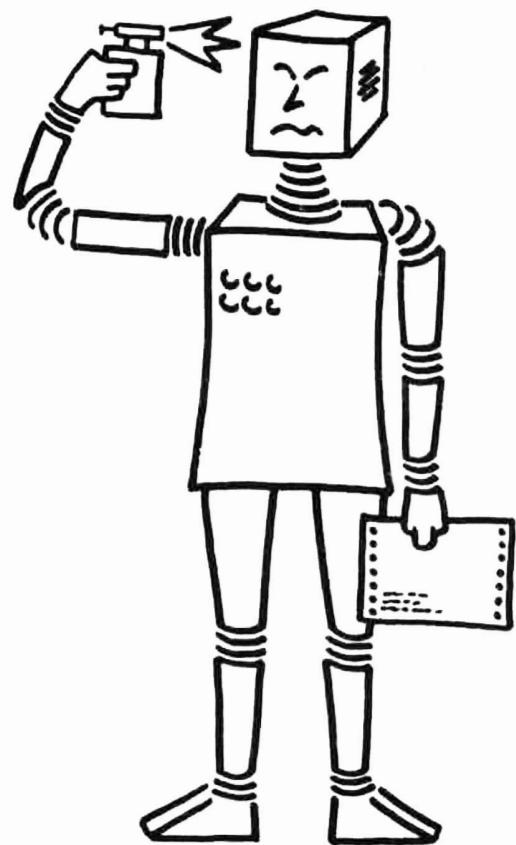


# SEKI - PROJEKT

Fachbereich Informatik  
Universität Kaiserslautern  
Postfach 3049  
D-6750 Kaiserslautern 1, W. Germany

**SEKI  
MEMO**



Entwurf und Implementierung von CSSA..

Teil E II: Programmdokumentation

MEMO-SEKI-82-03-E II

Christian Beilken,  
Friedemann Mattern,  
Michael Spenke



Entwurf und Implementierung von C S S A  
– Beschreibung der Sprache, des Compilers  
und des Mehrrechnersimulationssystems

Teil E :  
Programmdokumentation

Band II

Christian Beilken  
Friedemann Mattern  
Michael Spenke



**Design and Implementation of C S S A**  
- Description of the Language, the Compiler  
and the Multiprocessor Simulation System

**Volume E:**  
*Program-Documentation*

**Abstract:**

CSSA (*Computing System for Societies of Agents*) is an interactive programming language for asynchronous multiprocessor systems. Computations are done by concurrently working sequential modules, called agents which implement objects of data and control abstractions.

Communication is done by passing messages to acquainted agents. The receiving agent creates an instance of an operation capability referred to in the message. Since agents can be created during the computation and acquaintances can be transmitted in messages the heterarchical agent-net may dynamically change.

The language described in volume B of the CSSA-documentation has been successfully implemented for a multiprocessor simulation system running on a general purpose computer. This system allows the execution of CSSA-applications on a wide range of simulated multicomputer configurations. The use of the system is described in volume C.

This volume contains the documentation of the different system-components including program skeletons and complete source-listings. Portability problems are discussed in the description of the program development system which is realized in the Siemens BS2000 command language. The reader is expected to have a certain knowledge of this language.

Compiler and runtime system are written in SIMULA using to a large degree the more sophisticated language concepts of SIMULA, such as CLASS SIMULATION, hierarchical class definitions, virtual attributes of classes and prefix blocks. Therefore extensive knowledge of the SIMULA language is necessary for reading this volume.



## Vorwort

CSSA (*Computing System for Societies of Agents*) ist eine Programmiersprache für asynchron parallele Prozesse, die untereinander durch Übersenden von Nachrichten kommunizieren. Eine sequentielle Vorversion (CSSA-S) wurde 1979 implementiert, seit 1981 steht eine stark revidierte und erweiterte Fassung (CSSA-0) für Mehrprozessorsysteme zur Verfügung.

Die Entwicklung eines Übersetzers für CSSA wurde 1981 abgeschlossen, er ist in ein Multiprozessor-Simulationssystem integriert und erzeugt Code, der von diesem Simulationssystem ausgeführt wird. Erste Anwendungen und Erfahrungen mit dem auf der Siemens 7760 der GMD unter BS2000 implementierten System liegen bereits vor.

Dieser Teil der CSSA-Beschreibung enthält die Dokumentation aller Systemkomponenten. Neben Programmskeletten und überblicksartigen Beschreibungen ist auch der kommentierte Quelltext aller Komponenten abgedruckt.

Aus technischen Gründen mußte die Programmdokumentation in 2 Bände aufgeteilt werden. Der erste Band enthält die Dokumentation des Entwicklungssystems, das in der Siemens BS2000 Kommandosprache realisiert wurde. Kenntnisse dieser Sprache werden daher vorausgesetzt. Hier wird auch besonders auf Portabilitätsprobleme hingewiesen, die sich im wesentlichen auf eine Anpassung des Entwicklungssystems an die neue Installation beschränken. Eine Checkliste zur Portabilität soll diese Arbeit erleichtern.

Ebenfalls noch im ersten Band finden sich Erläuterungen zum CSSA-Compiler. Im Anhang ist der ausführlich kommentierte Quelltext abgedruckt.

Der zweite Band enthält Erklärungen zu Simulations- und Laufzeitsystem, zum interface-Agenten, zum interaktiven Kommandointerpreter, sowie zur Struktur des generierten codes. Für das Verständnis sind hier umfangreiche Kenntnisse des Sprache SIMULA nötig: Auch höhere Sprachkonzepte, wie die CLASS SIMULATION, Unter- und Oberklassen, virtuelle Attribute und Präfix-Blöcke werden extensiv verwendet.

Diese Programmdokumentation ist Teil der Diplomarbeit ("Entwurf und Implementierung von CSSA - Beschreibung der Sprache, des Compilers und des Mehrrechnersimulationssystems") von C. Beilken, F. Mattern und M. Spenke, andere Teile dieser Arbeit beschreiben genauer die CSSA-Konzepte und die CSSA-Sprache, ein weiterer den Umgang mit dem CSSA-System. CSSA-Beispielprogramme finden sich in Teil D.

BMS, im Juli 1982



Inhalt**Band I**

<b>1.</b>	<b>Das Entwicklungssystem</b>	<b>1</b>
<b>2.</b>	<b>Der Compiler</b>	<b>30</b>
2.1	Der Scanner	32
2.2	Die Fehlerbehandlungs routinen	34
2.3	Die Symboltabellenorganisation	36
2.4	Die statische Semantik	38
2.5	Der Parser	38
2.6	Die Codegenerierung	43
2.7	Das Hauptprogramm	47

**Anhang: Quelltext des Compilers****Band II**

<b>3.</b>	<b>Das Laufzeitsystem</b>	<b>1</b>
3.1	Die Klasse CSSA	3
3.2	Die Modul-Bibliothek	5
3.3	Der interface-Agent	8
3.4	Der generierte code	13
3.5	Das Hauptprogramm	24

**Anhang: Quelltext des Laufzeitsystems**



### 3. Das Laufzeitsystem

Das CSSA-Laufzeitsystem besteht aus der virtuellen stack-Maschine, der simulierten Mehrprozessorkonfiguration, dem verteilten multiprogramming-Betriebssystem, dem interface-Agenten, sowie einer Reihe von Hilfsprozeduren, die vom generierten code aus aufgerufen werden.

Der vom CSSA-Compiler generierte code enthält einige %COPY-Anweisungen, was dazu führt, daß beim Weiterübersetzen mit dem SIMULA-Compiler die Moduln des CSSA-Laufzeitsystems in den generierten code eingebracht werden. Einen Überblick über das dabei entstehende vollständige SIMULA-Programm gibt die unten gezeigte Abbildung. Ein komplettes derartiges Programm mit allen Moduln ist im Anhang abgedruckt.

Ein ganzes Paket von Klassen- und Prozedurdefinitionen wird durch die **CLASS CSSA** eingebracht: Hier sind die 3 aktiven Einheiten der Simulation (Prozessoren, Busse, Module) definiert; außerdem eine Fülle von Hilfsprozeduren zum Ausdrucken von Werten der verschiedenen CSSA-Typen in Systemmeldungen, sowie verschiedene Kontrollblöcke. Im ausführbaren Teil der **CLASS CSSA** werden die Angaben über die zu benutzende Konfiguration aus einer Datei eingelesen.

Je nachdem, welche Konstrukte im CSSA-Quellprogramm verwendet wurden, werden noch verschiedene class-Definitionen aus der Modul-Bibliothek **hinzugeladen**.

Die Datei mit den vom Benutzer definierten **externen Routinen** wird nicht durch %COPY, sondern durch ein EDOR-Programm vom Entwicklungssystem (→ E.1) einkopiert. Die **built-in Funktionen** werden allesamt einkopiert, falls im CSSA-Programm wenigstens eine solche Funktion verwendet wurde.

Das Laufzeitsystem stellt bereits die erste script-Definition zur Verfügung, nämlich die des **interface-Agenten**. Alle anderen scripts werden ebenfalls in Klassen der Oberklasse MODULE übersetzt. Der Compiler definiert außerdem für jeden Block des CSSA-Programms den Aufbau des entsprechenden Aktivierungssatzes mit den lokalen Variablen. Die Oberklasse AS aller **Aktivierungssätze** ist in der **CLASS CSSA** definiert.

Beim Ausführen des Programms bewirkt das Statement 'CSSA begin' die Generierung eines Objektes der Klasse CSSA. Dabei wird der Statementteil der Klasse ausgeführt und die Konfiguration eingelesen. Als nächstes wird dann der 'CSSA begin'-Block ausgeführt, wobei alle Attribute der **CLASS CSSA** sichtbar sind. Die ausführbaren Statements dieses Blocks bewirken neben einigen Initialisierungen die Generierung der Busse und Prozessoren (entsprechend der eingelesenen Beschreibung der Prozessoren). Außerdem wird bereits der interface-Agent gegründet, d.h. ein ACB (Agent Control Block) und ein MODULE vom Typ SCRIPT1 werden gegründet und auf Prozessor 1 gelegt. Schließlich wird die Simulation durch die Aktivierung von Prozessor 1 und ein HOLD(1000000) gestartet.

## Grobstruktur des generierten codes:

begin

Simulation class CSSA

```
Process class PROCESSOR
(* tracing-Routinen *)
(* stack-Operationen *)
(* BS-Kern *)
```

```
Process class BUS ...
Process class MODULE ...
class ACB ...
(* print+dump-Routinen *)
class MSG ...
class AS ...

(* Einlesen der Konfiguration*)
```

CSSA begin

```
(* externe benutzerdefinierte Funktionen *)
(* built-in Funktionen *)
```

```
class RELATION ...
class RECORD ...
...
...
class ARRAY_DESC ...
```

```
MODULE class SCRIPT1 (*interface*)
(* Scanner *)  

(* Symboltafelle *)  

(* Hilfsprozeduren *)  

(* Parser *)  

(* Semantische Routinen *)

(* Hauptprogramm *)
```

```
AS class AS1 ...
AS class AS2 ...
...
AS class ASn ...
```

```
MODULE class SCRIPT2 ...
MODULE class SCRIPT3 ...
...
MODULE class SCRIPTm
(* Initialisierungen *)
(* Generierung von Bussen und Prozessoren *)
(* Start der Simulation *)
```

end  
end

### 3.1 Die Klasse CSSA

Das größte Modul des CSSA-Laufzeitsystems ist die Klasse CSSA. Hier ist das Verhalten der Prozessoren definiert, die neben Bussen und Modulen (also den CSSA-scripts) die aktiven Einheiten der Simulation sind.

Die CLASS PROCESSOR enthält zunächst eine Routine zum Ausdrucken der STATUS-Information. Es folgt für jedes Ereignis, das beim tracing gemeldet werden kann, eine Routine. Die Aufrufe dieser Routinen werden vom Compiler an der entsprechenden Stelle in den generierten code eingebracht. Über den Zeiger ACTIVE\_ACB kann von den Routinen aus auf den Kontrollblock (ACB) des gerade auf dem Prozessor aktiven Agenten (der die Routine ja aufgerufen hat) zugegriffen werden. So kann beispielsweise abgefragt werden, ob überhaupt das trace-bit gesetzt ist. Mit der Routine NEW\_TRACE\_SECTION wird ein Abschnitt des tracings im Protokoll gekennzeichnet. Sie wird daher stets nach einer Stelle im code aufgerufen, wo der Agent von seinem Prozessor unterbrochen werden kann. Die Prozedur END\_OF\_SECTION kennzeichnet das Ende eines tracing-Abschnitts und wird vor einer möglichen Unterbrechung aufgerufen.

Als nächstes enthält die CLASS PROCESSOR eine Reihe von Prozeduren, die die möglichen Operationen des Prozessors beschreiben. Die Prozedur NEXT\_MESSAGE durchsucht die mailbox des aktiven Agenten (die als verkettete Liste organisiert ist) nach der nächsten Nachricht.

```
procedure NEXT_MESSAGE;
inspect ACTIVE_ACB do
CURRENT_MSG:- if CURRENT_MSG== none then MAILBOX.FIRST
else CURRENT_MSG.SUC;
```

Der Zeiger auf die gefundene Nachricht (CURRENT\_MSG) wird im ACB des Agenten abgespeichert. Wird nach Aufruf von NEXT\_MESSAGE CURRENT\_MSG einmal NONE, so ist der Agent 'idle'. Bei einem nochmaligen Aufruf von NEXT\_MESSAGE wird die mailbox wieder von vorn durchsucht. Für das Verständnis der Prozedur ist es wichtig zu wissen, daß beim Löschen einer Nachricht aus der mailbox und beim Facettieren der Zeiger CURRENT\_MSG wieder auf den Anfang der mailbox zurückgesetzt wird, da dann wieder alle Nachrichten für die Auswahl in Frage kommen (Fairness).

Die weiteren **Operationen** des Prozessors definieren die Arbeitsweise der **stack-Maschinen**. Jeder ACB enthält zu jedem Standard-Datentyp einen stack, der Werte dieses Typs speichern kann. Der Prozessor enthält für jeden dieser stacks PUSH- und POP-Operationen. Darüberhinaus sind hier die bei einem Datentyp jeweils definierten Verknüpfungen definiert. Beispielsweise bewirkt der Aufruf der Prozedur ADD, daß die beiden obersten Elemente des INT-stacks durch ihre Summe ersetzt werden. Arithmetische und logische Ausdrücke werden von Compiler in einen Folge von PUSH-, POP-, und Verknüpfungsaufrufen übersetzt.  
Beispiel: 5+6 --> PUSH(5);PUSH(6);ADD.

Der Statementteil der CLASS PROCESSOR stellt das interrupt-gesteuerte **multiprogramming-Betriebssystem** dar, das im Teil A dieser Arbeit ausführlich beschrieben ist. Nach Abarbeitung einer Unterbrechung wählt der Prozessor den nächsten zu aktivierenden Agenten (seinen ACB) aus der Warteschlange aus und lässt ihn rechnen, indem er das zugehörige script aktiviert (mit ACTIVATE) und selbst HOLD(TIMESLICE) aufruft. Das Betriebssystem erhält die Kontrolle, wenn entweder die Zeitscheibe (TIMESLICE) abgelaufen ist, der rechnende Agent per interrupt einen Betriebssystemdienst anfordert (z.B. Versenden einer Nachricht) oder der Prozessor von einem Bus unterbrochen wird, der eine Nachricht für ihn hat.

Neben den Prozessoren sind auch die **Datenbusse** aktive Einheiten der Simulation (PROCESS CLASS BUS). Die Busse wählen jeweils eine Nachricht aus ihrer Warteschlange aus, verbrauchen die Übertragungszeit (HOLD(TRANS\_TIME)) und unterbrechen dann den Zielprozessor, damit er die Nachricht entgegennimmt. Eine genauere Beschreibung der Zusammenhänge findet sich ebenfalls in Teil A. Bezuglich des agent control blocks (ACB) und der Oberklasse für alle vom Compiler generierten script-Module (PROCESS CLASS MODULE) sei ebenfalls auf Teil A verwiesen.

Als nächstes folgen in der CLASS CSSA eine Reihe von Klassen- und Prozedurdefinitionen:

Die in jedem ACB enthaltenen **stacks** für jeden CSSA-Datentyp sind als einfach verkettete Listen organisiert. Die Listenelemente enthalten genau ein stack-Element des entsprechenden Datentyps, sowie einen Zeiger auf das nächste Element.

Für jeden CSSA-Datentyp steht eine **print-Prozedur** zur Verfügung, die einen Wert in dem dem Datentyp zugeordneten Standardformat ausgibt. Beispielsweise werden INT-Zahlen genau in der notwendigen Feldweite ausgegeben.

Für jeden Datentyp ist außerdem eine **dump-Prozedur** definiert, die eine Zeile des Laufzeitkellers (siehe DUMP-Kommando im interface) eines Agenten ausgibt. Es werden die senkrechten Begrenzungslinien, der symbolische Name der Variablen sowie (mit Hilfe der entsprechenden print-Prozedur) ihr Wert ausgegeben. Den gesamten Laufzeitkeller gibt die Prozedur DUMP aus. Je nachdem, ob der Laufzeitkeller aufgrund der SNAPSHOT-option oder durch ein explizites DUMP-Kommando ausgegeben wird, wird eine andere Überschrift über den Keller gesetzt (OBSERVE\_DUMP / PRINT\_DUMP)

Die Prozedur **PRINT\_STATUS** gibt eine Zeile der STATUS-Tabelle aus (also alle Informationen zu einem Agenten). Wird an den Booleschen Parameter SENDING der Wert TRUE übergeben, so wird anstelle der mailbox die soeben vom Agenten versendete Nachricht hinter '==>' ausgegeben. (Vgl. EVENT-option im interface).

Mit Hilfe der Prozedur **PRINT\_STACKS** kann zu Testzwecken der Inhalt der jedem Datentyp zugeordneten stacks eines Agenten ausgegeben werden (vgl. STACKS-Kommando im interface).

**Nachrichten** werden im Laufzeitsystem als Objekte der CLASS MSG dargestellt. Sie haben die Oberklasse LINK und können daher in verkettete Listen wie z.B. die mailbox oder Warteschlangen vor Bussen eingetragen werden. Neben Sender und Empfänger der Nachricht finden sich dort Angaben, ob überhaupt, von wem und für welchen port eine Antwort gefordert ist. Schließlich enthalten Nachrichten ihrerseits eine verkettete Liste der Parameter. Ein Parameter-Objekt enthält genau einen Wert eines CSSA-Standarddatentyps.

Die CLASS AS ist die Oberklasse für alle vom Compiler generierten **Aktivierungssätze**. Der Laufzeitkeller eines Agenten ist eine verkettete Liste von solchen Aktivierungssätzen. Alle Angaben, die zu jedem Aktivierungssatz gehören, wie z.B. der Zeiger auf den dynamischen Vorgänger (LAST\_AS), den statischen Vorgänger (E wie environment) und die Rücksprungadresse sind in der Oberklasse gespeichert, und werden beim DUMP-Kommando des Interface im Druckbild des Laufzeitkellers über der gepunkteten Linie ausgegeben. Die speziellen lokalen Variablen sind in der Unterkategorie definiert. Dort wird auch die als 'virtual' spezifizierte Prozedur PRINT\_VARS definiert, die den Teil des Aktivierungssatzes unter der gepunkteten Linie ausgibt.

Während der interaktiven CSSA-Sitzung wird jegliche **Ausgabe** zunächst in den (extrem langen) Puffer der Datei BMSOUT geschrieben. Erst durch den Aufruf der Prozedur OUTLINE wird eine Zeile sowohl auf den Bildschirm, als auch in die Protokoll-Datei ausgegeben. Dabei muß der Pufferinhalt eventuell auf mehrere Zeilen umgebrochen werden. Die etwas umständliche Prozedur ist notwendig, da es in SIMULA keine Möglichkeit gibt, ein implizites OUTIMAGE abzufangen, um den Pufferinhalt in zwei verschiedene Dateien auszugeben.

Im ausführbaren Teil der Klasse CSSA wird die vom Benutzer spezifizierte **hardware-Konfiguration** eingelesen und abgespeichert. Die ersten 25 Zeilen der Datei dienen lediglich als Kommentar zur Beschreibung der Konfiguration und werden auf den Bildschirm ausgegeben.

### 3.2 Die Modul-Bibliothek

Am Anfang des 'CSSA begin'-Blocks stehen einige Definitionen, die nur bei Bedarf zu dem generierten code hinzukopiert werden: Der Compiler generiert nur für diejenigen Module ein %COPY, die tatsächlich benötigt werden. Beispielsweise wird die CLASS RELATION nur dann einkopiert, wenn im Quellprogramm wenigstens eine Relation verwendet wurde.

Als erstes kommen die vom Benutzer direkt in SIMULA geschriebenen **externen Prozeduren**, die zu jeder als 'external' vereinbarten Funktion oder Prozedur vorhanden sein müssen.

Es folgen die Definitionen der von CSSA zur Verfügung gestellten **built-in Funktionen**. Beispielsweise wird ein Aufruf der Funktion SUBSTR im CSSA-Programm in einen Aufruf der Funktion CSSASUBSTR übersetzt.

**Relationen** werden zur Laufzeit als Objekte der Klasse RELATION dargestellt. Ist in einem CSSA-Block eine Relation definiert, so enthält der entsprechende Aktivierungssatz eine Variable vom Typ REF(RELATION) (siehe AS CLASS AS1, Zeile 5102). Die Variable wird durch NEW RELATION initialisiert. Ein Objekt vom Typ RELATION enthält einen Verweis auf das erste Element einer verketteten Liste von Records; außerdem Prozeduren zum Aufsuchen, Einfügen und Löschen von Records aus der Liste. Die CSSA-Befehle 'find', 'insert' und 'delete' werden in Aufrufe solcher Prozeduren übersetzt.

Die CLASS RECORD stellt die Oberklasse für alle vom Compiler generierten **record-Definitionen** dar (siehe RECORD CLASS RECORD104, Zeile 5160). In der Unterklassse werden einige Prozeduren definiert, die schon in der Oberklasse als 'virtual' vereinbart sind:

- ASSIGN(I) – dem i-ten Feld des Records wird der Wert des obersten Elementes des entsprechenden stacks zugewiesen.
- PUSH(I) – der Wert des i-ten Feldes wird in den entsprechenden stack geschoben.
- EQUAL(X) – der Schlüssel des records wird mit dem eines anderen Records gleichen Typs verglichen.

Die Prozedur EQUAL wird in der Klasse RELATION zur Lokalisierung eines Records in der Relation verwendet. In der Oberklasse RECORD ist nur die Anzahl der Felder und die Anzahl der Schlüsselfelder bekannt. Mit Hilfe der virtuellen Prozeduren können dennoch Prozeduren definiert werden die unter anderem alle (oder nur die Schlüssel-) Felder des Records auf die stacks pushen, alle (oder nur die Schlüssel-) Felder mit Werten aus den stacks überschreiben, alle Felder mit den Parametern einer Nachricht überschreiben oder alle Felder als Parameter in eine Nachricht schicken.

Da CSSA dynamische array-Grenzen erlaubt, existiert zur Laufzeit ein **array-Deskriptor**, in dem zu jeder Dimension die array-Grenzen abgespeichert werden. Im Aktivierungssatz für denjenigen Block, in dem der array-Typ deklariert ist, steht eine Variable vom Typ REF(ARRAY\_DESC). Beim Anlegen des Aktivierungssatzes werden zunächst die array-Grenzen berechnet und dann ein Objekt vom Typ ARRAY\_DESC angelegt. Der array-Deskriptor liest bei seiner Generierung die array-Grenzen selbst aus dem INT-stack. Array-Objekte selbst (CLASS FIELD) haben zunächst einen Verweis auf den zugeordneten array-Deskriptor. Alle array-Objekte haben die gemeinsame Oberklasse FIELD. Als Unterklassse fungiert eine vom Basistyp des arrays abhängige Klasse: Bspw. FIELD CLASS INT\_ARRAY für arrays von INT-Zahlen. In der Oberklasse werden für diejenigen Operationen eine Reihe von Prozeduren definiert, die unabhängig vom Basistyp für alle arrays gleich aussehen. Allerdings werden hier schon die als 'virtual' vereinbarten Prozeduren der Unterklassse verwendet, die letztlich den Zugriff auf die Elemente des arrays ausführen:

- PUSH\_ALL schreibt alle Elemente des arrays in den entsprechenden stack. Benutzt wird die virtuelle Prozedur PUSH\_ELEMENT, die ein Element in den stack schreibt.
- ASSIGN\_ALL überschreibt alle Elemente des arrays mit Werten aus dem stack und benutzt dabei die virtuelle Prozedur ASSIGN\_ELEMENT.
- SET\_PARMS trägt alle Elemente als Parameter in eine message ein und benutzt SET\_PARM.
- GET\_PARM überschreibt ein Element mit dem Feld einer Nachricht und benutzt ASSIGN\_ELEMENT.
- GET\_PARMS überschreibt alle Elemente mit dem Inhalt einer Nachricht und benutzt ASSIGN\_ELEMENT.
- PUSH entnimmt die Indizes aus dem INT-stack und schreibt das entsprechende Element in den stack. Benutzt wird PUSH\_ELEMENT.
- ASSIGN entnimmt die Indizes aus dem INT-stack und überschreibt das entsprechende Element mit dem Werte aus dem stack des Basistyps.

**Mengen** werden im Laufzeitsystem als doppelt verkettete Listen dargestellt. Dabei wird die von SIMULA vorgegebene Systemklasse SIMSET mit ihren lokalen Klassen HEAD und LINK verwendet. Für jeden CSSA-Datentyp gibt es eine entsprechende Klasse. Bspw. HEAD CLASS INT\_SET für Mengen von INT-Zahlen. In einer solchen Liste werden Objekte vom Typ LINK CLASS INT\_ELEMENT verkettet, die genau eine INT-Zahl enthalten. In der Klasse INT\_SET gibt es Prozeduren zu Einfügen (PUT) und Entfernen (REMOVE) von Elementen, außerdem eine Prozedur, die überprüft, ob ein Element in der Liste enthalten ist (TEST). Die CSSA-Befehle PUT und REMOVE, sowie der IN-Operator werden in Aufrufe dieser Prozeduren übersetzt. Mit Hilfe der Prozeduren ASSIGN\_ALL und PUSH\_ALL kann die Zuweisung ganzer Mengen vorgenommen werden: PUSH\_ALL schreibt alle Elemente der Menge in den INT-stack und anschließend die Anzahl der Elemente. ASSIGN\_ALL löscht die Menge und entnimmt soviele Elemente wie die oberste Zahl im INT-stack angibt aus dem INT-stack. Schleifen für Mengen werden mit den Prozeduren START und INCREM realisiert. START setzt den Zeiger LOOP\_INDEX auf das erste Element, und INCREM setzt ihn um eine Position weiter. Da der Zeiger lokal in der Klasse INT\_SET deklariert ist, gibt es nur eine solchen Zeiger pro INT\_SET. Geschachtelte Schleifen für ein und denselben set sind daher nicht möglich.

### 3.3 Der interface-Agent

Das script des interface-Agenten ist wie alle anderen (vom Compiler generierten) scripts auch von der Oberklasse PROCESS CLASS MODULE. Er stellt sich daher nach außen genau wie alle anderen Agenten dar und wird vom Betriebssystem genauso behandelt.

Der interface-Agent besteht im wesentlichen aus dem **Interpreter** für die interaktiv eingegebenen Kommandos. Der Interpreter läßt sich in mancher Hinsicht mit dem CSSA-Compiler vergleichen und ist sehr ähnlich aufgebaut. Der Sprachumfang ist allerdings wesentlich kleiner und dementsprechend sind auch Scanner und Parser kürzer. Wegen der fehlenden Blockstruktur in der interface-Sprache ist die Organisation der Symboltabelle einfacher. Eine aufwendige Fehlerbehandlung ist naturgemäß in einem Interpreter überflüssig, so daß wesentlich einfachere Hilfsprozeduren für den Parser ausreichen. Und schließlich wird in den semantischen Routinen nicht code generiert, sondern die notwendigen Aktionen werden unmittelbar ausgeführt (z.B. bewirkt ein SEND-Kommando das Versenden einer Nachricht, und nicht, daß code für das Versenden einer Nachricht generiert wird).

Der **Scanner** besteht aus der Prozedur NEXTSYMB mit den lokalen Prozeduren NEWLINE und WARNING. Beim Erreichen des Zeilenendes wird NEWLINE aufgerufen. NEWLINE liest die nächste Zeile mit Hilfe des WRTRD (write/read) Makros vom Bildschirm oder aus einer Eingabedatei. C\_INFILe gibt stets an, von wo gelesen werden soll. Lexikalische Fehler werden mit Hilfe der Prozedur WARNING gemeldet.

Bei Einlesen des nächsten Symbols wird zunächst geprüft, ob das Zeilenende erreicht ist. In diesem Fall wird der Benutzer aufgefordert, das nächste Kommando einzugeben und NEWLINE wird aufgerufen. Anschließend werden alle blanks überlesen. Wird dabei Spalte 80 erreicht, so liefert der Scanner ein Semikolon; wird dabei ein '#' gefunden, so wird sofort die nächste Zeile gelesen, ohne daß das Zeilenende als Semikolon erkannt wird (wichtig für Fortsetzungszeilen). Beim Erreichen des ersten Nicht-blanks muß noch geprüft werden, ob ein Kommentar gefunden wurde.

Je nachdem, ob das gefundene Zeichen ein Buchstabe, eine Ziffer oder ein Sonderzeichen ist, wird ein Bezeichner, eine Zahlenkonstante oder eine Sonderzeichenkombination eingelesen. Bei Bezeichnern wird sofort in der Symboltabelle nachgesehen, ob es sich um einen neuen (NEW\_IDENTIFIER) oder einen bereits definierten (OLD\_IDENTIFIER) Bezeichner handelt.

In jedem Falle sind bei Verlassen der Prozedur NEXTSYMB die beiden globalen Größen SYMBOL und TOKEN gesetzt. SYMBOL ist das gelesene Symbol als Text, während TOKEN eine dem Symbol zugeordnete INTEGER-ZAHL ist. Die interne Verschlüsselung der Symbole kann der Initialisierungsphase des interface entnommen werden. Darüberhinaus verweist TBLREF stets auf den Symboltabelleneintrag des letzten gelesenen OLD\_IDENTIFIER. Im Falle von Zahlenkonstanten sind die globalen Größen REAL\_VALUE bzw. INT\_VALUE gesetzt.

Nach dem Scanner folgen zwei Prozeduren zum Ausdrucken von **Fehlermeldungen**: SEM\_ERROR meldet einen Fehler der statischen Semantik, der vom Interpreter erkannt wurde, wie beispielsweise ein Typ-Fehler. SYNTAX\_ERROR meldet einen syntaktischen Fehler. Dabei wird das fehlerhafte Symbol ausgegeben. Handelt es sich dabei um das END\_OF\_FILE Symbol, so wird das interface terminiert. Neben dem falschen Symbol werden auch alle an der Fehlerstelle erwarteten Symbole ausgegeben. Die Symbole stehen in dem array PASSED\_TOKENS, das bis zu der Stelle TOKENS\_PASSED gefüllt ist. Wie die Liste der erwarteten Symbole aufgestellt wird, ist weiter unten im Zusammenhang mit der Prozedur THERE\_IS beschrieben. Sowohl nach Semantik- als auch nach Syntaxfehlern wird zu der Marke FIND\_SEMIKOLON im Hauptprogramm des interface gesprungen. Dort wird solange Eingabe überlesen, bis das nächste Semikolon - also das Ende eines Statements - erreicht ist. Man beachte dabei, daß auch das Zeilenende als Semikolon erkannt wird.

Die **Symboltabelle** des Interpreters ist als binärer Suchbaum organisiert, wodurch ein schneller Zugriff möglich ist. Die Knoten des Baumes sind Objekte vom Typ ID, in denen zu jedem Bezeichner Name, Typ und Wert abgespeichert sind. Auch die Schlüsselwörter der interface-Sprache stehen in der Symboltabelle. Sie werden in der Initialisierungsphase des interface mit der Prozedur INSERTKW eingetragen. Dabei wird in dem Eintrag das flag KEYWORD gesetzt und unter TOKEN die interne Nummer des Schlüsselwortes abgespeichert.

Das Aufsuchen eines Bezeichners in der Symboltabelle geschieht durch die Prozedur LOOKUP, die vom Scanner aufgerufen wird. Stellt sich heraus, daß es sich bei dem Bezeichner um ein Schlüsselwort handelt, so wird die globale Größe TOKEN entsprechend gesetzt. Wird der Bezeichner nicht in der Tabelle gefunden, so wird TOKEN:=NEW\_IDENTIFIER gesetzt, andernfalls TOKEN:=OLD\_IDENTIFIER.

Mit Hilfe der rekursiven Prozedur PRINT(SUBTREE) kann die gesamte Variablenliste ausgegeben werden. Das DISPLAY-Kommando ruft dementsprechend PRINT(SYMBTBL) auf.

Mit Hilfe der Prozedur INSERT\_SCRIPT werden die Namen der vom Compiler übersetzten Scripts während der Initialisierungsphase des interface in die Symboltabelle eingetragen. Sie sind daher von Anfang an in der interaktiven CSSA-Sitzung bekannt.

Für die Strukturierung des **Parsers** stehen einige Hilfsprozeduren zur Verfügung. Mit Hilfe von READ, THERE\_IS und NEW\_IDENTIFIER ist eine sehr einfache und direkte Abbildung einer Regel in erweiterter BNF in eine Parsing-Prozedur für den rekursiven Abstieg möglich. Als Beispiel sei hier gezeigt, wie eine PORT-Deklaration umgesetzt wird:

---

Regel: <PORT\_DECL> ::= PORT '::' <IDENTIFIER> || ','

Die zugehörige Parser-Prozedur:

```

procedure PORT_DECL;
begin READ(PORTSY);
  READ(COLONSY);
  NEW_IDENTIFIER;
  while THERE_IS(COMMASY) do
    begin READ(COMMASY);
      NEW_IDENTIFIER;
    end;
end;

```

Die Prozedur READ vergleicht das lookahead-Symbol mit dem angegebenen Parameter. Bei Übereinstimmung wird das nächste Symbol vom Scanner eingelesen, andernfalls liegt ein Syntaxfehler vor.

Die Entscheidungen des Parsers werden durch die Prozedur THERE\_IS getroffen, die überprüft, ob ein bestimmtes lookahead-Symbol vorliegt. Falls ja, wird die entsprechende syntaktische Alternative eingeschlagen, andernfalls wird sie verworfen und das überprüfte Symbol wird in die Liste der momentan erlaubten Fortsetzungszeichen aufgenommen, die im Fehlerfall ausgegeben wird. Diese Liste wird jedesmal wieder gelöscht, wenn der Scanner ein neues Symbol einliest. Im Fehlerfall wird also die Steuersymbole aller unmittelbar vorher verworfenen Alternativen ausgegeben.

Die Prozedur NEW\_IDENTIFIER bewirkt i.W. das gleich wie READ(NEWIDSY), allerdings wird außerdem noch der neue Identifier in die Symboleitabelle eingetragen (durch DECLARE).

Für **Typ-Überprüfungen** stehen die beiden Prozeduren TYPE\_CHECK und COMPARE\_TYPES zur Verfügung. TYPE\_CHECK überprüft den Typ des zuletzt gelesenen Bezeichners, auf den ja der Zeiger TBLREF verweist. Mit COMPARE\_TYPES werden zwei als Texte gegebene Datentypen verglichen. Bei Ungleichheit erfolgt eine semantische Fehlermeldung.

Der Parser ist – wie oben bereits erwähnt – nach dem Prinzip des **rekursiven Abstiegs** geschrieben. Zu jeder syntaktischen Regel gibt es genau ein Parserprozedur, die die längste mögliche Zeichenkette der Eingabe einliest, die sich aus der Regel ableiten lässt. In jeder Prozedur steht als Kommentar die zugehörige Regel. Die im Handbuch abgedruckte Syntax wurde übrigens durch ein Programm aus diesen Kommentaren extrahiert; die Gefahr, daß sich Programm und Dokumentation auseinanderentwickeln würde durch diese Methode auf ein Minimum reduziert. Die Parserprozeduren sind weitgehend durch die systematische Umsetzung der Regeln in SIMULA-code nach dem oben erläuterten Prinzip entstanden. In das so entstandene Gerippe jeder Prozedur wurden dann die Überprüfung der statischen Semantik sowie die durch die Kommandos ausgelösten Aktionen eingebaut.

Aus der Beschreibung von Syntax und Semantik der verschiedenen Kommandos heraus sind die meisten Parserprozeduren unmittelbar verständlich, hier sollen daher einige stichpunktartige Anmerkungen genügen:

- Das TERMINATE-Statement drückt die SESSION-STATISTICS aus und beendet die interaktive Sitzung. Dazu genügt es nicht, daß der interface-Agent terminiert (die übrigen Agenten könnten dennoch weiterrechnen), sondern durch REACTIVATE MAIN wird die gesamte Simulation abgebrochen.
- Wird eine <AGENT-DENOTATION> bestehend aus script-Name und einer laufenden Nummer angegeben, so wird die globale Liste AGENT\_LIST durchsucht, in der Verweise auf alle existierenden ACB's gespeichert sind. Nach Aufruf von AGENT-DENOTATION ist in der globalen Variablen RES\_AGENT stets ein Verweis auf den bezeichneten Agenten gespeichert, der dann von der rufenden Prozedur benutzt werden kann.
- Die Kommandos zum Ein- und Ausschalten der verschiedenen options setzen lediglich flags, die entweder systemglobal oder in einem ACB abgespeichert sind.
- Es gibt zwei Zeiger PROT und PROTOKOL. PROTOKOL verweist stets auf die Protokoll-Datei und PROT zeigt entweder auf NONE oder ebenfalls auf die Protokoll-Datei, je nachdem ob die Protokollierung eingeschaltet ist oder nicht. Beim Einschalten des Protokolls wird der Kommentarteil der Beschreibung der hardware-Konfiguration nochmals eingelesen und ins Protokoll geschrieben. Am Anfang der Sitzung, wenn die Beschreibung der hardware-Konfiguration auf den Bildschirm ausgegeben wird, kann nämlich die Protokollierung noch nicht eingeschaltet sein.
- Im RUN-Statement verbraucht der interface-Agent die angegebene Simulationzeit durch einen HOLD-Aufruf. Allerdings wird nicht sofort die gesamte Zeit verbraucht, sondern es wird in einer Schleife HOLD(0.1) aufgerufen. Dadurch erhält das interface nach je 0.1 Sekunden zumindest kurz die Kontrolle und kann überprüfen, ob das Agentennetz womöglich bereits terminiert ist. Außerdem wird gemessen, ob für die Simulation von 0.1 Sekunden mehr als das mit 5 Sekunden vorbesetzte Realzeitlimit benötigt wurde. Auch in diesem Fall erhält das interface vorzeitig (d.h. ohne daß die gesamte hinter RUN angegebene Simulationszeit verbraucht wurde) die Kontrolle zurück.
- Das SEND- und das REPLY-Statement bewirken einen Aufruf an das Betriebssystem: Der dem interface zugeordnete Prozessor wird unterbrochen und das Betriebssystem versendet die gewünschte Nachricht.
- Der Aufruf von EXPRESSION oder FACTOR bewirkt stets, daß die globale Variable RES\_TYPE gesetzt wird und den Typ des soeben gelesenen Ausdrucks angibt. Außerdem steht der Wert des Ausdrucks dann in einem der stacks im ACB, die vom interface genauso benutzt werden, wie alle anderen Agenten ihre stacks benutzen.

Für einige mehrfach verwendete **semantische Aktionen** gibt es hinter dem Parser einige Prozeduren, die aus den Parserprozeduren heraus aufgerufen werden:

- SET\_PARM trägt das oberste Element des durch RES\_TYPE bezeichneten stacks in eine Nachricht ein.
- GET\_PARM schreibt den Wert des nächsten Parameters in den entsprechenden stack.
- ASSIGN(IDENT) weist dem durch IDENT angegebenen Identifier das oberste Element des dem Typ des Bezeichners entsprechenden stacks zu.
- PRINT\_TOP(TYPE) druckt das oberste Element eines stacks im CSSA-Standardformat aus.
- POP(TYPE) poppt einen stack
- PUSH(IDENT) schiebt den Wert eines Bezeichners aus der Symboltabelle in einen stack.

In der **Initialisierungsphase** des interface-Agenten werden zunächst den verschiedenen Token eindeutige Nummern zugeordnet. Um Token in Meldungen ausgeben zu können werden ihnen in dem array TOKENTEXT strings zugeordnet. Beispiel: TOKENTEXT(ASSIGNSY)=".=". Anschließend werden alle Schlüsselwörter mit INSERTKW in die Variablenliste eingetragen. Um die Namen der vom Compiler übersetzten scripts eintragen zu können, muß zunächst die vom Compiler generierte Prozedur DEFINE\_SCRIPT\_NAMES aufgerufen werden, die die globale Größe NR\_OF\_SCRIPTS setzt und die Namen aller scripts in das array SCRIPT\_NAME einträgt. Dieses array wird stets verwendet, wenn der print-Name eines scripts ausgegeben werden soll.

Das **Hauptprogramm** des interfaces sieht (in leicht komprimierter FORM) folgendermaßen aus:

```

NEXT_INPUT: COMMAND; goto NEXT_INPUT;
FIND_SEMICOLON: while NEXTCHAR=';' and C_INFILE.Pos/=/80 do
                 NEXTCHAR:=C_INFILE.Inchar;
                 goto NEXT_INPUT;

```

Es wird also immer wieder die Parserprozedur COMMAND aufgerufen, wodurch das nächste Kommando gelesen wird. Im Falle eines Fehlers kehrt die Kontrolle nach FIND\_SEMIKOLON zurück, wo bis zum nächsten Semikolon Eingabe überlesen wird.

### 3.4 Der generierte code

Es sollen hier einige Erläuterungen zur Struktur des vom Compiler generierten script-codes und der zugehörigen Aktivierungssätze gegeben werden. Dabei wird aber nicht - wie in den anderen Abschnitten dieses Kapitels - auf das im Anhang abgedruckte SIMULA-Programm Bezug genommen, sondern es wird der code zu dem folgenden kleinen CSSA-script zugrundegelegt.

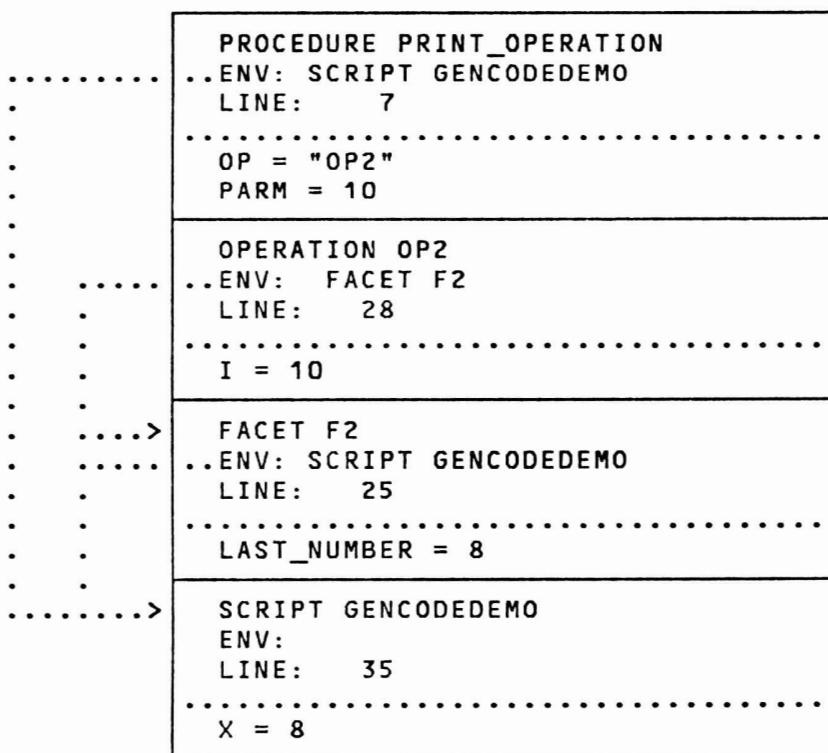
```

1 type GENCODEDEMO is
2 script
3
4   var int: X;
5
6   procedure PRINT_OPERATION( string: OP; int: PARM) is
7     print ("OPERATION ",OP," WURDE MIT ",PARM," AUFGERUFEN");
8   endprocedure;
9
10  facethead F2( int: LAST_NUMBER);
11
12  facet F1 is
13    public: OP1;
14
15    operation OP1 (--> X) is
16      const string: MY_NAME:="OP1";
17
18      call PRINT_OPERATION(MY_NAME,X);
19      replace by F2(X);
20    endoperation
21
22  endfacet
23
24  facet F2 is
25    public: OP2;
26
27    operation OP2 ( int: I) assert I>LAST_NUMBER is
28      call PRINT_OPERATION("OP2",I);
29      setup F1;
30    endoperation
31
32  endfacet
33
34  initial F1
35 endscript

```

Den Inhalt des Laufzeitkellers eines zugehörigen Agenten während der Ausführung der Prozedur PRINT\_OPERATION, wie er durch das DUMP-Kommando ausgegeben wird, zeigt die folgende Abbildung. Zur Veranschaulichung wurden nachträglich die Zeiger auf den jeweils statisch übergeordneten (also sichtbaren) Aktivierungssatz eingezeichnet.

+++ 0.500 RUNTIME STACK OF GENCODEDEMO(1)



Es folgt der bei der Übersetzung des scripts erzeugte code. Die der SIMULA-Syntax entsprechenden Kommentare (hinter COMMENT) wurden vom Compiler erzeugt. Weitere Kommentare (zwischen (\*) und \*) wurden nachträglich eingefügt. In den Erläuterungen wird auf die vom Compiler generierte Numerierung Bezug genommen, nach der der generierte code sortiert wurde.

%TITLE	GENCODE.AKTIVIERUNGSSAETZE	0	1A
AS CLASS AS1;		0	1A
BEGIN		0	1A
INTEGER VAR104;		0	1B
PROCEDURE PRINT_VARS;		0	1P
BEGIN		0	1P
DUMP_INT("X",VAR104);		0	1R
END -- PRINT_VARS --;		0	1X
SCOPE_TYPE:-COPY("SCRIPT");		0	1X
SCOPE_NAME:-COPY("GENCODEDEMO");		0	1X
END;		0	1Z
AS CLASS AS2;		0	2A
BEGIN		0	2A
TEXT VAR106;		0	2B
INTEGER VAR107;		0	2B
PROCEDURE PRINT_VARS;		0	2P
BEGIN		0	2P
DUMP_STRING("OP",VAR106);		0	2R
DUMP_INT("PARM",VAR107);		0	2R
END -- PRINT_VARS --;		0	2X
SCOPE_TYPE:-COPY("PROCEDURE");		0	2X

```

SCOPE_NAME:-COPY("PRINT_OPERATION");
0 2X
END;
0 2Z
AS CLASS AS3;
0 3A
BEGIN
0 3A
    INTEGER VAR109;
0 3B
    PROCEDURE PRINT_VARS;
0 3P
    BEGIN
0 3P
        DUMP_INT("LAST_NUMBER",VAR109);
0 3R
    END -- PRINT_VARS --;
0 3X
    SCOPE_TYPE:-COPY("FACET");
0 3X
    SCOPE_NAME:-COPY("F2");
0 3X
END;
0 3Z
AS CLASS AS4;
0 4A
BEGIN
0 4A
    PROCEDURE PRINT_VARS;
0 4P
    BEGIN
0 4P
    END -- PRINT_VARS --;
0 4X
    SCOPE_TYPE:-COPY("FACET");
0 4X
    SCOPE_NAME:-COPY("F1");
0 4X
END;
0 4Z
AS CLASS AS5;
0 5A
BEGIN
0 5A
    TEXT VAR112;
0 5B
    PROCEDURE PRINT_VARS;
0 5P
    BEGIN
0 5P
        DUMP_STRING("MY_NAME",VAR112);
0 5R
    END -- PRINT_VARS --;
0 5X
    SCOPE_TYPE:-COPY("OPERATION");
0 5X
    SCOPE_NAME:-COPY("OP1");
0 5X
END;
0 5Z
AS CLASS AS6;
0 6A
BEGIN
0 6A
    INTEGER VAR114;
0 6B
    PROCEDURE PRINT_VARS;
0 6P
    BEGIN
0 6P
        DUMP_INT("I",VAR114);
0 6R
    END -- PRINT_VARS --;
0 6X
    SCOPE_TYPE:-COPY("OPERATION");
0 6X
    SCOPE_NAME:-COPY("OP2");
0 6X
END;
0 6Z

%TITLE      GENCODE.SCRIPT  GENCODEDEMO      2  1A
COMMENT *** ANFANG DES SCRIPT GENCODEDEMO ***; 2  1A
MODULE CLASS SCRIPT2;
2  1A
BEGIN
2  1A
OWNMODE:=2;
2  1A
PRINT_NAME:-COPY("GENCODEDEMO");
2  1A
MAIN_ENTRY:
2  1A
INSPECT ASS_PROCESSOR DO INSPECT ACTIVE_ACB DO BEGIN 2  1A
SWITCH BRANCH:=
2  1B
RAS1,RAS2,RAS3,RAS4,RAS5;
2  1B
SWITCH ENTRY:=
2  1B
ENT1,ENT2,ENT3,ENT4,ENT5,ENT6,ENT7,ENT8,ENT9,ENT10,
2  1B
ENT11,ENT12,ENT13,ENT14;
2  1B
GOTO ENTRY(EINSPRUNG);
2  1C
ENT1:   (* erstmaliges Betreten des scripts *)
2  1C
CURRENT_LINE:=4;
2  1C
NEW_TRACE_SECTION;
2  1C

```

---

```

AKT_AS:-NEW AS1(AKT_AS,ENVIRONMENT);           2   1D
CURRENT_LINE:=4;                                2   1I
IF SNAPSHOT THEN OBSERVE_DUMP(SELF);           2   1I
GOTO START2; (* Sprung ans Ende des scripts *)  2   1Z

COMMENT *** ANFANG DER PROCEDURE PRINT_OPERATION ***; 2   2A
PROC2:
CURRENT_LINE:=6;
(* Aktivierungssatz anlegen *)
AKT_AS:-NEW AS2(AKT_AS,ENVIRONMENT);           2   2D
(* Parameter aus den stacks in AS holen *)
AKT_AS QUA AS2.VAR107:=INT_TOP.WERT;          2   2I
TRACE_INT("PARM");
INT_POP;                                         2   2I
AKT_AS QUA AS2.VAR106:=-STRING_TOP.WERT;        2   2I
TRACE_STRING("OP");
STRING_POP;                                       2   2I
IF SNAPSHOT THEN OBSERVE_DUMP(SELF);           2   2I
(* Statementteil der Prozedur *)
CURRENT_LINE:=7;                                2   2S
ELAPSED_TIME:=ELAPSED_TIME+ 0.1000;             2   2S
STRING_PUSH(COPY(
"OPERATION "
));
(* Simulationszeit verbrauchen *)
END_OF_SECTION;                                 2   2S
EINSPRUNG:=2;HOLD(ELAPSED_TIME);ELAPSED_TIME:=0; 2   2S
GOTO MAIN_ENTRY;ENT2:
NEW_TRACE_SECTION;                             2   2S
(* print-Befehl*)
OUTTEXT("****");PRINT_TIME;OUTTEXT(" ");
PRINT_AGENT(ACTIVE_ACB);OUTTEXT(" : ");
OUTTEXT(STRING_TOP.WERT);
STRING_POP;                                       2   2S
STRING_PUSH(AKT_AS QUA AS2.VAR106);            2   2S
OUTTEXT(STRING_TOP.WERT);
STRING_POP;                                       2   2S
STRING_PUSH(COPY(
" WURDE MIT "
));
OUTTEXT(STRING_TOP.WERT);
STRING_POP;
INT_PUSH(AKT_AS QUA AS2.VAR107);              2   2S
PRINT_INT(INT_TOP.WERT);
INT_POP;                                         2   2S
STRING_PUSH(COPY(
" AUFGERUFEN"
));
OUTTEXT(STRING_TOP.WERT);
STRING_POP;
OUTLINE;
(* Verlassen der Prozedur *)
AKT_AS:-AKT_AS.LAST_AS;                        2   2Z
GOTO BRANCH(AKT_AS.RAS);                      2   2Z
COMMENT *** ENDE DER PROCEDURE PRINT_OPERATION ***; 2   2Z

COMMENT *** ANFANG DER FACETTE F2 ***;         2   3A
FACET3:

