LEIBNIZ INSTITUT FÜR ASTROPHYSIK POTSDAM

KIEPENHEUER-INSTITUT FÜR SONNENPHYSIK

MAX PLANCK INSTITUT FÜR SONNENSYSTEM-FORSCHUNG

GREGOR:
MOTORIZATION EMULATOR

Document No.: **GRE-KIS-MAN-0021**

Version: **1**

Date: **2022-02-02**

Signatures & Approval

	Name	Signature	Date
Prepared by	Olivier Grassin - KIS Software engineer		2022-02-02
Reviewed by	Lucia Kleint - Univ. of Bern Professor		2023-02-19
Approved by			
Released by			

Change Log				
Vers.	Date	Author	Description of Changes	Sect./Para.
1	2022-02-02	O. Grassin	New Document	all

Table of Contents

List of Figures

1 Scope

The development of GREGOR interfaces that require the use of the small motorization of the telescope, has to be done without the real motorization, to avoid a loss of optical configurations, disturbing observations and without telescope using time. This document describes an interface that emulates this motorization.

2 Reference and applicable documents

SMS	- Small Motorization System
CoSM	- Communication of Short Messages
DCP	- Device Communication Protocol
MEI	- Motorization Emulator Interface
OS	- Operating system

3 Overview

At the GREGOR telescope, the Small Motorization System is the implementation of motorization devices, cabling, networks, computers and software for the telescope mechatronic capability.
The SMS allows a user to move optical and mechanical devices, such as cameras, mirrors, filter wheels, lenses, beam-splitters, hexapod, etc., to configure and tune the telescope state for the observations.

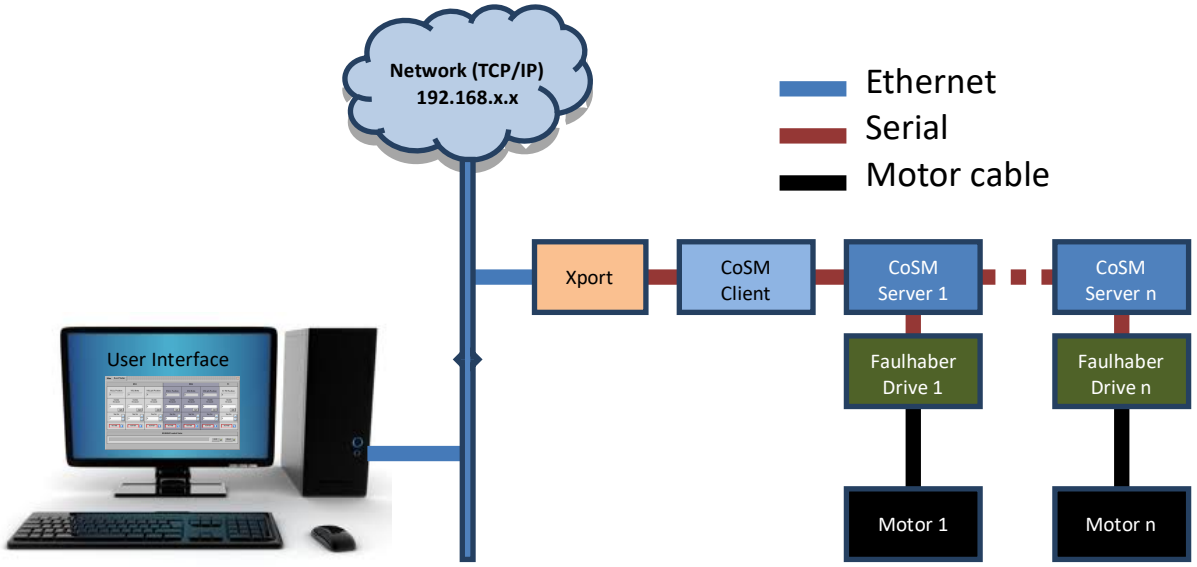

Figure 1 – SMS network overview

The purpose of the MEI is to emulate, as realistically as possible, the telescope small motorization physical environment. For example, an interface in a development state can then be tested, using this application working as a virtual motorization system.

