stichting
mathematisch
centrum

MC

AFDELING INFORMATICA IW 75/77 FEBRUARI
(DEPARTMENT OF COMPUTER SCIENCE)

R.VAN VLIET & W.WAKKER

# MULTICORE 8
# A TIMESHARING SYSTEM FOR THE PDP8 SERIES

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O).

AMS/MOS subject classification scheme (1970): 68-00

ACM-Computing Reviews-categories: 4.32, 4.35

# Multicore 8
# A timesharing system for the PDP8 series.

by

R.van vliet & W.Wakker

## Abstract

In this document we present the first version of the M C 8 (MULTICORE 8) system.

MC8 is an operating system for the D.E.C. PDP-8 computer. It offers the possibilities of multiprogramming (paralel execution of tasks) and timesharing.

Most of the concepts used have already been implemented in other operating systems for the PDP-8. The main concept we added is that of a virtual memory: 256k virtual core in 16-32k actual core. Due to this, the system is particularly powerful for those who implement large programs or programs requiring large buffers. The responsibility of swapping is then automatically deferred to the MC8 monitor.

In developing the system objectives like modularity, well-defined datastructures and clearness of flow of control competed with the need of writing a compact and fast system for a small machine. Much attention was paid to a strict intertask and task/monitor communication.

KEYWORDS & PHRASES: Real time operating System,peripheral processor, multi programming

CONTENTS

## Conventions and abreviations

Throughout this document a number of conventions and abreviations is used. We list them below.

Conventions.

1. Arrays are indexed from Ø upward. An entry is denoted as NAME[I], where NAME indicates the array and I the index.
2. Valid pointers never point at ØØØØ. This value thus may be used as nil. It indicates the end of queues, free entries etc.
3. All names in this document that correspond to symbolic names in the listing of MC8 are denoted in capitals.
4. In the description of monitor commands and pseudo instructions changes in registers are indicated with a "becomes"-sign ( := ).

Abreviations.

| abr | meaning | section | description |
|---|---|---|---|
| ac | accumulator | | 12-bit processor register. |
| arg1 | argument 1 | | first argument of monitor command. |
| arg2 | argument 2 | | second argument of monitor command. |
| config | configuration | | |
| dfr | datafield register | | 3-bit processor register. points at the current datafield. |
| fcchain | free core chain | 1.8.3 | list of free core junks used by the page allocator. |
| fldtab | fieldtable | 1.4.3 | table holding information on virtual fields. |
| ifr | instructionfield register | | 3-bit processor register. points at the current instructionfield. |
| intreq | interrupt request | 1.5 | |
| intslot | interrupt slot | 1.5.2 | 5 consecutive locations in the skipchain corresponding to one peripheral device. |
| mq | multiplier quotient | | 12-bit processor register. |
| msptr | message pointer | 1.2 | pointer to word2 (first information word) of a message. |
| pc | program counter | | 12-bit processor register. points at the next program instruction. |

| | | | |
|---|---|---|---|
| stl | static task list | 1.1.2 | table holding the keys to all tasks in the system. |
| st# | static task number | 1.1.2 | index of task´s key in the stl, identifies task. |
| swreq | swap request | 1.4.6 | request to swap in a virtual field. |
| tcb | task controlblock | 1.1.3 | blocklet of core, holding information on tasks whose state is not initial. |
| tcbptr | task controlblock pointer | 1.1.3 | pointer to tcb[5]. |
| toq | timeoutqueue | 1.9.6 | |

## Ø. Introduction

When near the end of 1975 we started at the design of a text manipulation sytem for the D.E.C. PDP-8 computer (small but cheap) *1, we were faced with the problem of choosing an operating system to base our system on. We plan a system [1] to which about 8 terminals can be connected. These terminals are to be used to type in or edit a text, or to control a more complicated text manipulation program, such as a formatter, a neatener, an indexing program or the like, that works on an interactive base. So we needed an operating system able to execute several fairly large programs in paralel *2 and in which a growing library of utility tasks could be made available to each user.

The system that best met these requirements was MULTI8 [2]. Its main restrictions were:

A. Only a limited number of tasks (about 64) could be kept in its task library. Moreover, the support of each terminal in MULTI8 took 5 tasks.

B. The total amount of core in use by active tasks cannot exceed the actual coresize. it is nice to get around this restriction, as in fact tasks are executed one after another. The restriction may lead to cumbersome swapping of program overlays and buffers.

Remedying these restrictions seemed quite complicated. Therefore we chose the challenging solution of writing our own operating system. MULTICORE 8 (abreviated to MC8) is the result.

In developing MC8 great advantage was taken from our knowledge of the systems MULTI8 [2] and RTS8 [3]. A number of concepts was copied without change. Important new concepts are:

A. The possibility of adding and removing families of tasks at run time.
The MC8 system can handle only 128 tasks at a time, but these tasks may be drawn from a large task library on disk. In the current task library 21 families of upto 31 tasks each may be stored.

*1 For those not familiar with the PDP-8 the most important features are listed in [7].

*2 The PDP-8 cannot actually execute tasks in paralel. By paralel we mean that tasks are worked upon for a while, one after another, so that to the terminal user it seems to occur simultaneously.

B. The implementation of a virtual memory.
The total amount of core that may be accessed by active tasks is 256k. Each time a field (4k) of this core is actually accessed, the system swaps that field in one of the actual corefields.

C. The highly formalised intertask communication.
This enables tasks to wait for multiple conditions. I.e., a task may set itself to wait for several conditions at a time. It will be continued as soon as one of these conditions is fulfilled. For example, a task may send messages to other tasks and subsequently wait for the first message that is reported.

MC8 can be run on a PDP-8/I, PDP-8/E or PDP-8/A computer. It requires at least 16k core, a fast background storage and a clock. The coreresident portion of the system (called the monitor) occupies about 2k core memory allocated in field 0. The rest of this field is used by the availlist system. In other fields taskcode and buffers are allocated.

In appendix A a first version of the MC8 monitor is listed. It includes the interrupt section, the scheduler, the storage allocator, the virtual core manager, the monitor command section, the availlist system and a number of socalled monitor tasks, such as the diskdriver, the timer, the swaptask, the taskfetcher and the idletask.

Upon this groundlevel, the monitor of MC8, a higher level must be built containing a file system, a terminal section, device drivers etc. This document only deals with the MC8 monitor, which is to be a flexible base for more complicated systems.

The smallness of the PDP-8 is a serious constraint. To some extent this is reflected in a loss of modularity and locality. Sometimes this constraint forced the use of bad or slow algorithms in favour of better or faster ones that require more core (e.g., the swap strategy for the virtual core). The most eyestriking drawbacks of the system are the lack of intertask protection and the possibility of uncontrolled resource consumption.

Our first experience with the performance of the system is encouraging, as may be seen from the system bulletin in appendix D.

Chapter 1 discusses the important features of the MC8 system.
In chapter 2 a number of miscelaneous remarks will be made on the performance of the system and how to use it.
The monitor as a whole is listed in appendix A and some examples of tasks are given in appendices C and D.

# 1. The MC8 system

Chapter 1 is not intended to give a precise description of the MC8 system. Much more it is the intention to explain the algorithms used and give some insight in the ideas involved.

## 1.1 tasks

The basic unit in the MC8 system is a task. The monitor of MC8 constitutes an environment in which tasks may be run. It determines the order in which tasks are run, it provides means for intertask communication and for communication between a task and a peripheral, it hands buffers to tasks and performs a lot of other useful functions.

Tasks in the MC8 system belong to one of three categories:

Monitor tasks.

These tasks are effectively a part of the monitor. Their code is built in the monitor and closely interrelated with other parts of the system. They are discussed in sec. 1.9.

System tasks.

These tasks constitute the multiprogramming level of the MC8 system. The rest of this section is mainly concerned with system tasks. In general they are written and assembled as separate modules, knowing little about the monitor and in principle nothing about other tasks. Of course one could implement a family of tasks communicating via a well-defined interface.

Protect tasks.

These tasks constitute the MC8 timesharing level. They have no access to any other part of the system and run with the hardware usermode flipflop set. In this paper little is said on this category of tasks, as we did not decide yet on the exact way of implementing these tasks (see however sec. 2.3).

### 1.1.1 relocatable code

A task is a piece of executable code. When it is loaded into the system, a status is attached to it. The code of a task covers 1-31 consecutive pages of core *1. During execution of a task, the code is stored in consecutive pages of one memoryfield, never in page Ø of a field. Once the code has been stored in the virtual core, it is left there until the task itself specifies a SWAPOUT or EXIT command to

*1 for an exception see sec. 2.3.

the monitor.

At assembly time it is uncertain where in a memoryfield the code of a task will be stored during execution. This complicates off-page referencing *2.

The following relocation conventions are used (compare to MULTI8 [2]):

A The task is assembled as if its first page were stored in page 1 (locØ2ØØ-Ø377 oct) of a memoryfield.

B. Each page of code starts with a, possibly empty, sequence of (off-page) pointers. The sequence is terminated by a location pointing at ØØØØ. The latter cannot be confused with an actual pointer, because -as a consequence of A- ØØØØ does not belong to the taskcode.

C. Before actually storing the taskcode in core, the contents of these first locations of each page are updated by substracting Ø2ØØ oct and adding the actual first address of the code.

The PDP-8 assembler PAL8 [4] has an option to generate pointers at the end of a page. Writing " (name ", where an operand is allowed, generates such a pointer to the location "name". We created a special version of the assembler, in which pointers at the beginning of a page are generated by writing " )name ", where an operand is allowed [5].

### 1.1.2 The static task list (stl)

To each task that is loaded in the system an entry in the "static task list" (stl) is assigned. The index of this entry is called the "static tasknumber" (st#) and identifies the task in the system. The number of entries in the stl (i.e., the maximum number of tasks that may be loaded in the system at a given instant of time) is determined by the config parameter MAXSTL. MAXSTL must be in the range Ø<=MAXSTL<=127.

Free entries (=entries not corresponding to a task) in the stl are marked by denoting ØØØØ in their first word.

Each entry consists of 2 words:

| word | bit | meaning |
|---|---|---|
| Ø | | If<Ø pointer to task controlblock,<br>If=Ø free entry,<br>If>Ø blocknumber of taskcode on disk. |
| 1 | Ø-2 | Task´s priority. Ø<priority<=7. Used by the scheduler (see sec. 1.3). |

*2 On-page referencing does not cause troubles, as these references are coded relative to the page-offset.

| | | |
|---|---|---|
| 1 | 3 | Zerorequestbit. If set, that task needs the "one-task-only" part of page 0 of the field in which it is run (see sec. 1.7.2). |
| 1 | 5 | Coreresidentbit. If set, the task must run in a coreresident field (see sec. 1.4.5). |
| 1 | 7-11 | Length of taskcode in pages. |

The contents of word1 and the blocknumber of the taskcode on disk may be found in the directory of the task library.

Its entry in the stl is a key to all information on a task in the system. This may be seen from the following scheme of operation. When a task is loaded, first a free entry in the stl is searched, designating the st#. The entry is filled from the directory of the task library. In word0 the blocknumber of taskcode on disk is denoted. Blocknumbers must be in the range 1<=blocknumber<=2047, that is a positive integer in the PDP-8. From the fact that word0 is >0 the system concludes that the taskcode is still on disk and the status of the task is initial. The latter implies that the task is not runnable, but waiting for a message. As soon as a message is sent to the task, a task controlblock (tcb) is requested. In the tcb the status of the task will be kept. The system prefills the tcb with initial values (see sec. 1.1.3). A pointer to the tcb is stored in word0 of the entry of the stl. A tcb is a blocklet of core, allocated in field 0, the monitor field, at addresses >=2048. Consequently the contents of word0 is now negative. This is the common situation. Normal message-receive procedures can now be executed.

The use of a positive blocknumber in word0 must be viewed as pure optimisation. Instead of denoting the blocknumber in word0, we might have attached a tcb to the task immediately when it was loaded. The blocknumber is denoted in tcb[15]. Many tasks are loaded into the system just to be addressable for other tasks. They provide functions that are rarely used, such as error recovery. Attaching a tcb to such tasks would cost a lot of scarce core in field 0. A positive value in word0 is perfectly equivalent to the situation that a tcb holding initial values is attached to the task.

When a task has completed its function it executes the EXIT command. This indicates that the task returns to its initial situation. Its code is erased from core and its tcb is returned. The disk blocknumber is rewritten in word0 of the entry in the stl.

A task is unloaded as follows:

First the task is reset to the initial situation. The only rudiment of the task in the system is now its entry in the stl.

Next this entry is marked as free by clearing word0. The task looses its st#, which makes it unidentifiable. It vanished from the system.

### 1.1.3 The task controlblock (tcb)

A task controlblock (tcb) is a blocklet of 16 consecutive locations of core, allocated in field Ø. When the status of a task is not initial, a tcb is attached to it. During execution the status of a task is loaded into the hardware registers. It is restored into the tcb when the task is interrupted or waiting for some event.
The tcb also holds information for the claim mechanism, the intertask communication, the scheduler and the taskfetcher.
Layout of the tcb:

| word | bit | meaning |
|---|---|---|
| Ø | | Claim word.<br>Bit 1-11 of the tcbptr of the task that last sent a message to this task. Used to accept a sequence of messages from one task.<br>ØØØØ means: ready to accept a message from any task (see sec. 1.2.1). |
| 1 | | Head of receivequeue (see sec. 1.2). |
| 2 | | Messagepointer of waited report.<br>Used only when the task waits for a report.<br>ØØØØ indicates that any report will be accepted (see sec. 1.2). |
| 3 | | Head of reportqueue (see sec. 1.2). |
| 4 | Ø-3 | Waitbits. When set they indicate a condition for which the task is waiting. |
| | Ø | Wait for report. |
| | 1 | Wait for message. |
| | 2 | Wait for timeout. |
| | 3 | Wait for code swap in (see sec. 1.3.3). |
| 4 | 7-1Ø | Task´s priority (see sec. 1.3). |
| 4 | 11 | Stoppedbit (see sec. 1.3.2). |
| 5 | | Runnable task: link in runqueue (see sec. 1.3).<br>nonrunnable task: pointer to timeoutnode, if any (see sec. 1.9.6). |
| 6 | Ø | Ondiskbit. When set, taskcode is still on disk. |
| 6 | 1 | Set when task is reentrant. |
| 6 | 5-11 | Static tasknumber. |
| 7 | Ø-5 | Virtual instructionfield.<br>Number of the virtual memoryfield in which the taskcode is stored. |
| 7 | 6-11 | Virtual datafield (see sec. 1.4.2). |
| 8 | | Ac. |
| 9 | | Pc. |
| 1Ø | | Mq. |
| 11 | Ø | Link. |
| 11 | 1 | Greater than flag. |

| | | |
|---|---|---|
| 11 | 3 | Scheduling inhibitionflag.<br>If set, the task runs with the scheduler inhibited (see sec. 1.6.3). |
| 11 | 4 | EAE-mode. |
| 11 | 5 | Usermode flipflop.<br>Set only for protect tasks. |
| 11 | 7-11 | Stepcounter. |
| 12 | | BASE. |
| 13 | | X. |
| 14 | 0-4 | First page of taskcode. |
| 14 | 6-11 | Requested datafield (see sec. 1.4.2). |
| 15 | | Blocknumber on disk. |

Remarks.

Those bits for which no meaning was indicated are unused.

Word0 is used for the claimmechanism.

Word0-3 are involved in the intertask communication.

Word4-5 are used by the scheduler.

Word7 (also bit6-11!) and word14 bit0-4 are undefined when the taskcode is not yet swapped in (i.e., as long as word6 bit0 is set).

Word8-10 and word11 except bit2,bit3 and bit6 correspond to the hardware status. These registers are loaded into the hardware registers during the execution of the task. They are restored in the tcb when the task is interrupted or set to wait.

Word12 and word13 correspond to two locations in page 0 of the field in which the taskcode is stored. Their contents is loaded into these locations and restored in the tcb together with the loading and the restoring of the hardware registers. To the task it looks as if it has full access to these two locations in page 0.

As word0 of the entry in the stl points to the tcb as soon as a tcb is attached to the task, its previous contents must be saved in the tcb. Therefore word15 holds this contents.

Each time a tcb is attached to a task, it is prefilled with initial values. These values are:

| word | bit | initial setting (octal) |
|---|---|---|
| 0 | | 0000 |
| 1 | | 0000 |
| 2 | | 0000 |
| 3 | | 0000 |
| 4 | 0 | 0 |
| 4 | 1 | 1 |
| 4 | 2 | 0 |
| 4 | 3 | 0 |
| 4 | 7-10 | Priority |
| 4 | 11 | 0 |

| | | |
|---|---|---|
| 5 | | Innocent pointer. |
| 6 | Ø | 1 |
| 6 | 1 | Ø |
| 6 | 5-11 | Static tasknumber |
| 7 | Ø-5 | Undefined |
| 7 | 6-11 | Undefined |
| 8 | | Undefined |
| 9 | | Undefined |
| 1Ø | | Undefined |
| 11 | Ø | Undefined |
| 11 | 1 | Undefined |
| 11 | 3 | Ø |
| 11 | 4 | Undefined |
| 11 | 5 | Ø |
| 11 | 7-11 | Undefined |
| 12 | | Undefined |
| 13 | | Undefined |
| 14 | Ø-4 | Undefined |
| 14 | 6-11 | Undefined |
| 15 | | Blocknumber on disk. |

### 1.2 The intertask communication

The communication between tasks and their mutual synchronisation is based on the use of "messages". One task sends a message to an other task and after a while the receiver "reports the message" (back to the sender). A message is a blocklet of 5 consecutive locations, allocated in field Ø. Messages are requested from and returned to an availlist system. About 15Ø messages may be requested. This is rather few compared to the 128 tasks that may be present in the system. Tasks must carefully use messages and return them as soon as possible.

The limited number of messages is one of the worst aspects of the MC8 system. We rejected two alternatives.
One alternative is [2] to specify that tasks "call" other tasks. At the moment of the call parameters are passed from the caller to the callee. A disadvantage is, that the call "fails" when the callee is still active. Upon failure the caller will probably wait for a while and try again. This highly increases the danger of deadlocks. Moreover, it provides a rather poor scheme of communication.
The other alternative is to store the messages in the code of the sendertask [3]. Tasks that want to send messages will never be confronted with a lack of messages, because it is their own responsibility to allocate them. Our difficulty is, that the sender and the receiver probably reside in different fields of the virtual

core. Multiple fieldswaps will occur when the receiver inspects and possibly changes the contents of the message. Field Ø is always present in the actual core, so that messages may be stored there without this objection.

Layout of a message:

| word | bit | meaning |
|---|---|---|
| Ø | Ø | Ø: message need not be reported.<br>1: message need be reported. |
| Ø | 1-11 | Bit1-11 of tcbptr of the task that sent the message. |
| 1 | | Link in receive- or reportqueue.<br>ØØØØ indicates that the message has no successor. |
| 2-4 | | User-defined information. |

The size of a message (3 words of user-defined information) is sufficient for most common communication. For instance buffer transfers may be specified as:

| word | bit | meaning |
|---|---|---|
| 2 | Ø | Read or write |
| 2 | 1-5 | Length in pages (max 4k) |
| 2 | 6-11 | Virtual field of the buffer |
| 3 | | Starting address of the buffer |
| 4 | | Other information. |

Scheme of communication.
Assume task1 wants to set up communication with task2. Task1 requests a message, using the MSREQ pseudo instruction (see sec. 1.7.4). It stores the information in the message and executes a monitor command to send the message to task2. The monitor denotes the tcbptr of task1 in the message and either attaches the message to the receivequeue of task2, or directly activates task2. In either case task2 eventually "receives" the message and runs for a while. Meanwhile task1 may or may not continue to run, but sooner or later it would like to receive a report from task2. When task2 is done, it denotes its completion status in the message and executes a monitor command to report the message back to the sender: task1. The monitor attaches the message to the reportqueue of task1 or, if task1 was waiting for this report, directly activates task1. Finally task1 no longer uses the message. It returns it to the availlist system by executing the MSFREE pseudo instruction (see sec. 1.7.4).

Messages are identified in the system by specifying a pointer to their third word. This pointer is called the "message pointer" (msptr), and, as all messages are allocated in field Ø, this pointer indeed identifies a message. Tasks also use this pointer to access the information in the message.

The monitor commands that have to do with message exchange are discussed below.

SNDMS
Send a message to a task.
Arg1: st# of receivertask.
Arg2: msptr of message to be sent.
Options: NONREP, DFPARM.

A. Bit 1-11 of the tcbptr of the sender is denoted in the message.
B. BitØ of wordØ of the message is set, unless the NONREP (nonreport) option was specified.
C. If the receiver is waiting for the message, then
C1. the message is "received" and the receiver is runnable again else
C2. the message is attached to the receivequeue of the receiver.

If the DFPARM option was specified, the actuaal datafield of the sendertask is copied into bit6-8 of word2 (the first informationword) of the message.

WTRP
Wait for a report.
Arg1: msptr of waited report.
Options: TIMEOUT, SWAPOUT.

First the reportqueue is checked to see if the message specified was already reported.
If so, the message is deleted from the reportqueue and its msptr is denoted in the task´s ac (the task is not actually set to wait!).
If not, the msptr is copied into tcb[2] and the task is set to wait with tcb[4] bitØ set.
If the argument =ØØØØ, any message reported to the task will continue or reactivate it.
For the meaning of the options see below.

SNDWTR
Send a message and wait for its report.
Arg1: st# of receivertask.
Arg2: msptr of message to be sent.
Options: NONREP, DFPARM, TIMEOUT, SWAPOUT.

This is just an optimisation. The task might as well have executed the SNDMS and WTRP commands in this order, specifying the same msptr.

RP
report a message.
Arg1: msptr of reported message.

A. If bitØ of wordØ of the message =Ø, then the message is returned to the availlist system (nonreport option).

B. Else the tcbptr of the sender was denoted in the message.

C. If the sender was waiting for this report, the msptr is stored in its ac and the sender is reactivated (runnable again). Else the message is attached to the reportqueue of the sender.

WTMS
Wait for a message.
Options: KEEP, TIMEOUT, SWAPOUT.

A. If the KEEP option is not specified, the claim word of the task (tcb[Ø]) is cleared.

B. The receivequeue is inspected to see if an appropriate message is present.
A message is appropriate if the tcbptr of its sender (message[Ø] bit1-11) agrees with the claim word of the task. If nothing is indicated in the claim word (tcb[Ø]=ØØØØ), any message is appropriate.

C. If an appropriate message is present, then

C1. this message is deleted from the receivequeue and the task "receives" the message (the task is not actually set to wait!)
else

C2. The task is set to wait with tcb[4] bit1 set.

For the meaning of the TIMEOUT and SWAPOUT options see below.

CHKRPQ
Check reportqueue.
Arg1: msptr of waited report.

This command is almost identical to WTRP. However, the task is not actually set to wait. If no appropriate report is found in the queue, the task is continued with ac=Ø. (Remember, that ac held the msptr if a report could be found.)

CHKRCQ
Check receivequeue.
Options: KEEP.

This command is almost identical to WTMS. However, the task is not actually set to wait. If no appropriate message can be found in the queue, the task is continued with ac=Ø. (Remember, that ac held the msptr if an appropriate message was found.)

Remarks.

"receiving" a message is accomplished in two steps:

First the tcbptr of the sender of the message is copied into tcb[Ø] of the receivertask. The task is now claimed by the task that sent the message, and will remain so until it executes a WTMS command without specifying the KEEP option.

Next the msptr is stored in the task´s ac.

The DFPARM option is a very ugly feature. It can be used to pass a fieldnumber in the message. As we pass the actual fieldnumber, this number might change before the receiveertask actually uses this number. This is prevented by locking the field whose number is passed (see sec. 1.4.4).

The option was added to facilitate communication with the system diskdriver (see sec. 1.9.2).

In wordØ of a message and in tcb[Ø] bit1-11 of the tcbptr are used to identify the sendertask. This is indeed allowed, as by convention tcbptr´s are always negative integers. Hence bitØ of a tcbptr is always set, and bit1-11 are sufficient to identify the tcb.

When a task sets itself to wait, it may always specify the TIMEOUT and/or the SWAPOUT option.

Specifying the TIMEOUT option (see sec. 1.9.6) guarantees that the task will be reactivated after some delay, even if the other waitcondition(s) (e.g., wait for report) are not fulfilled. The task may detect this and take appropriate action.

If a task expects that it will be waiting for quite a long time (e.g., several seconds) and if no important variables are stored in its code, it may specify the SWAPOUT option. If the task is actually set to wait, its code is erased from core. As soon as the task is runnable again, a fresh copy of its code is swapped in from disk. The tcb of the task is kept however, so that its registers are preserved. See also sec. 1.9.4.

### 1.2.1 Claiming tasks

In MC8 several processes may run simultaneously. A "proces" is an intuitive concept, which we are not going to define here. In this section we shall describe the way in whicch tasks are claimed in MC8.

When someone wants a job to be done under control of the MC8 system, a task is started that executes the job (termed maintask in this section). In simple cases this task just does what is needed without communicating with other tasks and finally executes the EXIT command.

A more complicated situation arises when the maintask wishes to delegate part of the work to other tasks (termed subtasks in this section). The maintask will send messages to subtasks in order to set them to work. If the maintask needs only one message to specify to the subtask what it wants to be done, no problems arise. If however a sequence of messages is sent to a given subtask, messages from other tasks must be prevented from interspersing in this sequence. We say that the maintask must "claim" the subtask.

In fact a task is claimed each time it receives a message. Usually this claim terminates when the next WTMS command is executed, thus being invisible to the task. When the KEEP option is specified together with the WTMS command, the claim is not terminated and hence the subtask will only accept messages from the task that sent the former message.

Claiming is accomplished by writing bit1-11 of the tcbptr of the sender into tcb[0] of the receiver. When a claimable subtask is started, it starts processing messages, specifying the KEEP option at each WTMS command. When it recognises the last message of a sequence, it omits the KEEP option from the next WTMS command, thus enabling other tasks to send messages to it.

## 1.3 The scheduler

Tasks in the MC8 system either are runnable or are waiting for some or several conditions to be fulfilled. Each time the scheduler is activated, it selects one of the runnable tasks and executes the runchecks (see sec. 1.3.3). If one of this checks fails another task is selected. Else the status of the task is loaded into the hardware registers and the execution of its code is started up.

To each task a priority in the range 0<=priority<=10 (octal) is attached. This priority is determined at assembly time. The only task of priority 10 is the idletask. System tasks and protect tasks have priorities in the range 1<=priority<=7. Priority 0 is preserved for some monitor tasks.

The use of these priorities is opposite to the common interpretation, that is, tasks of lower priority are run first.

Runnable tasks are organised in priority runqueues. Each priority has its own runqueue. The runqueues are threaded through the task controlblocks (tcb[5]) of the runnable tasks.

When the scheduler is activated, it selects the first task of the priority 0 runqueue. If this queue is empty (no tasks of priority 0 are runnable), it selects the first task of the priority 1 runqueue. If that queue is empty too, the priority 2 runqueue is inspected etc. If all other runqueues are empty, the idletask, the first and only task in the priority 10 runqueue, is selected. The idletask is never set to wait and accesses only field 0, so that the runchecks cannot fail. This guarantees that the scheduler can always select and start a runnable task.

Waitconditions expire as a consequence of a hardware interrupt (tasks waiting for some deviceflag) or as a consequence of the execution of a monitor command (tasks waiting for a message from an other task) *1. So during the execution of monitor commands or while treating hardware interrupts, tasks may be added to a runqueue. Therefore the scheduler is activated at the end of the interrupt section and after the execution of a monitor command. This insures that, if tasks having a lower priority than the task currently selected have become runnable, these tasks are selected first.

Some monitor commands cause the task that executes them (i.e., the task currently running) to be set to wait. The activation of the scheduler after such a command will select the next important task.

### 1.3.1 The routines OURUNQ and INRUNQ

Tasks are set to wait by calling the routine OURUNQ. This routine takes a pointer to the tcb of the task and one or more waitconditions or the stoppedcondition as parameter. It removes the task from its priority runqueue and sets up the apropriate bits in tcb[4].

As soon as a waitcondition for some task expires, an attempt is made to reinsert it in its priority runqueue. This is done by calling INRUNQ with a pointer to the tcb of the task as parameter. INRUNQ first clears tcb[4] bit0-3, erasing all waitconditions. But the stoppedcondition, if present, remains set. If the stoppedbit was not set, the task is now reinserted in its priority runqueue. Otherwise no further action is taken.

To optimise scheduling an integer, HPRIO, and a boolean SCDREQ are used. HPRIO generally holds the priority of the task currently selected. SCDREQ is usually false, indicating that no schedule request is pending. When INRUNQ inserts a task in its runqueue, it compares the priority of the task to HPRIO. If this priority is higher or equal than HPRIO, subsequent scheduling is obsolate as the current task will again be selected. If this priority is lower, subsequent scheduling is needed. SCDREQ is set to true and the new lowest priority is stored in HPRIO.

Before entering the scheduler, the boolean SCDREQ is tested. If it is false, no scheduling is needed and the execution of the current task is continued. If SCDREQ is true, it is reset to false and the scheduler reinspects the runqueues starting at the one indicated by

*1 For exceptions see sec. 1.9.4 and 1.9.6.

HPRIO. Each time an empty queue is found HPRIO is incremented.
Note: If a monitor command sets the current task to wait, scheduling is required independant of the value of SCDREQ. In that case the scheduler is activated without testing SCDREQ.

### 1.3.2 The monitor commands STOP and RESUME

Generally speaking it is undesirable that tasks block and unblock each others execution. A proper synchronisation of tasks is based on message exchange. To cope with casualties the STOP command is introduced.
Taskl blocks the execution of task2 by executing the STOP command with proper argument. The following occurs:
If task2 was runnable (no waitbits nor the stoppedbit set in tcb[4]), OURUNQ is called specifying the stopppedcondition.
If task2 was already set to wait, the stoppedbit is turned on in tcb[4].
As long as the stoppedbit is set in tcb[4], task2 won't be reinserted in its runqueue (see above), though one of the waitconditions may expire. So the execution of task2 is effectively blocked.
To remove the stoppedbit from tcb[4] some task must execute the RESUME command specifying task2. The monitor removes the stoppedbit from tcb[4], erasing the stoppedcondition. If no waitbits are set in tcb[4], INRUNQ is called to reinsert task2 in its runqueue. If the stoppedbit was not set at all, the RESUME command has no effect.
Note: a siide effect of stopping a task is, that it is removed out of the timeoutqueue (see sec. 1.9.6) if it was in. So any timeout is terminated when the task is resumed.

### 1.3.3 The runchecks.

Before the scheduler can start up the execution of the runnable task it has selected, two additional checks, the runchecks, must be made. These are discussed below.

1. Runcheckl: the taskcode must be in core.
If this check fails, the task is set to wait with tcb[4] bit3 (wait for code swap in) set. A message is sent to the taskfetcher specifying that this task must be swapped in.
As the task is now removed from the runqueue, subsequent scheduling will select some other task. Eventually the taskfetcher will swap in the code and make the task runnable again.

2. Runcheck2: the fields accessed by the task must be in the actual core.
If this check fails, the scheduler issues a swaprequest for the field, if possible (see sec. 1.4.6). Then it selects the next important task as if the previous one was not in the runqueue. Special care is taken for reentrant tasks (see sec. 2.2). Eventually the field will be swapped in the actual core and the scheduler will then select and start up the task that was skipped over (see sec. 1.4.6).

## 1.4 The virtual core

In the MC8 system a virtual core of upto 256k words (=64 fields of 4k words) is implemented in an actual core of 16-32k words (=4-8 fields). The number of virtual fields is set by the config parameter VIRMAX, Ø<=VIRMAX<=63. The number of actual fields is indicated in the config parameter ACTMAX.

To each virtual field an entry in the "fieldtable" (fldtab) corresponds. From the contents of the fldtab it can be determined wether a virtual field is in the actual core and, if so, in which actual field it is. To each actual field an entry in the coremap corresponds. Which virtual field is in a given actual field is denoted in the coremap. An actual field always contains some virtual field.

To each virtual field an area on the system disk, a "fieldslot", corresponds. When the virtual field is not in the actual core, its contents is stored there.

### 1.4.1 Paging algorithm

When a task wants to access a virtual field that is not in the actual core, a "swaprequest" may be done. That is, a request is done to swap the field under concern into one of the actual corefields. Before this can be accomplished the system must decide which of the actual fields must be evacuated. The algorithm that makes this decision is termed the paging algorithm.

Many paging algorithms have been published in literature [6]. As the PDP-8 has no special hardware for the implementation of a virtual core, most of these algorithms are rather time consuming or hard to implement on a PDP-8. We chose a very plain one.

The decision which virtual field is to be swapped out is made by the routine VFLESS. Each time it is called it either fails or succeeds. If it succeeds it delivers an integer indicating which actual field is to be evacuated. VFLESS uses a global integer,

VFNEXT, that wraps around in the range Ø<=VFNEXT<=ACTMAX. VFNEXT is unchanged between two calls. The procedure LOCKED checks wether the virtual field, that is stored in the actual field indicated by VFNEXT, is "locked". If so, that virtual field must remain in core (see sec. 1.4.4). VFLESS fails if all fields are locked.
This is the algorithm stated in ALGOL 68:

```
´INT´ ACTMAX =#highest actual field#,
[Ø:ACTMAX] ´INT´ COREMAP :=#some initial setting#,
´INT´ VFNEXT:=Ø,
´PROC´ LOCKED =(´INT´ I)´BOOL´:
    #virtual field stored in actual field I is locked#;
´PROC´ VFLESS =(´REF´ ´INT´ ACTFLD)´BOOL´:
    (´INT´ I:=-1, ´BOOL´ FOUND :=´FALSE´;
     ´WHILE´ (I+:=1) <=ACTMAX ´AND´ ´NOT´ FOUND
     ´DO´ VFNEXT:=(VFNEXT+1) ´MOD´ (ACTMAX+1);
          ´IF´ FOUND:=´NOT´ LOCKED(VFNEXT)
          ´THEN´ ACTFLD:=VFNEXT
          ´FI´
     ´OD´;
    FOUND)
```

This algorithm seems quite reasonable. It guarantees that the virtual fields most used are in the actual core most of the time. A disadvantage is, that if a task accesses more fields, say FA and FB, then FA is perhaps swapped out in order to swap in FB, which is useless. This objection can easily be removed at the cost of some core in field Ø.

### 1.4.2 Datafield and instructionfiled

Programs in the PDP-8 have access to two memoryfields: the datafield and the instructionfield. These fields are accessed via two 3-bit registers, the datafieldregister (dfr) and the instructionfieldregister (ifr). The instructions of a program are always fetched from the instructionfield. Special instructions are provided to change the contents of these registers. CDF changes the contents of the dfr and CIF changes the contents of the ifr.

Just like normal programs tasks have access to two fields: the virtual datafield and the virtual instructionfield. Before the execution of a task is started or resumed, the scheduler loads the dfr and ifr with the actual fields in which the corresponding virtual fields reside. If this is impossible because one of these fields is not in the actual core, runcheck2 fails and appropriate action is taken (see sec. 1.3.3). Changing the virtual fields that a task may access is slightly more complicated as it is for ordinary programs.

When designing MC8 we made the decision, that the code of a task should fit in (and is always stored in) one memoryfield (see sec. 1.1.1). This implies that the instructions of a task are fetched from the same virtual field as long as its code is stored in the virtual core. Changing the virtual, and hence the actual instructionfield is thus unnecessary and forbidden.

Changing the virtual datafield may be useful, for instance to access buffers in fields different from the virtual instructionfield. In fact, experience shows that large programs and programs requiring large buffers fairly often use the CDF instruction. Before a task can change the contents of the dfr, it has to find out in which actual field the desired virtual field resides, if it even is in the actual core. This means that the task has to inspect system tables, which is unadvisable. Moreover, the task might be interrupted halfway its investigations and the system tables might be changed by the time it is resumed.

The simplest alternative is to introduce a monitor command to change the datafield of a task and let the monitor do what is needed. Activating the full mechanism of a monitor command each time a task wants to change its datafield, would however be to inefficient. To obtain an easy to use and efficient implementation we introduced 3 pseudo instructions and 1 related monitor command. The table below gives the time estimates for each pseudo instruction compared to the time that the execution of the actual CDF instruction takes.

| pseudo ins | time estimate |
|---|---|
| CDFCUR | 5 |
| VRCDF | 6 |
| VCDF | 28 |

The effect of the pseudo instructions.

CDFCUR Sets the datafield equal to the instructionfield.

VCDF Takes the contents of the next location as an argument, holding the number of the desired virtual field. If this field is in the actual core, the dfr is set to the corresponding actual field. Otherwise runcheck2 fails. An emergency escape (see sec. 1.7.7) is then used and similar action as described in sec. 1.3.3 is taken.

VRCDF Sets the datafield to the "requested" (see below) virtual field. Each first time this pseudo instruction is encountered after start or restart of the task´s execution, a check similar to that described for VCDF is made. If it succeeds one extra side effect occurs: the routine implementing this pseudo instruction is patched with a "hard" CDF instruction to the correct actual field. Subsequent calls on this routine will cause the correct CDF instruction to be executed immediately without any check. The patch is removed when the task is interrupted or executes a monitor command. This makes the VRCDF pseudo instruction extremely powerful in cases where a burst of CDF´s to the same virtual field occurs.

A virtual field is "requested" by executing the monitor command REQCDF. The desired virtual field is indicated in the rightmost 6 bits of the ac. It is denoted in tcb[14] bit6-11.
None of the above pseudo instructions touches the ac, link etc. (neither does the actual CDF instruction).
A special remark need be made for changing the datafield to virtual field 0. This can be achieved by executing a "hard" CDF-0 instruction, as virtual field 0 is always present in actual field 0. Tasks wishing to access information stored in messages may benefit from this.

Another related pseudo instruction need be mentioned here. The VRDF pseudo instruction clears the ac and stores the number of the current virtual datafield in bit6-11 of the ac. It is more or less equivalent to the hardware RDF instruction.

### 1.4.3 Fieldtable and coremap

Information on each virtual field is stored in the fieldtable (fldtab). Each field has a 1-word entry in this table:

| bit | meaning |
|---|---|
| 0 | 1: Field in actual core.<br>0: Field on disk. |
| 1 | Set if the field is in the actual core. |
| 2 | 1: Whole field used for datastorage.<br>0: Otherwise. |
| 3 | 1: One-task-only part of this field is used (see sec. 1.7.2).<br>0: Otherwise. |
| 4 | Set if field in the actual core. |
| 5 | Coreresidentbit (see sec. 1.4.5). |
| 6-8 | Indicate the actual field in which the field is stored. Undefined if field on disk. |
| 9 | Moninbit. Set if and only if a copy of the general page 0 is present in the field (see sec. 1.7.8). |
| 10 | Datapresentbit. Used to optimise swapping (see sec. 1.9.2). |
| 11 | 1: Field is in use (code or data may be stored in it).<br>0: Field is free. |

To some extent the coremap can be viewed as a cross reference table for fldtab. Each actual field has a 1-word entry in the coremap:

| bit | meaning |
|---|---|
| 1-5 | Lockcount. Counts to 31 locks (see sec. 1.4.4). |
| 6-11 | Virtual field that currently resides in this actual field. |

### 1.4.4 Locking and unlocking fields

Sometimes it is desirable (or even necessary) to insure that a virtual field remains present in the actual core for a shorter or longer period. Asserting this is termed "locking" a field. The opposite is termed "unlocking". To prevent fields from being swapped out of the actual core the procedure LOCKED in the paging algorithm (see sec. 1.4.1) checks the lockcount (see sec. 1.4.3) of the field that is about to be swapped out. If the lockcount is nonzero, this field will not be swapped out and an other field will be tried.

We list a few "lockconditions" (reasons to lock a field in core):

A. Tasks are allocated in the field, that are activated so often that it is preferable to have them permanently in core.

B. Field Ø, containing the coreresident portion of the system, must remain in core.

C. I/O processes may access a field so often that it is undesirable (or, in case of real time restrictions, impossible) to swap it out.

D. Swapping out a field while a databreakproces accesses it, causes serious errors.

E. Fields containing routines connected to an interruptslot must remain in core (see sec. 1.5.4).

Lockconditions may overlap one another. Therefore the lockconditions that apply to a given field are counted in the lockcount. When a lockcondition is initiated, the field is locked, which is equivalent to incrementing its lockcount. When a lockcondition terminates, the field is unlocked, which is equivalent to decrementing its lockcount. The field can be swapped out again, as soon as its lockcount is reset to Ø.

Locking for conditions A and B is automatically performed by the monitor. Locking for other conditions must be treated by the tasks that give rise to these conditions.

The monitor command LOCK locks the datafield of the task that executes the command in core. Note: that this field is indeed in the actual core, as otherwise runcheck2 would have failed, and hence the task would not have been executed.

The monitor command UNLOCK unlocks the datafield of the task that executes the command.

### 1.4.5 GETFLD and FRFLD

Initially field Ø is the only field in use by the system. All other fields are free. Field Ø holds the coreresident portion of the system and is locked in actual field Ø. As the space occupied by the system will never be considered as free, field Ø will never be "returned" and hence remain in the actual core as long as MC8 is running.

Sooner or later tasks and buffers must be allocated in the virtual core. If this cannot be done in one of the fields that is already in use, GETFLD is called to select one of the free fields. The selected field is no longer free; bit11 of its entry in fldtab is set. When calling GETFLD the coreresident condition may be specified. If so, the coreresidentbit (bit5) will be set in the entry in fldtab. As soon as the field is actually swapped in, the coreresidentbit is checked and if it is set the field is locked.

When a buffer is returned or when a task executes the EXIT command or specifies the SWAPOUT option, the space occupied by such a buffer or by the taskcode is considered as free again. If in this way a whole virtual field is free again, it is "returned". That is, FRFLD is called with the corresponding virtual field number as parameter. If the coreresidentbit of the field was set, the field is first unlocked. Next bits2, 3, 5, 1Ø and 11 of the entry in fldtab are cleared, marking the field as free.

### 1.4.6 Swapping

The generation and treatment of swaprequests (swreqs) is closely interrelated with the operation of the scheduler. On one hand the scheduler, on failure of runcheck2, can issue a swreq. On the other hand the way in which swreqs are treated heavily depends on the priority mechanism embedded in the scheduler. The order in which fields of the virtual core are swapped in the actual core severely influences the performance of the system.

Each task can access at most two fields at a time: the instructionfield and the datafield. Both these fields must be in the actual core when the task is executed. Here follows the easy way to achieve this.

Simple field swap scheme.

Assume some task, say T1, wants to access a field that is not in the actual core. The scheduler will detect this and issue a field swap. Next it waits until the field swap is completed (about 0.15 seconds) and then starts up the execution of T1.

This scheme has the disadvantage that the system wastes time during field swaps. Probably other tasks could have been executed that access fields that do reside in the actual core.

More concise scheme of field swaps.

The next scheme removes the disadvantage mentioned above, but it introduces great danger for deadlocks.

When the scheduler selects a task whose fields are not in the actual core, it appends a swreq to a queue. Next it continues inspecting runqueues, adding swreqs to the queue as required. Finally it hits on a task that can be executed. This certainly happens as the idletask accesses only field 0, which is locked in core.

Paralel to this proces of task selection and task execution an other proces, the swapper, eats nodes from the swreq queue and performs the indicated field swaps. Each time the swapper completes a field swap, it interrupts the task currently in execution and sets the schedulerequestflag. The scheduler now reinspects all runqueues and probably starts up the task that caused the field swap just completed.

The example below demonstrates the danger for deadlocks.

Assume there are 4 virtual fields: VF0, VF1, VF2 and VF3. The actual core has 3 fields initially containing VF0 (locked), VF2 and VF3. Apart from the idletask there are two runnable tasks T1, accessing VF1 and VF3, and T2, accessing VF2 and VF3. In the sequel we indicate a swreq by first denoting the field to be swapped out and next the field to be swapped in. When the scheduler is activated, the following might occur.

A. Select T1.
B. Swreq VF2-VF1 to queue.
C. Select T2.
D. Swreq VF1-VF2 to queue.
E. Select idletask.

Note: Of course tasks are not allowed to access fields that are halfway being swapped in or out. As soon as the first field swap is completed, T1 will be reselected. Although its fields are now in core, it cannot be started up, as the next field swap (which is about to be performed by the swapper) will swap out VF1 again. Hence step A is retaken and so forth.

This is the very reason why organising swreqs in a queue is a bad principle. Things go better when we restrict the length of the queue to 1. This is effectively done in the MC8 system.

Field swap scheme of MC8.
A. The scheduler selects a runnable task.
B. If both its fields are in the actual core, the task is executed, otherwise step C is taken.
C. If a swreq is still pending then goto step E.
D. Issue a swreq, eventually starting the swapper.
E. Schedule the next important task and goto B.
When the field swap is completed, all runqueues are reinspected, effectively retaking step A. Probably the task that caused the swreq, say T1, will now be executed. It can however occur, that a task, say T2, of a lower priority than T1 has become runnable. T2 is more important than T1 and thus will be selected first. If its fields are not in the actual core, the sceduler might issue a swreq such, that the fields of T1 are swapped out again. This tedious event can happen repeatedly, but ultimately the task that caused the swreq must have priority Ø. This task will certainly be executed at the completion of the swreq.
Summarising it turns out, that in this scheme field swaps may be done in a very unlucky order, but tasks of a lower priority will always get their jobs done. This is reasonable in a priority scheme.

Detailed description of the MC8 field swap scheme.

At the selection of a task or at the execution of one of the change-datafield pseudo instructions (step A) the routine FCHEX is called to check wether the desired field is in the actual core (step B). If it is, the execution of the task is started up or continued respectively. Otherwise if no swreq is pending (step C) a swreq is issued (step D). This is achieved by sending a special message, the swapmessage, to the system diskdriver. In this message the field to be swapped out (determined by calling VFLESS) and the field to be swapped in (the desired field) are specified. Moreover, the "ondisk"condition of the field to be swapped out is set, to prevent subsequent access from other tasks, and the swreqpendflag is set.

Now the scheduler can continue operating. Sooner or later (perhaps immediately) the system diskdriver will be scheduled and accept the swapmessage. This certainly occurs as the system diskdriver has priority Ø and never waits for other tasks. It just accepts a message, executes the indicated transfer and waits for the next message.
When the system diskdriver has accepted the swapmessage, it exchanges the indicated virtual fields and reports the swapmessage. As wordØ of the swapmessage always holds the tcbptr of the swaptask, the message is now reported to this task.

The swaptask is a nonterminating loop. It waits for the report on the swapmessage, clears the "ondisk"condition of the field just swapped in, updates its general page Ø (see sec. 1.7.8), clears the swreqpendflag and sets itself to wait again. Note: no swreq can be done until the swaptask has set itself to wait.

As the swaptask has priority Ø, all runqueues will be reinspected when it executes the WTRP command.

Remark.

If VFLESS fails no swreq is issued. This means, that, if all fields are locked, no swreqs are done. The idletask will probably be started up. Sooner or later a proces locking a field will terminate. At that moment the runqueues are automatically reinspected from that priority downward. If this causes a swreq, everything is o.k., however if it does not the system may remain in high priorities unless some external event makes a priority Ø task runnable. This will probably be the system timer started by the clock.

Multiple fieldswaps will occur if all fields but 1 are locked and neither the instructionfield nor the datafield of a selected task is in the actual core. This is the bad situation in which the system keeps exchanging data- and instructionfield until one of the other fields is unlocked. Therefore it is advisable to have at least 4 actual fields. Field Ø is always locked. Coreresident tasks can be stored in field 1, which will then remain locked for quite a long time. The remaining two actual fields are available for the normal virtual core mechanism.

## 1.5 The interrupt section

The interrupt mechanism of PDP-8 computers is rather poor. Certain external events (a flag raised by a peripheral device interface) or internal events (executing I/O instructions with the usermode flipflop set) can cause an interruptrequest (intreq). To each event a flag is connected. These flags are termed deviceflags and userflag respectively. (In the sequel we will mention only deviceflags, where both deviceflags and userflag may be meant.) Causing an intreq is always accompanied by setting a deviceflag.

Some devices can be "disabled". When a flag of such a device is raised, it does not cause an intreq. Devices that are not disabled are termed "enabled".
To summarise:

An intreq is pending if and only if at least one deviceflag of an enabled device is raised.

When a program runs with the interruptsystem turned on (ION), an intreq is granted by an interrupt. The following occurs:

1. The current values of ifr, dfr and the usermode flipflop are stored in the socalled saveregister.
2. The value of the pc is stored in location 0000 of field 0, INTPC.
3. The interruptsystem is turned off (IOF, deaf).
4. A special part of the program, the interrupt section, is entered at location 0001 in field 0.

The interrupt section has to find out which device caused the interrupt. This is done by interrogating the deviceflags one by one. Once a raised flag of an enabled device has been found, it is cleared or the device is disabled to remove the intreq and the device is serviced. Now the interrupted program can be resumed.

### 1.5.1 The MC8 interrupt section

1. Upon entrance of the interrupt section the status of the interrupted program is temporarily saved.
2. The skipchain is rushed through to find a raised deviceflag of an enabled device. Such a flag must be present.
3. When a flag is found, some device-service routine is called. This routine either clears the flag or disables the device. Moreover, it may cause side effects to communicate with tasks in the system. All device-service routines end with calling the routine RPINTR. The location after the call either hoolds 0000, specifying that no message need be reported, or the msptr of the message to be reported.
4. After this call, the intreq will probably be removed. This is uncertain however, as meanwhile flags of other enabled devices may have been raised. Some PDP-8 computers allow to interrogate the "interrupt request line" to see if another intreq is still pending. If so, step 2 is repeated.
5. By now the interruptsystem can be turned on again. A decision must be made either to resume the interrupted program or to activate the scheduler in order to start up some task (see sec. 1.6.3).

### 1.5.2 The skipchain

The skipchain is a piece of code, that is rushed through in one direction (no jumps back). It is used to find a raised flag of an enabled device. The skipchain is built of socalled interruptslots (intslots): sequences of 5 locations corresponding to one deviceflag. Each deviceflag has its own intslot.

The first instruction of an intslot is a skipiot, skipping the next instruction if the corresponding flag is set. If it is not, the program must interrogate the next deviceflag, hence the second instruction is a "jmp .+4" to jump to the first instruction of the next intslot. The remaining 3 locations of an intslot check if the device is enabled and, if so, either constitute or activate the device-service routine.

The order in which intslots are arranged in the skipchain (i.e., the order in which deviceflags are tested) is very important, as this represents a software priority arbitration. Moreover, a bad arrangement may cause considerable (or even intolerable) overhead. E.g., the intslot of a device causing many interrupts must not be located at the end of the skipchain. The same goes for a device (e.g. DECtape or RK disk) requiring fast servicing. The order of the intslots is determined at assembly time by the contents of the config file (see appendix A).

### 1.5.3 Intslots and related monitor commands

The contents of the first two locations of an intslot is steady. They hold the skipiot and a jump to the next intslot. The skipiot is devicedependant and may be used to identify the intslot. The contents of the remaining locations depends on the status of the corresponding device (enabled/disabled) and of the way in which communication with a task is established.

The 4 possible states of an intslot and the monitor commands that set intslots in these states are discussed below. All these commands are executed with the skipiot in ac to identify the intslot to be changed. The other devicedependant parameter that must sometimes be specified (FRINTR and CLINTR) is the cleariot. The cleariot is an instruction to clear the deviceflag. If specified, it is passed in arg1.

As intslots establish the communication between tasks and peripheral devices, their contents must exclusively be controlled by one task at a time. The access of different tasks to an intslot must somehow be synchronised. In this first version no attempt is made to coordinate such access. Tasks have to be decent! In future versions another monitor task, the claimtask, is planned. Tasks then need the explicit permission of the claimtask, before they are allowed to access an intslot.

Free intslots.

When no task in the system is interested in the interrupts caused by some device, and yet the device is enabled, the corresponding intslot must be set to free. The only thing that need be done when such a device interrupts, is clearing its deviceflag.

The contents of the last three locations of the intslot is:

```
  CLEARIOT /clear deviceflag
  JMS RPINTR /clears ac and no report.
   Ø /when called with zero argument.
```

The monitor command FRINTR sets an intslot to frree.
argl =cleariot.

Disabled intslots.

If a device is disabled, the program must continue rushing through the skipchain, even if it finds the corresponding deviceflag set. This device did not cause the interrupt. Calling RPINTR when the program reaches the intslot of a disabled device would prevent proper device servicing and hang up the system.

The third instruction of a disabled intslot is "jmp .+3", that is a jump to the next intslot.

The DISINTR command sets the intslot to disabled. It uses no arguments.

Claimed intslots.

The choice of the name "claimed" is somewhat misleading. When an intslot is claimed, a message is attached to it, that is reported each time the corresponding device interrupts. The task that attached the message to the intslot may recognise the interrupts by inspecting its reportqueue. It may wait for the interrupt by executing the WTRP command (see sec. 1.2). This is the common way of synchronisation between a task and a peripheral.

The contents of the three last locations of a claimed intslot is:

```
     CLEARIOT      /clear flag, possibly read status
     JMS RPINTR    /store status in message and report it
     mmmm          /msptr of message to be reported
```

The routine RPINTR fetches the msptr, reports the message and jumps to FSTEXT. If no msptr is present (free intslot), control is immediately sent to FSTEXT. FSTEXT is the spot where we are about to leave the interrupt section (sec. 1.5.1 step 4).

The command CLINTR attaches a message to the intslot.
Argl =cleariot,
Arg2 =msptr.

The communication and synchronisation using a claimed intslot might go as follows:

A. Request a message (MSREQ pseudo instruction).
B. Execute the CLINTR command, specifying the desired device, the desired cleariot and the msptr.
C. Activate the device, using device dependant code.
   The device is now busy and will interrupt when it is ready to receive new directives.

D. The task may continue to run, executing the CHKRPQ command from time to time (polling) to see wether the device has already interrupted, or it may execute the WTRP command to wait until the interrupt occurs.

E. When the device interrupts, the cleariot is executed, possibly reading a status in ac, ac is stored in the message and the message is reported.

Example. Communication with a keyboard.

```
     KCC            /clear keyboardflag
     IAC;KIE;CLA    /enable device
     MSREQ          /request message
     DCA MSP        /store in arg2
     TAD MSP        /fetch msptr again
     DCA MSP1       /store for future use
     TAD (KSF       /skipiot in ac
     CALL;CLINTR    /execute CLINTR command
         KRB        /arg1 =cleariot
MSP,     Ø          /msptr is stored here
     ...
KRD, Ø              /entry of read char routine
     CALL;WTRP      /wait for report
MSP1,    Ø          /arg1 =msptr
     Cla            /remove msptr from ac
     CDF Ø
     TAD I MSP1     /fetch char from message
     CDFCUR
     JMP I KRD
```

Connected intslots.

A more concise communication between a task and a peripheral can be established by connecting a routine to the corresponding intslot. In principle this routine is called each time the corresponding device interrupts. The routine must be stored in a locked field.

The contents of the last three locations of the intslot is:

```
     CIF CDF FLD    /ifr and dfr to field of routine
     JMS I .+1      /subroutine call
     ssss           /pointer to subroutine entry
```

The CNINTR command connects an intslot and a routine.
Arg1 =pointer to routine entrypoint.
Arg2 =msptr.
Dfr =field in which the routine resides.
The message whose ptr was passed in arg2 may be used by the connected routine to report to the task. If no such message is used, arg2 must be =Ø.

From the instructions listed above it is seen, that the routine is called not only when the device interrupts, but each time the

intslot is checked and its skipiot skips. Therefore the routine is called as a proper subroutine. If the device is not enabled, and thus the routine is called without reason, it can return via its subroutine link in order to continue inspecting intslots. Note that this link must be incremented by 1 before using it and that ifr and dfr must be reset to Ø by a "CIF CDF Ø" instruction.
If the device is enabled, it is the responsibility of the routine to clear the flag or to disable the device. The routine must end by calling RPINTR. At the moment of the call ifr must be set to Ø (by a "CIF Ø" instruction) and dfr must be set to the current field (in order to insure a proper argument handling by RPINTR).
A special macro was introduced for this purpose: IEXIT. It calls RPINTR via a poiinter on the general page Ø. The end of a connected routine could look like:

```
CIF Ø
IEXIT
msptr   /or ØØØØ.
```

### 1.5.4 Connected routines

As stated above it is possible to connect a routine to an intslot. This routine is called to service a device each time it interrupts. Several restrictions apply to such routines.

A. As devices interrupt at unpredictable times and as the routine is called without any further check, the task that connects the routine must ensure that the routine remains in the actual core. This is done by locking the field in which the routine resides.
B. If the corresponding device is disabled, the routine must return control to the skipchain, as otherwise intslots closer to the end of the skipchain will be serviced incorrectly.
C. As the routine is directly called from the skipchain, it is not allowed to turn on the insterruptsystem. Nor is it allowed to use any variable or call any subroutine that is at the same time accessible from any task. Especially the general page zero must be used with care.
D. It is undesirable to have the interruptsystem turned off (to be deaf) over a long period. As the machine is deaf while executing connected routines, these routines must be as fast as possible.
E. As connected routines do not use system calls to return, they must carefully set the registers ifr and dfr before returning. Upon entrance ifr and dfr refer to the field of the routine. At return ifr must be set to 0, and dfr must refer to the field of the routine, as it is used to fetch the msptr.

Connected routines are used to speed up task-peripheral communication. We give two examples.

The RK disk.

The RK disk control raises its flag as soon as the transfer of one block is completed. If the next block is to be transferred too, this transfer must be initiated within 100 microsec. After that time the blockheader has passed under the heads, so the transfer of the next block is delayed by one revolution of the disk. As scheduling takes about 800 microsec, it would take too long to schedule a drivertask upon recognition of the disk interrupt.
The job must be done by a connected routine. The drivertask starts up the transfer and specifies to the routine the number of blocks. When a disk interrupt occurs, the routine initiates the transfer of the next block. When all blocks are transferred, it calls RPINTR to report a message to the drivertask.

Buffered I/O.

Some devices (e.g. a fast papertape reader) cause so many interrupts, that scheduling at each interrupt would give too much overhead. A driver for such a device might hand a buffer to a connected routine. The routine would handle the I/O and report a message as soon as the buffer is filled.

Generally a connected routine can be viewed as a loop. Each time the corresponding device interrupts one cycle through the loop is made, termiinating with the sequence:

```
CIF 0; IEXIT; 0
```

when the loop has been executed a number of times, some endcondition will be detected. This may be visualised to the outside by reporting some message:

```
CIF 0; IEXIT; msptr
```

To get the loop started, it is often easy to enter it halfway as if an interrupt had occurred. A task cannot simulate the occurrance of an interrupt. If a task jumps into a connected routine, serious errors occur at the ennd of the interrupt section. Therefore we introduced the monitor command SIMINTR.

The SIMINTR command requires a pointer in the loop as an argument and assumes that dfr refers to the field of the connected routine. The task seems to be interrupted at the point where it executed the SIMINTR command. The connected routine is entered at the spot indicated by the argument. When the routine executes the IEXIT macro the normal procedure at the end of the interrupt section will be performed. The task is resumed as soon as it is schedued.

### 1.5.5 Changing the state of an intslot

After this description of the commands that control the contents of the intslots, we will point out the possibilities of this mechanism. It will turn out that changing the enabled/disabled status of a device is a delicate action. Sometimes it can only be accomplished within a connected routine. When trying to implement a more concise control mechanism, one hits on the problem of the very baroque design of many device interfaces for the PDP-8 computers. Especially the enable and disable instructions often have undesired side effects on the performance of a device, and require strange bitsettings in the ac.

Each device in the system has a default status (enabled or disabled). This status is determined at assembly time *1.

When a given device is not used by any task, the corresponding intslot must be in the free state if the device is enabled, and in the disabled state if the device is disabled. Initially all devices, except those used by monitor tasks, have the default status and their intslots are free or disabled respectively. It is good practice that a task returns a device (and its intslot) in the default status after using it.

To use a device in its default status is straight forward. The commands to change the intslot can be executed without harm. Using a device in the nondefault status implies that its status must be changed. This is complicated, as at each instant of time the state of the intslot must be in accordance with the status of the device. The intslot of an enabled device must be free, claimed or connected. The intslot of a disabled device must be connected or disabled. As the enable or disable instruction and the change intslot command cannot be executed simultaneously, it is hard to fulfill this requirement.

First we note, that a possible disagreement between the status of a device and the state of its intslot can only be detected (and then blows up the system) when the corresponding skipiot skips. Consequently no errors occur as long as the deviceflag is cleared. So, changing both the status of a device and the state of its intslot is allowed within a part of the program where the deviceflag remains cleared.

*1 Usually the default status of a device is chosen equal to the status reached on receival of the "initialisation pulse". The "initialisation pulse" is sent by depressing the "start"- or "clear"-button on the console of the machine, or by executing the CAF instruction.

If no such part exists, the problem may be solved as follows. First a routine is connected to the intslot, using the CNINTR command. This routine uses a software flipflop to indicate the status of the device. If this flipflop is set to disabled, the routine returns immediately to the skipchain, else it services the device and jumps to FSTEXT. Initially the setting of the flipflop corresponds to the default status of the device. Now the above requirement reduces to the requirement that the setting of the flipflop agrees with the status of the device. But, as a possible disagreement can only be detected by calling the routine, it suffices to guarantee that the routine is not called during a period of disagreement. This can be achieved by changing both the flipflop and the status of the device either within one subroutine call, or in a section accessed using the SIMINTR command.
Note: in some cases a software flipflop is unnecessary. Some devices (e.g. DECtape) allow programs to interrogate their status. In such cases a connected routine can always decide between returning to the skipchain and servicing the device. Changing the status of the device then cannot cause troubles.