```

---

```

CURRENT_OPER:-NOTEXT;                                2  3A
CURRENT_LINE:=24;                                   2  3A
AKT_AS:-NEW AS3(AKT_AS,ENVIRONMENT);                2  3D
AKT_AS QUA AS3.VAR109:=INT_TOP.WERT;                2  3I
TRACE_INT("LAST_NUMBER");                           2  3I
INT_POP;                                         2  3I
IF SNAPSHOT THEN OBSERVE_DUMP(SELF);               2  3I
COMMENT *** FACETTENSCHLEIFE VON F2 ***;           2  3L
NEXT3:
(* Entnahme der naechsten Nachricht aus der mailbox *)
NEXT_MESSAGE;                                     2  3L
IF CURRENT_MSG==NONE THEN GOTO IDLE3;              2  3L
COMMENT *** AUFRUF DER OPERATION OP2 ***;          2  30
BOOL_PUSH(CURRENT_MSG.OPER="OP2");                 2  30
BRANCH_FLAG:=-BOOL_TOP.WERT;BOOL_POP;             2  30
IF BRANCH_FLAG THEN GOTO NXT030113;                2  30
ENVIRONMENT:-AKT_AS;                             2  30
AKT_AS.RAS:=3;AKT_AS.RAS_LINE:=25;GOTO OPER113;RAS3: 2  30
IF SUCCESS THEN GOTO SUCC3;                       2  30
NXT030113:                                         2  30
GOTO NEXT3;                                       2  3T
(* Nach erfolgreicher Ausfuehrung einer Op.      *)
(* wird zunaechst Sim.-zeit verbraucht          *)
SUCC3:                                              2  3T
CURRENT_OPER:-NOTEXT;                            2  3T
END_OF_SECTION;                                 2  3T
EINSPRUNG:=10;HOLD(ELAPSED_TIME);ELAPSED_TIME:=0; 2  3T
GOTO MAIN_ENTRY;ENT10:                           2  3T
NEW_TRACE_SECTION;                            2  3T
GOTO NEXT3;                                       2  3T
(* Der Agent ist idle und unterbricht daher seinen Proz.*)
IDLE3:                                              2  3X
END_OF_SECTION;                                2  3X
EINSPRUNG:=11;HOLD(ELAPSED_TIME);ELAPSED_TIME:=0; 2  3X
GOTO MAIN_ENTRY;ENT11:                           2  3X
EINSPRUNG:=12;INT_WEIGHT:=AGENT_IDLE;            2  3X
REACTIVATE ASS_PROZESSOR;                      2  3X
ELAPSED_TIME:=0;                                2  3X
GOTO MAIN_ENTRY;ENT12:                           2  3X
NEW_TRACE_SECTION;                            2  3X
GOTO NEXT3;                                       2  3X
COMMENT *** ENDE DER FACETTE F2 ***;           2  3Z

COMMENT *** ANFANG DER FACETTE F1 ***;
FACET4:
CURRENT_OPER:-NOTEXT;                            2  4A
CURRENT_LINE:=12;                                2  4A
AKT_AS:-NEW AS4(AKT_AS,ENVIRONMENT);            2  4D
IF SNAPSHOT THEN OBSERVE_DUMP(SELF);            2  4I
COMMENT *** FACETTENSCHLEIFE VON F1 ***;         2  4L
NEXT4:
NEXT_MESSAGE;                                    2  4L
IF CURRENT_MSG==NONE THEN GOTO IDLE4;            2  4L
COMMENT *** AUFRUF DER OPERATION OP1 ***;        2  40
BOOL_PUSH(CURRENT_MSG.OPER="OP1");               2  40
BRANCH_FLAG:=-BOOL_TOP.WERT;BOOL_POP;           2  40
IF BRANCH_FLAG THEN GOTO NXT040111;             2  40

```

---

```

ENVIRONMENT:-AKT_AS;                                2  40
AKT_AS.RAS:=1;AKT_AS.RAS_LINE:=13;GOTO OPER111;RAS1: 2  40
IF SUCCESS THEN GOTO SUCC4;                         2  40
NXT040111:                                         2  40
GOTO NEXT4;                                         2  4T
SUCC4:                                              2  4T
CURRENT_OPER:-NOTEXT;                             2  4T
END_OF_SECTION;                                    2  4T
EINSPRUNG:=5;HOLD(ELAPSED_TIME);ELAPSED_TIME:=0;   2  4T
GOTO MAIN_ENTRY;ENT5:                            2  4T
NEW_TRACE_SECTION;                               2  4T
GOTO NEXT4;                                         2  4T
IDLE4:                                              2  4X
END_OF_SECTION;                                    2  4X
EINSPRUNG:=6;HOLD(ELAPSED_TIME);ELAPSED_TIME:=0;   2  4X
GOTO MAIN_ENTRY;ENT6:                            2  4X
EINSPRUNG:=7;INT_WEIGHT:=AGENT_IDLE;             2  4X
REACTIVATE ASS_PROZESSOR;                        2  4X
ELAPSED_TIME:=0;                                 2  4X
GOTO MAIN_ENTRY;ENT7:                            2  4X
NEW_TRACE_SECTION;                               2  4X
GOTO NEXT4;                                         2  4X
COMMENT *** ENDE DER FACETTE F1 ***;            2  4Z

COMMENT *** ANFANG DER OPERATION OP1 ***;          2  5A
OPER111:                                           2  5A
CURRENT_LINE:=15;                                2  5A
(* Returnverpflichtung aus der Nachricht entnehmen *)
CALLER:-CURRENT_MSG.SENDER;                      2  5A
WAITING:=CURRENT_MSG.REPLY_EXPECTED;             2  5A
WAITING_AGENT:-CURRENT_MSG.WAITING_AGENT;        2  5A
PORT:-CURRENT_MSG.PORT;                          2  5A
SUCCESS:=FALSE;                                  2  5A
(* Aktivierungssatz anlegen *)
AKT_AS:-NEW AS5(AKT_AS,ENVIRONMENT);           2  5D
(* pattern-match *)
CURRENT_PARM:-CURRENT_MSG.PARMS.FIRST;          2  5E
BOOL_PUSH(CURRENT_PARM==NONE);                   2  5E
BRANCH_FLAG:=BOOL_TOP.WERT;BOOL_POP;             2  5E
IF BRANCH_FLAG THEN GOTO ENDOP111;              2  5E
BOOL_PUSH(CURRENT_PARM.TYP="INT");               2  5E
BRANCH_FLAG:=~BOOL_TOP.WERT;BOOL_POP;            2  5E
IF BRANCH_FLAG THEN GOTO ENDOP111;              2  5E
INT_PUSH(CURRENT_PARM.INTWERT);                  2  5E
AKT_AS.E.E QUA AS1.VAR104:=INT_TOP.WERT;         2  5E
TRACE_INT("X");      (* X steht im script *)     2  5E
INT_POP;                                         2  5E
CURRENT_PARM:-CURRENT_PARM.SUC;                 2  5E
(* Initialisierung der lok. Var. MY_NAME *)
CURRENT_LINE:=16;                                2  5I
STRING_PUSH(COPY(
"OP1"
));
AKT_AS QUA AS5.VAR112:-STRING_TOP.WERT;          2  5I
TRACE_STRING("MY_NAME");                         2  5I
STRING_POP;                                       2  5I
IF SNAPSHOT THEN OBSERVE_DUMP(SELF);            2  5I

```

---

```

(* Nachricht aus der mailbox entfernen *)
SUCCESS:=TRUE;                                     2   5M
TRACE_OPER_CALL;CURRENT_LINE:=15;                  2   5M
CURRENT_MSG.OUT;CURRENT_MSG:=NONE;                 2   5M
(* Aufruf von PRINT_OPERATION *)
CURRENT_LINE:=18;                                  2   5S
ELAPSED_TIME:=ELAPSED_TIME+ 0.1000;                2   5S
STRING_PUSH(AKT_AS QUA AS5.VAR112); (* MY_NAME pushen *) 2   5S
INT_PUSH(AKT_AS.E.E QUA AS1.VAR104); (* X pushen *) 2   5S
TRACE_PROC("PRINT_OPERATION");
ENVIRONMENT:-AKT_AS.E.E; (*Prozedur liegt 2 Level hoher*) 2   5S
AKT_AS.RAS:=2;AKT_AS.RAS_LINE:=18;GOTO PROC2;RAS2:
CURRENT_LINE:=18;                                  2   5S
(* replace by F2(X) *)
CURRENT_LINE:=19;                                  2   5S
ELAPSED_TIME:=ELAPSED_TIME+ 0.1000;                2   5S
INT_PUSH(AKT_AS.E.E QUA AS1.VAR104); (* X pushen *) 2   5S
END_OF_SECTION; (* zunaechst Sim.zeit verbrauchen *) 2   5S
EINSPRUNG:=3;HOLD(ELAPSED_TIME);ELAPSED_TIME:=0;    2   5S
GOTO MAIN_ENTRY;ENT3:
NEW_TRACE_SECTION;
AKT_AS:-AKT_AS.E.E; (* OP1 und F1 verlassen *)      2   5S
ENVIRONMENT:-AKT_AS;(* F2 wird unter das script gehaengt*) 2   5S
TRACE_FACETTING("F2");CURRENT_MSG:=NONE;            2   5S
GOTO FACET3;
(* Verlassen der Operation *)
ENDOP111:
END_OF_SECTION;                                    2   5Z
EINSPRUNG:=4;HOLD(ELAPSED_TIME);ELAPSED_TIME:=0;    2   5Z
GOTO MAIN_ENTRY;ENT4:
NEW_TRACE_SECTION;
AKT_AS:-AKT_AS.LAST_AS;                           2   5Z
PASSOP111:
GOTO BRANCH(AKT_AS.RAS);
COMMENT *** HIER ENDET DIE OPERATION OP1 ***;      2   5Z

COMMENT *** ANFANG DER OPERATION OP2 ***;          2   6A
OPER113:
CURRENT_LINE:=27;                                  2   6A
CALLER:-CURRENT_MSG.SENDER;                      2   6A
WAITING:=CURRENT_MSG.REPLY_EXPECTED;             2   6A
WAITING_AGENT:-CURRENT_MSG.WAITING_AGENT;        2   6A
PORT:-CURRENT_MSG.PORT;                          2   6A
SUCCESS:=FALSE;                                   2   6A
AKT_AS:-NEW AS6(AKT_AS,ENVIRONMENT);            2   6D
CURRENT_PARM:-CURRENT_MSG.PARMS.FIRST;           2   6E
BOOL_PUSH(CURRENT_PARM==NONE);                   2   6E
BRANCH_FLAG:=BOOL_TOP.WERT;BOOL_POP;             2   6E
IF BRANCH_FLAG THEN GOTO ENDOP113;              2   6E
BOOL_PUSH(CURRENT_PARM.TYP="INT");               2   6E
BRANCH_FLAG:=-BOOL_TOP.WERT;BOOL_POP;            2   6E
IF BRANCH_FLAG THEN GOTO ENDOP113;              2   6E
INT_PUSH(CURRENT_PARM.INTWERT);                  2   6E
AKT_AS QUA AS6.VAR114:=INT_TOP.WERT;
TRACE_INT("I");
INT_POP;
CURRENT_PARM:-CURRENT_PARM.SUC;                  2   6E
INT_PUSH(AKT_AS QUA AS6.VAR114);                  2   6E

```

```

INT_PUSH(AKT_AS.E QUA AS3.VAR109); 2 6E
BOOL_PUSH(INT_TOP. NEXT.WERT> INT_TOP.WERT); 2 6E
INT_POP; 2 6E
INT_POP; 2 6E
BRANCH_FLAG:=-BOOL_TOP.WERT;BOOL_POP; 2 6E
IF BRANCH_FLAG THEN GOTO ENDOP113; 2 6E
IF SNAPSHOT THEN OBSERVE_DUMP(SELF); 2 6I
SUCCESS:=TRUE; 2 6M
TRACE_OPER_CALL;CURRENT_LINE:=27; 2 6M
CURRENT_MSG.OUT;CURRENT_MSG:-NONE; 2 6M
CURRENT_LINE:=28; 2 6S
ELAPSED_TIME:=ELAPSED_TIME+ 0.1000; 2 6S
STRING_PUSH(COPY(
"OP2"
));
INT_PUSH(AKT_AS QUA AS6.VAR114); 2 6S
TRACE_PROC("PRINT_OPERATION"); 2 6S
ENVIRONMENT:-AKT_AS.E.E; 2 6S
AKT_AS.RAS:=4;AKT_AS.RAS_LINE:=28;GOTO PROC2,RAS4: 2 6S
CURRENT_LINE:=28; 2 6S
CURRENT_LINE:=29; 2 6S
ELAPSED_TIME:=ELAPSED_TIME+ 0.1000; 2 6S
END_OF_SECTION; 2 6S
EINSPRUNG:=8;HOLD(ELAPSED_TIME);ELAPSED_TIME:=0; 2 6S
GOTO MAIN_ENTRY;ENT8:
NEW_TRACE_SECTION; 2 6S
ENVIRONMENT:-AKT_AS.E.E; 2 6S
WHILE AKT_AS.SCOPE_TYPE!="FACET" DO 2 6S
AKT_AS:-AKT_AS.LAST_AS; 2 6S
TRACE_FACETTING("F1");CURRENT_MSG:-NONE; 2 6S
GOTO FACET4; 2 6S
ENDOP113: 2 6Z
END_OF_SECTION; 2 6Z
EINSPRUNG:=9;HOLD(ELAPSED_TIME);ELAPSED_TIME:=0; 2 6Z
GOTO MAIN_ENTRY;ENT9:
NEW_TRACE_SECTION; 2 6Z
AKT_AS:-AKT_AS.LAST_AS; 2 6Z
PASSOP113: 2 6Z
GOTO BRANCH(AKT_AS.RAS); 2 6Z
COMMENT *** HIER ENDET DIE OPERATION OP2 ***; 2 6Z

START2: (* Betreten der initialen Facette *)
CURRENT_FACET:-COPY("F1"); 29999S
ENVIRONMENT:-AKT_AS; 29999S
CURRENT_MSG:-NONE; 29999S
AKT_AS.RAS:=5;AKT_AS.RAS_LINE:=35;GOTO FACET4;RAS5: 29999S
TERM2: (* Terminierung des Agenten *)
END_OF_SECTION; 29999S
EINSPRUNG:=13;HOLD(ELAPSED_TIME);ELAPSED_TIME:=0; 29999S
GOTO MAIN_ENTRY;ENT13: 29999S
EINSPRUNG:=14;INT_WEIGHT:=TERMINATION; 29999S
REACTIVATE ASS_PROZESSOR; 29999S
ELAPSED_TIME:=0; 29999S
GOTO MAIN_ENTRY;ENT14: 29999S
29999S
NEW_TRACE_SECTION; 29999S
END; 29999S
END SCRIPT2; 29999S

```

```

COMMENT *** ENDE DES SCRIPTS GENCODEDEMO ***;           29999S

%TITLE      GENCODE.GENERAL PART                      999   1
REF(MODULE) PROCEDURE NEW_MODULE(SCR_ID);INTEGER SCR_ID;999   1
BEGIN
IF SCR_ID=1 THEN NEW_MODULE:-NEW_SCRIPT1;             999   1
IF SCR_ID=2 THEN NEW_MODULE:-NEW_SCRIPT2;             999   1
END NEW_MODULE ;                                     999   1

TEXT ARRAY SCRIPT_NAME(1:2);                         999   1
INTEGER ARRAY NR_OF_AGENTS(1:2);                     999   1
INTEGER NR_OF_SCRIPTS;                             999   1

PROCEDURE DEFINE_SCRIPT_NAMES;                      999   1
BEGIN
SCRIPT_NAME(2):-COPY("GENCODEDEMO");                999   2
NR_OF_SCRIPTS:=2;                                  999   3
SCRIPT_NAME(1):-COPY("INTERFACE");                  999   3
END  DEFINE_SCRIPT_NAMES;                          999   3

```

Zunächst sei darauf hingewiesen, daß der Compiler den code nicht chronologisch in der Reihenfolge generiert, in der er hier gezeigt ist. Stattdessen **numeriert** der Compiler die generierten Sätze am rechten Rand, und vor dem Weiterübersetzen mit dem SIMULA-Compiler werden sie entsprechend sortiert (—> Entwicklungssystem).

Zunächst kommt zu jedem Block des Quellprogramms ein zugehöriger **Aktivierungssatz**, von dem beim Betreten des Blocks eine Instanz erzeugt wird. Alle haben die gemeinsame Oberklasse AS, die in der CLASS CSSA deklariert ist. Der Laufzeitkeller ist eine einfach-verkettete Liste von Aktivierungssätzen. In der CLASS AS sind Pointer für die Verkettung (LAST\_AS) und einen Verweis auf den statischen Vorgänger-Block deklariert (E wie environment).

Jeder Aktivierungssatz enthält die lokalen Variablen des Blocks (z.B. VAR104) und eine Prozedur (PRINT\_VARS), die alle lokalen Variablen ausgibt (für das DUMP-Kommando). Typ und Name des AS werden in der Oberklasse eingetragen.

Das script GENCODEDEMO wurde in die MODULE CLASS SCRIPT2 übersetzt. Die Verschachtelung der Blöcke wie im Quellprogramm ist hier aufgehoben: Jedem Block ist ein disjunkter Abschnitt des codes zugeordnet, der mit einem Label beginnt und mit einer GOTO-Anweisung endet.

Das script beginnt nach zwei Initialisierungen mit dem label MAIN\_ENTRY (2 1A). Wird ein Agent aktiviert, so beginnt er die Ausführung des script-codes stets bei diesem label. Im ACB ist die Adresse EINSPRUNG abgespeichert, mit der die Programmausführung fortgeführt werden soll. Mit Hilfe des SIMULA-switches ENTRY kann entsprechend der Einsprungadresse verzweigt werden (2 1C). Während der Ausführung des scripts sind stets die Attribute des zugehörigen Prozessors und des ACB durch 'INSPECT ASS\_PROZESSOR DO INSPECT ACTIVE\_ACB DO' zugreifbar (2 1A). Der switch BRANCH (2 1B) ermöglicht es, aus Prozeduren, Facetten, Operationen usw. an die Aufrufstelle zurückzukehren.

ren: Als Rücksprungadresse wird eine INTEGER-Zahl gespeichert und beim Rücksprung wird entsprechend ihres Wertes zu einem der label RAS1, RAS2, ... verzweigt.

Hinter 'ENT1:' (2 1C) folgt der code, der unmittelbar nach der Erzeugung des Agenten ausgeführt wird. Dort wird ein Aktivierungssatz für den äußersten Block des scripts (AS1) angelegt (2 1D). AKT\_AS zeigt jeweils auf die Kellerspitze. Beim Anlegen eines AS wird jeweils der dynamische Vorgänger, sowie der statisch übergeordnete AS übergeben, in diesem Fall gibt es allerdings weder einen dynamischen noch einen statischen Vorgänger, da scripts abgeschlossene scopes bilden.

Es wird dann an das Ende des scripts gesprungen (2 1Z), wo die initiale Facette betreten wird (29999S).

Der code für die Prozedur PRINT\_OPERATION beginnt mit dem label PROC2 (2 2A). Die Prozedur kann aufgerufen werden, indem der globale Zeiger ENVIRONMENT auf den statisch der Prozedur übergeordneten AS gesetzt wird und nach PROC2 gesprungen wird (vgl. z.B. (2 5S)). Dort wird dann ein Aktivierungssatz für die Prozedur angelegt (2 2D) und statischer und dynamischer Vorgänger werden eingetragen. Anschließend (2 2I) werden die aktuellen Parameter den stacks entnommen und den formalen Parametern zugewiesen. AKT\_AS erlaubt dabei den Zugriff auf den Aktivierungssatz mit den lokalen Variablen der Prozedur. Hinter dem Statementteil der Prozedur (2 2S) folgt der code für das Verlassen der Prozedur (2 2Z): Der aktuelle AS wird gelöscht und mit Hilfe des switches BRANCH wird in den aufrufenden Block zurückgekehrt. Im AS des rufenden Blocks ist abgespeichert (RAS), an welcher Stelle die Ausführung des Blocks fortgesetzt werden muß.

Im Statementteil werden die verschiedenen zu druckenden Werte in einen stack gepusht. Anschließend wird jeweils das oberste Stackelement ausgegeben und wieder aus dem Keller entfernt.

Bevor der erste string ausgegeben wird, muß allerdings zunächst Simulationszeit verbraucht werden und eine Unterbrechung durch den Prozessor ermöglicht werden. Nach jedem Statement (des Quellcodes) wird nämlich durch 'ELAPSED\_TIME:=ELAPSED\_TIME+0.1' die zu verbrauchende Simulationszeit erhöht. Wird nun ein Punkt erreicht, an dem eine nach außen sichtbare Aktion stattfinden soll, so wird zunächst die simulierte Zeit durch 'HOLD(ELAPSED\_TIME)' verbraucht. Da möglicherweise, noch bevor die simulierte Zeit vollkommen verbraucht ist, ein anderer Agent vom Betriebssystem aktiviert wird, muß vor dem HOLD die Einsprungadresse in den ACB weggemeldet werden. Nach dem HOLD wird sofort zum MAIN\_ENTRY gesprungen, wo die beiden INSPECT-Statements erneut ausgeführt werden. Das script wird dann mit der Einsprungadresse fortgesetzt, die im ACB des nun aktiven Agenten angegeben ist.

Es folgt nun der code der Facette F2 (!) deren forward-Deklaration ja noch vor F1 steht. Der Kopf der Facette ist genauso aufgebaut wie der Anfang der Prozedur PRINT\_OPERATION: Nach dem Anlegen des Aktivierungssatzes (2 3D) wird der Parameter übernommen (2 3I).

Es folgt dann eine Schleife, in der immer wieder versucht wird, eine ausführbare Nachricht zu finden. Zunächst wird die nächste Nachricht aus der mailbox entnommen (NEXT\_MESSAGE). Lieferst der Aufruf keine Nachricht, so ist der Agent im idle-Zustand (s.u.). Anschließend wird die Operation OP2 aufgerufen, sofern eine passende Nachricht vorliegt (2 30). Ist das nicht der Fall, so wird der Aufruf übersprungen und anschließend wiederum die nächste Nachricht aus der mailbox entnommen. Vor dem Aufruf wird der globale Zeiger ENVIRONMENT gesetzt, in diesem Fall auf den AS der Facette, da es sich um eine lokale Operation handelt. Außerdem wird in den AS der Facette die Rücksprungadresse eingetragen und hinter die Verzweigung zur Operation ein entsprechender label gesetzt. Für das DUMP-Kommando wird sogar die Rücksprungadresse als Zeile im Quellcode eingetragen.

Nach der Rückkehr aus der Operation muß unterschieden werden, ob der Aufruf erfolgreich war, oder ob der entry-match mißlungen ist. Die Operation setzt entsprechend das flag SUCCESS. Falls das flag eingeschaltet ist, muß nämlich zunächst Simulationszeit für die Operation verbraucht werden und eine Unterbrechung durch das Betriebssystem ermöglicht werden, bevor die nächste Nachricht entnommen wird (2 3T).

Im idle-Fall (2 3X) verbraucht der Agent ebenfalls zunächst Simulationszeit und unterbricht dann seinen Prozessor mit dem AGENT\_IDLE interrupt. Er wird erst dann wieder vom Scheduler aktiviert, wenn eine neue Nachricht für ihn eingetroffen ist.

Der code für die Facette F1 ist entsprechend organisiert und soll daher hier nicht mehr besprochen werden.

Am Anfang der Operation OP1 (2 5A) werden zunächst einige Angaben aus der Nachricht in den ACB übernommen: Es wird gespeichert, wer die Operation aufgerufen hat (CALLER), ob ein Agent auf Antwort wartet (WAITING), wer gegebenenfalls auf Antwort wartet (WAITING\_AGENT) und an welchen PORT die Antwort gehen soll. (Wegen der Möglichkeit, Antwortverpflichtungen zu vererben, muß nicht unbedingt der Sender selbst auf die Antwort warten!) Anschließend wird wiederum der Aktivierungssatz angelegt (2 5D). Es folgt dann der code für den pattern-match (2 5E): Der Zeiger CURRENT\_PARM wandert durch alle Parameter der Nachricht. Es wird jeweils überprüft, ob der Parameter den passenden Typ hat. Wenn ja, wird er an die im pattern angegebene Variable zugewiesen, wenn nein, wird die Operation beendet, ohne daß das flag SUCCESS angeschaltet wird.

Nach dem pattern-match wird die lokale Variable MY\_NAME mit "OP1" initialisiert (2 5I). Vor dem Statementteil der Operation wird dann die Nachricht aus der mailbox entfernt (2 5M).

Vor dem Prozeduraufruf werden die aktuellen Parameter in die stacks geschrieben (MY\_NAME=VAR112, X=VAR104). Der ENVIRONMENT-pointer wird aus den AS für das script gesetzt. Die Rücksprungadresse wird im aktuellen AS gespeichert und nach der Verzweigung ein entsprechender label gesetzt.

Vor der Facettierung wird der Parameter X gepusht. Es wird dann zunächst Simulationszeit verbraucht, da eine Facettierung ja auch für andere Agenten relevant ist. Die Aktivierungssätze für die Operation OP1 und die Facette F1 werden entfernt und der ENVIRONMENT-pointer auf die neue Kellerspitze gesetzt. Schließlich muß noch der mailbox-pointer CURRENT\_MSG zurückgesetzt werden, da nach einer Facettierung wieder alle Nachrichten in Frage kommen. Nach dem Sprung zur neuen Facette ist keine Rückkehr an die Aufrufstelle möglich, daher muß keine Rücksprungadresse eingetragen werden (der AS für die Operation ist ja auch bereits vernichtet).

Am Ende der Operation (2 5Z) wird noch einmal Simulationszeit verbraucht, bevor in die rufende Facette zurückgekehrt wird.

Die Operation OP2 soll hier nicht gesondert erläutert werden, sie unterscheidet sich kaum von der Operation OP1. Es sei aber noch auf den code am Ende des scripts hingewiesen: Zu dem label TERM2 (29999S) wird gesprungen, wenn der Agent ein TERMINATE-Statement ausführt. Der Agent unterbricht dann seinen Prozessor mit dem TERMINATION-interrupt, was dazu führt, daß der ACB (nicht der script-code!) aus der Schlange der rechenwilligen Agenten entfernt wird.

Nach dem SCRIPT2 folgen schließlich noch einige Deklarationen, die zum Laufzeitsystem gehören, aber erst vom Compiler generiert werden können, wenn bekannt ist, welche scripts im CSSA-Quellprogramm definiert sind. Die Prozedur NEW\_MODULE generiert zu einer script-Nummer einen neuen Agenten des angegebenen Typs. Das array SCRIPT\_NAME wird in der passenden Größe angelegt und später mit den Namen der übersetzten scripts gefüllt, so daß jederzeit zu einer script-Nummer der im Quellprogramm verwendete Bezeichner ausgegeben werden kann. Im array NR\_OF\_AGENTS wird zu jedem script die Anzahl der existierenden Agenten festgehalten. Die Prozedur DEFINE\_SCRIPT\_NAMES wird in der Initialisierungsphase des interfaces aufgerufen und füllt das array SCRIPT\_NAME mit den symbolischen Namen der scripts.

### 3.5 Das Hauptprogramm

Im Hauptprogramm werden nach einigen Initialisierungen so viele Prozessoren und Busse gegründet, wie in der Spezifikation der hardware-Konfiguration angegeben wurde. Die Geräte werden durch ACTIVATE gestartet, gehen aber sofort in den idle-Zustand (da keine arbeitswilligen Agenten bzw. zu verschickende Nachrichten vorliegen), wo sie auf einen externen interrupt warten.

Anschließend wird der interface-Agent gegründet und auf Prozessor 1 gelegt, indem ein entsprechendes script-Modul und ein ACB erzeugt werden und in die Warteschlangen von Prozessor 1 eingetragen werden. Der interface-Agent wird auch als erster in die globale Liste AGENT\_LIST eingetragen, in der Verweise auf alle existierenden Agenten (ACB's) abgespeichert sind.

Gestartet wird die Simulation schließlich, indem Prozessor 1 durch einen externen interrupt (durch ACTIVATE) aufgeweckt wird, und das Hauptprogramm durch HOLD(1000000) die Kontrolle abgibt. Die Simulation wird beendet, wenn die 1000000 Sekunden Simulationszeit verbraucht sind (was nicht vorgesehen ist), oder (was der Normalfall sein soll) wenn durch den TERMINATE-Befehl das Hauptprogramm mit REACTIVATE MAIN wieder die Kontrolle erhält.



## COMPILER OPTIONS

```
NOLIST
NODECK
LOAD
WARN
SUBCHK
XREF
SOURCE
NOTERM

LINECNT =      88
MAXERROR=    50
INDENT =       0
RESWD =       3
EXTERN =       0
SIZE = (2048,2048,65536,0,1003520,2048)
SYMBDUMP=     3
```



```
0025  Simulation class CSSA;
0026  virtual : ref(MODULE) procedure NEW_MODULE;
0027      text array SCRIPT_NAME;
0028      integer array NR_OF_AGENTS;
0029  begin
0030
0031  comment EXTERNAL ASSEMBLY PROCEDURE CLEAR ; comment siemens;
0032  procedure Clear; begin comment SIEMENS;Sysout.Outchar(Char(#0C));
0033          Sysout.Outimage;
0034          end;
0035  procedure CLEAR_SCREEN;
0036  Clear;
0037
0038  Process class PROCESSOR(ID);integer ID;
0039  begin
0040
```

```

0042
0043  comment*****;
0044  comment TRACING FACILITIES ;
0045  comment*****;
0046
0047  comment ****;
0048  ** NAMEN EINES PROZESSORS AUSGEBEN **
0049  *****;
0050 procedure PRINT_ID;
0051 begin Outchar('P'); PRINT_INT(ID);Outtext(": ");end;                                B7 E7
0052
0053 external assembly procedure WRTRD; comment SIEMENS;
0054
0055 comment ****;
0056 ** AUSGEBEN DER STATUS TABELLE **
0057 *****;
0058 procedure STATUS(SENDING);boolean SENDING;
0059 begin ref(AGENT) A;
0060   if SINGLE_STEP then
0061     begin text REPLY ; REPLY:=-Blanks(140);
0062       WRTRD("PRESS ENTER TO CONTINUE --",REPLY);
0063     end;
0064   CLEAR_SCREEN;
0065   Outtext("++");PRINT_TIME;
0066   Outtext(" ALL EXISTING AGENTS:");OUTLINE;
0067   BMSOUT.Setpos(2);while BMSOUT.Pos<78 do Outchar('_');OUTLINE;
0068   Outtext("| ");
0069   Outtext(" AGENT ");
0070   BMSOUT.Setpos(20);Outtext(" | ");
0071   Outtext(" FACET");
0072   BMSOUT.Setpos(32);Outtext(" | ");
0073   Outtext("OPERATION");
0074   BMSOUT.Setpos(44);Outtext(" | ");
0075   Outtext("      MAILBOX      ");
0076   BMSOUT.Setpos(77);Outtext(" | ");OUTLINE;
0077   Outtext(" | ");while BMSOUT.Pos<78 do Outchar('_');Outtext(" | ");
0078   OUTLINE;
0079   A:=-AGENT_LIST.First;
0080   while A=/=none do
0081     begin if ~A.CB.STOPPED
0082       then PRINT_STATUS(A.CB,A.CB==ACTIVE_ACB and SENDING);          B10
0083       A:=-A.Suc;
0084     end;
0085   Outtext(" | ");while BMSOUT.Pos<78 do Outchar('_');Outtext(" | ");
0086   OUTLINE; OUTLINE;
0087 end;                                         E8
0088
0089 comment ****;
0090 ** JEDESMAL, WENN EIN AGENT VON SEINEM PROZESSOR AKTIVIERT WIRD, BE **
0091 ** GINNT EINE NEUE SECTION DES TRACING, DIE IM PROTOKOLL           **
0092 ** MARKIERT WIRD                                              **
0093 *****;
0094 procedure NEW_TRACE_SECTION;
0095   if ACTIVE_ACB.TRACE then
0096     begin OUTLINE; Outtext("___");PRINT_TIME;Outtext(" TRACING ");
0097       PRINT_AGENT(ACTIVE_ACB);Outtext(" / ");
0098       Outtext(ACTIVE_ACB.CURRENT_FACET);
0099       if ACTIVE_ACB.CURRENT_OPER=/=notext
0100         then begin Outtext(" / ");
0101           Outtext(ACTIVE_ACB.CURRENT_OPER);
0102         end;
0103       Outtext(" ");
0104       while BMSOUT.Pos<79 do Outchar('_'); OUTLINE;
0105     end;                                         E11
0106
0107 procedure END_OF_SECTION;
0108   if ACTIVE_ACB.TRACE then
0109     begin if BMSOUT.Pos>1 then OUTLINE;
0110       while BMSOUT.Pos<79 do Outchar('_'); OUTLINE; OUTLINE;          B13
0111     end;                                         E13
0112
0113 comment ****;
0114 ** ZU JEDEMEREIGNIS, DAS BEIM TRACING GEMELDET WIRD, GIBT ES EINE **
0115 ** ENTSPRECHENDE PROZEDUR, DIE DASEREIGNIS MELDET.                 **
0116 *****;
0117
0118 procedure TRACE_FACETTING(F);value F ; text F;
0119 inspect ACTIVE_ACB do
0120 begin if OBSERVE or TRACE then
0121   begin if TRACE
0122     then PRINT_LINENO
0123     else
0124       begin Outtext("++");PRINT_TIME;Outtext(" ");
0125         PRINT_ID;
0126         PRINT_AGENT(ACTIVE_ACB);Outtext(" PERFORMS ");
0127       end;                                         B14
0128                                         B15
0129                                         B16
0130                                         E16

```

```

0128          Outtext("FACETTING : ");
0129          Outtext(CURRENT_FACET);Outtext(" --> ");
0130          CURRENT_FACET:-F;
0131          Outtext(CURRENT_FACET);OUTLINE;
0132      end;
0133      else begin CURRENT_FACET:-F; CURRENT_OPER:-notext; end;
0134  end;
0135
0136 procedure TRACE_SEND;
0137 inspect ACTIVE_ACB do
0138 begin if OBSERVE or TRACE then
0139     begin if TRACE
0140         then begin NEW_TRACE_SECTION;
0141             PRINT_LINENO;
0142             Outtext("SEND ");
0143         end
0144     else
0145         begin Outtext("+++" );PRINT_TIME;Outtext("   ");
0146             PRINT_ID;
0147             PRINT_AGENT(ACTIVE_ACB);Outtext(" SENDS ");
0148         end;
0149         MSG_TOP.WERT.PRINT(true);
0150         Outtext(" TO ");
0151         PRINT_AGENT(MSG_TOP.WERT.RECEIVER);
0152         OUTLINE;
0153         END_OF_SECTION;
0154     end;
0155 end;
0156
0157 procedure TRACE_CREATION(A);ref(ACB) A;
0158 inspect ACTIVE_ACB do
0159 begin if OBSERVE or TRACE then
0160     begin if TRACE
0161         then begin NEW_TRACE_SECTION;
0162             PRINT_LINENO;
0163             Outtext("CREATE ");
0164         end
0165     else
0166         begin Outtext("+++" );PRINT_TIME;Outtext("   ");
0167             PRINT_ID;
0168             PRINT_AGENT(ACTIVE_ACB);Outtext(" CREATES ");
0169         end;
0170         PRINT_AGENT(A);
0171         MSG_TOP.WERT.PRINT(false);
0172         OUTLINE;
0173         END_OF_SECTION;
0174     end;
0175 end;
0176
0177 procedure TRACE_TERMINATION;
0178 inspect ACTIVE_ACB do
0179 begin if OBSERVE or TRACE then
0180     begin if TRACE
0181         then begin NEW_TRACE_SECTION;
0182             PRINT_LINENO;
0183             Outtext("TERMINATION ");
0184         end
0185     else
0186         begin Outtext("+++" );PRINT_TIME;Outtext("   ");
0187             PRINT_ID;
0188             PRINT_AGENT(ACTIVE_ACB);Outtext(" TERMINATED");
0189         end;
0190         OUTLINE;
0191         END_OF_SECTION;
0192     end;
0193 end;
0194
0195 procedure TRACE_SIGNAL(S); value S; text S;
0196 if ACTIVE_ACB.TRACE then
0197     begin PRINT_LINENO;
0198         Outtext("SIGNAL ");
0199         Outtext(S);OUTLINE;
0200     end;
0201
0202 procedure TRACE_CONTINUE(L); value L; text L;
0203 if ACTIVE_ACB.TRACE then
0204     begin PRINT_LINENO;
0205         Outtext("CONTINUE ");
0206         Outtext(if L=notext then "CURRENT LOOP" else L);OUTLINE;
0207     end;
0208
0209 procedure TRACE_ESCAPE(L); value L; text L;
0210 if ACTIVE_ACB.TRACE then
0211     begin PRINT_LINENO;
0212         Outtext("ESCAPE ");
0213         Outtext(if L=notext then "CURRENT LOOP" else L);OUTLINE;

```

```

0214     end;                                         E32
0215
0216 procedure TRACE_PROC(P); value P; text P;
0217   if ACTIVE_ACB.TRACE then
0218     begin PRINT_LINENO;
0219       Outtext("CALLING PROCEDURE ");
0220       Outtext(P);OUTLINE;
0221     end;                                         E33
0222
0223 procedure TRACE_FUNCT(F); value F; text F;
0224   if ACTIVE_ACB.TRACE then
0225     begin PRINT_LINENO;
0226       Outtext("CALLING FUNCTION ");
0227       Outtext(F);OUTLINE;
0228     end;                                         E34
0229
0230 procedure TRACE_OPER_CALL;
0231 inspect ACTIVE_ACB do
0232   begin if OBSERVE or TRACE then
0233     begin if TRACE then BMSOUT.Setpos(16)
0234       else
0235         begin Outtext("+++" );PRINT_TIME;Outtext("  ");
0236           PRINT_ID;
0237           PRINT_AGENT(ACTIVE_ACB);Outchar(' ');
0238         end;                                         E37
0239         Outtext("STARTING OPERATION ");
0240         inspect CURRENT_MSG do PRINT(false)
0241           otherwise Outtext("IDLE");
0242           OUTLINE;
0243         end;                                         E36
0244         inspect CURRENT_MSG do
0245           CURRENT_OPER:-OPER otherwise CURRENT_OPER:-Copy("IDLE");
0246       end;                                         E35
0247
0248 procedure TRACE_ARRAY(IDNAME);value IDNAME;text IDNAME;
0249   if ACTIVE_ACB.TRACE then
0250     begin PRINT_LINENO;Outtext(IDNAME);                                         B38
0251   end;                                         E38
0252
0253 procedure TRACE_RECORD(IDNAME);value IDNAME;text IDNAME;
0254   if ACTIVE_ACB.TRACE then
0255     begin PRINT_LINENO;Outtext(IDNAME);                                         B39
0256   end;                                         E39
0257
0258 procedure TRACE_REAL(IDNAME);value IDNAME;text IDNAME;
0259   if ACTIVE_ACB.TRACE then
0260     begin PRINT_LINENO;Outtext(IDNAME);Outtext(" := ");
0261       PRINT_REAL(ACTIVE_ACB.REAL_TOP.WERT);
0262       OUTLINE;                                         B40
0263   end;                                         E40
0264
0265 procedure TRACE_INT(IDNAME);value IDNAME;text IDNAME;
0266   if ACTIVE_ACB.TRACE then
0267     begin PRINT_LINENO;Outtext(IDNAME);Outtext(" := ");
0268       PRINT_INT(ACTIVE_ACB.INT_TOP.WERT);
0269       OUTLINE;                                         B41
0270   end;                                         E41
0271
0272 procedure TRACE_ENUM(IDNAME);value IDNAME;text IDNAME;
0273   if ACTIVE_ACB.TRACE then
0274     begin PRINT_LINENO;Outtext(IDNAME);Outtext(" := ");
0275       PRINT_ENUM(ACTIVE_ACB.ENUM_TOP.WERT);
0276       OUTLINE;                                         B42
0277   end;                                         E42
0278
0279 procedure TRACE_BOOL(IDNAME);value IDNAME;text IDNAME;
0280   if ACTIVE_ACB.TRACE then
0281     begin PRINT_LINENO;Outtext(IDNAME);Outtext(" := ");
0282       PRINT_BOOL(ACTIVE_ACB.BOOL_TOP.WERT);
0283       OUTLINE;                                         B43
0284   end;                                         E43
0285
0286 procedure TRACE_STRING(IDNAME);value IDNAME;text IDNAME;
0287   if ACTIVE_ACB.TRACE then
0288     begin PRINT_LINENO;Outtext(IDNAME);Outtext(" := ");
0289       PRINT_STRING(ACTIVE_ACB.STRING_TOP.WERT);
0290       OUTLINE;                                         B44
0291   end;                                         E44
0292
0293 procedure TRACE_AGENT(IDNAME);value IDNAME;text IDNAME;
0294   if ACTIVE_ACB.TRACE then
0295     begin PRINT_LINENO;Outtext(IDNAME);Outtext(" := ");
0296       PRINT_AGENT(ACTIVE_ACB.AGENT_TOP.WERT);
0297       OUTLINE;                                         B45
0298   end;                                         E45
0299

```

```
0300  procedure TRACE_SCRIPT(IDNAME);value IDNAME;text IDNAME;
0301    if ACTIVE_ACBTRACE then
0302    begin PRINT_LINENO;Outtext(IDNAME);Outtext(" := ");
0303      PRINT_SCRIPT(ACTIVE_ACB.SCRIPT_TOP.WERT);
0304      OUTLINE;
0305    end;                                              E46
0306
0307  procedure TRACE_OPER(IDNAME);value IDNAME;text IDNAME;
0308    if ACTIVE_ACBTRACE then
0309    begin PRINT_LINENO;Outtext(IDNAME);Outtext(" := ");
0310      PRINT_OPER(ACTIVE_ACB.OPER_TOP.WERT);
0311      OUTLINE;
0312    end;                                              E47
0313
0314  procedure TRACE_NEXT_MSG ;
0315  inspect ACTIVE_ACB do
0316  begin
0317    BMSOUT.Setpos(16);
0318    Outtext("SEARCHING NEXT_MESSAGE");
0319    if CURRENT_MSG==none
0320      then Outtext(" - NO MESSAGE FOUND")
0321    else begin OUTLINE; BMSOUT.Setpos(16);
0322      Outtext("FOUND : ");
0323      CURRENT_MSG.PRINT(true);
0324      end;
0325    OUTLINE;
0326  end;                                              E48
0327
0328  comment ****
0329  ** EIN AUFRUF VON NEW_STATEMENT MIT DER QUELLPROGRAMMZEILE WIRD    **
0330  ** VOM COMPILER NACH JEDEM CSSA-STATEMENT GENERIERT.                **
0331  ****
0332
0333  procedure NEW_STATEMENT(Line);integer Line;
0334  inspect ACTIVE_ACB do
0335  begin if Line==CURRENT_LINE
0336    then begin CURRENT_LINE:=Line;
0337      if TRACE then PRINT_LINENO;
0338      end;
0339  end NEW_STATEMENT;                                E50
0340
0341  procedure PRINT_LINENO;
0342  begin BMSOUT.Setpos(16);
0343    Outtext("LINE");
0344    Outint(ACTIVE_ACB.CURRENT_LINE,4);
0345    Outtext(": ");
0346  end;                                              E52
0347
```

```

0349
0350 comment ****
0351 ** AUSWAHL DER NAECHSTEN NACHRICHT AUS DER MAILBOX **
0352 ****
0353
0354 procedure NEXT_MESSAGE;
0355 inspect ACTIVE_ACB do
0356 begin
0357   comment : FETCHES THE NEXT MESSAGE FROM THE MAILBOX;
0358   CURRENT_MSG :=-
0359   if CURRENT_MSG ==none
0360     then MAILBOX.First
0361   else CURRENT_MSG.Suc ;
0362   if TRACE then TRACE_NEXT_MSG ;
0363 end NEXTMESSAGE;
0364
0365 comment ****
0366 ** OPERATIONEN DER STACK-MASCHINE **
0367 ****
0368
0369 procedure BOOL_PUSH(B);boolean B;
0370 inspect ACTIVE_ACB do
0371 begin ref(BOOLEL) HILF;
0372   HILF:-new BOOLEL(B);
0373   HILF.NEXT:=BOOL_TOP;
0374   BOOL_TOP:=HILF;
0375 end;
0376 procedure BOOL_POP;
0377 inspect ACTIVE_ACB do
0378   BOOL_TOP:=BOOL_TOP.NEXT;
0379
0380 procedure ENUM_PUSH(I);integer I; INT_PUSH(I);
0381 procedure ENUM_POP;INT_POP;
0382
0383 procedure REAL_PUSH(R); long real R ;
0384 inspect ACTIVE_ACB do
0385 begin ref(REALEL) HILF;
0386   HILF:-new REALEL(R);
0387   HILF.NEXT:=REAL_TOP;
0388   REAL_TOP:=HILF;
0389 end;
0390 procedure REAL_POP;
0391 inspect ACTIVE_ACB do
0392   REAL_TOP:=REAL_TOP.NEXT;
0393
0394 procedure INT_PUSH(I);integer I;
0395 inspect ACTIVE_ACB do
0396 begin ref(INTEL) HILF;
0397   HILF:-new INTEL(I);
0398   HILF.NEXT:=INT_TOP;
0399   ENUM_TOP:=INT_TOP:HILF;
0400 end;
0401 procedure INT_POP;
0402 inspect ACTIVE_ACB do
0403   ENUM_TOP:=INT_TOP:-INT_TOP.NEXT;
0404
0405 procedure SCRIPT_PUSH(S);integer S;
0406 inspect ACTIVE_ACB do
0407 begin ref(SCRIPTEL) HILF;
0408   HILF:-new SCRIPTEL(S);
0409   HILF.NEXT:=SCRIPT_TOP;
0410   SCRIPT_TOP:=HILF;
0411 end;
0412 procedure SCRIPT_POP;
0413 inspect ACTIVE_ACB do
0414   SCRIPT_TOP:=SCRIPT_TOP.NEXT;
0415
0416 procedure AGENT_PUSH(A);ref(ACB) A;
0417 inspect ACTIVE_ACB do
0418 begin ref(AGENTEL) HILF;
0419   HILF:-new AGENTEL(A);
0420   HILF.NEXT:=AGENT_TOP;
0421   AGENT_TOP:=HILF;
0422 end;
0423 procedure AGENT_POP;
0424 inspect ACTIVE_ACB do
0425   AGENT_TOP:=AGENT_TOP.NEXT;
0426
0427 procedure STRING_PUSH(S);text S;
0428 inspect ACTIVE_ACB do
0429 begin ref(STRINGEL) HILF;
0430   HILF:-new STRINGEL(S);
0431   HILF.NEXT:=STRING_TOP;
0432   STRING_TOP:=HILF;
0433 end;
0434 procedure STRING_POP;

```

```

0435      inspect ACTIVE_ACB do
0436          STRING_TOP:=-STRING_TOP.NEXT;
0437
0438      procedure OPER_PUSH(O);text O;
0439          inspect ACTIVE_ACB do
0440              begin ref(OPEREL) HILF;
0441                  HILF:=-new OPEREL(O);
0442                  HILF.NEXT:=-OPER_TOP;
0443                  OPER_TOP:=-HILF;
0444          end;
0445      procedure OPER_POP;
0446          inspect ACTIVE_ACB do
0447              OPER_TOP:=-OPER_TOP.NEXT;
0448
0449
0450      procedure MSG_PUSH(M);ref(MSG) M;
0451          inspect ACTIVE_ACB do
0452              begin ref(MSGEL) HILF;
0453                  HILF:=-new MSGEL(M);
0454                  HILF.NEXT:=-MSG_TOP;
0455                  MSG_TOP:=-HILF;
0456          end;
0457      procedure MSG_POP;
0458          inspect ACTIVE_ACB do
0459              MSG_TOP:=-MSG_TOP.NEXT;
0460
0461
0462      procedure LOG_AND;
0463          inspect ACTIVE_ACB do
0464              begin boolean B1,B2;
0465                  B1:=BOOL_TOP.WERT;BOOL_POP;
0466                  B2:=BOOL_TOP.WERT;BOOL_POP;
0467                  BOOL_PUSH(B1 and B2);
0468          end;
0469
0470      procedure LOG_OR;
0471          inspect ACTIVE_ACB do
0472              begin boolean B1,B2;
0473                  B1:=BOOL_TOP.WERT;BOOL_POP;
0474                  B2:=BOOL_TOP.WERT;BOOL_POP;
0475                  BOOL_PUSH(B1 or B2);
0476          end;
0477
0478      procedure LOG_NOT;
0479          inspect ACTIVE_ACB do
0480              begin BOOL_TOP.WERT:=-not BOOL_TOP.WERT;
0481          end;
0482
0483      procedure ADD;
0484          inspect ACTIVE_ACB do
0485              begin integer I1,I2;
0486                  I1:=INT_TOP.WERT;INT_POP;
0487                  I2:=INT_TOP.WERT;INT_POP;
0488                  INT_PUSH(I1 + I2);
0489          end;
0490
0491      procedure REALADD;
0492          inspect ACTIVE_ACB do
0493              begin long real R1,R2;
0494                  R1:=REAL_TOP.WERT;REAL_POP;
0495                  R2:=REAL_TOP.WERT;REAL_POP;
0496                  REAL_PUSH(R1 + R2);
0497          end;
0498
0499      procedure CONCAT;
0500          inspect ACTIVE_ACB do
0501              begin text S1,S2,S3;
0502                  S2:=-STRING_TOP.WERT;STRING_POP;
0503                  S1:=-STRING_TOP.WERT;STRING_POP;
0504                  S3:=-Blanks(S1.Length+S2.Length);
0505                  S3.Sub(1,S1.Length):=S1;
0506                  S3.Sub(S1.Length+1,S2.Length):=S2;
0507                  STRING_PUSH(S3);
0508          end;
0509
0510      procedure MULT;
0511          inspect ACTIVE_ACB do
0512              begin integer I1,I2;
0513                  I1:=INT_TOP.WERT;INT_POP;
0514                  I2:=INT_TOP.WERT;INT_POP;
0515                  INT_PUSH(I1 * I2);
0516          end;
0517
0518      procedure DIV;
0519          inspect ACTIVE_ACB do
0520              begin integer I1,I2;

```

```
0521      I1:=INT_TOP.WERT; INT_POP;
0522      if I1=0
0523      then RUNTIME_ERROR("ZERODIVIDE. DIVISION IGNORED")
0524      else begin I2:=INT_TOP.WERT; INT_POP;
0525          INT_PUSH(I2 //I1);
0526          end;
0527      end;
0528
0529 procedure POT;
0530     inspect ACTIVE_ACB do
0531     begin integer I1,I2;
0532         I1:=INT_TOP.WERT; INT_POP;
0533         I2:=INT_TOP.WERT; INT_POP;
0534         INT_PUSH(I2** I1);
0535     end;
0536
0537 procedure MINUS;
0538     inspect ACTIVE_ACB do
0539     begin integer I;
0540         I:=INT_TOP.WERT; INT_POP;
0541         INT_PUSH( -I);
0542     end;
0543
0544 procedure REALMULT;
0545     inspect ACTIVE_ACB do
0546     begin long real R1,R2;
0547         R1:=REAL_TOP.WERT;REAL_POP;
0548         R2:=REAL_TOP.WERT;REAL_POP;
0549         REAL_PUSH(R1 * R2);
0550     end;
0551
0552 procedure REALDIV;
0553     inspect ACTIVE_ACB do
0554     begin long real R1,R2;
0555         R1:=REAL_TOP.WERT; REAL_POP;
0556         if R1 = 0.0 then RUNTIME_ERROR("ZERODIVIDE. DIVISION IGNORED")
0557         else begin
0558             R2:=REAL_TOP.WERT; REAL_POP;
0559             REAL_PUSH(R2/R1);
0560             end;
0561         end;
0562
0563 procedure REALPOT ;
0564     inspect ACTIVE_ACB do
0565     begin long real R1,R2;
0566         R1:=REAL_TOP.WERT; REAL_POP;
0567         R2:=REAL_TOP.WERT; REAL_POP;
0568         REAL_PUSH(R2** R1);
0569     end;
0570
0571 procedure REALMINUS;
0572     inspect ACTIVE_ACB do
0573     begin long real R;
0574         R:=REAL_TOP.WERT; REAL_POP;
0575         REAL_PUSH( -R);
0576     end;
0577
```

```

0579
0580   comment ****
0581   ** DAS UNTERBRECHUNGSGESTEUERTE, VERTEILTE MULTIPROGRAMMING      **
0582   ** BETRIEBSSYSTEM, DAS AUF JEDEM PROZESSOR LAEUFT                  **
0583   ****;
0584
0585   ref(ACB) ACTIVE_ACB;
0586   ref(MODULE) ACTIVE_CODE;
0587   ref(Head) ACB_QUEUE;
0588   ref(Head) IDLE_QUEUE;
0589   ref(Head) MODULE_LIST;
0590   ref(Head) MESSAGE_LIST;
0591   integer INT_WEIGHT;
0592
0593   switch INTERRUPT:= LTIMESLICE_RUNOUT,
0594           LSEND_MESSAGE,
0595           LRECEIVE_MESSAGE,
0596           LCREATE_AGENT,
0597           LTERMINATION,
0598           LAGENT_IDLE;
0599
0600   procedure INITIALIZE;                                         B78
0601   begin
0602     MESSAGE_LIST:=new Head;
0603     ACB_QUEUE:=new Head;
0604     IDLE_QUEUE:=new Head;
0605     MODULE_LIST:=new Head;
0606   end;                                                       E78
0607
0608   real ACTIVE_TIME, LAST_ACTIVATION;
0609
0610   integer procedure UTILIZATION;                                 B79
0611   if Time>0 then
0612   begin
0613     if Idle
0614       then UTILIZATION:=ACTIVE_TIME/Time*100
0615       else UTILIZATION:=(ACTIVE_TIME+(Time-LAST_ACTIVATION))/Time*100;
0616   end;                                                       E79
0617
0618   procedure RUNTIME_ERROR(T); value T; text T;
0619   begin OUTLINE;Outtext("XXX");PRINT_TIME;Outtext(" ");
0620     PRINT_AGENT(ACTIVE_ACB);
0621     Outtext(": "); Outtext(T);OUTLINE;
0622     BMSOUT.Setpos(16);Outtext(" AT LINE ");
0623     Outint(ACTIVE_ACB.CURRENT_LINE,4);OUTLINE;
0624     if ACTIVE_ACB/=INTERFACE then PRINT_DUMP(ACTIVE_ACB);
0625   end;                                                       E80
0626   comment
0627     THE ACTIVE_ACB (IF ANY) IS SET TO THE END OF THE ACB_QUEUE.
0628     THE NEW ACTIVE_ACB IS SELECTED FROM THE ACB_QUEUE. THIS SIMPLE
0629     SCHEDULER ALWAYS ACTIVATES THE FIRST ACB IN THE ACB_QUEUE.
0630
0631 SCHEDULER: if INT_WEIGHT=0 then goto INTERRUPT(INT_WEIGHT);
0632   if ~MESSAGE_LIST.Empty then goto LRECEIVE_MESSAGE;
0633   inspect ACTIVE_ACB do
0634   begin
0635     Out;if ~STOPPED then Into(ACB_QUEUE);
0636     if SYSTEM then
0637       begin
0638         Outtext("///");PRINT_TIME;Outtext(" ");
0639         PRINT_ID;
0640         PRINT_AGENT(ACTIVE_ACB);Outtext(" SUSPENDED");
0641         OUTLINE;
0642       end;
0643     end;
0644     ACTIVE_ACB:=ACB_QUEUE.First;
0645     if ACTIVE_ACB==none then goto LIDLE;
0646     ACTIVE_CODE:=ACTIVE_ACB.ASS_SCRIPT;
0647   comment
0648     THE SELECTED ACB IS ACTIVATED.
0649     IN ADDITION A TIMER IS SET SO THAT THE PROCESSOR IS ACTIVATED
0650     AFTER AT LEAST ONE TIME_SLICE EVEN IF NO INTERRUPT OCCURS
0651
0652 AGENT_INITIATOR: activate ACTIVE_CODE delay ACTIVE_ACB.LEFT_TIME;
0653   ACTIVE_ACB.LEFT_TIME:=0;
0654   INT_WEIGHT:=TIMESLICE_RUNOUT;
0655   if SYSTEM then
0656     begin
0657       Outtext("///");PRINT_TIME;Outtext(" ");
0658       PRINT_ID;
0659       PRINT_AGENT(ACTIVE_ACB);Outtext(" ACTIVATED");OUTLINE;
0660     end;
0661     Hold(TIMESLICE);
0662   comment
0663     NOW ONE OF THE AGENTS IS ACTIVE UNTIL HIS NEXT INTERRUPT OR
0664     TIMESLICE_RUNOUT.

