PDV-E 133
Oktober 1979

PDV-Entwicklungsnotizen

Spezifikation der Zwischensprache CIMIC/P

B. F. Eichenauer, R. Henn, K. Lucas, A. Zeh
GPP, Gesellschaft für Prozeßrechnerprogrammierung mbH,
München

Kernforschungszentrum Karlsruhe

PDV-Berichte

Die Kernforschungszentrum Karlsruhe GmbH koordiniert und betreut im Auftrag des Bundesministers für Forschung und Technologie das im Rahmen der Datenverarbeitungsprogramme der Bundesregierung geförderte Projekt Prozeßlenkung mit Datenverarbeitungsanlagen (PDV). Hierbei arbeitet sie eng mit Unternehmen der gewerblichen Wirtschaft und Einrichtungen der öffentlichen Hand zusammen. Als Projektträger gibt sie die Schriftenreihe PDV-Berichte heraus. Darin werden Entwicklungsunterlagen zur Verfügung gestellt, die einer raschen und breiteren Anwendung der Datenverarbeitung in der Prozeßlenkung dienen sollen.

Der vorliegende Bericht dokumentiert Kenntnisse und Ergebnisse, die im Projekt PDV gewonnen wurden.

Druck und Verbreitung:

Kernforschungszentrum Karlsruhe GmbH
Postfach 3640 7500 Karlsruhe 1

Bundesrepublik Deutschland

Zusammenfassung

Es wird die Zwischensprache CIMIC/P beschrieben, in die PEARL-Programme, die nach der Spezifikation des PEARL-BASIS-Subset vom 8.7.1976 [1] geschrieben werden, durch den portablen GPP-PEARL-Compiler uebersetzt werden.

Inhalt

======

1. Einfuehrung

2. Die abstrakte Maschine

2.1 Aufbau der abstrakten Maschine
2.2 Ausfuehrung von Rechenoperationen
2.3 Registerverwendung

3. Konstituenten von CIMIC/P-Anweisungen

3.1 Aufbau von CIMIC/P-Anweisungen
3.2 Operationscodes
3.3 Operandentypen

3.3.1 Elementare Datentypen
3.3.2 Typen zusammengesetzter Objekte

3.4 Operanden
3.4.1 Konstanten
3.4.2 Variablen

4. Pseudo-Anweisungen

4.1 Moduln

4.2 Vereinbarungen
4.2.1 Task-und Prozedurentries
4.2.2 Vereinbarungen globaler Groessen
4.2.3 Block-und Variablenvereinbarung
4.2.4 Taskvereinbarung
4.2.5 Prozedurvereinbarung
4.2.6 Markenvereinbarungen
4.2.7 Vereinbarung von Files
4.2.8 Vereinbarung von Devices
4.2.9 Vereinbarung von Interrupts
4.2.10 Vereinbarung von Signals
4.2.11 Vereinbarung von Semaphoren
4.2.12 Formatvereinbarungen

4.3 Spezifikation globaler Groessen

5. Transfer-Anweisungen

6. Algorithmische Anweisung

6.1 Monadische Operatoren
6.2 Dyadische Operatoren

7. Anweisung fuer die Ablaufsteuerung

7.1 Sprunganweisung
7.2 Fallauswahl
7.3 ON-Statement
7.4 Prozeduraufruf
7.5 Verlassen einer Prozedur

8. Anweisungen fuer die parallele Ablaufsteuerung

8.1 Anweisungen zur direkten Tasksteuerung
8.2 Anweisungen fuer die Taskzuteilung
8.3 Unterbrechungsoperationen
8.4 Synchronisationsoperationen

9. Anweisungen fuer die Ein/Ausgabe

9.1 Einrichten und Vernichten von Files
9.2 Oeffnen und Schliessen von Files
9.3 Unformatierte Ein/Ausgabe
9.4 Formatierte Ein/Ausgabe
9.5 Ein/Ausgabe-Listen
9.6 Prozess-Ein/Ausgabe

Fussnoten

Literaturangaben

CIMIC/P-Programmbeispiel

Anhang A: Kurze Beschreibung der Notation zur Darstellung
der Syntax von CIMIC/P

A.1 Endsymbole

A.2 Produktionsregeln

Anhang B: Vollstaendige Syntax von CIMTC/P

B.1 Lexikalsymbole

B.2 Syntaxregeln

B.3 Querbezugsliste der Regelnamen

1. Einfuehrung

Eine wesentliche Aufgabe, die bei der Erstellung von Basis-Software fuer Prozessrechner heute geloest werden muss, besteht in der Bereitstellung problemangemessener Programmier-Werkzeuge, mit denen sich einerseits der durch die Personalleistungen anfallende zunehmende Kostendruck wenigstens teilweise auffangen laesst, und die es andererseits ermoeglichen, ganz oder in Teilen wiederverwendbare Programmpakete fuer die Prozessautomatisierung zu erstellen.

Mit der Definition solcher Programmier-Werkzeuge, zu denen insbesondere die Prozess-Programmiersprache PEARL gehoert, ist die angegebene Aufgabe jedoch noch nicht geloest. Beruecksichtigt man die Vielzahl der heute auf dem Markt vorhandenen und erst recht der in naher Zukunft zu erwartenden billigen Rechensysteme, so ist die Spezifikation solcher Programmier-Werkzeuge nur sinnvoll, wenn diese sich schnell und mit angemessenen Kosten bereitstellen lassen, d.h. bei der Erstellung von Prozess-Automatisierungsprogrammen tatsaechlich einsetzbar sind.

Mit der Spezifikation der Zwischensprache CIMIC/P fuer den PEARL-BASIS-SUBSET [1] soll ein erster Schritt in diese Richtung getan werden. Durch den z.Z. bei der GPP mbH in Entwicklung befindlichen PEARL-Compiler werden PEARL-Programme in CIMIC/P-Programme umgesetzt, bevor sie dann mit relativ einfachen Codegeneratoren in den Assembler oder die Binder/Laderform eines Zielrechners ueberfuehrt werden.

Die vorliegende Spezifikation beschreibt die Zwischensprache CIMIC/P.Sie setzt die Kenntnis von CIMIC/C und die Vertrautheit mit Problemen der Codegenerierung voraus.

2. Die abstrakte Maschine

CIMIC/P kann als Assemblersprache einer abstrakten Maschine angesehen werden. Zur Vereinfachung der Darstellung von CIMIC/P wird nachfolgend die Struktur und die Arbeitsweise dieser Maschine insoweit beschrieben, als sie mittels CIMIC/P-Anweisungen angesprochen und beeinflusst werden kann.

Es sei vorab bemerkt, dass die Eigenschaften des hier umrissenen Automaten gewaehlt wurden, um die Algorithmen, die bei der Implementierung bzw. bei der Verwendung hoeherer Programmiersprachen eingesetzt werden, auf einem relativ niedrigen Abstraktionslevel [2] darstellen zu koennen. Es ist keinesfalls erforderlich, dass eine reale Rechenanlage, die ueber CIMIC/P erreicht werden soll, mit einem Assoziativspeicher oder mit Operandenstacks der hier verwendeten Art ausgeruestet sein muss. Die Adressvergabe und die Abbildung von Stacks kann bei der Codegenerierung erledigt werden und obliegt dem Implementator des Codegenerators.

2.1 Aufbau der abstrakten Maschine

Die zugrundegelegte abstrakte Maschine ergibt sich aus dem von W.M. Waite vorgeschlagenen und fuer CIMIC eingesezten JANUS-Rechner [3,4,5] durch Hinzunahme von abstrakten Geraeten.

Der CIMIC/P-Rechner besteht aus einem Prozessor, einem Speicher und einem Operanden-Stack (vergl. Abb. 1). Der Speicher und der Stack bestehen jeweils aus adressierbaren Einheiten, die u.a. zur Aufnahme von Ganzzahlen und Adressen geeignet sind.

Im Speicher der abstrakten Maschine wird Platz fuer Befehle und benannte Operanden vorgesehen. Sofern diese innerhalb von Rechenoperationen auftreten, werden sie zuvor in den Operanden-Stack gebracht, wo auch die unbenannten Operanden, z.B. die waehrend der Auswertung von arithmetischen Ausdruecken auftretenden Zwischenergebnisse, Platz finden. Der Operanden-Stack wird streng im LAST-IN/FIRST-OUT-Mode betrieben. Je nach der Art des Operators (monadisch oder dyadisch) wird nur auf die oberste Stackzelle bzw. auf die beiden obersten Stackzellen zugegriffen.

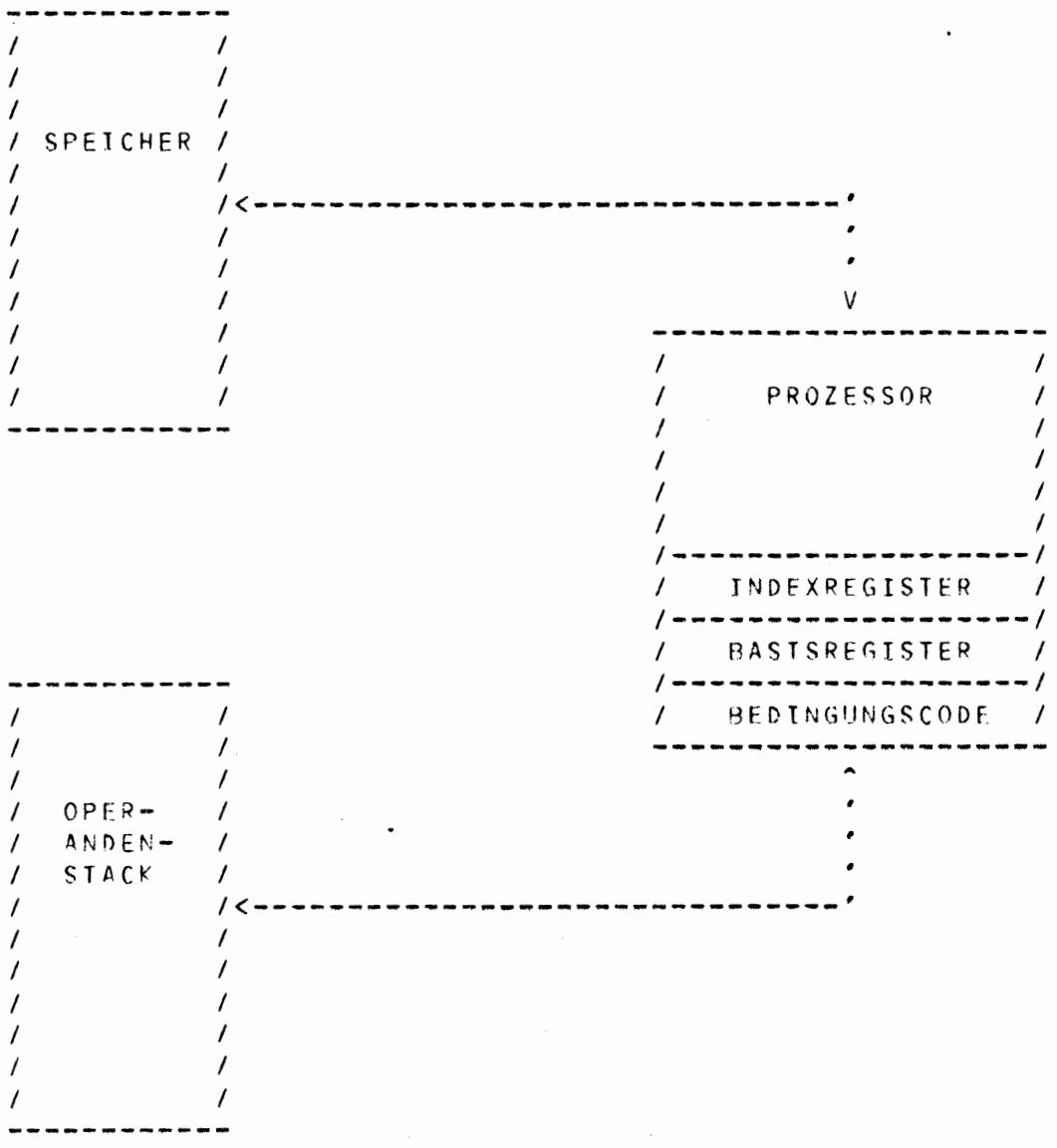

Abb.1: Struktur der CIMIC/P-Maschine

2.2 Ausfuehrung von Rechenoperationen

Eine Rechenoperation besteht aus einem Operator und, falls der rechte Operand noch nicht im Stack vorliegt, aus einem expliziten Operanden.

Sie wird in folgenden vier Schritten ausgefuehrt:

a) Falls ein expliziter Operand angegeben ist, wird er in den Stack geschafft. Andernfalls entfaellt Schritt a).

b) Eine vom Operator abhaengige Anzahl von Objekten wird aus dem Stack in den Prozessor ueberfuehrt.

   Bei monadischen Operatoren wird das oberste Stackelement in den Prozessor gebracht. Bei dyadischen Operatoren werden die beiden obersten Stackelemente ueberfuehrt. Dabei wird vorausgesetzt, dass sich der linke bzw. rechte Operand in der zweitobersten bzw. obersten besetzten Stackzelle befindet.

c) Der Prozessor fuehrt die angegebene Operation aus.

d) Das Ergebnis der Operation wird aus dem Prozessor in den Stack kopiert. Dieser Schritt kann durch Operatormodifikation (N-Flag) unterdrueckt werden. ((1))

2.3 Registerverwendung

Neben den Arbeitsregistern, auf die hier nicht eingegangen wird, da sie in CIMIC/P-Anweisungen nicht explizite ansprechbar sind, enthaelt der Prozessor ein Indexregister, das sog. BASIS Register und ein Register BEDINGUNGSCODE.Das Indexregister und das BASISREGISTER werden zur Adressierung des Speichers der abstrakten Maschine eingesetzt. Die Ausfuehrung einer TRANSFER-Operation vom Speicher in den Operanden-Stack oder umgekehrt erfolgt in zwei Schritten:

a) Falls ein expliziter Operand angegeben ist, wird seine Adresse in das BASIS-Register gebracht. Andernfalls enthaelt das BASIS-Register schon die Operandenadresse und Schritt a) entfaellt. ((2))

b) Die Transfer-Operation wird durchgefuehrt.

Das Indexregister der abstrakten Maschine wird fuer den indirekten Zugriff auf Feldelemente eingesetzt. Dazu wird zunaechst der "Index" in das Register gebracht. Eine darauffolgende Operation mit einer Laengenangabe in Indexposition wird wie folgt ausgewertet:

- Die Laengenangabe wird mit dem im Indexregister angegebenen Wert multipliziert.
- Falls ein fester Offset in der Operation angegeben wurde, wird dieser der Multiplikation zugeschlagen.
- Die Anfangsadresse des Feldes wird hinzuaddiert. Sie ist ueber den Dope-Vektor des Feldes zu erreichen und entspricht der Adresse eines (fiktiven) 0.Feldelements.

Bei der Durchfuehrung von Vergleichsanweisungen wird das Vergleichsergebnis im Register BEDINGUNGSCODE notiert. Abhaengig vom Inhalt des Registers kann dann der Programmablauf in nachfolgenden bedingten Sprunganweisungen verzweigt werden.

Folgende Vergleichsergebnisse sind in BEDINGUNGSCODE abspeicherbar:

LT Der Bedingungscode LT wird erzeugt, wenn der linke Operand kleiner als der rechte Operand ist.

EQ Der Bedingungscode EQ wird bei Gleichheit der Operanden erzeugt.

GT Der Bedingungscode GT wird erzeugt, wenn der linke Operand groesser als der rechte Operand ist.

3. Konstituenten von CIMIC/P-Anweisungen

3.1 Zum Aufbau von CIMIC/P-Anweisungen

Zur Darstellung der Operationen, die von der in Abschnitt 2 beschriebenen abstrakten Maschine ausgefuehrt werden, benoetigt man in der Regel drei Angaben:

a) den Operationscode, der bestimmt, welche Operation bzw. Pseudooperation ausgefuehrt werden soll,

b) den Operandentyp, der angibt, in welcher Weise der Operand bei der Ausfuehrung der Operation zu interpretieren ist und

c) die Operandenangabe, die das Objekt einfuehrt bzw. den Ort, an dem es zu finden ist, angibt.

Bei bestimmten Operationscodes fehlt die Angabe b) und/oder die Angabe c). Beispielsweise kann die Operandenangabe entfallen, wenn sich der Operand schon im Stack befindet (vergl. Abschnitt 3.4).

3.2 Operationscodes

Die CIMIC/P-Anweisungen koennen in Anlehnung an die uebliche Einteilung von Assembler-Instruktionen in Pseudo-Anweisungen und in eigentliche Anweisungen eingeteilt werden. Einige Operationscodes koennen durch Anhaengen von Flags modifiziert werden. Ein "KEIN-RESULTAT" - Flag:

(N)

bedeutet, dass Schritt d) der in Abschnitt 2.2 beschriebenen Auswertungsvorschrift entfaellt, d.h. es werden nach Durchfuehrung der Operation durch den Prozessor keine Werte in den CIMIC/P-Stack transferiert.

Beispiel: STORE INT BASED;

In diesem Fall enthaelt das Basis-Register die Adresse der Speicherzelle, in welche der in der obersten Stackzelle befindliche Wert ueberfuehrt werden soll. Der Wert wird in den Stack zurueckgespeichert.

STORE(N) INT BASED;

Wie im vorhergehenden Beispiel; der Wert wird nicht in den Stack zurueckgespeichert.

Durch Anhaengen des Vertauschflag

(R)

wird bestimmt, dass die im Abschnitt 2.2 b) angegebene Operandenablage umzukehren ist, d.h. bei der Ausfuehrung einer dyadischen Operation wird der linke Operand in der obersten und der rechte Operand in der zweitobersten Stackzelle erwartet. Das Vertauschungsflag wird bei nichtkommutativen Operationen verwendet.

Beispiel: SUB INT.5,INT.8 BASED;

Vor Ausfuehrung der Operation steht der linke Operand vom Typ INT.5 in der obersten besetzten Stackzelle. Die Adresse des rechten Operanden vom Typ INT.8 steht im Basisregister.

SUB(R) INT.8,INT.5 BASED;

Vor Ausfuehrung der Operation steht der rechte Operand vom Typ INT.8 in der obersten besetzten Stackzelle.Die Adresse des linken Operanden vom Typ INT.5 steht im Basisregister.

3.3 Operandentypen

Der Typ eines Operanden bestimmt die Interpretation des Operanden bei der Verarbeitung in der abstrakten Maschine. Bei der Uebersetzung von

CIMIC/P vermittelt der Operandentyp dem Codegenerator den Zugriff zu den Eigenschaften eines Operanden (z.B. Speicherbedarf zur Darstellung der Operanden). Die Eigenschaften werden bei der Erstellung des Codegenerators festgelegt.

3.3.1 Elementare Datentypen

In CIMIC/P sind folgende Operandentypen vorgesehen:

ADDR

Ein Operand vom Typ ADDR stellt eine Speicheradresse im Daten- oder Anweisungsbereich eines Programms dar.

Um bei der Adaptierung des PEARL-Compilers Adressarithmetik in einfacher Weise zu ermoeglichen, wird in der Untermenge CIMIC/C von CIMIC/P vorausgesetzt, dass Adressen unter Verwendung der Ganzzahl-Arithmetik berechnet werden koennen [9].

STR ['.' INTEGER]

Ein Operand vom Typ STR ist die interne Darstellung einer Zeichenkette. Durch die optional anzugebende Ganzzahl wird die Zahl der Zeichen festgelegt. Diese Angabe kann bei Zeichenkonstanten und bei Variablen, die nur ein Zeichen aufnehmen, entfallen.

INT ['.' INTEGER]

Ein Operand vom Typ INT ist die interne Darstellung einer positiven oder negativen ganzen Zahl.

INT ist gleichbedeutend mit INT.16.

REAL ['.' INTEGER]

Ein Objekt vom Typ REAL ist die interne Darstellung einer rationalen Zahl. REAL.32 ist gleichbedeutend mit REAL

DUR

Ein Objekt vom Typ DUR ist die interne Darstellung einer Dauer

CLOCK

Ein Objekt vom Typ CLOCK ist die interne Darstellung einer Uhrzeit.

BIT ['.' INTEGER]

Ein Objekt vom Typ BIT ist die interne Darstellung einer Bitkette. Hinsichtlich der Option gilt das bei STR Beschriebene entsprechend.

DVC

Ein Objekt vom Typ DVC ist die interne Darstellung einer Anschlussbeschreibung (Device-Kontrol-Block).

IPPT — Ein Objekt vom Typ IRPT ist die interne Beschreibung eines Interrupts.

SIGN — Ein Objekt vom Typ SIGN ist die interne Beschreibung eines Signals.

FILE — Ein Objekt vom Typ FILE ist die interne Beschreibung eines Files (File-Control-Block).

DOPE — Ein Objekt vom Typ DOPE ist die interne Darstellung einer Feldbeschreibung. Sie muss mindestens die reduzierte Feldanfangsadresse und die Feldgrenzen bzw. die Kantenlaengen enthalten.

3.3.2 Typen zusammengesetzter Objekte

Neben den in Abschnitt 3.3.1 beschriebenen elementaren Datentypen sind in CIMIC/P Typen zusammengesetzter Objekte (u.a. von sog. Strukturen) einfuehrbar. Der Typ eines zusammengesetzten Objekts wird durch

```
MODE-SYMBOL: $M DIGIT*.
```

angegeben.

3.4 Operanden

In CIMIC/P wird zwischen elementaren Operanden und zusammengesetzten Operanden unterschieden. Elementar heisst ein Operand, wenn ueber seine Darstellung in der Zwischensprache keine Aussage gemacht werden kann. Elementar sind solche Operanden, die normalerweise unmittelbar auf Operanden realer Rechenanlagen abbildbar sind.

Zusammengesetzte Operanden werden in CIMIC aus elementaren Operanden aufgebaut. In CIMIC/P sind Felder elementarer Operanden, Strukturen und Felder von Strukturen vorgesehen.

Eine Struktur ist ein Operand, dessen Elemente elementare Operanden sind, die nicht notwendig den gleichen Typ besitzen muessen.

Ein Operand besitzt die allgemeine Form:

OPERAND:

DATA-CONSTANT /
['I,'] VARIABLE.

3.4.1 Konstanten

Eine Konstante DATA-CONSTANT stellt einen Wert dar, der vor der Ausfuehrung eines Programmsystems bekannt ist und waehrend der Ausfuehrung eines Programmsystems nicht veraendert wird. Konstante koennen bei der Erstellung eines CIMIC/P- Programmsystems, bei der Erstellung des Codegenerators fuer eine Zielmaschine oder beim Binden/Laden eines Programmsystems festgelegt werden.

Um die Festlegung von Konstanten vor und waehrend der Phasen eines Uebersetzungsvorgangs zu ermoeglichen, sind in CIMIC/P mehrere Formen von Konstantenangaben vorgesehen:

DATA-CONSTANT:

DENOTATION /

MANIFEST .

Mittels DENOTATIONen werden Konstanten dargestellt, die bei der Erstellung eines CIMIC/P- Programmsystems festgelegt werden. Die unter Verwendung von DENOTATIONen angegebenen Konstanten stellen sich selbst dar. Keine weitere Information ausser der DENOTATION wird benoetigt, um den Operanden zu erkennen:

DENOTATION:

SIMPLE-DENOTATION /

BIT-STRING-DENOTATION /

CHARACTER-STRING-DENOTATION.