## 1.6 Flow of control

### 1.6.1 Execution states

To clarify the discussion below we distinguish 5 execution states. At each instant of time the system is in exactly one of these states. Sometimes the decision when precisely the system transides from one state to the other is somewhat arbitrary.
The execution states are:

A. Executing the code of some task.
B. Scheduling. Selecting or starting up some task.
C. Executing some pseudo instruction (i.e., executing the code of some routine, that implements a pseudo instruction).
D. Executing a monitor command.
E. Executing code of the interrupt section.

Note: monitor tasks, like other tasks, are executed in state A.

### 1.6.2 Task switching

In the states A, C and D the system is dedicated to one task: the current task. It is either directly executing the current task´s code (state A), or it is performing a pseudo instruction (state C) or monitor command (state D) for the current task.
The system will remain dedicated to the current task, until:

1. At the end of a monitor command the current task has set itself to wait; some other task must be selected.
2. At the end of a monitor command a task, more important than the current task, appears to be runnable; the more important task must be selected.
3. At the end of a pseudo instruction runcheck2 fails (the new datafield is not in the actual core); some other task must be selected.
4. After treating a device interrupt a task, more important than the current task, has become runnable; the more important task must be selected.

At such a moment the system has to switch tasks, save the current task´s status in its tcb, and activate the scheduler.

The first three situations are detected at well-defined spots in the monitor program. It is then easy to pick up the status of the current task, save it and enter the scheduler.

The fourth situation is detected at the end of the interrupt section. The proper action to be taken then depends upon the execution state of the system at the moment of the interrupt.

1. State A at the moment of the interrupt.
   At the moment of the interrupt the system was executing the code of the task. Hence the status of the task was present in the hardware registers and saved at the start of the interrupt section. This status may be copied into the tcb of the task, whereupon the scheduler can be activated.
2. State B at the moment of the interrupt.
   The system was scheduling when the interrupt occcurred. Consequently the system was not dedicated to a special task. The scheduler can be activated without saving a status.
3. State C or D at the moment of the interrupt.
   This is complicated.The system was halfway the execution of a monitor command or pseudo instruction when the interrupt occurred. Scheduling an other task, that subsequently might execute the same monitor command or pseudo instruction, inevitably leads to errors. Variables and subroutine entries will get overwritten and perhaps the contents of system tables will be lost.
   Therefore during state C or D either the ION/IOF flipflop is set to IOF (postponing interrupts hardwarewise), or SCDINH is set (preventing activation of the scheduler at the end of the interrupt section).

Remark.

Some tasks use tcb[11] bit3 (see sec. 2.2) to run with SCDINH set. This inhibits scheduling caused by device interrupts and effectively overrides the normal priority scheme. In such cases task switching only occurs at the end of some monitor call or at the end of some pseudo instruction.

### 1.6.3 Critical sections, scheduling inhibitionflag

Some sections of the system can be executed in more than one execution state. For instance, the section that saves the status of the current task may be executed at the end of a monitor command (state D), after the execution of some pseudo instruction (state C) and at the end of the interrupt section (state E). The message report routine may be called from the interrupt section (state E) and may be called to implement a REPORT command (state D).

Most changes in the execution state occur under control of the monitor program (albeit indirectly via a pseudo instruction), and do not cause troubles. Device interrupts (changes to state E) occur at unpredictable times. Critical sections (when executed in a state other than state E) must be protected against these interrupts. This protection is effected using the hardware (ION/IOF)flipflop and a software flag, SCHINH (scheduling inhibited).

To understand the use of these two flipflops, we divide the interrupt section into two parts: the first part contains the skipchain and the device-service routines (1.5.1 steps 1-4), the second part decides either to resume the interrupted program or to save the status of the current task and to activate the scheduler (1.5.1 step 5).

The simplest protection against device interrupts is turning the interruptsystem off (IOF). Intreqs are not granted by an interrupt; interrupts are postponed until an ION-instruction is executed to turn the interruptsystem on again. In this way one can exclude interrupts in critical sections.
Postponing interrupts for a rather long period is undesirable, as many devices like or need real time servicing. Therefore the second part of the interrupt section first checks the SCDINH flipflop. If it is set the interrupted program is always resumed. These sections that only conflict with part 2 of the interrupt section (status saving) are sufficiently protected by setting SCDINH and can be executed with ION. Devices can normally be serviced, but subsequent scheduling, if required, is delayed until SCDINH is cleared. (Note: this forces a test for SCDREQ at all places where SCDINH is cleared.)

In the listing in appendix A we tried to mark the critical sections. The interrupt section is marked with a "!" on each line. Sections of state C and D that are executed with IOF, because they conflict with the first part of the interrupt section, are marked with a "\" on each line. Some sections where the execution state changes are marked with "-" on each line.

Summary.

Tasks (state A) are executed with SCDINH cleared and ION (but see remark in sec. 1.6.2).

The scheduler (state B) runs with ION. Loading the status of a task into the hardware registers conflicts with the status save routine and hence is done with SCDINH set. Just before task start-up, SCDINH is cleared and SCDREQ is tested.

Most pseudo instructions (state C) are executed with IOF. Using SCDINH would take too much time. If a datafield is not in the actual core, SCDINH is set, the interrupt system is enabled and the status save routine is called. This is achieved by executing an emergency call (see sec. 1.7.7).

Monitor commands (state D) are executed with SCDINH set. Sections that conflict with part 1 of the interrupt section are executed with IOF.

The interrupt section (state E) is of course executed with IOF. Part 2 can only be executed if SCDINH is cleared.

Remark.

Parts of the monitor, that are accessed by monitor tasks as well as by routines to implement monitor commands, constitute another area of code accessed in several execution states. When a monitor task is running and no other tasks can force scheduling, no problems arise. Either the monitor task is in execution, or the system performs a monitor command for it. These two cannot get mixed.
As monitor tasks (except the idletask, which does not give conflicts) either have priority Ø or run with SCDINH set, no other tasks can interrupt them. Hence scheduling will not occur until a monitor task explicitly sets itself to wait!

## 1.7 The general page Ø

The lowest page of each field, page Ø, plays a special role in the PDP-8. From any other page in the field it may be accessed with a direct reference, whereas all other interpage communication requires indirect referencing. Therefore in many programs page Ø holds global constants, variables and subroutines. Moreover, locations ØØ1Ø-ØØ17 octal are in page Ø. These locations are the socalled autoincrement registers. They are very useful in accessing data in linear arrays.

In the MC8 system tasks are relocated, so that they cannot be sure to be located on page Ø. Hence they cannot benefit from the special properties of this page, and even worse, in some cases unmeant and unlucky use of the autoincrement registers might cause serious errors. We decided that tasks are never located on page Ø (see sec. 1.1.1).

In order to preserve the special advantages of page Ø for the MC8 system, we used it for the following purposes:

storage of task registers,
storage of the one-task-only part,
storage of general constants,
the implementation of pseudo instructions,
the entrance of the monitor command section.

### 1.7.1 Storage of task registers

In order to permit the use of two more global registers in a task, two locations on page Ø are used, namely BASE and X. They may be accessed from each page of a task. Their contents is saved and restored together with the contents of the hardware registers such as ac, pc etc. Besides that, BASE and X play a crucial role in group 2 pseudo instructions (see sec. 1.7.4).

In some PDP-8 machines a hardware mq register is not present. The usage of a number of features (e.g., the JUMS pseudo instruction) is much simpler when the mq is present. Therefore in those machines that lack a hardware mq we introduced a register similar to BASE and X, named MQ. The presence of a hardware mq can be determined from the type specification in the config file (PDP-8/I or PDP-8/E, EAE not defined or EAE=1).

To facilitate the use of this MQ register, we introduced the instructions STORMQ to load it, and GETMQ to fetch its contents. When a hardware mq is present, STORMQ is implemented as "MQL", GETMQ as "MQA". When a hardware mq is missing, STORMQ is implemented as "DCA MQ", GETMQ as "TAD MQ".

The effect of STORMQ is equivalent in both cases, apart from subsequent EAE instructions. GETMQ either ´or´s or adds the contents of mq and ac, and is equivalent only if ac is cleared before executing this instruction.

### 1.7.2 The one-task-only part

If a task wants to use substantially more of page Ø than the two or three registers mentioned above, it may request for the one-task-only part of page Ø. This must be specified at assembly time and results in setting the ZREQbit (zero request bit) in the task´s entry in the stl. While swapping this task into the virtual core the task fetcher recognises this bit, whereupon it allocates the task in a field in which the one-task-only part of page Ø is still free.

Wether the one-task-only part of page Ø of a given virtual field is free or not can be determined from the ZREQbit in its entry of the fieldtable (see sec. 1.4.3). It is clear that at most one task in a virtual field can have access to the one-task-only part of that field.

The one-task-only part consists of locations Ø-13 octal (containing 4 autoincrement registers) and locations Ø15Ø-Ø177 octal. When swapping in a task, nothing is stored in the one-task-only part. Therefore initially and after each SWAPOUT command the contents of these locations is undefined. For reasons that will become clear in section 1.7.6 no executable code must be stored in locations Ø-13 octal.

The one-task-only part of virtual field Ø is used by the interrupt section and other parts of the monitor.

### 1.7.3 General constants

To save space in many tasks a number of commonly used constants, such as 2, 3, 77 and the like, are stored in page Ø. They are listed in appendix A. One important constant on page Ø is a pointer to RPINTR. The IEXIT instruction used to return from a connected routine calls RPINTR via this pointer (see sec. 1.5.4).

### 1.7.4 Pseudo instructions

Page Ø holds a number of routines (or at least their entrypoints) which implement functions that cannot be executed by the task itself. Some routines have to access system tables and variables (group 1), others require task switch inhibition when applied in their full power (group 2).

Calling such a routine is termed executing a "pseudo instruction". To the programmer of the task these calls look like instructions that are executed, sometimes taking ac or the contents of the next location as an argument. Like ordinary instructions pseudo instructions have a single word mnemonic.

Below we discuss the pseudo instructions (in sofar this is not done elsewhere). For details we refer to appendix A. The routine implementing the pseudo instruction is indicated in the first line of each description.

Group 1.

MSREQ =JMS VMSNOD
No argument, but ac must be cleared at the call.
Request a message.
ac:=msptr (pointer to first informationword of message).
dfr:=ifr; link is disturbed.

MSFREE =JMS VMSNOD
ac=msptr (pointer to first informationword of message).
Return a message.
ac:=contents of first informationword of message.
dfr:=ifr; link is disturbed.

VRCDF =JMS VVRCDF
Change datafield to requested field (see sec. 1.4.2).

ERHLT =JMS VERHLT
Halt this task. Not yet implemented.

VCDF =JMS VVCDF
Argument: next location bit 6-11 (6-bit fieldnumber).
Change datafield to indicated field.

VRDF =JMS VVRDF
No argument, but ac must be cleared at the call.
Read datafield.
ac:=virtual datafield.
dfr:=ifr.

CDFCUR =JMS VCDIF
Change datafield to instructionfield (see sec. 1.4.2).

Group 2.

The group 2 pseudo instructions are designed for the implementation of reentrant tasks (see sec. 2.2). Each incarnation of the task has its own tcb. Its variables and subroutine links are stored in an array, called the reentrancy array. BASE holds a pointer to that array. The group 2 pseudo instructions are meant to access data in the reentrancy array.
Note: of course nonreentrant tasks can benefit from the possibilties of these pseudo instructions.

All these pseudo instructions perform an indexed address operation. BASE is used as base register to whicch the relative address of the data to be accessed is added. This relative address is usually indicated in the location after the pseudo instruction. At the conclusion of the pseudo instruction X holds the absolute address of the accessed data. This may be used for subsequent access of the same location.

Note, that the reentrancy array must reside in the instructionfield of the task.

```
GET      =JMS VGET
ac must be cleared at the call.
Relative address in next location.
Fetch the contents of the addressed location.
dfr:=ifr.

PUT      =JMS VPUT
ac holds data to be stored.
Relative address in next location.
Store data in ac in indicated location.
ac:=Ø.
     dfr:=ifr.

JUMS     =JMS VJUMS
ac points at subroutine entry.
Relative address in subroutine entry.
Reentrant subroutine call. Return address is stored in the indicated
     location.
ac:=Ø.
     dfr:=ifr.
```

Example of subroutine call.

```
     CLA            /ac:=Ø
     TAD    )TY     /get relocated address of subroutine
     JUMS           /call
     ...            /will return here
TY,  TYLINK         /subroutine entry holds relative address.
     ...            /subroutine body
     GET; TYLINK    /fetch return address
     DCA  X         /temporary. untouched in task switch
     JMP I X        /return.
```

This way of calling a subroutine makes the passing of arguments a bit complicated. Ac is used during the call operation proper and writing the arguments just behind the call does not work in reentrant code. We advise to pass arguments via the mq.

### 1.7.5 Entrance of the monitor command section

When a task executes the instruction CALL (=JMS VCALL), the monitor command section is entered. In fact a subroutine VCALL is called. VCALL is on the general page Ø. It disables the interruptsystem and jumps to the label MONITOR in field Ø. There the status of the task is saved temporarily, the specified command is fetched from the location after the CALL instruction and executed (see sec. 1.1Ø).

### 1.7.6 Saving the pc

As stated in sec. 1.6.3 pseudo instructions (state C) and monitor commands (state D) are executed either with the interruptsystem disabled or with SDCINH set. Therefore a task should execute an IOF instruction before each pseudo instruction and the CALL instruction. We do not like tasks to touch the interruptsystem and hence devised the following solution.

The entrypoints of the VCALL routine and all routines that implement pseudo instructions are located at addresses <=76 octal. The first instruction of these subroutines disables the interruptsystem. It will remain disabled until either the execution of taskcode is resumed, or the SCDINH flag is set. In this way dangerous interrupts can only occur just before the first instruction of such a subroutine, i.e. with pc <=77 octal. As no other code is stored at addresses <=77 octal (see sec. 1.7.2), the interrupt section may easily recognise this situation.
when the interrupt came just before the first instruction of the VCALL routine (monitor command section) the interrupt section acts as if the SCDINH flag had already been set. It resumes the program where it was interrupted, i.e. at VCALL+1. A possible schedule request will be treated at the end of the monitor command section.
If the interrupt came just before the first instruction of one of the other routines, the interrupt section fetches the pc from the entrypoint of the called routine. In this way the status of the task is saved as if it had not yet performed the pseudo instruction. This is effectively true as no instructions of the routine implementing it are executed yet. The pseudo instruction will be executed when the task is resumed.
These rules must of course not be applied when the usermode flipflop was set at the moment of the interrupt. The interrupted program then was not an ordinary task, but a user program running under timesharing control. Such programs do not use the general page Ø, and are permitted to execute all kinds of code in their page Ø.

When the interrupt section is about to enter the save status routine, it must perform the following check:

```
'if' usermode flipflop was set or pc >=100 octal then 'goto'
    save status 'fi';
'if' pc=VCALL+1 then resume interrupted program
'else' pc:=ifr[pc -1] -1 #value before pseudo instruction#
'fi'
save status: ...
```

### 1.7.7 Emergency escapes from pseudo instructions

Pseudo instructions are used so often, that the routines that implement them must be as efficient as possible. Although they could run very well with SCDINH set, setting and clearing this flag and the related testing of SCDREQ would take too much time. Generally they use a CIF instruction to inhibit interrupts until the next jump. Their code is all in line, and when the returnjump is executed the interruptsystem is automatically reenabled.

This construction does not allow testing of conditions. When such a test would fail and a jump were made to take proper action, this jump would reenable the interruptsystem prematurely with disastrous effects. The VCDF and VRCDF pseudo instructions however require testing to ensure that the new datafield is in the actual core. The required jump is implemented by a JMS VCALL instruction. This instruction virtually enters the monitor command section. As may be seen in the previous section, a possible interrupt immediately after this JMS VCALL instruction will not disturb the program. The monitor command section may recognise such emergency calls from the value of pc at the call. It knows which pseudo instruction executed the emergency call and which check failed. It takes appropriate action.

### 1.7.8 Storing and updating the general page 0

A copy of the general page 0 is always present in field 0. Initially no other fields hold a copy of this page. As soon as tasks are swapped into a virtual field a copy of the general page 0 must be present in that field. Therefore the page allocator called by the task fetcher checks if the general page 0 is present in the field in which it allocates the task. If page 0 is not yet in that field, it is copied into it from field 0 (see sec. 1.8.3). The presence or absence of the general page 0 in a given field is indicated by the moninbit (bit 9) in the fieldtable (see sec. 1.4.3).

The copy of the general page Ø may be erased from a field if the field is given to a task for datastorage (REQFLD command). In that case the moninbit is cleared and the datastoragebit is set.

The copies of the general page Ø are not identical in all fields. Some CIF and CDF instructions in the general page Ø must agree with the actual field in which they are executed (e.g. the instructions to implement the CDFCUR pseudo instruction). So each time a virtual field holding a copy of the general page Ø is swapped into an actual field these instructions must be updated. This is done by the swaptask, unless the datastoragebit is set.

## 1.8. Storage allocation

There are three storage allocation systems in MC8. The first works on small nodes (maximum 16 consecutive locations) in field Ø, the second allocates complete fields and the third system works on n consecutive pages of one field of the virtual core, where 1<=n<=31.

### 1.8.1 Free core in field Ø

During assembly all free core in field Ø is gathered in a singly linked list of nodes of consecutive free core. The first word of a node holds a pointer to the successor node (Ø indicates the end). The second word points to the last location of the node. The list is organised from high to low, i.e. nodes corresponding to a higher address precede those corresponding to a lower address.
At the initialisation of MC8 this list is reversed and nodes corresponding to an address <4ØØØ octal are broken such that they consist of at most 15 locations. The latter is done to obtain a proper allocation of tcb´s (see below). We will indicate this list as avhead, as AVHEAD is the location that points to its first node.

In field Ø 4 availlists are used. Named after the locations pointing at their first nodes these are: avl2, avl3, avl5 and avl2Ø working on nodes of 2, 3, 5 and 16 words respectively. (16 =2Ø octal.)
Nodes of length 2 are used for the timeoutqueue; nodes of length 3 are used in the free core chain; nodes of length 5 are used for message allocation; and nodes of length 16 are used to allocate tcb´s. Initially the 4 availlists are empty.

When a routine in the monitor needs a node of certain length, it calls the appropriate getnode routine GN2, GN3, GN5 or GN2Ø. The called routine first inspects the corresponding availlist, and, if possible, takes a node from that list. If the availlist is empty avhead is searched for a node of sufficient length. From this node a node of the requested length is broken and the remainder, if its length is still >=2, is returned to avhead.

A node may be returned by calling the freenode routine FN, specifying the appropriate availlist as an argument. This routine adds the node to the corresponding availlist.

Remarks.

It is very difficult to make a useful static division of the free core into nodes of 2, 3, 5 and 16 locations. The number of nodes of each length that is needed depends on the way the system is used. In MC8 this division is postponed as long as possible and done dynamically. A disadvantage is, that the initial behaviour of the system has far more influence than the behaviour at later times.

As a consequence of the initialisation of avhead, nodes of length 16 (used to store tcb´s) are always located at addresses >=4000 octal, as was required (see sec. 1.1.2).

When at a given instant of time avhead is searched for a node but does not contain a suitable node, the system halts as if all free core in field 0 were exhausted. This need not be the case. Other availlists may still contain nodes, that, when merged with one another and the nodes in avhead, would yield sufficient free core. Such a garbage collection is not built in MC8.

Note: garbage collection would erase the extra influence of initial behaviour.

### 1.8.2 Requesting and returning fields

There are upto 64 virtual fields each of which is either free or occupied. Which of these two is the case is indicated in the fieldtable (see sec. 1.4.3).

When the page allocator needs a field it calls GETFLD. GETFLD searches the fieldtable for a free field. If no more fields are free, the system halts.

The page allocator returns a field by calling FRFLD.

A task may request a field for buffer storage. This is done by executing the REQFLD command. This command is implemented by calling GETFLD. The virtual fieldnumber is returned in the task´s ac bit6-11. As page 0 of such a field may get overwritten, the moninbit in the fieldtable is cleared and the datastoragebit is set, to indicate that a copy of the general page 0 is no longer present in the field.

A task may return a field by executing the RTNFLD command, specifying the virtual fieldnumber of the returned field in ac bit6-11. To implement this command FRFLD is called.

### 1.8.3 The free core chain

To allocate tasks and buffers in the virtual core the "free core chain" (fcchain) is used. The fcchain is a singly linked, sorted list of 3-word nodes. Each node in the fcchain corresponds to a junk of 1-31 pages of free core in one field. If all 32 pages of a field are free, this is not denoted in the fcchain, but it is denoted in the corresponding entry of the fieldtable. The fcchain is sorted according to increasing field- and pagenumber.
Note: loosely speaking, we will often say that a junk is in the fcchain, meaning that the node corresponding to that junk is in the fcchain.

1-31 pages of free core may be requested from the page allocator. It returns a buffer of the requested length. The allocation of that buffer is indicated by specifying its virtual fieldnumber and the pagenumber of its first page.
The buffer will never be located on page Ø, because tasks may be allocated in the same field (perhaps this very buffer was meant for task storage). Page Ø is left free for a copy of the general page Ø.

Two more options are specified to the page allocator. The buffer may be requested in a coreresident field and the buffer may be requested in a field where the one-task-only part is still free (this latter option is only used by the task fetcher (see sec. 1.9.4)).

The page allocator first searches a junk that is long enough (note that the fcchain is sorted), then checks the required field properties. If this check fails, searching is continued, otherwise the junk is either deleted from the fcchain (if it has exactly the right length) or the junk is broken and the remainder is left in the fcchain.

Algorithm of page allocator.

```
    'INT' LREQ=#requested length#;
    'BOOL' CRES=#buffer in coreresident field requested#,
    'BOOL' OTO=#buffer in field with free one-task-only part
requested#;
    'INT' F,L;
    #initialise searching#;
S:  #search junk of sufficient length#;
        F:=#fieldnumber of junk#,
        L:=#length of junk#;
    'IF' 'NOT' (CRES =#cresbit 'of' fieldtable[F]#)
        'THEN' 'GOTO' S
    'FI';
    'IF' OTO 'AND' #zreqbit 'of' fieldtable[F]#
        'THEN' 'GOTO' S
    'FI';
    'IF' L=LREQ
        'THEN' #delete junk from fcchain#
        'ELSE' #brake junk#
    'FI'
```

Note: each time the 'GOTO' S is executed, the search is continued where it was stopped.

If during the search the end of the fcchain is reached, no sufficiently long junk is present in the fcchain. GETFLD is called to fetch an entirely free field; the general page 0 is copied into it; page 0 is broken off; and a fresh junk of 31 pages (certainly long enough) is created.

A buffer may be returned to the page allocator by specifying its fieldnumber, the pagenumber of its first page, its length and field properties. The latter is used to clear the zreqbit in the fieldtable if the one-task-only part is free again.
The page allocator creates a node corresponding to the returned junk, possibly merges the junk with adjacent junks in the same field and if the, possibly merged, junk has less than 31 pages, inserts it in the fcchain. If the junk is 31 pages long, the whole field is free again (mind, that no stuff was allocated in page 0). This is only indicated in the corresponding entry of the fieldtable, not in the fcchain.

Layout of a node in the fcchain.

| word | bit | meaning |
|---|---|---|
| 0 | | Pointer to successor. 0000 terminates the chain. |
| 1 | 0-5 | Virtual fieldnumber of junk. |
| | 7-11 | Pagenumber of first page of junk. |
| 2 | | Minus number of consecutive free pages. |

Remarks.

Buffers not requested from the page allocator (but for instance unused pages of a field requested with the REQFLD command) must not be returned to the page allocator. The general page 0 may not be present in this field. When subsequently tasks are loaded into the returned buffer, serious errors can occur.

As the fcchain is sorted, there is a tendency to use the lower virtual fields first. This decreases the number of virtual fields in use, and hence the number of fieldswaps.

The way in which the coreresidentcondition is checked ensures, that as few fields are made coreresident as possible. This diminishes the risk of having many or all actual fields locked.

### 1.8.4 The REQPAG and RTNPAG command.

A task may request 1-31 consecutive pages of one field by executing the REQPAG command.
Ac must be cleared;
argl bit7-11=requested length;
Argl bit 5=1: request buffer in coreresident field.
At return:
  ac bit0-4 hold number of first page of buffer;
  ac bit6-11 hold virtual fieldnumber of buffer.

To return a buffer, previously requested using the REQPAG command, a task may execute the RTNPAG command.
Ac bit0-4 =number of first page of buffer;
ac bit6-11 =virtual fieldnumber of buffer.
argl bit7-11 =length of returned buffer. ac:=0.

## 1.9 Monitor tasks

Monitor tasks are tasks that are built in and closely interrelated with the code of the monitor of MC8. They implement essential functions of the system. All monitor tasks (except the idletask) have priority 0 or run with SCDINH set, so that no other task can interrupt a monitor task when it is in execution. The idletask is the only task having priority 10 (octal). It is less important than all other tasks and each task may interrupt it.

### 1.9.1 The idletask

The algorithm of the scheduler is based upon the fact, that there is always a task that can be executed. Therefore the idletask has been built in the system. It does not perform pseudo instructions and monitor commands and its data- and instructionfield are both =ØØ. So it is never waiting and runchecks cannot fail.

The idletask performs some innocent loop, that can be interrupted at each instant of time. Moreover, it clears the variable CURTSK, suggesting that the system is scheduling. This spares time needed for saving the status if another task becomes runnable (see sec. 1.6.2).

### 1.9.2 The system diskdriver

The system disk is a peripheral device on which virtual core and tasks are stored. A driver for this disk must be present in the system, as the system cannot swap in a drivertask when no diskdriver is available.

The system disk is divided into 8 logical units of 4Ø96 blocks of 256 words each. Wether or not all units are (completely) available depends on the physical device used.
To each virtual field an area on logical unit Ø corresponds, a fieldslot. A fieldslot is 16 blocks (=4Ø96 words) long. When a virtual field is not in the actual core, it is stored in its fieldslot.
Fieldslots are arranged one after another on logical unit Ø. The blocknumber of the first fieldslot is determined by the value of VFBLOK in the config file.

In the stl an 11-bit blocknumber can be specified for each task. This blocknumber indicates where on logical unit Ø the taskcode may be found. The actual blocknumber of the taskcode is computed by adding the value of TSBLOK (specified in the config file) to the value given in stl.

The system diskdriver consists of two parts: a device-dependant section (the SDGO routine) and a device-independant section.

The device-dependant section accomplishes the actual datatransfer to or from the physical device on which the system disk is mounted. We shall not discuss it in this document. The listing in appendix A contains a version for the RK-8/E disk.

The device-independant section accepts, treats and reports messages. Two kinds of messages may be sent to the system diskdriver: normal messages and SWAPMS (the swap message).

Normal messages may be sent by any task. They specify in an obvious way a single datatransfer.

Layout of a normal message:

| word | bit | meaning |
|---|---|---|
| Ø-1 | | Used by the system. |
| 2 | Ø | Ø: read, 1: write. |
| | 1-5 | Number of pages to be transferred. Ø means 4Ø octal. |
| | 6-8 | Actual fieldnumber of field of transfer.<br>This seems a dangerous way to pass the fieldnumber, but it is safe if the DFPARM option is used (see sec. 1.2). |
| | 9-11 | Logical unitnumber. |
| 3 | | Core address of first word of transfer. |
| 4 | | Blocknumber on disk. |

When such a message is received, the SDGO routine is called to perform the actual datatransfer; the completion status is denoted in word 2 (the first informationword) of the message (Ø indicates no errors); and the message is reported.

The system diskdriver recognises SWAPMS from its core address. It is allocated at a fixed core address and is only sent to the system diskdriver. Word Ø of this message has a fixed contents, such that it is always reported to the swaptask.

SWAPMS may be sent by the system upon failure of runcheck2 (see sec. 1.4.6). It specifies that a fieldswap must occur by passing the actual field of the swap, the virtual field to be swapped out and the virtual field to be swapped in to the system diskdriver.

Layout of SWAPMS:

| word | bit | meaning |
|---|---|---|
| Ø | | Tcbptr of swaptask. |
| 1 | | Link in receive- or reportqueue. |
| 2 | | Pointer in CORMAP, corresponding to the actual field. |
| 3 | | Pointer in the fieldtable, corresponding to the virtual field to be swapped out. |
| 4 | | Pointer in the fieldtable, corresponding to virtual field to be swapped in. |

When this message is received, SDGO is called twice, once for the transfer of each virtual field. A possible error in the transfer halts the system. At the completion of the transfer, the message is reported.

Note the following optimisation. When the fieldtable indicates that the virtual field to be swapped does not contain data, SDGO is not called, so that no actual transfer takes place.

Remark.

As stated in sec. 1.2, the best way to pass actual fieldnumbers with the DFPARM option is, to lock the datafield before passing its actual fieldnumber. This ensures that the designated field is still present in that actual field when the receiver accesses it. Locking is unnecessary when sending a message to the system diskdriver. The field under concern cannot be swapped out meanwhile, because a possible message to the system diskdriver to do so, will be behind the current one in the receeivequeue, and hence be treated later.

### 1.9.3 The swaptask

The swaptask does some bookkeeping in system tables after a fieldswap. It is activated by reporting SWAPMS (see previous section). For a description of the swaptask we refer to sec. 1.4.6.

### 1.9.4 The taskfetcher

Runnable tasks, the code of which is still on disk, are swapped into the virtual core by the taskfetcher. Such a task must be built in the system, as if it were on disk it could not swap itself in.

When the scheduler has selected a runnable task, it checks the ondisk-bit (runcheckl, see sec. 1.3.3). If this bit is set, the task is removed from the runqueue and a message is sent to the taskfetcher to swap in this task.

Loop of the taskfetcher.

1. Wait for a message, specifying that some task must be swapped into the virtual core.
   From the information in the message a pointer to the task´s tcb is computed. Using tcb[6] (static tasknumber) the information in the stl may also be accessed.
2. Fetch length and fieldproperties (task uses one-task-only part, task runs in coreresident field) of the task from the stl.
3. Request an appropriate buffer from the page allocator.
   Note, that the page allocator guarantees that a copy of the general page Ø is present in the field of the buffer.
4. Send a message to the system diskdriver to read the code of the task into core. The blocknumber is computed from tcb[15].
5. Update pointers in the code of the task: relocation (see sec. 1.1.1).
6. Update the tcb of the task.
7. Reinsert the task in its runqueue.
8. Return to step 1.

The following locations in the tcb are updated:

tcb[7] Datafield and instructionfield are both set to the field in which the task is stored.

tcb[9] If no value (ØØØØ) was indicated in tcb[9], the pc, this is considered as the start of the task. Pc is made to point to the first location after the first relocatable pointer sequence (see sec. 1.1.1). That is, if no value was indicated for pc, the task is started at the first executable instruction on its first page.
If a value was present in tcb[9], this value is regarded as relative to the task-offset. Hence pc is then updated by adding the task-offset.

tcb[14] The new first page of the task is stored in bitØ-4 of this word.

When a task executes one of the wait commands STALL, WTRP, SNDWTR or WTMS, it may specify the SWAPOUT option.
If the task is actually set to wait (which need not be the case), the pages occupied by its code are returned to the page allocator. As soon as the task is runnable again, the taskfetcher swaps a fresh copy of its code back into core.

Specifying the SWAPOUT option is useful only if the task expects to be waiting for quite a long time (typically more than Ø.5 sec). By long we mean long when compared to the time required to swap in the taskcode, which is about Ø.1 sec.

Moreover, one must be aware of the side effects of the SWAPOUT option:
a fresh copy of the code is swapped in, so all information stored in the code is lost;
the datafield becomes equal to the instructionfield; if other tasks have pointers pointing into the code of the task under concern (which they shouldn´t), these pointers become meaningless.

Remark.

When the taskfetcher updates the code of a task, it acesses a field that perhaps is not in the actual core. Hence runcheck2 may fail when the taskfetcher is selected. To avoid the danger for deadlocks, the taskfetcher does not have priority Ø, but priority 1. As it uses code of the monitor, it must run with SCDINH set. The danger for deadlock is evident from the swap algorithm explained in sec. 1.4.6.

### 1.9.5 The timer

The timer keeps track of the system time and treats the "timeoutqueue". It is driven by clock-interrupts and runs once each Ø.1 sec.

When TIMEMS (a special message) is reported, the timer starts to work. It increments the system time: a 2-word counter that counts the time modulo 2^24 in units of Ø.1 sec.
Next the timer treats the timeoutqueue (see sec. 1.9.6) and resets itself to wait for the next report of TIMEMS.

How precisely TIMEMS is reported depends on the implementation. In the version listed in appendix A, a clock is used, causing interrupts each Ø.1 sec. The corresponding intslot is claimed and TIMEMS is attached to it.

### 1.9.6 The timeoutqueue (toq)

When a task executes the STALL command it sets itself to wait for some distinct period. In fact it delays its execution for a while, therefore such a task is termed "a task in delay". The length of the delay is determined from the socalled "timeout value" specified in ac at the execution of the STALL command. Each Ø.1 sec this timeout value is incremented by 1 and on overflow the task is reinserted in its runqueue. When for instance 7771 octal is specified in ac, the task is runnable again after Ø.7 sec.
Note, that ac=Ø when the execution of the task is restarted.

To implement this command the "timeoutqueue" (toq) is used. Upon execution of the STALL command a node corresponding to the task is inserted in the toq. The toq is a singly linked list of 2-word nodes. The first word of a node points to its successor (ØØØØ terminates the queue). The second word of a node points to tcb[8], ac, of the task in delay. Each time the timer treats the toq, it uses the latter pointer to increment the task´s ac. When overflow occurs the task is reinserted in its runqueue and the node is deleted from the toq. Note, that a pointer to tcb[8] cannot be ØØØØ.

When a task executes one of the commands WTRP, SNDWTR or WTMS, a task may specify the TIMEOUT option. If the task is actually set to wait (which need not occur), a node is inserted in the toq just as if a STALL command had been executed. Of course the other waitconditions specified in the command are also set.

As long as the task is waiting, the timeout value in ac is incremented each Ø.1 sec. If overflow occurs the task is reinserted in its runqueue with ac=0 and the other waitconditions are ignored. If one of the other waitconditions expires (a message is sent or reported to the task) before the end of the delay, the msptr is set into ac, as usual, and the task is reinserted in its runqueue. In the latter case the TIMEOUT option had no effect.

So the use of the TIMEOUT option ensures that the task waits only for a limited time (determined by the timeout value in ac). Either the specified waitcondition expires within that period, in which case task execution is restarted with a msptr in ac, or after the specified period task execution is restarted with ac=0.

Implementing the TIMEOUT option was difficult. Tasks in delay have a node in the toq, pointing at their ac. Each time the timer runs, it increments this ac. When the waitcondition expires, a msptr is loaded into the ac. From that moment it is no longer allowed to touch the task´s ac.

Removing the node from the toq takes too long, as the toq is singly linked. Therefore tcb[5] of a task in delay points to its node in the toq. When the task is reinserted in its runqueue, this pointer is used to clear word 2 (the pointer to ac) of the node. The timer recognises such a node and removes it from the toq without taking further action.

Note, that tcb[5] of a task in delay is not used to point in a runqueue as such a task is not runnable.

### 1.10 Monitor commands

The monitor provides a number of helpful functions for a task. To request for such help a task must execute a CALL instruction. Thereafter the commandword is denoted, followed by 0, 1 or 2 arguments, depending on the command given. Sometimes the ac of the task is also used as an argument. In fact a task "commands" the monitor to do something, therefore this is termed "executing a monitor command".

The command is specified in the commandword. In addition some commands allow options to be specified in the commandword.

Layout of the commandword

| bit | meaning |
|---|---|
| 0 | Check only option.<br>This option changes the WTMS and WTRP commands into CHKRCQ and CHKRPQ respectively (see sec. 1.2). |
| 1 | NONREP option (see sec. 1.2). |
| 2 | KEEP option (see sec. 1.2.1). |
| 3 | SWAPOUT option (see sec. 1.9.4). |
| 4 | TIMEOUT option (see sec. 1.9.6). |
| 5 | DFPARM option (see sec. 1.2). |
| 6-10 | Command specifier. |

11    1: arguments present; Ø: no arguments present.

When the CALL (=JMS VCALL) instruction is executed, the routine VCALL on the general page Ø is called. This routine disables the interruptsystem and jumps to the label MONITOR in field Ø. There the status of the calling task is saved temporarily, the SCDINH flag is set and the interruptsystem is reenabled. Now the monitor is ready to treat the call.

First remember that one of the routines on the general page Ø may have executed an emergency call. This may be derived from the value of the saved pc. If the pc points in one of the page Ø routines, an emergency call was indeed executed. The system retrieves the actual status of the task from the page Ø routine under concern, saves it temporarily and then takes further action depending on which routine called.

If the call was no emergency call, but a normal call executed by a task, the designated action is indicated in the commandword. The commandword and possible arguments are fetched from the field of the caller (thereby incrementing the value of the saved pc) and the system jumps to the appropriate routine, using bit6-1Ø of the commandword as a switch.

The effect of monitor commands is described in detail in other sections of this paper. Appendix E summarises all monitor commands and their options and refers to the sections in which they are described.

## 2 Miscelaneous remarks

### 2.1 Creating tasks

Occasionally a task may wish to create other tasks, e.g., to load a task from a device other than the system disk or to create a new incarnation of a reentrant task (see sec. 2.2). Such a newly created task is termed a descendant of the task that created it.

The code of the descendant must be provided somewhere in the virtual core, maybe in a buffer acquired using the REQPAG command or as part of the code of the task that created it. A static tasknumber and a task controlblock must be attached to the descendant. To create tasks and to remove created tasks the monitor commands discussed below have been added.

REQSTL.
Request entry in stl (request st#).
Arg1, arg2: contents of entry in stl.
ac:=st#.
Note, that a negative contents of arg1 is interpreted as tcbptr.

REQTCB.
Attach a tcb to, or retrieve the tcb of a task.
ac =st#.
ac:=tcbptr (pointer to tcb[5]).
If no tcb is attached yet to the specified entry in the stl (word 0 >0), a tcb is requested, prefilled with initial values (see sec. 1.1.3), and its tcbptr denoted in word 0 of the entry in the stl.
The tcbptr (now) present in word 0 of the entry in stl is returned in ac.

FILTCB.
Fill the tcb of a task.
ac =st#.
Arg1 =pointer to value array.
First attach a tcb to the task (or retrieve its tcb). Then overwrite the first 15 locations of the tcb with the values indicated in the value array. Arg1 points at word 0 of the value array.
Note: tcb[15] is not overwritten. This location holds the original contents of word 0 of the entry in stl.

RTNTCB.
Return a tcb.
ac =st#.
The tcb is detached from the specified entry in the stl and returned to the availlist system. Word Ø of the entry is overwritten with the contents of tcb[15].

The request and fill commands above allow tasks to assign a st# and a (properly filled) tcb to a descendant. The RTNTCB command may be used to return a requested tcb.

To return the entry in the stl used by a descendant that ceases to exist, the task must explicitly clear word Ø of that entry. A pointer to that word is computed by taking twice the st# and adding the constant ZSTL (general constant).

Summary.
Common scheme for creating a descendant:
1. Provide code.
2. Request st#, using REQSTL.
3. Request and fill tcb, using FILTCB.
4. Send message using the st#.

Common scheme for erasing a descendant:
1. Stop descendant if it is still runnable, using STOP.
2. Return its tcb, using RTNTCB.
3. Clear corresponding entry in stl.
4. If possible return the buffer in which the code was stored.

## 2.2 Reentrant tasks

A reentrant task is a task that operates on "reentrant code", i.e., a task that executes code that simultaneously may be executed by other tasks. All tasks that operate on the same code are termed incarnations of the reentrant task. Such incarnations have different tcb´s, and thus may have different values in their registers.

By nature the code of PDP-8 computers does not allow reentrancy. Simple tools like a hardware stack and hardware indexed addressing are missing. Moreover, the return address of a subroutine is stored in a fixed location each time the subroutine is called.

In MC8 the lack of hardware indexed addressing is remedied by the GET and PUT pseudo instructions (see sec. 1.7.4). These pseudo instructions use the register BASE as offset of the address and the location after the instruction as an index. As the value of BASE may differ from one incarnation to the other, each incarnation can store its variables in its "own" array. This array is termed the "reentrancy array".

The JUMS pseudo instruction (see sec. 1.7.4) may be used to store subroutine links in the reentrancy array.

If an incarnation would only store variables in its registers and its reentrancy array, its code would remain unchanged and hence there would be no special problems with reentrant tasks. This however is clumsy and time consuming. The programmer is continuously tempted to store some temporary value in some "local" variable. This would not be so bad, if not at each moment the execution of an incarnation could be interrupted and the execution of another incarnation of the same task be resumed. This latter incarnation might disturb the variable, before the former one is resumed.

Things would be much better if incarnations of a given reentrant task could only be interrupted by other incarnations of the same task at fixed places in the code. At such places the programmer could take care that all values to be used later are properly stored in registers or the reentrancy array, which are unaccessible from other incarnations. There are two methods to achieve this in MC8.

Reentrancybit (tcb[6] bitl).

The use of the reentrancybit requires that all incarnations of a reentrant task have the same priority and that they all have the reentrancybit set. This need not be the case in MC8, but usually it is.

When the scheduler selects a task, whose data- and instructionfield are not both in the actual core, it normally continues inspecting the runqueue of the same priority (see sec. 1.3.3). The scheduler will skip over this runqueue and start inspecting the next important runqueue if the selected task has its reentrancybit set.

This has the following effect.

The execution of an incarnation is started only if it is the first incarnation in the runqueue. Other incarnations of the same task cannot be executed until the current incarnation has been removed from the runqueue, i.e., until it has executed a "wait" command (STALL, WTRP, WTMS SNDWTR).

For instance let Cl and C2 be incarnations of a reentrant task, and let Cl precede C2 in the runqueue. The scheduler will always select Cl first, and either start up its execution or skip over the rest of the runqueue, including C2. Hence C2 will not be executed as long as Cl is present in the runqueue.

Using this method the only delicate places in a reentrant task are those places where wait commands (see above) are executed. A severe disadvantage is, that incarnations of other reentrant tasks (and possibly lots of other tasks) behind the first incarnation in the runqueue are not started until the first incarnation sets itself to wait, even if their fields are in the actual core. They all have to wait until the fields of the first incarnation have been swapped in.

The scdinhbit (tcb[11] bit3).

Tasks that have this bit set run with the SCDINH flag set. This prevents them from being interrupted by any other task, including other incarnations of the same reentrant task. Other tasks can only interfere when a monitor command is executed, or when the task tries to access a field that is not in the actual core.
Delicate places are those places where monitor commands are executed and places where a VCDF and a VRCDF is executed. The danger for the VRCDF instruction is restricted to the first call after a monitor command; later calls will find the requested field in the actual core.

A disadvantage of this method is, that it overrides the normal priority scheme. A task running with the scdinhbit set must provide enough monitor calls in order not to hamper normal progres of other tasks.

Remark.

A reentrant task must never execute the EXIT command or use the SWAPOUT option. This would erase the code from core, although other incarnations might still be using it.

## 2.3 Protect tasks

Protect tasks are tasks running with the usermode flipflop set. They constitute the timesharing mode of MC8. Although we have no definite ideas about the implementation of such tasks, a few words must be spent on this subject. Otherwise some parts of the code of MC8 would look very strange.

A protect task is meant to simulate a dedicated installation to a user. When the protect task runs, the program stored in the memory of the simulated machine is executed. In order to do so, the fields of the virtual core in which the simulated memory is stored are swapped into the actual core (for sofar they are accessed), and the program is started at the location and field indicated in the tcb. Note however, that the usermode flipflop is set.

When the simulated program tries to execute an instruction that might effect other parts of the system or perhaps the outside world, a "trap" occurs. That is, the protect task is interrupted hardwarewise, and when subsequently the deviceflags in the skipchain are interrogated (see sec. 1.5.2), the sint-instruction (6254) will skip to indicate that a trap occurred. Some emulator task (or perhaps a connected routine) may now be started to interprete the trapped instruction and to take appropriate action.

So all interaction between the outside world and the simulated program and all field access of the simulated program (note, that CDF and CIF instructions and the like also cause traps) is controlled by the emulator task.

The need for controling the field access is shown in th following example.
Assume the simulated program attempts to execute a CDF 1Ø instruction in order to access simulated field 1. The emulator task now must find out in which virtaul field field 1 of the simulated memory is stored, check if this virtual field is in the actual core and, if so, make the datafieldregister point to the correct actual field.

Before the simulation is started, the user must specify to the emulator task which configuration he wants to be simulated (e.g. the number of memoryfields and the type of peripheral devices). The emulator task requests buffers to store the simulated memory, using the REQFLD commmand. A simulated program of course has full access to the fields of its memory. Hence it is impossible to load a general page Ø in such fields. As a consequence the pc of a protect task may point into page Ø without special implications and registers such as BASE and X do not exist. To facilitate the treatment of such peculiarities the routine CDFUF is patched. Normally this routine gives the monitor access to the field in which the taskcode is stored. For protect tasks this routine is changed such that it skips. This is done during task start up and is used in the save status routine.
Note, that protect tasks cannot execute monitor commands or pseudo instructions.

## 2.4 The task library

The structure of the task library has little effect on the code of MC8. In the code of MC8 a few tasks, the monitor tasks, are incorporated. Other tasks must somehow be loaded into the system when they are required and unloaded afterwards. This must be done by a task, called the "taskloader". The structure of the task library does effect the taskloader, but not the code of MC8.

The tools available to the taskloader were discussed in sec. 2.1. In general it is sufficient to assign a st# to the task to be loaded and to specify the contents of its entry in the stl, including a blocknumber on disk. If this code is to be swapped in by the taskfetcher, the taskcode must reside on a special area of unit Ø of the system disk (see sec. 1.9.2).

A problem arises when a task wants to communicate with other tasks. It must know the st# of the task it wants to send a message to. The st# is determined dynamically and hence cannot be assembled in the code of a task. The only statically known identification of a task is its taskname. The taskloader assigns a st# to a loaded task and hence it is the job of the taskloader to replace tasknames by st#´s.

This can be done in various ways. It heavily depends on the structure of the task library and the way in which tasknames are indicated in the code.

The solution we chose is briefly outlined below. For some applications it may proof succesful, in other cases it may be inawkward and other methods must be used. In appendix C our version of the taskloader is listed.

### 2.4.1 Families of tasks

In our task library tasks are grouped in families. The taskloader loads (or unloads) complete families of tasks. A taskname consists of two parts: the familyname and the membername. There is one special family named GLOBAL. The members of GLOBAL are very important tasks, including the monitor tasks.

A task can communicate to members of its own family, and to members of GLOBAL. To specify tasknames at assembly time we added three pseudo operands to the assembler (see [5]), namely:

TASKNAME To specify the taskname of the task that is assembled;
GLTASK To indicate a member of GLOBAL;
LCTASK To indicate a member of the family of the task that is assembled.

In the task library each task is preceded by a socalled task info block. This block holds a list of tasknames used by the task, together with pointers indicating where in the code the corresponding st#´s must be inserted.

During the initialisation the taskloader loads GLOBAL. Other families are loaded or unloaded when appropriate messages are received. GLOBAL remains always loaded.

To load a family of tasks the taskloader first assigns a st# to each member of the family. Next it updates the code of each task using the task info block. Note, that in order to do so, the code of each task must be swapped in, updated and swapped out again, even for those members that are not actually used.

## 2.5 Configurating MC8.

MC8 can be used on a variety of PDP-8/I and PDP-8/E installations.
Minimum requirements are: 16k memory, a disk and a clock.

The major part of the code of MC8 is equal on all installations. Some sections may vary slightly and thus are enclosed in the angular brackets "<..>" of conditional assembly.
A number of parameters specifying the configuration are grouped in the config file. They either set values (such as ACTMAX) or control conditional assembly (for instance EAE).

The following must be specified in the config file: (Values indicated with H are prescribed by the hardware used. Values indicated with U are chosen by the user.)

1.H Processor type.
    MC8 running on PDP-8/I or PDP-8/E, EAE (Extended Arithmetic Element) present or not.
2.H Size of the actual core memory.
3.U Size of the virtual core memory.
4.U The length of the static tasklist.
    This defines the maximum number of tasks that can simultaneously be loaded in the system.
5.HU The devices known in the skipchain and their order.
6.U Assembly parameters for extra error check and statistics, if desired (see sec. 2.7).
7.U The devices on which the system disk and the system clock are to be implemented.
8.U The layout of logical unit Ø of the system disk.

The device-dependant parts of MC8 are assembled conditionally, controlled by parameters in the config file. If devices are to be used for which no code is yet available in MC8, the following must be added.
If the device can cause interrupts an initial intslot must be specified in the skipchain.
If the device is to be used as system disk, a device-dependant version of the SDGO routine must be provided.
If the device is to be used as the system clock, things must be arranged such that TIMEMS is reported each Ø.1 sec. This sets the timertask to work. the timertask itself is fully device-independant.

Initialisation.

At the start of MC8 an initialisation routine is called. This routine activates the scheduler when it is terminated.
The first task scheduled is MTINIT, an initialisation task built in

the system. MTINIT ends by erasing all initialisation code from core, and by clearing its entry in the static tasklist.

The initialisation routine organises the availlist systems, initialises all devices and starts the clock.
MTINIT can be used to do config-dependant initialisation that requires the use of monitor commands. Amongst others the taskloader may now be brought into the system, and it can be indicated in the task library that monitor tasks, such as the system diskdriver, are permanently loaded in the system.

The initialisation routine and MTINIT may be assembled separately. Their code must be loaded together with the code of MC8.

## 2.6 Error handling

The error handling in MC8 is very poor, due to the trade-off of constructing a compact and small system.
At present the errors that are detected generate an errormessage on the console teletype and halt the system. The errormessage specifies the address in the monitor where the error was detected.
If the parameter CHECK is defined in the config file, some extra error checking is performed.

Several types of errors can be distinguished:

Resource exhausted.
For instance a buffer can be requested while the virtual memory is completely in use, or a node in field Ø can be requested while the availlist and avhead are empty. Errors of this type can sometimes be recovered by garbage collection. Another improvement would be to issue some warning when a resource runs to its end.

Hardware errors.
Sometimes hardware errors may be detected, for instance errors in a disk transfer. Such errors are unrecoverable. The best that can be done, is to halt the system in such a state, that important data may easily be retrieved.

Software errors.
Checking arguments and registers at the execution of each monitor command would detect most software errors in a task. Such an erroneous task could be stopped immediately in order to limit its effect on other parts of the system.

One of the problems of error handling in MC8 is, that it is difficult to stop a task that is in error or that is about to exhaust some resource. Of course the task can be stopped, but it is impossible to retrieve which buffers and messages must be returned.

MC8 error handling can be improved at the cost of 1 extra field of coreresident code.

## 2.7 The system bulletin

To facilitate debugging of tasks and investigating the performance of the system, a special program, MC8BUL, was developed. MC8BUL produces a document, the system bulletin, in which the contents of important variables of MC8 is listed.

When the parameter STATX was defined in the config file, at the cost of some time and space a lot of counting is done while MC8 is running. The data so gathered are also listed in the system bulletin. They are meant to give insight in the performance of the system.

The present version of MC8BUL runs only under control of the OS/8 operating system. To use it, one must save the contents of field Ø after a run of MC8 on a file. This file may subsequently be fed to MC8BUL.

MC8BUL first calls the commanddecoder to find out where its input is to be found and where its output has to go. Moreover, it accepts two options: /Z, to indicate that the runcount must be reset, and /Y, to indicate that the contents of system variables must be dumped (normally only the result of the counting is listed).

Assuming that /Y was specified and STATX defined in the config file, MC8BUL gives the following output:

1. Date and runcount.
   The runcount is incremented each time MC8BUL is run and may be reset by specifying the /Z option in reply to the commanddecoder.
2. A number of system variables, specifying the current task, the status of the SCDINH and SCDREQ flags, the last status saved in the interrupt section and the monitor command section.
3. The contents of the runqueues.
4. The skipchain.
5. The coremap.
6. The timeoutqueue.
7. The contents of the fieldtable.
8. The free core chain (see sec. 1.8.3).
9. The number of nodes present in the availlists.
1Ø. The status of all tasks loaded in the system.
11. The number of unused entries in the static tasklist. Note, that this value gives the number of entries that still were available in the stl at the peak of its occupation.
12. The number of hardware interrupts seen.

13. The number of schedules at the end of the interrupt section and after a monitor command. Note, that schedules interrupting the idle loop are not counted.
14. The peak length of the messagequeues of the tasks.
15. The number of ordinary swaps to or from the system disk, the number of fieldswaps and the number of swap errors.
16. The number of virtual fields that was actually accessed.
17. The number of monitor commands executed.
18. The amount of space that was used by the availlist systems in field Ø.
19. The run time, the amount of time spent in the idle loop and the time inactively waited for fieldswaps.

Items 12-19 are only listed if STATX was defined in the config file.
To specify the status of a task, the following values are output:
- Its name, retrieved from the task library.
- The contents of its entry in the static tasklist.

If a tcb was attached:
- The contents of tcb[6-15] (see sec. 1.3.3).
- The contents of the claim word (tcb[Ø]) and the first 1Ø messages in the receivequeue.
- The report for which the task is waiting (tcb[2]), if it is, and the first 1Ø messages in the reportqueue.
- The contents of the wait word (tcb[4]) and the linkword (tcb[5]). For their use see sec. 1.3.

In appendix D a system bulletin is listed.

# 3 Literature

[1] Hagen P.J.W.ten, Vliet R.van, Tekstverwerking,
MC Rapport IN 11/76

[2] Anthony J.F. A real-time Foreground/Timesharing Background
Operating System for a minicomputer.
TH-Delft PHD Thesis 1975

[3] RTS-8 Users Manual
DEC-08-ORTMA-B-D
1973

[4] OS-8 Handbook
DEC-S8-OSHBA-A-D

[5] PDP DOC Series LD Series
Last edition

[6] Tanenbaum A.S, Structured Computer Organisation
CH5.5 Prentice Hall 1976

[7] Brouwer A.E, Vliet R.van, Wakker W,
PDP8 Simulation
MC Rapport IW 74/77

```
                  /APP. A0.   CONFIGURATION FILE.
               /PROCESSOR TYPE.
         0001  PDP8E=1
               /PDP8I=1
         0001  EAE=1

         0001  IFDEF EAE <HARDMQ=1>
         0001  IFDEF PDP8E <HARDMQ=1>
         7701  IFDEF  HARDMQ   <GETMQ=CLA MQA; STORMQ=MQL>
         7421
               IFNDEF HARDMQ   <GETMQ=TAD MQ; STORMQ=DCA MQ>

               /INDICATE ACTUAL CORESIZE (HIGHEST ACTUAL FIELD)
         0007  ACTMAX=7

               /INDICATE VIRTUAL CORESIZE (HIGHEST VIRTUAL FIELD)
         0077  VIRMAX=77                   /MUST BE <=77

               /INDICATE LENGTH OF STATIC TASK LIST (HIGHEST ST#)
               /THIS VALUE DETERMINES THE MAX NUMBER OF TASKS THAT
               /CAN RUN SIMULTANEOUSLY.
         0077  MAXSTL=77                   /MUST BE <=177

               /CONFIGURATE SKIPCHAIN.
               /INDICATE YOUR DEVICES IN THE ORDER OF THEIR
               /INTERRUPT PRIORITY.
               NOPUNCH          /IDEA OF MULTI 8.
         0000  *0
               INTDEF,
00000    0000  RK8E,   0
00001    0000  KM8E,   0
               /MULTIP,         0                /MULTIPLEXER CLOCK.
00002    0000  PDP8IE, 0;0      /TWO DEVICES !!!!!!!!!!!
00003    0000
               /PDP11, 0;0
               /RF08,  0
00004    0000  DK8EP,  0
00005    0000  TT1,    0
00006    0000  KB1,    0
00007    0000  TT2,    0
00010    0000  KB2,    0
00011    0000  TT3,    0
00012    0000  KB3,    0
               MAXDEV,
               ENPUNCH
         0370  IFDEF TT2 <HCT=6420-6030>
         0430  IFDEF TT3 <HP=6460-6030>

         6041  IFDEF TT1 <TSF1=TSF;TCF1=TCF;TPC1=TPC;TLS1=TLS>
         6042
         6044
         6046
         6031  IFDEF KB1 <KSF1=KSF;KCC1=KCC;KRS1=KRS;KRB1=KRB>
         6032
         6034
```

```
      6036

      6431  IFDEF TT2 <TSF2=TSF+HCT;TCF2=TCF+HCT;TPC2=TPC+HCT;TLS2=TLS+HCT>
      6432
      6434
      6436
      6421  IFDEF KB2 <KSF2=KSF+HCT;KCC2=KCC+HCT;KRS2=KRS+HCT;KRB2=KRB+HCT>
      6422
      6424
      6426

      6471  IFDEF TT3 <TSF3=TSF+HP;TCF3=TCF+HP;TPC3=TPC+HP;TLS3=TLS+HP>
      6472
      6474
      6476
      6461  IFDEF KB3 <KSF3=KSF+HP;KCC3=KCC+HP;KRS3=KRS+HP;KRB3=KRB+HP>
      6462
      6464
      6466

            /SYSTEM STATISTICS WANTED?
      0001  STATX=1
            /EXTRA ERROR CHECK WANTED?
      0001  CHECK=1

            IFDEF STATX <
            /SET VALUE FOR MEASURING TIME IN IDLELOOP.
            DECIMAL
      6331  IFDEF PDP8E < TCKLEN=3289        /ABOUT 100.000/30.4 FOR PDP8/E>
            IFDEF PDP8I < TCKLEN=2777        /ABOUT 100.000/36   FOR PDP8/I>
                                       /USED FOR IDLETIME STATISTICS.
            OCTAL
            >

            /SELECT SYSTEM DISK.
      0000  SYDISK=RK8E
            /SYDISK=RF08
            /SELECT SYSTEM CLOCK.
      0004  SYTIME=DK8EP
            /SYTIME=MULTIP

            /CONFIGURATE LOGICAL UNIT 0 OF SYSTEM DISK.
      4000  VFBLOK=4000      /START BLOCK OF VFLDSLOTS.
            IFNZRO VFBLOK^17 <ASS ERR! VFBLOK MUST BE MULTIPLE OF 20>
      6000  TSBLOK=VIRMAX+1↑20+VFBLOK        /STARTING BLOK OF TASK LIBRARY.

            /SOME STUFF FOR PROGRAMS USING THE TASKLIBRARY.
            /NOT REQUIRED FOR MC8!!!!!!!!!!!!
            /ASSUME DIRECTORY SWAPPED IN; TSBLOK+1 STARTS AT LOC200.
      0200  DCDIR=200
      0400  NRT=DCDIR+200
      5000  FDCMAX=5000                /FIRST FREE LOC AFTER DIRECTORIES.
```

```
/APP. A1.   MC8.1. PAGE 0, INTR SECTION, SCHEDULER.


/***** M C 8 *****
/MULTICORE 8.
/R VAN VLIET MATH' CENTR' A'DAM.
/DATE 2 DEC 1975.

/****************
/
/OPERATING SYSTEM FOR D.E.C. PDP8.
/
/A PDP8 HAVING UPTO 64 VIRTUAL MEMORYFIELDS IS IMPLEMENTED.
/THE SYSTEM REQUIRES AT LEAST 16K MEMORY AND A REASONABLY
/FAST DISK.
/
/****************
```

```
        /CONVENTIONS AND ABREVIATIONS.

        /       /!      DEAF.
        /               SECTIONS ACCESSIBLE ONLY FROM STATE C OR
        /               FROM STATE D.
        /       /\      CRITICAL SECTION OF STATE B ACCESSING
        /               VARIABLES COMMON WITH STATE D; DEAF.
        /       /!\     COMMON SBR OF STATES B AND D; DEAF.
        /       /-      STATE CHANGING SECTION; DEAF.
        /               NOTE: SOME ROUTINES ARE ACCESSED BY STATE C
        /               (DEAF) AND B (SOMETIMES DEAF). THIS CANNOT CAUSE
        /               TROUBLES AS THESE STATES DONT INTERRUPT EACH
        /               OTHER. THESE ROUTINES ARE NOT ALLOWED TO TOUCH TH
E
        /               INTERRUPT SYSTEM. THEY ARE NOT MARKED!!!
        /       [N]     ARRAY INDEX.
        /               ARRAY INDICES COUNT FROM 0.
        /               ARR[N] INDICATES THE NTH ENTRY IN ARRAY ARR.
        /       [N,M]   INDEX IN 2-DIMENSIONAL ARRAY.
        /       M (N) CYCLES.
        /               THE LENGTH OF SOME ROUTINES IS INDICATED IN
        /               MEMORYCYLCES.
        /               M REFERS TO PDP8/E, N REFERS TO PDP8/I.
        /       ASSXN   HELP REGISTERS ASSX1, ASSX2... ARE USED DURING
        /               ASSEMBLY.

        /       TSK     TASK
        /       TSKSW   TASKSWITCHING
        /       Q       QUEUE
        /       CHN     CHAIN
        /       TAB     TABLE
        /       AC      ACCUMULATOR
        /       IF      INSTRUCTIONFILED
        /       DF      DATAFIELD
        /       L       LINK
        /       FL      FLAGS. PROCESSOR STATUS (PDP8E).
        /       MQ      MULTIPLIER QUOTIENT. A REGISTER.
        /       EAE     EXTENDED ARITHMETIC ELEMENT (PROCESSOR OPTION).
        /       SC      STEPCOUNT.
        /       INTR    INTERRUPT
        /       INT     INTERRUPT
        /       LOC     LOCATION
        /       W       WORD
        /       PT      POINT
        /       CT      COUNT
        /       FLD     FIELD
        /       VFLD    VIRTUAL FIELD
        /       MON     MONITOR
        /       ARG     ARGUMNENT
        /       SBR     SUBROUTINE.
        /       PARAM   PARAMETER
        /       OPT     OPTION
        /       FUNC    FUNCTION
        /       MS      MESSAGE
        /       MSS     MESSAGES
        /       RP      REPORT
```

```
                /         RCQ      RECEIVE QUEUE (MSS SENT TO A TSK GO THERE)
                /         RPQ      REPORT QUEUE (REPORTED MSS GO THERE)
                /         TCB      TASK CONTROLBLOCK
                /         REQ      REQUEST
                /         RTN      RETURN
                /         SCED     SCHEDULE
                /         SCD      SCHEDULE
                /         INH      INHIBITED
                /         STL      STATIC TASK LIST
                /         GPR      GENERAL PAGE 0 ROUTINES (SYSTEM PAGE 0).
                /         STSK#    STATIC TASK NUMBER
                /         PRIO     PRIORITY
```

```


/*** PAGE 0 ****
/
/PAGE 0 OF EACH VFLD IN WHICH NONPROTECTED TASKS ARE RUNNING
/IS DIVIDED IN TWO PARTS:
/       THE COMMON PART (LOC 14/147)
/       AND THE ONE TASK ONLY (=OTO) PART (LOC0-13, LOC150-177).
/THE COMMON PART CONTAINS:
/A.     GENERAL CONSTANTS
/B.     AN ENTRYPOINT FOR THE MONITOR (USED IN MONITOR CALLS)
/C.     THE FOLLOWING ROUTINES
/               VERHLT  (EMERGENCY ERROR. STOP OPERATION AND TAKE
/                       APPROPRIATE ACTION)
/               VVCDF   (DO VIRTUAL CHANGE DATA FIELD)
/               VVRDF   (VIRTUAL READ DATAFIELD)
/               VVRCDF  (DO CHANGE DATAFIELD TO REQUESTED VFLD)
/               VCDIF   (CHANGE DATAFIELD TO THIS VFLD)
/               VMSNOD  (REQUEST OR FREE A MESSAGE)
/               VGET    (FETCH THE CONTENTS OF THE LOC POINTED AT
/                       BY THE ARGUMENT, USING BASE AS OFFSET;
/                       X POINTS TO THAT LOC AFTERWARDS)
/               VPUT    (STORE THE CONTENTS OF AC INTO THE LOC
/                       POINTED AT BY THE ARGUMENT, USING BASE AS
/                       OFFSET;
/                       X POINTS TO THE LOC AFTERWARDS)
/               VJUMS   (CALL SUBROUTINE WHOSE NAME IS IN AC;
/                       STORE SBRLINK IN LOC POINTED AT BY THE
/                       FIRST LOC OF THE SBR, USING BASE AS
/                       OFFSET;
/                       X POINTS TO THAT LOC AFTERWARDS).
/D.     THE GENERAL PAGE 0 REGISTERS BASE AND X AND
/               PERHAPS MQ.
/
/THE OTO PART IS USED BY ONLY ONE TASK AT A TIME.
/A TASK CAN GET CONTROL OF THE OTO PART BY SETTING UP THE "ZERO-
/REQUESTBIT" IN STL[TSK, 1] BEFORE IT IS SWAPPED IN (USUALLY AT
/ASSEMBLY TIME). IT IS NOT ALLOWED TO HAVE CODE IN THE LOWER LOCS
/OF THE OTO PART.
/
/THE OTC PART OF VFLD 0 IS USED BY THE MONITOR.
/
/HERE FCLLOWS THE CODE OF VFLD 0 PAGE 0.
```

```
        0000  FIELD 0
        0000  *0        /INTERRUPT GOES HERE.
00000   0000  INTPC,  0                 /!SAVE PC DURING INTERRUPT.
              IFDEF PDP8E <
00001   6000  X6000,  SKON              /!?
00002   5177          JMP     INTR      /!
        7002  BSWR=BSW
              >
              IFNDEF PDP8E <
                      JMP     INTR      /!
              BSWR=JMS I .
                      SBYTER
              >
00003   7402  HLTINS, HLT               /SOFT WARE ERROR!!!
              /SYSHLT=JMS HLTINS-1

00004   0000  MONAC,  0                 /AC DURING MONITOR CALL.
00005   0000  INTAC,  0                 /SAVE AC DURING INTR
00006   0000  INTFL,  0                 /INTFLAGS
00007   0000  MONFL,  0                 /FLAGS DURING MONITOR CALL.
              IFNZRO MONFL-MONAC-3 <ASS ERROR!!! MONAC-MONFL SHOULD BE 3>


        0010  *10                       /AUTO INDEXES.
                                        /INITIALISATION CODE HERE!
00010   6213  X0,     CIF CDF 10        /USED BY MONITOR.
00011   5412  X1,     JMP I   .+1       /USED BY MONITOR
00012   0177  X2,     INIT-1            /!USED BY DEAF ROUTINES.
        0200  INIT=200          /ENTRYPOINT OF INITIALISATION IN FLD 1.
00013   0000  X3,     0                 /!USED BY DEAF ROUTINES.