```

```

0665 | _____;
0666 ACTIVE_ACB.LEFT_TIME:=ACTIVE_CODE.Evttime-Time;
0667 Cancel(ACTIVE_CODE);
0668
0669 comment _____
0670 | THE PROCESSOR NOW REACT ACCORDING TO THE INTERRUPT WEIGHT.
0671 | AFTERWARDS THE SCHEDULER WILL SELECT THE NEXT ACTIVE_ACB.
0672 |
0673 goto INTERRUPT(INT_WEIGHT);
0674 Outtext("INTERNAL ERROR: INT_WEIGHT="); Outint(INT_WEIGHT,4);OUTLINE;
0675 comment _____
0676 | THE ACTIVE_ACB'S MESSAGE IS SENT OFF TO ANOTHER PROCESSOR
0677 |
0678 LSEND_MESSAGE:
0679 if EVENT then STATUS(true);
0680 TRACE_SEND;
0681 INT_WEIGHT:=0;
0682 begin ref(PROCESSOR) TARGET; B84
0683   ref(MSG) M; ref(BUS) B;
0684   M:-ACTIVE_ACB.MSG_TOP.WERT;
0685   if M.RECEIVER==none then
0686     begin RUNTIME_ERROR("SEND TO <NOAGENT> ISSUED. NO MESSAGE SENT"); B85
0687       goto SCHEDULER;
0688     end; E85
0689   TARGET:=M.RECEIVER.ASS_PROCESSOR;
0690   if TARGET==none then
0691     begin RUNTIME_ERROR("SEND TO TERMINATED AGENT ISSUED"); B86
0692       goto SCHEDULER;
0693     end; E86
0694   if TARGET==this PROCESSOR then M.Into(MESSAGE_LIST)
0695   else begin B87
0696     TARGET:-
0697       PROCESSORS(ROUTING_MATRIX(this PROCESSOR.ID,TARGET.ID));
0698       M.TARGET:=TARGET;
0699       B:=-BUSSES(CONNECTION_MATRIX(this PROCESSOR.ID,TARGET.ID));
0700       B.STORE_MESSAGE(M);
0701       if B.Idle then activate B after Current;
0702     end; E87
0703   TOTAL_MSGS:=TOTAL_MSGS+1;
0704 end; E84
0705 goto SCHEDULER;
0706
0707 comment _____
0708 | THE PROCESSOR HAS RECEIVED A MESSAGE AND ROUTES IT TO THE MAILBOX
0709 | OF THE ADDRESSED AGENT .
0710 |
0711 LRECEIVE_MESSAGE:
0712 INT_WEIGHT:=0;
0713 begin ref(ACB) RECEIVER; ref(PROCESSOR) TARGET; B88
0714   ref(MSG) M; ref(BUS) B;
0715   M:-MESSAGE_LIST.First;
0716   RECEIVER:=M.RECEIVER;
0717   TARGET:=RECEIVER.ASS_PROCESSOR;
0718   if TARGET==none then
0719     begin Outtext("XXX");PRINT_TIME; B89
0720       Outtext(" ");PRINT_ID;
0721       Outtext("TARGET AGENT TERMINATED. FOLLOWING MESSAGE LOST:");
0722       OUTLINE;
0723       BMSOUT.Setpos(16);
0724       MESSAGE_LIST.First qua MSG.PRINT(true);
0725       OUTLINE;
0726       MESSAGE_LIST.First.Out;
0727     goto SCHEDULER;
0728   end TARGET==NONE; E89
0729   if TARGET=/= this PROCESSOR then
0730     begin
0731       if SYSTEM then
0732         begin
0733           Outtext("//");PRINT_TIME;
0734           Outtext(" "); PRINT_ID;
0735           Outtext("PASSING ");
0736           MESSAGE_LIST.First qua MSG.PRINT(true);
0737           OUTLINE;
0738         end; E91
0739       TARGET:-
0740       PROCESSORS(ROUTING_MATRIX(this PROCESSOR.ID,TARGET.ID));
0741       M.TARGET:=TARGET;
0742       B:=-BUSSES(CONNECTION_MATRIX(this PROCESSOR.ID,TARGET.ID));
0743       B.STORE_MESSAGE(M);
0744       if B.Idle then activate B after Current;
0745     end else E90
0746     begin
0747       MESSAGE_LIST.First.Into(RECEIVER.MAILBOX);
0748       inspect RECEIVER do
0749         if not STOPPED then Into(ACB_QUEUE);
0750       comment OUT OF IDLE_QUEUE ;
0751     end; B92

```

```

0751      RECEIVER.IDLING:=false;
0752      if EVENT then STATUS(false);
0753      if OBSERVE then
0754      begin
0755          Outtext("+++";PRINT_TIME;
0756          Outtext(" "); PRINT_ID;PRINT_AGENT(RECEIVER);
0757          Outtext(" RECEIVES ");
0758          RECEIVER.MAILBOX.Last qua MSG.PRINT(true);
0759          Outtext(" FROM ");
0760          PRINT_AGENT(RECEIVER.MAILBOX.Last qua MSG.SENDER);
0761          OUTLINE;
0762      end;
0763  end;
0764  end;
0765  goto SCHEDULER;
0766
0767 comment _____
0768 | THE ACTIVE AGENT CREATES A NEW AGENT. THE PROCESSOR FIRST FINDS OUT
0769 | IF HE HAS ALREADY A MODULE OF THE REQUESTED TYPE IN HIS STORAGE.
0770 | IN THIS CASE HE MUST NOT LOAD A NEW MODULE(SCRIPT) SINCE SCRIPTS ARE
0771 | REENTRANT ( CAN BE SHARED BY SEVERAL AGENTS)
0772 |
0773 LCREATE_AGENT:
0774 INT_WEIGHT:=0;
0775 begin ref(MODULE) M;integer SCR_NR;ref(ACB) A;ref(PRECESSOR)P;
0776     P:-PROCESSORS(NEXT_CREATION_PROC);
0777     SCR_NR:=ACTIVE_ACB.SCRIPT_TOP.WERT;
0778     if SCR_NR=0 then
0779         begin RUNTIME_ERROR("NEW <NOSCRIPT> EXECUTED. NOAGENT RETURNED");
0780         AGENT_PUSH(NOAGENT);
0781         goto SCHEDULER;
0782     end;
0783     if SCR_NR=1 then goto NOT_FOUND; comment INTERFACE IS ~REENTRANT;
0784     if P.MODULE_LIST.First==none then goto NOT_FOUND;
0785     for M:-P.MODULE_LIST.First,M.Suc while M=/= none do
0786         if M.OWNMODE=SCR_NR then goto FOUND;
0787         NOT_FOUND: M:-NEW_MODULE(SCR_NR);
0788         M.Into(P.MODULE_LIST);
0789     FOUND: A:-new ACB(M); A.Into(P.ACB_QUEUE);
0790         new AGENT(A).Into(AGENT_LIST);
0791         M.NR_OF_ACBS:=M.NR_OF_ACBS+1;
0792         NR_OF_AGENTS(SCR_NR):=NR_OF_AGENTS(SCR_NR)+1;
0793         M.ASS_PROCESSOR:-P;
0794         A.ASS_PROCESSOR:-P;
0795         A.COPY_NR:=NR_OF_AGENTS(SCR_NR);
0796         A.CURRENT_MSG:=ACTIVE_ACB.MSG_TOP.WERT;
0797         M.PRINT_NAME:=SCRIPT_NAME(SCR_NR);
0798         AGENT_PUSH(A);
0799         NEXT_CREATION_PROC:=NEXT_CREATION_PROC+1;
0800         if NEXT_CREATION_PROC>NR_OF_PROCESSORS
0801             then NEXT_CREATION_PROC:=2;
0802         if EVENT then STATUS(false);
0803         TRACE_CREATION(A);
0804         TOTAL_AGENTS:=TOTAL_AGENTS+1;
0805         if P.Idle then activate P after Current;
0806     end;
0807  goto SCHEDULER;
0808
0809 comment _____
0810 | THE ACTIVE AGENT IS TERMINATED . THE PROCESSOR CAN DELETE THE ACTIVE
0811 | AGENTS ACB. IT HAS TO BE TESTED, IF THE DELETED ACB WAS THE ONLY ACB
0812 | ASSOCIATED TO HIS SCRIPT. IN THE CASE THE SCRIPT CAN BE DELETED AS
0813 | WELL .
0814 |
0815 LTERMINATION:
0816 TRACE_TERMINATION;
0817 INT_WEIGHT:=0;
0818 begin ref(AGENT)A;
0819     A:-AGENT_LIST.First;while A.CB=/=ACTIVE_ACB do A:-A.Suc;
0820     A.Out;
0821     ACTIVE_ACB.ASS_PROCESSOR:-none;
0822     ACTIVE_ACB.Out; comment ...OF ACB_QUEUE;
0823     ACTIVE_ACB:-none;
0824     ACTIVE_CODE.NR_OF_ACBS:=ACTIVE_CODE.NR_OF_ACBS-1;
0825     if ACTIVE_CODE.NR_OF_ACBS=0
0826         then begin ACTIVE_CODE.Out;
0827             ACTIVE_CODE:-none;
0828         end;
0829     end;
0830  goto SCHEDULER;
0831
0832 comment _____
0833 | THE INTERRUPTING AGENT IS IDLE BECAUSE HE HAS NO MATCHING MESSAGES
0834 | IN HIS MAILBOX. THEREFORE HE JOINS THE IDLE_QUEUE UNTIL HE RECEIVES
0835 | ANOTHER MESSAGE.
0836

```

```

0837  LAGENT_IDLE:
0838  if OBSERVE then
0839  begin
0840  Outtext("++");PRINT_TIME;Outtext(" ");PRINT_ID;
0841  PRINT_AGENT(ACTIVE_ACB);
0842  Outtext(" IS IDLE ");OUTLINE;
0843  end;
0844  INT_WEIGHT:=0;
0845  ACTIVE_ACB.Into(IDLE_QUEUE); ACTIVE_ACB.IDLING:=true;
0846  ACTIVE_ACB:=none;
0847  goto SCHEDULER;
0848

0849  comment _____
0850  | THE ACTIVE AGENT HAS WORKED FOR A FULL TIME_SLICE WITHOUT INTERRUPTING THE PROCESSOR. ANOTHER AGENT WILL BE ACTIVATED NOW.
0851
0852
0853  LTIMESLICE_RUNOUT:
0854  if SYSTEM then
0855  begin
0856  Outtext("//");PRINT_TIME;Outtext(" ");PRINT_ID;
0857  Outtext("TIMESLICE_RUNOUT ");OUTLINE;
0858  end;
0859  INT_WEIGHT:=0;
0860
0861  goto SCHEDULER;
0862
0863  comment _____
0864  | THE PROCESSOR IS IDLE BECAUSE ALL HIS AGENTS ARE IDLE. HE HAS TO WAIT FOR AN EXTERNAL INTERRUPT (E.G. RECEIVE MESSAGE) WHICH ALLOWS ONE OF THE AGENTS TO GO ON .
0865
0866
0867
0868  LIDLE:
0869  if SYSTEM then
0870  begin
0871  Outtext("//");PRINT_TIME; Outtext(" ");PRINT_ID;
0872  Outtext("PROCESSOR IS IDLE");OUTLINE;
0873  end;
0874  INT_WEIGHT:=0;
0875  ACTIVE_TIME:= Time-LAST_ACTIVATION+ACTIVE_TIME;
0876  Passivate;
0877  LAST_ACTIVATION:=Time;
0878  if SYSTEM then
0879  begin
0880  Outtext("//");PRINT_TIME; Outtext(" ");PRINT_ID;
0881  Outtext("PROCESSOR IS ACTIVATED BY EXTERNAL INTERRUPT");OUTLINE;
0882  end;
0883  goto SCHEDULER;
0884  end PROCESSOR;
0885

```

B98

E98

B99

E99

B100

E100

B101

E101

E6

```

0887
0888 comment **** DATENBUSSE SIND AKTIVE EINHEITEN DER SIMULATION. ****
0889 ** SIE WAEHLEN EINE NACHRICHT AUS DER WARTESCHLANGE AUS, VERBRAUCHEN**
0890 ** DIE UEBERTRAGUNGSZEIT UND UNTERBRECHEN DANN DEN ZIELPROZESSOR. **
0891 ****;
0892 ****;
0893
0894 Process class BUS(ID);integer ID;
0895 begin
0896   ref(Head) MESSAGE_LIST;
0897   ref(MSG) M; integer U; comment SIEMENS;
0898   real ACTIVE_TIME, LAST_ACTIVATION, AVG_WAIT;
0899   integer NR_OF_MSGS; comment ... ALREADY RECEIVED;
0900
0901   integer procedure UTILIZATION;
0902   if Time>0 then
0903     begin
0904       if Idle
0905         then UTILIZATION:=ACTIVE_TIME/Time*100
0906         else UTILIZATION:=(ACTIVE_TIME+(Time-LAST_ACTIVATION))/Time*100;
0907     end;
0908
0909   procedure INITIALIZE; MESSAGE_LIST:=-new Head;
0910
0911   procedure STORE_MESSAGE(M); ref(MSG) M;
0912   begin
0913     AVG_WAIT:=((AVG_WAIT*NR_OF_MSGS)+MESSAGE_LIST.Cardinal )
0914     /(NR_OF_MSGS+1);
0915     NR_OF_MSGS:=NR_OF_MSGS+1;
0916     M.Into(MESSAGE_LIST);
0917   end;
0918   U:=999; comment SIEMENS;
0919   TRANSMIT:
0920   M:=-MESSAGE_LIST.First;
0921   if M==none then
0922     begin if SYSTEM then
0923       begin Outtext("///");PRINT_TIME; Outtext(" ");
0924         Outchar('B');PRINT_INT(ID);
0925         Outtext(": BUS IS IDLE");OUTLINE;
0926       end;
0927       ACTIVE_TIME:= Time-LAST_ACTIVATION+ACTIVE_TIME;
0928       Passivate;
0929       LAST_ACTIVATION:=Time;
0930       goto TRANSMIT;
0931     end;
0932     comment SIEMENS;
0933     if BUS_QUEUEING='R' then
0934       begin integer I;
0935         for I:=Randint(0,MESSAGE_LIST.Cardinal-1,U) step -1
0936           until 1 do M:=M.Suc;
0937       end;
0938     if SYSTEM then
0939       begin Outtext("///");PRINT_TIME;Outtext(" ");
0940         Outchar('B');PRINT_INT(ID);
0941         Outtext(": STARTING TO TRANSMIT ");
0942         M.PRINT(false);OUTLINE;
0943       end;
0944     Hold(TRANS_TIME);
0945     M.Into(M.TARGET.MESSAGE_LIST);
0946     if M.TARGET.Idle then activate M.TARGET after Current;
0947     goto TRANSMIT;
0948   end BUS;
0949

```

```
0951
0952 comment ****
0953 ** JEDEM AGENTEN IST GENAU EIN ACB ZUGEORDNET, DER SEINEN **
0954 ** ZUSTAND BESCHREIBT. HIER FINDET SICH EIN VERWEIS AUF DEN **
0955 ** SCRIPT-CODE (ASS_SCRIPT) UND DIE EINSPRUNGADRESSE, MIT DER DER **
0956 ** SCRIPT-CODE FORTGESETZT WERDEN MUSS. DER ACB ENTHAELT **
0957 ** AUSSERDEM DIE MAILBOX, DIE STACKS ZU JEDEM DATENTYP, SOWIE **
0958 ** EINEN VERWEIS (AKT_AS) AUF DIE SPITZE DES LAUFZEITKELLERS, **
0959 ** DER ALLE LOKALEN DATEN DES AGENTEN ENTHAELT. DA IM SCRIPT-CODE **
0960 ** SELBST KEINE LOKALEN DATEN GESPEICHERT SIND, IST DER CODE REEN- **
0961 ** TRANT, D.H. ER KANN VON MEHEREN AGENTEN GLEICHEN TYPS GEMEINSAM **
0962 ** BENUTZT WERDEN.
0963 ****;
0964
0965 Link class ACB(ASS_SCRIPT);
0966   ref(MODULE) ASS_SCRIPT;
0967 begin
0968   ref(PROCESSOR) ASS_PROCESSOR;
0969   integer EINSPRUNG; comment ENTRY_ADDRESS OF ASS_SCRIPT;
0970   real LEFT_TIME;
0971   boolean SUCCESS,BRANCH_FLAG;
0972   ref(Head) MAILBOX;
0973   ref(MSG) CURRENT_MSG;
0974   ref(PARAM) CURRENT_PARM;
0975   short integer SET_SIZE;
0976   ref(AS) ENVIRONMENT,
0977     AKT_AS;
0978   ref(ACB) SELF,CALLER,WAITING_AGENT; boolean WAITING;
0979   integer COPY_NR; comment TO DISTINGUISH AGENTS OF THE SAME TYPE;
0980   text CURRENT_FACET,CURRENT_OPER,PORT;
0981
0982   ref(BOOLEL) BOOL_TOP;
0983   ref(STRINGEL) STRING_TOP;
0984   ref(AGENTEL) AGENT_TOP;
0985   ref(OPEREL) OPER_TOP;
0986   ref(SCRIPTEL) SCRIPT_TOP;
0987   ref(REALEL) REAL_TOP;
0988   ref(INTEL) INT_TOP;
0989   ref(INTEL) ENUM_TOP;
0990   ref(MSGEL) MSG_TOP;
0991   boolean TRACE,SNAPSHOT,IDLING,STOPPED,CANCELLED;
0992   integer CURRENT_LINE;
0993
0994 SELF:-this ACB;
0995 MAILBOX :- new Head;
0996
0997 end ACB;
0998
0999 Link class AGENT(CB);ref(ACB)CB;
1000   begin end;
1001
1002   ref(Head) AGENT_LIST;
1003
1004
1005
1006
1007
```

B109

E109

B110 E110

```
1009 comment ****
1010 ** DIE GEMEINSAME OBERKLASSE FUER ALLE VOM COMPILER GENERIERTEN **
1011 ** SCRIPT-MODULE.
1012 ****
1013
1014 Process class MODULE;
1015 begin text PRINT_NAME;
1016     integer NR_OF_ACBS;
1017     integer OWNMODE;
1018     real ELAPSED_TIME;
1019     ref(PROCESSOR) ASS_PROCESSOR;
1020     comment A MODULE IS EXECUTED BY SETTING ASS_PROCESSOR AND THEN
1021         ACTIVATING IT ;
1022 end MODULE;
1023
```

B111

E111

```
1025
1026 comment **** IN STACKS VERKETTETE ELEMENTE ****
1027 ** IN STACKS VERKETTETE ELEMENTE ****
1028 ****;
1029
1030 class BOOLEL(WERT);boolean WERT;
1031 begin ref(BOOLEL) NEXT; end; B112 E112
1032
1033 class REALEL(WERT);long real WERT;
1034 begin ref(REALEL) NEXT; end; B113 E113
1035
1036 class INTEL(WERT);integer WERT;
1037 begin ref(INTEL) NEXT; end; B114 E114
1038
1039 class SCRIPTEL(WERT);integer WERT;
1040 begin ref(SCRIPTEL) NEXT; end; B115 E115
1041
1042 class AGENTEL(WERT);ref(ACB) WERT;
1043 begin ref(AGENTEL) NEXT; end; B116 E116
1044
1045 class STRINGEL(WERT);text WERT;
1046 begin ref(STRINGEL) NEXT; end; B117 E117
1047
1048 class OPEREL(WERT);text WERT;
1049 begin ref(OPEREL) NEXT; end; B118 E118
1050
1051 class MSGEL(WERT);ref(MSG) WERT;
1052 begin ref(MSGEL) NEXT; end; B119 E119
```

```
1054
1055  comment ****
1056  ** AUSGABE VON WERTEN IM CSSA-STANDARDFORMAT      **
1057  ****
1058
1059  procedure PRINT_TIME;
1060    Outfix(Time,3,10);
1061
1062
1063  procedure PRINT_BOOL(B);boolean B;
1064    Outtext(if B then "TRUE" else "FALSE");
1065
1066  procedure PRINT_REAL(R);real R;
1067    Outfix(R,5,12);
1068
1069  procedure PRINT_INT(I); integer I;
1070    if I=0 then Outint(I,1) else
1071      if I>0 then Outint(I,Entier(Ln(Abs(I))*0.4343)+1) else
1072        if I<0 then Outint(I,Entier(Ln(Abs(I))*0.4343)+2);
1073
1074  procedure PRINT_ENUM(E);integer E;
1075  begin Outtext("//");PRINT_INT(E);Outtext("//");end;           B120 E120
1076
1077  procedure PRINT_OPER(O);text O;
1078    Outtext(O);
1079
1080  procedure PRINT_STRING(S); text S;
1081  begin Outchar('"'');
1082    Outtext(S);
1083    Outchar('"');
1084  end;                           E121
1085
1086  procedure PRINT_AGENT(A);ref(ACB) A;
1087  if A==none
1088    then Outtext("<NOAGENT>");
1089  else begin Outtext(A.ASS_SCRIPT.PRINT_NAME);
1090    Outtext("(");PRINT_INT(A.COPY_NR);
1091    Outtext(")");
1092  end;                           E122
1093
1094  procedure PRINT_SCRIPT(S);integer S;
1095  if S=0 then Outtext("<NOSCRIPT>")
1096  else Outtext(SCRIPT_NAME(S));
1097
1098  procedure PRINT_LITERAL(L);text L;
1099    Outtext(L);
```

```

1101  comment ****
1102  ** FUER DIE AUSGABE EINER ZEILE DES LAUFZEITKELLERS DES AGENTEN **
1103  ****
1104
1105  procedure DUMP_ENUM(IDNAME,I);value IDNAME;text IDNAME;integer I;
1106  begin BMSOUT.Setpos(16);Outtext(" | ");Outtext(IDNAME);Outtext(" = ");
1107      PRINT_ENUM(I);
1108      BMSOUT.Setpos(50); Outtext(" | ") ;
1109      OUTLINE;
1110  end;                                              E123
1111
1112  procedure DUMP_REAL(IDNAME,R);value IDNAME;text IDNAME;real R;
1113  begin BMSOUT.Setpos(16);Outtext(" | ");Outtext(IDNAME);Outtext(" = ");
1114      PRINT_REAL(R);
1115      BMSOUT.Setpos(50); Outtext(" | ") ;
1116      OUTLINE;
1117  end;                                              E124
1118
1119  procedure DUMP_INT(IDNAME,I);value IDNAME;text IDNAME;integer I;
1120  begin BMSOUT.Setpos(16);Outtext(" | ");Outtext(IDNAME);Outtext(" = ");
1121      PRINT_INT(I);
1122      BMSOUT.Setpos(50); Outtext(" | ") ;
1123      OUTLINE;
1124  end;                                              E125
1125
1126  procedure DUMP_BOOL(IDNAME,B);value IDNAME;text IDNAME;boolean B;
1127  begin BMSOUT.Setpos(16);
1128      Outtext(" | ");Outtext(IDNAME);Outtext(" = ");
1129      PRINT_BOOL(B);
1130      BMSOUT.Setpos(50); Outtext(" | ") ;
1131      OUTLINE;
1132  end;                                              E126
1133
1134  procedure DUMP_OPER(IDNAME,O);
1135  value IDNAME;text IDNAME;text O;
1136  begin BMSOUT.Setpos(16);Outtext(" | ");Outtext(IDNAME);Outtext(" = ");
1137      PRINT_OPER(O);
1138      BMSOUT.Setpos(50); Outtext(" | ") ;
1139      OUTLINE;
1140  end;                                              E127
1141
1142  procedure DUMP_STRING(IDNAME,S);value IDNAME;text IDNAME;text S;
1143  begin BMSOUT.Setpos(16);Outtext(" | ");Outtext(IDNAME);Outtext(" = ");
1144      PRINT_STRING(S);
1145      BMSOUT.Setpos(50); Outtext(" | ") ;
1146      OUTLINE;
1147  end;                                              E128
1148
1149  procedure DUMP_AGENT(IDNAME,A);value IDNAME;text IDNAME;ref(ACB)A;
1150  begin BMSOUT.Setpos(16);Outtext(" | ");Outtext(IDNAME);Outtext(" = ");
1151      PRINT_AGENT(A);
1152      BMSOUT.Setpos(50); Outtext(" | ") ;
1153      OUTLINE;
1154  end;                                              E129
1155
1156  procedure DUMP_SCRIPT(IDNAME,S);value IDNAME;text IDNAME;integer S;
1157  begin BMSOUT.Setpos(16);Outtext(" | ");Outtext(IDNAME);Outtext(" = ");
1158      PRINT_SCRIPT(S);
1159      BMSOUT.Setpos(50); Outtext(" | ") ;
1160      OUTLINE;
1161  end;                                              E130
1162
1163  procedure OBSERVE_DUMP(A);ref(ACB)A;
1164  begin if ~A.TRACE then
1165      begin
1166          OUTLINE; Outtext("++");PRINT_TIME;Outtext(" ");
1167          Outtext("RUNTIME STACK OF ");
1168          PRINT_AGENT(A);
1169          OUTLINE;
1170      end;
1171      DUMP(A);
1172  end;                                              E131
1173
1174  procedure PRINT_DUMP(A);ref(ACB)A;
1175  begin
1176      OUTLINE; Outtext("++");PRINT_TIME;Outtext(" ");
1177      Outtext("RUNTIME STACK OF ");
1178      PRINT_AGENT(A);
1179      OUTLINE;
1180      DUMP(A);
1181  end;                                              E133
1182
1183  procedure DUMP(A); ref(ACB) A;
1184  inspect A do
1185  begin ref(AS) STACK_EL;
1186      BMSOUT.Setpos(16);

```

```
1187     Outtext(" _____");OUTLINE;
1188     STACK_EL:-A.AKT_AS;
1189     while STACK_EL=/=none do inspect STACK_EL do
1190     begin BMSOUT.Setpos(16);
1191         Outtext(" | ");Outtext(SCOPE_TYPE);Outtext(" ");
1192         Outtext(SCOPE_NAME):BMSOUT.Setpos(51); Outtext(" | ");
1193         OUTLINE; BMSOUT.Setpos(16);
1194         Outtext(" | ENV: ");
1195         inspect E do begin
1196             Outtext(SCOPE_TYPE);Outtext(" ");
1197             Outtext(SCOPE_NAME);
1198         end;
1199         BMSOUT.Setpos(51); Outtext(" | ");
1200         OUTLINE;BMSOUT.Setpos(16);
1201         Outtext(" | LINE:");
1202         Outint(if RAS_LINE>0 then RAS_LINE else CURRENT_LINE,5);
1203         BMSOUT.Setpos(51); Outtext(" | ");
1204         OUTLINE;BMSOUT.Setpos(16);
1205         Outtext(" | ..... |");OUTLINE;
1206         STACK_EL.PRINT_VARS;
1207         BMSOUT.Setpos(16);
1208         Outtext(" | _____ |");OUTLINE;
1209         STACK_EL:-STACK_EL.LAST_AS;
1210     end;
1211     OUTLINE;
1212 end;
```

```

1214  comment ****AUSDRUCKEN EINER ZEILE DER STATUS-TABELLE ****
1215  ** ASS_PROCESSOR.PRINT_ID; A;boolean SENDING;
1216  ****;
1217
1218  procedure PRINT_STATUS(A,SENDING); ref(ACB) A;boolean SENDING;
1219  inspect A do
1220  begin ref(MSG)M;
1221      Outtext(" | ");
1222      ASS_PROCESSOR.PRINT_ID;
1223      PRINT_AGENT(SELF);
1224      BMSOUT.Setpos(20);Outtext(if IDLING then " " else "*");
1225      Outtext(" | ");
1226      Outtext(CURRENT_FACET);Outtext("      ");
1227      BMSOUT.Setpos(32);Outtext(" | ");
1228      Outtext(CURRENT_OPER); Outtext("      ");
1229      BMSOUT.Setpos(44);Outtext(" | ");
1230      if SENDING
1231      then begin Outtext("==> ");
1232          if MSG_TOP.WERT.PARMS.Cardinal<2
1233              then MSG_TOP.WERT.PRINT(false)
1234              else Outtext(MSG_TOP.WERT.OPER);
1235      end
1236  else
1237  begin
1238      M:=-MAILBOX.First;
1239      while M/=none do
1240      begin if M.OPER.Length>75-BMSOUT.Pos then
1241          begin BMSOUT.Setpos(77);Outtext(" |");OUTLINE;
1242              Outtext(" | ");BMSOUT.Setpos(20);Outtext(" | ");
1243              BMSOUT.Setpos(32);Outtext(" | ");
1244              BMSOUT.Setpos(44);Outtext(" | ");
1245          end;
1246          if M.PARMS.Cardinal<2 and MAILBOX.Cardinal=1
1247              then M.PRINT(false)
1248              else Outtext(M.OPER);
1249              Outtext(" ");
1250              M:=M.Suc;
1251      end;
1252  end;
1253  BMSOUT.Setpos(77);Outtext(" |");
1254  OUTLINE;
1255 end;
1256

```

```

1258  comment ****
1259  ** AUSGABE DER JEDEM DATENTYP ZUGEORDNETEN STACKS EINES AGENTEN **
1260  ****
1261
1262      procedure PRINT_STACKS(A);ref(ACB) A;
1263      begin ref(BOOLEL)P_BOOLEL;
1264          ref(AGENTEL)P_AGENTEL;
1265          ref(SCRIPTEL)P_SCRIPTEL;
1266          ref(INTEL)P_INTEL;
1267          ref(REALEL)P_REALEL;
1268          ref(STRINGEL)P_STRINGEL;
1269          ref(OPEREL)P_OPEREL;
1270          ref(MSGEL)P_MSGEL;
1271          Outtext("BOOLSTACK: ");
1272          P_BOOLEL:=A.BOOL_TOP;
1273          while P_BOOLEL/=none do
1274              begin PRINT_BOOL(P_BOOLEL.WERT);Outtext(" | ");
1275                  P_BOOLEL:=P_BOOLEL.NEXT;
1276              end;
1277          OUTLINE;
1278          Outtext("INTSTACK: ");
1279          P_INTEL:=A.INT_TOP;
1280          while P_INTEL/=none do
1281              begin PRINT_INT(P_INTEL.WERT);Outtext(" | ");
1282                  P_INTEL:=P_INTEL.NEXT;
1283              end;
1284          OUTLINE;
1285          Outtext("REALSTACK: ");
1286          P_REALEL:=A.REAL_TOP;
1287          while P_REALEL/=none do
1288              begin PRINT_REAL(P_REALEL.WERT);Outtext(" | ");
1289                  P_REALEL:=P_REALEL.NEXT;
1290              end;
1291          OUTLINE;
1292          Outtext("AGENTSTACK: ");
1293          P_AGENTEL:=A.AGENT_TOP;
1294          while P_AGENTEL/=none do
1295              begin PRINT_AGENT(P_AGENTEL.WERT);Outtext(" | ");
1296                  P_AGENTEL:=P_AGENTEL.NEXT;
1297              end;
1298          OUTLINE;
1299          Outtext("SCRIPTSTACK: ");
1300          P_SCRIPTEL:=A.SCRIPT_TOP;
1301          while P_SCRIPTEL/=none do
1302              begin PRINT_SCRIPT(P_SCRIPTEL.WERT);Outtext(" | ");
1303                  P_SCRIPTEL:=P_SCRIPTEL.NEXT;
1304              end;
1305          OUTLINE;
1306          Outtext("STRINGSTACK: ");
1307          P_STRINGEL:=A.STRING_TOP;
1308          while P_STRINGEL/=none do
1309              begin PRINT_STRING(P_STRINGEL.WERT);Outtext(" | ");
1310                  P_STRINGEL:=P_STRINGEL.NEXT;
1311              end;
1312          OUTLINE;
1313          Outtext("OPERSTACK: ");
1314          P_OPEREL:=A.OPER_TOP;
1315          while P_OPEREL/=none do
1316              begin PRINT_OPER(P_OPEREL.WERT);Outtext(" | ");
1317                  P_OPEREL:=P_OPEREL.NEXT;
1318              end;
1319          OUTLINE;
1320          Outtext("MSGSTACK: ");
1321          P_MSGEL:=A.MSG_TOP;
1322          while P_MSGEL/=none do
1323              begin P_MSGEL.WERT.PRINT(true);OUTLINE;
1324                  P_MSGEL:=P_MSGEL.NEXT;
1325              end;
1326          OUTLINE; OUTLINE;
1327      end;

```

```
1329  comment ****NACHRICHTEN SIND OBJEKTE VOM TYP MSG, KOENNEN ZU LISTEN ****
1330  ** ZUSAMMENGEFASST WERDEN UND ENTHALTEN IHRERSEITS EINE LISTE      **
1331  ** PARAMETERN          **
1332  ****
1333 ****
1334
1335 Link class MSG;
1336 begin
1337   text OPER;
1338   ref(Head) PARMs;
1339   boolean REPLY_EXPECTED;
1340   text PORT;
1341   ref(ACB) SENDER,RECEIVER,WAITING_AGENT;
1342   ref(PROCESSOR) TARGET;
1343   procedure PRINT(FULL);boolean FULL;
1344   begin ref(PARAM) P;
1345     PRINT_OPER(OPER);Outtext("(");
1346     P:-PARMS.First;
1347     while P/= none do
1348       begin P.PRINT;
1349         P:-P.Suc;
1350         if P=/=none then Outchar(',');
1351       end;
1352     Outtext(")");
1353     if REPLY_EXPECTED and FULL
1354       then begin Outtext(",REPLY TO: ");
1355         Outtext(PORT);
1356       end;
1357     end;
1358     PARMs:=new Head;
1359   end MSG;
1360
1361 Link class PARAM(TYP) ; value TYP ; text TYP ;
1362 begin
1363   integer INTWERT ;
1364   long real REALWERT;
1365   text STRINGWERT ;
1366   ref(ACB) AGENTWERT ;
1367   boolean BOOLWERT ;
1368   integer SCRIPTWERT ;
1369   text OPERWERT ;
1370   procedure PRINT;
1371
1372   begin Outtext(TYP);Outchar(':'');
1373     if TYP="INT" then PRINT_INT(INTWERT);
1374     if TYP="REAL" then PRINT_REAL(REALWERT);
1375     if TYP="BOOL" then PRINT_BOOL(BOOLWERT);
1376     if TYP="AGENT" then PRINT_AGENT(AGENTWERT);
1377     if TYP="STRING" then PRINT_STRING(STRINGWERT);
1378     if TYP="SCRIPT" then PRINT_SCRIPT(SCRIPTWERT);
1379     if TYP="OPER" then PRINT_OPER(OPERWERT);
1380   end;
1381 end PARAM;
1382
```

```
1384  comment ****
1385  ** DIE GEMEINSAME OBERKLASSE FUER ALLE VOM COMPILER GENERIERTEN   **
1386  ** AKTIVIERUNGSSAETZE.                                         **
1387  ****
1388
1389  class AS(LAST_AS,E);
1390    ref(AS) E,LAST_AS; comment STATISCHES ENVIRONMENT+STACK POINTER;
1391    virtual: procedure PRINT_VARS;
1392 begin integer RAS,RAS_LINE;
1393   text SCOPE_TYPE,SCOPE_NAME;
1394 end;
1395
1396  comment ****
1397  ** AUSGABE EINER ZEILE WAEHREND DER INTERAKTIVEN CSSA-SITZUNG AUF  **
1398  ** SOWOHL DEN BILDSCHIRM ALS AUCH DIE PROTOKOLL-DATEI             **
1399  ****
1400
1401  procedure OUTLINE;
1402 begin
1403   integer POSITION;
1404   POSITION:= if BMSOUT.Image.Pos<80 then BMSOUT.Image.Pos else 80;
1405   inspect PROT do
1406     begin Outtext(BMSOUT.Image.Sub(1,POSITION-1));
1407       while BMSOUT.Image.Pos>POSITION do
1408         begin
1409           Outchar('#'); Outimage;
1410           Outtext("      ");
1411           Outtext(BMSOUT.Image.Sub(POSITION,63));
1412           POSITION:=POSITION+63;
1413         end;
1414         Outimage;
1415       end;
1416   POSITION:= if BMSOUT.Image.Pos<80 then BMSOUT.Image.Pos else 80;
1417   inspect Sysout do
1418     begin Outtext(BMSOUT.Image.Sub(1,POSITION-1));
1419       while BMSOUT.Image.Pos>POSITION do
1420         begin
1421           Outimage;
1422           Outtext("      ");
1423           Outtext(BMSOUT.Image.Sub(POSITION,63));
1424           POSITION:=POSITION+63;
1425         end;
1426         if POSITION>1 then Outimage; comment KEINE LEERZEILEN ;
1427       end;
1428     BMSOUT.Image:=notext;
1429     BMSOUT.Image.Setpos(1);
1430   end;
1431
```

```

1433
1434   ref(Outfile)PROT;
1435   ref(Outfile)PROTOKOL;
1436   ref(Infile) CONFIG; integer I;
1437   integer NOSCRIPT;
1438   ref(ACB) NOAGENT,INTERFACE;
1439   text NOOPER;
1440   real TIMESLICE;
1441   character BUS_QUEUEING;
1442   integer NR_OF_PROCESSORS, TOTAL_AGENTS, TOTAL_MSGS;
1443   ref(PROCESSOR) array PROCESSORS(1:10);
1444   integer array ROUTING_MATRIX(1:10,1:10);
1445   integer NR_OF_BUSES;
1446   real TRANS_TIME;
1447   ref(BUS) array BUSSES(1:10);
1448   integer array CONNECTION_MATRIX(1:10,1:10);
1449   integer NEXT_CREATION_PROC;
1450   integer RECEIVE_MESSAGE,SEND_MESSAGE,CREATE_AGENT,TERMINATION
1451   ,AGENT_IDLE,TIMESLICE_RUNOUT;
1452   boolean OBSERVE,EVENT,SINGLE_STEP,SYSTEM;
1453
1454   INIT_ACTIONS ;
1455   CONFIG:-new Infile("CONFIG");
1456   PROTOKOL:-new Outfile("PROTOKOL");
1457   PROTOKOL.Open(Blanks(132));comment SIEMENS;
1458   TIMESLICE_RUNOUT:=1;
1459   SEND_MESSAGE:=2;
1460   RECEIVE_MESSAGE:=3;
1461   CREATE_AGENT:=4;
1462   TERMINATION:=5;
1463   AGENT_IDLE:=6;
1464
1465   NEXT_CREATION_PROC:=2;
1466
1467   NOSCRIPT:=0;
1468   NOAGENT:-none;
1469   NOOPER:-notext;
1470
1471   comment ****
1472   ** HIER WIRD DIE VOM BENUTZER SPEZIFIZIERTE HARDWARE-KONFIGURATION **
1473   ** EINGELESEN. DIE ERSTEN 25 ZEILEN DER DATEI DIENEN ALS **
1474   ** KOMMENTAR UND WERDEN AUF DEN BILDSCHIRM AUSGEGEBEN. **
1475   ****
1476
1477   CLEAR_SCREEN ;
1478   CONFIG.Open(Blanks(80));
1479 inspect Sysout do
1480 begin
1481   for I:=1 step 1 until 25 do
1482     begin Outtext(CONFIG.Intext(80).Sub(1,72));Outimage;end;
1483     Outtext("           PROGRAM GENERATED ON ") ; Outtext(GENDATE) ;
1484     Outtext(" AT ") ; Outtext(GENTIME) ; Outimage;
1485     Outtext("           BY BMS-CSSA-COMPILER (VERS. ");
1486     Outtext(RELEASEDATE) ; Outchar(')') ; Outimage ;Outimage;
1487 end;
1488 inspect CONFIG do
1489 begin integer I,J;
1490   NR_OF_PROCESSORS:=Inint; Inimage;
1491   TIMESLICE:=Inreal; Inimage;
1492   Inimage; comment LEER;
1493   Inimage; comment KOMMENTAR;
1494   Inimage; comment LEER;
1495   Inimage; comment SPALTENUEBERSCHRIFTEN;
1496   Inimage; comment LEER;
1497
1498   for I:=1 step 1 until NR_OF_PROCESSORS do
1499     begin Inint; comment ZEILENUEBERSCHRIFT;
1500       for J:=1 step 1 until NR_OF_PROCESSORS do
1501         ROUTING_MATRIX(I,J):=Inint;
1502         Inimage; comment NUMMERIERUNG AUSLASSEN;
1503         Inimage; comment LEERZEILE;
1504     end;
1505     Inimage; comment LEERZEILE NACH ROUTING_MATRIX;
1506     NR_OF_BUSES:=Inint; Inimage;
1507     TRANS_TIME:=Inreal; Inimage;
1508     BUS_QUEUEING:=Inchar;Inimage;
1509     Inimage; comment LEERZEILE;
1510     Inimage; comment KOMMENTAR;
1511     Inimage; comment LEERZEILE;
1512     Inimage; comment SPALTENUEBERSCHRIFTEN;
1513     Inimage; comment LEERZEILE;
1514
1515   for I:=1 step 1 until NR_OF_PROCESSORS do
1516     begin Inint; comment ZEILENUEBERSCHRIFT;
1517       for J:=1 step 1 until NR_OF_PROCESSORS do
1518         CONNECTION_MATRIX(I,J):=Inint;

```

```
1519      Inimage; comment NUMMERIERUNG AUSLASSEN;
1520      Inimage; comment LEERZEILE;
1521      end;
1522      Close;
1523      end;
1524      end ***** CLASS CSSA *****;
```

```
1526  %ENDCOPY
1527  CSSA begin
1528  comment ===== USER DEFINED EXTERNAL CODE =====;
1529  ref(Infle) INPUTFILE;
1530
1531  procedure CSSAOPENINPUT;
1532  begin
1533      INPUTFILE:-new Infle("BMSINPUT");
1534      INPUTFILE.Open(Blanks(80));
1535  end;
1536
1537  text procedure CSSAREADREC;
1538  begin INPUTFILE.Inimage;
1539      CSSAREADREC:-INPUTFILE.Intext(70);
1540  end;
1541
1542  boolean procedure CSSAEND_OF_FILE;
1543      CSSAEND_OF_FILE:=INPUTFILE.Endfile;
1544
1545  procedure CSSACLOSEINPUT;
1546      INPUTFILE.Close;
1547  comment ===== END OF USER DEFINED EXTERNAL CODE =====;
1548  %COPY RELATION          0  0A
```

```

1550  class RELATIONS;
1551  begin ref(RECORD) CURR,BEFORE_CURR,First,Last;                                B171
1552  procedure DUMP(N);text N;
1553  begin integer I;
1554      if First==none then
1555          begin
1556              BMSOUT.Setpos(16);Outtext(" |  ");Outtext(N);
1557              Outtext(" IS EMPTY ");
1558              BMSOUT.Setpos(50);Outtext(" | ");
1559              OUTLINE;
1560          end;
1561          CURR:=First;
1562          while CURR/=none do
1563              begin I:=I+1;
1564                  BMSOUT.Setpos(16);Outtext(" |  ");Outtext(N);Outchar('(');
1565                  PRINT_INT(I);
1566                  Outtext(" : ");
1567                  BMSOUT.Setpos(50);Outtext(" | ");
1568                  OUTLINE;
1569                  CURR.DUMP;
1570                  CURR:=CURR.NEXT;
1571          end;
1572      end;
1573  procedure Locate(X);ref(RECORD) X;
1574  begin CURR:=First; BEFORE_CURR:=none;
1575      while CURR/=none do
1576          begin if CURR.EQUAL(X) then goto FOUND;
1577              BEFORE_CURR:=CURR;
1578              CURR:=CURR.NEXT;
1579          end; FOUND:
1580      end;
1581  procedure DELETE(X);ref(RECORD) X;
1582  begin ref(RECORD) NEXT_REC;
1583
1584      Locate(X);
1585      inspect CURR do
1586          begin
1587              inspect BEFORE_CURR do NEXT:=-CURR.NEXT
1588                  otherwise First:=-CURR.NEXT;
1589                  if CURR==Last then Last:=-BEFORE_CURR;
1590                  inspect Current qua MODULE.ASS_PROCESSOR do
1591                      if ACTIVE_ACB	TRACE then
1592                          begin PRINT_LINENO;Outtext("RECORD DELETED");OUTLINE;end;
1593          end;
1594      end;
1595  procedure INSERT(X);ref(RECORD) X;
1596  begin DELETE(X);
1597      inspect Last do NEXT:=-X otherwise First:=-X;
1598      Last:=-X;
1599      inspect Current qua MODULE.ASS_PROCESSOR do
1600          if ACTIVE_ACB	TRACE then
1601              begin PRINT_LINENO;Outtext("RECORD INSERTED");OUTLINE;end;
1602      end;
1603  ref(RECORD) procedure FIND(X);ref(RECORD) X;
1604  comment RESULTS IN SETTING A POINTER TO THE RECORD IF EXISTENT -
1605      OTHERWISE RETURNS X ;
1606  begin Locate(X);
1607      if CURR/=none then
1608          begin Current qua MODULE.ASS_PROCESSOR.BOOL_PUSH(true);
1609              FIND:=-CURR;
1610              inspect Current qua MODULE.ASS_PROCESSOR do
1611                  if ACTIVE_ACB	TRACE then
1612                      begin PRINT_LINENO;Outtext("RECORD FOUND");OUTLINE;end;
1613          end else
1614              begin Current qua MODULE.ASS_PROCESSOR.BOOL_PUSH(false);
1615                  FIND:=-X;
1616                  inspect Current qua MODULE.ASS_PROCESSOR do
1617                      if ACTIVE_ACB	TRACE then
1618                          begin PRINT_LINENO;Outtext("RECORD NOT FOUND");OUTLINE;end;
1619          end;
1620      end;
1621
1622  procedure TEST(X);ref(RECORD) X;
1623  begin Locate(X);
1624      if CURR/=none
1625          then Current qua MODULE.ASS_PROCESSOR.BOOL_PUSH(true)
1626          else Current qua MODULE.ASS_PROCESSOR.BOOL_PUSH(false);
1627  end;
1628  end;
1629
1630  procedure DUMP_RELATION(IDNAME,R);
1631      value IDNAME;text IDNAME;ref(RELATION) R;
1632      inspect R do DUMP(IDNAME);
1633  %ENDCOPY
1634  %COPY RECORD

```

```

1636   class RECORD(NR_OF_FIELDS,NR_OF_KEYS);integer NR_OF_FIELDS,NR_OF_KEYS;
1637     virtual:procedure PUSH;
1638       procedure ASSIGN;
1639       procedure DUMP;
1640       boolean procedure EQUAL;
1641 begin integer I; text array FIELD_TYPE(1:NR_OF_FIELDS);
1642   ref(RECORD) NEXT;
1643   procedure DUMP_ALL(N);text N;
1644   begin EMSOUT.Setpos(16);Outtext(" | ");Outtext(N);Outtext(" : ");
1645     BMSOUT.Setpos(50);Outtext(" | ");
1646     OUTLINE;DUMP;
1647 end;
1648 procedure PUSH_ALL;
1649   for I:=NR_OF_FIELDS step -1 until 1 do PUSH(I);
1650 procedure ASSIGN_ALL;
1651   for I:=1 step 1 until NR_OF_FIELDS do ASSIGN(I);
1652 procedure PUSH_KEY;
1653   for I:=1 step 1 until NR_OF_KEYS do PUSH(I);
1654 procedure ASSIGN_KEY;
1655   for I:= NR_OF_KEYS step -1 until 1 do ASSIGN(I);
1656 procedure GET_PARMS;
1657 inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
1658 begin for I:=1 step 1 until NR_OF_FIELDS do
1659   begin if CURRENT_PARM==none
1660     then begin BOOL_PUSH(false); goto NOMATCH;end;
1661     if CURRENT_PARM.TYP~=FIELD_TYPE(I)
1662     then begin BOOL_PUSH(false); goto NOMATCH;end;
1663     if CURRENT_PARM.TYP=="STRING"
1664       then STRING_PUSH(CURRENT_PARM.STRINGWERT);
1665     if CURRENT_PARM.TYP=="INT"
1666       then INT_PUSH(CURRENT_PARM.INTWERT);
1667     if CURRENT_PARM.TYP=="REAL"
1668       then REAL_PUSH(CURRENT_PARM.REALWERT);
1669     if CURRENT_PARM.TYP=="BOOL"
1670       then BOOL_PUSH(CURRENT_PARM.BOOLWERT);
1671     if CURRENT_PARM.TYP=="AGENT"
1672       then AGENT_PUSH(CURRENT_PARM.AGENTWERT);
1673     if CURRENT_PARM.TYP=="OPER"
1674       then OPER_PUSH(CURRENT_PARM.OPERTWERT);
1675     if CURRENT_PARM.TYP=="SCRIPT"
1676       then SCRIPT_PUSH(CURRENT_PARM.SCRIPTWERT);
1677     ASSIGN(I);
1678     CURRENT_PARM:=-CURRENT_PARM.Suc;
1679   end;
1680   BOOL_PUSH(true);
1681   NOMATCH:
1682 end;
1683
1684 procedure SET_PARMS;
1685 inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
1686 for I:=1 step 1 until NR_OF_FIELDS do
1687 begin ref(PARM) NEW_PARM;
1688   NEW_PARM:=new PARAM(FIELD_TYPE(I));
1689   NEW_PARM.Into(MSG_TOP.WERT.PARMS);
1690   PUSH(I);
1691   if NEW_PARM.TYP=="INT"
1692     then begin NEW_PARM.INTWERT:=INT_TOP.WERT;INT_POP;INT_POP;end;
1693   if NEW_PARM.TYP=="REAL"
1694     then begin NEW_PARM.REALWERT:=REAL_TOP.WERT;
1695       REAL_POP;REAL_POP;end;
1696   if NEW_PARM.TYP=="BOOL"
1697     then begin NEW_PARM.BOOLWERT:=BOOL_TOP.WERT;
1698       BOOL_POP;BOOL_POP;end;
1699   if NEW_PARM.TYP=="AGENT"
1700     then begin NEW_PARM.AGENTWERT:=-AGENT_TOP.WERT;
1701       AGENT_POP;AGENT_POP;end;
1702   if NEW_PARM.TYP=="SCRIPT"
1703     then begin NEW_PARM.SCRIPTWERT:=SCRIPT_TOP.WERT;
1704       SCRIPT_POP;SCRIPT_POP;end;
1705   if NEW_PARM.TYP=="OPER"
1706     then begin NEW_PARM.OPERTWERT:=-OPER_TOP.WERT;
1707       OPER_POP;OPER_POP;end;
1708   if NEW_PARM.TYP=="STRING"
1709     then begin NEW_PARM.STRINGWERT:=-STRING_TOP.WERT;
1710       STRING_POP;STRING_POP;end;
1711   end;
1712 end RECORD;
1713
1714 procedure DUMP_RECORD(IDNAME,R);value IDNAME;text IDNAME;ref(RECORD)R;
1715   inspect R do DUMP_ALL(IDNAME);
1716
1717 ref(RECORD) NEWREC;
1718 %ENDCOPY
1719 %COPY ARRAY

