Ahrens, L.; van Almelo, J.; Rueck, Günter-Friedrich

Die Unterprogrammbibliothek BWLLIB

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 80

Provided in Cooperation with:
Christian-Albrechts-University of Kiel, Institute of Business Administration

Institut fuer Betriebswirtschaftslehre
der Christian-Albrechts-Universitaet Kiel

Lehrstuhl fuer Finanzwirtschaft
Direktor: Prof. Dr. R. Schmidt

Nr. 80

Die Unterprogrammbibliothek

B W L L I B

von

L. Ahrens, J. van Almelo, G.F. Rueck

Stand: 1. Juni 1980

Diese Dokumentation wurde mit Hilfes des Programms RUNOFF auf der PDP-11/45 der wirschafts- und sozialwissenschaftlichen Fakultaet der Christian-Albrechts-Universitaet Kiel erstellt.

Inhalt

Abschnitt 1

Grundlagen

1.1 Einleitung

Die Unterprogrammbibliothek DSKB:BWLLIB.REL[6400,64000] steht in erster Linie den Angehoerigen des Institutes fuer Betriebswirtschaftslehre zur Verfuegung, kann aber auch von allen anderen Benutzern der PDP-10 verwendet werden (1). Sie umfasst eine Sammlung von MACRO-10-Moduln (2), die im Rahmen der Entwicklung der Systeme DELPHI und PLASMA entstand.

Vor der Aufnahme in die Bibliothek wurden alle Moduln getestet und ggfs ueberarbeitet. Sollten dennoch in der Anwendung Fehler oder Probleme auftreten, so bitten die Autoren um umgehende Nachricht, moeglichst mit "Beweismaterial". Informationen ueber den Zustand der Bibliothek sind in dem File DSKB:BWLLIB.HLP[6400,64000] enthalten.

Die Bibliotheken MAC40.REL und MAC10.REL stehen nach der Freigabe von BWLLIB nicht mehr zur Verfuegung. Die Benutzer

---

(1) Bei der Benutzung der BWLLIB wird um Beachtung folgender Punkte gebeten.
- Es ist verboten, die Bibliothek zu kopieren.
  Ausnahme: Im Rahmen von Batch-Jobs darf die BWLLIB zum Hintergrundrechner KL1091 kopiert werden, um dort ein Laden der Programme zu ermoeglichen.
- Es wird gebeten, in Veroeffentlichungen, die unter Verwendung der BWLLIB entstanden sind, dieses Paper zu zitieren.

(2) MACRO-10 ist die Assemblersprache der PDP-10.

werden gebeten, ihre Programme und/oder Kommandofiles darauf abzustimmen.

1.2 Benutzung

Die Moduln koennen von in FORTRAN-10 (3) und MACRO-10 geschriebenen Programmen benutzt werden. Die Verwendung in ALGOL und in PASCAL erscheint moeglich, wurde jedoch nicht getestet.

Die Routinen der BWLLIB koennen (in FORTRAN-Programmen) wie Systemfunktionen verwendet werden, d.h. es sind keine besonderen Deklarationen erforderlich. Lediglich beim Laden muss die Bibliothek definiert werden :

.LOA <prog>,<sub1>,...,<subn>,DSKB:BWLLIB.REL[6400,64000]/SEARCH

(entsprechend fuer die Kommandos .EXE und .DEB)

<prog>,<sub1>,...,<subn> bezeichnen das Programm und die eigenen Subroutines des Benutzers. Der Switch /SEARCH bewirkt, dass nur die tatsaechlich benoetigten Moduln aus der BWLLIB geladen werden. Ein Fortlassen dieses Switches hat eine unnoetige Vergroesserung des Speicherbedarfs des Programms zur Folge, da in diesem Fall alle Moduln aus der Bibliothek geladen werden. Die Extension .REL darf fehlen, dies hat jedoch eine unnoetige Verlaengerung der Ausfuehrungszeit des Ladekommandos zur Folge, da zunaechst ein Source-File BWLLIB gesucht wird. (Bei der grossen Anzahl der zur Verfuegung stehenden Compiler u.U. eine lang dauernde Prozedur).

---

(3) FORTRAN-Programme muessen mit dem FORTRAN-10 Compiler uebersetzt werden.

Natuerlich kann DSKB:[6400,64000] mit Hilfe des System-Programms SETSRC auch als LIB: definiert werden. Wir verweisen hier auf die vom Rechenzentrum zur PDP-10 herausgegebene Literatur.

1.3 Parameterbeschreibungen

Die Beschreibung der Parameter und deren Wirkung enthaelt auch Angaben des Typs des jeweiligen Parameters. Die Typ-Angaben stellen sich formal als [<v>,<t>{(<d>)}] dar, wobei geschweifte Klammern {...} optionale Teile anzeigen. Die Felder haben im einzelnen folgende Bedeutung :

<v> Art des Parameters :

- A der Inhalt des Parameters wird durch den Aufruf nicht veraendert. Erwartet die Routine eine einfache Variable, so kann im Aufruf eine Konstante oder ein Ausdruck spezifiziert werden.
- V fuer den Parameter muss eine Variable angegeben werden, da der Inhalt durch den Aufruf moeglicherweise veraendert wird.

<t> Datentyp des Parameters :

I=integer, R=real, D=double precision, X=complex, L=logical, *=beliebiger Datentyp (z.B. bei Texten)

<d> Dimensionierung der Variablen (Vektoren oder Matrizen).

Fehlt diese Angabe, so wird eine einfache Variable oder ein Feldelement als Parameter erwartet. Die Angabe der Dimensionierung erfolgt in der in FORTRAN ueblichen Weise (DIMENSION-Anweisung). Ein * zeigt eine von den Parametern abhaengige Dimensionierung an.

Sind fuer einen Parameter mehrere Typen moeglich, z.B. bei der Uebergabe von Texten, sind in der Parameterbeschreibung auch mehrere Parameter-Definitionen enthalten.

Beispiele

[A,I]	Ausdruck vom Typ INTEGER
[V,D(2,4)]	2x4-Matrix vom Typ DOUBLE PRECISION
[A,I(2)]	Ausdruck vom Typ INTEGER der Laenge von zwei Worten (z.B. zur Uebergabe eines Textes von zehn Zeichen)
[V,*(*)]	Variable beliebigen Datentyps. Die Dimensionierung haengt von weiteren Parameteren ab und wird in der jeweiligen Beschreibung naeher erlaeutert.
[A,R(3,N)]	3xN-Matrix deren Inhalt durch den Aufruf nicht veraendert wird. Die Anzahl der Spalten N ist durch einen weiteren Parameter N gegeben, oder wird in der Beschreibung naeher erlaeutert.

1.4 Wortformate an der PDP-10

Ein Maschinenwort der PDP-10 hat 36 Bits, die von links nach rechts von 0 bis 35 durchnumeriert sind. Aus dem umfangreichen Satz von Wortformaten werden nur die fuer die Anwendung der BWLLIB relevanten Formate dargestellt.

1.4.1 Integer-Format

Ganzzahlige (INTEGER-) Werte werden in einem 36-Bit-Wort dargestellt, wobei das ganz links stehende Bit 0 als Vorzeichen verwendet wird (1=negativ, 0=positiv). Der Bereich der darstellbaren Zahlen umfasst die Werte von -2**35 bis +2**35-1, wobei negative Zahlen im Zweier-Komplement dargestellt werden.

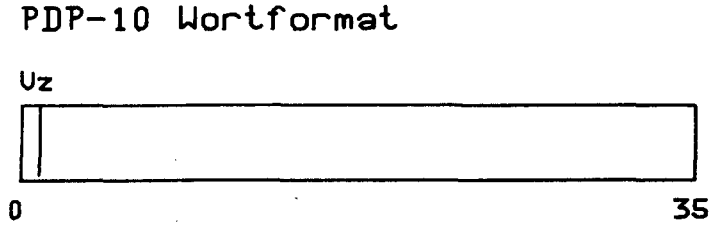


Abb 1: PDP-10 Wortformat

1.4.2 Halbworte

Der Befehlssatz der PDP-10 unterstuetzt die Behandlung von Halbworten zu 18 Bit. In einem Halbwort koennen Zahlen von 0 bis 2**18-1, oder, wenn die Inhalte mit Vorzeichen betrachtet werden, von -2**17 bis +2**17-1, dargestellt werden. Als Vorzeichenbit fungieren Bit 0 fuer das linke und Bit 18 fuer das rechte Halbwort.

