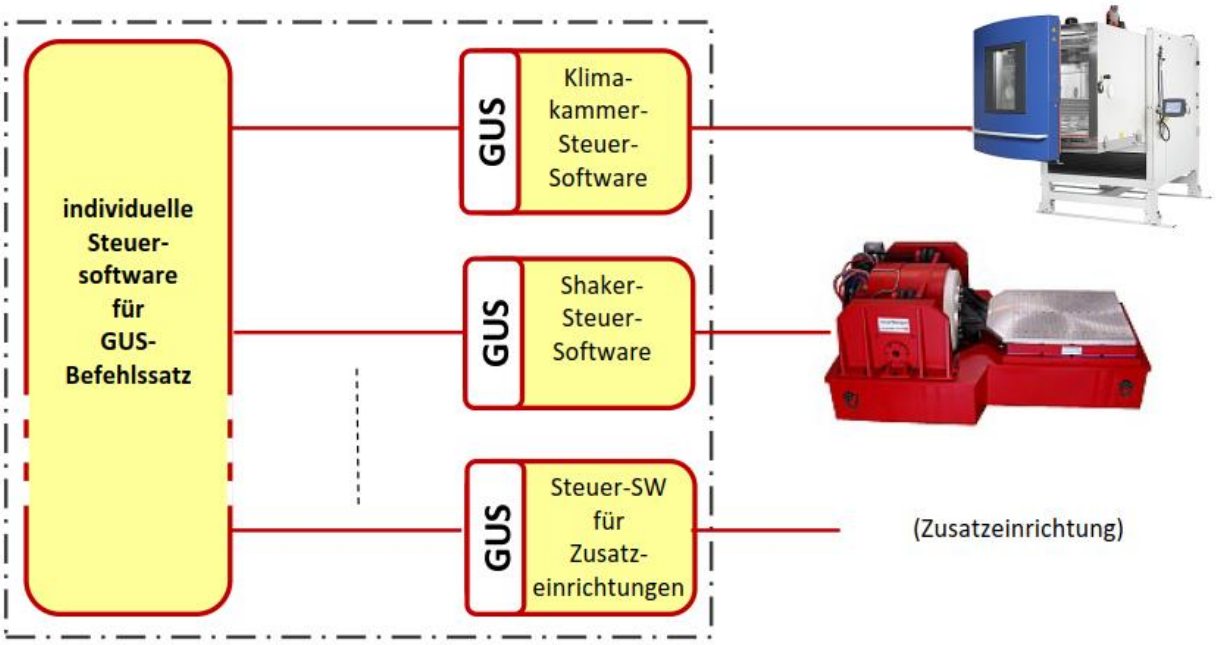


GUS-Standardschnittstelle Version 2.0

Softwareschnittstelle
für Kombinationsanlagen in der Umweltsimulation

Für kombinierte Prüfstände und Fertigungseinrichtungen, bei denen beispielsweise Temperatur und Feuchtigkeit zusammen mit einer Schwingungsregelung ein Profil durchlaufen müssen, gibt es eine Standard-Softwareschnittstelle.

Dieses Dokument beinhaltet die Schnittstellendefinition des GUS-Arbeitskreises "Standardschnittstellen für Kombinationsanlagen in der Umweltprüfung"

Stand: 16.04.2019 15:56:00

Schnittstellenversion: 2.0
Dokumentenversion: 1.01
Autoren: Josef Lenz, Ludwig Liedl

Inhalt

1 Konzept

1.1 Warum eine Standardschnittstelle?

Für kombinierte Prüfstände und Fertigungseinrichtungen, bei denen beispielsweise Temperatur und Feuchtigkeit zusammen mit einer Schwingungsregelung ein Profil durchlaufen müssen, war vorher keine Standardschnittstelle definiert. Es existierte weder ein Hardwarestandard, welcher Steckverbinder, Signalpegel oder Hardwareprotokolle definiert, noch ein Softwarestandard, um die verschiedenen, unabhängigen Regelsysteme gemeinsam anzusteuern und zu koordinieren. Jede Kombinationsanlage war eine individuelle Lösung mit folgenden Nachteilen:

- Kostenintensiv, da Einzellösung
- Schwer erweiterbar, muss einzeln gepflegt werden
- Unflexibel bei Anlagenkonfiguration
- Herstellerabhängig
- Bei Systemwechsel nicht kompatibel

Ein Standard für kombinierte Anlagen in der Umweltsimulation garantiert dem Anwender eine Flexibilität in der Anlagenkonfiguration sowie eine Kostenreduktion und die Sicherung langjähriger Investitionen.

1.2 Anwendungsbeispiele

- Temperaturzyklus mit Vibrationsprüfung
  Während Ablauf eines Temperaturprofils wird zyklisch eine Vibrationsbelastung an/ausgeschaltet. Bei Bedarf werden zusätzlich externe Schaltfunktionen angesteuert (Messfunktionen, Prüflingsansteuerung, etc.).
- "Wochenendabschaltung"
  Wenn bei Dauerprüfungen ein Teilsystem ausfällt, werden die anderen dadurch ebenfalls gestoppt. Vorteile: Energieeinsparung, definierter Testabbruch, Prüfling wird nicht unnötig belastet etc.

1.3 Der GUS-Arbeitskreis "Standardschnittstellen"

Das Ziel des Arbeitskreises ist es, eine Softwareschnittstelle für kombinierte Prüfanlagen, bestehend aus Klimakammer, Schwingprüfanlage und Zusatzgeräten zu definieren, mit folgenden Vorteilen für die Anwender:

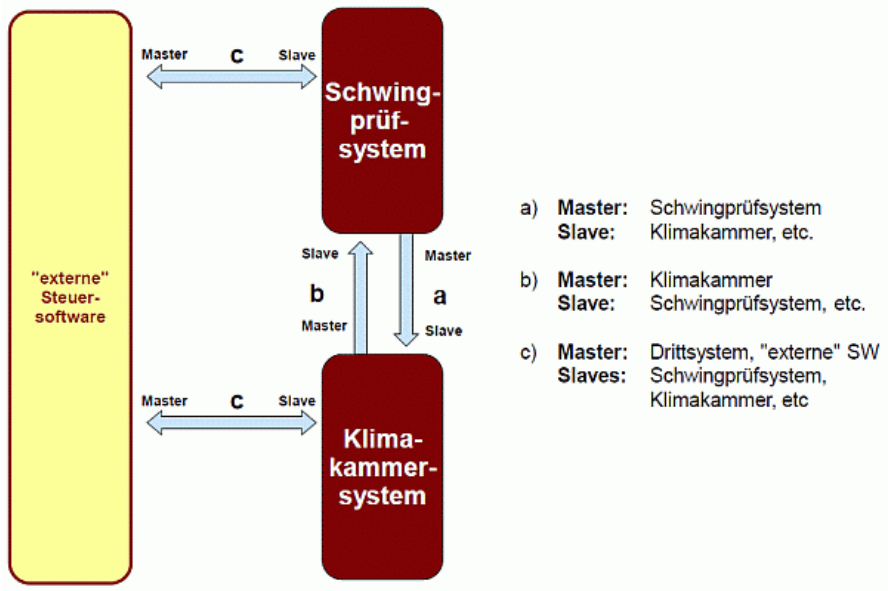
- Flexibilität in der Anlagenkonfiguration
- Kostenreduktion und Investitionsschutz
- Hersteller- und Hardware-unabhängig
- Einfache Programmerstellung in jeder SW-Umgebung (Visual Basic, C, Pascal, LabView, Matlab ...)
- Ansteuerung von Zusatzeinrichtungen möglich

Es soll eine Softwarelösung angestrebt werden, keine Hardwarelösung. Hierbei sollen auch Drittsysteme (als weitere Slaves) mit einbezogen werden können, als Beispiel wurde genannt "Schaltsignal für (Fahrzeug-) Aggregate". Die zu standardisierende Schnittstelle soll sowohl die Betriebsart "Schwingregelsoftware und Klimakammer-Software auf demselben PC", als auch die Betriebsart "beide Systeme auf zwei verschiedenen PCs" abdecken.

