Requirement Specifications for the

GUS Standard Interface

Version 0.4

27 February, 2015

1. Table of Contents

2. Introduction

In the business of environmental testing, the test requirements are getting more and more complex. Especially where a test in a climatic chamber, together with a vibration test has to be performed, the control of the different instruments or measurement devices can become complex. The proposal of the GUS (Gesellschaft für Umweltsimulation e.V. in Germany) wants to create an open environment, so the test engineer can easily connect and control any equipment as part of a test set-up.

In this context the GUS has defined a number of standard commands that allow to search for the connected devices and instruments, to select the required equipment or instruments and to control this equipment so the test can run according to the required test specifications.

The Requirement Specifications document describes a solution, based on the GUS Standard Interface Definition, to develop an application that makes the use of the standard interface easy and effective to the test engineer. For the test engineer it must be easy to integrate new equipment into a test set-up, to program a test sequence and to report the results of the test that has been run.

The project to develop such solution has been thought as an open source project. In this way users and developers can contribute to the improvement of the solution, adapt the solution to the local requirements, add new features or just create a user interface with the local language.

Some work already has been done, like the development of a demo solution, based on an EXCEL® spread sheet. Further information is available from the GUS AK (GUS Arbeitskreis) that developed the standard commands of the interface definition.

Ref.: http://www.gus-ev.de/index.php/arbeitskreise/standard-schnittstellen

3. Definitions

The devices, measurement systems, control system, shaker system or climatic chamber, cooling system, actuator or any equipment to be controlled by the application, further on will be called the “**Device**”.

The software to be developed, for which this tender document and list of requirements has been written, further will be called the “**Control Program**”.

The **Control Program** consists of two levels, 1 and 2: level 1 is the “**Device Definition**”. Level 2 will be called the “**Higher Level Control Program**” or the “HLCP”. The interface between the **HLCP** and a device is the GUS Standard Command Interface.

The HLCP is the translation of the German: **Übergeordnetes Steuerprogramm**.

4. GUS Standard Commands Interface Definition

Funktion	**Command**	**Action / Result**
Application Commands		
Application Selection	***GUS_Open_App***	The communication software is loaded.
Detect connected Equipment	***GUS_Scan_Devices***	Search for available Equipment. The equipment identifier is returned.
Read Equipment Properties	***GUS_GetDeviceInfo***	
Select Equipment	***GUS_OpenDevice***	The hardware to communicate with is defined. The operating and communication software is loaded. The communication is initialized.
Deselect Equipment	***GUS_CloseDevice***	The connection with the respective equipment is closed. The equipment becomes available for other control applications. The running process of the equipment is not influenced.
Deactivate Application	***GUS_CloseApp***	The communication is closed. Operating and communication software of the equipment is terminated.
Test Commands		
Test Selection and Initialization	***GUS_PrepareTest***	A predefined test is selected and loaded. The initialization is executed (e.g. selfcheck).
Start	***GUS_StartTest***	The equipment goes in RUN mode. The loaded test is started and starts to run until the end of test, or until the command STOP or PAUSE is received, or until eventually a failure terminates the test.
Stop	***GUS_StopTest***	The equipment goes in READY mode. The running test goes on hold and is terminated. All control parameters (elapsed time, run schedule, etc.) are reset and the equipment is ready for a new START. The status is identical as after the PREPARE command.
Pause	***GUS_PauseTest***	The equipment goes in PAUSE mode. The running test goes on hold but is not terminated. The elapsed time, status of the runs schedule, etc. within the test remain unchanged. The equipment remains ready to continue the test or to stop the test.
Continue	***GUS_ContinueTest***	The equipment goes in RUN mode again. The running test is continued. The run schedule, elasped time, etc. of the interrupted test are
Close Test	***GUS_CloseTest***	The loaded test is closed. The equipment returns to the default state. The equipment is subsequently ready to receive a new PREPARE command.
Response		
Query Equipment Status	***GUS_GetStatus***	Query of the equipment status: Ready, Stop, Pause, Run, Busy. See detailed list.
Query Equipment Malfunction Message	***GUS_GetError***	Query of the equipment failure condition.
Query Equipment Information	***GUS_GetInfo***	Query of additional equipment status information.

Table 1: List of the standard GUS interface commands.

The list above gives an overview of the existing commands that have been defined. The principle is that each device: test or measurement system, climatic chamber, shaker etc. in the list of available devices remains independent. Each device might have its own test program, project or run schedule, or just can be switched ON or OFF, as defined in the test specification.

5. Principle of Operation

For each device a certain sequence has to be run through. Depending on the available interface of the device, the interface can be very simple (e.g. just some relay contacts) or can be more complex (e.g. like an ActiveX interface).

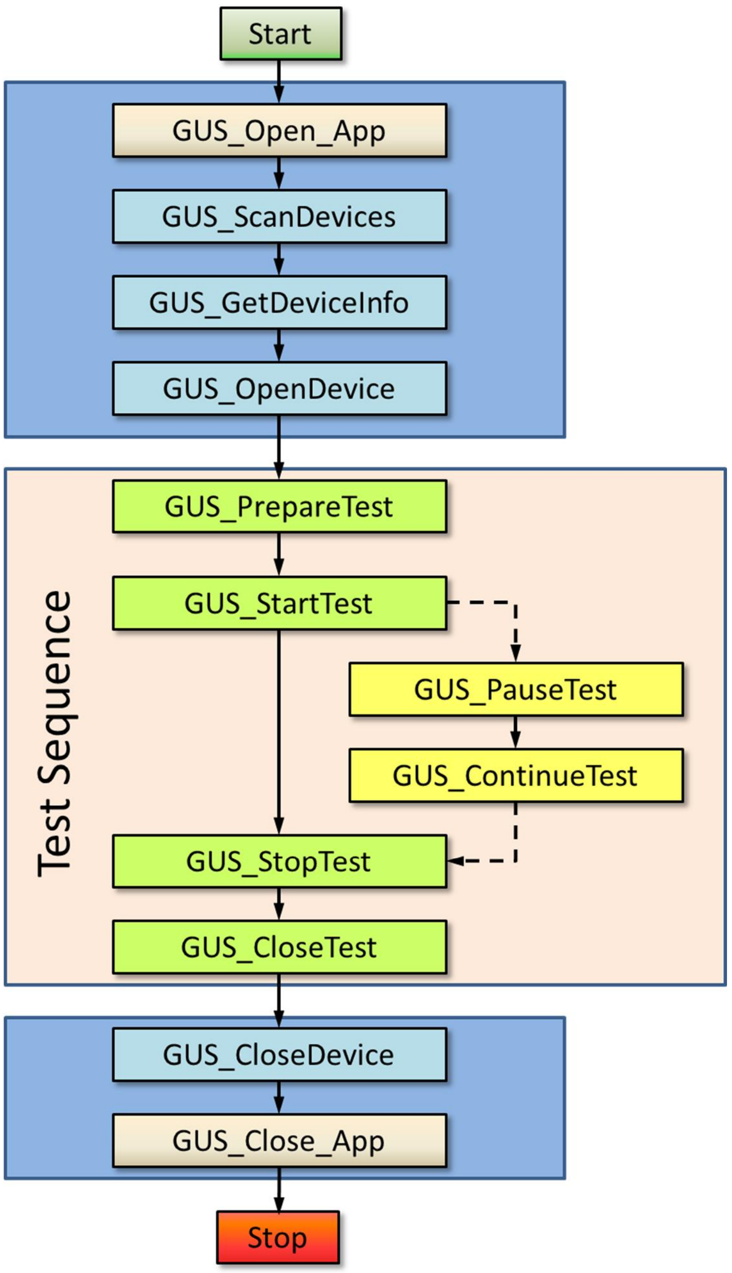


Figure 1: Normal sequence of the test program for one device.

