File No. S370-36
Order No. SY20-0880-1

Systems

IBM Virtual Machine Facility/370: Control Program (CP) Program Logic

Release 1 PLC 9

This publication describes the internal logic of the VM/370 control program. Major sections discuss:

- The function of the control program
- The control program's modules
- The control program's data areas

Diagnostic information is also included.

This publication is intended for IBM personnel responsible for program maintenance.

Prerequisites for a thorough understanding and for the effective use of this publication are:

IBM System/360 Principles of Operation, GA22-6821
IBM System/370 Principles of Operation, GA22-7000
IBM System/360 Operating System: Assembler Language, GC28-6514

For titles and abstracts of other associated publications, see the publication *IBM System/360 and System/370 Bibliography,* GA22-6822

IBM

Second Edition (April 1973)

This edition, together with Technical Newsletter SN20-2624, dated August 15, 1973, applies to Release 1 PLC 9 (Program Level Change) of IBM Virtual Machine Facility/370 (VM/370) and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters.

Changes are continually made to the specifications herein; before using this publication in conjunction with the operation of IBM systems, consult the latest IBM System/360 and System/370 Bibliography, Order No. GA22-6822, and the IBM System/370 Advanced Function Bibliography, Order No. GC20-1763, for editions that are applicable and current.

Changes and additions are indicated by a vertical line to the left of the change.

Requests for copies of IBM publications should be made to your local IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, VM/370 Publications, 24 New England Executive Park, Burlington, Massachusetts, 01803. Comments become the property of IBM.

PREFACE

This Program Logic Manual (PLM) is a detailed guide to the VM/370 control program. It supplements the program listings by providing descriptive text, diagrams, and flowcharts. It is intended for IBM personnel responsible for program maintenance and is to be used with the following PLMs for maintaining the System Control Program (SCP).

IBM Virtual Machine Facility/370: Conversational Monitor System (CMS), Program Logic, SY20-0881

IBM Virtual Machine Facility/370: Service Routines Program Logic, SY20-0882

IBM System/360 Operating System: Assembler (F), GY26-3700

IBM CALL/360-OS BASIC System Manual, GY20-0530

The logic described in this publication is about programs that are discussed in the following publication:

IBM Virtual Machine Facility/370: Introduction, GC20-1800

IBM Virtual Machine Facility/370: Planning and System Generation Guide, GC20-1801

IBM Virtual Machine Facility/370: Operator's Guide, GC20-1806

IBM Virtual Machine Facility/370: System Messages, GC20-1808

IBM Virtual Machine Facility/370: Command Language Guide for General Users, GC20-1804

IBM Virtual Machine Facility/370: System Programmer's Guide, GC20-1807

Information in this publication (if any) about the following is for planning purposes only:

- IBM System/370 Model 165 II

In this publication, the term "3330 series" is used in reference to the IBM 3330 Disk Storage, Models 1 and 2 and the IBM 3333 Disk Storage and Control, Model 1.

MANUAL ORGANIZATION

This publication is divided into seven sections:

- The "Introduction" presents a general discussion about the functions and program organization of the VM/370 control program.
- The section "Method of Operation" contains a detailed discussion about the functions of the control program.
- "Program Organization" contains the flowcharts.
- The "Directories" contain an alphabetical list of all the CP labels that are discussed within this manual. Accompanying the label is a brief description of the function or for the subroutines, the name of the module that it is in, and cross-reference to its location in the PLM.
- The section "Data Areas" contains a detailed description of the control program data areas.
- "Diagnostic Aids" contains cross-reference information about commands, messages, wait codes, and ABEND codes.
- The Appendixes contain coding conventions, system equates, and DASD record formats.

Summary of Amendments for SY20-0880-1 VM/370 Release 1 PLC 5

Addition of the Following VM/370 Programming Functions

New Programming Features

- The Virtual = Real Performance option
- The Dedicated Channel Performance option
- The Virtual and Real Channel-to-Channel Adapter

Support for the Following Devices

New Hardware Features

- The IBM 3211 Printer
- The IBM 3410/3411 Magnetic Tape Subsystem
- The IBM System/370 Models 155 II and 158

Recovery Management Support

Maintenance: Program and Documentation

The section on Recovery Management Support has been rewritten to include the following changes and additions:

- A revised explanation of the initial state of the recovery mode for main storage errors.
- A revised termination procedure where recovery via an automatic restart is attempted before placing the system in a disabled wait state.
- The addition of a Buffer Error Routine as part of the Machine Check Handler to perform error recovery on those CPUs that have high speed buffers.

Additional Modules

New: Program and Documentation

The following modules have been added as part of the Control Program:

- The new module, DMKCPB, now simulates the SYSTEM, EXTERNAL, READY, NOTREADY, RESET, and REWIND commands for the virtual machine. This function has been removed from module DMKCFM.
- The new module, DMKTRC, now contains all the TRACE processing routines. The initialization, modification, and termination of the TRACE function remains in module DMKTRA.
- The new module, DMKCFG, contains the SAVESYS command processing routine. This function was removed from module DMKCPV.

Additional Data Area

New: Program and Documentation

The "Extended Outboard Recording Block" (XOBR3211) has been added as a continuation of the "I/O Error Information Block" (IOERBLOK). It will hold additional sense data for devices that return more than 24 sense bytes.

Additional and Revised Flowcharts

Maintenance: Program and Documentation

The flowcharts and module/entry point directory entries for the following modules have been revised to reflect the above-cited new features and support:

DMKCCH
DMKCCW
DMKCDS
DMKCFG
DMKCFM
DMKCFP
DMKCNS
DMKCPB
DMKCPI
DMKCPV
DMKCSO
DMKDAS
DMKDEF
DMKDGD
DMKDIA
DMKDMP
DMKDSP
DMKFRE
DMKGEN
DMKIOE
DMKIOF
DMKIOG
DMKIOS
DMKLDG
DMKMCH
DMKNEM
DMKPAG
DMKPRG
DMKPRV
DMKPSA
DMKRSP
DMKSCN
DMKSPL
DMKTAP
DMKTDK
DMKTRA
DMKTRC
DMKUSO
DMKVAT
DMKVCA
DMKVCH
DMKVCN
DMKVDB
DMKVDS
DMKVIO
DMKVMI
DMKVSP

Error Messages and Codes

Maintenance: Program and Documentation

The following Error Messages have been added:

