DIETZ 6***

System-PASCAL

Erweiterung RTMS
Real-Time Management System
Benutzeranleitung

DIETZ Computer SYSTEME

DIETZ 6***

System-PASCAL
Erweiterung RTMS
Real-Time Management System
Benutzeranleitung

DIETZ Computer SYSTEME

DIETZ 6***

System-PASCAL

Erweiterung RTMS
Real-Time Management System
Aufgaben und Möglichkeiten
Konfigurierung

Dok.Nr. 2-8206-01-222
Schutzgebühr DM 10,00

DIETZ Computer SYSTEME

1. Aufgaben und Einsatzmöglichkeiten

1.1. Allgemeines

Da die Programmiersprache PASCAL in der Definition nach Professor N. Wirth (Eidgenössische Hochschule Zürich) alle Voraussetzungen erfüllt, um strukturierte und pflegbare Programmpakete zu implementieren, fiel die Entscheidung für ein neues Prozeßsystem zu Gunsten der Sprache PASCAL aus.

Bei der Auswahl des Grundkonzeptes für die Prozeßsynchronisation stehen folgende drei Methoden zur Verfügung, deren Funktionalität und Effizienz inzwischen allgemein anerkannt sind:

- das Hoare´sche Monitorkonzept;
- Synchronisation durch Exportprozeduren;
- Synchronisation durch Austausch von Nachrichten.

Da die beiden ersten Möglichkeiten eine Änderung der Sprache (Spracherweiterung) bedeutet hätten, wurde die Synchronisation durch Austausch von Nachrichten bevorzugt.

Programme können auf allen Modellen der Rechnerserie DIETZ 621 (DIETZ 612X0, DIETZ 621X2 und DIETZ 621X3) erstellt und ausgeführt werden.

Das Betriebssystem TSOS, das auf der Ebenenstruktur des Rechners basiert, ermöglicht die Steuerung und Überwachung von bis zu 12 pseudoparallelen Prozessen.

Diese 12 Prozesse werden von autonomen PASCAL-Programmen geführt und können sich gegenseitig informieren und synchronisieren.

Weiterhin ist das Erkennen externer Ereignisse (Events, Interrupts) und die Ein-/Ausgabe von digitalen und analogen Daten möglich.

1.2. Implementierte Funktionen

Die Funktionen des Realtime-Managementsystems (RTMS) unter PASCAL werden als Bibliotheksoperatoren ausgeliefert und dem PASCAL-Programm hinzugebunden.

Die Schnittstelle zwischen dem RT-Programm und den "Systemprozeduren" ist in einem Prolog definiert (Datei XOSPRO).

Folgende Funktionen sind enthalten:

- Datenaustausch über den Common Data Pool (CDP)
- Senden und Empfangen von Nachrichten
- Warten auf eine Nachricht
- Löschen einer Nachricht
- Formulierung eines Zeitauftrages
- Löschen eines Zeitauftrages
- Freigeben und Sperren eines Interrupteingangs
- Warten auf ein externes Ereignis
- Ändern der Prozeßpriorität
- Belegen bzw. Warten auf ein Semaphor
- Freigeben von Semaphoren
- Setzen und Abfragen der Systemzeit
- Ein-/Ausgabe von digitalen Daten
- Ein-/Ausgabe von analogen Daten

1.3. Voraussetzungen

1.3.1. Hardware

Folgende Hardwarekomponeten sind Minimalvoraussetzung für das Realtime-Managementsystem unter PASCAL:

Zentraleinheit	:	alle Modelle der Rechnerserie DIETZ 621 (621X0, 621X2, 621X3) mit mindestens 48 KByte Arbeitsspeicher
Benutzerterminal	:	alle vom Betriebssystem XOS in der Version TSOS ab Release 6 bzw. in der Version X3OS ab Release 7 unterstützten Modelle
Magnetplattenlaufwerk	:	alle vom Betriebssystem XOS in der Version TSOS ab Release 6 bzw. in der Version X3OS ab Release 7 unterstützten Modelle

Abweichend davon ist der Betrieb eines Rechners unter RTMS-PASCAL auch als Knoten eines DIXOS-Netzes ohne eigenen Plattenspeicher möglich.
Das System muß dann im Modus DLL (Down Line Loading) betrieben werden.

Einschränkung:

Für die Übersetzung von PASCAL-Programmen wird ein Magnetplattenlaufwerk mit einem für das Bediensystem COMSYS ausreichenden Volumen benötigt.

1.3.2. Software

Voraussetzung für das Übersetzen von PASCAL-Programmen ist ein zusammenliegender Speicherbereich (Partition) von mindestens 32 KByte (256 Sektoren).

Detaillierte Angaben für die Übersetzung von PASCAL-Programmen können der Benutzeranleitung "Bediensystem und Binder" entnommen werden.

Die folgenden Dateien und Dienstprogramme werden für die Ausführung bzw. Übersetzung von Programmen benötigt:

1.3.2.1. Ausführung von Programmen

TSOS-X
X3OS-X : Betriebssystem (residenter Anteil)

SERVIC
SERVX3 : Dienstprogramme und Teile des Betriebssystems, die in Überlagerungstechnik benutzt werden

PPSC.R : Scheduler; PASCAL-Programm für die Entgegennahme und Durchführung von Zeitaufträgen (Aktivierung durch 'SERV,START')

1.3.2.2. Übersetzung von Programmen

Für die Übersetzung von Programmen, die in PASCAL geschrieben sind und Funktionen des Realtime-Managementsystems (RTMS) benutzen, werden zusätzlich folgende Dateien benötigt:

MCOMSY : enthält das Bediensystem und den Binder in relozierbarem Kode (segmentiert)

COMSYS : enthält Start- und Überprüfungsroutinen

COMGEN : enthält Routinen, um allgemeine und übersetzerbezogene Schalter auf Anfangswerte zu setzen und erzeugt die Datei !INFOV

!INFOV : enthält Schalter und Informationen für die Übersetzer und den BINDER (benutzerbezogen gespeichert)

WF0u,WF1u WF2u : Arbeitsdateien für die Übersetzer und den Binder (WF = Work File, u = user/Benutzernummer)

XOSPAS : PASCAL-Übersetzer; enthält den Übersetzer in relozierbarem Kode (segmentiert)

XOSPRO : Prolog für PASCAL-Programme; enthält die Schnittstellendeklaration für die Systemprozeduren des RTMS

PASBIB : Bibliothek des PASCAL-Systems; enthält die Prozeduren des RTMS

Da einige Programme in relozierbarem Kode ausgeliefert werden und segmentiert sind, legt das System zur Ausführungszeit selbständig die Datei '!Cu' (u = user/Benutzernummer) an.
Die Größe der Datei '!Cu' resultiert aus der Größe des aufgerufenen Programms. Es erfolgt die Fehlermeldung "ERR: 69", wenn nicht genug Freiraum auf dem Externspeicher (Systemresidenz) vorhanden ist.
Mit Systemresidenz wird die Platteneinheit bezeichnet, von der das Betriebssystem/Sprachsystem geladen und aktiviert wurde.

Inhaltsverzeichnis der Systemresidenz

```
DIAL
*SERV,LIST,M,0,0,MCOMSY
    41   MCOMSY     16124      859   '37   '00

DIAL
*SERV,LIST,M,0,0,COM***
    35   COMSYS     13824        6   '30   '01
    40   COMGEN     16072       50   '37   '34

DIAL
*SERV,LIST,M,0,0,WF*0
     4   WF00        1200      500   '00   '34
     5   WF10        1700      500   '00   '34
    20   WF20        8320      500   '00   '34

DIAL
*SERV,LIST,M,0,0,XOSPAS
    42   XOSPAS     16984     1181   '36   '05

DIAL
*SERV,LIST,M,0,0,XOSPRO
    52   XOSPRO     18924        8   '19   '01

DIAL
*SERV,LIST,M,0,0,PASBIB
    90   PASBIB     25816     1347   '00   '34

DIAL
*SERV,LIST,M,0,0,SYSINI
    95   SYSINI     29364      264   '37   '34


*SERV,LIST,M,0,0,PPSC.R
    96   PPSC.R     29628       18   '37   '34

DIAL
*
```

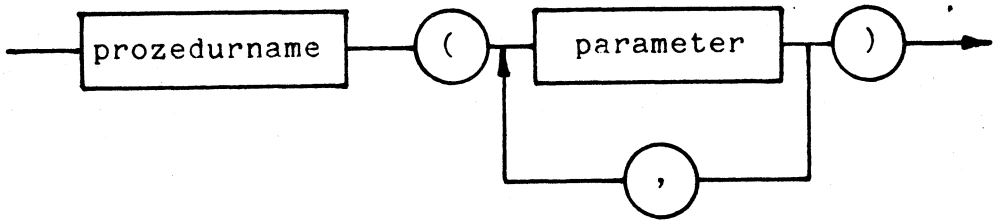
1.4. Alphabetische Liste der Aufrufe

Systemprozedur

1.5. Syntax der Aufrufe

1.5.1. Prozeduraufruf

Die Routinen des Realtime-Managementsystems (RTMS) werden über Systemprozedurnamen aufgerufen. Die Prozeduren sind im Prolog des PASCAL-Übersetzers deklariert.
Die sich durch die Deklaration ergebenden Schnittstellen sind im Kapitel 2 (Semantik der Aufrufe) erläutert.



Treten innerhalb der Routinen fehlerhafte Zustände auf oder sind übergebene Parameter nicht plausibel, so wird die Variable 'SYSTEMSTATE' mit einer Information für den aufrufenden Prozeß besetzt.

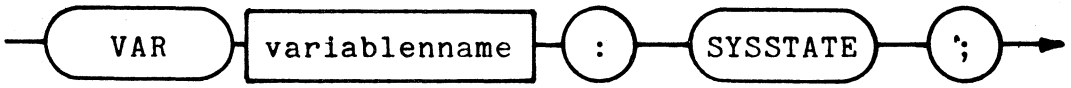
Diese Variable ist zu importieren und mit dem Typ SYSSTATE zu deklarieren (s. nächstes Kapitel).

1.5.2. Fehlervariable

Um fehlerhafte Zustände innerhalb des Prozeßsystems erkennen zu können, steht dem RT-Prozeß eine Informationsschnittstelle zur Verfügung.
Die Variable ist durch Aufruf in der IMPORTS-Liste zu importieren.

```
IMPORTS    SYSTEMSTATE
```

Die Informationsschnittstelle wird innerhalb des Programms durch Vereinbarung der Variablen SYSTEMSTATE mit dem Typ SYSSTATE erzeugt und kann bzw. muß nach jedem Aufruf überprüft werden (s.a. Kapitel 2.3, Fehlerbehandlung und Beispiel PBRTS1).



Beispiel:

```
PROGRAM PBXYZ (.....);

IMPORTS SYSTEMSTATE;
VAR SYSTEMSTATE : SYSSTATE;
BEGIN
.
.
.
END.
```

1.6. Typdeklarationen

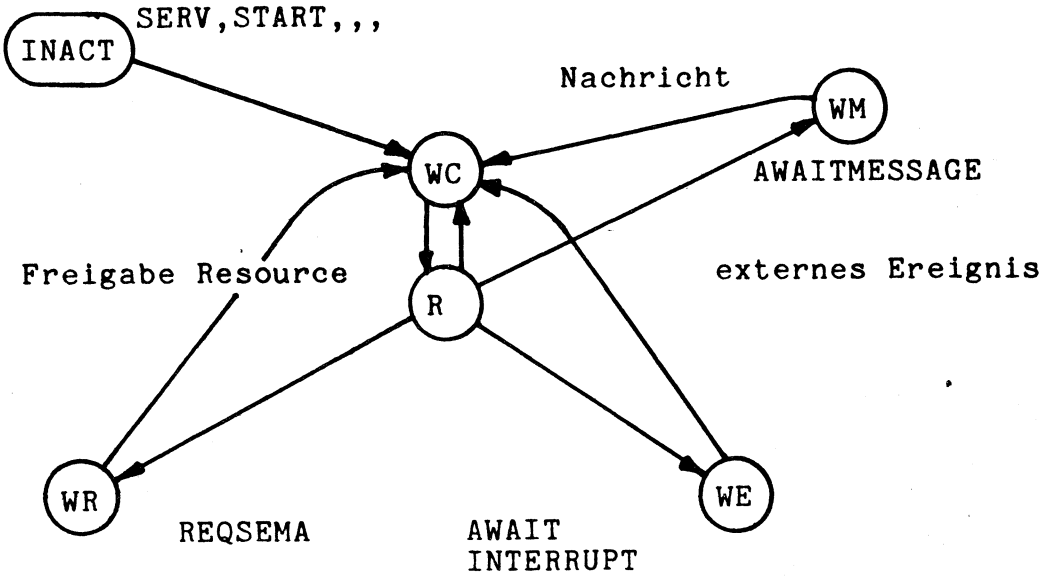
Folgende Typdeklarationen sind im Prolog des PASCAL-Übersetzers bereits enthalten und können zur Deklaration von Variablen benutzt werden.

```
INTEG   = -128..127;

INTEG3 = -8388608..8388607;

BIT     = 0..1;

BYTE    = 0..255;

BYTE2   = 0..65535;

BYTE3   = 0..16777215;

TIMETYPE     = RECORD
                  MSEC : BYTE3;
                  SEC, MIN, HOUR,
                  DAY, MONTH, YEAR : BYTE;
               END;

DAYTIMETYPE = RECORD
                MSEC : BYTE2;
                SEC, MIN, HOUR : BYTE;
              END;

STATETYPE   = (ALWAYS, WAITING);

MESSAGETYPE  =  ARRAY  0..15  OF BYTE;

SYSSTATE = (OK..POOLLENFAULT);  ! siehe Kapitel 2.11.2
```

Die Deklaration ´MESSAGETYPE´ durch den Prolog (ARRAY 0..15 OF BYTE) kann intern in einem RT-Programm durch die Möglichkeit der "Varianten Records" redefiniert werden. Redefinitionen für diesen Typ sind auch auf maximal 16 Byte begrenzt.

Anmerkung:
Formale Typdeklarationen im Prolog mit UNIV erlauben auf der Seite der RT-Programme das Einsetzen von beliebigen Datentypen. Die Auswirkung entspricht einer VAR-Deklaration. Der formelle Charakter der Typangabe wird durch die Angabe ANYTYPE aufgezeigt.

1.7. Zustandsdiagramm



INACT = Inaktiv

WM = Waiting Message (wartend auf eine Nachricht)

WE = Waiting Event (wartend auf ein Ereignis)

WR = Waiting Resource (wartend auf ein Betriebsmittel)

WC = Waiting CPU (wartend auf die Zentraleinheit; der Prozeß ist rechenwillig)

R = Running (Ausführung)

Die Zustände ´WC´ und ´R´ werden zusammengefaßt und als ´READY´ angezeigt. Der Zustand ´WM´ wird als SUS-pendiert angezeigt.
Die Aktivierung eines Prozesses wird durch das Dienstprogramm ´SERV,START´ vorgenommen.
Detaillierte Hinweise können dem Kapitel 1.8. (Start des Systems) entnommen werden.