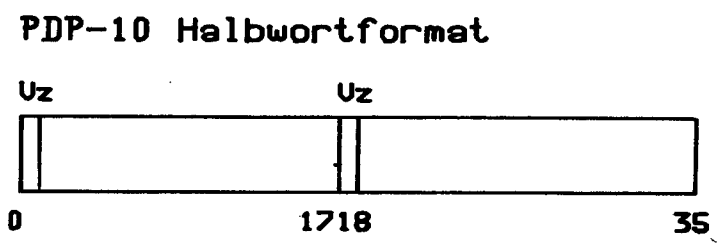


Abb 2: Halbwortformat

1.4.3 ASCII-Text

Fuer die Darstellung eines ASCII-Symbols werden sieben Bits benoetigt, so dass ein PDP-10 Wort fuenf Zeichen aufnehmen kann. ASCII-Texte werden immer linksbuendig dargestellt. Bit 35 (ganz rechts) wird fuer die Darstellung nicht benoetigt und hat immer den Wert 0.(1) Die linksbuendige Speicherung hat fuer den FORTRAN-Benutzer den entscheidenden Nachteil, dass ein Text als (Integer-) Zahl betrachtet, sowohl positive als auch negative Werte, abhaengig vom ersten Zeichen, annehmen kann: Texte, die mit einem Zeichen beginnen, dessen ASCII-Code kleiner als 64 ist,

---

(1) Bit 35 dient zur Unterscheidung von Zeilennummern und Text in zeilenorientierten Files, wie sie z.B. mit dem Editor LINED erstellt werden.

sind positiv, alle anderen negativ (siehe auch Anhang A). BWLLIB enthaelt einen Modul zur Sortierung von ASCII-Texten (TSORTO), der entsprechend den ASCII-Werten sortiert.

PDP-10 ASCII-Format

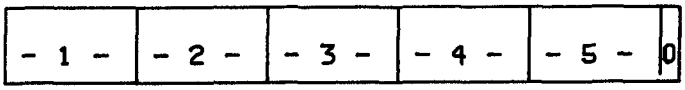


Abb 3: ASCII-Format

1.4.4 SIXBIT-Texte

Zur internen Darstellung von Texten (Filenamen, Variablennamen etc) benutzt die PDP-10 eine Teilmenge des ASCII-Codes, den SIXBIT-Code. Er enthaelt keine Kleinschreibung und Control-Characters. Fuer die Darstellung eines Zeichens werden sechs Bits benoetigt, so dass in einem Wort sechs Zeichen dargestellt werden koennen. BWLLIB unterstuetzt durch zwei Routinen die Konversion von ASCII- in SIXBIT-Texte und umgekehrt.

PDP-10 SIXBIT-Format

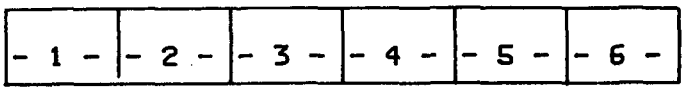


Abb 4: SIXBIT-Format

Abschnitt 2

Bitoperationen

2.1 Einleitung

Die Bits in einem PDP-10 Maschinenwort werden von links nach rechts von 0 bis 35 durchnumeriert. Die Routinen zur Behandlung einzelner Bits in der BWLLIB behandeln nicht nur die Operationen auf einzelnen Worten, sondern koennen Bits in Vektoren beliebiger Laenge setzen, loeschen und deren Wert abliefern. Zur Darstellung eines Bit-Vektors der Laenge N Bits werden (N-1)/36+1 Maschinenworte benoetigt, d.h. um einen Bit-Vektor der Laenge N muessen folgende Feldlaengen in FORTRAN definiert werden :

Typen INTEGER, LOGICAL, REAL : (N-1)/36+1

DOUBLE PRECISION, COMPLEX : (N-1)/72+1

Anmerkung

Besonders zu beachten ist, dass die Laenge des Bit-Vektors gleich der um 1 erhoehten groessten vorkommenden Bitnummer ist, da die Numerierung der Bits bei 0 beginnt.

2.2 B I T

Zum Setzen und Loeschen einzelner Bits dient die Routine BIT.

Aufruf: CALL BIT(FELD,NB,IV)

Parameter :

FELD	[V,*(*)]	Bit-Vektor. Die Variable FELD kann beliebigen Typs und beliebiger Dimension sein. (siehe Einleitung)
NB	[A,I]	Nummer des zu setzenden/loeschenden Bits. Die Numerierung beginnt mit 0.
IV	[A,I]	Das Bit NB wird geloescht, wenn IV=0, sonst gesetzt.

2.3 I B I T und J B I T

Zur Pruefung von Bits enthaelt die BWLLIB zwei Funktionen, IBIT und JBIT.

Aufruf: I=IBIT(FELD,NB)

J=JBIT(FELD,NB)

Parameter :

FELD	[V,*(*)]	Bit-Vektor. Die Variable FELD kann beliebigen Typs und beliebiger Dimension sein. (siehe Einleitung)
NB	[A,I]	Nummer des Bits, dessen Wert gepruefft werden soll. Die Numerierung beginnt mit 0.
I	[V,I]	Der von der Funktion IBIT zurueckgegebene Wert enthaelt entweder 0 oder den stellenrichtigen Wert des gesetzten Bits (also nicht 1), sodass bei Abfragen, ob Bit NB gesetzt ist, I.EQ.0 bzw I.NE.0 zu verwenden ist.
J	[V,I]	Der von der Funktion JBIT zurueckgegebene Wert enthaelt entweder 0 oder 1.

2.4 N O B I T S

Die Subroutine NOBITS erlaubt ein Auszaehlen der gesetzten Bits in einem oder mehreren aufeinanderfolgenden Maschinenworten.

Aufruf: CALL NOBITS(VAR,NW,NB)

Parameter :

VAR	[*,*(*)]	beliebige Variable oder Konstante (beliebiger Dimension) in/ab der die Bits gezaehlt werden sollen. (siehe Einleitung)
NW	[A,I]	Anzahl der Maschinenworte, in denen die Bits gezaehlt werden sollen. Bei einfachen Variablen oder Konstanten ist 1 anzugeben.
NB	[V,I]	Anzahl der gesetzten Bits in dem getesteten Bereich. NB=0, wenn keine Bits gesetzt waren oder NW<1 angegeben wurde. Die Anzahl der nicht gesetzten Bits ergibt sich aus 36*NW-NB

Abschnitt 3

Halbwortbehandlung

3.1 Einleitung

Die PDP-10 unterstuetzt die Behandlung von Halbworten zu 18 Bit, so dass in jedem Halbwort Zahlen zwischen 0 und 2**18-1 oder -2**17 und 2**17-1 bei Betrachtung von Vorzeichen dargestellt werden koennen. Mit Hilfe der in der Bibliothek BWLLIB enthaltenen Routinen koennen Halbworte kopiert werden, so dass ein Datentyp INTEGER*2 simuliert werden kann. Die Funktion der einzelnen Routinen H<x><y><z> ist aus dem Namen ersichtlich.

Aufruf: CALL H<x><y><z>(I,J)

Parameter :

I	[A,I]	Wort, aus dem ein Halbwort gemaess <x> kopiert werden soll. Der Inhalt von I bleibt unveraendert.
J	[V,I]	Wort, in das ein Halbwort in die Haelfte <y> kopiert werden soll.

Bedeutung von <x>,<y> und <z>

<x>::= L | R
Haelfte von I, der das zu uebertragende Halbwort entnommen wird. (L=links, R=rechts)

<y>::= L | R
Haelfte von J, in die das Halbwort kopiert wird. (L=links, R=rechts)

<z>::= <empty> | E | O | Z
Aktion, die auf der nicht betroffenen Haelfte von J ausgefuehrt werden soll.

<empty> (leere Zeichenfolge) : Die nicht betroffenen Haelfte von J bleibt unveraendert.

