TOPS-10

Version 7.01 Update Document

Customer Version

digital

EY-CD139-SG-001

TOPS-10 VERSION 7.01 UPDATE DOCUMENT

Educational Services
Digital Equipment Corporation
Marlborough, Massachusetts

October, 1980

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont.)

PART THREE KL10 SERVICE ENHANCEMENT PROJECT

Chapter 3 KL10 SERVICE ENHANCEMENT PROJECT

PART FOUR GALAXY VERSION 4.1

Chapter 4 GALAXY REORGANIZATION

TABLE OF CONTENTS (Cont.)

TABLE OF CONTENTS (Cont.)

TABLE OF CONTENTS (Cont.)

TABLE OF CONTENTS (Cont.)

TABLE OF CONTENTS (Cont.)

TABLE OF CONTENTS (Cont.)

FIGURES

TABLE OF CONTENTS (Cont.)

TABLE OF CONTENTS (Cont.)

Tables

PREFACE

This document presents the changes made to TOPS-1Ø Version 6.Ø3A that produced Version 7.Ø1. The major changes for this release are the KL1Ø Service Enhancement Project, Galaxy Version 4.1, and Symmetric Multiprocessing.

IMPORTANT

The information about Galaxy 4.1 and its related components (Usage Accounting, MDA, etc.) is subject to change. The release of Galaxy 4.1 has been decoupled from 7.Ø1 and will not occur until February 1981. Any information contained in this document about those features is preliminary and may change before the release.

PART ONE

INTRODUCTION

Chapter 1

INTRODUCTION TO VERSION 7.01

Release 7.01 of TOPS-10 is the first general release of TOPS-10 since 6.03A. The main thrust of 7.01 is to include the KL10 Service Enhancement Project items, Usage Accounting, and add support for Galaxy Version 4.1 (including MDA and tape labeling). It contains all the features of release 7.00 [an interim release for SMP (Symmetric Multiprocessing) systems only]. Release 7.01 extends SMP to KI10 systems (although there are hardware restrictions on the second CPU) and is the first TOPS-10 release that does not support KA10 processors and KA10-type devices. It is expected that 7.01 will be the last release of TOPS-10 that supports KI10s.

SYMMETRICAL MULTIPROCESSING

Previously, TOPS-10 dual CPU systems used a master/slave approach. One CPU was designated as the master and had all I/O devices connected to it. The slave CPU handled only compute-bound jobs. However, with a system of this type: 1) a large amount of overhead is required in order to context switch a job from the slave CPU to the master CPU when I/O must be done, and 2) if the master CPU goes down, the whole system is down.

The solution to these problems is Symmetrical Multiprocessing (SMP). Under this configuration, either CPU may have I/O devices (except for KI10 systems which cannot have I/O devices on the second CPU). If a job running on CPU0 must do input or output on CPU1, it queues an I/O request to the other CPU, eliminating the need for a context switch. In addition, disks may be dual-ported between CPUs, removing the need for the queued I/O request. Furthermore, if one CPU goes down, the other will still be able to run the system by itself.

The performance goals of SMP are to:

1. Be able to log in 2ØØ jobs, including operator jobs.

2. Provide 1.75 times the throughput of a single CPU system running Version 6.Ø3A of TOPS-1Ø.

3. Provide an acceptable level of response when the monitor is configured to run 175 jobs.

SMP is an unbundled product in release 7.Ø1.

KL10 SERVICE ENHANCEMENT PROJECT

The KL1Ø Service Enhancement Project is a major effort by Software Engineering, Product Support, and Field Service to improve TOPS-1Ø's ability to understand errors. This project grew out of the need to track down intermittent bugs in the KL1Ø processor. But the work has benefited TOPS-1Ø on all processors. Some of the new features include:

1. Better stopcode reporting and recovery.
2. TGHA for MOS memory error processing.
3. Better parity error reporting.
4. Automatic crash handling procedure.

These new features are augmented by better stopcode documentation and the complete testing/validation of all TOPS-1Ø error routines.

GALAXY

Galaxy Version 4.1 will be released four to five months after 7.Ø1. There are many changes on many levels for this product including:

1. The Operator Interface - The OPR program now allows the operator to control all system features using a single program. OPR is easy to use because of command recognition and improved help facilities.

2. Galaxy Components - All Galaxy programs now use a common library and communicate using IPCF messages. Because of OPR, the command parsers for individual elements (LPTSPL, BATCON, etc.) have been removed. UMOUNT has been replaced by MOUNT; OMOUNT was split and placed in QUASAR and PULSAR.

3. Mountable Device Allocator (MDA) - Galaxy will attempt to identify and assign disks and tapes without operator assistance. In addition, it can eliminate potential deadlock situations.

4. Labeled Tapes - This feature has been added to Galaxy to allow MDA to identify mounted tapes and provide an additional level of security for tape mounts.

USAGE ACCOUNTING

The procedure to collect and summarize accounting data has been simplified and now works in a similar fashion as TOPS-2Ø. Each individual component (input and output spoolers, disk usage, etc.) now sends its information directly to the accounting DAEMON (ACTDAE) using IPCF messages. ACTDAE collects the information and writes the account files. Modifications need only be made to ACTDAE, not to the individual components. Usage accounting depends on Galaxy Version 4.1.

Users may now divide charges among entities known as accounts which are accepted and validated at LOGIN time. Users may also flag terminal sessions using session remarks, also entered at LOGIN time.

NOTE

The information contained in the Galaxy and Usage Accounting sections is of a preliminary nature only. It is liable to change before the release of Galaxy 4.1 in February 1981.

CONFIGURATIONS

Support for the KA10 has stopped with Version 6.03A of the TOPS-10 monitor. Still supported are 1070s, 1080s, 1090s, 1091s, and 2020s. These 1070s, 1080s, and 1090s can be arranged in SMP configurations.

DECsystem-1070s and 1080s can support the following configurations (component name and number of units):

Processor:KI10,KL10A	1-2 [Not KI & KL mixed]
Memory:MH10,MG10,MF10	256-4096K [Single CPU] 512-4096K [Dual CPU]
External Channels: DF10,DF10C,DX10,DL10	
DF10C/DF10	
RH10:RP04/6	0-8
TM10B:TE10	0-8
TM10B:TU40,TU41	0-8
RH10:RS04	0-8
RP10:RP02/3	0-8
RH10:TU16	0-8
DX10	
TX01:TU70,TU71	0-8
TX02:TU70,TU71,TU72	0-8
DL10	0-2 [4 ports/DL10]
DN87	0-4 [Must be upgraded to 32K]
DN61	0-1 [Must be upgraded to 32K] [Must be on CPU0]
DN62	0-1 [Must be upgraded to 32K] [Must be on CPU0]
Option:DIA10 [I/O bus devices]	[0-3 LPT'S/CPU]
LP100:LP05,LP07,LP14	0-3/CPU
BA10:LP10F,LP10H	
BA10:CR10E,CR10F	[0-2]
CP10D:CP10D	[0-1]
BA10:XY10	
TD10:TU55,TU56	0-8 Drives
DK10	0-1

The DECsystem-1090 supports the following configurations (component name and number of units):

```
Processor:KL10D,KL10B          1-2 [Not mixed]

Memory:MH10,MG10,MF10          256-4096K [Single CPU]
                               512-4096K [Dual CPU]

External Channels:DF10,DF10C,DX10,DL10
  DF10C/DF10
     RH10:RP04/6                 0-8
     TM10B:TE10                  0-8
     TM10B:TU40,TU41             0-8
     RH10:RS04                   0-8
     RP10:RP02/3                 0-8
     RH10:TU16                   0-8
  DX10
     TX01:TU70,TU71              0-8
     TX02:TU70,TU71,TU72         0-8
  DL10                           0-2 [4 ports/DL10]
     DN87                        0-4 [Must be upgraded to 32K]
     DN61                        0-4 [Must be upgraded to 32K]
                                     [Must be on CPU0]
     DN62                        0-4 [Must be upgraded to 32K]
                                     [Must be on CPU0]
  Option:DIA10 [I/O bus devices] [0-3 LPT'S/CPU]
  LP100:LP05,LP07,LP14           0-3/CPU
  BA10:LP10F,LP10H
  BA10:CR10E,CR10F              [0-2]
  CP10D:CP10D                   [0-1]

Internal Channels:DTE,RH20
  DTE'S                          1-4
     RSX20F
     DN87S                       0-4 [Must be upgraded to 32K]
     DN20                        0-1 [Must be upgraded to 32K]
     DN61S                       0-1 [Must be upgraded to 32K]
  RH20'S                         0-8/CPU
     RP04/6                      0-8/RH20
     TM03[0-8]:TU45,TU77         0-4/TM03
     TM02[0-8]:TU45,TU77         0-4/TM02
     DX20[0-X]:TX02:TU70,TU71,TU72 0-8
```

The DECsystem-1Ø91 can support the following configurations (component name and number of units):

```
Processor:KL1ØE  [Only one processor allowed]

Memory:MA2Ø,MB2Ø,MF2Ø          256K-1536K

      [No external channels]
      Option:DIB2Ø

Internal Channels:DTE,RH2Ø
  DTE'S                         1-4
      DTE:RSX2ØF
          DC2Ø
          LP2Ø:LPØ5,LPØ7,LP14  Ø-2
          CD2Ø:CR1ØE,CR1ØF     Ø-1
      DTE:DN2Ø                  Ø-3 [Must be upgraded to 32K]
      DTE:DN87S                 Ø-3 [Must be upgraded to 32K]
      DTE:DN61S                 Ø-1 [Must be upgraded to 32K]
  RH2Ø'S                        1-8
      RH1Ø:RPØ4/6               Ø-16
      RH2Ø:TMØ2/3:TU45'S,TU77'S       Ø-16
      RH2Ø:DX2Ø:TXØ2:TU7Ø,TU71,TU72 Ø-8
```

The KS1Ø processor (DECSYSTEM-2Ø2Ø) can support the following configurations:

```
Processor:KS1Ø  [Only one processor allowed]

Memory:MS1Ø                     256K-512K

Controllers:RH11                2
  RH11:RPØ6,RMØ3                1-8
  RH11:TMØ3:TU45'S              1-4
  Option:Unibus Adaptor         2  [1 for disk,
                                    1 for tape and all else]
  Option:Unit Record
  LP2Ø:LPØ5,LP14                Ø-1
  CD2Ø:CR1ØE,CR1ØF              Ø-1

  Option:Communications
  DZ11                          1-4 8-32 Asynchronous lines
  KMC/DUP11                     Ø-2 Synchronous lines
```

The following network remote stations can be connected to any of the TOPS-1Ø systems:

1. DN92 (Except on a KS1Ø which does not support sequential nodes)

2. DN2ØØ

3. DN8Ø,DN81,DN82

The following terminals are understood by TOPS-1Ø:

LA36,LA37,VT5Ø,VT52,VTØ5,ASR33,ASR35,LA12Ø,LA18Ø
LA3Ø,VT1ØØ,2741,VT61,VT62

HARDWARE CHANGES

The TU77 is a new device, a 9-track 125-inch/second tape drive that works on a TMØ3/RH2Ø controller/channel. It supports recording densities of 8ØØ and 16ØØ bits/inch.

The TMØ3 tape controller is now supported on an RH2Ø controller; it is not supported on an RH1Ø. Both the TU45 and TU77 tape drives are supported on the TMØ3, which does not support the "standard ASCII" mode as does the TMØ2.

TU7Øs may now be connected to DX2Ø/RH2Ø controller/channels.

The LP14 line printer is supported on the LP1ØØ and LP2Ø controllers. The LP14 is a 132-column printer with two options:

1. The LP14-V with a 64-character print set, a speed of 89Ø lines per minute, and a programmable VFU.

2. The LP14-W with a 96-character print set, a speed of 65Ø lines per minute and a programmable VFU.

The DX1Ø variable length records have changed so that the DX1Ø microcode now allows the user to process variable length records without needing to handle only one at a time. Under Version 6.Ø3A, the user specified a maximum buffer size and read a shorter record into the buffer. The monitor then received a short record length error from the DX1Ø. The DX1Ø would determine the correct record length, return it to the user and restart the I/O. In Version 7.Ø1, the DX1Ø deposits the actual transfer length in memory and continues to process the request without stopping the tape.

The M93Ø1 ROMs are required to allow down-line loading of a DN87 front-end when the CPU to which it is connected (via a DTE) is down.

In the KL1Ø microcode the DPB instruction has been modified to use a read-modify-write cycle when inserting a byte into a memory word. This is necessary to preserve the monitor data base on an SMP system. To make the DPB work in this fashion on a 1Ø77 system, a special ECO must be installed. This ECO was not available for release 7.ØØ but has been approved for 7.Ø1.

The following KA1Ø-era devices are no longer supported:

DC1Ø	68ØI
TU43	TC1Ø
CP1ØA	RC1Ø: RD1Ø, RM1ØB
DC44	TM1ØA: TU1Ø, TU2Ø, TU3Ø
LP1ØA	LP1ØC
DC72	DC76
DC75	DC75-NP

PART TWO

SYMMETRICAL MULTIPROCESSING

Chapter 2
SYMMETRICAL MULTIPROCESSING

MASTER/SLAVE IMPLEMENTATIONS OF DUAL PROCESSORS

The master/slave approach to dual processors was first implemented in the 5.Ø4 release of TOPS-1Ø. It was done primarily for the University of Pittsburgh, who had two KA1Øs and a specific job mix - compute bound and I/O bound jobs. To solve problems encountered with that mix, the master/slave scheme had all I/O devices connected to the master CPU. All the I/O bound jobs ran on the master and all the compute-bound jobs (with little or no I/O) ran on the slave. The master CPU had other duties also. Since it controlled all the I/O devices, it was used to run all UUOs, because most UUOs are concerned with I/O. Command decoding, scheduling, and swapping were also handled by the master so that the monitor would not have to be reorganized and made re-entrant. This master/slave arrangement proved to be popular not only at Pittsburgh but at other sites. Master/slave was released for KI1Øs with TOPs-1Ø Version 5.Ø7 and for KL1Øs with Version 6.Ø3.

Problems

The master/slave implementation became increasingly unpopular with 5.Ø7 and later versions. The release of the KI1Ø and the KL1Ø brought forth progressively faster and more reliable hardware, but the increases were not matched by the performance of the master/slave monitor.

There were two main problems with master/slave dual CPU arrangements which in turn caused others. The first was that the entire system became unavailable if the master went down because all the I/O devices were connected to it. Further, if the master CPU encountered a stopcode, the slave could not reload the monitor. Some sites tried to circumvent this problem by using expensive and complicated bus-switching systems to give devices to the slave CPU, But this type of equipment was not justified for the majority of users.

The second problem showed up as a slow system, indicating too much overhead and inefficiency. This was because monitor calls could only be executed on one CPU. Every time a slave issued a monitor call, it was rescheduled to run on the master CPU; that required a context switch, a very time-expensive operation. TOPS-10 must switch the EPMP, UPMP, registers, and a certain amount of job information every time it performs a context switch. Master/slave caused many of these context switches to and from the master CPU. The result was that the master CPU had to run most of the jobs while the slave sat idle. The I/O bound/compute-bound job mix no longer held true - the role of computing had changed. And the users did not see the performance improvement that they expected.

An indirect result of the monitor call problem was that cache, introduced with the KL10, was not used properly. The master performed all important system tasks, including cache sweeps. The monitor was very defensive in cache programming, performing more sweeps than it had to. Context switching involves sweeping cache, so extra context switching forced more cache sweeps.

SYMMETRICAL MULTIPROCESSING

The solution to these problems was Symmetrical Multiprocessing (SMP). I/O devices can be connected to either CPU (or in the case of disks, dual-ported between CPUs) to prevent hardware or software problems on one CPU from crashing the whole system. To improve performance, the monitor was made re-entrant so that most monitor calls were executable from either CPU.

The only exception is SMP on KI10s. CPU devices can be connected to either CPU but only those devices connected to CPU0 can be used; this eliminates sharing of I/O devices.

The CPU that runs first when the system is brought up is known as the <u>boot CPU</u>. The CPU that handles command processing and swapping is called the <u>policy CPU</u>. When the system is started, the boot CPU and the policy CPU are one and the same. The non-boot CPU may become the policy CPU if, for example, the boot CPU crashes. One way to determine the identity of the policy CPU is to look at location PLCYCP in the running monitor; it holds the serial number of the policy CPU. Another way to determine the identity of the policy CPU is to check the CTYs. The following message prints on the CTY of the new policy CPU, if a policy switch occurs:

```
CPUx HAS ASSUMED THE ROLE OF POLICY CPU AT <date/time>
```

The clocks of the two CPUs do not interrupt at the same time. Instead, they are skewed by half a cycle.

The configuration is not perfectly symmetrical; command processing and swapping are handled by only one CPU. A few UUOs (DTE., PERF., DIAG., and RTTRP.) require a CPU argument, and can run only on the requested CPU, but scheduling and most UUOs can be handled by either CPU. The METER. and LOCK monitor calls can only be run on the policy CPU.

The software as it is currently designed could run with up to six CPUs.

I/O Devices on Either CPU

Once the devices are physically connected to either CPU, they must be software defined in MONGEN:

```
DECSYSTEM10(1070,1080,1090,1091,2020)[
1070 IS A SYSTEM WITH KI10 CPU's
1080 IS A SYSTEM WITH KL10 CPU's
1090 IS A SYSTEM WITH KL10 CPU's AND INTERNAL CHANNELS
1091 IS A SYSTEM WITH A KL10 CPU, INTERNAL MEMORY, AND
        DEVICES ON THE FRONT-END
2020 IS A SYSTEM WITH A KS10 CPU]: 1090
CPU's(1,1-4)[TOTAL NUMBER OF CPU's IN THE SYSTEM]: 2
CPU0 SERIAL  (1-10000): <SERIAL NUMBER OF FIRST CPU>
CPU1 SERIAL  (1-10000): <SERIAL NUMBER OF SECOND CPU>
```

NOTE

Do NOT mix a KI1Ø and KL1Ø together.

Next MONGEN asks for the channels, controllers, and devices connected to each CPU starting with CPUØ. These questions will not be asked of a DECsystem-1Ø77.

CHANNELS ON CPUØ(2,Ø-?):

<Define the controllers and channels here>

After all the channels have been defined for this CPU, the next question is:

CHANNELS ON CPU1(2,Ø-?):

The maximum number of channels on a CPU is eight. The dialogue to define a channel and the devices on it is basically the same as it was for Version 6.Ø3A.

DTEs may also be connected to either CPU. MONGEN asks the following questions for each CPU:

DTES ON CPUx(2,1-4):

TERMINALS ON THE MASTER FRONT-END ON CPUx(Ø,Ø-128):

LINE PRINTERS ON THE MASTER FRONT-END ON CPUx(Ø,Ø-2):

CARD READERS ON THE MASTER FRONT-END ON CPUx(Ø,Ø-1):

As an aid to making disks more available, they may be dual-ported between CPUs. In the event one CPU goes down, the disk is still available. Public structures should be dual-ported between CPUs. This eliminates switching disk packs if one CPU crashes.

The monitor is not told which drives are dual-ported when you run MONGEN. Instead, this function is performed when the system is loaded and initialized during the startup dialogue in ONCE. The serial numbers of the disk drives are read and compared. If a match occurs, the drive is marked as being dual-ported.

Re-Entrant Monitor

In order to allow monitor calls on either CPU, most of the TOPS-1Ø monitor had to be made re-entrant (i.e., while it is being executed, the other CPU may begin executing the same piece of monitor code). This required much of the monitor data base to be moved out of the monitor low segment. Thus, funny space was created.

When a job is running, pages 34Ø to 367 of the executive virtual address space are designated as the Per-Process Monitor Free Core, also known as funny space; this is 12K of monitor free core that swaps with the job. A job's funny space contains its disk DDBs, monitor buffers, temporary search lists, channel information, SWITCH.INI, and TMPCOR. Funny space corresponds to pages 1ØØØ to 1Ø27 in WSBTAB (the table in the UPMP that specifies which pages are in the working set). Table AABTAB, also in the UPMP, defines which parts of funny space are cached.

Some other ramifications of the reorganization are:

1. 8Ø software I/O channels are now available for one job at one time; this feature helps COBOL programs. Channels higher than 16 are known as extended channels and are used with COBOL V13 and DBMS V6.

2. Disk DDBs CANNOT be obtained reliably for jobs other than your own.

3. System DDBs are still chained with DEVLST pointing to the first entry.

4. By putting the information into a special area and not in the monitor's data base, the need for the MQ resource has been eliminated. No longer will jobs compete for the use of this resource.

5. Only disk DDBs are in funny space; network DDBs are not.

6. Funny space makes each job about two pages larger (on the average) and the monitor free core smaller.

Several data structures had to be modified for this change. JBTTMP, the monitor table containing TMPCOR files, was removed. The left half of JBTPDB contains the number of funny pages in/out of memory. The left half of .UPLST in the UPMP points to the first DDB for the job. The left half of .UPFCC points to the first TMPCOR file. RIB pointers and disk queues also had to be modified.

Other important locations in core have been changed as a result of the implementation of funny space. The following locations were changed (shown with their new values):

UPMP	370000	(page 370)
.JDAT	371000	(page 371)
.VJDT	372000	(page 372)
.TEMP	373000	(page 373)

To accommodate the second CPU, these tables are stored in the following locations:

TABLES	CPU0	CPU1
.EPMP	1000	2000
CDB	411000	412000

The EPMP(Executive Page Map Page) for the CPUs was moved out of page 0 to avoid problems with the RH20 channel logout area.

Figure 2-1 displays the old and new arrangement of Executive Virtual Memory.

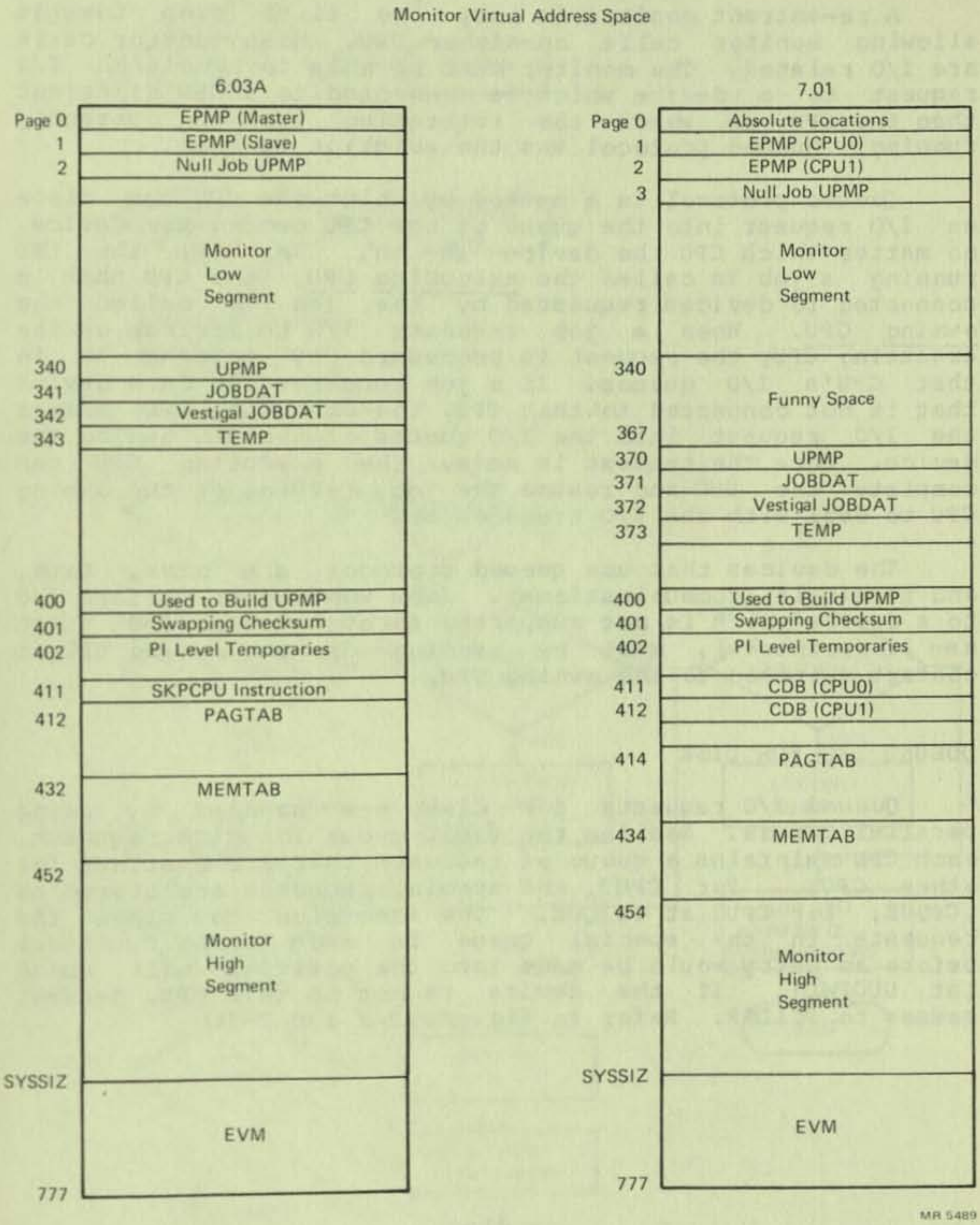


Figure 2-1 Executive Virtual Memory

Queued Protocol

A re-entrant monitor is only the first step towards allowing monitor calls on either CPU. Most monitor calls are I/O related. The monitor must be able to handle an I/O request to a device which is connected to a CPU different than the CPU on which the requesting job is currently running; queued protocol was the solution to this.

Queued protocol is a method by which one CPU can place an I/O request into the queue of the CPU owning the device, no matter which CPU the device is on. In SMP, the CPU running a job is called the executing CPU; the CPU that is connected to devices requested by the job is called the owning CPU. When a job requests I/O to devices on the executing CPU, the request is processed by putting it in that CPU's I/O queues. If a job requires I/O on a device that is not connected to that CPU, the executing CPU enters the I/O request into the I/O queues of the CPU owning the device. Once the request is made, the executing CPU can complete the UUO and resume the job, relying on the owning CPU to deal with the I/O transfer(s).

The devices that use queued protocol are disk, tape, and terminals (communications). Jobs wanting to perform I/O to a device which is not supported through queued I/O (such as lineprinters), must be running on the owning CPU or context switched to the owning CPU.

QUEUED I/O FOR DISK

Queued I/O requests for disk are handled by using parallel queues. Besides the usual queue for disk requests, each CPU maintains a queue of requests that are destined for other CPUs. For CPU0, the special requests are stored at .C0QUE; for CPU1 at .C1QUE. The decision to place the requests in the special queue is made during UUO level before an entry would be made into the position wait queue (at UUOPWQ). If the device is not on this CPU, control passes to PCLDSK. Refer to Figures 2-2 and 2-3.

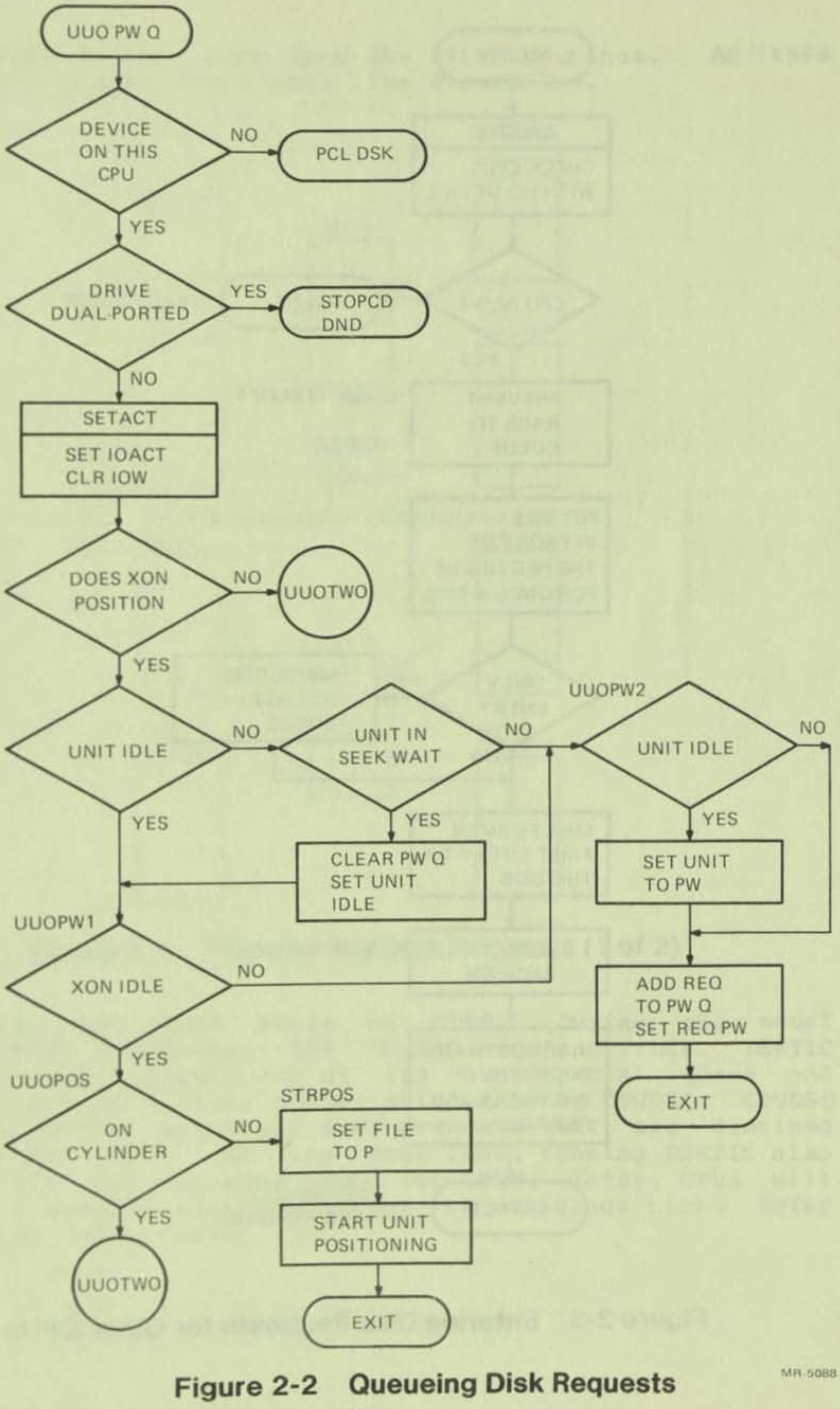


Figure 2-2 Queueing Disk Requests

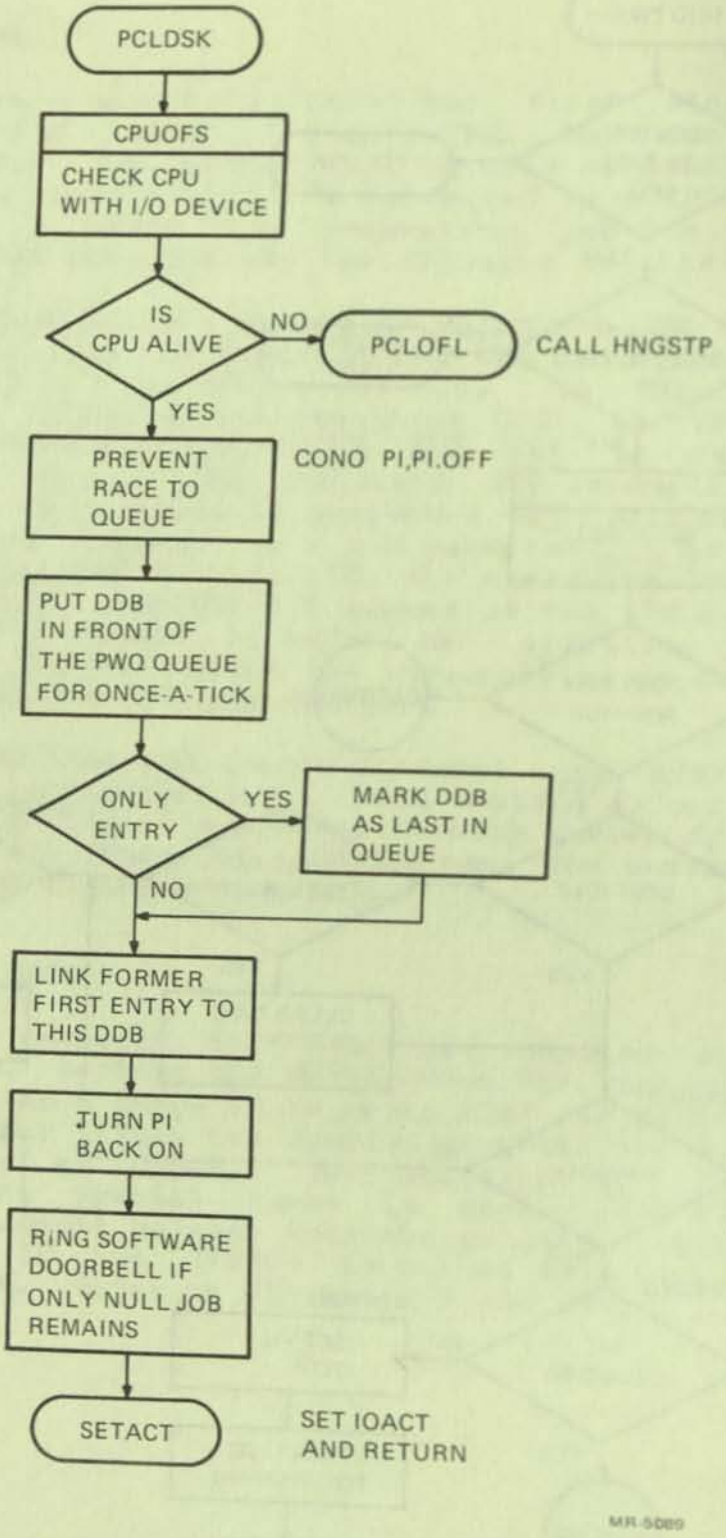


Figure 2-3 Entering Disk Requests for Other CPUs

Return is then made from the FILIO routines. At this point, the queues would look like Figure 2-4.

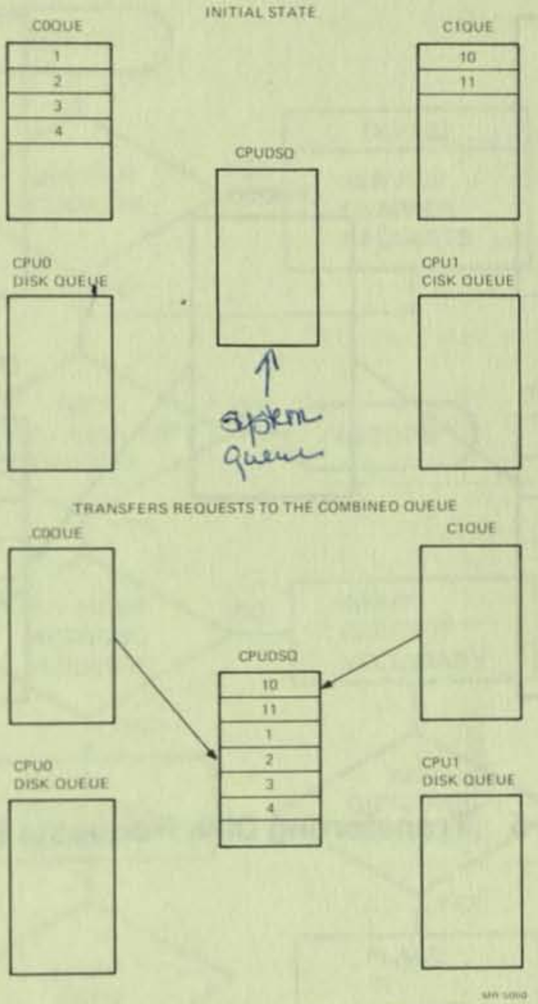


Figure 2-4 Transferring Disk Requests (1 of 2)

Later, the clock ticks on CPU0. During the usual clock-level functions, the DSKTIC routine runs. DSKTIC takes all I/O requests out of its own special queue and places them in another queue, pointed to by CPUDSQ. CPUDSQ (one per system) holds all disk requests that are destined for another CPU. At the same time, running DSKTIC also removes all disk requests bound for CPU0. Later, CPU1 will perform a similar function during its own clock tick. Refer to Figures 2-5 and 2-6.

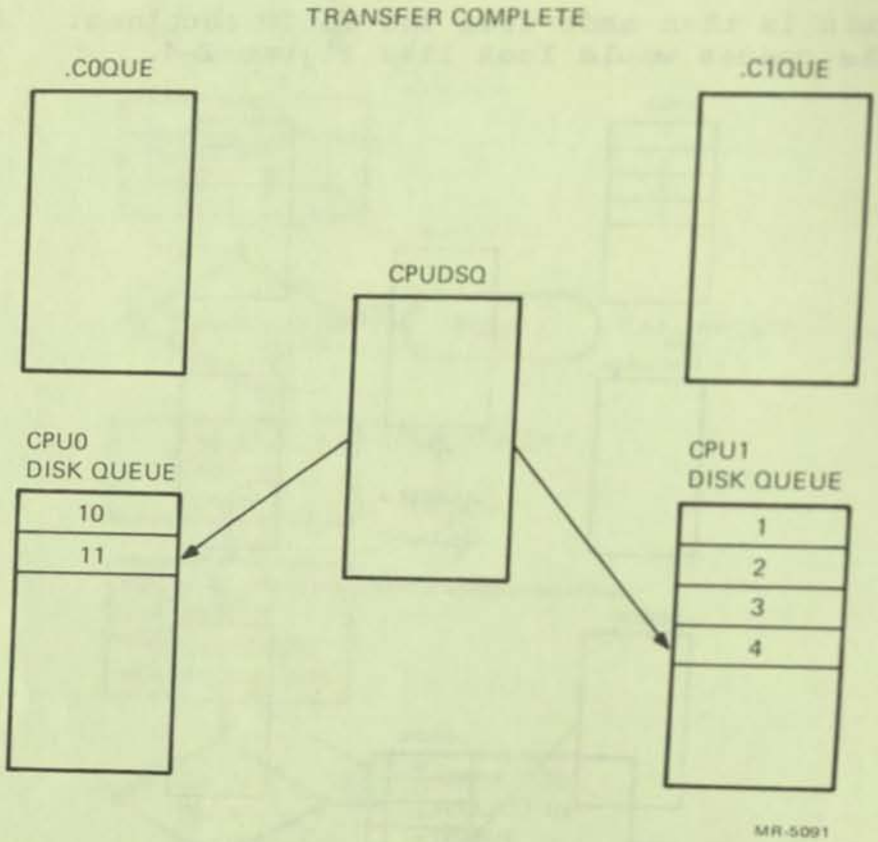


Figure 2-5 Transferring Disk Requests (2 of 2)

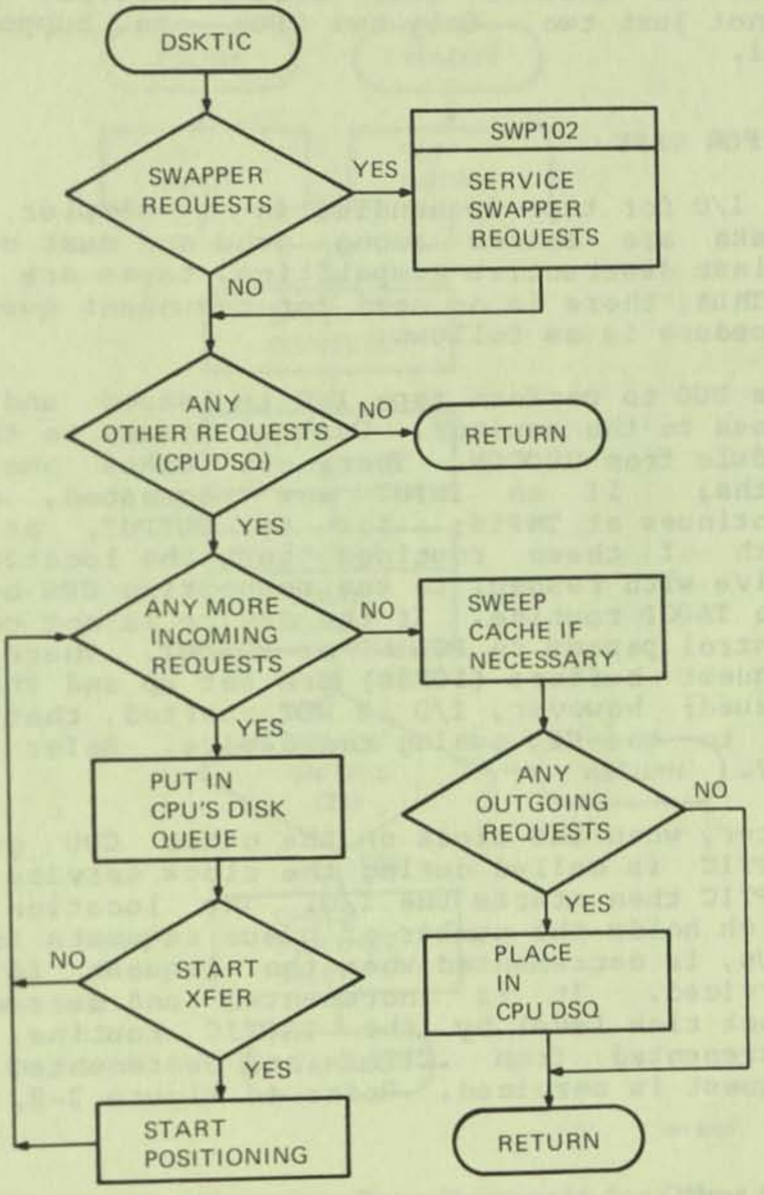


Figure 2-6 Once-A-Tick Disk Routine

Note that CPUDSQ, the one central clearing house for all requests on another CPU, allows multiple CPUs to be connected, not just two. Only two CPUs are supported for release 7.01.

QUEUED I/O FOR TAPE

Queued I/O for tape is handled in a simpler fashion. Whereas disks are shared among jobs and must constantly protect against destructive competition, tapes are owned by one job. Thus, there is no need for redundant queues. The general procedure is as follows:

1. The UUO to perform tape I/O is issued and control comes to the monitor. Control passes to the TAPUUO module from UUOCON. There it takes one of two paths; if an INPUT was requested, execution continues at TMPIN; for an OUTPUT, at TMPOUT. Both of these routines test the location of the drive with respect to the requesting CPU by calling the TAPCP routine. If the device is not connected, control passes to PCLINP or PCLOUT. There, the I/O request buffers (IORBs) are set up and the request queued; however, I/O is NOT started, that is left up to the CPU owning the device. Refer to Figure 2-7.

2. Later, when the clock on the other CPU goes off, TAPTIC is called during the clock service routine. TAPTIC then starts the I/O. The location CPUTPQ, which holds the number of queue requests from other CPUs, is decremented when the request is finally serviced. It is incremented and decremented at clock tick level by the TAPTIC routine. It is incremented from .CPTAP and decremented when the request is serviced. Refer to Figure 2-8.

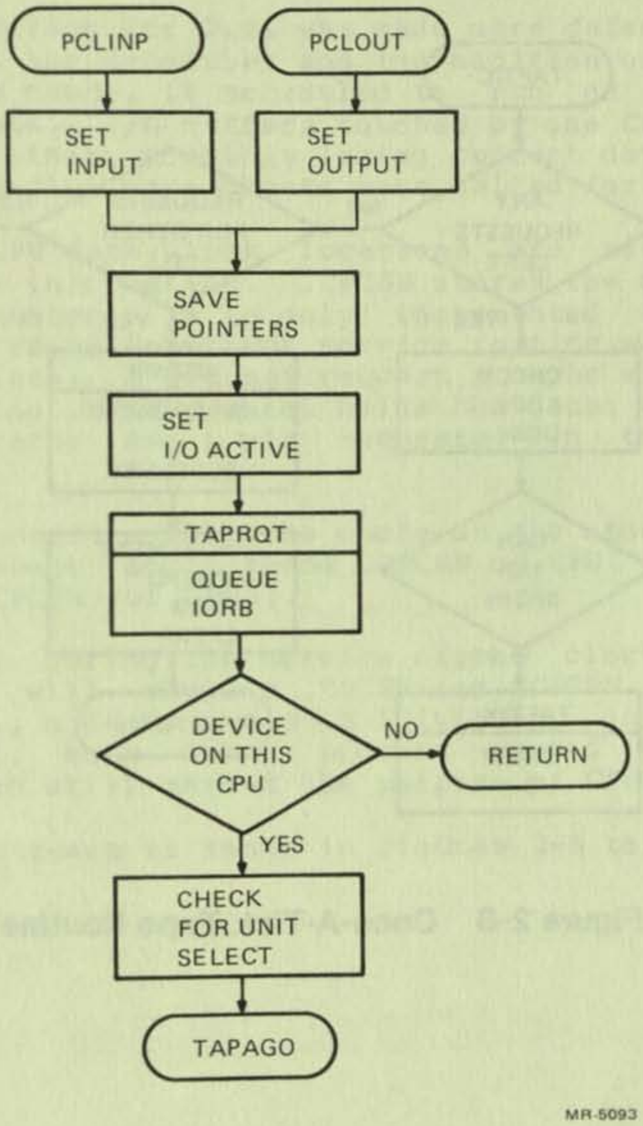


Figure 2-7 Entering Tape Requests for Other CPUs

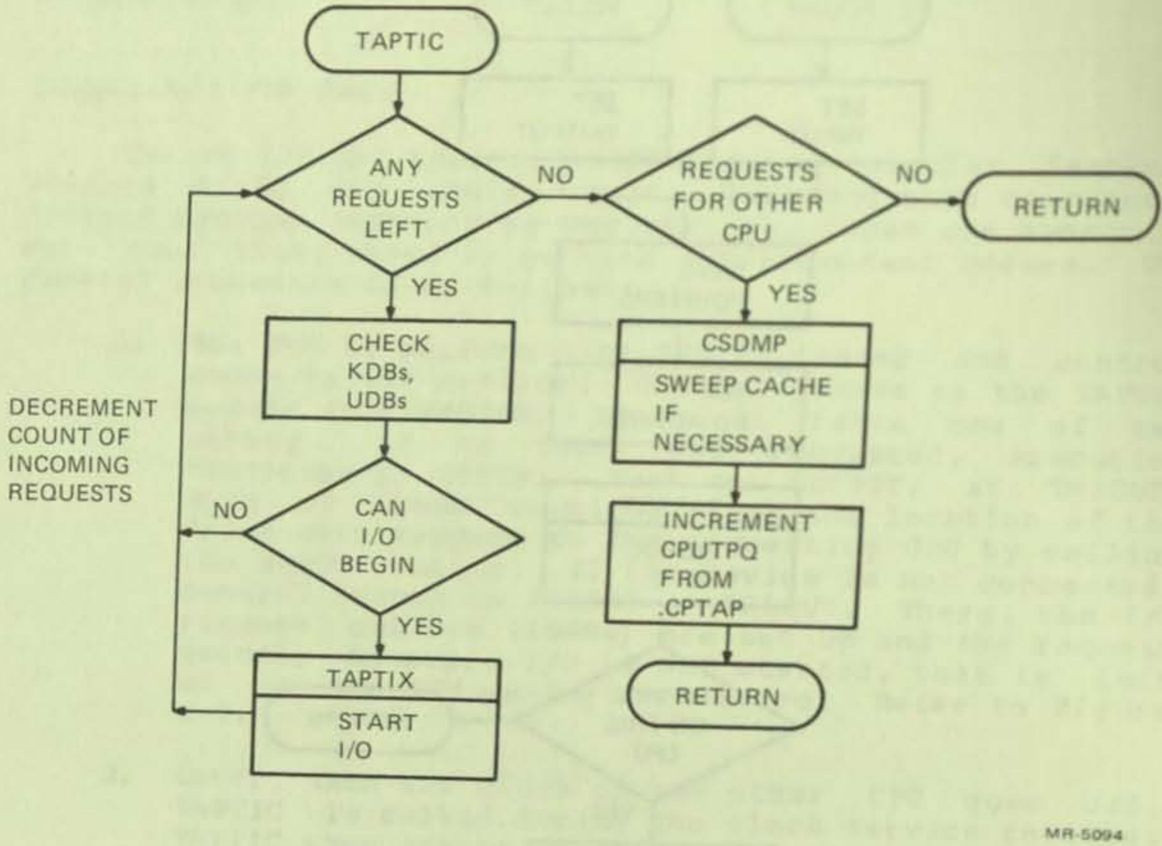


Figure 2-8 Once-A-Tick Tape Routine

Cache

Cache handling for 7.01 was made more defensive because of changes to the scheduler and the addition of queued I/O. Jobs in CPU0's cache, if scheduled to run on CPU1, could lose cached data. I/O buffers touched by one CPU can now be output on the other, possibly losing current data or current pointers; clearly, more checks were called for.

Several CPU data block locations are set aside for storing cache information. .CPCSN stores the current cache sweep serial number. It is only incremented at interrupt level by the cache interrupt service routine when the cache sweep is complete. A CPU may request a cache sweep on the other CPU using .CPCRN, which holds the cache sweep request number. A cache sweep is requested in the following fashion:

1. CPU0 decides that the cache on the other CPU must be swept so it reads .CPCSN of CPU1 and stores it in .CPCRN (of CPU1).

2. Later, during the service of the clock interrupt, CPU1 will compare .CPCSN and .CPCRN. If they are equal, a cache sweep is initiated. If they are not equal, some other process caused a cache sweep (which still serves the purpose of CPU0).

A sample cache sweep is shown in Figures 2-9 to 2-11.

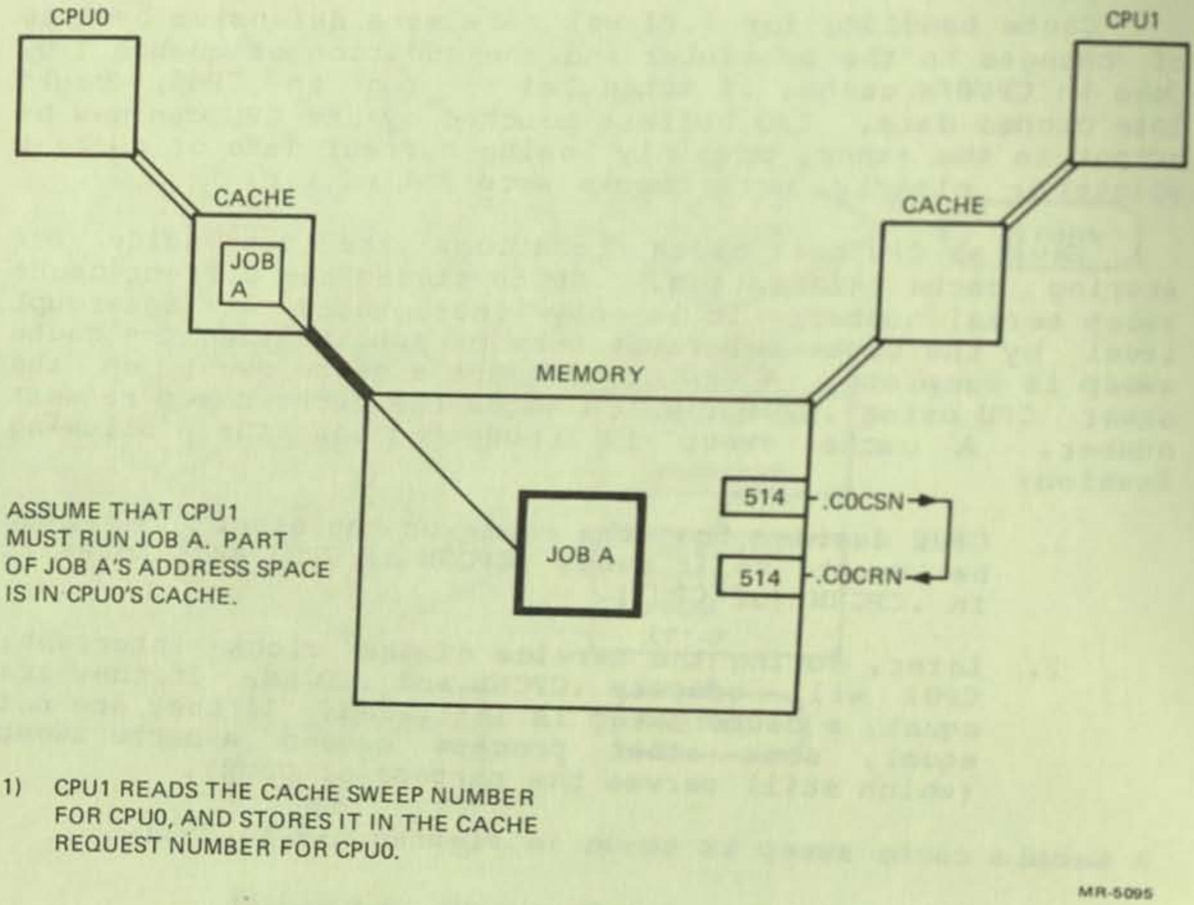
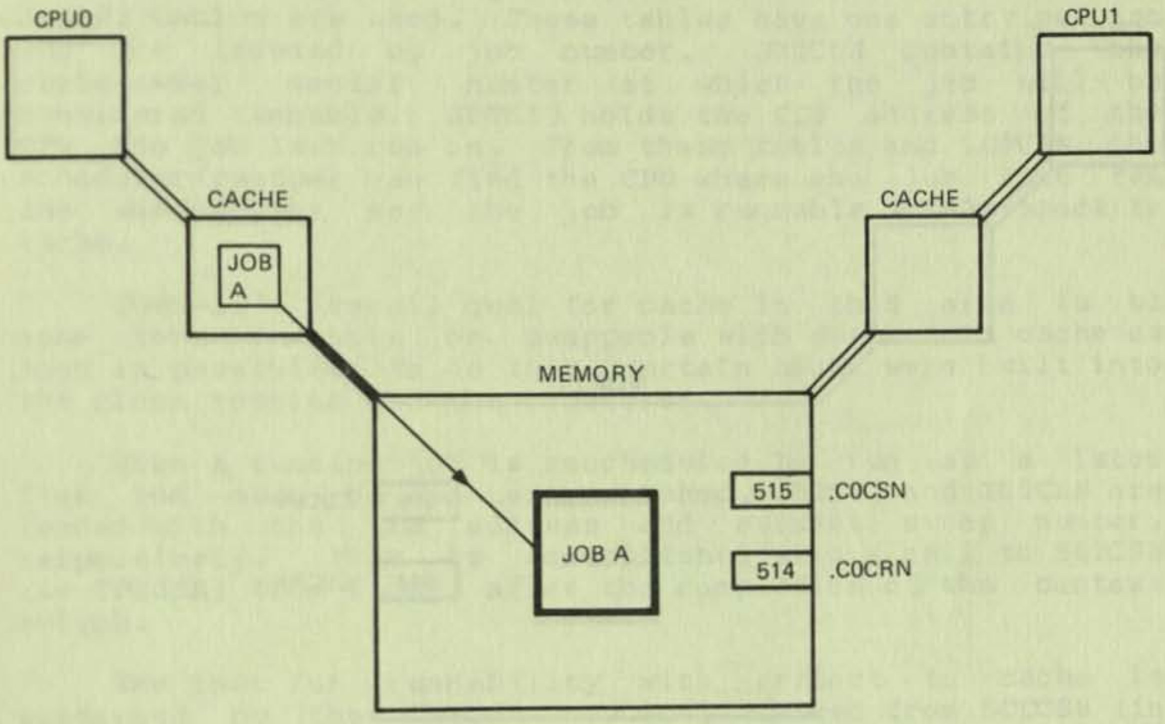


Figure 2-9 Cache Sweep Requests (1 of 3)



2) CPU0 COMPARES .COCSN AND .COCRN, FINDS THEY ARE EQUAL AND INITIATES A CACHE SWEEP. THEN .COCSN IS INCREMENTED.

Figure 2-10 Cache Sweep Requests (2 of 3)

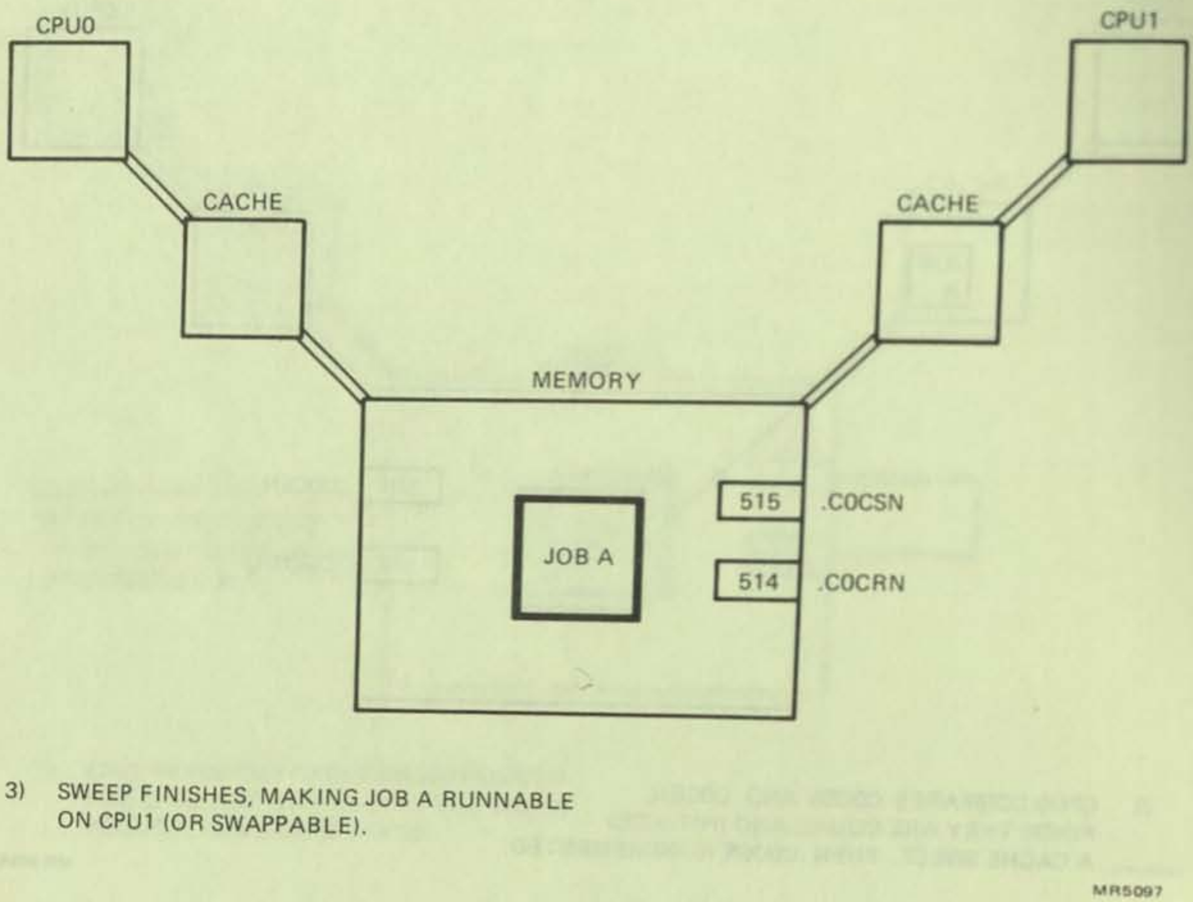


Figure 2-11 Cache Sweep Requests (3 of 3)

JOB SCHEDULING

To keep track of a job's cache status, the JBTCSN and JBTST3 tables are used. These tables have one entry per job and are indexed by job number. JBTCSN contains the cache-sweep serial number at which the job will be considered runnable. JBTST3 holds the CDB address of the CPU the job last ran on. From these tables and .CPCSN, the scheduler/swapper can find the CPU where the job last ran and whether or not the job is runnable with respect to cache.

TOPS-10's overall goal for cache in this area is to make jobs runnable or swappable with respect to cache as soon as possible. To do this, certain hooks were built into the clock routine and the scheduler.

When a running job is rescheduled to run at a later time and must be context switched, JBTST3 and JBTCSN are loaded with the CDB address and current sweep number, respectively. This is accomplished with a call to SETCSN (in CPNSER) from CLOCK1 after the completion of the context switch.

The test for runnability with respect to cache is performed by the CHKCSH routine, called from SCDCSH (in SCHED1) or from SWPCSH (in SWPSER). If, in CHKCSH, a job cannot be run or swapped, a cache sweep is requested. Refer to Figure 2-12.

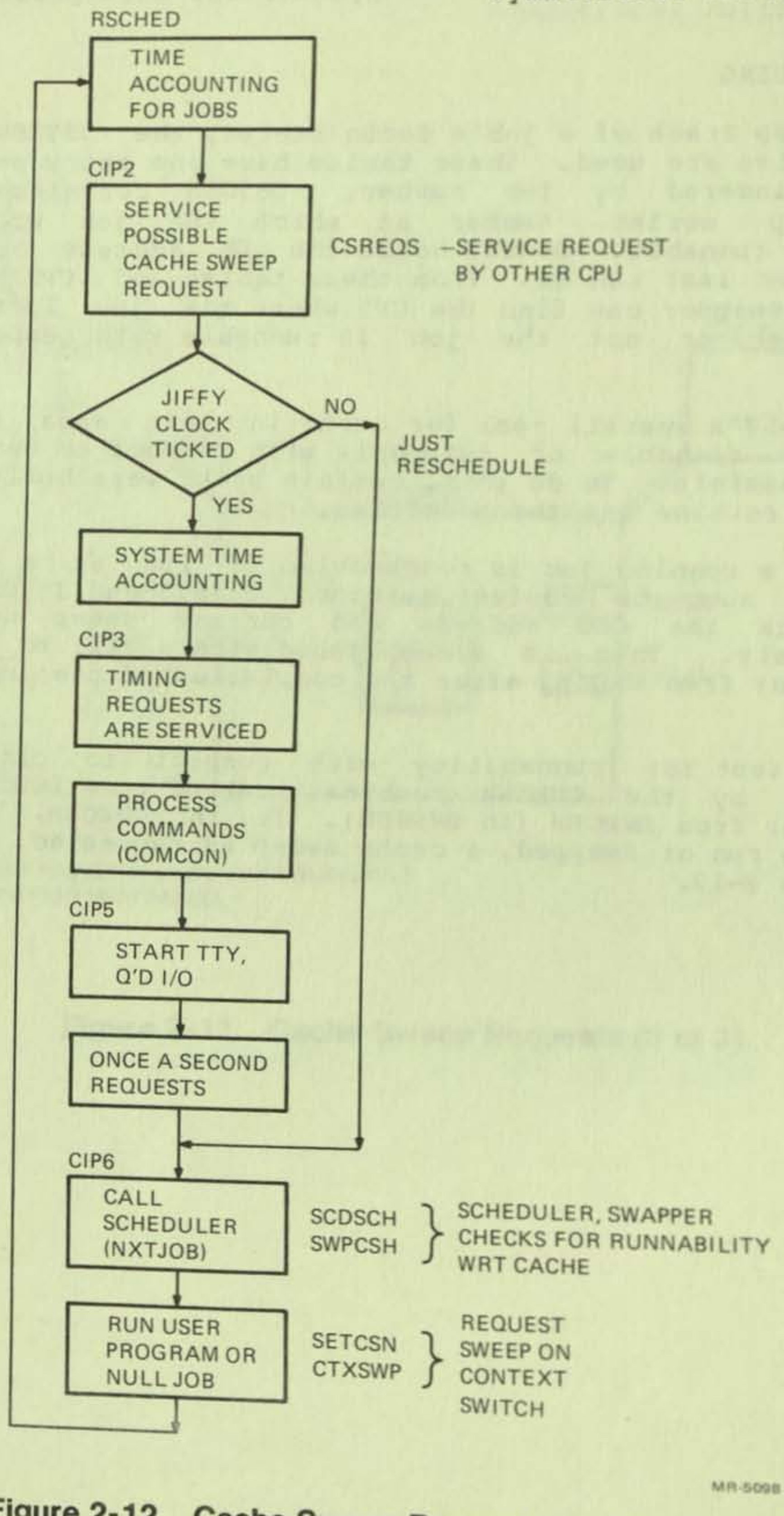


Figure 2-12 Cache Sweep Requests and the Clock Cycle

The request for a sweep is detected by the other CPU when it runs the CSREQS routine, in the clock service routine. CSREQS compares .CPCSN and .CPCRN, starting a sweep if necessary.

CSREQS has been altered to run faster. Previously, CSREQS used the CSDMP routine to sweep cache. CSDMP waited for the cache sweep to finish before continuing.

Here is the 6.03A code:

```
;ROUTINE TO INVALIDATE ENTIRE CACHE AND VALIDATE CORE
; ALWAYS RETURN CPOPJ, SAVING ALL T ACS
; LOOP IN ACS WHILE WAITING FOR FLUSH COMPLETION TO MINIMIZE
; MEMORY INTERFERENCE.

CSDMP:: PUSHJ   P,SAVT          ;SAVE T ACCUMULATORS
        DMOVE   T1,CSHWAT       ;CACHE WAITING INSTRUCTIONS
        MOVE    T3,CSHWT1       ;
        SWPUA                   ;SWEEP - UPDATE ALL PAGES,
                                ; INVALIDATE CACHE
        JSP     T4,T1           ;WAIT IN ACS
        POPJ    P,              ;RETURN

CSHWAT: CONSZ   APR,LP.CSB      ;WAIT FOR CACHE SWEEP TO
        JRST    T1              ; HAPPEN
CSHWT1: JRST    (T4)            ;RETURN TO CALLER
```

In 7.01, CSREQS does not wait for the cache sweep to finish, but continues execution of the clock cycle. Compare the code above with what follows.

```
CSREQS::MOVE    T1,.CPCSR##     ;GET REQUEST NUMBER
        CAMGE   T1,.CPCSN##     ;IS IT LESS THAN THE
                                ; CURRENT SWEEP NUMBER
        POPJ    P,              ;NO REQUESTED CACHE
                                ; SWEEP THIS TICK
        AOS     .CPCRN##        ;NO, EQUAL (HAPPENS) OR
                                ;GREATER (NEVER HAPPENS)
                                ; INCREMENT COUNT OF SWEEPS
                                ; DONE BY REQUEST
                                ;FALL INTO CTXSWP
```

```
CTXSWP::PUSH    P,T1            ;SAVE AN AC
        MOVSI   T1,(ST%LSC)     ;LOW SEGMENT CACHED
        TDNN    T1,CNFST2##     ;IF SO, NO NEED TO SWEEP
        SWPUA                   ;START SWEEP TO MAKE
                                ; JOB RUNNABLE
        JRST    TPOPJ##         ;RETURN
```

Sweep-and-go is only used for cache sweeps in a scheduling context.