1.8. Start des Systems

Nach Aufruf des Betriebssystems TSOS für die Zentraleinheiten der Serien DIETZ 621X0 und DIETZ 621X2 bzw. des Betriebssystems X30S für Zentraleinheiten der Serie DIETZ 621X3 können die RT-Prozesse und der Scheduler durch das Dienstprogramm 'SERV,START' aktiviert werden. Allerdings muß das Betriebsystem zuvor konfiguriert worden sein (siehe Kapitel 1.9).

Der Aufruf des Betriebssystems geschieht zunächst durch Drücken der Tasten 'RS' und 'BS'.
Es erscheint folgende Ausgabe des Rechners am Master-/Operatorplatz:

```
*** X0SBOOT - 07.02 ***

SYSTEMFILE  (UNIT, NAME)  _
```

Nach Eingabe der Nummer der Platteneinheit und des Dateinamens wird der residente Teil des Betriebssystems geladen bzw. ein Fehler gemeldet, wenn die Datei nicht vorhanden ist. Die Eingabe wird mit der Taste 'CR' oder 'RETURN' abgeschlossen. Nachdem das Betriebssystem geladen wurde, meldet sich der Kommandointerpreter mit der Ausgabe:

```
  DIAL
*  _
```

Nun können nacheinander die Anwenderprozesse und der Scheduler (optional) aktiviert werden.

Beispiel :

```
 DIAL
*SERV,START,0,RTME01,8,0,,R  <cr>

 DIAL
*SERV,START,0,PPSC.R,11,0,,R <cr>

 DIAL
*
```

1.8.1. Aktivieren der RT-Prozesse

Das Dienstprogramm ´START´ ist ab Release 6.02 (TSOS-X) und ab Release 7.01 (X3OS-X) Bestandteil der Datei SERVIC bzw. SERVX3.
Folgende drei Möglichkeiten sind durch dieses Dienstprogramm gegeben.

- Direkter Aufruf eines Programms ohne Dialoggerät:

  SERV,START,u,f,pnr,prio,adr,R

  f = Dateiname eines Maschinenkodeprogramms;
  Dateikode muß entsprechend gesetzt sein.

- Direkter Aufruf eines Programms mit Dialoggerät:

  SERV,START, u,f,pnr,prio,adr,D

  f = siehe oben

- Aufruf einer Kommandofolge für Prozeß (pnr):

  SERV,START,u,f,pnr,prio,adr,B

  f = Dateiname einer Kommandodatei;
  Dateikode muß ´1F sein

Parameterbeschreibung:

u = Platteneinheit (unit)
0,1,2... (default = 0)

f = Dateiname (filename)

pnr = Ebenen-Nummer 0,1,2...11

prio= Priorität 0..63 (default = 0)

adr = Startadresse des Programms
(default = Anfangsadresse der Partition)

1.9. Konfigurierung des Prozeßsystems

Aufgabe der Konfigurierung ist die Beschaffung und Vorbesetzung von Freispeicher im gemeinsamen, von allen Prozessen nutzbaren Speicherbereich (im folgenden COMMON genannt).

Der für die RT-Prozesse gemeinsame Datenbereich beginnt an der Adresse "CFSA" (Freispeicheranfang). Die Größe des benötigten Bereiches ist abhängig von der Konfiguration des Systems; diese wiederum von der Problemlösung.

Die Konfigurierung selbst wird im Dialog an der Masterkonsole durchgeführt. Die Dialogführung ist sicher; Eingaben, die zu einem inkonsistenten Systemzustand führen, werden abgelehnt. Gleichzeitig wird ein Hinweis auf die Art des Fehlers und seine Korrektur gegeben.

Das Konfigurierungsprogramm kann jederzeit durch Drücken der Taste 'BREAK' beendet werden, ohne daß das System in einen inkonsistenten Zustand überführt würde.

Für die Durchführung des Konfigurierungsprogramms wird ein zusammenliegender Speicherbereich von 16 KByte (128 Sektoren) benötigt.

Rechner der Serie DIETZ 621 (DIETZ 621XO, DIETZ 621X2), die mit einer Speichererweiterung im Adreßraum '2000 bis '3FFF ausgestattet sind, haben zwei COMMON-Bereiche.
Aktueller COMMON-Bereich ist derjenige, auf den zu Beginn der Konfigurierung die Systemvariable CFSA (Freispeicheranfang) zeigt.

Wird im Verlauf der Konfigurierung die Anlage eines Common Data Pools gewünscht, so kann dieser in den zweiten COMMON-Bereich gelegt werden, wenn der aktuelle Bereich nicht genügend Platz bietet.
Dabei ist die Adresse, ab der der zweite COMMON-Bereich genutzt werden soll, durch den Benutzer anzugeben.

Das Konfigurierungsprogramm kann dabei die Korrektheit der Angabe nur bedingt prüfen. Insbesondere kann nicht geprüft werden, ob im angegebenen Bereich nicht schon Betriebssystem-erweiterungen (Moduln etc.) abgelegt sind.
Wenn ein Common Data Pool angelegt wurde, wird dieser vom System mit ´00 initialisiert.

Wird die Konfigurierung normal beendet, so enthält die Systemvariable "CFSA" die Adresse des ersten freien Byte hinter dem gemeinsamen Datenbereich.

Bei den Zentraleinheiten DIETZ 621X0 und DIETZ 621X2 muß nach Beendigung des Konfigurierungsprogramms die nun modifizierte Version des Betriebssystems mit dem Kommando ´NEW´ auf einer Platteneinheit gesichert werden.

WICHTIG

Eine Konfigurierung darf nur auf der Basis eines unveränderten Betriebssystems erfolgen, d.h. direkt nach dem "Urladen".

Im einzelnen sind folgende Arbeitsgrößen anzugeben:

a) Anzahl der RT-Prozesse

b) Nummer der niedrigsten Ebene, von der ab aufwärts die RT-Prozesse ausgeführt werden (npnr)
Die im Kapitel 2.2.2 (Semantik) benutzte Größe hpnr ergibt sich aus den Angaben a) und b);
hpnr = npnr + Anzahl der RT-Prozesse.

c) Größe der Nachrichtenpuffer je Prozeß (Anzahl Einträge)

d) Größe der Auftragstabelle des Schedulers (Anzahl Aufträge).

e) Ebene des Schedulers (Prozeßnummer)

f) Anzahl der privilegierten Prozesse (Interruptverarbeitung)

g) Ebenen (Prozeßnummern) der privilegierten Prozesse

h) Umfang der Prozeßperipherie für Ein-/Ausgaben an den Anwendungsprozeß

i) Montageadressen der Ein-/Ausgabeschnittstellen für die Prozeßperipherie

j) Größe des Common Data Pools (CDP)

Die Größe des benötigten Bereiches im COMMON-Bereich richtet sich nach diesen Angaben und wird programmintern laufend überprüft; ist der Freiraum im COMMON zu klein, so wird die Konfigurierung abgebrochen.
Ein Beispiel für den Größenbedarf kann der folgenden Seite entnommen werden.
Das System kalkuliert die Größe des benötigten Bereichs automatisch.

Folgender Größenbedarf ist z.B. möglich:

Bedarf im COMMON in Byte =

70Byte + (A+1) * (N+19Byte) + 9*P + NP*N + PERIPH + CDP

Variablenbeschreibung:

A : Kapazität der Auftragstabelle des Schedulers (Einheit: Anzahl Aufträge)

P : Anzahl der RT-Prozesse

N : Größe einer Nachricht (16 Byte)

NP : Größe des Nachrichtenpuffers (Einheit: Anzahl Einträge)

PERIPH : Größe des Bereichs für die Prozeßperipherie (Einheit Byte)

CDP : Größe des Bereichs für die Anlage des Common Data Pools;
0, wenn kein CDP angelegt wird -
P * 3 + angegebene Größe des CDP, wenn ein CDP angelegt wird

Der Platzbedarf für die Prozeßperipherie (s. oben PERIPH) ergibt sich aus folgender Formel:

PERIPH := y + z

y = Anz. der Digitalausgänge/8
z = Anz. der Interrupteingänge/8

Die Dimension der Größe PERIPH wird in Byte angegeben.

DIETZ 6***

System-PASCAL
Erweiterung RTMS
Real-Time Management System
Semantik der Aufrufe

Dok.Nr. 2-8206-01-223
Schutzgebühr DM 22,50

DIETZ Computer SYSTEME

2. Semantik der Aufrufe

Die wesentliche Aufgabe des RTMS ist das zur Verfügung stellen von Synchronisationsmöglichkeiten für die RT-Prozesse. Die Synchronisation geschieht durch die Übermittlung von Nachrichten.
Dabei bilden die im folgenden beschriebenen Aufrufe ´SEND´, ´RECEIVE´ und ´AWAITMESSAGE´ die grundlegenden Synchronisationsmechanismen.
Darüberhinaus kann das Übermitteln von Nachrichten auch zeitbezogen vorgenommen werden (s. dazu Kapitel 2.3.

2.1. Synchronisationsaufträge

2.1.1. Senden einer Nachricht

Eine Nachricht kann zur Information anderer "aktiver" Prozesse über Vorkommnisse beliebiger Art eingesetzt, oder aber zur Aktivierung einer oder mehrerer "wartender" Prozesse benutzt werden.
Eine Nachricht kann zielgerichtet an einen bestimmten Empfänger abgesetzt oder aber auch allen vorhandenen Prozessen zugestellt werden.

Aufruf:

```
SEND (ep-nr, n, ep-st)
```

Erläuterung der Parameter:

- ep-nr : Nummer des Empfängerprozesses;
  Variable oder Konstante

  Typ : INTEG

  Wertebereich : ep-nr = -1 oder npnr <= ep-nr <= hpnr

  Ist der Wert des Parameters ep-nr beim Aufruf -1, so wird die Nachricht 'n' allen Prozessen im Nachrichtenpuffer zur Verfügung gestellt.
  Die Übertragung wird allerdings nur in Abhängigkeit vom dritten Parameter 'ep-st' vorgenommen.

  npnr = niedrigste Prozeßnummer
  hpnr = höchste Prozeßnummer

- n : Nachricht;
  Variable oder Konstante

  Typ : MESSAGETYPE;
  redefinierbar; Länge maximal
  16 Byte

Der Wert des Parameters ´n´ wird im Nachrichtenpuffer des Empfängerprozesses (bzw. der Empfängerprozesse) in Abhängigkeit vom Parameter ´ep-st´ abgelegt.

- ep-st : Empfängerstatus;
  Variable oder Konstante

  Typ : STATETYPE

In Abhängigkeit von dem Parameter ep-st geschieht die Weiterleitung der Nachricht an den oder die Empfängerprozesse.
Ist der Wert des Parameters ´ep-st´ WAITING, so wird die Nachricht nur dann in den Nachrichtenpuffer des Empfängerprozesses eingetragen, wenn dieser sich im Zustand "Wartend auf eine Nachricht" befindet.
Ist der Wert des Parameters ´ep-st´ ALWAYS, so wird -falls der Nachrichtenpuffer frei ist- die Nachricht eingetragen.

2.1.1.1. Beispiel

Durch Bewerten des Parameters ´ep-nr´ (Empfängerprozeß) mit -1 und des Parameters ´ep-st´ (Empfängerstatus) mit WAITING wird erreicht, daß die Nachricht ´n´ an alle Prozesse weitergegeben wird, die sich im Zustand "Wartend auf eine Nachricht" befinden.

SYSTEMSTATE

OK, RECUNKNOWN, RENOTDRY, RECBUFOFL, NOUSER

siehe dazu Kapitel 2.11.2.

2.1.2. Empfangen einer Nachricht

Falls eine Nachricht im eigenen Nachrichtenpuffer eingetragen ist, wird diese bei Aufruf der Prozedur ´RECEIVE´ an den Prozeß übergeben und der Nachrichtenpuffer danach gegebenenfalls als leer gekennzeichnet.
Sind mehrere Nachrichten eingetragen, so erhält der Prozeß die älteste übermittelt.
Ist der Nachrichtenpuffer zum Zeitpunkt des Aufrufes leer, so wird der Wert der Variablen ´SYSTEMSTATE´ auf ´BUFFERMT´ gesetzt.

Aufruf:

```
RECEIVE (nv)
```

Erläuterung des Parameters:

- nv : Nachrichtenvariable;
  Variable

  Typ : MESSAGETYPE;
  redefinierbar; Länge maximal 16 Byte

  Die Nachrichtenvariable ´nv´ dient zur Aufnahme von übermittelten Daten.

SYSTEMSTATE

OK, BUFFERMT, NOUSER, BUFFEROFL, TIMECONDITION

siehe dazu Kapitel 2.11.2.

2.1.3. Warten auf eine Nachricht

Falls eine Nachricht im eigenen Nachrichtenpuffer eingetragen ist, wird diese bei Aufruf der Prozedur an den RT-Prozeß übergeben und der Status des Nachrichtenpuffers danach gegebenenfalls als leer gekennzeichnet. Der RT-Prozeß wird nicht suspendiert.

Ist der Nachrichtenpuffer zum Zeitpunkt des Aufrufes leer, so wird der aufrufende RT-Prozeß suspendiert, d.h. in den Zustand "Wartend auf eine Nachricht" versetzt.
Im Gegensatz zur Prozedur ´RECEIVE´ wird die Meldung BUFFERMT nicht erzeugt.

Aufruf:

```
AWAITMESSAGE (nv)
```

Erläuterung des Parameters:

- nv : Nachrichtenvariable; Variable

  Typ : MESSAGETYPE; redefinierbar; Länge maximal 16 Byte

  Die Nachrichtenvariable ´nv´ dient zur Aufnahme von übermittelten Daten.

SYSTEMSTATE

OK, NOUSER, BUFFEROFL, TIMECONDITION

siehe dazu Kapitel 2.11.2.

2.1.4. Löschen einer Nachricht

Mit diesem Aufruf können die Nachrichten im eigenen Nachrichtenpuffer, im Puffer eines anderen RT-Prozesses oder in allen Puffern gelöscht werden.

Aufruf:

```
CLEARMESSAGE (ep-nr)
```

Erläuterung des Parameters:

- ep-nr : Nummer des Empfängerprozesses

  Typ : INTEG

  Bei Angabe von -1 als Parameter werden die Nachrichten aller Puffer gelöscht.

SYSTEMSTATE

OK, NOUSER, RECUNKNOWN

siehe dazu Kapitel 2.11.2.

2.2. Übertragung von Daten über den Common Data Pool

2.2.1. Allgemeines

Der Austausch größerer Datenmengen zwischen den Prozessen kann über den gemeinsamen Plattenspeicher erfolgen oder über einen Teil des Hauptspeichers, zu dem alle Prozesse zugreifen können. Dieser sogenannte COMMON-Bereich wird bei der Konfigurierung in Auftragstabellen und in den Common Data Pool (CDP) eingeteilt.
Die Größe des CDP wird bei der Konfigurierung festgelegt; die Struktur kann innerhalb eines Programms durch eine Deklaration eines Types ´commondatapooltpy´ festgelegt werden, wobei dies ein vom Anwender frei wählbarer Typ (u.Typname) ist, der die gewünschte Struktur dec CDP definiert. Die maximale Größe dec CDP darf dabei nicht überschritten werden. Der Zugriff erfolgt über die Prozeduren ´GETPOOL´ und ´PUTPOOL´.

