  - This file contains all motor settings that can be changed by the user (e.g. drive direction, default velocity, default acceleration, …) using local control GUI. This file is used by system driver during initialization or on user request via local control GUI.
- FESA device server instantiation file
  - This is standard FESA instantiation XML configuration file. It includes all initial values for parameters that are of specific use by FESA device server only.
- Startup files
  - These files are used for startup options (e.g. location of log files etc) for the local control server, LCD control and access control monitor.