I/O BUFFERS AND CACHE

When a request for I/O to a device connected to the other CPU is made, cache must be swept before the owning CPU can begin processing the request. If a sweep is not done, up-to-date information about the user buffers may be lost.

Three new words have been added to DDBs for cache on SMP systems:

1. DEVCSN - holds .CPCSN when the user makes an I/O request on another CPU

2. DEVNBF - LH equals the number of buffers swept for (OK for I/O)
   RH equals the I/O requests (not swept for)

3. DEVSBF - saved copy of DEVNBF

The specific use of each word will be explained in the following text.

Cache Handling for Output

When an OUT is issued, the current sweep number on the executing CPU is stored in DEVCSN of the DDB. This is done by calling STONBF. Any further sweep makes the buffers available to the other CPU. At the same time, the number of buffers being output is stored in the RH of DEVNBF. Refer to Figure 2-13.

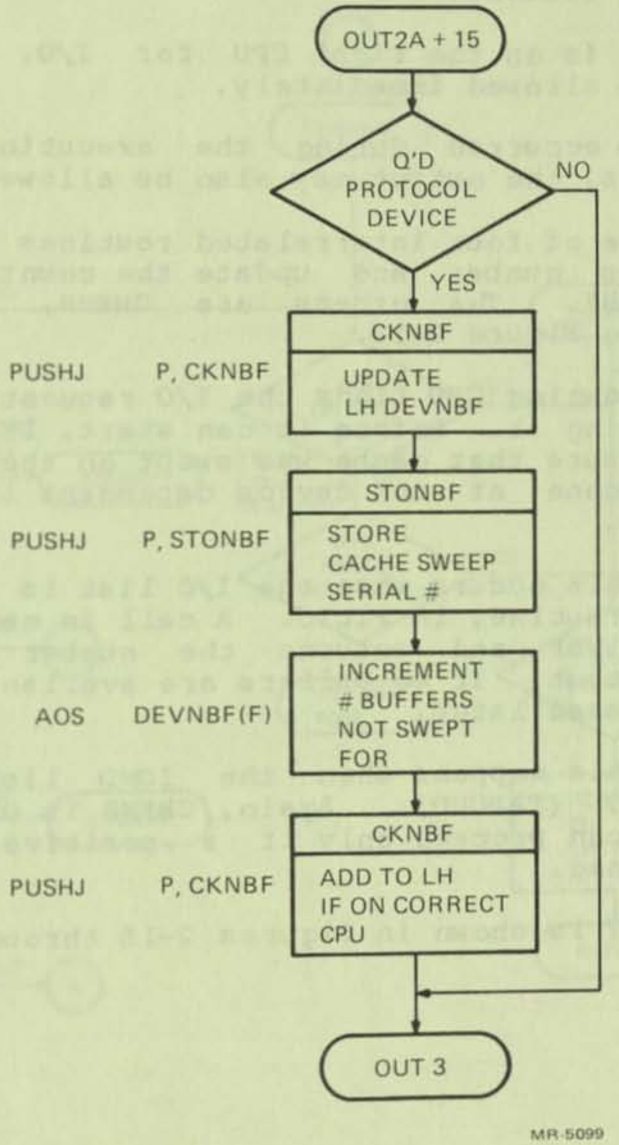


Figure 2-13 Adjusting Output Buffers for Cache

The first call to CKNBF updates DEVNBF. A call is made to CKNBF immediately after the information is stored for one of the following reasons:

1. The job is on the right CPU for I/O, the output will be allowed immediately.

2. A sweep occurred during the execution of these routines, the output may also be allowed to begin.

CKNBF is one of four interrelated routines that check the cache sweep number and update the count of uncached buffers in DEVNBF. The others are CHKNB, ADDNBF, and ADDNB1. Refer to Figure 2-14.

Later, the owning CPU finds the I/O request and tries to start processing it. Before it can start, DEVCSN must be checked to make sure that cache was swept on the other CPU. This must be done at the device dependent level, in the service routines.

For disk, this occurs when the I/O list is constructed by the SETLST routine, in FILIO. A call is made to CHKNB, which updates DEVNBF and returns the number of buffers available for output. If no buffers are available, then the request is processed later.

For tape, this happens when the IOWD list is being built in MAKLST (TAPUUO). Again, CHKNB is used to check DEVNBF. Output can proceed only if a positive number of buffers is returned.

A sample OUT is shown in Figures 2-15 through 2-18.

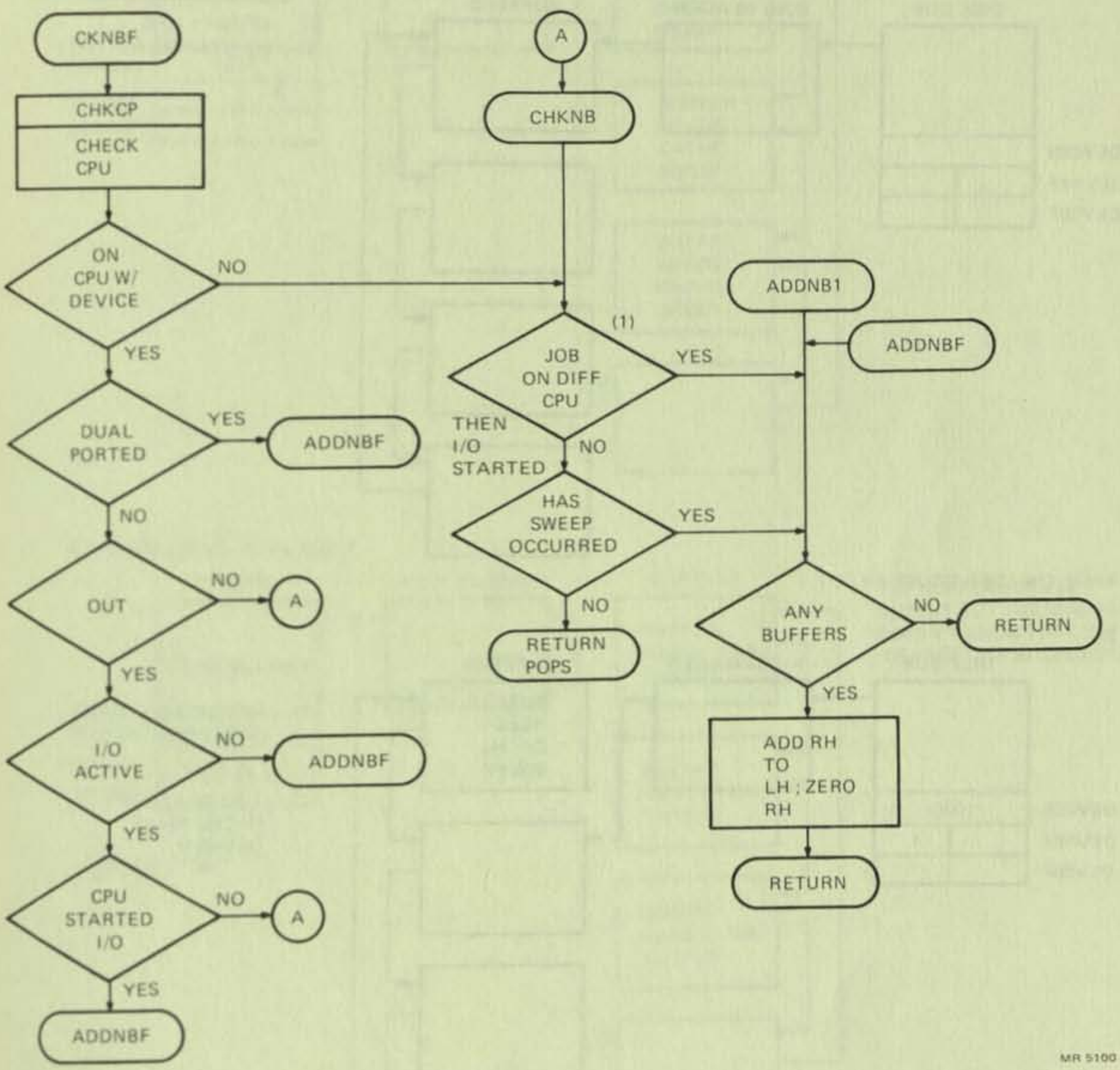


Figure 2-14 DDB Cache Words Update Routines

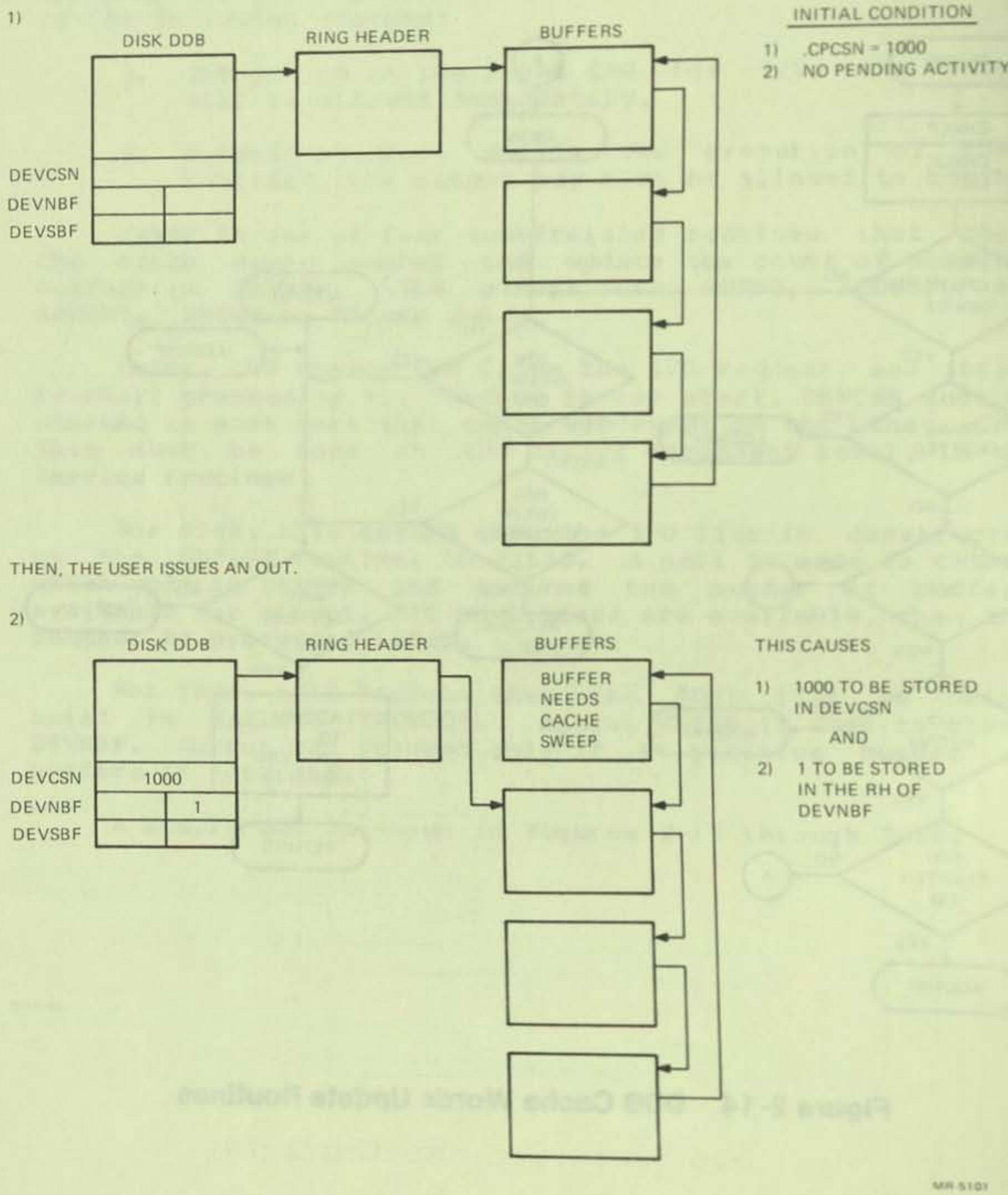
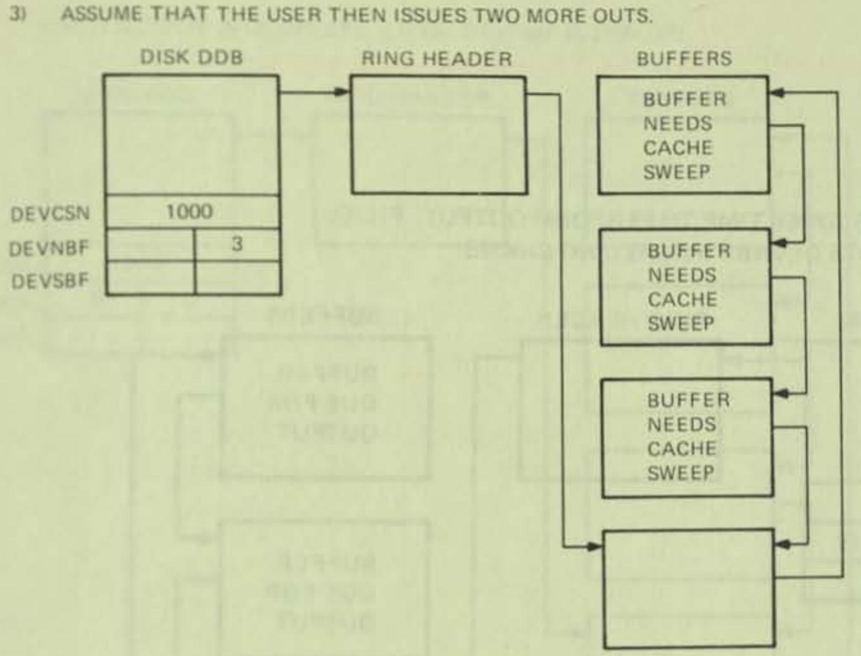


Figure 2-15 Output Cache Handling (1 of 4)



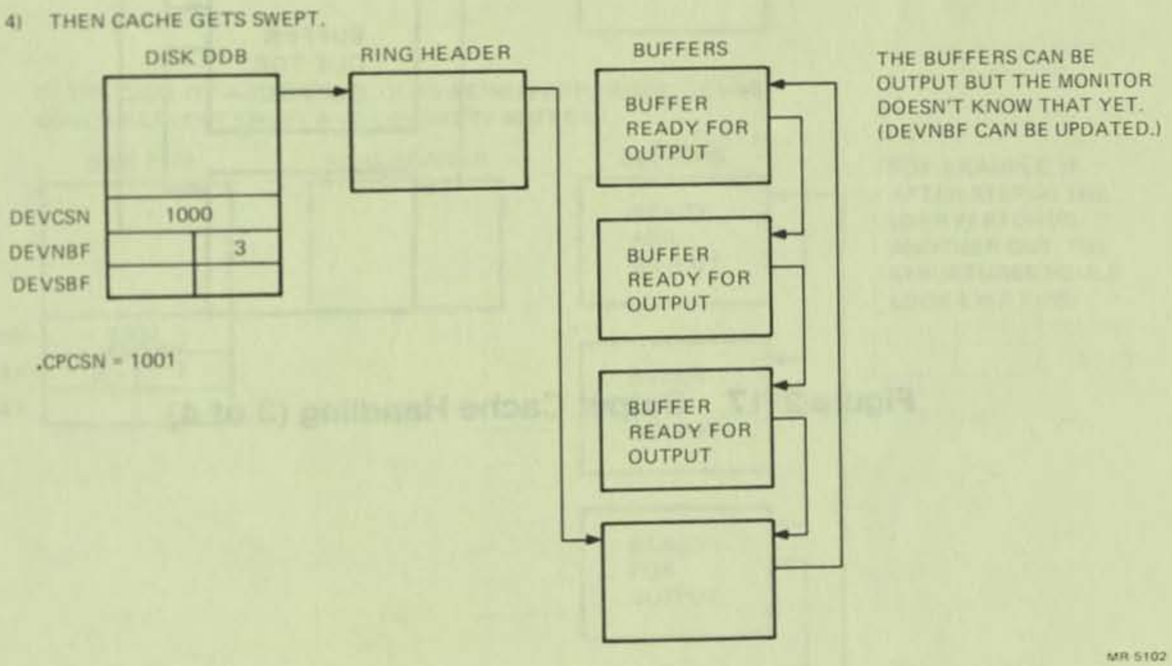


Figure 2-16 Output Cache Handling (2 of 4)

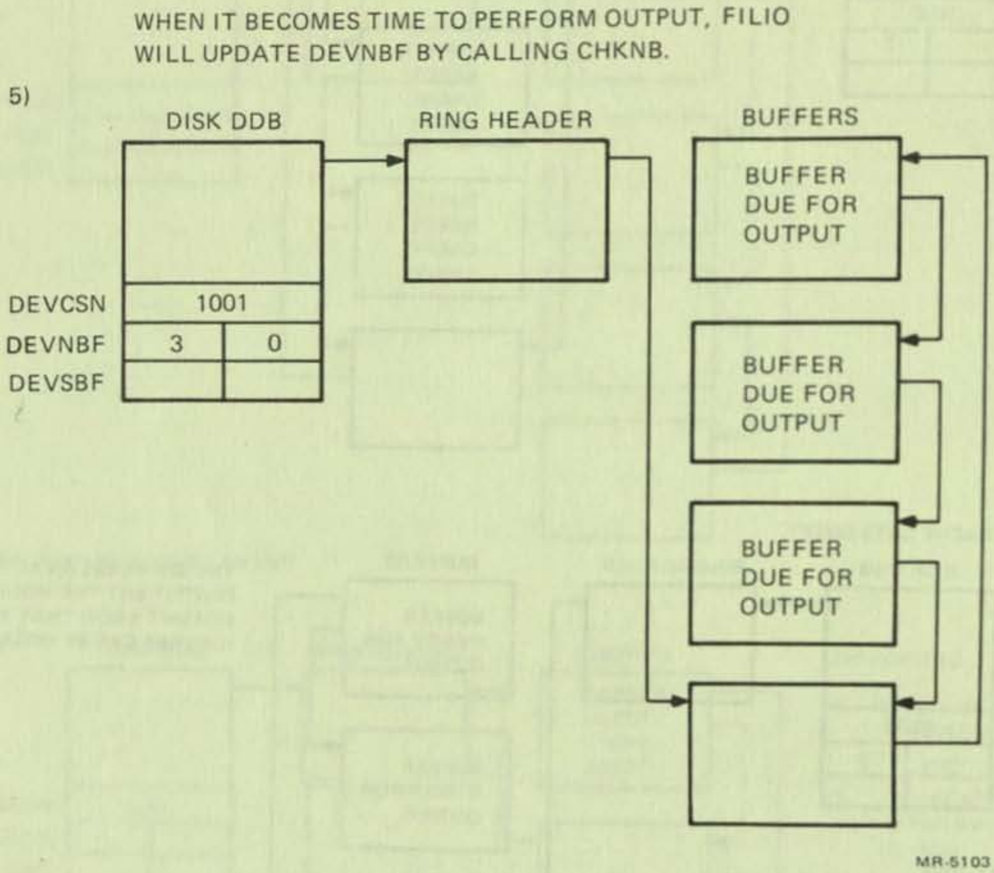


Figure 2-17 Output Cache Handling (3 of 4)

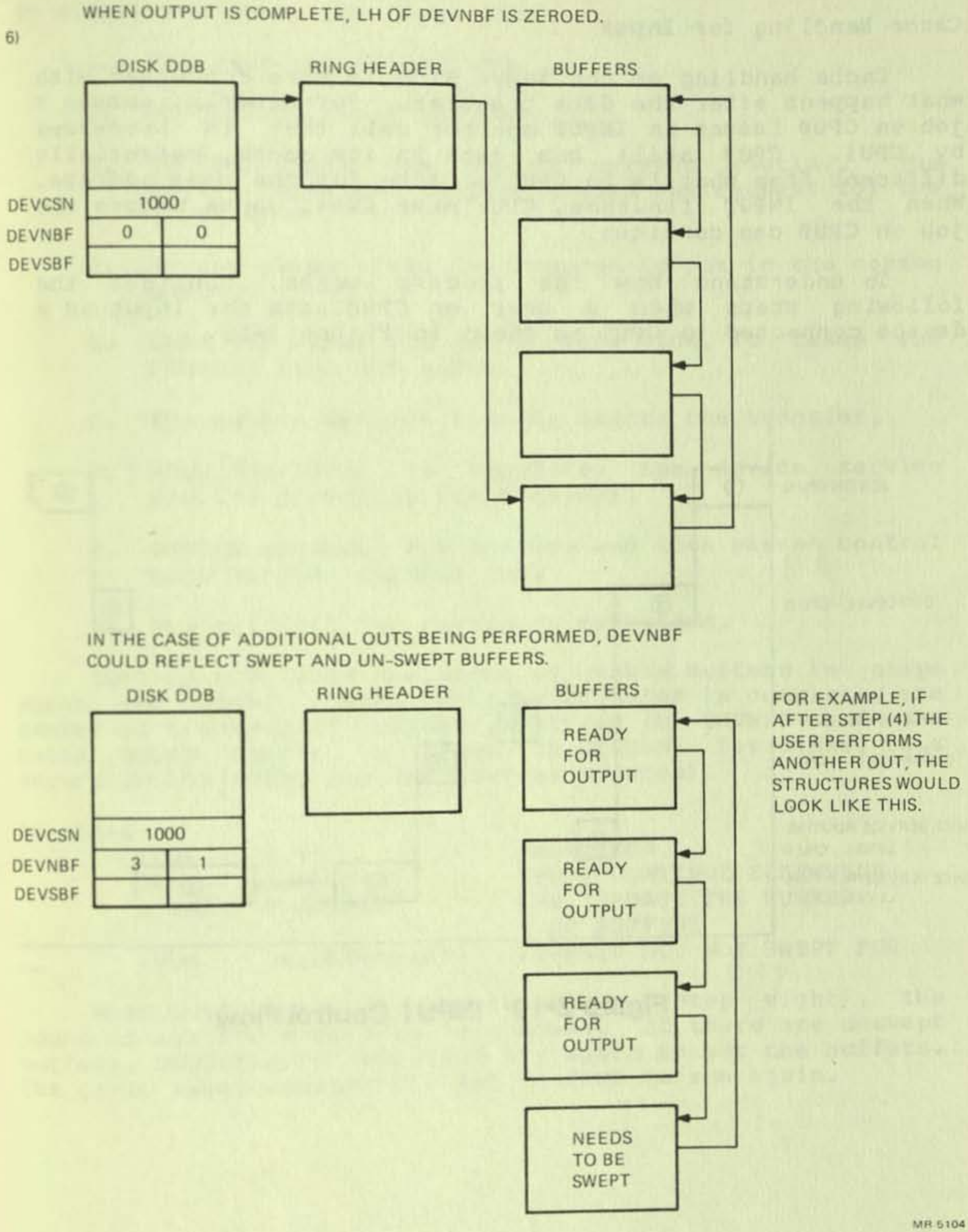


Figure 2-18 Output Cache Handling (4 of 4)

Cache Handling for Input

Cache handling on the input side is more concerned with what happens after the data transfer. For example, assume a job on CPUØ issues an INPUT monitor call that is processed by CPU1. CPUØ still has data in its cache, potentially different from what is in CPU1's cache for the same address. When the INPUT finishes, CPU1 must sweep cache before the job on CPUØ can continue.

To understand how the process works, consider the following steps when a user on CPUØ asks for input on a device connected to CPU1 as shown in Figure 2-19.

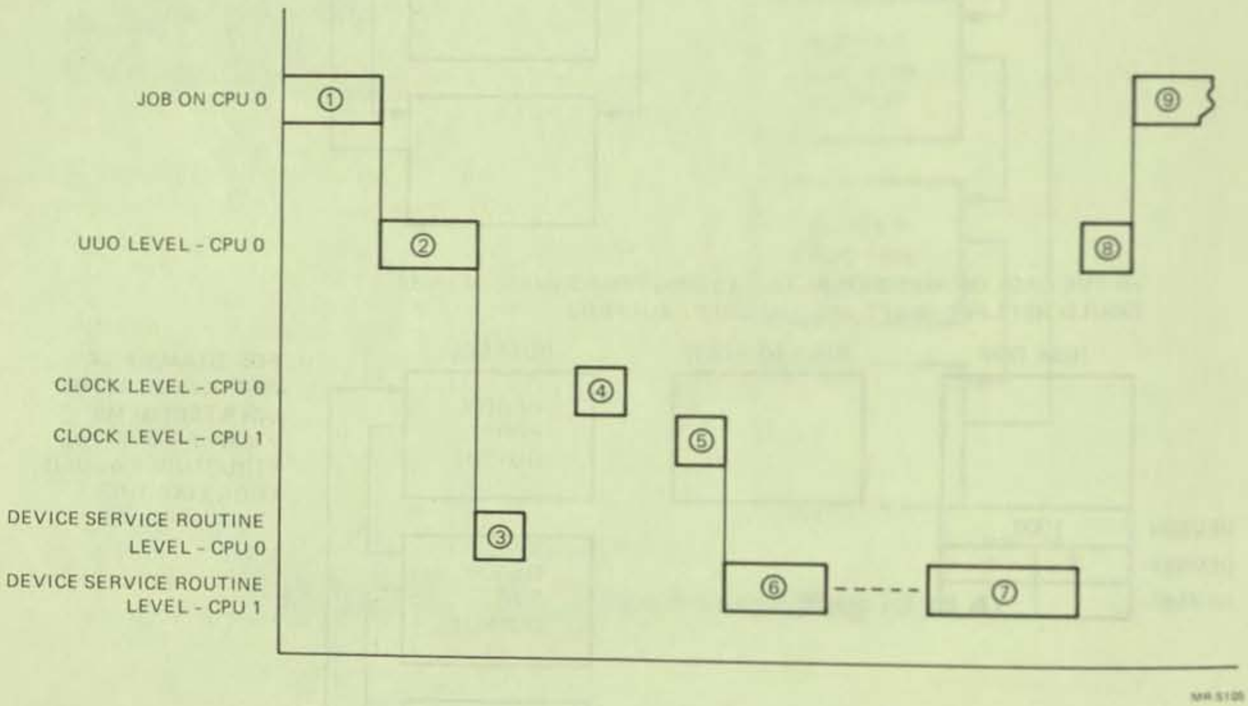


Figure 2-19 INPUT Control Flow

In simplified form, the steps are:

1. The user issues the UUO.

2. UUOCON processes the UUO.

3. UUOCON calls the device service routine. The device service routine queues the request for the other CPU.

4. At the clock tick, the request is put in the common queue.

5. When the other CPU has a clock tick, it takes the request from the queue.

6. The device service routine starts the transfer.

7. When the input is complete, the device service routine cleans up the transfer.

8. UUOCON advances the buffers and then passes control back to the original job.

9. The original job continues execution.

TOPS-10 maintains the count of usable buffers in steps seven and eight. When the data transfer is complete, the number of transferred buffers is stored in DEVNBF and the cache sweep number is stored in DEVCSN; for disks, this occurs in the BUFAD routine (buffer advance).

```
BUFADB+5
        TLNE    S,IO            ;WRITING?
        JRST    BUFAD1          ;YES, CONTINUE ELSEWHERE
        PUSHJ   P,STONBF        ;NO, UPDATE THE NUMBER
                                ; OF BUFFERS
        ADDM    P4,DEVNBF(F)    ;UPDATE NO. NOT SWEPT FOR
```

When control passes back to UUOCON (step eight), the count of available buffers is checked. If there are unswept buffers, UUOCON will sweep and try again to get the buffers. The cache sweep enables the job on CPU0 to run again.

```
INPTØZ:

PUSHJ   P,CKNBF             ;SET LH(DEVNBF) FOR ALL BUFFERS WE
                            ; CAN GET
MOVSI   T1,-1               ;IF THERE ARE NO BUFFERS FOR US TO
                            ; USE
TDNN    T1,DEVNBF(F)        ;
JRST    [SKIPN  DEVNBF(F) ;ANY FILLED BUFFERS AT ALL?
         JRST   INPTØX  ;NO. (SYSTEM ERROR?)
         PUSHJ  P,CSDMP ;YES, SWEEP FOR THEM
         JRST   INPTØZ  ;GO UPDATE LH(DEVNBF) AND TRY AGAIN
ADDM    T1,DEVNBF(F)        ;ACCOUNT FOR THE BUFFER WE ARE ABOUT
                            ; TO USE

INPTØX:

MOVEI   T1,-1(T2)           ;IN CASE USER LOOKED AT USE BITS
PUSHJ   P,KLBUFB            ;OUCHE BOTH ENDS OF BUFFER FOR HIM
```

MISCELLANEOUS CACHE CONSIDERATIONS

OUCHE is a small routine used extensively in cache handling. Its function is to get one word out of cache and into regular memory. OUCHE does this by executing no-ops on the same address in four consecutive pages to push the desired address out of cache. This saves a cache sweep in certain cases such as when UUOCON touches the buffer ring headers.

TOPS-10 will attempt to cache as much information as possible. In Version 7.01, the executive pushdown lists and parts of user funny space (specifically TMPCOR for this release) have been placed in cache. The code for caching the executive pushdown lists is in SYSINI near KIINI9:

```
MOVEI T1,CSHFIR##           ;START UP INTERRUPT LEVEL PDL'S AND
MOVEI   T2,CSHLAS##         ;STUFF CAN CACHE ALL OF THAT
PUSHJ   P,CSBRNG(P2)        ;SINCE NOT REALLY SHARED BETWEEN CPUS
```

The routine CSBRNG places all locations between CSHFIR(st) and CSHLAS(t) in cache.

COMTAB, the monitor commands table, has been moved to COMMON so that it can be cached.

CPU Doorbell

There is no hardware doorbell between the two CPUs. However, a software doorbell has been implemented so that a CPU can inform the other of:

1. The need for rescheduling, or
2. The availability of queued I/O.

The "doorbell" is a word in COMMON named DOORBL that each processor checks periodically for work that can be done. The doorbell is only checked during the execution of the null job. In essence, its function is to prevent the system from doing nothing (running the null job) when it could be doing something. If the null job sees the proper bits set in DOORBL, it will issue a WAKE. UUO. The UUO handler (at MUUO in COMMON) checks for just such an occasion and either starts queued I/O or calls the scheduler. Refer to Figure 2-2Ø.

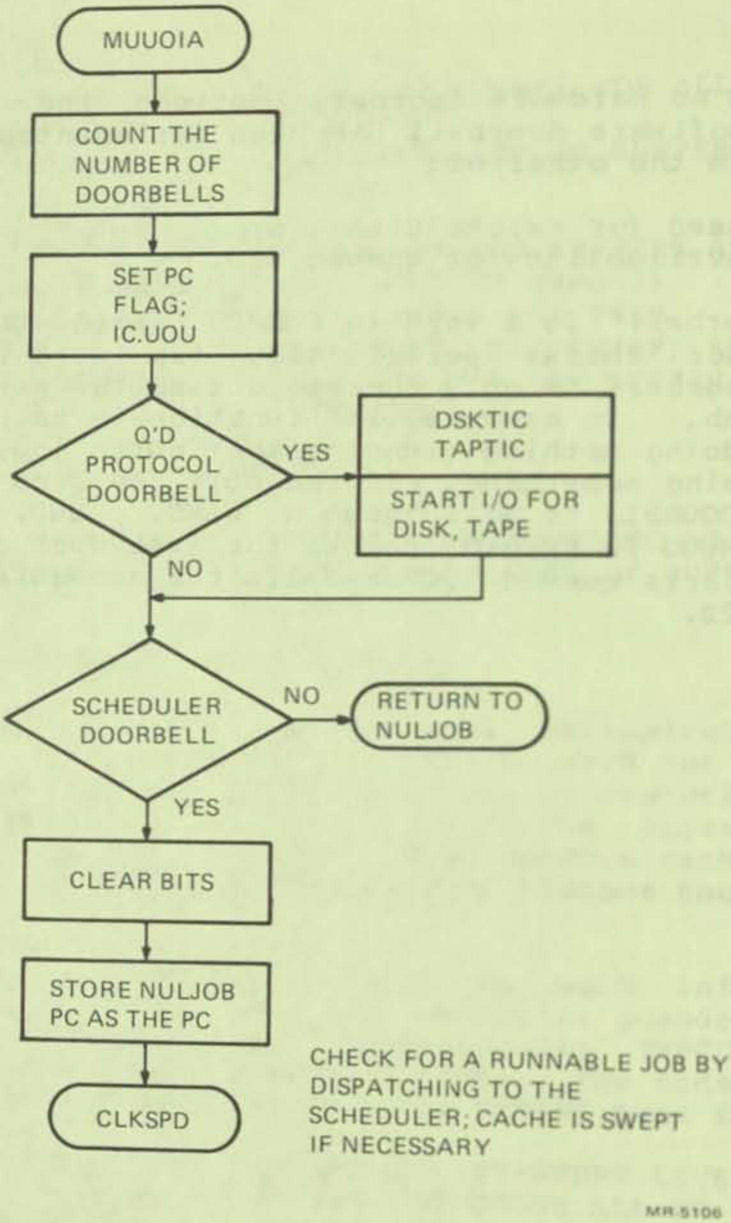


Figure 2-20 Detecting the CPU Doorbell

Each CPU has bits for its use in the doorbell word. These bits are defined in a roundabout way using the EVENTM macro in COMMON. Each CPU also has bit masks to:

1. Clear its own doorbell bits.
2. Set the doorbell bits of the other CPU.

There are five bit masks for each CPU, arranged so that the set bits for CPU1 equal the clear bits for CPU0 and vice versa.

The masks are used instead of directly setting specific bits so that the code is CPU independent. A CPU can use .CPSCS to set the scheduler bit on the other CPU without having to remember which bit is for the other CPU. Also, this structure permits the addition of more CPUs. Refer to Figures 2-21 and 2-22.

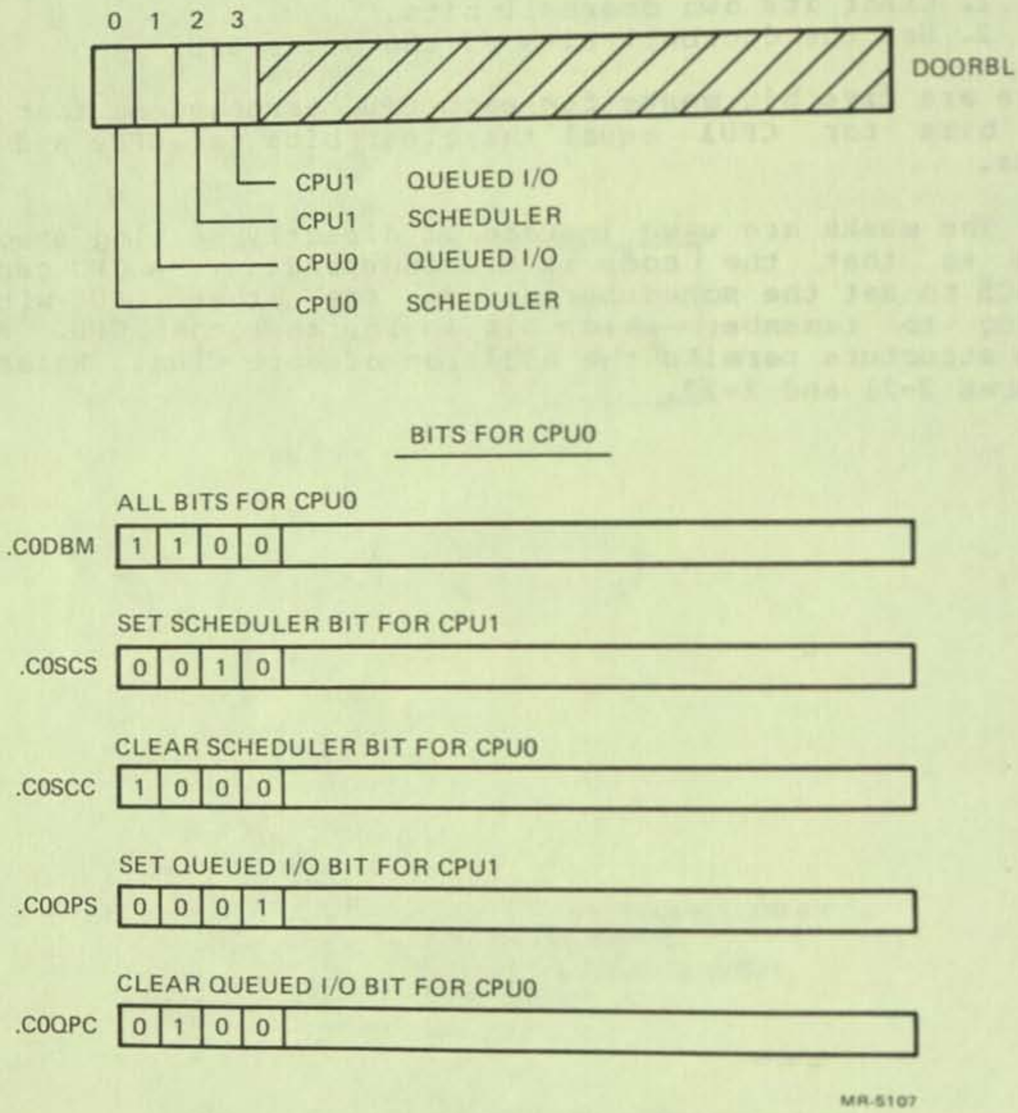


Figure 2-21 DOORBL Bits

	DISK I/O	TAPE I/O	SCHEDULER
SET	IN FILIO 1) @ PCLDSK, WHEN A REQUEST FOR ANOTHER CPU IS QUEUED AND 2) @ DSKTI6 WHEN, AFTER RECEIVING REQUESTS FROM ANOTHER CPU, REQUESTS ARE SENT OUT	IN TAPSER @ PCLTAP WHEN A REQUEST TO ANOTHER CPU IS MADE (VIA A CALL TO SETQPB)	IN CLOCK1 @ NJBTST THE BIT IS SET CALLED FROM 1) SWPSER, WHEN SWAPPING COMPLETE 2) SCNSER, WHEN TTY MONITOR COMMAND FINISHES (FROM SETRUN IN CLOCK1) 3) JOB COMES OUT OF I/O WAIT (STTIOD, STPIOD, ETC.)
CLEARED	IN FILIO @ DSKTIC, DURING THE ONCE A TICK SERVICE	IN TAPSER @ TAPTIC, DURING THE SERVICE OF REQUESTS FROM OTHER CPUS	IN SCHED1 @ NXTJB1, WHEN THE SCHEDULER IS ENTERED

MR-5108

Figure 2-22 Setting and Clearing the CPU Doorbell

Interlocks

With the reorganization of disk DDBs into funny space and the rewriting of monitor modules to be re-entrant, the possibility of the two CPUs competing for the use of the data base has been reduced, but it has not been eliminated. Simultaneous use of the scheduler could result in problems if both CPUs try to run the same job at once. An interlock has been created to prevent this situation. Before all calls to the scheduler, the LOKSCD routine is called. A call to LOKSCD will always return with the CPU owning the interlock. However, that result may not be immediate. If the other CPU owns the resource when LOKSCD is entered, the routine will loop in LOKSCD until the interlock is removed. The CPU that has a non-zero number in .CPSCD is considered the owner of the scheduler.

Another problem that developed was over the use of the core allocation routines. To solve this problem, the Memory Management (MM) resource was added. A CPU can request the use of the MM via the GETMM routine (in CPNSER).

MISCELLANEOUS MONITOR CHANGES

System Interlocks

To improve performance, the scope of certain interlocks has been reduced. Processes may not have to wait to use:

1. AU (Alter UFD) - This resource is no longer system wide. Instead, the interlock is per UFD per structure. The UFBAUJ word in the unit file block is checked to see if the AU resource for that UFD on that structure is free.

```
UPAU1: UUOLOK                          ;INTERLOCK
       SKIPLE  UFBAUJ##(P1)            ;ANYBODY OWN AU FOR
                                       ; THIS UFB?
       JRST    UPAU2                   ;YES, HAVE TO CAUSE
                                       ; THE JOB TO WAIT
       MOVE    P2,.CPJOB##             ;NO, WE OWN IT NOW
       MOVEM   P2,UFBAUJ##(P1)
       UUONLK                          ;UN-INTERLOCK
```

2. DA (Disk Allocation) - The DA resource interlock is now on a per unit basis. The code to handle this change is contained in FILIO. The unit data block is checked to see if it is in use.

```
UPDA1:  UUOLOK                          ;STOP INTERRUPTS
        SKIPL   UNIAJB##(U)             ;IS IT IN USE?
        JRST    UPDA2                   ;YES, HAVE TO WAIT
        PUSH    P,T1                    ;NO, WE OWN IT NOW
        LDB     T1,PJOBN##
        MOVEM   T1,UNIAJB##(U)          ;INDICATE JOB
                                        ; NUMBER OF OWNER
        UUONLK
```

3. MQ (Disk Monitor Buffer) - This resource has been eliminated because of the reorganization of monitor buffers into user funny space.

4. MM (Memory Management) - This is a new resource that handles core allocation. The CORE monitor command, RUN, GET, and the CORE. UUO must all contend for the use of the MM resource. Before obtaining this resource, the scheduler interlock must be obtained. If the attempt to get MM fails, the caller is responsible for trying again. The routine to obtain this resource is called GETMM and is located in CPNSER. This resource prevents competition between the CPUs when allocating memory.

5. SCNSER Interlocks - See Chapter 8 for details on the reduced scope of SCNSER interlocks.

PSISER Changes

PSISER has been totally rewritten to improve its reliability and simplify its operation. In Version 6.03A it operated both at interrupt and at UUO level, causing race conditions and context problems. The new code works only at UUO level to avoid the problems.

The functions and calling sequences of all the software interrupt-related monitor calls have not changed. There is a new monitor call, PIJBI, that allows cross job interrupts. It is discussed in Chapter 6.

Changed Conventions

The following conventions have been changed:

1. P4 no longer points to CDB.

2. .CØxxx or .C1xxx is not used to represent the CPU status block very often; instead, .CPxxx will point at the entries in the CSB of the currently running CPU.

3. R is no longer needed to reference JOBDAT.

PART THREE

KL10 SERVICE ENHANCEMENT PROJECT

Chapter 3

KL10 SERVICE ENHANCEMENT PROJECT

The KL1Ø Service Enhancement Project (SEP) improves the ability of TOPS-1Ø to detect, report, and recover from software and hardware errors. The changes that were made fall into three main categories:

1. Updating the Software Notebooks - to describe more completely what happens on errors and how the monitor recovers from them. Stopcodes are being documented more completely, including probable causes and monitor locations that may be useful during debugging.

2. Validation - to make sure the monitor correctly detects errors and reports the results. Existing error routines were validated by the deliberate insertion of known faults (both hardware and software) to test TOPS-1Ø's diagnosis and corrective action.

3. New Features - to include more information gathering and reporting as well as new algorithms to recover from nonfatal errors. The remainder of the chapter is devoted to those new features.

ERROR REPORTING

Stopcodes

The code having to do with stopcodes has been rewritten. Some of the changes are listed below.

1. The stopcode message is printed on the console terminal of the CPU on which the stopcode is detected. If the stopcode causes a monitor reload, the boot CPU will still load BOOTS.

2. When any CPU gets a stopcode, all other CPUs will take a CPU and device status block dump, and wait for the CPU to finish the stopcode processing. The location DIELOK is used to indicate that a CPU is processing a stopcode.

3. The DIE and REBOOT routines have been made symmetrical so that either CPU can run these routines. However, only the boot CPU will load BOOTS.

4. A new type of stopcode called CPU has been defined. This stopcode will act as a STOP stopcode either on a single CPU system, on the only processor running on a multiple CPU system, or if DF.CP1 is set in DEBUGF. In all other cases, it only stops the CPU in which the error was detected. Bit DF.CP1 has been redefined to mean "stop the system on a CPU stopcode on any CPU."

5. Most of the code for stopcodes in CPNSER has been removed since REBOOT is now symmetrical. The CP1CRS routine is called with the ACs and machine state saved to set up the AC loop.

When a stopcode occurs, TOPS-1Ø now prints additional information on the CTY along with the stopcode name and the date/time. For example:

```
?CPU1 MONITOR ERROR. STOPCODE NAME IS IME

CPU STATUS BLOCK AT 13-JUN-79 7:26:34

APRID = ØØØ215,,342Ø22
ERA = 2ØØØØØ,,Ø4Ø445
CONI APR, = ØØ776Ø,,ØØØØØ3
CONI PI, = ØØØØØØ,,ØØØ777
CONI PAG, = ØØØØØØ,,62ØØØ2
DATAI PAG, = 7ØØ1ØØ,,ØØØØØ3
AR ARX Data Word = 371Ø4Ø,,ØØØØ2Ø
IO Page Fail Word = ØØØØØØ,,ØØØØØØ

SBUS Diags:

CNTRLR FNC Ø              FNC 1
ØØØØØ4 ØØ624Ø,,Ø4Ø542 ØØØ2ØØ,,ØØØØØØ
```

The CPU status information shown above is stored along with other data in the new subtables of the CPU data block. It is placed there by the CPUSTS routine (called by ERRCON through the RCDSTB routine in COMMON). The following is a complete list of the information that is stored.

Word	Name	Data
0	.CPAPD	APRID
1	.CPACN	CONI APR,
2	.CPPIC	CONI PI,
3	.CPPGD	DATAI PAG,
4	.CPPGC	CONI PAG,
5-10	.CPUPO	UPT LOCS 424-427
11	.CPERA	RDERA
12-21	.CPRHC	CONI RH20, FOR ALL RH20s
22-25	.CPDTC	CONI DTEN,
26-65	.CPEP0	EPT LOCATIONS 0-37
66-125	.CPEP1	EPT LOCATIONS 140-177
126-131	.CPUP1	UPT LOCATIONS 500-503
132-136	.CP6	AC BLOCK 6 REGS 0-3 AND 12
137-141	.CP7	AC BLOCK 7 REGS 0-2
142-211	.CPSBD	SBUS DIAGNOSTIC DATA

In addition to being a part of the CPU status block subtable, the SBus diag block (142-211) is also a CDB subtable, pointed to by .CPSDP.

After the CPU status information is stored, TOPS-10 also stores device status information in the CDB using the DVCSTS routine (also called by RCDSTB). CONIs are used to gather the information.

```
;DEVICE STATUS BLOCK ENTRY FOR DEVICES ON THIS CPU
; THIS IS A CDB SUBTABLE. THE ORDER OF THESE ENTRIES MUST
; EXACTLY MATCH THE ORDER OF THE INSTRUCTIONS IN DVCSTS
; THAT FILL THE ENTRIES.

        V       (DVS,N,,,0)
        V       (TMR,N,Y)           ;INTERVAL TIMER
        V       (MTR,N,Y)           ;METER
        V       (TTY,N,Y)
```

```
V        (PRA,N,Y)
V        (PPA,N,Y)
V        (DLS,N,Y)        ;DATA LINE SCANNER
V        (DAC,N,Y)
V        (DAS,N,Y)
V        (CRA,N,Y)        ;CARD READER 0
V        (LPT,N,Y)        ;LINE PRINTER 0
V        (PLA,N,Y)        ;PLOTTER 0
V        (TMS,N,Y)
V        (TMC,N,Y)
V        (DX1,N,Y)
V        (DSK,N,Y)        ;DISK DEVICES
V        (FH2,N,Y)        ;    .
V        (FSD,N,Y)        ;    .
V        (FS2,N,Y)        ;    .
V        (FS3,N,Y)
V        (DPC,N,Y)
V        (DP2,N,Y)
V        (DP3,N,Y)
V        (DP4,N,Y)
V        (2DS,N,Y)
V        (2DC,N,Y)
V        (DLC,N,Y)
V        (DLB,N,Y)
V        (DC2,N,Y)
V        (DB2,N,Y)
V        (CDP,N,Y)        ;CARD PUNCH
V        (CRB,N,Y)        ;CARD READER 1
V        (LPB,N,Y)        ;LINE PRINTER 1
V        (LPC,N,Y)        ;LINE PRINTER 2
V        (PLB,N,Y)        ;PLOTTER 1
V        (DAK,N,Y)        ;ADDRESS BREAK CONDITIONS
V        (DDK,N,Y)
V        (DH2,N,Y)
V        (DFS,N,Y)        ;MAGNETIC TAPE DEVICES
V        (DS2,N,Y)
V        (DS3,N,Y)
V        (DDP,N,Y)
V        (DD2,N,Y)
V        (DD3,N,Y)
V        (DD4,N,Y)
V        (DDC,N,Y)
V        (DDB,N,Y)
V        (D2C,N,Y)
V        (D2B,N,Y)
```

Parity Error Type-Out Reformatting

The parity error reporting section has been substantially rewritten. This does not affect the procedure by which an operator responds, but it provides new information (and hence, new printouts) on the CTY and new entries in the SYSERR file. In some cases, TOPS-1Ø will retry the error and report the success/failure of that attempt. Error types have been broken down into five types:

1 - CPU Parity/NXM Traps
2 - CPU Page Table Parity Traps
3 - CPU Interrupts
4 - Channel Errors
5 - Memory Scans

In the case of nonrecoverable memory parity errors, the monitor still halts immediately with one of the following messages:

1. ?NON-RECOVERABLE MEMORY PARITY ERROR IN MONITOR

   [CPU HALT]

2. ?NON-EXISTENT MEMORY DETECTED IN MONITOR

   [CPU HALT]

CPU PARITY/NONEXISTENT MEMORY TRAPS (KSSER, KLSER)

This error relates only to the KL1Ø and the KS1Ø because the KI1Ø has no trap hardware. A CPU parity or NXM trap is initially handled as a page fault trap. TOPS-1Ø, in the module KLSER, separates these errors from regular page faults by the page fail code (36 or 37 for parity/NXM traps). Error processing is handled within KLSER, which prints the following messages.

1. KL10

When the error first occurs, the following is printed on the CTY:

```
***************
CPUx AR/ARX PARITY TRAP AT <EXEC or USER> PC xxxxxx ON
DD-MMM-YY HH:MM:SS
JOB xx[nnnnnn] WAS RUNNING
PAGE FAIL WORD = xxxxxx,,xxxxxx
MAPPED PAGE FAIL ADDRESS = xxxxxx,,xxxxxx
INCORRECT CONTENTS = xxxxxx,,xxxxxx
CONI PI, = xxxxxx,,xxxxxx
```

First, a cache sweep is performed to get data back into regular memory. If errors occur during the sweep, this message is printed:

```
MB PARITY ERROR OCCURRED DURING CACHE
SWEEP PRIOR TO RETRY ATTEMPT.
ERA = xxxxxx,xxxxxx
```

TOPS-10 then attempts to fix the error.
If the original error is corrected, the following is printed:

```
RETRY SUCCEEDED! CORRECT CONTENTS = xxxxxx,,xxxxxx
***************
```

In the case of nonrecoverable cache errors, this will be printed:

```
THREE NON-RECOVERABLE CACHE PARITY ERRORS HAVE OCCURRED
SINCE PROCESSOR STARTED.
CACHE HAS BEEN TURNED OFF.
```

If the retries fail, the bad location is zeroed:

```
RETRIES UNSUCCESSFUL. OFFENDING LOCATION ZEROED.
```

If a monitor page was in question, the monitor page is replaced and the following message printed:

```
REPLACED FROM DISK PHYSICAL MONITOR PAGE xxxx
```

The page is set off-line if errors persist:

```
LOCATION STILL BAD AFTER ZEROING.
SETTING OFF-LINE PHYSICAL PAGE xxxx
***************
```

Here is an example:

```
****************
* CPUØ AR/ARX PARITY TRAP AT USER PC ØØ1234 ON 12-APR-8Ø 12:33:44
* JOB 12 WAS RUNNING
* PAGE FAIL WORD = ØØØØØØ,,ØØØ111
* MAPPED PAGE FAIL ADDRESS = ØØØØØØ,,ØØØØØØ
* INCORRECT CONTENTS = 777777,,777777
* CONI PI, = ØØØØØØ,,4ØØØ77
RETRY SUCCEEDED! CORRECT CONTENTS = ØØØØØØ,,ØØØØ17
****************
```

2. KS1Ø

Reports for the KS1Ø are similar to those of the KL1Ø, but with no cache information.

```
****************
CPUx <PARITY or NON-EXISTENT MEMORY> TRAP AT <EXEC or USER> PC
                            xxxxxx ON DD-MMM-YY HH:MM:SS

JOB xx[nnnnnn] WAS RUNNING
PAGE FAIL WORD = xxxxxx,,xxxxxx
MAPPED PAGE FAIL ADDRESS = xxxxxx,,xxxxxx
INCORRECT CONTENTS = xxxxxx,,xxxxxx
CONI PI, = xxxxxx,,xxxxxx
```

and then either of the following responses may occur:

```
a. RETRY SUCCEEDED! CORRECT CONTENTS = xxxxxx,,xxxxxx
   ****************

b. RETRIES UNSUCCESSFUL. OFFENDING LOCATION ZEROED.
```

Several printouts may follow unsuccessful retries:

```
a. REPLACED FROM DISK PHYSICAL MONITOR PAGE xxxx
   ****************

b. LOCATION STILL BAD AFTER ZEROING.
   SETTING OFF-LINE PHYSICAL PAGE xxxx
   ****************
```

CPU PAGE TABLE PARITY TRAPS (KSSER, KLSER)

The testing for a page table parity trap is done whenever a page fault (page fault code 25) occurs for the EPT or UPT. These errors will occur for the KL1Ø and KS1Ø only; the KI1Ø has no trap hardware.

```
***************
CPUx PAGE TABLE PARITY TRAP AT <EXEC or USER> PC xxxxxx ON
DD-MM-YY HH:MM:SS
PAGE FAIL WORD = xxxxxx,,xxxxxx
CONI PI, = xxxxxx,,xxxxxx
***************
```

CPU INTERRUPTS (CLOCK1)

The parity/NXM interrupts are used only on the KI and KL processors. They are of no significance on the KS and hence are not used.

1. KI1Ø

```
***************
CPUx <PARITY ERROR or NON-EXISTENT MEMORY> INTERRUPT
AT <EXEC or USER> PC xxxxxx ON DD-MM-YY HH:MM:SS
JOB yyy WAS RUNNING
CONI APR, = xxxxxx,,xxxxxx
CONI PI, = xxxxxx,,xxxxxx
***************
```

2. KL1Ø

With the KL1Ø, the error can be broken down further to find the specific cause:

```
***************
CPUx <PARITY ERROR or NON-EXISTENT MEMORY> INTERRUPT
AT <EXEC or USER> PC xxxxxx ON DD-MMM-YY HH:MM:SS
CONI APR, = xxxxxx,,xxxxxx
CONI PI, = xxxxxx,,xxxxxx
ERA = xxxxxx,,xxxxxx
ERROR INVOKED BY A
```

The error could have been caused by a variety of problems, including:

a. CACHE WRITE-BACK FORCED BY A SWEEP INSTRUCTION.

b. CHANNEL STATUS WORD WRITE.

c. CHANNEL DATA WORD WRITE.

d. CHANNEL READ FROM MEMORY.

e. CHANNEL READ FROM CACHE.

f. CPU WRITE TO MEMORY (NOT CACHE).

g. CACHE WRITE-BACK FORCED BY A CPU WRITE.

h. CPU READ OR PAGE REFILL FROM MEMORY.

i. PAGE REFILL FROM CACHE.

In any case, the following additional information will be displayed:

```
SBUS DIAGS:
CNTRLR FNC Ø         FNC 1
xxxxxx xxxxxx,,xxxxxx xxxxxx,,xxxxxx
***************
```

CHANNEL ERRORS (ERRCON)

This report occurs on all hard and soft channel data (KI1Ø and KL1Ø only).

```
***************
CPUx CHANNEL <MEMORY PARITY or NON-EXISTENT MEMORY> ERROR ON
                                           DD-MMM-YY HH:MM:SS

DEVICE IN USE IS nnnnnn
CHANNEL TYPE IS <DF1Ø or DK1Ø-C or DX1Ø or RH2Ø>
TERMINATION CHANNEL PROGRAM ADDRESS = xxxxxx,,xxxxxx
TERMINATION DATA TRANSFER ADDRESS = xxxxxx,,xxxxxx
LAST THREE CHANNEL COMMANDS EXECUTED ARE:
```

There are two possible typeouts at this point:

```
1.    xxxxxx,,xxxxxx
      xxxxxx,,xxxxxx
      xxxxxx,,xxxxxx
   ***************

2. ** INDETERMINATE **
   ***************
```

MEMORY SCANS (ERRCON)

TOPS-1Ø initiates a memory scan to check for memory parity errors or nonexistent memory. It is called at APR interrupt level if there appears to be a serious error and PIs are in progress. When the scan is initiated, the following message is printed:

```
***************
<MEMORY PARITY or NON-EXISTENT MEMORY> SCAN INITIATED BY
<CPUx or CHANNEL x> ON CPUx on DD-MMM-YY HH:MM:SS
```

There are three possible endings to this message. They are:

```
1. NOTHING WAS FOUND.
   ***************

2. PARITY ERRORS DETECTED:
     AT xxxxxx (PHYS.), CONTENTS = xxxxxx,,xxxxxx,
     ERA = xxxxxx,,xxxxxx
   ***************

3. NON-EXISTENT MEMORY DETECTED:
     AT xxxxxxx (PHYS.)
   ***************
```

MONITOR DUMPS

MONBTS

MONBTS is a monitor-resident BOOTS that allows continuable stopcode dumps. A continuable stopcode dump occurs when system operation is suspended, a dump taken, and the monitor then continued. JOB or DEBUG stopcodes on single-processor systems and HALT or CPU stopcodes on SMP systems causes continuable stopcode dumps. MONBTS only supports EXE files, RPØ4/Ø5/Ø6 disk drives, and RH1Ø/11/2Ø controllers. It is capable of dumping memory very quickly (seven seconds to dump one megaword of memory on KL1Øs with RH2Øs and RPØ6s). Timing tests indicate that it takes about 2Ø seconds from the first character of the stopcode message to the "DECsystem-1Ø continued" message. MONBTS also replaces monitor high-segment pages on parity errors.

When the system detects and recovers from a continuable stopcode, the user sees the following on his/her terminal:

```
DECsystem-1Ø not running
DECsystem-1Ø continued
```

Network terminals will not see these messages; the system just does not respond. The operator sees this message on CTY:

```
[Dumping on Str:CRASH.EXE[1,4]]
```

And the following message appears on a reload:

```
[Loading from Str:SYSTEM.EXE[1,4]]
```

The MONBTS algorithms are based on the concept of a system dump list (SDL). The SDL is an ordered list of file structures on which dumps are written and from which monitors are loaded. The SDL is built at ONCE-only time from HOMSDL in the home block of each pack and stored in an in-core data base pointed to by SDLTAB. Each entry in SDLTAB contains four or more words:

```
  Ø - Structure name in SIXBIT
  1 - Pointer to the MFD
  2 - Pointer to CRASH.EXE
3-N - One word for each logical unit in the structure
```

The SDL allows MONBTS to find the monitor in as few as six disk reads and to set up to dump in as few as two disk reads. The only restriction is that MONBTS will only allow a dump to be written on CRASH.EXE[1,4]. Operations to structures not in the SDL are permitted but will result in performance degradation.

The position of a file structure in the system dump list is defined in a manner similar to the system search list.

Normally, MONBTS reads all its commands from BOOTXT (in COMMON). However, in two cases (a shutdown using KSYS or several quick crashes) the operator can type in a command. When the MONBTS prompt is printed on CTY, a command of the following form can be entered:

```
MONBTS> dev:file.EXE[path]/switches
```

The available switches are:

- /LOAD - load but do not start the monitor
- /GO:n - load and start the monitor at location n (default is .JBSA in the file being loaded)
- /DUMP - dump the monitor on the specified file
- /FORCE - ignore the in-core SDL data base and do exhaustive searches for units and files. This switch should only be necessary if the data base has been clobbered. MONBTS automatically reverts to this mode if the dev: is not in the SDL.

Default for dumping is:

```
SDL:CRASH.EXE[1,4]
```

Default for reload is:

```
SDL:SYSTEM.EXE[1,4]
```

In either case, DSK refers to all structures in the SDL. Defaults are used in all cases if the operator does not respond in a given period of time (about 32 seconds) or if there is no operator on duty (SCHED 4ØØ).

MONBTS has been written to provide clearly understandable error messages when errors occur. The error messages are:

?ONLY DEVICE MAY BE SPECIFIED FOR DUMP

?UNABLE TO DUMP ON ANY STRUCTURE IN THE SYSTEM DUMP LIST

%FILE NOT FOUND

%UNPROCESSED DUMP ON <structure-name>

%UNEXPECTED END-OF-FILE ON <structure-name>

%I/O ERROR ON <structure-name>

?NO MONITOR FOUND

```
?PAGE FAIL TRAP
PFW = xxxxxx,,xxxxxx
PF PC = xxxxxx
CONI APR, = XXXXXX,,XXXXXX
RDERA = XXXXXX,,XXXXXX
```

SDLCNV UTILITY

SDLCNV is a utility used in the conversion from the old style BOOTS to the new monitor resident boots, MONBTS. It is intended to be used ONLY as a conversion utility and is NOT to be used as a regular tool. SDLCNV allows the user to extend CRASH.EXE[1,4] on a specified structure to allow MONBTS to dump all of core plus a 4-block .EXE directory. Old versions of ONCE/TWICE did not allocate enough space to dump all of core plus the EXE directory.

To extend CRASH.EXE[1,4] on a specified structure, run SDLCNV:

.R SDLCNV

STRUCTURE ON WHICH TO EXTEND CRASH.EXE[1,4] BY 4 BLOCKS:

The user should type in the name of the structure on which CRASH.EXE[1,4] should be extended by four blocks. SDLCNV will extend the specified crash file and exit. Another method of accomplishing the same result is to refresh and restore the disk pack using the new ONCE/TWICE.

CRSCPY

CRSCPY (short for Crash Copy) is a program to copy dumps taken by MONBTS. It is run either by the operator or semiautomatically by the monitor using FRCLIN. TOPS-1Ø automatically runs CRSCPY when a continuable STOPCD dump is taken or when the monitor is reloaded. Without CRSCPY running automatically, useful dumps could be overwritten before they are saved. In the past, users have had to write their own dump program because a program of this type did not exist.

FRCLIN

TOPS-1Ø now reserves the TTY line FRCLIN for system use. It gives the monitor a mechanism for running programs itself (for example, CRSCPY). To run programs on FRCLIN, the monitor sends a command to the FRCLIN. If the command decoder (COMCON) does not find a match in COMTAB with the command, it runs SYS:name.EXE[1,4] where "name" is the name of the command. The program is started in much the same way as jobs are when they are not logged in. Programs run on FRCLIN do not need to be logged in. They are allocated a job number for the duration of the program's execution. Once execution finishes, the job does not return to monitor level but is automatically logged out. Any program run on the FRCLIN must detach itself as soon as possible to free the line. Other points to note about FRCLIN are:

1. The FRCSET routine is used by the monitor to type commands to itself.

2. When COMCON looks for commands to process, it always looks at FRCLIN first.

3. Messages from FRCLIN are prefaced ';;SYSTEM -', not ';;OPR -'.

4. Set TTY functions are not permitted for FRCLIN.

CRSCPY COMMANDS

CRSCPY has three types of commands: action, status setting, and report selection. Action commands perform an immediate function; status commands modify action commands, and report selection commands modify only the report action command. All commands take the form:

CRSCPY> command argument

The action commands are:

CLEAR filespec	Mark the specified file as having been processed so that MONBTS can dump on it without operator intervention. The use of this command is not usually required since the COPY command marks the crash file as having been copied when it finishes the process. It may be useful after stand-alone time or preventive maintenance to ensure that no old dumps are present.
COPY filespec=filespec	Copy input filespec to output filespec, make a log entry in SYS:CRASH.SYS, and clear the unprocessed dump bit in the file being copied so that MONBTS can dump on it without operator intervention. If only one filespec is typed, it is used as the filespec of the input file.

DISPOSITION seqnum	Give a disposition for the crash with sequence number "seqnum". The disposition is a one-line description of what caused the crash and is given after the crash is analyzed. The disposition may be printed using the DETAIL:DISPOSITION switch or command.
PURGE FILE	Delete the contents of SYS:CRASH.SYS but retain the header so that the crash sequence numbers do not start at one. Preferably use this command rather than simply deleting the file. The argument "FILE" is required to ensure that the user does not type the command by accident.
REPORT filespec	Generate a report on the specified file of the contents of SYS:CRASH.SYS. The contents of this report can be restricted by using one or more of the report selection commands described below.

All action commands use default arguments if the arguments are omitted. Defaults are as follows.

1. CLEAR SDL:CRASH.EXE[1,4]
2. COPY STR:sssnnn.EXE[10,1]=SDL:CRASH.EXE[1,4]
3. REPORT TTY:CRASH.LOG[-]

Where SDL: is all structures in the system dump list, STR: is the structure with the most free space selected from the STRUCTURE command, sss is the STOPCD name, and nnn is a sequence number obtained from the header of SYS:CRASH.SYS which is incremented each time a dump is copied.