Dabei ist der Zugriff auf beliebige Teilstrukturen des CDP möglich. Ein direkter Zugriff auf den CDP über die CDP-Zugriffsvariable unter Umgehung der Prozeduren ´GETPOOL´ bzw. ´PUTPOOL´ ist nicht möglich.

Für die Strukturkonsistenz des CDP innerhalb eines Prozeßsystems sind die RT-Programme verantwortlich. Die Synchronisation der Zugriffe kann durch Nachrichtenübermittlung geschehen (s. auch Kapitel 2.1.).

2.2.2. Initialisieren des CDP

Die Prozedur INITPOOL initialisiert die prozeßspezifischen CDP-Kontroll- variablen und die CDP-Zugriffsvariable des aufrufenden RT-Prozesses. Sie muß deshalb von jedem RT-Prozeß einmal vor seinem ersten Zugriff auf den CDP aufgerufen werden.

Aufruf:

```
INITPOOL ( pcdp )
```

Erläuterung des Parameters:

- pcdp: CDP-Zugriffsvariable;
  Pointervariable, (common data pool)

  Typ : UNIV/ANYTYPE (s.a. Kapitel 1.6.)

Die Pointervariable ´pcdp´ wird als Parameter für die CDP Zugriffsprozeduren ´getpool´ und ´putpool´, sowie für die Prozedur ´poolcheck´ benötigt.
Die Pointervariable ´pcdp´ ist nach fehlerfreier Durchführung der Prozedur INITPOOL initialisiert und kann nun als Parameter in den Prozeduren ´GETPOOL´ und ´PUTPOOL´ verwandt werden.

SYSTEMSTATE

OK, NOUSER

siehe dazu Kapitel 2.11.2.

2.2.3. Überprüfen des CDP

Mit Hilfe der Prozedur POOLCHECK werden zusätzlich zu der Prozedur INITPOOL weitere prozeßbezogene und globale CDP-Kontrollvariable gesetzt. Diese zusätzlichen Kontrollvariablen dienen dazu, die Konsistenz des CDP innerhalb des Prozeßsystems zu überwachen.
POOLCHECK überprüft, ob die Länge des Typs ´commondatapooltyp´ nicht größer als die maximale Länge des CDP ist. Danach wird die Größe des Typs einmal prozeßbezogen (für den aufrufenden Prozeß) und einmal global (für alle Prozesse) abgelegt.
Dadurch ist es möglich, daß die Prozeduren ´GETPOOL´ und ´PUTPOOL´ überprüfen können, ob alle auf den CDP zugreifenden Prozesse die gleiche Definition des ´commondatapooltyps´, zumindest was seine Größe betrifft, verwenden.

POOLCHECK muß nach der Prozedur INITPOOL, aber vor dem ersten Zugriff zum CDP aufgerufen werden.
Außerdem muß die Prozedur POOLCHECK entweder von allen RT-Prozessen aufgerufen werden oder aber keiner darf sie aufrufen.

Wird POOLCHECK nicht aufgerufen, so ist keine Typprüfung möglich; das RT-System meldet einen Ausführungsfehler, wenn außerhalb des CDP zugegriffen wird (Zugriffsvariable größer CDP).

Aufruf:

```
POOLCHECK ( vcdp )
```

Erläuterung der Parameter:

- vcdp : Übertragungsbereich;
  dereferenzierte CDP-Zugriffsvariable ´pcdp^´

  Typ : UNIV/ANYTYPE (s.a. Kapitel 1.6.)

Der Parameter gibt die dereferenzierte CDP-Zugriffsvariable 'vcps' (siehe vorheriges Kapitel) an. Es dürfen keine Selektionen der Variable angegeben werden.

Die Prozedur setzt die prozeßbezogenen CDP-Kontrollvariablen 'initlen' und 'actlen'. Die beiden Kontrollvariablen werden auf die Länge des Typs des 'commondatapooltyps' cdp gesetzt. Die Variable 'actlen' gibt die für alle Prozesse gültige Länge wieder.

SYSTEMSTATE

OK, NOUSER, NOPOOLINIT, POOLOVERFLOW

siehe dazu Kapitel 2.11.2.

2.2.4. Übernahme von Daten aus dem CDP

Die Prozedur GETPOOL überträgt Daten aus dem durch den ersten Parameter (cdps) spezifizierten Teil des CDP in die Variable, die durch den zweiten Parameter spezifiziert ist.

Aufruf:

```
GETPOOL (cdps, ziel)
```

Erläuterung der Parameter:

- cdps : Teilstruktur des Common Data Pools;

  Typ : UNIV/ANYTYPE (s.a. Kapitel 1.6.)
  dereferenzierte CDP-Zugriffsvariable

- ziel : Datenbereich des RT-Prozesses;

  Typ : UNIV/ANYTYPE (s.a. Kapitel 1.6.)
  Variable vom gleichen Typ, wie cdps
  für die Aufnahme der Daten des CDP

SYSTEMSTATE

OK, NOUSER, DATALENFAULT, NOPOOLINIT, POOLLENFAULT, POOLOVERFLOW

siehe dazu Kapitel 2.11.2.

2.2.5. Übergabe von Daten an den CDP

Die Prozedur PUTPOOL überträgt Daten aus der Variablen 'quelle' des RT-Programms an den durch 'cdps' spezifizierten Teilbereich des CDP.

Aufruf:

```
PUTPOOL (cdps, quelle)
```

Erläuterung der Parameter:

- cdps : Teilstruktur des Common Data Pools;

  Typ : UNIV/ANYTYPE (s.a. Kapitel 1.6.)
  dereferenzierte CDP-Zugriffsvariable

- quel : Datenbereich des RT-Prozesses;

  Typ : UNIV/ANYTYPE (s.a. Kapitel 1.6.)
  Variable vom gleichen Typ, wie cdps, deren Inhalt an den CDP übertragen werden soll

SYSTEMSTATE

OK, NOUSER, DATALENFAULT, NOPOOLINIT, POOLLENFAULT, POOLOVERFLOW

siehe dazu Kapitel 2.11.2.

2.3. Zeitaufträge

2.3.1. Allgemeines

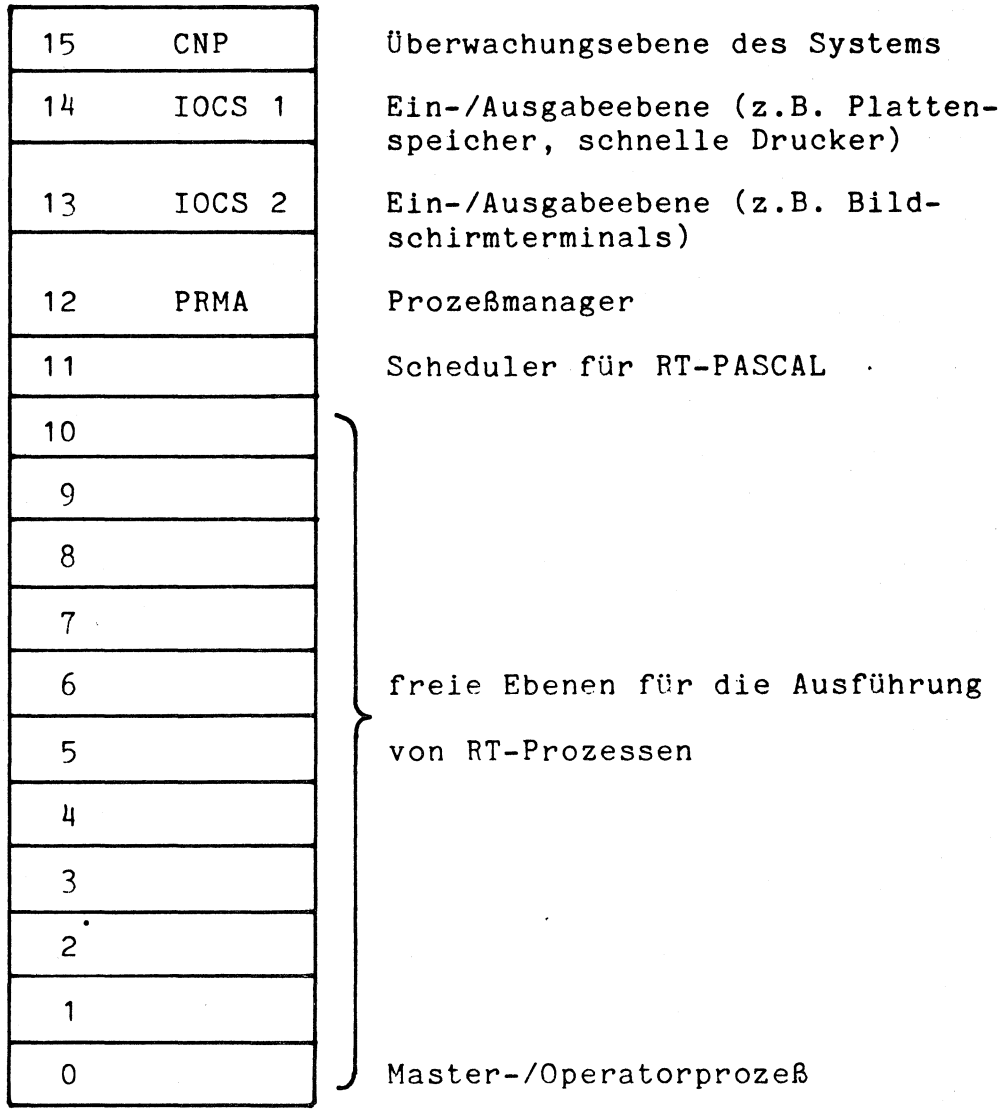
Zeitaufträge werden durch den sogenannten "Scheduler" verwaltet. Der Scheduler ist ein zum Lieferumfang von RT-PASCAL gehörendes Systemprogramm und wird auf einer eigenen Rechnerebene durchgeführt.
Resultat eines Auftrages an den Scheduler ist die Übermittlung einer Nachricht zu einem bestimmten Zeitpunkt an einen oder an alle RT-Prozesse. Die Verwendung des Schedulers ist optional und bedeutet, daß noch 11 quasiparallele Prozesse je Zentraleinheit ausgeführt werden können.
Der Scheduler benötigt neben einer Ebene des Rechners einen zusammenliegenden Speicherbereich (Partition) mit 4 Kbyte.

Der Scheduler wird durch Aufruf des Dienstprogrammes 'SERV,START' aktiviert. Detaillierte Angaben können dem Kapitel 1.8. (Start des Systems) entnommen werden.

Die Verwaltung der Auftragstabelle wird im Takt von einer Millisekunde durchgeführt. Dieses ist in der Regel auch die Genauigkeit, mit der Zeitaufträge gegeben und erfüllt werden können. Es kann jedoch zu Verschiebungen kommen, wenn mehrere Aufträge zum selben Zeitpunkt fällig werden. Die Verschiebungen betragen im Mittel 500 Mikrosekunden (theoretischer Wert) für jeden gleichzeitig durchzuführenden Auftrag.
Eine weitere Ursache für eine verspätete Ausführung eines Zeitauftrages kann die übermäßig hohe Belastung der Ein-/Ausgabeebenen des Rechners sein. Damit der Scheduler nicht noch zusätzlich durch andere Prozesse verzögert wird, empfiehlt sich die Benutzung der Ebene 11 des Rechners für den Scheduler.

Diese Empfehlung gilt natürlich nur dann, wenn das Erkennen externer Ereignisse (Interrupts, Events) nicht für noch wichtiger erachtet wird.

Ebenenaufteilung des Rechners DIETZ 621 unter dem Betriebssystem TSOS bzw. X30S mit Scheduler:



Eine Abweichung von der auf der vorigen Seite dargestellten Aufteilung der Ebenen kann dann sinnvoll sein, wenn zeitkritische externe Ereignisse auftreten können und darauf reagiert werden muß.
Die Unterbrechungssignale (Interrupts) dieser externen Ereignisse können dann auf die Ebenen 11, 10 und/oder 9 gelegt werden. Dementsprechend muß dann der Scheduler auf einer niedrigeren Ebene gestartet werden.

Die Hardwareprioritäten sind allerdings nur für den Zeitpunkt des Eintreffens externer Ereignisse (Interrupts) relevant.
Es erfolgt sofort nach dem Erkennen eines Interrupts eine Übergabe der Steuerung an den Prozeßmanager (siehe vorhergehende Seite), sodaß dann die Softwareprioritäten von Bedeutung sind.

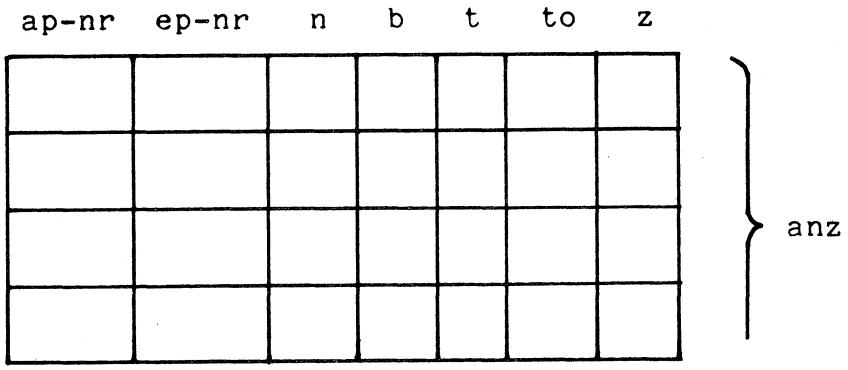
Die Prioritäten der einzelnen Prozesse können beim Binden der Programme durch das Kommando PRI festgelegt werden.
Darüber hinaus ist es natürlich möglich, daß die Prozesse ihre Priorität selbst festlegen (s.a. Kapitel 2.5., SETPRIORITY).

Standardmäßig haben alle Prozesse die Priorität 32.
Der RT-Scheduler PPSC.R hat die Priorität 48.

2.3.1.1. Durchführung eines Zeitauftrages

In einem Zeitraster von einer Millisekunde wird die Auftragstabelle des Schedulers auf zeitmäßig fällig gewordene Aufträge überprüft.
Anhand der, in der Auftragstabelle zusätzlich gespeicherten Auftragsdaten wird ermittelt, ob ein zeitlich fällig gewordener Auftrag ausgeführt werden muß. Falls ja, wird die gespeicherte Nachricht an den oder an die Anwenderprozesse übermittelt.
Gleichzeitig wird der Auftrag aus der Auftragstabelle entfernt bzw. aktualisiert (Wiederholungszähler dekrementieren.)