E : alle Bits in der nicht betroffenen Haelfte von J erhalten den Wert des Vorzeichens (ganz links stehendes Bit) des uebertragenen Halbwortes. Diese Funktion ist von Bedeutung, wenn in dem Halbwort positive und negative Zahlen vorkommen koennen und der Wert in einem Ganzwort (36 Bit) dargestellt werden soll.

O : alle Bits des nicht betroffenen Halbwortes werden auf 1 gesetzt, d.h. es erhaelt den Wert -1

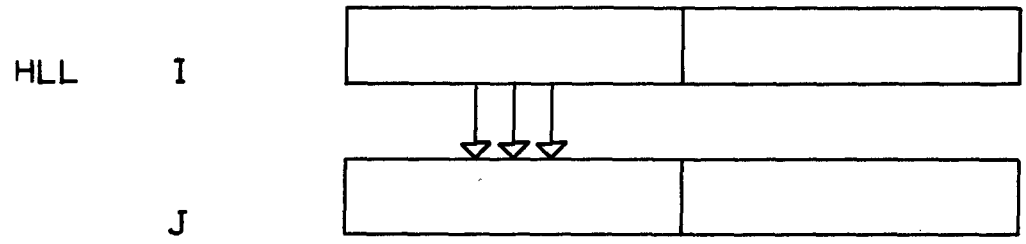
Z : alle Bits des nicht betroffenen Halbwortes werden auf 0 gesetzt.

Folgende Routinen zum Kopieren von Halbworten sind in der BWLLIB enthalten :

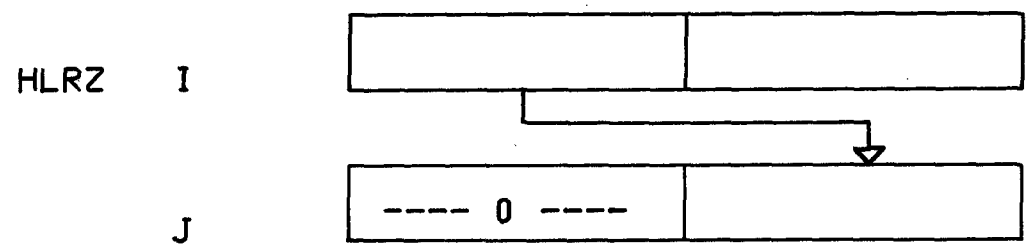
HLL	HLLE	HLLO	HLLZ
HLR	HLRE	HLRO	HRLZ
HRL	HRLE	HRLO	HRLZ
HRR	HRRE	HRRO	HRRZ

3.2 Wirkung der Halbwortroutinen (Auswahl)

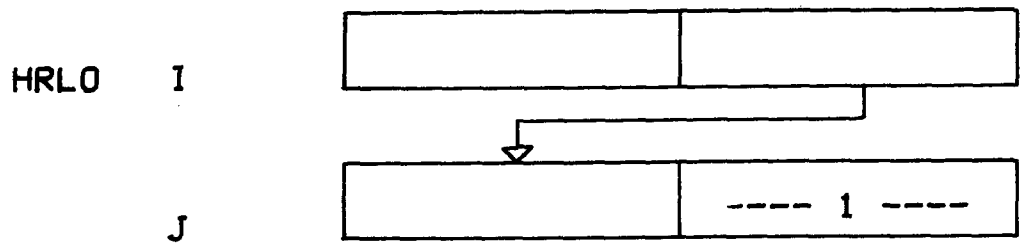
HLL kopiert von der linken in die linke Haelfte, die rechte Haelfte von J bleibt unveraendert.



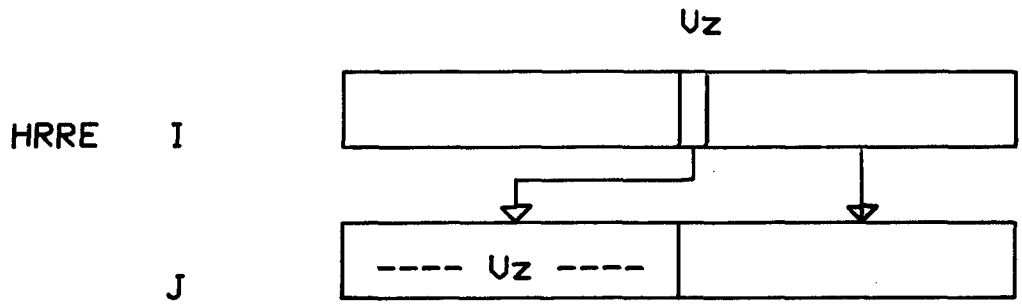
HLRZ kopiert die linke Haelfte von I in die rechte von J. Die linke Haelfte von J wird geloescht.



HRLO kopiert die rechte Haelfte von I in die linke von J. Alle Bits der rechten Haelfte von J werden auf 1 gesetzt.



HRRE kopiert die rechte Haelfte von I in die rechte Haelfte von J. Alle Bits der linken Haelfte von J erhalten der Wert des Vorzeichenbits der uebertragenen Haelfte, d.h. der Inhalt der uebertragenen Haelfte wird auf ein Ganzwort ausgedehnt.



Abschnitt 4

Textbearbeitung

4.1 C 6 T 7 und C 7 T 6

Zur Konversion von ASCII- in SIXBIT-Texte dient die Subroutine C7T6, die Umwandlung von SIXBIT nach ASCII wird in C6T7 durchgefuehrt. Eine genaue Darstellung der Textformate befindet sich in Abschnitt 1.4.

Aufruf: CALL C6T7(SIX,ASC,N,ERR) [SIXBIT -> ASCII]
CALL C7T6(ASC,SIX,N,ERR) [ASCII -> SIXBIT]

Parameter :

SIX [V,*(I6)] Feld beliebigen Typs, das den SIXBIT-Text enthaelt. Die Laenge des Feldes muss mindestens I6=(N-1)/6+1 betragen (bei DOUBLE PRECISION und COMPLEX mindestens I6=(N-1)/12+1). Im Aufruf von C6T7 darf SIX auch als Konstante oder Ausdruck angegeben werden.

ASC [V,*(I7)] Feld beliebigen Typs, das den ASCII-Text aufnimmt. Die Laenge von ASC muss mindestens I7=(N-1)/5+1 (bei DOUBLE PRECISION und COMPLEX mindestens I7=(N-1)/10+1) betragen. Im Aufruf von C7T6 kann ASC auch als Konstante oder Ausdruck angegeben werden.

N [A,I] Anzahl der zu konvertierenden Zeichen. N kann positiv oder negativ angegeben werden, wobei im ersten Fall die Restzeichen im letzten Wort, das konvertierten Text aufnimmt, unveraendert bleiben. Ein negatives N bewirkt, das IABS(N) Zeichen umgewandelt werden und das letzte Wort, das konvertierten Text aufnimmt, mit Zwischenraeumen (Blank) aufgefuellt wird. N=0 wirkt wie N=-1.

ERR [V,I]

Fehlercode, enthaelt 0, wenn die Konversion fehlerfrei durchgefuehrt werden konnte (bei C7T6 immer), sonst einen der folgenden Werte:

1 : der ASCII-Text enthielt Kleinbuchstaben, die vor der Konversion in Grossbuchstaben umgewandelt wurden.

2 : Der ASCII-Text enthielt ein Zeichen mit einem Wert < 32, das durch einen Zwischenraum ersetzt wurde.

3 : beide Fehler

4.2 P A C K und U N P A C K

Die Unterprogramme packen (PACK) und entpacken (UNPACK) ASCII-Texte. Ein gepackter Text entspricht einem im Format A5 eingelesenen Text. Ein ungepackter Text entspricht einem Vektor, dessen Elemente jeweils ein Zeichen enthalten. Die Zeichen werden hier durch ihren ASCII-Wert als INTEGER-Zahl dargestellt.

Aufruf: CALL PACK(TP,TU,IA,IE)
CALL UNPACK(TP,TU,IA,IE)

Parameter :

TP	[V,I(N)]	gepackter Text.
TU	[V,I(5*N)]	entpackter Text.
IA	[A,I]	Index des ersten zu (ent-) packenden Zeichens in TU.
IE	[A,I]	Index des letzten zu (ent-) packenden Zeichens in TU.

