Beispiele zu Assemblerbefehlen der GA 18130

ADRESSE	LABEL	OPERATION		VARIABLE	ab hier Kommentar möglich
0	*	1. LDX			
0		LDX	1	127	Wert 127 → (XR1)
1	LAB1	LDX	L1	128	Wert 128 → (XR1)
3		LDX	2	LAB8	rel. Adresse von LAB8 (= 8 - 4 = 4) → (XR2)
4		LDX	L2	LAB8	abs. Adresse von LAB8 (= 8) → (XR2)
6		LDX	I2	LAB8	(LAB8) → (XR2)
8	LAB8	LDX	3	LAB21-LAB20	rel. Adressendifferenz (= 12-11) → (XR3)
9		LDX	L3	LAB21-LAB20	abs. Adressendiff. (= 21-20) → (XR3)
11	REG1	LDX	L1	*-*	(REG1+1) → (XR1)
⋮		⋮			
	*	2. STX			
20	LAB20	DC		*-*	
21	LAB21	DC		LAB20	
22	LAB22	STX	1	LAB20	(XR1) → (LAB20)
23		STX	I1	LAB21	(XR1) → (LAB20)

Beispiele zu Assemblerbefehlen der GA 18/30

ADRESSE	LABEL	OPERATION		VARIABLE	ab hier Kommentar möglich
25		STX		LAB20	Inhalt des I-Reg. (=26) → (LAB20)
26		:			
.		:			
	*	3. LD (STO analog)			
30		LD		LAB21	Inhalt von LAB21 (=LAB20) → (A)
31		LD	I	LAB21	Inhalt von LAB20 (=26) → (A) vgl. Adresse 25
33		LD	2	0	Inhalt der Adresse, die XR2 angibt → (A)
34		LD	2	1	Inhalt der um 1 erhöhten Adresse, die ... → (A)
35		LD	L2	LAB20	Inhalt der um LAB20 (=20) erhöhten Adresse, ...
37		LD	2	LAB20	Inhalt der um rel. LAB20 (=20-38=-18) ...
38	LAB38	LD	L	LA168	(LA168) → (A); Langwort, weil 168-40 > 127
	*				
	*	4. BSC / BOSC (testen den Akku)			
40		BSC		Z	wenn (Akku) = 0, wird die nächste Adr. übersprungen
41		MDX		LAB38	muß 1-Wort-Befehl sein

Beispiele zu Assemblerbefehlen der GA 18/30

```
ADRESSE | LABEL   OPERATION   VARIABLE      ab hier Kommentar möglich
--------+-------------------------------------------------------------
   42           BSC     Z-+          die nächste Adresse überspringen
   43   LAB43   DC      LAB38
   44
    .
    .
    .           BSC     L   LAB38,+-  springe nach LAB38, wenn (Akku)=0
                .
                .
                .
                BSC     I   LAB43     springe indirekt über LAB43 nach LAB38, hier
                                      bedingungslos
*               .
                .
                .
* BOSC nur Langwort möglich. Wenn Sprung erfolgt, wird Interrupt zurückgesetzt.
* Sonst wie BSC.
*
* 5. MDX
   50           MDX     LAB38        Springe nach LAB38. Rel. Adressierung → +127 / -128 mögl.
* Sprünge > +127 / -128 können indirekt oder lang erfolgen mit bedingungslosem BSC.
```

Beispiele zu Assemblerbefehlen der GA 18/30

```
ADRESSE | LABEL  OPERATION  VARIABLE      ab hier Kommentar möglich
--------+-------------------------------------------------------------
   51           MDX    2  -1          (XR2) um -1 erhöhen.+
   52           MDX    2  LAB43       (XR2) um rel. LAB43 (=43-53=-10) erhöhen.+
        * Anwendung bei Schleifen (hier: 10 mal durchlaufen):
   53           LDX    2  10          Wert 10 → (XR2)
    :   LOOP    ...
    :           :
   57           MDX    2  -1          (XR2) um 1 erniedrigen.+
   58           MDX       LOOP        Springe nach LOOP
   59           MDX    L  REG1+1, 3   (REG1+1) um 3 erhöhen.⊕
   61           MDX    L2 LAB43       (XR2) um abs. Adresse LAB43 (=43) erhöhen.+
   63           MDX    I2 LAB43       (XR2) um Inhalt von LAB43 (=LAB38=38)
    :   *                             erhöhen.+
   65           :
    :           :
  168   LA168   :
```

+ : wird das Index-Reg. 0 oder findet ein Vorzeichenwechsel statt, so wird die nächste Adresse übersprungen. ⊕ sinngemäß für (REG1+1)

I) Interrupt-Verarbeitungs-Routine

Auf jedem Kanal, dessen MK = 1 ist
100 Werte einlesen (versetzt)
Bei IDATE(15) = 1 vorzeitig abbrechen

II) Auswerte-Unterprogramm

Wird vom Hauptprogramm aufgerufen

1) Fehlmessungen aussondern
   Fehlmessungen sind Abweichungen von mehr als (20% = PFEH) vom vorhergehenden Wert, werden durch vorhergehenden Wert ersetzt
   Ist Zahl der Fehlmessungen zu groß,
   → Fehlermeldung

3) Mittelwert u. Standardabweichung berechnen
   Ist Standardabweichung zu groß → Fehlermeldung
   sonst Mittelwert abspeichern

2) Drift berechnen, ist sie größer als zulässig, → Fehlermeldung

4) Fehlermeldung wird bei IDATE(14) = 1 unterdrückt
   Bei IDATE(13) = 1 wird Zahl der Fehlmessungen, Drift und Standardabweichung ausgegeben (auch Mittelwert)

GA 18/30 Information

Hauptprogramm KOREK, M, Mo

Das Programm KOREK dient zum seitenweisen Lesen, Verbessern und Zurückschreiben von Programmen oder Datenfeldern mit Hilfe des Bildschirmgerätes.

Aufruf: $S1 = ... (z.B. $S1 = 05(PROG2))

[$S0 = ...] (normalerweise WS)

$KOREK

Das Programm KOREK fragt, ob nur eingegeben werden soll. Wenn ja (z.B. wenn der betreffende Text erstmalig geschrieben werden soll) ist „JA" einzugeben, wenn nicht, ein oder zwei beliebige andere Buchstaben (z.B. „NO")
Wurde nicht „JA" eingegeben, so werden die ersten 26 Zeilen von S1 nach T5 geschrieben. In diesem Text kann der Benutzer jetzt die gewünschten Verbesserungen vornehmen; dabei werden keine Zeichen zum Rechner übertragen. Soll der Text in den Rechner eingelesen werden, so setzt der Benutzer unter die letzte Zeile in die erste Spalte ein "EOM-Zeichen, das durch einen senkrechten Pfeil dargestellt wird, und betätigt anschließend die Leertaste. Daraufhin wird der Text nach S0 geschrieben und die nächsten 26 Zeilen werden ausgegeben.
Dieser Vorgang wiederholt sich, bis ein END-Befehl eingelesen wird.

Nach der Korrektur den REPLACE-Befehl nicht vergessen!

Vorläufige GA 18/30 Information

Rechnen mit EXTENDED PRECISION

Von Hauptprogrammen mit „Ext. Prec." können nur solche Unterprogramme aufgerufen werden, deren Genauigkeit entweder nicht vereinbart oder ebenfalls „Ext. Prec." ist.
Assembler-Unterprogramme haben keine vereinbarte Genauigkeit, sie können folglich in jedem Fall aufgerufen werden.
Beispiele: ANOUT, ANIN, INTEF, ect.
Alle Fortran-Unterprogramme, gleichgültig ob REAL oder INTEGER, haben in UL eine Genauigkeitsvereinbarung und können nur dann vom Hauptprogramm aufgerufen werden, wenn diese mit der Vereinbarung des Hauptprogramms übereinstimmt.
Die folgenden Bibliotheksprogramme stehen in UL in beiden Genauigkeiten (unter verschiedenen Namen)

alter Name		neuer Name f. SPR	neuer Name für EPR
PØINT	✓	IPØIN	EPOIN
STAXY	✓	ISTAX	ESTAX
KØØRØ	✓	IKØØR	EKØØR
FDET	✓	FDET	EDET
IWNDL		IWNDL	EWNDL
DETER	✓	DETER	EDETR
DETS	✓	DETS	EDETS
MINFE	✓	MINFE	EMNFE
FUNCT	✓	FUNCT	EFUNC
FLNET	✓	FLNET	ELNET
IVAR	✓	IVAR	EVAR
KREUZ	✓	KREUZ	EKREU

Stufen

Overlaystrukturen

Ist ein Programm zu groß für den Kernspeicher, so kann der Benutzer durch Verwendung einer Overlaystruktur den Kernspeicher mehrfach ausnutzen.
Es stehen 2 Möglichkeiten zur Verfügung:

1. Unterprogramm-Overlay
   Es besteht ein Hauptprogramm und eine Anzahl von Overlay-Programmen, die wie Unterprogramme behandelt werden
2. Ketten-Overlay
   Das Programm wird in gleichberechtigte Teilprogramme zerlegt, die sich nacheinander aufrufen.

Aufruf des Programmsystems

→ $ØVLAY
Ausgegeben wird: GEBE FOLGENDERMASSEN AN
$HAUPT
$UP1

$EOD

→ $PRØG
?ØP1
?ØP2
$EOD

mit

PRØG : Name des Hauptprogramms bzw. des 1. Teilprogramms
ØP1 : Name des 1. Unterprogramms bzw. des 2. Teilprogramms

Beim <u>Unterprogramm-Overlay</u> wird nach dem Befehl
$ EJOB das Hauptprogramm geladen und abgearbeitet,
bis mit dem Aufruf

CALL ØVER(I)

das I-te Unterprogramm angesprungen wird. Das
Unterprogramm wird abgearbeitet und nach Erreichen des
Befehls STØP erfolgt ein Rücksprung in das
Hauptprogramm. Der Datenverkehr geht über den
unbenannten COMMON-Block, der bei jedem Programm
gleich aufgebaut sein muß.
Die Unterprogramme müssen wie Hauptprogramme erstellt
werden und in DC stehen.

Beim <u>Ketten-Overlay</u> steht am Ende des N-ten Teilprogrammes
(mit Ausnahme des letzten) die folgende Befehlsfolge:

CALL ØVER(N)
STØP
END

Dabei ist N die Nummer des rufenden Teilprogrammes. Mit
CALL ØVER(N) wird das nächste Teilprogramm aufgerufen.
Am Ende des letzten Teilprogrammes steht wie üblich
STØP und END.
Der Datenverkehr geht auch hier über den unbenannten COMMON-
Block, der bei jedem Teilprogramm gleich aufgebaut sein muß.
Alle Teilprogramme müssen als unabhängige Hauptprogramme erstellt
werden und in DC stehen

Unterprogramm DATER

Das Unterprogramm DATER für die schnelle Datenerfassung steht in UL.

Zweck:

Das Unterprogramm DATER dient zur schnellen Datenerfassung. Es lassen sich mit seiner Hilfe bis zu 16 Programmblöcke zur Erfassung, Auswertung und Ausgabe von Daten unterbrechungsgesteuert aufrufen.
Der unterbrechungsgesteuerte Aufruf ist aber nur dann wirksam, wenn der zuletzt aufgerufene Programmblock vollständig abgearbeitet ist und sich der Rechner in einer Warteschleife befindet. Dem Benutzer wird dies durch ein Fertigzeichen angezeigt.

Aufbau des Hauptprogramms und Aufruf (Beispiel):

```
      I=5
      CALL DATER(I)
      GOTO (1,2,3,4,5),I
1     .
      .
      .                  } Einlesen von Daten
      CALL BOSC
2     .
      .                  } Auswerten
      .
      CALL BOSC
3     .
      .                  } Schreibprogramm
      .
      CALL BOSC
4     .                  } Zeichenprogramm
      .
      CALL BOSC
5     STOP
      END
```

GA 16/30 Information

Datenerfassung

Die Datenerfassungsprogramme arbeiten mit einem Unterbrechungssystem, das die Möglichkeit schafft, bei einer vom Benutzer erzeugten Unterbrechung in bestimmte Programmblöcke (max 16) zu springen. Die Programmblöcke werden auf einer Programmkonsole am Prüfstand angewählt. Gleichzeitig befindet sich auch der Fernschreiber zur Bedienung des Rechners am Prüfstand. Die Meßsignale werden über ~~Spezial~~Verstärker mit begrenzter Ausgangsspannung auf die Analog-Digital-Wandler geleitet.

Der Benutzer hat die Wahl zwischen einer schnellen Datenerfassung mit geringer Speichermöglichkeit im Kernspeicher und einer etwas langsameren Datenerfassung mit mehrfacher Ausnutzung des Kernspeichers.

Dabei ist I die Anzahl der Programmblöcke, die angesprungen werden können. Der Rücksprung zu OATER erfolgt über den Aufruf CALL BOSC. Der Benutzer wählt den anzuspringenden Programmblock mit den Datenschaltern der Console an. Wird ein Programmblock angewählt, dessen Nummer höher als I ist, so erfolgt eine Fehlermeldung.
Erfolgt eine Unterbrechung, wenn kein Datenschalter gesetzt ist, so wird ein „Console Interrupt" ausgelöst und das ganze Programm beendet. Diese Unterbrechung funktioniert in jedem Fall.
Bei der schnellen Datenerfassung sollte der Benutzer versuchen, das Programm möglichst kurz zu halten, um im Kernspeicher noch ausreichend Platz für Daten zu lassen.

Programm INTIN

Das Programm INTIN für die Datenerfassung mit mehrfacher Kernspeicherausnutzung steht in OC.

Zweck

Das Programm INTIN dient zur Datenerfassung mit mehrfacher Kernspeicherausnutzung. Bei dieser Art Datenerfassung ist es möglich, das Hauptprogramm jederzeit zu unterbrechen und in eins der max. 16 Interrupt-Programme zu springen. Während der Bearbeitung eines Interrupt-Programmes ~~jedoch~~ ist keine weitere Unterbrechung außer des „Console Interrupts" mehr möglich.
Zu jedem Zeitpunkt steht entweder nur das Hauptprogramm oder eins der Interrupt-Programme im Kernspeicher.

Aufbau des Programmsystems

Das Hauptprogramm und alle Interrupt-Programme müssen als selbstständige Programme geschrieben sein und in ~~kompilierter Form~~ in OC zur Verfügung stehen. Soll an einer bestimmten Stelle des Hauptprogrammes ein Sprung in ein Interrupt-Programm erfolgen bzw. spätestens erfolgt sein, so muß sich an dieser Stelle eine Warteschleife befinden. Der Datenverkehr zwischen Haupt- und Interrupt Programmen erfolgt über den unbenannten Common-Block, der deshalb in allen Programmen gleich aufgebaut sein muß.

Aufruf des Programmsystems:

Der Aufruf erfolgt über den Fernschreiber oder ein peripheres Gerät.

→ $INTIN

Ausgegeben wird: GEBE FOLGENDERMASSEN AN

$HAUPT
$UP1

$EOD

→ $HAUPT
$IP1
$IP2
:
$EOD

mit

HAUPT : Name des Hauptprogramms
IP1 : Name des 1. Interrupt-Programms

Nach der Eingabe des Befehls „$EOD" wird das Hauptprogramm geladen und die Unterbrechung ermöglicht.
Werden 16 Interrupt-Programme angegeben, erfolgt eine Meldung und ein selbsttätiger Sprung zu „EOD".
Sind z.B. nur 6 Interrupt-Programme vereinbart und versucht der Benutzer das 7. IP zu aktivieren, so erfolgt eine Fehlermeldung und ein Rücksprung ins Hauptprogramm.

Hauptprogramm BODE

Das Programm befindet sich in DC und wird mit

[$LO=TY wenn Listenausdruck auf TY erwünscht]

$B

aufgerufen.

Zweck:

Mit den vom Ortskurvenrechner erstellten Daten des Frequenzgangs wird das Bodediagramm mit Amplituden- und Phasengang gezeichnet.

Ablauf

Das Programm arbeitet im Dialog mit dem Benutzer, der an 3 Stellen im Programm Steuerbefehle eingibt:

1) Eingabe:
   WFAK, LESE, WEITER, PLATTE
2) Modifikation
   RECHNE, ZEIGE, LISTE, WEITER
   [Modifikationsbefehl]
3) Ausgabe
   ECKO, ECKU, NETZ, DRUCKE, ZEICHNE,
   MODI, RECHNE, START, ENDE

Steuerbefehle

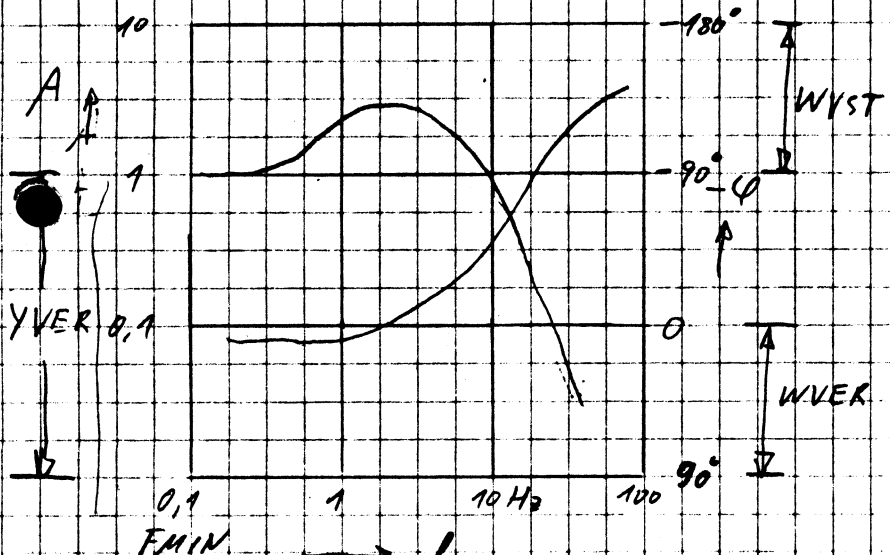
WFAK=I	Kennzahl zur Winkelkorrektur. Bei invertierten Ortskurven WFAK=2 eingeben; bei Winkeln kleiner als $-360°$ WFAK=3; ansonsten Befehl weglassen. Kennzahl muß ggf. vor jedem LESE- oder PLATTE-Befehl eingegeben werden.
LESE:	Die Frequenzgangdaten werden von SI [=PR] eingelesen. Steht der Frequenzgang auf mehreren Datenstreifen, ist wiederholt LESE einzugeben
PLATTE:	Früher schon von SI eingelesene Daten werden von IS [=WS] eingelesen
WEITER:	Das Programm geht zum nächsten Teil über
RECHNE:	Rücksprung in die Kennwerte-Berechnung.
LISTE:	Die Liste der Kennwerte wird nach LO [=TY] geschrieben
ZEIGE:	Die Liste der Kennwerte wird nach OM [=TS] geschrieben
ECKO	Schreiber fährt obere rechte Ecke im Diagramm an
ECKU	Schreiber fährt untere linke Ecke an
NETZ	Ausgabe des logarithmischen Netzes
DRUCKE	Frequenzgangdaten mit Winkel nach LO [=TY] schreiben
ZEICHNE	Amplitudengang und Phasengang zeichnen
MODI	Rücksprung zur Modifikation der Kennwerte

START Rücksprung zum Einlesen

ENDE Programm beenden

Modifikation der Kennwerte

Beispiel für Kennwerte



IDEZ = 3 DEK

YVER = 2 DEK

FMIN = 0,1 Hz

WVST = 90 °/DEK

WVER = 90 °

AFMI = 1.

AFAK = 1.

FFAK = 1.

YVER : Verschiebung der Amplitudenachse in Dekadenrastern

IDEZ : Anzahl der Dekadenraster

FMIN : Minimalfrequenz im Diagramm

WVST : Winkelverstärkung in Grad / Dekadenraster

WVER : Verschiebung der Winkelachse in Grad

AFMI : Bezugsamplitude für Amplitudengang,
(= Amplitude bei kleinster Frequenz)

AFAK : Amplitudenfaktor, verschiebt Amplitudengang vertikal

FFAK : Frequenzfaktor, verschiebt Bodediagramm horizontal

Modifikationsbefehl im Format A4, 1X, F10.4 2B

IDEZ=2.

Wird nach der Modifikation in die Kennwert-Berechnung

b.w.

zurückgesprungen, so erfolgt die Neuberechnung in der folgenden Reihenfolge:

$$\left\{\begin{matrix} AFAK \\ FFAK \end{matrix}\right\} \Rightarrow \left\{\begin{matrix} IOEZ \\ YVER \\ FMIN \end{matrix}\right\} \Rightarrow \left\{\begin{matrix} WVST \\ WVER \end{matrix}\right\}$$

Datenlochstreifen

Auf dem Datenlochstreifen müssen Frequenz, Realteil und Imaginärteil im Format 3(E10.4, 4X) stehen. Die letzte Datengruppe muß eine Frequenz größer als 10^{10} enthalten, dies gilt als „END"

Quadratisches Netz

Nach der Modifikation wird vom Schreiber der Mittelpunkt des log. Netzes angefahren, der Benutzer sollte den Stift auf die Mitte des Blattes einstellen, dann den Befehl ECKO geben um die Verstärkung des Schreibers einzustellen. (ECKU zur Kontrolle.)

Unterteilung der Dekaden des log. Netzes

X-Achse (Frequenz) : 1, 2, 5, 10

Y-Achse (Amplitude) : 1, 2, 4, 6, 8, 10

Anschluß des X-Y-Schreibers

Kanal 0 : X-Achse (Frequenz)

Kanal 1 : Y-Achse (Amplitude und Winkel)

Kanal 7 : Pen Lift

```
      1          WRITE(7,100)
      2  100     FORMAT('GEBE EXPONENT')
      3          READ(7,200)EXP
      4  200     FORMAT(F10.3)
      5          IF(IDATS(1))1,1,9
      6  1       CALL ISTAX(2000,2000)
      7          CALL ISTAX(-2000,-2000)
      8          READ(7,300)IA
      9  300     FORMAT(I1)
     10          IF(IA)2,2,1
     11  2       CALL IKOOR(4,6,5)
     12  9       READ(1,400)A,B
     13  400     FORMAT(2F10.5)
     14          IF(A)8,10,8
     15  8       BA=B**EXP
     16          IA=IFIX(2000.*A)
     17          IB=IFIX(2000.*BA)
     18          CALL IPOIN(IA,IB)
     19          GOTO9
     20  10      STOP
     21          END
```

GA 18/30 Information

LIST SOURCE IM DIN A 4 FORMAT

Dieses Programm bietet die Möglichkeit, ein vereinbartes Source-Programm im DIN A 4 Format normalerweise auf TTY ausdrucken zu lassen. Die einzelnen Seiten werden fortlaufend numeriert und können weiterhin mit einem Titel versehen werden. Ebenfalls werden die Zeilen fortlaufend numeriert.

Dieses LIST-Programm ist im DC-File und hat den Namen "LIST", so daß es mit

```
/$  LIST
```

aufgerufen werden muß.

Das Programm LIST benötigt folgende logische Einheiten

```
SI  =  Source Input
LO  =  List Output in DIN A 4
OM  =  Operrator message
CC  =  Control commands
```

z. B.

```
/$  JOB                OM, CC, LO = TY
/$  SI  =  DS (NAME)
/$  LIST
```

Nach Vereinbarung des "SI" und Aufruf des Programms meldet es sich über "OM" mit dem Ausdruck

"GEBE PROGRAMMNAME MAX 40 ZEICHEN!"

Die nun über "CC" einzugebenden Zeichen erscheinen beim Ausdruck zu Beginn jeder Seite.

Nach Eingabe dieser Zeichen meldet sich das Programm über "OM" mit

"GEBE STARTZEILE! FORMAT I5"

Über "CC" ist nun eine 5-stellige Integerzahl einschließlich Vorzeichen einzugeben, die der gewünschten Startzeile entspricht.

Das Programm beginnt dann mit dem Ausdruck der Seite, auf der die angegebene Zeile steht (auf einer Seite stehen 60 Zeilen).

Gibt man eine positive Zahl an, so erfolgt eine Numerierung der Zeilen. Bei Eingabe einer negativen Zahl ($\neq 0$) wird die Numerierung fortgelassen.

Ist die angegebene Zeilenzahl größer als die des Programms, so erfolgt die Fehlermeldung:

"ZEILENZAHL ZU GROSS".

KOPF Programm zum Kommentieren von ASSEMBLER-Programmen

Das Programm bietet die Möglichkeit, ein durch SI vorhandenes Source-programm mit einem einheitlichen Kopf zu versehen, Befehle, Kommentare anzufügen und Texte einzuschieben.

Das Kopf-Programm ist im DO-File und hat den Namen "KOPF", so daß es mit

$ KOPF

aufgerufen werden muß.

Das Programm KOPF benötigt folgende logische Einheiten

SI = Source Input
SO = Source Output
LO = List Output

z. B.

```
$ JOB                    (CH-CO-[illegible])
$ SI = DO (NAME)
$ SO = KS
$ KOPF
```

Nach Aufruf des Programms wartet das Programm auf folgende Kontrollbefehle:

1) K = Y   Durch diesen Kontrollbefehl wird der Kopf erstellt. Beendet man bei der Erstellung des Kopfes eine Textzeile mit einem ¥, so [illegible] das KOPF-Programm [illegible] bis zur nächsten Zeile warten. Schreibt man [illegible] am Zeilenende ein @, so wird eine weitere Zeile zur Eingabe eines Textes zur Verfügung gestellt, d. h. bei einem langen Text, der über mehrere Zeilen geht, braucht man nur jeweils die Zeile mit einem @ zu beenden, um eine weitere Zeile eingeben zu können.

a) C = W Durch diesen Kontrollbefehl wird der durch ST [illegible] eingelesene Sourceprogramm nicht [illegible] [illegible] erfolgt ein Übergang [illegible] BeOS [illegible].

b) C = F Durch diesen Kontrollbefehl wird das durch ST [illegible] Sourceprogramm [illegible] [illegible] SD [illegible] [illegible] [illegible]

c) [illegible] Durch diesen Kontrollbefehl [illegible] [illegible] Sourceprogramm [illegible] SD [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]

d) a Durch dieses Zeichen wird die [illegible] bis zum CR auf TTY [illegible] die gesamte Zeile nach SD [illegible].

e) @ Durch dieses Zeichen können zusätzliche [illegible] in das Sourceprogramm eingeschoben werden.

Nach Eingabe dieses Steuerzeichens wird [illegible] auf TTY ausgegebene Zeile [illegible] durch [illegible] gelöscht und danach die nächste Zeile [illegible] eröffnet. Wird die jetzt eingegebene [illegible] mit einem @ beendet, so wird eine [illegible] zwischengeschoben, welche [illegible] [illegible]. Ist die letzte [illegible] auf diese mit a beendet werden. [illegible] Zeile [illegible] [illegible] und die [illegible] Zeile wieder ausgegeben. [illegible] das Programm auf [illegible]

a) Kommentar CR

Durch diesen Steuerzeichen wird der alte Kommentar der Zeile gelöscht und der zuvor eingegebene Kommentar hinter den ASSEMBLER-Befehl gesetzt. Danach wird diese neue Zeile nach SO kopiert.

Bei falschen Steuerzahlen, z. B.:

- nicht Dezimalziffern
- $0 \leq i, n \leq$ 9999
- $n < n$ ($n = n$ kann eine Zeile kommentiert werden)
- $n_2 < n_1$ (zweite Anfangsadresse kleiner als erste Endadresse)

erscheint Fehlermeldung ERROR auf SFW. Es kann ein neuer Steuerbefehl eingegeben werden.

[illegible]-Programm zum Erstellen von verkürzten [illegible]-Programmen

Das Programm bietet die Möglichkeit, [illegible]

[illegible]

[illegible]

Dieses Programm ist [illegible] "KURZ" [illegible]

[illegible]

$ KURZ

aufgerufen werden kann.

Das Programm KURZ benötigt folgende logische Einheiten.

SI = Source Input
SO = Source Output

z. B.

$ JOB
$ SI = DS (Name)
$ SO = PP
$ KURZ

GA 18/30 - Information

Kopplung GA 18/30 - EAI 580

Werden beide Rechner zusammen als Hybridsystem benutzt, so müssen am Analogrechner die folgenden Tasten des Bedienungsfeldes gedrückt sein.

1) beide	PMT	- Tasten	(Betriebsarten)
2)	DIG	- Taste	(Komponenten)
3) alle vier	0	- Tasten	(RDAC)

Der Analogrechner wird jedesmal, wenn ein Rückkopplung in den Normbetrieb erfolgt, auf seinen Anfangszustand zurückgesetzt. Dieser Anfangszustand ist folgendermaßen festgelegt:

1. Analoge Betriebsart	OP
2. logische Betriebsart	SET
3. CLOCK	10^6
4. TIMER	1 sec
5. Komponenten	keine

Bei jedem Übergang in den Anfangszustand wird die Logik gelöscht.
Z. Z. stehen folgende Unterprogramme zur Verfügung:

1) SETP (IWERT, IHR)
zum Setzen der Servopotentiometer

2) DVM (IWERT)
zum Auslesen des Digitalvoltmeters

3) INTEF (PARAM)
zur Steuerung des Analogrechners
(siehe Programmbeschreibung)

PR-2

Hauptprogramm BODE

Das Programm befindet sich im DC und wird mit Ø BODE aufgerufen. Es arbeitet im Dialog mit dem Benutzer.

Zweck:

Die von dem Ortskurvenrechner erstellten Lochstreifen mit den Daten von Frequenz,Realteil und Imaginärteil werden gelesen. Für jeden Punkt werden daraus normierte Frequenz,normierte Amplitude und Phasenwinkel berechnet.Es wird ein zu diesen Werten passendes logarithmisches Netz gezeichnet und in dieses Netz das Bode-Diagramm mit Amplitude und Phasengang.

Ablauf des Dialogs

Benutzer : 1.Lochstreifen mit Daten im FORMAT(3(E 10.4,4X)) in FR einlegen,BODE aufrufen
Programm : Fordert Faktor FF an,durch den die Frequenz geteilt werden soll
Benutzer : Gibt Frequenzfaktor (im F 10.) ein
Programm : liest ersten Lochstreifen,fordert Steuerzeichen an
Ben. : wenn weitere Lochstreifen vorhanden:nächsten einlegen, Steuerzeichen 1
wenn Einlesen beendet : Steuerzeichen 0
Pr. : berechnet die folgenden Größen:
IZ : Anzahl der Dekaden des log. Netzes
AM : Amplitude bei kleinster Frequenz (f. Normierung)
VVER: Verschiebung der Achse A=1 in Dekadenrastern
FMIN: kleinste Frequenz des Rasters
NV : Anzahl der Dekadenraster für $\Delta\varphi = 2\pi$
VVPH: Verschiebung der Achse $\varphi=0$ in Dekadenrastern
Diese Werte werden ausgegeben,Steuerzeichen [illegible] neuer Wert [illegible] aufgerufen.

Bem. : Wenn der Benutzer von den eingegebenen Größen eine ändern will, so gibt er die Nr. dieser Größe und ihren neuen Wert ein (FORMAT (I1,F10.0))

Fr. : Kontrollausgabe der gelesenen Größe

Bem. : Wenn keine weitere Veränderung gewünscht, "0" eingeben.

Fr. : Soll auf die Datenstation ausgegeben werden

Bem. : Soll Lochen für Ausgabe

0 ⟹ Worte auszeichnen

1 ⟹ Worte ausdrucken

2 ⟹ beide zeichnen

Eingabefehler berücksichtigen

Fr. : Wenn ein Satz verlangt ist, werden 2 gegenüberliegende Zeilen des Satzes mit dem X-Y Schreiber angezeichnet Auszeichnen bzw. Ausdrucken der Worte des Satz-Blocks.

Anmerkungen zum Datenlochstreifen:

1) Der Datenlochstreifen muß mit einem Programmwort anfangen
2) Die letzte Datengruppe muß eine Frequenz größer als 10^{10} enthalten, dies gilt als "END"
3) Bei Meßpunkten mit sehr niedriger Frequenz ist darauf zu achten, daß der Phasenwinkel nicht aufgrund von Meßfehlern positiv wird (= Imaginärteil wird pos.)

Anschluß des X-Y-Schreibers

Kanal 0 : X-Achse : Frequenz
Kanal 1 : Y-Achse : Amplitude, Winkel
Kanal 7 : Pen Lift

Unterteilung der Sekunden beim log. Netz

X-Achse (Frequenz) : 1,2,5,10
Y-Achse (Ampl., Winkel): 1,2,4,6,8,10

Benötigte Unterprogramme : WERT u. ZEIT

Programm erstellt von : A. Schulz

Bei inversen Ortskurven WVER selbst berechnen:

$$WVER = \frac{-WV}{2} - \frac{I^2}{2}$$

FFR-13

GA 18/30 - Information

Darstellung alphanumerischer Zeichen

Bei der GA 18/30 können alphanumerische Zeichen auf drei Arten ein- bzw. ausgegeben werden:

1. H-FORMAT
2. Daten zwischen zwei Apostrophs
3. A-FORMAT

Die ersten beiden Möglichkeiten eignen sich besonders für alphanumerische Zeichen, welche nicht durch das Programm geändert werden sollen. Dagegen können die im A-Format eingelesenen Daten durch das Hauptprogramm geändert werden.

H-FORMAT

Der Angabe nH innerhalb eines FORMAT-statements folgen n alphanumerische Zeichen. Blank gilt als Zeichen und ist mitzuzählen.

Beispiel:

```
      READ (7,1)
    1 FORMAT (23HALPHANUMERISCHE ZEICHEN )
```

oder

```
      WRITE (7,2)
    2 FORMAT (13HUNTERSCHRIFT )
```

Daten zwischen zwei Apostrophs

Ähnlich dem H-FORMAT können auch alphanumerische Zeichen eingegeben werden, wenn sie im FORMAT-statement zwischen zwei Apostrophs gestellt werden.

Beispiel:
```
      READ (7,1)
    1 FORMAT ( 'ALPHANUMERISCHE ZEICHEN')
```

oder
```
      WRITE (7,2)
    2 FORMAT ( 'UEBERSCHRIFT' )
```

A - FORMAT

Das A-FORMAT wird gebraucht, um alphanumerische Zeichen in einer Variablen im Kernspeicher abzubilden oder aus dieser Variablen auszulesen.

Jede INTEGER - Variable kann maximal 2 Zeichen aufnehmen und es wird linksbündig im Kernspeicherwort geschrieben. Der nicht benötigte Platz im Kernspeicherwort wird durch ein Blank ergänzt.

Ist die Anzahl der gelesenen Zeichen kleiner als das vorher im Programm vereinbarte Feld, so wird der nicht benötigte Platz durch Blanks automatisch aufgefüllt. Ist die Anzahl jedoch größer, wird der Rest nicht in das Feld gelesen und abgeschnitten.

Bei der Ausgabe spielt die Feldgröße keine wesentliche Rolle, da beim Einlesen das gesamte Feld mit Blanks vollgeschrieben wurde.

Beispiel:
```
      DIMENSION IFELD(15)
      READ (7,1) (IFELD(K), K=1,15)
    1 FORMAT(15A2)
```

als Eingabe wird folgender Text geschrieben:
'ALPHANUMERISCHE ZEICHEN'

Im Feld IFELD wird diese Eingabe wie folgt abgebildet:

AL	PH	AN	UM	ER	IS
IFELD(1)	IFELD(2)	IFELD(3)	IFELD(4)	IFELD(5)	IFELD(6)

CR	E␣	ZS	IC	HE	N␣
IFELD(7)	IFELD(8)	IFELD(9)	IFELD(10)	IFELD(11)	IFELD(12)

␣␣	␣␣	␣␣
IFELD(13)	IFELD(14)	IFELD(15)

oder

```
      DIMENSION IFELD(10)
      READ (7,2) (IFELD(K),K=1,10)
    1 FORMAT (10A1)
```

als Eingabe wird folgender Text geschrieben:
ALPHANUMERISCHE ZEICHEN

Im Feld IFELD wird dieser Text wie folgt abgebildet:

A␣	L␣	P␣	H␣	A␣
IFELD(1)	IFELD(2)	IFELD(3)	IFELD(4)	IFELD(5)

N␣	U␣	M␣	E␣	R␣
IFELD(6)	IFELD(7)	IFELD(8)	IFELD(9)	IFELD(10)

der Rest des Eingabetextes wird nicht berücksichtigt und abgeschnitten.

Die Darstellung mit Hexadezimalzahlen im Kernspeicher beim Einlesen im FORMAT A1 ist der folgenden Tabelle zu entnehmen. Gleichzeitig ist dort auch die entsprechende Integerzahl angegeben, die sich aus dem Bitmuster des Kernspeicherwortes ergibt.

Alphanumerisches Zeichen	Darstellung im Kernspeicher durch eine Hexadezimalzahl	Integerzahl
A	C14Ø	-16064
B	C24Ø	-15808
C	C34Ø	-15552
D	C44Ø	-15296
E	C54Ø	-15040
F	C64Ø	-14784
G	C74Ø	-14528
H	C84Ø	-14272
I	C94Ø	-14016
J	D14Ø	-11968
K	D24Ø	-11712
L	D34Ø	-11456
M	D44Ø	-11200
N	D54Ø	-10944
O	D64Ø	-10688
P	D74Ø	-10432
Q	D84Ø	-10176
R	D94Ø	- 9920
S	E24Ø	- 7616
T	E34Ø	- 7360
U	E44Ø	- 7104
V	E54Ø	- 6848
W	E64Ø	- 6592
X	E74Ø	- 6336
Y	E84Ø	- 6080
Z	E94Ø	- 5824
Ø	F Ø4Ø	- 4032
1	F14Ø	- 3776
2	F24Ø	- 3520
3	F34Ø	- 3264
4	F44Ø	- 3008
5	F54Ø	- 2752
6	F64Ø	- 2496
7	F74Ø	- 2240
8	F84Ø	- 1984
9	F94Ø	- 1728

Alphanumerisches Zeichen	Darstellung im Rechner durch eine Hexadezimalzahl	Integerzahl
blank	4Ø4Ø	16448
.	4B4Ø	19264
<	4C4Ø	19520
(	4D4Ø	19776
+	4E4Ø	20032
&	5Ø4Ø	20544
!	5A4Ø	23104
$	5B4Ø	23360
*	5C4Ø	23616
)	5D4Ø	23872
;	5E4Ø	24128
-	6Ø4Ø	24640
/	614Ø	24896
,	6B4Ø	27456
%	6C4Ø	27712
>	6E4Ø	28224
?	6F4Ø	28480
:	7A4Ø	31296
#	7B4Ø	31552
@	7C4Ø	31808
'	7D4Ø	32064
=	7E4Ø	32320
"	7F4Ø	32576

<u>Beispiel:</u> Darstellung einer im Feld IFELD(2) eingelesenen Variablen durch die entsprechenden Hexadezimalzahlen

1. Im FORMAT (2A1) wird AB eingelesen

C14Ø IFELD(1)
C24Ø IFELD(2)

2. Im FORMAT (2A2) wird AB eingelesen

C1C2 IFELD(1)
4Ø4Ø IFELD(2)

Benennung von FØRTRAN-Unterprogrammen

FØRTRAN-Real-Unterprogramme mit einfacher Genauigkeit können von Hauptprogrammen mit doppelter Genauigkeit nicht aufgerufen werden. Deshalb sollte die Art des Unterprogrammes aus dem Namen ersichtlich sein. Der erste Buchstabe des Unterprogrammes sollte lauten:

Bei Integer-Unterprogrammen	: I
Bei Real-Unterpr. mit einfacher Genauigkeit	: F
Bei Real-Unterpr. mit doppelter Genauigkeit	: E

Soll ein FØRTRAN-Real-Unterprogramm mit einfacher GenauigkeitF (F...) für Rechnungen mit doppelter Genauigkeit benutzt werden, so sollte aus Gründen einer besseren Übersichtlichkeit der Unterprogrammname so geändert werden, daß er mit E beginnt und es muß vor dem Programm der Zusatz *EXTENDED PRECISION gesetzt werden.
Integer-Unterprogramme brauchen nicht geändert zu werden, sofern der Name mit einem I beginnt.
Beim Aufruf muß der volle Name (mit I,F oder E) verwandt werden.
Der Grund für diese Benennungsbedingung liegt darin, daß der FØRTRAN-Compiler bereits die Platzreservierung für Variable und Konstante vornimmt, wobei für INTEGER-Größen 1 Platz, für REAL-Größen mit einfacher Genauigkeit 2 Plätze und für REAL- Größen mit doppelter Genauigkeit 3 Plätze bereitgestellt werden.

USER LIBRARY (UL)

Alle nicht vom System benötigten Unterprogramme befinden sich in der UL und sind unter ihrem Namen abgelegt.

Die einzelnen Unterprogramme sind entweder:

1) In einem FORTRAN-Programm durch

```
CALL        Name (P1, P2, ....., Pn)
```

2) in einem ASSEMBLER-Programm durch

```
CALL        Name
DC          Adresse von P1
DC          Adresse von P2
DC          "
DC          "
DC          "
DC          Adresse von Pn
```

aufrufbar, mit P1, P2, ..., Pn als Parameter

Zu beachten ist, daß zum Aufruf des "CIC" (CORE - IMAGE - CONVERTER) folgende Steuerbefehle notwendig sind.

```
CIC
MAP *)
BUILD, UL
```

*) Dieser Steuerbefehl ist nur erforderlich, wenn CORE - MAP erwünscht.

GA 18/30 - Information

Unterprogramm ANOUT

Dieses Unterprogramm befindet sich in der UL. Es ist zuranalogen Ausgabe von digitalen Werten geeignet.

Die digitalen Werte müssen Integergrößen im Bereich von ± 2ØØØ sein. Die zugehörige Ausgangsspannung ist ± 5 V.

Das Unterprogramm wird:

1) Im FORTRAN-Programm durch

```
CALL    ANOUT  (IVAR, INR)
```

2) im ASSEMBLER-Programm durch

```
CALL        ANOUT
DC          Adresse von IVAR
DC          Adresse von INR
```

aufgerufen.

IVAR - auszugebende INTEGER-Variable
INR - analoger Ausgabekanal 0 7

IVAR muß, wenn erforderlich, im Hauptprogramm so normiert werden, daß IVAR maximal 2ØØØ wird. (siehe Beispiel)

Beispiel zum Unterprogramm ANOUT

```
      .
      .
      .
      .
      .
      .
      X = XØ * SIN (OMEGA * T    -    PHI)
      Y = YØ * COS (OMEGA * T    -    PHI)
     IX = IFIX  (X/XØ  *  2ØØØ.)
     IY = IFIX  (Y/YØ  *  2ØØØ.)
      CALL ANOUT (IX, Ø)
      CALL ANOUT (IY, 1)
      .
      .
      .
      .
      STOP
      END
```

Das ASSEMBLER-Programm dieses Unterprogramms ist auf Seite UL - 3 angegeben.

```
 1    ØØØØ 1  Ø1556923          ENT     ANOUT
 2    ØØØØ Ø  ØØØØ       ANOUT  DC      Ø
 3    ØØØ1 Ø  Ø82Ø               XIO     MASK
 4    ØØØ2 Ø  6B23               STX   3 REG3
 5    ØØØ3 Ø1 678ØØØØØ           LDX   13 ANOUT
 6    ØØØ5 ØØ C78ØØØ01           LD    I3 1
 7    ØØØ7 Ø  DØ1F               STO     OH
 8    ØØØ8 ØØ C78ØØØØØ           LD    I3 Ø
 9    ØØØA Ø  DØ1D               STO     BO
10    ØØØB Ø1 ØC0000001A         XIO   L DISSE
11    ØØØD Ø1 ØC0000001C         XIO   L DISOH
12    ØØØF Ø1 ØC0000001E         XIO   L BOUT
13    ØØ11 Ø1 ØC0000002Ø         XIO   L OHACA
14    ØØ13 Ø  Ø81Ø               XIO     DMASK
15    ØØ14 Ø1 678ØØØ26           LDX   I3 REG3
16    ØØ16 Ø1 74Ø2ØØØØ           MDX   L ANOUT,2
17    ØØ18 Ø1 4C8ØØØØØ           BSC   I ANOUT
18    ØØ1A    ØØØØ               BSS   E
19    ØØ1A Ø  ØØØØ       DISSE  DC      Ø
20    ØØ1B Ø  DC01               DC      /DC01
21    ØØ1C Ø  ØØØØ       DISOH  DC      Ø
22    ØØ1D Ø  DC02               DC      /DC02
23    ØØ1E 1  ØØ28       BOUT   DC      BO
24    ØØ1F Ø  D9ØØ               DC      /D9ØØ
25    ØØ2Ø 1  ØØ27       OHACA  DC      OH
26    ØØ21 Ø  D9Ø2               DC      /D9Ø2
27    ØØ22 Ø  FFFF       MASK   DC      /FFFF
28    ØØ23 Ø  Ø48Ø               DC      /Ø48Ø
29    ØØ24 Ø  ØØØØ       DMASK  DC      /Ø
30    ØØ25 Ø  Ø48Ø               DC      /Ø48Ø
31    ØØ26 Ø  ØØØØ       REG3   DC      Ø
32    ØØ27 Ø  ØØØØ       OH     DC      Ø
33    ØØ28 Ø  ØØØØ       BO     DC      Ø
34    ØØ2A    ØØØØ               END
```

GA 18/30 INFORMATION

Unterprogramm ANIN

Dieses Unterprogramm befindet sich in der UL. Es ist zum Einlesen von analogen Werten geeignet.

Die analogen Werte müssen im Bereich von ± 5V sein. Die zugehörigen digitalen Werte sind Integergrößen im Bereich von ± 2ØØØ.

Das Unterprogramm wird:

1) Im FORTRAN-Programm durch

```
CALL ANIN (IWERT, INR)
```

2) Im ASSEMBLER-Programm durch

```
CALL ANIN
DC   ADRESSE von IWERT
DC   ADRESSE von INR
```

aufgerufen.

IWERT = INTEGER-Größe der Eingangsspannung
INR = analoger Eingabekanal 0 15

Bemerkung: Den Kanälen 0 - 3 sind Sample-Hold-Verstärker vorgeschaltet, so daß diese Kanäle echt simultan eingelesen werden können. Zu diesem Zweck muß, um alle vier SH Verstärker auf Hold zu setzen, beim Einlesen dieser Kanäle zu INR 2048 addiert werden. Sobald irgendein Kanal unter seiner normalen NR eingelesen wird, gehen die SH-Verstärker in Sample-Mode.

Neu: POINT, mit Testprogramm TPOIN

GA 12/30 INFORMATION

Unterprogramm ~~POINT~~ bzw. TPOIN

Dieses Unterprogramm befindet sich in der UL. Es ist zur analogen Ausgabe eines digitalen Wertepaares (IX, IY) geeignet (Markierung als Punkt in einer X-Y-Darstellung).

Die digitalen Werte müssen Integergrößen im Bereich ± 2ØØØ sein. Die zugehörige Ausgangsspannung ist ± 5 V.

Das Unterprogramm wird:

1) Im FORTRAN-Programm durch

```
CALL POINT (IX, Iy)
```

2) Im ASSEMBLER-Programm durch

```
CALL POINT
DC   ADRESSE von IX
DC   ADRESSE von IY
```

aufgerufen.

IX, IY müssen, wenn erforderlich, im Hauptprogramm so normiert werden, daß IX, IY maximal 2ØØØ werden.

Zur Steuerung der Feder eines X-Y-Schreibers dient Kanal 7 (Punkt = + 5V, sonst 0V).

Bemerkung: Ein spezielles Kabel zur Ansteuerung des X-Y-Schreibers MOSELEY 7000 AM ist vorhanden.

Unterprogramm KREUZ mit Testprogramm TKREU

benötigtes Unterprogramm: VAR (auf UL)

Die Beschreibung zu KREUZ entspricht der von POINT. KREUZ markiert jedoch ein Wertepaar nicht als Punkt, sondern als Kreuz.

Aufruf:

1.) Fortran: CALL KREUZ (IX, IY)

2.) Assembler:

```
CALL   KREUZ
DC     Adresse von IX
DC     Adresse von IY
```

Unterprogramm LESB

Dieses Unterprogramm befindet sich in der UL. Es ist zum Einlesen von alphanumerischen Zeichen durch TTY in ein Feld des Hauptprogramms geeignet. Es werden je zwei Zeichen in ein Kernspeicherwort abgespeichert. Vor dem Einlesen wird das gesamte Feld mit Blanks überschrieben.

Das Einlesen der Zeichen beginnt an der Stelle, an der die TTY sich momentan befindet. Alle Zeichen werden ohne Echo eingelesen und auf bestimmte Zeichen kontrolliert (vgl. unten). Falls keines der zu kontrollierenden Zeichen vorliegt, wird das Eingabezeichen auch auf TTY geschrieben. Durch diese Abfrage ergibt sich eine gewisse Verzögerung zwischen Betätigung der Taste und Schreiben des Zeichens auf dem Papier. Dieses hat eine etwas langsamere Eingabe des Textes zur Folge, als es beim Lesen mit Echo (Normalbetrieb der TTY) der Fall ist.

Das Unterprogramm erkennt folgende Sonderzeichen:

1) WR — Durch dieses Zeichen wird das Einlesen beendet. Das Zeichen selbst wird nicht in das Feld des Hauptprogrammes abgespeichert, TTY führt WR nicht aus.

2) * — Durch dieses Zeichen wird das Einlesen beendet. Das Zeichen wird als letztes Zeichen in das Feld des Hauptprogrammes abgespeichert. TTY schreibt dieses Zeichen nicht. Es kann vom Hauptprogramm als Marke benutzt werden.

3) @ — Durch dieses Zeichen wird das Einlesen beendet. Das Zeichen wird als letztes Zeichen in das Feld des Hauptprogrammes abgespeichert. TTY schreibt dieses Zeichen nicht. Es kann vom Hauptprogramm als Marke benutzt werden.

4) Rubout Durch dieses Zeichen wird das Einlesen beendet,] auf der TTY ausgegeben und eine neue Zeile mit ?␣ begonnen. Das Feld des Hauptprogrammes wird auf den Ausgangszustand zurückgesetzt. Eine erneute Eingabe des Textes kann erfolgen.

5) ← Durch dieses Zeichen wird das jeweils letzte Zeichen des Textes gelöscht. Es werden soviel Zeichen des Textes (ausgehend vom Ende) gelöscht, wie Korrekturzeichen eingegeben werden. Übersteigt die Anzahl der Korrekturzeichen die Anzahl der Textzeichen, so erfolgt automatisch ein Rubout.

Das Unterprogramm wird:

1) im FORTRAN-Programm durch

```
CALL  LESB (IFELD)
```

2) im ASSEMBLER-Programm durch

```
      CALL    LESB
      DC      IFELD

IFELD DC      max. Anzahl der Worte
      BSS     zu reservierende Wortanzahl
```

aufgerufen.

Im ersten Wort des Feldes muß die Anzahl der einzulesenden Worte stehen (2 Zeichen in einem Wort). Beim Rücksprung aus dem Unterprogramm wird diese Stelle des Feldes nicht mit der tatsächlichen gelesenen Wortanzahl verändert.

Ist die Anzahl der eingelesenen Zeichen größer als die vorher vereinbarte Anzahl, so wird die Eingabe abgebrochen und es erfolgt ein Rücksprung ins Hauptprogramm.

Nach Beendigung des Einlesens stehen die einzelnen Zeichen ausgabegerecht im Feld des Hauptprogrammes, d. h.

1. Wort	2. Wort	3. Wort
max. Wortanzahl	1.Zeichen/ 2. Zeichen	3. Zeichen/.....

GA 18/30 - Information

Unterprogramm LESN

Dieses Unterprogramm befindet sich in der UL. Es ist zum Einlesen von alphanumerischen Zeichen durch TTY in ein Feld des Hauptprogrammes geeignet. Es werden je zwei Zeichen in ein Kernspeicherwort abgespeichert. Vor dem Einlesen wird das gesamte Feld mit /ØØØØ (binäre Nullen) überschrieben. Bei der Ausgabe dieser binären Nullen schreibt die TTY nicht (Here Is-Taste).

Dieses Unterprogramm unterscheidet sich vom UnterprogrammLESB durch das Überschreiben des Feldes mit binären Nullen anstelle von Blanks bei LESB. Alle anderen Einzelheiten sind gleich und können bei der Beschreibung des Unterprogrammes LESB nachgelesen werden.

Das Unterprogramm wird:

1) im FORTRAN-Programm durch

```
CALL LESN (IFELD)
```

2) im ASSEMBLER-Programm durch

```
        CALL    LESN
        DC      IFELD

IFELD   DC      max. Anzahl der Worte
        BSS     zu reservierende Wortanzahl
```

aufgerufen.

Unterprogramm LESTY

Dieses Unterprogramm befindet sich in der UL. Es ist zum Einlesen von alphanumerischen Zeichen durch TTY in ein Feld des Hauptprogrammes geeignet. Es werden je zwei Zeichen in ein Kernspeicherwort abgespeichert. Vor dem Einlesen wird das gesamte Feld mit binären Nullen überschrieben.

Das Einlesen der Zeichen beginnt am Anfang einer Zeile. Nach dem Aufruf des Unterprogrammes wird als erstes WR, ZV, ? , Blank ausgeführt. Alle Zeichen werden mit Echo eingelesen, d. h. es erfolgt eine simultane Ausgabe des Zeichens auf dem Papier der TTY.

Das Unterprogramm erkennt folgende Sonderzeichen:

1) WR — Durch dieses Zeichen wird das Einlesen beendet. Das Zeichen selbst wird nicht in das Feld des Hauptprogrammes abgespeichert. TTY führt WR aus. Die tatsächliche Wortanzahl wird auf die Anfangsadresse des Feldes im Hauptprogramm gespeichert. Danach erfolgt der Rücksprung ins Hauptprogramm.

2) Rubout — Durch dieses Zeichen wird das Einlesen beendet,] auf der TTY ausgegeben und eine neue Zeile mit ?␣ begonnen. Das Feld des Hauptprogrammes wird auf den Ausgangszustand zurückgesetzt. Eine erneute Eingabe des Textes kann erfolgen.

3) ← — Durch dieses Zeichen wird das jeweils letzte Zeichen des Textes gelöscht. Es wurden soviel Zeichen des Textes (ausgehend vom Ende) gelöscht, wie Korrekturzeichen eingegeben werden. Übersteigt die Anzahl der Korrekturzeichen die Anzahl der Textzeichen, so erfolgt automatisch ein Rubout.

Das Unterprogramm wird:

1) im FORTRAN-Programm durch

CALL LESTY (IFELD)

2) im ASSEMBLER-Programm durch

```
        CALL    LESB
        DC      IFELD

IFELD   DC      max. Anzahl der Worte
        BSS     zu reservierende Wortanzahl
```

aufgerufen.

Im ersten Wort des Feldes muß die Anzahl der einzulesenden Worte (2 Zeichen in ein Wort) stehen. Beim Rücksprung aus dem Unterprogramm wird diese Stelle des Feldes mit der tatsächlich gelesenen Wortanzahl verändert, um einem Ausgabeprogramm die tatsächliche Wortanzahl als Parameter zu übergeben. Bei einem erneuten Aufruf von LESTV und Benutzung des gleichen Feldes im Hauptprogramm ist deshalb die maximale Anzahl der einzulesenden Zeichen erneut auf diese Stelle zu speichern.

Ist die Anzahl der eingelesenen Zeichen größer als die vorher vereinbarte Anzahl, so wird die Eingabe abgebrochen und es erfolgt ein Rücksprung in Hauptprogramm.

Nach Beendigung des Einlesens stehen die einzelnen Zeichen ausgabegerecht im Feld des Hauptprogrammes, d. h.

1. Wort	2. Wort	3. Wort
Wortanzahl	1. Zeichen/ 2. Zeichen	3. Zeichen/.....

Eine Umspeicherung des Feldes ist nicht erforderlich.

Unterprogramm LEST1

Dieses Unterprogramm befindet sich in der UL. Es ist zum Einlesen von alphanumerischen Zeichen durch TTY in ein Feld des Hauptprogrammes geeignet. Es werden je zwei Zeichen in ein Kernspeicherwort abgespeichert. Vor dem Einlesen wird das gesamte Feld mit binären Nullen überschrieben.

Das Einlesen der Zeichen beginnt an der Stelle, an der die TTY sich momentan befindet. Alle Zeichen werden mit Echo eingelesen, d. h. es erfolgt eine simultane Ausgabe des Zeichens auf dem Papier der TTY.

Dieses Unterprogramm unterscheidet sich in den weiteren Punkten nicht vom Unterprogramm LESTY. Weitere Einzelheiten können bei der Beschreibung dieses Unterprogrammes nachgelesen werden.

Das Unterprogramm wird:

1) im FORTRAN-Programm mit

```
CALL LEST1 (IFELD)
```

2) im ASSEMBLER-Programm mit

```
        CALL    LEST1
        DC      IFELD

IFELD   DC      max. Anzahl der Worte
        DC      zu reservierende Wortanzahl
```

aufgerufen.

Unterprogramm TYPE

Dieses Unterprogramm befindet sich in der UL. Es ist zur Ausgabe von alphanumerischen Zeichen durch TTY aus einem Feld des Hauptprogrammes geeignet. Es können maximal zwei Zeichen je Kernspeicherwort ausgegeben werden. Der Text muß im Kernspeicherwort im ASCII-Code vorhanden sein, wobei das 1. Zeichen linksbündig und das 2. Zeichen rechtsbündig im Kernspeicherwort stehen muß. Ein durch die Unterprogramme LESTY, LEST1, LESB, LESN erstelltes Datenfeld kann direkt durch TYPE ausgegeben werden.

Die Ausgabe der Zeichen beginnt mit einer neuen Zeile.

Das Unterprogramm wird

1) im FORTRAN-Programm durch

```
CALL TYPE (IFELD)
```

2) im ASSEMBLER-Programm durch

```
        CALL    TYPE
        DC      IFELD
        .
        .
        .
IFELD   DC      6
        ASC     .Ausgabetext.
```

aufgerufen.

GA 18/30 - Information

Unterprogramm TYPE1

Dieses Unterprogramm befindet sich in der UL. Es ist zur Ausgabe von alphanumerischen Zeichen durch TTY aus einem Feld des Hauptprogrammes geeignet.

Dieses Unterprogramm unterscheidet sich vom Unterprogramm TYPE nur darin, daß die Ausgabe der Zeichen an der Stelle beginnt, an der sich die TTY momentan befindet. Alle weiteren Einzelheiten entsprechen dem Unterprogramm TYPE und können dort nachgelesen werden.

Das Unterprogramm wird

1) im FORTRAN-Programm durch

```
CALL TYPE1 (IFELD)
```

2) im ASSEMBLER-Programm durch

```
        CALL    TYPE1
        DC      IFELD
        .
        .
        .
        .
        .
IFELD   DC      6
        ASC     „Ausgabetext“.
```

aufgerufen.

GA 18/30 - Information

Unterprogramm STAXY mit Testprogramm TSTAX

Dieses Unterprogramm befindet sich in der UL. Es kann dazu benutzt werden, den X-Y-Schreiber in eine Startposition zu bringen, ohne daß die Feder aufsetzt. Die Startposition kann beliebig durch das Wertepaar (IX, IY) angegeben werden.

Die Werte IX und IY müssen Integergrößen im Bereich ± 2000 sein. Die zugehörige Ausgangsspannung ist ± 5V.

Das Unterprogramm wird

1) im FORTRAN-Programm durch

```
CALL STAXY (IX, IY)
```

2) im ASSEMBLER-Programm durch

```
CALL STAXY
DC    ADRESSE von IX
DC    ADRESSE von IY
```

aufgerufen.

IX, IY müssen, wenn erforderlich, im Hauptprogramm so normiert werden, daß sie maximal 2000 werden.

Ausgabekanäle: IX = Kanal Nr. 0
IY = Kanal Nr. 1
Federsteuerung = Kanal Nr. 7

GA 18/30 - Information

Unterprogramm KOORD

Dieses Unterprogramm befindet sich in der UL. Es kann zur Erstellung eines Koordinatennetzes benutzt werden.

Das Unterprogramm wird

1) im FORTRAN-Programm durch

```
CALL KOORD (NQUAD, NX, NY)
```

2) im ASSEMBLER-Programm durch

```
CALL KOORD
DC      ADRESSE von NQUAD
DC      ADRESSE von NX
DC      ADRESSE von NY
```

aufgerufen.

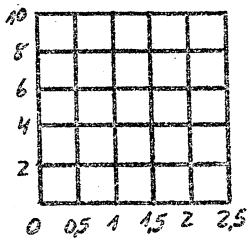
Es bedeutet:

NQUAD = 1 Es wird nur der 1. Quadrat ausgegeben.
= 2 Es wird der 1. + 2. Quadrat ausgegeben.
= 3 Es wird der 1. + 2. + 3. + 4. Quadrat ausgegeben.
= 4 Es wird der 1. + 2. + 3. + 4. Quadrat ausgegeben.

NX = Anzahl der Unterteilung der Abszisse (einschließlich der Null-Linie)

NY = Anzahl der Unterteilungen der Ordinate (einschließlich der Null-Linie)

z. B.



NQUAD = 1

NX = 6

NY = 6

Dieses Unterprogramm benötigt folgende Analogen-Ausgabekanäle

0 = Ausgabe der X-Werte

1 = Ausgabe der Y-Werte

7 = Triggersignal für Penlift

Unterprogramm SETP

Dieses Unterprogramm befindet sich in der UL. Es dient zum Setzen der Servopotentiometer des Analogrechners EAI-580.

Das Unterprogramm wird

1) im FORTRAN-Programm durch

```
CALL    SETP (IWERT, INR)
```

2) im ASSEMBLER-Programm durch

```
CALL    SETP
DC      ADRESSE von IWERT
DC      ADRESSE von INR
```

aufgerufen.

IWERT ist 10000 x Sollwert des Potis
INR ist die Adresse des Potis
z. B.

```
IWERT  =  8325 !
INR    =  11   !
```

Poti Nr. 11 wird dann auf den Wert 0,8325 gesetzt.

Unterprogramm DVM

Dieses Unterprogramm befindet sich in der UL. Es dient zum Auslesen des Digitalvoltmeters von Analogrechner EAI-580.

Das Unterprogramm wird

1) im FORTRAN-Programm durch

```
CALL    DVM (IWERT)
```

2) im ASSEMBLER-Programm durch

```
CALL    ANOUT
DC      ADRESSE von IWERT
```

aufgerufen.

IWERT ist eine Integerzahl und gibt die Anzeige des DVM in mV an.

Unterprogramm INTEF

Dieses Unterprogramm befindet sich in der UL. Es dient zur Steuerung des Analogrechners EAI - 580 (Betriebsarten auswählen, Komponenten Adressieren usw.).

Das Unterprogramm wird

1) im FORTRAN-Programm durch

```
CALL    INTEF (PARAM)
```

2) im Assembler-Programm durch

```
CALL    INTEF
DC      ADRESSE des PARAMETERS
```

aufgerufen.

Über die einzelnen Parameter wird festgelegt, welche Steuerfunktion ausgeführt werden soll.

	Parameter		
	Hexadezimal	Dezimal	Steuerfunktion
1. Adressierung	/00XX	00 - 99	XX = Adresse
2. Komponentenanwahl			
2.1	/1001	+ 4097	A
2.2	/1002	+ 4098	P
2.3	/1004	+ 4100	D/10
2.4	/1008	+ 4104	D
2.5	/1010	+ 4132	F
2.6	/1020	+ 4148	PP
2.7	/1040	+ 4224	Reserve
3. Analogteil			
3.1	/8000	- 32768	gesamtes Interface löschen
3.2 Set - Clear - Poti			
3.2.1	/7100	28888	Set Poti
3.2.2	/7200	29144	Clear Poti

		Parameter		
		Hexadezimal	Dezimal	Steuerfunktion
3.3	Timer			
3.3.1		/6 80	24704	1 sec
3.3.2		/6040	24640	2 ms
3.4	Betriebsarten			
3.4.1		/5001	20481	ST
3.4.2		/5002	20482	SP
3.4.3		/5004	20484	OP
3.4.4		/5008	20488	HOLD
3.4.5		/5010	20496	IC
3.4.6		/5020	20512	PP
4.	Logikteil			
4.1	Betriebsarten			
4.1.1		/4010	16400	RUN
4.1.2		/4020	16416	SET
4.1.3		/4040	16448	CLEAR
4.1.4		/4080	16512	PP
4.2	Clock			
4.2.1		/2001	8193	STEP
4.2.2		/2002	8194	10^{1}
4.2.3		/2004	8196	10^{5}
4.2.4		/2008	8200	10^{6}

Den einzelnen Parametern können in einem FORTRAN-Programm durch eine DATA - Vereinbarung Hexadezimale Werte zugewiesen werden, z. B.

```
DATA  IAMPL, IPOTI, ITIM1, ICLST/Z1001, Z1002, Z6080, Z2001/
```

```
CALL INTEF (IAMPL)      Verstärker anwählen
CALL INTEF (IPOTI)      Poti anwählen
CALL INTEF (ITIM1       Timer auf 1s setzen
CALL INTEF (ICLST)      Clock für die Logik auf STEP setzen.
```

Die Parameter müssen alle Integergrößen sein. Sie können auch durch Zuweisung ihrer Dezimalwerte vereinbart werden, z. B.

```
IAMPL  =  4097
IPOTI  =  4098
ITIM1  =  24704
ICLST  =  8193
```

Unterprogramm DBET
Dieses Unterprogramm befindet sich in der UA

Zweck:
Berechnung von Determinanten vom Rang K≤10

Aufruf:

CALL DBET (A, DET , K)

mit

A (10 ,10) = Determinante in Normalform abgespeichert
DET = Wert der Determinante
K = Rang der zu lösenden Determinante(max.10)

Ist die Determinante gleich null, so wird der Rang der darin enthaltenen Restdeterminante,die nicht verschwindet,nicht berechnet; K wird also nicht verändert.

Programm erstellt von : K. Schulz

Function IWNDL

Dieses Unterprogramm befindet sich in der UL

Zweck :

Umwandlung von eingelesenen Zahlen in zugeordnete Kennzahlen.
In eine Integer-Variable werden ein oder zwei Buchstaben im Format A1 oder A2 eingelesen und jeder Buchstabe in eine Kennzahl umgewandelt.Bei zwei Buchstaben werden die Kennzahlen addiert.

Aufruf:

```
      READ (7,101) IB
101   FORMAT (A1)
        .
        .
      NB = IWNDL (IB)
```

mit
IB = eingelesene Buchstaben
NB = Kennzahl der Buchstaben

Zuordnung von Buchstaben und Kennzahlen :

A	B	C	D	E	F	G	H	I	J	K	L	M
1	2	3	4	5	6	7	8	9	17	18	19	20

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
21	22	23	24	25	34	35	36	37	38	39	40	41

Ein Leerzeichen (Blank) hat die Kennzahl 0

Programm erstellt von : R. Schulz

UL-24

GA 18/30 Information

DETER

Unterprogramm DETER
befindet sich in der UL

Lösung [illegible]

Das lin. Gleichungssystem muß in Matrizenschreibweise [illegible] gegeben werden :

$$a_{11} x_1 + a_{12} x_2 \dots a_{1n} x_n = y_1$$
$$a_{21} x_1 + \dots a_{2n} x_n = y_2$$
$$\dots$$
$$a_{n1} x_1 \dots a_{nn} x_n = y_n$$

Aufruf:

CALL DETER (D , BS , E , L , M)

mit

D (6 , 6) = Determinante von (a_{ik})
BS (6) = Rechte Seite des Gleichungssystems ($\vec{y}$)
E (6) = Ergebnis ($\vec{x}$)
L = Aktuelle Länge der Determinanten (max.6)
M = Marke M=1 : Lösung o.k.
M=2 : Keine Lösung möglich weil Det.=0

Das Unterprogramm läuft auf Stop 6,wenn L<0 ist.

Benötigte Unterprogramme : DET 6

Programm erstellt von : V. Pialas

UL-25

Unterprogramm MINFE

Dieses Unterprogramm befindet sich in der UL

Zweck :

Bestimmung der Koeffizienten einer linearen Gleichung (max.5.Ordnung) aus einer Anzahl von Punkten(max.30) nach folgendem Ansatz :

$$Y=A_0 + A_1X + A_2X^2 + A_3X^3 + A_4X^4 + A_5X^5$$

Die Koeffizienten werden so bestimmt,daß die Summe der Fehlerquadrate minimal wird.

Aufruf: (Fortran)

CALL MINFE (XW , YW , IMAX , E , N , M)

mit

XW (50)	= Feld der X-Werte
YW (50)	= Feld der Y-Werte
IMAX	= Aktuelle Feldlänge ≤ 50
E(6)	= Koeffizienten $A_0 - A_5$ (A_0 = E(1).....A_5=E(6))
N	= Anzahl der gew. Koeffizienten (N-1=Ordnung des Polynoms)
	Nmax=6
M	= Marke M=1 : Lösung O.K
	M=2 : Keine Lösung möglich

Benötigte Unterprogramme: DETER (→DET 6) , FUNCT

Programm erstellt von : V. Bialas

UL-26

Unterprogramm FLNET

Dieses Unterprogramm befindet sich in der UL

Zweck:

Es wird zur Erstellung eines logarithmischen Koordinatennetzes benutzt.

Aufruf: (Fortran)

CALL FLNET (NX , NY , RE)

mit

NX = Anzahl der Dekaden in X-Richtung

NY = Anzahl der Dekaden in Y-Richtung

RE = Normierte Länge einer Dekade (1.0 ≙ 5V)

Wird RE=0. übergeben, so berechnet sich das Unterprogramm die günstigste Dekadenlänge selbst, und zwar so, daß der Ausgabespannungsbereich (-5V bis +5V) ganz genutzt wird. Die so berechnete Dekadenlänge wird dem Hauptprogramm übergeben und kann dort zur Normierung der Ausgabegrößen verwandt werden. Wird beim Aufruf von FLNET ein zu großes RE übergeben (RE > 2/NX oder RE > 2/NY), so erfolgt eine Fehlermeldung und es müssen neue Werte für NX , NY , und RE über TY eingegeben werden. (FORMAT 2I2,F 10.5)

Beispiel: 04040.5

Dem Benutzer wird die Möglichkeit gegeben, die Anzahl der Unterteilungen für eine Dekade (IRX bzw. IRY) und ihre Lage (RX(IRX) bzw. RY(IRY)) selbst zu bestimmen oder aber die folgenden Standardwerte zu belassen:

IRX = IRY = 06

RX (1) = RY (1) = 1.0
RX (2) = RY (2) = 2.0
RX (3) = RY (3) = 4.0
RX (4) = RY (4) = 6.0
RX (5) = RY (5) = 8.0
RX (6) = RY (6) = 10.0

LL-2F

Sollen diese Standardwerte benutzt werden,so sind IRX und IRY nach dem Aufruf mit 0 einzugeben.

Sonst aber sind die gewünschten Größen nach dem Aufruf untereinander in I2 bzw. F10. einzugeben.IRX und IRY dürfen maximal 10 sein.

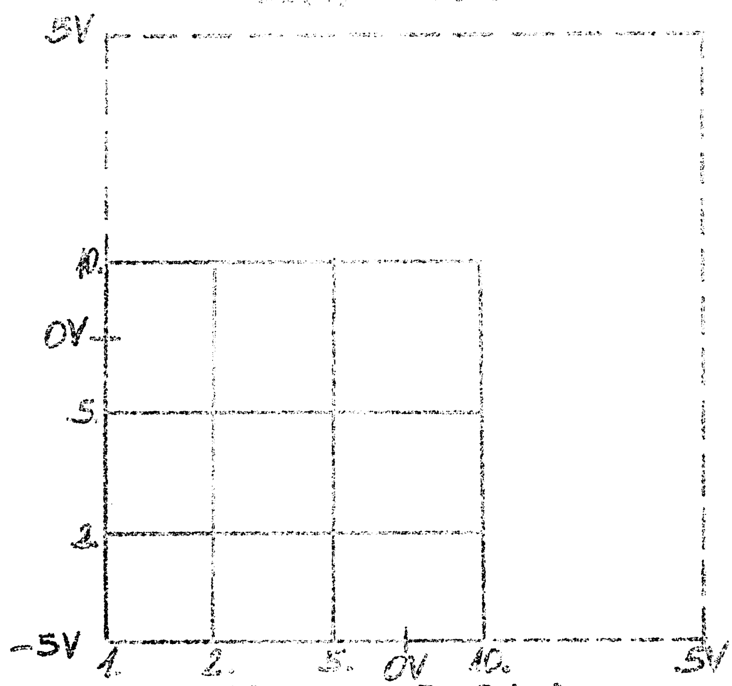
Dabei ist zu beachten,daß die ersten Werte von RX und RY 1.0 und die letzten Werte 10.0 sein sollten und sowohl der erste Wert (1.0) wie der letzte Wert (10.0) für die Anzahl der Unterteilungen mitzählen. Beispiel:

IRX = 04
RX(1) = 1.0
RX(2) = 2.0
RX(3) = 5.0
RX(4) = 10.0

Vor dem Zeichnen des Netzes werden zwei gegenüberliegenden Ecken mit dem Schreiber angefahren,damit der Benutzer den Schreiber einstellen kann.Bei Bedarf wird dieses wiederholt,wenn anschließend eine 1 eingegeben wird.

Beispiel: CALL FLNET (1, 1 , 1.25)

Eingaben:	IRX = 04	IRY = 04
	RX(1) = 1.0	RY(1) = 1.0
	RX(2) = 2.0	RY(2) = 2.0
	RX(3) = 5.0	RY(3) = 5.0
	RX(4) = 10.0	RY(4) = 10.0



Programm erstellt von : R. Schulz

LLL-28

Programmname: MIX12

Zweck: Aufbau eines Programms durch Mischen zweier Programme

Eingabe: Programm 1 von LUN = 1
Programm 2 von LUN = 15

Ausgabe: nach LUN = 2 (Standard: US)

Zu beachten: Die LUN's 1, 2, 15 dürfen nicht dieselben Namensfile benutzen

Also nicht:
~~$ S1 = LEC(Prog1)~~
~~$ 15 = LEC(Prog2)~~
sondern beispielsweise:
$ S1 = LEC(Prog1)
$ 15 = XX(Prog2)

Vor dem Aufruf des Programms MIX12 ist die laufende Kartennumerierung der Programme 1 und 2 vorzunehmen. Das geschieht mit Hilfe von $LIST (Eingabe von S1 / Ausgabe nach LO). Man notiert sich, welche Karten übernommen werden sollen (von ... bis ...), da MIX12 folgende Eingabe verlangt:
File-NR .., VON KARTE ... BIS ...

Bei der Zusammenstellung des neuen Programms ist die Reihenfolge der Karten beliebig, z.B. zunächst die Karten 30-50, dann die

Karten 5 - 20.

Nach dieser Vorarbeit kann die Programmzusammenstellung durchgeführt werden.

Wegen des erforderlichen Arbeitsaufwands lohnt sich dieses Verfahren nur für die Übernahme/Mischen größerer Programmteile.

Aufruf:

```
$ JOB
$ S1 = ...
$ 15 = ...
ggf. $ SO = ...
$ MIX12
```

Die Beendigung des Programms erfolgt, sobald ein "END" erkannt worden ist bzw. die Programmbeendigung ausdrücklich gewünscht wird (dann wird automatisch eine "END"-Marke geschrieben).

GA 18/30 - Information

LIBRARY (LB)

Alle vom System benötigten Unterprogramme befinden sich in der LB und sind dort unter ihrem Namen abgelegt. Sie werden durch das Programm "CIC" (CORE - IMAGE - CONVERTER) aufgerufen und dem Hauptprogramm hinzugefügt.

Der Steuerbefehl lautet:

```
$    CIC
x    MAP  +)
*    BUILD
```

+) Dieser Steuerbefehl ist nur erforderlich, wenn CORE - MAP

GA 18/30 - Information

Unterprogramm TYIN

Dieses Unterprogramm befindet sich in der LB. Es ist zur Eingabe eines Zeichens durch TTY gedacht, wobei dieses Zeichen nicht auf TTY ausgegeben wird (Schreiben ohne Echo). Dieses Zeichen wird nach dem Einlesen linksbündig ins A-Register (XR4) geladen und steht dort den weiteren Programmen zur Verfügung.

Das Unterprogramm wird im ASSEMBLER-Programm durch

```
.
.
.
CALL ... TYIN
.
.
.
```

aufgerufen.
Danach ist das Zeichen im A-Register wie folgt zu lesen:

XXØØ

XX : ASCII - Code des eingegebenen Zeichens

Beispiel:

```
.
.
.
CALL . . . . TYIN      Aufruf von TYIN
STO. . . . .TYØØ1      Eingabezeichen abspeichern
.
.
.
.
.
TYØØ1 .DC. . . . . /XXØØ    Eingabezeichen im ASC II-Code
.
.
.
```

GA 18/30 - Information

Unterprogramm TYINE

Dieses Unterprogramm befindet sich in der LB. Es ist zur Eingabe eines Zeichens durch TTY gedacht, wobei dieses Zeichen gleichzeitig auch auf TTY ausgegeben wird (Schreiben mit Echo). Dieses Zeichen wird nach dem Einlesen linksbündig ins A-Register (XR4) geladen und steht dort den weiteren Programmen zur Verfügung.

Das Unterprogramm wird im ASSEMBLER-Programm durch

```
        .
        .
        .
    CALL . . . . TYINE
        .
        .
        .
```

aufgerufen.

Danach ist das Zeichen im A-Register wie folgt zu lesen:

XXØØ

XX: ASCII-Code des eingegebenen Zeichens

Beispiel:

```
        .
        .
        .
CALL . . . . TYINE      Aufruf von TYINE
STO          TYØØ1      Eingabezeichen abspeichern
        .
        .
        .
        .
        .
TY001 . DC . . . . . /XXØØ    Eingabezeichen im ASCII-Code
```

Unterprogramm TYOUT

Dieses Unterprogramm befindet sich in der LB. Es ist zur Ausgabe eines Zeichens auf TTY gedacht. Dieses Zeichen muß beim Aufruf des Unterprogrammes linksbündig im A-Register (XR4) stehen.

Das Unterprogramm wird im ASSEMBLER-Programm durch

```
        .
        .
        .
CALL . . . . TYOUT
        .
        .
        .
```

aufgerufen.
Das Zeichen muß vor dem Aufruf wie folgt im A-Register stehen:

XXØØ

XX: ASCII-Code des auszugebenden Zeichens.

Beispiel:

```
        .
        .
        .
LD  . . . . . TY001   Ausgabezeichen laden
CALL          TYOUT   Aufruf von TYOUT
        .
        .
        .
        .
        .
TYØØ1 . DC . . . . . /XXØØ   Ausgabezeichen im
                              ASCII-Code
        .
        .
        .
```

Unterprogramm WDAT

Dieses Unterprogramm befindet sich in der LB. Es ist zum Auslesen der Console-Dataswitches (DATSW) geeignet. Der Wert der DATSW wird einer vorgegebenen Variablen zugewiesen.

1) Im FORTRAN-Programm durch

```
CALL    WDAT (INAME)
```

2) im ASSEMBLER-Programm durch

```
        .
        .
        .
      CALL . . . . WDAT
      DC.. . . . . INAME
        .
        .
        .
        .
INAME . DC.. . . . . 0
```

aufgerufen.

INAME = Adresse, auf der der Wert von DATSW gespeichert wird.

Unterprogramm IDATS

Dieses Unterprogramm befindet sich in der LB. Es ist zum Auslesen eines definierten Console-Dataswitches (DATSW) geeignet. Die Nummer des auszulesenden DATSW entspricht der Numerierung auf der Console und wird dem Programm als Parameter übergeben. Ist der entsprechende DATSW gesetzt, so wird im A-Register (XR4) eine 1 gespeichert, ist DATSW nicht gesetzt, so steht im A-Register (XR4) eine 0.

Das Unterprogramm wird

1) im FORTRAN-Programm durch

IDATS (N) z. B. IF (IDATS (N)) 10, 11, 12

2) im ASSEMBLER-Programm durch

```
CALL    IDATS
DC      N
```

aufgerufen.

N = Nr. des auszulesenden DATSW (0 - 15)

Unterprogramm NRDAT

Dieses Unterprogramm befindet sich in der LB. Es ist zur Bestimmung der Nummer des ersten gesetzten Console-Dataswitches (DATSW) geeignet. Die Nummer entspricht der Numerierung auf der Console und wird vom Programm einer definierten Variablen zugewiesen.

Das Unterprogramm wird

1) im FORTRAN-Programm durch

```
CALL NRDAT (N)
```

2) im ASSEMBLER-Programm durch

```
CALL    NRDAT
DC      N
```

aufgerufen.

N = Nr. des ersten gesetzten DATSW