DMKCCH605I
DMKCCH606I
DMKCFG044E
DMKCFG170E
DMKCFG171E
DMKCFG172E
DMKCFG173E
DMKCFG435E
DMKCFP174E
DMKCPB005E
DMKCPB006E
DMKCPB012E
DMKCPB022E
DMKCPB026E
DMKCPB040E
DMKCPI955W
DMKCPV144W
DMKCSO036E
DMKDAS956A
DMKDIA011E
DMKDMP909W
DMKMCH610I
DMKMCH611I
DMKMCH612W
DMKMCH614I
DMKMCH616I
DMKPAG415E
DMKPRG453W
DMKSPL501I
DMKSPL503A
DMKSPL504A
DMKSPL529I
DMKUDR475I
DMKVDB034E
DMKWRM911W

(See Over)

The following Error Messages have been deleted:

DMKCFP005E
DMKCFP006E
DMKCFP012E
DMKCFP022E
DMKCPV044E
DMKCPV170E
DMKCPV171E
DMKCPV172E
DMKCPV173E
DMKDIA110E
DMKMCH610W
DMKMCH611W
DMKMCH612I
DMKMCH614W
DMKMCH616W
DMKMCH620I
DMKSPL517I
DMKWRM910W

The following Wait state codes have been added:

00D
00F

The following ABEND codes have been added:

BLD001
CFM001
CNS008
DSP003
DSP004
FRE010
PTR008
PTR009
PTR010
SCH001
TRC001

The following ABEND code has been deleted:

TRA001

Miscellaneous

Maintenance: Documentation Only

This edition includes other minor technical and typographical changes too numerous to list.

Summary of Amendments
for SY20-0880-1
as updated by TNL SN20-2624
VM/370 Release 1 PLC 9

NEW DEVICE SUPPORT

New: Programming Feature

The IBM System/370 Model 168; the IBM 2860, 2870, 2880 standalone channels; and the IBM 2305 Fixed Head Storage, Model 1, are now supported.

USER ACCOUNTING OPTION

New: Programming Feature

It is now possible for a user to charge another user for CPU time. A new diagnose code (4C) is provided for this function. This option is described under "Accounting Card Processing" in the section on the "Real Spooling Manager." The new diagnose code is described in the section "Privileged Instructions."

VIRTUAL CONSOLE SPOOLING

New: Programming Feature

The virtual console is now supported for spooling operations. Documentation of this support appears in the section, "Virtual Spooling Manager."

CP INTERNAL TRACE TABLE IMPLEMENTATION

New: Programming Feature

A new CP command, MONITOR, allows the user to stop and restart the recording of real machine events in the internal trace tabel. Previously, the tracing was always active. This feature is described in the section, "CP Internal Trace Table."

STOP OPERAND AND PARAMETER PASSING FOR THE CP IPL COMMAND

New: Programming Feature

The STOP operand in the CP IPL command will halt execution and allow parameters to be passed resulting in the loading of an alternate nucleus. This function is described in "LOGON of User" in the section "System User Interface."

PERFORMANCE ENHANCEMENT

Maintenance: Program and Documentation

The Dispatcher/Scheduler routines have been modified to improve performance. The section on the Dispatcher/Scheduler has been rewritten.

ADDITIONAL MODULES

New: Program and Documentation

The following modules have been added as part of the Control Program:

- The modules DMKSIX, DMKSEV, and DMKEIG handle the channel logout analysis for the 2860, 2870, and 2880 standalone channels, respectively.

- The module, DMKGRA, handles VM/370 console spooling.

- The module, DMKLOC, locks and unlocks a system resource. This code was previously restricted to use in DMKUDR.

- The module, DMKMCC, handles the new CP command, MONITOR.

- The module, DMKRSE, retries and attempts recovery for real U/R device I/O errors. This function was originally in module DMKSPL.

ADDITIONAL DATA AREAS

New: Program and Documentation

ACCTBLOK -- User Accounting Block

New: Documentation Only

CCHREC -- Channel Check Handler Record
MCHAREA -- Machine Check Save Area
MCRECORD -- Machine Check Handler Record

The above blocks are defined in the "Data Areas -- Control Blocks" section.

ADDITIONAL AND REVISED FLOWCHARTS

New: Program and Documentation

DMKEIG
DMKGRA
DMKLOC
DMKMCC
DMKRSE
DMKSEV
DMKSIX

Maintenance: Program and Documentation

DMKACO
DMKCCH
DMKCDS
DMKCFP
DMSCFS
DMKCKP
DMKCPB
DMKCPI
DMKCPV
DMKCSP
DMKDEF
DMKDSO
DMKHVC
DMKIOE
DMKIOG
DMKIOS
DMKLNK
DMKLOG
DMKMCH
DMKPRG
DMKRSP
DMKSCH
DMKSPL
DMKTMR
DMKUDR
DMKUSO
DMKVCH
DMKVCN
DMKVDB
DMKVDS
DMKVMI
DMKVSP
DMKWRM

ERROR MESSAGES AND CODES

Maintenance: Program and Documentation

The following error messages have been added:

DMKCFP177E
DMKCPB059E
DMKMCC002E
DMKMCC026E
DMKMCH003E
DMKMCH026E
DMKRSE500I
DMKRSE501A
DMKRSE501I
DMKRSE502I
DMKRSE503A
DMKRSE503I
DMKRSE504A
DMKRSE504I
DMKRSE505A
DMKRSE508I
DMKRSE520A
DMKRSE520I
DMKRSE521I
DMKRSE524I
DMKRSE525I
DMKRSE529I

The following error messages have been deleted:

DMKSPL500I
DMKSPL501A
DMKSPL501I
DMKSPL502D
DMKSPL503A
DMKSPL503I
DMKSPL504A
DMKSPL504I
DMKSPL505D
DMKSPL508I
DMKSPL520I
DMKSPL521I
DMKSPL524I
DMKSPL525I
DMKSPL529I

The following ABEND codes have been added:

IOS001
IOS002
IOS003
UDR001

MISCELLANEOUS

Maintenance: Documentation Only

This Technical Newletter contains other minor technical and typographical changes, too numerous to mention.

CONTENTS

ILLUSTRATIONS

FIGURES

METHOD OF OPERATIONS DIAGRAMS

FLOWCHARTS

INTRODUCTION

The VM/370 Control Program (CP) manages the resources of a System/370 in order to provide virtual storage support through the implementation of virtual machines. This support is implemented in such a way that each terminal user appears to have the complete functional capabilities of a dedicated System/370 at his disposal, even though many other users may be running batch, teleprocessing, time sharing testing, or production jobs at the same time.

A user defines the configuration he requires -- input/output (I/O) device addresses, and a storage size up to 16 million bytes -- regardless of whether they match the real machine's configuration. Virtual devices must have real counterparts, but not always in a one-for-one ratio. For example, many users' readers, punches, and printers can be mapped onto common spool disks, and their virtual disk devices may be mapped as minidisks onto different sections of common disk packs, effectively multiplying the number of logical disk devices that are available on the real machine.