Bei jedem Auftrag werden folgende Daten in der internen Auf- tragstabelle gespeichert werden:



Die Größe der Auftragstabelle (anz = Anzahl der Einträge) wird bei der Konfigurierung festgelegt.
Die Bedeutung der einzelnen Spalten kann der nächsten Seite entnommen werden.

Bedeutung der eingetragenen Daten

ep-nr : Prozeßnummer des Empfängers

n : zu übermittelnde Nachricht

b : Bedingung für die Übermittlung der Nachricht; die Bedingung betrifft den Status des Empfängerprozesses; (s.a. Kap. 2.3.3., Formulierung eines Zeitauftrages)

t : time;
Zeitpunkt der ersten Ausführung

to : time offset;
relevant, wenn mehrmaliges Übermitteln beauftragt wurde oder bei einem Relativauftrag

z : Wiederholungszähler;
nur relevant bei mehrmaligem Übermitteln

2.3.2. Terminierung des Schedulers

Die Ausführungen des Schedulers, d.h. die Übermittlung fällig gewordener Zeitaufträge, können zu beliebigen Zeitpunkten unterbunden werden.

Durch Aufruf der Prozedur 'SHUTDOWN' wird ein Schalter innerhalb des Schedulers gesetzt. Die tatsächliche Terminierung erfolgt bei der nächsten Aktivphase des Schedulers.
Die Auftragstabelle kann trotz Terminierung weiterhin manipuliert werden, d.h. es können Aufträge eingefügt oder auch vorhandene Aufträge gelöscht werden (s.a. Aufrufe SCHEDULE und CANCELSCHEDULE Kap. 2.3.3. und 2.3.4.).

Aufruf:

```
SHUTDOWN
```

Der Scheduler kann jederzeit mit Hilfe des Dienstprogramms 'SERV,START' (s.a. Kapitel 1.8., Start des Systems) wieder aktiviert werden.

SYSTEMSTATE:

wird nicht beeinflußt

2.3.3. Formulierung von Zeitaufträgen

Die maximale Anzahl der Zeitaufträge wird bei der Konfigurierung des Systems festgelegt. Für die Speicherung der Aufträge wird eine Tabelle im COMMON-Bereich des Hauptspeichers genutzt.
Der COMMON-Bereich ist derjenige Teil des Hauptspeichers, der von allen Prozessen gemeinsam genutzt werden kann. Auch ohne Aktivierung des Schedulers können Zeitaufträge formuliert, d.h. eingetragen bzw. gelöscht (ausgetragen) werden. Die Ausführung eines Auftrages geschieht allerdings nur dann, wenn der Scheduler durch das Dienstprogramm 'SERV,START' aktiviert wurde.

Aufruf:

```
SCHEDULE (ep-nr, n, b, abs, t, to, anz)
```

Erläuterung der Parameter:

- ep-nr : Nummer des Empfängerprozesses

  Typ : BYTE;
  Variable oder Konstante

  Wertebereich : np-nr <= ep-nr <= hp-nr

- n : Nachricht;
  Variable oder Konstante

  Typ : MESSAGETYPE;
  redefinierbar; Länge maximal
  16 Byte

- b : Bedingung

  Typ : STATETYPE;
  Variable oder Konstante

  Bedeutung : WAITING

  Die Nachricht wird in den Nachrichtenpuffer des Empfängerprozesses geschrieben, wenn sich dieser im Zustand "Wartend auf eine Nachricht" befindet.

- abs : Zeitauftrag in Absolutzeit

  Typ : BOOLEAN;
  Variable oder Konstante

  Bedeutung : TRUE

  Hat der Parameter ´abs´ den Wert TRUE, so bestimmt der Parameter ´t´ (time) das erstmalige Fälligwerden des Auftrages. Der Parameter ´to´ (time offset) ist nur dann relevant, wenn der Auftrag mehrfach durchgeführt werden soll.

  Bedeutung : FALSE

  Hat der Parameter ´abs´ den Wert FALSE, so handelt es sich um einen Relativauftrag. Der Wert des Parameters ´to´ (time offset) beschreibt die zeitliche Differenz zwischen dem Aufruf (Eintrag in die Auftragstabelle) und dem ersten Fälligwerden des Auftrags. In diesem Fall ist der Parameter ´t´ (time) ohne Bedeutung.

- t : time (Zeitpunkt der ersten Ausführung)

  Typ : RECORD
  MSEC : 0..65535;
  SEC, DAYTIMETYPE
  MIN,
  HOUR : 0..255
  END

  Wertebereich : 0 <= MSEC <= 1000
  0 <= SEC <= 59
  0 <= MIN <= 59
  0 <= HOUR <= 23

  Der Parameter 't' bestimmt das erstmalige Fälligwerden eines Auftrages (der Wert des Parameters 'abs' ist TRUE).

- to : time offset (Zeitintervall)

  Typ : INTEGER;
  Variable oder Konstante

  Wertebereich : 0 <= to <= 86.400.000

  Die Angabe erfolgt in Millisekunden; es können maximal 24 Stunden (86.400.000 Millisekunden) angegeben werden.
  Der Parameter 'to' bestimmt die zeitliche Differenz bei mehrfacher Ausführung eines Auftrages, wenn der Parameter 'abs' mit TRUE bewertet wurde. Ist der Parameter 'abs' mit FALSE bewertet worden, so handelt es sich um einen Relativauftrag.
  Der Parameter 'to' bestimmt die zeitliche Differenz zwischen Formulierung des Auftrags und Ausführung.

Die Angaben von to = 0 und to >= 86.400.000 haben die gleiche Auswirkung. Der Auftrag wird -wenn es ein Relativauftrag war- 24 Stunden ausgeführt. Die Angabe von to = 0 bei einem Absolutzeitauftrag mit mehrfacher Ausführung ist unsinnig.

- anz : Anzahl

  Typ : INTEGER;
  Variable oder Konstante

Durch den Wert dieses Parameters wird die Anzahl der Übertragungen einer Nachricht an den Empfängerprozeß bestimmt.

Der zeitliche Abstand zwischen den Übertragungen wird durch to (time offset) bestimmt. Wird als Wert eine 0 übergeben, so wird die Nachricht so lange wiederholt, bis der Auftrag explizit gelöscht wird (s.a. Kap. 2.3.4., CANCELSCHEDULE).

Die Angabe des Wertes < 0 bei Anzahl ist nicht definiert und damit nicht erlaubt.

SYSTEMSTATE

OK, NOUSER, RECUNKNOWN, TASKLISTOFL

siehe dazu Kapitel 2.11.2.

2.3.4. Löschen von Zeitaufträgen

Wenn ein Zeitauftrag vollständig ausgeführt wurde, wird er aus der Auftragstabelle entfernt.
Das kann nach der ersten Ausführung der Fall sein, oder aber nach n-Ausführungen, wenn mehrfaches Ausführen beauftragt wurde.

Explizites Löschen von Zeitaufträgen ist immer dann notwendig, wenn die Anzahl der Auftragsdurchführungen nicht begrenzt wurde (Parameter anz = 0 bei SCHEDULE; s.a. Kap. 2.2.3.) oder aber sich die Ausführung des Auftrages erübrigt hat.

Aufruf:

```
CANCELSCHEDULE (ap-nr, ep-nr)
```

Erläuterung der Parameter:

- ap-nr : Nummer des Absenderprozesses

  Typ : INTEG;
  Variable oder Konstante

  Wertebereich : -1 oder 0 <= ap-nr <= 11

  Wird als Wert -1 angegeben, so wird der Zeitauftrag ohne Rücksicht auf den Absender ausgetragen (gelöscht);
  wird dagegen aber die Prozeßnummer des Absenders angegeben, so wird der Auftrag nur dann gelöscht, wenn er auch von diesem Prozeß eingetragen wurde.

- ep-nr : Nummer des Empfängerprozesses

  Typ : BYTE;
  Variable oder Konstante

  Wertebereich : 0 <= ep-nr <= 11

Es wird ein Parameterfehler gemeldet, wenn der Wert des Parameters (ap-nr) im Bereich 0 bis 11 angegeben wurde, diese Angabe aber nicht mit der Prozeßnummer übereinstimmt, die in der Auftragstabelle festgehalten wurde.

Die Identifizierung der zu löschenden Zeitaufträge erfolgt durch Angabe der Prozeßnummer ap-nr. Es werden alle Zeitaufträge des Absenderprozesses (ap-nr) bezogen auf den Empfängerprozeß (ep-nr) gelöscht.

SYSTEMSTATE

OK, PARAMERROR, NOTASK, NOTFOUND, NOUSER, RECUNKNOWN

siehe dazu Kapitel 2.11.2.

2.4. Behandlung von externen Ereignissen

Externe Ereignisse (events) werden über Unterbrechungsleitungen (interrupts) dem System mitgeteilt. Der Begriff Unterbrechung bezieht sich auf das System als Ganzes.
Der für die Behandlung des externen Ereignisses zuständige RT-Prozeß wird durch dieses "Unterbrechungssignal" wieder aktiviert.

Bis zu drei Prozesse können privilegiert sein (s.a. Kapitel 1.9., Konfigurierung), d.h. durch externe Ereignisse aktiviert werden.

2.4.1. Allgemeines

Jeweils 8 oder 16 Signale werden auf einer Karte zusammengefaßt. Die physikalische Aufteilung hat zwar prinzipiell keine Auswirkung auf RT-PASCAL, jedoch kann innerhalb eines Systems nur ein Kartentyp (8 oder 16 Signale) eingesetzt werden, da die Operatoren unterschiedlich sind.
Nach Aktivierung des Prozeßsystems durch das Dienstprogramm ´SERV,START´ sind alle Unterbrechungseingänge verriegelt.

Durch Aufruf der Prozedur ´ENABLE´ können Eingänge einzeln zum Öffnen angemeldet werden. Das Öffnen der angemeldeten Eingänge geschieht jedoch erst, wenn die Prozedur ´AWAITINTERRUPT´ aufgerufen wird.

Aufrufe der Prozedur ´ENABLE´ bewirken ein Setzen der im gemeinsamen Datenbereich gespeicherten Maskenbits. Es befindet sich damit immer ein aktuelles Abbild der vom privilegierten Prozeß gewünschten freien bzw. gesperrten Interrupteingänge im Hauptspeicher.

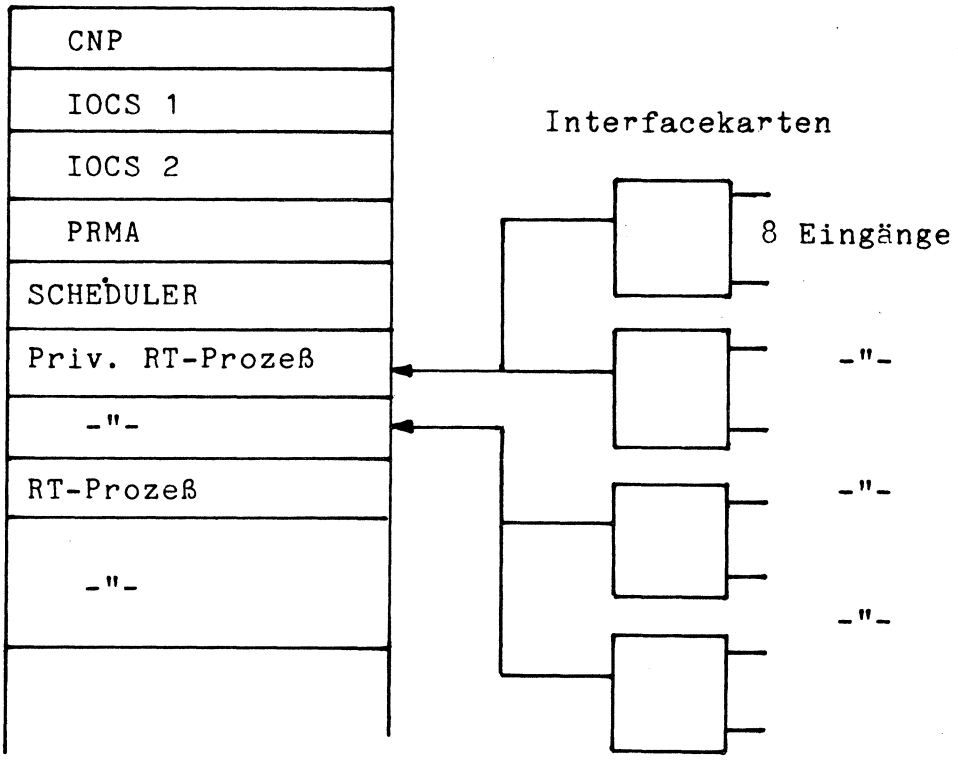
Diese Maske wird zum Zeitpunkt des Aufrufes "AWAITINTERRUPT" an die Eingangskarten übertragen. Durch die gesetzten ENABLE-Bits wird die Durchschaltung eines Unterbrechungssignals ermöglicht.

Wird der RT-Prozeß, der durch den Aufruf der Prozedur ´AWAITINTERRUPT´ in den Wartezustand versetzt wurde, bei Eintreffen eines externen Ereignisses fortgeführt, so werden sofort alle zu diesem Prozeß gehörenden Eingänge (s.a. Kapitel 1.9., Konfigurierung) gesperrt. Dieses verhindert das Wirksamwerden weiterer externer Ereignismeldungen.

Die Öffnungsmeldung bleibt jedoch bestehen, sodaß die Eingänge bei Aufruf der Prozedur ´AWAITINTERRUPT´ sofort wieder geöffnet werden.
Eingänge können nur explizit durch Aufruf der Prozedur ´DISABLE´ geschlossen werden.

Abhängig vom gewählten Kartentyp können maximal 128 oder 256 externe Ereignisse erkannt und verarbeitet werden (s.a. Kapitel 1.9., Konfigurierung). Alle Interrupteingänge eines privilegierten Prozesses haben die gleiche Priorität.