Denotationen fuer elementare Konstanten werden unter SIMPLE-DENOTATION zusammengefasst.

SIMPLE-DENOTATION:

```
        INTEGER-CONSTANT /

        REAL-CONSTANT /

        DURATION-CONSTANT /

        CLOCK-CONSTANT /

        LABEL-CONSTANT .
```

Folgende Denotationen sind in CIMIC/P vorgesehen:

```
    INTEGER-CONSTANT:

        'AINT ' [ '-' ] DIGIT*
```

stellt eine Konstante vom Typ INT dar.

```
  REAL-CONSTANT:      'AREAL ' [ '-' ] POSITIVE-REAL-CONSTANT.
  POSITIVE-REAL-CONSTANT:
                      INTEGER 'E' [ '-' ] INTEGER.
```

stellt eine rationale Zahl dar.

Zeitkonstanten koennen Zeitintervalle oder Uhrzeiten angeben :

```
  DURATION-CONSTANT:
                      'ADUR ' TIME-CONSTANT.
  TIME-CONSTANT:
                      INTEGER ':' INTEGER ':'
                      POSITIVE-REAL-CONSTANT.
  CLOCK-CONSTANT:
                      'ACLO ' TIME-CONSTANT.

  LABEL-CONSTANT:  'R ' LABEL.
  LABEL : SYMBOL .
```

stellt eine (abstrakte) Adresse im Befehlskoerper einer Task oder Prozedur dar.

```
CHARACTER-STRING-DENOTATION:
                    'S ' INTEGER-LIST CHARACTER-CONTINUATION$.
CHARACTER-CONTINUATION:
                    '/' EOL
                    'CONT STR S ' INTEGER-LIST.
```

stellt eine Zeichenketten-Konstante dar.

Die Zeichen werden als Ganzzahlen im ASCII-Code dargestellt.

Bitkettenkonstanten werden in alphanumerischer Form analog zu Zeichenkettenkonstanten angegeben:

```
BIT-STRING-DENOTATION:
                    'B ' INTEGER-LIST BIT-CONTINUATION$.
BIT-CONTINUATION:
                    '/' EOL
                    'CONT BIT B ' INTEGER-LIST.
```

Um ein in CIMIC/P vorliegendes Programmsystem auf eine Zielmaschine zu ueberfuehren, ist nicht nur die Abbildung der CIMIC-Anweisungen auf Anweisungen der Zielmaschine auszufuehren, sondern es sind auch bestimmte maschinenabhaengige Konstanten festzulegen. Maschinenabhaengige Konstanten werden im Programm ueber

```
MANIFEST:  'M ' IDENTIFIER.
```

angesprochen. Darin steht IDENTIFIER fuer den Bezeichner einer Konstanten, deren Wert bei der Codegenerierung festgelegt wird. In CIMIC/P werden folgende Manifestwerte verwendet:

```
M TRUE
```

M FALSE — Eine der Konstanten geht aus der anderen durch boolsche Negierung hervor.

M ADRSHIFT — Diese Manifest-Konstante tritt nur in der Untermenge CIMIC/C [9] von CIMIC/P auf.

Um den PEARL-Compiler auch einfach an Rechenanlagen mit Byte-Adressierung adaptieren zu koennen, wird in CIMIC/P angenommen, dass sich die Adressen zweier aufeinander folgender Speicherworte zur Aufnahme von Ganzzahlen oder Adressen um eine Potenz von 2 mit Exponent M ADRSHIFT voneinander unterscheiden ((5)). Unter dieser Voraussetzung kann die Adressierung beim Zugriff auf Feldelemente (Feldabbildung) schon in CIMIC/C, d. h. vor der Codegenerierung komplett durchgefuehrt werden.

3.4.2 Variablen

Speicherplaetze fuer Operanden, die waehrend des Ablaufs eines Programmsystems unterschiedliche Werte annehmen koennen (VARIABLE), muessen mittels CIMIC/P-Anweisungen explizite angefordert werden (vergl. Abschnitt 4.2). Fuer Variablen wird Speicherplatz im Speicher der abstrakten Maschine vor gesehen. Um den verschiedenartigen Speicher-Vergabe-Strategien und Speicher-Zugriffs-Mechanismen Rechnung zu tragen, sind mehrere Speicher-Kategorien vorgesehen:

```
VARIABLE:
                ( 'GLOBAL' / 'STATIC' / 'DISPO' )
                    ' ' REFERENCE /
                'BASED' [ MODIFICATION ].
```

Eine Variable besteht danach aus einer Kategorie und gegebenenfalls aus einer expliziten REFERENCE auf einen Speicherplatz. Waehrend REFERENCE den in der Speicherzelle gespeicherten Wert anspricht, wird durch die Kategorie der Zugriff zu dieser Speicherzelle festgelegt.

Die angegebenen Speicherkategorien bedeuten im einzelnen:

GLOBAL — Speicherplatz fuer globale Variablen bleibt waehrend des gesamten Ablaufs eines Programmsystems erhalten und ist in allen Moduln, aus denen ein

Programmsystem erzeugt wird, ansprechbar.

STATIC — Speicherplatz fuer statische Variable ,d.h. fuer Variable , die auf Modulebene vereinbart wurden.

DISPO — SYMBOL wird als Speicherplatz einer lokalen Variablen im Aktivierungsbereich einer Task interpretiert. Der dynamisch anzulegende Bereich fuer lokale Prozedurvariablen gehoert zum Aktivierungsbereich der Task, unter deren Regie die Prozedur ablaeuft. Innerhalb einer Prozedur sind in CIMIC/P nur globale Variable, statische Variable, lokale Prozedurvariable und Parameter ansprechbar, d.h. der dynamische Aufbau eines Displayvektors eruebrigt sich ((3)).

BASED — Das BASIS-Register enthaelt die Adresse einer Variablen.

CODE — Diese Kategorie gibt an, dass sich das Objekt im Koerper einer Prozedur oder Task befindet(vgl. 9.5).

Durch Voranstellen des I-Flags, kann statt des Wertes einer Variablen deren Adresse angesprochen werden.

Beispiel: LOAD INT I,DISPO $L127;

Die Adresse der lokalen Variablen $L127 wird in den Operanden-Stack geladen.

Eine Reference besteht aus einer Basis (abstrakte Adresse). Handelt es sich um die Anfangsadresse eines zusammengesetzten Objekts, so wird die Fortschaltung innerhalb des Objekts durch eine Offset-Angabe bestimmt.

REFERENCE:

(XSYMB / SYMBOL) [MODIFICATION].

```
MODIFICATION:       OFFSET + BIT-SELECTOR .
OFFSET:             '(' [FIX-OFFSET] ')'
                    [ SINGLE-SIMPLE-MODE /
                      STRUCTURE-MODE-IDENTIFICATION ].
BIT-SELECTOR:       ' BIT ' INTEGER.
FIX-OFFSET:         TERM // ('+'/'-').
TERM:               [ INTEGER '*' ] SINGLE-SIMPLE-MODE .
```

Die Offsetangabe ist wie folgt zu interpretieren:

Durch '('FIX-OFFSET')' wird eine feste Adressfortschaltung bewirkt.

Beispiel:

```
DCL A STRUCT ( S1 FIXED ,
               S2 FLOAT ,
               S3 FLOAT);

A.S3 := ...
```

Die Speicheroperation besitzt die Form:

```
STORE(N) INT DISP0 $L108(INT+REAL);
```

Durch Angabe eines Modes nach der FIX-OFFSET-Angabe, kann darueber hinaus ein veraenderlicher Offset beruecksichtigt werden. Die der Modeangabe zugeordnete Laengenangabe wird mit dem Inhalt des Indexregisters multipliziert und zu der ueber den Dope-Vektor zugaenglichen Feldanfangsadresse (vgl. 2.3 b) bzw. zu der um den Fix-Offset erweiterten Feldanfangsadresse addiert. Zur Berechnung des Index,sowie zur Ueberpruefung der Feldgrenzen dient die externe Funktionsprozedur GINDEX. Es bleibt der Implementierung ueberlassen, ob die Einhaltung aller Feldgrenzen oder nur die der Gesamtfeldgrenze oder ueberhaupt nichts zur Laufzeit geprueft wird.

Beispiel: a)

```
DCL A(20,30) FIXED(18);

A(I,J) := ...
```

Fuer das Abspeichern wird folgende Befehlsfolge abgesetzt:

```
NCALL INT.15 X GINDEX;
NARGS DOPE DISP0 $L264(*,*) INT.18;
```

```
NARGIS INT.15 DISPO $L300;

NARGIS INT.15 DISPO $L303;

NCEND INT.15 X GINDEX;

INDEX INT.15 ;

STORE(N) INT.18 DISPO $L264()INT.18;
```

b)
```
DCL A(10) STRUCT ( S1 FIXED,
                   S2 FLOAT,
                   S3 FLOAT );

A(4).S3 := ...
```

Fuer diesen Zugriff wird abgesetzt:

```
INDEX INT AINT 4;

STORE(N) INT DISPO $L50(INT+REAL)$M27;
```

Darin ist $M27 der Mode einer Struktur. Bei der Verarbeitung der Strukturdeklarationen bzw. Strukturspezifikationen durch den Codegenerator wird die zielmaschinenabhaengige Laenge der Struktur berechnet und unter $M27 abgespeichert.

Zur Selektion eines Bits aus einer Bitkette kann der Operand mit dem Bitselektor erweitert werden:

Beispiel:
```
DCL A(10) BIT(16);

A(10).BIT(14) := ....
```

Die Zuweisung resultiert in der folgenden Befehlsfolge:

```
INDEX INT AINT 10;

STORE(N) INT DISPO $L60()BIT.16 BIT 14;
```

4. Pseudo-Anweisungen

4.1 Moduln

Entsprechend der fuer PEARL vorgesehenen Modultechnik [6,7,8] kann ein CIMIC/P-Programmsystem aus einer Anzahl unabhaengig voneinander erstellter Moduln aufgebaut werden. Jeder Modul beginnt mit der CIMIC/P- Pseudoanweisung

'MODULE ' [IDENTIFIER] EOL

wobei mit IDENTIFIER der eventuelle Modul-name festgehalten wird

und endet mit der Pseudoanweisung:

'MODEND' EOL

Jeder Modul besteht aus einer Folge von Vereinbarungen und/oder Spezifikationen:

MODULE:

'MODULE ' EOL

([MODULE-ELEMENT] EOL)$

'MODEND' EOL.

MODULE-ELEMENT:

GLOBAL-DECLARATION-OR-SPECIFICATION /

STATIC-DECLARATION /

PROCEDURE-DECLARATION /

TASK-DECLARATION /

FORMAT-LIST .

4.2 Vereinbarungen

Die Deklaration bzw. Spezifikation globaler Groessen erfolgt in CIMIC nach der Syntax:

```
GLOBAL-DECLARATION-OR-SPECIFICATION:
    'GLOBAL ' ( GLOBAL-ENTRY /
                ' ' ( GLOBAL-DECLARATION /
                      GLOBAL-SPECIFICATION ) ) .
```

4.2.1 Task-und Prozedurentries

Fuer globale Prozeduren und Tasks werden in dem Modul, in dem sie vereinbart sind, modul-lokale abstrakte Adressen angegeben:

```
GLOBAL-ENTRY
    XSYMB ' ' LOCAL-ADDR .

LOCAL-ADDR:
    ('PRENTR ' / 'TAENTR ') LOCAL-NAME.
```

Durch XSYMB wird der globale Bezeichner einer Prozedur oder Task definiert, ueber den sie von anderen Moduln angesprochen werden kann. LOCAL-NAME bezeichnet den lokalen Namen (abstrakte Adresse) der Prozedur oder Task in dem Modul, in dem sie vereinbart ist.

Beispiel: Die globale Task PEARL besitzt die lokale abstrakte Adresse $L1:

```
MODULE ;
    .
    .
    .
GLOBAL PEARL TAENTR $L1;
```

```
      .
      .
      .
TASK  DISPO $L1;
      .
      .
      .
TAEND;
      .
      .
      .
MODEND;
```

4.2.2 Vereinbarungen globaler Groessen

Die Deklaration von globalen Programmgroessen erfolgt gemaess:

```
GLOBAL-DECLARATION:
                    'DSECT ' EOL
                      ( ( 'SPACE '
                              SIMPLE-DECLARATION /
                              SIMPLE-ARRAY-DECLARATION /
                              STRUCTURE-DECLARATION /
                              STRUCTURE-ARRAY-DECLARATION /
                              SEMA-DECLARATION /
                              FILE-DECLARATION /
                              DEVICE-DECLARATION /
                              IRPT-DECLARATION /
                              SIGNAL-DECLARATION )
                      )*
                    'ENDBLK  DSECT '.
```

Zwischen der einleitenden GLOBAL-Anweisung und der den Datenblock abschliessenden ENDBLK-Anweisung werden die Elemente des Datenblocks (d.h. die globalen Programmgroessen) vereinbart.

In

```
SIMPLE-DECLARATION:
                    SINGLE-SIMPLE-MODE ' '
                    ( GLOBAL-CATEGORY / S-D-CATEGORY )
                    ' ' ( XSYMB / SYMBOL )
                    [ ' ' DATA-CONSANT ].
```

werden einfache Programmgroessen eingefuehrt, welche u.U. mit einem vorgegeben Wert initialisiert werden. Diese einfachen Programmgroessen koennen vom Mode

```
  SINGLE-SIMPLE MODE:
                    'INT' [ '.' INTEGER ] /
                    'REAL' [ '.' INTEGER ] /
                    'DUR' /
                    'CLOCK' /
                    'BIT' [ '.' INTEGER ] /
                    'STR' [ '.' INTEGER ].
```

sein.

Bei der Vereinbarung von globalen Groessen wird die

```
  GLOBAL-CATEGORY:
                    [ 'INV,' ] [ 'I,' ]
                    ( 'DESECT' / 'RSECT' ).
```

angegeben. Die GLOBAL-CATEGORY legt fest, ob die Programmgroesse

- VARIANT oder INVARIANT ist und
- ob sie im verdraengbaren Speicherbereich (DSECT) oder im residenten Bereich des Arbeitsspeichers (RSECT) liegen soll.

Durch XSYMB wird ein Bezeichner fuer die globale Groesse eingefuehrt.

Einfache Programmgroessen koennen mit einer Konstanten initialisiert

werden. (vgl. 3.4.1)

Beispiel:

```
DCL A FIXED(7) GLOBAL INIT 3;
```

Wird uebersetzt in:

```
SPACE INT.7 DSECT A AINT 3;
```

In

```
SIMPLE-ARRAY-DECLARATION:
                    DOPE-MODE ' '
                    ( GLOBAL-CATEGORY / S-D-CATEGORY )
                    ' ' ( XSYMB / SYMBOL ) BOUNDS ' '
                    'R ' SYMBOL EOL
                    'SPACE ' SINGLE-SIMPLE-MODE ' '
                    ( GLOBAL-CATEGORY / S-D-CATEGORY )
                    ' ' SYMBOL BOUNDS
                    [ '+' EOL ARRAY-VALUE ].
```

werden Felder von einfachen Programmgroessen gleichen Modes eingefuehrt. Zu jedem Feld wird ein Dope-Vektor angelegt, in dem Eigenschaften des Feldes, wie Anfangsadresse, Feldgrenzen usw. beschrieben werden. Die 1. CIMIC-Zeile veranlasst die Speicherbeschaffung fuer den Dope-Vektor. BOUNDS gibt die Feldgrenzen an, XSYMB bezeichnet den Dope-Vektor und der Label "R SYMBOL" verweist auf das Feld. Die 2. CIMIC-Zeile dient der Speicherbeschaffung fuer das Feld selbst.

Ein Feld von einfachen Programmgroessen kann initialisiert werden ueber

```
ARRAY-VALUE:
          ( 'ELMV ' ARRAY-ELEMENT-INITIALIZATION EOL )$
          'ARVND ' ARRAY-ELEMENT-INITIALIZATION.
```

Jede ELMV-Anweisung definiert den Anfangswert fuer jeweils ein Feldelement. Die Feldvereinbarung wird durch eine ARVND-Anweisung abgeschlossen, in der dem Feldelement mit dem groessten Index ein Anfangswert zugewiesen wird.

Beispiel:

```
DCL A(2,2) FIXED GLOBAL RESIDENT
           INIT (5);
```

wird uebersetzt in:

```
SPACE DOPE RSECT A(2,2) R $L20;
SPACE INT RSECT $L20(2,2)+;
ARVND INT RSECT (4) AINT 5 ;
```

In

```
STRUCTURE-DECLARATION:
        STRUCTURE-MODE-IDENTIFICATION ' '
        ( GLOBAL-CATEGORY / S-D-CATEGORY )
        ' ' ( XSYMB / SYMBOL )
        ( '+' EOL STRUCTURE-VALUE /
          EOL STRUCTURE-BODY-DECLARATION ).
```

wird eine Struktur als Zusammenfassung einfacher Programmgroessen vereinbart. Der Mode der Struktur wird in der

```
STRUCTURE-MODE-IDENTIFICATION:  '$M' DIGIT*.
```

angegeben.

Wird eine Struktur nicht initialisiert, so steht die

```
STRUCTURE-BODY-DECLARATION:
        ( 'STCELM ' SINGLE-SIMPLE-MODE ' '
                ( GLOBAL-CATEGORY / S-D-CATEGORY )
                ' ' ( XSYMB/ SYMBOL ) EOL )$
        'STCEED ' SINGLE-SIMPLE-MODE ' '
                ( GLOBAL-CATEGORY / S-D-CATEGORY )
                ' ' ( XSYMB/ SYMBOL ).
```

in der die einzelnen Selektoren der Struktur beschrieben werden. Fuer jeden Selektor steht eine STCELM-Anweisung, fuer den letzten Selektor eine STCEED-Anweisung.

Wird eine Struktur initialisiert, so steht der

```
STRUCTURE-VALUE:
        ( 'FLDV ' STRUCTURE-ELEMENT-INITIALIZATION EOL )$
        'LFVDV ' STRUCTURE-ELEMENT-INITIALIZATION.
STRUCTURE-ELEMENT-INITIALIZATION:
        SINGLE-SIMPLE-MODE ' '
        ( GLOBAL-CATEGORY / S-D-CATEGORY ) ' '
        SYMBOL ' ' DATA-CONSTANT.
```

Hinter der Beschreibung der Selektoren steht jeweils eine Datenkonstante, mit welcher der Selektor initialisiert wird.

```
Beispiel: DCL A STRUCT ( S1 FIXED ,
                         S2 FLOAT ) GLOBAL
                  INIT ( 5, 3E6 );
```

wird uebersetzt in:

```
SPACE $M27 DSECT A+;

FLDV INT DSECT S1 AINT 5;

LFVDV REAL DSECT S2 AREAL 3E7;
```

In der

```
STRUCTURE-ARRAY-DECLARATION:
                  DOPE-MODE ' '
                  ( GLOBAL-CATEGORY / S-D-CATEGORY )
                  ' ' ( XSYMR / SYMBOL ) BOUNDS ' '
                  'R ' SYMBOL EOL
                  'SPACE ' STRUCTURE-MODE-IDENTIFICATION ' '
                  ( GLOBAL-CATEGORY / S-D-CATEGORY )
                  ' ' SYMBOL BOUNDS
                  ( '+' EOL STRUCTURE-ARRAY-VALUE /
                    EOL STRUCTURE-BODY-DECLARATION ).
```

werden Felder von Strukturen vereinbart. In Analogie zur SIMPLE-ARRAY-DECLARATION wird auch hier ein Dope-Vektor aufgebaut. Die 1. Zeile bezeichnet und beschreibt den Dope-Vektor mit dem Bezug auf das eigentliche Feld; die 2. Zeile beschreibt die Struktur.

Wird eine Struktur nicht initialisiert, so werden die Selektoren durch STRUCTURE-BODY-DECLARATION beschrieben, im Falle der Initialisierung steht

```
STRUCTURE-ARRAY-VALUE:
        ( 'ELMV ' STRUCTURE-ARRAY-ELEMENT-INITIALIZATION
                  STRUCTURE-VALUE EOL )$
        'ARVND '  STRUCTURE-ARRAY-ELEMENT-INITIALIZATION
                  STRUCTURE-VALUE.
```

Jede ELMV-Anweisung definiert in Analogie zur SIMPLE-ARRAY-DECLARATION ein Feldelement. Die Feldvereinbarung wird durch eine ARVND-Anweisung abgeschlossen, in der dem Feldelement mit dem groessten Index ein Anfangswert zugewiesen wird.

Die Selektoren eines Feldelementes werden durch die

```
STRUCTURE-ARRAY-ELEMENT-INITIALIZATION:
        STRUCTURE-MODE-IDENTIFICATION ' '
        ( GLOBAL-CATEGORY / S-D-CATEGORY ) ' '
        '(' INTEGER ')' '+' EOL.
```

beschrieben und mit dem STRUCTURE-VALUE initialisiert.

Beispiel:
```
DCL A(2) INV STRUCT ( S1 FIXED ,
                      S2 FLOAT ) GLOBAL

         INIT (0,0.,0,0. )
```

ergibt folgende CIMIC/P-Zeilen:

```
SPACE DOPE INV,DSECT A(2).R $L28;

SPACE $M30 INV,DSECT $L28(2)+;

ELMV $M30 INV,DSECT (1)+;
```

```
FLDV INT INV,DSECT S1 AINT 0;
LFVDV INT INV,DSECT S2 AREAL 0E0;
ARVND $M30 INV,DSECT (1)+;
FLDV INT INV,DSECT S1 AINT 0;
LFVDV INT INV,DSECT S2 AREAL 0E0;
```

Fuer die folgenden Alternativen der GLOBAL*dECLARATION wird auf folgende Abschnitte verwiesen.

SEMA-DECLARATION (Abschnitt 4.2.11)

FILE-DECLARATION (Abschnitt 4.2.7)

DEVICE-DECLARATION (Abschnitt 4.2.8)

IRPT-DECLARATION (Abschnitt 4.2.9)

SIGNAL-DECLARATION (Abschnitt 4.2.10)

4.2.3 Block und Variablenvereinbarung

Alle CIMIC/P-Variablen sind zu vereinbaren. Die Vereinbarung fuehrt den Bezeichner SYMBOL fuer die Variable ein und spezifiziert alle relevanten Eigenschaften der Variablen.

Statische Variablen werden gemaess:

```
STATIC-DECLARATION:
                    'SPACE ' (
                      SIMPLE-DECLARATION              /
                      SIMPLE-ARRAY-DECLARATION        /
                      STRUCTURE-DECLARATION           /
                      STRUCTURE-ARRAY-DECLARATION     /
                      SEMA-DECLARATION                /
                      FILE-DECLARATION ).
```

vereinbart.

Die STATIC-DECLARATIONen sind analog den GLOBAL-DECLARATIONEN aufgebaut. Die GLOBAL-CATEGORY ist durch die

S-CATEGORY: 'STATIC'.

zu ersetzen.

