Erklärender Text zum Statusdiagramm und zum Erweiterten Befehlssatz

Intro

Die GUS Standard Schnittstelle (Version 1.0) ist bisher als „einfache“ Automatisierung einer kombinierten Prüfung (Klimakammer und Shaker) entwickelt worden. Wie ein übergeordnetes Programm die Kontrolle und Überwachung übernehmen soll, war bis heute kein Thema.

Die definierten Status eines Prüfablaufs sind hauptsächlich auf die Prüfung selber zugeschnitten. Der Grundgedanke war, dass das übergeordnete Programm die Kontrolle der Prüfung nicht übernehmen sollte. Eine Prüfung wird auf der niedrigen Ebene durch den Regler eines Gerätes durchgeführt, während das übergeordnete Programm die Geräte steuert und überwacht. Dazu haben noch einige Status (9: device closed und 0: device open) in der ersten Version (1.0) der Standard GUS Schnittstelle gefehlt. Diese Status werden für die Steuerung und Überwachung der Geräte benötigt.

Andererseits sind einige Status manchmal artifiziell, weil jedes Gerät wieder anders funktioniert. So sind die Status bzgl. des Prüfablaufs möglicherweise nicht mit dem Gerät übereinstimmend, weil das Gerät an sich den definierten Status nicht hat. Trotzdem sind die Status und deren Übergang ganz wichtig, um jederzeit das Gerät steuern zu können und eventuelle Probleme bei Aufbau, Synchronisierung und Durchführung der Prüfung identifizieren zu können. Die Status und das zyklisches Abfragen von bestimmten Parametern der Geräte sind erforderlich, um erweiterte kombinierte Prüfungen synchronisieren zu können. Die Version 2.0 der Standard GUS Schnittstelle schlägt einen erweiterten Befehlssatz vor, um den Austausch von Parametern und Variablen zu ermöglichen,.

Überblick über die Status eines Gerätes

Status 9: closed

Für das übergeordnete Programm ist es wichtig zu wissen, ob ein Gerät verfügbar ist oder nicht. Es ist wichtig zu wissen, ob die Kommunikation mit dem Regler funktioniert oder nicht. Der Befehl „GUS_Open_App“ lädt den Driver. Der Parameter des Befehls ist der Name des Drivers, wie im Windows® Betriebssystem registriert.
Die Antwort des Gerätes muss der String „ACK: Gerät Serien-/Versionsnummer“ sein. Das Gerät, das mittels des gewählten Drivers kommuniziert, hat den Status 9 (closed). Die Kommunikation mit dem Gerät ist noch nicht aufgebaut. Der Status ist 9 und das Gerät ist noch nicht verfügbar.

Die Kommunikationssoftware wird durch den Befehl „GUS_CloseApp“ geschlossen. Für den Befehl „GUS_CloseApp“ wird keine Antwort vom Gerät erwartet.

Status 0: open

Mit dem Befehl „GUS_OpenDevice“ wird die Kommunikation mit dem Gerät aufgebaut. Wo die Kommunikationssoftware möglicherweise mit vielen Geräte kommunizieren kann (z.B. S!MPATI), ist die Gerätenummer dem Befehl als Parameter hinzuzufügen. Damit wird das Gerät identifiziert und es wird eine Eins-zu-Eins-Kommunikation hergestellt. Der Status ändert sich auf 0 (wenn das Gerät verfügbar ist und für einen Test verwendet werden kann). Nach der positiven Bestätigung („ACK“) und der Statusänderung von 9 (closed) auf 0 (open) ist das Gerät jetzt für eine Prüfung bereit.

Kann die Kommunikation mit dem Gerät nicht hergestellt werden, bleibt der Status weiterhin bei 9. Das Gerät steht nicht zur Verfügung.

Mit dem Befehl „GUS_CloseDevice“ wird die Kommunikation gestoppt. Die Verbindung mit dem Gerät existiert nicht mehr. Der Status ändert sich von 0 (open) auf 9 (close). Das Gerät steht nicht mehr zur Verfügung.

Status 1: ready

Mit dem Befehl „GUS_PrepareTest“ wird die Testdatei (oder das Test-Verzeichnis) durch das entsprechende Gerät geladen. Manche Geräte öffnen die Testdatei nicht; stattdessen benötigen sie die Information, in welchem Verzeichnis die Testdefinition und die Testparameter zu finden sind. Der Status ändert sich von 0 (open) auf 1 (ready): das Gerät steht zur Verfügung und die Testspezifikationen wurden geladen. Das Gerät ist betriebsbereit, um die Prüfung zu starten. Das Gerät antwortet mit dem String „ACK“. Falls der Test nicht geladen werden kann (die Testdatei oder das Testverzeichnis gibt es nicht oder kann nicht gefunden werden), muss die Antwort der String „ERR“ sein. Der Status bleibt 0 (open). Das Gerät steht nicht bereit, um den Test zu starten.

Der Befehl „GUS_StartTest“ startet den Test und der Status des Gerätes ändert sich von 1 (ready) auf 2 (PreTestRunning). Verschiedene Geräte führen beim Start keinen Pretest durch. Deswegen ist es möglich – und zulässig - , dass der Status sich von 1 (ready) sofort auf 3 (running) ändert.

Der Befehl „GUS_CloseTest“ schließt die Testdatei und der Status ändert sich von 1 (ready) auf 0 (open). Der Test ist nicht mehr geladen, aber das Gerät bleibt verfügbar und die Kommunikation ist noch hergestellt.

Status 2: PreTest Running

Wurde der Selfcheck erfolgreich durchgeführt, ändert sich der Status von 2 (PreTest Running) auf 3 (running).

In dem Status 2 (PreTest Running) kann auch der Befehl „GUS_StopTest“ gegeben werden. Damit kann jederzeit das Gerät gestoppt werden (z.B. in einem Notfall). Der Befehl „GUS_StopTest“ führt das Gerät zum Status 1 (ready) zurück. Die Testdatei ist immer noch geladen.

Wie schon erwähnt, führen nicht alle Geräte einen Pre-Test (oder Selfcheck) durch. Möglicherweise befindet sich ein Gerät nie im Status 2 (PreTest Running) und wechselt sofort auf Status 3 (running).