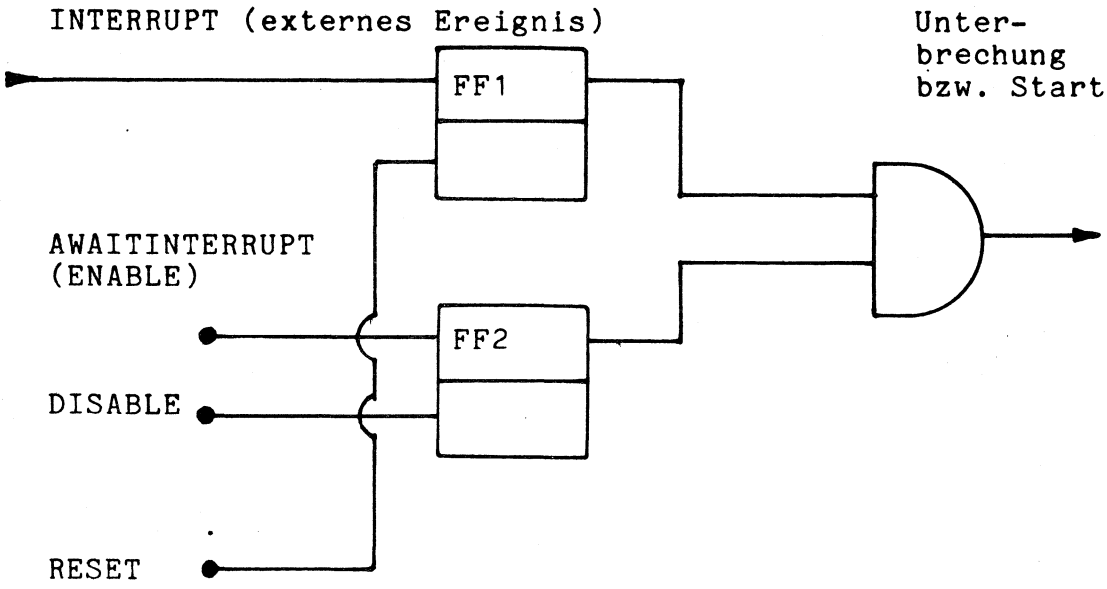
Beispiel eines Anschlusses



2.4.1.1. Erzeugung des Unterbrechersignals

Das Auftreten eines externen Ereignisses wird durch einen binären Speicher aufgefangen (FF1). Dieser Speicher kann durch Aufruf der Prozedur RESET zurückgesetzt werden.
Ein externes Ereignis kann nur dann eine Unterbrechung des Systems, d.h. einen Start des definierten RT-Prozesses auslösen, wenn der Eingang durch den Schalter FF2 freigegeben ist.

Eine Freigabe bzw. Sperre erfolgt durch die Prozeduren AWAITINTERRUPT und DISABLE.



2.4.2. Freigeben eines Eingangs für externe Ereignisse

Das Freigeben eines Eingangs wird durch Setzen eines korrespondierenden Bits innerhalb des gemeinsamen Hauptspeicherbereichs vorbereitet. Die tatsächliche Freigabe erfolgt zum Zeitpunkt des Aufrufs ´AWAIT-INTERRUPT´ (s.a. 2.4.4., Warten auf ein externes Ereignis). Es können durch einen RT-Prozeß nur die Eingänge freigegeben werden, die bei der Konfigurierung (s.a. Kap. 1.9.) diesem zugeordnet wurden.

Aufruf:

```
ENABLE (e-nr)
```

Erläuterung des Parameters:

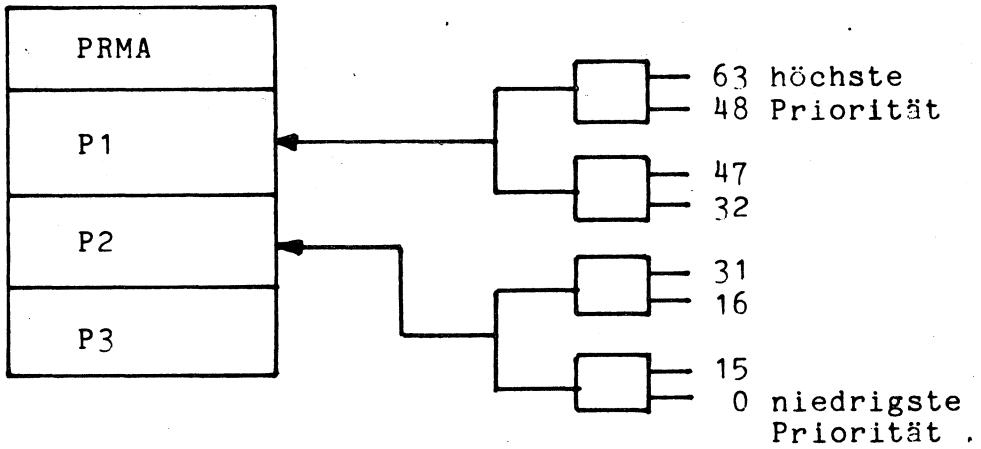
- e-nr : Ereignisnummer

  Typ : BYTE;
  Variable oder Konstante

  Wertebereich : ergibt sich aus der Projektierung und Konfigurierung

Durch den Parameter (e-nr) wird einer der Eingänge definiert, der bei einem nachfolgenden Aufruf der Prozedur ´AWAITINTERRUPT´ freigegeben werden soll. Die anzugebende Nummer liegt im Bereich von 0 bis 127, wenn sogenannte 8-Bit-Karten bzw. im Bereich von 0 bis 255, wenn 16-Bit-Karten verwendet werden.

Aufteilung der Eingänge bei vier 16-Bit Karten



P1, P2 = privilegierte RT-Prozeße
P3 = RT-Prozeß
PRMA = Betriebssystemprozeß (Prozeßmanager)

Die Prioritäten der einem Prozeß zugeordneten Eingänge sind gleich.

Die Freigabe von Eingängen ist privatisiert, d.h. ein RT-Prozeß kann nur die bei ihm wirksam werdenden Eingänge freigegeben.
Diese Zuordnung wiederum wird bei der Projektierung eines Systems bereits festgelegt und im Werk verdrahtet. Die bei der Konfigurierung (Kapitel 1.9.) notwendigen Angaben müssen diese Gegebenheiten widerspiegeln.

SYSTEMSTATE

OK, ILLIOADDRESS, CMDPRIVILEGED

siehe dazu Kapitel 2.11.2.

2.4.3. Sperren eines Eingangs für externe Ereignisse

Im Gegensatz zur Freigabe eines Eingangs wirkt der Aufruf der Prozedur 'DISABLE' auch direkt auf den Sperrmechanismus der Eingangskarte.
Die Sperre von Eingängen kann durch einen beliebigen Prozeß ausgeführt werden. Somit ist es insbesondere dann möglich, einen Eingang zu verriegeln, wenn der zu diesem Eingang gehörende Prozeß sich im Status "Wartend auf ein externes Ereignis" befindet.

Aufruf:

```
DISABLE (e-nr)
```

Erläuterung des Parameters:

- e-nr : Ereignisnummer

  Typ : BYTE;
  Variable oder Konstante

  Wertebereich : ergibt sich aus der Projektierung und Konfigurierung

  Der Parameter (e-nr) spezifiziert einen Eingang für externe Ereignisse.

SYSTEMSTATE

OK, ILLIOADRESS

siehe dazu Kapitel 2.11.2.

2.4.4. Warten auf ein externes Ereignis

Der Aufruf der Prozedur ´AWAIRINTERRUPT´ führt zur Suspendierung des aufrufenden RT-Prozesses. Gleichzeitig wird die im gemeinsamen Hauptspeicher (COMMON) abgelegte und durch die Prozeduren ´ENABLE´ und ´DISABLE´ manipulierbare Maske (s.a. Kap. 2.4.2. und 2.4.3.) an den Verriegelungsmechanismus der Eingangskarten ausgegeben. Die auszugebenden Masken sind den RT-Prozessen eindeutig zugeordnet.
Der suspendierte RT-Prozeß wird fortgesetzt, sobald ein zugeordnetes externes Ereignis erkannt und gemeldet wurde. Zur Identifizierung des externen Ereignisses wird dem RT-Prozeß die Nummer des Ereignisses übergeben.

Aufruf:

```
AWAITINTERRUPT (e-m)
```

Erläuterung des Parameters:

- e-m : Ereignismenge

  Typ : INTERRUPTSET;
  Variable

  Wertebereich : ergibt sich aus der Projektierung und Konfigurierung

In der Ereignismenge sind alle geöffneten und auch eingetretenen Ereignisse gesetzt.
Diese Prozedur kann nur von RT-Prozessen aufgerufen werden, die als privilegierte gekennzeichnet sind (s.a. Kap. 1.9., Konfigurierung).
Bevor der suspendierte RT-Prozeß bei Eintreffen eines externen Ereignisses wieder aktiviert wird, werden die, diesem RT-Prozeß zugeordneten Eingänge verriegelt. Das Abbild der Eingänge (durch ENABLE und DISABLE bewertet) bleibt erhalten und wird beim nächsten Aufruf der Prozedur ´AWAITINTERRUPT´ wieder ausgegeben.

2.4.5. Rücksetzen externer Ereignisse

Das Auftreten eines externen Ereignisses wird durch ein Register direkt am Eingang der 8-Bit bzw. 16-Bit Karte gespeichert. Die Speicherung erfolgt binär, sodaß nur zwischen "Ereignis aufgetreten" bzw. "Ereignis nicht aufgetreten" unterschieden werden kann.
Tritt das Ereignis mehrfach auf, bevor die erstmalige Bearbeitung beendet ist, so kann die Anzahl nicht festgestellt werden. Der binäre Speicherplatz für ein externes Ereignis muß nach Bearbeitung durch den zuständigen RT-Prozeß explizit zurückgesetzt werden, da das Ereignis sonst als nicht bearbeitet gilt und beim nächsten Aufruf der Prozedur ´AWAITINTERRUPT´ sofort wieder wirksam wird (sofern nicht DISABLE aufgerufen wurde).

Aufruf:

```
RESETINT (e-nr)
```

Erläuterung des Parameters:

- e-nr : Ereignisnummer

  Typ : BYTE;
  Variable oder Konstante

  Wertebereich : ergibt sich aus der Projektierung und Konfigurierung

  Der Parameter (e-nr) spezifiziert einen Eingang für externe Ereignisse.

SYSTEMSTATE

OK, ILLIOADRESS

siehe dazu Kapitel 2.11.2.

2.5. Änderung der Prozeßpriorität

Die Prozedur erlaubt nur die Priorität des aufrufenden Prozesses zu ändern, d.h. ein Prozeß kann nur seine eigene Priorität festlegen.

Zu beachten ist jedoch hierbei, daß es sich bei dem Grundbetriebssystem um ein Time-Sharing-System handelt.
Alle Prozesse mit gleicher SW-Priorität sind gleichberechtigte Timesharing-Teilnehmer. Unterschiedliche Prioritäten heben das Timesharingverhalten auf und erzeugen eine strenge Multiprogrammingverwaltung.

Aufruf:

```
SETPRIORITY (p)
```

Erläuterung des Parameters

- p : Priorität

  Typ : BYTE;
  Variable oder Konstante

  Wertebereich : 0 <= p <= 63

  Der Prozeß wird mit der angegebenen Priorität fortgesetzt.

SYSTEMSTATE

OK, PARAMERROR

siehe dazu Kapitel 2.11.2.

2.6. Betriebsmittelverwaltung

Für die Verwaltung von Betriebsmitteln wird die Semaphorentabelle des Grundbetriebssystems TSOS benutzt.

Die Tabelle hat folgenden Aufbau:

0 31	Langzeitbelegung von Geräten
32 50	Verwaltung von internen Betriebsmitteln des Betriebssystems; Belegung und Freigabe darf nur durch das Betriebssystem erfolgen
51 95	freie Semaphorennummern
96 127	Kurzzeitbelegung von Geräten (Auftrag); Belegung und Freigabe darf nur durch das Betriebssystem erfolgen

Die Nummern 32 bis 50 und 96 bis 127 dürfen unter keinen Umständen durch RT-Prozesse belegt oder freigegeben werden.
Die Ausführungen des Betriebssystems würden zwangsläufig gestört werden.

2.6.1. Belegung von Semaphoren

Der Aufruf der Prozedur 'REQSEMA' führt zu einer Suspendierung des aufrufenden RT-Prozesses, falls die Semaphorennummer schon durch einen anderen Prozeß belegt ist.
Der suspendierte Prozeß wird in eine Warteschlange zu diesem Semaphor eingereiht.
Er wird vom Betriebssystem selbständig wieder aktiviert, sobald die Belegung des Semaphors erfolgen konnte.

Aufruf:

```
REQSEMA (se-nr)
```

Erläuterung des Parameters

- se-nr : Semaphorennummer

  Typ : BYTE;
  Variable oder Konstante

  Wertebereich : 0 <= se-nr <= 127

  Um Störungen des Betriebssystems zu vermeiden, dürfen für Gerätebelegungen nur die Nummern von 0 bis 31, für beliebige Betriebsmittel nur die Nummern 50 bis 95 verwendet werden.

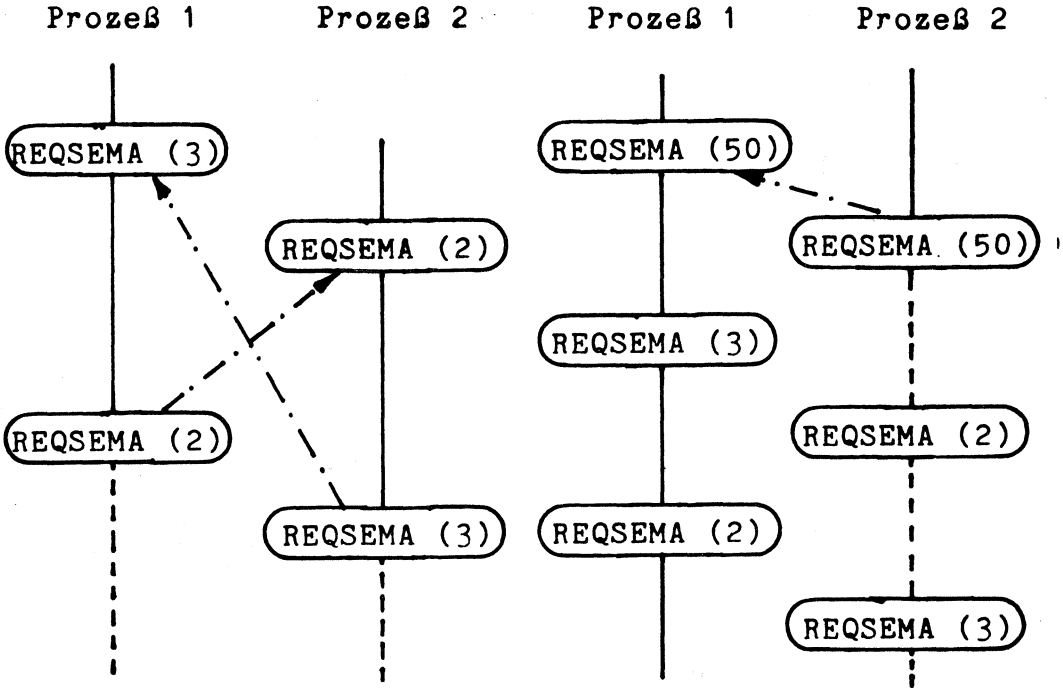
SYSTEMSTATE

OK, PARAMERROR

siehe dazu Kapitel 2.11.2.

Da die unüberlegte Belegung von Semaphoren zum Blockieren von Prozessen führen kann, sei an dieser Stelle noch einmal auf die Reihenfolge bei der Belegung mehrerer Semaphoren hingewiesen.

Entweder wird die Anforderung mehrerer Betriebsmittel "gleichzeitig" durchgeführt oder aber alle Prozesse belegen Semaphoren immer in der gleichen Reihenfolge (z.B. aufsteigende Nummer).
Die "gleichzeitige" Anforderung geschieht dadurch, daß der Anforderungsvorgang als weiteres Betriebsmittel definiert wird.