A test starts by searching for the devices connected to the PC on which the Control Program is installed. Devices can be connected directly to the PC through a LAN, USB, RS-232, RS-485 or other standard interface, or just can be hard wired through e.g. a digital I/O card. The application lists the available devices and shows the functions implemented for each device.
The test engineer selects the devices required for the test which, by its selection, connects the device to the application for the test.

Each device has its own test program, project and/or run schedule. The test program for a climatic chamber is the temperature profile which defines the high and low temperatures, time, slopes etc. which could be combined with humidity control, sun simulation or other parameters. Typically the test program is entered by means of the console of the climatic chamber and resides in its own control system.

In the same way the vibration control system is preprogrammed to run a test. The vibration control system has all information about the shaker system to be controlled, all measurement channels and accelerometers to be used during the test, the test profile, data which is stored into the project file. Finally it also contains a run schedule that defines how the test should be run.

While the climatic chamber needs time to reach its initial temperature (and humidity) condition before the real test can start, the vibration control system needs to perform a pre-check test, in order to make sure that the vibration test will run without any problem. So, during the start-up of a combined test (climatic + vibration), the specific need of each system has to be envisaged.

Further equipment that possibly would be connected are power supplies, air or oil pressure, signal generators, actuators or any other device or energy source needed to perform e.g. a functional test. Typically such devices are controlled over relay contacts, in- or outputs. In fact, a very high flexibility of the definition of a “device” in this context is required. Hence the layered structure through which a device will be connected to the HLCP to be developed.

In addition, there will be two modes of operation:

- ***Manual Mode:*** through the user interface it is possible to start, to stop, to pause or to continue a test project of a device. For “simple” devices e.g. the only functions would be to switch them ON or OFF.
- ***Automatic Mode:*** in this mode it will be possible to run through a sequential program that will control the course of actions to perform the complete test, controlling the action of each device (start, stop, wait, pause, continue,…).

In order to avoid confusion between the sequential programs in the devices and the sequential program, that defines the “automatic mode”, we will call the latter the “***script***”.

Figure 2 describes the different layers of the software model. The “command interface” layer is the interface that will be understood by the script (HLCP in automatic mode), the error handling program and the HLCP in manual mode. When the device adheres to the GUS standard command interface, it can be hooked up directly to the HLCP. The “Device Definition” is only required for the devices that cannot have a standard interface like a pump, actuators or just a measurement system that has to monitor e.g. the device under test.

So the development of the control program can be split into two parts: one part being the development of the HLCP. The next part, independent of the HLCP is the Device Definition part. The latter is not mandatory to run the HLCP and can be developed separately at a later point in time.

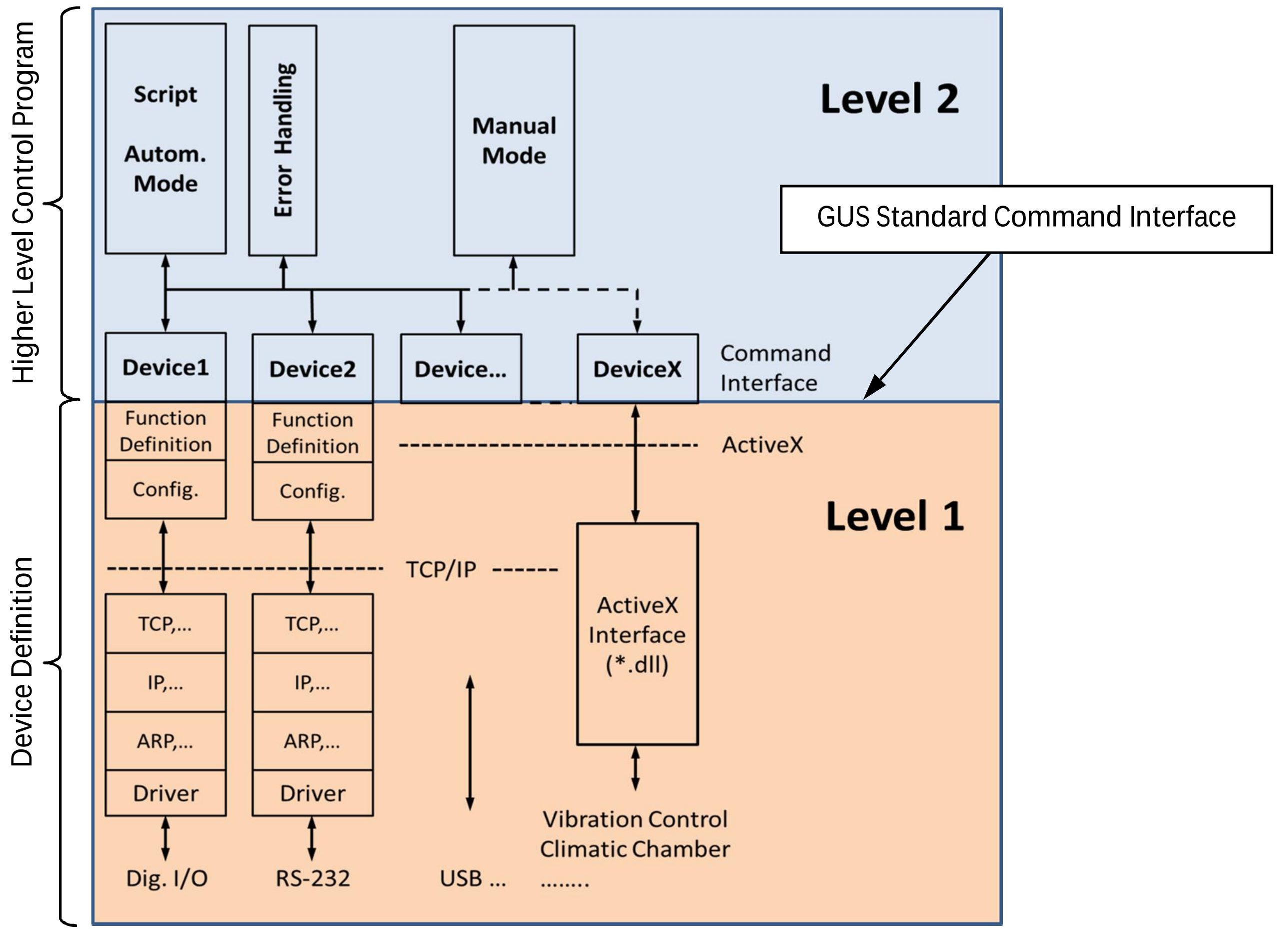


Figure 2: Model of the layered software structure.

In fact, the command interface is nothing else but the commands, defined in table 1.

In case the device already has an ActiveX® interface, compatible with the GUS Standard Commands, the device definition is complete and needs no further development. Then this device can directly plug into the HLCP.

In case the device does not have a compatible interface yet, it must be possible to build a virtual device in two steps. The first step is the definition of the hardware connection or I/O (analog and digital), which is presented as the “Configuration” block in figure 2. The second step is the definition of the actions to be taken for each GUS Standard Command, presented as the “Function Definition” block in figure 2.

The “Driver” integrates the hardware into the operating system. But the “Configuration” layer makes it possible for several devices to share one hardware interface, like a digital I/O or relay card. In many cases a meaningful configuration.

The preferred interface in the development of the control application is the TCP or the transport layer, according to the OSI Seven Layer Model.

However, it cannot be denied that today already, a number of ActiveX interfaces are available on the market and have been implemented in many test installations. Hence it must be possible to connect directly to an ActiveX interface, supplied by the manufacturer of the device one wants to integrate. It is the responsibility of the equipment or “device” supplier to guarantee the compatibility of the ActiveX interface with the GUS standard commands.

6. System Errors

Next to the sequential course of actions, the normal operation during an automatic or manual test, there will be a second permanently active program (a thread) that supervises the condition of each device. As far as possible, for each device for which the status can be checked, the HLCP will decide which action to take, depending on the status of the device and an error script. The error conditions of a device are passed to the HLCP through the GUS Standard Command Interface. When a new device is created, during the definition phase, the device errors conditions have to be defined.