Status 3: running

Nachdem das Gerät in Status 1 den Befehl „GUS_StartTest“ erhalten hat, startet die Prüfung und das Gerät erreicht den Status 3 (running). Mit dem Befehl „GUS_StopTest“ kann die Prüfung gestoppt werden. Der Status wechselt auf 1 (ready) und die Prüfung kann neu gestartet werden.

Eine Prüfung kann unterbrochen und fortgesetzt werden durch die Befehle „GUS_PauseTest“ bzw. „GUS_ContinueTest“.

Status 4: finished

Wurde die Prüfung ohne Störung erfolgreich durchgeführt, ändert sich der Status auf 4 (finished). Ein Gerät erreicht nur den Status 4 (finished), wenn der Test erfolgreich ohne Störung durchgeführt wurde.

Hinweis: Tritt während einer laufenden Prüfung eine Störung auf, geht das Gerät auf Status 5 (pause). Damit hat der Operator die Möglichkeit, den Test abzubrechen oder ihn – nach Behebung des Fehlers – weiter laufen zu lassen.
Hier ist zwischen einer Störung und einem Fehler zu unterscheiden. Eine Störung deutet auf eine Betriebsstörung hin, die möglicherweise durch den Anwender behoben werden kann. Ein Fehler ist typischerweise ein Versagen des Gerätes, wenn z.B. die Sicherheitsverriegelung ausgelöst wird (interlock tripped), was auf ein ernsthaftes Problem oder einen Notstopp hinweist, der nicht durch den Anwender schnell behoben werden kann.

In Status 4 (finished) ist der Test noch geladen. Der Befehl „GUS_CloseTest“ schließt die Testdatei und versetzt das Gerät zurück in Status 0 (open). Ein neuer Test kann geladen werden.

Der Befehl „GUS_StopTest“ setzt das Gerät in Status 1 (ready) zurück. Der Test kann neu gestartet werden.

Status 5: pause

Nach dem Befehl „GUS_PauseTest“ erreicht das Gerät den Status 5 (pause). Manchmal muss auch in diesem Status die Prüfung gestoppt oder unterbrochen werden. Der Befehl „GUS_StopTest“ setzt das Gerät in Status 1 (ready) zurück. Der Test ist immer noch geladen. Die Prüfung kann neu gestartet oder beendet werden, wie oben beschrieben.

Während des Status 5 (pause) werden die Prüfzeit und weitere Prüfparameter nicht zurückgesetzt. Der Test kann einfach mit dem Befehl „GUS_ContinueTest“ weitergefahren werden. Dann wird z.B. die Prüfzeit weiter aktualisiert. Während der Pause-Zeit zählt die Durchlaufzeit weiter.

Der Befehl „GUS_StopTest“ stoppt den Test und bringt das Gerät in Status 1 (ready) zurück.

Status -1: error

In jedem Status kann ein Problem auftreten. Wenn sich das Gerät in Status 1, 2, 3, 4 oder 5 befindet und eine Gerätestörung (z.B. ein Hardware-Versagen) auftritt, ändert sich der Status auf -1. Der Status -1 bezieht sich also wirklich auf das Versagen des Gerätes. Der Test ist immer noch geladen. Mit dem Befehl „GUS_CloseTest“ wird die Testdatei geschlossen und der Status wird gleich 0 (open). Die Kommunikation mit dem Gerät läuft noch, aber der Test ist nicht mehr geladen.

Der „error“ Status ist typischerweise ein Hardwareproblem (defektes Leistungsmodul im Leistungsverstärker / Notaus / Interlock tripped / Max. Temperatur der Klimakammer erreicht / …)

„GUS_StopTest“ Befehl

Dieser Befehl bringt das Gerät in Status 1 (ready) zurück. Dieser Befehl ist verfügbar, wenn sich das Gerät in Status 2 (pre-test running), 3 (running), 4 (finished) oder 5 (pause) befindet. Die Testdatei bleibt geladen oder geöffnet.

„GUS_CloseTest“ Befehl

Dieser Befehl bringt das Gerät in Status 0 (open) zurück. Dieser Befehl ist verfügbar, wenn sich das Gerät in Status 1 (ready), 4 (finished) oder -1 (error) befindet. Die Testdatei wird geschlossen.

Befehl, der nicht mit einem Status übereinstimmt

Falls ein Befehl gegeben wird, der sich unterscheidet von einem der Befehle für einen bestimmten Status gemäss der Dokumentation „State Machine“, ändert sich der Status des Gerätes nicht.
Zum Beispiel muss das Gerät, wenn es sich in Status 1 (ready) befindet und der Befehl „GUS_PausTest“ gegeben wird, mit „ERR“ antworten (statt „ACK“). Das Gerät bleibt in dem aktuellen Status 1 (ready). Die aktuelle Implementierung des übergeordneten Programms liest jeden „ACK“-String als eine positive Bestätigung des gegebenen Befehls. Jeder sonstiger String wird als Fehler erkannt, was darauf hindeutet, dass der Befehl für den aktuellen Status des Gerätes nicht gültig oder zutreffend war. Der Status des Gerätes ändert sich nicht.

Befehlsbestätigung

Nach jedem erfolgreichen Verfahren muss jeder Befehl mit dem String „ACK“ bestätigt werden, sofern nicht anders definiert. Wenn das Verfahren nicht erfolgreich durchgeführt werden kann, muss die Antwort der String „ERR“ sein.

Minimaler Befehlssatz

Für den einfachen automatischen Ablauf mittels des Start- und Stopp-Verfahrens durch ein übergeordnetes Programm, inklusive Überwachung der Geräte während des Prüfablaufes, definiert die folgende Liste den minimalen Befehlssatz, der in der Standard GUS Schnittstelle des Gerätes implementiert werden muss:

- GUS_Open_App
- GUS_CloseApp
- GUS_OpenDevice
- GUS_CloseDevice
- GUS_PrepareTest
- GUS_StartTest
- GUS_StopTest
- GUS_PauseTest
- GUS_ContinueTest
- GUS_CloseTest
- GUS_GetStatus

Weitere Einzelheiten werden in Anhang II beschrieben.