```

```

1721  class ARRAY_DESC(NR_OF_DIMS);integer NR_OF_DIMS;
1722  begin integer array LBOUND,UBOUND(1:5);
1723      procedure DUMP(IDNAME);value IDNAME;text IDNAME;
1724      begin BMSOUT.Setpos(16);Outtext("| ");Outtext(IDNAME);
1725          Outtext(" = ");
1726          for I:=1 step 1 until NR_OF_DIMS do
1727              begin PRINT_INT(LBOUND(I));Outtext("..");
1728                  PRINT_INT(UBOUND(I));
1729                  if I<NR_OF_DIMS then Outchar(',') else Outchar(')');
1730              end;
1731              BMSOUT.Setpos(50);
1732              Outtext(" | ");
1733              OUTLINE;
1734      end DUMP;
1735      integer I;
1736      comment --- DIE ARRAY-GRENZEN STEHEN RUECKWAERTS IM INT_STACK;
1737      inspect Current qua MODULE.ASS_PROCESSOR do
1738          for I:=NR_OF_DIMS step -1 until 1 do
1739              begin UBOUND(I):=ACTIVE_ACB.INT_TOP.WERT;INT_POP;
1740                  LBOUND(I):=ACTIVE_ACB.INT_TOP.WERT;INT_POP;
1741                  if UBOUND(I)<LBOUND(I) then
1742                      begin RUNTIME_ERROR("LBOUND>UBOUND AT ARRAY GENERATION");
1743                          UBOUND(I):=LBOUND(I);
1744                      end;
1745              end;
1746      end;
1747
1748
1749  class FIELD(DESC);ref(ARRAY_DESC) DESC;
1750      virtual: procedure PUSH_ELEMENT;
1751          procedure ASSIGN_ELEMENT;
1752          procedure SAVE_NEW_VALUE;
1753          procedure PRINT;
1754          procedure DUMP;
1755          procedure READ_PARM;
1756          procedure SET_PARM;
1757      begin integer I;
1758          text TYPE_OF_ARRAY;
1759          integer array INDEX(1:5);
1760          integer I1,I2,I3,I4,I5;
1761          procedure PUSH_ALL;
1762          begin for I1:=DESC.UBOUND(1) step -1 until DESC.LBOUND(1) do
1763              for I2:=DESC.UBOUND(2) step -1 until DESC.LBOUND(2) do
1764                  for I3:=DESC.UBOUND(3) step -1 until DESC.LBOUND(3) do
1765                      for I4:=DESC.UBOUND(4) step -1 until DESC.LBOUND(4) do
1766                          for I5:=DESC.UBOUND(5) step -1 until DESC.LBOUND(5) do
1767                              PUSH_ELEMENT(I1,I2,I3,I4,I5);
1768          end;
1769          procedure ASSIGN_ALL;
1770          inspect Current qua MODULE.ASS_PROCESSOR.ACTIVE_ACB do
1771              begin boolean NOT_FIRST_TIME;
1772                  for I1:=DESC.LBOUND(1) step 1 until DESC.UBOUND(1) do
1773                      for I2:=DESC.LBOUND(2) step 1 until DESC.UBOUND(2) do
1774                          for I3:=DESC.LBOUND(3) step 1 until DESC.UBOUND(3) do
1775                              for I4:=DESC.LBOUND(4) step 1 until DESC.UBOUND(4) do
1776                                  for I5:=DESC.LBOUND(5) step 1 until DESC.UBOUND(5) do
1777                                      begin if TRACE then
1778                                          begin if NOT_FIRST_TIME
1779                                              then begin Outtext(Blanks(25));Outchar('')end;
1780                                              PRINT_INDICES(I1,I2,I3,I4,I5);
1781                                              Outtext(" := ");
1782                                              NOT_FIRST_TIME:=true;
1783                                          end;
1784                                          SAVE_NEW_VALUE;
1785                                          ASSIGN_ELEMENT(I1,I2,I3,I4,I5);
1786                                      end;
1787          end;
1788          procedure SET_PARMS;
1789          inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
1790              begin
1791                  for I1:=DESC.LBOUND(1) step 1 until DESC.UBOUND(1) do
1792                      for I2:=DESC.LBOUND(2) step 1 until DESC.UBOUND(2) do
1793                          for I3:=DESC.LBOUND(3) step 1 until DESC.UBOUND(3) do
1794                              for I4:=DESC.LBOUND(4) step 1 until DESC.UBOUND(4) do
1795                                  for I5:=DESC.LBOUND(5) step 1 until DESC.UBOUND(5) do
1796                                      begin new PARAM(TYPE_OF_ARRAY).Into(MSG_TOP.WERT.PARMS);
1797                                          SET_PARM(I1,I2,I3,I4,I5);
1798                                      end;
1799          end;
1800          procedure GET_PARM;
1801          inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
1802              begin for I:= DESC.NR_OF_DIMS step -1 until 1 do
1803                  begin INDEX(I):=INT_TOP.WERT;
1804                      if INDEX(I)<DESC.LBOUND(I) or INDEX(I)>DESC.UBOUND(I)
1805                          then begin RUNTIME_ERROR
1806                              ("ARRAY BOUNDS ERROR. UBOUND USED.");
1807                      end;
1808              end;
1809          end;
1810      end;
1811  
```

```

1807           INDEX(I):=DESC.UBOUND(I);
1808           end;
1809           INT_POP;
1810           end;
1811           if CURRENT_PARM==none
1812             then begin BOOL_PUSH(false); goto NOMATCH;end;
1813           if CURRENT_PARM.TYP~=TYPE_OF_ARRAY
1814             then begin BOOL_PUSH(false); goto NOMATCH;end;
1815           if TRACE then
1816             begin PRINT_INDICES
1817               (INDEX(1),INDEX(2),INDEX(3),INDEX(4),INDEX(5));
1818               Outtext(" := ");
1819             end;
1820           READ_PARM;
1821           ASSIGN_ELEMENT
1822             (INDEX(1),INDEX(2),INDEX(3),INDEX(4),INDEX(5));
1823           CURRENT_PARM:=CURRENT_PARM.Suc;
1824           BOOL_PUSH(true);
1825           NOMATCH:
1826           end;
1827           procedure GET_PARMS;
1828           inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
1829           begin boolean NOT_FIRST_TIME;
1830             for I1:=DESC.LBOUND(1) step 1 until DESC.UBOUND(1) do
1831             for I2:=DESC.LBOUND(2) step 1 until DESC.UBOUND(2) do
1832             for I3:=DESC.LBOUND(3) step 1 until DESC.UBOUND(3) do
1833             for I4:=DESC.LBOUND(4) step 1 until DESC.UBOUND(4) do
1834             for I5:=DESC.LBOUND(5) step 1 until DESC.UBOUND(5) do
1835             begin if CURRENT_PARM==none
1836               then begin BOOL_PUSH(false); goto NOMATCH;end;
1837               if CURRENT_PARM.TYP~=TYPE_OF_ARRAY
1838                 then begin BOOL_PUSH(false); goto NOMATCH;end;
1839               if TRACE then
1840                 begin if NOT_FIRST_TIME
1841                   then begin Outtext(Blanks(25));Outchar('');end;
1842                   PRINT_INDICES(I1,I2,I3,I4,I5);
1843                   Outtext(" := ");
1844                   NOT_FIRST_TIME:=true;
1845                 end;
1846               READ_PARM;
1847               ASSIGN_ELEMENT(I1,I2,I3,I4,I5);
1848               CURRENT_PARM:=CURRENT_PARM.Suc;
1849             end;
1850             BOOL_PUSH(true);
1851             NOMATCH:
1852             end;
1853             procedure PUSH;
1854             inspect Current qua MODULE.ASS_PROCESSOR do
1855             begin for I:= DESC.NR_OF_DIMS step -1 until 1 do
1856               begin INDEX(I):=ACTIVE_ACB.INT_TOP.WERT;
1857                 if INDEX(I)<DESC.LBOUND(I) or INDEX(I)>DESC.UBOUND(I)
1858                   then begin RUNTIME_ERROR
1859                     ("ARRAY BOUNDS ERROR. UBOUND USED.");
1860                     INDEX(I):=DESC.UBOUND(I);
1861                   end;
1862                 INT_POP;
1863               end;
1864               PUSH_ELEMENT(INDEX(1),INDEX(2),INDEX(3),INDEX(4),INDEX(5));
1865             end;
1866             procedure ASSIGN;
1867             inspect Current qua MODULE.ASS_PROCESSOR do
1868             begin SAVE_NEW_VALUE;
1869               for I:= DESC.NR_OF_DIMS step -1 until 1 do
1870                 begin INDEX(I):=ACTIVE_ACB.INT_TOP.WERT;
1871                   if INDEX(I)<DESC.LBOUND(I) or INDEX(I)>DESC.UBOUND(I)
1872                     then begin RUNTIME_ERROR
1873                       ("ARRAY BOUNDS ERROR. UBOUND USED.");
1874                       INDEX(I):=DESC.UBOUND(I);
1875                     end;
1876                   INT_POP;
1877                 end;
1878                 if ACTIVE_ACB.TRACE then
1879                   begin PRINT_INDICES
1880                     (INDEX(1),INDEX(2),INDEX(3),INDEX(4),INDEX(5));
1881                     Outtext(" := ");
1882                   end;
1883                   ASSIGN_ELEMENT(INDEX(1),INDEX(2),INDEX(3),INDEX(4),INDEX(5))
1884                 end;
1885             procedure DUMP(IDNAME);text IDNAME;
1886             begin for I1:=DESC.LBOUND(1) step 1 until DESC.UBOUND(1) do
1887               for I2:=DESC.LBOUND(2) step 1 until DESC.UBOUND(2) do
1888               for I3:=DESC.LBOUND(3) step 1 until DESC.UBOUND(3) do
1889               for I4:=DESC.LBOUND(4) step 1 until DESC.UBOUND(4) do
1890               for I5:=DESC.LBOUND(5) step 1 until DESC.UBOUND(5) do
1891               begin BMSOUT.Setpos(16);Outtext(" | ");Outtext(IDNAME);
1892                 PRINT_INDICES(I1,I2,I3,I4,I5);

```

```
1893           Outtext(" = ");
1894           PRINT(I1,I2,I3,I4,I5);
1895           BMSOUT_Setpos(50);
1896           Outtext(" | ") ;
1897           OUTLINE;
1898       end;                                         E235
1899   end;
1900 procedure PRINT_INDICES(I1,I2,I3,I4,I5);integer I1,I2,I3,I4,I5;
1901 begin Outtext("(");
1902     PRINT_INT(I1);
1903     if DESC.NR_OF_DIMS>1
1904         then begin Outtext(",");PRINT_INT(I2);end;
1905     if DESC.NR_OF_DIMS>2
1906         then begin Outtext(",");PRINT_INT(I3);end;
1907     if DESC.NR_OF_DIMS>3
1908         then begin Outtext(",");PRINT_INT(I4);end;
1909     if DESC.NR_OF_DIMS>4
1910         then begin Outtext(",");PRINT_INT(I5);end;
1911     Outtext(")");
1912 end;
1913 end;                                         E236
1914
1915 procedure DUMP_ARRAY(IDNAME,A);value IDNAME;text IDNAME;ref(FIELD)A;
1916   inspect A do DUMP(IDNAME);
1917
1918 %ENDCOPY
1919 %COPY EXTERN                                         0  0A
1920
```

```

1922
1923  comment ****
1924  * HIER WERDEN DIE VON CSSA ZUR VERFUEGUNG GESTELLTEN BUILT-IN *
1925  * FUNKTIONEN IN SIMULA DEFINIERT.
1926  ****
1927
1928  integer procedure CSSALENGTH(STRING) ; text STRING ;
1929  CSSALENGTH := STRING.Length ;
1930
1931  text procedure CSSASUBSTR(STRING,LOW,UP);text STRING;integer LOW,UP;
1932  begin
1933    if LOW>STRING.Length or UP<1 then CSSASUBSTR:=-notext else
1934    begin
1935      if LOW<1 then LOW:=1 ;
1936      if UP>STRING.Length then UP:=STRING.Length ;
1937      if LOW>UP then CSSASUBSTR:=-notext
1938      else CSSASUBSTR:=-Copy(STRING.Sub(LOW,UP-LOW+1)) ;
1939    end ;
1940  end CSSASUBSTR ;
1941
1942  text procedure CSSABLANKS(L) ; integer L ;
1943  if L<=0 then CSSABLANKS:=-notext else
1944  CSSABLANKS := Blanks(L) ;
1945
1946  integer procedure CSSAORD(STRING) ; text STRING ;
1947  begin
1948    if STRING.Length=0 then CSSAORD:=0 else
1949    begin
1950      STRING.Setpos(1) ;
1951      CSSAORD := Rank(STRING.Getchar);
1952    end ;
1953  end CSSAORD ;
1954
1955  boolean procedure CSSANUMBER(STRING) ; text STRING ;
1956  begin
1957    integer I,J ; character C ;
1958    if STRING.Length = 0 then goto ENDPROC ;
1959    STRING.Setpos(1) ;
1960    for I:=1 step 1 until STRING.Length do
1961    begin
1962      if Digit(STRING.Getchar) then goto FOUND ;
1963    end ;
1964    goto ENDPROC ;
1965    FOUND:
1966    I:=STRING.Pos ;
1967    if I>2 then
1968    begin
1969      I:=I-2 ; STRING.Setpos(I) ;
1970      C:=STRING.Getchar ;
1971      if C='+' and C='-' and C='.' then goto ENDPROC ;
1972      STRING.Setpos(I) ;
1973      for J:=1 step 1 until I-1 do
1974      begin
1975        if STRING.Getchar ~= ' ' then goto ENDPROC ;
1976      end ;
1977    end ;
1978    CSSANUMBER :=true ;
1979    ENDPROC:
1980  end CSSANUMBER ;
1981
1982  text procedure CSSACHAR(NUM) ; integer NUM ;
1983  begin comment SIEMENS; text T ;
1984    if NUM<0 then NUM := 0 ;
1985    NUM := Mod(NUM,256) ;
1986    T:-Blanks(1) ; T.Putchar(Char(NUM));
1987    CSSACHAR:=T ;
1988  end CSSACHAR ;
1989
1990  integer procedure CSSAVALUE(STRING) ; text STRING ;
1991  begin
1992    if~CSSANUMBER(STRING) then CSSAVALUE :=0 else
1993    begin
1994      STRING.Setpos(1) ; CSSAVALUE := STRING.Getint ;
1995    end ;
1996  end CSSAVALUE ;
1997
1998  text procedure CSSAGENSTRING(NUM) ; integer NUM ;
1999  begin
2000    text T ; integer I ;
2001    T:-Blanks(10) ;
2002    T.Putint(NUM) ;
2003    T.Setpos(1) ;
2004    for I:=1 step 1 until 10 do
2005      if T.Getchar ~= ' ' then goto FOUND ;
2006    FOUND:
2007      CSSAGENSTRING := Copy(T.Sub(I,11-I));

```

```

2008    end CSSAGENSTRING ;
2009
2010    integer procedure CSSAPOS(ENUM) ; integer ENUM ;
2011    CSSAPOS := ENUM ;
2012
2013    integer procedure CSSASUCC(ENUM) ; integer ENUM ;
2014    CSSASUCC := ENUM+1 ;
2015
2016    integer procedure CSSAPRED(ENUM) ; integer ENUM ;
2017    if ENUM>0 then CSSAPRED := ENUM-1 ;
2018
2019    long real procedure CSSAFLOAT(I) ; integer I ;
2020    CSSAFLOAT := I ;
2021
2022    integer procedure CSSAROUND(R) ; long real R ;
2023    begin
2024        if R > 2147483647.0 or R < -2147483648.0 then
2025            RUNTIME_ERROR("OVERFLOW IN BUILT-IN FUNCTION ROUND") else
2026            CSSAROUND := R ;
2027    end ;
2028
2029    procedure CSSATIME(T); real T;
2030        inspect Current qua MODULE do ELAPSED_TIME:=ELAPSED_TIME+T;
2031
2032    boolean procedure CSSAEMPTY(SET_OR_REL);ref(Head) SET_OR_REL;
2033        CSSAEMPTY:=SET_OR_REL.Empty;
2034
2035
2036    integer procedure CSSACARDINAL(SET_OR_REL);ref(Head) SET_OR_REL;
2037        CSSACARDINAL:=SET_OR_REL.Cardinal;
2038
2039
2040    procedure RUNTIME_ERROR(T); value T; text T;
2041    begin OUTLINE;Outtext("XXX");PRINT_TIME;Outtext(" ");
2042        PRINT_AGENT(Current qua MODULE.ASS_PROCESSOR.ACTIVE_ACB);
2043        Outtext(": "); Outtext(T);OUTLINE;
2044        BMSOUT.Setpos(16);Outtext("AT LINE ");
2045        Outint(Current qua MODULE.ASS_PROCESSOR.ACTIVE_ACB.
2046        CURRENT_LINE,4);OUTLINE;
2047        if Current qua MODULE.ASS_PROCESSOR.ACTIVE_ACB/=INTERFACE
2048        then PRINT_DUMP(Current qua MODULE.ASS_PROCESSOR.ACTIVE_ACB);
2049    end;
2050
2051    integer procedure CSSAABS(I) ; integer I ;
2052    CSSAABS := Abs(I);
2053
2054    long real procedure CSSASIN(R) ; long real R ;
2055    CSSASIN := Sin(R);
2056
2057    long real procedure CSSACOS(R) ; long real R ;
2058    CSSACOS := Cos(R);
2059
2060    long real procedure CSSATAN(R) ; long real R ;
2061    CSSATAN := Tan(R);
2062
2063    long real procedure CSSALN(R) ; long real R ;
2064    begin
2065        if R <= 0.0 then RUNTIME_ERROR("ILLEGAL ARGUMENT TO FUNCTION LN")
2066        else CSSALN:= Ln(R);
2067    end ;
2068
2069    long real procedure CSSASQRT(R) ; long real R ;
2070    begin
2071        if R <0.0 then RUNTIME_ERROR("ILLEGAL ARGUMENT TO FUNCTION SQRT")
2072        else CSSASQRT:= Sqrt(R);
2073    end ;
2074
2075    integer procedure CSSAENTIER(R) ; long real R ;
2076    begin
2077        if R > 2147483647.0 or R < -2147483648.0 then
2078            RUNTIME_ERROR("OVERFLOW IN BUILT-IN FUNCTION ENTIER") else
2079            CSSAENTIER := Entier(R) ;
2080    end ;
2081
2082    integer procedure CSSAMOD(I,J) ; integer I,J ;
2083    begin
2084        if J=0 then
2085            begin
2086                RUNTIME_ERROR("ZERODIVIDE IN BUILT-IN FUNCTION MOD") ;
2087            end else
2088                CSSAMOD := Mod(I,J) ;
2089    end ;
2090
2091    %ENDCOPY
2092    %COPY INTARR

```

```
2094 FIELD class INT_ARRAY;
2095 begin integer array DATA(DESC.LBOUND(1):DESC.UBOUND(1)
2096 ,DESC.LBOUND(2):DESC.UBOUND(2)
2097 ,DESC.LBOUND(3):DESC.UBOUND(3)
2098 ,DESC.LBOUND(4):DESC.UBOUND(4)
2099 ,DESC.LBOUND(5):DESC.UBOUND(5));
2100 integer NEW_VALUE;
2101
2102 procedure SAVE_NEW_VALUE;
2103 inspect Current qua MODULE.ASS_PROCESSOR do
2104 begin NEW_VALUE:=ACTIVE_ACB.INT_TOP.WERT;INT_POP;end; B260 B261 E261
2105
2106 procedure READ_PARM;
2107 inspect Current qua MODULE.ASS_PROCESSOR do
2108 begin NEW_VALUE:=ACTIVE_ACB.CURRENT_PARM.INTWERT;end; B262 E262
2109
2110 procedure PUSH_ELEMENT(I1,I2,I3,I4,I5);
2111     integer I1,I2,I3,I4,I5;
2112 inspect Current qua MODULE.ASS_PROCESSOR do
2113     INT_PUSH(DATA(I1,I2,I3,I4,I5));
2114
2115 procedure SET_PARM(I1,I2,I3,I4,I5);
2116     integer I1,I2,I3,I4,I5;
2117 inspect Current qua MODULE.ASS_PROCESSOR do
2118 begin INT_POP;
2119     ACTIVE_ACB.MSG_TOP.WERT.PARMS.Last qua PARAM.INTWERT:=
2120     DATA(I1,I2,I3,I4,I5);
2121 end; B263 E263
2122
2123 procedure ASSIGN_ELEMENT(I1,I2,I3,I4,I5);
2124     integer I1,I2,I3,I4,I5;
2125 inspect Current qua MODULE.ASS_PROCESSOR do
2126 begin if ACTIVE_ACB.TRACE
2127     then begin PRINT_INT(NEW_VALUE);OUTLINE;end;
2128     DATA(I1,I2,I3,I4,I5):=NEW_VALUE;
2129 end; B264 E264
2130 procedure PRINT(I1,I2,I3,I4,I5);
2131     integer I1,I2,I3,I4,I5;
2132     PRINT_INT(DATA(I1,I2,I3,I4,I5));
2133     TYPE_OF_ARRAY:-Copy("INT");
2134 end; E260
2135 %ENDCOPY
2136 %COPY REALARR 0 0A
```

```

2138 FIELD class REAL_ARRAY;
2139 begin long real array DATA(DESC.LBOUND(1):DESC.UBOUND(1)
2140 ,DESC.LBOUND(2):DESC.UBOUND(2)
2141 ,DESC.LBOUND(3):DESC.UBOUND(3)
2142 ,DESC.LBOUND(4):DESC.UBOUND(4)
2143 ,DESC.LBOUND(5):DESC.UBOUND(5));
2144 long real NEW_VALUE;
2145
2146 procedure SAVE_NEW_VALUE;
2147 inspect Current qua MODULE.ASS_PROCESSOR do
2148 begin NEW_VALUE:=ACTIVE_ACB.REAL_TOP.WERT;REAL_POP;end;           B266 E267
2149
2150 procedure READ_PARM;
2151 inspect Current qua MODULE.ASS_PROCESSOR do
2152 begin NEW_VALUE:=ACTIVE_ACB.CURRENT_PARM.REALWERT;end;           B268 E268
2153
2154 procedure PUSH_ELEMENT(I1,I2,I3,I4,I5);
2155     integer I1,I2,I3,I4,I5;
2156 inspect Current qua MODULE.ASS_PROCESSOR do
2157     REAL_PUSH(DATA(I1,I2,I3,I4,I5));
2158
2159 procedure SET_PARM(I1,I2,I3,I4,I5);
2160     integer I1,I2,I3,I4,I5;
2161 inspect Current qua MODULE.ASS_PROCESSOR do
2162 begin REAL_POP;
2163     ACTIVE_ACB.MSG_TOP.WERT.PARMS.Last qua PARAM.REALWERT:=
2164     DATA(I1,I2,I3,I4,I5);                                         B269
2165 end;                                                               E269
2166
2167 procedure ASSIGN_ELEMENT(I1,I2,I3,I4,I5);
2168     integer I1,I2,I3,I4,I5;
2169 inspect Current qua MODULE.ASS_PROCESSOR do
2170 begin if ACTIVE_ACB.TRACE
2171     then begin PRINT_REAL(NEW_VALUE);OUTLINE;end;
2172     DATA(I1,I2,I3,I4,I5):=NEW_VALUE;                               B270 B271 E271
2173 end;
2174 procedure PRINT(I1,I2,I3,I4,I5);
2175     integer I1,I2,I3,I4,I5;
2176     PRINT_REAL(DATA(I1,I2,I3,I4,I5));
2177     TYPE_OF_ARRAY:-Copy("REAL");
2178 end;                                                               E266
2179 %ENDCOPY
2180 %COPY AGENTARR

```

0 0A

```
2182 FIELD class AGENT_ARRAY;
2183 begin ref(ACB)array DATA(DESC.LBOUND(1):DESC.UBOUND(1)
2184 ,DESC.LBOUND(2):DESC.UBOUND(2)
2185 ,DESC.LBOUND(3):DESC.UBOUND(3)
2186 ,DESC.LBOUND(4):DESC.UBOUND(4)
2187 ,DESC.LBOUND(5):DESC.UBOUND(5));
2188 ref(ACB)NEW_VALUE;
2189
2190 procedure SAVE_NEW_VALUE;
2191 inspect Current qua MODULE.ASS_PROCESSOR do
2192 begin NEW_VALUE:=ACTIVE_ACB.AGENT_TOP.WERT;AGENT_POP;end; B272 B273 E273
2193
2194 procedure READ_PARM;
2195 inspect Current qua MODULE.ASS_PROCESSOR do
2196 begin NEW_VALUE:=ACTIVE_ACB.CURRENT_PARM.AGENTWERT;end; B274 E274
2197
2198 procedure PUSH_ELEMENT(I1,I2,I3,I4,I5);
2199 integer I1,I2,I3,I4,I5;
2200 inspect Current qua MODULE.ASS_PROCESSOR do
2201 begin AGENT_PUSH(DATA(I1,I2,I3,I4,I5));
2202
2203 procedure SET_PARM(I1,I2,I3,I4,I5);
2204 integer I1,I2,I3,I4,I5;
2205 inspect Current qua MODULE.ASS_PROCESSOR do
2206 begin AGENT_POP; B275
2207 ACTIVE_ACB.MSG_TOP.WERT.PARMS.Last qua PARAM.AGENTWERT:-
2208 DATA(I1,I2,I3,I4,I5);
2209 end; E275
2210
2211 procedure ASSIGN_ELEMENT(I1,I2,I3,I4,I5);
2212 integer I1,I2,I3,I4,I5;
2213 inspect Current qua MODULE.ASS_PROCESSOR do
2214 begin if ACTIVE_ACB.TRACE
2215 then begin PRINT_AGENT(NEW_VALUE);OUTLINE;end; B276 B277 E277
2216 DATA(I1,I2,I3,I4,I5):=NEW_VALUE;
2217 end; E276
2218 procedure PRINT(I1,I2,I3,I4,I5);
2219 integer I1,I2,I3,I4,I5;
2220 begin PRINT_AGENT(DATA(I1,I2,I3,I4,I5));
2221 TYPE_OF_ARRAY:=Copy("AGENT");
2222 end; E272
2223 %ENDCOPY
2224 %COPY BOOLARR 0 0A
```

```

2226  FIELD class BOOL_ARRAY;
2227  begin boolean array DATA(DESC.LBOUND(1):DESC.UBOUND(1)
2228          ,DESC.LBOUND(2):DESC.UBOUND(2)
2229          ,DESC.LBOUND(3):DESC.UBOUND(3)
2230          ,DESC.LBOUND(4):DESC.UBOUND(4)
2231          ,DESC.LBOUND(5):DESC.UBOUND(5));
2232  boolean NEW_VALUE;
2233
2234  procedure SAVE_NEW_VALUE;
2235  inspect Current qua MODULE.ASS_PROCESSOR do
2236  begin NEW_VALUE:=ACTIVE_ACB.BOOL_TOP.WERT;BOOL_POP;end;           B278 E279
2237
2238  procedure READ_PARM;
2239  inspect Current qua MODULE.ASS_PROCESSOR do
2240  begin NEW_VALUE:=ACTIVE_ACB.CURRENT_PARM.BOOLWERT;end;           B280 E280
2241
2242  procedure PUSH_ELEMENT(I1,I2,I3,I4,I5);
2243  integer I1,I2,I3,I4,I5;
2244  inspect Current qua MODULE.ASS_PROCESSOR do
2245  BOOL_PUSH(DATA(I1,I2,I3,I4,I5));
2246
2247  procedure SET_PARM(I1,I2,I3,I4,I5);
2248  integer I1,I2,I3,I4,I5;
2249  inspect Current qua MODULE.ASS_PROCESSOR do
2250  begin BOOL_POP;
2251      ACTIVE_ACB.MSG_TOP.WERT.PARMS.Last qua PARAM.BOOLWERT:=
2252      DATA(I1,I2,I3,I4,I5);                                         B281
2253  end;                                                               E281
2254
2255  procedure ASSIGN_ELEMENT(I1,I2,I3,I4,I5);
2256  integer I1,I2,I3,I4,I5;
2257  inspect Current qua MODULE.ASS_PROCESSOR do
2258  begin if ACTIVE_ACB.TRACE
2259      then begin PRINT_BOOL(NEW_VALUE);OUTLINE;end;
2260      DATA(I1,I2,I3,I4,I5):=NEW_VALUE;                                B282
2261  end;                                                               E282
2262  procedure PRINT(I1,I2,I3,I4,I5);
2263  integer I1,I2,I3,I4,I5;
2264  PRINT_BOOL(DATA(I1,I2,I3,I4,I5));
2265  TYPE_OF_ARRAY:-Copy("BOOL");
2266
2267 %ENDCOPY
2268 %COPY STRNGARR

```

0 0A

```

2270 FIELD class STRING_ARRAY;
2271 begin text array DATA(DESC.LBOUND(1):DESC.UBOUND(1)
2272 ,DESC.LBOUND(2):DESC.UBOUND(2)
2273 ,DESC.LBOUND(3):DESC.UBOUND(3)
2274 ,DESC.LBOUND(4):DESC.UBOUND(4)
2275 ,DESC.LBOUND(5):DESC.UBOUND(5));
2276     text NEW_VALUE;
2277
2278 procedure SAVE_NEW_VALUE;
2279 inspect Current qua MODULE.ASS_PROCESSOR do
2280     begin NEW_VALUE:=ACTIVE_ACB.STRING_TOP.WERT;STRING_POP;end;          B284 E285
2281
2282 procedure READ_PARM;
2283 inspect Current qua MODULE.ASS_PROCESSOR do
2284     begin NEW_VALUE:=ACTIVE_ACB.CURRENT_PARM.STRINGWERT;end;            B286 E286
2285
2286 procedure PUSH_ELEMENT(I1,I2,I3,I4,I5);
2287     integer I1,I2,I3,I4,I5;
2288 inspect Current qua MODULE.ASS_PROCESSOR do
2289     STRING_PUSH(DATA(I1,I2,I3,I4,I5));
2290
2291 procedure SET_PARM(I1,I2,I3,I4,I5);
2292     integer I1,I2,I3,I4,I5;
2293 inspect Current qua MODULE.ASS_PROCESSOR do
2294     begin STRING_POP;
2295         ACTIVE_ACB.MSG_TOP.WERT.PARMS.Last qua PARAM.STRINGWERT:-
2296             DATA(I1,I2,I3,I4,I5);                                         B287
2297     end;                                                               E287
2298
2299 procedure ASSIGN_ELEMENT(I1,I2,I3,I4,I5);
2300     integer I1,I2,I3,I4,I5;
2301 inspect Current qua MODULE.ASS_PROCESSOR do
2302     begin if ACTIVE_ACB.TRACE
2303         then begin PRINT_STRING(NEW_VALUE);OUTLINE;end;
2304             DATA(I1,I2,I3,I4,I5):=NEW_VALUE;                            B288 E289
2305     end;
2306 procedure PRINT(I1,I2,I3,I4,I5);
2307     integer I1,I2,I3,I4,I5;
2308     PRINT_STRING(DATA(I1,I2,I3,I4,I5));
2309     TYPE_OF_ARRAY:=Copy("STRING");
2310 end;                                                               E284
2311 %ENDCOPY
2312 %COPY ENUMARR

```

0 0A

```
2314 FIELD class ENUM_ARRAY;
2315 begin integer array DATA(DESC.LBOUND(1):DESC.UBOUND(1)
2316 ,DESC.LBOUND(2):DESC.UBOUND(2)
2317 ,DESC.LBOUND(3):DESC.UBOUND(3)
2318 ,DESC.LBOUND(4):DESC.UBOUND(4)
2319 ,DESC.LBOUND(5):DESC.UBOUND(5));
2320 integer NEW_VALUE;
2321
2322 procedure SAVE_NEW_VALUE;
2323 inspect Current qua MODULE.ASS_PROCESSOR do
2324 begin NEW_VALUE:=ACTIVE_ACB.ENUM_TOP.WERT;ENUM_POP;end; B290 B291 E291
2325
2326 procedure READ_PARM;
2327 begin comment ENUM NICHT VERSCHICKBAR; end; B292 E292
2328
2329 procedure PUSH_ELEMENT(I1,I2,I3,I4,I5);
2330     integer I1,I2,I3,I4,I5;
2331 inspect Current qua MODULE.ASS_PROCESSOR do
2332     ENUM_PUSH(DATA(I1,I2,I3,I4,I5));
2333
2334 procedure SET_PARM(I1,I2,I3,I4,I5);
2335     integer I1,I2,I3,I4,I5;
2336 begin comment ENUM NICHT VERSCHICKBAR; end; B293 E293
2337
2338 procedure ASSIGN_ELEMENT(I1,I2,I3,I4,I5);
2339     integer I1,I2,I3,I4,I5;
2340 inspect Current qua MODULE.ASS_PROCESSOR do
2341 begin if ACTIVE_ACB.TRACE
2342     then begin PRINT_ENUM(NEW_VALUE);OUTLINE;end;
2343     DATA(I1,I2,I3,I4,I5):=NEW_VALUE;
2344 end; B294
2345 procedure PRINT(I1,I2,I3,I4,I5);
2346     integer I1,I2,I3,I4,I5;
2347     PRINT_ENUM(DATA(I1,I2,I3,I4,I5));
2348     TYPE_OF_ARRAY:-Copy("ENUM");
2349 end; E290
2350 %ENDCOPY
2351 %COPY OPERARR 0 0A
```

```

2353 FIELD class OPER_ARRAY;
2354 begin text array DATA(DESC.LBOUND(1):DESC.UBOUND(1)
2355 ,DESC.LBOUND(2):DESC.UBOUND(2)
2356 ,DESC.LBOUND(3):DESC.UBOUND(3)
2357 ,DESC.LBOUND(4):DESC.UBOUND(4)
2358 ,DESC.LBOUND(5):DESC.UBOUND(5));
2359 text NEW_VALUE;
2360
2361 procedure SAVE_NEW_VALUE;
2362 inspect Current qua MODULE.ASS_PROCESSOR do
2363 begin NEW_VALUE:=-ACTIVE_ACB.OPER_TOP.WERT;OPER_POP;end;           B296 E297
2364
2365 procedure READ_PARM;
2366 inspect Current qua MODULE.ASS_PROCESSOR do
2367 begin NEW_VALUE:=-ACTIVE_ACB.CURRENT_PARM.OPERWERT;end;             B298 E298
2368
2369 procedure PUSH_ELEMENT(I1,I2,I3,I4,I5);
2370     integer I1,I2,I3,I4,I5;
2371 inspect Current qua MODULE.ASS_PROCESSOR do
2372     OPER_PUSH(DATA(I1,I2,I3,I4,I5));
2373
2374 procedure SET_PARM(I1,I2,I3,I4,I5);
2375     integer I1,I2,I3,I4,I5;
2376 inspect Current qua MODULE.ASS_PROCESSOR do
2377 begin OPER_POP;                                                 B299
2378     ACTIVE_ACB.MSG_TOP.WERT.PARMS.Last qua PARAM.OPERWERT:-
2379     DATA(I1,I2,I3,I4,I5);
2380 end;                                                       E299
2381
2382 procedure ASSIGN_ELEMENT(I1,I2,I3,I4,I5);
2383     integer I1,I2,I3,I4,I5;
2384 inspect Current qua MODULE.ASS_PROCESSOR do
2385 begin if ACTIVE_ACB.TRACE
2386     then begin PRINT_OPER(NEW_VALUE);OUTLINE;end;
2387     DATA(I1,I2,I3,I4,I5):=NEW_VALUE;                           B300 B301 E301
2388 end;                                                       E300
2389 procedure PRINT(I1,I2,I3,I4,I5);
2390     integer I1,I2,I3,I4,I5;
2391     PRINT_OPER(DATA(I1,I2,I3,I4,I5));
2392     TYPE_OF_ARRAY:=Copy("OPER");
2393 end;                                                       E296
2394 %ENDCOPY
2395 %COPY SCRPTARR                                         0 0A

```

```

2397  FIELD class SCRIPT_ARRAY;
2398  begin integer array DATA(DESC.LBOUND(1):DESC.UBOUND(1)
2399          ,DESC.LBOUND(2):DESC.UBOUND(2)
2400          ,DESC.LBOUND(3):DESC.UBOUND(3)
2401          ,DESC.LBOUND(4):DESC.UBOUND(4)
2402          ,DESC.LBOUND(5):DESC.UBOUND(5));
2403  integer NEW_VALUE;
2404
2405  procedure SAVE_NEW_VALUE;
2406  inspect Current qua MODULE.ASS_PROCESSOR do
2407    begin NEW_VALUE:=ACTIVE_ACB.SCRIPT_TOP.WERT;SCRIPT_POP;end;           B302
2408
2409  procedure READ_PARM;
2410  inspect Current qua MODULE.ASS_PROCESSOR do
2411    begin NEW_VALUE:=ACTIVE_ACB.CURRENT_PARM.SCRIPTWERT;end;             B303 E303
2412
2413  procedure PUSH_ELEMENT(I1,I2,I3,I4,I5);
2414    integer I1,I2,I3,I4,I5;
2415  inspect Current qua MODULE.ASS_PROCESSOR do
2416    SCRIPT_PUSH(DATA(I1,I2,I3,I4,I5));
2417
2418  procedure SET_PARM(I1,I2,I3,I4,I5);
2419    integer I1,I2,I3,I4,I5;
2420  inspect Current qua MODULE.ASS_PROCESSOR do
2421    begin SCRIPT_POP;                                              B305
2422      ACTIVE_ACB.MSG_TOP.WERT.PARMS.Last qua PARAM.SCRIPTWERT:=
2423      DATA(I1,I2,I3,I4,I5);
2424    end;                                                       E305
2425
2426  procedure ASSIGN_ELEMENT(I1,I2,I3,I4,I5);
2427    integer I1,I2,I3,I4,I5;
2428  inspect Current qua MODULE.ASS_PROCESSOR do
2429    begin if ACTIVE_ACB.TRACE
2430      then begin PRINT_SCRIPT(NEW_VALUE);OUTLINE;end;                  B306
2431      DATA(I1,I2,I3,I4,I5):=NEW_VALUE;
2432    end;                                                       E306
2433  procedure PRINT(I1,I2,I3,I4,I5);
2434    integer I1,I2,I3,I4,I5;
2435    PRINT_SCRIPT(DATA(I1,I2,I3,I4,I5));
2436    TYPE_OF_ARRAY:-Copy("SCRIPT");
2437  end;                                                       E302
2438 %ENDCOPY
2439 %COPY INTSET

```

0 0A

```

2441  Head class INT_SET;
2442  begin procedure PUT;
2443    inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2444    begin Locate;
2445      if CURR==none then
2446        begin new INT_ELEMENT(INT_TOP.WERT).Into(this INT_SET);
2447          if TRACE then
2448            begin PRINT_LINENO;PRINT_INT(INT_TOP.WERT);
2449              Outtext(" PUT INTO A SET ");OUTLINE;
2450            end;
2451          end;
2452        INT_POP;
2453      end;
2454    procedure REMOVE;
2455    inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2456    begin Locate;
2457      if CURR=/=none then
2458        begin if CURR==LOOP_INDEX then LOOP_INDEX:=LOOP_INDEX.Pred;
2459          CURR.Out;
2460          if TRACE then
2461            begin PRINT_LINENO;PRINT_INT(INT_TOP.WERT);
2462              Outtext(" REMOVED FROM A SET ");OUTLINE;
2463            end;
2464          end;
2465        INT_POP;
2466      end;
2467    procedure TEST;
2468    inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2469    begin Locate;INT_POP;BOOL_PUSH(CURR=/=none); end;
2470  procedure Locate;
2471  inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2472  begin CURR:-First;
2473    while CURR=/=none do
2474      begin if CURR.WERT=INT_TOP.WERT then goto FOUND;
2475        CURR:-CURR.Suc;
2476      end;
2477      FOUND :
2478    end;
2479  procedure DUMP(N); value N; text N;
2480  begin ref(INT_ELEMENT) X;
2481    BMSOUT.Setpos(16);Outtext("| ");Outtext(N);Outchar(':'');
2482    BMSOUT.Setpos(50);Outtext(" | ");
2483    OUTLINE;
2484    X:-First;
2485    while X=/=none do
2486      begin BMSOUT.Setpos(16);Outtext("| ..... ");
2487        PRINT_INT(X.WERT);
2488        BMSOUT.Setpos(50);Outtext(" | ");
2489        OUTLINE;
2490        X:-X.Suc;
2491      end;
2492    end;
2493  procedure PUSH_ALL;
2494  inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2495  begin CURR:-First;SET_SIZE:=0;
2496    while CURR=/=none do
2497      begin INT_PUSH(CURR.WERT);SET_SIZE:=SET_SIZE+1;
2498        CURR:-CURR.Suc;
2499      end;
2500    INT_PUSH(SET_SIZE);
2501  end;
2502  procedure ASSIGN_ALL;
2503  inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2504  begin integer I;
2505    Clear;
2506    SET_SIZE:=INT_TOP.WERT;INT_POP;
2507    for I:=1 step 1 until SET_SIZE do
2508      begin new INT_ELEMENT(INT_TOP.WERT).Into(this INT_SET);
2509        INT_POP;
2510      end;
2511    end;
2512  procedure START;
2513  inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2514  begin LOOP_INDEX:-First;
2515    inspect LOOP_INDEX do INT_PUSH(WERT)
2516      otherwise INT_PUSH(0);
2517    end;
2518  procedure INCREM;
2519  inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2520  begin inspect LOOP_INDEX do LOOP_INDEX:-Suc
2521    otherwise LOOP_INDEX:-First;
2522    inspect LOOP_INDEX do INT_PUSH(WERT)
2523      otherwise INT_PUSH(0);
2524    end;
2525    ref(INT_ELEMENT) CURR,LOOP_INDEX;
2526  end;

```

```
2527
2528 Link class INT_ELEMENT(WERT);integer WERT;;
2529 %ENDCOPY
2530 %COPY REALSET
```

0 0A

```

2532 Head class REAL_SET;
2533 begin procedure PUT;
2534   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2535   begin Locate;
2536     if CURR==none then
2537       begin new REAL_ELEMENT(REAL_TOP.WERT).Into(this REAL_SET);
2538         if TRACE then
2539           begin PRINT_LINENO;PRINT_REAL(REAL_TOP.WERT);
2540             Outtext(" PUT INTO A SET ");OUTLINE;
2541           end;
2542         end;
2543       REAL_POP;
2544     end;
2545   procedure REMOVE;
2546   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2547   begin Locate;
2548     if CURR=/=none then
2549       begin if CURR==LOOP_INDEX then LOOP_INDEX:=LOOP_INDEX.Pred;
2550         CURR.Out;
2551         if TRACE then
2552           begin PRINT_LINENO;PRINT_REAL(REAL_TOP.WERT);
2553             Outtext(" REMOVED FROM A SET ");OUTLINE;
2554           end;
2555         end;
2556       REAL_POP;
2557     end;
2558   procedure TEST;
2559   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2560   begin Locate;REAL_POP;BOOL_PUSH(CURR=/=none); end;
2561   procedure Locate;
2562   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2563   begin CURR:=First;
2564     while CURR=/=none do
2565       begin if CURR.WERT==REAL_TOP.WERT then goto FOUND;
2566         CURR:=CURR.Suc;
2567       end;
2568     FOUND :
2569   end;
2570   procedure DUMP(N); value N; text N;
2571   begin ref(REAL_ELEMENT) X;
2572     BMSOUT.Setpos(16);Outtext("| ");Outtext(N);Outchar(':'');
2573     BMSOUT.Setpos(50);Outtext(" | ");
2574     OUTLINE;
2575     X:=First;
2576     while X=/=none do
2577       begin BMSOUT.Setpos(16);Outtext("| .... ");
2578         PRINT_REAL(X.WERT);
2579         BMSOUT.Setpos(50);Outtext(" | ");
2580         OUTLINE;
2581         X:=X.Suc;
2582       end;
2583     end;
2584   procedure PUSH_ALL;
2585   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2586   begin CURR:=First;SET_SIZE:=0;
2587     while CURR=/=none do
2588       begin REAL_PUSH(CURR.WERT);SET_SIZE:=SET_SIZE+1;
2589         CURR:=CURR.Suc;
2590       end;
2591     INT_PUSH(SET_SIZE);
2592   end;
2593   procedure ASSIGN_ALL;
2594   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2595   begin integer I;
2596     Clear;
2597     SET_SIZE:=INT_TOP.WERT;INT_POP;
2598     for I:=1 step 1 until SET_SIZE do
2599       begin new REAL_ELEMENT(REAL_TOP.WERT).Into(this REAL_SET);
2600         REAL_POP;
2601       end;
2602     end;
2603   procedure START;
2604   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2605   begin LOOP_INDEX:=First;
2606     inspect LOOP_INDEX do REAL_PUSH(WERT)
2607       otherwise REAL_PUSH(0);
2608   end;
2609   procedure INCREM;
2610   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2611   begin inspect LOOP_INDEX do LOOP_INDEX:=Suc
2612     otherwise LOOP_INDEX:=First;
2613     inspect LOOP_INDEX do REAL_PUSH(WERT)
2614       otherwise REAL_PUSH(0);
2615   end;
2616   ref(REAL_ELEMENT) CURR,LOOP_INDEX;
2617 end;

```

```
2618
2619 Link class REAL_ELEMENT(WERT);long real WERT;;
2620 %ENDCOPY
2621 %COPY AGENTSET
```

0 OA

```

2623 Head class AGENT_SET;
2624 begin procedure PUT;
2625   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2626   begin Locate;
2627     if CURR==none then
2628       begin new AGENT_ELEMENT(AGENT_TOP.WERT)
2629         .Into(this AGENT_SET);
2630         if TRACE then
2631           begin PRINT_LINENO;PRINT_AGENT(AGENT_TOP.WERT);
2632             Outtext(" PUT INTO A SET ");OUTLINE;
2633             end;
2634           end;
2635           AGENT_POP;
2636         end;
2637       procedure REMOVE;
2638       inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2639       begin Locate;
2640         if CURR=/=none then
2641           begin if CURR==LOOP_INDEX then LOOP_INDEX:=LOOP_INDEX.Pred;
2642             CURR.Out;
2643             if TRACE then
2644               begin PRINT_LINENO;PRINT_AGENT(AGENT_TOP.WERT);
2645                 Outtext(" REMOVED FROM A SET ");OUTLINE;
2646                 end;
2647               end;
2648               AGENT_POP;
2649             end;
2650   procedure DUMP(N); value N; text N;
2651   begin ref(AGENT_ELEMENT) X;
2652     BMSOUT.Setpos(16);Outtext("| ");
2653     BMSOUT.Setpos(50);Outtext(" | ");
2654     OUTLINE;
2655     X:-First;
2656     while X=/=none do
2657       begin BMSOUT.Setpos(16);Outtext("| ..... ");
2658         PRINT_AGENT(X.WERT);
2659         BMSOUT.Setpos(50);Outtext(" | ");
2660         OUTLINE;
2661         X:-X.Suc;
2662       end;
2663     end;
2664   procedure TEST;
2665   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2666   begin Locate;AGENT_POP;BOOL_PUSH(CURR=/=none); end;
2667   procedure Locate;
2668   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2669   begin CURR:-First;
2670     while CURR=/=none do
2671       begin if CURR.WERT==AGENT_TOP.WERT then goto FOUND;
2672         CURR:=CURR.Suc;
2673       end;
2674       FOUND :
2675     end;
2676   procedure PUSH_ALL;
2677   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2678   begin CURR:-First;SET_SIZE:=0;
2679     while CURR=/=none do
2680       begin AGENT_PUSH(CURR.WERT);SET_SIZE:=SET_SIZE+1;
2681         CURR:=CURR.Suc;
2682       end;
2683       INT_PUSH(SET_SIZE);
2684     end;
2685   procedure ASSIGN_ALL;
2686   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2687   begin integer I;
2688     Clear;
2689     SET_SIZE:=INT_TOP.WERT;INT_POP;
2690     for I:=1 step 1 until SET_SIZE do
2691       begin new AGENT_ELEMENT(AGENT_TOP.WERT)
2692         .Into(this AGENT_SET);
2693         AGENT_POP;
2694       end;
2695     end;
2696   procedure START;
2697   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2698   begin LOOP_INDEX:-First;
2699     inspect LOOP_INDEX do AGENT_PUSH(WERT)
2700       otherwise AGENT_PUSH(none);
2701     end;
2702   procedure INCREM;
2703   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2704   begin inspect LOOP_INDEX do LOOP_INDEX:=Suc
2705     otherwise LOOP_INDEX:=First;
2706     inspect LOOP_INDEX do AGENT_PUSH(WERT)
2707       otherwise AGENT_PUSH(none);
2708     end;

```

```
2709      ref(AGENT_ELEMENT) CURR,LOOP_INDEX;
2710  end;
2711  .
2712 Link class AGENT_ELEMENT(WERT);ref(ACB) WERT;;
2713 %ENDCOPY
2714 %COPY BOOLSET
```

E344

0 0A

```

2716 Head class BOOL_SET;
2717 begin procedure PUT;
2718   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2719     begin Locate;
2720       if CURR==none then
2721         begin new BOOL_ELEMENT(BOOL_TOP.WERT).Into(this BOOL_SET);
2722           if TRACE then
2723             begin PRINT_LINENO;PRINT_BOOL(BOOL_TOP.WERT);
2724               Outtext(" PUT INTO A SET ");OUTLINE;
2725             end;
2726         end;
2727       BOOL_POP;
2728     end;
2729   procedure REMOVE;
2730   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2731     begin Locate;
2732       if CURR=/=none then
2733         begin if CURR==LOOP_INDEX then LOOP_INDEX:=LOOP_INDEX.Pred;
2734           CURR.Out;
2735           if TRACE then
2736             begin PRINT_LINENO;PRINT_BOOL(BOOL_TOP.WERT);
2737               Outtext(" REMOVED FROM A SET ");OUTLINE;
2738             end;
2739         end;
2740       BOOL_POP;
2741     end;
2742   procedure TEST;
2743   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2744     begin Locate;BOOL_POP;BOOL_PUSH(CURR=/=none); end;
2745   procedure Locate;
2746   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2747     begin CURR:=-First;
2748       while CURR=/=none do
2749         begin if CURR.WERT eqv BOOL_TOP.WERT then goto FOUND;
2750           CURR:=-CURR.Suc;
2751         end;
2752       FOUND :
2753     end;
2754   procedure DUMP(N); value N; text N;
2755   begin ref(BOOL_ELEMENT) X;
2756     BMSOUT.Setpos(16);Outtext("| ");Outtext(N);Outchar(':'');
2757     BMSOUT.Setpos(50);Outtext(" | ");
2758     OUTLINE;
2759     X:=-First;
2760     while X=/=none do
2761       begin BMSOUT.Setpos(16);Outtext("| ..... ");
2762         PRINT_BOOL(X.WERT);
2763         BMSOUT.Setpos(50);Outtext(" | ");
2764         OUTLINE;
2765         X:=-X.Suc;
2766       end;
2767     end;
2768   procedure PUSH_ALL;
2769   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2770     begin CURR:=-First;SET_SIZE:=0;
2771       while CURR=/=none do
2772         begin BOOL_PUSH(CURR.WERT);SET_SIZE:=SET_SIZE+1;
2773           CURR:=-CURR.Suc;
2774         end;
2775       INT_PUSH(SET_SIZE);
2776     end;
2777   procedure ASSIGN_ALL;
2778   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2779     begin integer I;
2780       Clear;
2781       SET_SIZE:=INT_TOP.WERT;INT_POP;
2782       for I:=1 step 1 until SET_SIZE do
2783         begin new BOOL_ELEMENT(BOOL_TOP.WERT).Into(this BOOL_SET);
2784           BOOL_POP;
2785         end;
2786     end;
2787   procedure START;
2788   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2789     begin LOOP_INDEX:=-First;
2790       inspect LOOP_INDEX do BOOL_PUSH(WERT)
2791         otherwise BOOL_PUSH(false);
2792     end;
2793   procedure INCREM;
2794   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2795     begin inspect LOOP_INDEX do LOOP_INDEX:=-Suc
2796       otherwise LOOP_INDEX:=-First;
2797       inspect LOOP_INDEX do BOOL_PUSH(WERT)
2798         otherwise BOOL_PUSH(false);
2799     end;
2800   ref(BOOL_ELEMENT) CURR,LOOP_INDEX;
2801 end;

```

SIMULA 67 (VERS.08.00)

BOOL\_SET

22 JUL 1982 19:32:30.09 PAGE 51

2802  
2803 Link class BOOL\_ELEMENT(WERT);boolean WERT;;  
2804 %ENDCOPY  
2805 %COPY STRNGSET

0 0A

```

2807  Head class STRING_SET;
2808  begin procedure PUT;
2809    inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2810    begin Locate;
2811      if CURR==none then
2812        begin new STRING_ELEMENT(STRING_TOP.WERT)
2813          .Into(this STRING_SET);
2814          if TRACE then
2815            begin PRINT_LINENO;PRINT_STRING(STRING_TOP.WERT);
2816              Outtext(" PUT INTO A SET ");OUTLINE;
2817            end;
2818          end;
2819        STRING_POP;
2820      end;
2821    procedure REMOVE;
2822    inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2823    begin Locate;
2824      if CURR=/=none then
2825        begin if CURR==LOOP_INDEX then LOOP_INDEX:=~LOOP_INDEX.Pred;
2826          CURR.Out;
2827          if TRACE then
2828            begin PRINT_LINENO;PRINT_STRING(STRING_TOP.WERT);
2829              Outtext(" REMOVED FROM A SET ");OUTLINE;
2830            end;
2831        end;
2832      STRING_POP;
2833    end;
2834    procedure TEST;
2835    inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2836    begin Locate;STRING_POP;BOOL_PUSH(CURR=/=none); end;
2837  procedure Locate;
2838    inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2839    begin CURR:=-First;
2840      while CURR=/=none do
2841        begin if CURR.WERT==STRING_TOP.WERT then goto FOUND;
2842          CURR:=-CURR.Suc;
2843        end;
2844        FOUND :
2845      end;
2846    procedure DUMP(N); value N; text N;
2847    begin ref(STRING_ELEMENT) X;
2848      BMSOUT.Setpos(16);Outtext("| ");Outtext(N);Outchar(':'');
2849      BMSOUT.Setpos(50);Outtext(" | ");
2850      OUTLINE;
2851      X:=-First;
2852      while X=/=none do
2853        begin BMSOUT.Setpos(16);Outtext("| ..... ");
2854          PRINT_STRING(X.WERT);
2855          BMSOUT.Setpos(50);Outtext(" | ");
2856          OUTLINE;
2857          X:=-X.Suc;
2858        end;
2859      end;
2860    procedure PUSH_ALL;
2861    inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2862    begin CURR:=-First;SET_SIZE:=0;
2863      while CURR=/=none do
2864        begin STRING_PUSH(CURR.WERT);SET_SIZE:=SET_SIZE+1;
2865          CURR:=-CURR.Suc;
2866        end;
2867        INT_PUSH(SET_SIZE);
2868      end;
2869    procedure ASSIGN_ALL;
2870    inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2871    begin integer I;
2872      Clear;
2873      SET_SIZE:=INT_TOP.WERT;INT_POP;
2874      for I:=1 step 1 until SET_SIZE do
2875        begin new STRING_ELEMENT(STRING_TOP.WERT)
2876          .Into(this STRING_SET);
2877          STRING_POP;
2878        end;
2879      end;
2880    procedure START;
2881    inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2882    begin LOOP_INDEX:=-First;
2883      inspect LOOP_INDEX do STRING_PUSH(WERT)
2884        otherwise STRING_PUSH(notext);
2885      end;
2886    procedure INCREM;
2887    inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2888    begin inspect LOOP_INDEX do LOOP_INDEX:=-Suc
2889      otherwise LOOP_INDEX:=-First;
2890      inspect LOOP_INDEX do STRING_PUSH(WERT)
2891        otherwise STRING_PUSH(notext);
2892      end;

```

SIMULA 67 (VERS.08.00) STRING\_SET 22 JUL 1982 19:32:30.09 PAGE 53

```
2893      ref(STRING_ELEMENT) CURR,LOOP_INDEX;
2894  end;
2895
2896 Link class STRING_ELEMENT(WERT);value WERT;text WERT;;
2897 %ENDCOPY
2898 %COPY ENUMSET
```