Beispiel: Vereinbarung eines statischen Feldes von Ganzzahlen mit 3 Elementen. Die Anfangswerte fuer die Feldelemente lauten der Reihe nach: 3,4,5.

```
SPACE DOPE STATIC $L27(3) R $L29;

SPACE INT STATIC $L29(3)+;

ELMV INT STATIC (1) AINT 3;

ELMV INT STATIC (1) AINT 4;

ARVND INT STATIC (1) AINT 5;
```

Die Vereinbarung lokaler Prozedur- und Taskvariablen erfolgt nach der Produktionsregel:

```
LOCAL-DECLARATION:
                    ( SIMPLE-DECLARATION /
                      SIMPLE-ARRAY-DECLARATION /
                      STRUCTURE-DECLARATION /
                      STRUCTURE-ARRAY-DECLARATION /
                      LOCAL-BLOCK-DECLARATION )  EOL  .
```

Durch die Gesamtheit der lokalen Variablendeklarationen innerhalb einer Prozedur wird der Speicherbedarf fuer die lokalen Prozedurdaten festgelegt. Die Vergabe und Freigabe dieses Speicherplatzes erfolgt innerhalb des Aktivierungsbereiches der Task, unter deren Regie die Prozedur ausgefuehrt wird, dynamisch zur Laufzeit bei Eintreten bzw. Verlassen einer Prozedur.

Die ersten 4 Alternativen entsprechen den GLOBAL-DECLARATIONen. Die GLOBAL-CATEGORY ist durch die

```
D-CATEGORY:  'DISPO'.
```

zu ersetzen. Die fuenfte Alternative:

```
LOCAL-BLOCK-DECLARATION:
        'BLOCK '  INTEGER  ' DISPO ' EOL
              (CODE-ELEMENT EOL)*
        'BLEND '  INTEGER  ' DISPO '.
```

ermoeglicht u.a. eine oekonomische Verwendung des Speicherplatzes fuer lokale Prozedurdaten (Blockstruktur).

Durch die Block-Pseudoanweisung wird eine Anfangsadresse innerhalb des Aktivierungsbereichs definiert, relativ zu dem die innerhalb des Blocks vereinbarten lokalen Variablen angesprochen werden. Die BLEND-Anweisung gibt den fuer die lokalen Variablen zugeordneten Speicherplatz wieder frei. Um die Schachtelung lokaler Bloecke zu ermoeglichen, wird durch die in der BLOCK- bzw. BLEND-Anweisung angegebene Ganzzahl-Konstante INTEGER die Blocktiefe bzgl. der Prozedur oder Task festgelegt, die den Block enthaelt. Der aeusserste lokale Block besitzt die Blocktiefe 1. Prozeduren und Tasks liegen auf Blockniveau 0 ((4)).

Beispiel: Die folgende Prozedur $L27 benoetigt 2 lokale Speicherplaetze zur Aufnahme von Ganzzahlen.

```
.
.
.
BEGIN  DISPO $L27;
BLOCK 1 DISPO ;
SPACE INT DISPO $L50;
.
```

```
                .
                .
BLOCK 2 DISPO ;
SPACE INT DISPO $L57;
                .
                .
                .
BLEND 2 DISPO ;
BLOCK 2 DISPO ;
SPACE INT DISPO $L80;
                .
                .
                .
BLEND 2 DISPO ;
                .
                .
                .
BLEND 1 DISPO ;
END  PROC $L27;
                .
                .
                .
```

4.2.4 Taskvereinbarung

Ein CIMIC/P-Modul besteht (wie ein PEARL-Modul) aus einer Anzahl von Spezifikationen und Vereinbarungen. Die in Abschnitt 5 bis 9 beschriebenen ausfuehrbaren Anweisungen sind Bestandteile der Task- und Prozedurvereinbarungen.

Eine Taskvereinbarung besitzt die Form:

```
TASK-DECLARATION:
                    TASK-HEAD  TASK-BODY.
TASK-HEAD:
                    'TASK'  TASK-HEADING [ ' ' PRIORITY ] EOL
TASK-HEADING:
                     ' ' [ 'R' ] 'DISPO ' ( XSYMB / SYMBOL ).
PRIORITY:
                    'PRIO ' INTEGER .
TASK-BODY:
                    ( CODE-ELEMENT EOL )*
                    'TAEND' [ ' ' TASK-HEADING ].
CODE-ELEMENT:
                    LOCAL-DECLARATION /
                    LABEL-DECLARATION /
                    STATEMENT .
```

In der ersten Zeile der Taskvereinbarung wird der lokale Name der Task angegeben. Darauf folgt das Segment der Task. Eine Taskdeklaration wird mit der TAEND-Anweisung abgeschlossen.

4.2.5 Prozedurvereinbarung

Prozedurvereinbarungen bestehen aus einem Prozedurkopf und einem Prozedurkoerper:

```
PROCEDURE-DECLARATION:
     PROCEDURE-HEAD. PROCEDURE-BODY.
```

Der Prozedurkopf enthaelt alle Informationen, die fuer das Anbinden der Prozedur an eine Aufrufumgebung benoetigt werden:

```
PROCEDURE-HEAD:
                    [ 'N' / 'Q' ] 'BEGIN '
                                  PROCEDURE-HEADING EOL
                    ( [ 'N' / 'Q' ] 'PARAM '
                      ( SIMPLE-P-SPECIFICATION              /
                        SIMPLE-ARRAY-P-SPECIFICATION        /
```

```
                    STRUCTURE-P-SPECIFICATION          /
                    STRUCTURE-ARRAY-P-SPECIFICATION    /
                    FILE-P-SPECIFICATION               /
                    DEVICE-P-SPECIFICATION             )
                                                       )$
                ( 'N' / 'Q' / 'R' ) 'PAREND '
                                PROCEDURE-HEADING EOL .
PROCEDURE-HEADING:
                [ RESULT-MODE ] ' '
                  [ 'R' ] D-CATEGORY
                  ' ' ( XSYMB / SYMBOL ).
SIMPLE-P-SPECIFICATION:
                S-MODE ' '
                P-CATEGORY ' '
                [ SYMBOL ].
SIMPLE-ARRAY-P-SPECIFICATION:
                DOPE-MODE ' '
                P-CATEGORY ' '
                [ SYMBOL ]
                P-ARRAY-DIMENSION
                ' ' SINGLE-SIMPLE-MODE.
P-ARRAY-DIMENSION:
                '(' [ [ '*' ',' ] '*' ',' ] '*' ')'.
STRUCTURE-P-SPECIFICATION:
                STRUCTURE-MODE-IDENTIFICATION ' '
                P-CATEGORY ' '
                [ SYMBOL ] EOL
                STRUCTURE-MODE-DISPLAY.
STRUCTURE-ARRAY-P-SPECIFICATION:
                DOPE-MODE ' '
                P-CATEGORY ' '
                [ SYMBOL ]
                P-ARRAY-DIMENSION
                ' ' STRUCTURE-MODE-IDENTIFICATION EOL
                STRUCTURE-MODE-DISPLAY.
FILE-P-SPECIFICATION:
                FILE-MODE ' '
                P-CATEGORY ' '
                [ FILE-IDENTIFICATION ] EOL
                FILE-MODE-DISPLAY.
DEVICE-P-SPECIFICATION:
              ( ( DOPE-MODE ' '
                  P-CATEGORY ' '
                  [ SYMBOL ]
                  '(*)'
                  ' ' DEVICE-MODE ) /
                ( DEVICE-MODE ' '
                  P-CATEGORY ' '
                  [ SYMBOL ] ) ) EOL
                DEVICE-MODE-DISPLAY.
```

Aus der ersten Zeile des Prozedurkopfs geht hervor,

- ob die Prozedur einen Wert an die aufrufende Prozedur oder Task zurueckliefert (Funktionsprozedur) und
- unter welchem Namen die Prozedur innerhalb des Moduls angesprochen wird.

Bei Funktionsprozeduren wird der Funktionswert in der obersten besetzten Zelle des Operandenstack uebergeben.

Falls an eine Prozedur beim Aufruf Parameter uebergeben werden, folgen auf die BEGIN- Anweisung die Spezifikationen der formalen Parameter. Sie definieren jeweils den Parametertyp und die lokalen Namen, unter denen die Parameter in der Prozedur angesprochen werden.

Der Prozedurkopf wird mit der PAREND-Anweisung abgeschlossen, mit der die Prozedur in die aktuelle Programmumgebung eingebunden wird.

Der Prozedurkoerper besteht aus Deklarationen und ausfuehrbaren Anweisungen:

```
PROCEDURE-BODY:
                    ( CODE-ELEMENT EOL )*
                    [ 'N' / 'Q' ] 'END '
                            [ PROCEDURE-HEADING ].
```

Der Speicherbedarf fuer lokale Variable ist in CIMIC/P zur Uebersetzungszeit bekannt. Speicherplatz fuer lokale Variable wird zusammen mit dem Speicherplatz fuer aktuelle Parameter im Aktivierungsbereich der Task vorgesehen, unter der die Prozedur ausgefuehrt wird.

Beispiele fuer Prozedurvereinbarungen:
---

a) Uebertragung einer einfachen Variablen per Wert:

```
P1: PROC (A FIXED(7) );
              .
              .
              .
```

wird in die folgende CIMIC/P-Folge uebersetzt:

```
     NBEGIN  DISP0 $L87;

     NPARAM INT,7 DISP0 $L90;

     NPAREND  DISP0 $L87;
```

b) Die Uebertragung eines Feldes per Wert ist im PEARL-BASIS-SUBSET nicht vorgesehen, da die Implementierung grossen Aufwand erfordert (Dynamische Speicherverwaltung).

c) Uebertragung einer einfachen Variablen per Reference:

```
P3: PROC( A FIXED(7) IDENT);
              .
              .
              .
```

ergibt die CIMIC/P-Folge

```
     NBEGIN .....
     NPARAM INT,7 I,DISP0 $L20;
     NPAREND ....
```

d) Uebertragung eines Feldes per Reference:

ist gleichbedeutend mit der Uebertragung seines Dope per Wert:

```
P4: PROC( A(,) FIXED(7) IDENT);
              .
              .
              .
```

fuehrt zu der CIMIC/P-Folge:

```
     NBEGIN ....
     NPARAM DOPE DISP0 $L20(*,*) INT,7;
     NPAREND ....
```

4.2.6 Markenvereinbarungen

Eine Marke wird im CIMIC-Programmtext mit der Anweisung:

LABEL-DECLARATION:

'LOC ' SYMBOL.

eingefuehrt. Darin stellt SYMBOL die abstrakte Adresse der naechsten CIMIC-Anweisung dar. Bei der Ausfuehrung einer LOC-Anweisung ist der Operanden-Stack leer.

.2.7 Vereinbarung von Files

ine File-Deklaration besitzt in CIMIC/P die Form:

```
FILE-DECLARATION:
                    FILE-HEADING
                    FILE-MODE-DISPLAY.
FILE-HEADING:
                    FILE-MODE ' '
                    ( GLOBAL-CATEGORY / G-S-D-CATEGORY )
                    ' ' FILE-IDENTIFICATION EOL.
FILE-MODE-DISPLAY:
                    'FTYPE ' FILE-TYPE ' '
                             ( GLOBAL-CATEGORY / S-D-CATEGORY )
                             ' ' EOL
                    'FCONT ' FILE-DIMENSION ' '
                             ( FILE-CONTENTS-MODE ' '
                               ( GLOBAL-CATEGORY / S-D-CATEGORY )
                               ' ' ) /
                             ( DOPE-MODE ' '
                               ( GLOBAL-CATEGORY / S-D-CATEGORY )
                               ' ' CONTENTS-DIMENSION ' '
                               FILE-CONTENTS-MODE ) EOL
                    'FILEND ' .
```

Ein File kann im PEARL-BASIS-SUBSET auf Modulebene mit und ohne GLOBAL-Attribut vereinbart werden. Daneben kann es auch als formaler Parameter auftreten.

Durch die FILE-TYPE-Zeile wird der Typ des Files beschrieben. Dieser ergibt sich unmittelbar aus den entsprechenden Angaben im PEARL-Quellprogramm:

```
FILE-TYPE:
                    ( 'INL' / 'EXT' ) ','
                    USAGE ','
                    ( 'DIR' / 'SEQ' ).
USAGE:              'INP' / 'OUP' / 'IOP'.
```

Die FCONT-Anweisung beschreibt neben

a) der Dimension des Files:

```
FILE-DIMENSION:     '(' ( INTEGER / '*' )
                          [ ',' INTEGER [ ',' INTEGER ]] ')'.
```

b) die maximale Satzlaenge,die mit einem E/A-Aufruf von bzw. auf das File transportiert werden kann.

Der Inhalt des Files:

```
FILE-CONTENTS-MODE:
                    SINGLE-SIMPLE-MODE /
                    STRUCTURE-MODE-DISPLAY.
```

kann in Einheiten von SINGLE-SIMPLE-MODE-Groessen angegeben werden. Die zweite Alternative wird verwendet, wenn die Satzlaenge im PEARL-Quellprogramm ueber eine Strukturbeschreibung festgelegt wird.

Bei Feldern muss zusaetzlich noch die:

```
CONTENTS-DIMENSION:
                    '(' INTEGER-LIST ')'.
```

beruecksichtigt werden.

Die physikalische Satzlaenge ergibt sich somit waehrend der Codegenerierung.

Beispiel:

```
DCL A FILE EXTERNAL INOUT SEQUENTIAL
(*) (200) FIXED(7) GLOBAL;
```

ergibt die CIMIC/P-Anweisungen:

```
SPACE FILE DSECT A;
FTYPE EXT,IOP,SEQ DSECT ;
FCONT (*) DOPE DSECT (200) INT.7;
FILEND ;
```

4.2.8 Vereinbarung von Devices

Device-Deklarationen koennen in CIMIC/P nur innerhalb eines GLOBAL-Blocks auftreten, da alle Anschlussdefinitionen in PEARL automatisch global sind.

Die Syntax einer Device-Deklaration lautet:

```
DEVICE-DECLARATION:
                    ( ( DOPE-MODE ' '
                        GLOBAL-CATEGORY
                        ' ' XSYMB DEVICE-DIMENSION ' '
                        'R ' SYMBOL EOL
                        'SPACE ' DEVICE-MODE ' '
                        GLOBAL-CATEGORY
                        ' ' SYMBOL DEVICE-DIMENSION ) /
                      ( DEVICE-MODE ' '
                        GLOBAL-CATEGORY
                        ' ' XSYMB ) ) EOL
                    DEVICE-MODE-DISPLAY.
DEVICE-DIMENSION:
                    SINGLE-DIMENSION.
DEVICE-MODE-DISPLAY:
                    'DTYPE ' USAGE ' '
                             ( GLOBAL-CATEGORY / G-D-CATEGORY )
                             ' ' EOL
                    'DCONT ' [ DEVICE-CONTENTS-MODE ] ' '
                             ( GLOBAL-CATEGORY / G-D-CATEGOPY )
                             ' ' EOL
                    [ CONNECTION ]
                    'DVCEND '.
```

Die mit DTYPE und DCONT eingeleiteten Zeilen entsprechen denen von FTYPE und FCONT.

Im Gegensatz zu Files

a) koennen sie nur als ein-dimensionale Felder vereinbart werden.

b) ist eine physikalische Anschlussbeschreibung vorgesehen, die den PEARL-System-Teil in detaillierter Form wiedergibt:

```
CONNECTION:
                    ( 'CONN ' ( CONNECTION-POINT /
                                DEVICE-IDENTIFICATION
                                [ DEVICE-DIMENSION ] )
                                ' ' ( 'FROM' / 'TO' / 'FRTO' )
                                ' ' CONNECTION-POINT EOL )$.
```

Ein Anschlusspunkt wird gemaess der Syntax:

```
CONNECTION-POINT:
                    XSYMB  [ '/' INTEGER ]
                    [ ',' INTEGER
                      [ ',' INTEGER
                        [ ',' INTEGER ]]].
```

angegeben. Darin bedeuten die Ganzzahlen von links nach rechts:

- Nummer des Geraetes
- Kanalnummer
- Bitnummer
- Laenge

Beispiel:

Der Geraeteanschluss:

```
DIGIN(5)*3*5,1
```

wird in CIMIC/P in der Form dargestellt:

```
DIGIN/5,3,5,1
```

Werden Devices als formale Parameter vereinbart,entfallen diese Angaben.

4.2.9 Vereinbarung von Interrupts

Die Vereinbarung ist analog zu der von Devices (vgl. Abschnitt 4.2.8):

```
IRPT-DECLARATION:
                  ( ( DOPE-MODE ' '
                      GLOBAL-CATEGORY
                      ' ' XSYMB INTERRUPT-DIMENSION ' '
                      'R ' SYMBOL EOL
                      'SPACE ' INTERRUPT-MODE ' '
                      GLOBAL-CATEGORY
                      ' ' SYMBOL INTERRUPT-DIMENSION ) /
                    ( INTERRUPT-MODE ' '
                      GLOBAL-CATEGORY
                      ' ' XSYMB ) ) EOL
                    CONNECTION
                    'IRPTEND '.
INTERRUPT-DIMENSION:
                    SINGLE-DIMENSION.
```

Es entfallen Contents-Angaben.

Interrupts koennen nicht als formale Parameter auftreten.

4.2.10 Vereinbarung von Signals

a) Implementationsabhaengige Signals
Die Vereinbarung ist analog zu der von Interrupts (vgl. Abschnitt 4.2.9):

```
SIGNAL-DECLARATION:
                    SIGNAL-MODE ' '
                    GLOBAL-CATEGORY
                    ' ' XSYMB EOL
                    [ CONNECTION ]
                    'SIGNEND '.
```

b) Standard-Signals
Sie leiten sich direkt aus der PEARL-Syntax ab, d.h. ihre Vereinbarung entfaellt:

```
STANDARD-SIGNAL:
                    ( 'CONV '     /
                      'OVFL '     /
                      'SUBRG '    /
                      'STRRG '    /
                      'ZDIV '     )     /
                    ( ( 'UNDF ' / 'EOF ' )
                        FILE-IDENTIFICATION ).
```

4.2.11 Vereinbarung von Semaphoren

Sema-Vereinbarungen koennen in PEARL nur auf Modul-Ebene getroffen werden:

```
SEMA-DECLARATION:
                    SEMA-MODE ' '
                    ( GLOBAL-CATEGORY / S-CATEGORY )
                    ' ' ( XSYMB / SYMBOL ).
```

4.2.12 Formatvereinbarungen

Im PEARL-BASIS-SUBSET sind Formatvereinbarungen auf Modulebene eines PEARL-Moduls moeglich. Sie koennen in Ein/Ausgabe-Anweisungen im Koerper von Tasks und Prozeduren angesprochen werden. (sog. R-Format).

Formate, die auf Modulebene vereinbart werden, erhalten die Kategorie STATIC, waehrend Formate, die in Ein/Ausgabe-Anweisungen auftreten, die Kategorie CODE zugeordnet bekommen. Der generelle Aufbau einer Formatliste in CIMIC/P ist in Abschnitt 9.5 beschrieben.

4.3 Spezifikation globaler Groessen

Diese erfolgt nach der Syntax:

```
GLOBAL-SPECIFICATION:
                    'REFER ' EOL
                      ( 'SPCFY ' (
                        SIMPLE-SPECIFICATION              /
                        SIMPLE-ARRAY-SPECIFICATION        /
                        STRUCTURE-SPECIFICATION           /
                        STRUCTURE-ARRAY-SPECIFICATION     /
                        SEMA-SPECIFICATION        /
                        FILE-SPECIFICATION        /
                        DEVICE-SPECIFICATION      /
                        INTERRUPT-SPECIFICATION   /
                        SIGNAL-SPECIFICATION      /
                        TASK-SPECIFICATION        /
                        PROCEDURE-SPECIFICATION )
                      )*
                    'ENDBLK  REFER '.
```

Neben den schon in Abschnitt 4.2 erlaeuterten Deklarationen fuer Strukturen, Files und Devices treten im Spezifikationsblock auch Prozedurvereinbarungen und Taskvereinbarungen auf. Hierzu braucht jeweils nur der Kopf der Vereinbarung angegeben zu werden, da dieser schon die fuer den PEARL-Binder relevanten Angaben enthaelt.

Sofern Device-Spezifikationen in einem GLOBAL-REFER-Block auftreten, so enthalten sie keine Zeilen, die mit der Pseudo-Operation CONN beginnen.

5. Transfer-Anweisungen

Transfer-Anweisungen bewegen Information innerhalb der abstrakten Maschine. In CIMIC/P sind in Transfer-Anweisungen nur elementare Operanden zugelassen.

'LOAD ' S-MODE ' ' OPERAND

Der durch OPERAND spezifizierte Wert wird in den Prozessor und von dort in den Operanden-Stack kopiert.

'STORE' ['(N)'] ' ' S-MODE ' ' VARIABLE .

Das oberste Element des Operanden-Stack wird in den Prozessor und von dort in das durch VARIABLE angegebene Speicherelement uebertragen. Wenn kein N-Flag angegeben wurde, wird das Element danach in den Operandenstack zuruecktransferiert.

'TEST ' CONDITION-CODE ' ' [Variable]

wenn der im Register BEDINGUNGSCODE gespeicherte Wert mit dem in der TEST-Anweisung angegebenen CONDITION-CODE uebereinstimmt, wird durch den Prozessor der Wert M TRUE erzeugt. Andernfalls wird der Wert M FALSE generiert. Der erzeugte Wert wird in den durch VARIABLE angegebenen Speicherplatz gebracht. Fehlt die Speicherplatzangabe, so wird der Wert in den Operandenstack ueberfuehrt.

'BASE ADDR ' [OPERAND]

Der Inhalt der obersten, besetzten Stackzelle bzw. der durch OPERAND angegebene Wert wird in das BASIS-Register gebracht und nicht in den Stack zurueckkopiert.

'INDEX ' SINGLE-INT-MODE [OPERAND]

Der Inhalt der obersten besetzten Stackzelle bzw. der durch OPERAND angegebene Wert wird in das INDEX-Register gebracht.

6. Algorithmische Anweisungen

Eine algorithmische Anweisung besitzt die allgemeine Form:

```
COMPUTATION:
                    OPERATION ' ' [ OPERAND ].

OPERATION:
                    MONADIC-OPERATION /
                    DYADIC-OPERATION.
```

Fehlt die Operandenangabe, so befindet sich der Operand bzw. befinden sich die Operanden im Operandenstack.

6.1 Monadische Operatoren

Bei monadischen Operationen,die zwei durch ein Komma getrennte Modeangaben enthalten,bezieht sich die linke Modeangabe auf den Operanden, waehrend die rechte Modeangabe Typ und Genauigkeit des Resultats der monadischen Operation angibt. Falls die beiden Modes gleich sind, enthaelt die Operation nur eine Modeangabe. In CIMIC/P sind folgende monadische Operationen vorgesehen:

'BNOT ' (SINGLE-INT-MODE /

SINGLE-BIT-MODE)

Der Operand wird bitweise invertiert und das Ergebnis wird im Operandenstack gespeichert.

'TRUNC ' REAL-INT-MODE

Der ganzzahlige Anteil einer rationalen Zahl wird in den Operandenstack gebracht.

'TOINT ' (BIT-INT-MODE / CHAR1-INT-MODE)

Der Operand vom Typ "Bitkette" oder CHAR.1 wird in eine Ganzzahl gewandelt und in den Operandenstack gebracht.

'SIGNUM ' ARITHMETIC-INT-MODE

Das Signum des ganzzahligen oder rationalen Operanden wird in den Operandenstack gebracht.

$$\text{SIGNUM}(X) = \begin{cases} 1 & \text{fuer } X > 0 \\ -1 & \text{fuer } X < 0 \\ 0 & \text{fuer } X = 0 \end{cases}$$

'TOREAL ' INT-REAL-MODE

Der ganzzahlige Operand wird in eine rationale Zahl gewandelt und danach im Operandenstack gespeichert.