Each user's virtual machine comprises

- An operator's console (his remote terminal)
- A virtual CPU either with or without the Virtual Storage Addressing feature
- Virtual storage of up to 16 million bytes
- Virtual I/O devices

Virtual I/O devices are controlled by the virtual machine's operating system, not by the VM/370 control program. Thus, the support for the proper number and type of I/O devices must be provided by the operating system of the virtual machine for proper operation. The VM/370 control program monitors, translates, and schedules all real I/O operations to provide system integrity. It executes all virtual machine operation in a problem state by trapping, screening, and processing all the interrupts, and passing on the necessary information to the appropriate virtual machine. Only the VM/370 control program executes in the privileged state.

In order to increase the amount of real main storage available to user programs, parts of the VM/370 control program that are infrequently used are not required to be resident in main storage. Instead, they reside on part of the paging auxiliary storage used by the system, and are brought into main storage only when their functions are required.

Since the VM/370 Control Program nonresident modules are effectively paged into main storage, the control program itself must have virtual storage space associated with it. This space is anchored at the System VMBLOK, which is assembled into the resident control program in the module DMKSYS. The VMBLOK has a pointer to a segment table, which in turn references a set of page and swap tables that describe CP's virtual storage space.

The virtual space is divided into 2 parts; the first 4 segments (256K) is reserved for executable control program code, both resident and pageable; the remaining storage (at least another 256K) is dynamically allocated for spooling buffers and for user directory functions. In order for a routine to be pageable, a number of restrictions must be observed.

When the system is loaded, resolved, and written onto the system residence volume, those modules that are to be pageable must be loaded at addresses higher in main storage than the symbol DMKCPEND, which defines the last byte of the resident CP nucleus. This arrangement can be accomplished by reordering the LOADLIST EXEC used by the VMFLOAD procedure when punching out the text decks that will compose the CP system. Any pageable modules are listed after the entry for DMKCPE. In addition, each pageable module must be preceded by the 'SPB' loader control card. This 'Set Page Boundary' card forces the loader to start loading the succeeding module at the next higher 4k page boundary and ensures that the entire module will be resident when it is paged in.

If several pageable modules perform similar or related functions and it is felt that they are likely to be resident at the same time, they may be included in the same page by omitting the SPB cards that would normally have preceded the 2nd and subsequent modules. The group of modules to be loaded together must not exceed 4K as their total storage requirement; if they do, one or more must be loaded in separate pages, since no page boundary crossover in pageable control program is allowed. All currently pageable CP modues punch their own SPB card via an assembler PUNCH statement, except those that are designed to reside in a page along with other modules.

CP INITIALIZATION

The function of system initialization (IPL) is to prepare VM/370 for operation. Some of the tasks to be performed are:

- Main storage must be initialized
- Devices must be mounted
- Warm start records must be read from the warm start cylinder
- Space must be allocated for the system dump file
- The system operator must be logged on

In the case of a system restart following a failure, active files and the system log message must be written to the checkpoint cylinder before the Control Program nucleus can be brought into main storage. The user can now logon.

VIRTUAL MACHINE CONTROL

A virtual machine is created for a user when he logs into VM/370, on the basis of information stored in his user directory entry. The entry for each user identification includes a list of the virtual I/O devices associated with the particular virtual machine and the real device mappings.

Additional information about the virtual machine is maintained in the directory file. Included are the VM/370 command privilege class, accounting data, normal and maximum virtual storage sizes, and optional virtual machine characteristics such as extended control mode.

The Control Program supervises the execution of virtual machines by (1) permitting only problem state execution except in its own routines, and (2) receiving control after all real computing system interrupts. CP intercepts each privileged instruction and simulates it if the current program status word of the issuing virtual machine indicates a virtual supervisor state; if the virtual machine is executing in virtual problem state, the attempt to execute the privileged instruction is reflected back to the virtual machine as a program interrupt. All virtual machine interrupts (including those caused by attempting privileged instructions) are first handled by CP, and are reflected to the virtual machine if an analogous interrupt would have occurred on a real machine.

Virtual Machine Time Management

The real CPU is time sliced to simulate multiple virtual CPUs. Virtual machines that are executing in a conversational manner are given access to the real CPU more frequently than those that are not; these conversational machines are assigned the smaller of two possible time slices. CP determines execution characteristics of a virtual machine at the end of each time slice on the basis of the recent frequency of its console requests or terminal interrupts. The virtual machine is queued for subsequent CPU utilization according to whether it is a conversational or nonconversational user of system resources.

A virtual machine can gain control of the CPU only if it is not waiting for some activity or resource. The virtual machine itself may enter a virtual wait state after an I/O operation has begun. The virtual machine cannot gain control of the real CPU if it is waiting for a page of storage, if it is waiting for an I/O operation to be translated and started, or if it is waiting for a CP command to finish execution.

A virtual machine can be assigned a priority of execution. Priority is a parameter affecting the execution of a particular virtual machine as compared with other virtual machines that have the same general execution characteristics. Priority may be assigned by the real machine operator, but is more frequently a parameter of the virtual machine's directory entry.

Virtual Machine Storage Management

The normal and maximum storage sizes of a virtual machine are defined as part of the virtual machine configuration in the VM/370 directory. The virtual storage size can be temporarily redefined to any value that is a multiple of 4K and not greater than the maximum defined value. VM/370 implements this storage as virtual storage. The storage may appear as paged or nonpaged to the virtual machine, depending upon whether the extended control mode option has been specified for that virtual machine. This option is required if operating systems that control virtual storage, such as OS/VS1 or VM/370, are to be run in the virtual machine.

Storage in the virtual machine is logically divided into 4096 byte areas called pages. A complete set of segment and page tables is used to describe the storage of each virtual machine. These tables are maintained by CP and reflect the allocation of virtual storage pages to blocks of real storage. Virtual storage addressing is accomplished through use of these tables by the System/370 machine. Storage in the real machine is logically and physically divided into 4096 byte areas called page frames or blocks.

Only referenced virtual storage pages are kept in real storage, thus optimizing real storage utilization. Further, a page can be brought into any available page frame; the necessary relocation is done during program execution by a combination of VM/370 and dynamic address translation on the System/370. The active pages from all logged-in virtual machines and from the pageable routines of CP compete for available page frames. When the number of page frames available for allocation falls below a threshold value, CP determines which virtual storage pages currently allocated to real storage are relatively inactive and initiates suitable page-out operations for them.

Inactive pages are maintained on a direct access storage device. If an inactive page has been changed at some time during virtual machine execution, CP assigns it to a paging device, selecting the fastest such device with available space. If the page has not changed, it remains allocated in its original direct access location and is paged into real storage from there the next time the virtual machine references that page. A virtual machine program can use the DIAGNOSE instruction to communicate to CP that the information from specific pages of virtual storage is no longer needed; CP then releases the areas of the paging devices which had been assigned to hold the specified pages.