E380

0 0A

```

2900  Head class ENUM_SET;
2901  begin procedure PUT;
2902    inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2903      begin Locate;
2904        if CURR==none then
2905          begin new ENUM_ELEMENT(ENUM_TOP.WERT).Into(this ENUM_SET);
2906            if TRACE then
2907              begin PRINT_LINENO;PRINT_ENUM(ENUM_TOP.WERT);
2908                Outtext(" PUT INTO A SET ");OUTLINE;
2909              end;
2910            end;
2911            ENUM_POP;
2912        end;
2913      procedure REMOVE;
2914      inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2915      begin Locate;
2916        if CURR=/=none then
2917          begin if CURR==LOOP_INDEX then LOOP_INDEX:=LOOP_INDEX.Pred;
2918            CURR.Out;
2919            if TRACE then
2920              begin PRINT_LINENO;PRINT_ENUM(ENUM_TOP.WERT);
2921                Outtext(" REMOVED FROM A SET ");OUTLINE;
2922              end;
2923            end;
2924            ENUM_POP;
2925        end;
2926      procedure TEST;
2927      inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2928      begin Locate;ENUM_POP;BOOL_PUSH(CURR=/=none); end;
2929      procedure Locate;
2930      inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2931      begin CURR:=First;
2932        while CURR=/=none do
2933          begin if CURR.WERT==ENUM_TOP.WERT then goto FOUND;
2934            CURR:=CURR.Suc;
2935          end;
2936          FOUND :
2937        end;
2938      procedure DUMP(N); value N; text N;
2939      begin ref(ENUM_ELEMENT) X;
2940        BMSOUT.Setpos(16);Outtext("| ");Outtext(N);Outchar(':'');
2941        BMSOUT.Setpos(50);Outtext(" | ");
2942        OUTLINE;
2943        X:=First;
2944        while X=/=none do
2945          begin BMSOUT.Setpos(16);Outtext("| .... ");
2946            PRINT_ENUM(X.WERT);
2947            BMSOUT.Setpos(50);Outtext(" | ");
2948            OUTLINE;
2949            X:=X.Suc;
2950          end;
2951        end;
2952      procedure PUSH_ALL;
2953      inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2954      begin CURR:=First;SET_SIZE:=0;
2955        while CURR=/=none do
2956          begin ENUM_PUSH(CURR.WERT);SET_SIZE:=SET_SIZE+1;
2957            CURR:=CURR.Suc;
2958          end;
2959          INT_PUSH(SET_SIZE);
2960        end;
2961      procedure ASSIGN_ALL;
2962      inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2963      begin integer I;
2964        Clear;
2965        SET_SIZE:=INT_TOP.WERT;INT_POP;
2966        for I:=1 step 1 until SET_SIZE do
2967          begin new ENUM_ELEMENT(ENUM_TOP.WERT).Into(this ENUM_SET);
2968            ENUM_POP;
2969          end;
2970        end;
2971      procedure START;
2972      inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2973      begin LOOP_INDEX:=First;
2974        inspect LOOP_INDEX do ENUM_PUSH(WERT)
2975          otherwise ENUM_PUSH(-1);
2976        end;
2977      procedure INCREM;
2978      inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2979      begin inspect LOOP_INDEX do LOOP_INDEX:=Suc
2980        otherwise LOOP_INDEX:=First;
2981        inspect LOOP_INDEX do ENUM_PUSH(WERT)
2982          otherwise ENUM_PUSH(-1);
2983        end;
2984        ref(ENUM_ELEMENT) CURR,LOOP_INDEX;
2985      end;

```

SIMULA 67 (VERS.08.00)

ENUM\_SET

22 JUL 1982 19:32:30.09 PAGE 55

2986  
2987 Link class ENUM\_ELEMENT(WERT);integer WERT;;  
2988 %ENDCOPY  
2989 %COPY OPERSET

0 0A

```

2991 Head class OPER_SET;
2992 begin procedure PUT;
2993   inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
2994     begin Locate;
2995       if CURR==none then
2996         begin new OPER_ELEMENT(OPER_TOP.WERT)
2997           .Into(this OPER_SET);
2998           if TRACE then
2999             begin PRINT_LINENO;PRINT_OPER(OPER_TOP.WERT);
3000               Outtext(" PUT INTO A SET ");OUTLINE;
3001             end;
3002           end;
3003         OPER_POP;
3004       end;
3005     procedure REMOVE;
3006     inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
3007       begin Locate;
3008         if CURR=/=none then
3009           begin if CURR==LOOP_INDEX then LOOP_INDEX:=LOOP_INDEX.Pred;
3010             CURR.Out;
3011             if TRACE then
3012               begin PRINT_LINENO;PRINT_OPER(OPER_TOP.WERT);
3013                 Outtext(" REMOVED FROM A SET ");OUTLINE;
3014               end;
3015             end;
3016           OPER_POP;
3017         end;
3018     procedure TEST;
3019     inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
3020     begin Locate;OPER_POP;BOOL_PUSH(CURR=/=none); end;
3021   procedure Locate;
3022     inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
3023     begin CURR:-First;
3024       while CURR=/=none do
3025         begin if CURR.WERT=OPER_TOP.WERT then goto FOUND;
3026           CURR:=CURR.Suc;
3027         end;
3028         FOUND :
3029       end;
3030     procedure DUMP(N); value N; text N;
3031     begin ref(OPER_ELEMENT) X;
3032       BMSOUT.Setpos(16);Outtext("| ");Outtext(N);Outchar(':'');
3033       BMSOUT.Setpos(50);Outtext(" | ");
3034       OUTLINE;
3035       X:-First;
3036       while X=/=none do
3037         begin BMSOUT.Setpos(16);Outtext("| ..... ");
3038           PRINT_OPER(X.WERT);
3039           BMSOUT.Setpos(50);Outtext(" | ");
3040           OUTLINE;
3041           X:=X.Suc;
3042         end;
3043       end;
3044     procedure PUSH_ALL;
3045     inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
3046     begin CURR:-First;SET_SIZE:=0;
3047       while CURR=/=none do
3048         begin OPER_PUSH(CURR.WERT);SET_SIZE:=SET_SIZE+1;
3049           CURR:=CURR.Suc;
3050         end;
3051         INT_PUSH(SET_SIZE);
3052       end;
3053     procedure ASSIGN_ALL;
3054     inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
3055     begin integer I;
3056       Clear;
3057       SET_SIZE:=INT_TOP.WERT;INT_POP;
3058       for I:=1 step 1 until SET_SIZE do
3059         begin new OPER_ELEMENT(OPER_TOP.WERT)
3060           .Into(this OPER_SET);
3061           OPER_POP;
3062         end;
3063       end;
3064     procedure START;
3065     inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
3066     begin LOOP_INDEX:-First;
3067       inspect LOOP_INDEX do OPER_PUSH(WERT)
3068         otherwise OPER_PUSH(notext);
3069       end;
3070     procedure INCREM;
3071     inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
3072     begin inspect LOOP_INDEX do LOOP_INDEX:=-Suc
3073       otherwise LOOP_INDEX:-First;
3074       inspect LOOP_INDEX do OPER_PUSH(WERT)
3075         otherwise OPER_PUSH(notext);
3076     end;

```

```
3077      ref(OPER_ELEMENT) Curr,Loop_Index;
3078  end;
3079
3080 Link class OPER_ELEMENT(WERT);value WERT;text WERT;;
3081 %ENDCOPY
3082 %COPY SCRPTSET
```

E416

0 0A

```

3084  Head class SCRIPT_SET;
3085  begin procedure PUT;
3086      inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
3087      begin Locate;
3088          if CURR==none then
3089              begin new SCRIPT_ELEMENT(SCRIPT_TOP.WERT)
3090                  .Into(this SCRIPT_SET);
3091                  if TRACE then
3092                      begin PRINT_LINENO;PRINT_SCRIPT(SCRIPT_TOP.WERT);
3093                          Outtext(" PUT INTO A SET ");OUTLINE;
3094                      end;
3095                  end;
3096                  SCRIPT_POP;
3097              end;
3098          procedure REMOVE;
3099          inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
3100          begin Locate;
3101              if CURR=/=none then
3102                  begin if CURR==LOOP_INDEX then LOOP_INDEX:=LOOP_INDEX.Pred;
3103                      CURR.Out;
3104                      if TRACE then
3105                          begin PRINT_LINENO;PRINT_SCRIPT(SCRIPT_TOP.WERT);
3106                              Outtext(" REMOVED FROM A SET ");OUTLINE;
3107                          end;
3108                  end;
3109                  SCRIPT_POP;
3110              end;
3111          procedure TEST;
3112          inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
3113          begin Locate;SCRIPT_POP;BOOL_PUSH(CURR=/=none); end;
3114          procedure Locate;
3115          inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
3116          begin CURR:=First;
3117              while CURR=/=none do
3118                  begin if CURR.WERT==SCRIPT_TOP.WERT then goto FOUND;
3119                      CURR:=CURR.Suc;
3120                  end;
3121                  FOUND :
3122              end;
3123          procedure DUMP(N); value N; text N;
3124          begin ref(SCRIPT_ELEMENT) X;
3125              BMSOUT.Setpos(16);Outtext("| ");Outtext(N);Outchar(':');
3126              BMSOUT.Setpos(50);Outtext(" | ");
3127              OUTLINE;
3128              X:=First;
3129              while X=/=none do
3130                  begin BMSOUT.Setpos(16);Outtext("| .... ");
3131                      PRINT_SCRIPT(X.WERT);
3132                      BMSOUT.Setpos(50);Outtext(" | ");
3133                      OUTLINE;
3134                      X:=X.Suc;
3135                  end;
3136              end;
3137          procedure PUSH_ALL;
3138          inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
3139          begin CURR:=First;SET_SIZE:=0;
3140              while CURR=/=none do
3141                  begin SCRIPT_PUSH(CURR.WERT);SET_SIZE:=SET_SIZE+1;
3142                      CURR:=CURR.Suc;
3143                  end;
3144                  INT_PUSH(SET_SIZE);
3145              end;
3146          procedure ASSIGN_ALL;
3147          inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
3148          begin integer I;
3149              Clear;
3150              SET_SIZE:=INT_TOP.WERT;INT_POP;
3151              for I:=1 step 1 until SET_SIZE do
3152                  begin new SCRIPT_ELEMENT(SCRIPT_TOP.WERT)
3153                      .Into(this SCRIPT_SET);
3154                      SCRIPT_POP;
3155                  end;
3156              end;
3157          procedure START;
3158          inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
3159          begin LOOP_INDEX:=First;
3160              inspect LOOP_INDEX do SCRIPT_PUSH(WERT)
3161                  otherwise SCRIPT_PUSH(0);
3162              end;
3163          procedure INCREM;
3164          inspect Current qua MODULE.ASS_PROCESSOR do inspect ACTIVE_ACB do
3165          begin inspect LOOP_INDEX do LOOP_INDEX:=-Suc
3166                  otherwise LOOP_INDEX:=First;
3167                  inspect LOOP_INDEX do SCRIPT_PUSH(WERT)
3168                      otherwise SCRIPT_PUSH(0);
3169              end;

```

SIMULA 67 (VERS.08.00)

SCRIPT\_SET

22 JUL 1982 19:32:30.09 PAGE 59

```
3170      ref(SCRIPT_ELEMENT) CURR,LOOP_INDEX;
3171  end;
3172
3173 Link class SCRIPT_ELEMENT(WERT);integer WERT;;
3174 %ENDCOPY
3175 %COPY INTERF
```

E434

0 0I

```

3177  comment ****
3178  ** DAS SCRIPT DES INTERFACE-AGENTEN
3179  ****;
3180
3181  MODULE class SCRIPT1;
3182  inspect ASS_PROCESSOR do inspect ACTIVE_ACB do
3183  begin
3184  refInfile) C_INFILE;
3185  ref(PARAM) NEW_PARM;
3186  integer TOKEN;
3187  text INBUFFER,BUFFPOINTER ; comment SIEMENS ;
3188  text SYMBOL;
3189  text EXPONENT ;
3190  boolean RESWORD;
3191
3192  comment ****
3193  ** DER SCANNER LIEST DAS NAECHSTE SYMBOL VOM BILDSCHIRM BZW.    **
3194  ** AUS DER EINGABEDETEI.                                     **
3195  ****;
3196
3197  procedure NEXTSYMB;
3198  begin integer I;
3199
3200  comment ****
3201  ** EINLESEN EINER NEUEN ZEILE                                **
3202  ****;
3203
3204  procedure NEWLINE; comment SIEMENS-AENDERUNGEN ;
3205  inspect C_INFILE do begin
3206    if C_INFILE =/= Sysin then Inimage
3207    else begin external assembly procedure WRTRD;
3208      BUFFPOINTER:=INBUFFER;
3209      BUFFPOINTER:=" ";
3210      WRTRD("*",BUFFPOINTER);
3211      if BUFFPOINTER == notext then Sysin.Image:=" "
3212        else Sysin.Image:=BUFFPOINTER.Sub(1,
3213          if BUFFPOINTER.Length>79 then 79
3214            else BUFFPOINTER.Length);
3215      Sysin.Image.Setpos(1);
3216      end;
3217      if C_INFILE =/= Sysin then C_INFILE.Image.Sub(73,8):=" "      ";
3218      if C_INFILE.Endfile then
3219        begin external assembly procedure WRTRD;
3220          if C_INFILE =/= Sysin then C_INFILE.Close ;
3221          C_INFILE := Sysin ;
3222          BUFFPOINTER:=INBUFFER;
3223          BUFFPOINTER:=" ";
3224          WRTRD("*",BUFFPOINTER);
3225          if BUFFPOINTER == notext then Sysin.Image:=" "
3226            else Sysin.Image:=BUFFPOINTER.Sub(1,
3227              if BUFFPOINTER.Length>79 then 79
3228                else BUFFPOINTER.Length);
3229          Sysin.Image.Setpos(1);
3230          inspect PROT do
3231            begin Outtext("==> ");
3232              Outtext(C_INFILE.Image);
3233              Outtext("<==");
3234              Outimage;
3235            end;
3236            goto START_NEXTSYMB; comment AUS INSPECT HERAUS;
3237          end ;
3238          if C_INFILE=/=Sysin then
3239            begin Sysout.Outtext(C_INFILE.Image.Sub(1,79));comment SIEMENS;
3240              Sysout.Outimage;
3241            end;
3242            inspect PROT do
3243              begin Outtext("==> ");
3244                Outtext(C_INFILE.Image);
3245                Outtext("<==");
3246                Outimage;
3247              end;
3248            end NEWLINE ;
3249
3250  comment ****
3251  ** MELDEN EINES LEXIKALISCHEN FEHLERS                         **
3252  ****;
3253
3254  procedure WARNING(T1,C,T2); value T1,T2; text T1,T2; character C;
3255  begin integer J;
3256    OUTLINE;
3257    Outtext("==> ERROR ") ;
3258    Outtext(" : ") ;
3259    Outtext(T1);Outchar(C);Outtext(T2) ; OUTLINE ;
3260    goto START_NEXTSYMB;
3261  end WARNING ;
3262

```

```

3263  START_NEXTSYMB:
3264  inspect C_INFILE do
3265  begin
3266    TOKENS_PASSED:=0;
3267    LASTSYMB:=-Copy(SYMBOL);
3268    if TOKEN=NEWIDSY or TOKEN=OLDIDSY
3269      then begin TBLREF:=-LASTID;TBLPREV:=-Prev;end;
3270  comment ****
3271  ** LESEN DES NAECHSTEN SYMBOLS
3272  ****
3273
3274  NEXT_SYMBOL:
3275  comment ****
3276  ** EVENTUELLE BENUTZER PROMPTEN UND NAECHSTE ZEILE EINLESEN
3277  ****
3278
3279  if C_INFILE.Pos=80
3280    then begin Outtext(">>>");PRINT_TIME;
3281      Outtext(" "); PRINT_AGENT(SELF);
3282      Outtext(" : ENTER CSSA COMMAND -");
3283      OUTLINE;
3284      NEWLINE ;
3285    end;
3286  comment ****
3287  ** BLANKS UEBERLESEN
3288  ****
3289
3290  NEXTCHAR:=' ';
3291  while NEXTCHAR=' ' do
3292    begin comment ZEILENENDE=SEMIKOLON;
3293      if C_INFILE.Pos=80 then NEXTCHAR:=';';
3294        else NEXTCHAR := Inchar ;
3295      if NEXTCHAR='#' then
3296        begin
3297          NEWLINE ;
3298          NEXTCHAR:=' ';
3299        end;
3300    end;
3301  comment ****
3302  ** KOMMENTARE UEBERLESEN
3303  ****
3304
3305  if NEXTCHAR='('
3306    then begin NEXTCHAR:=Inchar;
3307      if NEXTCHAR='*' then begin character LASTCHAR;
3308        LASTCHAR:=Inchar;NEXTCHAR:=Inchar;
3309        while NEXTCHAR~=')' or LASTCHAR~='*' do
3310          begin LASTCHAR:=NEXTCHAR;
3311            NEXTCHAR:=Inchar;
3312              if C_INFILE.Pos>73 then
3313                begin I:=2;
3314                  WARNING("'*' MISSING.",',','');
3315                  "'*') ASSUMED AT END-OF-LINE");
3316                end;
3317              end;
3318              goto NEXT_SYMBOL ;
3319            end;
3320          end;
3321        else begin NEXTCHAR:='(';
3322          C_INFILE.Setpos(C_INFILE.Pos-1);
3323        end;
3324      end;
3325  comment : END OF SCANNING-ROUTINE ;
3326  RESWORD:=false;
3327
3328  comment ****
3329  ** EINLESEN EINES IDENTIFIERS.
3330  ** ANSCHLIESSEND SYMBOLTABELLENAUFSUCHE, DIE ERGIBT , OB DER
3331  ** BEZEICHNER EIN KEYWORD, EIN BEREITS DEKLARIERTER ODER NEUER
3332  ** IDENTIFIER IST.
3333  ****
3334
3335  if Letter(NEXTCHAR)
3336    then begin I:=0;
3337      while NEXTCHAR='_ ' or NEXTCHAR>='A' do
3338        begin I:=I+1;
3339          NEXTCHAR := Inchar ;
3340        end;
3341      if Rank(NEXTCHAR) >= 129 and Rank(NEXTCHAR) <= 169
3342        then SEM_ERROR("LOWER-CASE LETTERS NOT ALLOWED"
3343                      ,",","");
3344        C_INFILE.Setpos(C_INFILE.Pos-I-1);
3345        SYMBOL:=-Intext(I);
3346        LOOKUP(SYMBOL);
3347      end;
3348    else
3349      B461
3350      B462 E462
3351      B463
3352      E463
3353      B464
3354      B465
3355      E465
3356      E464
3357      B466
3358      B467
3359      B468
3360      B469
3361      E469
3362      E468
3363      E467
3364      B470
3365      E470
3366      E466
3367      B471
3368      B472
3369      E472
3370      E471

```

```

3349  comment ****
3350  ** ZAHLENKONSTANTE EINLESEN
3351  ****
3352
3353  if Digit(NEXTCHAR)
3354    then begin
3355      I:=0;
3356      while Digit(NEXTCHAR) do
3357        begin I:=I+1; NEXTCHAR:=Inchar; end;
3358      TOKEN := INT_CONST;
3359      if NEXTCHAR='.' then
3360        begin
3361          NEXTCHAR:=Inchar;
3362          I:=I+1;
3363          if Digit(NEXTCHAR) then
3364            begin
3365              TOKEN := REAL_CONST;
3366              while Digit(NEXTCHAR) do
3367                begin I:=I+1; NEXTCHAR:=Inchar; end;
3368              end else
3369              begin I:=I-1;
3370                TOKEN := INT_CONST;
3371                C_INFILE.Setpos(C_INFILE.Pos-1);
3372                goto END_OF_NUMBER;
3373              end;
3374            end;
3375            if NEXTCHAR='E' then
3376              begin
3377                EXPONENT := Blanks(3) ;
3378                NEXTCHAR:=Inchar;
3379                I:=I+1;
3380                if NEXTCHAR='+' or NEXTCHAR='-' then begin NEXTCHAR:=Inchar;
3381                  I:=I+1;
3382                  end;
3383                TOKEN:=REAL_CONST;
3384                if ~Digit(NEXTCHAR) then
3385                  begin
3386                    WARNING("ILLEGAL CONSTANT",' ','');
3387                    goto END_OF_NUMBER;
3388                  end;
3389                while Digit(NEXTCHAR) do
3390                  begin
3391                    if EXPONENT.More then EXPONENT.
3392                      Putchar(NEXTCHAR) ;
3393                      I:=I+1; NEXTCHAR:=Inchar;
3394                    end ;
3395                    if ~EXPONENT.More or EXPONENT.Getint>=75 then
3396                      WARNING("EXPONENT OVERFLOW IN REAL CONSTANT:",
3397                        ' ',EXPONENT) ;
3398                    end;
3399                  end;
3400                  END_OF_NUMBER:
3401                  C_INFILE.Setpos(C_INFILE.Pos-I-1);
3402                  SYMBOL:=Intext(I);
3403                  if TOKEN = REAL_CONST then REAL_VALUE := SYMBOL.Getreal
3404                  else INTEGER_VALUE:= SYMBOL.Getint;
3405                  end
3406                else
3407                  comment ****
3408                  ** SONDERZEICHENKOMBINATION EINLESEN
3409                  ****;
3410
3411  begin
3412    LASTID:-none;
3413    I:=1;
3414    if NEXTCHAR=';' then TOKEN:=SEMICLSY
3415    else
3416    if NEXTCHAR=':' then begin NEXTCHAR:=Inchar;I:=I+1;
3417      if NEXTCHAR='=' then TOKEN:=ASSIGNSY
3418      else if NEXTCHAR='-' then TOKEN:=DENOTESY
3419      else begin I:=I-1;
3420        C_INFILE.Setpos(C_INFILE.Pos-1);
3421        TOKEN:=COLONSY;
3422        end;
3423      end;
3424    end
3425  else
3426  if NEXTCHAR='/' then begin NEXTCHAR:=Inchar;I:=I+1;
3427    if NEXTCHAR='*' then TOKEN:=ENDFILESY
3428    else begin I:=I-1;
3429      C_INFILE.Setpos(C_INFILE.Pos-1);
3430      WARNING("ILLEGAL CHARACTER "','",
3431        "IGNORED");
3432    end;
3433  end
3434

```

```
3435      else
3436          if NEXTCHAR=',' then TOKEN:=COMMASY
3437      else
3438          if NEXTCHAR='?' then TOKEN:=QMARKSY
3439      else
3440          if NEXTCHAR='-' then TOKEN:=MINUSSY
3441      else
3442          if NEXTCHAR='(' then TOKEN:=LBRACKSY
3443      else
3444          if NEXTCHAR=')' then TOKEN:=RBRACKSY
3445      else
3446          if NEXTCHAR="''"
3447              then begin TOKEN:=STRING_CONST;I:=I+1;
3448                  while Inchar ~=''' and C_INFILE.Pos<80 do I:=I+1;
3449                  C_INFILE.Setpos(C_INFILE.Pos-1);
3450                  NEXTCHAR:=Inchar;
3451                  if NEXTCHAR="''"
3452                      then begin C_INFILE.Setpos(C_INFILE.Pos-I+1);
3453                          I:=0;
3454                          WARNING("SECOND DOUBLE QUOTE MISSING,");
3455                          ' ','FIRST '''' IGNORED");
3456                  end
3457                  else begin C_INFILE.Setpos(C_INFILE.Pos-I+1);
3458                      STRING_VALUE:=-Intext(I-2);
3459                      Inchar;
3460                  end;
3461              end else
3462                  WARNING("ILLEGAL CHARACTER '',NEXTCHAR,'' IGNORED");
3463                  C_INFILE.Setpos(C_INFILE.Pos-I);
3464                  SYMBOL:=-Intext(I);
3465                  if SYMBOL=" " then SYMBOL:=-Copy("END_OF_LINE");
3466              end;
3467      end INSPECT ;
3468      end NEXTSYMB;
3469
3470      character NEXTCHAR;
3471
```

```

3473  comment ****
3474  ** AUSDRUCKEN EINER SEMANTISCHEN FEHLERMELDUNG      **
3475  ****
3476
3477  procedure SEM_ERROR(T1,T2,T3);value T1,T2,T3;text T1,T2,T3;
3478  begin OUTLINE; Outtext("SEMANTIC ERROR : ");
3479    OUTLINE;Outtext(T1);Outtext(T2);Outtext(T3);
3480    OUTLINE;
3481    goto FIND_SEMICOLON;
3482  end;                                              B491
3483
3484  comment ****
3485  ** AUSDRUCKEN EINER SYNTAXFEHLERMELDUNG.          **
3486  ** DAS FALSCHE SYMBOL UND DIE LISTE ALLER ERWARTETEN SYMBOLE   **
3487  ** WERDEN GEMELDET.          **
3488  **          **
3489  ****
3490
3491  procedure SYNTAX_ERROR;
3492  begin integer I;                                B492
3493    OUTLINE;
3494    Outtext("==> SYNTAX-ERROR : ");
3495    if TOKEN = ENDFILESY then
3496      begin Outtext(" END OF SOURCE-TEXT") ; OUTLINE ;
3497      INT_WEIGHT:=TERMINATION;
3498      reactivate ASS_PROCESSOR;
3499    end;
3500    Outtext("ILLEGAL SYMBOL "); Outtext(SYMBOL);
3501    Outtext(" ");
3502    OUTLINE;
3503    Outtext(" EXPECTED SYMBOLS: ");
3504    for I:=TOKENS_PASSED step -1 until 1 do          B493
3505      begin
3506        if TOKENTEXT(PASSED_TOKEN(I)).Sub(1,1)~=<""
3507          then Outchar(' ');
3508          Outtext(TOKENTEXT(PASSED_TOKEN(I)));
3509        if TOKENTEXT(PASSED_TOKEN(I)).Sub(1,1)~=<""
3510          then Outchar(' ');
3511        if I=1 then Outchar('|');
3512      end;
3513    OUTLINE;
3514    goto FIND_SEMICOLON;
3515  end SYNTAX_ERROR;                               E492
3516

```

```

3518
3519 comment ****VERWEIS AUF DAS NEUSTE ELEMENT UND DIE WURZEL DER TABELLE ****
3520 ** VERWEIS AUF DAS NEUSTE ELEMENT UND DIE WURZEL DER TABELLE      **
3521 ****VERWEIS AUF DAS NEUSTE ELEMENT UND DIE WURZEL DER TABELLE ****;
3522
3523 ref(ID) STACKTOP,SYMBTBL;
3524
3525
3526 comment ****DIE SYMBOLTABELLE IST ALS BINAERER SUCHBAUM ORGANISIERT.    **
3527 ** DIE SYMBOLTABELLE IST ALS BINAERER SUCHBAUM ORGANISIERT.          **
3528 ** EIN EINTRAG (KNOTEN) ENTHAELT I.W. NAME, TYP UND WERT DES          **
3529 ** BEZEICHNERS.                                                 **
3530 ****DIE SYMBOLTABELLE IST ALS BINAERER SUCHBAUM ORGANISIERT.    **
3531
3532 class ID(IDNAME) ; text IDNAME ;
3533 begin
3534     ref(ID) LEFT,RIGHT,NEXT,Pred;                                     B495
3535     boolean KEYWORD;
3536     integer TOKEN;
3537
3538     text TYPE;
3539     boolean CONSTANT;
3540     integer INT_VALUE,SCRIPT_VALUE;
3541     long real REAL_VALUE;
3542     ref(ACB) AGENT_VALUE;
3543     text STRING_VALUE,OPER_VALUE;
3544     boolean BOOL_VALUE;
3545 end ID;                                                               E495
3546
3547 ref(ID) LASTID,Prev,TBLPREV;
3548
3549 comment ****AUFSUCHE EINES BEZEICHNERS IN DER TABELLE ****
3550 ** AUFSUCHE EINES BEZEICHNERS IN DER TABELLE          **
3551 ****AUFSUCHE EINES BEZEICHNERS IN DER TABELLE ****;
3552
3553 procedure LOOKUP(SYMBNAME); value SYMBNAME; text SYMBNAME;
3554 begin ref(ID) POINTER;                                              B496
3555     POINTER := SYMBTBL;
3556     LASTID:=none;
3557     while POINTER /= none do
3558         begin if SYMBNAME=POINTER.IDNAME then
3559             begin if POINTER.KEYWORD
3560                 then begin TOKEN:=POINTER.TOKEN;
3561                     RESWORD:=true;
3562                     goto FIN;
3563                 end
3564             else LASTID:=POINTER;
3565         end;
3566         Prev:=POINTER;
3567         if SYMBNAME<POINTER.IDNAME then POINTER:=POINTER.LEFT
3568             else POINTER:=POINTER.RIGHT;
3569     end LOOP;
3570     if LASTID==none then TOKEN:=NEWIDSY
3571         else TOKEN:=OLDIDSY;
3572     FIN:
3573 end LOOKUP;                                                       E496
3574
3575 comment ****EINTRAGEN EINES BEZEICHNERS IN DIE TABELLE NACH AUFRUF VON ****
3576 ** EINTRAGEN EINES BEZEICHNERS IN DIE TABELLE NACH AUFRUF VON      **
3577 ** LOOKUP.                                                 **
3578 ****EINTRAGEN EINES BEZEICHNERS IN DIE TABELLE NACH AUFRUF VON ****;
3579
3580 procedure DECLARE;
3581 begin
3582     if LASTSYMB < TBLPREV.IDNAME
3583         then TBLREF:=TBLPREV.LEFT:=-new ID(LASTSYMB)
3584         else TBLREF:=TBLPREV.RIGHT:=-new ID(LASTSYMB);
3585     TBLREF.Pred:=TBLPREV;TBLREF.NEXT:=-STACKTOP;STACKTOP:=-TBLREF;
3586 end DECLARE;                                                       E500
3587
3588 comment ****REKURSIVE PROZEDUR ZUM AUSDRUCKEN EINES UNTERBAUMS (BZW. DES ****
3589 ** REKURSIVE PROZEDUR ZUM AUSDRUCKEN EINES UNTERBAUMS (BZW. DES      **
3590 ** GANZEN BAUMS)                                                 **
3591 ****REKURSIVE PROZEDUR ZUM AUSDRUCKEN EINES UNTERBAUMS (BZW. DES ****;
3592
3593 procedure PRINT (SUBTREE);ref(ID) SUBTREE;
3594 inspect SUBTREE do
3595 begin
3596     if LEFT /= none then PRINT(LEFT);
3597     if ~KEYWORD then
3598         begin Outtext(" | ");Outtext(IDNAME);
3599             BMSOUT.Setpos(25); Outtext(" | ");Outtext(TYPE);
3600             BMSOUT.Setpos(50); Outtext(" | ");
3601             if TYPE=="BOOL" then PRINT_BOOL(BOOL_VALUE);
3602             if TYPE=="REAL" then PRINT_REAL(REAL_VALUE);
3603             if TYPE=="INT" then PRINT_INT(INT_VALUE);
3604         end
3605     end
3606 end PRINT;                                                       B501
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048
4049
4050
4051
4052
4053
4054
4055
4056
4057
4058
4059
4060
4061
4062
4063
4064
4065
4066
4067
4068
4069
4070
4071
4072
4073
4074
4075
4076
4077
4078
4079
4080
4081
4082
4083
4084
4085
4086
4087
4088
4089
4090
4091
4092
4093
4094
4095
4096
4097
4098
4099
4099
4100
4101
4102
4103
4104
4105
4106
4107
4108
4109
4110
4111
4112
4113
4114
4115
4116
4117
4118
4119
4119
4120
4121
4122
4123
4124
4125
4126
4127
4128
4129
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4149
4150
4151
4152
4153
4154
4155
4156
4157
4158
4159
4159
4160
4161
4162
4163
4164
4165
4166
4167
4168
4169
4169
4170
4171
4172
4173
4174
4175
4176
4177
4178
4179
4179
4180
4181
4182
4183
4184
4185
4186
4187
4188
4189
4189
4190
4191
4192
4193
4194
4195
4196
4197
4198
4199
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219
4219
4220
4221
4222
4223
4224
4225
4226
4227
4228
4229
4229
4230
4231
4232
4233
4234
4235
4236
4237
4238
4239
4239
4240
4241
4242
4243
4244
4245
4246
4247
4248
4249
4249
4250
4251
4252
4253
4254
4255
4256
4257
4258
4259
4259
4260
4261
4262
4263
4264
4265
4266
4267
4268
4269
4269
4270
4271
4272
4273
4274
4275
4276
4277
4278
4279
4279
4280
4281
4282
4283
4284
4285
4286
4287
4288
4289
4289
4290
4291
4292
4293
4294
4295
4296
4297
4298
4299
4299
4300
4301
4302
4303
4304
4305
4306
4307
4308
4309
4309
4310
4311
4312
4313
4314
4315
4316
4317
4318
4319
4319
4320
4321
4322
4323
4324
4325
4326
4327
4328
4329
4329
4330
4331
4332
4333
4334
4335
4336
4337
4338
4339
4339
4340
4341
4342
4343
4344
4345
4346
4347
4348
4349
4349
4350
4351
4352
4353
4354
4355
4356
4357
4358
4359
4359
4360
4361
4362
4363
4364
4365
4366
4367
4368
4369
4369
4370
4371
4372
4373
4374
4375
4376
4377
4378
4379
4379
4380
4381
4382
4383
4384
4385
4386
4387
4388
4389
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398
4399
4399
4400
4401
4402
4403
4404
4405
4406
4407
4408
4409
4409
4410
4411
4412
4413
4414
4415
4416
4417
4418
4419
4419
4420
4421
4422
4423
4424
4425
4426
4427
4428
4429
4429
4430
4431
4432
4433
4434
4435
4436
4437
4438
4439
4439
4440
4441
4442
4443
4444
4445
4446
4447
4448
4449
4449
4450
4451
4452
4453
4454
4455
4456
4457
4458
4459
4459
4460
4461
4462
4463
4464
4465
4466
4467
4468
4469
4469
4470
4471
4472
4473
4474
4475
4476
4477
4478
4479
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4489
4489
4490
4491
4492
4493
4494
4495
4496
4497
4498
4498
4499
4499
4500
4501
4502
4503
4504
4505
4506
4507
4508
4509
4509
4510
4511
4512
4513
4514
4515
4516
4517
4518
4519
4519
4520
4521
4522
4523
4524
4525
4526
4527
4528
4529
4529
4530
4531
4532
4533
4534
4535
4536
4537
4538
4539
4539
4540
4541
4542
4543
4544
4545
4546
4547
4548
4549
4549
4550
4551
4552
4553
4554
4555
4556
4557
4558
4559
4559
4560
4561
4562
4563
4564
4565
4566
4567
4568
4569
4569
4570
4571
4572
4573
4574
4575
4576
4577
4578
4579
4579
4580
4581
4582
4583
4584
4585
4586
4587
4588
4589
4589
4590
4591
4592
4593
4594
4595
4596
4597
4598
4599
4599
4600
4601
4602
4603
4604
4605
4606
4607
4608
4609
4609
4610
4611
4612
4613
4614
4615
4616
4617
4618
4619
4619
4620
4621
4622
4623
4624
4625
4626
4627
4628
4629
4629
4630
4631
4632
4633
4634
4635
4636
4637
4638
4639
4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4659
4660
4661
4662
4663
4664
4665
4666
4667
4668
4669
4669
4670
4671
4672
4673
4674
4675
4676
4677
4678
4679
4679
4680
4681
4682
4683
4684
4685
4686
4687
4688
4689
4689
4690
4691
4692
4693
4694
4695
4696
4697
4698
4698
4699
4699
4700
4701
4702
4703
4704
4705
4706
4707
4708
4709
4709
4710
4711
4712
4713
4714
4715
4716
4717
4718
4719
4719
4720
4721
4722
4723
4724
4725
4726
4727
4728
4729
4729
4730
4731
4732
4733
4734
4735
4736
4737
4738
4739
4739
4740
4741
4742
4743
4744
4745
4746
4747
4748
4749
4749
4750
4751
4752
4753
4754
4755
4756
4757
4758
4759
4759
4760
4761
4762
4763
4764
4765
4766
4767
4768
4769
4769
4770
4771
4772
4773
4774
4775
4776
4777
4778
4779
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4798
4799
4799
4800
4801
4802
4803
4804
4805
4806
4807
4808
4809
4809
4810
4811
4812
4813
4814
4815
4816
4817
4818
4819
4819
4820
4821
4822
4823
4824
4825
4826
4827
4828
4829
4829
4830
4831
4832
4833
4834
4835
4836
4837
4838
4839
4839
4840
4841
4842
4843
4844
4845
4846
4847
4848
4849
4849
4850
4851
4852
4853
4854
4855
4856
4857
4858
4859
4859
4860
4861
4862
4863
4864
4865
4866
4867
4868
4869
4869
4870
4871
4872
4873
4874
4875
4876
4877
4878
4879
4879
4880
4881
4882
4883
4884
4885
4886
4887
4888
4889
4889
4890
4891
4892
4893
4894
4895
4896
4897
4898
4898
4899
4899
4900
4901
4902
4903
4904
4905
4906
4907
4908
4909
4909
4910
4911
4912
4913
4914
4915
4916
4917
4918
4919
4919
4920
4921
4922
4923
4924
4925
4926
4927
4928
4929
4929
4930
4931
4932
4933
4934
4935
4936
4937
4938
4939
4939
4940
4941
4942
4943
4944
4945
4946
4947
4948
4949
4949
4950
4951
4952
4953
4954
4955
4956
4957
4958
4959
4959
4960
4961
4962
4963
4964
4965
4966
4967
4968
4969
4969
4970
4971
4972
4973
4974
4975
4976
4977
4978
4979
4979
4980
4981
4982
4983
4984
4985
4986
4987
4988
4989
4989
4990
4991
4992
4993
4994
4995
4996
4997
4998
4998
4999
4999
5000
5001
5002
5003
5004
5005
5006
5007
5008
5009
5009
5010
5011
5012
5013
5014
5015
5016
5017
5018
5019
5019
5020
5021
5022
5023
5024
5025
5026
5027
5028
5029
5029
5030
5031
5032
5033
5034
5035
5036
5037
5038
5039
5039
5040
5041
5042
5043
5044
5045
5046
5047
5048
5049
5049
5050
5051
5052
5053
5054
5055
5056
5057
5058
5059
5059
5060
5061
5062
5063
5064
5065
5066
5067
5068
5069
5069
5070
5071
5072
5073
5074
5075
5076
5077
5078
5079
5079
5080
5081
5082
5083
5084
5085
5086
5087
5088
5089
5089
5090
5091
5092
5093
5094
5095
5096
5097
5098
5099
5099
5100
5101
5102
5103
5104
5105
5106
5107
5108
5109
5109
5110
5111
5112
5113
5114
5115
5116
5117
5118
5119
5119
5120
5121
5122
5123
5124
5125
5126
5127
5128
5129
5129
5130
5131
5132
5133
5134
5135
5136
5137
5138
5139
5139
5140
5141
5142
5143
5144
5145
5146
5147
5148
5149
5149
5150
5151
5152
5153
5154
5155
5156
5157
5158
5159
5159
5160
5161
5162
5163
5164
5165
5166
5167
5168
5169
5169
5170
5171
5172
5173
5174
5175
5176
5177
5178
5179
5179
5180
5181
5182
5183
5184
5185
5186
5187
5188
5189
5189
5190
5191
5192
5193
5194
5195
5196
5197
5198
5199
5199
5200
5201
5202
5203
5204
5205
5206
5207
5208
5209
5209
5210
5211
5212
5213
5214
5215
5216
5217
5218
5219
5219
5220
5221
5222
5223
5224
5225
5226
5227
5228
5229
5229
5230
5231
5232
5233
5234
5235
5236
5237
5238
5239
5239
5240
5241
5242
5243
5244
5245
5246
5247
5248
5249
5249
5250
5251
5252
5253
5254
5255
5256
5257
5258
5259
5259
5260
5261
5262
5263
5264
5265
5266
5267
5268
5269
5269
5270
5271
5272
5273
5274
5275
5276
5277
5278
5279
5279
5280
5281
5282
5283
5284
5285
5286
5287
5288
5289
5289
5290
5291
5292
5293
5294
5295
5296
5297
5298
5299
5299
5300
5301
5302
5303
5304
5305
5306
5307
5308
5309
5309
5310
5311
5312
5313
5314
5315
5316
5317
5318
5319
5319
5320
5321
5322
5323
5324
5325
5326
5327
5328
5329
5329
5330
5331
5332
5333
5334
5335
5336
5337
5338
5339
5339
5340
5341
5342
5343
5344
5345
5346
5347
5348
5349
5349
5350
5351
5352
5353
5354
5355
5356
5357
5358
5359
5359
5360
5361
5362
5363
5364
5365
5366
5367
5368
5369
5369
5370
5371
5372
5373
5374
5375
5376
5377
5378
5379
5379
5380
5381
5382
5383
5384
5385
5386
5387
5388
5389
5389
5390
5391
5392
5393
5394
5395
5396
5397
5398
5399
5399
5400
5401
5402
5403
5404
5405
5406
5407
5408
5409
5409
5410
5411
5412
5413
5414
5415
5416
5417
5418
541
```

```

3604      if TYPE=="STRING" then PRINT_STRING(STRING_VALUE);
3605      if TYPE=="AGENT" then PRINT_AGENT(AGENT_VALUE);
3606      if TYPE=="SCRIPT" then PRINT_SCRIPT(SCRIPT_VALUE);
3607      if TYPE=="OPER" then PRINT_OPER(OPER_VALUE);
3608      BMSOUT.Setpos(78); Outtext("|");
3609      OUTLINE;
3610      end;
3611      if RIGHT /= none then PRINT(RIGHT);
3612  end;                                         E502
3613
3614 comment ****
3615 ** EINTRAGEN EINES KEYWORDS IN DIE TABELLE WAEHREND DER      **
3616 ** INITIALISIERUNGSPHASE DES INITERFACES.                      **
3617 ****
3618
3619 procedure INSERTKW(KWNAME,TOKEN);
3620   value KWNAME; text KWNAME; integer TOKEN;
3621 begin ref(ID) POINTER;
3622   POINTER:-SYMBTBL;
3623   while true do
3624     begin if KWNAME<POINTER.IDNAME then
3625       begin if POINTER.LEFT==none then
3626         begin POINTER.LEFT:-new ID(KWNAME);
3627           POINTER.LEFT.Pred:-POINTER;
3628           POINTER:-POINTER.LEFT;
3629           POINTER.TOKEN:=TOKEN;
3630           POINTER.KEYWORD:=true;
3631           POINTER.NEXT:-STACKTOP;
3632           goto FIN;
3633         end else POINTER:-POINTER.LEFT;
3634       end else
3635         begin if POINTER.RIGHT==none then
3636           begin POINTER.RIGHT:-new ID(KWNAME);
3637             POINTER.RIGHT.Pred:-POINTER;
3638             POINTER:-POINTER.RIGHT;
3639             POINTER.TOKEN:=TOKEN;
3640             POINTER.KEYWORD:=true;
3641             POINTER.NEXT:-STACKTOP;
3642             goto FIN;
3643           end else POINTER:-POINTER.RIGHT;
3644         end;
3645       end LOOP;
3646     FIN: STACKTOP:-POINTER;
3647   end INSERTKW;                                         E503
3648
3649 comment ****
3650 ** IN DER INITIALISIERUNGSPHASE DES INTERFACE WERDEN DIE NAMEN    **
3651 ** DER VOM COMPILER UEBERSETZEN SCRIPTS IN DIE VARIABLENTABELLE    **
3652 ** EINGETRAGEN UND SIND DAHER VON VORNERHEREIN IN DER INTERAKTIVEN  **
3653 ** SITZUNG BEKANNT.                                                 **
3654 ****
3655
3656 procedure INSERT_SCRIPT(I);integer I;
3657 begin ref(ID) POINTER;
3658   POINTER:-SYMBTBL;
3659   while true do
3660     begin if SCRIPT_NAME(I)<POINTER.IDNAME then
3661       begin if POINTER.LEFT==none then
3662         begin POINTER.LEFT:-new ID(SCRIPT_NAME(I));
3663           POINTER.LEFT.Pred:-POINTER;
3664           POINTER:-POINTER.LEFT;
3665           POINTER.NEXT:-STACKTOP;
3666           goto FIN;
3667         end else POINTER:-POINTER.LEFT;
3668       end else
3669         begin if POINTER.RIGHT==none then
3670           begin POINTER.RIGHT:-new ID(SCRIPT_NAME(I));
3671             POINTER.RIGHT.Pred:-POINTER;
3672             POINTER:-POINTER.RIGHT;
3673             POINTER.NEXT:-STACKTOP;
3674             goto FIN;
3675           end else POINTER:-POINTER.RIGHT;
3676         end;
3677       end LOOP;
3678     FIN: STACKTOP:-POINTER;
3679     POINTER.TYPE:-Copy("SCRIPT");
3680     POINTER.SCRIPT_VALUE:=I;
3681     POINTER.CONSTANT:=true;
3682   end INSERT_SCRIPT;                                         E509
3683

```

```
3685      text LASTSYMB;
3686      ref(ID) TBLREF;
3687      integer INTEGER_VALUE ;
3688      long real REAL_VALUE ;
3689      text STRING_VALUE ;
3690
3691      text array TOKENTEXT(0:100);
3692
3693
```

```

3695  comment ****
3696  ** HILFSFUNKTIONEN FUER DEN PARSER
3697  ****;
3698
3699  integer TOKENS_PASSED;
3700  integer array PASSED_TOKENS(0:100);
3701
3702  comment ****
3703  ** EINLESEN EINES TOKENS ALS TEIL EINER SYNTAKTISCHEN EINHEIT. **
3704  ** LIEGT EIN ANDERER TOKEN VOR, SO IST DAS EIN SYNTAXFEHLER. **
3705  ****;
3706
3707  procedure READ(TOK); integer TOK;
3708  if ~THERE_IS(TOK) then SYNTAX_ERROR
3709  else NEXTSYMB;
3710
3711  comment ****
3712  ** EINLESEN EINES ZU DEKLARIERENDEN BEZEICHNERS UND EINTRAGEN **
3713  ** IN DIE SYMBOLTABELLE. **
3714  ** LIEGT KEIN NEWIDSY VOR, DANN SYNTAXFEHLER. **
3715  ****;
3716
3717  procedure NEW_IDENTIFIER;
3718  if ~THERE_IS(NEWIDSY)
3719  then SYNTAX_ERROR
3720  else begin
3721      NEXTSYMB;
3722      DECLARE;
3723  end;                                B515
3724
3725
3726  comment ****
3727  ** UEBERPRUEFUNG, OB EIN BESTIMMTER LOOKAHEAD-TOKEN VORLIEGT. **
3728  ** LIEGT ER NICHT VOR, SO KOMMT ER IN DIE LISTE DER AN DIESER STELLE**
3729  ** ERWARTETEN SYMbole. **
3730  ****;
3731
3732  boolean procedure THERE_IS(TOK);integer TOK;
3733  if TOK~=TOKEN
3734  then begin TOKENS_PASSED:=TOKENS_PASSED+1;
3735      PASSED_TOKEN(TOKENS_PASSED):=TOK;
3736  end;                                 E516
3737  else THERE_IS:=true;
3738
3739  text RES_TYPE;
3740  ref(ACB) RES_AGENT;
3741
3742  comment ****
3743  ** UEBERPRUEFEN DES TYPS DES ZULETZT GELESENEN BEZEICHNERS. **
3744  ****;
3745
3746  procedure TYPE_CHECK(TYPE);value TYPE;text TYPE;
3747  begin
3748      if TBLREF.TYPE~=TYPE
3749          then SEM_ERROR(TBLREF.IDNAME," IS NOT OF TYPE ",TYPE);
3750  end TYPE_CHECK;                      E517
3751
3752  comment ****
3753  ** VERGLEICH ZWEIER ALS TEXTE GEGEBENER DATENTYPEN **
3754  ****;
3755
3756  boolean procedure COMPARE_TYPES(REQ_TYPE,ACT_TYPE);
3757  value REQ_TYPE,ACT_TYPE;text REQ_TYPE,ACT_TYPE;
3758  begin
3759      if REQ_TYPE~=ACT_TYPE
3760          then SEM_ERROR(ACT_TYPE," IS USED INSTEAD OF ",REQ_TYPE)
3761          else COMPARE_TYPES:=true;
3762  end COMPARE_TYPES;                   B518
3763
3764

```

```

3766
3767 comment **** DER PARSER FUER DIE LL(1) GRAMMATIK DER INTERFACE-SPRACHE ****
3768 ** IST NACH DEM PRINZIP DES REKURSIVEN ABSTIEGS GESCHRIEBEN **
3769 ****;
3770
3771
3772 procedure COMMAND;
3773 comment _____
3774   <STATEMENT>
3775     ::= <DISPLAY_STATEMENT>
3776     ::= <STATUS_STATEMENT>
3777     ::= <SYSSTATUS_STATEMENT>
3778     ::= <EVENT_STATEMENT>
3779     ::= <NOEVENT_STATEMENT>
3780     ::= <SYSTEM_STATEMENT>
3781     ::= <NOSYSTEM_STATEMENT>
3782     ::= <OBSERVE_STATEMENT>
3783     ::= <PROT_STATEMENT>
3784     ::= <NOPROT_STATEMENT>
3785     ::= <NOOBSERVE_STATEMENT>
3786     ::= <TRACE_STATEMENT>
3787     ::= <NOTRACE_STATEMENT>
3788     ::= <SNAPSHOT_STATEMENT>
3789     ::= <NOSNAPSHOT_STATEMENT>
3790     ::= <CONST_VAR_DECL>
3791     ::= <ASSIGNMENT>
3792     ::= <OPER_DECL>
3793     ::= <PORT_DECL>
3794     ::= <SEND_STATEMENT>
3795     ::= <RECEIVE_STATEMENT>
3796     ::= <REPLY_STATEMENT>
3797     ::= <MAILBOX_STATEMENT>
3798     ::= <DELETE_STATEMENT>
3799     ::= <LIMIT_STATEMENT>
3800     ::= <RUN_STATEMENTS>
3801     ::= <DUMP_STATEMENT>
3802     ::= <STACKS_STATEMENT>
3803     ::= <STOP_STATEMENT>
3804     ::= <START_STATEMENT>
3805     ::= <READ_STATEMENT>
3806     ::= <HELP_STATEMENT>
3807     ::= <TIME_STATEMENT>
3808     ::= <TERMINATE_STATEMENT>
3809     ::= <EMPTY>
3810
3811 begin

```

B519

```

3812   NEXTSYMB;
3813   if THERE_IS(SENDSY) then SEND_STATEMENT
3814   else if THERE_IS(RECEIVESY) then RECEIVE_STATEMENT
3815   else if THERE_IS(DELETESY) then DELETE_STATEMENT
3816   else if THERE_IS(REPLYSY) then REPLY_STATEMENT
3817   else if THERE_IS(CONSTSY) then CONST_VAR_DECL
3818   else if THERE_IS(VARSY) then CONST_VAR_DECL
3819   else if THERE_IS(OPERSY) then OPER_DECL
3820   else if THERE_IS(PORTSY) then PORT_DECL
3821   else if THERE_IS(OLDIDSY) then ASSIGNMENT
3822   else if THERE_IS(DISPLAYSY) then DISPLAY_STATEMENT
3823   else if THERE_IS(MAILBOXSY) then MAILBOX_STATEMENT
3824   else if THERE_IS(QMARKSY) then DISPLAY_STATEMENT
3825   else if THERE_IS(LIMITSY) then LIMIT_STATEMENT
3826   else if THERE_IS(RUNSY) then RUN_STATEMENT
3827   else if THERE_IS(TERMINATESY) then TERMINATE_STATEMENT
3828   else if THERE_IS(DUMPSY) then DUMP_STATEMENT
3829   else if THERE_IS(OBSERVESY) then OBSERVE_STATEMENT
3830   else if THERE_IS(SYSTEMSY) then SYSTEM_STATEMENT
3831   else if THERE_IS(NOCOBSEVESY) then NOOBSERVE_STATEMENT
3832   else if THERE_IS(NOSYSTEMSY) then NOSYSTEM_STATEMENT
3833   else if THERE_IS(SNAPSHOTSY) then SNAPSHOT_STATEMENT
3834   else if THERE_IS(NOSNAPSHOTSY) then NOSNAPSHOT_STATEMENT
3835   else if THERE_IS(STACKSSY) then STACKS_STATEMENT
3836   else if THERE_IS(STARTSY) then START_STATEMENT
3837   else if THERE_IS(TRACESY) then TRACE_STATEMENT
3838   else if THERE_IS(NOTRACESY) then NOTRACE_STATEMENT
3839   else if THERE_IS(EVENTSY) then EVENT_STATEMENT
3840   else if THERE_IS(NOEVENTSY) then NOEVENT_STATEMENT
3841   else if THERE_IS(PROTSY) then PROT_STATEMENT
3842   else if THERE_IS(STOPSY) then STOP_STATEMENT
3843   else if THERE_IS(READSY) then READ_STATEMENT
3844   else if THERE_IS(NOPROTSY) then NOPROT_STATEMENT
3845   else if THERE_IS(STATUSSY) then STATUS_STATEMENT
3846   else if THERE_IS(SYSSTATUSSY) then SYSSTATUS_STATEMENT
3847   else if THERE_IS(TIMESY) then TIME_STATEMENT
3848   else if THERE_IS(HELPSY) then HELP_STATEMENT;
3849   if ~THERE_IS(SEMICLSSY) then SYNTAX_ERROR;
3850
3851 end COMMAND;