'TOCHAR ' INT-CHAR1-MODE

Der ganzzahlige Operand wird in die interne Darstellung eines Zeichens gewandelt. Diese wird im Operandenstack gespeichert.

'TOBIT ' INT-BIT-MODE

Der ganzzahlige Operand wird in eine Bitkette gewandelt. Das Wandlungsergebnis wird im Operandenstack gespeichert.

'NEG ' SINGLE-ARITHMETIC-OR-TIME-MODE

Der ganzzahlige, rationale oder Zeitoperand wird negiert. Das Ergebnis wird in den Operandenstack ueberfuehrt.

6.2 Dyadische Operatoren

Eine dyadische Operation enthaelt zwei durch ein Komma getrennte Modeangaben,wenn die Modes der beiden Operanden verschieden sind.Bei

gleichen Modes wird der Mode der beiden Operanden nur einmal angegeben. Der rechts vom Komma stehende Mode bezieht sich immer auf den Operanden an oberster,der links stehende Mode auf den Operanden an zweitoberster Stelle im Operandenstack (vgl. 2.2b).

('BAND' /'BOR') ' ' (SINGLE-INT-MODE /

SINGLE-BIT-MODE)

Die Operanden werden bitweise mit logischem UND bzw. ODER verknuepft und das Ergebnis wird im Operandenstack abgelegt.

'BNEQV ' SINGLE-BIT-MODE

wie zuvor beschrieben; die Operation ist ein exklusives ODER.

'CMP' ['(N)'] ' ' SIMPLE-MODE

Die Operanden werden verglichen. Im Register BEDINGUNGSCODE wird der Wert LT bzw. EQ bzw. GT gespeichert, wenn der linke Operand kleiner bzw. gleich bzw. groesser als der rechte Operand ist. Fehlt das N-Flag, so wird der linke Operand in den Stack zurueckgespeichert.

'ADD ' ARITHMETIC-OR-TIME-MODE

Die Summe der Operanden wird im Operandenstack abgelegt.

'SUB' ['(R)'] ' ' ARITHMETIC-OR-TIME-MODE

Der rechte Operand wird vom linken Operanden subtrahiert und das Resultat in den Operandenstack gebracht.

'MPY ' ARITHMETIC-OR-TIME-MODE

Das Produkt der Operanden wird im Operandenstack abgelegt.

('DIV' / 'REM') ['(P)'] ' ' ARITHMETIC-OR-TIME-MODE

Der linke Operand wird durch den rechten Operanden dividiert und der Quotient bzw. Rest im Operandenstack abgelegt.

'POWER' ['(R)'] ' ' ARITHMETIC-MODE

Der linke Operand wird mit dem rechten Operanden potenziert und das Ergebnis im Operandenstack abgelegt.

'SHIFTL' ['(R)'] ' ' STRING-AND-INT-MODE

Der linke Operand wird um den Wert des rechten Operanden bei positivem Vorzeichen nach links und bei negativem Vorzeichen nach rechts geschoben und das Ergebnis im Operandenstack abgelegt.

'SHIFTR' ['(R)'] ' ' STRING-AND-INT-MODE

Der linke Operand wird um den Wert des rechten Operanden bei positivem Vorzeichen nach rechts und bei negativem Vorzeichen nach links geschoben und das Ergebnis im Operandenstack abgelegt.

'CSHIFT' ['(R)'] ' ' STRING-AND-INT-MODE

Der linke Operand wird um den Wert des rechten Operanden zirkular geschoben und das Ergebnis im Operandenstack abgelegt.

'INTDIV' ['(R)'] ' ' INT-MODE

Der linke Operand wird durch den rechten Operanden dividiert und der ganzzahlige Anteil des Quotienten auf dem Operandenstack abgelegt.

'FIT ' ['(R)'] ' ' (SINGLE-ARITHMETIC-MODE / SINGLE-STRING-MODE)
',' SINGLE-INT-MODE

Der linke Operand wird bezgl. seiner Genauigkeit bzw. Laenge auf die sich durch den rechten Operanden ergebende Genauigkeit bzw. Laenge gewandelt und das Ergebnis im Operandenstack abgelegt.

'CAT' ['(R)'] ' ' STRING-MODE

Der linke Operand wird mit dem rechten Operanden verkettet und das Ergebnis im Operandenstack gespeichert.

7. Anweisungen fuer die sequentielle Ablaufsteuerung

7.1 Sprunganweisungen

Mittels Sprunganweisungen kann die sequentielle Ausfuehrung von CIMIC/P-Anweisungen unbedingt oder bedingt unterbrochen und der Ablauf einer Task an einer vorgegebenen Stelle fortgesetzt werden.

In einer Sprunganweisung wird ein Sprungziel und wahlweise ein CONDITION-CODE angegeben:

```
'JMP '  [ CONDITION-CODE ]
     ' '  ( 'P '  LABEL / VARIABLE )
```

Die folgende Tabelle gibt zu jedem der sechs zugelassenen CONDITION-CODES die Werte im Register BEDINGUNGSCODE an, die zur Durchfuehrung der bedingten Sprunganweisung fuehren:

CONDITION-CODE	Wert im Register BEDINGUNGSCODE
'EQ'	EQ
'NE'	LT oder GT
'LT'	LT
'LE'	LT oder EQ
'GE'	EQ oder GT
'GT'	GT

wird eine bedingte Sprunganweisung ausgefuehrt, so ist danach der Wert im Register BEDINGUNGSCODE undefiniert. Andernfalls bleibt der Wert in BEDINGUNGSCODE erhalten.

Der Taskablauf wird unbedingt verzweigt, wenn kein CONDITION-CODE angegeben wurde.

7.2 Fallauswahl

In einer Fallauswahl wird eine Anzahl von Adressen:

```
'R ' LABEL
```

angegeben, von denen eine aufgrund eines Vergleichs ausgewaehlt wird. Jeder Adresse ist eine vorgegebene Ganzzahl-Konstante zugeordnet. Wenn der Inhalt des obersten besetzten Stackelements mit einer der Konstanten uebereinstimmt, wird der Taskablauf bei der zugeordneten Adresse fortgesetzt. Andernfalls wird zu der Adresse verzweigt, die in der die Fallauswahl abschliessenden Anweisung angegeben ist.

Eine Fallauswahl wird durch die Anweisung:

```
'BEGCAS ' INT-MODE ' ' [ OPERAND ] EOL
```

eingeleitet. Durch sie wird der Operand in den CIMIC/P-Prozessor transferiert und nicht in den Stack abgelegt. Auf die BEGCAS-Anweisung folgen Anweisungen, in denen Adressen mit Ganzzahl-Konstanten verknuepft werden:

```
'CASE ' INT-MODE 'R ' LABEL ' ' INTEGER-CONSTANT EOL
```

Die in einer Fallauswahl angegebenen Konstanten sind saemtliche voneinander verschieden (und in PEARL-Programmen monoton steigend).

Die Fallauswahl wird mit der Anweisung:

```
'ENDCAS ADDR R ' LABEL
```

beendet. Wie oben schon vermerkt, wird zu der darin angegebenen Marke verzweigt, wenn der Wert des in der BEGCAS-Anweisung angesprochenen

Oneranden mit keiner der Ganzzahl-Konstanten uebereinstimmt.

7.3 ON-Statement

Ueber ein Signal kann der Programmablauf in ein ON-Statement verzweigt werden.

```
ON-STATEMENT:
                ( 'ON '  SIGNAL-MODE
                         ' ' G-CATEGORY ' '
                         ( XSYMB / STANDARD-SIGNAL ) EOL )*
                ( CODE-ELEMENT EOL )$
                'ONEND ' SIGNAL-MODE
                         ' ' G-CATEGORY ' '.
STANDARD-SIGNAL:
                ( 'CONV '     /
                  'OVFL '     /
                  'SUBRG '    /
                  'STRRG '    /
                  'ZDIV '     )     /
                ( ( 'UNDF ' / 'EOF ' )
                    FILE-IDENTIFICATION ).
```

Das ON-Statement wird mit dem Wortsymbol ON eingeleitet. XSYMB bzw. STANDARD-SIGNAL bezeichnet ein Signal. Beim Eintreffen eines Signals wird der Koerper des ON-Statements in Form von CODE-ELEMENTen ausgefuehrt. Soll der Koerper des ON-Statements in Abhaengigkeit von mehreren Signalen durchlaufen werden, wird jedes Signal in jeweils einer Zeile des ON-Kopfes aufgefuehrt. Wird bei der Ausfuehrung des ON-Statements die ONEND-Zeile erreicht, wird das Programm an der Stelle fortgesetzt, an der das Signal verursacht wurde.

7.4 Prozeduraufruf

Ein Prozeduraufruf ermoeglicht eine Programmverzweigung mit automatischer Rueckkehr an die Aufrufstelle. An die Prozedur koennen Parameter uebergeben werden und die Prozedur kann einen Wert an die aufrufende Umgebung zurueckliefern. Eine Aufrufsequenz enthaelt alle Anweisungen zur Verzweigung des Taskablaufs und fuer die Berechnung der

ktuellen Parameter.

ie beginnt mit der Anweisung:

```
( 'N' / 'Q' / 'R' ) 'CALL '

CALL-HEADING EOL


CALL-HEADING:

        [SINGLE-SIMPLE-MODE ] ' ' TARGET.
```

durch die ein Prozeduraufruf der durch TARGET bezeichneten Prozedur eingeleitet wird. NCALL resp. QCALL resp. RCALL steht darin fuer den Aufruf einer nicht reentrant und nicht rekursiven resp. reentrant resp. rekursiven Prozedur. Rekursive Prozeduren werden nur im CIMIC/C verwendet [9]. Die Typangabe SINGLE-SIMPLE-MODE muss bei Funktionsprozeduren stehen.

Auf die Aufruf-Anweisung folgen gegebenenfalls Anweisungen zur Berechnung aktueller Parameter. Dabei sind alle Transfer-Anweisungen, algorithmische Anweisungen und Anweisungen fuer die Ablaufsteuerung zugelassen. Insbesondere kann eine Aufrufsequenz weitere Aufrufsequenzen enthalten.

Jeder aktuelle Parameter wird durch:

```
[ 'N' / 'Q' ] 'ARGIS ' (SIMPLE-MODE / DOPE-MODE /
                        'ADDR'  / STRUCTURE-MODE-IDENTIFICATION
                        / DEVICE-MODE / FILE-MODE )
                        [ OPERAND /
                        ARRAY-OR-SLICE-IDENTIFICATION ].
```

spezifiziert.

Bei Uebergabe von Feldern oder Slices tritt an die Stelle des einfachen Operanden eine

```
ARRAY-OR-SLICE-IDENTIFICATION:
```

```
                ( 'GLOBAL' / 'STATIC' / 'DISPO' )
                ' ' ( XSYMB / SYMBOL )
                '(' ( ( '*' / INTEGER [':' INTEGER] ) // ',' ) ')'
                ' ' ( SINGLE-SIMPLE-MODE / FILE-MODE / DEVICE-MODE
               /    STRUCTURE-MODE-IDENTIFICATION ) .
```

Mit der Anweisung:

```
( 'N' / 'O' / 'R' ) 'CEND ' CALL-HEADING
```

wird eine Aufrufsequenz beendet. Wie bei der Aufruf Anweisung ist die Typangabe nur bei Funktionsprozeduren erforderlich.

Beispiel fuer Prozeduraufrufe:

Aufgerufen werden die in Abschnitt 4.2.5 vereinbarten Prozeduren P1 bis P4.

a) Uebergabe einer einfachen Variablen per Wert:

```
DCL B FIXED(6);
          .
          .
          .
          .
CALL P1(B);
```

ergibt die CIMIC/P-Folge:

```
NCALL ..
NARGIS INT.6,INT.7 DISPO $L70;
NCEND ....
```

Dabei entspricht die links vom Komma stehende Modeangabe dem Mode des aktuellen Parameters,waehrend die rechts stehende Modeangabe dem Mode des formalen Parameters entspricht.Bei Uebereinstimmung wird der Mode der Parameter nur einmal angegeben.

b) Uebergabe eines Feldes per Wert ist wegen Implementierungsschwierigkeiten im PEARL-BASIS-SUBSET nicht enthalten und wird daher hier nicht beschrieben.

c) Uebergabe einer einfachen Variablen per Reference

```
DCL B FIXED(7);
     .
     .
     .
     .
CALL P3(B);
```

ergibt die CIMIC/P-Kommando-Folge:

```
NCALL ..
NARGIS INT.7 I,DISP0 $L20;
NCEND ...
```

d) Uebergabe eines Feldes per Reference

```
DCL B(70,23) FIXED(7);
       .
       .
       .
       .
CALL P4(B);
```

fuehrt zu der CIMIC/P-Folge:

```
NCALL ...
NARGIS DOPE DISP0 $L52(*,*) INT.7;
NCEND ...
```

Die Uebergabe eines Feldes per Reference ist gleichbedeutend mit der Uebergabe seines Dope per Wert.

7.5 Verlassen einer Prozedur

Die Rueckkehr aus einer Prozedur in die aufrufende Prozedur bzw. Task wird durch die Anweisung:

('N' / 'Q' / 'R') 'RETURN ' PROCEDURE-HEADING

bewirkt.

8. Anweisungen fuer die parallele Ablaufsteuerung

In Analogie zu PEARL sind in CIMIC/P parallele Ablaeufe moeglich. Programmteile, die eine kollaterale oder pseudokollaterale Ausfuehrung erlauben, nennen wir Tasks. Im zeitlichen Verlauf bewegen sich Tasks in einem Zustandsraum. Der Uebergang zwischen Zustaenden im Zustandsraum wird durch Operationen fuer die Tasksteuerung veranlasst.

8.1 Anweisungen zur direkten Tasksteuerung

Bei Anweisungen zur direkten Tasksteuerung wird das Betriebssystem unmittelbar beauftragt, einen Zustandswechsel fuer ein Taske vorzunehmen.

```
DIRECT-CONTROL:
                    ACTIVATE-COMMAND /
                    SUSPEND-COMMAND /
                    CONTINUE-COMMAND /
                    TERMINATE-COMMAND /
                    PREVENT-COMMAND .
ACTIVATE-COMMAND:
                    'ACTI ' TASK-MODE ' '
                   [  TASK-IDENTIFICATION ' ' [ PRIORITY ]].
 TASK-MODE:
                   'TASK'.
TASK-IDENTIFICATION:
                    'STATIC ' SYMBOL /
                    'GLOBAL ' XSYMB.
SUSPEND-COMMAND:
                    'SUSP ' TASK-MODE ' ' .
CONTINUE-COMMAND:
                    'CONIT ' TASK-MODE ' ' TASK-IDENTIFICATION.
TERMINATE-COMMAND:
          'STOP' [ ' ' TASK-MODE ' '
          [ TASK-IDENTIFIKATION ]].
PREVENT-COMMAND:
            'PREV ' TASK-MODE ' ' [ TASK-IDENTIFICATION ].
ABORT-COMMAND:
                    'ABORT'.
```

Die Semantik des ACTIVATE-, SUSPEND-, CONTINUE-, TERMINATE-, PREVENT-, COMMAND entspricht der Semantik des ACTIVATE-, SUSPEND-, CONTINUE-, TERMINATE- und PREVENT-Statement von PEARL.

TASK-IDENTIFICATION enthaelt einen Verweis auf den Task-Control-Block, bei modulinternen Tasks in Form von der Anfangsadresse des Task-Control-Blocks; bei modulexternen Tasks in Form des Tasknamens aus dem PEARL-Quelltext. PRIORITY beschreibt die Prioritaet, mit der die Ausfuehrung der angegebenen Task im Betriebssystem durchgefuehrt wird.

Fuer die abnormale Taskbeendigung gibt es das ABORT-COMMAND. Es wird nur in CIMIC/C benutzt.

8.2 Anweisungen fuer die Taskzuteilung

Im Gegensatz zu den Anweisungen zur direkten Tasksteuerung koennen ueber Anweisungen fuer die Taskzuteilung Taskoperationen mit einer eingeplanten Zeitverzoegerung ausgefuehrt werden, d.h. eine Taskoperation wird in Abhaengigkeit von einer Bedingung bzw. von Bedingungen angestossen. Die Bedingung wird formuliert in einem Schedule:

```
SCHEDULED-CONTROL:
                    'SCHREG ' EOL
                        ( SCHEDULE ACTIVATE-COMMAND /
                        SCHEDULE-1 ( CONTINUE-COMMAND /
                                     RESUME-COMMAND )
                        ) EOL
                    'SCHEND ' .
SCHEDULE:
                    SCHEDULE-1 /
                    SCHEDULE-2 .
SCHEDULE-1:
            ( ( CODE-ELEMENT EOL)$
                     ( 'AT CLOCK ' /
                       'AFTER DUR ' ) [ OPERAND ] /
            'WHEN IRPT GLOBAL '
            DEVICE-IDENTIFICATION [ SINGLE-DIMENSION ].
            ) EOL .
SCHEDULE-2:
                    ( CODE-ELEMENT EOL )$
                    'ALL DUR ' [ OPERAND ] EOL
                    [   ( CODE-ELEMENT EOL )$
                        ( 'UNTIL CLOCK ' /
                          'DURING DUR ' ) [ OPERAND ] EOL   ] .
RESUME-COMMAND:
```

'RESU ' TASK-MODE ' ' .

ei SCHEDULE-1 ist eine Bedingung erfuellt zu einer Uhrzeit, nach einem eitintervall oder bei Eintreffen eines Interrupt. Der Zeitpunkt bzw. as Zeitintervall wird in OPERAND angeliefert. Fehlt die OPERAND-ngabe, handelt es sich um einen Ausdruck fuer Zeitangaben; sein Wert, er in CODE-ELEMENT ermittelt wurde, wird dem Stack entnommen. Haengt ie Erfuellung der Bedingung vom Eintreffen eines Interrupt ab, so ist er Interrupt in der DEVICE-DIMENSION angegeben. Die SINGLE-DIMENSION teht bei einem indizierten Interrupt.

n SCHEDULE-2 wird eine Zeitbedingung periodisch nach einer konstanten eitdauer erfuellt. Die Laenge des Zyklus steht bei dem 1. OPERAND der auf dem Stack, wenn die Laenge des Zeitintervalls in Form von einem usdruck angegeben ist. Die entsprechende Konstruktion gilt fuer die eendigung der zyklischen Fortschaltung. OPERAND gibt den Endzeitpunkt bzw. die Enddauer an, zu dem bzw. zu der die Wiederholung abgebrochen ird. Fehlt der OPERAND ist der Wert des Endzeitpunktes oder der nddauer mittels einer Folge von CODE-ELEMENTen auf dem Stack abgelegt.

ine Taskaktivierung kann sowohl in Verbindung mit SCHEDULE-1 und SCHEDULE-2 auftreten, dagegen die Fortsetzung einer Taskausfuehrung bzw. die bedingte Suspendierung einer Task (RESUME-COMMAND) nur in Verbindung mit SCHEDULE-1. Das RESUME-COMMAND bewirkt die Suspendierung der gerade laufenden Task, solange bis die in SCHEDULE-1 angegebene Bedingung erfuellt ist. Sobald die Bedingung erfuellt ist, wird die Task wieder fortgesetzt.

8.3 Unterbrechungsoperationen

In CIMIC/P koennen Unterbrechungssignale (Interrupt oder Signals) gesetzt, gesperrt oder freigegeben werden mit Hilfe der folgenden Unterbrechungsoperationen:

```
INTERRUPTION:
                    INTERRUPT-COMMAND /
                    INDUCE-COMMAND.
INTERRUPT-COMMAND:
                    ( 'ENABLE' / 'DISABLE' ) ' '
                        INTERRUPT-MODE ' ' VARIABLE.
INDUCE-COMMAND:
```

'INDUCE ' SIGNAL-MODE ' ' VARIABLE.

VARIABLE beschreibt im 1. Fall einen Interrupt, im 2. Fall ein Signal. Die DISABLE-Operation sperrt den angegebenen Interrupt aus; die ENABLE-Operation laesst als Gegenstueck zur DISABLE-Operation den angegebenen Interrupt wieder wirksam werden.

Die INDUCE-Operation setzt das angegebene Signal und bewirkt damit die unmittelbare Programmverzweigung in ein ON-Statement.

8.4 Synchronisationsoperationen

Wie im PEARL-BASIS-SUBSET gibt es in CIMIC/P zwei Synchronisationsoperationen fuer Semaphore.

SYNCHRONIZATION:
('REQU' / 'RELE') ' '
SEMA-MODE ' ' VARIABLE .

VARIABLE bezeichnet das Semaphor, auf dem die Synchronisationsoperation ausgefuehrt werden soll. Die Semantik der beiden Synchronisationsoperationen entsprechen der Semantik von PEARL.

9. Anweisungen fuer die Ein/Ausgabe

9.1 Einrichten und Vernichten von Files

Fuer alle Files, die in einem CIMIC/P-Programm benutzt werden, muss eine CREATE-Anweisung durchlaufen werden.Ein File mit einer TITL-Angabe befindet sich entweder in einem permanenten Filesystem oder muss in ein solches ueberfuehrt werden. Ist keine TITL-Angabe vorhanden wird ein temporaeres File eingerichtet, das bei Programmbeendigung wieder verschwindet.

```
CREATE-COMMAND:
                    'CREATE' FILE-OPERAND EOL
                    [ ( CODE-ELEMENT EOL )$
                      'TITL ' SINGLE-CHARACTER-MODE [ OPERAND ]
                        EOL ]
                    'UPON' DEVICE-OPERAND EOL
                    'CRDED '.
FILE-OPERAND:
                    ' ' FILE-MODE ' ' VARIABLE.
DEVICE-OPERAND:
                    ' ' DEVICE-MODE ' ' VARIABLE.
```

Mit Hilfe von FILE-OPERAND wird der Filename angegeben,so wie mit DEVICE-OPERAND das Geraet bezeichnet wird,auf dem das File eingerichtet werden soll. OPERAND nach TITL enthaelt einen Verweis auf den String aus der TITLE-Angabe des PEARL-Programmes.

Das explizite Loeschen eines Files wird ueber das DELETE-COMMAND durchgefuehrt.

```
DELETE-COMMAND:
                    'DELET'  FILE-OPERAND.
```

Hier bezeichnet FILE-OPERAND das zu loeschende File.

9.2 Oeffnen und Schliessen von Files

Wenn exclusiver Zugriff auf ein File gewuenscht wird, muss dieses File ueber die OPEN-Anweisung geoeffnet und ueber die CLOSE-Anweisung wieder geschlossen werden:

```
OPEN-COMMAND:
                    'OPEN'  FILE-OPERAND.
CLOSE-COMMAND:
                    'CLOSE'  FILE-OPERAND.
```

9.3 Unformatierte Ein/Ausgabe

Fuer den Datentransport in interner Darstellung wird in CIMIC/P die READ- und WRITE-Anweisung eingefuehrt.Es wird unterschieden zwischen fortlaufender und positionierter Ein/Ausgabe:

```
READ-OR-WRITE-COMMAND:
                    ( 'READ' / 'WRIT' )  ( 'P' / 'C' )
                                FILE-OPERAND EOL
                    [ POSITION ]
                    DATA-LIST-REFERENCE
                    'TOEND '.
POSITION:
                    (( CODE-ELEMENT EOL )$
                    'POS INT ' [ OPERAND ] EOL )$ .
DATA-LIST-REFERENCE:
                    'RDLIST ADDR T,'
                    LIST-CATEGORY ' ' SYMBOL EOL.
```

Der Zusatz P resp. C hinter READ oder WRIT beschreibt die positionierte resp. fortlaufende Ein/Ausgabe.