Typically the status of a vibration control system or a climatic chamber can be checked, as in most cases they provide an intelligent interface.

However, some devices do not always offer a feedback about their status. One example could be a signal generator for a functional test that does not offer an interface to interrogate its status.

For the handling of the device error conditions, see below, chapter 12.

7. File Format

All configuration data must be stored into an XML-file. Several files support the application:

- ***Script.xml:*** the script-file that contains the command sequence for the automatic operation mode.
- ***[Device]-error.xml:*** the error-handling file that defines the actions, according to the status of the device. There will be one XML-file for each device.
  [Device] stands for the device name. e.g. shaker-config.xml
- ***[Device]-config.xml:*** the configuration file for each device. This file contains all information, parameters and definitions for one device from the “config” layer up to the “function definition” layer.
- ***log.txt:*** at all times a log file has to be updated with all actions taken by the control application, be it manually or automatically. The format must be .txt
  The log file must also record all interruptions and errors during manual or automatic operation.

In the event of an ActiveX® interface, the manufacturer of the device has to provide the required files (*.dll, drivers, etc.) to guarantee a solution, compatible with the Command Interface layer or the GUS standard commands as described in table 1 and figure 2.

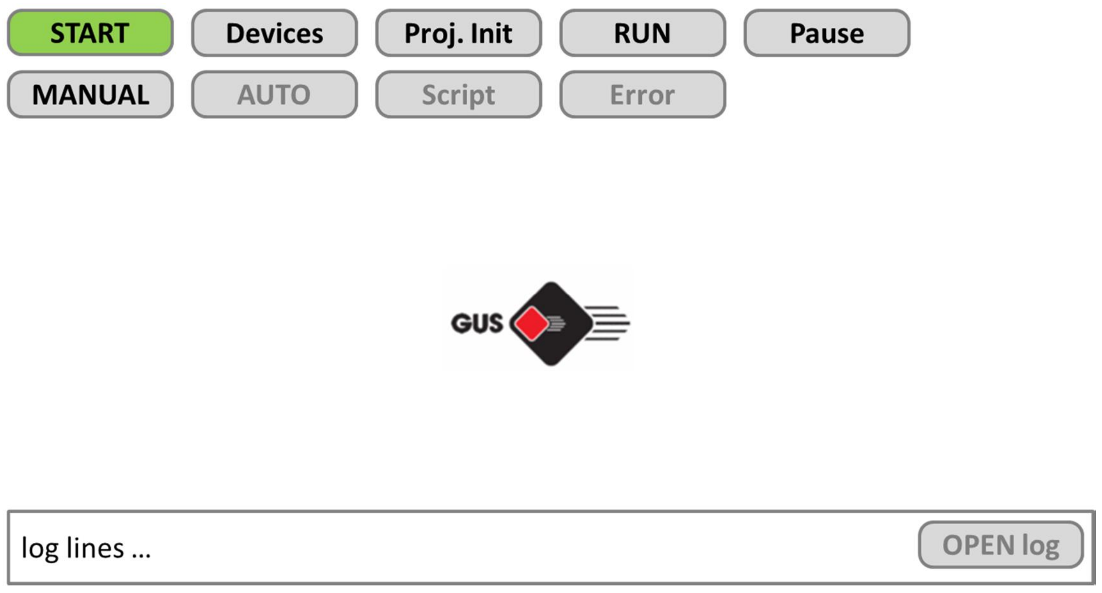
8. Operating System and Development Environment

The minimum supported platform must be Windows® 7 with .NET Framework 4.5. The source code must be written in AINSI C/C++ for reasons of portability.
It is advised that the HLCP will run in a Web Browser window.

Supported local interfaces: USB, RS-232, digital I/O card *(to be defined)*.
Supported network interfaces: TCP/IP and ActiveX.

9. User Interface – Normal (Manual) Operation

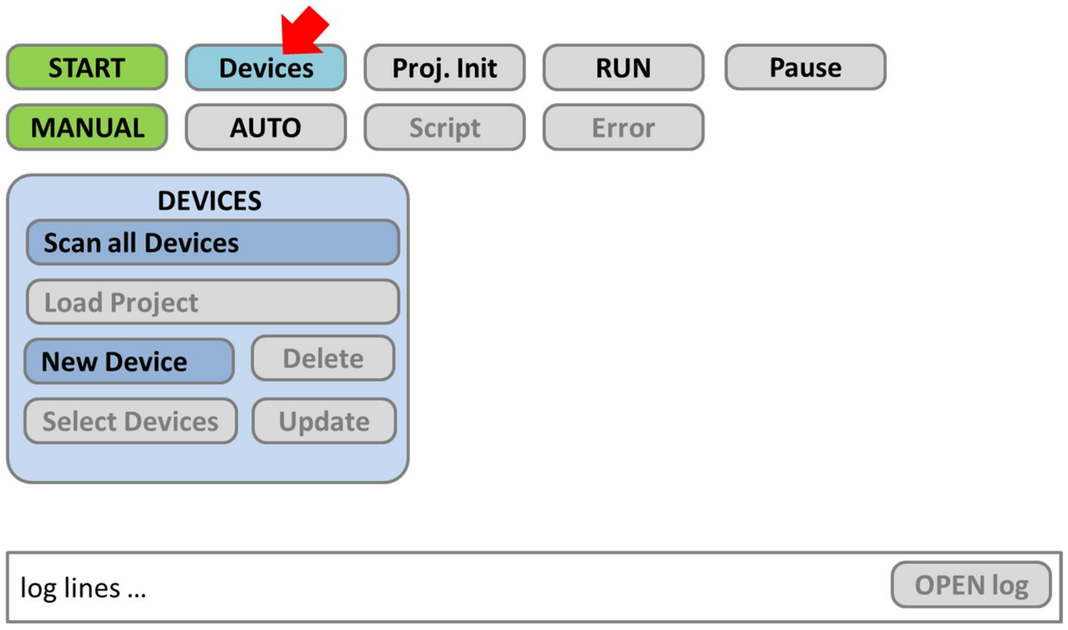
When the HLCP starts, the user interface (UI) as presented in figure 3 will open.
Until the "START" button has been clicked, no further button will be active. The "START" button will effectively launch the "***GUS_Open_App***" command.



log lines ...

Figure 3: Window with the User Interface when the HLCP starts.

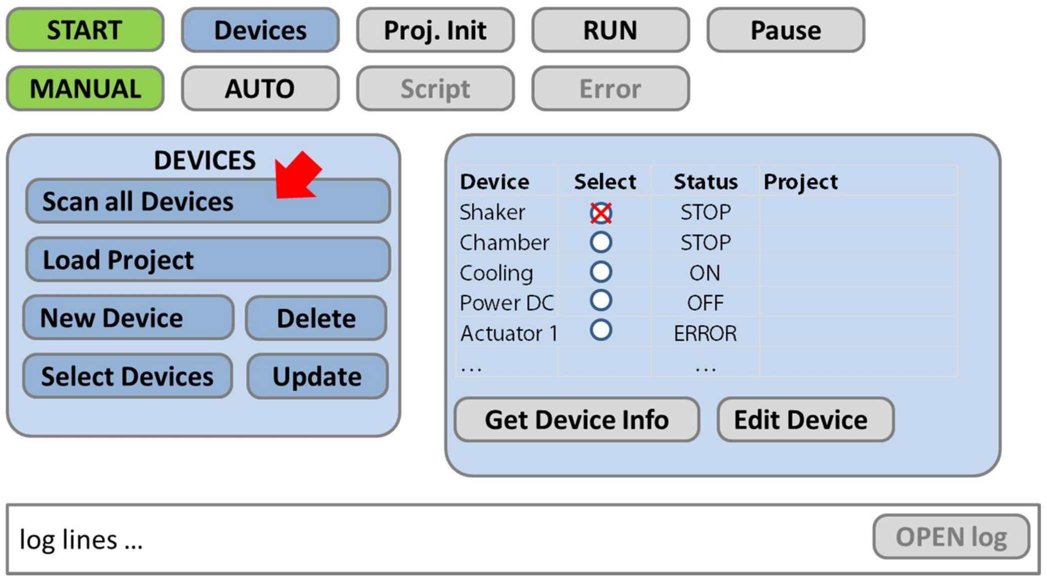
Then the buttons "Devices", "MANUAL" and "AUTO" will become active as shown in figure 4. The button "MANUAL" will turn green, indicating that the HLCP starts in the manual mode. The lower part of the screen is reserved for system messages as they will be logged in one log file. The log file will record all system messages (like error messages), but will also log all actions taken: all buttons clicked, tests started, paused, stopped, …all manual and automatic actions.
Clicking the "Devices" button will open a new window that shows all functions, available to configure or to prepare the required devices to execute a given test. The option to configure a "new device" is given, making it possible to configure even a complex set-up with new (even virtual) devices e.g. for a functional test. It offers the possibility to adapt the HLCP to a specific installation (e.g. a shaker system without remote control) or to prepare a test set-up with devices that are particular for one test.



log lines ...