Blockierung Abhilfe

Bei obigem Beispiel wurde die Blockierung durch Einführung und Verwendung des Semaphors (50) beseitigt.
Prozeß 2 wird fortgesetzt sobald Prozeß 1 den Semaphor (50) freigibt. Diese Freigabe sollte immer als letzte erfolgen. Im Betriebssystem selbst ist keine Automatik für die Erkennung und Behandlung von Verriegelungen (Deadlocks) vorgesehen.
Verriegelungen dieser Art sind gefährlich, da sie nicht unbedingt während der Testphase auftreten. Das Auftreten ist abhängig von der zeitlichen Reihenfolge der Ausführungen.

2.6.2. Freigabe von Semaphoren

Durch Aufruf der Prozedur RELSEMA können einzelne Betriebsmittel (explizite Angabe der Semaphorennummer) oder aber auch alle bisher durch den aufrufenden RT-Prozeß angeforderten Betriebsmittel (Angabe von -1) freigegeben werden.
Ein RT-Prozeß kann nur selbst angeforderte Betriebsmittel freigeben.

Das zwangsweise Entziehen von Betriebsmitteln ist nur dem Master-/Operatorprozeß im Kommandomode möglich.

Aufruf:

```
RELSEMA (se-nr)
```

Erläuterung des Parameters

- se-nr : Semaphorennummer

  Typ : INTEG;
  Variable oder Konstante

  Wertebereich : -1 oder
  0 <= se-nr <= 127

  Dabei ist, wie auch bei der Anforderung von Betriebsmitteln, zu beachten, daß den RT-Prozessen nur die Nummern 0 bis 31 und 50 bis 95 zur Verfügung stehen.

SYSTEMSTATE

OK, NOSEMANUMBER

siehe dazu Kapitel 2.11.2.

2.7. Systemzeit setzen und abfragen

Uhrzeit und Datum werden vom Betriebssystem geführt. Dabei werden bezüglich der Monatstage auch Schaltjahre berücksichtigt.

Das Setzen der Zeit und des Datums kann über Kommandos des Betriebssystems XOS oder über nachfolgend beschriebene Prozeduren vorgenommen werden.

Die Systemzeit darf jedoch nicht verändert werden, wenn Zeitaufträge des Schedulers noch nicht fällig geworden sind, da die Zeitrelationen zwischen den Aufträgen sich ändern können.

2.7.1. Systemzeit setzen

Bei Aufruf der Funktion WRITETIME und des aktivierten "Scheduler" werden alle Zeitaufträge ausgeführt, die im Intervall (alte Systemzeit, neue Systemzeit) liegen.

Aufruf:

```
WRITETIME (t-d)
```

Erläuterung des Parameters

- t-d : time, date (Zeit und Daten)

  Typ :
```
RECORD
  MSEC : 0..16777215;
  SEC, MIN, HOUR, DAY, MON,
  YEAR : 0..255
END
```

  Wertebereich :
```
0 <= MSEC <= 86.400
0 <= SEC <= 59
0 <= MIN <= 60
0 <= HOUR <= 23
0 <= DAY <= 31
0 <= MON <= 12
0 <= YEAR <= 255
```

  Der erste Parameter (MSEC) ist modulo 1000 anzugeben.

2.7.2. Systemzeit abfragen

Ebenso wie beim Setzen der Systemzeit wird auch hier als Schnittstelle der Datentyp RECORD verwendet.

Aufruf:

```
READTIME (t-d)
```

Erläuterung des Parameters

- t-d : time, date (Zeit und Datum)

  Typ :
```
RECORD
 MSEC : 0..16777215;
 SEC,
 MIN,
 HOUR,
 DAY,
 MON,
 YEAR : 0..255
END
```

  Wertebereich : siehe Kapitel 2.7.1. (Setzen der Systemzeit)

2.8. Ausgabe des Prozeßsystemstatus

Die Prozedur 'STATUS' erlaubt die Ausgabe aller relevanten Informationen über den aktuellen Zustand des gesamten Prozeßsystems.
Darüber hinaus werden die Konfigurierungsgrößen ausgegeben (s. dazu Kapitel 1.9.). Wird der zweite Parameter mit TRUE bewertet, so wird der Inhalt des Common Data Pools an das spezifizierte Gerät ausgegeben.

Aufruf:

```
STATUS (ge,dump)
```

Erläuterung der Parameter:

- ge : Ausgabegerät;
  Variable oder Konstante

  Typ : BYTE

- dump : Schalter;
  Variable oder Konstante

  Typ : BOOLEAN
  TRUE = Ausgabe des CDP's

SYSTEMSTATE

wird nicht beeinflußt

2.9. Steuerung und Überwachung einer Anwendung

Die Steuerung und Überwachung einer Anwendung wird im wesentlichen durch folgende Grundkomponenten in der Prozeßperipherie vorgenommen:

a) digitale und analoge Eingangsdaten

b) digitale und analoge Ausgangsdaten

Analoge Daten dienen zur wechselseitigen Information zwischen dem Prozeßrechner und dem Anwendungsprozeß über den jeweiligen Istzustand (Anwendungsprozeß -> Prozeßrechner) bzw. Sollzustand (Prozeßrechner -> Anwendungsprozeß). Die Initiative für den Datentransfer geht dabei immer vom Prozeßrechner aus.

Bei digitalen Daten unterscheidet man zwischen statischen und dynamischen Daten. Diese Unterscheidung ergibt sich aufgrund des Anschlusses an den Rechner.
Statische digitale Daten dienen ebenso wie Analogdaten zur wechselseitigen Information zwischen Anwendungsprozeß und Prozeßrechner. Die Initiative geht auch hier vom Prozeßrechner aus.
Dynamische Signale (Daten) informieren den Rechner über das spontane Auftreten von Ereignissen. Die durch externe Ereignisse erzeugten Signale dürfen Impulsform haben, da der Rechnereingang zur Speicherung von Impulsen ausgelegt ist.
Das Eintreffen eines solchen Signals führt zu einer Unterbrechung des Gesamtsystems und einer Reaktion durch den privilegierten RT-Prozesses.
Die Prozeduren für das Erkennen dieser Signale sind im Kapitel 2.3. (Behandlung externer Ereignisse) detailliert erläutert.

2.9.1. Voraussetzungen

2.9.1.1. Adressen der Schnittstellen

Für die Ein- und Ausgabe statischer digitaler Daten können maximal jeweils 64 Schnittstellen angeschlossen werden. Für die Ein- und Ausgabe analoger Daten können jeweils bis zu 16 Schnittstellen angeschlossen werden.
Bei der Installation der Schnittstellen muß darauf geachtet werden, daß Karten eines Typs (digitale Eingänge, digitale Ausgänge, analoge Eingänge, analoge Ausgänge) in einem zusammenhängenden Bereich ohne unbelegte Lücken montiert werden.

Eine Belegung könnte so aussehen:

'0000	Registerpool
'0FFF	
'1000	Geräteperipherie
'1FFF	
'2000	Sonderprozeßperipherie
'2FFF	
'3000	digitale Eingänge (statisch)
'33FF	
'3400	digitale Ausgänge (statisch)
'37FF	
'3800	digitale Eingänge (dynamisch)
'39FF	
'3A00	analoge Eingänge
'3AFF	
'3B00	analoge Ausgänge
'3BFF	

2.9.1.2. Typ der Schnittstellenkarten

Mit Hilfe von RT-PASCAL können sowohl Hardwarekonfigurationen mit 16-Bit Karten, als auch solche mit 32-Bit-Karten für die Ein- und Ausgabe statischer digitaler Daten gesteuert werden. Da jedoch für die beiden Typen unterschiedliche Verarbeitungs- routinen vorgesehen sind, ist die parallele Benutzung beider Kartentypen nicht möglich.
Außerdem müssen bei Verwendung von 32-Bit Karten für die Ein- und Ausgabe digitaler Daten, 16-Bit Karten für den Anschluß dynamischer Signale (Interrupts) benutzt werden.
Ebenso impliziert die Wahl von 16-Bit Karten für die Ein- und Ausgabe digitaler Daten die Verwendung von 8-Bit Interruptkarten.

2.9.1.3. Platzbedarf im COMMON-Speicher

Der benötigte Platz für das Prozeß-E/A-System errechnet sich aus folgender Formel:

```
x := Anzahl der (1-Bit) Digitalausgänge/8 +
     Anzahl der Interrupteingänge/8
```

Die Dimension von x wird in Byte angegeben.

Die Basisadressen der einzelnen Bereiche (z.B. ´3400 für digitale Ausgänge) sind im Prinzip (Verdrahtung im Werk !) frei wählbar und müssen bei der Konfigurierung des Systems bekanntgegeben werden (s.a. Kapitel 1.9., Konfigurierung).
Ebenso ist die Anzahl der Karten zu jedem Typ (Digitale Ausgänge, Interrupteingänge) bei der Konfigurierung anzugeben.

2.10. Aufrufe für die Ein- und Ausgabe von Prozeßdaten

Ebenso wie die Aufrufe für die Synchronisation und Erkennen externer Ereignisse wird die Ein- und Ausgabe von Prozeßdaten über Prozeduren vorgenommen, die im Prolog des PASCAL-Übersetzers definiert sind (Dateiname XOSPRO).

Es sind folgende Prozeduren vordefiniert:

Eingaben

READBIT	: Eingabe eines Bits
READWORD	: Eingabe eines 16-Bit-Wortes
READDECIMAL	: Eingabe eines 16-Bit-Wortes mit Interpretation des gelesenen Wertes als BCD-Zahl mit vier Ziffern (0..9999)
READANALOG	: Eingabe eines Analogwertes ADE 30

Ausgaben

WRITEBIT	: Ausgabe eines Bits
WRITEWORD	: Ausgabe eines 16-Bit-Wortes
WRITEDECIMAL	: Ausgabe eines 16-Bit-Wortes in BCD-Format (0..9999)
WRITEANALOG	: Ausgabe eines Wertes an einen Analogausgang (DAU 1010/DAI 1020)

Zu beachten ist unbedingt, daß die Anschlüsse für die Eingabe bzw. Ausgabe von digitalen Daten je nach Prozedur unterschiedlich interpretiert werden.

So führen die drei folgenden Sequenzen alle zum gleichen Ergebnis:

1) WRITEDECIMAL (0, 10)

2) WRITEWORD (0, 16)

3) FOR BITZAEHL := 0 TO 15 DO WRITEBIT (BITZAEHL,0);

   WRITEBIT (4, 1)

Diese unterschiedliche Interpretation derselben Anschlüsse gilt ebenso für READBIT, READWORD und READDECIMAL.

2.10.1. Eingabe einzelner Bit

Durch Aufruf folgender Prozedur wird der Wert eines einzelnen binären Eingangs übertragen.

Aufruf:

```
READBIT (bit-nr, bit-wert)
```

Erläuterung der Parameter

- bit-nr : Nummer des zu lesenden Eingangs

  Typ : BYTE;
  Variable oder Konstante

  Wertebereich : 0 <= bit-nr <= 1023 (16-Bit Karten)
  (maximal) 0 <= bit-nr <= 2047 (32-Bit Karten)

  Der tatsächliche Wertebereich ergibt sich aus der Konfigurierung. Der Eingang mit der Nummer 0 entspricht dem niederwertigsten Bit auf Karte 0, wobei Karte 0 auf der Adresse 'Inputbasis' liegt. Der Eingang mit der Nummer 2047 entspricht dem höchstwertigsten Bit auf der Karte 63. Diese Eingangsadresse existiert allerdings nur bei Verwendung von 32-Bit-Karten; 16-Bit-Karten lassen als höchste Adresse (bit-nr) 1023 zu.

- bit-wert : Parameter für die Aufnahme des Eingangswertes

  Typ : BIT;
  Variable

SYSTEMSTATE

OK, ILLIOADRESS

siehe dazu Kapitel 2.11.2.

2.10.2. Eingabe eines Wortes (duale Interpretation)

Durch die Prozedur READWORD werden 16 binäre Eingänge zu einem Wort zusammengefaßt. Die Adressierung der so gebildeten Worte erfolgt gruppenweise.

Aufruf:

```
READWORD (w-nr, w-wert)
```

Erläuterung der Parameter

- w-nr : Nummer des zu lesenden Eingangswortes

  Typ : BYTE;
  Variable oder Konstante

  Wertebereich : 0 <= w-nr <= 63
  0 <= w-nr <= 127

  Bei 16-Bit-Karten entspricht die Nummer des Wortes der Kartennummer. 32-Bit-Karten haben dagegen Platz für jeweils zwei Worte. Somit sind bei 16-Bit-Karten maximal 64 und bei 32-Bit Karten maximal 128 Worte möglich.

- w-wert : Parameter für die Aufnahme des Eingangswertes

  Typ : BYTE2;
  Variable

  Wertebereich : ist identisch mit dem Typ

SYSTEMSTATE

OK, ILLIOADDRESS

siehe Kapitel 2.11.2.

2.10.3. Eingabe eines Wortes (dezimale Interpretation)

Durch die Prozedur READDECIMAL werden -ebenso wie bei READWORD- 16 binäre Eingänge zu einem Wort zusammengefaßt. Die Adressierung dieser so gebildeten Worte erfolgt gruppenweise. Aufgrund dieser Interpretation ergeben sich vier BCD-Zahlen (Tetraden). Pseudotetraden (sedezimal 'A..'F) werden durch die Prozedur unterdrückt.

Aufruf:

```
READDECIMAL (w-nr, w-wert)
```

Erläuterung der Parameter:

- w-nr : Nummer des zu lesenden Eingangswortes

  Typ : BYTE;
  Variable oder Konstante

  Wertebereich : 0 <= w-nr <= 63 oder
  0 <= w-nr <= 127

  Bei 16-Bit-Karten entspricht die Nummer des Wortes der Kartennummer; 32-Bit-Karten haben dagegen Platz für jeweils zwei Worte. Somit sind bei 16-Bit-Karten maximal 64 und bei 32-Bit Karten maximal 128 Worte möglich.

- w-wert : Parameter für die Aufnahme des Eingangswertes

  Typ : BYTE2;
  Variable

  Wertebereich : 0 <= w-wert <= 9999

SYSTEMSTATE

OK, ILLIOADRESS

siehe dazu Kapitel 2.11.2.

2.10.4. Eingabe eines Analogwertes

Für die Aufnahme von Analogwerten stehen zwei unterschiedliche Prozeduren zur Verfügung. Der Analog-Digitalwandler ADE30 ist mit einem Multiplexer ausgerüstet; somit kann neben der Wandlernummer auch noch die Multiplexerkanalnummer angegeben werden.

Dagegen wird der Analog-Digitalwandler ADE12 nur über die Wandlernummer angesprochen. Die installierten ADE30-Wandler sind von 0 bis n-1 (bei n Eingängen) durchnummeriert; dabei entspricht Eingang 0 demjenigen ADE30, der auf der niedrigsten Adresse montiert ist und Eingang n-1 demjenigen ADE30, der auf der höchsten Adresse montiert ist.