1.4 Schnittstellenkonzept

Das Konzept sieht grundsätzlich einen Master/Slave-Betrieb vor, mit folgenden Kombinationsmöglichkeiten:

- Master: Schwingregelung, Slave: Klimakammer
- Master: Klimakammer, Slave: Schwingregelung
- Master: Drittsystem, Slaves: Schwingregelung, Klimakammer



1.5 Standardisierungstiefe und Implementierung

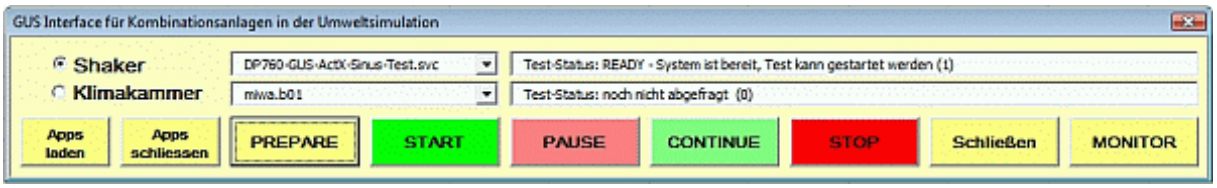
Die Standardisierung erfolgt auf der "Test-Ebene", die Befehlsebene verbleibt in den einzelnen Systemen. D.h. die gegenseitige Steuerung erfolgt über das Aufrufen und Steuern von vordefinierten Tests, mit Möglichkeiten zur gegenseitigen Rückmeldung. Die detaillierte Befehlsprogrammierung inkl. Parametereinstellung erfolgt dabei in den einzelnen Systemen selbst.

Da die Systemkommunikation nicht auf Parameterebene durchgeführt wird, sondern auf Testebene, ist eine direkte Ansteuerung z.B. der Klimakammer auf Netzwerkebene (TCP/IP über Ethernet oder RS232) nicht erforderlich, sondern es ist ausreichend, wenn die jeweilige Bediensoftware untereinander bzw. mit einer externen Steuersoftware kommuniziert.

Die Befehlsdefinition, d.h. die Beschreibung der einzelnen Kommandos und deren Wirkung erfolgen systemunabhängig und sind nicht auf eine bestimmte Systemtechnik (z.B. Windows-PC) festgelegt. Damit ist sichergestellt, dass die auf dieser Ebene definierten Ergebnisse zukunftssicher sind.

Da die ActiveX-Technologie von Microsoft sehr verbreitet ist und auch relativ einfach in der Anwendung wird in der praktischen Umsetzung dieser Technik der Vorzug gegeben. Für Beispielsoftware und Systemtests wird vom Arbeitskreis deshalb ActiveX-Technik verwendet.

1.6 Beispielprogramm:



Der Arbeitskreis stellt ein funktionsfähiges Beispielprogramm unter MSExcel™/VB zur Verfügung. Es kann leicht für eigene Einsatzzwecke geändert oder erweitert werden, und so die Grundlage für eigene Programme sein. Außerdem kann es zur Überprüfung der grundsätzlichen GUS-kompatibilität der Schnittstelle eines Systems verwendet werden.

1.7 Version 1.0

Im Jahre 2010 wurde auf der GUS-Tagung in Pfinztal im Rahmen eines Vortrages ein erstes Ergebnis des Arbeitskreises vorgestellt und die Basisfunktionen der Schnittstelle anhand einer funktionierenden Demoanordnung (Schwingregelsystem mit GUS-Schnittstelle und Klimakammer-Demosoftware mit GUS-Schnittstelle) präsentiert. Der damalige Stand der Schnittstelle wurde als Version 1.0 bezeichnet. Seither wurde die Schnittstellendefinition kontinuierlich weiterentwickelt.

1.8 Version 2.0

Die bisherige Schnittstellendefinition (Basisfunktionen, verabschiedet als Version 1.0) erfüllt die ursprünglichen Ziele des Arbeitskreises (Schnittstelle für kombinierte Prüfanlagen, ermöglicht eine Steuerung der Geräte auf Testebene durch Starten, Stoppen etc. von vordefinierten Tests, inkl. Überwachung auf Testabbruch). Diese Basisfunktionen (Befehle und Statusmeldungen) stellen den verbindlichen Mindestumfang der GUS-Schnittstelle dar.

Die detaillierte Befehlsprogrammierung und Parametereinstellung sowie die Speicherung und Dokumentation der Ergebnisse wird nach wie vor in den jeweiligen Systemen (Klimakammer bzw. Shaker) selbst durchgeführt. Die Steuerung erfolgt dann aus einer individuellen Steuersoftware (mittels ActiveX-Befehlen) durch Aufrufen und Koordinieren derart vordefinierten Tests der jeweiligen Systeme, mit entsprechenden Statusabfragen und Rückmeldungen.

In der Version 2 werden darüber hinaus noch optionale Erweiterungsfunktionen definiert, die über diese grundsätzlichen Funktionen hinausgehen und für komplexere Aufgaben verwendet werden können. Diese zusätzlichen Befehle sind im Kapitel "Erweiterter Befehlssatz (optional)" beschrieben.

2 Befehlsbeschreibung

2.1 Allgemeines

Für die Syntax der Befehle ist eine eindeutige Kennzeichnung vorzusehen, die den Bezug auf den Arbeitskreis verdeutlicht. Es wurde die Vorsilbe „GUS“ vorgeschlagen und angenommen. Die GUS hat hierzu ihre Zustimmung gegeben.

Der Befehlssatz besteht aus einem Basisbefehlssatz der in jedem GUS-kompatiblen Gerät vorhanden sein muss, sowie einem erweiterten Befehlssatz, der implementiert sein kann, aber nicht muss (optional). Der Basisbefehlssatz ist grundsätzlich für die Funktion der Schnittstelle erforderlich, der optionale, erweiterte Befehlssatz kann verwendet werden, um zusätzliche Informationen aus dem angesprochenen Gerät auszulesen, insofern und inwieweit diese Funktionen vom jeweiligen Gerätehersteller über die GUS-Schnittstelle zur Verfügung gestellt werden.

Der Umfang des erweiterten Befehlssatzes, der von einem Gerät angeboten wird, hängt von der jeweiligen Geräteart und vom Gerätehersteller ab. Unterschiedliche Gerätearten werden deshalb einen unterschiedlichen Umfang zur Verfügung stellen, z.B. werden Klimakammern eher einen Eingriff in einen laufenden Test erlauben als Schwingprüfsysteme. Über den Befehl „GUS_GetDeviceInfo“ kann ausgelesen werden, welche erweiterten Funktionen das jeweilige Gerät zur Verfügung stellt. Somit kann jeder Softwareentwickler feststellen, welche erweiterten Funktionen er nutzen kann. Alternativ kann ein Hersteller diesen Befehlssatz auch in einem Datenblatt definieren. Das Auslesen über die Schnittstelle ist aber die bevorzugte Variante.

Zur Verwendung der XML-Rückmeldungen des erweiterten Befehlssatzes in einem übergeordneten Steuerprogramm ist die Verwendung der Schemadatei GUS_DeviceInfo.xsd erforderlich (siehe Anhang).

Die Geräteansprache in der Steuersoftware geschieht durch die Einbindung des jeweiligen Treibers als Objekt. Bei mehreren angeschlossenen Geräten müssen mehrere Objekte erstellt werden. Bei Treibern, die ihrerseits mehrere Geräte ansprechen können, d.h. bei denen über ein Objekt mehrere Geräte angesprochen werden, muss die Ansprache der einzelnen Geräte über einen zu übergebenden Identifier erfolgen (z.B. Seriennummer, Laufvariable o.ä.).
Wenn an einen Mehrgeräte-Treiber kein Identifier übergeben wird (z. B. weil das Gerät mit dem Mehrgerätetreiber ein Gerät mit Eingerätetreiber ersetzen soll), dann muss im Sinne einer Austauschbarkeit ein "Default"-Gerät angesprochen werden, bzw. das mit dem "niederwertigsten" Identifier, um zu verhindern, dass die Steuersoftware hier durch eine Fehlermeldung ("missing identifier" o.Ä.) ausgebremst wird.