Beispiele:

```
TEXT(1)='ABCDE'
CALL UNPACK(TEXT1,TEXT2,1,5)
```

TEXT2 enthaelt in den ersten Elementen die Werte 65,66,67,68 und 69.

```
CALL PACK(TEXT1,TEXT2,3,4)
```

TEXT1(1) enthaelt jetzt 'CDCDE' TEXT1(2) enthaelt jetzt 'CDxxx'. TEXT1(1) bleibt unveraendert.

4.3 D E C F I L

DECFIL dekodiert eine in einem Strins enthaltene Standard-Filespezifikation der Form <dev>:<file>.<ext>[<ppn>] so, dass die Uebernahme der Parameter in ein OPEN-Statement direkt moeglich ist. Darueberhinaus koennen Behandlunsen von Standardseraeten und -Extentions durchsefuehrt werden.

Aufruf : CALL DECFIL(TEXT,LT,OPT,EXT,DEV,FIL,PPN,ERR)

Parameter :

TEXT [A,*(L)] Feld, das den zu dekodierenden Strins enthaelt. Die Laense L ersibt sich aus dem Parameter LT (siehe dort).

LT [A,I] Laense des zu dekodierenden Strinss in Maschinenworten (LT<0 => L=IABS(LT)) oder in Symbolen (LT>0 => L=LT*5).

OPT [A,I] Optionen fuer die Dekodieruns in Form einer dreistellisen Dezimalzahl ABC mit folsenden Bedeutunsen :

- A Behandluns der Extension :
- A=0 es erfolst keine Behandluns der Extension
- A=1 enthaelt die Filespezifikation in TEXT keine Extension, d.h. keinen Punkt ".", so wird die im Parameter EXT enthaltenen Standardextension einsesetzt.
- A=2 Die durch EXT sesebene Extension wird immer einsesetzt, unabhaensis davon, ob in TEXT eine Extension vorkommt oder nicht.
- B Standard-Geraetename (-Device), der einsesetzt werden soll, wenn die Spezifikation einen Filenamen, nicht aber einen Geraetenamen enthaelt. (Kodierunsen s.u.)
- C Standard-Geraetename (-Device), der einsesetzt werden soll, wenn die Spezifikation in TEXT weder einen File- noch einen Geraetenamen enthaelt. (Kodierunsen s.u.)

EXT [A,*] Extension im Format A3, die semaess Option A (siehe OPT) einsesetzt werden kann oder soll.

DEV	[V,D]	Geraetename im Format A10, wie er in der Spezifikation gefunden bzw gemaess Optionen B und C eingesetzt wurde.
FIL	[V,D]	Filename und Extension im Format A10. Der Filename umfasst immer sechs Stellen (gsfs mit Zwischenraeumen aufgefuellt), so dass der Punkt zwischen Filenamen und Extension immer an der 7. Stelle in FIL steht.
PPN	[V,I(3)]	PPN im FORTRAN-Format, d.h. PPN(1) = Projektnummer PPN(2) = Programmierernummer PPN(3) = 0
ERR	[V,I]	Fehlercode : 0 kein Fehler 1 Geraetename mehrfach gefunden 2 Filename mehrfach gefunden 3 PPN mehrfach gefunden 4 SFD's koennen nicht behandelt werden (Fehler in der PPN)

Kodierung der Geraetenamen fuer Optionen B und C

0 keine Behandlung

1=<blank>	2=DSK	3=LPT	4=TTY
5=NUL	6=SYS	7=HLP	8=LIB
9=NEW			

Benoetigte Unterprogramme :

DIVIDE	HLRZ	HRRZ	JOBPPN
PACK	UNPACK		

4.4 E N C F I L

ENCFIL kodiert aus den Parametern eines OPEN-Statements (FORTRAN-Filespezifikation) eine Filespezifikation im Standard-PDP10-Format.

Parameter :

TEXT	[V,*(7)]	erzeuste Filespezifikation
LT	[V,I]	Laenge des erzeugten Textes
DEV	[A,D]	Geraetename im Format A10
FIL	[A,D]	Filename im Format A10
PPN	[A,I(3)]	PPN im FORTRAN-Format : PPN(1) = Projektnummer PPN(2) = Programmierernummer PPN(3) = 0 (fuer ENCFIL beliebig)
ERR	[V,I]	Fehlercode (in der vorliegenden Version immer 0)

benoetigte Unterprogramme

DIVIDE PACK UNPACK

4.5 T S O R T 0

In der PDP-10 werden ASCII-Texte in einem Wort linksbuendig dargestellt. Dies hat zur Folge, dass sich die Wortinhalte, als Zahl interpretiert, sowohl als positive wie auch als negative Werte darstellen, so dass ein Sortieren mit gewissen Problemen behaftet ist. TSORT0 sortiert ASCII-Texte entsprechend den ASCII-Werten der Zeichen. Dabei wird das vom Text nicht benutzte Bit ausgenutzt. Dabei werden die Wortinhalte um ein Bit nach rechts verschoben, so dass alle Texte nur positive Werte annehmen. Aus diesem Grund kann TSORT0 nicht fuer die Sortierung von Zahlen oder SIXBIT-Texten eingesetzt werden.

Aufruf: CALL TSORT0(TEXT,IND,HELP,N)

Parameter

TEXT	[V,*(N)]	Vektor, der N zu sortierende Texte im Format A5 enthaelt. TEXT darf vom Typ INTEGER, REAL oder LOGICAL sein. Der Inhalt wird durch den Aufruf nicht veraendert.
IND	[V,I(N)]	Indexvektor, der die Reihenfolge der Texte nach der Sortierung festlegt. INDEX(1) zeigt auf den ersten Text, INDEX(2) auf den zweiten Text in der sortierten Reihenfolge u.s.w.
HELP	[V,*(N)]	Hilfsvektor fuer die Sortierung
N	[A,I]	Anzahl der zu sortierenden Texte. (Laenge der Felder TEXT, IND und HELP)

Abschnitt 5

Terminaloperationen

5.1 C L E A N

CLEAN positioniert den Cursor (die Schreibmarke) in die linke obere Ecke des Bildschirms und loescht ihn. CLEAN kann nur auf Bildschirmen der Typen VT05 und VT52 verwendet werden (siehe auch CLEAR).

Aufruf: CALL CLEAN

Anmerkung

CLEAN sollte in neuen Programmen nicht mehr verwendet werden, da diese Routine in einer neuen Version in der BWLLIB nicht mehr enthalten sein wird. Aus dem gleichen Grunde sollten bestehende Programme auf CLEAR umgestellt werden.

5.2 C L E A R

CLEAR positioniert den Cursor (die Schreibmarke) in die linke obere Ecke des Bildschirms und loescht ihn. Auf Teletypes (Fernschreibern) werden drei Leerzeilen ausgegeben. Der Terminaltyp kann mit Hilfe der Routine TRMTYP bestimmt werden.

Aufruf: CALL CLEAR(TYP)

Parameter

TYP	[A,I]	Terminaltyp 0 = VT05B 1 = VT52 2 = Eurotech 3 = Teletype (Fernschreiber)

5.3 C U R P O S

CURPOS positioniert den Cursor (die Schreibmarke) an eine beliebige Stelle des Bildschirms. Der Terminaltyp kann mit Hilfe der Routine TRMTYP bestimmt werden. Auf Teletypes bleibt CURPOS ohne Wirkung.

Aufruf: CALL CURPOS(T,Z,S,ERR)

Parameter

T	[A,I]	Terminaltyp 0 = VT05 1 = VT52 2 = Eurotech 3 = Teletype (Fernschreiber)
Z	[A,I]	Zeilennummer auf dem Bildschirm (obere Zeile hat die Nummer 1)
S	[A,I]	Spaltennummer auf dem Bildschirm (linke Spalte hat die Nummer 1)
ERR	[V,I]	Fehlercode

Anmerkung

Werden fehlerhafte Werte in den Parametern angegeben, so erfolgt keine Positionierung.

5.4 T R M T Y P

Mit Hilfe der Routine TRMTYP kann der Terminaltyp bestimmt werden. Da eine direkte Pruefung des Typs ausser bei VT52-Bildschirmgeraeten nicht moeglich ist, wird der Typ anhand der Leitung bestimmt, ueber die das Terminal an der PDP-10 angeschlossen ist. Aus diesem Grunde kann die Routine u.U. zu nicht befriedigenden Ergebnissen fuehren.