Aufruf:

```
READANALOG (w-nr, k-nr, wert)
```

Erläuterung der Parameter

- w-nr : Wandlernummer

  Typ : BYTE;
  Variable oder Konstante

  Wertebereich : 0 <= w-nr <= 15

  Die höchste angebbare Wandlernummer ist abhängig von der tatsächlichen Installation und den Angaben bei der Konfigurierung.

Auch bei der Kanalnummer ist der größtmögliche Wert, abhängig von der Auslegung des Multiplexers innerhalb des Wandlers.

- k-nr : Kanalnummer
  (nur beim Wandler ADE30)

  Typ : BYTE;
  Variable oder Konstante

  Wertebereich : 0 <= k-nr <= 31

- wert : Analogwert

  Typ : BYTE2;
  Variable

  Wertebereich : 0 <= wert <= 4095

SYSTEMSTATE

OK, ILLIOADRESS

siehe dazu Kapitel 2.11.2.

2.10.5. Ausgabe einzelner Bit

Da die Ausgabe digitaler Werte an die Prozeßperipherie nur byteweise (8 Bit) möglich ist, wird ein aktuelles Abbild der Zustände aller Ausgänge im Speicher gehalten (IOIMAGE). Dieses Abbild wird zur Findung der fehlenden 7 Bit bei der Ausgabe eines Bit benutzt.

Aufruf:

```
WRITEBIT (bit-nr, bit-wert)
```

Erläuterung der Parameter:

- bit-nr : Nummer des digitalen Ausgangs

  Typ : BYTE2;
  Variable oder Konstante

  Wertebereich : 0 <= bit-nr <= 1023 (16-Bit-Karten)
  0 <= bit-nr <= 2047 (32-Bit-Karten)

Der tatsächliche Wertebereich ergibt sich aus der Konfigurierung. Der Eingang mit der Nummer 0 entspricht dem niederwertigsten Bit auf Karte 0, wobei Karte 0 auf der Adresse 'Outputbasis' liegt.
Der anzugebende Wert wird sowohl im Hauptspeicher (IOIMAGE) geführt als auch direkt an die Ausgangsleitungen weitergegeben.

Der Eingang mit der Nummer 2047 entspricht dem höchstwertigsten Bit auf der Karte 63. Diese Eingangsadresse existiert allerdings nur bei Verwendung von 32-Bit-Karten; 16-Bit-Karten lassen als höchste Adresse (bit-nr) 1023 zu.

- bit-wert : Auszugebender Wert

  Typ : BIT;
  Variable oder Konstante

SYSTEMSTATE

OK, ILLIOADRESS

siehe Kapitel 2.11.2.

2.10.6. Ausgabe eines Wortes (duale Interpretation)

Durch die Prozedur 'WRITEWORD' werden 16 binäre Ausgänge zu einem Wort zusammengefaßt. Die Adressierung der so gebildeten Worte erfolgt gruppenweise.

Aufruf:

```
WRITEWORD (w-nr, w-wert)
```

Erläuterung der Parameter

- w-nr : Nummer des digitalen Ausgangs

  Typ und Wertebereich siehe Kapitel 2.10.2. (Eingabe eines Wortes)

- w-wert : Auszugebender Wert

  Typ und Wertebereich siehe Kapitel 2.10.2.

SYSTEMSTATE

OK, ILLIOADRESS

siehe dazu Kapitel 2.11.2.

2.10.7. Ausgabe eines Wortes (dezimale Interpretation)

Ebenso wie bei der Prozedur ´WRITEWORD´ werden auch bei dieser Prozedur 16 binäre Eingänge als ein Wort betrachtet. Desweiteren werden dann noch jeweils vier binäre Eingänge zu einer Tetrade zusammengefaßt. Die Adressierung erfolgt gruppenweise zu jeweils vier Tetraden. Bei Überschreitung des Wertebereichs (w-wert > 9999) wird die Prozedur nicht ausgeführt.

Aufruf:

```
WRITEDECIMAL (w-nr, w-wert)
```

Erläuterung der Parameter

- w-nr : Nummer des Ausgangs

  Typ und Wertebereich siehe Kapitel 2.10.3. Eingabe eines Wortes (dezimale Interpretation).

Der auszugebende Wert wird sowohl im Hauptspeicher (IOIMAGE) geführt, als auch direkt an die Ausgangsleitungen weitergegeben. Das Protokollieren des Ausgangsstatus wird für die Ausgabe einzelner Bit benötigt.

- w-wert : Auszugebender Wert

  Typ : BYTE2;
  Variable oder Konstante

  Wertebereich : 0 <= w-wert <= 9999

SYSTEMSTATE

OK, ILLIOADRESS, IOOVERFLOW

siehe dazu Kapitel 2.11.2.

2.10.8. Ausgabe eines Analogwertes

Für die Ausgabe analoger Werte stehen zwei Digital-Analogwandler zur Verfügung:

einfacher Umsetzer : DAU 1010

integrierender Umsetzer : DAI 1020

Beide Wandler arbeiten im Wertebereich zwischen 0 und 1023.

Aufruf:

```
WRITEANALOG (w-nr, wert)
```

Erläuterung der Parameter:

- w-nr : Wandlernummer

  Typ : BYTE;
  Variable oder Konstante

  Wertebereich : 0 <= wert <= 1023

  Die höchste mögliche Wandlernummer ist abhängig von der tatsächlichen Installation und den daraus resultierenden Angaben bei der Konfigurierung.

Bei Überschreitung des Wertebereiches, wird der Ausgangswert nicht geändert und eine Fehlermeldung an den aufrufenden Prozeß übergeben.

- wert : Analogwert

  Typ : BYTE2;
  Variable oder Konstante

  Wertebereich : 0 <= wert <= 1023

SYSTEMSTATE

OK, ILLIOADRESS, IOOVERFLOW

siehe dazu Kapitel 2.11.2.

2.10.9. Schnittstellen

Standardinterrupteingänge:	PDSE 12.60 PDSE/FK PDSE/R PDSE 16/5
Statische Eingänge:	PSSE 16/12.60 PSSE 16/FK PSSE 16/5 PSSE 32/5
Speichernde Ausgänge:	PSSA 16 PSSA 16/FK PSSA 16/R PSSA 32
Analog-Digitalanwender:	ADE 30
Digital-Analogwandler:	DAU 1010 DAI 1020

2.11. Fehlerbehandlung

2.11.1. Allgemeines

Werden bei der Ausführung einer RTMS-Routine fehlerhafte Zustände erkannt oder befinden sich Parameter außerhalb des erlaubten Wertebereichs, so wird dem aufrufenden RT-Prozeß eine Information über die Art des Fehlers übergeben.
Die Statusvariable ´SYSTEMSTATE´ ist jeweils nach einem RTMS-Aufruf zu prüfen. Sie ist in der IMPORTS-Liste aufzuführen und im Hauptprogramm wie folgt zu vereinbaren:

```
VAR     SYSTEMSTATE     :     SYSSTATE
```

2.11.2. Fehlermeldung

Die Variable vom Typ SYSSTATE kann folgende Zustände annehmen.

OK :
Kein Fehler aufgetreten

RECUNKNOWN :
Empfängerprozeß nicht bekannt

RENOTDRY :
Empfängerprozeß nicht bereit. Dieser Zustand kann nur bei Aufruf der Prozedur ´SEND´ auftreten, wenn der Parameter ´es´ (empfängerstatus) mit WAITING bewertet ist.

RECBUFOFL :
Der Puffer des Empfängerprozesses ist übergelaufen. Der Empfängerprozeß erhält beim Abholen der nächsten Nachricht die Meldung BUFFEROFL (Pufferüberlauf). Dieser Zustand kann nur bei Aufruf der Prozedur ´SEND´ auftreten.

BUFFERMT :
Eigener Nachrichtenpuffer ist leer. Dieser Zustand kann nur bei Aufruf der Prozedur ´RECEIVE´ auftreten.

BUFFEROFL :
Eigener Nachrichtenpuffer ist übergelaufen; Verlust von Nachrichten möglich. Der Absender der Nachricht, die zum Überlauf führte, hat bereits die Meldung RECBUFOFL erhalten. Dies gilt nicht für Nachrichten, die vom Scheduler stammen; dieser benutzt die Zustandsanzeige ´TIMECONDITION´ (s.u.). Dieser Zustand kann nur bei Aufruf der Prozeduren ´RECEIVE´ und ´AWAITMESSAGE´ auftreten.

PARAMERROR :
Parameterfehler; dieser Zustand wird nur bei Aufruf der Prozedur ´CANCELSCHEDULE´ und ´SETPRIORITY´ gemeldet.

NOTASK :
Dieser Zustand wird nur bei Aufruf der Prozedur ´CANCELSCHEDULE´ gemeldet.

NOTFOUND :
Es existiert kein Auftrag, der dem angegebenen Parameter entspricht. Dieser Zustand kann bei Aufruf der Prozedur ´CANCELSCHEDULE´ auftreten.

CMDPRIVILEDGED :
Die Prozedur darf durch den aufrufenden Prozeß nicht ausgeführt werden. Dieses gilt nur für die Prozeduren ´AWAITINTERRUPT´ und ´ENABLE´.

NOUSER :
Die genannte Ebene des Rechners wurde bei der Konfigurierung nicht genannt.

ILLIOADDRESS :
Es wurde eine nicht vorhandene Ein-/Ausgabeadresse (Prozeßperipherie) angegeben.

IOOVERFLOW :
Der auszugebende Wert ist zu groß; dieser Zustand wird nur bei ´WRITEDECIMAL´ und ´WRITEANALOG´ gemeldet.

TASKLISTOFL :
Die Auftragsliste des Schedulers ist übergelaufen; der letzte Auftrag konnte nicht mehr angenommen werden.

TIMECONDITION :
Dem Scheduler liegt ein Auftrag vor, zyklisch eine Nachricht zu senden; die letzte Nachricht konnte nicht übergeben werden, da der Puffer des Empfängers gefüllt war oder der Empfänger nicht bereit war. Dieser Zustand wird nur bei ´RECEIVE´ oder ´AWAITMESSAGE´ gemeldet.

NOSEMANUMBER :
Die Nummer des angegebenen Semaphors existiert nicht.

NOPOOLINIT:
Der aufrufende Prozeß hat bislang die Prozedur INITPOOL noch nicht aufgerufen (GETPOOL und PUTPOOL)

POOLOVERFLOW:
Bei Aufruf der Prozedur POOLCHECK wurde festgestellt, daß die bei der Konfigurierung festgelegte Länge des CDP für den angegebenen 'commondatapooltyp' nicht ausreicht, oder ein CDP-Zugriff zeigt auf Adressen, die außerhalb des bei der Konfigurierung festgelegten CDP-Bereichs liegen (GETPOOL und PUTPOOL).

DATALENFAULT:
Der Typ der Zielvariablen bei GETPOOL bzw. der Quellvariablen bei PUTPOOL ist ungleich dem Typ der durch die dereferenzierte CDP-Zugriffsvariablen selektierte Teilstruktur des CDP.

POOLLENFAULT:
Dieser Zustand tritt nur ein, wenn mindestens ein Prozeß die Prozedur POOLCHECK aufgerufen hat.
Die CDP-Längendefinition des aufrufenden Prozesses stimmt nicht mit der aktuellen CDP-Länge überein oder der Prozeß hat bisher die Prozedur POOLCHECK nicht aufgerufen.