```

E519

```

3852 procedure DISPLAY_STATEMENT;
3853 comment
3854 | <DISPLAY_STATEMENT> ::= (DISPLAY | '?') (. <FACTOR> .)
3855 | ;
3856 begin integer I;
3857     if THERE_IS(DISPLAYSY) then READ(DISPLAYSY)
3858         else READ(QMARKSY);
3859
3860     if THERE_IS(SEMICLSY)
3861         then begin
3862             comment CLEAR_SCREEN; comment SIEMENS;
3863             Outchar(' ');
3864             for I:= 1 step 1 until 76 do Outchar('_');OUTLINE;
3865             Outtext(" | ");Outtext(" IDENTIFIER ");
3866             BMSOUT.Setpos(25);Outtext(" | ");Outtext(" TYPE      ");
3867             BMSOUT.Setpos(50);Outtext(" | ");Outtext(" VALUE      ");
3868             BMSOUT.Setpos(78);Outtext(" | ");OUTLINE;
3869             Outchar(' | ');
3870             for I:= 1 step 1 until 76 do Outchar('_');Outchar(' | ');
3871             BMSOUT.Setpos(25);Outtext(" | ");
3872             BMSOUT.Setpos(50);Outtext(" | ");
3873             BMSOUT.Setpos(78);Outtext(" | ");
3874             OUTLINE;
3875
3876             PRINT(SYMBTBL);
3877
3878             Outchar(' | ');
3879             for I:= 1 step 1 until 76 do Outchar('_');Outchar(' | ');OUTLINE;
3880             OUTLINE;
3881             end
3882             else begin FACTOR;PRINT_TOP(RES_TYPE); end;
3883 end DISPLAY_STATEMENT;                                         E521
3884
3885 procedure READ_STATEMENT;
3886 comment
3887 | <READ_STATEMENT> ::= READ <IDENTIFIER>
3888 | ;
3889 begin
3890     READ(READSY);
3891     NEXTSYMB;
3892     if C_INFILE /= Sysin then C_INFILE.Close;
3893     C_INFILE:=new Infile(LASTSYMB);
3894     C_INFILE.Open(Blanks(80));
3895     C_INFILE.Image.Setpos(80);
3896 end READ_STATEMENT;                                         E522 E522
3897
3898 procedure TERMINATE_STATEMENT;
3899 comment
3900 | <TERMINATE_STATEMENT> ::= TERMINATE
3901 | ;
3902 begin ref(AGENT) A;
3903     READ(TERMINATESY);
3904     STATUS(false);
3905     Sysout.Outimage; OUTLINE;
3906     Outtext(" CSSA-SESSION-STATISTICS");
3907     OUTLINE;
3908     Outtext(" ======");
3909     OUTLINE;Sysout.Outimage; OUTLINE; OUTLINE;
3910     Outtext("SESSION STARTED AT "); Outtext(STARTTIME); OUTLINE;
3911     Outtext("SESSION TERMINATED AT "); Outtext(TIMEOFTDAY);
3912     Outtext(" ON "); Outtext(TODAY); OUTLINE; comment SIEMENS;
3913     Outtext("REAL-TIME USED :"); Outfix((CLOCK-STARTCLOCK)/100,2,8);
3914     Outtext(" SEC."); OUTLINE;
3915     Outtext("CPU-TIME USED :");
3916     Outfix((CPUTIME-STARTCPU)/1000000,2,8);
3917     Outtext(" SEC."); OUTLINE;
3918     Outtext("SIMULATION TIME USED :"); Outfix(Time,4,12);
3919     Outtext(" SEC."); OUTLINE;
3920     Outtext("NUMBER OF AGENTS CREATED :");
3921     Outint(TOTAL_AGENTS,5);
3922     OUTLINE;
3923     Outtext("NUMBER OF MESSAGES SENT :");
3924     Outint(TOTAL_MSGS,5);
3925     OUTLINE;
3926     if C_INFILE/= Sysin then C_INFILE.Close;
3927     reactivate Main before Current;
3928 end TERMINATE_STATEMENT;                                         E524
3929
3930 procedure AGENT_DENOTATION;
3931 comment
3932 | <AGENT_DENOTATION> ::= <IDENTIFIER> (. ('' <INT_CONST> '') .)
3933 | ::= SELF | INTERFACE
3934 | ;
3935 if THERE_IS(INTERFACESY)
3936 then begin READ(INTERFACESY);RES_AGENT:=SELF;end
3937 else if THERE_IS(SELFSY)                                         E525 E525

```

```

3938  then begin READ(SELFSY);RES_AGENT:=-SELF;end else           B526 E526
3939  begin
3940      READ(OLDIDSY);
3941      if THERE_IS(LBRACKSY)
3942          then begin ref(AGENT) A;
3943              TYPE_CHECK("SCRIPT");
3944              READ(LBRACKSY);
3945              READ(INT_CONST);
3946              READ(RRACKSY);
3947              A:-AGENT_LIST.First;
3948              while A=/=none do
3949                  begin if A.CB.ASS_SCRIPT.OWNMODE=TBLREF.SCRIPT_VALUE    B529
3950                      and A.CB.COPY_NR=INTEGER_VALUE
3951                      then goto FOUND;
3952                          A:-A.Suc;
3953                      end;
3954                      SEM_ERROR("AGENT NOT FOUND","","","");
3955                      FOUND:RES_AGENT:=A.CB;
3956                  end
3957          else begin TYPE_CHECK("AGENT");
3958              if TBLREF.AGENT_VALUE==none
3959                  then SEM_ERROR("UNDEFINED AGENT","","","");
3960                  RES_AGENT:=TBLREF.AGENT_VALUE;
3961              end;
3962      end AGENT_DENOTATION;
3963
3964  procedure NOTRACE_STATEMENT;
3965  comment
3966  | <NOTRACE_STATEMENT> ::= NOTRACE <AGENT_DENOTATION> || ',' |
3967  ;
3968  begin
3969      READ(NOTRACESY);
3970      AGENT_DENOTATION;
3971      RES_AGENT TRACE:=false;
3972      while THERE_IS(COMMASY) do
3973          begin READ(COMMASY);
3974              AGENT_DENOTATION;
3975              RES_AGENT TRACE:=false;
3976          end;
3977  end NOTRACE_STATEMENT;                                         E531
3978
3979  procedure TRACE_STATEMENT;
3980  comment
3981  | <TRACE_STATEMENT> ::= TRACE <AGENT_DENOTATION> || ',' |
3982  ;
3983  begin
3984      READ(TRACESY);
3985      AGENT_DENOTATION;
3986      RES_AGENT TRACE:=true;
3987      while THERE_IS(COMMASY) do
3988          begin READ(COMMASY);
3989              AGENT_DENOTATION;
3990              RES_AGENT TRACE:=true;
3991          end;
3992  end TRACE_STATEMENT;                                         E532
3993
3994  procedure STOP_STATEMENT;
3995  comment
3996  | <STOP_STATEMENT> ::= STOP <AGENT_DENOTATION> || ',' |
3997  ;
3998  begin
3999      READ(STOPSY);
4000      AGENT_DENOTATION;
4001      RES_AGENT STOPPED:=true;
4002      RES_AGLNT.Out; comment ... OF ACB_QUEUE;
4003      while THERE_IS(COMMASY) do
4004          begin READ(COMMASY);
4005              AGENT_DENOTATION;
4006              RES_AGENT STOPPED:=true;
4007              RES_AGENT.Out; comment ... OF ACB_QUEUE;
4008          end;
4009  end STOP_STATEMENT;                                         E533
4010
4011  procedure START_STATEMENT;
4012  comment
4013  | <START_STATEMENT> ::= START <AGENT_DENOTATION> || ',' |
4014  ;
4015  begin
4016      READ(STARTSY);
4017      AGENT_DENOTATION;
4018      RES_AGENT STOPPED:=false;
4019      RES_AGENT.Into(RES_AGENT.ASS_PROCESSOR.AC_B_QUEUE);
4020      inspect RES_AGENT.ASS_PROCESSOR do
4021          if Idle then activate ASS_PROCESSOR after Current;
4022      while THERE_IS(COMMASY) do
4023          begin READ(COMMASY);                                         B538

```

```

4024           AGENT_DENOTATION;
4025           RES_AGENT.STOPPED:=false;
4026           RES_AGENT.Into(RES_AGENT.ASS_PROCESSOR.ACB_QUEUE);
4027           inspect RES_AGENT.ASS_PROCESSOR do
4028             if Idle then activate ASS_PROCESSOR after Current;
4029           end;
4030       end START_STATEMENT;                                         E538
4031
4032   procedure NOEVENT_STATEMENT;
4033   comment _____
4034   | <NOEVENT_STATEMENT> ::= NOEVENT
4035   |
4036   begin
4037     READ(NOEVENTSY);
4038     EVENT:=false;SINGLE_STEP:=false;
4039   end NOEVENT_STATEMENT;                                         E539
4040
4041   procedure EVENT_STATEMENT;
4042   comment _____
4043   | <EVENT_STATEMENT> ::= EVENT (. STOP .)
4044   |
4045   begin
4046     READ(EVENTSY);
4047     if THERE_IS(STOPSY) then begin READ(STOPSY);SINGLE_STEP:=true;end
4048     else SINGLE_STEP:=false;
4049     EVENT:=true;
4050   end EVENT_STATEMENT;                                         E540
4051
4052   procedure NOSNAPSHOT_STATEMENT;
4053   comment _____
4054   | <NOSNAPSHOT_STATEMENT> ::= NOSNAPSHOT <AGENT_DENOTATION> || ','
4055   |
4056   begin
4057     READ(NOSNAPSHOTSY);
4058     AGENT_DENOTATION;
4059     RES_AGENT.SNAPSHOT:=false;
4060     while THERE_IS(COMMASY) do
4061       begin READ(COMMASY);
4062         AGENT_DENOTATION;
4063         RES_AGENT.SNAPSHOT:=false;
4064       end;
4065   end NOSNAPSHOT_STATEMENT;                                         E542
4066
4067   procedure SNAPSHOT_STATEMENT;
4068   comment _____
4069   | <SNAPSHOT_STATEMENT> ::= SNAPSHOT <AGENT_DENOTATION> || ','
4070   |
4071   begin
4072     READ(SNAPSHOTSY);
4073     AGENT_DENOTATION;
4074     RES_AGENT.SNAPSHOT:=true;
4075     while THERE_IS(COMMASY) do
4076       begin READ(COMMASY);
4077         AGENT_DENOTATION;
4078         RES_AGENT.SNAPSHOT:=true;
4079       end;
4080   end SNAPSHOT_STATEMENT;                                         E545
4081
4082   procedure STATUS_STATEMENT;
4083   comment _____
4084   | <STATUS_STATEMENT> ::= STATUS
4085   |
4086   begin
4087     READ(STATUSSY);
4088     STATUS(false);
4089   end STATUS_STATEMENT;                                         E546
4090
4091   procedure SYSSTATUS_STATEMENT;
4092   comment _____
4093   | <SYSSTATUS_STATEMENT> ::= SYSSTATUS
4094   |
4095   begin
4096     integer I; ref(MSG) MESS; ref(ACB) A;
4097     READ(SYSSTATUSSY);
4098     Outtext("+++" );PRINT_TIME;Outtext(" SYSTEM STATUS:" );OUTLINE;
4099     BMSOUT.Setpos(2);while BMSOUT.Pos<78 do Outchar('_');OUTLINE;
4100     Outtext("|| PROC. | UTIL. | AGENTS");BMSOUT.Setpos(78);
4101     Outchar('||'); OUTLINE;
4102     Outchar('||');while BMSOUT.Pos<78 do Outchar('_');Outchar('||');
4103     OUTLINE;
4104     for I:=1 step 1 until NR_OF_PROCESSORS do begin
4105       inspect PROCESSORS(I) do begin
4106         Outtext("|| P"); PRINT_INT(ID); BMSOUT.Setpos(9);Outtext("|| ");
4107         PRINT_INT(UTILIZATION); Outchar('%'); BMSOUT.Setpos(17);
4108         Outtext("|| ");
4109         A:=ACB_QUEUE.First;

```

```

4110
4111      while A /= none do
4112          begin
4113              if A.ASS_SCRIPT.PRINT_NAME.Length+BMSOUT.Pos>72 then
4114                  begin
4115                      BMSOUT.Setpos(78); Outchar(' ');
4116                      OUTLINE; Outchar('|');BMSOUT.Setpos(9); Outchar(' ');
4117                      BMSOUT.Setpos(17); Outchar('|'); BMSOUT.Setpos(20);
4118                  end;
4119                  PRINT_AGENT(A); Outtext(" * ");
4120                  A:=A.Suc;
4121          end;
4122          A:=-IDLE_QUEUE.First;
4123          while A /= none do
4124              begin
4125                  if A.ASS_SCRIPT.PRINT_NAME.Length+BMSOUT.Pos>72 then
4126                      begin
4127                          BMSOUT.Setpos(78); Outchar(' ');
4128                          OUTLINE; Outchar('|');BMSOUT.Setpos(9); Outchar(' ');
4129                          BMSOUT.Setpos(17); Outchar('|'); BMSOUT.Setpos(20);
4130                      end;
4131                      PRINT_AGENT(A); Outtext(" ");
4132                      A:=A.Suc;
4133          end;
4134          BMSOUT.Setpos(78); Outchar('|');
4135          OUTLINE;
4136      end;
4137      Outchar('');while BMSOUT.Pos<78 do Outchar('_');Outchar(' ');
4138      OUTLINE; OUTLINE;
4139      BMSOUT.Setpos(2);while BMSOUT.Pos<78 do Outchar('_');OUTLINE;
4140      Outtext(" | BUS | UTIL. | #MSGS | AVG-QLEN | MESSAGES");
4141      BMSOUT.Setpos(78); Outchar(''); OUTLINE;
4142      Outchar('');while BMSOUT.Pos<78 do Outchar('_');Outchar(' ');
4143      OUTLINE;
4144      for I:=1 step 1 until NR_OF_BUSES do
4145          inspect BUSSES(I) do begin
4146              Outtext(" | B"); PRINT_INT(ID); BMSOUT.Setpos(9);Outtext(" | ");
4147              PRINT_INT(UTILIZATION); Outchar('%'); BMSOUT.Setpos(17);
4148              Outtext(" | ");PRINT_INT(NR_OF_MSGS); BMSOUT.Setpos(25);
4149              Outtext(" | ");Outfix(AVG_WAIT,2,7); BMSOUT.Setpos(36);
4150              Outtext(" | ");
4151              MESS:=MESSAGE_LIST.First;
4152              while MESS=/=none do
4153                  begin
4154                      if MESS.OPER.Length+BMSOUT.Pos>76 then
4155                          begin
4156                              BMSOUT.Setpos(78); Outchar(' ');
4157                              OUTLINE; Outchar('|');BMSOUT.Setpos(9); Outchar(' ');
4158                              BMSOUT.Setpos(17); Outchar('|'); BMSOUT.Setpos(25);
4159                              Outchar(' |'); BMSOUT.Setpos(36);
4160                              Outchar(' |'); BMSOUT.Setpos(38);
4161                          end;
4162                          PRINT_OPER(MESS.OPER);Outchar(' ');
4163                          MESS:=MESS.Suc;
4164          end;
4165          BMSOUT.Setpos(78); Outchar(' ');
4166          OUTLINE;
4167      end;
4168      Outchar(' ');
4169      while BMSOUT.Pos<78 do Outchar('_');Outchar(' ');
4170      OUTLINE; OUTLINE;
4171  end SYSSTATUS_STATEMENT;
4172
4173  procedure NOPROT_STATEMENT;
4174  comment _____;
4175  | <NOPROT_STATEMENT> ::= NOPROT
4176  begin
4177      READ(NOPROTSY);
4178      PROT:=-none;
4179  end NOPROT_STATEMENT;
4180
4181  procedure PROT_STATEMENT;
4182  comment _____;
4183  | <PROT_STATEMENT> ::= PROT
4184
4185  begin
4186      READ(PROTSY);
4187      if PROT == none then
4188          begin
4189              PROT:=-PROTOKOL;
4190              inspect PROT do
4191                  begin integer I;
4192                      Outimage;
4193                      CONFIG.Open(Blanks(80));
4194                      for I:=1 step 1 until 25 do
4195                          begin Outtext(CONFIG.Intext(80).Sub(1,72));Outimage;end;
4196
4197
4198
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219
4220
4221
4222
4223
4224
4225
4226
4227
4228
4229
4230
4231
4232
4233
4234
4235
4236
4237
4238
4239
4240
4241
4242
4243
4244
4245
4246
4247
4248
4249
4250
4251
4252
4253
4254
4255
4256
4257
4258
4259
4260
4261
4262
4263
4264
4265
4266
4267
4268
4269
4270
4271
4272
4273
4274
4275
4276
4277
4278
4279
4280
4281
4282
4283
4284
4285
4286
4287
4288
4289
4290
4291
4292
4293
4294
4295
4296
4297
4298
4299
4300
4301
4302
4303
4304
4305
4306
4307
4308
4309
4310
4311
4312
4313
4314
4315
4316
4317
4318
4319
4320
4321
4322
4323
4324
4325
4326
4327
4328
4329
4330
4331
4332
4333
4334
4335
4336
4337
4338
4339
4340
4341
4342
4343
4344
4345
4346
4347
4348
4349
4350
4351
4352
4353
4354
4355
4356
4357
4358
4359
4360
4361
4362
4363
4364
4365
4366
4367
4368
4369
4370
4371
4372
4373
4374
4375
4376
4377
4378
4379
4380
4381
4382
4383
4384
4385
4386
4387
4388
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398
4399
4400
4401
4402
4403
4404
4405
4406
4407
4408
4409
4410
4411
4412
4413
4414
4415
4416
4417
4418
4419
4420
4421
4422
4423
4424
4425
4426
4427
4428
4429
4430
4431
4432
4433
4434
4435
4436
4437
4438
4439
4440
4441
4442
4443
4444
4445
4446
4447
4448
4449
4450
4451
4452
4453
4454
4455
4456
4457
4458
4459
4460
4461
4462
4463
4464
4465
4466
4467
4468
4469
4470
4471
4472
4473
4474
4475
4476
4477
4478
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4489
4490
4491
4492
4493
4494
4495
4496
4497
4498
4499
4500
4501
4502
4503
4504
4505
4506
4507
4508
4509
4510
4511
4512
4513
4514
4515
4516
4517
4518
4519
4520
4521
4522
4523
4524
4525
4526
4527
4528
4529
4530
4531
4532
4533
4534
4535
4536
4537
4538
4539
4540
4541
4542
4543
4544
4545
4546
4547
4548
4549
4550
4551
4552
4553
4554
4555
4556
4557
4558
4559
4560
4561
4562
4563
4564
4565
4566
4567
4568
4569
4570
4571
4572
4573
4574
4575
4576
4577
4578
4579
4580
4581
4582
4583
4584
4585
4586
4587
4588
4589
4590
4591
4592
4593
4594
4595
4596
4597
4598
4599
4600
4601
4602
4603
4604
4605
4606
4607
4608
4609
4610
4611
4612
4613
4614
4615
4616
4617
4618
4619
4620
4621
4622
4623
4624
4625
4626
4627
4628
4629
4630
4631
4632
4633
4634
4635
4636
4637
4638
4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4660
4661
4662
4663
4664
4665
4666
4667
4668
4669
4670
4671
4672
4673
4674
4675
4676
4677
4678
4679
4680
4681
4682
4683
4684
4685
4686
4687
4688
4689
4690
4691
4692
4693
4694
4695
4696
4697
4698
4699
4700
4701
4702
4703
4704
4705
4706
4707
4708
4709
4710
4711
4712
4713
4714
4715
4716
4717
4718
4719
4720
4721
4722
4723
4724
4725
4726
4727
4728
4729
4730
4731
4732
4733
4734
4735
4736
4737
4738
4739
4740
4741
4742
4743
4744
4745
4746
4747
4748
4749
4750
4751
4752
4753
4754
4755
4756
4757
4758
4759
4760
4761
4762
4763
4764
4765
4766
4767
4768
4769
4770
4771
4772
4773
4774
4775
4776
4777
4778
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799
4800
4801
4802
4803
4804
4805
4806
4807
4808
4809
4810
4811
4812
4813
4814
4815
4816
4817
4818
4819
4820
4821
4822
4823
4824
4825
4826
4827
4828
4829
4830
4831
4832
4833
4834
4835
4836
4837
4838
4839
4840
4841
4842
4843
4844
4845
4846
4847
4848
4849
4850
4851
4852
4853
4854
4855
4856
4857
4858
4859
4860
4861
4862
4863
4864
4865
4866
4867
4868
4869
4870
4871
4872
4873
4874
4875
4876
4877
4878
4879
4880
4881
4882
4883
4884
4885
4886
4887
4888
4889
4890
4891
4892
4893
4894
4895
4896
4897
4898
4899
4900
4901
4902
4903
4904
4905
4906
4907
4908
4909
4910
4911
4912
4913
4914
4915
4916
4917
4918
4919
4920
4921
4922
4923
4924
4925
4926
4927
4928
4929
4930
4931
4932
4933
4934
4935
4936
4937
4938
4939
4940
4941
4942
4943
4944
4945
4946
4947
4948
4949
4950
4951
4952
4953
4954
4955
4956
4957
4958
4959
4960
4961
4962
4963
4964
4965
4966
4967
4968
4969
4970
4971
4972
4973
4974
4975
4976
4977
4978
4979
4980
4981
4982
4983
4984
4985
4986
4987
4988
4989
4990
4991
4992
4993
4994
4995
4996
4997
4998
4999
4999

```

```

4196      CONFIG.Close;
4197      Outimage;
4198      Outtext("      PROGRAM GENERATED ON ");Outtext(GENDATE);
4199      Outtext(" AT "); Outtext(GENTIME); Outimage;
4200      Outtext("      BY BMS-CSSA-COMPILER (VERS. ");
4201      Outtext(RELEASEDATE); Outchar(''); Outimage;Outimage;
4202      Outtext("      PROTOCOL OF CSSA SESSION ON ");
4203      Outtext(TODAY);comment SIEMENS;
4204      Outtext(" AT ");Outtext(TIMEOFDAY); Outimage;
4205      Outtext("      =====");
4206      Outtext("=====");
4207      Outimage; Outimage;
4208      end;
4209      end ;
4210  end PROT_STATEMENT;                                         E559
4211
4212 procedure NOOBERVE_STATEMENT;
4213 comment _____
4214 | <NOOBERVE_STATEMENT> ::= NOOBERVE
4215 | _____;
4216 begin
4217   READ(NOOBERSY);
4218   OBSERVE:=false;
4219 end NOOBERVE_STATEMENT;                                         E558
4220
4221 procedure OBSERVE_STATEMENT;
4222 comment _____
4223 | <OBSERVE_STATEMENT> ::= OBSERVE
4224 | _____;
4225 begin
4226   READ(OBSERVERSY);
4227   OBSERVE:=true;
4228 end OBSERVE_STATEMENT;                                         E557
4229
4230 procedure NOSYSTEM_STATEMENT;
4231 comment _____
4232 | <NOSYSTEM_STATEMENT> ::= NOSYSTEM
4233 | _____;
4234 begin
4235   READ(NOSYSTEMSY);
4236   SYSTEM:=false;
4237 end NOSYSTEM_STATEMENT;                                         B561
4238
4239 procedure SYSTEM_STATEMENT;
4240 comment _____
4241 | <SYSTEM_STATEMENT> ::= SYSTEM
4242 | _____;
4243 begin
4244   READ(SYSTEMSY);
4245   SYSTEM:=true;
4246 end SYSTEM_STATEMENT;                                         E561
4247
4248 procedure STACKS_STATEMENT;
4249 comment _____
4250 | <STACKS_STATEMENT> ::= STACKS <AGENT_DENOTATION>
4251 | _____;
4252 begin
4253   READ(STACKSSY);
4254   AGENT_DENOTATION;
4255   CLEAR_SCREEN;
4256   PRINT_STACKS(RES_AGENT);
4257 end STACKS_STATEMENT;                                         B562
4258
4259 procedure DUMP_STATEMENT;
4260 comment _____
4261 | <DUMP_STATEMENT> ::= DUMP <AGENT_DENOTATION>
4262 | _____;
4263 begin
4264   READ(DUMPSY);
4265   AGENT_DENOTATION;
4266   comment    CLEAR_SCREEN; comment SIEMENS;
4267   PRINT_DUMP(RES_AGENT);
4268 end DUMP_STATEMENT;                                         E563
4269
4270 procedure TIME_STATEMENT;
4271 comment _____
4272 | <TIME_STATEMENT> ::= TIME
4273 | _____;
4274 begin
4275   external assembly integer procedure CPUTIME;comment SIEMENS;
4276   READ(TIMESY);
4277   comment SIEMENS;
4278   Outtext(TODAY);Outtext("  "); Outtext(TIMEOFDAY);
4279   Outtext("  CPU =");
4280   Outfix((CPUTIME-STARTCPU)/1000000,2,8);
4281   Outtext(" SEC."); comment SIEMENS;

```

```

4282      Outtext("    CSSA-SESSION =");Outfix((CLOCK-STARTCLOCK)/100,2,8);
4283      Outtext(" SEC.");
4284      OUTLINE;
4285 end TIME_STATEMENT;                                         E567
4286
4287 procedure RUN_STATEMENT;
4288 comment _____
4289 | <RUN_STATEMENT> ::= RUN (. <INT_CONST> | <REAL_CONST> .) |
4290 |_____;
4291 begin integer T,T_OLD; real T_MAX; boolean NET_ACTIVE;          B568
4292     READ(RUNSY);
4293     if THERE_IS(INT_CONST) then
4294     begin
4295         READ(INT_CONST);
4296         T_MAX:=INTEGER_VALUE;
4297     end
4298     else if THERE_IS(REAL_CONST) then                                B569
4299     begin
4300         READ(REAL_CONST);
4301         T_MAX:=REAL_VALUE;
4302     end else T_MAX:=1.0&50;                                         E570
4303     if ~THERE_IS(SEMICLSY) then SYNTAX_ERROR;
4304     T:=CLOCK;
4305     while T_MAX > 0 do
4306     begin T_OLD:=T;T:=CLOCK;                                       B571
4307         if T>T_OLD+TIME_LIMIT then
4308         begin
4309             if EVENT then STATUS(false);
4310             Outtext(">>>");PRINT_TIME;
4311             Outtext(" REAL-TIME LIMIT EXCEEDED"); OUTLINE;
4312             goto INTERRUPTED;
4313         end;                                                 E572
4314         Hold(0.1);
4315         NET_ACTIVE:=not PROCESSORS(1).MESSAGE_LIST.Empty;
4316         for I:=2 step 1 until NR_OF_PROCESSORS do
4317             if ~PROCESSORS(I).Idle then NET_ACTIVE:=true;
4318         for I:=1 step 1 until NR_OF_BUSES do
4319             if ~BUSES(I).Idle then NET_ACTIVE:=true;
4320         if not NET_ACTIVE then
4321             begin
4322                 if EVENT then STATUS(false);
4323                 Outtext(">>>");PRINT_TIME;
4324                 Outtext(" SYSTEM TERMINATED"); OUTLINE;
4325                 goto INTERRUPTED;
4326             end;                                                 E573
4327             T_MAX := T_MAX-0.1;
4328         end;
4329         INTERRUPTED:
4330     end RUN_STATEMENT;                                              E568
4331
4332     integer TIME_LIMIT;
4333
4334 procedure LIMIT_STATEMENT;
4335 comment _____
4336 | <LIMIT_STATEMENT> ::= LIMIT <INT_CONST> |
4337 |_____;                                                       B574
4338 begin integer I,T,T_OLD;
4339     READ(LIMITSY);
4340     READ(INT_CONST);
4341     if ~THERE_IS(SEMICLSY) then SYNTAX_ERROR;
4342     TIME_LIMIT:=INTEGER_VALUE*100;
4343 end LIMIT_STATEMENT;                                         E574
4344
4345 procedure MAILBOX_STATEMENT;
4346 comment _____
4347 | <MAILBOX_STATEMENT> ::= MAILBOX (. <AGENT_DENOTATION> .) |
4348 |_____;                                                       B575
4349 begin ref(MSG) M;ref(ACB) A;integer MSG_NR;
4350     READ(MAILBOXSY);
4351     A:=-SELF;
4352     if THERE_IS(OLDIDSY)
4353         then begin AGENT_DENOTATION;
4354             A:=-RES_AGENT;
4355         end;                                                 E576
4356
4357     comment CLEAR_SCREEN; comment SIEMENS;
4358     Outtext("MAILBOX OF "); PRINT_AGENT(A); Outtext(" :"); OUTLINE;
4359     while BMSOUT.Pos<77 do Outchar('_'); OUTLINE;
4360     M:=-A.MAILBOX.First;
4361     while M/=none do
4362         begin MSG_NR:=MSG_NR+1;                               B577
4363             Outchar('(');PRINT_INT(MSG_NR);Outtext(" ) ");
4364             M.PRINT(true); OUTLINE;
4365             M:=-M.Suc;
4366         end;
4367     while BMSOUT.Pos<77 do Outchar('_'); OUTLINE; OUTLINE;
4368

```

```

4368
4369 end MAILBOX_STATEMENT;                                         E575
4370
4371 procedure HELP_STATEMENT;
4372 comment _____
4373 | <HELP_STATEMENT> ::= HELP
4374 |
4375 begin integer I;
4376   Outtext("A COMMAND STARTS WITH ONE OF THE FOLLOWING SYMBOLS:");
4377   OUTLINE;
4378
4379   for I:=TOKENS_PASSED step -1 until 1 do
4380     begin
4381       if TOKENTEXT(PASSED_TOKEN(I)).Sub(1,1)~=<""
4382         then Outchar(' ');
4383         Outtext(TOKENTEXT(PASSED_TOKEN(I)));
4384         if TOKENTEXT(PASSED_TOKEN(I)).Sub(1,1)~=<""
4385           then Outchar(' ');
4386           if BMSOUT.Pos>60 then OUTLINE;
4387           if I=1 then Outchar('|');
4388         end;
4389
4390   OUTLINE;
4391
4392   READ(HELPSY);
4393 end HELP_STATEMENT;                                              E578
4394
4395 procedure PATTERN;
4396 comment _____
4397 | <PATTERN> ::= '(' (. <IDENTIFIER> .) || ',' ')'
4398 |
4399 begin
4400   READ(LBRACKSY);
4401   CURRENT_PARM:=-CURRENT_MSG.PARMS.First;
4402   if CURRENT_PARM==none
4403     then SEM_ERROR("NOT ENOUGH PARMS IN THE MESSAGE","","","");
4404   if THERE_IS(OLDIDSY)
4405     then begin READ(OLDIDSY);
4406       if COMPARE_TYPES(CURRENT_PARM.TYP,TBLREF.TYPE)
4407         then begin GET_PARM;ASSIGN(TBLREF);end;
4408       end;
4409   while THERE_IS(COMMASY) do
4410     begin
4411       CURRENT_PARM:=-CURRENT_PARM.Suc;
4412       if CURRENT_PARM==none
4413         then SEM_ERROR("NOT ENOUGH PARMS IN THE MESSAGE","","","");
4414       READ(COMMASY);
4415       if THERE_IS(OLDIDSY)
4416         then begin READ(OLDIDSY);
4417           if COMPARE_TYPES(CURRENT_PARM.TYP,TBLREF.TYPE)
4418             then begin GET_PARM;ASSIGN(TBLREF);end;
4419           end;
4420       end;
4421       READ(RBRACKSY);
4422     end PATTERN;                                                 E580
4423
4424 procedure PORT_DECL;
4425 comment _____
4426 | <PORT_DECL> ::= PORT '::' <IDENTIFIER> || ','
4427 |
4428 begin
4429   READ(PORTSY);
4430   READ(COLONSY);
4431   NEW_IDENTIFIER;
4432   TBLREF.TYPE:=-Copy("PORT");TBLREF.CONSTANT:=true;
4433   while THERE_IS(COMMASY) do
4434     begin READ(COMMASY);
4435       NEW_IDENTIFIER;
4436       TBLREF.TYPE:=-Copy("PORT");TBLREF.CONSTANT:=true;
4437     end;
4438   end PORT_DECL;                                              E587
4439
4440 procedure OPER_DECL;
4441 comment _____
4442 | <OPER_DECL> ::= OPER '::' <IDENTIFIER> || ','
4443 |
4444 begin
4445   READ(OPERSY);
4446   READ(COLONSY);
4447   NEW_IDENTIFIER;
4448   TBLREF.TYPE:=-Copy("LITERAL");TBLREF.CONSTANT:=true;
4449   TBLREF.OPER_VALUE:=-LASTSYMB;
4450   while THERE_IS(COMMASY) do
4451     begin READ(COMMASY);
4452       NEW_IDENTIFIER;
4453       TBLREF.TYPE:=-Copy("LITERAL");TBLREF.CONSTANT:=true;

```

```

4454         TBLREF.OPER_VALUE:=-LASTSYMB;
4455         end;
4456     end OPER_DECL;
4457
4458 procedure CONST_VAR_DECL;
4459 comment
4460     <CONST_VAR_DECL> ::= (CONST|VAR) <TYPE> '::' <IDENTIFIER>||','
4461             (. ('-'|':') <EXPRESSION> .)
4462
4463 begin boolean CONSTANT;ref(ID)LAST_IDENT,IDENT;
4464     if THERE_IS(VARSY)
4465         then READ(VARSY)
4466     else begin
4467         READ(CONSTS);
4468         CONSTANT:=true;
4469         end;
4470     TYPE;
4471     READ(COLONSY);
4472     LAST_IDENT:=-STACKTOP;
4473     NEW_IDENTIFIER; TBLREF.TYPE:=-RES_TYPE;
4474     while THERE_IS(COMMASY) do
4475         begin READ(COMMASY);
4476             NEW_IDENTIFIER; TBLREF.TYPE:=-RES_TYPE;
4477         end;
4478     if THERE_IS(ASSIGNSY) or THERE_IS(DENOTESY)
4479     then begin
4480         NEXTSYMB;
4481         EXPRESSION(RES_TYPE);
4482         for IDENT:=-STACKTOP,IDENT.NEXT while IDENT=/=LAST_IDENT do
4483             ASSIGN(IDENT);
4484             POP(RES_TYPE);
4485         end;
4486     end CONST_VAR_DECL;
4487
4488 procedure TYPE;
4489 comment
4490     <TYPE> ::= BOOL | INT | STRING | AGENT | OPER | SCRIPT | REAL
4491
4492 begin
4493     if THERE_IS(BOOLSY) then READ(BOOLSY)
4494     else if THERE_IS(REALSY) then READ(REALSY)
4495     else if THERE_IS(INTSY) then READ(INTSY)
4496     else if THERE_IS(STRINGSY) then READ(STRINGSY)
4497     else if THERE_IS(AGENTSY) then READ(AGENTSY)
4498     else if THERE_IS(OPERSY) then READ(OPERSY)
4499     else READ(SCRIPTSY);
4500     RES_TYPE:=-LASTSYMB;
4501 end TYPE;
4502
4503 ref(ACB) LAST_TARGET;
4504 procedure SEND_STATEMENT;
4505 comment
4506     <SEND_STATEMENT> ::= SEND <OPER_EXPRESSION> <MESSAGE>
4507             (. TO <AGENT_EXPRESSION>
4508                 (. REPLY TO <IDENTIFIER> | INHERIT .) .)
4509
4510 begin
4511     READ(SENDSY);
4512     EXPRESSION("OPER");
4513     MESSAGE;           MSG_TOP.WERT.OPER:=-OPER_TOP.WERT;
4514     if THERE_IS(TOSY) then
4515         begin
4516             READ(TOSY);
4517             EXPRESSION("AGENT");
4518             if THERE_IS(REPLYSY)
4519                 then begin READ(REPLYSY); READ(TOSY);
4520                     READ(OLDIDSY); TYPE_CHECK("PORT");
4521                     inspect MSG_TOP.WERT do
4522                         begin REPLY_EXPECTED:=true;
4523                             PORT:=-TBLREF.IDNAME;
4524                             WAITING_AGENT:=-SELF;
4525                         end;
4526                     end
4527                 else if THERE_IS(INHERITSY)
4528                     then begin READ(INHERITSY);
4529                         inspect MSG_TOP.WERT do
4530                             begin REPLY_EXPECTED:=SELF.WAITING;
4531                                 PORT:=-SELF.PORT;
4532                                 WAITING_AGENT:=-SELF.WAITING_AGENT;
4533                             end;
4534                     end;
4535                 end else AGENT_PUSH(LAST_TARGET);
4536                 if ~THERE_IS(SEMICLSY) then SYNTAX_ERROR;
4537
4538             LAST_TARGET:=-MSG_TOP.WERT.RECEIVER:=-AGENT_TOP.WERT;
4539             MSG_TOP.WERT.SENDER:=-SELF;

```

```

4540           INT_WEIGHT:=SEND_MESSAGE;
4541           reactivate ASS_PROCESSOR;
4542           AGENT_POP;OPER_POP;MSG_POP;
4543
4544   end SEND_STATEMENT;                                         E595
4545
4546
4547   procedure REPLY_STATEMENT;
4548   comment _____
4549   | <REPLY_STATEMENT> ::=_
4550   |   REPLY (. <INT_CONST> .) <MESSAGE>
4551
4552   begin boolean MSG_SPECIFIED;integer I;
4553     READ(REPLYSY);
4554     if THERE_IS(INT_CONST)
4555       then begin READ(INT_CONST);
4556         MSG_SPECIFIED:=true;
4557         if INTEGER_VALUE>MAILBOX.Cardinal or INTEGER_VALUE<1
4558           then SEM_ERROR("ILLEGAL MESSAGE NR.", "", "");
4559         CURRENT_MSG:=MAILBOX.First;
4560         for I:=2 step 1 until INTEGER_VALUE do
4561           CURRENT_MSG:=CURRENT_MSG.Suc;
4562       end;
4563     MESSAGE;
4564     if ~THERE_IS(SEMICLSY) then SYNTAX_ERROR;
4565     if MSG_SPECIFIED
4566       then begin MSG_TOP.WERT.PORT:=-CURRENT_MSG.PORT;          B601
4567         MSG_TOP.WERT.RECEIVER:=-CURRENT_MSG.WAITING_AGENT;
4568       end
4569     else begin MSG_TOP.WERT.PORT:=-PORT;                         B602
4570       MSG_TOP.WERT.RECEIVER:=-WAITING_AGENT;
4571     end;
4572     MSG_TOP.WERT.SENDER:=-SELF;
4573     MSG_TOP.WERT.OPER:=-Copy("*REPLY");
4574     INT_WEIGHT:=SEND_MESSAGE;
4575     reactivate ASS_PROCESSOR;
4576     MSG_POP;
4577
4578   end REPLY_STATEMENT;                                         E601
4579
4580   procedure DELETE_STATEMENT;
4581   comment _____
4582   | <DELETE_STATEMENT> ::=_
4583   |   DELETE <INT_CONST>
4584
4585   begin integer I;                                         B605
4586     READ(DELETESY);
4587     READ(INT_CONST);
4588     if INTEGER_VALUE>MAILBOX.Cardinal or INTEGER_VALUE<1
4589       then SEM_ERROR("ILLEGAL MESSAGE NR.", "", "");
4590     CURRENT_MSG:=MAILBOX.First;
4591     for I:=2 step 1 until INTEGER_VALUE do
4592       CURRENT_MSG:=CURRENT_MSG.Suc;
4593     CURRENT_MSG.Out;
4594     if ~THERE_IS(SEMICLSY) then SYNTAX_ERROR;
4595     NOFOUND:
4596   end DELETE_STATEMENT;                                       E605
4597
4598   procedure RECEIVE_STATEMENT;
4599   comment _____
4600   | <RECEIVE_STATEMENT> ::=_
4601   |   RECEIVE <INT_CONST>      (. <PATTERN> .)
4602
4603   begin integer I;                                         B606
4604     READ( RECEIVESY );
4605     READ( INT_CONST );
4606     if INTEGER_VALUE>MAILBOX.Cardinal or INTEGER_VALUE<1
4607       then SEM_ERROR("ILLEGAL MESSAGE NR.", "", "");
4608     CURRENT_MSG:=MAILBOX.First;
4609     for I:=2 step 1 until INTEGER_VALUE do
4610       CURRENT_MSG:=CURRENT_MSG.Suc;
4611     WAITING:=CURRENT_MSG.REPLY_EXPECTED;
4612     CALLER:=CURRENT_MSG.SENDER;
4613     WAITING_AGENT:=CURRENT_MSG.WAITING_AGENT;
4614     if WAITING then PORT:=-CURRENT_MSG.PORT else PORT:=-notext;
4615     if WAITING then
4616       begin Outtext("PLEASE REPLY TO ");
4617         PRINT_AGENT(WAITING_AGENT);OUTLINE;
4618         Outtext("USING THE <REPLY_STATEMENT>");OUTLINE;
4619         if WAITING_AGENT=/=CALLER then
4620           begin OUTLINE;
4621             Outtext("MESSAGE WAS SENT BY ");
4622             PRINT_AGENT(CALLER);
4623           end;
4624           OUTLINE;
4625         end;                                              E607
4626       end;                                              E608
4627     end;                                              E609

```

```

4626      if THERE_IS(LBRACKSY) then PATTERN;
4627      NOMATCH:
4628  end RECEIVE_STATEMENT;                                         E606
4629
4630 procedure MESSAGE;
4631 comment _____
4632   <MESSAGE> ::= '(' <EXPRESSION> || ',' ')
4633   ::= <EMPTY>
4634
4635 begin
4636   MSG_PUSH(new MSG);
4637   if THERE_IS(LBRACKSY)
4638   then begin
4639     READ(LBRACKSY);
4640     EXPRESSION(notext); SET_PARM;POP(RES_TYPE);
4641     while THERE_IS(COMMASY) do
4642       begin
4643         READ(COMMASY);
4644         EXPRESSION(notext); SET_PARM;POP(RES_TYPE);
4645       end;
4646       READ(RBRACKSY);
4647     end;
4648   end MESSAGE;                                                 E609
4649
4650 procedure ASSIGNMENT;
4651 comment _____
4652   <ASSIGNMENT> ::= <IDENTIFIER> ('='|':-') <EXPRESSION>
4653
4654 begin ref(ID) TARGET;
4655   READ(OLDIDSY);
4656   TARGET:=TBLREF;
4657   if THERE_IS(ASSIGNSY)
4658   then READ(ASSIGNSY)
4659   else READ(DENOTESY);
4660   EXPRESSION(TARGET.TYPE);
4661   if ~THERE_IS(SEMICLSY) then SYNTAX_ERROR;
4662   ASSIGN(TARGET);
4663   POP(TARGET.TYPE);
4664
4665 end ASSIGNMENT;                                               E612
4666
4667 procedure EXPRESSION(TYPE);
4668 value TYPE; text TYPE;
4669 comment _____
4670   <INT_EXPRESSION> ::= <EXPRESSION>
4671
4672 comment _____
4673   <BOOL_EXPRESSION> ::= <EXPRESSION>
4674
4675 comment _____
4676   <STRING_EXPRESSION> ::= <EXPRESSION>
4677
4678 comment _____
4679   <SCRIPT_EXPRESSION> ::= <EXPRESSION>
4680
4681 comment _____
4682   <AGENT_EXPRESSION> ::= <EXPRESSION>
4683
4684 comment _____
4685   <OPER_EXPRESSION> ::= <EXPRESSION>
4686
4687 comment _____
4688   <REAL_EXPRESSION> ::= <EXPRESSION>
4689
4690 comment _____
4691   <EXPRESSION> ::= <FACTOR>
4692
4693 begin
4694   FACTOR;
4695   if TYPE~=notext then COMPARE_TYPES(TYPE,RES_TYPE);
4696 end EXPRESSION;                                              E613
4697
4698 procedure FACTOR;
4699 comment _____
4700   <FACTOR> ::= <INT_CONST>
4701   <REAL_CONST>
4702   <STRING_CONST>
4703   OWNMODE
4704   NOSCRIPT
4705   NEW <SCRIPT_EXPRESSION>    <MESSAGE>
4706   NOAGENT
4707   HOOPER
4708   SELF
4709   INTERFACE
4710   <IDENTIFIER>
4711 begin boolean NEGATIVE;text TYPE_OF_FACTOR;                  B614

```

```

4712      if THERE_IS(MINUSSY) then begin NEGATIVE:=true;READ(MINUSSY);end;           B615 E615
4713      if THERE_IS(INT_CONST)
4714          then begin
4715              READ(INT_CONST);
4716              TYPE_OF_FACTOR:=-Copy("INT");
4717              INT_PUSH(INTEGER_VALUE);
4718          end
4719      else if THERE_IS(REAL_CONST)
4720          then begin
4721              READ(REAL_CONST);
4722              TYPE_OF_FACTOR:=-Copy("REAL");
4723              REAL_PUSH(REAL_VALUE);
4724          end
4725      else if THERE_IS(STRING_CONST)
4726          then begin
4727              READ(STRING_CONST);
4728              TYPE_OF_FACTOR:=-Copy("STRING");
4729              STRING_PUSH(STRING_VALUE);
4730          end
4731      else if THERE_IS(OWNMODESY)
4732          then begin
4733              READ(OWNMODESY);
4734              TYPE_OF_FACTOR:=-Copy("SCRIPT");
4735              SCRIPT_PUSH(OWNMODE);
4736          end
4737      else if THERE_IS(NOSCRIPTSY)
4738          then begin
4739              READ(NOSCRIPTSY);
4740              TYPE_OF_FACTOR:=-Copy("SCRIPT");
4741              SCRIPT_PUSH(NOSCRIPT);
4742          end
4743      else if THERE_IS(NEWSY)
4744          then begin
4745              READ(NEWSY);
4746              EXPRESSION("SCRIPT");
4747              MESSAGE;
4748              TYPE_OF_FACTOR:=-Copy("AGENT");
4749              INT_WEIGHT:=CREATE_AGENT;
4750              reactivate ASS_PROCESSOR;
4751              SCRIPT_POP;MSG_POP;
4752          end
4753      else if THERE_IS(NOOPERSY)
4754          then begin
4755              READ(NOOPERSY);
4756              TYPE_OF_FACTOR:=-Copy("OPER");
4757              OPER_PUSH(notext);
4758          end
4759      else if THERE_IS(NOAGENTSY)
4760          then begin
4761              READ(NOAGENTSY);
4762              TYPE_OF_FACTOR:=-Copy("AGENT");
4763              AGENT_PUSH(NOAGENT);
4764          end
4765      else if THERE_IS(INTERFACESY)
4766          then begin
4767              READ(INTERFACESY);
4768              TYPE_OF_FACTOR:=-Copy("AGENT");
4769              AGENT_PUSH(INTERFACE) ;
4770          end
4771      else if THERE_IS(SELFSY)
4772          then begin
4773              READ(SELFSY);
4774              TYPE_OF_FACTOR:=-Copy("AGENT");
4775              AGENT_PUSH(SELF);
4776          end
4777      else if THERE_IS(TRUESY)
4778          then begin
4779              READ(TRUESY);
4780              TYPE_OF_FACTOR:=-Copy("BOOL");
4781              BOOL_PUSH(true) ;
4782          end
4783      else if THERE_IS(FALSESY)
4784          then begin
4785              READ(FALSESY);
4786              TYPE_OF_FACTOR:=-Copy("BOOL");
4787              BOOL_PUSH(false);
4788          end
4789      else
4790          begin
4791              READ(OLDIDSY);
4792              PUSH(TBLREF);
4793              TYPE_OF_FACTOR:=-TBLREF.TYPE;
4794              if TYPE_OF_FACTOR="LITERAL"
4795                  then TYPE_OF_FACTOR:=-Copy("OPER");
4796          end;
4797          RES_TYPE:=-TYPE_OF_FACTOR;

```



```

4807  comment ****
4808  ** PROZEDUREN FUER SEMANTISCHE AKTIONEN, DIE VOM PARSER AUS AUF-  **
4809  ** GERUFEN WERDEN.          **
4810  ****;
4811
4812  procedure SET_PARM;
4813  begin NEW_PARM:-new PARM(RES_TYPE);
4814    if RES_TYPE=="STRING" then NEW_PARM.STRINGWERT:=STRING_TOP.WERT;
4815    if RES_TYPE=="BOOL" then NEW_PARM.BOOLWERT:=BOOL_TOP.WERT;
4816    if RES_TYPE=="AGENT" then NEW_PARM.AGENTWERT:=AGENT_TOP.WERT;
4817    if RES_TYPE=="OPER"
4818      then NEW_PARM.OPERWERT:=OPER_TOP.WERT;
4819    if RES_TYPE=="SCRIPT" then NEW_PARM.SCRIPTWERT:=SCRIPT_TOP.WERT;
4820    if RES_TYPE=="REAL" then NEW_PARM.REALWERT:=REAL_TOP.WERT;
4821    if RES_TYPE=="INT" then NEW_PARM.INTWERT:=INT_TOP.WERT;
4822    NEW_PARM.Into(MSG_TOP.WERT.PARMS);
4823  end;                                E631
4824
4825  procedure GET_PARM;
4826  begin
4827    if CURRENT_PARM.TYP=="STRING"
4828      then STRING_PUSH(CURRENT_PARM.STRINGWERT);
4829    if CURRENT_PARM.TYP=="BOOL"
4830      then BOOL_PUSH(CURRENT_PARM.BOOLWERT);
4831    if CURRENT_PARM.TYP=="AGENT"
4832      then AGENT_PUSH(CURRENT_PARM.AGENTWERT);
4833    if CURRENT_PARM.TYP=="OPER"
4834      then OPER_PUSH(CURRENT_PARM.OPERWERT);
4835    if CURRENT_PARM.TYP=="SCRIPT"
4836      then SCRIPT_PUSH(CURRENT_PARM.SCRIPTWERT);
4837    if CURRENT_PARM.TYP=="INT"
4838      then INT_PUSH(CURRENT_PARM.INTWERT);
4839    if CURRENT_PARM.TYP=="REAL"
4840      then REAL_PUSH(CURRENT_PARM.REALWERT);
4841  end;                                E632
4842
4843  procedure ASSIGN(IDENT); ref(ID) IDENT;
4844  begin
4845    if IDENT.TYPE=="STRING" then IDENT.STRING_VALUE:=STRING_TOP.WERT;
4846    if IDENT.TYPE=="BOOL" then IDENT.BOOL_VALUE:=BOOL_TOP.WERT;
4847    if IDENT.TYPE=="AGENT" then IDENT.AGENT_VALUE:=AGENT_TOP.WERT;
4848    if IDENT.TYPE=="OPER"
4849      then IDENT.OPER_VALUE:=OPER_TOP.WERT;
4850    if IDENT.TYPE=="SCRIPT" then IDENT.SCRIPT_VALUE:=SCRIPT_TOP.WERT;
4851    if IDENT.TYPE=="INT" then IDENT.INT_VALUE:=INT_TOP.WERT;
4852    if IDENT.TYPE=="REAL" then IDENT.REAL_VALUE:=REAL_TOP.WERT;
4853  end;                                E633
4854
4855  procedure PRINT_TOP(TYPE); text TYPE;
4856  begin
4857    if TYPE=="STRING" then PRINT_STRING(STRING_TOP.WERT);
4858    if TYPE=="BOOL" then PRINT_BOOL(BOOL_TOP.WERT);
4859    if TYPE=="AGENT" then PRINT_AGENT(AGENT_TOP.WERT);
4860    if TYPE=="OPER" then PRINT_OPER(OPER_TOP.WERT);
4861    if TYPE=="SCRIPT" then PRINT_SCRIPT(SCRIPT_TOP.WERT);
4862    if TYPE=="INT" then PRINT_INT(INT_TOP.WERT);
4863    if TYPE=="REAL" then PRINT_REAL(REAL_TOP.WERT);
4864    OUTLINE;
4865  end;                                E634
4866
4867  procedure POP(TYP); value TYP; text TYP;
4868  begin
4869    if TYP=="STRING" then STRING_POP;
4870    if TYP=="BOOL" then BOOL_POP;
4871    if TYP=="AGENT" then AGENT_POP;
4872    if TYP=="OPER" then OPER_POP;
4873    if TYP=="SCRIPT" then SCRIPT_POP;
4874    if TYP=="INT" then INT_POP;
4875    if TYP=="REAL" then REAL_POP;
4876  end;                                E635
4877
4878  procedure PUSH(T); ref(ID) T;
4879  begin
4880    if T.TYPE=="STRING" then STRING_PUSH(T.STRING_VALUE);
4881    if T.TYPE=="BOOL" then BOOL_PUSH(T.BOOL_VALUE);
4882    if T.TYPE=="AGENT" then AGENT_PUSH(T.AGENT_VALUE);
4883    if T.TYPE=="OPER" then OPER_PUSH(T.OPER_VALUE);
4884    if T.TYPE=="LITERAL" then OPER_PUSH(T.IDNAME);
4885    if T.TYPE=="SCRIPT" then SCRIPT_PUSH(T.SCRIPT_VALUE);
4886    if T.TYPE=="INT" then INT_PUSH(T.INT_VALUE);
4887    if T.TYPE=="REAL" then REAL_PUSH(T.REAL_VALUE);
4888  end;                                E636