Aufruf : CALL TRMTYP(ITYP)

Parameter :

ITYP [V,I] Terminaltyp
0 = VT05B
1 = VT52
2 = Eurotech
3 = Fernschreiber (Teletype)

Anmerkungen:
Waehrend der Ausfuehrung von TRMTYP wird der Terminal-Input-Buffer geloescht, d.h. alle vor dem Aufruf eingegeben Zeichen gehen verloren.
Auf dem Terminal koennen, bedingt durch den Test auf VT52, einige Zeichen erscheinen. Wir sind bemueht, diesen Mangel zu beheben.

5.5 E C H O

ECHO schaltet das Echo ein, d.h. die auf der Tastatur eingegeben Zeichen erscheinen auf dem Bildschirm (siehe auch NOECHO). Die Wirkung von ECHO entspricht der des Monitor-Befehls ".SET TTY ECHO".

Aufruf: CALL ECHO

5.6 N O E C H O

NOECHO schaltet das Echo ab, d.h. die auf der Tastatur eingegebenen Zeichen erscheinen nicht auf dem Bildschirm. Die Wirkung von NOECHO entspricht der des Monitor-Befehls ".SET TTY NO ECHO".

Aufruf : CALL NOECHO

5.7 I N S Y M

INSYM liest ein Zeichen vom Terminal ein und liefert den ASCII-Wert des Zeichens ab. Die Besonderheit dieser Routine liest darin, dass ein abschliessendes <RETURN> nicht erforderlich ist, und dass INSYM auch nicht auf eine Eingabe wartet.

Aufruf : CALL INSYM(CHAR)

Parameter

CHAR	[V,I]	enthaelt nach dem Aufruf den ASCII-Wert des einselesenen Zeichen, oder -1, wenn keine Eingabe am Terminal erfolgt war.

5.8 O U T S Y M

OUTSYM sibt ein einzelnes Zeichen auf dem Bildschirm an der Cursor-Position aus.

Aufruf : CALL OUTSYM(CHAR)

Parameter

CHAR [A,I] ASCII-Wert des auszusebenden Zeichen.

Abschnitt 6

Betriebssystemaufrufe

6.1 A D D R

Die Funktion ADDR ist im eigentlichen Sinne kein Betriebssystemaufruf. Sie liefert lediglich die Speicheradresse der als Parameter angegebenen Variablen, ein Datum, das fuer Betriebssystemaufrufe haeufig benoetigt wird. ADDR muss im aufrufenden Programm als INTEGER definiert werden.

Aufruf : IADR = ADDR(VAR)

Parameter :

VAR	[V,*(*)]	Variable deren Adresse im Speicher bestimmt werden soll.
IADR	[V,I]	von der Funktion zurueckgegebener Wert, d.h. die Adresse der Variablen. Bei Vektoren und Matrizen enthaelt IADR die Adresse des ersten Elementes.

6.2 M C A L L

MCALL erlaubt den direkten Aufruf von Betriebssystemfunktionen (Monitor Calls). Der Inhalt des Akkumulators wird im aufrufenden Programm aufgebaut und als Parameter uebergeben. Fuer einige Monitor Calls sind Bereiche als Parameter notwendig, deren Adresse im Akkumulator uebergeben wird. Sie sind ebenfalls im rufenden Programm einzurichten und mit Werten zu versorgen. Die Adresse der Bereiche kann mit ADDR bestimmt werden.

Die Wirkung und der Aufruf der einzelnen Betriebssystemfunktionen sind dem MONITOR CALLS MANUAL zu entnehmen. Als Akkumulator wird immer AC 1 verwendet. Dies ist bei einzelnen Monitor Calls zu beachten (z.B. EXIT). BWLLIB enthaelt einige Routinen, die haeufiger benoetigte Betriebsysteme direkt ansprechen.

Aufruf : CALL MCALL(UNR,ACI,ACO,ERR)

Parameter

UNR	[A,I]	Nummer des Monitor Calls
ACI	[A,I]	Inhalt des Akkumulators vor dem Aufruf. Fuer den Aufruf wird immer AC1 verwendet.
ACO	[V,I]	Inhalt des Akkumulators AC1 nach dem Aufruf, i.a. der vom Monitor Call zurueckgegebene Wert.
ERR	[V,I]	Indikator fuer den aus dem Monitor Call genommenen Ruecksprung. 0 : skip return (i.a. der Ruecksprung nach fehlerfreier Ausfuehrung). 1 : non skip return (i.a. der Ruecksprung nach einem Fehler). Monitor Calls, die nur einen Ruecksprung haben, liefern in ERR immer den Wert -1.

6.3 D A T E 1

DATE1 liefert das Datum des jeweiligen Tages.

Aufruf: CALL DATE1(IT,MO,JA)

Parameter:

IT	[V,I]	Tag des Monats
MO	[V,I]	Monat (1=Jan,2=Feb,...,12=Dez)
JA	[V,I]	Jahr (die letzten beiden Stellen)

6.4 R T I M

RTIM liefert die verbrauchte Rechenzeit seit Losin in Millisekunden.

Aufruf: CALL RTIM(ICPU)

Parameter:

ICPU [V,I] CPU-Zeit in Millisekunden.

6.5 R T I M E

RTIME liefert die verbrauchte Rechenzeit seit Login in Millisekunden und die Tageszeit seit Mitternacht, ebenfalls in Millisekunden.

Aufruf: CALL RTIME(ICPU,IDAY)

Parameter:

ICPU	[V,I]	CPU-Zeit in Millisekunden.
IDAY	[V,I]	Tageszeit in Millisekunden.

Anmerkungen

1. Bei der Tageszeit betraegt die Genauigkeit trotz der Angabe in Millisekunden nur 1/100 Sekunde.
2. Ein Tag hat 86400000 Millisekunden.

6.6 J O B N R

JOBNR liefert die Jobnummer des aufrufenden Jobs.

Aufruf: J=JOBNR(DMY)

Parameter:

J	[V,I]	Jobnummer
DMY	[A,*]	beliebiser Wert oder Name

6.7 J O B P P N

JOBPPN liefert die PPN (Benutzernummer) des aufrufenden Jobs im Halbwortformat.

Aufruf: J=JOBPPN(DMY)

Parameter:

J	[V,I]	Benutzernummer im Halbwortformat, d.h. die Projektnummer in der linken und die Programmierernummer in der rechten Haelfte(1).
DMY	[A,*]	beliebiger Wert oder Name

---

(1) Bei der Ausgabe der Benutzernummer ist, um zu bekannten Ergebnissen zu kommen, ein Format fuer oktale Darstellung zu verwenden.

6.8 J O B T T Y

JOBTTY liefert den Namen des Terminals (TTYnnn), von dem der Job gestartet wurde, in SIXBIT. Ist der Job detached, so enthaelt die linke Haelfte des Wortes 3 Zwischenraeume (=0 in SIXBIT), in der rechten Haelfte steht dann in SIXBIT die Nummer des Terminals, ueber das der Job zuletzt attached war.

Aufruf: J=JOBTTY(DMY)

Parameter:

J	[V,I]	Name des Terminals in SIXBIT
DMY	[A,*]	beliebiger Wert oder Name

6.9 E X I T 1

EXIT1 unterbricht das Programm, der Job kehrt auf Monitor-Level zurueck, ohne etwa eroeffnete Files abzuschliessen. Die Wirkung entspricht einer Unterbrechung mit <ctrl-C>. Das Programm kann auf Monitor-Level durch .CON bzw. .CCON fortgesetzt werden.

Aufruf: CALL EXIT1

Parameter:

keine.

6.10 R U N

RUN erlaubt das Aufrufen von Folgeprogrammen direkt aus FORTRAN-Programmen heraus. Vom Programm eroeffnetet Files muessen vor dem Aufruf explizit abgeschlossen werden (CLOSE-Anweisung), da sonst erzeugte Ausgaben wieder geloescht werden. Optional kann eine Meldung auf das Terminal und/oder ueber einen FORTRAN-Kanal ausgegeben werden. Das aufgerufene Programm muss in geladener Form (.SAV- oder .EXE-File) auf einer Platte vorhanden sein.