In der DATA-LIST-REFERENCE wird durch SYMBOL der Verweis auf die Datenliste gegeben.Der Aufbau der Datenliste wird in Abschnitt 9.5 beschrieben.

Bei positionierter Ein/Ausgabe (nur in Verbindung mit READP oder WRITP) wird bei der angegebenen File-Position der Datentransfer durchgefuehrt. In POSITION koennen bis zu 3 Indizes aufgefuehrt werden. CODE-ELEMENT steht fuer die Aufloesung von Ausdruecken fuer die Indizes. OPERAND gibt einen Index an.

Beispiel

```
        READ FROM BINFILE TO (A,B,C);
```

Dieses PEARL-Statement wird ueberfuehrt in die folgenden CIMIC/P-Anweisungen:

```
        READC FILE STATIC $L204;

        RDLIST ADDR I,CODE $L256;

        IOEND ;
```

$L204 beschreibt das File BINFILE,

$L256 verweist auf den Anfang der Datenliste.

9.4 Formatierte Ein/Ausgabe

Bei der formatierten Ein/Ausgabe werden Daten von interner Darstellung in eine externe bzw. umgekehrt ueberfuehrt.Diese Wandlung der Daten erfolgt entweder listengesteuert oder formatgesteuert. Waehrend bei der listengesteuerten Ein/Ausgabe die Datenwandlung nach den in PEARL festgelegten Standardformaten durchgefuehrt wird,erfolgt sie bei der formatgesteuerten Ein/Ausgabe nach den Formatierungsvorschriften aus der Formatliste.Darueber hinaus gibt es wie im vorigen Abschnitt die fortlaufende und explizite Positionierung:

```
GET-OR-PUT-COMMAND:
                    ( 'GET' / 'PUT' )
                          ( 'FP' / 'LP' / 'FC' / 'LC' )
                          FILE-OPERAND EOL
```

```
                    [ POSITION ]
                    DATA-LIST-REFERENCE
                    [ FORMAT-LIST-REFERENCE ]
                    'IOEND '.
FORMAT-LIST-REFERENCE:
                    'RELIST ADDR I,'
                    LIST-CATEGORY ' ' SYMBOL EOL.
```

GET und PUT werden mit Zusaetzen versehen.Dabei bedeutet:

FP	formatgesteuert und explizit positioniert
LP	listengesteuert und explizit positioniert
FC	formatgesteuert und fortlaufende Positionierung
LC	listengesteuert und fortlaufende Positionierung

DATA-LIST-REFERENCE verweist auf die Datenliste (vgl. Abschnitt 9.5).

POSITION steht nur in Verbindung mit expliziter Positionierung (vgl. Abschnitt 9.3).

FORMAT-LIST-REFERENCE steht nur bei formatgesteuerter Ein/Ausgabe und enthaelt einen Verweis auf eine Formatliste (vgl. Abschnitt 9.5).

Beispiel
---

a)

```
GET FROM INPUT TO (A,B,C);
```

wird uebersetzt zu folgenden CIMIC/P-Anweisungen:

```
GETLC FILE STATIC $L201;

RDLIST ADDR I,CODE $L256;

IOEND ;
```

$L201 beschreibt das File und $L256 die Datenliste.

b)

```
PUT FROM (A,B,C) TO OUTPUT THRU (3)E(13,7);
```

wird in folgende CIMIC/P-Anweisungen uebersetzt:

```
PUTFC FILE STATIC $L202;
RDLIST ADDR I,CODE $L256;
RFLIST ADDR I,CODE $L257;
IOEND ;
```

$L202 beschreibt das File OUTPUT, $L256 verweist auf die Datenliste und $L257 verweist auf die Formatliste.

c)

```
PUT FROM (A,B,C) TO BACKSTORE AT (2*I,6) THRU ((3)E(13,7));
```

wird in folgende CIMIC/P-Anweisungen uebersetzt:

```
PUTFP FILE STATIC $L203;
LOAD INT DISPO $L65;
MPY INT AINT 2;
POS INT;
POS INT AINT 6;
RDLIST ADDR I,CODE $L256;
RFLIST ADDR I,CODE $L257;
IOEND ;
```

$L203 beschreibt das File BACKSTORE, $L256 die Datenliste. In Zeile 2 und 3 wird der Ausdruck 2*I berechnet und in Zeile 4 sein Wert an die 1. Indexposition fuer die Filepositionierung gespeichert. In der Zeile 5 wird die 2. Indexposition besetzt. $L257 aus Zeile 7 uebergibt die Adresse der Formatliste.

9.5 Ein/Ausgabe-Listen

Bei Beginn einer Ein/Ausgabe-Liste wird ein Symbol eingefuehrt, das in den E/A-Statements zur Adressierung der Liste verwendet wird. Man unterscheidet zwischen Daten- und Formatlisten gemaess der folgenden Syntax:

```
TRANSPUT-LISTS:
                    DATA-LIST /
                    FORMAT-LIST .
DATA-LIST:
                    'DLIST'  LIST-HEADING ' ' EOL
                         ( DATA-LIST-ELEMENT EOL ) *
                    'LISED'  LIST-HEADING ' '.
DATA-LIST-ELEMENT:
                    ( CODE-ELEMENT EOL )$
                    'DELM ' ( SINGLE-SIMPLE-MODE / DOPE-MODE /
                              STRUCTURE-MODE-IDENTIFICATION )
                    [ OPERAND / ARRAY-OR-SLICE-IDENTIFICATION ].
LIST-HEADING:
                    ' ' LIST-CATEGORY ' ' SYMBOL.
LIST-CATEGORY:
                    ( 'STATIC' / 'CODE' ).
```

In DATA-LIST werden die Objekte angeordnet und beschrieben,die als Quelle oder Senke einer E/A-Operation verwendet werden.Der Zugriff geschieht in ueblicher Form. So verweist z.B. OPERAND in DATA-LIST-ELEMENT auf das betreffende Objekt (vgl. Abschnitt 3.4). Fehlt eine derartige Angabe steht der durch CODE-ELEMENT bereitgestellte Ausdruck im Operandenstack.

Die LIST-CATEGORY beschreibt die Speicherklasse der Liste.Sie ist STATIC,wenn sich die Ein/Ausgabe-Liste auf Modulebene befindet (bei REMOTE-Formaten). Dagegen ist sie CODE,wenn sich die entsprechende Liste in einer Prozedur oder Task befindet (die Formatangabe ist Teil des E/A-Statements).

In die FORMAT-LIST hingegen werden die Formate des entsprechenden PEARL-Statements uebertragen:

```
FORMAT-LIST:
                    'FLIST'  LIST-HEADING ' ' EOL
                        ( FORMAT-LIST-ELEMENT EOL ) *
                    'LISED'  LIST-HEADING ' '.
FORMAT-LIST-ELEMENT:
                    'FELM FORMAT F ' FORMAT-DENOTATION /
                    REPLICATION-BLOCK .
FORMAT-DENOTATION:
                    CONTROL-FORMAT-DENOTATION /
                    DATA-FORMAT-DENOTATION.
CONTROL-FORMAT-DENOTATION:
                    ( 'LINE' /
                      'X' /
                      'SKIP' /
                      'PAGE' /
                      'COL'    )  SINGLE-DIMENSION .
DATA-FORMAT-DENOTATION:
                    ( 'A' / 'B' / 'B3' / 'B4' )
                                SINGLE-DIMENSION /
                    ( 'E' / 'F' / 'D' / 'T' )
                                '(' INTEGER-LIST ')' .
REPLICATION-BLOCK:
                    'REP ' INTEGER ' ' INTEGER-CONSTANT EOL
                        ( FORMAT-LIST-ELEMENT EOL )*
                    'REPED ' INTEGER EOL.
```

Im REPLICATION-Block beschreibt die INTEGER-Constant den Wiederholungsfaktor fuer die FORMAT-LIST-ELEMENTe. INTEGER dient zur Kennzeichnung der Wiederholungsklammerung.

Beispiel

Die Daten-und Formatliste zu dem PEARL-Statement

```
GET FROM INPUT TO (I,A,J,B,K,C)
THRU (COL(10), (3)((2)X(15), F(10), E(20,7)))
```

sehen wie folgt aus:

```
DLIST  CODE $L205;
DELM INT.15 I,STATIC $L63;
DELM REAL.27 I,DISPO $L198;
DELM INT.15 I,DISPO $L195;
DELM REAL.27 I,STATIC $L58;
DELM INT.15 I,STATIC $L77;
DELM REAL.27 I,DISPO $L312;
LISED  CODE $L205;

FLIST  CODE $L206;
FELM FORMAT F COL(10);
REP 1 AINT 3;
REP 2 AINT 2;
FELM FORMAT F X(15);
REPED 2;
FELM FORMAT F F(10);
FELM FORMAT F E(20,7);
REPED 1;
LISED  CODE $L206;
```

9.6 Prozess-Ein/Ausgabe

Bei der Ein/Ausgabe von Prozesssignalen wird unterschieden zwischen direkter Uebertragung und solcher,der eine Prozedur zugeordnet ist.Hier werden in der Regel implementationsabhaengige Prozeduren angeschlossen,die das jeweilige Prozess-Geraet betreiben.Dieser Unterschied wird durch den Zusatz von einem D bzw. F in der TAKE- und SEND-Anweisung angezeigt:

```
DEVICE-TRANSPUT-COMMANDS:
                    ( 'TAKE' / 'SEND' ) ('D'/'F')
                              DEVICE-OPERAND
                    DATA-LIST-ELEMENT
                    [ CALL EOL ]
                    'IOEND '.
```

Fussnoten

((1)) Das N-Flag ist nicht bei allen Rechenoperationen sinnvoll. Die Operationen, bei denen Flags erlaubt sind, koennen aus der Syntaxnotation (siehe Anhang B) entnommen werden.

((2)) Mit der BASE-Operation kann eine Adresse aus dem Stack in das BASIS-Register geladen werden.

((3)) Mit anderen Worten: es gibt nur Prozeduren auf Modulebene.

((4)) Die Speichervergabe fuer lokale Prozedurdaten erfolgt bei der Codegenerierung relativ zum aktuellen Belegungszeiger fuer den Aktivierungsbereich. Letzterer wird bei Eintreten bzw. Verlassen einer Prozedur dynamisch gesetzt bzw. zurueckgesetzt.

((5)) Die Konstante M ADRSHIFT wird nur in SHIFT-Anweisungen verwendet. Ist der Adressabstand zwischen zwei aufeinander folgenden Speicherworten zur Aufnahme von Ganzzahlen gleich 1 bzw. ist M ADRSHIFT gleich 0 (Wort-Adressierung), so sind alle CIMIC/C-Anweisungen, in denen M ADRSHIFT vorkommt, bei der Codegenerierung zu eliminieren.

Literaturangaben

[1] B.F. Eichenauer:
Spezifikation des PEARL-BASIS-SUBSET
Teil 1: Syntax des PEARL-BASIS-SUBSET
Muenchen, den 8. Juli 1976
GPP mbH

[2] W.M.Waite:
Software Portability via an Intermediate Language
GFK-GI-GMR Fachtagung Prozessrechner 1974
Karlsruhe, 10.-11. Juni 1974
erschienen in:
Lecture Notes in Computer Science, Bd.12,
Springer Verlag 1974

[3] B.Eichenauer, A.Muehlhahn, P.C.Poole,
J.Reh und W.M.Waite:
CIMIC/1 Vorlaeufige Spezifikation
PDV-Entwicklungsnotizen, PDV-E 15 (1973)

[4] M.C.Newey, P.C.Poole and W.M.Waite:
Abstract Machine Modelling to produce

portable Software - a Rewiev and Evaluation.
Software - Practice and Experience, Vol.2,
107-136 (1972)

[5] S.S.Coleman, P.C.Poole and W.M.Waite:
The mobile Programming System, JANUS.
Software - Practice and Experience, Vol.4,
5-23 (1974)

[6] K.Timmesfeld et.al.:
PEARL, A Proposal for a Process- and
Experiment Automation Realtime Language,
Bericht KFK-PDV 1
Gesellschaft fuer Kernforschung mbH Karlsruhe
April 1973

[7] Ueberarbeitung von PEARL zur Erhoehung der
Uebereinstimmung mit PL/1
Bericht PDV-E 13
Gesellschaft fuer Kernforschung mbH Karlsruhe
Januar 1974

[8] B.Eichenauer et.al.:
PEARL, eine prozess- und experimentorientierte
Programmiersprache
Angewandte Informatik 9 73

[9] B.F.Eichenauer:

(PEARL-Compiler-Set)

Spezifikation der Zwischensprache CIMIC/P

PDV-Entwicklungsnotizen, PDV-E 88 (Oktober 1976)

Anhang A: Kurze Beschreibung der Notation zur Darstellung der Syntax von CIMIC/P.

Die nachfolgende beschriebene Notation zur Darstellung der Syntax von CIMIC/P unterscheidet sich nur geringfuegig von BNF und lehnt sich an eine von DeRemer und Frank eingefuehrte Darstellungsmethode an.

A.1 Endsymbole

Ein Endsymbol des Vokabulars wird entweder durch einen Bezeichner, ein Sonderzeichen oder durch ein Literal dargestellt.

A.1.1 Bezeichner

Ein Bezeichner ist eine Folge von Buchstaben, Ziffern und Bindestrichen. Er beginnt und endet mit einem Buchstaben oder einer Ziffer; unmittelbar aufeinander folgende Bindestriche sind nicht zugelassen.

Bezeichner werden zur Identifizierung von Grammatikregeln (vergl. Abschnitt A.2) oder zur Identifizierung von Lexikalsymbolen, d.h. von Symbolen, deren Aufbau innerhalb der Grammatik nicht beschrieben wird, eingesetzt. Treten in einer Grammatik Lexikalsymbole auf, so sind diese nach dem Wortsymbol LEXIKAL-SYMBOLE aufzulisten. Die bei der Beschreibung von CIMIC/P verwendeten Lexikalsymbole sind in Abschnitt B.1 beschrieben.

A.1.2 Sonderzeichen

Die folgenden Sonderzeichen werden zur Darstellung von Operatoren eingesetzt:

: * / + $ ( ) [ ] .

A.1.3 Literale

Literale sind sich selbst definierende Zeichenfolgen. Ein Literal wird in Hochkommata eingeschlossen. Das Hochkomma selbst wird durch zwei aufeinander folgende Hochkommata dargestellt.

A.1.4 Zwischenraeume

Aufeinander folgende Endsymbole sind durch Zwischenraeume gegeneinander abzugrenzen. Ist die Identifizierung aufeinander folgender Endsymbole auch ohne Einschub von Zwischenraeumen moeglich, so duerfen diese entfallen.

A.2 Produktionsregeln

Zwischen den Wortsymbolen GRAMMATIK-REGELN und FINIS wird eine Folge von Produktionsregeln angegeben, mit denen die Syntax von CIMIC/P beschrieben wird.

A.2.1 Aufbau einer Produktionsregel

Eine Produktionsregel besteht aus:

- einem Bezeichner (Regelnamen), ueber den der zugeordnete Regelkoerper in den Regelkoerpern anderer Produktionsregeln angesprochen werden kann,
- einem Doppelpunkt zur Abgrenzung des Regelnamens vom Koerper der Regel,
- den Regelkoerper, in dem die oder ein Teil der darzustellenden Syntax notiert ist und
- einem Punkt, mit dem die Produktionsregel beendet wird.

Es gelten folgende Einschraenkungen:

- Jeder Regelname identifiziert genau einen Regelkoerper.
- Genau ein Regelname wird in keinem Regelkoerper verwendet (Startregel).

A.2.2 Alternativen

Alternativen innerhalb eines Regelkoerpers werden durch Schraegstriche gegeneinander abgegrenzt.

Beispiel: A: B / C.

A produziert entweder B oder C.

A.2.3 Optionen

Stimmen zwei Alternativen bis auf eine bestimmte Symbolfolge ueberein, so kann die fragliche Symbolfolge durch Einschliessen in eckige Klammern als optional gekennzeichnet werden.

Beispiel: Die Produktionsregel

A: B C / B.

kann unter Verwendung von eckigen Klammern auch in der Form:

A: B [C].

dargestellt werden.

A.2.4 Folgen

Eine Folge eines Symbols, in der das Symbol mindestens einmal auftreten muss, kann durch Angabe des Symbols mit nachfolgendem Sternzeichen dargestellt werden:

Beispiel: Die Produktionsregel:

A: B [A].

kann auch in der Form:

A: B* .

geschrieben werden.

Ist auch die Nullfolge zugelassen, so wird statt des Sternzeichens ein Dollarzeichen verwendet.

Beispiel: A: [B*].

ist gleichbedeutend mit:

```
A: B$.
```

4.2.5 Unterteilung

Haeufig treten in Produktionsregeln Aneinandereihungen bestimmter Endsymbole (Symbolketten) auf, die durch eine aus anderen Endymbolen bestehende Kette gegeneinander abgegrenzt sind. Syntaxmuster dieser Art lassen sich bequem unter Verwendung des Operators // darstellen, der als "unterteilt durch" zu lesen ist.

Beispiel: Die Regel

```
A: B / B ',' A.
```

produziert:

```
B
B,B
B,B,B
.
.
.
```

und kann unter Verwendung des Operators // in der Form:

```
A: B // ','.
```

geschrieben werden.

A.2.6 Wahlweise Kettung

Mit dem Operator + kann festgelegt werden, dass sowohl die vor dem Operator, als auch die nach ihm angegebene Symbolkette alleine auftreten darf. Treten beide Konstruktionen gleichzeitig auf, so muss die in der Regel angegebene Reihenfolge eingehalten werden.

Beispiel: A: B [C] / C.

ist gleichbedeutend mit:

A: B + C.

A.2.7 Klammerung

Jedes Auftreten eines Regelnamens im Koerper einer Regel kann durch den dem Regelnamen zugeordneten Regelkoerper ersetzt werden. Falls letzterer nicht nur aus einem Symbol besteht, ist er bei der Ersetzung in runde Klammern einzuschliessen. Wurde ein Regelname ueberall in dieser Weise ersetzt, so kann die durch den Regelnamen identifizierte Produktionsregel entfernt werden.

Die beiden Operatoren // und + sind linksassoziativ. Deshalb koennen Klammernpaare in Regeln, die diese Operatoren enthalten, haeufig weggelassen werden:

Beispiel: Die folgenden drei Definitionen von

A sind gleichbedeutend:

A: X // D. X: B // C.

A: (B // C) // D.

A: B // C // D.

Anhang B: Vollstaendige Syntax von CIMIC/P

B.1 Lexikalsymbole

Wie in Abschnitt B.2 angegebene vollstaendige Syntax von CIMIC/P zeigt werden die folgenden 4 Bezeichner nicht durch die Grammatik-Regel erfasst:

END-OF-LINE: ist ein Steuerzeichen, das im Eingabestrom auftritt und ueblicherweise einem Wagenruecklauf oder einem Kartenende entspricht.

LETTER: steht fuer ein Zeichen des Alphabets.

DIGIT: steht fuer eine Ziffer.

COMMENT: CHARACTER *.

Darin ist CHARACTER eines der folgenden Zeichen:

CHARACTER: LETTER / DIGIT /

'+' / '-' / '*' / '/' /

'(' / ')' / '.' / ',' /

'=' / ' ' / '$' / ';'.

Zwischenraeume sind in CIMIC relevant. Ihre genaue Position wird durch die in Abschnitt B.2 dargestellten Syntaxregeln angegeben.

