DISCOM PRÜFSTANDS-KOMMANDOS

Inhalt

Generelles zur Steuerung des Mess-Systems

In einer Prüfstandsumgebung steht das Messsystem mit einer Prüfstandssteuerung in Verbindung. Diese liefert Informationen wie z.B. den geladenen Typ (Prüfvorschrift) oder den aktuellen Prüfschritt. Das Messsystem liefert Informationen wie z.B. das Bewertungsergebnis.

Die Kommunikation zwischen der Prüfstandssteuerung und dem Messprogramm *TasAlyser* wird durch den Austausch von Text-Kommandos realisiert. Die Text-basierte Kommunikation hat drei wesentliche Vorzüge gegenüber anderen Methoden wie z.B. einem bit-parallelen Interface:

- **Sicherheit**: Da der Austausch von Kommandos und Antworten zwischen Messprogramm und Prüfstand direkt beobachtet und überwacht werden kann, können jegliche Protokollfehler sehr einfach gefunden werden. Es entstehen keine Zweifel über die Reihenfolge von Kommandos, und fehlende oder falsch verwendete Befehle werden sofort entdeckt.
- **Flexibilität**: Das Text-basierte Protokoll kann sehr einfach um neue Kommandos erweitert werden. Auf Seiten des *TasAlysers* gibt es einen „Plug-In"-Mechanismus, mit dem Spezialkommandos und Sonderlösungen realisiert werden können, die nicht für jedes Projekt benötigt werden. Wenn der Bedarf für ein neues Kommando entsteht, kann dieses hinzugefügt werden, ohne die bestehende und getestete Kommunikation zu beeinflussen.
- **Unabhängigkeit von der Hardware**: Das Protokoll ist unabhängig von der zum Austausch der Texte verwendeten Hardware-Verbindung. Dieser physische Austausch der Kommandos geschieht üblicherweise über das UDP-Netzwerk-Protokoll oder über eine serielle RS232-Verbindung. Alternativ ist auch die Verwendung eines Profibus-Speicherbereichs möglich, über den die Kommando-Texte und -Antworten ausgetauscht werden. Weitere Details dazu stehen im Anhang ab Seite 32*ff*.

Grundsätzlich arbeitet der Messrechner als passives System, das nur auf Kommandos von der Prüfstandssteuerung antwortet, aber selbst keine Kommunikation initiiert. Die Prüfstandssteuerung muss daher nicht auf „asynchrone" Anfragen des Messrechners vorbereitet sein. Allerdings muss die Prüfstandssteuerung die Quittierung jedes von ihr gesendeten Kommandos korrekt verarbeiten.

Dieses Dokument beschreibt in seinem ersten Teil (ab Seite 4) den Standard-Kommandovorrat des TasAlysers, der in jedem Projekt verfügbar ist. Danach folgen (ab Seite 13) Beispiele und weitere Erläuterungen. Der Standard-Kommandovorrat kann, wie oben erwähnt, durch sogenannte Plug-Ins erweitert werden. Die am häufigsten verwendeten Plug-Ins werden im zweiten Teil dieses Dokuments beschrieben. Im Anhang finden sich Hinweise zur Realisierung der Kommunikation über UDP, serielle Schnittstelle und Profibus/net.

Prinzipieller Ablauf einer Prüfung

Ein Prüflauf (d.h. die Prüfung eines einzelnen Aggregats) hat immer folgenden Ablauf:

1. Beginn des Prüflaufs. Dies geschieht, indem die Steuerung den Namen des zu prüfenden Aggregate-Typs (die „Prüfvorschrift“) übermittelt. Dieses Kommando heißt „Insert“.
2. Prüfzustände. Der Prüflauf ist in Prüfschritte unterteilt. Ein Prüfschritt kann z.B. eine Drehzahlrampe sein. Jeder Prüfschritt hat einen Namen, den „Prüfzustand“. Innerhalb eines Prüflaufs können ein oder mehrere Prüfzustände geprüft werden, Prüfzustände können wiederholt oder übersprungen werden. Der Prüfschritt wird durch Übermittlung des Prüfzustands-Namens eingeleitet (Kommando „Mode“) und endet mit Bekanntgabe des nächsten Prüfzustandes oder der expliziten Beendigung des Prüfzustandes.
3. Ende des Prüflaufs: dieser findet in zwei Schritten statt, die aber in einem Befehl zusammengefasst sein können: Ende aller Prüfschritte, und Ende des Prüflaufs. Nach „Ende des Prüflaufs“ speichert das Messsystem die Messergebnisse und Bewertungen und übermittelt diese ggf. an eine Ergebnisdatenbank. Das Ende des Prüflaufs wird durch das „Remove“-Kommando angezeigt.
4. Abfrage der Bewertung: zu jedem Zeitpunkt kann der bisherige Stand der Bewertung abgefragt werden (Kommando „Result“). Auch nach „Ende des Prüflaufs“ können noch alle Ergebnisse des letzten Prüflaufs abgefragt werden.

Weitere Details stehen im Abschnitt „Beispiele“ ab Seite 13. Siehe auch die Grafik auf Seite 16.

Protokoll der Kommunikation

Der TasAlyser empfängt Klartext-Kommandos und beantwortet diese mit Klartext-Meldungen. Ein typisches Kommando besteht aus einem Schlüsselwort, gefolgt von einem Doppelpunkt und Argumenten. Alle Kommandos werden quittiert, die Antwort auf die meisten Kommandos besteht aus einer Zahl (d.h. Ziffer).

Beispiel: Das Kommando „`Measure: 1`“ startet die Messwertaufnahme. Es wird beantwortet mit „`On`“, wenn die Messung gestartet werden konnte, und „`Error`“, wenn das Kommando gescheitert ist (z.B. weil kein gültiger Prüfzustand angewählt ist). Nicht alle Kommandos haben Argumente; in diesem Fall kann der Doppelpunkt weggelassen werden.

Zwischen dem Kommandowort und dem/den Argumenten dürfen beliebig viele Leerzeichen stehen. Bei den Kommandos wird Groß- und Kleinschreibung unterschieden, d.h. das Kommando `MEASURE: ON` ist nicht identisch mit `Measure: On`.

Kommandozeilen, die nicht als gültige Kommandos interpretiert werden können, werden mit „?“ quittiert. Korrekte Kommandos mit ungültigen Argumenten haben eigene Quittierungen, die in der Kommando-Beschreibung aufgeführt ist.

Wird über eine serielle Schnittstelle kommuniziert, so sind die Kommandos im Klartext (gefolgt von einem Zeilenende `cr/lf`) zu übertragen. Die Antwort kommt ebenfalls im Klartext, also z.B. eine Zeile „1“, gefolgt von einem Zeilenende.

Die Ausführung von Kommandos darf sich nicht überlappen. Die Prüfstandssteuerung muss die Quittierung eines Kommandos abwarten, bevor sie das nächste Kommando schickt.

Die Ausführungszeit (und damit die Zeit bis zur Quittierung) für die meisten Kommandos beträgt deutlich unter ½ Sekunde. Nur die Kommandos `Insert:` und `Remove:` (siehe Liste unten) können bis zu einigen Sekunden brauchen, bis sie quittiert werden.

Zeichencodierung

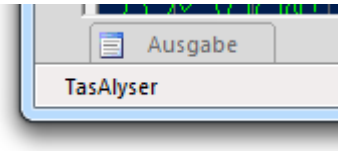
Die Kommandos und Antworten werden als 8 Bit Zeichen unter Verwendung der aktuell im Computer eingestellten Codepage verarbeitet. Beispielsweise können auf einem System, das auf die westeuropäische Codepage 1252 eingestellt ist, auch Umlaute verwendet werden, jedoch keine griechischen Buchstaben.

Protokoll-Varianten

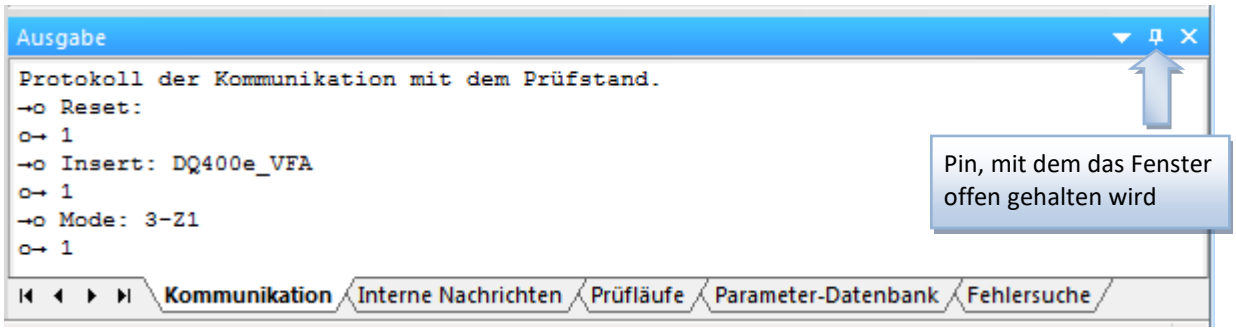
Dieses Dokument beschreibt das „Handshake-Standard“-Protokoll, das sich nur in einigen Kommando-Quittierungen vom „Basis-Standard“-Protokoll unterscheidet. Aus Gründen der Kompatibilität zu alten Prüfstandssteuerungen ist im TasAlyser auch noch das „Dpm42“-Protokoll verfügbar, welches aber nicht Gegenstand dieser Dokumentation ist. Das zu verwendende Protokoll wird in den Decoder-Einstellungen (siehe Seite 17) ausgewählt.

Kommunikation überwachen

Im Ausgabe-Fenster des Messprogramms können Sie die Nachrichten überwachen, die zwischen Prüfstandssteuerung und Messprogramm ausgetauscht werden. Das Ausgabefenster ist normalerweise am unteren Rand des TasAlyser-Hauptfensters angedockt. Klicken Sie auf den Reiter **Ausgabe**, um es zu öffnen. Mit dem „Pin“ in der oberen rechten Ecke des geöffneten Fensters können sie verhindern, dass es automatisch wieder eingefaltet wird (siehe Abbildung unten).



In der Abteilung **Kommunikation** können Sie den Austausch der Kommandos verfolgen. Nachrichten vom Prüfstand werden im Protokoll durch das Symbol →○ gekennzeichnet, die Antworten des Messprogramms durch ○→.



Dauer der Ausführung von Kommandos, Timing

Die Ausführungsdauer der verschiedenen Kommandos hängt von diversen Faktoren ab, die teilweise völlig außerhalb des TasAlysers liegen. Beispielsweise können Windows Netzwerkzugriffe, Virenscanner oder RAID-Controller unvorhersagbare und unregelmäßige Verzögerungen hervorrufen. Auch Festplattenzugriffe können plötzlich verlangsamt sein, wenn Windows durch Energiesparmodi die Speichermedien in den Ruhezustand versetzt hat, was wiederum die Ausführungszeit bestimmter Kommandos verlängert.

Daher sind die hier angegebenen typischen Ausführungszeiten für Kommandos nur als Richtwerte zu verstehen und können niemals garantiert werden. Die beste Herangehensweise ist, einen möglichst langen Ausführungs-Timeout zuzulassen, aber bei Eintreffen einer Antwort vom TasAlyser sofort zu reagieren.

Typische Ausführungszeiten (für Details über die Kommandos siehe die folgenden Abschnitte):

- Der Start eines Prüflaufs (`Insert:`) kann innerhalb von 3 Sekunden erfolgt sein, kann aber auch 10 Sekunden oder länger brauchen (beispielsweise wenn der Benutzer eine Änderung in der Parameterdatenbank gemacht hat). Das `Insert`-Kommando sollte so früh wie möglich im Prüfablauf geschickt werden, so dass möglichst viel Zeit für dessen Ausführung (vor Beginn der eigentlichen Akustikprüfung) zur Verfügung steht.
- Das Beenden des Prüflaufs (`Remove:`) benötigt ebenfalls in den meisten Fällen 3 Sekunden, kann aber deutlich länger dauern (10 Sekunden), wenn beispielsweise der Festplattenzugriff verlangsamt ist. Verwenden Sie zusätzlich das `EndOfTest`-Kommando, um den Zeitbedarf für die Ausführung aufzuteilen.
- Eine Prüfzustands-Änderung (`Mode:`) dauert normalerweise ½ Sekunde oder weniger. Auch hier empfiehlt sich, das `Mode`-Kommando so früh wie möglich zu schicken. Es darf aber nicht zu früh geschickt werden, damit nicht irrtümlich eine Messung getriggert wird. (Beispiel: wenn im Prüfschritt eine Drehzahlrampe verwendet wird, die bei 300 UpM beginnt, so muss sichergestellt sein, dass die Drehzahl nach Eintreffen des `Mode`-Kommandos unterhalb von 300 UpM bleibt, bis die tatsächliche Messrampe beginnt.)
- Ergebnisabfragen (`Result`, `Report`), Informationsabfragen oder das Setzen von Informationen (z.B. Seriennummer setzen) haben Antwortzeiten von unter 500 ms und werden nur wenig durch externe Faktoren beeinflusst. Daher ist hier das Timing relativ stabil und sicher.

Wie bereits oben erwähnt ist die beste Strategie, zeitkritische Kommandos so früh wie möglich zu schicken, um mehr Puffer für einen Timeout zu haben.

Überprüfen der Zeiten, Logbuch-Datei

Wie auf der vorigen Seite dargestellt, kann die Kommunikation im Ausgabefenster (Reiter **Kommunikation**) überwacht werden. Durch einen Rechts-Klick im Ausgabefenster rufen Sie ein Kontextmenü auf. Aktivieren Sie in diesem Menü den Befehl **Zeitstempel hinzufügen** (**Add time stamps**), um im Kommunikationsprotokoll Zeitmarken zu erhalten, anhand derer das Timing überprüft werden kann.

Der Inhalt des Ausgabefensters wird zusätzlich in einer Logbuch-Datei gespeichert. Diese Datei befindet sich im Ordner

`C:\Discom\Measurement\MulitRot\(`*`ProjectName`*`)\Locals\Logs\`

Wenn die Logbuch-Datei eine bestimmte Größe überschreitet, wird sie umbenannt, indem ein Datums-Text dem Namen hinzugefügt wird, und dann ein neues Logbuch begonnen. Daher ist die aktuell verwendete Datei diejenige ohne eine Datums-Zeitangabe.

Wenn Sie Probleme mit der Kommunikation haben (nicht nur Timing, sondern beispielsweise auch unerwartete Antworten), senden Sie uns bitte zusammen mit Ihrem Fehlerbericht die Logbuch-Datei zu, oder gleich den ganzen Logbuch-Ordner (komprimiert, natürlich).

Kommandovorrat des TasAlysers

Die folgenden Tabellen listen die Kommandos auf, die der TasAlyser in der Standardkonfiguration interpretieren und ausführen kann, und die Antworten gemäß dem „Handshake-Standard“-Protokoll. Sofern sich diese von denen des „Basis-Standard“-Protokolls unterscheiden, werden auch letztere aufgeführt. Weitere Kommandos werden durch Plugins realisiert, die in folgenden Abschnitten beschrieben sind.