Paging is done on demand by CP. This means that a page of virtual storage is not read (paged) from the paging device to a real storage block until it is actually needed for virtual machine execution. No attempt is made by CP to anticipate what pages might be required by a virtual machine. While a paging operation is being performed for one virtual machine, another virtual machine can be executing. Any paging operation initiated by CP is transparent to the virtual machine.

If the virtual machine is executing in extended control mode with translate on, then two additional sets of segment and page tables are maintained. The virtual machine operating system is responsible for mapping the virtual storage created by it to the storage of the virtual machine. CP uses this set of tables in conjunction with the page and segment tables created for the virtual machine at login time to build shadow page tables for the virtual machine. These shadow tables map the virtual storage created by the virtual machine operating system to the storage of the real computing system. The tables created by the virtual machine operating system may describe any page and segment size permissible in the IBM System/370.

The system operator may assign the reserved page frames option to a single virtual machine. This option, specified by the SET RESERVE command, assigns a specific amount of the storage of the real machine to the virtual machine. CP dynamically builds a set of reserved real storage page frames for this virtual machine during its execution until the maximum number "reserved" has been reached. Since other virtual machines' pages are not allocated from this reserved set, the effect is that the most active pages of the selected virtual machine remains in real storage.

During the process of CP system generation, the installation may specify that a single virtual machine is to be given an option called virtual=real. With this option, the virtual machine's storage is allocated directly from real storage at the time CP is initially loaded, and remains so allocated unless released via operator command. All pages except page zero are allocated to the corresponding real storage locations. In order to control the real computing system, real page zero must be controlled by CP. Consequently, the real storage size must be large enough to accommodate the CP nucleus, the entire Virtual=Real virtual machine, and the remaining pageable storage requirements of CP and the other virtual machines.

The virtual=real option improves performance in the selected virtual machine since it removes the need for CP to perform paging operations for the selected virtual machine. The virtual=real option is necessary whenever programs that contain dynamically modified channel programs (excepting those of OS ISAM) are to execute under control of CP.

Virtual Machine I/O Management

A real disk device can be shared among multiple virtual machines. Virtual device sharing is specified in the directory entry or by a user command. If specified by the user an appropriate password must be supplied before gaining access to the virtual device. A particular virtual machine may be assigned read-only or read/write access to a shared disk device. CP verifies each virtual machine I/O operation against the parameters in the virtual machine configuration to ensure device integrity.

The virtual machine operating system is responsible for the operation of all virtual devices associated with it. These virtual devices may be defined in the directory entry of the virtual machine, or they may be attached to (or detached from) the virtual machine's configuration while it remains logged on. Virtual devices may be dedicated, as when mapped to a fully equivalent real device; shared, as when mapped to a minidisk or when specified as a shared virtual device; or spooled by CP to intermediate direct access storage.

In a real machine running under control of OS, I/O operations are normally initiated when a problem program requests OS to issue a START I/O instruction to a specific device. Device error recovery is handled by the operating system. In a virtual machine, OS can perform these same functions, but the device address specified and the storage locations referenced are both virtual. It is the responsibility of CP to translate the virtual specifications to real.

In addition, the interrupts caused by the I/O operation are reflected to the virtual machine for its interpretation and processing. If I/O errors occur, CP records them but does not initiate error recovery operations. These are the responsibility of the virtual machine operating system.

I/O operations initiated by CP for its own purposes (paging and spooling), are performed directly and are not subject to translation.

Spooling

A virtual unit record device, which is mapped directly to a real unit record device, is said to be dedicated. The real device is then controlled completely by the virtual machine's operating system.

CP facilities allow multiple virtual machines to share unit record devices. Since virtual machines controlled by CMS ordinarily have modest requirements for unit record I/O, such device sharing is quite advantageous, and it is the standard mode of system operation.

Spooling operations cease if the direct access storage space assigned to spooling has been exhausted, and the virtual unit record devices appear in a not ready status. The system operator may make additional spooling space available by purging existing spool files or by assigning additional direct access storage space to the spooling function.

Specific files can be transferred from the spooled card punch or printer of a virtual machine to the card reader of the same or another virtual machine. Files transferred between virtual unit record devices by the spooling routines are not physically punched or printed. With this method, files can be made available to multiple virtual machines, or to different operating systems executing at different times in the same virtual machine.

CP spooling includes many desirable options for the virtual machine user and the real machine operator. These options include printing multiple copies of a single spool file, backspacing any number of printer pages, and defining spooling classes for the scheduling of real output.

Console Functions

The CP console functions allow the user to control the virtual machine from the terminal, much as an operator controls a real machine. Virtual machine execution can be stopped at any time by use of the terminal's attention key; it can be restarted by typing in the appropriate CP command. External, attention, and device ready interrupts can be simulated on the virtual machine. Virtual storage and virtual machine registers can be inspected and modified, as can status words such as the PSW and the CSW. Extensive trace facilities are provided for the virtual machine, as well as a single-instruction mode. Commands are available to invoke the spooling and disk sharing functions of CP.

Console functions are divided into privilege classes. The directory entry for each user assigns one or more privilege classes. The classes are:

- System operator
- Operator
- System programmer
- Spooling operator
- Systems analysts
- Customer engineering
- General users

Commands in the system analysts class may be used to inspect real storage locations, but may not be used to make modifications to real storage. Commands in the operator class provide real resource control capabilities. System operator commands include all those relating to virtual machine performance options, such as assigning a set of reserved page frames to a selected virtual machine. See the "CP Commands" sections of this chapter for more information.

PROGRAM STATES

When instructions in the Control Program are being executed, the real computer is in the supervisor state; at all other times, when running virtual machines, it is in the problem state. Therefore, privileged instructions can only be executed by the Control Program. Programs running on a virtual computer can issue privileged instructions; such an instruction causes an interruption that is handled by the Control Program. CP examines the operating status of the virtual machine PSW. If the virtual machine indicates that it is functioning in supervisor mode, then the privileged instruction is simulated according to its type. If the virtual machine is in problem mode, then the privileged interrupt is reflected to the virtual machine.

Only the Control Program may operate in the supervisor state on the real machine. All programs other than CP operate in the problem state on the real machine. All user interrupts, including those caused by attempted privileged operations, are handled by CP, which then reflects to the user program only those interrupts that the user program would expect from a real machine. A problem program executes on the virtual machine in a manner identical to its execution on a real System/370 CPU, as long as it does not violate the CP restrictions.