B.2 Syntaxregeln

Um ein bequemes Nachschlagen zu ermoeglichen, ist im folgenden di komplette Syntax von CIMIC/P angegeben. Das Auffinden einer bestimmte Produktionsregel wird durch die in Abschnitt B.3 angegebene, bzgl. de Regelnamen alphabetisch geordnete Querbezugsliste erleichtert. In ih wird jedem Regelnamen u.a. die Zeilennummer in der Syntaxbeschreibun zugeordnet, in welcher der Regelkoerper beginnt.

```
1       LEXIKAL-SYMBOLE
2
3
4
5         END-OF-LINE
6         LETTER
7         DIGIT
8         COMMENT
9
10
11
12      GRAMMATIK-REGELN
13
14
15
16        EOL: ';' [ COMMENT ] END-OF-LINE [ 'LINE ' INTEGER EOL ].
17
18
19
20        SYMBOL: '$L' DIGIT*.
21
22
23
24        MODE-SYMBOL: '$M' DIGIT*.
25
26
27
28        STRUCTURE-MODE-IDENTIFICATION: MODE-SYMBOL.
29
30
31
32        FILE-IDENTIFICATION:           XSYMB/SYMBOL.
33
34
35
36        DEVICE-IDENTIFICATION:         XSYMB/SYMBOL.
37
38
39
```

```
40      INTEGER: DIGIT*.
41
42
43
44      IDENTIFIER: LETTER (LETTER / DIGIT)$.
45
46
47
48      INTEGER-LIST: INTEGER (',' INTEGER)$.
49
50
51
52      BOUNDS:              '(' INTEGER-LIST ')' .
53
54
55
56      DATA-CONSTANT:       DENOTATION /
57                           MANIFEST .
58
59
60
61      DENOTATION:          SIMPLE-DENOTATION /
62                           BIT-STRING-DENOTATION /
63                           CHARACTER-STRING-DENOTATION.
64
65
66
67      SIMPLE-DENOTATION:
68                           INTEGER-CONSTANT /
69                           REAL-CONSTANT /
70                           DURATION-CONSTANT /
71                           CLOCK-CONSTANT /
72                           LABEL-CONSTANT .
73
74
75
76      INTEGER-CONSTANT:    'AINT ' [ '-' ] INTEGER.
77
78
79
80      REAL-CONSTANT:       'AREAL ' [ '-' ] POSITIVE-REAL-CONSTANT.
81
82
83
84      POSITIVE-REAL-CONSTANT:
85                           INTEGER 'E' [ '-' ] INTEGER.
86
87
88
89      DURATION-CONSTANT:
90                           'ADUR ' TIME-CONSTANT.
91
92
```

```
93
94      TIME-CONSTANT:
95                              INTEGER ':' INTEGER ':'
96                              POSITIVE-REAL-CONSTANT.
97
98
99
100     CLOCK-CONSTANT:
101                             'ACLO ' TIME-CONSTANT.
102
103
104
105     LABEL-CONSTANT:         'R ' LABEL.
106
107
108
109     LABEL:                  SYMBOL.
110
111
112
113     BIT-STRING-DENOTATION:
114                             'B ' INTEGER-LIST BIT-CONTINUATION$.
115
116
117
118     BIT-CONTINUATION:
119                             '/' EOL
120                             'CONT BIT B ' INTEGER-LIST.
121
122
123
124     CHARACTER-STRING-DENOTATION:
125                           'S ' INTEGER-LIST CHARACTER-CONTINUATION$.
126
127
128
129     CHARACTER-CONTINUATION:
130                             '/' EOL
131                             'CONT STR S ' INTEGER-LIST.
132
133
134
135     MANIFEST:               'M ' IDENTIFIER.
136
137
138
139     G-CATEGORY:
140                             'GLOBAL'.
141
142
143
144     S-CATEGORY:
145                             'STATIC'.
```

```
146
147
148
149     D-CATEGORY:
150                         'DISP0'.
151
152
153
154     G-D-CATEGORY:
155                         'GLOBAL' / 'DISP0'.
156
157
158
159     S-D-CATEGORY:
160                         'STATIC' / 'DISP0'.
161
162
163
164     G-S-D-CATEGORY:
165                         'GLOBAL' / 'STATIC' / 'DISP0'.
166
167
168
169     GLOBAL-CATEGORY:
170                         [ 'INV,' ] [ 'I,' ]
171                           ( 'DSECT' / 'RSECT' ).
172
173
174
175     P-CATEGORY:
176                         [ 'INV,' ] [ 'I,' ] 'DISP0'.
177
178
179
180     STRUCTURE-VALUE:
181               ( 'FLDV ' STRUCTURE-ELEMENT-INITIALIZATION EOL )$
182               'LFVDV ' STRUCTURE-ELEMENT-INITIALIZATION.
183
184
185
186     STRUCTURE-ELEMENT-INITIALIZATION:
187               SINGLE-SIMPLE-MODE ' '
188               ( GLOBAL-CATEGORY / S-D-CATEGORY ) ' '
189               SYMBOL ' ' DATA-CONSTANT.
190
191
192
193     ARRAY-VALUE:
194               ( 'ELMV ' ARRAY-ELEMENT-INITIALIZATION EOL )$
195               'ARVND ' ARRAY-ELEMENT-INITIALIZATION.
196
197
198
```

```
199     ARRAY-ELEMENT-INITIALIZATION:
200               S-MODE ' '
201               ( GLOBAL-CATEGORY / S-D-CATEGORY ) ' '
202               '(' INTEGER ')' DATA-CONSTANT.
203
204
205
206     STRUCTURE-ARRAY-VALUE:
207               ( 'ELMV ' STRUCTURE-ARRAY-ELEMENT-INITIALIZATION
208                         STRUCTURE-VALUE EOL )$
209               'ARVND '  STRUCTURE-ARRAY-ELEMENT-INITIALIZATION
210                         STRUCTURE-VALUE.
211
212
213
214     STRUCTURE-ARRAY-ELEMENT-INITIALIZATION:
215               STRUCTURE-MODE-IDENTIFICATION ' '
216               ( GLOBAL-CATEGORY / S-D-CATEGORY ) ' '
217               '(' INTEGER ')' '+' EOL.
218
219
220
221     SINGLE-SIMPLE-MODE:
222                         SINGLE-ARITHMETIC-MODE /
223                         SINGLE-TIME-MODE /
224                         SINGLE-STRING-MODE.
225
226
227
228     SINGLE-ARITHMETIC-MODE:
229                         SINGLE-INT-MODE /
230                         SINGLE-REAL-MODE.
231
232                  .
233
234     SINGLE-INT-MODE:
235                         'INT' [ '.' INTEGER ].
236
237
238
239     SINGLE-REAL-MODE:
240                         'REAL' [ '.' INTEGER ].
241
242
243
244     SINGLE-TIME-MODE:
245                         SINGLE-DURATION-MODE /
246                         SINGLE-CLOCK-MODE.
247
248
249
250     SINGLE-DURATION-MODE: 'DUR' .
251
```

```
252
253
254     SINGLE-CLOCK-MODE: 'CLOCK'.
255
256
257
258     SINGLE-STRING-MODE:
259                         SINGLE-BIT-MODE /
260                         SINGLE-CHARACTER-MODE.
261
262
263
264     SINGLE-BIT-MODE:
265                         'BIT' [ '.' INTEGER ].
266
267
268
269     SINGLE-CHARACTER-MODE:
270                         'STR' [ '.' INTEGER ].
271
272
273
274     RESULT-MODE:        SINGLE-SIMPLE-MODE.
275
276
277
278     S-MODE:
279                         SINGLE-SIMPLE-MODE /
280                         'ADDR'.
281
282
283
284     MODULE:
285                         [ 'LINE ' INTEGER EOL ]
286                         'MODULE ' [ IDENTIFIER ] EOL
287                         ( [ MODULE-ELEMENT ] EOL )$
288                         'MODEND' EOL.
289
290
291
292     MODULE-ELEMENT:
293                         GLOBAL-DECLARATION-OR-SPECIFICATION /
294                         STATIC-DECLARATION /
295                         PROCEDURE-DECLARATION /
296                         TASK-DECLARATION /
297                         FORMAT-LIST.
298
299
300
301     GLOBAL-DECLARATION-OR-SPECIFICATION:
302                         G-CATEGORY ' ' ( GLOBAL-ENTRY /
303                                        ' ' ( GLOBAL-DECLARATION /
304                                              GLOBAL-SPECIFICATION )).
```

```
305
306
307
308     GLOBAL-ENTRY:
309                         XSYMB ' ' LOCAL-ADDR .
310
311
312
313     XSYMB:              IDENTIFIER.
314
315
316
317     LOCAL-ADDR:         ( 'PRENTR' /
318                           'TAENTR' ) ' ' LOCAL-NAME .
319
320
321
322     LOCAL-NAME:         SYMBOL.
323
324
325
326     GLOBAL-DECLARATION:
327                         'DSECT' ' ' EOL
328                           ( 'SPACE '
329                           ( GLOBAL-SIMPLE-DECLARATION          /
330                             GLOBAL-SIMPLE-ARRAY-DECLARATION    /
331                             GLOBAL-STRUCTURE-DECLARATION       /
332                             GLOBAL-STRUCTURE-ARRAY-DECLARATION/
333                             SEMA-DECLARATION                   /
334                             FILE-DECLARATION                   /
335                             DEVICE-DECLARATION                 /
336                             IRPT-DECLARATION                   /
337                             SIGNAL-DECLARATION                  )
338                           )*
339                         'ENDBLK  DSECT '.
340
341
342
343     GLOBAL-SIMPLE-DECLARATION:
344                         SIMPLE-DECLARATION.
345
346
347
348     GLOBAL-SIMPLE-ARRAY-DECLARATION:
349                         SIMPLE-ARRAY-DECLARATION.
350
351
352
353     GLOBAL-STRUCTURE-DECLARATION:
354                         STRUCTURE-DECLARATION.
355
356
357
```

```
358     GLOBAL-STRUCTURE-ARRAY-DECLARATION:
359                         STRUCTURE-ARRAY-DECLARATION.
360
361
362
363
364
365
366     SIMPLE-DECLARATION:
367                         SINGLE-SIMPLE-MODE ' '
368                         ( GLOBAL-CATEGORY / S-D-CATEGORY )
369                         ' ' ( XSYMB / SYMBOL )
370                         [ ' ' DATA-CONSTANT ].
371
372
373
374     SIMPLE-ARRAY-DECLARATION:
375                         DOPE-MODE ' '
376                         ( GLOBAL-CATEGORY / S-D-CATEGORY )
377                         ' ' ( XSYMB / SYMBOL ) BOUNDS ' '
378                         'R ' SYMBOL EOL
379                         'SPACE ' SINGLE-SIMPLE-MODE ' '
380                         ( GLOBAL-CATEGORY / S-D-CATEGORY )
381                         ' ' SYMBOL BOUNDS
382                         [ '+' EOL ARRAY-VALUE ].
383
384
385
386     DOPE-MODE:          'DOPE' .
387
388
389
390     SEMA-MODE:          'SEMA'.
391
392
393
394     FILE-MODE:          'FILE'.
395
396
397
398     DEVICE-MODE:        'DVC'.
399
400
401
402     INTERRUPT-MODE:     'IRPT'.
403
404
405
406     SIGNAL-MODE:        'SIGN'.
407
408
409
410     STRUCTURE-DECLARATION:
```

```
411                         STRUCTURE-MODE-IDENTIFICATION ' '
412                         ( GLOBAL-CATEGORY / S-D-CATEGORY )
413                         ' ' ( XSYMB / SYMBOL )
414                         ( '+' EOL STRUCTURE-VALUE /
415                           EOL STRUCTURE-BODY-DECLARATION ).
416
417
418
419     STRUCTURE-ARRAY-DECLARATION:
420                         DOPE-MODE ' '
421                         ( GLOBAL-CATEGORY / S-D-CATEGORY )
422                         ' ' ( XSYMB / SYMBOL ) BOUNDS ' '
423                         'R ' SYMBOL EOL
424                         'SPACE ' STRUCTURE-MODE-IDENTIFICATION ' '
425                         ( GLOBAL-CATEGORY / S-D-CATEGORY )
426                         ' ' SYMBOL BOUNDS
427                         ( '+' EOL STRUCTURE-ARRAY-VALUE /
428                           EOL STRUCTURE-BODY-DECLARATION ).
429
430
431
432     STRUCTURE-BODY-DECLARATION:
433                         ( 'STCELM ' SINGLE-SIMPLE-MODE ' '
434                                 ( GLOBAL-CATEGORY / S-D-CATEGORY )
435                                 ' ' ( XSYMB/ SYMBOL ) EOL )$
436                         'STCEED ' SINGLE-SIMPLE-MODE ' '
437                                 ( GLOBAL-CATEGORY / S-D-CATEGORY )
438                                 ' ' ( XSYMB/ SYMBOL ).
439
440
441
442     SEMA-DECLARATION:
443                         SEMA-MODE ' '
444                         ( GLOBAL-CATEGORY / S-CATEGORY )
445                         ' ' ( XSYMB / SYMBOL ).
446
447
448
449     FILE-DECLARATION:
450                         FILE-HEADING
451                         FILE-MODE-DISPLAY.
452
453
454
455     FILE-HEADING:
456                         FILE-MODE ' '
457                         ( GLOBAL-CATEGORY / G-S-D-CATEGORY )
458                         ' ' FILE-IDENTIFICATION EOL.
459
460
461
462     FILE-MODE-DISPLAY:
463                         'FTYPE ' FILE-TYPE ' '
```

```
464                                     ( GLOBAL-CATEGORY / S-D-CATEGORY )
465                                     ' ' EOL
466                             'FCONT ' FILE-DIMENSION ' '
467                                     ( FILE-CONTENTS-MODE ' '
468                                      ( GLOBAL-CATEGORY / S-D-CATEGORY )
469                                       ' ' ) /
470                                     ( DOPE-MODE ' '
471                                      ( GLOBAL-CATEGORY / S-D-CATEGORY )
472                                       ' ' CONTENTS-DIMENSION ' '
473                                       FILE-CONTENTS-MODE ) EOL
474                             'FILEND '.
475
476
477
478     FILE-TYPE:
479                             ( 'INL' / 'EXT' ) ','
480                             USAGE ','
481                             ( 'DIR' / 'SEQ' ).
482
483
484
485     USAGE:                  'INP' / 'OUP' / 'IOP'.
486
487
488
489     FILE-DIMENSION:         '(' ( INTEGER / '*' )
490                                  [ ',' INTEGER [ ',' INTEGER ]] ')'.
491
492
493
494     FILE-CONTENTS-MODE:
495     '                       SINGLE-SIMPLE-MODE /
496                             STRUCTURE-MODE-DISPLAY.
497
498
499
500     STRUCTURE-MODE-DISPLAY:
501                             STRUCTURE-MODE-IDENTIFICATION EOL
502                             ( 'STELM ' SINGLE-SIMPLE-MODE ' '
503                                       ( GLOBAL-CATEGORY / S-D-CATEGORY )
504                               'STPEED ' SINGLE-SIMPLE-MODE ' '
505                                       ' ' [ SYMBOL ] EOL )$
506                                       ( GLOBAL-CATEGORY / S-D-CATEGORY )
507                                       ' ' [ SYMBOL ].
508
509
510
511     CONTENTS-DIMENSION:
512                             '(' INTEGER-LIST ')'.
513
514
515
516     DEVICE-DECLARATION:
```

```
570     IRPT-DECLARATION:
571                       ( ( DOPE-MODE ' '
572                           GLOBAL-CATEGORY
573                           ' ' XSYMB INTERRUPT-DIMENSION ' '
574                           'R ' SYMBOL EOL
575                           'SPACE ' INTERRUPT-MODE ' '
576                           GLOBAL-CATEGORY
577                           ' ' SYMBOL INTERRUPT-DIMENSION ) /
578                         ( INTERRUPT-MODE ' '
579                           GLOBAL-CATEGORY
580                           ' ' XSYMB ) ) EOL
581                       CONNECTION
582                       'IRPTEND '.
583
584
585
586     INTERRUPT-DIMENSION:
587                       SINGLE-DIMENSION.
588
589
590
591     SIGNAL-DECLARATION:
592                       SIGNAL-MODE ' '
593                       GLOBAL-CATEGORY
594                       ' ' XSYMB EOL
595                       [ CONNECTION ]
596                       'SIGNEND '.
597
598
599
600     GLOBAL-SPECIFICATION:
601                       'REFER ' EOL
602                         ( 'SPCFY ' (
603                             GLOBAL-SIMPLE-SPECIFICATION              /
604                             GLOBAL-SIMPLE-ARRAY-SPECIFICATION        /
605                             GLOBAL-STRUCTURE-SPECIFICATION           /
606                             GLOBAL-STRUCTURE-ARRAY-SPECIFICATION     /
607                             SEMA-SPECIFICATION       /
608                             FILE-SPECIFICATION       /
609                             DEVICE-SPECIFICATION     /
610                             INTERRUPT-SPECIFICATION  /
611                             SIGNAL-SPECIFICATION     /
612                             TASK-SPECIFICATION       /
613                             PROCEDURE-SPECIFICATION )
614                           )*
615                       'ENDBLK  REFER '.
616
617
618
619     GLOBAL-SIMPLE-SPECIFICATION:
620                       SIMPLE-DECLARATION.
621
622
```

```
623
624     GLOBAL-SIMPLE-ARRAY-SPECIFICATION:
625                         SIMPLE-ARRAY-DECLARATION.
626
627
628
629     GLOBAL-STRUCTURE-SPECIFICATION:
630                         STRUCTURE-DECLARATION.
631
632
633
634     GLOBAL-STRUCTURE-ARRAY-SPECIFICATION:
635                         STRUCTURE-ARRAY-DECLARATION.
636
637
638
639     SEMA-SPECIFICATION:
640                         SEMA-DECLARATION.
641
642
643
644     FILE-SPECIFICATION:
645                         FILE-DECLARATION.
646
647
648
649     DEVICE-SPECIFICATION:
650                       ( ( DOPE-MODE ' '
651                           GLOBAL-CATEGORY
652                           ' ' XSYMB DEVICE-DIMENSION
653                           ' ' DEVICE-MODE ) /
654                         ( DEVICE-MODE ' '
655                           GLOBAL-CATEGORY
656                           ' ' XSYMB ) ) EOL
657                         DEVICE-MODE-DISPLAY.
658
659
660
661     INTERRUPT-SPECIFICATION:
662                       ( ( DOPE-MODE ' '
663                           GLOBAL-CATEGORY
664                           ' ' XSYMB INTERRUPT-DIMENSION
665                           ' ' INTERRUPT-MODE ) /
666                         ( INTERRUPT-MODE ' '
667                           GLOBAL-CATEGORY
668                           ' ' XSYMB ) ) EOL
669                         'IRPTEND '.
670
671
672
673     SIGNAL-SPECIFICATION:
674                         SIGNAL-DECLARATION.
675
```

```
676
677
678     TASK-SPECIFICATION:
679                         TASK-HEAD.
680
681
682
683     PROCEDURE-SPECIFICATION:
684                         PROCEDURE-HEAD.
685
686
687
688     STATIC-DECLARATION:
689                         'SPACE ' (
690                           STATIC-SIMPLE-DECLARATION             /
691                           STATIC-SIMPLE-ARRAY-DECLARATION       /
692                           STATIC-STRUCTURE-DECLARATION          /
693                           STATIC-STRUCTURE-ARRAY-DECLARATION    /
694                           SEMA-DECLARATION                      /
695                           FILE-DECLARATION ).
696
697
698
699     STATIC-SIMPLE-DECLARATION:
700                         SIMPLE-DECLARATION.
701
702
703
704     STATIC-SIMPLE-ARRAY-DECLARATION:
705                         SIMPLE-ARRAY-DECLARATION.
706
707
708
709     STATIC-STRUCTURE-DECLARATION:
710                         STRUCTURE-DECLARATION.
711
712
713
714     STATIC-STRUCTURE-ARRAY-DECLARATION:
715                         STRUCTURE-ARRAY-DECLARATION.
716
717
718
719     SINGLE-DIMENSION:   '(' INTEGER ')'.
720
721
722
723     PROCEDURE-DECLARATION:
724                         PROCEDURE-HEAD
725                         PROCEDURE-BODY.
726
727
728
```

```
729     PROCEDURE-HEAD:
730                         [ 'N' / 'Q' ] 'BEGIN '
731                                       PROCEDURE-HEADING EOL
732                         ( [ 'N' / 'Q' ] 'PARAM '
733                           ( SIMPLE-P-SPECIFICATION                /
734                             SIMPLE-ARRAY-P-SPECIFICATION          /
735                             STRUCTURE-P-SPECIFICATION             /
736                             STRUCTURE-ARRAY-P-SPECIFICATION       /
737                             FILE-P-SPECIFICATION                  /
738                             DEVICE-P-SPECIFICATION                )
739                                                                   )$
740                         ( 'N' / 'Q' / 'R' ) 'PAREND '
741                                       PROCEDURE-HEADING EOL .
742
743
744
745     PROCEDURE-HEADING:
746                         [ RESULT-MODE ] ' '
747                           [ 'R' ] D-CATEGORY
748                           ' ' ( XSYMB / SYMBOL ).
749
750
751
752     SIMPLE-P-SPECIFICATION:
753                         S-MODE ' '
754                         P-CATEGORY ' '
755                         [ SYMBOL ].
756
757
758
759     SIMPLE-ARRAY-P-SPECIFICATION:
760                         DOPE-MODE ' '
761                         P-CATEGORY ' '
762                         [ SYMBOL ]
763                         P-ARRAY-DIMENSION
764                         ' ' SINGLE-SIMPLE-MODE.
765
766
767
768     P-ARRAY-DIMENSION:
769                         '(' [ [ '*' ',' ] '*' ',' ] '*' ')'.
770
771
772
773     STRUCTURE-P-SPECIFICATION:
774                         STRUCTURE-MODE-IDENTIFICATION ' '
775                         P-CATEGORY ' '
776                         [ SYMBOL ] EOL
777                         STRUCTURE-MODE-DISPLAY.
778
779
780
781     STRUCTURE-ARRAY-P-SPECIFICATION:
```

```
782                         DOPE-MODE ' '
783                         P-CATEGORY ' '
784                         [ SYMBOL ]
785                         P-ARRAY-DIMENSION
786                         ' ' STRUCTURE-MODE-IDENTIFICATION EOL
787                         STRUCTURE-MODE-DISPLAY.
788
789
790
791     FILE-P-SPECIFICATION:
792                         FILE-MODE ' '
793                         P-CATEGORY ' '
794                         [ FILE-IDENTIFICATION ] EOL
795                         FILE-MODE-DISPLAY.
796
797
798
799     DEVICE-P-SPECIFICATION:
800                       ( ( DOPE-MODE ' '
801                           P-CATEGORY ' '
802                           [ SYMBOL ]
803                           '(*)'
804                           ' ' DEVICE-MODE ) /
805                         ( DEVICE-MODE ' '
806                           P-CATEGORY ' '
807                           [ SYMBOL ] ) ) EOL
808                         DEVICE-MODE-DISPLAY.
809
810
811
812     PROCEDURE-BODY:
813                         ( CODE-ELEMENT EOL )*
814                         [ 'N' / 'Q' ] 'END '
815                                 [ PROCEDURE-HEADING ].
816
817
818
819     TASK-DECLARATION:
820                         TASK-HEAD
821                         TASK-BODY.
822
823
824
825     TASK-HEAD:
826                         'TASK' ' ' TASK-HEADING
827                         [ ' ' PRIORITY ] EOL.
828
829
830
831     TASK-HEADING:
832                         ' ' [ 'R' ] D-CATEGORY
833                         ' ' ( XSYMB / SYMBOL ).
834
```

```
835
836
837     PRIORITY:
838                         'PRIO ' INTEGER .
839
840
841
842     TASK-BODY:
843                         ( CODE-ELEMENT EOL )*
844                         'TAEND' [ ' ' TASK-HEADING ].
845
846
847
848     CODE-ELEMENT:
849                         LOCAL-DECLARATION /
850                         LABEL-DECLARATION /
851                         STATEMENT .
852
853
854
855     LOCAL-DECLARATION:
856                         LOCAL-SIMPLE-DECLARATION                 /
857                         LOCAL-SIMPLE-ARRAY-DECLARATION           /
858                         LOCAL-STRUCTURE-DECLARATION              /
859                         LOCAL-STRUCTURE-ARRAY-DECLARATION        /
860                         LOCAL-BLOCK-DECLARATION .
861
862
863
864     LOCAL-SIMPLE-DECLARATION:
865                         'SPACE ' SIMPLE-DECLARATION.
866
867
868
869     LOCAL-SIMPLE-ARRAY-DECLARATION :
870                         'SPACE ' SIMPLE-ARRAY-DECLARATION.
871
872
873
874     LOCAL-STRUCTURE-DECLARATION:
875                         'SPACE ' STRUCTURE-DECLARATION.
876
877
878
879     LOCAL-STRUCTURE-ARRAY-DECLARATION:
880                         'SPACE ' STRUCTURE-ARRAY-DECLARATION.
881
882
883
884     LOCAL-BLOCK-DECLARATION:
885                         'BLOCK ' INTEGER ' ' D-CATEGORY ' ' EOL
886                             ( CODE-ELEMENT EOL )$
887                         'BLEND ' INTEGER ' ' D-CATEGORY ' '.
```

```
888
889
890
891     LABEL-DECLARATION:
892                         'LOC ' LABEL.
893
894
895
896     STATEMENT:
897                         TRANSMISSION /
898                         COMPUTATION /
899                         SEQUENTIAL-CONTROL /
900                         ON-STATEMENT /
901                         TASKING-STATEMENT /
902                         TRANSPUT.
903
904
905
906     TRANSMISSION:
907                         'LOAD ' S-MODE ' ' OPERAND /
908                         'STORE' [ '(N)' ] ' ' S-MODE
909                                 ' ' VARIABLE /
910                         'TEST ' CONDITION-CODE ' ' [ VARIABLE ] /
911                         'BASE ADDR ' [ OPERAND ] /
912                         'INDEX  ' INT-MODE  ' ' [ OPERAND ] .
913
914
915
916     OPERAND:
917                         DATA-CONSTANT /
918                         [ 'I,' ] VARIABLE.
919
920
921
922     VARIABLE:
923                         G-S-D-CATEGORY
924                             ' ' REFERENCE /
925                         'BASED' [ MODIFICATION ].
926
927
928
929     REFERENCE:
930                         ( XSYMB / SYMBOL ) [ MODIFICATION ].
931
932
933
934     MODIFICATION:
935                         OFFSET + BIT-SELECTOR.
936
937
938
939     OFFSET:             '(' [FIX-OFFSET] ')'
940                         [ SINGLE-SIMPLE-MODE /
```

```
941                             STRUCTURE-MODE-IDENTIFICATION ].
942
943
944
945     BIT-SELECTOR:       ' BIT ' INTEGER .
946
947
948
949     FIX-OFFSET:         TERM // ( '+' / '-' ).
950
951
952
953     TERM:               [ INTEGER '*' ] SINGLE-SIMPLE-MODE.
954
955
956
957     CONDITION-CODE:
958                         'LT' / 'LE' / 'EQ' / 'NE' / 'GE' / 'GT'.
959
960
961
962     COMPUTATION:
963                         OPERATION ' ' [ OPERAND ].
964
965
966
967     OPERATION:
968                         MONADIC-OPERATION /
969                         DYADIC-OPERATION.
970
971
972
973     MONADIC-OPERATION:
974                         'BNOT '     ( SINGLE-INT-MODE      /
975                                       SINGLE-BIT-MODE )    /
976                         'TRUNC '      REAL-INT-MODE        /
977                         'TOINT '    ( BIT-INT-MODE         /
978                                       CHAR1-INT-MODE )     /
979                         'TOREAL '     INT-REAL-MODE        /
980                         'TOCHAR '     INT-CHAR1-MODE       /
981                         'TOBIT '      INT-BIT-MODE         /
982                         'NEG ' SINGLE-ARITHMETIC-OR-TIME-MODE /
983                         'SIGNUM ' ARITHMETIC-INT-MODE .
984
985
986
987     REAL-INT-MODE:
988                         SINGLE-REAL-MODE ',' SINGLE-INT-MODE.
989
990
991
992     BIT-INT-MODE:
993                         SINGLE-BIT-MODE ',' SINGLE-INT-MODE.
```

```
994
995
996
997     INT-REAL-MODE:
998                         SINGLE-INT-MODE ',' SINGLE-REAL-MODE.
999
1000
1001
1002    INT-CHAR1-MODE:
1003                        SINGLE-INT-MODE ',' 'STR.1'.
1004
1005
1006
1007    CHAR1-INT-MODE:
1008                        'STR.1' ',' SINGLE-INT-MODE.
1009
1010
1011
1012    ARITHMETIC-INT-MODE:
1013                        SINGLE-ARITHMETIC-MODE ',' SINGLE-INT-MODE.
1014
1015
1016
1017    STRING-AND-INT-MODE:
1018                        ( SINGLE-STRING-MODE ',' SINGLE-INT-MODE ) /
1019                        ( SINGLE-INT-MODE ',' SINGLE-STRING-MODE ) .
1020
1021
1022
1023    INT-BIT-MODE:
1024                        SINGLE-INT-MODE ',' SINGLE-BIT-MODE.
1025
1026
1027
1028    SINGLE-ARITHMETIC-OR-TIME-MODE:
1029                        SINGLE-ARITHMETIC-MODE /
1030                        SINGLE-TIME-MODE.
1031
1032
1033
1034    DYADIC-OPERATION:
1035                        ( 'BAND' / 'BOR' ) ' '
1036                        ( SINGLE-INT-MODE / SINGLE-BIT-MODE ) /
1037                        'BNEQV ' SINGLE-BIT-MODE   /
1038                        ( 'SHIFTL' / 'SHIFTR' ) [ '(R)' ] ' ' /
1039                        ( SINGLE-INT-MODE / STRING-AND-INT-MODE ) /
1040                        'CSHIFT' [ '(R)' ] ' ' STRING-AND-INT-MODE /
1041                        ( 'CMP' [ '(N)' ] /
1042                          'ADD' / 'MPY' /
1043                          ('SUB' / 'DIV' / 'REM' ) [ '(R)' ]  )
1044                                    ' ' ARITHMETIC-OR-TIME-MODE /
1045                        'POWER' [ '(R)' ] ' ' ARITHMETIC-MODE /
1046                        'INTDIV' [ '(R)' ]  ' ' INT-MODE  /
```

```
1047                        'CAT' [ '(R)' ] ' ' STRING-MODE .
1048
1049
1050
1051    INT-MODE:
1052                        SINGLE-INT-MODE
1053                           [ ',' SINGLE-INT-MODE ].
1054
1055
1056
1057    ARITHMETIC-MODE:
1058                        SINGLE-ARITHMETIC-MODE
1059                              [',' SINGLE-ARITHMETIC-MODE ] .
1060
1061
1062
1063    ARITHMETIC-OR-TIME-MODE:
1064                        SINGLE-ARITHMETIC-OR-TIME-MODE
1065                         [ ',' SINGLE-ARITHMETIC-OR-TIME-MODE ].
1066
1067
1068
1069    STRING-MODE:
1070                        SINGLE-STRING-MODE
1071                         [ ',' SINGLE-STRING-MODE ].
1072
1073
1074
1075    SEQUENTIAL-CONTROL:
1076                        'JMP ' [ CONDITION-CODE ] ' ' TARGET /
1077                        ( 'BEGCAS ' SINGLE-INT-MODE ' '
1078                                              [ OPERAND ] EOL
1079                             ( 'CASE ' SINGLE-INT-MODE 'R ' LABEL
1080                                          ' ' INTEGER-CONSTANT EOL )*
1081                         'ENDCAS ADDR R ' LABEL )  /
1082                        CALL /
1083                        ( 'N' / 'Q' / 'R' ) 'RETURN '
1084                                  PROCEDURE-HEADING.
1085
1086
1087
1088    TARGET:
1089                        'X ' XSYMB /
1090                        VARIABLE .
1091
1092
1093
1094    CALL:
1095                        ( 'N' / 'Q' / 'R' ) 'CALL '
1096                                  CALL-HEADING EOL
1097                        ( ( CODE-ELEMENT EOL )$
1098                          [ 'N' / 'Q' ] 'ARGIS '
1099                        ( SIMPLE-MODE / DOPE-MODE / 'ADDR' /
```

```
1100                        FILE-MODE / DEVICE-MODE /
1101                        STRUCTURE-MODE-IDENTIFICATION )
1102                        [ OPERAND / ARRAY-OR-SLICE-IDENTIFICATION ]
1103                        EOL )$
1104                        ( 'N' / 'Q' / 'R' ) 'CEND'
1105                                  CALL-HEADING.
1106
1107
1108
1109    ARRAY-OR-SLICE-IDENTIFICATION:
1110                        G-S-D-CATEGORY ' ' ( XSYMB / SYMBOL )
1111                        '(' ( ( '*' / INTEGER
1112                            [ ':' INTEGER ] ) // ',' ) ')'
1113                        ' ' ( SINGLE-SIMPLE-MODE / FILE-MODE /
1114                            DEVICE-MODE /
1115                            STRUCTURE-MODE-IDENTIFICATION ).
1116
1117
1118
1119    SIMPLE-MODE:
1120                        SINGLE-SIMPLE-MODE
1121                          [ ',' SINGLE-SIMPLE-MODE ]  .
1122
1123
1124
1125    CALL-HEADING:
1126                        [ SINGLE-SIMPLE-MODE ] ' ' TARGET.
1127
1128
1129
1130    ON-STATEMENT:
1131                        ( 'ON '  SIGNAL-MODE
1132                                 ' ' G-CATEGORY ' '
1133                                 ( XSYMB / STANDARD-SIGNAL ) EOL )*
1134                        ( CODE-ELEMENT EOL )$
1135                        'ONEND ' SIGNAL-MODE
1136                                 ' ' G-CATEGORY ' '.
1137
1138
1139
1140    STANDARD-SIGNAL:
1141                        ( 'CONV '     /
1142                          'OVFL '     /
1143                          'SUBRG '    /
1144                          'STRRG '    /
1145                          'ZDIV '     )    /
1146                        ( ( 'UNDF ' / 'EOF ' )
1147                            FILE-IDENTIFICATION ).
1148
1149
1150
1151    TASKING-STATEMENT:
1152                        PARALLEL-CONTROL /
```

```
1153                        SYNCHRONIZATION /
1154                        INTERRUPTION .
1155
1156
1157
1158    PARALLEL-CONTROL:
1159                        SCHEDULED-CONTROL /
1160                        DIRECT-CONTROL .
1161
1162
1163
1164    DIRECT-CONTROL:
1165                        ACTIVATE-COMMAND /
1166                        SUSPEND-COMMAND /
1167                        CONTINUE-COMMAND /
1168                        TERMINATE-COMMAND /
1169                        PREVENT-COMMAND /
1170                        ABORT-COMMAND .
1171
1172
1173
1174    ACTIVATE-COMMAND:
1175                        'ACTI ' TASK-MODE ' '
1176                        [ TASK-IDENTIFICATION ' ' [PRIORITY ]].
1177
1178
1179
1180    TASK-MODE:          'TASK'.
1181
1182
1183
1184    TASK-IDENTIFICATION:
1185                        G-D-CATEGORY ' ' ( XSYMB / SYMBOL ).
1186
1187
1188
1189    SUSPEND-COMMAND:
1190                        'SUSP ' TASK-MODE ' ' .
1191
1192
1193
1194    CONTINUE-COMMAND:
1195                        'CONTI ' TASK-MODE ' '
1196                        TASK-IDENTIFICATION.
1197
1198
1199
1200    TERMINATE-COMMAND:
1201                        'STOP'
1202                        [ ' ' TASK-MODE ' '
1203                          [ TASK-IDENTIFICATION ]].
1204
1205
```

```
1206
1207    PREVENT-COMMAND:
1208                        'PREV ' TASK-MODE ' '
1209                        [ TASK-IDENTIFICATION ].
1210
1211
1212
1213    ABORT-COMMAND:
1214                        'ABORT'.
1215
1216
1217
1218    SCHEDULED-CONTROL:
1219                        'SCHBEG ' EOL
1220                            ( SCHEDULE ACTIVATE-COMMAND /
1221                            SCHEDULE-1 ( CONTINUE-COMMAND /
1222                                         RESUME-COMMAND )
1223                            ) EOL
1224                        'SCHEND ' .
1225
1226
1227
1228    SCHEDULE:
1229                        SCHEDULE-1 /
1230                        SCHEDULE-2 .
1231
1232
1233
1234    SCHEDULE-1:
1235                        ( CODE-ELEMENT )$
1236                        ( ( 'AT CLOCK ' /
1237                            'AFTER DUR ' ) [ OPERAND ] ) /
1238                        ( 'WHEN ' INTERRUPT-MODE
1239                                  ' ' VARIABLE ) EOL.
1240
1241
1242
1243    SCHEDULE-2:
1244                        ( CODE-ELEMENT EOL )$
1245                        'ALL DUR ' [ OPERAND ] EOL
1246                        [  ( CODE-ELEMENT EOL )$
1247                           ( 'UNTIL CLOCK ' /
1248                             'DURING DUR ' ) [ OPERAND ] EOL  ] .
1249
1250
1251
1252    RESUME-COMMAND:
1253                        'RESU ' TASK-MODE ' ' .
1254
1255
1256
1257    SYNCHRONIZATION:
1258                        ( 'REQU' / 'RELE' ) ' '
```

```
1259                            SEMA-MODE ' ' VARIABLE .
1260
1261
1262
1263    INTERRUPTION:
1264                        INTERRUPT-COMMAND /
1265                        INDUCE-COMMAND.
1266
1267
1268
1269    INTERRUPT-COMMAND:
1270                        ( 'ENABLE' / 'DISABLE' ) ' '
1271                             INTERRUPT-MODE ' ' VARIABLE.
1272
1273
1274
1275    INDUCE-COMMAND:
1276                        'INDUCE ' SIGNAL-MODE ' ' VARIABLE.
1277
1278
1279
1280    TRANSPUT:
1281                        FILE-MANAGEMENT-COMMANDS /
1282                        FILE-TRANSPUT-COMMANDS /
1283                        TRANSPUT-LISTS /
1284                        DEVICE-TRANSPUT-COMMANDS .
1285
1286
1287
1288    FILE-MANAGEMENT-COMMANDS:
1289                        CREATE-COMMAND /
1290                        OPEN-COMMAND /
1291                        CLOSE-COMMAND /
1292                        DELETE-COMMAND .
1293
1294
1295
1296    CREATE-COMMAND:
1297                        'CREATE' FILE-OPERAND EOL
1298                        [ ( CODE-ELEMENT EOL )$
1299                          'TITL ' SINGLE-CHARACTER-MODE [ OPERAND ]
1300                            EOL ]
1301                        'UPON' DEVICE-OPERAND EOL
1302                        'CRDED '.
1303
1304
1305
1306    FILE-OPERAND:
1307                        ' ' FILE-MODE ' ' VARIABLE.
1308
1309
1310
1311    DEVICE-OPERAND:
```

```
1312                        ' ' DEVICE-MODE ' ' VARIABLE.
1313
1314
1315
1316    OPEN-COMMAND:
1317                        'OPEN'  FILE-OPERAND.
1318
1319
1320
1321    CLOSE-COMMAND:
1322                        'CLOSE'  FILE-OPERAND.
1323
1324
1325
1326    DELETE-COMMAND:
1327                        'DELET'  FILE-OPERAND.
1328
1329
1330
1331    FILE-TRANSPUT-COMMANDS:
1332                        GET-OR-PUT-COMMAND /
1333                        READ-OR-WRITE-COMMAND .
1334
1335
1336
1337    GET-OR-PUT-COMMAND:
1338                        ( 'GET' / 'PUT' )
1339                                ( 'FP' / 'LP' / 'FC' / 'LC' )
1340                                FILE-OPERAND EOL
1341                        [ POSITION ]
1342                        DATA-LIST-REFERENCE
1343                        [ FORMAT-LIST-REFERENCE ]
1344                        'IOEND '.
1345
1346
1347
1348    DATA-LIST-REFERENCE:
1349                        'RDLIST ADDR I,'
1350                        LIST-CATEGORY ' ' SYMBOL EOL.
1351
1352
1353
1354    POSITION:
1355                        (( CODE-ELEMENT EOL )$
1356                        'POS INT ' [ OPERAND ] EOL )$ .
1357
1358
1359
1360    FORMAT-LIST-REFERENCE:
1361                        'RFLIST ADDR I,'
1362                        LIST-CATEGORY ' ' SYMBOL EOL.
1363
1364
```

```
1365
1366    READ-OR-WRITE-COMMAND:
1367                        ( 'READ' / 'WRIT' )  ( 'P' / 'C' )
1368                                    FILE-OPERAND EOL
1369                        [ POSITION ]
1370                        DATA-LIST-REFERENCE
1371                        'IOEND '.
1372
1373
1374
1375    TRANSPUT-LISTS:
1376                        DATA-LIST /
1377                        FORMAT-LIST .
1378
1379
1380
1381    DATA-LIST:
1382                        'DLIST'  LIST-HEADING ' ' EOL
1383                             ( DATA-LIST-ELEMENT EOL ) *
1384                        'LISED'  LIST-HEADING ' '.
1385
1386
1387
1388    DATA-LIST-ELEMENT:
1389                        ( CODE-ELEMENT EOL )$
1390                        'DELM ' ( SINGLE-SIMPLE-MODE / DOPE-MODE /
1391                                  STRUCTURE-MODE-IDENTIFICATION )
1392                        [ OPERAND / ARRAY-OR-SLICE-IDENTIFICATION ].
1393
1394
1395
1396    LIST-HEADING:
1397                        ' ' LIST-CATEGORY ' ' SYMBOL.
1398
1399
1400
1401    LIST-CATEGORY:
1402                        ( 'STATIC' / 'CODE' ).
1403
1404
1405
1406    FORMAT-LIST:
1407                        'FLIST'  LIST-HEADING ' ' EOL
1408                            ( FORMAT-LIST-ELEMENT EOL ) *
1409                        'LISED'  LIST-HEADING ' '.
1410
1411
1412
1413    FORMAT-LIST-ELEMENT:
1414                        'FELM FORMAT F ' FORMAT-DENOTATION /
1415                        REPLICATION-BLOCK .
1416
1417
```

```
1418
1419     FORMAT-DENOTATION:
1420                          CONTROL-FORMAT-DENOTATION /
1421                          DATA-FORMAT-DENOTATION.
1422
1423
1424
1425     CONTROL-FORMAT-DENOTATION:
1426                          ( 'LINE' /
1427                            'X' /
1428                            'SKIP' /
1429                            'PAGE' /
1430                            'COL'    )  SINGLE-DIMENSION .
1431
1432
1433
1434     DATA-FORMAT-DENOTATION:
1435                          ( 'A' / 'B' / 'B3' / 'B4' )
1436                                       SINGLE-DIMENSION /
1437                          ( 'E' / 'F' / 'D' / 'T' )
1438                                       '(' INTEGER-LIST ')' .
1439
1440
1441
1442     REPLICATION-BLOCK:
1443                          'REP ' INTEGER ' ' INTEGER-CONSTANT EOL
1444                              ( FORMAT-LIST-ELEMENT EOL )*
1445                          'REPED ' INTEGER EOL.
1446
1447
1448
1449     DEVICE-TRANSPUT-COMMANDS:
1450                          ( 'TAKE' / 'SEND' ) ('D'/'F')
1451                                    DEVICE-OPERAND
1452                          DATA-LIST-ELEMENT
1453                          [ CALL EOL ]
1454                          'IOEND '.
1455
1456
1457
1458  FINIS
```