Wird ein Kommando empfangen, das nicht interpretiert werden kann (auch nicht durch ein Plugin – s.u.), antwortet der TasAlyser mit „?“.

Einige Befehle bedürfen genauerer Erläuterungen. Diese sind in der Tabelle mit (*) markiert und werden im Anschluss an die Tabelle diskutiert.

Prüfablauf

Kommando	Argument	Funktion	Antwort
`Insert:` (*)	Name eines Typs bzw. einer Prüfvorschrift *Optional:* danach getrennt durch ‚|' eine Seriennummer. Beispiel: `Insert: ABC|12345`	Lädt die Daten für diesen Typ und aktiviert die betreffende Prüfvorschrift.	`Inserted` bei Erfolg, `Failed` bei Misserfolg. Mögliche Gründe: Typ/Prüfvorschrift unbekannt, oder voriger Prüflauf noch nicht beendet (`Remove` fehlt). *Basis-Standard*: `1` bei Erfolg, `0` bei Misserfolg.
`Serial:`	Seriennummer	Setzt oder ändert die Seriennummern-Information	1
`Timestamp:`	Zeitstempel im Format *yyyy mm dd hh mm ss* (*yy* zweistellig wird als 20*yy* interpretiert)	Setzt den Zeitstempel für die aktuelle Prüfung. (Der Standard-Zeitstempel ist der Zeitpunkt des `Insert`s.)	1 wenn Format korrekt 0 bei Format-Fehler
`EndOfTest:` (*)	_	Ende aller Prüfschritte; es folgen keine Prüfungen mehr. Ende der Wave-Aufnahme.	1 bei Erfolg, 0 bei Misserfolg.
`Remove:`	_	Beendet den Prüflauf regulär und endgültig	`Done-`*x* bei Erfolg, wobei ‚*x*' der Ergebnis-Code wie beim `Result`-Kommando ist; `Failed` bei Misserfolg (typischerweise, weil kein `Insert` durchgeführt wurde). *Basis-Standard*: 1 bei Erfolg, 0 bei Misserfolg
`Reset:`	_	Bricht den Prüflauf ab und bring das System in den Grundzustand	`Reset OK` *Basis-Standard*: 1
`Mode:` (*)	Name des Prüfzustands oder `$Nil`	Aktiviert einen Prüfzustand bzw. beendet den aktuellen	`OK` bei Erfolg, `Error` bei Misserfolg *Basis-Standard*: 1 bei Erfolg, 0 bei Misserfolg
`Measure:`	‚1' oder ‚`On`' ‚0' oder ‚`Off`' ‚x' oder ‚`Cancel`'	Startet bzw. beendet die Messwert-Erfassung	`On/Off/Cancel` bei Erfolg, `Error` bei Misserfolg *Basis-Standard*: 1 bei Erfolg, 0 bei Misserfolg

Prüfparameter ändern

Kommando	Argument	Funktion	Antwort
`TestProcedure:` `(*)`	Name eines Prüfverfahrens	Optional: setzt das zu verwendende Prüfverfahren	1 bei Erfolg, 0 bei Misserfolg.
`TestStandName:` `(*)`	Name für den Prüfstand	Optional: setzt für die folgende Prüfung den Namen des Prüfstands	1 bei Erfolg, 0 bei Misserfolg.

`TestKind:`
`SetTestKind:`
`(*)`
	n (Numerischer Code für die Art der Messung)	Setzt die „Art der Messung“ (z.B. auf „Referenzmessung“ oder „Test“)	1 bei Erfolg,
			0 bei Misserfolg.

`SetTestProperty:` `(*)`	Einer oder mehrere Eigenschafts-Marker	Setzt eine „Eigenschaft der Prüfung“	1 bei Erfolg 0 bei Misserfolg

Ergebnisse abrufen

Kommando	Argument	Funktion	Antwort
`Result:` `(*)`	–	Liefert die (bisherige) Gesamtbewertung	`Result` *x* Werte für Ergebnis-Code *x*: 1 = keine Fehler gefunden 0 = Fehler gefunden 2 = keine Messung/Bewertung 3 = Systemfehler aufgetreten *Basis-Standard*: nur der Ergebniscode `x`.
`Result:` `(*)`	Prüfzustands-Name	Abfrage des (bisherigen) Bewertungsergebnisses für den angegebenen Prüfzustand	Wie beim generellen `Result`, aber bezogen nur auf den angegebenen Prüfzustand.
`ClearResult:`	keines oder Name eines Prüfzustands	Löschen aller Fehlerberichte bzw. aller aus dem benannten Prüfzustand	1 (auch wenn Prüfzustand unbekannt)

Fehlerberichte abfragen

Kommando	Argument	Funktion	Rückgabewert
`Report:`	`Text`	Liefert den Messbericht (Gesamtergebnis + Fehlermeldungen)	Der Text des Messberichts, zeilenweise
`Report:`	`Count`	Abfrage der Anzahl der Fehlermeldungen	Anzahl der Fehlermeldungen (0, wenn keine Fehler vorhanden)
`Report:` `(*)`	`TextLine` *n*	Abfrage Fehlerbeschreibung Zeile *n*	Fehlertext Zeile *n* (ab 1), wenn vorhanden, sonst „-“
`Report:` `(*)`	`Codes`	Abfrage der Fehlercodes	Zeilenweise je ein Fehlercode. Als letztes wird immer zusätzlich „0“ ausgegeben, um das Ende der Liste zu markieren.
`Report:` `(*)`	`CodesLine` `CodesLine` *n*	Abfrage der ersten 10 Fehlercodes auf einer Zeile Optional: Anzahl der Ziffern pro Fehlercode (Standard = 4)	In einer Zeile die ersten 10 Fehlercodes, jeweils vierstellig, mit „0“ aufgefüllt. Beispiel: 01230133900300000000.... enthält die Fehlercodes 123, 133 und 9003.

`Report:` `(*)`	`CodeNr` *n*	Abfrage von Fehlercode Nummer *n*	Der *n*-te Fehlercode, gezählt ab 1. Ist *n* größer als die Anzahl an Fehlern, wird „0“ zurückgeliefert.
`ReportDigest:` `(*)`	`Formatanweisung` optional: `Zeilennr`	Liefert einen frei formatierten Fehlerbericht	Die Fehlerberichte, zeilenweise, dahinter die Zeile <end>
`ReportCodesMode:` `(*)`	*`Prüfzustandsname`*	Abfrage der Fehlercodes für einen Prüfzustand	Wie ‘`Report: Codes`‘
`Severity:` `(*)`	_ optional: Prüfzustand	Abfrage der „Schwere des schlimmsten Fehlers“	0 = keine Fehler gefunden, sonst wie in Param.-Db festgelegt
`SeverityText:` `(*)`	_ optional: Prüfzustand	Abfrage eines Textes zur aktuellen Severity	wie in Param.-Db festgelegt
`InstrResult:` `(*)`	Instrumenten-Bezeichner, dann optional Name eines Prüfzustands	Abfrage der Bewertungsergebnisse für dieses Instrument (in diesem Prüfzustand).	1 = keine Fehler gefunden 0 = Fehler gefunden

Statusabfrage, Sonderbefehle

Kommando	Argument	Funktion	Rückgabewert
`Status:`	_	Fragt den Zustand des Messprogramms ab	0 = System nicht bereit 1 = bereit für „`Insert`“ 2 = „Insert“ bereits erfolgt
`Ping:`	(optional: beliebiger Text)	Test der Kommunikation; ggf. Initialisieren des Puffers	Der als Argument übergebene Text, oder „`OK`“
`PauseWaveRec:`	1 / 0	1 = Aufnahme pausieren 0 = Aufnahme fortsetzen	1 in fast allen Fällen, 0, wenn kein Prüflauf gestartet ist.
`SetComment:` `(*)`	Text	Setzt einen Kommentar zur Messung, der mit den Messdaten archiviert wird	1
`SetInfo:` `(*)`	Name Wert	Setzt eine zu archivierende Zusatzinformation „Name“ auf den Wert „Wert“	0 bei Syntaxfehler, 1 bei Erfolg
`SetComponentInfo:` `(*)`	Komponente Eigenschaft Wert	Setzt die Eigenschaft einer Komponente; siehe Anmerkungen	0 bei Syntaxfehler, 1 bei Erfolg
`Message:`	Text der Meldung oder „`x`“	Zeigt ein Fenster mit dem Meldungstext an. „x“ schließt das Fenster	1
`SetExtError:`	Fehlercode, optional Messwerte	Fügt den Akustikfehlern einen Prüfstandsfehler hinzu	1 bei Erfolg, 0 im Fehlerfall
`OperatorRemove:`	(siehe Beschreibung unten)	Öffnet ein Fenster zur manuellen Quittierung der Prüfung und Eingabe von Zusatzinformationen	1 bei Erfolg 0 Kommando konnte nicht angenommen werden (z.B. kein Prüflauf aktiv)

Anmerkungen zu einigen Befehlen

Reset: Dieses Kommando bricht eine eventuell laufende Prüfung ab. Es werden keine Ergebnisse gespeichert und die Wave-Aufzeichnung gelöscht. Das System kehrt in den Grundzustand zurück.

Insert: Dieses Kommando startet einen neuen Prüflauf. Der Typ-Name, der als Argument verwendet wird, muss in der Parameterdatenbank bekannt sein. (Mit den speziellen Typ-Namen „`$Repeat`” und

„`$Again`" wird ein neuer Prüflauf mit demselben Typ wie im vorausgehenden Prüflauf gestartet, was etwas weniger Zeit verbraucht, da die Parameter nicht neu geladen werden). Das Insert-Kommando wird mit `Inserted` quittiert, wenn alle Parameter erfolgreich geladen wurden, und mit `Failed` im Fehlerfall. (Im Basis-Standard-Protokoll sind die Antworten die Ziffern 1 bzw. 0.) Die Prüfstandssteuerung muss unbedingt den Erfolg des Insert-Kommandos prüfen und entsprechend reagieren!

EndOfTest: Dieses Kommando ist optional (aber empfohlen). Es gibt bekannt, dass keine weiteren Prüfzustände folgen werden. Falls die Aufzeichnung von Wave-Dateien aktiviert ist, wird sie mit diesem Kommando beendet. Nachdem das `EndOfTest`-Kommando ausgeführt wurde, liegen sämtliche Ergebnisse bereit und können abgefragt werden; es ist aber noch möglich, weitere Informationen hinzuzufügen (z.B. die Seriennummer zu ändern oder externe Fehler zu übermitteln). Wird `EndOfTest` nicht verwendet, impliziert das Remove-Kommando ein `EndOfTest`.

Mode: dieses Kommando liefert `Error` (Basis-Standard: 0) zurück, wenn kein Prüflauf gestartet ist (kein `Insert:`) oder wenn der angegebene Prüfzustandsname unbekannt ist. Der Prüfzustandsname „`$Nil`" legt *keinen* Prüfzustand ein, d.h. der aktuelle Prüfzustand wird beendet, ohne dass ein neuer gesetzt wird. Die Wave-Aufzeichnung wird pausiert, bis ein neuer Prüfzustand eingelegt wird.

Measure: Um eine Messung zu starten, muss ein Prüflauf im Gange sein (`Insert:`) und ein Prüfzustand angewählt sein (`Mode:`). Vielfach ist das Messsystem so eingerichtet, dass die Messung automatisch anhand von Drehzahlgrenzen gesteuert wird, so dass der `Measure`-Befehl nicht benötigt wird.

TestProcedure und **TestStandName:** Diese Kommandos sind optional. Für denselben Typ können in der Parameterdatenbank verschiedene Prüfverfahren oder Prüfstandsnamen definiert werden, die mit diesen Befehlen ausgewählt werden. (Dadurch ist es z.B. möglich, am selben Prüfstand wahlweise eine kurze oder eine ausführliche Prüfung durchzuführen.) Die Befehle müssen vor dem `Insert:`-Befehl geschickt werden, und gelten nur für die nächste Prüfung. Die Bezeichnungen der Prüfverfahren bzw. Prüfstände sind mit der Parameterdatenbank abzugleichen.

Im Messprogramm ist der Standard-Prüfstandsname fest hinterlegt. Gibt es nur ein Prüfverfahren, ist die Verwendung dieser Befehle nicht erforderlich.

TestKind bzw. **SetTestKind**: Beide Schreibweisen dieses Befehls haben dieselbe Funktion: sie ändern die „Art der Messung", die als Eigenschaft der Gesamtmessung mit den Ergebnissen abgespeichert wird. Das Argument ist der numerische Code für die „Art der Messung". Zulässige Werte sind: 1 = Serienmessung, 2 = Referenzmessung, 3 = „Spezialmessung", 4 = „Testmessung".

Die über diesen Befehl gesetzte „Art der Messung" gilt nur für den aktuellen Prüflauf und wird für den nächsten Lauf automatisch wieder auf „Serienmessung" zurückgesetzt. Der Befehl kann vor dem `Insert:`-Kommando gesendet werden und muss vor dem `Remove:` kommen.

SetTestProperty: Zusätzlich zur „Art der Messung" (s.o.) kann eine Prüfung auch eine oder mehrere Eigenschaften haben, z.B. „Ist eine Kunden-Rücklieferung". Jede Eigenschaft wird durch einen Buchstaben codiert. Die verfügbaren Eigenschaften sind „R" = Reparatur und „D" = Kundenrücklieferung. Das Kommando `SetTestProperty: R` setzt also z.B. die „Reparatur"-Eigenschaft. Durch ein Minuszeichen vor den Eigenschafts-Buchstaben (z.B. „`-R`") kann eine Eigenschaft zurückgenommen werden. Es können mehrere Eigenschaften mit einem Befehl gesetzt werden.

Result: Eine Prüfung hat vier mögliche Ergebnisse: nicht in Ordnung, in Ordnung, ohne Bewertung oder Systemfehler[1]. Wenn erforderlich, können Ergebnisse 2 (ohne Bewertung) und 3 (Systemfehler) als 1 (i.O.) bzw. 0 (n.i.O.) gemeldet werden. Weitere Details siehe Seite 17.
Auch wenn während des Prüflaufs die Ergebnisse jedes einzelnen Prüfzustands per „`Result: PrfzstdName`“ abgefragt wurde, *muss* am Ende des Prüflaufs ein allgemeines „`Result:`“ abgerufen werden, da es Fehler gibt (z.B. aus der Sensorüberwachung), die keinem normalen Prüfzustand zugeordnet sind!

Severity: In der Parameterdatenbank sind die Fehler in Gruppen unterschiedlicher Schwere eingeteilt. Mit diesem Befehl kann die Gruppe mit den „schlimmsten Fehler“ ermittelt werden. Rückgabewert ist die Gruppen-Nummer. Mit **SeverityText** kann ein in der Datenbank für die fragliche Gruppe hinterlegter Text abgerufen werden.