PREFERRED VIRTUAL MACHINE

CP supports four special virtual machine operating environment functions. Each function can be applied to one virtual machine at a time. Although each function could be applied to a different virtual machine, optimum performance would not be achieved. Each function is discussed separately following.

FAVORED EXECUTION

CP attempts to provide a specified percentage of CPU time to a particular virtual machine. CP attempts to provide up to the specified percentage of CPU time to a particular virtual machine, provided that the virtual machine is functioning so that it can fully utilize the CPU time. At regular time intervals the CP dispatcher checks the CPU time used by the particular virtual machine. If the specified percentage is exceeded, the machine becomes the lowest priority user in the system. If the percentage used is lower than that specified, the virtual machine has highest priority execution for the remainder of the interval. The percentage of CPU time assured is specified in the privileged class command that invokes the function.

CP can also assure that a designated user will never be dropped from the active (in queue) subset by the scheduler. When the user is runnable, he is placed in the dispatchable list at his normal priority.

RESERVED PAGE FRAMES

CP uses chained lists of table entries for available and pageable pages. Pages for users are assigned from the available lists which is replenished from the pageable list.

Pages which are temporarily locked in real storage are not available or pageable. Paging proceeds using demand paging with a "reference bit" algorithm to select the best page for swapping. The reserved page frames option gives a particular virtual machine an essentially "private" set of pages. The pages are not locked, that is, they can be swapped, but usually only for the specified virtual machine. The number of reserved pages for the virtual machine are specified as a maximum. The page selection routine will select an available page for a reserved user and mark that page "reserved" if the maximum specified for the user has nct been reached. If an available, unreferenced "reserved" page is encountered during page replenishment for the reserved user, it is used whether or not the maximum has been reached. If the page selection routine cannot locate an available page for other users because they are all "reserved", the routine may have to steal the reserved pages.

DEDICATED CHANNELS

Since the devices on a channel are often shared between virtual machines (minidisks and dedicated devices) and shared with system functions (paging and spooling), CP schedules all the I/O requests to achieve a balance between machines. In addition, CP simulates the reflection of the subsequent I/O interrupts to the virtual machines. By specifying a dedicated channel(s) for a virtual machine, the CP channel scheduling function is bypassed. The virtual device addresses on the dedicated channel must match the real device addresses. Since the channels are dedicated, CP uses the virtual machine masking to control the real channel masking. I/O interrupts from the dedicated channel are presented in the order of occurrence using a single element stack and the real channel masking.

A single virtual machine may have multiple dedicated channels. Also, multiple virtual machines may each have a separate dedicated channel.

VIRTUAL=REAL

This feature requires that the CP nucleus be reorganized to provide a "hole" in real storage large enough to contain the entire storage area of the virtual machine. For the virtual machine, each page from page 1 to the last page (n) is in its true real storage location; only page zero is relocated. The virtual machine is still run in relocate mode, but since the virtual page address is the same as the real page address, no CCW translation is required for the virtual machine. Since no CCW translation is performed, no check is made of the I/O data addresses. The virtual machine must ensure that no I/O data transfer is specified into page zero or into any page not in the virtual machine's domain.

There are several considerations for the virtual=real option of preferred machine support that affect overall system operation:

- The area of contiguous storage built for the virtual=real machine must be large enough to contain the entire addressing space of that machine.

- While allocated as such, the storage reserved for the virtual=real machine can only be used by a virtual machine with that option. It is not available to other users for paging space nor for VM/370 usage, even when the virtual=real machine is not logged on. For this reason, it is expected that the virtual=real machine will be a high availability, high throughput machine.

  The virtual=real storage can be released by the operator. That storage is then available for paging. Once virtual=real storage space is released by the operator, a VM/370 IPL is necessary to again allocate that storage to that virtual=real machine.

- The virtual machine with the virtual=real option operates in the pre-allocated storage area with normal CCW translation in effect until the execution of the SET NOTRANS ON command. At that time, all subsequent I/O operations are performed from the virtual CCWs in the virtual=real space without translation. In this mode, the virtual machine must not perform I/O operations into page zero nor beyond its addressable limit. Violation of this requirement causes destruction of the VM/370 system and/or other virtual machines.

- If the virtual=real machine performs a virtual reset or IPL, then the normal CCW translation is performed until the issuance of the SET NOTRANS ON command. Only the virtual=real virtual machine can issue the command. A message is issued if normal translation mode is entered.

CP INTERRUPTION HANDLING

I/O INTERRUPT

I/O interrupts from completed I/O operations initiate various completion routines and the scheduling of further I/O requests. The I/O interrupt handling routine also gathers device sense information.

PROGRAM INTERRUPT

Program interrupts can occur in two states. If the CPU is in supervisor state, the interrupt indicates a system failure in the CP nucleus and causes a system abend. If the CPU is in problem state, then a virtual machine is executing. If the program interrupt indicates that the Dynamic Address Translation (DAT) feature has an exception, a virtual machine issued a privileged instruction, or a protection exception occurred for a shared segment system, then CP takes control to perform any required processing to satisfy the exception. Usually, the interrupt is transparent to the virtual machine execution. Most other program interrupts result from virtual machine processing and are reflected to the machine for handling. For a complete discussion of this subject, see the appropriate explanation in the section "Method of Operation".

MACHINE CHECK INTERRUPT

When a machine check occurs, the CP Recovery Management Support (RMS) gains control to save data associated with the failure for FE maintenance. RMS analyzes the failure to determine the extent of damage.

Damage assessment results in one of the following actions being taken:

- System Termination
- Selective Virtual User Termination
- Refreshing of damaged information with no affect on system configuration
- Refreshing of damaged information with the defective storage page removed from further systems use
- Error recording only for certain soft machine checks

The system operator is informed of all actions taken by the RMS routines. When a machine check occurs during VM/370 startup (before the system is set up well enough to permit RMS to operate successfully), the CPU goes into a disabled wait state and places a completion code of X'00B' in the high-order bytes of the current PSW.

SVC INTERRUPT

When an SVC interrupt occurs, the SVC interrupt routine is entered. If the machine is in problem mode, the type of interrupt is reflected back to the pseudo-supervisor (that is, the supervisor operating in the user's virtual machine). If the machine is in supervisor mode, the SVC interrupt code is determined, and a branch is taken to the appropriate SVC interrupt handler.

EXTERNAL INTERRUPT

If a timer interrupt occurs, CP processes it according to type. The interval timer indicates time-slice end for the running user. The clock comparator indicates that a specified timer event has occurred, such as midnight, scheduled shutdown, or user event reached. The CPU timer indicates that a virtual machine's allowed execution interval (time in queue) has expired.

The external console interrupt invokes CP processing to switch from the 3210 or 3215 to an alternate operator's console.