4 Motorization Emulator Interface description

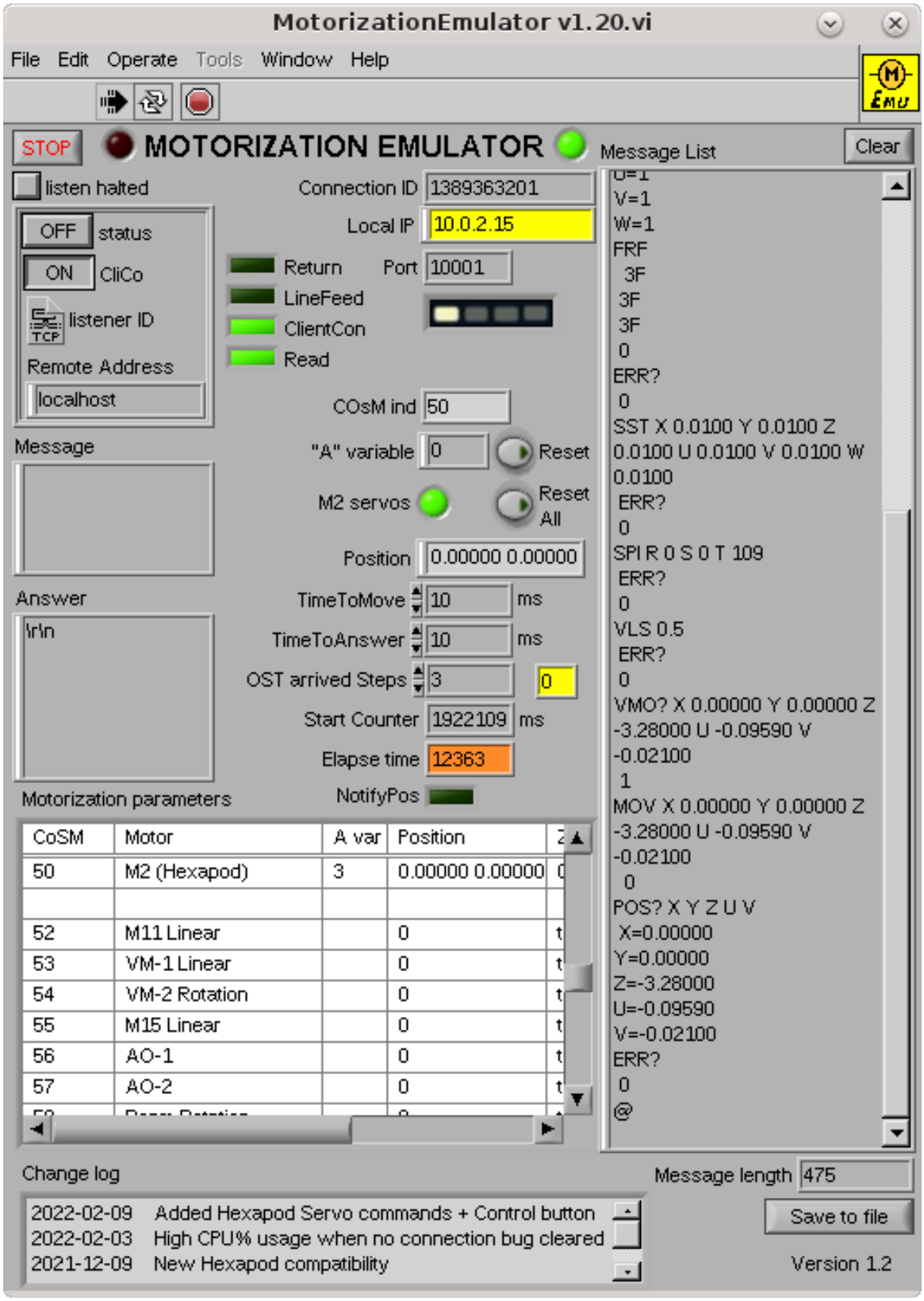

Figure 2 - Motorization Emulator front panel

The interface implementation started in 2013 due to the need for a simulated testing environment. It became a great tool for debugging, viewing and verifying the communications of the motorization system. Its evolution goes hand in hand with the real motorization system evolution and the functionalities of new devices.