2.2 Basisbefehlssatz

Der Basisbefehlssatz dient zur eigentlichen Steuerung der angeschlossenen Geräte. Er besteht aus Applikationsbefehlen (für Kommunikationsaufbau etc.), Testbefehlen (zur Ablaufsteuerung mittels Start-, Stopp- etc.- Befehlen) und Rückmeldungen (Informationen über den aktuellen Betriebszustand, d.h. ob das angesprochene Gerät bereit ist, oder gerade eine Prüfung läuft etc.).

Mit Hilfe des Basisbefehlssatzes kann ein übergeordnetes Steuerprogramm die Kommunikation zu den angeschlossenen Geräten aufbauen (Schwingprüfsystem, Klimakammer, Zusatzgeräte etc.) und anschließend die Geräte anweisen, in den Geräten vordefinierte Programme zu laden, starten, stoppen, pausieren oder weiterzufahren oder alle Prüfprogramme gleichzeitig anzuhalten, wenn ein Gerät eine Störung

meldet. Der Basisbefehlssatz ist ausreichend, um eine kombinierte Prüfung automatisch ablaufen zu lassen. Für einen solchen einfachen automatischen Ablauf mittels des Start- und Stopp-Verfahrens, gesteuert durch ein übergeordnetes Programm, inklusive Überwachung der Gerätetätigkeit während des Prüfablaufes, definiert die folgende Liste den Basisbefehlssatz, der in der GUS-Schnittstelle des anzusteuernden Gerätes implementiert sein muss:

2.2.1 Applikationsbefehle

Funktion	Kommando
Applikationsauswahl	GUS_Open_App
Angeschlossene Geräte detektieren	GUS_Scan_Devices
Geräteeigenschaften auslesen	GUS_GetDeviceInfo
Geräteauswahl	GUS_OpenDevice
Geräteabwahl	GUS_CloseDevice
Applikation deaktivieren	GUS_CloseApp

2.2.1.1 Applikationsauswahl: GUS_Open_App

Kommando:
GUS_Open_App (App=Treiber/DLL ...)

Relevanter Anlagenzustand:
Beteiligte Geräte sind betriebsbereit, jeweilige Betriebs- und Kommunikationssoftware ist noch nicht gestartet.

Wirkung:
Kommunikationssoftware wird geladen

Parameter für Aufruf:
APP=jeweiliger Gerätetreiber/DLL ...

Aufrufformat:
z.B. "SIGNALSTAR_GUS.Application"

Rückgabe-Parameter:
Bestätigung, Versionsnummer

Rückgabeformat:
1 string

2.2.1.2 Angeschlossene Geräte detektieren: GUS_Scan_Devices

Kommando:
GUS_Scan_Devices

Relevanter Anlagenzustand:
Beteiligte Geräte sind betriebsbereit, jeweilige Betriebs- und Kommunikationssoftware ist gestartet, Kommunikation ist initialisiert.

Wirkung:
Suche nach verfügbaren Geräten, liefert Identifier zurück

Parameter für Aufruf:
keine

Aufrufformat:
keines

Rückgabe-Parameter:
Identifiers von allen verfügbaren Geräten (z.B. Seriennummern, Index ..)

Rückgabeformat:
1 string (z.B.: "Klimakammer #109372")

Geräte, bei denen pro Gerät ein separater Treiber eingebunden wird (Geräte, die normalerweise einzeln verwendet werden) ist dieser Befehl nicht erforderlich. Wenn ein angesprochenes Gerät auf diesen Befehl nichts zurückgibt, da nur ein einziges Gerät verfügbar ist, muss die Steuersoftware das als "kein Multigerätetreiber verfügbar" interpretieren, und nicht als Fehler.

2.2.1.3 Geräteeigenschaften auslesen: GUS_GetDeviceInfo

Kommando:
GUS_GetDeviceInfo

Relevanter Anlagenzustand:
Beteiligte Geräte sind betriebsbereit, jeweilige Betriebs- und Kommunikationssoftware ist gestartet, Kommunikation ist initialisiert.

Wirkung:
Das Gerät informiert über seine Eigenschaften und möglichen Parameter und deren Soll- und Istwerte sowie die Wertebereiche.

Parameter für Aufruf:
keine

Aufrufformat:
Keines

Rückgabe-Parameter:
Auflistung aller Eigenschaften und möglichen Parameter des angesprochenen Gerätes (Details siehe Kapitel "GUS_GetDeviceInfo im Detail" im Anhang).

In der Version 1.0 der GUS-Schnittstelle war dieser Befehl anders definiert. Im Sinne einer Rückwärtskompatibilität ist es deshalb erforderlich, dass für den Fall, dass ein angesprochenes Gerät auf den Befehl GUS_GetDeviceinfo keine Information wie hier definiert zurückgibt, die Steuersoftware das als "kein erweiterter Befehlssatz verfügbar" interpretiert, und nicht als Fehler.

Rückgabeformat:
XML gemäß vorgegebener Schemadatei GUS_GetDeviceInfo.xsd (siehe Anhang)

2.2.1.4 Geräteauswahl: GUS_OpenDevice

Kommando:
GUS_OpenDevice(Device=Object)

Relevanter Anlagenzustand:
Beteiligte Geräte sind betriebsbereit, jeweilige Betriebs- und Kommunikationssoftware ist noch nicht gestartet. Applikationsauswahl ist erfolgt.

Wirkung:
Gerät, mit dem kommuniziert werden soll, wird festgelegt, Betriebs- und Kommunikationssoftware wird geladen, Kommunikation wird initialisiert.

Parameter für Aufruf:
Identifier

Aufrufformat:
Keines

Rückgabe-Parameter:
Bestätigung (ACK)

Rückgabeformat:
String

Für Applikationen, die nur ein Gerät steuern, kann GUS_OpenDevice entfallen, wenn GUS_Open_App bereits die entsprechende Funktion enthält.

2.2.1.5 Geräteabwahl: GUS_CloseDevice

Kommando:
GUS_CloseDevice(Device=Object)

Relevanter Anlagenzustand:
Jeder

Wirkung:
Verbindung zum Gerät wird getrennt (damit Gerät frei wird für anderweitige Ansteuerung, laufender Geräteprozess wird dabei nicht beeinflusst)

Parameter für Aufruf:
Identifier

Aufrufformat:
Keines

Rückgabe-Parameter:
Bestätigung (ACK)

Rückgabeformat:
String

Für Applikationen, die nur ein Gerät steuern, kann GUS_CloseDevice entfallen, wenn GUS_Close_App bereits die entsprechende Funktion enthält.

2.2.1.6 Applikation deaktivieren: GUS_CloseApp

Kommando:
GUS_CloseApp

Relevanter Anlagenzustand:
Beteiligte Geräte sind betriebsbereit, jeweilige Betriebs- und Kommunikationssoftware ist gestartet, Kommunikation ist initialisiert.

Wirkung:
Kommunikation wird beendet, Betriebs- und Kommunikationssoftware der Geräte wird geschlossen

Parameter für Aufruf:
Keine

Aufrufformat:
Keines

Rückgabe-Parameter:
Bestätigung (ACK)

Rückgabeformat:
String

2.2.2 Testbefehle

Funktion	Kommando
Testauswahl und -vorbereitung	GUS_PrepareTest
Start	GUS_StartTest
Stop	GUS_StopTest
Pause	GUS_PauseTest
Continue	GUS_ContinueTest
Test schliessen	GUS_CloseTest

Hinweis: Als Alternative zu "GUS_PrepareTest" gibt es in der Version 2 auch den optionalen Befehl "GUS_LoadTest" (siehe Kapitel Testauswahl: GUS_LoadTest).

2.2.2.1 Testauswahl und –vorbereitung: GUS_PrepareTest

Kommando:
GUS_PrepareTest

Relevanter Anlagenzustand:
Angesprochenes Gerät ist betriebsbereit, jeweilige Betriebs- und Kommunikationssoftware ist gestartet, Kommunikation ist initialisiert.

Wirkung:
Ein vordefinierter Test wird ausgewählt und geladen, vorbereitende Routinen (z.B. Selfcheck) werden durchgeführt.