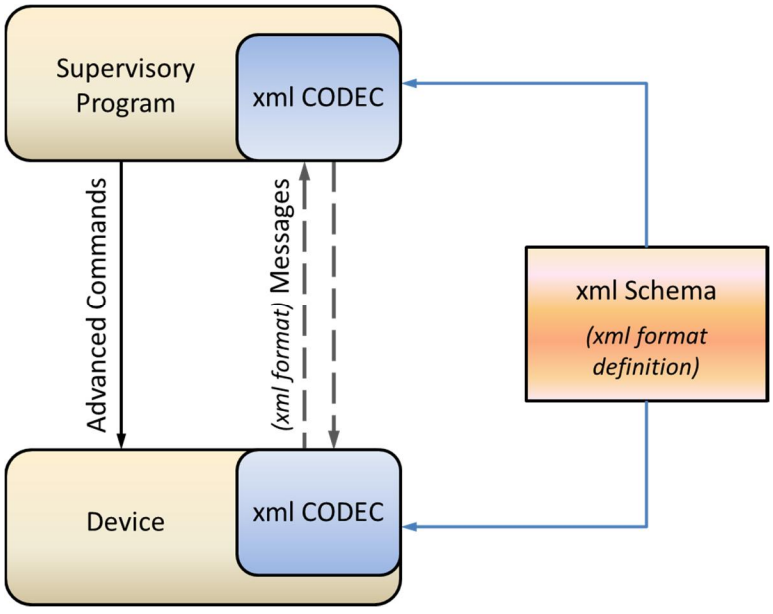
Mittels des minimalen Befehlssatzes kann das übergeordnete Programm die Kommunikation mit dem Gerät aufbauen, das Gerät beauftragen, den Test zu laden, zu starten usw. Wenn die Kommunikation mit den unterschiedlichen Geräten aufgebaut ist, kann das übergeordnete Programm die Tests der Geräte starten (Schwingtisch, Klimakammer, Steuergerät etc.), den Test eines Gerätes stoppen oder pausieren, weiterfahren oder alle Tests anhalten, wenn ein Gerät eine Störung meldet. Es ist der minimale Befehlssatz, um eine kombinierte Prüfung ohne Beaufsichtigung durch einen Anlagenbediener laufen zu lassen.

Erweiterter Befehlssatz

Wenn das übergeordnete Programm in der Lage sein muss, den Wert bestimmter Geräte-Parameter oder Variabler zu erhalten oder wenn für die Synchronisierung eines Tests detailliertere Informationen der Geräte erforderlich sind, muss der erweiterte Befehlssatz implementiert werden. Der erweiterte Befehl „GUS_SetParameter“ bietet dem übergeordneten Programm sogar die Möglichkeit, einen bestimmten Wert im Gerät zu bestimmen, wenn der entsprechenden Geräte-Parameter durch das Gerät als verfügbar und veränderbar deklariert wurde.

- GUS_GetDeviceInfo
- GUS_GetInfo
- GUS_GetParameter
- GUS_SetParemeter

Durch diese Befehle sind die zwischen dem übergeordneten Programm und den Geräten ausgetauschten Informationen komplexer und benötigen eine Struktur, die das Format und den Typ der Informationen, EU (Engineering Unit), Grenzwerte, Einschränkungen, usw. bestimmt.



Deshalb ist die Kommunikation der obengenannte Befehle in einem XML-Format strukturiert. XML (Extendible Markup Language) ist eine erweiterbare Auszeichnungssprache, die eine Reihe von Regeln definiert, um Dokumente in einem Format zu kodieren, das sowohl von Menschen als auch von Maschinen lesbar ist. Das Format wird durch die W3C XML 1.0 Spezifikation und weitere dazu gehörige Spezifikationen definiert, die alle lizenzkostenfreie, offene Standards sind.

Ein XML-Schema definiert die Regeln, wie die XML-Sprache der Nachrichten zwischen dem übergeordneten Programm und den Geräten strukturiert ist. Das XML-Schema wird sowohl vom übergeordneten Programm als auch vom Gerät verwendet, um den XML-formatierten Strom von Daten oder Nachrichten zu Codieren und zu Decodieren (daher der Name CODEC). Der CODEC muss die Regeln des XML-Schemas erfüllen.

Hinweis: Eine XML-Datei, die ein Schema darstellt, hat die Endung „.xsd“. Das vorgeschlagene Schema für den erweiterten Befehlssatz findet man in der Datei „GusDeviceInfo.xsd“. Siehe auch Anhang IV für weitere Informationen über das Schema.

Wirkprinzip:

a. Das übergeordnete Programm sendet den Befehl „GUS-GetDeviceInfo“ zum Gerät. Das Gerät antwortet mit einem XML-String, der die Information, die zwischen übergeordnetem Programm und Gerät ausgetauscht werden kann, definiert. Der Inhalt ist eine strukturierte Liste von Namen der Variablen und Geräteparametern, deren entsprechende Einschränkungen, Grenzwerte, EU (Engineering Unit), ob diese Geräteparameter oder Variablen schreibgeschützt sind oder nicht, usw.
b. Wenn das übergeordnete Programm den Befehl „GUS_GetInfo“ schickt, antwortet das Gerät mit allen aktuellen Werten der Geräteparameter und Variablen, die vorher als Antwort auf „GUS_GetDeviceInfo“ deklariert wurden.
c. Der Befehl „GUS_GetParameter“ definiert eindeutig den Geräteparameter oder die Variable, wofür das Gerät den aktuellen Wert zurücksenden muss. Siehe Anhang V für weitere Informationen.
d. Der Befehl „GUS_SetParameter“ definiert eindeutig den Geräteparameter oder die Variable, wofür das Gerät den neuen Wert setzen muss. Der Befehl „GUS_SetParameter“ ermöglicht dem übergeordneten Programm, die Kontrolle über die Gerätezustandsgrößen zu übernehmen: Definition eines neuen Sollwertes, Öffnen und Schließen eines Relais, Setzen eines analogen Ausgangs etc.. Siehe Anhang V für weitere Informationen.

XML Format

Referenz: https://en.wikipedia.org/wiki/XML und http://www.w3schools.com/xml/xml_whatis.asp

Oben genannten Definitionen ist für die Implementierung des erweiterten Befehlssatzes zu folgen. Für weitere Einzelheiten zum XML Format: siehe Anhang III.

Anhang IV gibt eine detaillierte Beschreibung der „Definition der Parameter in der XML-Datei des erweiterten Befehlssatzes“.

Eine detaillierte Beschreibung des erweiterten Befehlssatzes wird in Anhang V gegeben. Dieser Anhang beschreibt das XML-Format, um Informationen zwischen dem übergeordneten Programm und dem Gerät auszutauschen.