The next sections describe the interface elements and functions.

4.1 Interface behavior

In the development environment, the interface, which wants to communicate with a specific motorization has to be configured for the network virtual Xport address to “localhost”, instead of the “192.168.90.xxx” address specific to the motorization network line of the telescope.

The TCP/IP port has to be assigned to the usual port “10001”.

At the start, the interface reads its configuration file to setup the motorization parameters for each emulated motorization. It then enters a mode “listen to a connection” from a client interface.

The design of the interface can handle only one client that establishes the communication, at a time. In general, there is no need to test multiple motorization interfaces at the same time, as the processing can be separated according to the specific interface development.

It should also be possible to use a new instance of the MEI with a different port number, that could emulate a new Xport node, where each MEI instance is configured with a dedicated set of motors. (Not tested, but it needs only some trivial software modifications, as setting the configuration file path value to a relative path of the MEI executable location…)

This is not limiting the use of other interfaces like Conductor that works with the tested interface at the DCP level, and where the motorization can be simulated correctly by the MEI, for a complete testing operation.

At this point, the client interface can connect to the MEI, now ready to enter the receive loop for commands from the client to a specific motor, reading the commands and simulating the CoSM/Motorization answers.

4.2 Network parameters

The main panel offers the view of the main communication parameters as Communication ID (LabVIEW specific), The local IP address/port of the connection, the remote client IP, the status of the connection (ClientCon), the detected “End of line” command character (Return, LineFeed), and the “Read” detection.

Remark: The connection ID changes for each new client connection. The former connection is automatically closed when a new one opens.

4.3 CoSM

Following the CoSM protocol, the client has to send a routing command, as each motor has its proper CoSM server module, to establish the connection with a motor driver. The CoSM attribute has a "#" character + a number (ex: #50). The last activated CoSM number is shown as the "CoSM ind" (index) indicator of the MEI.

The "Motorization parameters" table shows the list of the available motors. The columns are: the CoSM address, the motor name, the "A variable" (*), the position(s) value(s), the requested character (to be sent back) for homing detection, and the end of line character(s).

For ease of use, the parameters of the array can be edited manually, with the corresponding risks assumed by the user. In case of issues, the interface can always be restarted…

4.4 Control parameters

The reaction of the emulator to the commands, can be controlled by parameters assigned in the front panel:

- [Time to Move], is the time of the motorization to simulate the movement execution. Because of synchronous commands, this time should not be greater than the client request time-out loop, to avoid the motor loosing reference status at client side.

- [Time to Answer], is the simulated time to process a command.

- [OST arrived Steps], is the number of steps to receive the OST Faulhaber driver command (Asking for the end of movement execution driver status) used in case of parallelized motorization commands, to finalize an execution move. This option simulates a motor moving time on the way to its destination position.

- (*) "A" variable [Reset] button. The "A" variable, mainly used with the Faulhaber drivers, indicates the state of the motor referencing. When a motor is referenced, its driver "A" variable is stored to the value "70", inside the driver after a real referencing; a trick to tell the client interface if it is needed or not to execute a real referencing of the motor. In case of power down, the real driver reset this variable, so the emulator parameter can be reset to "0", to tell the client that a real referencing is needed before moving the motor. Event more, if the "A" variable is assigned to the "err" string, in the Motorization Parameters table for a specific motor, it will emulate a CoSM server or a motor driver failure, if the corresponding CoSM address is accessed.

- [Reset all] button performs a reset of all the "A" variables of the table, letting all the simulate driver as "Not referenced".

  These options are useful to test the client interface reaction for a specific motor.