```

```
              /COMMON PART,
              /       COPIED TO EACH VFLD IN WHICH NONPROTECTED TASKS ARE
              /RUNNING. THE PRESENCE OF THIS PART IN A VFLD IS INDICATED BY THE
              /MONINBIT IN THE FLDTAB,
              /
              /
        0014  *14
        0000  CUR=0           /CURRENT FIELD. MUST BE ADAPTED WHEN COPIED TO
                              /OTHER FLDS,

              /REQUEST OR FREE A MESSAGE.
              /NOTE: DISTURBS LINK, DF:=IF!!!
              /REQUEST:
              /       CLA
              /       MSREQ   /AC PTS AT MS[2] AT RETURN.
              /FREE:
              /       TAD     MSPTR   /AC PTS AT MS[2] (NEVER ZERO!)
              /       MSFREE  /AC = CONTENTS OF MS[2] AT RETURN.
        4014  MSREQ=JMS .
        4014  MSFREE=JMS .
00014   0000  VMSNOD, 0
00015   6002          IOF             /!ENTER STATE C,
00016   6203  XCDIF,  CIF CDF  0              /!
00017   5514          JMP I   XMSNOD1 /!RETURNS IN TASK.


              /MINIMUM TIME CDF ROUTINE FOR CDF TO REQUESTED VFLD,
              /CALLING SEQUENCE:
              /       VRCDF
              /IF THE REQUESTED VFLD IS IN CORE WE JUST DO
              /THE CDF, OTHERWISE CONTROL IS TRANSFERRED TO THE MONITOR,
              /IN THE LATTER CASE THE TSK WILL EVENTUALLY BE RESCHEDULED
              /WITH THE CORRECT DF,
              /
        4020  VRCDF=  JMS .
              /6 (10) CYCLES,
00020   0000  VVRCDF, 0               /TEMP FOR VVCDF,
00021   6202  MYCIF,  CIF CUR         /!ENTERING STATE C,
00022   4054          JMS     VCALL   /!OVERWRITTEN BY CDF,
                                      /MONITOR WILL RECOGNISE THIS CALL,
              /       JMP I   VVRCDF

              /HALT ON ERROR ROUTINE,
              /CALLING SEQUENCE:
              /       ERHLT   /STOP OPERATION; EMERGENCY ERROR!
              /
        4023  ERHLT=  JMS .
00023   5420  VERHLT, JMP I   VVRCDF  /MUST BE RESTORED AFTER EACH CALL!!
00024   4054          JMS     VCALL   /-MON WILL RECOGNISE THIS SPECIAL CALL,

              /VIRTUAL CHANGE DATAFIELD ROUTINE.
              /CALLING SEQUENCE:
              /       VCDF
              /               NN      /6-BIT VFLD#,
              /
```

```
                    /IF THE DESIRED VFLD IS ACTUALLY IN CORE THE CDF IS EXECUTED
                    /IMMEDIATELY, OTHERWISE CONTROL IS TRANSFERRED TO THE MONITOR AND
                    /THE TSK WILL BE RESCHEDULED EVENTUALLY HAVING THE CORRECT
                    /DATAFIELD.
              4025  VCDF=   JMS .
                    /28 (34) CYCLES.
                    /
      00025   0000  VVCDF,  0
      00026   6203          CIF CDF CUR     /!ENTER STATE C.
      00027   3020          DCA     VVRCDF  /!SAVE AC FOR A WHILE.
      00030   1425          TAD I   VVCDF   /!GET VFLD#
      00031   2025          ISZ     VVCDF   /!
      00032   1112          TAD     XFLDTAB /!
      00033   3042          DCA     VCDFTM  /!
      00034   6201  CCDF,   CDF 0           /!
      00035   1442          TAD I   VCDFTM  /!
      00036   7500          SMA             /!VFLD IN CORE NOW?
      00037   4054  XVCALL, JMS     VCALL   /-MON WILL RECOGNISE THIS SPECIAL CALL.
      00040   0117          AND     XCDF70  /!YES. GET FLD.
      00041   3042          DCA     .+1     /!
      00042   7402  VCDFTM, HLT             /!EXECUTE THE CDF.
      00043   1020          TAD     VVRCDF  /!
      00044   5425          JMP I   VVCDF

                    /VIRTUAL READ DATAFIELD ROUTINE.
                    /CALLING SEQUENCE:
                    /       CLA
                    /       VRDF
                    /       ...             /RETURNS WITH 6-BIT VFLD# IN AC.
                    /DF!=IF!!!!
              4045  VRDF=   JMS .
      00045   0000  VVRDF,  0
      00046   6002          IOF             /!ENTER STATE C.
      00047   6202          CIF 0           /!
      00050   5513          JMP I   XVRDF1  /!WILL RETURN IN TASK.

                    /NOTE! IN FLD 0 THIS ROUTINE IS USED TO SET THE DATAFLD TO THE
                    /USER'S IF.
              0051  CDFUF=.
                    /CHANGE DATAFIELD TO INSTRUCTIONFIELD.
              4051  CDFCUR= JMS .
      00051   0000  VCDIF,  0
      00052   6203  MYCDIF, CIF CDF CUR     /!ENTER STATE C.
      00053   5451          JMP I   VCDIF

                    /MONITOR ENTRYPOINT.
                    /
                    /CALLING SEQUENCE:
                    /       TAD     ACARG   /GET ARG IN AC IF ANY.
                    /       CALL
                    /               FUNCTION/FUNCTION SPECIFICATION.
                    /               ARGS    /ARGUMENTS, IF ANY.
                    /       JMP     ERROR   /POSSIBLE ERRORRETURN.
                    /       ...             /NORMAL RETURN, IF ANY.
```

```
         4054  CALL=   JMS .
               MONPC,                     /PC OF CALLING PROGRAM.
  00054  0000  VCALL,  0
  00055  6202          CIF 0              /-
  00056  6002          IOF                /-
  00057  5511          JMP I    XMONIT    /-
```

```
             /COMMON PART.
             /REENTRANT TASK SUPPORT.

             /GET ROUTINE.
             /CALLING SEQUENCE:
             /       GET
             /               PTR      /VIRTUAL PTR.
             /                        /AC :=CONTENTS OF POINTED LOC,
             /                        X POINTS TO LOC AT RETURN.
             /USES BASE AND X.

       4060  GET=    JMS .
             /17 (19) CYCLES.
00060  0000  VGET,   0
00061  6203          CIF CDF CUR      /!ENTER STATE C.
00062  1107          TAD     BASE     /!
00063  1460          TAD I   VGET     /!
00064  3110          DCA     X        /!
00065  1510          TAD I   X        /!
00066  2060          ISZ     VGET     /!
00067  5460          JMP I   VGET

             /SUBROUTINE CALLING ROUTINE.
             /CALLING SEQUENCE:
             /       CLA              /ARG GOES IN MQ
             /       TAD     SUBNAME /AC PTS TO ROUTINE.
             /       JUMS             /
             /NOTE:  THE FIRST LOC OF THE ROUTINE MUST CONTAIN THE ENTRY
             /       OF THE REENTRANCY ARRAY WHERE THE SBRLINK MUST BE STORED.
             /       USES BASE AND CHANGES X (POINTS TO LOC AFTERWARDS).
       4070  JUMS=   JMS      .
             /25 (29) CYCLES.
00070  0000  VJUMS,  0                /LUCKY. THIS IS A TEMP FOR VPUT!!!
00071  6203          CIF CDF CUR      /!ENTER STATE C.
00072  3075          DCA     VPUT     /!AS IF VPUT WERE CALLED TO STORE THE
00073  1070          TAD     VJUMS    /!SBRLINK, PREPARE ENTERING VPUT.
00074  7410          SKP              /!JUMP INTO VPUT. IT WILL DO A LUCKY
                                      /!RETURN.

             /PUT ROUTINE.
             /CALLING SEQUENCE:
             /       TAD     VAL      /GET VALUE TO BE STORED.
             /       PUT
             /               PTR      /VIRTUAL POINTER.
             /                        /STORE VALUE IN THE POINTED LOC,
             /                        X POINTS TO THE LOC AT RETURN.

       4075  PUT=    JMS .
             /21 (23) CYCLES.
             /JUMS ROUTINE JUMPS INTO IT!!!!!.
00075  0000  VPUT,   0
00076  6203          CIF CDF CUR      /!ENTER STATE C.
00077  3070          DCA     VJUMS    /!TEMP. MUST BE THAT ONE!!!!
00100  1475          TAD I   VPUT     /!
00101  1107          TAD     BASE     /!
```

```
00102  3110          DCA     X        /!
00103  1070          TAD     VJUMS    /!
00104  3510          DCA I   X        /!
00105  2075          ISZ     VPUT     /!
00106  5475          JMP I   VPUT


             IFZERO VPUT+1-100^4000 <ASS ERR!>

             /PAGE 0 REGISTERS.
00107  0000  BASE,   0
00110  0000  X,      0
             IFNDEF HARDMQ < MQ,      0        /IN CASE THERE IS NO EAE>

             /SOME PTRS.
00111  3000  XMONIT, MONITOR
00112  4600  XFLDTAB,FLDTAB
00113  2671  XVRDF1, VRDF1
00114  2717  XMSNOD1,MSNOD1
       4515  IEXIT=JMS I .
00115  0402          RPINTR           /PTR FOR CONNECTED ROUTINES.
00116  4400  ZSTL,   STLST            /POINTER TO STL.
                                      /USED TO RETURN ENTRIES IN STL.
00117  6271  XCDF70, CDF 70
```

```
              /COMMON PART.
              /GENERAL CONSTANTS.

              M7776,
00120  0002   C2,       2
              M7775,
00121  0003   C3,       3
              M7774,
00122  0004   C4,       4
              M7771,
00123  0007   C7,       7
00124  0010   C10,      10
00125  0037   C37,      37
00126  0040   C40,      40
00127  0070   C70,      70
00130  0077   C77,      77
00131  0100   C100,     100
00132  0177   C177,     177
00133  0200   C200,     200
00134  0377   C377,     377

              M1,
00135  7777   C7777,    7777
              M2,
00136  7776   C7776,    7776
              M3,
00137  7775   C7775,    7775
              M4,
00140  7774   C7774,    7774
              M5,
00141  7773   C7773,    7773
              M6,
00142  7772   C7772,    7772
              M10,
00143  7770   C7770,    7770
              M20,
00144  7760   C7760,    7760
              M100,
00145  7700   C7700,    7700
              M200,
00146  7600   C7600,    7600

              IFNZRO 150-.^4000 <ASS ERR!>
```

```
                     /ONE TASK ONLY PART.

                     /FLD 0: MONITOR ROUTINES AND CONSTANTS.
              0150   *150


                     /4 TEMPORARIES FOR MONITOR.
       00150  0000   MON0,    0
       00151  0000   MON1,    0
       00152  0000   MON2,    0
       00153  0000   MON3,    0

                     /TEMPORARIES FOR INTERRUPT SECTION AND DEAF ROUTINES.
       00154  0000   INT0,    0                 /!
       00155  0000   INT1,    0                 /!
       00156  0000   INT2,    0                 /!
       00157  0000   INT3,    0                 /!

                     /MONITOR CALL PARAMETERS.
       00160  0000   MONFUNC,0                  /MONITOR FUNCTION WORD
       00161  0000   MARG1,   0                 /MONITOR ARG1
       00162  0000   MARG2,   0                 /MONITOR ARG2.

       00163  0000   VCURIF, 0                  /CURRENT VINSTRFLD IN B0-5.
       00164  0000   CURTSK, 0                  /PTR TO TCB[5] OF CURRENT TSK.
       00165  4700   HPRIO,   RQPTRS            /HIGHEST RUNNABLE PRIORITY
       00166  0000   SCDREQ, 0                  /SCHEDULE REQUEST FLAG.
                                      /0: SCDREQ PENDING, OTHER: NO SCDREQ PENDING.
       00167  0000   SCDINH, 0                  /SCHEDULE INHIBIT FLAG.
                                                /0: SCHEDULING ALLOWED
                                                />=1: SCHEDULING INHIBITED.
                     /NOTE!  WHEN SCDINH IS CLEARED SCDREQ MUST ALWAYS
                     /BE CHECKED!

       00170  4722   ZCORMAP,CORMAP
       00171  1000   ZSNDREP,SNDREP             /PTR TO SNDREP ROUTINE.
       00172  4305   ZFN,     FN                /PTR TO FN ROUTINE.
       00173  1000   Z1000,   1000

              0000   AVPT=0                     /INITIALISE AVAILLIST BUILDER.

                     IFNZRO 176-.^4000 <ASS ERR!>

                     /***** END OF PAGE 0 *****
              0176   *176
              4576   SYSHLT=JMS I  .
       00176  0002          HLTINS-1
```

```
                /***** INTERRUPT SECTION *****

        0177    *177
                /SAVE STATUS.
                IFDEF PDP8E <
00177   3005    INTR,   DCA     INTAC   /!
                /       GTF             /!
        6004    INS200=GTF              /THIS INSTRUCTION WILL BE STORED HERE BY
                                        /INIT.
00200   5010            JMP     X0      /TO START UP INIT. GETS OVERWRITTEN.
                /ASSERTION: BIT2=1, BIT3-4=0. WE DONT USE BIT2.
                /       AND     C6177   /! REMOVE DIRTY FLAGS.
00201   3006            DCA     INTFL   /!
                >
                IFDEF PDP8I <
                INTR,   DCA     INTAC   /!
                /       RAR             /!
                INS200=RAR
                        JMP     X0      /!TO START UP INIT. GETS OVERWRITTEN.
                        RIB             /!/READ SAVE FIELD REGISTER.
                        DCA     INTFL   /!
                >
                INTR2,                  /!COME HERE IF ANOTHER FLAG IS PENDING.

                IFDEF STATX <
00202   2301            ISZ     INTCT   /!
00203   5207            JMP     SKPCHN  /!
00204   2302            ISZ     INTCT+1 /!
00205   5207            JMP     SKPCHN  /!
00206   2303            ISZ     INTCT+2 /!OVERFLOW EACH 2**14 HOURS.
                /       JMP     SKPCHN  /!
                >

                /ALLOCATE THE SKIPCHAIN.
                IFNDEF SKPCHN <
                SKPCHN=.                /IF SKIPCHAIN FITS ON CURRENT PAGE.

                SKPEND=MAXDEV↑5+.+2 /END OF SKIPCHAIN.
                IFDEF CHECK <SKPEND=SKPEND+1> /1 LOC FOR EXTRA CHECK.
                SKPFIT=400 /SKIPCHAIN MUST FIT BEFORE 400.
                IFDEF STATX <SKPFIT=SKPFIT-3> /RESERVE 3 LOCS FOR COUNTER.
                NONFIT= SKPFIT-SKPEND^4000

                IFNZRO NONFIT < /IF SKIPCHAIN EXCEEDS CURRENT PAGE
                IFDEF STATX <SKPCHN=SKPCHN+4 > /A JMP AND 3 COUNTERS.

                /PROPER ALLOCATION FOR PAGE CROSSING.
                /FORCE AN INTSLOT AT LOC 377
                QUOT5=377-SKPCHN%5↑5 /5 LOCS FOR EACH INTSLOT.
                SKPSHF=377-QUOT5-SKPCHN
                SKPCHN=SKPCHN+SKPSHF
                >
                >

                IFNZRO NONFIT <
        IFNZRO SKPCHN-. <JMP SKPCHN>
        IFDEF STATX <INTCT, 0;0;0>
        >




        /***** SKIPCHAIN *****
        /HERE COMES THE SKIPCHAIN. IT IS A LIST OF
        /SOCALLED INTSLOTS. AN INTSLOT CONSISTS OF 5 CONSECUTIVE LOCS.
        /ON THIS PAGE THERE IS ROOM FOR ABOUT 22 INTSLOTS. IF ONE NEEDS
        /MORE, THE SKIPCHAIN MUST BE CAREFULLY ALLOCATED TO
        /CROSS THE PAGEBOUND PROPERLY.
        /THE ORDER OF THE SLOTS IN THE SKPCHN IS DETERMINED
        /IN THE CONFIGURATIONSECTION AT THE LABEL INTDEF.
        /
        /THE FIRST TWO LOCS OF AN INTSLOT CONTAIN:
        /       SKIPIOT; JMP .+4
        /THE CONTENTS OF THE LAST THREE LOCS IS CONTROLLED BY THE
        /FOLLOWING MONITOR FUNCS:
        /CLINTR (CLAIM INTSLOT).
        /       CLEARIOT
        /       IEXIT
        /               MSPTR
        /       A TSK CAN GET CONTROL OF AN INTSLOT BY EXECUTING THIS
        /       MONFUNC. A MS IS RESERVED WHOSE PTR IS DENOTED IN THE
        /       INTSLOT. EACH TIME AN INT OCCURS THIS MS IS REPORTED TO
        /       THE TSK. NO TWO TSKS MAY LOOK AT THE SAME INTSLOT.
        /FRINTR (FREE INTSLOT).
        /       CLEARIOT
        /       IEXIT
        /               0
        /       SET INTSLOT IN DISCARD MODE. CAN BE USED AFTER A CONNECT,
        /       CLAIM OR DISABLE FUNCTION. IF THE CORRESPONDING DEVICE
        /       INTERRUPTS THE CLEARIOT IS EXECUTED, AC IS CLEARED AND
        /       THE FAST EXIT IS TAKEN.
        /DSINTR (DISABLE INTSLOT).
        /       JMP     .+3
        /       0
        /       0
        /       IF A TSK DISABLES THE INTERRUPTFUNCTION OF SOME DEVICE,
        /       IT MUST DISABLE THE INTSLOT AS WELL. IF NOT, GREAT
        /       PROBLEMS WILL ARISE IN TREATING DEVICES CORRECTLY. WHEN
        /       THE INTSLOT IS DISABLED FLAGS OF THAT DEVICE WILL BE
        /       IGNORED. NOTE: IF A ROUTINE IS CONNECTED TO THE INTSLOT,
        /       THAT ROUT'NE MAY IGNORE THE DEVICEFLAG BY TAKING A SIMPLE
        /       RETURN. IN THAT CASE SEARCHING THE SKIPCHAIN IS
        /       CONTINUED.
        /CNINTR (CONNECT TO INTSLOT).
        /       CIF CDF FLD
        /       JMS I   .+1
        /       SBRNAME
        /       CONNECT A DEAF ROUTINE TO THE INTSLOT.
        /       EACH TIME THE DEVICEFLAG IS ENCOUNTERED IN THE SKPCHN,
        /       THAT  ROUTINE IS CALLED. IF THE DEVICE INTERRUPT FUNCTION
```

```
/        WAS DISABLED, IT MUST IMMEDIATELY RETURN FROM THAT
/        ROUTINE, THUS ENABLING CORRECT TREATMENT OF OTHER
/        DEVICES. (THE SBRLINK MUST BE INCREMENTED ONCE BEFORE
/        RETURNING!!!).
/        NORMALLY A CONNECTED ROUTINE IS TERMINATED BY EXECUTING
/        THE IEXIT INSTRUCTION, SPECIFYING A MS TO BE REPORTED
/        IN THE NEXT LOC. IF NO MS NEED BE REPORTED, 0000 IS
/        SPECIFIED.
/        NOTE: CONNECTED ROUTINES MUST ALWAYS BE IN CORE, IT IS
/        NOT ALLOWED TO TOUCH THE ION IN A CONNECTED ROUTINE.
/
/MOST INTSLOTS ARE INITIALLY EITHER FREE OR DISABLED, DEPENDING
/ON THE DEFAULT ATTITUDE OF THE DEVICE.
/SOME ARE ALREADY CLAIMED BY OR CONNECTED TO MONTSKS.
```

```
              IFDEF TT1 <
       0240   *TT1↑5+SKPCHN
00240  6041           TSF                /!
00241  5245           JMP      .+4       /!
00242  6042           TCF                /!
00243  4515           IEXIT              /!
00244  0000                    0         /!
              >

              IFDEF KB1 <
       0245   *KB1↑5+SKPCHN
00245  6031           KSF                /!
00246  5252           JMP      .+4       /!
00247  6036           KRB                /!PERHAPS A PITY FOR THE PDP8/I
00250  4515           IEXIT              /!
00251  0000                    0         /!
              >

              IFDEF TT2 <
       0252   *TT2↑5+SKPCHN
00252  6431           TSF2               /!
00253  5257           JMP      .+4       /!
00254  6432           TCF2               /!
00255  4515           IEXIT              /!
00256  0000                    0         /!
              >

              IFDEF KB2 <
       0257   *KB2↑5+SKPCHN
00257  6421           KSF2               /!
00260  5264           JMP      .+4       /!
00261  6426           KRB2               /!
00262  4515           IEXIT              /!
00263  0000                    0         /!
              >

              IFDEF TT3 <
       0264   *TT3↑5+SKPCHN
00264  6471           TSF3               /!
00265  5271           JMP      .+4       /!
00266  6472           TCF3               /!
00267  4515           IEXIT              /!
00270  0000                    0         /!
              >

              IFDEF KB3 <
       0271   *KB3↑5+SKPCHN
00271  6461           KSF3               /!
00272  5276           JMP      .+4       /!
00273  6466           KRB3               /!
00274  4515           IEXIT              /!
00275  0000                    0         /!
              >

              IFDEF DTA <
```

```
                *DTA↑5+SKPCHN
                        DTSF                /!
                        JMP       .+4       /!
                        JMP       .+3       /!
                        0                   /!
                        0                   /!
                >

                IFDEF ANALEX <
                *ANALEX↑5+SKPCHN
                        PRSF                /!
                        JMP       .+4       /!
                        JMP       .+3       /!
                        0                   /!
                        0                   /!
                >

                IFDEF RK8E <
         0207   *RK8E↑5+SKPCHN
00207    6741           DSKP                /!
00210    5214           JMP       .+4       /!
                IFNZRO SYDISK-RK8E <
                        JMP       .+3       /!DISABLED INITIALLY.
                        0                   /!
                        0                   /!
                >
                IFZERO SYDISK-RK8E <
00211    6203           CIF CDF 0           /!CONNECTED TO SYWAIT ROUTINE.
00212    4613           JMS I     .+1       /!CALL CONNECTED ROUTINE
00213    2333           RKINTR  /!
                >
                >

                IFDEF RF08 <
                *RF08↑5+SKPCHN
                        6623                /!
                        JMP       .+4       /!
                IFNZRO SYDISK-RF08 <
                        JMP       .+3       /!DISABLED INITIALLY.
                        0                   /!
                        0                   /!
                >
                IFZERO SYDISK-RF08 <
                        CIF CDF 0           /!CONNECTED ROUTINE.
                        JMS I     .+1       /!
                        RFINTR              /!
                >
                >

                IFDEF DK8EP <
         0233   *DK8EP↑5+SKPCHN
00233    6131           CLSK                /!
00234    5240           JMP       .+4       /!
                IFNZRO SYTIME-DK8EP <
                        JMP       .+3       /!DISABLED INITIALLY.
```

```
                      0               /1
                      0               /!
              >
              IFZERO SYTIME-DK8EP <
00235   6135          CLSA            /!CLAIMED BY MTTIME.
00236   4515          IEXIT           /!
00237   2666                  TIMEMS  /!
              >
              >

              IFDEF MULTIP <  /PDP-8/I MULTIPLEXER.
              *MULTIP↑5+SKPCHN
                      T1SKP           /!
                      JMP     .+4     /!
              IFNZRO SYTIME-MULTIP <
                      T1ON            /!A PITY!
                      IEXIT           /!
                              0       /!
              >
              IFZERO SYTIME-MULTIP <
                      CIF CDF 0       /!
                      JMS I   .+1     /!
                      MULINTR         /!
              >
              >

              IFDEF PDP8IE <  /MC PDP8/I-PDP8/E INTERFACE.
        0221  *PDP8IE↑5+SKPCHN              /THESE TWO DEVICES GO TOGETHER.
00221   6571          P8SIF           /!INPUT?
00222   5226          JMP     .+4     /!
00223   5226          JMP     .+3     /!DISABLED INITIALLY.
00224   0000          0               /!
00225   0000          0               /!
00226   6573          P8SOF   /!OUTPUT?
00227   5233          JMP     .+4     /!
00230   5233          JMP     .+3     /!DISABLED INITIALLY.
00231   0000          0               /!
00232   0000          0               /!
              >

              IFDEF PDP11 <   /MC PDP-8 PDP-11 INTERFACE.
              *PDP11↑5+SKPCHN         /TWO DEVICES!
                      P11SOF          /!OUTPUT?
                      JMP     .+4     /!
                      JMP     .+3     /!DISABLED INITIALLY
                      0               /!
                      0               /!
                      P11SIF          /!INPUT?
                      JMP     .+4     /!
                      JMP     .+3     /!DISABLED INITIALLY.
                      0               /!
                      0               /!
              >

              IFDEF FPTP <
```

```
                 *FPTP↑5+SKPCHN
                         FPSF                /!
                         JMP       .+4       /!
                         FPCR                /!
                         IEXIT               /!
                                   0         /!
                 >

                 IFDEF FPTR <
                 *FPTR↑5+SKPCHN
                         FRSF                /!
                         JMP       .+4       /!
                         FRCR                /!
                         IEXIT               /!
                                   0         /!
                 >

                 IFDEF KM8E <
           0214  *KM8E↑5+SKPCHN
    00214  6254          SINT                /!USER MODE.
    00215  5221          JMP       .+4       /!
    00216  6204          CINT                /!TYPICAL RESULT IF NOT CONNECTED.
    00217  4515          IEXIT               /!
    00220  0000                    0         /!
                 >

           0276  *MAXDEV↑5+SKPCHN
    00276  0000          0                   /!SIGNAL END OF SKPCHN; INNOCENT.
                 IFDEF CHECK <
    00277  6004  IFDEF PDP8E <GTF> /SHOW STATUS IN AC.
                 IFDEF PDP8I <RIS> /SHOW STATUS IN AC. (MC ONLY!)
                 >
    00300  4576          SYSHLT              /UNKNOWN INTERRUPT.

                 IFDEF STATX <
    00301  0000  IFZERO NONFIT <INTCT, 0;0;0>
    00302  0000
    00303  0000
                 >

                 IFNZRO .-1^177-175^4000 < /AT LEAST 3 FREE LOCS ON THIS PAGE
    00304  0000          AVPT;(0;AVPT=.-2
    00305  0377
           0304
                 >

    00377  0000
           0400  PAGE
```

```
00400  0000   IFDEF STATX <INTSCD, 0;0>
00401  0000

              /!EXIT ROUTINE. LEAVE INTERRUPT SECTION.
              /CALLED EITHER DIRECTLY FROM SKPCHN, OR FROM CONNECTED ROUTINES.
              /ARG1 =MSPTR. PTR TO MS[2].
              /STORE AC (POSSIBLE I/O STATUS) IN MS[2].
              /!F MSPTR=0 NO MS NEED BE REPORTED.
              /CALLED WITH DF TO CALLING FIELD.
00402  0000   RPINTR, 0                 /!POSSIBLE I/O STATUS IN AC.
00403  3155           DCA     INT1      /!
00404  1602           TAD I   RPINTR    /!
00405  6201           CDF 0             /!CONNECTED ROUTINES MIGHT CALL US FROM
                                        /!OTHER FIELDS.
00406  7450           SNA               /!IF NO MS ATTACHED, THE SLOT IS FREE
00407  5217           JMP     FSTEXT    /!AND WE JUMP DIRECTLY TO THE END.
00410  3154           DCA     INT0      /!PTR TO MS[2]
00411  1155           TAD     INT1      /!STATUS TO MS
00412  3554           DCA I   INT0      /!
00413  7344           ACM2              /!
00414  1154           TAD     INT0      /!
00415  3154           DCA     INT0      /!PTR TO MS[0] SENDER WORD.
                                        /!CANNOT BE 0 FOR INTERRUPT MS.
00416  4571           JMS I   ZSNDREP   /!REPORT.


              FSTEXT,
              IFDEF PDP8E <
00417  6003           SRQ               /!SAVING TAKES ABOUT 60 CYCLES. THIS
00420  7410           SKP     /!CHECK DECREASES RESPONSE TIME.
              >
              IFDEF PDP8I <
                      RIS               /!READ INTERRUPT STATUS. ONLY MC!!
                      SPA CLA           /!
              >
00421  5777'          JMP     INTR2     /!

       5776'  EXTPRV!=JMP MONEX
00422  0000   EXTALL, 0                 /!EXIT ALLOWED SWITCH.
                                        /CHANGED INTO JMP MONEX BY SIMINTR
                                        /!COMMAND.
00423  1166           TAD     SCDREQ    /!SCEDREQ PENDING
00424  1167           TAD     SCDINH    /!AND SCHEDULING NOT INHIBITED?
00425  7640           SZA CLA           /!
00426  5245           JMP     FSTXT2    /!NO. LEAVE INTR SECTION.
              /YES. CHECK HOW TO SAVE.

00427  1164           TAD     CURTSK    /!IF NO TASK RUNS WE SCHEDULE WITHOUT
00430  7650           SNA CLA           /!SAVING
00431  5775'          JMP     SCED      /!
              IFDEF STATX <   /INTSCD IS ONLY COUNTED IF SOME TSK IS
00432  2200           ISZ     INTSCD    /!/INTERRUPTED.
00433  5236           JMP     .+3       /!
00434  2201           ISZ     INTSCD+1/!
00435  7000           NOP               /!
```

```
                     >
00436  4051                  JMS     CDFUF   /!IF USERMODE SKP, ELSE CDF INSTRFLD,
00437  7410                  SKP             /!NO USERMODE.
00440  5264                  JMP     ITSAF   /!NO TROUBLES, SAVE.
00441  1000                  TAD     INTPC   /!MIND: DF NONZERO!!!
00442  1374                  TAD     (-VCALL-1        /!DONT SWITCH TSK IF PC=VCALL+1
00443  7640                  SZA CLA         /!FOR SAVING IS DIFFICULT AND SCDREQ WILL
                                             /!BE TESTED SOON ANYHOW.
00444  5252                  JMP     TSKSWT  /!NO; GO SWITCHING.

                     FSTXT2,
                     IFDEF PDP8E <
00445  1006                  TAD     INTFL   /!MIND: DF NONZERO!!!!
00446  6005                  RTF             /!
00447  7200                  CLA             /!
00450  1005                  TAD     INTAC   /!
00451  5400                  JMP I   INTPC   /!
                     >
                     IFDEF PDP8I <
                             TAD     INTFL   /!FETCH LINK.
                             RAL             /!
                             CLA             /!
                             TAD     INTAC   /!
                             RMF             /!
                             ION             /!
                             JMP I   INTPC   /!
                     >



                     /TASK SWITCHING.
00452  1000          TSKSWT, TAD     INTPC   /!INTERRUPTED IN ONE OF THE GENERAL
00453  0145                  AND     C7700   /!PAGE 0 ROUTINES?
00454  7640                  SZA CLA         /!
00455  5264                  JMP     ITSAF   /!GO FOR SIMPLE SAVE INTRSTATUS.

00456  7240                  CLA CMA         /!INTERRUPTED IN GENERAL PAGE 0 ROUTINE.
00457  1000                  TAD     INTPC   /!SAVE STATUS BEFORE CALL.
00460  3000                  DCA     INTPC   /!
00461  7240                  CLA CMA         /!MIND: DF STILL TO USERS' INSTRFLD!!!
00462  1400                  TAD I   INTPC   /!PC BEFORE CALL
00463  3000                  DCA     INTPC   /!
                     /FALL INTO SAVE ROUTINES.

```

```
/***** SCHEDULING SCHEME AND SAVING. *****
/
/THESE FOUR ARE THE IMPORTANT VARIABLES IN THE SCHEDULER:
/CURTSK POINTER TO TCB[5] (LINKWORD) OF THE TASK CURRENTLY
/RUNNING
/HPRIO  INDICATING THE HIGHEST RUNNABLE PRIORITY
/SCDREQ SCHEDULE REQUESTFLAG, 0: SCEDREQ PENDING
/SCDINH SCHEDULING INHIBITION FLAG, >=1: SCHEDULING INHIBITED.
/
/CURTSTK=0 IF NO TASK IS RUNNING (JUST AFTER INITIALIZATION,
/   WHILE  SCHEDULING AND PERHAPS WHILE IDLING).
/HPRIO INDICATES THE HIGHEST RUNNABLE PRIORITY BY POINTING IN ONE
/     OF THE PRIORITY RUNQS. IF NO TASKS ARE RUNNABLE IT POINTS
/    IN THE RUNQ CONTAINING THE IDLELOOP. IN GENERAL THE TASK
/    CURRENTLY RUNNING IS THE FIRST IN THE RUNQ POINTED AT BY
/    HPRIO.

/TASKS OF A HIGHER PRIORITY THAN THE CURRENT ONE CAN
/BECOME RUNNABLE AS A CONSEQUENCE OF AN INTERRUPT OR AS A
/CONSEQUENCE OF SOME MONITOR CALL (FOR INSTANCE A MESSAGE
/REPORT). THEREFORE BEFORE LEAVING THE INTERRUPT SECTION OR
/THE MONITOR WE CHECK THAT THE CURRENT TASK HAS STILL THE
/HIGHEST PRIORITY OF ALL RUNNABLE TASKS. IF IT HAS NOT, THE
/SYSTEM SWITCHES TASKS AS SOON AS POSSIBLE.
/AFTER A MONITOR CALL THE CURRENT TASK MAY NO LONGER BE
/RUNNABLE (FOR INSTANCE AFTER A SEND MESSAGE AND WAIT
/REPORT). THE STATUS IS SAVED AND THE TASK IS REMOVED FROM
/THE RUNQ.  SUBSEQUENT THE SCHEDULER IS ENTERED.
/
/***** SAVING.
/BEFORE A NEW TASK CAN BE RUN THE CURRENT TASKS' STATUS MUST BE
/SAVED. THE STATUS IS SAVED AS FOLLOWS:
/VIF+VDF            TCB[7]  VIRTUAL FLDS.
/AC                 TCB[8]  ACCUMULATOR.
/PC                 TCB[9]  PROGRAM COUNTER.
/MQ                 TCB[10] MULTIPLIER QUOTIENT.
/EAEL               TCB[11] EAE STATUS +LINK.
/BASE               TCB[12] VALUE OF BASE (NOT IN PROTECT TASKS).
/X                  TCB[13] VALUE OF X (NOT IN PROTECT TASKS).
/PROTECT TASKS ARE TASKS RUNNING WITH USER MODE ON.
/AT MONITOR ENTRANCE PART OF THE STATUS IS SAVED IN
/MONPC, MONAC AND MONFL (FLDS +L +SOME EAE). THIS IS USED IF
/SAVING IS REQUIRED AFTER THIS CALL.
/AT INTERRUPT PART OF THE STATUS IS SAVED IN
/INTPC, INTAC AND INTFL (FLDS +L +SOME EAE). THIS IS USED
/WHEN TASK SWITCHING OCCURS AFTER THE INTERRUPT.
/
/***** SCHEDULING.
/IT SHOULD BE CLEAR THAT WE ARE NOT ALLOWED TO SWITCH TASKS
/IF THE INTERRUPT CAME DURING A MONITOR CALL. TWO THINGS
/MIGHT GO WRONG: MONITOR CALLS OF DIFFERENT TASKS CAN INTERFERE,
/AND INTPC, INTAC AND INTFL DO NOT CONTAIN THE PROPER STATUS.
/THIS IS REMEDIED BY THE VARIABLE SCDINH. WHEN WE ENTER A
/CRITICAL SECTION (FOR INSTANCE AT MONITOR CALL) IN WHICH WE DONT
/LIKE TO BE INTERRUPTED BY OTHER TSKS WE SET SCDINH TO 1, THEREBY
```

```
/INHIBITING  SCHEDULING. AT THE END OF THE CRITICAL SECTION WE
/CLEAR SCDINH AND CHECK SCDREQ TO SEE WETHER A SCHEDULE REQUEST
/IS STILL PENDING.  IF IT IS WE SAVE THE CURRENT TSK AND ENTER
/THE SCHEDULER.
/TSKS ARE ENTERED IN THE RUNQ BY CALLING INRUNQ.
/THIS ROUTINE CHECKS  WETHER THE TSK INSERTED HAS A HIGHER PRIO
/THAN THE CURRENT TSK. IF SO IT CLEARS SCDREQ, SIGNALLING THE
/SCHEDULE REQUEST. AT THE END OF THE INTR SECTION AND WHEN
/LEAVING THE MONITOR THIS VARIABLE IS TESTED.
/
/AFTER SAVING WE INSPECT THE PRIORITY RUNQS ONE BY ONE
/STARTING AT THE ONE INDICATED BY HPRIO. IF NECESSARY
/HPRIO IS UPDATED. WHEN WE FIND A RUNNABLE TASK WE CHECK FOR ITS
/FLDS TO BE IN CORE. IF THEY ARE THE TASK IS STARTED, OTHERWISE
/WE ISSUE A SWAPREQ (SEE FCHECK AND MTSWAP)
/AND CONTINUE SEARCHING THE RUNQS TO FIND THE NEXT IMPORTANT
/TASK. IN THIS SITUATION WE MIGHT START A TASK THAT IS NOT THE
/FIRST IN ITS RUNQ.
/
/NOTE!!!: THREE PITFALLS:
/1.  WE DONT ISSUE A SWAP IF ANOTHER SWAPREQ WAS STILL PENDING.
/    IF WE DID THE SYSTEM MIGHT BE SWAPPING WITHOUT EVER RUNNING
/    A TASK.
/2.  THE INTERRUPT THAT INDICATES SWAP COMPLETION MAKES
/    THE MONITOR SWAP TASK RUNNABLE. THIS CAUSES HPRIO TO POINT
/    AT THE PRIO 0 RUNQ, THUS GUARANTEEING THAT ALL RUNQS ARE
/    REINSPECTED. THE TASK THAT CAUSED THE SWAP REQUEST IS
/    PROBABLY THE FIRST TO BENEFIT FROM THIS.
/3.  IF THE TASK WHOSE FLDS ARE NOT IN CORE IS REENTRANT
/    (BIT1 OF TCB[6] SET), WE SKIP OVER ALL THE REST OF THAT
/    PRIO RUNQ. IF WE DIDNOT DIFFERENT VERSIONS OF THAT
/    REENTRANT TASK MIGHT INTERRUPT ONE ANOTHER AT INPREDICTABLE
/    MOMENTS.


```

```
                     /SAVE AFTER INTERRUPT.
00464  6201          ITSAF,  CDF 0               /!MAKE SURE ABOUT DF.
00465  1006                  TAD     INTFL       /!COPY INTR STATUS INTO MONTIORSTATUS.
00466  3007                  DCA     MONFL       /!
00467  1005                  TAD     INTAC       /!
00470  3004                  DCA     MONAC       /!
00471  1000                  TAD     INTPC       /!
00472  3054                  DCA     MONPC       /!
00473  5300                  JMP     CMNSAF      /!JUMP INTO COMMON SAFE.


                     MONOUT,                     /SAVE CURTSK AND SCHEDULE.
                     IFDEF STATX <
00474  2343                  ISZ     MONSCD
00475  5300                  JMP     .+3
00476  2344                  ISZ     MONSCD+1
00477  7000                  NOP
                     >
                     CMNSAF,                     /COMMON SAFE ROUTINE.
00500  1164          MONSAF, TAD     CURTSK      /!\?SCDINH>=1, OR DEAF.
00501  7001                  IAC                 /!\?
00502  3010                  DCA     X0          /!\?
00503  1007                  TAD     MONFL       /!\?SAFE STATUS AFTER MONCALL.
00504  0123                  AND     C7          /!\?
00505  1170                  TAD     ZCORMAP     /!\?PTR TO ACTUAL FLD.
00506  3150                  DCA     MONO        /!\?
00507  1550                  TAD I   MONO        /!\?GET VFLD#
00510  0130          MONSF2, AND     C77         /!\?SAFE AFTER FCHECK JUMPS HERE.
00511  1163                  TAD     VCURIF      /!\?ADD VINSTRFLD
00512  3410                  DCA I   X0          /!\?
00513  1004                  TAD     MONAC       /!\?
00514  3410                  DCA I   X0          /!\?
00515  1054                  TAD     MONPC       /!\?
00516  3410                  DCA I   X0          /!\?
                     IFDEF PDP8E <
                     IFDEF EAE   <
00517  7441                  SCA                 /!\?
00520  7521                  SWP                 /!\?SC TO MQ
00521  3410                  DCA I   X0          /!\?MQ
00522  7403                  SCL                 /!\?SKIPS IF MODE A
00523  1133                  TAD     C200        /!\?
00524  1007                  TAD     MONFL       /!\?NOTE: 200 NOT IN FLAGS!.
00525  0145                  AND     C7700       /!\?
00526  7501                  MQA                 /!\?ORS STEPCOUNT IN LOW ORDER BITS
00527  3410                  DCA I   X0          /!\?
                     >
                     >
                     IFDEF PDP8I <
                     IFNDEF EAE  <
                             JMS     CDFUF       /!\?SKIPS IF USER MODE.
                             TAD I   (MQ         /!\?
                             CDF 0               /!\?
                             DCA I   X0          /!\?
                             TAD     MONFL       /!\?
                             AND     C7700       /!\?
```

```
                        DCA I   X0      /!\?
               >
               >
00530  4051             JMS     CDFUF   /!\IF USERMODE SKP, ELSE CDF INSTRFLD,
00531  7410             SKP             /!\?
00532  5775'            JMP     SCED    /!\?DONT TOUCH LOCS IN USERS IF,
00533  1773             TAD I   (X      /!\?
00534  7421             STORMQ          /!\?
00535  1772             TAD I   (BASE   /!\?
00536  6201             CDF 0           /!\?
00537  3410             DCA I   X0      /!\?
00540  7701             GETMQ           /!\?
00541  3410             DCA I   X0      /!\?
00542  5775'            JMP     SCED    /!\?JUMP TO THE SCHEDULER,
00543  0000    IFDEF STATX <MONSCD, 0;0 >
00544  0000

               /SET CURRENT TSK TO WAIT; WAITCONDITIONS IN AC.
00545  0000    CURWT,   0
00546  6002             IOF             /\
00547  3157             DCA     INT3    /\WAITCONDITIONS IN INT3.
00550  7040             CMA             /\
00551  1164             TAD     CURTSK  /\PTR TO TCB[4] IN AC,
00552  4771'            JMS     OURUNQ  /\
00553  5745             JMP I   CURWT   /\

00554  1051    TSKSAF,  TAD     CDFUF   /IF WE CAME JUST FROM THE SCEDULER
00555  7650             SNA CLA         /CDFUF IS STILL 0
00556  5775'            JMP     SCED    /AND WE NEED NOT SAVE STATUS,
00557  5300             JMP     MONSAF

               IFNDEF PDP8E <
               SBYTER, 0
                        CLL RTR;RTR;RTR
                        JMP I   SBYTER
               >

00560  0304             AVPT;(0;AVPT=.-2
00561  0570
       0560

00570  0000
00571  1136
00572  0107
00573  0110
00574  7723
00575  0617
00576  0731
00577  0202
       0600    PAGE
```

```
             /A TSK WAS SCHEDULED THAT IS ONDISK (TCB[6] B0 SET).
             /SET TSK TO WAIT WITH FWTSWP CONDITION AND SEND A MS TO
             /MTFTCH TO FETCH THE TSK FROM DISK.
00600  1377  ONDISK, TAD     (FWTSWP
00601  4776'         JMS     CURWT   /\CALLING DEAF ROUTINES AND CHANGING TCB.
00602  4775'         JMS     GN5     /\GET MS NODE
00603  3410          DCA I   X0      /\CLEAR SENDERWORD.
00604  1010          TAD     X0      /\
00605  3154          DCA     INT0    /\PTR TO MS[0].
00606  3410          DCA I   X0      /\CLEAR LINKWORD.
00607  1164          TAD     CURTSK  /\
00610  3410          DCA I   X0      /\PTR TO TCB[5] IN MS
00611  1374          TAD     (MTFTCH-1/\
00612  4571          JMS I   ZSNDREP /\SEND MS TO MTFTCH.
00613  2164          ISZ     CURTSK  /\
00614  7350          AC3777          /\
00615  0564          AND I   CURTSK  /\CLEAR ONDISK BIT.
00616  3564          DCA I   CURTSK  /\
             /       JMP     SCED    /\
                   /SCHEDULER.
00617  7201        SCED,    CLA IAC          /-?
00620  3166                 DCA     SCDREQ   /-?SCDREQ NO LONGER PENDING.
00621  6001                 ION              /-?
00622  3164        SCEDLP,  DCA     CURTSK
00623  3167                 DCA     SCDINH   /SCHEDULING NO LONGER INHIBITED.
00624  1565                 TAD I   HPRIO    /INSPECT PRIO RUNQS.
00625  7440        SCDLP2,  SZA              /FCHEX FALLS IN HERE.
                                             /NOTE: POSSIBLE CHANGES IN HPRIO WILL
                                             /KICK US BACK TO SCED.
00626  5233                 JMP     SCDF     /WE FOUND A TASK.
00627  6202                 CIF 0            /\DEAF WHILE UPDATING HPRIO!
00630  2165                 ISZ     HPRIO    /\UPDATE HPRIO TO NEXT PRIO RUNQ.
00631  2165                 ISZ     HPRIO    /\
00632  5222                 JMP     SCEDLP
00633  3010        SCDF,    DCA     X0
00634  1010                 TAD     X0
00635  2167                 ISZ     SCDINH   /NO SCHEDULING WHILE BUILDING TASKS
00636  3164                 DCA     CURTSK   /STATUS, STORE PTR TO TCB[5].
00637  3051                 DCA     CDFUF    /SIGNAL FOR TSKSAF AND FCHEX ROUTINE.
00640  1410                 TAD I   X0       /B0=1: CODE OF TSK STILL ON DISK.
00641  7710                 SPA CLA
00642  5200                 JMP     ONDISK
00643  1410                 TAD I   X0
00644  3107                 DCA     BASE     /TEMP
00645  1107                 TAD     BASE
00646  4773'                JMS     FCHECK   /ASSERT VFLD IN CORE. IF NOT ISSUE
                                             /SWAPREQ IF POSSIBLE AND RETURN TO
                                             /SCEDNI.

                   IFDEF PDP8E <
                   IFDEF EAE <
00647  0127                 AND     C70      /CDF TO ACT DF IN AC.
00650  7110                 CLL RAR;RTR
00651  7012
00652  3007                 DCA     MONFL
00653  1145                 TAD     C7700
00654  0107        PTBASE,  AND     BASE     /GET VFLDS AGAIN. LOC PTS AT BASE.
00655  3163                 DCA     VCURIF   /CURRENT VINSTRFLD.
00656  1163                 TAD     VCURIF
00657  7002                 BSWR
00660  4773'                JMS     FCHECK
00661  3321                 DCA     CDIFI    /STORE CDF TO VINSTRFLD.
00662  1321                 TAD     CDIFI
00663  0127                 AND     C70
00664  1007                 TAD     MONFL
00665  3007                 DCA     MONFL
00666  1410                 TAD I   X0
00667  3004                 DCA     MONAC
00670  1410                 TAD I   X0
00671  3054                 DCA     MONPC
00672  1410                 TAD I   X0       /GET MQ
00673  7431                 SWAB    /LOAD MQ, ASSERT MODE B.
00674  1410                 TAD I   X0       /EAE +L
00675  3110                 DCA     X        /TEMP.
```

```
00676  1110           TAD     X
00677  7403           ACS
00700  1133           TAD     C200
00701  0110  PTX,     AND     X        /LOC PTS AT X.
00702  7650           SNA CLA          /MODE A OR B?
00703  7447           SWBA
00704  1110           TAD     X        /L+GTFL+UF+SCDINH TO FLAGS.
00705  0145           AND     C7700
00706  1007           TAD     MONFL
00707  3007           DCA     MONFL
00710  1131           TAD     C100     /USERMODE?
00711  0007           AND     MONFL
00712  7650           SNA CLA
00713  5316           JMP     SCDF2    /SET BASE, X AND REQCDF.
00714  1372           TAD     (ISZ CDFUF/SET CDFUF ROUTINE TO SKIP,
00715  5330           JMP     SCDF3    /AND DONT TOUCH ANY LOCS IN USER'S FIELD.
00716  1410  SCDF2,   TAD I   X0
00717  3150           DCA     MONO
00720  1410           TAD I   X0
00721  6201  CDIFI,   CDF              /CDF TO VINSTRFLD.
00722  3701           DCA I   PTX
00723  1150           TAD     MONO
00724  3654           DCA I   PTBASE
00725  1037           TAD     XVCALL   /REINITIALIZE VVRCDF FOR A NEW TASK.
00726  3771  VRCDF3,  DCA I   (VVRCDF+2/BOOOH! VRCDF FALLS IN HERE.
00727  1321           TAD     CDIFI
00730  3052  SCDF3,   DCA     CDFUF+1 /NOT SURE ABOUT DF!

             /FALL INTO MONITOR RETURN.
00731  6203  MONEX,   CIF CDF 0        /-MAKE SURE ABOUT DF.
00732  3770'          DCA     EXTALL   /-FSTEXT ALLOWED AGAIN.
00733  1166           TAD     SCDREQ   /-SCDREQ PENDING?
00734  7650           SNA CLA          /-
00735  5767'          JMP     TSKSAF   /-YES; GO SCHEDULING.
00736  1007           TAD     MONFL    /-JUST A SHORT RETURN.
00737  6005           RTF              /-
00740  0377           AND     (400     /-OVERRIDE PRIO SCHEDULING?
00741  3167           DCA     SCDINH   /-ADJUST SCDINH.
00742  1004           TAD     MONAC    /-
00743  5454           JMP I   MONPC
             >
             >

             IFDEF PDP8I <
             IFNDEF EAE <
                      DCA     CDFINS   /CDF TO USERS' DF.
                      TAD     C7700
             PTBASE,  AND     BASE     /GET VFLDS AGAIN. LOC PTS AT BASE.
                      DCA     VCURIF   /CURRENT VIRTUAL INSTRFLD.
                      TAD     VCURIF
                      BSWR
                      JMS     FCHECK
                      DCA     CDIFI    /STORE CDF TO USERS' INSTRFLD.
                      TAD     CDIFI
                      IAC
```

```
        DCA     CIFINS  /SET CIFINS TO USERS' IF.
        TAD I   X0
        DCA     MONAC
        TAD I   X0
        DCA     MONPC
        TAD I   X0
        STORMQ  /TEMP
        TAD I   X0
        DCA     MONFL
        TAD     C100    /USERMODE?
        AND     MONFL
        SNA CLA
        JMP     SCDF2   /NO, SET BASE, X, AND REQCDF.
        TAD     (ISZ CDFUF/SET CDFUF ROUTINE TO SKIP
        JMP     SCDF3   /AND DONT TOUCH ANY LOCS IN USERS' FIELD.
SCDF2,  TAD I   X0
        DCA     MON0
        TAD I   X0
CDIFI,  CDF             /TO USERS' INSTRFLD.
        DCA I   (X
        GETMQ
        DCA I   (MQ
        TAD     MON0
        DCA I   PTBASE
        TAD     XVCALL  /REINITIALIZE VVRCDF FOR A NEW TASK.
VRCDF3, DCA I   (VVRCDF+2/BOOH!  VRCDF FALLS IN HERE.
        TAD     CDIFI
SCDF3,  DCA     CDFUF+1 /NOT SURE ABOUT DF!

/FALL INTO MONITOR RETURN.

MONEX,  CIF CDF 0       /-MAKE SURE ABOUT DF.
        DCA     EXTALL  /-FAST EXIT ALLOWED AGAIN.
        TAD     SCDREQ  /-SCDREQ PENDING?
        SNA CLA         /-
        JMP     TSKSAF  /-YES; GO SCHEDULING.
        TAD     MONFL   /-
        AND     (400    /-
        DCA     SCDINH  /-ADJUST SCDINH FLAG.
CIFINS, CIF             /-TO USERS' IF
CDFINS, CDF             /-TO USERS' DF
        TAD     MONFL   /-
        RAL             /-SET LINK.
        AND     C200    /-USERMODE?
        SZA CLA         /-
        SUF     /-
        TAD     MONAC   /-
        ION             /-
        JMP I   MONPC
>
>


/IF CURRENT TASK IS PROTECT (USERMODE ON) THEN SKIP,
/ELSE CDF CURRENT INSTRFLD.
```

```
          /CALLED EITHER DEAF OR WITH SCDINH >0.
          /CDFUF, 0                 /!\?
          /        ISZ     CDFUF    /!\?
          /        JMP     CDFUF    /!\?
          /WE PATCHED CDFUF ROUTINE OVER VCDIF ROUTINE.
```

```
                     /FCHECK FAILED. VFLDS NOT IN CORE.
                     /IF CALLED FROM A TSK (CDFUF 'NE' 0),
                     /SAVE IT WITH THE NEW DF. OTHERWISE (CALLED FROM
                     /SCHEDULER) SCHEDULE THE NEXT IMPORTANT TSK.
                     /IF CURTSK IS REENTRANT (B1 OF TCB[6] SET), DONT SCHEDULE OTHER
                     /TSKS OF THIS PRIORITY.
00744  1164          FCHEX,  TAD     CURTSK
00745  3010                  DCA     X0      /PTR TO TCB[6].
00746  1051                  TAD     CDFUF   /CALLED FROM TASK?
00747  7650                  SNA CLA
00750  5354                  JMP     SCEDNI  /NO.
00751  2010                  ISZ     X0
00752  1153                  TAD     MON3    /GET REQUESTED DATAFLD.
                     /       TAD     (-FLDTAB        /FLDTAB AT BEGIN OF PAGE?
00753  5766'                 JMP     MONSF2  /JMP INTO SAVE ROUTINE.

                     /WE CANNOT START UP THE FIRST TASK IN THIS RUNQ, BECAUSE ITS
                     /VFLDS ARE NOT IN ACTUAL CORE.
                     /TAKE THE NEXT ONE IN THIS RUNQ, UNLESS IT STARTS
                     /WITH A REENTRANT TASK.
00754  3167          SCEDNI, DCA     SCDINH
00755  1166                  TAD     SCDREQ  /IF SCHEDULE REQ PENDING
00756  7650                  SNA CLA         /JUMP RIGHT INTO THE SCHEDULER
00757  5217                  JMP     SCED
00760  7332                  AC2000          /CURTSK REENTRANT
00761  0410                  AND I   X0
00762  7650                  SNA CLA         /IF SO, SKIP OVER THE REST OF THIS RUNQ.
00763  1564                  TAD I   CURTSK
00764  5225                  JMP     SCDLP2


                     IFDEF PDP8I <AVPT;(0;AVPT=.-2>
00766  0510
00767  0554
00770  0422
00771  0022
00772  2051
00773  2000
00774  5050
00775  4346
00776  0545
00777  0400
       1000          PAGE
```

```
                   /***** SOME SMALL DEAF ROUTINES USED BY THE SCHEDULER *****

                   /SEND.
                   /INSERT A MS IN TSKS RCQ. IF TSK WAS WAITING FOR MSS FROM
                   /THIS PROCES, INSERT IT IN RUNQ (BY CALLING INRUNQ).

                   /INT0=PTR TO MS[0] SENDERWORD.
                   /AC  =PTR TO TCB[4] TSK WAITWORD.
                   /USES INT0,INT2.

                   /REPORT.
                   /INSERT A MS IN TSKS RPQ. IF TSK WAS WAITING FOR THAT MS INSERT
                   /TSK IN RUNQ (BY CALLING INRUNQ).
                   /
                   /INT0   =PTR TO MS[0] SENDERWORD.
                   /USES INT1,INT2.
01000  0000        SNDREP, 0                     /!\
01001  7100                CLL                   /!\
01002  7440                SZA                   /!\
01003  5206                JMP     .+3           /!\
01004  7240                CLA CMA               /!\
01005  1554                TAD I   INT0          /!\CML
01006  3155                DCA     INT1          /!\
01007  7420                SNL                   /!\SKIPS IF REPORT
01010  7344                ACM2                  /!\
01011  1136                TAD     M2            /!\
01012  1155                TAD     INT1          /!\
01013  3156                DCA     INT2          /!\PTR TO TCB[4] IF SEND
                                                 /!\PTR TO TCB[2] IF REPORT
                                                 /!\L=0 IF REP, L=1 IF SND
01014  7010                RAR                   /!\AC=0: REP, AC4000: SND.
01015  7051                CIA RAR               /!\AC=4000: REP, AC=2000: SND.
       4000        FWTRP=4000;FWTMS=2000         /!\
       2000
01016  0555                AND I   INT1          /!\
01017  7450                SNA                   /!\WAITING FOR SEND OR REPORT?
01020  5251                JMP     MSAPP         /!\NO. ONLY APPPEND.
01021  4777'               JMS     CHKMS         /!\MS ALLOWED TO REACTIVATE TSK?
01022  5251                JMP     MSAPP         /!\
01023  7307        MSIN,   AC4                   /!\
01024  1155                TAD     INT1          /!\
01025  3156                DCA     INT2          /!\PTR TO TSKS AC.
01026  1155                TAD     INT1          /!\
01027  7040                CMA                   /!\
01030  1164                TAD     CURTSK        /!\REPORT MSG TO CUR TSK?
01031  7640                SZA CLA               /!\
01032  5235                JMP     .+3           /!\
01033  1376                TAD     (MONAC        /!\NOTE ONLY HERE IF CURTSK IS WAITING
                                                 /!\FOR THIS REPORT AND YET ITS STATUS
                                                 /!\ IS NOT SAVED!
01034  3156                DCA     INT2          /!\PTR TO TASKS AC
01035  7326                AC2                   /!\
01036  1154                TAD     INT0          /!\
01037  3556                DCA I   INT2          /!\MSPTR IN TASKS AC.
01040  4276                JMS     INRUNQ        /!\INSERT IN RUNQ.
```

```
01041  7325           AC3                /!\
01042  1156           TAD      INT2      /!\NOTE MONFL=MONAC+3!
01043  3156           DCA      INT2      /!\
01044  7330           AC4000             /!\
01045  0555           AND I    INT1      /!\0 IF SND, 4000 IF REPORT
01046  1556           TAD I    INT2      /!\
01047  3556           DCA I    INT2      /!\SET TSKS L IF REPORT
01050  5600           JMP I    SNDREP    /!\

01051  2156   MSAPP,  ISZ      INT2      /!\PTR TO HEAD OF MSQ,
01052  2154           ISZ      INT0      /!\
01053  1554           TAD I    INT0      /!\MS ALREADY IN SOME Q?
01054  7640           SZA CLA            /!\IF AC=0, PROBABLY NOT.
01055  5600           JMP I    SNDREP    /!\YES. LEAVE. DONT MAKE CIRCULAR QS,
              IFDEF STATX <
01056  1275           TAD      MSQMAX    /!\SET UP COUNTER.
01057  7040           CMA                /!\
01060  3155           DCA      INT1      /!\CTR
              >
01061  1556   MSAPP1, TAD I    INT2      /!\SEARCH MSQTAIL.
01062  7450           SNA                /!\
01063  5271           JMP      MSAPP2    /!\
01064  3156           DCA      INT2      /!\
              IFDEF STATX <
01065  2155           ISZ      INT1      /!\INCREMENT CTR.
01066  5261           JMP      MSAPP1    /!\
01067  2275           ISZ      MSQMAX    /!\IF OVERFLOW INCREMENT QMAX,
              >
01070  5261           JMP      MSAPP1    /!\
01071  1154   MSAPP2, TAD      INT0      /!\APPEND MESSAGE
01072  3556           DCA I    INT2      /!\
01073  3554           DCA I    INT0      /!\WARNING: IT SEEMS VERY ROTTEN FOR
                                         /INTMS!!! BUT RPQ WON'T BECOME CIRCULAR,
                                         /AS WE ONLY COME HERE IF INTMS WAS THE
01074  5600           JMP I    SNDREP    /!\LAST ONE IN RPQ!!!!!

01075  0000   IFDEF STATX <MSQMAX, 0 >