B.3 Querbezugsliste der Regelnamen

Die folgende Querbezugsliste besteht aus 3 Feldern. Im ersten Feld sin
unter der Ueberschrift "SYMBOL" die Regelnamen aufgelistet, die bei de
Beschreibung von CIMIC/C verwendet werden. Das zweite Feld mit de
Ueberschrift "DEFINITION" gibt die Zeilennummer in der Syntaxdarstellun
an, in welcher der Regelname definiert ist. Im dritten Feld sind unte
der Ueberschrift "VERWENDUNG" die Zeilennummern angegeben, in denen de
im ersten Feld angegebene Regelname im Koerper einer Syntaxrege
verwendet wird.

SYMBOL	DEFINITION	VERWENDUNG
ABORT-COMMAND	1213	1170
ACTIVATE-COMMAND	1174	1165,1220
ARITHMETIC-INT-MODE	1012	983
ARITHMETIC-MODE	1057	1045
ARITHMETIC-OR-TIME-MODE	1063	1044
ARRAY-ELEMENT-INITIALIZATION	199	194,195
ARRAY-OR-SLICE-IDENTIFICATION	1109	1102,1392
ARRAY-VALUE	193	382
BIT-CONTINUATION	118	114
BIT-INT-MODE	992	977
BIT-SELECTOR	945	935
BIT-STRING-DENOTATION	113	62
BOUNDS	52	377,381,422,426
CALL	1094	1082,1453
CALL-HEADING	1125	1096,1105
CHAR1-INT-MODE	1007	978
CHARACTER-CONTINUATION	129	125
CHARACTER-STRING-DENOTATION	124	63
CLOCK-CONSTANT	100	71
CLOSE-COMMAND	1321	1291
CODE-ELEMENT	848	813,843,886,1097,1134,1235, 1244,1246,1298,1355,1389
COMPUTATION	962	898
CONDITION-CODE	957	910,1076
CONNECTION	553	543,581,595
CONNECTION-POINT	562	554,558
CONTENTS-DIMENSION	511	472
CONTINUE-COMMAND	1194	1167,1221
CONTROL-FORMAT-DENOTATION	1425	1420
CREATE-COMMAND	1296	1289
D-CATEGORY	149	747,832,885,887
DATA-CONSTANT	56	189,202,370,917
DATA-FORMAT-DENOTATION	1434	1421
DATA-LIST	1381	1376
DATA-LIST-ELEMENT	1388	1383,1452
DATA-LIST-REFERENCE	1348	1342,1370

DELETE-COMMAND	1326	1292
DENOTATION	61	56
DEVICE-CONTENTS-MODE	548	540
DEVICE-DECLARATION	516	335
DEVICE-DIMENSION	531	519,523,556,652
DEVICE-IDENTIFICATION	36	555
DEVICE-MODE	398	521,524,653,654,804,805,1100, 1114,1312
DEVICE-MODE-DISPLAY	536	527,657,808
DEVICE-OPERAND	1311	1301,1451
DEVICE-P-SPECIFICATION	799	738
DEVICE-SPECIFICATION	649	609
DEVICE-TRANSPUT-COMMANDS	1449	1284
DIRECT-CONTROL	1164	1160
DOPE-MODE	386	375,420,470,517,571,650,662, 760,782,800,1099,1390
DURATION-CONSTANT	89	70
DYADIC-OPERATION	1034	969
EOL	16	16,119,130,181,194,208,217, 285,286,287,288,327,378,382, 414,415,423,427,428,435,458, 465,473,501,505,520,526,539, 542,558,574,580,594,601,656, 668,731,741,776,786,794,807, 813,827,843,885,886,1078,1080, 1096,1097,1103,1133,1134,1219, 1223,1239,1244,1245,1246,1248, 1297,1298,1300,1301,1340,1350, 1355,1356,1362,1368,1382,1383, 1389,1407,1408,1443,1444,1445, 1453
FILE-CONTENTS-MODE	494	467,473
FILE-DECLARATION	449	334,645,695
FILE-DIMENSION	489	466
FILE-HEADING	455	450
FILE-IDENTIFICATION	32	458,794,1147
FILE-MANAGEMENT-COMMANDS	1288	1281
FILE-MODE	394	456,792,1100,1113,1307
FILE-MODE-DISPLAY	462	451,795
FILE-OPERAND	1306	1297,1317,1322,1327,1340,1368
FILE-P-SPECIFICATION	791	737
FILE-SPECIFICATION	644	608
FILE-TRANSPUT-COMMANDS	1331	1282
FILE-TYPE	478	463
FIX-OFFSET	949	939
FORMAT-DENOTATION	1419	1414
FORMAT-LIST	1406	297,1377
FORMAT-LIST-ELEMENT	1413	1408,1444
FORMAT-LIST-REFERENCE	1360	1343
G-CATEGORY	139	302,1132,1136
G-D-CATEGORY	154	538,541,1185
G-S-D-CATEGORY	164	457,923,1110
GET-OR-PUT-COMMAND	1337	1332

GLOBAL-CATEGORY	169	188,201,216,368,376,380,412, 421,425,434,437,444,457,464, 468,471,503,506,518,522,525, 538,541,572,576,579,593,651, 655,663,667
GLOBAL-DECLARATION	326	303
GLOBAL-DECLARATION-OR-SPECIFICATION	301	293
GLOBAL-ENTRY	308	302
GLOBAL-SIMPLE-ARRAY-DECLARATION	348	330
GLOBAL-SIMPLE-ARRAY-SPECIFICATION	624	604
GLOBAL-SIMPLE-DECLARATION	343	329
GLOBAL-SIMPLE-SPECIFICATION	619	603
GLOBAL-SPECIFICATION	600	304
GLOBAL-STRUCTURE-ARRAY-DECLARATION	358	332
GLOBAL-STRUCTURE-ARRAY-SPECIFICATION	634	606
GLOBAL-STRUCTURE-DECLARATION	353	331
GLOBAL-STRUCTURE-SPECIFICATION	629	605
IDENTIFIER	44	135,286,313
INDUCE-COMMAND	1275	1265
INT-BIT-MODE	1023	981
INT-CHAR1-MODE	1002	980
INT-MODE	1051	912,1046
INT-REAL-MODE	997	979
INTEGER	40	16,48,48,76,85,85,95,95,202, 217,235,240,265,270,285,489, 490,490,563,564,565,566,719, 838,885,887,945,953,1111,1112, 1443,1445
INTEGER-CONSTANT	76	68,1080,1443
INTEGER-LIST	48	52,114,120,125,131,512,1438
INTERRUPT-COMMAND	1269	1264
INTERRUPT-DIMENSION	586	573,577,664
INTERRUPT-MODE	402	575,578,665,666,1238,1271
INTERRUPT-SPECIFICATION	661	610
INTERRUPTION	1263	1154
IRPT-DECLARATION	570	336
LABEL	109	105,892,1079,1081
LABEL-CONSTANT	105	72
LABEL-DECLARATION	891	850
LIST-CATEGORY	1401	1350,1362,1397
LIST-HEADING	1396	1382,1384,1407,1409
LOCAL-ADDR	317	309
LOCAL-BLOCK-DECLARATION	884	860
LOCAL-DECLARATION	855	849
LOCAL-NAME	322	318
LOCAL-SIMPLE-ARRAY-DECLARATION	869	857
LOCAL-SIMPLE-DECLARATION	864	856
LOCAL-STRUCTURE-ARRAY-DECLARATION	879	859
LOCAL-STRUCTURE-DECLARATION	874	858
MANIFEST	135	57
MODE-SYMBOL	24	28
MODIFICATION	934	925,930
MODULE	284	NICHT VERWENDET