The status setting commands are:

DELETE — If this command is specified, CRSCPY automatically deletes the crash file when the crash is disposed. NODELETE disables this action and is the default.

INFORM name — Select destination of all output. Legal values of "name" are USER and OPR. USER is the default if CRSCPY is run manually and causes output to go to the user's terminal. OPR is the default if CRSCPY is run by the system and causes output to go to device OPR:. This command should not be used in normal operation.

STRUCTURE <str:blk,str:blk,...>,<str:blk,str:blk,...>,...

Select the destination structure for the crash that is being copied. This command is necessary only if no output structure is specified in the COPY command. The name of a structure is "str" and "blk" is the number of blocks which must remain on that structure after the copy is completed. CRSCPY will not copy to a structure unless it meets this minimum block restriction. The angle brackets (<>) group structures into sets. CRSCPY scans the sets from left to right and selects a structure from the first set which meets all restrictions. Within each set, CRSCPY selects the structure which meets the minimum block restrictions and will contain the most space after the copy. This command usually appears in SWITCH.INI with a line of the form:

CRSCPY/STRUCTURE:(str:blk,str:blk,...>,<str:blk,...>,...)

It allows the system administrator to specify to which structures crash may be copied if not explicitly overridden by the operator. Note that since CRSCPY runs logged out ([2,5]) when run by the system, the SWITCH.INI containing the STRUCTURE command must be placed in [2,5].

The report selection commands are:

BEGIN date:time	Reports only on crashes dumped after the specified date and time.
CBEGIN date:time	Reports only crashes copied after the specified date and time.
CEND date:time	Reports only crashes copied before the specified date and time.
(NO)DETAIL value	(Do not) give a detailed report. Legal arguments are ALL (gives a full report) and DISPOSITION (gives only the disposition.)
END date:time	Reports only on crashes dumped before the specified date and time.
MONVER nnn	Reports only on those crashes running the specified monitor version. The argument to this command is the version number from the monitor location MONVER, not the one contained in .JBVER.
PRIMETIME	Reports only those crashes which occurred during prime time (Ø8ØØ-17ØØ).
SEQUENCE n	Reports only on the crash with sequence number 'n'.

STOPCD xxx — Reports only on those crashes which occurred because of the specified STOPCD.

UNDISPOSED — Reports only those crashes which have not been disposed of yet.

The report selection commands simply define a set of entries in SYS:CRASH.SYS to be reported upon. They have no effect on any action command other than REPORT.

Examples of the CRSCPY reports are shown in the following listing. Reports may be normal (the first listing) or detailed (the second listing). The default report shows all crashes recorded in SYS:CRASH.SYS.

```
REPORT BY CRSCPY V1(14)  27-FEB-8Ø 16:29:Ø5  PAGE 1

SEQ      MONITOR NAME          VER  WHY  CRASH DATE/TIME       COPIED TO
---  ----------------------- ----- ---  ------------------ --------------------
 1        RS35Q  KS #41Ø1     7ØØ35 IME 22-FEB-8Ø:19:ØØ:56 DSKB:IMEØ68.EXE[1Ø,1]
 2        RSØ34A KS #41Ø1     7ØØ34     26-FEB-8Ø:14:32:13 DSKB:SERØ69.EXE[1Ø,1]
 3        RSØ34A KS #41Ø1     7ØØ34     26-FEB-8Ø:2Ø:Ø1:Ø8 DSKB:SERØ7Ø.EXE[1Ø,1]
 4        RSØ34A KS #41Ø1     7ØØ34     26-FEB-8Ø:2Ø:25:22 DSKB:SERØ71.EXE[1Ø,1]
 5U       RSØ36A KS #41Ø1     7ØØ36 BAC 27-FEB-8Ø:11:56:15 DSKB:BACØ72.EXE[1Ø,1]
 6U       RSØ36A KS #41Ø1     7ØØ36 BAC 27-FEB-8Ø:11:57:56 DSKB:BACØ73.EXE[1Ø,1]
 7U       RSØ36A KS #41Ø1     7ØØ36 IPM 27-FEB-8Ø:11:59:19 DSKB:IPMØ74.EXE[1Ø,1]
 8U       RSØ36A KS #41Ø1     7ØØ36 IPM 27-FEB-8Ø:12:ØØ:2Ø DSKB:IPMØ75.EXE[1Ø,1]
 9U       RSØ36A KS #41Ø1     7ØØ36     27-FEB-8Ø:16:24:27 DSKB:SERØ76.EXE[1Ø,1]


REPORT BY CRSCPY V1(14)  27-FEB-8Ø 17:14:Ø9  PAGE 1

SEQ      MONITOR NAME          VER  WHY  CRASH DATE/TIME     UPTIME    COPY DATE/TIME       COPIED FROM           COPIED TO
---  ----------------------- ----- ---  ------------------ -------- ------------------ ------------------- --------------------
 1        RS35Q  KS #41Ø1     7ØØ35 IME 22-FEB-8Ø:19:ØØ:56  Ø:Ø8:55 22-FEB-8Ø:19:ØØ:Ø7 DSKB:CRASH.EXE[1,4] DSKB:IMEØ68.EXE[1Ø,1]
     GIVE UP
 2        RSØ34A KS #41Ø1     7ØØ34     26-FEB-8Ø:14:32:13  Ø:11:11 26-FEB-8Ø:19:29:1Ø DSKB:CRASH.EXE[1,4] DSKB:SERØ69.EXE[1Ø,1]
     NOT A CRASH, JUST SHUT
 3        RSØ34A KS #41Ø1     7ØØ34     26-FEB-8Ø:2Ø:Ø1:Ø8  Ø:32:Ø6 26-FEB-8Ø:2Ø:Ø4:17 DSKB:CRASH.EXE[1,4] DSKB:SERØ7Ø.EXE[1Ø,1]
     NOT A CRASH, JUST SHUT
 4        RSØ34A KS #41Ø1     7ØØ34     26-FEB-8Ø:2Ø:25:22  Ø:21:19 26-FEB-8Ø:2Ø:52:16 DSKB:CRASH.EXE[1,4] DSKB:SERØ71.EXE[1Ø,1]
     NOT A CRASH, JUST SHUT
 5        RSØ36A KS #41Ø1     7ØØ36 BAC 27-FEB-8Ø:11:56:15  Ø:5Ø:13 27-FEB-8Ø:11:56:17 DSKB:CRASH.EXE[1,4] DSKB:BACØ72.EXE[1Ø,1]
 6        RSØ36A KS #41Ø1     7ØØ36 BAC 27-FEB-8Ø:11:57:56  Ø:51:54 27-FEB-8Ø:11:57:58 DSKB:CRASH.EXE[1,4] DSKB:BACØ73.EXE[1Ø,1]
 7        RSØ36A KS #41Ø1     7ØØ36 IPM 27-FEB-8Ø:11:59:19  Ø:53:17 27-FEB-8Ø:11:59:24 DSKB:CRASH.EXE[1,4] DSKB:IPMØ74.EXE[1Ø,1]
 8        RSØ36A KS #41Ø1     7ØØ36 IPM 27-FEB-8Ø:12:ØØ:2Ø  Ø:54:18 27-FEB-8Ø:12:ØØ:13 DSKB:CRASH.EXE[1,4] DSKB:IPMØ75.EXE[1Ø,1]
 9        RSØ36A KS #41Ø1     7ØØ36     27-FEB-8Ø:16:24:27  Ø:Ø2:25 27-FEB-8Ø:16:25:14 DSKB:CRASH.EXE[1,4] DSKB:SERØ76.EXE[1Ø,1]
```

PROGRAMS

TGHA

TGHA (The Great Heuristic Algorithm) is a program that manages and maintains MOS (Metal Oxide Semiconductor) memory in MF20s. Since MOS memory is more prone to errors but has a greater ability to correct errors than core memory, TGHA is a necessary and valuable service tool.

MOS MEMORY

When a user programs a 1091 with MOS memory, he or she sees the usual 36-bit word. In reality, each word of MOS memory is really 43 bits. MOS is organized in groups of 16K words. Each group is composed of 43 chips. Each chip will contain a certain bit for each of the 16K words in the group. For example, the third chip will contain bit two for all words in the group. When a request for a memory address is made, the correct bit in each chip is referenced. The seven extra parity bits allow the detection and correction of single bit errors and the detection of double bit errors.

Another error correction mechanism built into MOS memory is the "space bit". When one entire chip starts to go bad (in other words, the same bit in every word of a module), there is a spare chip that can be used to substitute for the chip that is going bad. The spare chip is also known as the "spare bit". TGHA can be used to substitute the spare bit when the need arises.

A new monitor module (MOSSER) has been added to directly control MF20 MOS memory and to provide the interface with TGHA. MOSSER detects MOS errors and sends IPCF messages to TGHA. TGHA gathers the information from MOSSER and asks for MOSSER to substitute the spare bit. In addition, MOSSER contains two new DIAG. monitor call functions that allow a program like TGHA to control MOS memory. MOSSER must be compiled and linked with the monitor explicitly; it is not included in the standard command files.

DATA FILES

TGHA stores data in two files: 1) the history file (TGHAV2.DAT) and 2) the trace file (TGHA.TRA). This history file contains data about MOS memory currently on the system:

1. Modules on/off line
2. If the spare bit is on
3. How the spare bit is being used
4. If error reporting is turned on
5. Correctable errors that have occurred

The trace file records the time and corrective action taken by TGHA. If the history file is corrupted, TGHA attempts to reconstruct the history file using the actions recorded in the trace file.

All correctable errors and TGHA actions are recorded in the SYSERR file.

TGHA OPERATIONS

System Startup

At initial system startup, TGHA should be run in startup mode. This enables single bit error reporting throughout MF20 memory. TGHA then either builds the history file if it does not already exist, or verifies that it knows about all of the on-line MF20 hardware. If the history file exists and new MF20 hardware appears, TGHA adds this new hardware to its history file.

The following commands to start TGHA should be placed in OPR.ATO after DAEMON is started:

```
:SLOG
:DEF TGHA=
R TGHA
```

TGHA responds with:

```
TGHA 2(3) RUNNING FIRST TIME
```

This message indicates that TGHA is running for the first time since the monitor reload. If the system has only core memory, no message is printed. Since TGHA makes entries in the SYSERR file via DAEMON, DAEMON must be running before TGHA is started. Once the initialization is complete, TGHA then looks for any MF20 errors that have occurred since startup.

Error Reporting

When an MF20 correctable error occurs, MOSSER sends a message to TGHA, which stores the information in the history file. Up to 256 entries can be stored for each group (16K). The spare bit is not used immediately. When the table for a particular module becomes full, TGHA tries to analyze the error, using all 256 entries to determine its extent. TGHA then determines the best way to use the spare bits to correct the largest amount of errors reported by the module.

Parity errors are handled as usual. That is, if the monitor successfully continues after the parity error, it attempts to place the page of memory containing the parity error off-line. If the page was successfully removed, the monitor then runs TGHA. In some cases, the monitor may not be able to take the page with the parity error off-line. In this case, TGHA enters parity errors in the trace file and SYSERR entry file.

The following message is printed on the CTY, when serious errors occur, to inform the operator of MOS problems:

```
*******************************************************
* <Date/Time>
* TGHA HAS TEMPORARILY CORRECTED A SERIOUS MOS MEMORY FAILURE
*** CALL FIELD SERVICE TO REPORT THIS CONDITION
*******************************************************
```

The specific error type is recorded in greater detail in the trace file. Some of the trace file entries are:

```
**********
<Date/time>
* PARITY ERROR AT ADDRESS xxxxxxx, BLOCK y
* STORAGE MODULE SERIAL NUMBERS BY FIELD:
  Ø = xxxxxxx 1 = xxxxxxx 2 = xxxxxxx 3 = xxxxxxx
**********
```

```
*****
<Date/time>
THE MEMORY BOOT IN KLI HAS USED THE SPARE BIT TO PREVENT
A PARITY ERROR
THIS CONDITION SHOULD BE CORRECTED AS SOON AS POSSIBLE.
CONTROLLER      GROUP   BLOCK   WORD (BITS 33-35)
xx              x       x       x
*****
```

```
*******************************************************
* <Date/time>
* THE FOLLOWING BLOCKS ARE MARKED AS BAD
* AND ARE NOT ON LINE
*  CONTROLLER   GROUP   BLOCK
*      xx         x       x
* THIS CONSISTS OF xxxxK MEMORY THAT IS OFF LINE

*** CALL FIELD SERVICE TO REPORT THIS CONDITION

*******************************************************
```

Running TGHA Manually

TGHA can be run manually to produce readable versions of the TGHA data base and the trace file. No changes in the memory configuration or the use of the spare bits can be done by TGHA in user mode.

To run TGHA type:

```
.R TGHA
```

It responds with the prompt.

```
TGHA>
```

There are four commands:

1. EXIT - Exit from TGHA
2. HELP - Type the list of commands
3. HISTORY - Dump the history file
4. TRACE - Dump the trace file

The history and trace dump files are created in the area where TGHA is run. They are called HISTRY.LST and TRACE.LST.

DAEMON

DAEMON Version 20 is a major rewrite from Version 16. All of the code specific to monitors before Version 6.03 has been removed and many bugs have been fixed. All of Version 6.03 code has been placed in MACRO conditionals and will be available only if recompiled with the correct program switches. The main thrust of the rewrite is to update the code to Version 7.01.

The following new functions have been implemented.

1. DX20 device errors (TU7x tape errors reported by TAPSER) are logged.
2. The CPU status block is dumped on CPU errors with error code 63.
3. The device status block is dumped on CPU errors with error code 64.
4. Software events of interest are logged with error code 14.
5. If the system date/time is changed, the incremental change is logged with subfunction 3 of error code 15.
6. Disk error information has been removed from error code 5 to avoid overflowing the entry. The information is stored via error code 45.

7. Nodes on the network going off-line and on-line are logged via subcodes 6 and 7 of the configuration status change entry.

8. Five new error codes were added to provide support for RSX20F general error logging. These codes report on:

   a. DL-11/DM-11BB Errors
   b. DEX Errors
   c. EBus Parity Errors
   d. RH-11 (SY:) Errors
   e. Configuration Information
      (1) DL11 Configuration
      (2) DH11 Configuration
      (3) LP20 Configuration
      (4) CD11 Configuration

There is one more major change in DAEMON. To measure the availability of a system, DAEMON now wakes up every six minutes and logs the state of the system in a file, SYS:AVAIL.SYS. The format of the file is the same as ERROR.SYS. File entries consist of reload information, device status change data, date/time changes, and other pertinent information. DAEMON closes the AVAIL file every Sunday night at midnight and renames it to AVAIL.X??, where ?? is an incremental number.

A program called AVAIL.EXE can process AVAIL.SYS files and produce a report describing system availability. It is a COBOL program with MACRO subroutines to read the binary file. To run the program, type:

```
.R AVAIL
SYSTEM AVAILABILITY REPORTER -- TYPE [CR] FOR ALL DEFAULTS
TYPE "H" FOR HELP ANYTIME

SELECTED OPTION: H

REPORTS SYSTEM AVAILABILITY BY TRANSLATING AVAIL.SYS
        OR AVAIL.A??
OPTIONS:
        1. REPORT A SINGLE WEEK (ONE AVAIL FILE)
        2. REPORT MULTIPLE WEEKS IN ONE LISTING
        3. GENERATE MULTIPLE REPORTS (ONE FOR EACH
           AVAIL FILE)
```

SELECTED OPTION:

Reports can be regular or detailed. Detailed reports produce one file with reload information and one file with device status change information. A regular report produces a summary listing with both types of information. Samples of the listings are shown in Figures 3-1 through 3-4.

```
SYSTEM AVAILABILITY REPORT FOR SYSTEM SERIAL NUMBER: 1Ø26                                    PERIOD: 1Ø-JUN-8Ø TO 15-JUN-8Ø

                                   ***** WARNING *****
                        THIS REPORT IS QUESTIONABLE BECAUSE OF INTERNAL ERRORS.
                        FOLLOWING IS A DESCRIPTION  OF THOSE ERRORS FOUND.

                  ?-- TIMES IN AVAIL FILE REVERSED   9 TIMES

                          CUSTOMER SATISFIED(Y OR N)?____   CUSTOMER SIGNATURE____________________

***** SYSTEM STATISTICS *****(ALL TIMES IN HOURS)

AVAILABILITY FIGURES           SYSTEM EFFECTIVENESS FIGURES    RUNTIME FIGURES                  DOWNTIME FIGURES

  OPERATIONAL CYCLE  :  1Ø8.7    T= .1HRS:     96.5%          TOTAL RUN TIME     :  88.7      SYSTEM NOT RUNNING:   2Ø.Ø
  SYSTEM AVAILABILITY: 97.5%     T= .5HRS:     9Ø.9%          MAXIMUM RUN TIME   :  22.6      MAXIMUM DOWNTIME  :   47.8
  USER AVAILABILITY  : 97.1%     T=1.ØHRS:     84.8%          MINIMUM RUN TIME   :   Ø.1      MINIMUM DOWNTIME  :  -42.4
  NUMBER OF RELOADS  :   27      T=4.ØHRS:     54.6%          MEAN RUN TIME      :   3.3      MEAN DOWNTIME     :    Ø.7
                                                              STANDARD DEVIATION:    4.6

***** RELOADS AFFECTING MEASURED AVAILABILITY *****

MONITOR NAME                   VERSION  STOPCD   HALT   PARITY   HDW    NXM    HUNG   LOOP    CM    TOTALS
  RZ54B  KL #1Ø26/1Ø42         7ØØ54    2                                                            2   COUNT
                                        Ø.2                                                          Ø.2 TIME
  RZØ54A KL #1Ø26/1Ø42         7ØØ54    2                                      1                     3   COUNT
                                        Ø.2                                    Ø.2                   Ø.4 TIME
  RZØ55A KL #1Ø26/1Ø42         7ØØ55    2        1                             1                     4   COUNT
                                        Ø.4      Ø.1                           Ø.2                   Ø.6 TIME
  RJØ55A KL #1Ø26/1Ø42         7ØØ55    2                                                            2   COUNT
                                        Ø.6                                                          Ø.6 TIME
  RZ55B  KL #1Ø26/1Ø42         7ØØ55                     2                                           2   COUNT
                                                         Ø.8                                         Ø.8 TIME
                                        ____     ____    ____    ____   ____   ____   ____    ____  _____
                                        8        1       2                     2                     13  COUNT
                                        1.5      Ø.1     Ø.8                   Ø.4                   2.6 TIME

***** RELOADS NOT AFFECTING MEASURED AVAILABILITY *****

MONITOR NAME                   VERSION  POWER  STATIC   OPR     PM     NEW    SCHED   S/A    OTHER  TOTALS
  RZ54B  KL #1Ø26/1Ø42         7ØØ54                                                  1      1        2    COUNT
                                                                                      Ø.1    Ø.1      Ø.2  TIME
  RZØ54A KL #1Ø26/1Ø42         7ØØ54                                                  1               1    COUNT
                                                                                                           TIME
  RZØ55A KL #1Ø26/1Ø42         7ØØ55    1                                     3       2      1        7    COUNT
                                        Ø.2                                   -41.9   2.2    Ø.1    -39.4  TIME
  RJØ55A KL #1Ø26/1Ø42         7ØØ55                                          2       2               4    COUNT
                                                                              48.8    7.7            56.5  TIME
                                        ____   ____     ____    ____   ____   ____    ____   ____   _____
                                        1                                     5       6      2       14    COUNT
                                        Ø.2                                   6.9     1Ø.1   Ø.2     17.4  TIME
                                                                                                          MR 5109
```

Figure 3-1 Regular AVAIL Report (1 of 2)

SYSTEM AVAILABILITY REPORT FOR SYSTEM SERIAL NUMBER: 1026 PERIOD: 10-JUN-80 TO 15-JUN-80

***** DEVICE OUTAGES NOT AFFECTING MEASURED AVAILABILITY *****

DEVICE NAME	OCCURANCES	OUTAGE TIME
CPU	12	
MEM	2	
COMM	301	184.5
DISK	70	155.3

**********STOPCODES ORDERED BY FREQUENCY *****

STOPCD NAME	OCCURANCES
S..WTP	2
S..WEM	1
S..KAF	1
S..NPN	1
S..UIL	1
S..$	1
S..RPZ	1

MR-5110

Figure 3-2 Regular AVAIL Report (2 of 2)

```
DETAILED LISTING OF RELOAD DATA FOR SYSTEM SERIAL NUMBER: 1Ø26
    RZ54B  KL #1Ø26/1Ø42          7ØØ54   S..WEM   LOAD  :1Ø-JUN-8Ø ØØ:ØØ:ØØ
                                                   CRASH :1Ø-JUN-8Ø 13:29:32
                                                                        2.26Ø    RUN TIME
                                                   RELOAD 1Ø-JUN-8Ø 13:37:19
                                                                        Ø.131    DOWN TIME
    RZ54B  KL #1Ø26/1Ø42          7ØØ54   OTHER    LOAD  :1Ø-JUN-8Ø 13:37:19
                                                   CRASH :1Ø-JUN-8Ø 14:13:34
                                                                        Ø.611    RUN TIME
                                                   RELOAD 1Ø-JUN-8Ø 14:18:15
                                                                        Ø.Ø79    DOWN TIME
    RZØ54A KL #1Ø26/1Ø42          7ØØ54   S..WTP   LOAD  :1Ø-JUN-8Ø 14:18:15
                                                   CRASH :1Ø-JUN-8Ø 14:42:32
                                                                        Ø.4Ø9    RUN TIME
                                                   RELOAD 1Ø-JUN-8Ø 14:46:47
                                                                        Ø.Ø72    DOWN TIME
    RZØ54A KL #1Ø26/1Ø42          7ØØ54   S..WTP   LOAD  :1Ø-JUN-8Ø 14:46:47
                                                   CRASH :1Ø-JUN-8Ø 14:53:Ø3
                                                                        Ø.1Ø6    RUN TIME
                                                   RELOAD 1Ø-JUN-8Ø 15:ØØ:26
                                                                        Ø.124    DOWN TIME
    RZØ54A KL #1Ø26/1Ø42          7ØØ54   HUNG     LOAD  :1Ø-JUN-8Ø 15:ØØ:26
                                                   CRASH :1Ø-JUN-8Ø 18:Ø1:Ø6
                                                                        3.Ø45    RUN TIME
                                                   RELOAD 1Ø-JUN-8Ø 18:11:Ø1
                                                                        Ø.167    DOWN TIME
    RZ54B  KL #1Ø26/1Ø42          7ØØ54   S..KAF   LOAD  :1Ø-JUN-8Ø 18:11:Ø1
                                                   CRASH :1Ø-JUN-8Ø 2Ø:35:21
                                                                        2.433    RUN TIME
                                                   RELOAD 1Ø-JUN-8Ø 2Ø:42:Ø1
                                                                        Ø.112    DOWN TIME
    RZ54B  KL #1Ø26/1Ø42          7ØØ54   SA       LOAD  :1Ø-JUN-8Ø 2Ø:42:Ø1
                                                   CRASH :11-JUN-8Ø Ø3:ØØ:4Ø
                                                                        6.11Ø    RUN TIME
                                                   RELOAD 11-JUN-8Ø Ø3:Ø7:Ø4
                                                                        Ø.1Ø8    DOWN TIME
    RZØ54A KL #1Ø26/1Ø42          7ØØ54   SA       LOAD  :11-JUN-8Ø Ø3:Ø7:Ø4
                                                   CRASH :11-JUN-8Ø Ø3:13:17
                                                                        Ø.1Ø5    RUN TIME
                                                   RELOAD 11-JUN-8Ø Ø3:16:Ø1
                                                                        Ø.Ø46    DOWN TIME
    RZØ55A KL #1Ø26/1Ø42          7ØØ55   S..NPN   LOAD  :11-JUN-8Ø Ø3:16:Ø1
                                                   CRASH :11-JUN-8Ø Ø8:4Ø:31
                                                                        5.47Ø    RUN TIME
                                                   RELOAD 11-JUN-8Ø Ø8:55:Ø3
                                                                        Ø.245    DOWN TIME
                                                                                   MR-5111
```

Figure 3-3 Regular AVAIL Report – Reloads

DETAILED LISTING OF DEVICE STATUS CHANGES FOR SYSTEM SERIAL NUMBER: 1Ø26

NODE/ DEVICE NAME	SUBSYS NAME				RUN #	OUTAGE TIME	SUBSYSTEM TOTAL
RNC3	DISK	OFF	11-JUN-8Ø	15:Ø1:5Ø	11		
RNDØ	DISK	OFF	11-JUN-8Ø	15:Ø1:51	11		
RND1	DISK	OFF	11-JUN-8Ø	15:Ø1:52	11		
RND2	DISK	OFF	11-JUN-8Ø	15:Ø1:53	11		
RND3	DISK	OFF	11-JUN-8Ø	15:Ø1:54	11		
33	COMM	OFF	13-JUN-8Ø	12:25:52	24		
33	COMM	ON	13-JUN-8Ø	12:26:39	24		
					-----	Ø.Ø13 HRS	4.818 HRS
32	COMM	OFF	13-JUN-8Ø	12:26:59	24		
32	COMM	ON	13-JUN-8Ø	12:27:46	24		
					-----	Ø.Ø13 HRS	4.831 HRS
16	COMM	ON	13-JUN-8Ø	12:38:14	24		
					-----	18.6Ø4 HRS	23.435 HRS
14	COMM	ON	13-JUN-8Ø	12:38:17	24		
					-----	2Ø.433 HRS	43.868 HRS
14	COMM	OFF	13-JUN-8Ø	12:38:42	24		
16	COMM	OFF	13-JUN-8Ø	12:38:43	24		
16	COMM	ON	13-JUN-8Ø	12:39:32	24		
					-----	Ø.Ø14 HRS	43.882 HRS
14	COMM	ON	13-JUN-8Ø	12:39:36	24		
					-----	Ø.Ø15 HRS	43.897 HRS
77	COMM	OFF	13-JUN-8Ø	14:44:39	24		
71	COMM	OFF	13-JUN-8Ø	14:44:41	24		
75	COMM	OFF	13-JUN-8Ø	14:44:42	24		
RPEØ	DISK	OFF	13-JUN-8Ø	14:55:Ø6	24		
RPE1	DISK	OFF	13-JUN-8Ø	14:55:Ø8	24		
RPF3	DISK	OFF	13-JUN-8Ø	14:55:1Ø	24		
RPF5	DISK	OFF	13-JUN-8Ø	14:55:13	24		
RPF6	DISK	OFF	13-JUN-8Ø	14:55:16	24		
CPU1	CPU	OFF	13-JUN-8Ø	14:55:18	24		
MEM	MEM	OFF	13-JUN-8Ø	14:55:35	24		
MEM	MEM	OFF	13-JUN-8Ø	14:55:53	24		

MR-5112

Figure 3-4 Detailed AVAIL Report – Device Status Change

SYSERR

Many of the new SYSERR features were made in conjunction with changes to DAEMON. The following are new features in SYSERR.

1. /DEV: switch, which can handle partial matches. For example, /DEV:KLE is sufficient for SYSERR to recognize the KLERR device.
2. Added support for CPU status block reports.
3. Added support for device status block reports.
4. Added support for software event reports.
5. Added support for disk statistics reports.
6. Fixed KLERR report bug.
7. Increased support for configuration status changes by adding support for date/time changes and setting the CPU off-line/on-line.
8. Added support for hard copy controller error retry reports which replace the earlier line printer error reports.
9. Support for RSX2ØF general configuration and error logging.
10. Added the /NOW switch which sets up an IPCF link with DAEMON to report errors in real-time. It may be used in combination with any other SYSERR switch.

RSX20F Changes

Several features have been added to RSX2ØF that make it a better diagnostic tool. New PARSER commands control the file interface that reads command files and outputs log files, performs current KLERR functions, and expands the PARSER's diagnostic ability. Eventually PARSER will replace KLERR as the KL1Ø snapshot program.

PARSER COMMANDS

Formerly, the PARSER was overlayed to conserve memory usage. The new version of PARSER has the overlays removed; it runs as a single piece of code. This allows PARSER to run faster from the floppies.

The new commands are: TAKE, OUTPUT, AC-BLOCK, SWEEP, SHOW, WHAT HARDWARE, MARK-MICROCODE, SET RETRY, CLEAR DATE, EXAMINE DTE, and XCT.

1. TAKE Command - This command takes commands from a file and executes them as PARSER commands. All legal PARSER commands are allowed except another TAKE command. The commands in the file are executed until an end-of-file is detected. At this point, the message <EOF> is output, and input is then taken from the CTY. The form of the command is:

   ```
   PAR> TAKE filespec
   ```

   The default file name is PARSER, and the default extension is CMD. TAKE files for the PARSER must reside in the front-end file system.

   Example:

   ```
   PAR> TAKE SETUP.CMD
   PAR> !THIS IS AN INDIRECT COMMAND FILE TO SETUP OUTPUT
   PAR> !TO THE APPROPRIATE DEVICES
   PAR> SET OUTPUT LOG
   PAR> SHOW OUTPUT
    OUTPUT DEVICES: LOG
   PAR> SHOW RELOAD
    RELOAD ENABLE: OFF
   PAR> <EOF>
   PAR>
   ```

2. OUTPUT Command - This command causes CTY output to be directed to various devices. The available devices are the file PARSER.LOG, the CTY, and LPT. The output can be turned on with a SET command and turned off using a CLEAR command. If the log file does not exist, it is created. If after issuing a CLEAR OUTPUT command no devices are active, then the console terminal is made active. The WHAT or SHOW commands report on active

devices. All three devices may be turned on at the same time or any combination of two.

The form of the command is:

```
SET OUTPUT dev
CLEAR OUTPUT dev
WHAT OUTPUT
```

where the device may be: LOG, LPT, TTY.

3. AC-BLOCK Command - This command changes the current KL1Ø AC block to be the one specified by the number in the command by executing a DATAO PAG, instruction. It is intended for use by Field Service.

4. SWEEP Command - This command performs a sweep of a specified AC block or all AC blocks. The sweep consists of reading the contents of all the registers in a block and then checking for a parity error after each read.

   The output is a message when the block to be swept is set as the current one, and a message when an error is found. When a parity error is found, the following is output:

   ```
   FW-SWP> n:aa/dddddd dddddd
   ```

   where n is the AC block number, aa is the AC address, and d is the AC contents.

   If at the execution of this command the KL1Ø is in a state of a clock error stop from a FM error, the current contents of the FM output register is output in the form:

   ```
   FM PARITY ERROR-(BLOCK:ADDR/DATA) n:aa/dddddd dddddd
   ```

5. SHOW Command - This command is a synonym for the WHAT command.

6. WHAT HARDWARE Command - This command displays the environmental report that KLINIT generates. The report includes the KL serial number, model type, line frequency, and hardware options.

```
PAR> WHAT HARDWARE
 KL10 S/N 1026., MODEL B, 60 HERTZ
       MOS MASTER OSCILLATOR
       EXTENDED ADDRESSING
       INTERNAL CHANNELS
       CACHE
```

7. MARK/UNMARK-MICROCODE Command - This command sets or clears the mark bit in a specified CRAM location.

8. SET RETRY Command - RSX20F now provides a means by which the instruction in KL10 location 71 is executed on a KEEP-ALIVE-CEASE error. The following sequence occurs when RSX20F starts execution at location 71 (KAFLOC):

   ```
   KAFLOC: JSR @.CPKAF
   ```

   .CPKAF contains the location APxKAF (where x is the CPU number). At APxKAF, control passes to KAFSTP where a KAF stopcode is issued, and if possible, all the regular information stored by a stopcode is obtained.

   A control switch and associated PARSER command (SET RETRY) are provided for the operator to selectively disable the XCT 71 feature. This allows the operator to make a snapshot of the KL10 on all occurrences of KEEP-ALIVE-CEASE.

9. CLEAR DATE - This command has been modified so that when in maintenance mode the date validity is cleared and there is a prompt for a new date.

10. EXAMINE DTE - This command has been expanded to display descriptions in certain DTE registers. Refer to the following section (KLERR Changes) for an example of the output.

11. XCT Command - The XCT command takes a 36-bit numerical expression as an argument and executes this expression as a KL10 instruction. It accepts input in the form:

   ```
   PAR> XCT func dev addr
   ```

where:

func is one of the following:

a. CONI
b. CONO
c. DATAI
d. DATAO
e. BLKI
f. BLKO
g. CONSO
h. CONSZO

dev is the octal device code

addr is the I/O instruction right half

The input is decoded to create a 36-bit KL I/O instruction that is then executed. This form allows the user to obtain device status information without knowing the opcodes. The user need only know the device code of a few standard devices.

KLERR CHANGES

The KLERR program of RSX20F has been modified to make it a more useful tool in diagnosing KL10 hardware-related problems. Included in these modifications are:

1. Stopping KL10 clock. Upon start up KLERR will no longer attempt to force the KL10 into the microcode halt loop, but will instead turn off the KL10 clock to stop the machine and get it into a known state. Forcing the KL10 into the halt loop corrupted the state of the machine, thus invalidating the snapshot.

2. Display of the DTE-20 registers and printing a description of bit patterns in certain key words. CTY output from KLERR will now look like this:

```
KLERR -- VERSION V02-06 RUNNING

DLYCNT: 037777
DEXWD3: 000000
```

```
DEXWD2: 000000
DEXWD1: 000000
   KL10 DATA=000000,,000000
TENAD1: 104000  TENAD2: 000000
   ADDRESS SPACE=PHYSICAL
   OPERATION=EXAMINE
   PROTECTION-RELOCATION IS OFF
   KL10 ADDRESS=000000
TO10BC: 010000  TO11BC: 130000
TO10AD: 071330 TO11AD: 071356
TO10DT: 000000 TO11DT: 005000
   KL IN HALT LOOP
   MAJOR STATE IS DEPOSIT-EXAMINE
DIAG2 : 040000
STATUS: 002504
   DEX WORD 1
   11 REQUESTED 10 INTERRUPT
   E BUFFER SELECT
DIAG3 : 004000

KLERR -- KL IN HALT LOOP
KLERR -- KL ERROR OTHER THAN CLOCK ERROR STOP
KLERR -- KL VMA: 000000 005202 PC: 000000 005202
KLERR -- PI STATE: OFF, PI ON: 177,  PI HLD: 020
                    PI GEN: 000
KLERR -- EXIT FROM KLERR
```

For a complete description of the bits in the DTE-20 registers, see the DTE-20 Ten-Eleven Interface Unit Description (EK-DTE20-UD).

KLINIT

KLINIT has been changed to allow the operator greater flexibility in configuring memory, to display additional information concerning the machine environment, and to report and fix microcode errors.

An operator can now configure the memory controllers in reverse order and then save this configuration in the configuration file, KL.CFG. To do this, the operator must respond with "REVERSE" when asked the following question during the KLINIT dialog:

CONFIGURE KL MEMORY [FILE, ALL, REVERSE, FORCE, YES, NO]?

The "FORCE" option to this question is also new. Specifying "FORCE" causes the MOS memory code to do a double-bit error scan. This scan is then repeated if the memory is inconsistent during a reload.

KLINIT has been changed to give an expanded error report when CRAM or DRAM errors are encountered. The following is an example of these reports:

```
KLI -- ? C-RAM DIFFERS AT 43
KLI -- BAD  ØØ2556 Ø126ØØ ØØ2ØØØ ØØ264Ø 1ØØØØ2 1Ø
KLI -- GOOD ØØ2575 Ø127ØØ ØØ2ØØØ ØØ264Ø 1ØØØØ2 1Ø
KLI -- XOR  ØØØØ23 ØØØ1ØØ ØØØØØØ ØØØØØØ ØØØØØØ ØØ


KLI -- ? D-RAM DIFFERS AT 1Ø6
KLI -- BAD  A:2 B:Ø P:Ø J:1ØØ2 A:2 B:Ø P:Ø J:1ØØ2
KLI -- GOOD A:4 B:Ø P:Ø J:1412 A:2 B:Ø P:Ø J:1412
KLI -- XOR  A:6 B:Ø P:Ø J:Ø41Ø A:Ø B:Ø P:1 J:Ø41Ø
```

These error printouts may occur during the KLINIT dialog (if checking is explicitly requested) or during an automatic reload. To request microcode checking, the operator must repspond with "VERIFY" or "FIX" to the following dialog question:

```
RELOAD MICROCODE [YES, VERIFY, FIX, NO]?
```

FIX is a new option for this question. Specifying "FIX" causes the RAM values to be compared to the correct ones, a report (such as those above) to be produced when an error is found, and an attempt to be made to correct the error.

The reponse to KLINIT dialog question WRITE CONFIGURATION FILE [YES, NO]? has been changed. When the operator responds with "YES" or <CR>, KLINIT replies "CONFIGURATION FILE WRITTEN" instead of "ALTERED" as in previous releases.

KLINIT has also been modified to give an environmental report when bringing up the system. The report is similar to that produced by the PARSER command "WHAT HARDWARE".

KLINIT has added a tracking feature that reports on each operation during the initialization procedure. It is started by typing "T+" at any point during the initialization dialog and is stopped by typing "T-". The tracking capability prints a great deal of information on the CTY and takes a considerable amount of time, so use this feature wisely.

EXECUTIVE

The RSX20F data base has been moved in frontend memory so that it begins at location 1000. The data base is now referred to as the front-end status block.

Two new stopcodes have been added: 1) SAQ occurs if a negative send-all count is loaded and 2) SAI occurs if the send-all count becomes negative at interrupt level.

To get more information about device status, the I/O page is dumped when the frontend crashes. The I/O page is placed in a 4K word block starting at the base of the GEN partition (location 100000). If an address in the I/O page is nonexistent, the value 123456 is placed in the corresponding buffer position.

Modem and Terminal Service

The modem control section of the DM11-BB interrupt service routine has been modified to send a "line hang-up" message to TOPS-10 prior to sending the "line ring" message, when the line is in "carrier wait" state. This cures a previously serious breach of system security. Before, a user's job remained active after the modem was hung-up, thus allowing a small period of time during which any user could become attached to the job without logging in.

Terminal service has been modified to include auto-baud detection of 1200 baud.

In an attempt to reduce the latency inherent in the processing of terminal control characters, the Terminal Input Service has been modified to process each character upon receipt, instead of allowing input to stack up in the DH11 silos.

To prevent open lines from crashing the frontend by sending it too many characters, a change has been made to the count framing errors per line as they occur. After four consecutive framing errors are detected on a line (between timeout servicings), the line is shut down. When such a line shutdown occurs, the event is logged in the ERROR.SYS file. The line will stay down for 1Ø seconds.

Other miscellaneous changes to the Exec include:

1. Support of non-contiguous DH11s.

2. System start up configuration report to SYSERR.

3. Expanded error logging to DL11s, DM11BBs and RH11s.

4. EBus parity errors are retried and snapshot.

5. Local handling of XON/XOFF by RSX2ØF in selected circumstances.

PART FOUR

GALAXY VERSION 4.1

Chapter 4
GALAXY REORGANIZATION

WARNING

The information about Galaxy 4.1 and its related components (Usage Accounting, MDA, etc.) is subject to change. The release of Galaxy 4.1 has been decoupled from 7.01 and will not occur until February 1981. Any information contained in this document about those features is preliminary and may change before the release.

ORGANIZATION

The reorganization of the Galaxy system for Version 4.1 has involved a considerable change to the components of the system and the way in which these components communicate. All components now use IPCF messages for communications. QUASAR was given added responsibility for component parameters, as well as more queues to maintain. SPRINT was split into two components: one called CDRIVE, which acts as a device driver for the card readers, and the other still called SPRINT, which converts card images to ASCII. The UMOUNT program has been replaced by MOUNT, and OMOUNT has been split and put into QUASAR and PULSAR. PULSAR has been given the added responsibility for volume recognition of disk, as well as tapes. In addition to these component changes, two new components were added to the Galaxy system: ORION, to handle message routing and OPR, to provide an interface for the operator. All these changes are discussed in greater detail in this chapter.

One primary goal of the reorganization was to provide the operator with a unified interface to the components of the system. In the past, commands that were entered to BATCON or LPTSPL, for instance, were parsed by these components and were acted upon by them. This led to problems because the operator had to deal with a separate set of commands for each component and could talk to only one component at a time (except at OPSER command level, where directing the commands to the various components was awkward).

In Version 4.1, the operator interface has been unified into OPR. OPR allows the operator to deal with all Galaxy components and handle user requests. The operator has a single set of commands to learn for the whole Galaxy system. Command recognition, guide words, and help text for "?" are all provided via the TOPS-10 version of the TOPS-20 COMND JSYS (part of GLXLIB).

The operator controls the entire Galaxy system through commands entered to OPR. OPR parses these commands and passes the valid ones on to ORION using an IPCF message. ORION recognizes that the message relates to Galaxy and passes it on to QUASAR. The figure below shows the IPCF message paths used when an operator enters a command.

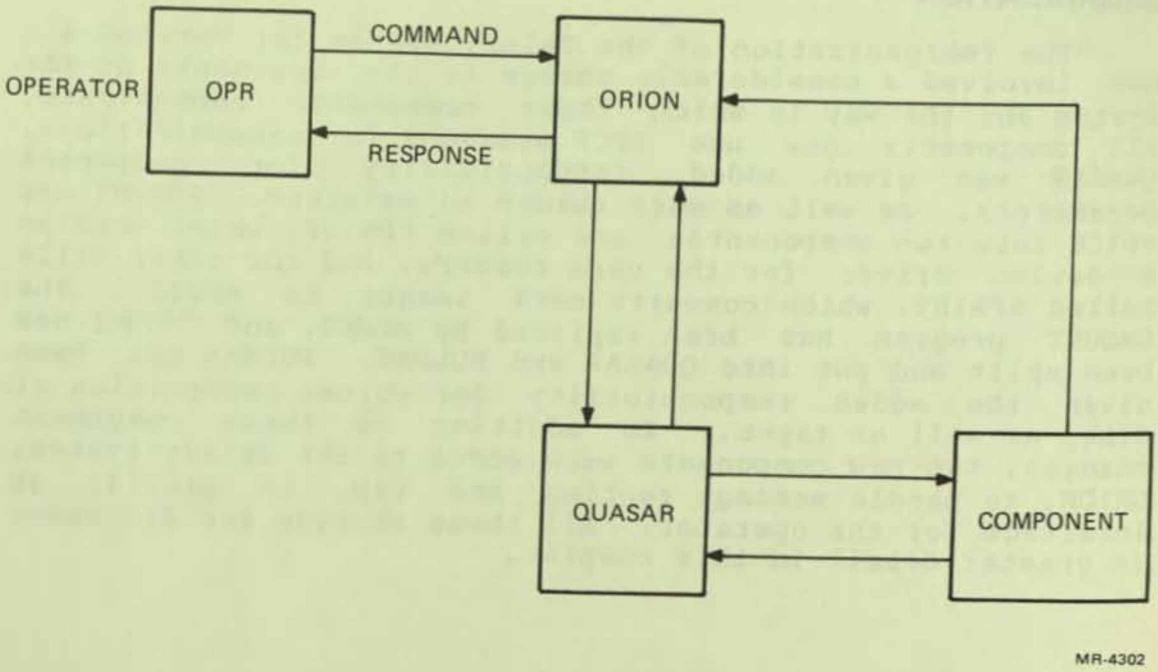


Figure 4-1 Operator Interface to Galaxy

The component can be any of the following: BATCON, LPTSPL, PULSAR, SPROUT, SPRINT, CDRIVE, or possibly IBMSPL.

When QUASAR receives the message, it interprets the command to the extent of determining which component the command pertains to. If the command is a request for information or sets a parameter, QUASAR handles it without notifying the component, and returns a response to ORION. If the command requires component action, as for a CANCEL, QUASAR sends a message to the component indicating the required action. In the latter case, after the component performs the action, it responds directly to ORION. If QUASAR also requires a response, the component sends a separate message back to QUASAR. When the response comes back to ORION, a reply message is returned to OPR to notify the operator.

Messages that come to the operator are of two types: WTO (Write-to-Operator) messages, which do not have a reply and are used to respond to commands and advise the operator of events; and the WTOR (Write-to-Operator-with-Reply) messages, which require a reply and are used to request some operator action. These two functions have been implemented as part of the QUEUE. monitor call.

Since OPR handles all operator commands, the command parsers were removed from the components. This removal made it unnecessary to run the components as subjobs under OPSER. OPR is best run as a stand-alone job so that the operator need not be concerned with the workings of OPSER.

The QUEUE. UUO also changes the user interface to Galaxy. A user can enter requests directly into the queues using this monitor call's subfunctions that send IPCF messages to the appropriate Galaxy components.

Figure 4-2 shows the structure of all the jobs that run in the new Galaxy system.

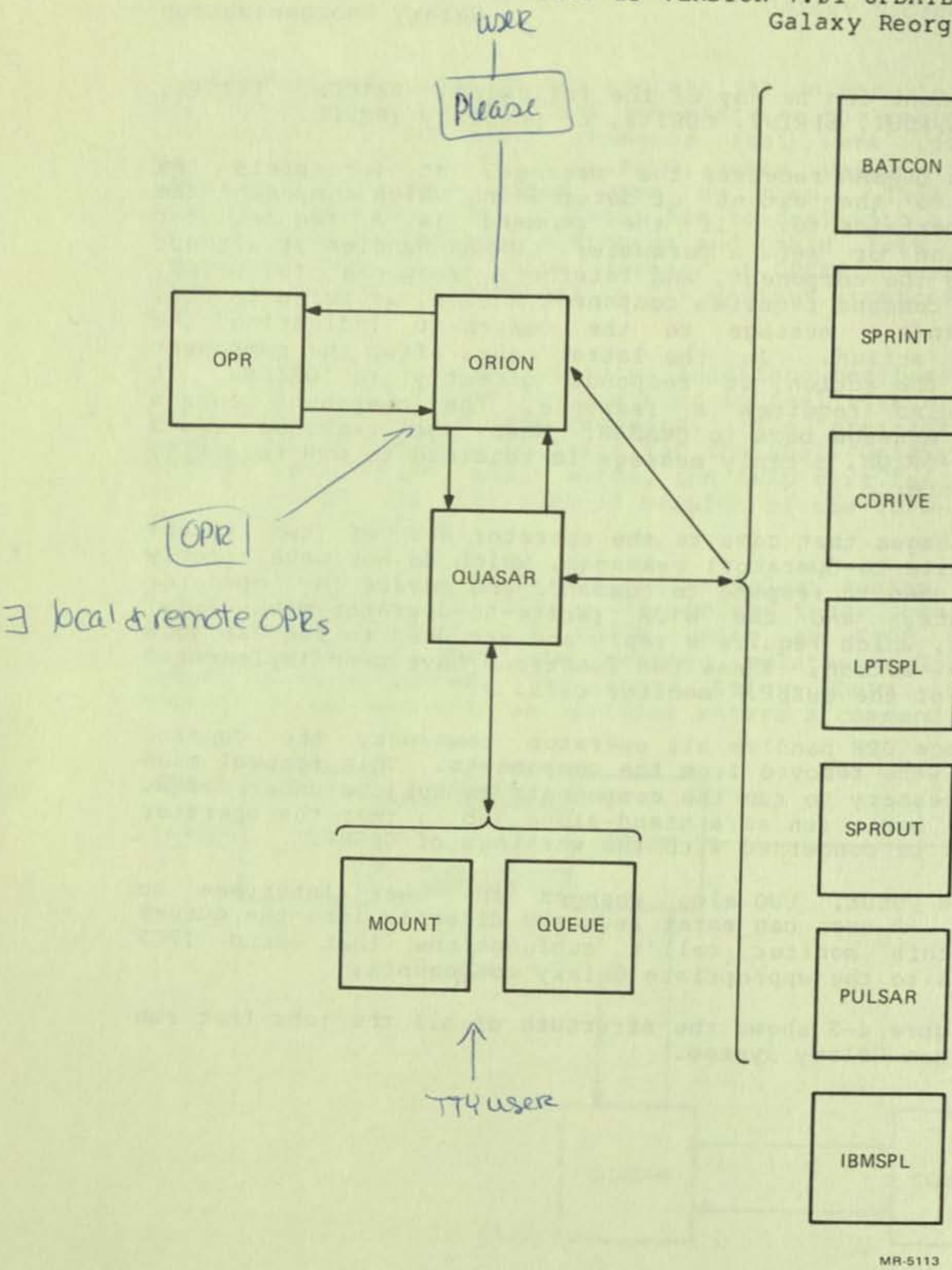


Figure 4-2 Galaxy Components

Galaxy Library

Two other goals for Galaxy Version 4.1 were to make the code for the components more consistent and to reduce the amount of repeated code in various components. To accomplish these things, a Galaxy library was developed.

The Galaxy library, GLXLIB, is essentially a shareable runtime system for Galaxy. It consists of a set of modules each containing related global subroutines. These routines provide an interface with the operating system for performing various functions such as reading and writing files, managing memory, communicating with IPCF, etc.

The GLXLIB routines are designed to be used in a manner that is independent of the operating system. However, use of the library places two restrictions on the way host programs work. Programs using the library must use the library's memory manager and library routines may not be called at interrupt level. All components of the Galaxy system use GLXLIB, and each acts as a host program. The address space for a host program using GLXLIB is shown below.

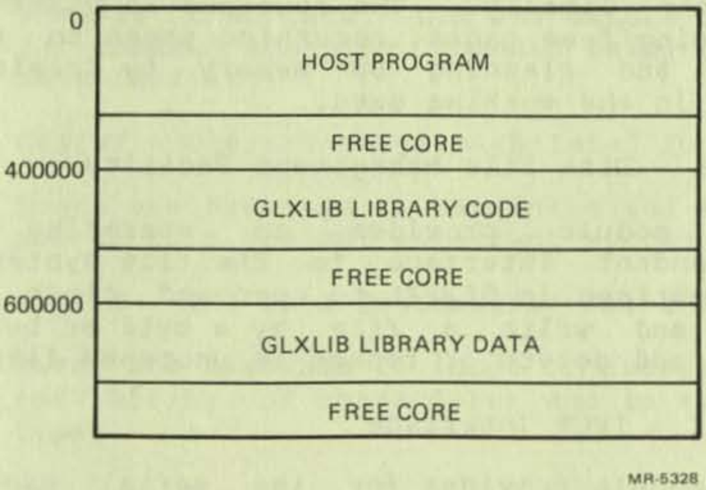


Figure 4-3 Galaxy Component Address Space

Note that the host program must fit between locations 0 and 377777. All modules of the library are mapped into the GLXLIB library code area at runtime by PUSHJ'ing to a library bootstrap routine which eventually does a GETSEG on GLXLIB.EXE. Although each component does not use all modules, there is enough space in the library code area to contain them all. The user who creates his/her own Galaxy system should find the availability of all routines convenient for possible Galaxy modifications that would call previously unused routines.

The Galaxy library has accomplished both of its goals for improving the Galaxy system. It has reduced the amount of code in the various components by having each component use library routines for common tasks. GLXLIB has also imposed an overall consistency on the coding within the components since the interface to library routines is always the same.

There are eight modules in the Galaxy library (individual modules cannot be split out of Galaxy). Following is a brief description of each of the modules and the routines they contain.

1. GLXMEM - Memory Management Facility

   This module must be present for a host program to use the library. The routines in GLXMEM involve acquiring free pages, returning pages to the free pool, and cleaning up memory by freeing unused pages in the working sets.

2. GLXFIL - Disk File Management Facility

   This module provides an operating system independent interface to the file system. There are routines in GLXFIL to open and close a file, read and write a file by a byte or buffer at a time, and delete or rename an unopened file.

3. GLXIPC - IPCF Interface

   This module provides for the serial handling of IPCF messages. There are routines to read IPCF parameters, block reception of messages, release the space for the last message, as well as for sending and receiving messages.

4. GLXLNK - Linked List Manipulation Facility

   All manipulations of the lists are based on a CURRENT entry field. If CURRENT entry is lost, attempting to access the list generates an error. There are routines to create and destroy a list, add and delete an entry, transverse a list, and change the CURRENT entry field.

5. GLXTXT - Format ASCII Strings

   This module is the support module for the $TEXT macro which allows variable text to be embedded in the message. The variable text is indicated by a circumflex followed by a qualifier.

6. GLXPFH - TOPS-10 Page Fault Handler

   This module is the page fault handler for GLXLIB and the component that uses it. The code starts at the beginning of the library high segment (400000) and the data is in the low segment at the page before (377000).

7. GLXSCN - TOPS-10 COMND JSYS Simulator

   This module contains the code that simulates the TOPS-20 COMND JSYS. OPR and GALGEN both rely on it for command scanning, command recognition, and the help facility.

8. GLXINT - Operating-System-Related Functions

   There are routines to terminate and suspend a job and to turn the PSI interrupt system on and off.

9. GLXKBD - I/O to a Controlling Job's Terminal

   There are routines to input or output a character (or string of characters) and to set the terminal type.

10. GLXCOM - Utility Routines

This module contains routines to handle several minor details. There are routines to save various accumulators and to zero a portion of memory. These routines are used by almost every Galaxy component.

System Control

The principal job of OPR is to parse the commands as they are entered by the operator. These messages are then sent to ORION for disposition.

Operator privileges are assigned by the system administrator on a [P,PN] basis. OPR's do not need to be [1,2] in order to control the Galaxy subsystem.

ORION

ORION functions as a message clearinghouse. It directs messages to QUASAR and to OPRs. Version 4.1 of Galaxy allows for multiple operators (users running OPR) at local or remote locations. ORION distinguishes among these OPRs for message routing purposes. ORION allows local operators to see or change any aspect of the system for which there is a command. Remote operators are restricted to seeing information about or changing parameters for only those devices present at their remote site.

All messages that pass through ORION are logged in a special buffer file. The logged message includes the time, date, and source of the message in addition to the message text. There is an OPR command to write the buffer file to a log file (the default log file is ORNLOG.ØØ1[3,3]). There is also an OPR command to tell ORION to stop logging messages.

By default, ORION sends messages coming from all components to each operator. This may be inconvenient for an operator since the components generate many messages. However, an operator has the capability (using an OPR command) to choose which categories of messages will or will not be sent to his/her terminal. ORION uses the operator's selection of messages to screen out those that are not wanted. A remote operator can receive only messages pertaining to his/her remote station.

ORION also handles PLEASE and SEND OPR requests by users. The PLEASE program sends the user's message to ORION and, depending on the switches specified, may set up a link to receive a reply and/or block until the reply is received. ORION passes the message on to the operator by sending a WTO or WTOR message to OPR, receives any operator reply, and passes the reply back to the user. The following figure illustrates the message paths for the users of PLEASE.

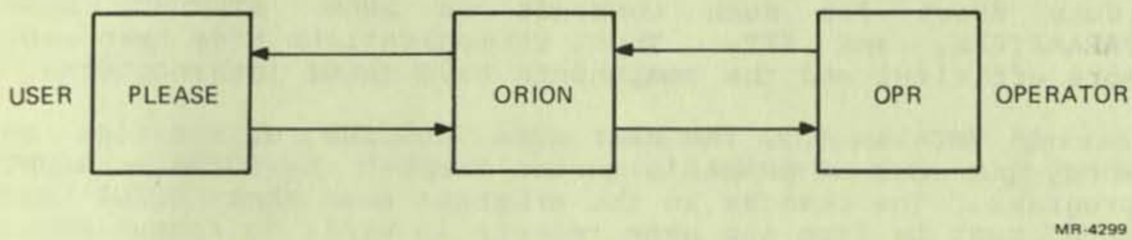


Figure 4-4 ORION's Role for PLEASE

The PLEASE program has been modified for Version 4.1 to allow the user to specify, by means of switches, whether or not the operator has to reply and whether or not the PLEASE program should wait for a reply. The PLEASE program can also be used to send several messages at a time, send messages to operators at other nodes, or cancel messages already sent.

Finally, ORION handles OPR commands entered at remote stations in a special way. OPR parses commands entered at the DN200 console and passes them on to ORION where they are handled normally. For an IBM remote station , ORION itself parses incoming commands and directs them to QUASAR for action.

QUASAR

As in previous versions, QUASAR maintains the queues used by the Galaxy system components. In Version 4.1, QUASAR must maintain a new queue for MOUNT requests. This queue was previously handled by OMOUNT.

The QUASARs of Versions 2 and 4.1 are not compatible. The entries in the QUASAR master queue file differ in form in the two releases. Consequently, in switching from one release to the other, this file must be deleted. To avoid losses, the switch between releases should not be made until all queues are empty.

QUASAR has been given the added responsibility of keeping track of the parameters for the system components. OPR commands that request information about the status of one of the spooler devices or the parameters for that device only need to go as far as QUASAR to get the information. Giving QUASAR this new responsibility has kept the message route short for such commands as SHOW STATUS, SHOW PARAMETERS, and SET. Thus, communications have been made more efficient and the components have fewer interruptions.

In Version 4.1, the user adds, deletes, or modifies an entry in one of QUASAR's queues through the QUEUE or MOUNT programs. The changes in the programs mean that QUEUE and MOUNT must be from the same release in order to communicate; mismatches in releases do not work.

From the user's point of view, communication with the Galaxy system is as shown in Figure 4-5.

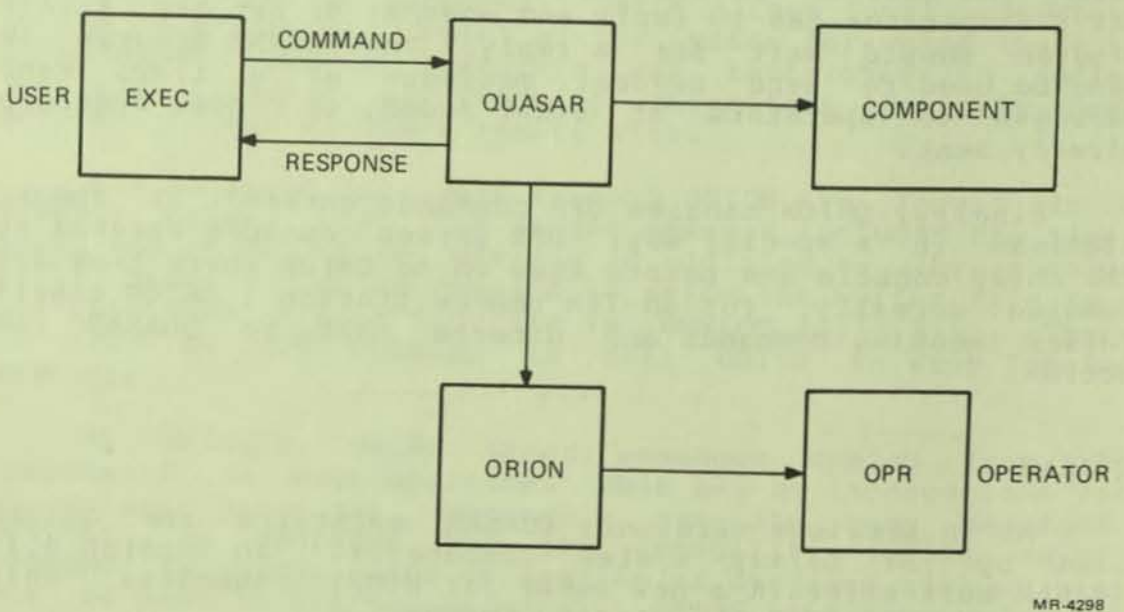


Figure 4-5 QUASAR's Handling of User Requests

Generally, QUASAR alone can handle the user's requests since most user commands deal with a queue entry or information about the queues as a whole. However, certain commands cause QUASAR to pass messages on to other Galaxy components. For example, the CANCEL command requires that a message be sent to the component if the current job is being canceled.

System Components

The following is a description of how the Galaxy components have changed and what the new components do.

1. BATCON

   BATCON has changed in four ways:

   a. BATCON no longer has a command parser. OPR parses commands, BATCON receives direction through IPCF messages.

   b. BATCON has been adapted to use the Galaxy library.

   c. BATCON now uses full duplex PTYs, allowing it to run faster since it no longer must handle character echoing.

   d. The header to the batch log files has changed.

   In terms of running a batch job, the user should not notice any functional difference from BATCON in Version 2.

2. LPTSPL

   LPTSPL is different in the following ways:

   a. LPTSPL no longer has a parser. OPR parses commands, LPTSPL receives directions from QUASAR via IPCF messages.

   b. LPTSPL has been adapted to use the Galaxy library.

c. In working with an IBM remote job entry station in termination mode, LPTSPL handles the printer spooling rather than D6ØSPL.

d. One copy of LPTSPL can run up to 15 line printers.

3. SPRINT

The SPRINT of Version 2 has been split into CDRIVE and SPRINT for Version 4.1. SPRINT takes the spooled files produced by CDRIVE that contain Hollerith card images and converts them to ASCII. SPRINT also interprets the control cards in the file as it builds the INP file, which is passed to BATCON for processing. One copy of CDRIVE can control up to 15 card readers at one time.

SPRINT has also been adapted to use the Galaxy library. It has no operator commands.

4. CDRIVE

CDRIVE's function was formerly handled by SPRINT. CDRIVE spools the cards read onto disks; in so doing, it performs a device driver function. It checks the card images for only $JOB and $EOJ cards to determine the beginning and end of a job. No conversion of the card images is made; the spooled files contain Hollerith images.

CDRIVE also directs OPR commands that come in from an IBM remote station to ORION. Incoming OPR commands are preceded by the characters "$$". These commands may not be inside a job, that is, they may not appear between the $JOB and $EOJ cards. CDRIVE strips off the "$$" and passes the command to ORION for parsing and action.

CDRIVE can handle multiple card readers. By separating the card reading function from the control card interpretation, the card reader(s) run faster. The reading speed is only limited by what the monitor can sustain and should be equivalent to that obtained by a .COPY DSK:=CDR: command.

5. SPROUT

SPROUT takes its initial parameters for each job from a SPFORM.INI file, which is similar in form to the LPFORM.INI file used by LPTSPL. There are switches to set parameters for all three devices that SPROUT controls. The switches in SPFORM.INI that are relevant to card punches specify the size of the banner, header, and trailer as a number of cards. Similar switches for paper tape punches and plotters specify these sizes in number of folds and number of units, respectively. The plotter has additional switches to specify the number of steps per unit and the size of the overall display.

As an example of a command for a card punch, consider the following:

```
CDP NORMAL/BANNER:3/HEADER:1/TRAILER:2
```

This command indicates that NORMAL cards are to be used with the specified numbers of banner, header, and trailer cards.

6. PULSAR

PULSAR has been modified to handle all mountable devices, that is, disk and tape drives. To be able to do this, PULSAR absorbed part of OMOUNT and was given the responsibility for performing volume recognition on both those devices. To perform volume recognition for tapes, PULSAR was also given the responsibility for tape labeling.

a. Disk Structures

When a disk drive is powered on, an interrupt will be received by TOPS-1Ø which notifies QUASAR. PULSAR will read the home blocks of the drive and recognize the structure. It then passes the information to QUASAR to process any MOUNT requests. QUASAR will attempt to handle any MOUNT requests without operator intervention.

Users' requests to mount structures and tapes are handled by placing them in a queue. This queue can be listed by users (using the monitor commands SHOW

QUEUE MOUNT-REQUESTS or MOUNT/CHECK) or by the operator (using the OPR command SHOW QUEUES MOUNT-REQUESTS). A user requesting this information, or submitting or canceling a request causes the MOUNT program to send a message to QUASAR. In responding to the request, the operator causes a message to be sent from ORION to QUASAR, which associates the request with the response and adjusts the structure's status.

b. Tapes

To handle tape drives, MDA must be given control of them explicitly. This control is given by issuing the following command to OPR:

```
SET TAPE-DRIVE MTxx: AVAILABLE
```

This command causes all tape drives (seven-track and nine-track) to come under MDA's control. Without this command, the tape drives can only be treated as assignable devices. The MOUNT command will not work without PULSAR and QUASAR controlling the drives. This is more restrictive than the :RESTRICT command in OPSER; not even [1,2] jobs may assign the tape drives.

If it is desirable to remove any tape drive from MDA's control, the OPR command SET TAPE-DRIVES UNAVAILABLE can be used. To return a tape drive to MDA's control, SET TAPE-DRIVES AVAILABLE can be used. Those tape drives not under MDA's control can be treated as assignable devices.

MDA provides the option of using automatic volume recognition on tape drives under its control. If automatic volume recognition (AVR) is enabled on a tape drive (with an OPR command), PULSAR attempts to read the first record of any tape mounted on that drive as soon as the drive is made ready. PULSAR attempts to interpret the first record on the tape as a volume label in order to identify the tape. PULSAR tries to match the record against the prototype of an EBCDIC label and an ANSI label. If no match is found, the tape is considered unlabeled.

but some drives don't generate interrupts when are made ready so system won't check

The operator can enable or disable AVR on any tape drive under MDA's control. Without AVR, the operator must do more to identify each tape mounted on a drive. It is expected that AVR will be enabled on all nine-track drives. AVR is not supported on seven-track drives.

It is not wise to mount and make ready a blank (virgin) tape on a drive that has AVR enabled. In attempting to read the first record, PULSAR may cause the tape to spin off the end of the reel. To avoid this problem, AVR should be disabled on a drive whenever it is known that a virgin tape is being mounted there. However, AVR need not be disabled in order to initialize (write a label on) a previously used tape.