Figure 4: User Interface after clicking the "START" button.

The normal sequence of operations is to first "scan" all the devices, then to select a device of interest and then make the necessary changes to a given device
- if required. "Scan all Devices" launches the GUS command "***GUS_Scan_Devices***".



Device	Select	Status	Project
Shaker	☒	STOP	
Chamber	○	STOP	
Cooling	○	ON	
Power DC	○	OFF	
Actuator 1	○	ERROR	
…		…	

log lines ...

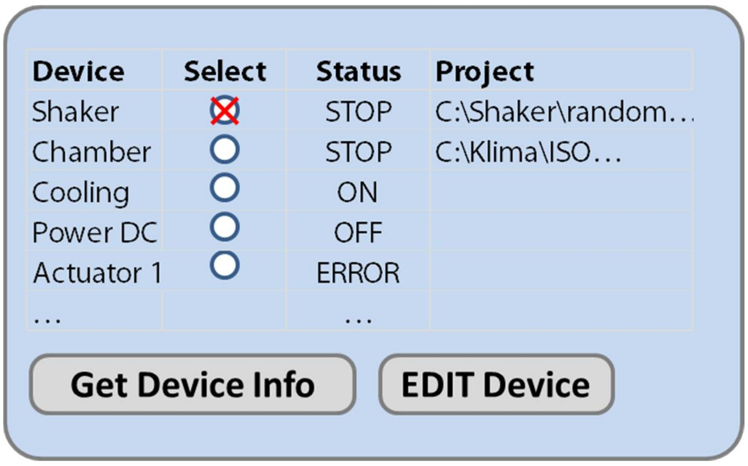
Figure 5: Available functions and new device window that open after clicking "Scan all Devices".

The new window that opens, lists all devices than can be detected, or that have been defined previously. The window gives an overview of the "status" of each device and the project that has been loaded. A device can be selected by clicking on the button in the second column. Only one device at the time can be selected. When a device has been selected, and the user clicks on the select button of another device, the previous device will be de-selected. Selecting one device will make all functions available on the left hand side: including "Load Project", "Select Devices" and "Update".

"Selecting" a device will launch the GUS commands: "***GUS_OpenDevice***" *and* "***GUS_GetDeviceInfo***".

By clicking the button “Load Project”, the Windows standard browser will open. Then the project for the selected device can be searched for and can be selected. Typically these are the sine, random, shock tests etc. for the shaker and the temperature profiles for the climatic chamber. These projects are defined within the device itself and need to be selected before the test starts.

After the project has been loaded, the project name will appear in the right window, in the line of the corresponding device. It is important to note here that the project is NOT started yet and has NOT been loaded in the device itself yet. In fact, the selected project will be loaded and opened in the device, only when the “Proj. Init” button will be clicked.



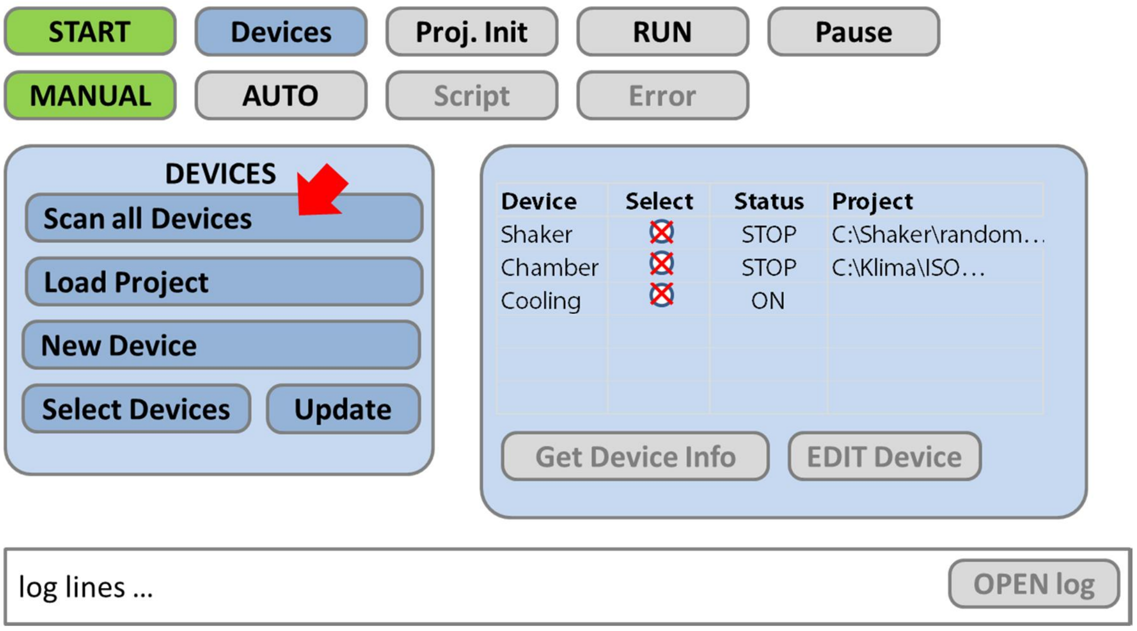
Device	Select	Status	Project
Shaker	☒	STOP	C:\Shaker\random…
Chamber	○	STOP	C:\Klima\ISO…
Cooling	○	ON	
Power DC	○	OFF	
Actuator 1	○	ERROR	
…		…	

Figure 6: List of devices with loaded projects.

When a device has been selected, the buttons “Get Device Info” and “EDIT Device” will become active. The “Get Device Info” will show all info from the device as available through the command “***GUS_GetDeviceInfo***”. The command button “EDIT Device” opens the possibility to edit specific settings or definitions of the respective device, e.g. change the name of the device or e.g. change the run schedule as described further.

The function key “Select Devices” (see figure 5) makes it possible to select multiple devices, needed to execute the environmental test for the DUT (Device under Test or test specimen). The button “Update” will then make the choice final and will update the device list window on the right hand side.

Then the HLCP UI will look like shown in figure 7. The buttons “Get Device Info” and “EDIT Device” become inactive as multiple devices are selected now.



Device	Select	Status	Project
Shaker	☒	STOP	C:\Shaker\random…
Chamber	☒	STOP	C:\Klima\ISO…
Cooling	☒	ON	

log lines ...

Figure 7: UI after the required devices have been selected and the list has been updated.

10. Definition of a New Device – Level 1 of the Control Program

However, it is more than likely that a device has no GUS standard interface or new devices have to be added to the list for a new test. Therefore the option is given to add a new device and to build a GUS standard interface for it through the command button "**New Device**". That button will open a new window with a dialog to configure the new device in three steps as described in figure 8.