Aufruf: CALL RUN(DEV,FIL,PPN,OFF,UNIT)

Parameter :

DEV	[A,D]	Device-Name. Hier muss eine Platte DSKx oder ein Ersatzdevice (SYS, LIB usw) angegeben werden.
FIL	[A,D]	Filename des aufzurufenden Programms ohne Extension
PPN	[A,I(3)]	Directory, in der der Programmfile steht PPN(1)=Projektnummer, PPN(2)=Programmierernummer, PPN(3)=0.
OFF	[A,I]	Offset, d.h. die Adresse relativ zur Startadresse des aufgerufenen Programms, an der dessen Ausfuehrung begonnen werden soll. Ein Wert ungleich Null sollte nur bei entsprechender Behandlung durch das aufgerufenen Programm angegeben werden (siehe auch Anmerkungen).
UNIT	[A,I]	Steuerung der Ausgabe der Meldung.

Ausgabe der Meldung (Parameter UNIT)

Die Ausgabe einer Meldung
<Prog> STARTING (WITH OFFSET <off>) <filespec>
ist optional und kann auf dem Terminal und/oder in eine Datei erfolgen. <Prog> zeigt den Namen des aufrufenden Programms an. Das aufgerufene Programm wird aus dem File <filespec> geladen und mit einem Offset <off> gestartet. Enthaelt OFF den Wert Null, so

wird der Teil "(WITH OFFSET <off>)" unterdrueckt. Zur Steuerung der Aussabe kann der Parameter UNIT folsende Werte annehmen :

0 : es wird keine Melduns ausseseben
999 : die Melduns erscheint auf dem Terminal (TTY:)
positiv : die Melduns wird in die auf Unit UNIT eroeffnete Datei seschrieben.
negativ : die Melduns erscheint auf dem Terminal und in der auf Kanal UNIT eroeffneten Datei.

A c h t u n s !

Die Datei, in die die Melduns seschrieben werden soll, darf natuerlich noch nicht abseschlossen sein. Nach der Aussabe fuehrt RUN die Anweisuns CLOSE(UNIT=IABS(UNIT)) aus.

Benoetiste Unterprosramme :

C6T7 C7T6 ENCFIL HRLZ HRR MCALL RUNP1

6.11 RUNP1

RUNP1 erlaubt das Aufrufen von Folgeprogrammen direkt aus FORTRAN-Programmen heraus. Vom Programm eroeffnetet Files muessen vor dem Aufruf explizit abgeschlossen werden (CLOSE-Anweisung), da sonst erzeugte Ausgaben wieder geloescht werden. Das aufgerufene Programm muss in geladener Form (.SAV- oder .EXE-File) auf einer Platte vorhanden sein.

Aufruf: CALL RUNP1(IRUN,IOFF)

Parameter

IRUN	[V,I(6)]	Spezifikation des aufzurufenden Programms.
IOFF	[A,I]	Adresse relativ zur normalen Startadresse, von der ab das aufgerufene Programm gerechnet werden soll.

Aufbau des Feldes IRUN:

IRUN(1)	Devicename in SIXBIT
IRUN(2)	Filename in SIXBIT
IRUN(3)	0
IRUN(4)	0
IRUN(5)	PPN im internen (Halbwort-) Format oder 0, wenn das Programm in der eigenen Directory steht.
IRUN(6)	0

Anmerkung :

Der Modul RUNP1 wird von RUN benoetigt, das den gleichen Zweck bei vereinfachter Parameteruebergabe (Namen in ASCII) erfuellt.

6.12 S L E E P

SLEEP desaktiviert den aufrufenden Job fuer eine bestimmte Zeit.

Aufruf: CALL SLEEP(ITIM)

Parameter:

ITIM	[A,I]	Dauer der Desaktivierung in Sekunden. (max. 82)

Abschnitt 7

Numerik

7.1 D I V I D E

Das Unterprogramm DIVIDE liefert von einer Integer-Division in einem Aufruf den Quotienten und den Rest.

Aufruf: CALL DIVIDE(I1,I2,IQ,IR)

Parameter :

I1	[A,I]	Dividend
I2	[A,I]	Divisor
IQ	[V,I]	Quotient
IR	[V,I]	Rest der Division

Wirkung :

Der Aufruf hat die Wirkung der folgenden Befehlsfolge :

```
IQ = I1 / I2
IR = MOD(I1,I2)
```

Anmerkungen :

Aufrufe der Art CALL DIVIDE(K,L,K,L) sind zugelassen (eine direkte Uebertragung in die FORTRAN-Befehlsfolge ist nicht moeglich). Der Aufruf CALL DIVIDE(I,J,I,I) liefert in I den Rest der Division I/J.

7.2 Z U F A L L

ZUFALL berechnet mit Hilfe des Generators RAN des FORTRAN-Systems Zufallszahlen unterschiedlicher Verteilungen aus einem frei waehlbaren Intervall.

Aufruf : ZZ=ZUFALL(XU,XO,XE,IT,IE)

Parameter :

XU	[A,R]	untere Grenze des Intervalls, aus dem die Zufallszahl gezogen werden soll.
XO	[A,R]	obere Grenze des Intervalls
XE	[A,R]	Erwartungswert der Dreiecksverteilung. Es muss XU<XE<XO erfuellt sein. Bei Gleich- und Normalverteilung ist dieser Wert ohne Bedeutung.
IT	[A,I]	Verteilungstyp 1 = Rechteck- (Gleich-) Verteilung 2 = Normalverteilung 3 = Dreiecksverteilung mit Erwartungswert XE
IE	[V,I]	Fehlercode 0 = kein Fehler 1 = unzulaessige Intervallgrenzen (i.a. XU>XO) 2 = Erwartungswert liegt ausserhalb des Intervalls (nur bei Dreiecksverteilung) 3 = unzulaessiger Verteilungstyp
ZZ	[V,R]	Ergebnis der Funktion Zufall

Anmerkungen

Bei der Verwendung der Normalverteilung (IT=2) ist zu beachten, dass fuer die Berechnung des Funktionswertes 12 Zahlen gezogen werden.

Eine Wiederholung der Zufallsreihe durch Retten und Setzen des Startwertes des Generators RAN ist durch die Systemroutinen SAVRAN und SETRAN moeglich.

Anhang A

Codetabelle

Die Tabelle auf den folgenden Seiten enthaelt die SIXBIT- und ASCII-Codes aller vorkommenden Zeichen. Die erste Spalte (Char) gibt das Zeichen an, bzw bei den Control-Characters eine Abkuerzung der Funktion (Erlaeuterungen am Ende der Tabelle). Die naechsten beiden Spalten den SIXBIT-Code des Zeichens (sofern definert) in dezimaler und oktaler Darstellung. Die restlichen sieben Spalten betreffen den ASCII-Code, wobei die ersten beiden den Zeichencode in dezimaler und oktaler Darstellung angeben. Die letzten fuenf Spalten "Position 1" bis "Position 5" geben den Inhalt eines Wortes in oktaler Darstellung an, wenn ausschliesslich das Zeichen an entsprechender Stelle in dem Wort steht.

Anwendung :

Mit Hilfe der Tabelle koennen Zeichenkonstanten, die nicht direkt im Programm angegeben werden koennen, weil sie nicht druckbare Zeichen enthalten, erzeugt werden. Dazu addiert man die Konstanten der entsprechenden Zeichen und Positionen und weist die Summe einer Variablen in einem DATA-Statement zu.

