

---

# remes

L I P S

Listengestützte  
Prüfstandsteuerung  
Version 3.0

© remes 1996  
rechnergestütztes Messen und Prüfen GmbH  
An der Lukasmühle 7/3  
85435 Erding  
Tel.: 08122/9723-0



---

# INHALTSVERZEICHNIS

1. Allgemeine Beschreibung	5
2. Definition der Konfigurationsdateien	6
BESTUECK.SYS (ASCII), Rechnerkonfiguration	6
PSDEKL.SYS (ASCII), Konfiguration des Prüfstands	6
Sollwerte	7
Istwerte	8
Bitmasken	9
Formelkanäle	9
Umskalierung von analogen Sollwerten	10
3. Steuerung der Bildschirmdarstellung	11
Vorgabe der Laufmaske	11
Deklaration der Platzhalter	11
Typ und Bezeichnung	12
Farbe und Hintergrund	13
Update	13
Schriftart (Auswahl von Breitschrift)	13
Meßwertabhängige Attribute	14
4. Syntax einer Prüflistendatei (*.LIP)	15
Kopfteil	15
Definition von Kanallisten	15
Die Programmliste	16
1. Schrittausgabe	16
2. Rampenzeile	16
3. kontinuierliche Analogdatenausgabe (ADAM)	17
4. Zuweisungszeile	17
5. Sprungzeile, bedingte Sprungzeile	17
6. Marke	17
7. Sichern	17
8. Installieren oder Abbauen einer Displayzeile,	18
9. CALL : Aufruf einer Anwenderroutine	19
10. Neuaufbau eines Bildschirms	19
11. Gang n	19

12. Status /Statusvorgabe	19
13. Messung	19
Zusatzzeilen	20
1. Messen	20
2. Zusatzrampe	20
3. Regelung	20
4. Warten auf Ereignis	21
5. Programmbedienung	23
6. Die Benutzerschnittstelle	29
Benutzerprozeduren	29
Benutzervariablen	31

---

# 1. Allgemeine Beschreibung

---

LIPS ist ein Prüfstandskontrollprogramm. Es gibt Sollwerte aus, erfasst Meßdaten und stellt den Prozess auf dem Bildschirm dar.

Der Prüfablauf wird dabei vom Anwender in Form von Fahrlisten definiert. Die Fahrlisten enthalten Kommandos für die Sollwertvorgabe in Stufen, Rampen oder Kurvenform, für das Erfassen, Darstellen und Speichern von Meßwerten, für die Steuerung der Bildschirmdarstellung und zur Prüflaufsteuerung.

Kommandos werden zeit- oder ereignisgesteuert abgearbeitet. Die Sollwerte werden je nach Vorgabe als Analogwert, Bitmuster oder als serielle Zeichenkette (RS232) ausgegeben.

Das Programm stützt sich auf einen IBM-kompatiblen Rechner, der wahlweise mit Ein-Ausgabekarten (Modular-4), einer 8-fach seriellen Schnittstelle und einem Zähler/Zeitgeber ausgerüstet ist.

Der Prüfablauf und die Prüfumgebung ist in den nachfolgend aufgeführten Datensätzen festgehalten.

BESTUECK.SYS definiert die Ausrüstung des Rechners. Dabei sind bei Modular-4 Karten (SORCUS) nur die Kartenadressen anzugeben, die tatsächliche Ausrüstung mit Modulen wird aus dem EEPROM der Module gelesen.

PSDEKL.SYS enthält alle Informationen über Verdrahtung und Kalibrierung des Prüfstands. Daneben können auch Formelverknüpfungen und nichtlineare Zusammenhänge zwischen Steuerspannung und physikalischem Sollwert festgelegt werden.

Die Maskendateien \*.LMS steuern die Bildschirmdarstellung.

Die Steuerdateien \*.LIP endlich enthalten die Kommandos zur Steuerung des Prüflaufs.

Bei der Ausgabe von Kurven (ADAM = Analog Daten Ausgabe vom Massenspeicher) müssen die Ausgabedaten im "DIGISKOP"-Format vorliegen. (Verwandte Formate (z.B. .EVA-Daten) können automatisch bei Information des Anwenders korrigiert werden.) Die Zuordnung von Meß- und Ausgabekanälen, der auszugebende Abschnitt und der Ausgabetakts müssen einmal beim Start der Prüfung festgelegt werden.

Meßdaten werden in "Meßjobs" erfaßt, welche in der Prüfliste aufgerufen werden. Jedem Meßjob wird ein Parametersatz zugeordnet, in dem Werte wie Meßtakt, Puffertiefe, Kanalliste usw. abgelegt sind. Die Standardmeßprozedur mißt von den angegebenen Kanälen der Kanalliste im vorgegebenen Takt. Es können aber auch beliebige eigene Prozeduren eingegeben werden. Meßjobs können an eine Ablaufzeile gebunden werden oder nur zeitkontrolliert ablaufen. Mehrere Jobs mit unterschiedlichen Parametern können parallel aufgesetzt werden.

Die Sicherung der Meßdaten erfolgt "jobweise" automatisch bei einem Prüflaufstop oder unter Programmkontrolle. Jedes Sichern wird in einer Directory-Datei festgehalten. Außerdem kann jedem Prüflauf ein eigenes Unterverzeichnis zugeordnet werden. Unterbrechungen des Prüflaufs werden in einer Protokolldatei festgehalten, in der neben Rundenzähler, Schritt und Zeitpunkt auch noch Platz für einen optionalen Kommentar ist.

---

## 2. Definition der Konfigurationsdateien

---

### BESTUECK.SYS (ASCII) , Rechnerkonfiguration

Pro Karte ist eine Zeile eingetragen, die jeweils 3 Parameter enthält. Die Parameter sind mit (") eingeklammert und durch "," getrennt.

Parameter 1: Kurzbezeichnung der Karte (SOR, RTC oder MOXA)  
Parameter 2: Port adresse der Karte (hexadezimal)  
Parameter 3: ein Kommentar, könnte z.b. die Seriennummer enthalten

Nicht bekannte Kurznamen werden ohne Fehlermeldung überlesen.

Beispiel für eine Zeile der BESTUECK.SYS

"RTC", "310", "PCL-830 SN xxx"

Anmerkung: Ist auf einer SORCUS ein Modul MD40, so kann in der BESTUECK.SYS im zugehörigen Kommentar durch einen Zusatz der Form OB:X die Anzahl der Outputbytes festgelegt werden. X ist dabei [0...5]

### PSDEKL.SYS (ASCII), Konfiguration des Prüfstands

Die Prüfstandsbeschreibung ist in 5 Bereiche unterteilt:

- Sollwerte
- Istwerte
- Bitmuster
- Formeln
- Angaben zur nichtlinearen Skalierung

Die Bereiche sind voneinander durch Zeilen getrennt, welche mit einem "-" beginnen.

Kommentarzeilen beginnen mit einem Apostroph (') und dürfen überall vorkommen

## Sollwerte

Hier wird für jede Steuereinheit des Prüfrechners der zugehörige Sollwert festgelegt.

Jeweils eine Zeile beschreibt einen Sollwert, sie enthält maximal 7 Parameter, welche mit (,) getrennt und mit (") eingerahmt sind.

1. Parameter	langer Name
2. Parameter	kurzer Name
3. Parameter	Kanalnummer
4. Parameter	Kalibrierwert 0V
5. Parameter	Kalibrierwert 10V
6. Parameter	Benennung
7. Parameter	Rampensteilheit

### "langer Name"

Wird verwendet beim Anlegen einer Sicherungsdatei und beim Darstellen eines Bezeichners in einer Maskendatei.

### Über "kurzer Name"

Wird auf den Kanal in der Fahrliste und in der PSDEKL (Umskalierungen) zugegriffen. Daneben entscheidet dieser Parameter auch über die Ausgabekarte

Der Anfangsbuchstabe "S" ist reserviert für serielle Ausgabe (z.B. Schaltplott)

"D" ist für digitale Ausgabe reserviert, wobei das zweite Zeichen des Kurznamens als Bytenummer ausgewertet wird. Die Ausgabebytes sind dabei fortlaufend durchnummeriert, die Reihenfolge wird durch die Anordnung in der BESTUECK.SYS festgelegt und innerhalb einer Modular-4 durch den Steckplatz. Ein OPT1B-Modul beherbergt 2 Ausgabebyte, ein MD40-Modul soviel wie in "Outputbytes" festgelegt ist. Möglich ist ein Wert zwischen 0..5. Vorgabewert ist 5, d.h. alle Bytes des Moduls werden zur Ausgabe verwendet.

"R" ist ebenfalls für Digitalausgabe reserviert, wird aber z.Zt. nicht unterstützt.

Alle anderen Anfangsbuchstaben werden einer Analogausgabe zugeordnet.

### Die Kanalnummer

Analogwerte:

Legt den Ausgabekanal und das Ausgabegerät fest. Hierbei werden die in der BESTUECK.SYS definierten Ausgabekanäle oder Ausgabebytes fortlaufend durchnummeriert. SORCUS-Module werden mit aufsteigendem Steckplatz nummeriert. Sind z.B. auf einer Modular-4 auf Steckplatz 3 und 4 DAC4-Module montiert, dann würden dem Modul auf Steckplatz 3 die Nummern 0..3, dem Modul auf Steckplatz 4 die Nummern 4..7 zugeordnet. Wäre auf einer weiteren Modular-4 noch ein DAC4 montiert, dann würden diesem Modul die Nummern 8..11 zugeordnet. Bei digitalen Kanälen wird diese Eingabe ignoriert

Digitalwerte:

Dieser Wert wird ignoriert, da die Bytenummer Teil des Kurznamens ist.

serielle Kanäle:

Die Systemnummer der seriellen Schnittstelle

### Kalibrierwert 0V

Wird bei Digitalkanälen ignoriert.

Bei Analogkanälen ist es derjenige physikalische Wert, dessen Vorgabe die Ausgabe von 0V bewirkt.

### Kalibrierwert 10V

Ist dieser Wert bei einem Digitalkanal kleiner 0, dann bewirkt er die Ausgabe des invertierten Bitmusters.

Bei einem Analogkanal ist es derjenige physikalische Wert, dessen Vorgabe die Ausgabe von 10V bewirkt.

## Benennung

Ist ein optional anzugebender Wert, er wird bei Bedarf dokumentiert.

Bei seriellen Kanälen wird allerdings hiermit die Schnittstelle definiert in der Form "bbbb spd"

bbbb : Baudrate z.B. 9600

s : Anzahl der Stopbits [ 1,2]

p : Parität [N,E,O]

d : Datenbits [7,8]

## Rampensteilheit

Ist ebenfalls optional und gibt den maximalen Wert an, mit dem diese Ausgabegröße automatisch geändert wird. Eine Änderung wird vom Programm nur beim Start von kontinuierlichen Meßwertausgaben dann vorgenommen, wenn die aktuellen Werte mit den Startwerten der Datei nicht übereinstimmen.

## Istwerte

Hier wird für jeden Eingangskanal des Prüfrechners der zugehörige Istwert festgelegt.

Jeweils eine Zeile beschreibt einen Istwert, sie enthält maximal 7 Parameter, welche mit (,) getrennt und mit (") eingerahmt sind.

- |              |                   |
|--------------|-------------------|
| 1. Parameter | langer Name       |
| 2. Parameter | kurzer Name       |
| 3. Parameter | Kanalnummer       |
| 4. Parameter | Kalibrierwert 0V  |
| 5. Parameter | Kalibrierwert 10V |
| 6. Parameter | Benennung         |
| 7. Parameter | Mittelung         |

### "langer Name"

Wird verwendet beim Anlegen einer Sicherungsdatei und beim Darstellen eines Bezeichners in einer Maskendatei.

### Über "kurzer Name"

Wird auf den Kanal in der Fahrliste und in der PSDEKL (Formelvorgaben) zugegriffen.

## Kanalnummer

Legt das Eingabegerät (ADC) fest. Hierbei werden die in der BESTUECK.SYS definierten Eingangskanäle oder Ausgabebytes fortlaufend durchnummeriert (16 pro AD16). SORCUS-Module werden mit aufsteigendem Steckplatz nummeriert. Sind z.B. auf einer Modular-4 auf Steckplatz 3 und 4 AD16-Module montiert, dann würden dem Modul auf Steckplatz 3 die Nummern 0..15, dem Modul auf Steckplatz 4 die Nummern 16..31 zugeordnet.

## Kalibrierwert 0V

Ist derjenige physikalische Wert, der vom Sensor mit der Ausgabe von 0 V angezeigt wird.

## Kalibrierwert 10V

Ist derjenige physikalische Wert, der vom Sensor mit der Ausgabe von 10 V angezeigt wird.

## Benennung

Ist ein optional anzugebender Wert, er wird bei Bedarf dokumentiert.

## Mittelung

Ist ein Parameter, welcher die Darstellung des Istwertes in einer Ausgabemaske steuert. Je größer dieser Wert ist, desto öfter wird der Istwert gemittelt, bevor er auf dem Bildschirm ausgegeben wird.

## Bitmasken

Werden Digitalkanäle in der Fahrliste über ihren Kurznamen (z.B. D2) angesprochen, dann muß immer das gesamte Byte angegeben werden. Durch die Deklaration von Bitmasken ist es möglich, einzelne Bits oder Bitgruppen anzusprechen, ohne den Zustand der übrigen Leitungen dieses Bytes zu kennen. Da bei der Vorgabe auch die Bytenummer festgelegt wird, ist damit der Zugriff auf Digitalleitungen über Symbolnamen realisiert.

Folgende 5 Parameter sind vorzugeben

### Name

über diesen Namen wird auf die Bitgruppe zugegriffen. Wird in der Liste dem Namen ein "-" nachgestellt, dann wird die Gruppe invertiert ausgegeben.

Der reservierte Name "SYNC" definiert ein Bit, welches vor jeder Änderung von Sollwerten durch die Prüfliste gesetzt und nach der Änderung wieder gelöscht wird. Die (logisch) fallende Flanke zeigt also das Anliegen neuer Sollwerte auf den Analogleitungen an. Ist keine Gruppe mit diesem Namen definiert, dann wird auch kein Synchronsignal ausgegeben.

Bitmuster mit Namen START1 .. START4 werden beim Listenstart gesetzt

Bitmuster mit Namen STOP1..STOP4 werden am Programmende gesetzt.

Es ist durchaus erlaubt, dasselbe Bit in mehreren Mustern mit unterschiedlicher Polarität aufzuführen. Natürlich sollten solche Muster dann nicht in ein und derselben Schrittausgabe (DB=...) auftauchen.

### Port

Es ist die via "Rechnerbestückung" vorgegebene Bytenummer anzugeben. Es können nur Bits auf einem Byte zu einer Gruppe zusammengefaßt werden .

### Maske

Es ist die numerische Repräsentation der Bitgruppe einzutragen. Hexadezimale Vorgabe ist hier möglich, sie ist durch ein vorangestelltes "\$"-Zeichen anzuzeigen. Sollen also die Bits 0,4,5 von der Ausgabe betroffen sein, dann kann in der Maske "\$31" oder auch "49" eingegeben werden.

### Wert

Der Wert der Ausgabe ist einzutragen. Wird bei einer vorgegebenen Maske "49" der Wert "16" eingetragen, dann wird bei Aufruf der Gruppe Bit 0 und 5 gelöscht, Bit 4 wird gesetzt. Die anderen Bits des Bytes bleiben unverändert.

### Polarität

Wird ein Wert kleiner 0 eingetragen, dann werden die Bits der Gruppe vor der Ausgabe nochmals invertiert. Beachten Sie, daß die eine eventuelle Anweisung zum Invertieren bei der Definition des Sollwertkanals weiterhin für das gesamte Byte gültig bleibt.

## Formelkanäle

Formelkanäle erlauben die arithmetische Verknüpfung verschiedener Eingabekanäle. Sie können auf dem Bildschirm dargestellt werden, das Ergebnis kann aber auch in einem Meßjob registriert werden.

Die Formel besteht aus der Deklaration eines neuen Namens und der Berechnungsvorschrift. Die Berechnungsvorschrift kann Meßkanäle oder bereits definierte Formelnamen enthalten. Bezüge auf Formeln oder Meßwerte sind mit "." einzurahmen, also

z.B.

$$\text{Pab} = \text{.Nab} \cdot \text{.Mab} / 9549$$

Formelnamen können bei der Deklaration von Ausgabefenstern verwendet werden oder in Kanallisten (sh. Kopfteil der Steuerliste) auftauchen.

Optional kann beim Formelnamen auch eine Benennung in eckigen Klammern angegeben werden.

Wird auf eine Formel in einer Kanalliste zugegriffen, dann sollte zusätzlich "Nullwert" und positiver Maximalwert angegeben werden.

Hintergrund: Wenn ein Meßjob eine Kanalliste abarbeitet, dann werden alle Werte als Integer. Ohne weitere Angaben werden als Umrechnungswerte 0 und 1 genommen. Dies kann bei kleinen physikalischen Werten zu Rundungsfehlern, bei physikalischen Werten über 32767 zu völlig falschen Werten führen. Durch die Angabe des Maximums wird das vermieden.

Die Formelangabe kann also auch erweitert geschrieben werden

z.B.

$$Pab [kW] = .Nab. * .Mab. / 9549 \# 0,300$$

## Umskalierung von analogen Sollwerten

Ist die Umsetzung von vorgegebener Sollspannung zu dargestelltem physikalischen Wert nicht ausreichend linear, dann kann für diesen Sollwert eine nichtlineare Umsetzung zwischen Sollwert und ausgegebener Sollspannung vereinbart werden. Beachten Sie, daß trotz einer solchen Vorgabe die korrekte Eingabe der Stützpunkte 0 und 10V in der Sollwertdeklaration notwendig ist. Die Umskalierung kann über eine Tabelle von Stützpunkten oder über eine analytische Funktion eingegeben werden.

Syntax bei Stützpunktvorgabe:

Die erste Zeile enthält nur den Kurznamen des Sollwertes, in den folgenden Zeilen ist in beliebig vielen Wertepaaren die Umskaliervorschrift in Tabellenform aufgezeichnet, die zwei Werte pro Stützpunkt können durch ein ",", Wertepaare untereinander durch "/" getrennt werden. Die Tabellenwerte gewinnen Sie wie folgt: Arbeiten Sie mit der linearen Sollwertvorgabe und tragen Sie in die Tabelle für jeden Stützpunkt das Wertepaar "tatsächlich dargestellter Sollwert", "von LIPS vorgegebener Sollwert" ein.

Syntax bei Formelvorgabe:

Kann ein analytischer Zusammenhang zwischen den beiden Werte hergestellt werden, dann kann an Stelle der Tabelle eine Formel vorgegeben werden. Die Syntax ist dann

$$\text{"kurzer Name"} = f(.X.)$$

Ist .X. der bei bei linearer Skalierung von LIPS vorgegebene Sollwert, dann soll f(.X.) der dann vom Prüfstand tatsächlich dargestellte physikalische Wert sein.

Beispiel:

$$A = 141 * \text{SIN} (.X./100*3.1416/4)$$

---

## 3. Steuerung der Bildschirmdarstellung

---

### Vorgabe der Laufmaske

Das Layout des Bildschirms während des Prüflaufs wird weitgehend (bis auf 4 "Systemzeilen" ) durch die Laufmaskendatei bestimmt (\*.LMS).

Die Maskendatei kann beim Programmeinstieg gewählt werden. Sie enthält beliebigen Text mit eingestreuten Platzhaltern,

Jeder Text, der nicht Platzhalter ist, erscheint zur Laufzeit auf dem Bildschirm, die Platzhalter werden je nach Deklaration laufend durch aktuelle Werte ersetzt.

Ein Platzhalter ist eine Folge von # welche von . oder 0 unterbrochen sein können z.B. ####.# oder #####00 . Der Platzhalter gibt an wo und in welcher Formatierung ein Wert innerhalb der Maske ausgegeben werden soll.

Ein Dezimalpunkt legt die Anzahl der Nachkommastellen fest, angehängte Nullen unterdrücken die Ausgabe von Vorkommastellen. Bei einer angehängten Null werden z. B. Zahlen nur in Zehnerschritten ausgegeben.

Die Vorgabe der Laufmaske wird innerhalb der Datei.LMS durch eine Zeile

----- (mindestens 10 führende '-')

getrennt. Danach folgt die Deklaration der Platzhalter

### Deklaration der Platzhalter

Für jeden Platzhalter ist anzugeben, was und in welcher Form an seiner Stelle zur Laufzeit erscheinen soll. Jeder Platzhalter wird durch 1 Zeile beschrieben, die Reihenfolge richtet sich nach dem Auftreten in der Laufmaske ( Die Zählung erfolgt von links nach rechts, dann von oben nach unten).

Eine Deklarationszeile hat folgende Syntax:

Typ Bezeichnung Farbe Hintergrund update Schriftart [ Komparator limit Farbe2 Hintergrund2 ]

Als Trennzeichen sind Leerzeichen und/oder Komma erlaubt.

Typ ::= <A|D|F||S>

Bezeichnung ::= typabhängiger String

Farbe ::= <0..15> { eine EGA-Farbe }

Hintergrund ::= <0..7> { ein EGA-Hintergrund }

update ::= Häufigkeit, mit der der Eintrag erneuert wird

(0 : immer, 1: jeder Schritt, 2 : Zyklus , 3 : Start)

Schriftart ::= <1|2> { 1: normal 2: 2x2 Felder/Zeichen }

Bei Istwerten und Formeln können optional weitere 4 Parameter angegeben werden, mit welchen der angezeigte Wert auf Grenzwertverletzung überprüft werden kann. Wird eine Grenzwertverletzung erkannt, dann wird der entsprechende Wert in einer anderen Farbkombination angezeigt. Außerdem werden globale Variable gesetzt, welche von einer Benutzeroutine abgefragt werden können.

Komparator ::= <|<|>|>

limit ::= <Dezimalzahl>

Farbe2 {siehe Farbe}

Hintergrund2 { siehe Hintergrund }

## Typ und Bezeichnung

Die beiden ersten Parameter bestimmen, was dargestellt wird.

Das erste Zeichen gibt den generellen Typ des Feldes an, die zweite Zeichenkette ist für die Unterauswahl zuständig, welche dieser Typ bietet.

5 generelle Typen sind implementiert :

(I)stwerte, (S)ollwerte, (F)ormeln, (D)isplayzeilen, (A)llgemeine oder interne Parameter.

- I: Es wird ein in der PSDEKL.SYS angegebener Istwert ausgegeben, dieser wird während des Laufs kontinuierlich gemessen. Als Bezeichnung ist der Kurzname des Istwertes einzufügen
- S: Es wird ein in der PSDEKL.SYS angegebener Sollwert ausgegeben, dieser wird während Rampen und ADAM-Läufen kontinuierlich erneuert. Als Bezeichnung ist der Kurzname des Sollwertes einzufügen
- F: Als Bezeichnung kann ein Formelname der PSDEKL.SYS angegeben werden. Enthält die Formel Meßwerte oder vorher definierte Formeln, dann wird der ausgegebene Wert kontinuierlich erneuert.
- D: Eine von drei möglichen Displayzeilen, welche in der Liste definiert wurden, wird hier angezeigt. (Definition einer Displayzeile sh. Definition der .LIP Datei Befehl DISPL) Bezeichner ist <1|2|3>
- A: Damit kann auf programminterne Texte und Größen zurückgegriffen werden. Folgende Bezeichner werden unterstützt:

### NAME<sub>n</sub>

Gibt den ausführlichen Namen (langer Name) eines Ist- bzw. Sollwertes aus der PSDEKL an, der auf einem anderen Platzhalter ausgegeben wird. n ist der Versatz der beiden Platzhalter. +1 bedeutet, der zugehörige Ist- oder Sollwert wird auf dem nachfolgenden Platzhalter ausgegeben. (Numerierung von links nach rechts und von oben nach unten)

### KOMM<sub>n</sub>

Gibt eine Kommentarzeile anstelle des Platzhalters aus. KOMM1 ist der in der Liste durch # markierte Kommentar, KOMM2 ist der beim Prüfstart eingegebene Kommentar

### PROG<sub>n</sub>

Eine Programmzeile der Fahrliste relativ zum aktuellen Programmschritt.

### VERSION

Gibt die Programmvariable "Anwlf": String[20] aus

### STAND

Gibt die Programmvariable "Pruefstand":String[20] aus

Die nachfolgend aufgeführten Größen erscheinen unabhängig von der Laufmaske in einer der 3 Statuszeilen. Sollen sie besonders hervorgehoben werden, dann können sie nochmals aufgeführt werden.

ZYKLUS	aktueller Zyklenzähler
SCHRITT	aktueller Schrittzähler
ZEIT	aktuelle Zeit
DATUM	aktuelles Datum
SZEIT	letzte Startzeit von LIPS
SDATUM	Startdatum
RESTZEIT	verbleibende Schrittzeit
LISTE	Name der Programmliste
LAUF	Name der Protokolldatei

Die restlichen Parameter beschreiben neben der im Platzhalter enthaltenen Formatangabe (Länge und eventueller Dezimalpunkt), wie die genannte Größe ausgegeben wird.

## Farbe und Hintergrund

Diese Parameter legen fest, mit welchen Attributen der Platzhalter auszugeben ist. Es sind Zahlen zwischen 0 und 15 entsprechend der EGA-Tabelle erlaubt. Alle "Nichtplatzhalter" werden in vordefinierten Farben ausgegeben werden. Zur Compilezeit wird die Farbe in LIPS.COS über die Zuweisung zu "bildhint" und "bildvorn" vorgegeben. Diese Voreinstellung kann vom Anwender mit graphischer Unterstützung durch das Programm SELCOLOR geändert werden. Achtung: sollen die Voreinstellungen zur Compilezeit wieder wirksam werden, dann muß die Datei LIPS.COD gelöscht werden

## Update

Mit "update" kann die Häufigkeit des Bildschirmzugriffs pro Platzhalter gesteuert werden.

- 0: der Parameter wird so oft wie möglich aufgefrischt
- 1: einmal neu nach jedem Schrittwechsel
- 2: einmal neu bei jedem Zykluswechsel
- 3: für im Lauf nicht veränderte Größen wie Namen oder Kommentare

## Schriftart (Auswahl von Breitschrift)

"schriftart" kann zwischen Normalschrift und "2x2"- Schrift umschalten. Es können nur die Zeichen "0..9", ".", "-", " " umgesetzt werden (also nur numerische Felder wie Istwerte, Sollwerte oder Formeln). Die Definition der Zeichen ist in der ASCII-Datei BIGCHA.DAT abgelegt und kann auch editiert werden. Beachten Sie aber, daß das Format der Datei nicht geändert werden darf. (Sicherheitskopie anlegen!)

## Meßwertabhängige Attribute

Die restlichen 4 optionalen Parameter können bei Istwerten und Formeln angegeben werden, sie steuern den Farbumschlag bei Über- oder Unterschreiten von vorgebbaren Schwellwerten.

<Komparator><limit> legt die Bedingung für die Ausgabe in anderer Farbe fest, dabei ist <limit> in physikalischen Einheiten des ausgewählten Wertes anzugeben.

Farbe2 und Hintergrund2 legen die alternativen Attribute fest.

eine Angabe

..... < 800 15 4

bedeutet also, daß der Ist- oder Formelwert weiß auf rot ausgegeben wird, wenn er kleiner ist als 800

{ 15: weiß 4 : rot vgl. EGA-Tabelle}

Für die Auswertung der Alarmer in einer Benutzerprozedur stehen folgende globale Variable zur Verfügung:

istwert\_limit : Array[1..maxist] of Boolean, True, falls der zugehörige Grenzwert verletzt ist  
(Numerierung gemäß PSDEKL.SYS)  
formel\_limit : Array[1..maxfor] of Boolean.  
istwertalarm : Byte; der größte Index für den gilt "istwert\_limit[istwertalarm]=True"  
formelalarm : Byte; dito für die Formelkanäle. Sind istwertalarm und formelalarm beide = 0 ,  
dann liegt kein Alarm vor.

---

## 4. Syntax einer Prüflistendatei (\*.LIP)

---

Prüflisten enthalten die prüfungsspezifischen Informationen. Sie enthalten einen Kopfteil mit allgemeinen Daten wie Kommentar, Definition von Meßwertlisten und die Anzahl der vorgegebenen Zyklen für die Laufliste. Daran schließt sich das eigentliche Prüfprogramm an.

### Kopfteil

Beim Einlesen des Kopfteils werden folgende Sonderzeichen behandelt

' :	Die Zeile wird überlesen (erklärender Text)
# :	Diese Zeile kann als KOMM2 in einer Laufmaske ausgegeben werden (vgl. Definition von .LMS Maskendateien)
& :	Die folgende Zahl wird "Sollzyklen" interpretiert
- :	leitet den Anweisungsteil ein

### Definition von Kanallisten

In Kanallisten werden Größen zusammengefaßt, welche von einem Meßjob erfaßt und gespeichert werden sollen. Es kann sich dabei um Istwerte, Programmvariablen oder Formeln handeln.

Die Definition einer Kanalliste wird eingeleitet durch eine Zeile

KANALLISTE name

name ist dabei eine Zeichenkette, über die im Prüfprogramm auf die Liste zugegriffen wird (sh. Definition eines Meßjobs)

Es folgen Zeilen mit Listen von Eingabegrößen, welche mit "," voneinander getrennt sind

Eingabegrößen sind

- Kurznamen von Istwerten
- Formelbezeichnung
- interne Variable (Vxx)  
Damit auch in der Meßdokumentation Kalibrierwerte für diese Meßgröße zur Verfügung gestellt werden können, kann nach der Angabe der Variablen mit "/" getrennt der Kurzname eines Ist- oder Sollwertes folgen. Dies zeigt dem Auswerteprogramm an, nach welcher Vorschrift die gespeicherte Größe in physikalische Werte umzurechnen ist.  
Beispiel: V12/Nab  
Anwendungsbeispiele sind die Sollwertvorgabe über Variable oder die Nachbearbeitung von Istwerten, deren Resultat von der Benutzerprozedur in einer Programmvariablen abgelegt wird.

## Die Programmliste

Das Programm beschreibt sequentiell Vorgaben für den Prüfablauf, wobei Verzweigungen erlaubt sind. Jede Zeile enthält eine neue Vorgabe, wobei in einigen Fällen Erweiterungen möglich sind (z. B. Anweisungen zur Meßwerterfassung oder zur Regelung von Sollwerten).

Allen Zeilen kann ein Kommentar angehängt werden, er ist mit "\*" einzuleiten

Folgende Anweisungszeilen sind erlaubt

### 1. Schrittausgabe

Die Zeile besteht aus Anweisungselementen der Form

Kurzname = Wert

Die Anweisungselemente werden durch "/" getrennt, Wert hat je Typ des angesprochenen Ausgabekanals unterschiedliche Syntax und Bedeutung

- |      |   |
|------|---|
| DAC  | a) Der physikalische Wert, der nach Maßgabe der Kalibrierfaktoren in Digits umgerechnet wird.<br>b) Der Inhalt einer internen Variable welcher ohne Umrechnung ausgegeben wird. Variable könne durch Zuweisung oder Regelung belegt werden.   |
| PIO  | a) Eine Folge von Ziffern [1..8] durch Komma getrennt: die aufgeführten Bits werden gesetzt, die übrigen werden gelöscht<br>b) 0 oder C: alle Bits werden nicht aktiv gesetzt   |
| DB   | Der reservierte Kurzname DB leitet die Übergabe von Bitmustern ein. Mehrere Muster können durch Komma getrennt werden, wird einem Namen ein "-" nachgestellt, dann wird das Muster invertiert ausgegeben<br>DB=FBREMSE-,LUFT1<br>könnte heißen : Setze Lüftungsstufe 1, löse die Feststellbremse                                  |
| SIO  | Eine beliebige Zeichenkette. In ASCII nicht darstellbare Zeichen können durch Hexadezimaldarstellung in spitzen Klammern ( z.B. <0D> für CarriageReturn ) eingegeben werden. Auch ein "/" kann nur über die Hex-Darstellung ausgegeben werden, da das Zeichen sonst als Trennzeichen zum nächsten Sollelement interpretiert wird. |
| Zeit | Dauer des Schritts in Sekunden, maximal 1 000 000 s. Die Zeitauflösung ist 1/60 Sekunde   |

Beispiele

N=5000/D=1,2/ R = C / T=10.5 / \* Comment

S=G1<0D>/T=1

\* reine Kommentarzeile

### 2. Rampenzeile

Eine Rampenzeile bewirkt die kontinuierliche Veränderung eines Parameters. Sprungartige Veränderungen können nicht gleichzeitig angewiesen werden. Wohl aber kann in einer Zusatzzeile ein weiterer Parameter (max 4) kontinuierlich verändert werden. Rampen können nur auf DAC (Kartentyp O,1,2) ausgegeben werden.

Syntax:

R/ Kurzname , Zielwert , t-Ramp , t-gesamt

t-Ramp ist die Zeit innerhalb welcher der Zielwert erreicht wird

t-gesamt ist die Zeit, welche insgesamt in diesem Schritt zugebracht wird

### 3. kontinuierliche Analogdatenausgabe (ADAM)

Wird aufgerufen mit einer Zeile ADAM (name, Zyklenzahl).

Beim Aufruf einer solchen Zeile werden die Daten der angegebenen Datei entsprechend den zugehörigen Parametern über DAC (oder auch PIO) an den Prüfstand ausgegeben. Ist in der PSDEKL ein Bitmuster "SYNC" deklariert, dann wird, wie bei allen anderen Ausgaben, bei jeder Änderung von Sollwerten das Sync-Muster aktiviert (sh. auch Definition der PSDEKL).

Beim Start einer Liste wird für jede ADAM-Zeile mindestens einmal ein Satz von Steuerparametern abgefragt. Hier wird festgelegt, welcher Zeitabschnitt ausgegeben wird und welcher Meßkanal über welchen Sollwertkanal an den Prüfstand geleitet wird. Beachten Sie, daß wenn der Sollwert nicht linear deklariert ist (PSDEKL), auch die Umrechnung der Meßdatei diese nichtlineare Vorgabe berücksichtigt (sh. auch Beschreibung der Parametermaske für ADAM-Dateien im Kapitel Programmbedienung). Optional kann zur ADAM-Ausgabe zu einer graphischen Maske gewechselt werden. Dies wird zur Compilezeit über eine Definition "ADAMGRAF" festgelegt. Ist diese Compilervariable nicht definiert, dann bleibt die ausgewählte Laufmaske aktiv.

### 4. Zuweisungszeile

Sie wird definiert durch das Auftreten einer Zeichenkette := in der Zeile

Syntax :

Vnn := W1 [⟨+|-|MOD⟩ W2 ]

Vnn ist eine interne Ganzzahlvariable V00..V36

Wx kann eine interne Variable oder eine Konstante sein. Als Operatoren werden akzeptiert +,- oder MOD. Operator und W2 sind optional.

Einer Konstanten kann optional eine Umrechnungsvorschrift beigefügt werden ( Syntax ; " /Kurzname"). Das bewirkt, daß der an die interne Variable übergebene Wert vorher entsprechend der Kalibrierung des durch den Kurznamen beschriebenen Ist- oder Sollwertkanal umgerechnet wird.

### 5. Sprungzeile, bedingte Sprungzeile

Sie wird definiert durch das Auftreten von GOTO in der Zeile

Syntax:

[.IF W1 ⟨ >|=|<⟩ W2.] GOTO Lbl1 [ .,Lbl2 .]

Der Bedingungsteil ist optional, ebenso die zweite Marke. Wenn vorhanden, dann ist sie das Sprungziel für 'FALSE'. W1 und W2 können wie in der Zuweisungszeile Konstante oder Variable sein. (sh. 4.)

### 6. Marke

Eine Markenzeile ist definiert durch ":" als erstes lesbares Zeichen. Der anschließende Name kann als Ziel in Sprungzeilen oder bei Ereignisüberprüfung eingesetzt werden.

### 7. Sichern

SICHERN [(rundenzähler[,jobnr])]

Sichert die Puffer von Meßjobs.

Ist "rundenzähler" angegeben, dann wird dann gesichert, wenn der aktuelle Zyklenzähler modulo rundenzähler den Wert 0 ergibt.

Ist die Jobnummer angegeben, dann wird der Puffer des betreffenden Jobs gesichert, andernfalls werden alle Meßjobs gesichert.

### Format der Sicherung

Für jeden Job wird eine Datei mit Daten und eine Datei mit Parametern angelegt. Die Meßdaten werden entsprechend der Kanalliste zeitsortiert abgelegt. In der Parameterdatei werden alle dem Meßjob zugeordneten Daten abgelegt. Beide Dateien werden auf dem in der Startmaske vorgegeben Pfad angelegt. (Jede Prüfung sollte ihre Meßdaten der Übersichtlichkeit halber in einem eigenen Verzeichnis ablegen). Das Format der Dateinamen ist

rrrrrjj.zza

rrrrr ist die Rundenzahl 6-stellig mit führenden Nullen

jj ist die Jobnummer 2-stellig mit führenden Nullen

zz startet bei 00. Wird versucht, eine Datei gleichen Namens anzulegen, dann wird zz inkrementiert, ab zz=99 werden die Dateien überschrieben.

a ist <P|A>, D: Datendatei, P: Parameterdatei

Die Parameterdatei hat folgendes Aussehen:

Sa- 27.06.1992/11:20:31

1 : Messroutine  
24 : Takt  
0 : Delay  
81 : Sollpunkte  
324 : Puffergröße  
324 : Anzahl Speicherwerte

4 Kanäle

M 1	: Motordrehzahl	Nmot	0.0000	3.4180	
M 2	: Abtriebsdrehzahl	Nab	0.0000	3.3984	
M 3	: Motoroeltemp	Tmot	0.0000	0.0977	
M 4	: Kuehlwassertemp	Tmwa	0.0000	0.0977	
	5	26	37	48	59

----

1 : Kennung ob (M)eßwert, (F)ormel oder (I)nterne Variable  
2..4: Kanalnummer  
6..25 lange Kanalbezeichnung  
27..36 kurze Kanalbezeichnung  
38..47 physikalischer Wert bei 0 Digits (offs)  
49..58 physikalischer Wert pro Digit (gain)

d.h. Meßwert = offs + Digits \* gain

Die Dateien werden in einem eigenen Subdirectory abgelegt, welches beim Programmstart ausgewählt oder angelegt werden kann.

In einer Logbuchdatei wird jede Sicherung festgehalten.

## 8. Installieren oder Abbauen einer Displayzeile,

Eine von 3 Displayzeilen kann mit einem String belegt werden. Wird als erstes eine Variable angegeben, dann wird diese, laufend aktualisiert, bis zum Deaktivieren der Zeile ausgegeben.

DISPL (<1|2|3>, String)

'String' ist ein Stringkonstante, welche Platzhalter für Variable enthalten darf (&Vxx& oder &Vxx/n/ziff/nachk&).

Form 1 gibt an der Stelle des Platzhalters die Ganzzahlvariable xx aus,

Form 2 wandelt den Wert entsprechend den Kalibrierfaktoren von Eingabekanal n um und stellt ihn entsprechend der Formatangabe dar. (Voreinstellung 7:1). n ist als Kurzname einzugeben.

Wird eine Ziffer eingetragen, dann bezieht sich diese auf die Reihenfolge in der Prüfstandsdeklaration.

z ist eine von 3 Bildschirmzeilen für Anwendertext.  
DISPL (-z, ) deaktiviert und löscht die Zeile z

## 9. CALL : Aufruf einer Anwenderoutine

CALL n . n gibt die Nummer der gewünschten Anwenderprozedur aus der "LIPSUSER.DLL" an.  
Ist der Prozedur über die DLL-Prozedur "SucheProc" ein Namen zugewiesen, dann kann dieser Name anstelle der Indexnummer verwendet werden. Der Rest der Zeile wird der Prozedur unverändert als Parameter übergeben

## 10. Neuaufbau eines Bildschirms

NEUBILD

## 11. Gang n

Ruft die Benutzerroutine "Gangwechseln" mit n als Sollgang auf. Als "alter Gang" wird die globale Variable "Istgang" übergeben, welche nach dem Prozeduraufruf mit n belegt wird.

## 12. Status /Statusvorgabe

Statusvorgabe : : = [-] Statuswort [-][, Statusvorgabe]  
Statuswort : : = <MESSWART|SIOSTUMM>

Ist in der Statusvorgabe ein "-" enthalten, dann wird der Status rückgesetzt, sonst gesetzt.  
Das Setzen eines Status verändert den Programmablauf wie folgt:

SIOSTUMM: Es wird nach dem Aufsetzen einer SIO-Ausgabe nicht automatisch eine Wartesequenz aufgesetzt.  
MESSWART: Betrifft nur den manuellen Betrieb. Ein Schritt mit Messung wird erst verlassen, wenn die Messung beendet ist.

Als Vorbelegung sind alle Stati zurückgesetzt.

## 13. Messung

Syntax :

MESSUNG xx ("Meßjobdeklaration")

Die Meßjobdeklaration ist identisch mit der nachfolgenden Deklaration eines Meßjobs. Der Prozedur wird die nachfolgende Zeile unverändert übergeben. Der Anwender kann nach Belieben in dieser Zeile Parameter an die Meßprozedur übertragen.

Achtung: Im Gegensatz zum Aufsetzen eines Meßjobs in einer Zusatzzeile werden die Eingaben in den Feldern "DELAY" und "TAKT" vom Programmsystem nicht beachtet, die angegebene Prozedur wird genau einmal und das sofort aufgerufen (natürlich ist es sinnvoll, wenn die aufgerufene Prozedur diese Parameter im Bedarfsfall beachtet). Der Vorteil einer Handhabung als Meßjob liegt in der einheitlichen Parametrierung von Kanallisten, in der Möglichkeit der Benutzerprozedur, Methoden des Meßjobs aufzurufen (z.B. befuelle\_liste) sowie in der Möglichkeit die Daten systemkonform zu sichern

## Zusatzzeilen

Zusatzzeilen werden mit "+" eingeleitet, sie können nur auf Schrittausgaben, Rampenzeilen oder ADAM-Aufrufen folgen.

# 1. Messen

+Mxx(L:liste,D:delay,T:takt,M:messzeit,R:routine,P:puffer,C:con)

Vom Schlüsselwort vor dem Doppelpunkt wird nur der 1. Buchstabe ausgewertet, Sie können um der besseren Verständlichkeit willen die Schlüsselwörter auch ausschreiben.

xx 00 .. 99

- L: Name der Kanalliste, darf auch ein Eingabekanal sein (Liste der Länge 1)
- D: Start der Messung nach Aufruf in Sekunden
- T: Meßtakt in Sekunden
- M: Meßzeit welche der Job aktiv ist. Meßzeit/Takt ist die Anzahl der Punkte in einem Puffer.
- P: Anzahl der Puffer die für den Job reserviert werden. Die Puffer werden zyklisch überschrieben. Wird ein Job gesichert, dann werden damit die letzten P Messungen gespeichert.
- R: Nummer der aufgerufenen Meßroutine. Ist die Routine in der DLL in "SucheProc" aufgeführt, dann darf auch der zugeordnete Namen angegeben werden
- C: 0 Job wird bei Erreichen der Anzahl oder nach Schrittdende abgebrochen  
1 Job wird bei Erreichen des Schrittdendes nicht abgebrochen, sondern immer wieder neu aufgesetzt. Ein kontinuierlicher Meßjob wird durch Aufruf mit C:0 auf normal zurückgesetzt.

Die Angabe eines jeden Parameters ist optional, ohne Angabe wird die Eingabe des letzten Aufrufs dieser Nummer übernommen, bei Programmstart werden alle Meßjobs mit Defaultwerten belegt.

Das Sichern erfolgt auf Dateien mit fortlaufender Nummer mit einem festen Namensanfang, die Parameter der jeweiligen Sicherung werden in eine Logbuch-Datei festgehalten.

# 2. Zusatzrampe

R/ Kanalname , Ziel , tRampe  
(vgl. Rampenzeile). Pro Anweisung sind maximal 4 Rampen erlaubt. +R (A,5,10)

# 3. Regelung

F ( Ausgabekanal, Istwertkanal, Sollwert, dt, Faktor, Vn )

Ausgabekan:

Die errechnete Stellgröße wird auf diesem Kanal ausgegeben

Istwertkan:

Auf diesem Meßwert wird geregelt

Sollwert:

Die Regelung soll erreichen, daß der Istwertkanal diesen Wert annimmt

dt:

Zeitabstand in s in der eine neue Regelgröße errechnet und ausgegeben wird.

Faktor:

Der Sollwert wird um einen Betrag geändert, der proportional zur Regelabweichung ist, "Faktor" ist der Proportionalitätsfaktor

Vn:

Ist eine Variable, auf ihr wird der aktuelle Stellwert abgelegt

Mit "Regelung" kann eine Stellgröße nachgeführt werden, dem Zeitverhalten sind allerdings Grenzen gesetzt (ca. 50 Hz) z.B. +F(Mab,A,200,2,.001,10)

Es dürfen maximal 4 Regelungen pro Schritt aufgerufen werden

---

## 4. Warten auf Ereignis

---

Mit diesem Befehl kann das Eintreten eines bestimmten Ereignisses abgefragt werden. Mögliche Ereignisse sind

- Auftreten eines Bitmusters
- Eintreffen eines Strings über die serielle Schnittstelle
- Über- oder Unterschreiten eines bestimmten Analogwerts

Der Zeitpunkt des Ereignisses kann gespeichert werden. Das Ereignis kann zum Verlassen des Schritts führen, tritt das Ereignis während der Schritts nicht ein, so kann eine Marke angesprungen werden.

a) serielles Ereignis

W ( SIO, String, vn , Marke1, Marke2 )

String

Dieser String wird erwartet, das letzte Zeichen des Strings wird als generelles Schlußzeichen interpretiert, es darf also nicht im restlichen Text vorkommen. Sonderzeichen können im HexFormat eingegeben werden (vgl. Sollwert)

vn

Eine gültige Variablennummer ( 0 .. 36 ) dient als Ziel für die Ablage des Ereigniszeitpunkts (in 1/60 s).

Marke1

Ist Marke1 mit einer gültigen Marke belegt, dann wird nach Eintritt des Ereignisses der aktuelle Schritt verlassen und zur Marke1 gesprungen. Es ist hier auch die Pseudomarkte NEXT erlaubt, welche einfach zum nächsten Schritt weiterschaltet.

Marke2

Trat das Ereignis innerhalb der Schritzeit nicht ein, so wird Marke 2 angesprungen (leer: nächster Schritt)

b) Bitmuster

W ( PIO, Bitmuster, vn , Mrk1, Mrk2)

Bitmuster

Ist ein in der Prüfstandsdeklaration aufgeführtes Bitmuster

Rest wie oben

c) Analogwert

W ( KName , (>,<) , Wert , ..... )

KName

Der Istwertkanal von dem gemessen wird

>,<

Der Operator zeigt an ob Unter- oder Überschreiten als Ereignis zählt.

Wert

Schwellwert in physikalischen Einheiten des Istwertkanals

Rest wie oben ..

## 5. Programmbedienung

LIPS3 wird mit einer "TurboVison"-Oberfläche gestartet. Unter dieser Oberfläche werden angepaßte Editoren zum Korrigieren der Systemdateien bereitgestellt. Daneben können Systemeinstellungen, wie Auswahl der Systemdateien, Vorgabe des Prüfstands usw. vorgenommen werden. z.B. wird bei der Stellung "Bürosystem" auf die Abfrage der Hardwarekomponenten verzichtet, Prüfläufe können damit ohne vorhandene Prüfumgebung vorbereitet werden.

Der Start des LIPS-Kerns erfolgt je nach Konfiguration sofort oder aus der TurboVision-Oberfläche  
Zuerst wird dem Benutzer die Startmaske angeboten

Listenorientierte Prüfstandsteuerung LIPS									
26.04.92					renes-GmbH				
Istzyklen:	34	Soll:	3002	Schritt:	1	Zeit:	0.0	s	
Steuerliste :	SCHALTST.LIP								
Prüfung :	Testprogramm für Adamausgabe								
Prüfling :	Kommentar zum Prüfling								
Laufname :	PROTNAME								
Maskendatei:	MASKE.LMS								
Verzeichnis für Speicherdateien									
	C:\LIPS\LAUF2\								
temporäres Verzeichnis (nur über SET TEMP=.... einzustellen)									
	D:\TEMP\								
eventuelle ADAM-Parameter überarbeiten .....								ja	
1	2	3	4	5	6	7	8	9	10
	Abbruch				Datei wählen		Datei bearbe.		START

Die Startmaske ist immer mit den letzten Parametern vorbelegt und korrekt nachgeführt, so daß bei Unterbrechungen eines Prüflaufs die Maske nur bestätigt werden braucht.

### Die Bedeutung der Parameter

Istzyklen:

Die aus der Protokolldatei gelesene Anzahl der in diesem Prüflauf bereits abgearbeiteten Zyklen

Soll:

Der Wert, der aus der ausgewählten Steuerdatei gelesen wird. Wird aus der beim Aufruf vorgegebenen Datei eingelesen.

Schritt:

Wird aus der zugehörigen Protokolldatei gelesen und gibt an, bei welchem Schritt der Prüflauf zuletzt unterbrochen wurde (und wo er gegebenenfalls fortgesetzt werden soll).

Zeit:

Abbruchzeit innerhalb des Schritts

Steuerliste:

Gibt an, welche Steuerliste bearbeitet werden soll. Befindet sich der Cursor in diesem Feld, dann wird nach Drücken von F 6 die Liste der vorhandenen Listen angeboten, eine neue kann über Rollbalken ausgewählt werden.

Prüfung:

Ist der Prüfkomentar aus der Steuerliste und kann nur in der Steuerdatei geändert werden.

Prüfling:

Dieser Kommentar kann hier geändert werden.

Laufname:

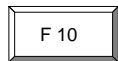
Legt fest, unter welchem Namen Protokoll und Logbuch angelegt werden.

Maskendatei:

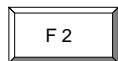
Definiert die Laufmaske welche zur Laufzeit auf dem Bildschirm dargestellt werden soll. Auch in diesem Feld ist F 6 aktiv und erlaubt die cursorunterstützte Auswahl einer neuen Datei.

#### Verzeichnis der Speicherdateien

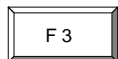
In diesem Subdirectory werden Meßjobs, das Protokoll und das Logbuch gesichert. Wird in diesem Feld F 6 gedrückt, dann erscheint ein Verzeichnispfad, durch den sich der Anwender per Pfeiltasten bewegen kann. Bei der Auswahl sind folgende Funktionstasten aktiv:



Wählt das angewählte Verzeichnis zum Sicherungsverzeichnis



Kreiert ein neues Unterverzeichnis am Ort des Cursors (mit Tastatureingabe des Namens)



Löscht das angewählte Unterverzeichnis auf Wunsch komplett (und unwieder-ruflich!!!)

#### Temporäres Verzeichnis

In dieses Verzeichnis werden Arbeitsdateien kopiert. Es ist die Anzeige der Einstellung der Environmentvariablen TEMP und kann entsprechend nur unter DOS in einem SET TEMP Befehl geändert werden.

#### ADAM-Parameter überarbeiten

nein: Die Abfrage der Laufparameter ja/nein erfolgt nur das erste Mal, d.h. wenn die passende ".LAD"-Datei nicht gefunden wird.

ja: Dem Anwender wird nach Bestätigung der Startmaske in jedem Fall für jede ADAM-Datei die ADAM-Parametermaske angeboten.



In den folgenden Zeilen sind für jeden Meßkanal 3 Werte einzutragen.

Die Zeile beginnt mit dem Namen des Meßkanals, wie er bei der Messung festgelegt wurde.

In der Rubrik "Ausgabekanal" ist einzutragen, über welchen Ausgabekanal der Meßwert auszugeben ist. Es handelt sich um ein "Schalterfeld", d.h. die Eingabe erfolgt nicht alphanumerisch. Statt dessen wird mit den Tasten "+" und "-" zwischen den möglichen Eingaben umgeschaltet.

Angeboten werden:

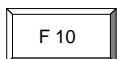
- Alle in PSDEKL deklarierten Ausgabekanäle. Analogkanäle werden dabei so umgerechnet, daß entsprechend den Vorgaben in der PSDEKL und in der Meßwertbeschreibung (.DSP) am Prüfstand derselbe physikalische Wert dargestellt wird, wie er vom Meßwerterfassungssystem registriert wurde.  
Wird ein Digitalkanal als Ausgabekanal angewählt, dann wird das Low-Bytw des aufgenommenen Signals ohne Änderung dem Prüfstand angeboten.
- "Gangsignal" bedeutet, daß dieser Kanal als Gangspur interpretiert wird. Ein solcher Kanal wird nicht ausgegeben, sondern er beeinflußt den Programmablauf. Jedesmal wenn sich der Wert der Gangspur ändert, stoppt die Ausgabekarte den Datentransfer vom PC. LIPS ruft dann die Routine "Gangwechseln" auf und übergibt als Parameter den alten und den neuen Wert der Gangspur. Nach der Rückkehr aus der Prozedur wird mit der Datenausgabe fortgefahren. Die Definition der Prozedur "Gangwechseln" wird im Punkt "Benutzerschnittstelle" erläutert.
- Keine Ausgabe  
Der Meßwert wird ignoriert.

Die Rubriken "y-unten" und "y-oben" werden nur ausgewertet, wenn die ADAM-Ausgabe graphisch visualisiert wird. (Compileroption "ADAMGRAF"). Sie stellen dann die physikalischen Grenzen dar, innerhalb derer der jeweilige Meßwert am Bildschirm dargestellt wird.

Die unterstützten Funktionstasten



Abbruch  
Verzweigt zur TurboVision Oberfläche von LIPS3



Start  
Überprüft die Plausibilität der Eingaben und startet die Fahrliste

## Die Laufzeitmaske

Das Aussehen des Bildschirms zur Laufzeit kann weitgehend über eine Datei festgelegt werden (\*.LMS). Allerdings werden 4 Zeilen (1,23,24,25) vom Programm fest belegt.

Liste :	Lauf :	Start :
Sollwerte		Istwerte
Drosselklappe	: 0.0	
Motordrehzahl	: 0.0	
Binaerausgabe D3	: 00000000	
Schaltpilote	:	
Programmzeilen		
N=1000/A=50/M=700		
N=0/A=0/M=0 /T=3 /* Stillstand		
ADAM (NB754HA1 , 1)		
errechnete Werte		
Abtriebsleistung (kw)		
Runde :	von	Schritt :
Status :	Zeit:	Ist: >
F1: nächster Schritt F3: Start Ablauf F5: Start mit Zeitvorgabe F7: Ende		

### Zeile 1

- Liste: Name der angewählten Steuerliste
- Lauf: Vorgegebener Name für Protokoll und Logbuch
- Start: Datum und Uhrzeit des letzten Prüfstands

### Zeile 23

- Runde: Der aktuelle Zyklenzähler
- von: Die in der Steuerliste vorgegebene Anzahl von Prüfzyklen
- Schritt: Nummer der gerade bearbeiteten Zeile der Prüfliste. Im Status "manuell" wird derjenige Schritt rot unterlegt, bei dem der letzte Abbruch erfolgte.
- von: Anzahl der Zeilen in der Prüfliste.

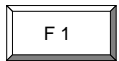
Wird eine ADAM-Zeile aufgerufen bei alphanumerischer Anzeige, dann wird im letzten Viertel der Zeile 23 die Nummer der Runde innerhalb des ADAM-Aufrufs ausgegeben.

### Zeile 24

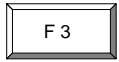
- Status: Manuell oder auto zeigt an, ob die Schrittweilerschaltung entsprechend der vorgegebenen Zeit oder per Tastendruck erfolgt.
- Zeit: Wird nur im Zustand "auto" angezeigt und gibt die maximale Zeit an, welche in diesem Schritt zugebracht wird. Die Zeit kann kürzer sein, wenn eine Zusatzzeile installiert ist, welche ereignisgesteuert die Zeile vorzeitig beendet.
- Ist: Zeigt fortlaufend die aktuelle Schrittzeit alphanumerisch und semigraphisch in 3%-Schritten an. Auch dieser Zeilenteil wird nur im Zustand "auto" angezeigt.

Die Zeile 25 zeigt die aktiven Funktionstasten

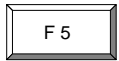
Status manuell:



Springt zur nachfolgenden Programmzeile



Startet ausgehend von der aktuellen Programmzeile



Startet mit dem aktuellen Programmschritt und gestattet die Vorgabe der restlichen Verweilzeit

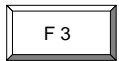
Ist die Anzeige des aktuellen Schrittes rot hinterlegt (d.h. in diesem Schritt werden der Lauf zuletzt unterbrochen), dann wird auch die laut Protokoll fehlende Verweilzeit im Schritt eingeblendet. Für einen möglichst nahtlosen Fortgang der Prüfung sollte dieser Wert eingetragen werden.



Der Prüflauf wird unterbrochen.

Es erscheint ein Fenster, welches die Eingabe zusätzlicher Zeilen ins Protokoll erlaubt. Es kann auch der Protokolleintrag gänzlich unterdrückt werden.

Status auto:



Stoppt den aktuellen Schritt und schaltet in den Status "manuell".

---

## 6. Die Benutzerschnittstelle

---

### Benutzerprozeduren

Die Programmierschnittstelle ist eine dynamische Bibliothek (DLL) welche zur Laufzeit eingebunden wird. Aus dieser Bibliothek wird vom Betriebssystem die Start-, Stop- und Überwachungsprozedur gelesen. Je nach Listenaufwurf kann auf weitere Prozeduren der Bibliothek zugegriffen werden.

Die Standardmeßprozedur (Messung1), welche pro Aufruf die zugehörige Kanalliste in den Meßpuffer überträgt, ist ebenfalls enthalten.

Damit die Benutzerbibliothek auf Systemdaten zugreifen kann, werden bei der Initialisierung der DLL Zeiger auf Systemdaten und Systemprozeduren übergeben. Sie werden weiter unten beschrieben.

Außerdem wird der Prototyp eines Meßjobs mit ausnahmslos virtuellen Methoden in der Datei "LIPTYPES" definiert. Eine Meßprozedur, welche auf Methoden des als Parameter übergebenen Meßjobs zugreifen will, muß allerdings vorher die Prozedur "loadlipsdseg" aufrufen, da sonst die leeren Methoden des Prototypen aufgerufen werden.

Das Betriebssystem ruft beim Start außerdem die Prozedur "SetzeProcPointer" auf und übergibt ihr den in der TV-Oberfläche eingegeben (und im dortigen Desktop gespeicherten) Prüfstandsnamen. Abhängig vom Namen können nun verschiedene Systemprozeduren installiert werden.

Sollen Prozeduren per Namen aufgerufen werden, dann ist dieser Name in die Funktion "SucheProc" einzutragen. Die Funktion wird mit dem Namen als Parameter aufgerufen, falls der Name aufgelistet ist, muß der Zeiger darauf übergeben werden. d.h. soll eine Prozedur Userproc unter dem Namen BENUTZER-PROC aufgerufen werden, dann muß "SucheProc" eine Programmzeile der Form

```
if s='BENUTZERPROC' Then SucheProc := @Userproc;
```

enthalten.

Die Benutzerprozeduren werden nicht nur während des Programmablaufs sondern auch bei der Syntaxprüfung aufgerufen. Dies wird im Flag "action" angezeigt. Eine gut integrierte Prozedur wertet das Flag aus und informiert bei fehlgeschlagener Syntaxprüfung über die Funktion "PFehlerausstieg" (sh. unten) das Programm über das Syntaxproblem.

An Systemprozeduren werden zur Verfügung gestellt

Device = 0..1;

PGetE16 = function (iKnr : Integer) : Integer;  
Holt einen Wert vom AD-Kanal iKnr (sh. PIndexVon)

PKanOut = procedure (oK,SW : Integer);  
gibt den Wert SW am Ausgabekanal oK aus (sh. PIndexVon)

PLiesByte = Function (Nr : Byte) : Byte;  
Liest ein Byte vom Inputbyte "Nr"

PSchreibByte = Procedure ( Nr,wert : Byte );  
schreibt auf das Byte "Nr" den Wert "wert"

PAusgaenge\_wechseln= Procedure;  
Zeigt am "Sync-kanal" das Wechseln der Ausgänge an

PAusgaenge\_stabil = Procedure;  
Beendet die Anzeige am "Sync-kanal"

PReset\_Sorcus = Procedure (Sorcusadresse : Word);  
setzt die angegebene Modular-4 zurück

Save\_Screen = Function : Boolean;  
speichert den kompletten Bildschirminhalt (False wenn kein Speicherplatz)

Restore\_Screen = Function : Boolean;  
stellt den Bildschirminhalt wieder her

PContrCmd	= Procedure ( dev : Device; cmd : Byte; gibt ein Kontrollwort an die Timerkarte aus
PContrDin	= Function ( dev : Device; src : Byte): Word; Liest ein Wort von einem Register der Timerkarte
PContrDout	= Procedure ( dev : Device; dest: Byte; data : Word); schreibt das Wort "data" an das in dev und dest bezeichnete Register der Timerkarte
PFehlerausstieg	= Procedure (i:Integer; Offs : Byte; Zusatz : String) ; Beendet den Programmablauf, wobei i die Fehlernummer übergibt (sh. LIPSFEHL.TXT), Offs ist auf 0 zu setzen. Der String Zusatz wird neben der Fehlernummer im Fehlerfenster ausgegeben.
PIndexvon	= Function (kt : TKanaltyp; name : String) : Integer; TKanaltyp = (ist_kan,sollkan,digikan); Die Funktion sucht zum angegebenen Namen den über die Prüfstandsdeklaration definierten Index
PReal_aus_I	= Function (kt : TKanaltyp; idx,wert : Integer) : Real; wandelt entsprechend der Vorgaben des über kt und idx bestimmten Kanals die Integergröße "wert" in die physikalischen Größe um
PInt_aus_R	= Function (kt : TKanaltyp; idx : Integer; r: Real) : Integer; wandelt umgekehrt eine physikalische Größe in die dem Ein- oder Ausgabe- gerät entsprechende Integergröße um
PBit_in	=Function (idx : Integer) : Boolean; liest den Zustand des Bitmusters "idx"
PBit_out	= Procedure (idx : Integer; wert : Byte); setzt oder löscht (wert=\$ff oder 0) das Bitmuster idx
PRefresh	= Procedure; retriggert den Watchdogtimer
PDauerlife	= Procedure; schaltet bis zum nächsten Aufruf von PRefresh den Watchdog aus
PSendSio	= procedure (sionr : Byte; out : Char); sendet das Zeichen "out" auf COMNr sionr
PReadSio	= function (sionr : Byte) : Char; Liest ein Zeichen von COMNr "sionr"
PChar_available	= function (sionr : Byte) : Boolean; meldet "True" falls ein Byte an COMNr "sionr" anliegt
PSIO_leer	= Procedure (Portnummer : Byte); löscht den Inputbuffer zu ComNr (Portnummer)
PSetSeriell	= procedure (PortNummer,BaudRate,StopBits,DataBits: word;Parity: __ParityType); spezifiziert das angegebene COM-Port neu.

Vom LIPS-Betriebssystem werden aus der DLL aufgerufen :

Startprozedur

Vor dem Abarbeiten der Steuerliste

Checkprozedur

wird während des Prüflaufs kontinuierlich aufgerufen und kann die Variable "pruefstandfehler" im  
Record "LIPSDaten" setzen, was den Prüflauf abbricht

Stopprozedur

wird am Ende eines Prüflaufs aufgerufen

check\_Backup

Ist eine Funktion vom Typ Boolean, welche vor dem Aufruf eines Sicherungszugriffes aufgerufen

wird und z.B. die Verfügbarkeit von Medien prüfen kann.

### Gangwechseln

Ist eine Prozedur mit 2 Wertparametern vom Typ Integer. Diese Prozedur wird innerhalb von ADAM aufgerufen, wenn sich Gangspur ändert. Die Parameter werden dabei mit altem Wert und mit neuem Wert belegt. Die Benutzerprozedur wertet die Eingaben aus und bedient dementsprechend den Gangschaltautomaten. Diese Prozedur kann auch per Listenbefehl (GANG x) aufgerufen werden. Die Prozedur sollte die globale Variable "Istgang" setzen.

### userexit1

ist eine Prozedur, welche beim Ausstieg über einen Prüfstandfehler aufgerufen wird. Als Parameter wird die geöffnete Protokolldatei übergeben, in welche zusätzlich zum Standardeintrag beliebiger Text geschrieben werden kann

### userexit2

wie userexit1, wird jedoch nur dann aufgerufen, wenn der Ausstieg über die Liste (z.B. durch ein GOTO ENDE) erfolgte.

## Benutzervariablen

Der der Bibliothek übergebene Pointer "LIPSDaten" zeigt auf folgende Struktur :

TLIPSDaten = Record

SollZyk,	{Sollzyklen des Prüflaufs}
AnzScr,	{Anzahl der Schritte in der Liste}
SKan,	{Sollkanäle aus der PS-Deklaration}
IKan,	{ Istkanäle}
Dkan,	{ Bitmuster }
FKan,	{ Formeln}
IDes,	{ Maskendeskriptoren}
SyncZeiger,	{ Bitmuster für "Syncsignal" }
SyncKanal,	{ Ausgabekanal des Syncsignals }
InBytes,	{ Anzahl der Inputbytes}
OutBytes : Integer;	{ Anzahl der Outputbytes}
InByteD,	{ Deskriptoren der Inputbytes}
OutByteD : Array [1..10] of Bytedesk;	{ Deskriptoren Outputbytes}
sollslot : Array [1..10] of Byte;	{ Sollkanäle für ADAM-Ausgabe }
VArr : Array [0 .. LimCon] of Integer;	{ Systemvariable V00..Vxx}
VArr[39]:	Schrittzähler
VArr[38]:	Zyklenzähler
VArr[37]:	aktuelles Datum im Format Tag x 100 x Monat
VArr[36]:	aktuelle Zeit im Format 100 x Stunde x Minute
SollW : Array [ 1 .. maxsoll] of KanDf;	{ Sollwertdeskriptoren }
IstW : Array [ 1 .. maxist ] of KanDf;	{ Istwertdeskriptoren }
Abbrist,	{ Istwerte beim Abbruch R6}
Istwert: Array [ 1 .. maxist ] of Real;	{ Istwerte R6}
istwert_limit : Array [ 1 .. maxist ] of Boolean;	{ Abbruchflags }

```

DigW  : Array [ 1 .. maxdig ] of DigDf;      { Bitmusterdeskriptoren}
Form  : Array [ 1 .. maxfor ] of Formeldesk; { Formeldeskriptoren }
formel_limit : Array [ 1 .. maxfor ] of Boolean; { Abbruchflags für Formeln }
IntVD  : Array [ 1 .. maxkvar] of IntVarDesk; { Beschreibung interner Variabler in Kanallisten}
Abbrsol,
      { Sollwerte beim Abbruch }
Wert  : Array [ 1 .. maxsoll ] of Integer;   { aktuelle Sollwerte }
SorcusFehlerflag,
      { Merker für Kartenfehler }
Pruefstand_Fehler : Boolean;                { Merker aus Überwachungsprozedur }
HeadLin : String[80];                       { Zeile aus der Steuerliste }
tempf_nr : Longint ;                        { in Meßjobauslagerung benutzte Variable }
tcons   : Real ;                            { Frequenz von F5 }
dtRamp  : Byte;                             { errechnete Rampticks }
OutputBytes : Byte;                         { Anzahl der Outputbytes auf einem Modul M40 }
SORCnt  : Byte;                             { Anzahl der SORCUS-Karten }
SIOCnt  : Byte;                             { Anzahl der SIO-Kanäle }
SIO     : Array[1..4] of Byte;               { Nummern der ersten 4 SIO-Kanäle }
parameter : String;                         { Übergabe für Anwenderprozeduren }
backup_dir : LgLine;                        { Pfad für Sicherungsdateien }
FNum    : Byte;                             { Fehlernummer beim Abbruch }

```

end;