FREE STORAGE MANAGEMENT

During its execution, CP occasionally requires small blocks of storage that are used for the duration of a task. This storage is obtained from the free storage area. The free storage area is divided into various size subpools. The requestor informs the Free Storage Manager the size of the block required and the smallest available subpool that fulfills the request is allocated to the requestor. When the block is no longer needed, the requestor informs the Free Storage Manager and the block is returned to free storage.

If the request for free storage cannot be fulfilled the Free Storage Manager requests the temporary use of a page of storage from the Dynamic Paging Area. If a page is obtained, then the page is chained to the free storage area and used for that purpose until it is no longer needed and subsequently returned to the Dynamic Paging Area.

If the request for a page cannot be fulfilled, the requestor waits until free storage becomes available.

EXECUTING THE PAGEABLE CONTROL PROGRAM

Calls to pageable routines are recognized at execution time by the SVC 8 linkage manager in DMKPSA. For every SVC 8, the called address (in the caller's GPR15) is tested to see if it is within the resident nucleus. If it is less than DMKCPEND and greater than DMKSLC, the called routine's base address is placed in GPR12 and control is passed to the called routine in the normal way. However, if the called address is above DMKCPEND or below DMKSLC, the linkage manager issues a TRANS macro, requesting the paging manager to locate and, if necessary, page-in the called routine. The TRANS is issued with LOCK option. Thus, the lock count associated with the called routine's real page indicates the responsibility count of the module.

- When the module is called, the count is incremented.

- When the routine exits via SVC 12, the count is decremented.

When the count reaches zero, the pageable routine is unlocked and is eligible to be paged out of the system. However, since all CP pageable modules are reentrant, the page is never swapped out, but when stolen is placed directly on the free page list.

Since unlocked pageable routines participate in the paging process in a manner similar to user virtual storage pages, the Least Recently Used approximation used by page selection tends to make highly used control program routines, even when not locked, remain resident. The called routine is locked into real storage until it exits. Thus, it can request asynchronously scheduled function, such as I/O or Timer interrupts, as long as it dynamically establishes the interrupt return address for the requested operation and does not give up control via an EXIT macro prior to receiving the requested interrupt.

Addressability for the module while it is executing is guaranteed since the CALL linkage loads the real address of the paged module into GPR12 (the module base register) prior to passing control. If all addressing is done in a base/displacement form, the fact that the module is executing at an address different from that at which it was loaded is transparent. Although part of the control program is pageable it never runs in relocate mode. Thus, the CPU is not degraded by the DAT feature being active, and there is no problem of handling disabled page-faults.

SYSTEM SUPPORT MODULES

The system support modules provide CP several common functions in the area of data conversion and control block scanning and verification. Since most of the routines operate at the lowest level of control, they are linked to via the BALR option of the CALL macro, and make use of the BALRSAVE and TEMPSAVE workareas in DMKPSA. Two exceptions to this are the virtual and real I/O control block scan routines DMKSCNVU and DMKSCNRU. These routines do not alter the contents of the BALRSAVE area, and hence may be called by another low level BALR routine.

CONTROL REGISTER USAGE

Every IBM System/370 CPU provides the program with 16 logical control registers (logical registers since the number that are active depends on the features installed in the machine at any one time) that are addressable for loading and storing from BC mode. VM/370 provides only a single control register, control register zero, for normal virtual machines, for processing systems that do not require the full set of registers (for example, CMS, DOS, or other operating systems for System/360.

Any user whose virtual machine operating system requires the use of control registers other than control register zero can request the full set of 16 registers by specifying the ECMODE option in the VM/370 user-directory entry for his virtual machine. Specifying this option does not imply that the virtual machine will encounter any of the additional overhead associated with use of the Extended Control mode but permits the use of all 16 control registers from either BC or EC mode.

A virtual machine, which utilizes any System/370 features that use the control registers, requires the ECMODE option. Some of these features are expanded timer support of the System/370, (CPU timer, clock comparator, etc.), the virtual relocate-mode and its instructions, RRB, LRA, PTLB, virtual monitor calls, virtual Program Event Recording (PER), etc.

RESTRICTIONS AND CONVENTIONS FOR PAGEABLE CP MODULES

CP modules that are to be pageable must observe the following restrictions and conventions when they are designed and coded:

1. The module should be completely reentrant. Any messages to be modified, temporary work or scratch areas, or program switches must be allocated from system free storage or from the caller's save area.

2. The module must be entered via the standard SVC 8 CALL linkage. Modules entered via BALR or GOTO cannot be pageable.

3. The module cannot contain any A or V type address constants that point to locations within itself or within other pageable modules, and it cannot contain any CCWs that contain data address within itself. The only exceptions are address constant literals generated as the result of CALLs to other modules (since these addresses are dynamically relocated at execution time, they must be resolved by the loader to the loaded address of the called module) and a pageable module that locks itself into storage. In practice, this restriction simply means that data or instructions within the pageable routine must be referenced via base/displacement addressing, and the address in register 15 for a CALL may not be generated via a LOAD ADDRESS.

4. The pageable module must be no more than 4096 bytes in length.

If the above design and coding restrictions are adhered to, the CP module can be added to the existing pageable nucleus modules by utilizing the service routine, VMFLOAD, which is described in the "VM/370 Maintenance Procedures" chapter of the publication _IBM Virtual Machine Facility/370: Service Routines Program Logic_, Order No. SY20-0882. Additional information can be found in "Appendix I" of the publication _IBM Virtual Machine Facility/370: Planning and System Generation Guide_, Order No. GC20-1801.

MODULES

Executable Resident Modules

DMKCCH
DMKCCW
DMKCFM
DMKCNS
DMKCVT
DMKDAS
DMKDGD
DMKDMP
DMKDSP
DMKFRE
DMKGEN
DMKHVC
DMKIOE
DMKIOS
DMKMCH
DMKMSW
DMKPAG
DMKPGS
DMKPGT
DMKPRG
DMKPRV
DMKPSA
DMKPTR
DMKQCN
DMKRPA
DMKRSP
DMKSCH
DMKSCN
DMKSTK
DMKTMR
DMKUNT
DMKVAT
DMKVCN
DMKVIO
DMKVSP

Executable Pageable Modules

DMKACO
DMKBLD
DMKCDB
DMKCDS
DMKCFD
DMKCFG
DMKCFP
DMKCFS
DMKCFT
DMKCKP
DMKCPB
DMKCPI
DMKCPV
DMKCQG
DMKCQP
DMKCSO
DMKCSP
DMKCSU
DMKDEF
DMKDIA
DMKDRD
DMKEIG
DMKERM
DMKGRA
DMKIOF
DMKIOG
DMKISM
DMKLNK
DMKLOC
DMKLOG
DMKMCC
DMKMID
DMKMSG
DMKNEM
DMKRSE
DMKSAV
DMKSEP
DMKSEV
DMKSIX
DMKSPL
DMKTAP
DMKTDK
DMKTRA
DMKTRC
DMKTRM
DMKUDR
DMKUSO
DMKVCA
DMKVCH
DMKVDB
DMKVDS
DMKVMI
DMKWRM