Beispiel :

Beim Auftreten eines Fehlers soll der Text <BEL>ERR<BEL> ausgegeben werden. (<BEL>=<ctrl-G>, ASCII-Code 7)

<BEL>	an Position 1	034000000000
E	an Position 2	002120000000
R	an Position 3	000012200000
R	an Position 4	000000051000
<BEL>	an Position 5	000000000014

Summe		036132251014

Das entsprechende Unterprogramm zu Ausgabe der Meldung koennte dann folgendermassen aussehen:

```
        SUBROUTINE ERR
        DATA MSG/'036132251014/
        TYPE 1000,MSG
1000    FORMAT(1X,A5)
        RETURN
        END
```

** Codetabelle: ASCII und SIXBIT mit interner Darstellung von ASCII-Symbolen ** Seite 1 **

Char	-SIXBIT-		-ASCII-					
	dez	okt	dez	okt	Position 1	Position 2	Position 3	Position 4	Position 5
NUL			0	000	000000000000	000000000000	000000000000	000000000000	000000000000
SOH			1	001	004000000000	000020000000	000000100000	000000000400	000000000002
STX			2	002	010000000000	000040000000	000000200000	000000001000	000000000004
ETX			3	003	014000000000	000060000000	000000300000	000000001400	000000000006
EOT			4	004	020000000000	000100000000	000000400000	000000002000	000000000010
ENQ			5	005	024000000000	000120000000	000000500000	000000002400	000000000012
ACK			6	006	030000000000	000140000000	000000600000	000000003000	000000000014
BEL			7	007	034000000000	000160000000	000000700000	000000003400	000000000016
BS			8	010	040000000000	000200000000	000001000000	000000004000	000000000020
TAB			9	011	044000000000	000220000000	000001100000	000000004400	000000000022
LF			10	012	050000000000	000240000000	000001200000	000000005000	000000000024
VT			11	013	054000000000	000260000000	000001300000	000000005400	000000000026
FF			12	014	060000000000	000300000000	000001400000	000000006000	000000000030
CR			13	015	064000000000	000320000000	000001500000	000000006400	000000000032
SO			14	016	070000000000	000340000000	000001600000	000000007000	000000000034
SI			15	017	074000000000	000360000000	000001700000	000000007400	000000000036
DLE			16	020	100000000000	000400000000	000002000000	000000010000	000000000040
DC1			17	021	104000000000	000420000000	000002100000	000000010400	000000000042
DC2			18	022	110000000000	000440000000	000002200000	000000011000	000000000044
DC3			19	023	114000000000	000460000000	000002300000	000000011400	000000000046
DC4			20	024	120000000000	000500000000	000002400000	000000012000	000000000050
NAK			21	025	124000000000	000520000000	000002500000	000000012400	000000000052
SYN			22	026	130000000000	000540000000	000002600000	000000013000	000000000054
ETB			23	027	134000000000	000560000000	000002700000	000000013400	000000000056
CAN			24	030	140000000000	000600000000	000003000000	000000014000	000000000060
EM			25	031	144000000000	000620000000	000003100000	000000014400	000000000062
SUB			26	032	150000000000	000640000000	000003200000	000000015000	000000000064
ESC			27	033	154000000000	000660000000	000003300000	000000015400	000000000066
FS			28	034	160000000000	000700000000	000003400000	000000016000	000000000070
GS			29	035	164000000000	000720000000	000003500000	000000016400	000000000072
RS			30	036	170000000000	000740000000	000003600000	000000017000	000000000074
US			31	037	174000000000	000760000000	000003700000	000000017400	000000000076

** Codetabelle: ASCII und SIXBIT mit interner Darstellung von ASCII-Symbolen ** Seite 2 **

Char	-SIXBIT-		-ASCII-						
	dez	okt	dez	okt	Position 1	Position 2	Position 3	Position 4	Position 5
	0	000	32	040	200000000000	001000000000	000004000000	000000020000	000000000100
!	1	001	33	041	204000000000	001020000000	000004100000	000000020400	000000000102
"	2	002	34	042	210000000000	001040000000	000004200000	000000021000	000000000104
#	3	003	35	043	214000000000	001060000000	000004300000	000000021400	000000000106
$	4	004	36	044	220000000000	001100000000	000004400000	000000022000	000000000110
%	5	005	37	045	224000000000	001120000000	000004500000	000000022400	000000000112
&	6	006	38	046	230000000000	001140000000	000004600000	000000023000	000000000114
'	7	007	39	047	234000000000	001160000000	000004700000	000000023400	000000000116
(	8	010	40	050	240000000000	001200000000	000005000000	000000024000	000000000120
)	9	011	41	051	244000000000	001220000000	000005100000	000000024400	000000000122
*	10	012	42	052	250000000000	001240000000	000005200000	000000025000	000000000124
+	11	013	43	053	254000000000	001260000000	000005300000	000000025400	000000000126
,	12	014	44	054	260000000000	001300000000	000005400000	000000026000	000000000130
-	13	015	45	055	264000000000	001320000000	000005500000	000000026400	000000000132
.	14	016	46	056	270000000000	001340000000	000005600000	000000027000	000000000134
/	15	017	47	057	274000000000	001360000000	000005700000	000000027400	000000000136
0	16	020	48	060	300000000000	001400000000	000006000000	000000030000	000000000140
1	17	021	49	061	304000000000	001420000000	000006100000	000000030400	000000000142
2	18	022	50	062	310000000000	001440000000	000006200000	000000031000	000000000144
3	19	023	51	063	314000000000	001460000000	000006300000	000000031400	000000000146
4	20	024	52	064	320000000000	001500000000	000006400000	000000032000	000000000150
5	21	025	53	065	324000000000	001520000000	000006500000	000000032400	000000000152
6	22	026	54	066	330000000000	001540000000	000006600000	000000033000	000000000154
7	23	027	55	067	334000000000	001560000000	000006700000	000000033400	000000000156
8	24	030	56	070	340000000000	001600000000	000007000000	000000034000	000000000160
9	25	031	57	071	344000000000	001620000000	000007100000	000000034400	000000000162
:	26	032	58	072	350000000000	001640000000	000007200000	000000035000	000000000164
;	27	033	59	073	354000000000	001660000000	000007300000	000000035400	000000000166
<	28	034	60	074	360000000000	001700000000	000007400000	000000036000	000000000170
=	29	035	61	075	364000000000	001720000000	000007500000	000000036400	000000000172
>	30	036	62	076	370000000000	001740000000	000007600000	000000037000	000000000174
?	31	037	63	077	374000000000	001760000000	000007700000	000000037400	000000000176

Char	-SIXBIT-		-ASCII-						
	dez	okt	dez	okt	Position 1	Position 2	Position 3	Position 4	Position 5
@	32	040	64	100	400000000000	002000000000	000010000000	000000040000	000000000200
A	33	041	65	101	404000000000	002020000000	000010100000	000000040400	000000000202
B	34	042	66	102	410000000000	002040000000	000010200000	000000041000	000000000204
C	35	043	67	103	414000000000	002060000000	000010300000	000000041400	000000000206
D	36	044	68	104	420000000000	002100000000	000010400000	000000042000	000000000210
E	37	045	69	105	424000000000	002120000000	000010500000	000000042400	000000000212
F	38	046	70	106	430000000000	002140000000	000010600000	000000043000	000000000214
G	39	047	71	107	434000000000	002160000000	000010700000	000000043400	000000000216
H	40	050	72	110	440000000000	002200000000	000011000000	000000044000	000000000220
I	41	051	73	111	444000000000	002220000000	000011100000	000000044400	000000000222
J	42	052	74	112	450000000000	002240000000	000011200000	000000045000	000000000224
K	43	053	75	113	454000000000	002260000000	000011300000	000000045400	000000000226
L	44	054	76	114	460000000000	002300000000	000011400000	000000046000	000000000230
M	45	055	77	115	464000000000	002320000000	000011500000	000000046400	000000000232
N	46	056	78	116	470000000000	002340000000	000011600000	000000047000	000000000234
O	47	057	79	117	474000000000	002360000000	000011700000	000000047400	000000000236
P	48	060	80	120	500000000000	002400000000	000012000000	000000050000	000000000240
Q	49	061	81	121	504000000000	002420000000	000012100000	000000050400	000000000242
R	50	062	82	122	510000000000	002440000000	000012200000	000000051000	000000000244
S	51	063	83	123	514000000000	002460000000	000012300000	000000051400	000000000246
T	52	064	84	124	520000000000	002500000000	000012400000	000000052000	000000000250
U	53	065	85	125	524000000000	002520000000	000012500000	000000052400	000000000252
V	54	066	86	126	530000000000	002540000000	000012600000	000000053000	000000000254
W	55	067	87	127	534000000000	002560000000	000012700000	000000053400	000000000256
X	56	070	88	130	540000000000	002600000000	000013000000	000000054000	000000000260
Y	57	071	89	131	544000000000	002620000000	000013100000	000000054400	000000000262
Z	58	072	90	132	550000000000	002640000000	000013200000	000000055000	000000000264
[	59	073	91	133	554000000000	002660000000	000013300000	000000055400	000000000266
\	60	074	92	134	560000000000	002700000000	000013400000	000000056000	000000000270
]	61	075	93	135	564000000000	002720000000	000013500000	000000056400	000000000272
^	62	076	94	136	570000000000	002740000000	000013600000	000000057000	000000000274
_	63	077	95	137	574000000000	002760000000	000013700000	000000057400	000000000276

