LEIBNIZ INSTITUT FÜR ASTROPHYSIK POTSDAM

LEIBNIZ-INSTITUT FÜR SONNENPHYSIK (KIS)

MAX PLANCK INSTITUT FÜR SONNENSYSTEM-FORSCHUNG

GREGOR:

Document No.: GRE-KIS-MAN-0012

Version: <2>

Date: <30.9.2019>

Signatures & Approval

	Name	Signature	Date
Prepared by	Olivier Grassin – KIS Software engineer		2015-09-30
Reviewed by	Lucia Kleint - KIS GREGOR lead scientist		2019-10-11
Approved by			
Released by			

Change Log				
Vers.	Date	Author	Description of Changes	Sect./Para.
1.0	2015-9-30	O.G.	New Document	
1.2	2015-12-10	O.G.	Added: “logfile” {-dt} option.	
1.3	2016-04-22	O.G.	Change “Logs”, “Reports” and “Scripts” folders location. (see Chapter 8 – Files & Folders) Bug correction for long TimeOut; Now response button to “Stop” action (user aborting sequence for a non-responsive command) is no longer than 1 sec preventing the waiting end of TimeOut.	
1.31	2016-05-20	O.G.	Software changes in deep, due to remote panel adaptation, involving creation of file browser and message boxes. The displayed version of Conductor is v1.2.9.	
1.4	2016-06-01	O.G.	Added commands: “Answer” and “Timing”	
1.5	2016-09-08	O.G.	Added new math functions: “mult,div,abs,sqroot,log10,ln,exp,sin,cos,tan,sqr,inv,round,int,neg,pow” Added commands: “GetTimeStamp”, “Chain” Added default value/string to “:inbox:” declaration command.	
2	2019-07-30	O.G.	Office 2019 version. Minor corrections.	

1 Abbreviations & acronyms 4

2 Scope 4

3 Interface description 4

3.1 Script window and control panel 5

3.2 Variables for script interaction 8

3.3 Sequence window 8

3.4 Top subpanel 9

4 How to write Scripts 9

5 Conductor Instructions List 11

5.1 General instructions 12

5.2 Mathematical operations 15

5.3 Special instructions 18

6 Script examples 20

7 Answer messages 28

8 Files & folders 29

9 Remote panel 30

List of Tables
(insert if needed)

List of Figures
(insert if needed)

1 Abbreviations & acronyms

DCP	:	Device Communication Protocol
ACK	:	Acknowledgement
GCS	:	GREGOR Control System
GTCS	:	GREGOR Telescope Control System
ICS	:	Instrument Control System
M2M3control	:	Motorization interface for M2, M3, M5, M6 & M7
gdbs	:	GREGOR Data Base Server
TTL	:	Time To Live (valid time for data stored in database)

2 Scope

Conductor can be defined as a multi-client control-command manager using the DCP protocol in the GREGOR framework; a tool for developers, assistants, and observers.

As a sequencer, it sends requests through the network to specific clients and waits for execution within a time-out delay. A sequential command list called script, written by the user, is stored and processed step by step, giving back the results and answer messages.

Scripts can cover a variety of tasks from setup and tests, to configuration, database interaction, observation programs, data logging and more.

Conductor was preliminary written in LabVIEW 8.0 for Windows and Linux, now deprecated. The current version is developed and running on the GCS as an executable in LabVIEW 15 for Linux.

3 Interface description

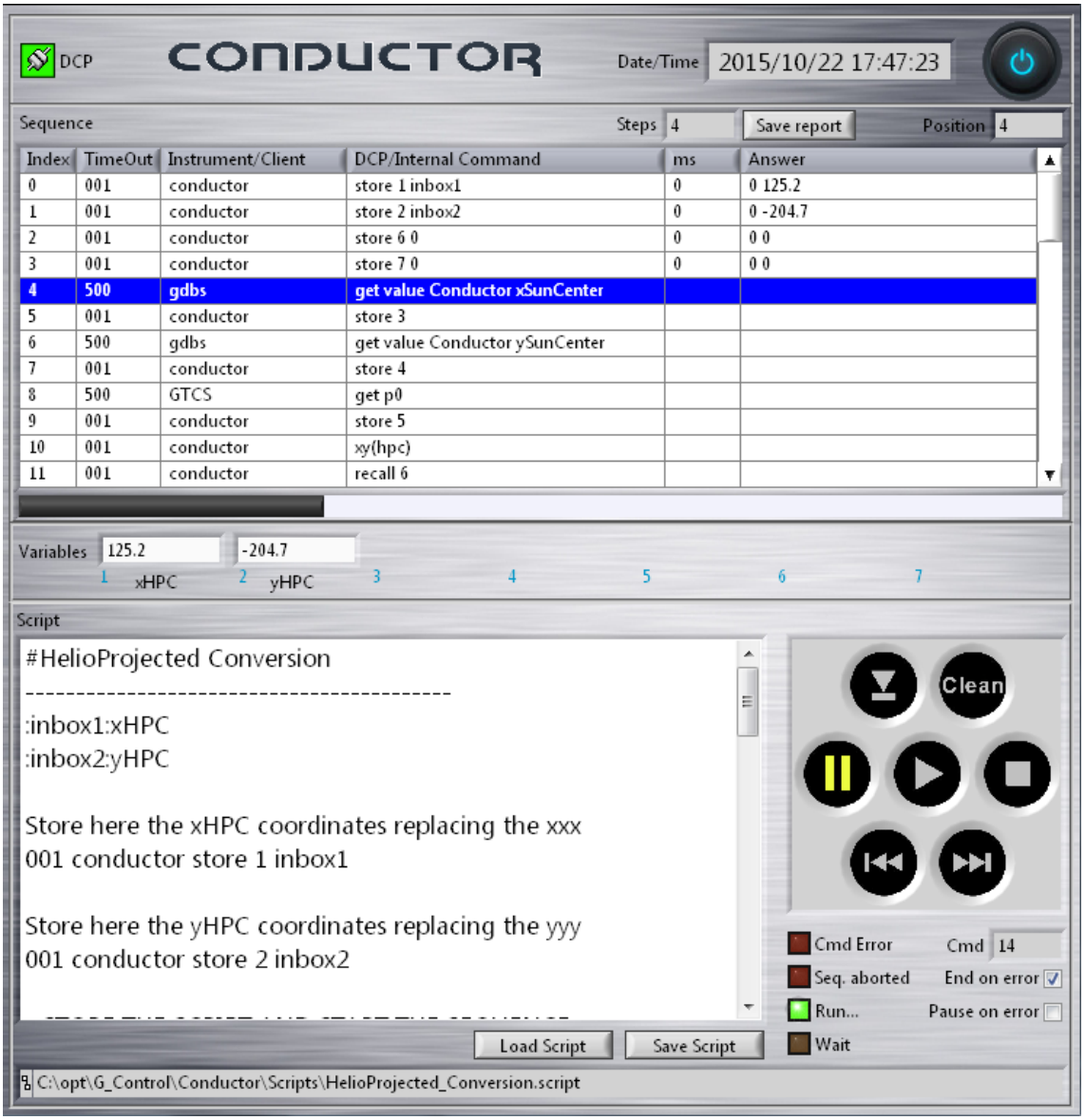
The main panel of the Conductor interface is composed of 4 subpanels, from bottom to top:

Script window and control panel…… (3.1)

Variables for script interaction…….. (3.2)

Sequence window…….. (3.3)

Top subpanel……. (3.4)



Conductor interface while executing a sequence…

3.1 Script window and control panel

Summary of user interaction:

- Edit a script
- Load a script from Script folder
- Save a script to Script folder
- Indicators and check boxes
- Command Error
- Sequence aborted
- Run…

- Wait
- On error behavior
- Sequence Controller
- Store the script
- Clear the script window
- Pause sequence execution
- Start/Continue sequence
- Stop/Abort sequence execution
- Go to top/end of sequence (visual only)

Edit a script

In this window, the script can be written or edited by the user.

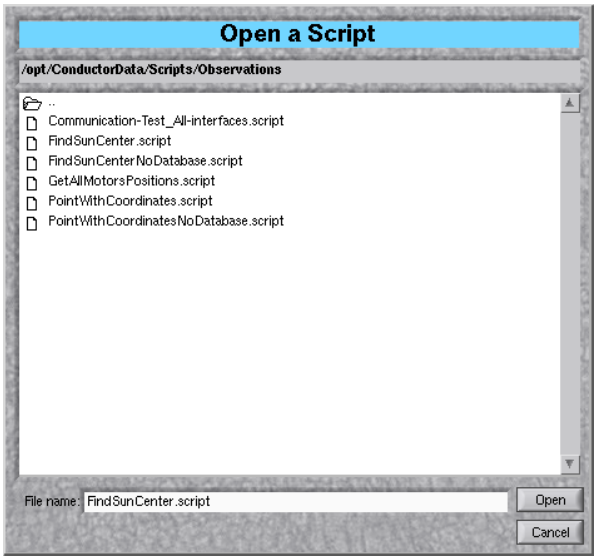
Use the scroll bar to navigate up and down inside the script, or navigation keys.

Edit functions as Ctrl-C, Ctrl-V and Ctrl-X are available.

More details in the "How to write scripts" section.

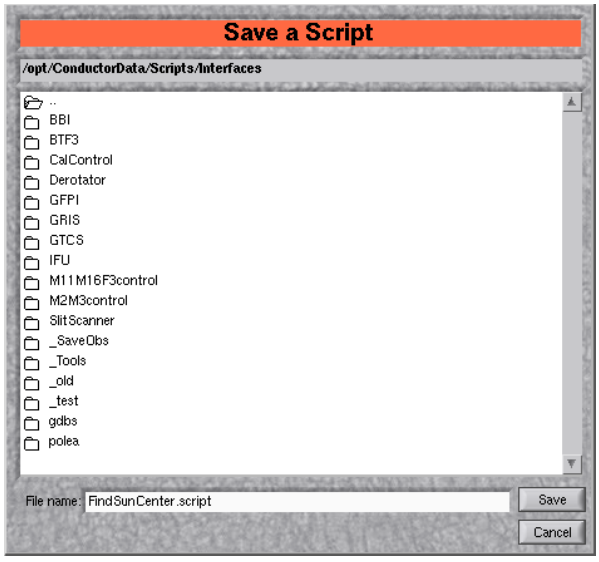
Load a script from Script folder

The "Load Script" button allows the user to retrieve a saved script from the "Scripts" folder. A dialog box is opened and the user has the possibility to navigate outside the "Scripts" folder, but it is recommended to use the "Scripts" folder and subfolders to store them.



Save a script to Script folder

The "Save Script" button opens a dialog box to navigate inside the folders, to define a filename, or using a "Shift" click (keyboard left shift key) as a quick "save and replace" method that just asks to confirm the saving.

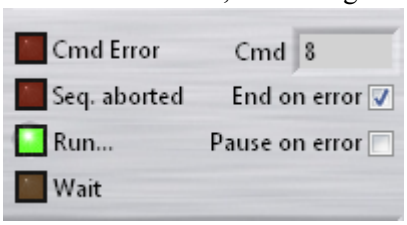


Indicators and check boxes

Command Error: This indicator lights up if during the sequence execution the answer of a command is not "0" (error occurred or warning). It will be reset after loading the script to the sequencer window.

Sequence aborted: If during the execution of the sequence, the user presses the "Stop" button or an error occurs in the function of the "… on Error" check boxes, the indicator will remain on, until the script is newly loaded.

Run…: This indicator will remain green from the start of the sequence until the end, indicating the execution of the script.



Wait: When the Conductor command "Wait" is running this indicator will flash.

On error behavior: The user can select how an error affects the running of the sequence; if "End on error" is checked, the sequence will abort. If "Pause on error" is checked, the user can then decide whether to continue or abort the sequence. To continue, the user has to click the "Play" button again. The state of these check boxes can be modified programmatically, as a Conductor command. This can be useful to skip warning errors during sequence phases.

Sequence Controller

Store/load the script: When the script is ready for execution, it has to be loaded into the sequencer. In this phase the script is filtered of unused information, such as comments, extra spaces/tabulations, checked for step numbers, and stored. A step number is visible in the corresponding indicator before the start of the sequence that is now ready to start.

Clear the script window: This button clears the sequence spreadsheet of all data, so no command can be inadvertently sent.