```

```
                /THE RUNQS.
                /THE SCHEDULER USES A PRIORITY RUNQ, I.E. A RUNQ FOR EACH
                /PRIORITY. EACH TSK HAS A PRIORITY IN THE RANGE 1-7. PRIORITY 0
                /IS RESERVED FOR MONTSKS, PRIORITY 10 FOR THE IDLETSK.
                /THE HIGHEST PRIORITY IS PRIO0.
                /THE ARRAY RQPTRS HOLDS THE HEADS AND TAILS OF EACH RUNQ AS
                /FOLLOWS:
                /       RQPTRS[2N] HEAD OF RUNQ OF PRIO N.
                /       RQPTRS[2N+1] TAIL OF RUNQ OF PRIO N.
                /THE ITEMS IN THE RUNQ ARE TCBS OF RUNNABLE TSKS. THEY ARE
                /THREADED  THROUGH THE TCB LINKWORDS (TCB[5]).
                /IN PRINCIPLE EACH PRIO RUNQ IS A FIFO Q, BUT FCHEX
                /MAY SKIP OVER TASKS WHOSE FLDS ARE NOT IN ACTUAL CORE.

                /INRUNQ.
                /ERASE ALL WAIT CONDITIONS, BUT NOT STOPD.
                /INSERT A TSK IN ITS PRIO RUNQ, IF IT IS NOT STOPPED (STOPDBIT
                /SET). SIGNAL SCEDREQ BY CLEARING SCDREQ.
                /
                /INT1 =PTR TO TCB[4] WAIT WORD.
                /USES INT0, INT2.
01076  0000     INRUNQ, 0                 /!\
01077  1555             TAD I    INT1     /!\REMOVE ANY WAIT CONDITIONS EXCEPT
01100  0125             AND      C37      /!\STOP
01101  3555             DCA I    INT1     /!\
01102  1555             TAD I    INT1     /!\2*PRIO
01103  7150             CLL CMA RAR       /!\STOPDBIT TO LINK.
01104  7420             SNL               /!\STOPPDBIT SET?
01105  5327             JMP      INRQEX   /!\YES. SKIP THE REST. AC>=0.
01106  7164             STL CMA RAL       /!GET PRIO*2+1
01107  1375             TAD      (RQPTRS  /!\
01110  3156             DCA      INT2     /!\PTR TO TAIL OF RUNQ.
01111  2155             ISZ      INT1     /!\INT1:=PTR TO TCB[5] LINK WORD.
01112  1555             TAD I    INT1     /!\GET PTR IN TIMEOUT Q IF ANY.
01113  3154             DCA      INT0     /!\PTR IN TOQ OR INNOCENT PTR.
01114  3554             DCA I    INT0     /!\CLEAR POSSBILE TIMEOUTNODE.
01115  1556             TAD I    INT2     /!\PREVIOUS LAST ITEM.
01116  3154             DCA      INT0     /!\PTR TO PRECESSOR.
01117  1155             TAD      INT1     /!\
01120  3554             DCA I    INT0     /!\APPEND.
01121  3555             DCA I    INT1     /!\ZERO TCB[5].
01122  1155             TAD      INT1     /!\UPDATE TAIL.
01123  3556             DCA I    INT2     /!\
01124  1165             TAD      HPRIO    /!\COMPARE NEW PRIO TO CURRENT HIGHEST.
01125  7041             CIA               /!\-HPRIO
01126  1156             TAD      INT2     /!\-HPRIO+NPRIO+1
01127  7700     INRQEX, SMA CLA           /!\MUST WE SET SCDREQ?
01130  5676             JMP I    INRUNQ   /!\NO.
01131  7040             CMA               /!\
01132  1156             TAD      INT2     /!\YES. SET NEW HPRIO.
01133  3165             DCA      HPRIO    /!\
01134  3166             DCA      SCDREQ   /!\
01135  5676             JMP I    INRUNQ   /!\
                   /OURUNQ.
                   /REMOVE A TASK FROM THE RUNQ.
                   /SET UP WAITCONDITIONS IN TSK WAITWORD TCB[4].
                   /WAITCONDITIONS:
            4000   FWTRP=  4000               /WAIT REPORT.
            2000   FWTMS=  2000               /WAIT FOR MS TO BE RECEIVED.
            0200   FWTTM=  200                /WAIT FOR TIME OUT.
            0400   FWTSWP= 400                /WAIT FOR SWAP IN.
            0001   STOPD=1 /STOPPED. WAITING FOR RESUME COMMAND.
                   /INT3 CONTAINS WAITCONDITIONS.
                   /AC=PTR TO TCB[4] TSK WAITWORD.
                   /USES INT1,INT2,INT3.
                   /
    01136   0000   OURUNQ, 0                  /\
    01137   3155           DCA      INT1      /\PTR TO TCB[4].
    01140   1555           TAD I    INT1      /\FETCH 2*PRIO.
                   /       AND      (36       /\REMOVE WAITS?
    01141   1375           TAD      (RQPTRS   /\
    01142   3156           DCA      INT2      /\PTR TO PRIO RUNQ.
    01143   1157           TAD      INT3      /\GET WAITCONDITIONS.
    01144   1555           TAD I    INT1      /\SET UP WAIT CONDITIONS.
    01145   3555           DCA I    INT1      /\
    01146   2155           ISZ      INT1      /\PTR TO TCB[5] LINK WORD.
    01147   1156           TAD      INT2      /\INITIATE SEARCH FOR PRECESSOR.
    01150   3157   OURQLP, DCA      INT3      /\PTR IN Q.
    01151   1557           TAD I    INT3      /\FETCH LINK
    01152   7041           CIA                /\
    01153   1155           TAD      INT1      /\WHO PTS AT THE TSK TO BE REMOVED?
    01154   7650           SNA CLA            /\
    01155   5360           JMP      OURQF     /\PRECESSOR FOUND.
    01156   1557           TAD I    INT3      /\NOT FOUND YET. SEARCH DOWN THE Q.
    01157   5350           JMP      OURQLP    /\
    01160   1555   OURQF,  TAD I    INT1      /\DELETE TSK FROM RUNQ.
    01161   3557           DCA I    INT3      /\
    01162   1155           TAD      INT1      /\
    01163   3555           DCA I    INT1      /\INNOCENT PTR IN LINKWORD (SEE INRUNQ).
    01164   1557           TAD I    INT3      /\IS THE REMOVED TSK THE LAST ONE OF THE
    01165   7640           SZA CLA            /\Q
    01166   5736           JMP I    OURUNQ    /\NO.
    01167   2156           ISZ      INT2      /\YES; UPDATE TAIL. INT2:=PTR TO TAIL.
    01170   1157           TAD      INT3      /\NEW LAST TSK.
    01171   3556           DCA I    INT2      /\
    01172   5736           JMP I    OURUNQ    /\
                   /NOTE: QS CONTAINING ONLY ONE TSK ARE TREATED WELL THIS WAY!

                   IFNDEF STATX < AVPT;(0;AVPT=.-2 >
    01175   4700
    01176   0004
    01177   3514
            1200   PAGE
```

```
        /APP. A2.   MC8.2. VIRTUAL CORE MANAGER, PAGE- AND FIELDREQUEST.
        /***** VIRTUAL CORE MANAGER *****
        /
        /THIS OPERATING SYSTEM IS EQUIPED WITH UPTO 64 VIRTUAL MEMORY
        /FIELDS. THE EXACT NUMBER OF VIRTUAL FIELDS IS CONTROLLED BY THE
        /PARAMETER VFMAX IN THE CONFIG FILE.
        /EACH VIRTUAL FIELD MAY BE SWAPPED INTO AN ACTUAL FIELD OF THE
        /PDP8. THE NUMBER OF ACTUAL FIELDS IS DETERMINED BY THE PARAMETER
        /ACTMAX IN THE CONFIG FILE.
        /IF A VIRTUAL FIELD IS NOT IN CORE, IT IS STORED ON THE DISK IN
        /ITS OWN FIELDSLOT (A SPECIFIC AREA ON THE DISK).
        /
        /THE STATUS OF A VIRTUAL FIELD IS SPECIFIED IN THE FIELDTABLE
        /(FLDTAB). EACH VIRTUAL FIELD OCCUPIES ONE WORD IN FLDTAB.
        /BIT ASSIGNMENT OF FLDTAB:
        /B0:    0: ON DISK, 1: IN CORE; SEE BELOW.
        /B1:    0: NEVER USED, 1: USED (STATISTICS).
        /B2:    GIVEN OUT AS WHOLE FIELD FOR DATA; GENERAL PAGE 0
        /       ROUTINES  CANNOT BE COPIED INTO THIS FIELD.
1000    FDATA=1000
        /B3:    OTO PART OF PAGE 0 OF THIS FIELD IS OCCUPIED.
0400    FZREQ=400
        /B5:    CORERESIDENT; NOT ALLOWED TO SWAP THIS FIELD OUT OF CORE.
0100    FCRES=100
        /B6-8   ACTUAL FIELD IF THE FIELD IS IN CORE.
        /B9:    MONIN BIT; A COPY OF THE GENERAL PAGE 0 ROUTINES IS IN
        /       THIS  FIELD.
0004    FMONIN=4
        /B10:   0: NO DATA PRESENT, 1: DATA PRESENT (USED TO
        /       OPTIMIZE SWAPPING).
0002    FDP=2
        /B11:   0: FREE, 1: OCCUPIED.
0001    FOCCUP=1
        /
        /NOTES:
        /   IF A FIELD IS IN CORE AND DATA ARE PRESENT, WE COMPUTE A CDF
        /TO ITS ACTUAL FIELD BY ANDING THE BITS IN FLDTAB WITH 6271.
        /THIS TRIC FAILS IF THE FIELD IS FREE!!!.
        /   SWAPPING IN OR OUT A FIELD IS A DUMMY ACTION IF THE DATA
        /PRESENT BIT (B10) IS LOW.
        /
        /TWO DIFFERENT STORAGE ALLOCATION SYSTEMS ARE USED TO ALLOCATE
        /STUFF IN THESE VIRTUAL FIELDS.
        /THE ROUTINES GETFLD AND FRFLD ARE USED TO REQUEST AND RETURN
        /COMPLETE FIELDS.
        /THESE ROUTINES DIRECTLY ACCESS THE FIELDTABLE TO CHECK, SET AND
        /CLEAR THE NONFREE BIT (B11).
        /A NUMBER OF UPTO 37 PAGES CAN BE REQUESTED AND RETURNED BY
        /CALLING  THE ROUTINE PGRQN. IT USES A CHAIN OF FREE CORE JUNKS
        /(FCCHAN). ON REQUEST PGRQN TRIES TO FIND A JUNK OF SUFFICIENT
        /LENGTH IN FCCHAN. IF THIS FAILS GETFLD IS CALLED. LIKEWISE ON
        /RETURN FRFLD IS CALLED IF A FREE CORE JUNK OF 37 PAGES IS
        /RETURNED. PAGE 0 IS NEVER GIVEN OUT  BY PGRQN. IT IS RESERVED
        /FOR THE GENERAL PAGE 0 ROUTINES.  NONPROTECT TASKS (USERMODE
        /OFF) HAVE A MAXIMUM LENGTH OF 37 PAGES AND USE THE GENERAL
```

```
                /PAGE 0. THEY ARE ALLOCATED USING PGRQN.
                /A TASK HAS ACCESS TO AT MOST TWO DIFFERENT FIELDS (IF AND DF).
                /(NOTE: WE DIDNOT ALLOW THE CIF INSTRUCTION IN A TASK).
                /BEFORE RUNNING A TASK THE ROUTINE FCHECK CHECKS WETHER THESE
                /FIELDS ARE ACTUALLY IN CORE. IF A TASK WANTS TO CHANGE ITS
                /DATAFIELD  IT CALLS EITHER VVCDF OR VVRCDF. THESE ROUTINES CHECK
                /WETHER THE NEW DATAFIELD IS IN CORE.
                /
                /SWAPPING FIELDS.
                /IF SOMEONE WANTS TO ACCESS A FIELD WHICH IS NOT IN CORE AT THAT
                /MOMENT, IT MUST BE SWAPPED IN. BEFORE THAT ANOTHER FIELD MUST BE
                /SWAPPED OUT. WHICH FIELD IS TO BE REMOVED IS DETERMINED BY THE
                /PAGING ROUTINES. WHICH FIELDS ARE ACTUALLY IN CORE IS DENOTED IN
                /CORMAP (COREMAP). IT IS AN ARRAY OF ACTMAX ENTRIES, AN ENTRY FOR
                /EACH  ACTUAL FIELD.

```

```
                   /***** PAGE REQUEST AND RETURN SECTION.
                   /
                   /1-37 PAGES OF FREE CORE MAY BE REQUESTED BY CALLING THE ROUTINE
                   /PGRQN.
                   /JUNKS OF FREE CORE ARE ORGANIZED IN A CHAIN: FCCHAN (FREE CORE
                   /CHAIN). THE CHAIN IS SORTED ACCORDING TO INCREASING PAGE AND
                   /FLDNUMBER. WHEN A CERTAIN AMOUNT OF PAGES IS REQUESTED WE FIRST
                   /SEARCH THE  CHAIN TO FIND A JUNK OF SUFFICIENT LENGTH. IF THE
                   /SEARCH FAILS WE CALL GETFLD TO FETCH A FRESH EMPTY FLD. THIS IS
                   /BROKEN INTO TWO JUNKS (NOT IF 37 PAGES WERE REQUESTED), ONE JUNK
                   /IS HANDED TO THE CALLER, THE OTHER IS INSERTED IN THE CHAIN.
                   /PAGE 0 OF EACH FLD IN THIS SYSTEM IS RESERVED FOR THE MONITOR.
                   /
                   /PAGES PREVIOUSLY REQUESTED BY CALLING PGRQN MUST BE RETURNED BY
                   /CALLING PGRQN. THE NEW JUNK IS INSERTED
                   /IN THE CHAIN POSSIBLY MERGED WITH OTHER JUNKS. IF A WHOLE FLD
                   /(37 PAGES) IS FREE IT ISNOT INSERTED IN THE CHAIN.
                   /FRFLD IS CALLED TO SET THE EMPTY CONDITION IN FLDTAB.
                   /
                   /LAYOUT OF A NODE IN THE FREE CORE CHAIN:
                   /W0:     PTR TO NEXT NODE; 0 INDICATES THE END.
                   /W1:     B0-5 VFLD#, B7-11 FIRST FREE PAGE.
                   /W2:     MINUS NUMBER OF FREE PAGES.
                   /


                   /WE HOPE THESE ARE INNOCENT TEMPS.
            0070   PGRTEM=VJUMS
            0045   PGRREM=VVRDF
            0025   PGRLEN=VVCDF
            0020   PGROLD=VVRCDF

                   /HELP ROUTINE FOR PAGE RETURN AND REQUEST.
                   /THIS ROUTINE IS CALLED WHEN INITIALIZING PGRQ AND PGRTN.
                   /THE ROUTINE IS REENTERED HALFWAY EACH TIME A NEW NODE HAS
                   /TO BE INSPECTED.
                   /        JMS     PGR
                   /        SNA                   /AC=0 IF END OF CHAIN REACHED.
                   /        JMP     EXH           /FCCHAN EXHAUSTED.
                   /        ///
                   /LENGTH IN MARG1 B7-11.
                   /AT RETURN AC CONTAINS FIRST ITEM OF NEXT NODE.

    01200   0000   PGR,     0
    01201   1161            TAD     MARG1   /GET LENGTH.
    01202   0125            AND     C37     /REMOVE OPTIONS.
                   IFDEF CHECK      <
    01203   7450            SNA
    01204   4576            SYSHLT          /DONT ACCEPT ZERO LENGTH
                   >
    01205   3025            DCA     PGRLEN  /LENGTH
    01206   1377            TAD     (FCCHAN /FETCH PTR TO HEAD OF FREE CORE CHAIN.
    01207   5212            JMP     .+3     /FALL INTO LOOP.
    01210   7200   PGRLP,   CLA             /NEXT NODE. ROUTINE IS REENTERED HERE.
    01211   1420            TAD I   PGROLD  /UPDATE PTR TO PRECESSOR.
    01212   3020            DCA     PGROLD  /STORE PTR TO PRECESSOR.
```

```
01213  1420  PGRLPM, TAD I    PGROLD  /JUMP HERE AFTER A MERGE.
01214  7450          SNA              /LIST EXHAUSTED?
01215  5600  PGREXH, JMP I    PGR     /RETURN WITH AC=0.
01216  3010          DCA      X0      /PTR TO NODE TO BE INSPECTED.
01217  1410          TAD I    X0      /FIRST ITEM= FLD+PAG. NEVER 0!
01220  5600          JMP I    PGR


```

```
                        /PGRQN. PAGE REQUEST/RETURN.
                        /REQUEST: AC=0, RETURN: AC=FLD+PAG.
       01221  0000      PGRQN,  0
       01222  7440              SZA                 /REQUEST OR RETURN?
       01223  5266              JMP     PGRTN
                        /PGRQ. PAGE REQUEST.
                        /REQUEST 1-37 PAGES OF FREE CORE.
                        /OPTIONS:       CRES        REQUEST PAGES IN CORERESIDENT FLD.
                        /               ZREQ        REQUEST TO USE THE OTOPART OF PAGE 0 IN
                        /                           FLD OF THESE PAGES.
                        /       JMS     PGRQN
                        /OPTIONS+LENGTH IN MARG1.
                        /AC=FLD+PAG AT RETURN. B0-4 PAGE#, B6-11 VFDL#.
                        /USES X0, MON3, MARG1, MONAC.
                        /
                        PGRQ,
       01224  4200              JMS     PGR         /FALL INTO INSPECT NODES LOOP.
       01225  7450              SNA
       01226  5776'             JMP     PGRQX       /JUMP FOR EXHAUSTED LIST.
       01227  3070      PGRQLP, DCA     PGRTEM      /STORE FLD+PAG.
       01230  1410              TAD I   X0          /FETCH MINUS LENGTH
       01231  1025              TAD     PGRLEN      /COMPARE SIZE.
       01232  7540              SMA SZA
       01233  5210              JMP     PGRLP       /TO SHORT, NEXT NODE.
       01234  3045              DCA     PGRREM      /STORE MINUS REMAINDER.
       01235  1070              TAD     PGRTEM
       01236  4355              JMS     RDFLD       /MON3:=PTR IN FLDTAB, CONTENTS OF FLDTAB
       01237  0375              AND     (FCRES+FZREQ    /IN AC.
                        /NOW A  VERY COMPLICATED TEST FOR THE OPTIONS FOLLOWS.
                        /THIS FLD AND THE OPTIONS MATCH IF:
                        /NOT BOTH OF THEM HAS THE ZREQBIT SET
                        /AND EITHER BOTH OR NONE HAS THE CRESBIT SET.
       01240  1161              TAD     MARG1       /NOTE: ZREQ AND CRES ARE NON-ADJACENT
       01241  0374              AND     (FZREQ↑2+FCRES  /BITS.
       01242  7640              SZA CLA
       01243  5210              JMP     PGRLP       /NO MATCH.
       01244  1161              TAD     MARG1       /SET UP ZREQBIT IN FLDTAB IF NECESSARY.
       01245  0373              AND     (FZREQ
       01246  1553              TAD I   MON3        /WE KNOW IT ISNOT SET YET.
       01247  3553              DCA I   MON3
       01250  1420              TAD I   PGROLD      /BREAK OR DELETE THIS NODE.
       01251  3010              DCA     X0
       01252  1070              TAD     PGRTEM      /FLD+PAG
       01253  1025              TAD     PGRLEN      /ADD LENGTH
       01254  3410              DCA I   X0          /NEW FLD+PAG
       01255  1045              TAD     PGRREM      /MINUS NEW LENGTH
       01256  7450              SNA                 /BAD TRICK:
       01257  4772'             JMS     DF3         /FIRST DELETE AND FREE NODE.
       01260  3410              DCA I   X0          /AND THEN STILL CHANGE ITS LAST ITEM.
       01261  1070      PGREX,  TAD     PGRTEM
       01262  7112              CLL RTR;RTR;RTR
       01263  7012
       01264  7012
       01265  5621              JMP I   PGRQN
```

```

                      /PGRTN. PAGE RETURN.
                      /RETURN PAGES OF FREE CORE PREVIOUSLY REQUESTED BY CALLING PGRQN.
                      /THE NEW JUNK OF FREE CORE IS INSERTED IN THE FREE CORE CHAIN.
                      /IF POSSIBLE IT IS MERGED WITH ADJACENT JUNKS. IF A WHOLE FLD IS
                      /FREE AGAIN (37  FREE PAGES) IT IS REMOVED FROM THE CHAIN AND THE
                      /EMPTY CONDITION IS SET IN FLDTAB.
                      /       TAD                 /FLD+PAG IN AC. B0-4 PAGE#, B6-11 VFLD#.
                      /       JMS       PGRQN
                      /LENGTH+FZREQBIT IN MARG1
                      /
                      PGRTN,
01266  7106                   CLL RTL;RTL;RTL
01267  7006
01270  7006
01271  3045                   DCA       PGRREM  /FLD+PAG.
01272  1373                   TAD       (FZREQ
01273  0161                   AND       MARG1
01274  7040                   CMA
01275  3200                   DCA       PGR     /SAVE THIS OPTION.
01276  1045                   TAD       PGRREM
01277  4355                   JMS       RDFLD   /GET CONTENTS OF FLDTAB
01300  0200                   AND       PGR     /CLEAR POSSBILE ZREQBIT.
01301  3553                   DCA I     MON3
01302  3070                   DCA       PGRTEM  /CLEAR PTR FOR RETURN.
                      PGRTN2,                     /PGRQ MAY FALL IN HERE.
01303  4200                   JMS       PGR     /FALL INTO INSPECT NODE LOOP.
01304  7450                   SNA
01305  5332                   JMP       PGRNM   /JUMP FOR EXHAUSTED LIST. APPEND HERE.
01306  7141                   CIA CLL           /AC!=MINUS FLD+PAG  OF ITEM. NONZERO!
01307  1045                   TAD       PGRREM  /COMPARE WITH FLD+PAG TO BE INSERTED.
01310  7420                   SNL               /INSERT IT HERE?
01311  5327                   JMP       PGRMB   /YES.
01312  1410                   TAD I     X0      /NO. PERHAPS FORWARD MERGE? ADD MINUS
01313  7640                   SZA CLA           /LENGTH.
                                                /AC=FLD+PAG TO BE INSERTED
                                                /-(FLD+PAG +LENGTH) OF NODE.
01314  5210                   JMP       PGRLP   /NO FORWARD MERGE, NEXT NODE.
01315  1420   PGRMF,          TAD I     PGROLD  /FORWARD MERGE.
01316  3010                   DCA       X0
01317  1410                   TAD I     X0      /UPDATE FLD+PAG TO BE INSERTED.
01320  3045                   DCA       PGRREM
01321  1410   PGRM2,          TAD I     X0      /BACKWARD MERGE JUMPS HERE.
01322  7041                   CIA
01323  1025                   TAD       PGRLEN  /UPDATE LENGTH TO BE INSERTED.
01324  3025                   DCA       PGRLEN
01325  4772'                  JMS       DF3     /DELETE AND FREE MERGED NODE.
01326  5213                   JMP       PGRLPM  /WE WILL FIND THE PLACE TO INSERT
                                                /IMMEDIATELY.
01327  1025   PGRMB,          TAD       PGRLEN  /ADD LENGTH
01330  7650                   SNA CLA           /AC=FLD+PAG+LENGTH TO BE INSERTED -
                                                /-(FLD+PAG) OF NODE.
01331  5321                   JMP       PGRM2   /BACKWARD MERGE.
01332  1371   PGRNM,          TAD       (-37    /NO MERGE.
01333  1025                   TAD       PGRLEN  /WHOLE FLD FREE?
```

```
01334  7650          SNA CLA
01335  5351          JMP     PGRFF   /YES.
01336  4770'         JMS     GN3     /GET NODE. X0 PTS TO NODE AT RETURN.
01337  1420          TAD I   PGROLD
01340  3410          DCA I   X0
01341  1010          TAD     X0
01342  3420          DCA I   PGROLD  /INSERTED IN CHAIN.
01343  1045          TAD     PGRREM
01344  3410          DCA I   X0
01345  1025          TAD     PGRLEN
01346  7041          CIA             /MINUS LENGTH.
01347  3410          DCA I   X0
01350  5261          JMP     PGREX
01351  1045  PGRFF,  TAD     PGRREM  /SET UP EMPTY CONDITION IN FLDTAB.
01352  7002          BSWR
01353  4767'         JMS     FRFLD
01354  5261          JMP     PGREX

01355  0000  RDFLD,  0
01356  7002          BSWR
01357  0130          AND     C77
01360  1112          TAD     XFLDTAB
01361  3153          DCA     MON3
01362  1553          TAD I   MON3
01363  5755          JMP I   RDFLD

01364  0560          AVPT;(0;AVPT=.-2
01365  1366
       1364

01366  0000
01367  1507
01370  4340
01371  7741
01372  4317
01373  0400
01374  1100
01375  0500
01376  1400
01377  4721
       1400  PAGE
```

```
01400  1377   PGRQX,  TAD     (FCRES+FZREQ/LIST EXHAUSTED, GET A NEW EMPTY FLD.
01401  0161           AND     MARG1
01402  4247           JMS     GETFLD  /SETS BITS IN AC IN FLDTAB.
                                      /RETURNS VFLD# IN AC B6-11.
01403  7106           CLL RTL;RTL;RTL
01404  7006
01405  7006
01406  7001           IAC             /LEAVE PAGE 0 FOR MONITOR.
01407  3070           DCA     PGRTEM
01410  1553           TAD I   MON3    /GET STATUS OF FLD.
                                      /IF IT IS IN CORE AND MONINBIT NOT
                                      /SET, WE MUST COPY THE GENERAL PAGE ZERO
                                      /INTO IT.
01411  7510           SPA     /SKIP IF NOT IN CORE; AC 'NE' 0.
01412  0122           AND     C4;MONIN=4
       0004
01413  7650           SNA CLA
01414  4776'          JMS     GPZRIN  /INSERT GENERAL PAGE ZERO STUFF.
01415  1025           TAD     PGRLEN  /COMPUTE LENGTH OF REMAINDER
01416  7041           CIA
01417  1125           TAD     C37
01420  7450           SNA
01421  5775'          JMP     PGREX   /NO REMAINDER
01422  3161           DCA     MARG1
01423  1070           TAD     PGRTEM
01424  1025           TAD     PGRLEN  /FLD+PAG OF REMAINDER
01425  3045           DCA     PGRREM
01426  5774'          JMP     PGRTN2  /RETURN THIS JUNK AND LEAVE PGRQN.
```

```
             /***** LOCKING AND UNLOCKING FIELDS.
             /
             /FLDS MAY BE LOCKED IN CORE FOR SEVERAL REASONS:
             /A.     THE TASKS IN THIS FLD ARE USED SO OFTEN THAT IT IS WORTH
             /       TO HAVE THEM CORERESIDENT (SEE THE USE OF THE CRESBIT IN
             /       THE PAGEREQUEST ROUTINES).
             /B.     ROUTINES CONNECTED TO AN INTSLOT MUST RESIDE IN
             /       CORERESIDENT FLDS.
             /C.     I/O PROCESSES MAY TEMPORARILY LOCK FLDS IN CORE (E.G.,
             /       DATABREAK).
             /FLDS ARE LOCKED (UNLOCKED) BY CALLING THE ROUTINE VFLOCK, WITH
             /AC=100 (AC=-100).
             /THEY ASSUME THE VFLD TO BE LOCKED (UNLOCKED) TO BE PRESENT IN
             /CORE  AND MON3 TO POINT TO ITS ENTRY IN FLDTAB.
             /LOCKS ARE COUNTED IN CORMAP B0-5.
       0100  FCRES=100        /CORERESIDENCYBIT.

01427  0000  VFLOCK, 0
01430  3246          DCA      VFLTM2  /TEMP
01431  1553          TAD I    MON3
01432  0127          AND      C70
01433  7112          CLL RTR
01434  7010          RAR
01435  1170          TAD      ZCORMAP
01436  3245          DCA      VFLTEM
01437  1645          TAD I    VFLTEM
01440  1246          TAD      VFLTM2  /CHANGE LOCK COUNT.
             IFDEF CHECK <
01441  7510          SPA
01442  4576          SYSHLT           /WE DONT ACCEPT MORE THAN 37 LOCKS.
             >
01443  3645          DCA I    VFLTEM
01444  5627          JMP I    VFLOCK

01445  0000  VFLTEM, 0
01446  0000  VFLTM2, 0
```

```
             /***** GETFLD AND FRFLD.
             /
             /FIELDS ARE CLAIMED BY CALLING GETFLD. THEY ARE FREED BY CALLING
             /FRFLD. GETFLD SEARCHES THE FIRST FREE FIELD FROM THE FLDTAB AND
             /RETURNS ITS NUMBER IN AC B6-11. POSSIBLE SPECIAL CONDITIONS THAT
             /WERE SET IN AC AT THE CALL ARE INDICATED IN FLDTAB.
             /FRFLD SETS THE FIELD FREE CONDITION IN FLDTAB.
             /

             /       TAD     SPFLAG  /FETCH SPECIAL FLAGS (E.G. CRESBIT) IN AC
             /       JMS     GETFLD
             /AC B6-11 HOLD VFLD#. IF NO MORE FREE FLDS ARE AVAILABLE A
             /SYSTEM HALT OCCURS.
             /USES MON3.

01447  0000  GETFLD, 0
01450  1373          TAD     (2001   /ADD USED AND OCCUPIED CONDITIONS TO
01451  3324          DCA     FLDTEM  /FLAGS. STORE FLAGS.
01452  1112          TAD     XFLDTAB /SEARCH FLDTAB FOR FREE FLD.
01453  3153          DCA     MON3
01454  1372          TAD     (-VIRMAX-1
01455  3307          DCA     FRFLD   /JUST A CTR.
01456  7001  GETFLP, IAC             /SEARCH LOOP.
01457  0553          AND I   MON3    /FREE?
01460  7650          SNA CLA
01461  5266          JMP     GETFLF  /YES.
01462  2153          ISZ     MON3
01463  2307          ISZ     FRFLD   /INCREMENT COUNT.
01464  5256          JMP     GETFLP
01465  4576          SYSHLT          /FREE FIELDS EXHAUSTED!!!
01466  1324  GETFLF, TAD     FLDTEM  /GET FLAGS
01467  7040          CMA
01470  0553          AND I   MON3
01471  1324          TAD     FLDTEM  /ADD TO FLDTAB ENTRY
01472  7500          SMA             /IN CORE?
01473  5303          JMP     GETFL2  /NO.
01474  1120          TAD     C2;FDP=2/YES, SET DATAPRESENT BIT.
       0002
01475  3553          DCA I   MON3
01476  1131          TAD C100;FCRES=100 /WILL IT BE CORERESIDENT?
       0100
01477  0324          AND     FLDTEM
01500  7440          SZA
01501  4227          JMS     VFLOCK  /YES, LOCK IT.
01502  5304          JMP     GETFLX
01503  3553  GETFL2, DCA I   MON3
01504  1153  GETFLX, TAD     MON3
01505  0130          AND     C77     /FLDTAB AT BEGIN OF PAGE
01506  5647          JMP I   GETFLD

             /       TAD     FLDPTR  /VFLD# IN AC B6-11, OTHER BITS DISCARDED.
             /       JMS     FRFLD
             /USES MON3.

01507  0000  FRFLD,  0               /TEMP FOR GETFLD.
```

```
01510  0130           AND     C77
01511  1112           TAD     XFLDTAB
01512  3153           DCA     MON3
01513  1553           TAD I   MON3      /GET CONTENTS OF FLDTAB.
01514  0131           AND     C100
01515  7041           CIA               /0 OR -100
01516  7440           SZA
01517  4227           JMS     VFLOCK    /YES, SO IT WAS LOCKED, UNLOCK IT.
01520  1371           TAD     (7777-FOCCUP-FDP-FCRES-FZREQ-FDATA
01521  0553           AND I   MON3
01522  3553           DCA I   MON3      /RESET CONDITIONS IN FLDTAB.
01523  5707           JMP I   FRFLD
01524  0000  FLDTEM,  0

01525  1364           AVPT;(0;AVPT=.-2
01526  1570
       1525

01570  0000
01571  6274
01572  7700
01573  2001
01574  1303
01575  1261
01576  1600
01577  0500
       1600  PAGE
```

```
                     /STORE STUFF OF GENERAL PAGE ZERO INTO FIELD.
                     /MON3 PTS TO ENTRY OF FIELD IN FLDTAB.
01600  0000  GPZRIN, 0
01601  1122          TAD     C4
01602  0553          AND I   MON3
01603  7112          CLL RTR;RAR     /MONINBIT IN LINK.
01604  7010
01605  1553          TAD I   MON3
01606  0141          AND     C7773   /MASK OUT POSSIBLE MONINBIT.
01607  1122          TAD     C4      /DONT TOUCH LINK!
01610  3553          DCA I   MON3    /SET MONINBIT.
01611  1117          TAD     XCDF70
01612  0553          AND I   MON3    /GET CDF TO ACT FLD.
01613  3214          DCA     GPZCDF
01614  6201  GPZCDF, CDF
01615  7430          SZL             /WAS MONINBIT ALREADY SET?
01616  5232          JMP     GPZRI2  /YES; ONLY ADJUST (CIF CDF CUR).
01617  1377          TAD     (13
01620  3010          DCA     X0
       1634          GPZCT=P0CTAD
01621  1376          TAD     (-150+14
01622  3234          DCA     GPZCT
01623  1375          TAD     (TAD 14
01624  3225          DCA     GPZTAD
01625  1000  GPZTAD, TAD
01626  2225          ISZ     GPZTAD
01627  3410          DCA I   X0
01630  2234          ISZ     GPZCT
01631  5225          JMP     GPZTAD


                     /UPDATE GENERAL PAGE 0.
                     /WHEN THE GENERAL PAGE 0 IS PRESENT IN A FLD, A NUMBER
                     /OF LOCS OF THAT FLD MUST CONTAIN CIF CDF CUR,
                     /WHERE CUR IS THE CURRENT ACTUAL FLD.
                     /THIS ROUTINE FILLS THESE LOCATIONS.
                     /MON3 POINTS TO THE ENTRY IN FLDTAB OF THE
                     /FLD TO BE UPDATED.
01632  1374  GPZRI2, TAD     (TAD P0CLST
01633  3234          DCA     P0CTAD
01634  1000  P0CTAD, TAD             /GET PTR IN PAGE 0.
01635  7450          SNA             /0 TERMINATES LIST OF LOCS.
01636  5245          JMP     P0CEX
01637  3225          DCA     GPZTAD  /TEMP
01640  7326          AC2
01641  1214          TAD     GPZCDF  /CIF CDF CUR IN AC.
01642  3625          DCA     GPZTAD
01643  2234          ISZ     P0CTAD
01644  5234          JMP     P0CTAD
01645  6201  P0CEX,  CDF 0
01646  5600          JMP I   GPZRIN

01647  0021  P0CLST, VVRCDF+1
01650  0026          VVCDF+1
01651  0052          MYCDF
```

```
01652  0061                  VGET+1
01653  0071                  VJUMS+1
01654  0076                  VPUT+1
01655  0000                  0

             /***** PAGING ROUTINES
             /
             /PAGING ROUTINES ARE USED TO DECIDE WHICH VIRTUAL FIELDS ARE IN
             /CORE AND WHICH ARE TEMPORARILY STORED ON DISK.
             /FROM A VARIETY OF POSSIBLE PAGING ALGORITHMS WE CHOOSE A
             /VERY SIMPLE ONE: WE SWAP OUT THE FIELD THAT WAS LONGEST IN CORE.
             /THIS ALGORITHM WAS CHOSEN BECAUSE IT IS EASY TO PROGRAM (SEE
             /VFLESS), DOESNOT TAKE ANY TIME AS LONG AS NO FIELDSWAPS ARE
             /NEEDED AND DOESNOT WORK TOO BAD.
             /BETTER ALGORITHMS MIGHT TAKE ADVANTAGE OF THE FACT THAT WE KNOW
             /WHICH FIELDS ARE LAST USED. TO DO SO ONE MUST REMOVE THE SLASHES
             / IN FCHECK AND WRITE SOME ROUTINE VFSIGN. FCHECK IS CALLED
             /AT LEAST TWICE FOR EACH TASK SWITCH!
             /THE PAGING ROUTINES  CONTROL THE CONTENTS OF THE ARRAY:
             /CORMAP[0:ACTMAX], ONE ENTRY FOR EACH ACTUAL FIELD.
             /LAYOUT OF THE ENTRIES IN CORMAP:
             /       B0-5:   COUNTS HOW MANY TIMES THIS FIELD IS LOCKED IN
             /               CORE (E.G., WHEN DATABREAK PROCESSES USE THIS
             /               FIELD).
             /       B6-11:  THE VIRTUAL FIELD THAT RESIDES IN THIS FIELD.

             /VFLESS.
             /ROUTINE TO DETERMINE WHICH VFLD IS TO BE SWAPPED OUT.
             /WE TAKE THE ONE THAT WAS LONGEST IN CORE, UNLESS IT WAS LOCKED.
             /THEN WE TAKE THE ONE THAT WAS NEXT LONGEST IN CORE ETC....
             /IF ALL FIELDS ARE LOCKED, THE ERROR RETURN IS TAKEN.
             /       JMS     VFLESS
             /       JMP     ERROR    /AC=0. ALL FLDS ARE LOCKED.
             /       ...              /AC=PTR IN CORMAP.

01656  4722  VFNEXT, CORMAP  /POINTER TO (FLD THAT WAS LONGEST IN CORE)-1.

01657  0000  VFLESS, 0
01660  1373          TAD     (-ACTMAX-1
01661  3200          DCA     VFLCT   /COUNT THE NUMBER OF FLDS WE TRIED.
01662  1256  VFLTRY, TAD     VFNEXT  /KEEP THIS POINTER CIRCULING IN CORMAP.
01663  1372          TAD     (-CORMAP-ACTMAX
01664  7640          SZA CLA
01665  5270          JMP     .+3
01666  1170          TAD     ZCORMAP
01667  3256          DCA     VFNEXT
01670  2256          ISZ     VFNEXT
01671  1656          TAD I   VFNEXT  /IS THE CANDIDATE FIELD LOCKED?
01672  0145          AND     C7700
01673  7650          SNA CLA
01674  5300          JMP     VFLF    /NO. FIELD TO BE SWAPPED OUT IS FOUND
01675  2200          ISZ     VFLCT   /ALL FLDS CHECKED?
01676  5262          JMP     VFLTRY  /NO.
01677  5657          JMP I   VFLESS  /YES. ALL LOCKED. TAKE ERROR RETURN.
01700  1256  VFLF,   TAD     VFNEXT  /RETURN PTR IN CORMAP.
```

```
2156   01701  2257          ISZ     VFLESS
2157   01702  5657          JMP I   VFLESS
2158
2159          1600   VFLCT=GPZRIN
2160   01703  1525          AVPT;(0;AVPT=.-2
2161   01704  1771
2162          1703
2163   01771  0000
2164   01772  3047
2165   01773  7770
2166   01774  1247
2167   01775  1014
2168   01776  7644
2169   01777  0013
2170          2000   PAGE
```

```
/APP. A3.   MC8.3. MONITOR TASKS.
/***** MONITOR TASKS *****
/
/TASKS IN MC8 MAY BE DIVIDED INTO 3 GROUPS:
/MONITOR TASKS  THESE TASKS ARE EFFECTIVELY A PART OF THE
/               MONITOR. THEY ARE BUILT AS A TASK JUST BECAUSE IT
/               IS CONVENIENT TO THINK OF THEM AS TASKS.
/               SOMETIMES THEY ARE ACTIVATED VIA THE SCHEDULER,
/               SOMETIMES THE MONITOR DIRECTLY JUMPS INTO SUCH
/               TASKS.
/               NOTE: NO OTHER TASKS (NOT EVEN MONITOR
/               TASKS) CAN INTERRUPT ACTIVE MONITOR TASKS.
/SYSTEM TASKS   THESE TASKS CONSTITUTE THE MULTIPROGRAMMING MODE
/               OF MC8. THEY ARE WRITTEN AND ASSEMBLED AS
/               SEPERATE MODULES, KNOWING LITTLE ABOUT THE
/               MONITOR AND IN PRINCIPLE NOTHING ABOUT OTHER
/               TASKS. OF COURSE ONE COULD WRITE FAMILIES OF
/               TASKS USING WELL DEFINED  INTERFACES.
/PROTECT TASKS  THESE TASKS CONSTITUTE THE MC8 TIME SHARING MODE.
/               TO EACH TIME SHARING USER A BARE MACHINE HAVING
/               AN APPROPRIATE CONFIGURATION IS SIMULATED. OTHER
/               PARTS OF THE SYSTEM ARE PROTECTED AGAINST ANY
/               OBSCURE PROGRAMS EXECUTED BY THESE TASKS. THEY
/               HAVE NO DIRECT ACCESS TO ANY PART OF THE SYSTEM
/               AND RUN WITH USERMODE ON.
/
/THIS SECTION CONTAINS THE FOLLOWING MONITOR TASKS:
/MTSWAP.
/       THIS TASK SWAPS VIRTUAL FIELDS INTO AND OUT OF ACTUAL
/       CORE. IT UPDATES THE ARRAYS FIELDTABLE AND CORMAP.
/       IT IS ENTERED FROM THE SCHEDULER (UPON SWAP COMPLETION).
/MTSDSK.
/       SYSTEM DISK MANAGER. THIS TASK SWAPS INFORMATION FROM AND
/       TO THE SYSTEM DISK.
/       IT ACCEPTS MESSAGES FROM ANY MONITOR AND SYSTEM TASK.
/       THE MESSAGE SWAPMS (COMMUNICATION MESSAGE WITH MTSWAP) IS
/        GIVEN SPECIAL TREATMENT.
/       IT IS ENTERED ONLY FROM THE SCHEDULER.
/MTTIME.
/       THIS TASK KEEPS TRACK OF THE SYSTEM TIME (DOUBLE
/       PRECISION COUNTER IN UNITS OF 0.1 SEC). ON START
/       SYSTEM TIME IS SET TO 0.
/       MOREOVER IT HANDLES THE TIME OUT QUEUE. TASKS THAT WANT
/       TO BE CONTINUED AFTER SOME SECONDS CAN SPECIFY A TIME OUT
/       REQUEST TO THE MONITOR. THEIR TIME OUT VALUE MUST BE
/       STORED IN AC. THESE TASKS ARE ENTERED IN THE TIME OUT
/       QUEUE AND EACH 0.1 SEC THEIR TIME OUT VALUE IS
/       INCREMENTED. ON OVERFLOW THEY ARE REINSERTED IN THE
/       RUNQ.
/MTFTCH.
/       A RUNNABLE TSK THAT IS NOT IN CORE IS ALLOCATED IN CORE,
/       SWAPPED IN AND INSERTED IN THE RUNQ.
```

```
                        /***** FCHECK AND MTSWAP.
                        /
                        /FCHECK (ASSERT FIELD IN CORE).
                        /CALLED WITH VFLD# IN AC.
                        /IF VFLD IN CORE, RETURN CDF TO ACTFLD IN AC.
                        /ELSE IF NO SWAP IS PENDING YET (SWAPMS[2]=0) ISSUE A SWAP
                        /    REQUEST. SAFE THE CURRENT TASK (IF ANY) WITH THE REQUESTED
                        /    VFLD AS DF.
                        /    RETURN TO THE SCHEDULER.
                        /
                        /EACH TASK CAN ACCESS TWO VFLDS: IF AND DF. WHEN SCHEDULING A
                        /TASK  THE SCHEDULER CALLS FCHECK TO ASSERT THAT BOTH THESE
                        /FIELDS ARE IN ACTUAL CORE. IF THEY ARE NOT FCHECK INITIATES A
                        /SCHEDULE OF THE NEXT IMPORTANT TASK (JMP SCEDNI).
                        /
                        /IF A TASK WANTS TO CHANGE ITS DF (NOTE: CHANGING IF IS NOT
                        /ALLOWED) IT CALLS FCHECK TO ASSERT THAT THE NEW DF IS IN ACTUAL
                        /CORE. IF IT IS NOT, FCHECK JUMPS INTO MONSAF, SAFING THE
                        /CURRENT TASK WITH THE REQUESTED NEW DF.

      02000   0000      FCHECK, 0                       /AC =VFLD#
      02001   0130              AND     C77
      02002   1112              TAD     XFLDTAB
      02003   3153              DCA     MON3
      02004   1553              TAD I   MON3
                                                        /AC!=CONTENTS OF FLDTAB.
                        /       SMA                     /IN CORE?
                        /       JMP     FCSWAP          /NO.
                        /       AND     C70
                        /       CLL RTR
                        /       RAR
                        /       JMS     VFSIGN          /SIGNAL TO VIRTUAL CORE MANAGER WHICH
                        /                               /FLDS ARE USED (USED IN PAGING
                        /                               /ALGORITHM).
                        /       TAD I   MON3
                        /       AND     XCDF70          /YES; COMPUTE CDF TO ACT FLD.
                        /       JMP I   FCHECK
      02005   0117              AND     XCDF70
      02006   7510              SPA
      02007   5600              JMP I   FCHECK

      02010   7200      FCSWAP, CLA                     /WE HAD RUBBISH IN AC.
      02011   1237              TAD     SWAPMS          /SWAP REQUEST PENDING?
      02012   7650              SNA CLA                 /IF NOT WHICH FIELD SHALL WE SWAP OUT?
      02013   4777'             JMS     VFLESS
      02014   5776'             JMP     FCHEX           /NONE! ALL LOCKED. DANGER FOR DEAD LOCK!!
      02015   3237              DCA     SWAPMS          /PTR IN CORMAP.
      02016   1637              TAD I   SWAPMS          /GET VFLD# OF SWAP OUT FIELD
      02017   1112              TAD     XFLDTAB
      02020   3240              DCA     SWAPMS+1/SWAP OUT-ENTRY IN FIELDTABLE.
      02021   7350              AC3777
      02022   0640              AND I   SWAPMS+1
      02023   3640              DCA I   SWAPMS+1/CLEAR INCORE BIT.
      02024   1153              TAD     MON3
      02025   3241              DCA     SWAPMS+2/SWAP IN-ENTRY IN FLDTAB.
```

```
02026  6002             IOF             /\
02027  1375             TAD     (SWAPMS-2/\PTR TO MS LINKWORD
02030  3154             DCA     INT0    /\
02031  1374             TAD     (MTSDSK-1/\
02032  4571             JMS I   ZSNDREP /\SEND THIS MS TO MTSDSK.
                                        /IT WILL BE REPORTED TO MTSWAP.
02033  6001             ION             /\
02034  5776'            JMP     FCHEX   /

                /SWAP MESSAGE.
02035  5035             MTSWAP          /SENDER WORD
02036  0000             0               /LINK WORD
02037  0000     SWAPMS, 0               /PTR TO COREMAP[ACTFLD]
02040  0000             0               /PTR TO FLDTAB[SWAP OUT FLD]
02041  0000             0               /PTR TO FLDTAB[SWAP IN  FLD]
```

```
             /MTSWAP.
             /THIS TASK IS ACTIVATED AFTER A SWAP OF VFLDS,
             /IT UPDATES FLDTAB AND CORMAP. MOREOVER ITS SCHEDULE GUARANTEES
             /A COMPLETE REEXAMINATION OF THE RUNQ'S, GENERALLY THE TASK IN
             /CHARGE OF WHICH THE SWAP WAS ISSUED WILL PROFIT FROM THAT
             /REEXAMINATION. THE SWAPMS IS ALWAYS SENT TO MTSDSK BY FCHECK AND
             /REPORTED TO MTSWAP.

             VFSWLP,
             /       CALL            /PERPETUAL LOOP.
             /               WTRP
             /               SWAPMS
02042  7200  MTSWIN, CLA             /PTR TO SWAPMS WAS IN AC.
             /       TAD     SWAPMS
             /       TAD     (-CORMAP
             /       JMS     VFSIGN  /SIGNAL TO VIRT,CORE MANAGER WHICH FLDS
02043  1641          TAD I   SWAPMS+2/ARE USED. GET STATUS OF SWAP IN FLD FROM
02044  0373          AND     (7777-4270-FDP  /FLDTAB.
02045  1350          TAD     SWNODE  /ACTFLD STILL IN SWNODE B6-8.
02046  1372          TAD     (4200+FDP/SET UP IN CORE +DATA PRESENT.
02047  3641          DCA I   SWAPMS+2
02050  1131          TAD     C100;FCRES=100 /SWAP IN FIELD CORERESIDENT?
       0100
02051  0641          AND I   SWAPMS+2
02052  1241          TAD     SWAPMS+2/PERHAPS AC=100, ONE LOCK.
02053  0132          AND     C177    /NOTE: FLDTAB AT BEGIN OF PAGE!
02054  3637          DCA I   SWAPMS  /SET NEW VFLD IN COREMAP.
02055  1241          TAD     SWAPMS+2
02056  3153          DCA     MON3
02057  1641          TAD I   SWAPMS+2        /GET STATUS OF SWAP IN FLD.
02060  0173          AND Z1000;FDATA=1000 /NO NEED FOR UPDATING PAGE 0
       1000
02061  7650          SNA CLA         /IF IT IS GIVEN OUT FOR DATA,
02062  4771'         JMS     GPZRIN  /ADJUST GENERAL PAGE 0.
02063  3237  MTSWEX, DCA     SWAPMS  /SIGNAL FREE FOR NEW SWAP.
             /       JMP     VFSWLP

             /SET MONTSK TO WAIT FOR REPORT WITHOUT SAVING. THIS SPARES TIME.
02064  1370          TAD     (-SWAPMS+2
02065  3767'         DCA     MTSWAP-3
02066  5766'         JMP     WTRPMT
```

```
                        /MTSDSK SYSTEM DISK MANAGER.
                        /SWAP TO AND FROM THE SYSTEM DISK.
                        /THE SYSTEM DISK HOLDS:
                        /       VFLDSLOTS; ROOM ON BACKGROUND STORAGE FOR EACH VFLD.
                        /       TASK LIBRARY; SYSTEM TASKS THAT MAY BE RUN.
                        /       OTHER; ANY ROOM LEFT ON THE SYSTEM DISK.
                        /THE SYSTEM DISK IS DIVIDED INTO UPTO 8 LOGICAL UNITS, THAT
                        /EACH CONTAIN UPTO 4096 BLOCKS OF 256 WORDS.
                        /A VFLDSLOT OCCUPIES 16 BLOCKS.
                        /THE VFLDSLOTS ARE ALLOCATED FROM VFBLOK (MULTIPLE OF 200CT)
                        /UPWARD. THE TASK LIBRARY IS ALLOCATED FROM TSBLOK UPWARD.
                        /BOTH THE VFLDSLOTS AND THE TASK LIBRARY ARE ASSUMED TO RESIDE ON
                        /LOGICAL UNIT 0.
                        /
                        /MTSDSK OPERATES IN TWO MODES: NORMAL MODE AND FIELDSWAP MODE.
                        /EACH TIME A NEW MESSAGE IS ACCEPTED (BY CALL;WTMS) FIRST THE
                        /MODE IS SELECTED. FIELDSWAP MODE IS ENTERED IF THE ACCEPTED
                        /MESSAGE IS THE SWAPMS. OTHERWISE NORMAL MODE IS ENTERED.
                        /
                        /NORMAL MODE.
                        /MESSAGE LAYOUT:
                        /W0:    R/W, NNNNN, FFF, UUU
                        /       W/R=1: WRITE ON, W/R=0: READ FROM DISK.
                        /       NNNNN: 5-BIT NUMBER OF PAGES. 0=32 PAGES.
                        /               NOTE: 1 BLOCK =2 PAGES.
                        /       FFF:    3-BIT ACTFLD# OF TRANSFER.
                        /       UUU:    3-BIT LOGICAL UNIT NUMBER.
                        /W1:    CORE ADDRESS.
                        /W2:    BLOCKNUMBER.
                        /NOTE!!!!!      THE CALLING TASK MUST ASURE THAT THE ACTUAL
                        /       FIELD IN WHICH THE TRANSFER TAKES PLACE IS IN
                        /       CORE. THIS IS MOST EASILY DONE BY USING
                        /       THE DFPARM OPTION WHILE SENDING THE MS.
                        /       THIS OPTION COPIES THE DATAFIELD OF THE TASK
                        /       INTO BIT 6-8 OF THE FIRST INFO WORD OF THE MS.
                        /THE FIELD CANNOT BE SWAPPED OUT MEANWHILE, AS POSSIBLE MSS TO
                        /SWAP OUT THAT FLD, WILL BE AFTER THE CURRENT ONE IN THE RCQ.
                        /
                        /FIELDSWAP MODE.
                        /MESSAGE LAYOUT:
                        /W0:    ACTFLD# +CORMAP
                        /W1:    SWAP OUT VFLD# +FLDTAB
                        /W2:    SWAP IN VFLD# +FLDTAB.
                        /IN THIS MODE FIRST THE SWAP OUT FIELD IS WRITTEN ON DISK IN ITS
                        /VFLDSLOT. SUBSEQUENT THE SWAP IN FIELD IS READ IN CORE FROM
                        /ITS VFLDSLOT.
                        /OPTIMIZATION OCCURS IF IN ONE OF THESE FLDS NO DATA ARE PRESENT.

                        SDNORM,
                        IFDEF STATX <
      02067  2353               ISZ     SDNCT
      02070  5273               JMP     .+3
      02071  2354               ISZ     SDNCT+1
      02072  7000               NOP
                        >
```

```
02073  1300            TAD     SDMSP   /ENTER NORMAL MODE.
02074  4765'           JMS     SDGO    /PASS MESSAGE TO SDGO ROUTINE.
                                       /RETURNS COMPLETION STATUS IN AC.
02075  3700            DCA I   SDMSP   /COMPLETION STATUS TO MS.
02076  4054    SDEX,   CALL            /REPORT MESSAGE; BOTH MODES.
02077  0013                    RP
02100  0000    SDMSP,          0
02101  4054    SDIN,   CALL            /WAIT FOR MESSAGE
02102  0014                    WTMS
02103  3300            DCA     SDMSP   /STORE PTR TO MESSAGE
02104  1300            TAD     SDMSP   /SELECT MODE, NORMAL OR FIELDSWAP.
02105  1364            TAD     (-SWAPMS
02106  7640            SZA CLA
02107  5267            JMP     SDNORM  /NORMAL.
               IFDEF STATX <
02110  2355            ISZ     SDFCT
02111  5314            JMP     .+3
02112  2356            ISZ     SDFCT+1
02113  7000            NOP
               >
02114  7330            AC4000          /FIELDSWAP MODE.
                                       /COMPUTE MESSAGE FOR SDGO ROUTINE.
02115  1640            TAD I   SWAPMS+1/GET ACTFLD# OF SWAPFLD.
02116  0363            AND     (4070   /AC=40X0, WRITE FLD X ON DISK 0.
02117  3350            DCA     SWNODE
02120  1240            TAD     SWAPMS+1
02121  4330            JMS     SDSWP   /JUST A LITTLE ROUTINE TO SAFE CODE.
02122  7350    SDSW1,  AC3777
02123  0350            AND     SWNODE  /SET TO READ
02124  3350            DCA     SWNODE
02125  1241            TAD     SWAPMS+2/SWAP IN FIELD.
02126  4330            JMS     SDSWP
02127  5276            JMP     SDEX

02130  0000    SDSWP,  0               /COMPUTE MESSAGE IN SWNODE AND CALL SDGO.
02131  3352            DCA     SWNODE+2/TEMP
02132  7326            AC2;FDP=2
       0002
02133  0752            AND I   SWNODE+2/DATA PRESENT IN FLD TO BE SWAPPED?
02134  7650            SNA CLA
02135  5730            JMP I   SDSWP   /NO, NO SWAP NEEDED.
02136  1352            TAD     SWNODE+2
02137  1362            TAD     (VFBLOK/20-FLDTAB/GET VFLD# +(VFBLOK/20)
02140  7106            CLL RTL;RTL     /*20. COMPUTE BLOCKNUMBER
02141  7006
02142  3352            DCA     SWNODE+2
02143  1361            TAD     (SWNODE /PASS PTR TO MS TO SDGO ROUTINE.
02144  4765'           JMS     SDGO    /RETURNS COMPLETIONSTATUS IN AC.
02145  7440            SZA             /ANY ERROR?
02146  4576            SYSHLT          /THAT'S THE END OF ALL!
02147  5730            JMP I   SDSWP

02150  0000    SWNODE, 0;0;0   /3TEMPS FOR MS.
02151  0000
02152  0000
```

```
2445
2446    02153  0000   IFDEF STATX <SDNCT, 0;0 >
2447    02154  0000
2448    02155  0000   IFDEF STATX < SDFCT, 0;0 >
2449    02156  0000
2450
2451                  IFNDEF STATX <AVPT;(0;AVPT=.-2 >
2452
2453    02161  2150
2454    02162  3400
2455    02163  4070
2456    02164  5741
2457    02165  2202
2458    02166  2644
2459    02167  5032
2460    02170  5743
2461    02171  1600
2462    02172  4202
2463    02173  3505
2464    02174  5102
2465    02175  2035
2466    02176  0744
2467    02177  1657
2468           2200   PAGE
```

```
02200  0000  SDPTR,  0                  /MUST BE AT BEGIN OF PAGE.
02201  0000  IFDEF STATX <SDERCT, 0 /ERROR COUNT >

             /SDGO ROUTINE.
             /CALLED WITH PTR TO MS (NORMAL MODE MESSAGE LAYOUT) IN AC.
             /EXECUTE SWAP.
             /REPORT STATUS AFTER TRANSFER IN AC; 0: OK, OTHER: SOME ERROR.

             IFDEF RK8E <
             IFZERO SYDISK-RK8E <
02202  0000  SDGO,   0
02203  3200          DCA     SDPTR      /PTR TO MS.
02204  7346          ACM3
02205  3321          DCA     RKERCT     /RKSYS! 3 TIMES TRY OVER.
02206  4054          CALL               /START UP THE INTERRUPTDRIVEN SECTION TO
02207  0027                  SIMINTR    /EXECUTE THE SWAP.
02210  2240                  RKRTRY     /START UP AT RKRTRY.
02211  4054          CALL
02212  0005                  WTRP
02213  2325  RKMSP,          RKMS
02214  7200          CLA                /SWAP COMPLETED. REMOVE DIRT FROM AC.
02215  1200          TAD     SDPTR      /GET COMPLETIONSTATUS IN AC.
02216  5602          JMP I   SDGO

             /DRIVEN BY RK8E INTERRUPTS.
             /RK8E IS CONNECTED TO SYWAIT+2.
             /DEAF ROUTINE.

02217  0377  RKER2,  AND     (1002      /!TREAT ERROR.
02220  7650          SNA CLA            /!
02221  5235          JMP     RKER3      /!NO NEED FOR RECALLIBRATION.
02222  7326          AC2                /!RECALLIBRATE.
02223  6742          DCLR               /!
02224  4330          JMS     SYWAIT     /!WAIT FOR COMMAND ACCEPT.
02225  7327          AC6                /!
02226  0322          AND     RKCMD      /!GET DRIVE#.
02227  1245          TAD     RK600      /!ADD INTR WHEN SEEK DONE.
02230  6746          DLDC               /!
02231  4330          JMS     SYWAIT     /!WAIT FOR RECALLIBRATE COMPLETE.
02232  1333          TAD     RKINTR     /!GET STATUS
02233  7440          SZA                /!ANY ERROR BITS IN STATUS?
02234  5312          JMP     RKERR      /!YES.
02235  7344  RKER3,  ACM2               /!RESET PTR FOR RETRY.
02236  1200          TAD     SDPTR      /!
02237  3200          DCA     SDPTR      /!
             /FALL INTO RETRY.

02240  1600  RKRTRY, TAD I   SDPTR      /!GET R/W +FFF+UUU
02241  0376          AND     (4077      /!
02242  1375          TAD     (400       /!ENABLE INTERRUPT WHEN DONE.
02243  3322          DCA     RKCMD      /!
02244  1145          TAD     C7700      /!
02245  0600  RK600,  AND I   SDPTR      /!GET NNNNN
02246  2200          ISZ     SDPTR      /!
02247  7045          CIA RAL            /!SET RK PAGE COUNT.
```

```
02250  7130           STL RAR           /!AC MIGHT BE =0.
02251  3326           DCA      RKPAG    /!
02252  1600           TAD I    SDPTR    /!
02253  2200           ISZ      SDPTR    /!
02254  6744           DLCA              /!LOAD CORE ADDRESS REGISTER.
02255  1600           TAD I    SDPTR    /!
02256  3327           DCA      RKBLK    /!BLOCK NUMBER.
02257  3325           DCA      RKCHEK   /!FORCE CHECK HEADER FIRST TIME.
02260  7100    RKLP,  CLL      /!
02261  1326           TAD      RKPAG    /!
02262  7450           SNA               /!READY?
02263  5315           JMP      RKDONE   /!
02264  1133           TAD      C200     /!UPDATE THIS COUNTER
02265  7420           SNL               /!OVERFLOW?
02266  3326           DCA      RKPAG    /!NO. STORE NEW VALUE.
                                        /IF OVERFLOW, AC=0 OR =100.
                                        /100 MEANS ONE PAGE ONLY.
02267  1325           TAD      RKCHEK   /!CHECK HEADER BIT?
02270  1322           TAD      RKCMD    /!GET COMMAND
02271  6746           DLDC              /!
02272  1327           TAD      RKBLK    /!
02273  6743           DLAG              /!
02274  7430           SZL               /!
02275  3326           DCA      RKPAG    /!IF OVERFLOW, CLEAR RKPAG TO SIGNAL END
02276  2327           ISZ      RKBLK    /!OF TRANSFER. INCREMENT BLOKNUM
02277  7410           SKP               /!
02300  2322           ISZ      RKCMD    /!NOTE: ONLY EVEN LOGICAL UNITS CAN
                                        /OVERFLOW. IN THAT CASE WE CONTINUE
                                        /TRANSFERRING TO THE NEXT LOGICAL UNIT.
02301  1327           TAD      RKBLK    /!TEST FOR CHECK HEADER CONDITION.
02302  0125           AND      C37
02303  7640           SZA CLA           /!
02304  1374           TAD      (1000    /!NO CHECK HEADER NEXT TIME.
02305  3325           DCA      RKCHEK   /!
02306  4330           JMS      SYWAIT   /!WAIT FOR FLAG.
02307  1333           TAD      RKINTR   /!GET STATUS AGAIN
02310  7450           SNA               /!RETURNS WITH DRST;RAL IN AC, 0 IF
                                        /NO ERROR OCCURRED.
02311  5260           JMP      RKLP     /!

               RKERR,                   /!
02312  2201    IFDEF STATX <ISZ SDERCT/;NOP> /!
02313  2321           ISZ      RKERCT   /!INCREMENT ERROR COUNT.
02314  5217           JMP      RKER2    /!TRY ONCE MORE
02315  3200    RKDONE, DCA     SDPTR    /!SAFE STATUS.
02316  1213           TAD      RKMSP    /!REPORT MS.
02317  4330           JMS      SYWAIT   /!
02320  5317           JMP      .-1      /!IGNORE OBSCURE INTERRUPTS.

02321  0000    RKERCT, 0                /3 ERROR COUNT.
02322  0000    RKCMD,  0                /RK COMMAND.

02323  5103           MTSDSK            /SENDERWORD
02324  0000           0        /LINK WORD.
               RKMS,
02325  0000  RKCHEK, 0
02326  0000  RKPAG,  0
02327  0000  RKBLK,  0

02330  0000  SYWAIT, 0                         /!WAIT HERE UNTIL NEW INTERRUPT OCCURS.
02331  3333          DCA      RKINTR  /!POSSIBLE MS TO BE REPORTED.
02332  4515          IEXIT            /!
02333  0000  RKINTR, 0                         /!
02334  6745          DRST             /!READ STATUS
02335  7104          CLL RAL          /!
02336  3333          DCA      RKINTR  /!SAVE TATUS
02337  6742          DCLR             /!CLEAR FLAGS
02340  5730          JMP I    SYWAIT  /!
             >>
```

```
        IFDEF RF08 <
        IFZERO SYDISK-RF08 <

                RFWC=7750; RFCA=7751
                DXAL=6643
                DIML=6615
                DIMA=6616
                DCMA=6601

SDGO,           0
                DCA     SDPTR   /PTR TO MS.
                TAD I   SDPTR   /FETCH UNIT NUMBER.
                AND     C7
                SNA
                JMP     .+3
                AC2             /SET NXTENT UNIT CONDITION.
                JMP I   SDGO
                TAD I   SDPTR   /GET FIELD
                AND     C70
                TAD     RF500   /ENABLE ERROR AND COMPLETION.
                DIML            /LOAD STATUS
                TAD I   SDPTR   / R/W
                SPA CLA
                AC2
                TAD     (6603
                DCA     RFINS   /6603 IF READ, 6605 IF WRITE.
                TAD I   SDPTR
                ISZ     SDPTR
                AND     C7700   /NUMBER OF PAGES IN B1-5
                CLL RAL         /NUMBER OF WORDS
                CIA
                DCA     RFWC    /WORD COUNT.
                CMA
                TAD I   SDPTR   /CORE ADDRESS.
                ISZ     SDPTR
                DCA     RFCA
                TAD I   SDPTR   /BLOCK # ON DISK
                CLL RTR;RTR
                DCA     RFTEMP
                TAD     RFTEMP
                DXAL            /LOAD DMA
                TAD     RFTEMP
                RAR
                AND     C7600
RFINS,          HLT             /GO!
                CALL
                        WTRP
                        RFMS
                CLA
                TAD     RFSTAT
                JMP I   SDGO    /RETURNS STATUS IN AC.

RFINTR,         0               /!INTERRUPT ARRIVES HERE!
                DIML            /!CLEAR INTR ENABLE.
                DIMA            /!READ STUTUS
                AND     C7      /!
        IFDEF STATX <
                SZA             /!
                ISZ     SDERCT  /!
                NOP             /!
        >
                DCA     RFSTAT  /!
                DXAL            /!CLEAR POSSIBLE NXTENT CONDITION.
                DCMA            /!CLEAR FLAGS
                IEXIT           /!
                        RFMS    /!

                MTSDSK          /SENDER WORD
                0               /LINK WORD
        RFMS,
        RFTEMP, 0
        RFSTAT, 0
        RF500,  500
                >>
```

```
                     /IDLE.
                     /THE IDLELOOP.
                     /THIS IS ESSENTIALLY NOT A MONITORTASK,
                     /IN FACT WE START BY CLEARING CURTSK, THUS ENABLING ANY OTHER
                     /TASK TO INTERRUPT US WITHOUT LOOSING TIME FOR SAFING,

                     IFDEF STATX <
      02341  1447    IDNSHL, -TCKLEN
      02342  0000    IDNS,   0;0
      02343  0000
      02344  1447    IDSWHL, -TCKLEN
      02345  0000    IDSW,   0;0
      02346  0000
      02347  0000    IDPTR,  0

      02350  6001    IDLP,   ION             /!
      02351  3564            DCA I   CURTSK  /SPEND SOME TIME AND CLEAR AC, INNOCENT.
                     >
      02352  3164    IDLEIN, DCA     CURTSK
                     IFNDEF STATX <IDLP, JMP .>
                     IFDEF STATX < /COUNT IDLE TIME
      02353  1773            TAD I   (SWAPMS /SWAPPING?
      02354  7650            SNA CLA
      02355  1372            TAD     (IDNSHL-1/NO.
      02356  7450            SNA
      02357  1371            TAD     (IDSWHL-1/YES.
      02360  3347            DCA     IDPTR   /PTR TO COUNT VARIABLES.
      02361  1370            TAD     (-TCKLEN
      02362  6002            IOF             /! DONT INTERRUPT OVERFLOW CONDITION
      02363  2347    IDLP2,  ISZ     IDPTR   /!
      02364  2747            ISZ I   IDPTR   /!
      02365  5350            JMP     IDLP    /!
      02366  3747            DCA I   IDPTR   /!
      02367  5363            JMP     IDLP2   /!
                     >

                     IFNDEF RK8E <AVPT;(0;AVPT=.-2>
                     IFDEF RK8E <
                     IFNZRO SYDISK-RK8E <AVPT;(0;AVPT=.-2>
                     >


      02370  1447
      02371  2343
      02372  2340
      02373  2037
      02374  1000
      02375  0400
      02376  4077
      02377  1002
             2400    PAGE
```

```

                    /MTFTCH.
02400  4054  MTFTIN, CALL
02401  0014                  WTMS
02402  4014          MSFREE  /FREE MS, GET CONTENTS OF W0.
02403  3107          DCA     BASE     /PTR TO TCB[5].
02404  4060          GET;    1        /GET STSK#
02405  0001
02406  0132          AND     C177
02407  7124          STL RAL
02410  1116          TAD     ZSTL     /
02411  3151          DCA     MON1     /PTR TO STATIC TASK LIST[TSK,1]
02412  1551          TAD I   MON1     /GET LENGTH +FCRES +FZREQ
02413  0377          AND     (537
02414  3161          DCA     MARG1    /ARG FOR PGRQN.
02415  1161          TAD     MARG1    /BUILD MS FOR MTSDSK IN FTCHMS.
02416  0125          AND     C37
02417  7002          BSWR
                    IFNDEF PDP8E <RAR>
02420  3334          DCA     FTCHMS   /STORE LENGTH AND LOGICAL UNIT 0.
02421  1161          TAD     MARG1    /SET UP COUNTER FOR RELOCATIONLOOP.
02422  0125          AND     C37
02423  7041          CIA
02424  3331          DCA     MTFCT
02425  4776'         JMS     PGRQN    /REQUEST FOR APPROPRIATE SPACE
02426  3335          DCA     MTFPAG   /TEMP.
02427  1335          TAD     MTFPAG   /GET FLD BITS.
02430  0130          AND     C77
02431  3255          DCA     MTFFLD
02432  1335          TAD     MTFPAG           /PAGE BITS
02433  0146          AND     C7600
02434  3335          DCA     MTFPAG   /ADDRESS OF FIRST PAGE.
02435  2110          ISZ     X        /PTR TO TCB[7] FLDS.
02436  1255          TAD     MTFFLD
02437  7002          BSWR
                    IFNDEF PDP8E <RAR>
02440  1255          TAD     MTFFLD
02441  3510          DCA I   X        /IF AND DF TO FLD.
02442  4060          GET;    11       /FETCH TCB[14]
02443  0011
02444  0132          AND     C177
02445  1335          TAD     MTFPAG
02446  3510          DCA I   X        /UPDATE TCB[14]
02447  2110          ISZ     X        /PTR TO TCB[15] DISK ADDRESS
02450  7350          AC3777           /11-BIT BLOCKNUMBER.
02451  0510          AND I   X        /GET DISK ADDRESS
02452  1375          TAD     (TSBLOK /ADD STARTING BLOCK OF TSK AREA.
02453  3336          DCA     FTCHMS+2
02454  4025          VCDF             /FORCE CORRECT DF AND FLD IN CORE.
02455  0000  MTFFLD, 0
02456  4054          CALL             /SEND MS TO SWAP TASK IN.
02457  0107                  SNDWTR+DFPARM
02460  2534                  FTCHMS   /FORCE DATAFLD AS PARAMETER IN MS.
02461  0004                  SMTSDSK
02462  7240          CLA CMA
```

```
02463  3255          DCA     MTFFLD  /FLAG FIRST TIME THROUGH RELOCATION LOOP.

              /THE TSK IS NOW SWAPPED IN.
              /DF STILL IS ITS FLD.
              /UPDATE RELOCATABLE CODE.
02464  1334          TAD     FTCHMS  /GET COMPLETION STATUS
02465  7440          SZA             /MUST BE 0.
02466  4576          SYSHLT
02467  1335          TAD     MTFPAG
02470  3151          DCA     MON1    /PTR TO TREATED PAGE.
02471  1551   RLOCLP, TAD I  MON1    /ADD VALUE OF MTFPAG (FIRST PAGE) TO
                                     /INITIAL LOCS OF EACH PAGE.
02472  7450          SNA             /TREAT MORE LOCS ON THIS PAGE?
02473  5301          JMP     RLOC1   /NO.
02474  1335          TAD     MTFPAG
02475  1146          TAD     M200    /REMEMBER TSK OFFSET OF 200.
02476  3551          DCA I   MON1
02477  2151          ISZ     MON1
02500  5271          JMP     RLOCLP
02501  2255   RLOC1,  ISZ    MTFFLD  /FIRST TIME HERE?
02502  5307          JMP     RLOC2   /NO.
02503  1151          TAD     MON1
02504  7001          IAC
02505  0132          AND     C177
02506  3334          DCA     FTCHMS  /PERHAPS FUTURE PC.
02507  1151   RLOC2,  TAD    MON1
02510  0146          AND     C7600
02511  1133          TAD     C200
02512  3151          DCA     MON1    /PTR TO NEXT PAGE
02513  2331          ISZ     MTFCT   /MORE PAGES TO BE TREATED?
02514  5271          JMP     RLOCLP  /YES.

              /ALL RELOCATION DONE. UPDATE PC.
02515  4060          GET;4   /FETCH OLD PC, IF ANY. (NOTE: DF:=0).
02516  0004
02517  7450          SNA             /IS THERE AN OLD PC?
02520  1334          TAD     FTCHMS  /IF NOT TAKE THE FIRST CODE LOCATION.
02521  1335          TAD     MTFPAG
02522  3510          DCA I   X
02523  7240          CLA CMA
02524  1107          TAD     BASE
02525  6002          IOF             /\
02526  3155          DCA     INT1    /\PTR TO TCB[4] IN INT1 FOR INRUNQ.
02527  4774'         JMS     INRUNQ  /\
              /      ION     /\INTERRUPTS WILL BE ON SOON.
02530  5200          JMP     MTFTIN

02531  0000   MTFCT,  0

02532  5051          MTFTCH
02533  0000          0
02534  0000   FTCHMS, 0              /FETCH MESSAGE.
02535  0000   MTFPAG, 0
02536  0000          0