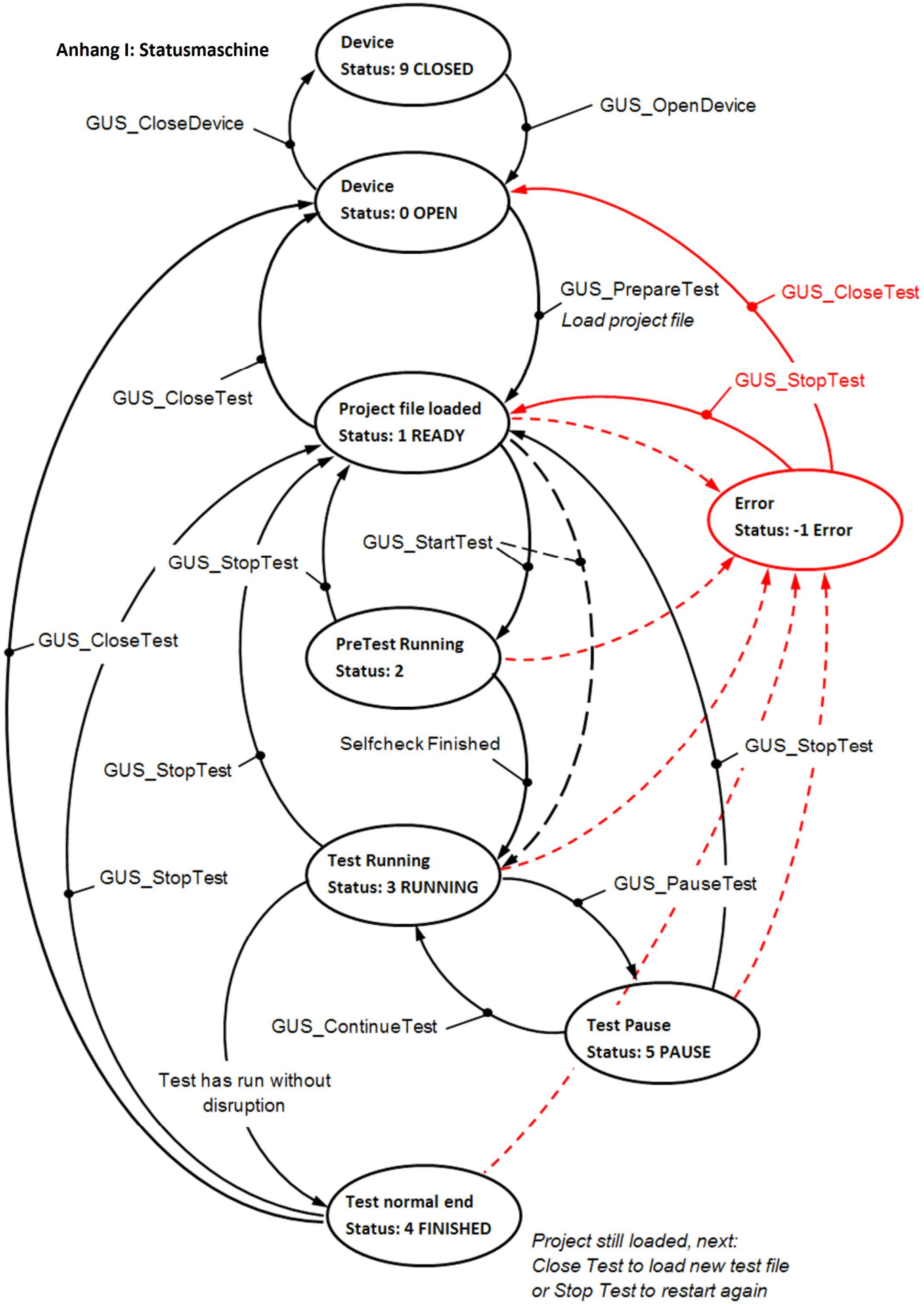
Datums- und Zeitformat

Für die Kodierung und Darstellung von Zeit und Datum im XML-Format gilt der Standard ISO 8601.

Mehr Einzelheiten sind zu finden auf: https://en.wikipedia.org/wiki/ISO_8601

Ein Überblick über die allgemeinen Prinzipien ist in Anhang VI enthalten.

Anhang I: Statusmaschine



Überblick der zugelassenen Übergänge: „Status von“ nach „Status zu“

State to / state from		9 closed	0 open	1 ready	2 pre-test running	3 running	4 finished	5 pause	6 busy	-1 error
9	closed	X	1							
0	open	2	X	3						
1	ready		8	X	4	4				ERR
2	pre-test running			5	X					ERR
3	running			5		X	END	6		ERR
4	finished		8	5			X			ERR
5	pause			5		7		X		ERR
6	busy								X	
-1	error		8	5						X

1	GUS_OpenDevice
2	GUS_CloseDevice
3	GUS_PrepareTest
4	GUS_StartTest
5	GUS_StopTest
6	GUS_PauseTest
7	GUS_ContinueTest
8	GUS_CloseTest

Hinweise:

End : Wenn sich das Gerät in Status 3 (running) befindet, und die Prüfung läuft normal zu Ende, ändert sich der Status auf 4 (finished).

ERR : Wenn sich das Gerät in Status 1 (ready), 2 (pre-test running), 4 (finished) oder 5 (pause) befindet, und eine Fehler passiert, ändert sich der Status auf -1 (error). Für die Definition von „error“ oder „Fehler“ siehe den erklärenden Text oben. Ein Fehler deutet üblicherweise auf ein Hardware-Versagen des Gerätes hin, das nicht leicht durch den Anwender zu beheben ist.

Status 6 : „Busy“

Im Standard 1.0 wurde der Status 6 (busy) eingeführt für den Fall, dass das Gerät irgendetwas tut, was nicht unmittelbar mit einem aktuellen Status verbunden ist.. Zum Beispiel das Öffnen einer Datei, das Schreiben von Daten auf die Festplatte, die Vornahme eines Übergangs etc. Dieser Status ist nicht als „einzigartiger“ Status definiert und steht so, wie er ausgelegt wurde, nicht mit einem spezifischen Zustand des Gerätes in Zusammenhang. Der Status kann in der Schnittstelle des Gerätes implementiert werden, wird allerdings durch das übergeordnete Programm ignoriert. Wenn der Status 6 (busy) durch das Gerät gemeldet wird, wartet das übergeordnete Programm, bis das Gerät zu einem anderen Status wechselt.