Pause sequence execution: The user can pause a sequence during execution when desired. This does not affect the current running command. "Left Shift" key will also generate a pause.

Start/Continue sequence: Click this "Play" button to start or continue the sequence execution. If the "Left Shift" key is held down during a "Start/Continue" click, the sequence is executed step by step.

Stop/Abort sequence execution: The user can stop the sequence when desired by clicking this button. Conductor will wait for the time-out of the current running command to stop. (see below: "Answer messages" section.)

Go to top/end of sequence (visual only): Move the sequence cursor to the top or to the next line to execute the sequence. If the script has just been loaded, these buttons have no effect; use the right side scroll bar instead.

3.2 Variables for script interaction

An input box was implemented for user-defined values.

These boxes allow the user to directly enter data inside, then store/load and execute the sequence without having to modify the script manually.

Variables can be text or numeric values. How to use variables will be explained in the "Special instructions / *inbox*" (5.2) section.

3.3 Sequence window

The sequence window is essentially a display of the current instructions list.

As the user stores/loads the script with the corresponding command button, the script is analyzed and stored into the sequencer spreadsheet as a list of commands.

Conductor and DCP instruction words are generally separated by one space, so extra spaces and tabulations will be removed automatically to avoid problems. All the script command lines will then be visible in the sequence spreadsheet, each parameter distributed to the corresponding column:

Index: *Internal command line numbering*

TimeOut: *Defined by user*

Instrument/Client: *Defined by user*

DCP/Internal command: *Defined by user*

As answer of execution, the result will appear into the following columns:

ms: *Total communication latency (Timing)*

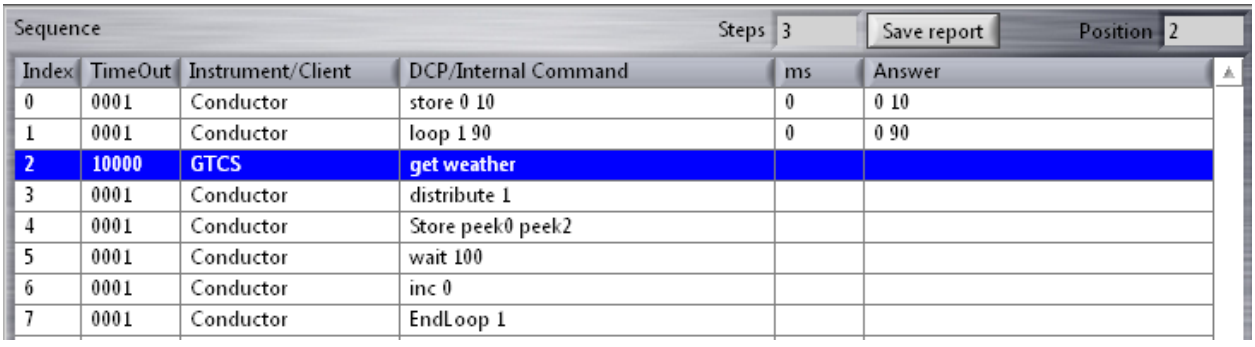
Answer: *Result of command line execution*

The number of the total sequence steps will be then processed and updated in the "Steps" indicator, either in linear or loop scripts before execution. While running the sequence, the indicator shows the number of executed steps.

The "Position" indicator shows the current instruction line index.

A "Save report" button allows the user to save the result of the sequence execution to a text file in the "Reports" folder, for later viewing, statistics…

At the bottom of this subpanel, a progress bar indicates the progress of the completed steps of the whole sequence. This bar does not indicate the time for comletion, because it depends on the time response of the involved systems during the script execution.



Index	TimeOut	Instrument/Client	DCP/Internal Command	ms	Answer
0	0001	Conductor	store 0 10	0	0 10
1	0001	Conductor	loop 1 90	0	0 90
2	10000	GTCS	get weather		
3	0001	Conductor	distribute 1		
4	0001	Conductor	Store peek0 peek2		
5	0001	Conductor	wait 100		
6	0001	Conductor	inc 0		
7	0001	Conductor	EndLoop 1		

3.4 Top subpanel

It shows from left to right, the status of the DCP connection, the date and time, a “Power interface” button.

DCP indicator:

At program startup, Conductor tries to connect to the DCP server;

if the indicator is green, the connection is established. If the connection fails, the indicator remains red. In case a second Conductor session is started the indicator will toggle between red and green, indicating that one of the sessions has to be closed.

Tip: Passing the mouse over this indicator will show the Conductor client name and DCP server name, if connected.

Date/Time:

Gives the current time of the computer; it will be used to generate part of the report and logs file name and files data.

Power interface button:

Clicking this button shuts down the interface; a message box will appear on screen waiting for user confirmation. The operation can be cancelled.

At shutdown, the interface saves the memories and will reload them at the next startup.

4 How to write Scripts

The user has to follow the following rules and obviously the DCP command structure, parameters and knowledge of possible answers. Please, refer to the specific client DCP documentation, in order to elaborate safe and reliable scripts.

Only lines beginning with a number (and different than zero) will be taken as command lines, so all lines starting with text will be taken as comment, which is helpful to keep an explicit script. Spaces can also be used at the beginning of the line to create indentation and easy reading, especially for loops.

End of line comments are not allowed.

Script name Title

The user can tag the script with "#" on the first line, first character, followed by a title name, to define a proposed name for saving the script and report.

Space characters will be filtered and replaced with "_".

Script name extension is ".script".

Ex: #SunCenteringProcedure will give the name:

"SunCenteringProcedure.script".

Declaration:

Declarations must be set inside the first 50 lines of the script. Please, see the "Special instructions" section below.

Comment:

A line beginning with text characters is always considered as a comment line.

Command line:

The command line is composed of 3 parameters.

First parameter:

The command line must begin with a number that indicates the time-out for execution. The time-out is always expressed in milliseconds.

For commands that need some time to execute, such as motorization movements, telescope tracking moves, etc., please evaluate the time and add a margin to avoid false time-out errors. The time-out value for Conductor commands can be set to a value greater than 0. "1", "001" or "0001" is allowed and easy readable as the command is executed internally: the time-out is not taken in account.