Anmerkung:
Im Zusammenhang mit Diskussionen über den Befehl „GUS_StartTest“ und dessen Wirkung aus den unterschiedlichen Status wurde erkannt, dass die Kombination von "Testdatei laden" und "Pretest starten" in einem einzigen Befehl (=GUS_PrepareTest) nicht optimal ist. Deshalb wurde ein neuer Befehl definiert: „GUS_LoadTest“. Dieser Befehl soll eine Testdatei laden, ohne dann automatisch einen Pretest zu starten.
Der Pretest ist normalerweise sowieso eine Funktion, die von der Gerätesoftware (z.B. Schwingregelsystem) gesteuert wird, und nach dem Starten des Tests durchgeführt wird, je nach Programmierung des jeweiligen Tests bzw. nach den Erfordernissen des Tests. Der Befehl „GUS_PrepareTest“ soll aus Kompatibilitätsgründen weiterhin behalten werden.

Parameter für Aufruf:
Testname mit Pfad, falls erforderlich Vorlagenname

Aufrufformat:
Keines

Rückgabe-Parameter:
Bestätigung (ACK), Fertigmeldung erfolgt über Statusabfrage

Rückgabeformat:
String

2.2.2.2 Test-Start: GUS_StartTest

Kommando:
GUS_StartTest

Relevanter Anlagenzustand:
Geräte befinden sich im READY-Zustand, keine relevante Störmeldung liegt vor.

Wirkung:
Gerät geht in den "RUN-Modus": geladener Test wird gestartet und läuft, bis er fertig abgelaufen ist, oder bis Stopp oder Pause ausgelöst wird, bzw. bis ein Störereignis auftritt, das zum Abbruch führt.

Parameter für Aufruf:
keine

Aufrufformat:
Keines

Rückgabe-Parameter:
Bestätigung (ACK)

Rückgabeformat:
String

2.2.2.3 Test-Stopp: GUS_StopTest

Kommando:
GUS_StopTest

Relevanter Anlagenzustand:
Gerät befindet sich im RUN-Modus, Error-Modus, Finished-Modus oder im Pause-Modus.

Wirkung:
Gerät geht in den "READY-Modus": laufender Test wird angehalten und beendet. Zeitzähler, Ablaufsteuerung etc. innerhalb des Tests werden zurückgesetzt, Test-Parameter werden auf Anfangswerte gesetzt (=Zustand nach "PrepareTest"). Gerät ist bereit für erneutes Starten.

Parameter für Aufruf:
Keine

Aufrufformat:
Keines

Rückgabe-Parameter:
Bestätigung (ACK)

Rückgabeformat:
String

2.2.2.4 Test-Pause: GUS_PauseTest

Kommando:
GUS_PauseTest

Relevanter Anlagenzustand:
Gerät befindet sich im RUN-Modus.

Wirkung:
Gerät geht in den "PAUSE-Modus": laufender Test wird angehalten aber nicht beendet. Zeitzähler, Ablaufsteuerung etc. innerhalb des Tests werden angehalten aber nicht zurückgesetzt. Gerät ist bereit für weiterführen des Tests oder für Stoppen des Tests.

Parameter für Aufruf:
Keine

Aufrufformat:
Keines

Rückgabe-Parameter:
Bestätigung (ACK)

Rückgabeformat:
String

2.2.2.5 Test-Fortsetzung; GUS_ContinueTest

Kommando:
GUS_ContinueTest

Relevanter Anlagenzustand:
Gerät befindet sich im Pause-Modus.

Wirkung:
Gerät geht in den "RUN-Modus": laufender Test wird weitergeführt. Zeitzähler, Ablaufsteuerung etc. innerhalb des Tests laufen wieder weiter.

Parameter für Aufruf:
Keine

Aufrufformat:
Keines

Rückgabe-Parameter:
Bestätigung (ACK)

Rückgabeformat:
String

2.2.2.6 Test schließen: GUS_CloseTest

Kommando:
GUS_CloseTest

Relevanter Anlagenzustand:
Beteiligte Geräte sind betriebsbereit, jeweilige Betriebs- und Kommunikationssoftware ist gestartet, Kommunikation ist initialisiert.

Wirkung:
Geladener Test wird geschlossen, "Grundzustand" wird wieder hergestellt. System ist anschließend bereit für neuen Prepare-Befehl.

Parameter für Aufruf:
Keine

Aufrufformat:
Keines

Rückgabe-Parameter:
Bestätigung (ACK), Fertigmeldung erfolgt über Statusabfrage

Rückgabeformat:
String

2.2.3 Rückmeldungen

2.2.3.1 Gerätestatus (Betriebszustand) abfragen: GUS_GetStatus

Rückgabewert	Rückmeldung	Bedeutung
1	Ready	System ist bereit, Test kann gestartet werden
2	PreTest Running	PreTest/SelfCheck läuft
3	Running	Test läuft
4	Finished	Test ist beendet
5	Pause	Test wurde angehalten, kann weitergefahren oder beendet werden
6	Busy	Gerät befindet sich in einem Übergangszustand
-1	Error	Es ist eine unbekannte Störung aufgetreten (Softwarestörung, Gerätestörung). Störungsbehebung vor Ort

Kommando: GUS_GetStatus

Relevanter Anlagenzustand: jeder

Wirkung: Abfrage des Geräte-STATUS

Parameter für Aufruf: -
Keine

Aufrufformat:
Keines

Rückgabe-Parameter:
String mit Statuscode laut Tabelle

Rückgabeformat:
1 string

2.3 Erweiterter Befehlssatz (optional)

Da die GUS-Schnittstelle grundsätzlich auch für umfangreichere Aufgaben geeignet ist und hierfür auch ein Bedarf besteht, wurde der Befehlssatz um Zusatzfunktionen erweitert, die über die ursprüngliche Zielsetzung hinausgehen. Damit soll die Möglichkeit geschaffen werden, die GUS-Schnittstelle auch für weitergehende Automatisierungen, sowie zur Steuerung von Geräteparks (z.B. eine Reihe von Klimakammern) verwenden zu können. Hierdurch darf aber nicht die Möglichkeit genommen werden, auch einfache Kombinationsanlagen über die Basisbefehle und Statusmeldungen ohne großen Aufwand zu steuern.

Der Umfang des erweiterten Befehlssatzes, der von einem Gerät angeboten wird, hängt von der jeweiligen Geräteart und vom Gerätehersteller ab. Unterschiedliche Gerätearten werden einen unterschiedlichen Umfang zur Verfügung stellen, z.B. werden Klimakammern eher einen Eingriff in einen laufenden Test erlauben als Schwingprüfsysteme. Der Basisbefehlssatz bleibt hiervon unberührt. Dieser muss auf jeden Fall zur Verfügung stehen, um GUS-Schnittstellen-kompatibel zu sein.

2.3.1 Parameterbefehle (optional)

Folgende erweiterte Befehle können in einem Gerät im Rahmen der GUS-Schnittstelle vorgesehen werden. Sie sind für die grundsätzliche Kommunikation zwischen den Geräten nicht erforderlich, können aber für komplexere Steuerungen hilfreich sein.

- **GUS_GetDeviceInfo**
- **GUS_GetInfo**
- **GUS_GetParameter**
- **GUS_SetParameter**

2.3.1.1 GUS_GetDeviceInfo

Über den Befehl „GUS_GetDeviceInfo“ kann ausgelesen werden welche Funktionen und Parameter das jeweilige Gerät zur Verfügung stellt. Somit kann jeder Softwareentwickler feststellen, welche erweiterten Funktionen und Parameter er nutzen kann. Alternativ kann ein Hersteller diesen Befehlssatz auch in einem Datenblatt definieren. Das Auslesen über die Schnittstelle ist aber die bevorzugte Variante.

Wenn ein übergeordnetes Programm den Befehl GUS-GetDeviceInfo an ein Gerät sendet, dann antwortet dieses Gerät mit einer XML-Rückmeldung, in der die Definition aller von diesem Gerät über die GUS-Schnittstelle verfügbaren Funktionen und Parameter aufgelistet sind. D.h. im Gegensatz zum Befehl GUS_GetInfo liefert der Befehl GUS_GetDeviceInfo keine Inhalte (Messwerte etc.) sondern er liefert Metadaten, mit Informationen über die verfügbaren Funktionen und Parameter eines Gerätes.