```

```
2828   02537  1703          AVPT;(0;AVPT=.-2
2829   02540  2573
2830          2537
2831
2832   02573  0000
2833   02574  1076
2834   02575  6000
2835   02576  1221
2836   02577  0537
2837          2600   PAGE
```

```
             /MTTIME.
             /SYTEM TIMER.
             /KEEP TRACK OF SYSTEM TIME AND TIME OUT QUEUE.
             /THE TIME OUT Q (TOQ) IS BUILT OF 2-WORD TIMEOUT NODES.
             /LAYOUT OF TIME OUT NODE:
             /W0:    LINK TO NEXT NODE
             /W1:    PTR TO TCB[8] (AC) OF TASK;
             /       IF W1=0 THE TASK HAS BEEN REINSERTED IN RUNQ, AND THIS
             /       NODE MUST BE DELETED FROM TOQ.
             /
             /INSERTING A TSK IN TOQ:
             /1.     DELETE TSK FROM RUNQ.
             /2.     REQUEST 2-WORD NODE.
             /3.     LET W1 OF NODE PT TO TSKS AC, LINK NODE IN TOQ.
             /4.     LET TCB[5] PT TO NODE W1.
             /       THE LATTER IS NEEDED TO DELETE THE TSK FROM TOQ WHEN IT
             /       IS  REINSERTED IN RUNQ (FOR WHATEVER REASON).
             /THE TIME OUT VALUE IS KEPT IN AC OF THE TASK.
             /EACH 0.1 SEC THE TIME OUT VALUE IS INCREMENTED. ON OVERFLOW THE
             /TASK IS REINSERTED IN THE RUNQ.
             /NOTE! EXCEDING TIME OUT LIMITS IS THUS INDICATED BY AC=0 AT
             /RETURN.

             /TIMELP,        CALL
             /               WTRP      /WAIT FOR CLOCK INTERRUPT.
             /               TIMEMS
02600  7200  TIMEIN, CLA
02601  2267          ISZ     TIME
02602  5205          JMP     .+3
02603  2270          ISZ     TIME+1
02604  7000          NOP
02605  1377          TAD     (TQHEAD /INSPECT TIME OUT Q.
02606  3150          DCA     MON0      /PTR TO PRECESSOR.
02607  1550  TOQLP,  TAD I   MON0      /\GET PTR TO NEXT NODE.
02610  7450          SNA               /END REACHED?
02611  5242          JMP     TOQEX
02612  3010          DCA     X0        /PTR TO NODE.
02613  6202          CIF 0             /\DEAF. POSSIBLY CHANGING TSK'S AC.
02614  1410          TAD I   X0        /\GET PTR TO TSKS AC.
02615  7450          SNA               /\0 INDICATES TASK NO LONGER IN TOQ.
02616  5230          JMP     TOQFN     /FREE THIS NODE.
02617  3151          DCA     MON1      /\PTR TO TSKS AC.
02620  2551          ISZ I   MON1      /\INCREMENT TIME OUT VALUE.
02621  5240          JMP     TOQLP2    /NO OVERFLOW.
02622  6002          IOF               /\OVERFLOW. INSERT TSK IN RUNQ.
02623  1140          TAD     M4        /\
02624  1151          TAD     MON1      /\
02625  3155          DCA     INT1      /\PTR TO TCB[4] WAIT WORD.
02626  4776'         JMS     INRUNQ    /\
02627  6001          ION               /\
02630  1550  TOQFN,  TAD I   MON0
02631  3151          DCA     MON1      /PTR TO NODE TO BE DELETED
02632  1551          TAD     MON1      /DELETE FROM TOQ
02633  3550          DCA     MON0
02634  1151          TAD     MON1      /PTR TO NODE IN AC
```

```
02635  4572           JMS I   ZFN
02636  4334                   AVL2
02637  5207           JMP     TOQLP
02640  1550   TOQLP2, TAD I   MONO
02641  5206           JMP     TOQLP-1
02642  1375   TOQEX,  TAD     (-TIMEMS+2
02643  3774'          DCA     MTTIME-3
02644  7330   WTRPMT, AC4000
02645  4773'          JMS     CURWT
02646  5772'          JMP     SCED

              /INSERT CURRENT TSK IN TOQ.
02647  0000   TQIN,   0               /\
02650  4771'          JMS     GN2     /\GET 2-WORD NODE.
02651  1263           TAD     TQHEAD  /\
02652  3410           DCA I   X0      /\
02653  1010           TAD     X0      /\
02654  3263           DCA     TQHEAD  /\APPEND IN TOQ.
02655  7325           AC3             /\
02656  1164           TAD     CURTSK  /\
02657  3410           DCA I   X0      /\PTR TO TSKS AC IN NODE W1.
02660  1010           TAD     X0      /\
02661  3564           DCA I   CURTSK  /\TCB[5] PT TO NODE W1.
02662  5647           JMP I   TQIN    /\

02663  0000   TQHEAD, 0               /HEAD OF TIME OUT QUEUE.

02664  5067           MTTIME          /SENDER WORD.
02665  0000           0               /LINK WORD
02666  0000   TIMEMS, 0               /SOME STATUS MIGHT GO HERE.
02667  0000   TIME,   0
02670  0000           0
              IFDEF MULTIP <
              IFZERO SYTIME-MULTIP <
              /CONNECTED ROUTINE. COUNTS 55 MULTIPLEXER INTERRUPTS
              /BEFORE IT REPORTS TIMEMS.
              MULINTR,0               /!
                      T1ON            /!
                      ISZ     MULTCT  /!
                      JMP     FSTEXT  /!
                      TAD     M55     /!
                      DCA     MULTCT  /!
                      IEXIT   /!
                              TIMEMS  /!
                      DECIMAL
              MULTCT, -55
              M55,    -55
                      OCTAL
              >>
```

```
                /APP. A4.   MC8.4. MONITOR CALLS AND RELATED STUFF.
                /***** MONITOR CALLS AND RELATED STUFF *****
                /
                /BODY OF VIRTUAL READ DATAFIELD ROUTINE.
                /CALLED USING COMMON PART OF PAGE 0 AS FOLLOWS:
                /        JMS      VVRDF
                /        ....
                /VVRDF, 0
                /        IOF
                /        CIF 0
                /        JMP I    (VRDF1
                /NOTE: PRESERVE LINK, AC:=VFLD#, DF:=IF.
02671   7630    VRDF1,   SZL CLA          /!PRESERVE L.
02672   7307             AC4              /!
02673   6214             RDF              /!
02674   7112             CLL RTR          /!
02675   7010             RAR              /!ACT DF IN AC.
02676   1170             TAD      ZCORMAP /!
02677   3150             DCA      MONO    /!
02700   6201             CDF 0            /!
02701   1550             TAD I    MONO    /!GET VFLD#
02702   0130             AND      C77     /!REMOVE RUBBISH
02703   3004             DCA      MONAC   /!
02704   1370             TAD      (VVRDF  /!
                /FALL INTO RETURN JUMP

                RTN,                      /AC IN MONAC, KEEP L, PTR TO PC IN AC.
02705   3150             DCA      MONO    /!STORE PTR TO PC OF CALLING TSK.
02706   1052             TAD      CDFUF+1 /!GET CDF TO CALLING FLD.
02707   1120             TAD      C2      /!CHANGE INTO CIF CDF
02710   3311             DCA      .+1     /!
02711   6203             CIF CDF          /!TO CALLING FLD
02712   1550             TAD I    MONO    /!GET PROGRAM COUNTER
02713   3150             DCA      MONO    /!
02714   1004             TAD      MONAC   /!
02715   6001             ION              /!
02716   5550             JMP I    MONO


                /BODY OF MSNODE ROUTINE.
                /CALLED USING COMMON PART OF PAGE 0 AS FOLLOWS:
                /TO REQUEST A MESSAGE:
                /                 CLA
                /                 JMS VMSNODE
                /TO RETURN A MESSAGE:
                /                 TAD (MS[2]
                /                 JMS VMSNODE      /RETURN CONTENTS OF MS[2]
                /                                  /IN AC.
                /VMSNODE,0
                /         IOF
                /         CIF CDF 0
                /         JMP I   XMSNOD1
02717   7450    MSNOD1,  SNA              /!REQUEST OR RETURN?
02720   5330             JMP      MSNREQ  /!REQUEST.
02721   3150             DCA      MONO    /!RETURN. SET PTR TO MS[2]
```

```
02722  7344            ACM2              /!
02723  1150            TAD       MONO    /!GET PTR TO MS[0].
02724  4572            JMS I     ZFN     /!
02725  4350                      AVL5
02726  1550            TAD I     MONO    /!FETCH CONTENTS OF MS[2].
02727  5335            JMP       MSNOD2  /!
02730  4767' MSNREQ,   JMS       GN5     /!REQUEST. X0 PTS TO MS[0]-1
02731  3410            DCA I     X0      /!
02732  3410            DCA I     X0      /CLEAR MS[1] LINKWORD.
02733  1010            TAD       X0      /!
02734  7001            IAC               /!
02735  3004  MSNOD2,   DCA       MONAC   /!
02736  1366            TAD       (VMSNODE/!GET PTR TO PC.
02737  5305            JMP       RTN     /!RETURN JUMP INTO CALLING TSK.

             /HELP ROUTINE FOR NORMAL MONITOR CALLS.
             /GET SECOND ARGUMENT.
02740  0000  INARG2, 0
02741  4051            JMS       CDFUF
02742  1454            TAD I     MONPC
02743  6201            CDF 0
02744  2054            ISZ       MONPC
02745  3162            DCA       MARG2
02746  5740            JMP I     INARG2

02747  2537            AVPT;(0;AVPT=.-2
02750  2765
       2747

02765  0000
02766  0014
02767  4346
02770  0045
02771  4332
02772  0617
02773  0545
02774  5064
02775  5114
02776  1076
02777  2663
       3000  PAGE


```

```
              /MONITOR ENTRY.
              /CALLED USING COMMON PART OF PAGE 0 AS FOLLOWS:
              /       JMS     VCALL
              /       ....
              /VCALL, 0
              /       CIF 0
              /       IOF
              /       JMP I   (MONITOR
              /NOTE! A POSSIBLE SCEDREQ MAY HAVE SET SCDREQ TO 0.

              /SCHEDULING IS INHIBITED DURING MONITOR CALLS BY SETTING SCDINH
              />0. A SCDREQ IS SIGNALLED BY CLEARING SCDREQ. AT
              /END OF EACH MONITOR CALL SCDREQ IS TESTED TO SEE WETHER
              /SCHEDULING IS REQUIRED (SEE MONEX AND MONOUT).
              /TWO TYPES OF MONITOR CALLS ARE DISTINGUISHED:
              /NORMAL CALLS:  CALLS DIRECTLY FROM A TASK
              /SPECIAL CALLS: SOME PAGE 0 SERVICE ROUTINE CALLED THE MONITOR.
              /PC IS USED TO DISTINGUISH BETWEEN THESE CALLS:
              /PC<VERHLT+1    CALL FROM VRCDF
              /PC=VERHLT+1    CALL FROM VERHLT
              /PC>VERHLT+1
              /  <100         CALL FROM VCDF
              /PC>100         CALL FROM TASK

03000  3004   MONITOR,DCA     MONAC   /-
              IFNDEF PDP8E <
                      RDF             /-
                      DCA     MON0    /-SAFE DF FOR CDFINS.
              >
              IFDEF PDP8E <
03001  6004           GTF             /-GET L +GTFL.
03002  0001           AND     X6000   /-
03003  7004           RAL
              >
03004  7006           RTL             /-TRICK! LINK TO B10
03005  6214           RDF             /-
03006  7112           CLL RTR;RAR     /-
03007  7010
03010  1167           TAD     SCDINH  /-ADD (OVERRIDE PRIO SCED)-CONDITION.
03011  2167           ISZ     SCDINH  /-INHIBIT SCHEDULING DURING MONCALLS.
03012  6001           ION             /-SCHEDULING IS NOW INHIBITED.
03013  4051           JMS     CDFUF   /CDF CALLING FLD
03014  6214           RDF             /GET IN INSTRUCTION FIELD.
03015  3007           DCA     MONFL
03016  1777           TAD I   (VCALL
03017  3054           DCA     MONPC   /STORE PC OF CALLING TASK.
03020  1054           TAD     MONPC
03021  0145           AND     C7700
03022  7640           SZA CLA
03023  5255           JMP     MONIT2  /NORMAL CALL.
03024  1054           TAD     MONPC   /SPECIAL CALL. WHICH ONE?
03025  1376           TAD     (-VERHLT-2
03026  7450           SNA
03027  4576           SYSHLT          /ERHLT
03030  7710           SPA CLA
```

```
03031    5242                JMP       VRCDF1    /VRCDF

                 VCDF1,                          /VCDF.
03032    1775                TAD I     (VVRCDF   /WE SAVED AC THERE
03033    3004                DCA       MONAC
03034    1774                TAD I     (VVCDF    /GET PC OF CALLING TASK.
03035    3054                DCA       MONPC
03036    1773                TAD I     (VCDFTM   /GET PTR IN FLDTABLE
03037    6201                CDF 0
03040    3153                DCA       MON3
03041    5772'               JMP       FCSWAP    /ISSUE A SWAPREQUEST IF POSSIBLE.

                 /DO CDF TO REQUESTED FIELD.
                 /CALL OF MINIMUMTIME CDF ROUTINE.
                 /IF THE REQUESTED FLD IS IN CORE, STORE ITS CDF IN CALLING FLD.
03042    7040    VRCDF1, CMA
03043    1775                TAD I     (VVRCDF   /GET PC
03044    3054                DCA       MONPC
                                                 /WE SAVED THE PC BEFORE THE CALL OF
                                                 /VRCDF. SO THE FIRST INSTR EXECUTED BY
                                                 /THE TSK WILL BE: CALLING VRCDF.
                                                 /IF THE REQUESTED FLD IS NOT IN CORE, TSK
                                                 /IS SAVED WITH REQUESTED DF, AND THIS
                                                 /EXTRA CALL WILL DO A CHECK WHICH WOULD
                                                 /PROBABLY OCCUR SOON ANYHOW.
                                                 /IF REQUESTED DF IN CORE, WE USE THIS
                                                 /EXTRA CALL TO SET THE PROPER DF, WHICH
03045    6201                CDF 0               /COSTS NO TIME.
03046    1164                TAD       CURTSK    /GET REQUESTED DATAFIELD
03047    1124                TAD       C10
03050    3010                DCA       X0        /PTR TO TCB[14]; REQCDF.
03051    1410                TAD I     X0
03052    4771'               JMS       FCHECK    /CHECK IF IT IS IN CORE
                                     /IF NOT THE TSK IS SAVED WITH THE NEW DATAFIELD.
03053    4051                JMS       CDFUF     /IT IS IN CORE.
03054    5770'               JMP       VRCDF3    /STORE THE CDF IN CALLING FLD
                                                 /AND FALL INTO MONEX.
```

```
          /NORMAL MONITOR CALLS.
          /CALLING SEQUENCE IN TASK:
          /         CALL
          /                   MONFUNC /FUNCTIONWORD
          /                   ...     /POSSIBLE ARGS.
          /         ...               /POSSIBLE ERROR RETURN.
          /         ...               /NORMAL RETURN.
          /
          /THERE ARE ABOUT 25 DIFFERENT MONCALLS DISTINGUISHED BY THE
          /FUNCTONWORD. THE CALL IS SPECIFIED IN B6-10 OF THE FUNCTION
          /WORD.  B11 OF THE FUNCTIONWORD IS SET IF THE CALL USES
          /ARGUMENTS.

          /SOME CALLS ACCEPT OPTIONS THAT ARE SPECIFIED IN THE OTHER BITS
          /OF THE FUNCTIONWORD.
          /
          /LIST OF MONITORCALLS.
    0200  STALL=0↑2+TIMEOUT
          /         OPTIONS:  SWPOUT.
          /         INSERT TSK IN TIMEOUT QUEUE (SEE TIMEOUT OPTION).
    0003  SNDMS=1↑2+1       /(SEND A MESSAGE)
          /         OPTIONS:  NONREP, DFPARM.
          /         SEND MS TO A TSK.
          /         ARG1:   PTR TO MS[2]
          /         ARG2:   STSK# OF ADDRESSED TSK.
    0005  WTRP=2↑2+1        /(WAIT FOR REPORT).
          /         OPTIONS:  TIMEOUT, SWPOUT.
          /         EITHER SPECIFIC WAIT, (I.E., WAIT FOR THE REPORT ON THE
          /                 MS  PTED TO BY ARG1),
          /         OR GENERAL WAIT REPORT (WAIT FOR THE FIRST INCOMING
          /                 REPORT).
          /         AT RTN AC =PTR TO MS[2] OF REPORTED MS, L=1.
          /         ARG1:   SPECIFIC WAIT: PTR TO MS[2], GENERAL WAIT: 0.
    0007  SNDWTR=3↑2+1
          /         OPTIONS:  AS IN 1 AND 2.
          /         SEND MESSAGE AND WAIT FOR ITS REPORT.
          /         ARG1:   PTR TO MS[2] OF SENT MS.
          /         ARG2:   STSK# OF ADDRESSED TSK.
    0013  RP=5↑2+1                  /(REPORT A MESSAGE).
          /         REPORT A MS TO ITS SENDER (I.E. INSERT IT IN ITS RPQ).
          /                 IF MS[0]B0=0 NO ONE IS WAITING FOR THE REPORT,
          /                 SO THE MS CAN IMMEDIATELY BE FREED.
          /         ARG1:   PTR TO MS[2] OF REPORTED MS.
    0014  WTMS=6↑2          /(WAIT FOR MESSAGE).
          /         OPTIONS:  KEEP, TIMEOUT, SWPOUT.
          /         WAIT UNTIL A MS IS SENT TO THIS TSK.
          /                 IF KEEP OPTION SPECIFIED, KEEP TSK CLAIMED:
          /                 I.E., WAIT FOR MS SENT BY THE SAME TASK AS BEFORE
          /         AT RTN: AC=PTR TO MS[2] OF RECEIVED MS, L=0.
    0016  DISINTR=7↑2       /(DISCONNECT FROM INTERRUPT).
          /         DISABLE DEVICE INTERRUPT.
          /                 CRITICAL SYNCHRONIZATION!!! PATCH THE SKPCHN SUCH
          /                 THAT FLAGS OF THE SPECIFIED DEVICE ARE NOT
          /                 TREATED AS INTERRUPTS.
          /         CLEARS AC.
```

```
3187                  /         AC=SKIPIOT.
```

```
     0021  CNINTR=10↑2+1    /(CONNECT A ROUTINE TO INTERRUPTSLOT)
           /         A ROUTINE TO TREAT INTERRUPTS OF THE SPECIFIED DEVICE IS
           /                   ATTACHED TO THE DEVICES INTSLOT. THE ROUTINE MUST
           /                   BE CORERESIDENT AND RESIDE IN THE CALLER'S DF.
           /         DENOTE CURTSK AS SENDER IN THE COMMUNICATION MS, IF ANY.
           /         CLEARS AC.
           /         AC=SKIPIOT.
           /         ARG1:     PTR TO ENTRY POINT OF ROUTINE TO BE CONNECTED.
           /         ARG2:     PTR TO MS[2] OF COMMUNICATION MS, IF ANY,
           /                   0 OTHERWISE.
     0023  CLINTR=11↑2+1    /(CLAIM AN INTSLOT)
           /         ATTACH A MS TO THE SPECIFIED INTSLOT. THE MS IS REPORTED
           /                   TO THE TSK, EACH TIME AN INTERRUPT ON THAT DEVICE
           /                   OCCURS.
           /         CLEARS AC.
           /         AC=SKIPIOT.
           /         ARG1:     CLEARIOT
           /         ARG2:     PTR TO MS[2] OF MS TO BE ATTACHED.
     0025  FRINTR=12↑2+1    /(FREE INTSLOT)
           /         SET INTSLOT OF SPECIFIED DEVICE IN DISCARD MODE (I.E.,
           /                   INTERRUPTS OF THAT DEVICE ARE DISCARDED).
           /         CLEARS AC.
           /         AC=SKIPIOT.
           /         ARG1:     CLEARIOT (TO CLEAR THE DEVICEFLAG).
     0027  SIMINTR=13↑2+1   /(SIMULATE INTERRUPT)
           /         ENTER A DEAF SECTION THAT MUST BE LEFT BY
           /                   "CIF 0; IEXIT; 0". THIS IS MOST OFTEN USED TO
           /                   START UP LOOPING CONNECTED ROUTINES. WHEN THE
           /                   RETURN FROM INTERRUPT "IEXIT" IS TAKEN,
           /                   THE CALLER IS CONTINUED.
           /         THE DEAF SECTION MUST RESIDE IN CALLER'S DF.
           /         LEAVES AC UNCHANGED.
           /         ARG1:     ENTRYPOINT OF DEAF SECTION.
     0031  REQPAG=14↑2+1    /(REQUEST PAGES).
           /         1-37 CONSECUTIVE PAGES OF CORE ARE REQUESTED.
           /                   AT RTN: AC B0-4 =PAGE# OF FIRST PAGE,
           /                                  B6-11 =VFLD#.
           /         ARG1:     B7-11 LENGTH (1<=LENGTH<=37), B5 CORERESIDENT.
     0033  RTNPAG=15↑2+1    /(RETURN PAGES PREVIOUSLY REQUESTED USING
           /                             REQPAG).
           /         CLEARS AC.
           /         AC B0-4 =PAGE# OF FIRST PAGE, B6-11 =VFLD#.
           /         ARG1:     B7-11 LENGTH (1<=LENGTH<=37), B5 CORERESIDENT.
     0034  REQFLD=16↑2      /(REQ WHOLE FLD FOR DATA STORAGE).
           /         THE VFLD# IS RETURNED IN AC B6-11.
     0036  RTNFLD=17↑2      /(RETURN FLD PREVIOUSLY REQUESTED USING REQFLD).
           /         CLEARS AC.
           /         AC B6-11 =VFLD#.
     0041  REQSTL=20↑2+1    /(REQUEST ENTRY IN STATIC TASK LIST).
           /         A FREE ENTRY IN THE STATIC TSK LIST IS SEARCHED, ARG1 AND
           /                   ARG2 ARE ASSIGNED TO IT.
           /                   NOTE: THE ENTRY IS FREEED AGAIN BY CLEARING
           /                   ITS FIRST WORD (NO SPECIAL MONITORCALL!!).
           /         AT RETURN AC=STSK#.
           /         ARG1:     CONTENTS OF TCB[15] (>0).
```

```
      /       ARG2:   B7-11 LENGTH, B5 TSK CORERESIDENT, B3 TSK USES
      /               OTO PART OF PAGE 0, B0-2 TSKS PRIO,
0042  REQTCB=21↑2     /(ATTACH TCB TO ENTRY IN STL),
      /       SET UP TCB USING THE SPECIFIED ENTRY IN STL,
      /       AC=STSK#.
      /       AT RETURN AC=PTR TO TCB[5].
0044  RTNTCB=22↑2     /(RETURN TCB).
      /       DETACH TCB FROM ENTRY IN STL.
      /       NOTE!!!!! TSK MUST BE STOPPED BEFORE!!!!.
      /       CLEARS AC,
      /       AC=STSK#.
0046  LOCK=23↑2       /(LOCK FLD IN CORE)
      /       LOCK THE CURRENT DATAFIELD IN CORE,
      /       LEAVES AC UNCHANGED,
0050  UNLOCK=24↑2     /(UNLOCK FLD)
      /       UNLOCK THE CURRENT DATAFIELD.
      /       LEAVES AC UNCHANGED.
0052  REQCDF=25↑2     /(REQUEST DATAFLD).
      /       SET UP FOR USING THE MINIMUMTIME CDF ROUTINE VRCDF.
      /               SUBSEQUENT CALLS ON VRCDF WILL CHANGE DF TO THE
      /               REQUESTED DF.
      /       LEAVES AC UNCHANGED.
      /       AC B6-11:       REQUESTED DF.
0054  EXIT=26↑2       /(DISMISS TSK AND TCB FROM CORE).
      /       THE CORE OCCUPIED BY THE TSK IS RETURNED, AS IS ITS TCB.
      /               THE TSK IS RESTARTED FROM SCRATCH BY SENDING A MS
      /               TO IT.
0056  STOP=27↑2       /(STOP A TSK).
      /       A TSK IS STOPPED. IT WON'T BE RERUN UNTIL THE RESUME
      /               COMMAND IS GIVEN FOR IT. IT IS REMOVED FROM THE
      /               TIMEOUT QUEUE. ALL OTHER WAITFUNCTIONS ARE
      /               NORMALLY PERFORMED AS LONG AS IT IS STOPPED.
      /       CLEARS AC.
      /       AC=STSK#.
0060  RESUME=30↑2     /(RESUME A TSK).
      /       RESUME A TSK PREVIOUSLY STOPPED BY THE STOP COMMAND.
      /               THE TSK WILL BE RESUMED AS SOON AS OTHER
      /               WAITFUNCTIONS EXPIRE. IF NO WAITS WERE IN, THE
      /               TSK IS RESUMED IMMEDIATELY.
      /       CLEARS AC.
      /       AC=STSK#.
0065  FILTCB=32↑2+1   /(FILL A TCB)
      /       AC=STSK#
      /       CLEARS AC.
      /       ARG1=PTR TO CONTENTS OF TCB.
```

```
                /LIST OF OPTIONS.
        4000    CHKONLY=4000
        4014    CHRCQ=WTMS+CHKONLY /CHECK RECEIVE Q.
                /       EQUAL TO WTMS. BUT IF THE TASK SHOULD BE SET TO WAIT IT
                /        IS INSTEAD CONTINUED WITH AC=0.
        4005    CHRPQ=WTRP+CHKONLY /CHECK REPORT Q.
                /       EQUAL TO WTRP. BUT IF THE TASK SHOULD BE SET TO WAIT, IT
                /        IS CONTINUED WITH AC=0.
        2000    NONREP=2000     /THE MS SENT NEED NOT BE REPORTED
                                /(CLEAR MS[0] B0).
        1000    KEEP=1000       /WAIT FOR MS WITHOUT CLEARING THE CLAIM WORD,
                                /( TCB[0]) SO THAT WE WAIT FOR A MS OF THE
                                /SAME TASK.
        0400    SWPOUT=400      /SWAP TSK OUT. THE CORE OCCUPIED BY
                                /THE TSK IS FREED AGAIN, BUT THE TCB IS KEPT, AS
                                /SOON AS THE TSK BECOMES RUNNABLE A FRESH COPY IS
                                /FETCHED FROM DISK.
        0200    TIMEOUT=FWTTM
        0200    TIMEOUT=200     /INSERT TSK IN TIMEOUT Q. AS LONG AS THE TSK IS
                                /NOT RUNNABLE EACH 0.1 SEC AC WILL BE
                                /INCREMENTED.
                                /ON OVERFLOW THE TSK IS FORCED IN THE RUNQ (WITH
                                /AC=0!)
        0100    DFPARM=100      /STORE ACT FLD# OF CURRENT DF IN
                                /MS[2] B6-8.
```

```
                        /NORMAL CALL.
                        /DF IS CALLING FLD.
      03055  1454      MONIT2, TAD I    MONPC     /GET FUNCTION WORD
      03056  3160              DCA      MONFUNC
      03057  2054              ISZ      MONPC
      03060  1454              TAD I    MONPC
      03061  3161              DCA      MARG1     /STORE FIRST ARG.
      03062  6201              CDF 0
                        IFNDEF PDP8E < /UPDATE CDFINS
                               TAD      MON0
                               TAD      CCDF
                               DCA      CDFINS
                        >
                        IFDEF STATX <
      03063  1160              TAD      MONFUNC
      03064  0272              AND      MON76     /2*FUNCTIONCODE
      03065  1367              TAD      (MONSTAT-1
      03066  3010              DCA      X0        /PTR IN MONITOR STATISTICS.
      03067  2410              ISZ I    X0
      03070  5273              JMP      .+3
      03071  2410              ISZ I    X0
      03072  0076      MON76,  76                 /INNOCENT AND.
                        >
      03073  1160              TAD      MONFUNC
      03074  7010              RAR
      03075  0125              AND      C37
      03076  1303              TAD      MONJMP
      03077  3302              DCA      .+3
      03100  7430              SZL                /ARGUMENTS USED?
      03101  2054              ISZ      MONPC     /YES.
      03102  5000              JMP

      03103  5704      MONJMP, JMP I    .+1
      03104  3315              XSTALL             /INSERT IN TIMEOUT QUEUE.
      03105  3200              XSNDMS             /SEND MESSAGE
      03106  3236              XWTRP              /WAIT FOR REPORT
      03107  3200              XSNDWTR            /SEND MESSAGE AND WAIT FOR ITS REPORT.
      03110  3144              XERROR
      03111  3251              XRP                /REPORT A MESSAGE
      03112  3303              XWTMS              /WAIT UNTIL A MESSAGE IS RECEIVED
      03113  3677              XDISINTR           /DISCONNECT A DEVICE FROM INTERRUPT
      03114  3701              XCNINTR            /CONNECT A ROUTINE TO INTSLOT
      03115  3714              XCLINTR            /CLAIM (ATTACH A MS TO) AN INTSLOT
      03116  3737              XFRINTR            /FREE INTSLOT (BACK INTO DISCARD MODE)
      03117  3741              XSIMINTR           /SIMULATE INTR (START UP LOOPING
                                                  /CONNECTED ROUTINE)
      03120  4042              XREQPAG            /REQUEST A NUMBER OF CONSECUTIVE PAGES
      03121  4043              XRTNPAG            /RETURN A NUMBER OF CONSECUTIVE PAGES
      03122  4051              XREQFLD            /REQUEST FLD FOR DATA STORAGE
      03123  4060              XRTNFLD            /RETURN FLD
      03124  4132              XREQSTL            /REQUEST ENTRY IN STL AND SET ITS
                                                  /CONTENTS.
      03125  4160              XREQTCB            /ATTACH TCB TO ENTRY IN STL.
      03126  4034              XRTNTCB            /DETACH TCB FROM ENTRY IN STL AND FREE
                                                  /IT.
```

```
3368    03127  4201            XLOCK           /LOCK DF IN CORE
3369    03130  4200            XUNLOCK         /UNLOCK DF
3370    03131  3600            XREQCDF         /SET UP FOR USING VRCDF ROUTINE
3371    03132  3267            XEXIT           /DISMISS TSK AND TCB FROM CORE
3372    03133  3630            XSTOP           /STOP A TSK.
3373    03134  3647            XRESUME         /RESUME A TSK.
3374    03135  3144            XERROR          /PROCES IDENTIFY.
3375    03136  3466            XFILTCB         /FILL TCB
3376    03137  3144            XERROR
3377    03140  3144            XERROR
3378    03141  3144            XERROR
3379    03142  3144            XERROR
3380    03143  3144            XERROR
3381                   IFNZRO .-MONJMP-41 <ALLOCATION ERROR!>
3382
3383
3384    03144  4576    XERROR, SYSHLT
3385
3386    03145  2747            AVPT;(0;AVPT=.-2
3387    03146  3166
3388           3145
3389
3390    03166  0000
3391    03167  4731
3392    03170  0726
3393    03171  2000
3394    03172  2010
3395    03173  0042
3396    03174  0025
3397    03175  0020
3398    03176  7753
3399    03177  0054
3400           3200    PAGE
```

```
/***** MESSAGE EXCHANGE *****
/
/TASKS ARE SYNCHRONIZED AND COMMUNICATE TO ONE ANOTHER BY SENDING
/ AND REPORTING MESSAGES.
/A MESSAGE CONSISTS OF 5 CONSECUTIVE LOCATIONS IN FLD 0.
/MESSAGE LAYOUT:
/W0     SENDERWORD.
/       B0=0:   MESSAGE NEED NOT BE REPORTED.
/       B0=1:   MESSAGE MUST BE REPORTED.
/       B1-11   BIT 1-11 OF TCBPTR OF SENDER (B0 OF TCBPTR ALWAYS
/               SET!).
/W1     LINKWORD (USED TO LINK THE MS IN RCQ OR RPQ).
/W2-4   USER DEFINED INFORMATION.
/MESSAGES ARE DRAWN FROM A POOL OF STORAGE AND !MUST! BE RETURNED
/THERE AS SOON AS THEY ARE NO LONGER USED.
/
/REQUESTING AND RETURNING MESSAGES.
/A TASK REQUESTS A MESSAGE BY EXECUTING THE "MSREQ"-INSTRUCTION
/       WITH AC=0!.
/       THE INSTRUCTION IS EXECUTED USING THE GENERAL PART OF
/       PAGE 0. IT RETURNS A PTR TO MS[2] IN AC.
/A TASK RETURNS A MESSAGE BY EXECUTING THE "MSFREE"-INSTRUCTION
/       WITH AC=PTR TO MS[2].
/       THE INSTRUCTION ALSO USES THE GENERAL PAGE 0.
/       IT RETURNS THE MESSAGE TO THE POOL OF STORAGE IN FLD0
/       AND RETURNS THE CONTENTS OF MS[2] IN AC. SO THE
/       INSTRUCTION CAN SIMULTANEOUSLY BE USED TO FETCH INFO FROM
/       THE MS.
/IN PRINCIPLE A TASK SENDS A MESSAGE TO ANOTHER TASK THEREBY
/ACTIVATING THE OTHER TASK. WHEN THE SECOND TASK IS DONE IT
/REPORTS THE MESSAGE. THE SENDER MAY WAIT FOR THE REPORT, CHECK
/THE CONTENTS OF THE MESSAGE, THUS KNOWING THAT THE MS WAS
/ACCEPTED AND THE JOB PROPERLY EXECUTED.
/BELOW *A AND *B INDICATE SEPARATE TASKS. *A EXECUTES THE
/MONITORCALL. THE FOLLOWING MONITORCALLS HAVE TO DO WITH
/MESSAGES:
/SNDMS  SEND MESSAGE.
/       *A SENDS A MS TO *B.
/       ARG1=PTR TO MS[2]. ARG2=STSK# OF *B.
/       WHEN A TASK IS FULLY INACTIVE (TASK ON DISK, AND NO TCB
/       IN CORE) A SNDMS COMMAND MAY ACTIVATE IT. THE STSK# OF *B
/       IDENTIFIES THE ENTRY IN THE STL CORRESPONDING TO *B.
/       THE MONITOR CHECKS WETHER A TCB OF *B IS IN CORE, IF IT
/       IS NOT IT ALLOCATES ONE FOR *B.
/       SUBSEQUENT IT DENOTES THE TCBPTR OF *A IN MS[0],
/       BUT BIT0 IS CLEARED AGAIN IF THE NONREP OPTION WAS
/       SPECIFIED. FINALLY *B "RECEIVES" THE MS.
/WTRP   WAIT FOR REPORT.
/       WAIT FOR A MESSAGE TO BE REPORTED.
/       IF ARG1=0 ANY REPORT THAT IS RECEIVED WILL REACTIVATE THE
/       TASK, OTHERWISE ARG1 IS A PTR TO MS[2] OF THE MS WHOSE
/       REPORT IS ALLOWED TO REACTIVATE THE TSK.
/SNDWTR SEND MS AND WAIT FOR ITS REPORT.
/       JUST A COMBINATION OF THE TWO FORMER CALLS.
/RP     REPORT ON A MS.
```

```
                /       ARG1= PTR TO MS[2] OF THE REPORTED MS,
                /       THE MONITOR LOOKS IN THE SENDERWORD (MS[0]) TO FIND OUT
                /       WHICH TASK SENT THE MS. IF MS[0]B0 =0 NO TSK IS
                /       INTERESTED IN THE REPORT AND THE MS IS RETURNED TO THE
                /       POOL OF STORAGE IN FLD0, ELSE THE TSK THAT SENT THE MS
                /       "RECEIVES" ITS REPORT,
                /WTMS   WAIT FOR MS,
                /       THE TASK IS REACTIVATED AS SOON AS IT "RECEIVES" AN
                /       APPROPRIATE MS, NORMALLY THIS CALL DECLAIMS THE TSK, IT
                /       IS AGAIN READY TO  RECEIVE MSS FROM ANY TASK, AS SOON
                /       AS IT RECEIVES ONE, IT IS CLAIMED BY THAT TASK,
                /       IF THE KEEP OPTION IS SPECIFIED THE TSK IS NOT DECLAIMED,
                /       IT THEN ONLY ACCEPTS MSS FROM THE TASK THAT CLAIMS IT,
                /
                /RECEIVING MESSAGES AND REPORTS,
                /IN THE TCB HEADS OF TWO QUEUES ARE ALLOCATED,
                /ONE IS THE RECEIVE Q (HEAD IN TCB[1]), THE OTHER IS THE
                /REPORT Q (HEAD IN TCB[3]), BESIDES THIS TCB[0]
                /IDENTIFIES THE CLAIMING TASK (SEE KEEP OPTION)
                /AND TCB[2] HOLDS A PTR TO THE REPORT THAT IS ALLOWED TO
                /REACTIVATE THE TSK (SEE WTRP).
                /IF WTMS OR WTRP IS EXECUTED, FIRST THE APPROPRIATE QUEUES ARE
                /INSPECTED. IF NO MESSAGE IN THESE QUEUES IS ALLOWED TO
                /REACTIVATE THE TASK, THEN THE WAITCONDITION IS SET, OTHERWISE
                /THE MSPTR OF THAT MS IS DENOTED IN THE TASK'S AC AND THE TASK
                /IS CONTINUED.
                /IF A MS IS SENT OR REPORTED, FIRST WE CHECK WETHER IT IS ALLOWED
                /TO REACTIVATE THE RECEIVER, IF IT IS THE RECEIVER'S
                /WAITCONDITION IS ERASED AND THE MSPTR DENOTED IN ITS AC,
                /OTHERWISE SENT MSS ARE ADDED TO THE RECEIVE Q
                /AND REPORTED MSS ARE ADDED TO THE REPORT Q,
                      /SEND MESSAGE
                      XSNDMS,
                      /SEND MS AND WAIT FOR REPORT.
                      /ARG1=PTR TO MS[2]
                      /ARG2=STSK# OF RECEIVER.
03200  4777'          XSNDWTR,JMS      INARG2
03201  1131                   TAD      C100;DFPARM=100
       0100
03202  0160                   AND      MONFUNC /DFPARM OPTION SPECIFIED?
03203  7640                   SZA CLA
03204  1007                   TAD      MONFL   /IF IT IS, DENOTE ACT DF IN MS.
03205  0123                   AND      C7      /GET DF
03206  7106                   CLL RTL;RAL
03207  7004
03210  1561                   TAD I    MARG1
03211  3561                   DCA I    MARG1   /INSERT IN MS[2]

                      XSND2,
03212  1162                   TAD      MARG2   /STSK# OF RECEIVER
03213  4776'                  JMS      GETTCB  /TCB[5] OF RECEIVER
03214  1135                   TAD      M1
03215  6002                   IOF              /\
03216  3155                   DCA      INT1    /\TCB[4] OF RECEIVER.
03217  7344                   ACM2             /\
03220  1161                   TAD      MARG1   /\
03221  3154                   DCA      INT0    /\PTR TO MS[0]
03222  7332                   AC2000;NONREP=2000/\
       2000
03223  0160                   AND      MONFUNC /\NONREP SPECIFIED?
03224  7164                   STL CMA RAL      /\3777 IF NONREP, 7777 ELSE
03225  0164                   AND      CURTSK  /\
03226  3554                   DCA I    INT0    /\
03227  1155                   TAD      INT1    /\
03230  4571                   JMS I    ZSNDREP /\SEND TO RECEIVER.
03231  6001                   ION              /\
03232  7307                   AC4              /WAS IT SNDWTR?
03233  0160                   AND      MONFUNC
03234  7650                   SNA CLA
03235  5775'                  JMP      MONEX   /NO.

                      /WAIT FOR REPORT.
                      /IF ARG1=0 ANY REPORT IS ACCEPTED, OTHERWISE ARG1=PTR TO MS[2]
                      /OF WAITED REPORT.
                      XCHRPQ,
03236  7346           XWTRP,  ACM3     /LET MONO PT TO TCB[2] WAITED REPORT WORD.
03237  1164                   TAD      CURTSK
03240  3150                   DCA      MONO
03241  1161                   TAD      MARG1
03242  7041                   CIA
03243  7440                   SZA
03244  1120                   TAD      C2
03245  3550                   DCA I    MONO    /INDICATE EITHER ZERO OR MINUS PTR TO
03246  4774'                  JMS      SMSG    /MS[0]. SEARCH RPQ FOR THIS MS.
03247  4000                            4000
                              /AC4000          /MS NOT IN RPQ. SET TASK TO WAIT.
```

```
03250  5316               JMP     TSWAIT



                  /REPORT A MS.
                  /ARG1=PTR TO MS[2].
03251  7344       XRP,    ACM2
03252  1161               TAD     MARG1
03253  6002               IOF               /\CALLING SNDREP.
03254  3154               DCA     INT0      /\PTR MS[0]
03255  1554               TAD I   INT0      /\GET CONTENTS OF SENDERWORD MS[0].
03256  7700               SMA CLA           /\ANYONE INTERESTED IN THIS REPORT?
03257  5263               JMP     XRP2      /\NO.
03260  4571               JMS I   ZSNDREP   /\YES. REPORT IT.
03261  6001       MONEX0, ION               /\
03262  5775'              JMP     MONEX
03263  1154       XRP2,   TAD     INT0      /\FREE THE MS.
03264  4572               JMS I   ZFN       /\
03265  4350                       AVL5      /\
03266  5261               JMP     MONEX0    /\

                  /EXIT.
                  /FULL TASK DISMISS FROM CORE.
                  /A COMPLETELY FRESH COPY IS SWAPPED IN AS SOON AS A MS IS SENT TO
                  / THE TSK. BEFORE THAT TCB IS RETURNED.
03267  3166       XEXIT,  DCA     SCDREQ    /FORCE SCHEDULING EVEN IF THERE IS A MS
                                  /FOR THIS TASK.
03270  4773'              JMS     RTNTSK    /DISMISS TASKS CODE FROM CORE.
03271  3054               DCA     MONPC     /SET PC TO START OVER AGAIN.
03272  4060               GET;    -4        /ANY MS IN RCQ. (BASE PTED TO TCB[5]).
03273  7774
03274  7640               SZA CLA
03275  5303               JMP     XWTMS     /YES
03276  4772'              JMS     CURWT     /\REMOVE CURTSK FROM RUNQ.
03277  7040               CMA               /\
03300  1162               TAD     MARG2     /\
03301  4771'              JMS     RNTCB     /\AC PTS AT STL[TSK,0].
03302  5770'              JMP     SCED      /\

                  /WAIT FOR MS.
                  /NO ARGUMENTS.
                  /IF NO KEEP OPTION CLEAR TCB[0]CLAIM WORD. THIS CAUSES MSS
                  /       OF ANY TASK TO BE ACCEPTED.
                  /ELSE DONT CLEAR TCB[0], KEEPING ON THE CLAIM.
                  XCHRCQ,
03303  1141       XWTMS,  TAD     M5
03304  1164               TAD     CURTSK
03305  3150               DCA     MON0      /PTR TO TCB[0] CLAIM WORD.
03306  1173               TAD     Z1000     /KEEP OPTION SPECIFIED?
03307  0160               AND     MONFUNC
03310  7650               SNA CLA           /
03311  3550               DCA I   MON0      /NO. CLEAR CLAIM WORD.
03312  4774'              JMS     SMSQ      /SEARCH MSQ FOR APPROPRIATE MS.
03313  2000                       2000
                          /AC2000           /NO SUITABLE MS IN Q. SET TASK TO WAIT.
```

```
03314  7410                SKP


                   /SET TASK TO WAIT,
                   /PERHAPS A WAITCONDITION IN AC.
03315  1173   XSTALL, TAD     Z1000
03316  4772'  TSWAIT, JMS     CURWT   /\REMOVE TASK FROM RUNQ.
03317  1133           TAD     C200;TIMEOUT=200 /\
       0200
03320  0160           AND     MONFUNC /\TIMEOUT OPTION SPECIFIED?
03321  7640           SZA CLA         /\
03322  4767'          JMS     TQIN    /\INSERT TASK IN TIMEOUT Q.
03323  6001           ION             /\
03324  1366           TAD     (400    /SWAPOUT OPTION SPECIFIED?
03325  0160           AND     MONFUNC
03326  7640           SZA CLA
03327  4773'          JMS     RTNTSK  /YES. RETURN PAGES CONTAINING TASKS CODE.
03330  5765'          JMP     MONOUT

03331  3145           AVPT;(0;AVPT=.-2
03332  3364
       3331
03364  0000
03365  0474
03366  0400
03367  2647
03370  0617
03371  3450
03372  0545
03373  4000
03374  3400
03375  0731
03376  4063
03377  2740
       3400           PAGE
```

```
              /SMSQ, SEARCH MESSAGE Q.
              /SEARCHES EITHER THE RCQ OR THE RPQ FOR MSS THAT ARE ALLOWED TO
              /REACTIVATE THE CURRENT TASK. IF THERE ARE, A MS IS PASSED TO THE
              /TSK AND DELETED FROM MSQ.
              /ELSE IF THE CHECK ONLY BIT WAS SET (B0 IN MONFUNC) WE CONTINUE
              /THE TASK WITH AC=0.
              /OTHERWISE WE RETURN FROM THIS SUBROUTINE IN ORDER TO SET THE
              /TASK TO WAIT.

              /SEARCH RCQ: MON0=PTR TO TCB[0] CLAIM WORD.
              /SEARCH RPQ: MON0=PTR TO TCB[2] WAITED RP WORD.
              /MON0*PTR TC HEAD OF Q -1.
              /ARG1*2000 IF WTMS, 4000 IF WTRP.
03400   0000  SMSQ,   0
03401   7350          AC3777
03402   0007          AND     MONFL
03403   3007          DCA     MONFL   /CLEAR TASKS LINK ANYHOW.
03404   1150          TAD     MON0    /PTR HEAD OF Q -1
03405   6002          IOF             /\
03406   3156          DCA     INT2    /\
03407   1150          TAD     MON0
03410   7001          IAC             /\
03411   7410          SKP             /\
03412   1551  SMSQLP, TAD  I  MON1    /\
03413   3151          DCA     MON1    /\PTR TO PRECESSOR MS.
03414   7140          CMA CLL         /\NOTE: 0 ENDS THE Q.
03415   1551          TAD  I  MON1    /\
03416   3154          DCA     INT0    /\PTR TO (NEXT MS)[0].
03417   7420          SNL             /\END OF Q?
03420   5241          JMP     SMSQEX  /\
03421   1600          TAD  I  SMSQ    /\FETCH WAIT CONDITIONS
03422   4314          JMS     CHKMS   /\MS ALLOWED TO REACTIVATE TSK?
03423   5212          JMP     SMSQLP  /\NO; KEEP SEARCHING.

              /PASS MS TO TASK AND DELETE MS FROM Q.
03424   2154  MSTOAC, ISZ     INT0    /\PTR TO MS[1] LINK WORD.
03425   1554          TAD  I  INT0    /\GET SUCCESSOR.
03426   3551          DCA  I  MON1    /\DELETE MS FROM Q.
03427   3554          DCA  I  INT0    /\CLEAR LINKWORD
03430   7330          AC4000          /\
03431   0600          AND  I  SMSQ    /\
03432   1007          TAD     MONFL   /\SET TASKS L TO INDICATE
03433   3007          DCA     MONFL   /\ SEND OR REPORT.
03434   7001          IAC             /\
03435   1154          TAD     INT0    /\GET PTR TO MS[2]
03436   3004  SETAC,  DCA     MONAC   /\STORE IN TASKS AC,
03437   6001          ION             /\
03440   5777'         JMP     MONEX

              /SEARCH FAILED.
              /IF CHECK ONLY BIT SET CONTINUE TASK WITH AC=0.
              /ELSE RETURN FROM SUBROUTINE.
03441   6001  SMSQEX, ION             /\
03442   1160          TAD     MONFUNC
03443   7710          SPA CLA         /CHECK ONLY?
```

```
03444  5236            JMP     SETAC   /YES. CONTINUE WITH AC=0.
03445  1600            TAD I   SMSQ
03446  2200            ISZ     SMSQ    /NO. RETURN.
03447  5600            JMP I   SMSQ

       3400    TSTTM=SMSQ
               /DETACH TCB FROM ENTRY IN STL.
               /PTR TO STL[TSK,0] IN AC.
03450  0000    RNTCB,  0
03451  3200            DCA     TSTTM   /TEMP
03452  1600            TAD I   TSTTM   /GET PTR TO TCB[5].
03453  1376            TAD     (11
03454  3010            DCA     X0      /PTR TO TCB[15]
03455  1600            TAD I   TSTTM   /PTR TO TCB[5]
03456  7500            SMA             /ANY TCB ACTUALLY ATTACHED TO STL?
03457  5264            JMP     RTNTC2  /NO. DONT TOUCH THE ENTRY.
03460  1141            TAD     M5      /PTR TO TCB[0].
03461  4572            JMS I   ZFN
03462  4356                    AVL20
               /BAD TRIC: FIRST DELETE TCB, AND THEN STILL GET SOME INFO OUT.
03463  1410            TAD I   X0      /GET CONTENTS OF TCB[15]; DISK ADDRESS,
03464  3600    RTNTC2, DCA I   TSTTM   /STORE IN STL.
03465  5650            JMP I   RNTCB