Data Area Modules

In addition to the executable resident and pageable modules there are certain modules that only contain data areas and do not execute. These modules are:

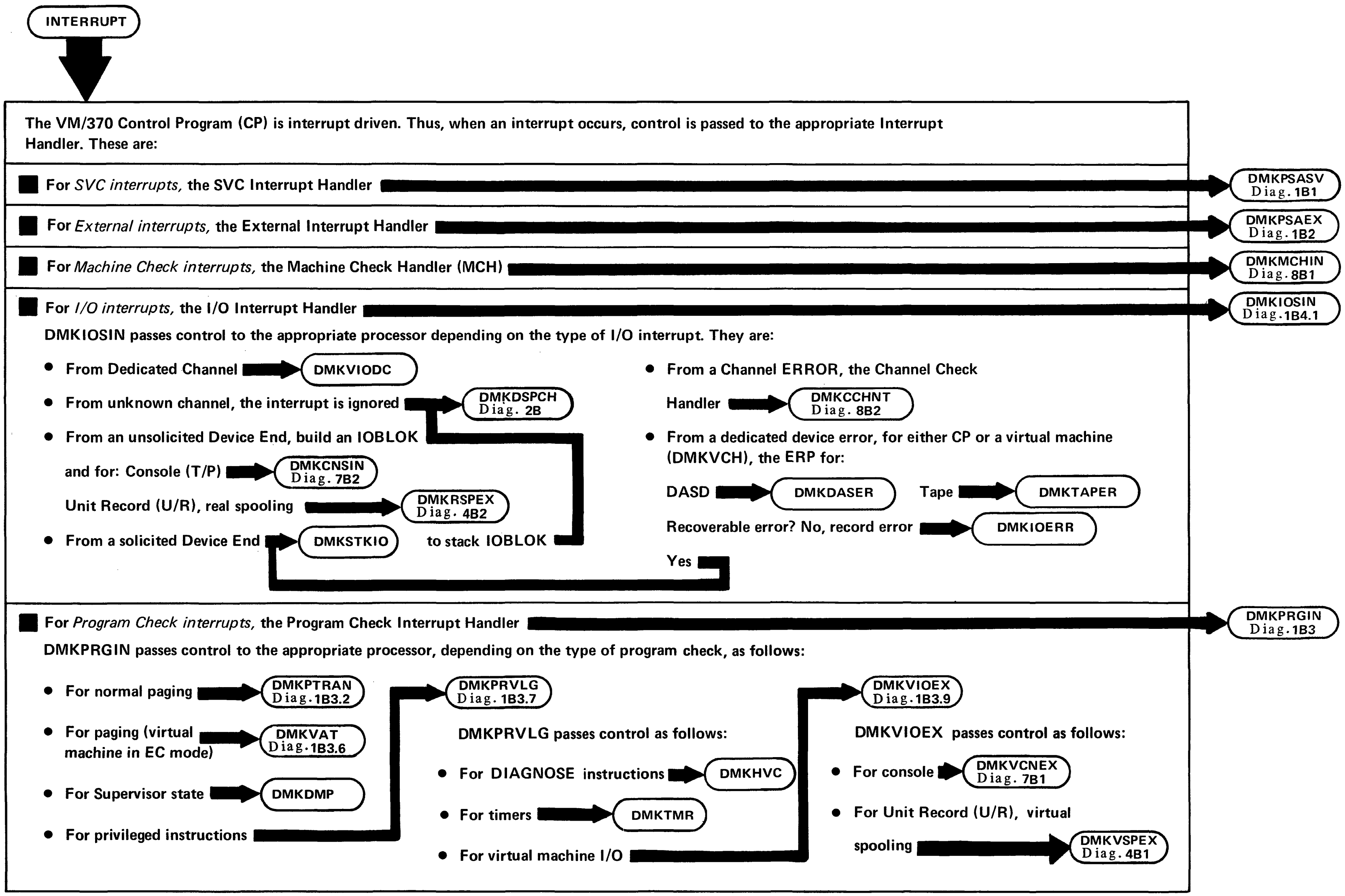
Resident Module	Contents
DMKCPE	Defines the end of the CP nucleus
DMKRIO	I/O device blocks
DMKSYS	System constants
DMKTBL	Terminal translate table

Pageable Module	Contents
DMKBOX	Output separator table
DMKFCB	3211 Forms control Buffer (FCB) load tables
DMKSNT	System name table
DMKSYM	System symbol table
DMKUCB	3211 Universal Character Set Buffer (UCSB) load tables
DMKUCS	1403 Universal Character Set (UCS) load tables

The data areas within these modules are discussed throughout this publication; most are illustrated in the section "Data Areas".

METHOD OF OPERATION

Diag. 1A. Overview of Method of Operation Diagrams



INTERRUPT

The VM/370 Control Program (CP) is interrupt driven. Thus, when an interrupt occurs, control is passed to the appropriate Interrupt Handler. These are:

■ For *SVC interrupts,* the SVC Interrupt Handler → DMKPSASV Diag. 1B1

■ For *External interrupts,* the External Interrupt Handler → DMKPSAEX Diag. 1B2

■ For *Machine Check interrupts,* the Machine Check Handler (MCH) → DMKMCHIN Diag. 8B1

■ For *I/O interrupts,* the I/O Interrupt Handler → DMKIOSIN Diag. 1B4.1

DMKIOSIN passes control to the appropriate processor depending on the type of I/O interrupt. They are:

- From Dedicated Channel → DMKVIODC
- From unknown channel, the interrupt is ignored → DMKDSPCH Diag. 2B
- From an unsolicited Device End, build an IOBLOK

  and for: Console (T/P) → DMKCNSIN Diag. 7B2

  Unit Record (U/R), real spooling → DMKRSPEX Diag. 4B2
- From a solicited Device End → DMKSTKIO to stack IOBLOK
- From a Channel ERROR, the Channel Check Handler → DMKCCHNT Diag. 8B2
- From a dedicated device error, for either CP or a virtual machine (DMKVCH), the ERP for:

  DASD → DMKDASER Tape → DMKTAPER

  Recoverable error? No, record error → DMKIOERR

  Yes

■ For *Program Check interrupts,* the Program Check Interrupt Handler → DMKPRGIN Diag. 1B3