Für Shaker und Klimakammern sind die möglichen (aber nicht erforderlichen) Parameter in einer Parameterliste definiert (siehe Anhang). Diese Parameter sind zwar optional, d.h. sie können entfallen, aber wenn sie verwendet werden, dann müssen Sie in dem Format und mit den Bezeichnungen verwendet werden, die in der jeweiligen Parameterliste dieses Standards festgelegt sind.

Kommando:
GUS_GetDeviceInfo

Relevanter Anlagenzustand:
Angesprochenes Gerät ist betriebsbereit, jeweilige Betriebs- und Kommunikationssoftware ist gestartet, Kommunikation ist initialisiert.

Wirkung:
Wenn das angesprochene Gerät einen erweiterten Befehlssatz enthält, dann sendet es eine XML-Rückmeldung in der die Definition aller Geräteparameter aufgelistet ist.

Parameter für Aufruf:
Keiner (oder bei Bedarf Gerätename/-nummer)

Aufrufformat:
Keines

Rückgabe-Parameter:
XML-Rückmeldung in der die Definition (nicht die aktuellen Werte!) aller Geräteparameter aufgelistet ist
Rückgabeformat:
XML (Formatbeschreibung siehe Anhang)

Beispiele für mögliche Parameterlisten eines Shakers oder einer Klimakammer sind im Anhang dargestellt (GUS_GetDeviceInfo_Chamber_Example.xml)

2.3.1.2 GUS_GetInfo

Der Befehl GUS_GetInfo liefert alle über die GUS-Schnittstelle verfügbaren Informationen eines Gerätes. Bei Aufruf dieses Befehls durch das übergeordnete Steuerprogramm antwortet das Gerät mit einer XML-Rückmeldung, die den Inhalt der von GUS_GetDeviceInfo aufgelisteten verfügbaren Informationen auflistet. Im Gegensatz zum Befehl GUS_GetDeviceInfo, der Metadaten (Strukturinformationen) liefert, überträgt der Befehl GUS_GetInfo die Informationen selbst (Messwerte, Zustandsinformationen etc.). Diese Parameter sind zwar optional, d.h. ein Gerät muss sie nicht liefern, aber wenn sie verwendet werden, dann müssen Sie in dem Format und mit den Bezeichnungen verwendet werden, die in der jeweiligen Parameterliste dieses Standards festgelegt sind.

Kommando:
GUS_GetInfo

Relevanter Anlagenzustand:
Angesprochenes Gerät ist betriebsbereit, jeweilige Betriebs- und Kommunikationssoftware ist gestartet, Kommunikation ist initialisiert.

Wirkung:
Wenn das angesprochene Gerät einen erweiterten Befehlssatz enthält, dann sendet es eine XML-Rückmeldung, in der die aktuellen Werte aller Geräteparameter aufgelistet ist.

Parameter für Aufruf:
Keiner (oder bei Bedarf Gerätename/nummer)

Aufrufformat:
Keines

Rückgabe-Parameter:
XML-Rückmeldung in der die aktuellen Werte aller Geräteparameter aufgelistet ist

Rückgabeformat:
XML (Format wie in der XML-Rückmeldung beschrieben, die als Antwort auf GUS_GetDeviceInfo vom Gerät gesendet wurde)

2.3.1.3 GUS_GetParameter

Relevanter Anlagenzustand:
Angesprochenes Gerät ist betriebsbereit, jeweilige Betriebs- und Kommunikationssoftware ist gestartet, Kommunikation ist initialisiert.

Wirkung:
Parameter des Geräts können einzeln während der Laufzeit des übergeordneten Steuerprogramms ausgelesen werden.

Parameter für Aufruf:
Parametername

Aufrufformat:
XML-Format

Rückgabe-Parameter:
Wert

Rückgabeformat:
XML-Format

2.3.1.4 GUS_SetParameter

Relevanter Anlagenzustand:
Angesprochenes Gerät ist betriebsbereit, jeweilige Betriebs- und Kommunikationssoftware ist gestartet, Kommunikation ist initialisiert.

Wirkung:
Nicht schreibgeschützte Parameter des Geräts können einzeln während der Laufzeit des übergeordneten Steuerprogramms geändert werden.

Parameter für Aufruf:
Parametername und Wert

Aufrufformat:
XML-Format

Rückgabe-Parameter:
ACK

Rückgabeformat:
String

2.3.2 Sonstige Erweiterungen (optional)

Gegenüber der Version 1.0 der GUS-Schnittstelle wurden folgende Erweiterungen eingeführt.

2.3.2.1 Alternative Testauswahl: GUS_LoadTest (optional)

Funktion	Kommando
Testauswahl	GUS_LoadTest

Kommando:
GUS_LoadTest

Relevanter Anlagenzustand:
Angesprochenes Gerät ist betriebsbereit, jeweilige Betriebs- und Kommunikationssoftware ist gestartet, Kommunikation ist initialisiert.

Wirkung:
Ein vordefinierter Test wird ausgewählt und geladen. Es wird kein Pretest ausgeführt.

Parameter für Aufruf:
Testname mit Pfad, falls erforderlich Vorlagenname

Aufrufformat:
Keines

Rückgabe-Parameter:
Bestätigung (ACK), Fertigmeldung erfolgt über Statusabfrage

Rückgabeformat:
String

2.3.2.2 Zusätzliche Rückmeldung: Störmeldung: GUS_GetError (optional)

Funktion	Kommando
Störmeldung (Alarm, Abbruch, allg.) abfragen	GUS_GetError

Kommando:
GUS_GetError

Relevanter Anlagenzustand:
Jeder

Wirkung:
Abfrage von Gerätefehlern

Parameter für Aufruf:
keine

Aufrufformat:
Keines

Rückgabe-Parameter:
Textstring mit Zusammenfassung aller Gerätefehler im Klartext (nur Geräte- und Systemfehler, keine Testabbruchinfos etc.)

Rückgabeformat:
String

2.3.2.3 Zusätzlicher Gerätestatus (optional)

Rückgabewert	Rückmeldung	Bedeutung
0	Open	Kommunikation zum Gerät (Device) ist hergestellt.
9	Closed	Applikation ist geöffnet, aber die Kommunikation mit dem Gerät ist noch nicht aufgebaut.

Kommando: GUS_GetStatus

Relevanter Anlagenzustand: jeder

Wirkung: Abfrage des Geräte-STATUS

Parameter für Aufruf:
Keine

Aufrufformat:
Keines

Rückgabe-Parameter:
String mit Statuscode laut Tabelle

Rückgabeformat:
String

2.4 Sonstige Geräte

Für Erweiterungen des Gerätepools (zur Einbindung beliebiger zusätzlicher Geräte wie z.B. Schaltgeräte, SPS, Multimeter, Stromversorgungen) müssen diese Geräte ebenfalls nach der oben beschriebenen Methode eingebunden werden können.

3 Anhang

Hinweis: Die in Kapitel 3.1 und 3.2 beschriebenen XML-Strukturen basieren auf einem Vorschlag, den Ben Haest/Fa. QED für den GUS-AK erstellt hat.

3.1 Schemadatei GUS_DeviceInfo.xsd

Zur Verwendung der XML-Rückmeldungen des erweiterten Befehlssatzes in einem übergeordneten Steuerprogramm ist die Verwendung der Schemadatei GUS_DeviceInfo.xsd erforderlich: In dieser Datei sind die Strukturen und Datentypen definiert, die in den XML-Informationen der Befehle GUS_GetDeviceInfo und GUS_GetInfo verwendet werden.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--W3C Schema generated by XMLSPY v5 rel. 4 U (http://www.xmlspy.com)-->
<xs:schema targetNamespace="http://www. ...tbd... /GUS_DeviceInfo" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http:// ... tbd .. /GUS_DeviceInfo" xmlns:o="http:// ... tbd ... /GUS_DeviceInfo" elementFormDefault="qualified">
	<xs:element name="Device">
		<xs:complexType>
			<xs:sequence>
				<xs:element name="Group" type="GroupType" maxOccurs="unbounded"/>
			</xs:sequence>
		</xs:complexType>
		<xs:unique name="GroupNameUnique">
			<xs:selector xpath="o:Group"/>
			<xs:field xpath="@Name"/>
		</xs:unique>
 </xs:element>
	<xs:complexType name="GroupType">
		<xs:sequence>
			<xs:element name="Attribute" type="AttributeType" maxOccurs="unbounded">
			</xs:element>
		</xs:sequence>
		<xs:attribute name="Name" type="xs:NCName" use="required"/>
	</xs:complexType>
	<xs:complexType name="AttributeType">
		<xs:all>
			<xs:element name="IsReadOnly" type="xs:boolean" minOccurs="0"/>
			<xs:element name="Type" type="BaseType"/>
		</xs:all>
		<xs:attribute name="Name" type="xs:NCName" use="required"/>
	</xs:complexType>
	<xs:complexType name="BaseType" abstract="true"/>
	<xs:complexType name="Boolean">
		<xs:complexContent>
			<xs:extension base="BaseType"/>
		</xs:complexContent>
	</xs:complexType>
	<xs:complexType name="NumericType" abstract="true">
		<xs:complexContent>
			<xs:extension base="BaseType">
				<xs:sequence minOccurs="0">
					<xs:element name="EngineeringUnit" type="xs:string" minOccurs="0"/>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:complexType name="Integer">
		<xs:complexContent>
			<xs:extension base="NumericType">
```