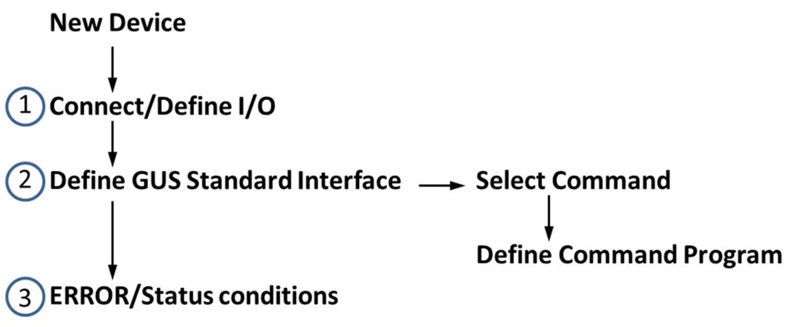


Figure 8: Three level definition and creation of a new device.

The three level process for the creation and definition of a new device corresponds to the hardware/software structure of figure 2. First the new device has been given a name, after which all inputs and outputs have to be defined. This is done by defining all parameters involved. Each parameter has a name (*Paramname* in figure 9) and is linked to an input or output. A parameter also simply can be information (e.g. text) that describes the device and could be recalled by the GUS standard command "***GUS_GetDeviceInfo***". The I/O (Input/Output) can be digital or analog. For analog I/O it can be decided to define a minimum and maximum value, either in EU (Engineering Units) or in Volts. In the event a voltage has to be converted into EU's, a scale designer will be available which makes it easy to calculate the conversion.
The scale designer at first instance only will be available for analog inputs.

The first option is a linear or proportional conversion with/without an offset and the multiplication factor.
As a second option, a polynomial can be defined to convert a voltage into an EU.

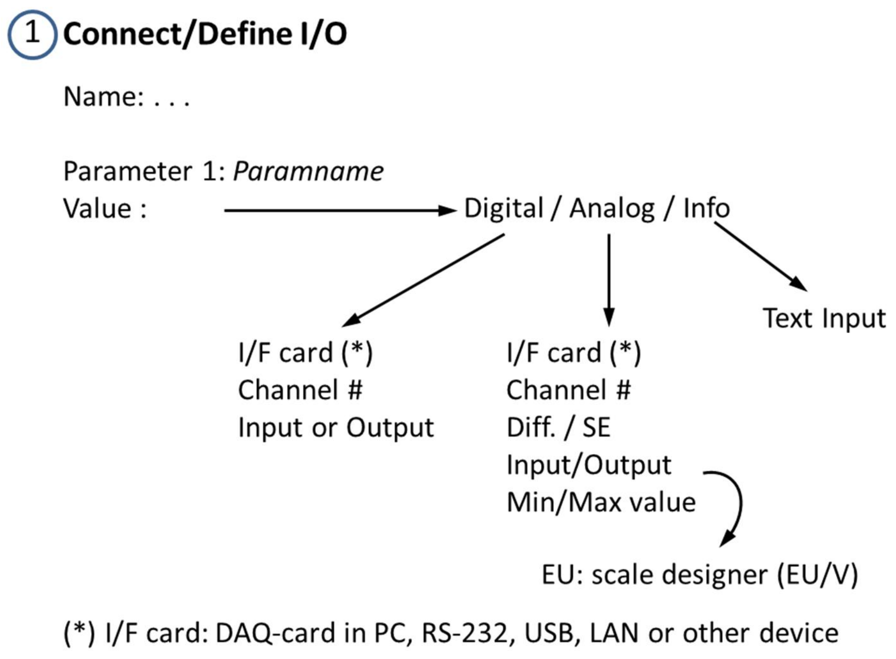


Figure 9: First level of the creation process to define a new device.

For one device it is possible that the I/O is distributed over different measurement cards and different channels, or even is shared with a test device over a LAN network.
An overview is given in figure 9. In such a way it is even possible to design a remote control solution for a shaker or climatic chamber system through a simple relay I/O card and to monitor certain parameters during a test.

Once the I/O has been defined, in the second stage the link between the I/O and the GUS standard commands is built. Possibly for the new device no “project” can be defined or the device cannot be “paused”. Then the device has to return an ERROR to the HLCP in the event e.g. the GUS standard command “***GUS_PauseTest***” is executed for this device.

An interface and overview of the process to build the link between the I/O and the GUS standard commands for a new device is given in figure 10. All GUS standard commands are listed and for each command a run schedule is defined. A run schedule could simply be a write command to set e.g. a digital output to start a pump or a cooling system. Then the run schedule is just two lines (the last line is always “END”). But the link could be a more complex run schedule so that e.g. a cooling system first has to check a number of conditions before it can start.

A run schedule is a sequential command list, acting on the parameter list that has been defined before.

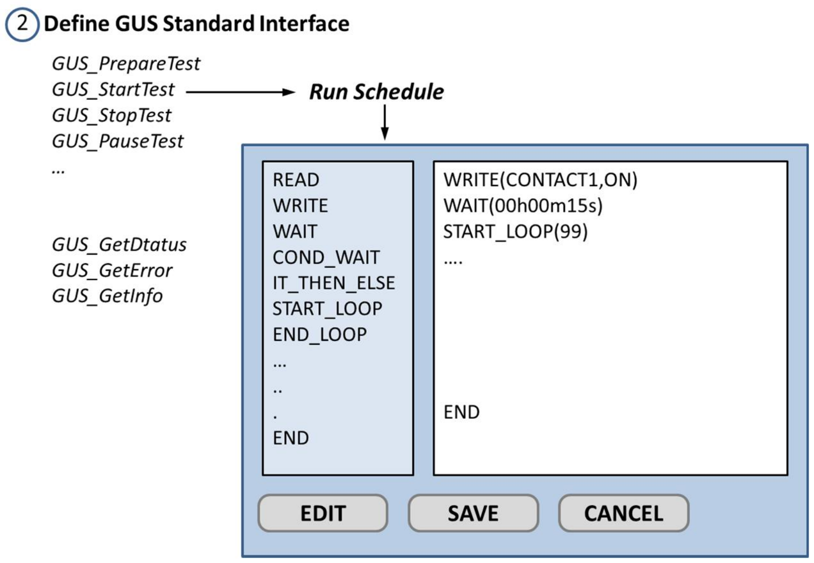


Figure 10: Definition of the run schedule for each GUS standard command.

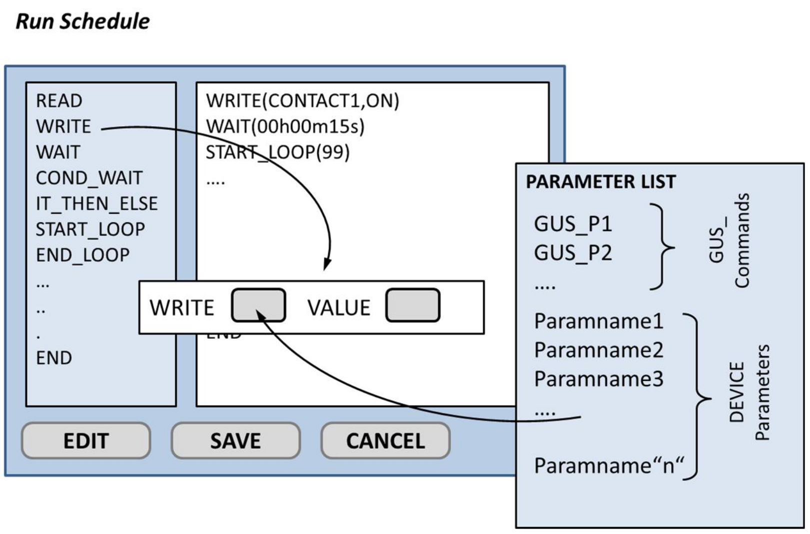


Figure 11: Building the run schedule, using a command list and the device parameters.

The figure 11 suggest a way to build the run schedule for each GUS standard command, using a list of commands that use the parameters, previously defined for the new device.