Wenn der Status 6 (busy) auftritt, besteht die potentielle Gefahr, dass das Gerät in Status 6 (busy) verbleibt, d.h. das Gerät bleibt hängen und kann nicht wiederhergestellt werden.

Anhang II: Minimaler Befehlssatz

Command	Parameter included in the command	Response Format	Response (success)	Response (Fail)
GUS_Open_App	Driver name	string	ACK	ERR
GUS_CloseApp			none	
GUS_OpenDevice	Device ID (number)	string	ACK	ERR
GUS_CloseDevice		string	ACK	ERR
GUS_PrepareTest	Pfad\Prüfprofil	string	ACK	ERR
GUS_StartTest		string	ACK	ERR
GUS_StopTest		string	ACK	ERR
GUS_PauseTest		string	ACK	ERR
GUS_ContinueTest		string	ACK	ERR
GUS_CloseTest		string	ACK	ERR
GUS_GetStatus		string	State Number(*)	ERR

(*) übereinstimmend mit der Statusmaschine

Anhang III: XML-Format

XML-Dokumente bestehen ausschließlich aus Zeichen des Unicode--Zeichensatzes.

Für die Liste der gültigen Zeichen siehe: https://en.wikipedia.org/wiki/Valid_characters_in_XML

Wohlgeformte Dokumente

Die XML-Spezifikation bestimmt ein Dokument als wohlformatierten Text – was bedeutet, dass es eine Liste von Syntaxregeln wie in der Spezifikation vorgesehen einhält. Einige Kernpunkte der ziemlich langen Liste sind:

- Das Dokument enthält nur richtig kodierte, gültige Unicode-Zeichen.
- Keine der besondere Syntaxzeichen wie < und & tauchen auf, ausgenommen wenn diese ihre Funktion als Auszeichnungselement erfüllen.
- Die Anfangs-, Ende- und Empty-Element-Tags, die die Elemente abgrenzen, sind richtig geschachtelt und es gibt keine Auslassungen oder Überlappungen.
- Die Element-Tags unterscheiden Groß- und Kleinschreibung; die Anfangs- und Ende-Tags müssen genau übereinstimmen.
- Die Tags dürfen keine der Zeichen !"#$%&'()*+,/;<=>?@[\]^`{|}~ enthalten, auch kein Leerzeichen, und können nicht mit -, ., oder einer numerischen Ziffer beginnen.
- Ein einzelnes "root" Element enthält alle anderen Elemente.

Escaping

XML bietet Auswegmöglichkeiten für die Übernahme von Zeichen, die schwierig direkt einzugeben sind. Zum Beispiel:

- Die Zeichen „<“ und „&“ sind Schlüsselsyntaxmarken und dürfen nie im Inhalt außerhalb eines CDATA-Bereiches vorkommen.
- Einige Zeichencodierungen unterstützen nur eine Teilmenge von Unicode. Beispielsweise kann ein XML-Dokument in ASCII kodiert werden, aber ASCII fehlen Kodierpunkte für Unicode-Zeichen wie „é“.
- Möglicherweise fehlt ein Zeichen auf der Tastatur des Anwenders.
- Manche Zeichen haben eine Glyphe, die man visuell nicht von anderen Zeichen unterscheiden kann. Beispiele sind:
    - Geschütztes Leerzeichen ( ) " "
      vergleiche Leerzeichen ( ) " "
    - Kyrillischer Großbuchstabe A (А) "A"
      vergleiche Lateinischer Großbuchstabe A (A) "A"

Es gibt fünf vorgegebene Entitäten:

- < steht für: <
- > steht für: >
- & steht für: &
- ' steht für: '
- " steht für: "

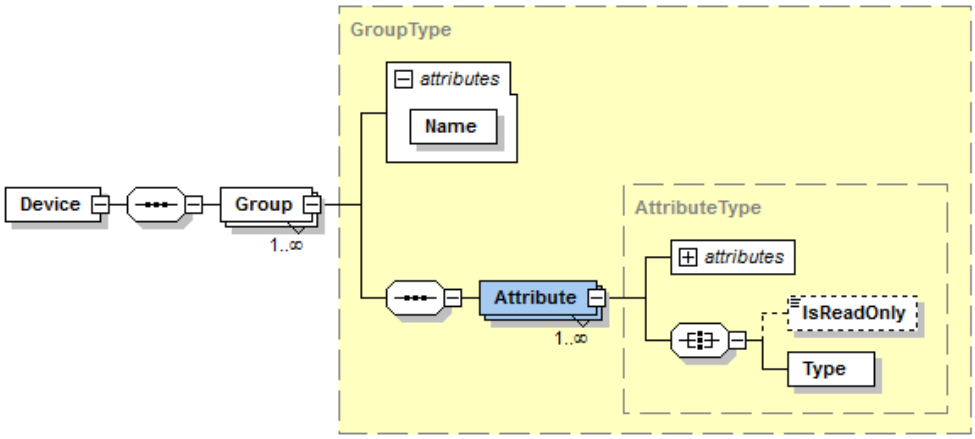
Alle zugelassenen Unicode-Zeichen können durch eine numerische Zeichenreferenz dargestellt werden. Betrachten Sie das chinesische Zeichen "中", mit dem numerischen Kode 4E2D hexadezimal oder 20013 dezimal. Wenn die Tastatur eines Anwenders nicht über dieses Zeichen verfügt, kann es in ein XML-Dokument als Code 中 oder 中 eingefügt werden. Ähnlich kann der String "I <3 Jörg" für eine Einbindung in ein XML-Dokument als "I <3 Jörg" kodiert werden.

Allerdings ist "&#0;" nicht zulässig, weil das Zeichen 0 eines der Steuerzeichen ist, die von XML ausgeschlossen sind, auch wenn eine numerische Zeichenreferenz verwendet wird. Eine alternative Zeichenkodierung wie Base64 ist für die Darstellung solcher Zeichen notwendig.