4.5 Other indicators and buttons

- "Message" is the received message from the client that will be treated by the MEI.
- "Answer" is the MEI reply to the incoming message.
- "Message list" is a registry of the message/answer dialogue. The user can scroll up and down to view the history of dialogs.
- [Save to file] button: The whole "Message list" text can be exported to a file. When clicked, a message requester pop-up window "write to file", asks for the file name and location to save it.
- "Message length" indicator shows the number of characters in the message list.
- "Clear" button cleans the message list indicator; all communications are lost.
- "Position" indicates the last position value received. It is stored in the Motorization parameters table, for a further use, like the asked position to send back. If "Err" is manually entered in the "Position" cell of the device, the interface will generator a driver communication error. This behavior will help to determine how the client manages a driver communication error.
- "M2 Servos" is a control button that simulates the state of the Hexapod servos. Normal state is ON. It can be switch OFF to test an interface behavior.
- A small scrolling text area shows at the bottom the version revisions.

5 Software

The MEI is written in LabVIEW 15-64bits for Linux, and works in a CentOS 7 environment. Other Linux distributions could also work, having installed the free LabVIEW 15 runtime engine.

An executable version, compiled with the LabVIEW15 project manager, can be used in other simulated environments.

It automatically starts in the listen mode at launch, waiting for incoming instructions.

The "Loop step" indicator  shows a yellow LED starting to move step by step each second of time, when an interface connects to its local port (10001). For each new command received, it will move instantly a new step.

The "Motorization parameters" table will automatically show the last CoSM/Motor accessed in its first line.

5.1 Configuration file

The path of the configuration file in the OS development environment is currently:

/opt/G_Control/MotorizationEmulator/Config/MotorEmu-config.ini

The file is composed of a list of motorization parameters, starting with the CoSM number followed by the motor name, the "A variable", the position, the returned "On position" character and the "Home" character.

As example, here down are shown the 2 first motors of the list:

[CoSM 10]

Name="BBI Wheel 1"

A var=

Position=0

Zchar="p\r\n"

GOHOchar="p\r\n"

[CoSM 11]

Name="M5 axis 1"

A var=

Position=0

Zchar="t\r\n"

GOHOchar="t\r\n"

…

5.2 Known issues

Closing interfaces: The user has to close first the remote client interface properly, using its "stop" control button, which shuts down the connections properly. Subsequently, one can close the MEI the same way, with the red "STOP" upper/left button.

In case of forcing with the  close button, the connection remains active during a time-out delay, given by the OS network internal behavior. The user has to wait around 30sec, to be able to restart the MEI correctly.

This is an example of the connection error display message:

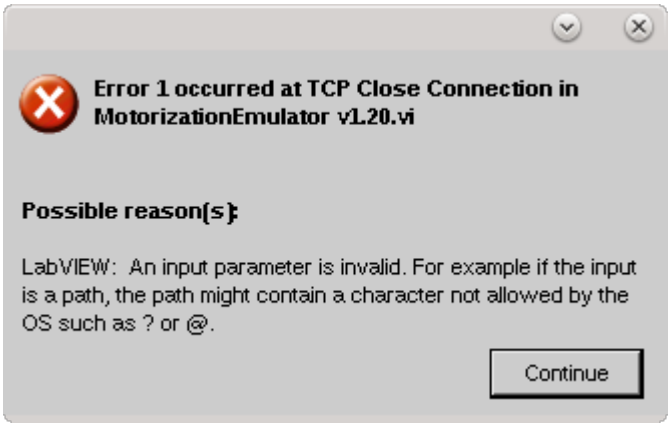

Figure 3 - Error message

6 What's new

2022-02-09 Added Hexapod Servo commands + Control button
2022-02-03 High CPU% usage when no connection; bug cleared
2021-12-09 New Hexapod compatibility
2020-03-11 Added HOSP / GHOSP functions
2019-08-08 Bugfix CPU high (28%) in executable
2019-03-16 "Err" at "Position" generates an error of com for tests
2017-03-22 HO added for Set Home position
2016-05-11 GOIX added
2016-03-03 OST counter / message length
2015-09-15 Notify Position answer for M3 bugfix
2015-07-09 Notify Position addon / OST > position attained
2015-06-16 M2 full emulation
2015-03-11 "LV" command implemented