```

```
03466  1004  XFILTCB,TAD      MONAC
03467  4775'          JMS     GETTCB   /RETURNS POINTER TO TCB[5]
03470  1142           TAD     M6
03471  3010           DCA     X0
03472  7240           CLA CMA
03473  1161           TAD     MARG1
03474  3011           DCA     X1
03475  1141           TAD     M5
03476  3151           DCA     MON1
03477  1374           TAD     (-17
03500  3152           DCA     MON2
03501  4051  FILLP,   JMS     CDFUF
03502  1411           TAD I   X1
03503  6201           CDF 0
03504  3410           DCA I   X0
03505  2151           ISZ     MON1
03506  5311           JMP     .+3
03507  2011           ISZ     X1
03510  2010           ISZ     X0
03511  2152           ISZ     MON2
03512  5301           JMP     FILLP
03513  5236           JMP     SETAC

             /CHECK IF THIS MS CAN REACTIVATE THE TASK.
             /INT0=PTR TO MS[0]
             /IF MS IS RECEIVED, INT2 POINTS AT TCB[0], AC=2000
             /IF MS IS REPORTED, INT2 POINTS AT TCB[2], AC=4000.
             /SKIPS IF CHECK SUCCEEDS.
03514  0000  CHKMS,  0                /!\
03515  7710           SPA CLA /!\
03516  5322           JMP     CHKMS2  /!\
03517  7350           AC3777  /\!SEND.
03520  0554           AND I   INT0     /!\
03521  7410           SKP              /!\
03522  1154  CHKMS2,  TAD     INT0     /!\REPORT
03523  1556           TAD I   INT2     /!\COMPARE TO CLAIM WORD OR WAITED REP
03524  7640           SZA CLA          /!\
03525  1556           TAD I   INT2     /!\ANYTHING SPECIFIED?
03526  7640           SZA CLA          /!\
03527  5714           JMP I   CHKMS    /!\
03530  2314           ISZ     CHKMS    /!\
03531  7350           AC3777           /!\
03532  0554           AND I   INT0     /!\
03533  7041           CIA              /!\
03534  3556           DCA I   INT2     /!\COPY B1-11 OF TCB OF SENDER
                                       /INTO TCB[0] OF RECEIVER IN ORDER
                                       /TO CLAIM.
03535  5714           JMP I   CHKMS    /!\

03536  3331           AVPT;(0;AVPT=.-2
03537  3573
       3536

03573  0000
03574  7761
```

```
3766    03575  4063
3767    03576  0011
3768    03577  0731
3769           3600  PAGE
```

```
                   /REQCDF FUNCTION.
                   /REQUEST A DATAFLD TO BECOME ACCESSIBLE VIA THE QUICK VRCDF
                   /ROUTINE. AC B6-11 IS THE REQUESTED FLD.
03600  1377        XREQCDF,TAD      (11
03601  1164                  TAD      CURTSK
03602  3150                  DCA      MON0     /PTR TO TCB[14].
03603  1550                  TAD I    MON0
03604  0145                  AND      C7700
03605  3550                  DCA I    MON0     /OLD REQCDF OUT.
03606  1004                  TAD      MONAC
03607  0130                  AND      C77
03610  1550                  TAD I    MON0
03611  3550                  DCA I    MON0     /NEW REQCDF IN.
03612  4051                  JMS      CDFUF    /PATCH THE VRCDF ROUTINE IN USERS FIELD
03613  5776'                 JMP      VRCDF3-1

                   /HELP ROUTINE FOR STOPPING AND RESUMING TSKS.
                   /STSK# IN MONAC.
03614  0000        HLPST,    0
03615  1004                  TAD      MONAC
03616  4775'                 JMS      GETTCB
03617  1135                  TAD      M1       /PTR TO TCB[4], WAITWORD.
03620  6002                  IOF      /\
03621  3155                  DCA      INT1     /\
03622  1555                  TAD I    INT1     /\
03623  5614                  JMP I    HLPST    /\

                   /STOP A TSK.
                   /STSK# IN MONAC.
03624  7001        STP0,     IAC               /\REMOVE TSK FROM RUNQ AND SET UP
03625  3157                  DCA      INT3     /\STOPDBIT.
03626  1155                  TAD      INT1     /\
03627  4774'                 JMS      OURUNQ   /\
                   /FALL BACK INTO XSTOP AGAIN.
03630  4214        XSTOP,    JMS      HLPST    /\RETURNS CONTENTS OF WAITWORD.
03631  0373                  AND      (7701    /\ANY WAITCONDITION?
03632  7650                  SNA CLA           /\
03633  5224                  JMP      STP0     /\NO; SO FIRST REMOVE TASK FROM RUNQ.
03634  7344                  ACM2              /\
03635  0555                  AND I    INT1     /\
03636  7001                  IAC               /\SET UP STOPDBIT.
03637  3555                  DCA I    INT1     /\
03640  2155                  ISZ      INT1     /\PTR TCB[5]
03641  1555                  TAD I    INT1     /\GET PTR TO TIMEOUT NODE IF ANY.
03642  3214                  DCA      HLPST    /\
03643  3614                  DCA I    HLPST    /\REMOVE TSK FROM TOQ (OR INNOCENT
03644  1155                  TAD      INT1     /\CLEAR)
03645  3555                  DCA I    INT1     /\SET INNOCENT PTR.
03646  5772'                 JMP      SETAC    /\

                   /RESUME TSK PREVIOUSLY STOPPPED.
                   /MONAC =STSK#.
03647  4214        XRESUME,JMS      HLPST    /\
03650  0136                  AND      C7776    /\REMOVE STOPDBIT
03651  3555                  DCA I    INT1     /\
3825   03652  1555          TAD I   INT1    /\GET WAITWORD
3826   03653  0145          AND     C7700   /\ANY OTHER WAITCONDITIONS?
3827   03654  7650          SNA CLA         /\
3828   03655  4771'         JMS     INRUNQ  /\NO SO INSERT IN RUNQ.
3829   03656  5772'         JMP     SETAC   /\
              /***** CHANGING INTERRUPT SLOTS.
              /
              /THE FOLLOWING INSTRUCTIONS CHANGE INTSLOTS (SEE INTR SECTION):
              /DISINTR (DISABLE DEVICE INTERRUPT), CNINTR (CONNECT SBR TO
              /INTSLOT), CLINTR (ATTACH MS TO INTSLOT), FRINTR (SET INTSLOT
              /INTO DISCARD MODE). WE ADDED HERE ONE RELATED FUNCTION, SIMINTR,
              /TO START UP INTERRUPT DRIVEN SECTIONS.
              /
              /THE INTSLOT TO BE CHANGED IS ALWAYS DETERMINED BY ITS SKPIOT,
              /THE SKIPFLAG INSTRUCTION IT STARTS WITH. THIS SKPIOT IS PASSED
              /IN AC.

              /SEARCH SKPCHAIN FOR APPROPRIATE INTSLOT. MONAC=SKPIOT.
              /AT RETURN X0 PTS AT SKPIOT+2 AND AC HOLDS CONTENTS OF SKIPIOT+1.
03657  0000   MCINTR, 0
03660  1370           TAD     (SKPCHN-1
03661  3010   MCILP,  DCA     X0
03662  1410           TAD I   X0       /GET SKPIOT.
03663  7450           SNA              /0 TERMINATES THE SKPCHN.
03664  4576           SYSHLT           /WE HIT ON THE 0-INSTR AT THE END OF THE
                                       /SKPCHN, SO A NONEXISTING INTSLOT WAS
                                       /SPECIFIED.
03665  7041           CIA
03666  1004           TAD     MONAC
03667  7650           SNA CLA
03670  5274           JMP     MCILF    /WE FOUND THE INTSLOT.
03671  7307           AC4
03672  1010           TAD     X0
03673  5261           JMP     MCILP    /NEXT INTSLOT.
03674  6002   MCILF,  IOF              /\CHANGING INTSLOTS MUST BE DONE DEAF OF
03675  1410           TAD I   X0       /\COURSE
03676  5657           JMP I   MCINTR   /\


              /DISABLE DEVICE INTERRUPT.
              /CHANGE INTSLOT INTO
              /       SKPIOT
              /       JMP     .+4
              /       JMP     .+3
              /       ....
              /NOTE!!!!!
              /       BE SURE THAT THE FLAG REMAINS CLEARED WHILE EXECUTING THE
              /       DISINTR CALL!
03677  4257   XDISINTR,JMS    MCINTR   /\
03700  5312           JMP     MCIOUT   /\


              /CONNECT A SBR TO INTSLOT.
              /SBR RESIDES IN TASKS DF. BE SURE THAT IT IS LOCKED!
              /INTSLOT IS CHANGED INTO
              /       SKPIOT
              /       JMP     .+4
              /       CIF CDF          /TO FLD OF SBR (CALLER'S DF).
              /       JMS I   .+1
              /       ENTRY            /PTR TO SBR ENTRYPOINT.
```

```
                        /ARG1=PTR TO SBR ENTRYPOINT.
                        /ARG2=PTR TO MS[2] OF COMMUNICATION MS.
03701  4325     XCNINTR,JMS     STTOMS  /STORE TCBPTR INTO COMM.MS, IF ANY.
03702  4257             JMS     MCINTR  /\
03703  1367             TAD     (-1+4600-5200/\CHANGE "JMP .+4" INTO "JMS I .+3"
03704  3257             DCA     MCINTR  /\TEMP
03705  4350             JMS     CDIFDF  /\RETURNS CIF CDF TO DATAFIELD (LOCKED!).
03706  3410             DCA I   X0      /\
03707  1257             TAD     MCINTR  /\
03710  3410             DCA I   X0      /\
03711  1161             TAD     MARG1   /\
03712  3410     MCIOUT, DCA I   X0      /\
03713  5772'            JMP     SETAC   /\


                        /CLAIM AN INTSLOT.
                        /ATTACH A MS TO INTSLOT.
                        /THE MS WILL BE REPORTED TO THE CALLER EACH TIME THE SPECIFIED
                        /DEVICE INTERRUPTS.
                        /INTSLOT WILL BE CHANGED INTO:
                        /       SKPIOT
                        /       JMP     .+4
                        /       CLRIOT          /CLEAR DEVICEFLAG
                        /       IEXIT           /REPORT MS
                        /               MS      /PTR TO MS[2].
                        /ARG1=CLEAR IOT (TO CLEAR DEVICE FLAG)
                        /ARG2=PTR TO MS[2].
03714  4325     XCLINTR,JMS     STTOMS  /STORE TCBPTR INTO SENDERWORD OF MS.
03715  4257     XCLI2,  JMS     MCINTR  /\
03716  7200             CLA             /\
03717  1161             TAD     MARG1   /\
03720  3410             DCA I   X0      /\
03721  1366             TAD     (IEXIT  /\
03722  3410             DCA I   X0      /\
03723  1162             TAD     MARG2   /\
03724  5312             JMP     MCIOUT  /\

03725  0000     STTOMS, 0               /STORE TCBPTR IN MS[0]
03726  4765'            JMS     INARG2
03727  1162             TAD     MARG2   /
03730  7450             SNA
03731  5725             JMP I   STTOMS  /NO MESSAGE INDICATED
03732  1136             TAD     M2
03733  3150             DCA     MONO    /PTR TO MS[0]
03734  1164             TAD     CURTSK  /
03735  3550             DCA I   MONO    /DENOTE IN SENDERWORD.
03736  5725             JMP I   STTOMS


                        /FREE AN INTSLOT.
                        /SET INTO DISCARD MODE.
                        /INTSLOT IS CHANGED INTO
                        /       SKPIOT
                        /       JMP     .+4
                        /       CLRIOT          /CLEAR DEVICE FLAG
```

```
                      /       IEXIT
                      /       0
                      /ARG1=CLRIOT.
      03737  3162     XFRINTR,DCA     MARG2
      03740  5315             JMP     XCLI2


                      /START UP A DEAF INTERRUPTDRIVEN SECTION.
                      /THE CALLER IS CONTINUED AS SOON AS THE NORMAL EXIT FROM
                      /CONNECTED ROUTINE (IEXIT) IS TAKEN.
                      /THE DEAF SECTION TO BE STARTED MUST RESIDE IN CALLERS DF.
                      /ARG1=PTR TO ENTRYPOINT OF DEAF SECTION.
                      /NOTE! SCED IS INHIBITED, SO THE DEAF SECTION CAN ONLY END BY
                      /TAKING FSTEXIT!
      03741  6002     XSIMINTR,IOF            /-
      03742  1364             TAD     (EXTPRVI /\PATCH EXIT FROM INTR SECTION, SO THAT
IT
      03743  3763'            DCA     EXTALL  /\JUMPS TO MONEX.
      03744  4350             JMS     CDIFDF  /-RETURNS CIF CDF TO DATAFIELD IN AC.
      03745  3346             DCA     .+1     /-
      03746  6203             CIF CDF         /-TO CALLER'S DATAFLD.
      03747  5561             JMP I   MARG1   /!

                      /COMPUTE CIF CDF TO DATAFIELD. (MUST BE LOCKED!)
      03750  0000     CDIFDF, 0
                      IFDEF PDP8E <
      03751  1007             TAD     MONFL
      03752  0123             AND     C7
      03753  7106             CLL RTL;RAL
      03754  7004
      03755  1016             TAD     XCDIF
                      >
                      IFNDEF PDP8E <
                              AC2
                              TAD     CDFINS
                      >
      03756  5750             JMP I   CDIFDF

      03757  3536             AVPT;(0;AVPT=.-2
      03760  3762
             3757

      03762  0000
      03763  0422
      03764  5776
      03765  2740
      03766  4515
      03767  7377
      03770  0206
      03771  1076
      03772  3436
      03773  7701
      03774  1136
      03775  4063
      03776  0725
      03777  0011
```

```
3995          4000  PAGE
```

```
                     /DISMISS CURRENT TSK FROM CORE.
                     /THE PAGES CONTAINING TSKS CODE ARE RETURNED BUT THE TCB IS KEPT.
                     /BASE:=CURTSK, MARG2:=PTR TO STL[TSK,1].
                     /SET UP ONDISK CONDITION IN TCB.
                     /USES MARG1.
04000  0000          RTNTSK, 0
04001  1164                  TAD     CURTSK
04002  3107                  DCA     BASE
04003  4060                  GET;1
04004  0001
04005  0132                  AND     C177
04006  7124                  STL RAL
04007  1116                  TAD     ZSTL
04010  3162                  DCA     MARG2   /PTR TO STL[TSK,1]
04011  1562                  TAD I   MARG2   /LENGTH +ZREQ+CRES
04012  0377                  AND     (537
04013  3161                  DCA     MARG1   /ARGUMENT FOR PGRQN.
04014  7330                  AC4000          /TURN ON ONDISK BIT.
04015  1510                  TAD I   X
04016  3510                  DCA I   X
04017  4060                  GET;11  /CONTENTS OF TCB[14]; FIRST PAGE
04020  0011
04021  0146                  AND     C7600
04022  3110                  DCA     X       /TEMP
04023  1110                  TAD     X       /SUBSTRACT PAGE# FROM PC.
04024  7041                  CIA             /FOR WE MUST SAVE IT RELATIVE.
04025  1054                  TAD     MONPC
04026  3054                  DCA     MONPC
04027  1163                  TAD     VCURIF  /THE FLD IN WHICH CURTSK RESIDES
04030  7002                  BSWR
04031  1110                  TAD     X
04032  4776'                 JMS     PGRQN
04033  5600                  JMP I   RTNTSK


                     /DETACH TCB FROM ENTRY IN STL.
                     /STSK# IN MONAC.
04034  1004          XRTNTCB,        TAD     MONAC
04035  0132                  AND     C177    /COMPUTE PTR TO STL[TSK,0].
04036  7104                  CLL RAL
04037  1116                  TAD     ZSTL
04040  4775'                 JMS     RNTCB
04041  5774'                 JMP     SETAC



                     /REQUEST A NUMBER OF PAGES.
                     /ARG1   B5=1: PAGES IN CORERESIDENT FLD, B7-11 LENGTH.
                     /AT RETURN: FIRST PAGE IN AC B0-4, FLD IN AC B6-11.

                     /RETURN PAGES PREVIOUSLY REQUESTED BY REQPAG.
                     /ARG1   B5=1: CRES, B7-11 LENGTH.
                     /AC     B0-4: FIRST PAGE, B6-11: FLD.
04042  3004          XREQPAG,DCA     MONAC   /JUST FOR SURE.
04043  1161          XRTNPAG,TAD     MARG1
```

```
04044  0132          AND     C177    /DONT BOTHER ABOUT BIT40.
04045  3161          DCA     MARG1
04046  1004          TAD     MONAC
04047  4776'         JMS     PGRQN
04050  5774'         JMP     SETAC

             /REQUEST A FLD FOR DATA STORAGE.
             /AT RETURN: AC B6-11 =FLD.
04051  1173  XREQFLD,TAD Z1000;FDATA=1000 /SET UP DATA STORAGE BIT IN FLDTAB.
       1000
04052  4773'         JMS     GETFLD
04053  3004          DCA     MONAC
04054  1141          TAD     C7773;FMININ=4
       0004
04055  0553          AND I   MON3
04056  3553          DCA I   MON3    /CLEAR MONINBIT.
04057  5772'         JMP     MONEX

             /RETURN FLD PREVIOUSLY REQUESTED BY REQFLD.
             /AC B6-11 =FLD.
04060  1004  XRTNFLD,TAD     MONAC
04061  4771'         JMS     FRFLD
04062  5774'         JMP     SETAC
```

```
                  /GET PTR TO TCB[5] TSK LINKWORD.
                  /AC=STSK#.
                  /USES BASE,X,X0.
04063     0000    GETTCB, 0
04064     0132              AND     C177
04065     7104              CLL RAL
04066     1116              TAD     ZSTL
04067     3200              DCA     RTNTSK  /JUST A TEMP.
04070     1600              TAD I   RTNTSK  /FETCH STL[TSK,0]
04071     7550              SPA SNA         /IS A TCB ATTACHED TO THIS ENTRY?
04072     5663              JMP I   GETTCB  /YES, THEN THIS IS THE CORRECT PTR.
                  IFDEF CHECK <
04073     7650              SNA CLA
04074     4576              SYSHLT          /0 INDICATES FREE ENTRY IN STL, ERROR.
                  >
04075     4770'             JMS     GN20    /REQUEST SPACE FOR SETTING UP TCB
04076     1010              TAD     X0
04077     3107              DCA     BASE    /PTR TO TCB[0]-1
04100     1144              TAD     M20
04101     3110              DCA     X       /CTR.
04102     3410              DCA I   X0      /CLEAR TCB
04103     2110              ISZ     X
04104     5302              JMP     .-2
04105     1600              TAD I   RTNTSK  /GET CONTENTS OF STL[TSK,0]
04106     4075              PUT;    17+1    /STORE IN TCB[15].
04107     0020
04110     7327              AC6
04111     1107              TAD     BASE
04112     3600              DCA I   RTNTSK  /STORE PTR TO TCB[5] LINKWORD IN
04113     2200              ISZ     RTNTSK  /STL[TSK,0]. PTR TO STL[TSK,1].
04114     1600              TAD I   RTNTSK  /GET PRIO
04115     7002              BSWR
04116     0127              AND     C70
04117     7132              STL RTR         /2*PRIO+FWTMS
04120     4075              PUT;4+1         /IN TCB[4].
04121     0005
04122     1200              TAD     RTNTSK  /PTR TO STL[TSK,1].
04123     0134              AND     C377    /COMPUTE STSK#
04124     7130              STL RAR         /STSK# +ONDISKBIT.
04125     4075              PUT;    6+1     /IN TCB[6].
04126     0007
04127     7040              CMA
04130     1110              TAD     X       /PTR TO TCB[5] LINKWORD
04131     5663              JMP I   GETTCB

                  /REQUEST ENTRY IN STL.
                  /ARG1,ARG2 CONTENTS OF THE ENTRY.
04132     4767'   XREQSTL,JMS     INARG2
04133     1116              TAD     ZSTL
04134     3110              DCA     X       /PTR IN STL.
04135     1366              TAD     (-MAXSTL
04136     3200              DCA     RTNTSK  /CTR
04137     1510    RQSTLP, TAD I   X
04140     7650              SNA CLA
04141     5347              JMP     RQSTLF  /FOUND.
```

```
4130   04142  2110            ISZ     X
4131   04143  2110            ISZ     X
4132   04144  2200            ISZ     RTNTSK
4133   04145  5337            JMP     RQSTLP  /NEXT.
4134   04146  4576            SYSHLT          /ALL ENTRIES IN STL USED.
4135   04147  1161  RQSTLF,   TAD     MARG1
4136   04150  3510            DCA I   X
4137   04151  2110            ISZ     X
4138   04152  1162            TAD     MARG2
4139   04153  3510            DCA I   X
4140   04154  1110            TAD     X
4141   04155  7010            RAR
4142   04156  0132            AND     C177
4143   04157  5774'           JMP     SETAC
4144
4145                  /ATTACH TCB TO ENTRY IN STL.
4146   04160  1004  XREQTCB,TAD       MONAC   /GET STSK#.
4147   04161  4263            JMS     GETTCB
4148   04162  5774'           JMP     SETAC   /RETURN PTR TO TCB[5]
4149
4150   04163  3757            AVPT;(0;AVPT=.-2
4151   04164  4165
4152          4163
4153
4154   04165  0000
4155   04166  7701
4156   04167  2740
4157   04170  4354
4158   04171  1507
4159   04172  0731
4160   04173  1447
4161   04174  3436
4162   04175  3450
4163   04176  1221
4164   04177  0537
4165          4200  PAGE
```

```
                       /***** LOCKING AND UNLOCKING FIELDS *****
                       /FUNCTIONS: LOCK AND UNLOCK
                       /THE FIELD TO BE LOCKED OR UNLOCKED IS ALWAYS PASSED IN DF
                       /(TASKS DATAFIELD).

        04200  1146    XUNLOCK,TAD      M200
        04201  1131    XLOCK,  TAD      C100
                       /CHANGE IN LOCK COUNT IN AC.
                       /AC=100: LOCK, AC=-100: UNLOCK TASKS DF.
        04202  3150            DCA      MON0     /TEMP.
        04203  1007            TAD      MONFL    /GET DF
        04204  0123            AND      C7
        04205  1170            TAD      ZCORMAP
        04206  3151            DCA      MON1     /PTR IN CORMAP.
        04207  1551            TAD I    MON1
        04210  1150            TAD      MON0
                       IFDEF CHECK <
        04211  7510            SPA
        04212  4576            SYSHLT            /WE DONT ACCEPT MORE THAN 37 LOCKS
                       >
        04213  3551            DCA I    MON1
        04214  5777'           JMP      MONEX    /NOTE: THIS LOCK IS TEMPORARY, SO WE NEED
                                                 /NOT SET THE CRESBIT IN FLDTAB.


```

```
             /APP A5,   MC8.5. TABLES, STORAGE ALLOCATOR,
             /***** TABLES AND STORAGE ALLOCATION *****

             /GET NODE.
             /      JMS     GN
             /AVLN,         0       /HEAD OF AVAIL LIST.
             /              NN      /LENGTH OF NODE, NEEDED IF NEW NODES ARE
             /                      /CREATED.
             /IFDEF STATX <AVLNCT, 0 > /REQUESTED NODES COUNTER
             /NORMAL,...            /X0= (W0 OF NODE)-1.

04215  0000  GN,    0
04216  1615         TAD I    GN      /GET FIRST NODE
04217  7450         SNA
04220  5233         JMP      GNA     /THIS CHAIN EXHAUSTED; CREATE NEW NODE.
04221  3362         DCA      GNFTEM
04222  1762         TAD I    GNFTEM
04223  3615         DCA I    GN      /UPDATE AVAIL LIST.
04224  2215         ISZ      GN
04225  2215         IFDEF STATX < ISZ GN >
04226  2215  GNEX,  ISZ      GN
04227  7040         CMA
04230  1362         TAD      GNFTEM
04231  3010         DCA      X0      /SET AUTOINDEX TO NODE.
04232  5615         JMP I    GN
             /BREAK A NEW NODE FROM THE LARGE AVILLIST IN CORE.
04233  2215  GNA,   ISZ      GN      /GN PTS TO LENGTH NEEDED.
04234  1376         TAD      (AVHEAD /SET PTR TO PRECESSOR.
04235  3317  GNALP, DCA      GNPR    /SEARCH FOR NODE OF SUFFICIENT LENGTH.
04236  1717         TAD I    GNPR    /GET NODE
04237  7450         SNA
04240  4576         SYSHLT           /ALL AVAIL SPACE EXHAUSTED.
04241  3362         DCA      GNFTEM  /PTR TO W0 OF NODE
04242  1362         TAD      GNFTEM
04243  3010         DCA      X0      /PTR TO PTR TO LAST LOC (AUTO IND).
04244  1410         TAD I    X0
04245  3010         DCA      X0      /PTR TO LAST LOC.
04246  1010         TAD      X0      /(LAST LOC)
04247  7160         CMA STL          /L=1, AC=-(LAST LOC) -1
04250  1362         TAD      GNFTEM  /L=1; -LENGTH IN AC.
04251  1615         TAD I    GN      /ADD REQUESTED LENGTH
                                     /OK IF AC=0, OR NO OVERFLOW (L=1).
                                     /-LENGTH OF REMAINDER IN AC.
04252  7460         SZA SNL
04253  5300         JMP      GNNEXT  /NOT LONG ENOUGH.
04254  7001         IAC              /L=0 IF REMAINDER <2.
             IFDEF STATX <  /COUNT LOST SPACE
04255  7450         SNA              /IF AC=0, LENGTH OF REMAINDER=1,
04256  2304         ISZ      AVDIRT  /SO WE ARE GOING TO LOOSE 1 LOC,
             >
04257  7620         SNL CLA          /IF (LENGTH REMAINDER)<2 (L=0),
04260  5276         JMP      GNDEL   /THEN DELETE REMAINDER FROM AVAILLIST,
             GNBR,                   /ELSE BREAK THE NODE.
04261  1362         TAD      GNFTEM  /PTR TO FIRST LOC OF NODE
04262  1615         TAD I    GN      /ADD LENGTH
```

```
04263  3717            DCA I    GNPR      /UPDATE PRECESSOR.
04264  1717            TAD I    GNPR
04265  3317            DCA      GNPR      /PTR TO W0 OF REMAINDER NODE.
04266  1762            TAD I    GNFTEM    /PTR TO NEXT NODE
04267  3717            DCA I    GNPR
04270  2317            ISZ      GNPR
04271  1010            TAD      X0        /STORE THERE NODE'S LAST LOC.
04272  3717    GNEX2,  DCA I    GNPR
               IFDEF STATX <
04273  2215            ISZ      GN
04274  2615            ISZ I    GN        /COUNT THE CREATED NODES
               >
04275  5226            JMP      GNEX
               GNDEL,                     /DELETE REMAINDER NODE FROM AVAILLIST.
04276  1762            TAD I    GNFTEM    /GET SUCCESSOR
04277  5272            JMP      GNEX2     /STORE IN PRECESSOR.
04300  7200    GNNEXT, CLA                /NODE TOO SHORT, TRY NEXT ONE
04301  1362            TAD      GNFTEM
04302  5235            JMP      GNALP

04303  0000    AVHEAD, 0
04304  0000    IFDEF STATX <AVDIRT, 0>
       4317    GNPR=DF3


               /       TAD      PTR       /PTR NODE W0
               /       JMS      FN
               /                AVLN      /PTR TO AVAIL LIST.

04305  0000    FN,     0
04306  3362            DCA      GNFTEM    /PTR TO NODE W0.
04307  1705            TAD I    FN
04310  3215            DCA      GN        /PTR TO AVAIL LIST.
04311  1615            TAD I    GN
04312  3762            DCA I    GNFTEM
04313  1362            TAD      GNFTEM
04314  3615            DCA I    GN        /INSERTED IN AVAIL LIST
04315  2305            ISZ      FN
04316  5705            JMP I    FN

               /DELETE AND FREE A NODE FROM THE FCCHAIN.
               /PGROLD PTS TO PRECESSOR.

               GNPR,
04317  0000    DF3,    0
04320  1020            TAD      PGROLD    /PTR TO PRECESSOR
04321  3362            DCA      GNFTEM
04322  1762            TAD I    GNFTEM
04323  3215            DCA      GN        /PTR TO NODE TO BE DELETED.
04324  1615            TAD I    GN
04325  3762            DCA I    GNFTEM    /DELETED FROM FCCHAN.
04326  1215            TAD      GN
04327  4305            JMS      FN
04330  4342                     AVL3
04331  5717            JMP I    DF3
```

```

04332  0000  GN2,    0
04333  4215          JMS     GN
04334  0000  AVL2,           0
04335  0002                  2
04336  0000  IFDEF STATX <AV2CT, 0 >
04337  5732          JMP I   GN2

04340  0000  GN3,    0
04341  4215          JMS     GN
04342  0000  AVL3,           0
04343  0003                  3
04344  0000  IFDEF STATX <AV3CT, 0 >
04345  5740          JMP I   GN3

04346  0000  GN5,    0
04347  4215          JMS     GN
04350  0000  AVL5,           0
04351  0005                  5
04352  0000  IFDEF STATX <AV5CT, 0 >
04353  5746          JMP I   GN5

04354  0000  GN20,   0
04355  4215          JMS     GN
04356  0000  AVL20,          0
04357  0020                  20
04360  0001  IFDEF STATX <AV20CT, 1 /TCB FOR MTINIT >
04361  5754          JMP I   GN20

04362  0000  GNFTEM, 0

04363  4163          AVPT;(0;AVPT=.-2
04364  4375
       4363

04375  0000
04376  4303
04377  0731
       4400  PAGE
```

```
                       /FORCE STATIC LIST AT MULTIPLE OF 400.
                       IFNZRO .^200 <  /ONE EXTRA PAGE
                               AVPT;(0;AVPT=.-2
                               PAGE
                       >

                       /STATIC TASK LIST.
                       STLST,
                 0000  SIDLE=.%2^177
         04400   5113          IDLE;-1 /IDLELOOP TASK.
         04401   7777
                 0001  SMTSWAP=.%2^177
         04402   5035          MTSWAP;-1       /MONITOR SWAP TASK.
         04403   7777
                 0002  SMTFTCH=.%2^177
         04404   5051          MTFTCH;-1       /MONITOR FETCH TASK, PRIO=1!
         04405   7777
                 0003  SMTTIME=.%2^177
         04406   5067          MTTIME;-1       /MONITOR TIMER TASK
         04407   7777
                 0004  SMTSDSK=.%2^177
         04410   5103          MTSDSK;-1       /MONITOR SYSTEM DISK TASK
         04411   7777
                 0005  SMTINIT=.%2^177
         04412   5127          MTINIT;1537     /INITIALIZATION TASK.
         04413   1537
         04414   0000  ZBLOCK MAXSTL↑2+2+STLST-.        /CLEAR REMAINING ENTRIES.

                       IFNZRO .-1^177-175^4000 <AVPT;(0;AVPT=.-2 > /SPACE LEFT ON THIS
                                                                  /PAGE.

                       IFNZRO .^177 <PAGE>

```

```

                        /FIELDTABLE
                        IFNZRO .^177 <FLDTAB MUST BE ON PAGE BEGIN!>
                        FLDTAB,
      04600  6707               6200+FMONIN+FOCCUP+FDP+FCRES+FZREQ
      04601  4220               4220
      04602  4230               4230
      04603  4240               4240
      04604  4250               4250
      04605  4260               4260
      04606  4270               4270
             4607       *FLDTAB+ACTMAX
      04607  0000               ZBLOCK VIRMAX-ACTMAX
             4677       *FLDTAB+VIRMAX
      04677  6713               6210+FZREQ+FCRES+FOCCUP+FDP /USED BY INIT TASK.

                        /PRIORITY RUNQS.
                        RQPTRS,
      04700  0000               0;.-1   /PRIO 0
      04701  4700
      04702  5127               MTINIT; MTINIT  /PRIO 1.  INIT TASK IN.
      04703  5127
      04704  0000               0;.-1   /PRIO 2
      04705  4704
      04706  0000               0;.-1   /PRIO 3
      04707  4706
      04710  0000               0;.-1   /PRIO 4
      04711  4710
      04712  0000               0;.-1   /PRIO 5
      04713  4712
      04714  0000               0;.-1   /PRIO 6
      04715  4714
      04716  0000               0;.-1   /PRIO 7
      04717  4716
      04720  5113               IDLE    /PRIO 10. NO TAIL WORD NEEDED.

      04721  0000       FCCHAN, 0                 /HEAD OF FREE CORE CHAIN.

                        /COREMAP.
                        CORMAP,
      04722  0100               100     /FLD 0 LOCKED IN CORE.
      04723  0177               VIRMAX+100        /THE HIGHEST VIRTUAL FLD INITIALLY LOCKED
                                                  / IN FLD 1.
      04724  0001               1;2;3;4;5;6
      04725  0002
      04726  0003
      04727  0004
      04730  0005
      04731  0006

             4732       *CORMAP+ACTMAX+1

      04732  0000       IFDEF STATX <MONSTAT, ZBLOCK 100 /DOUBLE PRECISION COUNTERS OF
                                                             /MONCALLS>