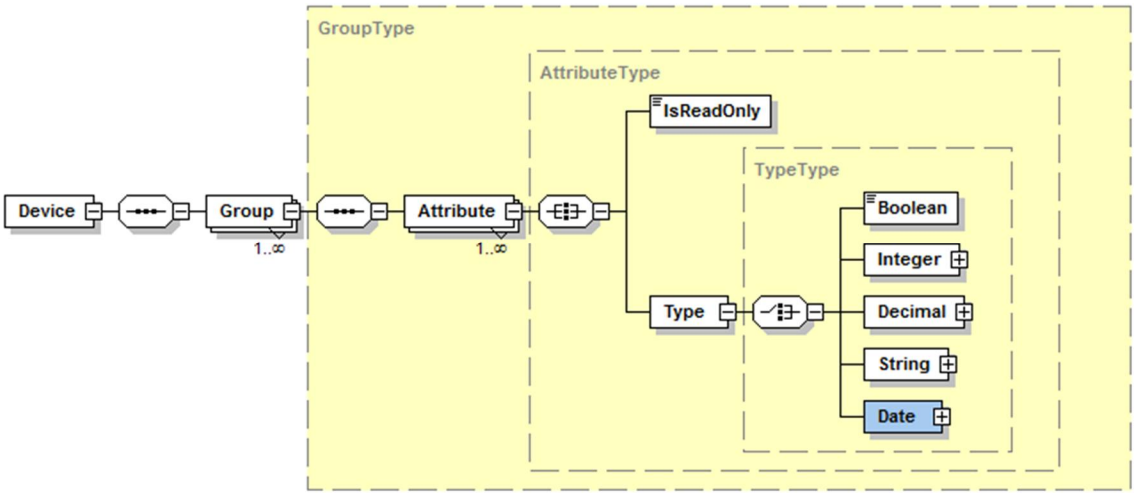
Anhang IV: Definition der Parameter im XML-Schema des erweiterten Befehlssatzes

Referenz: Datei GUSDeviceInfo.xsd

Nach der Definition der GUS Standard Version 2.0 sind die Variablen und Parameter, die ein Gerät definieren, in Gruppen unterteilt:



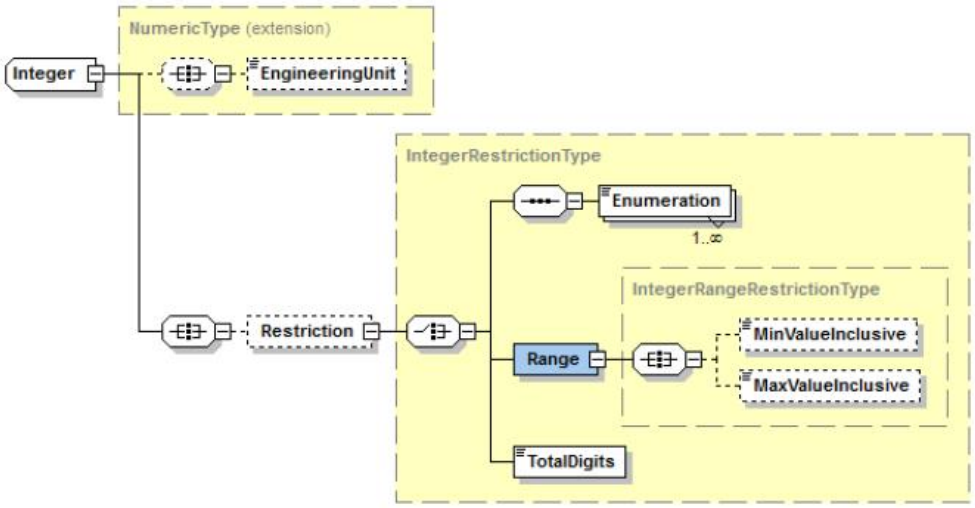
Die Anzahl von Gruppen ist nicht begrenzt. Typischerweise enthält die erste Gruppe die allgemeinen Angaben über das Gerät wie Name, Hersteller, Typ oder Modell etc. Die zweite Gruppe enthält die Kontrollparameter und Variablen (wie Sollwert, aktueller Wert etc.). Die dritte Gruppe enthält zusätzliche Ein- und Ausgänge etc. Jede Gruppe muss ein Attribut NAME haben. Der Wert dieses Attributs (= Gruppenname) muss für alle in XML definierten Gruppen einmalig sein. Gruppen werden verwendet, um auf logische Art die Attribute (= Geräteparametern und Variablen) zu bündeln und zum Zweck des „Namespacing“ (= einzigartige Identifizierung von Attributen mit dem gleichen Namen).



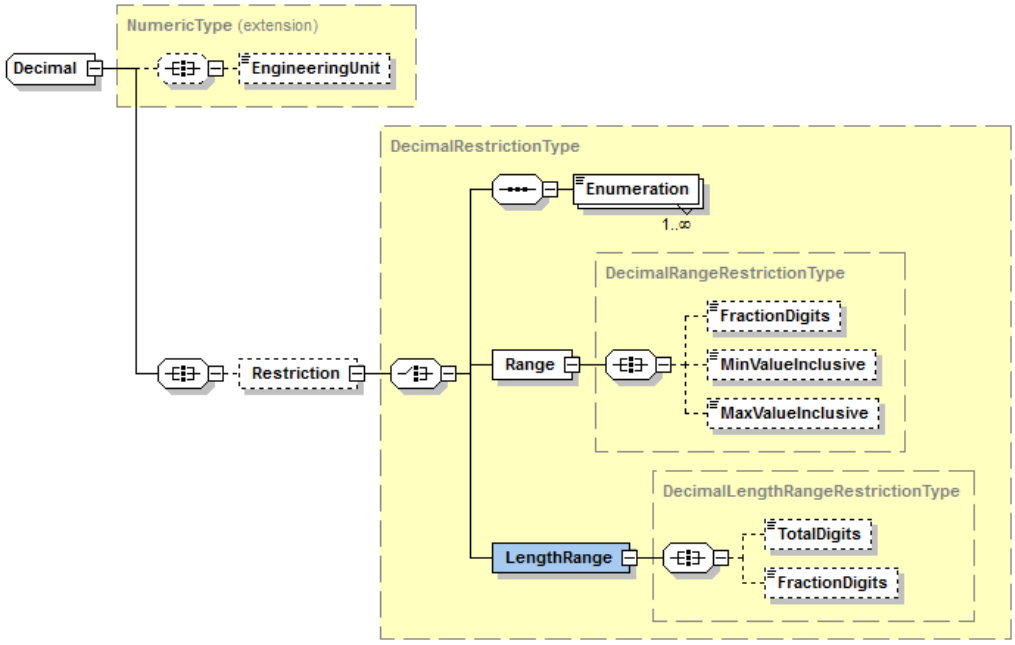
Gruppen enthalten ein oder mehrere Attribute. Jedes Attribut muss ein Attribut NAME haben. Der Wert dieses Attributs muss für alle Attribute, die in einer Gruppe definiert sind, einzigartig sein. Das Kontrollprogramm adressiert das Attribut (= Geräteparameter oder Variable) mit seinem Namen (kombiniert mit dem Gruppennamen). Ein Attribut kann als schreibgeschützt definiert werden oder nicht, wodurch bestimmt wird, ob das übergeordnete Programm den Wert ändern kann oder nicht. Siehe Anhang V für einige Beispiele: „GUS_GetParameter“ und „GUS_SetParameter“.