InstrResult: Mit diesem Kommando kann das Bewertungsergebnissen nach dem „Instrument“ gefiltert werden, um beispielsweise nur die Übersetzungsprüfung abfragen zu können. Als erstes Argument muss eine Instrumenten-Bezeichnung angegeben werden. Die existierenden Instrumente sind projektspezifisch in der Parameterdatenbank festgelegt. Als zweites, optionales Argument kann ein Prüfzustandsname folgen.

Report: Codes / **CodesLine** / **CodeNr**: Die Fehlercodes werden nach Fehler-Prioritäten sortiert ausgegeben. Doppelte Codes werden dabei unterdrückt, d.h. auch wenn der Fehler 177 fünf Mal aufgetreten ist, wird nur ein Fehlercode 177 gemeldet. Die Nummerierung der Fehler beginnt bei 1.

Report: TextLine: es gilt dasselbe wie oben für `Report: CodeNr`. Die Antwort-Zeile besteht aus dem Fehlertext (aus der Parameterdatenbank) und der Spezifikation (Prüfzustand, Sensor usw.) Die maximale Zeilenlänge beträgt 120 Zeichen. Ist *n* größer als die Anzahl der Fehler, wird mit „–“ geantwortet.

ReportDigest: Mit diesem Befehl können die Elemente des Fehlerberichts (Fehlercodes, Text, Prüfzustand usw.) einzeln oder in beliebiger Zusammenstellung abgefragt werden. Das Format des Befehls ist

`ReportDigest: Formatanweisung`

Die Formatanweisung besteht aus Buchstaben, mit denen die einzelnen Elemente des Berichts angeordnet werden. Diese sind:

C	E	T	M	S	V	P	D	N
Fehlercode	„externer“ Fehlercode	Fehlertext	Prüfzustand	Spezifi-kation	Wert und Grenze	Position	Differenz Wert – Grenze	lfd. Nummer

Beispiel: das Kommando

```
ReportDigest: CMT
```

liefert den Fehlercode, den Prüfzustand und den Fehlertext, also z.B.

```
583 3-D Ordnung laut
```

Die Elemente werden mit Leerzeichen getrennt. Als erstes Zeichen der Formatanweisung kann jedoch ein nicht-alphanumerisches Zeichen als Trennzeichen angegeben werden, z.B.

```
ReportDigest: |TMS
Ordnung laut|3-D|Spektrum Zwischenwelle Sync
```

Der `ReportDigest`-Befehl liefert so viele Zeilen, wie Fehlerberichte vorhanden sind, und zum Abschluss die Zeile

```
<end>
```

[1] „Ohne Bewertung“ ist das Ergebnis, wenn keine Prüfung durchgeführt wurde – zum Beispiel direkt am Anfang eines Prüflaufs. „Systemfehler“ zeigt ein Problem an, das die Durchführung einer Prüfung unmöglich macht, z.B. einen Sensordefekt.

Sind gar keine Fehlerberichte vorhanden, wird nur `<end>` geliefert.

Optional kann als zweites Argument des Befehls eine Zeilennummer angegeben werden. Die Zeilennummer wird ab 1 gezählt. Ist sie vorhanden, wird nur diese Zeile des Berichts zurückgeliefert. Ist die Nummer größer als die Anzahl der Berichte (vergl. das Kommando `Report: Count`), ist die Antwort `<end>`.

Ping: Dieses Kommando wird direkt vom Decoder beantwortet, ohne in die eigentliche Steuerung des Messsystems einzugreifen. Die Antwort ist der als Argument übergebene Text. Wurde kein Argument übergeben, ist die Antwort „`OK`“.

Das Ping-Kommando kann verwendet werden, um zu jedem Zeitpunkt den Status der Kommunikation zu testen und ggf. um einen Rückgabewert auf der Schnittstelle zu erhalten, der sich von den Ziffern „0“ und „1“ unterscheidet.

Achtung: das `Ping`-Kommando ist absichtlich nicht dagegen abgesichert, sich mit anderen Kommandos zu überschneiden. Wenn also beispielsweise die Kommandos „`Insert`:“ und „`Ping`:“ direkt hintereinander gesendet werden, dann kann es passieren, dass man zuerst die Antwort auf das `Ping` erhält, und danach erst die Quittierung des `Insert`.

PauseWaveRec: In der üblichen Konfiguration werden alle Sensordaten in einer Wave-Datei aufgezeichnet, beginnend mit dem ersten `Mode`-Kommando und endend mit `EndOfTest`. Manchmal enthalten Prüfläufe in der Mitte Abschnitte von signifikanter Dauer, die für die akustische Analyse uninteressant sind. Am Beginn eines solchen Abschnitts kann mit `PauseWaveRec: 1` die Wave-Aufzeichnung pausiert werden, um die Größe der Wave-Dateien zu reduzieren. Die Aufzeichnung wird fortgesetzt mit dem Kommando `PauseWaveRec: 0` oder mit dem nächsten `Mode`-Kommando.

SetComment, SetInfo: Mit diesen Befehlen können Zusatzinformationen im Messdatenarchiv abgelegt werden. SetComment setzt den Mess-Kommentar, SetInfo die frei definierbaren Zusatzinformationen. Beispiel: `SetInfo: Achstyp Abc123` legt im Archiv für die Information „Achstyp“ den Wert „Abc123“ ab. SetInfo- und SetComment-Kommandos müssen vor dem `Remove:` erfolgen. Der `Reset:`-Befehl setzt auch alle Zusatzinfos zurück.

SetComponentInfo: Dieses Kommando setzt die Information bezüglich einer Eigenschaft eines Bauteils des Testobjekts, beispielsweise die Seriennummer eines Zahnrades in einem Getriebe. Das Kommando hat drei Argumente: den Komponenten-Namen (z.B. „PrimGear“), den Namen der Eigenschaft (z.B. „GearSerial“) und den Wert (die Seriennummer). Bezüglich des Ablaufs verhält sich `SetComponentInfo` genauso wie `SetInfo`.

SetExtError: Dieser Befehl wurde früher durch ein Plug-In behandelt, ist jetzt jedoch Teil des Standard-Kommandoumfangs. (Das Plugin gibt es weiterhin, mit zusätzlichen Funktionen.) Weitere Einzelheiten zum `SetExtError`-Befehl finden Sie in der Plugin-Beschreibung ab Seite 20.

OperatorRemove: Dieses Kommando bewirkt, dass ein Fenster geöffnet wird, in dem der Bediener den Prüflauf quittieren muss. Der Bediener hat die Optionen „OK“ (Prüflauf regulär beenden, entspricht `Remove`), „Abbruch“ (Prüflauf verwerfen, entspricht `Reset`) und „Zurück“ (Prüflauf noch nicht beenden, sondern zurück in die Durchführung von Prüfschritten). Außerdem kann der Bediener in diesem Fenster weitere Informationen eingeben (z.B. einen Kommentar), die mit den Messergebnissen gespeichert werden.

Als Argument kann dem Kommando eine Zahl mitgegeben werden, die durch bit-weises Oder der untenstehenden Optionen gebildet wird. (Beispiel: die Optionen 1 und 4 werden zu 5 kombiniert, so dass das Kommando `OperatorRemove: 5` lautet.)

Options-Wert	Funktion
0	Das Fenster wird geöffnet und wartet auf den Bediener, aber das Kommando `OperatorRemove` wird sofort beantwortet
1	Die Option „Zurück“ (Rückkehr zur Prüfung) wird nicht angeboten
2	Der Bediener kann im Fenster die Seriennummer nicht ändern
4	Das `OperatorRemove`-Kommando wird erst beantwortet, wenn der Bediener das Fenster durch Auswahl einer Option geschlossen hat (d.h. den Prüflauf beendet hat)

Wird mit dem Kommando gar kein Argument geschickt, gilt die Option 0.

Solange das Fenster geöffnet ist (d.h. solange der Bediener keine Auswahl getroffen hat), akzeptiert das Messsystem keine weiteren Ablauf-Kommandos vom Prüfstand (kein `Reset`, `Remove`, `Mode` oder neues `Insert`).

Unterschiede Handshake-Standard und Basis-Standard

Die folgende Tabelle fasst die unterschiedlichen Quittierungen von Kommandos für das Handshake-Standard gegenüber dem Basis-Standard-Protokoll zusammen:

Kommando	Basis-Std.	Handshake-Standard
`Reset:`	`1`	`Reset OK`
`Insert:` *`ABCD`*	`1 / 0`	`Inserted / Failed`
`Mode:` *`1-A`*	`1 / 0`	`OK / Error`
`Measure: On/Off/Cancel`	`1 / 0`	`On / Off / Cancel / Error`
`Remove:`	`1 / 0`	`Done-`*`x`*` / Failed` *(x = Result Code)*
`Result:`	*x*	`Result` *x* *(x = Result Code)*

Beispiele

Dieses Beispiel zeigt einen einfachen Prüfablauf, jeweils mit Kommandos und Antworten. (Auf der nächsten Seite folgt ein komplexerer Beispiel-Ablauf mit mehr Kommandos):

Prüfstand	**TAS**	**Beschreibung**
`Reset:`	`Reset OK`	Rücksetzen: Abbruch eines eventuell nicht zuende geführten vorigen Prüflaufs
`Status:`	`1`	Muss „1“ sein, damit ein Prüflauf begonnen werden kann
`Insert: A17`		Typ „A17“ soll geprüft werden.
	`Inserted`	Parameter für „A17“ wurden erfolgreich geladen
`Serial: 4711`	`1`	Bekanntgabe der Seriennummer
`Mode: Up`		Nächster Prüfzustand: „Up“
	`OK`	Bestätigung: bereit für „Up“
(hier Fahren einer Drehzahlrampe)		
`Result: Up`		Abfrage des Prüfergebnisses für „Up“
	`Result 1/0`	Ergebnis der Prüfung für „Up”: 1 = keine Fehler, 0 = Fehler
`Mode: Down`		Nächster Prüfzustand ist „Down“
	`OK`	Bestätigung: bereit für „Down“
(nächste Drehzahlrampe)		
Weitere Prüfzustände mit `Mode:` *aufrufen und fahren, dann:*		
`EndOfTest:`	`1`	Alle Prüfschritte beendet; Endergebnis wird bestimmt
`Result:`		
	`Result 1/0`	Gesamtergebnis des Prüflaufs kann nach `EndOfTest` erfragt werden
`Remove:`	`Done-1`	Prüfung beendet, Prüfergebnis OK (in diesem Beispiel)
`Result:`	`Result 1/0`	Gesamtergebnis kann auch jetzt abgefragt werden
`Report: …`	…	Jetzt Abfrage von Reports möglich
`Reset:`	`Reset OK`	Vorbereiten auf nächsten Prüflauf
`Insert: A17`		Nächster Prüfzyklus beginnt – weiter wie oben.

Erläuterungen

- Das System merkt sich alle Messergebnisse, Fehlermeldungen etc., bis mit `Insert` ein neuer Prüfzyklus gestartet wird. Auch noch nach `Remove:` können daher alle Arten von Reports, auch mehrfach, abgefragt werden.
- Das Kommando `Serial:` kann zu jeder Zeit vor `Remove:` kommen. Es kann wiederholt werden, um die Seriennummer zu ändern. Die Seriennummer kann eine beliebige Zeichenkette ohne Leerzeichen sein, ohne ernsthafte Längenbeschränkung.
- In der Parameterdatenbank sind die möglichen Typenbezeichnungen hinterlegt. Alle anderen Typen werden bei `Insert` abgewiesen.
- Ebenso sind in der Parameterdatenbank alle möglichen Prüfzustände (Modes) hinterlegt. Innerhalb eines Prüflaufs müssen nicht alle Prüfzustände gemessen werden.
- Die Reihenfolge der Messung der Prüfzustände ist beliebig. Ebenso können Prüfzustände wiederholt gemessen werden. Durch das erneute Kommando `Mode: X` werden dabei alle eventuellen Fehler, die in einer früheren Messung des Prüfzustandes X aufgetreten sind, gelöscht.
- Das Messprogramm ist meistens so konfiguriert, dass es selbsttätig das Durchfahren von Drehzahlrampen überwacht und bei entsprechenden Drehzahlgrenzen die Messung beginnt und beendet. (Zur Verwendung des `Measure: 1/0` –Kommandos siehe das zweite Beispiel.)

- Nach dem `EndOfTest`-Kommando steht das Endergebnis fest. `Result:` und `Report:`-Kommandos können ab `EndOfTest:` (auch schon vor `Remove:`) verwendet werden. (Siehe Erläuterung der Befehle). `SetExtError:`-Befehle sind ebenfalls weiterhin möglich.
- Auch wenn während des Prüflaufs die Ergebnisse jedes einzelnen Prüfzustands per (im Beispiel) „`Result: Up`“ abgefragt wurde, muss am Ende des Prüflaufs ein allgemeines „`Result:`“ abgerufen werden, da es Fehler gibt (z.B. aus dem System-Selbsttest), die keinem normalen Prüfzustand zugeordnet sind!
- Die Antwortzeiten auf alle Kommandos außer `Insert:` und `Remove:` liegen deutlich unter 1 Sekunde. Das `Insert`-Kommando kann länger dauern (bis 10 Sekunden), wenn Änderungen in der Parameter-Datenbank vorgenommen wurden oder ein bisher noch nicht geprüfter Typ angefordert wird; ansonsten dauert auch das `Insert`-Kommando nicht länger als 1–2 Sekunden. Die Dauer des `Remove`-Kommandos hängt von den daraufhin erforderlichen Datenbank-Zugriffen (Schreiben des Messdatenarchivs etc.) ab, sollte aber 10 Sekunden nicht überschreiten. Lesen Sie auch den Abschnitt „Dauer der Ausführung von Kommandos, Timing“ auf Seite 4.

Zwischen Kommandos und Argumenten und auch dahinter dürfen beliebig Leerzeichen auftreten. Zwingend erforderlich ist allerdings die Terminierung jeder Befehlszeile mit `\r\n`.

Komplexeres Beispiel

Das folgende Beispiel zeigt einen Ablauf mit zusätzlichen optionalen Kommandos, die z.T. durch Erweiterungen (Plug-Ins, siehe ab Seite 19) ausgeführt werden. Die Antworten des TAS-Systems sind nicht aufgeführt; siehe dazu und zu den Einzelheiten der Kommandos die Beschreibungen im vorigen Kapitel.