```
                                        <xs:sequence>
                                                <xs:element name="Restriction" type="IntegerRestrictionType" minOc-
curs="0"/>
                                        </xs:sequence>
                                </xs:extension>
                        </xs:complexContent>
                </xs:complexType>
                <xs:complexType name="Decimal">
                        <xs:complexContent>
                                <xs:extension base="NumericType">
                                        <xs:sequence>
                                                <xs:element name="Restriction" type="DecimalRestrictionType" minOc-
curs="0"/>
                                        </xs:sequence>
                                </xs:extension>
                        </xs:complexContent>
                </xs:complexType>
                <xs:complexType name="String">
                        <xs:complexContent>
                                <xs:extension base="BaseType">
                                        <xs:sequence>
                                                <xs:element name="Restriction" type="StringRestrictionType" minOc-
curs="0"/>
                                        </xs:sequence>
                                </xs:extension>
                        </xs:complexContent>
                </xs:complexType>
                <xs:complexType name="Date">
                        <xs:complexContent>
                                <xs:extension base="BaseType">
                                        <xs:sequence>
                                                <xs:element name="Restriction" type="DateRestrictionType" minOccurs="0"/>
                                        </xs:sequence>
                                </xs:extension>
                        </xs:complexContent>
                </xs:complexType>
                <xs:complexType name="ComplexType">
                        <xs:complexContent>
                                <xs:extension base="BaseType">
                                        <xs:sequence>
                                                <xs:element name="Attribute" type="AttributeType" maxOccurs="un-
bounded"/>
                                        </xs:sequence>
                                </xs:extension>
                        </xs:complexContent>
                </xs:complexType>
                <xs:complexType name="IntegerRestrictionType">
                        <xs:choice>
                                <xs:sequence>
                                        <xs:element name="Enumeration" type="xs:integer" maxOccurs="unbounded"/>
                                </xs:sequence>
                                <xs:element name="Range" type="IntegerRangeRestrictionType"/>
                                <xs:element name="TotalDigits" type="xs:positiveInteger"/>
                        </xs:choice>
                </xs:complexType>
                <xs:complexType name="DecimalRestrictionType">
                        <xs:choice>
                                <xs:sequence>
                                        <xs:element name="Enumeration" type="xs:decimal" maxOccurs="unbounded"/>
                                </xs:sequence>
                                <xs:element name="Range" type="DecimalRangeRestrictionType"/>
                                <xs:element name="LengthRange" type="DecimalLengthRangeRestrictionType"/>
```

```
                </xs:choice>
        </xs:complexType>
        <xs:complexType name="StringRestrictionType">
                <xs:choice>
                        <xs:sequence>
                                <xs:element name="Enumeration" type="xs:string" minOccurs="0" maxOccurs="un-
bounded"/>
                        </xs:sequence>
                        <xs:element name="Length" type="StringLengthRestrictionType" minOccurs="0"/>
                </xs:choice>
        </xs:complexType>
        <xs:complexType name="DateRestrictionType">
                <xs:choice>
                        <xs:sequence>
                                <xs:element name="Enumeration" type="xs:date" maxOccurs="unbounded"/>
                        </xs:sequence>
                </xs:choice>
        </xs:complexType>
        <xs:complexType name="StringLengthRestrictionType">
                <xs:all>
                        <xs:element name="MinLength" type="xs:positiveInteger" minOccurs="0"/>
                        <xs:element name="MaxLength" type="xs:positiveInteger" minOccurs="0"/>
                </xs:all>
        </xs:complexType>
        <xs:complexType name="IntegerRangeRestrictionType">
                <xs:all>
                        <xs:element name="MinValueInclusive" type="xs:integer" minOccurs="0"/>
                        <xs:element name="MaxValueInclusive" type="xs:integer" minOccurs="0"/>
                </xs:all>
        </xs:complexType>
        <xs:complexType name="DecimalRangeRestrictionType">
                <xs:all>
                        <xs:element name="FractionDigits" type="xs:positiveInteger" minOccurs="0"/>
                        <xs:element name="MinValueInclusive" type="xs:decimal" minOccurs="0"/>
                        <xs:element name="MaxValueInclusive" type="xs:decimal" minOccurs="0"/>
                </xs:all>
        </xs:complexType>
        <xs:complexType name="DecimalLengthRangeRestrictionType">
                <xs:all>
                        <xs:element name="TotalDigits" type="xs:positiveInteger" minOccurs="0"/>
                        <xs:element name="FractionDigits" type="xs:positiveInteger" minOccurs="0"/>
                </xs:all>
        </xs:complexType>
</xs:schema>
```

3.2 GUS_GetDeviceInfo im Detail

Für den Befehl GUS_GetDeviceInfo sind folgende Gruppen vorgesehen:

Gruppe	Inhalt
DeviceInfo	Allgemeine Geräteinformationen, wie z.B. Hersteller, Modell, Seriennummer etc.
ControlledValues	Soll- und Ist-Werte des Gerätes
MeasurementValues	Messwerte von Messeingängen des Gerätes
Operation	Informationen über den Funktionszustand des Gerätes (z.B. Klimakammer: Temperatur ist EIN, Feuchte ist AUS, Schwingprüfsystem: SINUS-Prüfung ist aktiviert)
Messages	Alarm- und Warnmeldungen aus dem Prüfprogramm (z.B.: Prüfabbruch, oder Überschreitung von Warngrenzen)
Testing	Informationen zum Prüfablauf, wie z.B.: "Vergangene Prüfzeit", "Verbleibende Prüfzeit", Aktueller Schritt im programmierten Ablauf,

Für Shaker und Klimakammern sind die möglichen (aber nicht erforderlichen) Parameter in folgendem XML-Listing definiert (Details siehe Datei "GUS_GetDeviceInfo_Chamber_Example.xml"). Diese Parameter sind zwar optional, d.h. sie können entfallen, aber wenn sie verwendet werden, dann müssen Sie in genau dem Format und mit den Bezeichnungen verwendet werden, die in der jeweiligen Parameterliste (XML-Listing) dieses Standards festgelegt sind. Beispiel für ein GUS_GetDeviceInfo-XML-Listing:

3.2.1 Übersicht:

```
−<Device xsi:schemaLocation="http://www. .. tbd ....com/GUS_DeviceInfo GUS_DeviceInfo.xsd">
  −<Group Name="DeviceInfo">
    +<Attribute Name="Name"></Attribute>
    +<Attribute Name="DeviceType"></Attribute>
    +<Attribute Name="Manufacturer"></Attribute>
    +<Attribute Name="DeviceModel"></Attribute>
    +<Attribute Name="SerialNumber"></Attribute>
    +<Attribute Name="Remark"></Attribute>
   </Group>
  −<Group Name="ControlledValues">
    +<Attribute Name="Temperature"></Attribute>
    +<Attribute Name="Humidity"></Attribute>
   </Group>
  −<Group Name="Measurements">
    +<Attribute Name="Measurement01"></Attribute>
    +<Attribute Name="Measurement02"></Attribute>
   </Group>
  −<Group Name="Operation">
    +<Attribute Name="Temperature"></Attribute>
    +<Attribute Name="Humidity"></Attribute>
    +<Attribute Name="Solar"></Attribute>
    +<Attribute Name="Relay01"></Attribute>
    +<Attribute Name="Relay02"></Attribute>
   </Group>
  −<Group Name="Message">
    +<Attribute Name="SecurityAlert"></Attribute>
    +<Attribute Name="TestAlert"></Attribute>
    +<Attribute Name="TestAlarm"></Attribute>
   </Group>
  −<Group Name="Testing">
    +<Attribute Name="TimeElapsedInTolerance"></Attribute>
    +<Attribute Name="TimeElapsedSinceStart"></Attribute>
    +<Attribute Name="TimeRemaining"></Attribute>
    +<Attribute Name="StepInProgram"></Attribute>
   </Group>
 </Device>
```

3.2.2 Detaillierte Auflistung:

```
−<Device xsi:schemaLocation="http://www. .. tbd ....com/GUS_DeviceInfo.xsd">
  −<Group Name="DeviceInfo">
    −<Attribute Name="Name">
        <IsReadOnly>true</IsReadOnly>
      −<Type xsi:type="String">
       +<Restriction></Restriction>
        </Type>
      </Attribute>
    −<Attribute Name="DeviceType">
        <IsReadOnly>true</IsReadOnly>
      −<Type xsi:type="String">
        −<Restriction>
            <Enumeration>Climatic</Enumeration>
          </Restriction>
        </Type>
      </Attribute>
    −<Attribute Name="Manufacturer">
        <IsReadOnly>true</IsReadOnly>
      −<Type xsi:type="String">
        −<Restriction>
          −<Length>
              <MinLength>5</MinLength>
            </Length>
          </Restriction>
        </Type>
      </Attribute>
    −<Attribute Name="DeviceModel">
        <IsReadOnly>true</IsReadOnly>
      −<Type xsi:type="String">
        −<Restriction>
          −<Length>
              <MinLength>3</MinLength>
            </Length>
          </Restriction>
        </Type>
      </Attribute>
    −<Attribute Name="SerialNumber">
        <IsReadOnly>true</IsReadOnly>
      −<Type xsi:type="String">
        −<Restriction>
          −<Length>
              <MinLength>5</MinLength>
            </Length>
          </Restriction>
        </Type>
      </Attribute>
    −<Attribute Name="Remark">
        <IsReadOnly>true</IsReadOnly>
      −<Type xsi:type="String">
        −<Restriction>
          −<Length>
              <MinLength>5</MinLength>
            </Length>
          </Restriction>
        </Type>
      </Attribute>
    </Group>
```

```
−<Group Name="ControlledValues">
  −<Attribute Name="Temperature">
    −<Type xsi:type="ComplexType">
      −<Attribute Name="CurrentValue">
          <IsReadOnly>true</IsReadOnly>
        −<Type xsi:type="Decimal">
            <EngineeringUnit>°C</EngineeringUnit>
          −<Restriction>
            −<Range>
                <MinValueInclusive>-70.0</MinValueInclusive>
                <FractionDigits>1</FractionDigits>
                <MaxValueInclusive>200.0</MaxValueInclusive>
              </Range>
            </Restriction>
          </Type>
        </Attribute>
      −<Attribute Name="DemandValue">
          <IsReadOnly>false</IsReadOnly>
        −<Type xsi:type="Decimal">
            <EngineeringUnit>°C</EngineeringUnit>
          −<Restriction>
            −<Range>
                <MinValueInclusive>-70.0</MinValueInclusive>
                <FractionDigits>1</FractionDigits>
                <MaxValueInclusive>200.0</MaxValueInclusive>
              </Range>
            </Restriction>
          </Type>
        </Attribute>
      −<Attribute Name="DemandValueAchieved">
          <IsReadOnly>true</IsReadOnly>
          <Type xsi:type="Boolean"/>
        </Attribute>
      −<Attribute Name="ChangeRate">
          <IsReadOnly>true</IsReadOnly>
        −<Type xsi:type="Decimal">
            <EngineeringUnit>K/min</EngineeringUnit>
          −<Restriction>
            −<Range>
                <MinValueInclusive>-4.0</MinValueInclusive>
                <FractionDigits>1</FractionDigits>
                <MaxValueInclusive>4.0</MaxValueInclusive>
              </Range>
            </Restriction>
          </Type>
        </Attribute>
      </Type>
    </Attribute>
```

```
−<Attribute Name="Humidity">
  −<Type xsi:type="ComplexType">
    −<Attribute Name="CurrentValue">
        <IsReadOnly>true</IsReadOnly>
      −<Type xsi:type="Decimal">
          <EngineeringUnit>%RH</EngineeringUnit>
        −<Restriction>
          −<Range>
              <MinValueInclusive>5.0</MinValueInclusive>
              <FractionDigits>1</FractionDigits>
              <MaxValueInclusive>95.0</MaxValueInclusive>
            </Range>
          </Restriction>
        </Type>
      </Attribute>
    −<Attribute Name="DemandValue">
        <IsReadOnly>false</IsReadOnly>
      −<Type xsi:type="Decimal">
          <EngineeringUnit>%RH</EngineeringUnit>
        −<Restriction>
          −<Range>
              <MinValueInclusive>5.0</MinValueInclusive>
              <FractionDigits>1</FractionDigits>
              <MaxValueInclusive>95.0</MaxValueInclusive>
            </Range>
          </Restriction>
        </Type>
      </Attribute>
    −<Attribute Name="DemandValueAchieved">
        <IsReadOnly>true</IsReadOnly>
        <Type xsi:type="Boolean"/>
      </Attribute>
    </Type>
  </Attribute>
</Group>
−<Group Name="Measurements">
  −<Attribute Name="Measurement01">
    −<Type xsi:type="Decimal">
        <EngineeringUnit>°C</EngineeringUnit>
      −<Restriction>
        −<Range>
            <MinValueInclusive>-70.0</MinValueInclusive>
            <FractionDigits>1</FractionDigits>
            <MaxValueInclusive>200.0</MaxValueInclusive>
          </Range>
        </Restriction>
      </Type>
    </Attribute>
  −<Attribute Name="Measurement02">
    −<Type xsi:type="Decimal">
        <EngineeringUnit>V</EngineeringUnit>
      −<Restriction>
        −<Range>
            <MinValueInclusive>0.0</MinValueInclusive>
            <FractionDigits>1</FractionDigits>
            <MaxValueInclusive>10.0</MaxValueInclusive>
          </Range>
        </Restriction>
      </Type>
    </Attribute>
</Group>
```

```
−<Group Name="Operation">
  −<Attribute Name="Temperature">
      <IsReadOnly>true</IsReadOnly>
      <Type xsi:type="Boolean"/>
    </Attribute>
  −<Attribute Name="Humidity">
      <IsReadOnly>true</IsReadOnly>
      <Type xsi:type="Boolean"/>
    </Attribute>
  −<Attribute Name="Solar">
      <IsReadOnly>true</IsReadOnly>
      <Type xsi:type="Boolean"/>
    </Attribute>
  −<Attribute Name="Relay01">
      <IsReadOnly>false</IsReadOnly>
      <Type xsi:type="Boolean"/>
    </Attribute>
  −<Attribute Name="Relay02">
      <IsReadOnly>false</IsReadOnly>
      <Type xsi:type="Boolean"/>
    </Attribute>
  </Group>
−<Group Name="Message">
  −<Attribute Name="SecurityAlert">
      <IsReadOnly>true</IsReadOnly>
      <Type xsi:type="Boolean"/>
    </Attribute>
  −<Attribute Name="TestAlert">
      <IsReadOnly>true</IsReadOnly>
      <Type xsi:type="Boolean"/>
    </Attribute>
  −<Attribute Name="TestAlarm">
      <IsReadOnly>true</IsReadOnly>
      <Type xsi:type="Boolean"/>
    </Attribute>
  </Group>
```