** Codetabelle: ASCII und SIXBIT mit interner Darstellung von ASCII-Symbolen ** Seite 4 **

Char	-SIXBIT-		-ASCII-					
	dez	okt	dez	okt	Position 1	Position 2	Position 3	Position 4	Position 5
`			96	140	600000000000	003000000000	000014000000	000000060000	000000000300
a			97	141	604000000000	003020000000	000014100000	000000060400	000000000302
b			98	142	610000000000	003040000000	000014200000	000000061000	000000000304
c			99	143	614000000000	003060000000	000014300000	000000061400	000000000306
d			100	144	620000000000	003100000000	000014400000	000000062000	000000000310
e			101	145	624000000000	003120000000	000014500000	000000062400	000000000312
f			102	146	630000000000	003140000000	000014600000	000000063000	000000000314
g			103	147	634000000000	003160000000	000014700000	000000063400	000000000316
h			104	150	640000000000	003200000000	000015000000	000000064000	000000000320
i			105	151	644000000000	003220000000	000015100000	000000064400	000000000322
j			106	152	650000000000	003240000000	000015200000	000000065000	000000000324
k			107	153	654000000000	003260000000	000015300000	000000065400	000000000326
l			108	154	660000000000	003300000000	000015400000	000000066000	000000000330
m			109	155	664000000000	003320000000	000015500000	000000066400	000000000332
n			110	156	670000000000	003340000000	000015600000	000000067000	000000000334
o			111	157	674000000000	003360000000	000015700000	000000067400	000000000336
p			112	160	700000000000	003400000000	000016000000	000000070000	000000000340
q			113	161	704000000000	003420000000	000016100000	000000070400	000000000342
r			114	162	710000000000	003440000000	000016200000	000000071000	000000000344
s			115	163	714000000000	003460000000	000016300000	000000071400	000000000346
t			116	164	720000000000	003500000000	000016400000	000000072000	000000000350
u			117	165	724000000000	003520000000	000016500000	000000072400	000000000352
v			118	166	730000000000	003540000000	000016600000	000000073000	000000000354
w			119	167	734000000000	003560000000	000016700000	000000073400	000000000356
x			120	170	740000000000	003600000000	000017000000	000000074000	000000000360
y			121	171	744000000000	003620000000	000017100000	000000074400	000000000362
z			122	172	750000000000	003640000000	000017200000	000000075000	000000000364
{			123	173	754000000000	003660000000	000017300000	000000075400	000000000366
|			124	174	760000000000	003700000000	000017400000	000000076000	000000000370
}			125	175	764000000000	003720000000	000017500000	000000076400	000000000372
~			126	176	770000000000	003740000000	000017600000	000000077000	000000000374
RUB			127	177	774000000000	003760000000	000017700000	000000077400	000000000376

** Codetabelle: ASCII und SIXBIT mit interner Darstelluns von ASCII-Symbolen ** Seite 5 **

****** Bedeutuns der Control-Characters ******

NUL		Nullzeichen	SOH	<ctrl-A>	Start Of Heading
STX	<ctrl-B>	Start of TEXT	ETX	<ctrl-C>	End of TeXt
EOT	<ctrl-D>	End Of Transmission	ENQ	<ctrl-E>	ENQuiry
ACK	<ctrl-F>	ACKnowledge	BEL	<ctrl-G>	Bell
BS	<ctrl-H>	BackSpace	TAB	<ctrl-I>	horizontal TABulator
LF	<ctrl-J>	Line Feed	VT	<ctrl-K>	Vertical Tabulator
FF	<ctrl-L>	Form Feed	CR	<ctrl-M>	Carriage Return
SO	<ctrl-N>	Shift Out	SI	<ctrl-O>	Shift In
DLE	<ctrl-P>	Data Link Escape	DC1	<ctrl-Q>	Device Control 1
DC2	<ctrl-R>	Device Control 2	DC3	<ctrl-S>	Device Control 3
DC4	<ctrl-T>	Device Control 4	NAK	<ctrl-U>	Negative AcKnowledge
SYN	<ctrl-V>	SYNchronous idle	ETB	<ctrl-W>	End of Transmission Block
CAN	<ctrl-X>	CANcel	EM	<ctrl-Y>	End of Medium
SUB	<ctrl-Z>	SUBstitute	ESC	<ctrl-[>	ESCape (Altmode)
FS	<ctrl-\>	File Seperator	GS	<ctrl-]>	Group Seperator
RS	<ctrl-^>	Record Seperator	US	<ctrl-_>	Unit Seperator

RUB Delete (RUBout)

Anhang B

Verzeichnis der Routinen

Die folgende Aufstellung enthaelt die in der BWLLIB vorhandenen Routinen in alphabetischer Reihenfolge. In Klammern ist jeweils die Quellensprache des Moduls angegeben.

ADDR	(MACRO)	Bestimmung der Adresse einer Variablen im Arbeitsspeicher
BIT	(MACRO)	Setzen und Loeschen einzelner Bits
C6T7	(MACRO)	Konvertierung von SIXBIT- Texten in ASCII-Texte
C7T6	(MACRO)	Konvertierung von ASCII-Texten in SIXBIT-Texte
CLEAN	(MACRO)	Loeschen des Bildschirm (alte Version)
CLEAR	(MACRO)	Loeschen des Bildschirms (neue Version)
CURPOS	(MACRO)	Cursor-Positionierung
DATE1	(MACRO)	Tagesdatum
DIVIDE	(MACRO)	Division mit Rest
DECFIL	(FORTRAN)	Dekodieren eine Filespezifikation
ECHO	(MACRO)	Einschalten des Ecohs
ENCFIL	(FORTRAN)	Kodieren einer Filespezifikation
EXIT1	(MACRO)	Programmunterbrechung
HLL	(MACRO)	Kopieren eines Halbwortes

HLLE	(MACRO)	Kopieren eines Halbwortes
HLLO	(MACRO)	Kopieren eines Halbwortes
HLLZ	(MACRO)	Kopieren eines Halbwortes
HLR	(MACRO)	Kopieren eines Halbwortes
HLRE	(MACRO)	Kopieren eines Halbwortes
HLRO	(MACRO)	Kopieren eines Halbwortes
HLRZ	(MACRO)	Kopieren eines Halbwortes
HRL	(MACRO)	Kopieren eines Halbwortes
HRLE	(MACRO)	Kopieren eines Halbwortes
HRLO	(MACRO)	Kopieren eines Halbwortes
HRLZ	(MACRO)	Kopieren eines Halbwortes
HRR	(MACRO)	Kopieren eines Halbwortes
HRRE	(MACRO)	Kopieren eines Halbwortes
HRRO	(MACRO)	Kopieren eines Halbwortes
HRRZ	(MACRO)	Kopieren eines Halbwortes
IBIT	(MACRO)	Pruefen von Bits
INSYM	(MACRO)	Lesen eines Zeichen vom TTY
JBIT	(MACRO)	Pruefen eines Bits
JOBNR	(MACRO)	Jobnummer
JOBPPN	(MACRO)	Benutzernummer
JOBTTY	(MACRO)	Terminalnummer
MCALL	(MACRO)	Betriebssystemaufruf
NOBITS	(MACRO)	Zaehlen von Bits
NOECHO	(MACRO)	Abschalten des Echos
OUTSYM	(MACRO)	Aussabe eines Zeichens
PACK	(MACRO)	Packen von Texten
RTIM	(MACRO)	CPU-Zeit
RTIME	(MACRO)	CPU- und Tageszeit
RUN	(FORTRAN)	Programmaufruf

RUNP1	(MACRO)	Programmaufruf
SLEEP	(MACRO)	Desaktivierung
TRMTYP	(MACRO)	Terminaltyp
TSORTO	(MACRO)	Sortieren von Texten
UNPACK	(MACRO)	Entpacken von Texten
ZUFALL	(FORTRAN)	Zufallszahlengenerator

Anhang C

Index