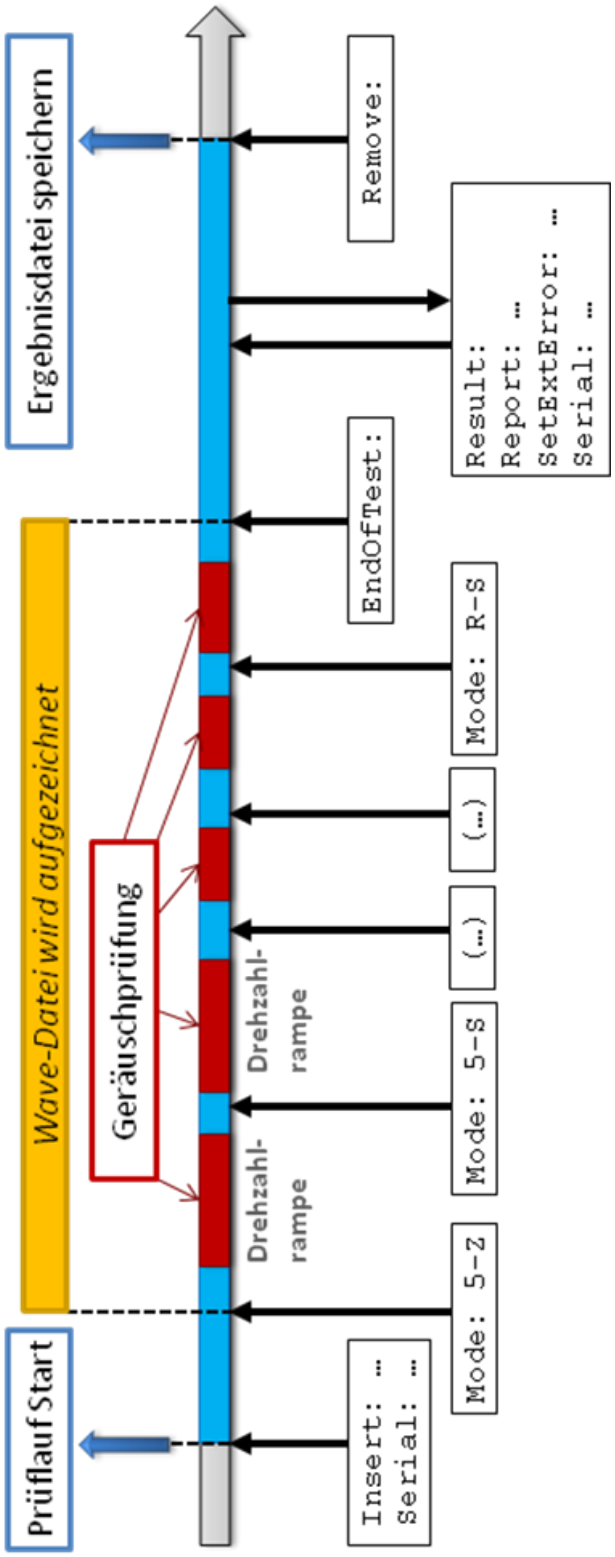
`Reset:`	
`SetInfo: BoxType 5A`	Setzen einer Zusatzinformation für die Ergebnis-Datei
`Serial: 1234567`	
`TestProcedure: SpecialTest`	Statt des normalen Prüflaufs folgt ein Spezial-Prüflauf
`Insert: PQR`	
`Mode: 1-Z`	
(Drehzahlrampe)	
`Mode: TqR`	Prüfzustand „Torque Ramp“
(Drehmoment-Rampe)	
(weitere Prüfzustände mit Mode:-Kommandos aufrufen und durchfahren)	
`SGW: 4 5`	Beginn eines Gangschalt-Vorgangs von 4 nach 5*
`Mode: 5-S`	
`EGW:`	Ende des Gangwechsels*
(Drehzahlrampe)	
`Result: 5`	Abfrage des Ergebnisses nur für den 5. Gang (5-S und 5-Z)
`Mode: $Nil`	Durch dieses Kommando wird die Wave-Aufzeichnung auf Pause geschaltet (bis zum nächsten `Mode`-Kommando). Dies ist sinnvoll, wenn während eines Prüflaufs längere Abschnitte ohne Geräuschprüfung vorkommen.
(andere Prüfungen ohne Akustik)	
`Mode: Steady`	Dies sei im Beispiel ein Prüfzustand ohne Drehzahlrampe.
`Measure: 1`	Daher muss die Messung durch den Prüfstand angestoßen…
(zeitgesteuerte Prüfung)	
`Measure: 0`	…und beendet werden.
`EndOfTest:`	
`GetValueByName: StdRMS`	Wert der Größe „StdRMS“ abfragen*

`SetExtError 567 33.8 25`	Prüfstandsfehler Nr. 567 zu den Akustikfehlern hinzufügen
`Result:`	Immer das Gesamtergebnis abfragen!
`Serial: 987654BX856432A`	Ändern der Seriennummer
`Remove:`	

* Diese Kommandos werden durch Plugins behandelt; siehe nächstes Kapitel ab Seite 19.

Zeitlicher Ablauf

Die folgende Grafik verdeutlicht den zeitlichen Ablauf einer Prüfung:



Das System zeichnet alle Sensordaten in einer Wave-Datei auf, beginnend mit dem ersten `Mode`-Kommando und endend mit `EndOfTest`.

Das Kommando `SetExtError` ist eine Erweiterung, die auf Seite 19 beschrieben wird. Damit kann der Prüfstand dem Messsystem Fehlermeldungen übermitteln, die zusammen mit dem Messergebnissen abgespeichert werden.

Über die Ergebnis-Codes (Result)

Beim Test eines Prüflings in der Produktion gibt es nur zwei Möglichkeiten: das Teil kann verkauft werden (Ergebnis i.O.) oder nicht (Ergebnis n.i.O.). Die weitere Behandlung von n.i.O.-Teilen hängt von den gefundenen Fehlern oder deren Schwere ab, die mit anderen Kommandos erfragt werden kann.

Genauer betrachtet gibt es noch zwei weitere mögliche Ergebnisse eines Prüflaufs: es hat gar keine Prüfung stattgefunden, oder es ist ein Problem aufgetreten, das die Prüfung verhinderte.

Entsprechend diesen vier Möglichkeiten gibt es vier mögliche Antworten auf das `Result:`-Kommando:

Code	Bedeutung	Erläuterung
`1`	Test i.O.	Keine Fehler gefunden (und auch nicht per `SetExtError` gesetzt). Das geprüfte Teil kann verkauft werden.
`0`	Test n.i.O.	Fehler gefunden. Das geprüfte Teil sollte nicht direkt verkauft werden, sondern repariert oder recycelt werden.
`2`	ohne Bewertung	Es wurde noch kein Test durchgeführt. Dies ist das Ergebnis zu Beginn eines Prüflaufs. Wenn das System am Ende des Prüflaufs mit 2 antwortet, prüfen Sie, ob Prüfschritte (`Mode:`) durchgeführt wurden.
`3`	Systemfehler	Es ist ein Problem aufgetreten, das eine normale Prüfung verhindert. Es kann nicht entschieden werden, ob das geprüfte Teil i.O. oder n.i.O. ist. Typische Beispiele für Systemfehler sind fehlende Drehzahlen oder defekte Sensoren.

Ein Prüfstand muss alle vier Ergebnis-Codes korrekt behandeln. (Die Reaktion auf einen Code 3 besteht typischerweise in einem Stopp und dem Ruf nach einem Bediener.)

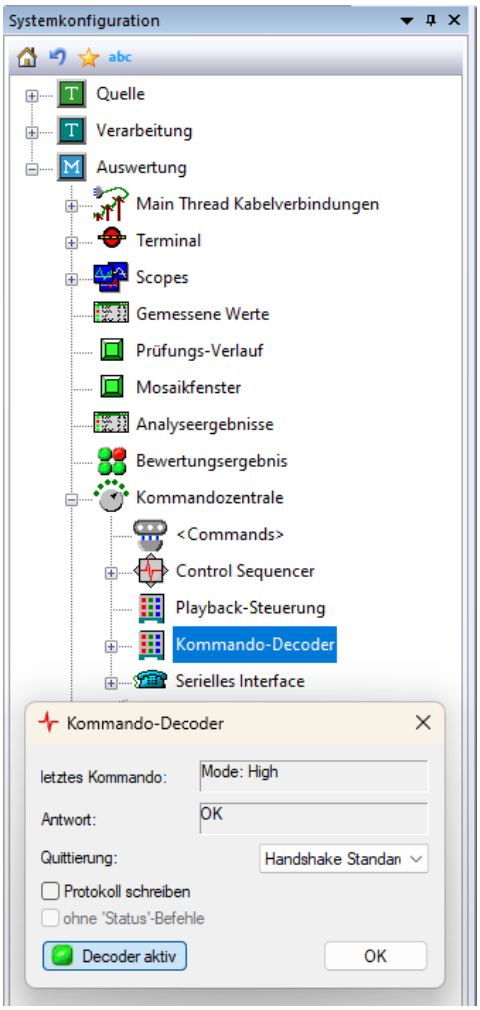
Obwohl man in der TasAlyser Messapplikation den Ergebniscode 2 auf 1 und Code 3 auf 0 umleiten kann, ist diese Option für alte Prüfstände reserviert, die nur simple Abläufe leisten können, und soll nicht für neue Prüfstände verwendet werden.

Es ist möglich, nach jedem Prüfschritt das Ergebnis (`Result`) für diesen Schritt abzufragen, z.B. um einen Prüflauf vorzeitig zu beenden, wenn bereits ein früher Schritt das Ergebnis n.i.O. liefert. Es ist dennoch absolut unverzichtbar, dass die Prüfstandssoftware am Ende der Prüfung (nach `EndOfTest`) das generelle Ergebnis abfragt (`Result:` ohne Argument). Manche Systemfehler werden keinem normalen Prüfschritt zugeordnet und würden daher ohne die generelle Result-Abfrage vom Prüfstand unbemerkt bleiben und zu fehlerhaft verkauften Teilen führen.

Einstellungen und Test-Modus

Um an die Einstellungen des Decoders zu gelangen, klappen Sie das **Systemkonfigurations**-Fenster aus. Öffnen Sie den Knoten **Auswertung** und darin den Knoten **Kommandozentrale**. Doppelklicken Sie auf das Element **Kommando-Decoder**, um dessen Einstell-Fenster zu öffnen:

(Auf englischen Systemen heißt das Element **Command Decoder** und befindet sich innerhalb des Knotens **Command Center**.)



Um die Applikation testen zu können, zeigt der Tas-Decoder in seinem Fenster das letzte eingetroffene Kommando und das Ergebnis der Ausführung.

Wenn Sie im Dialog das Kontrollkästchen **Protokoll schreiben** (**write minutes** auf englischen Systemen) aktivieren, werden im Ausgabefenster alle empfangenen Kommandos und die erfolgten Antworten protokolliert.

Die Interface-Module (Serielles Interface usw.) bieten ebenfalls die Option, den gesamten ein- und ausgehenden Verkehr im Ausgabefenster mitzuschreiben. Die Ausgabe des Tas-Decoders ist allerdings kompakter gehalten, daher sollten Sie diese wählen, wenn Sie nur ein Protokoll des Ablaufs und nicht der exakten Kommunikation wünschen.

Wenn Sie zusätzlich die Option **Ohne ‚Status'-Befehle** einschalten, werden im Protokoll die `Status`-Kommandos weggelassen. Dies ist sinnvoll, wenn die Prüfstandssteuerung regelmäßig (z.B. jede Minute) den Status des Messsystems abfragt, um zu vermeiden, dass die protokollierten Status-Befehle das Ausgabefenster überfluten.

Wenn Sie die unter **Quittierung** die Option **Dpm42-Syntax** auswählen, werden die Kommandos nicht wie in obigen Tabellen beantwortet, sondern mit den Antworten, die bei alten Dpm42-Messsystemen verwendet wurden. (Die Antwort auf `Insert:` ist dann beispielsweise nicht `1` sondern `<R>Inserted: 1`.) Wenn die Prüfstandssteuerung ein Programm verwendet, das für ein altes DPM42-System geschrieben wurde, können Sie dieses mit Hilfe der **Dpm42-Syntax**-Option weiterverwenden. Mit der Option „**–> [Kommando]**“ erhalten Sie die in diese Dokumentation beschriebenen Basic-Standard-Antworten, und dahinter das Kommando, zu dem diese Antwort gehört (also z.B. „`1 [Insert]`“).

Kommando-Plugins

Ein *Decoder-Plugin* ist ein Modul, das als Erweiterung des TasAlysers in die Konfiguration eingetragen wird. Ein Decoder-Plugin erweitert die Funktionalität des TasAlysers um zusätzliche Kommandos und Funktionen.

Dies bedeutet, dass die im Folgenden beschriebenen Plugins Projekt-abhängig vorhanden sein können. Dies hängt von Umfang und Inhalt des Projektes ab.

Ein Plugin kann auch Kommandos aus dem oben beschriebenen Standard-Umfang abändern oder Standard-Kommandos mit anderen Befehlstexten ausstatten.

DummyCommands-Plugin

Das DummyCommands-Plugin beantwortet Kommandos von der Prüfstandssteuerung mit voreingestellten Antworten, ohne dass die Kommandos sonst etwas bewirken. Damit können beliebige Steuerbefehle quittiert werden. Es können auch Standard-Kommandos (etwa `Measure:`) abgefangen werden.

Kommando	Argument	Rückgabewert
`(beliebig)`	`(irrelevant)`	Wie gewünscht

Die Liste der Kommandos und Antworten wird im Applikationssee gespeichert:

```
TasDecoderDummyCommands: {
      RESET  <R1>:RESET
}
```

Mit diesem Beispiel fängt das Plugin das Kommando „RESET:“ des Prüfstands ab und beantwortet es mit „<R1>:RESET“, ohne dass innerhalb des Messprogramms sonst irgendetwas passiert. Der optionale Doppelpunkt hinter dem Kommando sowie etwaige Argumente werden nicht aufgeführt.

Zu beachten ist, dass bei den Kommandos Groß- und Kleinschreibung unterschieden wird. Das obige Beispiel fängt also `RESET:` ab, nicht aber `Reset:`. Die Antworten können nur aus jeweils einer Zeile bestehen. (Falls eine Antwort Leerzeichen enthält, muss sie in obiger Liste in Anführungszeichen eingeschlossen werden.)

Weiterhin ist das DummyCommandsPlugin in der Lage, Kommandos in andere Kommandos zu übersetzen (z.B. ein vom Prüfstand gesendetes „Messung“ in „Measure“ zu verwandeln). Außerdem kann es in Kommandos die Namen von Prüfzuständen austauschen (beispielsweise aus „3-Z“ ein „3-rD“ machen).

Alle diese Funktionen werden durch Einträge in der Datei `Application.sea` des Messprojektes konfiguriert. Das Einstellfenster des Plugins gibt weitere Hinweise.

Prüfstandsfehler übertragen: das „ExtError“-Plugin

Dieses Plugin erlaubt es der Prüfstandssteuerung, zusätzliche Fehlermeldungen in die Fehlerliste der Geräuschprüfung einzufügen. Diese Fehler werden wie Geräuschfehler behandelt, d.h. führen zu einer n.i.O.-Bewertung, werden archiviert und in der Messwerte-Datenbank gespeichert, usw.