```
−<Group Name="Testing">
  −<Attribute Name="TimeElapsedInTolerance">
      <IsReadOnly>true</IsReadOnly>
    −<Type xsi:type="Integer">
        <EngineeringUnit>s</EngineeringUnit>
      −<Restriction>
        −<Range>
            <MinValueInclusive>0</MinValueInclusive>
          </Range>
        </Restriction>
      </Type>
    </Attribute>
  −<Attribute Name="TimeElapsedSinceStart">
      <IsReadOnly>true</IsReadOnly>
    −<Type xsi:type="Integer">
        <EngineeringUnit>s</EngineeringUnit>
      −<Restriction>
        −<Range>
            <MinValueInclusive>0</MinValueInclusive>
          </Range>
        </Restriction>
      </Type>
    </Attribute>
  −<Attribute Name="TimeRemaining">
      <IsReadOnly>true</IsReadOnly>
    −<Type xsi:type="Integer">
        <EngineeringUnit>s</EngineeringUnit>
      −<Restriction>
        −<Range>
            <MinValueInclusive>0</MinValueInclusive>
          </Range>
        </Restriction>
      </Type>
    </Attribute>
  −<Attribute Name="StepInProgram">
      <IsReadOnly>true</IsReadOnly>
    −<Type xsi:type="Integer">
      −<Restriction>
        −<Range>
            <MinValueInclusive>0</MinValueInclusive>
          </Range>
        </Restriction>
      </Type>
    </Attribute>
  </Group>
</Device>
```

3.2.3 Beispiel

Der Befehl GUS_GetInfo würde bei einem Gerät, welches das vorstehende GUS_DeviceInfo-XML-Listing liefert, dann z.B. den folgenden Inhalt als XML-Rückmeldung zurückliefern (Details siehe beiliegende Beispieldatei " GUS_GetInfo_Chamber_Example.xml"):

```
−<Device>
  −<DeviceInfo>
      <Name>Chamber_001</Name>
      <DeviceType>Climatic</DeviceType>
      <Manufacturer>ABC_Company</Manufacturer>
      <DeviceModel>VT XX</DeviceModel>
      <SerialNumber>XXXXX</SerialNumber>
      <Remark>Do not use during weekend</Remark>
    </DeviceInfo>
  −<ControlledValues>
    −<Temperature>
        <CurrentValue>20.0</CurrentValue>
        <DemandValue>23.0</DemandValue>
        <DemandValueAchieved>1</DemandValueAchieved>
        <ChangeRate>0.0</ChangeRate>
      </Temperature>
    −<Humidity>
        <CurrentValue>49.70</CurrentValue>
        <DemandValue>50.0</DemandValue>
        <DemandValueAchieved>1</DemandValueAchieved>
      </Humidity>
    </ControlledValues>
  −<Measurements>
      <Measurement01>19.5</Measurement01>
      <Measurement02>5.2</Measurement02>
    </Measurements>
  −<Operation>
      <Temperature>true</Temperature>
      <Humidity>true</Humidity>
      <Solar>false</Solar>
      <Relay01>false</Relay01>
      <Relay02>false</Relay02>
    </Operation>
  −<Message>
      <SecurityAlert>false</SecurityAlert>
      <TestAlert>false</TestAlert>
      <TestAlarm>false</TestAlarm>
    </Message>
  −<Testing>
      <TimeElapsedInTolerance>0</TimeElapsedInTolerance>
      <TimeElapsedSinceStart>0</TimeElapsedSinceStart>
      <TimeRemaining>0</TimeRemaining>
      <StepInProgram>0</StepInProgram>
    </Testing>
</Device>
```

3.3 Anlagenzustände

Folgende Befehle sind in den jeweiligen Anlagenzuständen verwendbar:

Kommando	Relevanter Anlagenzustand	Status dieses Zustandes
BASIS-Befehlssatz (kompatibel zu V 1.0)		
GUS_Open_App (App=Treiber/DLL ...)	Beteiligte Geräte sind betriebsbereit, jeweilige Betriebs- und Kommunikationssoftware ist noch nicht gestartet.	-1
GUS_Scan_Devices	Beteiligte Geräte sind betriebsbereit, jeweilige Betriebs- und Kommunikationssoftware ist gestartet, Kommunikation ist initialisiert.	0
GUS_GetDeviceInfo	Beteiligte Geräte sind betriebsbereit, jeweilige Betriebs- und Kommunikationssoftware ist gestartet, Kommunikation ist initialisiert.	0, 1
GUS_OpenDevice	Beteiligte Geräte sind betriebsbereit, jeweilige Betriebs- und Kommunikationssoftware ist noch nicht gestartet. Applikationsauswahl ist erfolgt.	9
GUS_CloseDevice	Jeder	jeder
GUS_CloseApp	Beteiligte Geräte sind betriebsbereit, jeweilige Betriebs- und Kommunikationssoftware ist gestartet, Kommunikation ist initialisiert	0, 1, -1
GUS_PrepareTest	Angesprochenes Gerät ist betriebsbereit, jeweilige Betriebs- und Kommunikationssoftware ist gestartet, Kommunikation ist initialisiert.	0, -1
GUS_StartTest	Geräte befinden sich im READY-Zustand, keine relevante Störmeldung liegt vor	1
GUS_StopTest	Gerät befindet sich im RUN-Modus oder im Pause-Modus	3, 5
GUS_PauseTest	Gerät befindet sich im RUN-Modus	3
GUS_ContinueTest	Gerät befindet sich im PAUSE-Modus	5
GUS_CloseTest	Beteiligte Geräte sind betriebsbereit, jeweilige Betriebs- und Kommunikationssoftware ist gestartet, Kommunikation ist initialisiert.	1
GUS_GetStatus	Jeder	jeder
Erweiterter Befehlssatz (optional)		
GUS_LoadTest	Angesprochenes Gerät ist betriebsbereit, jeweilige Betriebs- und Kommunikationssoftware ist gestartet, Kommunikation ist initialisiert.	0, -1
GUS_GetError	Jeder	jeder
GUS_GetInfo	Jeder	jeder

Hinweise:
- Die Statusrückmeldungen 0 und 9 sind erst ab Version 2.0 definiert.

3.4 Status-Matrix

Rückgabe-wert	Format	Rückmeldung	Bedeutung
BASIS-Befehlssatz (kompatibel zu V 1.0)			
1	String	Ready	System ist bereit, Test kann gestartet werden
2	String	PreTest Running	PreTest/SelfCheck läuft
3	String	Running	Test läuft
4	String	Finished	Test ist abgelaufen
5	String	Pause	Test wurde angehalten, kann weitergefahren oder beendet werden
6	String	Busy	Gerät befindet sich in einem Übergangszustand
-1	String	Error	Es ist eine Störung aufgetreten (Softwarestörung, Gerätestörung)
Erweiterter Befehlssatz (optional)			
0	String	Device Open	Kommunikation zum Gerät (Device) ist herge-stellt.
9	String	Device Closed	Applikation ist geöffnet, aber die Kommunikation mit dem Gerät ist noch nicht aufgebaut.

Die nachfolgende Tabelle ist so zu lesen, dass aus den Statuszuständen der obersten Zeile nur jeweils die mit Häkchen gekennzeichneten Befehle erlaubt sind.

	-1	0	1	2	3	4	5	6	9
GUS_OpenApp	x	x	x	x	x	x	x	x	x
GUS_Scan_Devices	x	√	√	x	√	x	√	x	√
GUS_GetDeviceInfo	x	x	√	x	√	√	√	x	x
GUS_OpenDevice	x	x	x	x	x	x	x	x	√
GUS_CloseDevice	√	√	√	x	x	x	x	x	x
GUS_CloseApp	√	√	√	x	x	x	x	x	√
GUS_PrepareTest	x	√	√	x	x	x	x	x	x
GUS_StartTest	x	x	√	x	x	x	x	x	x
GUS_StopTest	x	x	x	√	√	√	√	x	x
GUS_PauseTest	x	x	x	x	√	x	x	x	x
GUS_ContinueTest	x	x	x	x	x	x	√	x	x
GUS_CloseTest	√	x	√	x	x	x	x	x	x
GUS_GetStatus	x	√	√	√	√	√	√	x	x
GUS_LoadTest	x	√	√	√	√	x	x	x	x
GUS_GetError	√	√	√	√	√	√	√	x	x
GUS_GetInfo	x	√	√	√	√	√	√	x	x

Hinweise:
- Die Statusrückmeldungen 0 und 9 sind erst ab Version 2.0 definiert.