Second parameter:

This is the name of the client. Use the correct name taking care of respecting the case, as specified in the client's documentation. To be sure that the client is connected to the DCP server, a "set nop" DCP command can be sent to ensure the connection.

For all Conductor instructions, this parameter is "Conductor" and is not case sensitive. The word "Conductor" can also be replaced by "&" to speed up script writing.

Third parameter:

This is the command line as described in the client DCP Command documentation. This parameter is usually composed of multiple strings of command words and values. Please, respect spaces and case.

For Conductor internal instructions please refer to the list below.

5 Conductor Instructions List

In this document, the reader must understand:

[memory]	as numerical data for memory number. The content of memory is always a string of characters that can also be interpreted as number.
[data string]	as text or numerical data.
[value]	as numerical data.
[index]	as numerical data for loop index.
[index box]	as numerical data for input box index.
[caption string]	as text string.
{…}	as optional content.

…these data have to be inserted inside the command line and will be processed. Conductor instructions are not case sensitive.

General instructions	**Wait**
	Pause
	PauseOnError
	EndOnError
	ClearError
	Store
	Recall
	Clear
	End
	Loop
	EndLoop
	Chain
	Distribute
	xy(hpc)
	Answer
	Timing
Mathematical Operations	**Inc, Dec, Add, Sub, Mult, Div, SQRoot, Sqr, Pow, Log10, Ln, Exp, Sin, Cos, Tan, Abs, Inv, Neg, Round**
Special	**Peek**
	Inbox
	Logfile

5.1 General instructions

Wait
001 Conductor Wait [value]

Parameter	Data format	Meaning
[value]	DOUBLE	Waits for a time [value] expressed in milliseconds. Accuracy depends on computer/OS/graphics behavior and is shown in the “ms” sequence answer column.
Ex:	001 Conductor Wait 2000	

Pause
001 Conductor Pause

Parameter	Data format	Meaning
None	none	Pauses the sequence execution. Wait for the "Play" button to be clicked again to continue. "Stop" button can abort the sequence.
Ex:	001 Conductor Pause	

PauseOnError
001 Conductor PauseOnError [data string]

Parameter	Data format	Meaning
[data string]	STRING	Configures interface to pause when an error occurs. Enabled with value "1" or "On", disabled with "0" or "Off".
Ex:	001 Conductor PauseOnError On	

End
001 Conductor End

Parameter	Data format	Meaning
none	none	Terminates the script at command position. The following instructions are not executed. This is useful for debugging or testing a script.
Ex:	001 Conductor End	

EndOnError
001 Conductor EndOnError [data string]

Parameter	Data format	Meaning
[data string]	STRING	Configures interface whether to end the sequence or not when an error occurs. Enabled with value "1" or "On", disabled with "0" or "Off".
Ex:	001 Conductor EndOnError On	

ClearError
001 Conductor ClearError

Parameter	Data format	Meaning
none	none	Resets the "CMD error" status to avoid aborting the sequence, if the "EndOnError" is set back.
Ex:	001 Conductor ClearError	

Store
001 Conductor Store [memory]
001 Conductor Store [memory] [data string]

Parameter	Data format	Meaning
[memory] [data string]	U8 STRING	Stores the query value from the last command answered by the client at memory [memory], or if a [data string] is present, this data is stored after the memory value. 0 ≤ [memory] ≤ 99.
Ex:	001 Conductor Store 3 001 Conductor Store 10 153.61	

Recall
001 Conductor Recall [memory]

Parameter	Data format	Meaning
[memory]	U8	Recalls and shows only the data stored at [memory] 0 ≤ [memory] ≤ 99.
Ex:	001 Conductor Restore 7	

Clear
001 Conductor Clear [memory]

Parameter	Data format	Meaning
[memory]	U8	Clears the data stored at [memory]. Confirmed answer is "0 ". 0 ≤ [memory] ≤ 99.
Ex:	001 Conductor Clear 7	

Distribute
001 Conductor Distribute [memory]

Parameter	Data format	Meaning
[memory]	U8	Parses the last answer message and distributes all its parameters beginning at the indicated memory number. The parameters number is limited to 10, so last "Distribute" memory number is 89. This function has to be carefully used as it could overlap other memories. The answer of "gdbs get value" command is automatically parsed and does not need the Distribute command.
Ex:	001 Conductor Distribute 10	

Loop
001 Conductor Loop [index] [value]

Parameter	Data format	Meaning
[index] [value]	U8 I32	Sets loop number [index] start counter to [value]. 0 ≤ [index] ≤ 99
Ex:	001 Conductor Loop 1 25	

EndLoop
001 Conductor EndLoop [index]

Parameter	Data format	Meaning
[index]	U8	Decrements loop number [index] counter and tests with 0; - if counter = 0 the loop ends and continues to next instruction. - if not, the sequence returns to the position after the corresponding index: "001 Conductor Loop [index] [value]" instruction.
Ex:	001 Conductor EndLoop 1	

Chain
001 Conductor Chain [memory]
001 Conductor Chain [memory] [data string]

Parameter	Data format	Meaning
[memory] [data string]	U8 STRING	Concatenate memory [memory] with the query value from last command answered if no [data string] is present; otherwise memory [memory] is concatenated with [data string] data. All spaces are removed. $0 \leq$ [memory] ≤ 99.
Ex:	001 Conductor Chain 3 001 Conductor Chain 3 :peek1	

xy(hpc)
001 Conductor xy(hpc)

Parameter	Data format	Meaning
none	none	Converts the x,y heliographic projected coordinates to GREGOR telescope x,y coordinates. For the calculation the function needs to have the following parameters in memory: **Memory 1**: xHPC > have to be written in script. **Memory 2**: yHPC > have to be written in script. **Memory 3**: xSunCenter > Retrieved from database / (Previously stored by SunCentering script.) **Memory 4**: ySunCenter > Retrieved from database / (Previously stored by SunCentering script.) **Memory 5**: p0 angle > Retrieve from GTCS. Result is stored in memory 6 and 7: **Memory 6**: x **Memory 7**: y
Ex:	001 Conductor xy(hpc)	

Answer
001 Conductor Answer

Parameter	Data format	Meaning
none	none	Returns all last command parameters; ”status” and “parameter(s)”
Ex:	001 Conductor Answer	

Timing
001 Conductor Timing

Parameter	Data format	Meaning
none	none	Returns the timing value of the last command.
Ex:	001 Conductor Timing	

GetTimeStamp
001 Conductor GetTimeStamp [data string]

Parameter	Data format	Meaning
[data string]	STRING	None : YYYYMMDDhhmmss000 "Year" : YYYY "Month" : MM "Day" : DD "Hour" : hh "Minute" : mm "Second" : ss "ms" : 000 (milliseconds)
Ex:	001 Conductor GetTimeStamp 001 Conductor GetTimeStamp Day	

5.2 Mathematical operations

Inc
001 Conductor Inc [memory] {[value]}

Parameter	Data format	Meaning
[memory] [value]	U8 I32	Increments the target memory by value as integer number; if memory is text string it will be considered as 0. If no [value] is present the increment is 1.
Ex:	001 Conductor Inc 22 10	

Dec
001 Conductor Dec [memory] {[value]}

Parameter	Data format	Meaning
[memory] [value]	U8 I32	Decrements the target memory by value as integer number; if memory is text string it will be considered as 0. If no [value] is present the decrement is 1.
Ex:	001 Conductor Dec 22 10	

Add
001 Conductor Add [memory] [value]

Parameter	Data format	Meaning
[memory] [value]	U8 DOUBLE	Adds the target memory by value as fractional number; if memory is text string it will be considered as 0. If no [value] is present, it adds 0 to memory.
Ex:	001 Conductor Add 22 10.5	

Sub
001 Conductor Sub [memory] [value]

Parameter	Data format	Meaning
[memory] [value]	U8 DOUBLE	Subtracts the target memory by value as fractional number; if memory is a text string it will be considered as 0. If no [value] is present, it subtracts 0 from memory.
Ex:	001 Conductor Sub 44 3.2	

Mult
001 Conductor Mult [memory] [value]

Parameter	Data format	Meaning
[memory] [value]	U8 DOUBLE	Multiplies the target memory by value as fractional number; if memory is a text string it will be considered as 0. If no [value] is present, it multiplies the memory by 1.
Ex:	001 Conductor Mult 3 1.4	

Div
001 Conductor Div [memory] [value]

Parameter	Data format	Meaning
[memory] [value]	U8 DOUBLE	Divides the target memory by value as fractional number; if memory is a text string it will be considered as 0. If no [value] is present, it divides the memory by 1.
Ex:	001 Conductor Div 1 10	

SQRoot
001 Conductor SQRoot [memory]

Parameter	Data format	Meaning
[memory]	U8	Applies the square root to the target memory; if memory is a text string it will be considered as 0.
Ex:	001 Conductor SQRoot 1	

Sqr
001 Conductor Sqr [memory]

Parameter	Data format	Meaning
[memory]	U8	Applies the square to the target memory; if memory is a text string it will be considered as 0.
Ex:	001 Conductor Sqr 1	

Pow
001 Conductor Pow [memory] [value]

Parameter	Data format	Meaning
[memory] [value]	U8 DOUBLE	Applies to the target memory the power by value; if memory is a text string it will be considered as 0. If no [value] is present, memory is not changed.
Ex:	001 Conductor Pow 1 3	

Log10
001 Conductor Log10 [memory]

Parameter	Data format	Meaning
[memory]	U8	Applies the logarithm in base 10 to the target memory; if memory is a text string it will be considered as 0.
Ex:	001 Conductor Log10 1	

Ln
001 Conductor Ln [memory]

Parameter	Data format	Meaning
[memory]	U8	Applies the natural logarithm to the target memory; if memory is a text string it will be considered as 0. Ln[textstring] will yield the answer -1 NaN (with -1 standing for 'warning').
Ex:	001 Conductor Ln 1	

Exp
001 Conductor Exp [memory]

Parameter	Data format	Meaning
[memory]	U8	Applies to memory, exponential of the target memory; if memory is a text string it will be considered as 0, and the answer will be 1.
Ex:	001 Conductor Exp 1	

Sin
001 Conductor Sin [memory]

Parameter	Data format	Meaning
[memory]	U8	Applies to memory (angle in degree), sine to target memory; if memory is a text string it will be considered as 0.
Ex:	001 Conductor Sin 1	

Cos
001 Conductor Cos [memory]

Parameter	Data format	Meaning
[memory]	U8	Applies to memory (angle in degree), cosine to target memory; if memory is a text string it will be considered as 0.
Ex:	001 Conductor Cos 1	

Tan
001 Conductor Tan [memory]

Parameter	Data format	Meaning
[memory]	U8	Applies to memory (angle in degree), tangent to target memory; if memory is a text string it will be considered as 0.
Ex:	001 Conductor Sin 1	

Abs
001 Conductor Abs [memory]

Parameter	Data format	Meaning
[memory]	U8	Applies the absolute value to the target memory; if memory is a text string it will be considered as 0.
Ex:	001 Conductor Abs 1	

Inv
001 Conductor Inv [memory]

Parameter	Data format	Meaning
[memory]	U8	Inverts the memory value; if memory is a text string it will be considered as 0 and answer will be “inf” for infinite.
Ex:	001 Conductor Inv 1	

Neg
001 Conductor Neg [memory]

Parameter	Data format	Meaning
[memory]	U8	Changes the sign of memory value; if memory is a text string it will be considered as 0.
Ex:	001 Conductor Neg 1	

Round
001 Conductor Round [memory]

Parameter	Data format	Meaning
[memory]	U8	Rounds to nearest integer value the target memory; if memory is a text string it will be considered as 0.
Ex:	001 Conductor Round 1	

5.3 Special instructions

Peek
[time out] [Client] [DCP command …Parameter as {Peek[memory]}…]

Parameter	Data format	Meaning
[memory]	U8	Replaces the "Peek[memory]" string by the string value stored at [memory] in the command line and sends it to the client for execution. As a hybrid command, this is used to transmit data stored and processed in Conductor memories to the client. Multiples replacements are possible. Note: no space between "Peek" and [memory], as it needs to be one string.
Ex:	In script: 500 SlitScanner set position abs SM1 1 Peek2 If memory 2 contains value 13674 The DCP command sent will be: SlitScanner set position abs SM1 1 13674	