Die Befehle „`SetExtError`“ oder nur „`ExtError`“ sind Teile des Standard-Kommandoumfangs. Das Plugin behandelt zusätzlich den Befehl `CheckForError`:

Kommando	Argument	Rückgabewert
`SetExtError:`	`Fehlercode` *`Wert Grenze Position`*	1: Fehler akzeptiert 0: Fehler nicht akzeptiert, weil keine Prüfung läuft. 2: Fehlercode unbekannt
`ExtError:`	`Fehlercode, Fehlercode...`	(wie oben)
`CheckForError:`	`Fehlercode`	1: Fehlercode kommt in aktuellen Fehlern vor 0: Fehlercode kommt nicht vor

Die Angabe von Wert, Grenze usw. für `SetExtError` ist optional. In der einfachsten Form lautet der Befehl

```
SetExtError: 1234
```

Optional kann ein Messwert (Fließkommazahl mit Dezimal*punkt*) angegeben werden. Ebenfalls optional können zusätzlich ein Grenzwert und eine Position (ebenfalls Fließkommazahlen) übermittelt werden. Die Elemente werden durch (beliebig viele) Leerzeichen getrennt. Beispiel:

```
SetExtError: 1234  14.7  10.0  1200
```

Nicht angegebene Werte werden auf Null gesetzt. Folgende Punkte sind zu beachten:

- Der `SetExtError:`-Befehl kann nur während eines Prüflaufs, d.h. nach `Insert:` und vor `Remove:`, verwendet werden.
- Es können mit einem Befehl mehrere Fehler, mit Kommata getrennt, übermittelt werden. Beispiele:

  ```
  ExtError: 309  14.7  10.0  1200, 312 159.4 150.0 800
  ```

  ```
  SetExtError: 309 , 312 , 433
  ```

- Der Fehlercode ist eine Zahl[2]. Jeder Fehlercode muss vor der Verwendung in der Parameterdatenbank des Messprojektes angelegt und die Details (insbes. der Fehlertext) spezifiziert werden. Die Auswahl der Fehlercodes ist völlig frei, d.h. es können beliebige Zahlen verwendet werden, sie müssen weder fortlaufend sein noch bei 1 beginnen. Null ist nicht als Fehlercode zulässig. (Es ist natürlich darauf zu achten, dass es nicht zu Überschneidungen mit den Akustik-Fehlercodes kommt.)
- Der Fehlercode im `SetExtError`-Befehl muss nicht notwendig identisch zu dem Fehlercode sein, der im Messprogramm für diese Fehlermeldung verwendet wird. Die Zuordnung Befehls-Fehlercode zu internem Fehlercode wird ebenfalls in der Parameterdatenbank getroffen. In den meisten Fällen wird jedoch der `SetExtError`-Fehlercode gleich internem Fehlercode sein.
- Es ist nicht möglich, innerhalb eines Prüflaufs zwei Fehler mit demselben Code zu melden. Das zweite `SetExtError:` mit einem bereits verwendeten Fehlercode überschreibt die erste Fehlermeldung. Es können aber beliebig viele Fehler mit unterschiedlichen Fehlercodes gemeldet werden.

[2] Genauer gesagt eine positive ganze Zahl $< 2^{31}$

- Indem man `SetExtError:` mit einem negativen Fehlercode aufruft (also z.B. `SetExtError: -309`), kann man den entsprechenden Fehler-Eintrag wieder löschen.

Durch den `SetExtError`-Befehl wird den Ergebnisdaten der Akustikmessung eine Fehlermeldung hinzugefügt und ggf. das Prüfergebnis auf n.i.O. gesetzt. Es wird jedoch *kein Messwert* eingetragen, auch wenn mit `SetExtError` ein Wert übermittelt wird. Dieser Wert erscheint nur in der Fehlermeldung, nicht in den Messergebnissen. Um einen Messwert einzutragen, muss das weiter unten beschriebene **GetValues**-Plugin (s. Seite 22) benutzt werden.

Mit `CheckForError: 1234` kann geprüft werden, ob in den Fehlerberichten zur aktuellen Messung der Fehlercode 1234 vorkommt. (Dies ist eine einfache Alternative zur Verwendung eines der `Report: Codes` Befehle und des Durchsuchens der Liste aller Fehlercodes.)

Das Schaltvorgangs-Plugin

Das Schaltvorgangs-Plugin behandelt die Kommandos „SGW“ (=“Start Gangwechsel“) und „EGW“ (=“Ende Gangwechsel“). Durch diese Befehle werden Module gesteuert, die Schaltvorgänge überwachen, etwa das Schaltgeräusch.

Dieses Plugin kennt und bearbeitet die beiden folgenden Kommandos:

Kommando	Argument	Funktion	Rückgabewert
`SGW:`	`Gang1 Gang2`	Startet die Messung des Schaltvorgangs von Gang 1 nach Gang 2. Beispiel: `SGW: 2 3`	1: Messung gestartet 0: wenn das Kommando syntaktisch nicht korrekt war. 2: wenn schon ein `SGW` „unterwegs“ ist
`EGW:`		Beenden der Messung des aktuellen Schaltvorgangs	1: keine Fehler aufgetreten (i.O.) 0: Fehler aufgetreten (Schaltung n.i.O.) 2: wenn es zuvor kein gültiges `SGW` - Kommando gab.

Das SGW-Kommando erwartet die Angabe zweier Gänge (Schaltung von-nach). Die Gang-Namen sollten „physikalische Gänge“ (also R, 1, 2, 3...) sein, nicht Prüfzustände. Als Bezeichnung für den Leerlauf sind „N“, „L“ oder „0“ (Null) vorgesehen. Tatsächlich gibt es aber für das Plugin keine Beschränkung oder Überprüfung der Gang-Namen (außer, dass es zwei sein müssen). Die Gang-Namen werden erst von den Bewertungsmodulen verwendet, die durch das Plugin angestoßen werden.

Auf `EGW:` antwortet das Messprogramm mit dem Bewertungsergebnis. Hier kann durch die Auswertung eine kleine Verzögerung (weniger als ½ Sekunde) auftreten. Eventuelle Fehlermeldungen z.B. der Schaltgeräuschprüfung kann nach `EGW` mit einem der `Result`-Kommandos, beispielsweise `InstrResult:`, abgefragt werden.

Beispiel:

`SGW: 2 3`	Kommando: Start des Schaltvorgangs vom 2. in den 3. Gang
`1`	Antwort: Schaltvorgangs-Messung gestartet
`EGW:`	Kommando: Ende des Schaltvorgangs
`0`	Antwort: Schaltkraft-Messung war n.i.O.

Im Messprogramm kann die Option eingeschaltet werden, dass auf EGW: mit drei Ziffern geantwortet wird, durch Leerzeichen getrennt:

`EGW:` → `a b c` `(a, b, c jeweils = 0 oder 1)`

Die Ziffer `a` gibt weiterhin das Gesamtergebnis für den Schaltvorgang an, die Ziffer `b` das Ergebnis für die Gang-‘raus-Messung und Ziffer `c` das Ergebnis für Gang einlegen.

Hinweis: im Messprogramm wird eine maximale Messzeit für den Schaltvorgang angegeben (typischerweise 10 Sekunden). Wenn das `EGW`-Kommando nicht innerhalb dieser Messzeit auf das `SGW`-Kommando folgt, wird die Messung abgebrochen.

Das Übersetzungsprüfungs-Plugin

Dieses Plugin ist erforderlich, um die Übersetzungsprüfung und die Differenzialprüfung anzustoßen. Es interpretiert die folgenden Kommandos:

Kommando	Argument	Funktion	Rückgabewert
`StartRatioTest:`	_	Startet die Übersetzungsprüfung	1
`StartDiffTest:`	_	Startet die Differenzialprüfung	1
`EndRatioTest:` `EndDiffTest:`	_	Beendet eine laufende Prüfung	1: keine Fehler aufgetreten (i.O.) 0: Fehler aufgetreten (n.i.O.)
`GetRatioValue:`	Prüfzustand	Fragt das gemessene Ratio ab	(Messwert)
`GetInvRatioValue:`	Prüfzustand	Fragt den Kehrwert des gemessenen Ratio ab	(Kehrwert des Messwerts)

Durch die ersten beiden Kommandos wird die Übersetzungs- bzw. Differenzialprüfung gestartet. Die Übersetzungsprüfung bezieht sich auf den aktuell eingelegten Gang. Beide Prüfungen werden mit `EndRatioTest:` oder `EndDiffTest:` (gleiche Funktion) abgeschlossen; die Antwort auf `End-RatioTest:` ist das Bewertungsergebnis. Hier kann durch die Auswertung eine kleine Verzögerung (weniger als ½ Sekunde) auftreten. Falls kein `EndRatioTest:` gesendet wird, endet die Prüfung automatisch mit dem nächsten Rampenende oder `Measure: 0` Kommando.

Mit den Kommandos `GetRatioValue:` bzw. `GetInvRatioValue:` kann die gemessene Übersetzung zu einem Prüfzustand bzw. deren Kehrwert abgefragt werden. Die Antwort auf beide Kommandos besteht in einem (als Text ausgegebenen) Zahlenwert:

```
GetRatioValue:  3-D
4.186
GetInvRatioValue:  3-D
0.239
```

Wird der Messwert zu einem Prüfzustand abgefragt, in dem kein Ratio gemessen wurde, lautet die Antwort 0.

Werte auslesen und setzen: das „Get/SetValues“-Plugin

Das GetValues-Plugin ermöglicht das Abfragen und auch das Setzen von Messwerten. Die Messwerte können einzeln oder als Liste abgefragt werden.

Um die Messgrößen zu identifizieren, werden in der Parameterdatenbank im Formular **Schlüssel-Aliase** Namen (genannt *ValueKeys*) für alle Messgrößen festgelegt, die abgefragt oder gesetzt werden sollen. Nur Messgrößen mit einem *ValueKey* können abgefragt bzw. gesetzt werden. Als zusätzliche Information muss bei Verwendung der Befehle der Name des Prüfzustandes übergeben werden, für den die Werte geliefert oder gesetzt werden sollen. (Wird kein Prüfzustandsname übergeben, wird der aktuelle Prüfzustand verwendet.)

Die Werteabfrage kann auf zwei Weisen erfolgen: für einzelne Werte oder für die Liste aller Werte.

Das GetValues-Plugin kennt dazu die folgenden Befehle:

Kommando	Argument	Rückgabewert
`GetValueByName:`	`ValueKey Prüfzustand`	Liefert die Daten zu einem einzelnen Wert.
`GetValueKeys:`	-	Liste der *ValueKeys* wie unten beschrieben
`GetValueList:`	`Prüfzusandsname`	Liste der Werte zu den *ValueKeys* wie unten beschrieben.
`SetValueByName:`	`ValueKey Prüfzust. Wert (Grenze) ((Position))`	1 bei Erfolg. 0, wenn der ValueKey oder der Prüfzustand unbekannt sind.

`SetValuesFromFile:`	`File name`	1 bei Erfolg; 0, wenn Fehler aufgetreten sind.

Abfrage einzelner Werte

Um einen einzelnen Wert abzufragen, verwendet der Prüfstand den Befehl `GetValueByName`. Die Antwort ist eine Zeile, die einen oder mehrere Zahlenwerte enthält:

```
GetValueByName: RMS-Wert 3-Z
2.45 2500.00 12.0
```

In der Parameterdatenbank wird für jeden *ValueKey* festgelegt, was geliefert werden soll. Je nach Einstellung dort erhält man (in dieser Reihenfolge) den Messwert, eine Positionsangabe, den Grenzwert und den Lern-Mittelwert. Im obigen Beispiel ist also 2,45 der Messwert für RMS im Prüfzustand 3-Z, und 12,0 der Grenzwert. Die Bedeutung der Positionsangabe hängt von der Messgröße ab: für Spektralwerte beispielsweise ist die Position die Spektral-Ordnung des Wertes.

Falls der abgefragte *ValueKey* in der Datenbank nicht parametriert wurde, lautet die Antwort „`undefined`“. Falls der *ValueKey* bekannt ist, aber im fraglichen Prüfzustand kein Messwert vorliegt, lautet die Antwort „`n.a.`“.

Wird bei der Abfrage der Prüfzustand weggelassen (also nur „`GetValueByName: RMS-Wert`“), dann wird der Wert für den aktuellen Prüfzustand geliefert.

Die Anzahl der Nachkommastellen kann im Messprogramm eingestellt werden (Doppelklick auf das Plugin in der Systemkonfiguration).

Verfügbare Werte

Ein *ValueKey* kann jede beliebige Messgröße im System bezeichnen, nicht nur Einzahlkennwerte. Mit den `GetValue`-Befehlen erhält man:

Bei *Einzahlkennwerten* den Messwert und, wenn in der Datenbank bestellt, auch Grenze, Position und Lern-Mittelwert.

Bei *Pegelverläufen* und *Spektren* (allgemein bei Kurven) die maximale Differenz von Messkurve zur Grenzkurve. Falls die Messkurve komplett unter der Grenzkurve liegt (Messung ist i.O.), so ist diese Differenz < 0 und gibt die dichteste Annäherung der Messung an die Grenze an. Überschreitet die Messkurve die Grenze, ist der Wert > 0 und liefert die höchste Überschreitung der Grenze. Es wird immer nur dieser Differenz-Wert geliefert, kein Grenzwert oder Position. Wenn die Messgröße keine Grenzkurve hat (z.B. weil die Messgröße nicht bewertet wird), ist die Differenz 0.

Für *Spektrogramme* und andere mehrdimensionale Messgrößen kann kein Wert abgefragt werden. Das Ergebnis lautet immer „`n.a.`“.

Abfrage aller Werte

Sollen alle Messwerte abgefragt werden, verwendet der Prüfstand das Kommando `GetValueList`.

Um die Übertragung der Werte möglichst kompakt zu halten, wird nicht mit jedem Wert sein *ValueKey*-Name mitgeliefert, sondern nur eine Schlüssel-Nummer (*KeyNr*). Die Zuordnung zwischen *KeyNr* und *ValueKey* muss separat abgefragt werden.

Der Befehl

```
GetValueKeys:
```

veranlasst das Messprogramm zur Übermittlung der Zuordnungstabelle zwischen *KeyNr* und Bezeichnungen. Die Antwort besteht aus mehreren Zeilen:

```
<ValueKeys>
key1 "Bezeichner1"
key2 "Bezeichner2"
...
```

```
</ValueKeys>
```

Beispiel:

```
<ValueKeys>
7 "RMS-Wert"
44 "LKon_H1"
</ValueKeys>
```

Die Bezeichner (*ValueKeys*) werden immer in Anführungszeichen eingeschlossen. Es werden nur diejenigen Bezeichner aufgeführt, für die im Formular **Schlüssel-Aliase** unter „Export“ nicht „keine“ ausgewählt ist. Die *ValueKeys*-Liste kann leer sein (wenn die entsprechende Tabelle leer ist oder keine Messgröße aktiviert wurde). Die `GetValueKeys`-Abfrage kann jederzeit (unabhängig vom Beginn eines Prüflaufs durch `Insert`) ausgeführt werden.