Galaxy Generation

The GALGEN program is used to generate the GALCNF.MAC file, which contains various parameters related to the capabilities of the components in the Galaxy system. GALCNF.MAC is assembled with the other Galaxy modules to create the Galaxy system. Using GALGEN has been made easier through the use of the COMND JSYS simulator to parse the entries and the addition of explanatory text for each question. This text is listed for all questions by specifying the LONG option or for individual questions by entering "?". The dialogue in the short form will look like this:

```
.R GALGEN

GALGEN VERSION 4(2003)


[STARTING GALAXY GENERATION PROCEDURE FOR TOPS-10 SYSTEM]
[WRITING GALAXY CONFIGURATION FILE DSKB:GALCNF.MAC[77,4655]]

IN THE FOLLOWING DIALOG, ALL QUESTIONS ARE OF THE FORM:

TEXT OF QUESTION (LIST OR RANGE OF ANSWERS) [DEFAULT ANSWER]

THE LINE STARTS WITH THE ACTUAL TEXT OF THE QUESTION.
FOLLOWING THE QUESTION IS A DESCRIPTION OF THE POSSIBLE
```

```
ANSWERS ENCLOSED IN PARENTHESES.  THIS DESCRIPTION MIGHT BE
A  RANGE  OF  NUMBERS, A LIST  OF  KEYWORDS,  OR A TEXTUAL
DESCRIPTION.  FOLLOWING  THIS  DESCRIPTION  IS  THE  DEFAULT
ANSWER  (WHAT WILL BE ASSUMED IF YOU SIMPLE PRESS THE RETURN
KEY.)


YOU HAVE THE CHOICE OF CARRYING ON A LONG DIALOG IN WHICH AN
EXPLANATION OF EACH QUESTION IS PROVIDED AUTOMATICALLY, OR A
SHORT DIALOG IN WHICH IT IS ASSUMED THAT  YOU  ARE  FAMILIAR
WITH  THE  GALAXY GENERATION PROCEDURE.  IN EITHER CASE, THE
HELP TEXT CAN ALWAYS BE GOTTEN BY  TYPING  A  QUESTION  MARK
("?") AS THE RESPONSE TO ANY QUESTION.

ANSWER THE  FOLLOWING  QUESTION  EITHER  LONG  (FOR  A  LONG
DIALOG)  OR  SHORT  (FOR  A SHORT ONE).  SIMPLY PRESSING THE
RETURN KEY WILL IMPLY SHORT.

DIALOG LENGTH(SHORT,LONG) [SHORT] SHORT


                ADMINISTRATIVE CONTROLS AND PARAMETERS
                -------------- -------- --- ----------

OPERATOR LOG FILENAME(3-6 CHARCTERS) [ORNLOG]
MAXIMUM PRIORITY NON-PRIVILEGED USERS MAY SPECIFY(1Ø-62) [2Ø]
DEFAULT FOR /PRIORITY ON BATCH AND SPOOLING REQUESTS(1-62) [1Ø]
NUMBER OF MINUTES BETWEEN CHECKPOINTS(1-1Ø) [1]
DO YOU WANT REDUNDANT MASTER QUEUE FILE(YES,NO) [NO]
FILE-STRUCTURE TO USE FOR MASTER QUEUE(3-6 CHARACTERS) [SYS]


                        BATCH JOB DEFAULTS
                        ----- --- --------

DEFAULT BATCH jOB RUNTIME LIMIT(5-9999 SECONDS) [3ØØ]
DEFAULT LINEPRINTER OUTPUT LIMIT(5-9999 PAGES) [2ØØ]
DEFAULT CARD-PUNCH OUTPUT LIMIT(5-9999 CARDS) [1ØØØ]
DEFAULT FOR BATCH /FEET: SWITCH(1Ø-5ØØØ) [2ØØ]
```

WHAT IS THE MAXIMUM PLOT TIME A BATCH JOB MAY GENERATE
(10-5000 MINUTES) [200]
DEFAULT FOR BATCH /OUTPUT SWITCH(LOG,NOLOG,ERROR) [LOG]
SHOULD MEMORY LIMITS BE ENFORCED(YES,NO) [NO]

LINEPRINTER DEFAULTS AND PARAMETERS

NUMBER OF JOB BANNER PAGES(0-5) [2]
NUMBER OF JOB TRAILER PAGES(0-5) [2]
NUMBER OF FILE HEADER PAGES(0-5) [2]
NAME FOR STANDARD OUTPUT FORMS(4-6 CHARACTERS) [NORMAL]
NUMBER OF CHARACTERS WHICH UNIQUELY IDENTIFIES FORM(2-6) [4]

MISCELLANEOUS DEFAULTS AND PARAMETERS

DO YOU WANT DEFAULT LIMIT COMPUTATION(YES,NO) [YES]
DEFAULT OUTPUT LIMIT EXCEEDED ACTION(ABORT,ASK,IGNORE) [ASK]

[END OF GALGEN DIALOG]

If help is needed during the short dialogue, typing a "?" will print out the long form of the question:

DEFAULT BATCH JOB RUNTIME LIMIT(5-9999) SECONDS) [300] ?

THE BATCH USER CAN SPECIFY A MAXIMUM RUNTIME FOR HIS BATCH
JOB USING THE /TIME SWITCH. IF HE DOES NOT SPECIFY THIS
SWITCH, THE FOLLOWING DEFAULT IS USED.

DEFAULT BATCH JOB RUNTIME LIMIT(5-9999 SECONDS) [300]

The GALCNF.MAC file contains the questions asked (as comments), as well as the instructions they generate.

QUEUE. Monitor Call (CALLI 201)

The QUEUE. monitor call, allows user software to interface with the various facilities of the Galaxy system. It supports the following functions:

.QUPRT	Print a file.
.QUCDP	Punch a file on cards.
.QUPTP	Punch a file on paper tape.
.QUPLT	Plot a file.
.QUBAT	Process a file under batch.
.QUALC	Allocate a volume set.
.QUDAL	Deallocate a volume set.
.QUMNT	Mount a volume set.
.QUDIS	Dismount a volume set.
.QUWTO	Write to operator.
.QUWTR	Write to operator with reply.
.QUVAL	Validate an account.
.QUMAE	Send a message to the account DAEMON.

Each one of these functions roughly corresponds to a monitor command (e.g., .QUPRT corresponds to the PRINT command).

The main purpose of this UUO is to take the user arguments and build an IPCF package which is sent to the appropriate Galaxy component. The monitor maintains a table that maps each function into a specific component.

The calling sequence for the monitor call is:

```
MOVE    ac,[len,,addr]
QUEUE.  ac,
error return
normal return
```

where:

len - is the length of the argument block
addr - is the address of the argument block

The format of the argument block is:

1. Addr: flags,,function (as described above)
   node
   rlen,,rspblk
   arg #1
   arg #2
   .
   .
   .
   arg (#len-3)/2

2. Node - is the network node used to process the request (0 is current node, -1 for central site)

3. Rlen - is the length of the response block

4. Rspblk - is the address of the response block

5. Arg n - are two-word blocks holding the arguments for the function. Each two-word block is organized as follows:

I		Length	Type
	Value or Address		

If I is not set, the second word is an address containing more parameters. If I is set, the second word contains a value.

"Length" is the length of the data block pointed to by "Address".

The argument TYPES are:

.QBFIL==10	File Spec Block
.QBCOP==11	Number of Copies Block
.QBFRM==12	Forms Type Block
.QBPTP==13	File Format Block
.QBODP==14	/DISPOSITION Output Block
.QBUNT==15	Unit Type Block
.QBAFT==16	/AFTER Block
.QBLIM==17	Limit Block
.QBUNI==20	Unique Block

.QBAFT==16	/AFTER Block
.QBLIM==17	Limit Block
.QBUNI==20	Unique Block
.QBRES==21	/RESTART Value
.QBLOG==22	/OUTPUT: (LOG,NOLOG,ERROR)
.QBACT==23	/ACCOUNT: Account String Block
.QBNOD==25	/NODE: Destination node Block
.QBNAM==26	User's Name Block
.QBOID==27	Owner's PPN Block
.QBNOT==30	/NOTIFY Block
.QBBLT==31	/BATLOG: Block
.QBJBN==32	Job Name
.QBCDI==33	Connected Directory Block
.QBNTE==34	/NOTE: Block
.QBBGN==35	/BEGIN: Block
.QBPRI==36	/PRIORITY Block
.QBVSN==37	ASCIZ Volume Set Name
.QBMSG==40	WTO/WTOR Message Block
.QBTYP==41	Privileged WTO/WTOR Message Block
.QBDEN==42	Tape Density Block
.QBTRK==43	Tape Track Block
.QBLTP==44	Label Type Block
.QBRMK==45	Remark Block
.QBVOL==46	Tape Volume List Block
.QBLNM==47	Volume Set Logical Name Block
.QBMFG==50	Mount/Dismount Flag Block
.QBAFN==51	Account DAEMON Subfunction (See ACTSYM on the CUSP tape)
.QBAET==52	Type of Usage Entry

The flags halfword has only one bit defined, QF.RSP. This specifies that a response is requested and requires that a response buffer be specified. The .QWTOR function implies a response. Upon successful return from the UUO, the accumulator will contain the length of the actual response in the right half and the flags in the left half.

Refer to UUOSYM.MAC for more information on how to set up the argument blocks.

OPERATOR INTERFACE

Command Comparison

To make the transition from Version 2 to Version 4.1 easier in terms of adjusting to the operator commands, the following tables of BATCON, LPTSPL, and SPRINT commands from Version 2 are provided for comparison. Refer to Tables 4-1, 4-2, and 4-3. Some Version 2 commands were not carried over to Version 4.1 as they were not considered necessary. These have no counterpart Version 4.1 OPR commands and show "None" under the OPR command list. Some Version 2 commands did not quite fit into the Version 4.1 philosophy of Galaxy and thus have their functions handled in a different way. For such commands, an associated OPR command is given and indicated by "(assoc)". An associated OPR command performs a related action that is not quite equivalent to the Version 2 command. The OPR commands listed as unqualified counterparts to the Version 2 commands provide at least as much (and often more) functionality than the corresponding Version 2 commands.

Table 4-1 BATCON Command Comparison

BATCON Commands	OPR Commands
CURRENT	SHOW PARAMETERS BATCH-STREAM
EXAMINE	SHOW CONTROL-FILE
EXIT	SHUTDOWN BATCH-STREAM (assoc)
GO	CONTINUE BATCH-STREAM RESPOND
INFORM	None
KILL	ABORT BATCH-STREAM CANCEL BATCH-REQUEST
MJOB	START BATCH-STREAM (assoc)
MONJOB	SHOW STATUS BATCH-STREAM
MTIME	SET BATCH-STREAM TIME-LIMITS

Table 4-1 BATCON Command Comparison (Cont.)

BATCON Commands	OPR Commands
NEXT	MODIFY BATCH-REQUEST (assoc)
OPERATOR	RESPOND
REQUEUE	REQUEUE BATCH-STREAM
RESET	SHUTDOWN BATCH-STREAM
ROUTE	ROUTE SHOW ROUTE-TABLE (assoc)
START	START BATCH-STREAM
STOP	STOP BATCH-STREAM
TELL	SEND
TIME	None
WHAT	SHOW STATUS BATCH-STREAM

The OPR command CONTINUE is equivalent to the BATCON GO command if the current job is halted with a STOP command. The RESPOND command is equivalent to BATCON GO if the job is blocked with a PLEASE request. Note that RESPOND is also used for operator responses in dialogue mode (RESPOND replaces OPERATOR; see the TOPS-10 and TOPS-20 Batch Reference Manual, AA-H374A-TK). The use of ABORT or CANCEL in place of the BATCON command KILL depends on whether or not the job being terminated is the current job. The SHOW ROUTE-TABLE command is used to display current routing.

Table 4-2 LPTSPL Command Comparison

LPTSPL Commands	OPR Commands
BACKSPACE	BACKSPACE
CHKPNT	None
CURRENT	SHOW PARAMETERS PRINTER
EXIT	SHUTDOWN PRINTER (assoc)
FORMS	SET PRINTER FORMS-TYPE
FORWARD	FORWARDSPACE
FREEZE/UNFREEZE	None
GO	CONTINUE PRINTER RESPOND
KILL	ABORT PRINTER CANCEL PRINTER-REQUEST
LIMIT	RESPOND in conjunction with SET PRINTER LIMIT-EXCEEDED-ACTION (assoc)
LOCK/UNLOCK	None
MESSAGE	ENABLE/DISABLE OUTPUT-DISPLAY (assoc)
MLIMIT	SET PRINTER PAGE-LIMITS
MONITOR	None
NEXT	MODIFY PRINTER (assoc)
PAUSE	SHUTDOWN PRINTER (assoc)

Table 4-2 LPTSPL Command Comparison (Cont.)

LPTSPL Commands	OPR Commands
REPRINT	REQUEUE PRINTER
REQUEUE	REQUEUE PRINTER
RESET	SHUTDOWN PRINTER (assoc)
SKPCOPY	FORWARDSPACE
SKPFILE	FORWARDSPACE
START	START PRINTER
STOP	STOP PRINTER
SUPPRESS/NOSUPPRESS	SUPPRESS
WHAT	SHOW STATUS PRINTER

The default frequency for taking a checkpoint is one per minute. This can be set to any desired value when GALGEN is run, thus, an equivalent of CHKPNT should not be needed. The CONTINUE command is equivalent to LPTSPL GO when the job is halted with STOP; RESPOND is equivalent to it if the printer is halted on a forms change. Capability similar to the LPTSPL command LIMIT can be attained by setting the LIMIT-EXCEEDED-ACTION to "ASK" beforehand and then using RESPOND when asked. The OPR REQUEUE differs from LPTSPL's REPRINT and REQUEUE in that the OPR command also holds the requeued job.

Table 4-3 SPRINT Command Comparison

SPRINT Commands	OPR Commands
EXIT	SHUTDOWN READER (assoc)
GO	CONTINUE READER
KILL	ABORT READER
MSGLVL	ENABLE/DISABLE OUTPUT-DISPLAY (assoc)
PAUSE	SHUTDOWN READER (assoc)
RESET	SHUTDOWN READER (assoc)
START	START READER
STOP	STOP READER
TELL	None
WHAT	SHOW STATUS READER

Most of these OPR commands are concerned with CDRIVE and not the Version 4.1 SPRINT. These commands control the action of one or more readers. SPRINT in Version 4.1 requires little control since it simply interprets control cards and converts card images to ASCII whenever CDRIVE creates a spooled file.

In addition to the counterpart commands listed above, there are OPR commands to handle queue entries in new ways; these include HOLD and RELEASE.

Operations

Jobs with privileges or running as [1,2] can run OPR. As far as ORION is concerned, anyone who runs OPR is an operator. Consequently, all information that could be of interest to an operator is sent to all OPRs via WTO messages. Each operator can select which messages to receive by disabling unwanted output display messages. See the description of the DISABLE command later in the command summary.

The operators receive WTOR messages when an event occurs for which some operator action MUST take place. The WTOR messages are easily distinguished from the WTO (information only) messages in two ways: WTOR messages ring the bell at the terminal and WTOR messages have a message number. Pending WTOR messages can also be recalled by using the SHOW MESSAGES command. Pending messages are those messages for which no operator has taken action.

A typical operator will most likely want to disable certain frequently occurring WTO messages, such as those for the beginning and end of a print job or batch job, as this eliminates a lot of unwanted messages. While the operator is present, he/she can reply immediately to incoming requests (WTORs). If the operator is absent for short periods of time, he/she can catch up with pending requests by entering "SHOW MESSAGES" and then replying. This is especially useful if the operator is not on a hard copy terminal.

In addition to keeping up on pending WTORs, the operator should periodically check the status of the devices and the length of the queues. The operator receives a WTO message when an entry is made in the MOUNT-REQUEST. The MOUNT-REQUEST queue should frequently be checked to maintain full utilization of the disk and tape drives.

OPR Commands

Admittedly, an operator's adjustment to Version 4.1 will require some time. The syntax for the OPR commands is different from that used for any the component parsers in Version 2. However, once the adjustment is the operator should find OPR easier to use than previous Galaxy interfaces.

USING THE COMMANDS

There are six important points to consider regarding OPR commands:

1. COMND JSYS simulator is used to parse the commands, so that the local operator can use recognition on the commands; noise words are often provided; CTRL/H can be used after a syntax error; typing "?" at any point will list the available options.

2. In addition to the commands listed here, there is a HELP command that prints descriptive information about the named command. The form of the HELP command is:

   HELP <command name>

3. Many of the commands dealing with devices that may be at a remote station, ANF1Ø, IBM, or an IBM host have an additional switch that may be specified. The switch is "/NODE:node-name" and is used to refer to devices at a particular node. Without specifying the node name, the node at which the operator is located is assumed.

4. Operators at remote sites cannot use OPR commands to affect any devices except those at their site, nor any batch jobs except those that originated at their site. Local operators can use OPR commands to affect ALL jobs/devices on the system.

5. Where file specifications are given in OPR commands, wildcard characters (* and %) may not be used.

For a complete description of all OPR commands with many examples of each command, see the Operator Command Language Reference Manual, AA-4074A-TB.

COMMAND DESCRIPTIONS

In the following descriptions of the OPR commands, the conventions used are:

1. Square brackets [] surround optional parts of a command. The word(s) inside the brackets describe the optional part.

2. Angle brackets <> surround parts of a command that must be entered. The word(s) inside the brackets describe the part that must be there.

3. Words in the commands that appear in lower case, such as "keyword," "argument," or "switch," have several options, all of which are listed in the description. A switch always begins with a "/".

4. In explaining the commands, anything appearing in parentheses () is a comment. The comment is not part of the command, it is there only to provide additional explanation.

5. At many places where a stream, unit, or priority number is indicated, a range of numbers may be entered in the form of n:m where n and m are stream/unit/priority numbers and n is less than m. For example, 0:3 could represent unit numbers 0, 1, 2, and 3.

ABORT

The ABORT command terminates a job request in progress on an input or output device, or in a batch stream.

The format is:

ABORT keyword <stream/unit number> [switch]

where keyword can be one of the following:

BATCH-STREAM
CARD-PUNCH
PAPER-TAPE-PUNCH
PLOTTER
PRINTER
READER

The possible values for the optional switch are:

```
/ERROR-PROCESSING          (use user error recovery)
/NOERROR-PROCESSING        (no error recovery available)
/REQUEST-ID:<request id number>
/PURGE
/REASON:single or multiple lines of text
```

The /ERROR-PROCESSING and /NOERROR-PROCESSING switches can only be used with the keyword "BATCH-STREAM". The /PURGE switch is used to eliminate all output associated with the job request which is only useful for output spoolers. The /REASON can be continued onto multiple lines with a hyphen and then terminated with a CTRL/Z.

Example:

OPR> ABORT BATCH-STREAM 2 /REQUEST-ID:2Ø3/PURGE

This command aborts the batch job with request-id 2Ø3 from batch stream 2.

ALIGN

The ALIGN command causes a forms-alignment file to be printed repeatedly on the line printer to align the forms.

The format is:

ALIGN PRINTER <unit number> <switch/argument>

The four possible switches or arguments are:

```
Alignment Filespec            (file to use for alignment)
/PAUSE:<number of seconds> (between printings)
/REPEAT-COUNT:<number of times to print>
/STOP                        (to end alignment)
```

Example:

OPR> ALIGN PRINTER Ø /PAUSE:5 PAYCHK.ALP

This specifies that printer Ø is to be aligned using the control file PAYCHK.ALP with a 5 second pause between repeats of the print job.

BACKSPACE

The BACKSPACE command reprints previously output pages in the job currently printing on the line printer.

The format is:

BACKSPACE PRINTER <unit number> <switch>

The possible switch values are:

```
/COPIES:<number of copies>
/FILE                         (one file is assumed)
/PAGES:<number of pages>
```

Example:

OPR> BACKSPACE PRINTER Ø /PAGES:2

This command causes printer Ø to back up the current job by 2 pages, that is, print the last 2 pages again.

CANCEL

The CANCEL command deletes from a QUASAR queue any job request that is processing or is waiting to be processed.

The format is:

CANCEL keyword argument

where keyword can be one of the following:

BATCH-REQUEST
CARD-PUNCH-REQUEST
MOUNT-REQUEST
PAPER-TAPE-PUNCH-REQUEST
PLOTTER-REQUEST
PRINTER-REQUEST

The argument for any keyword except MOUNT-REQUEST can be one of the following:

request id number	(for a single job)
PPN	(for a certain PPN)
*	(for all job requests)

For MOUNT-REQUEST the argument is a request id number. In addition, the optional switch /REASON:, can be used to explain the cancellation to the user.

Example:

OPR> CANCEL PRINTER-REQUEST [77,4655]

This command cancels all print requests for all jobs logged in as [77,4655].

CLOSE

The CLOSE command closes the current ORION log file buffer and creates a new log file buffer. The closed log file is written to disk so that an up-to-date file can be copied or printed.

The format is:

OPR> CLOSE LOG

CONTINUE

The CONTINUE command continues a job request that was temporarily stopped with the STOP command.

The format is:

CONTINUE keyword <stream/unit number>

where keyword can be one of the following:

BATCH-STREAM
CARD-PUNCH
PAPER-TAPE-PUNCH
PLOTTER
PRINTER
READER

Example:

OPR> CONTINUE CARD-PUNCH Ø

This command causes card punch Ø to continue after it had been stopped temporarily.

DEFINE

The DEFINE command defines a node to be used for IBM Emulation/Termination. This command is only applicable if IBM Communication software is available.

The format is:

DEFINE NODE <node name> <type> <mode> <port #> <line #>

The node name is one to six characters in length. The type can be any one of the following:

2780
3780
HASP

The mode can be either of the following:

EMULATION
TERMINATION

The port number and line number are for the associated DN20 or DN22. The port number range is 11 to 13 (KL10) and 10 to 11 (KS10); the line number range is 0 to 5 (KL10) and 0 to 1 (KS10).

Example:

OPR> DEFINE NODE IBM21:: 3780 TERMINATION 11 1

This command defines node IBM21 as being a 3780. It is being used in termination mode through port 11 line 1.

DISABLE

The DISABLE command turns off various system processes as defined by the available keywords.

The format is:

DISABLE keyword argument

where keyword can be one of the following:

LOGGING	(of OPR actions)
OUTPUT-DISPLAY	(of messages)
QUEUE-REQUESTS	(of all queues)
STRUCTURE-RECOGNITION	(of disk structures)
VOLUME-RECOGNITION	(of tapes and disks)

Disabling LOGGING causes ORION to stop recording messages related to all operators. Disabling all queue entry requests means that no one can use the MOUNT, PRINT, SUBMIT, etc. commands.

The arguments for OUTPUT-DISPLAY are:

```
ALL-MESSAGES [switch]
BATCH-MESSAGES [switch]
CARD-PUNCH-MESSAGES [switch]
CARD-READER-INTERPRETER-MESSAGES [switch]
IBM-MESSAGES [switch]
MOUNT-MESSAGES [switch]
PAPER-TAPE-PUNCH-MESSAGES [switch]
PLOTTER-MESSAGES [switch]
PRINTER-MESSAGES [switch]
READER-MESSAGES [switch]
USER-MESSAGES [switch]  (such as PLEASE)
```

The optional switch may be any of the following:

```
/INFORMATION-MESSAGES
/JOB-MESSAGES          (from stopping/starting job
                        processing)
/OPR-ACTION-MESSAGES   (from user/job requests)
```

The OUTPUT-DISPLAY messages are disabled only for the operator entering the command.

The arguments for STRUCTURE-RECOGNITION are:

```
DISK-DRIVES
TAPE-DRIVES
disk-drive-name
tape-drive-name
```

Example:

```
OPR> DISABLE OUTPUT-DISPLAY PRINTER-MESSAGES
```

This command stops the display of messages related to the printer.

DISMOUNT

The DISMOUNT command dismounts (and removes) structures and unloads tape drives currently known to the system.

The format is:

DISMOUNT keyword argument

where the keyword and argument can be the following:

```
STRUCTURE  <structure name> /switch
TAPE-DRIVE  <tape drive number>
```

The values for <switch> may be:

/REMOVE causes the pack to spin down and stop.
/NOCHECK dismounts the structure without validation

DISMOUNT STRUCTURE str: removes the structure from the system. A user dismount just removes the structure from his search list. DISMOUNT TAPE xxx: rewinds and unloads the tape on drive xxx:. If a user has it assigned, the operator gets an error.

Example:

```
OPR> DISMOUNT TAPE-DRIVE MTA3:
```

This command rewinds and unloads the tape on MTA3.

ENABLE

The ENABLE command turns on various system processes as defined by the available keywords.

The format is:

ENABLE keyword argument

where keyword can be one of the following:

LOGGING	(of OPR actions)
OUTPUT-DISPLAY	(of messages)
QUEUE-REQUESTS	(of all queues)
STRUCTURE-RECOGNITION	(of disk structures)
VOLUME-RECOGNITION	(of tapes and disks)

Enabling LOGGING causes ORION to record messages related to all operators. Enabling queue requests allows users to place requests into the processing queues.

The arguments for VOLUME-RECOGNITION are:

DISK-DRIVES
TAPE-DRIVES
<disk drive name>
<tape drive name>

The arguments for OUTPUT-DISPLAY are:

ALL-MESSAGES [switch]
BATCH-MESSAGES [switch]
CARD-PUNCH-MESSAGES [switch]
CARD-READER-INTERPRETER-MESSAGES [switch]
IBM-MESSAGES [switch]
MOUNT-MESSAGES [switch]
PAPER-TAPE-PUNCH-MESSAGES [switch]
PLOTTER-MESSAGES [switch]
PRINTER-MESSAGES [switch]
READER-MESSAGES [switch]
USER-MESSAGES [switch] (such as PLEASE)

The optional switch may be any of the following:

/INFORMATION-MESSAGES	
/JOB-MESSAGES	(from starting/stopping job processing)
/OPR-ACTION-MESSAGES	(from user/job requests)

The default is to have all OUTPUT-DISPLAY messages enabled.

Example:

OPR> ENABLE OUTPUT-DISPLAY PRINTER-MESSAGES

This command causes a message to be printed for all print requests.

EXIT

The EXIT command leaves OPR command level and returns to monitor command level.

The format is:

OPR> EXIT

FORWARDSPACE

The FORWARDSPACE command skips forward without printing certain portions of the job currently being printed.

The format is:

FORWARDSPACE PRINTER <unit number> <switch>

The possible switch values are:

```
/COPIES:<number of copies>
/FILE                        (one file is assumed)
/PAGES:<number of pages>
```

Example:

OPR> FORWARDSPACE PRINTER 1 /COPIES:3

This command causes the printing of the next 3 copies of the current file to be skipped.

HOLD

The HOLD command keeps particular requests in the queues from being processed. When the current job cannot be held, the RELEASE command must be used to allow the jobs being held to be processed.

The format is:

HOLD keyword argument

where keyword must be one of the following:

BATCH-JOBS
CARD-PUNCH-JOBS
PAPER-TAPE-PUNCH-JOBS
PLOTTER-JOBS
PRINTER-JOBS

and the argument is one of the following:

request id number	(for a single job)
[Proj,Prog]	(for a certain PPN)
*	(for all job requests)

Example:

OPR> HOLD PRINTER-JOBS 434

This command causes the printer job with request id 434 to be held.

IDENTIFY

This command can be used to perform one of the following actions with a tape on a specific drive: give a tape a volume identifier, or associate a tape with a mount request.

The format is:

IDENTIFY <tape drive> keyword [argument]

The tape drive is in the form of MTAn:. The keyword and possible argument may be one of the following.

```
REQUEST-ID   <request id number>
VOLUME-ID    <volid>
```

REQUEST-ID is used to associate a user mount request with a tape volume. VOLUME-ID specifies the volume identifier for the tape on the indicated drive.

Example:

```
OPR> IDENTIFY MTA2: REQUEST-ID 35
```

This command associates the tape on MTA2 with tape mount request 35.

LOCK

The LOCK command restricts users from accessing particular structure so that the structure may be removed from the system. Timesharing jobs using the structure are refused access when they complete the currently running program; batch jobs when they finish.

The format is:

```
LOCK <structure name> <date and time>
```

<date and time> refers to the time when the LOCK action should take place.

Example:

```
OPR> LOCK LSCD: 8-MAR-8Ø 17:ØØ
```

MODIFY

The MODIFY command changes the priority of a job request waiting in a queue for processing or the system lists. When changing queue requests, the higher the priority number, the sooner the job will be processed.

The format for modifying queue requests is:

MODIFY keyword argument PRIORITY <priority number>

where keyword must be one of the following:

BATCH-REQUEST
CARD-PUNCH-REQUEST
PAPER-TAPE-PUNCH-REQUEST
PLOTTER-REQUEST
PRINTER-REQUEST

and the argument is one of the following:

request id number	(for a single job)
PPN	(for a certain PPN)
*	(for all job requests)

The priority number must be in the range 1 to 63.

Example:

OPR> MODIFY BATCH-REQUEST [10,701] PRIORITY 60

This command sets the priority to 60 for all batch jobs of [10,701].

The format for modifying system lists is:

MODIFY keyword argument structure

where keyword may be:

ACTIVE-SWAPPING-LIST
CRASH-DUMP-LIST
SYSTEM-SEARCH-LIST

The possible arguments are:

EXCLUDE
INCLUDE

and structure is the name of a structure.

Example:

OPR> MODIFY CRASH-DUMP-LIST INCLUDE DSKB:

will include DSKB: in the crash dump list.

MOUNT

The MOUNT command allows the operator to mount structures and, optionally, give the newly mounted structure an alias.

The format is:

MOUNT STRUCTURE Structure-name (as) Alias-name

Example:

MOUNT STRUCTURE LSCD: (AS) CD:

mounts LSCD: as the structure CD:.

RECOGNIZE

The RECOGNIZE command forces the system to read the volume labels of a tape or disk. This can be useful when automatic volume recognition has been turned off. If the tape drive is unavailable, the SET TAPE-DRIVE AVAILABLE command must be issued first.

The format is:

RECOGNIZE (labels on) argument

where argument is either a disk drive name or a tape drive name.

Example:

OPR> RECOGNIZE MTA2:

will force PULSAR to perform volume recognition on MTA2:.

RELEASE

The RELEASE command releases a job request that was held with the HOLD command.

The format is:

RELEASE keyword argument

where keyword must be one of the following:

BATCH-JOBS
CARD-PUNCH-JOBS
PAPER-TAPE-PUNCH-JOBS
PLOTTER-JOBS
PRINTER-JOBS

The argument can be one of the following:

request id number	(for a single job)
PPN	(for a certain PPN)
*	(for all job requests)

Example:

OPR> RELEASE PRINTER-JOBS 434

Allows the printer job with request id 434 to be considered for printing. This job had been previously held.

REPORT

The REPORT command makes journal entries into the system error file (ERROR.SYS).

The format is:

REPORT [user name] [device name] <message>

The user name identifies the writer of the report. The device name is optional; it need only be included if a particular device is involved. Single-line messages begin on the same line as the command and end with a carriage return; multiple-line ones begin on the line after the command and end with CTRL/Z.

OPR> REPORT LPT1 SHUTDOWN DUE TO TOO MANY ERRORS

OPR> REPORT SMITH
NETWORKS ARE ACTING FLAKEY<CTRL/Z>

REQUEUE

The REQUEUE command stops an in-progress job request, puts it back in the queue in a HOLD state, and aborts the in-progress request. The RELEASE command is used to activate the job request after it has been requeued. The job is continued from the point it was stopped.

The format is:

REQUEUE keyword <stream/unit number> [switch/argument]

where keyword can be one of the following:

BATCH-STREAM
CARD-PUNCH
PAPER-TAPE-PUNCH
PLOTTER
PRINTER

The argument can be one of the following:

BEGINNING-OF <COPY, FILE, or JOB>
CURRENT-POSITION

These arguments specify the restart point of the job and they are NOT valid for BATCH-STREAM.

Example:

OPR> REQUEUE BATCH-STREAM 1

This command aborts the current job in batch stream 1 and places it back in the queue in a held state.

RESPOND

The RESPOND command replies to WTOR messages that were sent to the operator from a user, such as a PLEASE request.

The format is:

RESPOND <message number> <response>

where the message number must be one of the pending messages. The response may be of one or more lines. Use CTRL/Z to end the response. In some cases, the first word of the reply may have special meaning, such as "REFUSE". Any such special first words are listed in the WTOR message.

ROUTE

The ROUTE command transfers output from one node to another node. The nodes in question must be either the host node or an RJE station.

The format is:

ROUTE keyword unit source-node destination-node unit

where source-node is the node name from which the output will be routed, and destination-node is the node name to which the output will be routed. Routing to a fictitious node is allowed; however, errors will occur if any output is directed to such a node.

Keyword may be one of the following:

ALL-DEVICES
CARD-PUNCH
PAPER-TAPE-PUNCH
PLOTTER
PRINTER

Example:

```
OPR> ROUTE PRINTER 1 1Ø26:: CTCH22:: Ø
```

This command causes the output from printer 1 on 1026 to be sent to printer 0 on CTCH22.

For the keyword ALLOCATION, the arguments are

ALL-REQUESTS
BATCH-REQUESTS <request-id/ALL>
JOB <job-number/ALL>

SEND

The SEND command sends single or multiple-line messages to destinations as defined by the available keywords.

The format is:

SEND keyword <number> <message>

where keyword can be one of the following:

ALL
BATCH-STREAM
JOB
OPERATOR
TERMINAL

The number is not applicable for ALL or OPERATOR. The number refers to a batch-stream number, a job number, or a terminal number depending on the keyword that is chosen. A multiple-line message is terminated with a CTRL/Z.

Example:

OPR> SEND ALL NEW STAND-ALONE SCHEDULE IS POSTED

This command sends the message to all terminals.

SET

The SET command sets various system parameters for the system devices as defined by the available keywords and arguments.

The format is:

SET keyword argument-1 argument-2

where keyword can be one of the following:

```
BATCH-STREAM
CARD-PUNCH
JOB
NODE
PAPER-TAPE-PUNCH
PLOTTER
PRINTER
TAPE-DRIVE
TERMINAL
USAGE
```

SET BATCH-STREAM

For the keyword BATCH-STREAM, argument-1 is the stream number. Argument-2 can be one of the following:

```
MEMORY-LIMITS
NOOPR-INTERVENTION
OPR-INTERVENTION
PRIORITY-LIMITS <number or range of priorities>
TIME-LIMITS <number or range in minutes>
```

SET <CARD-PUNCH, PAPER-TAPE-PUNCH, PLOTTER, or PRINTER>

For the keywords CARD-PUNCH, PAPER-TAPE-PUNCH, PLOTTER, and PRINTER, argument-1 is the unit number. Argument-2 can be one of the following:

```
FORMS-TYPE name
LIMIT-EXCEEDED-ACTION <ASK, CANCEL, or IGNORE>
OUTPUT-LIMITS <maximum number of output units>
              (not an argument for PRINTER)
PAGE-LIMITS <number or range of pages>
              (only available for PRINTER)
PRIORITY-LIMITS <number or range of priorities>
```

The output units for OUTPUT-LIMITS are cards, folds, and steps (in thousands) for the card punch, paper tape punch, and plotter, respectively.

SET JOB

For the keyword JOB, argument-1 is the job number. Argument-2 can be one of the following:

```
NOOPR-INTERVENTION
OPR-INTERVENTION
SCHEDULER-CLASS nn
```

SET NODE

For the keyword NODE, argument-1 is an IBM node name. Argument-2 can be one of the following:

```
BYTES-PER-MESSAGES  <number>     (for the front end)
CLEAR-SEND-DELAY  <time in jiffies>
DATA-TERMINAL-READY  <OFF or ON>
NO-SIGNON-REQUIRED
RECORDS-PER-MESSAGE  <number>    (for 2780 mode)
SIGNON-REQUIRED
SILO-WARNING-LEVEL  <warning level>
TIMEOUT-CATEGORY <PRIMARY or SECONDARY>
TRANSPARENCY  <OFF or ON>
```

SET TAPE-DRIVE

For the keyword TAPE-DRIVE, argument-1 is the tape drive number in the form MTAn:. Argument-2 can be one of the following:

```
AVAILABLE
INITIALIZE
UNAVAILABLE
```

AVAILABLE and UNAVAILABLE refer to whether MDA is handling the drive. If UNAVAILABLE is specified, a reason must be given. INITIALIZE is used to create labels on tapes. INITIALIZE takes any of the following switches:

```
/COUNT: <number of volumes to be initialized>
/DENSITY: <density value>  (200, 556, 800, 1600,
                             or 6250)
```

```
/INCREMENT:nn                     (Value to increment
                                   the tape volumes by)
/LABEL-TYPE: <type>               (ANSI, EBCDIC, or UNLABELED)
/OVERRIDE-EXPIRATION: <YES or NO>
/OWNER: <owner's name>            (Default is scratch)
/PROTECTION: <2-digit octal value>
/TAPE-DISPOSITION: <HOLD or UNLOAD>
                                  (Default is UNLOAD)
/VOLUME-ID:"volid"                (Volid for a tape, must
                                   be used with double
                                   quotes if any non-
                                   alphanumeric
                                   characters)
```

SET TERMINAL

For the keyword TERMINAL, argument-1 is:

TYPE
KEYPAD
NOKEYPAD

Argument-2 is the model name and can be any one of the following:

33	VT05
35	VT50
LA120	VT52
LA36	VT100

The SET TERMINAL KEYPAD command specifies that your terminal has an additional set of keys to the right of the keyboard. After issuing this command, pressing one of the numbered keys will perform a specific OPR function. This is valid for VT52 and VT100 terminals only, and if used, the terminal type must be set to VT52 or VT100 before issuing this command. The functions are:

0 - No function
1 - SHOW STATUS
2 - SHOW QUEUES
3 - SHOW PARAMETERS
4 - SHOW MESSAGES

5 - SHOW ROUTE-TABLE
6 - No function
7 - Clear screen, move OPR> prompt to the top of the screen
8 - SHOW OPERATORS
9 - SHOW QUEUE MOUNT-REQUESTS

SET USAGE

This command allows the operator to control the usage accounting system. SET USAGE can close the accounting files or turn off usage accounting

The format is:

SET USAGE argument-1

where argument-1 can be one of the following:

BILLING-CLOSURE <time>	(turn off usage accounting)
FILE-CLOSURE <date/time>	(close accounting files)

Example:

```
OPR> SET USAGE BILLING-CLOSURE 17:00
```

will turn off usage accounting at 5:00 PM.

SHOW

The SHOW command displays various types of information as defined by the available keywords.

The format is:

SHOW keyword argument-1 [switch/argument-2] where keyword can be one of the following:

ALLOCATION	(to see MDA allocation)
CONTROL-FILE	(to see a batch control file)
MESSAGES	(to see pending WTORs)
OPERATORS	(to see who is running OPR)
PARAMETERS	(to see parameters of a device)
QUEUES	(to see a QUASAR queue)
ROUTE-TABLE	(to see any routing of output)
STATUS	(to see the status of a device)
SYSTEM-LIST	(to see one of the system lists)
TIME	(to see the time of day)

There are no arguments for the keywords OPERATORS, ROUTE-TABLE, and TIME.

For the keyword ALLOCATION, the possible values for argument-1 are:

ALL-REQUESTS (for all jobs)
BATCH-REQUEST <Batch-stream number or CR for all>
JOB <job number>

For the keyword CONTROL-FILE, the only value for argument-1, argument-2, and the optional switch is:

BATCH-STREAM <stream number> [/LINES:<number>]

The default number of lines to examine in the control file is 1Ø.

For the keyword MESSAGES, an optional message number may be entered as argument-1. If no message number is given, all pending WTORs are listed. For SHOW MESSAGES to work, OPR-ACTION-MESSAGES must be enabled.

For the keyword PARAMETERS, the possible values of argument-1 and argument-2 are:

BATCH-STREAM	[stream number]
CARD-PUNCH	[unit number]
NETWORK-NODE	[node name]
PAPER-TAPE-PUNCH	[unit number]
PLOTTER	[unit number]
PRINTER	[unit number]

Not entering an argument-2 causes parameters for all objects of argument-1 type to be listed.

For the keyword QUEUES, the possible values of argument-1 are:

```
ALL-JOBS                    (default for SHOW QUEUES)
BATCH-JOBS
CARD-PUNCH-JOBS
MOUNT-REQUESTS
PAPER-TAPE-PUNCH-JOBS
PLOTTER-JOBS
PRINTER-JOBS
```

Argument-1 may be followed by either of the following optional switches:

```
/ALL                        (for all details)
/SHORT                      (for basic information)
```

For the keyword STATUS, the possible values of argument-1 and argument-2 are:

```
BATCH-STREAM                [stream number]
CARD-PUNCH                  [unit number]
DISK-DRIVES                 [switch]
NETWORK-NODE                [node name]
PAPER-TAPE-PUNCH            [unit number]
PLOTTER                     [unit number]
PRINTER                     [unit number]
READER                      [unit number]
STRUCTURE                   [structure name]
TAPE-DRIVE                  [tape drive]     [switch]
```

If no argument-1 is specified, the status of all devices except disk drives and tape drives is listed. If no argument-2 is specified, the status of all devices of argument-1 type is listed. In addition to or in place of argument-2, each of the above (except DISK-DRIVES, NETWORK-NODE, and TAPE-DRIVE) may use the optional switch /SHORT to get an abbreviated listing. The possible switch values for DISK-DRIVES are:

```
/ALL            (for all disk drives)
/FREE           (for all free drives)
/MOUNTED        (for all drives in use)
```

The possible switch values for STRUCTURE are:

```
/USERS    (to show all users of that structure)
```

The possible switch values for TAPE-DRIVE are:

```
/ALL            (adds label type, AVR,
                 and read/write status)
/FREE           (lists unassigned drives and
                 their status)
```

The NETWORK-NODE argument has no switches.

The possible arguments for SYSTEM-LIST are:

```
ACTIVE-SWAPPING-LIST
CRASH-DUMP-LIST
SYSTEM-SEARCH-LIST
```

Examples:

1. SHOW CONTROL-FILE BATCH-STREAM 0
2. SHOW MESSAGES
3. SHOW STATUS TAPE-DRIVE MTA1: /ALL

The first command lists the first 10 lines for the current job in batch stream 0. The second command lists all pending WTOR messages. The third command shows the status of MTA1, that is, shows the volume, label type, density, state of the drive, and other information.

SHUTDOWN

The SHUTDOWN command terminates the scheduling of job requests for the devices specified, as defined by the available keywords.

The format is:

SHUTDOWN keyword <stream/unit number/node name>

where keyword can be one of the following:

BATCH-STREAM
CARD-PUNCH
NODE
PAPER-TAPE-PUNCH
PLOTTER
PRINTER
READER

Using the keyword NODE shuts down all devices at the node.

Example:

SHUTDOWN BATCH-STREAM Ø:3

This command terminates scheduling of batch jobs for batch streams Ø, 1, 2, and 3.

START

The START command starts the scheduling of job requests for the devices specified, as defined by the available keywords.

The format is:

START keyword <stream/unit number/node name> [switch]

where keyword can be one of the following:

BATCH-STREAM
CARD-PUNCH
NODE
PAPER-TAPE-PUNCH
PLOTTER
PRINTER
READER

The PRINTER keyword has the following optional switch to allow LPTSPL to spool onto tape or drive a TTY:

/DEVICE: <device name> (usually an MTA or a TTY)

The NODE keyword takes a remote station node name or the host node name as an argument. START NODE is equivalent to START PRINTER Ø and START READER Ø at the specified node.

Example:

START PRINTER Ø:1

This command starts scheduling of the print jobs for printers Ø and 1.

STOP

The STOP command temporarily stops any device as defined by the available keywords. The CONTINUE command is used to resume the processing of job requests by the device.

The format is:

STOP keyword <stream/unit number> argument-1

where keyword can be one of the following:

BATCH-STREAM
CARD-PUNCH
PAPER-TAPE-PUNCH
PLOTTER
PRINTER
READER

and argument-1 can be:

AFTER <CURRENT-REQUEST, EVERY-REQUEST>
IMMEDIATELY

Example:

STOP PRINTER 0 IMMEDIATELY /NODE:DN200::

This command temporarily stops printer 0 at node DN200.

SUPPRESS

The SUPPRESS command suppresses the carriage control on the line printer. All lines are single-spaced. Blank lines and form feeds are ignored.

The format is:

SUPPRESS PRINTER <unit number> [switch]

The possible switches are:

/FILE	(for the current file)
/JOB	(for the current job)
/STOP	(to resume normal printing)

If no switch is specified, /JOB is assumed.

TAKE

The TAKE command executes a series of OPR commands from a specified command file.

The format is:

TAKE <filespec> [switch]

where filespec is the name of the command file. The optional switch may be either of the following:

/DISPLAY	Displays actions of TAKE file
/NODISPLAY	Does not display actions of TAKE file

WAIT

The WAIT command is invisible (that is, it does not appear when "?" is typed). It is used to delay for a specified time (using the DISMS JSYS).

The format is:

WAIT <number of seconds>

where the number of seconds must be in the range 1 to 6Ø.

The WAIT command is intended for use when running OPR in a batch job. The delay that it provides is necessary to allow action to be taken on previously entered OPR commands before the batch job ends.

UNLOCK

The UNLOCK command permits timesharing and batch jobs to access a particular structure that was previously locked.

The format is:

UNLOCK <structure-name> <date-time)

USER COMMANDS

The user components of Galaxy have also been improved. All user requests now run one of two programs: QUEUE or MOUNT. All requests for mounts and dismounts run the MOUNT program. You may notice that the UMOUNT program appears to still exist. In fact, this is the MOUNT program simply renamed. Both QUEUE and MOUNT communicate with Galaxy by sending IPCF messages to QUASAR and from there to the correct component (the QUEUE. UUO is not used because it blocks). The operator communicates to QUASAR through ORION, the user directly to QUASAR or the components. The relationship between monitor commands and the programs that they run is shown in the following chart:

PRINT QUEUE PLOT TPUNCH CPUNCH SUBMIT	Uses SYS:QUEUE.EXE
MOUNT DISMOUNT DEALLOCATE ALLOCATE	Uses SYS:MOUNT.EXE
CANCEL SHOW	Uses SYS:MOUNT.EXE or SYS:QUEUE.EXE

There are two new commands: CANCEL and SHOW. The CANCEL command removes user requests from the queues. The SHOW command displays the queues.

CANCEL Command

The CANCEL command removes requests from QUASAR's queues. It may also be used to cancel requests that are currently being processed. The format of the CANCEL command is:

```
.CANCEL keyword <request-id or job-name>
```

Keyword represents the request type to cancel and must be one of the following:

```
BATCH-REQUEST
CARD-PUNCH-REQUEST
MOUNT-REQUEST
PAPER-TAPE-REQUEST
PLOTTER-REQUEST
PRINTER-REQUEST
```

Request-id represents the individual request identifier and may be one of the following:

1. The request-id that was assigned to the request; this number is displayed with the SHOW QUEUE command.

2. The job name assigned to the request; this is a six character name and is available with the SHOW QUEUE command.

3. A wild-card job name may refer to several requests. The wild characters are "*" to match a wild string and "%" to match all characters in a character position. The standard wild card "?" is also accepted at monitor command level.

For example:

```
.PRINT SD.GEN/AFTER:17:00
[JOB SD QUEUED, REQUEST-ID 700, LIMIT 6]
.

.CANCEL PRINT 700
1 JOB CANCELED
.
```

SHOW Command

The SHOW command gives information about user requests in the system. By default, only information about the current job is shown, but it may be used to find out about other jobs also. The format of the SHOW command is:

```
.SHOW keyword /Switches
```

The keyword must be one of the following:

QUEUES	to list your job's queue requests
ALLOCATION	to list your job's resource allocation.

The possible switches are:

/ALL

Displays the requested information for all users on the system. Normally the SHOW command only displays information for your job.

/BRIEF

Displays a brief description of the requested information.

/FULL

Displays an expanded description of the requested information.

/USER:[p,pn]

Displays the requested information for a specified user.

SHOW QUEUE COMMAND

The format of the SHOW QUEUE command is:

.SHOW QUEUE keyword

The keyword must be one of the following:

ALL-REQUESTS
BATCH-REQUESTS
CARD-PUNCH-REQUESTS
MOUNT-REQUESTS
OUTPUT-REQUESTS
PAPER-TAPE-REQUESTS
PLOTTER-REQUESTS
PRINTER-REQUESTS

As is seen in the following example, the output of the SHOW QUEUE command is similar to the output produced by commands that perform the same function:

```
.SHOW QUEUE PRINT

PRINTER QUEUE:
JOB NAME  REQ_#  LIMIT              USER
________  _____  _____  _______________________
* A         707      6  GREELEY [77,4655]
THERE IS 1 JOB IN THE QUEUE
```

```
.PRINT

PRINTER QUEUE:
JOB NAME  REQ_#  LIMIT               USER
________  _____  _____  ____________________________________
* DBM129   677    222  ASG       [30,4571]      ON UNIT:0
  STARTED AT 14:25:13, PRINTED 53 OF 222 PAGES
  MEJIA     15    258  IRA       [30,5232]      /LOWER
  REPORT   688     36  ELWERTOWSKI [30,5516]
  A        707      6  GREELEY [77,4655]
THERE ARE 4 JOBS IN THE QUEUE (1 IN PROGRESS)
```

PLEASE Program

The PLEASE program has been modified to fit in with the rest of the Galaxy system. It sends IPCF messages directly to ORION, through which OPR can respond.

PLEASE is invoked using the PLEASE monitor command followed by optional switches and an optional line of text. There are two switches:

1. /HELP to display the help text

2. /NODE to specify the node where the messages should be sent. The line of text may be terminated by an ALTMODE to indicate that no reply is expected:

```
.PLEASE PLEASE MOUNT THE AISA: PACK <ALTMODE>
[PLSOPN OPERATOR AT KL1026(26) HAS BEEN NOTIFIED AT 9:19:49]
.
```

Several lines of text may be entered by omitting the text on the command line:

```
.PLEASE/NODE:76::
ENTER TEXT, TERMINATE WITH ALTMODE OR ^Z
MY RESPONSE HAS BEEN VERY SLOW. IS SOMEONE
RUNNING IN HPQ OR AM I BEING DUMPED ON BY THE
SCHEDULER? <ALTMODE>
[PLSOPN OPERATOR AT KS4101(76) HAS BEEN NOTIFIED AT 13:14:15]
.
```

A CTRL-Z will send the text and wait for a reply:

```
.PLEASE WHEN IS THE SYSTEM SPLITTING?
[PLSOPN OPERATOR AT KL1Ø26(26) HAS BEEN NOTIFIED AT 14:59:3Ø]
AT THREE O'CLOCK THIS AFTERNOON
ENTER TEXT, TERMINATE WITH ALTMODE OR ^Z
GULP!  NOT MUCH TIME FOR OUR HEROS. <ALTMODE>
[PLSOPN OPERATOR AT KL1Ø26(26) HAS BEEN NOTIFIED AT 14:59:5Ø]
```

Chapter 5
LABELED TAPES

WARNING

The information about Galaxy 4.1 and its related components (Usage Accounting, MDA, etc.) is subject to change. The release of Galaxy 4.1 has been decoupled from 7.Ø1 and will not occur until February 1981. Any information contained in this document about those features is preliminary and may change before the release.

Tape labels are used to identify magnetic tape volumes and the files contained on them. There are several kinds of labels on a labeled tape. The first block on a labeled tape contains a volume label (VOL) that identifies the volume and specifies the access allowed to it. Other volume labels supplying additional identification may follow. Following the volume label(s) are the files written on the tape. Each file consists of three parts:

1. One or more header labels (HDR) followed by a tape mark (TM).

2. The file data followed by a tape mark.

3. One or more trailer labels (EOF) followed by a tape mark.

The following is an illustration of a labeled tape with two files.

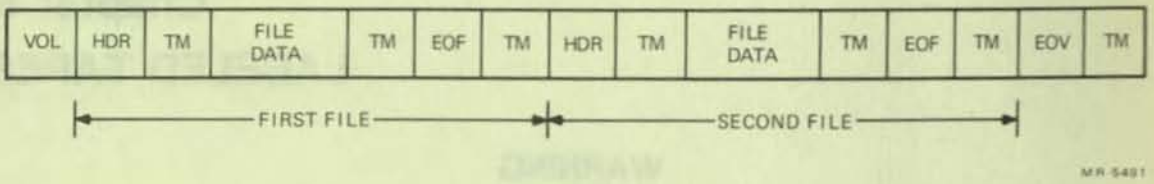


Figure 5-1 A Labeled Tape

Here, VOL, HDR, and EOF represent all volume, header, and trailer labels, respectively (regardless of the number of each). A tape mark is a hardware-recognizable bit pattern. Two adjacent tape marks indicate the logical end of tape.

Galaxy 4.1 can recognize two types of volume labels: ANSI and IBM. ANSI labels must conform to ANSI Standard X3.27, Magnetic Tape Labels, and File Structures for Information Interchange. IBM labels (often referred to as EBCDIC labels) must conform to IBM Specification GC26-3795-2 for OS/VS Tape Labels. These standards provide specifications for file sets, labels, and the record and block structures on these tapes.

EBCDIC volume labels contain a single record providing volume identification. There are two volume labels on an ANSI tape to provide extra identification and access control for the tape. Up to nine user volume labels (UVL) are allowed on ANSI tapes. These may contain any information about the volume that the user desires. EBCDIC tapes do not have UVLs. Galaxy does not allow users to read or write UVLs; if they are present, they are ignored.

The file header label(s) identify the file, indicating the file name, block size, logical record size, and data format, among other things. The trailer label contains the same information as the header label(s) except that it also contains the number of blocks in the file. Tapes written with ANSI labels use two header records to contain the file information for each file on tape.

Labeled tapes provide a definite security advantage over unlabeled ones. Because a labeled tape has the volume identifier for the tape written right on the tape (in the volume label), the operating system can recognize a tape by its label rather than relying on the operator to identify it. Conformance to standards in using labeled tapes also makes interchange of these tapes between systems more reliable; little or no allowance is required for one system's tape handling peculiarities.

MDA does not enforce the data written into a file to conform to the file attributes specified on the labels (i.e. record format, block size, record size, etc.). In other words, MDA supports label processing but does not support labeled tapes.

The price for the increase in reliability and security is a slight increase in overhead. MDA must do a little extra to recognize the tape by its label, to verify access to a file through its header label, and to write header and trailer labels. Reading and writing the data is essentially the same for labeled and unlabeled tapes.

For TOPS-1Ø and Galaxy to handle the recognition, label writing, and data formats associated with labeled tapes, several UUO changes have been made. Several of the functions for TAPOP. have been changed; several new functions have been added for this UUO. These changes are described in greater detail later in this chapter.

FILE ATTRIBUTES

Several attributes associated with tape files are not associated with disk files; the header and trailer labels for the tape files contain this attribute information. When reading a file, the attributes in the file header are used by the monitor for accessing the file data. The file attributes associated with tape files are record formats, block lengths, record lengths, and expiration dates. As mentioned above, MDA does not make the data in a file conform to the attributes in the label.

TAPE LABEL FORMATS

The TOPS-1Ø implementation of labeled tapes is based on DEC Standard 149, which describes tape labeling for all DEC systems (the DEC standard is, in turn, based on ANSI Standard X3.27). The TOPS-1Ø implementation defines several label fields that the DEC or ANSI standards describe as system- or implementation-dependent. In the following descriptions, fields defined by the DEC standard are marked with an at-sign (@), and fields defined in the TOPS-1Ø implementation are marked with an asterisk (*). All other fields are defined in the ANSI Standard X3.27.

TOPS-1Ø writes ANSI labels in ASCII characters. IBM labels are written in EBCDIC. All labels are 8Ø characters in length. Labels on ANSI tapes from other systems may be longer than 8Ø characters. The following are descriptions of these labels. Some differences between ANSI and EBCDIC labels are pointed out; other differences are described under "EBCDIC Tape Differences."

In the label descriptions, "CP" represents character position, "LEN" represents the length of a field (in characters), and an "a" character is an alphanumeric character (A-Z or Ø-9) or one of the following special characters:

space ! " % & ' ( ) * + , - / : ; < = > ? .

Where specific values are shown for the fields in these labels, the values shown are the ones written by the monitor on ANSI or EBCDIC tapes. For more information on label formats, see the TOPS-1Ø Tape Processing Manual (AA-5Ø84A-TB).

Volume Label (VOL1)

CP	FIELD NAME	LEN	CONTENTS
1 - 3	LABEL IDENTIFIER	3	"VOL"
4	LABEL NUMBER	1	"1"
5 - 10	VOLUME IDENTIFIER	6	Any 6 "a" chars Not all spaces
@ 11	ACCESSIBILITY	1	ANSI has " " , "1" EBCDIC has a "0"
12 - 37	RESERVED	13	Spaces
@ 38 - 50	OWNER IDENTIFIER	13	See below
@ 51	DEC STANDARD VERSION	1	"1"
52 - 79	RESERVED	28	Spaces
80	LABEL STD VERSION	1	"1"

The owner identifier field really identifies the installation; it is broken down in the following way for the TOPS-10 implementation:

CP 38-39	D%
CP 41-45	Machine Code
CP 41	Operating System Code, "T10 " for TOPS-10
CP 46-50	System Serial Number

The machine code in CP 4Ø indicates the type of system that wrote the volume label. The code used is according to the following table:

Machine Code	System
8	PDP-8
A	PDP-1Ø
B	PDP-11
C	VAX-11/78Ø
F	PDP-15
K	DECSYSTEM-2Ø

Auxiliary Volume Label (UVL1)

	CP	FIELD NAME	LEN	CONTENTS
*	1 - 3	LABEL IDENTIFIER	3	"UVL"
*	4	LABEL NUMBER	1	"1"
*	5 - 1Ø	ACCESS CODE	6	6-digit file set protection code
*	11 - 22	OWNER'S DIRECTORY	12	User PPN
*	23 - 61	OWNER'S NAME	39	User name or spaces for a scratch tape
*	62 - 8Ø	RESERVED FOR DEC	19	Spaces

Note that an auxiliary volume label is only used in the TOPS-1Ø implementation and is written only on ANSI label tapes. Consequently, all fields of the UVL1 label are marked with an asterisk.

The access code is the protection for files on this tape. It is only three characters long and is stored right justified in the six character field.

First File Header Label (HDR1)

CP	FIELD NAME	LEN	CONTENTS
1 - 3	LABEL IDENTIFIER	3	"HDR"
4	LABEL NUMBER	1	"1"
5 - 21	FILE IDENTIFIER	17	File name
22 - 27	FILE-SET IDENTIFIER	6	Same as volume identifier
28 - 31	FILE SECTION NUMBER	4	Starts at 0001 Incremented on volume switches
32 - 35	FILE SEQUENCE NUMBER	4	File number within file set Starts at 0001
36 - 39	GENERATION NUMBER	4	Always zero
40 - 41	GENERATION VERSION	2	Always zero
42 - 47	CREATION DATE	6	Julian date as " YYDDD"
48 - 53	EXPIRATION DATE	6	Julian date as " YYDDD"
@ 54	ACCESSIBILITY	1	"1" or " " for ANSI, "0", "1" or "3" for EBCDIC
55 - 60	BLOCK COUNT	6	"000000"
61 - 73	SYSTEM CODE	13	"DECSYSTEM-10 "
74 - 80	RESERVED	7	Spaces

Note the following about HDR1:

1. The file identifier includes the file name, a period (.), and the file extension.

2. The dates are in Julian form such that February 4, 1979 is represented as " 79Ø35".

3. If the access code is "1" (i.e., it is a ANSI label tape), access to the file is determined by the HDR2 file access field.

4. The system code shows the kind of system that wrote the file.

Second File Header Label (HDR2)

	CP	FIELD NAME	LEN	CONTENTS
	1 - 3	LABEL IDENTIFIER	3	"HDR"
	4	LABEL NUMBER	1	"2"
@	5	RECORD FORMAT	1	"F" = Fixed "D" = Variable "S" = Spanned "U" = Undefined
	6 - 1Ø	BLOCK LENGTH	5	Decimal number
	11 - 15	RECORD LENGTH	5	Decimal number
@	16 - 21	FILE ACCESS CODE	6	File protection
@	22 - 33	OWNER'S DIRECTORY	12	Owner PPN
@	34 - 36	RESERVED	3	Spaces
*@	37	FORM CONTROL	1	A, M, or space
*	38 - 5Ø	RESERVED	6	Octal number
	51	BUFFER OFFSET	1	"Ø"
	52 - 8Ø	RESERVED	28	Spaces

Note the following things about HDR2:

1. The undefined record format is only for use on ANSI label tapes. This format can be used for binary files.

2. The decimal and octal numbers are in character form. For example, if the block length is 4Ø96, it is recorded as "Ø4Ø96".

3. The DEC standard defines the form controls "A", "M", and " " (space). "A" indicates that the first character of the record is a FORTRAN control character. "M" indicates that the record contains all required form control (carriage return, line feed, etc.). A space indicates that the record contains no form control.

4. The file access code contains the same file protection information as is used for disk files.

User File Labels

CP	FIELD NAME	LEN	CONTENTS
1 - 3	LABEL IDENTIFIER	3	"UHL" or "UTL" as appropriate
4	LABEL NUMBER	1	Any "a" character
5 - 80	RESERVED FOR USER APPLICATIONS	76	Any "a" characters

User header labels begin with UHL; user trailer labels begin with UTL. Each such label can contain up to 76 characters of user information (with four identifying characters). Galaxy can not write UDLs and will ignore them on input.

Other Labels

File trailer (or end-of-file) labels contain the same fields as file header labels. Their only differences are in the identification fields (EOF1 instead of HDR1 and EOF2 instead of HDR2) and their block count fields. The block count in EOF1 contains the number of blocks of data in the file.

File trailer labels are used only for identification while reading a tape backwards. TOPS-2Ø makes no use of trailer labels since reading labeled tapes backwards is not supported.

End-of-volume labels are used instead of file trailer labels at the end of a volume for a multivolume file. These labels contain the same fields as file trailer labels. The only differences in these two labels are in the identification fields (EOV1 instead of EOF1 and EOV2 instead of EOF2) and their block count fields. The block count in EOV1 contains the number of blocks on this volume for this file.

When a file is spread over more than one volume, it is said to be in sections. A section is that portion of the file on one volume. Each section is numbered (CP 32-35 in HDR1) and has its own file header and user header labels.

EBCDIC TAPE DIFFERENCES

Release 4.1's support of labeled tapes is primarily designed to handle ANSI-type labels. Support of EBCDIC tapes was not a primary goal of Release 4.1. The TOPS-1Ø monitor does not include support for writing EBCDIC data on tapes; that must be handled by the user.

There are a number of significant differences between EBCDIC and ANSI. Not only are EBCDIC labels different, the implementation of the various record formats on them is different. MDA handles these differences when reading EBCDIC tapes and makes them invisible to the user.

The following list shows the fields of an IBM (EBCDIC) label that are different from other labels discussed in this chapter.

1. The accessibility field in VOL1 is always "Ø".

2. The owner identifier field in VOL1 is in CP 42-51.

3. CP 12-41 and CP 8Ø in VOL1 must contain blanks.

4. CP 16-36, CP 38, and CP 4Ø-52 in HDR2 contain miscellaneous special information and reserved fields.

5. CP 39 in HDR2 contains the block attribute field.

The following is a list of differences between the record formats described in this module and those used on EBCDIC tapes:

1. CP 39 in HDR2 indicates whether records in the file are blocked. If so, each block contains a four-byte block descriptor word (BDW) indicating the length of the block. The length is contained as a 16-bit binary number in the first two bytes; the other two bytes are zero.

2. The RCW in the variable records on EBCDIC tapes store the record length in a similar form, as a 16-bit binary value in the first two bytes; the other two bytes are zero.

3. The spanning indicator values are different for spanned records on EBCDIC tapes. Values Ø and 1 have the same meaning as for ANSI tapes. Values 2 and 3 have their meanings switched; for example, 2 means that the record ends but does not begin in the segment. The other four bytes of the SCW are in the same form as the RCW on EBCDIC tapes.

TAPE FILES

The ANSI, DEC, and IBM standards for labeled tapes define various forms for file sets and file attributes. For the most part, the DEC standard as implemented on TOPS-1Ø adheres to the ANSI standard level one. There are a few extensions to allow for processing unique to TOPS-1Ø.

File Sets

A file set is composed of one or more files recorded consecutively on a volume set. A volume set is composed of one or more tape volumes (reels) containing one and only one file set. TOPS-1Ø allows file sets to be stored on tape volumes in four ways:

1. Single file, single volume - one file on a tape.

2. Multifile, single volume - several files on a tape.

3. Single file, multivolume - one file spread across two or more tapes.

4. Multifile, multivolume - several files written on several tapes. Files may be spread across more than one volume or on a single volume within the volume set.

A file set may span several tape volumes. The file set name written into the HDR1 label is the volume identifier for the tape volume containing the first file of the file set. This file set nᵃme can be used to check the volumes for membership in the file set.

OPERATIONS

Labeled tape operations can be explained by considering their several functions individually. These functions include setup of tape control, initialization of tapes, mounting and dismounting tapes, tape I/O, and tape positioning. These functions are described below, with an explanation of the commands used to obtain information about tape drives and file sets.

Setup

Setting up the monitor to handle labeled tapes consists of allowing MDA to control the tape drives and specifying whether automatic volume recognition (AVR) is being used. Giving MDA control of the tape drives is accomplished by issuing the following command to OPR:

SET (TAPE-DRIVE) MTxxx AVAILABLE

Without giving this command, tapes can be treated only as unlabeled and must be assigned by the user. (The ENABLE STRUCTURE-RECOGNITION or ENABLE VOLUME-RECOGNITION DISK-DRIVE is used to turn on AVR for disk drives.)

In trying to recognize a tape, PULSAR reads the first block and goes through the following checks, in the following order.

1. If a tape mark is read or there is no information in the first block, the tape is considered unlabeled.

2. If the first block contains exactly 80 characters, the first four of which are "VOL1" in EBCDIC, the tape is considered to have an EBCDIC label.

3. If the first block contains 80 or more characters, the first four of which are "VOL1" in ASCII, it is considered to be label type ANSI.

4. Otherwise, it is unlabeled.

If AVR is not enabled on some drives, PULSAR does not attempt to recognize a tape when it is made ready on those drives. The operator must identify these tapes by using the RECOGNIZE command of OPR to force label processing on a drive. Tape drives can be taken from MDA's control with the OPR command SET TAPE-DRIVE UNAVAILABLE. Tape drives not under MDA's control are treated as ASSIGNable devices.

Initialization

– can specify increment for volid to init many tapes successively

All tapes start out unlabeled. Before a tape can be regarded as a ANSI or EBCDIC labeled tape, some labels must be written on it. In particular, the volume label(s) must be written. A volume label holds two vital pieces of information about the tape: a label type and a volume identifier. A labeled tape is recognized by these two items.

The operator labels a tape by issuing the following OPR command:

```
OPR> SET TAPE-DRIVE <MTAn:> INITIALIZE <switches>
```

The tape on the indicated MTA is labeled. PULSAR can label a tape with ANSI or EBCDIC labels or write two tape marks to initialize an unlabeled tape.

The switches allow the operator to label one or more tapes at a time, giving them consecutive or non-consecutive volume identifiers. The operator can specify the label type, volume identifier, volume protection, tape owner, density, and whether to keep the tape mounted after the labels are written. See Chapter 4 for the appropriate switches to make these specifications. See the <u>TOPS-10 Operator Command Language Reference Manual</u> (AA-H599A-TB) for a complete explanation of this command.

When a tape is initialized, the volume label(s) are written, followed by a dummy file (named DUMMY-FILE-0000), which contains no data. It consists of a set of header labels followed immediately by a set of trailer labels. This file is usually overwritten on the first tape write.

Mounting and Dismounting Tapes

MOUNTS

If MDA is not given control of the tape drives, the mounting and dismounting of tapes is handled as in Release 2 using OMOUNT and UMOUNT. Without PULSAR, tape drives must be assigned and deassigned by the user. Even tapes with labels are treated as unlabeled. The user sees the labels as extra files on the tape.

The usual case for Release 4.1 is to give MDA control of the tape drives. To get a tape mounted, the user gives a mount request with the following monitor command.

(valid list)

.MOUNT volume-set-name: logical-name: /Qualifers

The volume set name is a logical name by which the user can refer to a tape volume or set of volumes containing one file set. The switches indicate the volume(s) to be mounted and how they are to be treated. The following is a list of the switches for the MOUNT command. Note that the resource represents the name of the resource to be MOUNTED and must be one of the following.

A disk volume set name such as DSKB:

A tape volume set definition such as PAY-WK43(PM43A,PM43B):

A tape volume identifier for a single volume TAPE volume set

A logical-name previously associated with a resource

A physical device name

Logical-name represents the name your program will use to reference the resource. If this field is not specified and the resource was a tape volume set defintion, a default will be generated by truncating the volume set name to six characters and truncating the result before the first non-alphanumeric. For example, PAY-WK43 would produce a default logical name of PAY.

/WAIT -- Ensures that the MOUNT operation is complete before returning to command level. This is the default for BATCH jobs.

/NOWAIT -- Allows you to return to command level before the MOUNT operation is complete. This is the default for all non-BATCH jobs.

/CHECK -- Will type the status of MOUNT requests for your job.

/HELP -- Will type a description of the MOUNT command.

/TAPE -- Indicates that the resource is a tape volume set.

/D SK -- Indicates that the resource is a disk volume set.

/EXCLUSIVE -- Ensures that no other users are allowed to share the resource. This is the default for tape volume sets.

/SHARABLE -- Allows other users to share the resource. This is the default for disk volume sets.

/READ-ONLY -- Ensures that the resource is not write enabled when it is mounted. This is the default for tape volume sets.

/WRITE-ENABLE -- Ensures that the resource is write enabled when it is mounted. This is the default for disk volume sets.

/REMARK:text -- Allows a remark up to 5Ø characters in length to be sent to the operator with this mount request.

/ACTIVE -- Ensures that the disk volume set is placed in your jobs active search list. This is the default for disk mount requests.

/PASSIVE -- Requests that the disk volume set be placed in your jobs passive search list.

/CREATE -- Allows creation of files on a disk volume set when generic device DSK: is specified. Note that files may still be created if it is explicitly required by name.

/NOCREATE -- Prohibits creation of files on a disk volume set when generic device DSK: is specified. Note that files may still be created if it is explicitly requested by name.

/DENSITY:<16ØØ-BPI, 8ØØ-BPI, 556-BPI, 2ØØ-BPI, 625Ø-BPI> -- Specifies the density of the tape.

/LABEL-TYPE:<ANSI,BYPASS,EBCDIC,IBM,NOLABELS,NONE,UNLABELED, BLP> -- Specifies the label type.

ANSI - for ANSI labels
IBM, EBCDIC - for EBCDIC labels
NOLABELS, NONE - for an unlabeled tape

BYPASS,BLP - MDA does not do label processing (privileged users only)
UNLABELED - for an unlabeled tape (user sees EOT)

/NEW-VOLUME-SETS -- Same as /SCRATCH - does not need a REELID to be specified.

/REELID: (arg, arg, ...) -- States that the identifier on the outside of the volume is arg.

/RONLY -- See the /READ-ONLY switch.

/SCRATCH -- Asks for a scratch tape to be mounted without needing to specify a REELID.

/SINGLE -- Same as /EXCLUSIVE.

/VID -- See the /REMARK switch.

/TRACKS: <7-TRACK, 9-TRACK> -- Specifies whether the tape drive has seven or nine tracks.

/VOLID -- Declares the volume id.

/WENABLE -- See the /WRITE-ENABLE switch.

/WLOCK -- Write-protect the structure.

Mount requests made with MOUNT form the MOUNT-REQUEST queue. The operator may satisfy requests in this queue in any order he/she wishes.

A mount request may be canceled by a user using the CANCEL command:

.CANCEL MOUNT <vol. set name/request #>

If a user gives a mount request and uses CTRL/C to get out of the blocked state, the request is still pending. Only CANCEL can get rid of it. Note that there is no colon as part of the volume set name in the CANCEL command.

To find out which mount requests are pending, the user can give the monitor command:

.SHOW QUEUES MOUNT-REQUESTS or .MOUNT/CHECK

This lists all mount requests pending and being serviced, including requests for structures; it shows the volume set name, user, volid, the write status (locked or enabled), the type (tape or structure), and the status of the request. The status may be "WAITING" (tape not yet mounted).

ASSOCIATING REQUESTS WITH TAPE

The information in the mount request is sent to QUASAR, which associates it with a tape mounted on a drive. If the request is for a labeled tape, the volid is given or implied, and AVR is being used, QUASAR makes the association automatically. If any of these conditions is not true, the operator must tell QUASAR which tape (and drive) to associate with the request, using the OPR command:

```
OPR> IDENTIFY <MTAn:> REQUEST-ID <request number>
```

When QUASAR makes the association between request and tape, it notifies the user which device is assigned to him/her. If the mount request did not specify /NOWAIT, this message gets the user out of the blocked state and returns the user to monitor level. If the user did specify /NOWAIT, the message is sent to the user when he/she is at monitor level.

From the time the user receives the notification message until he/she dismounts the tape(s), the tape drive(s) assigned by PULSAR are under the user's control and may be handled in any way the user sees fit.

VOLUME SWITCHING

If a user has several volumes in a volume set, he/she does not need to worry about the switching between volumes of a set. Switching is handled by the monitor. If a volume is being read and the end of the volume is reached, the operator is notified that the next volume in the set should be mounted. Depending on the circumstances, the operator may have to identify the volume, or AVR may handle it. PULSAR can then make the association and reading can continue. This action is invisible to the user.

If the user program is writing and the EOT marker is reached, the operator is similarly notified. If the user has not made provision for extending the volume set by defining an additional volid, the request to the operator is for a scratch tape of the correct label type (up to sixty volumes). This scratch tape is then made part of the user's volume set.

The operator should notify the user if a volume set is extended with a scratch tape; without this notification, the user may not realize that the extension took place. If the user specifies /NOTIFY, he/she will be informed automatically of the volume switch and the volume-id of the current volume.

DISMOUNTS

When the user is finished with a volume set, he/she can use the command:

.DISMOUNT resource

where resource is the volume set name, logical device name or physical device name. This command deassigns the MTA and returns it to MDA's control. The DISMOUNT makes the resource unavailable to your job until another MOUNT command is issued. If the /REMOVE switch is used with this command, the operator is notified that the user wishes the device to be physically dismounted.

DISMOUNT causes the tape to be unloaded from the drive.

The DEASSIGN command can also be used to dismount a tape. Using the following commands is equivalent to DISMOUNTing the volume set:

.DEASSIGN <Logical-name or MTn:>

The operator should make sure that he/she does not manually unload any tape drive. With MDA in control, all unloading should be done with the software (DISMOUNT in OPR). Manual unloading confuses MDA especially when a new tape is mounted.

attribute is specified, the search for that position may be forward or backward. Any volume switching required by a search is handled automatically.

USER I/O PROGRAMMING

Access

Access to labeled tapes is not checked when the tape is mounted; that function is performed when the user attempts an INPUT or OUTPUT. There are two checks performed. First, the access code field in the auxiliary volume label (UVL1) is checked to make sure the user can use the tape. Secondly, when the first INPUT is done on a file, the access code of the file, obtained from the second file header label (HDR2), is checked to make sure the file may be read.

Reading and Writing

The actual reading and writing of user data is controlled by the user program. PULSAR is responsible for reading and writing the file headers.

The implementation of labels for Galaxy 4.1 differs from the ANSI standard. As mentioned before, MDA supports label processing but not labeled tape. Data is not checked to make sure that it conforms to the information written in the labels; because of this, very little file information is automatically written into the HDR1 and HDR2 labels. This does not make tape labeling as convenient as possible.

The solution to this problem requires that the user take an active part in the creation of files on tape. MDA and the monitor do not automatically supply the information that is stored in the HDR1/HDR2 labels but the user may via the .TFLPR function of the TAPOP. monitor call. If file attributes are supplied before the INPUT or OUTPUT, MDA will use them to perform the appropriate function.

There are two parameters that may be used to get to a specific file on a labeled tape: the file name and the file sequence number. The file sequence number is the position of the file relative to the beginning of the volume-set.

new TAPOP. functions

When neither a file name nor sequence number is specified, the following action is taken:

1. INPUT - the volume-set is opened at the current file position and no checking is done on the file name.

2. OUTPUT - the file and file-headers are written using the default file name.

If a file name is specified but the sequence number is not, the following action is taken:

1. INPUT - the volume-set is rewound and a search through the volume-set is made for a matching file. If the file is found, it is opened for input. If it is not found, the "FILE NOT FOUND" error code is set and control is returned to the user program.

2. OUTPUT - the volume-set is rewound, and a search through the volume set is made for a matching file. If the file is found, the file is opened for output. If the file is not found, the file is opened for output at the end of the volume-set.

If the sequence number is specified without a file name, the following action is taken:

1. INPUT - the volume-set is positioned at the location indicated by the sequence number. If a file exists there, it is opened for input. If a file does not exist there, the positioning error return is given.

2. OUTPUT - the volume-set is positioned at the location indicated by the sequence number. If a file exists there, it is opened for output and a default file name is written. If a file does not exist at that location but the file sequence number was equal to 99,999 or equal to the number of files contained in the volume-set plus one, the file is opened at the logical end of tape and a default file name is written.

When both the file name and sequence are specified, the following action is taken:

1. INPUT - the volume-set is positioned at the location indicated by the sequence number. If a file exists, a comparison is made between the file name specified in the TAPOP. and the name in the label. If they match, the file is opened for input. If they do not match, a file positioning error is returned.

2. OUTPUT - the volume-set is positioned at the location indicated by the sequence number. No check is made for a matching file name. The file is opened for output, and the file name specified supersedes any file name that may exist.

Closing

When the user issues a CLOSE monitor call, one of two things happen:

1. If input is done to a file, the tape is positioned to the beginning of the next file. If there are no more files, the tape is left positioned end of user data for the file that was located.

2. If output was being done, end-of-file labels are written after the user data. If a RELEAS monitor call is then issued, end-of-volume labels are written on the tape.

TAPE-RELATED COMMANDS

In previous releases, the monitor has been responsible for the monitor commands BACKSPACE, SKIP, EOF, UNLOAD, and REWIND; but with Galaxy 4.1, this arrangement has changed. If MDA is used with a labeled tape and the tape is not mounted in bypass mode, PULSAR will control the operation of these monitor commands. If MDA is not used or the tape is mounted in bypass mode, the commands are handled by TOPS-1Ø as they always were.

MDA will do the following for each command:

1. UNLOAD - This command only rewinds the tape. User UNLOAD commands would only confuse MDA and cause the wrong volume of a volume-set to be mounted.

2. REWIND - The REWIND command will position the tape at the beginning of user data on the tape; the volume labels and the header labels will be skipped over.

3. EOF - This command will skip over the remainder of data in the file and position the tape to the beginning of the next file. If there are no more files, the tape is positioned at the end of data on the last file.

4. SKIP and BACKSPACE - Both of these commands are under the control of PULSAR when labeled tapes are used. PULSAR handles volume switches when these commands cause the tape to reach EOF or BOT. PULSAR also positions the tape to the next/previous file if HDR labels are encountered.

The DIRECT command has been modified to read labeled tapes. It can handle both ANSI and EBCDIC labels. Here is an example:

```
.MOUNT HPPØØ2:/READ/LABEL:IBM/REELID:HPPØØ2
[MOUNT REQUEST HPPØØ2 QUEUED, REQUEST-ID #26[
[TAPE VOLUME HPPØØ2 MOUNTED ON MTA76Ø AS LOGICAL NAME HPPØØ2]

.DIRECT HPPØØ2:

 READ DENSITY:16ØØ  PARITY:ODD  9-TRACK  READ ONLY  REELID:HPPØØ2

 LABELED TAPE FILE INFORMATION:
   FILE NAME:"IMPRESS              ", GENERATION:Ø, VERSION:Ø
   PROTECTION:ØØØ, CREATION DATE:23-FEB-76, EXPIRATION DATE:(UNDATED)
   BLOCK SIZE:2741, RECORD SIZE:16Ø
   RECORD FORMAT:UNDEFINED, FORM CONTROL:NONE
   UNKNOWN FORMAT: LENGTH =4689
```

```
LABELED TAPE FILE INFORMATION:
  FILE NAME:"BIGCAT                ", GENERATION:0, VERSION:0
  PROTECTION:000, CREATION DATE:27-AUG-76, EXPIRATION DATE:(UNDATED)
  BLOCK SIZE:800, RECORD SIZE:80
  RECORD FORMAT:FIXED, FORM CONTROL:NONE
```

Chapter 6
MOUNTABLE DEVICE ALLOCATOR

WARNING

The information about Galaxy 4.1 and its related components (Usage Accounting, MDA, etc.) is subject to change. The release of Galaxy 4.1 has been decoupled from 7.Ø1 and will not occur until February 1981. Any information contained in this document about those features is preliminary and may change before the release.

The Mountable Device Allocation (MDA) is a set of functions implemented in Galaxy 4.1 to control mountable devices. MDA is designed to prevent the deadlock conditions that may arise in a timesharing environment when two or more jobs are competing for the same set of resources. MDA is not a specific program, rather it refers to a group of related programs that provide the desired functions.

INTRODUCTION

MDA is organized around several concepts and structures. The first of these is the volume and volume-set. These terms have been used in the previous chapter. A volume is a unit of storage medium, such as a tape or disk. A volume-set is a set of volumes that the user may treat logically as one volume. Examples of this would be a sequential set of data that resides on one large tape or a disk structure that takes up two packs. The name of a volume-set refers to all the volumes in the volume-set, whether there are one or more parts in it.

Each volume requires a label (a machine-readable set of data on a volume, identifying the volume). For disks, this is the home block; for tapes, it is the tape label. MDA uses labels to recognize a volume when it is mounted. When the operator puts a tape on the drive or mounts a disk pack, the device will interrupt TOPS-10. The monitor informs MDA (in this case, QUASAR) and PULSAR will read the label from the new volume. QUASAR will then attempt to process the user request corresponding to the mounted structure. This process is known as automatic volume recognition. For disks, the volume may be automatically defined as a structure to the monitor. This process is known as structure recognition. MDA can handle unlabeled volumes but it is designed to function with ones that are labeled.

DEADLOCKS

Deadlocks over volume usage can occur in several forms. One is the deadly embrace, where one user has resource A and needs resource B while another user has resource B and needs resource A. A more common form has a batch job waiting and waiting for a resource to be freed up, but that resource has been mounted by another job and is not being used. The goal of MDA is to minimize the time that jobs wait for resources and the time resources remain idle.

AVOIDING DEADLOCKS

The MDA system is designed around the premise that a user requires a group of mountable devices to perform a task, but that these devices are not necessarily all used at once. The user informs the system of all volume-sets that will be required for his task.MDA will then selectively permits MOUNTs, to avoid deadlocks. Two basic steps must be performed to use mountable devices: the first step is to allocate all of the devices that will be needed, and the second step is to MOUNT the devices as they are needed.

The allocation step allows the system to ensure that when the devices and volumes are needed later on, the job will not have to wait forever (literally) to get them. This situation could occur if another job has mounted the device or volumes requested and also is waiting for the requesting job to release some of the devices or volumes which it has mounted. It is important to note that allocation does not 'assign' or 'reserve' devices or volumes for that job. It merely notifies MDA that they will be needed at a later time.

Allocation is performed using a new monitor command, ALLOCATE. This command runs the MOUNT program and takes the following form:

```
.ALLOCATE volume-set-name logical-name /switches
```

The logical name will always be truncated to six characters. ALLOCATE uses the exact same set of switches as does the MOUNT command.

For example

```
.ALLOCATE WEEKLY-PAYROLL:  PYRLL
```

In this case, WEEKLY-PAYROLL will be truncated to WEEKLY. For disks, the volume set name is the structure name.

Most of the time, the ALLOCATE command returns immediately since it merely notifies Galaxy of future requirements. When you begin a job, allocate all resources necessary for the entire session using one or more ALLOCATE commands. Once allocation has been made, the use the logical names defined in the ALLOCATE command.

After allocation of the required devices and volumes, they can be MOUNTed and DISMOUNTed as needed. MOUNTing a device that will not be needed for another hour is an undesirable practice, since it ties up the device and prevents others from using it.

Issue the MOUNT command when you need the resource immediately. when the resource is needed immediately. The MOUNT command, unlike the ALLOCATE command, usually requires you to wait. MOUNT usually implies that the operator must perform some action and the command does not complete until the action has been completed.

The new form of the MOUNT command is similar to that of the ALLOCATE command. Chapter 5 contains a complete description of the MOUNT command. If a MOUNT command is issued for a device not already ALLOCATEd, the MOUNT command does an implied ALLOCATE.

Once the resource has been ALLOCATEd and MOUNTed, you may use it. If you do not presently need the resource, use a DISMOUNT command. This command will not remove the allocation for the resource. The complete form of the DISMOUNT command is discussed toward the end of Chapter 5.

When the resource is no longer needed, the user issues a DEALLOCATE command to free the resource. The format of the DEALLOCATE command is:

```
.DEALLOCATE resource
```

Resource represents the name of the resource to be DEALLOCATEd and must be either the volume set name, the physical name of the device, or the logical name of the device. DEALLOCATE makes the resource unavailable to your job until the resource is allocated to your job explicitly with the ALLOCATE command or implicitly with the MOUNT command. DEALLOCATE will DISMOUNT any resource that is currently mounted.

Here is an example of the effects of various MOUNT, DISMOUNT, ALLOCATE and DEALLOCATE commands.

```
.R QUOLST

User:   77,4655
Str     used   left:(in)  (out)   (sys)
DSKB:     20     4980      1980   10490

.SHOW ALLOCATION

Allocation for job 14 GREELEY [77,4655]

           Resource          Allocated   Mounted
        ________________     _________   _______
             DSKB                1          1
```

```
.DISMOUNT DSKB:
[Volume Set DSKB has been Dismounted]

.SHOW ALLOCATION

Allocation for job 14 GREELEY [77,4655]

              Resource              Allocated  Mounted
       --------------------         ---------  -------
       DSKB                             1         Ø

.R QUOLST

User:  77,4655
Str    used    left:(in) (out)  (sys)

.DEALLOCATE DSKB:
[Allocate Request DSKB Queued, Request-ID #14]

.SHOW ALLOCATION

Allocation for job 14 GREELEY [77,4655]

              Resource              Allocated  Mounted
       --------------------         ---------  -------

.R QUOLST

User:  77,4655
Str    used    left:(in) (out)  (sys)

.MOUNT DSKB:
[Mount Request DSKB Queued, Request-ID #14]
[Structure DSKB Mounted]

.SHOW ALLOCATION

Allocation for job 14 GREELEY [77,4655]

              Resource              Allocated  Mounted
       --------------------         ---------  -------
       DSKB                             1         1

.MOUNT DSKA:
[Mount Request DSKA Queued, Request-ID #21]
[Structure DSKA Mounted]
```

```
.SHOW ALLOCATION

Allocation for job 14 GREELEY [77,4655]

          Resource          Allocated   Mounted
     ------------------     ---------   -------
     DSKB                       1          1
     DSKA                       1          1
```

OPERATOR CONTROL

Although MDA has been designed to function with as little operator involvement as possible, there are times when the operator must control the specific volumes that are mounted. OPR, the new operator interface in Galaxy 4.1, makes this task easier than in previous versions.

As discussed in Chapter 1, the operator has the power to:

1. Decide whether disk and tape drives will be automatically controlled. This is done via the SET TAPE-DRIVE command or the ENABLE VOLUME-RECOGNITION command.

2. Choose which requests are to be serviced by using the SHOW QUEUE MOUNT command.

3. Decide to honor certain requests or defer processing of specific requests.

4. Force recognition of labeled tapes using the RECOGNIZE command.

All these features are controlled with only one program, OPR. OPR also provides many other useful operator functions.

The OPR program also provides an easy way to gain all the necessary information about MOUNT requests and the devices that can be mounted. Some of the information available to the operator is shown in the following examples.

```
OPR> SHOW STATUS STRUCTURE

OPR>
13:Ø3:22              -- Disk File Structures --

Name  Time   Free   Mount  Volume     Type Drive
----  ----  ------  -----  ---------  ---- -----
BLKX  4:51   2Ø322      2  BLKXØ  1/1 RPØ4  RPA3
DSKC  4:51   77565     51  DSKCØ  1/2 RPØ6  RPDØ
                           DSKC1  2/2 RPØ6  RPB2
DSKB  4:51   5944Ø     53  DSKBØ  1/2 RPØ6  RPD4
                           DSKB1  2/2 RPØ6  RPD7
BLKY  4:51   22812      2  BLKYØ  1/1 RPØ4  RPEØ
BLKK  4:51  14198Ø      1  BLKKØ  1/1 RPØ6  RPB1
GALØ  8:Ø8   18922      3  GALØØ  1/1 RPØ4  RPA1
SIRS  8:56   16675      1  SIRSØ  1/1 RPØ4  RPB6
GALT 11:16      2Ø      1  GALTØ  1/1 RPØ6  RPB3
Total of 8 file structures
(8 mounted, Total of 357736 free blocks)


OPR> SHOW STATUS STRUCTURE GALØ:/USER

OPR>
13:Ø3:4Ø              -- Disk File Structure --

Name  Time   Free   Mount  Volume     Type Drive
----  ----  ------  -----  ---------  ---- -----
GALØ  8:Ø8   18922      3  GALØØ  1/1 RPØ4  RPA1
 Users: Job 23 User DPM               [1Ø,56]
        Job 31 User GALAXY DEV        [5Ø,12]
        Job 44 User CORNELIUS,D       [5Ø,12]
```

```
OPR> SHOW STATUS DISK-DRIVES

OPR>
13:04:10            -- Disk Drive Status --

Drive  Type   Status       AVR  STR   Volume  Unit_#
-----  ----   ---------    ---  ----  ------  ------
RPA1   RP04   Mounted      Yes  GAL0  GAL00     0
RPA3   RP04   Mounted      Yes  BLKX  BLKX0     0
RPA5   RP06   Mounted      Yes  EGFX  602319    0
RPB1   RP06   Mounted      Yes  BLKK  BLKK0     0
RPB5   RP06   Mounted      Yes  HAL9  HAL9      0
RPB6   RP04   Mounted      Yes  SIRS  SIRS0     0
RPD0   RP06   Mounted      Yes  DSKC  DSKC0     0
RPB2   RP06   Mounted      Yes        DSKC1     1
RPD3   RP04   Mounted      Yes  ROTT  ROTT0     0
RPD4   RP06   Mounted      Yes  DSKB  DSKB0     0
RPD7   RP06   Mounted      Yes        DSKB1     1
RPE0   RP04   Mounted      Yes  BLKY  BLKY0     0
RPA2   RP04   Free         Yes
RPB0   RP04   Free         Yes
RPE1   RP04   Free         Yes


OPR> SHOW STATUS TAPE-DRIVE

OPR>
13:04:34            -- Tape Drive Status --

Drive   Status       AVR  Write    Volume  Job#   User
------  -----------  ---  -------  ------  ----   ----------------
MTA260  Free         Yes
MTA261  Free         Yes
MTB260  Unavailable  No
MTB261  Online       Yes  Enabled  SCRATC   35    KING, B [30,5420]
MTB262  Free         Yes
MTB263  Unavailable  No
MTB264  Online       Yes  Enabled  ALEC
```

The information shown above can all be obtained easily by using the SHOW command in OPR.

Chapter 7

INSTALLATION AND DEBUGGING

WARNING

The information about Galaxy 4.1 and its related components (Usage Accounting, MDA, etc.) is subject to change. The release of Galaxy 4.1 has been decoupled from 7.01 and will not occur until February 1981. Any information contained in this document about those features is preliminary and may change before the release.

GALAXY INSTALLATION

The changes to Galaxy have made Versions 2 and 4.1 incompatible. There has been a dramatic rearrangement of components, the functions within the components, and the structures of the queues. To switch to the new Galaxy system, you must perform the following steps.

1. Completely shut down the old Galaxy system in an orderly way.

2. Copy the old Galaxy components to OLD:.

3. Delete the master queue files (and the backup copies) from [3,3].

4. Delete the old Galaxy components from SYS:.

5. Copy the new Galaxy components to SYS:. They are:

   OPR.EXE
   QUASAR.EXE
   ORION.EXE
   LPTSPL.EXE
   BATCON.EXE
   SPRINT.EXE
   SPROUT.EXE

CDRIVE.EXE
PULSAR.EXE
MOUNT.EXE
QUEUE.EXE
GLXLIB.EXE
PLEASE.EXE
QMANGR.EXE

This procedure is satisfactory if the default values are acceptable. If any changes are to be made, you must perform two other steps: run the GALGEN program to generate the new default values, and recompile all the Galaxy components.

Running GALGEN

The GALGEN program generates the GALCNF.MAC file, which contains parameters of the Galaxy system components. GALCNF.MAC is assembled with the other Galaxy modules to create the Galaxy system. GALGEN has been made easier to use through the addition of the COMND JSYS simulator to parse the entries and provide explanatory text. You can list this text for all questions by specifying the LONG option, or for individual questions by entering "?". The dialogue in the short form will look like this:

```
.R GALGEN

GALGEN VERSION 4(2ØØ3)

[STARTING GALAXY GENERATION PROCEDURE FOR TOPS-1Ø SYSTEM]
[WRITING  GALAXY CONFIGURATION FILE DSKB:GALCNF.MAC[77,4655]]

IN THE FOLLOWING DIALOG, ALL QUESTIONS ARE OF THE FORM:

  TEXT OF QUESTION (LIST OR RANGE OF ANSWERS) [DEFAULT ANSWER]

THE LINE STARTS WITH THE ACTUAL TEXT OF THE QUESTION.
FOLLOWING THE QUESTION IS A DESCRIPTION OF THE POSSIBLE
ANSWERS ENCLOSED IN PARENTHESES. THIS DESCRIPTION MIGHT BE
A RANGE OF NUMBERS, A LIST OF KEYWORDS, OR A TEXTUAL
DESCRIPTION. FOLLOWING THIS DESCRIPTION IS THE DEFAULT
ANSWER (WHAT WILL BE ASSUMED IF YOU SIMPLY PRESS THE RETURN
KEY.)
```

```
YOU  HAVE  THE  CHOICE  OF CARRYING ON A LONG DIALOG IN WHICH AN
EXPLANATION OF EACH QUESTION IS  PROVIDED  AUTOMATICALLY,  OR  A
SHORT  DIALOG  IN WHICH IT IS ASSUMED THAT YOU ARE FAMILIAR WITH
THE GALAXY GENERATION PROCEDURE.  IN EITHER CASE, THE HELP  TEXT
CAN  ALWAYS  BE  GOTTEN  BY  TYPING A QUESTION MARK ("?") AS THE
RESPONSE TO ANY QUESTION.

ANSWER THE FOLLOWING QUESTION EITHER LONG (FOR A LONG DIALOG) OR
SHORT  (FOR A SHORT ONE).   SIMPLY PRESSING THE  RETURN KEY WILL
IMPLY SHORT.

DIALOG LENGTH(SHORT,LONG) [SHORT] SHORT


      ADMINISTRATIVE CONTROLS AND PARAMETERS
      -------------- -------- --- ----------

OPERATOR LOG FILENAME(3-6 CHARCTERS) [ORNLOG]
MAXIMUM PRIORITY NON-PRIVILEGED USERS MAY SPECIFY(1Ø-62) [2Ø]
DEFAULT FOR /PRIORITY ON BATCH AND SPOOLING REQUESTS(1-62) [1Ø]
NUMBER OF MINUTES BETWEEN CHECKPOINTS(1-1Ø) [1]
DO YOU WANT REDUNDANT MASTER QUEUE FILE(YES,NO) [NO]
FILE-STRUCTURE TO USE FOR MASTER QUEUE(3-6 CHARACTERS) [SYS]


               BATCH JOB DEFAULTS
               ----- --- --------

DEFAULT BATCH JOB RUNTIME LIMIT(5-9999 SECONDS) [3ØØ]
DEFAULT LINEPRINTER OUTPUT LIMIT(5-9999 PAGES) [2ØØ]
DEFAULT CARD-PUNCH OUTPUT LIMIT(5-9999 CARDS) [1ØØØ]
DEFAULT FOR BATCH /FEET: SWITCH(1Ø-5ØØØ) [2ØØ]
WHAT IS THE MAXIMUM PLOT TIME A BATCH JOB MAY GENERATE(1Ø-5ØØØ MINUTES) [2ØØ]
DEFAULT FOR BATCH /OUTPUT SWITCH(LOG,NOLOG,ERROR) [LOG]
SHOULD MEMORY LIMITS BE ENFORCED(YES,NO) [NO]
```

LINEPRINTER DEFAULTS AND PARAMETERS

```
NUMBER OF JOB BANNER PAGES(Ø-5) [2]
NUMBER OF JOB TRAILER PAGES(Ø-5) [2]
NUMBER OF FILE HEADER PAGES(Ø-5) [2]
NAME FOR STANDARD OUTPUT FORMS(4-6 CHARACTERS) [NORMAL]
NUMBER OF CHARACTERS WHICH UNIQUELY IDENTIFIES FORM(2-6) [4]
```

MISCELLANEOUS DEFAULTS AND PARAMETERS

```
DO YOU WANT DEFAULT LIMIT COMPUTATION(YES,NO) [YES]
DEFAULT OUTPUT LIMIT EXCEEDED ACTION(ABORT,ASK,IGNORE) [ASK]

[END OF GALGEN DIALOG]
```

If help is needed during the short dialogue, typing a "?" will print out the long form of the question:

```
DEFAULT BATCH JOB RUNTIME LIMIT(5-9999) SECONDS) [3ØØ] ?

THE BATCH USER CAN SPECIFY A MAXIMUM RUNTIME FOR HIS BATCH
JOB USING THE /TIME SWITCH. IF HE DOES NOT SPECIFY THIS
SWITCH, THE FOLLOWING DEFAULT IS USED.

DEFAULT BATCH JOB RUNTIME LIMIT(5-9999 SECONDS) [3ØØ]
```

The GALCNF.MAC file contains the questions asked (as comments), as well as the instructions generated.

Recompiling Galaxy Components

Galaxy 4.1 provides control files to faciliate the component-building process. To recompile the entire system, use the GALAXY.CTL file. This submits control files for all the other components. Specific control files can be used to recompile individual Galaxy components.

DEBUGGING

Crash Recovery

Crash recovery under Galaxy 4.1 is much more complex than it was with previous Galaxy releases. There are a number of reasons why one or more Galaxy components may crash:

1. A monitor call (UUO or JSYS) which should not fail takes the error return, and a stopcode occurs.

2. An error is detected by the built-in consistency checks within the various components, causing a stopcode.

3. An undetected error eventually causes an illegal memory reference or an address check, and a stopcode occurs.

4. A hung device condition causes the monitor to ^C the program or, if detached, to put it into terminal output (TO) state.

5. A component goes into event wait (EW) state for an extended length of time while waiting for an event that is not likely to occur. In this case, the program does not crash, but instead hangs in the UUO or JSYS.

The following list shows the recovery procedures to follow if one of the Galaxy components crashes.

COMPONENT	CRASH RECOVERY PROCEDURES
LPTSPL	If LPTSPL crashes, it automatically restarts the jobs that were being processed at the time of the crash, from the page number recorded in the last checkpoint. The recovery procedure depends on which of the problems listed above caused the crash: Reasons 1, 2, or 3 - Attach to the crashed job, and save the crash in DSK:[3,3] as LPTxxx, where xxx is the 3-character stopcode name. Restart LPTSPL and detach it. Reason 4 - First determine which device caused the monitor to terminate the program. This can usually be found by checking on the CTY for a ?HUNG DEVICE XXXX message. Next, run OPR and shut the device down. A hung device error usually indicates a hardware problem. Finally, restart LPTSPL and detach it. Reason 5 - Check to make sure that ACTDAE is running. If not, perform the ACTDAE crash recovery (described below) before continuing; attach to LPTSPL, ^C it, restart it, and detach it.
CDRIVE	If CDRIVE crashes, all jobs currently being read from the card reader are lost and will have to be restarted. Process CDRIVE crashes the same as LPTSPL.
SPRINT	If SPRINT crashes, it can be restarted without loss of data. Process SPRINT crashes the same as LPTSPL.
SPROUT	If SPROUT crashes, all jobs currently being processed are restarted from the beginning of the current file. Process SPROUT crashes the same as LPTSPL.

ORION — If ORION crashes, restart it in the same job slot. All OPR programs running at the time automatically reset the OPR/ORION link with the first command typed to the OPRs. The first command that reestablishes the OPR/ORION link is lost. Any DN6Ø termination remote stations should be shut down and restarted. Process ORION crashes the same as LPTSPL (Reasons 1, 2, or 3).

OPR — If OPR crashes, just restart it. Process OPR crashes the same as LPTSPL (Reasons 1, 2, or 3).

BATCON — If BATCON crashes, first display the batch stream status from OPR. For all active batch streams, attach to the job running in that stream and log it out. Logging the jobs out does not delete the queue entry, so the job will be restarted when BATCON is restarted. When all active batch jobs have been logged out, attach to BATCON and process the crash the same as LPTSPL.

ACTDAE — If ACTDAE crashes, it can be restarted. However, the restart may leave some users/jobs (LPTSPL/SPROUT/SPRINT/QUEUE) hung in event wait state. Messages queued up for ACTDAE are lost on the restart. Users are not able to log in or log out while ACTDAE is down. Restarting ACTDAE may imply restarting other programs in order to restore full operational capability. Process ACTDAE crashes the same as LPTSPL.

PULSAR — PULSAR handles all tape positioning requests and structure mounts/dismounts. Users accessing tapes controlled by MDA may find themselves in event wait state waiting for the label processor to get them going. Unfortunately, since PULSAR was restarted, it does not know anything about the label process request, so the user/job will wait forever. Users/jobs in this state must log out and log back in; there is no easy way to clear this event wait condition. Users requesting structure mounts/dismounts will have to ^C the mount command, deallocate the structure, and remount it if appropriate. Process PULSAR crashes the same as LPTSPL (Reasons 1, 2, or 3).

QUASAR A QUASAR crash poses a major problem, both in its effect on the user community and in its recovery procedures. User requests for queue services are lost, although the failsoft master queue file is rebuilt when QUASAR is restarted. If MDA was running, user requests for tape/disk mounts/dismounts are not serviced. Programs using the QUEUE. UUO to interface with QUASAR hang in event wait state in the UUO. Tape drives allocated to users are set unavailable when QUASAR is restarted. However, users can continue to access the drives during the crash and after the restart without problems.

To restart QUASAR, first determine what batch jobs were running at the time of the crash, attach to them, and log them out. Next, restart QUASAR in the same job slot. Now attach to LPTSPL, BATCON, CDRIVE, SPRINT, SPROUT, and IBMSPL; ^C them, and restart them. From an OPR terminal, TAKE the SYSTEM.CMD file and any other command files used on system startup. From here on, process QUASAR crashes the same as LPTSPL (Reasons 1, 2, or 3).

Debugging

Galaxy 4.1 is much easier to modify/debug than the previous Galaxy releases. You no longer need a stand-alone system to debug the Galaxy system. Under Release 4.1, you can run a private Galaxy system under any PPN (one system per PPN). Talk to your private system just as you would talk to the real system. The Galaxy library handles the debugging system and named PIDs handle communication between processes and the user.

One way to run your private Galaxy system is to use OPSER to control all the system components. To convert a system Galaxy component to a debugging component, just poke location 135 in the job data area and make it non-zero. When the program starts, it initializes itself in debugging mode and will look for other debugging components with which to communicate. If you have MDA enabled on your 'real' system, and you want to debug/modify MDA-related functions, you must bring up your private Galaxy system under PPN [1,2]. A private system under [1,2] will not interfere with the real system which is also running under [1,2].

The only restriction in debugging mode is that the debugging Galaxy system cannot receive monitor messages; the monitor only communicates with the real Galaxy system. The minimum set of Galaxy components needed for a debugging system is OPR, ORION, and QUASAR. To debug MDA, you also need PULSAR. IPCF privileges are needed in order to run a private Galaxy system; without them, the following error message is generated when you try to run a Galaxy component:

```
.GET SYS:QUASAR
 Job setup

.E 135
135/ ØØØØØØ ØØØØØØ
.D 1 1
.E 135
135/ ØØØØØ1 ØØØØØ1
.ST

%QUASAR GLXIPC        Becoming [77,4655]QUASAR     (PID=43Ø16)
?QUASAR GLXIPC IPCF privileges required to get IPCF quotas
```

•

Crash Analysis

Whenever a GALAXY component stopcodes, it saves the contents of the ACs and the text description of the crash. In the Galaxy code, the $STOP macro is used to process the fatal condition. Internal to Galaxy, the stopcode name is stored as S.xxx, where xxx is the 3-letter stopcode name. Stopcodes stop program execution, and execution cannot continue unless DDT has been loaded with the program. If DDT is loaded with the program, the stopcode processor transfers control to DDT, and the AC and crash block information (shown below) is suppressed. Here is a sample crash:

```
     --Program QUASAR terminated--
   Job 1 [1,2] OPR at terminal 22
?Stop code - APT - in module GLXINT on 19-Jun-8Ø at 12:Ø1:32
  Reason: Illegal memory reference at PC 4Ø7245
  Program is QUASAR Version 4(672) using GLXLIB Version 1(6Ø7)

Contents of the ACs  (Crash block starts at location 6ØØØ72)
  Ø/        2ØØØØ            65               Ø             Ø
  4/       377136             6    2Ø2354572357  67552ØØ5253Ø
 1Ø/            Ø  4463534200ØØ               Ø  67552ØØ5253Ø
 14/       1Ø7Ø74          4Ø67          6ØØ23Ø  7775150Ø1624

Last 9 Stack Locations:
-1(P)/  3Ø4ØØØ4Ø1215  -2(P)/         52526  -3(P)/  31ØØØØ4Ø6226
-4(P)/  31ØØØØ4Ø6126  -5(P)/  31ØØØØ4Ø7135  -6(P)/        41Ø5Ø7
-7(P)/             Ø  -8(P)/             Ø  -9(P)/             Ø
```

Stopcodes are conditionally sent to ORION to be recorded in the system log, and to any jobs running OPR. A flag in the program's initialization block enables this function.

The ACs are saved in an area called .SACS, from locations .SACS+Ø to .SACS+17. The address of the ASCIZ text describing the crash is saved in WTOADR. Should you wish to continue a program that has stopcoded, first determine where the stopcode occurred. The address where the stopcode was executed is located at S..xxx, where xxx is the 3-character stopcode name. Next POP P,<alt>X to phase the stack to its pre-crash value. You can now continue the program from the point following the stopcode, provided that you have corrected the situation which caused the program to stopcode in the first place. Other data areas which may help resolve a crash are:

.SPC	PC of the stopcode
.SCODE	Sixbit name of the stopcode
.SPTBL	Page table address

```
.SPRGM   Sixbit program name
.SPVER   Program version number
.SPLIB   Glxlib version number
.LGERR   Last Galaxy error code
.LGEPC   PC of the last $RETE
STPFLG   A stopcode is in progress (-1)
MAXPAK   Maximum IPCF packet length
PIDTAB   System PID table; indexed by SP.xxx,
         Where xxx is the processor name (QSR, MDA, OPR etc)
RCVBLK   PDB of the last IPCF message received
PACKET   The contents of the last IPCF msg received as a
         packet
PAGTBL   Memory page map (one word per page)
         Bits PG.USE==1B35 - Page is in use
              PG.WRK==1B34 - Page is in working set
              PG.ADR==1B33 - Page is addressable
              PG.INI==1B32 - Page is part of initial core
                             image
              PG.SWP==1B31 - Page is swappable on a timer
                             trap

         If no bits are lit, it indicates that the page
         does not exist.
```

Chapter 8
USAGE ACCOUNTING

WARNING

The information about Galaxy 4.1 and its related components (Usage Accounting, MDA, etc.) is subject to change. The release of Galaxy 4.1 has been decoupled from 7.01 and will not occur until February 1981. Any information contained in this document about those features is preliminary and may change before the release.

Usage accounting is a system that records the use of system resources. When a person uses a system resource, a record is made so that he/she may eventually be charged. Information that is collected includes CPU time, connect time, spool requests, disk space used, etc.

This feature is not totally new to TOPS-10, having been previously implemented via DAEMON. But several problems have always existed. One is that the entries were stored in a compressed binary format, which required a program to convert the entries into ASCII for processing. DIGITAL never supported such a program, which forced customers to write their own conversion program. DIGITAL also did not supply a report program. Another problem was that entries were made directly into the file by many components. Modifications to the entries required modifications to the monitor and the batch/queueing system. The existing system also lumped all usage charges together by PPN. The user or administrator could not organize charges within PPNs or across PPNs.

These problems have been solved with the addition of a new program, ACTDAE, and the reorganization of usage accounting to include account numbers. All resource usage entries from the various components are sent to ACTDAE via IPCF messages, which are then stored in ASCII by ACTDAE. Modifications to existing entries can be made by modifying ACTDAE instead of the individual components. By adding accounts, charges can be divided as the system administrator sees fit.

ACCOUNTS

An account is a string of up to 39 ASCII characters that can be used for billing by a report program. All entries into the usage file will have both a PPN and an account. A PPN may be part of an account or may have several accounts under it. Refer to Figure 8-1.

The programs that produce reports are then able to summarize by accounts, not just PPNs.

The system administrator can force a user to type an account by setting the account bit (bit 25) in the profile word of the user's entry in ACCT.SYS (via REACT). When the user logs-in, two new prompts are given:

```
.LOGIN
RZØ16A KL #1Ø26/1Ø42
# 77,4655
PASSWORD:
ACCOUNT: COURSE DEVELOPMENT
REMARK: MONITOR INTERNALS WORK
```

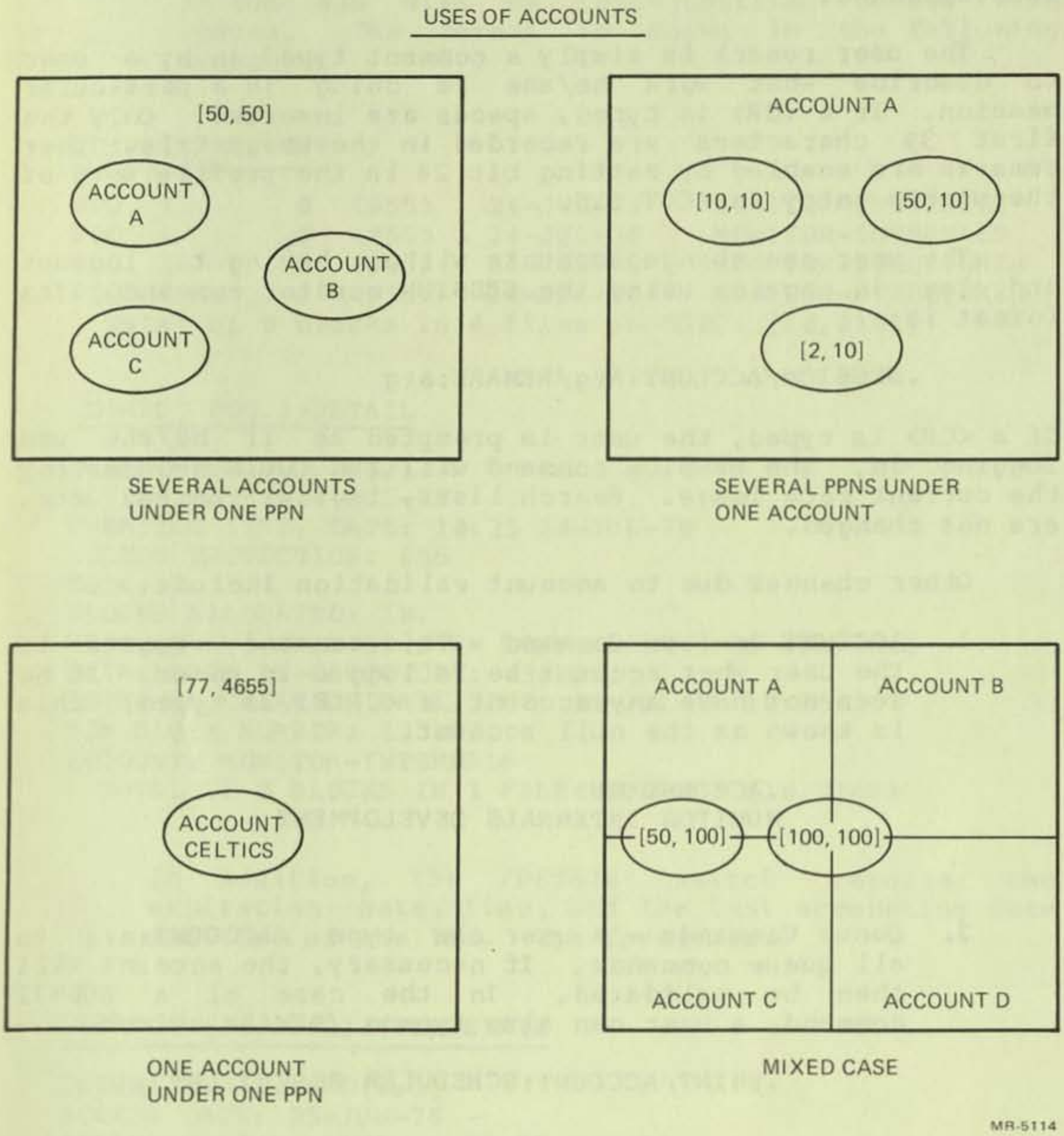


Figure 8-1 Uses of Accounts

The account is checked against a file called PROJCT.SYS and validated.

The user remark is simply a comment typed in by a user to describe what work he/she is doing in a particular session. If a <CR> is typed, spaces are inserted. Only the first 39 characters are recorded in the usage file. User remarks are enabled by setting bit 24 in the profile word of the user's entry in ACCT.SYS.

The user can change accounts without having to log-out and log in again using the SESSION monitor command. Its format is:

```
.SESSION/ACCOUNT:Arg/REMARK:arg
```

If a <CR> is typed, the user is prompted as if he/she was logging in. The SESSION command will run LOGIN and destroy the current core image. Search lists, logical names, etc. are not changed.

Other changes due to account validation include:

1. ACCOUNT Monitor Command - This command reports to the user what account he is logged-in under. If he does not have any account, a <CRLF> is typed; this is known as the null account.

   ```
   .ACCOUNT<CR>
   MONITOR INTERNALS DEVELOPMENT
   ```

2. Queue Commands - A user can type /ACCOUNT:arg to all queue commands. If necessary, the account will then be validated. In the case of a SUBMIT command, a user can also type a /REMARK switch.

   ```
   .PRINT/ACCOUNT:SCHEDULER SCHED1.LST
   ```

3. DIRECT Commands - There is a new switch in DIRECT, the /ACCOUNT switch, which types-out the file and the account associated with that file. Whenever a file is created or superseded, the account is stored in the RIB with the account the user is

logged in under. In all cases, the account stored in the RIB will be left-justified padded with spaces. The format is shown in the following examples.

```
.DIRECT/ACCOUNT

FOO  1       Ø  <Ø55>   24-JUL-78     MONITOR-INTERNALS
FOO  2       Ø  <Ø55>   24-JUL-78     MONITOR-INTERNALS
FOO  3       Ø  <Ø55>   24-JUL-78     MONITOR-STRUCTURES
FOO  4       Ø  <Ø55>   24-JUL-78     MONITOR-STRUCTURES
  Total of Ø blocks in 4 files on DSKC: [1Ø,2162]
```

```
.DIRECT FOO.1/DETAIL

DSKCØ:FOO.1[1Ø,2162]
ACCESS DATE: 24-JUL-78
CREATION TIME, DATE: 1Ø:35 24-JUL-78
ACCESS PROTECTION: Ø55
MODE: Ø
BLOCKS ALLOCATED: 1Ø.
WRITTEN ON: UNIT(S) Ø ON CONTROLLER 1 ON CPU 1Ø26
DATA BLOCK IN DIRECTORY: 43915Ø.
INTERNAL CREATION DATE,TIME: 24-JUL-78 1Ø:35:3Ø
RIB BLOCK NUMBER: 13303Ø.
ACCOUNT: MONITOR-INTERNALS
  TOTAL OF Ø BLOCKS IN 1 FILE ON DSKC: [1Ø,2162]
```

In addition, the /DETAIL switch reports the expiration date, time, and the last accounting date and time of the UFD. The format is:

```
.DIRECT [16,2162].UFD/DETAIL

DSKBØ:16,2162.UFD[1,1]
ACCESS DATE: 25-JUN-76
CREATION TIME, DATE: 5:16 25-JUN-76
ACCESS PROTECTION: 775
MODE: Ø
WORDS WRITTEN: 128.
BLOCKS ALLOCATED: 5.
WRITTEN ON: UNIT(S) 4 ON CONTROLLER Ø ON CPU 514
```

```
LOGGED IN QUOTA: 3ØØØ.
LOGGED OUT QUOTA: 25ØØ.
BLOCKS USED: 585.
DATA BLOCK IN DIRECTORY: 1125.
INTERNAL CREATION DATE,TIME: 11-JUN-76  2:54:2Ø
LAST ACCOUNTING DATE,TIME: 3Ø-JUN-76  18:45:29
DIRECTORY EXPIRATION DATE,TIME: 11-JAN-96  Ø:3Ø:ØØ
RIB BLOCK NUMBER: 69765.

  TOTAL OF 1 BLOCK IN 1 FILE ON DSKB: [1,1]
```

PROJCT.SYS File

PROJCT.SYS contains the list of valid accounts and the PPNs to which they apply. It is created from an ASCII source file called PROJCT.ACT by using any text editor. After PROJCT.ACT is created, the program, PROJCT.EXE, must be run to convert the ASCII information into a mixed mode file called PROJCT.SYS. This file is designed so that programs reading the file execute faster than if they read PROJCT.ACT. PROJCT.SYS resides on the ERSATZ device ACT: (project-programmer number [1,71]).

The format of PROJCT.ACT entries will be:

```
[P,PN] = Account
```

Examples:

```
[1Ø,7] = MONITOR-DEVELOPMENT
[77,4655] = COURSE DEVELOPMENT
```

Wildcarding is supported using the characters ? and * with the following definitions and restrictions:

1. * matches any arbitrary string, including the null string
2. ? does not match a null; therefore, for every question mark, there must be a character.
3. Restriction: this wildcarding does not support *'s in combination with numbers. For example, [1,2*] is illegal.

The project-programmer entries in PROJCT.ACT must be in ascending numerical order. In the case of wildcarding, a '?' becomes a logical 7 and A '*' becomes a logical 777777. In ascending numerical order, a logical 7 occurs after a real 7. The following is an example of hierarchy in PROJCT.ACT.

```
[1Ø,1Ø]=ABC
[1Ø,237Ø]=DEF
[1Ø,*]=GHI
[*,*]=JKL
```

If the project programmer entries are not in ascending numerical order, the following error message is typed when the program PROJCT.EXE is run:

```
SPECIFIED [P,PN] IS NOT IN ASCENDING [P,PN] SEQUENCE
```

Accounts can also be wildcarded. In the following examples:

```
[1Ø,2162]=???ABC*
```

All accounts typed with "ABC" beginning at character position four and ending at character position six will be valid for [1Ø,2162]. Note that if A "*" is used, the scan and compare of the ASCIZ account strings stops. This implies that any arbitrary number of characters (maximum is 39 total) typed after the "ABC" in the above example is considered valid.

To run PROJCT.EXE, have PROJCT.ACT in your area, then type:

```
.R PROJCT<CR>
```

If there are no errors, PROJCT.SYS is created and the program exits with:

```
END OF JOB
EXIT
```

ACTDAE

The ACTDAE (Accounting DAEMON) program is the clearinghouse for all resource usage entries and maintains the output usage files. It also performs account validation. ACTDAE is run as a detached operator [1,2] job, started when the system is brought up.

Data Flow

The data collected in the USAGE file (ACT:USAGE.OUT) is a record of computer resource usage. The system starts to collect this data on a user's system resource usage when that user logs in, and collects data until that user logs-out. The system collects this new data in two ways: by checkpointing and by an event-driven process.

CHECKPOINTING

Checkpointing is the periodic collection of data by the system of jobs measured in units of time. For example, checkpointing collects the total runtime for the user's job and the user's console-connect time. Checkpointing does not collect data on system resources that are measured by the event, such as a tape mount.

When a user logs in, ACTDAE stores accounting data for that user in a checkpointing file. The data stored in this file on disk includes the user's console connect time, run time, and the time and date the current session started. This accounting data becomes the USAGE file session entry.

ACTDAE updates all the user job slots in the checkpoint file at the end of each checkpoint interval. Checkpointing allows the system to keep track of resource use even if the system crashes. After a crash, the checkpoint file on the disk contains accounting data current at the last checkpoint before the crash. When the operator restarts the system, the system writes the data in the checkpoint file as incomplete session entries in the USAGE file. No job that causes accounting entries proceeds until the system writes incomplete session entries for each job that was running when the crash occurred.

ACTDAE maintains the file ACT:USEJOB.BIN on the disk as an open file. When a user logs in, ACTDAE opens a job slot in USEJOB.BIN for that user. This job slot contains the data that will become a USAGE file session entry.

ACTDAE also maintains the file ACT:USEDEV.BIN on the disk. Each time a user mounts a device, ACTDAE opens a device slot in this file for the user. This job slot contains checkpoint data concerning the use of devices and file structures by the user.

When a checkpoint interval ends, ACTDAE updates the USEJOB.BIN and USEDEV.BIN job slots for each active job, mounted device, and file structure. ACTDAE updates the job slots in USEJOB.BIN by updating the fields that contain the console connect times, the run times of the users, and other fields that make up the data that form the session entry. ACTDAE updates USEDEV.BIN device slots by updating the fields that contain the number of disk reads and writes, the device connect time for each user, and other fields that make up the data that form the USAGE file device entries.

The default length of the checkpoint interval is ten minutes. For most systems this interval ensures accurate accounting. The length of the checkpoint interval can be changed by reassembling ACTDAE with new parameters.

When a logged in user types the SESSION command, ACTDAE updates that user's USEJOB.BIN job slot and that user's USEDEV.BIN device slot. ACTDAE then writes a session entry, appropriate device, and file structure entries in the USAGE file. Then ACTDAE copies the contents of that USEJOB.BIN job slot into a new job slot in the auxiliary checkpoint file ACT:USEJOA.BIN.

ACTDAE also copies the device slot from USEDEV.BIN, the primary device checkpoint file, to the auxiliary checkpoint file ACT:USEDEA.BIN. The auxiliary checkpoint file now contains the checkpoint data current at the beginning of the second session.

The user's second session now starts, and the user can again use system resources. ACTDAE continues to update the job slots in USEJOB.BIN and USEDEV.BIN at the end of every checkpoint interval. ACTDAE does not update the job slots in USEJOA.BIN or USEDEA.BIN at the end of each checkpoint interval. Instead, USEJOA.BIN and USEDEA.BIN contain the checkpoint data current at the time the user gave the SESSION command.

If the user types a second SESSION command, the user's second session ends. ACTDAE obtains the accounting data for this session by subtracting the data in USEJOA.BIN from the data in USEJOB.BIN. For example, to find the connect time for the second session, ACTDAE obtains the difference between the connect time recorded in USEJOB.BIN and the accumulated connect time in USEJOA.BIN.

ACTDAE uses this data to write a session entry in the file ACT:USAGE.OUT. Then ACTDAE copies the contents of USEJOB.BIN into USEJOA.BIN. ACTDAE checkpoints the job slot in USEJOB.BIN as usual at the end of each succeeding checkpoint period.

ACTDAE obtains the data needed for the file structure and device use entries by the same process as for session entries. ACTDAE obtains the difference between the data in USEDEV.BIN (the current data) and the data in USEDEA.BIN (the data that was current at the beginning of the session). The difference between these two sets of data reflects the information on the data on device and file structure use for the second session. Refer to Figure 8-2.

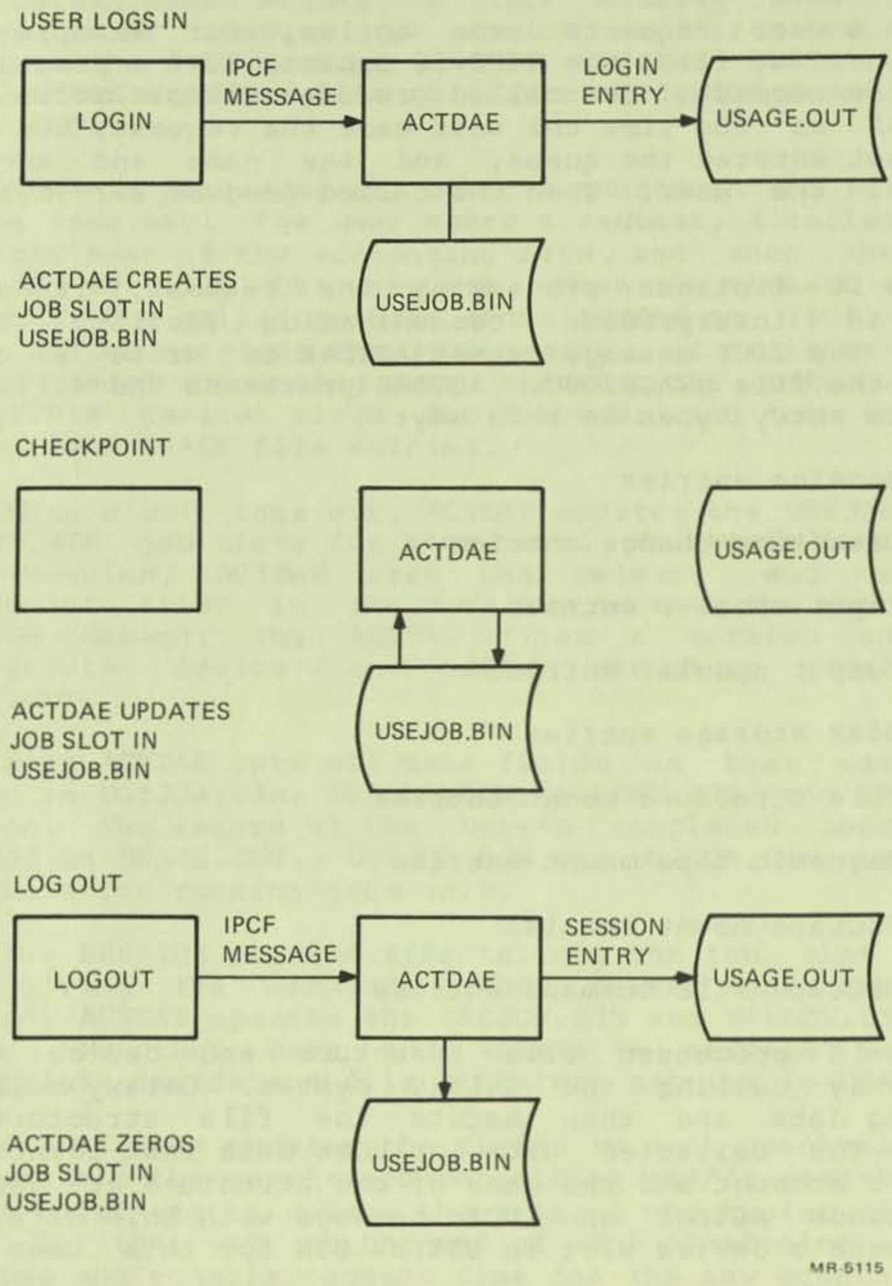


Figure 8-2 Collecting Session Data

EVENT-DRIVEN DATA COLLECTION

When a user requests some action, for example the printing of a file, the TOPS-10 monitor runs a program to service the request. The called program collects accounting data such as the time the user made the request, the time the request entered the queue, and the name and account number of the user. Then the called program services the request.

When it finishes processing the request, it sends ACTDAE an interprocess communication facility (IPCF) message. The IPCF message causes ACTDAE to write a USAGE entry in the file USAGE.OUT. ACTDAE processes the following USAGE file entry types in this way:

1. Session entries

2. Date/time change entries

3. Input spooler entries

4. Output spooler entries

5. Disk storage entries

6. File structure mount entries

7. Magnetic tape mount entries

8. DECtape mount entries

9. DECtape FILE command entries

TOPS-10 processes file structure and device mount requests by calling the Galaxy system. Galaxy collects accounting data and then mounts the file structure or device. The collected data includes data that identifies the user's account and the name of the structure or device. Galaxy sends ACTDAE an IPCF message with this data, and ACTDAE opens a device slot in USEDEV.BIN for this user and device.

Galaxy sends ACTDAE an IPCF message when the user dismounts or removes the device or structure. When it receives the IPCF message, ACTDAE updates USEDEV.BIN, and then ACTDAE writes a file structure or device mount entry in USAGE.OUT.

TOPS-10 processes all other requests in the list above in the same way: the user makes a request, a called program collects some of the accounting data, and then the called program services the request. When the request servicing finishes, the called program sends ACTDAE an IPCF message. When ACTDAE receives the IPCF message, it collects the rest of the needed accounting data, updates the USEJOB.BIN and USEDEV.BIN device slots for the user, and then writes the appropriate USAGE file entries.

When a user logs out, ACTDAE updates the USEJOB.BIN and USEDEV.BIN job slots for that user. To obtain the data for that session, ACTDAE uses the primary and auxiliary checkpoint files in the same way as when the user types a SESSION command. Then ACTDAE writes a session entry and appropriate device and file structure entries into USAGE.OUT.

Next ACTDAE sets all data fields in that user's job slots in USEJOB.BIN, USEJOA.BIN, USEDEV.BIN, and USEDEA.BIN to zero. The record of the user's completed session now resides in USAGE.OUT. USEJOB.BIN and USEJOA.BIN now contain job slots for running jobs only.

The SESSION command affects only the job slot of the user giving the command. When a user types the SESSION command, ACTDAE updates the USEJOB.BIN and USEDEV.BIN device slots for that user and writes a session entry and appropriate device and file structure entries in USAGE.OUT.

ACTDAE then updates the fields in each checkpoint file that contain the user's account string and/or remark string. ACTDAE continues to update the primary checkpoint file job slots for that user at the end of each checkpoint interval. Run time and console connect time for the new account accrue from the time the user gave the SESSION command.

SYSTEM CRASHES

If a system crash occurs, TOPS-10 has the most recent checkpoint data in the disk files USEJOB.BIN and USEDEV.BIN, the primary checkpoint files. These files contain data that ordinarily become USAGE file session entries and device and file structure entries.

Whenever the operator restarts the system, ACTDAE writes a system restart entry in the file USAGE.OUT. If the operator restarts the system after a crash, ACTDAE also uses the checkpoint data in the USEJOB.BIN file to write an incomplete session entry for each job that was logged in when the system crashed. The monitor will not process any job that uses accountable system resources until it writes these incomplete session entries in USAGE.OUT.

Entry Descriptions

All data is ASCII with each record terminated by a carriage return-line feed. Entry types from 0000 through 5000 are DEC-reserved and from 5001 through 9999 are customer-reserved.

Each entry of the file has two logical parts: a header record followed by one or more data records. The header record of each entry has the same format for all entries. The subsequent data records vary among entry types. The entries defined are:

1. System restart entries
2. Session entries
3. Incomplete session entries
4. Usage file header entries
5. Date/time change entries
6. Batch processor entries
7. Input spooler entries

8. Output spooler entries

9. Disk usage entries

1Ø. Spindle usage entries

11. File structure mount entries

12. Magtape mount entries

13. DECtape mount entries

14. DECtape file command entries

NOTE

For complete descriptions of all entries, refer to the TOPS-1Ø/TOPS-2Ø Usage File Specification Manual (AA-4181B-TK).

1. System Restart Entry (Entry Type ØØØ1) - A system restart entry contains an entry header record and a system restart record. This entry is written for every system restart. When a system is restarted, this entry is the first to be written. Afterward, all the incomplete session entries are written from the checkpoint files.

2. Session Entry (Entry Type ØØØ2) - A session entry is written whenever a user logs out a job or types a successful session command. This entry consists of an entry header record, session record number one, session record number two, and a TOPS-1Ø user identification record.

3. Incomplete Session Entry (Entry Type ØØØ3) - Incomplete session entries are written only if the monitor has stopped and is being restarted again. These entries have the same format as session entries, except the entry type is ØØØ3 instead of ØØØ2. On a monitor restart, a system restart entry

is made, then the checkpoint file is scanned and any job slots that have data in them are appended to the usage file.

4. Usage File Header Entry (Entry Type 0004) - The usage file header entry is always at the beginning of every usage file. It gives system information where the file was made. This entry consists of an entry header record followed by a usage file header entry record.

5. Date/Time Change Entry (Entry Type 0005) - The date/time change entry is written every time the system's date and time is changed by executing the set date or set daytime commands. These entries consist of an entry header record and a date/time change record. The old date and time are recorded in the date/time change record. The new date and time are found in the date/time field in the entry header record.

6. Batch Processor Entry (Entry Type 0006) - The batch processor entry consists of an entry header record, a batch processor record, and a user identification record. The date/time recorded in the entry header record is the time the batch job was completed. Note that a session entry was also made when the job running under batch was logged out.

7. Input Spooler Entry (Entry Type 0007) - This entry contains an entry header record, an input spooler record, and the user identification record.

8. Output Spooler Entry (Entry Type 0008) - The output spooler entry consists of an entry header record, output spooler record, and a user identification record. The scheduled date/time of the request compared with the date/time in the header record is the time an output device was busy and unavailable for other users.

9. Disk Usage Entry (Entry Type 0009) - This entry enables an installation to keep track of permanent disk storage. One disk usage entry is made for each non-empty UFD. An entry contains an entry record header, a disk usage directory record, and

disk usage account string records. There is one disk usage account string record for every account string occurring in the project-programmer number specified in the disk usage directory record. A count of the number of account string records is included in the disk usage directory record.

1Ø. Device Mount Usage Entries - The device mount usage entry records applicable computer usage, and connect time, whenever a user is using a mountable peripheral device, such as magtape drives, DECtape drives, and disk drives. The first two are stand-alone entries. The latter is a stand-alone entry in a sense, but is extended by the disk spindle usage entries.

11. Disk Spindle Usage Entry (Entry Type ØØ1Ø) - The disk spindle usage entry gathers data on disk drive usage. This entry consists of an entry header record, and one disk spindle usage record for each physical disk drive in the file structure. The disk spindle usage record tells which disk drive in the file structure that record describes. The entry header record contains the date and time the disk drive is dismounted. The disk spindle usage record contains the date and time the disk drive is first mounted.

12. File Structure Mount Entry (Entry Type ØØ11) - The file structure mount entry provides data whenever a user mounts a file structure. It is then appended to the usage file when the user dismounts or logs out. This entry contains an entry header record, a file structure mount record, and a user identification record.

13. Magtape Mount Request Entry (Entry Type ØØ12) - This entry consists of an entry header record, a magtape mount request record, and a user identification record. There is one entry for every magtape mounted.

14. DECtape Mount Request Entry (Entry Type ØØ13) - This entry is created when a user mounts a DECtape. File commands are recorded in the DECtape file command entry. A DECtape mount request entry

contains an entry record header, a DECtape mount request record, and a user identification record. This entry is appended to the usage file when the user dismounts the tape or logs out.

15. DECtape File Command Entry (Entry Type ØØ14) - This entry is appended to the usage file for every file command. An entry contains an entry header record, a DECtape file command record, and a user identification record.

EXAMPLES OF USAGE FILES

This section will give examples of two usage files. Dashes denote record boundaries; equal signs denote entry boundaries.

1. Regular Usage File -- This is a typical file with no system restarts occurring after the file has been created.

Entry Header Record	Usage Header Entry
Usage Header Record	" " "
Entry Header Record	Session Entry
Session Record #1	" "
Session Record #2	Session Entry
User Identification Record TOPS-1Ø	" "
Entry Header Record	Session Entry
Session Record #1	" "
Session Record #2	" "
User Identification Record TOPS-1Ø	" "
Entry Header Record	Session Entry
Session Record #1	" "

Session Record #2	Session Entry
User Identification Record TOPS-1Ø	" "
Entry Header Record	Output Spooler Entry
Output Spooler Record	" " "
User Identification Record TOPS-1Ø	" " "

2. Usage File With A Restart - The incomplete session entries in this example will have the same format as session entries.

Entry Header Record	Usage Header Entry
Usage Header Record	" " "
Entry Header Record	Session Entry
Session Record #1	" "
Session Record #2	" "
User Identification Record TOPS-1Ø	" "
Entry Header Record	Session Entry
Session Record #1	" "
Session Record #2	" "
User Identification Record TOPS-1Ø	" "
Entry Header Record	Session Entry
Session Record #1	" "
Session Record #2	" "
User Identification Record TOPS-1Ø	" "
Entry Header Record	Output Spooler Entry

Output Spooler Record	Output Spooler Entry
User Identification Record TOPS-1Ø	" " "
* * * * * * C R A S H * * * * * * * *	
Entry Header Record	System Restart Entry
System Restart Record	" " "
Entry Header Record	Incomplete Session Entry
Session Record #1	" " "
Session Record #2	" " "
User Identification Record TOPS-1Ø	" " "
Entry Header Record	Incomplete Session Entry
Session Record #1	" " "
Session Record #2	" " "
User Identification Record TOPS-1Ø	" " "
Entry Header Record	Session Entry
Session Record #1	" "
Session Record #2	" "
User Identification Record TOPS-1Ø	" "

PART FIVE

MISCELLANEOUS CHANGES

Chapter 9

CHANGES TO MONITOR CALLS

UUO ADDRESS CHECKING

Before 7.Ø1, TOPS-1Ø spent a significant percentage of UUO-processing time to perform range-checking of user arguments. To improve performance, addresses are now referenced without range-checking. If the address is out of range, an IME stopcode results, but TOPS-1Ø has been instructed to trap the error. This is implemented by following any instruction that could fail with an "ERJMP addr". Control passes to addr in case of an error. For example:

```
    EXCTUX  <MOVE T1,CMJ>
      ERJMP ERR3A           ;USER GAVE ILLEGAL ADDRESS
```

This can be used only at UUO or scheduler level.

NEW MONITOR CALLS

MERGE. Monitor Call (CALLI 173)

The MERGE. monitor call merges the contents of an .EXE file into the current program's core image and is used to load page fault handlers or VMDDT. This function is also implemented as the MERGE command at monitor level. During the loading, page zero of the .EXE file is discarded.

The sixth word of the argument block allows the .EXE file to be merged into a specified range of pages. With this feature, a high segment can be merged into the low segment to avoid an overlap problem with an existing high segment.

It is called in the following manner:

```
        MOVEI   ac,addr
        MERGE.  ac,
          error return
        normal return

addr:   SIXBIT  /device/
        SIXBIT  /filename/
        SIXBIT  /extension/
        EXP     Ø
        XWD     proj ,prog     ;or PATH. pointer
        XWD     starting page, ending page
```

where: addr is the address of the 6-word argument block.

device is the name of the device where the .EXE file resides.

filename specifies the name of the file containing the new high-segment.

extension specifies the extension of the file containing the new high segment.

proj ,prog is the project-programmer number representing the directory where the high segment file resides. If this word is zero, the user's project-programmer number is assumed. If this word is [XWD Ø,,addr], addr points to a PATH. block containing the full path specification for the file. Refer to the description of the PATH. monitor call in the *TOPS-1Ø Monitor Calls Manual*, AA-D974D-TB, for information pertaining to the format of the PATH. pointer.

starting page is the page number where loading should begin, if offsets are desired.

ending page is the page number where the loading should end, if offsets are desired.

The error return is taken if any errors are detected, with the monitor returning an error code in the AC. The error codes are the same as those returned by RUN, GETSEG and FILOP. A program can attempt to recover from an error and continue the program's execution. If you set the left half of the error return location to a HALT, the monitor does not return to the program but prints an error message and halts.

UTRP. Monitor Call (CALLI 174)

This monitor call allows a user to handle arithmetic overflow or pushdown list overflow by depositing the address of a user trap routine into UPMP locations 42Ø or 421. At the present time, only these two conditions can be handled. Upon either error, control is transferred directly to the user routine (as opposed to indirectly, as was done in 6.Ø3). UTRP. can set/read either or both locations. The standard call is:

```
        MOVE ac,[XWD fcncode,arglst]
        UTRP. ac,
          error return
        normal return

arglst: number of pairs
        trapno
        trapaddr
        trapno
        trapaddr
```

may be wrong -need instr. here?

where: fcncode is one of these function codes.

Ø - Read trap address
1 - Set trap address

arglst is the address of the argument list.

pairs is the number of two-word pairs in the argument list.

trapno is the number of a trap.

1 - Arithmetic overflow
2 - Pushdown list overflow

trapaddr is the address of the trap routine.

If there is an error, one of the following error codes is returned in the accumulator:

Code	Symbol	Error
1	UTIAD%	Illegal address
2	UTUKF%	Unknown function
3	UTITN%	Illegal trap number

PIJBI. Monitor Call (CALLI 175)

The PIJBI. monitor call is part of the programmed software interrupt service that allows one job to interrupt another (software event JBI). The receiver enables cross job interrupts by using the PISYS. monitor call.

The call is in the following form:

```
MOVE    ac,[XWD job-to-interrupt, status-to-give]
PIJBI.  ac,
  Error Return
Normal Return
```

A negative one (777777) in the left half of the accumulator will interrupt the job executing the monitor call. The target job must be enabled for cross job interrupts or else the call will fail. Note that this is another situation where cooperation is vital between two jobs, much like Enqueue/Dequeue or IPCF. If the target job already has one interrupt, the sender must try again, because requests are not stored. There is only one place to store the request.

The interrupted job receives a word of the following form:

Job number sending interrupt,, status

The exchanged status bits can contain whatever information the sending job desires; there is no pre-established meaning.

SNOOP. Monitor Call (CALLI 176)

The SNOOP. monitor call allows a privileged program to insert breakpoints that trap to a user program in the monitor. The program must be locked in contiguous Executive Virtual Memory. This feature is used for fault insertion, performance analysis and trace functions. Only one job can use SNOOP at one time. The maximum number of breakpoints that can be defined is given in item %CNBPM in Gettab Table .GTCNF (see Appendix C). The default number is 64 and can be set when running MONGEN by setting the decimal symbol MAXNBP equal to the desirable number of breakpoints.

The SNOOP. monitor call has four functions:

```
.SODBP (0) - Define breakpoints
.SOIBP (1) - Insert breakpoints
.SORBP (2) - Remove breakpoints
.SOUBP (3) - Undefine breakpoints
```

Breakpoints must be defined before they can be inserted. For each defined breakpoint (function 0), a breakpoint code block is created in user funny space. This code block holds the monitor instruction that is being replaced, the address of that instruction and the user breakpoint instruction:

AOS .+SNPCNT
User instruction
SOSA at error return of user instruction
SOSA at normal return of user instruction
Replaced monitor instruction
JRST at error return of monitor instruction
JRST at normal return of monitor instruction Current use count for this breakpoint

When the breakpoints are actually inserted, in function 1, the old monitor instruction is stored in the breakpoint code block and the user instruction (usually a JRST to the user routine) overwrites the monitor instruction. This is done with all interrupts disabled. The only breakpoints that will be inserted are those that were defined using function 0. If any error occurs when defining breakpoints, all previously defined breakpoints are automatically undefined.

When breakpoints are removed, the reverse procedure is used. A remove breakpoint instruction (function 2) moves the original monitor instruction back. The undefine instruction (function 3) then deletes the breakpoint code blocks. The remove instruction must be done before the undefine function. The RESET monitor call both removes and undefines all breakpoints set by the SNOOP. call.

User programs must be careful not to destroy any ACs or create any "?ILL MEM REF"s as this code is executed in Exec mode. A simple programming error may result in a hung monitor or a stopcode.

The call is made in the following manner:

```
MOVE    ac,[XWD fcncode,arglst]
SNOOP.  ac,
  error return
normal return
```

where: <u>fcncode</u> is one of the function codes described below.

<u>arglst</u> is the address of the argument list. The words at arglst depend on the given function.

The function codes and their meanings are:

Code	Symbol	Function
0	.SODBP	Define breakpoints. This function is illegal if breakpoints have been inserted.

Code	Symbol	Function
1	.SOIBP	Inserts defined breakpoints. Your program must be locked in contiguous Executive Virtual Memory to use this function (see the LOCK monitor call in the TOPS-1Ø Monitor Calls Manual, AA-D974D-TB).
2	.SORBP	Removes inserted breakpoints (but does not undefine them).
3	.SOUBP	Undefines defined breakpoints (which must first have been removed by .SORBP).

The argument list for the .SODBP function is

```
arglst: EXP    arglength

        EXP monitor symbol table checksum

        EXP monitor address
        instruction
        . . .
        EXP monitor address
        instruction
```

where: arglength is the length of the argument list, which must be 2 greater than the number of address-instruction pairs in the argument list.

monitor symbol table checksum is the checksum from the current monitor's symbol table. The monitor symbol table checksum is required in this argument list to ensure that the user is setting breakpoints in the intended monitor. You can obtain the monitor symbol table checksum by examining the MTSCKS word in the monitor.

monitor address is the address where the breakpoint is to be inserted.

instruction is an instruction to be inserted at the corresponding breakpoint.

One of the following error codes is returned in the AC:

Code	Symbol	Error
1	SOIAL%	Illegal argument list
2	SONPV%	Not enough privileges
3	SOSAS%	Another program already SNOOPing
4	SOMBX%	Maximum number of breakpoints exceeded
5	SOIBI%	Breakpoints already inserted
6	SONFS%	No monitor free core available
7	SOADC%	Address check
1Ø	SOINL%	Program not locked in contiguous Executive Virtual Memory
11	SOWMS%	Monitor symbol table checksum does not match

Here is a sample code showing how to use the SNOOP. monitor call.

```
                .
                .
                .
        MOVE    AC1,[XWD .SODBP,PUTEM]
        SNOOP.  AC1,
          JRST  NOGOOD
        MOVE    AC1,[XWD .SOIBP,Ø]
        SNOOP.  AC1,
          JRST  NOBTTR
                .
                .
PUTEM:  EXP     6
        EXP     MONITOR-CHECKSUM
        EXP     12345
        JRST    HOOK1
        EXP     12355
        JRST    HOOK2
```

At this point the breakpoints have been inserted. Note that user symbols are used in the inserted monitor instruction. This is possible because the user job is locked in Executive Virtual Memory. To remove them:

```
                .
                .
        MOVE    AC1,[XWD .SORBP,Ø]
        SNOOP.  AC1,
          JRST  BUMMER
        MOVE    AC1,[XWD .SOUBP,Ø]
        SNOOP.  AC1,
          JRST  LOSTIT
                .
                .
```

KDP. Monitor Call (CALLI 200)

A description of the KDP. monitor call can be found in the Chapter 11, Networks.

QUEUE. Monitor Call (CALLI 201)

A description of the QUEUE. monitor call can be found in the Chapter 4, Galaxy.

TSK. Monitor Call (CALLI 177)

A description of the TSK. monitor call can be found in the Chapter 11, Networks.

ADDITIONS TO EXISTING MONITOR CALLS

PAGE. Monitor Call (CALLI 145)

The PAGE. UUO has been modified so that if word zero of the argument block (the count word) is negative, it is treated as a range over which the next argument is to be applied. For example, if the function was to create pages and the argument block was:

```
ADDR:   -100
         200
```

then 100 pages are created starting at virtual page 200. This form is valid for any PAGE. UUO functions that take as an argument a list of page numbers or a list of page number pairs.

The purpose of this change was to make the RUN, R, and GET commands faster. Previously, pages were created one at a time, greatly adding to swapping when the system was loaded. This new feature allows all pages to be created at once.

The PAGE. monitor call can now map monitor low segment pages to user high segment pages, one page at a time, using function 11 (.PAGSP). This feature requires that the user have spy privileges. The form of the monitor call is:

```
        MOVE    AC,[XWD .PAGSP,addr]
        PAGE.   AC,
          error return
        normal return

addr:   Number of arguments + 1
        argument 1
           .
           .
        argument n
```

Each word of the argument list contains:

X	Monitor Page	User Page

where: X equals one if the monitor page should be removed from the user map (PA.GAF).

ENQC. Monitor Call (CALLI 153)

When the status of a given lock is requested via function 0 (.ENQCS) of ENQC., the number of jobs waiting for that given lock is now returned in the left half of the third word of the status block.

The argument block for this function (.ENQCS) has also been changed. Previously, the argument was in the form:

```
argblk: number of locks,,length
        0,,request i.d.
        flags,, channel-number
        string-pointer
        # of resources in pool,,# requested

buffer: BLOCK <number of locks> * 3
```

Now, the argblk takes this form:

```
argblk: EXP     parameters
        XWD     0,ident
        first word of first lock block
        ...
        last word of last lock block

buffer: BLOCK <locks>*3
```

where: parameters is a word of the form:
<headersize>B5+<locks>B17+<locksize>B35

ident is an enqueue request identifier.

headersize is the size of each lock block (0,3,4 or 5).

locks is the number of lock blocks in the argument list.

locksize is the length of each lock block (headersize plus any intervening words).

SCHED. Monitor Call (CALLI 150)

A new function has been added to the SCHED. monitor call, providing further flexibility in tuning the scheduler. The in-core protect time interval may now be set or read using function 27 (.SCSSC). JACCT or [1,2] privileges are required to set the in-core protect time interval.

TRMOP. Monitor Call (CALLI 116)

There are eleven new functions for the TRMOP. monitor call:

1. .TOBCT - returns the number of break characters input on the terminal in the RH and the number of commands input in the LH.

2. .TOICT - returns the total number of characters input from the terminal (except for characters used by MIC or the .TOTYP function of TRMOP.).

3. .TOOCT - returns the total number of characters output on the terminal. This does not include characters ^O or characters echoed locally by PDP-11s. However, it does include fill characters and characters echoed by MIC/.TOTYP.

4. .TOOSA - suppresses output, same as a ^O.

5. .TOFCS - returns one if the terminal is using the full character set.

6. .TOBKA - returns the break on character/line mode setting. If the mode setting is one, break on all characters.

7. .TOTIC - returns the number of available characters in a terminal's input buffer.

8. .TOBKC - returns the number of available break characters in the input buffer.

9. .TOECC - returns the number of characters in the input buffer not yet echoed.

10. .TOTTC - returns the total input characters count (.TOECC + .TOTIC).

11. .TOTOC - returns the total output character count.

The character counts are no longer kept in DDBs; they are stored in LDBs and should be set/read using TRMOP. functions. Users must have privileges enabled to get the count of characters mentioned above.

The carriage width for terminals can now be set to 255 characters using function 2Ø12 of the TRMOP. monitor call.

DSKCHR Monitor Call (CALLI 45)

The DSKCHR monitor call now returns four more words of data:

.DCALT	Unit name for alternate port
.DCOWN	Owner PPN of structure
.DCPAS	Position in active swapping list
.DCPSD	Position in system dump list

Words 27 and 3Ø will contain negative one if the structure is not in the appropriate system list.

JOBPEK Monitor Call (CALLI 103)

The JOBPEK monitor call has been modified to read another user's funny space. Two new flags have been defined in the flags byte of the first word of the argument block:

Flag	Symbol	Meaning
1B2	JK.EVA	Source address is between 340000 and 377777; treat it as though it were an executive virtual address mapped through the specified job's UPMP. Both JK.WRT and JK.UPM must be off.
1B3	JK.AIO	Do not block if the data is inaccessible.

JK.EVA is used to get at user funny space. However, reading another job's funny space may fail if the job is swapped out. If it is desirable not to block when reading funny space, turn on the JK.AIO bit also.

SETUUO Monitor Call (CALLI 75)

To provide support for the ENABLE/DISABLE monitor command, the SETUUO has a new function. The .STPRV, function 34, sets the privilege word for the job executing the monitor call. The procedure is:

```
MOVE      ac,[XWD .STPRV,arglst]
SETUUO    ac,
```

where the argument list holds:

```
arglst:   length,,subfunction
          argument
```

length: length of the argument list

subfunction (one of the following):

```
Ø  ST.CPW  Sets whole privilege word
1  ST.CPS  Sets specified bit of privilege word
2  ST.CPC  Clears specified bits of privilege word
3  ST.CCW  Sets whole capability word
4  ST.CCS  Sets specified bits of capability word
5  ST.CCC  Clears specified bits of capability word
```

There are two words that contain a job's privilege information. The capability word is filled from ACCT.SYS when the user logs in and holds the privileges that the system administrator has given the user. The privilege word holds those privileges from the capability word that are presently usable; it is also filled from ACCT.SYS. The user may disable the use of those privileges using the SETUUO. If the user wants to regain the use of privileges at a later point, he/she must issue another SETUUO. Both of these functions are also implemented as monitor commands (ENABLE/DISABLE) and are described in Chapter 10.

JOBSTS Monitor Call (CALLI 61)

An extra bit has been defined in the STATUS word returned by JOBSTS. Bit six will be turned on if the job is running; it will be zero if the job is in a wait state. Other new bits for terminals are:

JB.UFC - terminal is opened in full character set mode.

JB.UBK - terminal in break-on-all-characters mode.

JB.UNE - terminal is in no echo mode.

JB.UTO - job is blocked for terminal I/O.

JB.UCC - terminal characteristics changed since last JOBSTS was executed.

PEEK./POKE. Monitor Calls (CALLIs 33/114)

The PEEK. and POKE. monitor calls now have the ability to read/write physical addresses as well as monitor virtual address space. If a user sets UU.PHY in the call statement:

```
PEEK    AC1,UU.PHY
```

the address is treated as a physical address.

FILOP. Monitor Call (CALLI 155)

Nine new functions have been added to the FILOP. monitor call.

1. Function 16 opens a device for super I/O, disk I/O to specific physical block numbers. The format for the argument list for this function, .FOSIO is:

   ```
   arglst:     XWD    channo number,.FOSIO

                      mode
                      device-specification
                      buffer-address
   ```

   where: <u>channo</u> is the channel number

   <u>mode</u> is usually dump mode (.IODMP)

   <u>device-specification</u> is the device name in SIXBIT

   <u>buffer-address</u> is 0 for dump mode

2. Function 17 (.FOINP) performs an IN from the file opened on the specified channel. The argument list is:

   ```
   ADDR:  XWD    channo,.FOINP
          address of the next buffer
   ```

3. Function 20 (.FOOUT) performs an OUT to the file opened on the specified channel. The argument list is:

   ```
   ADDR:  XWD    channo number,.FOOUT
          address of next buffer
   ```

4. Function 21 (.FOSET) performs the SETSTS monitor call. The argument list is:

   ```
   ADDR:  XWD    channo number,.FOSET
          EXP    SETSTS bits
   ```

5. Function 22 (.FOGET) performs the GETSTS monitor call. The argument list is:

   ADDR: XWD channo number,.FOGET

   The status bits are returned in the AC.

6. Function 23 (.FOREL) performs the RELEAS monitor call. The argument list is:

   ADDR: XWD channo number,.FOREL

7. Function 24 - (.FOWAT) allows a user to wait for I/O to finish on a channel. The FILOP. argument list for this function is:

   ADDR: XWD channo number,.FOWAT

8. Function 25 (.FOSEK) performs the SEEK monitor call. The argument list is:

   ADDR: XWD channo number,.FOSEK

9. Function 26 (.FORRC) will cause TOPS-1Ø to rewrite the RIB of a file. Nothing will be done if the RIB has not changed or if the device is not a disk. The format of the argument list is:

   ADDR: XWD channo number,.FORRC

   where the channel is connected to an open UFD.

The FILOP. monitor call was also modified to provide for extended channels. When performing any FILOP. operation with extended channels, bit one (FO.ASC) of the first word of the argument block must be set.

NODE. Monitor Call (CALLI 157)

For a description of the new functions to the NODE. UUO, refer to Chapter 11, Networks.

SPPRM. Monitor Call (CALLI 172)

A second function has been added to the SPPRM. UUO. Function 2 (.SPSPR) allows spooled parameters to be set for renamed files. Previously, spooled files that were renamed could not have parameters set. The first word of the argument block (.SPPFN) contains:

0 - Set spooled file parameters
1 - Set spooled parameters for renamed files

DISK. Monitor Call (CALLI 121)

The DISK. UUO has six new functions, numbers 7 to 14.

1. Function 7 (.DUUFD) calls the UFD compressor. The UFD compressor shrinks the size of a UFD by eliminating deleted files from the UFD. The argument block contains a channel number, one open to the UFD.

2. The next two functions, 10 (.DUSWP) and 11 (.DUASW), remove from or add to the active swapping list. The argument block holds the drive name to add/remove.

3. Functions 12 (.DUASD) and 13 (.DURSD) perform the same function as functions 10 and 11 except they apply to the system dump list. The argument block holds the structure to add/remove from the system dump list.

4. Function 14 (.DULEN) returns the length of a file in the accumulator. The argument block holds the channel number that the file has been OPENed on.

GETTAB Changes

For a complete list of GETTAB changes, see Appendix C.

CPU-DEPENDENT MONITOR CALLS

DTE. Monitor Call (CALLI 170)

The DTE. monitor call had to be modified because DTEs may now bee connected to either CPU in a multiprocessing system. The following functions now require a CPU number in the left half of the first word in the argument list.

0	Clear a DTE
1	Start primary protocol on a DTE
2	Set byte pointer to the KL10
3	Set byte pointer to the PDP-11
4	Return the PDP-11 reload ROM word
5	Return master DTE number for this CPU
6	Press the PDP-11 reload number
7	Return the DTE status word
11	Return the front-end device unit number
12	Front-end device input
13	Front-end device output
14	Return front-end device status
15	Set front-end device status
16	Release a front-end device
17	Release KL error chunks

20 Release KL error timer

21 Return RSX2ØF line numbers for the DTE and CPU

PERF. Monitor Call (CALLI 162)

The PERF. monitor call now requires the number of the CPU that is to be measured for performance. The CPU number should be placed in the left half of the third word of the argument block.

RTTRP. Monitor Call (CALLI 57)

The RTTRP. monitor call has been modified to run on either CPU; however, three restrictions still apply to its use. The job must be:

1. locked in core

2. on the correct CPU with

3. the real-time trap privilege bit turned on

DIAG. Monitor Call (CALLI 163)

The DIAG. monitor call performs diagnostic monitoring of devices and CPUs. The DIAG. monitor call has been enhanced to work on either of the CPUs and to report its findings to ERROR.SYS via DAEMON.

There are four new functions.

1. Function 1Ø - .DIACS reads the CPU status block on a CPU and forces DAEMON to make an error entry (code 63) in ERROR.SYS. The call is made in this fashion:

```
                MOVE    ac,[2,,addr]
                DIAG.   ac,

        addr:   EXP     .DIACS
                EXP     CPU-number
```

2. Function 11 - .DIADS reads the device status block on a CPU and requests DAEMON to make an error entry (code 64) in ERROR.SYS. To perform this function, use the same calling sequence and argument block as above but insert .DIADS instead of .DIACS.

3. Function 100 - reads the status of MOS memory.

4. Function 101 - sets the status of MOS memory.

Functions 100 and 101 are used primarily by TGHA.

Chapter 10

CHANGES TO SYSTEMS PROGRAMS

The following system programs have been modified or created for TOPS-1Ø Version 7.Ø1. The latest version number follows each CUSP.

AVAIL	2	BOOT11	4A
BOOTDX	3	BOOTM	5
BOOTS	23	CRSCPY	1
CONFIG	1	DAEMON	2Ø
DDT	41	DTELDR	4
FILDAE	2	HELP	6
INIITA	7A	KDPLDR	1
~~LINK~~	~~4A~~	LOGIN	61
~~MACRO~~	~~53A~~	MAKPFH	-Eliminated-
MAKVFU	2	MONGEN	52
MOUNT	1	NETLDR	3
NETWOR	1	OMOUNT	27
OPSER	5D	PFH	2
SYSDPY	434	SYSERR	15
SYSTAT	474	TGHA	2
TWICE	4B	UUOSYM	15

There are six new CUSPs: AVAIL, for reporting system availability; CRSCPY, to copy system crashes from CRASH.EXE; MOUNT, to handle user mount and allocation requests; KDPLDR, for loading the microcode on the KMC/DUP11 on the 2Ø2Ø; CONFIG, for dynamically reconfiguring an SMP system; and NETWOR, which is a replacement for the NODE command.

AVAIL

For a description of AVAIL, refer to Chapter 3, KL Service Enhancement Project.

BOOTSTRAP PROGRAMS

The changes to the bootstrap programs are as follows.

BOOTS - This is a mai ntenance release to add support for all eight RH2Ø controllers on a KL1Ø processor.

BOOT11 - For information refer to Chapter 11, Network Changes.

BOOTM - This has added support for TU7xs running on a RH2Ø-DX2Ø.

BOOTDX - This has been modified to read .BIN or .A8 files. The /ERR:n switch has been removed. It also has four new switches:

1. /CPU:n Select CPU on which device exists.
2. /DX1Ø Device is a DX1Ø.
3. /DX2Ø:k Device is a DX2Ø with controller k. At the present time, only TXØ2 is supported.
4. /TYPE:T Look for DX2Ø device-type T. Low-order bits of RH2Ø register 6; e.g. 6Ø for TXØ2

CONFIG

CONFIG is a program to simplify the addition and removal of devices, CPUs, disks, and memory, to and from the system. CONFIG has four commands:

1. ADD - to add a piece of hardware
2. REMOVE - to remove a piece of hardware
3. HELP - for help in using CONFIG
4. EXIT - to exit from CONFIG

For a further description of CONFIG, see Chapter 9, Operator Changes.

CRSCPY

For a description of the CRSCPY program, refer to Chapter 3, KL Service Enhancement Project.

DAEMON

For a description of the changes to DAEMON, refer to Chapter 3, KL Service Enhancement Project.

DDT

DDT Version 41 is significantly different from previous versions and has many new features, some of which are not compatible with those of previous versions. DDT now takes better advantage of features now available, such as cache and paging.

Some of the command changes involve:

1. Memory and address control
2. Specifying the starting address
3. Symbolic expression type-in and type-out
4. ASCII type-out
5. Command files
6. Automatic patch insertion
7. Breakpoints
8. Single-stepping the program
9. Searches
10. Zeroing memory
11. RADIX-5Ø symbol type-in

12. New DDT runtime information

13. Obsolete commands

14. FILDDT startup and commands

For a full description of changes to DDT, see Appendix C.

DDT.VMX is no longer shipped because the MERGE. monitor call has eliminated all files with the .VMX extension.

DTELDR

For a description of the changes to DTELDR, refer to Chapter 11, Network Changes.

FILDAE

The purpose of this release of file DAEMON is to support full file specifications on the /PROGRAM switch, including SFDs. The switch may have the following form:

/PROGRAM:STR:FILE.EXT[P,PN,SFD1,SFD2,...SFDn]

File DAEMON will consider the program accessing the file to be the same as the program specified in the /PROGRAM switch only if the full file specification matches the argument in the /PROGRAM switch.

HELP

The HELP program has the following changes:

1. Uses TTY width to calculate the number of columns of output.

2. Types the names of the terminals supported by the monitor.

3. Will type-out customer-added commands.

4. Has a better sort algorithm that runs faster.

INITIA

INITIA has been modified so that it can set the terminal type. For example:

```
.INITIA SET TTY TYPE VT52
```

The output from the TTY command has been modified to include the TTY TYPE and PAGE SIZE, and to use tabs to make the display more readable.

```
.INITIA/TTY

     RZ357A KL #1026/1042 15:23:25 TTY30 SYSTEM 1026/1042
CONNECTED TO NODE KL1026(26) LINE #30
DSKC:       KL1026 SYSTEM DISK DSKC
DSKB:       KL1026 SYSTEM DISK DSKB
TYPE:VT52      LC             TABS           FORM
ECHO           CRLF           WIDTH:80       GAG
DISPLA         FILL:0         NOTAPE         PAGE:23
BLANKS         NOALTMOD       NODEBREA       NOTIDY
NORTCOMP       NOSLAVE        NOHDX          NOREMOTE
RCVSPEED:150   XMTSPEED:9600
```

INITIA also has been modified to perform more functions upon system startup. For a description of these new features, read Chapter 9, Changes to Operator Procedures.

KDPLDR

For a description of the KDPLDR program, refer to Chapter 11, Networks.

LINK

LINK Version 4A is a maintenance release and as such has no new features. For a list of bug fixes, see LNK4A.DOC on the monitor tape.

LOGIN

Several new switches and prompts have been added to LOGIN, including the following:

1. If a user is logging in and has a detached job logged in under the same PPN, he/she will be prompted in the following manner:

```
.LOG 77,4655
JOB 20    RZ364A KL  1026/1042    TTY 60
PASSWORD:

OTHER JOBS DETACHED WITH SAME PPN: 20
TYPE JOB NUMBER TO ATTACH OR CARRIAGE RETURN TO
LOGIN A NEW JOB:
```

   When the user does not want this prompt when he/she has other detached jobs, a new switch has been provided: /ATTACH:ASK or /ATTACH:IGNORE. If /ATTACH:IGNORE is used, the user will never be prompted about other detached jobs. If /ATTACH:ASK is used, the user will be prompted. /ATTACH:ASK is the default.

2. The /TERMINAL switch takes a list of parameters to specify terminal attributes. Multiple keywords may be given to the /TERMINAL switch, in which case they should be enclosed in parenthesis and separated by commas. Commands that can be used with the SET TTY monitor command may be used with this switch. For example:

```
.LOGIN 10,1/TERMINAL:(TYPE:VT52,PAGESIZE:0,NOGAG)
```

3. The /WATCH:FILE implements the SET WATCH FILE command for LOGIN. For more information on this SET command, see Chapter 10, User Level Changes.

4. The /ACCOUNT and /REMARK switch as well as prompts of the same name are discussed in Chapter 5, Usage Accounting.

5. Three new switches have been implemented for Galaxy Version 4; /BATNAM:arg, /BATSEQ:n, and /REQID:arg. All are discussed in Chapter 4, Galaxy.

MACRO

MACRO Version 53A is a more reliable version of the product. The following eleven external changes were made:

1. BYTE Pseudo-Op

   Externals must be loaded into either fullword (36-bit) or halfword (18-bit) bytes. Relocatables must be loaded into fullword or halfword bytes, with the exception that a right-half-only relocatable symbol can only be loaded into a right-justified byte that is between 18 and 36 bits in length. Examples:

   ```
   BYTE (18) REL1, REL2 (6) 3 (30) RHREL3
   BYTE (10) 5 (26) RHREL4
   ```

2. Unary Operators

   Unary operators now properly take precedence over all binary and shift operators. This conflicts with the existing documentation. The hierarchy is now as follows:

   a. All unary operators (+, -, D, O, B, F, L, E, K, M, G)

   b. Shift operators (B-shift, underscore-shift)

   c. Logical binary operators (!, _!, &)

d. Multiplicative operators (*, /)

e. Additive operators (+, -)

3. Error Message Standard

On TOPS-10 systems, MACRO now observes the setting of the message-level bits during error message typeout (See Monitor GETTAB Table .GTWCH(35)).

4. MACRO Call Statements and XALL

MACRO now lists the entire statement in which a macro call appears if XALL is in effect. This may cause the source line to be broken up in the listing if multiple macro calls appear on the same line, and if the called macros have imbedded line terminators. Comments on macro calls imbedded within conditionals will also be listed, with the comment appearing along with the last line of the macro expansion, even if no code is generated for that line.

5. .IF/.IFN Pseudo-Ops

Contrary to what the documentation states, the .IF and .IFN pseudo-ops were never intended to take angle-bracketed expressions as arguments. Instead, they are intended to provide a powerful mechanism for querying symbol table attributes, given a valid mnemonic consisting of up to six RADIX50 characters. The newly defined attribute "NAME" can be used to determine if the argument is a single RADIX50 symbol name. Such a symbol may be enclosed in angle brackets and nested to any depth. The use of angle-bracketed expressions is allowed, provided they are treated as described below, but their use is not recommended.

Limited expression handling has been implemented to handle situations in which angle-bracketed expressions are used as arguments. Such an argument returns the combined, IORed, attributes of its components, except in the case of relocation its components, except in the case of relocation attributes, which are handled correctly only for a symbol. In addition, an angle-bracketed expression has the newly defined attribute "EXPRESSION", providing a means of defending against angle-bracketed expressions passed as arguments to macro calls. Note that all arguments that work outside angle brackets work inside them as well.

6. PSECTS and Literals

Proper termination of literals within code bounded by .PSECT/.ENDPS is now enforced. Early termination of a literal that was started in the same PSECT but at a higher level due to nesting, or the failure to terminate a literal that was started in the current PSECT will produce an L error and generate the new MCRLNI literal nesting incorrect message.

7. Suppressed Symbols

Symbols declared as suppressed internal, but defined by externals or Polish will now retain the suppressed attribute and will not be typed out by DDT.

8. PSECT Attributes

Valid PSECT attributes and their definitions are as follows:

CONCATENATED — When loading multiple modules, all PSECT blocks having the same name will be concatenated at load time.

OVERLAID — When loading multiple modules, all PSECT blocks having the same name will be loaded starting at the same origin. Each module will "overlay" the previous one.

RWRITE	The PSECT can be read and written on TOPS-20 only.
RONLY	The PSECT can be read on TOPS-20 only.
PALIGNED	When loading relocatable PSECTs (which is not yet supported), the PSECT origin will be started at a page boundary.

9. PSECTs and PRGENDs

PRGENDs can now occur in a source file containing PSECTs.

10. Assembler Type-Out for PRGENDed Programs

When MACRO is executed via a RUN command, and the listing device is not the terminal, the informational messages (BREAK, CPU TIME, CORE USED, etc.) will be typed out for each program module in a PRGENDed source file. The particular program module will be identified by a line following the error counts, indicating "PROGRAM XXXX", where "XXXX" is the information appearing in the TITLE statement for the module.

If no TITLE is specified, the default TITLE, ".MAIN", will be typed. The program identification line will no longer be output to the listing file. However, it will now appear with the errors typed out when MACRO is invoked via the CCL entry point (as done by the COMPIL class commands under TOPS-10).

11. New Error Messages

a. "Q" errors:

(1) @ appearing in or before AC field.

(2) Index value has left half for

POLISH(INDEX).

(3) BYTE (n) RELOCATABLE where n is not 36 or halfword 18.

(4) @ in unbracketed expression to DEC/EXP/OCT.

(5) Multiple TITLEs or TITLE/UNIVERSAL conflicts, was M error.

b. "E" errors:

(1) BYTE (n) EXTERNAL where n is not 36 or halfword 18.

(2) BYTE (n) POLISH symbol, where n is not 36 or halfword 18.

(3) EXTERN/INTERN conflicts.

c. "X" errors:

(1) Created symbol exceeds ..7777.

d. "A" errors:

(1) OPDEF produces no code.
(2) PSECT origin not absolute.

e. "L" errors:

(1) PSECT literal-level mismatch.

(2) Assignment involves a label defined within a literal.

f. New MCR error message:

(1) MCRLNI - Literal nesting incorrect at end of PSECT XXXXXX.

g. Modified MCR error messages:

(1) MCRPTC - Polish too complex for location nnnnnn.

(2) MCRPTC - Polish too complex for symbol XXXXXX.

h. Deleted MCR error messages:

(1) MCRPIP - PRGEND illegal with PSECT.

MAKPFH

All page fault handlers (PFHs) are now created using MACRO and loaded using the MERGE command or MERGE. monitor call. PFH are standard EXE files. This means that the program MAKPFH, for making page fault handlers, no longer exists. The extension of the user's PFH is now .EXE and not .VMX. Refer to Chapters 6 and 1Ø for more information on how to load page fault handlers using MERGE.

MAKVFU

Three new commands have been added to MAKVFU. The TYPE command sets defaults for a specific line printer type. The CLEAR command clears all settings for a specific VFU channel. The NEW command clears the entire VFU.

MOUNT

MOUNT replaces the UMOUNT program. It sends user ALLOCATE and MOUNT requests to the Galaxy system using IPCF messages. Although the UMOUNT program may still appear to exist, it is really a copy of the MOUNT program.

NETLDR

Changes to NETLDR are discussed in Chapter 11, Networks.

NETWOR

The new program NETWOR is discussed in detail in Chapter 11, Networks.

OMOUNT

OMOUNT now allows disk packs in the active swapping list to be mounted or dismounted.

OMOUNT will warn the operator if the structure being removed comprises packs in the active swapping list. If the operator proceeds, OMOUNT will have the monitor migrate swapping to the pack(s) prior to unloading the drives.

```
.R OMOUNT

/REMOVE DSKA
MOUNT COUNT = 23
IN ACTIVE SWAPPING LIST
PROCEED?
DSKA   REMOVED... DRIVE(S) FSA0
--DONE--
/
```

When a structure is being mounted, OMOUNT will check the home blocks of the pack(s) to see if swapping space exists. If the space does exist and the structure fits into the slot in the active swapping list that it had before, the structure will be used for swapping. If the pack has space for swapping, but was not previously marked as being in the active swapping list, the monitor will not add that pack's unit to the active swapping list.

When Galaxy 4.1 is released, OMOUNT will be replaced by the MOUNT program.

OPSER

The following changes have been made to OPSER:

1. When KSYS reaches zero, all subjobs are killed.

2. When OPSER is run via INITIA to automatically run an auto file, the following steps are used to determine the name of the auto file to process.

   a. If run on the "OPR" line (usually CTY), OPSER uses SYS:OPR.ATO.

b. If run on any other local (non-network) line, SYS:TTYnnn.ATO is used where nnn is the line number. For example, if run on line four, the file would be called SYS:TTY4.ATO.

c. If run on the CTY of a remote station, the file is called SYS:OPRn.ATO where n is the node number.

d. If run on any other network line, SYS:nnnlll.ATO is used where nnn is the node number and lll is the line number; leading zeroes are supplied. (e.g., SYS:031042.ATO for line 42 on node 31.)

SYSDPY

Apart from network changes, the only improvements to SYSDPY are that it now supports VT50s, VT52s, and VT61s, and it displays SMP statistics and batch queues. VT100s are treated as VT52s by SYSDPY. For network changes to SYSDPY, see Chapter 11.

SYSERR

For a description of the changes to SYSERR, refer to Chapter 3, KL Service Enhancement Project.

SYSTAT

SYSTAT Version 474 is a maintenance release designed to handle Version 7.01 of TOPS-10. The following six changes have been made:

1. The /R switch has been removed.

2. Disk DDBs are only printed in the busy device status if your job is specified because disk DDBs are now kept in funny space and can only be read reliably by your job. SYS X will continue to list the disk DDBs for those jobs in core at the time of the crash.

3. The SYS F command now prints "PRIVATE STRUCTURE" for those structures that are private.

4. SYSTAT now accepts constructs of the form [,N], [N,] and [,] for wildcarding. For example, ".SYS [1Ø,]" will print all jobs with a project number of 1Ø.

5. The SYS X command now accepts a full SFD list in the crash file specification.

6. The SYS U command displays user names.

TGHA

For a description of the TGHA program, refer to Chapter 3, KL Service Enhancement Project.

TWICE

The following changes have been made to TWICE:

1. The "ALL" response is no longer valid to the "WHICH STRUCTURE TO REFRESH?" question.

2. TWICE now asks "NOT NORMALLY DONE, ARE YOU SURE?" when asked to refresh a structure not marked as needing refreshing. In addition, if the structure is marked as in the system search list, TWICE also asks "THIS IS A SYSTEM STR, ARE YOU POSITIVE?".

3. If "ALL" is typed as an answer to the question asking which disks to read or write, TWICE does not type off-line messages for those disks which are off-line.

4. All changes to the ONCE dialog hold for TWICE also (e.g. the system dump list questions).

UUOSYM

UUOSYM has been updated to reflect the new and modified monitor calls of TOPS-1Ø Version 7.Ø1.

Chapter 11

MONITOR BUILDING CHANGES

MONGEN CHANGES

Customer-Defined ERSATZ Devices

In the hardware configuration section of MONGEN, users are now able to define their own ERSATZ devices. MONGEN will ask:

```
TYPE "ERSATZ-DEVICE,P,PN,SEARCH-LIST-TYPE"[
FOR CUSTOMER DEFINED ERSATZ DEVICES.
THE "ERSATZ=DEVICE" MUST BE EXACTLY 3 CHARACTERS LONG
"P,PN"IS THE PROJECT/PROGRAMMER NUMBER ASSOCIATED.
"SEARCH-LIST-TYPE" IS ONE OF ALL, SYS, JOB.
TYPE EXTRA CARRIAGE RETURN WHEN THROUGH.]
```

The user may then respond:

```
RSG,77,4655,ALL
```

to create the ersatz device RSG: (as [77,4655] searching all structures). The search-list-types are:

JOB	Use the job's search list.
SYS	Use the SYS search list.
ALL	Use the ALL search list.

KA10 LONG FLOATING-POINT INSTRUCTION SIMULATION

To include the KA10 floating-point instruction simulation, type a "Y" to the following question in the hardware configuration section:

```
KASER(Y,N)[INCLUDE SUPPORT FOR SIMULATION OF KA10
LONG FLOATING POINT INSTRUCTIONS]:
```

This is used in conjunction with the "SET FLOATING POINT SIMULATION" monitor command to allow the execution of KA10 floating-point instructions.

Account Verification

This feature is needed to include account verification, to be released with Galaxy 4.1. To include account verification, type a "Y" to the following question in the hardware configuration section:

```
ACCOUNT VERIFICATION(Y,N)[LOGIN AND THE BATCH SYSTEM WILL
VERIFY THAT THE USER HAS SPECIFIED A VALID ACCOUNT]:
```

TMPCOR Change

Because TMPCOR is now kept in user funny space, the question:

```
CCL COMMANDS TO STAY IN CORE(Y,N)[SYSTEM PROGRAMS PASS
COMMANDS TO EACH OTHER VIA CORE (TMPCOR UUO) RATHER
THAN DISK]:
```

has been eliminated from MONGEN.

MONBTS

To include the monitor-resident BOOTS (MONBTS) described in Chapter 3, type a "Y" to the following question:

```
MONITOR RESIDENT BOOTS(Y,N)[CORE RESIDENT BOOTS
ALLOWING FAST DUMP/RELOAD, AND CONTINUABLE STOPCD DUMP,
ETC.]:
```

If this feature is selected, be sure to include the module MONBTS when the monitor is compiled and loaded.

MOS Memory Support

If the monitor is being made for a KL1Ø with MF2Ø MOS memory, 1Ø9Ø or 1Ø91, and TGHA support for MOS is desired, type "Y" to the following question:

```
MOS MEMORY(Y,N)[INCLUDE SUPPORT FOR USER MODE MOS MEMORY
DIAGNOSTIC TGHA WHICH COLLECTS STATISTICS ON SINGLE BIT
MEMORY ERRORS AND SUBSTITUTES THE SPARE BIT WHEN A SINGLE
BIT ERROR IS DETERMINED TO BE HARD]:
```

Be sure to include the module MOSSER when making the monitor if this feature is chosen.

Unsupported Devices

Because 7.Ø1 does not support KA1Øs or KA1Ø-type devices, those MONGEN questions related to DC1Øs, DC76s, DA28s, DC68s, and display devices have been placed in a MACRO conditional, FTUNSUPP, within MONGEN. The supported version of MONGEN will not include questions about these devices.

Feature Test Switch Changes

The following test switches have been removed:

1. FTKA1Ø Removed as though OFF.
   CPU is a KA1Ø.

2. FTPDBS Removed as though OFF.
   Swap PDB with job.

3. FTSHF Removed as though OFF.
   The KA1Ø shuffling code.

4. FTSWAP Removed as though ON.
   Allow jobs to be swapped.

5. FTLOGIN Removed as though ON.
   A single user system.

6. FT2REL Removed as though ON.
   Use relocation registers.

7. FTDISK Removed as though ON.
   A disk system.

8. FTVM Removed as though ON.
   Virtual memory is used.

9. FTTMP Removed as though OFF.
   CCL files kept in core.

The changes were made to eliminate KA1Ø code and remove unused options.

The FTMS (master/slave) feature test switch has been renamed to FTMP (multiprocessing).

There are three sets of standard feature test switches: KSFULL, KIFULL, and KLFULL.

Terminals

DECsystem-1Ø8Ø and -1Ø9Ø systems can now have terminals connected to the RSX2ØF front-end. The question:

TERMINALS ON THE MASTER FRONT END ON CPUx(Ø,Ø-128):

should be answered if any terminals are to be connected.

Because network lines run INITIA only when connected, the question "WHICH TERMINAL LINES RUN INITIA?" is more accurately stated as "WHICH LINES RUN INITIA AT ALL?" or "WHICH LINES RUN INITIA WHEN CONNECTED TO THE HOST?".

Renamed Modules

The following two modules have been renamed:

1. KILOCK is now LOKCON.
2. CP1SER is now CPNSER.

MONITOR CONTROL FILES

In an effort to make the monitor-building procedure simpler, a series of control files have been created to help the systems programmer. Even if the control files are not followed exactly, they can still serve as the basis for an individually tailored system-building procedure. The necessary steps are summarized by the following flowchart in Figure 11-1.

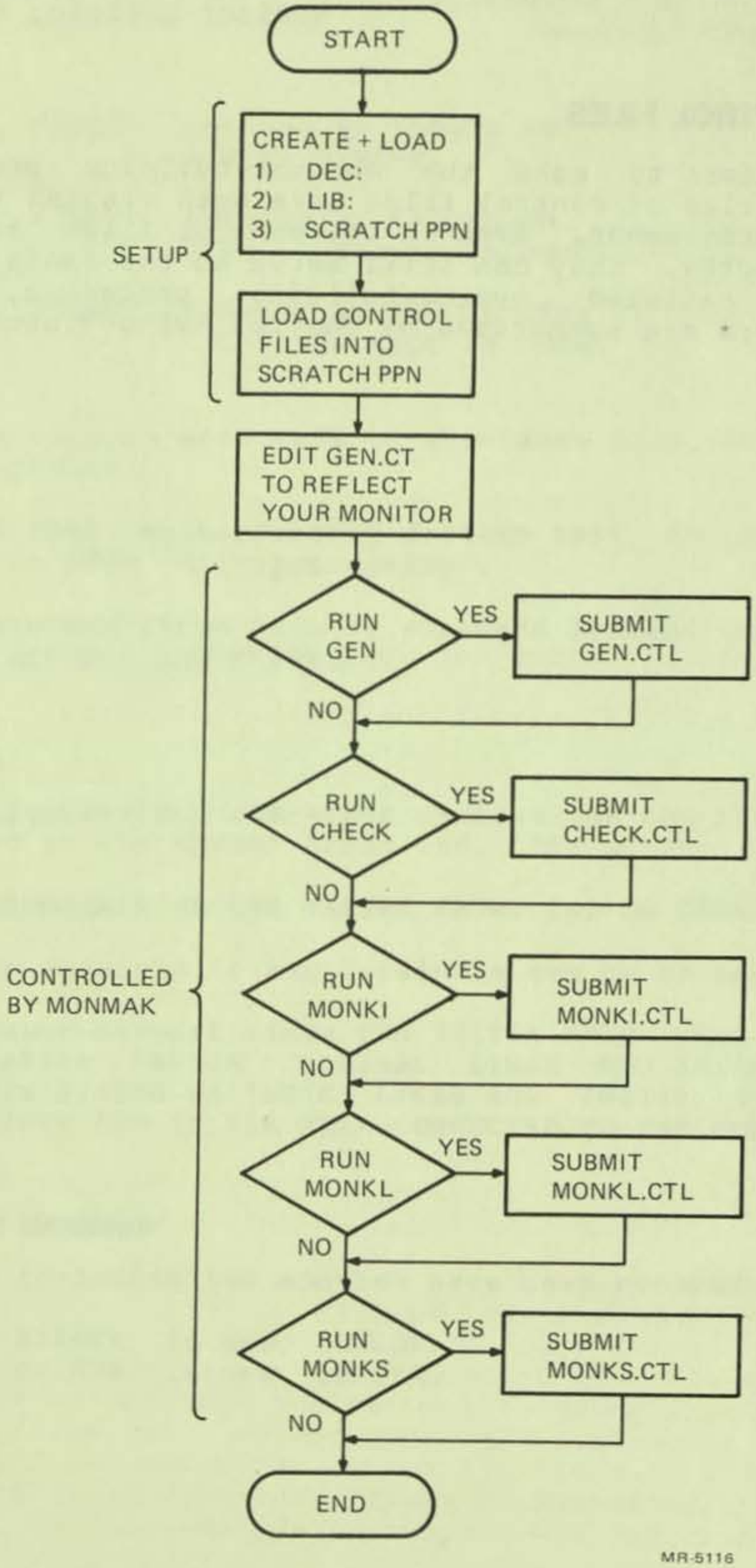


Figure 11-1 Monitor Building Procedure Using Control Files

Setup

The new monitor-building procedures require the use of three areas:

1. DEC: ([1Ø,7]) - to hold the most current versions of the CUSPS: PIP, CREF, MACRO, etc.

2. LIB: - to hold the monitor sources. This library will be set only for the job using SWITCH.INI or SETSRC.

3. Scratch PPN - where you will build the control files and monitor. The PPN should have at least 24,ØØØ blocks for a complete monitor build.

The following control files might be needed at various times and should be loaded into the scratch PPN:

1. MONMAK.CTL - overall monitor-build control file.

2. CHECK.CTL - control file to check the syntax of conditionals in monitor sources.

3. Kx1Ø.FIL - holds lists of monitor modules necessary to make a monitor. X equals S for KS1Ø, I for KI1Ø and L for KL1Ø.

4. ORDER.FIL - checklist of monitor-building steps.

5. MONKx.CTL - control file to compile, link, load, and save a monitor. See Kx1Ø.FIL for naming conventions.

6. GEN.CT - control file to run MONGEN and produce the configuration files.

Modifying Configuration Control Files

Once the PPNs and control files are in place, the next step is to modify GEN.CT to meet the specific needs of the system being generated. GEN.CT produces the configuration files necessary to compile the monitor by running the MONGEN program. As distributed, GEN.CT generates six or seven sample monitors that are currently used in-house at DIGITAL.

Remove the unneeded MONGENs from GEN.CT and modify the dialogue closest to the system being generated. The output files from GEN.CT should be Fx.MAC, HDWKx.MAC, TTYKx.MAC and NETKx.MAC where x equals S for the KS1Ø, I for the KI1Ø and L for the KL1Ø.

MONMAK.CTL

MONMAK is the control file that initializes the building process. It modifies the .CT files to include the current load number, creates sorted listings of the important directories, and optionally submits other control files. It asks the following questions in this order:

```
ENTER 3 CHARACTER VERSION NUMBER -
DO YOU WANT TO START CHECK ?
DO YOU WANT TO START GEN ?
DO YOU WANT TO START MONKI ?
DO YOU WANT TO START MONKL ?
DO YOU WANT TO START MONKS ?
```

MONMAK submits the control files to run in the order the questions are asked.

CHECK.CTL <Optional>

CHECK is an optional control file that determines whether or not all angle brackets in the monitor module conditionals are matched properly. It may run at any time after MONMAK and is not dependent on any other jobs. All imbalances are recorded in CHECK.ERR.

GEN.CT

If selected to do so, MONMAK takes GEN.CT, renames it GEN.CTL, and inserts the proper load number. GEN.CTL produces all the configuration files and must be run before MONKx.CTL runs.

MONKx.CTL

There are three control files: MONKI for KI1Øs, MONKL for KL1Øs, and MONKS for KS1Øs. They will compile, link, load, and save the TOPS-1Ø monitors that must be built. In addition, the control files ask if CREF listings should be produced and do the necessary work to make the listings. At this point, monitor building is over.

Chapter 12

OPERATOR PROCEDURE CHANGES

Many things have changed for the TOPS-10 operator; the new OPR/ORION Galaxy interface, the MONBTS/CRSCPY programs, new RSX20F commands, and more. Despite the fact that many things are new, the overall job of the operator has been simplified. All dealings with user requests, including PLEASE, now go through OPR. OPR standardizes the commands for dealing with all user requests. Continuable stopcodes mean that the system will have to be reloaded less often and CRSCPY will automatically copy the dumps. Operators will see more output on the CTY but will have to respond to less. The chapters on Galaxy, usage accounting, and the KL service enhancement project contain most of the new features that the operator will have to deal with and should be studied closely. This chapter will present the changes that do not fall into any of the previous categories.

NORMAL OPERATING PROCEDURES

Starting the System

Version 7.01 does not automatically reject a date/time if it is previous to a known time. Instead, the question "Is this correct? (Y or N)" is asked. If the operator supplied date is before that of a known crash, the following message is printed.

```
    RZ356A KL #1026/1042 06-06-79
DATE: 12-MAR-80
TIME: 13:10:00

%SPECIFIED DATE IS PRIOR TO LAST CRASH.
LAST CRASH DATE: 14-APR-80
IS THIS CORRECT? (Y OR N)
```

If the monitor creation date is askew, the following is printed.

```
%SPECIFIED DATE IS PRIOR TO MONITOR CREATION DATE.
CREATION DATE: 14-SEP-8Ø
IS THIS CORRECT? (Y OR N)
```

The questions shown above can be repeated if the date is correct but the supplied time is previous to a known time.

The startup option must then be supplied:

```
STARTUP OPTION:
```

There are seven switches to use with the startup option:

/NOVALIDATE - Turns off account validation for Galaxy Version 4.1.

/NOPRIMARY - Brings the system up with no Primary Protocol, used for debugging KL1Øs.

/ASK - Asks if the unit should be up or down.

/HELP - Explains the ONCE options.

/NOERROR - Do not type messages preceded by the "%" sign.

/OFF - Types messages about off-line units.

/STANDALONE - Brings monitor up in stand-alone mode.

CHANGING THE SYSTEM DUMP LIST

If MONBTS (Monitor Resident Boots) has been included in the monitor, it is necessary to use the ONCE dialogue to configure the system dump list, SDL. This is done by answering the following question:

```
TYPE STRUCTURE NAMES FOR SYSTEM DUMP LIST(EXTRA <CR> WHEN DONE)
LSCD:
RSGØ:
```

The example shown above places LSCD in the SDL first and in RSG0: second.

For KL10 SMP systems, RSX-20F stores the serial number of the boot CPU in the configuration file. This means that the same CPU will be used as the boot CPU until role switching or a J406 is executed on the non-boot CPU. Both CTYs will only be needed on a cold start.

JUMP Instructions

Besides location 400, there are three other possible starting addresses for use with SMP systems: 405, 406, and 407. They are described below.

JUMP 405 - Starting the boot CPU at 405 is the same as a 407 restart. On all other CPUs, a 405 start causes the CPU to save the machine state and jump into the ACs until it is manually restarted with a JUMP 400 command.

JUMP 406 - This command is used to force BOOTS to be loaded on this CPU regardless of which processor is the boot CPU. The CPU that executes a JUMP 406 automatically becomes the boot CPU and then executes a JUMP 407.

JUMP 407 - A JUMP 407 is used to dump the current monitor and reload the system. A programmer using DDT can get back to BOOTS and dump the monitor with this start location.

SYSJOB.INI Command File

A new feature has been added to INITIA which enables it to execute a command file using the FRCLIN function. When the system first comes up and INITIA runs on the CTY, INITIA will take commands from a file, SYS:SYSJOB.INI. It is expected that this file will be used to start [1,2] jobs that run detached. This enables the operator jobs to start faster and not use up OPSER job slots. Here is a sample SYSJOB.INI file:

```
;THIS IS A SAMPLE SYSJOB.INI FILE
SET DEFAULT ACCOUNT MUMBLE
LOG
GET SYS:ACTDAE
```

```
CSTART
DETACH
LOG
DAEMON
LOG
FILDAE
LOG
ORION
LOG
QUASAR
LOG
SYSINF
```

GALAXY OPERATOR

INITIA has been modified to automatically run OPR on a terminal when the system starts up. This is useful to see Galaxy messages and have control during the first few phases of running the system. The terminal that runs OPR is chosen by placing "GALOPR" in SYS:TTY.INI next to the chosen terminal line.

Time Stamp

A new routine called BIGBEN prints a time stamp (system uptime and date/time) on the CTY. By default, the time stamp is printed once an hour.

```
!Ø6:53:56(OPR\3)
        6:53:56   --NETWORK NODE KL1Ø26(26) IS OFFLINE--
        OPR>
!
MONITOR          RSØ44A KS #41Ø1
SYSTEM UPTIME 66:54:54
CURRENT DATE/TIME 14-APR-8Ø 7:ØØ:ØØ

Ø7:ØØ:22(OPR\3)
        7:ØØ:22   --NETWORK NODE KL1Ø26(26) IS ONLINE--
```

To change the time stamp interval, define the symbol M.OMSM in MONGEN to the number of minutes between time stamps.

Role Switching

When the non-policy CPU becomes the policy CPU, the following message is printed on the CTY:

CPUx HAS ASSUMED THE ROLE OF POLICY CPU ON <date/time>

If a role switch is forced due to the removal of one CPU, the later addition of that CPU will not force another role switch back to the previous state.

RECONFIGURING THE SYSTEM

There are two ways to reconfigure a system: using privileged monitor commands or using the CONFIG program. It is recommended that the CONFIG program be used because CONFIG forces the operator to deal with conditions explicitly and reports the consequences of its actions. CONFIG prints extra periods and commands when it runs; it adds/removes devices by opening a PTY and sending DETACH/ATTACH commands down the PTY. All PTY output is printed by CONFIG, including the TOPS-1Ø prompt (a period) at the conclusion of a command.

Putting Disks On/Off-Line

There are two ways to remove a disk pack from the system: with the REMOVE command in CONFIG or the DETACH monitor command. When using the REMOVE command of the CONFIG program to remove a drive, that drive must be dismounted using OPR; otherwise, an error will result.

```
.R CONFIG
CONFIG>REMOVE RPEØ
? Following Packs MUST be dismounted:
        BLKY on RPEØ
CONFIG> CTRL/C

.R OPR

OPR> DISMOUNT STRUCTURE BLKY:

OPR>
16:Ø1:18        --STRUCTURE BLKY DISMOUNTED--
                FROM UNIT: RPEØ
```

```
OPR> CTRL/C

.R CONFIG

CONFIG>REMOVE RPEØ
DETACH RPEØ

.

CONFIG>
```

To detach a disk using the monitor DETACH command, type:

```
.DETACH RPEØ
```

If the DETACH cannot be allowed, the only error message printed by TOPS-1Ø is:

```
?DETACH RPEØ?
```

The operator must choose the operation carefully. The REMOVE command in CONFIG causes the pack to spin down. If the drive is dual-ported with the console front-end PDP-11, problems could result.

To add a previously removed disk, use the ADD command in CONFIG:

```
.R CONFIG

CONFIG>ADD RPEØ
ATTACH RPEØ

.
CONFIG>
```

or, use the ATTACH monitor command.

```
.ATTACH RPEØ
```

Putting Disk Controllers On/Off-Line

Disk controllers can be removed from the system by using the CONFIG program or the DETACH monitor command. CONFIG removes the controller only if all structures connected to it are dismounted and the connected CPU is running. Remove controllers in the following manner:

```
CONFIG> REMOVE CONTROLLER RPE
? Following Packs MUST be dismounted:
        BLKY on RPE0
CONFIG>CTRL/C

.R OPR

OPR> DISMOUNT BLKY:
OPR>
7:05:23          --STRUCTURE BLKY DISMOUNTED--
                 FROM UNIT: RPE0

OPR> CTRL/C

.R CONFIG

CONFIG>REMOVE CONTROLLER RPE
DETACH RPE:

.
CONFIG>
```

Disk controllers are named according to the following scheme:

```
RPnm:   RP04/06 on RH10/20

DPnm:   RP02/03 on RP10 or RP10C

FSnm:   RS04 on RH10 (not sold anymore)
```

where "n" is the controller number (A-Z) and "m" is the unit number (0-7).

The monitor DETACH command can be used to perform this function also.

```
.DETACH RPE:

.
```

To regain use of a controller for the system, use either the ADD command in CONFIG:

```
CONFIG> ADD CONTROLLER RPE

ATTACH RPE:

.

CONFIG>
```

or the ATTACH monitor command:

```
.ATTACH RPE:

.
```

Taking a CPU Off-Line

To take a CPU off-line in an orderly fashion, four dependent processes must be considered.

1. Check connected disk structures. Only the non-dual-ported drives must be removed from the system.

2. Check jobs running on the CPU.

3. Swap space used on disk drives connected to the CPU.

4. Check devices currently in use that are available only on the CPU.

It is important to handle these conditions carefully so that system operation is not disrupted. This is the main reason why the CONFIG program should be used. CONFIG checks and forces the operator to handle, either directly or indirectly, the conditions that are mentioned above. It can be run at any time. The monitor DETACH command will detach only non-running CPUs.

CONFIG first checks for disk structures that must be dismounted. If a structure has swapping space, the swapping space is migrated to another structure by using OPR. CONFIG then removes disk packs as part of the REMOVE CPU command.

```
.R CONFIG

CONFIG>REMOVE CPUØ
? Following Packs MUST be dismounted:
        DSKAØ on FSAØ
        BLKXØ on RPBØ
CONFIG>CTRL/C

.R OPR

OPR> MODIFY ACTIVE-SWAPPING-LIST EXCLUDE DSKA

OPR>
12:33:Ø1  DEVICE FSAØ -- REMOVED FROM ACTIVE SWAPPING LIST--

OPR> DISMOUNT STRUCTURE DSKA

OPR>
12:34:56  --STRUCTURE DSKA DISMOUNTED--
                 FROM UNIT: FSAØ

OPR> DISMOUNT STRUCTURE BLKX
12:35:ØØ  --STRUCTURE BLKX DISMOUNTED--
          FROM UNIT: RPBØ

OPR> CTRL/C
```

At this point, an attempt to remove CPUØ may produce the following messages:

```
.R CONFIG

CONFIG>REMOVE CPUØ
% Following Jobs will lose devices:
        15 has CDR26Ø
        12 has LPT26Ø
        12 has LPT261
%Detaching Jobs: 2 4 5 14Ø
DETACH FSAØ
```

```
.DETACH RPA3

.DETACH RPA5

.DETACH RPB0

.DETACH RPB1

CONFIG>
```

CONFIG lists the devices that will be lost and the jobs that will be detached as a result of their terminal's removal.

Even if one CPU is down, you should run CONFIG to remove it from the system because of cache benefits and to prevent the operator from receiving a message about the downed CPU once a minute.

The DETACH command at monitor level can remove a CPU; however, all the disks must have been removed first. The use of the DETACH monitor command to remove a CPU is strongly discouraged. The CONFIG program should be used.

Adding a CPU

Returning a CPU to an SMP system is relatively easy compared to removing one. The ADD command in CONFIG or the ATTACH CPUx monitor command can be used. The disks must then be added to the system for the whole system to function normally by one of the following statements:

1. CONFIG> ADD CPU1
2. .ATTACH CPU1

Putting Memory On/Off-Line

The CONFIG program can be used to put memory on- and off-line. It is used as a substitute for the SET MEMORY monitor command. When the operator is reconfiguring memory, the memory switches must be switched manually to memory banks on-line or off-line; 1091 core or MOS systems can set memory off-line without manually throwing switches. To run diagnostics on one CPU, memory must be taken off-line. The CONFIG memory commands are:

1. CONFIG> REMOVE MEMORY FROM xxxxxx TO xxxxxx
2. CONFIG> ADD MEMORY FROM xxxxxx TO xxxxxx

The equivalent monitor commands are:

1. SET MEMORY OFF-LINE FROM xxxxxx TO xxxxxx
2. SET MEMORY ON-LINE FROM xxxxxx TO xxxxxx

Memory can only be removed or added on module boundaries.

Splitting the System

Sometimes it may be desirable to let Field Service troubleshoot the system or run diagnostics without taking the whole system down. This can be done by splitting the system. The operator removes one CPU, the disks connected to that CPU, and part of memory. The diagnostics can then be run on the removed CPU in the removed memory.

To split the system, perform the following steps. First, run CONFIG to remove the CPU. The following example shows how to remove CPU1.

```
.R CONFIG

CONFIG> REMOVE CPU1

.DETACH RPE1

.DETACH RPE2

.DETACH RPE4

.DETACH RPF0

.DETACH RPF4

.DETACH RPF5
```

```
.DETACH RPF7

.

CONFIG>
```

The above example assumes that the disk drives were dismounted first. The next step is to remove memory. Do this only on systems where enough memory remains for one CPU to run comfortably.

```
CONFIG> REMOVE MEMORY 2000000 TO 3777777

SET MEMORY OFF FROM #2000000 TO #3777777

.

CONFIG>
```

You can also say:

```
CONFIG> REMOVE MEMORY 512K TO 1024K
```

Then the following steps must be done at the CTY of the CPU that is being removed:

```
CTRL/Backslash

PAR> RESET

PAR#
```

The RESET command performs a master reset of the CPU, stopping the clocks.

The next step is to deselect the memory banks that belong to the CPU being removed. If you have not split the system before, find the location of each switch from a Field Service engineer. The port switches, memory bank switches, memory base address switches, and memory interleave switches must all be deselected. Make sure that you do not leave any of these switches in the middle position.

To prevent writing on dual-ported disks, the following steps must be performed before you restart the second CPU.

1. Go to the OPR terminal.

2. Make a record of the switch settings.

3. Type ".XCHNGE drivename drivename <return>" where the drivename, which you must type twice, is the name of the drive you want to leave on.

4. Power down the drive you specified in step two.

5. Set the controller select switch on the powered-down drive to A or B, depending on which CPU you are leaving on.

6. Power up the drive.

7. Repeat step two through five for each drive that you want to leave on.

At this point, the split CPU may be started and run as a separate system.

The first step in rejoining the system is to go to the memory cabinet and put all the switches back to their original positions. Next, go to the CTY of the removed CPU and type everything that is underlined in the following.

```
PAR# RESET
PAR# MCR KLINIT

KLI -- VERSION VA12-Ø2 RUNNING
KLI -- ENTER DIALOG [NO,YES,EXIT,BOOT]?
KLI> YES
KLI -- RELOAD MICROCODE [YES,VERIFY,FIX,NO]?
KLI> YES
KLI -- MICROCODE VERSION 231 LOADED
KLI -- RECONFIGURE CACHE [FILE,ALL,YES,NO]?
KLI> ALL
KLI -- ALL CACHES ENABLED
KLI -- CONFIGURE KL MEMORY [FILE,ALL,REVERSE,YES,NO]?
KLI> ALL
```

```
LOGICAL MEMORY CONFIGURATION
  ADDRESS  SIZE  INT  TYPE CONTROLLER
 00000000 1024K   4  DMA20   4

KLI -- LOAD KL BOOTSTRAP [FILE,YES,NO,FILENAME]?
KLI> NO
KLI -- WRITE CONFIGURATION FILE [YES,NO]?
KLI> NO
KLI -- EXIT [YES,RESTART]?
KLI> YES

CTRL/BACKSLASH

PAR% RESET
PAR# JUMP 400          ;This starts the CPU
```

Next, run CONFIG program to inform the monitor that there are two CPUs.

```
.R CONFIG

CONFIG> ADD CPU1

CONFIG> ADD MEMORY 2000000 TO 3777777
SET MEMORY ONLINE FROM #2000000 TO #3777777

.

CONFIG> CTRL/C
```

If the system has joined correctly, the following must be printed on the CTY of the recently joined CPU:

```
[CPU1]
```

RSX20F CHANGES

There are five new PARSER commands that the operator should know about. Also, TOPS-1Ø will now preserve the existing front-end file system when ONCE or TWICE is run.

1. SET RETRY - The SET RETRY command enables the XCT 71 feature. Refer to Chapter 2, KL1Ø Service Enhancement Project, for more information.

   a. PAR> SET RETRY
   b. PAR> SET NO RETRY

2. TAKE COMMAND - This command takes commands from a file and executes them as PARSER commands. All legal PARSER commands are allowed except another TAKE command. The commands in the file are executed until an end-of-file is detected. At this point, the message <EOF> is output and input is then taken from the CTY. The form of the command is:

   PAR> TAKE filespec

   The default file name is PARSER, and the default extension is CMD. TAKE files are created using RSX2ØF PIP (using device TT:) or using TOPS-1Ø and then transferring them to RSX2ØF using RSXT1Ø. In either case, the file resides in the front-end file system.

   Example:

```
PAR> TAKE SETUP.CMD
PAR> !THIS IS AN INDIRECT COMMAND FILE TO SETUP OUTPUT
PAR> !TO THE APPROPRIATE DEVICES
PAR> SET OUTPUT LOG
PAR> SHOW OUTPUT
          OUTPUT DEVICES: LOG
PAR> SHOW RELOAD
 RELOAD ENABLE: OFF
PAR> <EOF>
PAR>
```

3. OUTPUT Command - This command causes CTY output to be directed to various devices. The available devices are the file PARSER.LOG, the CTY and front-end LPTs. The output can be turned on with a SET command and turned off using a CLEAR command. If the log file does not exist, PARSER creates a log file. If, after issuing a CLEAR OUTPUT command no devices are active, then the console terminal is made active. The WHAT or SHOW commands report on active devices.

   The form of the command is:

```
SET OUTPUT dev
CLEAR OUTPUT dev
WHAT OUTPUT
```

   where:

   device may be: LOG, LPT, TTY.

4. SHOW Command - is identical to the WHAT command.

5. WHAT HARDWARE Command - displays the environmental report that KLINIT generates. The report includes the KL serial number, model type, line frequency, and hardware options.

```
PAR> WHAT HARDWARE
 KL1Ø S/N 1Ø26., MODEL B, 6Ø HERTZ
            MOS MASTER OSCILLATOR
            EXTENDED ADDRESSING
            INTERNAL CHANNELS
            CACHE
```

Chapter 13

USER LEVEL CHANGES

KA10 LONG FLOATING-POINT INSTRUCTION SIMULATION

The following six KA1Ø instructions are no longer supported by the KL1Ø microcode:

1. UFA Unnormalized Floating Add
2. DFN Double Floating Negate
3. FADL Floating Add Long
4. FSBL Floating Subtract Long
5. FMPL Floating Multiply Long
6. FDVL Floating Divide Long

The TOPS-1Ø monitor may simulate the execution of these instructions but each instruction takes approximately ten times as long to run as before. If floating-point simulation has been enabled during the MONGEN dialogue, all attempts to execute one of the instructions trap to a new monitor module (KASER) that handles the operation.

To enable KA1Ø floating-point simulation, issue the following command:

```
.SET FLOATING POINT SIMULATION
```

To turn off KA1Ø floating-point simulation, type:

```
.SET FLOATING POINT NO SIMULATION
```

Any program that executes a KA1Ø instruction without this feature enabled receives this error message:

```
?KA1Ø FLOATING POINT INSTRUCTION AT USER PC xxxxxx
```

Floating-point simulation will be supported through Version 7.Ø1. All releases after Version 7.Ø1 will not support this feature.

MERGE Command

The MERGE command uses the MERGE. monitor call to merge the contents of an .EXE file with the current program's core image. The general form of the command is:

```
.MERGE <filename>
```

This command allows versatility in loading specialized page fault handlers or VMDDT.

```
.LOAD WHO.REL
LINK:    LOADING

EXIT

.MERGE MYPFH.EXE
MYPFH Merged
```

If there is overlapping among the pages of the core image and the merged .EXE file, the page in core will NOT be overwritten. Instead, the operation is halted and the following error message prints.

```
.GET SYS:PIP
Job Setup

.MERGE SYS:DDT
?Page overlap error
.
```

VMDDT is no longer a .VMX but an .EXE file and should be loaded using the MERGE command or the MERGE. monitor call. All files with the VMX extension have been removed because of the addition of this feature.

ENABLE/DISABLE Command

The ENABLE command enables user privileges that have been set up in the privilege word in ACCT.SYS. Once a user logs in, all of his/her privileges are automatically turned on. In Version 7.01, the privileges may be turned off using the DISABLE command. The ENABLE command turns on all privileges. Selective powers cannot be enabled/disabled. The privileges that are affected by the ENABLE/DISABLE command are:

1. Job allowed to spy at monitor using PEEK/SPY.

2. Job allowed to spy at all of core using PEEK/SPY.

3. Job can poke the monitor using the POKE. monitor call.

4. Job allowed to use TRPSET monitor call.

5. Job allowed to use LOCK monitor call.

6. Job allowed to use real-time trapping UUO.

7. Job allowed to use other real-time monitor features.

To enable privileges, type:

```
.ENABLE
```

and to disable privileges, type:

```
.DISABLE
```

The [1,2] account uses ENABLE/DISABLE as any other user. However, in many cases the monitor does not check the privilege word and permits the special function simply because the user is [1,2].

MODIFYING TERMINAL CHARACTERISTICS

The command to set terminal characteristics now has four equivalent forms:

1. .TERMINAL <command>
2. .TTY <command>
3. .SET TERMINAL <command>
4. .SET TTY <command>

Three commands have been added to Version 7.Ø1: DEFER, TYPE and DISPLAY. Two commands have been altered (WIDTH and PAGE) for Version 7.Ø1.

The DEFER command suppresses echoing until the user program asks for the characters. Programs doing cursor control or handling their own rubouts (e.g., DDT) can get the screen image correct without doing their own echos. To issue the command, type:

.SET TTY DEFER

To turn this function off, type:

.SET TTY NO DEFER

The default setting is "NO DEFER."

The terminal TYPE command sets up the terminal characteristics for a given terminal type. The general form of the command is:

.TERMINAL TYPE xxxxx

where xxxxx may be one of the following:

DAS21
LA12Ø
LA3Ø
LA34
LA36
LA38
TTY
TTY33
TTY35
VTØ5

VT06
VT100
VT50
VT52
VT61
2741

The default characteristics for these types are:

Type	Width	Page-Size	FF	Tab	LC	Alt
TTY	72	0	N	N	N	Y
VT05	72	20	N	Y	N	N
VT06	72	25	N	N	N	N
VT50	80	12	N	Y	N	N
VT52	80	24	N	Y	Y	N
VT61	80	24	N	Y	Y	N
DAS21	80	24	N	Y	Y	N
VT100	80	24	N	Y	Y	N
TTY33	72	0	N	N	N	Y
TTY35	72	0	Y	Y	N	Y
LA30	72	0	N	N	N	N
LA36	132	0	N	N	Y	N
2741	128	0	N	N	Y	N
LA38	132	0	N	Y	Y	N
LA120	132	0	Y	Y	Y	N

The DISPLAY command tells the monitor that the terminal is a video display terminal (VT05, VT06, VT50, VT52, or VT61). When a terminal is declared as DISPLAY, DELETE will erase the deleted character from the screen.

.SET TERM DISPLAY

The PAGE command has been extended to provide a new feature for video terminals. An optional numerical argument may appear in the command:

.TTY PAGE nn

This sets the screen size for your terminal to n, a positive integer greater than or equal to 0 and less than or equal to 63. Every n lines of uninterrupted type-out generates a CTRL/S automatically. Thus the user will not miss any information as it scrolls by the screen. A bell (CTRL/G) sounds to notify the user when the screen has filled up. CTRL/Q continues printing for another n lines. A page size of 0 (the default value) turns off the automatic CTRL/S feature but leaves regular use of CTRL/S and CTRL/Q enabled.

The maximum page width can now be set to 256 with the "SET TTY WIDTH" command.

With the rewrite of SCNSER, several control characters take on new meanings: DELETE removes the character from the input buffer, backs up one character, and erases the deleted character from the screen (if it is a video terminal). CTRL/W performs the same operation as DELETE except that it acts on words (alphabetic characters bounded by spaces) or a symbol of punctuation. CTRL/R erases and reprints the current line; CTRL/U erases the current line. These functions work much the same as they do on TOPS-20.

RUN Command from SFDS

The RUN, GET, and SAVE commands now support subfile directory names in the file specification. Prior to Version 7.01, only a project and programmer number could be specified within the square brackets. The following two commands now work correctly:

1. .RUN DSKB:EMPIRE[77,4655,GAMES]
2. .GET BLKX:PIP[7,2,PIP,NEW]

SET DEFAULT BUFFERS Command

Users can now specify the default number of disk buffers for their jobs by using a command of the form:

```
.SET DEFAULT BUFFERS n
```

The number n is an octal number between 1 and 777. Until this command is issued, the system default number of buffers is used. That number has been changed from two to six to improve disk throughput. The system default can be modified by declaring the MONGEN symbol M.DFNB to be the desired number of buffers.

SET WATCH FILES Command

A new feature has been added to the WATCH command that causes file lookups, file renames, and file enters to be printed on the user's terminal. The general format of the display is:

```
[fxn: dsk:file.ext[p,pn,sfdl,sfd2,..], ERROR y]
```

where:

F is typed if FILOP. is used
x is either L(Lookup), E (Enter), or R (Rename)
n is the channel number

If there is an error, the octal error code y is printed, which is the Lookup/Enter/Rename error code.

For example, consider editing a file using SOS:

```
.SOS NAMES.DAT
[L7: DSKB1:SWITCH.INI[77,4655]]
[L2: DSKB1:NAMES.DAT[77,4655]]
[L3: DSKB1:068SOS.TMP[77,4655], error 0]
Edit: NAMES.DAT
[E3: DSKB0:068SOS.TEM[77,4655]]
*
```

No lookup was done on SOS.EXE because it was a sharable high segment already in core. Output is also suppressed for monitor Lookup/Enter/Renames, jobs running JACCT, and execute-only programs. To watch the monitor find SOS in this case, the SET WATCH VERSION should be used also.

```
.SOS NAMES.DAT
[S:SOS 21(134) + ]
[L7: DSKB1:SWITCH.INI[77,4655]]
[L2: DSKB1:NAMES.DAT[77,4655]]
[L3: DSKB1:068SOS.TMP[77,4655], error 0]
Edit: NAMES.DAT
[E3: DSKB0:068SOS.TEM[77,4655]]
*
```

USE OF LIB:

The action of LIB: has changed for 7.01.

1. The construct LIBX:, meaning DSKX: for the LIB PPN is no longer valid.

2. LIB: is only searched if the device which was opened is D:, DS:, DSK:, or LIB:. This causes an obvious difference between 6.03A and 7.01. If FOO.BAR exists in the LIB: PPN, .DIRECT FOO.BAR in 6.03A will list the file. In 7.01 this same command does not list the file unless the /NOSTR switch is given to DIRECT. This is because DIRECT opens each STR in the job's search list in order to lookup the file, rather than opening device DSK:.

3. If a file exists only in the library PPN with a protection that makes it inaccessible, 7.01 reports a protection failure; Release 6.03A reported file-not-found.

Chapter 14
NETWORK CHANGES

The three new functions added to TOPS-1Ø networks are: multipathing, bringing the 2Ø2Ø into the network, and adding HASP to IBM communication. The remaining changes were aimed at fixing bugs or improving performance.

MULTIPATHING

NETSER was rewritten to make each KI1Ø or KL1Ø CPU a full node rather than a sequential node. This means that the CPUs can now perform route through, which makes multipathing work. The software in the communications frontends already have the capability to multipath; TOPS-1Ø NETSER limitations had previously prevented this.

Any configuration is now legal except:

1. Connecting a node directly to itself.
2. The DECnet-Compatible Port cannot multipath through RSX-11M because RSX-11M is a sequential node in an ANF-1Ø Network.

When a line or node goes down, the network determines an alternate path automatically, using the fewest number of links to the destination. The speed of the links is not taken into account. Paths are computed whenever a topology change occurs.

Full use of 7.Ø1 multipath capabilities requires installation of M93Ø1WA ROMs in the DN87S frontends on CPUØ and CPU1 for SMP systems. This allows loading the front-end code either across the DTE or across the synchronous line connecting pairs of frontends.

HASP

HASP (Houston Automatic SPooler) is an IBM communications product available with the release of Galaxy Version 4.1. There are now two products in this area; 2780/3780 and 2780/3780/HASP. Customers desiring HASP must purchase a new license even if they already own a DN60 running 2780/3780 Emulation/Termination.

An IBM HASP multileaving station is similar to the 2780, with two important distinctions:

1. It can have an operator's console that allows it to communicate with the host system, to inquire about the status of a batch job, and to control station devices. This feature depends on Galaxy Version 4.1.

2. Messages both to and from devices at the remote station can be interleaved. The interleaving allows messages in both directions to use the same line at the same time. The card punch is not supported. With HASP, a station can read cards and print jobs simultaneously.

An IBM 2780 is a remote terminal with attached card reader and line printer. It submits jobs to a host computer using the card reader and prints the output from the job on the line printer at the remote site.

COMMUNICATIONS ON THE 2020

Support for the 2020 has been added to the network system, allowing the 2020 user to use the SET HOST command. The link is made via the KMC/DUP-11 lines. The 2020 is a full routing node but cannot support sequential nodes such as a DN92. The KDP. monitor call and the KDPLDR program were added to support the KMC/DUP-11 lines and are discussed in greater detail later in this chapter.

NETWORK MONITOR MODULES

SCNSER

Large portions of SCNSER have been rewritten to improve performance, making output faster than before. This improvement is due to four factors:

1. Reorganization of chunk handling
2. Using macros instead of subroutines
3. Reduced scope of SCNSER PI interlocks
4. Grouping output characters

TTY chunks are now handled the same way as in Version 6.02. In Version 6.03A, each line was assigned a chunk quota which prevented the monitor from using the entire chunk space. In Version 7.01, all free chunks are linked together and chunks that are in use are linked together. This shortens the search time for free chunks and the time to put a chunk back in the list. The first free chunk is pointed to by TTFTAK; the last by TTFPUT. To aid in the reconstruction of the events leading to a crash, the left half of each chunk header contains the LDB address of the line that last used it.

When SCNSER must be interlocked against interrupts, only SCNSER's PI level three is turned off. In Version 6.03A and before, ALL interrupts were turned off during interlock in this fashion:

```
CONO    PI,PIOFF
```

In Version 7.01, the following command is used:

```
CONO    PI,SCNOFF
```

This change speeds up the service of other devices, especially real-time devices.

Some of the small routines that are often used are no longer subroutines; instead, macros are used to generate in-line code. The following macros are used:

```
;MACRO TO TAKE A CHARACTER OUT OF A TERMINAL BUFFER
;STREAM, ADVANCE THE POINTER SPECIFIED BY "LOC", BUT
;NOT TO GIVE BACK ANY CHUNKS THAT ARE PASSED OVER. USES
;T1, T2 AND T3.  CHARACTER MAY BE RETURNED IN T3 IF
;DESIRED.

DEFINE LDCHK(AC,LOC),<
        IFNDEF  CK.BDY,<EXTERNAL CK.BDY>
        IFNDEF  NEWCKS,<EXTERNAL NEWCKS>
        SKIPA   T2,[770000,,CK.BDY]
        JRST    .+4
        TDNN    T2,LOC
        PUSHJ   P,NEWCKS
        ILDB    AC,LOC>

;MACRO TO TAKE A CHARACTER OUT OF A CHUNK AND TO RETURN
;ANY CHUNKS THAT ARE PASSED BY.  USES T1, T2 AND T3.
;CHARACTER MAY BE RETURNED IN T3 IF DESIRED. BECAUSE
;LDCHKR RETURNS CHUNKS TO THE FREELIST, IT SHOULD
;ONLY BE CALLED AFTER A CALL TO SCNOFF.

DEFINE LDCHKR(AC,LOC),<
        IFNDEF  CK.BDY,<EXTERNAL CK.BDY>
        IFNDEF  NEWCKI,<EXTERNAL NEWCKI>
        SKIPA   T2,[770000,,CK.BDY]
        JRST    .+4
        TDNN    T2,LOC
        PUSHJ   P,NEWCKI
        ILDB    AC,LOC>

;MACRO TO STORE A CHARACTER IN A TERMINAL OUTPUT STREAM.
;DEPOSITS CHARACTER FROM "AC" USING BYTE POINTER
;SPECIFIED BY "LOC". CHUNKS ARE ALLOCATED FROM THE
;FREELIST AS NEEDED.  USES T1 AND T2. BECAUSE STCHK
;ALLOCATES CHUNKS FROM THE FREELIST, IT SHOULD ONLY BE
;CALLED AFTER A CALL TO SCNOFF HAS BEEN MADE.

DEFINE STCHK(AC,LOC),<
        IFNDEF  CK.BDY,<EXTERNAL CK.BDY>
        IFNDEF  NEWCKO,<EXTERNAL NEWCKO>
        SKIPA   T2,[770000,,CK.BDY]
        JRST    .+4
        TDNN    T2,LOC
        PUSHJ   P,NEWCKO
        IDPB    AC,LOC>
```

The output speedup is largely due to the grouping of output characters. In order to accommodate DC76s and DC1Øs, neither of which is supported, subroutines are provided for character-by-character output. The DC1Ø is no longer supported but seems to work for lightly loaded systems.

For a comparison of character handling and CPU load, observe the graph in Figure 14-1.

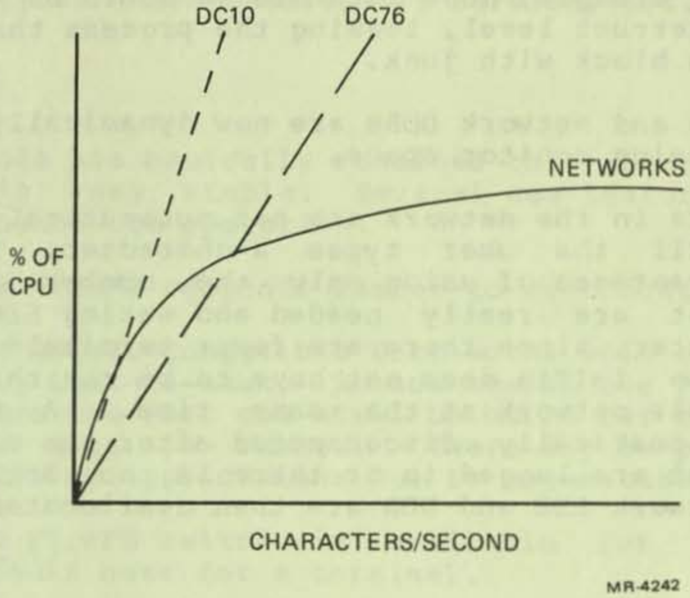


Figure 14-1 CPU Load from SCNSER

In keeping with the strategy of SMP, SCNSER is reentrant and can be run by either processor. Terminal I/O will not block even if the job is not running on the CPU to which the terminal is connected. I/O still blocks if no buffers are available to the job.

RDXSER

This module has been rewritten, but does nothing new. The main focus is on bug fixes, improved performance, and making the code reentrant.

NETSER

As was mentioned earlier, NETSER has been rewritten so that the KL10 is now a full node on the network and not just a sequential node. This allows multipathing to work. Other changes involve the following items:

1. Most of the interrupt code has been eliminated in order to prevent race conditions. In Version 6.03A for example, node data blocks could be removed at interrupt level, leaving the process that was using the block with junk.

2. NDB and network DDBs are now dynamically allocated, freeing monitor space.

3. TTYs in the network are not automatically connected until the user types a character. This has the advantages of using only the number of connects that are really needed and making SEND ALLs work better, since there are fewer terminals to send to. Also INITIA does not have to be run throughout the whole network at the same time. A terminal is automatically disconnected after two minutes if no jobs are logged in or there is no activity. The network LDB and DDB are then deallocated.

TSKSER

TSKSER has been completely rewritten with the result that it works more reliably. However, programs that worked with Version 6.03A may not work with Version 7.01 because the new version of TSKSER checks the PPN field. In Version 6.03A, the PPN field was ignored. Version 6.03A hosts may encounter some problems in message segmentation when talking task-to-task with a Version 7.01 host.

Because of the improvement in PSISER, the completion of TSK I/O can be reliably announced to the operating system and user programs.

TSKSER contains the new TSK. monitor call, which reads or changes the status of a task. For a full description, see the section on monitor calls later in this chapter.

NETDEV

This is a new module which has been created from other network modules. NETDEV contains all the network device service routines: MCR device, TTY service, LPT, and CDR.

DN8x CODE

This code has basically remained the same; it works well and is very stable. Several new features have been added to enhance its operation:

1. XON - XOFF responds faster to user control.
2. The DECnet-Compatible Port works well but is still only task-to-task. It now checks the PPN field. A Phase One Port can still be built for those who do not want to change. There may be problems with existing applications due to segmentation problems.
3. The FT.PFH switch is available for setting the default host for a terminal.
4. TTY parameters should be assembled in the node if possible, especially TTY speed, to avoid race conditions.
5. The "TTY Not Connected" message has been replaced by a series of messages that express the problem behind the failure to connect. The new messages are:

   NO HOST AVAILABLE
   CONNECT SENT
   WAITING FOR CONNECT CONFIRM
   HOST SENT DISCONNECT
   HOST WENT AWAY
   LOCAL INPUT BUFFER FULL
   CHARACTER NOT STORED

RSX20F

In an attempt to reduce the latency inherent in the processing of terminal control characters, the Terminal Input Service has been modified to process each character upon receipt, instead of allowing input to stack up in the DH11 silos.

Other Network Codes

DC72s will work no better with Version 7.Ø1. No changes have been made; there is no room for them. DN92s suffer from a similar space problem but may be slightly better than DC72s because they have the ability to SET HOST. Terminals connected to a DC76 (not supported in 7.Ø1), now have the ability to SET HOST.

NETWORK MONITOR CALL CHANGES

TSK. Monitor Call (CALLI 177)

This new monitor call performs miscellaneous functions for network nodes. It can be used by applications that wish to perform nonblocking connects and disconnects. Also, it can be used by applications translating ANF-1Ø protocol into another protocol. These applications usually require more control over the connect message than that provided by the standard LOOKUP/ENTER sequence.

The TSK. monitor call is an alternative to using the LOOKUP/ENTER method of opening/defining network links. Once the link enters the run state .TKSOK, the normal OUT and IN monitor calls can be used to send or receive data via the network link. The TSK device cannot be designated as an MPX-controlled device, but asynchronous I/O can be performed. The TSK. UUO is called in the following manner:

```
MOVE  ac,[XWD length,addr]
TSK.  ac,
  error return
normal return
```

```
addr:   EXP   function
        EXP   channo
        EXP   arg1
        EXP   arg2
        EXP   arg3
```

where: function is one of the function codes listed below.

channo is the I/O channel number on which the device TSK has been opened. Each arg is an argument for the specified function code.

Most arguments will be pointers to Network Process Descriptors (NPDs), pointed to by a word in the following format:

```
XWD length,addr
```

where: length is the maximum length of the NPD.

addr is the location of the NPD.

Associated with each task link are two processes: the local process and the remote process. The processes are named by the NPD. The format for the NPD is:

```
EXP     node-number (or -1)   ;.TKDNM
EXP     name-length (or -1)   ;.TKDNL
ASCII   /process-name/        ;.TKDPN
```

There are seven functions in the TSK. monitor call:

Code	Function
1	Return the state of a link.
2	Enter the link into the active state.
3	Enter the link into the passive state.
4	Enter the link into the idle state.
5	Put the link into the wait state.
6	Input with notification of DAP type.
7	Output with control of DAP message type.

Function 1 (.TKFRS) - Return the state of a link.

The possible states that the link may be in are:

Code	State	Meaning
0	.TKSID	The link is idle. It destroys the contents of arg2 and stores the reason for the disconnect. Arg3 is unchanged.
1	.TKSCI	The link is waiting for a connect initiate. It returns the local NPD in the area pointed to by arg2. It returns the remote NPD in the area pointed to by arg3.
2	.TKSCC	The link is waiting for a connect confirm. The address of the local NPD in arg2 and the address of the remote NPD are returned.
3	.TKSOK	The link is operational. It returns the address of the local NPD in arg2 and the address of the remote NPD in arg3.
4	.TKSDC	The link is waiting for a disconnect confirm. It returns the address of the local NPD in arg2 and the address of the remote NPD in arg3.

Function 2 (.TKFEP) - Enter the link into the active state.

Before issuing this function, the link must have been in the idle state. All other states cause an error code to be returned. When this code is issued, the monitor reads the local NPD pointed to by arg1 and the remote NPD pointed to by arg2. It then sends a connect initiate request to the node/task specified by arg2. If the link is waiting for a connect confirm, the normal return is taken. The link remains in the .TKSCC state until a connect confirm or disconnect function is issued.

If a connect confirm is issued, the monitor discards the remote NPD pointed to by arg2. It builds a new remote NPD using the information in the connect confirm message, so that it can be read by the read function. The link is placed in the operational state and the controlling job is given a "device on-line" interrupt if the condition was enabled using the Software Interrupt System.

If a disconnect function is issued, the monitor discards both the local and remote NPD specifications. It places the link in the idle state and gives the controlling job a "device off-line" interrupt, if the job enabled this condition using the Software Interrupt System.

Function 3 (.TKFEA) - Enter the link into the passive state.

The link must be in the idle state. The monitor reads and stores the local and remote NPDs in arg2 and arg3 and places the link in the .TKSCI state.

If at a later time, the monitor receives a connect initiate that "matches" the remote NPD, the following occurs:

a. The monitor deletes the remote NPD.

b. The monitor builds a new remote NPD from the information given in the connect message. The job may read the new NPD by using the .TKFRS function to determine which process initiated the connection.

c. The monitor enters the link into the .TKSOK state.

d. The monitor issues a "device on-line" interrupt to the job if the job enabled this condition using the Software Interrupt System.

Function 4 (.TKFEI) - Place the link in the idle state.

This function is illegal for those tasks in the .TKSDC or .TKSCC states and is a no-op for those already in the idle state. The monitor performs the following for those links in .TKSCI and .TKSOK states:

.TKSCI Both NPDs are released. The link state is set to .TKSID.

.TKSOK A Disconnect Initiate is sent. The link state is set to .TKSDC.

When a disconnect confirm is issued at a later time, the monitor frees both NPDs, sets the link state to .TKSID, and issues a "device off-line" interrupt.

Function 5 (.TKFWT) - Put the link into the wait state.

If the link is in either the .TKSID or .TKSOK state, the monitor takes the normal return immediately. The monitor performs the following for those links in the other states:

.TKSCI Waits for a transition to the .TKSOK state and then returns.

.TKSCC Waits for a transition to either the .TKSOK or .TKSID states, then returns.

.TKSDC Waits for a transition to .TKSID and then returns.

Function 6 (.TKFOT) - Input with notification of DAP type.

This function performs an IN monitor call on the specified channel and is valid only for operational links. If the IN is successful (nonskip return) the monitor stores the DAP type in arg1 and gives a skip return. If the IN fails, the monitor places error code TKDDW% in the AC and stores the device status word in arg1.

Function 7 (.TKFIN) - Output with control of DAP message type.

This function performs an OUT monitor call on links in the operational state on the specified channel. If the OUT is successful, the contents of the buffer will be sent as a DAP message of the type specified. If unsuccessful, the monitor places error code TKDDW% and the device status word in the AC.

If an error occurs during the execution of the TSK. UUO, one of the following error codes will be returned in the AC:

```
TKTNL%==1  ;TSKSER NOT LOADED (TASK-TO-TASK NOT SUPPORTED)

TKATS%==2  ;ARGUMENT LIST WAS TOO SHORT

TKUNP%==3  ;USER IS NOT PRIVILEGED TO PERFORM THIS FUNCTION

TKILF%==4  ;ILLEGAL FUNCTION

TKILC%==5  ;ILLEGAL CHANNEL (NOT A TSK DEVICE OR NOT OPEN)

TKILN%==6  ;ILLEGAL NPD

TKNTS%==7  ;NPD TOO SHORT

TKILS%==10 ;FUNCTION IS ILLEGAL IN THIS STATE

TKNFC%==11 ;NOT ENOUGH MONITOR FREE-CORE TO PERFORM THIS
            FUNCTION

TKNFL%==12 ;NO FREE LINKS.  (NETLAT IS FULL)

TKNXN%==13 ;ATTEMPT TO CONNECT TO A NON-EXISTENT NODE

TKDDW%==14 ;IN OR OUT UUO (.TKFOT OR .TKFIN) DIDN'T SKIP
```

NODE. Monitor Call (CALLI 157)

Five new subfunctions have been added to the NODE. monitor call. They are:

1Ø	Terminal connect
11	Terminal disconnect
12	List the nodes in the network
13	Return information from a node data block
14	Set/read the node greeted flag

The reasons for these new features are:

1. To allow user programs to examine the network data base in a monitor independent way (e.g., SYSDPY/SYSV52 and NETWOR).

2. To make it easier for user programs to determine what network resources are available (e.g., GALAXY).

3. To provide the basis for a better mechanism for the control of remote terminals.

Function 1Ø (.NDTCN) - Terminal connect.

If the terminal is connected to TOPS-1Ø, the name of the terminal is returned; if it is not connected, an attempt is made to connect the terminal by sending it a connect message. Upon a successful return the AC will contain the SIXBIT /TTYnnn/ indicating the line number that the terminal was connected to.

```
        MOVE    ac,[XWD .NDTCN,argblk]
        NODE.   ac,
          error return
        normal return

argblk: EXP     2                     ;LENGTH OF ARG BLOCK.
        XWD     node,line             ;NODE AND LINE NUMBER
                                      ;OF THE DESIRED TERMINAL.
```

If an error code occurs, the error code is returned in the AC. The possible errors are:

NDIAL%==1	Illegal argument list, too long or too short.
NDTNA%==2	Terminal not available. Either the specified node was not up, the terminal was in use by another system, the terminal did not exist, or the use of the terminal was "restricted".

Function 11 (.NDTDS) - Terminal disconnect.

This function will disconnect terminals that are not in use by the system and not connected to the job executing the monitor call. If the third argument is not specified, a simple disconnect (Reason = RSN.OK) will be sent. If the third argument is specified, a Disconnect/Reconnect (Reason = RSN.RC) message will be sent with the third argument being the number of the node to reconnect to.

```
        MOVE    ac,[XWD .NDTDS,argblk]
        NODE.   ac,
          error return
        normal return

argblk: EXP     length              ;ARG BLOCK SIZE (2 OR 3)
        EXP     SIXBIT/ TTYnnn/     ;NAME OF TTY TO DISCONNECT.
        EXP     node-number         ;OPTIONAL - NUMBER OF NODE
                                    ;TO RECONNECT TO.
```

Possible Errors:

NDIAL%==1	An illegal arg list; the length was wrong.
NDIAJ%==3	The terminal was in use by another job, i.e., user did not have right to disconnect.
NDNLT%==4	Not a legal terminal; the second arg was not a terminal presently attached.

Function 12 (.NDLND) - List the nodes in the network.

Upon a successful return, the AC contains the number of nodes currently in the network. ARGBLK+1 through ARGBLK+N contain the numbers of the nodes in the network.

NOTE

> No error is given if the return value area is too short to hold data for all the nodes. The block is filled but is not allowed to overflow. If only the number of nodes in the network is wanted, the arg-block length can be one.

```
        MOVE    ac,[XWD .NDLND,argblk]
        NODE.   ac,
          error return
        normal return

argblk: EXP     n+1              ;LENGTH OF THE ARG BLOCK. N >= Ø
        BLOCK   n                ;RETURNED NODE NUMBERS.
```

Function 13 (.NDNDB) - Return information from a node data block.

The requested field is returned in the return-value area of the argument block.

```
        MOVE    ac,[XWD .NDNDB,argblk]
        NODE.   ac,
          error return
        normal return

argblk: EXP     n+3              ;ARG BLOCK LENGTH
        EXP     node-number      ;NUMBER OF EXAMINED NODE
        EXP     field-number     ;DESCRIPTOR OF THE FIELD
                                 ;TO RETURN
        BLOCK   n                ;BLOCK OF N WORDS TO HOLD
                                 ;RETURNED INFORMATION.
```

The valid field numbers for the third word are:

```
ND.NNM==1                 ;THE NUMBER OF THE SPECIFIED NODE
ND.SNM==2                 ;THE NAME IN 6 OR LESS SIXBIT CHARS
ND.SID==3                 ;THE SOFTWARE ID IN "ASCIZ"
ND.DAT==4                 ;THE SOFTWARE DATE IN "ASCIZ"
ND.LMA==5                 ;THE NCL LAST MESSAGE ASSIGNED
ND.LMS==6                 ;THE NCL LAST MESSAGE SENT
ND.LAR==7                 ;THE NCL LAST ACK RECEIVED
ND.LAP==10                ;THE NCL LAST ACK PROCESSED
ND.LMR==11                ;THE NCL LAST MESSAGE RECEIVED
ND.LMP==12                ;THE NCL LAST MESSAGE PROCESSED
ND.LAS==13                ;THE NCL LAST ACK SENT
ND.MOM==15                ;THE MAXIMUM OUTSTANDING
                          ;MESSAGE COUNTER
ND.TOP==16                ;THE TOPOLOGY TABLE-THIS IS UP TO 16
                          ;WORDS OF THE FORM "XWD LEVEL,NODE"
ND.CNF==17                ;THE CONFIGURATION TABLE.  THIS IS
                          ;VARIABLE LENGTH OF THE FORM
                          ;"XWD OBJ,NUMBER"
```

The possible errors returned in the AC are:

```
NDIAL%==1                 ;ILLEGAL ARE LIST (TOO SHORT)
NDINN%==2                 ;INVALID NODE NUMBER.
                          ;(NODE DID NOT EXIST)
NDISF%==3                 ;ILLEGAL SUB FUNCTION.
                          ;(FUNCTION OUT OF RANGE.)
```

Function 14 (.NDGNF) - Set/read the node greeted flag.

This function can be used by privileged programs that wish to perform a node-specific function to a node when it comes on-line. The calling sequence is:

```
        MOVE    ac,[XWD 14,argblk]
        NODE.   ac,
          error return
        normal return

argblk: EXP     2
        argument
```

where: The argument is Ø or a node number. If Ø, the node number of the first "ungreeted" node is returned in the argument. If a node number, the "greeted" flag for that node is set.

KDP. Monitor Call (CALLI 200)

The KDP. monitor call allows a programmer on a DECSYSTEM-2Ø2Ø running TOPS-1Ø to control KMC/DUP-11 communications lines. With it, the microcode on the KMC-11 can be loaded, verified, or stopped; DDCMP on a DUP-11 line can be started or stopped; and status information can be returned. The calling sequence is:

```
        MOVE    ac,[XWD length,addr]
        KDP.    ac,
          error return
        normal return

addr:   EXP     function
        arg #1
        .
        .
        arg #N
```

where: length is the number of words in the argument block
addr is the address of the argument block
function is one of the functions below

Function 1 - returns the number of KMC-11s in the system. It is called with 1 in addr: and the number is returned in addr+1.

Function 2 - returns the number of DUP-11s in the system. It is called in the same manner as function 1.

Function 3 - returns the status of a KMC-11. Addr holds the function (3), addr+1 holds the KMC-11 number, and the status is returned in addr+2. The status bits are defined as follows:

Bit Ø - is 1 is the KMC-11 is running

Function 4 - halts the KMC-11 as specified by addr+1. This function does not declare DUP-11 lines down since setting the run bit would allow the KMC-11 to continue. Function 4 is a privileged command.

Function 5 - master-clears the KMC-11 specified by addr+1. This function is also a privileged function, and does not allow the KMC-11 to be continued as does function 4.

Function 6 - starts the KMC-11 as specified by addr+1. Specifically, function 6 starts the KMC-11, sets up the interrupt vectors, enables KMC-11 interrupts, and declares the KMC-11 to be running. The KMC-11 cannot have been running previously. This is a privileged function.

Function 7 - reads a CRAM location in the KMC-11. Addr+1 holds the KMC-11 number, addr+2 holds the CRAM address, and addr+3 holds the contents of the CRAM location. The KMC-11 must be halted to perform this function.

Function 1Ø - writes a CRAM location in the KMC-11. Function 1Ø acts in the same manner as function 7 and uses the same argument block, except that addr+3 holds the value to be written into the CRAM location.

Function 1Ø1 - reads the line status of a DUP-11 line. The status bits are:

Bit Ø Need to send a maintenance message

Bit 1 Need to send a STRT message

Bit 2 Need to send a STACK message

Bit 3 Need to send a NAK message

Bit 4 Need to send a REP message

Bit 5 Need to send a data message

Bit 6 Need to send an ACK message

Bits 15-17 are interpreted as follows:

Ø - Control-out to start not yet given

1 - (Base!Control) out has been done

2 - Shutting line down, buffers flushing

3 - Maintenance mode

4 - Sending STARTs state

5 - Sending STACK state

6 - Running

Bit 18 The KMC-11 should give the next receive buffer

Bit 19 The KMC-11 should return the next XMIT buffer

Bit 2Ø Receive flush in progress

Bit 21 XMIT flush in progress

Function 102 - stops DDCMP on a line. This privileged command has the KMC-11 number in addr+1 and the DUP-11 number in addr+2. Executing this function automatically flushes the transfers in progress.

Function 103 - starts DDCMP on a line. This is also a privileged command with the same argument block as function 102. This command fails if DDCMP is already running on the specified DUP-11 line.

NETWORK MONGEN

The MONGEN dialogue for networks has been simplified so that:

1. The only devices that can be connected are DN87Ss and DN87s. On KL10 SMP systems, they can only be connected to CPU0.
2. The number of nodes in the network is no longer requested.
3. The number of connects includes TTYs, CDRs, LPTs, and all other network devices.
4. The answer to the "number of TTYs" question can be smaller than before because TOPS-10 no longer tries to connect to all TTYs in the network when TOPS-10 comes up.

The first step in defining the network is setting FTNET=-1 in FGEN, which permits networks in the system. The network configuration section in MONGEN must then be answered. The following sample dialogue lists all the network questions appearing in a system configured with:

```
CPU0    - two DN87s (DL10)
          one DN87S (DTE)

CPU1    - one DN87S (DTE)
```

NOTE

In the following dialogue, user reponses are underlined.

```
.R MONGEN
MONGEN FOR 7Ø1 MONITORs
/HELP(PROMPT,SHORT,LONG)]: LONG

WHICH GEN(HDW,TTY,NET,F)[
HDW TO DEFINE HARDWARE CONFIGURATION
TTY TO DEFINE TERMINAL CONFIGURATION
NET TO DEFINE NETWORK CONFIGURATION
F TO DEFINE SOFTWARE FEATURES]: NET

OUTPUT(DSK:NETCNF.MAC): <CR>

NETWORK SOFTWARE(Y,N)[
SOFTWARE TO SUPPORT REMOTE COMPUTERS: DECSYSTEM-1Ø'S
PDP-11'S, PDP-8'S (REQUIRES FTNET TO BE -1)]: Y

CPU'S(1,1-4)[TOTAL NUMBER OF CPU'S IN THE SYSTEM]: 2
HOW MANY DC75NP'S OR DN87'S ON CPUØ(1,Ø-8)[NETWORK
FRONT-ENDS CONNECTED TO DL1Ø'S.]: 2

FOR FRONT END NUMBER 1:
TO WHICH DL1Ø PORT IS THE DC75 OR DN87 CONNECTED (Ø,Ø-7)[]:
Ø
FOR FRONT END NUMBER 2:
TO WHICH DL1Ø PORT IS THE DC75 OR DN87 CONNECTED (Ø,Ø-7)[]:
1

HOW MANY DN87S'S ON CPUØ(1,Ø-3)[NETWORK
FRONT-ENDS CONNECTED TO DTE-2Ø'S]: 1

FOR FRONT END NUMBER 3:
TO WHICH DTE2Ø IS THE DN87S CONNECTED (1,1-3)[]: 1

HOW MANY DN87S'S ON CPU1(1,Ø-3)[NETWORK
FRONT-ENDS CONNECTED TO DTE-2Ø'S]: 1
FOR FRONT END NUMBER 4:
TO WHICH DTE2Ø IS THE DN87S CONNECTED (1,1-3)[]: 1
```

```
NODE NUMBER OF CENTRAL SITE(1,1-77)[

UNIQUE OCTAL NUMBER IDENTIFYING SYSTEM TO NETWORK.]: 11

NAME OF CENTRAL SITE[SIX CHARACTERS OR LESS.]: SAMPLE

# OF REMOTE TTY'S(1,Ø-512)[
MAXIMUM NUMBER OF TELETYPES ON NETWORK NODES TO BE
HANDLED AT ANY GIVEN TIME.]: 4Ø

NETWORK VIRTUAL TERMINALS(Y,N)[
CODE TO ALLOW LOCAL TERMINALS TO "SET HOST" TO OTHER
SYSTEMS.]: Y

REMOTE CARD READERS(Y,N)[
CODE TO ALLOW ACCESS TO CARD READERS ON REMOTE STATIONS.]: Y

REMOTE LINE PRINTERS(Y,N)[
CODE TO ALLOW ACCESS TO LINE PRINTERS ON REMOTE STATIONS.]:
Y

REMOTE DATA ENTRY TERMINALS(Y,N)[
CODE TO ALLOW ACCESS TO REMOTE DATA ENTRY TERMINALS (RDX
DEVICES).]: Y

REMOTE TASK-TO-TASK(Y,N)[
CODE TO ALLOW ACCESS TO REMOTE JOBS (TSKSER).]: Y

# OF CONNECTS(256,1-512)[
MAXIMUM NUMBER OF SIMULTANEOUS CONNECTIONS.]: 1ØØ

FILE DSK:NETCNF.MAC CLOSED [NETGEN FINISHED]
```

The following questions have been removed because the devices they pertain to no longer are supported for networks:

```
REMOTE PAPER TAPE READERS(Y,N):

REMOTE PAPER TAPE PUNCHES(Y,N):

REMOTE MAGNETIC TAPE DRIVES(Y,N):

HOW MANY NODES DO YOU WISH TO SUPPORT( ):
```

NETWORK PROGRAMS

NETWORK

The NETWORK program lists data about the nodes in a network in the same format as the NODE command does, and allows the user greater flexibility in listing the nodes and devices in the network. The NETWORK program should be used as a replacement to the NODE command. The NODE command no longer lists all the nodes in the network, only the node the terminal is connected to. NETWORK is run by typing either

1. .NETWORK <NODE-LIST>

2. .R NETWORK<CR>
   *<node-list>

A <NODE-LIST> is a string of node names or numbers separated by commas and optionally followed by switches. The <NODE-LIST> may consist only of switches, in which case all nodes are selected. When the program is started by typing "NETWORK" followed by a carriage return, all nodes are also selected. This program will accept wildcard characters in node names but not node numbers. Indirect command files are also permitted. The command line may take the format specified below.

NODE-NAME,NODE-NUMBER,.../SWITCH/SWITCH...

Node names can be repeated, but switches cannot. The output switches are:

1. /(NO)BRIEF Suppress the output of devices.

2. /(NO)ERROR Output all error messages (default).

3. /(NO)FAST Output only the NODE name and number of all the nodes. If /NOBRIEF is specified then the configuration information is also output.

4. /(NO)HELP Output this text.

5. /(NO)SILENC Suppress all non-error output.

6. /SORT Sort nodes by node number.

7. /TOPOLOGY Display the topology of the network.

The node selection switches are:

1. /(NO)CDP Only output nodes with one or more "CDP".
2. /(NO)CDR Only output nodes with one or more "CDR".
3. /(NO)DTA Only output nodes with one or more "DTA".
4. /(NO)LPT Only output nodes with one or more "LPT".
5. /(NO)MCR Only output nodes with one or more "MCR".
6. /(NO)MTA Only output nodes with one or more "MTA".
7. /(NO)PLT Only output nodes with one or more "PLT".
8. /(NO)PTP Only output nodes with one or more "PTP".
9. /(NO)PTR Only output nodes with one or more "PTR".
1Ø. /(NO)RDA Only output nodes with one or more "RDA".
11. /(NO)TSK Only output nodes with one or more "TSK".
12. /(NO)TTY Only output nodes with one or more "TTY".

Switches used for selection of nodes can be used to select a node either with or without a specific attribute. This is done by using the proper switch or prefixing the switch with "NO". If the switch is not used, the selection of a node does not depend upon that attribute.

If no node is specified (e.g., NETWORK <carriage-return>), the default output is /BRIEF. Any node specification at all causes the default to print the configuration information.

The default NETWORK display is similar to that of the NODE command. The detailed display gives node and device information as shown by the following example.

```
.NETWORK

NODE    KL1Ø26  (26)    RZ371A KL #1Ø26/1Ø42   Ø8-16-79
        CDP[1] TSK[1] DTA[8] MTA[15] PLT[1] PTP[2] PTR[2]
        TTY[137] MCR[1]
NODE    COMET   (7Ø)    DN2ØØ V21(121)  17-AUGUST-79
        TSK[2] TTY[17]
NODE    ENCORE  (32)    DN87S V21(121)  17-AUGUST-79
        TSK[2] TTY[65]
NODE    DN87Ø4  (4)     DN87S V21(121)  14-AUGUST-79
        TSK[2] TTY[65]
```

```
NODE    KS4101  (76)    RS371A KS #4101 08-17-79
        TSK[1] MTA[2] LPT[1] CDR[1] TTY[33] MCR[1]
NODE    KL1279  (10)    RW366A KL10 SYS#1279   08-01-79
        TSK[1] DTA[2] MTA[2] PTP[1] PTR[1] LPT[1] TTY[10]
        MCR[1]
NODE    DS401B  (2)     DN87 V11        29-MAY-79
        TTY[16]
NODE    CTCH22  (22)    DN82 V21(121)   17-AUGUST-79
        TSK[2] LPT[1] CDR[1] TTY[33]
NODE    NEXT    (27)    DN87 V21(121)   17-AUGUST-79
        TTY[1]
NODE    NOVA    (31)    DN87S V21(117)  18-JULY-79
        TTY[47]
```

The topology display looks like this:

```
.NETWORK/TOPOLOGY/SORT

NODE    2102    (20)    NONE
NODE    CTCH22  (22)    27(8)   76(8)
NODE    KL1026  (26)    32(10)  31(10)  27(10)
NODE    NEXT    (27)    26(64)  22(16)  75(16)
NODE    NOVA    (31)    26(64)  32(8)   20(8)
NODE    ENCORE  (32)    26(64)  31(8)
NODE    COMET   (70)    76(8)
NODE    TWINKY  (71)    77(10)  75(10)
NODE    WIZARD  (75)    71(8)   27(8)
NODE    KS4101  (76)    22(10)  70(10)
NODE    SOFDCP  (77)    71(8)
```

SYSDPY

The SYSDPY program has two commands for printing network information: "T" and "\".

1. The "T" command displays the current topology of the network and information about the most recent messages. If a node goes off-line, the display is updated. A user must be privileged to run this display. The following is a sample of the SYSDPY "T" display.

```
THIS IS RW340A KL10 SYS#1279 01-MAR-79 04:13:41 UP:29:33:39
 NODE        NEIGHBORS      OPR  CTL  LAR  LAP  LMS  LMA  LMR  LMP
KL1279(10)    3,  2,         72       0    0    0    0    0    0
KL1026(26)   27,                      15   15   15   15   15   15
CYNIC(66)    76,                      3    3    3    3    3    3
KS4101(76)   66, 27,                  11   11   15   15   245  245
CTCH22(22)   27,                      3    3    3    3    3    3
NEXT(27)     26,22,2,76,              3    3    3    3    4    4
RSX45(53)                             2    2    2    2    7    7
DS401B(2)    10, 27,                  101  101  101  101  211  211
DN87CP(3)    10, 53,                  110  110  110  110  180  180
```

The first line is the standard SYSDPY "ID" line, containing the name of the system preceded by the header "This is", the current date and time, and the system uptime.

Then the nodes currently up in the ANF-10 network are listed with the following information:

Node	The name of the node, and the node number in parenthesis.
Neighbors	The numbers of that node's immediate neighbors. If the node is listed without any neighbors, it is a "sequential" node such as a DN92 or an RSX-11M/DECnet node linked to the ANF-10 network through a DECnet compatible port.
OPR	The TTY line number of the node's OPR terminal.
CTL	The job number doing station control.
LAR	The last ACK received (NCL message number).
LAP	The last ACK processed (NCL message number).
LMS	The last message sent (NCL message number).
LMA	The last message ACKed (NCL message number).
LMR	The last message received (NCL message number).
LMP	The last message processed (NCL message number).

2. The "\" command displays statistics about messages transmitted and received. It can only be run by a user with privileges. The following is an example of the display.

```
THIS IS RW340A KL10 SYS#1279 Ø1-MAR-79 Ø4:13:53 UP:ØØ:33:51
NTCOR= 35ØØ    NTMAX=  4176    NTBAD=   5
UNNUMBERED CTL  XMIT'ED  RECV'ED          NUMBERED CTL    XMIT'ED  RECV'ED
Ø DAP/DATA      17637    14628            1 CONNECT       38       26
1 ACK           77Ø7     5417             2 DISCONNECT    5        17
2 NAK           5        Ø                3 NEIGHBORS     8        8
3 REP           Ø        5                4 REQ CONFIG    8        7
4 START         6        3                5 CONFIG        7        14
5 STACK         3        5                6 DATA REQUEST  8        8425
6 NODE ID       2        2                7 STATION CTL   Ø        Ø
XMIT'ED=2536Ø AVERAGE=12.48/SEC             RECV'ED=2ØØ6Ø AVERAGE=9.87/SEC
 2**N  Ø%  2Ø%  4Ø%  6Ø%  8Ø%  99%         2**N  Ø%  2Ø%  4Ø%  6Ø%  8Ø%  99%
1 ØØ%                                     1 ØØ%
2 Ø1%  *                                  2 25%  *******
3 34%  *********                          3 68%  *****************
4 Ø5%  **                                 4 Ø4%  *
5 Ø1%  *                                  5 Ø1%  *
6 Ø2%  *                                  6 ØØ%
7 25%  *******                            7 ØØ%
8 ØØ%                                     8 ØØ%
9 29%  ********                           9 ØØ%
```

The first line is the SYSDPY ID line. The second line displays the following general network values:

NTCOR	Present monitor free core in use by NETSER
NTMAX	Maximum value of NTCOR
NTBAD	Number of bad network messages received

The next portion of the display is devoted to the total number of network messages received and transmitted, broken down by message type. Note that the "Numbered CTL" messages are also counted in the "Unnumbered CTL" DAP/DATA messages. They are the DAP messages. The total number of data messages can be found by subtracting the total numbered CTL messages from the DAP/DATA unnumbered CTL messages.

The next line summarizes total messages received and transmitted.

The last portion of the screen is a histogram of the data messages received and transmitted, broken down as a function, log base two, of the data message size.

This display does not scroll, and on terminals with less than 2Ø (decimal) lines on the screen (e.g., VT5Øs), the histogram is not displayed.

DTELDR

There have been two main changes to DTELDR. The first is improved error reporting to DAEMON and AVAIL.SYS. All dumps and loads are now logged via error codes 2Ø2 and 2Ø3. The second change is improved intelligence of the program when it is in /AUTO mode. DTELDR now reads the front-end control block and checks the reload bit itself.

NETLDR

The following changes have been made made to NETLDR:

1. NETLDR does not have to be locked in core to run. Instead, the station control messages are copied into NETLDR's program space from the station control blocks when needed.

2. NETLDR can now handle KMC-11s and DUP-11s connected to a PDP-11/4Ø.

3. INITIA does not run when the remote station comes up, rather, only when the line connects to the host.

4. Error logging is done in the same method as DTELDR.

KDPLDR

KDPLDR is a program that controls the KMC-11/DUP-11 synchronous lines attached to a KS10. Its functions are:

1. To start, stop, and clear KMC-11 line controllers.

2. To load and verify the KMC-11 microcode.

3. To initialize or halt DDCMP on the DUP-11 synchronous lines controlled by the KMC-11.

The functions of this program are performed by using the KDP. monitor call, new with 7.01.

Switches to KDPLDR fall into three categories: those for the KMC-11, those for the DUP-11, and the general startup switch.

KMC-11 SWITCHES

/MCLEAR:arg — This switch will cause KDPLDR to master clear the KMC-11 controller specified by arg. The argument may be either a KMC-11 controller number or the word "ALL" specifying all KMC-11s.

```
KDL> /MCLEAR:1
   or
KDL> /MCLEAR:ALL
```

/LOAD:arg — This switch loads the control RAMs of the specified KMC-11 controllers. KDPLDR loads its own microcode unless used with the /BFILE switch. "Arg" may be a KMC-11 number or the word "ALL".

```
KDL> /LOAD:2
```

/VERIFY:arg — This switch compares the microcode of the KMC-11 with either KDPLDR's internal microcode or the microcode specified by the /BFILE switch. The argument may take the same forms as for the previous two switches.

```
KDL> /VERIFY:ALL
```

/BFILE:file — This switch is used to modify the /LOAD or the /VERIFY switch by specifying a microcode file to be used instead of KDPLDR's internal microcode.

```
KDL> /LOAD:1/BFILE:NEWMIC.TST

KDL> /VERFIY:ALL/BFILE:NEW:UCODE.VØ2[1Ø,7]
```

/USTART:arg — This switch starts the microcode of the specified KMC-11s. The argument is either a KMC-11 controller number or the word "ALL", specifying all KMC-11s.

```
KDL> /USTART:ALL
```

DUP-11 SWITCHES

/START:arg — This switch initiates DDCMP on the lines specified by "arg". The argument is either a DUP-11 line number or the word "ALL". This switch must be modified by the /KMC switch to specify the line's controller.

```
KDL> /START:ALL/KMC:1
```

/STOP:arg — This switch stops DDCMP on the lines specified by "arg", and must be accompanied by the /KMC switch to specify the KMC-11 that the DUP-11 lines are connected to.

```
KDL> /STOP:ALL/KMC:ALL
```

/KMC:arg — This switch is the required modifier for the /START and the /STOP switches, and takes as an argument either a KMC-11 controller number of the word "ALL".

GENERAL STARTUP - THE /AUTO SWITCH

Usually, the /AUTO switch is the only switch that needs to be given to KDPLDR. It is equivalent to the following sequence of commands:

```
/MCLEAR:ALL
/LOAD:ALL
/VERIFY:ALL
/USTART:ALL
/START:ALL/KMC:ALL
```

BOOT11

BOOT11 Version 4A is a maintenance release that corrects deficiencies in Version 4. It has been improved to function correctly in the following areas:

1. Dual CPU
2. Timing for PDP-11/4Øs
3. DL1Ø 8K option
4. Dump and reload logging, by using DAEMON monitor calls

BOOT11 also requires a port switch when more than one PDP-11 exists on the DL1Ø. The format of the switch is:

```
/PORTNO:n
```

where n is the port number in the range zero to three.

If BOOT11 is not given a port switch when one is required, e.g., PDP-11s exist on ports Ø, 1, and 2, it responds with:

```
PDP-11'S EXIST ON PORTS:
Ø 1 2
? PORT SWITCH REQUIRED
FILE:
```

At this point, the user should re-enter the command string and include the appropriate port number.

Appendix A

DDT NEW FEATURES

This appendix is designed as a user's guide to features of DDT that represent changes from previous versions. This is not a complete user's guide to all the wonders of DDT -- just those new features that have recently been implemented. Although directed primarily at new features only in DDT Version 41, some documentation is included to describe other aspects of DDT which have been around for a longer time, but were never fully understood or documented.

Throughout this appendix it is assumed that the reader is already familiar with DDT and the MACRO assembly language in general, as well as the appropriate operating system(s).

DDT Version 41 will run on KA1Øs, KI1Øs, KL1Øs, and KS1Øs, using either no paging, KI-paging, or KL-paging, with or without extended addressing in user or executive mode. User and file DDTs run only in user mode, with no special assembly needed. DDT Version 41 must be assembled to run under either the TOPS-1Ø or the TOPS-2Ø operating system. Traditionally, it has been a goal to maintain one single set of source files from which all flavors of DDT are built. This goal has been maintained.

MEMORY AND ADDRESS CONTROL

The single biggest change to DDT Version 41 from earlier versions involves memory control and how the user addresses memory locations.

Extended Addressing

All flavors of DDT, except FILDDT, will run in any memory section. Full extended addressing is supported, as are "large" addresses; DDT will now accept a full 36-bit expression as an address although obviously only FILDDT can actually handle an address over 30 bits wide. In all cases the actual address must be positive, i.e., effectively a 35-bit address.

SYMBOL TABLE RESTRICTIONS

There are certain restrictions which must be adhered to in order for DDT to function correctly. The first restriction is that the symbol table logic is essentially section-dependent, i.e., the symbol table and its pointers (.JBSYM=116 and .JBUSY=117, also .JBHSM=6 relative to the start of the "high segment") must be mapped in the same section as that in which DDT itself is running. Further, the symbol table can be no longer than 128K words and must be in RADIX-50 format.

Much thought is being given towards the implementation of a totally new symbol table scheme which would address all of these problems. The single biggest problem is how extended addressing going to be used: (1) as a single fixed address space with one or more "global" symbol tables as the TOPS-20 monitor currently works, or (2) as a collection of independent sections, each of which has section-local symbols/symbol tables.

BREAKPOINT RESTRICTIONS

The second restriction of which the user must be aware concerns breakpoints. The hardware has no facility to unconditionally transfer control to DDT using only in 36-bits. Therefore, DDT must be mapped, at the same relative address, into each section which the user wishes to place breakpoints.

Location Examining Restrictions

If DDT is running in section 0, only locations within section 0 (addresses 0 to 777777) may be manipulated, even if running on an extended addressing machine. DDT will make no effort to outsmart the combined efforts of the user and the operating system by sneaking into a non-zero section even momentarily to do the memory reference.

Effective Address Calculation

DDT Version 41 can calculate effective address references using either "local", IFIW (Instruction Format Indirect Word), "global" or EFIW (Extended Format Indirect Word) formats. In a normal DDT address-opening command ("/", "\", <TAB>, etc.) a single <ESC> delimiting the address expression (e.g., "MOVE 3,@200(10)$/" or just "$[") instructs DDT to treat the expression as an IFIW word and calculate the effective address exactly as the hardware would, were the hardware to execute that 36-bit word as an instruction at location "." (whether or not location "." is currently open).

Two <ESC>s delimiting the address expression instructs DDT to treat the 36-bit expression as an EFIW word and to calculate the effective address exactly as the hardware would, were the hardware to indirectly address the 36-bit expression at location "." (whether or not location "." is currently open). A strange case can occur of which the user should be cautioned. There is an ambiguity as to the section in which to start effective address translation. DDT assumes the left half of "." (i.e., the last location opened by the user). If, for example, having opened location 0,,1234, which contains 7,,4321, the user issues the command "$$[", DDT will then calculate the effective address as the contents of location 4321 in section 0 indexed by the right half of register 7. If bit 13 is on, it will treat that word as an IFIW and continue the address calculation. Although this is probably not expected, it is exactly what the hardware would do since the indirect word came from section 0. Had the user opened location 1,,1234, containing 7,,4321, DDT would take the contents of location 7004321 and continue from there.

If no <ESC>s delimit the address expression, DDT simply uses the full 36-bit expression as the address (e.g., "30,,30/" says open location 30000030; "-1/" says open location 777777777777). It is rare for any user to have that much disk space but the hardware will not permit an address over 30 bits wide. Again, only FILDDT can actually reference an address greater than 30 bits wide and in all cases the address must be a non-negative, 36-bit integer.

There is a special case in which DDT does something "kinky". If a space was typed in entering the address expression, or if no explicit address was typed, DDT will form the 36-bit actual address by using only the right half of the 36-bit address expression plus the left half of "." as the section number. An example of this is a person using the "last word typed" by simply using a <TAB>. This not-at-all-obvious behavior is implemented so that the user can type in expressions such as "JRST PAT<TAB>" and have DDT go to location PAT in the same section as the JRST PAT instruction rather than going to address 254000000000+(PAT modulo 2**18). Another common usage of this "feature" would be in chaining down linked lists where the link pointer is an 18-bit, section-local address in the left half of a word. To do this the user may type "sp$$Q/", where "sp" means space. This is one of those cases where usefulness outweighs cleanliness of implementation and documentation.

Modifying Memory

Two new commands have been added to facilitate DDT's manipulation of the user address space.

AUTOMATIC WRITE-ENABLE

If the user attempts to deposit into a write-protected memory location, the $W or $0W command instructs DDT to attempt automatically to write-enable the memory location, do the memory deposit, then finally rewrite-protect the memory location (the default for TOPS-10). The $$W or $$0W command instructs DDT to simply give an error indication if the user attempts to change a write-protected memory location and is the default for TOPS-20. For FILDDT the use of this command is restricted to non-file usage such as "DDT-ing" the running monitor/memory space.

AUTOMATIC PAGE-CREATION

The $1W command instructs DDT to try automatically to create the page the user is trying to deposit into if it does not already exist. This is the default for TOPS-20. The $$1W command instructs DDT to give an error indication if the user attempts to write into a nonexistent page and is the default for TOPS-10. EDDT and FILDDT doing super I/O or "DDT-ing" an .EXE file will NEVER attempt to create a nonexistent page. For FILDDT the user must specify patching the file when he/she starts FILDDT in order to be able to create new pages (e.g., extend the file or fill in a gap in the middle of the file, TOPS-20 only).

Page Mapping and Physical Addressing

In DDT Version 41 all flavors of DDT support page mapping and address relocation as well as register and physical address manipulation. All of these functions use some variation of the $U/$$U DDT command. In general these functions may be mixed together as in the case of address relocation and page mapping.

CAUTION

The $U command syntax in DDT is totally different and generally incompatible, from previous versions of DDT! The user is MOST strongly urged to carefully read this section on memory mapping and addressing!

PHYSICAL ADDRESSING

DDT now has the concept of "physical" addressing in addition to its normal "virtual" addressing. The $U command instructs DDT to use normal virtual addressing, which was the previous method. The $$U command instructs DDT to manually track down the honest physical address rather than the virtual address space in which DDT finds itself running. Physical addressing is really applicable only to EDDT or to FILDDT looking at the running monitor/memory. User mode DDT, EDDT running in user mode, VMDDT, and FILDDT looking at a disk all treat $U and $$U identically. In physical addressing location Ø is not register Ø but rather physical memory location Ø, page Ø, bank Ø, and box Ø (that memory location on the hardware memory bus that responds to all address bits equal to Ø).

When the $$U DDT command is issued, "physical" locations Ø to 17 become "registers" Ø to 17. For user mode DDT, this means locations Ø to 17 become DDT's registers rather than the user's registers. Although the user's registers will be properly restored on DDT-exit, $$U merely directs DDT not to use the internal "fake", i.e., user, registers. For FILDDT this means file words Ø to 17, if mapped by the .EXE directory, become locations Ø to 17 normal for a data file.

Subsequent issuance of the $U DDT command will redirect locations Ø to 17 to being DDT's internal "fake" registers again, except for FILDDT looking at a data file or doing super I/O to a disk.

Note that for executive mode EDDT to utilize physical addressing, the paging hardware must have been enabled <u>prior</u> to DDT-entry. This requirement exists because EDDT, in order to access all of physical memory, needs to map the desired physical address into its own executive virtual address space, which it does by manipulating the existing page maps. For EDDT to provide physical addressing capability without this restriction would require two more memory pages to be dedicated to EDDT for building temporary page maps, plus support code, etc.

For FILDDT to examine/modify physical memory, a 7.ØØ or later release of the TOPS-1Ø monitor is required; no release of TOPS-2Ø supports FILDDT-ing physical memory.

PAGE MAPPING

All flavors of DDT now support page mapping in both the KI- and the KL-tradition. EDDT in executive mode will dynamically figure out which style of paging is in effect and operate accordingly. All other flavors of DDT, including EDDT running in user mode, will assume the mode of paging used by the operating system for which DDT was assembled, KI-paging for TOPS-1Ø and KL-paging for TOPS-2Ø.

The flg$1ØU command is used to select KI-paging emulation; to select KL-paging, the flg$11U command is issued. In either case, if flg is zero, the paging emulation is disabled; if flg is non-zero, the appropriate paging emulation is enabled.

In executive mode EDDT or FILDDT looking at the running monitor/memory space, DDT will internally utilize physical addressing in order to provide the user the true mapped virtual address space if desired.

KI-PAGING -- For KI-paging, which is the TOPS-1Ø default, the page mapping command for the executive virtual addressing space is [upt<]ept$[Ø]U. Upt is the optional physical memory page number of the user process table for setting the "per-process" addressing space -- exec virtual addresses 34ØØØØ through 377777. EPT is the physical memory page number of the executive process table. The user virtual addressing space is selected by the upt$1U command. The command $U returns DDT to regular unmapped virtual addressing.

KL-PAGING -- For KL-paging, which is the TOPS-2Ø default, the page mapping command for the executive virtual addressing space can take two forms. One is EPT$[Ø]U where EPT is the physical memory page number of the executive process table. The second is epx$$[Ø]U where epx is the index into the SPT of the executive process table pointer. To select the user virtual addressing space, the command is upt$1U or upx$$1u. Upt is the physical memory page number of the user process table. If upx$$1U is used, upx is the index into the SPT of the user process table pointer. The command $U returns DDT to regular unmapped virtual addressing.

To map a single section, 256K address space under KL-paging the command is either sec$2U where sec is the physical memory page number of a KL-paging section map, or sex$$2U where sex is the index into the SPT of the section map.

Basically, under KL-paging, $ØU selects the EPT, $1U selects the UPT, and $2U selects a single section. A single $ indicates the physical memory page number and two $s indicate an SPT index.

SETTING THE SPT

FILDDT will automatically define the start of the SPT from the symbol "SPT" for disk files that are assumed to contain monitor dumps; this assumption is made for TOPS-2Ø only. The command spt$6U specifies to DDT that the SPT starts at address "spt".

REGISTER ADDRESSING

The command acs$5U instructs DDT to use the 2Ø consecutive locations starting at ACs as the registers. DDT maintains an internal copy of the registers so changing "register" 3 will not affect, for example, acs+3. FILDDT, when reading an .EXE file, will automatically load its internal "fake" registers as though the user had typed CRSHAC$5U if TOPS-1Ø or BUGACS$5U if TOPS-2Ø. Note that if physical addressing mode has been entered by the user issuing the $$U command, the internal "fake" registers are ignored. If the user subsequently re-enters virtual addressing (via some form of the $U command), an acs$5U command may also have to be reissued to get the registers back. This does not affect the saving and restoring of the hardware registers in user or executive DDT, only what DDT will use for typing-out locations Ø to 17.

The command flg$3U explicitly controls the usage of DDT's internal "fake" registers. If flg is Ø, the "fake" registers are ignored, i.e., Ø to 17 are taken from the true current addressing space. If flg is non-zero, addresses Ø to 17 are taken from DDT's internal copies of the registers.

The $U command, except for FILDDT-ing a data file or doing super I/O to a disk, will return DDT to its internal "fake" registers. The selection of registers is completely independent of any page mapping in effect. Changing virtual address spaces does not change the "registers."

In executive mode DDT, only the command n$4U will switch DDT to use and thus display hardware AC block n, available only for KL1Øs and KS1Øs. The user is warned that 7$4U on a KL1Ø will bring rapid and rabid death, as the microcode uses AC block 7. On DDT exit, DDT will restore the AC block context to the state it was in at DDT entry.

ADDRESS RELOCATION AND PROTECTION

As an aid to looking at data structures that are formed using pointers as offsets rather than pointers as absolute values, DDT Version 41 will allow the user to set both a base relocation address to be added to all addresses used in location examining commands and a protection address beyond which the user "virtual" address is illegal. In this case virtual is taken as meaning prerelocated. This is exactly analogous to the KA10 hardware relocation and protection strategy. In fact this could be used as such to "mimic" the $U KI/KL/KS10 functionality on a KA10 in executive mode. The form of this command is bas$8U where bas is the base virtual address, and prt$9U where prt is the maximum address the user will be allowed to type in. Note that page mapping, address relocation, and protection are independent mechanisms, with address relocation and protection being performed before any mapping is done. The protection address has no effect on the final "physical" address generated by any mapping currently in effect.

$U COMMAND SUMMARY

All $U/$$U commands take the following form:

1. $U - Unmapped virtual addressing
2. $$U - Unmapped physical addressing
3. ept$[$][0]U - Select executive virtual addressing
4. upt$[$]1U - Select user virtual addressing
5. sec$[$]2U - Select single section
6. flg$3U - Select (deselect) internal fake registers
7. acb$4U - Select hardware ac block
8. acs$5U - Load internal fake registers
9. spt$6U - Select base of SPT

10. bas$8U - Set base relocation address

11. prt$9U - Set protection address

12. flg$10U - Select (deselect) KI-paging

13. flg$11U - Select (deselect) KL-paging

where:

acb := Integer AC block number
acs := Address of 20-word register block
bas := Base relocation address
ept := Executive process table page number
flg := Selection flag, zero to deselect, non-zero to select
prt := Protection (maximum allowable) address
sec := Section map page number
spt := Address of SPT
upt := User process table page number

ADDRESS CHECKING (EXECUTIVE EDDT ONLY)

When running in executive mode, EDDT Version 41 is now much more extensive in validity-checking memory references. In particular, EDDT will not cause a NXM (page fault) trap to the resident operating system if the user types in a nonexistent or unmapped address, but rather will simply type its ubiquitous ?<DINK><TAB> error message.

ADDRESS BREAKING

DDT will no longer cause an address break to occur when examining or depositing a location at which an address break condition has been set. This condition applies only to "user" examines and deposits; an address break set in DDT will still cause an address break to occur.

SPECIFYING THE START ADDRESS

The $G command now expects a 36-bit address, obviously with bits 0 to 5 off, at which to start the user program. This means that the users of programs such as the TOPS-10 monitor which define symbols like "DEBUG=:<JRST .>" can no longer go either DEBUG$G or DEBUG$X at the user's whim, but must decide on one form or the other. The default is to do nothing -- i.e., to settle for the DEBUG$X form.

SYMBOLIC EXPRESSION TYPE-IN AND TYPE-OUT

DDT Version 41 has expanded the range of both symbolic type-in and symbolic type-out.

Symbolic Type-In

The JSYS opcode, number 1Ø4, has been added to TOPS-2Ø DDT as have all the TOPS-1Ø UUOs. This allows debugging of programs that run under the compatibility package. However, TOPS-1Ø CALLIs have not been added.

Multiply-Defined Symbol Type-In

If the user types an ambiguous symbol (a symbol defined at two or more places outside of the current local symbol table and not in the current local symbol table), DDT will issue an "M" error message.

Selecting No local Symbol Table

The $: command is issued without an explicit module name to use the local or 'opened' symbol. This is the initial state in which DDT starts.

Symbol Cache

DDT now has a symbol cache of symbols recently used to type out values. This cache is used primarily for type-out; type-in will check the symbol cache for a matching symbol from the currently opened or local symbol table. If no match is found, the cache is ignored and the regular symbol table is used. The symbol cache is "flushed" on the issuance of any $: command.

Symbolic Type-Out

DDT now goes to great pains to find any possible user-defined symbol, such as an OPDEF, to match the expression DDT is trying to type out. The order in which DDT searches for a symbol match in symbolic type-out mode for non-I/O instructions is:

1. Full 36-bit match; OP, AC, I, X, and Y fields; e.g., the TOPS-2Ø monitor calls such as GTJFN

2. OP, I, X, and Y fields; e.g., the TOPS-1Ø monitor calls such as FILOP

3. OP and AC fields; e.g., the TOPS-1Ø monitor calls such as INCHWL or "instructions" such as HALT

4. OP field only; e.g., user UUOs or "OPDEF XMOVEI [SETMI]"

5. DDT's internal hardware opcode table

The order in which DDT searches for a symbol match in symbolic type-out mode for I/O instructions is:

1. I/O OP and DEV fields; bits Ø to 12 -- e.g., KL1Ø APRID or KS1Ø RDCSB

2. Regular, non-I/O, OP field; e.g., KS1Ø UMOVE

ASCII TYPE-OUT

DDT Version 41 adds the type-out mode commands $8T and $9T to type out 8-bit ASCII or 9-bit ASCII, respectively; i.e., pick up 8- or 9-bit bytes and "type" them straight as is -- which with current TOPS-1Ø and TOPS- 2Ø operating systems means as 7-bit ASCII.

COMMAND FILES

The $Y command, used in TOPS-1Ø DDT only, has been changed somewhat, both in input and output logging functions.

Command Input

If the user does not type a 36-bit expression to be used as a file name, such as $""FILNAM"$Y, but just types $Y by itself, DDT will prompt with "File: ". After the prompt the user can enter a TOPS-10 file specification in the form dev:name.type[directory]/switches where [directory] can of course contain SFDs.

/A SWITCH

The /A switch instructs DDT to abort the command file if a DDT-detected command error occurs, such as reference to an undefined symbol.

Command Output (Logging)

When reading a command file, $Y command, DDT will no longer "log" all output onto device LPT: but rather just type-out onto the user terminal.

AUTOMATIC PATCH INSERTION

The automatic patch insertion facility, $< and $> commands, are basically the same as in Version 40 of DDT with only minor differences.

Patch Opening

The user may specify patching either by sym$< where sym is the name of a symbol which will be automatically updated at the termination of the patch, or via exp$< where exp is any 36-bit expression representing the address of the resultant patch. If the latter form of the patch command is used, no symbol will be updated to the end of the patch.

Default Patching Symbol

The list and order of default patching symbols which DDT uses when the user does not supply an explicit patching symbol is now:

1. PAT - TOPS-10 EDDT only
2. FFF - TOPS-20 EDDT and MDDT only
3. PAT.. - all flavors
4. PATCH - all flavors
5. PAT - all flavors except TOPS-10 EDDT

Default Patching Address

If the user does not supply an explicit patching symbol and DDT is unable to find one of the default patching symbols, the address specified by the right half of location .JBFF (even on TOPS-20) is used. If on patch close ($> command) the patching address was defaulted to via .JBFF, both the right half of location .JBFF and the left half of location .JBSA are updated.

Patch Closing Confusion and Restriction

With DDT Version 41 it no longer matters how or when the user types the $> command, either immediately after the final word expression, or after a <CR> or <LF>, to terminate the final word expression -- DDT will never generate a zero word for free.

There is a very obscure restriction, however, on the use of the command in conjunction with the $> command. If the user is referencing an undefined symbol in the expression for the last word of the patch, that expression must explicitly be terminated in such a fashion as to close the location before terminating the patch. For example, "MOVE T1,BLETCH $>" is illegal but "MOVE T1,BLETCH cr$> (where "cr" indicates a carriage return) is okay.

BREAKPOINTS

The breakpoint logic in DDT Version 41 has been extensively revamped in order to support extended addressing. The default number of breakpoints is now 12 decimal and can be set arbitrarily high by defining the symbol NBP equal to the number of desired breakpoints. The number of breakpoints is then restricted by memory space rather than 9 or 36 decimal, depending on which code restriction one chooses to believe.

Setting Breakpoints

DDT can now set a breakpoint in code running in any section, with two restrictions:

1. If DDT is currently running in section Ø, breakpoints can only be set in section Ø. Refer to "Location Examining Restrictions" discussed at the beginning of this appendix.

2. DDT must be mapped in the section containing the code in which breakpoints are to be placed. The logic of this is that since there is no way for DDT to cause unconditional transfer of control to DDT with only 36 bits, some portion of the section address space must be devoted to DDT. Therefore, given this restriction, one might just as well put all of DDT in that section since it makes for a cleaner and simpler implementation. Note that this does not mean DDT must be running in that section, but merely that DDT must be mapped in that section!

It does not matter into how many different sections the same code is mapped, as long as DDT is mapped into the same sections since DDT is "section-independent." For example, the TOPS-2Ø monitor maps section Ø and section 1 identically. If a breakpoint is set at address 1ØØ4567 in section 1, but the PC was at 4567 in section Ø when the breakpoint was executed, DDT does not care, as long as DDT is mapped in that section, as it is in the example of the TOPS-2Ø monitor.

The syntax for setting a breakpoint is now opn<bpt$nB where n is optional and, if specified, declares the breakpoint number to be assigned to that address. Bpt is the 36-bit address at which to place a breakpoint. Opn is an optional 36-bit address to open and display upon execution of the breakpoint. The syntax was changed because two full 3Ø-bit addresses could not be squeezed into two halfwords.

DDT will no longer assign two different breakpoints to the same address, either accidentally or under user control. If the user attempts to set a breakpoint at a location at which a different breakpoint is already set, the old breakpoint is cleared first.

Breakpoint Type-Out

Upon execution of a breakpoint, DDT will now always type out the user instruction in instruction format regardless of the permanent type-out mode, and set "." to the breakpoint address. If opn was specified as in the "Setting Breakpoints" section above, DDT will also display the contents of location opn in the permanently set type-out mode, and "." will be updated to opn, with the breakpoint address itself becoming the previous PC sequence and so available via the $<CR>, etc., commands.

Examining Breakpoint Locations

The $nB command continues to be the "address" of breakpoint n's data base, but $nB is no longer equal to $n-1B+3. The breakpoint data base of interest to the user now has the following format:

1. $nB+Ø/ If nonzero, the address for breakpoint n
2. $nB+1/ The conditional break instruction, break if skips
3. $nB+2/ The proceed count, break on transition to Ø
4. $nB+3/ If greater than or equal to zero, the address is displayed

The rest of the breakpoint data base should not be of use to the user.

Unsolicited Breakpoints

DDT Version 41 has a new breakpoint facility, the ability to handle unsolicited breakpoints (i.e., breakpoints that DDT did not itself set). If control passes to location $ØBPT+1, where $ØBPT is a global DDT symbol, DDT will act as if a breakpoint had been set at the address-1 contained in location $ØBPT. The address in $ØBPT must be set up as if the CPU executed a JSR $ØBPT instruction. After "hitting" an unsolicited breakpoint the user can proceed with program execution with the $P command. All arguments to the $P command such as proceed count or auto-proceed, $$P, are ignored.

Although this facility gives programs the ability to cause breakpoints at any time, thus getting into DDT with the program state carefully preserved, it is intended to be of most use in conjunction with an as-yet-unimplemented monitor command, such as control-D, to "force" a breakpoint on a program without having to control-C/DDT the program. Then the user could simply continue with the program by typing $P.

SINGLE-STEPPING THE PROGRAM

The $X DDT command has been significantly modernized, and speeded up in general with Version 41 of DDT.

New Opcodes

The ADJSP, DADD, DSUB, DMUL, and DDIV instructions have been added to DDT's $X table although double- and quad-word integers, for DADD, etc., are still typed-out as two or four single words rather than one big multiple-precision integer. All of the extended JRST-class instructions are correctly simulated/traced. A user-UUO being executed in a non-zero section is simply XCTed and is not traced.

Byte-Manipulation Type-Out

A rudimentary byte-manipulation-instruction type-out facility was added to DDT Version 4Ø, to display the byte pointer and the contents of the effective address of the byte pointer. The EXTEND-class instructions are not handled.

Effective Address Calculation

DDT now always calculates the effective address of the instruction being $Xed rather than just blindly "doing it" in order to both prevent DDT from getting an illegal memory reference, as well as to make DDT be independent of the section in which the user PC resides. Note that DDT does not have to be mapped into the user PC section to handle

$Xes although if the user PC is in a non-zero section, DDT must be in a non-zero section; besides, it is usually faster too!

KS10 I/O Instruction Trace

The KS1Ø-specific I/O instructions that reference the Unibus, in the executive mode only, are not traced; only the contents of the register specified in the AC field are displayed. Since the Unibus device registers can be reference-volatile (i.e., merely referencing one can cause it to change -- such as the DL11 data registers) DDT does not type-out the contents of the referenced Unibus address. Further, since the effective address of the instruction is not calculated in a standard format, at least as far as DDT is concerned, the effective address itself is not even displayed.

PC Skipping

If the user instruction being $Xed skips, DDT will (1) type "<SKIP>" if the PC skips by one location, or (2) "<SKIP n>" if the PC skips by n locations, where n is less than or equal to the DDT assembly parameter SKPMAX, by default 3. If the PC changes more drastically than that, e.g., going to a smaller address, DDT will type "<JUMP> instead.

ERCAL/ERJMP

DDT on TOPS-2Ø will now handle instructions followed by either an ERCAL or an ERJMP instruction, which is really just a 72-bit instruction with two effective addresses. If the instruction being executed does not take the error jump, DDT will print "<ERSKP>" after the normal instruction trace to indicate to the user that an ERCAL or ERJMP was just skipped (i.e., the PC incremented by two rather than one) and will not display the ERCAL or ERJMP instruction. If the instruction does take the error jump, the ERCAL or ERJMP instruction will be displayed. If an ERCAL instruction is displayed, register 17 will also be displayed, and the PC will be changed to the error address.

DDT will print "<ERSKP>" rather than showing the ERCAL or ERJMP instruction since DDT has no way of telling whether or not the instruction itself caused the skip, as in a SKIPA, or if the PC merely "fell through" the ERCAL or ERJMP instruction, as in a successful MOVE.

Users of EDDT and MDDT should be cautioned about $Xing instructions followed by an ERCAL or ERJMP in non-zero sections -- the monitor has a tendency to transfer control to the error address in section 0, which will cause a BUGHLT.

$Xing and INIT

DDT will now let the user $X an INIT monitor call. DDT will print out <SKIP 2> if the INIT fails or <SKIP 3> if the INIT succeeds.

$X Speed-Up

By building into DDT a table of instructions that can cause the state of the known world to change, and assuming the state of the world does not change if the instruction being $Xed is not so marked, the time required to $X an instruction is cut by roughly a factor of ten. This results in a dramatic performance increase especially for EDDT on KL10s where waiting for the console front end to switch between secondary and primary protocol is very time-consuming.

Repetitive $Xes

The $$X command now takes an optional address range. Normally $$X will terminate when the user PC inclusively enters the range .+1 to .+ SKPMAX (default value of SKPMAX is 3). The user may specify lwr<upr>$$X where lwr is the lower address boundary and upr is the upper address boundary which, if the user PC ever inclusively enters the range so specified, terminates the $$X. If only lwr is specified, upr defaults to lwr+SKPMAX. For example, this command is very useful for recovering from having $Xed a PUSHJ instead of having $$Xed the PUSHJ.

$Xing from instr$X

If the user $Xes a return from a subroutine which was entered by doing an instr$X, for example "PUSHJ P,SUBRTN$X" where SUBRTN has a breakpoint in it, DDT simply "returns" from the original instr$X rather than proceeding to $X, the internals of DDT itself.

$$X Status

DDT will now respond to a ? character being typed during an $$X sequence by typing "Executing: " followed by the current user "pc" and instruction being executed. Typing any other character terminates the $$X immediately.

$X PC

The $. command now acts like the . command, only $. returns the value of the $X PC; i.e., the address of the next instruction to be $Xed. The $$. command returns the previous $. value, useful for $$.<$$X as in the "Repetitive $Xs" section above.

SEARCHES

Most of the differences in how DDT Version 41 handles searches simply involves bug fixes, not major changes in the logic of searching.

Nonexistent Pages

DDT Version 41 now simply skips over pages that do not exist in the address space being searched, rather than terminating the search as soon as a hole has been found.

Missed Matches

The bug that caused TOPS-2Ø DDT to miss many valid matches is fixed in DDT Version 41.

Effective Address Searches

Since almost all address calculations start with an IFIW basis, Version 41 will assume that each word it examines is an instruction and will perform an IFIW effective address calculation. The final result must match in all 3Ø bits. DDT will do a full 36-bit comparison internally, so the address being searched for should not contain anything in bits Ø to 5.

The exceptions are such things as interrupt vectors on the KL1Ø and KS1Ø.

Address Limit Defaults

With the advent of extended addressing and physical addressing, the address limits are defaulted somewhat differently than in previous versions of DDT:

1. EDDT, MDDT (TOPS-2Ø only), UDDT, and VMDDT

   a. Lower Limit: <current section>,,Ø

   b. Upper Limit: <current section>,,777777

2. FILDDT looking at an .EXE file

   a. Lower Limit: Ø

   b. Upper Limit: highest virtual address mapped

3. FILDDT looking at a data file

   a. Lower Limit: Ø

   b. Upper Limit: highest word written in file

4. FILDDT looking at disk structure/unit

   a. Lower Limit: Ø

   b. Upper Limit: highest word in disk structure/unit

5. FILDDT looking at running monitor

   a. Lower Limit: Ø

   b. Upper Limit: 777777

6. FILDDT looking at physical memory (TOPS-1Ø only)

   a. Lower Limit: Ø

   b. Upper Limit: Highest extant memory address

As with any defaults, DDT may not properly "guess" the correct address limits. In particular, if the user has mapping or address relocation, in effect, the virtual address range produced may have nothing whatsoever in common with the address limit defaults chosen by DDT.

Search Matches

DDT will leave each address matched by its search on the "pc stack" available to commands like $<CR>. When the search is terminated DDT will set "." to the last address searched.

Searching Status

DDT will now respond to a ? character during a search by typing "Searching: " followed by the current location and value being searched. Typing any other character terminates the search immediately.

WATCHING

DDT Version 41 allows the user to "watch" a location, waiting for it to change. Although primarily useful for FILDDTing the running monitor, it is present in all flavors of DDT for completeness. The syntax of the watching command is exp$V, where exp is the address to be watched. If no explicit address is specified, the last location opened by the user will be used.

Upon initial issuance of the $V command, the location is displayed. Thereafter the location is continuously monitored, and will be displayed every time its contents change. In user mode DDTs, and this includes TOPS-20 MDDT as well, the location is checked once a clock tick, approximately 50 to 60 times a second. In exec mode EDDT, the location is continuously being monitored; no "pause" is attempted.

DDT will respond to a ? character being typed during an $V sequence by typing "Watching: " followed by the current location and contents being watched. Typing any other character terminates the $V immediately.

ZEROING MEMORY

The algorithm used by DDT previous to Version 41 has only limited usefulness in today's modern virtual world, especially on TOPS-20. However, to avoid "breaking" already existing control or MIC files which may use the $$Z command, it remains unchanged. A new command has been implemented - lwr<upr>exp$z where lwr is the starting address, upr is the ending address, and exp is the 36-bit quantity to deposit in each word inclusively bounded by lwr and upr. Both lwr and upr must be specified. If exp is not specified, 0 is used as the default.

A special note: The creation of zeroed pages, which formerly were nonexistent, by the $Z and $$Z commands is under the control of the automatic page create flag, i.e., the $1W and $$1W commands -- see section "Automatic Page Creation" above.

DDT will now respond to a ? character being typed during an $Z sequence by typing "Zeroing: " followed by the current location and value being "zeroed." Typing any other character terminates the $Z immediately.

SPECIAL MASKS

DDT Version 41, actually having started with DDT Version 40, has several new "masks" of interest to the user. None of these masks is currently displayed (e.g., "$3M/") in FILDDT, although they may be set normally.

$0M - Search Mask

The operation of the search mask continues unchanged. The search mask may now be referenced by either the old style $M or the new style $0M commands. The default value remains 777777777777.

$1M - TTY Control Mask

This mask controls special TTY behavior, primarily for TOPS-10 and exec mode EDDT.

TAB SEPARATOR DISPLAY

Bit 17 controls whether DDT will print its usual <TAB> or three spaces for the <TAB> separator. A 0 (the default) selects three spaces, a 1 selects a <TAB>.

TAB SIMULATION

Bit 34 controls tab simulation. A 0 forces the terminal to handle <TAB>$ directly while a 1 selects space fill instead. This condition is automatically set for user mode DDT in which <TAB>$s are always output literally,

RUBOUT CONTROL

Bit 35 controls rubout and Control/W operation. A 0 selects "hardcopy" operation. DDT will echo a ^ and the character being deleted. A 1 will cause rubouts to echo as would a backspace, space, backspace sequence. This condition is automatically set for user mode DDTs. If TTY DISPLAY is set, rubouts echo as <BS><SP><BS>. It is only useful to manually set fancy rubouts in exec mode EDDT.

$2M - Offset Range

The 36-bit "mask" in this case is really a value, used as the maximum offset allowable for typing addresses in the form symbol+offset. The default offset is 1000 octal.

$3M - Byte Mask

This mask is used in conjunction with the $O command for typing bytes in a word that are not necessarily evenly spaced. Whenever an $O command is issued without an explicit byte size, the byte boundaries are determined by bits in the byte mask -- each bit in the byte mask marks the low order bit of a byte. Bit 35 is always considered on. The default value is 0, i.e., one 36-bit byte. For example, the DDT command 040100200401$3M sets the byte mask for typing right-justified eight-bit bytes preceded by a leading four-bit byte.

RADIX-50 SYMBOL TYPE-IN

Since prehistoric times DDT has supported RADIX-50 symbol type-in, but that fact was never documented. The syntax for using a RADIX-50 symbol as a 36-bit item in an expression is sym$5" where sym is the desired RADIX-50 symbol. For example, to search for all occurrences of the symbol PAT.. the DDT commands 37777,,-1$M (only look at low order 32 bits) and PAT..$5"$W suffice.

NEW DDT RUNTIME INFORMATION

A number of new words have been added to DDT's runtime table which describe the state of the machine upon DDT entry. These words are all accessible in executive mode via the DDT command $I+offset, which is not available in FILDDT:

1. $I-01/ APR CONI word
2. $I+00/ PI CONI word
3. $I+01/ Mask of PI channels turned off by EDDT
4. $I+02/ Executive virtual address of EPT
5. $I+03/ Executive virtual address of UPT
6. $I+04/ Executive virtual address of CST

7. $I+05/ Executive virtual address of SPT

8. $I+06/ Original AC-block word; DATAI PAG, if acb$4U

OBSOLETE COMMANDS

The $R, $J, and $L commands, which were used for executive mode paper tape control, are no longer supported. The code is left in the source file for reference purposes but will soon be removed.

FILDDT STARTUP AND COMMANDS

FILDDT is a special version of DDT with the facilities for "DDTing" address spaces other than its own, such as disk files and in particular .EXE files. FILDDT has existed for years but has always been off in the background as a specialized "tool" for the exclusive use of monitor programmers looking at crash dumps. With DDT Version 41, FILDDT is now a general-purpose utility for use by the "general public," particularly people who have data bases resident in disk files. One example of this is .REL files.

Symbols

For efficiency, FILDDT builds the symbol table(s) it will actually use at runtime in its own address space. Virgin FILDDT has no symbols. The symbol table, if any, for FILDDT in FILDDT.EXE is completely independent of the address space being FILDDTed and does not count. There are special commands to instruct FILDDT to extract and build internal-to-FILDDT copies of symbol tables from .EXE files. These commands are shown below. Once FILDDT has set up its internal symbol table(s), it may then be saved with the internal symbol table(s) for later use by exiting to monitor level, with the ^Z FILDDT command, and typing the "SAVE" command.

TOPS-10

When FILDDT is started it will prompt "FILE: ". The user may at this time optionally enter a standard TOPS-1Ø file specification followed by switches. At least one function is mandatory. SFDs are of course legal in the directory specification.

/D COMMAND

The /D command or function switch instructs FILDDT that the file specified is a data file -- i.e., do not map the file as an .EXE file and use real file words Ø to 17 for locations Ø to 17.

/F COMMAND

The /F command or function switch instructs FILDDT to "DDT this file anyway". It is useful only in conjunction with the /S command or function switch which normally re-prompts for another file specification. Used in conjunction with /S, which implies an .EXE file, FILDDT will use the file from which symbols were extracted as the file to be "DDTed".

/H COMMAND

The /H command or function switch instructs FILDDT to type-out a brief help text, abort the current command, and prompt the user for another command.

/J COMMAND

The /J command or function switch is applied to a job number rather than a file specification and instructs FILDDT to "DDT" the address space of the job specified. Since FILDDT uses JOBPEK monitor calls to access the specified job's address space, the success or failure of any given memory reference is dependent on the job being resident in main memory. If the job is swapped out or if the memory reference is to a page which is paged out, the memory reference will fail. This is a privileged command.

/M COMMAND

The /M command or function switch instructs FILDDT to "DDT" the currently running monitor and physical memory address space. It is controlled by use of the $U and $$U commands and is a privileged command.

/P COMMAND

The /P command or function switch instructs FILDDT to enable for writing as well as reading the specified address space. Note that DDT's internal fake registers are always writable.

/S COMMAND

The /S command or function switch instructs FILDDT to extract only the symbol table from the file specified, replacing any symbol table FILDDT may already have. Unless overridden by the inclusion of a /F command FILDDT will, after having read the symbol table, again prompt the user for the next FILDDT command.

/U COMMAND

The /U command or function switch is applied to a file structure or disk unit, rather than to a complete file specification. It indicates to FILDDT that the user wants the entire physical address space represented by that file structure or disk unit independent of any "file structure mapping" normally imposed by the monitor. This is a privileged command.

TOPS-20

With DDT Version 41 FILDDT on TOPS-2Ø runs in native mode and uses the PMAP monitor call for all regular file access. FILDDT will also type a brief message telling what address space is about to be "DDTed" before going into DDT mode.

DRIVE COMMAND

The DRIVE command allows examination of a disk unit on a system channel without regard for whether the unit is mounted as part of a file structure or whether it even has the necessary information to be mounted. The latter case could occur if the HOME blocks were wiped out. If, however, the drive is part of a mounted file structure, FILDDT will type a message indicating the structure to which it belongs. This is a privileged command.

The format of the DRIVE command is:

```
DRIVE (FOR PHYSICAL I/O IS ON CHANNEL) c (UNIT) u
```

ENABLE DATA-FILE COMMAND

The ENABLE DATA-FILE command instructs FILDDT to treat the file as pure data, even if a valid .EXE directory is detected, and in particular to use real file words Ø to 17 as locations Ø to 17.

ENABLE PATCHING COMMAND

The ENABLE PATCHING command instructs FILDDT to enable any subsequently specified address space for patching. This command is ignored when looking at the running monitor since there is no monitor call to "poke" the running monitor.

EXIT COMMAND

The EXIT command instructs FILDDT to return to command level. If FILDDT has an internal symbol table, due to a previous LOAD or GET FILDDT command, a SAVE command will save FILDDT with the symbols preloaded.

GET COMMAND

The GET command instructs FILDDT to set up the disk file filespec as the address space to be "DDTed", as modified by the optional switches or previous ENABLE commands. The available switches are:

1. /DATA - The /DATA switch is equivalent to a previous ENABLE DATA-FILE command.

2. /PATCH - The /PATCH switch is equivalent to a previous ENABLE PATCHING command.

3. /SYMBOL - The /SYMBOL switch instructs FILDDT to extract symbols from the specified .EXE file before "DDTing" the file, discarding any symbols that FILDDT may already have. This switch is legal only with .EXE files.

The format of the GET command is:

GET (FILE) filespec (optional switches)

HELP COMMAND

The HELP command instructs FILDDT to type-out a short summary of the available FILDDT commands.

LOAD COMMAND

The LOAD command instructs FILDDT to extract symbols from the disk file filespec, which must be an .EXE file, then to return to FILDDT command level. This command is legal only for .EXE files.

The format of the LOAD command is:

LOAD (SYMBOLS FROM) filespec

PEEK COMMAND

The PEEK command instructs FILDDT to use the currently running monitor as the address space to be "DDTed". The available address space is currently limited to monitor executive virtual addresses Ø to 777777, since the PEEK monitor call will only accept 18-bit address arguments for executive virtual addresses. Physical memory addressing is not available. This is a privileged command.

STRUCTURE COMMAND

The STRUCTURE command instructs FILDDT to use as the address space an entire disk file structure. All mapping is independent of "file structure mapping" normally imposed by the monitor. This is a privileged command.

The format of the STRUCTURE command is:

STRUCTURE (FOR PHYSICAL I/O IS) str:

Defaults

The following is a list of the various defaults supplied by FILDDT:

1. DSK: is the default file device unless super I/O is specified (which requires an explicit file structure or disk unit name).
2. .EXE is the default file type or extension unless either a data file or super I/O is specified, in which case there is no default file type or extension.
3. The default directory is the user's default directory.

4. The specified address space is read-only.

5. If "DDTing" an .EXE file and FILDDT does not already have a symbol table, extract the symbol table (if any) from the .EXE file first.

6. If "DDTing" an .EXE file and the symbol CRSHAC (in TOPS-1Ø) or BUGACS (in TOPS-2Ø) exists, give a "free" CRSHAC$5U or BUGACS$5U command. When the CRSHAC/BUGACS symbol does not exist, use file words Ø to 17 (if any) as mapped by the .EXE directory for locations Ø to 17. For TOPS-2Ø only, if the symbol SPT exists, also give a free SPT$6U command as well.

Other FILDDT-Specific Commands

Following are the commands that are unique to FILDDT.

1. ^E Command - instructs FILDDT to exit the current address space and prompt the user for a new address space. The ^E command is equivalent to a ^Z, START command sequence.

2. ^Z Command - instructs FILDDT to exit to monitor level after having written out any changes to the current file. It is most important that the user exit only via ^Z, or ^E, which does an implicit ^Z, in order to guarantee the integrity of the file data. A ^C can leave a file in an indeterminate state with some changes written out to the disk and some not.

3. I/O ERRORS - should FILDDT incur an I/O error when reading or writing a disk file, a warning message will be issued. However, FILDDT will ignore the error. This is to allow the user the ability to fix manually a file with bad data by rewriting the data correctly, hoping the rewriting operation clears the error condition. If the physical disk surface itself is at fault, this is probably hopeless.

Appendix B

CHANGES TO STOPCODES

DELETED STOPCODES

Name	Routine	Type	Message and Explanation
28B	XTCSER	DEBUG	DA28 Broken
6ID	D6SINT	DEBUG	PDP-11 Trying to Give
6PP	D6SINT	STOP	DC76 Putter Problems
8BI	D78INT	JOB	Bad IOWD
8IN	D78INT	JOB	Input Character Count Not Zero
8NC	D78INT	JOB	Not Enough Free Monitor Core
8ON	D78INT	JOB	Output Character Count Not Zero
8PI	D78INT	JOB	Positive IOWD
8VI	D78INT	DEBUG	Version Incorrect
APE	KSSER	JOB	Absolute Page Exceeded
BFU	NETSER	DEBUG	Busy Fouled Up
BNF	COMMON	HALT	BOOTS Not Found
BSY	XTCSER	DEBUG	DA28 Busy
BTL	SCNSER	DEBUG	Backward TTY Link
CØP	COMMON	DEBUG	CPU Ø Power Failure
C1N	CP1SER	DEBUG	CPU 1 Nonexistent Memory

DELETED STOPCODES (Cont.)

Name	Routine	Type	Message and Explanation
C1P	COMMON	DEBUG	CPU 1 Power Failure
CDW	DATMAN	STOP	CORE1 Did Not Work
CRP	CORE1	JOB	Cannot Return PDL
DPO	SEGCON	DEBUG	Directory Page Overlap
HCR	SEGCON	JOB	Hiseg Cannot be Remapped
HNF	ONCE	HALT	High Segment Not Found
ICC	PSISER	HALT	Invalid Condition Code
IFU	NETSER	DEBUG	Interrupt Flag Unrecognized
IRE	KSSER	JOB	IOWD Relocation Error
JJW	CORE1	STOP	Job JDA Wrong
JNE	XTCSER	STOP	JBTADR Not Equal to CORTAL
KN1	CP1SER	HALT	KT10A Not on CPU 1
KNF	XTCSER	STOP	Kontroller Not Free
KR3	D85INT	STOP	Message too Large
LDN	TAPUUO	DEBUG	TT Label DDB Not Found
MAU	CP1SER	DEBUG	Master Already Unlocked
MBE	NETSER	DEBUG	Monitor Buffer Exists
MMN	ERRCON	HALT	Monitor Memory Nonexistent
MMP	ERRCON	HALT	Monitor Memory Parity Error
NDA	FILFND	JOB	No Dummy Access Table
NDC	ONCMOD	STOP	No DF10C Code

DELETED STOPCODES (Cont.)

Name	Routine	Type	Message and Explanation
NED	ONCE	HALT	No Executive DDT
NEM	LP2SER	JOB	No Executive Virtual Memory
NMF	NETSER	DEBUG	No Monitor Buffer Found
NMS	SYSINI	HALT	No Map Slot
NPP	KSSER	STOP	No PI in Progress
NSJ	FILFND	JOB	No Such Job
NUI	XTCSER	DEBUG	Nonexistent Unit Interrupting
NUN	FILUUO	DEBUG	NMB Use-Count Negative
OIP	D8SINT	DEBUG	Output in Progress
OJA	DATMAN	JOB	Outside JOBDAT Area
OMR	KSSER	DEBUG	Out of Mapping Registers
PFA	KILOCK	STOP	Page Free Already
PFC	VMSER	STOP	Page on Free Core List
PNB	DATMAN	DEBUG	Process Number Bad
POP	CLOCK1	STOP	PI on Progress
PTH	KSSER	HALT	Parity Trap Halt
RBQ	SCHED1	STOP	Requeueing to Beginning of Queue
RIE	XTCSER	DEBUG	Remote Interrupt Error
RIP	D8SINT	DEBUG	Read in Progress
SB0	COMMON	STOP	SBus Error on CPU 0
SB1	COMMON	STOP	SBus Error on CPU 1

DELETED STOPCODES (Cont.)

Name	Routine	Type	Message and Explanation
SCB	XTCSER	DEBUG	Spurious CONI Bit
SCO	CP1SER	HALT	Split Cycle On
SCR	SEGCON	DEBUG	Segment Could Not be Read
SFI	FILUUO	JOB	Structure Free-Count Inconsistent
SIB	SWPSER	STOP	Swapper is Busy
TCØ	XTCSER	DEBUG	Task Control Error Ø
TC1	XTCSER	STOP	Task Control Error 1
TC2	XTCSER	DEBUG	Task Control Error 2
TC3	XTCSER	DEBUG	Task Control Error 3
TC4	XTCSER	DEBUG	Task Control Error 4
TC5	XTCSER	DEBUG	Task Control Error 5
TC6	XTCSER	DEBUG	Task Control Error 6
TC7	XTCSER	STOP	Task Control Error 7
TND	TSKSER	DEBUG	Task Not Defined
TOW	CP1SER	HALT	Trap Offset Wrong
UAF	KSSER	STOP	Unibus Addressing Failure
UIP	XTCSER	DEBUG	Not a Unique Interrupt
ULE	LP2SER	JOB	Unexpected LP2Ø Error
ULP	KSSER	DEBUG	UBA Lost PI Assignment
UOD	D8SINT	DEBUG	Unexpected Output Done

NEW STOPCODES

Name	Module	Type	Message and Explanation
ANU	FILIO	DEBUG	AU Not Owned by Us
ARD	DTESER	STOP	Run Away Driver
ARM	FILFND	DEBUG	Access Rings Messes Up
AVE	QUESER	DEBUG	Already have EO
BAA	DTESER	STOP	Buffer Already There
BBS	D85INT	STOP	Bad Byte Size
BDN	DTESER	STOP	Bad Device Number
CIB	CLOCK1	CPU	CPU Interlocks Broken
CSP	SEGCON	JOB	Cannot Store Path
CUØ	NETMCR	STOP	Cannot Use Zero Dispatch
DND	FILIO	DEBUG	Drive Not Dual Ported
DNE	DTESER	STOP	Data Count Not Even
DNH	DTESER	STOP	Driver Not Hungry
DNL	QUESER	DEBUG	DEO Not Interlocked
DOM	KLSER KSSER	STOP	Do Not Own MM Resource
DSS	VMSER	DEBUG	DLTSP Skipped
EFI	DTESER	STOP	Eleven Function Illegal
FEM	ERRCON	HALT	Fatal Error in Monitor
FNG	DTESER	STOP	Function No Good
FON	VMSER	STOP	Funny Address Overlaps Next

NEW STOPCODES (Cont.)

Name	Module	Type	Message and Explanation
FOP	VMSER	STOP	Funny Address Overlaps Previous
FPE	VMSER	DEBUG	Funny Page Must Exist
IOP	COMMON	CPU	I/O Page Failure
IPA	DTESER	STOP	Illegal Post Address
IPC	KLSER KSSER	CPU	Illegal Page Failure Trap Code
LND	FILUUO	DEBUG	Logical Name Not Found
LNF	QUESER	DEBUG	Lock Not Found
MDM	DTESER	STOP	Master DTE Missing
MNA	FILIO	DEBUG	Monitor Buffer Not Available
MNR	ERRCON	HALT	MASTER -11 Not Running
N4C	CPNSER	JOB	Not Four Cached Pages
NCM	IPCSER	JOB	No Core for Message
NET	FILUUO	DEBUG	No Extended RIB
NFB	FEDSER	STOP	No Front-End Device Block
NIS	DTESER	STOP	DTE Not in Indirect State
NJT	ERRCON	STOP	Null Job has TTY
NWA	NETMCR	STOP	No One Wrote Anything
OVA	ONCE	HALT	Out of Virtual Address Space
PCI	DTESER	STOP	Previously Checked Function Code Illegal
PFL	VMSER	DEBUG	Piece on Free List
PFR	VMSER	DEBUG	Piece Out of Free Range

NEW STOPCODES (Cont.)

Name	Module	Type	Message and Explanation
PFS	LOKCON	STOP	Page is Free in Segment
PNW	VMSER	DEBUG	Page Not in Working Set
PRF	KLSER	CPU	Page Refill Failure
PTL	DTESER	STOP	Packet too Large
PUF	SEGCON	JOB	PATH UUO Failed
OEF	DTESER	STOP	Queue Entry Full
OFU	QUESER	JOB	O-blocks Fouled Up
RBO	SCHED1	STOP	Requeueing to Beginning of Queue
RDN	TAPUUO	DEBUG	Regular DDB Not Found
RSJ	CLOCK1	DEBUG	Requeue Same Job
RTM	NETMCR	STOP	Requested Too Much
RWD	FILIO	DEBUG	Returning Wrong Unit's DA
SHU	SCHED1	DEBUG	Swapper Hung Up
SLO	FILFND	JOB	Search List Overflow
SNS	NETMCR	STOP	NTRPCB Not Setup
T1E	DTESER	STOP	To-11 Error
TNI	DTESER	STOP	To-10 Not Idle
TOP	DTESER	STOP	Found To-11 Queue Pointer
WCN	DTESER	STOP	Wrong CPU Number
WEM	NETSER	STOP	William E. Matson
WTP	CLOCK1	DEBUG	Wrong Type of PDL
XPW	LOKCON	STOP	Exchanged Page Went Away

Appendix C
GETTAB CHANGES

This appendix lists all the GETTAB tables that have changed. The GETTAB number is listed to the far left and its constituents are indented to the right.

```
SUBTTL  GETTAB CONSTITUENTS

.GTCNF==11        ;CONFIGURATION
        %CNSUP==136,,11 ;SYSTEM UP TIME
        %CNBCP==137,,11 ;BOOTSTRAP CPU NUMBER
        %CNBCL==140,,11 ;BOOTSTRAP CTY LINE NUMBER
        %CNNCR==141,,11 ;NUMBER OF CPU'S ALLOWED TO RUN
        %CNMBS==142,,11 ;MONITOR BOOTSTRAP FILE STRUCTURE (FROM BOOTS)
        %CNMBF==143,,11 ;MONITOR BOOTSTRAP FILE NAME
        %CNMBX==144,,11 ;MONITOR BOOTSTRAP EXTENSION
        %CNMBD==145,,11 ;MONITOR BOOTSTRAP DIRECTORY
        %CNBPM==146,,11 ;MAXIMUM NUMBER OF SNOOP. BREAKPOINTS WHICH CAN BE DEFINED
        %CNMXF==147,,11 ;FIRST FREE VIRTUAL ADDRESS ABOVE THE MONITOR
        %CNLVO==150,,11 ;VIRTUAL ORIGIN OF LDBS
        %CNHXC==151,,11 ;MAXIMUM NUMBER OF FILOP. EXTENDED CHANNELS
        %CNVSH==152,,11 ;MONITOR VIRTUAL START ADDRESS OF HIGH SEGMENT
        %CNRST==153,,11 ;UNIVERSAL DATE/TIME OF LAST ROLE
                        ;SWITCH ON MULTIPLE CPU SYSTEMS
        %CNDCH==154,,11 ;OFFSET INTO LDB OF LDBDCH
        %CNSF1==155,,11 ;MONITOR BOOTSTRAP 1ST SFD
        %CNSF2==156,,11 ;2ND
        %CNSF3==157,,11 ;3RD
        %CNSF4==160,,11 ;4TH
        %CNSF5==161,,11 ;5TH
        %CNFLN==162,,11 ;TTY LINE NUMBER OF FRCLIN

.GTODP==15        ;ONCE ONLY DISK PARAMETERS
        %ODPMN==4,,15   ;MINIMUM ICPT AFTER REQUEUE TO BACK OF PQ2
        %ODPMX==5,,15   ;MAXIMUM VALUE OF ICPT
```

```
.GTLVD==16      ;LEVEL-D PARAMETERS
     %LDSLB==111,,16 ;OFFSET OF UNISLB IN UDB
     %LDUTP==112,,16 ;UETP PPN [5,33]
     %LDINI==113,,16 ;INI PPN [5,34]
     %LDESZ==114,,16 ;SIZE OF 1 ENTRY IN ERPTBK

.GTDBS==21      ;OBSOLETE

.GTTDB==22      ;OBSOLETE

.GTWCH==35      ;WATCH BITS
     JW.WFI==1B8     ;WATCH FILE

.GTQQQ==41      ;OBSOLETE

.GTQJB==42      ;OBSOLETE

;CPU DATA BLOCKS CONSTANTS AND VARIABLES
.GTCØC==55      ;CPUØ CDB CONSTANTS
.GTCØV==56      ;CPUØ CDB VARIABLES
.GTC1C==57      ;CPU1 CDB CONSTANTS
.GTC1V==6Ø      ;CPU1 CDB VARIABLES
.GTC2C==61      ;CPU2 CDB CONSTANTS
.GTC2V==62      ;CPU2 CDB VARIABLES
.GTC3C==63      ;CPU3 CDB CONSTANTS
.GTC3V==64      ;CPU3 CDB VARIABLES
.GTC4C==65      ;CPU4 CDB CONSTANTS
.GTC4V==66      ;CPU4 CDB VARIABLES
.GTC5C==67      ;CPU5 CDB CONSTANTS
.GTC5V==7Ø      ;CPU5 CDB VARIABLES

;ENTRIES IN CDB CONSTANTS TABLE
     %CCTOS==3,,55   ;TRAP OFFSET FOR KA INTERRUPT LOCATIONS (ADDRESS OF EPT KI/KL/KS)
     %CCTYP==6,,55   ;TYPE OF PROCESSOR (LH-DEC, RH-CUST)
          .CC166==1       ;PDP-6 (OBSOLETE IN 7.Ø1)
          .CCKAX==2       ;KA-1Ø (OBSOLETE IN 7.Ø1)
     %CCMPT==7,,55   ;REL. GETTAB POINTER TO BAD ADDRESS TABLE
          CC%BLN==777B8   ;LENGTH-1 OF BAD ADDRESS SUBTABLE
          CC%BRA==777777B35 ;RELATIVE ADDRESS OF FIRST WORD IN SUBTABLE
```

```
        %CCPAR==12,,55  ;REL. GETTAB POINTER TO PARITY SUMMARY
                CC%PLN==777B8   ;LENGTH-1 OF PARITY SUBTABLE
                CC%PRA==777777B35 ;RELATIVE ADDRESS OF FIRST WORD IN SUBTABLE
        %CCRSP==13,,55  ;REL. GETTAB POINTER TO RESPONSE SUMMARY
                CC%RLN==777B8   ;LENGTH-1 OF RESPONSE SUBTABLE
                CC%RRA==777777B35 ;RELATIVE ADDRESS OF FIRST WORD IN SUBTABLE
        %CCNXT==17,,55  ;POINTER TO NXM SUBTABLE IN VARIABLES AREA
                CC%NLN==777B8   ;LENGTH-1 OF NXM SUBTABLE
                CC%NRA==777777B35 ;RELATIVE ADDRESS OF FIRST WORD IN SUBTABLE
        %CCCSB==2Ø,,55  ;POINTER TO CPU STATUS BLOCK SUBTABLE IN VARIABLES AREA
                CC%CLN==777B8   ;LENGTH-1 OF CPU STATUS BLOCK
                CC%CRA==777777B35 ;RELATIVE ADDRESS OF FIRST WORD IN SUBTABLE
        %CCDSB==21,,55  ;POINTER TO DEVICE STATUS BLOCK IN VARIABLES AREA
                CC%DLN==777B8   ;LENGTH-1 OF DEVICE STATUS BLOCK
                CC%DRA==777777B35 ;RELATIVE ADDRESS OF FIRST WORD IN SUBTABLE
        %CCSDP==22,,55  ;POINTER TO SBDIAG SUBTABLE IN VARIABLES AREA
                CC%SLN==777B8   ;LENGTH-1 OF SBDIAG SUBTABLE
                CC%SRA==777777B35 ;RELATIVE ADDRESS OF FIRST WORD IN THE SUBTABLE
        %CCBPA==23,,55  ;POINTER TO PERF. COUNTS IN VARIABLE SUBTABLE

;ENTRY IN CDB VARIABLES TABLE
        %CVSBR==1Ø7,,56 ;STATUS BLOCKS READ ON THIS CPU
        %CVBPF==11Ø,,56 ;.LE. Ø IF PERFORMANCE COUNTS BEING KEPT (%CCBPA)
        %CVFBI==111,,56 ;NUMBER OF FILE BLOCKS INPUT (READ)
        %CVFBO==112,,56 ;NUMBER OF FILE BLOCKS OUTPUT (WRITTEN)
        %CVSBI==113,,56 ;NUMBER OF SWAPPING BLOCKS INPUT (READ)
        %CVSBO==114,,56 ;NUMBER OF SWAPPING BLOCKS OUTPUT (WRITTEN)
        %CVSNC==115,,56 ;NUMBER OF CPU STOPCDS ON THIS CPU
        %CVSND==116,,56 ;NUMBER OF DEBUG STOPCDS ON THIS CPU
        %CVSNJ==117,,56 ;NUMBER OF JOB STOPCDS ON THIS CPU
        %CVSJN==12Ø,,56 ;LAST STOPCD ON THIS CPU -- JOB NUMBER
        %CVSNM==121,,56 ;LH=NAME OF LAST STOPCD ON THIS CPU
                        ;RH=ADDRESS+1 OF LAST STOPCD ON THIS CPU
        %CVSPN==122,,56 ;LAST STOPCD ON THIS -- PROGRAM NAME
        %CVSPP==123,,56 ;LAST STOPCD ON THIS CPU -- USER PPN
        %CVSTN==124,,56 ;LAST STOPCD ON THIS CPU --TTY NAME
        %CVSUP==125,,56 ;LAST STOPCD ON THIS CPU -- USER PC
        %CVSUU==126,,56 ;LAST STOPCD ON THIS CPU -- UUO
        %CVEJN==127,,56 ;LAST PARITY/NXM ERROR ON THIS CPU -- JOB NUMBER
        %CVEPN==13Ø,,56 ;LAST PARITY/NXM ERROR ON THIS CPU -- JOB NAME
        %CVPPI==131,,56 ;CONI PI, AT LAST PARITY/NXM INTERRUPT
        %CVTPI==132,,56 ;CONI PI, AT LAST ERROR TRAP

;RESPONSE SUBTABLE
        ;ENTRIES 24-31 (KA1Ø LONG FLOATING POINT INSTRUCTIONS) ARE OBSOLETE IN 7.Ø2
        %CVFAD==24      ;FADL'S SIMULATED
```

```
        %CVFSB==25      ;FSBL'S SIMULATED
        %CVFMP==26      ;FMPL'S SIMULATED
        %CVFDV==27      ;FDVL'S SIMULATED
        %CVUFA==30      ;UFA'S SIMULATED
        %CVDFN==31      ;DFN'S SIMULATED

;CPU STATUS SUBTABLE
        %CVSAI==0       ;APR ID
        %CVSAP==1       ;CONI APR,
        %CVSPI==2       ;CONI PI,
        %CVSPD==3       ;DATAI PAG,
        %CVSPC==4       ;CONI PAG,
        ;END OF LOCATIONS COLLECTED ON KI10
        %CVSER==11      ;RDERA
        %CVSRD==12      ;CONI RH20, FOR ALL 8 RH'S
        %CVSDT==22      ;CONI DTEN,
        %CVSE0==26      ;EPT LOCS 0-37
        %CVSE1==66      ;EPT LOCS 140-177
        %CVSU1==126     ;UPT LOCS 500-503
        %CVSA6==132     ;AC BLOCK 6, REGS 0-3 AND 12
        %CVSA7==136     ;AC BLOCK 7, REGS 0-2
        %CVSSB==141     ;START OF SBDIAG DATA (MAXIMUM LENGTH = 50 WORDS)

;KL10 BACKGROUND PERFORMANCE ANALYSIS FIGURES SUBTABLE
        %CVCH0==0       ;RH20 #0 USAGE
        %CVCH1==4       ;RH20 #1 USAGE
        %CVCH2==10      ;RH20 #2 USAGE
        %CVCH3==14      ;RH20 #3 USAGE
        %CVCH4==20      ;RH20 #4 USAGE
        %CVCH5==24      ;RH20 #5 USAGE
        %CVCH6==30      ;RH20 #6 USAGE
        %CVCH7==34      ;RH20 #7 USAGE
        %CVPI0==40      ;PI LEVEL 0 (DTE) USAGE
        %CVPI1==44      ;PI LEVEL 1 USAGE
        %CVPI2==50      ;PI LEVEL 2 USAGE
        %CVPI3==54      ;PI LEVEL 3 USAGE
        %CVPI4==60      ;PI LEVEL 4 USAGE
        %CVPI5==64      ;PI LEVEL 5 USAGE
        %CVPI6==70      ;PI LEVEL 6 USAGE
        %CVPI7==74      ;PI LEVEL 7 USAGE

.GTFET==71      ;FEATURE TEST SETTINGS
        %FTUU2==15,,71  ;UUOS
                F%MPB==15,,1B35  ;OLD VBATCH CODE
```

```
.GTSCN==73          ;SCANNER DATA
        %SCNTA==1Ø,,73  ;(OBSOLETE)
        %SCTFT==11,,73  ;ADDRESS OF FIRST TTY CHUNK ON FREE LIST
        %SCTFP==12,,73  ;ADDRESS OF LAST TTY CHUNK ON FREE LIST

.GTIPC==77          ;IPCF MISC. DATA
        %IPCSP==13,,77  ;PID OF [SYSTEM]GOPHER

.GTCMP==112         ;OBSOLETE
        %CMPMT==Ø,,112  ;OBSOLETE
        %CMPCV==1,,112  ;OBSOLETE
        %CMPDV==2,,112  ;OBSOLETE

.GTVM==113          ;GENERAL VIRTUAL MEMORY DATA
        %VMLST==25,,113 ;OFFSET OF POINTER TO SWAPPABLE DDBS IN UPMP
        %VMUPM==26,,113 ;VIRTUAL ADDRESS OF UPMP
        %VMLNM==27,,113 ;OFFSET OF POINTER TO LOGICAL NAMES IN UPMP

.GTSST==115         ;SCHEDULER STATISTICS
        %SSICM==5,,115  ;OBSOLETE
        %SSEAF==11,,115 ;OBSOLETE
        %SSEAT==12,,115 ;OBSOLETE

.GTCQJ==122         ;OBSOLETE

.GTSQH==124         ;OBSOLETE

.GTSQ==125          ;OBSOLETE

.GTSID==126         ;SPECIAL PID TABLE
        %SIIPC==Ø,,126  ;[SYSTEM]IPCC
        %SIINF==1,,126  ;[SYSTEM]INFO
        %SIQSR==2,,126  ;[SYSTEM]QUASAR
        %SIMDA==3,,126  ;MOUNTABLE DEVICE ALLOCATOR
        %SITLP==4,,126  ;MAGTAPE LABELING PROCESS
        %SIFDA==5,,126  ;FILE DAEMON
        %SIMDC==6,,126  ;MOUNTABLE DEVICE COORDINATOR (HISTORICAL)
```

```
        %SITOL==6,,126  ;TAPE AVR PROCESS
        %SIACT==7,,126  ;[SYSTEM]ACCOUNTING
        %SIOPR==1Ø,,126 ;OPERATOR INTERFACE
        %SISEL==11,,126 ;SYSTEM ERROR LOGGER
        %SIDOL==12,,126 ;DISK AVR PROCESS
        %SITGH==13,,126 ;[SYSTEM]TGHA

.GTENQ==127             ;ENQ./DEQ. STATISTICS
        %EQLTL==1Ø,,127 ;MINUTES UNUSED LONG TERM LOCKS STAY AROUND

.GTDFL==14Ø             ;USER'S DEFAULTS
        JD.PRT==777B8   ;DEFAULT PROTECTION
        JD.SDP==1B9     ;SET IF USER SET DEFAULT PROTECTION
        JD.DAD==1B12    ;SET IF LOGIN SHOULDN'T ASK ABOUT DETACHED JOBS

.GTNTP==141             ;NETWORK PERFORMANCE ANALYSIS DATA
        %NTBYI==13,,141 ;NUMBER OF INPUT BYTES PROCESSED
        %NTBYO==14,,141 ;NUMBER OF OUTPUT BYTES PROCESSED

.GTSPA==142             ;SCHEDULER PERFORMANCE ANALYSIS DATA
        %SPDGS==Ø,,142  ;DTA GENERATED SLEEPS
        %SPMGS==1,,142  ;MTA GENERATED SLEEPS
        %SPEWS==2,,142  ;EVENT WAIT SATISFIED
        %SPTIS==3,,142  ;TTY INPUT SATISFIED
        %SPTOS==4,,142  ;TTY OUTPUT SATISFIED
        %SPPIS==5,,142  ;PTY INPUT SATISFIED
        %SPPOS==6,,142  ;PTY OUTPUT SATISFIED
        %SPRS1==7,,142  ;REQUEUES FROM SS INTO PQ1
        %SPRW1==1Ø,,142 ;REQUEUES FROM WAKE INTO PQ1
        %SPRD1==11,,142 ;REQUEUES FROM DAEMON SATISFIED INTO PQ1
        %SPRO1==12,,142 ;OTHER REQUEUES INTO PQ1
        %SPQR1==13,,142 ;PQ1 JOBS WHICH EXPIRED QUANTUM RUNTIME
        %SPQR2==14,,142 ;PQ2 JOBS WHICH EXPIRED QUANTUM RUNTIME
        %SPQRH==15,,142 ;HPQ JOBS WHICH EXPIRED QUANTUM RUNTIME
        %SPIP1==16,,142 ;PQ1 JOBS WHICH EXPIRED INCORE PROTECT TIME
        %SPIP2==17,,142 ;PQ2 JOBS WHICH EXPIRED INCORE PROTECT TIME
        %SPIPH==2Ø,,142 ;HPQ JOBS WHICH EXPIRED INCORE PROTECT TIME
        %SPKS1==21,,142 ;K SWAPPED IN FOR PQ1 JOBS
        %SPKS2==22,,142 ;K SWAPPED IN FOR PQ2 JOBS
        %SPKSH==23,,142 ;K SWAPPED IN FOR HPQ JOBS
        %SPNJ1==24,,142 ;NUMBER OF PQ1 JOBS SWAPPED IN
        %SPNJ2==25,,142 ;NUMBER OF PQ2 JOBS SWAPPED IN
        %SPNJH==26,,142 ;NUMBER OF HPQ JOBS SWAPPED IN
        %SPTC1==27,,142 ;TICS CHARGED TO PQ1
        %SPTC2==3Ø,,142 ;TICS CHARGED TO PQ2
        %SPTCH==31,,142 ;TICS CHARGED TO HPQ
        %SPNRS==32,,142 ;NUMBER OF RESPONSES FOR PQ1/CMQ SWAP IN
        %SPNTS==33,,142 ;TOTAL TICS OF RESPONSE FOR PQ1/CMQ SWAP IN
        %SPSSS==34,,142 ;SUM SQUARES OF PQ1/PQ2 SWAP IN (2 WORD INTEGER)
        %SPMWC==36,,142 ;NUMBER OF MEASUREMENTS OF WASTED CORE
        %SPSWC==37,,142 ;SUM OF WASTED CORE IN PAGES
        %SPSSC==4Ø,,142 ;SUM SQUARES OF WASTED CORE (2 WORD INTEGER)
```

Appendix D
MANUAL NUMBERS

Manual Name	SDC Order Number
TOPS-1Ø Monitor Calls, Volume 1	AA-D974D-TB
TOPS-1Ø Monitor Calls, Volume 2	AA-KØ39A-TB
TOPS-1Ø Commands Manual	AA-Ø916D-TB
TOPS-1Ø Operator's Guide	AA-H283A-TB
TOPS-1Ø Monitor Installation Guide	AA-5Ø56B-TB
TOPS-1Ø Network Software Installation Guide	AA-5156E-TB
TOPS-1Ø/2Ø RSX-2ØF System Reference Manual	AA-H352B-TK
TOPS-1Ø/2Ø SYSERR Manual	AA-D533B-TK
TOPS-1Ø Crash Analysis Guide	AA-H2Ø6B-TB

EDUCATIONAL SERVICES