Inbox *as declaration*
:Inbox[box index]:{[caption string]} {[data string]}

Parameter	Data format	Meaning
[box index] [caption string] [data string]	U8 STRING STRING	Declares and sets visible/editable an input box. Input boxes declarations must be in the first 50 script lines. The declared input box will be visible in the main panel after storing the script ("Store" button), or when the script containing the declaration is loaded. A string caption will appear under its box. Do not use more than 6 or 7 letters for string name to fit in the available space. 1 ≤ [box index] ≤ 7 Default value for the inbox. It will update only when the script is loaded.
Ex:	:Inbox1:xHelioP 100	

Inbox *as special instruction*
[time out] [Client] [DCP/Conductor command …Parameter{ Inbox[box index]}…]

Parameter	Data format	Meaning
[box index]	U8	Replaces instruction parameter in the command line by the Inbox[box index] indicator data. This is useful to quickly send a sequence after updating the inbox value, without having to modify the parameters in the script. Note: no space between "Inbox" and [box index], as it needs to be one string. 1 ≤ [box index] ≤ 7
Ex:	In script: :inbox1:xHelioProj 001 Conductor Store 1 Inbox1 If Inbox1 indicator contains "6.18", at sequence execution command to send will be: 001 Conductor Store 1 6.18	

Logfile *as declaration*
:Logfile:{append/}[string]

Parameter	Data format	Meaning
[string] append/	STRING	Declares a log filename to store data. The file is opened when the sequence starts and closed when sequence ends or is aborted. The complete name of the file is composed of the current date and time at the start of sequence, the string "-seq-", the [string] word and the filename extension ".log". A new file is generated for each time the execution is started. If option "append/" is set, the data is added to the file. If the file does not exist, it is created. This allows recording data to the same file, e.g. to facilitate creating charts.
Ex:	:Logfile:weather Creates a file as : 20151027_112120-seq-weather.log :Logfile:append/positions Opens/creates a file as : positions.log	

Logfile *as special instruction*
001 Conductor Logfile {-dt}{[string][space]}{Peek[index]{[space]Peek[index]…}}

Parameter	Data format	Meaning
[string]	STRING	Optional parameters can be placed after the instruction to define a memory to record into the file line. More than one parameter can be set.
Peek[index]	U8	If no parameter is set, the result of the last answer will be recorded. The first file line data record is the current date and time followed by parameters to record. The time includes milliseconds separated by a dot. Data record separator is a [Space] character.
-dt	STRING	Option "-dt" at first argument, avoids printing the date/time at the beginning of the line. It can be used to create spreadsheet headers into the script before collecting data.
Ex:	In script: 001 Conductor Logfile M5-1 Peek24 File line content: "20151027 11:21:23.118 M5-1 20485" (Memory 24 contains 20485 … position of M5-1 motor, for ex.) In script: 001 Conductor Logfile –dt Date Time M5-1 File line content: "Date Time M5-1"	

6 Script examples

Move M2 focus motor from a start position to a final position by steps with the use of variables (and loop):

Script example 1:

#Move M2 focus in loop
(script name)

Declare variables:
:Inbox1:StartPos
:Inbox2:StepInterval
:Inbox3:StepsNumber

Configure Conductor for stopping on error:
001 Conductor EndOnError On

Test M2M3control interface for communication:
500 M2M3control set nop

Move motor to start position (consider time-out for moving):
5000 M2M3control set position abs M2 3 inbox1

Begin the loop, setting StepsNumber:
001 Conductor loop 1 inbox3
Delay at the current position:
    001 Conductor Wait 2000
    Move M2 one step with the StepInterval value:
    5000 M2M3control set position rel M2 3 inbox2
001 Conductor EndLoop 1

View final position of M2 focus:
500 M2M3control get position abs M2 3

End of script

The user has to enter the respective values before starting the sequence.
In this example the user can quickly switch between 3 spots on the Sun:

Script example 2:

#Move Telescope to 3 spots
(script name)

Declare variables for (x,y) positions of spots:
:Inbox1:Spot1x
:Inbox2:Spot1y
:Inbox3:Spot2x
:Inbox4:Spot2y
:Inbox5:Spot3x
:Inbox6:Spot3y

Configure Conductor for stopping on error:
001 Conductor EndOn Error On

Test telescope control interface for communication:
500 GTCS set nop

Request time of start:

500 GTCS get time

Begin a loop of 20 repetitions:

001 Conductor loop 1 20

Wait for “Play/Continue” click:

001 Conductor Pause

Move Telescope to Spot1 position:

10000 GTCS set discPos 0 0 inbox1 inbox2

Wait for “Play/Continue” click:

001 Conductor Pause

Move Telescope to Spot2 position:

10000 GTCS set discPos 0 0 inbox3 inbox4

Wait for “Play/Continue” click:

001 Conductor Pause

Move Telescope to Spot1 position:

10000 GTCS set discPos 0 0 inbox5 inbox6

001 Conductor EndLoop 1

End of script

Before starting the sequence, the user has to enter the target coordinates into respective boxes. Sequence can be stopped at any time by pressing the “Abort” button. Target coordinates can be updated on the fly…

Sun Centering (AO) procedure:

Script example 3:

#SunCenteringProcedure

(script name)

Declare the log file to record the offset Sun center.

:Logfile:append/SunCenteringPositions

To execute this procedure,

the user has to follow these points:

* Assume telescope is in tracking mode
* Center F3 beam with M5
* Enter menu: 'Run AO...'
* Center beam on DM (AO) with
  - Automatic pupil steering (M11) ON
  - then wait for AO to send a few commands to M11
  - Automatic pupil steering (M11) OFF
* Focus telescope with M2
* Store and Start this script

The operation takes about 4:10 minutes and the x,y values
at the GTCS interface are the center offsets...