Die **Schlüssel-Aliase** -Liste ist unabhängig von Prüftyp und Prüfzustand. Die Bezeichnungen können beliebig gewählt werden, dürfen aber keine Leerzeichen, Doppelpunkte oder Anführungszeichen enthalten. Die Bezeichnungen *müssen nicht eindeutig* sein! Es ist erlaubt, für verschiedene Messgrößen denselben Namen einzuführen. Das kann z.B. sinnvoll sein, wenn die Einträge semantisch dasselbe Objekt bezeichnen, welches aber in unterschiedlichen Prüfzuständen unterschiedliche Datenbank-Schlüssel hat (etwa Wellen in Doppelkupplungsgetrieben).

Die Schlüssel-Nummern (K*eyNr*) sind ganze Zahlen, und sie sind eindeutig innerhalb der Tabelle (auch wenn Bezeichnungen mehrfach auftreten). Es wird allerdings weder garantiert, dass sie fortlaufend sind, noch dass sie erhalten bleiben, wenn Änderungen an der Datenbank vorgenommen werden. Es wird daher empfohlen, die GetValueKey-Abfrage vor jedem Prüflauf erneut durchzuführen. Die <ValueKeys> -Ausgabe ist nicht nach den *KeyNrs* sortiert.

Abfrage der Werteliste

Es werden immer alle Werte für einen Prüfzustand gemeinsam abgefragt. Das Kommando lautet

```
GetValueList: (Prüfzustand)
```

Zum Beispiel:

```
GetValueList: 4-NZ
```

Daraufhin werden die Messwerte und, wenn in der Parameterdatenbank bestellt, auch Positionen, Grenzen und Lern-Mittelwerte der Messgrößen aus der **ClavisFavoriten**-Tabelle zurückgeliefert, in der Form

```
<Values Prüfzustand>
key1 Wert1 (Position1) (Grenze1) (Lern-MW1)
key2 Wert2 (Position2) (Grenze2) (Lern-MW2)
...
</Values>
```

Beispiel:

```
<Values 4.NZ>
44 75.00 98.34 88.62
7 2500.00 3.78
</Values>
```

Schlüssel und Wert werden durch ein Leerzeichen getrennt. Wenn zu einem Schlüssel auch Position usw. gefragt sind, werden diese hinter dem Wert ebenfalls durch Leerzeichen abgetrennt aufgeführt. Die Liste kann leer sein (s.o.). Sie ist nicht nach den Schlüssel-Nummern sortiert.

Die <Values>-Liste enthält alle Werte aus der **ClavisFavoriten**-Tabelle, die bestellt sind und die in diesem Prüfzustand laut Parametrierung vorhanden sein *sollten*. Wurde einer dieser Werte nicht gemessen (z.B. weil eine Zieldrehzahl nicht erreicht wurde), so wird statt des Wertes „`n.a.`“ ausgegeben:

```
<Values 4-NZ>
...
```

```
44 n.a.
...
</Values>
```

Werte, die zwar in der Tabelle stehen, aber in diesem Prüfzustand nicht gemessen werden, werden auch nicht bemängelt. (Beispielsweise könnte der Ordnungspegel eines Zahnrades fehlen, weil sich dieses Zahnrad in diesem Prüfzustand nicht dreht. Dies wird nicht bemängelt, sondern die Zeile fehlt einfach.)

Werte setzten

Die Prüfstandssteuerung kann dem Messsystem Werte übermitteln, die zusammen mit den akustischen Messergebnissen gespeichert werden. Dazu dient der Befehl

`SetValueByName:` *`ValueKey Prüfzustand Wert (Einheit) (Grenze) (Position)`*

Die Angabe von Einheit, Grenze und Position ist optional. Beispiel:

```
SetValueByName: OilTemp Function 89.4 °C 100
```

setzt den Wert 89.4 °C für die Messgröße „OilTemp“ im Prüfzustand „Function“ (beides muss in der Parameterdatenbank angelegt sein). Als Grenze wird 100.0 gesetzt, die Position wird nicht angegeben.

`SetValueByName` muss nach `Insert:` und vor `Remove:` verwendet werden. Beachten Sie, dass zwischen allen Elementen, insbesondere auch dem Zahlenwert und der Einheit, mindestens ein Leerzeichen, Komma oder Semikolon stehen muss. Als Einheiten können nur die im TasAlyser vordefinierten Einheiten verwendet werden (die man z.B. auch im Sensor-Definitions-Formular in der Parameterdatenbank ablesen kann).

Werte setzen aus einer Datei

Anstatt viele Werte einen nach dem anderen mit dem `SetValueByName`-Kommando zu setzen, kann die Prüfstandssteuerung auch eine Textdatei erzeugen, die viele Werte enthält, und dann diese Datei in die TasAlyser-Ergebnisse importieren lassen. Das Kommando lautet:

`SetValuesFromFile:` *`Filename`*

Der ‚Filename‘ kann ein absoluter Pfad sein (z.B. `C:\tmp\ExternalValues.txt`) oder relativ zu einem Standardverzeichnis, das in den Einstellungen des Plugins im TasAlyser festgelegt wird.

Die Text-Datei enthält einen Wert pro Zeile mit derselben Syntax wie bei `SetValueByName` (siehe oben):

`ValueKey Prüfzustand Wert (Einheit) (Grenze) (Position)`

Der Name „ValueKey“ und der Prüfzustand müssen in der Parameterdatenbank angelegt sein. Einheit, Grenze und Positions-Wert sind optional und können weggelassen oder durch einen Strich – ersetzt werden. Die einzelnen Elemente jeder Zeile müssen durch Leerzeichen, Komma oder Semikolon getrennt werden. Hier einige Beispiele für gültige Zeilen (unter der Annahme, dass die entsprechenden ValueKeys und Prüfzustände vorhanden sind):

```
Fir1     3-D   17.5  -  20.0
Fir1, 3-C, 22,7; -; 25
OilTemp Function 89.4 °C
```

Leerzeilen sind erlaubt, und Zeilen, die mit einem Semikolon ; beginnen, werden als Kommentarzeilen betrachtet.

Das Kommando wird mit 1 quittiert, nachdem die Datei erfolgreich verarbeitet wurde, oder mit 0, falls ein Fehler aufgetreten ist. Nach erfolgreicher Verarbeitung wird die Datei gelöscht.

Die Textdatei kann die ASCII-Codierung oder UTF-8 (mit BOM) verwenden.

Das Kommando kann während eines Prüflaufs auch mehrmals verwendet werden. Falls Werte doppelt vorkommen, so überschreibt das letzte Auftreten alle früheren. Beachten Sie, dass mit dem Beginn der Messung für einen Prüfzustand alle bisherigen Werte für diesen Prüfzustand gelöscht werden, einschließlich solcher, die mit SetValue-Befehlen gesetzt wurden. Verwenden Sie diese Befehle daher erst nach den regulären Messungen.

Vektoren aus einer Datei setzen

Mit `SetValuesFromFile` ist es möglich, Vektor-Daten (wie z.B. ein Spektrum) zu übertragen. (Dieses Feature ist für den `SetValueByName`-Befehl nicht verfügbar.) Zum Setzen eines Vektors wird eine Zeile in der Datei mit `$V` begonnen und folgende Informationen angegeben:

```
$V ValueKey Prüfzustand x0 delta dlen (x-Einheit) (y-Einheit)
```

Danach müssen exakt `dlen` Zeilen mit jeweils einer Fließkomma-Zahl folgen (mit Dezimalpunkt; siehe oben).

x0 ist die erste x-Position des Vektors (häufig 0 oder 1), *delta* ist die Schrittweite der x-Achse, und *dlen* die Gesamtzahl an Werten. Beispielsweise könnte ein Spektrum *x0*=0, *delta*=0.5, *dlen*=512 haben, womit der höchste x-Wert bei 255.5 ist. Die Angabe der Einheiten ist optional.

Beim Anlegen des ValueKey für einen Vektor in der Parameterdatenbank muss ein Clavis-„Instrument“ gewählt werden, das Vektordaten erzeugt, wie z.B. Spektrum oder Zeitsignal.

Zusatzfunktionen

Das „GetValues“-Plugin kennt noch zwei weitere Befehle:

`EndMode:`	–	1: Prüfzustand beendet; 0: Fehler
`LookupResource:`	`Quelle Ressource-Name`	Die Werte-Zeile eines Ressource-Datei-Eintrags

Der Befehl `EndMode` beendet eine Prüfzustand (der mit `Mode:` eingelegt wurde), ohne einen anderen Prüfzustand zu aktivieren.

Mit `LookupRessource` kann eine Zeile der Messprogramm-internen Parameterdateien abgefragt werden. Als Quellen-Bezeichnungen sind „Application“, „Status“ und „Messages“ möglich. Falls die Ressource nicht vorhanden ist, wird eine leere Zeile (bestehend aus nur einem Leerzeichen) zurückgeliefert.

Das Triggerparameter-Plugin

Die Durchführung von Rampenmessungen über einer Führungsgröße (also z.B. eine Drehzahl-Rampe von 1500 bis 3200 UpM) wird im Messsystem durch sogenannte *Trigger* gesteuert. Die Trigger-Parameter (also beispielsweise, dass die Drehzahlrampe im 3. Gang Zug von 1500 bis 3200 UpM gehen soll), werden in der Parameter-Datenbank verwaltet.

Mit Hilfe des Triggerparameter-Plugins können die Trigger-Parameter durch die Prüfstandssteuerung geändert werden. Außerdem kann die Kalibrierdatei umgeschaltet werden:

Kommando	Argumente	Rückgabewert

`SetTriggerParams:`	`Trigger-Name`
	`Prüfzustand`
	`Startwert`
	`Endwert`
	`Schrittweite`
	`Sensor`
	`Dimension`
		?: Syntaxfehler (z.B. Argument fehlt)
		0: nicht ausgeführt (z.B. falscher Name)
		1: Parameter gesetzt
		(Bei eingeschalteter „Dpm42-Syntax“ wird den Antworten jeweils „`<R>SetTriggerParams:`“ vorangestellt.)

`SelectCalibration`	`Name der Kalibrierdatei`	?: Dateiname fehlt 0: Datei existiert nicht 1: Erfolg

SetTriggerParams

Die ersten vier Argumente sind erforderlich; ab `Schrittweite` können sie (von hinten) weggelassen werden. Ein Beispiel:

```
SetTriggerParams: Standard 3-Z 1500 3200 25
```

setzt für den Prüfzustand 3-Z die Parameter des Triggers „Standard“ auf eine Rampe von 1500 bis 3200 mit der Schrittweite 25. (Der Sensor bleibt unverändert wie in der Parameterdatenbank angegeben, die Dimension ist implizit 1.) Möchte man auch den Sensor ändern, kann der Befehl so aussehen:

```
SetTriggerParams: OrderTrack 3-Z 1500 3200 25 DzAbtrieb
```

Weiterhin sind folgende Punkte zu beachten:

- Trigger-Parameter können nur *geändert* werden, aber nicht *erzeugt*. Es muss in der Parameter-datenbank für den fraglichen Trigger und Prüfzustand bereits Werte geben, die dann durch den SetTriggerParams-Befehl geändert werden.
- Der `SetTriggerParams`-Befehl mit den Parametern für einen Prüfzustand muss gesendet werden, bevor dieser Prüfzustand das erste Mal mit dem `Mode`-Kommando eingelegt wird. Andererseits muss SetTriggerParams nach dem `Insert` gesendet werden. (Man kann aber nach dem `Insert` die `SetTriggerParams`-Befehle für alle Prüfzustände zusammen schicken.)
- Die gesetzten Parameter gelten bis zum Ende des aktuellen Prüflaufs (oder bis sie mit einem erneuten `SetTriggerParams` geändert werden). Bei Wiederholung eines Prüfzustandes innerhalb des Prüflaufs müssen die Parameter also nicht erneut gesendet werden.
- Für steigende Rampen (z.B. Zug-Rampen) ist der Startwert kleiner als der Endwert (z.B. 1500 3200). Bei fallenden (Schub-) Rampen ist der Startwert größer als der Endwert (etwa „`SetTriggerParams: Standard 3-S 3200 1500`“).
- Die Namen der Trigger und der Sensoren sind der Parameterdatenbank zu entnehmen.

Da die Trigger-Parameter unmittelbaren Einfluss auf die Gewinnung von Messwerten haben, führt eine Änderung der Triggerparameter fast immer auch zu einer Änderung der Messergebnisse und somit auch der Bewertungsergebnisse!

SelectCalibration

Das Argument des `SelectCalibration`-Kommandos ist der Dateiname der entsprechenden Kalibrierdatei, ohne Dateierweiterung. Die Kalibrierdateien befinden sich üblicherweise im Unterverzeichnis *Locals* des Projektordners und sind xml-Dateien.

Beispiel: das Kommando

```
SelectCalibration: Calib-X1
```

aktiviert die Kalibrier-Informationen aus der Datei Calib-X1.xml.

Das Kommando muss vor dem `Insert` verwendet werden.

Das Sensorkonfigurations-Plugin

In manchen Projekten kommt es vor, dass beispielsweise von mehreren Geräuschsensoren jeweils nur einer für die Prüfung verwendet wird, jedoch im Wechsel für unterschiedliche Prüfungen. Für solche Situationen werden in der Parameterdatenbank sogenannte Sensor-Konfigurationen definiert. Es ist möglich, innerhalb der Parameterdatenbank jedem einzelnen Prüflings-Typ eine der vorhandenen Sensor-Konfigurationen zuzuweisen, so dass eine typabhängige Sensorumschaltung ohne weiteres Zutun allein über die Parameterdatenbank bewirkt werden kann.

Für den Fall, dass die Sensor-Konfiguration nicht vom Typ abhängt, sondern über den Prüfstand gesteuert wird, kann dieser dem Messprogramm über einen eigenen Befehl mitteilen, welche Sensorkonfiguration für die nächste Prüfung zu verwenden ist. Für diesen Befehl gibt es drei gleichwertige Schreibweisen:

Kommando	Argumente	Rückgabewert

`SetSensorConfiguration:`
`SetSensorConfig:`
`SensorConfiguration:`
	`Konfigurations-Name`	0: nicht ausgeführt (z.B. falscher Name)
		1: Parameter gesetzt

Das Argument des Befehls ist der Name einer in der Parameterdatenbank vordefinierten Sensor-Konfiguration.

Der Befehl muss *vor dem* `Insert:`*–Befehl* geschickt werden. Die Umschaltung der Sensor-Konfiguration gilt nur für den unmittelbar folgenden Prüflauf, muss also ggf. vor Beginn des nächsten Prüflaufs wiederholt werden.

Wenn dieses Kommando empfangen wird, erzeugt der TasAlyser automatisch einen Zusatzinfo-Eintrag in den Messdaten, der die gewählte Konfiguration dokumentiert.

Das Sensorkonfigurations-Plugin kennt noch einen weiteren Befehl, der in Zusammenhang mit dem „Signal Guard“-Modul verwendet werden kann. Dieses Modul sorgt für ein Alarmsignal, wenn ein Sensorpegel einen festgelegten Spitzenwert überschreitet, und kann für einen Not-Halt des Prüfstandes verwendet werden.