DIETZ 6***

System-PASCAL
Erweiterung RTMS
Real-Time Management System
Beispielsammlung

Dok.Nr. 2-8206-01-224
Schutzgebühr DM 5,00

DIETZ Computer SYSTEME

3. Beispielsammlung

```
 1   PROGRAM PBRTS1 (KEYB,SCREEN,DRUCK);
 2
 3   (* ANFORDERUNGS-PROZESS *)
 4   (* ERSTELLT AM 09.06.82 *)
 5
 6   IMPORTS SYSTEMSTATE;
 7   CONST
 8     T21=21;
 9     D3=3;
10   TYPE
11     CSTATYPE =(FREIGABE,PUFFERLEER,PUFFERVOLL);
12     STR20 = ARRAY [1..20] OF CHAR;
13     STR8  = ARRAY [1.. 8] OF CHAR;
14     (* IM PROZESS DEFINIERTE STRUKTUR DES COMMON-DATA-POOL *)
15     KUNDENTYPE = RECORD
16                    NAME :STR20;
17                    PLZ : 0:8999;
18                    ORT : STR20;
19                  END;
20   VAR
21     PTCOMMONP1 : ^KUNDENTYPE ;
22
23     (* CDP-ZUGRIFFSVARIABLE *)
24
25     ANSCHRIFT  : KUNDENTYPE ;
26
27     (* SATZVARIABLE FUER DIE DATENAUFNAHME *)
28
29     SYSTEMSTATE : SYSSTATE ;
30
31     (* VARIABLE FUER DIE AUFAHME DES STATUS *)
32
33     KEYB,SCREEN,DRUCK : FILE OF CHAR;
34     MES : RECORD
35             CASE BYTE OF
36                  0 : (E:MESSAGETYPE);
37                  1 : (I:CSTATYPE)
38           END;
39     T:TIMETYPE;
40     P1,P2: 0..10;
41     INITPROCEDURE;
42      BEGIN
43       P1:=0;
44       P2:=1;
45      END;
46     PROCEDURE AUSGABEZEIT;
47      BEGIN
48       READTIME(T);
49       WRITELN(DRUCK,' DATUM :  ',T.DAY:3,'.',T.MONTH:3,'.',T.YEAR:3);
50       WRITELN(DRUCK,' ZEIT  :  ',T.HOUR:3,'.',T.MIN:3,'.',T.SEC:3);
51      END;
```

```
52    BEGIN
53
54     (* OEFFNEN DER DATEIEN (GERAETE) *)
55
56     REWRITE(SCREEN,0,T21);
57     RESET(KEYB,0,T21);
58     REWRITE(DRUCK,0,D3);
59     WRITELN(DRUCK,'AKTIVIERUNG ANFORDERUNGS-PROZESS');
60     AUSGABEZEIT;
61     WRITELN(SCREEN,'WARTEN AUF AKTIVIERUNG DES AUSFUEHRUNGS-PROZESSES');
62
63     (* VERBINDUNG ZUM COMMON-DATA-POOL AUFBAUEN UND GROESSE PRUEFEN *)
64
65     INITPOOL(PTCOMMONP1);
66     POOLCHECK(PTCOMMONP1^);
67
68
69     (* LOESCHEN ALLER EVENTUELL GESPEICHERTEN NACHRICHTEN *)
70
71     CLEARMESSAGES (-1);
72     (* WARTEN AUF FREIGABE DURCH PROZESS P2 *)
73      REPEAT
74       AWAITMESSAGE (MES.E);
75      UNTIL MES.I = FREIGABE;
76
77       (* ZYKLISCHE ENTGEGENNAHME UND UEBERTRAGUNG DER DATEN *);
78
79      REPEAT
80       WRITE(SCREEN,' BITTE NAMEN EINGEBEN ( 20 ZEICHEN) : ');
81       READLN(KEYB,ANSCHRIFT.NAME);
82       WRITE(SCREEN,' POSTLEITZAHL          ( 0..8999)   : ');
83       READLN(KEYB,ANSCHRIFT.PLZ);
84       WRITE(SCREEN,' ORT                   ( 20 ZEICHEN) : ');
85       READLN(KEYB,ANSCHRIFT.ORT);
86
87       (* UEBERTRAGUNG AN DEN COMMON-DATA-POOL *)
88
89       PUTPOOL(PTCOMMONP1^,ANSCHRIFT);
90
91       (* NACHRICHT AN DEN AUSFUEHRUNGS-PROZESS *)
92
93       MES.I:=PUFFERVOLL;
94       SEND( P2,MES.E,WAITING);
95
96       (* WARTEN AUF LEERUNG DES PUFFERS *)
97
98        REPEAT
99         AWAITMESSAGE ( MES.E);
100       UNTIL MES.I=PUFFERLEER;
101     UNTIL ANSCHRIFT.NAME [1]='%';
102
103      (* ENDE DES PROZESSES *)
104
105    WRITELN(DRUCK,' TERMINIERUNG DES ANFORDERUNGS-PROZESSES ');
106    AUSGABEZEIT;
107
108    (* ENDE DES PROGRAMMS *)
109
110   END.
```

```
 1  PROGRAM PBRTS2 (KEYB,SCREEN,DRUCK);
 2
 3  (* ANFORDERUNGS-PROZESS *)
 4  (* ERSTELLT AM 09.06.82 *)
 5
 6  IMPORTS SYSTEMSTATE;
 7  CONST
 8    T22=22;
 9    D3=3;
10  TYPE
11    CSTATYPE =(FREIGABE,PUFFERLEER,PUFFERVOLL);
12    STR20 = ARRAY [1..20] OF CHAR;
13    STR8  = ARRAY [1.. 8] OF CHAR;
14    (* IM PROZESS DEFINIERTE STRUKTUR DES COMMON-DATA-POOL *)
15    SATZTYPE = RECORD
16                  NAME :STR20;
17                  PLZ : 0:8999;
18                  ORT : STR20;
19                END;
20  VAR
21    PTCOMMONP2 : ^SATZTYPE ;
22
23    (* CDP-ZUGRIFFSVARIABLE *)
24
25    DATENSATZ  : SATZTYPE ;
26
27    (* SATZVARIABLE FUER DIE DATENAUFNAHME *)
28
29    SYSTEMSTATE : SYSSTATE ;
30
31    (* VARIABLE FUER DIE AUFAHME DES STATUS *)
32
33    KEYB,SCREEN,DRUCK : FILE OF CHAR;
34    MES : RECORD
35            CASE BYTE OF
36                 0 : (E:MESSAGETYPE);
37                 1 : (I:CSTATYPE)
38          END;
39    T:TIMETYPE;
40    P1,P2: 0..10;
41    INITPROCEDURE;
42     BEGIN
43      P1:=0;
44      P2:=1;
45     END;
46    PROCEDURE AUSGABEZEIT;
47     BEGIN
48      READTIME(T);
49      WRITELN(DRUCK,' DATUM :  ',T.DAY:3,'.',T.MONTH:3,'.',T.YEAR:3);
50      WRITELN(DRUCK,' ZEIT  :  ',T.HOUR:3,'.',T.MIN:3,'.',T.SEC:3);
51     END;
52   BEGIN
53
54    (* OEFFNEN DER DATEIEN (GERAETE) *)
55
56    REWRITE(SCREEN,0,T22);
57    RESET(KEYB,0,T22);
58    REWRITE(DRUCK,0,D3);
59    WRITELN(DRUCK,' AUSFUEHRUNGS-PROZESS AKTIVIERT');
60    AUSGABEZEIT;
61
```

```
62      (* VERBINDUNG ZUM COMMON-DATA-POOL AUFBAUEN UND GROESSE PRUEFEN *)
63
64      INITPOOL(PTCOMMONP2);
65      POOLCHECK(PTCOMMONP2^);
66
67
68      WRITELN(DRUCK,' FREIGABE DES ANFORDERUNGS-PROZESSES ');
69      AUSGABEZEIT;
70
71      (* FREIGABE DES ANFORDERUNGS-PROZESSES *)
72
73      MES.I:=FREIGABE;
74      SEND(P1,MES.E,WAITING);
75      IF SYSTEMSTATE <> OK
76      THEN
77      WRITELN(SCREEN,' FEHLER BEI FREIGABE ',ORD(SYSTEMSTATE))
78      ELSE
79       BEGIN
80
81         (* ZYKLISCHE UEBERNAHME DER DATEN UND AUSGABE AN DRUCK *)
82
83          REPEAT
84           WRITELN(DRUCK,' WARTEN AUF STATUS ''PUFFERVOLL'' ');
85           AUSGABEZEIT;
86
87           (* WARTEN AUF PROZESS P1 / PUFFERVOLL *)
88
89            REPEAT
90             AWAITMESSAGE(MES.E);
91            UNTIL MES.I = PUFFERVOLL;
92
93              (* UEBERNAHEM DER DATEN AUS DEM COMMON-DATA-POOL *)
94
95           GETPOOL(PTCOMMONP2^,DATENSATZ);
96           IF SYSTEMSTATE <> OK
97           THEN
98           WRITELN(DRUCK,' FEHLER BEI GETPOOL ', ORD(SYSTEMSTATE));
99
```

```
100
101          (* AUSGABE DER DATEN AUF DEN DRUCKER *)
102
103          WITH DATENSATZ DO
104           BEGIN
105            WRITELN(DRUCK,NAME);
106            WRITELN(DRUCK,PLZ,' ',ORT)
107           END;
108
109          (* NACHRICHT AN DEN ANFORDERUNGS-PROZESS *)
110
111          IF DATENSATZ.NAME[1] <> '%'
112          THEN
113           BEGIN
114            MES.I:=PUFFERVOLL;
115            SEND(P1,MES.E,WAITING)
116           END;
117         UNTIL DATENSATZ.NAME[1] ='%';
118          (* ENDE-ZEICHEN VON PROZESS P1 *)
119
120          (* ENDE DES PROZESSES *)
121
122        WRITELN(DRUCK,' ENDE DES PROGRAMMS FUER DEN ANFORDERUNGS-PROZESS');
123        AUSGABEZEIT;
124      END
125    END.
```

```
PROGRAM PBRTS3 ( KEYB,SCREEN );
(* ANFORDERUNGS-PROZESS *)
(* ERSTELLT AM 11.06.82 *)

IMPORTS SYSTEMSTATE;
CONST
  T21 = 21;
  D3  =  3;
VAR
  SYSTEMSTATE : SYSSTATE;
  (* VARAIBLE FUER DIE AUFNAHME DES STATUS *)
  KEYB,SCREEN : FILE OF CHAR;
  M : RECORD
        CASE BYTE OF
             0:(E:MESSAGETYPE);
             1:(NACHRICHT:(STOP,START))
      END;
  T:TIMETYPE;
  P1,P2 : 0..10;
  STARTMINUTE,STARTSEKUNDE,ENDMINUTE,ENDSEKUNDE : INTEGER;
  PXS,PXE : INTEGER;
  INTERVALL : INTEGER;
  WEITER:(N,J);
  STZ,ENDZ : DAYTIMETYPE;
  INITPROCEDURE;
   BEGIN
    P1:=0;
    P2:=1;
   END;
  PROCEDURE EINGABE_0_59 ( VAR WERT : INTEGER );
   BEGIN
     REPEAT
      WRITELN( SCREEN,' EINGABE WERTE ZWISCHEN 0 UND 59 ');
      WRITE(SCREEN,'*');
      READLN( KEYB,WERT)
     UNTIL ( WERT >=0 ) AND (WERT <= 59)
   END;

  PROCEDURE AUSGABEZEIT;
   BEGIN
    READTIME(T);
    WRITELN( SCREEN,' DATUM :  ',T.DAY:3,'.',T.MONTH:3,'.',T.YEAR:3);
    WRITELN( SCREEN,'ZEIT   :  ',T.HOUR:3,':',T.MIN:3,':',T.SEC:3)
   END;
 BEGIN
  REWRITE(SCREEN,0,T21);
  RESET(KEYB,0,T21);
  WRITELN( SCREEN,' ANFORDERUNGS-PROZESS *PBRT.3* AKTIVIERT');
  WRITELN( SCREEN,' HEUTIGES DATUM UND SYSTEMZEIT');
  WRITELN(SCREEN);
  WRITELN(SCREEN);
  AUSGABEZEIT;
   REPEAT (* BIS STARTZEIT > SYSTEMZEIT *)
    WRITELN( SCREEN ,' BITTE STARTZEIT EINGEBEN, MINUTE UND SEKUNDE');
    WRITELN( SCREEN ,' START-ZEIT MUSS GROESSER ALS DIE SYSTEMZEIT SEIN');
    WRITELN( SCREEN ,' DIE STUNDE WIRD VON DER SYSTEMZEIT BESTIMMT');
    WRITE(SCREEN ,' MINUTE : ');
    EINGABE_0_59 ( STARTMINUTE );
    WRITE(SCREEN,' SEKUNDE : ');
    EINGABE_0_59 ( STARTSEKUNDE )
   UNTIL STARTMINUTE > T.MIN;
```

```
  REPEAT  (* BIS ENDEZEIT > STARTZEIT + 5 SEKUNDEN *)
   WRITELN(SCREEN,' BITTE ENDEZEIT EINGEBEN, MINUTE UND SEKUNDE');
   WRITELN(SCREEN,' ENDEZEIT > STARTZEIT + 5 SEKUNDEN');
   WRITE(SCREEN,' MINUTE : ');
   EINGABE_0_59 ( ENDMINUTE );
   WRITE(SCREEN,' SEKUNDE : ');
   EINGABE_0_59 ( ENDSEKUNDE );
   PXS:=STARTMINUTE*60+STARTSEKUNDE;
   PXE:=ENDMINUTE*60+ENDSEKUNDE;
   IF PXE < PXS+5
   THEN
   WRITELN(SCREEN,' ENDE-ZEIT MUSS GROESSER GEWAEHLT WERDEN');
  UNTIL PXE > PXS+5;

  REPEAT  (* BIS DER EINGABEWERT ZWISCHEN 100 UND 3000 LIEGT *)
   WRITELN(SCREEN ,' BITTE AKTIVIERUNGS-FREQUENZ EINGEBEN IM  BEREICH ');
   WRITELN(SCREEN,' ZWISCHEN 100 MSEC UND  3000 MSEC ');
   READLN(KEYB,INTERVALL);
  UNTIL (INTERVALL >= 100) AND (INTERVALL <= 3000);

 STZ.HOUR:=T.HOUR;
 STZ.MIN:=STARTMINUTE;
 STZ.SEC:=STARTSEKUNDE;
 STZ.MSEC:=0;
 ENDZ.HOUR:=T.HOUR;
 ENDZ.MIN:=ENDMINUTE;
 ENDZ.SEC:=ENDSEKUNDE;
 ENDZ.MSEC:=0;

 M.NACHRICHT:=START;
 SCHEDULE(P2,M.E,WAITING,TRUE,STZ,INTERVALL,0);
 WRITELN(SCREEN,' ITERATIVER PROZESS AKTIVIERT');
 WRITELN(SCREEN);
 AUSGABEZEIT;
 M.NACHRICHT:=STOP;
 SCHEDULE(P1,M.E,ALWAYS,TRUE,ENDZ,0,1);

  REPEAT  (* BIS RICHTIGE NACHRICHT EINGETROFFEN *)
   AWAITMESSAGE (M.E);
  UNTIL M.NACHRICHT=STOP;

 CANCELSCHEDULE (P1,P2);

 WRITELN(SCREEN,' TERMINIERUNG DES ITERATIVEN PROZESSES ',P2:3);
 AUSGABEZEIT;
END.
```

```
 1 PROGRAM PBRTS4 (SCREEN);
 2 (* ITERATIVER PROZESS  *)
 3 (* ERSTELLT AM 14.6.82 *)
 4
 5 IMPORTS SYSTEMSTATE;
 6
 7 CONST
 8   T22 = 22;
 9   D3  = 3;
10   STERNCHEN = '*';
11
12 VAR
13   SYSTEMSTATE : SYSSTATE;
14   (* VARAIBLE FUER DIE AUFNAHME DES STATUS *)
15   SCREEN : FILE OF CHAR;
16   M:RECORD
17       CASE BYTE OF
18            0:  ( E:MESSAGETYPE);
19            1:  ( NACHRICHT :( STOP,START ))
20     END;
21   T:TIMETYPE;
22   RINGBELL:CHAR;
23
24   PROCEDURE AUSGABEZEIT;
25    BEGIN
26     READTIME(T);
27     WRITELN(SCREEN,' DATUM :  ',T.DAY:3,'.',T.MONTH:3,'.',T.YEAR:3);
28     WRITELN(SCREEN,' ZEIT  :  ',T.HOUR:3,':',T.MIN:3,':',T.SEC:3)
29    END;
30
31  BEGIN
32   REWRITE(SCREEN,0,T22);
33   WRITELN(SCREEN,' ITERATIVER PROZESS VORBEREITET');
34   AUSGABEZEIT;
35   RINGBELL:=CHR(7);
36    REPEAT (* OHNE ENDE-BEDINGUNG *)
37      REPEAT (* BIS DIE NACHRICHT ''START'' EINTRIFFT' *);
38       AWAITMESSAGE(M.E);
39       IF M.NACHRICHT <> START
40       THEN
41       WRITELN(SCREEN,' FALSCHE NACHRICHT');
42      UNTIL M.NACHRICHT = START;
43     WRITE(SCREEN,RINGBELL,STERNCHEN);
44    UNTIL FALSE (* DAUER-ZYKLUS *)
45  END.
```

```
 1 PROGRAM PBRTS5 (INPUT,OUTPUT);
 2 (* TEST DER EIN-AUSGAENGE FUER DIGITALE SIGNALE *)
 3
 4 IMPORTS SYSTEMSTATE;
 5
 6 VAR
 7   SYSTEMSTATE:SYSSTATE;
 8   RDIG,RDEZ,WDIG,WDEZ : BYTE2;
 9   ANTWORT : CHAR;
10
11  BEGIN
12    REPEAT (* BIS ENDE AM TERMINAL SIGNALISIERT WIRD *)
13     READWORD (0,RDIG);
14     WDIG:=RDIG;
15     WRITEWORD (0,WDIG);
16
17     READDECIMAL(1,RDEZ);
18     WDEZ:=RDEZ;
19     WRITEDECIMAL(1,WDEZ);
20
21     WRITELN(' DIGITALWERT :  ',RDIG:5);
22     WRITELN(' DEZIMALWERT :  ',RDEZ:5);
23
24     WRITE(' WEITER ? (J)  :  ');
25     READLN(ANTWORT);
26    UNTIL ANTWORT <> 'J'
27  END.
```

DIETZ COMPUTER-SYSTEME
Solinger Straße 9
Postfach 13 02 20
4330 Mülheim-Ruhr
Tel.: 0208/44 34-1
Telex: 856770

DIETZ Computer SYSTEME