In the same way, the final step is to define the status conditions of the device. Which are the conditions to decide whether the device is ON, has been STOPPED, is RUNNING, shows an ERROR, etc. ? In order to define the conditions for each status, all possible STATUS definitions are listed. For each status a logical combination of the parameters of the device can be built. Possible parameters must be inputs only, but can be digital or analog. A logical expression can be TRUE or FALSE. Analog values can be < or > or must be within a range (…< x >…). For the input of numerical values it is suggested to use the keyboard of the PC (no touch screen inputs). The value of an analog input is in the EU as defined during the definition process (Volts or EU).

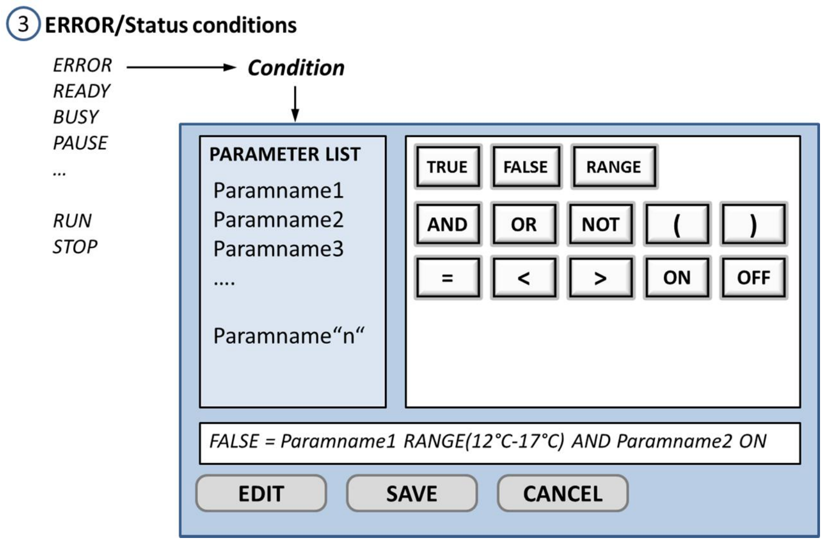


Figure 12: Definition of the status conditions.

A status for which no logical condition has been defined, is not available to the HLCP and will not show up when the GUS standard command “***GUS_GetStatus***” is executed.

11. Running a Test

At this stage the devices, needed to execute the environmental test” with their corresponding projects have been defined. The list of devices (figure 7) only shows the devices that will be needed to execute the environmental test. Each device can be selected and started/stopped/paused …etc. manually. First select the device and then click on the command button at the top of the window.

The button “**Proj. Init**” will launch the GUS command “***GUS_PrepareTest***”. For the selected device, the project will be loaded and the self-check will start. The status of the device will be updated and shown in the right hand window.

The other buttons are self-explaining and will launch the GUS standard commands accordingly:

RUN…	***GUS_StartTest***	The device goes in RUN mode
STOP…	***GUS_StopTest***	The device goes in READY mode, the test goes on hold and is terminated
PAUSE…	***GUS_PauseTest***	The device goes in pause mode the test goes on hold, but NOT terminated
CONTINUE…	***GUS_ContinueTest***	The device goes in run mode and the test is continued
CLOSE…	***GUS_CloseTest***	The loaded test is closed, the device goes in default and can receive the “prepare”

The command buttons become inactive or inactive depending on three conditions:

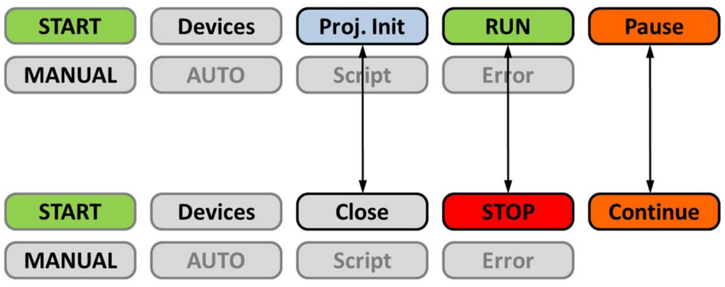
- Depending on the definition of the device, inasmuch as the command is available in the device or not.
  E.g. one cannot pause a cooling device of a shaker.
- Depending on the sequence. One cannot start a test before the device has done its self-check (prepare command). One cannot pause a test when it already has been paused. One cannot close a test before it has been stopped.
  etc. …
- A vibration test can be paused through the run schedule of the shaker. This means that the “continue” command button can become active, also when the pause command was not executed through the GUS standard command. The status of a device is checked continuously and should be taken into account as well in order to make certain command buttons active or inactive. Another example is a device that stopped through an error. Of course this device cannot be paused or stopped through the command buttons.

Note: the buttons “Proj. Init”, “RUN” and “PAUSE” will change function (and color) when they get activated.

Proj. Init will become **CLOSE**

RUN becomes **STOP**

and **PAUSE** will turn into **CONTINUE**:



The position of the buttons remains unchanged.

When a test needs to continue after a pause, click “Continue” and the test will resume. The command button becomes “Pause” again. In the same way, when a test is stopped (after clicking “STOP”), the command button becomes “RUN” again. When a test is finished, the project(s) need to be closed. By clicking “Close”, the project in the (slected) device will close and the command button will show the function “Proj. Init” again.

It is important to mention that the function of these command buttons reflect the status of the selected device. So, by selecting a device, these buttons can mean (and show) different functions. E.g. the shaker is running, by selecting the device “shaker”, the “RUN” button will show “STOP”. However, when at the same time the climatic chamber is “paused”, then by selecting the climatic chamber, the “Pause” button will show “Continue”.

In automatic mode the user cannot select a device for a manual operation. The command buttons will operate on the script instead. The automatic mode can be stopped, paused, and continued, etc. but the command buttons will not have any effect on a device. In order to execute a command on a (selected) device, the operator has to go in “manual” mode first.

a. Automatic mode

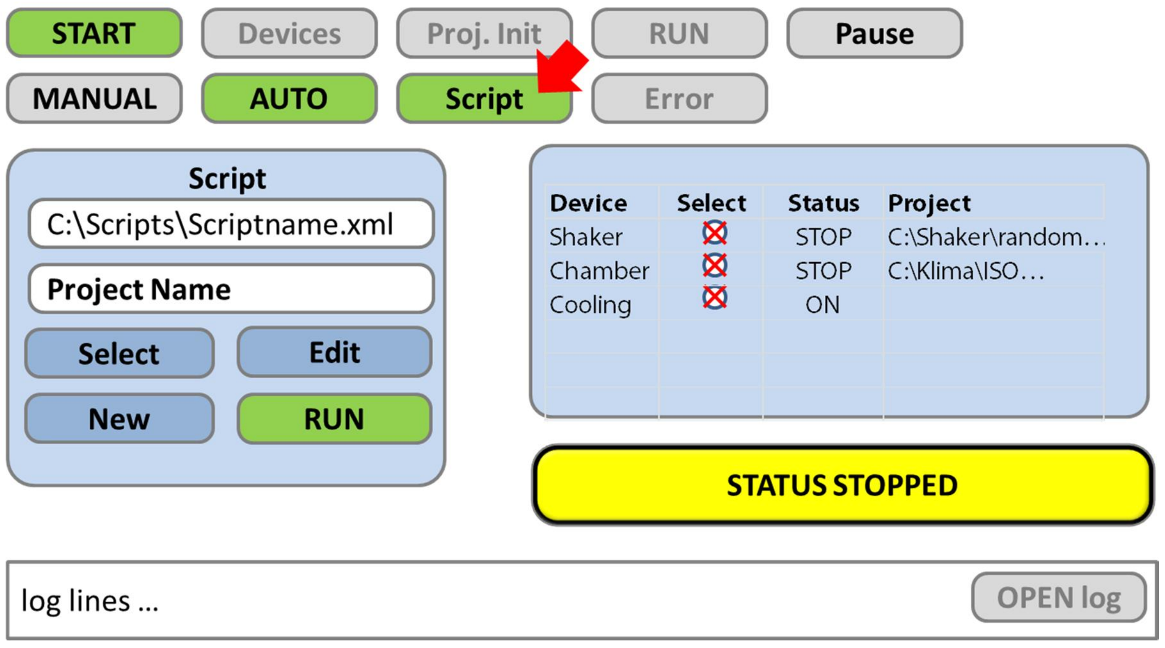
In order to avoid a number of manipulations to run through the start-up sequence of all devices, the HLCP can be switched into "Automatic Mode" by just clicking the command button "AUTO". Then the HLCP will run through a script (see chapter 4.) that has been defined before.