Kommando	Argumente	Rückgabewert
`SignalGuardSetActive:`	`0 / 1` `Off / On`	1

Durch das Kommando-Argument 0 (Off) wird die Signalüberwachung deaktiviert, entweder bis das Kommando mit einem Argument 1 (On) gesendet wird, oder bis zum Beginn des nächsten Prüflaufs.

Das Umcodierungs-Plugin

Durch das Umcodierungs-Plugin (in Kombination mit der entsprechenden Kommandozentralen-Erweiterung) kann am Ende des Prüflaufs der Aggregate-Typ geändert werden, um beispielsweise eine n.i.O.-Prüfung des ursprünglichen Typs in eine i.O.-Prüfung eines „schwächeren“ Typs umzuwidmen.

Um das Umcodierungs-Plugin zu verwenden, müssen in der Parameterdatenbank für jeden Basistyp entsprechende Umcodierungs-Regeln angelegt werden (bitte konsultieren Sie die entsprechende Dokumentation).

Das Umcodierungs-Plugin kennt folgenden Befehl:

Kommando	Argument	Rückgabewert
`Retype:`	*optional*: Liste von Typ-Namen (durch Leerzeichen getrennt)	0: Umcodierung nicht möglich sonst: der Name des neuen Typs, in den umcodiert wurde

Der `Retype`-Befehl kann nur ein Mal pro Prüflauf verwendet werden. Sinnvollerweise wird man ihn am Ende des Prüflaufs benutzen, vor dem `Remove`. Es empfiehlt sich, vor dem `Retype:` entweder das Kommando `EndOfTest:` oder ein `Mode: $Nil` zu senden.

Das Umcodierungs-Plugin geht folgendermaßen vor:

1. Zunächst wird (im Falle einer n.i.O.-Prüfung) die aktuelle Fehlerschwere ermittelt (d.h. die *Severity* des aktuell „schlimmsten“ Fehlers).
2. Diese *Severity* wird mit den Maximal-*Severities* der in der Parameterdatenbank (für den aktuellen Basistyp) angegebenen Umcodierungs-Typen verglichen. Falls als Argument des `Retype`-Befehls eine Liste von Typ-Namen angegeben wurde, werden nur diese Typen in Erwägung gezogen.
3. Aus der Liste wird derjenige Typ gewählt, der die geringste Maximal-*Severity* (laut Datenbank) hat, die noch über der aktuellen *Severity* liegt. Falls kein solcher Typ in der Liste vorkommt (z.B. weil die aktuelle *Severity* zu hoch ist), kann nicht umcodiert werden, und als Antwort auf das `Retpye`-Kommando wird „0“ (Null) zurückgeliefert. Falls das Prüfergebnis „i.O.“ ist, ist die aktuelle *Severity* 0 und aus der Liste wird der Typ mit der niedrigsten Maximal-*Severity* gewählt.
4. Wenn umcodiert werden kann, wird der Typ der aktuellen Prüfung auf den Umcodier-Typ geändert, sowie das Prüfergebnis auf „i.O.“ gesetzt. Die Typ-Änderung und auch das i.O.-Ergebnis werden in das Messdaten-Archiv (und damit die Ergebnisdatenbank) übernommen. Bei erfolgreichem Umcodieren wird als Antwort auf das `Retype`-Kommando der Name des neuen Typs zurückgemeldet.
5. Zusätzlich wird im Messdaten-Archiv eine „Zusatzinformation“ („Additional Info“) eingetragen des Inhalts „wurde umcodiert von Typ *Xyz*“. Je nach Einstellung im Dialog des Plugins bleiben die Fehlerberichte der Prüfung erhalten (zur späteren Information) oder werden gelöscht.
6. Option 1: Gemäß Einstellung im Dialog kann vor dem Umcodieren ein Messdaten-Archiv mit dem alten Typ, aber dem Ergebnis-Code „Umcodiert“ (statt „n.i.O.“) geschrieben werden. Auf diese Weise können auch die umcodierten Prüfungen in die Ergebnisdatenbank eingetragen werden.
7. Option 2: einfache Änderung der Seriennummer. In der vorhandenen Seriennummer können ab einer wählbaren Zeichen-Position die vorhandenen Zeichen durch den neuen Typ-Namen ersetzt werden. Alternativ kann nach dem Umcodieren der Prüfstand mit dem `Serial:`-Kommando (vor dem `Remove`!) eine neue Seriennummer für den umcodierten Prüfling vergeben.

Den Dialog des Umcodierungs-Plugin erreicht man über die Systemkonfiguration, als Untermodul des **Kommando-Decoders** in der Abteilung **Kommandozentrale**.

RecorderControl Plugin

Dieses Plugin erlaubt es, die Aufzeichnung der Sensordaten in Wave-Dateien fernzusteuern.

Normalerweise geschieht die Aufzeichnung der Wave-Dateien automatisch im Messprogramm. Für besondere Anwendungen, in denen die automatische Kontrolle nicht verwendbar ist, kann die Aufzeichnung per Kommando gesteuert werden.

Kommando	Argumente	Rückgabewert
`WaveRecorderControl:`	`(Aktionscode)`	1 oder 0, Bedeutung nach Kommando

Folgende Aktionscodes (jeweils in drei Varianten, als Zahl oder als Name) sind verfügbar:

Aktionscode	Aktion
`1 / AUTO / Auto`	Automatisches Aufzeichnen ein- oder ausschalten. Ein weiteres Argument gibt an, ob das automatische Aufzeichnen eingeschaltet werden soll (1/ON/On) oder aus (0/OFF/Off). Rückgabewert ist der bisherige Zustand. Ohne ein zweites Argument erhält man nur den Rückgabewert, ohne den Zustand zu ändern.
`2 / REC / Rec`	Statusabfrage: läuft gerade eine Aufnahme (Rückgabewert 1) oder nicht?
`3 / START / Start`	Startet eine neue Aufnahme. Dies ist nur innerhalb eines Prüflaufs (nach „Insert:“) möglich. Die Aufnahme endet automatisch mit dem Prüflauf („EndOfTest:“), oder kann mit dem Aktionscode 6 beendet werden. Als zweites, optionales Argument kann ein Text übergeben werden, der in den Dateinamen der Wave-Datei übernommen wird.
`4 / PAUSE / Pause`	Aufzeichnung pausieren.
`5 / CONT / Cont`	Pausierte Aufzeichnung fortsetzen.
`6 / END / End`	Aufzeichnung beenden, Wave-Datei speichern.
`7 / DEL / Del`	Aufzeichnung abbrechen, Wave-Datei löschen.

Die Aktionen 4 bis 7 interagieren mit der automatischen Aufzeichnungskontrolle des TasAlysers. Beispielsweise pausiert der Aktionscode PAUSE auch eine automatisch gestartete Aufnahme, nicht nur solche, die mit dem Aktionscode START gestartet wurden. END beendet jede laufende Aufnahme, unabhängig davon, wie sie gestartet wurde. Die Rückgabewerte der Aktionen 3 bis 7 sind jeweils 1, wenn die Aktion ausgeführt wurde, und 0, wenn sie nicht ausgeführt werden konnte (z.B. PAUSE, wenn gar keine Aufnahme läuft).

Der Aktionscode START hat keinen Effekt, wenn bereits eine Aufnahme läuft.

Beispiel für die Verwendung: zunächst wird eine Aufnahme gestartet, wobei der Text „MySong“ Teil des Dateinamens wird. Dann wird die Aufnahme pausiert, später fortgesetzt und schließlich beendet.

```
WaveRecorderControl: REC MySong
WaveRecorderControl: PAUSE
WaveRecorderControl: 5
WaveRecorderControl: End
```

Innerhalb eines Prüflaufes können auch mehrere Aufnahmen ausgeführt werden. Der Dateiname der Aufnahme wird nach den Regeln konstruiert, die im TasAlyser beim WaveRecorder-Modul festgelegt werden. Wenn für den Dateinamen der Bestandteil „Zeitstempel“ ausgewählt wird, dann wird die Zeit des Beginns der Aufnahme verwendet (und nicht die Zeit des Beginns des Prüflaufs, wie es bei automatischen Aufnahmen der Fall ist).

Anhang: Serielle, Profibus und UDP-Kommunikation

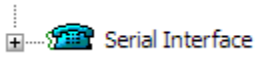
Wie im ersten Teil dieser Dokumentation beschrieben, findet die Kommunikation zwischen dem Rotas-System und der Prüfstandssteuerung grundsätzlich durch den Austausch von Text-Nachrichten statt.

Dieser Austausch kann über verschiedene Medien durchgeführt werden. Das einfachste, robusteste und am weitesten verbreitete Medium in dieser Hinsicht ist eine normale RS232-Schnittstelle. Andere übliche Verfahren sind die UDP- und die Profibus-Kommunikation. Dieser Anhang enthält Hinweise bezüglich dieser Schnittstellen.

Kommunikation über eine serielle Schnittstelle

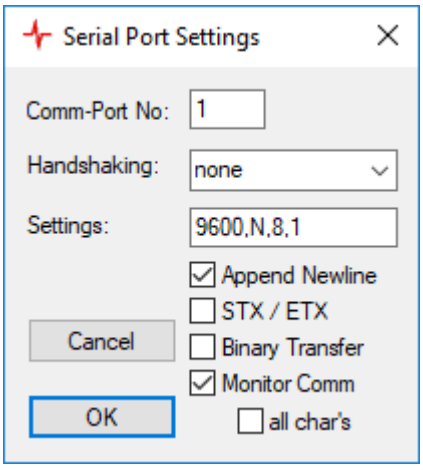
Obwohl (oder weil) die serielle RS232-Schnittstelle aus den Anfangstagen der Computertechnik stammt, ist sie an Einfachheit und Robustheit kaum zu überbieten. Dieses Erfolgskonzept hat seine Fortsetzung im Universellen Seriellen Bus – USB – gefunden, der ja auch zur Anbindung der TAS-Hardware verwendet wird. Selbst moderne Rechner, die keinen seriellen Schnittstellen-Anschluss mehr eingebaut haben, kann man mit einem USB-nach-Seriell-Adapter leicht nachrüsten. Und auch die Übertragungsgeschwindigkeit ist bei gegenwärtigen Computern kein Thema mehr: die kurzen Textbotschaften der Steuerkommandos des Messsystems werden praktisch instantan übertragen.

Die Einrichtung der seriellen Kommunikation im Messprogramm ist sehr einfach: öffnen Sie das Andockfenster „Systemkonfiguration“, klappen Sie den Zweig „Auswertung“ („Evaluation“) aus, und darin den Unterknoten „Kommandozentrale“ („Command Center“) – siehe auch die Abbildung auf Seite 17. In diesem Bereich finden Sie das Modul für die serielle Schnittstelle (es trägt das Bild eines Telefons):



Doppelklicken Sie auf diesen Eintrag, um das Fenster dieses Moduls zu öffnen:

Drücken Sie auf den Knopf **Einstellungen**, um die serielle Schnittstelle zu parametrieren (COM-Portnummer, Baud-Rate usw.). Die üblichen Einstellungen sind 9600 Baud, no parity, 8 Data Bits, 1 Stop Bit, kein Handshaking:



Reset setzt die gewählte serielle Schnittstelle zurück. Beim Start des TasAlysers wird eine Fehlermeldung gezeigt, wenn die ausgewählte Schnittstelle nicht verfügbar ist.

Um die Schnittstelle zu testen, können Sie im Eingabefeld unter **Testen** einen Text eingeben und auf **Senden** drücken. Der Text wird dann über die serielle Schnittstelle ausgegeben. Wenn Sie **Empfangen** drücken, wird stattdessen der Empfang des Textes simuliert.

Im Andockfenster „Ausgabe“ wird die serielle Kommunikation protokolliert. Hier können Sie ablesen, welche Texte verschickt und empfangen werden.

Sollte die serielle Kommunikation nicht auf Anhieb funktionieren, können Sie auch das Hilfsprogramm **VcTerm** zum Testen verwenden. Dieses finden Sie im Discom-Installationsordner, also in C:\Programme\Discom\bin, C:\Program Files (x86)\Discom\bin oder entsprechend.

Beenden Sie das Messprogramm, um die serielle Schnittstelle freizugeben, und starten Sie dann VcTerm.exe.

Über das Menü **CommPort** können Sie die Einstellungen der seriellen Schnittstelle setzen und danach den Port öffnen. Jetzt wird im Hauptfenster von VcTerm alles wiedergegeben, was über die Schnittstelle eintrifft, und jede Textzeile, den Sie eingeben und mit `<Enter>` abschließen, wird über die Schnittstelle weitergegeben.

Profibus/Profinet-Kommunikation

Zur Profibus/Profinet-Kommunikation werden die Messrechner mit einer CIF Profibus-Karte bzw. CIFx Profinet-Karte der Firma Hilscher ausgestattet. Die entsprechende GSD-Datei erhalten Sie von uns auf Anfrage.

Zunächst müssen Sie die Karte konfigurieren. Dazu verwenden Sie das Programm **Sycon** bzw. **netx** der Firma Hilscher. Machen Sie folgende Einstellungen:

- Adressen und Bereiche für die Kommunikation müssen so eingestellt werden, wie von der SPS vorgegeben. „Input“ und „Output“ sind in der Sicht der SPS zu verstehen. Wenn dort zuerst „Input“ definiert wurde, muss das auch im Sycon-Programm so eingestellt werden.
- Der Pufferbereich für die Kommunikation muss genügend groß gewählt werden. Wenn Sie nicht die Übertragung von ganzen Messberichts-Texten planen, reicht eine Puffergröße von 64 Bytes. Der Datentyp des Puffers (Bytes, Wörter, Doppelwörter) ist irrelevant[3], muss aber auf beiden Seiten gleich eingestellt werden.
- Kommunikations-Master: Sie müssen die Adresse der SPS angeben (die vermutlich der Profibus-Master ist). Welchen Typ von Profibus-Karte sie an dieser Stelle wählen, ist irrelevant.

Weitere Informationen finden Sie in der Dokumentation der Firma Hilscher.

Die Kommunikation zwischen Prüfstandssteuerung und TasAlyser funktioniert auch beim Profibus über den Austausch von Texten. Die Texte werden in den Puffer geschrieben, und zwar in folgender Weise:

Byte 0	Byte 1	Byte 2	Byte 3	…			Byte N		
Zähler	Text-Länge incl. 0	Nachrichten-Text (ohne cr/lf am Ende)					0 (NULL)		

Der sendende Kommunikationspartner schreibt zunächst den Text der Nachricht in den Puffer ab Byte 2. Der Text wird abgeschlossen durch ein Null-Byte. Im Byte 1 des Puffers wird die Gesamtlänge des Textes inklusive des Null-Bytes eingetragen.