```

```

4890    integer
4891        ENDFILESY,
4892        VARSY, CONSTSY,
4893        INTSY, BOOLSY, AGENTSY, STRINGSY, REALSY, REAL_CONST,
4894        ASSIGNSY, LBRACKSY, RBRACKSY, DENOTESY,
4895        COMMASY, SEMICLSY,
4896        STRING_CONST, INT_CONST, TRUESY, FALSESY,
4897        SCRIPTSY,
4898        NOAGENTSY, NOSRIPTSY, OWNMODESY, INTERFAESY, SELFSY,
4899        SENDSY, TOSY, RECEIVESY, PORTSY, COLONSY,
4900        NEWSY, DEMANDSY, FROMSY, TIMESY,
4901        NEWIDSY, OLDDIDSY, OPERSY, DISPLAYSY, HELPSY, QMARDSY,
4902        LIMITSY, MAILBOXSY, TERMINATESY, MINUSSY, DUMPSY,
4903        TRACESY, NOTRACESY, PROTSY, NOPROTSY, OBSERVESY, NOOBSEVESY,
4904        SNAPSHOTSY, NOSNAPSHOTSY, TERMINALSY, READSY, WAITINGSY, REPLYSY,
4905        CALLERSY, BLOCKEDSY, INHERITSY, NOOPERSY, DELETESY, SYSSTATUSSY,
4906        STACKSSY, STATUSSY, EVENTSY, NOEVENTSY, STOPSY, STARTSY, RUNSY,
4907        SYSTEMSY, NOSYSTEMSY;
4908
4909 PRINT_NAME:=-Copy("INTERFACE");
4910 OWNMODE:=1;
4911
4912 SYMBTBL:=-new ID(notext);
4913 STACKTOP:=-SYMBTBL;
4914 NEXTCHAR:=' ';
4915
4916 ENDFILESY:=1;
4917 VARSY:=2 ;CONSTSY:=3 ;
4918 INTSY:=4 ;BOOLSY:=5 ;AGENTSY:=6 ;STRINGSY:=7 ;
4919 ASSIGNSY:=8 ;LBRACKSY:=9 ;RBRACKSY:=10;DENOTESY:=11 ;
4920 COMMASY:=12;SEMICLSY:=13;
4921 STRING_CONST:=14;INT_CONST:=15;TRUESY:=16;FALSESY:=17;
4922 SCRIPTSY:=18;
4923 NOAGENTSY:=19;NOSRIPTSY:=20;OWNMODESY:=21;INTERFAESY:=22;SELFSY:=23;
4924 SENDSY:=24;TOSY:=25;RECEIVESY:=26;REAL_CONST:=27;
4925 NEWSY:=28;DEMANDSY:=29;FROMSY:=30;
4926 NEWIDSY:=31;OLDDIDSY:=32;OPERSY:=33;
4927 DISPLAYSY:=34;HELPSY:=35;QMARDSY:=36;LIMITSY:=37;PORTSY:=38;
4928 MAILBOXSY:=39;TERMINATESY:=40;MINUSSY:=41;DUMPSY:=42;
4929 TRACESY:=44;NOTRACESY:=45;PROTSY:=46;NOPROTSY:=47;
4930 OBSERVESY:=48;NOOBSEVESY:=49;SNAPSHOTSY:=50;NOSNAPSHOTSY:=51;
4931 TERMINALSY:=52;READSY:=53;WAITINGSY:=54;REPLYSY:=55;CALLERSY:=56;
4932 BLOCKEDSY:=57;INHERITSY:=58;REALSY:=59;NOOPERSY:=60;DELETESY:=61;
4933 STACKSSY:=62;STATUSSY:=63;EVENTSY:=64;NOEVENTSY:=65;STOPSY:=66;
4934 STARTSY:=67;COLONSY:=68;TIMESY:=69;RUNSY:=70;SYSSTATUSSY:=71;
4935 SYSTEMSY:=72; NOSYSTEMSY:=73;
4936
4937 TOKENTEXT(1 ):-Copy("<END_OF_PROGRAM>");
4938 TOKENTEXT(2 ):-Copy("VAR");
4939 TOKENTEXT(3 ):-Copy("CONST");
4940 TOKENTEXT(4 ):-Copy("INT");
4941 TOKENTEXT(5 ):-Copy("BOOL");
4942 TOKENTEXT(6 ):-Copy("AGENT");
4943 TOKENTEXT(7 ):-Copy("STRING");
4944 TOKENTEXT(8 ):-Copy(":=");
4945 TOKENTEXT(9 ):-Copy("(");
4946 TOKENTEXT(10):-Copy(")");
4947 TOKENTEXT(11):-Copy(":");
4948 TOKENTEXT(12):-Copy(",");
4949 TOKENTEXT(13):-Copy("<END_OF_LINE> OR ';' ");
4950 TOKENTEXT(14):-Copy("<STRING-CONSTANT> ");
4951 TOKENTEXT(15):-Copy("<INTEGER-CONSTANT> ");
4952 TOKENTEXT(16):-Copy("TRUE");
4953 TOKENTEXT(17):-Copy("FALSE");
4954 TOKENTEXT(18):-Copy("SCRIPT");
4955 TOKENTEXT(19):-Copy("NOAGENT");
4956 TOKENTEXT(20):-Copy("NOSRIPT");
4957 TOKENTEXT(21):-Copy("OWNMODE");
4958 TOKENTEXT(22):-Copy("INTERFACE");
4959 TOKENTEXT(23):-Copy("SELF");
4960 TOKENTEXT(24):-Copy("SEND");
4961 TOKENTEXT(25):-Copy("TO");
4962 TOKENTEXT(26):-Copy("RECEIVE");
4963 TOKENTEXT(27):-Copy("<REAL_CONSTANT> ");
4964 TOKENTEXT(28):-Copy("NEW");
4965 TOKENTEXT(29):-Copy("DEMAND");
4966 TOKENTEXT(30):-Copy("FROM");
4967 TOKENTEXT(31):-Copy("<NEW_IDENTIFIER> ");
4968 TOKENTEXT(32):-Copy("<OLD_IDENTIFIER> ");
4969 TOKENTEXT(33):-Copy("OPER");
4970 TOKENTEXT(34):-Copy("DISPLAY");
4971 TOKENTEXT(35):-Copy("HELP");
4972 TOKENTEXT(36):-Copy("?");
4973 TOKENTEXT(37):-Copy("LIMIT");
4974 TOKENTEXT(38):-Copy("PORT");
4975 TOKENTEXT(39):-Copy("MAILBOX");

```

```

4976 TOKENTEXT(40):-Copy("TERMINATE");
4977 TOKENTEXT(41):-Copy("-");
4978 TOKENTEXT(42):-Copy("DUMP");
4979 comment 43 FEHLT;
4980 TOKENTEXT(44):-Copy("TRACE");
4981 TOKENTEXT(45):-Copy("NOTRACE");
4982 TOKENTEXT(46):-Copy("PROT");
4983 TOKENTEXT(47):-Copy("NOPROT");
4984 TOKENTEXT(48):-Copy("OBSERVE");
4985 TOKENTEXT(49):-Copy("NOOBSERVE");
4986 TOKENTEXT(50):-Copy("SNAPSHOT");
4987 TOKENTEXT(51):-Copy("NOSNAPSHOT");
4988 TOKENTEXT(52):-Copy("TERMINAL");
4989 TOKENTEXT(53):-Copy("READ");
4990 TOKENTEXT(54):-Copy("WAITING");
4991 TOKENTEXT(55):-Copy("REPLY");
4992 TOKENTEXT(56):-Copy("CALLER");
4993 TOKENTEXT(57):-Copy("BLOCKED");
4994 TOKENTEXT(58):-Copy("INHERIT");
4995 TOKENTEXT(59):-Copy("REAL");
4996 TOKENTEXT(60):-Copy("NOOPER");
4997 TOKENTEXT(61):-Copy("DELETE");
4998 TOKENTEXT(62):-Copy("STACKS");
4999 TOKENTEXT(63):-Copy("STATUS");
5000 TOKENTEXT(64):-Copy("EVENT");
5001 TOKENTEXT(65):-Copy("NOEVENT");
5002 TOKENTEXT(66):-Copy("STOP");
5003 TOKENTEXT(67):-Copy("START");
5004 TOKENTEXT(68):-Copy(":");
5005 TOKENTEXT(69):-Copy("TIME");
5006 TOKENTEXT(70):-Copy("RUN");
5007 TOKENTEXT(71):-Copy("SYSSTATUS");
5008 TOKENTEXT(72):-Copy("SYSTEM");
5009 TOKENTEXT(73):-Copy("NOSYSTEM");
5010
5011 comment ****
5012 ** SCHLUESSELWOERTER IN SYMBOLTABELLE EINTRAGEN **
5013 ****;
5014
5015 INSERTKW("RUN",RUNSY);
5016 INSERTKW("INT",INTSY);
5017 INSERTKW("VAR",VARSY);
5018 INSERTKW("CONST",CONSTSY);
5019 INSERTKW("REAL",REALSY);
5020 INSERTKW("TIME",TIMESY);
5021 INSERTKW("PORT",PORTSY);
5022 INSERTKW("BOOL",BOOLSY);
5023 INSERTKW("ACENT",AGENTSY);
5024 INSERTKW("STRING",STRINGSY);
5025 INSERTKW("TRUE",TRUESY);
5026 INSERTKW("FALSE",FALSESY);
5027 INSERTKW("TRACE",TRACESY);
5028 INSERTKW("NOTRACE",NOTRACESY);
5029 INSERTKW("EVENT",EVENTSY);
5030 INSERTKW("START",STARTSY);
5031 INSERTKW("STOP",STOPSY);
5032 INSERTKW("NOEVENT",NOEVENTSY);
5033 INSERTKW("PROT",PROTSY);
5034 INSERTKW("NOPROT",NOPROTSY);
5035 INSERTKW("SCRIPT",SCRIPTSY);
5036 INSERTKW("NOAGENT",NOAGENTSY);
5037 INSERTKW("NOSCRIPT",NOSCRIPTSY);
5038 INSERTKW("OWNMODE",OWNMODESY);
5039 INSERTKW("INTERFACE",INTERFACESY);
5040 INSERTKW("OPER",OPERSY);
5041 INSERTKW("SNAPSHOT",SNAPSHOTSY);
5042 INSERTKW("NOSNAPSHOT",NOSNAPSHOTSY);
5043 INSERTKW("OBSERVE",OBSERVESY);
5044 INSERTKW("NOOBSERVE",NOOBSERVESY);
5045 INSERTKW("SYSTEM",SYSTEMSY);
5046 INSERTKW("NOSYSTEM",NOSYSTEMSY);
5047 INSERTKW("SELF",SELFSY);
5048 INSERTKW("SEND",SENDSY);
5049 INSERTKW("RECEIVE",RECEIVESY);
5050 INSERTKW("TO",TOSY);
5051 INSERTKW("NEW",NEWSY);
5052 INSERTKW("DEMAND",DEMANDSY);
5053 INSERTKW("FROM",FROMSY);
5054 INSERTKW("DISPLAY",DISPLAYSY);
5055 INSERTKW("LIMIT",LIMITSY);
5056 INSERTKW("DUMP",DUMPSY);
5057 INSERTKW("HELP",HELPSY);
5058 INSERTKW("TERMINATE",TERMINATESY);
5059 INSERTKW("MAILBOX",MAILBOXSY);
5060 INSERTKW("TERMINAL",TERMINALSY);
5061 INSERTKW("READ",READSY);

```

```
5062  INSERTKW("WAITING",WAITINGSY);
5063  INSERTKW("REPLY",REPLYSY);
5064  INSERTKW("CALLER",CALLERSY);
5065  INSERTKW("BLOCKED",BLOCKEDSY);
5066  INSERTKW("INHERIT",INHERITSY);
5067  INSERTKW("NOOPER",NOOPERSY);
5068  INSERTKW("DELETE",DELETESY);
5069  INSERTKW("STACKS",STACKSSY);
5070  INSERTKW("STATUS",STATUSSY);
5071  INSERTKW("SYSSTATUS",SYSSTATUSSY);
5072
5073  comment ****
5074  ** SCRIPTS IN SYMBOLTABELLE EINTRAGEN      **
5075  ****;
5076
5077  DEFINE_SCRIPT_NAMES;
5078  begin integer I;
5079    for I:=2 step 1 until NR_OF_SCRIPTS do INSERT_SCRIPT(I);
5080  end;                                         B637
                                                E637
```

```
5082    TIME_LIMIT:=500;
5083      INBUFFER:-Blanks(140); comment SIEMENS ;
5084      C_INFILE:=new Infile("INIT");
5085      C_INFILE.Open(Blanks(80));
5086      C_INFILE.Image.Setpos(80);
5087      comment ****;
5088      ** LESEN DES NAECHSTEN KOMMANDOS      **
5089      ****;
5090
5091      NEXT_INPUT:COMMAND;goto NEXT_INPUT;
5092      FIND_SEMICOLON: while NEXTCHAR~=';' and C_INFILE.Pos~=80 do
5093        NEXTCHAR:=C_INFILE.Inchar;
5094        if C_INFILE /= Sysin then C_INFILE.Close ;
5095        if C_INFILE /= Sysin then C_INFILE:=Sysin;
5096        comment SWITCH TO TERMINAL-INPUT ;
5097        goto NEXT_INPUT;
5098      end SCRIPT1 ;
5099
5100      %ENDCOPY
```

E452

```

5102  AS class AS1;
      begin
5103    ref(RELATION) VAR107;
5104    ref(ARRAY_DESC) VAR114;
5105    ref(FIELD) VAR115;
5106    ref(ARRAY_DESC) VAR116;
5107    ref(FIELD) VAR117;
5108    ref(ARRAY_DESC) VAR118;
5109    ref(FIELD) VAR119;
5110    ref(ARRAY_DESC) VAR120;
5111    ref(FIELD) VAR121;
5112    ref(ARRAY_DESC) VAR122;
5113    ref(FIELD) VAR123;
5114    ref(ARRAY_DESC) VAR124;
5115    ref(FIELD) VAR125;
5116    ref(ARRAY_DESC) VAR126;
5117    ref(FIELD) VAR127;
5118    ref(ARRAY_DESC) VAR128;
5119    ref(FIELD) VAR129;
5120    ref(INT_SET) VAR131;
5121    ref(REAL_SET) VAR133;
5122    ref(AGENT_SET) VAR135;
5123    ref(BOOL_SET) VAR137;
5124    ref(STRING_SET) VAR139;
5125    ref(ENUM_SET) VAR141;
5126    ref(OPER_SET) VAR143;
5127    ref(SCRIPT_SET) VAR145;
5128    procedure PRINT_VARS;
5129    begin
5130      DUMP_RELATION("PERSONS",VAR107);
5131      inspect VAR114 do DUMP("ANONYM");
5132      DUMP_ARRAY("A1",VAR115);
5133      inspect VAR116 do DUMP("ANONYM");
5134      DUMP_ARRAY("A2",VAR117);
5135      inspect VAR118 do DUMP("ANONYM");
5136      DUMP_ARRAY("A3",VAR119);
5137      inspect VAR120 do DUMP("ANONYM");
5138      DUMP_ARRAY("A4",VAR121);
5139      inspect VAR122 do DUMP("ANONYM");
5140      DUMP_ARRAY("A5",VAR123);
5141      inspect VAR124 do DUMP("ANONYM");
5142      DUMP_ARRAY("A6",VAR125);
5143      inspect VAR126 do DUMP("ANONYM");
5144      DUMP_ARRAY("A7",VAR127);
5145      inspect VAR128 do DUMP("ANONYM");
5146      DUMP_ARRAY("A8",VAR129);
5147      inspect VAR131 do DUMP("S1");
5148      inspect VAR133 do DUMP("S2");
5149      inspect VAR135 do DUMP("S3");
5150      inspect VAR137 do DUMP("S4");
5151      inspect VAR139 do DUMP("S5");
5152      inspect VAR141 do DUMP("S6");
5153      inspect VAR143 do DUMP("S7");
5154      inspect VAR145 do DUMP("S8");
5155    end -- PRINT_VARS --;
5156    SCOPE_TYPE:=Copy("SCRIPT");
5157    SCOPE_NAME:=Copy("RTSDEMO");
5158  end;
5159  RECORD class RECORD104;
5160  begin
5161    procedure ASSIGN(I);integer I;
5162    inspect Current qua MODULE.ASS_PROCESSOR do
5163    inspect ACTIVE_ACB do
5164    begin
5165      if I=1 then
5166        begin VAR105:=STRING_TOP.WERT;
5167          TRACE_STRING("NAME");STRING_POP;
5168        end;
5169      if I=2 then
5170        begin VAR106:=INT_TOP.WERT;
5171          TRACE_INT("ALTER");INT_POP;
5172        end;
5173    end -- ASSIGN --;
5174    boolean procedure EQUAL(X);ref(RECORD104)X;
5175    begin EQUAL:=
5176      VAR105=X.VAR105 and
5177      true;
5178    end -- ASSIGN --;
5179    text VAR105;
5180    integer VAR106;
5181    procedure DUMP;
5182    begin
5183      DUMP_STRING("---NAME",VAR105);
5184      DUMP_INT("---ALTER",VAR106);
5185    end -- DUMP --;
5186    procedure PUSH(I);integer I;
5187

```

```

5188      inspect Current qua MODULE.ASS_PROCESSOR do          0   2P
5189      begin                                              0   2P B646
5190          if I=1 then STRING_PUSH(VAR105);
5191          if I=2 then INT_PUSH(VAR106);
5192      end -- PUSH --;                                     0   2P
5193      FIELD_TYPE(1):=Copy("STRING");
5194      FIELD_TYPE(2):=Copy("INT");
5195  end;
5196 AS class AS3;
5197 begin
5198     real VAR113;
5199     procedure PRINT_VARS;
5200     begin
5201         DUMP_REAL("X",VAR113);
5202     end -- PRINT_VARS --;
5203     SCOPE_TYPE:=Copy("FUNCTION");
5204     SCOPE_NAME:=Copy("SIN");
5205 end;
5206 AS class AS4;
5207 begin
5208     procedure PRINT_VARS;
5209     begin
5210     end -- PRINT_VARS --;
5211     SCOPE_TYPE:=Copy("FACET");
5212     SCOPE_NAME:=Copy("A");
5213 end;
5214 AS class AS5;
5215 begin
5216     procedure PRINT_VARS;
5217     begin
5218     end -- PRINT_VARS --;
5219     SCOPE_TYPE:=Copy("OPERATION");
5220     SCOPE_NAME:=Copy("IDLE");
5221 end;

```

```

5223  comment *** ANFANG DES SCRIPT RTSDEMO ***;                                2  IA
5224  MODULE class SCRIPT2;                                                 2  IA
5225  begin                                                               2  IA B653
5226  OWNMODE:=2;                                                       2  IA
5227  PRINT_NAME:-Copy("RTSDEMO");                                         2  IA
5228  MAIN_ENTRY:;                                                       2  IA
5229  inspect ASS_PROCESSOR do inspect ACTIVE_ACB do begin                2  IA B654
5230  switch BRANCH:=;                                                 2  IB
5231  RAS1,RAS2;                                                       2  IB
5232  switch ENTRY:=;                                                 2  IB
5233  ENT1,ENT2,ENT3,ENT4,ENT5,ENT6,ENT7,ENT8;                           2  IB
5234  procedure RETURN(I);integer I; begin                               2  IB B655
5235  end;                                                               2  IB E655
5236  procedure START_EXEC(I);integer I; begin                           2  IB B656
5237  end;                                                               2  IB E656
5238  goto ENTRY(EINSPRUNG);START_EXEC(EINSPRUNG);                      2  IC
5239  ENT1:                                                               2  IC
5240  CURRENT_LINE:=3;                                                 2  IC
5241  NEW_TRACE_SECTION;                                              2  IC
5242  AKT_AS:-new AS1(AKT_AS,ENVIRONMENT);                            2  ID
5243  AKT_AS qua AS1.VAR107:-new RELATION;                           2  II
5244  INT_PUSH(1);                                                 2  II
5245  INT_PUSH(5);                                                 2  II
5246  AKT_AS qua AS1.VAR114:-new ARRAY_DESC(1);                         2  II
5247  AKT_AS qua AS1.VAR115                                         2  II
5248  :-new INT_ARRAY(AKT_AS qua AS1.VAR114);                         2  II
5249  CURRENT_LINE:=9;                                               2  II
5250  INT_PUSH(1);                                                 2  II
5251  INT_PUSH(5);                                                 2  II
5252  AKT_AS qua AS1.VAR116:-new ARRAY_DESC(1);                         2  II
5253  AKT_AS qua AS1.VAR117                                         2  II
5254  :-new REAL_ARRAY(AKT_AS qua AS1.VAR116);                         2  II
5255  CURRENT_LINE:=10;                                              2  II
5256  INT_PUSH(1);                                                 2  II
5257  INT_PUSH(5);                                                 2  II
5258  AKT_AS qua AS1.VAR118:-new ARRAY_DESC(1);                         2  II
5259  AKT_AS qua AS1.VAR119                                         2  II
5260  :-new AGENT_ARRAY(AKT_AS qua AS1.VAR118);                         2  II
5261  CURRENT_LINE:=11;                                              2  II
5262  INT_PUSH(1);                                                 2  II
5263  INT_PUSH(5);                                                 2  II
5264  AKT_AS qua AS1.VAR120:-new ARRAY_DESC(1);                         2  II
5265  AKT_AS qua AS1.VAR121                                         2  II
5266  :-new BOOL_ARRAY(AKT_AS qua AS1.VAR120);                         2  II
5267  CURRENT_LINE:=12;                                              2  II
5268  INT_PUSH(1);                                                 2  II
5269  INT_PUSH(5);                                                 2  II
5270  AKT_AS qua AS1.VAR122:-new ARRAY_DESC(1);                         2  II
5271  AKT_AS qua AS1.VAR123                                         2  II
5272  :-new STRING_ARRAY(AKT_AS qua AS1.VAR122);                        2  II
5273  CURRENT_LINE:=13;                                              2  II
5274  INT_PUSH(1);                                                 2  II
5275  INT_PUSH(5);                                                 2  II
5276  AKT_AS qua AS1.VAR124:-new ARRAY_DESC(1);                         2  II
5277  AKT_AS qua AS1.VAR125                                         2  II
5278  :-new ENUM_ARRAY(AKT_AS qua AS1.VAR124);                         2  II
5279  CURRENT_LINE:=14;                                              2  II
5280  INT_PUSH(1);                                                 2  II
5281  INT_PUSH(5);                                                 2  II
5282  AKT_AS qua AS1.VAR126:-new ARRAY_DESC(1);                         2  II
5283  AKT_AS qua AS1.VAR127                                         2  II
5284  :-new OPER_ARRAY(AKT_AS qua AS1.VAR126);                         2  II
5285  CURRENT_LINE:=15;                                              2  II
5286  INT_PUSH(1);                                                 2  II
5287  INT_PUSH(5);                                                 2  II
5288  AKT_AS qua AS1.VAR128:-new ARRAY_DESC(1);                         2  II
5289  AKT_AS qua AS1.VAR129                                         2  II
5290  :-new SCRIPT_ARRAY(AKT_AS qua AS1.VAR128);                        2  II
5291  CURRENT_LINE:=16;                                              2  II
5292  AKT_AS qua AS1.VAR131:-new INT_SET;                             2  II
5293  CURRENT_LINE:=17;                                              2  II
5294  AKT_AS qua AS1.VAR133:-new REAL_SET;                            2  II
5295  CURRENT_LINE:=18;                                              2  II
5296  AKT_AS qua AS1.VAR135:-new AGENT_SET;                           2  II
5297  CURRENT_LINE:=19;                                              2  II
5298  AKT_AS qua AS1.VAR137:-new BOOL_SET;                            2  II
5299  CURRENT_LINE:=20;                                              2  II
5300  AKT_AS qua AS1.VAR139:-new STRING_SET;                           2  II
5301  CURRENT_LINE:=21;                                              2  II
5302  AKT_AS qua AS1.VAR141:-new ENUM_SET;                            2  II
5303  CURRENT_LINE:=22;                                              2  II
5304  AKT_AS qua AS1.VAR143:-new OPER_SET;                            2  II
5305  CURRENT_LINE:=23;                                              2  II
5306  AKT_AS qua AS1.VAR145:-new SCRIPT_SET;                           2  II
5307  CURRENT_LINE:=24;                                              2  II
5308  if SNAPSHOT then OBSERVE_DUMP(SELF);                            2  II

```

```

5309  goto START2;                                2  1Z
5310  comment *** ANFANG DER FUNKTION SIN ***;    2  3A
5311  FUNCT3:                                     2  3A
5312  CURRENT_LINE:=8;                            2  3A
5313  AKT_AS:-new AS3(AKT_AS,ENVIRONMENT);       2  3D
5314  AKT_AS qua AS3.VAR113:=REAL_TOP.WERT;      2  3I
5315  TRACE_REAL("X");                           2  3I
5316  REAL_POP;                                 2  3I
5317  if SNAPSHOT then OBSERVE_DUMP(SELF);        2  3I
5318  REAL_PUSH(CSSASIN)                         2  3S
5319  AKT_AS qua AS3.VAR113));                   2  3S
5320  AKT_AS:-AKT_AS.LAST_AS;                    2  3Z
5321  goto BRANCH(AKT_AS.RAS);RETURN(AKT_AS.RAS); 2  3Z
5322  comment *** ENDE DER FUNKTION SIN ***;      2  3Z
5323  comment *** ANFANG DER FACETTE A ***;      2  4A
5324  FACET4:                                     2  4A
5325  CURRENT_OPER:-notext;                      2  4A
5326  CURRENT_LINE:=25;                          2  4A
5327  AKT_AS:-new AS4(AKT_AS,ENVIRONMENT);       2  4D
5328  if SNAPSHOT then OBSERVE_DUMP(SELF);        2  4I
5329  comment *** FACETTENSCHLEIFE VON A ***;     2  4L
5330  NEXT4:                                      2  4L
5331  NEXT_MESSAGE;                            2  4L
5332  if CURRENT_MSG==none then goto IDLE4;      2  4L
5333  goto NEXT4;                               2  4T
5334  SUCC4:                                     2  4T
5335  CURRENT_OPER:-notext;                      2  4T
5336  END_OF_SECTION;                          2  4T
5337  EINSPRUNG:=3;Hold(ELAPSED_TIME);ELAPSED_TIME:=0; 2  4T
5338  goto MAIN_ENTRY;ENT3:                     2  4T
5339  NEW_TRACE_SECTION;                      2  4T
5340  goto NEXT4;                               2  4T
5341  IDLE4:                                     2  4X
5342  comment *** AUFRUF DER OPERATION IDLE ***; 2  4X
5343  ENVIRONMENT:-AKT_AS;                      2  4X
5344  AKT_AS.RAS:=1;AKT_AS.RAS_LINE:=26;goto OPER147;RAS1: 2  4X
5345  END_OF_SECTION;                          2  4X
5346  EINSPRUNG:=4;Hold(ELAPSED_TIME);ELAPSED_TIME:=0; 2  4X
5347  goto MAIN_ENTRY;ENT4:                     2  4X
5348  NEW_TRACE_SECTION;                      2  4X
5349  goto NEXT4;                               2  4X
5350  END_OF_SECTION;                          2  4X
5351  EINSPRUNG:=5;Hold(ELAPSED_TIME);ELAPSED_TIME:=0; 2  4X
5352  goto MAIN_ENTRY;ENT5:                     2  4X
5353  EINSPRUNG:=6;INT_WEIGHT:=AGENT_IDLE;reactivate ASS_PROCESSOR ; 2  4X
5354  ELAPSED_TIME:=0;                          2  4X
5355  goto MAIN_ENTRY;ENT6:                     2  4X
5356                                         2  4X
5357  NEW_TRACE_SECTION;                      2  4X
5358  goto NEXT4;                               2  4X
5359  comment *** ENDE DER FACETTE A ***;      2  4Z
5360  comment *** ANFANG DER OPERATION IDLE ***; 2  5A
5361  OPER147:                                    2  5A
5362  CURRENT_LINE:=26;                        2  5A
5363  AKT_AS:-new AS5(AKT_AS,ENVIRONMENT);       2  5D
5364  if SNAPSHOT then OBSERVE_DUMP(SELF);        2  5I
5365  TRACE_OPER_CALL;CURRENT_LINE:=26;           2  5M
5366  CURRENT_LINE:=26;                        2  5S
5367  ELAPSED_TIME:=ELAPSED_TIME+ 0.1000;        2  5S
5368  ENDOP147:                                2  5Z
5369  END_OF_SECTION;                          2  5Z
5370  EINSPRUNG:=2;Hold(ELAPSED_TIME);ELAPSED_TIME:=0; 2  5Z
5371  goto MAIN_ENTRY;ENT2:                     2  5Z
5372  NEW_TRACE_SECTION;                      2  5Z
5373  AKT_AS:-AKT_AS.LAST_AS;                  2  5Z
5374  PASSOP147:                                2  5Z
5375  goto BRANCH(AKT_AS.RAS);RETURN(AKT_AS.RAS); 2  5Z
5376  comment *** HIER ENDET DIE OPERATION IDLE ***; 2  5Z
5377  START2:                                     299995
5378  CURRENT_FACET:-Copy("A");                 299995
5379  ENVIRONMENT:-AKT_AS;                      299995
5380  CURRENT_MSG:=none;                        299995
5381  AKT_AS.RAS:=2;AKT_AS.RAS_LINE:=29;goto FACET4;RAS2: 299995
5382  TERM2:                                     299995
5383  END_OF_SECTION;                          299995
5384  EINSPRUNG:=7;Hold(ELAPSED_TIME);ELAPSED_TIME:=0; 299995
5385  goto MAIN_ENTRY;ENT7:                     299995
5386  EINSPRUNG:=8;INT_WEIGHT:=TERMINATION;reactivate ASS_PROCESSOR ; 299995
5387  ELAPSED_TIME:=0;                          299995
5388  goto MAIN_ENTRY;ENT8:                     299995
5389                                         299995
5390  NEW_TRACE_SECTION;                      299995
5391  end;                                       299995 E654
5392  end SCRIPT2;                           299995 E653
5393  comment *** ENDE DES SCRIPTS RTSDEMO ***; 299995

```

```

5395 ref(MODULE) procedure NEW_MODULE(SCR_ID);integer SCR_ID;          999 1
5396 begin                                         999 1 B657
5397 if SCR_ID=1 then NEW_MODULE:-new SCRIPT1;      999 1
5398 if SCR_ID=2 then NEW_MODULE:-new SCRIPT2;      999 1
5399 end NEW_MODULE ;                           999 1 E657
5400 text array SCRIPT_NAME(1:2);                 999 1
5401 integer array NR_OF_AGENTS(1:2);            999 1
5402 integer NR_OF_SCRIPTS;                     999 1
5403 procedure DEFINE_SCRIPT_NAMES;             999 1
5404 begin                                         999 1 B658
5405 SCRIPT_NAME(2):-Copy("RTSDEMO");           999 2
5406 NR_OF_SCRIPTS:=2;                         999 3
5407 SCRIPT_NAME(1):-Copy("INTERFACE");         999 3
5408 end DEFINE_SCRIPT_NAMES;                  999 3 E658

```

```

5410 %COPY MAINPGM
5411 external assembly integer procedure CLOCK ;
5412 external assembly text procedure TODAY,TIMEOFDAY ;comment SIEMENS;
5413 external assembly integer procedure CPUTIME ; comment SIEMENS;
5414 ref(MODULE) IFCODE;
5415 real STARTCLOCK;
5416 integer STARTCPU;
5417 text STARTTIME;
5418
5419 comment ****
5420 ** INITIALISIERUNGEN
5421 ****;
5422
5423 STARTCLOCK:=CLOCK ;
5424 STARTCPU:=CPUTIME;
5425 STARTTIME:=Copy(TIMEOFDAY);
5426 comment ****
5427 ** GENERIEREN DER PROZESSOREN
5428 ****;
5429
5430 for I:=1 step 1 until NR_OF_PROCESSORS do
5431 begin
5432   PROCESSORS(I):=new PROCESSOR(I);
5433   PROCESSORS(I).INITIALIZE;
5434   activate PROCESSORS(I) before Current;
5435 end;
5436 comment ****
5437 ** GENERIEREN DER BUSSE
5438 ****;
5439
5440 for I:=1 step 1 until NR_OF_BUSSES do
5441 begin
5442   BUSSES(I):=new BUS(I);
5443   BUSSES(I).INITIALIZE;
5444   activate BUSSES(I) before Current;
5445 end;
5446 comment ****
5447 ** INTERFACE GENERIEREN UND AUF PROZESSOR 1 LEGEN
5448 ****;
5449
5450 IFCODE:=new SCRIPT1; IFCODE.PRINT_NAME:=-Copy("INTERFACE");
5451 IFCODE.ASS_PROCESSOR:=-PROCESSORS(1);
5452 IFCODE.Into(PROCESSORS(1).MODULE_LIST);
5453
5454 INTERFACE:=new ACB(IFCODE);
5455 INTERFACE.COPY_NR:=1;
5456 INTERFACE.ASS_PROCESSOR:=-PROCESSORS(1);
5457 INTERFACE.Into(PROCESSORS(1).ACB_QUEUE);
5458
5459 comment ****
5460 ** DIE GLOBALE AGENTEN-LISTE WIRD INITIALISIERT
5461 ****;
5462
5463 AGENT_LIST:=new Head;
5464 new AGENT(INTERFACE).Into(AGENT_LIST);
5465 NR_OF_AGENTS(1):=1;
5466
5467 comment ****
5468 ** START DER SIMULATION
5469 ****;
5470
5471 activate PROCESSORS(1) after Current;
5472 Hold(1000000);
5473 PROTOKOL.Close;
5474 end;
5475 BMSOUT.Close;
5476 %ENDCOPY
5477 end CSSA ;
5478 end PGM ;

```

999 3 E3  
999 3 E1

A 59 79 80 81 82\* 83\* 157\* 170 416\* 419 775 789\* 790 794 795 796 798 803 818 819\*  
820 1086\*1087 1089 1090 1149\*1151 1163\*1164 1168 1171 1174\*1178 1180 1183\*1184 1188 1218\*1219 1262\*  
1272 1279 1286 1293 1300 1307 1314 1321 1915\*1916 3902 3942 3947 3948 3949 3950 3952\*3955 4096 4109  
4110 4112 4118 4119\*4121 4122 4124 4130 4131\*4349 4351 4354 4358 4360  
ABS 1071 1072 2052  
ACB 157 416 585 713 775 789 965 978 994 999 1042 1086 1149 1163 1174 1183 1218 1262 1341 1366  
1438 2183 2188 2712 3542 3740 4096 4349 4503 5454  
ACB\_QUEUE 587 603 635 644 749 789 4019 4026 4109 5457  
ACT\_TYPE 3756 3757\*3759 3760  
ACTIVE\_ACB 82 95 97 98 99 101 108 119 126 137 147 158 168 178 188 196 203 210 217 224  
231 237 249 254 259 261 266 268 273 275 280 282 287 289 294 296 301 303 308 310  
315 334 344 355 370 377 384 391 395 402 406 413 417 424 428 435 439 446 451 458  
463 471 479 484 492 500 511 519 530 538 545 553 564 572 585 620 623 624\* 633 640  
644 645 646 652 653 659 666 684 777 796 819 821 822 823 841 845\* 846 1591 1600 1611  
1617 1657 1685 1739 1740 1789 1801 1828 1856 1870 1878 2042 2045 2047 2048 2104 2108 2119 2126  
2148 2152 2163 2170 2192 2196 2207 2214 2236 2240 2251 2258 2280 2284 2295 2302 2324 2341 2363 2367  
2378 2385 2407 2411 2422 2429 2443 2455 2468 2471 2494 2503 2513 2519 2534 2546 2559 2562 2585 2594  
2604 2610 2625 2638 2665 2668 2677 2686 2697 2703 2718 2730 2743 2746 2769 2778 2788 2794 2809 2822  
2835 2838 2861 2870 2881 2887 2902 2914 2927 2930 2953 2962 2972 2978 2993 3006 3019 3022 3045 3054  
3065 3071 3086 3099 3112 3115 3138 3147 3158 3164 3182 5164 5229  
ACTIVE\_CODE 586 646 652 666 667 824\* 825 826 827  
ACTIVE\_TIME 608 614 615 875\* 898 905 906 927\*  
ADD 483  
AGENT 59 790 818 999 3902 3942 5464  
AGENT\_ARRAY 2182 5260  
AGENT\_DENOTA 3930 3970 3974 3985 3989 4000 4005 4017 4024 4058 4062 4073 4077 4254 4265 4353  
AGENT\_ELEMEN 2628 2651 2691 2709 2712  
AGENT\_IDLE 1451 1463 5353  
AGENT\_INITIA 652  
AGENT\_LIST 79 790 819 1002 3947 5463 5464  
AGENT\_POP 423 1701\*2192 2206 2635 2648 2666 2693 4542 4871  
AGENT\_PUSH 416 780 798 1672 2201 2680 2699 2700 2706 2707 4535 4763 4769 4775 4832 4882  
AGENT\_SET 2623 2629 2692 5123 5296  
AGENT\_TOP 296 420 421 4258 984 1293 1700 2192 2628 2631 2644 2671 2691 4538 4816 4847 4859  
AGENT\_VALUE 3542 3605 3958 3960 4847 4882  
AGENTTEL 418 419 984 1042 1043 1264  
AGENTSY 4497\*4893 4918 5023  
AGENTWERT 1366 1376 1672 1700 2196 2207 4816 4832  
AKT\_AS 977 1188 5242\*5243 5246 5247 5248 5252 5253 5254 5258 5259 5260 5264 5265 5266 5267 5270 5271 5272 5276  
5277 5278 5282 5283 5284 5288 5289 5290 5292 5294 5296 5298 5300 5302 5304 5306 5313\*5314 5319 5320\*  
5321\*5327\*5343 5344\*5363\*5373\*5375\*5379 5381\*  
ARRAY\_DESC 1721 1749 5105 5107 5109 5111 5113 5115 5117 5119 5246 5252 5258 5264 5270 5276 5282 5288  
AS 976 1185 1389 1390 5102 5196 5206 5214  
ASS\_PROCESSO 689 717 793 794 821 968 1019 1222 1590 1599 1608 1610 1614 1616 1625 1626 1657 1685 1737 1770  
1789 1801 1828 1854 1867 2042 2045 2047 2048 2103 2107 2112 2117 2125 2147 2151 2156 2161 2169 2191  
2195 2200 2205 2213 2235 2239 2244 2249 2257 2279 2283 2288 2293 2301 2323 2331 2340 2362 2366 2371  
2376 2384 2406 2410 2415 2420 2428 2443 2455 2468 2471 2494 2503 2513 2519 2534 2546 2559 2562 2585  
2594 2604 2610 2625 2638 2665 2668 2677 2686 2697 2703 2718 2730 2743 2746 2769 2778 2788 2794 2809  
2822 2835 2838 2861 2870 2881 2887 2902 2914 2927 2930 2953 2962 2972 2978 2993 3006 3019 3022 3045  
3054 3065 3071 3086 3099 3112 3115 3138 3147 3158 3164 3182 3498 4019 4020 4021 4026 4027 4028 4541  
4575 4750 5163 5183 5229 5353 5386 5451 5456  
ASS\_SCRIPT 646 965 966 1089 3949 4112 4124  
ASSIGN 1638 1651 1655 1677 1866 4407 4418 4483 4662 4843 5162  
ASSIGN\_ALL 1650 1769 2502 2593 2685 2777 2869 2961 3053 3146  
ASSIGN\_ELEME 1751 1785 1821 1847 1883 2123 2167 2211 2255 2299 2338 2382 2426  
ASSIGN\_KEY 1654  
ASSIGNMENT 3821 4650  
ASSIGNSY 3418 4478 4657 4658 4894 4919  
AS1 5102 5242 5243 5246 5247 5248 5252 5253 5254 5258 5259 5260 5264 5265 5266 5270 5271 5272 5276 5277  
5278 5282 5283 5284 5288 5289 5290 5292 5294 5296 5298 5300 5302 5304 5306  
AS3 5196 5313 5314 5319  
AS4 5206 5327  
AS5 5214 5363  
AVG\_WAIT 898 913\*4148  
B 369\* 372 683 699 700 701\* 714 742 743 744\*1063\*1064 1126\*1129  
BEFORE\_CURR 1551 1574 1577 1587 1589  
BLANKS 21 61 504 1457 1478 1534 1779 1841 1944 1986 2001 3377 3894 4193 5083 5085  
BLOCKEDSY 4905 4932 5065  
BMSOUT 19 20 21 22 67\* 70 72 74 76 77 85 104 109 110 233 317 321 342 622 723  
1106 1108 1113 1115 1120 1122 1127 1130 1136 1138 1143 1145 1150 1152 1157 1159 1186 1190 1192 1193  
1199 1200 1203 1204 1207 1224 1227 1229 1240 1241 1242 1243 1244 1253 1404\*1406 1407 1411 1416\*1418  
1419 1423 1428 1429 1556 1558 1564 1567 1644 1645 1724 1731 1891 1895 2044 2481 2482 2486 2488 2572  
2573 2577 2579 2652 2653 2657 2659 2756 2757 2761 2763 2848 2849 2853 2855 2940 2941 2945 2947 3032  
3033 3037 3039 3125 3126 3130 3132 3599 3600 3608 3866 3867 3868 3871 3872 3873 4099\*4100 4102 4106  
4107 4112 4114 4115 4116\*4124 4126 4127 4128\*4133 4136 4138\*4140 4141 4145 4146 4147 4148 4153 4155  
4156 4157\*4158 4159 4164 4168 4359 4367 4386 5475  
BOOL\_ARRAY 2226 5266  
BOOL\_ELEMENT 2721 2755 2783 2800 2803  
BOOL\_POP 376 465 466 473 474 1698\*2236 2250 2727 2740 2744 2784 4870  
BOOL\_PUSH 369 467 475 1608 1614 1625 1626 1660 1662 1670 1680 1812 1814 1824 1836 1838 1850 2245 2469 2560  
2666 2744 2772 2790 2791 2797 2798 2836 2928 3020 3113 4781 4787 4830 4881  
BOOL\_SET 2716 2721 2783 5124 5298  
BOOL\_TOP 282 373 374 378\* 465 466 473 474 480\* 982 1272 1697 2236 2721 2723 2736 2749 2783 4815 4846  
4858

BOOL\_VALUE 3544 3601 4846 4881  
BOOLEL 371 372 982 1030 1031 1263  
BOOLSY 4493\*4893 4918 5022  
BOOLHERT 1367 1375 1670 1697 2240 2251 4815 4830  
BRANCH 5230 5321 5375  
BRANCH\_FLAG 971  
BUFFPOINTER 3187 3208 3209 3210 3211 3212 3213 3214 3222 3223 3224 3225 3226 3227 3228  
BUS 683 714 894 1447 5442  
BUS\_QUEUEING 933 1441 1508  
BUSES 699 742 1447 4144 4319 5442 5443 5444  
B1 464 465 467 472 473 475  
B2 464 466 467 472 474 475  
C 1957 1970 1971\*3254\*3259  
C\_INFILE 3184 3205 3206 3217\*3218 3220\*3221 3232 3238 3239 3244 3264 3279 3293 3313 3322\*3344\*3371\*3401\*3421\*  
3430\*3448 3449\*3452\*3457\*3463\*3892\*3893 3894 3895 3926\*5084 5085 5086 5092 5093 5094\*5095\*  
CALLER 978 4612 4622  
CALLERSY 4905 4931 5064  
CANCEL 667  
CANCELLED 991  
CARDINAL 913 935 1232 1246\*2037 4557 4588 4606  
CB 81 82\* 819 999\*3949 3950 3955  
CHAR 32 1986  
CLEAR 32 36 2505 2596 2688 2780 2872 2964 3056 3149  
CLEAR\_SCREEN 35 64 1477 4255  
CLOCK 3913 4282 4304 4306 5411 5423  
CLOSE 1522 1546 3220 3892 3926 4196 5094 5473 5475  
COLONSY 3422 4430 4446 4471 4899 4934  
COMMAND 3772 5091  
COMMASY 3436 3972 3973 3987 3988 4003 4004 4022 4023 4060 4061 4075 4076 4409 4414 4433 4434 4450 4451 4474  
4475 4641 4643 4895 4920  
COMPARE\_TYPE 3756 3761 4406 4417 4694 4801  
CONCAT 499  
CONFIG 1436 1455 1478 1482 1488 4193 4195 4196  
CONNECTION\_M 699 742 1448 1518  
CONST\_VAR\_DE 3817 3818 4458  
CONSTANT 3539 3681 4432 4436 4448 4453 4463 4468  
CONSTSY 3817 4467 4892 4917 5018  
COPY 13 14 15 245 1938 2007 2133 2177 2221 2265 2309 2348 2392 2436 3267 3465 3679 4432 4436 4448  
4453 4573 4716 4722 4728 4734 4740 4748 4756 4762 4768 4774 4780 4786 4795 4909 4937 4938 4939 4940  
4941 4942 4943 4944 4945 4946 4947 4948 4949 4950 4951 4952 4953 4954 4955 4956 4957 4958 4959 4960  
4961 4962 4963 4964 4965 4966 4967 4968 4969 4970 4971 4972 4973 4974 4975 4976 4977 4978 4980 4981  
4982 4983 4984 4985 4986 4987 4988 4989 4990 4991 4992 4993 4994 4995 4996 4997 4998 4999 5000 5001  
5002 5003 5004 5005 5006 5007 5008 5009 5157 5158 5193 5194 5203 5204 5211 5212 5219 5220 5227 5378  
5405 5407 5425 5450  
COPY\_NR 795 979 1090 3950 5455  
COS 2058  
CPUETIME 3916 4275 4280 5413 5424  
CREATE\_AGENT 1450 1461 4749  
CSSA 25 1527  
CSSAABS 2051 2052  
CSSABLANKS 1942 1943 1944  
CSSACARDINAL 2036 2037  
CSSACHAR 1982 1987  
CSSACLOSEINP 1545  
CSSACOS 2057 2058  
CSSAEMPTY 2032 2033  
CSSAEND\_OF\_F 1542 1543  
CSSAENTIER 2075 2079  
CSSAFLOAT 2019 2020  
CSSAGENSTRIN 1998 2007  
CSSALENGTH 1928 1929  
CSSALN 2063 2066  
CSSAMOD 2082 2088  
CSSANUMBER 1955 1978 1992  
CSSAOPENINPU 1531  
CSSAORD 1946 1948 1951  
CSSAPOS 2010 2011  
CSSAPRED 2016 2017  
CSSAREADREC 1537 1539  
CSSAROUND 2022 2026  
CSSASIN 2054 2055 5318  
CSSASQRT 2069 2072  
CSSASUBSTR 1931 1933 1937 1938  
CSSASUCC 2013 2014  
CSSATAN 2060 2061  
CSSATIME 2029  
CSSAVALUE 1990 1992 1994  
CURR 1551 1561 1562 1569 1570\*1574 1575 1576 1577 1578\*1585 1587 1588 1589 1607 1609 1624 2445 2457 2458  
2459 2469 2472 2473 2474 2475\*2495 2496 2497 2498\*2525 2536 2548 2549 2550 2560 2563 2564 2565 2566\*2586 2587 2588 2589\*2616 2627 2640 2641 2642 2666 2669 2670 2671 2672\*2678 2679 2680 2681\*2709 2720  
2732 2733 2734 2744 2747 2748 2749 2750\*2770 2771 2772 2773\*2800 2811 2824 2825 2826 2836 2839 2840  
2841 2842\*2862 2863 2864 2865\*2893 2904 2916 2917 2918 2928 2931 2932 2933 2934\*2954 2955 2956 2957\*  
2984 2995 3008 3009 3010 3020 3023 3024 3025 3026\*3046 3047 3048 3049\*3077 3088 3101 3102 3103 3113  
3116 3117 3118 3119\*3139 3140 3141 3142\*3170

CURRENT 701 744 805 946 1590 1599 1608 1610 1614 1616 1625 1626 1657 1685 1737 1770 1789 1801 1828 1854  
1867 2030 2042 2045 2047 2048 2103 2107 2112 2117 2125 2147 2151 2156 2161 2169 2191 2195 2200 2205  
2213 2235 2239 2244 2249 2257 2279 2283 2288 2293 2301 2323 2331 2340 2362 2366 2371 2376 2384 2406  
2410 2415 2420 2428 2443 2455 2468 2471 2494 2503 2513 2519 2534 2546 2559 2562 2585 2594 2604 2610  
2625 2638 2665 2688 2697 2703 2718 2730 2743 2746 2769 2778 2788 2794 2809 2822 2835 2838  
2861 2870 2881 2887 2902 2914 2927 2930 2953 2962 2972 2978 2993 3006 3019 3022 3045 3054 3065 3071  
3086 3099 3112 3115 3138 3147 3158 3164 3927 4021 4028 5163 5188 5434 5444 5471  
CURRENT\_FACE 98 129 130 131 133 980 1226 5378  
CURRENT\_LINE 335 336 344 623 992 1202 2046 5240 5249 5255 5261 5267 5273 5279 5285 5291 5293 5295 5297 5299  
5301 5303 5305 5307 5312 5326 5362 5365 5366  
CURRENT\_MSG 240 244 319 323 358 359 361 796 973 4401 4559 4561\*4566 4567 4590 4592\*4593 4608 4610\*4611  
4612 4613 4614 5332 5380  
CURRENT\_OPER 99 101 133 245\* 980 1228 5325 5335  
CURRENT\_PARM 974 1659 1661 1663 1664 1665 1666 1667 1668 1669 1670 1671 1672 1673 1674 1675 1676 1678\*1811 1813  
1823\*1835 1837 1848\*2108 2152 2196 2240 2284 2367 2411 4401 4402 4406 4411\*4412 4417 4827 4828 4829  
4830 4831 4832 4833 4834 4835 4836 4837 4838 4839 4840  
DATA 2095 2113 2120 2128 2132 2139 2157 2164 2172 2176 2183 2201 2208 2216 2220 2227 2245 2252 2260 2264  
2271 2289 2296 2304 2308 2315 2332 2343 2347 2354 2372 2379 2387 2391 2398 2416 2423 2431 2435  
DECLARE 3580 3722  
DEFINE\_SCRIPT 5077 5403  
DELETE 1581 1596  
DELETE\_STATE 3815 4580  
DELETESY 3815 4586 4905 4932 5068  
DEMANDSY 4900 4925 5052  
DENOTESY 3419 4478 4659 4894 4919  
DESC 1749\*1762\*1763\*1764\*1765\*1766\*1772\*1773\*1774\*1775\*1776\*1791\*1792\*1793\*1794\*1795\*1802 1804\*1807 1830\*  
1831\*1832\*1833\*1834\*1855 1857\*1860 1869 1871\*1874 1886\*1887\*1888\*1889\*1890\*1903 1905 1907 1909 2095\*  
2096\*2097\*2098\*2099\*2139\*2140\*2141\*2142\*2143\*2183\*2184\*2185\*2186\*2187\*2227\*2228\*2229\*2230\*2231\*2271\*  
2272\*2273\*2274\*2275\*2315\*2316\*2317\*2318\*2319\*2354\*2355\*2356\*2357\*2358\*2398\*2399\*2400\*2401\*2402\*  
DIGIT 1962 3353 3356 3363 3366 3385 3390  
DISPLAY\_STAT 3822 3824 3852  
DISPLAYSY 3822 3857\*4901 4927 5054  
DIV 518  
DUMP 1171 1180 1183 1552 1569 1632 1639 1646 1723 1754 1885 1916 2479 2570 2650 2754 2846 2938 3030 3123  
5132 5134 5136 5138 5140 5142 5144 5146 5148 5149 5150 5151 5152 5153 5154 5155 5182  
DUMP\_AGENT 1149  
DUMP\_ALL 1643 1715  
DUMP\_ARRAY 1915 5133 5135 5137 5139 5141 5143 5145 5147  
DUMP\_BOOL 1126  
DUMP\_ENUM 1105  
DUMP\_INT 1119 5185  
DUMP\_OPER 1134  
DUMP\_REAL 1112 5201  
DUMP\_RECORD 1714  
DUMP\_RELATIO 1630 5131  
DUMP\_SCRIPT 1156  
DUMP\_STATEME 3828 4259  
DUMP\_STRING 1142 5184  
DUMPSY 3828 4264 4902 4928 5056  
E 1074\*1075 1195 1389 1390  
EINSPRUNG 969 5238\*5337 5346 5351 5353 5370 5384 5386  
ELAPSED\_TIME 1018 2030\*5337\*5346\*5351\*5354 5367\*5370\*5384\*5387  
EMPTY 632 2033 4315  
END\_OF\_NUMBE 3372 3388 3400  
END\_OF\_SECTI 107 153 173 191 5336 5345 5350 5369 5383  
ENDFILE 1543 3218  
ENDFILESY 3428 3495 4891 4916  
ENDOP147 5368  
ENDPROC 1958 1964 1971 1975 1979  
ENTIER 1071 1072 2079  
ENTRY 5232 5238  
ENT1 5233 5239  
ENT2 5233 5371  
ENT3 5233 5338  
ENT4 5233 5347  
ENT5 5233 5352  
ENT6 5233 5355  
ENT7 5233 5385  
ENT8 5233 5388  
ENUM 2010\*2011 2013\*2014 2016\*2017\*  
ENUM\_ARRAY 2314 5278  
ENUM\_ELEMENT 2905 2939 2967 2984 2987  
ENUM\_POP 381 2324 2911 2924 2928 2968  
ENUM\_PUSH 380 2332 2956 2974 2975 2981 2982  
ENUM\_SET 2900 2905 2967 5126 5302  
ENUM\_TOP 275 399 403 989 2324 2905 2907 2920 2933 2967  
ENVIRONMENT 976 5242 5313 5327 5343 5363 5379  
EQUAL 1576 1640 5175 5176  
EVENT 679 752 802 1452 4038 4049 4309 4322  
EVENT\_STATEM 3839 4041  
EVENTSY 3839 4046 4906 4933 5029  
EVTIME 666  
EXPONENT 3189 3377 3392\*3396\*3398  
EXPRESSION 4481 4512 4517 4640 4644 4660 4666 4746