The script also will make it possible to synchronize certain steps of the projects or run schedules of the different devices. Therefore it will be possible to define certain conditions on which a project or run schedule can "wait", "start", "continue", etc.
The script makes it possible to prepare complex tests with different devices so they can run in a full automatic mode.

When the button "Auto" has been clicked, the user interface will appear as shown in figure 13. It shows the options for the script and will show the list of the selected devices.

In the script window the function buttons "Select", "Edit", "New" and "RUN" will appear. The "RUN" button will turn green after a script has been selected. In order to select a script: click "Select". Then the standard Windows® browser will open that makes it possible to select a script that has been saved.
A selected script will show up in the two white info fields. The first field contains the path where the script has been saved on disk. The field below will display the name of the project. The project name has been entered during the script definition.

Once a script has been selected, the test in automatic mode can be started by clicking the "RUN" button. Then a status window (can become an active button) will appear below the device list. The status of each device will be updated continuously in the device list window.



Device	Select	Status	Project
Shaker	☒	STOP	C:\Shaker\random…
Chamber	☒	STOP	C:\Klima\ISO…
Cooling	☒	ON	

```
log lines ...
```

Figure 13: The user interface in automatic mode.

But the status of the script will show the progress of the script, as also during a test, the script can be interrupted by: an ERROR, a WAIT command, a conditional PAUSE or STOP as discussed below. The log window will show the actual command and line number of the script that is executed.

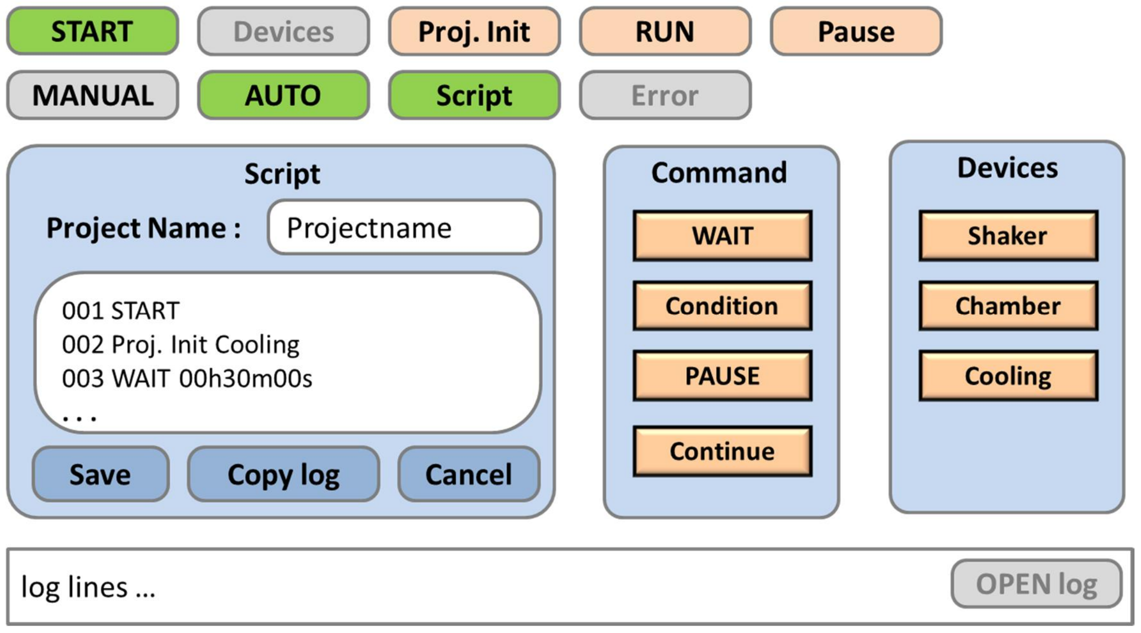
b. Definition of a Script

A script starts with the "START" command (first line) and ends with the "END" command (last line). A script can be built in two ways:

- Lines can be copied from a log file.
- Commands are entered as a sequential list by means of command buttons, similar as a run schedule is entered for a device during its definition.

When the "New" button in the script window (Figure 13) is clicked, the user interface to build a script is opened, as shown in figure 14.

Three new windows are opened: the script window, the extra command window and the window that lists all selected devices.
Note: before a script can be built, it is mandatory that first all devices have been selected in the manual mode.
The standard commands, as available in the manual mode will become available when a device has been selected for the script. The commands in the command list window only apply to the script and have NO relationship with the devices.



```
001 START
002 Proj. Init Cooling
003 WAIT 00h30m00s
. . .
```

```
log lines ...
```

Figure 14: User interface to define a new script.

How a script is built:
Always a project name must be entered to identify the script.

1. Select a device from the device list.
2. Then the commands, available for that device will become available. This will be shown by the change of color of the corresponding buttons. These are the same buttons as available in the manual mode.
3. Then click the command button as needed.
4. Then the new line is entered into the script.

Insert and delete lines
Lines can be deleted or new lines can be created by means of the “Delete” and “Enter” buttons of the keyboard as usual in any text editor like Notepad® or WordPad®. The line numbers of the script will be updated automatically.

Using a log file
As a script can be edited like a simple text editor, lines can be copy-pasted from a log file. A log file is recorded during the manual operation of a test. The lines of that log file can be copied and the pasted into a new script. When the “Copy log” button is clicked, the standard Windows® browser is opened. A log file can be selected. The selected log file open in a text editor (Notepad is suggested). Then, with the mouse, lines can be selected, copied and pasted into a script.

After the copy-paste of the log lines, in the script lines still can be deleted, new lines can be added etc.

The extra commands and how to link projects or run schedules of the devices
The **WAIT** command: clicking this command button will open a small window, by which a fixed time to wait can be entered. When the HLCP encounters this command, it will simply wait the defined time before the next script line will be executed.

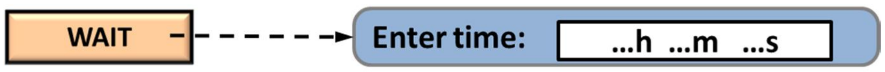


Figure 15: Enter time to wait in the script.

The **CONDITION** command line in the script links the status of different devices to each other. The continuation of the script will WAIT until a defined status of a device is fulfilled. E.g. the cooling system of the shaker will be switched OFF, only after and under the condition that the shaker has been switched OFF. Click the command button “Condition”. Then a device can be selected from the device list. After selection of the device, a new window will open that lists the possible status conditions of that device. By selecting and confirming the status selection, the new line will be entered into the script.

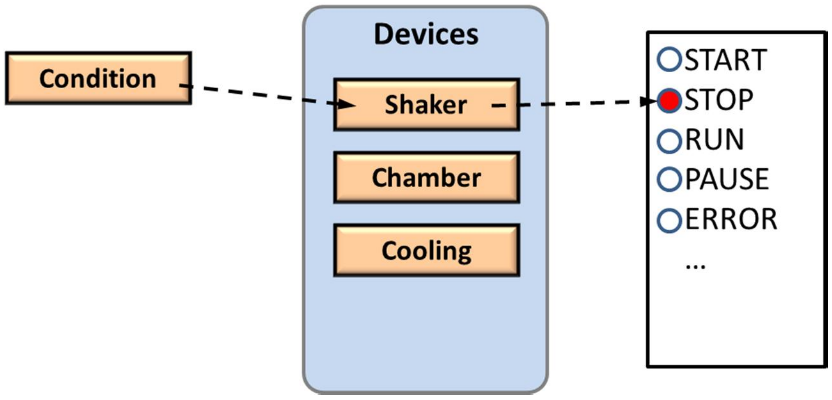


Figure 16: User interface to enter the Conditional wait command in the script.

