State Machine – GUS Standard Interface – Comments

Introduction

Following information relates to a higher level program under development to control a climatic test with a vibration test. The basis of operation is the standard, developed by the GUS (Gesellschaft für Umweltsimulation e.V.) in Germany that defines the state of the devices involved and the commands that acts on these devices. Following is the list of states.

State	Name	Format	Description
9	Device Closed	String	The communication with the device has not been established yet
0	Device Open	String	The communication with the device has been established
1	Ready	String	The project file has been loaded and the pre-test was successful, the device is waiting for the start command to start the test
2	PreTest Running	String	The device is loading the project file and is executing the pre-test
3	Running	String	The test is running
4	Finished	String	The test has been finished, but the project file is still open
5	Pause	String	The test has been paused
6	Busy	String	Will be ignored by the higher level control program. This state is not uniquely defined yet.
-1	Error	String	An error occurred during operation. The project file is still open
-2	ProjLoadFailed	String	Loading the project file failed

Compared to the definition of the standard 2.0, 3 states are suggested and will be implemented in the higher level control program:

9 and 0: “Device Closed” and “Device Open” respectively. These states are required to identify every state of the devices involved.
-2: “ProjLoadFailed” to identify how the device comes out of the “PreTest Running” state when something went wrong in this state. This will be discussed below in detail.

Normal Operation

The command “GUS_Open_App” will load the driver. The parameter of the command is the name of the driver, as registered by the Windows® operating system.
The response must be a string: “ACK: device serial/version number”. The device(s) that communicates by means of the selected driver is in the state “Closed” 9.

The App is closed again by means of the “GUS_CloseApp” command. For this command, actually no response from the device is expected.

By means of the “GUS_OpenDevice”, the device to which the communication has to be established is selected and the communication is started. The parameter of the command is the device number or identifier. The device switches to the “Open” 0 state.

The “GUS_PrepareTest” will initiate two actions: the device has to load the project test file, which has been predefined in the device (typically all test parameters, the test profile and the run schedule that define a test) and then the device has to execute the self-check and pre-test. The parameter of the command “GUS_prepareTest” is the name of the project test file and its path.

After the pre-test has been finished successfully, the device will go to the “READY” 1 state.

The “GUS_StartTest” will start the test itself. During the execution of the test, the device will remain in the “RUNNING” 3 state. The test can be paused, resumed or stopped by means of the “GUS_PauseTest”, “GUS_ContinueTest” and the “GUS_StopTest” respectively.

When the test has been finished or has been stopped, the state changes to “FINISHED” 4.

For all commands (except for GUS_CloseApp”) the response “ACK” from the device is expected. This confirms that the device acknowledges the command for execution. It does not confirm that the execution of the command was successful! The success of the command execution is checked through the state condition of the device.

GUS-StopTest

At any moment in time, one must be able to give a STOP command. The outcome of the STOP command depends on the actual state. Basically, when a test is running (in READY, RUNNING or PAUSE) and a STOP command is given, the state will change to FINISHED. The reason being, that the project file of the test is still open. The “PreTest Running” state is a special case. In fact, when the open device receives the command “GUS_PrepareTest”, two actions are initiated:

- First the device has to open the project test file.
- Then the device has to start and to run the self-check/pre-test.

The state “PreTest Running” 2 does not specify exactly in which action the device is involved (loading the project file or executing the self-check/pre-test?). When a device is in the “PreTestRunning” 2 state, a STOP command will cause the device to go into an error state. When the self-check/pre-test was running and the device receives the STOP command, it will go into the ERROR state -1. Then the project file is still loaded / open.

In the other case, where the device was opening the project file (trying to load the project test file), and the device receives a STOP command, it will stop loading the project file and will change to the “ProjLoadFailed” -2 state. The project test file has not been opened and the next STOP command will bring the device back to the “Device Open” 0 state.