```

```
4428                      IFZERO .^4000 <*4000 /FORCE TCB'S ABOVE 4000>
4429                      /TCB OF MONITOR SWAP TASK.
4430                      /         0;0               /CLAIMWORD, HEAD OF RCQ; UNUSED.
4431    05032  5743                 -SWAPMS+2         /WAITED RP.
4432    05033  0000                 0                 /HEAD OF RPQ.
4433    05034  4000                 0↑2+FWTRP         /WAITS +PRIO
4434    05035  5035       MTSWAP,   .                 /INNOCENT PTR IN LINKWORD
4435    05036  0001                 SMTSWAP           /STSK#.
4436    05037  0000                 0                 /FLDS
4437    05040  0000                 0                 /AC
4438    05041  2042                 MTSWIN            /PC
4439    05042  0000                 0                 /MQ; UNUSED.
4440    05043  0000                 0                 /EAE+L
4441                      /         0;0;0;0           /BASE, X, FIRST PAGE,DISKADDR;UNUSED.
4442
4443                      /TCB OF MONITOR FETCH TASK.
4444    05044  0000                 0;0               /CLAIMWORD, HEAD OF RCQ.
4445    05045  0000
4446    05046  0000                 0;0               /WAITED RP, HEAD OF RPQ.
4447    05047  0000
4448    05050  2002                 1↑2+FWTMS         /WAITS +PRIO
4449    05051  5051       MTFTCH,   .                 /LINK WORD.
4450    05052  0002                 SMTFTCH           /STSK#.
4451    05053  0000                 0                 /FLDS
4452    05054  0000                 0                 /AC
4453    05055  2402                 MTFTIN+2                    /PC
4454    05056  0000                 0                 /MQ
4455    05057  0400                 400               /EAE+L. RUNS WITH SCDINH.
4456    05060  0000                 0;0;0;0           /BASE,X,FIRST PAGE,DISKADDR.
4457    05061  0000
4458    05062  0000
4459    05063  0000
4460
4461                      /TCB OF MONITOR TIMER TASK.
4462                      /         0;0               /CLAIMWORD, HEAD OF RCQ; UNUSED.
4463    05064  5114                 -TIMEMS+2         /WAITED RP
4464    05065  0000                 0                 /HEAD OF RPQ.
4465    05066  4000                 0↑2+FWTRP         /WAITS +PRIO
4466    05067  5067       MTTIME,   .                 /LINK WORD.
4467    05070  0003                 SMTTIME           /STSK#.
4468    05071  0000                 0                 /FLDS
4469    05072  0000                 0                 /AC
4470    05073  2600                 TIMEIN            /PC
4471    05074  0000                 0                 /MQ
4472    05075  0000                 0                 /EAE+L
4473                      /         0;0;0;0           /BASE,X,FIRST PAGE,DISK ADDR; UNUSED.
4474
4475                      /TCB OF MONITOR SYSTEM DISK TASK.
4476    05076  0000                 0;0               /CLAIMWORD, HEAD OF RCQ.
4477    05077  0000
4478    05100  0000                 0;0               /WAITED RP, HEAD OF RPQ.
4479    05101  0000
4480    05102  2000                 0↑2+FWTMS         /WAITS +PRIO
4481    05103  5103       MTSDSK,   .                 /LINK WORD.
4482    05104  0004                 SMTSDSK           /STSK#
```

```
05105  0000             0                  /FLDS
05106  0000             0                  /AC
05107  2103             SDIN+2             /PC
05110  0000             0                  /MQ
05111  0000             0                  /EAE+L
              /         0;0;0;0            /BASE,X,FIRST PAGE, DISK ADDR; UNUSED.

              /TCB OF IDLE LOOP TASK.
              /         0;0                /CLAIMWORD, HEAD OF RCQ; UNUSED.
              /         0;0                /WAITED RP, HEAD OF RPQ; UNUSED.
05112  0020             10↑2               /WAITS +PRIO
05113  0000   IDLE,     0                  /LINK WORD
05114  0000             SIDLE              /STSK#.
05115  0000             0                  /FLDS
05116  0000             0                  /AC
05117  2352             IDLEIN             /PC
05120  0000             0                  /MQ
05121  0000             0                  /EAE+L
              /         0;0;0;0            /BASE,X,FIRST PAGE,DISK ADDR; UNUSED.

              /TCB OF INIT TSK.
05122  0000             0;0       /CLAIMWORD, HEAD OF RCQ.
05123  0000
05124  0000             0;0       /WAITED RP, HEAD OF RPQ.
05125  0000
05126  0002             2         /PRIO 1.
05127  0000   MTINIT,   0         /LINKWORD, LAST IN ITS Q.
05130  0005             SMTINIT            /STSK#.
05131  7777             VIRMAX↑100+VIRMAX/FLDS
05132  0000             0         /AC
05133  2000             INITIN    /PC
       2000   INITIN=2000
05134  0000             0         /MQ
05135  0000             0         /EAE+L
05136  0000             0;0       /BASE AND X.
05137  0000
05140  0200             200       /FIRSTPAGE +REQCDF
05141  0000             0         /DUMMY DISK ADDRESS. WILL CLEAR THE ENTRY IN STL
                                  /ON EXIT.

05142  4363             AVPT;7377;AVPT=.-2/LOTS OF AVAIL SPACE.
05143  7377
       5142


              /NOTE: WE RESERVED TWO PAGES FOR SIMPLE ERRORHANDLING
              /AND RESTART OF OS/8. IF THESE PAGES ARE ALSO TO BE USED
              /IN THE AVAILLIST SYSTEM, THEN BE AWARE OF (WORDCOUNT-
              /CORE ADDRESS) LOCATIONS OF SOME DATABREAK DEVICES.
              /PREFERABLY DONT USE LOC 7750-7755!
```

```
4533            $
```

```
ACTMAX 0007         C7777  0135         GN2    4332         KRB3   6466
AVDIRT 4304         DCDIR  0200         GN20   4354         KRS1   6034
AVHEAD 4303         DFPARM 0100         GN3    4340         KRS2   6424
AVL2   4334         DF3    4317         GN5    4346         KRS3   6464
AVL20  4356         DISINT 0016         GPZCDF 1614         KSF1   6031
AVL3   4342         DK8EP  0004         GPZCT  1634         KSF2   6421
AVL5   4350         EAE    0001         GPZRIN 1600         KSF3   6461
AVPT   5142         ERHLT  4023         GPZRI2 1632         LOCK   0046
AV2CT  4336         EXIT   0054         GPZTAD 1625         MARG1  0161
AV20CT 4360         EXTALL 0422         HARDMQ 0001         MARG2  0162
AV3CT  4344         EXTPRV 5776         HCT    0370         MAXDEV 0013
AV5CT  4352         FCCHAN 4721         HLPST  3614         MAXSTL 0077
BASE   0107         FCHECK 2000         HLTINS 0003         MCILF  3674
BSWR   7002         FCHEX  0744         HP     0430         MCILP  3661
CALL   4054         FCRES  0100         HPRIO  0165         MCINTR 3657
CCDF   0034         FCSWAP 2010         IDLE   5113         MCIOUT 3712
CDFCUR 4051         FDATA  1000         IDLEIN 2352         MONAC  0004
CDFUF  0051         FDCMAX 5000         IDLP   2350         MONEX  0731
CDIFDF 3750         FDP    0002         IDLP2  2363         MONEX0 3261
CDIFI  0721         FILLP  3501         IDNS   2342         MONFL  0007
CHECK  0001         FILTCB 0065         IDNSHL 2341         MONFUN 0160
CHKMS  3514         FLDTAB 4600         IDPTR  2347         MONIN  0004
CHKMS2 3522         FLDTEM 1524         IDSW   2345         MONIT0 3000
CHKONL 4000         FMININ 0004         IDSWHL 2344         MONIT2 3055
CHRCQ  4014         FMONIN 0004         IEXIT  4515         MONJMP 3103
CHRPQ  4005         FN     4305         INARG2 2740         MONOUT 0474
CLINTR 0023         FOCCUP 0001         INIT   0200         MONPC  0054
CMNSAF 0500         FRFLD  1507         INITIN 2000         MONSAF 0500
CNINTR 0021         FRINTR 0025         INRQEX 1127         MONSCD 0543
CORMAP 4722         FSTEXT 0417         INRUNQ 1076         MONSF2 0510
CUR    0000         FSTXT2 0445         INS200 6004         MONSTA 4732
CURTSK 0164         FTCHMS 2534         INTAC  0005         MON0   0150
CURWT  0545         FWTMS  2000         INTCT  0301         MON1   0151
C10    0124         FWTRP  4000         INTDEF 0000         MON2   0152
C100   0131         FWTSWP 0400         INTFL  0006         MON3   0153
C177   0132         FWTTM  0200         INTPC  0000         MON76  3072
C2     0120         FZREQ  0400         INTR   0177         MSAPP  1051
C200   0133         GET    4060         INTR2  0202         MSAPP1 1061
C3     0121         GETFLD 1447         INTSCD 0400         MSAPP2 1071
C37    0125         GETFLF 1466         INT0   0154         MSFREE 4014
C377   0134         GETFLP 1456         INT1   0155         MSIN   1023
C4     0122         GETFLX 1504         INT2   0156         MSNOD1 2717
C40    0126         GETFL2 1503         INT3   0157         MSNOD2 2735
C7     0123         GETMQ  7701         ITSAF  0464         MSNREQ 2730
C70    0127         GETTCB 4063         JUMS   4070         MSQMAX 1075
C7600  0146         GN     4215         KB1    0006         MSREQ  4014
C77    0130         GNA    4233         KB2    0010         MSTOAC 3424
C7700  0145         GNALP  4235         KB3    0012         MTFCT  2531
C7760  0144         GNBR   4261         KCC1   6032         MTFFLD 2455
C7770  0143         GNDEL  4276         KCC2   6422         MTFPAG 2535
C7772  0142         GNEX   4226         KCC3   6462         MTFTCH 5051
C7773  0141         GNEX2  4272         KEEP   1000         MTFTIN 2400
C7774  0140         GNFTEM 4362         KM8E   0001         MTINIT 5127
C7775  0137         GNNEXT 4300         KRB1   6036         MTSDSK 5103
C7776  0136         GNPR   4317         KRB2   6426         MTSWAP 5035
```

```
$

MTSWEX 2063          REQCDF 0052          SDSWP  2130          TSF1   6041
MTSWIN 2042          REQFLD 0034          SDSW1  2122          TSF2   6431
MTTIME 5067          REQPAG 0031          SETAC  3436          TSF3   6471
MYCDIF 0052          REQSTL 0041          SIDLE  0000          TSKSAF 0554
MYCIF  0021          REQTCB 0042          SIMINT 0027          TSKSWT 0452
M1     0135          RESUME 0060          SKPCHN 0207          TSTTM  3400
M10    0143          RKBLK  2327          SKPEND 0301          TSWAIT 3316
M100   0145          RKCHEK 2325          SKPFIT 0375          TT1    0005
M2     0136          RKCMD  2322          SMSQ   3400          TT2    0007
M20    0144          RKDONE 2315          SMSQEX 3441          TT3    0011
M200   0146          RKERCT 2321          SMSQLP 3412          UNLOCK 0050
M3     0137          RKERR  2312          SMTFTC 0002          VCALL  0054
M4     0140          RKER2  2217          SMTINI 0005          VCDF   4025
M5     0141          RKER3  2235          SMTSDS 0004          VCDFTM 0042
M6     0142          RKINTR 2333          SMTSWA 0001          VCDF1  3032
M7771  0123          RKLP   2260          SMTTIM 0003          VCDIF  0051
M7774  0122          RKMS   2325          SNDMS  0003          VCURIF 0163
M7775  0121          RKMSP  2213          SNDREP 1000          VERHLT 0023
M7776  0120          RKPAG  2326          SNDWTR 0007          VFBLOK 4000
NONFIT 0000          RKRTRY 2240          STALL  0200          VFLCT  1600
NONREP 2000          RK600  2245          STATX  0001          VFLESS 1657
NRT    0400          RK8E   0000          STLST  4400          VFLF   1700
ONDISK 0600          RLOCLP 2471          STOP   0056          VFLOCK 1427
OURQF  1160          RLOC1  2501          STOPD  0001          VFLTEM 1445
OURQLP 1150          RLOC2  2507          STORMQ 7421          VFLTM2 1446
OURUNQ 1136          RNTCB  3450          STP0   3624          VFLTRY 1662
PDP8E  0001          RP     0013          STTOMS 3725          VFNEXT 1656
PDP8IE 0002          RPINTR 0402          SWAPMS 2037          VFSWLP 2042
PGR    1200          RQPTRS 4700          SWNODE 2150          VGET   0060
PGREX  1261          RQSTLF 4147          SWPOUT 0400          VIRMAX 0077
PGREXH 1215          RQSTLP 4137          SYDISK 0000          VJUMS  0070
PGRFF  1351          RTN    2705          SYSHLT 4576          VMSNOD 0014
PGRLEN 0025          RTNFLD 0036          SYTIME 0004          VPUT   0075
PGRLP  1210          RTNPAG 0033          SYWAIT 2330          VRCDF  4020
PGRLPM 1213          RTNTCB 0044          TCF1   6042          VRCDF1 3042
PGRMB  1327          RTNTC2 3464          TCF2   6432          VRCDF3 0726
PGRMF  1315          RTNTSK 4000          TCF3   6472          VRDF   4045
PGRM2  1321          SCDF   0633          TCKLEN 6331          VRDF1  2671
PGRNM  1332          SCDF2  0716          TIME   2667          VVCDF  0025
PGROLD 0020          SCDF3  0730          TIMEIN 2600          VVRCDF 0020
PGRQ   1224          SCDINH 0167          TIMEMS 2666          VVRDF  0045
PGRQLP 1227          SCDLP2 0625          TIMEOU 0200          WTMS   0014
PGRQN  1221          SCDREQ 0166          TLS1   6046          WTRP   0005
PGRQX  1400          SCED   0617          TLS2   6436          WTRPMT 2644
PGRREM 0045          SCEDLP 0622          TLS3   6476          X      0110
PGRTEM 0070          SCEDNI 0754          TOQEX  2642          XCDF70 0117
PGRTN  1266          SDERCT 2201          TOQFN  2630          XCDIF  0016
PGRTN2 1303          SDEX   2076          TOQLP  2607          XCHRCQ 3303
PTBASE 0654          SDFCT  2155          TOQLP2 2640          XCHRPQ 3236
PTX    0701          SDGO   2202          TPC1   6044          XCLINT 3714
PUT    4075          SDIN   2101          TPC2   6434          XCLI2  3715
P0CEX  1645          SDMSP  2100          TPC3   6474          XCNINT 3701
P0CLST 1647          SDNCT  2153          TQHEAD 2663          XDISIN 3677
P0CTAD 1634          SDNORM 2067          TQIN   2647          XERROR 3144
RDFLD  1355          SDPTR  2200          TSBLOK 6000          XEXIT  3267
```

```
XFILTC 3466
XFLDTA 0112
XFRINT 3737
XLOCK  4201
XMONIT 0111
XMSNOD 0114
XREQCD 3600
XREQFL 4051
XREQPA 4042
XREQST 4132
XREQTC 4160
XRESUM 3647
XRP    3251
XRP2   3263
XRTNFL 4060
XRTNPA 4043
XRTNTC 4034
XSIMIN 3741
XSNDMS 3200
XSNDWT 3200
XSND2  3212
XSTALL 3315
XSTOP  3630
XUNLOC 4200
XVCALL 0037
XVRDF1 0113
XWTMS  3303
XWTRP  3236
X0     0010
X1     0011
X2     0012
X3     0013
X6000  0001
ZCORMA 0170
ZFN    0172
ZSNDRE 0171
ZSTL   0116
Z1000  0173
```

```
ERRORS DETECTED: 0
LINKS GENERATED: 81
4534
```

```
ACM2      910  1441  2511  2997  3510  3548  3808
ACM3     2481  3531
ACS      1260
ACTMAX     14# 2139  2142  4384  4385  4423
ACTUAL     13
AC2      1468  2090  2426  2500  2608  2616  3972
AC2000   1398  3513
AC3      1472  2911
AC3777   1200  2273  2417  2762  3646  3742  3752
AC4      1456  2955  3522  3856
AC4000   1475  2410  2900  3671  4013
AC6      2503  4101
ALLOCA   3381
ANALEX    740   741
ASS        99   256   446   505   547
AVDIRT   4239  4267#
AVHEAD   4218  4266#
AVL2     2894  4304#
AVL20    3705  4325#
AVL3     4299  4311#
AVL5     3000  3560  4318#
AVPT      545   883   885  1172  1174  1405  1407  1608  1610  1883
         1885  2033  2035  2160  2162  2451  2454  2703  2709  2828
         2830  3022  3024  3386  3388  3616  3618  3760  3762  3977
         3979  4150  4152  4332  4334  4342  4368  4523  4525#
AV2CT    4306#
AV20CT   4327#
AV3CT    4313#
AV5CT    4320#
BASE      399   438   449# 1143  1228  1229  1241  1306  2723  2813
         4003  4092  4102
BE         99   256  4375
BEGIN    4375
BSWR      242   246# 1244  1309  1871  1876  2735  2752  4025  4106
CALL      377# 2394  2397  2483  2486  2638  2720  2768
CCDF      336# 3323
CDFCUR    362#
CDFINS   1304  1354# 3324  3973
CDFUF     360#  947  1129  1138  1161  1223  1273  1285  1327  1341
         1382  2970  3015  3081  3129  3722  3783
CDIFDF   3891  3957  3963# 3975
CDIFI    1246  1247  1278  1284  1311  1312  1332# 1340
CHECK      79#  591   870  1729  1954  4086  4182
CHKMS    1454  3663  3739# 3750  3751  3758
CHKMS2   3741  3745#
CHKONL   3289# 3290  3293
CHRCQ    3290#
CHRPQ    3293#
CIFINS   1314  1353#
CINT      863
CLINTR   3198#
CLSA      791
CLSK      783
CMNSAF   1088  1098#
CNINTR   3188#
CORMAP    540  2136  2142  4412# 4423
CUR       274#  304   330   364   398   417   435
CURTSK    530#  938  1099  1157  1195  1199  1201  1202  1208  1222
         1380  1401  1461  2683  2685  2912  2915  3123  3517  3532
```

```
          3588  3774  3929  4002
CURWT     1153# 1159  1189  2901  3575  3603
C10        474# 3124
C100       479# 1269  1323  2003  2023  2315  3493  4172
C177       480# 2319  2726  2758  2796  4006  4034  4051  4079  4142
C2         467# 2000  2971  3537
C200       481# 1120  1261  1357  2536  2800  3604
C3         469#
C37        475# 1533  1728  1918  2554  2734  2739  3338
C377       482# 4112
C4         471# 1912  2049  2055
C40        476#
C7         473# 1103  2605  2648  3498  3966  4177
C70        477# 1236  1248  1947  2611  4107
C7600      503# 2636  2748  2799  4018
C77        478# 1107  1877  2011  2019  2244  2745  2963  3780
C7700      501#  978  1122  1134  1240  1266  1305  2149  2520  2621
          3087  3777  3826
C7760      499#
C7770      497#
C7772      495#
C7773      493# 2054  4063
C7774      491#
C7775      489#
C7776      487# 3823
C7777      485#
DCDIR      105#  106
DCLR      2501  2590
DCMA      2600# 2656
DFPARM    2769  3311  3494#
DF3       1794  1848  4268  4290# 4300
DIMA      2599# 2647
DIML      2598# 2613  2646
DISINT    3181#
DK8EP       37#   94   781   782   785   790
DLAG      2545
DLCA      2528
DLDC      2506  2543
DRST      2587
DSKP       751
DTA        731   732
DTSF       733
DXAL      2633  2655
EAE          5#    7  1115  1128  1235  1303
ERHLT      313#
ERR         99   446   505   547
ERROR      256  3381
EXIT      3266#
EXTALL     929# 1289  1346  3956
EXTPRV     928# 3955
FCCHAN    1734  4409#
FCHECK    1230  1245  1310  2243# 2262  3127
FCHEX     1380# 2268  2285
FCRES     1636  1774  1780  1898  1942  2004  2027  2316# 4377  4387
FCSWAP    2264# 3105
FDATA     1632  2027  2325  4060#
FDCMAX     107#
FDP       1643  2001  2027  2311  2313  2427# 4377  4387
FILLP     3722# 3731
FILTCB    3284#
```

```
FLDTAB      455   2432  4375  4376# 4384  4386
FLDTEM     1981   1994  1997  2005  2031#
FMININ     4064#
FMONIN     1640#  4377
FN          542   4275# 4277  4283  4284  4298
FOCCUP     1645#  2027  4377  4387
FPCR        845
FPSF        843
FPTP        841    842
FPTR        850    851
FRCR        854
FRFLD      1872   1985  1991  2018# 2030  4072
FRINTR     3206#
FRSF        852
FSTEXT      906    917# 2932
FSTXT2      935    956#
FTCHMS     2737   2765  2770  2778  2797  2809  2824#
FWTMS      1450   1568# 4448  4480
FWTRP      1449   1567# 4433  4465
FWTSWP     1188   1570#
FWTTM      1569#  3305
FZREQ      1634#  1774  1780  1784  1818  1898  2027  4377  4387
GET         395#  2724  2756  2806  3571  4004  4016
GETFLD     1900   1979# 2012  4061
GETFLF     1989   1994#
GETFLP     1986#  1992
GETFLX     2008   2010#
GETFL2     1999   2009#
GETMQ         9     11# 1146  1334
GETTCB     3506   3712  3790  4078# 4085  4118  4147
GN         4202#  4203  4208  4209  4210  4211  4215  4217  4231  4245
           4255   4256  4278  4279  4282  4294  4295  4297  4303  4310
           4317   4324
GNA        4205   4217#
GNALP      4219#  4264
GNBR       4243#
GNDEL      4242   4259#
GNEX       4211#  4258
GNEX2      4253#  4261
GNFTEM     4206   4207  4213  4223  4224  4230  4244  4249  4260  4263
           4276   4280  4281  4292  4293  4296  4330#
GNNEXT     4235   4262#
GNPR       4219   4220  4246  4247  4248  4250  4251  4253  4268  4289#
GN2        2906   4302# 4307
GN20       4090   4323# 4328
GN3        1859   4309# 4314
GN5        1190   3003  4316# 4321
GPZCDF     2059   2060# 2091
GPZCT      2065#  2067  2073
GPZRIN     1915   2048# 2096  2159  2327
GPZRI2     2062   2084#
GPZTAD     2069   2070# 2071  2074  2089  2092
HARDMQ        7      8#    9    11   451
HCT          46#    58    59    60    61    62    63    64    65
HEST         13
HLPST      3788#  3795  3804  3814  3815  3822
HLTINS      249#   552
HP           47#    67    68    69    70    71    72    73    74
HPRIO       531#  1210  1216  1217  1553  1560
IDLE       4349   4407  4494#
```

```
IDLEIN    2685# 4498
IDLP      2682  2686# 2698
IDLP2     2696# 2700
IDNS      2675#
IDNSHL    2674# 2690
IDPTR     2680# 2693  2696  2697  2699
IDSW      2678#
IDSWHL    2677# 2692
IEXIT      458#  682   691   700   709   718   727   792   803   846
           855   864  2585  2657  2935  3917
INARG2    3014# 3020  3492  3923  4122
INIT       263   264#
INITIN    4513  4514#
INRQEX    1538  1556#
INRUNQ    1471  1531# 1557  1562  2816  2886  3828
INS200     560   570#
INTAC      253#  558   568   961   968  1084
INTCT      578   580   582   609   877#
INTDEF      29#
INTFL      254#  565   573   958   965  1082
INTPC      238#  950   962   971   977   983   984   986   987  1086
INTR       241   245   558   568#
INTR2      575#  926
INTSCD     891#  942   944
INT0       519#  907   909   911   912  1193  1438  1469  1482  1483
          1501  1503  1544  1545  1547  1549  2280  3512  3518  3551
          3552  3558  3659  3667  3668  3670  3676  3743  3745  3753
INT1       520#  901   908  1439  1443  1451  1457  1459  1476  1489
          1496  1532  1534  1535  1542  1543  1548  1550  1551  1577
          1578  1583  1584  1585  1590  1595  1597  1598  2815  2885
          3509  3519  3793  3794  3801  3809  3811  3812  3813  3816
          3817  3824  3825
INT2       521# 1444  1458  1467  1470  1473  1474  1477  1478  1481
          1491  1494  1502  1541  1546  1552  1555  1559  1581  1586
          1602  1604  3651  3746  3748  3755
INT3       522# 1155  1582  1587  1588  1593  1596  1599  1603  3800
ITSAF      949   980  1081#
JUMS       414#
KB1         39#   53   686   687
KB2         41#   62   704   705
KB3         43#   71   722   723
KCC1        54#
KCC2        63#
KCC3        72#
KEEP      3298#
KM8E        31#  859   860
KRB1        56#
KRB2        65#  708
KRB3        74#  726
KRS1        55#
KRS2        64#
KRS3        73#
KSF1        53#
KSF2        62#  706
KSF3        71#  724
LOCK      3254#
MARG1      526# 1727  1779  1783  1819  1899  1921  2732  2733  2738
          3319  3501  3502  3511  3534  3549  3716  3895  3915  3960
          4012  4050  4052  4135
MARG2      527# 3019  3505  3577  3919  3924  3943  4009  4010  4138
```

```
MAXDEV      44#  590   868
MAXSTL      22# 4125  4366
MCILF     3855  3859#
MCILP     3846# 3858
MCINTR    3844# 3861  3873  3888  3890  3893  3913
MCIOUT    3874  3896# 3920
MONAC      252#  256   256  1085  1110  1252  1297  1316  1360  1464
          2964  2976  3008  3064  3099  3677  3711  3779  3789  3853
          4033  4049  4053  4062  4071  4146
MONEX      928  1288  1345# 3525  3557  3679  4067  4187
MONEX0    3556# 3561
MONFL      255#  256   256  1083  1102  1121  1133  1239  1249  1250
          1267  1268  1270  1293  1322  1324  1350  1355  3083  3497
          3647  3648  3673  3674  3965  4176
MONFUN     525# 3316  3327  3336  3495  3515  3523  3591  3606  3611
          3685
MONIN     1913#
MONITO     454  3064#
MONIT2    3089  3315#
MONJMP    3339  3345# 3381
MONOUT    1091# 3614
MONPC      378# 1087  1112  1254  1298  1318  1362  3016  3018  3085
          3086  3090  3101  3112  3315  3317  3318  3342  3570  4022
          4023
MONSAF    1099# 1164
MONSCD    1093  1095  1149#
MONSF2    1107# 1388
MONSTA    3329  4425#
MON0       513# 1105  1106  1276  1280  1330  1336  2870  2871  2888
          2891  2896  2960  2962  2969  2974  2975  2978  2996  2998
          3001  3067  3322  3533  3538  3589  3593  3649  3652  3775
          3776  3778  3781  3782  3928  3930  4175  4181
MON1       514# 2729  2730  2782  2783  2789  2790  2794  2798  2801
          2879  2880  2884  2889  2890  2892  3655  3656  3658  3669
          3719  3726  4179  4180  4186
MON2       515# 3721  3730
MON3       516# 1386  1785  1786  1825  1879  1880  1907  1946  1983
          1987  1990  1996  2002  2009  2010  2021  2022  2028  2029
          2050  2053  2056  2058  2246  2247  2276  2322  3104  4065
          4066
MON76     3328  3334#
MQ          11   451# 1130  1335
MSAPP     1453  1455  1481#
MSAPP1    1491# 1497  1500
MSAPP2    1493  1501#
MSFREE     286# 2722
MSIN      1456#
MSNOD1     457  2994#
MSNOD2    3002  3008#
MSNREQ    2995  3003#
MSQMAX    1487  1498  1508#
MSREQ      285#
MSTOAC    3667#
MTFCT     2741  2802  2820#
MTFFLD    2746  2751  2754  2767# 2773  2792
MTFPAG    2743  2744  2747  2749  2759  2781  2787  2810  2825#
MTFTCH    1197  2822  4355  4449#
MTFTIN    2720# 2818  4453
MTINIT    4364  4393  4394  4509#
MTSDSK    2281  2576  2660  4361  4481#
```

```
MTSWAP    2288  2333  4352  4434#
MTSWEX    2328#
MTSWIN    2306# 4438
MTTIME    2899  2920  4358  4466#
MULINT     809  2929#
MULTCT    2931  2934  2938#
MULTIP      99   797   798   801   806  2925  2926
MUST        99  4375
MYCDIF     364# 2100
MYCIF      304#
M1         484# 3507  3791
M10        496#
M100       500#
M2         486# 1442  3927
M20        498# 4093
M200       502# 2788  4171
M3         488#
M4         490# 2883
M5         492# 3587  3703  3718
M55       2933  2939#
M6         494# 3713
M7771      472#
M7774      470#
M7775      468#
M7776      466#
NONFIT     594#  596   607   877
NONREP    3296  3514#
NRT        106#
OF          99
ON        4375
ONDISK    1188# 1226
OURQF     1592  1595#
OURQLP    1587# 1594
OURUNQ    1158  1576# 1601  1605  3802
PDP11      827   828
PDP8E        3#    8    84   239   244   557   871   918   957  1114
          1166  1234  2736  2753  3065  3069  3321  3964  3971
PDP8I       85   567   872   922   964  1127  1302  1405
PDP8IE      33#  813   814
PGR       1726# 1741  1744  1763  1821  1824  1828
PGREX     1796# 1869  1873  1920
PGREXM    1741#
PGRFF     1858  1870#
PGRLEN    1712# 1733  1768  1790  1846  1847  1851  1856  1866  1916
          1923
PGRLP     1736# 1770  1782  1839
PGRLPM    1739# 1849
PGRMB     1834  1851#
PGRMF     1840#
PGRM2     1844# 1854
PGRNM     1830  1855#
PGROLD    1713# 1737  1738  1739  1787  1840  1860  1863  4291
PGRQ      1762#
PGRQLP    1766#
PGRQN     1749# 1800  2742  4027  4054
PGRQX     1765  1898#
PGRREM    1711# 1771  1792  1817  1822  1832  1843  1864  1870  1924
PGRTEM    1710# 1766  1772  1789  1796  1826  1906  1922
PGRTN     1751  1813#
PGRTN2    1827# 1925
```

```
PRSF       742
PTBASE    1241  1281  1306# 1337
PTX       1262# 1279
PUT        431# 4099  4109  4114
P0CEX     2088  2095#
P0CLST    2084  2098#
P0CTAD    2065  2085  2086# 2093  2094
P11SIF     834
P11SOF     829
P8SIF      815
P8SOF      820
QUOT5      601#  602
RDFLD     1773  1823  1875# 1881
REQCDF    3260#
REQFLD    3231#
REQPAG    3221#
REQSTL    3236#
REQTCB    3245#
RESUME    3277#
RFCA      2628
RFINS     2618  2637#
RFINTR     777  2645#
RFMS      2640  2658  2662#
RFSTAT    2642  2654  2664#
RFTEMP    2631  2632  2634  2663#
RFWC      2596# 2624
RF08       765   766   769   774  2593  2594
RF500     2612  2665#
RIS        872   923
RKBLK     2530  2544  2548  2553  2581#
RKCHEK    2531  2541  2557  2579#
RKCMD     2504  2519  2542  2550  2574#
RKDONE    2535  2568#
RKERCT    2482  2566  2573#
RKERR     2510  2564#
RKER2     2497# 2567
RKER3     2499  2511#
RKINTR     761  2508  2559  2584  2586# 2589
RKLP      2532# 2562
RKMS      2488  2578#
RKMSP     2488# 2569
RKPAG     2525  2533  2538  2547  2580#
RKRTRY    2485  2516#
RK600     2505  2521#
RK8E        30#   91   749   750   753   758  2477  2478  2703
RLOCLP    2783# 2791  2803
RLOC1     2786  2792#
RLOC2     2793  2798#
RNTCB     3578  3695# 3709  4037
RP        2395  3170#
RPINTR     459   900#  902
RQPTRS     531  1540  1580  4390#
RQSTLF    4129  4135#
RQSTLP    4127# 4133
RTN       2968# 3010
RTNFLD    3233#
RTNPAG    3226#
RTNTCB    3249#
RTNTC2    3702  3708#
RTNTSK    3569  3613  4001# 4028  4082  4083  4098  4103  4104  4105
```

```
          4111  4126  4132
SBYTER     247  1167# 1169
SCA       1116
SCDF      1214  1219#
SCDF2     1272  1275  1326  1329#
SCDF3     1274  1285  1328  1341#
SCDINH     534#  933  1209  1221  1296  1352  1394  3078  3079
SCDLP2    1211# 1402
SCDREQ     532#  932  1206  1290  1347  1395  1561  3567
SCED       940  1140  1148  1163  1205# 1397  2902  3579
SCEDLP    1208# 1218
SCEDNI    1384  1394#
SCL       1119
SDERCT    2470# 2565  2651
SDEX      2394# 2422
SDFCT     2405  2407  2448#
SDGO      2391  2437  2479  2491  2602# 2609  2643
SDIN      2397# 4485
SDMSP     2390  2393  2396# 2399  2400
SDNCT     2385  2387  2446#
SDNORM    2383# 2403
SDPTR     2469# 2480  2490  2512  2513  2516  2521  2522  2526  2527
          2529  2568  2603  2604  2610  2614  2619  2620  2626  2627
          2629
SDSWP     2416  2421  2424# 2430  2440
SDSW1     2417#
SETAC     3677# 3687  3732  3818  3829  3897  4038  4055  4073  4143
          4148
SHOULD     256
SIDLE     4348# 4495
SIMINT    2484  3212#
SINT       861
SKPCHN     579   581   587   588   597   597   601   602   603#  603
           608   608   678   687   696   705   714   723   732   741
           750   766   782   798   814   828   842   851   860   868
          3845
SKPEND     590   591#  591   594
SKPFIT     592   593#  593   594
SKPSHF     602#  603
SMSQ      3539  3594  3645# 3662  3672  3688  3689  3690  3692
SMSQEX    3661  3684#
SMSQLP    3655# 3664
SMTFTC    4354# 4450
SMTINI    4363# 4510
SMTSDS    2771  4360# 4482
SMTSWA    4351# 4435
SMTTIM    4357# 4467
SNDMS     3152#
SNDREP     541  1433# 1479  1485  1506
SNDWTR    2769  3165#
STALL     3149#
STATX       77#   81   577   593   597   609   876   891   941  1092
          1149  1486  1495  1508  1608  2384  2404  2446  2448  2451
          2470  2565  2649  2673  2686  2687  3326  4210  4237  4254
          4267  4306  4313  4320  4327  4425
STLST      460  4347# 4366
STOP      3270#
STOPD     1571#
STORMQ      10# 1142  1320
STP0      3799# 3807
```

```
STTOMS    3887  3912  3922# 3926  3931
SUF       1359
SWAB      1256
SWAPMS    2265  2269  2270  2272  2274  2275  2277  2279  2290# 2310
          2314  2317  2318  2320  2321  2323  2328  2332  2401  2412
          2415  2420  2688  4431
SWBA      1264
SWNODE    2312  2414  2418  2419  2425  2428  2431  2435  2436  2442#
SWPOUT    3301#
SYDISK      91#  753   758   769   774  2478  2594
SYSHLT     551#  874  1731  1956  1993  2439  2780  3093  3384  3849
          4088  4134  4184  4222
SYTIME      94#  785   790   801   806  2926
SYWAIT    2502  2507  2558  2570  2583# 2591
TCF1        50#
TCF2        59#  699
TCF3        68#  717
TCKLEN      84    85# 2674  2677  2694
TIME      2865  2867  2923#
TIMEIN    2864# 4470
TIMEMS     793  2898  2922# 2936  4463
TIMEOU    3149  3305  3306  3605#
TLS1        52#
TLS2        61#
TLS3        70#
TOQEX     2873  2898#
TOQFN     2878  2888#
TOQLP     2871# 2895  2897
TOQLP2    2881  2896#
TPC1        51#
TPC2        60#
TPC3        69#
TQHEAD    2869  2907  2910  2918#
TQIN      2905# 2916  3608
TSBLOK     100# 2764
TSF1        49#
TSF2        58#  697
TSF3        67#  715
TSKSAF    1161# 1292  1349
TSKSWT     954   977#
TSTTM     3692# 3696  3697  3700  3708
TSWAIT    3542  3603#
TT1         38#   49   677   678
TT2         40#   46    58   695   696
TT3         42#   47    67   713   714
T1ON       802  2930
T1SKP      799
UNLOCK    3257#
VCALL      305   315   339   379#  951  3084
VCDF       326# 2766
VCDFTM     335   337   342# 3102
VCDF1     3097#
VCDIF      363#  365
VCURIF     529# 1108  1242  1243  1307  1308  4024
VERHLT     314# 3091
VFBLOK      98#   99    99   100  2432
VFLCT     2140  2152  2159#
VFLESS    2138# 2154  2156  2157  2267
VFLF      2151  2155#
VFLOCK    1944# 1959  2007  2026
```

```
VFLTEM    1951  1952  1958  1961#
VFLTM2    1945  1953  1962#
VFLTRY    2141# 2153
VFNEXT    2136# 2141  2146  2147  2148  2155
VFSWLP    2302#
VGET       397#  400   403   404  2101
VIRMAX      17#  100  1984  4385  4386  4414  4511  4511
VJUMS      416#  419   436   440  1710  2102
VMSNOD     287# 3009
VPT       2705
VPUT       418   434#  437   442   443   446  2103
VRCDF      301#
VRCDF1    3095  3110#
VRCDF3    1283  1339# 3130  3784
VRDF       352#
VRDF1      456  2954#
VVCDF      329#  332   333   344  1712  2099  3100
VVRCDF     303#  314   331   343  1283  1339  1713  2098  3098  3111
VVRDF      353# 1711  2965
WTMS      2398  2721  3175# 3290
WTRP      2487  2639  3157# 3293
WTRPMT    2334  2900#
X          401   402   439   441   450# 1141  1258  1259  1262  1265
          1333  2750  2755  2760  2761  2763  2811  4014  4015  4019
          4020  4026  4094  4117  4124  4127  4130  4131  4136  4137
          4139  4140  4102
XCDF70     340   462# 2057  2260
XCDIF      289# 3969
XCHRCQ    3586#
XCHRPQ    3530#
XCLINT    3355  3912#
XCLI2     3913# 3944
XCNINT    3354  3887#
XDISIN    3353  3873#
XERROR    3350  3374  3376  3377  3378  3379  3380  3384#
XEXIT     3371  3567#
XFILTC    3375  3711#
XFLDTA     334   455# 1878  1982  2020  2245  2271
XFRINT    3356  3943#
XLOCK     3368  4172#
XMONIT     382   454#
XMSNOD     290   457#
XREQCD    3370  3773#
XREQFL    3361  4059#
XREQPA    3359  4049#
XREQST    3363  4122#
XREQTC    3365  4146#
XRESUM    3373  3822#
XRP       3351  3548#
XRP2      3554  3558#
XRTNFL    3362  4071#
XRTNPA    3360  4050#
XRTNTC    3366  4033#
XSIMIN    3357  3954#
XSNDMS    3347  3488#
XSNDWT    3349  3492#
XSND2     3504#
XSTALL    3346  3602#
XSTOP     3372  3804#
XUNLOC    3369  4171#
```

```
XVCALL      339# 1282  1338
XVRDF1      356   456#
XWTMS      3352  3574  3587#
XWTRP      3348  3531#
X0          261#  562   571  1101  1109  1111  1113  1118  1124  1132
           1135  1145  1147  1191  1192  1194  1196  1219  1220  1224
           1227  1251  1253  1255  1257  1275  1277  1315  1317  1319
           1321  1329  1331  1381  1385  1399  1742  1743  1767  1788
           1791  1795  1835  1841  1842  1844  1861  1862  1865  1868
           2064  2072  2874  2876  2908  2909  2913  2914  3004  3005
           3006  3125  3126  3330  3331  3333  3699  3707  3714  3725
           3729  3846  3847  3857  3860  3892  3894  3896  3916  3918
           4091  4095  4214  4225  4226  4227  4228  4252
x1          262# 3717  3723  3728
x2          263#
x3          265#
x6000       240# 3071
ZCORMA      540# 1104  1950  2145  2959  4178
ZFN         542# 2893  2999  3559  3704
ZSNDRE      541#  914  1198  2282  3520  3555
ZSTL        460# 2728  4008  4036  4081  4123
Z1000       543# 2324  3590  3602  4059
_00377      884
_00570     1173
_00572     1143
_00573     1141
_00574      951
_00710     1406
_00771     1283
_00772     1273
_00774     1197
_00777     1188  1295
_01100     1609
_01175     1540  1580
_01176     1464
_01366     1884
_01371     1855
_01373     1784  1818
_01374     1780
_01375     1774
_01377     1734
_01570     2034
_01571     2027
_01572     1984
_01573     1980
_01577     1898
_01771     2161
_01772     2142
_01773     2139
_01774     2084
_01775     2068
_01776     2066
_01777     2063
_02150     2453
_02161     2436
_02162     2432
_02163     2413
_02164     2401
_02170     2332
_02172     2313
```

```
_02173     2311
_02174     2281
_02175     2279
_02370     2694
_02371     2692
_02372     2690
_02373     2688
_02374     2556
_02375     2518
_02376     2517
_02377     2497
_02573     2829
_02575     2764
_02577     2731
_02765     3023
_02766     3009
_02770     2965
_02775     2898
_02777     2869
_03166     3387
_03167     3329
_03173     3102
_03174     3100
_03175     3098   3111
_03176     3091
_03177     3084
_03364     3617
_03366     3610
_03573     3761
_03574     3720
_03576     3698
_03762     3978
_03764     3955
_03766     3917
_03767     3889
_03770     3845
_03773     3805
_03777     3773
_04165     4151
_04166     4125
_04177     4011
_04375     4333
_04376     4218

V3A
```

```
                        /APP. A6. MC8IN.
                /INITIALISE AND START MC8 SYSTEM.

        0001    FIELD 1
        0200    *200
10200   4777'   INIT,    JMS      PRTXT;TESTRT
10201   0600
10202   4776'            JMS      CRLF
10203   6202             CIF 0
10204   4775'            JMS      PG0IN   /GET GENERAL PAGE 0 IN.
10205   4774'            JMS      AVINIT
10206   1373             TAD      (INS200 /RESTORE START ADDRESS IN FLD0
10207   6201             CDF 0
10210   3772             DCA I    (200
10211   6211             CDF 10
10212   4776'   INIT2,   JMS      CRLF
10213   6041             TSF
10214   5213             JMP      .-1
                IFDEF PDP8E <
10215   6007             CAF >
                IFDEF PDP8I <
                         P11IED
                         P8ID
                         6611
                         6601
                         DTCA DTXA
                         P11SOF
                         NOP
                         P11CLF
                         P11SIF
                         NOP
                         P11CLF
                         P8CRD
                         P8COF
                         RRB
                         PCF
                         TCF
                         CCF
                         CRF
                         PLCF
                         FPCR
                         FRCR
                         PRCF
                         BRCI
                         KCC >

                /INIT CLOCK
                IFDEF DK8EP <
                IFZERO SYTIME-DK8EP <
10216   7240             CLA CMA
10217   6130             CLZE
10220   7200             CLA
10221   1371             TAD      (-12
10222   6133             CLAB
10223   1370             TAD      (5210+12
10224   6132                    CLOE
10225   7200                    CLA
10226   6135                    CLSA
10227   7200                    CLA
                 >>
                IFDEF MULTIP <
                IFZERO SYTIME-MULTIP <
                        T1ON
                 >>

10230   6203                    CIF CDF 0
10231   5767'                   JMP     SCED

10367   0617
10370   5222
10371   7766
10372   0200
10373   6004
10374   0400
10375   5200
10376   1066
10377   1123
        0400'   PAGE
```

```
             /AVAILLIST IN FLD 0.
             /DURING ASSEMBLY AN AVAILLIST IS BUILT IN FLD 0.
             /ITS FIRST NODE IS POINTED TO BY AVPT.
             /EACH NODE HAS THE FOLLOWING LAYOUT:
             /W0:    PTR TO NEXT NODE (0 INDICATES THE END)
             /W1:    PTR TO LAST LOC OF NODE.
             /EACH NODE IS AT LEAST 3 LOCS LONG.
             /WE WILL INVERT THIS CHAIN AND COUNT ITS TOTAL LENGTH.
             /NODES ALLOCATED IN THE RANGE 1-4000 ARE BROKEN SUCH
             /THAT THEIR MAX LENGTH <=17 (NO TCBS <=4000).

10400  0000  AVINIT, 0                /INITIALIZE AVAILLIST.
10401  1377          TAD    (AVPT
10402  3307          DCA    AVPTR    /PTR TO NEXT NODE.
10403  1376          TAD    (20      /TCB OF MTINIT WILL BE ADDED TO
10404  3312          DCA    AVCT     /AVAILLIST. INITIALIZE COUNTER
10405  3311          DCA    AVOLD    /WE WILL INVERT THE CHAIN AS IT IS
                                     /IN CORE, SO THIS NODE WILL BECOME
                                     /THE LAST ONE.
10406  6201          CDF 0
10407  1307  AVINLP, TAD    AVPTR    /PTR TO NEW NODE.
10410  7450          SNA
10411  5233          JMP    AVIN2    /NO MORE NODES TO DO.
10412  7001          IAC             /MAKE PTR TO PTR TO NODE'S LAST LOC.
10413  3310          DCA    AVPTR2
10414  1307          TAD    AVPTR    /FIRST LOC OF NODE
10415  7041          CIA
10416  7001          IAC             /-(FIRST LOC) +1
10417  1710          TAD I  AVPTR2   /-(FIRST LOC) +1 +(LAST LOC)=LENGTH.
10420  1312          TAD    AVCT
10421  3312          DCA    AVCT     /ADDED LENGTH TO COUNTER.
10422  1707          TAD I  AVPTR
10423  3310          DCA    AVPTR2   /PTR TO NEXT NODE.
10424  1311          TAD    AVOLD    /INVERT CHAIN, STORE PTR TO PRECESSOR.
10425  3707          DCA I  AVPTR
10426  1307          TAD    AVPTR    /SAFE THE NEW PRECESSOR
10427  3311          DCA    AVOLD
10430  1310          TAD    AVPTR2   /NEXT NODE BECOMES NEW NODE.
10431  3307          DCA    AVPTR
10432  5207          JMP    AVINLP
10433  1311  AVIN2,  TAD    AVOLD
10434  3775          DCA I  (AVHEAD /PASS HEAD OF AVAILLIST TO MC8.
10435  1311          TAD    AVOLD
10436  3307          DCA    AVPTR    /KEEP LOW NODES SHORTER THAN 20.
10437  1307  AVNLP,  TAD    AVPTR    /MORE TO DO?
10440  7450          SNA
10441  5300          JMP    AVEND
10442  7710          SPA CLA         /LOW NODE?
10443  5275          JMP    AVNLP2   /NO
10444  1307          TAD    AVPTR
10445  7001          IAC             /PTR TO PTR TO LAST LOC
10446  3310          DCA    AVPTR2
10447  1307          TAD    AVPTR
10450  7041          CIA
10451  7001          IAC             /-(FIRST LOC) +1
```

```
10452  1710            TAD I   AVPTR2  /LENGTH
10453  7100            CLL
10454  1374            TAD     (-20    /LOW NODE AND TOO LONG?
10455  7620            SNL CLA
10456  5275            JMP     AVNLP2  /NOT TOO LONG.
10457  1307            TAD     AVPTR   /TOO LONG NODE. BREAK OFF 16 WORDS.
10460  1373            TAD     (16
10461  3311            DCA     AVOLD   /JUST A PTR TO NEW NODE.
10462  1707            TAD I   AVPTR
10463  3711            DCA I   AVOLD   /NEW NODE IN CHAIN.
10464  2311            ISZ     AVOLD
10465  1710            TAD I   AVPTR2  /GET PTR TO LAST LOC
10466  3711            DCA I   AVOLD   /AND STORE
10467  7040            CMA
10470  1311            TAD     AVOLD
10471  3707            DCA I   AVPTR   /UPDATE PRECESSOR.
10472  7344            ACM2
10473  1311            TAD     AVOLD
10474  3710            DCA I   AVPTR2
10475  1707    AVNLP2, TAD I   AVPTR   /NEXT NODE.
10476  3307            DCA     AVPTR
10477  5237            JMP     AVNLP
10500  6211    AVEND,  CDF 10
10501  4772'           JMS     CRLF
10502  4771'           JMS     PRTXT;TEAV
10503  0623
10504  1312            TAD     AVCT
10505  4770'           JMS     PRDEC
10506  5600            JMP I   AVINIT

10507  0000    AVPTR,  0
10510  0000    AVPTR2, 0
10511  0000    AVOLD,  0
10512  0000    AVCT,   0

10570  1000
10571  1123
10572  1066
10573  0016
10574  7760
10575  4303
10576  0020
10577  5142
       0600    PAGE
```

```
10600  4040  TESTRT, TEXT    '                *** MULTI CORE 8 ***'
10601  4040
10602  4040
10603  4040
10604  4040
10605  4040
10606  4040
10607  4040
10610  5252
10611  5240
10612  1525
10613  1424
10614  1140
10615  0317
10616  2205
10617  4070
10620  4052
10621  5252
10622  0000
10623  1116  TEAV,   TEXT    'INITIAL SPACE IN AVAILLIST: '
10624  1124
10625  1101
10626  1440
10627  2320
10630  0103
10631  0540
10632  1116
10633  4001
10634  2601
10635  1114
10636  1411
10637  2324
10640  7240
10641  0000

       1000  PAGE
```

```
             /ROUTINE IN FLD0 TO GET PAGE 0  INTO FLD 1.
             /NOTE! THE LAST NODE IN THE AVAIL LIST MUST
             /BE LONG ENOUGH TO HOLD THIS ROUTINE.
       0000  FIELD 0
       5144  *AVPT+2
       5200  PAGE
05200  0000  PG0IN,  0
05201  3217          DCA     PT0
05202  3220          DCA     PT1
05203  1222          TAD     M150
05204  3221          DCA     CTX
05205  6201  PG0LP,  CDF 0
05206  1617          TAD I   PT0
05207  6211          CDF 10
05210  3620          DCA I   PT1
05211  2217          ISZ     PT0
05212  2220          ISZ     PT1
05213  2221          ISZ     CTX
05214  5205          JMP     PG0LP
05215  6213          CIF CDF 10
05216  5600          JMP I   PG0IN
             /WARNING:!!!!!!
             /THE "CIF CDF CUR"S ARE NOT CORRECT.

05217  0000  PT0,    0
05220  0000  PT1,    0
05221  0000  CTX,    0
05222  7630  M150,   -150
```

```
                  /MTINIT.
                  /MONITOR TASK FOR INITIALISATION.
                  /BE AWARE OF ERRORS IN PAGE 0.
                  /THE "CIF CDF CUR"S ARE INCORRECT!!!

          0001    FIELD 1
          2000    *INITIN
   12000  4054            CALL;   REQSTL  /INSERT TASKLOADER IN STL.
   12001  0041
   12002  0001            1;1437          /ALLOACTED IN TASKLIBRARY FROM BLOCK 1.
   12003  1437
   12004  3253            DCA     TSLDX
   12005  4014            MSREQ
   12006  3236            DCA     MSP1
   12007  1236            TAD     MSP1
   12010  3110            DCA     X
   12011  6201            CDF 0
   12012  1377            TAD     (0714   /GL
   12013  3510            DCA I   X
   12014  2110            ISZ     X
   12015  1376            TAD     (1702   /OB
   12016  3510            DCA I   X
   12017  2110            ISZ     X
   12020  1375            TAD     (0114   /AL
   12021  3510            DCA I   X
   12022  6211            CDF 10          /WE KNOW IN WHICH FLD WE ARE.
   12023  1236            TAD     MSP1
   12024  4246            JMS     SEND    /SEND MS TO LOAD FAMILY OF GLOBAL TASKS.
   12025  4014            MSREQ
   12026  3245            DCA     MSX
   12027  6201            CDF 0
   12030  1253            TAD     TSLDX   /PASS STATIC TASKNUM OF TSLD
   12031  3645            DCA I   MSX
   12032  1245            TAD     MSX
   12033  4246            JMS     SEND    /PASS STSK# OF TSLD TO TASKLOADER.
   12034  4054            CALL;   WTRP
   12035  0005
   12036  0000    MSP1,           0       /WAIT UNTIL LOADING COMPLETED.
   12037  4014            MSFREE          /GET COMPLETION STATUS OF TSLD.
   12040  7440            SZA             /MUST BE 0.
   12041  5244            JMP     LDER    /LOAD ERROR.
   12042  4054            CALL
   12043  0054            EXIT
   12044  4023    LDER,   ERHLT

   12045  0000    MSX,    0

   12046  0000    SEND,   0               /SEND MS TO TASKLOADER.
   12047  3252            DCA     MSP
   12050  4054            CALL
   12051  0003            SNDMS
   12052  0000    MSP,            0
   12053  0000    TSLDX,          0
   12054  5646            JMP I   SEND
```

```
 296    12175  0114
 297    12176  1702
 298    12177  0714
 299                 $
```

```
AVCT    0512
AVEND   0500
AVINIT  0400
AVINLP  0407
AVIN2   0433
AVNLP   0437
AVNLP2  0475
AVOLD   0511
AVPTR   0507
AVPTR2  0510
CRLF    1066
CTCTST  1111
CTX     5221
DECBIN  1043
INIT    0200
INIT2   0212
LDER    2044
MSP     2052
MSP1    2036
MSX     2045
M150    5222
PG0IN   5200
PG0LP   5205
PRDBOX  1041
PRDEC   1000
PRDECI  1015
PRDEC0  1007
PRDEC1  1011
PRDEC2  1013
PRHALF  1140
PRKCT   1104
PRKLP   1052
PRKTM   1103
PROCT   1046
PRTXLP  1127
PRTXT   1123
PT0     5217
PT1     5220
SEND    2046
SPACE   1105
TEAV    0623
TESTRT  0600
TSLDX   2053
TYPE    1074
ZFLAG   1042
```

```
ERRORS DETECTED! 0
LINKS GENERATED! 9
 300
```

```
ACM2        150
AVCT         94   108   109   160   167#
AVEND       125   156#
AVHEAD      120
AVINIT       11    90#  162
AVINLP       99#  118
AVIN2       101   119#
AVNLP       123#  155
AVNLP2      127   138   153#
AVOLD        95   112   115   119   121   141   143   144   146   148
            151   166#
AVPT         91   218
AVPTR        92    99   104   110   113   114   117   122   123   128
            131   139   142   149   153   154   164#
AVPTR2      103   107   111   116   130   134   145   152   165#
BRCI         44
CALL        249   277   283   291
CCF          38
CLAB         54
CLOE         56
CLSA         58
CLZE         51
CRF          39
CRLF          8    16   157
CTX         224   231   240#
DK8EP        48    49
DTCA         26
DTXA         26
ERHLT       285
EXIT        284
FPCR         41
FRCR         42
INIT          6#
INITIN      248
INIT2        16#
INS200       12
LDER        282   285#
MSFREE      280
MSP         290   293#
MSP1        255   256   268   279#
MSREQ       254   270
MSX         271   274   275   287#
MULTIP       61    62
M150        223   241#
PDP8E        19
PDP8I        21
PG0IN        10   220#  234
PG0LP       225#  232
PLCF         40
PRCF         43
PRDEC       161
PRTXT         6   158
PT0         221   226   229   238#
PT1         222   228   230   239#
P11CLF       29    32
P11IED       22
P11SIF       30
P11SOF       27
P8COF        34
```

```
P8CRD         33
P8ID          23
REQSTL       250
SCED          67
SEND         269     276     289#    295
SNDMS        292
SYTIME        49      62
TEAV         159     197#
TESTRT         7     178#
TSLDX        253     273     294#
T1ON          63
WTRP         278
X            257     260     261     263     264     266
_10370        55
_10371        53
_10372        14
_10373        12
_10573       140
_10574       136
_10575       120
_10576        93
_10577        91
_12175       265
_12176       262
_12177       259

V3A
```

```
                 /APP. B1. RTTY.TK
             /READ CHARS FROM CONSOLE TELETYPE, BUILD A LINE OF TEXT
             /AND SEND IT TO WTTY.
             /TO BE ASSEMBLED WITH MC8PAL.
             /
             /CHARS ARE READ FROM THE CONSOLE TELETYPE, A LINE OF TEXT IS
             /ASSEMBLED, RUBOUTS ERASE THE LAST CHAR TYPED,
             /MULTIPLE RUBOUTS ARE ALLOWED,
             /THE LINE IS TERMINATED BY A CARRIAGE RETURN (215).
             /THE TEXT LINES ARE STORED IN A BUFFER REQUESTED FROM
             /THE PAGE ALLOCATOR.
             /CHARACTERS ARE ECHOED BY SENDING CHARACTER-MESSAGES TO WTTY.
             /WHEN A LINE IS COMPLETED (215 SEEN), THE LINE IS SENT TO WTTY
             /USING ALINE-MESSAGE.
             /WTTY RETURNS THE BUFFER TO THE PAGE ALLOCATOR.
             /
             /LAYOUT OF CHARACTER-MESSAGE:
             /       W1 B4-11        CHARACTER
             /       W2              -1
             /LINE-MESSAGE:
             /       W1 B0-4         PAGE ADDRESS OF TEXT BUFFER
             /       W1 B6-11        VIRTUAL FIELDNUMBER OF TEXT BUFFER
             /       W2              0

       0000  *0
00000  4002  4002    /CONTENTS OF ENTRY IN STL: PRIORITY 4, LENGTH 2.
00001  4000  4000    /AUTOSTART. IS STARTED WHEN LOADED.
00002  2405  TASKNAME TEST.RTTY
00003  2324
00004  0000
00005  2224
00006  2431
       0203  PAGE

       0107  CHAR=BASE               /STORE CHAR IN BASE REGISTER.

00203  0000          0               /STOP RELOCATION.
             /INITIALISATION.
00204  3211  START,  DCA     MSCL    /ARRIVES WITH MSPTR IN AC.
00205  1377          TAD     (KSF    /CLAIM KEYBOARD INTSLOT
00206  4054          CALL;   CLINTR
00207  0023
00210  6036                  KRB     /CLEAR AND READ
00211  0000  MSCL,           0
00212  1211          TAD     MSCL
00213  3600          DCA I   )MSRD   /HAND MESSAGE TO READ ROUTINE.

             /START AT A NEW LINE.
00214  4054  NEWLIN, CALL;   REQPAG  /REQUEST ONE PAGE BUFFER FROM PAGE
00215  0031
                                     /ALLOCATOR.
00216  0001                  1
00217  4054          CALL;   REQCDF  /PREPARE FOR FIELD ACCESS USING THE VRCDF
00220  0052
                                     /PSEUDO INSTRUCTION
```

```
00221  3314           DCA     BUFAD   /STORE BUFFER ADDRESS
00222  1314           TAD     BUFAD   /SET BUFFER POINTER IN X, SET COUNTER.
00223  0146           AND     C7600
00224  3110           DCA     X
00225  1376           TAD     (-177   /ALLOW 127 CHARS ON A LINE.
00226  3315           DCA     CT

               /READ CHARACTER LOOP.
00227  4601    CHARLP, JMS I   )CHREAD /READS A CHAR, STORES IN CHAR.
00230  1107           TAD     CHAR    /IS IT RUBOUT
00231  1375           TAD     (-377
00232  7450           SNA
00233  5277           JMP     RUB     /YES IT IS.
00234  1374           TAD     (377-215/IS IT CR?
00235  7650           SNA CLA
00236  5250           JMP     LINEND  /YES, TERMINATE LINE.
               /NO SPECIAL CHARS, ECHO AND INSERT IN BUFFER.
00237  1107           TAD     CHAR
00240  4602           JMS I   )ECHO
00241  1107           TAD     CHAR
00242  4020           VRCDF           /CDF TO BUFFER FIELD.
00243  3510           DCA I   X
00244  4051           CDFCUR
00245  2110           ISZ     X
00246  2315           ISZ     CT      /BUFFER FULL?
00247  5227           JMP     CHARLP  /NO.
               /FULL BUFFER; FALL INTO LINE END.

00250  1373    LINEND, TAD     (215    /ECHO CR.
00251  4602           JMS I   )ECHO
00252  1373           TAD     (215    /INSERT CR INTO BUFFER
00253  4020           VRCDF
00254  3510           DCA I   X
00255  4051           CDFCUR
               /SEND THE LINE-MESSAGE.
00256  4014           MSREQ
00257  3110           DCA     X
00260  1110           TAD     X
00261  3272           DCA     MSSD    /PASS TO SEND ROUTINE.
00262  1314           TAD     BUFAD
00263  6201           CDF 0
00264  3510           DCA I   X
00265  2110           ISZ     X
00266  3510           DCA I   X
00267  4051           CDFCUR
00270  4054           CALL;   SNDWTR
00271  0007
00272  0000    MSSD,           0
00273  0000           LCTASK WTTY
               /NOTE, WTTY RETURNS THE BUFFER.
00274  4014           MSFREE          /RETURN THE MESSAGE.
00275  7200           CLA             /REMOVE RUBBISH
00276  5214           JMP     NEWLIN

               /RUBOUT. ECHO "\" IF A CHAR IS ACTUALLY DELETED.
```

```
00277  1110  RUB,    TAD     X
00300  0132          AND     C177     /FETCH NUMBER OF CHARS IN BUFFER.
00301  7650          SNA CLA
00302  5227          JMP     CHARLP   /EMPTY BUFFER.
00303  1372          TAD     ("\
00304  4602          JMS I   )ECHO
00305  7240          CLA CMA
00306  1110          TAD     X
00307  3110          DCA     X        /DECREMENT POINTER
00310  7240          CLA CMA
00311  1315          TAD     CT
00312  3315          DCA     CT       /DECREMENT COUNTER
00313  5227          JMP     CHARLP

00314  0000  BUFAD,  0
00315  0000  CT,     0

00200  0403
00201  0400
00202  0412
00372  0334
00373  0215
00374  0162
00375  7401
00376  7601
00377  6031
       0400  PAGE
```

```
                     /READ CHARACTER.
 00400   0000   CHREAD, 0        /THIS 0 STOPS RELOCATION.
 00401   4054           CALL;    WTRP     /WAIT FOR REPORT OF INTERRUPT MESSAGE
 00402   0005
 00403   0000   MSRD,            0
 00404   7200           CLA               /ERASE MSPTR FROM AC.
 00405   6201           CDF 0             /FETCH CHAR FROM MESSAGE
 00406   1603           TAD I    MSRD
 00407   4051           CDFCUR
 00410   3107           DCA      CHAR
 00411   5600           JMP I    CHREAD

                /ECHO THE CHARACTER IN AC.
 00412   0000   ECHO,   0
 00413   3234           DCA      TEM
 00414   4014           MSREQ
 00415   3231           DCA      MSSD2
 00416   1234           TAD      TEM      /FETCH CHAR AGAIN
 00417   6201           CDF 0
 00420   3631           DCA I    MSSD2    /STORE MESSAGE W1
 00421   7001           IAC
 00422   1231           TAD      MSSD2
 00423   3234           DCA      TEM      /POINTER MESSAGE W2
 00424   7240           CLA CMA
 00425   3634           DCA I    TEM      /-1 TO MESSAGE W2.
 00426   4051           CDFCUR
 00427   4054           CALL;    SNDMS+NONREP    /DONT WAIT FOR THIS REPORT.
 00430   2003
 00431   0000   MSSD2,           0
 00432   0000           LCTASK WTTY
 00433   5612           JMP I    ECHO

 00434   0000   TEM,    0
 171    $
```

```
BUFAD  0314
CHAR   0107
CHARLP 0227
CHREAD 0400
CT     0315
ECHO   0412
LINEND 0250
MSCL   0211
MSRD   0403
MSSD   0272
MSSD2  0431
NEWLIN 0214
RUB    0277
START  0204
TEM    0434
```

```
ERRORS DETECTED! 0
LINKS GENERATED! 0
```

```
BASE       35
BUFAD      56     57     95    125#
CALL       41     49     53    101    140    164
CDFCUR     78     89    100    146    163
CHAR       35#    65     73     75    147
CHARLP     64#    81    114    123
CHREAD     64    139#   148
CLINTR     42
CT         61     80    121    122    126#
C177      112
C7600      58
ECHO       74     85    116    151#   168
LCTASK    104    167
LINEND     71     84#
MSCL       39     44#    45
MSFREE    106
MSRD       46    142#   145
MSREQ      91    153
MSSD       94    103#
MSSD2     154    157    159    166#
NEWLIN     49#   108
NONREP    165
REQCDF     54
REQPAG     50
RTTY       28
RUB        68    111#
SNDMS     165
SNDWTR    102
START      39#
TASKNA     28
TEM       152    155    160    162    170    171#
TEST       28
VRCDF      76     87
WTRP      141
WTTY      104    167
X          59     77     79     88     92     93     97     98     99    111
          118    119
_00200     46
_00201     64
_00202     74     85    116
_00372    115
_00373     84     86
_00374     69
_00375     66
_00376     60
_00377     40

V3A
```

```
                    /APP. B2. WTTY.TK
                    /WRITE CHARACTERS ON CONSOLE TELETYPE
                    /TO BE ASSEMBLED WITH MC8PAL.
                    /THIS TASK OPERATES IN TWO MODES:
                    /LINEMODE, ECHOING A FULL LINE OF TEXT, TERMINATED BY
                    /       A CARRIAGE RETURN (215);
                    /CHARACTERMODE, ECHOING A SINGLE CHARACTER.
                    /MODE CHANGES CAN OCCUR AFTER EACH CARRIAGE RETURN.
                    /IN CHARACTERMODE THE TASK REMAINS CLAIMED UNTIL THE
                    /NEXT CARRIAGE RETURN.
                    /AFTER EACH CARRIAGE RETURN A LINEFEED IS ADDED.
                    /IN LINEMODE THE TEXTBUFFER IS RETURNED TO THE PAGE ALLOCATOR.

                    /MESSAGE LAYOUT:
                    /CHARACTERMODE: W1 B4-11 CHARACTER
                    /               W2       -1
                    /LINEMODE:      W1 B0-4  PAGE ADDRESS OF TEXT LINE
                    /               W1 B6-11 VFLD# OF TEXT LINE
                    /               W2       0

              0000  *0
       00000  4001  4001            /CONTENTS OF ENTRY IN STL: PRIORITY 4, LENGTH 1.
       00001  0000  0000            /NO AUTOSTART
       00002  2405  TASKNAME TEST.WTTY
       00003  2324
       00004  0000
       00005  2724
       00006  2431
              0200  PAGE

       00200  0000          0                   /STOP RELOCATION
       00201  3316  START,  DCA      MSREP      /SAVE MESSAGE PTR

                    /INITIALIZATION: CLAIM INTSLOT OF TELETYPE.

       00202  4014          MSREQ
       00203  3210          DCA      MSCL
       00204  1377          TAD      (TSF
       00205  4054          CALL
       00206  0023                   CLINTR
       00207  6042                   TCF
       00210  0000  MSCL,            0          /PTR TO INTERRUPT MESSAGE
       00211  1210          TAD      MSCL
       00212  3270          DCA      MSWR       /PASS MESSAGE PTR TO WRITE ROUTINE
       00213  1316          TAD      MSREP      /FETCH MESSAGEPTR IN AC
       00214  5217          JMP      NEWMOD+2 /FALL INTO NEWMODE LOOP

                    /WE HAVE SEEN A CARRIAGE RETURN, READY TO CHANGE MODE.
       00215  4054  NEWMOD, CALL
       00216  0014                   WTMS       /TERMINATE PREVIOUS CLAIM
       00217  4273          JMS      MSREAD     /READ MESSAGE INTO W1 AND W2
       00220  2314          ISZ      W2
       00221  5234          JMP      LINMOD     /LINEMODE

                    /CHARACTERMODE
```

```

00222   1313    CHRMOD,   TAD      W1
00223   4264              JMS      WRITE
00224   1313              TAD      W1
00225   1376              TAD      (-215
00226   7650              SNA CLA
00227   5261              JMP      MODEND   /LAST CHAR, ECHO LF.
00230   4054              CALL
00231   1014                       WTMS+KEEP /KEEP CLAIMED.
00232   4273              JMS      MSREAD
00233   5222              JMP      CHRMOD

                /LINEMODE

00234   1313    LINMOD,   TAD      W1
00235   4054              CALL
00236   0052                       REQCDF   /PREPARE FOR ACCESSING THE TEXTLINE
                                            /WITH THE VRCDF PSEUDO INSTRUCTION.
00237   0146              AND      C7600
00240   3110              DCA      X        /SET BUFFER PTR
00241   4020    LNLP,     VRCDF
00242   1510              TAD I    X
00243   4051              CDFCUR
00244   2110              ISZ      X
00245   3312              DCA      CHAR
00246   1312              TAD      CHAR
00247   4264              JMS      WRITE
00250   1312              TAD      CHAR
00251   1376              TAD      (-215
00252   7640              SZA CLA
00253   5241              JMP      LNLP
00254   1313              TAD      W1       /CONTAINS STILL PAGE ADDRESS + VFLD#
00255   4054              CALL
00256   0033                       RTNPAG   /RETURN TEXT BUFFER
00257   0001                       1
00260   7200              CLA

                /READY WITH BOTH MODES. ECHO LF
00261   1375    MODEND,   TAD      (212
00262   4264              JMS      WRITE
00263   5215              JMP      NEWMOD   /READY. START NEW MODE

00264   0000    WRITE,    0
00265   6044              TPC
00266   4054              CALL
00267   0005                       WTRP
00270   0000    MSWR,              0
00271   7200              CLA
00272   5664              JMP I    WRITE


00273   0000    MSREAD,   0
00274   3310              DCA      MSP
00275   1310              TAD      MSP
00276   3315              DCA      TEMP
```

```
    00277  6201           CDF 0
    00300  1715           TAD I     TEMP
    00301  2315           ISZ       TEMP
    00302  3313           DCA       W1
    00303  1715           TAD I     TEMP
    00304  4051           CDFCUR
    00305  3314           DCA       W2
    00306  4054           CALL
    00307  0013                     RP
    00310  0000  MSP,               0
    00311  5673           JMP I     MSREAD

    00312  0000  CHAR,    0
    00313  0000  W1,      0
    00314  0000  W2,      0
    00315  0000  TEMP,    0
    00316  0000  MSREP,   0
```

```
 128    00375   0212
 129    00376   7563
 130    00377   6041
 131    $
```

```
$

CHAR    0312
CHRMOD  0222
LINMOD  0234
LNLP    0241
MODEND  0261
MSCL    0210
MSP     0310
MSREAD  0273
MSREP   0316
MSWR    0270
NEWMOD  0215
START   0201
TEMP    0315
WRITE   0264
W1      0313
W2      0314
```

```
ERRORS DETECTED: 0
LINKS GENERATED: 0
```

```
CALL       39    49    63    71    88   100   118
CDFCUR     78   116
CHAR       80    81    83   123#
CHRMOD     57#   66
CLINTR     40
C7600      74
KEEP       64
LINMOD     53    70#
LNLP       76#   86
MODEND     62    94#
MSCL       37    42#   43
MSP       108   109   120#
MSREAD     51    65   107#  121
MSREP      32    45   127#
MSREQ      36
MSWR       44   102#
NEWMOD     46    49#   96
REQCDF     72
RP        119
RTNPAG     89
START      32#
TASKNA     24
TEMP      110   112   113   115   126#
TEST       24
VRCDF      76
WRITE      58    82    95    98#  104
WTMS       50    64
WTRP      101
WTTY       24
W1         57    59    70    87   114   124#
W2         52   117   125#
X          75    77    79
_00375     94
_00376     60    84
_00377     38

V3A
```

```
                  /APP. A0.   CONFIGURATION FILE.
                /PROCESSOR TYPE.
        0001    PDP8E=1
                /PDP8I=1
        0001    EAE=1

        0001    IFDEF EAE <HARDMQ=1>
        0001    IFDEF PDP8E <HARDMQ=1>
        7701    IFDEF  HARDMQ   <GETMQ=CLA MQA; STORMQ=MQL>
        7421
                IFNDEF HARDMQ   <GETMQ=TAD MQ; STORMQ=DCA MQ>

                /INDICATE ACTUAL CORESIZE (HIGHEST ACTUAL FIELD)
        0007    ACTMAX=7

                /INDICATE VIRTUAL CORESIZE (HIGHEST VIRTUAL FIELD)
        0077    VIRMAX=77                 /MUST BE <=77

                /INDICATE LENGTH OF STATIC TASK LIST (HIGHEST ST#)
                /THIS VALUE DETERMINES THE MAX NUMBER OF TASKS THAT
                /CAN RUN SIMULTANEOUSLY.
        0077    MAXSTL=77                 /MUST BE <=177

                /CONFIGURATE SKIPCHAIN.
                /INDICATE YOUR DEVICES IN THE ORDER OF THEIR
                /INTERRUPT PRIORITY.
                NOPUNCH          /IDEA OF MULTI 8.
        0000    *0
                INTDEF,
00000   0000    RK8E,    0
00001   0000    KM8E,    0
                /MULTIP,         0                /MULTIPLEXER CLOCK.
00002   0000    PDP8IE, 0;0      /TWO DEVICES !!!!!!!!!!!
00003   0000
                /PDP11, 0;0
                /RF08,   0
00004   0000    DK8EP,   0
00005   0000    TT1,     0
00006   0000    KB1,     0
00007   0000    TT2,     0
00010   0000    KB2,     0
00011   0000    TT3,     0
00012   0000    KB3,     0
                MAXDEV,
                ENPUNCH
        0370    IFDEF TT2 <HCT=6420-6030>
        0430    IFDEF TT3 <HP=6460-6030>

        6041    IFDEF TT1 <TSF1=TSF;TCF1=TCF;TPC1=TPC;TLS1=TLS>
        6042
        6044
        6046
        6031    IFDEF KB1 <KSF1=KSF;KCC1=KCC;KRS1=KRS;KRB1=KRB>
        6032
        6034
```

```
     6036

     6431   IFDEF TT2 <TSF2=TSF+HCT;TCF2=TCF+HCT;TPC2=TPC+HCT;TLS2=TLS+HCT>
     6432
     6434
     6436
     6421   IFDEF KB2 <KSF2=KSF+HCT;KCC2=KCC+HCT;KRS2=KRS+HCT;KRB2=KRB+HCT>
     6422
     6424
     6426

     6471   IFDEF TT3 <TSF3=TSF+HP;TCF3=TCF+HP;TPC3=TPC+HP;TLS3=TLS+HP>
     6472
     6474
     6476
     6461   IFDEF KB3 <KSF3=KSF+HP;KCC3=KCC+HP;KRS3=KRS+HP;KRB3=KRB+HP>
     6462
     6464
     6466

            /SYSTEM STATISTICS WANTED?
     0001   STATX=1
            /EXTRA ERROR CHECK WANTED?
     0001   CHECK=1

            IFDEF STATX <
            /SET VALUE FOR MEASURING TIME IN IDLELOOP.
            DECIMAL
     6331   IFDEF PDP8E < TCKLEN=3289        /ABOUT 100.000/30.4 FOR PDP8/E>
            IFDEF PDP8I < TCKLEN=2777        /ABOUT 100.000/36   FOR PDP8/I>
                                     /USED FOR IDLETIME STATISTICS.
            OCTAL
            >

            /SELECT SYSTEM DISK.
     0000   SYDISK=RK8E
            /SYDISK=RF08
            /SELECT SYSTEM CLOCK.
     0004   SYTIME=DK8EP
            /SYTIME=MULTIP

            /CONFIGURATE LOGICAL UNIT 0 OF SYSTEM DISK.
     4000   VFBLOK=4000     /START BLOCK OF VFLDSLOTS.
            IFNZRO VFBLOK^17 <ASS ERR! VFBLOK MUST BE MULTIPLE OF 20>
     6000   TSBLOK=VIRMAX+1↑20+VFBLOK       /STARTING BLOK OF TASK LIBRARY.

            /SOME STUFF FOR PROGRAMS USING THE TASKLIBRARY.
            /NOT REQUIRED FOR MC8!!!!!!!!!!!!
            /ASSUME DIRECTORY SWAPPED IN; TSBLOK+1 STARTS AT LOC200.
     0200   DCDIR=200
     0400   NRT=DCDIR+200
     5000   FDCMAX=5000              /FIRST FREE LOC AFTER DIRECTORIES.
```

```
              /APP. C. MC8 TASKLOADER TASK.

      0000   *0
00000 1437          1437    /PRIO=1, ZREQ, LENGTH=37.
00001 0000          0000    /NO AUTOSTART.
00002 0714          TASKNAME GLOBAL.TSLD
00003 1702
00004 0114
00005 2423
00006 1404

      0200   PAGE

             /R VAN VLIET, MATH' CENTR', A'DAM; 6/1/76.

             /TASK TO LOAD AND UNLOAD TASKS.

             /RUNNING A TASK IN THE MC8 SYSTEM GOES IN THE FOLLOWING STEPS:
             /A.     A TWO-WORD ENTRY IN THE STATIC TASK LIST IS REQUESTED AND
             /       ASSIGNED TO THE TASK, THE SOCALLED STATIC TASKNUMBER
             /       -STSK#- WHICH IDENTIFIES THAT ENTRY IN THE STATIC TASK
             /       LIST. IF P IS W0 OF THE ENTRY, STSK#=(P/2)^177.
             /       IN THE ENTRY SUFFICIENT INFO IS STORED TO START UP THE
             /       TASK (ITS BLOCKNUMBER ON DISK, LENGTH, PRIORITY ETC.).
             /B.     THE STATIC TASK NUMBER CAN BE USED TO ADDRESS A MS TO THE
             /       TASK. AS SOON AS SUCH A MS IS SENT, THE SYSTEM ALLOCATES
             /       A TASK CONTROLBLOCK FOR THE TASK AND ADDS IT TO THE RUNQ.
             /C.     WHEN THE RUNQ IS SCANNED THE TASK IS FOUND TO BE RUNNABLE
             /       BUT ITS CODE IS STILL ON DISK. THE SYSTEM REMOVES IT FROM
             /       THE RUNQ AND SENDS A MS TO THE MONITOR FETCH TASK MTFTCH.
             /D.     MTFTCH ALLOCATES SPACE FOR THE TASKS CODE, SWAPS IT IN
             /       CORE, DOES THE RELOCATION AND REINSERTS IT IN THE RUNQ.
             /E.     FINALLY THE RUNQ IS SCANNED, THE TASK IS FOUND TO BE
             /       RUNNABLE AND IN CORE, SO IT CAN BE STARTED.

             /THE QUESTION IS "HOW CAN TASKS COMMUNICATE?" I.E., "HOW CAN THEY
             /ADDRESS MESSAGES TO EACH OTHER?". THE STSK# OR THE TASKS TCB
             /CANNOT BE USED TO IDENTIFY A TSK, AS THEY ARE UNKNOWN AT
             /ASSEMBLYTIME. THE ONLY THING ABOUT OTHER TSKS THAT IS KNOWN AT
             /ASSEMBLYTIME IS  THEIR NAME. HOWEVER WE WOULD NOT LIKE THE
             /SYSTEM TO SEARCH DOWN A LONG LIST OF NAMES (PREFERABLY STORED ON
             /DISK) FOR EACH MESSAGE THAT IS SENT.
             /THE PROBLEM IS SOLVED BY INSERTING THE STATIC TSK NUMBER IN THE
             /CODE ON THE SPOT WHERE - AT ASSEMBLYTIME - THE NAME WAS WRITTEN.
             /THIS IS DONE BY THE TASK LOADER.
             /TO BE ABLE TO DO SO, THE TSK LOADER MUST KNOW AHEAD TO WHICH
             /TSKS  A TSK WILL REFER. THEREFORE TASKS ARE GROUPED IN FAMILIES
             /OF TASKS THAT REFER TO ONE ANOTHER.
             /
             /"LOADING" A FAMILY:
             /A.     FOR EACH MEMBER OF THE FAMILY AN ENTRY IN THE STATIC TASK
             /       LIST (STSK#) IS REQUESTED.
             /B.     ONE BY ONE EACH MEMBER IS SWAPPED IN, ITS ENTRY IN THE
             /       STATIC LIST IS FILLED WITH APPROPRIATE INFO, NAMES
             /       OF OTHER TASKS IN ITS CODE ARE REPLACED BY THE
/       CORRESPONDING STATIC TASK NUMBERS, ITS UPDATED CODE IS
/       RESTORED ON DISK
/"UNLOADING" A FAMILY OF TASKS:
/A,     STOP ALL MEMBERS (THEY SHOULD BE STOPPED, IF NOT, THIS
/       INDICATES SOME ERROR).
/B,     REMOVE THEIR CODE FROM CORE (IF NECESSARY).
/C,     CLEAR THEIR ENTRIES IN THE STATIC TASKLIST.

/AS LONG AS A FAMILY IS NOT YET LOADED, NO ONE CAN REFER TO ONE
/OF ITS TASKS. ONCE IT IS LOADED ALL ITS MEMBERS ARE AVAILABLE
/(THOUGH NOT NECESSARILY IN CORE).
/AFTER UNLOADING A FAMILY, ALL ITS MEMBERS ARE COMPLETELY
/DISMISSED FROM CORE, SO THAT NO SPURIOUS MESSAGES WILL ARRIVE AT
/OTHER TASKS THAT  LATER USE THE SAME STATIC TASK NUMBERS.


/TO FACILITATE UPDATING THE CODE A SOCALLED TASK INFO BLOCK (TIB)
/PRECEDES THE TASKS CODE ON DISK. THIS BLOCK HOLDS PTRS TO ALL
/PLACES (MAX 83) IN THE CODE WHERE A NAME MUST BE REPLACED BY A
/STSK#. WHEN THE TASK IS ASSEMBLED WITH MC8PAL TASKNAMES ARE
/INDICATED  BY USING THE GLTASK OR LCTASK PSEUDO OPS. THESE
/PSEUDO OPS RESERVE ONE LOC AT THE PLACE WHERE THEY ARE USED, AND
/OUTPUT SOME MISTIC BINARY CODE, SO THAT AN APPROPRIATE ENTRY IN
/THE TIB CAN BE BUILT WHEN THE TASK IS INSERTED IN THE TASKLIBRARY.
/LAYOUT OF TIB:
/W0:    B0-2 TASKS PRIORITY
/       B3   ZREQBIT (TASK NEEDS ONE TASK ONLY PART OF PAGE 0).
/       B5   TASK MUST BE LOADED IN CORERESIDENT FIELD
/       B7-11TASKS LENGTH IN PAGES.
/W1     B0      AUTOSTARTBIT. IF THIS BIT IS SET, MCTSLD SENDS A MS
/               TO THAT TASK TO START IT UP.
/W2-4   FAMILY NAME
/W5-6   TASK NAME
/W7-255 3-WORD REFERENCE ENTRIES.
/LAYOUT OF A REFERENCE ENTRY:
/W0-1   TASK NAME (TASKS NAME TO BE REPLACED)
/       W0B0=0: TASK OF GLOBAL FAMILY,
/       W0B0=1: TASK OF OWN FAMILY.
/W2     LOCATION WHERE THE STSK# MUST BE INSERTED.

/IN PRINCIPLE SOME TASKS CAN BE MEMBERS OF DIFFERENT FAMILIES.
/MOREOVER THERE IS ONE FAMILY (NAMED GLOBAL) WHOSE TASKS CAN BE
/ADDRESSED FROM ANY OTHER FAMILY. THE FAMILY GLOBAL IS LOADED
/WHEN THE MC8 SYSTEM IS INITIALISED AND IS NEVER UNLOADED. SO ITS
/MEMBERS ARE ALWAYS ADDRESSABLE.
```

```
          /ORGANISATION OF THE TASK LIBRARY.

          /TO MINIMIZE SWAPPING, THE COMPLETE DIRECTORY OF THE TASK LIBRARY
          /IS STORED IN THE CODE OF TSLD.
          /THIS CAUSES THE COMPLETE DIRECTORY TO BE IN CORE WHEN TSLD IS
          /RUN. EACH FAMILY HAS A FAMILY DECLARER (FDC). IT IS ALLOCATED IN
          /ONE PAGE.
          /LAYOUT OF A FDC:
          /W0     0       (TO KEEP THE RELOCATOR OFF).
          /W1     NUMBER OF TASKS IN THE FAMILY.
          /W2-3   UNUSED.
          /W4-127         4-WORD TASK ENTRIES.
          /LAYOUT OF TASK ENTRY IN FDC:
          /FREE (UNUSED) ENTRIES HAVE ALL WORDS 0.
          /USED ENTRIES:
          /W0-1   TASK NAME.
          /       TASK NAMES CONSIST OF UPTO 4 ALFANUMERIC SYMBOLS, THE
          /       FIRST OF WHICH MUST BE A LETTER. THEY ARE STORED IN 6-BIT
          /       TRIMMED ASCII, PADDED WITH ZEROES.
          /W2     B0      AUTOSTARTBIT.
          /               IF SET THE LOADER WILL SEND A MS TO START THAT
          /               TSK.
          /W2     B1-11   TASKS BLOCKNUMBER ON DISK.
          /       IF W2=0, THE TASK IS ALWAYS KNOWN BY THE SYSTEM, AND W3
          /       HOLDS THE STATIC TASK NUMBER.
          /W3     STSK# IF THE TASK IS LOADED, 0 OTHERWISE.
          /NOTE:  MAX 31 TASK ENTRIES CAN BE ALLOCATED IN A FDC, SO A
          /FAMILY HAS MAX 31 MEMBERS.

          /ALL FAMILY DECLARERS ARE DENOTED IN ONE DIRECTORY: THE DIRECTORY
          /OF FAMILY DECLARERS (DCDIR).
          /LAYOUT OF DCDIR:
          /W0     PTR TO START OF TSLD CODE.
          /W1     0       (TO KEEP OFF THE RELOCATOR)
          /W2     JMP I .-2. THIS INSTRUCTION WILL START UP TSLD.
          /W3     NUMBER OF PAGES AVAILABLE FOR FDCS.
          /W4-127 FDC ENTRIES.
          /LAYOUT OF FDC ENTRY IN DCDIR:
          /FREE (UNUSED) ENTRIES HAVE ALL WORDS 0.
          /USED ENTRIES:
          /W0-2   FAMILY NAME.
          /       FAMILY NAMES CONSIST OF UPTO 6 ALFANUMERIC SYMBOLS, THE
          /       FIRST OF WHICH IS A LETTER. THESE SYMBOLS ARE STORED IN
          /       6-BIT TRIMMED ASCII, PADDED WITH ZEROES.
          /W3     B0:     LOADBIT.
          /               1: FAMILY IS LOADED, 0: FAMILY IS UNLOADED.
          /       B1-5:   PAGENUMBER OF FAMILY DECLARER.
          /       B6-11:  MUST BE 0.

          /TASK STORAGE ON DISK.
          /
          /A TASK IS STORED IN A NUMBER OF CONSECUTIVE BLOCKS ON DISK
          /(LOGICAL UNIT 0). THE FIRST IS THE TIB.
          /REFERENCES TO A BLOCKNUMBER OF A TASK ARE ALWAYS GIVEN WITH AN
          /OFFSET OF TSBLOK (A SYSTEM CONSTANT).
```

```
        /BLOCKNUMBERS OF TASKS MUST BE POSITIVE I.E. MAX 2047.
        /THE FIRST BLOCK (TSBLOK) OF THE TASK STORAGE AREA IS USED AS A
        /BITMAP OF OCCUPIED BLOCKS. IF A BLOCK IS USED FOR TASK STORAGE
        /THE CORRESPONDING BIT IS SET. WE USE B4-11 OF EACH WORD.
        /SO W0B11 CORESSPONDS TO TSBLOK, B10 TO TSBLOK+1...
        /W1 B11 TO TSBLOK+10, ETC.
        /THE FIRST TASK MUST ALWAYS BE "GLOBAL.TSLD", AND IT MUST BE
        /ALLOCATED FROM TSBLOK UPWARD.
        /SO ITS TIB IS STORED ON TSBLOK+0 AND ITS CODE FROM TSBLOK+1
        /UPWARD. AS THE DIRECTORIES OF THE TASK LIBRARY ARE STORED AS
        /PART OF THE CODE OF "GLOBAL.TSLD" THEY MAY BE FOUND AS FOLLOWS:
        /BITMAP AT      TSBLOK+0. (NO ACTUAL TIB OF TSLD).
        /DCDIR  AT      TSBLOK+1, FIRST PAGE.
        /NRT    AT      TSBLOK+1, SECOND PAGE.
        /FDCS   AT      TSBLOK+2, SSQQ.


        /THE NAME REFERENCE TABLE (NRT).
        /
        /IT IS CONVENIENT, THOUGH NOT NECESSARY, TO KEEP A LIST OF TASKS
        /THAT ARE LOADED. THIS LIST IS CALLED THE NAME REFERENCE TABLE,
        /AND IT IS KEPT ON THE SECOND PAGE OF TSBLOK+1.
        /W0 OF THE NRT MUST 0, TO KEEP THE RELOACTOR OFF.
        /ALL OTHER WORDS CORRESPOND TO ENTRIES IN THE STL IN AN EVIDENT
        /WAY.
        /LAYOUT OF AN ENTRY IN THE NRT:
        /0      NO TASK IS LOADED IN THE CORRESPONDING ENTRY OF THE STL
        /       (WARNING: TSLD DID NOT USE THAT ENTRY, BUT PERHAPS OTHERS
        /       DID!)
        /B1-5   NUMBER OF FDC IN DCDIR.
        /B7-11  NUMBER OF TASK ENTRY IN FDC.
```

```
               /DIRECTORY OF FAMILY DECLARERS.

               /NOTE: USING PTRS IS OK, AS TSLD IS 37 PAGES, SO THAT
               /NO ACTUAL RELCCATION CAN OCCUR.

00200  5205    DCDIR,  START              /RELOCATED PTR TO START.
00201  0000            0                  /END OF RELOCATED CODE.
00202  5600            JMP I   DCDIR
00203  0021            FDCMAX-GLOBAL%200          /NUMBER OF PAGES AVAILABLE FOR
                                                  /FDCS.
00204  0714            0714;1702;0114;GLOBAL%2
00205  1702
00206  0114
00207  0300
00210  0000            ZBLOCK 200-.^177
```

```
               /NAME REFERENCE TABLE.
               /EACH ENTRY CORRESPONDS TO AN ENTRY IN THE STATIC TASK
               /LIST (NRT[1] TO STL[0], NRT[2] TO STL[1] ...).
               /NRT[0] MUST BE 0 TO KEEP THE RELOCATOR OFF.

               IFNZRO 176-MAXSTL6^4000 <ASS ERROR!>
               /THE NRT CLEARLY HAS ONLY 176 ENTRIES.

               /INITIALLY CLEARED.
00400  0000    NRT,     ZBLOCK 200
```

```
               /FAMILY DECLARERS.
               /EACH FAMILY DECLARER OCCUPIES 1 PAGE OF CODE.

               /INITIAL GLOBAL DECLARER.
               GLOBAL,
00600   0000            0;6;0;0
00601   0006
00602   0000
00603   0000
00604   1104            DEVICE IDLE;0;SIDLE
00605   1405
00606   0000
00607   0000
00610   2327            DEVICE SWAP;0;SMTSWAP
00611   0120
00612   0000
00613   0001
00614   0624            DEVICE FTCH;0;SMTFTCH
00615   0310
00616   0000
00617   0002
00620   2411            DEVICE TIME;0;SMTTIME
00621   1505
00622   0000
00623   0003
00624   2304            DEVICE SDSK;0;SMTSDSK
00625   2313
00626   0000
00627   0004
00630   2423            DEVICE TSLD;0;XTSLD, 0/SMTTSLD
00631   1404
00632   0000
00633   0000
00634   0000            ZBLOCK 200-.^177 /ALL OTHERS UNUSED.

               /WHEN BUILDING THE SYSTEM, HERE IS THE INITIAL BITMAP.
01000   0377   BITMAP, 377;377;1        /TSBLOK+16 BLOCKS OF TSLD.
01001   0377
01002   0001


               /ALL OTHER FDCS INITIALLY CLEARED.
01003   0000            ZBLOCK FDCMAX-.

        5000   PAGE                /FORCE RELOCATION TO NEXT PAGE=5000.
```

```
              /BUFFER FOR TASK INFO BLOK.
       5000   TIBBUF=.^7600

              /IN THIS FIRST PAGE OF TIBBUF GOES THE CODE TO BUILD
              /INITIAL BITMAP AND DIRECTORIES.
              /ONE FAMILY DECLARER FOR GLOBAL IS SET UP. ALL OTHERS
              /ARE ZERO.
              /THE WHOLE TSLD IS WRITTEN ON THE DISK FROM TSBLOK+1.

05000  0000           0                 /TO SATISFY FUTURE RELOCATION.

              /NOTE: THIS CODE IS ONLY EXECUTED WHEN BUILDING A
              /COMPLETELY FRESH TASK LIBRARY. THIS CODE IS NOT INTENDED
              /TO BE EXECUTED AS TASKCODE, SO IT IS NOT RELOCATED.
              MCTSLD,
              /FIRST THE INITIAL BITMAP MUST BE WRITTTEN ON DISK, ALLOCATED ON
              /LOGICAL UNIT 0 AT TSBLOK.
              /NEXT THIS COMPLETE FLD MINUS PAGE 0 MUST BE WRITTEN ON DISK.
              /IT MUST BE ALLOCATED ON THE FUTURE MC8 SYSTEM DISK
              /LOGICAL UNIT 0, FROM TSBLOK+1.
              /THE WAY TO ACCOMPLISH THIS DEPENDS ON THE SYSTEM WHICH
              /IS CURRENTLY RUNNING.
              /BELOW FOLLOWS A VERSION, ASSUMING THAT THE MC8 SYSTEM DISK
              /LU 0 BLOCK 0 CORRESPONDS TO THE OS/8 SYSTEMDEVICE
              /BLOCK 0.

       0020   CURFLD=20                 /WE ARE RUNNING IN FLD 2.
05001  6202           CIF 0
05002  4777           JMS I   (7607
05003  4220                   4200+CURFLD/WRITE TWO PAGES.
05004  1000                   BITMAP
05005  6000                   TSBLOK
05006  7402           HLT
05007  3776           DCA I   (BITMAP  /BITMAP WAS STORED IN SOME FDC, AND
05010  3775           DCA I   (BITMAP+1/HENCE MUST BE CLEREARED BE FORE THE
05011  3774           DCA I   (BITMAP+2/DIRECTORIES ARE WRITTEN OUT.


05012  6202           CIF 0
05013  4777           JMS I   (7607
05014  7720                   7700+CURFLD
05015  0200                   200
05016  6001                   TSBLOK+1
05017  7402           HLT
05020  6203           CIF CDF 0
05021  5773           JMP I   (7600

05173  7600
05174  1002
05175  1001
05176  1000
05177  7607
       5204   PAGE
```

```
                /INITIALISATION CODE IN SECOND PAGE OF TIB BUFFER.
                /CODE TO INITIALISE DIRECTORIES AND TO LOAD GLOBAL.

05204  0000               0                  /KEEP OFF RELOCATOR.

05205  3157     START,    DCA     MSP        /STORE PTR TO MS.

                /IF A MS IS RECEIVED TO LOAD GLOBAL, WE ASSUME THAT THE
                /SYSTEM MUST BE REINITIALISED. OTHER MSS ARE PASSED
                /TO MSGIN.
05206  1157               TAD     MSP
05207  3110               DCA     X
05210  6201               CDF 0
05211  1510               TAD I   X
05212  1377               TAD     (-0714     /GL
05213  7640               SZA CLA
05214  5600               JMP I   )MSGIN
05215  2110               ISZ     X
05216  1510               TAD I   X
05217  1376               TAD     (-1702     /OB
05220  7640               SZA CLA
05221  5600               JMP I   )MSGIN
05222  2110               ISZ     X
05223  1510               TAD I   X
05224  1375               TAD     (-0114     /AL
05225  7640               SZA CLA
05226  5600               JMP I   )MSGIN
05227  4051     TSLIN,    CDFCUR
05230  7307               AC4
05231  1201               TAD     )DCDIR
05232  3150               DCA     DCPTR      /PTR IN DECLARER DIRECTORY.
05233  1374               TAD     (-37
05234  3151               DCA     DCCTR      /AT MOST 37OCT DECLARERS.
                /CLEAR THE LOADBITS OF ALL FAMILIES.
05235  7325     DCINIT,   AC3
05236  1150               TAD     DCPTR
05237  3150               DCA     DCPTR      /SKIP OVER THE FAMILYNAME.
05240  7350               AC3777
05241  0550               AND I   DCPTR
05242  7450               SNA                /IS THIS ENTRY USED?
05243  5267               JMP     DCIN2      /NO. SKIP INIT OF FDC.
05244  3550               DCA I   DCPTR      /LOADBIT CLEARED.

05245  1550               TAD I   DCPTR
05246  7104               CLL RAL
05247  0146               AND     C7600
05250  1122               TAD     C4
05251  3152               DCA     FDCPTR
05252  1374               TAD     (-37
05253  3153               DCA     FDCCTR
                /CLEAR ALL STATIC TASK NUMBERS IN FAMILY DECLARERS.
                /NOTE: IF NO DISKBLOCK IS INDICATED FOR A GIVEN TASK,
                /THEN THAT TASK IS ALWAYS PRESENT IN THE SYSTEM,
                /AND SO ITS STSK# MUST NOT BE CLEARED.
05254  7326     FDINIT,   AC2
05255  1152          TAD     FDCPTR
05256  3152          DCA     FDCPTR  /SKIP OVER TASK NAME.
05257  7350          AC3777
05260  0552          AND I   FDCPTR  /GET BLOCK#; 0 INDICATES TSK ALWAYS IN
05261  2152          ISZ     FDCPTR  /SYSTEM.
05262  7640          SZA CLA
05263  3552          DCA I   FDCPTR  /CLEAR STSK#
05264  2152          ISZ     FDCPTR
05265  2153          ISZ     FDCCTR
05266  5254          JMP     FDINIT

05267  2150  DCIN2,  ISZ     DCPTR
05270  2151          ISZ     DCCTR   /ALL DECLARERS INITIALISED?
05271  5235          JMP     DCINIT


             /CLEAR THE NAME REFERENCE TABLE.
05272  1202          TAD     )NRT
05273  3150          DCA     DCPTR    /JUST A PTR.
05274  1373          TAD     (-MAXSTL-1
05275  3151          DCA     DCCTR    /LENGTH OF STATIC TASK LIST +1.
05276  3550          DCA I   DCPTR
05277  2150          ISZ     DCPTR
05300  2151          ISZ     DCCTR
05301  5276          JMP     .-3

             /WE STILL DONT KNOW OUR OWN STSK#.
             /IT WILL BE PASSED IN THE NEXT MS.
05302  4054          CALL;   WTMS+KEEP
05303  1014
05304  4014          MSFREE
05305  3603          DCA I   )XTSLD


             /ALL INITIALISATION DONE: SIMULATE A NORMAL CALL.
05306  5600          JMP I   )MSGIN

05200  5404
05201  0200
05202  0400
05203  0633
05373  7700
05374  7741
05375  7664
05376  6076
05377  7064
       5403  PAGE
```

```
05403  0000          0

               /A NORMAL MS WAS RECEIVED TO LOAD OR UNLOAD A FAMILY
               /OF TASKS.
05404  6201    MSGIN,  CDF 0
05405  1157            TAD     MSP      /STORE MSPTR FOR MSREPORT
05406  3301            DCA     MSPREP
05407  3161            DCA     SWFLD    /NO FLD CLAIMED YET.
05410  3162            DCA     SWMSP    /NO MS CLAIMED YET.
05411  1157            TAD     MSP
05412  3110            DCA     X
05413  7350            AC3777
05414  0510            AND I   X        /GET FAMILY NAME OUT OF MS.
05415  7041            CIA
05416  3154            DCA     FNAME
05417  2110            ISZ     X
05420  1510            TAD I   X
05421  7041            CIA
05422  3155            DCA     FNAME+1
05423  2110            ISZ     X
05424  1510            TAD I   X
05425  7041            CIA
05426  3156            DCA     FNAME+2
05427  4051            CDFCUR
05430  7307            AC4
05431  1200            TAD     )DCDIR
05432  3150            DCA     DCPTR    /SEARCH DCDIR FOR A DECLARER OF THIS
05433  1377            TAD     (-37     /NAME.
05434  3151            DCA     DCCTR

05435  1550    SDCLP,  TAD I   DCPTR
05436  2150            ISZ     DCPTR
05437  7450            SNA              /ANYTHING IN THIS ENTRY?
05440  5257            JMP     SDCSKP   /NO.
05441  1154            TAD     FNAME
05442  7640            SZA CLA
05443  5257            JMP     SDCSKP
05444  1550            TAD I   DCPTR
05445  2150            ISZ     DCPTR
05446  1155            TAD     FNAME+1
05447  7640            SZA CLA
05450  5260            JMP     SDCSKP+1
05451  1550            TAD I   DCPTR
05452  2150            ISZ     DCPTR
05453  1156            TAD     FNAME+2
05454  7640            SZA CLA
05455  5261            JMP     SDCSKP+2
05456  5304            JMP     SDCFND
05457  2150    SDCSKP, ISZ     DCPTR
05460  2150            ISZ     DCPTR
05461  2150            ISZ     DCPTR
05462  2151            ISZ     DCCTR
05463  5235            JMP     SDCLP
05464  7201    ER1,    CLA IAC          /SIGNAL ERROR CONDITION 1.
                                        /INCORRECT FAMILY NAME.
```

```
                /END OF THIS RUN.
05465  6201     DONE,   CDF 0
05466  3557             DCA I   MSP     /COMPLETION OR ERROR CONDITION IN MS.
05467  1162             TAD     SWMSP   /ANY MS CLAIMED?
05470  7440             SZA
05471  4014             MSFREE  /YES, FREE IT.
05472  7200             CLA
05473  1161             TAD     SWFLD   /ANY FLD CLAIMED
05474  7440             SZA
05475  4054             CALL;   RTNFLD  /YES RETURN IT, OR: INNOCENT AND.
05476  0036
05477  4054             CALL;   RP
05500  0013
05501  0000     MSPREP,         0
05502  4054             CALL;   EXIT
05503  0054

                /WE FOUND THE SPECIFIED FAMILY DECLARER.
                /DCPTR PTS AT ITS ENTRY W3.
05504  1550     SDCFND, TAD I   DCPTR   /SET FSTRT TO FDC+4 (FIRST 4 LOCS
05505  7104             CLL RAL         /UNUSED).
05506  0146             AND     C7600
05507  1122             TAD     C4
05510  3160             DCA     FSTRT
05511  6201             CDF 0
05512  1557             TAD I   MSP
05513  4051             CDFCUR
05514  7700             SMA CLA         /LOAD OR UNLOAD FAMILY?
05515  5601             JMP I   )LOAD
05516  5602             JMP I   )UNLOAD


                /INITIALISE FOR SEARCHING ONE FDC.
                /FSTRT=FDC+4.
05517  0000     FDCSET, 0
05520  1160             TAD     FSTRT   /PTRS.
05521  3152             DCA     FDCPTR
05522  1377             TAD     (-37    /MAX 37OCT TSKS IN FDC.
05523  3153             DCA     FDCCTR
05524  5717             JMP I   FDCSET
05400  0200
05401  5611
05402  6307
05577  7741
       5611     PAGE
```

```
                /LOAD A FAMILY OF TASKS.
                /MINUS ITS NAME IS DENOTED IN FNAME,FNAME+1,FNAME+2.
                /DCPTR PTS AT ITS ENTRY IN DCDIR W3.

05611  1550     LOAD,   TAD I   DCPTR   /GET ENTRY[3]; NEGATIVE MEANS ALREADY
05612  7500             SMA             /LOADED.
05613  5216             JMP     .+3
05614  7326     ER2,    AC2             /ERROR CONDITION 2: ALREADY LOADED.
05615  5600             JMP I   )DONE
05616  4054             CALL;   REQFLD  /REQUEST FLD TO STORE TASKS CODE.
05617  0034
05620  4054             CALL;   REQCDF  /SET UP FOR MINIMUMTIME CDF ROUTINE.
05621  0052
05622  3161             DCA     SWFLD
05623  4601             JMS I   )FDCSET /SET UP FOR SEARCHING ONE FDC.

                /LOOP TO REQUEST ENTRIES IN STL.
05624  7326     LD1LP,  AC2
05625  1152             TAD     FDCPTR
05626  3152             DCA     FDCPTR  /SKIP OVER TASK NAME
05627  7350             AC3777
05630  0552             AND I   FDCPTR  /FETCH PTR TO TIB OF TASK
05631  2152             ISZ     FDCPTR
05632  7450             SNA             /0 INDICATES TASK ALREADY KNOWN
                                        /IN SYSTEM OR FREE ENTRY.
05633  5243             JMP     LD1SKP
05634  7001             IAC             /PTR TO TIB+1 (FIRST CODE BLOCK)
05635  3240             DCA     RQARG   /IN ARG1.
05636  4054             CALL;   REQSTL
05637  0041
05640  0000     RQARG,          0;0
05641  0000
05642  3552             DCA I   FDCPTR  /STORE STSK# RETURNED IN AC IN FDC.
05643  2152     LD1SKP, ISZ     FDCPTR
05644  2153             ISZ     FDCCTR
05645  5224             JMP     LD1LP

05646  4601             JMS I   )FDCSET /AGAIN LOOKING AT ALL TASKS.
05647  4014             MSREQ           /REQUEST MS FOR SWAPPING.
05650  3162             DCA     SWMSP

                /LOOP TO UPDATE THE CODE OF EACH TASK AND TO SET STL[TASK,1].
05651  7326     LD2LP,  AC2
05652  1152             TAD     FDCPTR
05653  3152             DCA     FDCPTR  /SKIP OVER TSKNAME.
05654  7350             AC3777
05655  0552             AND I   FDCPTR  /GET BLOKNUMBER
05656  2152             ISZ     FDCPTR
05657  7450             SNA             /0 INDICATES TASK ALREADY IN SYSTEM
                                        /OR FREE ENTRY.
05660  5602             JMP I   )LD2SKP
05661  1377             TAD     (TSBLOK
05662  3163             DCA     TBLOK   /BLOCKNUMBER OF TASK.
05663  1162             TAD     SWMSP
```

```
05664  3303           DCA      MSP0
05665  1162           TAD      SWMSP    /SEND MS TO FETCH TIB.
05666  3110           DCA      X        /PTR IN MS.
05667  6201           CDF 0
05670  1133           TAD      C200     /READ TWO PAGES FROM LOGICAL UNIT 0.
05671  3510           DCA I    X
05672  2110           ISZ      X
05673  1203           TAD      )TIBBUF
05674  3510           DCA I    X
05675  2110           ISZ      X
05676  1163           TAD      TBLOK    /BLOCKNUMBER
05677  3510           DCA I    X
05700  4051           CDFCUR
05701  4054           CALL;    SNDWTR+DFPARM
05702  0107
05703  0000  MSP0,             0
05704  0004                    SMTSDSK
05705  3334           DCA      MSP1     /SHIFT THIS USEFUL PTR AHEAD.
05706  6201           CDF 0             /CHECK REPORTED MS FOR I/O ERROR
05707  1703           TAD I    MSP0
05710  4051           CDFCUR
05711  7640           SZA CLA           /NONZERO INDICATES ERROR.
05712  5604           JMP I    )LDIOER
             /SEND MS TO SWAP TASKS CODE IN.
05713  1603           TAD I    )TIBBUF /GET LENGTH
05714  0125           AND      C37
05715  7106           CLL RTL;RTL;RTL
05716  7006
05717  7006
05720  6201           CDF 0
05721  3703           DCA I    MSP0     /NUMBER OF PAGES FROM UNIT 0.
05722  2303           ISZ      MSP0
05723  1133           TAD      C200
05724  3703           DCA I    MSP0     /CODE BUFFER STARTS AT 200.
05725  2303           ISZ      MSP0
05726  2163           ISZ      TBLOK    /PTR TO FIRST CODE BLOCK.
05727  1163           TAD      TBLOK    /FIRST CODE BLOCK.
05730  3703           DCA I    MSP0     /BLOCKNUMBER OF TASKS CODE.
05731  4020           VRCDF             /DF TO TASK CODE BUFFER FIELD
05732  4054           CALL;    SNDMS+DFPARM
05733  0103
05734  0000  MSP1,             0
05735  0004                    SMTSDSK /SWAP TASK CODE IN.
05736  4051           CDFCUR
05737  1334           TAD      MSP1
05740  3605           DCA I    )MSP2    /SHIFT PTR AHEAD.
05741  1552           TAD I    FDCPTR   /UPDATE STATIC TASK LIST.
05742  0132           AND      C177
05743  7104           CLL RAL
05744  1376           TAD      (STLST
05745  3010           DCA      X0       /PTR TO STL[TASK,1]
05746  1603           TAD I    )TIBBUF /LENGTH+CRES+ZREQ+PRIO
05747  6201           CDF 0
05750  3410           DCA I    X0
05751  4051           CDFCUR
```

```
                /REPLACE THE TASKNAMES IN THE 3-WORD ENTRIES IN THE TIB BY THE
                /CORRESPONDING STATIC TASKNUMBERS.
05752  1206             TAD     )TIBBUF+7
05753  3164             DCA     TIBPTR
05754  1375             TAD     (-123   /NUMBER OF ENTRIES.
05755  3151             DCA     DCCTR   /JUST A CTR HANGING AROUND
05756  4607             JMS I   )TNSET  /REPLACE.
05757  5362             JMP     .+3     /RTN1: ALL ENTRIES DONE
05760  2151             ISZ     DCCTR   /MORE ENTRIES TO DO?
05761  5356             JMP     .-3     /YES.

                /ALL ENTRIES IN TIB DONE.
                /PREPARE FOR UPDATING CODE.
05762  1206             TAD     )TIBBUF+7
05763  3164             DCA     TIBPTR
05764  1375             TAD     (-123   /NUMBER OF ENTRIES IN TIB
05765  3151             DCA     DCCTR
05766  5610             JMP I   )LD2LP2

05600  5465
05601  5517
05602  6057
05603  5000
05604  6305
05605  6013
05606  5007
05607  6204
05610  6011
05775  7655
05776  4400
05777  6000
       6010     PAGE
```

```
06010  0000  FDCNR,  0

06011  4054  LD2LP2, CALL;    WTRP      /WAIT FOR CODE SWAP IN COMPLETED.
06012  0005
06013  0000  MSP2,            0
06014  3247          DCA      MSP3      /KEEP SHIFTING THIS PTR AHEAD.
06015  6201          CDF 0
06016  1613          TAD I    MSP2      /I/O ERROR?
06017  4051          CDFCUR
06020  7640          SZA CLA
06021  5600          JMP I    )LDIOER

             /UPDATE TASKS CODE.
06022  4601          JMS I    )TSKUPD /UPDATE LOC INDICATED BY TIBENTRY.
06023  5226          JMP      .+3       /RTN1: ALL ENTRIES DONE
06024  2151          ISZ      DCCTR     /RTN2: PERHAPS MORE TO DO
06025  5222          JMP      .-3

             /CODE UPDATED. SEND MS TO SWAP TASK OUT AGAIN.
06026  1602          TAD I    )TIBBUF /GET LENGTH
06027  0125          AND      C37
06030  7106          CLL RTL;RTL;RTL
06031  7006
06032  7006
06033  1377          TAD      (4000
06034  6201          CDF 0
06035  3613          DCA I    MSP2      /WRITE N PAGES ON LOGICAL UNIT 0.
06036  2213          ISZ      MSP2
06037  1133          TAD      C200      /TASK CODE BUFFER STARTS AT 200
06040  3613          DCA I    MSP2      /CORE ADDRESS
06041  2213          ISZ      MSP2
06042  1163          TAD      TBLOK     /BLOCKNUMBER
06043  3613          DCA I    MSP2
06044  4020          VRCDF              /DF TO TASK CODE BUFFER FIELD
06045  4054          CALL;    SNDWTR+DFPARM
06046  0107
06047  0000  MSP3,            0
06050  0004                   SMTSDSK
06051  3360          DCA      MSP4      /STILL SHIFTING AHEAD.
06052  6201          CDF 0
06053  1647          TAD I    MSP3
06054  4051          CDFCUR
06055  7640          SZA CLA
06056  5600          JMP I    )LDIOER

06057  2152  LD2SKP, ISZ      FDCPTR    /NEXT TASK
06060  2153          ISZ      FDCCTR
06061  5603          JMP I    )LD2LP

             /ACTUAL LOADING COMPLETED. SET LOADED BIT.
06062  7350          AC3777
06063  0550          AND I    DCPTR
06064  1377          TAD      (4000     /SET LOADEDBIT
06065  3550          DCA I    DCPTR
```

```
                /UPDATE NAME REFERENCE TABLE.
                /EACH ENTRY IN THE NRT CORRESPONDS TO ONE ENTRY IN THE STATIC TASK
                /LIST. B1-5 CONTAIN THE FAMILY DECLARER NUMBER, B7-11 CONTAIN THE
06066  7346             ACM3            /TASKNUMBER. COMPUTE FDCNUMBER.
06067  1150             TAD     DCPTR
06070  0132             AND     C177    /FDCNUMBER*4
06071  7106             CLL RTL;RTL
06072  7006
06073  3210             DCA     FDCNR   /FDCNUMBER IN APPROPRIATE BITS.
06074  4604             JMS I   )FDCSET /INITIALISE FOR READING
                                        /OUT ALL STATIC TASKNUMBERS.

                /LOOP TO UPDATE NRT AND START UP TASKS IF THE AUTOSTARTBIT IS
                /SET.

06075  1552     LD3LP,  TAD I   FDCPTR  /0 INDICATES EMPTY ENTRY
06076  2152             ISZ     FDCPTR
06077  2152             ISZ     FDCPTR
06100  7650             SNA CLA
06101  5334             JMP     LD3SKP
06102  1552             TAD I   FDCPTR
06103  2152             ISZ     FDCPTR
06104  7700             SMA CLA         /AUTOSTARTBIT?
06105  5321             JMP     LD3LP2  /NO
06106  4014             MSREQ           /YES; SEND MS TO THIS TASK.
06107  3317             DCA     LD3MS
06110  1552             TAD I   FDCPTR
06111  3320             DCA     LD3MS+1
06112  6201             CDF 0
06113  3717             DCA I   LD3MS
06114  4051             CDFCUR
06115  4054             CALL;   SNDMS+NONREP
06116  2003
06117  0000     LD3MS,          0;0
06120  0000
06121  1552     LD3LP2, TAD I   FDCPTR  /GET STSK#
06122  0132             AND     C177
06123  1205             TAD     )NRT+1  /ADD NRT OFFSET
06124  3150             DCA     DCPTR   /PTR HANGING AROUND
06125  7346             ACM3            /COMPUTE TASKNUMBER
06126  1152             TAD     FDCPTR
06127  0132             AND     C177    /TSKNUMBER *4
06130  7112             CLL RTR
06131  1210             TAD     FDCNR   /ADD DECLARERNUMBER
06132  3550             DCA I   DCPTR   /AND STORE IN NRT.
06133  7410             SKP
06134  2152     LD3SKP, ISZ     FDCPTR
06135  2152             ISZ     FDCPTR
06136  2153             ISZ     FDCCTR
06137  5275             JMP     LD3LP

                /EVERYTHING IS OK NOW.
                /REWRITE TASKDIRECTORIES ON DISK.
06140  1162             TAD     SWMSP   /SET UP PTRS TO SWAPMESSAGE.
06141  3360             DCA     MSP4
```

```
06142  1162            TAD     SWMSP   /IT IS NOT SURE THAT THEY ARE SET
06143  3110            DCA     X       /CORRECT.
06144  6201            CDF 0
       2400  DLENG=FDCMAX-DCDIR%2+100^3600
06145  1376            TAD     (DLENG+4000 /WRITE OUT DLENG PAGES.
06146  3510            DCA I   X
06147  2110            ISZ     X
06150  1206            TAD     )DCDIR
06151  3510            DCA I   X       /CORE ADDRESS.
06152  2110            ISZ     X
06153  1375            TAD     (TSBLOK+1/STARTING BLOCK OF DIRECTORIES
06154  3510            DCA I   X
06155  4051            CDFCUR
06156  4054            CALL;   SNDWTR+DFPARM
06157  0107
06160  0000  MSP4,             0
06161  0004                    SMTSDSK
06162  4014            MSFREE  /FREE MS AND CHECK I/O STATUS
06163  3210            DCA     FDCNR   /JUST A TEMP.
06164  3162            DCA     SWMSP   /WE FREED THE MS.
06165  1210            TAD     FDCNR
06166  7440            SZA
06167  7307            AC4             /ERROR 4: DIRECTORY WRITE OUT ERROR.
06170  5607            JMP I   )DONE

06000  6305
06001  6264
06002  5000
06003  5651
06004  5517
06005  0401
06006  0200
06007  5465
06175  6001
06176  6400
06177  4000
       6201  PAGE
```

```
06201  0604  GLSTRT, GLOBAL+4
06202  0000  TNPTR,  0
06203  0000  TNCTR,  0

             /SOME USEFUL ROUTINES FOR THE LOADER.


             /REPLACE TASKNAME PTD AT BY TIBPTR BY THE CORRESPONDING STSK#.
             /NOTE: THE GLOBAL DECLARER STARTS AT GLOBAL, THE START OF THE
             /FAMILY DECLARER IS COMPUTED USING DCPTR.
06204  0000  TNSET,  0
06205  1564          TAD I   TIBPTR  /MORE ENTRIES TO DO?
06206  7450          SNA
06207  5604          JMP I   TNSET   /NO.
06210  7700          SMA CLA
06211  1160          TAD     FSTRT   /OWN FAMILY
06212  7450          SNA
06213  1201          TAD     GLSTRT  /GLOBAL FAMILY
06214  3202          DCA     TNPTR
06215  1377          TAD     (-37
06216  3203          DCA     TNCTR
06217  7350          AC3777
06220  0564          AND I   TIBPTR
06221  2164          ISZ     TIBPTR
06222  7041          CIA
06223  3310          DCA     TNAME
06224  1564          TAD I   TIBPTR
06225  7041          CIA
06226  3311          DCA     TNAME+1
06227  1602  TNSLP,  TAD I   TNPTR   /GET NAME FROM DECLARER
06230  2202          ISZ     TNPTR
06231  7450          SNA
06232  5242          JMP     TNSKP
06233  1310          TAD     TNAME
06234  7640          SZA CLA
06235  5242          JMP     TNSKP
06236  1602          TAD I   TNPTR   /SECOND WORD OF TASKNAME
06237  1311          TAD     TNAME+1
06240  7650          SNA CLA
06241  5254          JMP     TNFND
06242  2202  TNSKP,  ISZ     TNPTR   /NOT THIS TASK OF DECLARER, KEEP
06243  2202          ISZ     TNPTR   /SEARCHING.
06244  2202          ISZ     TNPTR
06245  2203          ISZ     TNCTR   /DECLARER EXHAUSTED?
06246  5227          JMP     TNSLP
06247  4054  ER3,    CALL
06250  0005                  WTRP
06251  0000                  0       /WAIT FOR REPORT OF SWPMS
06252  7325          AC3             /WRONG TIB.
06253  5600          JMP I   )DONE

06254  2202  TNFND,  ISZ     TNPTR   /WE FOUND THE CORRECT TASK.
06255  2202          ISZ     TNPTR   /PTR TO ITS STSK#
06256  1602          TAD I   TNPTR   /GET IT.
06257  3564          DCA I   TIBPTR  /STORE IN TIB. W1 OF THE ENTRY.
```

```
06260  2164            ISZ     TIBPTR  /UPDATE PTR FOR NEXT ENTRY
06261  2164            ISZ     TIBPTR
06262  2204            ISZ     TNSET   /TAKE SECOND RETURN.
06263  5604            JMP I   TNSET

               /UPDATE TASK CODE.
               /TIBPTR PTS AT A 3-WORD ENTRY IN TIB.
               /W1 HOLDS THE STSK#
               /W2 PTS AT THE LOC WHERE IT MUST BE STORED.
06264  0000    TSKUPD, 0
06265  1564            TAD I   TIBPTR  /MORE ENTRIES TO DO?
06266  7650            SNA CLA
06267  5664            JMP I   TSKUPD  /NO, TAKE RETURN1.
06270  2264            ISZ     TSKUPD  /YES, TAKE RETURN 2,
06271  2164            ISZ     TIBPTR
06272  1564            TAD I   TIBPTR  /GET STSK#
06273  3203            DCA     TNCTR   /JUST A TEMP,
06274  2164            ISZ     TIBPTR  /PTR TO W2
06275  1564            TAD I   TIBPTR  /GET PTR IN TASKS CODE,
06276  2164            ISZ     TIBPTR
06277  3202            DCA     TNPTR   /STORE IN OWN PAGE.
06300  1203            TAD     TNCTR   /REFETCH STSK#
06301  4020            VRCDF
06302  3602            DCA I   TNPTR
06303  4051            CDFCUR
06304  5664            JMP I   TSKUPD


               /I/O ERROR WHILE LOADING,
06305  7330    LDIOER, AC4000
06306  5600            JMP I   )DONE

06307  4023    UNLOAD, ERHLT

06310  0000    TNAME,  0;0
06311  0000

06200  5465
06377  7741
       6400    PAGE

06400  0000    ZBLOCK 7777-.   /FORCE USING ALL PAGES,
                /ZREQ AREA.
                NOPUNCH
        0150    *150
00150   0000    DCPTR,  0
00151   0000    DCCTR,  0
00152   0000    FDCPTR, 0
00153   0000    FDCCTR, 0
00154   0000    FNAME,  0;0;0
00155   0000
00156   0000
00157   0000    MSP,    0
00160   0000    FSTRT,  0         /FAMILY DECLARER +4.
00161   0000    SWFLD,  0
00162   0000    SWMSP,  0
00163   0000    TBLOK,  0
00164   0000    TIBPTR, 0

                ENPUNCH
```

```
$
$
```

```
BITMAP 1000       TNSKP  6242
CURFLD 0020       TNSLP  6227
DCCTR  0151       TSKUPD 6264
DCINIT 5235       TSLIN  5227
DCIN2  5267       UNLOAD 6307
DCPTR  0150       XTSLD  0633
DLENG  2400
DONE   5465
ER1    5464
ER2    5614
ER3    6247
FDCCTR 0153
FDCNR  6010
FDCPTR 0152
FDCSET 5517
FDINIT 5254
FNAME  0154
FSTRT  0160
GLOBAL 0600
GLSTRT 6201
LDIOER 6305
LD1LP  5624
LD1SKP 5643
LD2LP  5651
LD2LP2 6011
LD2SKP 6057
LD3LP  6075
LD3LP2 6121
LD3MS  6117
LD3SKP 6134
LOAD   5611
MCTSLD 5001
MSGIN  5404
MSP    0157
MSPREP 5501
MSP0   5703
MSP1   5734
MSP2   6013
MSP3   6047
MSP4   6160
RQARG  5640
SDCFND 5504
SDCLP  5435
SDCSKP 5457
START  5205
SWFLD  0161
SWMSP  0162
TBLOK  0163
TIBBUF 5000
TIBPTR 0164
TNAME  6310
TNCTR  6203
TNFND  6254
TNPTR  6202
TNSET  6204
```

```
ERRORS DETECTED! 0
LINKS GENERATED! 0
```