The example mentioned above will show up in the script like:

```
…
007  Condition Shaker STOP
008  Cooling OFF
009  END
```

The **PAUSE** command in the script will invoke the interaction of the test technician or engineer. When the HLCP encounters the PAUSE command in the script, the status field or button of the script becomes active and shows following message:

Figure 17: The script status button becomes active when the script is paused.

After clicking the button (confirming the message), the HLCP will continue to run through the script. During the pause of the script, all devices will remain in the run mode and will continue to execute their projects. The pause command in the script has NO influence on the devices. Of course the pause command can be combined with the condition command. When the pause command follows the condition of the shaker to be paused, it offers the possibility of a manual interaction during a test, e.g. with stepped sine to save certain intermediate results. Another possibility is to “pause” a certain device in the script, and then enter the pause command into the script. So again, a manual interaction is made possible.

The **Continue** script command acts in a similar way like the **Condition** command. A device could be in the PAUSE status, waiting to continue. …

```
…
005  Shaker PAUSE
006  …
007  Chamber RUN
008  Continue Shaker on condition Chamber RUN
009  …
…
```

In the example above, the project of the vibration test will continue, once the status of the climatic chamber is confirmed to be RUN. So, in order to program the **Continue** command into a script line, the user interface would look like figure 18.
Click on the **Continue** command button for the script. Select the device that has to continue, and then select on which condition of the second device the first one has to continue.

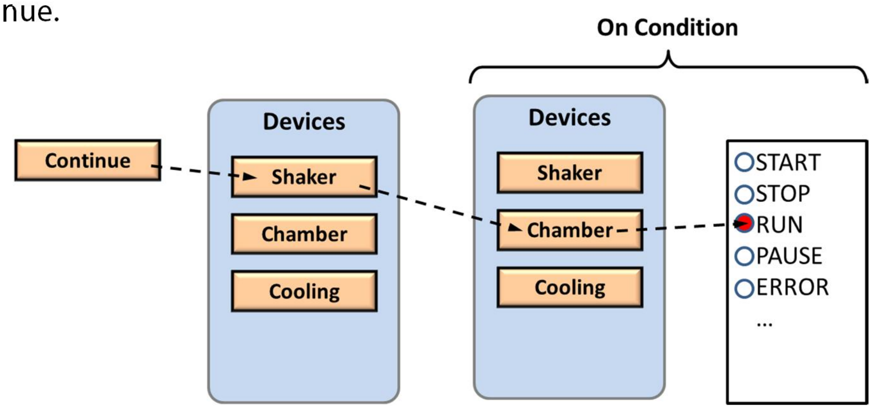


Figure 18: User interface to define the conditional "continue" of a device.

c. The log file

The window, at the lower part of the user interface, is reserved for viewing the log file. Depending on the size of the window, one or more lines can be viewed. The window is updated when a new command is executed or when the HLCP logs a certain action or alarm. When clicking on the "OPEN log" button, the log will be opened in a new window, showing the whole file. Then the user can scroll through the log file as in a similar typical application like Notepad.

12. Error Handling

As discussed, the error conditions of a device are defined during the creation of that device (chapter 8.b, figure 12). These conditions still can be edited afterwards.
The error status of a device does not introduce any action of the HLCP yet.
To make sure that proper action is taken as a consequence of a device error status, one has to tell the HLCP which action to take. As described in chapter 5, a separate thread will continuously check the device conditions and take action as defined through the error script.

<u>The error handling or thread is active in both manual and automatic mode.</u>

Following figure 19 gives an example of the error handling of the shaker, depending on the error condition of the climatic chamber or the cooling system.

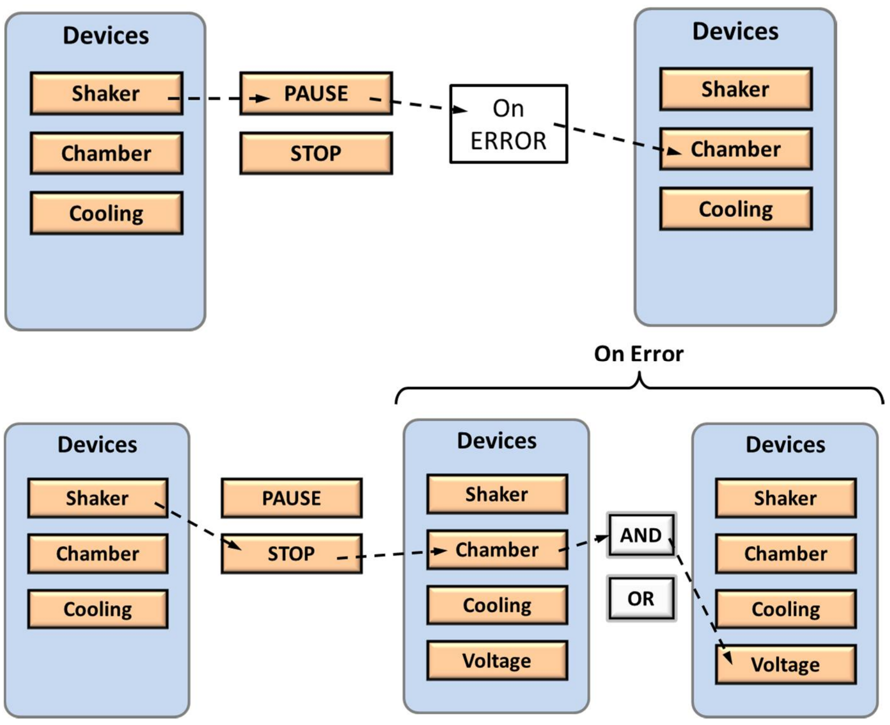


Figure19: Error handling of the error conditions of the climatic chamber or the cooling system.

For obvious reasons the shaker should be stopped when the cooling system fails. However, when an error occurs in the climatic chamber, maybe the test can continue after the problem has been solved. So it could be advisable just to “pause” the vibration test and wait for the intervention of the test technician or engineer.

At another instance, it might be advisable to stop the shaker when the climatic chamber has the error status and also a power supply (the device “Voltage” in figure 19) shows an error. Then both conditions can be combined with the “**AND**” operator. In order to build a multi error conditional action, both the **AND** and **OR** operators must be included in the user interface. There can be more than one **AND** and **OR** operator in one line of the error script.

The lines in the error script could look like:

```
…
009  Shaker PAUSE on Chamber
010  Shaker STOP on ERROR Cooling
011  Shaker STOP on Chamber AND Voltage
…
```

When an error occurs in manual mode, the test can continue once the problem has been solved. The command buttons as described in paragraph 8: “Proj. Init”, “RUN”, “STOP”, etc. remain active.

When an error occurs in automatic mode, the HLCP will switch to the manual mode. However, at any time, when the problem has been solved, it must be possible to continue the test script where it has been interrupted.

13. DAQ I/O cards to be supported

Required specifications:

Analog Inputs:

SE or differential.
Range +/- 10 Volt min. (protected to 100 V/DC).
1 MΩ min. impedance.

Analog Output:

SE (Single Ended).
Range +/- 10 Volt min.
Output current 50 mA min.

Digital Inputs:

Min. 8 channels.
Isolated up to 600 V_{RMS}/AC min.
Max. input range 24 V/DC, protected.
Low input: 0-1 V / high input: 3-24 V.
Input impedance ≥ 2kΩ.

Digital Outputs:

Min. 8 channels, non-latching relays.
Relay contacts: min. DC 24 V @1A / AC 125V @0.5A.
NO (Normaly Open) contacts.
Operations: min. 3x 10^5

Interface:

PCI

Terminals:

Terminal board for DIN-rail mounting (Schienenmontage).

References:

Advantech PCI-1761.
ADLink PCI-7250.
NI PCI-6520.