The above is also valid in the normal operating mode. When the project test file is corrupt, the file is not available or the path is not OK, the device will move to the “ProjLoadFailed” -2 state. In the event the self-check/pre-test fails, the device will go to the “Error” state -1. In the latter case the project test file is still open.

State 4 : “FINISHED”

The device comes to the state “FINISHED” 4 after the test has been finished under normal operating conditions, without passing through any error state. Or the device will come to the “FINISHED” state after the “GUS_StopTest” command, coming from the “ERROR” -1 state, the “READY” 1 state, the “RUNNING” 3 state or the “PAUSE” 5 state. In the “FINISHED” 4 state the project test file is still loaded in the device. The state of the project test file is still open! From the “FINISHED” 4 state one can close the file and the device returns to the “OPEN” 0 state. Or one can start the test again by means of the “GUS_StartTest” command. In fact, then the device will run through the self-check/pre-test again, which is the normal behavior of a vibration control system e.g.

State -1 : “Error”

At an time, a device can go into the “Error” -1 state. The only way to get out of the error state is by means of the “GUS_StopTest” command. The “GUS_StopTest” command will make the device to go to the “Finished” 4 state. The project test file is still loaded (the file is still open).

Command not matching a state

In the event a command to a device is given, different from one of the commands for a given state as described in the document “State Machine”, the state of the device will not change.
e.g. when the device is in the “READY” 1 state and the command “GUS_PauseTest” is given, the device must respond with an error: “ERR” instead of “ACK”. The device remains in the actual state “READY” 1. The actual implementation of the higher level control program will read every “ACK” string as a positive acknowledgment of the given command. Any other string will be recognized as being an error, meaning the given command was not valid for the actual state of the device. The device will not change its actual state.

State 6 : “Busy”

In the standard, the state “Busy” 6 was introduced for any event where the device was “busy” doing something, not immediately related to an actual state. E.g. opening a file, writing data to disk, is busy with a transition,… This state has not been defined as a “unique” state, and as it has been interpreted, does not relate to a unique condition of a device. Hence, the busy state will be ignored by the higher level control program. It is not recommended to implement this state in the device, until this state has been defined uniquely. Then the state machine will be updated accordingly.

At this time, when a “Busy” 6 state would occur, the potential risk is that the device remains in the “Busy” 6 state, the device “hangs”, and cannot be recovered.

Synchronization of a Climatic Test with a Vibration Test

In the actual definition of the standard, both devices (the climatic chamber and the vibration controller) have loaded a predefined test project. Both tests (vibration and climatic) are started when both devices arrive into the “Ready” 0 state. Then both tests run independently from each other.

For more complex tests (with different temperatures and humidity levels and different vibration tests/profiles) the synchronization between both devices will be defined by an event driven run schedule.

Minimum Requirements

At first instance, not all commands of the GUS standard interface have to be implemented to guarantee the automatic operation of a combined test. Following list gives an overview:

	Implemented	Minimum Requirement	Expected Response	
GUS_Open_App (App=Treiber/DLL ...)		Yes	ACK: number	string
GUS_Scan_Devices		No		xml
GUS_GetDeviceInfo		No		xsd
GUS_OpenDevice (Device=Objekt)		Yes	ACK	string
GUS_CloseDevice (Device=Objekt)		Yes	ACK	string
GUS_CloseApp		Yes		void
GUS_PrepareTest		Yes	ACK	string
GUS_StartTest		Yes	ACK	string
GUS_StopTest		Yes	ACK	string
GUS_PauseTest		Yes	ACK	string
GUS_ContinueTest		Yes	ACK	string
GUS_CloseTest		Yes	ACK	string
GUS_GetStatus		Yes	-2 … 9	string
GUS_GetError		No		xml
GUS_GetInfo_"Gerätetyp/nummer"		No		xml

The commands “GUS_Scan_Devices” and “GUS_GetError” will be implemented in the higher level control program in a later stage.

The commands “GUS_GetDeviceInfo” and GUS_GetInfo” will need further discussion in the work group of the GUS.