MODULE-ELEMENT	292	287
MONADIC-OPERATION	973	968
OFFSET	939	935
ON-STATEMENT	1130	900
OPEN-COMMAND	1316	1290
OPERAND	916	907,911,912,963,1078,1102, 1237,1245,1248,1299,1356,1392
OPERATION	967	963
P-ARRAY-DIMENSION	768	763,785
P-CATEGORY	175	754,761,775,783,793,801,806
PARALLEL-CONTROL	1158	1152
POSITION	1354	1341,1369
POSITIVE-REAL-CONSTANT	84	80,96
PREVENT-COMMAND	1207	1169
PRIORITY	837	827,1176
PROCEDURE-BODY	812	725
PROCEDURE-DECLARATION	723	295
PROCEDURE-HEAD	729	684,724
PROCEDURE-HEADING	745	731,741,815,1084
PROCEDURE-SPECIFICATION	683	613
READ-OR-WRITE-COMMAND	1366	1333
REAL-CONSTANT	80	69
REAL-INT-MODE	987	976
REFERENCE	929	924
REPLICATION-BLOCK	1442	1415
RESULT-MODE	274	746
RESUME-COMMAND	1252	1222
S-CATEGORY	144	444
S-D-CATEGORY	159	188,201,216,368,376,380,412, 421,425,434,437,464,468,471, 503,506
S-MODE	278	200,753,907,908
SCHEDULE	1228	1220
SCHEDULE-1	1234	1221,1229
SCHEDULE-2	1243	1230
SCHEDULED-CONTROL	1218	1159
SEMA-DECLARATION	442	333,640,694
SEMA-MODE	398	443,1259
SEMA-SPECIFICATION	639	607
SEQUENTIAL-CONTROL	1075	899
SIGNAL-DECLARATION	591	337,674
SIGNAL-MODE	486	592,1131,1135,1276
SIGNAL-SPECIFICATION	673	611
SIMPLE-ARRAY-DECLARATION	374	349,625,705,870
SIMPLE-ARRAY-P-SPECIFICATION	759	734
SIMPLE-DECLARATION	366	344,620,700,865
SIMPLE-DENOTATION	67	61
SIMPLE-MODE	1119	1099
SIMPLE-P-SPECIFICATION	752	733
SINGLE-ARITHMETIC-MODE	228	222,1013,1029,1058,1059
SINGLE-ARITHMETIC-OR-TIME-MODE	1028	982,1064,1065
SINGLE-BIT-MODE	264	259,975,993,1024,1036,1037
SINGLE-CHARACTER-MODE	269	260,1299

SINGLE-CLOCK-MODE 254 246
SINGLE-DIMENSION 719 532,587,1430,1436
SINGLE-DURATION-MODE 250 245
SINGLE-INT-MODE 234 229,974,988,993,998,1003,1008,
1013,1018,1019,1024,1036,1039,
1052,1053,1077,1079
SINGLE-REAL-MODE 239 230,988,998
SINGLE-SIMPLE-MODE 221 187,274,279,367,379,433,436,
495,502,504,549,764,940,953,
1113,1120,1121,1126,1390
SINGLE-STRING-MODE 258 224,1018,1019,1070,1071
SINGLE-TIME-MODE 244 223,1030
STANDARD-SIGNAL 1140 1133
STATEMENT 896 851
STATIC-DECLARATION 688 294
STATIC-SIMPLE-ARRAY-DECLARATION 704 691
STATIC-SIMPLE-DECLARATION 699 690
STATIC-STRUCTURE-ARRAY-DECLARATION 714 693
STATIC-STRUCTURE-DECLARATION 709 692
STRING-AND-INT-MODE 1017 1039,1040
STRING-MODE 1069 1047
STRUCTURE-ARRAY-DECLARATION 419 359,635,715,880
STRUCTURE-ARRAY-ELEMENT-INITIALIZATION 214 207,209
STRUCTURE-ARRAY-P-SPECIFICATION 781 736
STRUCTURE-ARRAY-VALUE 206 427
STRUCTURE-BODY-DECLARATION 432 415,428
STRUCTURE-DECLARATION 410 354,630,710,875
STRUCTURE-ELEMENT-INITIALIZATION 186 181,182
STRUCTURE-MODE-DISPLAY 500 496,777,787
STRUCTURE-MODE-IDENTIFICATION 28 215,411,424,501,774,786,941,
1101,1115,1391
STRUCTURE-P-SPECIFICATION 773 735
STRUCTURE-VALUE 180 208,210,414
SUSPEND-COMMAND 1189 1166
SYMBOL 20 32,36,109,189,322,369,377,
378,381,413,422,423,426,435,
438,445,505,507,520,523,574,
577,748,755,762,776,784,802,
807,833,930,1110,1185,1350,
1362,1397
SYNCHRONIZATION 1257 1153
TARGET 1080 1076,1126
TASK-BODY 842 821
TASK-DECLARATION 819 296
TASK-HEAD 825 679,820
TASK-HEADING 831 826,844
TASK-IDENTIFICATION 1184 1176,1196,1203,1209
TASK-NODE 1180 1175,1190,1195,1202,1208,1253
TASK-SPECIFICATION 678 612
TASKING-STATEMENT 1151 981
TERM 953 949
TERMINATE-COMMAND 1200 1168
TIME-CONSTANT 94 90,101

TRANSMISSION	906	897
TRANSMIT	1280	902
TRANSMIT-LISTS	1375	1283
USAGE	485	480,537
VARIABLE	922	909,910,918,1090,1239,1259, 1271,1276,1307,1312
XSYMB	313	32,36,309,369,377,413,422, 435,438,445,519,526,563,573, 580,594,652,656,664,668,748, 833,930,1089,1110,1133,1185
' '		187,188,189,200,201,215,216, 302,303,309,318,327,367,369, 370,375,377,377,379,381,411, 413,420,422,422,424,426,433, 435,436,438,443,445,456,458, 463,465,466,467,469,470,472, 472,502,504,505,507,517,519, 519,521,523,524,526,537,539, 540,542,557,558,571,573,573, 575,577,578,580,592,594,650, 652,653,654,656,662,664,665, 666,669,746,748,753,754,760, 761,764,774,775,782,783,786, 792,793,800,801,804,805,806, 826,827,832,833,844,885,885, 887,887,907,908,909,910,912, 924,963,1035,1038,1040,1044, 1045,1046,1047,1076,1077,1080, 1110,1113,1126,1132,1132,1136, 1136,1175,1176,1185,1190,1195, 1202,1202,1208,1239,1253,1258, 1259,1270,1271,1276,1307,1307, 1312,1312,1350,1362,1382,1384, 1397,1397,1407,1409,1443
' BIT '		945
'$L'		20
'$M'		24
'('		52,202,217,489,512,719,769, 939,1111,1438
'(*)'		933
'(N)'		908,1041
'(R)'		1038,1040,1043,1045,1046,1047
')'		52,202,217,490,512,719,769, 939,1112,1438
'*'		489,769,769,769,953,1111
'+'		217,382,414,427,949
','		48,479,490,490,490,564,565, 566,769,769,988,993,998,1003, 1008,1013,1018,1019,1024,1053, 1059,1065,1071,1112,1121
'-'		76,80,85,949
'.'		235,240,265,270
'/'		119,130,563

':'	95,95,1112
';'	16
'A'	1435
'ABORT'	1214
'ACLO '	101
'ACTI '	1175
'ADD'	1042
'ADDR'	280,1099
'ADUR '	90
'AFTER DUR '	1237
'AINT '	76
'ALL DUR '	1245
'AREAL '	80
'ARGIS '	1098
'ARYND '	195,209
'AT CLOCK '	1236
'B'	1435
'B '	114
'B3'	1435
'B4'	1435
'BAND'	1035
'BASE ADDR '	911
'BASED'	925
'BEGCAS '	1077
'BEGIN '	730
'BIT'	265
'BLENE '	887
'BLOCK '	885
'BNEQV '	1037
'BNOT '	974
'BOR'	1035
'C'	1367
'CALL '	1095
'CASE '	1079
'CAT'	1047
'CEND'	1104
'CLOCK'	254
'CLOSE'	1322
'CMP'	1041
'CODE'	1402
'COL'	1430
'CONN '	554
'CONT BIT B '	120
'CONT STR S '	131
'CONTI '	1195
'CONV '	1141
'CRDED '	1302
'CREATE'	1297
'CSHIFT'	1040
'D'	1437,1453
'DCONT '	540
'DELET'	1327
'DELM '	1300

'DIR' 481
'DISABLE' 1270
'DISPD' 150,155,160,165,176
'DIV' 1043
'DLIST' 1382
'DOPE' 386
'DSECT' 171,327
'DTYPE ' 537
'DUR' 250
'DURING DUR ' 1248
'DVC' 338
'DVCEND ' 544
'E' 85,1437
'ELMV ' 194,207
'ENABLE' 1270
'END ' 814
'ENDBLK DSECT ' 339
'ENDBLK REFER ' 615
'ENDCAS ADDR R ' 1381
'EOF ' 1146
'EQ' 958
'EXT' 479
'F' 1437,1458
'FC' 1339
'FCONT ' 466
'FELM FORMAT F ' 1414
'FILE' 394
'FILEND ' 474
'FLDV ' 181
'FLIST' 1407
'FP' 1339
'FROM' 557
'FRTO' 557
'FTYPE ' 463
'GE' 958
'GET' 1338
'GLOBAL' 140,155,165
'GT' 958
'I,' 170,176,918
'INDEX ' 912
'INDUCE ' 1276
'INL' 479
'INP' 485
'INT' 235
'INTDIV' 1046
'INV,' 170,176
'IOEND ' 1344,1371,1454
'IOP' 485
'IRPT' 482
'IRPTEND ' 582,669
'JMP' 1376
'LC' 1339
'LE' 958

'LFVOV '	182
'LINE'	1426
'LINE '	16,285
'LISED'	1384,1409
'LOAD '	907
'LOC '	892
'LP'	1339
'LT'	958
'M '	135
'MODEND'	288
'MODULE '	286
'MPY'	1042
'N'	730,732,740,814,1083,1095, 1098,1104
'NE'	958
'NEG '	982
'ON '	1131
'ONEND '	1135
'OPEN'	1317
'OUP'	485
'OVFL '	1142
'P'	1367
'PAGE'	1429
'PARAM '	732
'PAREND '	740
'POS INT '	1356
'POWER'	1045
'PRENTR'	317
'PREV '	1208
'PRIO '	838
'PUT'	1338
'Q'	730,732,740,814,1083,1095, 1098,1104
'R'	740,747,832,1083,1095,1104
'R '	185,378,423,520,574,1079
'RDLIST ADDR I,'	1349
'READ'	1367
'REAL'	240
'REFER '	601
'RELE'	1258
'REM'	1043
'REP '	1443
'REPED '	1445
'REQU'	1258
'RESU '	1253
'RETURN '	1083
'RFLIST ADDR I,'	1361
'RSECT'	171
'S '	125
'SCHREG '	1219
'SCHEND '	1224
'SEMA'	390
'SEND'	1450

'SEQ' 481
'SHIFTL' 1038
'SHIFTR' 1038
'SIGN' 406
'SIGNEND ' 596
'SIGNUM ' 983
'SKIP' 1428
'SPACE ' 328,379,424,521,575,689,865,
870,875,880
'SPCFY ' 602
'STATIC' 145,160,165,1402
'STCEED ' 436
'STCELM ' 433
'STELM ' 502
'STOP' 1201
'STORE' 908
'STPEED ' 504
'STR' 270
'STR 1' 1003,1009
'STRRG ' 1144
'SUB' 1043
'SUBRG ' 1143
'SUSP ' 1190
'T' 1437
'TAEND' 844
'TAENTR' 318
'TAKE' 1450
'TASK' 826,1180
'TEST ' 910
'TITL ' 1299
'TO' 557
'TOBIT ' 981
'TOCHAR ' 980
'TOINT ' 977
'TOREAL ' 979
'TRUNC ' 976
'UNDF ' 1146
'UNTIL CLOCK ' 1247
'UPON' 1301
'UHEN ' 1238
'WRIT' 1367
'X' 1427
'X ' 1089
'ZDIV ' 1145

G P P P E A R L - COMPILER PEARL-BASIS-SUBSET

PRELUDE FUER DEN GPP-PEARL-COMPILER
==========

```
1         PRELUDE ;
2            LENGTH     FIXED(15),FLOAT(27),BIT(1),CHAR(1) ;
3         PREEND ;
```

DIAGNOSE 0 FEHLER

ENDE G P P PEARL COMPILER

```
GPP  P E A R L - COMPILER              PEARL-BASIS-SUBSET

SY M B O L T A F E L  (BLOCKWEISE SORTIERT) .
==========================================

 BEZEICHNER              FELD      TYPE         LAENGE  ATTRIBUTE
 ----------------------  --------  ----------   ------  ----------------------


BL O C K - B E G I N N     ZEILE=2  SCHACHTELUNGSTIEFE=0

*LP                                DEVICE                OUTPUT GLOBAL SPEZIFI
                                   INHALT:


*LI                                DEVICE                INPUT GLOBAL SPEZIFIZ
                                   INHALT:


*FPROT                             FILE                  EXTERNAL SEQUENTIAL F
                                   INHALT:
                                   CHAR          1

*FINPUT                            FILE                  EXTERNAL SEQUENTIAL F
                                   INHALT:
                                   CHAR          1

*TMAIN                             TASK                  PRIORITAET=100

*T10                               TASK                  PRIORITAET=10

*T20                               TASK                  PRIORITAET=20

*T30                               TASK                  PRIORITAET=30


*MALOCHEN                          PROZEDUR

                                   F O R M A L E   P A R A M E T E R :

 TNAME                             CHAR          3

                                   P A R A M E T E R - L I S T E   E N D E


BL O C K - B E G I N N     ZEILE=10  SCHACHTELUNGSTIEFE=1
```

```
G P P  P E A R L - COMPILER                PEARL-BASIS-SUBSET

 BEZEICHNER                FELD       TYPE          LAENGE  ATTRIBUTE
 ----------                ----       ----          ------  ---------

B L O C K - B E G I N N      ZEILE=24  SCHACHTELUNGSTIEFE=1


B L O C K - B E G I N N      ZEILE=28  SCHACHTELUNGSTIEFE=1


B L O C K - B E G I N N      ZEILE=32  SCHACHTELUNGSTIEFE=1


B L O C K - B E G I N N      ZEILE=36  SCHACHTELUNGSTIEFE=1
**TNAME                              CHAR          3
**LOOPCOUNT                          FIXED                 INITIAL
**A                                  FIXED
**B                                  FIXED
**C                                  FIXED
**D                                  FIXED


B L O C K - B E G I N N      ZEILE=40  SCHACHTELUNGSTIEFE=2


      ANZAHL DER BLOECKE = 7
      GROESSTE SCHACHTELUNGSTIEFE = 2
```

```
G P P  P E A R L - COMPILER                PEARL-BASIS-SUBSET


1          MODULE BEISPIEL ;
2          PROBLEM ;
3             SPC  LP         DEVICE OUTPUT GLOBAL ;
4             SPC  LI         DEVICE INPUT  GLOBAL ;
5
6             DCL  FPROT      FILE EXTERNAL OUTPUT SEQL (*,72,110) CHAR(1) ;
7             DCL  FINPUT     FILE EXTERNAL INPUT  SEQL (*)        CHAR(1) ;
8          /*****************************************************************
9
10         TMAIN: TASK PRIO 100 ;
11            CREATE FPROT UPON LP ;
12            OPEN FPROT ;
13            CREATE FINPUT UPON LI ;
14            OPEN FINPUT ;
15
16            ACTIVATE T10 ;
17            AFTER 5 SEC RESUME ;
18            ACTIVATE T20 ;
19            AFTER 5 SEC RESUME ;
20            ACTIVATE T30 ;
21         END ;
22         /*****************************************************************
23
24         T10: TASK PRIO 10 ;
25            CALL MALOCHEN( 'T10' ) ;
26         END ;
27         /*****************************************************************
28         T20: TASK PRIO 20 ;
29            CALL MALOCHEN( 'T20' ) ;
30         END ;
31         /*****************************************************************
32         T30: TASK PRIO 30 ;
33            CALL MALOCHEN( 'T30' ) ;
34         END ;
35         /*****************************************************************
36         MALOCHEN: PROC( TNAME CHAR(3) ) ;
37            DCL  LOOPCOUNT FIXED INIT( 0 ) ;
38            DCL( A,B,C,D ) FIXED ;
39
40            REPEAT ;                              /* DAUERBESCHAEFTIGUNG
41                 LOOPCOUNT = LOOPCOUNT + 1 ;
42                 A = -32000 ;
43                 B = -32000 ;
44                 C = -32000 ;
45                 D = -32000 ;
46                 GET FROM  FINPUT
47                     TO(   A, B, C, D ) ;
48                 PUT FROM( TNAME, LOOPCOUNT, A, B, C, D )
49                     TO    FPROT
50                     THRU( COL(10), A(3), (4)( X(5), F(5) ) ) ;
51            END ;
52         END ;
53         /*****************************************************************
54         MODEND ;
55         /*****************************************************************


KEINE FEHLER

ENDE DER UEBERSETZUNG
```

```
/#CIMICP
LINE 1;
MODULE BEISPIEL;
LINE 3;
GLOBAL   REFER ;
SPCFY DVC DSECT LP;
DTYPE OUP DSECT ;
DCONT  DSECT ;
DVCEND ;
SPCFY DVC DSECT LI;
DTYPE INP DSECT ;
DCONT  DSECT ;
DVCEND ;
ENDBLK REFER ;
SPACE FILE STATIC $L42;
FTYPE EXT,OUP,SEQ STATIC ;
FCONT (*,72,110) STR.1 STATIC ;
FILEND ;
SPACE FILE STATIC $L48;
FTYPE EXT,INP,SEQ STATIC ;
FCONT (*) STR.1 STATIC ;
FILEND ;
LINE 10;
TASK  DISPO $L3 PRIO 100;
BLOCK 1 DISPO ;
LINE 11;
CREATE FILE STATIC $L42;
UPON DVC GLOBAL LP;
CRDED ;
LINE 12;
OPEN FILE STATIC $L42;
LINE 13;
CREATE FILE STATIC $L48;
UPON DVC GLOBAL LI;
CRDED ;
LINE 14;
OPEN FILE STATIC $L48;
LINE 16;
ACTI TASK DISPO $L6;
LINE 17;
SCHREG ;
AFTER DUR ADUR 0:0:5E1;
RESU TASK ;
SCHEND ;
LINE 18;
ACTI TASK DISPO $L12;
LINE 19;
SCHREG ;
AFTER DUR ADUR 0:0:5E1;
RESU TASK ;
SCHEND ;
LINE 20;
ACTI TASK DISPO $L18;
LINE 21;
BLEND 1 DISPO ;
TAEND  DISPO $L3;
LINE 24;
TASK  DISPO $L4 PRIO 10;
BLOCK 1 DISPO ;
SPACE STR.3 DISPO $L9 S 84,49,48;
LINE 25;
NCALL  DISPO $L24;
NARGIS STR.3 DISPO $L9;
NCEND  DISPO $L24;
LINE 26;
BLEND 1 DISPO ;
```

```
TASK  DISPO $L12 PRIO 20;
BLOCK 1 DISPO ;
SPACE STR.3 DISPO $L15 S 84,50,48;
LINE 29;
NCALL  DISPO $L24;
NARGIS STR.3 DISPO $L15;
NCEND  DISPO $L24;
LINE 30;
BLEND 1 DISPO ;
TAEND  DISPO $L12;
LINE 32;
TASK  DISPO $L18 PRIO 30;
BLOCK 1 DISPO ;
SPACE STR.3 DISPO $L21 S 84,51,48;
LINE 33;
NCALI  DISPO $L24;
NARGIS STR.3 DISPO $L21;
NCEND  DISPO $L24;
LINE 34;
BLEND 1 DISPO ;
TAEND  DISPO $L18;
LINE 36;
NBEGIN  DISPO $L24;
NPARAM STR.3 DISPO $L27;
NPAREND  DISPO $L24;
BLOCK 1 DISPO ;
SPACE INT.15 DISPO $L54 AINT 0;
SPACE INT.15 DISPO $L57;
SPACE INT.15 DISPO $L60;
SPACE INT.15 DISPO $L63;
SPACE INT.15 DISPO $L66;
LINE 40;
LOC $L2001;
BLOCK 2 DISPO ;
LINE 41;
LOAD INT.15 DISPO $L54;
ADD INT.15,INT.1 AINT 1;
STORE(N) INT.15 DISPO $L54;
LINE 42;
LOAD INT.15 AINT -32000;
STORE(N) INT.15 DISPO $L57;
LINE 43;
LOAD INT.15 AINT -32000;
STORE(N) INT.15 DISPO $L60;
LINE 44;
LOAD INT.15 AINT -32000;
STORE(N) INT.15 DISPO $L63;
LINE 45;
LOAD INT.15 AINT -32000;
STORE(N) INT.15 DISPO $L66;
LINE 47;
JMP  R $L2007;
DLIST  CODE $L69;
DELM INT.15 I,DISPO $L57;
DELM INT.15 I,DISPO $L60;
DELM INT.15 I,DISPO $L63;
DELM INT.15 I,DISPO $L66;
LISED  CODE $L69;
LOC $L2007;
GETLC FILE STATIC $L48;
RDLIST ADDR I,CODE $L69;
IOEND ;
LINE 48;
JMP  R $L2010;
DLIST  CODE $L72;
DELM STR.3 DISPO $L27;
DELM INT.15 DISPO $L54;
DELM INT.15 DISPO $L57;
```

```
FLIST  CODE $L75;
FELM FORMAT F COL(10);
FELM FORMAT F A(3);
REP 1 AINT 4;
FELM FORMAT F X(5);
FELM FORMAT F F(5);
PEPED 1;
LISED  CODE $L75;
LOC $L2010;
PUTFC FILE STATIC $L42;
RDLIST ADDR I,CODE $L72;
RFLIST ADDR I,CODE $L75;
IOEND ;
LINE 51;
BLEND 2 DISPO ;
JMP  R $L2001;
LINE 52;
BLEND 1 DISPO ;
NEND  DISPO $L24;
LINE 54;
MODEND;
/#
```