Achtung: *innerhalb* des Textes dürfen keine Null-Bytes enthalten sein! Beispiel: wenn der Textpuffer die Byte-Werte 65, 66, 67, **0**, 68, 69, 0 enthält, wird der TasAlyser nur den Text „ABC“ sehen, selbst wenn die Text-Länge als 7 angegeben ist. Ein Null-Byte terminiert den Text. Verwenden Sie den Byte-Wert 32 (ASCII Leerzeichen), falls Sie den Text-Puffer initialisieren wollen. Der Textpuffer-Inhalt 65, 66, 67, **32**, 68, 69, 0 wird korrekt als „ABC DE“ interpretiert.

Dann muss eine kurze Wartezeit (25 bis 50 Millisekunden, ein SPS-Zyklus o.ä.) abgewartet werden, damit der Profibus den Inhalt des Puffers komplett übernimmt.

Danach wird der Zähler in Byte 0 des Puffers erhöht. (Nach Erreichen von 255 läuft der Zähler um; wichtig für die Kommunikation ist nur die *Änderung* des Zähler-Wertes.) Durch die Änderung des

[3] Beachten Sie: der *Inhalt* des Puffers ist immer gleich, unabhängig vom bei der Einrichtung behaupteten Datentyp.

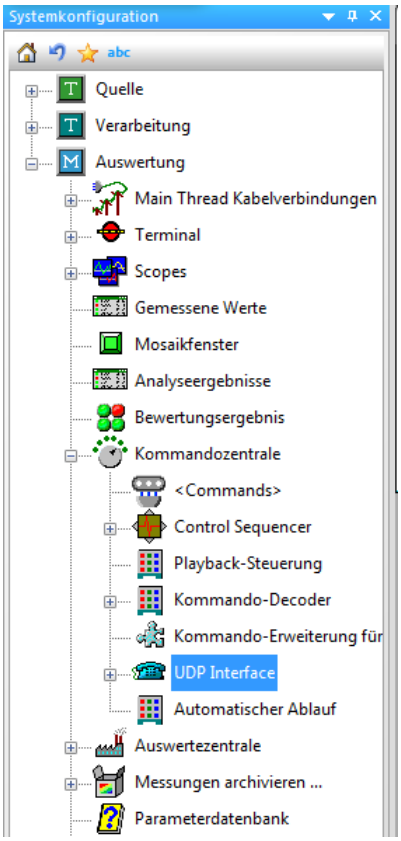
Zähler-Wertes wird die Nachricht freigegeben und dem Kommunikationspartner signalisiert, dass die Nachricht vollständig im Puffer enthalten ist.

Achtung: in den Einstellungen des Profibus-Moduls in der TasAlyser-Software kann die Verwendung des Zählers deaktiviert werden. Wurde der Zähler irrtümlich deaktiviert, so erscheinen vor den Nachrichten-Texten ein bis zwei zusätzliche Zeichen.

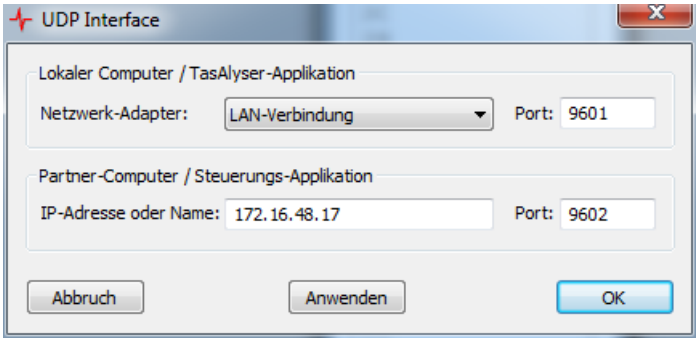
Kommunikation über UDP

Die Prüfstandssoftware steuert das Messprogramm, indem sie UDP-Datenpakete an den Messrechner und dort an einen bestimmten Port schickt. Die Nutzdaten der UDP-Pakete sind die Kommando-Texte[4] (inklusive einem terminierenden Null-Byte). Die Nutzdaten-Länge ist also (mindestens) gleich der Kommando-Länge + 1. Das Messprogramm antwortet auf dieselbe Weise.

Im TasAlyser-Messprogramm müssen die IP-Adresse und der Empfangs-Port der Gegenseite wie auch die eigene Port-Nummer eingestellt werden. Das Modul für die UDP-Kommunikation findet man über die **Systemkonfiguration** in der Abteilung **Auswertung** unterhalb der **Kommandozentrale**:



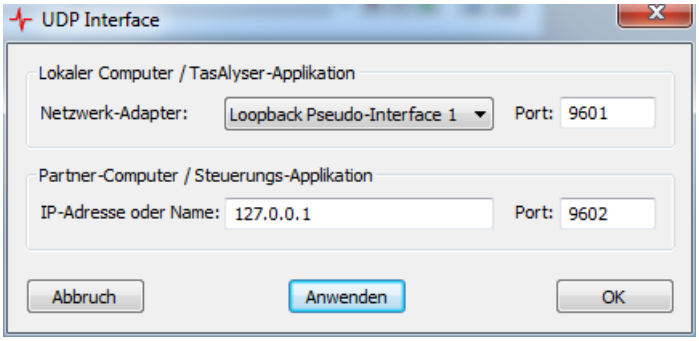
Durch einen Doppelklick auf das Modul im Systembaum öffnet sich der Einstell-Dialog.



Im oberen Feld **Lokaler Computer** können Sie die Port-Nummer ablesen, auf der das TasAlyser-Programm auf Kommandos lauscht (siehe dazu die Anmerkung unten).

Geben Sie im unteren Bereich **Partner-Computer** die IP-Adresse der Prüfstandssteuerung ein, sowie den Port, auf dem die Steuerung auf Nachrichten vom Messprogramm wartet. Statt der IP-Adresse können Sie auch den Namen des Partner-Computers (ohne führende \\ o.ä.) eingeben.

Wenn das Messprogramm und die Steuerungs-Software beide auf demselben Computer laufen, sehen die Einstellungen so aus:



Auch hier könnten Sie statt der IP-Adresse `127.0.0.1` den Namen `localhost` verwenden.

Die Kommunikations-Ports sind prinzipiell frei wählbar. Der empfohlene Empfangs-Port für den lokalen Computer / TasAlyser ist 9601. Dieser Port ist üblicherweise frei – siehe „Liste der standardisierten Ports“ in der Wikipedia:
http://de.wikipedia.org/wiki/Liste_der_standardisierten_Ports

Das TasAlyser-Messprogramm „lauscht“ auf dem eingestellten Port, sobald es gestartet wird. Im Ausgabe-Fenster wird (wie beim seriellen Interface) die Kommunikation protokolliert.

[4] Die Texte werden in herkömmlicher 8 Bit (ASCII) Codierung übertragen, nicht in 16 Bit Unicode UTF-16 o.ä.

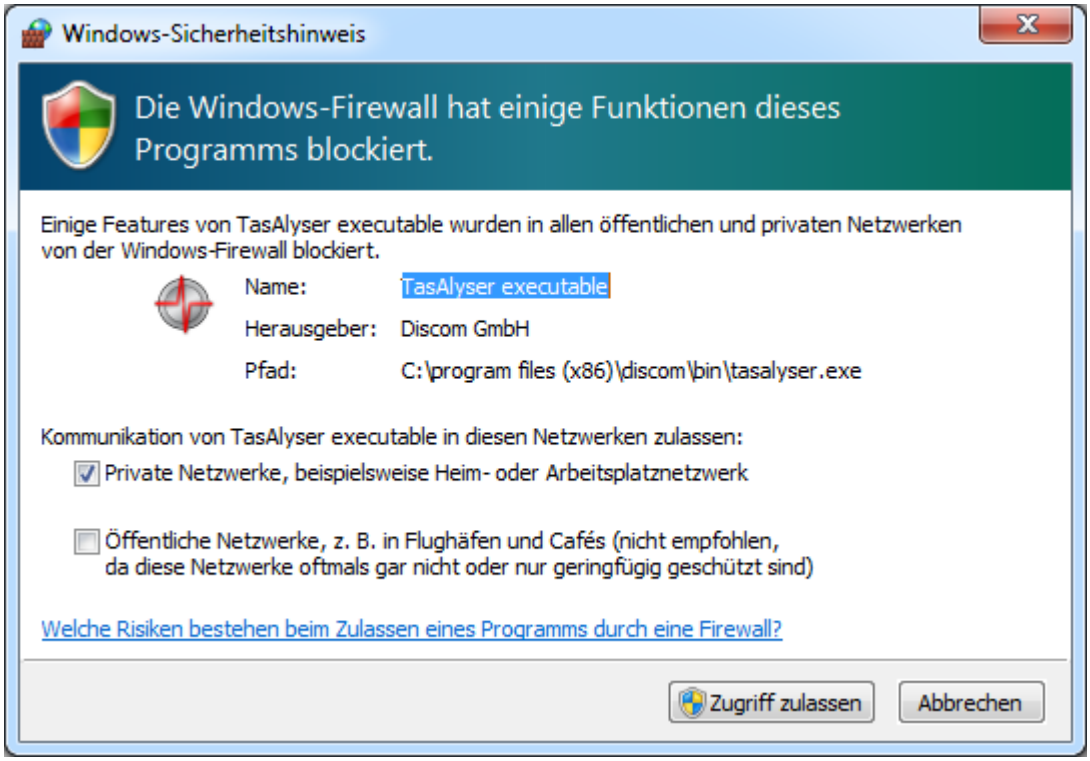
Hinweise zum Einstellen des Netzwerk-Adapters

Der Messrechner kann mit mehreren Netzwerk-Anschlüssen ausgestattet sein, z.B. mehreren LAN-Karten, Drahtlosnetzwerk, VPN und mehr. Einer davon muss für die UDP-Kommunikation ausgewählt werden.

Der Dialog des UDP-Moduls bietet automatisch die Liste aller im System konfigurierten und aktiven Netzwerk-Adapter an. Diese Auswahl wird eingeschränkt auf diejenigen Adapter für IPv4, da das UDP-Modul selbst nur IPv4 verwendet.

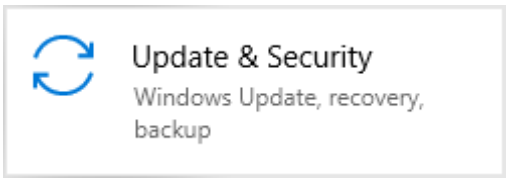
Sie finden die Liste der Netzwerkadapter in der Systemkonfiguration (Control Panel) in der Abteilung **Netzwerk und Internet**, Unterabteilung **Netzwerk- und Freigabecenter** (Angaben für Windows7; auf englischen Systemen „Network and Sharing Center“). Hier wird die aktive Netzwerk-Verbindung angezeigt. In der linken Spalte des Fensters findet sich die Funktion **Adaptereinstellungen ändern**. Klicken Sie auf diesen Link, um zur Liste der verfügbaren Netzwerkadapter zu gelangen.

Beim allerersten Start des TasAlysers auf einem Computer, und auch beim ersten Mal, wenn Sie einen neuen Netzwerkadapter konfigurieren, wird der Windows Defender sich von Ihnen die Erlaubnis einholen, dass der TasAlyser über Netzwerke kommunizieren darf:

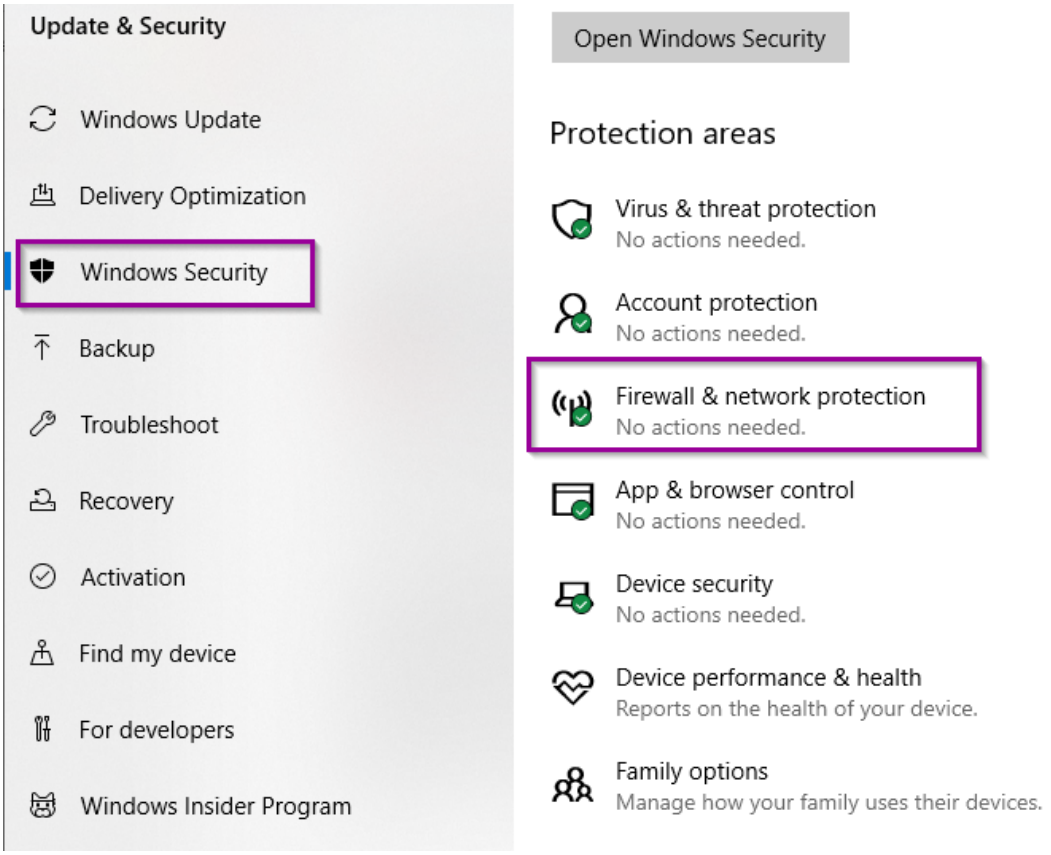


Diesen Zugriff müssen Sie zulassen, damit die UDP-Steuerung funktioniert.

Sollten Sie diese Erlaubnis aus Versehen verweigert haben, oder nicht sicher sein, dass die Einstellung korrekt ist (z.B. weil die UDP-Kommunikation trotz korrekter Port-Einstellungen nicht funktioniert), können Sie die Firewall-Einstellungen über das Windows Control Panel überprüfen und korrigieren. Wählen Sie dazu im Windows Control Panel den Bereich „Update & Security“:



In dieser Abteilung wählen Sie die „Windows Security“ und dort auf der rechten Seite den Bereich „Firewall & Network Protection“:



Innerhalb von „Firewall & Network Protection“ klicken Sie rechts auf „Allow an app through firewall“. Dadurch öffnet sich ein neues Fenster mit einer Liste von Applikationen, die Netzwerk-Kommunikation betreiben. Scrollen Sie bis zum *TasAlyser executable* und setzen das Häkchen beim entsprechenden Netzwerk-Typ – dies wird normalerweise das „private“ Netzwerk sein: