PDV-E 143
Juli 1980

PDV-Entwicklungsnotizen

Ein PEARL-Betriebssystem für die SIEMENS 310

A. Fleischmann, F.-J. Prester
Physikalisches Institut der
Universität Erlangen-Nürnberg

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

PDV

PDV-E 143

PROJEKT PROZESSLENKUNG MIT DV-ANLAGEN
ENTWICKLUNGSNOTIZ PDV-E 143

EIN PEARL-BETRIEBSSYSTEM FÜR DIE SIEMENS 310

VON
A. FLEISCHMANN
F.-J. PRESTER
PHYSIKALISCHES INSTITUT DER
UNIVERSITAET ERLANGEN-NUERNBERG

259 SEITEN

JULI 1980

1. Einleitung

Im Rahmen der Arbeiten des Physikalischen Instituts III der Universität Erlangen sollte die Realzeitprogrammiersprache PEARL auf einer Siemens 310 implementiert werden. Dabei stellte sich heraus, daß die Funktionen des Standardbetriebssystems der S 310 (ORG 310) zur Realisierung von bestimmten PEARL-Anweisungen nicht ausreichen. Dies betrifft vor allem das Einplanen einer beliebigen Anzahl von parallelen Programmen (Tasks). Deshalb mußte das ORG 310 erweitert werden.

Da aber die meisten Betriebssysteme die von PEARL benötigten Funktionen nicht haben, hat Herr Rössler die nötigen Algorithmen für eine virtuelle Maschine realisiert [ROES79]. Die dabei verwendete assemblerähnliche Programmiersprache hat den Namen SPASS (Systemprogrammiersprache auf Assemblerebene).
Das von Herrn Rössler entworfene PEARL Betriebssystem (PBS) muß bei der Realisierung auf einem Zielrechner in dessen Assemblercode übertragen und durch einige zielrechnerabhängige Assemblerroutinen angepaßt werden. Diese Übertragung und Anpassung wurde bereits für die Siemens 306 und für einen Mikrorechner [ROES78] gemacht. Dabei wurde das PBS jeweils auf die leere Maschine gebracht.

In dieser Arbeit wird beschrieben, wie das PBS auf die Siemens 310 gebracht wurde. Dabei wurde ein neuer Weg eingeschlagen. Das PBS sollte als normales Benutzerprogramm auf der S 310 laufen, ohne daß das ORG 310 geändert werde. Dies hat den Vorteil, daß vorhandene ORG-Leistungen bei der Implementierung der zielrechnerabhängigen Anpassroutinen genutzt werden können. Dazu mußte ein Übersetzungsprogramm geschrieben werden, das SPASS in den Assembler der S 310 (ASS 10) übersetzt, und es mußten die nötigen Anpassroutinen programmiert werden. Um eine spätere Übertragung des PBS auf einen anderen Zielrechner zu erleichtern, wurde der Übersetzungsvorgang in zwei Teile aufgespalten. Der erste Teil ist die zielrechnerunabhängige Vorübersetzung: Diese führt die Syntaxprüfung des SPASS-Programms durch und erzeugt einen Zwischencode, der leichter weiterverarbeitet werden kann. Der zweite Teil ist die zielrechnerabhängige Assemblertexterzeugung: Diese verarbeitet die Zwischeninformation weiter und erzeugt den Assemblertext.

2. Gegebener Hintergrund

2.1. Beschreibung der Programmiersprache PEARL

Die Programmiersprache PEARL hat eine modulare Programmstruktur. Ein Modul besteht aus dem Systemteil und/oder dem Problemteil.
Im Systemteil wird die benötigte Konfiguration beschrieben. Der Problemteil ist eine Zusammenfassung von "Tasks" und Prozeduren sowie von globalen Größen. Jeder Modul ist einzeln übersetzbar. Task-und Prozedurrümpfe haben eine konventionelle Blockstruktur.
Als "Task" wird der dynamische Ablauf einer Folge von PEARL-Anweisungen unter der Kontrolle des Betriebssystems verstanden. Werden in einem Modul nur Prozeduren deklariert und sind diese Prozeduren auch von anderen Moduln aus ansprechbar, so spricht man von einem Prozedurmodul. Prozeduren und Variable, die in einem anderen Modul global (d.h. diese Einheiten sind auch außerhalb des Moduls bekannt) deklariert werden, können in einem anderen Modul durch entsprechende Spezifikation bekanntgemacht werden, und zwar am Anfang des Problemteils. Dadurch kann ein Programmierer leicht feststellen, welche Prozeduren und Variable aus anderen Moduln benutzt werden. Durch Spezifikation erhält ein Modul Zugriffsrechte auf Größen und Prozeduren anderer Module. Im ASME-PEARL-Subset können maximal 9 solcher Prozedurmodule zu der steuernden Task dazugebunden werden. Näheres über die PEARL-Programmstruktur ist [ASME76] zu entnehmen. Im folgenden wird in dieser Arbeit der Begriff Modul im Sinne eines PEARL-Moduls gebraucht.

Ein PEARL-Programmsystem zur Steuerung und Auswertung von Experimenten und Prozessen besteht meistens aus mehreren Programmteilen (Tasks), die autonom ablaufen, aber sich unter Umständen koordinieren müssen.

Diese Tasks können sich in verschiedenen Zuständen befinden (siehe Bild 2.1) In PEARL gibt es Anweisungen, um für eine Task einen Zustandswechsel zu veranlassen. Einen solchen Zustandswechsel kann eine Task für sich selbst herbeiführen (nur wenn sie laufend ist) oder er kann durch eine andere Task veranlaßt werden.

Für die Prozeßprogrammierung ist es wesentlich, asynchron ablaufende Aufgaben einplanen zu können, vor allem müssen plötzlich auftretende Ereignisse (Interrupts) beantwortet werden.

Für die Synchronisierung von asynchronen Prozessen gibt es in PEARL Semaphoroperationen (im Dijkstra'schen Sinne).
Einen Überblick über die verschiedenen Zustände einer Task, sowie die Ursachen für einen Wechsel gibt das Bild 2.1 Dabei stehen in den Kästen der Knoten die Taskzustände und an den Kanten die Ursachen (Anweisungen) für einen Zustandswechsel. In dieser Skizze wird nicht berücksichtigt, daß einige dieser Zustandswechsel auch eingeplant werden können.

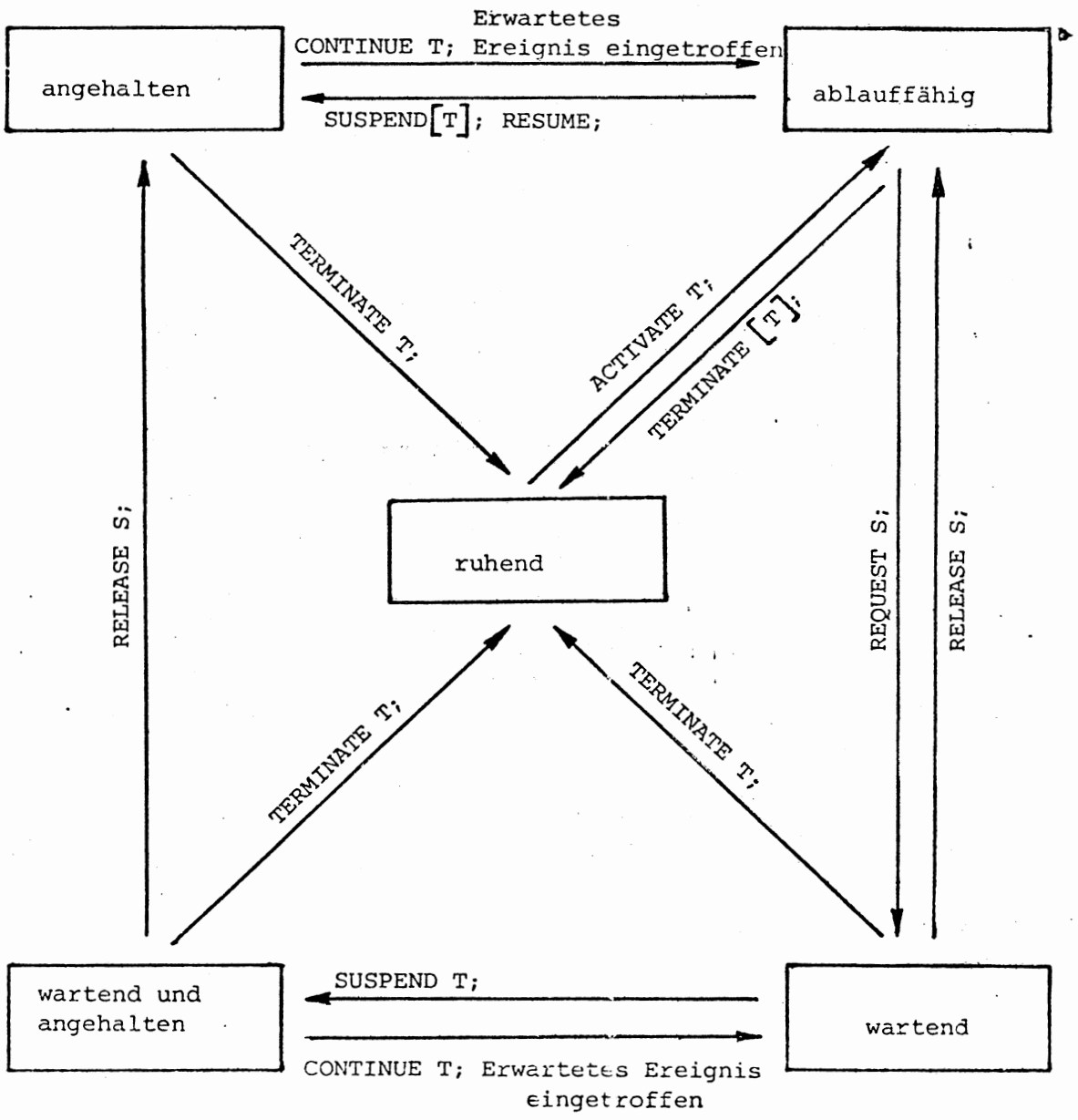


Bild 2.1.: Mögliche Übergänge von Taskzuständen

Beschreibung der Taskzustände:

- ruhend

  Die Task ist im System vorhanden und belegt, außer dem beim Laden zugeteilten Speicherplatz, keine anderen Betriebsmittel.

- ablauffähig

  Der Task ist der Zentralprozessor zugeteilt (laufende Task).

  Die Task wartet auf ein Betriebsmittel, das vom Betriebssystem vergeben wird.

  Die Task benutzt ein vom Betriebssystem zugeteiltes Betriebsmittel.

- angehalten

  Die Task kann auch dann nicht weiterlaufen, wenn ihr der Zentralprozessor zugeteilt wird. Aus diesem Zustand kann die Task nur durch eine andere Task oder durch ein erwartetes Ereignis befreit werden.

- wartend

  Die Task wird wartend, wenn ihr Ablauf über Semaphore gesteuert wird. Eine Task kann nur durch Eigeninitiative wartend werden.

- wartend und angehalten

  Eine Task wird in diesen Zustand versetzt, wenn eine wartende Task durch eine andere Task angehalten wird. Der Weiterlauf erfolgt erst dann, wenn die Warte- und die Anhaltebedingung aufgehoben ist.

PEARL enthält außerdem Anweisungen zur Übertragung von Daten zur Standard- und Prozeßperipherie, sowie Anweisungen zur Dateiverwaltung. Die Übertragung zur Standardperipherie (Drucker, Blattschreiber etc.) geschieht zeichenweise formatiert. Bei der Ein/Ausgabe der Prozeßperipherie werden die Daten satzweise unformatiert (Binär) übertragen.

Während einer Übertragung ist es durch die Angabe einer Option möglich, Transformationen auf dem Datenstring auszuführen, oder Steuerinformationen an die Geräte weiterzuleiten (z.B. Gerät einschalten).

Graphische Informationsinhalte können mit PEARL-Anweisungen von und zu graphischen Geräten transferiert werden.

Die Ausführung der beschriebenen Zustandswechsel, deren Einplanung, sowie die Datenübertragungen bei der Ein/Ausgabe sind Aufgabe des Betriebssystems. Dies stellt bestimmte Anforderungen an das Betriebssystem, die in Kapitel 2.3. beschrieben werden.

Eine ausführliche Beschreibung des ASME-PEARL-Subsets findet sich in [ASME76].

2.2. Beschreibung der SIEMENS 310

2.2.1. Übersicht über die Hardwarestruktur der S310

Die Siemens Rechenanlage S310 ist eine 16-Registermaschine mit 16 Bit Wortlänge. Die S310 hat 16 Programmebenen. Jeder dieser Programmebenen ist eine feste Priorität zugeordnet. Jede Ebene besitzt einen eigenen Satz von Rechenregistern. Wird aufgrund eines Ereignisses (Interrupt) die Programmebene gewechselt, d.h. wird einer anderen Ebene der Prozessor zugeteilt, so wird mit dem Registersatz dieser Programmebene weitergerechnet. Die Registerinhalte müssen also bei einem Ebenenwechsel nicht gerettet werden. Eine Ausnahme bilden dabei die Register RØ und R1. In das Register RØ wird das Programmzustandsregister (PZR) der zugehörigen Ebene gerettet und in das Register R1 der Befehlszählerstand der unterbrochenen Programmebene.

Bekommt eine Ebene den Prozessor zugeteilt, wird der Inhalt von R1 aus der entsprechenden Registertafel in den Befehlszähler übernommen und der Inhalt von RØ in das PZR übertragen. Die 16 Registersätze à 16 Register sind auch über die Adressen 0 - 255 ansprechbar.
Das sogenannte Tafelzeigerregister (TZR) zeigt auf das Register RØ der Programmebene, der der Prozessor zugeteilt ist.

Eine Beschreibung der Hardwarestruktur der S 310 findet sich in [ZE3105] .

2.2.2. Überblick über die Systemsoftware der S 310 [ORG 310]

Das Betriebssystem der S 310 benötigt in der zur Verfügung stehenden Version 10 Programmebenen. Die drei höchstprioren Ebenen werden vom Systemkern beansprucht. Jedes angeschlossene Peripheriegerät belegt eine weitere Ebene. An der zur Verfügung stehenden Anlage sind angeschlossen:
Blattschreiberausgabe, zwei Floppy-Disk-Laufwerke, eine Prozeßeingabe, eine Prozeßausgabe, ein Drucker und eine Teletypeschnittstelle, die für die Kopplung zu einer S 306 verwendet wird.
Letztlich stehen dem Benutzer die Ebenen zehn bis fünfzehn zur Verfügung. Ein Anwender kann also bestenfalls gleichzeitig sechs Programme parallel arbeiten lassen. Um den Parallellauf von Programmen koordinieren zu können, stellt das ORG sogenannte Koordinierungszähler (KOOR) zur Verfügung. Diese haben im wesentlichen die gleiche Funktion wie Dijkstrasche Semaphore. Mit diesen Koordinierungszählern ist es aber auch möglich, Interruptreaktionen einzuplanen. Durch den Aufruf entsprechender Systemmakros wird eine Programmebene vorbereitet, beim Eintreffen eines entsprechenden Interrupts laufend (lauffähig) zu werden. Mit Hilfe von Bedienanweisungen kann von einer Programmebene aus ein auf einer anderen Ebene geladenes Programm gestartet werden, bzw. lauffähig gemacht werden. Durch Systemaufrufe ist es möglich, Zeiteinplanungen vorzunehmen, d.h. daß eine Ebene nach Ablauf einer bestimmten Zeit bzw. zu einem festen Zeitpunkt aktiv wird.

Ein Programm (Ebene) kann sich durch Systemaufrufe selbst beenden. Das Beenden eines Programms ist von einer anderen Ebene aus nicht möglich.
Die Möglichkeiten des ORG 310 sind im folgenden noch einmal zusammengefaßt dargestellt; dabei wurde die Syntaxnotation von PEARL verwendet.

WHEN interrupt RESUME;
ALL duration ACTIVATE task;
AFTER duration ACTIVATE task;
TERMINATE:
REQUEST semaphore;
RELEASE semaphore;
Maximal 6 Programme können parallel arbeiten.

2.3. Anlagenunabhängiges PBS

2.3.1. Allgemeines

Die Funktionen von Realzeitbetriebssystemen reichen in vielen Fällen nicht aus, den Anforderungen von PEARL gerecht zu werden. Ein Beispiel dafür ist das im vorigen Kapitel beschriebene Betriebssystem ORG 310 der Siemens 310. Deshalb wurden die Verwaltungsfunktionen, die PEARL von einem Betriebssystem verlangt, rechnerunabhängig auf einer abstrakten Maschine formuliert. Dieses PEARL-Betriebssystem (PBS) muß bei der Implementierung auf einer realen Maschine noch in die entsprechende Maschinensprache übertragen werden. Die Formulierung des PBS wurde von Herrn Rössler im Rahmen seiner Dissertationsschrift [ROES79] gemacht.

Das PBS kann sowohl als Betriebssystemkern für eine leere Maschine (realisiert von Herrn Rössler an der S 306 und einem Microcomputer mit einem Z80 Microprozessor) als auch als Ergänzung für ein vorhandenes Betriebssystem verwendet werden, wie es im Rahmen dieser Arbeit beschrieben wird.

Durch den modularen Aufbau können für eine konkrete Anwendung (PEARL Subset) nicht benötigte PBS Bausteine problemlos entfernt, oder in ihrem Funktionsumfang eingeschränkt werden. An der Siemens 306 wurde das PBS in einem Funktionsumfang implementiert, wie er von Full-PEARL [FULL77] benötigt wird. Am Microrechner Z80 wurde der Funktionsumfang stark eingeschränkt [ROES78] . Es wurde dort z.B. die ganze Ein/Ausgabe-Verwaltung weggelassen und der Leistungsumfang des Bausteins Taskverwaltung stark eingeengt.

Für die S 310 wurde die Taskverwaltung so geändert, daß das PBS ungefähr den Leistungsumfang hat, wie er vom ASME-PEARL-Subset [ASME76] gefordert wird. Im folgenden wird der Funktionsumfang der Taskverwaltung für die S 310 in PEARL Notation dargestellt.

```
[WHEN interrupt / AFTER duration]
[ALL duration] ACTIVATE task;
[WHEN interrupt / AFTER duration] RESUME;
 SUSPEND task  ;
[WHEN Interrupt / AFTER duration] CONTINUE task;
 TERMINATE [task] ;
 REQUEST  semaphore;
 RELEASE  semaphore;
 PREVENT [task] ;
```

Das PBS ermöglicht auch, daß prinzipiell beliebig viele Tasks parallel laufen können.

2.3.2. Beschreibung der PBS Bausteine

Einen Überblick über die PBS-Bausteine und deren Zusammenwirken gibt Bild 2.2 das [ROES79] entnommen ist.

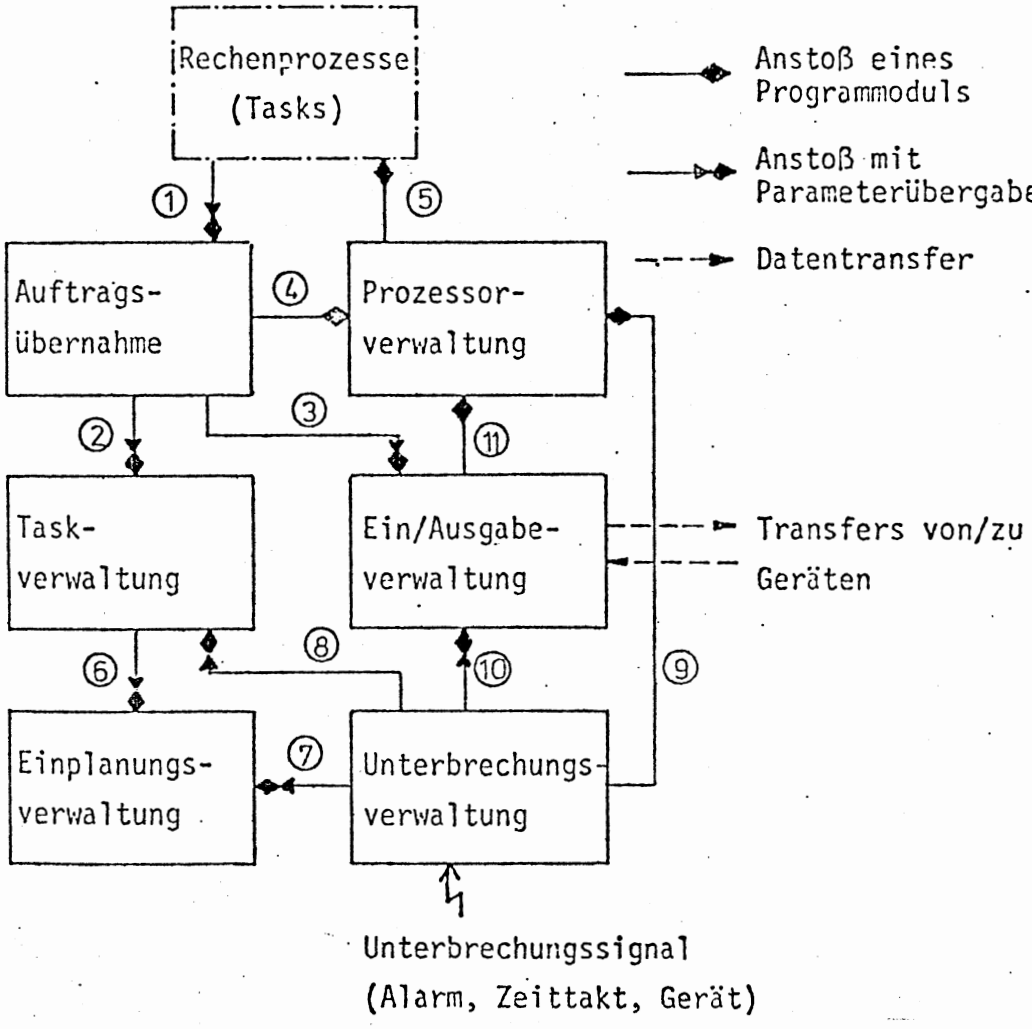


Bild 2.2: Aufbau des PEARL-Betriebssystems

Im folgenden sollen die einzelnen Funktionseinheiten näher beschrieben werden. Die Nummern an den Kanten der 'Aufrufpfeile' geben die Ursache für den Aufruf eines Moduls an. Auf die Bedeutung dieser Nummern wird bei der Beschreibung der einzelnen Programmbausteine näher eingegangen.

Auftragsübernahme (Schnittstelle zur Anwendertask)

Dieser Verwaltungsblock wird aufgerufen, wenn das PEARL-Programm Betriebssystemfunktionen benötigt (Aufrufpfeil 1). Als Parameter wird die Adresse des Auftragsparameterblocks übergeben. Aus den dort vorliegenden

Informationen kann das PBS die Art des Auftrags entnehmen und die entsprechende Routine aufrufen. Dies kann entweder die Taskverwaltung (Aufrufpfeil 2) oder die Ein-/Ausgabeverwaltung (Aufrufpfeil 3) sein. Danach wird die Prozessorverwaltung beauftragt, den Prozessor wieder an die höchstpriore, lauffähige Task zu geben (Aufrufpfeil 4).

Taskverwaltung

Die Taskverwaltung führt Buch über die Zustände der Tasks. In diesem Baustein werden auch die Synchronisationsaufrufe behandelt. Die Taskverwaltung bewirkt entweder unmittelbar vorgesehene Zustandsänderungen oder veranlaßt die Einplanung derselben über einen Aufruf der Einplanungsverwaltung (Aufrufpfeil 6).

Einplanungsverwaltung:

Ist die Zustandsänderung für eine Task an eine Bedingung geknüpft, (Auftreten eines bestimmten Interrupts) so wird diese Zustandsänderung zurückgestellt und der Auftrag in die entsprechende Liste eingereiht.

Unterbrechungsverwaltung:

Die Unterbrechungsverwaltung stellt beim Eintreffen eines Alarmsignals bzw. Zeittaktes fest, ob für das betreffende Ereignis Einplanungen vorliegen und ruft in diesem Fall die Einplanungsverwaltung auf (Aufrufpfeil 7). Nach der Bearbeitung aller möglichen Einplanungen wird die Prozeßverwaltung angestoßen (Aufrufpfeil 9).

Die Unterbrechungsverwaltung erhält auch Fertigmeldungen von E/A-Geräten. In diesem Fall wird die E/A-Verwaltung aufgerufen (Aufrufpfeil 10).

Prozessorverwaltung:

Dieser Baustein gibt den Prozessor an die ablauffähige Anwendertask, die an erster Stelle in der Prozessor-Warteschlange steht (Aufrufpfeil 5).

E/A-Verwaltung

Die E/A-Verwaltung startet den E/A-Auftrag bzw. reiht ihn in die entsprechende Geräte-Warteschlange ein. Wird sie nach der Beendigung eines E/A-Auftrags von der Unterbrechungsverwaltung aufgerufen, wird geprüft, ob weitere Aufträge für das entsprechende Gerät vorliegen. In diesem

Fall erfolgt ein erneuter Transfer-Anstoß. Die E/A-Verwaltung ruft nach der Ausführung ihrer "Aufträge" die Prozeßorverwaltung auf (Aufrufpfeil 11).

Nach dem Start des PBS durchläuft dieses eine Initialisierungsroutine (in Bild 2.2 nicht eingezeichnet). Diese anlagenunabhängige Initialisierung besetzt die verschiedenen Kontrollblöcke (z.B. Taskkontrollblöcke) vor. Die Kontrollblöcke bilden die Schnittstelle zwischen PEARL-Programm und PBS. Näheres ist dem folgenden Kapitel zu entnehmen.

2.3.3. Schnittstellen des PBS nach 'außen'

2.3.3.1. Schnittstelle zum PEARL-Programm

Die Kommunikation von Anwenderebene und PBS erfolgt über Parameterblöcke (Kontrollblöcke). Die verschiedenen Typen dieser Kontrollblöcke werden im folgenden kurz erläutert:

Taskkontrollblock:
Jeder Task ist ein solcher Taskkontrollblock zugeordnet. Er enthält Informationen über den aktuellen Zustand der Task.

Gerätekontrollblock:
Jedem E/A-Gerät, das vom PEARL-Programm angesprochen (im Systemteil vereinbart) wird, ist ein Gerätekontrollblock zugeordnet. Er enthält Daten über den Zustand des betreffenden E/A-Gerätes und bildet zugleich den Warteschlangenkopf für die E/A-Aufträge.

Semaphorkontrollblock:
Jeder Sempahorvariablen ist ein Semaphorkontrollblock zugeordnet. Er enthält die Informationen über den Zustand (Stand) der Semaphorvariablen und bildet den Kopf der Warteschlange, in der die Taskkontrollblöcke eingekettet werden, die durch diese Semaphore blockiert sind. Diese Kontrollblöcke müssen vom Übersetzungssystem angelegt werden und enthalten außerdem Informationen über den Anfangszustand der Task, des Geräts oder der Semaphorvariablen um einen Wiederstart des Systems zu ermöglichen. Nach dem Start des PBS initialisiert dieses noch einige Komponenten der einzelnen Kontrollblöcke, daher muß dem PBS die Länge der Kontrollblöcke bekannt sein und sie müssen in der folgenden Weise angeordnet sein: Alle Kontrollblöcke müssen im Speicher lückenlos hintereinander abgelegt sein, und zwar zuerst die Taskkontrollblöcke, dann die Semaphorkontrollblöcke und zum Schluß die Gerätekontrollblöcke.

Der Anfang der Taskkontrollblöcke (erste Zelle des ersten Taskkontrollblocks) muß mit der Marke FRSTCB markiert werden, der Anfang der Semaphorkontrollblöcke mit FRSTSM und der Anfang der Gerätekontrollblöcke mit FRSTIO. Die Zelle nach dem letzten Gerätekontrollblock muß mit LASTIO markiert werden.

Während des Laufs kann ein PEARL-Programm an das PBS drei verschiedene Arten von Aufträgen abgeben. Es kann ein Auftrag für die Einplanung einer Taskzustandsänderung, für die sofortige Ausführung einer Taskzustandsänderung und für die Durchführung einer Ein/Ausgabe gegeben werden. In jedem dieser Fälle muß dem PBS ein anderer Auftragsparameterblock (dieser enthält die genauen Angaben zu dem Auftrag) übergeben werden. Die Auftragsparameterblöcke können an beliebiger Stelle im Anwenderprogramm liegen. Sie müssen dem PBS auch nicht vor der Übergabe des Auftrags bekannt sein. Die Auftragsübergabe geschieht dadurch daß dem PBS die Adresse des entsprechenden Auftragsparameterblocks mitgeteilt wird.

Der Aufbau dieser Kontrollblöcke ist [ROES79] und [ROES78] zu entnehmen.
Für die S310 ist der Aufbau der Kontrollblöcke [TRAU78] zu entnehmen.

2.3.3.2. Schnittstelle zu einer konkreten Anlage

Für die Realisierung des PBS auf einer konkreten Maschine muß es noch durch maschinenabhängige Programmteile ergänzt werden. So ist z.B. das Vorgehen beim Sperren bzw. Freigeben von Interrupts von der entsprechenden Zielmaschine abhängig. Für solche Maschinenabhängigkeiten wurden sogenannte funktionelle Schnittstellen vorgesehen. Es gibt solche organisatorischen Befehle (andere Bezeichnung für funktionelle Schnittstelle) zum

- Retten der Register einer Task
- Register restaurieren und ausgewählte Task fortsetzen
- Interrupts sperren
- Interrupts freigeben
- Geräteversorgung
- Fehlerstop
- Warten auf Interrupt.

Im folgenden sollen diese Befehle näher beschrieben werden. Dabei wird auch die Syntax dieser Befehle, wie sie in der Definitionssprache des PBS (SPASS) festgelegt ist, mit angegeben.

Retten der Register einer Task:

Syntax: SAVEREGISTER oder SAVEALLREGISTER

Mit dieser Anweisung werden die Rechenregister der laufenden Task in den dieser Task zugeordneten Registerrettbereich kopiert. Dieser Rettbereich ist im Taskkontrollblock enthalten. Ebenso muß der aktuelle Befehlszählerstand in den Taskkontrollblock gerettet werden.

Register restaurieren und ausgewählte Task fortsetzen

Syntax: RESUMEPROCESS

Dieser Befehl kopiert den Inhalt des Registerrettbereichs der Task, die als nächste laufen soll, in die Rechenregister. Dann muß der Befehlszählerstand auf den Wert gebracht werden, an dem die ausgewählte Task beginnen soll. Der Befehlszählerstand ist im entsprechenden Taskkontrollblock enthalten.

Diese Änderung des Befehlszählerstandes bedeutet meist zugleich, daß die Anwendertask nun den Prozessor erhält.
RESUMEPROCESS beinhaltet auch ein Freigeben der Interrupts.

Interrupts sperren

Syntax: DISABLE

Mit dieser Anweisung werden ankommende Interrupts gesperrt.

Interrupts freigeben

Syntax: ENABLE

Mit ENABLE werden Interrupts wieder freigegeben.

Geräteversorgung

Syntax: DOIO

Mit diesem Befehl wird die Ein/Ausgabe angestoßen. Dabei wird vorausgesetzt, daß die entsprechenden Versorgungsparameter bereitgestellt sind.

Diese Parameter sind die Anfangs- und Endeadresse des E/A Puffers und eine Kennung, welches Gerät den E/A-Verkehr abwickelt.

Fehlerstop

Syntax: ERROR.zahl 'name'

Tritt im PBS eine nicht behebbare Fehlersituation auf, wird dieser Befehl ausgeführt. Die im Befehl angegebene Zahl und der Name dienen zur Identifizierung des Fehlers.

Die Folgen dieses Befehls sind implementationsabhängig.

Warten auf Interrupt

Syntax: HALT

Ist keine Anwendertask lauffähig, wird dieser Befehl ausgeführt. Damit wird das PBS angehalten. Aus dieser Situation kann es nur durch eintreffende erwartete Interrupts (Einplanungen) befreit werden.

Näheres über die Bedeutung dieser organisatorischen Befehle ist [ROES79] zu entnehmen.

3. Anlagenabhängiges PBS

3.1. Allgemeines

Das im vorigen Kapitel beschriebene PBS wurde im Rahmen dieser Arbeit auf dem Prozeßrechner Siemens 310 implementiert. Das PBS läuft auf der Siemens 310 als Benutzerprogramm im Sinne des Betriebssystems ORG 310. (Liste der benötigten ORG Bausteine s. Anhang VII).

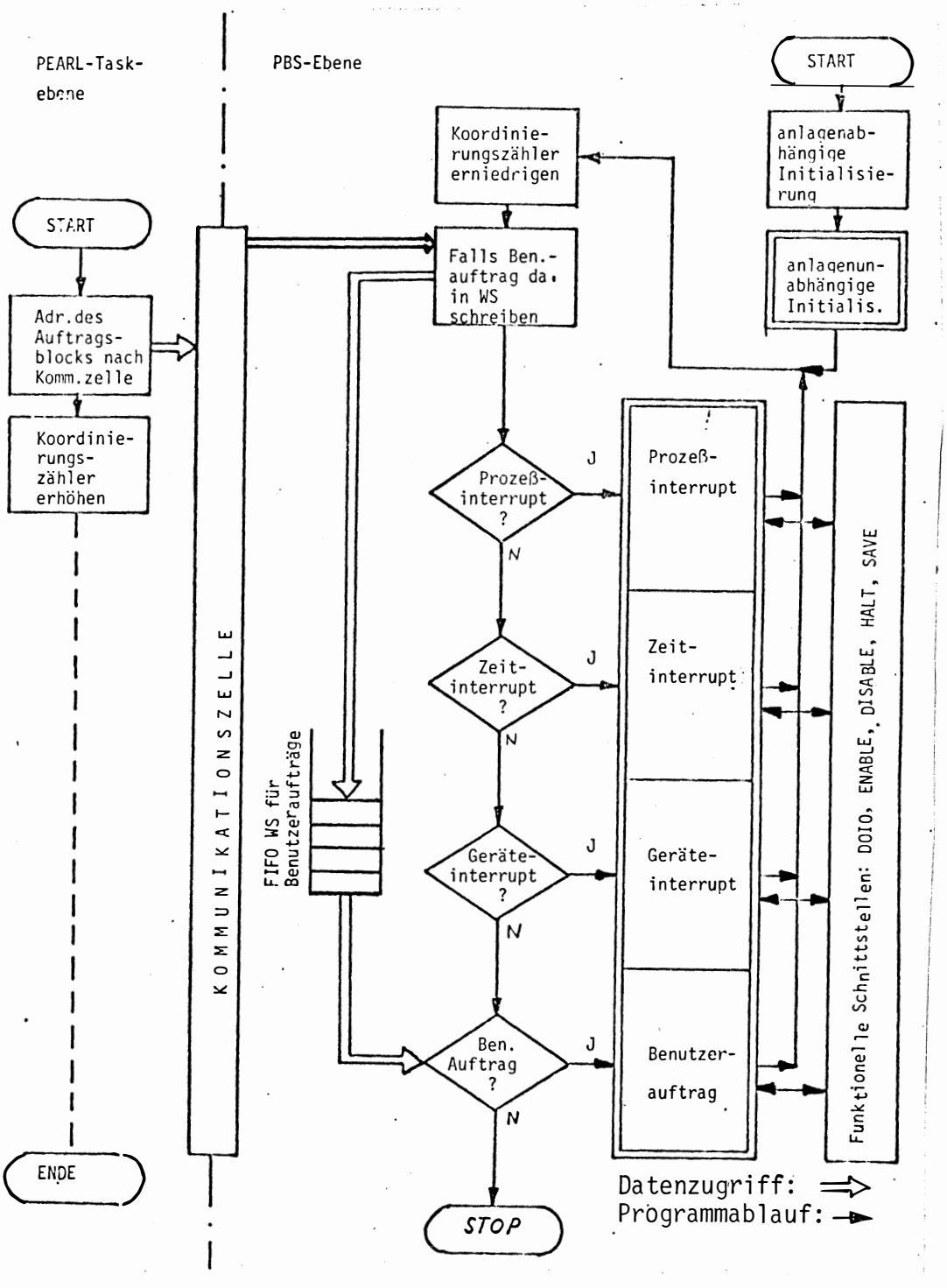


Bild 3.1
Das PBS auf der S310

Das PBS und das Anwenderprogramm laufen auf jeweils einer Programmebene der S310. Dies hat den Vorteil, daß das PBS und die Anwendertasks nicht zusammengebunden werden müssen, außerdem ist dadurch eine scharfe Trennung von Anwendertasks und PEARL-Betriebssystem möglich. Dabei ist jedoch zu beachten, daß das PBS auf eine höher priore Programmebene als die Anwendertasks geladen werden muß, damit das PBS nicht von den Anwendertasks unterbrochen werden kann. Das PEARL-System erwartet die Anwendertasks auf einer fest vorgegebenen Programmebene (z.Z. Ebene 13, die Nummer der Programmebene wird in einer 'Ident'-Anweisung in der anlagenabhängigen Initialisierung des PBS festgelegt, siehe Anhang IV), weil das PBS bei der Ausführung der Befehle SAVEREGISTER bzw. RESUMEPROCESS wissen muß, welche Registertafel restauriert bzw. gerettet werden soll (siehe auch Beschreibung von SAVEREGISTER und RESUMEPROCESS in Kapitel 2.3.3.2). Im folgenden soll die Programmebene für die Anwendertasks mit 'PEARL-Taskebene' und die für das PBS mit 'PBS-Ebene' bezeichnet werden. Die Programmebene für das PBS kann beliebig gewählt werden (höher prior als PEARL-Taskebene). Die Ladeadressen für das PBS und für die Anwendertasks sind beliebig. Zu beachten ist noch, daß eine fest vorgegebene Zelle benutzt wird, um Daten zwischen PBS und Anwendertasks austauschen zu können. Diese Zelle wird sowohl vom PEARL-System als auch von den Anwendertasks durch absolute Adressierung angesprochen, sie wird im folgenden 'Kommunikationszelle' genannt. Die Synchronisation von PBS und Anwendertasks wird mit Hilfe von Prioritäten und einem Koordinierungszähler durchgeführt. Dieser Synchronisierungsmechanismus wird noch näher erläutert.

Der anlagenabhängige Abschnitt des PBS gliedert sich in 3 Teile

- anlagenabhängige Initialisierung
- Aufrufvorverarbeitung
- Implementierung der funktionellen Schnittstellen.

Nach dem Laden des PBS und der Anwendertasks muß das PBS gestartet werden. Das PBS startet dann die PEARL-Taskebene. Durch die höhere Priorität bleibt die PBS-Ebene laufend. Werden in einem PEARL-Programm Zeiteinplanungen verwendet, benötigt das PBS eine weiter Programmebene (höher prior als das PBS z.Z. Ebene 11) um die Zeitinterrupts in ein Erhöhen des Koordinierungszählers umzuwandeln. (siehe Kapitel 3.2)

3.2. Anlagenabhängige Initialisierung

Nach dem Start des PBS (z.Z. Programmebene 12) wird zunächst die anlagenabhängige Initialisierung durchlaufen. Während der anlagenabhängigen Initialisierung wird ein Koordinierungszähler (KOOR) eingerichtet, der zur Synchronisiation der PBS- und der PEARL-Ebene verwendet wird. Dieser Koordinierungszähler wird mit dem Anfangswert Null vorbesetzt.

Danach startet das PBS die PEARL-Anwenderebene (z.Z. Programmebene 13), das PBS bleibt aber laufend, da es höhere Priorität hat. Durch ein 'Koordinierungszähler erniedrigen' blockiert sich das PBS, die Anwenderebene wird nun laufend und bereitet die Initialisierungsdaten für das PBS vor. Dazu baut die PEARL-Taskebene einen Datenblock (Initialisierungsdatum) mit folgender Struktur auf:

1. Wort: Adresse des 1. Taskkontrollblocks
2. Wort: Adresse des 1. Semaphorkontrollblocks
3. Wort: Adresse des 1. Gerätekontrollblocks
4. Wort: Endeadresse der Kontrollblöcke

Die Anfangsadresse dieses Datenblocks wird in die Kommunikationszelle geschrieben und danach der Koordinierungszähler erhöht. Dadurch wird die PBS-Ebene wieder laufend (höhere Priorität). Mit den erhaltenenen Informationen ergänzt das PBS die Vorbesetzung der Geräteblocke. Danach blockiert das PBS für die weitere anlagenabhänge Initialisierung die PEARL-Taskebene. Dies ist notwendig, um zu vermeiden, daß die PEARL-Taskebene weiterläuft, während das PBS kurze Zeit blockiert ist (Eingaben vom Benutzer z.B. 'Zeit einschalten siehe unten).

Zum Blockieren der PEARL-Taskebene wird der gleiche Mechanismus wie bei der Realisierung des 'HALT'Befehls (siehe Kapitel 3.4 dabei wird ein Koordinierungszähler 'SPERRE' verwendet) benutzt, mit dem Unterschied, daß der Befehlszählerstand der PEARL-Taskebene (Register R1) gerettet wird, um beim Freigeben der PEARL-Taskebene den Ausgangszustand wieder herstellen zu können (siehe unten).

Prozeßinterrupts werden vom ORG 310 entgegengenommen und in Erhöhen des Koordinierungszähler umgewandelt (siehe Kapitel 3.3.2). Diese Umwandlung wird durch den Aufruf des Makros 'Alarm anmelden' eingeschaltet (siehe ORG 310 und Kapitel 3.3.2).

Der Timer braucht nur eingeschaltet zu werden, falls im Benutzerprogramm Zeitschedules enthalten sind. Bei ausgeschaltetem Timer benötigt das PEARL-System

statt drei nur zwei Programmebenen, außerdem wird der Verwaltungsaufwand verringert. Der Timer belastet das PBS alle 0,1 Sec mit einem Zeittakt.

Die anlagenabhängige Initialisierung fragt den Benutzer über den Bedienbildschirm, ob der Timer eingeschaltet werden soll.

Wünscht dies der Benutzer, wird die Zeitgeberprozedur einer höher prioren Ebene als (Funktion siehe Kapitel 3.3.3) die PBS Ebene zugeordnet (Makroaufruf: 'Programm anmelden') und ein zyklisches Aktivieren der Zeitebene veranlaßt (alle 0,1 Sec).

Nach der Initialisierung für den Timer wird die PEARL-Task-Ebene freigegeben. Dazu wird von der PBS-Ebene der ursprüngliche Befehlszählerstand wieder restauriert (Eintragen in das Register R1 der PEARL-Taskebene) und der Koordinierungszähler 'SPERRE' erhöht. Dadurch wird die PEARL-Taskebene wieder lauffähig. Die anlagenabhängige Initialisierung ruft nun die anlagenunabhängige Initialisierung auf. Diese macht diejenige Task lauffähig, deren Taskkontrollblock in der Liste der TKB's an erster Stelle steht.

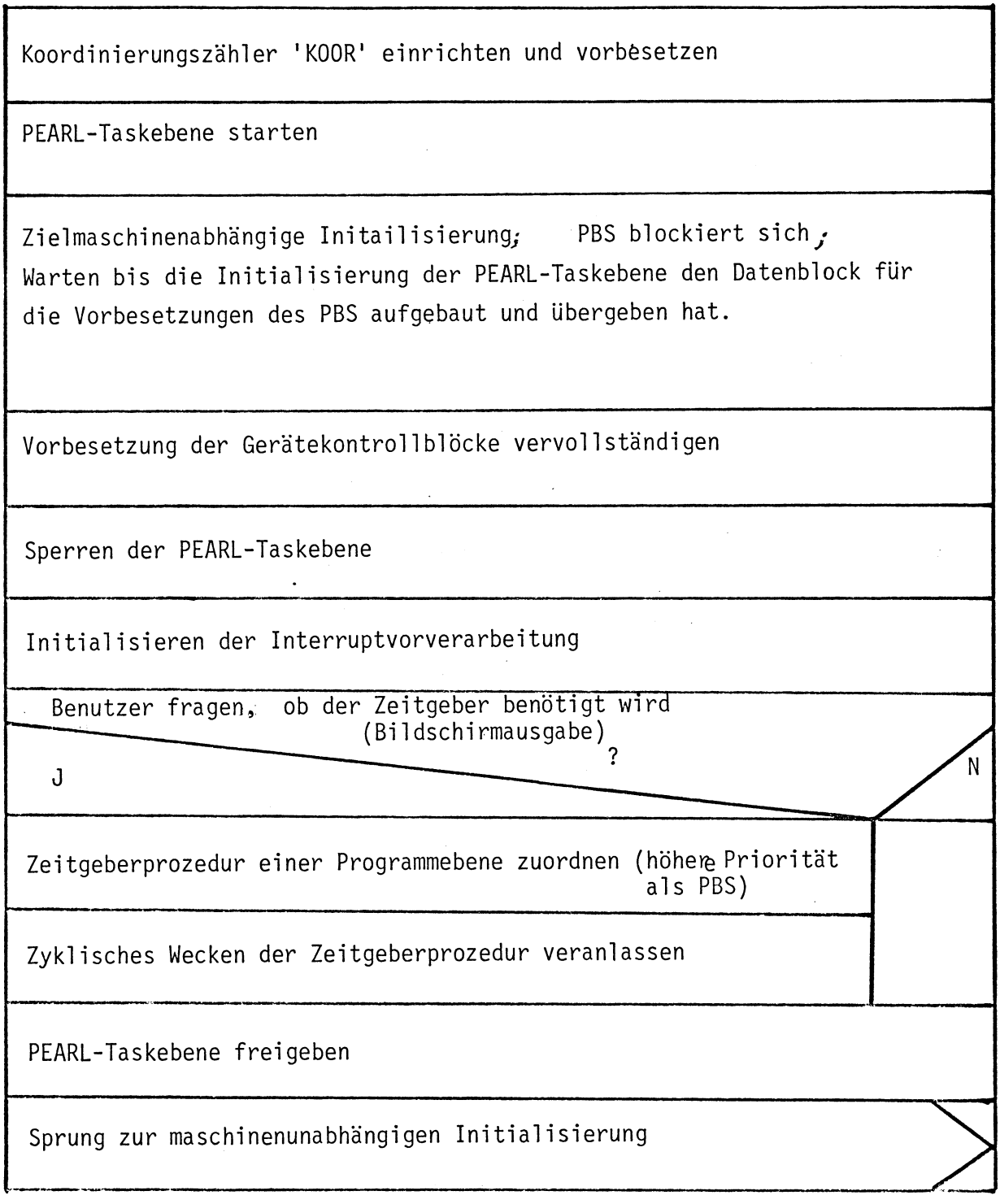


Bild 3.2
Anlagenabhängige Initialisierung

3.3 Anlagenabhängige Auftragsvorverarbeitung

3.3.1 Allgemeines

Werden an das PBS Anforderungen gestellt, muß dies durch ein Erhöhen des Koordinierungszählers angezeigt werden. Ein Erhöhen des Koordinierungszählers kann folgende Ursachen haben.

- Auftrag von der Benutzerebene
- Prozeßinterrupt
- Zeitinterrupt
- Geräteinterrupt

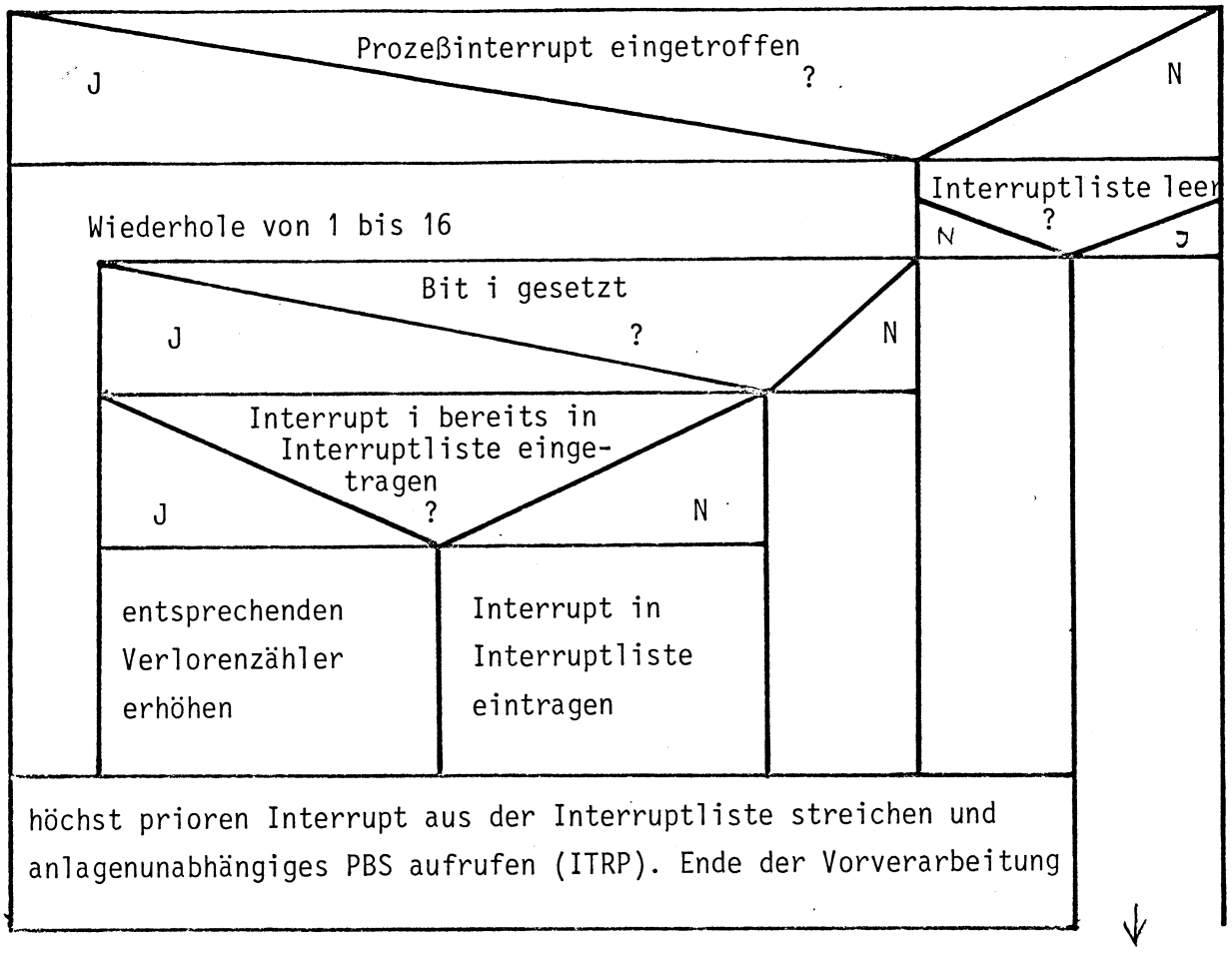
Durch das Erhöhen des Koordinierungszählers wird die PBS-Ebene laufend (im Sinne des ORG 310). Die PEARL-Taskebene wird blockiert. (siehe Kapitel 3.2). Die Auftragsvorverarbeitung überprüft zuerst, ob in der Kommunikationszelle die Adresse eines Auftragsparameterblocks steht (Inhalt der Kommunikationszelle ungleic Null). Ist die Kommunikationszelle ungleich Null, wird deren Inhalt in die Benutzerauftragsliste übernommen. Die Begündung für das Retten des Inhalts der Kommunikationszelle wird in Kapitel 3.3.5 gegeben. Die Auftragsvorverarbeitung stellt nun die Ursache für das Erhöhen des Koordinierungszählers fest, da zusätzlich zu einem Benutzerauftrag noch Interrupts eingetroffen sein können, die zuerst bearbeitet werden müssen.

3.3.2 Prozeßinterrupt

An der S 310 sind 16 Interrupteingänge in Form eines Digitaleingaberegisters (16 Bit) angeschlossen (PE3600).
Kippt mindestens ein Bit dieses Registers aufgrund des angeschlossenen Ereignisses von Null nach Eins (aufsteigende Flanke), so wird ein Sammelinterrupt (es können gleichzeitig mehrere Interrupts eingetroffen sein) an das ORG gegeben, das diesen entgegennimmt und ein Erhöhen des Koordinierungszählers veranlaßt.

Im Parameterblock des Makros 'Alarm anmelden' wird durch das ORG beim Eintreffen eines Interrupts eine Kennung (Ereignisbit) gesetzt. Dadurch kann die Auftragsvorverarbeitung feststellen, daß mindestens ein Prozeßinterrupt eingetroffen ist.

Die Auftragsvorverarbeitung liest nun das Prozeßinterruptregister (das dadurch gelöscht wird und nun wieder Interrupts entgegennehmen kann), stellt aufgrund der gesetzten Bits fest, welche Prozeßinterrupts eingetroffen sind und merkt sich diese in der Prozeß-Interruptliste.



Prüfen ob das Erhöhen des Koordinierungszählers von einem Zeittakt verursacht wurde
(siehe Kapitel 3.3.3)

Bild 3.2
Bearbeitung eines Prozeßinterrupts in der anlagenabhängigen Vorverarbeitung

Treffen bis zum Auslesen des Prozeßinterruptregisters weitere Interrupts ein, so werden diese mit dem bisherigen Registerinhalt verodert, ohne daß ein weiterer Interrupt an das ORG gesendet wird, d.h. der Koordinierungszähler wird nur beim ersten Interrupt erhöht. Trifft der gleiche Interrupt bis zum Auslesen des Prozeßinterruptregisters mehrmals ein, so gehen diese Interrupts verloren, ohne daß dies festgestellt werden kann. Traf ein Prozeßinterrupt ein, der bereits in der Prozeßinterruptliste als zur Bearbeitung anstehend vermerkt ist, wird der diesem Interrupt zugeordnete Verlorenzähler erhöht. (Interruptrate für PBS zu hoch). Der Koordinierungszähler wird entsprechend der eingetroffenen Prozeßinterrupts erhöht (Anzahl der eingetroffenen Interrupts die nicht bereits in der Interruptliste vermerkt waren minus eins).

Den Prozeßinterrupts sind Prioritäten von eins (höchste Priorität) bis sechzehn (niedrigste Priorität) zugeordnet. Der Interrupt mit der höchsten Priorität wird aus der Interruptliste gestrichen und bearbeitet. (Aufruf des anlagenunabhängigen PBS). Ist kein Prozeßinterrupt eingetroffen (Ereignisbit nicht gesetzt), wird in der Interruptliste nachgesehen, ob dort noch weitere Interrupts vermerkt sind. Wenn ja, wird der höchstpriore aus der Interruptliste gelöscht und bearbeitet. Traf kein Prozeßinterrupt ein und sind auch keine in der Prozeßinterruptliste vermerkt, wird geprüft, ob ein Zeitinterrupt eingetroffen ist (siehe folgendes Kapitel). Die folgende Skizze faßt die Bearbeitung der Prozeßinterrupts in der anlagenabhängigen Vorverarbeitung zusammen.

3.3.3. Zeitinterrupt

Falls der Zeittakt eingeschaltet ist (siehe Kapitel 3.2) wird alle 0.1 Sekunden die Ebene 11 aktiviert. Dadurch daß diese höhere Priorität als PBS- und Taskebene hat, wird sie auch laufend. Das Programm auf der Ebene 11 erhöht den Koordinierungszähler (KOOZ), erhöht eine Variable ('TAKTE') um 1 und beendet sich dann (siehe Bild 3.4). Durch das Erhöhen des KOOZ wird nun die PBS-Ebene laufend, die nun aufgrund der Variablen 'TAKTE' feststellt, daß ein Zeitinterrupt eingetroffen ist. Die Variable 'TAKTE' verhindert, daß Zeitinterrupts verlorengehen. Dadurch daß in der Vorverarbeitung zuerst untersucht wird, ob der Koordinierungszähler durch einen Prozeßinterrupt erhöht wurde, kann bei einer hohen Interruptrate mehrmals ein Zeitinterrupt eintreffen, ohne daß er abgearbeitet wird (d.h. Zeiteinplanungen können bei einer hohen Prozeßinterruptrate um einige Zehntel Sekunden zu spät bearbeitet werden). Das PBS ruft nach dem Erniedrigen von 'TAKTE' um eins, die entsprechende rechnerunabhängige PBS-Prozedur auf.Dies wird solange wiederholt (nur unterbrochen von der Bearbeitung eventuell eintreffender Prozeßinterrupts) bis die Variable 'TAKTE' den Wert Null hat (und damit auch der Koordinierungszähler 'KOOZ' entsprechend der Anzahl der vergangenen Zeittakte erniedrigt wurde).

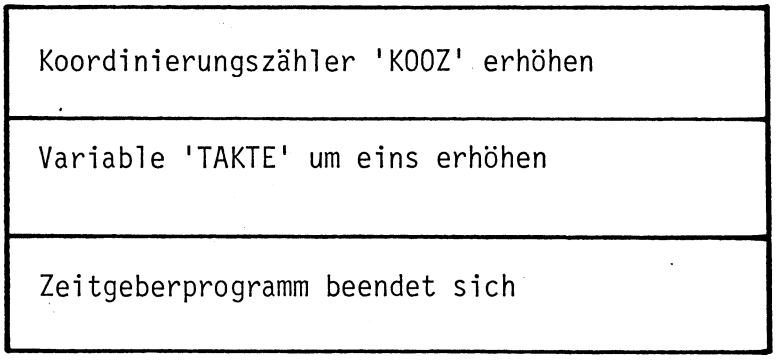
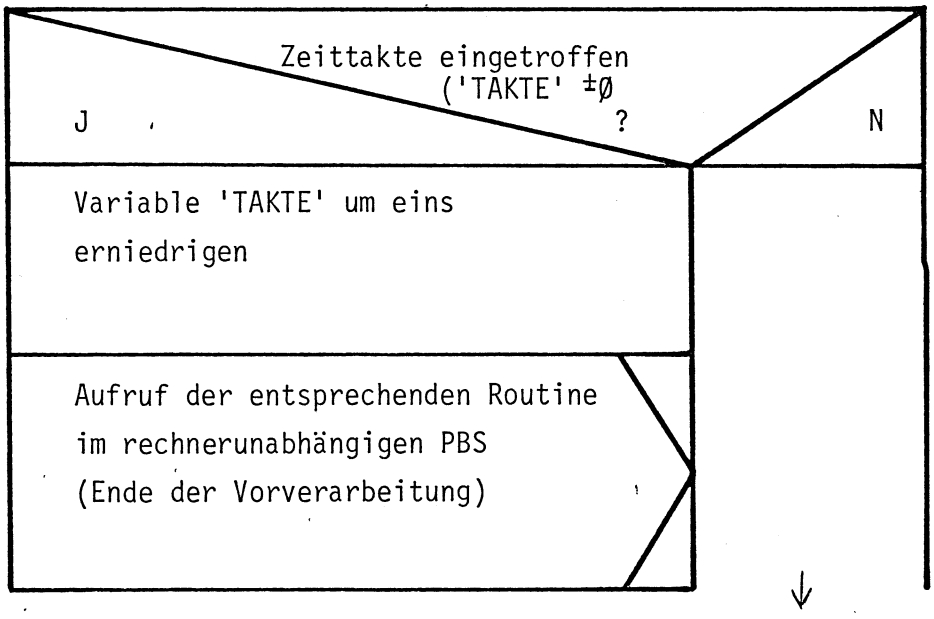


Bild 3.3
Zeitgeberprogramm



Prüfen, ob das Erhöhen
des Koordinierungszählers
von einem Geräteinterrupt
verursacht wurde (siehe
Kapitel 3.3.4)

Bild 3.4
Bearbeitung eines Zeitinterrupts in der anlagenabhängigen Vorverarbeitung

3.3.4 Geräteinterrupt

Liegt weder ein Zeit- noch ein Prozeßinterrupt vor, wird geprüft, ob ein Gerät einen Interrupt gesandt hat, d.h. ob ein Gerät meldet, daß es mit dem entsprechenden E/A-Auftrag fertig ist (Interrupt wird in ein Erhöhen des Koordinierungszählers umgewandelt, siehe Kapitel 3.2). Die Vorverarbeitung stellt fest, welches Gerät den Interrupt gesandt hat (Gerätezustand gleich eins, Tätigkeitsbit im Makro-Parameterblock gleich Null) und ruft dann mit den entsprechenden Parametern die anlagenunabhängige Geräteinterruptverwaltung auf.

3.3.5 Auftrag von Benutzer

In Kapitel 3.3.1 wurde geschildert, daß die Auftragsvorverarbeitung nach jedem Erhöhen des Koordinierungszählers den Inhalt der Kommunikationszelle in die Auftragsliste kopiert (falls der Inhalt der Kommunikationszelle ungleich Null ist).

Warum das Retten des Inhalts der Kommunikationszelle nötig ist, soll in diesem Kapitel begründet werden. Will eine Task an das PBS einen Auftrag geben und wird dabei (zwischen dem Eintragen der Adresse des Auftragsparameterblocks in die Kommunikationszelle und dem Erhöhen des Koordinierungszählers KOOZ) durch einen Interrupt unterbrochen, so kann dieser Auftrag (Inhalt der Kommunikationszelle) verlorengehen.

Aufgrund eines Interrupts kann durch das PBS eine andere, höher priore Task laufend werden. Wenn nun diese Task auch einen Auftrag an das PBS gibt, wird der Inhalt der Kommunikationszelle überschrieben und vom PBS in der Auftragsvorverarbeitung auf Null gesetzt.

Wird zu einem späteren Zeitpunkt die niederpriore Task wieder laufend, so wird als erster Befehl das Erhöhen des Koordinierungszählers aufgeführt. Das PBS kann jetzt die Ursache für das Erhöhen des Koordinierungszählers nicht mehr feststellen (kein Interrupt und Inhalt der Kommunikationszelle gleich Null).

Dadurch daß der Inhalt der Kommunikationszelle aber nach jedem Erhöhen des Koordinierungszählers (unabhängig von der Ursache) gerettet wird, kann auf die oben geschilderte Weise kein Auftrag mehr verlorengehen. Die Liste für das Retten der Benutzeraufträge ist 20 Elemente lang, d.h. es müssen mindestens 20 Tasks vorhanden sein, damit ein Listenüberlauf möglich wird (20 verschiedene Tasks müßten jeweils zwischen dem Beschreiben der Kommunikationszelle und dem KOOZ-Erhöhen unterbrochen werden).

Will eine Task, daß ein Auftrag eingeplant wird (Schedule) so muß die Auftragsvorverarbeitung den Scheduleparameterblock in den Taskkontrollblock, mit Ausnahme der Komponenten SPCB (Schedule Kette) und SCHART (Zeiger auf Zeiger auf Schedule), kopieren. Dies ist notwendig, da die PEARL-Taskebene den Speicherplatz für den Scheduleparameterblock unter Umständen für weitere Aufträge benutzt.

Da das anlagenunabhängige PBS pro Task nur eine Einplanung (die zuletzt veranlaßte) zuläßt, bedeutet dies keine Einschränkung seiner Möglichkeiten.

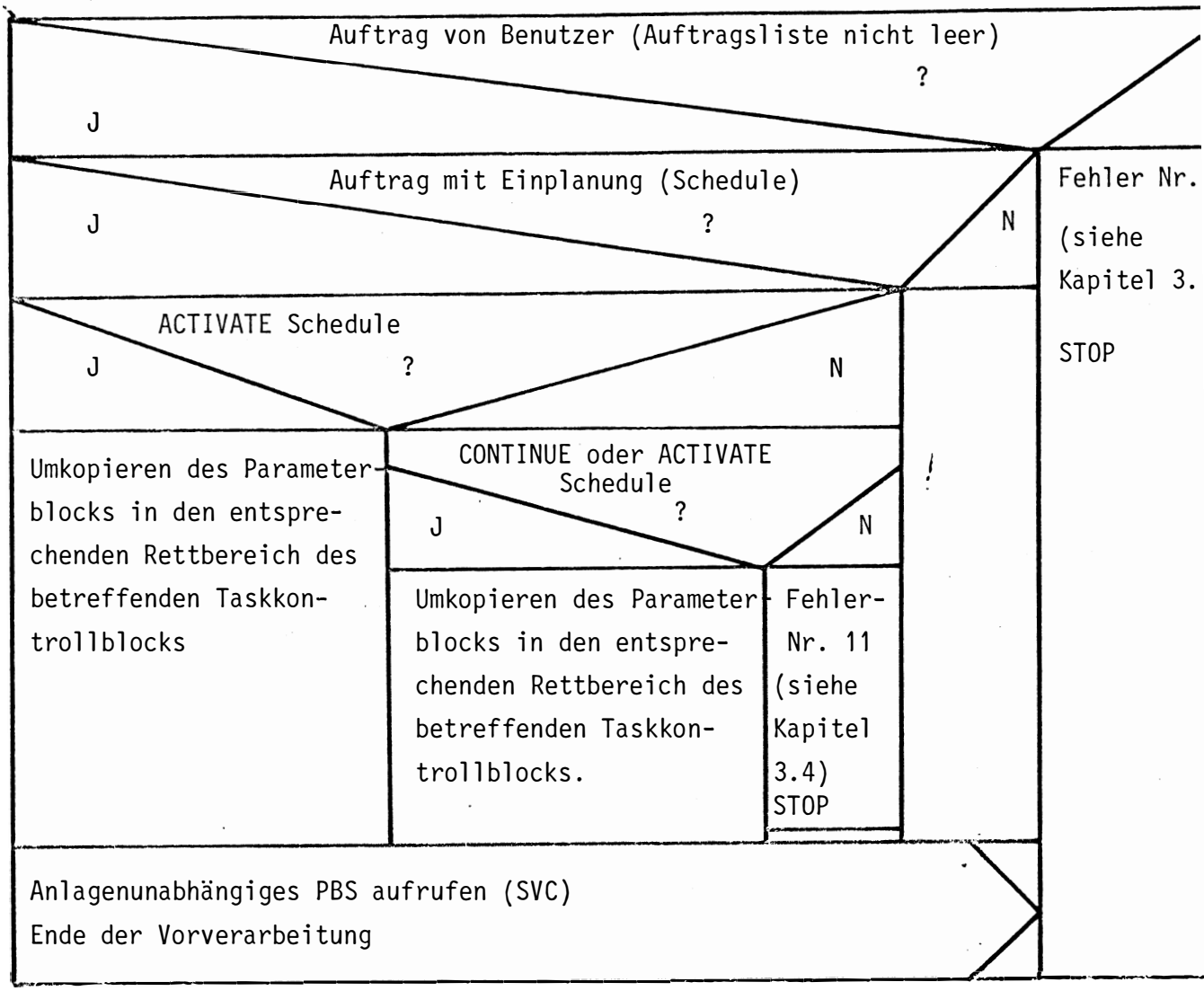


Bild 3.5
Vorverarbeitung eines Benutzerauftrags

3.4. Fehler

Treten im rechnerabhängigen Teil des PBS Fehler auf, so wird in das Register R7 eine Fehlerkennung geschrieben und der Rechner gestoppt. Nach einem Restart kann mit dem Testhilfeprogramm der Inhalt von R7 angesehen werden.
Einen Überblick über die auftretenden Fehler und ihre Kennziffer (Inhalt von R7) gibt die folgende Tabelle:

Fehler-Nr.	Fehler
10:	Fehler bei der Initialisierung der Interruptvorverarbeitung.
11:	Kennziffer der Schedules unzulässig (weder RESUME, ACTIVATE oder CONTINUE)
12:	Die Ursache für das Erhöhen des Koordinierungszählers konnte nicht erkannt werden (vermutlich ORG- oder Hardwarefehler).

3.5. Zeitverhalten des PBS an der S 310

3.5.1. Interruptrate

Die Interruptrate an dem PEARL-System für die Siemens 310 beträgt 100 Hz (100 Interrupts pro Sekunde). Bei dieser Interruptrate ist gewährleistet, daß die Interrupts nicht nur vom PBS und dem ORG noch erkannt werden, sondern auch noch Zeit für Benutzeraktivitäten bleibt

Natürlich verringert sich die Interruptrate entsprechend, wenn vom Benutzer umfangreiche Interruptbehandlungen vorgesehen sind.

Die Interruptrate von 100 Hz wurde mit dem beiliegenden Programm gemessen, wobei die Interruptreaktion das Erhöhen eines Zählers ist. Das Testprogramm besteht aus 2 Tasks. Die Task 'MASTER' ist eine bedienbare Steuertask. Die Task 'COUNT' zählt je nach der eingegebenen Meßzeit die Interrupts, die in der angegebenen Zeit eintreffen. Aus der Interruptfrequenz, die durch einen Pulsgenerator erzeugt wurde und der Meßzeit ergibt sich der Sollstand des Interruptzählers. Durch den Vergleich des Ist- und des Sollstandes des Interruptzählers wurde der Grenzfall ermittelt, ab welchem keine sichere Interruptreaktion (Zähler hochzählen) mehr möglich ist. Das Bild 3.7 zeigt das Verhalten des Systems bei verschiedenen Interruptraten. Auf der X-Achse ist die Interruptrate aufgetragen und auf Y-Achse das Verhältnis von Zähleristand zum Zählersollstand (Frequenz multipliziert mit der Meßzeit).

Das PBS benötigt für das Retten und das Restaurieren der Registertafel der PEARL-Taskebene ca. 1 ms (geschätzt). Von der Verwaltungszeit für die Interruptreaktion des PBS (t_2) werden ca. 0,9 ms (geschätzt) benötigt, um festzustellen welche Interrupts eingetroffen sind.

```
1    MODULE TAKT;
2    SYSTEM;
3         BSEA←→BILDSCHIRM:;
4         ITRP(1)←-TAKT:;
5         ITRP(15)←-START:;
6    PROBLEM;
7    DCL BILDSCHIRM VAL DEVICE GLOBAL;
8    DCL START VAL INTERRUPT GLOBAL;
9    DCL TAKT VAL INTERRUPT GLOBAL;
10   DCL WART SEMA GLOBAL INITIAL (0);
11   DCL  STOP FIXED GLOBAL;
12
13   /* STARTTASK:EINGEBEN DER MESSZEIT */
14   MASTER: TASK GLOBAL RESIDENT PRIORITY 10
15        DCL MESS FIXED;
16        REPEAT;
17        PUT BILDSCHIRM EDIT ('GIB MESSZEIT F3')(A(15));
18        GET BILDSCHIRM EDIT (MESS)(F(3));
19        PUT BILDSCHIRM EDIT ('START DRUECKEN')(A(14));
20        STOP =0;
21        WHEN START ACTIVATE COUNT;
22        REQUEST WART;
23        AFTER MESS*1 SEC RESUME;
24        STOP =1;
25        CONTINUE COUNT;
26        REQUEST WART;
27        END;
28   END; /* ENDE DER TASAK */
29
30
31   /* ZAEHLTASK*/
32   COUNT: TASK GLOBAL RESIDENT PRIORITY 11
33        DCL ZAEHLER FIXED;
34        RELEASE WART;
35        ZAEHLER =0;
36        /* EINPLANUNG DES INTERRUPTS */
37   LOOP:WHEN  TAKT  RESUME;
38        /* INTERRUPTREAKTION ANFANG */
39        /* ZAEHLEN DER EINGETROFFENEN INTERRUPTS */
40        ZAEHLER =ZAEHLER+1;
41        IF ZAEHLER GT 10000 THEN GOTO AUS;FI;
42        IF STOP EQ 0 THEN GOTO LOOP;FI;
43        /* INTERRUPTREAKTION ENDE */
44   AUS:  PUT BILDSCHIRM EDIT ('ZAEHLERSTAND:')(A(13));
45        PUT BILDSCHIRM EDIT (ZAEHLER)(F(5));
46        RELEASE WART;
47        TERMINATE;
48   END; /* ENDE VON COUNT */
49   MODEND;
```

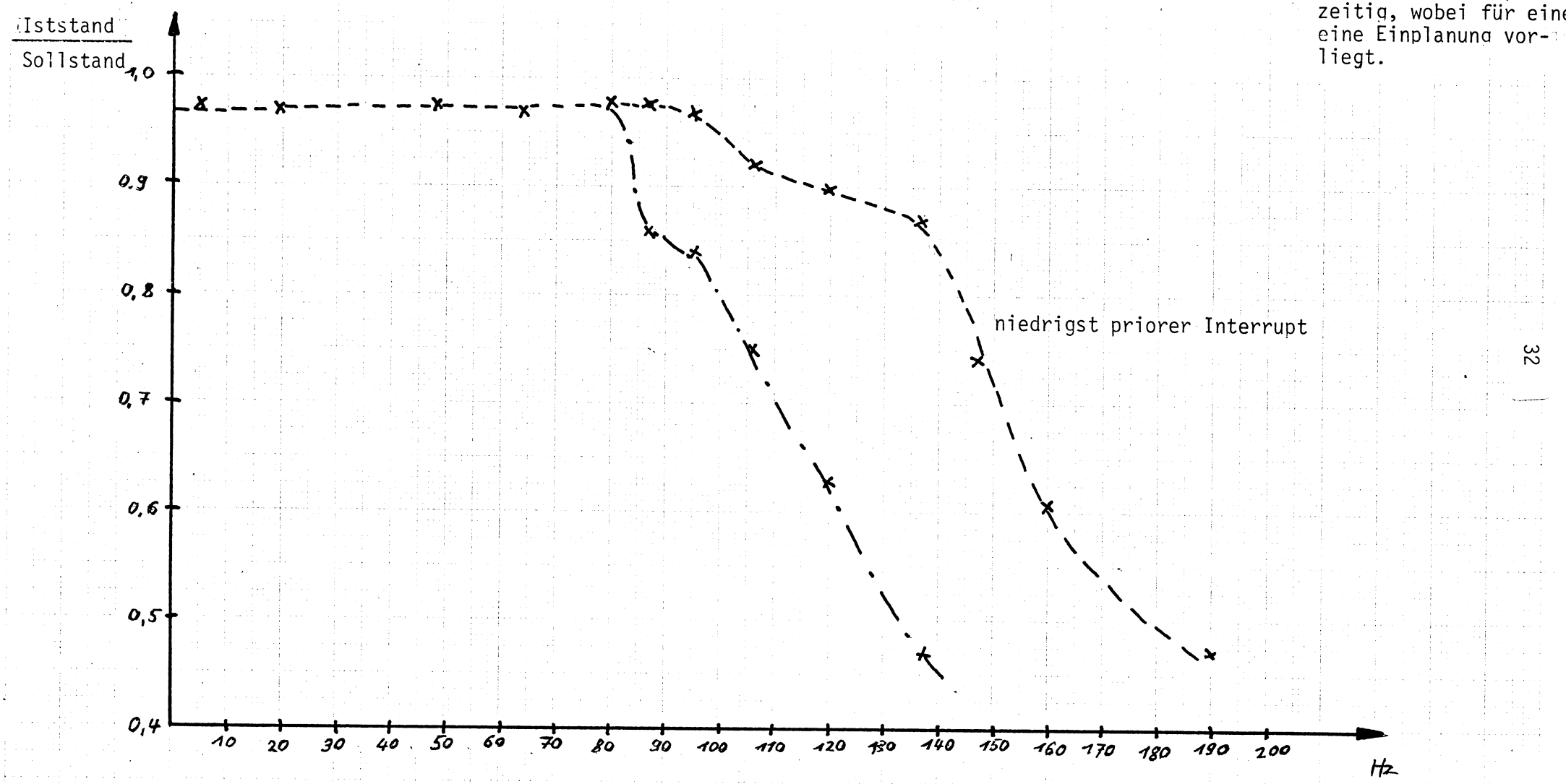


Bild 3.5: Interruptraten

3.5.2. Benutzeraufträge an das PBS

Die PBS Reaktion auf einen Benutzerauftrag beträgt zwischen 3-5 ms. Diese Zeiten sind abhängig von der Art des Auftrags. Eine Semaphoreoperation (RELEASE) dauert 3,2 ms (gemessener Wert). Einplanungen von Interruptreaktionen dauern wegen des erhöhten Verwaltungsaufwands entsprechend länger (geschätzt: 5 ms).
Diese Antwortzeiten teilen sich wie folgt auf:

1,2 ms Koordinierungszähler erhöhen

1,3 ms Koordinierungszähler erniedrigen

Rest: PBS Verwaltungsaufgaben

3.6. Implementierung der funktionellen Schnittstellen des rechnerunabhängigen PBS an dem Prozeßrechner S310.

Die Aufgaben der funktionellen Schnittstellen (organisatorische Befehle) wurden bereits in Kapitel 2.3.3.2 beschrieben. In diesem Kapitel soll die Realisierung dieser Schnittstellen für die Siemens 310 beschrieben werden (Programmtext siehe Anhang). Die Realisierung erfolgte mit Hilfe von Funktionen des Standardbetriebssystems der S310.

Funktionelle Schnittstellen ENABLE u. DISABLE

Diese beiden Organisationsbefehle können bei dieser Implentierung des PBS durch eine leere Anweisungsfolge dargestellt werden. Die Befehle sind durch die höhere Priorität des PBS und durch die Umwandlung der Interrupts in ein Erhöhen des Koordinierungszählers überflüssig.

Nach dem ORG 310 Aufruf 'Alarm anmelden' nimmt das ORG 310 alle Hardwareinterrupts entgegen. Die PBS Ebene wird bei einem Interrupt vom Standardbetriebssystem unterbrochen (höhere Priorität), das den Interrupt entgegennimmt, und in ein Erhöhen des Koordinierungszählers umwandelt. Interrupts gehen deshalb nicht verloren. Danach gibt das ORG 310 den Prozessor wieder an die PBS Ebene. Das PBS macht an der unterbrochenen Stelle weiter. Wird die PEARL-Taskebene von einem Interrupt unterbrochen, wird dieser vom ORG in ein Erhöhen des Koordinierungszählers KOOR umgewandelt, dadurch wird die PBS-Ebene wieder lauffähig und wird, nachdem das ORG den Prozessor abgegeben hat, auch laufend. Die PEARL-Taskebene bleibt unterbrochen und wird nach der Abarbeitung des Interrupts durch das PBS wieder laufend.

Funktionelle Schnittstelle HALT

Mit diesem Befehl soll die PEARL-Taskebene angehalten werden (im ORG 310 Sinn), wenn keine lauffähige Task vorhanden ist. Da das ORG 310 keinen entsprechenden Makro besitzt, würde nach der Selbstblockierung des PBS die Anwenderebene entsprechend dem Inhalt des Registers R1 seiner Registertafel weiterlaufen. Deshalb muß, wenn keine Task laufen darf, die Anwenderebene durch die PBS Verwaltung angehalten werden. Dies soll mit dem HALT Befehl geschehen (siehe Kapitel 2.3.3.2). Realisiert wird HALT mit Hilfe eines weiteren Koordinierungszählers (siehe Kapitel 3.2), der im folgenden Hilfskoordinierungszähler (HKOOR) genannt wird. In das Register R1 der Anwenderregistertafel (Programmebene 13) wird die Adresse des Befehls 'Hilfskoordinierungszähler erniedrigen' eingetragen. Blockiert sich das PBS, läuft die PEARL-Taskebene und führt diesen Befehl

(HKOOR erniedrigen) aus. Da der Hilfskoordinierungszähler bei der anlagenabhängigen Initialisierung mit Null vorbesetzt wurde, blockiert sich die PEARL-Taskebene. Somit ist die PEARL-Taskebene durch den Hilfskoordinierungszähler und die PBS-Ebene durch den Koordinierungszähler blockiert. Aus diesem Zustand kann das System (PEARL-Taskebene und PBS) nur durch einen Interrupt gelöst werden (z.B. Ende einer E/A), der den Koordinierungszähler des PBS erhöht und dieses lauffähig macht.

Funktionelle Schnittstellen SAVEREGISTERS u. SAVEALLREGISTERS

In der Implementierung dieser organisatorischen Befehle besteht kein Unterschied. In beiden Fällen werden alle Rechenregister der Zielmaschine gerettet. Bei dieser Implementierung des PBS an der S310 wird die Registertafel der Anwenderebene in den Registerrettbereich, der zuletzt laufenden Task, kopiert. Dieser Registerrettbereich ist ein Teil des Taskkontrollblocks.

Der Befehlszählerstand der Anwenderebene (entspricht dem Inhalt von R1 der entsprechenden Registertafel siehe Kapitel 2.2) wird noch extra in eine Zelle des Taskkontrollblocks (aktueller Befehlszählerstand der Task) übertragen. SAVEREGISTERS bzw. SAVEALLREGISTERS sind durch ein Unterprogramm implementiert. Die Schnittstellen werden auf einen entsprechenden Unterprogrammsprung abgebildet.

Funktionelle Schnittstelle: Resume Prozess

Durch diese Anweisung werden die Vorbereitungen für den Wechsel des Prozessors von der PBS- zur Anwenderebene durchgeführt.

Es werden die Register der Anwenderebene für die höchstpriore, lauffähige Task vorbereitet. Diese Task soll nach der Selbstblockierung des PBS laufend werden.

Der Inhalt des Registerrettbereichs wird in die Registertafel der Anwenderebene übertragen. In das Register R1 der Anwenderebene wird der aktuelle Befehlszählerstand, der dem entsprechenden Taskkontrollblock entnommen wird, eingetragen. Wurde das PBS durch einen Interrupt angestoßen und war keine Task laufend, wird der Hilfskoordinierungszähler noch erhöht und damit die PEARL-Taskebene wieder lauffähig. Am Ende von RESUME PROCESS wird in der Warteposition (Koordinierungszähler erniedrigen) gesprungen.

Funktionelle Schnittstelle DOIO

Mit DOIO wird der Datentransfer durchgeführt. Bei der Implementierung auf der S310 wird von DOIO ein Parameterblock vorbereitet und dann die Ausführung der Ein-/Ausgabe durch einen ORG Aufruf angestoßen. Für die Vorbereitung des Parameterblocks (ORG310) benötigt DOIO als Parameter die Adresse des dem E/A-Gerät zugeordneten Gerätekontrollblocks und die Adresse des Taskkontrollblocks, dessen zugeordnete Task den E/A-Auftrag gegeben hat. Durch Informationen aus dem Gerätekontrollblock wird das zugeordnete E/A-Gerät ermittelt. Dann wird der entsprechende Gerätetreiber aufgerufen, der den Datentransfer durchführt. Hier wird aus dem Taskkontrollblock die Anfangsadresse des E/A-Puffer geholt, der in seiner 1. Speicherzelle seine Endadresse enthalten muß. Die Pufferanfangs- und Pufferendadresse werden in den Parameterblock für den ORG-Aufruf eingetragen. Der genaue Aufbau dieser Parameterblöcke ist [TRAU78] zu entnehmen.

Dann wird noch eine Kenngröße gesetzt, die anzeigt, daß dieses Gerät einen Datentransfer bearbeitet. Dies ist nötig, damit die Vorverarbeitung feststellen kann, ob ein Gerät den Koordinierungszähler erhöht hat und damit das Ende des Transfers gemeldet hat. Abschließend wird wieder an die Warteposition zurückgesprungen (Koordinierungszähler erniedrigen). Bei den ORG-Aufrufen für die Ausführung des Datentransfers werden die Aufrufe vom Typ 'Warten mit Koordinierungszähler' [ORG310] verwendet. Dies hat den Vorteil, daß die auftraggebende Programmebene (in diesem Fall die PBS-Ebene) lauffähig bleibt und das Ende der E/A durch ein Erhöhen des im Parameterblock angegebenen Koordinierungszählers angezeigt wird. Im Rahmen dieser Arbeit wurden in DOIO die Bildschirmein- und -ausgabe, sowie die Druckerausgabe realisiert. Weitere Geräte können noch angeschlossen werden.

Funktionelle Schnittstelle ERROR

Diese funktionelle Schnittstelle wurde in dieser Arbeit nicht implementiert. Da sie nur Fehler des PBS anziegen soll, wurde ein Stop-Befehl eingesetzt.

Die Programmtexte für alle funktionellen Schnittstellen sind dem Anhang III zu entnehmen.

4. Übersetzen des PEARL-Betriebssystem von seiner Definitionssprache in die Zielsprache

4.1. Kurzbeschreibung der Definitionssprache (SPASS)

Der rechnerunabhängige Teil des PBS wurde in einer Programmiersprache implementiert, die eigens zu diesem Zweck entwickelt wurde, da an den Zielmaschinen, an denen das PBS implementiert werden sollte (S306, S310, Z80), keine einheitliche Systemsprache zur Verfügung stand. Der Name dieser Sprache ist SPASS (System-Programmiersprache auf ASSemblerniveau). [ROES79]

Die Eigenschaften von SPASS wurden so gewählt, daß eine einfache Übersetzung in Zielmaschinen-Assemblercode erfolgen kann, daß beim Einsatz moderner Maschinen effektiver Code entsteht, und daß trotzdem die Erstellung möglichst dokumentations- und wartungsfreundlicher Programme ermöglicht wird. Dies soll vor allem durch die weitgehende Formatfreiheit, durch die sich an höhere Sprachen anlehnende Notation und durch beliebig schachtelbare Strukturdefinitionen erreicht werden.

Die Sprache SPASS basiert auf einer virtuellen Maschine, die mit einem beliebig großen Speicher und 7 Registern arbeitet. Die Wortlänge des Speichers und der Register ist ebenfalls nicht festgelegt.

Konstante, Register und Speicherzellen können nach bestimmten Vorschriften durch Operatoren verknüpft werden. Speicherzellen können sowohl unmittelbar über ihren Namen (symbolische Adresen), als auch indirekt über die Register 1-3 - bei zusätzlicher Angabe eines festen Adressanteils (Offset) - adressiert werden. Außerdem kann auf Speicherzellen noch substituiert zugegriffen werden, d.h. eine Speicherzelle oder ein Register enthält die Adresse des Operanden.

Als Datentypen sind in SPASS INTEGER, POINTER und BITKETTE zugelassen. Aus diesen sogenannten Grundtypen können zusammengesetzte Typen (Datenstrukturen) aufgebaut werden.

Durch sogenannte Längendefinitionen wird angegeben, wieviel Speichereinheiten durch den jeweiligen Datentyp belegt werden. Die Datentypen INTEGER und POINTER können in arithmetischen Ausdrücken vorkommen. Daten vom Typ BITKETTE können nur bei Vergleichen und über eine Bitselektion als Entscheidung für bedingte Sprünge verwendet werden.

In SPASS sind folgende Anweisungen möglich:

Zuweisungen,
Sprungbefehle (bedingt und unbedingt, mit direkter oder indirekter Zielangabe, Unterprogrammsprünge, Rücksprung aus einem Unterprogramm),
Organisatorische Befehle (siehe Kapitel 2.3.3.2 und 3.4),
Strukturvereinbarungen,
Speicherplatzvereinbarungen (mit u. ohne Vorbesetzung),
Längenvereinbarungen,
Prozedurvereinbarungen.

In SPASS gibt es keine Blockstruktur. Je Karte ist eine Befehlszeile erlaubt. Eine Syntaxbeschreibung von SPASS befindet sich in Anhang I. Diese Syntaxbeschreibung weicht allerdings geringfügig von der Originalbeschreibung, wie sie von Herrn Rössler in [ROES79] angegeben wurde, ab. Bei der Übersetzung von SPASS (siehe Kapitel 4.2.) wird nach der Methode des rekursiven Abstiegs vorgegangen. Um den Programmieraufwand etwas zu verringern wurden die Produktionen 'Indizierung' und 'Substitution' aus der Originalsyntax in die Produktion 'Speicherzelle' eingesetzt, d.h.

statt: speicherzelle::= addresse/indizierung/substituion
indizierung::= offset, universalregister
substituion::= $0,addresse $0, register

wird nur noch:

speicherzelle::= adresse/offset, universalregister/$0,addresse/$0, registe

geschrieben. Die Sprache wird dadurch nicht verändert.

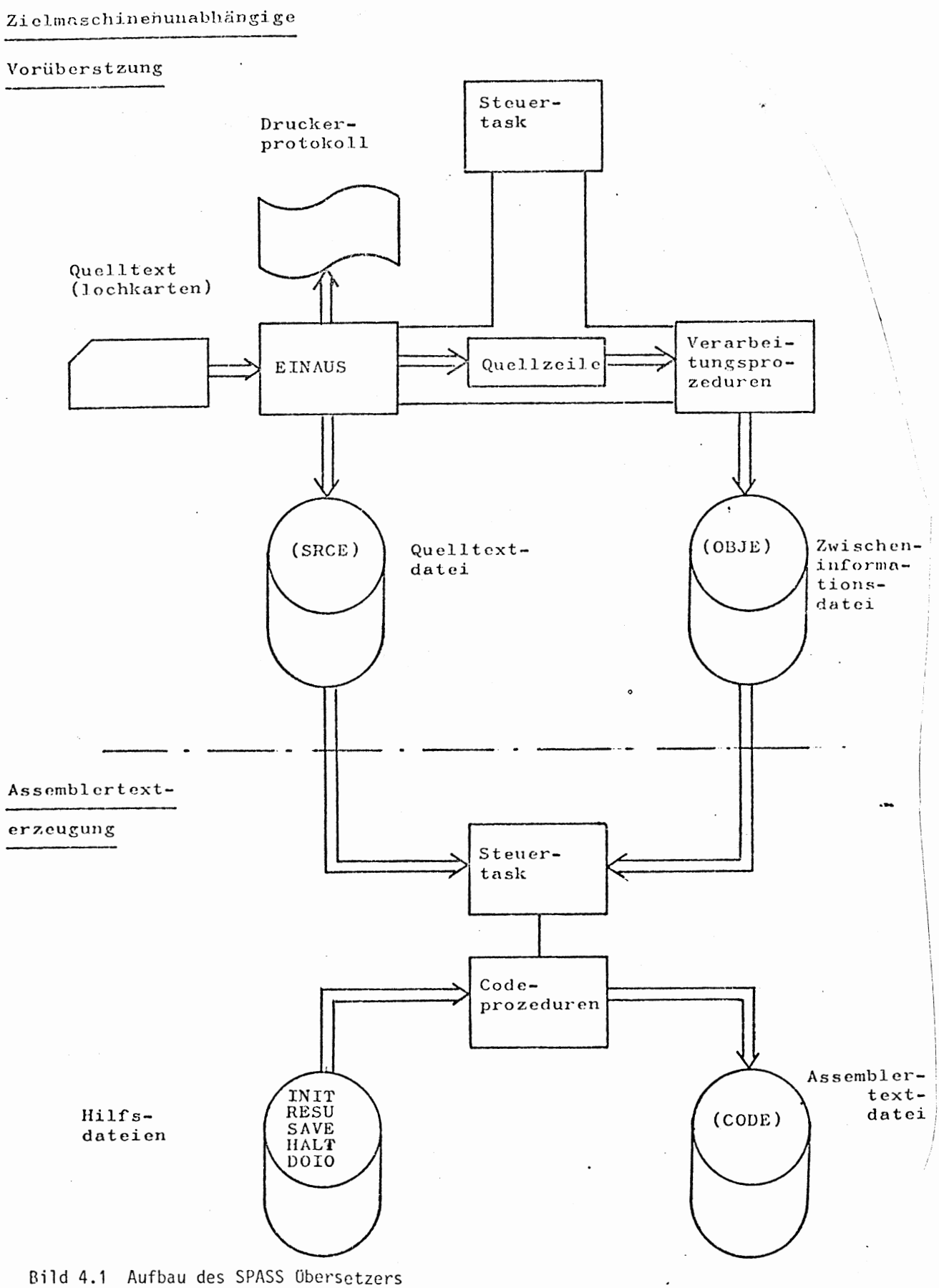


Bild 4.1 Aufbau des SPASS Übersetzers

4.2. Aufbau des SPASS-Übersetzers

Der SPASS-Übersetzer gliedert sich in 2 Teile: in eine zielmaschinenunabhängige Vorübersetzung und eine zielmaschinenabhängige Assemblercodeerzeugung. Beide Programme wurden in PEARL formuliert. Der Vorübersetzer liest den SPASS-Text vom Lochkartenleser ein und kopiert den Quelltext auf die Datei OBJE. Eine eingelesene Lochkarte (Befehlszeile) wird sofort übersetzt. Nachdem alle Karten eingelesen wurden, ist auch die Vorübersetzung zu Ende. Die vom Vorübersetzer erzeugte Zwischeninformation wird auf der Datei SRCE abgelegt.

Die Dateien 'OBJE' und 'SRCE' müssen im Lochkartenformat (80 Zeichen, alphanumerisch) beschrieben sein. Sie sind die Schnittstellen zur Assemblercodeerzeugung.
Diese holt sich die Informationen von diesen Dateien und erzeugt den entsprechenden Assemblertext. Für die Erzeugung des Assemblercodes ist die Datei OBJE nicht notwendig. Sie wurde nur angelegt, um die Fehlersuche in der Assemblercodeerzeugung zu erleichtern (siehe Kapitel 4.2.2.2).

Die Zweiteilung des SPASS-Compilers erhöht zwar den Programmieraufwand gegenüber einer total zielmaschinenabhängigen Übersetzung, erleichtert aber die Umstellung des SPASS-Compilers auf eine andere Zielmaschine. Der Zwischencode ist leichter zu verarbeiten und außerdem wurde die gesamte Syntaxprüfung schon im Vorübersetzer gemacht. Es muß nur noch die Assemblercodeerzeugung programmiert werden.

Bei der Übersetzung von SPASS für die Siemens 310 betrug der Anteil der Assemblercodeerzeugung ein Drittel des Gesamtumfangs. Einen Überblick über den Aufbau des SPASS-Compilers gibt Bild 4.1.

4.2.1. Zielmaschinenunabhängiger Vorübersetzer

4.2.1.1. Beschreibung des Aufbaus

Der zielrechnerunabhängige Vorübersetzer besteht, wie im vorigen Kapitel bereits erwähnt, aus mehreren Programmbausteinen bzw. PEARL-Moduln.
Die Steuertask liest eine Karte in ein 80-elementiges Zeichenfeld ein und kopiert dieses in die Datei OBJE. Nun werden die einzelnen Prozeduren für die Definitionen des nicht terminalen Symbols Programm

aufgerufen. Die Routinen sind Funktionsprozeduren. Gibt eine Prozedur den Wert 'true' zurück, war sie für die Bearbeitung der eingelesenen SPASS-Zeile zuständig und eine neue Karte kann eingelesen werden. Ist der Funktionswert 'false', wird die nächste Prozedur aufgerufen. Liefern alle Prozeduren den Wert 'false', wird eine Fehlermeldung gebracht und dann mit der Übersetzung fortgefahren.

Erkennt eine Routine, daß sie für die Bearbeitung einer SPASS-Zeile zuständig ist, so wird diese, mit Hilfe weiterer Routinen, analysiert und der entsprechende Zwischencode erzeugt.

Bild 4.2.1. gibt einen Überblick über die Struktur des Vorübersetzers. Die Namen in den Kästchen sind Prozedurnamen. Die Zuordnung von Produktion und Prozedurnamen findet man in Bild 4.2.2.

Im folgenden wird kurz erläutert, welche Prozeduren in welchem Prozedurmodul enthalten sind.

Der Modul PR Ø (Prozedurmodul Ø) enthält zwei Hilfsprozeduren zur Konvertierung von Zahlen von der Zeichendarstellung in die Integerdarstellung und umgekehrt. In diesem Modul befindet sich auch die Prozedur zur Ausgabe des Fehlertextes. Dieser wird bei etwaigen Fehlern innerhalb des Druckerprotokolls unterhalb der fehlerhaften Zeile ausgegeben.

Im Modul PR1 sind die Prozeduren für die Zeichenverarbeitung (NEXTCHAR, TESTWORD, NAMES) zusammengefaßt. Jede dieser Prozeduren greift auf das Feld TEXTZEILE zu. In diesem steht als Zeichenfeld die zu übersetzende SPASS-Zeile. Eine Größe "Textpointer" zeigt dabei auf das nächste zu verarbeitende Zeichen. Die Prozedur EINAUS liest die nächste zu verarbeitende SPASS-Zeile vom Kartenleser ein, beschreibt TEXTZEILE und setzt TEXTPOINTER auf das 1. Zeichen. Außerdem wird das Quellprotokoll des SPASS-Programms mit der entsprechenden Zeilennumerierung ausgegeben. EINAUS speichert die Quellfassung auf eine Datei. Dies ist nötig, um bei der Codegenerierung die SPASS-Zeile als Kommentarzeile über die Assemblerimplementierung zu setzen. Dies erleichtert das Testen des Vorübersetzers und des Codegenerators.

In Modul PR2 sind die einzelnen Routinen zur Verwaltung (Lesen, Schreiben) der Listen für die Datentypen (Namen, benötigte Anzahl der Speicherworte) und der Datenadressen (Namen) zusammengefaßt.

Außerdem befindet sich in diesem Modul die Routine (COMPF) zur Verwaltung (Eröffnen, Beschreiben, Schließen) der Datei für die Zwischeninformation (ZINFO).

In den Modulen PR3-PR8 sind die Produktionsregeln der SPASS-Syntax als Prozeduren wiedergegeben. Die untenstehende Tabelle gibt dabei die Zuordnung von nichtterminalen Symbol und Prozedur wieder. (Abb. 4.2.2.)

In den Modulen PR3-PR8 ist jeweils eine Prozedur PUTnDISK enthalten (n=Modulnummer). Diese Prozedur mußte eingebaut werden, um einen Fehler im PEARL-Compiler auszugleichen. Es ist nicht möglich, Prozeduren zu spezifizieren, die als formale Parameter mehr als ein Zeichenfeld enthalten. Diese Prozeduren PUTnDISK fassen mehrere Zeichenfelder zu einem großen Feld zusammen und dann erst wird die Prozedur COMPF (Compilerfehler) im Modul PR aufgerufen. Diese Prozedur zerlegt das großen Zeichenfeld wieder in seine Einzelfelder und gibt die Zwischeninformation auf Platte aus. Für formale Parameter, die als Integerzahlen spezifiziert sind, gibt es dieses Problem nicht. Wichtig zu erwähnen ist noch, daß einige SPASS-Zeilen in mehrere Zeilen Zwischeninformation aufgespalten werden. Dies ist bei der Zuweisung, dem Sprungbefehl und der Speicherreservierung der Fall. Bei einem Zuweisungsausdruck wird eine Zwischeninformationszeile für den arithmetischen Ausdruck, und je eine Zwischeninformationszeile für jede Zuweisung erzeugt.
Die Sprungzuweisung wird aufgespalten in Zwischeninformationszeilen für die Zuweisung oder den arithmetischen Ausdruck und eine Zwischeninformationszeile für den Sprung. Ein Beispiel für die Arbeitsweise des Vorübersetzers bei der Erzeugung des Zwischencodes ist im nächsten Kapitel zu finden.

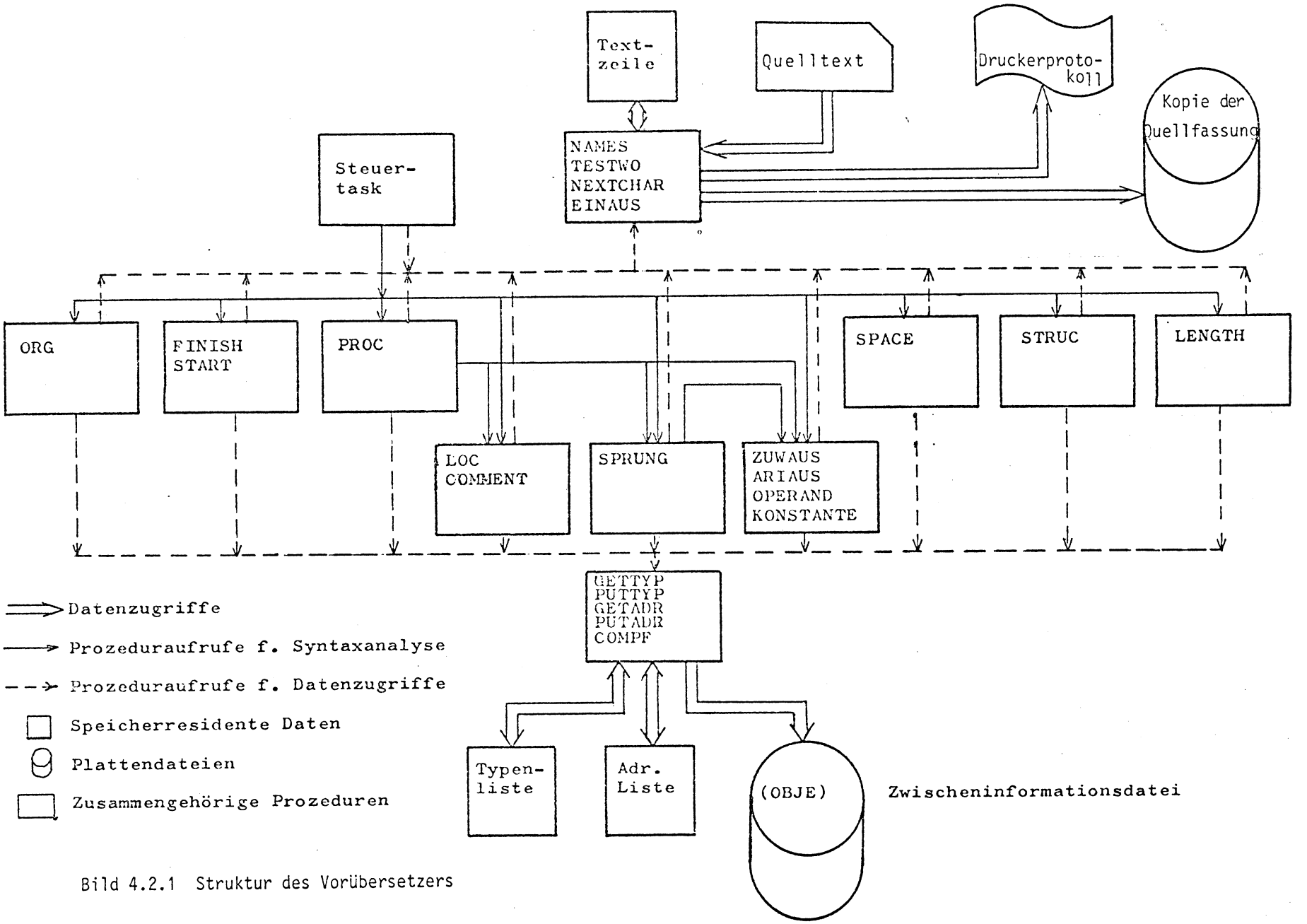


Bild 4.2.1 Struktur des Vorübersetzers

Modul	Prozeduren	zugeordnete Produktion	verwendet in Modul	verwendete Prozeduren	aus Modul
PR3	FINISH	FINSIH	./.	TESTWO,NEXTCHAR	PR1
	START	START		NAMES	
				FEHLER	PRØ
				COMPF	PR2
PR4	PROC	Prozedur	./.	ZUWAUS	PR6
				LOG,ORG	PR5
				COMMENT	
				SPRUNG	PR7
				COMPF	PR2
				TESTWO,NEXTCHAR	PR1
				NAMES,EINAUS	
				FEHLER	PRØ
PR5	ORG	Org.Befehl	PR4	TESTWO,NEXTCHAR,NAMES	PR1
	LOC	Markenvereinb.	PR4	COMPF	PR2
	COMMENT	eol	PR4	FEHLER	PRØ
PR6	ZUWAUS	Zuweisung	PR7,PR4	TESTWO,NEXTCHAR,NAMES	PR1
	ZUWEIS	Zuweisungs-ausdruck	./.	PUTADR,PUTTYP,COMPF	PR2
	ARIAUS	arith.Ausdr.	PR7	FEHLER	PRO
	OPERAND	Operand	PR7		
	REGISTER	Register	PR7		
	SPEICHER	Speicher	-		
	KONSTANTE	Konstante	PR8		
PR7	SPRUNG	Sprung	PR4	TESTWO,NEXTCHAR,NAMES	PR1
	BEDING	Bedingung	./.	ZUWAUS,ARIAUS,OPERAND	PR6
	VERGLEI	Vergleich	./.	REGISTER	
	SELEKT	Selektion	./.	FEHLER	PRØ
	LABEL	label	./.	COMPF	PR2
PR8	STRUC	Strukturver-einb.	./.	TESTWO,NEXTCHAR,NAMES	PR1
	SPACE	Speicherreser-vierung	./.	KONSTANTE	PR6
	LENGTH	Längenvereinb.	./.	FEHLER	
				GETADR,PUTADR	PR2
				GETTYP,PUTTYP,COMPF	

Bild 4.2.2. Zuordnung von Prozedur und Produktion sowie Verweis auf Prozeduren, die in einem Modul benutzt werden.

4.2.1.2. Beschreibung des vom Vorübersetzer erzeugten Zwischencodes

Der vom Vorübersetzer erzeugte Zwischencode ist streng formatiert; dadurch soll eine leichte Weiterverarbeitung ermöglicht werden. Eine Zwischenstringzeile enthält numerische Informationen (verschiedene Kennziffern) und mehrere Zeichenfelder, die zusätzliche Informationen enthalten und die Codeerzeugung erleichtern sollen. Diese Zeichenfelder sind jeweils 8 Zeichen lang. Jede Zwischeninformationszeile kann maximal acht Kennziffern (Bezeichnung: F1-F8) und sechs Zeichenfelder (Bezeichnung: Z1-Z6) enthalten.

In der Spalte F1 steht jeweils die Kennziffer des Befehlstyps (z.B. Zuweisung = 3) so daß in der Assemblercodeerzeugung schnell entschieden werden kann, welche Codeprozedur aufgerufen werden muß. Durch die feste Formatierung kann die Zwischeninformation leicht ausgewertet und in den entsprechenden Assemblercode umgewandelt werden. Einen Überblick über den Aufbau der Zwischeninformation gibt Bild 4.2.3.
Der Inhalt nicht benutzter Spalten einer Zwischeninformationszeile ist nicht definiert. Die Spalte F1 einer Zwischenstringzeile enthält jeweils die Kennziffer einer Anweisung. Der Inhalt der Spalte F1 gibt somit das Format der Zwischeninformationszeile an. Bei allen Formaten enthält die Spalte F8 die Nummer der Quellkarte, aus der die Zwischeninformation erzeugt wurde.

Bei den Zwischenstrings für Sprunganweisungen, arithmetische Ausdrücke, Speicherplatzvereinbarungen und Zuweisungen enthält die Spalte F7 der Zwischeninformation (siehe nächstes Kapitel) den Grundtyp der beteiligten Operanden und Adressen.

Der Grundtyp gibt an, ob eine Adresse oder eine Strukturkomponente, die beim Aufbau eines Operanden beteiligt ist, vom Typ POINTER, INTEGER oder BIT ist bzw. ob der Operand ein Register ist. Bei Integer-und Bitkonstanten wird ebenfalls die entsprechende Kennung angegeben

F7: 1 Register
2 POINTER
3 INTEGER
4 BITS
5 STRUKTUR
Ø sonst

Bei einem arithmetischen Ausdruck mit 2 Operanden steht in F7 eine 2-stellige Zahl. Die Zehner sind die Kennung für den Grundtyp des 1.Operanden und die Einerstellen für den zweiten.

Bedeutung von Art	F1 (Format)	F2	F3	F4	F5	F6	F7	Z1	Z2	Z3	Z4	Z5	Z6	F8	
START	1													Karten-nummer	
SPACE	2	Art der Vorbes.	Konst.typ bei Vorbes.				Grundtyp der Operanden	Adresse	Reservier-te Zellen	belegte Zellen	Konstante als Text			Karten-nummer	
ZUWEISUNG	3	Typ der Daten-senke					Grundtyp der Operanden	Datensenke	Offset					Karten-nummer	
SPRUNG	4	Sprung-art	Typ der Sprungziel-angabe	Art der Frage	Operator (Vergleich)	Typ des Operanden	Grundtyp der Ver-gleichs-operanden	Name des Sprung-ziels	Operanden-name	Offset	Zahl bei Selektion			Karten-nummer	
ORGANISAT. BEFEHL	5	Typ der org. An-weisung						Name (nur bei ERROR)	Zahl (nur bei ERROR)					Karten-nummer	
MARKE	6							Marken-name						Karten-nummer	
FINISH	7													Karten-nummer	
ARITH. AUSDRUCK	8	Art des arith.Aus-drucks	Operanden-typ 1	Operations-zeichen	Operanden-typ 2		Grundtyp der Operanden	Operand 1	Offset 1	Operand 2	Offset 2			Karten-nummer	
PROCEDUR	9	Beginn oder Ende						Name der Prozedur						Karten-nummer	
COMMENTAR	10													Karten-nummer	

Im folgenden soll Bild 4.2.3. näher erläutert werden.

Format: 1

Startanweisung

Der Inhalt aller Spalten ist bedeutungslos (außer F8)

Format: 2

Speicherreservierung

Bedeutung des Inhalts der numerischen Spalten

F2: Typ der Speicherreservierung

1: Beginn Speicherreservierung, ohne Vorbesetzung

2: Beginn Speicherreservierung, mit Vorbesetzung

3: Vorbesetzen von Speicherzellen

4: Ende einer Speicherreservierung

F3: Konstantentyp bei einer Vorbesetzung

11: Positive Ganzzahl

12: negative Ganzzahl

13: Offset

14: Adresskonstante

15: undefinierte Größe

16: Binärzahl

F7: Grundtyp der beteiligten Operanden

Bedeutung des Inhalts der Zeichenfelder

Z1: In dieser Spalte steht die Adresse der ersten reservierten Zelle, die durch eine Spaceanweisung angelegt wird.

Z2: Anzahl der zu reservierenden Zellen

Z3: Anzahl der mit einer Konstanten vorzubesetzenden Zellen

Z4: Konstante zur Vorbesetzung als Textstring

Die Bedeutung der Spalten Z1-Z4 richtet sich natürlich nach der Information aus F2.

Format: 3

Wertzuweisung

Bedeutung des Inhalts der numerischen Spalten

F2: Speicherplatztyp der Datensenke
(Kennziffern siehe Format 8)

F7: Grundtyp der Datensenke

Bedeutung des Inhalts der Zeichenfelder

Z1: Name des Speichers als Zeichenfolge

Z2: eventuell vorhandener Index (Offset) als Zahl in Zeichendarstellung

Format: 4

Sprunganweisung

Bedeutung des Inhalts der numerischen Spalten

F2: Art des Sprungs
- 1: unbedingter Sprung
- 2: bedingter Sprung
- 3: Sprung im Unterprogramm
- 4: Rücksprung aus einem Unterprogramm

F3: Art der Zielangabe
- 1: Direkte Sprungadresse
- 4: Substitution mit Adresse
- 3: Substitution mit Register

F4: Art der Abfrage bei bedingten Sprüngen
- 1: IF Abfrage
- 2: IFNOT Abfrage

F5: Art des Vergleichsoperators
- 1: Gleich (EQ)
- 2: Ungleich (NE)
- 3: Größer als (GT)
- 4: Kleiner gleich (LE)
- 5: Kleiner als (LT)
- 6: Größer gleich (GE)

F6: Typ des Operanden auf der rechten Seite des Vergleichsoperators
Kennziffern siehe F3 von Format 8 (Arith. Ausdruck)

F7: Grundtyp des Vergleichsoperanden (Z2)

Bedeutung des Inhalts der Zeichenfelder

Z1: Name des Sprungziels oder der Zelle, die das Sprungziel enthält als Zeichenfolge

Z2: Name der Operanden auf der rechten Seite des Vergleichsoperators

Z3: Name des eventuell vorhanden Offset (Zahl als Zeichenfolge)

Z5: Falls eine Selektion vorliegt, wird hier angegeben, das wievielte Bit des Operanden (Z3) abgefragt wird.

Format: 5

Organisatorische Befehle

Bedeutung des Inhalts der numerischen Spalten

F2: Typ der organisatorischen Anweisung

1: SAVE REGISTERS
2: SAVEALLREGISTERS
3: RESUMEPROCESS
4: DISABLE
5: ENABLE
6: DOIO
7: HALT
8: ERROR

Bedeutung des Inhalts der Textfelder

Z1: Bei ERROR enthält Z1 die Fehlerkennung

Z2: Bei ERROR enthält Z2 den Fehlertext

Format:6

Markenvereinbarung

Die numerischen Spalten (außer F1) sind bedeutungslos + F8

Bedeutung des Inhalts der Zeichenfelder

Z1: Name der Marke

Format: 7

Endeanweisung

Außer F1 hat keine Spalte eine Bedeutung

Format: 8

Arithmetischer Ausdruck

Bedeutung des Inhalts der numerischen Spalten

F2: Art des arithmetischen Ausdrucks
- 1: mit einem Operanden
- 2: mit zwei Operanden

F3: Art des 1. Operanden
- 11: positive Integerzahl
- 12: negative Integerzahl
- 13: Offset
- 14: Adresskonstante
- 16: Bitkonstante
- 2n: Register, wobei n die Registernummer
- 31: Adresse
- 32: Adressubstitutuion mit Register
- 33: Adressubstitition mit Speicherzelle
- 34: Indizierte Adresse

F4: Art der Verknüpfung bei 2 Operanden
- 1: Addition (+)
- 2: Multiplikation (*)
- 3: Subtraktion (-)

F5: Art des zweiten Operanden, falls vorhanden
Kennziffer siehe F3

F7: Grundtyp der beteiligten Operanden

Bedeutung des Inhalts der Zeichenfelder

Z1: Name des ersten Operanden

Z2: eventuell vorhandener Offset (Zahl als Zeichenfolge)

Z3 und Z4 analog Z1 und Z2 nur für einen eventuell vorhandenen zweiten Operanden

Format: 9

Prozedurvereinbarung

Bedeutung des Inhalts der numerischen Spalten

F2: Prozedurvereinbarungsbeginn oder -ende

1: Beginn der Prozedurvereinbarung

2: Ende der Prozedurvereinbarung

Bedeutung des Inhalts der Zeichenfelder

Z1: Prozedurname

Format: 10

Kommentarzeile

Außer F1 hat keine Spalte eine Bedeutung

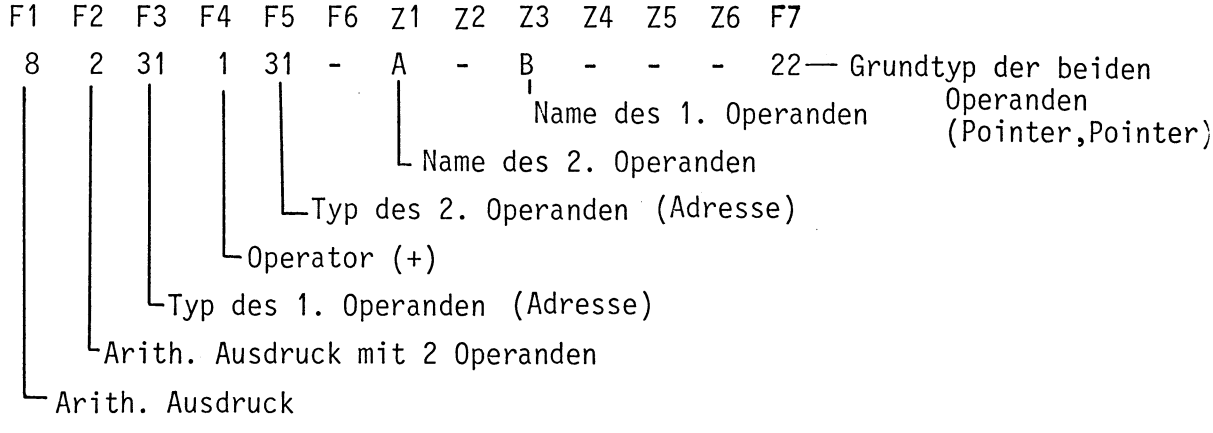
Die Arbeitsweise der Vorübersetzers bei der Erzeugung des Zwischenstrings soll nun an Hand eines Beispiels genauer beschrieben werden:

Beispiel:

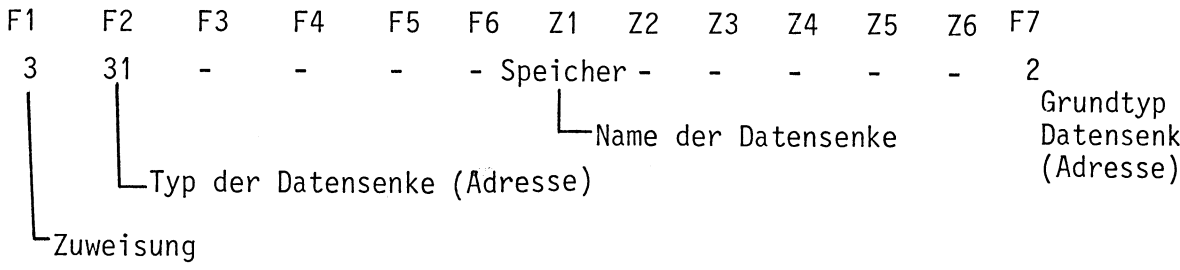
Quelltext R1:= SPEICHER:=A+B

Es handelt sich hier um eine Mehrfachzuweisung. Der Wert der Addition der Variablen A und B wird nach Speicher und in das Register R1 geschrieben.

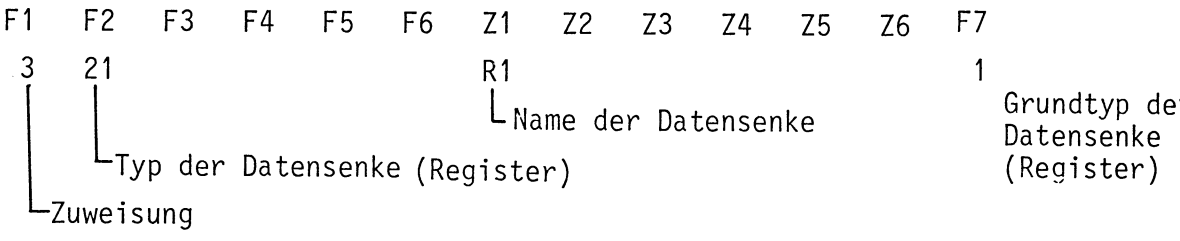
Der Vorübersetzer stellt am Ergibtzeichen fest, daß es sich um eine Zuweisung handelt. Dann wird das am weitesten rechts stehende Zuweisungszeichen gesucht. Der Vorübersetzer bearbeitet nun den arith. Ausdruck, also alles, was rechts vom letzten Ergibtzeichen steht. Es wird folgender Zwischenstring abgesetzt:



Für die Zeichenfolge links vom Arith. Ausdruck werden entsprechende Zwischeninformationen erzeugt und auf der Datei ZINF gespeichert (hinter dem obigen Datensatz). Für die Zuweisung ganz rechts wird folgender Zwischenstring erzeugt



Für die zweite Zuweisung wird folgende Zwischeninformation abgelegt:



Für diese Mehrfachzuweisungen werden also 3 Zeilen Zwischeninformation erzeugt.

4.2.1.3. Fehlermeldungen

Wie schon im Kapitel 4.2.1.1. beschrieben, ist jeder Produktion in der SPASS-Syntax (Ausnahme: Speicher) eine Prozedur im Vorübersetzer zugeordnet. Stellt nun eine Prozedur im SPASS-Quelltext einen Syntaxfehler fest, wird eine Prozedur FEHLER aufgerufen, der als Parameter eine Kennziffer übergeben wird.

Jeder Prozedur und damit jeder SPASS-Produktionsregel ist eine Kennziffer zugeordnet. Tritt ein Syntaxfehler auf, wird die Nummer der Prozedur, die den Fehler feststellt, an die Routine FEHLER übergeben. Diese meldet den Fehler im Druckerprotokoll unter der fehlerhaften Zeile. Durch die Nummer in der Fehlermeldung kann somit festgestellt werden, welche Produktionsregel verletzt wurde. Trat ein Fehler auf, wird an die Steuertask trotzdem der Wert 'true' zurückgegeben. Es können Folgefehlermeldungen auftreten. Entscheidend ist die erste Meldung. Damit konnte mit geringem Aufwand eine relativ einfache Fehlerdiagnose programmiert werden.

Die folgende Tabelle enthält die Zuordnungen Kennziffer - SPASS Produktionsregel-Übersetzerprozedur. So bedeutet die Meldung Fehler:9, daß ein falscher arithmetischer Ausdruck vorliegt. Die Fehlermeldung wird unter der falschen Quellzeile im Quellprotokoll ausgedruckt. Tritt ein Syntaxfehler auf, wird an das Ende der Prozedur gegangen. Der weitere Ablauf des Übersetzers ist nach dem Auftreten eines Fehlers, bis wieder in die Steuertask zurückgekehrt wird, nicht definiert. Die Steuertask liest eine Karte ein und der Übersetzungsvorgang kann weiterlaufen.

Fehlermeldungen

Fehler-Nr.	Produktions-regel	Übersetzer-prozedur	Übersetzer-Prozedur liegt in Modul
1	START	START	PR3
2	FINISH	FINISH	PR3
3	Prozedurver-einbarung	PROCEDURE	PR4
4	Organisatori-sche Befehle	ORG	PR5
7	Zuweisungs-ausdruck	ZWAUS	PR6
8	Zuweisung	ZUWEIS	
9	Arith.Ausdr.	ARIAUS	
10	Operand	OPERAND	
11	Register	REGIST	PR6
12	Speicher	SPEICH	PR6
13	Konstante	KONST	PR6
14	Sprungan-weisung	SPRUNG	PR7
15	Bedingung	BEDING	PR7
16	Vergleich	VERGLEICH	PR7
17	Selektion	SELEKTION	PR7
18	Label	LABEL	PR7
19	Struktur-vereinbarung	STRUC	PR8
20	Längenver-einbarung	LENGTH	PR8
21	Speicherplatz-vereinbarung	SPACE	PR8

4.2.2. Assemblercodeerzeugung für die Siemens 310

Nach dem Ende der Vorübersetzung muß die zielmaschinenabhängige Codeerzeugung gestartet werden. Dieser Teil des Übersetzers holt sich die einzelnen Datensätze der Zwischeninformation und erzeugt den entsprechenden Assemblercode. Jeder Art des Zwischenstrings (Anweisungstyp, Kennziffer im F1-Feld) ist eine Codeerzeugungsroutine zugeordnet.

Die erzeugten Assemblerzeilen werden auf ein entsprechendes File geschrieben. Dies geschieht mit einer weiteren Hilfsroutine (DISKAUS), die auch die gesamte Verwaltung (Eröffnen, Beschreiben, Schließen) des Codefiles (CODE) macht.

Diese Codeerzeugungsroutinen und Hilfsfunktionen sind wieder in einzelne PEARL-Module zusammengefaßt. Einen Überblick über die Codeerzeugung gibt Bild 4.2.4.

Der erzeugte Assemblertext wird auf der S306 mit Hilfe einer entsprechenden Crossoftware in Maschinencode für die S310 übersetzt. Der Maschinencode wird mit Hilfe einer Kopplung S306 - S 310 auf die S310 gebracht.
Für die Codeerzeugung sind noch folgende Hilfsdateien nötig:

INIT: Enthält den Code für die zielmaschinenabhängige Initialisierung und den Code für die Organisationsanweisung SAVE REGISTERS (Prozedur)
Diese Datei wird für die START Anweisung in den erzeugten Code eingesetzt (siehe Dokumentation von CODSTA).

SAVE: Enthält den Prozeduraufruf für die Ausführung von SAVE REGISTERS. Der Code für die aufgerufene Prozedur ist in der Datei INIT enthalten.

RESU: Enthält den Code für den Befehl RESUME PROCESS.

HALT: Enthält de- Code für den Befehl HALT.

DOIO: Enthält de- Code für den Befehl DOIO.

Die Dateien SAVE, RESU, HALT und DOIO werden nur von der Prozedur CODORG verwendet (siehe Dokumentation).

Der Inhalt der Dateien ist dem Anhang V zu entnehmen.

Die Codeerzeugung benutzt zum Speichern von Zwischenergebnissen Adressen, die mit 'Q' beginnen, d.h. sie fügt an SPASS-Adressen vorne ein Q an. Deshalb sollen in SPASS-Programmen keine Namen (Adressen) mit 'Q' beginnen.

Die Routinen zur Codeerzeugung sind relativ einfach, deshalb sollen sie hier nicht näher beschrieben werden. Die Programmdokumentation im Anhang dürfte ausreichend sein (Zielassembler). Eine Beschreibung des ASS 300 findet sich in [ASS300] .

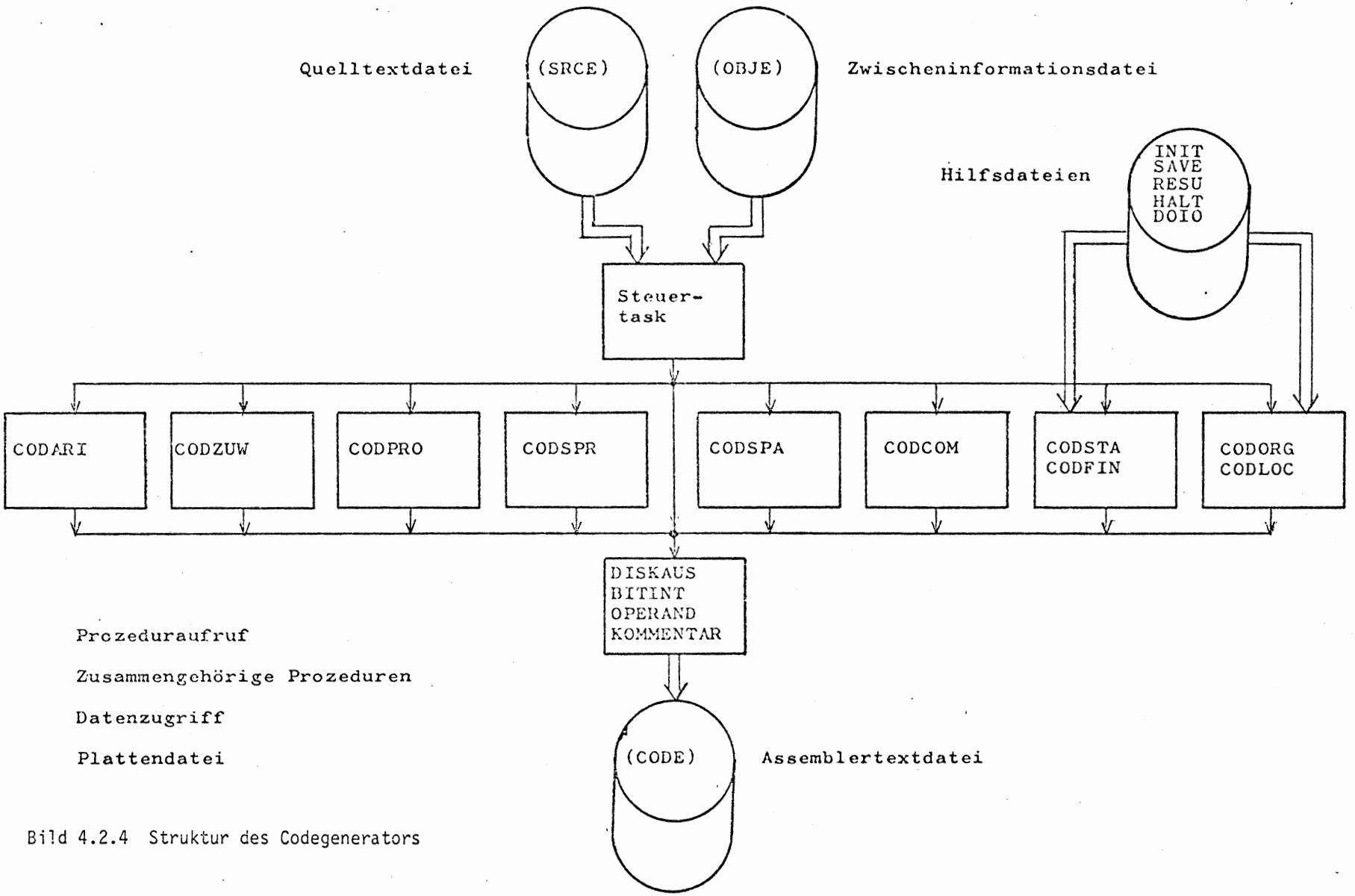


Bild 4.2.4 Struktur des Codegenerators

5. Integration und Test des PBS

Um das PBS für die S310 von seiner Quellform (Kartenpaket) in die auf der S310 lauffähige Form zu bringen, müssen mehrere Umsetzprogramme durchlaufen werden, die alle auf einer Rechenanlage Siemens 306 laufen. Der Maschinencode für den Rechner S310 wird anschließend mittels einer Rechnerkopplung auf diesen übertragen. Die einzelnen Schritte zur Erzeugung des Maschinencodes werden im folgenden näher beschrieben.

Der SPASS-Vorübersetzer liest den Quelltext des PBS, der auf Lochkarten vorliegen muß, ein, und erzeugt den entsprechenden Zwischencode, der auf der Datei 'OBJE' abgelegt wird. Zum Testen des Vorübersetzers wurde diese Datei am Schnelldrucker ausgegeben und dann manuell geprüft, ob die richtigen Zwischeninformationen erzeugt wurden.

Die Assemblertexterzeugung liest die Zwischeninformationsdatei Satz für Satz ein und erzeugt mit Hilfe von Hilfsdateien den entsprechenden Assemblertext, der auf der Datei 'CODE' abgelegt wird.

Zum Testen der Assemblertexterzeugung und der Vorübersetzung wurde der Assemblercode auf dem Schnelldrucker ausgegeben und dann manuell geprüft, ob der erzeugte Assemblercode der Zwischeninformation bzw. dem Quelltext entspricht.

Die Beschreibung dieser beiden Übersetzungskomponenten findet sich in Kapitel 4.

Für die Weiterverarbeitung des Assemblercodes steht auf der Siemens 306 Crossoftware für die S310 zur Verfügung.

Dazu mußte der Assemblertext erst von Siemens Interncode in ASCI-Code umgewandelt werden. Danach wurden die im Assemblertext enthaltenen Makros mit einem Makroübersetzer [SUMU75] bearbeitet. Die Ergebnisdatei dieses Programmdurchlaufs ist dann die Eingabedatei des Assemblers, der den erzeugten Maschinencode ebenfalls auf einer Datei ablegt.

Durch die Syntaxprüfung des Assemblers wurde nochmals die Richtigkeit der Assemblertexterzeugung überprüft.

Mit Hilfe einer Kopplung der Rechenanlagen S306 und S310 [PRES78] wird der erzeugte Maschinencode auf die S310 übertragen. Nun erst konnte das funktionelle Testen des PBS erfolgen. Dazu mußte noch die PEARL Taskebene durch ein entsprechendes Assemblerprogramm simuliert werden, da der PEARL-

Compiler für die S310 noch nicht fertig war. Anweisungen an das Betriebssystem wurden durch äquivalente Assemblerbefehlsfolgen ersetzt. Die Funktionen des PBS wurden schrittweise getestet. So wurde zunächst die Ein/Ausgabe getestet und dann die Synchronisationsmechanismen für Tasks (Beispieltestprogramm siehe Anhang VI). Als Unterstützung für den Funktionstest steht an der S310 ein Testhilfeprogramm [TH10 77] zur Verfügung, das aber zum Testen von parallelen Programmen ungeeignet ist. Durch das Setzen von Haltepunkten entweder in der PEARL-Taskebene oder im PBS wird der Koordinierungsmechanismus zwischen PBS- und PEARL-Taskebene gestört. Statt Haltepunkten wurde an den gewünschten Stellen mit Hilfe des Testhilfsprogramms "Sprünge auf sich selbst" (aktives Warten) eingetragen, dadurch konnte ein definiertes Verhalten des Gesamtsystems (PBS-Ebene und PEARL-Taskebene) erreicht werden.

Wurde nach einem der oben geschilderten Teilschritte ein Fehler festgestellt, so mußten, nachdem die Fehler behoben waren, je nach der Art der Fehler vorhergehende Teilschritte wiederholt werden.

Einen Überblick über den Übersetzungsweg von der Quellfassung des PBS zum Maschinencode gibt das folgende Bild 5.

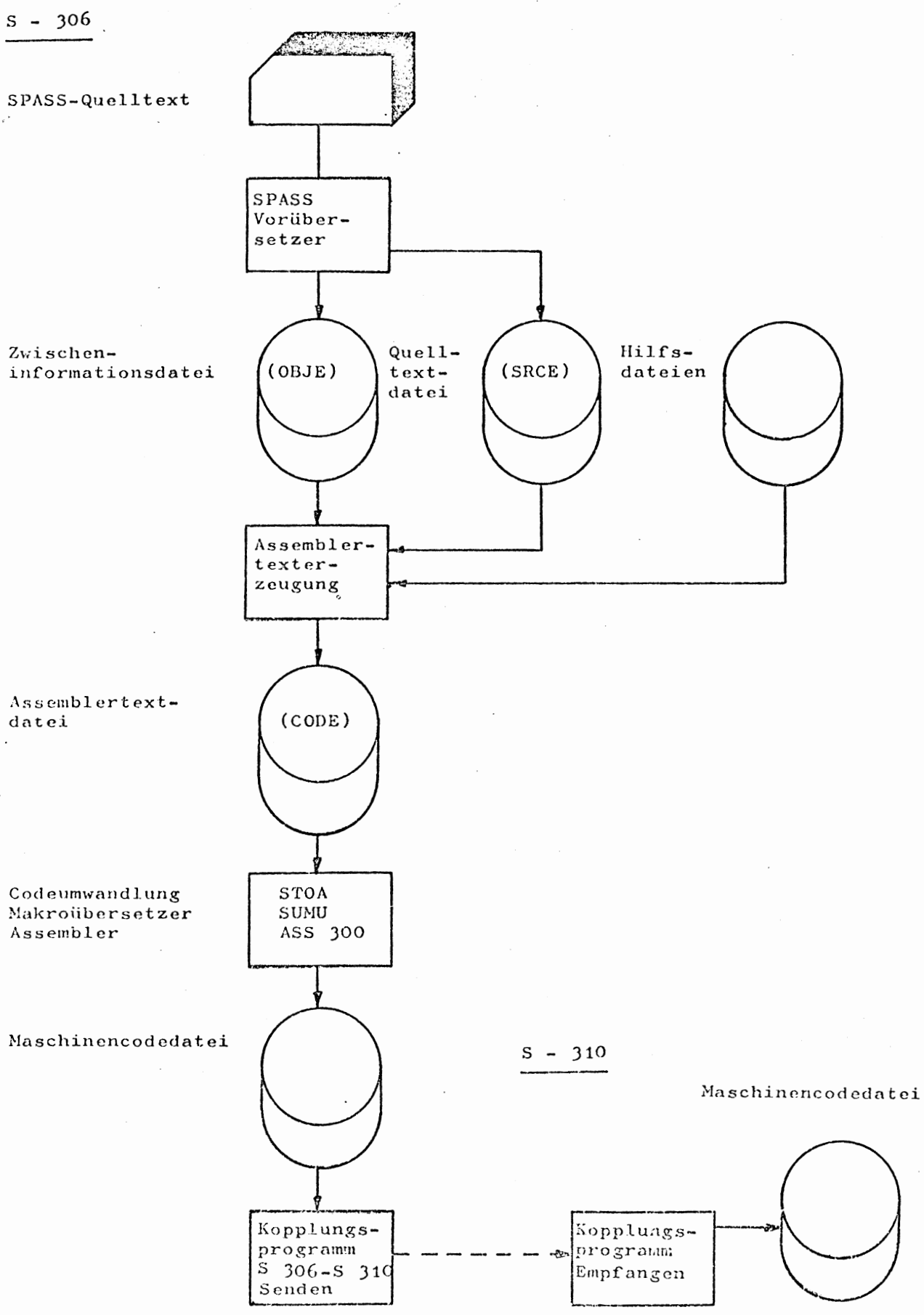


Bild 5 Übersetzung des PBS für die S310

Literaturliste

ASME76 Programmieranleitung für das ASME 1-PEARL-SUBSET.
PDV-Bericht 100 (1976) Gesellschaft für Kernforschung mbH, Karlsruhe

SCHN75 Schneider, H.J.: Compiler.
De Gruyter, Berlin (1975)

HOFM77 Hofmann, F.: Vorlesung zur Betriebsprogrammierung I
Univ. Erlangen (1977)

HOFM78 Hofmann, F.: Vorlesung zur Betriebsprogrammierung II
Univ. Erlangen (1978)

PRES78 Prester, F.-J.: Beschreibung der Rechnerkopplung S306-S310
Interner Bericht des 3. Physikalischen Instituts der Univ. Erlangen (1978

TRAU78 Trautner, M.: Beschreibung der Schnittstellen des PBS für die S310
zu den PEARL-Anwendertasks.
Interner Bericht des 3. Physikalischen Instituts der Univ.Erlangen (1978)

ASS300 SIEMENS SYSTEM300 Assembler. ASS300
Beschreibung P71100-D2000-x-x-35 (Dezember 1974)

ORG310 SIEMENS SYSTEM 300 Organisationsprogramm ORG310/MOBI.
Beschreibung P71100-B0310-x-A3-35

ZE310S SIEMENS SYSTEM 300 Zentraleinheit 310S.
Beschreibung. Bestell-Nr. E STE 4-121/00

ROES79 Roessler, R.: Betriebssystemstrategien zur Bewältigung von Zeit-
problemen in der Prozeßautomatisierung.
Eingereichte Dissertation Stuttgart 1979

ROES78 Rössler, R.: PEARL-Betriebssystem für den Z80.
PDV-Entwicklungsnotizen 119 (Juni 1978), Kernforschungszentrum Karlsruhe

SUMU75 SIEMENS SYSTEM 300 SUMU Sprachunabhängiger Makroübersetzer
Beschreibung 11M1.04.503 (1975)

ASSB74 SIEMENS SYSTEM 300 ASSB Assemblerübersetzer für Assemblersprache
der Prozeßrechner 320/330
Beschreibung. Best. Nr. 11K1.04.502 (1974)

TH1077 SIEMENS SYSTEM 300, Testhilfe TH10 Beschreibung
Bestell-Nr. P71100-E6019-x-x-35

FULL77 FULL-PEARL Language Description, PDV-Berich 130 (1977)
Gesellschaft für Kernforschung GmbH, Karlsruhe

Anhang I

Syntax von SPASS

```
programm::= START; eol
             längenvereinbarung ···
            [strukturvereinbarung ··· ]
            [speicherreservierung]{markenvereinbarung | zuweisung |
             sprung | orgbefehl | prozedur } ···
            FINISH; eol

prozedur::= PROC : adresse; eol
           {markenvereinbarung | zuweisung | sprung | orgbefehl }···
           END : adresse; eol

orgbefehl::={SAVE REGISTERS | SAVE ALL REGISTERS |
            RESUME PROCESS | DISABLE | ENABLE | DOIO | HALT |
             .ERROR. zahl 'name' } ; eol

sprung::=  {.TO. label |
           .TO. label {.IF. | .IFNOT.} bedingung |
           . CALL. label |
           . RETURN. adresse } ; eol

label::=     adresse | substitution

zuweisung::= zuweisungsausdruck; eol

zuweisungsausdruck::= {{register | speicherzelle} :=}···  arithausdruck

bedingung::= { zuweisungsausdruck | arithausdruck } vergleich | selektio

vergleich::=  vergleichsoperator  operand | .NIL. | .NULL.

vergleichsoperator::= .EQ. | .NE. | .GT. | .LT. | .GE. | .LE.

selektion::= operand (zahl)

arithausdruck::= operand [arithoperator  operand ]
```

```
arithoperator::=  + | - | *

operand::=        konstante | register | speicherzelle

speicherzelle::=    adresse | offset,universalregister |
                    $ Ø,adresse | $ Ø,register


register::=       universalregister | R4 | R5 | R6 | R7

universalregister::=   R1 | R2 | R3

markenvereinbarung::=  LOC  : adresse ; eol

speicherreservierung::=  SPACE : adresse = [zahl *] typ[+] ; eol
       [[{[zahl*] SPEL = konstante + ; eol }··· ]

       [zahl *] SPEL = konstante; eol ]

adresse::= name

konstante::= 'Kzahl' | 'KMzahl'|$ adresse | $ offset |
             $NIL | 'B binärzahl'

strukturvereinbarung::= STRUCT : strukturtyp; eol
                      [{ELEM : offset = typ; eol }··· ]
                        STREND : strukturtyp; eol

offset::= name

längenvereinbarung::= LENGTH : elementartyp = zahl; eol

eol::= kommentar EOL[{/kommentar EOL··· }]

strukturtyp::= name

typ::= elementartyp | strukturtyp

elementartyp::=  INT | POINTER | BITS
```

Anhang II

Dokumentation des Vorübersetzers

Allgemeines zur Dokumentation

Im folgenden wird der Vorübersetzer dokumentiert. Der Vorübersetzer wurde in PEARL geschrieben, die verwendeten Unterprogramme wurden aus PEARL-Übersetzungs- und Übersichtlichkeitsgründen in Prozedurmodule zusammengefaßt, die in aufsteigender Numerierung (PR Ø - PR 8) dokumentiert werden. Die einzelnen Prozeduren sind strukturiert programmiert, für die Darstellung der Algorithmen werden Nassi-Schneidermann-Diagramme benutzt.

Im Programmtext ist der Text der Diagramme an der entsprechenden Stelle als Kommentar eingefügt. Dadurch ist der Bezug Diagramm-Protokoll leichter herzustellen. Bei Prozeduren, deren Wirkungsweise einfach aus dem Programmtext zu ersehen ist, wird auf ein Nassi-Schneidermann-Diagramm verzichtet. In PEARL gibt es keine While Schleife, sie wurde durch Abfragen und Sprüng simuliert.

Die Beschreibung der einzelnen Module gliedert sich in eine allgemeine Übersicht, die angibt, welche Prozeduren in einem Modul enthalten sind, und in die Dokumentation der Prozeduren eines Moduls. Die Beschreibung der Prozeduren ist wie folgt gegliedert:

LEISTUNG:
Beschreibung, was die Prozedur macht und welche globalen Größen sie wie verändert.

PARAMETER:
Bedeutung und Typ der einzelnen Parameter und welche Werte sie vor und nach dem Durchlauf der Prozedur enthalten.

AUFGERUFENE PROZEDUREN IN MODUL:
Auflistung welche Prozeduren in welchem Modul durch die beschriebene Prozedur benutzt werden.

AUFRUFENDE PROZDUREN AUS MODUL
Auflistung der Prozeduren, die die beschriebene Prozedur aufrufen.

FEHLERAUSGAENGE:
Reaktion der Prozedur, falls sie einen Syntaxfehler feststellt. Es wird noch die Kennziffer der Prozedur (Fehler) angegeben und von wieviel Stellen eine Fehlermeldung erfolgen kann.

STRUKTOGRAMM

Darstellung der Arbeitsweise der Prozedur durch ein Nassi-Schneidermann-Diagramm. Falls der Algorithmus einer Prozedur aus dem Programmtext leicht zu ersehen ist, wurde das Diagramm weggelassen.

Am Ende der Beschreibung eines Moduls folgt sein Programmtext. Um die Dokumentation verstehen zu können, ist es nötig, die Syntax von SPASS, den Aufbau der Zwischeninformation, wie er in Kapitel 4.2.1.2. beschrieben ist, und die allgemeine Beschreibung des Vorübersetzers (siehe Kapitel 4.2.1.) zu kennen.

Allgemeines zum Vorübersetzer

Der Vorübersetzer wurde im PEARL-Subset der ASME-Stufe-1 programmiert, als Rechner wurde dabei eine Siemens 306 benutzt. Der Vorübersetzer benötigt als Massenspeicher eine Platteneinheit. Es werden zwei Plattenfiles (Namen der Files: QINFO, Z INFO) angelegt mit einem Gesamtbedarf an Platz von ca. der dreifachen Zeilenanzahl der Quellfassung des SPASS-Programms, die in Lochkartenform vorliegen muß. Der Vorübersetzer benötigt an der S306 16 K-Worte (24 Bit je Wort) Kernspeicher.

Die gesamte Textverarbeitung des Vorübersetzers erfolgt mit Zeichenfeldern. Dies war wegen der unzureichenden Möglichkeiten der Textverarbeitung in PEARL nötig. Die zu verarbeitende SPASS-Zeile (entspricht einer Lochkarte) wird in ein 80-elementiges Zeichenfeld (TEXTZEILE) eingelesen und von links nach rechts abgearbeitet. Wie weit die Zeile bereits ausgewertet wurde, wird durch einen Zeiger (TEXTPOINTER) angezeigt, der immer auf das nächste zu verarbeitende Zeichen zeigt.

In SPASS ist jeder Befehl durch ein Schlüsselwort gekennzeichnet (z.B. LOC = Markenvereinbarung). Dadurch ist es für ein Übersetzungsprogramm einfach, eine Befehlszeile zu klassifizieren. In diesem Vorübersetzer kann dies sehr einfach durch die Prozedur TESTWO erfolgen. Eine Ausnahme ist dabei die Wertzuweisung. Diese wird durch die Abfrage auf das Zuweisungszeichen klassifiziert.

Modul SPA

Der Hauptmodul SPA enthält die Deklarationen aller globalen Größen sowie die Task, die den gesamten Übersetzungsablauf steuert.

Task LAUF1

Leistung:

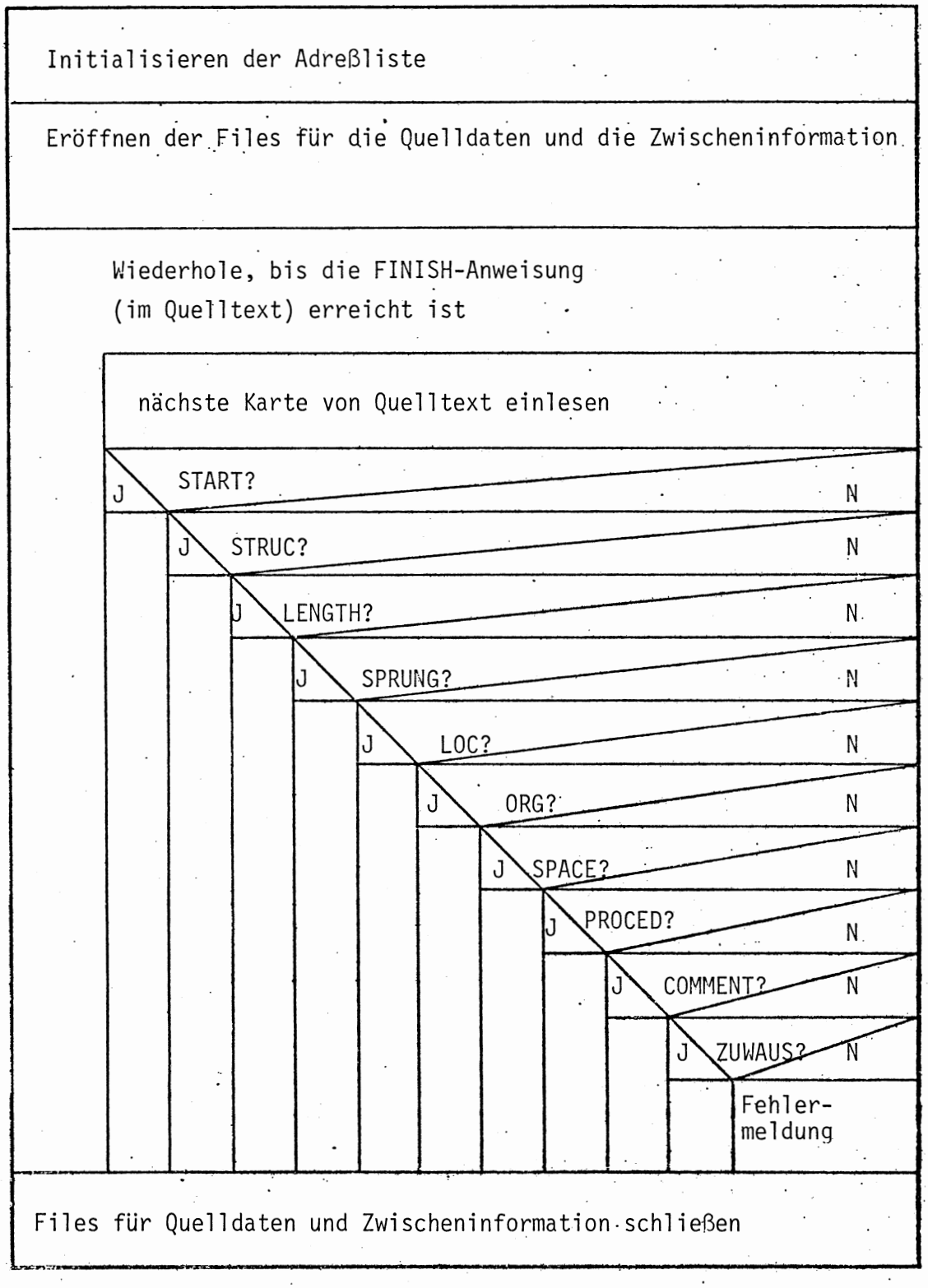
Lauf1 macht als erstes die Initialisierung einzelner globaler Größen. Sie veranlaßt, daß die Files und Dateien für die Kopie der Quelldaten und der Zwischeninformationen angelegt werden (Aufruf von EINAUS).
Außerdem steuert diese Task das Einlesen der einzelnen Quelldatenkarten nach TEXTZEILE und ruft dann die einzelnen Funktionsprozeduren auf, die die TEXTZEILE analysieren und die Werte TRUE oder FALSE zurückliefern. Bei TRUE ist die Syntaxanalyse und Übersetzung der TEXTZEILE in die Zwischeninformation bereits abgeschlossen. Bei FALSE zeigt der TEXTPOINTER wieder auf den Anfang von TEXTZEILE. Die Aufgaben der Steuertask sind Bild 3.1 zu entnehmen.

Aufgerufene Prozeduren	aus Modul
GETADR	PR2
COMPF	PR2
EINAUS	PR1
FINISH	PR3
START	PR3
STRUC	PR8
LENGTH	PR8
SPRUNG	PR7
LOC	PR5
ORG	PR5
ZUWEIS	PR6
SPACE	PR8
PROC	PR4

Fehlerausgänge:

Kennziffer: Ø

Falls eine Anweisung nicht identifiziert werden kann.
Die Übersetzung wird nicht abgebrochen.


Initialisieren der Adreßliste
Eröffnen der Files für die Quelldaten und die Zwischeninformation
Wiederhole, bis die FINISH-Anweisung
(im Quelltext) erreicht ist
nächste Karte von Quelltext einlesen
START?
J
N
STRUC?
J
N
LENGTH?
J
N
SPRUNG?
J
N
LOC?
J
N
ORG?
J
N
SPACE?
J
N
PROCED?
J
N
COMMENT?
J
N
ZUWAUS?
J
N
Fehler-
meldung
Files für Quelldaten und Zwischeninformation schließen

```
 1    MODULE SPA;
 2    SYSTEM;
 3    BSEA←→KONSOLE:;
 4    PSEA←→DISK:;
 5    SDAU→DRUCKER:;
 6          LKEI←LESER:;
 7    SIGN(2)←LKEUNKL:;
 8    SIGN(2Ø1)←NOFILE:;
 9    PROBLEM;
10    /* SPEZIFIKATION DER IN DIESEM MODUL VERWENDETEN PROZEDUREN */
11    DCL START ENTRY RETURNS (BIT) GLOBAL;
12    DCL FINISH ENTRY RETURNS (BIT) GLOBAL;
13    DCL SPACE ENTRY RETURNS (BIT) GLOBAL;
14    DCL SPRUNG ENTRY RETURNS (BIT) GLOBAL;
15    DCL ORG ENTRY RETURNS (BIT) GLOBAL;
16    DCL LOC ENTRY RETURNS (BIT) GLOBAL;
17    DCL PROCEDUR ENTRY RETURNS (BIT) GLOBAL;
18    DCL ZUWAUS ENTRY  RETURNS (BIT) GLOBAL;
19    DCL COMMENT ENTRY RETURNS (BIT) GLOBAL;
20    DCL STRUC   ENTRY RETURNS (BIT) GLOBAL;
21    DCL LENGTH  ENTRY RETURNS (BIT) GLOBAL;
22    DCL FEHLER ENTRY (FIXED) GLOBAL;
23    DCL EINAUS ENTRY (FIXED) GLOBAL;
24    DCL GETADR ENTRY ((8) CHAR,FIXED) GLOBAL;
25    DCL COMPF ENTRY (FIXED,FIXED,FIXED,FIXED,FIXED,FIXED,FIXED,
26                                                (4Ø)CHAR) GLOBAL;
27    /*****************************************************************
28    /*DEKLARATION DER GLOBALEN VARIABLEN */
29          DCL DISK VAL DEVICE GLOBAL;
30          DCL DRUCKER VAL DEVICE GLOBAL;
31          DCL LESER VAL DEVICE GLOBAL;
32          DCL TEXTZEILE(8Ø) CHAR GLOBAL;
33          DCL J FIXED GLOBAL;
34          DCL (ENDTYP,ENDADR) FIXED GLOBAL INITIAL(Ø);
35          DCL ADR(1ØØ,8) CHAR GLOBAL INITIAL(' ');
36          DCL TYP(1ØØ,8) CHAR GLOBAL INITIAL(' ');
37          DCL ARTTYP (1ØØ) FIXED GLOBAL INITIAL(Ø);
38          DCL ARTADR(1ØØ) FIXED GLOBAL INITIAL(Ø);
39          DCL KARTENZAEHLER FIXED GLOBAL;
40          DCL TEXTPOINTER FIXED GLOBAL;
41          DCL SATZZAHL FIXED GLOBAL INITIAL(1);
42          DCL SATZNR FIXED GLOBAL INITIAL(Ø);
43          DCL KONSOLE VAL DEVICE GLOBAL;
44
45    /*****************************************************************
46    /* STEUERTSK                                                     *
47    /*****************************************************************
48    LAUF1: TASK GLOBAL RESIDENT
49          DCL ZIEL(8) CHAR;
50          DCL J FIXED;
51          DCL K FIXED;
52          DCL VOR(5) CHAR(7) INITIAL('FIRSTCB','FIRSTSM','FIRSTIO','LAST
53                                                        ,'START
54          DCL ANTWORT CHAR;
55          DCL HILFE(4Ø) CHAR;
56          DCL QINFOM FIXED INITIAL(Ø);
57          DCL OPART FIXED INITIAL(2);
58          DCL FEHLART FIXED INITIAL(Ø);
59          /* INITIALISIERUNG DER ADRESSLISTE */
60          FOR K TO 5 REPEAT;
```

```
61            FOR J TO 7 REPEAT;
62               ZIEL(J)=J CHAR VOR(K);
63            END;
64            CALL GETADR(ZIEL,OPART);
65         END;
66         J=21;
67        /* ERÖFFNEN DER FILES FÜER DIE QELLDATEN UND DIE
68         ZWISCHENINFORMATION */
69         CALL COMPF(J,K,K,K,K,K,K,HILFE);
70         CALL EINAUS(QINFOM);
71         QINFOM=1;
72     /* BEGINN DER UEBERSETZUNG*/
73         KARTENZAEHLER=0;
74
75         /* WIEDRHOLE BIS DIE 'FINISH'-ANWEISUNG ERREICHT WIRD */
76   NEWLIN:;
77         /* NAECHSTE QELLKARTE EINLESEN */
78         CALL EINAUS(QINFOM);
79         /* ENDEANWEISUNG$ */
80         IF FINISH .EQ '1'B1 THEN GOTO ENDE;FI;
81         /* STARTANWEISUNG$ */
82         IF START   EQ '1'B1 THEN GOTO NEWLIN;FI;
83         */ STRUKTURVEREINBARUNG$ */
84         IF STRUC   EQ '1'B1 THEN GOTO NEWLIN;FI;
85         /* LAENGENVEREINBARUNG$ */
86         IF LENGTH  EQ '1'B1 THEN GOTO NEWLIN;FI;
87         /* SPRUNGANWEISUNG$ */
88         IF SPRUNG  EQ '1'B1 THEN GOTO NEWLIN;FI;
89         /* MARKENVEREINBARUNG$ */
90         IF LOC     EQ '1'B1 THEN GOTO NEWLIN;FI;
91         /* ORGANISATORISCHER BEFEHL$ */
92         IF ORG     EQ '1'B1 THEN GOTO NEWLIN;FI;
93         /* SPEICHERPLATZVEREINBARUNG$ */
94         IF SPACE   EQ '1'B1 THEN GOTO NEWLIN;FI;
95         /* PROCEDURVEREINBARUNG$ */
96         IF PROCED  EQ '1'B1 THEN GOTO NEWLIN;FI;
97         /* KOMMENTAR$ */
98         IF COMMENT EQ '1'B1 THEN GOTO NEWLIN;FI;
99         /* ZUWEISUNGSAUSDRUCK $ */
100        IF ZUWAUS   EQ '1'B1 THEN GOTO NEWLIN;FI;
101        CALL FEHLER(FEHLART);
102        GOTO NEWLIN;
103  ENDE:;      /*ENDE SCHLEIFE*/
104
105        /* FILES FUER QUELLDATEN UND ZWISCHENINFORMATION SCHLIESSEN */
106        QINFOM=2;
107        CALL EINAUS(QINFOM);
108        J=22;
109        CALL COMPF(J,K,K,K,K,K,K,HILFE);
110        PUT KONSOLE EDIT ('ENDE DER VORUEBERSETZUNG')(A(24));
111  END;
112  MODEND;
```

MODUL PRØ

Diese Modul enthält die Prozeduren zur Umwandlung von Zahlen, die als Zeichenfolge vorliegen in Integerzahlen und umgekehrt.

KONVERT

Leistung:

Die Prozedur "Konvert" ist eine Funktionsprozedur. Sie konvertiert vorzeichenlose Zahlen von der Zeichendarstellung in die Integerdarstellung. Der Funktionswert ist dabei der erhaltene Integerbetrag. Es können nur natürliche Zahlen mit maximal 8 Stellen konvertiert werden.

Parameter:

ALPHA: 8-elementiges Zeichenfeld.
Darin steht die zu konvertierende Zahl linksbündig. Im Feldelement ALPHA(1) steht die signifikanteste Stelle. Nicht benötigte Feldelemente (Zahl hat weniger als 8 Stellen) müssen mit blanks aufgefüllt werden. Der aktuelle Parameter wird nicht verändert.

Aufgerufene Prozeduren:	in Modul:
keine	

Aufrufende Prozeduren:	aus Modul:
SPACE	PR3

Fehlerausgänge:

keine

RKONVERT:

Leistung:

Die Prozedur RKONVERT ist eine CALL Prozedur. Mit ihr kann eine bis zu 3-stellige positive Integerzahl in eine entsprechende Ziffernfolge umgewandelt werden. Bei Zahlen mit mehr als 3 Stellen werden die Stellen nach der dritten Stelle nicht weiter beachtet. Es tritt also ein Konvertierungsfehler auf, der nicht gemeldet wird.

Parameter:

ZAHL: FIXED Größe
Enthält die zu konvertierende Integerzahl
Wird durch RKONVERT nicht verändert.

ZEICHEN: 8-elementiges Zeichenfeld.
Hier wird das Ergebnis der Konvertierung zurückgegeben.
Es werden in jedem Fall die ersten 3 Feldelemente benutzt, d.h. es treten führende Nullen auf. Die restlichen 5 Feldelemente werden mit Leerzeichen beschrieben.

Aufgerufene Prozeduren:	in Modul:
keine	

Aufrufende Prozeduren:	aus Modul:
SPACE	PR8

Fehlerausgänge:
keine

```
 1  MODULE PR0;
 2  SYSTEM;
 3        PSEA<-->DISK:;
 4        LKEI<-LESER:;
 5        SDAU->DRUCKER:;
 6  PROBLEM;
 7  DCL DRUCKER VAL DEVICE GLOBAL;
 8
 9
10  /*****************************************************************/
11  /* MIT 'KONVERT' KOENNEN ZAHLEN VON DER ZIFFERNDARSTELLUNG IN DIE  */
12  /* INTEGERDARSTELLUNG KONVERTIERT WERDEN                           */
13  /*****************************************************************/
14  KONVERT:PROC(ALPHA) RETURNS(FIXED)
15  GLOBAL
16        DCL ALPHA(8)  CHAR ;
17        DCL ZIFFERN(8) FIXED INITIAL(0);
18        DCL (I,K,L) FIXED ;
19        DCL HILFE  CHAR ;
20        DCL ZAHLEN(10) CHAR  INITIAL('0','1','2','3','4','5','6','7',
21                                                          '8','9');
22        L=1;
23        FOR I TO 8 REPEAT;
24             IF ALPHA(I) NE ' ' THEN
25                  BEGIN;
26                       FOR K TO 10 REPEAT;
27                            IF ALPHA(I) EQ ZAHLEN(K) THEN BEGIN;
28                                                          ZIFFER(L
29                                                               K-1
30                                                          L=L+1;
31                                                          END;
32                            FI;
33                       END;/*ENDE DER INNEREN SCHLEIFE*/
34                  END;
35             ELSE GOTO LOOPEND;
36             FI;
37        END;/*ENDE DER AEUSSEREN SCHLEIFE*/
38  LOOPEND:;
39        K=0;
40        IF L EQ 2 THEN K=ZIFFER(1)
41                                                  ;
42         ELSE
43          BEGIN;
44             FOR I TO (L-1) REPEAT;
45                  K=K+10**(L-1-I)*ZIFFER(I);
46             END;
47          END;
48        FI;
49        RETURN(K);
50  END;/*ENDE VON KONVERT*/
51
52  /*****************************************************************/
53  /* AUSGABE DER FEHLERMELDUNGEN                                   */
54  /*****************************************************************/
55  FEHLER:PROC (FEHLART) GLOBAL
56        DCL FEHLART FIXED;
57        DCL FEHLTEXT CHAR(10) INITIAL('FEHLER NR:');
58        PUT DRUCKER EDIT (FEHLTEXT,FEHLART)(A(10),F(3));
59        RETURN;
60  END;
```

```
 61
 62    /*************************************************************************/
 63    /* MIT RKONVERT KOENNEN INTEGERZAHLEN IN DIE ZIFFERNDARSTELLUNG     */
 64    */ KONVERTIERT WERDEN                                                */
 65    /*************************************************************************/
 66    RKONVERT:PROC(ZAHL,ZEICHEN)
 67    GLOBAL
 68          DCL ZAHL FIXED;
 69          DCL ZEICHEN(8) CHAR;
 70          DCL (I,K,L) FIXED;
 71          DCL ZIFFER(1Ø) CHAR INITIAL ('Ø','1','2','3','4','5','6','7','8'
 72                                                                    '9');
 73          K=ZAHL;
 74          L=Ø;
 75          /* HUNDERTERSYELLEN */
 76          FOR I TO 1Ø REPEAT;
 77                L=L+1;
 78                K=K-1ØØ;
 79                IF K LT Ø THEN BEGIN;
 80                                 ZEICHEN(1)=ZIFFER(L);
 81                                 K=K+1ØØ;
 82                                 GOTO END1ØØ;
 83                               END;
 84                FI;
 85          END;
 86    END1ØØ:L=Ø;
 87          /* ZEHNERSTELLEN */
 88          FOR I TO 1Ø REPEAT;
 89                L=L+1;
 90                K=K-1Ø;
 91                IF K LT Ø THEN BEGIN;
 92                                 ZEICHEN(2)=ZIFFER(L);
 93                                 K=K+1Ø;
 94                                 GOTO END1Ø;
 95                               END;
 96                FI;
 97          END;
 98    END1Ø:L=Ø;
 99          */ EINERSTELLEN */
100          FOR I TO 1Ø REPEAT;
101                L=L+1;
102                K=K-1;
103                IF K LT Ø THEN BEGIN;
104                                    ZEICHEN(3)=ZIFFER(L);
105                                    GOTO END1;
106                               END;
107                FI;
108          END;
109    END1:;
110          FOR I FROM 4 TO 8 REPEAT;
111                ZEICHEN(I)=' ';
112          END;
113          RETURN;
114    END;
115    MODEND;
```

MODUL PR1

Dieser Modul enthält die Prozeduren zur Textverarbeitung. Sämtliche dieser Prozeduren beziehen sich auf das Feld TEXTZEILE, in dem die zu verarbeitende SPASS-Zeile steht. Die Verarbeitung beginnt ab dem Zeichen (Feldelement), auf das der TEXTPOINTER zeigt. Nach dem Durchlauf einer Textverarbeitungsprozedur zeigt TEXTPOINTER auf das nächste zu verarbeitende Zeichen. Der Modul PR1 enthält außerdem die Prozedur für das Einlesen der SPASS-Quellform für die Erstellung des Druckerprotokolls und die Verwaltung des Plattenfiles für die Kopie des Quelltextes (QINFO).

TESTWO

Leistung:

TESTWO ist eine Funktionsprozedur. Funktionswerte können 'true' und 'false' sein. TESTWO prüft, ob die Zeichenfolge im aktuellen Parameter (Vergleichszeichenfolge) mit der Zeichenfolge in der TEXTZEILE, ab der Stellung des TEXTPOINTER übereinstimmt. In diesem Fall hat die Prozedur den Funktionswert true, ansonsten false. Bei true zeigt der TEXTPOINTER auf das nächste Zeichen nach der überprüften Zeichenfolge , das ungleich blank ist. Bei false wird die Stellung des TEXTPOINTERS nicht verändert.

Die Vergleichszeichenfolge darf maximal 8 Zeichen lang sein und muß mit '<' abgeschlossen werden (Endekriterium).

Parameter

SUCHWO: Parameter der Referenzstufe Ø, vom Typ CHAR(9). Die Vergleichsfolge wird mit dem Zeichen '<' abgeschlossen. Die restlichen Zeichen können beliebig sein. Wegen der Übersichtlichkeit sollten es jedoch immer blanks sein.

Aufgerufene Prozeduren:	in Modul:
NEXTCHAR	PR1

Aufrufende Prozeduren:	in Modul:
ZUWAUS	PR6
KONSTANTE	PR6
SPRUNG	PR6
VERGLEI	PR7
STRUC	PR8
SPACE	PR8
LENGTH	PR8
FINISH	PR3
START	PR3

Fehlerausgänge:

keine

NAMES

Leistung:

Die Prozedur NAMES ist eine CALL Prozedur. Diese Prozedur sucht in TEXTZEILE ab der Stellung von TEXTPOINTER die Zeichenfolge bis zum nächsten Sonderzeichen (delimiter) (mögliche Sonderzeichen siehe SPASS Syntax). Als Ergebnis liefert diese Prozedur die Textfolge bis zum nächsten Sonderzeichen und die Kennummer dieses Sonderzeichens.

1	2	3	4	5	6	7	8	9	10	11
+	*	-	.	;	:	=	,	(	)	'

Die Zeichenfolge von der Stellung des TEXTPOINTERS bis zum nächsten Sonderzeichen darf allerdings nicht länger als 8 Zeichen sein (Blanks werden nicht gezählt) (Die SPASS Syntax erlaubt nur Namen mit maximal 6 Zeichen).
Ist die Zeichenfolge länger, wird das Feld für den gefundenen Text mit blanks besetzt. Der Textpointer zeigt auf das Zeichen nach dem Sonderzeichen, das ungleich blank ist.

Parameter:

ERGEBNIS: 8-elementiges Zeichenfeld.
In diesem Feld steht linksbündig die gefundene Zeichenfolge bis zum nächsten Sonderzeichen. Nicht benötigte Feldelemente sind mit blanks besetzt.

SONDER: Integervariable.
In diesem Parameter wird die Kennummer des abschließenden Sonderzeichens zurückgegeben (siehe oben).

Aufgerufene Prozeduren:	in Modul:
NEXTCHAR	PR1

Aufrufende Prozeduren:	aus Modul:
alle Prozeduren	PR4
alle Prozeduren	PR5
alle Prozeduren	PR6
alle Prozeduren	PR7
alle Prozeduren	PR8

Fehlerausgänge:

keine

NEXTCHAR:

Leistung:

NEXTCHAR ist eine Funktionsprozedur. Die Prozedur sucht ab der Stellung des TEXTPOINTER das nächste Zeichen in TEXTZEILE, das ungleich blank ist. Der Funktionswert ist das gefundene Zeichen. Findet diese Prozedur bis zum achtzigsten Feldelement kein Zeichen, das ungleich blank ist, ist der Funktionswert der Prozedur NEXTCHAR gleich blank (der TEXTPOINTER zeigt anschließend auf das achtzigste Zeichen).

Parameter:
keine

Aufgerufene Prozeduren:	in Modul:
keine	

Aufrufende Prozeduren:	aus Modul:
ARIAUS	PR6
KONSTANTE	PR6
VERGLEI	PR7
SELEKT	PR7

Fehlerausgänge:
keine

EINAUS

Leistung:

EINAUS ist eine CALL Prozedur. Mit ihr wird eine neue Quellkarte nach TEXTZEILE eingelsen. Außerdem wird der TEXTPOINTER auf eins gesetzt. Die eingelesene Karte wird mit einer Zeilennummer versehen und am Drucker ausgegeben. Mit dieser Prozedur wird auch das File für die Kopie des Quelltextes verwaltet (eröffnen, beschreiben, schließen). Die Datei SRCE wird bei der Eröffnung des Quellfiles angelegt.

Parameter:

MODUS: FIXED Größe

Mit diesem Parameter kann der Auftrag für die Prozedur angegeben werden.

Kennummer	Auftrag
Ø	Datei SRCE einrichten und darauf das File QINF eröffn
1	Karte einlesen und nach TEXTZEILE schreiben Karte mit Zeilennummer auf Drucker ausgeben Karte nach QINFO schreiben
2	Quellfile (QINF) schließen

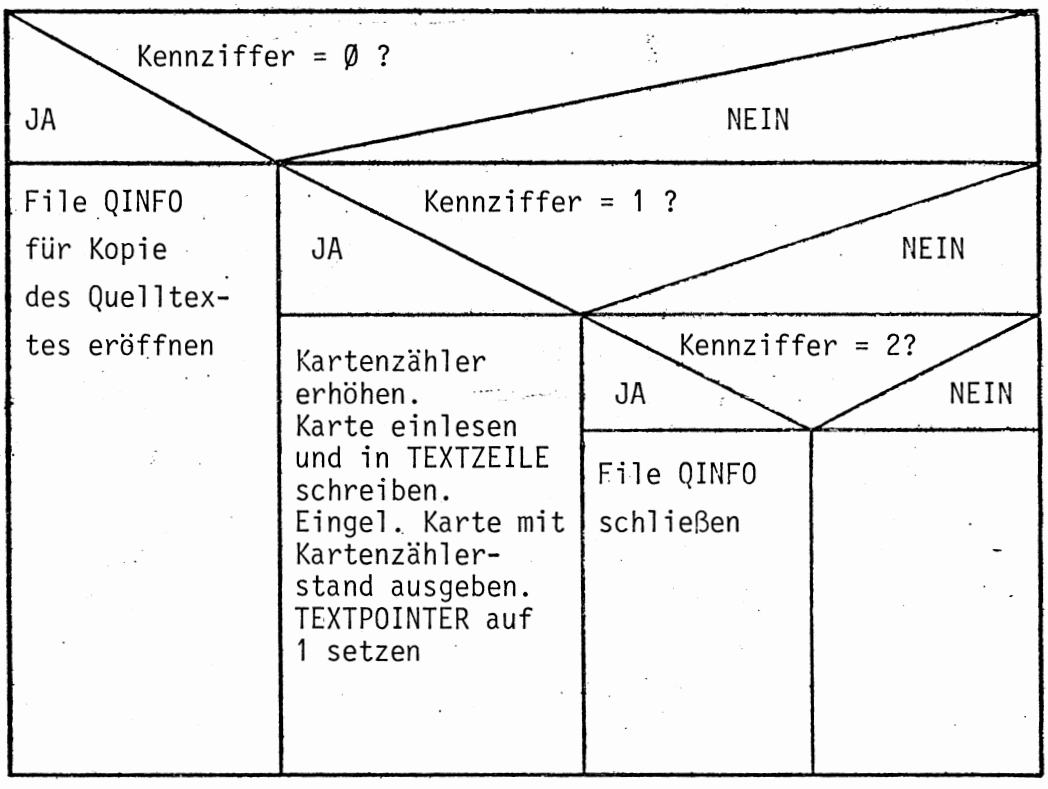
Andere Nummern werden zwar angenommen, aber haben keine Wirkung (auch keine Fehlermeldung).

Aufgerufene Prozeduren:	in Modul:
keine	

Aufrufende Prozeduren:	aus Modul:
Task Lauf1	SPA
STRUC	PR8
SPACE	
PROC	PR4

Fehlerausgänge:

Existiert auf der Platte eine Datei mit Namen SRCE, so bringt die S306 das Signal CRDX. Das Programm wird abgebrochen.


Kennziffer = Ø ?
JA
NEIN
File QINFO für Kopie des Quelltextes eröffnen
Kennziffer = 1 ?
JA
NEIN
Kartenzähler erhöhen. Karte einlesen und in TEXTZEILE schreiben. Eingel. Karte mit Kartenzählerstand ausgeben. TEXTPOINTER auf 1 setzen
Kennziffer = 2?
JA
NEIN
File QINFO schließen

```
1   MODULE PR1;
2   SYSTEM;
3         PSEA←→DISK:;
4         LKEI←LESER:;
5         SDAU→DRUCKER:;
6   PROBLEM;
7   DCL (DRUCKER,LESER) VAL DEVICE GLOBAL;
8   DCL DISK VAL DEVICE GLOBAL;
9   /* SPEZIFIKATION DER IN DIESEM MODUL VERWENDETEN GLOBALEN DATEN */
10  DCL QINFOM FIXED GLOBAL;
11  DCL KARTENZAEHLER FIXED GLOBAL;
12  DCL TEXTPOINTER FIXED GLOBAL;
13        DCL TEXTZEILE(8Ø) CHAR GLOBAL;
14
15  /*************************************************************************/
16  /* 'TESTWO' VERGLEICHT ZEICHENFOLGEN (SIEHE BESCHREIBUNG)                */
17  /*************************************************************************/
18  TESTWO:PROC (SUCHWO) RETURNS(BIT)
19  GLOBAL
20        DCL SUCHWO VAL CHAR(9);
21        DCL (BSUCH,BTEXT)CHAR;
22        DCL I FIXED;
23        DCL ERGEBNIS BIT;
24        DCL RETTEXTPOINT FIXED;
25        RETTEXTPOINT=TEXTPOINTER;
26        FOR I TO 9 REPEAT;
27             BSUCH=I CHAR SUCHWO;
28             BTEXT=NEXTCHAR;
29             IF BSUCH EQ '←' THEN BEGIN;
30                                  ERGEBNIS='1'B1;
31                                       TEXTPOINTER=TEXTPOINTER-1;
32                                       GOTO ENDE;
33                                   END;
34             FI;
35             IF BSUCH NE BTEXT THEN BEGIN;
36                                       TEXTPOINTER=RETTEXTPOINT;
37                                  ERGEBNIS='Ø'B1;
38                                       GOTO ENDE;
39                                    END;
40             FI;
41        END;/*ENDE DER SCHLEIFE*/
42  ENDE  :RETURN (ERGEBNIS);
43  END;/*ENDE VON TESTWORD*/
44
45  /*************************************************************************/
46  /* 'NAMES' SUCHT DAS NAECHSTE SONDERZEICHEN IN 'TEXTZEILE'                */
47  /*************************************************************************/
48  NAMES:PROC (ERGEBNIS,SONDER)
49  GLOBAL
50        DCL ERGEBNIS(8)  CHAR;
51        DCL SONDERNR  FIXED ;
52        DCL (I,K) FIXED;
53        DCL J FIXED;
54        DCL SZEICH(11) VAL CHAR(1)
55              IDENTICAL ('+','*','-','.',';',':','=',',','(',')','''');
56        SONDER=Ø;
57        FOR I TO 8 REPEAT;
58             ERGEBNIS(I)=NEXTCHAR;
59             FOR J TO 11 REPEAT;
60             /* SONDERZEICHEN GEFUNDEN$ */
```

```
61                IF ERGEBNIS(I) EQ SZEICH(J) THEN BEGIN;
62                                                  /* JA */
63                                                  FOR K FROM I TO 8 REPEAT
64                                                  ERGEBNIS(K)=' ';
65                                                  END;
66                                                  SONDERNR=J;
67                                                  GOTO ENDE;
68                                                END;
69                FI;
70                END;
71          END;/*ENDE DER SCHLEIFE*/
72    FALSCH:;
73          FOR K TO 8 REPEAT;
74                ERGEBNIS(K)=' ';
75          END;
76    ENDE: RETURN;
77    END;/*ENDE VON NAMES*/
78
79    /*************************************************************************
80    /* UND VERWALTET DIE DATEI 'SRCE'                                        *
81    /* 'EINAUS' LIEST DIE QUELLKARTEN EIN,GIBT SIE AM DRUCKER AUS            *
82    /*************************************************************************
83    EINAUS:PROC (QINFOM) GLOBAL
84          DCL QINFOM FIXED;
85          DCL QINFO VAL FILE;
86          DCL I  FIXED ;
87          /* DATEI 'SRCE' KREIEREN$    */
88          IF QINFOM EQ 0 THEN
89            BEGIN;
90               CREATE QINFO TITLE 'SRCE' UPON DISK OUTPUT DIR (80) ALPHA;
91               GOTO ENDE;
92            END;
93          FI;
94          /* DATEI SRCE SCHLIESSEN$*/
95          IF QINFOM EQ 2 THEN
96            BEGIN;
97               CLOSE QINFO;
98               GOTO ENDE;
99            END;
100         FI;
101         /*  QUELLKARTE EINLESEN$*/
102         IF QINFOM EQ 1 THEN
103           BEGIN;
104              KARTENZAEHLER=KARTENZAEHLER+1;
105              GET LESER EDIT (TEXTZEILE)((80)A(1));
106              PUT DRUCKER EDIT (KARTENZAEHLER,TEXTZEILE)(F(5),(5)X,(80)A
107
108              PUT QINFO POS (KARTEN) EDIT (TEXTZEILE)((80)A(1));
109              TEXTPOINTER=1;
110           END;
111         FI;
112   ENDE:;
113         RETURN;
114   END;/*ENDE VON EIN/AUS*/
115
116   /*************************************************************************
117   /* 'NEXTCHAR' SUCHT IN 'TEXTZEILE' DAS NAECHSTE       ZEICHEN DAS        *
118   /* UNGLEICH ' ' IST                                                      *
119   /*************************************************************************
120   NEXTCHAR;PROC RETURNS (CHAR)
```

```
121   GLOBAL
122         DCL NEXT  CHAR ;
123         DCL I FIXED;
124               FOR I FROM TEXTPOINTER TO 80 REPEAT;
125               NEXT=TEXTZEILE(I);
126               TEXTPOINTER=TEXTPOINTER+1;
127               IF NEXT NE ' ' THEN GOTO FIN ;FI;
128         END;
129   FIN:  RETURN(NEXT);
130   END;/*ENDE VON NEXTCHAR*/
131   MODEND;
```

MODUL PR2

=========

In diesem Modul sind die Prozeduren für die Verwaltung der Typenlisten, der Adreßlisten und der Datei für die Zwischeninformation.

GETADR.

Leistung:

GETADR ist eine CALL Prozedur
Diese Prozedur trägt neue Adressen und deren Grundtyp in die Adreßliste bzw. Grundtypliste ein. Die neue Adresse wird in einem 8-elementigen Feld als Zeichenfolge übergeben.
Die Adreßliste kann maximal 100 Adressen aufnehmen.

Parameter:

NEWADR: 8-elementiges Zeichenfeld
In diesem Feld wird der Prozedur der Name der einzutragenden Adresse übergeben.

ART: Fixed Größe
In diesem Parameter wird der Prozedur die Kennziffer des Grundtyps der einzutragenden Adresse (Inhalt von NEWADR) übergeben.

Aufgerufene Prozeduren:	in Modul:
keine	

Aufrufende Prozeduren:	aus Modul:
SPACE	PR8
Task: LAUF1	SPA

Fehlerausgänge:

Sollen mehr als 100 Adressen gespeichert werden, so gibt es einen Feldüberlauf, der durch das PEARL-Laufzeitsystem angezeigt wird (Signal: IXUS). Der Lauf des Vorübersetzers wird abgebrochen. In diesem Fall muß das globale Feld ADR verlängert werden.

PUTADR

Leistung:

Mit dieser CALL Prozedur kann abgefragt werden, ob ein bestimmter Name in der Adreßliste enthalten ist. Der zu überprüfende Name wird der Prozedur in einem 8-elementigen Zeichenfeld übergeben. Ist der Name enthalten, so wird er nicht

geändert, ansonsten wird er mit blanks überschrieben. In einem weiteren Parameter wird der aufrufenden Stelle der Grundtyp der gefundenen Adresse zurückgegeben. Ist der angegebene Name nicht in der Adressenliste enthalten, ist der Wert dieses Parameters undefiniert.

Parameter

ADRESSE: 8-elementiges Zeichenfeld.
In diesem Feld wird der Prozedur der zu überprüfende Name übergeben.

ART: Integer Wert
Grundtyp der gefundenen Adresse
(Schlüssel siehe Beschreibung der Zwischeninformation)

Aufgerufene Prozeduren:	in Modul:
keine	

Aufrufende Prozeduren:	aus Modul:
SPEICHER	PR6
KONSTANTE	PR6

Fehlerausgänge

keine

GETTYP

Leistung

Mit der CALL Prozedur GETTYP werden die Namen von neuen Typen und deren Länge (Anzahl der benötigten Speicherzellen, falls Speicher mit diesem Typ reserviert wird) in die Typenliste eingetragen. Für die Angabe der Typenlänge sind 2 Stellen vorgesehen. Diese Angaben erfolgen alle als Zeichenkette. Auch die Länge der Typen wird mit Zahlen in der Zeichendarstellung angegeben. Den Namen und die Länge eines einzutragenden Typs erhält die Prozedur in einem 8-elementigen Zeichenfeld. (6 Elemente für Namen, 2 Elemente für Länge).

Die Länge wird in der Typenliste rechtsbündig abgelegt. Es können bis zu 100 Typen gespeichert werden. Wird diese Zahl überschritten, gibt es einen Feldüberlauf. Bei einem aufzunehmenden Typ wird nicht geprüft, ob ein Typ gleichen Namens schon existiert. In einer weiteren Liste wird der Grundtyp des neuen Typs eingetragen.

Parameter

NEWTYP: 8-elementiges Zeichenfeld
In ihm werden der Prozedur in den ersten 6 Feldelementen der einzutragende Typname und in den letzten beiden die dazugehörige Typlänge übergeben.

ART: Fixed Größe
In diesem Parameter wird der Grundtyp des neuen Typs übergeben.

Aufgerufene Prozeduren:	in Modul:
keine	

Aufrufende Prozeduren:	aus Modul:
STRUC	PR8

Fehlerausgänge:

Sollen mehr als 100 Typen in die Liste eingetragen werden, meldet das System der S306 einen Feldüberlauf (Signal: IXUS) und bricht das Programm ab. In diesem Fall muß das globale Feld TYP verlängert werden.

PUTTYP

Leistung:

Mit dieser CALL-Prozedur kann überprüft werden, ob ein Typ bereits in die Typenliste eingetragen ist. Es muß dabei in einem 8-elementigen Feld der Name des Typs angegeben werden. Ist dieser enthalten, wird in den Feldelementen sieben und acht die dazugehörige Typlänge eingetragen. In einem weiteren Parameter wird an die aufrufende Stelle der Grundtyp des Typs zurückgegeben.
Ist er jedoch nicht enthalten, wird das gesamte übergebene Zeichenfeld gelöscht.

Parameter:

SUCHTYP: 8-elementiges Zeichenfeld
In den ersten 6 Feldelementen wird der zu überprüfende Typname übergeben. Falls er in der Typenliste enthalten ist, wird in das 7. und 8. Feldelement die Typlänge zurückgegeben. Ansonsten wird das gesamte Feld mit blanks beschrieben.

ART: Fixed Größe
In diesem Parameter wird der Grundtyp des gesuchten Typs an die aufrufende Stelle zurückgegeben. Ist der gesuchte Typ nicht in der Liste enthalten, ist der Inhalt von ART undefiniert.

Aufgerufene Prozeduren:	in Modul:
keine	

Aufrufende Prozeduren:	aus Modul:
SPECIHER	PR6
KONSTANTE	PR6
STRUC	PR8
SPACE	PR8

Fehlerausgänge:
keine

COMPF:

Leistung:
COMPF ist eine CALL Prozedur. Ihr wird die erzeugte Zwischeninformation übergeben, und zwar sind die einzelnen Felder des Zwischenstrings wegen eines Fehlers des PEARL Compilers zu einem großen Feld zusammengefaßt. Diese Prozedur zerlegt dieses große Feld in seine Telfelder und schreibt den Zwischencode auf das entsprechende FILE (ZINFO). Vorher wird an den Zwischenstring noch die Nummer der Karte des Quelltextes, von der der Zwischenstring kommt, angehängt. Dies wurde aus Testgründen gemacht. Außerdem verwaltet COMPF die Datei für die Zwischeninformation. Steht in dem Parameter ANWTYP 21 wird das File ZINFO eröffnet, bei ANWTYP = 22 geschlossen und ansonsten werden die übergebenen Informationen auf das File ZINFO geschrieben.

Parameter:

F1:	
F2:	
F3:	FIXED Größen
F4:	Bedeutung siehe Beschreibung
F5:	der Zwischeninformation
F6:	
F7:	

HILFE: 80-elementiges Zeichenfeld

Aufgerufene Prozeduren:	in Modul:
keine	

Aufrufende Prozeduren:	aus Modul:
PUT3DISK	PR3
PUT4DISK	PR4
PUT5DISK	PR5
PUT6DISK	PR6
PUT7DISK	PR7
PUT8DISK	PR8

Fehlerausgänge:

keine

```
  1   MODULE PR2;
  2   SYSTEM;
  3         PSEA<-->DISK:;
  4         SDAU->DRUCKER:;
  5         LKEI<-LESER:;
  6   PROBLEM;
  7   DCL DISK VAL DEVICE GLOBAL;
  8   /* SPEZIFIKATION DER IN DIESEM MODUL VERWENDETEN GLOBALEN DATEN*/
  9         DCL (ENDTYP,ENDADR) FIXED GLOBAL;
 10         DCL ADR(100,8) CHAR GLOBAL;
 11         DCL TYP(100,8) CHAR GLOBAL;
 12         DCL ARTADR(100) FIXED GLOBAL;
 13         DCL ARTTYP(100) FIXED GLOBAL;
 14   DCL (SATZNR,KARTENNR) FIXED GLOBAL;
 15
 16   /*************************************************************************
 17   /* 'GETADR' TRAEGT NEUE ADRESSEN IN DIE ADRESSENLISTE EIN              */
 18   /*************************************************************************/
 19   GETADR:PROC (NEWADR,ART)
 20   GLOBAL
 21         DCL NEWADR(8) CHAR;
 22         DCL ART FIXED;
 23         DCL I FIXED;
 24         ENDADR=ENDADR+1;
 25         FOR I TO 8 REPEAT;
 26               ADR(ENDADR,I)=NEWADR(I);
 27         END;
 28         ARTADR(ENDADR)=ART;
 29         RETURN;
 30   END;
 31
 32   /*************************************************************************/
 33   /* 'PUTTYP' SIEHT IN DER TYPENLISTE NACH OB EIN DATENTYP BEREITS       */
 34   /* EINGETRAGEN IST */
 35   /*************************************************************************/
 36   PUTTYP:PROC (SUCHTYP,ART)
 37   GLOBAL
 38         DCL SUCHTYP(8) CHAR;
 39         DCL ART FIXED;
 40         DCL (I,K,J) FIXED;
 41         J=0;
 42         FOR I TO ENDTYP REPEAT;
 43           BEGIN;
 44               J=J+1;
 45               FOR K TO 6 REPEAT;
 46                     IF TYP(I,K) NE SUCHTYP(K) THEN GOTO FIN;FI;
 47               END;
 48               GOTO GEFUNDEN;
 49               FIN:;
 50           END;
 51         END;
 52         /* DATENTYP IST NICHT IN DER TYPENLISTE ENTHALTEN */
 53         FOR K TO 8 REPEAT;
 54               SUCHTYP(K)=' ';
 55         END;
 56   GEFUNDEN:SUCHTY(7)=TYP(J,7) ;
 57           SUCHTY(8)=TYP(J,8);
 58         ART=ARTTYP(J);
 59         RETURN;
 60   END;
```

```
61
62    /*****************************************************************
63    /* 'GETTYP' TRAEGT NEUE DATENTYPEN IN DIE TYPENLISTE EIN         *
64    /*****************************************************************
65    GETTYP:PROC(NEWTYP,ART)
66    GLOBAL
67          DCL NEWTYP(8) CHAR;
68          DCL ART FIXED;
69          DCL I FIXED;
70          IF NEWTYP(8) EQ ' ' THEN
71            BEGIN;
72               NEWTYP(8)=NEWTYP(7);
73               NEWTYP(7)='Ø';
74            END;
75          FI;
76          ENDTYP=ENDTYP+1;
77          FOR I TO 8 REPEAT;
78               TYP(ENDTYP,I)=NEWTYP(I);
79          END;
80          ARTTYP(ENDTYP)=ART;
81          RETURN;
82    END;
83
84    /*****************************************************************
85    /* 'PUTADR' SIEHT IN DER ADRESSENLISTE NACH OB EINE ADRESSE IN DER *
86    /* ADRESSENLISTE ENTHALTEN IST                                   *
87    /*****************************************************************
88    PUTADR:PROC (ADRESSE,ART)
89    GLOBAL
90          DCL ADRESSE(8) CHAR;
91          DCL ART FIXED;
92          DCL (I,K,J) FIXED INITIAL (Ø);
93          FOR I TO ENDTYP REPEAT;
94               BEGIN;
95                    J=J+1;
96                    FOR K TO 6 REPEAT;
97                         IF ADR(I,K) NE ADRESSE(K) THEN GOTO FIN;FI;
98                    END;
99                    GOTO GEFUNDEN;
100                   FIN:;
101              END;
102         END;
103         /* ADRESSE IST NICHT IN DER ADRESSENLISTE ENTHALTEN */
104         FOR K TO 8 REPEAT;
105              ADRESSE(K)=' ';
106         END;
107   GEFUND:ART=ARTADR(J);
108         RETURN;
109   END;
110
111   /*****************************************************************
112   /* 'COMPF' VERWALTET DIE DATEI FUER DIE ZWISCHENINFORMATION       *
113   /*****************************************************************
114   COMPF:PROC (ANWTYP,F1,F2,F3,F4,F5,F6,HILFE) GLOBAL
115         DCL (ANWTYP,F1,F2,F3,F4,F5,F6) FIXED;
116         DCL HILFE(4Ø) CHAR;
117         DCL (Z1,Z2,Z3,Z4,Z5)(8) CHAR;
118         DCL ZINFO VAL FILE;
119         DCL I FIXED;
120         /* DATEI EROEFFNEN$ */
```

```
121         IF ANWTYP EQ 21 THEN
122           BEGIN;
123              CREATE ZINFO TITLE 'OBJE' UPON DISK OUTPUT DIR (8Ø) ALPHA;
124              GOTO ENDE;
125           END;
126         FI;
127         /* DATEI SCHLIESSEN$ */
128         IF ANWTYP EQ 22 THEN
129           BEGIN;
130              CLOSE ZINFO;
131              GOTO ENDE;
132           END;
133         FI;
134         /* DATEI BESCHREIBEN */
135         FOR I TO 8 REPEAT;
136              Z1(I)=HILFE(I);
137              Z2(I)=HILFE(I+8);
138              Z3(I)=HILFE(I+16);
139              Z4(I)=HILFE(I+24);
140              Z5(I)=HILFE(I+32);
141         END;
142         SATZNR=SATZNR+1;
143         PUT ZINFO POS(SATZNR) EDIT (ANWTYP,F1,F2,F3,F4,F5,F6,Z1,Z2,Z3,Z4,
144                                                              Z5,KARTEN)
145                    ((7)F(2),(8)A(1),(8)A(1),(8)A(1),(8)A(1) ,(8)A(1),
146                             (5)X,F(3));
147     ENDE:;
148         RETURN;
149     END;/*ENDE VON COMPF */
150     MODEND;
```

MODUL PR3

==========

Diese Modul enthält die Prozeduren zur Erkennung der SPASS Start- und Ende-anweisung.

START:

Leistung:

Die Funktionsprozedur START liefert den Wert true, falls in TEXTZEILE die SPASS-Startanweisung steht, ansonsten den Wert false. Diese Prozedur erzeugt auch den entsprechenden Zwischenstring und ruft die Ausgabeprozedur dafür auf. Liegt keine Startanweisung vor (Funktionswert = false), wird die Stellung des TEXTPOINTERS nicht verändert.

Parameter

keine

Aufgerufene Prozeduren:	in Modul:
TESTWO	PR1
PUT3DISK	PR3

Aufrufende Prozeduren:	aus Modul:
Task LAUF1	SPA

Fehlerausgänge:

keine

FINISH

Diese Prozedur verhält sich genauso wie die Prozedur START. Sie bearbeitet die Endeanweisung analog wie START die Startanweisung.

```
ASNE-PEARL-SUBSET/1   VERSION Ø575/Ø4   ***QUELLPROGRAMM***   MODUL PR3

 1   MODULE PR3;
 2   SYSTEM;
 3         PSEA←→DISK:;
 4         SDAU→DRUCKER:;
 5         LKEI←LESER:;
 6   PROBLEM;
 7   /* SPEZIFIKATION DER IN DIESEM MODUL VERWENDETEN PROZEDUREN */
 8   DCL COMPF ENTRY(FIXED,FIXED,FIXED,FIXED,FIXED,FIXED,FIXED,(4Ø)CHAR)
 9         GLOBAL;
1Ø   DCL TESTWO ENTRY (VAL CHAR(9) ) RETURNS (BIT) GLOBAL;
11   DCL FEHLER ENTRY (FIXED) GLOBAL;
12   /*************************************************************************/
13   /* PROZEDUR ZUM ERKENNEN DES SPASS-PROGRAMMANFANGS                       */
14   /*************************************************************************/
15   START: PROC RETURNS (BIT) GLOBAL
16         DCL ERGEBNIS BIT;
17         DCL ANTWORT BIT;
18         DCL RETTEXT FIXED;
19         DCL (F1,F2,F3,F4,F5,F6,ANWTYP) FIXED INITIAL(Ø);
2Ø         DCL (Z1,Z2,Z3,Z4,Z5)(8) CHAR INITIAL(' ');
21         ANTWORT=TESTWORD('START;←  ');
22         IF ANTWORT EQ '1'B1 THEN BEGIN;
23                                   ANWTYP=1;
24                                   ERGEBNIS='1'B1;
25                                    CALL PUT3DISK(ANWTYP,F1,F2,F3,F4,F5,
26                                    Z1,Z2,Z3,Z4,Z5);
27                                END;
28                           ELSE ERGEBNIS='Ø'B1;FI;
29         RETURN(ERGEBNIS);
3Ø   END;/*ENDE VON START*/
31
32   /*************************************************************************
33   /* PROZEDUR ZUM ERKENNEN DES SPASS-PROGRAMMENDES */
34   /*************************************************************************/
35   FINISH:PROC RETURNS(BIT)
36   GLOBAL
37         DCL ANTWORT BIT;
38         DCL ERGEBNIS BIT;
39         DCL RETTEXT FIXED;
4Ø         DCL (F1,F2,F3,F4,F5,F6,ANWTYP) FIXED INITIAL(Ø);
41         DCL (Z1,Z2,Z3,Z4,Z5)(8) CHAR INITIAL(' ');
42         ANTWORT=TESTWORD('FINISH;← ');
43         IF ANTWORT EQ '1'B1 THEN BEGIN;
44                                   ERGEBNIS='1'B1;
45                                    ANWTYP=7;
46                                    CALL PUT3DISK(ANWTYP,F1,F2,F3,F4,F5,
47                                    Z1,Z2,Z3,Z4,Z5);
48                                  END;
49                                  ELSE ERGEBNIS='Ø'B1;FI;
5Ø         RETURN(ERGEBNIS);
51   END;/*ENDE VON FINISH*/
52
53   /*************************************************************************
54
55   PUT3DISK:PROC (ANWTYP,F1,F2,F3,F4,F5,F6,Z1,Z2,Z3,Z4,Z5) GLOBAL
56         DCL (ANWTYP,F1,F2,F3,F4,F5,F6) FIXED;
57         DCL (Z1,Z2,Z3,Z4,Z5)(8) CHAR;
58         DCL I FIXED;
59         DCL HILFE(4Ø) CHAR;
6Ø         FOR I TO 8 REPEAT;
```

```
61              HILFE(I   )=Z1(I);
62              HILFE(I+8)=Z2(I);
63              HILFE(I+16)=Z3(I);
64              HILFE(I+24)=Z4(I);
65              HILFE(I+32)=Z5(I);
66            END;
67            CALL COMPF(ANWTYP,F1,F2,F3,F4,F5,F6,HILFE);
68            RETURN;
69      END;/*ENDE VON PUTDISK*/
70      MODEND;
```

MODUL PR4
=========

Dieser Modul enthält die Prozedur zum Erkennen einer Prozedurvereinbarung.

PROC:

Leistung:

Diese Funktionsprozedur prüft, ob eine SPASS-Prozedurvereinbarung vorliegt. Ist dies der Fall, dann wird der Zwischenstring für den Prozeduranfang erzeugt die Anweisungen in der Prozedur klassifiziert, die dazugehörigen Zwischenstringt erzeugt, der Zwischenstring für das Prozedurende (Rücksprung) erzeugt und der Funktionswert auf true gesetzt. Diese Prozedur muß Quellkarten einlesen können, da die einzelnen Anweisungen innerhalb der Prozedur auf jeweils einer Karte stehen (Zugriff auf die Prozedur EINAUS).
Liegt keine Prozedurvereinbarung vor, zeigt TEXTPOINTER auf das Zeichen wie vor dem Aufruf von PROC.

Parameter

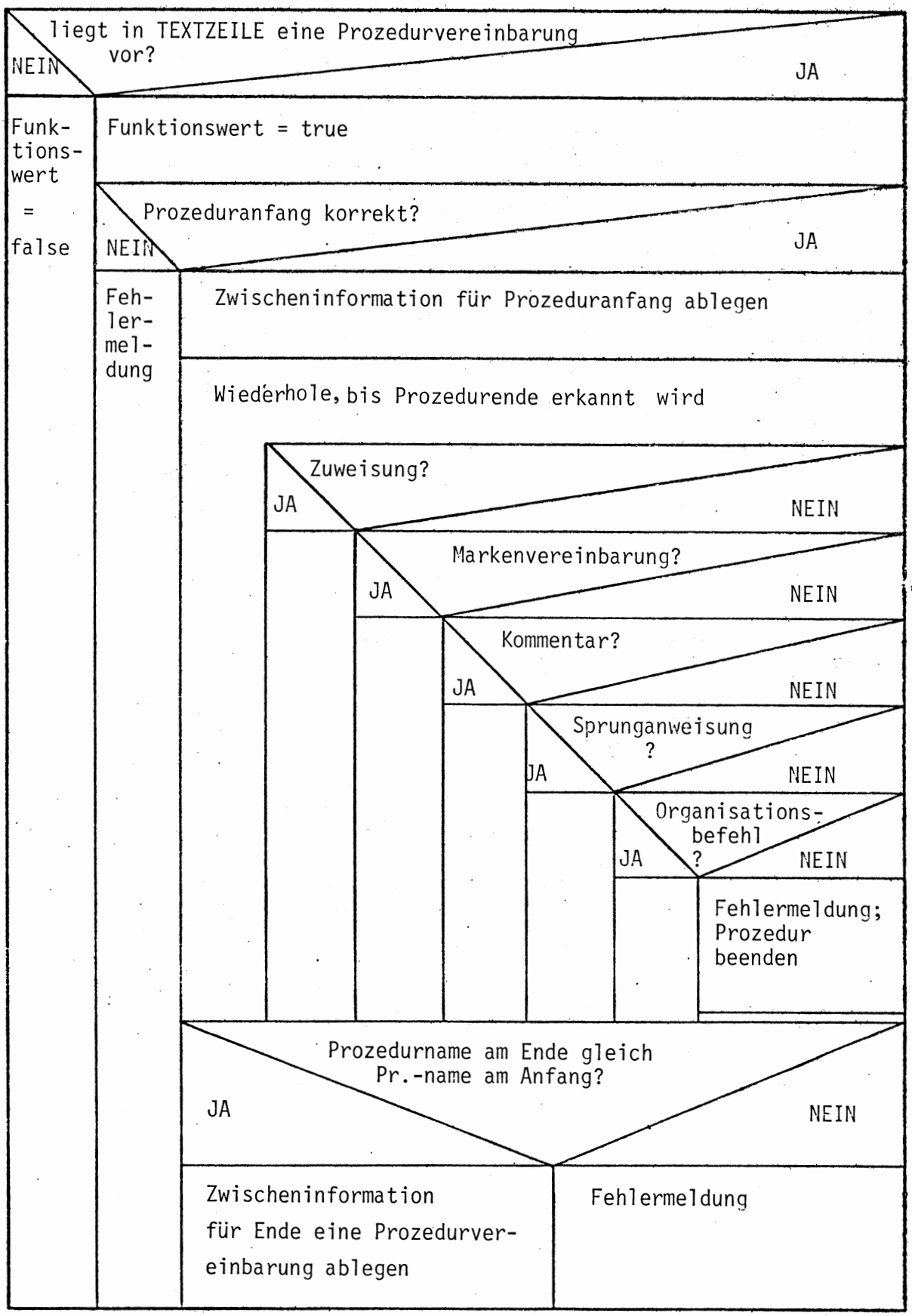
keine

Aufgerufe Prozeduren:	In Modul:
TESTWO	PR1
NAMES	PR1
FEHLER	PR1
PUT4DISK	PR4
ZUWAUS	PR6
LOC	PR5
COMMENT	PR5
SPRUNG	PR7
ORG	PR5

Aufrufende Prozeduren:	aus Modul:
TASK LAUF1	SPA

Fehlerausgänge:

Erkennt die Prozedur, daß in der SPASS-Quellzeile ein Syntaxfehler vorliegt, wird eine Fehlermeldung (Aufruf der Prozedur FEHLER) gebracht (Fehlernr.:3) und die Prozedur PROC beendet. Die Stellen, an denen eine Fehlermeldung gebracht wird, sind im Diagramm sichtbar.


liegt in TEXTZEILE eine Prozedurvereinbarung vor?
NEIN
JA
Funk-tions-wert = false
Funktionswert = true
Prozeduranfang korrekt?
NEIN
JA
Fehler-mel-dung
Zwischeninformation für Prozeduranfang ablegen
Wiederhole, bis Prozedurende erkannt wird
Zuweisung?
JA
NEIN
Markenvereinbarung?
JA
NEIN
Kommentar?
JA
NEIN
Sprunganweisung ?
JA
NEIN
Organisations-befehl ?
JA
NEIN
Fehlermeldung; Prozedur beenden
Prozedurname am Ende gleich Pr.-name am Anfang?
JA
NEIN
Zwischeninformation für Ende eine Prozedurver-einbarung ablegen
Fehlermeldung

```
   1    MODULE PR4;
   2    SYSTEM;
   3          PSEA←→DISK::;
   4          LKEI←LESER::;
   5          SDAU→DRUCKER::;
   6    PROBLEM;
   7    /* SPEZIFIKATION DER IN DIESEM MODUL VERWENDETEN GLOBALEN PROZEDUREN*
   8    DCL TESTWO ENTRY (VAL CHAR(9) ) RETURNS (BIT) GLOBAL;
   9    DCL NAMES ENTRY ((8) CHAR,FIXED) GLOBAL;
  10    DCL FEHLER ENTRY(FIXED) GLOBAL;
  11    DCL COMPF ENTRY(FIXED,FIXED,FIXED,FIXED,FIXED,FIXED,FIXED,(40)CHAR)
  12          GLOBAL;
  13    DCL COMMENT ENTRY RETURNS (BIT) GLOBAL;
  14    DCL ZUWAUS ENTRY  RETURNS (BIT) GLOBAL;
  15    DCL SPRUNG ENTRY RETURNS (BIT) GLOBAL;
  16    DCL LOC ENTRY RETURNS (BIT) GLOBAL;
  17    DCL ORG ENTRY RETURNS (BIT) GLOBAL;
  18    DCL EINAUS ENTRY (FIXED) GLOBAL;
  19
  20    /*************************************************************************/
  21    /* 'PROC' ERKENNT UND ANALYSIERT DIE PROZEDURVEREINBARUNGEN              */
  22    /*************************************************************************/
  23    PROCEDURE: PROC RETURNS (BIT) GLOBAL
  24         DCL ERGEBNIS BIT;
  25         DCL NAME(8) CHAR;
  26         DCL PNAME(8) CHAR;
  27         DCL SONDER FIXED;
  28         DCL CODNR FIXED;
  29         DCL ZIEL(8) CHAR;
  30         DCL I FIXED;
  31         DCL (Z2,Z3,Z4,Z5)(8) CHAR INITIAL(' ');
  32         DCL (F2,F3,F4,F5,F6) FIXED INITIAL(Ø);
  33          DCL FEHLART FIXED INITIAL (3);
  34         DCL ANWTYP FIXED;
  35          /* LIEGT EINE PROZEDURVEREINBARUNG VOR$ */
  36         IF TESTWO('PROC←     ') THEN
  37           /* JA */
  38           BEGIN;
  39              /* FUNKTIONSWERT=TRUE ; KENNZIFFER=9 */
  40              ANWTYP=9;
  41              ERGEBNIS='1'B1;
  42              /* PROZEDURANFANG KORREKT$ */
  43              CALL NAMES(NAME,SONDER);
  44              IF SONDER NE 6 THEN BEGIN;
  45                                 /* NEIN FEHLERMELDUNG*/
  46                                    CALL FEHLER(FEHLART);
  47                                    GOTO FIN;
  48                                 END;
  49              FI;
  50               CALL NAMES(NAME,SONDER);
  51               CODNR=1;
  52               /* ZWISCHENINFORMATION FUER PROZEDURANFANG ABLEGEN */
  53               CALL PUT4DISK (ANWTYP,CODNR,F2,F3,F4,F5,F6,NAME,Z2,Z3,Z4,Z
  54               I=1;
  55
  56               /* WIEDERHOLE BIS PROZEDURENDE ERKANNT WIRD */
  57               NEXT:;
  58               /* EINLESEN EINER QUELLKARTE */
  59               CALL EINAUS(I);
  60               /* ZUWEISUNGSAUSDRUCK */
```

```
 61              IF ZUWAUS THEN GOTO NEXT;FI;
 62              /* MARKENVEREINBARUNG$ */
 63              IF LOC          THEN GOTO NEXT;FI;
 64              /* KOMMENTAR$ */
 65              IF COMMENT      THEN GOTO NEXT;FI;
 66              /* SPRUNGANWEISUNG$ */
 67              IF SPRUNG       THEN GOTO NEXT;FI;
 68              /* ORGANISATIONSBEFEHL$ */
 69              IF ORG          THEN GOTO NEXT;FI;
 70              IF TESTWO('END¢     ') THEN GOTO END;
 71              FI;
 72              /* FEHLERMELDUNG 'PROC' ABBRECHEN*/
 73              CALL FEHLER (FEHLART);
 74              GOTO FIN;
 75              END;;
 76              /* ENDE DER SCHLEIFE */
 77
 78              CALL NAMES(PNAME,SONDER);
 79              CALL NAMES(PNAME,SONDER);
 80              FOR I TO 8 REPEAT;
 81              /* PROZEDURNAME AM ANFANG GLEICH PROZEDURNAME AM ENDE$ */
 82              IF PNAME(I) NE NAME(I) THEN
 83                /*NEIN */
 84                BEGIN;
 85                   /* FEHLERMELDUNG,ABBRUCH VON 'PROC' */
 86                   CALL FEHLER (FEHLART);
 87                   GOTO FIN;
 88                END;
 89              FI;
 90              END;
 91              /* ZWISCHENINFORMATION FUER ENDE EINER PROZEDURVEREIN-  */
 92              /* BARUNG ABLEGEN */
 93              CODNR=2;
 94              CALL PUT4DISK(ANWTYP,CODNR,F2,F3,F4,F5,F6,PNAME,Z2,Z3,Z4,Z5)
 95           END;
 96           ELSE
 97              ERGEBNIS='Ø'B1;
 98        FI;
 99   FIN;  RETURN(ERGEBNIS);
100   END;/*ENDE VON PROC*/
101
102   /*****************************************************************************
103
104   PUT4DISK:PROC (ANWTYP,F1,F2,F3,F4,F5,F6,Z1,Z2,Z3,Z4,Z5) GLOBAL
105         DCL (ANWTYP,F1,F2,F3,F4,F5,F6) FIXED;
106         DCL (Z1,Z2,Z3,Z4,Z5)(8) CHAR;
107         DCL HILFE(4Ø) CHAR INITIAL(' ');
108         DCL I FIXED;
109         FOR I TO 8 REPEAT;
110           HILFE(I   )=Z1(I);
111           HILFE(I+8)=Z2(I);
112           HILFE(I+16)=Z3(I);
113           HILFE(I+24)=Z4(I);
114           HILFE(I+32)=Z5(I);
115         END;
116         CALL COMPF(ANWTYP,F1,F2,F3,F4,F5,F6,HILFE);
117         RETURN;
118   END;/*ENDE VON PUTDISK*/
119   MODEND;
```

MODUL PR5
=========

Diese Modul enthält die Prozeduren zum Erkennen von Organisationsbefehlen, Markenvereinbarungen und von Kommentarzeilen.

ORG:

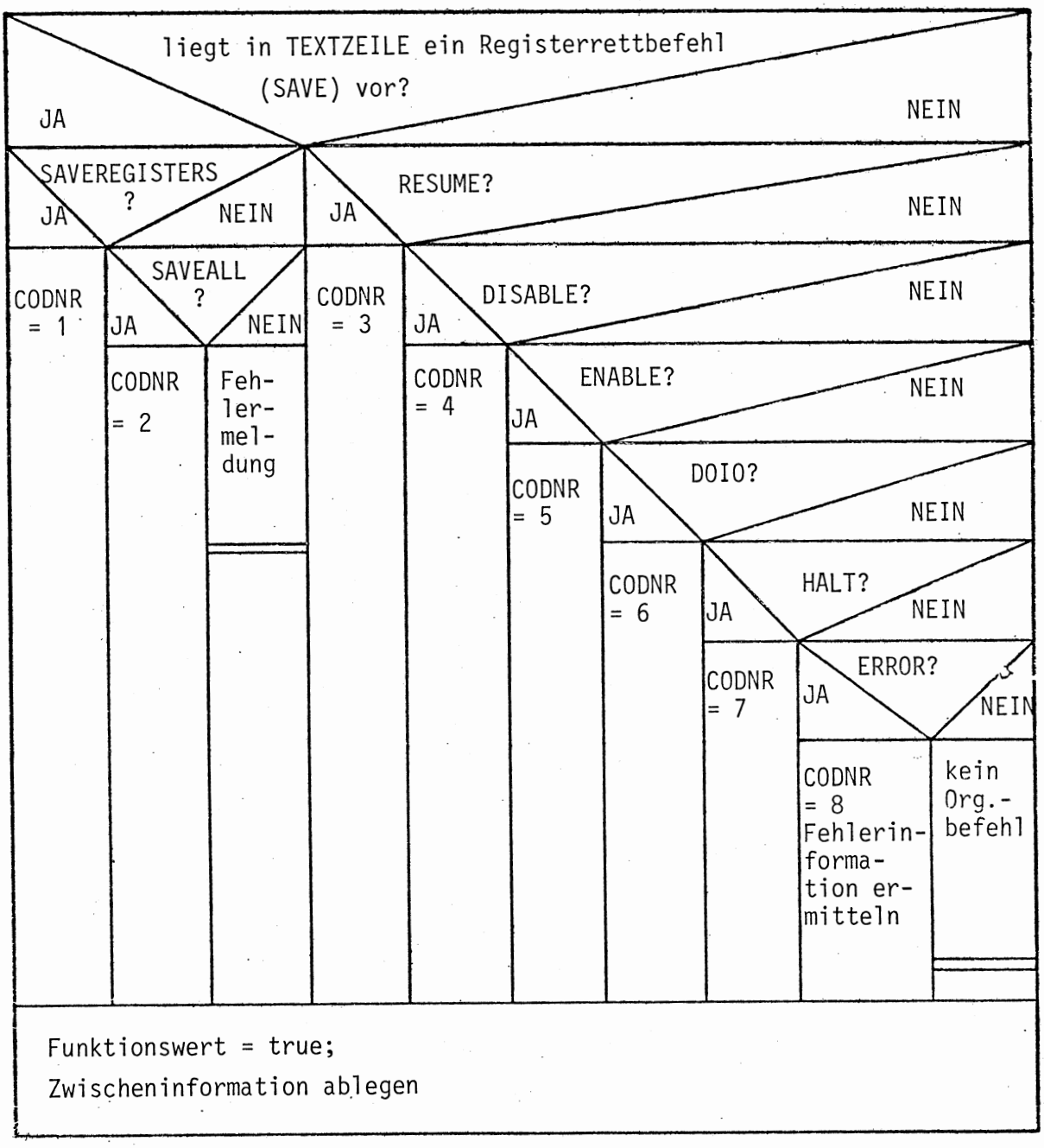
Leistung:
Diese Funktionsprozedur stellt fest, ob ein organisatorischer Befehl vorliegt und klassifiziert ihn. Es wird der entsprechende Zwischenstring erzeugt und abgelegt. Liegt ein Org.-befehl vor, ist der Funktionswert true, ansonsten false. Bei false bleibt die Stellung des Textpointers unverändert.

Parameter:
keine

Aufgerufene Prozeduren:	in Modul:
TESTWO	PR1
NAMES	PR1
FEHLER	PRØ

Aufrufende Prozeduren:	in Modul:
PRO	PR4
Task LAUF1	SPA

Fehlerausgänge:
Stellt die Prozedur einen Syntaxfehler fest, wird die Fehlernummer 4 gemeldet. Die Stellen, an denen eine Fehlermeldung gebracht wird, können dem Diagramm entnommen werden.
Anzahl der Fehlerausgänge: 1


liegt in TEXTZEILE ein Registerrettbefehl (SAVE) vor?
JA
NEIN
SAVEREGISTERS ?
JA
NEIN
RESUME?
JA
NEIN
SAVEALL ?
JA
NEIN
CODNR = 1
CODNR = 2
Fehler-meldung
CODNR = 3
DISABLE?
JA
NEIN
CODNR = 4
ENABLE?
JA
NEIN
CODNR = 5
DOIO?
JA
NEIN
CODNR = 6
HALT?
JA
NEIN
CODNR = 7
ERROR?
JA
NEIN
CODNR = 8 Fehlerin-forma-tion er-mitteln
kein Org.-befehl
Funktionswert = true;
Zwischeninformation ablegen

LOC:

Leistung:

Diese Funktionsprozedur überprüft, ob eine Markenvereinbarung vorliegt. Ist dies der Fall, wird der Funktionswert auf true gesetzt, der Zwischenstring erzeugt und abgelegt. Liegt keine Markenvereinbarung vor, wird dem Funktionswert false zugewiesen. Die Stellung des TEXTPOINTERS bleibt dann unverändert.

Parameter:

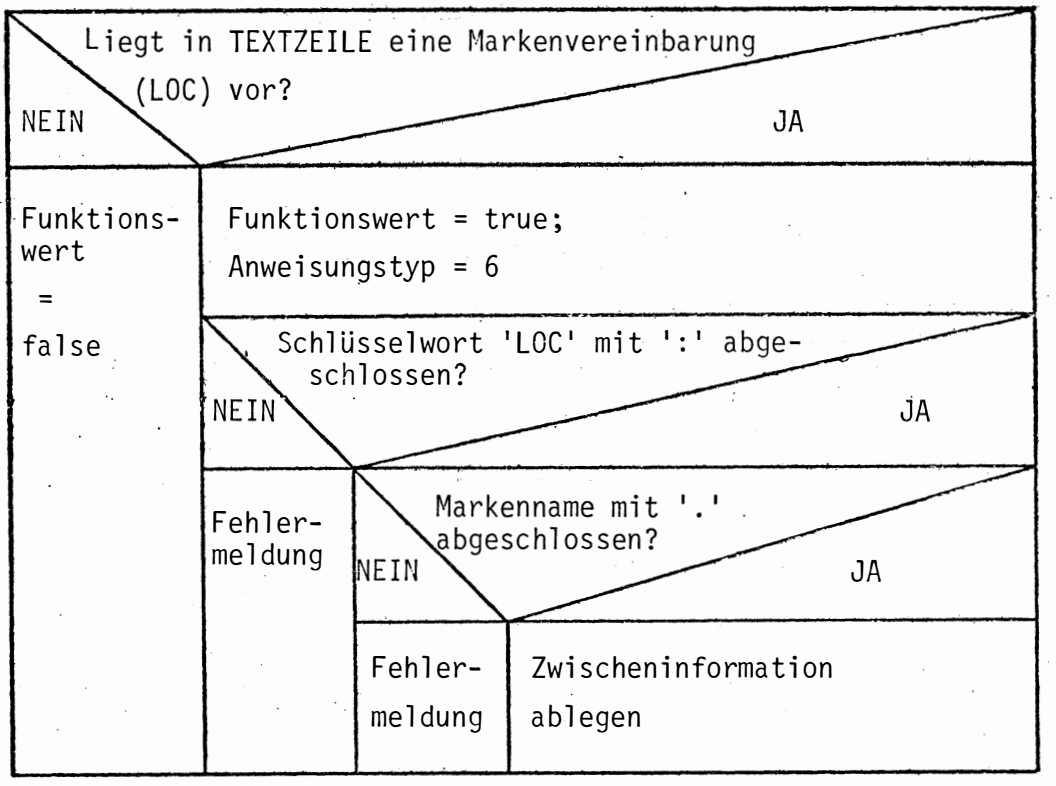
keine

Aufgerufene Prozeduren:	in Modul:
TESTWO	PR1
NAMES	PR1
FEHLER	PRØ
PUT5DISK	PR5

Aufrufende Prozeduren:	aus Modul:
PROC	PR3
Task LAUF1	SPA

Fehlerausgänge:

Bei Syntaxfehlern wird die Fehlernummer 5 meldet und die Prozedur beendet. Die Stellen, an denen eine Fehlermeldung erfolgt, sind dem Diagramm zu entnehmen.
Anzahl der Fehlerausgänge: 2


Liegt in TEXTZEILE eine Markenvereinbarung (LOC) vor?
NEIN
JA
Funktions-wert = false
Funktionswert = true;
Anweisungstyp = 6
Schlüsselwort 'LOC' mit ':' abgeschlossen?
NEIN
JA
Fehler-meldung
Markenname mit '.' abgeschlossen?
NEIN
JA
Fehler-meldung
Zwischeninformation ablegen

COMMENT:

Leistung:

Diese Funktionsprozedur stellt fest, ob eine Kommentarzeile vorliegt (beginnt mit '/'); ist dies der Fall, wird der Funktionswert auf true gesetzt, der Zwischenstring erzeugt und abgelegt. Bei false wird die Stellung des TEXTPOINTERS nicht verändert.

Parameter:

keine

Aufgerufene Prozeduren:	in Modul:
TESTWO	PR1
PUT5DISK	PR5

Aufrufende Prozeduren:	aus Modul:
PROC	PR3
Task LAUF1	SPA

Fehlerausgänge:

keine

```
 1   MODULE PR5;
 2   SYSTEM;
 3         PSEA←→DISK:;
 4         SDAU→DRUCKER:;
 5         LKEI←LESER:;
 6   PROBLEM;
 7   /* SPEZIFIKATION DER IN DIESEM MODUL VERWENDETEN GLOBALEN PROZEDUREN*/
 8   DCL TEXTPOINTER FIXED GLOBAL;
 9         DCL TEXTZEILE(8Ø) CHAR GLOBAL;
10   DCL TESTWO ENTRY (VAL CHAR(9) ) RETURNS (BIT) GLOBAL;
11   DCL NAMES ENTRY ((8) CHAR,FIXED) GLOBAL;
12   DCL NEXTCHAR ENTRY RETURNS (CHAR) GLOBAL;
13   DCL FEHLER ENTRY(FIXED) GLOBAL;
14   DCL COMPF ENTRY(FIXED,FIXED,FIXED,FIXED,FIXED,FIXED,FIXED,(4Ø)CHAR)
15         GLOBAL;
16
17   /*************************************************************************
18   /* 'ORG' ERKENNT UND ANALYSIERT DIE ORGANISATORISCHEN BEFEHLE        */
19   /*************************************************************************
20   ORG: PROC RETURNS (BIT) GLOBAL
21         DCL NAME(8) CHAR INITIAL(' ');
22         DCL ZAHL(8) CHAR INITIAL(' ');
23         DCL ORGNR FIXED;
24         DCL ERGEBNIS BIT;
25         DCL SONDER FIXED;
26         DCL CODNR FIXED;
27         DCL (F2,F3,F4,F5,F6,ANWTYP) FIXED INITIAL(Ø);
28         DCL (Z3,Z4,Z5)(8) CHAR INITIAL(' ');
29         DCL FEHLART FIXED INITIAL(4);
30         /* LIEGT EIN REGISTERRETTBEFEHL VOR$ */
31         IF TESTWO('SAVE←    ') THEN
32           /* JA */
33           BEGIN;
34              /* SAVE REGISTERS */
35              IF TESTWO('REGISTER←') THEN CODNR=1;
36              FI;
37              /* SAVE ALL REGISTERS$ */
38              IF TESTWO('ALL←     ') THEN CODNR=2;
39              FI;
40              GOTO FIN;
41           END;
42         FI;
43         /* RESUME$ */
44         IF TESTWO('RESUME←  ') THEN
45           /* JA */
46           BEGIN;
47              CODNR=3;
48              GOTO FIN;
49           END;
50         FI;
51         /*DISABLE$ */
52         IF TESTWO('DISABLE← ') THEN
53           /* JA */
54           BEGIN;
55              CODNR=4;
56              GOTO FIN;
57           END;
58         FI;
59         /* ENABLE$ */
60         IF TESTWO('ENABLE←  ') THEN
```

```
61            /* JA */
62            BEGIN;
63               CODNR=5;
64               GOTO FIN;
65            END;
66          FI;
67          /* DOIO$ */
68          IF TESTWO('DOIO¢     ') THEN
69            /* JA */
70            BEGIN;
71               CODNR=6;
72               GOTO FIN;
73            END;
74          FI;
75          /* HALT$ */
76          IF TESTWO('HALT¢     ') THEN
77            /* JA */
78            BEGIN;
79               CODNR=7;
80               GOTO FIN;
81            END;
82          FI;
83          /* ERROR$ */
84          IF TESTWO('.ERROR.¢ ') THEN
85            /* JA */
86            BEGIN;
87               CODNR=8;
88               CALL NAMES(ZAHL,SONDER);
89               CALL NAMES (NAME,SONDER);
90               GOTO FIN;
91            END;
92          FI;
93          /* KEIN ORGANISATORISCHER BEFEHL        */
94          ERGEBNIS='Ø'B1;
95          GOTO NOT;
96    FIN:  ERGEBNIS='1'B1;
97          ANWTYP=5;
98          CALL PUT5DISK (ANWTYP,CODNR,F2,F3,F4,F5,F6,NAME,ZAHL,Z3,Z4,Z5);
99          NOT:;
100         RETURN(ERGEBNIS);
101   END;/*ENDE VON ORG*/
102
103   /*****************************************************************
104   /* 'LOC' ERKENNT UND ANALYSIERT MARKENVEREINBARUNGEN              *
105   /*****************************************************************
106   LOC:  PROC RETURNS (BIT) GLOBAL
107         DCL NAME(8) CHAR;
108         DCL ERGEBNIS BIT INITIAL ('Ø'B1);
109         DCL SONDER FIXED;
110         DCL (F1,F2,F3,F4,F5,F6) FIXED INITIAL (Ø);
111         DCL (Z2,Z3,Z4,Z5)(8) CHAR INITIAL(' ');
112         DCL ANWTYP FIXED;
113         DCL FEHLART FIXED INITIAL(5);
114         /* LIEGT IN TEXTZEILE EINE MARKENVEREINBARUNG VOR$ */
115         IF TESTWO('LOC¢      '  ) THEN
116           /* JA */
117           BEGIN;
118              ERGEBNIS='1'B1;
119              ANWTYP=6;
120              /* SCHLUESSELWORT 'LOC' MIT ':' ABGESCHLOSSEN$ */
```

```
121               CALL NAMES(NAME,SONDER);
122               IF SONDER NE 6 THEN
123                 /* NEIN */
124                 BEGIN;
125                    /* FEHLERMELDUNG */
126                    CALL FEHLER(FEHLART);
127                    GOTO FIN;
128                 END;
129               FI;
130               /* MARKENNAME MIT '.' ABGESCHLOSSEN$ */
131               CALL NAMES(NAME,SONDER);
132               IF SONDER NE 5 THEN
133                 /* NEIN */
134                 BEGIN;
135                    /* FEHLERMELDUNG */
136                    CALL FEHLER(FEHLART);
137                    GOTO FIN;
138                 END;
139               FI;
140               /* ZWISCHENINFORMATION ABLEGEN */
141               CALL PUT5DISK(ANWTYP,F1,F2,F3,F4,F5,F6,NAME,Z2,Z3,Z4,Z5);
142            END;
143            ELSE
144          /* NEIN */
145            ERGEBNIS='0'B1;
146          FI;
147    FIN: RETURN(ERGEBNIS);
148    END;/*ENDE VON LOC*/
149
150    /*************************************************************************
151    /* 'COMMENT' ERKENNT UND ANALYSIERT KOMMENTARZEILEN                      */
152    /*************************************************************************
153    COMMENT: PROC RETURNS (BIT) GLOBAL
154          DCL ERGEBNIS BIT;
155          DCL (F1,F2,F3,F4,F5,F6,ANWTYP) FIXED INITIAL(0);
156          DCL (Z1,Z2,Z3,Z4,Z5)(8) CHAR INITIAL(' ');
157          /* LIEGT EINE KOMMENTARZEILE VOR$ */
158          IF TESTWO('/*       ')  THEN
159            /* JA */
160            BEGIN;
161               ERGEBNIS='1'B1;
162               ANWTYP=10;
163               CALL PUT5DISK  (ANWTYP,F1,F2,F3,F4,F5,F6,Z1,Z2,Z3,Z4,Z5);
164            END;
165          ELSE
166            /* NEIN */
167               ERGEBNIS='0'B1;
168          FI;
169          RETURN(ERGEBNIS);
170    END;/*ENDE VON COMMENT*/
171
172    /*************************************************************************
173
174    PUT5DISK:PROC (ANWTYP,F1,F2,F3,F4,F5,F6,Z1,Z2,Z3,Z4,Z5) GLOBAL
175          DCL (ANWTYP,F1,F2,F3,F4,F5,F6) FIXED;
176          DCL (Z1,Z2,Z3,Z4,Z5)(8) CHAR;
177          DCL HILFE(40) CHAR INITIAL(' ');
178          DCL I FIXED;
179          FOR I TO 8 REPEAT;
180            HILFE(I  )=Z1(I);
```

```
181             HILFE(I+8)=Z2(I);
182             HILFE(I+16)=Z3(I);
183             HILFE(I+24)=Z4(I);
184             HILFE(I+32)=Z5(I);
185           END;
186           CALL COMPF(ANWTYP,F1,F2,F3,F4,F5,F6,HILFE);
187           RETURN;
188     END;/*ENDE VON PUTDISK*/
189     MODEND;
```

MODUL PR6
=========

In diesem Modul sind die Prozeduren zum Erkennen von Wertzuweisungen, Arith. Ausdrücken und den verschiedenen Operanden (Speicher, Register, Konstante) zusammengefaßt.

ZUWAUS:

Leistung:

Mit dieser Funktionsprozedur wird festgestellt, ob eine Zuweisungsanweisung vorliegt. Ist dies der Fall, wird das letzte (am weitesten rechts stehende) Zuweisungszeichen gesucht und der TEXTPOINTER auf das dem '=' Zeichen folgende Zeichen (kann auch ein blank sein) gesetzt; ab dieser Stellung müßte nun ein arith. Ausdruck stehen. Nun wird die Prozedur zur Bearbeitung des Arith. Ausdrucks aufgerufen. Danach beginnt die Bearbeitung der Zuweisung. Dazu wird die entsprechende Prozedur (ZUWEIS) aufgerufen, die als Parameter die Stellung des TEXTPOINTERS erhält, wie er bei Eintritt in ZUWAUS war.

Liegt kein Zuweisungsausdruck vor, bleibt die Stellung des TEXTPOINTERS unverändert (wie vor Aufruf von ZUWAUS).

Parameter:

keine

Aufgerufene Prozeduren:	in Modul:
TESTWO	PR1
NAMES	PR1
ARIAUS	PR6
ZUWEIS	PR6
FEHLER	PRØ

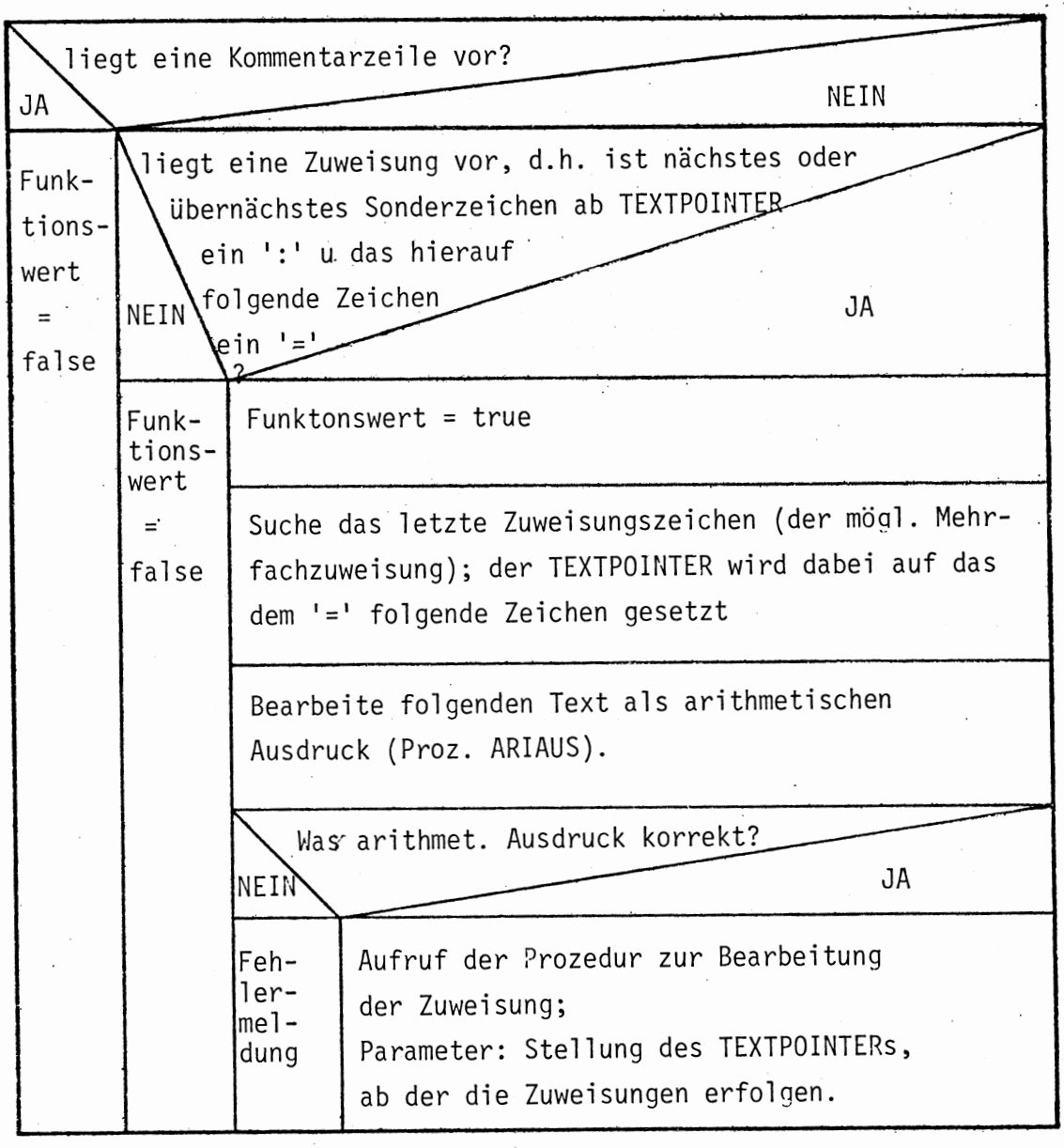
Aufrufende Prozeduren:	aus Modul:
SPRUNG	PR7
PROC	PR3
Task LAUF1	SPA

Fehlerausgänge:

Fehlernummer:7

Die Stellen, von denen aus eine Fehlermeldung erfolgt, sind dem Diagramm zu entnehmen. Im Fehlerfall wird an das Ende der Prozedur gesprungen.

Anzahl der Fehlerausgänge: 1


liegt eine Kommentarzeile vor?
JA
NEIN
Funk-
tions-
wert
=
false
liegt eine Zuweisung vor, d.h. ist nächstes oder
übernächstes Sonderzeichen ab TEXTPOINTER
ein ':' u. das hierauf
folgende Zeichen
ein '='
?
NEIN
JA
Funk-
tions-
wert
=
false
Funktonswert = true
Suche das letzte Zuweisungszeichen (der mögl. Mehr-
fachzuweisung); der TEXTPOINTER wird dabei auf das
dem '=' folgende Zeichen gesetzt
Bearbeite folgenden Text als arithmetischen
Ausdruck (Proz. ARIAUS).
Was arithmet. Ausdruck korrekt?
NEIN
JA
Feh-
ler-
mel-
dung
Aufruf der Prozedur zur Bearbeitung
der Zuweisung;
Parameter: Stellung des TEXTPOINTERs,
ab der die Zuweisungen erfolgen.

ZUWEIS:

Leistung:

Diese Prozedur erzeugt die Zwischeninformationen für die Zuweisungen, nachdem bereits der Zwischenstring für die rechte Seite vom Zuweisungsausdruck erzeugt wurde (Datenquelle). ZUWEIS erzeugt die Zwischenstrings für die Datensenken. Die Prozedur erhält durch einen Parameter die Stellung des TEXTPOINTERS, ab der mit der Bearbeitung der TEXTZEILE begonnen werden soll. ZUWEIS sucht jeweils die Datensenken, stellt deren Typ fest und speichert die erhaltenen Informationen in einem Keller. Dies wird solange wiederholt, bis das letzte Zuweisungszeichen erreicht wird.

Nun werden die Informationen aus dem Keller geholt, die entsprechend den Zwischeninformationen erzeugt und auf den Zwischenstringfile abgelegt werden. Dies wird solange wiederholt, bis der Keller leer ist. Die Stellung des TEXTPOINTERS wird nicht verändert. Der Keller kann maximal 10 Operanden aufnehmen.

Parameter:

START: Integer Größe
In diesem Parameter wird angegeben, von welcher Stelle in der TEXTZEILE die Bearbeitung beginnen soll.

Aufgerufene Prozeduren:	In Modul:
REGISTER	PR6
SPEICHER	PR6
FEHLER	PRØ
NAMES	PR1
PUT6DISK	PR6

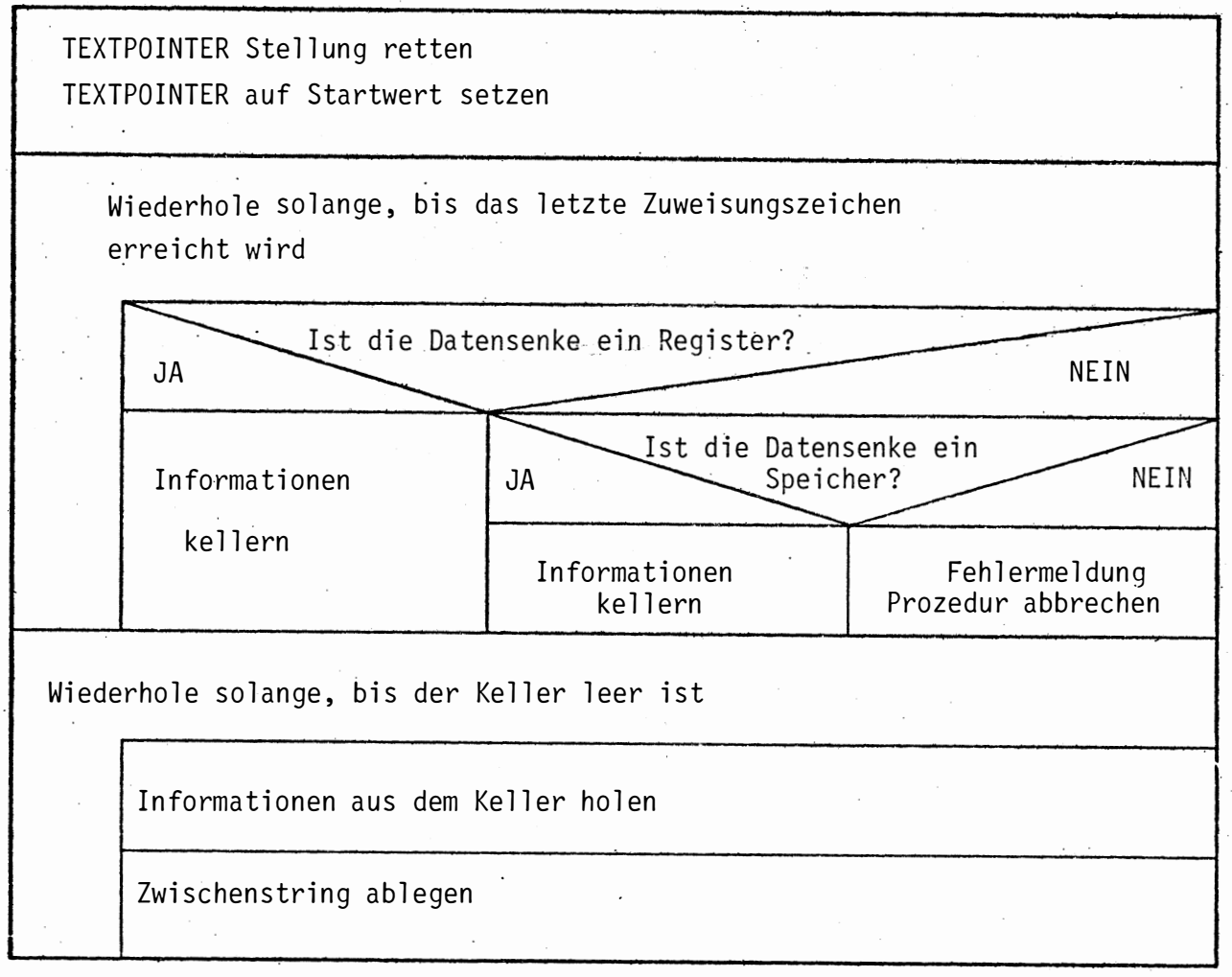
Aufrufende Prozeduren:	aus Modul:
ZUWAUS	PR6

Fehlerausgänge:

Fehlernummer: 8

Die Stellen, von denen aus eine Fehlermeldung erfolgt, sind dem Diagramm zu entnehmen. Im Falle eines Syntaxfehlers wird an das Ende der Prozedur gegangen.

Anzahl der Fehlerausgänge: 1


TEXTPOINTER Stellung retten
TEXTPOINTER auf Startwert setzen
Wiederhole solange, bis das letzte Zuweisungszeichen erreicht wird
Ist die Datensenke ein Register?
JA
NEIN
Informationen kellern
Ist die Datensenke ein Speicher?
JA
NEIN
Informationen kellern
Fehlermeldung Prozedur abbrechen
Wiederhole solange, bis der Keller leer ist
Informationen aus dem Keller holen
Zwischenstring ablegen

ARIAUS:

Leistung:

Diese Funktionsprozedur stellt fest, ob es sich um einen arith. Ausdruck mit einem oder zwei Operanden handelt. Mit Hilfe weiterer Prozeduren wird die Art der Operanden ermittelt. ARIAUS stellt auch fest, welches Operationszeichen eventuell zwei vorhandene Operanden verbindet (Kennziffer). All diese erhaltenen Informationen werden zum Zwischenstring zusammengefaßt und abgelegt. Liegt ein arith. Ausdruck vor, wird der Funktionswert auf 1 gesetzt, ansonsten auf Ø und die Stellung des Textpointers bleibt unverändert.

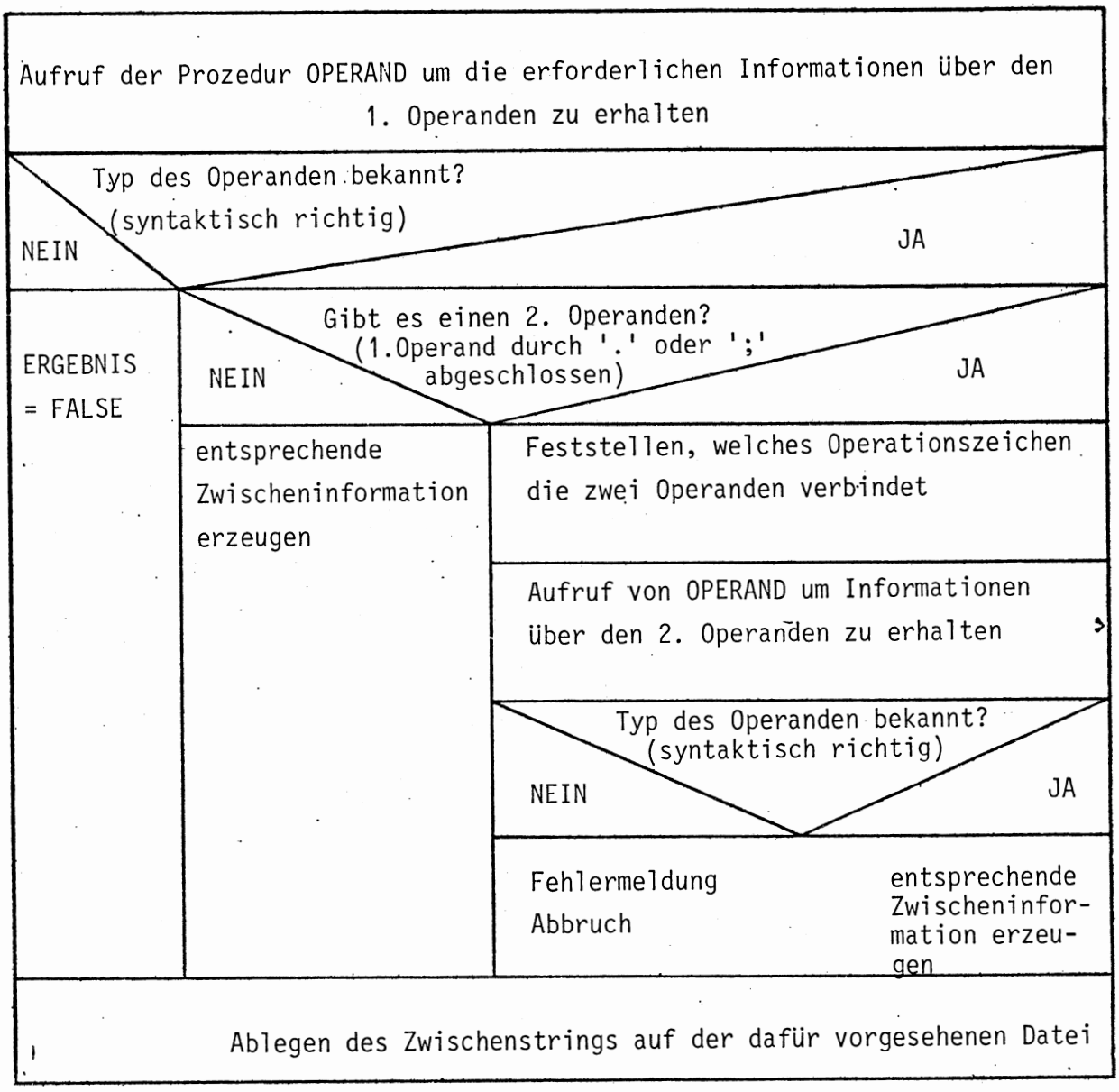
Parameter: keine

Aufgerufene Prozeduren:	in Modul:
OPERAND	PR6
NEXTCHAR	PR1
FEHLER	PRØ
PUT6DISK	PR6

Aufrufende Prozeduren:	auf Modul:
BEDING	PR7
ZUWAUS	PR6

Fehlerausgänge:

Fehlernummer:9
Die Stelle, von der aus eine Fehlermeldung erfolgen kann, ist dem Diagramm zu entnehmen.


Aufruf der Prozedur OPERAND um die erforderlichen Informationen über den 1. Operanden zu erhalten
Typ des Operanden bekannt? (syntaktisch richtig)
NEIN
JA
ERGEBNIS = FALSE
Gibt es einen 2. Operanden? (1.Operand durch '.' oder ';' abgeschlossen)
NEIN
JA
entsprechende Zwischeninformation erzeugen
Feststellen, welches Operationszeichen die zwei Operanden verbindet
Aufruf von OPERAND um Informationen über den 2. Operanden zu erhalten
Typ des Operanden bekannt? (syntaktisch richtig)
NEIN
JA
Fehlermeldung Abbruch
entsprechende Zwischeninformation erzeugen
Ablegen des Zwischenstrings auf der dafür vorgesehenen Datei

OPERAND:

Leistung:

Diese Prozedur stellt mit Hilfe weiterer Prozeduren fest, welche Typen von Operanden vorliegen (Register, Speicher, Kosntante) und bildet die entsprechenden Informationen für den Zwischenstring.

Parameter:

OPTYP: Integer Größe
In dieser Variablen gibt die Prozedur OPERAND eine Kennziffer für den Operandentyp an die aufrufende Stelle zurück. Diese Kennziffer ist eine zweistellige Zahl. Die Zehnerstelle gibt eine Unterteilung, ob eine Konstante, ein Register oder eine Speicherzelle vorliegt. Die Einerstelle gibt innerhalb dieser Gruppen noch eine genauere Aufgliederung. (Schlüssel siehe Kapitel 4.2.1.2).

HILFE: Zeichenfeld mit 16 Elementen
Diese Variable ist eine Zusammenfassung von zwei 8-elementigen Feldern (Compilerfehler). In den ersten 8 Elementen steht der Name bzw. Wert des Operanden linksbündig als Zeichenstring (nicht benötigte Felelemente sind mit blanks besetzt).

In der zweiten Hälfte steht, ebenfalls linksbündig, ein eventueller Offset (Indizierung) als Zahl in Zeichendarstellung.

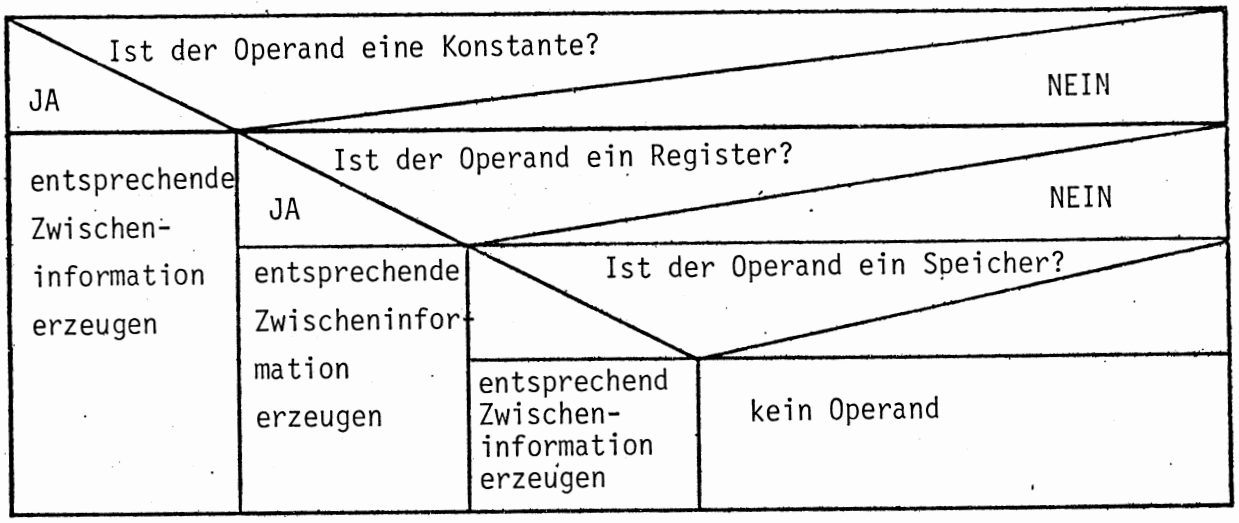
OPTYP: Integer Größe
Der Grundtyp eines Operanden wird in diesem Parameter an die aufrufende Stelle zurückgegeben.

Aufgerufene Prozeduren:	in Modul:
KONSTANTE	PR6
REGISTER	PR6
SPEICHER	PR6

Aufrufende Prozeduren:	aus Modul:
ARIAUS	PR6
VERGLEICH	PR7

Fehlerausgänge:

keine


Ist der Operand eine Konstante?
JA
NEIN
entsprechende Zwischen-information erzeugen
Ist der Operand ein Register?
JA
NEIN
entsprechende Zwischeninfor-mation erzeugen
Ist der Operand ein Speicher?
entsprechend Zwischen-information erzeugen
kein Operand

REGIST:

Leistung:

Diese Prozedur stellt fest, ob die nächsten zu verarbeitenden Zeichen (bis zum nächsten Sonderzeichen) ein Registername sind. Ist dies der Fall, wird die Kennziffer des Registers und der Name des Registers als Zeichenfolge an die aufrufende Stelle zurückgegeben.
Liegt kein Registername vor, wird die Kennziffer gleich Ø gesetzt und die Stellung des TEXTPOINTERS nicht verändert.

Parameter:

REGNR: Integer Variable
Enthält die Kennziffer des Registers
Ø: kein Registername
1: Register R1
2: Register R2
.
.
.
7: Register R7

REGNAM: 8-elementiges Zeichenfeld enthält den Namen des Registers.

OPART: Integer Größe
In diesem Parameter wird an die aufrufende Stelle der Grundtyp (Kennnr. = 1) zurückgegeben.

Aufgerufene Prozeduren:	In Modul:
NAMES	PR1

Aufrufende Prozeduren:	aus Modul:
OPERAND	PR6
SPEICHER	PR6
LABEL	PR7

Fehlerausgänge:

keine

SPEICHER:

Leistung:

Die Prozedur Speicher stellt fest, ob die vorliegende Zeichenfolge in TEXTZEILE de Produktionen für Speicher entspricht und ermittelt weitere Informationen für die Erstellung von Zwischeninformationen. Liegt kein "Speicher" vor, bleibt die Anfangsstellung des TEXTPOINTERS erhalten und eine entsprechende Kennziffer wird an die aufrufende Stelle zurückgegeben.

Parameter:

SPTYP: Integer Variable
Art des "Speichers"
Ø: Zeichenfolge in TEXTZEILE entspricht keinem Speicher
1: Adresse
2: induzierte Adresse (Offset)
3: indirekte Adressierung mit Adresse (Substitution mit Adresse)
4: indirekte Adressierung mit Register (Substitutuion mit Register)

SPNAME: Zeichenfeld mit 8 Elementen
In diesem Feld steht der Name des Speichers (Adresse, Register) als Text.

SPOFSET: Zeichenfeld mit 8 Elementen
Bei einer indizierten Adressierung steht in diesem Feld der Index als Zahl in Zeichendarstellung.

OPART: Integer Variable
In dieser Größe wird der aufrufenden Stelle der Grundtyp des "Speichers" zurückgegeben.

Aufgerufene Prozeduren:	in Modul:
PUTADR	PR2
PUTTYP	PR2
REGISTER	PR6
FEHLER	PRØ
NAMES	PR1

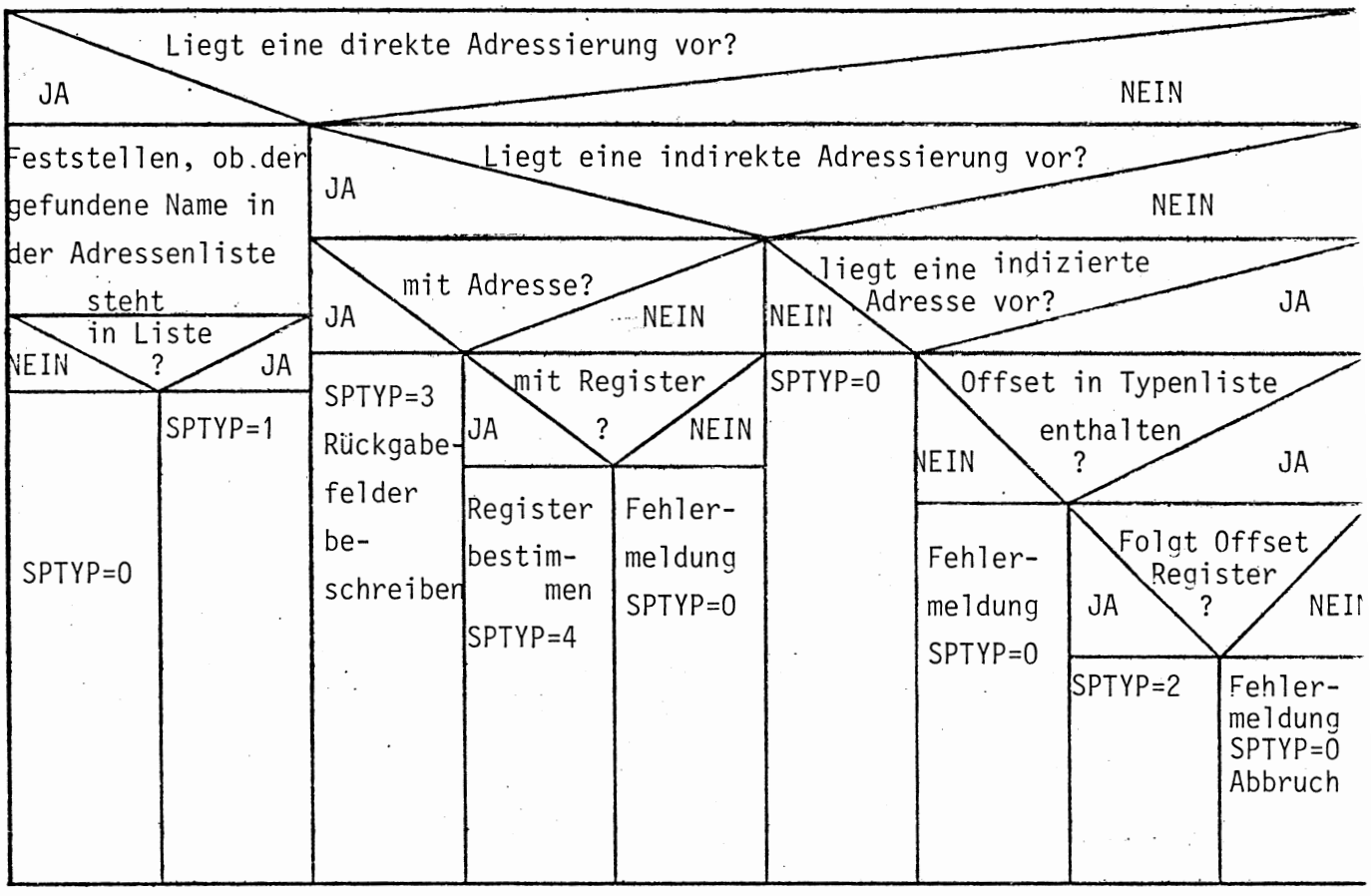
Aufrufende Prozeduren:	aus Modul:
OPERAND	PR6

Fehlerausgänge:

Fehlerkennung: 12

Die Stellen von denen aus Fehlermeldungen erfolgen können, sind dem Diagramm zu entnehmen.

Anzahl der Fehlerausgänge: 3


Liegt eine direkte Adressierung vor?
JA
NEIN
Feststellen, ob der gefundene Name in der Adressenliste steht
in Liste
NEIN
?
JA
SPTYP=0
SPTYP=1
Liegt eine indirekte Adressierung vor?
JA
NEIN
mit Adresse?
JA
NEIN
SPTYP=3 Rückgabefelder beschreiben
mit Register
JA
?
NEIN
Register bestimmen SPTYP=4
Fehlermeldung SPTYP=0
liegt eine indizierte Adresse vor?
NEIN
JA
SPTYP=0
Offset in Typenliste enthalten
NEIN
?
JA
Fehlermeldung SPTYP=0
Folgt Offset Register
JA
?
NEIN
SPTYP=2
Fehlermeldung SPTYP=0 Abbruch

KONSTANTE:

Leistung:

Die Prozedur Konstante prüft, ob die der TEXTPOINTER Stellung folgende Zeichenreihe eine Konstante ist. Als Konstante sind positive und negative Zahlen möglich, sowie Bitstrings, Adresskonstanten und Symbole für 'undefiniert'. Diese Prozedur ermittelt nähere Informationen über den Typ der Konstanten.

Parameter:

ERGEBNIS: 8-elementiges Zeichenfeld.
In diesem Feld steht die Konstante als Zeichenfolge.

KNTTY: Integer variable
In dieser Variablen steht eine Kennziffer, die den Typ der Variablen näher klassifiziert. (siehe Kapitel 4.2.1.2).

Aufgerufene Prozeduren:	in Modul:
TESTWO	PR1
NAMES	PR1
PUTTYP	PR2
PUTADR	PR2

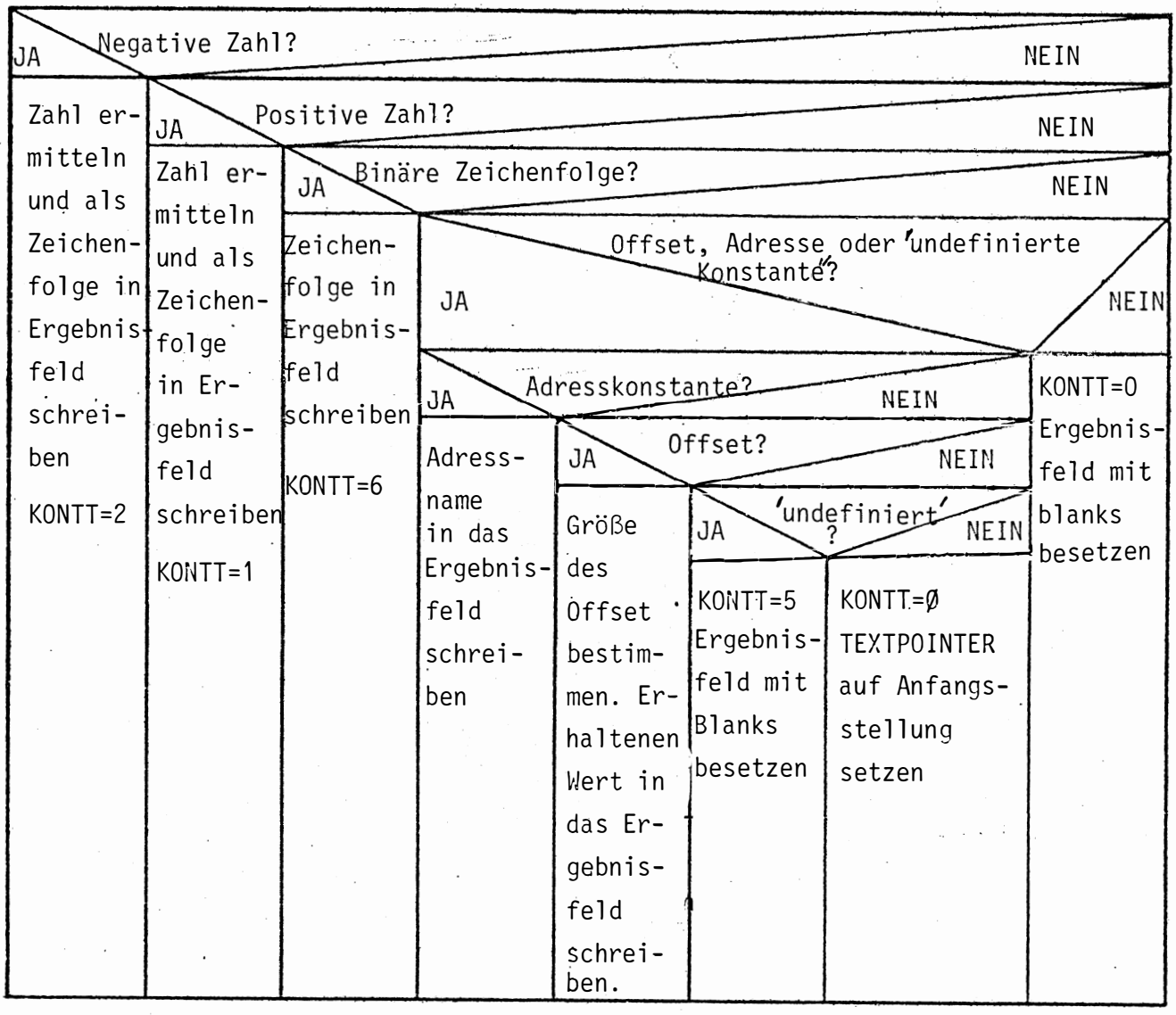
Aufrufende Prozeduren:	aus Modul:
OPERAND	PR6
SPACE	PR8

Fehlerausgänge:

Fehlerkennung: 13

Die Stellen, von denen aus eine Fehlermeldung erfolgen kann, sind dem Diagramm zu entnehmen.

Anzahl der Fehlerausgänge: 1


Negative Zahl?
JA
NEIN
Zahl ermitteln und als Zeichenfolge in Ergebnisfeld schreiben
KONTT=2
Positive Zahl?
JA
NEIN
Zahl ermitteln und als Zeichenfolge in Ergebnisfeld schreiben
KONTT=1
Binäre Zeichenfolge?
JA
NEIN
Zeichenfolge in Ergebnisfeld schreiben
KONTT=6
Offset, Adresse oder 'undefinierte Konstante'?
JA
NEIN
Adresskonstante?
JA
NEIN
Adressname in das Ergebnisfeld schreiben
Offset?
JA
NEIN
Größe des Offset bestimmen. Erhaltenen Wert in das Ergebnisfeld schreiben.
'undefiniert'?
JA
NEIN
KONTT=5 Ergebnisfeld mit Blanks besetzen
KONTT=Ø TEXTPOINTER auf Anfangsstellung setzen
KONTT=0 Ergebnisfeld mit blanks besetzen

```
   1   MODULE PR6;
   2   SYSTEM;
   3         PSEA←→DISK:;
   4         SDAU→DRUCKER:;
   5         LKEI←LESER:;
   6   PROBLEM;
   7
   8   /* SPEZIFIKATION DER IN DIESEM MODUL BENUTZTEN GLOBALEN VARIABLEN  *
   9   DCL TEXTPOINTER FIXED GLOBAL;
  10         DCL TEXTZEILE(8Ø) CHAR GLOBAL;
  11
  12   /* SPEZIFIKATION DER IN DIESEM MODUL VERWENDETEN GLOBALEN PROZEDUREN
  13   DCL TESTWO ENTRY (VAL CHAR(9) ) RETURNS (BIT) GLOBAL;
  14   DCL NEXTCHAR ENTRY RETURNS (CHAR) GLOBAL;
  15   DCL NAMES ENTRY ((8) CHAR,FIXED) GLOBAL;
  16   DCL COMPF ENTRY(FIXED,FIXED,FIXED,FIXED,FIXED,FIXED,FIXED,(4Ø)CHAR)
  17         GLOBAL;
  18   DCL PUTADR ENTRY ((8) CHAR,FIXED) GLOBAL;
  19   DCL PUTTYP ENTRY ((8) CHAR,FIXED) GLOBAL;
  20   DCL FEHLER ENTRY (FIXED) GLOBAL;
  21
  22   /*************************************************************************
  23   /* 'ZUWAUS' STELLT FEST OB EIN ZUWEISUNGSAUSDRUCK VORLIEGT */
  24   /*************************************************************************
  25   ZUWAUS: PROC RETURNS (BIT) GLOBAL
  26         DCL ERGEBNIS BIT;
  27         DCL NAME(8) CHAR;
  28         DCL MERK FIXED;
  29         DCL START FIXED;
  30         DCL TRENNZ(3) CHAR INITIAL('←');
  31         DCL SONDER FIXED;
  32         DCL OPART FIXED INITIAL(Ø);
  33         DCL FEHLART FIXED INITIAL(7);
  34         START=TEXTPOINTER;
  35         /* LIEGT EINE KOMMENTARZEILE VOR$ */
  36         IF TESTWO('/←        ') THEN
  37               /* JA */
  38           BEGIN;
  39               ERGEBNIS='Ø'B1;
  40               TEXTPOINTER=START;
  41               GOTO FIN;
  42           END;
  43         FI;
  44         /* LIEGT EINE ZUWEISUNG VOR , D.H. IST NAECHSTES ODER */
  45         */ UEBERNAECHSTES SONDERZEICHEN AB DER TEXTPOINTERSTELLUNG */
  46         /* EIN ':'  */
  47         CALL NAMES(NAME,SONDER);
  48         IF SONDER NE 6 THEN
  49           BEGIN;
  50               MERK=TEXTPO;
  51               CALL NAMES(NAME,SONDER);
  52               IF SONDER NE 6 THEN
  53                 BEGIN;
  54                     ERGEBNIS='Ø'B1;
  55                     TEXTPO=START;
  56                     GOTO FIN;
  57                 END;
  58               FI;
  59           END;
  60         FI;
```

```
61          /*    UND DAS HIERAUF FOLGENDE ZEICHEN EIN '=' $ */
62          IF SONDER EQ 6 AND TEXTZEI(TEXTPOIN) EQ '=' THEN
63            BEGIN;
64               ERGEBNIS='1'B1;
65               /* SUCHE DAS LETZTE ZUWEISUNGSZEICHEN (DER MOEGL.       */
66               /* MEHRFACHZUWEISUNGEN); DER TEXTPOINTER WIRD DABEI  AUF */
67               /* DAS DEM LETZTEN '=' FOLGENDE ZEICHEN GESETZT */
68    NEW:       TEXTPOINTER=TEXTPOINTER+1;
69               MERK=TEXTPOINTER;
70               CALL NAMES(NAME,SONDER);
71               IF SONDER EQ 6 THEN GOTO NEW;FI;
72               CALL NAMES(NAME,SONDER);
73               IF SONDER EQ 6 THEN GOTO NEW;FI;
74               /* BEARBEITE DIE FOLGENDEN ZEICHEN IN TEXTZEILE ALS */
75               /*ARITHMETISCHEN AUSDRUCK (AUFRUF VON 'ARIAUS'). */
76               TEXTPOINTER=MERK;
77               /*WAR DER ARITH. AUSDRUCK KORREKT$ */
78               IF ARIAUS EQ 'Ø'B1 THEN
79                 /* NEIN */
80                 BEGIN;
81                    /* FEHLERMELDUNG,ABBRUCH VON 'ZUWAUS' */
82                    CALL FEHLER(FEHLART);
83                    GOTO FIN;
84                 END;
85               FI;
86               /* AUFRUF DER PROZEDUR 'ZUWEIS ZUM BEARBEITEN DER     */
87               /* ZUWEISUNGEN */
88               CALL ZUWEIS (START);
89            END;
90          ELSE
91          BEGIN;
92            /* FUNKTIONSWERT=FALSE */
93            TEXTPOI=START;
94            ERGEBNIS='Ø'B1;
95          END;
96          FI;
97          FIN:;
98          RETURN(ERGEBNIS);
99    END;/*ENDE VON ZUWEIS*/
100
101   /*********************************************************************/
102   /* 'ZUWEIS' ANALYSIERT DIE LINKE SEITE EINES ZUWEISUNGSAUSDRUCKS   */
103   /*********************************************************************/
104   ZUWEIS: PROC (START)  GLOBAL
105         DCL START FIXED;
106         DCL (SSTACK,OSTACK)(1Ø,8) CHAR;
107         DCL TSTACK(1Ø) FIXED INITIAL (Ø);
108         DCL ARTSTACK(1Ø) FIXED INITIAL (Ø);
109         DCL STACKPOINTER FIXED INITIAL (Ø);
110         DCL RETTEXT FIXED;
111         DCL MERK FIXED;
112         DCL TRENNZ(3) CHAR;
113         DCL SONDER FIXED;
114         DCL NAME(8) CHAR;
115         DCL ROPTYP FIXED;
116         DCL TYP FIXED,SENKE(8) CHAR;
117         DCL OFSET(8) CHAR INITIAL(' ');
118         DCL SENKOF(8) CHAR INITIAL(' ');
119         DCL I FIXED;
120         DCL Z3(8) CHAR INITIAL(' ');
```

```
121          DCL (Z4,Z5)(8) CHAR INITIAL(' ');
122          DCL (F2,F3,F4,F5,F6) FIXED INITIAL(Ø);
123          DCL ANWTYP FIXED;
124          DCL FEHLART FIXED INITIAL(8);
125          DCL OPART FIXED INITIAL (Ø);
126          ANWTYP=3;
127          /* STELLUNG DES 'TEXTPOINTER' RETTEN  */
128          /* 'TEXTPOINTER AUF STARTWERT SETZEN */
129          RETTEXT=TEXTPOINTER;
130          TEXTPOINTER=START;
131
132          /* WIEDERHOLE SOLANGE,BIS DAS LETZTE ZUWEISUNGSZEICHEN        *
133          /* ERREICHT WIRD */
134     NEW: MERK=TEXTPOINTER;
135          CALL NAMES(NAME,SONDER);
136          IF SONDER NE 6 THEN
137               CALL NAMES(NAME,SONDER);
138          FI;
139          IF SONDER EQ 6 THEN
140            BEGIN;
141               TEXTPOINTER=MERK;
142               /* IST DIE DATENSENKE EIN REGISTER$ */
143               CALL REGISTER (TYP,OPART,SENKE);
144               IF TYP NE Ø THEN
145                 BEGIN;
146                    ROPTYP=2Ø+TYP;
147                    /* INFORMATIONEN KELLERN */
148                    GOTO PUSH;
149                 END;
150               FI;
151               TEXTPOINTER=MERK;
152               /* IST DIE DATENSENKE EIN SPEICHER */
153               CALL SPEICHER (TYP,OPART,SENKE,SENKOF);
154               IF TYP NE Ø THEN
155                 BEGIN;
156                    ROPTYP=3Ø+TYP;
157                    /* INFORMATIONEN KELLERN */
158                    GOTO PUSH;
159                 END;
160               FI;
161               /* FEHLERMELDUNG,'ZUWEIS' ABBRECHEN */
162               CALL FEHLER(FEHLART);
163              GOTO ENDE;
164
165            /* PROGRAMMSTUECK ZUM KELLERN DER INFORMATIONEN */
166     PUSH:     STACKPOINTER=STACKPOINTER+1;
167               TSTACK(STACKPOINTER)=ROPTYP;
168               ARTSTACK(STACKP)=OPART;
169               FOR I TO 8 REPEAT;
170                 SSTACK(STACKP,I)=SENKE(I);
171                 OSTACK(STACKP,I)=SENKOF(I);
172               END;
173               TEXTPOINTER=TEXTPOINTER+2;
174               GOTO NEW;
175          /* ENDE DER SCHLEIFE */
176
177            END;
178          FI;
179
180          /* WIEDERHOLE SOLANGE BIS DER KELLER LEER IST */
```

```
181   NEXT :IF STACKPOIN      EQ Ø THEN GOTO ENDE;FI;
182         /* INFORMATIONEN AUS DEM KELLER HOLEN */
183         OPART=ARTSTACK(STACKPO);
184         TYP=TSTACK(STACKP);
185         FOR I TO 8 REPEAT;
186           BEGIN;
187              OFSET(I)=OSTACK(STACKP,I);
188              SENKE(I)=SSTACK(STACKP,I);
189           END;
190         END;
191         /* ZWISCHENSTRING ABLEGEN */
192         CALL PUT6DISK(ANWTYP,TYP,F2,F3,F4,F5,OPART,SENKE,OFSET,Z3,Z4,Z5);
193         STACKPOIN=STACKPOINTER-1;
194         GOTO NEXT;
195   ENDE: TEXTPOINTER=RETTEXT;
196         FIN:;
197         RETURN;
198   END;/*ENDE VON ZUWEIS*/
199
200   /*************************************************************************/
201   /* 'ARIAUS' ERKENNT UND ANALYSIERT EINEN ARITH. AUSDRUCK */
202   /*************************************************************************/
203   ARIAUS:PROC RETURNS(BIT) GLOBAL
204         DCL (NAME1,OFSET1,NAME2,OFSET2)(8) CHAR INITIAL(' ');
205         DCL (TYP1,TYP2,CODNR,OPZEICH) FIXED INITIAL(Ø);
206         DCL NEXT CHAR;
207         DCL I FIXED;
208         DCL HILFE(16) CHAR INITIAL(' ');
209         DCL ANWTYP FIXED;
210         DCL F5 FIXED INITIAL(Ø);
211         DCL RETTEXT FIXED;
212         DCL ERGEBNIS BIT;
213         DCL Z5(8) CHAR INITIAL(' ');
214         DCL (OPART1,OPART2) FIXED INITIAL(Ø);
215         DCL FEHLART FIXED INITIAL (9);
216         RETTEXT=TEXTPOINTER;
217         ANWTYP=8;
218         /* AUFRUF VON 'OPERAND' UM DIE ERFORDERLICHEN INFORMATIONEN */
219         /* UEBER DEN 1. OPERANDEN ZU ERHALTEN */
220         CALL OPERAND(TYP1,OPART1,HILFE);
221         FOR I TO 8 REPEAT;
222              NAME1(I)=HILFE(I);
223              OFSET1(I)=HILFE(I+8);
224         END;
225         /*TYP DES OPERANDEN BEKANNT (SYNT. RICHTIG) */
226         IF TYP1 EQ Ø THEN
227           /* NEIN */
228           BEGIN;
229              ERGEBNIS='Ø'B1;
230              TEXTPOINTER=RETTEXT;
231              GOTO FIN;
232           END;
233         FI;
234         ERGEBNIS='1'B1;
235         RETTEXT=TEXTPOINTER;
236         NEXT=NEXTCHAR;
237         I=TEXTPOINTER;
238         TEXTPOINTER=RETTEXT;
239         /* GIBT ES EINEN 2. OPERANDEN */
240        IF NEXT EQ ';' OR NEXT EQ '.' THEN
```

```
241           /* NEIN */
242           BEGIN;
243              /* ENTSPRECHENDE ZWISCHENINFORMATION ERZEUGEN */
244              CODNR=1;
245              OPZEICH=0;
246           END;
247         ELSE
248           /* JA */
249           BEGIN;
250              /* FESTSTELLEN WELCHES OPERATIONSZEICHEN DIE BEIDEN   */
251              /* OPERANDEN VERBINDET */
252              IF NEXT EQ '+' THEN OPZEICH=1;FI;
253              IF NEXT EQ '-' THEN OPZEICH=2;FI;
254              IF NEXT EQ '*' THEN OPZEICH=3;FI;
255              TEXTPOINTER=I;
256              /* AUFRUF VON 'OPERAND' UM INFORMATIONEN UEBER DEN   */
257              /* 2. OPERANDEN */
258              CALL OPERAND(TYP2,OPART2,HILFE);
259               /*TYP DES OPERANDEN BEKANNT (SYNT. RICHTIG) */
260              IF TYP2 EQ 0 THEN
261                /* NEIN */
262                BEGIN;
263                   /* FEHLERMELDUNG,ABBRUCH VON 'ARIAUS' */
264                   CALL FEHLER(FEHLART);
265                   GOTO FIN;
266                END;
267              FI;
268              /* ENTSPRECHENDE ZWISCHENINFORMATION ERZEUGEN */
269              FOR I TO 8 REPEAT;
270                   NAME2(I)=HILFE(I);
271                   OFSET2(I)=HILFE(I+8);
272              END;
273              CODNR=2;
274           END;
275         FI;
276         OPART1=OPART1*10+OPART2;
277         /* ABLEGEN DES ZWISCHENSTRINGS */
278         CALL PUT6DISK(ANWTYP,CODNR,TYP1,CPZEICH,TYP2,F5,OPART1,NAME1,
279              OFSET1,NAME2,OFSET2,25);
280   FIN:;
281         RETURN(ERGEBNIS);
282   END;/*ENDE VON ARIAUS*/
283
284
285   /*************************************************************************
286   /* 'OPERAND' ERKENNT UND ANALYSIERT OPERANDEN                            *
287   /*************************************************************************
288   OPERAND:PROC (OPTYP,OPART,HILFE)
289   GLOBAL
290         DCL OPTYP FIXED;
291         DCL OPART FIXED;
292         DCL HILFE(16) CHAR;
293         DCL (OPNAME,OPOFSET)(8) CHAR;
294         DCL TYP FIXED,NAME(8) CHAR;
295         DCL I FIXED;
296         /* IST DER OPERAND EINE KONSTANTE$ */
297         CALL KONSTANTE(OPNAME,TYP,OPART);
298         IF TYP NE 0 THEN
299           BEGIN;
300              OPTYP=TYP+10;
```

```
301               FOR I TO 8 REPEAT
302                    HILFE(I)=OPNAME(I);
303                    HILFE(I+8)=' ';
304               END;
305               GOTO FIN;
306             END;
307           FI;
308           /* IST DER OPERAND EIN REGISTER$ */
309           CALL REGIST(TYP,OPART,OPNAME);
310           IF TYP NE Ø THEN
311             BEGIN;
312               OPTYP=TYP+2Ø;
313               FOR I TO 8 REPEAT
314                    HILFE(I)=OPNAME(I);
315                    HILFE(I+8)=' ';
316               END;
317               GOTO FIN;
318             END;
319           FI;
320           /* IST DER OPERAND EIN SPEICHER$ */
321           CALL SPEICHER(TYP,OPART,OPNAME,OPOFSET);
322           IF TYP NE Ø THEN
323             BEGIN;
324               OPTYP=TYP+3Ø;
325               FOR I TO 8 REPEAT;
326                    HILFE(I)=OPNAME(I);
327                    HILFE(I+8)=OPOFSET(I);
328               END;
329               GOTO FIN;
330             END;
331           FI;
332           /* OPERAND IST SYNTAKTISCH FALSCH ODER ES LIEGT KEIN  */
333           /* OPERAND VOR */
334           OPTYP=Ø;
335     FIN:  RETURN;
336     END;/*ENDE VON OPERAND*/
337
338
339     /*************************************************************************/
340     /* 'REGIST' ERKENNT OB EIN REGISTER VORLIEGT UND ERZEUGT DIE ENT-  */
341     /* SPRECHENDE ZWISCHENINFORMATION                                  */
342     /*************************************************************************/
343     REGIST:PROC (REGNR,OPART,REGNAM)
344     GLOBAL
345           DCL (REGNR,OPART) FIXED ;
346           DCL REGNAM(8) CHAR;
347           DCL REG(7) VAL CHAR(2) IDENTICAL('R1','R2','R3','R4','R5','R6',
348                                                          'R7');
349           DCL (RETTEXT,SONDER,I) FIXED;
350           RETTEXT=TEXTPOINTER;
351           OPART=1;
352           /* REGISTERNAMEN FESTSTELLEN */
353           CALL NAMES (REGNAM,SONDER);
354           TEXTPOINTER=TEXTPOINTER-1;
355           /* REGISTERNUMMER FESTSTELLEN */
356           FOR I TO 7 REPEAT;
357             IF ((1 CHAR REG(I) EQ REGNAM(1)) AND (2 CHAR REG(I) EQ REGNAM(
358                                                          )) THEN
359               BEGIN;
360                    REGNR=I;
```

```
361                     GOTO GEFUND;
362             END;
363           FI;
364         END;
365         /* ES LIEGT KEIN REGISTER VOR */
366         REGNR=Ø;
367         TEXTPOINTER=RETTEXT;
368   GEFUND:RETURN;
369   END;/*ENDE VON REGIST*/
370
371   /*************************************************************************
372   /* 'SPEICHER' ERKENNT UND ANALYSIERT DIE VERSCHIEDENEN SPEICHER-    *
373   /* PLATZTYPEN (ADRESSIERUNGSARTEN)                                  *
374   /*************************************************************************
375   SPEICHER:PROC (SPTYP,OPART,SPNAME,SPOFSET)
376   GLOBAL
377         DCL SPTYP FIXED;
378         DCL OPART FIXED;
379         DCL (SPNAME ,SPOFSET)(8) CHAR;
380         DCL (RETTEXT,MERKE,SONDER) FIXED;
381         DCL NAME(8) CHAR(1),TRENNZ(3) CHAR;
382         DCL FEHLART FIXED INITIAL (12);
383         DCL I FIXED;
384         FOR I TO 8 REPEAT;
385           SPOFSET(I)=' ';
386         END;
387         RETTEXT=TEXTPOINTER;
388         /* LIEGT EINE DIREKTE ADRESSIERUNG VOR$ */
389         CALL NAMES(NAME,SONDER);
390        IF SONDER LE 6 THEN
391           BEGIN;
392              /* STEHT DER GEFUNDENE NAME IN DER ADRESSENLISTE$ */
393              CALL PUTADR(NAME,OPART);
394              IF NAME(1) EQ ' ' THEN
395                    /* NEIN */
396                BEGIN;
397                   SPTYP=Ø;
398                   TEXTPOINTER=RETTEXT;
399                END;
400               ELSE
401                    /* JA */
402                BEGIN;
403                   SPTYP=1;
404                   FOR I TO 8 REPEAT;
405                        SPNAME(I)=NAME(I);
406                   END;
407                END;
408              FI;
409              TEXTPOINTER=TEXTPOINTER-1;
410              GOTO FIN;
411           END;
412         FI;
413         /* LIEGT EINE INDIEREKTE ADRESSIERUNG VOR$ */
414         IF ((NAME(1) EQ '*') AND (NAME(2) EQ 'Ø')) THEN
415           BEGIN;
416              MERKE=TEXTPOINTER;
417              CALL NAMES(NAME,SONDER);
418              TEXTPOINTER=TEXTPOINTER-1;
419              /* MIT ADRESSE$ */
420              CALL PUTADR(NAME,OPART);
```

```
421               IF NAME(1) NE ' ' THEN
422                 /* JA */
423                 BEGIN;
424                    SPTYP=3;
425                    FOR I TO 8 REPEAT;
426                         SPNAME(I)=NAME(I);
427                    END;
428                 END;
429                ELSE
430                 /* NEIN */
431                 BEGIN;
432                    TEXTPOINTER=MERKE;
433                    /* MIT REGISTERN */
434                    CALL REGISTER(SPTYP,OPART,SPNAME);
435                    IF SPTYP EQ Ø THEN
436                      /* NEIN */
437                      BEGIN;
438                         /* FEHLERMELDUNG,ABBRUCH VON SPEICHER */
439                         CALL FEHLER(FEHLART);
440                         GOTO FIN;
441                      END;
442                    FI;
443                    SPTYP=4;
444                 END;
445               FI;
446               GOTO FIN;
447             END;
448           FI;
449           /* LIEGT EINE INDIZIERTE ADRESSIERUNG VOR? */
450           IF (SONDER EQ 8) AND (NAME(1) NE '(') THEN
451             BEGIN;
452               /* IST DER OFFSET IN DER TYPENLISTE ENTHALTEN? */
453               CALL PUTTYP(NAME,OPART);
454               IF NAME(1) EQ ' ' THEN BEGIN;
455                                        /* NEIN */
456                                        /* FEHLERMELDUNG,ABBRUCH VON SP. */
457                                        CALL FEHLER(FEHLART);
458                                       GOTO FIN;
459                                      END;
460               FI;
461               SPOFSET(1)=NAME(7);
462               SPOFSET(2)=NAME(8);
463               FOR I FROM 3 TO 8 REPEAT;
464                    SPOFSET(I)=' ';
465               END;
466               /* FOLGT DEM OFSET EIN REGISTER? */
467               CALL REGIST(SPTYP,I    ,SPNAME);
468               IF SPTYP EQ Ø THEN BEGIN;
469                                        /* NEIN */
470                                        /* FEHLERMELDUNG,ABBRUCH VON SP. */
471                                        CALL FEHLER(FEHLART);
472                                        GOTO FIN;
473                                      END;
474               FI;
475               SPTYP=2;
476             GOTO FIN;
477             END;
478           FI;
479           SPTYP=Ø;
480   FIN:    RETURN;
```

```
481    END;/*ENDE VON SPEICHER*/
482    /*************************************************************************
483    /* 'KONSTANTE' ERKENNT UND ANALYSIERT DIE VERSCH. KONSTANTENTYPEN
484    /*************************************************************************
485    KONSTANTE:PROC(ERGEBNIS,KONTTY,OPART)
486    GLOBAL
487           DCL ERGEBNIS(8) CHAR;
488           DCL KONTTY FIXED;
489           DCL OPART FIXED;
490           DCL FEHLART FIXED INITIAL(13);
491           DCL HILFE(8) CHAR INITIAL(' ');
492           DCL SONDER FIXED;
493           DCL (I,J) FIXED;
494           DCL RETTEXT FIXED;
495           RETTEXT=TEXTPOINTER;
496    /* DIE NAECHSTEN BEIDEN IF ANWEISUNGEN DUERFEN NICHT VERTAUSCHT WERDE
497           /* NEGATIVE ZAHL$ */
498           IF TESTWO('''KM¢      ') THEN BEGIN;
499                                        /* JA */
500                                        ERGEBNIS(1)='-';
501                                        /* ZAHL ERMITTELN UND ALS        *
502                                        /*ZEICHENFOLGE IN DAS ERGEBNIS- *
503                                        /* FELD SCHREIBEN                *
504                                        FOR I FROM 2 TO 7 REPEAT;
505                                          ERGEBNIS(I)=NEXTCHAR;
506                                          IF ERGEBNIS(I) EQ '''' THEN
507                                            BEGIN;
508                                              FOR J FROM I TO 8 REPEAT;
509                                                ERGEBNIS(J)=' ';
510                                              END;
511                                              KONTTY=2;
512                                              OPART=3;
513                                              GOTO FIN;
514                                            END;
515                                          FI;
516                                        END;
517                                         /* ZAHL HAT MEHR ALS 8 STELLEN *
518                                         CALL FEHLER(FEHLART);
519                                        END;
520           FI;
521           /* POSITIVE ZAHL$ */
522           IF TESTWO('''K¢       ') THEN BEGIN;
523                                        /* ZAHL ERMITTELN UND ALS        *
524                                        /*ZEICHENFOLGE IN DAS ERGEBNIS- *
525                                        /* FELD SCHREIBEN                *
526                                        FOR I TO 8 REPEAT
527                                          ERGEBNIS(I)=NEXTCHAR;
528                                          IF ERGEBNIS(I) EQ '''' THEN
529                                             BEGIN;
530                                               FOR J FROM I TO 8 REPEAT;
531                                                  ERGEBNIS(J)=' ';
532                                               END;
533                                               KONTTY=1;
534                                               OPART=3;
535                                               GOTO FIN;
536                                             END;
537                                          FI;
538                                        END;
539                                        END;
540           FI;
```

```
541         /* BINAERZAHL$ */
542         IF TESTWO('''B<       ') THEN
543           BEGIN;
544                                       /* ZAHL ERMITTELN UND ALS       */
545                                       /*ZEICHENFOLGE IN DAS ERGEBNIS- */
546                                       /* FELD SCHREIBEN               */
547               FOR I TO 8 REPEAT;
548                 ERGEBNIS(I)=NEXTCHAR;
549                 IF ERGEBNIS(I) EQ ''' THEN
550                     BEGIN;
551                       FOR J FROM I TO 8 REPEAT;
552                          ERGEBNIS(J)=' ';
553                       END;
554                       KONTTYP=6;
555                       OPART=4;
556                       GOTO FIN;
557                     END;
558                 FI;
559               END;
560           END;
561         FI;
562         /* OFSET- ODER ADRESSKONSTANTE BZW. 'UNDEFINIERTE KONSTANTE'$ */
563         IF ((TESTWO('A<       ') EQ '1'B1) AND (TEXTZEI(TEXTPOI) NE 'Ø'))
564         THEN
565           BEGIN;
566               CALL NAMES (ERGEBNIS,SONDER);
567               FOR I TO 8 REPEAT;
568                    HILFE(I)=ERGEBNIS(I);
569               END;
570               TEXTPOINTER=TEXTPOINTER-1;
571               /* OFSET$ */
572                         /* GROESSE DES OFSET FESTSTELLEN */
573               CALL PUTTYP(ERGEBNIS,OPART);
574               IF ERGEBNIS(1) NE ' ' THEN
575                    /* JA */
576                    BEGIN;
577                         KONTTY=3;
578                         /* ERHALTENEN WERT IN DAR ERGEBNISFELD SCHREIBEN *
579                         ERGEBNIS(1)=ERGEBNIS(7);
580                         ERGEBNIS(2)=ERGEBNIS(8);
581                         FOR J FROM 3 TO 8 REPEAT;
582                              ERGEBNIS(J)=' ';
583                         END;
584                         GOTO FIN;
585                    END;
586               FI;
587               /* ADRESSKONSTANTE$ */
588               /* ADRESSNAME IN DAS ERGEBNISFELD SCHREIBEN */
589               FOR I TO 8 REPEAT;
590                    ERGEBNIS(I)=HILFE(I);
591               END;
592               CALL PUTADR(ERGEBNIS,OPART);
593               IF ERGEBNIS(1) NE ' ' THEN BEGIN;
594                                       /* JA */
595                                          KONTTY=4;
596                                          GOTO FIN;
597                                       END;
598               FI;
599               TEXTPOI=RETTEXT;
600               /* 'UNDEFINIERT'$ */
```

```
601                IF TESTWO('ANIL4     ') THEN
602                  BEGIN;
603                     KONTTYP=5;
604                       OPART=2;
605                     FOR I TO 8 REPEAT;
606                       ERGEBNIS(I)=' ';
607                     END;
608                       ERGEBNIS(1)='0';
609                     GOTO FIN;
610                  END;
611                FI;
612              END;
613           FI;
614           /* KEINE KONSTANTE ODER SYNT. FALSCH */
615           TEXTPOINTER=RETTEXT;
616           KONTTY=0;
617           FOR I TO 8 REPEAT;
618             ERGEBNIS(I)=' ';
619           END;
620     FIN:  RETURN;
621     END;
622
623     /*************************************************************************
624
625     PUTtoDISK:PROC (ANWTYP,F1,F2,F3,F4,F5,F6,Z1,Z2,Z3,Z4,Z5) GLOBAL
626           DCL (ANWTYP,F1,F2,F3,F4,F5,F6) FIXED;
627           DCL (Z1,Z2,Z3,Z4,Z5)(8) CHAR;
628           DCL I FIXED;
629           DCL HILFE(40) CHAR;
630           FOR 1 TO 8 REPEAT;
631             HILFE(I   )=Z1(I);
632             HILFE(I+8)=Z2(I);
633             HILFE(I+16)=Z3(I);
634             HILFE(I+24)=Z4(I);
635             HILFE(I+32)=Z5(I);
636           END;
637           CALL COMPF(ANWTYP,F1,F2,F3,F4,F5,F6,HILFE);
638           RETURN;
639     END;/*ENDE VON PUTDISK*/
640     MODEND;
```

MODUL PR 7

In diesem Modul befinden sich die Prozeduren zum Erkennen der einzelnen Sprunganweisungen. Diese Prozeduren erzeugen auch die einzelnen Informationen zur Erstellung des Zwischenstrings.

SPRUNG:

Leistung:

Diese Funktionsprozedur stellt mit Hilfe weiterer Prozeduren fest, welcher Sprungtyp (siehe Syntax) vorliegt, sammelt die erzeugten Informationen, stellt sie zum Zwischenstring zusammen und legt sie auf der entsprechenden Datei ab. Nachdem diese Prozedur durchlaufen ist, ist der Inhalt von TEXTZEILE ausgewertet. Liegt eine Sprunganweisung vor, wird der Funktionswert true gesetzt, ansonsten auf false.

Parameter:

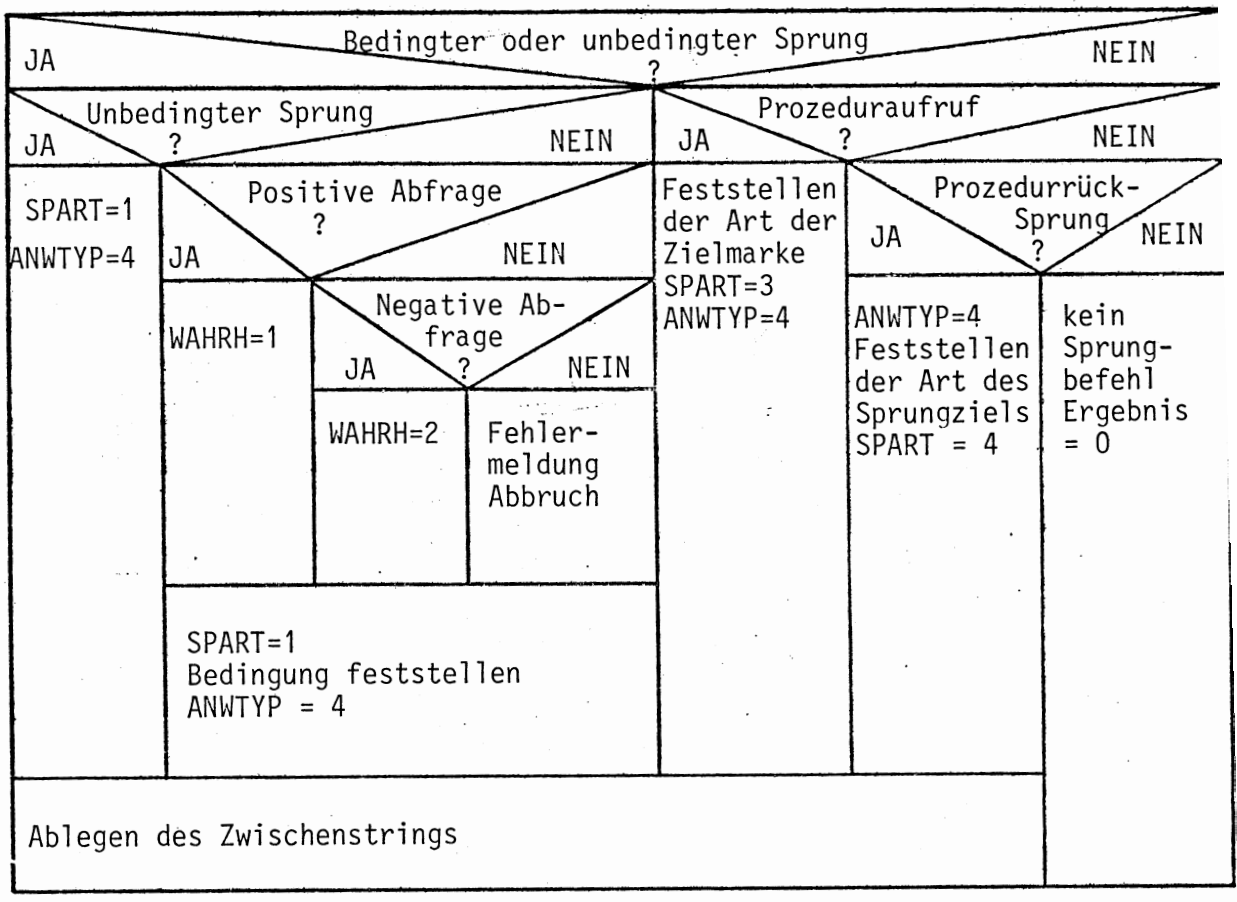
keine

Aufgerufene Prozeduren:	in Modul:
TESTWO	PR1
LABEL	PR7
BEDINGUNG	PR7
PUT7DISK	PR7

Aufrufende Prozeduren:	aus Modul:
PROC	PR3
Task LAUF1	SPA

Fehlerausgänge:

keine


Bedingter oder unbedingter Sprung
JA
?
NEIN
Unbedingter Sprung
JA
?
NEIN
Prozeduraufruf
JA
?
NEIN
SPART=1
ANWTYP=4
Positive Abfrage
?
JA
NEIN
Feststellen der Art der Zielmarke
SPART=3
ANWTYP=4
Prozedurrück-Sprung
JA
?
NEIN
WAHRH=1
Negative Ab-frage
JA
?
NEIN
ANWTYP=4
Feststellen der Art des Sprungziels
SPART = 4
kein Sprung-befehl
Ergebnis = 0
WAHRH=2
Fehler-meldung
Abbruch
SPART=1
Bedingung feststellen
ANWTYP = 4
Ablegen des Zwischenstrings

BEDINGUNG:

Leistung:

Stellt die Art der Bedingung fest (Selektion oder eine Zuweisung bzw. arith. Ausdruck wird mit einem anderen Operanden verglichen). In jedem Fall werden durch den Aufruf der entsprechenden Prozedur die Informationen für den Zwischenstring gewonnen.
Liegt keine Selektion vor, werden die Prozeduren ZUWAUS oder ARIAUS und dann VERGLEI aufgerufen, um die Kennziffer des Vergleichsoperators, sowie Informationen über die Art des Operanden rechts vom Vergleichsoperanden zu erhalten. Liegt eine Selektion vor, werden alle Informationen durch die Prozedurch SELEKT gewonnen.

Parameter:

OPTYP: Integer Variable
Typ des Operanden, der rechts vom Vergleichsoperator steht.

OPNAM: 8-elementiges Zeichenfeld
Name des Operanden

OPOFSET: 8-elementiges Zeichenfeld
Offset dieses Operanden, falls vorhanden.

OPERATOR: Integer Variable
Kennzahl des Vergleichsoperators

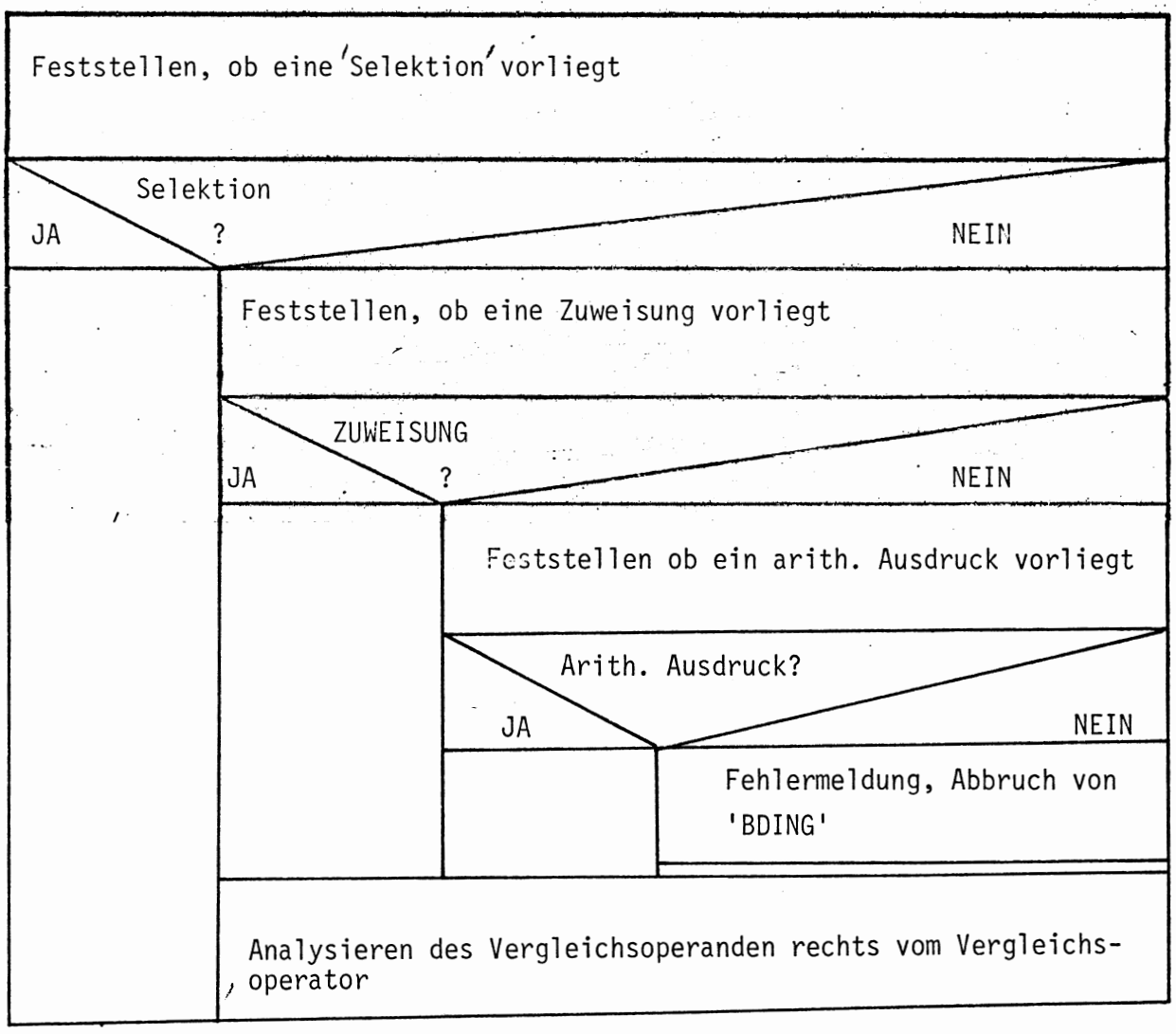
ZAHL: 8-elementiges Zeichenfeld
Selektionszahl, falls statt eines Vergleichs eine Selektion (siehe Syntax) vorliegt.

OPART: Integer Variable
Grundtyp des Operanden rechts vom Vergleichsoperators.

Aufgerufene Prozeduren:	in Modul:
SELEKT	PR7
ZUWAUS	PR6
ARIAUS	PR6
VERGLEICH	PR7
FEHLER	PRØ

aufrufende Prozeduren:	aus Modul:
SPRUNG	PR7

Fehlerausgänge: Kennziffer: 15
Die Stellen, von denen aus eine Fehlermeldung erfolgen kann, sind dem Diagramm zu entnehmen.
Anzahl der Fehlerausgänge: 1


Feststellen, ob eine 'Selektion' vorliegt
Selektion
JA
?
NEIN
Feststellen, ob eine Zuweisung vorliegt
ZUWEISUNG
JA
?
NEIN
Feststellen ob ein arith. Ausdruck vorliegt
Arith. Ausdruck?
JA
NEIN
Fehlermeldung, Abbruch von 'BDING'
Analysieren des Vergleichsoperanden rechts vom Vergleichs-operator

VERGLEICH:

Leistung:

Liegt keine Selektion vor, wird zur Bestimmung der Vergleichsoperators und zur Gewinnung von Informationen über den Operanden rechts vom Operator diese Prozedur aufgerufen. Die gewonnenen Informationen werden über die Parameter an die aufrufende Stelle zurückgegeben.

Parameter:

OPTYP: Integer Variable
Liegt ein Vergleich vor, wird an die aufrufende Stelle in diesem Parameter der Typ der Operanden rechts vom Vergleichsoperator zurückgegeben.

OPNAM: 8-elementiges Zeichenfeld
Im Fall eines Vergleichs wird in diesem Feld der Name des Operanden an die aufrufende Stelle zurückgegeben.

OPOFSET: 8-elementiges Zeichenfeld
möglicher Offset

OPERATOR: Integer Variable
Kennziffer des Vergleichsoperators

OPART: Integer Variable
Grundtyp des Operanden

Aufgerufene Prozeduren:	in Modul:
TESTWO	PR1
OPERAND	PR6
FEHLER	PRØ

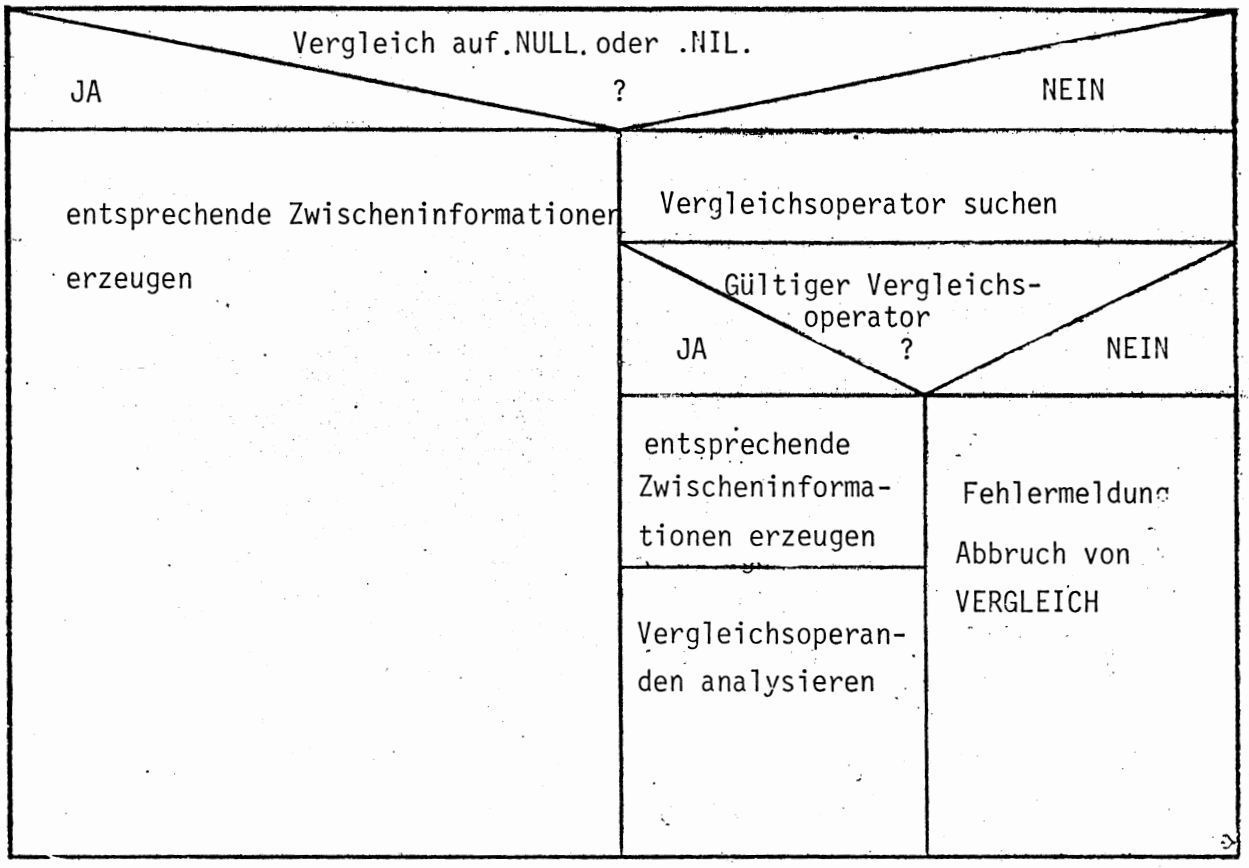
Aufrufende Prozeduren:	aus Modul:
SPRUNG	PR7

Fehlerausgänge:

Kennziffer: 16

Die Stellen, von denen aus Fehlermeldungen erfolgen können, sind dem Diagramm zu entnehmen.

Anzahl der Fehlerausgänge: 1


Vergleich auf.NULL.oder .NIL.
JA
?
NEIN
entsprechende Zwischeninformationen erzeugen
Vergleichsoperator suchen
Gültiger Vergleichs-operator
JA
?
NEIN
entsprechende Zwischeninforma-tionen erzeugen
Vergleichsoperan-den analysieren
Fehlermeldung
Abbruch von VERGLEICH

SELEKT:

Leistung:

Selekt stellt fest, ob eine Selektion vorliegt.
Mit dieser Prozedur werden auch die nötigen Informationen über die Art des Operanden und die Größe der Selektionszahl gewonnen. Über die einzelnen Parameter werden die gewonnenen Informationen an die aufrufende Stelle zurückgegeben. Falls keine Selektion als BEdingung vorliegt, wird eine Kenngröße auf Ø gesetzt und die Stellung des TEXTPOINTERS nicht verändert, ansonsten, wird die Kenngröße auf 1 gesetzt und der TEXTPOINTER zeigt auf das nächste zu verarbeitende Zeichen.

Parameter:

OPTYP: Integer Variable
In diesem Parameter wird im Fall einer Selektion der Typ des Selektionsoperanden an die aufrufende Stelle zurückgegeben.

OPNAM: 8-elementiges Zeichenfeld
Name des Selektionsoperanden, falls eine Selektion vorliegt.

OPOFSET: 8-elementiges Zeichenfeld
eventueller Offset des Selektionsoperanden.

OPERATOR: Integer Variable
1 falls eine Selektion vorliegt
0, sonst alle anderen Parameter sind dann undefiniert.

ZAHL: 8-elementiges Zeichenfeld
Enthält nach Abschluß der Prozedur im Fall einer Selektion die Selektionszahl.

OPART: Integer Variable
Grundtyp des Selektionsoperanden

Aufgerufene Prozeduren:	in Modul:
NAMES	PR1
NEXTCHAR	PR1
OPERAND	PR6

Aufrufende Prozeduren:	aus Modul:
BEDING	PR7

Fehlerausgänge:

keine

TEXTPOINTER retten, um die Ausgangssituation wiederherstellen zu können

Feststellen, ob in 'TEXTZEILE' als nächstes Sonderzeichen eine '(' steht

War das nächste Sonderzeichen eine '('?

JA

NEIN

./.

Feststellen, ob in 'TEXTZEILE' als nächstes Sonderzeichen '(' steht

War das nächste oder übernächste Sonderzeichen eine '('?

JA

NWIN

OPERATOR mit 1 besetzen (positive Antwort)

Es liegt keine 'Selektion' vor

Operanden analysieren

Selektionszahl feststellen

Textpointer in die Ausgangsposition bringen

LABEL:

Leistung:

Diese Prozedur stellt die Art des Sprungziels (indirekt, direkt) fest. und gibt die gewonnenen Informationen in den Parametern an die aufrufende Stelle zurück.

Parameter:

LTYP: Fixed Variable
In dieser Variablen wird der Typ des Sprungziels zurückgegeben
4: Indirekte Sprungzielangabe in Registerspeicherzelle
3: " " " Adresse
1: direkte Sprungzielangabe

LNAME: 8-elementiges Zeichenfeld.
Dieses Feld enthält den Namen des Sprungziels als Zeichenkette.

Aufgerufene Prozeduren:	in Module:
NAMES	PR1
REGISTER	PR6

aufrufende Prozeduren:	aus Modul:
SPRUNG	

Fehlerausgänge:

keine

<table>
<tr><th colspan="2">JA — liegt eine indirekte Adressierung vor? (Substitution) — NEIN</th><th rowspan="2">entsprechende Zwischeninformation erzeugen</th></tr>
<tr><th>JA — Substitution mit Register ? — NEIN</th><th></th></tr>
<tr><td>entsprechende Zwischeninformation erzeugen</td><td>entsprechende Zwischeninformation erzeugen</td><td></td></tr>
</table>

```
 1   MODULE PR7;
 2   SYSTEM;
 3         PSEA←→DISK:;
 4         SDAU→DRUCKER:;
 5         LKEI←LESER:;
 6
 7   PROBLEM;
 8   /* SPEZIFIKATION DER IN DIESEM MODUL BENUTZTEN GLOBALEN DATEN */
 9   DCL TEXTPOIN FIXED GLOBAL;
10         DCL TEXTZEILE(8Ø) CHAR GLOBAL;
11   /* SPEZIFIKATION DER IN DIESEM MODUL  VERWENDETEN PROZEDUREN */
12   DCL ZUWAUS ENTRY RETURNS (BIT) GLOBAL;
13   DCL ARIAUS ENTRY RETURNS (BIT) GLOBAL;
14   DCL OPERAND ENTRY (FIXED,FIXED,(16)CHAR)GLOBAL;
15   DCL COMPF ENTRY(FIXED,FIXED,FIXED,FIXED,FIXED,FIXED,FIXED,(4Ø)CHAR)
16         GLOBAL;
17   DCL FEHLER ENTRY(FIXED) GLOBAL;
18   DCL REGISTER ENTRY (FIXED,FIXED,(8)CHAR) GLOBAL;
19   DCL NEXTCHAR ENTRY RETURNS (CHAR) GLOBAL;
20   DCL NAMES ENTRY ((8) CHAR,FIXED) GLOBAL;
21   DCL TESTWO ENTRY (VAL CHAR(9) ) RETURNS (BIT) GLOBAL;
22
23   /*************************************************************************/
24   /* SPRUNG ERKENNT UND ANALYSIERT SPRUNGANWEISUNGEN                       */
25   /*************************************************************************/
26   SPRUNG: PROC RETURNS (BIT) GLOBAL
27         DCL ERGEBNIS BIT;
28         DCL NAME(8) CHAR;
29        DCL SONDER FIXED;
30         DCL LWERT(8) CHAR;
31         DCL OPTYP FIXED INITIAL (Ø);
32         DCL OFSET(8) CHAR INITIAL(' ');
33         DCL LTYP FIXED INITIAL(Ø);
34         DCL LNAME(8) CHAR INITIAL(' ');
35         DCL OPERATOR FIXED INITIAL(Ø);
36         DCL (OPNAM,OPOFSET,SZAHL)(8) CHAR INITIAL(' ');
37         DCL SPART FIXED INITIAL(Ø);
38         DCL WAHRH FIXED INITIAL(Ø);
39         DCL ANWTYP FIXED;
40         DCL Z5(8) CHAR INITIAL(' ');
41         DCL OPART FIXED INITIAL (Ø);
42         DCL FEHLART FIXED INITIAL(14);
43         /* BEDINGTER ODER UNBEDINGTER SPRUNG$ */
44         IF TESTWO('.TO.←     ') THEN
45           BEGIN;
46              ERGEBNIS='1'B1;
47              ANWTYP=4;
48              CALL LABEL(LTYP,LNAME);
49              /* UNBEDINGTER SPRUNG$ */
50              IF TEXTZEILE(TEXTPOINTER-1) EQ ';' THEN SPART=1; /* JA */
51              ELSE
52                /* NEIN */
53                BEGIN;
54                   /* POSITIVE ABFRAGE$ */
55                   IF TESTWO('IF.←      ') THEN WAHRH=1;
56                   FI;
57                   /* NEGATIVE ABFRAGE$ */
58                   IF TESTWO('IFNOT.←   ') THEN WAHRH=2;FI;
59                   SPART=2;
60                   /* BEDINGUNG ANALYSIEREN */
```

```
61                    CALL BEDINGUNG(LWERT,OPTYP,OPNAM,OPOFSET,OPERATOR,SZA
62                                                                    OPAR
63                  END;
64                FI;
65                    GOTO FIN;
66            END;
67          FI;
68          /* PROZEDURAUFRUF$ */
69          IF TESTWO('.CALL.€  ') THEN
70            /* JA */
71            BEGIN;
72               ERGEBNIS='1'B1;
73               ANWTYP=4;
74               /* FESTSTELLEN DER ART DER SPRUNGZIELANGABE */
75               CALL LABEL(LTYP,LNAME);
76               SPART=3;
77               GOTO FIN;
78            END;
79          FI;
80         /* PROZEDURRUECKSPRUNG$ */
81         IF TESTWO('.RETURN.€') THEN
82           /* JA */
83           BEGIN;
84               ERGEBNIS='1'B1;
85               ANWTYP=4;
86               /* FESTSTELLEN DER ART DER SPRUNGZIELANGABE */
87               CALL LABEL(LTYP,LNAME);
88               SPART=4;
89               GOTO FIN;
90            END;
91          FI;
92          /* KEINE SPRUNGANWEISUNG */
93          ERGEBNIS='Ø'B1;
94    FIN:  IF ERGEBNIS THEN
95            CALL PUT7DISK(ANWTYP,SPART,LTYP,WAHRH,OPERATOR,OPTYP,OPART,
96               LNAME,OPNAM,OPOFSET,SZAHL,Z5);
97          FI;
98          RETURN(ERGEBNIS);
99    END;/*ENDE VON SPRUNG*/
100
101   /*************************************************************************
102   /* 'BEDING' ANALYSIERT DEN BEDINGUNGSTEIL BEI EINEM BEDINGTEN SPRUNG
103   /*************************************************************************
104   BEDING:PROC (LWERT,OPTYP,OPNAM,OPOFSET,OPERATOR,ZAHL,OPART)
105         GLOBAL
106         DCL LWERT(8) CHAR;
107         DCL OPTYP FIXED;
108         DCL (OPNAM,OPOFSET)(8) CHAR;
109         DCL OPERATOR FIXED;
110         DCL ZAHL(8) CHAR;
111         DCL OPART FIXED;
112         DCL FEHLART FIXED INITIAL(15);
113         DCL RETTEXT FIXED;
114         /* FESTSTELLEN OB EINE SELEKTION VORLIEGT */
115         CALL SELEKT (OPTYP,OPNAM,OPOFSET,OPERATOR,ZAHL,OPART);
116         /* SELEKTION$ */
117         IF OPERATOR EQ Ø THEN
118           BEGIN;
119              /* FESTSTELLEN OB EINE ZUWEISUNG VORLIEGT */
120              /* ZUWEISUNG$ */
```

```
121              IF ZUWAUS EQ 'Ø'B1 THEN
122                BEGIN;
123                /* FESTSTELLEN OB EINE
124                /* FESTSTELLEN OB EIN ARIRITH. AUSDRUCK VORLIEGT */
125                /* ARITH. AUSDRUCK$ */
126                IF ARIAUS EQ 'Ø'B1 THEN
127                   /* NEIN */
128                   BEGIN;
129                     /* FEHLERMELDUNG,ABBRUCH VON 'BEDING' */
130                     CALL FEHLER(FEHLART);
131                     GOTO FIN;
132                   END;
133                FI;
134                END;
135             FI;
136             /* ANALYSIEREN DES VERGLEICHSOPERANDEN RECHTS VOM */
137             /* SCHLUESSELWORT 'IF' */
138             CALL VERGLEICH(OPTYP,OPNAM,OPOFSET,OPERATOR,OPART);
139          END;
140       FI;
141  FIN:;
142       RETURN;
143  END;/*ENDE VON BEDING*/
144
145  /*************************************************************************/
146  /* 'VERGLEICH' ANALYSIERT DEN VERGLEICH IN EINEM BEDINGTEN SPRUNG    */
147  /*************************************************************************/
148  VERGLEICH:PROC (OPTYP,OPNAM,OPOFSET,OPERATOR,OPART)
149       GLOBAL
150       DCL OPTYP FIXED;
151       DCL (OPNAM,OPOFSET)(8) CHAR;
152       DCL OPERATOR FIXED;
153       DCL OPART FIXED;
154       DCL FEHLART FIXED INITIAL(16);
155       DCL VEROPERATOR(6) VAL CHAR(4) IDENTICAL
156                     ('.EQ.','.NE.','.GT.','.LE.','.LT.','.GE.');
157       DCL (RETTEXT,I,J) FIXED;
158       DCL FLAG FIXED;
159       DCL HILF(16)CHAR;
160       OPERATOR=Ø;
161       /* VERGLEICH AUF '.NULL.' ODER '.NIL.'$ */
162       IF TESTWO('.NIL.<   ') THEN BEGIN;
163                                      OPERATOR=9;
164                                      GOTO FIN;
165                                   END;
166       FI;
167       IF TESTWO('.NULL.<  ') THEN BEGIN;
168                                      OPERATOR=8;
169                                      GOTO FIN;
170                                   END;
171       FI;
172       /* VERGLEICHSOPERATOR SUCHEN */
173       FOR I TO 6 REPEAT;
174         BEGIN;
175            FLAG=TEXTPOINTER;
176            FOR J TO 4 REPEAT;
177               IF (J CHAR VEROPE(I)) NE NEXTCHAR THEN
178                 GOTO WEITER;
179               FI;
180            END ;
```

```
181               /* ENTSPRECHENDE ZWISCHENINFORMATION ERZEUGEN */
182               OPERATOR=I;
183               GOTO FERTIG;
184               WEITER:;
185               TEXTPOINTER=FLAG;
186             END;
187           END;
188           /* GUELTIGER VERGLEICHSOPERATOR$ */
189           /* NEIN */
190           /* FEHLERMELDUNG, ABBRUCH VON 'VERGLEICH' */
191           CALL FEHLER(FEHLART);
192           GOTO FIN;
193           /* VERGLEICHSOPERANDEN ANALYSIEREN */
194     FERTIG:CALL OPERAND(OPTYP,OPART,HILF);
195           FOR I TO 8 REPEAT;
196             OPNAM(I)=HILF(I);
197             OPOFSET(I)=HILF(I+8);
198           END;
199     FIN:RETURN;
200     END;/*ENDE VON VERGLEICH*/
201
202     /*****************************************************************
203     /* 'SELEKT ERKENNT UND ANALYSIERT EINE 'SELEKTION'                *
204     /*****************************************************************
205     SELEKT:PROC (OPTYP,OPNAM,OPOFSET,OPERATOR,ZAHL,OPART)
206           GLOBAL
207           DCL OPTYP FIXED;
208           DCL (OPNAM,OPOFSET)(8) CHAR;
209           DCL OPERATOR FIXED;
210           DCL ZAHL(8) CHAR;
211           DCL OPART FIXED;
212           DCL (RETTEXT,SONDER) FIXED;
213           DCL NAME(8) CHAR;
214           DCL NEXT CHAR;
215           DCL I FIXED;
216           DCL HILF(16)CHAR;
217           /*'TEXTPOINTER' RETTEN UM DIE AUSGANGSSITUATION WIEDERHERSTELLE
218           /* ZU KOENNEN */
219           RETTEXT=TEXTPOINTER;
220           /* FESTSTELLEN OB IM TEXTSTRING ALS NAECHSTES SONDERZEICHEN */
221           /* EINE '(' STEHT
222           CALL NAMES(NAME,SONDER);
223           /* FALLS NEIN FESTSTELLEN OB NUN DAS NAECHSTE SONDERZEICHEN /*
224           /* EINE '( ' IST */
225           IF SONDER NE 9 THEN CALL NAMES(NAME,SONDER);FI;
226           /* WAR DAS NAECHSTE ODER UEBERNAECHSTE SONDERZEICHEN EINE '('$
227           IF SONDER NE 9 THEN
228             /* NEIN */
229             BEGIN;
230               /* ES LIEGT KEINE SELEKTION VOR,'TEXTPOINTER' WIEDER IN
231                  AUSGANGSPOSITION BRINGEN */
232               OPERATOR=0;
233               TEXTPOINTER=RETTEXT;
234             END;
235           ELSE
236             BEGIN;
237               /* ES LIEGT EINE SELEKTION VOR$ */
238               OPERATOR=1;
239               TEXTPOINTER=RETTEXT;
240               FOR I TO 8 REPEAT;
```

```
241               HILF(I)=OPNAM(I);
242               HILF(I+8)=OPOFSET(I);
243               END;
244               /* TYP DES SELEKTIONSOPERANDEN FESTSTELLEN */
245               CALL OPERAND(OPTYP,OPART,HILF);
246                  /* SELEKTIONSZAHL BESTIMMEN */
247               FOR I REPEAT;
248                 NEXT=NEXTCHAR;
249                 IF NEXT NE ')' THEN
250                    ZAHL(I)=NEXT;
251                 ELSE
252                    GOTO FIN;
253                 FI;
254               END;
255            END;
256         FI;
257   FIN:RETURN;
258   END;/*ENDE VON SELEKT*/
259
260   /*************************************************************************/
261   /* 'LABEL' STELLT FEST IN WELCHER FORM DIE SPRUNGZIELANGABE VORLIEGT */
262   /*************************************************************************/
263   LABEL:PROC (LTYP,LNAME) GLOBAL
264         DCL LTYP FIXED;
265         DCL LNAME(8) CHAR;
266         DCL SONDER FIXED;
267         DCL OPART FIXED ;
268         /* LIEGT EINE INDIEREKTE ADRESSIERUNG VOR$ */
269         CALL NAMES(LNAME,SONDER);
270         IF SONDER EQ 8 THEN
271           /* JA */
272           BEGIN;
273              /* SUBSTITUTION MIT REGISTER$ */
274                   /* ENTSPRECHENDE ZWISCHENINFORMATION ERZEUGEN */
275              CALL REGISTER(SONDER,OPART,LNAME);
276              IF SONDER EQ 0 THEN
277                /* NEIN */
278                /* SUBSTITUTION MIT SPEICHERZELLE */
279                BEGIN;
280                   /* ENTSPRECHENDE ZWISCHENINFORMATION ERZEUGEN */
281                   CALL NAMES(LNAME,SONDER);
282                LTYP=4;
283                END;
284              ELSE
285                   LTYP=3;
286              FI;
287           END;
288         ELSE
289              LTYP=1;
290         FI;
291         RETURN;
292   END;/*ENDE VON LABEL*/
293
294   /*************************************************************************/
295
296   PUT7DISK:PROC (ANWTYP,F1,F2,F3,F4,F5,F6,Z1,Z2,Z3,Z4,Z5) GLOBAL
297         DCL (ANWTYP,F1,F2,F3,F4,F5,F6) FIXED;
298         DCL (Z1,Z2,Z3,Z4,Z5)(8) CHAR;
299         DCL I FIXED;
300         DCL HILFE(40) CHAR;
```

```
301           FOR I TO 8 REPEAT;
302             HILFE(I   )=Z1(I);
303             HILFE(I+8)=Z2(I);
304             HILFE(I+16)=Z3(I);
305             HILFE(I+24)=Z4(I);
306             HILFE(I+32)=Z5(I);
307           END;
308           CALL COMPF(ANWTYP,F1,F2,F3,F4,F5,F6,HILFE);
309           RETURN;
310     END;/*ENDE VON PUTDISK*/
311     MODEND;
```

MODUL PR8
=========

Dieser Modul enthält die Prozeduren zur Speicherplatzvereinbarung, der Strukturvereinbarung und der Längenvereinbarung. Diese 3 Prozeduren sind voneinander total unabhängig und wurden nur deshalb zu einem PEARL-Modul zusammengefaßt, da der PEARL-Compiler an der S306 nur 9 Prozedurmodule zuläßt.

STRUC:

Leistung:

Die Funktionsprozedur STRUC analysiert eine Strukturvereinbarung und speichert die Kenndaten (Struktur) in einer Liste (TYPLIST); Diese Prozedur ermittelt die Abstände der Strukturkomponenten zum Strukturkopf. Dieser Abstand und der Name der Komponente wird durch die Prozedur GETTYP in die Liste eingetragen.

STRUC stellt nicht fest, ob ein Strukturname oder ein Komponentenname mehr als einmal definiert wird. Bei einer Strukturvereinbarung darf nur die erste Komponente eine zusammengesetzte Komponente sein. Alle anderen Strukturkomponenten müssen Elementartypen sein.

Diese Prozedur liest solange SPASS-Zeilen (Karten) ein, bis das Schlüsselwort für das Ende der Strukturvereinbarung kommt (STREND) und kehrt dann an die aufrufende Stelle zurück. Der Funktionswert ist true, wenn eine Strukturvereinbarung vorliegt, ansonsten false. Die Stellung des Textpointers bleibt dann unverändert.

Parameter:
keine

Aufgerufene Prozeduren:	in Modul:
TESTWO	PR1
NAMES	PR1
PUTTYP	PR2
GETTYP	PR2
KONVERT	PRØ
RKONVERT	PRØ
FEHLER	PRØ

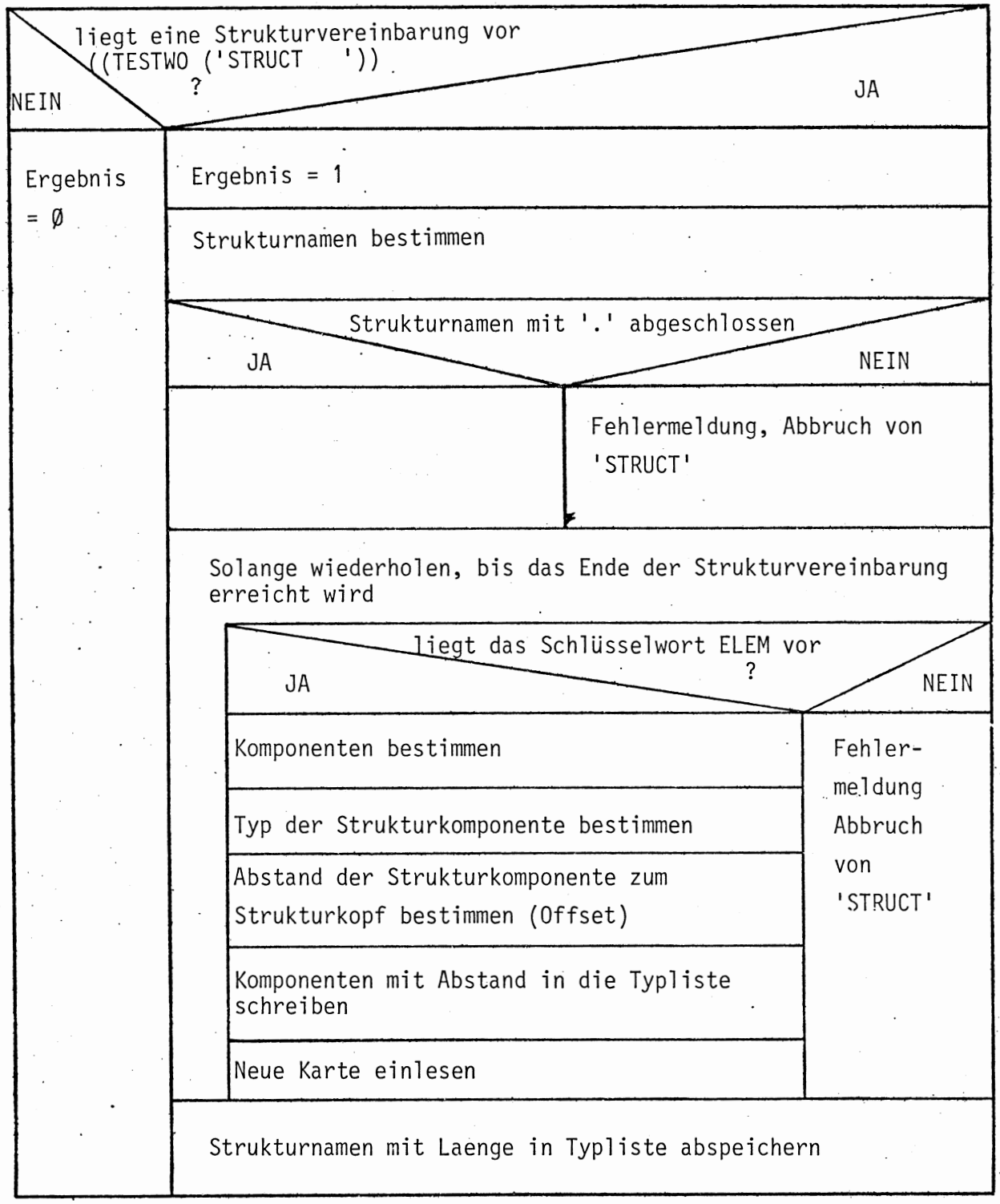
Aufrufende Prozeduren:	aus Modul:
Task LAUF1	SPA

Fehlerausgänge:

Fehlerkennung: 19

Die Stellen, von denen aus eine Fehlermeldung erfolgen kann, sind dem Diagramm zu entnehmen.

Anzahl der Fehlerausgänge: 3


liegt eine Strukturvereinbarung vor
((TESTWO ('STRUCT ')) ?
NEIN
JA
Ergebnis = Ø
Ergebnis = 1
Strukturnamen bestimmen
Strukturnamen mit '.' abgeschlossen
JA
NEIN
Fehlermeldung, Abbruch von 'STRUCT'
Solange wiederholen, bis das Ende der Strukturvereinbarung erreicht wird
liegt das Schlüsselwort ELEM vor ?
JA
NEIN
Komponenten bestimmen
Typ der Strukturkomponente bestimmen
Abstand der Strukturkomponente zum Strukturkopf bestimmen (Offset)
Komponenten mit Abstand in die Typliste schreiben
Neue Karte einlesen
Fehler-meldung Abbruch von 'STRUCT'
Strukturnamen mit Laenge in Typliste abspeichern

LENGTH:

Leistung:

Mit der Längenvereinbarung wird die Anzahl der Speicherplätze für die Elementartypen festgelegt. Diese Längenvereinbarung soll die Anpassung von SPASS-Programmen an konkrete Maschinen erleichtern.

Die Funktionsprozedur Length stellt die Längen fest und trägt die Namen des Grundtyps (POINTER, INTEGER, BITS) und dessen Länge mit der Prozedur GETTYP in die Typenliste ein.

Liegt eine Längenvereinbarung vor, hat LENGTH den Funktionswert true und ansonsten false. In diesem Fall bleibt die Stellung des TEXTPOINTERS unverändert.

Parameter:
keine

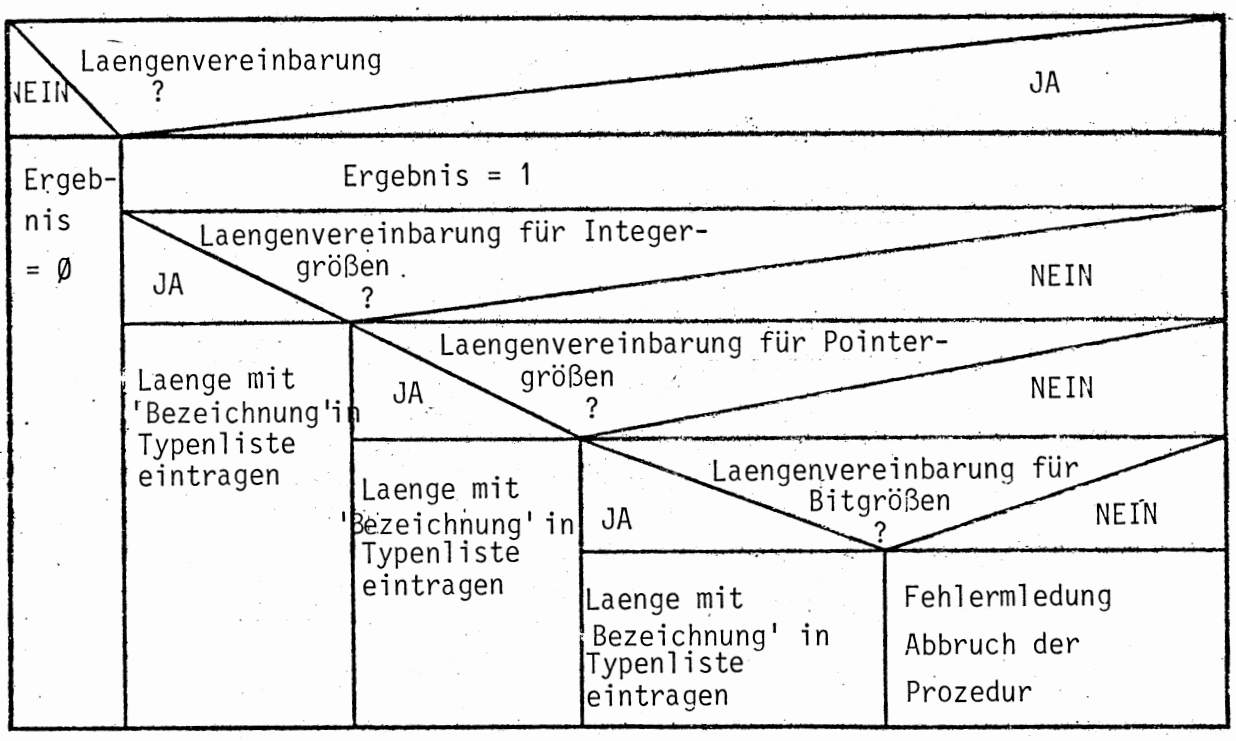
Aufgerufene Prozeduren:	in Modul:
TESTWO	PR1
NAMES	PR1
GETTYP	PR2

Aufrufende Prozeduren:	aus Modul:
Task LAUF1	SPA

Fehlerausgänge:

Fehlerkennung: 2Ø

Die Stellen, von denen aus eine Fehlermeldung erfolgen kann, sind dem Diagramm zu entnehmen.
Anzahl der Fehlerausgänge: 1


NEIN
Laengenvereinbarung ?
JA
Ergeb-
nis
= Ø
Ergebnis = 1
Laengenvereinbarung für Integer-
größen ?
JA
NEIN
Laenge mit
'Bezeichnung'in
Typenliste
eintragen
Laengenvereinbarung für Pointer-
größen ?
JA
NEIN
Laenge mit
'Bezeichnung' in
Typenliste
eintragen
Laengenvereinbarung für
Bitgrößen ?
JA
NEIN
Laenge mit
'Bezeichnung' in
Typenliste
eintragen
Fehlermledung
Abbruch der
Prozedur

SPACE:

Leistung:

Die Funktionsprozedur SPACE überprüft die Speicherplatzvereinbarungen auf die Richtigkeit und ermittelt die nötigen Informationen zur Erstellung des Zwischenstrings. Diese Prozedur liest solange SPASS-Text (Karten) ein und erzeugt den entsprechenden Zwischencode, bis das Ende einer Speicherplatzreservierung erreicht ist.

Liegt eine Speicherplatzvereinbarung vor, ist der Funktionswert von SPACE true und ansonsten false.

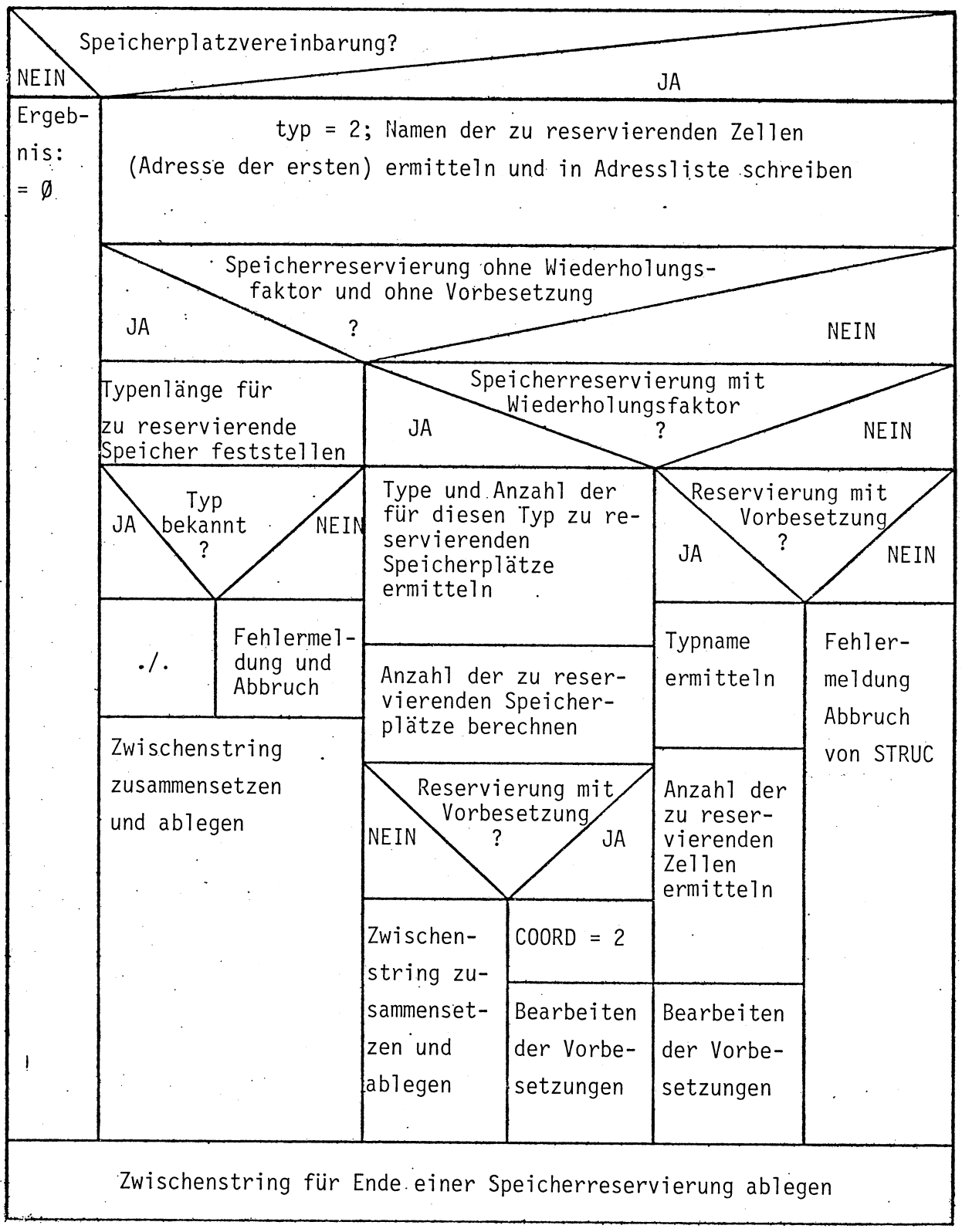
Parameter:
keine

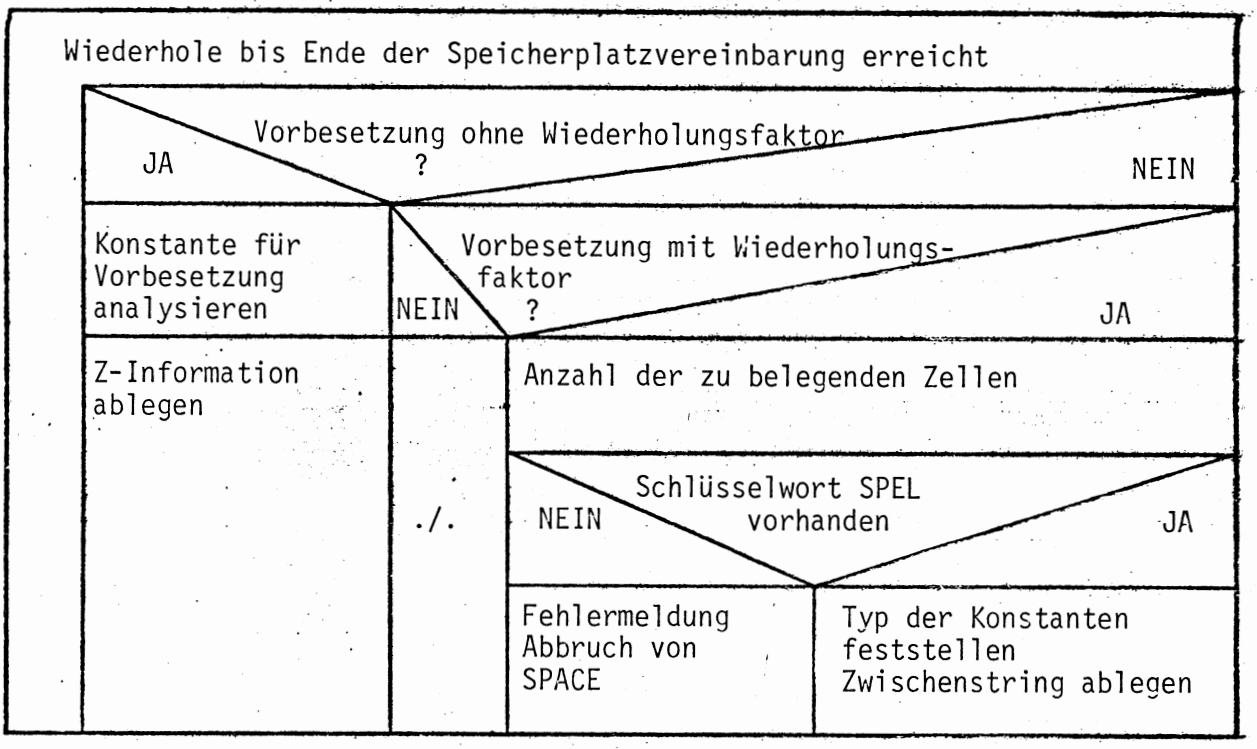
Aufgerufene Prozeduren:	in Modul:
TESTWORD	PR1
NAMES	PR1
EINAUS	PR1
PUTTYP	PR2
GETADR	PR2
KONSTANTE	PR6
KONVERT	PRØ
RKONVERT	PRØ
PUT8DISK	PR8

Aufrufende Prozeduren:	aus Modul:
Task LAUF1	SPA

Fehlerausgänge:
Kennung: 21
Die Stellen, von denen aus Fehlermeldungen kommen können, sind dem Diagramm zu entnehmen.

Anzahl der Fehlerausgänge: 3


Speicherplatzvereinbarung?
NEIN
JA
Ergeb-
nis:
= Ø
typ = 2; Namen der zu reservierenden Zellen
(Adresse der ersten) ermitteln und in Adressliste schreiben
Speicherreservierung ohne Wiederholungs-
faktor und ohne Vorbesetzung
JA
?
NEIN
Typenlänge für
zu reservierende
Speicher feststellen
Speicherreservierung mit
Wiederholungsfaktor
JA
?
NEIN
Typ
bekannt
?
JA
NEIN
./.
Fehlermel-
dung und
Abbruch
Type und Anzahl der
für diesen Typ zu re-
servierenden
Speicherplätze
ermitteln
Anzahl der zu reser-
vierenden Speicher-
plätze berechnen
Reservierung mit
Vorbesetzung
?
NEIN
JA
Zwischenstring
zusammensetzen
und ablegen
Zwischen-
string zu-
sammenset-
zen und
ablegen
COORD = 2
Bearbeiten
der Vorbe-
setzungen
Reservierung mit
Vorbesetzung
?
JA
NEIN
Typname
ermitteln
Anzahl der
zu reser-
vierenden
Zellen
ermitteln
Bearbeiten
der Vorbe-
setzungen
Fehler-
meldung
Abbruch
von STRUC
Zwischenstring für Ende einer Speicherreservierung ablegen


Wiederhole bis Ende der Speicherplatzvereinbarung erreicht
Vorbesetzung ohne Wiederholungsfaktor ?
JA
NEIN
Konstante für Vorbesetzung analysieren
Vorbesetzung mit Wiederholungs-faktor ?
NEIN
JA
Z-Information ablegen
./.
Anzahl der zu belegenden Zellen
Schlüsselwort SPEL vorhanden
NEIN
JA
Fehlermeldung Abbruch von SPACE
Typ der Konstanten feststellen Zwischenstring ablegen

```
-PEARL-SUBSET/1   VERSION 0575/04   ***QUELLPROGRAMM***   MODUL PR8        18.

1    MODULE PR8;
2    SYSTEM;
3          PSEA←→DISK:;
4          SDAU→DRUCKER:;
5          LKEI←LESER:;
6
7    PROBLEM;
8    /* SPEZIFIKATION DER IN DIESEM MODULE BENUTZTEN GLOBALEN DATEN */
9    DCL TEXTPOINTER FIXED GLOBAL;
Ø          DCL TEXTZEILE(8Ø) CHAR GLOBAL;
1    /* SPEZIFIKATION DER IN DIESEM MODULE BENUTZTEN PROZEDUREN */
2    DCL NAMES ENTRY ((8)CHAR,FIXED) GLOBAL;
3    DCL TESTWO ENTRY (VAL CHAR(9))RETURNS (BIT) GLOBAL;
4    DCL FEHLER ENTRY (FIXED) GLOBAL;
5    DCL NEXTCHAR ENTRY RETURNS(CHAR) GLOBAL;
6    DCL KONSTANTE ENTRY ((8) CHAR,FIXED,FIXED) GLOBAL;
7    DCL KONVERT ENTRY ((8) CHAR) RETURNS (FIXED) GLOBAL;
8    DCL RKONVERT ENTRY (FIXED,(8) CHAR) GLOBAL;
9    DCL PUTTYP ENTRY ((8) CHAR,FIXED) GLOBAL;
Ø    DCL PUTADR ENTRY ((8) CHAR,FIXED) GLOBAL;
1    DCL GETTYP ENTRY ((8) CHAR,FIXED) GLOBAL;
2    DCL GETADR ENTRY ((8) CHAR,FIXED) GLOBAL;
3    DCL EINAUS ENTRY (FIXED) GLOBAL;
4    DCL COMPF ENTRY(FIXED,FIXED,FIXED,FIXED,FIXED,FIXED,FIXED,(4Ø)CHAR)
5          GLOBAL;
6
7    /*************************************************************************/
8    /* 'STRUC' ERKENNT UND ANALYSIERT EINE STRUKTURVEREINBARUNG              */
9    /*************************************************************************/
Ø    STRUC:PROC RETURNS (BIT) GLOBAL
1          DCL ERGEBNIS BIT;
2          DCL ALPHA(8) CHAR INITIAL(' ');
3          DCL NAME(8) CHAR;
4          DCL K FIXED;
5          DCL SONDER FIXED;
6          DCL RETTYP FIXED;
7          DCL STNAME(8) CHAR;
8          DCL XTYP(8) CHAR;
9          DCL MODUS FIXED INITIAL(1);
Ø          DCL OPART FIXED;
1          DCL FEHLART FIXED INITIAL(19);
2          K=-1;
3          /* LIEGT EINE STRUKTURVEREINBARUNG VOR$ */
4          IF TESTWO('STRUCT←  ') THEN
5            /* JA */
6            BEGIN;
7               ERGEBNIS='1'B1;
8               /* STRUKTURNAMEN BESTIMMEN */
9               CALL NAMES(NAME,SONDER);
5               /* STRUKTURNAMEN MIT ':' ABGESCHLOSSEN$ */
               IF SONDER NE 6 THEN
                              /* NEIN */
                              BEGIN;
                                /* FEHLERMELDUNG,ABBRUCH VON 'STRUC' */
                                CALL FEHLER(FEHLART);
                                GOTO FIN;
                              END;
               FI;
               CALL NAMES(STNAME,SONDER);
               CALL EINAUS(MODUS);
```

```
61                /* SOLANGE WIEDERHOLEN BIS DAS ENDE DER STRUKTUR- */
62                /* VEREINBARUNG ERREICHT WIRD */
63    NEW:        IF TESTWO('STREND4  ') EQ 'Ø'B1 THEN
64                  BEGIN;
65                     /* LIEGT DAS SCHLUESSELWORT 'ELEM' VOR$ */
66                     IF TESTWO ('ELEM4    ') EQ 'Ø'B1 THEN
67                                                    /* NEIN */
68                                                    BEGIN;
69                                                    /* FEHLERMELDUNG*/
70                                                    /* ABBRUCH VON */
71                                                    /* 'STRUCT' */
72                                                      CALL FEHLER
73                                                        (FEHLART);
74                                                      GOTO FIN;
75                                                    END;
76                     FI;
77                     /* KOMPONENTENNAMEN BESTIMMEN */
78                     CALL NAMES(NAME ,SONDER);
79                /* KOMPONENTENNAMEN MIT ':' ABGESCHLOSSEN$ */
80                     IF SONDER NE 6 THEN
81                                    /* NEIN */
82                                    BEGIN;
83                                     /* FEHLERMELDUNG,ABBRUCH VON 'STRUCT'
84                                     CALL FEHLER(FEHLART);
85                                      GOTO FIN;
86                                    END;
87                     FI;
88                     /* TYP DER STRUKTURKOMPONENTE BESTIMMEN */
89                     CALL NAMES(NAME,SONDER);
90                     CALL NAMES(XTYP,SONDER);
91                     /* ABSTAND DER STRUKTURKOMPONENTE ZUM STRUKTURKOPF */
92                     /* BESTIMMEN */
93                     CALL PUTTYP(XTYP,OPART);
94                     ALPHA(1)=XTYP(7);
95                     ALPHA(2)=XTYP(8);
96                     K=KONVERT(ALPHA)+K;
97                     CALL RKONVERT(K,ALPHA);
98                     NAME(7)=ALPHA(2);
99                     NAME(8)=ALPHA(3);
100                    ALPHA(1)=' ';
101                    ALPHA(2)=' ';
102                    ALPHA(3)=' ';
103                    /* KOMPONENTENNAMEN MIT ABSTAND IN DIE TYPLISTE */
104                    /* SCHREIBEN */
105                    CALL GETTYP(NAME,OPART);
106                    /* NEUE KARTE EINLESEN */
107                    CALL EINAUS(MODUS);
108                    GOTO NEW;
109                 END;
110               FI;
111               /* ENDE DER SCHLEIFE */
112               K=K+1;
113               CALL RKONVERT(K,ALPHA);
114               STNAME(7)=ALPHA(2);
115               STNAME(8)=ALPHA(3);
116               OPART=5;
117               /* STRUKTURNAMEN MIT LAENGE IN TYPLISTE ABSPEICHERN */
118               CALL GETTYP(STNAME,OPART);
119            END;
120          ELSE
```

```
            ERGEBNIS='Ø'B1;
         FI;
FIN: RETURN(ERGEBNIS);
END;/*ENDE VON STRUCT*/

/*************************************************************************/
/* 'LENGTH' ERKENNT UND ANALYSIERT LAENGENVEREINBARUNGEN                 */
/*************************************************************************/
LENGTH:PROC RETURNS(BIT)
GLOBAL
      DCL (ERGEBNIS,ANTWORT)  BIT ;
      DCL NAME(8)  CHAR(1) ;
      DCL SONDERZ  FIXED ;
      DCL RETTEXT FIXED;
      DCL POINT(8) CHAR INITIAL('P','O','I','N','T','E',' ');
      DCL INT(8) CHAR INITIAL('I','N','T',' ');
      DCL BIT(8) CHAR INITIAL('B','I','T','S',' ');
      DCL OPART FIXED;
      DCL FEHLART FIXED INITIAL(2Ø);
      /* LAENGENVEREINBARUNGS */
      ANTWORT=TESTWORD('LENGTH:¢ ');
      IF ANTWORT NE '1'B1 THEN BEGIN;
                               /* NEIN */
                               ERGEBNIS='Ø'B1;
                               GOTO FIN;
                            END;
      FI;
           /* LAENGENVEREINBARUNG FUER INTEGERGROESSENS */
           IF TESTWO('INT=¢     ') THEN
             BEGIN;
                /* LAENGE MIT BEZEICHNUNG IN DIE TYPLISTE EINTRAGENS */
                CALL NAMES(NAME,SONDER);
                IF NAME(2) NE ' ' THEN
                BEGIN;
                INT(7)=NAME(1);
                INT(8)=NAME(2);
                END;
                ELSE INT(7)=NAME(1);
                FI;
                OPART=3;
                CALL GETTYP(INT,OPART);
                GOTO ENDE;
             END;
           FI;
           /* LAENGENVEREINBARUNG FUER POINTERGROESSENS */
           IF TESTWO('POINTER=¢') THEN
             BEGIN;
                /* LAENGE MIT BEZEICHNUNG IN DIE TYPLISTE EINTRAGENS */
                CALL NAMES(NAME,SONDER);
                IF NAME(2) NE ' ' THEN
                BEGIN;
                POINT(7)=NAME(1);
                POINT(8)=NAME(2);
                END;
                ELSE POINT(7)=NAME(1);
                FI;
                OPART=2;
                CALL GETTYP(POINT,OPART);
                GOTO ENDE;
             END;
```

```
181                FI;
182                /* LAENGENVEREINBARUNG FUER BITGROESSEN$ */
183                IF TESTWO('BITS=<   ') THEN
184                  BEGIN;
185                     /* LAENGE MIT BEZEICHNUNG IN DIE TYPLISTE EINTRAGEN$ *
186                     CALL NAMES(NAME,SONDER);
187                     IF NAME(2) NE ' ' THEN
188                     BEGIN;
189                     BIT(7)=NAME(1);
190                     BIT(8)=NAME(2);
191                     END;
192                     ELSE
193                     BIT(7)=NAME(8);
194                     FI;
195                     OPART=4;
196                     CALL GETTYP(BIT,OPART);
197                     GOTO ENDE;
198                  END;
199                FI;
200          /* FEHLERMELDUNG */
201                     CALL FEHLER(FEHLART);
202     ENDE:        ERGEBNIS='1'B1;
203     FIN:  RETURN(ERGEBNIS);
204     END;/*ENDE VON LENGTH*/
205
206     /*************************************************************************/
207     /* 'SPACE' ERKENNT UND ANALYSIERT SPEICHERPLATZVEREINBARUNGEN            */
208     /*************************************************************************/
209     SPACE:PROC RETURNS(BIT)
210     GLOBAL
211            DCL QINFO VAL FILE;
212            DCL ADRESSE(8) CHAR;
213            DCL HILFE(8) CHAR;
214            DCL ERGEBNIS BIT;
215            DCL(CODNR,CONSTTYP) FIXED INITIAL(0);
216            DCL CONST(8) CHAR INITIAL (' ');
217            DCL BELZELL(8) CHAR INITIAL(' ');
218            DCL RESZELL(8) CHAR INITIAL (' ');
219            DCL (F3,F4,F5) FIXED INITIAL(0);
220            DCL Z5(8) CHAR INITIAL(' ');
221            DCL ANWTYP FIXED;
222            DCL NAME(8) CHAR;
223            DCL (SONDER,I) FIXED;
224            DCL (J,K,L) FIXED;
225            DCL RETTEXT FIXED;
226            DCL FEHLART FIXED INITIAL(21);
227            DCL OPART FIXED;
228            DCL MODUS FIXED INITIAL(1);
229            /* SPEICHERPLATZVEREINBARUNG$ */
230            IF TESTWORD('SPACE;<  ') THEN
231             /* JA */
232              BEGIN;
233                 ERGEBNIS='1'B1;
234                 ANWTYP=2;
235                 /* NAMEN DER ZU RESERVIERENDEN ZELLEN ERMITTLN */
236                 CALL NAMES(NAME,SONDER);
237                 FOR I TO 8 REPEAT;
238                   ADRESSE(I)=NAME(I);
239                 END;
240                 /* SPEICHERRESERVIERUNG OHNE WIEDERHOLUNGSFAKTOR UND */
```

```
1               /* VORBESETZUNGEN$ */
2               CALL NAMES(NAME,SONDER);
3               IF SONDER EQ 5 THEN
4                 /* JA */
5                 BEGIN;
6                    /* TYPENLAENGE FUER ZU RESERVIERENDEN SPEICHER FEST- */
7                    /* STELLEN */
8                    CALL PUTTYP(NAME,OPART);
9                    CALL GETADR (ADRESSE,OPART);
Ø                    /* TYP BEKANNT$ */
1                    IF NAME(1) EQ ' ' THEN BEGIN;
2                                           /* NEIN */
3                                        /* FEHLERMELDUNG,ABBRUCH VON */
4                                        /*'SPACE' */
5                                        CALL FEHLER(FEHLART);
6                                        GOTO ENDE;
7                                       END;
8                    FI;
9                    /* ZWISCHENSTRING ZUSAMMENSETZEN UND ABLEGEN */
Ø                    RESZELL(1)=' ';
1                    RESZELL(2)=NAME(7);
2                    RESZELL(3)=NAME(8);
3                    CODNR=1;
4                    CALL PUT8DISK(ANWTYP,CODNR,CONSTTYP,F3,F4,F5,OPART,
5                          ADRESSE,RESZELL,BELZELL,NAME ,Z5);
6                          GOTO ENDE;
7               END;
8               FI;
9               /* SPEICHERRESERVIERUNG MIT WIEDERHOLUNGSFAKTOR$ */
Ø               IF SONDER EQ 2 THEN
1                 /* JA */
2                 BEGIN;
3                   /* TYP UND ANZAHL DER FUER DIESEN TYP ZU */
4                    /* RESERVIERENDEN SPEICHERPLAETZE ERMITTELN */
5                    CALL NAMES(HILFE,SONDER);
6                    CALL PUTTYP(HILFE,OPART);
7                    CALL GETADR (ADRESSE,OPART);
8                    CONST(1)=HILFE(7);
9                    CONST(2)=HILFE(8);
Ø                    /* ANZAHL DER ZU RESERVIERENDEN SPEICHERPLAETZE  */
1                    /* BERECHNEN */
2                    K=KONVERT(CONST)*KONVERT(NAME);
3                    CALL RKONVERT(K,CONST);
4                    RESZELL(1)=CONST(1);
5                    RESZELL(2)=CONST(2);
6                    RESZELL(3)=CONST(3);
7                    /* RESERVIERUNG MIT VORBESETZUNG$ */
8                    IF  SONDER EQ 5 THEN BEGIN;
9                                        /* NEIN */
Ø                                           /* ZWISCHENSTRING ZUSAMMEN- */
1                                           /* SETZEN UND ABLEGEN */
2                                           CODNR=1;
3                                           CALL PUT8DISK (ANWTYP,CODNR,
                                             CONSTTYP,F3,F4,F5,OPART,
                                                          ADRESSE,
5                                               RESZELL,BELZELL,Z5,Z5);
5                                       GOTO ENDE;
7                                      END;
3                    FI;
9                    CODNR=2;
```

```
301                    /* BEARBEITEN DER VORBESETZUNGEN */
302                    GOTO NEW;
303                  END;
304                FI;
305                /* RESERVIERUNG MIT VORBESETZUNG (OHNE WIEDERHOLUNGSFAKTOR)
306                IF SONDER EQ 1 THEN
307                  BEGIN;
308                    IF NAME(1) EQ ' ' THEN BEGIN;
309                                       /* NEIN */
310                                         /* FEHLERMELDUNG,ABBRUCH VON */
311                                         /* 'SPACE' */
312                                       CALL FEHLER(FEHLART);
313                                          GOTO ENDE;
314                                         END;
315                    FI;
316                    CODNR=2;
317                    /* TYPNAMEN ERMITTELN */
318                    CALL PUTTYP(NAME,OPART);
319                    CALL GETADR(ADRESSE,OPART);
320                    RESZELL(1)=' ';
321                    RESZELL(2)=NAME(7);
322                    RESZELL(3)=NAME(8);
323                    /*BEARBEITUNG DER VORBESETZUNGEN */
324                    GOTO NEW ;
325                  END;
326                FI;
327
328                    /* BEARBEITUNG DER VORBESETZUNGEN */
329                    /* WIEDERHOLE SOLANGE BIS DAS ENDE DER SPEICHER- */
330                    /* PLATZVEREINBARUNG ERREICHT WIRD */
331    NEW:        CALL EINAUS(K);
332                CALL NAMES(NAME,SONDER);
333                    /* VORBESETZUNG OHNE WIEDERHOLUNGSFAKTOR$ */
334                IF SONDER EQ 7 THEN
335                  /* JA */
336                  BEGIN;
337                    /* KONSTANTE FUER DIE VORBESETZUNG ANALYSIEREN */
338                    CALL KONSTANTE(NAME,CONSTTYP,OPART);
339                    CONSTTYP=CONSTTYP+1Ø;
340                    BELZELL(2)=' ';
341                    BELZELL(1)=' ';
342                    BELZELL(3)='1';
343                    /* ZWISCHENINFORMATION ABLEGEN */
344                    CALL PUT8DISK(ANWTYP,CODNR,CONSTTYP,F3,F4,F5,OPART.
345                         ADRESSE,RESZELL,BELZELL,NAME,Z5);
346                  END;
347                FI;
348                    /* VORBESETZUNG MIT WIEDERHOLUNGSFAKTOR$ */
349                IF SONDER EQ 2 THEN
350                  BEGIN;
351                    BELZELL(1)=NAME(1);
352                    BELZELL(2)=NAME(2);
353                    BELZELL(3)=NAME(3);
354                    IF TESTWO('SPEL=4    ') EQ 'Ø'B1 THEN BEGIN;
355                                                    CALL FEHLER
356                                                       (FEHLART);
357                                                    GOTO ENDE;
358                                                  END;
359                    FI;
360                    /* TYP DER KONSTANTEN FESTSTELLEN */
```

```
                    CALL KONSTANTE(NAME,CONSTTY,OPART);
                    CONSTTYP=CONSTTYP+1Ø;
                    /* ZWISCHENSTRING ABLEGEN */
                    CALL PUT8DISK(ANWTYP,CODNR,CONSTTYP,F3,F4,F5,OPART,
                         ADRESSE,RESZELL,BELZELL,NAME ,Z5);
                  END;
                FI;
               CALL NAMES(NAME,SONDER);
                IF SONDER EQ 1 THEN BEGIN;
                                     CODNR=3;
                                     GOTO NEW;
                                    END;
                FI;
                /* ZWISCHENSTRING FUER DAS ENDE EINER SPEICHERPLATZ- */
                /* RESERVIERUNG ABLEGEN */
                CODNR=4;
                  CALL PUT8DISK(ANWTYP,CODNR,CONSTTYP,F3,F4,F5,OPART,
                         ADRESSE,RESZELL,BELZELL,NAME,Z5);
             END;
          ELSE
          ERGEBNIS='Ø'B1;
          FI;
    ENDE:;
          RETURN(ERGEBNIS);
    END;

    /*************************************************************************/

    PUT8DISK:PROC (ANWTYP,F1,F2,F3,F4,F5,F6,Z1,Z2,Z3,Z4,Z5) GLOBAL
          DCL (ANWTYP,F1,F2,F3,F4,F5,F6) FIXED;
          DCL (Z1,Z2,Z3,Z4,Z5)(8) CHAR;
          DCL HILFE(4Ø) CHAR;
          DCL I FIXED;
          FOR I TO 8 REPEAT;
            HILFE(I   )=Z1(I);
            HILFE(I+8)=Z2(I);
            HILFE(I+16)=Z3(I);
            HILFE(I+24)=Z4(I);
            HILFE(I+32)=Z5(I);
          END;
          CALL COMPF(ANWTYP,F1,F2,F3,F4,F5,F6,HILFE);
          RETURN;
    END;/*ENDE VON PUTDISK*/
    MODEND;
```

ANHANG III

DOKUMENTATION DER ASSEMBLERTEXTERZEUGUNG FÜR DIE S310

1. Allgemeines zur Dokumentation

Für die Dokumentation der Assemblertexterzeugung gilt analoges wie für die Dokumentation des Vorübersetzers.

Die Dokumentation der PEARL-Module erfolgt ebenfalls in aufsteigender Numerierung (PCØ - PC8).
Ansonsten siehe Anhang II.
Bei der Gliederung der Prozedurbeschreibung fehlt der Punkt FEHLER. In der Assemblertexterzeugung wird nicht mehr auf Fehler geprüft. Um die Assemblertexterzeugung zu verstehen, ist eine genaue Kenntnis des Assembler AS300 und der Zwischeninformation notwendig.

2. Allgemeines zur Assemblertexterzeugung (AS300)

Die Assemblertexterzeugung wurde im PEARL-Subset der ASME-Gruppe programmiert [ASME76] . Als Rechner wurde dabei eine Siemens 306 benutzt. Die AS 300-Texterzeugung benötigt als Massenspeicher ein Platteneinheit. Zur Assemblertexterzeugung ist notwendig, daß auf der Platte bereits folgende Dateien im Lochkartenformat vorliegen.

DOIO:
enthält die Implementierung des organisatorischen Befehls DOIO. (siehe Kapitel 3.4).

SAVE:
enthält die Implementierung des organisatorischen Befehls SAVEREGISTERS (Unterprogrammaufruf, siehe Kapitel 3.4).

RESUME:
enthält die Implementierung des organisatorischen Befehls RESUMEPROCESS. (siehe Kapitel 3.4).

HALT:
enthält die Implementierung des organisatorischen Befehls HALT (siehe Kapitel 3.4).

Diese vier Dateien werden von der Prozedur CODORG in Modul PC7 benutzt.

INIT:

enthält die maschinenabhängige Initialisierung, die maschinenabhängige Auftragsvorverarbeitung und den Programmtext von SAVEREGISTERS.

Diese Datei wird von der Prozedur CODSTA benutzt.

Der erzeugte Assemblertext wird mit Hilfe der Prozedur DISKAUS auf die Datei CODE geschrieben.

3. Beschreibung der einzelnen Module

MODUL SCOD

Dieser Modul enthält nur die Steuertask für die Assemblertexterzeugung für die S310.

Task LAUF2:

Leistung:

Die Task Lauf2 liest eine Zwischeninformationszeile ein und ruft dann die entsprechende Codeprozedur auf. Welche Codeprozedur aufgerufen wird, wird aufgrund des Inhalts des F1-Feldes (siehe Kapitel 4.2.1.2) entschieden. Nachdem für eine Zwischeninformationszeile der entsprechende Assemblertext erzeugt und auf der Datei CODE abgelegt wurde, wird die nächste Zwischeninformationszeile eingelesen und der entsprechende Code erzeugt. So wird fortgefahren, bis das Ende der Zwischeninformationsdatei (Inhalt F1 = 7) erkannt wird.
LAUF2 schreibt die SPASS-Quellzeilen als Kommentarzeilen auf die Datei CODE.

Das Einfügen des Quelltextes war notwendig, um die korrekte Arbeitsweise der Codeprozeduren einfacher testen zu können.
LAUF2 hat keine Fehlerausgänge.

Auf Dateien OBJE und SPRCE Files eröffnen

Wiederhole, bis Dateiende von OBJE erreicht (F1=7)

Ausgeben der Quellzeile, von der das Assembler-Äquivalent erzeugt werden soll, als Kommentarzeile auf die Datei CODE

Code für Startanweisung? (F1=1 ?)
NEIN
JA
Codeprozedur CODSTA aufrufen

Code für Speicherplatz-Reservierung? (F1 = 2 ?)
NEIN
JA
Codeprozedur CODSPA aufrufen

Code für Zuweisung? (F1=3 ?)
NEIN
JA
Codeprozedur CODZUW aufrufen

Code für Sprunganweisung ? (F1=4 ?)
NEIN
JA
Codeprozedur CODSPR aufrufen

Code für Org.-Befehl ? (F1=5 ?)
NEIN
JA
Codeprozedur CODORG aufrufen

Code für Markenvereinbarung ? (F1 = 6 ?)
NEIN
JA
Codeprozedur CODLOC aufrufen

Code für arithmetischen Ausdruck ? (F1=8 ?)
NEIN
JA
Codeprozedur CODARI aufrufen

Code für Prozedurvereinbarung (F1 = 9 ?)
NEIN
JA
Codeprozedur CODPROC aufrufen

Code für Kommentar ? (F1=1Ø ?)
NEIN
JA
Codeprozedur CODCOM aufrufen

Abschlußarbeiten für die Übersetzung

```
 1  MODULE SCOD;
 2  SYSTEM;
 3        SDAU->DRUCKER:;
 4        LKEI<-LESER:;
 5        PSEA<->DISK:;
 6  PROBLEM;
 7  /* SPEZIFIKATION DER IN DIESEM MODUL BENUTZTEN PROZEDUREN */
 8  DCL CODSTART ENTRY GLOBAL;
 9  DCL CODFIN ENTRY GLOBAL;
10  DCL CODSPA ENTRY(FIXED,FIXED,(40)CHAR) GLOBAL;
11  DCL CODZUW ENTRY(FIXED,(40)CHAR) GLOBAL;
12  DCL CODSPR ENTRY (FIXED,FIXED,FIXED,FIXED,FIXED,(40)CHAR) GLOBAL;
13  DCL CODORG ENTRY(FIXED,(40)CHAR) GLOBAL;
14  DCL CODLOC ENTRY((40)CHAR) GLOBAL;
15  DCL CODARI ENTRY(FIXED,FIXED,FIXED,FIXED,(40)CHAR) GLOBAL;
16  DCL CODPROC ENTRY (FIXED,(40)CHAR) GLOBAL;
17  DCL CODCOM ENTRY (FIXED) GLOBAL;
18  DCL (LESER,DRUCKER,DISK) VAL DEVICE GLOBAL;
19  DCL KOMMENTAR ENTRY ((80) CHAR) GLOBAL;
20  DCL DISKAUS ENTRY (FIXED,(80) CHAR) GLOBAL;
21
22  /*************************************************************************
23
24  DCL MERK FIXED GLOBAL INITIAL (Ø);
25
26  /*************************************************************************
27  /* STEUERTASK FUER DIE CODEERZEUGUNG
28  /*************************************************************************
29  LAUF2: TASK GLOBAL RESIDENT
30        DCL (F1,F2,F3,F4,F5,F6,ANWTYP,SATZ) FIXED INITIAL(Ø);
31        DCL (Z1,Z2,Z3,Z4,Z5)(8) CHAR INITIAL(' ');
32        DCL (MODUS,ZNR,QNR) FIXED INITIAL(1);
33        DCL SCRATCH(80) CHAR INITIAL(' ');
34        DCL ZINFO VAL FILE;
35        DCL QINFO VAL FILE;
36        DCL HILF1(40) CHAR;
37        DCL I FIXED;
38        DCL IV(4) VAL CHAR IDENTICAL('''','V','A','''');
39        DCL KENN FIXED;
40        DCL GØ(4) VAL CHAR IDENTICAL ('G','Ø',':','=');
41       /* AUF DEN DATEIEN OBJE UND 'SRCE' FILES ERÖFFNEN */
42        QNR=Ø;
43        MODUS=Ø;
44        CALL DISKAUS(MODUS,SCRATCH);
45        OPEN ZINFO TITLE 'OBJE' UPON DISK INPUT DIR (80) ALPHA;
46        OPEN QINFO TITLE 'SRCE' UPON DISK INPUT DIR (80) ALPHA;
47        MODUS=1;
48        QNR=Ø;
49        /* WIEDERHOLE BIS DATEIENDE VON 'OBJE' ERREICHT */
50  NEW:  GET ZINFO POS(ZNR) EDIT
51              (ANWTYP,F1,F2,F3,F4,F5,F6,Z1,Z2,Z3,Z4,Z5,SATZ)
52              ((7)F(2),(8)A(1),(8)A(1),(8)A(1),(8)A(1),(8)A(1),(5)X,F(3
53        FOR I TO 8 REPEAT;
54          HILF1(I)=Z1(I);
55          HILF1(I+8)=Z2(I);
56          HILF1(I+16)=Z3(I);
57          HILF1(I+24)=Z4(I);
58          HILF1(I+32)=Z5(I);
59        END;
60        ZNR=ZNR+1;
```

```
/* AUSGEBEN DER QUELLZEILE, VON DER DAS ASSEMBLERAEQUIVALENT */
/* ERZEUGT WERDEN SOLL,ALS KOMMENTARZEILE AUF DIE DATEI 'CODE' *
IF SATZ GT (QNR+1) THEN
  BEGIN;
FOR I FROM QNR TO (SATZ-1) REPEAT;
       QNR=QNR+1;
       GET QINFO POS(QNR) EDIT (SCRATCH)((80)A(1));
       CALL KOMMENTAR(SCRATCH);
    END;
  END;
FI;
  IF SATZ EQ (QNR+1) THEN
  BEGIN;
     QNR=QNR+1;
     GET QINFO POS(QNR) EDIT (SCRATCH)((80)A(1));
     CALL KOMMENTAR(SCRATCH);
  END;
FI;
/* CODE FUER STARTANWEISUNG$ */
IF ANWTYP EQ 1 THEN CALL CODSTART;FI;
/* CODE FUER SPEICHERPLATZRESERVIERUNG$ */
IF ANWTYP EQ 2 THEN CALL CODSPA(F1,F2,HILF1);FI;
/* CODE FUER ZUWEISUNG$ */
IF ANWTYP EQ 3 THEN CALL CODZUW(F1,HILF1);FI;
/* CODE FUER SPRUNGANWEISUNG$ */
IF ANWTYP EQ 4 THEN CALL CODSPR(F1,F2,F3,F4,F5,HILF1);FI;
/* CODE FUER ORG. BEFEHL$ */
IF ANWTYP EQ 5 THEN CALL CODORG(F1,HILF1);FI;
/* CODE FUER MARKENVEREINBARUNG$ */
IF ANWTYP EQ 6 THEN CALL CODLOC(HILF1);FI;
/* CODE FUER ARITH. AUSDRUCK$ */
IF ANWTYP EQ 8 THEN CALL CODARI(F1,F2,F3,F4,HILF1);FI;
/* CODE FUER PROZEDURVEREINBARUNG$ */
IF ANWTYP EQ 9 THEN CALL CODPROC(F1,HILF1);FI;
/* CODE FUER KOMMENTARZEILEN */
IF ANWTYP EQ 10 THEN CALL CODCOM(SATZ);FI;
IF ANWTYP EQ 7 THEN GOTO READY;FI;
GOTO NEW;
READY:;
/* ABSCHLUSSARBEITEN */
  BEGIN;
     CALL CODFIN;
     MODUS=2;
     CALL DISKAUS(MODUS,SCRATCH);
     CLOSE ZINFO;
     CLOSE QINFO;
  END;
END;/*ENDE DER TASK*/
MODEND;
```

MODUL PCØ
=========

Dieser Modul enthält einige Hilfsprozeduren für die Codeprozeduren.

CODOPE:

Leistung:
Diese Prozedur erzeugt mit Hilfe der übergebenen Parameter den AS 300 Code für die verschiedenen Operandentypen von SPASS.

Parameter:
HILFE:
36-elementiges Zeichenfeld.
Dieses Feld ist eine Zusammenfassung mehrerer Zeichenfelder, wegen des in Kapitel 4.2.1.1. erwähnten PEARL-Compilerfehlers.
Die Feldelemente HILFE(20) bis HILFE(27) enthalten den Namen des Operanden (Adresse, Konstante usw. siehe Kapitel 4.2.1.2.) und die Feldelemente HILFE(28) bis HILFE(36) einen etwa vorhanden Offset (siehe Kapitel 4.2.1.2.).
In den Feldelementen HILFE(1) bis HILFE(20) wird an die aufrufende Stelle der Assemblercode für den Operanden zurückgegeben.

TYP: Integer Größe
Dieser Parameter enthält die Kennziffer für den Typ des Operanden, dessen Assembleräquivalent erzeugt werden soll. (Kennziffern siehe Kapitel 4.2.1.2.).

Aufrufende Prozeduren:	aus Modul:
CODARI	PC
CODZUW	PC

Aufgerufene Prozeduren:	in Modul:
BITINT	PL Ø

BITINT:

Leistung:

Diese Prozedur wandelt Bitzahlen (in Zeichendarstellung) in Integerzahlen (in Zeichendarstellung) um. Es ist möglich, maximal eine 8-stellige Bitzahl zu konvertieren.

Parameter:

HILFE: 16-elementiges Zeichenfeld.
Dieser Parameter ist eine Zusammenfassung von zwei achtelementigen Zeichenfeldern (wegen Compilerfehler). Die Feldelemente HILFE(1) bis HILFE(8) enthalten die zu konvertierende Bitzahl. In den Feldelementen HILFE(9) bis HILFE(16) wird an die aufrufende Stelle das Ergebnis (Integerzahl) zurückgegeben. Das Ergebnis steht dabei in den Feldelementen HILFE(9) bis HILFE(11), es müssen alle drei Feldelemente beachtet werden, da das Ergebnis in diesen drei Elementen rechtsbündig steht.

Aufgerufene Prozeduren:	in Modul:
RKONVERT	PCØ

Aufrufende Prozeduren:	aus Modul:
CODOPE	PCØ

KOMMENTAR:

Leistung:

Diese Prozedur fügt an den übergebenen aktuellen Parameter führende '***' an, d.h. der aktuelle Parameter wird um 3 Feldelemente nach rechts verschoben und die Feldelemente eins bis drei mit '*' aufgefüllt, dabei gehen die Feldelemente 78 bis 80 verloren. Mit '***' wird im AS300 der Beginn einer Kommentarzeile gekennzeichnet.

Parameter:

SCRATCH: 80-elementiges Zeichenfeld
In diesem Feld erhält die Prozedur die Zeichenkette, an die vorne die Sterne eingefügt werden sollen. In diesem Feld wird auch an die aufrufende Stelle der geänderte Textstring zurückgegeben.

Aufrufende Prozeduren:	aus Modul:
Task LAUF2	SCOD

Aufgerufene Prozeduren:	in Modul:
keine	

DISKAUS:

Leistung:

Diese Prozedur verwaltet die Datei CODE. Durch verschiedene Kennziffern kann die Prozedur veranlaßt werden, die Datei CODE einzurichten und darauf das File ASS zu einzurichten, die Datei CODE zu schließen oder auf CODE zu schreiben.

Parameter:

KENN Integerzahl

Dieser Parameter enthält die Auftragskennung für die Prozedur

- Ø: Einrichten der Datei CODE und darauf den File ASS eröffnen
- 1: Inhalt des Parameters SCRATCH auf die Datei schreiben. Die Datei wird dabei sequentiell beschrieben.
- 2: File ASS löschen und Datei CODE schließen.

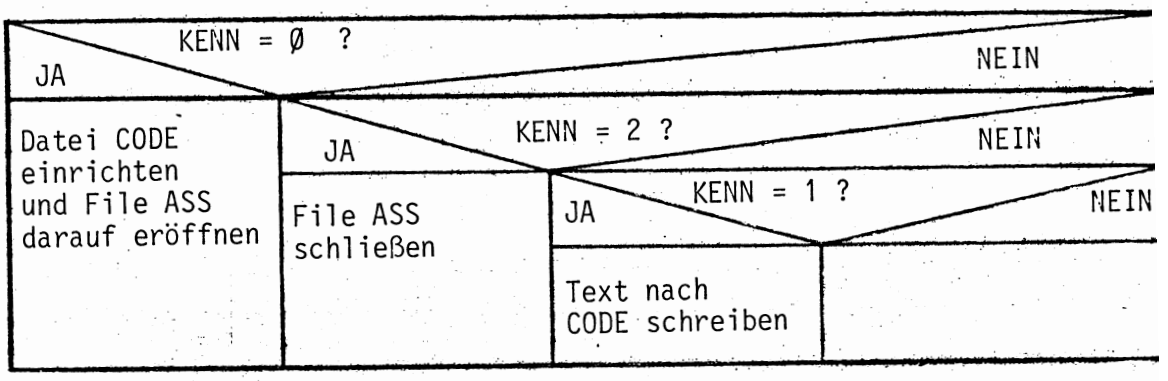
Andere Kennziffern haben keine Wirkung.

SCRATCH: 80-elementiges Zeichenfeld.

Dieser Parameter enthält die nach CODE zu schreibende Zeichenfolge, falls kein KENN gleich eins ist. Ansonsten ist SCRATCH bedeutungslos.

Aufrufende Prozeduren:	aus Modul:
CODFINISH	PC1
CODSTART	PC1
CODSPA	PC2
CODSPR	PC3
CODLAB	PC3
CODZUW	PC4
CODARI	PC5
CODPRO	PC6
CODORG	PC7
CODLOC	PC8

Aufgerufene Prozeduren:	in Modul:
keine	


KENN = Ø ?
JA
NEIN
Datei CODE einrichten und File ASS darauf eröffnen
KENN = 2 ?
JA
NEIN
File ASS schließen
KENN = 1 ?
JA
NEIN
Text nach CODE schreiben

```
1    MODULE PC0;
2    SYSTEM;
3          SDAU->DRUCKER:;
4          LKEI<-LESER:;
5    PSEA<->DISK:;
6    PROBLEM;
7    DCL DISK VAL DEVICE GLOBAL;
8    DCL (LESER,DRUCKER) VAL DEVICE GLOBAL;
9
10   /*************************************************************************
11   /* 'CODOPE' ERZEUGT DEN ASS-320 TEXT FUER DIE VERSCHIEDENEN              *
12   /* OPERANDENTYPEN                                                        *
13   /*************************************************************************
14   CODOPE:PROC (HILFE,TYP) GLOBAL
15         DCL HILFE(36) CHAR;
16         DCL TYP FIXED;
17         DCL STRING(20) CHAR;
18         DCL (NAME,OFSET)(8) CHAR;
19         DCL SCRATCH(80) CHAR INITIAL(' ');
20         DCL IA(4) VAL CHAR IDENTICAL ('''','V','A','''');
21         DCL WTL(5) VAL CHAR IDENTICAL ('(','W','T','L','+');
22         DCL R7(4) VAL CHAR IDENTICAL('R','7',':','=');
23         DCL I FIXED;
24         DCL MODUS FIXED;
25         DCL HILF1(16) CHAR;
26         DCL (ERGEBNIS,ZAHL) FIXED;
27         FOR I TO 8 REPEAT;
28           NAME(I)=HILFE(I+20);
29           OFSET(I)=HILFE(I+28);
30         END;
31   FOR I TO 20 REPEAT;
32     STRING(I)=' ';
33   END;
34                   IF TYP EQ 16 THEN
35                     BEGIN;
36                        FOR I TO 8 REPEAT;
37                          HILF1(I)=NAME(I);
38                        END;
39                        CALL BITINT(HILF1);
40                        FOR I TO 8 REPEAT
41                          STRING(I)=HILF1(I+8);
42                        END;
43                     END;
44                   FI;
45         IF TYP EQ 14 THEN
46           BEGIN;
47              STRING(1)='4';
48              STRING(12)=SYMBOL(14);
49              FOR I TO 8 REPEAT;
50                STRING(I+1)=NAME(I);
51              END;
52           END;
53         FI;
54         IF (TYP LE 13) OR (TYP EQ 15) THEN
55           BEGIN;
56              FOR I TO 8 REPEAT;
57                STRING(I)=NAME(I);
58              END;
59           END;
60         FI;
```

```
61          IF ((TYP GT 20) AND (TYP LT 30)) THEN
62            BEGIN;
63               STRING(1)='G';
64               STRING(2)=NAME(2);
65            END;
66          FI;
67          IF TYP GT 30 THEN
68            BEGIN;
69               IF TYP EQ 31 THEN
70                 BEGIN;
71                    STRING(1)='(';
72                    FOR I TO 8 REPEAT;
73                      STRING(I+1)=NAME(I);
74                    END;
75                    STRING(10)=')';
76                 END;
77               FI;
78            END;
79          FI;
80          IF TYP EQ 32 THEN
81            BEGIN;
82               STRING(1)='(';
83               FOR I TO 4 REPEAT;
84                 STRING(I+1)=OFSET(I);
85                 STRING(I+6)=WTL(I+1);
86               END;
87               STRING(11)='G';
88               STRING(12)=NAME(2);
89               STRING(13)=')';
90            END;
91          FI;
92          IF TYP EQ 34 THEN
93            BEGIN;
94               STRING(1)='(';
95               STRING(2)='G';
96               STRING(3)=NAME(2);
97               STRING(4)=')';
98            END;
99          FI;
100         IF TYP EQ 33 THEN
101           BEGIN;
102              STRING(1)='(';
103              FOR I TO 8 REPEAT;
104                STRING(I+1)=NAME(I);
105              END;
106              STRING(10)=')';
107           END;
108         FI;
109   FIN:;
110         FOR I TO 20 REPEAT;
111           HILFE(I)=STRING(I);
112         END;
113         RETURN;
114   END;/*ENDE VON COD OPERAND*/
115
116   /*************************************************************************
117   /* 'BITINT' KONVERTIERT BINAERZAHLEN IN INTEGERZAHLEN
118   /*************************************************************************
119   BITINT: PROC (HILFE) GLOBAL
120         DCL HILFE(16) CHAR;
```

```
121         DCL BIT(8) CHAR;
122         DCL INT(8) CHAR;
123         DCL I FIXED;
124         DCL K FIXED;
125         DCL (ERGEBNIS,ZAHL) FIXED INITIAL(0);
126         K=0;
127         FOR I TO 8 REPEAT;
128           BIT(I)=HILFE(I);
129         END;
130         FOR I FROM 8 BY (-1) TO 1 REPEAT;
131           IF BIT(I) NE ' ' THEN
132              BEGIN;
133                   IF BIT(I) EQ '0' THEN ZAHL=0; ELSE ZAHL=1;FI;
134                   ERGEBNIS=ERGEBNIS+ZAHL*2**K;
135                   K=K+1;
136              END;
137           FI;
138         END;
139         CALL RKONVERT(ERGEBNIS,INT);
140         FOR I TO 8 REPEAT;
141           HILFE(I+8)=INT(I);
142         END;
143         RETURN;
144   END;/*ENDE VON BITINT*/
145
146   /*************************************************************************
147   /* 'KOMMENTAR' WANDELT EINE SPASS-QUELLZEILE IN EINE ASS-310          *
148   /* KOMMENTARZEILE UM.                                                 *
149   /*************************************************************************
150   KOMMENTAR: PROC (SCRATCH) GLOBAL
151         DCL SCRATCH(80) CHAR;
152         DCL I FIXED;
153         DCL KOMM(80) CHAR INITIAL ('*');
154         DCL MODUS FIXED INITIAL(1);
155         FOR I TO 77 REPEAT;
156           KOMM(I+3)=SCRATCH(I);
157         END;
158         CALL DISKAUS(MODUS,KOMM);
159         RETURN;
160   END;/*ENDE VON KOMMENTAR*/
161
162   /*************************************************************************/
163   /* 'DISKAUS VERWALTET DIE DATEI 'CODE'                                   */
164   /*************************************************************************/
165   DISKAUS: PROC (KENN,SCRATCH) GLOBAL
166         DCL KENN FIXED;
167         DCL SCRATCH(80) CHAR;
168         DCL ASS VAL FILE;
169         DCL ZEICHEN(8) CHAR INITIAL(' ');
170         /* DATEI ERÖFFNEN    */
171         IF KENN EQ 0 THEN
172         BEGIN;
173           CREATE ASS TITLE 'CODE' UPON DISK OUTPUT SEQ (80) ALPHA;
174           GOTO ENDE;
175         END;
176         FI;
177         /* DATEI SCHLIESSEN */
178         IF KENN EQ 2 THEN
179           BEGIN;
180              CLOSE ASS;
```

```
181                GOTO ENDE;
182              END;
183           FI;
184           /* DATEI BESCHREIBEN */
185           PUT ASS EDIT (SCRATCH)((80)A(1));
186           ENDE:;
187           RETURN;
188     END;/*ENDE VON DISKAUS*/
189     /*****************************************************************
190
191     MODEND;
```

MODUL PC1

Dieser Modul enthält die Prozeduren zum Erzeugen des Codes für den Beginn bzw. das Ende eines Programms.
Bei dem Code für den Start eines Programms wird auch der Code für die anlagenabhängige Initialisierung und Vorverarbeitung erzeugt.

CODESTART:

Leistung:

Diese Prozedur erzeugt zunächst den Code für den Beginn eines AS300 Programms. Dann wird der Code für die anlagenabhängige Initialisierung und Vorverarbeitung von der Hilfsdatei INIT auf die Datei CODE kopiert.

Parameter:

keine Parameter

Aufrufende Prozeduren:	aus Modul:
Task LAUF2	SCOD

Aufgerufende Prozeduren:	in Modul:
DISKAUS	PCØ

CODFINISH

Leistung:

Diese Prozedur erzeugt den Code für das Ende eines AS300 Programms.

Parameter:

keine Parameter

Aufrufende Prozeduren:	aus Modul:
TASK LAUF2	SCOD

Aufgerufene Prozeduren:	in Modul:
DISKAUS	PCØ

```
MODULE PC1;
SYSTEM;
      SDAU->DRUCKER;;
      PSEA<->DISK;;
PROBLEM;
DCL DRUCKER VAL DEVICE GLOBAL;
      DCL DISK VAL DEVICE GLOBAL;
DCL DISKAUS ENTRY (FIXED,(80)CHAR) GLOBAL;

/*****************************************************************
/* 'CODFINISH' ERZEUGT DEN ASS-310 TEXT FUER DAS PROGRAMMENDE
/*****************************************************************
CODFINISH:PROC
GLOBAL
      DCL SCRATCH (80)  CHAR  INITIAL(' ');
      DCL ENDE  CHAR(6)  INITIAL ('''ENDE''');
      DCL I  FIXED ;
      DCL MODUS FIXED;
      FOR I TO 6 REPEAT;
           SCRATCH(10+I)=I CHAR ENDE;
      END;/*ENDE DER VORBESETZUNGSSCHLEIFE*/
      MODUS=1;
      CALL DISKAUS(MODUS,SCRATCH);
END;/*ENDE VON CODFINISH*/

/*****************************************************************
/* 'CODSTART' ERZEUGT DEN ASS-310 TEXT FUER DEN PROGRAMMANFANG
/*****************************************************************
CODSTART:PROC
GLOBAL
      DCL SCRATCH (80)  CHAR  INITIAL(' ');
      DCL START  CHAR(11)  INITIAL ('''NAME'' OBJE');
      DCL SATZ CHAR(11) INITIAL ('''SATZ'' OBJS');
      DCL I  FIXED ;
      DCL MODUS FIXED;
      DCL FINI VAL FILE;
      MODUS=1;
      FOR I TO 11 REPEAT;
           SCRATCH(10+I)=I CHAR START;
      END;/*ENDE DER VORBESETZUNGSSCHLEIFE*/
      CALL DISKAUS(MODUS,SCRATCH);
      FOR I TO 11 REPEAT;
        SCRATCH(10+I)=I CHAR SATZ;
      END;
      CALL DISKAUS(MODUS,SCRATCH);
      /* EINKOPIEREN DER HILFSDATEI 'INIT' */
      I=1;
      OPEN FINI TITLE 'INIT' UPON DISK INPUT DIR (80) ALPHA;
  NEW:GET FINI POS(I) EDIT (SCRATCH) ((80)A(1));
      IF SCRATCH(1) NE '/' THEN
        BEGIN;
           CALL DISKAUS(MODUS,SCRATCH);
           I=I+1;
           GOTO NEW;
        END;
      FI;
      CLOSE FINI;
      RETURN;
END;/*ENDE VON CODSTART*/
MODEND;
```

MODUL PC2
=========

Dieser Modul enthält die Prozedur für das Erzeugen des Assemblercodes der Speicherreservierung.

CODSPA

Leistung:

Diese Prozedur erzeugt den Assemblercode für die Speicherreservierung. Mit Hilfe des Parameters CODNR wird festgestellt, welcher Typ der Speicherreservierung (siehe Kapitel 4.2.1.2.) vorliegt und dann werden die entsprechenden Codezeilen erzeugt, die mit Hilfe der Prozedur DISKAUS auf die Datei CODE geschrieben werden.
Bei einer Speicherplatzreservierung mit Vorbesetzung mit einer Bitkonstanten wird diese in eine Integergröße umgewandelt.

Parameter:

CODNR: Integergröße
Dieser Parameter enthält die Kennziffer des Typs der Speicherreservierung.
Die Bedeutung der Kennummern siehe Kapitel 4.2.1.2.

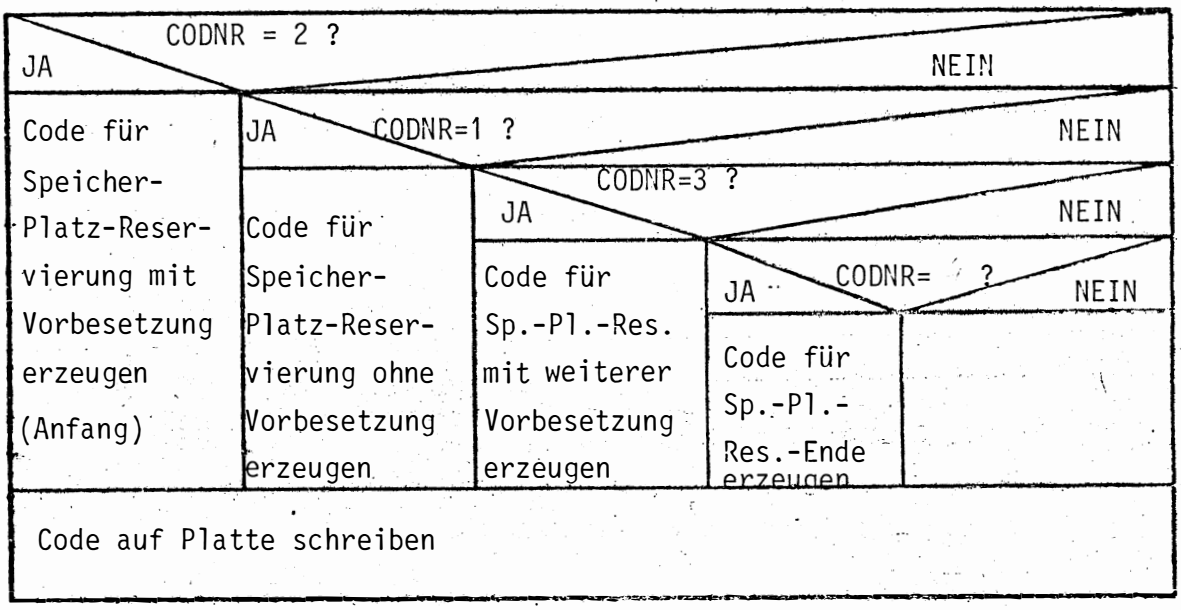
CONSTTYP:
Liegt eine Speicherplatzreservierung mit Vorbesetzung vor, enthält dieser Parameter die Kennummer des Typs der Konstanten, mit der eine Zelle vorbesetzt werden soll.
Die Bedeutung der Kennziffern siehe Kapitel 4.2.1.2.

HILF1: 40-elementiges Zeichenfeld
Dieses Feld ist eine Zusammenfassung der Felder Z1-Z5 des Zwischenstrings (Compilerfehler).
HILF1(1)-HILF1(8)
Diese Feldelemente enthalten die Anfangsadresse des zu reservierenden Speicherbereichs.
HILF1(9)-HILF1(16)
Diese Feldelemente enthalten die Anzahl der zu reservierenden Speicherzellen in Zeichendarstellung, und zwar in den Feldelementen HILF1(9)-HILF1(10).
Führende Nullen können dabei auftreten.
HILF1(17)-HILF1(24)
Diese Feldelemente enthalten die Anzahl der Speicherzellen, die mit einer Konstanten belegt werden sollen. Diese Zahl ist ebenfalls als Zeichenfolge dargestellt und in diesem Teilfeld mit HILF1 linksbündig abgelegt.

HILF1(25)-HILF1(32)
Sollen Speicherplätze mit einer Konstanten vorbesetzt werden, enthalten diese Feldelemente diese Konstante linksbündig in Zeichendarstellung (z.B. Name einer Adresse, Zahl in Zifferndarstellung mit Vorzeichen).

Aufgerufene Prozeduren:	in Modul:
BITINT	PCØ
DISKAUS	PCØ

Aufrufende Prozeduren:	aus Modul:
Task LAUF2	$COD


CODNR = 2 ?
JA
NEIN
Code für Speicher-Platz-Reservierung mit Vorbesetzung erzeugen (Anfang)
JA
CODNR=1 ?
NEIN
Code für Speicher-Platz-Reservierung ohne Vorbesetzung erzeugen
CODNR=3 ?
JA
NEIN
Code für Sp.-Pl.-Res. mit weiterer Vorbesetzung erzeugen
JA
CODNR= ?
NEIN
Code für Sp.-Pl.-Res.-Ende erzeugen
Code auf Platte schreiben

```
 1  MODULE PC2;
 2  SYSTEM;
 3        SDAU→DRUCKER;;
 4  PROBLEM;
 5  DCL DRUCKER VAL DEVICE GLOBAL;
 6  DCL DISKAUS ENTRY (FIXED,(80) CHAR) GLOBAL;
 7  /* SPEZIFIKATION DER IN DIESEM MODUL BENUTZTEN PROZEDUREN */
 8  DCL BITINT ENTRY ((16) CHAR) GLOBAL;
 9  /*****************************************************************
10  /*'CODSPA' ERZEUGT DEN ASS-310 TEXT FUER DIE SPEICHERPLATZRESERVIERU
11  /*****************************************************************
12  CODSPA: PROC (CODNR,CONSTTYP,HILF1) GLOBAL
13        DCL (CODNR,CONSTTYP) FIXED;
14        DCL HILF1(40) CHAR;
15        DCL NAME(8) CHAR;
16        DCL RESZELL(8) CHAR;
17        DCL BELZELL(8) CHAR;
18        DCL CONST(8) CHAR;
19        DCL SCRATCH(80) CHAR INITIAL(' ');
20        DCL HILFE(16) CHAR INITIAL(' ');
21        DCL AP CHAR(15) INITIAL ('''AP''/   ''IDENT''');
22        DCL (I,K) FIXED;
23        DCL GROESSER CHAR;
24        DCL MODUS FIXED;
25        MODUS=1;
26        FOR I TO 8 REPEAT;
27          NAME(I)=HILF1(I);
28          RESZELL(I)=HILF1(I+8);
29          BELZELL(I)=HILF1(I+16);
30          CONST(I)=HILF1(I+24);
31        END;
32        SCRATCH(1)='''';
33        SCRATCH(2)='V';
34        SCRATCH(3)='A';
35        SCRATCH(4)='''';
36        GROESSER=SYMBOL(14);
37        /* SPEICHERPLATZRESERVIERUNG OHNE VORBESETZUNG$ */
38        IF CODNR EQ 1 THEN
39             BEGIN;
40             FOR I FROM 6 TO 13 REPEAT;
41                  SCRATCH(I)=NAME(I-5);
42             END;
43             SCRATCH(15)='/';
44             SCRATCH(16)='=';
45             I=20;
46             FOR K FROM I TO (I+2) REPEAT;
47                  SCRATCH(K)=RESZELL(K-I+1);
48             END;
49             I=I+3;
50             SCRATCH(I+1)='+';
51             SCRATCH(I+2)=GROESSER;
52             CALL DISKAUS(MODUS,SCRATCH);
53             FOR I TO 80 REPEAT;
54                  SCRATCH(I)=' ';
55             END;
56          END;
57        FI;
58        /* SPEICHERPLATZRESERVIERUNG MIT VORBESETZUNG (ANFANG) $ */
59        IF CODNR EQ 2 THEN
60          BEGIN;
```

```
61              FOR I FROM 6 TO 13 REPEAT
62                   SCRATCH(I)=NAME(I-5);
63              END;
64              SCRATCH(15)='/';
65              SCRATCH(16)='=';
66              I=20;
67              FOR K FROM I TO (I+2) REPEAT;
68                   SCRATCH(K)=BELZELL(K-I+1);
69              END;
70              I=I+4;
71              SCRATCH(I)='↑';
72              I=I+1;
73              IF CONSTTYP EQ 16 THEN
74                BEGIN;
75                   FOR I TO 8 REPEAT;
76                     HILFE(I)=CONST(I);
77                   END;
78                   CALL BITINT(HILFE);
79                   FOR I TO 8 REPEAT;
80                      CONST(I)=HILFE(I+8);
81                   END;
82                END;
83              FI;
84              FOR K FROM I TO I+7 REPEAT;
85                   SCRATCH(K)=CONST(K-I+1);
86              END;
87              SCRATCH(I+8)=GROESSER;
88              CALL DISKAUS(MODUS,SCRATCH);
89              FOR I TO 80 REPEAT;
90                   SCRATCH(I)= ' ';
91              END;
92              I=16;
93            END;
94          FI;
95          /* SPEICHERPLATZRESERVIERUNG MIT VORBESETZUNG */
96          /* (WEITERE VORBESETZUNGEN ) */
97          IF CODNR EQ 3 THEN
98            BEGIN;
99               SCRATCH(16)='=';
100              FOR I FROM 20 TO 22 REPEAT;
101                  SCRATCH(I)=BELZELL(I-19);
102              END;
103              SCRATCH(23)='↑';
104              IF CONSTTYP EQ 16 THEN
105                BEGIN;
106                   FOR I TO 8 REPEAT;
107                     HILFE(I)=CONST(I);
108                   END;
109                   CALL BITINT(HILFE);
110                   FOR I TO 8 REPEAT;
111                      CONST(I)=HILFE(I+8);
112                   END;
113                END;
114             FI;
115             FOR I FROM 24 TO 30 REPEAT;
116                  SCRATCH(I)=CONST(I-23);
117             END;
118             SCRATCH(31)=GROESSER;
119             CALL DISKAUS(MODUS,SCRATCH);
120             FOR I TO 80 REPEAT;
```

```
121                     SCRATCH(I)=' ';
122                 END;
123               END;
124             FI;
125             IF CODNR EQ 4 THEN
126               BEGIN;
127             I=6;
128                 FOR K FROM I TO 20 REPEAT;
129                     SCRATCH(K)=(K-I+1) CHAR AP;
130                 END;
131                 FOR K FROM 22 TO 29 REPEAT;
132                     SCRATCH(K)=NAME(K-21);
133                 END;
134                 SCRATCH(28)='+';
135                 FOR I FROM 29 TO 31 REPEAT;
136                     SCRATCH(I)=RESZELL(I-28);
137                 END;
138                 CALL DISKAUS(MODUS,SCRATCH);
139               END;
140             FI;
141             RETURN;
142     END;
143     MODEND;
```

MODUL PC3
=========

Dieser Modul enthält die Prozedur für die Erzeugung des Codes der verschiedenen Sprunganweisungen, sowie die lokale Hilfsprozedur für die Codeerzeugung der Sprungzielangabe.

CODSPR:

Leistung:

Mit Hilfe des Parameter START wird an die entsprechende Stelle in der Prozedur verzweigt, die den Code für den angegebenen Sprungtyp erzeugt (Bedeutung der Kennziffern siehe Kapitel 4.2.1.2).

Die erzeugten Assemblerzeilen werden auf der Datei CODE mit Hilfe der Prozedur DISKAUS abgelegt.
Bei einem bedingten Sprung wird erwartet, daß der Inhalt des Registers GØ dem Operanden links vom Vergleichsoperator entspricht (siehe auch CODARI und CODZUW).

Parameter:

START: Integergröße
Dieser Parameter enthält die Kennziffer des Sprungtyps (siehe Kapitel 4.2.1.2)

LTYP: Integergröße
Dieser Parameter enthält die Kennziffer des Typs der Sprungzielangabe.

WAHRH: Integergröße
Bei bedingte Sprüngen gibt dieser Parameter an, ob bei erfüllter oder bei nicht erfüllter Bedingung gesprungen werden soll (siehe Kapitel 4.2.1.2.)

OPERATOR: Integergröße
Bei bedingten Sprüngen enthält dieser Parameter den Typ des Vergleichsoperators (siehe Kapitel 4.2.1.2.) bzw. des Selektionsoperators.

OPTYP: Integergröße
Bei einem bedingten Sprung enthält dieser Parameter die Kennziffer des Operanden rechts vom Vergleichsoperator (siehe Kapitel 4.2.1.2).

HILF1: 40-elementiges Zeichenfeld.
Dieses Feld ist ebenfalls eine Zusammenfassung von fünf 8-elementigen Zeichenfeldern des Zwischenstrings.

HILF1(1)-HILF1(8)
Diese Feldelemente enthalten entweder den Namen des Sprungziels (direkte Zielangabe) oder wo die Sprungzieladresse steht (indirekte Sprungzielangabe).
Bei einem Unterprogrammsprung enthalten diese Feldelemente den Namen der aufzurufenden Prozedur. Bei einem Rücksprung aus einer Prozedur enthalten diese Feldelemente den Namen der Prozedur aus der zurückgesprungen wird. (siehe auch Beschreibung von CODPRO).
Wird an den Prozedurnamen vorne ein 'Q' angefügt, so erhält man die Speicherzelle, die die Rücksprungadresse enthält.

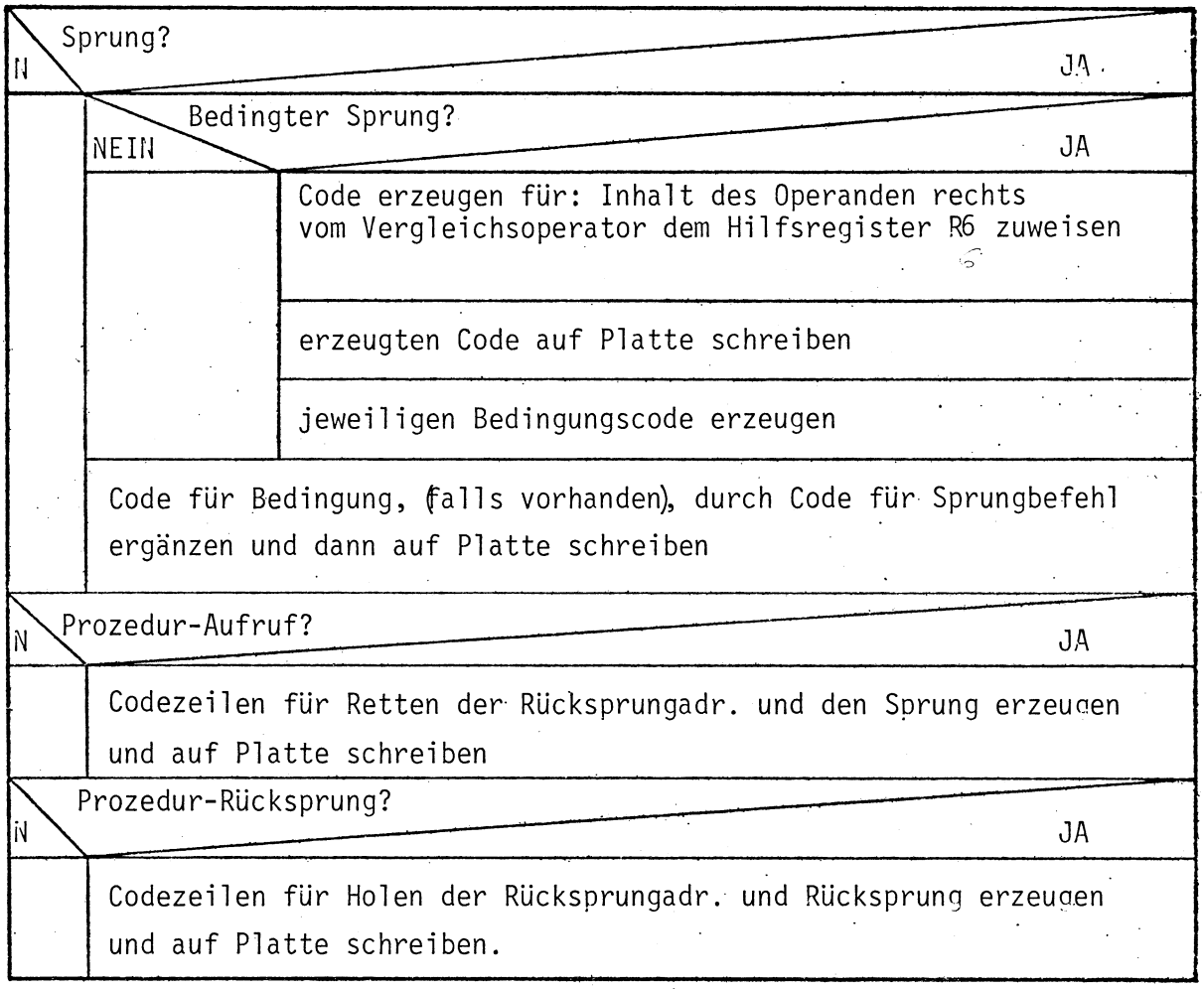
HILF1(9)-HILF1(16)
Diese Feldelemente enthalten den Namen des Operanden rechts vom Vergleichsoperator.
HILF1(17)-HILF1(24)
Diese Feldelemente enthalten einen eventuell vorhandenen Offset des in HILF1(9)-HILF1(16) angegebenen Operanden (Zahl in Zeichendarstellung).

HILF1(25)-HILF1(32)
Bei einem bedingten Sprung, der mit Hilfe einer Bitselektion entschieden wird, enthalten diese Feldelemente die Nummer des Bits aus dem Selektionsoperanden, das zur Entscheidung herangezogen wird.

HILF(33)-HILF(40) sind bedeutungslos .

Aufgerufene Prozeduren:	an Modul:
CODOPE	PCØ
DISKAUS	PCØ

Aufrufende Prozeduren:	in Modul:
Task LAUF 2	SCOD


Sprung?
N
JA
Bedingter Sprung?
NEIN
JA
Code erzeugen für: Inhalt des Operanden rechts vom Vergleichsoperator dem Hilfsregister R6 zuweisen
erzeugten Code auf Platte schreiben
jeweiligen Bedingungscode erzeugen
Code für Bedingung, (falls vorhanden), durch Code für Sprungbefehl ergänzen und dann auf Platte schreiben
Prozedur-Aufruf?
N
JA
Codezeilen für Retten der Rücksprungadr. und den Sprung erzeugen und auf Platte schreiben
Prozedur-Rücksprung?
N
JA
Codezeilen für Holen der Rücksprungadr. und Rücksprung erzeugen und auf Platte schreiben.

CODLAB:

Leistung:

Diese Prozedur ist eine Hilfsprozedur für CODSPR. CODLAB erzeugt den Code für die Sprungzielangabe (indirekt, direkt). Der erzeugte Code wird an CODSPR zurückgegeben, wo er weiterverarbeitet wird.

Parameter:

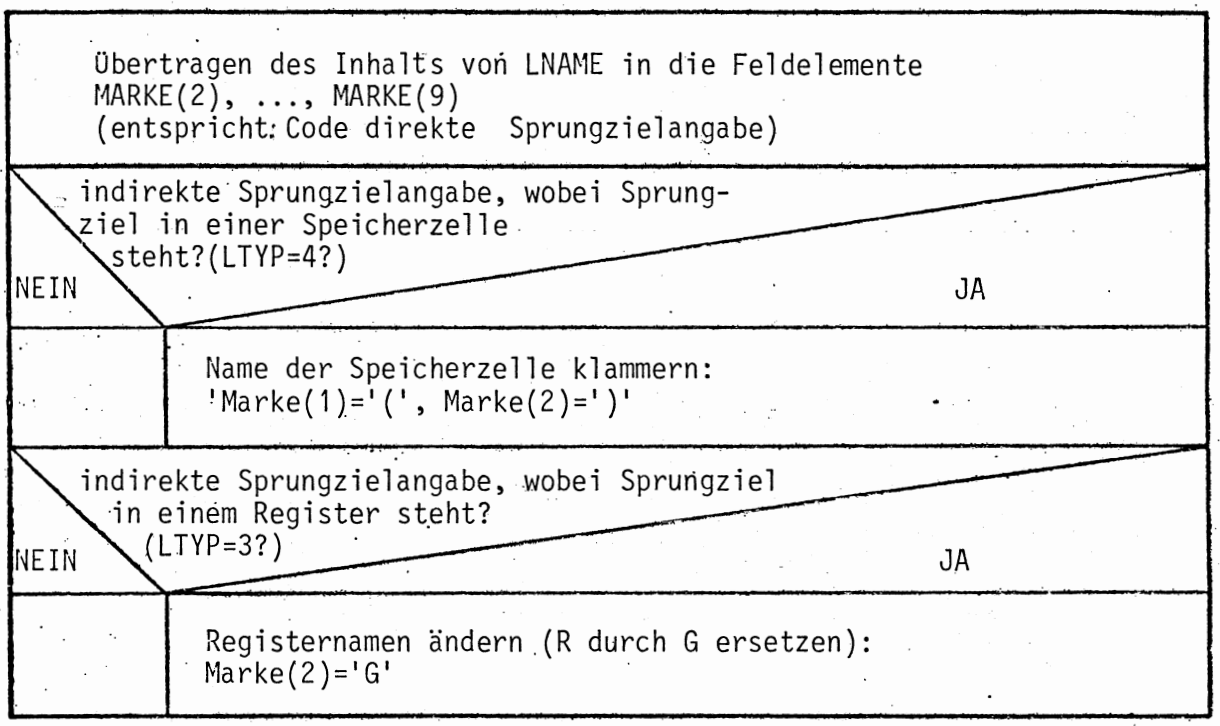
MARKE: 10-elementiges Zeichenfeld.
In diesem Parameter wid an die aufrufende Stelle der Code für die Sprungzielangabe zurückgegeben.

LTYP: Integergröße
Dieser Parameter enthält die Kennziffer des Typs der Sprungzielangabe (siehe Kapitel 4.2.1.2.)

LNAME: 8-elementiges Zeichenfeld
In diesem Feld erhält CODLAB den Namen des Sprungziels bzw. den Namen der Zelle, wo das Sprungziel steht (je nach Inhalt von LTYP).

Aufgerufene Prozeduren:	in Modul:
keine	

Aufrufende Prozeduren:	aus Modul:
CODSPR	PC3


Übertragen des Inhalts von LNAME in die Feldelemente
MARKE(2), ..., MARKE(9)
(entspricht Code direkte Sprungzielangabe)
indirekte Sprungzielangabe, wobei Sprung-
ziel in einer Speicherzelle
steht?(LTYP=4?)
NEIN
JA
Name der Speicherzelle klammern:
'Marke(1)='(', Marke(2)=')'
indirekte Sprungzielangabe, wobei Sprungziel
in einem Register steht?
(LTYP=3?)
NEIN
JA
Registernamen ändern (R durch G ersetzen):
Marke(2)='G'

```
MODULE PC3;
SYSTEM;
      SDAU→DRUCKER:;
PROBLEM;
/* SPEZIFIKATION DER IN DIESEM MODUL VERWENDETEN PROZEDUREN */
DCL CODOPE ENTRY ((36) CHAR,FIXED) GLOBAL;
DCL DRUCKER VAL DEVICE GLOBAL;
DCL DISKAUS ENTRY (FIXED,(80) CHAR) GLOBAL;

/*************************************************************************
/* 'CODSPR' ERZEUGT DEN ASS-310 TEXT FUER DIE VERSCHIEDENEN
/* SPRUNGBEFEHLE
/*************************************************************************
CODSPR: PROC (SPART,LTYP,WAHRH,OPERATOR,OPTYP,HILF1) GLOBAL
      DCL SPART FIXED;
      DCL LTYP FIXED;
      DCL WAHRH FIXED;
      DCL OPERATOR FIXED;
      DCL OPTYP FIXED;
      DCL HILF1(40) CHAR;
          DCL LNAME(8) CHAR;
      DCL LWERT(8) CHAR;
      DCL OPNAME(8) CHAR;
      DCL OPOFSET(8) CHAR;
      DCL SZAHL(8) CHAR;
      DCL HILFE(36) CHAR INITIAL(' ');
      DCL STRING(20) CHAR;
      DCL MARKE(10) CHAR;
      DCL NOP(6) CHAR(2) INITIAL ('  ','= ','<=',' =','  ','< ');
      DCL OP(6) CHAR(2) INITIAL ('= ','  ','  ','< ','<=',' =');
      DCL UNGLEICH CHAR;
      DCL UP(8) VAL CHAR IDENTICAL ('R','7',' ',' ',':','U','S',' ');
      DCL R6(4) VAL CHAR IDENTICAL ('R','6',':','=');
      DCL R7(4) VAL CHAR IDENTICAL ('R','7',':','=');
      DCL SP VAL CHAR(7) IDENTICAL(':SP  R7');
      DCL SCRATCH(80) CHAR INITIAL(' ');
      DCL IA(4) CHAR INITIAL('''','V','A','''');
      DCL MODUS FIXED;
      DCL I FIXED;
      DCL GROESSER CHAR;
      /* VORBESETZUNGEN */
      GROESSER=SYMBOL(14);
      UNGLEICH=SYMBOL(26);
      NOP(5)=GROESSER//' ';
      OP(3)= GROESSER//' ';
      NOP(4)=GROESSER//'=';
      OP(6) =GROESSER//'=';
      OP(2)=UNGLEICH//' ';
      NOP(1)=UNGLEICH//' ';
      MODUS=1;
      FOR I TO 8 REPEAT;
        LNAME(I)=HILF1(I);
        OPNAME(I)=HILF1(I+8);
        OPOFSET(I)=HILF1(I+16);
        SZAHL(I)=HILF1(I+24);
      END;
      FOR I TO 4 REPEAT;
        SCRATCH(I)=IA(I);
      END;
      /* BEDINGTER ODER UNBEDINGTER SPRUNG$ */
```

```
61          IF SPART LE 2 THEN
62            BEGIN;
63               /* BEDINGTER SPRUNG */
64               IF SPART EQ 2 THEN
65                 BEGIN;
66                    /* CODE ERZEUGEN FUER: INHALT DES OPERANDEN RECHTS */
67                    /* VOM VERGLEICHSOPERATOR DEM REGISTER R6 ZUWEISEN */
68                    IF OPERATOR GT 6 THEN
69                      BEGIN;
70                         OPTYP=11;
71                         OPNAME(1)='Ø';
72                      END;
73                    FI;
74                    FOR I TO 8 REPEAT;
75                      HILFE(I+2Ø)=OPNAME(I);
76                      HILFE(I+28)=OPOFSET(I);
77                    END;
78                    CALL CODOPE(HILFE,OPTYP);
79                    FOR I TO 2Ø REPEAT;
80                      STRING(I)=HILFE(I);
81                    END;
82                    FOR I FROM 16 TO 19 REPEAT;
83                      SCRATCH(I)=R6(I-15);
84                    END;
85                    FOR I FROM 24 TO 43 REPEAT;
86                      SCRATCH(I)=STRING(I-23);
87                    END;
88                    /* ERZEUGTEN CODE AUF PLATTE SCHREIBEN */
89                    CALL DISKAUS(MODUS,SCRATCH);
90                    FOR I FROM 5 TO 8Ø REPEAT;
91                      SCRATCH(I)=' ';
92                    END;
93                    /* JEWEILIGEN BEDINGUNGSCODE ERZEUGEN */
94                    SCRATCH(16)='G';
95                    SCRATCH(17)='Ø';
96                    IF WAHRH EQ 2 THEN
97                    BEGIN;
98                      IF OPERATOR GT 6 THEN OPERATOR=1;FI;
99                     SCRATCH(24)=1 CHAR NOP(OPERATOR);
00                      SCRATCH(25)=2 CHAR NOP(OPERATOR);
01                    END;
02                    FI;
03                    IF WAHRH EQ 1 THEN
04                      BEGIN;
05                        IF OPERATOR GT 6 THEN OPERATOR=1;FI;
06                         SCRATCH(24)=1 CHAR OP(OPERATOR);
07                         SCRATCH(25)=2 CHAR OP(OPERATOR);
08                      END;
09                    FI;
10                    FOR I TO 2 REPEAT;
11                      SCRATCH(I+25)=R6(I);
12                    END;
13               END;
14             FI;
15    /* CODE FUER BEDINGUNG (FALLS VORHANDEN) DURCH CODE FUER SPRUNGBEFEHL *.
16          /* ERGAENZEN UND DANN AUF PLATTE SCHREIBEN */
17          SCRATCH(3Ø)=':';
18          SCRATCH(31)='S';
19          SCRATCH(32)='P';
20          CALL CODLAB (MARKE,LTYP,LNAME);
```

```
121           FOR I TO 10 REPEAT;
122             SCRATCH(I+32)=MARKE(I);
123             END;
124                CALL DISKAUS(MODUS,SCRATCH);
125           END;
126           FI;
127           /* PROZEDURAUFRUF$ */
128           IF SPART EQ 3 THEN
129             /* JA */
130             BEGIN;
131                /* CODZEILEN FUER RETTEN DER RUECKSPRUNGADR. UND DEN SPRU
132                * ERZEUGEN UND AUF DIE PLATTE SCHREIBEN */
133                CALL CODLAB(MARKE,LTYP,LNAME);
134                FOR I TO 8 REPEAT;
135                  SCRATCH(I+15)=UP(I);
136                END;
137                FOR I TO 10 REPEAT;
138                  SCRATCH(I+25)=MARKE(I);
139                END;
140                CALL DISKAUS(MODUS,SCRATCH);
141             END;
142           FI;
143           /* PROZEDURRUECKSPRUNG$ */
144           IF SPART EQ 4 THEN
145             /* JA */
146             BEGIN;
147                /* CODEZEILEN FUER HOLEN DER RUECKSPRUNADR UND    */
148                /* RUECKSPRUNG ERZEUGEN UND AUF PLATTE SCHREIBEN */
149                FOR I TO 4 REPEAT;
150                  SCRATCH(I+15)=R7(I);
151                END;
152                SCRATCH(21)='(';
153                SCRATCH(22)='Q';
154                FOR I TO 8 REPEAT;
155                  SCRATCH(I+22)=LNAME(I);
156                END;
157                SCRATCH(32)=')';
158                CALL DISKAUS(MODUS,SCRATCH);
159                FOR I FROM 5 TO 80 REPEAT;
160                  SCRATCH(I)=' ';
161                END;
162                FOR I TO 7 REPEAT;
163                  SCRATCH(I+15)=I CHAR SP;
164                END;
165                CALL DISKAUS(MODUS,SCRATCH);
166             END;
167           FI;
168           RETURN;
169     END;/*ENDE VON CODSPR*/
170
171     /*************************************************************************
172     /*'CODLAB' ERZEUGT DEN CODE FUER DIE VERSCH. ARTEN DER
173     /* SPRUNGZIELANGABE
174     /*************************************************************************
175     CODLAB: PROC (MARKE,LTYP,LNAME) GLOBAL
176           DCL MARKE(10) CHAR;
177           DCL LTYP FIXED;
178           DCL LNAME(8) CHAR;
179           DCL I FIXED;
180           FOR I TO 8 REPEAT;
```

```
81            MARKE(I+1)=LNAME(I);
82          END;
83          MARKE(1)=' ';
84          MARKE(1Ø)=' ';
85          /* INDIEREKTE SPRUNGZIELANGABE,SPRUNGZIEL STEHT IM SPEICHER */
86          IF LTYP EQ 4 THEN
87            BEGIN;
88               MARKE(1)='(';
89               MARKE(1Ø)=')';
9Ø            END;
91          FI;
92          /* INIEREKTE SPRUNGZIELANGABE ,SPRUNGZIEL STEHT IM REGISTER */
93          IF LTYP EQ 3 THEN MARKE(2)='G';FI;
94          RETURN;
95    END;/*ENDE VON CODLAB*/
96    MODEND;
```

MODUL PC4

Dieser Modul enthält nur die Prozedur CODZUW zum Erzeugen des Assemblertextes für die Wertzuweisung.

CODZUW:

Leistung:

Diese Prozedur erzeugt den Assemblercode für die Wertzuweisung. Die Parameter TYP und HILF1 enthalten Informationen über den Operanden links vom Zuweisungszeichen. Auf der rechten Seite des Zuweisungszeichens steht immer das Register GØ. (siehe Beschreibung von CODARI) d.h. GØ ist immer die Datenquelle.

Parameter:

TYP: Integergröße
Dieser Parameter gibt den Typ des Operanden links vom Zuweisungszeichen an (siehe Kapitel 4.2.1.2).

HILF1: 40-elementiges Zeichenfeld
Dieses Feld ist eine Zusammenfassung von fünf 8-elementigen Zeichenfeldern

HILF1(1)-HILF1(8)
Diese Feldelemente enthalten den Namen des Operanden.

HILF1(9)-HILF1(16)
Diese Feldelemente enthalten einen eventuell vorhandenen Offset zu den Operanden.

Aufgerufene Prozeduren:	in Modul:
CODOPE	PCØ
DISKAUS	PCØ

Aufrufende Prozeduren:	aus Modul:
Task LAUF2	SCOD

```
1   MODULE PC4;
2   SYSTEM;
3         SDAU->DRUCKER;;
4   PROBLEM;
5   DCL DRUCKER VAL DEVICE GLOBAL;
6   /* SPEZIFIKATION DER IN DIESEM MODUL VERWENDETEN PROZEDUREN */
7   DCL DISKAUS ENTRY (FIXED,(80) CHAR) GLOBAL;
8   DCL CODOPE ENTRY ((36) CHAR,FIXED) GLOBAL;
9
10  /* 'CODZUW' ERZEUGT DEN ASS-310 TEXT FUER ZUWEISUNGEN                  *
11  /*******************************************************************/
12  /*******************************************************************/
13  CODZUW:PROC (TYP,HILF1) GLOBAL
14        DCL TYP FIXED;
15        DCL HILF1(40) CHAR;
16        DCL (ZIEL,SENKE)(8) CHAR INITIAL(' ');
17        DCL OFSET(8) CHAR;
18        DCL IA(4) VAL CHAR IDENTICAL('''','V','A','''');
19        DCL I FIXED;
20        DCL SCRATCH(80) CHAR INITIAL (' ');
21        DCL HILFE(36) CHAR;
22        DCL G0 VAL CHAR(4) IDENTICAL (':=G0');
23        DCL MODUS FIXED;
24        MODUS=1;
25        FOR I TO 40 REPEAT;
26          ZOPTI(I)=' ';
27        END;
28        FOR I TO 8 REPEAT;
29          SENKE(I)=HILF1(I);
30          OFSET(I)=HILF1(I+8);
31        END;
32        FOR I TO 4 REPEAT;
33          SCRATCH(I)=IA(I);
34        END;
35        FOR I TO 8 REPEAT;
36          HILFE(I+20)=SENKE(I);
37          HILFE(I+28)=OFSET(I);
38        END;
39        CALL CODOPERAND(HILFE,TYP);
40        FOR I FROM 9 TO 25 REPEAT;
41          SCRATCH(I)=HILFE(I-8);
42        END;
43        FOR I TO 4 REPEAT;
44          SCRATCH(I+25)=I CHAR G0;
45        END;
46        CALL DISKAUS(MODUS,SCRATCH);
47  RETURN;
48  END;/*ENDE VON CODZUW*/
49  MODEND;
```

MODUL PC5
=========

Dieser Modul enthält nur die Prozedur CODARI zum Generieren des Codes für einen arithmetischen Ausdruck.

Leistung:

Diese Prozedur erzeugt aus den erhaltenen Zwischeninformationen den entsprechenden Assemblertext. Das Ergebnis des arithmetischen Ausdrucks wird dem Register GØ zugewiesen (siehe Beschreibung von CODZUW und CODSPR). Der erzeugte Assemblertext wird auf die Datei CODE geschrieben.

Parameter:

CODNR: Integergröße
In CODNR steht die Kennziffer der Art des arithmetischen Ausdrucks (siehe Kapitel 4.2.1.2).

TYP1: Integergröße
Kennziffer des Operandentyps des ersten Operanden
TYP2: Integergröße
Kennziffer des Operandentyps des zweiten Operanden, falls ein solcher vorhanden ist.
OPZEICH: Integergröße
Kennziffer des Operators mit dem der 1. Operand mit dem zweiten verbunden ist. Gibt es keinen zweiten Operanden, ist dieser Parameter bedeutungslos.

HILF1: 40-elementiges Zeichenfeld.
HILF1 ist eine Zusammenfassung von fünf 8-elementigen Zeichenfeldern
HILF1(1)-HILF1(8)
Diese Feldelemente enthalten den Namen des ersten Operanden.

HILF1(9)-HILF1(16)
Diese Feldelemente enthalten einen eventuell vorhandenen Offset zum ersten Operanden.

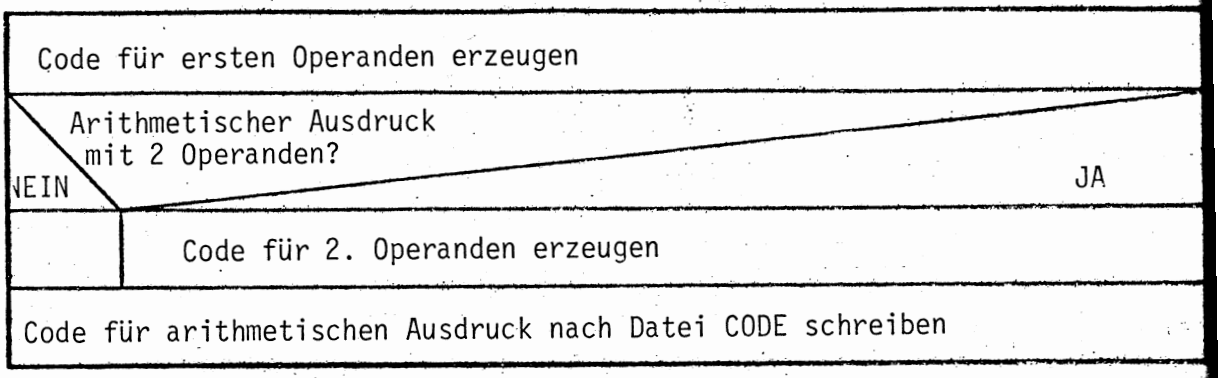
HILF1(17)-HILF1(24)
Diese Feldelemente enthalten den Namen eines eventuell vorhandenen zweiten Operanden.

HILF1(25)-HILF(32)
Diese Feldelemente enthalten den eventuell vorhandenen Offset für den zweiten Operanden.

Die Feldelemente HILF1(33)-HILF1(40) sind bedeutungslos.

Aufgerufene Prozeduren:	aus Modul:
CODOPE	PCØ
DISKAUS	PCØ

aufrufende Prozeduren:	aus Modul:
Task LAUF2	SCOD


Code für ersten Operanden erzeugen
Arithmetischer Ausdruck mit 2 Operanden?
NEIN
JA
Code für 2. Operanden erzeugen
Code für arithmetischen Ausdruck nach Datei CODE schreiben

```
MODULE PC5;
SYSTEM;
      SDAU-→DRUCKER;;
PROBLEM;
DCL DRUCKER VAL DEVICE GLOBAL;
/* SPEZIFIKATION DER IN DIESEM MODUL VERWENDETEN PROZEDUREN */
DCL DISKAUS ENTRY (FIXED,(8Ø) CHAR) GLOBAL;
DCL CODOPE ENTRY ((36) CHAR,FIXED) GLOBAL;

/*************************************************************************
/* 'CODARI' ERZEUGT DEN ASS-31Ø TEXT FUER ARITH. AUSDRUECKE
/*************************************************************************
CODARI:PROC (CODNR,TYP1,OPZEICH,TYP2,HILF1) GLOBAL
      DCL CODNR FIXED;
      DCL TYP1 FIXED;
      DCL (OPZEICH,TYP2) FIXED;
      DCL HILF1(4Ø) CHAR;
      DCL ZIEL(8) CHAR;
      DCL (NAME1,OFSET1)(8) CHAR;
      DCL (NAME2,OFSET2)(8) CHAR;
      DCL I FIXED;
      DCL SCRATCH(8Ø) CHAR INITIAL(' ');
      DCL GØ(4) VAL CHAR IDENTICAL ('G','Ø',':','=');
      DCL IA(4) VAL CHAR IDENTICAL('''','V','A','''');
      DCL OP(3) VAL CHAR IDENTICAL('+','-','*');
      DCL HILFE(36) CHAR;
      DCL STRING(2Ø) CHAR;
      DCL MODUS FIXED;
      MODUS=1;
      FOR I TO 8 REPEAT;
        NAME1(I)=HILF1(I);
        OFSET1(I)=HILF1(I+8);
        NAME2(I)=HILF1(I+16);
        OFSET2(I)=HILF1(I+24);
      END;
      FOR I TO 4 REPEAT;
        SCRATCH(I)=IA(I);
        SCRATCH(I+12)=GØ(I);
      END;
      FOR I TO 8 REPEAT;
        HILFE(I+2Ø)=NAME1(I);
        HILFE(I+28)=OFSET1(I);
      END;
      /* CODE FUER ERSTEN OPERANDEN ERZEUGEN */
      CALL CODOPE(HILFE,TYP1);
      FOR I TO 2Ø REPEAT;
        SCRATCH(I+16)=HILFE(I);
      END;
      /* ARITH. AUSDRUCK MIT 2 OPERANDEN$ */
      IF CODNR EQ 2 THEN
        BEGIN;
          FOR I TO 8 REPEAT;
            HILFE(I+2Ø)=NAME2(I);
            HILFE(I+28)=OFSET2(I);
          END;
          /* CODE FUER 2. OPERANDEN ERZEUGEN */
      /* CODE FUER ARITH. AUSDRUCK AUF DIE PLATTE SCHREIBEN¢ */
          CALL CODOPE(HILFE,TYP2);
      SCRATCH(37)=OP(OPZEICH);
      FOR I TO 2Ø REPEAT;
```

```
61            SCRATCH(I+37)=HILFE(I);
62          END;
63            END;
64          FI;
65          CALL DISKAUS(MODUS,SCRATCH);
66          RETURN;
67    END;/*ENDE VON CODARI*/
68
69    /*****************************************************************************
70    MODEND;
```

MODUL PC6:
==========

Dieser Modul enthält nur die Prozedur CODPRO für die Codeerzeugung des Prozeduranfangs.

CODPRO

Leistung:

Diese Prozedur erzeugt den Assemblercode für den Prozeduranfang.
Dies bedeutet, daß zunächst eine Speicherzelle reserviert wird, wo die Rücksprungaddresse aufgehoben wird. Der Name dieser Zelle wird dadurch erzeugt, daß an den Prozedurnamen vorne noch der Buchstabe Q angefügt wird.
Dann wird der Befehl erzeugt, der die Rücksprungaddresse (Inhalt von R7) in diese Zelle speichert.
Beim Ende einer Prozedur wird diese Codeprozedur ebenfalls aufgerufen (CODNR=2), erzeugt aber keinen Assemblertext.

Parameter:

CODNR: Integergröße
Dieser Parameter enthält eine Kennziffer, ob Code für den Prozeduranfang oder das Prozedurende erzeugt werden soll (siehe Kapitel 4.2.1.2).

HILF1: 40-elementiges Zeichenfeld
Von Bedeutung sind in diesem Zeichenfeld nur die ersten acht Feldelemente. HILF1(1) bis HILF1(8) enthält den Namen der Prozedur, für die der Anfangscode erzeugt werden soll.

Aufrufende Prozeduren:	aus Modul:
Task LAUF2	SCOD
Aufgerufene Prozeduren:	**in Modul:**
DISKAUS	PCØ

```
   1    MODULE PC6;
   2    SYSTEM;
   3          SDAU-→DRUCKER:;
   4    PROBLEM;
   5    DCL DRUCKER VAL DEVICE GLOBAL;
   6    /* SPEZIFIKATION DER IN DIESEM MODUL VERWENDETEN PROZEDUREN */
   7    DCL DISKAUS ENTRY (FIXED,(80) CHAR) GLOBAL;
   8    /*****************************************************************
   9    /* 'CODPRO' ERZEUGT DEN CODE FUER DEN ANFANG UND FUER DAS ENDE
  10    /* EINER PROZEDUR
  11    /*****************************************************************
  12    CODPRO:PROC (CODNR,HILF1) GLOBAL
  13          DCL CODNR FIXED;
  14          DCL HILF1(40) CHAR;
  15          DCL PNAME(8) CHAR;
  16          DCL IA VAL CHAR(4) IDENTICAL ('''VA''');
  17          DCL WTL VAL CHAR(10) IDENTICAL('-1WTL):=R7');
  18          DCL SCRATCH(80) CHAR INITIAL(' ');
  19          DCL I FIXED;
  20          DCL R7 VAL CHAR(6) IDENTICAL('R7:= (');
  21          DCL WTL1 VAL CHAR(6) IDENTICAL('-1WTL)');
  22          DCL MODUS FIXED;
  23          DCL GROESSER CHAR;
  24          /* VORBESETZUNGEN */
  25          GROESSER=SYMBOL(14);
  26          MODUS=1;
  27          FOR I TO 8 REPEAT;
  28            PNAME(I)=HILF1(I);
  29          END;
  30          FOR I TO 4 REPEAT;
  31            SCRATCH(I)=I CHAR IA;
  32          END;
  33               /* CODE FUER DAS RESERVIEREN DER RETTZELLE ERZEUGEN */
  34               SCRATCH(2)='V';
  35               SCRATCH(6)='Q';
  36               FOR I FROM 7 TO 13 REPEAT;
  37                 SCRATCH(I)=PNAME(I-6);
  38               END;
  39               SCRATCH(14)='/';
  40               SCRATCH(15)='=';
  41               SCRATCH(16)='1';
  42               SCRATCH(17)='+';
  43               SCRATCH(18)=GROESSER;
  44               /* CODE ZUM SPEICHERN DER RUECKSPRUNGADRESSE ERZEUGEN */
  45               CALL DISKAUS(MODUS,SCRATCH);
  46               FOR I FROM 5 TO 80 REPEAT;
  47                 SCRATCH(I)=' ';
  48               END;
  49               SCRATCH(16)='Q';
  50               SCRATCH(15)='(';
  51               FOR I TO 7 REPEAT;
  52                 SCRATCH(I+16)=PNAME(I);
  53                 SCRATCH(I+6)=PNAME(I);
  54               END;
  55               SCRATCH(14)='/';
  56               SCRATCH(24)=')';
  57               SCRATCH(25)=':';
  58               SCRATCH(26)='=';
  59               SCRATCH(27)='R';
  60               SCRATCH(28)='7';
```

```
61                CALL DISKAUS(MODUS,SCRATCH);
62                GOTO ENDE;
63           ENDE:;
64           RETURN;
65      END;/*ENDE VON CODPROC*/
66      MODEND;
```

MODUL PC7
=========

Dieser Modul enthält nur die Prozedur CODORG, die je nachdem welcher organisatorische Befehl vorliegt, von entsprechenden Hilfsdateien den Assemblercode für den organisatorischen Befehl in die Codedatei hineinkopiert.

CODORG:

Leistung:

Diese Prozedur kopiert von der entsprechenden Hilfsdatei den Code für den organisatorischen Befehl in die Codedatei hinein.
Als Hilfsdateien benötigt CODORG:

- SAVE: enthält den Assemblercode für SAVE(ALL) REGISTERS
- RESU: enthält den Assemblercode für RESUME PROCESS
- HALT: enthält den Assemblercode für HALT
- DOIO: enthält den Assemblercode für DOIO

Diese Dateien sind im Lochkartenformat angelegt.
Im letzten Kartenäquivalent steht an erster Stelle als Endezeichen für eine Datei ein '/'.
Für die organisatorischen Befehle ENABLE, DISABLE und ERROR wird kein Assemblercode erzeugt (siehe Kapitel 3.4).

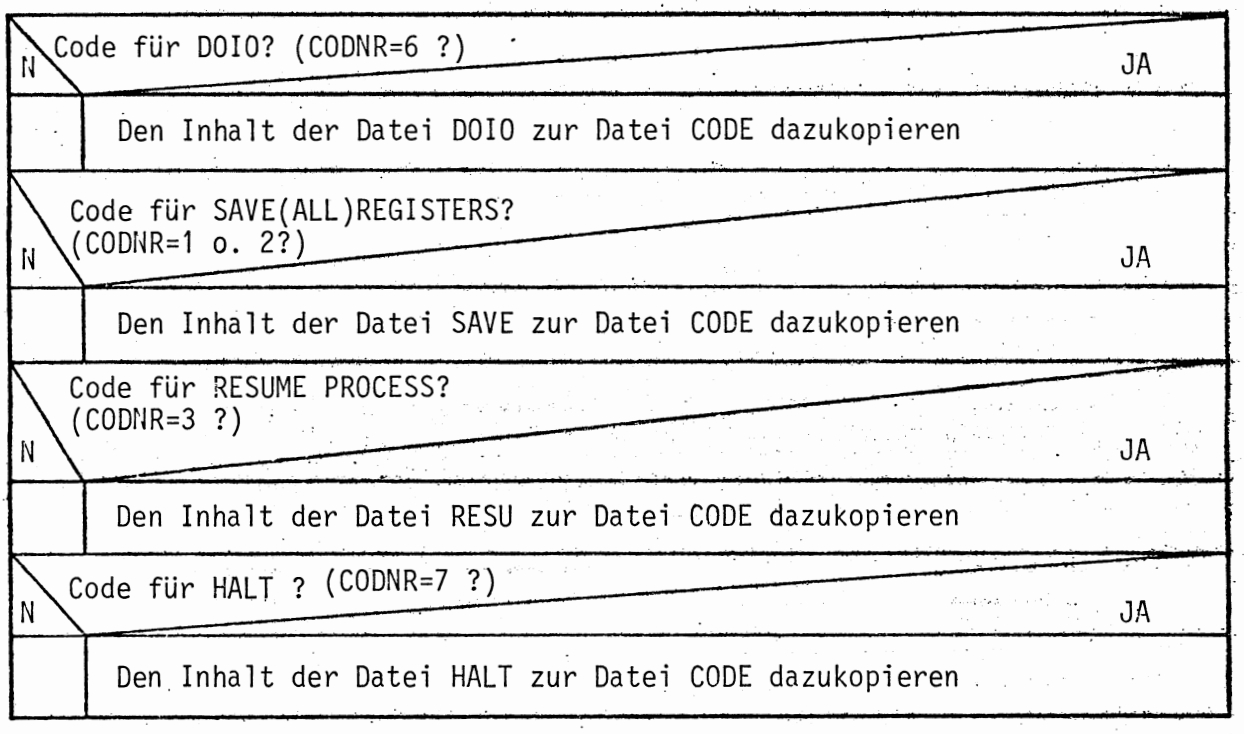
Parameter:

CODNR: Integergröße
Dieser Parameter enthält die Kennziffer des organisatorischen Befehls, dessen Assembleräquivalent zur Codedatei dazukopiert werden soll.
(Kennziffern siehe Kapitel 4.2.1.2).

HILF1: 40-elementiges Zeichenfeld
keine Bedeutung

Aufrufende Prozeduren:	aus Modul:
Task LAUF2	SCOD

Aufgerufene Prozeduren:	in Modul:
keine	


Code für DOIO? (CODNR=6 ?)
N
JA
Den Inhalt der Datei DOIO zur Datei CODE dazukopieren
Code für SAVE(ALL)REGISTERS?
(CODNR=1 o. 2?)
N
JA
Den Inhalt der Datei SAVE zur Datei CODE dazukopieren
Code für RESUME PROCESS?
(CODNR=3 ?)
N
JA
Den Inhalt der Datei RESU zur Datei CODE dazukopieren
Code für HALT ? (CODNR=7 ?)
N
JA
Den Inhalt der Datei HALT zur Datei CODE dazukopieren

```
   1   MODULE PC7;
   2   SYSTEM;
   3         SDAU-→DRUCKER:;
   4         LKEI←-LESER:;
   5         PSEA←-→DISK:;
   6   PROBLEM;
   7         DCL (LESER,DRUCKER,DISK) VAL DEVICE GLOBAL;
   8   /* SPEZIFIKATION DER IN DIESEM MODUL VERWENDETEN PROZEDUREN */
   9         DCL DISKAUS ENTRY(FIXED,(8Ø)CHAR) GLOBAL;
  10
  11   /*************************************************************************
  12   /* 'CODORG' ERZEUGT DEN CODE FUER DIE ORG. BEFEHLE
  13   /*************************************************************************
  14   CODORG: PROC (CODNR,HILF1) GLOBAL
  15         DCL HILF1(4Ø) CHAR;
  16         DCL CODNR FIXED;
  17         DCL (NAME,ZAHL)(8) CHAR;
  18         DCL I FIXED;
  19         DCL (FDOI,FRES,FHAL,FSAV) VAL FILE;
  20         DCL SCRATCH(8Ø) CHAR INITIAL(' ');
  21         DCL MODUS FIXED INITIAL(1);
  22         I=1;
  23         /* CODE FUER 'DOIO' */
  24         IF CODNR EQ 6 THEN
  25           /* JA */
  26           BEGIN;
  27              OPEN FDOI TITLE 'DOIO' UPON DISK INPUT DIR (8Ø) ALPHA;
  28         NEW1:GET FDOI POS(I) EDIT (SCRATCH) ((8Ø)A(1));
  29              IF SCRATCH(1) NE '/' THEN
  30                BEGIN;
  31              /* DEN INHALT DER DATEI 'DCIO' ZUR DATEI 'CODE'
  32              /* HINZUKOPIEREN
  33                   CALL DISKAUS(MODUS,SCRATCH);
  34                   I=I+1;
  35                   GOTO NEW1;
  36                END;
  37              FI;
  38              CLOSE FDOI;
  39           END;
  40         FI;
  41         I=1;
  42         /* CODE FUER 'SAVE'$ */
  43         IF (CODNR EQ 2) OR (CODNR EQ 1) THEN
  44           /* JA */
  45           BEGIN;
  46              /* DEN INHALT DER DATEI 'SAVE' ZUR DATEI 'CODE'
  47              /* HINZUKOPIEREN
  48              OPEN FSAV TITLE 'SAVE' UPON DISK INPUT DIR (8Ø) ALPHA;
  49           NEW2:GET FSAV POS(I) EDIT (SCRATCH) ((8Ø)A(1));
  50              IF SCRATCH(1) NE '/' THEN
  51                BEGIN;
  52                   CALL DISKAUS (MODUS,SCRATCH);
  53                   I=I+1;
  54                   GOTO NEW2;
  55                END;
  56              FI;
  57              CLOSE FSAV;
  58           END;
  59         FI;
  60         /* CODE FUER 'RESUME PROCESS'$ */
```

```
          I=1;
          IF CODNR EQ 3 THEN
            /* JA */
            BEGIN;
               /* DEN INHALT DER DATEI 'RESU' ZUR DATEI 'CODE'          *
               /* HINZUKOPIEREN                                         *
               OPEN FRES TITLE 'RESU' UPON DISK INPUT DIR (80) ALPHA;
               NEW3:;
               GET FRES POS(I) EDIT (SCRATCH) ((80)A(1));
               IF SCRATCH(1) NE '/' THEN
                 BEGIN;
                    CALL DISKAUS (MODUS,SCRATCH);
                    I=I+1;
                    GOTO NEW3;
                 END;
               FI;
               CLOSE FRES;
            END;
          FI;
          I=1;
          /* CODE FUER 'HALT'S */
          IF CODNR EQ 7 THEN
            /* JA */
            BEGIN;
               /* DEN INHALT DER DATEI 'HALT' ZUR DATEI 'CODE'          *
               /* HINZUKOPIEREN                                         *
               OPEN FHAL TITLE 'HALT' UPON DISK INPUT DIR (80) ALPHA;
               NEW4:;
               GET FHAL POS(I) EDIT (SCRATCH) ((80)A(1));
               IF SCRATCH(1) NE '/' THEN
                 BEGIN;
                    CALL DISKAUS (MODUS,SCRATCH);
                    I=I+1;
                    GOTO NEW4;
                 END;
               FI;
               CLOSE FHAL;
            END;
          FI;
          RETURN;
    END;/*ENDE VON CODORG*/
    MODEND;
```

MODUL PC8
==========

Dieser Modul enthält nur die Prozedur CODLOC zur Assemblertexterzeugung für eine Markenvereinbarung.

CODLOC:

Leistung:
Die Prozedur CODLOC erzeugt den Assemblertext für eine Markenvereinbarung. (siehe ASS300)

Parameter:
HILF1: 40-elementiges Zeichenfeld
In den Feldelementen HILF1(1) bis HILF1(8) steht der Name der Marke. Die anderen Feldelemente sind bedeutungslos.

Aufrufende Prozeduren:	aus Modul:
Task LAUF2	SCOD

Angerufene Prozeduren:	in Modul:
DISKAUS	PCO

```
MODULE PC8;
SYSTEM;
      SDAU->DRUCKER:;
PROBLEM;
DCL DRUCKER VAL DEVICE GLOBAL;
/* SPEZIFIKATION DER IN DIESEM MODUL VERWENDETEN PROZEDUREN */
DCL DISKAUS ENTRY (FIXED,(8Ø) CHAR) GLOBAL;

/*************************************************************************
/* 'CODLOC' ERZEUGT DEN CODE FUER DIE MARKENVEREINBARUNGEN               *
/*************************************************************************/
CODLOC : PROC (HILF1) GLOBAL
      DCL HILF1(4Ø) CHAR;
      DCL NAME(8) CHAR;
      DCL SCRATCH(8Ø) CHAR INITIAL(' ');
      DCL I FIXED;
      DCL IV VAL CHAR(4) IDENTICAL('''VA''');
      DCL MODUS FIXED INITIAL(1);
      FOR I TO 8 REPEAT;
        NAME(I)=HILF1(I);
      END;
      FOR I TO 4 REPEAT;
        SCRATCH(I)=I CHAR IV;
      END;
      FOR I TO 8 REPEAT;
        SCRATCH(I+5)=NAME(I);
      END;
      SCRATCH(14)='/';
      CALL DISKAUS(MODUS,SCRATCH);
      RETURN;
END;/*ENDE VON CODLOC*/

/*************************************************************************

CODCOM: PROC (SATZ) GLOBAL
      DCL SATZ FIXED;
      RETURN;
END;/*END OF CODCOM*/
MODEND;
```

Anhang IV Listing des rechnerunabhängigen PEARL-Betriebssystems

```
1       START;
2     /                                 SPASS-PEARL-BETRIEBSSYSTEM;
3     /                                 ==========================;
4     /;
5     /                                 SIEMENS-31?-VERSION;
6     /                                 ROLAND ROESSLER;
7     /;
8     /;
9         LENGTH: POINTER=1;                        TYPLAENGENVEREINBARUNGEN
10        LENGTH: INT=1;
11    /;
12        STRUCT:CHAIN;                             WARTESCHLANGENELEMENT
13              ELEM:SUCC=POINTER;                  ZEIGER AUF NACHFOLGER
14              ELEM:PRED=POINTER;                  ZEIGER AUF VORGAENGER
15        STREND:CHAIN;
16    /;
17        STRUCT:DRIVER;                            GERAET
18              ELEM:QUEUE=CHAIN;                   WARTESCHLANGE
19              ELEM:SDP=POINTER;                   ZEIGER AUF BELEGENDEN PROZES
20              ELEM:IO=POINTER;                    ZEIGER AUF ANL.ABH. ROUTINE
21              ELEM:STW=POINTER;                   ADRESSE DES STATUSWORTES
22        STREND:DRIVER;
23    /;
24        STRUCT:PARBLOC;                           AUFRUF-PARAMETERBLOCK
25              ELEM:SKETT=POINTER;                 SCHEDULE-KETTE
26              ELEM:SPCB=POINTER;                  ZEIGER AUF TCB,SEMA,GERAET
27              ELEM:PSVC=POINTER;                  ZEIGER AUF ROUTINE
28              ELEM:STYP=INT;                      SCHEDULE-TYP
29        STREND:PARBLOC;
30    /;
31        STRUCT:SCHED;                             SCHEDULE
32              ELEM:SCPAR=PARBLOC;                 PARAMETERBLOCK
33              ELEM:SCPRIO=INT;                    'SCHEDULE'-PRIORITAET
34              ELEM:TYP=INT;                       AKTUELLER TYP
35              ELEM:SCHART=POINTER;                ZEIGER AUF ZEIGER AUF SCHEDU
36              ELEM:REST=INT;                      RESTZEIT
37              ELEM:SBEG=INT;                      INTERRUPT-NUMMER
38              ELEM:SAFT=INT;                      'AFTER'-INTERVALL
39              ELEM:ZYKL=INT;                      'ALL'-INTERVALL
40        STREND:SCHED;
41    /;
42        STRUCT:IOBLOC;                            I/O-PARAMETERBLOCK
43              ELEM:IOPAR=PARBLOC;                 STANDARD-PARAMETERBLOCK
44              ELEM:IOBUF=POINTER;                 ZEIGER AUF I/O-PUFFER
45        STREND:IOBLOC;
46    /;
47        STRUCT:SEMA;                              SEMAPHORE
48              ELEM:SWS=CHAIN;                     SWS
49              ELEM:PRIO=INT;                      WERT
50              ELEM:STATUS=INT;                    ZUSTANDSKENNUNG(PCB) BZW.ANI
51        STREND:SEMA;
52    /;
53        STRUCT:PCB;                               TASKKONTROLLBLOCK
54              ELEM:KOPF=SEMA;                     KOPF
55              ELEM:SCHPNT=POINTER;                ZEIGER AUF ACTIVATE-SCHEDULE
56              ELEM:SCHRES=POINTER;                ZEIGER AUF CONTINUE-SCHEDULE
57              ELEM:IOB=POINTER;                   ZEIGER AUF E/A-PUFFER
58              ELEM:ERROR=INT;                     FEHLERKENNUNG NACH E/A-AUFRU
59              ELEM:PC=POINTER;                    AKT. BEFEHLSZAEHLER
```

```
60              ELEM;STPC=POINTER;                  STARTADRESSE
61              ELEM;REG1=INT;                      REGISTERRETTBEREICH
62              ELEM;REG2=INT;                      REGISTERRETTBEREICH
63              ELEM;REG3=INT;                      REGISTERRETTBEREICH
64              ELEM;REG4=INT;                      REGISTERRETTBEREICH
65              ELEM;REG5=INT;                      REGISTERRETTBEREICH
66              ELEM;REG6=INT;                      REGISTERRETTBEREICH
67              ELEM;REG7=INT;                      REGISTERRETTBEREICH
68              ELEM;REG8=INT;                      REGISTERRETTBEREICH
69              ELEM;REG9=INT;                      REGISTERRETTBEREICH
70              ELEM;REG10=INT;                     REGISTERRETTBEREICH
71              ELEM;REG11=INT;                     REGISTERRETTBEREICH
72              ELEM;REG12=INT;                     REGISTERRETTBEREICH
73              ELEM;REG13=INT;                     REGISTERRETTBEREICH
74              ELEM;REG14=INT;                     REGISTERRETTBEREICH
75              ELEM;REG15=INT;                     REGISTERRETTBEREICH
76              ELEM;REG16=INT;                     REGISTERRETTBEREICH
77          STREND;PCB;
78      /;
79      /                               DATEN;
80      /                               =====;
81      /;
82          SPACE;ORGSCH=SCHED+;                    INITIALSCHEDULE
83                SPEL=ANIL+;
84                SPEL=AFIRSTCB+;
85                SPEL='K1'+;
86                SPEL='K0'+;
87                SPEL='K0';
88      /;
89          SPACE;ILANG=INT+;                       LAENGE DER ITR-LISTE
90                SPEL='K20';
91          SPACE;TLANG=INT+;                       LAENGE DER ZEITLISTE
92                SPEL='K100';
93      /                               RAM-DATEN;
94      /                               =========;
95      /;
96          SPACE;ITRLIST=20*POINTER;               INTERRUPTLISTE
97      /;
98          SPACE;TIMLST=100*POINTER;               ZEITLISTE
99          SPACE;TIMEND=POINTER;
100         SPACE;TPOINT=POINTER;
101     /;
102         SPACE;RUNNING=PCB;                      PWS-KOPF
103     /;
104         SPACE;TLANGH=INT;
105         SPACE;THZ=POINTER;
106         SPACE;THZ1=POINTER;
107     /;
108     /;
109     /;
110     /                               INITIALISIERUNGSROUTINE;
111     /                               ======================;
112     /;
113     /                                 DER CODEGENERATOR MUSS ABLEGEN:;
114     /                                 ADRESSDEFINITION 'FIRSTCB';
115     /                                 TASKKONTROLLBLOECKE;
116     /                                 ADRESSDEFINITION 'FIRSTSM';
117     /                                 SEMAS;
118     /                                 TREIBER-BLOECKE;
119     /                                 ADRESSDEFINITION 'FIRSTIO';
120     /                                 ADRESSDEFINITION 'LASTIO'
```

```
121         LOC: INIT;
122              R1:= ADITRLIST;
123              TLANGH:=TLANG-'K1';
124              R2:= ILANG;
125              .CALL. INI1;                        INITIALISIEREN DER ITR-LIST
126              R1:= ATIMLST;
127              R2:= TLANG;
128              .CALL. INI1;                        INITIALISIEREN DER ZEITLIST
129              R1:= RUNNING:= ARUNNING;
130              PRED,R1:=ARUNNING;
131              R1:= FIRSTCB;
132         LOC: INIT1;
133              .TO. SEMAS .IF. R1 .EQ.  FIRSTSM;
134              SUCC,R1:=PRED,R1:=SCHPNT,R1:=SCHRES,R1:=ANIL;
135              STATUS,R1:= 'B0000';
136              R1:=R1+APCB;
137              .TO. INIT1;
138     /;
139         LOC: SEMAS;
140         LOC: INIT3;
141              .TO. DRIVS .IF. R1 .EQ.  FIRSTIO;
142              SUCC,R1:=PRED,R1:=R1;
143              PRIO,R1:=STATUS,R1;                 ANFANGSWERT EINSETZEN
144              R1:=R1+ASEMA;                       NAECHSTE SEMA
145              .TO. INIT3;
146     /;
147         LOC: DRIVS;
148         LOC: INIT4;
149              .TO. ACTV  .IF. R1 .EQ.  LASTIO;
150              SUCC,R1:=PRED,R1:=R1;
151              SDP,R1:=ANIL;
152              R1:=R1+ADRIVER;                     NAECHSTER TREIBER
153              .TO. INIT4;
154     /;
155         LOC: ACTV;
156              R1:= AORGSCH;
157              .TO. SVC;
158     /;
159         PROC: INI1;                              INITIALISIEREN VON SCHEDULE
160         LOC: INI2;
161              A0,R1:= R1;                         ZEIGER ZEIGT AUF SICH SELBS
162              R1:= R1+APOINTER;
163              .TO. INI2 .IFNOT. R2:= R2-'K1' .NULL.;
164              .RETURN. INI1;
165         END: INI1;
166     /;
167     /;
168     /;
169     /         SUPERVISOR-CALL;
170     /         ===============;
171     /;
172     /                                           R1: ZEIGER AUF SCHEDULE BZW
173     /                                           PARAMETERBLOCK;
174     /                                           ERGEBNIS:;
175     /                                           R2: ZEIGER AUF PCB, SEMA, G
176     /                                           R4: NULL, WENN KEIN SCH.;
177     /;
178         LOC: SVC;
179              DISABLE;
180              SAVE REGISTERS;
```

```
181             R2 := SPCB,R1;                      ZEIGER AUF PCB ODER SEMA ODE
182             .TO. SVC1 .IF. R4:= STYP,R1 .NULL.;     WENN LEER=SCHEDULE
183             TYP,R1 := R4;                       SCHEDULE-TYP INITIALISIEREN
184             REST,R1:='KØ';
185         LOC: SVC1;
186             R3 := PSVC,R1;                      ZEIGER AUF SVC-ROUTINE
187             .CALL. EXOPE;
188     /;
189         LOC: ASSIGN;                            PROZESSOR ZUTEILEN
190             .TO. NOTASK .IF. ARUNNING .EQ. RUNNING;
191             RESUME PROCESS;                       TASK AUFNEHMEN
192     /;
193         LOC: NOTASK;                            KEINE TASK LAUFEND
194             ENABLE;
195             HALT;                                 WARTEN AUF INTERRUPT
196     /;
197     /;
198     /;
199     /;
200     /;
201     /;
202     /;
203     /           STANDARD=TREIBER;
204     /           ================;
205     /;
206      PROC: DRIV;
207             .TO. DFREE .IFNOT. R4 .NIL.;    WENN VON INTERRUPT
208             R7:= R2;                        E/A-STRUKTUR
209             R2:= RUNNING;
210             IOB,R2:= IOBUF,R1;              PUFFERADRESSE RETTEN
211             .CALL. UNCH;                    AUS PROZESSORSCHLANGE
212             R1:= R7;
213             .TO. DFREE .IF. SDP,R1 .NIL.; GERAET FREI
214             .CALL. CHAIN;                   IN GWS
215             .RETURN. DRIV;
216     /;
217         LOC: DFREE;
218             SDP,R1:= R2;                    AUSFUEHRENDE TASK
219             DOIO;                             GERAETEABHAENGIG
220             .RETURN. DRIV;
221      END: DRIV;
222     /;
223     /;
224     /           STANDARD=ITR=BEARBEITUNG;
225     /           ========================;
226     /           (FUER GERAETE);
227     /;
228     /           VERSORGUNG: R3..ADRESSE DES GERAETES;
229     /;
230         LOC: DRITR;
231             SAVE ALL REGISTERS;
232             R1:= ARUNNING;
233             .TO. ASSIGN .IF. R2:=SDP,R3 .NIL.;      UNERWARTETER INTERR
234             R7:=STW,R3;                       ADRESSE DES STATUSWORTES
235             ERROR,R2:=ØØ,R7;                  STATUSWORT IN PCB
236     /;
237         LOC: DRI1;
238     /;
239             R7:= R3;
240             SDP,R3:= ANIL;
```

```
241             .CALL. CHAIN;
242             R1:= R7;
243             .TO. ASSIGN .IF. R2:=SUCC,R1 .EQ. R1;      KEINE TASK WAR
244             .CALL. UNCH;
245             .CALL. DRIV;
246             .TO. ASSIGN;
247     /;
248     /                                                  BEFEHLSAUSFUEHRUNG WIEDERHO
249         LOC: REPT;
250     /;
251     /       KANN VON GERAETEABHAENGIGEM TEIL ANGESPRUNGEN WERDEN, WEN
252     /       ANSTOSS ZU WIEDERHOLEN IST (Z.B. WENN GERAET UNKLAR WAR)
253     /;
254             .TO. ASSIGN .IF. R2:=SDP,R3 .NIL.;      UNERWARTETER INTE
255             R4:=R1:=R3;                            GERAET
256             .CALL. DRIV;
257             .TO. ASSIGN;
258     /;
259     /;
260     /;
261     /       'PROZESS'-INTERRUPT;
262     /       ====================;
263     /                                              R1: INTERRUPTNUMMER;
264     /;
265         LOC: ITR;
266             SAVE ALL REGISTERS;
267             R1:= R1*$POINTER;
268             THZ:= $ITRLIST + R1;                   ZEIGER AUF ITR-ZELLE
269             .TO. TIM1;
270     /;
271     /;
272     /       TIMER;
273     /       =====;
274     /;
275         LOC: TIMER;
276             SAVE ALL REGISTERS;
277             TPOINT:= $POINTER + TPOINT;    SCHEDULE-ZEIGER ERHOEHEN
278             .TO. TIM2 .IF. $TIMEND .NE. TPOINT;
279             TPOINT:= $TIMLST;                      WENN LISTENUEBERLAUF
280         LOC: TIM2;
281             THZ;= TPOINT;
282     /;
283         LOC: TIM1;
284             THZ1:= @0,THZ;
285         LOC: TIM3;
286             .TO. ASSIGN .IF. R1:=THZ1 .EQ. THZ;
287             THZ1:= SKETT,R1;
288             R2:= SPCB,R1;                          ZEIGER AUF PCB ODER SEMA
289             .CALL. SCH;                            SCHEDULE-VERARBEITUNG
290             .TO. TIM3 .IF. R3 .NIL.;               KEIN STATEMENT AUSZUFUEHREN
291             .CALL. EXOPE;
292             .TO. TIM3;
293     /;
294     /;
295     /;
296     /       SCHEDULE-VERARBEITUNG;
297     /       =====================;
298     /;
299     /       EINGABE;
300     /       R1: ZEIGER AUF SCHEDULE;
```

```
/          R2: ZEIGER AUF PCB BZW. SEMA;
/;
/          VERAENDERT;
/          R2: HILFSGROESSE, WENN KEINE OPERATION ANZUSTOSSEN;
/          R3: ADRESSE DER OPERATION, WENN ANZUSTOSSEN, SONST NIL;
/          R4: HILFSREGISTER;
/          R5: HILFSREGISTER;
/          R6: SCHEDULE-PRIORITAET;
/;
 PROC: SCH;
          .TO. SCHNF .IFNOT. REST,R1 .NULL.;      WENN NOCH RESTZEIT
     LOC: SCHØ;
          .TO. SCHOP .IF. TYP,R1 .EQ. 'K4';       SCHEDULE ABGELAUFEN
          R5:= R1;                                REGISTER 1 RETTEN
          R6:= SCHART,R1;
          R1:= Ø,R6;                              ZEIGER AUF ALTEN SCHEDULE
          .CALL. DEQU;                            ALTEN SCHEDULE AUSKETTEN
          Ø,R6:= R1:= R5;                         ZEIGER AUF NEUEN SCHEDULE
          .TO. SCHWH .IF. TYP,R1 .GE. 'K12';      WHEN-SCHEDULE
          .TO. SCHAF .IF. TYP,R1 .GE. 'K8';       AFTER-SCHEDULE
/;
          R3:= ZYKL,R1;                           ALL-SCHEDULE
          .CALL. ENQ;                             NEU EINKETTEN
          .TO. SCHO1;
/;
     LOC: SCHOP;                                  OPERATION ANSTOSSEN
          .TO. SCHO2 .IF. STYP,R1 .LT. 'K12';     KEIN 'WHEN'-SCHEDULE
          .TO. SCHO1 .IF. PSVC,R1 .NE. ARESUME;   KEIN RESUME
     LOC: SCHO2;
          .CALL. DEQU;
          R3:= SCHART,R1;                         ZEIGER AUF SCHEDULE
          Ø,R3:= NIL;
     LOC: SCHO1;
          R6:= SCPRIO,R1;                         PRIORITAET
          R3:= PSVC,R1;                           ADRESSE DER ROUTINE
          R4:= 'KØ';                              KENNUNG: OHNE SCHEDULE
          .RETURN. SCH;
/;
     LOC: SCHWH;                                  WHEN-SCHEDULE
          TYP,R1:= TYP,R1 - 'K8';                 TYP WEITER SCHALTEN
          R2:= SBEG,R1*APOINTER;                  INTERRUPTNUMMER IN SBEG,R1
          R4:=AITRLIST*R1;
          SKETT,P1:=R4;
          Ø,R4:=R1;
          .TO. SCH3;
/;
     LOC: SCHAF;                                  AFTER-SCHEDULE
          TYP,R1:= TYP,R1 - 'K4';                 TYP WEITERSCHALTEN
          .TO. SCHØ .IF. R3:=SAFT,R1 .NULL.;      WENN AFTER Ø SEC
          .CALL. ENQ;                             SCHEDULE EINKETTEN
/;
     LOC: SCH3;
          R3:= NIL;                               KEINE OPERATION ANZUSTOSSEN
          .RETURN. SCH;
/;
     LOC:SCHNF;                                   ZEIT WAR GROESSER ALS LISTENLA
          .CALL. DEQU;                            SCHEDULE AUSKETTEN
          R3:=REST,R1;                            VERBLEIBENDE RESTZEIT
          .CALL. ENQ;                             SCHEDULE WIEDER EINKETTEN
          .TO. SCH3;
```

```
361       END: SCH;
362      /;
363      /;
364      /          EINKETTEN EINES SCHEDULES IN ZEITLISTE;
365      /          ======================================;
366      /;
367      /          EINGABE;
368      /          R3 RELATIVE ZEIT;
369      /;
370      /          VERAENDERT;
371      /          R3: HILFSREGISTER;
372      /          R4: HILFSREGISTER;
373      /;
374       PROC: ENQ;
375                 R4:=TLANGH;                          LAENGE DER ZEITLISTE - 1
376                 .TO. ENQ2 .IF. R3 .GT. R4;           ZEITANGABE ZU GROSS FUER LI
377                 REST,R1:='KØ';                       KEINE RESTZEIT (MEHR)
378            LOC:ENQ3;
379                 R3:=R3*ÅPOINTER;
380                 R3:=R3+TPOINT;                       AKT. ZEIT + ZEITDIFFERENZ
381                 R4:=TLANG*ÅPOINTER;                  LISTENLAENGE IN SPEICHEREINH
382            LOC: ENQØ;
383                 .TO. ENQ1 .IF. ÅTIMEND .GT. R3; KEINUEBERLAUF
384                 R3:= R3 - R4;
385                 .TO. ENQØ;
386            LOC: ENQ1;
387                 SKETT,R1:= ÅØ,R3;                    EINKETTEN
388                 .RETURN. ENQ;
389            LOC:ENQ2;
390                 REST,R1:=R3-R4;                      RESTZEIT:=ZEIT-(LISTENLAENGE
391                 R3:=R4;                              LISTENLAENGE-1
392                 .TO. ENQ3;
393       END: ENQ;
394      /;
395      /;
396      /;
397      /                              AUSKETTEN AUS SCHEDULE-KETTE;
398      /                              =============================;
399      /;
400      /                              VERSORGUNG: R1...ZEIGER AUF SCHED. ODE
401      /;
402           PROC:DEQU;
403                 .TO. DEQ1 .IFNOT. R1 .NIL.;
404                 .RETURN. DEQU;                       KEIN ELEMENT AUSZUKETTEN
405            LOC:DEQ1;
406                 R3:=R1;
407                 .TO. DEQ2 .IFNOT. R4:=SKETT,R1 .NIL.;
408                 .RETURN. DEQU;                       ELEMENT IST IN KEINER KETTE
409            LOC:DEQ2;
410                 R3:=SKETT,R3;
411                 .TO. DEQ2 .IF. SKETT,R3 .NE. R1;
412                 SKETT,R3:=R4;
413                 SKETT,R1:=ÅNIL;
414                 .RETURN. DEQU;
415            END:DEQU;
416      /;
417      /;
418      /                              ACTIVATE;
419      /                              =========;
420      /;
```

```
421     /                                VERSORGUNG:     R1...ZEIGER AUF SCHEDULE
422     /                                                R2...ZEIGER AUF TCB;
423     /                                                R4...SCHEDULE-TYP;
424     /;
425      PROC:START;
426               .TO. STA1 .IF. R4 .NULL.;       KEIN SCHEDULE
427               SCHART,R1:=R2+&SCHPNT;
428               .CALL. SCH;                     SCHEDULE-VERARBEITUNG
429               .RETURN. START;
430          LOC:STA1;
431               .TO. LOOK .IFNOT. STATUS,R2 .NULL.;
432               STATUS,R2:='B1000';
433               PC,R2:=STPC,R2;                 PC INITIALISIEREN
434               R1:=&RUNNING;          IN RUNNING-KETTE EINKETTEN
435               .CALL. CHAIN;
436          LOC:LOOK;
437               .RETURN. START;
438      END:START;
439     /;
440     /;
441     /                                REQUEST;
442     /                                =======;
443     /;
444     /                                VERSORGUNG: R2...ZEIGER AUF SEMA;
445     /;
446      PROC:REQU;
447               R1:=R2;                         SEMA-ADRESSE
448               R2:=RUNNING;
449               .TO. REQU1 .IF. PRIO,R1 .NULL.;
450               PRIO,R1:=PRIO,R1-'K1';
451               .RETURN. REQU;
452          LOC:REQU1;
453               STATUS,R2:='B0001';             TASK UNTERBRECHEN
454               .CALL. UNCH;                    AUSKETTEN AUS PWS
455               .CALL. CHAIN;                   EINKETTEN IN SWS
456               .RETURN. REQU;
457      END:REQU;
458     /;
459     /;
460     /                                RELEASE;
461     /                                =======;
462     /;
463     /                                VERSORGUNG: R2...ZEIGER AUF SEMA;
464     /;
465      PROC:RELE;
466               R1:=R2;                         SEMA-ADRESSE
467               .TO. REL2 .IF. R2:=&0,R1 .EQ. R1;  KETTE LEER
468               R1:=&RUNNING;
469               .CALL. UNCH;                    AUS SEMA-KETTE
470               .CALL. CHAIN;                   IN RUNNING-KETTE
471               STATUS,R2:='B1000';
472               .RETURN. RELE;
473          LOC:REL2;
 74               PRIO,R1:=PRIO,R1+'K1';
 75               .RETURN. RELE;
 76      END:RELE;
 77     /;
```

```
478      /;
479      /                                TERMINATE;
480      /                                =========;
481      /;
482      /                                VERSORGUNG: R2...ZEIGER AUF TCB ODER Ø
483      /;
484       PROC:STOP;
485               R2:=RUNNING;
486               .CALL. UNCH;                    AUS PWS
487               STATUS,R2:='KØ';
488               .RETURN. STOP;
489       END:STOP;
490      /;
491      /;
492      /                                PREVENT;
493      /                                =======;
494      /;
495      /                                VERSORGUNG: R2...ZEIGER AUF TCB ODER Ø
496      /;
497       PROC:PREV;
498               .TO. PREV1 .IFNOT. R2 .NIL.;
499               R2:=RUNNING;
500          LOC:PREV1;
501               R1:=SCHPNT,R2;                  ACT-SCHEDULE ELIMINIEREN
502               .CALL. DEQU;
503               SCHPNT,R2:=&NIL;
504               R1:=SCHRES,R2;                  RESUME-SCHEDULE ELIMINIEREN
505               .CALL. DEQU;
506               SCHRES,R2:=&NIL;
507               .RETURN. PREV;
508       END:PREV;
509      /;
510      /;
511      /                                SUSPEND;
512      /                                =======;
513      /;
514      /                                VERSORGUNG: KEINE;
515      /;
516       PROC:SUSP;
517               R2:=RUNNING;
518               STATUS,R2:='BØ1ØØ';
519               .CALL. UNCH;
520               .RETURN. SUSP;
521       END:SUSP;
522      /;
523      /;
524      /                               CONTINUE;
525      /                               ========;
526      /;
527      /                               VERSORGUNG:     R1...ZEIGER AUF SCHEDUL
528      /                                               R2...ZEIGER AUF TCB ODE
529      /                                               R4...SCHEDULE-TYP;
530      /;
531       PROC:CONT;
532               .TO. CONTØ .IFNOT. R2 .NIL.;
533               R2:=RUNNING;
534          LOC:CONTØ;
535               .TO. CONT1 .IF. R4 .NULL.;      KEIN SCHEDULE
536               SCHART,R1:=R2+&SCHRES;
537               .CALL. SCH;
```

```
538             .RETURN. CONT;
539         LOC:CONT1;
540             .TO. CONT2 .IF. STATUS,R2 .NE. 'B0100';
541             STATUS,R2:='B1000';
542             R1:=ARUNNING;
543             .CALL. CHAIN;
544         LOC:CONT2;
545             .RETURN. CONT;
546       END:CONT;
547      /;
548      /;
549      /                              RESUME;
550      /                              ======;
551      /;
552      /                              VERSORGUNG:     R1...ZEIGER AUF SCHEDULE
553      /                              R2...0;
554      /                              R4...SCHEDULE-TYP;
555      /;
556       PROC:RESU;
557             .TO. RESU1 .IF. R4 .NULL.;
558             SPCB,R1:=R2;=RUNNING;
559             .CALL. CONT;
560             .CALL. SUSP;
561             .RETURN. RESU;
562         LOC:RESU1;
563             .CALL. CONT;
564             .RETURN. RESU;
565       END:RESU;
566      /;
567      /;
568      /                              EINKETTEN;
569      /                              =========;
570      /;
571      /                              VERSORGUNG:     R1...ZEIGER AUF KETTENAN
572      /                                              R2...ZEIGER AUF ELEMENT;
573      /;
574          PROC:CHAIN;
575               R5:=R1;
576               R3:=SUCC,R1;                    NACHFOLGER
577               R6:=PRIO,R2;                    PRIORITAET
578          LOC:LOOP;
579               .TO. IN .IF. R3 .EQ. R5;        KETTENENDE
580               .TO. IN .IF. R6 .LT. PRIO,R3;   HOEHERE PRIO
581               R1:=SUCC,R1;
582               R3:=SUCC,R1;
583               .TO. LOOP;
584          LOC:IN;
585               SUCC,R1:=PRED,R3:=R2;
586               PRED,R2:=R1;
587               SUCC,R2:=R3;
588               .RETURN. CHAIN;
589          END:CHAIN;
590      /;
591      /;
```

```
592       /                                   AUSKETTEN;
593       /                                   =========;
594       /;
595       /                                   VERSORGUNG: R2..ZEIGER AUF ELEMENT;
596       /;
597            PROC:UNCH;
598                 R4:=R3:=PRED,R2;
599                 R3:=SUCC,R3:=SUCC,R2;
600                 PRED,R3:=R4;
601                 .RETURN. UNCH;
602            END:UNCH;
603       /;
604        PROC: EXOPE;
605                 .TO. XSTAR .IF. R3 .EQ. 'K1';
606                 .TO. XREQU .IF. R3 .EQ. 'K2';
607                 .TO. XRELE .IF. R3 .EQ. 'K3';
608                 .TO. XSTOP .IF. R3 .EQ. 'K4';
609                 .TO. XPREV .IF. R3 .EQ. 'K5';
610                 .TO. XSUSP .IF. R3 .EQ. 'K6';
611                 .TO. XCONT .IF. R3 .EQ. 'K7';
612                 .TO. XRESU .IF. R3 .EQ. 'K8';
613                 .TO. XDRIV .IF. R3 .EQ. 'K9';
614                 .TO. XFEHL ;
615                 LOC:XSTAR;
616                 .CALL. START;
617                 .TO. XEND;
618                 LOC:XREQU;
619                 .CALL. REQU;
620                 .TO. XEND;
621                 LOC: XRELE;
622                 .CALL. RELE;
623                 .TO. XEND;
624                 LOC: XSTOP;
625                 .CALL. STOP;
626                 .TO. XEND;
627                 LOC: XPREV;
628                 .CALL. PREV;
629                 .TO. XEND;
630                 LOC: XSUSP;
631                 .CALL. SUSP;
632                 .TO. XEND;
633                 LOC: XCONT;
634                 .CALL. CONT;
635                 .TO. XEND;
636                 LOC:XRESU;
637                 .CALL. RESU;
638                 .TO. XEND;
639                 LOC:XDRIV;
640                 .CALL. DRIV;
641                 .TO. XEND;
642                 LOC: XFEHL; FEHLER AUSGANG
643                 LOC:XEND;
644                 .RETURN. EXOPE;
645        END: EXOPE;
646       /;
647        FINISH;
/////////
```

Anhang V

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:
QUELLDATEI: CODE          ABSCHNITT: OBJE     SATZ:    OBJS


                        *** START:
                                'NAME' OBJE
                                'SATZ' OBJS
                                 'PRIV'

                        ********************************************************
                        ***
                        *** ZIELMASCHINENABHAENGIGE INITIALISIERUNG
                        ***
                        ********************************************************

                        *** SPEICHERSTELLEN DIE DER BENUTZEREBENE UND DER
                        *** BEKANNT SEI MUESSEN
                 17991       BRIEF/'IDENT' 17991
                 17992 'VA'  KOOZ/ 'IDENT' 17992
                     5       ASNR/ 'IDENT' 5  *** ASNR DER INTERRUPTEINGAE
                        *** FESTLEGEN DER BENUTZEREBENE
                    15       BENEBE/ 'IDENT' 15    *** BEN. EBENE
                    13       ZEITEB/ 'IDENT' 13       *** ZEITGEBEREBENE
                   240       TZR/  'IDENT' 240    *** ADR. AB DER DIE REG.

                        *** KOOZ VORBESETZEN
  128. 0                'VA'        G1    :=0
  129. 0           17992            (KOOZ):=G1
  131. 0                            G2       :=16380
  133. 0           17993            (KOOZ+1):=G2
  135. 0           17994            (KOOZ+2):=G1
                        ********************************************************
                        *** KOORDINATION MIT BENUTZEREBENE
                        ********************************************************


                            *** BENUTZEREBENE STARTEN
  137. 0               15           R3              := BENEBE
  139. 0  V       144               (BENU)          :=R3

  141. 0                            R7              : US 12    *** PROG. STARTEN
  143. 0                                            ='H=0102'
  144. 0                            BENU/=

                        *** WARTEN BIS BENUTZEREBENE KOOZ ERHOEHT
                        *** SYSTEM ERWARTET IN BRIEFKASTEN ADRESSE DES IN
                        *** FIRSTCB.FIRSTSM.FIRSTIO.LASTIO
                                $KOOR/KOOZ/-:
  150. 0           17991 'VA'       R7              :=(BRIEF)
  152. 0                            R5              :=0
  153. 0           17991            (BRIEF)         :=R5     *** BRIEF ZURUECKSE
  155. 0                        NEXTPA/R6           :=(R7')
  156. 0  V       164               (FIRSTCB+R5 ):=R6
  158. 0                            R5              :=R5+1
  159. 0  V       155               R5<4            :SP NEXTPA
  162. 0  V       168                               : SP GBINT
                            *** DATEN FUER MUNAB. INIT.
  164. 0                'VA' FIRSTCB/=
  165. 0                     FIRSTSM/=
  166. 0                     FIRSTIO/=
  167. 0                     LASTIO/=
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:                    BIA
QUELLDATEI: CODE          ABSCHNITT: OBJE     SATZ:   OBJS

                          ********************************************************
                          *** INITIALISIERUNG DER GERAETETREIBERBLOECKE
                          ********************************************************

168. 0                        GBINT/R3          :=1
169, 0  V    166                    G1          :=(FIRSTIO)
171. 0                        NEU/  R6          :=(3+G1)    *** DRUCKER
173. 0  V    184                    R6#0        : SP BSA
176. 0  V   1209                    R5          :=  DRANZ
178. 0                              (4 WTL +G1) :=R5
180. 0  V    227                   (GDRUCK)     :=G1
182. 0  V    217                                : SP FIN
184. 0  V    195              BSA/  R6#1        : SP BSE    *** BLATTSCHREIBER
187. 0  V   1227                    R5          :=BAANZ
189, 0                              (4 WTL +G1) :=R5
191. 0  V    228                   (GBA )       :=G1
193. 0  V    217                                : SP FIN
195. 0  V    206              BSE/  R6#2        : SP GIAE   *** BLATTSCHREIBER
198, 0  V   1241                    R5          :=BEANZ
200. 0                              (4 WTL +G1) :=R5
202, 0  V    229                   (GBE )       :=G1
204. 0  V    217                                : SP FIN
206. 0  V    224              GIAE/ R6#3        : SP LAPSUS   ***  ANALOGEINGA
209. 0  V   1271                    R5          :=AEANZ
211. 0                              (4 WTL + G1):=R5
213. 0  V    230                   (GAE )       :=G1
215. 0  V    217                                : SP FIN
217. 0                        FIN/  G1          :=G1+5
218. 0  V    167                    G1#=(LASTIC) :SP INTINIT
        V    231
222. 0  V    171                                : SP NEU

224. 0                        LAPSUS/ R7        :=13
225. 0  V    742                                : SP FEHLER

                          *** INFORMATIONEN FUER MAB. INIT.
227. 0                    GDRUCK/=
228, 0                    GBA/=
229. 0                    GBE/=
230. 0                    GAE/=
231. 0                    INTINIT/

                          *** BENUTZEREBENE SPERREN
                          *** ====================
231. 0         241                R3           :=(TZR+1)
233. 0  V      421                (PC)         :=R3
235. 0  V      413                R3           :=BREMSE
237. 0         241                (TZR+1)      :=R3


                          ********************************************************
                          *** INITIALISIERUNG FUER DIE INTERRUPTVORVERARBEITUN
                          ********************************************************

239. 0                              R6          :=0
240. 0                    ALAN/ R7              :US 12  ***ALAN ''5
242. 0                                          := 'H=1005'    *** MIT EXP. WAR
243. 0                    ALANZ/                :=0
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:
QUELLDATEI: CODE          ABSCHNITT: OBJE     SATZ:    OBJS


124    244. 0                                          =5    ***ASNR
125    245. 0                          ALBANZ/         =0     *** BETRIEBSANZEIGEN
126    246. 0      17992                               =KOOZ
127                                    *** PRUEFEN OB DER AUFRUF KORREKT
128    247. 0  V    243                    R3          :=(ALANZ)
129    249. 0  V    297                    R3 ≠ 0      : SP INTEND   *** AUFRUF O
130                                    *** FALLS AUFRUF TAETIG ALARM ABMELDEN
131    252. 0                              R6          :=R6+1
132    253. 0                              R7          : US 12   *** ALAB ''5
133    255. 0                                          = 'H=1043'
134    256. 0                                          :=0
135    257. 0                                          :=5
136    258. 0  V    240                    R6=1        : SP ALAN
137                                    *** FEHLER
138    261. 0                              R7          :=10
139    262. 0  V    742                                : SP FEHLER
140                                    *** TABELLEN FUER INTERRUPTVORVERARBEITUNG
141    264. 0                          INTAB/ =16¢0¢
142    280. 0                          INTZAE/ =17¢0¢
143    297. 0                          INTENC/
144
145                          ************************************************
146                          *** ZEITGEBERINITIALISIERUNG
147                          ************************************************
148
149                                    *** AUSBABE DES BFRAGETEXTES
150    297. 0                          ABFR/ R7 :US 12
151    299. 0                                =H=0886'
152    300. 0                                ='H=0000'
153    301. 0  V    360                      =ABTEA
154    302. 0  V    370                      =ABTEE
155    303. 0                                =0
156    304. 0                                =0
157
158                                    *** WARTEN AUF ANTWORT
159    305. 0                              R7 :US 12
160    307. 0                              ='H=0886'
161    308. 0                              ='H=0000'
162    309. 0  V    371                    = ANTA
163    310. 0  V    372                    = ANTE
164    311. 0                              = 0
165    312. 0                              = 0
166
167                                    *** AUSWERTEN DER ANTWORT
168    313. 0  V    371                    G1          :=(ANTA)
169    315. 0  V    373                    G2          :=(JA)
170    317. 0  V    335                    G1=G2 : SP ZEITJA
171    320. 0  V    374                    G2          :=(NEIN)
172    322. 0  V    402                    G1=G2 : SP TIEND
173
174                                    *** FALSCHE EINGABE
175    325. 0                              R7 : US 12
176    327. 0                              ='H=0886'
177    328. 0                              = 'H=0000'
178    329. 0  V    375                    = FALA
179    330. 0  V    387                    = FALE
180    331. 0                              = 0
181    332. 0                              = 0
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:                          RLA
QUELLDATEI: CODE          ABSCHNITT: OBJE     SATZ:   OBJS


182
183    333. 0  V     297              : SP ABFR
184
185    335. 0                  ZEITJA/
186                            ***  ZEITGEBERPROZEDUR DER REG.TAFEL 13 ZUORDNE
187    335. 0                        R3              :=0
188    336. 0  V     401             (TAKTE)         :=R3
189    338. 0  V     388             R3              :=TIME
190    340. 0  V     352             (STADR)         :=R3
191    342. 0         13             R3:=ZEITEB
192
193    344. 0  V     351             (ZENR1):=R3
194    346. 0  V     356             (ZENR2):=R3
195    348. 0                        R7 : US 12     *** MAKRO 'REGISTERTAFEL ZUO
196    350. 0                        ='H=0303'
197    351. 0                  ZENR1/=
198    352. 0                  STADR/=
199
200    353. 0                        R7 : US 12    *** WECKER EINSCHALTEN
201    355. 0                        ='H=1413'
202    356. 0                  ZENR2/=
203    357. 0                        = 1
204
205    358. 0  V     402             : SP TIEND
206
207                            *** DATEN
208                                  *** ABFRAGETEXT
209    360. 0                        ABTEA/          ='Z=ZEIT EINSCHALTEN J/N'
210    370. 0                        ABTEE/          ='H=0303'
211                                  *** ANTWORT
212    371. 0                        ANTA/           =0
213    372. 0                        ANTE/           ='H=0303'
214                                  *** HILFSGROESSEN
215    373. 0                        JA/             ='Z=J(03)'
216    374. 0                        NEIN/           ='Z=N(03)'
217                                  *** FALSCHE ANTWORT
218    375. 0                        FALA/           ='Z=FALSCHE ANTWORT IIIIIIII'
219    387. 0                        FALE/           ='H=0303'
220
221                       ********************************************************
222                       ***  ZEITGEBERPROZEDUR
223                       ********************************************************
224
225                            *** ZEITGEBERPROZEDUR WIRD DER REG.TAFEL 11 ZUG
226    388. 0                  TIME/
227    388. 0                        R7 : US 12     *** KOOZ ERHOEHEN
228    390. 0                        ='H=0503'
229    391. 0                        ='H=0000'
230    392. 0      17992             =KOOZ
231    393. 0  V     401             R4              :=(TAKTE)
232    395. 0                        R4              :=R4+1
233    396. 0  V     401             (TAKTE)         :=R4
234    398. 0                        R7 : US 12    ***PROG. ENDE
235    400. 0                        ='H=0201'
236                            *** DATEN
237    401. 0                  TAKTE/ =0
238
239    402. 0                  TIEND/
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:
QUELLDATEI: CODE          ABSCHNITT: OBJE     SATZ:   OBJS


 240
 241
 242                        *** BENUTZEREBENE FREIGEBEN
 243                        *** ========================
 244   402. 0   V    421              R3            :=(PC)
 245   404. 0        241              (TZR+1)       :=R3
 246   406. 0                         R7            : US 12     *** SPERRE
 247   408. 0                         ='H=0503'
 248   409. 0                         ='H=0000'
 249   410. 0   V    418              = SPERRE
 250
 251   411. 0   V    929                            :SP INIT    *** SPRUNG IN CA
 252
 253   413. 0                   BREMSE/ R7 : US 12
 254   415. 0                         ='H=0503'
 255   416. 0                         ='H=0001'
 256   417. 0   V    418              = SPERRE
 257   418. 0                   SPERRE/= 0.1000.0
 258   421. 0                   PC/    =
 259
 260
 261                        ************************************************
 262                        *** UNTERPROGRAMM FUER SAVE REGISTERS
 263                        ************************************************
 264
 265   422. 0                   'VA'  SAVE/R3            :=0
 266   423. 0        240              R4            :=TZR
 267   425. 0   V    878              R6            :=(RUNNING)
 268   427. 0        240         NEXTS/ R5            :=(TZR+R3)
 269   429. 0                         (10 +TL + R6):=R5
 270   431. 0                         R6            :=R6+1
 271   432. 0                         R3            :=R3+1
 272   433. 0   V    427              R3# 16        :SP NEXTS
 273   437. 0        241              R6            :=(TZR+1)
 274   439. 0   V    878              R2            :=(RUNNING)+8
 275   442. 0                         (R2)          :=R6
 276   443. 0                          :SP R7
 277
 278
 279
 280                        ************************************************
 281                        ***
 282                        *** V C R V E R A R B E I T U N G
 283                        ***
 284                        ************************************************
 285
 286   444. 0                   WART/ $KOOR/KOOZ/-:
 295                        *** RETTEN DES BRIEFKASTENINHALTS
 296   449. 0      17991              R3            :=(BRIEF)
 297   451. 0   V    468              R3=0          :SP NOSVC
 298                         *** BENUTZERAUFTRAG IN DIE AUFTR.LISTE EINTRA
 299   454. 0   V    473              R4            :=(PCBPOI)
 300   456. 0   V    474              (PCBLIS+R4)   :=R3
 301   458. 0                         R4            :=R4+1
 302   459. 0   V    470              R4#20         :SP PLISVO *** LISTE VOLL
 303   463. 0   V    473             (PCBPOI)       :=R4
 304                         *** KOMMUNIKATIONSZELLE LOESCHEN
 305   465. 0                         R4            :=0
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:                    RIA
QUELLDATEI: CODE        ABSCHNITT: CBJE    SATZ:   OBJS

306   466. 0      17991          (BRIEF)      :=R4
307   468. 0  V     494      NCSVC/           :SP PROALA
308                           *** FEHLER
309   470. 0                 PLISVC/R7        :=14
310   471. 0  V     742                       : SP FEHLER
311                           *** DATEN
312   473. 0                 PCBPOI/=0
313   474. 0                 PCBLIS/=20¢0¢    ***AUFTRAGSLISTE
314
315                       *** PROZESSALARM
316   494. 0  V     243      PROALA/R3        := (ALANZ)
317   496. 0                        R3        :=R3 .U 'H=000R'
318   497. 0  V     573              R3=0      : SP INTRLI  *** KEIN INTERRI
319   500. 0  V     243              R3        :=(ALANZ)  *** ANZEIGEN LEASC
320   502. 0                        R3        := R3 .U 'H=FFF7'
321   504. 0  V     243              (ALANZ)   :=R3
322   506. 0  V     245              R3        :=(ALBANZ)
323   508. 0                        R3        :=R3 .U 'H=FF7F'
324   510. 0  V     245              (ALBANZ)  :=R3
325   512. 0          5              R5        :=ASNR
326   514. 0                        ='H=6056'    *** ALARMWORT LESEN
327                          *** FESTSTELLEN WELCHE INTERRUPTS EINGETROFFEN
328   515. 0                        G5        :=0
329   516. 0                        R5        :=1
330   517. 0                        R7        :=0
331   518. 0  V     546      WELINT/ R7=16    :SP READY
332   522. 0                        R7        :=R7+1
333   523. 0                        G1        :=1
334   524. 0                        G1        :=G1 .U R6
335   525. 0                        R6        :=R6 .V -1
336   526. 0  V     518              G1#1      :SP WELINT
337                                 *** INTERRUPTS IN TABELLE EINTRAGEN
338   529. 0  V     263              G7        :=(INTAB-1+R7)
339   531. 0  V     539              G7#0      : SP INTVER
340   534. 0                        G5        :=G5+1
341   535. 0  V     263              (INTAB-1+R7):=R5
342   537. 0  V     518                        : SP WELINT
343   539. 0  V     279      INTVER/ G7:=(INTZAE-1+R7)  *** VERL. INT. ZAEHL
344   541. 0                        G7        :=G7+1
345   542. 0  V     279              (INTZAE-1+R7):=G7
346   544. 0  V     518                        : SP WELINT
347   546. 0                 READY/ G5        :=G5-1
348                          *** KOORDINIERUNGSZAEHLER ENTSPRECHEND ERHOEHEN
349   547. 0             'VF'
350   547. 0  V     558      KOORER/ G5=¢0    : SP KOOREN
351   550. 0             'VA'
352   550. 0                        R7        : US 12    *** KOOR ERHOEHEN
353   552. 0                        ='H=0503'
354   553. 0                        ='H=0000'
355   554. 0      17992             = KOOZ
356   555. 0                        G5        :=G5-1
357   556. 0  V     547                        :SP KOORER
358   558. 0                 KOOREN/
359                          *** FESTSTELLEN OB DAS 'ORG' BEREITS INTERRUPTS
360   558. 0  V     243              R3        :=(ALANZ)
361   560. 0  V     573              R3#0      : SP INTRLI
362   563. 0  V     296              G7        :=(INTZAE+16 WTL)+1
363   566. 0  V     296              (INTZAE+16 WTL):=G7
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:
QUELLDATEI: CODE          ABSCHNITT: CBJE     SATZ:   CBJS


 364    568. 0                              R3            :=R3 .U 'H=7FDF'
 365    570. 0   V     243                  (ALANZ)       :=R3
 366    572. 0                              ='H=6056' ***ALARMWORT LOESCHEN
 367                                *** INTERRUPTLISTE BEARBEITEN
 368    573. 0                      INTRLI/ G1            :=0
 369    574. 0   V     264          NEXTEL/ R4            :=(INTAB+G1)
 370    576. 0   V     585                  R4=1          : SP LOEINT
 371    579. 0   V     590                  G1=15         : SP ITREND
 372    582. 0                              G1            :=G1+1
 373    583. 0   V     574                                : SP NEXTEL
 374    585. 0                      LOEINT/ R3            :=0    *** INT ALS INTAB STR
 375    586. 0   V     264                  (INTAB+G1)    :=R3
 376    588. 0   V    1345                                : SP ITR    *** SPRUNG IN CA
 377    590. 0                      ITREND/
 378
 379                              *** ZEITINTERRUPT
 380    590. 0   V     401                  R3            :=(TAKTE)
 381    592. 0   V     600                  R3=0          : SP NOTIME
 382    595. 0                              R3            :=R3-1
 383    596. 0   V     401                  (TAKTE)       :=R3
 384    598. 0   V    1356                  : SP TIMER
 385    600. 0                      NOTIME/
 386
 387
 388                              *** GERAETEINTERRUPT
 389                              *** INTERRUPT VOM DRUCKER
 390    600. 0                    'VA'      R6            :=0
 391    601. 0   V    1275           DR/    R5            :=(DRZUST)
 392    603. 0   V     618                  R5=0          :SP BE
 393    606. 0   V    1209                  R4            :=(DRANZ)
 394    608. 0                    'VA'       R4     :=R4 .U 'H=0002'
 395    609. 0   V     618                   R4   = 2 :SP BE
 396    612. 0   V    1275                  (DRZUST)   :=R6
 397    614. 0   V     227                  G3            :=(GDRUCK)
 398    616. 0   V    1291                                : SP DRITR
 399                              *** INTERRUPT VON DER BLATTSCHREIBEREINGABE
 400    618. 0   V    1277           BE/    R5            :=(BEZUST)
 401    620. 0   V     635                  R5 = 0        :SP BA
 402    623. 0   V    1241                  R4            :=(BEANZ)
 403    625. 0                    'VA'       R4     :=R4 .U 'H=0002'
 404    626. 0   V     635                   R4   = 2 :SP BA
 405    629. 0   V    1277                  (BEZUST)      :=R6
 406    631. 0   V     229                  G3            :=(GBE)
 407    633. 0   V    1291                                : SP DRITR
 408                              *** INTERRUPT VON DER BLATTSCHREIBERAUSGABE
 409    635. 0   V    1276           BA/    R5            :=(BAZUST)
 410    637. 0   V     652                  R5 = 0        :SP IAE
 411    640. 0   V    1227                  R4            :=(BAANZ)
 412    642. 0                    'VA'       R4     :=R4 .U 'H=0002'
 413    643. 0   V     652                   R4   = 2 :SP IAE
 414    646. 0   V    1276                  (BAZUST)      :=R6
 415    648. 0   V     228                  G3            :=(GBA)
 416    650. 0   V    1291                  : SP DRITR
 417                              *** INTERRUPT VON DER ANALOGEINGABE
 418    652. 0   V    1278           IAE/   R5            := (AEZUST)
 419    654. 0   V     675                  R5=0          :SP SVCA
 420    657. 0   V    1271                  R4            :=(AEANZ)
 421    659. 0                              R4            :=R4 .U 'H=0002'
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:                          BIA
QUELLDATEI: CODE          ABSCHNITT: OBJE     SATZ:   OBJS

 422    660. 0  V     675              R4=2          :SP SVCA
 423    663. 0  V    1268              R3           :=(LIADR)
 424    665. 0                        R3            :=R3-2
 425    666. 0  V    1270             R2            :=(LASTME)
 426    668. 0                        (R3)          :=R2
 427    669. 0  V    1278             (AEZLST)      :=R6
 428    671. 0  V     230             G3            :=(GAE)
 429    673. 0  V    1291                           : SP DRITR
 430                          *** AUFTRAG VON BENUTZER
 431                              *** AUFTRAGSLISTE WIRD NACH LIFO ABGEARBEITET
 432    675. 0  V     473        SVCA/   R6          :=(PCBPOI)-1
 433    678. 0                 'VF'
 434    678. 0  V     741              R6≠0          :  SP  ALLFEH  *** WARUM KOOZ
 435    681. 0                 'VA'
 436    681. 0  V     473              (PCBPOI)      :=R6
 437    683. 0  V     474              R2            :=(PCBLIS+R6)
 438    685. 0                         R6           :=(3+R2)
 439    687. 0                         G1           :=R2
 440    688. 0  V    1059                 R6 =0      :SP SVC     *** KEIN SCHEDULE
 441    691. 0                         R6           :=(2+R2)
 442    693. 0  V     705                 R6 =1      :SP UMKOA *** ACT. SCHEDULE
 443    696. 0  V     714                 R6 =7      :SP UMKOR      *** CONT. SCHED
 444    699. 0  V     714                 R6 =8      :SP UMKOR     *** RESUME SCHED
 445                                   *** NICHT ZULEI. SCHEDULE
 446    702. 0                         R7           :=11
 447    703. 0  V     742                           : SP FEHLER
 448    705. 0                  UMKOA/ R6           :=0
 449    706. 0                         R3           :=(1+R2)
 450    708. 0                         R4           := R3 + 26
 451    711. 0                         G1           :=R4
 452    712. 0  V     721                           : SP KOP
 453    714. 0                  UMKOR/ R6:=0
 454    715. 0                         R3           :=(1+R2)
 455    717. 0                         R4           :=R3+37
 456    720. 0                         G1           :=R4
 457
 458                                   *** UMKOPIEREN DER SCHEDULE PAR.BLOECKE
 459                                   *** ZEIGER SCHART UND ZEIGER SKETT WERDEN NICHT
 460    721. 0                  KOP/   R6           :=R6+1
 461    722. 0                         R2           :=R2+1
 462    723. 0                         R4           :=R4+1
 463    724. 0                  AB1/   R5           :=(R2')
 464    725. 0                         (R4')        :=R5
 465    726. 0                         R6           :=R6+1
 466    727. 0  V     724              R6≠5 : SP AB1
 467    730. 0                         R2           :=R2+1
 468    731. 0                         R4           :=R4+1
 469    732. 0                         R6           :=R6+1
 470    733. 0                  AB2/   R5           :=(R2')
 471    734. 0                         (R4')        :=R5
 472    735. 0                         R6           :=R6+1
 473    736. 0  V     733              R6≠11 : SP AB2
 474
 475    739. 0  V    1059                           : SP SVC
 476                          *** FEHLER
 477    741. 0                 ALLFEH/R7            :=12
 478    742. 0                  FEHLER/
 479    742. 0                        ='H=AF00'
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:                              R
QUELLDATEI: CODE          ABSCHNITT: OBJE     SATZ:   OBJS

480             ***************************************************
481             ***/                                  SPASS-PEARL-BETRIE
482             ***/                                  ================
483             ***/:
484             ***/                                  SIEMENS-310-VERSIO
485             ***/                                  ROLAND RCESSLER:
486             ***/;
487             ***/:
488             ***     LENGTH: POINTER=1;                          TYPLAENGE
489             ***     LENGTH: INT=1:
490             ***/:
491             ***     STRUCT:CHAIN:                               WARTESCHL
492             ***          ELEM:SUCC=POINTER;                     ZEIGER AU
493             ***          ELEM:PRED=POINTER:                     ZEIGER AU
494             ***     STREND:CHAIN:
495             ***/:
496             ***     STRUCT:DRIVER:                              GERAET
497             ***          ELEM:QUEUE=CHAIN;                      WARTESCHL
498             ***          ELEM:SCP=POINTER:                      ZEIGER AU
499             ***          ELEM:IC=POINTER:,                      ZEIGER AU
500             ***          ELEM:STW=POINTER:                      ADRESSE D
501             ***     STREND:DRIVER:
502             ***/:
503             ***     STRUCT:PARBLCC:                             AUFRUF-PA
504             ***          ELEM:SKETT=POINTER:                    SCHEDULE-
505             ***          ELEM:SPCB=POINTER;                     ZEIGER AU
506             ***          ELEM:PSVC=POINTER:                     ZEIGER AU
507             ***          ELEM:STYP=INT:                         SCHEDULE-
508             ***     STREND:PARBLCC:
509             ***/:
510             ***     STRUCT:SCHED:                               SCHEDULE
511             ***          ELEM:SCPAR=PARBLOC:                    PARAMETER
512             ***          ELEM:SCPRIC=INT:                       'SCHEDLIE
513             ***          ELEM:TYP=INT:                          AKTUELLER
514             ***          ELEM:SCHART=POINTER:                   ZEIGER AU
515             ***          ELEM:REST=INT;                         RESTZEIT
516             ***          ELEM:SBEG=INT;                         INTERRUPT
517             ***          ELEM:SAFT=INT;                         'AFTER'-I
518             ***          ELEM:ZYKL=INT:                         'ALL'-INT
519             ***     STREND:SCHED:
520             ***/:
521             ***     STRUCT:IOBLOC:                              I/O-PARAM
522             ***          ELEM:IOPAR=PARBLOC;                    STANDARD-
523             ***          ELEM:IOBUF=POINTER:                    ZEIGER AU
524             ***     STREND:IOBLOC:
525             ***/:
526             ***     STRUCT:SEMA:                                SEMAPHORE
527             ***          ELEM:SWS=CHAIN:                        SWS
528             ***          ELEM:PRIO=INT:                         WERT
529             ***          ELEM:STATUS=INT:                       ZUSTANDSK
530             ***     STREND:SEMA:
531             ***/:
532             ***     STRUCT:PCB:                                 TASKKONTR
533             ***          ELEM:KOPF=SEMA:                        KOPF
534             ***          ELEM:SCHPNT=POINTER;                   ZEIGER AU
535             ***          ELEM:SCHRES=POINTER:                   ZEIGER AU
536             ***          ELEM:IOB=POINTER:                      ZEIGER AU
537             ***          ELEM:ERROR=INT:                        FEHLERKEN
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSR-V3  33  DATUM:                                  BLA
QUELLDATEI: CODE          ABSCHNITT: OBJE     SATZ:   OBJS
```

```
538                     ***           ELEM:PC=POINTER;                    AKT. BEFEH
539                     ***           ELEM:STPC=POINTER;                  STARTADRES
540                     ***           ELEM:REG1=INT;                      REGISTERRE
541                     ***           ELEM:REG2=INT;                      REGISTERRE
542                     ***           ELEM:REG3=INT;                      REGISTERRE
543                     ***           ELEM:REG4=INT;                      REGISTERRE
544                     ***           ELEM:REG5=INT;                      REGISTERRE
545                     ***           ELEM:REG6=INT;                      REGISTERRE
546                     ***           ELEM:REG7=INT;                      REGISTERRE
547                     ***           ELEM:REG8=INT;                      REGISTERRE
548                     ***           ELEM:REG9=INT;                      REGISTERRE
549                     ***           ELEM:REG10=INT;                     REGISTERRE
550                     ***           ELEM:REG11=INT;                     REGISTERRE
551                     ***           ELEM:REG12=INT;                     REGISTERRE
552                     ***           ELEM:REG13=INT;                     REGISTERRE
553                     ***           ELEM:REG14=INT;                     REGISTERRE
554                     ***           ELEM:REG15=INT;                     REGISTERRE
555                     ***           ELEM:REG16=INT;                     REGISTERRE
556                     ***           ELEM:ASKETT=POINTER;          SCHEDULERETTBERI
557                     ***           ELEM:ASPCB=POINTER;
558                     ***           ELEM:APSVC=POINTER;
559                     ***           ELEM:ASTYP=INT;
560                     ***           ELEM:ASCPRI=INT;
561                     ***           ELEM:ATYP=INT;
562                     ***           ELEM:ASCHAR=POINTER;
563                     ***           ELEM:AREST=INT;
564                     ***           ELEM:ASBEG=INT;
565                     ***           ELEM:ASAFT=INT;
566                     ***           ELEM:AZYKL=INT;
567                     ***           ELEM:RSKETT=POINTER;          CONT. SCH. RETTE
568                     ***           ELEM:RSPCB=POINTER;
569                     ***           ELEM:RPSVC=POINTER;
570                     ***           ELEM:RSTYP=INT;
571                     ***           ELEM:RSCPRI=INT;
572                     ***           ELEM:RTYP=INT;
573                     ***           ELEM:RSCHAR=POINTER;
574                     ***           ELEM:RREST=INT;
575                     ***           ELEM:RSBEG=INT;
576                     ***           ELEM:RSAFT=INT;
577                     ***           ELEM:RZYKL=INT;
578                     ***     STREND:PCB;
579                     ***/;
580                     ***/                                  DATEN;
581                     ***/                                  =====;
582                     ***/;
583                     ***     SPACE:ORGSCH=SCHED+;                        INITIALSCHE
584                     ***           SPEL=ANIL+;
585   743. 0                    'VA' ORGSCH   /=     1 ¢0      ≠
586                     ***           SPEL=AFIRSTCB+;
587   744. 0  V     164 'VA'            =     1¢FIRSTCB≠
588                     ***           SPEL='K1'+;
589   745. 0                    'VA'      =     1¢1       ≠
590                     ***           SPEL='K0'+;
591   746. 0                    'VA'      =     1¢0       ≠
592                     ***           SPEL='KC';
593   747. 0                    'VA'      =     1¢0       ≠
594             V     754 'VA' 'AP'/   'IDENT' ORGSCH+ 11
595                     ***/;
```

```
SIEMENS ASS-SPRACHE 320 /330 :  ASSB-V3  33  DATUM:                              RL
QUELLDATEI: CODE          ABSCHNITT: OBJE     SATZ:    OBJS

598                       ***     SPACE:ILANG=INT;                       LAENGE DE
597                       ***           SPEL='K20';
598   754. 0              'VA' ILANG     /=      1 ¢20       ¥
599           V    755    'VA' 'AP'/    'IDENT' ILANG + 01
600                       ***     SPACE:TLANG=INT;                       LAENGE DE
601                       ***           SPEL='K100';
602   755. 0              'VA' TLANG     /=      1 ¢100      ¥
603           V    756    'VA' 'AP'/    'IDENT' TLANG + 01
604                       ***/                                RAM-DATEN;
605                       ***/                                =========;
606                       ***/;
607                       ***     SPACE:ITRLIST=20*POINTER;              INTERRUPT
608   756. 0              'VA' ITRLIST   /=    020 ¢¥
609                       ***/;
610                       ***     SPACE:TIMLST=100*POINTER;              ZEITLISTE
611   776. 0              'VA' TIMLST    /=    100 ¢¥
612                       ***     SPACE:TIMEND=POINTER;
613   876. 0              'VA' TIMEND    /=     01 ¢¥
614                       ***     SPACE:TPOINT=POINTER;
615   877. 0              'VA' TPOINT    /=     01 ¢¥
616                       ***/;
617                       ***     SPACE:RUNNING=PCB;                     PWS-KOPF
618   878. 0              'VA' RUNNING   /=     48 ¢¥
619                       ***/;
620                       ***     SPACE:TLANGH=INT;
621   926. 0              'VA' TLANGH    /=     01 ¢¥
622                       ***     SPACE:THZ=POINTER;
623   927. 0              'VA' THZ       /=     01 ¢¥
624                       ***     SPACE:THZ1=POINTER;
625   928. 0              'VA' THZ1      /=     01 ¢¥
626                       ***/;
627                       ***/;
628                       ***/;
629                       ***/                                INITIALISIERUNGSROU
630                       ***/                                ==================
631                       ***/;
632                       ***/                                  DER CODEGENERATOR
633                       ***/                                  ADRESSDEFINITION
634                       ***/                                  TASKKONTROLLBLOCK
635                       ***/                                  ADRESSDEFINITION
636                       ***/                                  SEMAS:
637                       ***/                                  TREIBER-BLOECKE;
638                       ***/                                  ADRESSDEFINITION
639                       ***/                                  ADRESSDEFINITION
640                       ***     LOC: INIT;
641   929. 0              'VA' INIT      /
642                       ***           / ORGSCH VORBESETZEN;
643                       ***           R1:=ÄORGSCH;
644   929. 0  V    743    'VA'          G0:=¢ORGSCH      ¥
645   931. 0              'VA'     G1                    :=G0
646                       ***           SPCB.R1:=FIRSTCB;
647   932. 0  V    164    'VA'          G0:=(FIRSTCB )
648   934. 0              'VA'     (01   ,WTL+G1)       :=G0
649                       ***      TPOINT:=ÄTIMLST;
650   936. 0  V    776    'VA'          G0:=¢TIMLST      ¥
651   938. 0  V    877    'VA'     (TPOINT  )            :=G0
652                       ***           R1:= ÄITRLIST;
653   940. 0  V    756    'VA'          G0:=¢ITRLIST     ¥
```

```
SIEMENS ASS-SPRACHE 320 /330 :  ASSB-V3  33  DATUM:                                    RIA
QUELLDATEI: CODE          ABSCHNITT: OBJE     SATZ:    OBJS

 654    942. 0                'VA'     G1                        :=G0
 655                          ***           TLANGH:=TLANG-'K1';
 656    943. 0  V     755     'VA'          G0:=(TLANG    )              -1
 657    946. 0  V     926     'VA'     (TLANGH  )             :=G0
 658                          ***           R2:= ILANG;
 659    948. 0  V     754     'VA'          G0:=(ILANG    )
 660    950. 0                'VA'     G2                        :=G0
 661                          ***           .CALL. INI1;                          INITIALISI
 662    951. 0  V    1042     'VA'             R7   :US     INI1
 663                          ***           R1:= ATIMLST;
 664    953. 0  V     776     'VA'          G0:=#TIMLST    #
 665    955. 0                'VA'     G1                        :=G0
 666                          ***           R2:= TLANG;
 667    956. 0  V     755     'VA'          G0:=(TLANG    )
 668    958. 0                'VA'     G2                        :=G0
 669                          ***           .CALL. INI1;                          INITIALISI
 670    959. 0  V    1042     'VA'             R7   :US     INI1
 671                          ***           R1:= RUNNING:= ARUNNING;
 672    961. 0  V     878     'VA'          G0:=#RUNNING   #
 673    963. 0  V     878     'VA'     (RUNNING )             :=G0
 674    965. 0                'VA'     G1                        :=G0
 675                          ***           PRED.R1:=ARUNNING;
 676    966. 0  V     878     'VA'          G0:=#RUNNING   #
 677    968. 0                'VA'     (01   WTL+G1)          :=G0
 678                          ***           R1:= FIRSTCB;
 679    970. 0  V     164     'VA'          G0:=(FIRSTCB )
 680    972. 0                'VA'     G1                        :=G0
 681                          ***       LOC: INIT1;
 682    973. 0                'VA' INIT1    /
 683                          ***           .TC. SEMAS .IF. R1 .EQ.  FIRSTSM;
 684    973. 0                'VA'          G0:=G1
 685    974. 0  V     165     'VA'             R6:=     (FIRSTSM )
 686    976. 0  V     997     'VA'             G0       = R6   :SP SEMAS
 687                          ***           SUCC.R1:=PRED.R1:=SCHPNT.R1:=SCHRES.R1:=A
 688    979. 0                'VA'          G0:=0
 689    980. 0                'VA'     (05   WTL+G1)          :=G0
 690    982. 0                'VA'     (04   WTL+G1)          :=G0
 691    984. 0                'VA'     (01   WTL+G1)          :=G0
 692    986. 0                'VA'     (00   WTL+G1)          :=G0
 693                          ***           STATUS.R1:= 'B0000';
 694    988. 0                'VA'          G0:=000
 695    989. 0                'VA'     (03   WTL+G1)          :=G0
 696                          ***           R1:=R1+APCB;
 697    991. 0                'VA'          G0:=G1                    +48
 698    994. 0                'VA'     G1                        :=G0
 699                          ***           .TC. INIT1;
 700    995. 0  V     973     'VA'                                  :SP INIT1
 701                          ***/;
 702                          ***       LOC: SEMAS;
 703    997. 0                'VA' SEMAS    /
 704                          ***       LOC: INIT3;
 705    997. 0                'VA' INIT3    /
 706                          ***           .TC. DRIVS .IF. R1 .EQ.  FIRSTIO;
 707    997. 0                'VA'          G0:=G1
 708    998. 0  V     166     'VA'             R6:=     (FIRSTIO )
 709   1000. 0  V    1017     'VA'             G0       = R6   :SP DRIVS
 710                          ***           SUCC.R1:=PRED.R1:=R1;
 711   1003. 0                'VA'          G0:=G1
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:                                  RE
QUELLDATEI: CODE          ABSCHNITT: OBJE     SATZ:    OBJS

 712  1004. 0              'VA'     (01   WTL+G1)     :=G0
 713  1006. 0              'VA'     (00   WTL+G1)     :=G0
 714                       ***          PRIC.R1:=STATUS.R1:              ANFANGSWE
 715  1008. 0              'VA'          G0:=(03   WTL+G1)
 716  1010. 0              'VA'     (02   WTL+G1)     :=G0
 717                       ***          R1:=R1+ASEMA:                    NAECHSTE
 718  1012. 0              'VA'          G0:=G1                    +04
 719  1014. 0              'VA'     G1                :=G0
 720                       ***          .TC. INIT3:
 721  1015. 0  V    997   'VA'                              :SP INIT3
 722                       ***/:
 723                       ***     LOC: DRIVS:
 724  1017. 0              'VA' DRIVS    /
 725                       ***     LOC: INIT4:
 726  1017. 0              'VA' INIT4    /
 727                       ***          .TC. ACTV  .IF. R1 .EQ.  LASTIO:
 728  1017. 0              'VA'          G0:=G1
 729  1018. 0  V    167   'VA'             R6:=     (LASTIO  )
 730  1020. 0  V   1036   'VA'             G0       = R6  :SP ACTV
 731                       ***          SUCC.R1:=PRED.R1:=R1:
 732  1023. 0              'VA'          G0:=G1
 733  1024. 0              'VA'     (01   WTL+G1)     :=G0
 734  1026. 0              'VA'     (00   WTL+G1)     :=G0
 735                       ***          SDP.R1:=ANIL:
 736  1028. 0              'VA'          G0:=0
 737  1029. 0              'VA'     (02   WTL+G1)     :=G0
 738                       ***          R1:=R1+ACRIVER:                  NAECHSTER
 739  1031. 0              'VA'          G0:=G1                    +05
 740  1033. 0              'VA'     G1                :=G0
 741                       ***          .TC. INIT4:
 742  1034. 0  V   1017   'VA'                              :SP INIT4
 743                       ***/:
 744                       ***     LOC: ACTV:
 745  1036. 0              'VA' ACTV     /
 746                       ***          R1:= ACRGSCH:
 747  1036. 0  V    743   'VA'          G0:=¢ORGSCH     ¥
 748  1038. 0              'VA'     G1                :=G0
 749                       ***          .TC. SVC:
 750  1039. 0  V   1059   'VA'                              :SP SVC
 751                       ***/:
 752                       ***     PROC: INI1:                              INITIALIS
 753  1041. 0              'VA' QINI1    /=1¢¥
 754  1042. 0  V   1041   'VA'  INI1    /(QINI1    ):=R7
 755                       ***     LOC: INI2:
 756  1044. 0              'VA' INI2     /
 757                       ***          A0.R1:= R1:                      ZEIGER ZF
 758  1044. 0              'VA'          G0:=G1
 759  1045. 0              'VA'     (G1)              :=G0
 760                       ***          R1:= R1+APCINTER:
 761  1046. 0              'VA'          G0:=G1                    +01
 762  1048. 0              'VA'     G1                :=G0
 763                       ***          .TC. INI2 .IFNOT. R2:= R2-'K1' .NULL.:
 764  1049. 0              'VA'          G0:=G2                    -1
 765  1051. 0              'VA'     G2                :=G0
 766  1052. 0              'VA'             R6:=     0
 767  1053. 0  V   1044   'VA'             G0       :# R6  :SP INI2
 768                       ***          .RETURN. INI1:
 769  1056. 0  V   1041   'VA'             R7:= (QINI1      )
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:                                  RIA1
QUELLDATEI: CODE          ABSCHNITT: OBJE     SATZ:    OBJS

 770  1058. 0              'VA'              :SP  R7
 771                       ***     END: INI1:
 772                       ***/;
 773                       ***/;
 774                       ***/:
 775                       ***/          SUPERVISOR-CALL:
 776                       ***/          ================:
 777                       ***/;
 778                       ***/                                                  R1: ZEIGER
 779                       ***/                                                  PARAMETERBL
 780                       ***/                                                  ERGEBNIS:,
 781                       ***/                                                  R2: ZEIGER
 782                       ***/                                                  R4: NULL. (
 783                       ***/:
 784                       ***     LOC: SVC:
 785  1059. 0              'VA' SVC      /
 786                       ***           DISABLE:
 787                       ***           SAVE REGISTERS;
 788                       ************************************************************
 789                       *** S A V E  R E G I S T E R
 790                       ============================
 791  1059. 0  V     422 'VA'         R7             :US SAVE
 792                       ************************************************************
 793                       ***           R2 := SPCB.R1;                          ZEIGER AUF
 794  1061. 0              'VA'          G0:=(01   WTL+G1)
 795  1063. 0              'VA'     G2                  :=G0
 796                       ***           .TC. SVC1 .IF. R4:= STYP.R1 .NULL.;
 797  1064. 0              'VA'          G0:=(03   WTL+G1)
 798  1066. 0              'VA'     G4                  :=G0
 799  1067. 0              'VA'             R6:=     0
 800  1068. 0  V    1077 'VA'              G0       := R6   :SP SVC1
 801                       ***           TYP.R1 := R4;                           SCHEDULE-TY
 802  1071. 0              'VA'          G0:=G4
 803  1072. 0              'VA'     (05   WTL+G1)      :=G0
 804                       ***           REST.R1:='K0';
 805  1074. 0              'VA'          G0:=0
 806  1075. 0              'VA'     (07   WTL+G1)      :=G0
 807                       ***     LOC: SVC1:
 808  1077. 0              'VA' SVC1     /
 809                       ***           R3 := PSVC.R1;                          ZEIGER AUF
 810  1077. 0              'VA'          G0:=(02   WTL+G1)
 811  1079. 0              'VA'     G3                  :=G0
 812                       ***           .CALL. EXOPE;
 813  1080. 0  V    1915 'VA'              R7   :US     EXOPE
 814                       ***/:
 815                       ***     LOC: ASSIGN:                                  PROZESSOR Z
 816  1082. 0              'VA' ASSIGN   /
 817                       ***           .TC. NOTASK .IF. ARUNNING .EC. RUNNING;
 818  1082. 0  V     878 'VA'          G0:=#RUNNING    #
 819  1084. 0  V     878 'VA'             R6:=     (RUNNING )
 820  1086. 0  V    1121 'VA'             G0       = R6   :SP NOTASK
 821                       ***           RESUME PROCESS:                         TASK AUF
 822                       ************************************************************
 823                       *** R E S U M E  P R O C E S S
 824                       *** ========================
 825  1089. 0              'VA'          R3             :=0
 826                       *** FESTSTELLEN VON WELCHER TASK DIE REGISTER RESTAURI
 827  1090. 0       240             R4             :=TZR
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:                              R
QUELLDATEI: CODE          ABSCHNITT: CBJE     SATZ:   OBJS


 828
 829                        *** RESTAURIEREN DER REGISTER
 830  1092, 0   V    878              R6          :=(RUNNING)
 831  1094, 0          NEXTR/ R5      :=(10 WTL + R6)
 832  1096, 0                 (R4)    :=R5
 833  1097, 0                 R4      :=R4+1
 834  1098, 0                 R6      :=R6+1
 835  1099, 0                 R3      :=R3+1
 836  1100, 0   V   1094              R3<=15      :SP NEXTR
 837  1103, 0   V    878              R6          :=(RUNNING)+8
 838  1106, 0                         R6          :=(R6)
 839  1107, 0         241             (TZR+1)     :=R6
 840  1109, 0   V   1133              R6          :=(HKOOZ)
 841                        *** FESTSTELLEN OB DAS SYSTEM ANGEHALTEN WAR
 842  1111, 0   V   1119              R6 =0:SP WEIT
 843                        *** WENN JA .HKOOZ. ERHOEHEN
 844  1114, 0                         $KOCR/HKOOZ/+:
 853  1119, 0   V    444       WEIT/ :SP WART
 854                        ***/:
 855                        ***     LOC: NCTASK:                          KEINE TAS
 856  1121, 0              'VA' NCTASK  /
 857                        ***           ENABLE:
 858                        ***           HALT:                             WARTEN
 859                        ************************************************
 860                        *** H A L T
 861                        *** ========
 862  1121, 0   V   1127              R6:=HALT
 863                        *** REGISTER R1 (BEFEHLSZAEHLER) DER TASKEBENE ALF
 864                        *** SETZEN
 865  1123, 0         241             (TZR+1):=R6
 866  1125, 0   V    444              :SP WART
 867  1127, 0                 HALT/ $KOCR/HKOOZ/-:    *** EBENE 13 ANHALTEN
 876  1132, 0                       ='H=AF00'         ***FEHLERFEHLER
 877  1133, 0                 HKOOZ/=0.1000.0
 878                        ************************************************
 879                        ***/;
 880                        ***/:
 881                        ***/;
 882                        ***/;
 883                        ***/;
 884                        ***/:
 885                        ***/:
 886                        ***/          STANDARD-TREIBER;
 887                        ***/,         ================:
 888                        ***/:
 889                        *** PROC: DRIV:
 890  1136, 0              'VA' QDRIV    /=1<>
 891  1137, 0   V   1136   'VA'  DRIV    /(QDRIV    ):=R7
 892                        ***           .TO. DFREE .IFNOT. R4 .NIL.:  WENN VON
 893  1139, 0              'VA'          GO:=G4
 894  1140, 0              'VA'             R6:=     0
 895  1141, 0   V   1168   'VA'             GO       # R6  :SP DFREE
 896                        ***           R7:= R2:                          E/A-STRUK
 897  1144, 0              'VA'          GO:=G2
 898  1145, 0              'VA'     G7                    :=GO
 899                        ***           R2:= RUNNING:
 900  1146, 0   V    878   'VA'          GO:=(RUNNING )
 901  1148, 0              'VA'     G2                    :=GO
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:                              RLA
QUELLDATEI: CODE          ABSCHNITT: OBJE     SATZ:    OBJS


 902                          ***          IOB.R2:= IOBUF.R1;                 PUFFERADRE
 903   1149. 0                'VA'         G0:=(04   WTL+G1)
 904   1151. 0                'VA'    (06   WTL+G2)     :=G0
 905                          ***          .CALL. UNCH;                       ALS PROZESS
 906   1153. 0  V   1897      'VA'           R7  :US    UNCH
 907                          ***          R1:= R7;
 908   1155. 0                'VA'         G0:=G7
 909   1156. 0                'VA'    G1                 :=G0
 910                          ***          .IO. DFREE .IF. SDP.R1 .NIL.; GERAET FREI
 911   1157. 0                'VA'         G0:=(02   WTL+G1)
 912   1159. 0                'VA'           R6:=    0
 913   1160. 0  V   1168      'VA'           G0      =R6  :SP DFREE
 914                          ***          .CALL. CHAIN;                      IN GWS
 915   1163. 0  V   1853      'VA'           R7  :US  CHAIN
 916                          ***          .RETURN. DRIV;
 917   1165. 0  V   1136      'VA'           R7:= (GDRIV      )
 918   1167. 0                'VA'           :SP  R7
 919                          ***/;
 920                          ***    LOC: DFREE;
 921   1168. 0                'VA' DFREE   /
 922                          ***          SDP.R1:= R2;                       AUSFUEHRENI
 923   1168. 0                'VA'         G0:=G2
 924   1169. 0                'VA'    (02   WTL+G1)     :=G0
 925                          ***          DOIO;                                GERAETEAI
 926
 927                          ************************************************************
 928                          *** D O I O
 929                          *** ========
 930                          ***
 931                          *** G1: INHALT=ADRESSE DES GERAETEBLOCKS
 932                          *** 4.TES WORT IN GB ENTHAELT ZEIGER AUF IO-ROUTINE
 933                          *** G2: TASK, DIE DEN AUFTRAG GEGEBEN HAT
 934                          *** INHALT 5.TES WORT IN GR=PUFFERANFANGSADRESSE
 935                          *** INHALT 1.TES WORT IN PUFFER = PUFFERENDADRESSE
 936   1171. 0  V   1280              R4           :=SCHMAR
 937   1173. 0       241              (TZR+1)      :=R4
 938
 939                          *** FESTSTELLEN MIT WELCHEM GERAET EIN TRANSFER DURCH
 940                          *** WERDEN SOLL
 941   1175. 0                'VA'         R4           :=1
 942   1176. 0                             R5           :=0
 943   1177. 0                             R6           :=(6+G2)     *** PUFFERANF.ADR
 944   1179. 0                             R7           :=(3+G1)     *** GERAETEKENNUNG
 945   1181. 0                             R3           :=(R6)
 946   1182. 0  V   1195      *           R7 = 0       :SP DRUCK
 947   1185. 0  V   1213                  R7 = 1       :SP BSAUS
 948   1188. 0  V   1231                  R7 = 2       :SP BSEIN
 949   1191. 0  V   1253                  R7 = 3       : SP ANEIN
 950   1194. 0                                         :SP R7
 951
 952                          *** DRUCKERAUSGABE
 953   1195. 0                 DRUCK/ R6           :=R6+1
 954   1196. 0  V   1206              (DRANF)      :=R6
 955   1198. 0  V   1207              (DREND):=R3
 956   1200. 0  V   1275              (DRZLST)     :=R4
 957   1202. 0                    $BSAA/2/ / / /A2@DRANF//A3@DREND//AANZ@DRANZ//KOC
 972   1211. 0  V   1284                          :SP ENDE
 973
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:
QUELLDATEI: CODE          ABSCHNITT: OBJE     SATZ:    OBJS


 974                          *** BLATTSCHREIBERAUSGABE
 975  1213. 0                      BSAUS/ R6           :=R6+1
 976  1214. 0  V   1224                  (BAANF )     :=R6
 977  1216. 0  V   1225                  (BAEND)       :=R3
 978  1218. 0  V   1276                  (BAZUST):=R4
 979  1220. 0                     $BSAA/0/ / / /A2@BAANF//A3@BAEND//AANZ@BAANZ//
 994  1229. 0  V   1284                              :SP ENDE
 995
 996                          *** BLATTSCHREIBEREINGABE
 997  1231. 0                      BSEIN/ R6           :=R6+1
 998  1232. 0  V   1242                   (BEANF):=R6
 999  1234. 0  V   1243                  (BEEND)      := R3
1000  1236. 0  V   1277                  (BEZUST)     :=R4
1001  1238. 0                     BLAUS/
1002  1238. 0                     $BEDI/ / /A1@BEANF//A2@BEEND//AANZ@BEANZ//KOCR
1014                                     *** ADRESSES LETZT. ZEICHENS AN DIE ERSTE
1015                                     *** DATENPUFFERS (ENDEADR. WIRD DADURCH
1016  1246. 0  V   1244                  GO           :=(BLAUS+6)-1
1017  1249. 0                            R6           :=R6-1
1018  1250. 0                            (R6)           :=GO
1019  1251. 0  V   1284                              :SP ENDE
1020                          *** ANALOGEINGABE
1021  1253. 0                    ANEIN/  R2           :=(1+R6)
1022  1255. 0  V   1269                  (ANZAHL  )   :=R2
1023  1257. 0  V   1270                  (LASTME)    :=R6
1024  1259. 0                            R6           :=R6+2
1025  1260. 0  V   1268                 (LIADR)       :=R6
1026  1262. 0  V   1278                 (AEZUST)      :=R4
1027  1264. 0                            R7           :US 12
1028  1266. 0                            ='H=1107'
1029  1267. 0                            ='H=0000'
1030  1268. 0                       LIADR/=
1031  1269. 0                       ANZAHL/=
1032  1270. 0                       LASTME/=
1033  1271. 0                       AEANZ/=
1034  1272. 0      17992                 = KOCZ
1035  1273. 0  V   1284                              :SP ENDE
1036                          *** AN DIESER STELLE KOENNEN WEITERE GERAETE ANGESC
1037                              *** GERAETEZUSTAND
1038  1275. 0                 'VA' DRZUST/=0
1039  1276. 0                      BAZUST/=0
1040  1277. 0                      BEZUST/=0
1041  1278. 0                      AEZUST/=0
1042  1279. 0                      VERL/= 0
1043  1280. 0                      SCHMAR/GO           :=0
1044  1281. 0                      SIELOO/GO           :=GO+1
1045  1282. 0  V   1281                              : SP SIELOO
1046  1284. 0                      ENDE/ ***ENDE VON DOIO
1047  1284. 0       248                 GO           :=(TZR+8)
1048  1286. 0  V   1279                (VERL)        :=GO
1049                          ********************************************************
1050                          ***          .RETURN. DRIV;
1051  1288. 0  V   1136 'VA'              R7:= (QDRIV      )
1052  1290. 0           'VA'               ;SP  R7
1053                          *** END: DRIV;
1054                          ***/;
1055                          ***/;
1056                          ***/          STANDARD-ITR-BEARBEITUNG;
```

```
1057                    ***/          ======================;
1058                    ***/          (FLER GERAETE);
1059                    ***/;
1060                    ***/          VERSORGUNG: R3..ADRESSE DES GERAETES;
1061                    ***/;
1062                    ***       LOC: DRITR;
1063  1291. 0           'VA' DRITR    /
1064                    ***           SAVE ALL REGISTERS;
1065                    ************************************************************
1066                    *** S A V E  R E G I S T E R
1067                    ***=========================
1068  1291. 0  V   422 'VA'          R7            :US SAVE
1069                    ************************************************************
1070                    ***           R1:= ARUNNING;
1071  1293. 0  V   878 'VA'           G0:=<RUNNING   >
1072  1295. 0          'VA'      G1                    :=G0
1073                    ***           .TC. ASSIGN .IF. R2:=SDP.R3 .NIL.;
1074  1296. 0          'VA'           G0:=(02   !WTL+G3)
1075  1298. 0          'VA'      G2                    :=G0
1076  1299. 0          'VA'             R6:=     0
1077  1300. 0  V  1082 'VA'             G0       := R6  ;SP ASSIGN
1078                    ***           R7:=STW.R3;                          ADRESSE DES
1079  1303. 0          'VA'           G0:=(04   !WTL+G3)
1080  1305. 0          'VA'      G7                    :=G0
1081                    ***           ERROR.R2:=A0.R7;                     STATUSWORT
1082  1306. 0          'VA'           G0:=(G7)
1083  1307. 0          'VA'      (07   !WTL+G2)        :=G0
1084                    ***/;
1085                    ***       LOC: DRI1;
1086  1309. 0          'VA' DRI1     /
1087                    ***/;
1088                    ***           R7:= R3;
1089  1309. 0          'VA'           G0:=G3
1090  1310. 0          'VA'      G7                    :=G0
1091                    ***           SDP.R3:= ANIL;
1092  1311. 0          'VA'           G0:=0
1093  1312. 0          'VA'      (02   !WTL+G3)        :=G0
1094                    ***           .CALL. CHAIN;
1095  1314. 0  V  1853 'VA'             R7  :US    CHAIN
1096                    ***           R1:= R7;
1097  1316. 0          'VA'           G0:=G7
1098  1317. 0          'VA'      G1                    :=G0
1099                    ***           .TC. ASSIGN .IF. R2:=SUCC.R1 .EQ. R1;
1100  1318. 0          'VA'           G0:=(00   !WTL+G1)
1101  1320. 0          'VA'      G2                    :=G0
1102  1321. 0          'VA'             R6:=     G1
1103  1322. 0  V  1082 'VA'             G0       := R6  ;SP ASSIGN
1104                    ***           .CALL. UNCH;
1105  1325. 0  V  1897 'VA'             R7  :US    UNCH
1106                    ***           .CALL. DRIV;
1107  1327. 0  V  1137 'VA'             R7  :US    DRIV
1108                    ***           .TC. ASSIGN;
1109  1329. 0  V  1082 'VA'                        ;SP ASSIGN
1110                    ***/;
1111                    ***/                                          BEFEHLSAUSF
1112                    ***       LOC: REPT;
1113  1331. 0          'VA' REPT     /
1114                    ***/;
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:
QUELLDATEI: CODE          ABSCHNITT: OBJE     SATZ:   OBJS

1115                        ***/          KANN VON GERAETEABHAENGIGEM TEIL ANGES
1116                        ***/          ANSTOSS ZU WIEDERHOLEN IST (Z.B. WENN
1117                        ***/:
1118                        ***           .TC. ASSIGN .IF. R2:=SDP.R3 .NIL.:
1119  1331. 0               'VA'          G0:=(02   WTL+G3)
1120  1333. 0               'VA'     G2                    :=G0
1121  1334. 0               'VA'             R6:=     0
1122  1335. 0  V   1082     'VA'             G0       = R6   :SP ASSIGN
1123                        ***           R4:=R1:=R3:                        GERAET
1124  1338. 0               'VA'          G0:=G3
1125  1339. 0               'VA'     G1                    :=G0
1126  1340. 0               'VA'     G4                    :=G0
1127                        ***           .CALL. DRIV:
1128  1341. 0  V   1137     'VA'             R7   :US    DRIV
1129                        ***           .TC. ASSIGN:
1130  1343. 0  V   1082     'VA'                              :SP ASSIGN
1131                        ***/;
1132                        ***/;
1133                        ***/:
1134                        ***/          'PRCZESS'-INTERRUPT;
1135                        ***/          ==================:
1136                        ***/                                          R1: INTER
1137                        ***/:
1138                        ***     LOC: ITR:
1139  1345. 0               'VA' ITR      /
1140                        ***           SAVE ALL REGISTERS:
1141                        ***************************************************
1142                        *** S A V E R E G I S T E R
1143                        ***=======================
1144  1345. 0  V    422     'VA'        R7          :US SAVE
1145                        ***************************************************
1146                        ***           R1:= R1*APOINTER:
1147  1347. 0               'VA'          G0:=G1
1148  1348. 0               'VA'     G1                    :=G0
1149                        ***           THZ:= AITRLIST + R1:          ZEIGER AI
1150  1349. 0  V    756     'VA'          G0:=#ITRLIST    #          +G1
1151  1352. 0  V    927     'VA'     (THZ      )          :=G0
1152                        ***           .TC. TIM1:
1153  1354. 0  V   1378     'VA'                              :SP TIM1
1154                        ***/;
1155                        ***/:
1156                        ***/          TIMER;
1157                        ***/          =====:
1158                        ***/:
1159                        ***     LOC: TIMER:
1160  1356. 0               'VA' TIMER    /
1161                        ***           SAVE ALL REGISTERS:
1162                        ***************************************************
1163                        *** S A V E R E G I S T E R
1164                        ***=======================
1165  1356. 0  V    422     'VA'        R7          :US SAVE
1166                        ***************************************************
1167                        ***           TPOINT:= APOINTER + TPOINT:   SCHEDULE-
1168  1358. 0  V    877     'VA'          G0:=01                  +(TPOINT  )
1169  1361. 0  V    877     'VA'     (TPOINT  )          :=G0
1170                        ***           .TC. TIM2 .IF. ATIMEND .NE. TPOINT:
1171  1363. 0  V    876     'VA'          G0:=#TIMEND    #
1172  1365. 0  V    877     'VA'             R6:=     (TPOINT  )
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:                              BIA
QUELLDATEI: CODE          ABSCHNITT: CRJE    SATZ:   OBJS


1173   1367. 0  V   1374 'VA'            GO      # R6 ,:SP TIM2
1174                    ***            TPCINT:= ATIMLST:                  WENN LISTE
1175   1370. 0  V    776 'VA'            GO:=#TIMLST    #
1176   1372. 0  V    877 'VA'      (TPOINT   )          :=GO
1177                    ***      LOC: TIM2:
1178   1374. 0            'VA' TIM2     /
1179                    ***            TH2:= TPCINT:
1180   1374. 0  V    877 'VA'            GO:=(TPOINT   )
1181   1376. 0  V    927 'VA',     (TH2      )          :=GO
1182                    ***/:
1183                    ***      LOC: TIM1:
1184   1378. 0            'VA' TIM1     /
1185                    ***            TH21:= A0.TH2:
1186   1378. 0  V    927              R7:= (TH2      )
1187   1380. 0            'VA'         GO:=(R7)
1188   1381. 0  V    928 'VA'      (TH21     )          :=GO
1189                    ***      LOC: TIM3:
1190   1383. 0            'VA' TIM3     /
1191                    ***            .TC. ASSIGN .IF. R1:=TH21 .EQ. TH2:
1192   1383. 0  V    928 'VA'            GO:=(TH21     )
1193   1385. 0            'VA'     G1                  :=GO
1194   1386. 0  V    927 'VA'            R6:=     (TH2      )
1195   1388. 0  V   1082 'VA'            GO       =,R6, :SP ASSIGN
1196                    ***            TH21:= SKETT.R1:
1197   1391. 0            'VA'         GO:=(00   'WTL+G1)
1198   1393. 0  V    928 'VA'      (TH21     )          :=GO
1199                    ***            R2:= SPCB.R1:                      ZEIGER AUF
1200   1395. 0            'VA'         GO:=(01   'WTL+G1)
1201   1397. 0            'VA'     G2                    :=GO
1202                    ***            .CALL. SCH:                        SCHEDULE-VE
1203   1398. 0  V   1410 'VA'            R7  :US     SCH
1204                    ***            .TC. TIM3 .IF. R3 .NTI.:           KEIN STATEM
1205   1400. 0            'VA'         GO:=G3
1206   1401. 0            'VA'            R6:=     0
1207   1402. 0  V   1383 'VA'            GO       =,R6  :SP TIM3
1208                    ***            .CALL. EXCPE:
1209   1405. 0  V   1915 'VA'            R7  :US     EXCPE
1210                    ***            .TC. TIM3:
1211   1407. 0  V   1383 'VA'                              :SP TIM3
1212                    ***/:
1213                    ***/:
1214                    ***/:
1215                    ***/           SCHEDULE-VERARBEITUNG:
1216                    ***/,          ====================:
1217                    ***/:
1218                    ***/           EINGABE:
1219                    ***/           R1: ZEIGER AUF SCHEDULE:
1220                    ***/           R2: ZEIGER AUF PCB BZW. SEMA:
1221                    ***/:
1222                    ***/           VERAENDERT:
1223                    ***/           R2: HILFSPROZESSE, WENN KEINE OPERATION AN
1224                    ***/           R3: ADRESSE DER OPERATION, WENN ANZUSTOSS
1225                    ***/           R4: HILFSREGISTER:
1226                    ***/           R5: HILFSREGISTER:
1227                    ***/,          R6: SCHEDULE-PRIORITAET:
1228                    ***/:
1229                    *** PROC: SCH:
1230   1409. 0            'VA' QSCH     /=1#
```

```
SIEMENS ASS-SPRACHE 320 /330 :  ASSB-V3  33  DATUM:                                  R
QUELLDATEI: CODE          ABSCHNITT: OBJE     SATZ:    OBJS

1231  1410. 0  V   1409 'VA'  SCH     /(QSCH     ):=R7
1232                      ***           .TC. SCHNF .IFNOT. REST.R1 .NULL.:
1233  1412. 0             'VA'          G0:=(07   !WTL+G1)
1234  1414. 0             'VA'             R6:=     0
1235  1415. 0  V   1523 'VA'             G0,      # R6  :SP SCHNF
1236                      ***     LOC: SCHC0:
1237  1418. 0             'VA' SCHOO    /
1238                      ***           .TC. SCHOP .IF. TYP.R1 .EQ. 'K4':
1239  1418. 0             'VA'          G0:=(05   !WTL+G1)
1240  1420. 0             'VA'             R6:=     4
1241  1421. 0  V   1455 'VA'             G0       = R6  :SP SCHOP
1242                      ***           R5:= R1:                              REGISTER
1243  1424. 0             'VA'          G0:=G1
1244  1425. 0             'VA'      G5                    :=G0
1245                      ***           R6:= SCHART.R1:
1246  1426. 0             'VA'          G0:=(06   !WTL+G1)
1247  1428. 0             'VA'      G6                    :=G0
1248                      ***           R1:= A0.R6:                           ZEIGER AU
1249  1429. 0             'VA'          G0:=(G6)
1250  1430. 0             'VA'      G1                    :=G0
1251                      ***           .CALL. DEQL:                          ALTEN SCH
1252  1431. 0  V   1585 'VA'             R7  :US    DEQU
1253                      ***           A0.R6:= R1:= R5:                      ZEIGER AU
1254  1433. 0             'VA'          G0:=G5
1255  1434. 0             'VA'      G1                    :=G0
1256  1435. 0             'VA'      (G6)                  :=G0
1257                      ***           .TC. SCHWH .IF. TYP.R1 .GE. 'K12':
1258  1436. 0             'VA'          G0:=(05   !WTL+G1)
1259  1438. 0             'VA'             R6:=     12
1260  1439. 0  V   1485 'VA'             G0       #=R6  :SP SCHWH
1261                      ***           .TC. SCHAF .IF. TYP.R1 .GE. 'K8':
1262  1442. 0             'VA'          G0:=(05   !WTL+G1)
1263  1444. 0             'VA'             R6:=     8
1264  1445. 0  V   1504 'VA'             G0       #=R6  :SP SCHAF
1265                      ***/:
1266                      ***           R3:= ZYKL.R1:                         ALL-SCHED
1267  1448. 0             'VA'          G0:=(10   !WTL+G1)
1268  1450. 0             'VA'      G3                    :=G0
1269                      ***           .CALL. ENQ:                           NEU EINKE
1270  1451. 0  V   1533 'VA'             R7  :US    ENQ
1271                      ***           .TC. SCHC1:
1272  1453. 0  V   1474 'VA'                            :SP SCHO1
1273                      ***/:
1274                      ***     LOC: SCHOP:                                 OPERATION
1275  1455. 0             'VA' SCHOP    /
1276                      ***           .TC. SCHC2 .IF. STYP.R1 .LT. 'K12':
1277  1455. 0             'VA'          G0:=(03   !WTL+G1)
1278  1457. 0             'VA'             R6:=     12
1279  1458. 0  V   1467 'VA'             G0       #=R6  :SP SCHO2
1280                      ***           .TC. SCHO1 .IF. PSVC.R1 .NE. ARESUME:
1281  1461. 0             'VA'          G0:=(02   !WTL+G1)
1282  1463. 0             'VA'             R6:=     12
1283  1464. 0  V   1474 'VA'             G0,      # R6  :SP SCHO1
1284                      ***     LOC: SCHC2:
1285  1467. 0             'VA' SCHO2    /
1286                      ***           .CALL. DEQL:
1287  1467. 0  V   1585 'VA'             R7  :US    DEQU
1288                      ***           R3:= SCHART.R1:                       ZEIGER AU
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:                                  RIA
QUELLDATEI: CODE          ABSCHNITT: OBJE     SATZ:    OBJS

1289   1469. 0                  'VA'          G0:=(06   WTL+G1)
1290   1471. 0                  'VA'     G3                  :=G0
1291                            ***           A0.R3:= ANIL;
1292   1472. 0                  'VA'          G0:=0
1293   1473. 0                  'VA'     (G3)                :=G0
1294                            ***      LOC: SCH01:
1295   1474. 0                  'VA' SCH01    /
1296                            ***           R6:= SCPRIC.R1;                       PRIORITAET
1297   1474, 0                  'VA'          G0:=(04   WTL+G1)
1298   1476. 0                  'VA'     G6                  :=G0
1299                            ***           R3:= PSVC.R1;                         ADRESSE DES
1300   1477, 0                  'VA'          G0:=(02   WTL+G1)
1301   1479. 0                  'VA'     G3                  :=G0
1302                            ***           R4:= 'KO';                            KENNUNG: OI
1303   1480. 0                  'VA'          G0:=0
1304   1481. 0                  'VA'     G4                  :=G0
1305                            ***           .RETURN. SCH:
1306   1482, 0  V    1409       'VA'             R7:= (QSCH       )
1307   1484. 0                  'VA'             :SP  R7
1308                            ***/:
1309                            ***      LOC: SCHWH:                           WHEN-SCHEDU
1310   1485. 0                  'VA' SCHWH    /
1311                            ***           TYP.R1:= TYP.R1 - 'KR';               TYP WEITER
1312   1485, 0                  'VA'          G0:=(05   WTL+G1)       -8
1313   1488. 0                  'VA'     (05   WTL+G1)     :=G0
1314                            ***           R2:= SBEG.R1*APOINTER;                INTERRUPTNU
1315   1490. 0                  'VA'          G0:=(08   WTL+G1)
1316   1492. 0                  'VA'     G2                  :=G0
1317                            ***           R4:=AITRLIST+R2;
1318   1493. 0  V     756       'VA'          G0:=#ITRLIST    #       +G2
1319   1496. 0                  'VA'     G4                  :=G0
1320                            ***           SKETT.R1:=R4;
1321   1497, 0                  'VA'          G0:=G4
1322   1498. 0                  'VA'     (00   WTL+G1),    :=G0
1323                            ***           A0.R4:=R1;
1324   1500. 0                  'VA'          G0:=G1
1325   1501. 0                  'VA'     (G4)                :=G0
1326                            ***           .TC. SCH3:
1327   1502. 0  V    1518       'VA'                              :SP SCH3
1328                            ***/:
1329                            ***      LOC: SCHAF:                           AFTER-SCHED
1330   1504. 0                  'VA' SCHAF    /
1331                            ***           TYP.R1:= TYP.R1 - 'K4';               TYP WEITERS
1332   1504, 0                  'VA'          G0:=(05   WTL+G1)       -4
1333   1507. 0                  'VA'     (05   WTL+G1)     :=G0
1334                            ***           .TC. SCH00 .IF. R3:=SAFT.R1 .NULL.;   W
1335   1509. 0                  'VA'          G0:=(09   WTL+G1)
1336   1511, 0                  'VA'     G3                  :=G0
1337   1512, 0                  'VA'        R6:=     0
1338   1513. 0  V    1418       'VA'        G0       := R6   :SP SCH00
1339                            ***           .CALL. ENQ;                           SCHEDULE F1
1340   1516. 0  V    1533       'VA'          R7   :US     ENQ
1341                            ***/:
1342                            ***      LOC: SCH3:
1343   1518. 0                  'VA' SCH3     /
1344                            ***           R3:= ANIL;                            KEINE OPERA
1345   1518, 0                  'VA'          G0:=0
1346   1519. 0                  'VA'     G3                  :=G0
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:                                  BI
QUELLDATEI: CODE          ABSCHNITT: OBJE     SATZ:   OBJS
```

```
1347                          ***           .RETURN. SCH:
1348  1520. 0   V    1409 'VA'                R7:= (GSCH       )
1349  1522. 0              'VA'                :SP  R7
1350                          ***/:
1351                          ***     LOC:SCHNF:                                    ZEIT WAR
1352  1523. 0              'VA' SCHNF   /
1353                          ***           .CALL. DEQL:                            SCHEDULE
1354  1523. 0   V    1585 'VA'                R7  :US    DEQU
1355                          ***           R3:=REST.R1:                            VERBLEIBE
1356  1525. 0              'VA'             G0:=(07  !WTL+G1)
1357  1527. 0              'VA'     G3                    :=G0
1358                          ***           .CALL. ENQ:                             SCHEDULE
1359  1528. 0   V    1533 'VA'                R7  :US    ENQ
1360                          ***           .TC. SCH3:
1361  1530. 0   V    1518 'VA'                                  :SP SCH3
1362                          *** END: SCH:
1363                          ***/:
1364                          ***/:
1365                          ***/          EINKETTEN EINES SCHEDULES IN ZEITLISTE;
1366                          ***/          ======================================;
1367                          ***/:
1368                          ***/          EINGABE:
1369                          ***/          R3 RELATIVE ZEIT:
1370                          ***/:
1371                          ***/          VERAENDERT:
1372                          ***/:
1373                          ***/          R4: HILFSREGISTER;
1374                          ***/          R3: HILFSREGISTER:
1375                          *** PROC: ENQ:
1376  1532. 0              'VA' QENQ    /=1¢¢
1377  1533. 0   V    1532 'VA'  ENQ     /(QENQ     ):=R7
1378                          ***           R4:=TLANGH:
1379  1535. 0   V     926 'VA'             G0:=(TLANGH   )
1380  1537. 0              'VA'     G4                    :=G0
1381                          ***           .TC. ENQ2 .IF. R3 .GT. R4:    ZEITANGAB
1382  1538. 0              'VA'             G0:=G3
1383  1539. 0              'VA'                R6:=     G4
1384  1540. 0   V    1576 'VA'                G0       ¢ R6  :SP ENQ2
1385                          ***           REST.R1:='KO':                          KEINE RES
1386  1543. 0              'VA'             G0:=0
1387  1544. 0              'VA'     (07  !WTL+G1)     :=G0
1388                          ***     LOC:ENQ3:
1389  1546. 0              'VA' ENQ3    /
1390                          ***           R3:=R3*APOINTER:
1391  1546. 0              'VA'             G0:=G3
1392  1547. 0              'VA'     G3                    :=G0
1393                          ***           R3:=R3+TPOINT:                          AKT. ZEIT
1394  1548. 0   V     877 'VA'             G0:=G3                    +(TPOINT   )
1395  1551. 0              'VA'     G3                    :=G0
1396                          ***           R4:=TLANG*APOINTER:                     LISTENLAE
1397  1552. 0   V     755 'VA'             G0:=(TLANG    )
1398  1554. 0              'VA'     G4                    :=G0
1399                          ***     LOC: ENQ0:
1400  1555. 0              'VA' ENQ0    /
1401                          ***           .TC. ENQ1 .IF. ATIMEND .GT. R3; KEINLEE
1402  1555. 0   V     876 'VA'             G0:=¢TIMEND     ¢
1403  1557. 0              'VA'                R6:=     G3
1404  1558. 0   V    1566 'VA'                G0       ¢ R6  :SP ENQ1
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:                              BLA
QUELLDATEI: CODE          ABSCHNITT: OBJE     SATZ:   OBJS


1405                        ***          R3:= R3 - R4;
1406   1561. 0              'VA'         GO:=G3                          -G4
1407   1563. 0              'VA'    G3                        :=GO
1408                        ***          .TC. ENQ0:
1409   1564. 0  V   1555    'VA'                                :SP ENQ0
1410                        ***    LOC: ENQ1:
1411   1566. 0              'VA' ENQ1    /
1412                        ***       R4:=A0.R3:
1413   1566. 0              'VA'         GO:=(G3)
1414   1567. 0              'VA'    G4                        :=GO
1415                        ***       A0.R3:=R1:
1416   1568. 0              'VA'         GO:=G1
1417   1569. 0              'VA'    (G3)                      :=GO
1418                        ***       SKETT.R1:=R4:
1419   1570. 0              'VA'         GO:=G4
1420   1571. 0              'VA'    (00   WTL+G1)             :=GO
1421                        ***          .RETURN. ENQ:
1422   1573. 0  V   1532    'VA'            R7:= (QENQ       )
1423   1575. 0              'VA'             ;SP  R7
1424                        ***    LOC:ENQ2:
1425   1576. 0              'VA' ENQ2    /
1426                        ***          REST.R1:=R3-R4:                  REST7EIT:=
1427   1576. 0              'VA'         GO:=G3                      -G4
1428   1578. 0              'VA'    (07   WTL+G1)             :=GO
1429                        ***          R3:=R4:                          LISTENLAEN
1430   1580. 0              'VA'         GO:=G4
1431   1581. 0              'VA'    G3                        :=GO
1432                        ***          .TC. ENQ3:
1433   1582. 0  V   1546    'VA'                                :SP ENQ3
1434                        *** END: ENQ:
1435                        ***/:
1436                        ***/;
1437                        ***/:
1438                        ***/                                AUSKETTEN AUS SCHEDUL
1439                        ***/                                ====================:
1440                        ***/:
1441                        ***/                                VERSORGUNG: R1...7EIT(
1442                        ***/:
1443                        ***    PROC:DEQU:
1444   1584. 0              'VA' QDEQU   /=1
1445   1585. 0  V   1584    'VA'  DEQU   /(QDEQU   ):=R7
1446                        ***          .TC. DEQ1 .IFNOT. R1 .NIL.:
1447   1587. 0              'VA'         GO:=G1
1448   1588. 0              'VA'            R6:=    0
1449   1589. 0  V   1595    'VA'            GO      -# R6  :SP DEQ1
1450                        ***          .RETURN. DEQU:                   KEIN ELEMEN
1451   1592. 0  V   1584    'VA'            R7:= (QDEQU      )
1452   1594. 0              'VA'             ;SP  R7
1453                        ***    LOC:DEQ1:
1454   1595. 0              'VA' DEQ1    /
1455                        ***          R3:=R1:
1456   1595. 0              'VA'         GO:=G1
1457   1596. 0              'VA'    G3                        :=GO
1458                        ***          .TC. DEQ2 .IFNOT. R4:=SKETT.R1 .NIL.:
1459   1597. 0              'VA'         GO:=(00   WTL+G1)
1460   1599. 0              'VA'    G4                        :=GO
1461   1600. 0              'VA'            R6:=    0
1462   1601. 0  V   1607    'VA'            GO      -# R6  :SP DEQ2
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSR-V3  33  DATUM:                                  BL
QUELLDATEI: CODE          ABSCHNITT: OBJE     SATZ:   OBJS

1463                          ***          .RETURN. DEQU:                          ELEMENT I
1464  1604. 0  V   1584 'VA'                  R7:= (QDEQU      )
1465  1606. 0            'VA'                  ;SP  R7
1466                          ***     LOC:DEQ2:
1467  1607. 0            'VA' DEQ2     /
1468                          ***          R3:=SKETT.R3:
1469  1607. 0            'VA'            G0:=(00   WTL+G3)
1470  1609. 0            'VA'       G3                  :=G0
1471                          ***          .TC. DEQ2 .IF. SKETT.R3 .NE. R1:
1472  1610. 0            'VA'            G0:=(00   WTL+G3)
1473  1612. 0            'VA'               R6:=     G1
1474  1613. 0  V   1607 'VA'               G0       #,R6   :SP DEQ2
1475                          ***          SKETT.R3:=R4:
1476  1616. 0            'VA'            G0:=G4
1477  1617. 0            'VA'       (00   WTL+G3)     :=G0
1478                          ***          SKETT.R1:=ANIL:
1479  1619. 0            'VA'            G0:=0
1480  1620. 0            'VA'       (00   WTL+G1)     :=G0
1481                          ***          .RETURN. DEQU:
1482  1622. 0  V   1584 'VA'               R7:= (QDEQU      )
1483  1624. 0            'VA'               ;SP  R7
1484                          ***     END:DEQU:
1485                          ***/;
1486                          ***/:
1487                          ***/                                  ACTIVATE;
1488                          ***/                                  ========:
1489                          ***/:
1490                          ***/                                  VERSORGUNG:    R1...
1491                          ***/                                                 R2...
1492                          ***/                                                 R4...
1493                          ***/:
1494                          *** PROC:START:
1495  1625. 0            'VA' QSTART   /=1⌀
1496  1626. 0  V   1625 'VA'  START   /(QSTART   ):=R7
1497                          ***          .TC. STA1 .IF. R4 .NULL.:      KEIN SCHED
1498  1628. 0            'VA'            G0:=G4
1499  1629. 0            'VA'               R6:=     0
1500  1630. 0  V   1642 'VA'               G0       := R6  :SP STA1
1501                          ***          SCHART.R1:=R2+ASCHPNT:
1502  1633. 0            'VA'            G0:=G2                    +04
1503  1635. 0            'VA'       (06   WTL+G1)     :=G0
1504                          ***          .CALL. SCH:                            SCHEDULE-V
1505  1637. 0  V   1410 'VA'               R7  :US     SCH
1506                          ***          .RETURN. START:
1507  1639. 0  V   1625 'VA'               R7:= (QSTART     )
1508  1641. 0            'VA'               ;SP  R7
1509                          ***     LOC:STA1:
1510  1642. 0            'VA' STA1     /
1511                          ***          .TC. LOCK .IFNOT. STATUS.R2 .NULL.:
1512  1642. 0            'VA'            G0:=(03   WTL+G2)
1513  1644. 0            'VA'               R6:=     0
1514  1645. 0  V   1660 'VA'               G0       # R6  :SP LOCK
1515                          ***          STATUS.R2:='B1000':
1516  1648. 0            'VA'            G0:=008
1517  1649. 0            'VA'       (03   WTL+G2)     :=G0
1518                          ***          PC.R2:=STPC.R2:                        PC INITIAL
1519  1651. 0            'VA'            G0:=(09   WTL+G2)
1520  1653. 0            'VA'       (08   WTL+G2)     :=G0
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:                              RIA
QUELLDATEI: CODE          ABSCHNITT: OBJE     SATZ:    OBJS

1521                        ***          R1:=ARUNNING;          IN RUNNING-KETTE EIN
1522  1655. 0  V    878 'VA'             G0:=(RUNNING    )
1523  1657. 0             'VA'     G1                    ,:=G0
1524                        ***          .CALL. CHAIN;
1525  1658. 0  V   1853 'VA'              R7  :US     CHAIN
1526                        ***     LOC:LOCK;
1527  1660. 0             'VA' LOCK     /
1528                        ***          .RETURN. START;
1529  1660. 0  V   1625 'VA'              R7:= (GSTART     )
1530  1662. 0             'VA'              :SP  R7
1531                        *** END:START;
1532                        ***/;
1533                        ***/;
1534                        ***/                              REQUEST;
1535                        ***/                              =======;
1536                        ***/;
1537                        ***/                              VERSORGUNG: R2...ZEI(
1538                        ***/;
1539                        *** PROC:REQU;
1540  1663. 0             'VA' QREQU    /=1(2)
1541  1664. 0  V   1663 'VA'  REQU     /(QREQU    ):=R7
1542                        ***          R1:=R2;                          SEMA-ADRESS
1543  1666. 0             'VA'         G0:=G2
1544  1667. 0             'VA'     G1                    ,:=G0
1545                        ***          R2:=RUNNING;
1546  1668. 0  V    878 'VA'          G0:=(RUNNING )
1547  1670. 0             'VA'     G2                    :=G0
1548                        ***          .TC. REQU1 .IF. PRIO.R1 .NULL.;
1549  1671. 0             'VA'         G0:=(02   WTL+G1)
1550  1673. 0             'VA'           R6:=    0
1551  1674. 0  V   1685 'VA'           G0      = R6  :SP REQU1
1552                        ***          PRIO.R1:=PRIO.R1-'K1';
1553  1677. 0             'VA'         G0:=(02   WTL+G1)       -1
1554  1680. 0             'VA'     (02   WTL+G1)     ,:=G0
1555                        ***          .RETURN. REQU;
1556  1682. 0  V   1663 'VA'           R7:= (QREQU      )
1557  1684. 0             'VA'           :SP  R7
1558                        ***     LOC:REQU1;
1559  1685. 0             'VA' REQU1    /
1560                        ***          STATUS.R2:='80001';              TASK UNTERB
1561  1685. 0             'VA'         G0:=001
1562  1686. 0             'VA'     (03   WTL+G2)     ,:=G0
1563                        ***          .CALL. UNCH;                     AUSKETTEN A
1564  1688. 0  V   1897 'VA'           R7  :US     UNCH
1565                        ***          .CALL. CHAIN;                    EINKETTEN I
1566  1690. 0  V   1853 'VA'           R7  :US     CHAIN
1567                        ***          .RETURN. REQU;
1568  1692. 0  V   1663 'VA'           R7:= (QREQU      )
1569  1694. 0             'VA'           :SP  R7
1570                        *** END:REQU;
1571                        ***/;
1572                        ***/;
1573                        ***/                              RELEASE;
1574                        ***/                              =======;
1575                        ***/;
1576                        ***/                              VERSORGUNG: R2...ZEI(
1577                        ***/;
1578                        *** PROC:RELE;
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:                                  BL
QUELLDATEI: CODE          ABSCHNITT: OBJE     SATZ:    OBJS

1579  1695, 0             'VA' QRELE    /=1¢¥
1580  1696. 0  V   1695    'VA'  RELE    /(QRELE    ):=R7
1581                       ***           R1:=R2:                          SEMA-ADRE
1582  1698. 0              'VA'          G0:=G2
1583  1699. 0              'VA'     G1                    :=G0
1584                       ***           .TC. REL2 .IF. R2:=A0.R1 .EQ. R1:  KETT
1585  1700. 0              'VA'          G0:=(G1)
1586  1701. 0              'VA'     G2                    :=G0
1587  1702. 0              'VA'          R6:=     G1
1588  1703. 0  V   1719    'VA'          G0       =,R6   :SP REL2
1589                       ***           R1:=ARUNNING:
1590  1706. 0  V    878    'VA'          G0:=¢RUNNING   ¥
1591  1708. 0              'VA'     G1                    :=G0
1592                       ***           .CALL. UNCH:                          AUS SEMA-
1593  1709. 0  V   1897    'VA'          R7   :US    UNCH
1594                       ***           .CALL. CHAIN:                         IN RUNNIN
1595  1711. 0  V   1853    'VA'          R7   :US    CHAIN
1596                       ***           STATUS.R2:='B1000':
1597  1713. 0              'VA'          G0:=008
1598  1714. 0              'VA'     (03   WTL+G2)     ;=G0
1599                       ***           .RETURN. RELE:
1600  1716. 0  V   1695    'VA'          R7:= (QRELE      )
1601  1718. 0              'VA'          :SP  R7
1602                       ***    LOC:REL2:
1603  1719. 0              'VA' REL2     /
1604                       ***           PRIO.R1:=PRIO.R1+'K1':
1605  1719. 0              'VA'          G0:=(02   'WTL+G1)        +1
1606  1722. 0              'VA'     (02   WTL+G1)     ;=G0
1607                       ***           .RETURN. RELE:
1608  1724. 0  V   1695    'VA'          R7:= (QRELE      )
1609  1726. 0              'VA'          :SP  R7
1610                       *** END:RELE:
1611                       ***/;
1612                       ***/:
1613                       ***/                                TERMINATE;
1614                       ***/                                =========:
1615                       ***/:
1616                       ***/                                VERSORGUNG: R2...7F
1617                       ***/:
1618                       *** PROC:STOP:
1619  1727, 0              'VA' QSTOP    /=1¢¥
1620  1728. 0  V   1727    'VA'  STOP    /(QSTOP    ):=R7
1621                       ***           R2:=RUNNING:
1622  1730. 0  V    878    'VA'          G0:=(RUNNING )
1623  1732. 0              'VA'     G2                    :=G0
1624                       ***           .CALL. UNCH:                          AUS PWS
1625  1733. 0  V   1897    'VA'          R7   :US    UNCH
1626                       ***           STATUS.R2:='K0':
1627  1735, 0              'VA'          G0:=0
1628  1736. 0              'VA'     (03   WTL+G2)     ;=G0
1629                       ***           .RETURN. STOP:
1630  1738. 0  V   1727    'VA'          R7:= (QSTOP      )
1631  1740. 0              'VA'          :SP  R7
1632                       *** END:STOP:
1633                       ***/;
1634                       ***/:
1635                       ***/                                PREVENT;
1636                       ***/                                =======:
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:                              RIAT
QUELLDATEI: CODE          ABSCHNITT: OBJE    SATZ:   OBJS


1637                        ***/:
1638                        ***/                                  VERSORGUNG: R2...ZEIG
1639                        ***/:
1640                        *** PROC:PREV;
1641   1741. 0              'VA' QPREV   /=1⌀;
1642   1742. 0  V   1741    'VA'  PREV   /(QPREV   ):=R7
1643                        ***          .TC. PREV1 .IFNOT. R2 .NIL.;
1644   1744. 0              'VA'         G0:=G2
1645   1745. 0              'VA'            R6:=    0
1646   1746. 0  V   1752    'VA'            G0      ;# R6  ;SP PREV1
1647                        ***          R2:=RUNNING;
1648   1749. 0  V    878    'VA'         G0:=(RUNNING )
1649   1751. 0              'VA'    G2                  ;=G0
1650                        ***   LOC:PREV1;
1651   1752. 0              'VA' PREV1   /
1652                        ***          R1:=SCHPNT.R2;                    ACT-SCHEDU
1653   1752. 0              'VA'         G0:=(04   WTL+G2)
1654   1754. 0              'VA'    G1                  ;=G0
1655                        ***          .CALL. DEQL;
1656   1755. 0  V   1585    'VA'            R7  ;US    DEQU
1657                        ***          SCHPNT.R2:=ANIL;
1658   1757. 0              'VA'         G0:=0
1659   1758. 0              'VA'    (04   WTL+G2)    ;=G0
1660                        ***          R1:=SCHRES.R2;                    RESUME-SCHE
1661   1760. 0              'VA'         G0:=(05   WTL+G2)
1662   1762. 0              'VA'    G1                  ;=G0
1663                        ***          .CALL. DEQL;
1664   1763. 0  V   1585    'VA'            R7  ;US    DEQU
1665                        ***          SCHRES.R2:=ANIL;
1666   1765. 0              'VA'         G0:=0
1667   1766. 0              'VA'    (05   WTL+G2)    ;=G0
1668                        ***          .RETURN. PREV;
1669   1768. 0  V   1741    'VA'            R7:= (QPREV     )
1670   1770. 0              'VA'            ;SP  R7
1671                        *** END:PREV;
1672                        ***/:
1673                        ***/:
1674                        ***/                                  SUSPEND;
1675                        ***/                                  =======;
1676                        ***/:
1677                        ***/                                  VERSORGUNG: KEINE;
1678                        ***/:
1679                        *** PROC:SUSP;
1680   1771. 0              'VA' QSUSP   /=1⌀;
1681   1772. 0  V   1771    'VA'  SUSP   /(QSUSP   ):=R7
1682                        ***          R2:=RUNNING;
1683   1774. 0  V    878    'VA'         G0:=(RUNNING )
1684   1776. 0              'VA'    G2                  ;=G0
1685                        ***          STATUS.R2:='B0100';
1686   1777. 0              'VA'         G0:=004
1687   1778. 0              'VA'    (03   WTL+G2)    ;=G0
1688                        ***          .CALL. UNCH;
1689   1780. 0  V   1897    'VA'            R7  ;US    UNCH
1690                        ***          .RETURN. SUSP;
1691   1782. 0  V   1771    'VA'            R7:= (QSUSP     )
1692   1784. 0              'VA'            ;SP  R7
1693                        *** END:SUSP;
1694                        ***/:
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:                                  BL
QUELLDATEI: CODE          ABSCHNITT: OBJE     SATZ:   OBJS

1695                        ***/:
1696                        ***/                                  CONTINUE;
1697                        ***/,                                 ========:
1698                        ***/:
1699                        ***/                                  VERSORGUNG:     R1...
1700                        ***/                                                  R2...
1701                        ***/                                                  R4...
1702                        ***/:
1703                        *** PROC:CONT;
1704   1785. 0              'VA' QCONT    /=1≠≠
1705   1786. 0  V   1785    'VA'  CONT    /(QCONT    ):=R7
1706                        ***           .TC. CONTO .IFNOT. R2 .NIL.:
1707   1788. 0              'VA'          GO:=G2
1708   1789. 0              'VA'             R6:=     0
1709   1790. 0  V   1796    'VA'             GO       # R6  ;SP CONTO
1710                        ***           R2:=RUNNING;
1711   1793. 0  V    878    'VA'          GO:=(RUNNING )
1712   1795. 0              'VA'     G2                    :=GO
1713                        ***     LOC:CONT0:
1714   1796. 0              'VA' CONTO    /
1715                        ***           .TC. CONT1 .IF. R4 .NULL.:     KEIN SCHEM
1716   1796. 0              'VA'          GO:=G4
1717   1797. 0              'VA'             R6:=     0
1718   1798. 0  V   1810    'VA'             GO       = R6  ;SP CONT1
1719                        ***           SCHART.R1:=R2+ASCHRES:
1720   1801. 0              'VA'          GO:=G2                      +05
1721   1803. 0              'VA'     (06   WTL+G1)     :=GO
1722                        ***           .CALL. SCH;
1723   1805. 0  V   1410    'VA'             R7  ;US    ,SCH
1724                        ***           .RETURN. QCNT;
1725   1807. 0  V   1785    'VA'             R7:= (QCONT      )
1726   1809. 0              'VA'             ;SP  R7
1727                        ***     LOC:CONT1:
1728   1810. 0              'VA' CONT1    /
1729                        ***           .TC. CONT2 .IF. STATUS.R2 .NE. 'B0100';
1730   1810. 0              'VA'          GO:=(G3   WTL+G2)
1731   1812. 0              'VA'             R6:=     004
1732   1813. 0  V   1824    'VA'             GO       # R6  ;SP CONT2
1733                        ***           STATUS.R2:='B1000';
1734   1816. 0              'VA'          GO:=008
1735   1817. 0              'VA'     (03   WTL+G2)     :=GO
1736                        ***           R1:=ARUNNING;
1737   1819. 0  V    878    'VA'          GO:=≠RUNNING   ≠
1738   1821. 0              'VA'     G1                    :=GO
1739                        ***           .CALL. CHAIN;
1740   1822. 0  V   1853    'VA'             R7  ;US    CHAIN
1741                        ***     LOC:CONT2:
1742   1824. 0              'VA' CONT2    /
1743                        ***           .RETURN. QCNT;
1744   1824. 0  V   1785    'VA'             R7:= (QCONT      )
1745   1826. 0              'VA'             ;SP  R7
1746                        *** END:CONT;
1747                        ***/;
1748                        ***/:
1749                        ***/                                  RESUME;
1750                        ***/,                                 ======:
1751                        ***/:
1752                        ***/                                  VERSORGUNG:     R1...
```

```
1753                    ***/                                    R2...0;
1754                    ***/                                    R4...SCHEDULE-TYP;
1755                    ***/:
1756                    *** PROC:RESU;
1757  1827. 0           'VA' QRESU   /=1≠≠
1758  1828. 0  V  1827  'VA'  RESU   /(QRESU   ):=R7
1759                    ***          .TO. RESU1 .IF. R4 .NULL.:
1760  1830. 0           'VA'         GO:=G4
1761  1831. 0           'VA'            R6:=     0
1762  1832. 0  V  1847  'VA'            GO       = R6  :SP RESU1
1763                    ***          SPCB.R1:=R2:=RUNNING;
1764  1835. 0  V   878  'VA'         GO:=(RUNNING )
1765  1837. 0           'VA'    G2                  :=GO
1766  1838. 0           'VA'    (01   WTL+G1)       :=GO
1767                    ***          .CALL. CONT;
1768  1840. 0  V  1786  'VA'            R7  :LS   CONT
1769                    ***          .CALL. SUSP;
1770  1842. 0  V  1772  'VA'            R7  :LS   SUSP
1771                    ***          .RETURN. RESU;
1772  1844. 0  V  1827  'VA'            R7:= (QRESU     )
1773  1846. 0           'VA'            :SP  R7
1774                    ***    LOC:RESU1:
1775  1847. 0           'VA' RESU1   /
1776                    ***          .CALL. CONT;
1777  1847. 0  V  1786  'VA'            R7  :US   CONT
1778                    ***          .RETURN. RESU;
1779  1849. 0  V  1827  'VA'            R7:= (QRESU     )
1780  1851. 0           'VA'            :SP  R7
1781                    *** END:RESU;
1782                    ***/;
1783                    ***/:
1784                    ***/                                    EINKETTEN;
1785                    ***/                                    =========:
1786                    ***/:
1787                    ***/                                    VERSORGUNG:    R1...?
1788                    ***/                                                   R2...?
1789                    ***/:
1790                    ***    PROC:CHAIN;
1791  1852. 0           'VA' QCHAIN  /=1≠≠
1792  1853. 0  V  1852  'VA'  CHAIN  /(QCHAIN  ):=R7
1793                    ***          R5:=R1;
1794  1855. 0           'VA'         GO:=G1
1795  1856. 0           'VA'    G5                  :=GO
1796                    ***          R3:=SUCC.R1;                    NACHFOLGER
1797  1857. 0           'VA'         GO:=(00   WTL+G1)
1798  1859. 0           'VA'    G3                  :=GO
1799                    ***          R6:=PRIO.R2;                    PRIORITAET
1800  1860. 0           'VA'         GO:=(02   WTL+G2)
1801  1862. 0           'VA'    G6                  :=GO
1802                    ***    LOC:LOOP;
1803  1863. 0           'VA' LOOP    /
1804                    ***          .TC. IN .IF. R3 .EQ. R5;        KETTENENDE
1805  1863. 0           'VA'         GO:=G3
1806  1864. 0           'VA'            R6:=     G5
1807  1865. 0  V  1882  'VA'            GO       = R6  :SP IN
1808                    ***          .TC. IN .IF. R6 .LT. PRIO.R3;   NOCHMAL P
1809  1868. 0           'VA'         GO:=G6
1810  1869. 0           'VA'            R6:=     (02   WTL+G3)
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:                          RI
QUELLDATEI: CODE          ABSCHNITT: OBJE    SATZ:   OBJS

1811   1871. 0  V   1882 'VA'            GO      ,#=R6   ;SP IN
1812                    ***           R1:=SUCC.R1;
1813   1874. 0          'VA'           GO:=(00   WTL+G1)
1814   1876. 0          'VA'     G1                    :=GO
1815                    ***           R3:=SUCC.R1;
1816   1877. 0          'VA'           GO:=(00   WTL+G1)
1817   1879. 0          'VA'     G3                    :=GO
1818                    ***           .TC. LOOP;
1819   1880. 0  V   1863 'VA'                                ;SP LOOP
1820                    ***     LOC:IN;
1821   1882. 0          'VA' IN       /
1822                    ***           SUCC.R1:=PRED.R3;=R2;
1823   1882. 0          'VA'           GO:=G2
1824   1883. 0          'VA'     (01   WTL+G3)         :=GO
1825   1885. 0          'VA'     (00   WTL+G1)         :=GO
1826                    ***           PRED.R2:=R1;
1827   1887. 0          'VA'           GO:=G1
1828   1888. 0          'VA'     (01   WTL+G2)         :=GO
1829                    ***           SUCC.R2:=R3;
1830   1890. 0          'VA'           GO:=G3
1831   1891. 0          'VA'     (00   WTL+G2)         :=GO
1832                    ***           .RETURN. CHAIN;
1833   1893. 0  V   1852 'VA'              R7:= (QCHAIN     )
1834   1895. 0          'VA'              ;SP  R7
1835                    ***     END:CHAIN:
1836                    ***/;
1837                    ***/:
1838                    ***/                                  AUSKETTEN;
1839                    ***/                                  =========:
1840                    ***/:
1841                    ***/                                  VERSORGUNG: R2..ZEIG
1842                    ***/:
1843                    ***     PROC:UNCH:
1844   1896. 0          'VA' QUNCH    /=1#;
1845   1897. 0  V   1896 'VA'  UNCH    /(QUNCH    );=R7
1846                    ***           R4:=R3:=PRED.R2;
1847   1899. 0          'VA'           GO:=(01   WTL+G2)
1848   1901. 0          'VA'     G3                    :=GO
1849   1902. 0          'VA'     G4                    :=GO,
1850                    ***           R3:=SUCC.R3:=SUCC.R2;
1851   1903. 0          'VA'           GO:=(00   WTL+G2)
1852   1905. 0          'VA'     (00   WTL+G3)         :=GO
1853   1907. 0          'VA'     G3                    :=GO
1854                    ***           PRED.R3:=R4;
1855   1908. 0          'VA'           GO:=G4
1856   1909. 0          'VA'     (01   WTL+G3)         ;=GO
1857                    ***           .RETURN. UNCH;
1858   1911. 0  V   1896 'VA'              R7:= (QUNCH      )
1859   1913. 0          'VA'              ;SP  R7
1860                    ***     END:UNCH;
1861                    ***/:
1862                    *** PROC: EXOPE;
1863   1914. 0          'VA' QEXOPE   /=1#;
1864   1915. 0  V   1914 'VA'  EXOPE   /(QEXOPE   ):=R7
1865                    ***           .TC. XSTAR .IF. R3 .EQ. 'K1';
1866   1917. 0          'VA'           GO:=G3
1867   1918. 0          'VA'              R6:=     1
1868   1919. 0  V   1964 'VA'              GO       = R6   ;SP XSTAR
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSR-V3  33  DATUM:                          BLA
QUELLCATEI: CODE          ABSCHNITT: OBJE     SATZ:    OBJS


1869                          ***          .TC. XREQU .IF. R3 .EQ. 'K2';
1870  1922. 0                 'VA'         GO:=G3
1871  1923. 0                 'VA'            R6:=     2
1872  1924. 0  V    1968       'VA'            GO       := R6  :SP XREQU
1873                          ***          .TC. XRELE .IF. R3 .EQ. 'K3';
1874  1927. 0                 'VA'         GO:=G3
1875  1928. 0                 'VA'            R6:=     3
1876  1929. 0  V    1972       'VA'            GO       := R6  :SP XRELE
1877                          ***          .TC. XSTOP .IF. R3 .EQ. 'K4';
1878  1932. 0                 'VA'         GO:=G3
1879  1933. 0                 'VA'            R6:=     4
1880  1934. 0  V    1976       'VA'            GO       := R6  :SP XSTOP
1881                          ***          .TC. XPREV .IF. R3 .EQ. 'K5';
1882  1937. 0                 'VA'         GO:=G3
1883  1938. 0                 'VA'            R6:=     5
1884  1939. 0  V    1980       'VA'            GO       := R6  :SP XPREV
1885                          ***          .TC. XSUSP .IF. R3 .EQ. 'K6';
1886  1942. 0                 'VA'         GO:=G3
1887  1943. 0                 'VA'            R6:=     6
1888  1944. 0  V    1984       'VA'            GO       := R6  :SP XSUSP
1889                          ***          .TC. XCONT .IF. R3 .EQ. 'K7';
1890  1947. 0                 'VA'         GO:=G3
1891  1948. 0                 'VA'            R6:=     7
1892  1949. 0  V    1988       'VA'            GO       := R6  :SP XCONT
1893                          ***          .TC. XRESU .IF. R3 .EQ. 'K8';
1894  1952. 0                 'VA'         GO:=G3
1895  1953. 0                 'VA'            R6:=     8
1896  1954. 0  V    1992       'VA'            GO       := R6  :SP XRESU
1897                          ***          .TC. XDRIV .IF. R3 .EQ. 'K9';
1898  1957. 0                 'VA'         GO:=G3
1899  1958. 0                 'VA'            R6:=     9
1900  1959. 0  V    1996       'VA'            GO       := R6  :SP XDRIV
1901                          ***          .TC. XFEHL :
1902  1962. 0  V    2000       'VA'                                :SP XFEHL
1903                          ***          LOC:XSTAR:
1904  1964. 0                 'VA' XSTAR    /
1905                          ***          .CALL. START:
1906  1964. 0  V    1626       'VA'            R7  :US     START
1907                          ***          .TC. XEND:
1908  1966. 0  V    2000       'VA'                                :SP XEND
1909                          ***          LOC:XREQU:
1910  1968. 0                 'VA' XREQU    /
1911                          ***          .CALL. REQU:
1912  1968. 0  V    1664       'VA'            R7  :US     REQU
1913                          ***          .TC. XEND:
1914  1970. 0  V    2000       'VA'                                :SP XEND
1915                          ***          LOC: XRELE:
1916  1972. 0                 'VA' XRELE    /
1917                          ***          .CALL. RELE:
1918  1972. 0  V    1696       'VA'            R7  :US     RELE
1919                          ***          .TC. XEND:
1920  1974. 0  V    2000       'VA'                                :SP XEND
1921                          ***          LOC: XSTOP:
1922  1976. 0                 'VA' XSTOP    /
1923                          ***          .CALL. STOP:
1924  1976. 0  V    1728       'VA'            R7  :US     STOP
1925                          ***          .TC. XEND:
1926  1978. 0  V    2000       'VA'                                :SP XEND
```

```
SIEMENS ASS.SPRACHE 320 /330 :  ASSB-V3  33  DATUM:                              RI
QUELLDATEI: CODE          ABSCHNITT: OBJE    SATZ:   OBJS


1927                          ***           LOC: XPREV;
1928  1980. 0                 'VA' XPREV     /
1929                          ***           .CALL. PREV;
1930  1980. 0  V    1742 'VA'                  R7  :US     PREV
1931                          ***           .TC. XEND;
1932  1982. 0  V    2000 'VA'                                  :SP XEND
1933                          ***           LOC: XSUSP;
1934  1984. 0                 'VA' XSUSP     /
1935                          ***           .CALL. SUSP;
1936  1984. 0  V    1772 'VA'                  R7  :US     SUSP
1937                          ***           .TC. XEND;
1938  1986. 0  V    2000 'VA'                                  :SP XEND
1939                          ***           LOC: XCONT;
1940  1988. 0                 'VA' XCONT     /
1941                          ***           .CALL. CONT;
1942  1988. 0  V    1786 'VA'                  R7  :US     CONT
1943                          ***           .TC. XEND;
1944  1990. 0  V    2000 'VA'                                  :SP XEND
1945                          ***           LOC:XRESU;
1946  1992. 0                 'VA' XRESU     /
1947                          ***           .CALL. RESU;
1948  1992. 0  V    1828 'VA'                  R7  :US     RESU
1949                          ***           .TC. XEND;
1950  1994. 0  V    2000 'VA'                                  :SP XEND
1951                          ***           LOC:XDRIV;
1952  1996. 0                 'VA' XDRIV     /
1953                          ***           .CALL. DRIV;
1954  1996. 0  V    1137 'VA'                  R7  :US     DRIV
1955                          ***           .TC. XEND;
1956  1998. 0  V    2000 'VA'                                  :SP XEND
1957                          ***           LOC: XFEHL; FEHLER AUSGANG
1958  2000. 0                 'VA' XFEHL     /
1959                          ***           LOC:XEND;
1960  2000. 0                 'VA' XEND      /
1961                          ***           .RETURN. EXOPE;
1962  2000. 0  V    1914 'VA'                  R7:= (QEXOPE     )
1963  2002. 0                 'VA'                 :SP  R7
1964                          *** END: EXOPE;
1965                          ***/;
1966                          *** FINISH;
1967                                   'ENDE'
```

Anhang VI Testprogramm

```
1                                       *** ZWEITES TESTPROGRAMM FUER PBS-310
2
3     'NAME'          SEMA
4     'SATZ'          SEMA
5     'PRIV'
6
7     'VA'
8           KOOZ/     'IDENT'         5000
9           BRIEF/    'IDENT'         5003
10
11          START/    'IDENT'         1
12          REQU/     'IDENT'         2
13          RELE/     'IDENT'         3
14          STOP/     'IDENT'         4
15          PREV/     'IDENT'         5
16          SUSP/     'IDENT'         6
17          CONT/     'IDENT'         7
18          RESU/     'IDENT'         8
19          DRIV/     'IDENT'         9
20
21          DRUCK/    'IDENT'         0
22          BSAUS/    'IDENT'         1
23          BSEIN/    'IDENT'         2
24
25          ***INITIALISIERUNG ANFANG
26          ***
27    'VA'
28          INIT/     G1              := FRSTCB
29                    (BRIEF)         := G1
30                    G1              := FRSTSM
31                    (BRIEF+1)       := G1
32                    G1              := FRSTIO
33                    (BRIEF+2)       := G1
34                    G1              := LASTIO
35                    (BRIEF+3)       := G1
36                    $KOORM%KOOZM+;
37
38    'VA'            G1              := DRUCB
39                    (BRIEF)         := G1
40                    G1              := BSACB
41                    (BRIEF+1)       := G1
42                    G1              := BSECB
43                    (BRIEF+2)       := G1
44                    $KOORM%KOOZM+;
45          ***
46          *** INITIALISIERUNG ENDE
47
```

```
48      *** T A S K 1
49      ************
50
51           TASK1/
52      'VA'
53                      *** ACTIVATE TASK2
54                      GØ              := AC2PB
55                      (BRIEF)         := GØ
56                      $KOORMKOOZM+;
57      'VA'
58                      *** ACTIVATE TASK3
59                      GØ              := AC3PB
60                      (BRIEF)         := GØ
61                      $KOORMKOOZM+;
62      'VA'
63                      *** TERMINATE
64                      GØ              := TER1PB
65                      (BRIEF)         := GØ
66                      $KOORMKOOZM+;
67
68      'VA'            $ENDE;
69
70      *** T A S K 2
71      ************

72
73           TASK2/
74      'VA'
75                      *** REQUEST SEME
76                      GØ              := RQSEPB
77                      (BRIEF)         := GØ
78                      $KOORMKOOZM+;
79
80      'VA'
81                      G5              := 1Ø
82                      (TAETIG)        := G5
83
84      'VA'
85                      *** GET BLATTSCHREIBER EDIT (TEXTA)
86                      GØ              := BEPB
87                      (BRIEF)         := GØ
88                      $KOORMKOOZM+;
89      'VA'
90                      *** RELEASE SEMA
91                      GØ              := RLSAPB
92                      (BRIEF)         := GØ
93                      $KOORMKOOZM+;
94      'VA'
95                      *** PUT DRUCKER EDIT (TEXTA)
96           T2DRUK/    GØ              := DRUPB
97                      (BRIEF)         := GØ
98                      $KOORMKOOZM+;
```

```
 99
100     *** ------------------------------------------------
101     *** JETZT WERDEN ALLE GROSS-BUCHSTABEN IM PUFFER
102     *** BIS ZUM 'ETX' IN KLEIN-BUCHSTABEN UMGEWANDELT
103     *** UND UMGEKEHRT
104
105     'VA'            G0              := 'H=0020'
106                     R6              := TEXTA
107           T2LOOP/   G0 = 'H=0020'   :SP T2NEXT
108                     G0              := 'H=0020'
109                     G4              := (R6) .U  'H=00FF'
110                     :SP             T2TEST
111           T2NEXT/   G0              := 'H=2000'
112                     R6              := R6 + 1
113                     G4              := (R6) .V -8
114           T2TEST/   G4 = 3          :SP WEITER
115                     R6 > TEXTE      :SP WEITER
116                     G4              := G4 .U 'H=00DF'
117                     G4 < 65         :SP T2LOOP
118                     G4 > 90         :SP T2LOOP
119                     G4              := (R6)
120                     G4              := G4 .X G0
121                     (R6)            := G4
122                     :SP             T2LOOP
123           WEITER/
124
125     *** ENDE DER UMWANDLUNG
126     ***-------------------------------------------------------
127
128     'VA'
129                     G5              := G5 - 1
130                     G5 > 0          :SP T2DRUK
131                     (TAETIG)        := R0
132
133     'VA'
134                     :SP             TASK2
135
136     'VA'            $ENDE;
137
138     *** T A S K 3
139     ***============
140
141           TASK3/
142     'VA'

143                     *** REQUEST SEMA
144                     G0              := RQSAPB
145                     (BRIEF)         := G0
146                     $KOORMKOOZM+;
147
148     'VA'
149                     *** PUT BLATTSCHREIBER EDIT (TEXTA)
150           T3BSAU/   G0              := BAPB
151                     (BRIEF)         := G0
152                     $KOORMKOOZN+;
153     'VA'
154                     G0              := (TAETIG)
155                     G0 > 0          :SP T3BSAU
156
157     'VA'
```

```
158                 *** RELEASE SEME
159                 GØ              := RLSEPB
160                 (BRIEF)         := GØ
161                 $KOORDKOOZM+;
162     'VA'
163                 :SP             TASK3
164
165     'VA'        $ENDE;
166
167     *** TEXTPUFFER
168     ***============
169     'VA'
170          TEXTA/                 =  TEXTE
171                                 =  39+Ø+
172          TEXTE/                 =  Ø
173
174          TAETIG/                =
175
176     ********************************************************************************
177
178     *** K O N T R O L L B L O E C K E
179     ********************************
180
181     *** TASKKONTROLLBLOECKE
182     'VA' FRSTCB/
183          TCB1/                  =  2+-+
184                                 =  Ø
185                                 =  6+-+
186                                 =  TASK1
187                                 =  16+-+
188          TCB2/                  =  2+-+
189                                 =  5
190                                 =  6+-+
191                                 =  TASK2
192                                 =  16+-+
193          TCB3/                  =  2+-+
194                                 =  5
195                                 =  6+-+
196                                 =  TASK3
197                                 =  16+-+
198
199     *** SEMAPHOR-KONTROLLBLOECKE
200          FRSTSM/
201          SEME/                  =  3+-+
202                                 =  1
203          SEMA/                  =  3+-+
204                                 =  Ø
205
206     ***GERAETEKONTROLLBLOECKE
207          FRSTIO/
208          DRUCB/                 =  3+-+
209                                 =  DRUCK
210                                 =
211          BSACB/                 =  3+-+
212                                 =  BSAUS
213                                 =

214          BSECB/                 =  3+-+
215                                 =  BSEIN
216                                 =
217          LASTIO/
218
```

```
219     *** AUFTRAGSPARAMETERBLOECKE
220     ***============================
221           ***
222           AC2PB/                    =
223                                     =   TCB2
224                                     =   START
225                                     =   Ø
226           AC3PB/                    =
227                                     =   TCB3
228                                     =   START
229                                     =   Ø
230           TER1PB/                   =
231                                     =   TCB1
232                                     =   STOP
233                                     =   Ø
234           RQSEPB/                   =
235                                     =   SEME
236                                     =   REQU
237                                     =   Ø
238           RQSAPB/                   =
239                                     =   SEMA
240                                     =   REQU
241                                     =   Ø
242           RLSEPB/                   =
243                                     =   SEME
244                                     =   RELE
245                                     =   Ø
246           RLSAPB/                   =
247                                     =   SEMA
248                                     =   RELE
249                                     =   Ø
250           BEPB/                     =
251                                     =   BSECB
252                                     =   DRIV
253                                     =   Ø
254                                     =   TEXTA
255           BAPB/                     =
256                                     =   BSACB
257                                     =   DRIV
258                                     =   Ø
259                                     =   TEXTA
260           DRUPB/                    =
261                                     =   DRUCB
262                                     =   DRIV
263                                     =   Ø
264                                     =   TEXTA
265
266     ***
267     ***============================
268     ***
269
270     'ENDE'
```