Ein Attribut hat ein Element namens „Type“, das von der Art her boolesch, integer, dezimal, ein String oder Datum sein kann. Jede Type wird in diesem Abschnitt weiter besprochen.

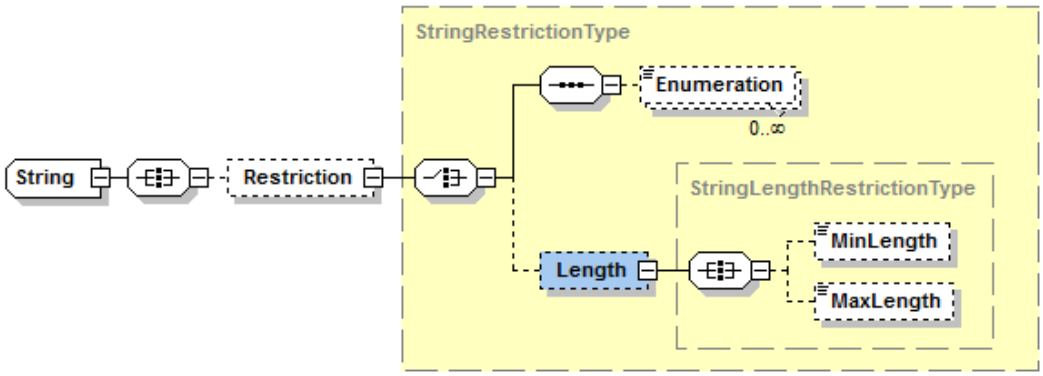
Der ***Boolean Typ*** kann 0 oder 1, false oder true sein.



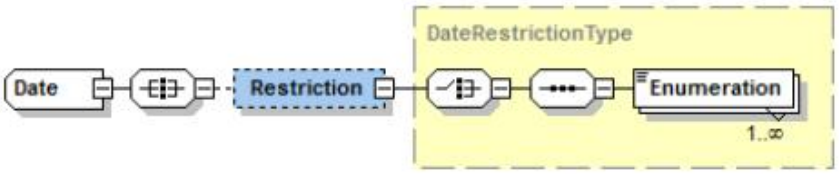
Dem ***Integer Typ*** kann neben seiner Einschränkungen wie minimaler und maximaler Wert, Gesamtzahl der Digits und eventuell eine begrenzte Anzahl von Werten auch eine EU (Engineering Unit) zugewiesen werden.



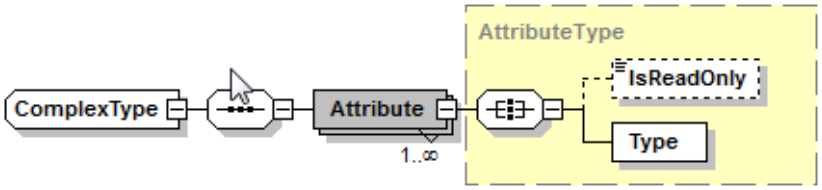
Die Definition des ***Dezimalen Typs*** ist ähnlich wie der integer Typ mit Ausnahme der Tatsache, dass auch die Nachkommastellen definiert werden können. Für viele Anwendungen macht es Sinn, die Nachkommastellen auf 1 zu begrenzen (Temperatur, Feuchte, gRMS, …).



Die Definition eines ***String*** ist einfach. Manchmal macht es Sinn, den möglichen Inhalt der übertragenen Nachrichten mit vordefinierten Text-Strings zu begrenzen. Dazu sollte eine Aufzählung (= enumeration) verwendet werden.



Die Formatierung des ***Datums*** ist in Anhang VI beschrieben. Der Standard gemäß ISO 8601 wird eingehalten.



Letztendlich kann der Typ von einem Attribut auch ein ***Komplexer Typ*** sein: das Ziel dieses Typs sind verschachtelte (= nested) Attribute. Der Wert NAME jedes (verschachtelten) Attributs muss für alle Attribute des ComplexType einzigartig sein.

Anhang V: Vorinformationen über den erweiterten Befehlssatz

GUS_GetDeviceInfo

Wenn das übergeordnete Programm diesen Befehl an das Gerät sendet, antwortet dieses mit einer XML-Datei, die die Definition aller Geräteparameter enthält. Im vorgeschlagenen Schema von XML-Dateien für Shaker und Klimakammer sind folgende Gruppen definiert:

- DeviceInfo: Enthält die allgemeinen Angaben über das Gerät. Name, Hersteller, Typ, Modell, Seriennummer und Hinweise.
- ControlledValues: Sollwert und aktuelle (Mess-) Werte, durch das Gerät kontrolliert oder geregelt. gRMS, Temperatur, Feuchte, … Ein Sollwert kann durch das übergeordnete Programm bestimmt werden oder schreibgeschützt sein.
- Measurements: Zusätzliche Ein- und Ausgänge des Gerätes. Diese Ein- und Ausgänge können analog oder digital sein, können schreibgeschützt sein oder durch das übergeordnete Programm bestimmt werden; sie sind für weitere Kontroll- oder Messaufgaben verfügbar.
- Operation: Diese Geräteparametern beschreiben, welcher Test läuft (z.B. Sinus, Rausch etc. für den Schwingungsregler und Temperatur, Feuchte etc. für die Klimakammer).
- Message: Die in dieser Gruppe definierten Nachrichten fügen zusätzliche Informationen zur allgemeinen Kommunikation zwischen übergeordnetem Programm und Gerät hinzu: Warnung, Alarm etc.
- Testing: dieser Geräteparametersatz stellt zusätzliche Information über den laufenden Test dar: Durchlaufzeit, Restlaufzeit, Step-Nummer im Testprogramm, Wiederholung etc.

Detaillierte Beispiele sind in den Dateien „***Gus_Device_Shaker_V01.xml***“ und „***Gus_Device_Chamber_V01.xml***” für den Shaker bzw. die Klimakammer enthalten.

Es ist darauf hinzuweisen, dass die Informationsliste verkürzt oder erweitert werden kann. Die Struktur der XML-Datei ist so flexibel, dass neue noch nicht definierte Parametern hinzugefügt oder Parameter, die nicht verfügbar sind, ausgelassen werden können.

Wenn das Gerät mit dieser XML-Datei antwortet, kennt das übergeordnete Programm alle Parameter, die verfügbar sind und durch das Gerät gesendet oder empfangen werden können. Die XML-Datei definiert das Format, eventuelle Einschränkungen, EU (Engineering Unit) etc. für jeden Parameter.

GUS_GetInfo

Dieser Befehl fragt alle aktuelle Werte der verfügbaren Parameter vom Gerät ab. Das Gerät antwortet dem übergeordneten Programm mit einer XML-Datei, die wie oben (GUS_GetDeviceInfo) beschrieben strukturiert ist, aber diesmal nur mit der Liste der Parameter und deren entsprechenden Werten. Natürlich müssen die Werte das Format der XML-Datei, die als Antwort auf den Befehl „GUS_GetDeviceInfo“ gesendet wird, einhalten.

GUS_GetParameter

Statt der Abfrage aller Informationen gemeinsam über den Befehl „GUS_GetInfo“ ist es in vielen Fällen sinnvoll, den Wert für nur einen spezifischen Geräteparameter oder Variable abzufragen. Der Befehl „GUS_GetParameter“ ermöglicht dem übergeordneten Programm die Abfrage eines ganz bestimmten Wertes.

Der Parameter des Befehls (= Call des übergeordnetes Programms) ist ein String, der eindeutig den Geräteparameter oder die Variable definiert, für den/die der aktuelle Wert abgefragt wird. Das Format des Strings ist das XML-Format. Inhalt, Name und Struktur sind durch das Gerät in der Antwort auf „GUS_GetDeviceInfo“ definiert.

Beispiel der Abfrage des aktuellen (Soll-) Wertes für die Temperatur der Klimakammer:

```
<Device>
        <ControlledValues>
                <Temperature>
                        <CurrentValue></CurrentValue>
                </Temperature>
        </ControlledValues>
</Device>
```

Im Beispiel oben muss der Parameter (= String) des Befehls „GUS_GetParameter“ an die Klimakammer zur Abfrage des aktuellen Wertes wie folgt sein:

```
<Device><ControlledValues><Temperature><CurrentValue></CurrentValue></Temperature></ControlledValues></Device>
```

Das Gerät antwortet in der gleichen Struktur zur Bestimmung des Parameters, aber diesmal mit dem aktuellen Wert des Parameters (im Beispiel 101,4 °C):

```
<Device><ControlledValues><Temperature><CurrentValue>101.4</CurrentValue></Temperature></ControlledValues></Device>
```

Wenn der Geräteparameter oder die Variable nicht existiert, muss das Gerät mit „ERR“ antworten.

GUS_SetParameter

Das übergeordnete Programm kann den Wert von manchen Parametern bestimmen. Dies ist nur möglich und zulässig, wenn die Besonderheit des Parameters richtig definiert ist:

```
<IsReadOnly>false</IsReadOnly>
```

Ausgehend vom Beispiel oben könnte der Parameter des Befehls des übergeordneten Programms sein:

```
<Device>
        <ControlledValues>
                <Temperature>
                        <DemandValue>150.0</DemandValue>
                </Temperature>
        </ControlledValues>
</Device>
```

Dann der Parameterstring des Befehls „GUS_SetParameter“ muss sein:

```
<Device><ControlledValues><Temperature><DemandValue>150.0</DemandValue></Temperature>
</ControlledValues></Device>
```

Dieser Befehl setzt den Sollwert der Temperatur in der Klimakammer auf +150.0°C fest. Wenn der Geräteparameter oder die Variable nicht existiert oder schreibgeschützt ist, muss das Gerät mit „ERR“ antworten.

Anhang VI: Zeit- und Datumsformat

Allgemeines Prinzip der ISO 8601

- Datum und Zeit sind nach Größe geordnet, also von der größten bis zur kleinsten Zeiteinheit: Jahr, Monat, (eventuell Woche), Tag, Stunde, Minute, Sekunde
- Jedes Datum und jeder Zeitwert haben eine feste Stellenanzahl, die mit führenden Nullen aufgefüllt werden muss.
- Die Darstellung kann in einem von zwei Formaten erfolgen – im Grundformat mit minimaler Anzahl Trennzeichen oder im erweiterten Format mit Trennzeichen für verbesserte Lesbarkeit.
- Für eine reduzierte Genauigkeit kann die Anzahl von Datums- und Zeitwerten verringert werden, aber in der Reihenfolge von niedrigst- zu höchstwertig.
- Wenn für eine bestimmte Anwendung erforderlich, unterstützt der Standard auch die Ergänzung mit einer Dezimale zur Darstellung des kleinsten Zeitwerts.
- Wochen werden von 01 bis 53 durchnummeriert und ihnen wird ein „W“ vorangestellt.

Beispiele

Datum und Zeit kombiniert:

YYYMMDDThhmmss.sss oder YYYY-MM-DDThh:mm:ss.sss

YYYMMDDThhmm oder YYYY-MM-DDThh:mm

Beispiel Datum:

YYYYMMDD oder YYYY-MM-DD

Ausnahmefall: YYYYMM ist nicht erlaubt. Es muss immer lauten: YYYY-MM

YYYYWxx or YYYY-Wxx (Hinweis: xx = Wochennummer 01 bis 53)

YYYYWxxD or YYYY-Wxx-D (Hinweis: D = Wochentag nummeriert 1 bis 7)

Beispiel Zeit:

hhmm oder hh:mm

hhmmss oder hh:mm:ss