F 118\* 130 133 223\* 227  
FACET4 5324 5381  
FACTOR 3882 4693 4697  
FALSESY 4783 4785 4896 4921 5026  
FIELD 1749 1915 2094 2138 2182 2226 2270 2314 2353 2397 5106 5108 5110 5112 5114 5116 5118 5120  
FIELD\_TYPE 1641 1661 1688 5193 5194  
FIN 3562 3572 3632 3642 3646 3666 3674 3678  
FIND 1603 1609 1615  
FIND\_SEMICOL 3481 3514 5092  
FIRST 79 360 644 715 724 726 736 747 784 785 819 920 1238 1346 1551 1554 1561 1574 1588 1597  
2472 2484 2495 2514 2521 2563 2575 2586 2605 2612 2655 2669 2678 2698 2705 2747 2759 2770 2789 2796  
2839 2851 2862 2882 2889 2931 2943 2954 2973 2980 3023 3035 3046 3066 3073 3116 3128 3139 3159 3166  
3947 4109 4121 4150 4360 4401 4559 4590 4608  
FOUND 786 789 1576 1579 1962 1965 2005 2006 2474 2477 2565 2568 2671 2674 2749 2752 2841 2844 2933 2936  
3025 3028 3118 3121 3951 3955  
FROMSY 4900 4925 5053  
FULL 1343\*1353  
FUNCT3 5311  
GENDATE 10 13 1483 4198  
GENTIME 10 14 1484 4199  
GET\_PARM 1800 4407 4418 4825  
GET\_PARMS 1656 1827  
GETCHAR 1951 1962 1970 1975 2005  
GETINT 1994 3396 3404  
GETREAL 3403  
HEAD 587 588 589 590 602 603 604 605 896 909 972 995 1002 1338 1358 2032 2036 2441 2532 2623  
2716 2807 2900 2991 3084 5463  
HELP\_STATEME 3848 4371  
HELPSY 3848 4392 4901 4927 5057  
HILF 371 372 373 374 385 386 387 388 396 397 398 399 407 408 409 410 418 419 420 421  
429 430 431 432 440 441 442 443 452 453 454 455  
HOLD 661 944 4314 5337 5346 5351 5370 5384 5472  
I 380\* 394\* 397 539 540 541 934 935 1069\*1070\*1071\*1072\*1105\*1107 1119\*1121 1436 1481 1489 1498  
1501 1515 1518 1553 1563\*1565 1641 1649\*1651\*1653\*1655\*1658 1661 1677 1686 1688 1690 1726 1727 1728  
1729 1735 1738 1739 1740 1741\*1743\*1757 1802 1803 1804\*1807\*1855 1856 1857\*1860\*1869 1870 1871\*1874\*  
1957 1960 1966 1967 1969\*1973 2000 2004 2007\*2019\*2020 2051\*2052 2082\*2088 2504 2507 2595 2687  
2690 2779 2782 2871 2874 2963 2966 3055 3058 3148 3151 3198 3314 3336 3338\*3344 3345 3355 3357\*3362\*  
3367\*3369\*3379\*3382\*3394\*3401 3402 3413 3417\*3420\*3427\*3429\*3447\*3448\*3452 3453 3457 3458 3463 3464  
3492 3504 3506 3508 3509 3511 3656\*3660 3662 3670 3680 3856 3864 3870 3879 4096 4104 4105 4143 4144  
4191 4194 4316 4317 4318 4319 4338 4375 4379 4381 4383 4384 4387 4552 4560 4585 4591 4603 4609 5078  
5079\*5162\*5166 5170 5187\*5190 5191 5234\*5236\*5430 5432\*5433 5434 5440 5442\*5443 5444  
ID 38\* 51 697\* 699\* 740\* 742\* 894\* 924 940 3523 3532 3534 3547 3554 3583 3584 3593 3621 3626 3636  
3657 3662 3670 3687 4106 4145 4463 4654 4843 4878 4912  
IDENT 4463 4482\*4483 4043\*4845\*4846\*4847\*4848 4849 4850\*4851\*4852\*  
IDLE 613 701 744 805 904 946 4021 4028 4317 4319  
IDLE\_QUEUE 588 604 845 4121  
IDLE4 5332 5341  
IDLING 751 845 991 1224  
IDNAME 248\* 250 253\* 255 258\* 260 265\* 267 272\* 274 279\* 281 286\* 288 293\* 295 300\* 302 307\* 309  
1105\*1106 1112\*1113 1119\*1120 1126\*1128 1134 1135\*1136 1142\*1143 1149\*1150 1156\*1157 1630 1631\*1632  
1714\*1715 1723\*1724 1885\*1891 1915\*1916 3532\*3558 3567 3582 3598 3624 3660 3749 4523 4884  
IFCODE 5414 5450\*5451 5452 5454  
IMAGE 1404\*1406 1407 1411 1416\*1418 1419 1423 1428 1429 3211 3212 3215 3217 3225 3226 3229 3232 3239 3244  
3895 5086  
INBUFFER 3187 3208 3222 5083  
INCHAR 1508 3294 3306 3309\*3312 3339 3357 3361 3367 3378 3381 3394 3417 3427 3448 3450 3459 5093  
INCREM 2518 2609 2702 2793 2886 2977 3070 3163  
INDEX 1759 1803 1804\*1807 1817\*1822\*1856 1857\*1860 1864\*1870 1871\*1874 1880\*1883\*  
INFILe 1436 1455 1529 1533 3184 3893 5084  
INHERITSY 4527 4528 4905 4932 5066  
INIMAGE 1490 1491 1492 1493 1494 1495 1496 1502 1503 1505 1506 1507 1508 1509 1510 1511 1512 1513 1519 1520  
1538 3206  
ININT 1490 1499 1501 1506 1516 1518  
INIT\_ACTIONS 11 1454  
INITIALIZE 600 909 5433 5443  
INPUTFILE 1529 1533 1534 1538 1539 1543 1546  
INREAL 1491 1507  
INSERT 1595  
INSERT\_SCRIPT 3656 5079  
INSERTKw 3619 5015 5016 5017 5018 5019 5020 5021 5022 5023 5024 5025 5026 5027 5028 5029 5030 5031 5032 5033  
5034 5035 5036 5037 5038 5039 5040 5041 5042 5043 5044 5045 5046 5047 5048 5049 5050 5051 5052 5053  
5054 5055 5056 5057 5058 5059 5060 5061 5062 5063 5064 5065 5066 5067 5068 5069 5070 5071  
INT\_ARRAY 2094 5248  
INT\_CONST 3358 3370 3945 4293 4295 4340 4554 4555 4587 4605 4713 4715 4896 4921  
INT\_ELEMENT 2446 2480 2508 2525 2528  
INT\_POP 381 401 486 487 513 514 521 524 532 533 540 1692\*1739 1740 1809 1862 1876 2104 2118 2452  
2465 2469 2506 2509 2597 2689 2781 2873 2965 3057 3150 4874 5172  
INT\_PUSH 380 394 488 515 525 534 541 1666 2113 2497 2500 2515 2516 2522 2523 2591 2683 2775 2867 2959  
3051 3144 4717 4838 4886 5191 5244 5245 5250 5251 5256 5257 5262 5263 5268 5269 5274 5275 5280 5281  
5286 5287  
INT\_SET 2441 2446 2508 5121 5292  
INT\_TOP 268 398 399 403\* 486 487 513 514 521 524 532 533 540 988 1279 1692 1739 1740 1803 1856  
1870 2104 2446 2448 2461 2474 2506 2508 2597 2689 2781 2873 2965 3057 3150 4821 4851 4862 5171  
INT\_VALUE 3540 3603 4851 4886

INT\_WEIGHT 591 631\* 654 673 674 681 712 774 817 844 859 874 3497 4540 4574 4749 5353 5386  
INTEGER\_VALU 3404 3688 3950 4296 4342 4557\*4560 4588\*4591 4606\*4609 4717  
INTEL 396 397 988 989 1036 1037 1266  
INTERFACE 624 1438 2047 4769 5454 5455 5456 5457 5464  
INTERFACESY 3935 3936 4765 4767 4898 4923 5039  
INTERRUPT 593 631 673  
INTERRUPTED 4312 4325 4329  
INTEXT 1482 1539 3345 3402 3458 3464 4195  
INTO 635 694 747 749 788 789 790 845 916 945 1689 1796 2446 2508 2537 2599 2629 2692 2721 2783  
2813 2876 2905 2967 2997 3060 3090 3153 4019 4026 4822 5452 5457 5464  
INTSY 4495\*4893 4918 5016  
INTWERT 1363 1373 1666 1692 2108 2119 4821 4838  
I1 485 486 488 512 513 515 520 521 522 525 531 532 534 1760 1762 1767 1772 1780 1785 1791  
1797 1830 1842 1847 1886 1892 1894 1900\*1902 2110 2111 2113 2115 2116 2120 2123 2124 2128 2130 2131  
2132 2154 2155 2157 2159 2160 2164 2167 2168 2172 2174 2175 2176 2198 2199 2201 2203 2204 2208 2211 2212  
2212 2216 2218 2219 2220 2242 2243 2245 2247 2248 2252 2255 2256 2260 2262 2263 2264 2286 2287 2289 2291  
2291 2292 2296 2299 2300 2304 2306 2307 2308 2329 2330 2332 2334 2335 2338 2339 2343 2345 2346 2347 2347  
2369 2370 2372 2374 2375 2379 2382 2383 2387 2389 2390 2391 2413 2414 2416 2418 2419 2423 2426 2427  
2431 2433 2434 2435  
I2 485 487 488 512 514 515 520 524 525 531 533 534 1760 1763 1767 1773 1780 1785 1792 1797  
1831 1842 1847 1887 1892 1894 1900\*1904 2110 2111 2113 2115 2116 2120 2123 2124 2128 2130 2131 2132  
2154 2155 2157 2159 2160 2164 2167 2168 2172 2174 2175 2176 2198 2199 2201 2203 2204 2208 2211 2212  
2216 2218 2219 2220 2242 2243 2245 2247 2248 2252 2255 2256 2260 2262 2263 2264 2286 2287 2289 2291  
2292 2296 2299 2300 2304 2306 2307 2308 2329 2330 2332 2334 2335 2338 2339 2343 2345 2346 2347 2369  
2370 2372 2374 2375 2379 2382 2383 2387 2389 2390 2391 2413 2414 2416 2418 2419 2423 2426 2427 2431  
2433 2434 2435  
I3 1760 1764 1767 1774 1780 1785 1793 1797 1832 1842 1847 1888 1892 1894 1900\*1906 2110 2111 2113 2115  
2116 2120 2123 2124 2128 2130 2131 2132 2154 2155 2157 2159 2160 2164 2167 2168 2172 2174 2175 2176  
2198 2199 2201 2203 2204 2208 2211 2212 2216 2218 2219 2220 2242 2243 2245 2247 2248 2252 2255 2256  
2260 2262 2263 2264 2286 2287 2289 2291 2292 2296 2299 2300 2304 2306 2307 2308 2329 2330 2332 2334  
2335 2338 2339 2343 2345 2346 2347 2369 2370 2372 2374 2375 2379 2382 2383 2387 2389 2390 2391 2413  
2414 2416 2418 2419 2423 2426 2427 2431 2433 2434 2435  
I4 1760 1765 1767 1775 1780 1785 1794 1797 1833 1842 1847 1889 1892 1894 1900\*1908 2110 2111 2113 2115  
2116 2120 2123 2124 2128 2130 2131 2132 2154 2155 2157 2159 2160 2164 2167 2168 2172 2174 2175 2176  
2198 2199 2201 2203 2204 2208 2211 2212 2216 2218 2219 2220 2242 2243 2245 2247 2248 2252 2255 2256  
2260 2262 2263 2264 2286 2287 2289 2291 2292 2296 2299 2300 2304 2306 2307 2308 2329 2330 2332 2334  
2335 2338 2339 2343 2345 2346 2347 2369 2370 2372 2374 2375 2379 2382 2383 2387 2389 2390 2391 2413  
2414 2416 2418 2419 2423 2426 2427 2431 2433 2434 2435  
I5 1760 1766 1767 1776 1780 1785 1795 1797 1834 1842 1847 1889 1892 1894 1900\*1910 2110 2111 2113 2115  
2116 2120 2123 2124 2128 2130 2131 2132 2154 2155 2157 2159 2160 2164 2167 2168 2172 2174 2175 2176  
2198 2199 2201 2203 2204 2208 2211 2212 2216 2218 2219 2220 2242 2243 2245 2247 2248 2252 2255 2256  
2260 2262 2263 2264 2286 2287 2289 2291 2292 2296 2299 2300 2304 2306 2307 2308 2329 2330 2332 2334  
2335 2338 2339 2343 2345 2346 2347 2369 2370 2372 2374 2375 2379 2382 2383 2387 2389 2390 2391 2413  
2414 2416 2418 2419 2423 2426 2427 2431 2433 2434 2435  
J 1489 1500 1501 1517 1518 1957 1973 2082\*2084 2088 3255  
KEYWORD 3535 3559 3597 3630 3640  
KWNNAME 3619 3620\*3624 3626 3636  
L 202\* 206\* 209\* 213\*1098\*1099 1942\*1943 1944  
LAGENT\_IDLE 598 837  
LAST 758 760 1551 1589\*1597 1598 2119 2163 2207 2251 2295 2378 2422  
LAST\_ACTIVAT 608 615 875 877 898 906 927 929  
LAST\_AS 1209 1389 1390 5320 5373  
LAST\_IDENT 4463 4472 4482  
LAST\_TARGET 4503 4535 4538  
LASTCHAR 3308 3309 3310 3311  
LASTID 3269 3412 3547 3556 3564 3570  
LASTSYMB 3267 3582 3583 3584 3686 3893 4449 4454 4500  
LBOUND 1722 1727 1740 1741 1743 1762 1763 1764 1765 1766 1767 1772 1773 1774 1775 1776 1791 1792 1793 1794 1795  
1804 1830 1831 1832 1833 1834 1857 1871 1886 1887 1888 1889 1890 2095 2096 2097 2098 2099 2139 2140  
2141 2142 2143 2183 2184 2185 2186 2187 2227 2228 2229 2230 2231 2271 2272 2273 2274 2275 2315 2316  
2317 2318 2319 2354 2355 2356 2357 2358 2398 2399 2400 2401 2402  
LBRACKSY 3442 3941 3944 4400 4626 4637 4639 4894 4919  
LCREATE\_AGEN 596 773  
LEFT 3534 3567 3583 3596\*3625 3626 3627 3628 3633 3661 3662 3663 3664 3667  
LEFT\_TIME 652 653 666 970  
LENGTH 504\* 505 506\*1240 1929 1933 1936\*1948 1958 1960 3213 3214 3227 3228 4112 4124 4153  
LETTER 3335  
LIDLE 645 868  
LIMIT\_STATEM 3825 4334  
LIMITSY 3825 4339 4902 4927 5055  
LINE 333\* 335 336  
LINK 965 999 1335 1361 2528 2619 2712 2803 2896 2987 3080 3173  
LN 1071 1072 2066  
LOCATE 1573 1584 1606 1623 2444 2456 2469 2470 2535 2547 2560 2561 2626 2639 2666 2667 2719 2731 2744 2745  
2810 2823 2836 2837 2903 2915 2928 2929 2994 3007 3020 3021 3087 3100 3113 3114  
LOG\_AND 462  
LOG\_NOT 478  
LOG\_OR 470  
LOOKUP 3346 3553  
LOOP\_INDEX 2458\*2514 2515 2520\*2521 2522 2525 2549\*2605 2606 2611\*2612 2613 2616 2641\*2698 2699 2704\*2705 2706  
2709 2733\*2789 2790 2795\*2796 2797 2800 2825\*2882 2883 2888\*2889 2890 2893 2917\*2973 2974 2979\*2980  
2981 2984 3009\*3066 3067 3072\*3073 3074 3077 3102\*3159 3160 3165\*3166 3167 3170  
LOW 1931\*1933 1935\*1937 1938\*  
LRECEIVE\_MES 595 632 711

LSEND\_MESSAGE 594 678  
LTERMINATION 597 815  
LTIMESLICE\_R 593 853  
M 4508 453 683 684 685 689 694 698 700 714 715 716 741 743 775 785\* 786 787 788 789  
791\* 793 797 897 911\* 916 920 921 936\* 942 945\* 946\* 1220 1238 1239 1240 1246 1247 1248 1250\*  
4349 4360 4361 4364 4365\*  
MAILBOX 360 747 758 760 972 995 1238 1246 4360 4557 4559 4588 4590 4606 4608  
MAILBOX\_STAT 3823 4345  
MAILBOXSY 3823 4350 4902 4928 5059  
MAIN 3927  
MAIN\_ENTRY 5228 5338 5347 5352 5355 5371 5385 5388  
MESS 4096 4150 4151 4153 4161 4162\*  
MESSAGE 4513 4563 4630 4747  
MESSAGE\_LIST 590 602 632 694 715 724 726 736 747 896 909 913 916 920 935 945 4150 4315  
MINUS 537 4802  
MINUSSY 3440 4712\* 4902 4928  
MOD 1985 2088  
MODULE 26 586 775 966 1014 1590 1599 1608 1610 1614 1616 1625 1626 1657 1685 1737 1770 1789 1801 1828  
1854 1867 2030 2042 2045 2047 2048 2103 2107 2112 2117 2125 2147 2151 2156 2161 2169 2191 2195 2200  
2205 2213 2235 2239 2244 2249 2257 2279 2283 2288 2293 2301 2323 2331 2340 2362 2366 2371 2376 2384  
2406 2410 2415 2420 2428 2443 2455 2468 2471 2494 2503 2513 2519 2534 2546 2559 2562 2585 2594 2604  
2610 2625 2638 2665 2668 2677 2686 2697 2703 2718 2730 2743 2746 2769 2778 2788 2794 2809 2822 2835  
2838 2861 2870 2881 2887 2902 2914 2927 2930 2953 2962 2972 2978 2993 3006 3019 3022 3045 3054 3065  
3071 3086 3099 3112 3115 3138 3147 3158 3164 3181 5163 5188 5224 5395 5414  
MODULE\_LIST 589 605 784 785 788 5452  
MORE 3392 3396  
MSG 450 683 714 724 736 758 760 897 911 973 1051 1220 1335 4096 4349 4636  
MSG\_NR 4349 4362\* 4363  
MSG\_POP 457 4542 4576 4751  
MSG\_PUSH 450 4636  
MSG\_SPECIFIE 4552 4556 4565  
MSG\_TOP 149 151 171 454 455 459\* 684 796 990 1232 1233 1234 1321 1689 1796 2119 2163 2207 2251 2295  
2378 2422 4513 4521 4529 4538 4539 4566 4567 4569 4570 4572 4573 4822  
MSGEL 452 453 990 1051 1052 1270  
MULT 510  
N 1552\* 1556 1564 1643\* 1644 2479\* 2481 2570\* 2572 2650\* 2652 2754\* 2756 2846\* 2848 2938\* 2940 3030\* 3032 3123\*  
3125  
NEGATIVE 4711 4712 4798  
NET\_ACTIVE 4291 4315 4317 4319 4320  
NEW\_IDENTIFI 3717 4431 4435 4447 4452 4473 4476  
NEW\_MODULE 26 787 5395 5397 5398  
NEW\_PARM 1687 1688 1689 1691 1692 1693 1694 1696 1697 1699 1700 1702 1703 1705 1706 1708 1709 3185 4813 4814  
4815 4816 4818 4819 4820 4821 4822  
NEW\_STATEMEN 333  
NEW\_TRACE\_SE 94 140 161 181 5241 5339 5348 5357 5372 5390  
NEW\_VALUE 2100 2104 2108 2127 2128 2144 2148 2152 2171 2172 2188 2192 2196 2215 2216 2232 2236 2240 2259 2260  
2276 2280 2284 2303 2304 2320 2324 2342 2343 2359 2363 2367 2386 2387 2403 2407 2411 2430 2431  
NEWIDSY 3268 3570 3718 4901 4926  
NEWLINE 3204 3284 3297  
NEWREC 1717  
NEWSY 4743 4745 4900 4925 5051  
NEXT 373 378 387 392 398 403 409 414 420 425 431 436 442 447 454 459 1031 1034 1037 1040  
1043 1046 1049 1052 1275 1282 1289 1296 1303 1310 1317 1324 1570 1578 1587\* 1588 1597 1642 3534 3585  
3631 3641 3665 3673 4482  
NEXT\_CREATIO 776 799\* 800 801 1449 1465  
NEXT\_INPUT 5091\* 5097  
NEXT\_MESSAGE 354 5331  
NEXT\_REC 1582  
NEXT\_SYMBOL 3274 3319  
NEXTCHAR 3290 3291 3293 3294 3295 3298 3305 3306 3307 3309 3310 3311 3312 3321 3335 3337\* 3339 3341\* 3353 3356  
3357 3359 3361 3363 3366 3367 3375 3378 3380\* 3381 3385 3390 3393 3394 3414 3416 3417 3418 3419 3426  
3427 3428 3436 3438 3440 3442 3444 3446 3450 3451 3462 3470 4914 5092 5093  
NEXTSYMB 3197 3709 3721 3812 3891 4480  
NEXT4 5330 5333 5340 5349 5358  
NOAGENT 780 1438 1468 4763  
NOAGENTSY 4759 4761 4898 4923 5036  
NOEVENT\_STAT 3840 4032  
NOEVENTSY 3840 4037 4906 4933 5032  
NOFOUND 4595  
NOMATCH 1660 1662 1681 1812 1814 1825 1836 1838 1851 4627  
NOOBSERVE\_ST 3831 4212  
NOOBSERVE\$Y 3831 4217 4903 4930 5044  
NOOPER 1439 1469  
NOOPERSY 4753 4755 4905 4932 5067  
NOPROT\_STATE 3844 4172  
NOPROTSY 3844 4177 4903 4929 5034  
NOSCRIPT 1437 1467 4741  
NOSCRIPTSY 4737 4739 4898 4923 5037  
NOSNAPSHOT\_S 3834 4052  
NOSNAPSHOTSY 3834 4057 4904 4930 5042  
NOSYSTEM\_STA 3832 4230  
NOSYSTEMSY 3832 4235 4907 4935 5046  
NOT\_FIRST\_TI 1771 1778 1782 1829 1840 1844  
NOT\_FOUND 783 784 787

NOTRACE\_STAT 3838 3964  
NOTRACESY 3838 3969 4903 4929 5028  
NR\_OF\_ACBS 791\* 824\* 825 1016  
NR\_OF\_AGENTS 28 792\* 795 5401 5465  
NR\_OF\_BUSSES 1445 1506 4143 4318 5440  
NR\_OF\_DIMS 1721\*1726 1729 1738 1802 1855 1869 1903 1905 1907 1909  
NR\_OF\_FIELDS 1636\*1641 1649 1651 1658 1686  
NR\_OF\_KEYS 1636\*1653 1655  
NR\_OF\_MSGS 899 913 914 915\*4147  
NR\_OF\_PROCES 800 1442 1490 1498 1500 1515 1517 4104 4316 5430  
NR\_OF\_SCRIPT 5079 5402 5406  
NUM 1982\*1984\*1985\*1986 1998\*2002  
O 438\* 441 1077\*1078 1134 1135 1137  
OBSERVE 120 138 159 179 232 753 838 1452 4218 4227  
OBSERVE\_DUMP 1163 5308 5317 5328 5364  
OBSERVE\_STAT 3829 4221  
OBSERVERSY 3829 4226 4903 4930 5043  
OLDIDSY 3268 3571 3821 3940 4352 4404 4405 4415 4416 4520 4655 4791 4901 4926  
OPEN 21 1457 1478 1534 3894 4193 5085  
OPER 245 1234 1240 1248 1337 1345 4153 4161 4513 4573  
OPER\_ARRAY 2353 5284  
OPER\_DECL 3819 4440  
OPER\_ELEMENT 2996 3031 3059 3077 3080  
OPER\_POP 445 1707\*2363 2377 3003 3016 3020 3061 4542 4872  
OPER\_PUSH 438 1674 2372 3048 3067 3068 3074 4757 4834 4883 4884  
OPER\_SET 2991 2997 3060 5127 5304  
OPER\_TOP 310 442 443 447\* 985 1314 1706 2363 2996 2999 3012 3025 3059 4513 4818 4849 4860  
OPER\_VALUE 3543 3607 4449 4454 4849 4883  
OPEREL 440 441 985 1048 1049 1269  
OPERSY 3819 4445 4498\*4901 4926 5040  
OPERWERT 1369 1379 1674 1706 2367 2378 4818 4834  
OPER147 5344 5361  
OUT 635 726 820 822 826 2459 2550 2642 2734 2826 2918 3010 3103 4002 4007 4593  
OUTCHAR 32 51 67 77 85 104 110 237 924 940 1081 1083 1350 1372 1409 1486 1564 1729\*1779 1841  
2481 2572 2652 2756 2848 2940 3032 3125 3259 3507 3510 3511 3863 3864 3869 3870\*3878 3879\*4099 4101  
4102\*4107 4114 4115\*4116 4126 4127\*4128 4133 4136\*4138 4140 4141\*4146 4155 4156\*4157 4158 4159 4161  
4164 4167 4168\*4201 4359 4363 4367 4382 4385 4387  
OUTFILE 19 20 1434 1435 1456  
OUTFIX 1060 1067 3913 3916 3918 4148 4280 4282  
OUTIMAGE 33 1409 1414 1421 1426 1482 1484 1486\*3234 3240 3246 3905 3909 4192 4195 4197 4199 4201\*4204 4207\*  
OUTINT 344 623 674 1070 1071 1072 1202 2045 3921 3924  
OUTLINE 66 67 76 78 86\* 96 104 109 110\* 131 152 172 190 199 206 213 220 227 242 262  
269 176 283 290 297 304 311 321 325 619 621 623 641 659 674 722 725 737 761 842  
857 872 881 925 942 1109 1116 1123 1131 1139 1146 1153 1160 1166 1169 1176 1179 1187 1193 1200  
1204 1205 1208 1211 1241 1254 1277 1284 1291 1298 1305 1312 1319 1323 1326\*1401 1559 1568 1592 1601  
1612 1618 1646 1733 1897 2041 2043 2046 2127 2171 2215 2259 2303 2342 2386 2430 2449 2462 2483 2489  
2540 2553 2574 2580 2632 2645 2654 2660 2724 2737 2758 2764 2816 2829 2850 2908 2921 2942 2948  
3000 3013 3034 3040 3093 3106 3127 3133 3256 3259 3283 3478 3479 3480 3493 3496 3502 3513 3609 3864  
3868 3874 3879 3880 3905 3907 3909\*3910 3912 3914 3917 3919 3922 3925 4098 4099 4101 4103 4115 4127  
4134 4137\*4138 4140 4142 4156 4165 4169\*4284 4311 4324 4358 4359 4364 4367\*4377 4386 4390 4617 4618  
4620 4624 4864  
OUTTEXT 51 65 66 68 69 70 71 72 73 74 75 76 77\* 85\* 96\* 97 98 100 101 103  
124\* 126 128 129\* 131 142 145\* 147 150 163 166\* 168 183 186\* 188 198 199 205 206 212  
213 219 220 226 227 235\* 239 241 250 255 260\* 267\* 274\* 281\* 288\* 295\* 302\* 309\* 318 320  
322 343 345 619\* 621\* 622 638\* 640 657\* 659 674 719 720 721 733 734 735 755 756 757  
759 840\* 842 856 857 871\* 872 880\* 881 923\* 925 939\* 941 1064 1075\*1078 1082 1088 1089 1090  
1091 1095 1096 1099 1106\*1108 1113\*1115 1120\*1122 1128\*1130 1136\*1138 1143\*1145 1150\*1152 1157\*1159  
1166\*1167 1176\*1177 1187 1191\*1192\*1194 1196\*1197 1199 1201 1203 1205 1208 1221 1224 1225 1226\*1227  
1228\*1229 1231 1234 1241 1242\*1243 1244 1248 1249 1253 1271 1274 1278 1281 1285 1288 1292 1295 1299  
1302 1306 1309 1313 1316 1320 1345 1352 1354 1355 1372 1406 1410 1411 1418 1422 1423 1482 1483\*1484\*  
1485 1486 1556\*1557 1558 1564\*1566 1567 1592 1601 1612 1618 1644\*1645 1724\*1725 1727 1732 1779 1781  
1818 1841 1843 1881 1891\*1893 1896 1901 1904 1906 1908 1910 1911 2041\*2043\*2044 2449 2462 2481\*2482  
2486 2488 2540 2553 2572\*2573 2577 2579 2632 2645 2652\*2653 2657 2659 2724 2737 2756\*2757 2761 2763  
2816 2829 2848\*2849 2853 2855 2908 2921 2940\*2941 2945 2947 3000 3013 3032\*3033 3037 3039 3093 3106  
3125\*3126 3130 3132 3231 3232 3233 3239 3243 3244 3245 3257 3258 3259\*3280 3281 3282 3478 3479\*3494  
3496 3500\*3501 3503 3509 3598\*3599\*3600 3608 3865\*3866\*3867\*3868 3871 3872 3873 3906 3908 3910\*3911\*  
3912\*3913 3914 3915 3917 3918 3919 3920 3923 4098\*4100 4106\*4108 4118 4130 4139 4145\*4147 4148 4149  
4195 4198\*4199\*4200 4201 4202 4203 4204\*4205 4206 4278\*4279 4281 4282 4283 4310 4311 4323 4324 4358\*  
4363 4376 4383 4616 4618 4621  
OWNMODE 786 1017 3949 4735 4910 5226  
OWNMODESY 4731 4733 4898 4923 5038  
P 216\* 220 775 776 784 785 788 789 793 794 805\*1344 1346 1347 1348 1349\*1350  
P\_AGENTTEL 1264 1\*93 1294 1295 1296\*  
P\_BOOLEL 1263 1272 1273 1274 1275\*  
P\_INTEL 1266 1279 1280 1281 1282\*  
P\_MSGEL 1270 1321 1322 1323 1324\*  
P\_OPEREL 1269 1314 1315 1316 1317\*  
P\_REALEL 1267 1286 1287 1288 1289\*  
P\_SCRIPTEL 1265 1300 1301 1302 1303\*  
P\_STRINGEL 1268 1307 1308 1309 1310\*  
PARAM 974 1344 1361 1687 1688 1796 2119 2163 2207 2251 2295 2378 2422 3185 4813  
PARMS 1232 1246 1338 1346 1358 1689 1796 2119 2163 2207 2251 2295 2378 2422 4401 4822  
PASSED\_TOKEN 3506 3508 3509 3700 3735 4381 4383 4384  
PASSIVATE 876 928

PASSOP147 5374  
PATTERN 4395 4626  
POINTER 3554 3555 3557 3558 3559 3560 3564 3566 3567\*3568\*3621 3622 3624 3625 3626 3627\*3628\*3629 3630 3631  
3633\*3635 3636 3637\*3638\*3639 3640 3641 3643\*3644 3657 3658 3660 3661 3662 3663\*3664\*3665 3667\*3669  
3670 3671\*3672\*3673 3675\*3678 3679 3680 3681  
POP 4484 4640 4644 4663 4867  
PORT 980 1340 1355 4523 4531\*4566\*4569\*4614\*  
PORT\_DECL 3820 4424  
PORTSY 3820 4429 4899 4927 5021  
POS 67 77 85 104 109 110 1240 1404\*1407 1416\*1419 1966 3279 3293 3313 3322 3344 3371 3401 3421  
3430 3448 3449 3452 3457 3463 4099 4102 4112 4124 4136 4138 4141 4153 4168 4359 4367 4386 5092  
POSITION 1403 1404 1406 1407 1411 1412\*1416 1418 1419 1423 1424\*1426  
POT 529  
PRED 2458 2549 2641 2733 2825 2917 3009 3102 3534 3585 3627 3637 3663 3671  
PREV 3269 3547 3566  
PRINT 149 171 240 323 724 736 758 942 1233 1247 1323 1343 1348 1370 1753 1894 2130 2174 2218 2262  
2306 2345 2389 2433 3593 3596 3611 3876 4364  
PRINT\_AGENT 97 126 147 151 168 170 188 237 296 620 640 659 756 760 841 1086 1151 1168 1178 1223  
1295 1376 2042 2215 2220 2631 2644 2658 3281 3605 4118 4130 4358 4617 4622 4859  
PRINT\_BOOL 282 1063 1129 1274 1375 2259 2264 2723 2736 2762 3601 4858  
PRINT\_DUMP 624 1174 2048 4267  
PRINT\_ENUM 275 1074 1107 2342 2347 2907 2920 2946  
PRINT\_ID 50 125 146 167 187 236 639 658 720 734 756 840 856 871 880 1222  
PRINT\_INDICE 1780 1816 1842 1879 1892 1900  
PRINT\_INT 51 268 924 940 1069 1075 1090 1121 1281 1373 1565 1727 1728 1902 1904 1906 1908 1910 2127 2132  
2448 2461 2487 3603 4106 4107 4145 4146 4147 4363 4862  
PRINT\_LINENO 122 141 162 182 197 204 211 218 225 250 255 260 267 274 281 288 295 302 309 337  
341 1592 1601 1612 1618 2448 2461 2539 2552 2631 2644 2723 2736 2815 2828 2907 2920 2999 3012 3092  
3105  
PRINT\_LITERA 1098  
PRINT\_NAME 797 1015 1089 4112 4124 4909 5227 5450  
PRINT\_OPER 310 1077 1137 1316 1345 1379 2386 2391 2999 3012 3038 3607 4161 4860  
PRINT\_REAL 261 1066 1114 1288 1374 2171 2176 2539 2552 2578 3602 4863  
PRINT\_SCRIPT 303 1094 1158 1302 1378 2430 2435 3092 3105 3131 3606 4861  
PRINT\_STACKS 1262 4256  
PRINT\_STATUS 82 1218  
PRINT\_STRING 289 1080 1144 1309 1377 2303 2308 2815 2828 2854 3604 4857  
PRINT\_TIME 65 96 124 145 166 186 235 619 638 657 719 733 755 840 856 871 880 923 939 1059  
1166 1176 2041 3280 4098 4310 4323  
PRINT\_TOP 3882 4855  
PRINT\_VARS 1206 1391 5129 5199 5208 5216  
PROCESS 38 894 1014  
PROCESSOR 38 682 694 697 699 713 729 740 742 775 968 1019 1342 1443 5432  
PROCESSORS 697 740 776 1443 4105 4315 4317 5432 5433 5434 5451 5452 5456 5457 5471  
PROT 1405 1434 3230 3242 4178 4187 4189 4190  
PROT\_STATEME 3841 4181  
PROTOKOL 1435 1456 1457 4189 5473  
PROTSY 3841 4186 4903 4929 5033  
PUSH 1637 1649 1653 1690 1853 4792 4878 5187  
PUSH\_ALL 1648 1761 2493 2584 2676 2768 2860 2952 3044 3137  
PUSH\_ELEMENT 1750 1767 1864 2110 2154 2198 2242 2286 2329 2369 2413  
PUSH\_KEY 1652  
PUT 2442 2533 2624 2717 2808 2901 2992 3085  
PUTCHAR 1986 3393  
PUTINT 2002  
QMARKSY 3438 3824 3858 4901 4927  
R 383\* 386 573 574 575 1066\*1067 1112\*1114 1630 1631 1632 1714\*1715 2022\*2024\*2026 2054\*2055 2057\*  
2058 2060\*2061 2063\*2065 2066 2069\*2071 2072 2075\*2077\*2079  
RANDINT 935  
RANK 1951 3341\*  
RAS 1392 5321\*5344 5375\*5381  
RAS\_LINE 1202\*1392 5344 5381  
RAS1 5231 5344  
RAS2 5231 5381  
RBRACKSY 3444 3946 4421 4646 4894 4919  
READ 3707 3857 3858 3890 3903 3936 3938 3940 3944 3945 3946 3969 3973 3984 3988 3999 4004 4016 4023 4037  
4046 4047 4057 4061 4072 4076 4087 4097 4177 4186 4217 4226 4235 4244 4253 4264 4276 4292 4295 4300  
4339 4340 4350 4392 4400 4405 4414 4416 4421 4429 4430 4434 4445 4446 4451 4465 4467 4471 4475 4493  
4494 4495 4496 4497 4498 4499 4511 4516 4519\*4520 4528 4553 4555 4586 4587 4604 4605 4639 4643 4646  
4655 4658 4659 4712 4715 4721 4727 4733 4739 4745 4755 4761 4767 4773 4779 4785 4791  
READ\_PARM 1755 1820 1846 2106 2150 2194 2238 2282 2326 2365 2409  
READ\_STATEME 3843 3885  
READY 3843 3890 4904 4931 5061  
REAL\_ARRAY 2138 5254  
REAL\_CONST 3365 3384 3403 4298 4300 4719 4721 4893 4924  
REAL\_ELEMENT 2537 2571 2599 2616 2619  
REAL\_POP 390 494 495 547 548 555 558 566 567 574 1695\*2148 2162 2543 2556 2560 2600 4875 5316  
REAL\_PUSH 383 496 549 559 568 575 1668 2157 2588 2606 2607 2613 2614 4723 4840 4887 5318  
REAL\_SET 2532 2537 2599 5122 5294  
REAL\_TOP 261 387 388 392\*494 495 547 548 555 558 566 567 574 987 1286 1694 2148 2537 2539 2552  
2565 2599 4820 4852 4863 5314  
REAL\_VALUE 3403 3541 3602 3689 4301 4723 4852 4887  
REALADD 491  
REALDIV 552

REALEL 385 386 987 1033 1034 1267  
REALMINUS 571 4799  
REALMULT 544  
REALPOT 563  
REALSY 4494\*4893 4932 5019  
REALWERT 1364 1374 1668 1694 2152 2163 4820 4840  
RECEIVE\_MESS 1450 1460  
RECEIVE\_STAT 3814 4598  
RECEIVER 151 685 689 713 716\* 717 747 748 751 756 758 760 1341 4538 4567 4570  
RECEIVESY 3814 4604 4899 4924 5049  
RECORD 1551 1573 1581 1582 1595 1603\*1622 1636 1642 1714 1717 5160  
RECORD104 5160 5175  
RELATION 1550 1631 5104 5243  
RELEASEDATE 10 15 1486 4201  
REMOVE 2454 2545 2637 2729 2821 2913 3005 3098  
REPLY 61\* 62  
REPLY\_EXPECT 1339 1353 4522 4530 4611  
REPLY\_STATEM 3816 4547  
REPLYSY 3816 4518 4519 4553 4904 4931 5063  
REQ\_TYPE 3756 3757\*3759 3760  
RES\_AGENT 3740 3936 3938 3955 3960 3971 3975 3986 3990 4001 4002 4006 4007 4018 4019\*4020 4025 4026\*4027 4059  
4063 4074 4078 4256 4267 4354  
RES\_TYPE 3739 3882 4473 4476 4481 4484 4500 4640 4644 4694 4797 4799 4801 4813 4814 4815 4816 4817 4819 4820  
4821  
RESWORD 3190 3326 3561  
RETURN 5234 5321 5375  
RIGHT 3534 3568 3584 3611\*3635 3636 3637 3638 3643 3669 3670 3671 3672 3675  
ROUTING\_MATR 697 740 1444 1501  
RUN\_STATEMEN 3826 4287  
RUNSY 3826 4292 4906 4934 5015  
RUNTIME\_ERRO 523 556 618 686 691 779 1742 1805 1858 1872 2025 2040 2065 2071 2078 2086  
R1 493 494 496 546 547 549 554 555 556 559 565 566 568  
R2 493 495 496 546 548 549 554 558 559 565 567 568  
S 195\* 199 405\* 408 427\* 430 1080\*1082 1094\*1095 1096 1142\*1144 1156\*1158  
SAVE\_NEW\_VAL 1752 1784 1868 2102 2146 2190 2234 2278 2322 2361 2405  
SCHEDULER 631 687 692 705 727 765 781 807 830 847 861 883  
SCOPE\_NAME 1192 1197 1393 5158 5204 5212 5220  
SCOPE\_TYPE 1191 1196 1393 5157 5203 5211 5219  
SCR\_ID 5395\*5397 5398  
SCR\_NR 775 777 778 783 786 787 792\* 795 797  
SCRIPT\_ARRAY 2397 5290  
SCRIPT\_ELEME 3089 3124 3152 3170 3173  
SCRIPT\_NAME 27 797 1096 3660 3662 3670 5400 5405 5407  
SCRIPT\_POP 412 1704\*2407 2421 3096 3109 3113 3154 4751 4873  
SCRIPT\_PUSH 405 1676 2416 3141 3160 3161 3167 3168 4735 4741 4836 4885  
SCRIPT\_SET 3084 3090 3153 5128 5306  
SCRIPT\_TOP 303 409 410 414\* 777 986 1300 1703 2407 3089 3092 3105 3118 3152 4819 4850 4861  
SCRIPT\_VALUE 3540 3606 3680 3949 4850 4885  
SCRIPTEL 407 408 986 1039 1040 1265  
SCRIPTSY 4499 4897 4922 5035  
SCRIPTWERT 1368 1378 1676 1703 2411 2422 4819 4836  
SCRIPT1 3181 5397 5450  
SCRIPT2 5224 5398  
SELF 978 994 1223 3281 3936 3938 4351 4524 4530 4531 4532 4539 4572 4775 5308 5317 5328 5364  
SELFSY 3937 3938 4771 4773 4898 4923 5047  
SEM\_ERROR 3342 3477 3749 3760 3954 3959 4403 4413 4558 4589 4607  
SEMICLSY 3414 3849 3860 4303 4341 4536 4564 4594 4661 4895 4920  
SEND\_MESSAGE 1450 1459 4540 4574  
SEND\_STATEM 3813 4504  
SENDER 760 1341 4539 4572 4612  
SENDING 58\* 82 1218\*1230  
SENDSY 3813 4511 4899 4924 5048  
SET\_OR\_REL 2032\*2033 2036\*2037  
SET\_PARM 1756 1797 2115 2159 2203 2247 2291 2334 2374 2418 4640 4644 4812  
SET\_PARMS 1684 1788  
SET\_SIZE 975 2495 2497\*2500 2506 2507 2586 2588\*2591 2597 2598 2678 2680\*2683 2689 2690 2770 2772\*2775 2781  
2782 2862 2864\*2867 2873 2874 2954 2956\*2959 2965 2966 3046 3048\*3051 3057 3058 3139 3141\*3144 3150  
3151  
SETPOS 67 70 72 74 76 233 317 321 342 622 723 1106 1108 1113 1115 1120 1122 1127 1130 1136  
1138 1143 1145 1150 1152 1157 1159 1186 1190 1192 1193 1199 1200 1203 1204 1207 1224 1227 1229 1241  
1242 1243 1244 1253 1429 1556 1558 1564 1567 1644 1645 1724 1731 1891 1895 1950 1959 1969 1972 1994  
2003 2044 2481 2482 2486 2488 2572 2573 2577 2579 2652 2653 2657 2659 2756 2757 2761 2763 2848 2849  
2853 2855 2940 2941 2945 2947 3032 3033 3037 3039 3125 3126 3130 3132 3215 3229 3322 3344 3371 3401  
3421 3430 3449 3452 3457 3463 3599 3600 3608 3866 3867 3868 3871 3872 3873 3895 4099 4100 4106 4107  
4114 4115 4116\*4126 4127 4128\*4133 4138 4140 4145 4146 4147 4148 4155 4156 4157\*4158 4159 4164 5086  
SIMULATION 25  
SIN 2055  
SINGLE\_STEP 60 1452 4038 4047 4048  
SNAPSHOT 991 4059 4063 4074 4078 5308 5317 5328 5364  
SNAPSHOT\_STA 3833 4067  
SNAPSHOTSY 3833 4072 4904 4930 5041  
SQRT 2072  
STACK\_EL 1185 1188 1189\*1206 1209\*  
STACKS\_STATE 3835 4248

STACKSSY 3835 4253 4906 4933 5069  
STACKTOP 3523 3585\*3631 3641 3646 3665 3673 3678 4472 4482 4913  
START 2512 2603 2696 2787 2880 2971 3064 3157  
START\_EXEC 5236 5238  
START\_NEXTSY 3236 3260 3263  
START\_STATEM 3836 4011  
STARTCLOCK 3913 4282 5415 5423  
STARTCPU 3916 4280 5416 5424  
STARTSY 3836 4016 4906 4934 5030  
STARTTIME 3910 5417 5425  
START2 5309 5377  
STATUS 58 679 752 802 3904 4088 4309 4322  
STATUS\_STATE 3845 4082  
STATUSSY 3845 4087 4906 4933 5070  
STOP\_STATEME 3842 3994  
STOPPED 81 635 749 991 4001 4006 4018 4025  
STOPSY 3842 3999 4047\*4906 4933 5031  
STORE\_MESSAG 700 743 911  
STRING 1928\*1929 1931\*1933 1936\*1938 1946\*1948 1950 1951 1955\*1958 1959 1960 1962 1966 1969 1970 1972 1975  
1990\*1992 1994\*  
STRING\_ARRAY 2270 5272  
STRING\_CONST 3447 4725 4727 4896 4921  
STRING\_ELEM 2812 2847 2875 2893 2896  
STRING\_POP 434 502 503 1710\*2280 2294 2819 2832 2836 2877 4869 5168  
STRING\_PUSH 427 507 1664 2289 2864 2884 2890 2891 4729 4828 4880 5190  
STRING\_SET 2807 2813 2876 5125 5300  
STRING\_TOP 289 431 432 436\* 502 503 983 1307 1709 2280 2812 2815 2828 2841 2875 4814 4845 4857 5167  
STRING\_VALUE 3458 3543 3604 3690 4729 4845 4880  
STRINGEL 429 430 983 1045 1046 1268  
STRINGSY 4496\*4893 4918 5024  
STRINGWERT 1365 1377 1664 1709 2284 2295 4814 4828  
SUB 505 506 1406 1411 1418 1423 1482 1938 2007 3212 3217 3226 3239 3506 3509 4195 4381 4384  
SUBTREE 3593\*3594  
SUC 83 361 785 819 936 1250 1349 1678 1823 1848 2475 2490 2498 2520 2566 2581 2589 2611 2661 2672  
2681 2704 2750 2765 2773 2795 2842 2857 2865 2888 2934 2949 2957 2979 3026 3041 3049 3072 3119 3134  
3142 3165 3952 4119 4131 4162 4365, 4411 4561 4592 4610  
SUCCESS 971  
SUCC4 5334  
SYMBNAME 3553\*3558 3567  
SYMBOL 3188 3267 3345 3346 3402 3403 3404 3464 3465\*3500  
SYMBTBL 3523 3555 3622 3658 3876 4912 4913  
SYNTAX\_ERROR 3491 3708 3719 3849 4303 4341 4536 4564 4594 4661  
SYSIN 3206 3211 3212 3215 3217 3220 3221 3225 3226 3229 3238 3892 3926 5094 5095\*  
SYSOUT 32 33 1417 1479 3239 3240 3905 3909  
SYSSTATUS\_ST 3846 4091  
SYSSTATUSSY 3846 4097 4905 4934 5071  
SYSTEM 636 655 731 854 869 878 922 938 1452 4236 4245  
SYSTEM\_STATE 3830 4239  
SYSTEMSY 3830 4244 4907 4935 5045  
S1 501 503 504 505\* 506  
S2 501 502 504 506\*  
S3 501 504 505 506 507  
T 618\* 621 1983 1986\*1987 2000 2001 2002 2003 2005 2007 2029\*2030 2040\*2043 4291 4304 4306\*4307 4338  
4878\*4880\*4881\*4882\*4883\*4884\*4885\*4886\*4887\*  
T\_MAX 4291 4296 4301 4302 4305 4327\*  
T\_OLD 4291 4306 4307 4338  
TAN 2061  
TARGET 682 689 690 694 696 697 698\* 699 713 717 718 729 739 740 741\* 742 945 946\*1342 4654  
4656 4660 4662 4663  
TBLPREV 3269 3547 3582 3583 3584 3585  
TBLREF 3269 3583 3584 3585\*3687 3748 3749 3949 3958 3960 4406 4407 4417 4418 4432\*4436\*4448\*4449 4453\*4454  
4473 4476 4523 4656 4792 4793  
TERMINALSY 4904 4931 5060  
TERMINATE\_ST 3827 3898  
TERMINATESY 3827 3903 4902 4928 5058  
TERMINATION 1450 1462 3497 5386  
TERM2 5382  
TEST 1622 2467 2558 2664 2742 2834 2926 3018 3111  
THERE\_IS 3708 3718 3732 3737 3813 3814 3815 3816 3817 3818 3819 3820 3821 3822 3823 3824 3825 3826 3827 3828  
3829 3830 3831 3832 3833 3834 3835 3836 3837 3838 3839 3840 3841 3842 3843 3844 3845 3846 3847 3848  
3849 3857 3860 3935 3937 3941 3972 3987 4003 4022 4047 4060 4075 4293 4298 4303 4341 4352 4404 4409  
4415 4433 4450 4464 4474 4478\*4493 4494 4495 4496 4497 4498 4514 4518 4527 4536 4554 4564 4594 4626  
4637 4641 4657 4661 4712 4713 4719 4725 4731 4737 4743 4753 4759 4765 4771 4777 4783  
TIME 611 614 615\* 666 875 877 902 905 906\* 927 929 1060 3918  
TIME\_LIMIT 4307 4332 4342 5082  
TIME\_STATEME 3847 4270  
TIMEOFDAY 3911 4204 4278 5412 5425  
TIMESLICE 661 1440 1491  
TIMESLICE\_RU 654 1451 1458  
TIMESY 3847 4276 4900 4934 5020  
TODAY 3912 4203 4278 5412  
TOK 3707\*3708 3732\*3733 3735  
TOKEN 3186 3268\*3358 3365 3370 3384 3403 3414 3418 3419 3422 3428 3436 3438 3440 3442 3444 3447 3495 3536  
3560\*3570 3571 3619 3620 3629\*3639\*3733

TOKENS\_PASSE 3266 3504 3699 3734\*3735 4379  
TOKENTEXT 3506 3508 3509 3692 4381 4383 4384 4937 4938 4939 4940 4941 4942 4943 4944 4945 4946 4947 4948 4949  
4950 4951 4952 4953 4954 4955 4956 4957 4958 4959 4960 4961 4962 4963 4964 4965 4966 4967 4968 4969  
4970 4971 4972 4973 4974 4975 4976 4977 4978 4980 4981 4982 4983 4984 4985 4986 4987 4988 4989 4990  
4991 4992 4993 4994 4995 4996 4997 4998 4999 5000 5001 5002 5003 5004 5005 5006 5007 5008 5009  
TOSY 4514 4516 4519 4899 4924 5050  
TOTAL\_AGENTS 804\*1442 3921  
TOTAL\_MSGS 703\*1442 3924  
TRACE 95 108 120 121 138 139 159 160 179 180 196 203 210 217 224 232 233 249 254 259  
266 273 280 287 294 301 308 337 362 991 1164 1591 1600 1611 1617 1777 1815 1839 1878 2126  
2170 2214 2258 2302 2341 2385 2429 2447 2460 2538 2551 2630 2643 2722 2735 2814 2827 2906 2919 2998  
3011 3091 3104 3971 3975 3986 3990  
TRACE\_AGENT 293  
TRACE\_ARRAY 248  
TRACE\_BOOL 279  
TRACE\_CONTIN 202  
TRACE\_CREATI 157 803  
TRACE\_ENUM 272  
TRACE\_ESCAPE 209  
TRACE\_FACETT 118  
TRACE\_FUNCT 223  
TRACE\_INT 265 5172  
TRACE\_NEXT\_M 314 362  
TRACE\_OPER 307  
TRACE\_OPER\_C 230 5365  
TRACE\_PROC 216  
TRACE\_REAL 258 5315  
TRACE\_RECORD 253  
TRACE\_SCRIPT 300  
TRACE\_SEND 136 680  
TRACE\_SIGNAL 195  
TRACE\_STATEM 3837 3979  
TRACE\_STRING 286 5168  
TRACE\_TERMIN 177 816  
TRACESY 3837 3984 4903 4929 5027  
TRANS\_TIME 944 1446 1507  
TRANSMIT 919 930 947  
TRUESY 4777 4779 4896 4921 5025  
TYP 1361\*1372 1373 1374 1375 1376 1377 1378 1379 1661 1663 1665 1667 1669 1671 1673 1675 1691 1693 1696  
1699 1702 1705 1708 1813 1837 4406 4417 4827 4829 4831 4833 4835 4837 4839 4867\*4869 4870 4871 4872  
4873 4874 4875  
TYPE 3538 3599 3601 3602 3603 3604 3605 3606 3607 3679 3746\*3748\*3749 4406 4417 4432 4436 4448 4453 4470  
4473 4476 4488 4660 4663 4666 4667\*4694\*4793 4845 4846 4847 4848 4850 4851 4852 4855\*4857 4858 4859  
4860 4861 4862 4863 4880 4881 4882 4883 4884 4885 4886 4887  
TYPE\_CHECK 3746 3943 3957 4520  
TYPE\_OF\_ARRA 1758 1796 1813 1837 2133 2177 2221 2265 2309 2348 2392 2436  
TYPE\_OF\_FACT 4711 4716 4722 4728 4734 4740 4748 4756 4762 4768 4774 4780 4786 4793 4794 4795 4797  
T1 3254\*3259 3477\*3479  
T2 3254\*3259 3477\*3479  
T3 3477\*3479  
U 897 918 935  
UBOUND 1722 1728 1739 1741 1743 1762 1763 1764 1765 1766 1772 1773 1774 1775 1776 1791 1792 1793 1794 1795  
1804 1807 1830 1831 1832 1833 1834 1857 1860 1871 1874 1886 1887 1888 1889 1890 2095 2096 2097 2098  
2099 2139 2140 2141 2142 2143 2183 2184 2185 2186 2187 2227 2228 2229 2230 2231 2271 2272 2273 2274  
2275 2315 2316 2317 2318 2319 2354 2355 2356 2357 2358 2398 2399 2400 2401 2402  
UP 1931\*1933 1936\*1937 1938  
UTILIZATION 610 614 615 901 905 906 4107 4146  
VARSY 3818 4464 4465 4892 4917 5017  
VAR105 5167 5177\*5180 5184 5190  
VAR106 5171 5181 5185 5191  
VAR107 5104 5131 5243  
VAR113 5198 5201 5314 5319  
VAR114 5105 5132 5246 5248  
VAR115 5106 5133 5247  
VAR116 5107 5134 5252 5254  
VAR117 5108 5135 5253  
VAR118 5109 5136 5258 5260  
VAR119 5110 5137 5259  
VAR120 5111 5138 5264 5266  
VAR121 5112 5139 5265  
VAR122 5113 5140 5270 5272  
VAR123 5114 5141 5271  
VAR124 5115 5142 5276 5278  
VAR125 5116 5143 5277  
VAR126 5117 5144 5282 5284  
VAR127 5118 5145 5283  
VAR128 5119 5146 5288 5290  
VAR129 5120 5147 5289  
VAR131 5121 5148 5292  
VAR133 5122 5149 5294  
VAR135 5123 5150 5296  
VAR137 5124 5151 5298  
VAR139 5125 5152 5300  
VAR141 5126 5153 5302

VAR143 5127 5154 5304  
VAR145 5128 5155 5306  
WAITING 978 4530 4611 4614 4615  
WAITING\_AGEN 978 1341 4524 4532\*4567 4570 4613\*4617 4619  
WAITINGSY 4904 4931 5062  
WARNING 3254 3315 3387 3397 3431 3454 3462  
WERT 149 151 171 261 268 275 282 289 296 303 310 465 466 473 474 480\* 486 487 494 495  
502 503 513 514 521 524 532 533 540 547 548 555 558 566 567 574 684 777 796 1030\*  
1033\*1036\*1039\*1042\*1045\*1048\*1051\*1232 1233 1234 1274 1281 1288 1295 1302 1309 1316 1323 1689 1692  
1694 1697 1700 1703 1706 1709 1739 1740 1796 1803 1856 1870 2104 2119 2148 2163 2192 2207 2236 2251  
2280 2295 2324 2363 2378 2407 2422 2446 2448 2461 2474\*2487 2497 2506 2508 2515 2522 2528\*2537 2539  
2552 2565\*2578 2588 2597 2599 2606 2613 2619\*2628 2631 2644 2658 2671\*2680 2689 2691 2699 2706 2712\*  
2721 2723 2736 2749\*2762 2772 2781 2783 2790 2797 2803\*2812 2815 2828 2841\*2854 2864 2873 2875 2883  
2890 2896\*2905 2907 2920 2933\*2946 2956 2965 2967 2974 2981 2987\*2996 2999 3012 3025\*3038 3048 3057  
3059 3067 3074 3080\*3089 3092 3105 3118\*3131 3141 3150 3152 3160 3167 3173\*4513\*4521 4529 4538\*4539  
4566 4567 4569 4570 4572 4573 4814 4815 4816 4818 4819 4820 4821 4822 4845 4846 4847 4849 4850 4851  
4852 4857 4858 4859 4860 4861 4862 4863 5167 5171 5314  
WRTRD 53 62 3207 3210 3219 3224  
X 1573\*1576 1581\*1584 1595\*1596 1597\*1598 1603\*1606 1615 1622\*1623 2480 2484 2485 2487 2490\*2571 2575  
2576 2578 2581\*2651 2655 2656 2658 2661\*2755 2759 2760 2762 2765\*2847 2851 2852 2854 2857\*2939 2943  
2944 2946 2949\*3031 3035 3036 3038 3041\*3124 3128 3129 3131 3134\*5175\*5177

TOTAL NUMBER OF DIFFERENTLY SPELLED IDENTIFIERS AVAILABLE IN THIS PROGRAM : 763