Set Conductor to stop on error:
001 Conductor EndOnError On

Test communications with clients:
500 M2M3control set nop
500 GTCS set nop

Set Conductor to continue on error:
001 Conductor EndOnError Off
001 Conductor PauseOnError Off

Retrieve and show the motor positions for M5 & M2-z axis:
500 M2M3control get position abs M5 1
500 M2M3control get position abs M5 2
500 M2M3control get position abs M2 3

Execute the AO find center procedure:
300000 gaos set_kaos_findcenter

Get time from GTCS:
500 GTCS get time

Ask for tracking position on solar disk:
500 GTCS get discPos

Store the disk position x,y to internal memory starting at memory 10:

001 Conductor Distribute 10

Set Conductor to pause if error occurs:

001 Conductor PauseOnError On

Store the center coordinates into GREGOR Database with TTL of 1200 s or 20 min:

500 gdbs set value xSunCenter Peek10 - 1200

500 gdbs set value ySunCenter Peek11 - 1200

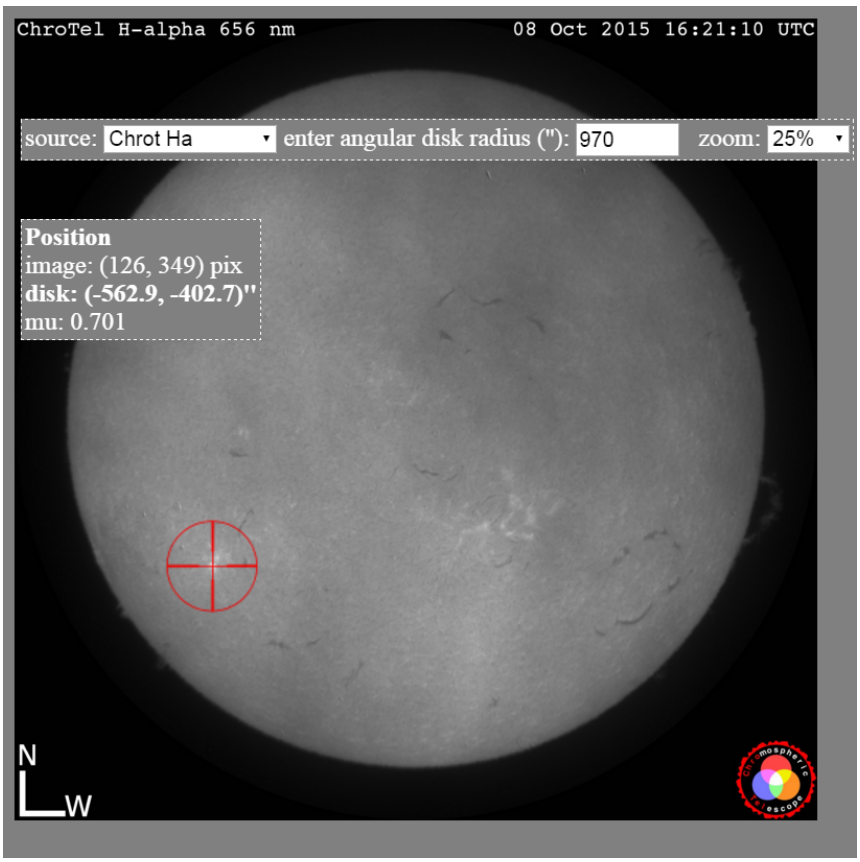
Record the Sun center position (x,y) to log file:

001 Conductor Logfile Peek10 Peek11

End of script

After script execution, the report can be saved with the “Save report” button.

In this example, the telescope (x,y) solar coordinates will be calculated for solar heliographic projected coordinates obtained from *http://k2.tt.iac.es/target.html*



For this purpose, the following data are needed:

xHPC : Heliographic projected x value (in this case = -562.9)

yHPC : Heliographic projected y value (in this case = -402.7)

p0 : Solar angle obtained from tracking (GTCS)

xCenter: Retrieved from gdbs (database)

yCenter: Retrieved from gdbs (database)

The user will have to enter the heliographic projected coordinate values into variable boxes before starting the sequence and the next script will point the telescope to these coordinates.

Script example 4:

#HelioProjected Conversion

(script name)

Declare the variable boxes for heliographic projected coordinate:

:inbox1:xHPC

:inbox2:yHPC

PLEASE, STORE THE SCRIPT,

ENTER THE xHPC and yHPC

COORDINATES TO THE

CORRESPONDING BOX,

AND START THE SEQUENCE

Store the xHPC value in memory 1:

001 conductor store 1 inbox1

Store the yHPC value in memory 2:

001 conductor store 2 inbox2

Clearing result memories:

001 conductor store 6 0

001 conductor store 7 0

Get from database xSunCenter & store in memory 3:

500 gdbs get value Conductor xSunCenter

001 conductor store 3

Get from database xSunCenter & store in memory 4:

500 gdbs get value Conductor ySunCenter

001 conductor store 4

Get the p0 angle from GTCS & store in memory 5:

500 GTCS get p0

001 conductor store 5

Execute the Conductor conversion procedure:

001 conductor xy(hpc)

Show the (x,y) calculated values of memories 6 & 7:

001 conductor recall 6

001 conductor recall 7

Move the telescope to the (x,y) position:

10000 GTCS set discPos 0 0 peek6 peek7

Help and comments

------Set the appropriate memories

M1 = xHelioProjected

M2 = yHelioProjected

------Get the values from system

M3 = xCenter >>> given by Centering Procedure (gdbs)

M4 = yCenter >>> given by Centering Procedure (gdbs)

M5 = p0 angle >>> From GTCS on tracking

-------Result is written in Memories M6 and M7

M6 = x

M7 = y

End of script

Record the wind speed, temperature, and atmospheric pressure from GTCS, with one acquisition every 15 seconds, 240 times…

Script example 5:

#Save Weather data to file

(script name)

Declare the log file name:

:logfile:weather_wind_temp_pressure

Set Conductor to stop on error:

001 Conductor EndOnError On

Test communication with clients:

500 GTCS set nop

Set Conductor to continue on error:

001 Conductor EndOnError Off

001 Conductor PauseOnError Off

Set the Loop number 1 to repeat 240 times:

001 Conductor Loop 1 240

Get the weather data from the GTCS:

500 GTCS get weather

Distribute the data to successive memories starting at memory 10:

001 Conductor Distribute 10

Record the memories 11, 12 and 15 data to the log file:

001 Conductor Logfile Peek11 Peek12 Peek15

Delay of 15 seconds:

001 Conductor Wait 15000

Go back to the first command line of the Loop number 1:

001 Conductor EndLoop 1

End of script

Some data of the file:

…

20151027 16:44:33.287 6.6 8.0 766.1

20151027 16:44:48.728 8.1 8.0 766.0

20151027 16:45:04.215 7.9 8.0 766.0

20151027 16:45:19.694 6.5 7.9 766.0

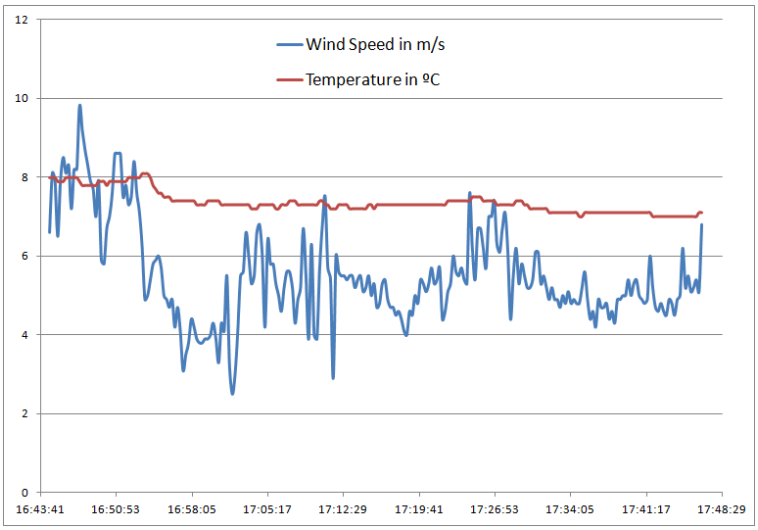
20151027 16:45:35.159 7.9 7.9 766.1

20151027 16:45:50.818 8.5 7.9 766.1

20151027 16:46:06.318 8.1 8.0 766.1

20151027 16:46:21.830 8.3 8.0 766.0

…



7 Answer messages

When a command line is processed during the sequence execution, an answer message is expected from the client within the defined time-out delay. Usually a sent DCP command will produce first an acknowledge message, and then the final answer. For acknowledgement, the time-out is determined in the Conductor configuration file, and usually fixed to 1000 ms as it is sufficient for a client to answer the request and report that it is listening over the network. A network communication problem (with client, DCP server or Pc not responding) will generate error after the ACK time-out. The "in script" time-out value refers to the final answer latency for processing or executing the command as motor movements, telescope moves, etc.

It can be represented as this:

Conductor sends:		500 SlitScanner get position abs SM1 1
[Waits…for acknowledge for 1000 ms]		
Conductor gets:	0 ACK	from SlitScanner after 10 ms
[Waits for final answer for 500 ms]		
Conductor gets:	0 FIN	from SlitScanner after 81 ms

In this case, the total answer time shown in the sequencer window is 10 + 81 = 91 ms, and "Answer" = "0 FIN". As we can see, the total maximum time can be more than 500 ms and in this case up to 1500 ms, but never higher. If the waiting time exceeds one of the time-outs, an error message is generated as the answer.

Example of an answer message:

Script command line: 8000 SlitScanner set position abs SM1 1 50000

[Waits…for acknowledgement for 1000 ms, but no answer from client]

Conductor prints: 1 TimeOut_on_Sending(DCP connection problem)

or 1 TimeOut_no_ACK (Client not responding)

First character "0" means "no error" while "1" means error. Conductor will not wait for 8000 ms, but will give an answer after 1000 ms. (config. file parameter)

If acknowledge "0 ACK" is sent but no final message received:

Conductor prints: 1 TimeOut_("ACK [message]")

After the 8000 ms + ACK latency.

If acknowledge is different of "0":

Conductor prints: 1 Error_ACK("error number")

If an unknown Conductor instruction is entered:

Conductor prints: 1 Unknown_Command

An error message or bad command will affect the running of the sequence, regarding the state of the "…On Error" check boxes, continuing or not the execution ("Pause" or "Abort"). If the sequence pauses, the user can continue or abort, depending on the circumstances.

8 Files & folders

Conductor is installed in the "opt/G_Control/Conductor" folder. It needs a subfolder named "config" in order to load its "Cdr_config.ini" configuration file that contains the following general settings:

[Network]

ConductorClientName=Conductor

GreCoServerIP=gcs.tt.iac.es

GreCoServerPort=2001

GreCoServerTimeOut=1000

Theoretically Conductor could be installed in another place with its subfolder structure and modifying the "ConductorClientName" key of the configuration file, to give the possibility to run more than one Conductor interface at the same time, with the restrictions of the knowledge of system operations…

The "GreCoServerIP" parameter can be set to "localhost" for testing purposes, with of course the DCP server running in local. The user must be aware of the risk when using the real "gcs.tt.iac.es" server for tests.

The standard port for the DCP server is 2001.

The “GreCoServerTimeOut” parameter is set to “1000” milliseconds by default, as it is sufficient to determine errors over the network or clients' problems.

“Logs”, “Reports” and “Scripts” folders stored at “opt/ConductorData” hold the corresponding files.

When Conductor is closed, the memories are saved in the “memory.txt” file of “config” folder. This file is read again at startup, so no stored memory is lost.

9 Remote panel

Conductor, as most of the GREGOR control command applications, is running on the GCS computer. This means that this application can be used from observing computers through the “Remote Panel” LabVIEW feature.

For this purpose, the user can start Conductor from the ICS menu, by clicking on the Conductor icon, after verifying that no other instance of the program is already running on the desktop or task bar.

Trying to start multiple instances of the program will show a communication problem, with a blinking communication indicator. Please, keep only one instance of the program open.

Closing the remote panel interface is done by only clicking the most top-right cross button. Do not use the “blue stop” button, which must be used for the main interface. In case of shutting down the interface from the remote one, user will need to restart it on the GCS computer from the ICS interface.

To take control on a remote panel, right click over some unused place of the interface and a menu will appear with the following options:

Request Control of VI

Release Control of VI

Show Last Message

Just select the appropriate option. Other options are available on the main interface (on the GCS computer):

Remote Server Panel >>

Regain Control

Unlock Control

Lock Control

Switch Controller

Show Last Message

When server side interface regains control, the other remote panels connected to it cannot take control, until the server panel unlocks the control.