APPENDIX D.
THE SYSTEM BULLETIN PRESENTED HEREAFTER WAS PREPARED AFTER RUNNING THE TASKS RTTY AND WTTY FOR A WHILE.
NOTE: GLOBAL.LDTS IS A TASK THAT WAS RUN TO LOAD THE TASKS TEST.RTTY AND TEST.WTTY.

```
                          MC8   SYSTEM BULLETIN.
DATE:    14-FEB-77
RUN NUMBER:     3


RUNNED DURING:      0 HRS    6 MIN 13.2 SEC

SYSTEM VARIABLES.
CURTSK:           0000
SCDREQ,SCDINH:              0001 0000
INTAC,INTFL,INTPC:  0000 0000 1447
MONAC,MONFL,MONPC:  0000 0000 2352
MONFUNC,MARG1,MARG2:  1014 4273 0010

HPRIO=PRIORITY  0010
RUN QUEUES.
0000     0000
0001     0000
0002     0000
0003     0000
0004     0000
0005     0000
0006     0000
0007     0000
0010     5113     0000


SKIP CHAIN.
0207     6741 5214 6203 4613 2333
0214     6254 5221 6204 4515 0000
0221     6571 5226 5226 0000 0000
0226     6573 5233 5233 0000 0000
0233     6131 5240 6135 4515 2666
0240     6041 5245 6042 4515 0306
0245     6031 5252 6036 4515 0313
0252     6431 5257 6432 4515 0000
0257     6421 5264 6426 4515 0000
0264     6471 5271 6472 4515 0000
0271     6461 5276 6466 4515 0000


         CORMAP
0000     0100
0001     0077
0002     0001
0003     0002
0004     0003
0005     0004
0006     0005
0007     0006

TIME OUT QUEUE:
NODE     NEXT TASK VAL
```

```
FIELD TABLE:
0000     6707
0001     6224
0002     6230
0003     6247
0004     4250
0005     4260
0006     4270
0007     0000
0010     0000
0011     0000
0012     0000
0013     0000
0014     0000
0015     0000
0016     0000
0017     0000
0020     0000
0021     0000
0022     0000
0023     0000
0024     0000
0025     0000
0026     0000
0027     0000
0030     0000
0031     0000
0032     0000
0033     0000
0034     0000
0035     0000
0036     0000
0037     0000
0040     0000
0041     0000
0042     0000
0043     0000
0044     0000
0045     0000
0046     0000
0047     0000
0050     0000
0051     0000
0052     0000
0053     0000
0054     0000
0055     0000
0056     0000
0057     0000
0060     0000
0061     0000
0062     0000
0063     0000
0064     0000
0065     0000
0066     0000
0067     0000
0070     0000
0071     0000
```

```
0072      0000
0073      0000
0074      0000
0075      0000
0076      0000
0077      6210
```

```
AVAIL CHAINS.
FREE CORE CHAIN;
 [0316  0000 0305 -0033


FREE NODES IN AVAIL CHAINS:
AVAIL2     0
AVAIL3     1
AVAIL5    82
AVAIL20    1
```

```
STATIC TASK LIST AND TASKS

STL[   0]: GLOBAL.IDLE
        5113 7777
TCB[6:11]:    0000 0000 0000 2352 0000 0000
WAITWORD: 0020  LINKWORD: 0000

STL[   1]: GLOBAL.SWAP
        5035 7777
TCB[6:11]:    0001 0000 0000 2042 0000 0000
MESSAGES IN REPORT QUEUE. WAITED RP:-2037
WAITWORD: 4000  LINKWORD: 5035

STL[   2]: GLOBAL.FTCH
        5051 7777
TCB[6:11]:    0002 0000 0000 2402 0000 0400
TCB[12:15]:  5167 5173 0000 0000
MESSAGES IN RECEIVE QUEUE. CLAIM WORD:  0000
MESSAGES IN REPORT QUEUE. WAITED RP:-
WAITWORD: 2002  LINKWORD: 5051

STL[   3]: GLOBAL.TIME
        5067 7777
TCB[6:11]:    0003 0000 2666 2600 0000 0000
MESSAGES IN REPORT QUEUE. WAITED RP:-2666
WAITWORD: 4000  LINKWORD: 5067

STL[   4]: GLOBAL.SDSK
        5103 7777
TCB[6:11]:    0004 0000 0000 2103 0000 0000
MESSAGES IN RECEIVE QUEUE. CLAIM WORD:  0000
MESSAGES IN REPORT QUEUE. WAITED RP:-
WAITWORD: 2000  LINKWORD: 5103

STL[   5]: TEST  .RTTY
        5147 4002
TCB[6:11]:    0005 0303 0000 0604 0000 0000
TCB[12:15]:  0240 0201 0403 0024
MESSAGES IN RECEIVE QUEUE. CLAIM WORD:  5127
MESSAGES IN REPORT QUEUE. WAITED RP:-0313
WAITWORD: 4010  LINKWORD: 5147

STL[   6]: GLOBAL.TSLD
        0001 1437

STL[   7]: GLOBAL.LDTS
        0022 2001

STL[   8]: TEST  .WTTY
        5167 4001
TCB[6:11]:    0010 0303 0000 1032 0000 0000
TCB[12:15]:  0000 1327 1003 0026
MESSAGES IN RECEIVE QUEUE. CLAIM WORD:  5147
MESSAGES IN REPORT QUEUE. WAITED RP:-
WAITWORD: 2010  LINKWORD: 5167
UNUSED ENTRIES IN STL:     55
```

```
SYSTEM STATISTICS.

INTERRUPTS:          5770
SCHEDULES INTERRUPTING ACTIVE TASKS:            3
SCHEDULES AT MONITOR EXIT:          2163
MAX LENGTH OF RECEIVE/REPORT QUEUE:    81
SD NORMAL SWAPS:             16
SD VFIELD SWAPS:              0
SWAP ERRORS:     0
FIELDS USED:     4

                    MONITOR CALLS
STALL                     0
SEND MS                 587
WAIT REPORT            1798
SND MS AND WAIT          40
ERROR                     0
REPORT MS               624
WAIT FOR MS             628
DISABL DEVINTR            0
SBR TO INTR               0
CLAIM INTR                2
FREE INTR                 0
SIMULATE INTR            16
REQ PAGES                27
RTN PAGES                26
REQ FIELD                 2
RTN FIELD                 2
REQ ENTR IN STL           4
REQ TSK CONTRBL           0
RTN TSK CONTRBL           0
LOCK DF IN CORE           0
UNLOCK DF                 0
REQ SPECIAL CDF          55
EXIT                      4
STOP                      0
RESUME                    0
ERROR                     0
FILL TCB                  0
ERROR                     0
ERROR                     0
ERROR                     0
ERROR                     0
ERROR                     0

NODES USED IN AVAIL CHAINS.
AVAIL2     0
AVAIL3     2
AVAIL5    84
AVAIL20    3
LOST SPACE:     3
TOTAL AMOUNT OF SPACE USED OR LOST:    477

RUNNED DURING:      0 HRS     6 MIN 13.2 SEC
IDLE TIME:      0 HRS     6 MIN 06.2 SEC
IDLE DURING FIELD SWAP:      0 HRS     0 MIN 00.0 SEC
```

## APPENDIX E.

## LIST OF MONITOR COMMANDS.

CALLING SEQUENCE IN TASK:

```
        CALL
                MONFUNC /COMMANDWORD
                ...     /POSSIBLE ARGUMENTS.
        ...             /NORMAL RETURN.
```

THERE ARE ABOUT 25 MONITOR COMMANDS DISTINGUISHED BY THE COMMANDWORD. THE COMMAND IS SPECIFIED IN BIT6-10 OF THE COMMANDWORD. BIT11 OF THE COMMANDWORD IS SET IF THE COMMAND USES ARGUMENTS.

SOME COMMANDS ACCEPT OPTIONS THAT ARE SPECIFIED IN THE OTHER BITS OF THE COMMANDWORD.

IN GENERAL COMMADS DO PRESERVE LINK AND AC, UNLESS OTHERWISE STATED.

## LIST OF MONITOR COMMANDS.

STALL
INSERT TASK IN TIMEOUTQUEUE (SEC. 1.9.6).
AC =MINUS TIMEOUT VALUE.
OPTIONS: SWPOUT.
AC :=0.

SNDMS
SEND A MESSAGE TO A TASK (SEC. 1.2).
ARG1 =MSPTR.
ARG2 =STATIC TASKNUMBER OF RECEIVERTASK.
OPTIONS: NONREP, DFPARM.

WTRP
WAIT FOR A REPORT (SEC. 1.2).
ARG1 =MSPTR OF WAITED REPORT.
OPTIONS: TIMEOUT, SWPOUT.
WAIT FOR THE REPORT ON THE SPECIFIED MESSAGE. IF NO MESSAGE WAS SPECIFIED (ARG1 =0), WAIT FOR THE FIRST INCOMING REPORT.
AC :=MSPTR OF REPORTED MESSAGE.
LINK:=1.

CHKRPQ
CHECK REPORTQUEUE (SEC. 1.2).

ARG1 =MSPTR OF WAITED REPORT.

IF THE SPECIFIED REPORT IS PRESENT IN THE REPORTQUEUE, OR IN CASE ARG1 =0, IF ANY REPORT IS PRESENT IN THE REPORTQUEUE, AC :=MSPTR OF REPORT. ELSE AC:=0.

LINK:=1.

SNDWTR

SEND A MESSAGE AND WAIT FOR ITS REPORT (SEC. 1.2).

ARG1 =MSPTR OF MESSAGE SENT.

ARG2 =STATIC TASKNUMBER OF RECEIVERTASK.

OPTIONS: DFPARM, TIMEOUT, SWPOUT.

THE SPECIFIED MESSAGE IS SENT TO THE TASK AND THE SENDERTASK IS SET TO WAIT UNTIL THE MESSAGE IS REPORTED.

AC:=MSPTR.

LINK:=1.

RP

REPORT A MESSAGE (SEC. 1.2).

ARG1 =MSPTR OF REPORTED MESSAGE.

IF THE NONREP OPTION WAS USED WHILE SENDING THIS MESSAGE, THE MESSAGE IS NOW RETURNED TO THE AVAILLIST SYSTEM, ELSE THE MESSAGE IS REPORTED TO THE SENDERTASK.

WTMS

WAIT FOR A MESSAGE (SEC. 1.2).

OPTIONS: KEEP, TIMEOUT, SWPOUT.

ERASE PREVIOUS CLAIM, UNLESS KEEP OPTION WAS SPECIFIED. WAIT UNTIL A NEW MESSAGE ARRIVES.

AC:=MSPTR OF RECEIVED MESSAGE.

LINK:=0.

CHKRCQ

CHECK RECEIVEQUEUE (SEC. 1.2).

OPTION: KEEP.

ERASE PREVIOUS CLAIM, UNLESS KEEP OPTION WAS SPECIFIED. IF A NEW MESSAGE HAS ARRIVED, AC:=MSPTR OF THAT MESSAGE. ELSE AC:=0.

LINK:=0.

DISINTR

DISCONNECT FROM INTERRUPT (SEC. 1.5.3).

AC =SKIPIOT.

UPDATE THE SKIPCHAIN SUCH THAT THAT FLAGS OF THE SPECIFIED DEVICE ARE IGNORED (DEVICE DISABLED STATUS).

AC:=0.

CNINTR

CONNECT SUBROUTINE TO INTSLOT (SEC. 1.5.3).

AC =SKIPIOT.

ARG1 =POINTER TO ENTRYPOINT OF SUBROUTINE.

ARG2 =MSPTR OF COMMUNICATION MESSAGE.

THE SUBROUTINE MUST RESIDE IN THE CURRENT DATAFIELD OF THE TASK. IF A COMMUNICATION MESSAGE IS SPECIFIED (ARG2 'NE' 0), THIS MESSAGE MAY BE REPORTED FROM THE CONNECTED ROUTINE TO THE TASK.

AC:=0.

CLINTR

CLAIM AN INTSLOT (SEC. 1.5.3).

AC =SKIPIOT.

ARG1 =CLEARIOT.

ARG2 =MSPTR OF MESSAGE.

ATTACH THE SPECIFIED MESSAGE TO THE INTSLOT OF THE SPECIFIED DEVICE. THIS MESSAGE WILL BE REPORTED TO THE TASK EACH TIME THE CORRESPONDING DEVICE INTERRUPTS.

AC:=0.

FRINTR

SET INTSLOT INTO DISCARD MODE (SEC. 1.5.3).

AC =SKIPIOT.

ARG1 =CLEARIOT.

WHEN THE INTSLOT IS IN DISCARD MODE, INTERRUPTS OF THE CORRESPONDING DEVICE ARE DISCARDED.

AC:=0.

SIMINTR

SIMULATE INTERRUPT (SEC. 1.5.4).

ARG1 =ENTRYPOINT OF DEAF SECTION.

ENTER A DEAF SECTION AS IF IT WERE ENTERED FROM THE INTERRUPT SECTION. IT MUST BE LEFT USING THE IEXIT INSTRUCTION. THE SECTION MUST RESIDE IN THE CURRRENT DATAFIELD OF THE TASK.

REQPAG

REQUEST A BUFFER OF 1-37 OCT CONSECUTIVE PAGES OF CORE (SEC. 1.8.4).

AC =0.

ARG1 BIT5 =CORERESIDENT CONDITION.

ARG1 BIT7-11 =LENGTH OF REQUESTED BUFFER.

IF THE CORERESIDENT CONDITION IS SPECIFIED (BIT5 SET) THE BUFFER WILL BE ALLOCATED IN A CORERESIDENT FIELD.

AC BIT0-4 ;=NUMBER OF FIRST PAGE OF BUFFER.

AC BIT6-11 :=VIRTUAL FIELDNUMBER OF BUFFER.

```
RTNPAG
    RETURN PAGES PREVIOUSLY REQUESTED USING THE REQPAG COMMAND
    (SEC. 1.8.4).
AC BIT0-4 =PAGENUMBER OF FIRST PAGE OF RETURNED BUFFER.
AC BIT6-11 =VIRTUAL FIELDNUMBER OF RETURNED BUFFER.
ARG1 BIT7-11 =NUMBER OF PAGES RETURNED.
AC:=0.

REQFLD
    REQUEST A FIELD FOR DATASTORAGE (SEC. 1.8.2).
AC:=VIRTUAL FIELDNUMBER.

RTNFLD
    RETURN A FIELD (SEC. 1.8.2).
AC =VIRTUAL FIELDNUMBER OF RETURNED FIELD.
AC:=0.

REQSTL
    REQUEST ENTRY IN STL (SEC. 2.1).
ARG1,ARG2 =CONTENTS OF THE REQUESTED ENTRY.
    SEARCH A FREE ENTRY IN THE STL AND DENOTE THE INDICATED CONTENTS
    IN IT. RETURN THE CORRESPONDING STATIC TASKNUMBER IN AC.
AC:=STATIC TASKNUMBER.

REQTCB
    ATTACH TCB TO ENTRY IN STL, OR RETRIEVE TCB ATTACHED TO ENTRY IN
    STL (SEC. 2.1).
AC =STATIC TASKNUMBER.
    IF A TCB WAS NOT YET ATTACHED TO THE TASK, THEN ATTACH ONE,
    PREFILLED WITH INITIAL VALUES.
AC:=TCBPTR.

RTNTCB
    DETACH TCB FROM TASK AND RETURN IT TO THE AVAILLIST (SEC. 2.1).
AC =STATIC TASKNUMBER.
AC:=0.

LOCK
    LOCK DATAFIELD IN CORE (SEC. 1.4.4).

UNLOCK
    UNLOCK DATAFIELD (SEC. 1.4.4).

REQCDF
    REQUEST QUICK ACCESS TO DATAFIELD (SEC. 1.4.2).
AC =REQUESTED DATAFIELD.
```

SETS THE DATAFIELD LOADED INTO THE DFR AT THE EXECUTION OF A VRCDF PSEUDO INSTRUCTION.
NOTE: AC REMAINS UNCHANGED!

EXIT
TERMINATE THE EXECUTION OF THIS TASK (SEC. 1.1.2).
TASK IS RESET INTO INITIAL STATUS. A NEW MESSAGE SENT TO IT WILL START UP A FRESH COPY OF THE TASK.

STOP
STOP A TASK (SEC. 1.3.2).
AC =STATIC TASKNUMBER.
AC:=0.

RESUME
RESUME A TASK (SEC. 1.3.2).
AC =STATIC TASKNUMBER.
AC:=0.

FILTCB
WRITE VALUES INTO THE TCB OF A TASK (SEC. 2.1).
AC =STATIC TASKNUMBER.
ARG1 =POINTER TO VALUE LIST.
IF A TCB IS NOT YET ATTACHED TO THE TASK, THEN ATTACH ONE. FILL THE FIRST 15 OCT (ALL BUT THE LAST) ENTRIES IN THE TCB WITH THE VALUES INDICATED IN THE VALUE LIST. THE VALUE LIST MUST RESIDE IN THE INSTRUCTIONFIELD OF THE TASK.
AC:=0.

LIST OF OPTIONS.

NONREP
NONREPORT (SEC. 1.2).
THIS OPTION MAY BE SPECIFIED WHEN A MESSAGE IS SENT. IF IT IS, THE MESSAGE WILL BE RETURNED TO THE AVAILLIST SYSTEM WHEN IT IS REPORTED. IF THE OPTION WAS NOT SPECIFIED, THE MESSAGE IS REPORTED TO THE SENDER.

KEEP
KEEP PREVIOUS CLAIM (SEC. 1.2.1).
WHEN SPECIFIED IN CONJUNCTION WITH A WTMS OR CHKRCQ COMMAND, THE TASK REMAINS CLAIMED BY THE SAME TASK AS BEFORE. OTHERWISE THIS CLAIM TERMINATES.

SWPOUT

SWAP TASKCODE OUT (SEC. 1.9.4).
WHEN SPECIFIED IN CONJUNCTION WITH ONE OF THE COMMANDS WTMS, WTRP, SNDWTR, OR STALL, THE PAGES OCCUPIED BY THE TASKCODE ARE RETURNED IF THE TASK IS SET TO WAIT. A FRESH COPY OF THE TASKCODE IS SWAPPED IN, AS SOON AS THE TASK IS RUNNABLE AGAIN.

TIMEOUT

RESTART TASK AFTER A WHILE (SEC. 1.9.6).
IF THIS OPTION IS SPECIFIED IN CONJUNCTION WITH THE COMMANDS WTMS, WTRP, SNDWTR, THE TASK IS RESUMED AFTER THE SPECIFIED DELAY EVEN IF THE OTHER WAITCONDITIONS ARE NOT YET FULFILLED. MINUS THE TIMEOUT VALUE (LENGTH OF DELAY IN UNITS OF 0.1 SEC) MUST BE SPECIFIED IN AC. IF ONE OF THE OTHER WAITCONDITIONS EXPIRES BEFORE THE END OF THE DELAY, THE TIMEOUT OPTION HAD NO EFFECT. OTHERWISE THE TASK IS RESUMED AFTER THE DELAY WITH AC=0.

DFPARM

ACTUAL DATAFIELD AS PARAMETER (SEC. 1.2, SEC. 1.9.2).
WHEN THIS OPTION IS SPECIFIED IN CONJUNCTION WITH ONE OF THE COMMANDS SNDMS OR SNDWTR, THE ACTUAL DATAFIELDNUMBER IS COPIED INTO BIT6-8 OF MESSAGE[2] (FIRST INFORMATIONWORD).

THE TEXT PRESENTED BELOW MAY BE USED TO BUILD THE MC8 SYSTEM
UNDER CONTROL OF THE OS/8 OPERATING SYSTEM, USING
THE PROGRAM BATCH.SV.
THE PROGRAM GLOBL.SV SELECTS SYMBOLS FROM THE PAL8 SYMBOLTABLE
AND ARRANGES THEM IN A 'DIRECT ASSIGMENT' FILE.
THIS FILE IS USED AS PREFILE WHEN ASSEMBLING TASKS.
OP.PA IS A PREFILE HOLDING OBVIOUS DEFENITIONS.

```
$JOB BUILD MC8 FOR PDP8E.
.PAL MC,MC<OP,MCCONF.E,MC81,MC82,MC83,MC84,MC85/N/D/H
.R GLOBL
*MC8.SY<MCTSYM,MC.LS
.R PIP
*OPMC8E.PA<OP.PA,MC8.SY$
.R GLOBL
*MC8SYS.SY<MCSYSM,MC.LS
.PAL MC8HLT<OPMC8E.PA,MC8HLT
.PAL MC8IN<OP,MC8SYS.SY,MC8IN
.R ABSLDR
*MC,MC8HLT,MC8IN$
.SA SYS MC8
.SA SYS FLD0 0-5000
.PAL OP,MCCONF.E,MC8SYS,SY,MC8BUL,IO/L=10200
.SA SYS MC8BUL
.PAL OP,MCCONF.E,MCTBLD,IO/L
.SA SYS MCTBLD
.R MC8PAL
*OP,MCCONF,E,MC8SYS.SY,MCTSLD/L/Z$
.SA SYS MCILIB 20000-27777;25000
.R MCILIB
.R MC8BUL
*X.TM<SYS:FLD0/Y/Z
$END
```

```
FEB 21 11:08 1977  TEST PAGE    1

 *****  ******    ****    *****
   *    *        *          *
   *    *****     ****      *
   *    *             *     *
   *    *        *    *     *
   *    ******    ****      *

PROGRAM TEST(INPUT,OUTPUT);
BEGIN
WRITELN('SOME ','CONSTANTS');
WRITELN();
WRITELN(1.0E50,1.0E+50,-1.0E50,-1.0E+50);
WRITELN(1.0E38,1.0E+38,-1.0E38,-1.0E+38);
WRITELN(1.0E10,1.0E+10,-1.0E10,-1.0E+10);
END.


PC TEST
SCAN 1: 115 (=0163) BYTES FOR CODE
SCAN 2: 113 (=0161) BYTES FOR CODE
SCAN 3: 113 (=0161) BYTES FOR CODE
30 (=036) BYTES FOR CONSTANTS


PASCAL
SOME CONSTANTS

 +1.694765E+38 +1.000000E+00 -1.694765E+38 -1.000000E+00
 +9.999999E+37 +1.000000E+00 -9.999999E+37 -1.000000E+00
 +1.000000E+10 +1.000000E+00 -1.000000E+10 -1.000000E+00
```

```
FEB 21 11:16 1977  TST PAGE   1

 *****   ****    *****
   *    *          *
   *     ****      *
   *         *     *
   *    *    *     *
   *     ****      *

PROGRAM TEST (INPUT,OUTPUT);
BEGIN
WRITELN('OUTPUT!');
WRITELN(1.0E10,1.0E+10,-1.0E10,-1.0E+10)
END.
OUTPUT:
 +1.000000E+10 +1.000000E+00 -1.000000E+10 -1.000000E+00
```