DMKPRGIN passes control to the appropriate processor, depending on the type of program check, as follows:

- For normal paging → DMKPTRAN Diag. 1B3.2
- For paging (virtual machine in EC mode) → DMKVAT Diag. 1B3.6
- For Supervisor state → DMKDMP
- For privileged instructions → DMKPRVLG Diag. 1B3.7

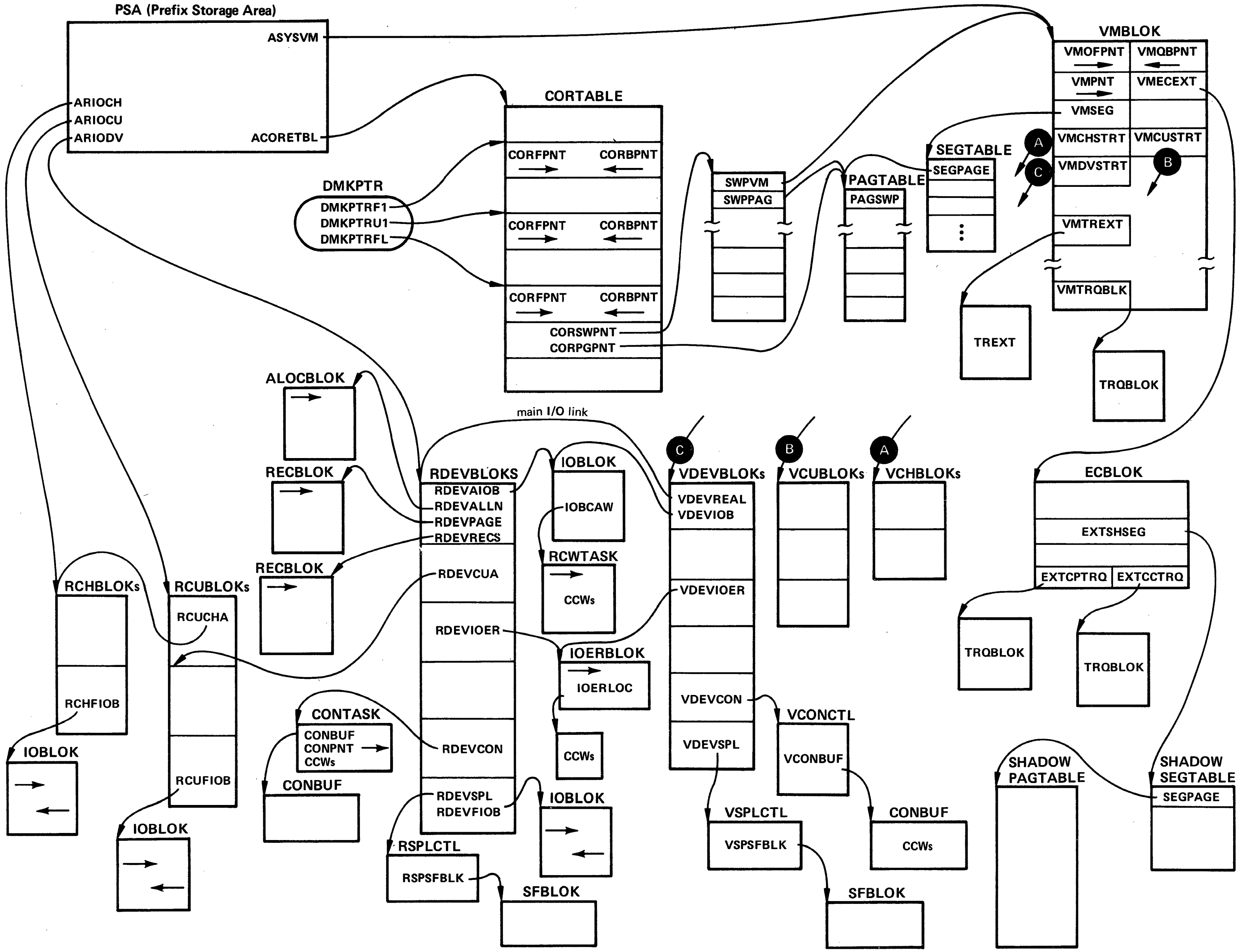
DMKPRVLG passes control as follows:

- For DIAGNOSE instructions → DMKHVC
- For timers → DMKTMR
- For virtual machine I/O → DMKVIOEX Diag. 1B3.9

DMKVIOEX passes control as follows:

- For console → DMKVCNEX Diag. 7B1
- For Unit Record (U/R), virtual spooling → DMKVSPEX Diag. 4B1

Diag. 1A0. CP Control Block Relationships



SVC INTERRUPTIONS

When an SVC interruption occurs, the SVC interruption routine (DMKPSASV) is entered. If the machine is in problem mode, DMKPSASV takes the following action:

- The SVC interrupt code is examined to determine if the interrupt was the result of an ADSTOP SVC code X'B3'. If it was, the message ADSTOP AT XXXXX is sent to the user's terminal, the overlaid instruction is replaced, and the virtual machine is placed in console function mode via DMKCFMBK; otherwise, the virtual machine's mode (BC or EC) is determined.

- If the virtual machine was in EC mode or its page 0 was not in real storage, then all general and floating-point registers are saved, the user's VMBLOK is flagged as being in an instruction wait, and control is transferred (via GOTO) to DMKPRGRF to reflect the interrupt to the virtual machine.

- If the virtual machine was in BC mode and if his page 0 is in main storage, then an appropriate SVC old PSW is stored in his page 0 and the interrupt is reflected to the virtual machine, bypassing unnecessary register saving. If the new virtual PSW indicates the wait state, all registers are saved in the VMBLOK and control transfers to DMKDSPB for PSW validation.

If the machine is in supervisor mode, the SVC interruption code is determined and a branch is taken to the appropriate SVC interruption handler.

SVC 0: Impossible condition or fatal error. The SVCDIE routine initiates an ABEND by going to the DMKDMPDK routine.

SVC 4: Reserved for IBM use.

SVC 8: Link request (transfer control from calling routine to called routine specified by register 15). The SVCLINK routine sets up a new save area, and then saves the caller's addressability (register 12) and save-area address (register 13), and the return-address (from the SVCOPSW) in the new save area. If the called routine (specified by register 15) is within the resident CP nucleus, SVCLINK places its address in register 12 and branches directly to the called routine. If the called routine is in a pageable module, a TRANS is performed on register 12 to ensure that the page containing the called routine is in storage. Upon return from the TRANS, the real address of the pageable routine is placed in register 12 and SVCLINK branches to the called routine. The real storage location of DMKCPE is the end of the resident CP nucleus. Any modules loaded at a higher real storage address are defined as pageable modules.

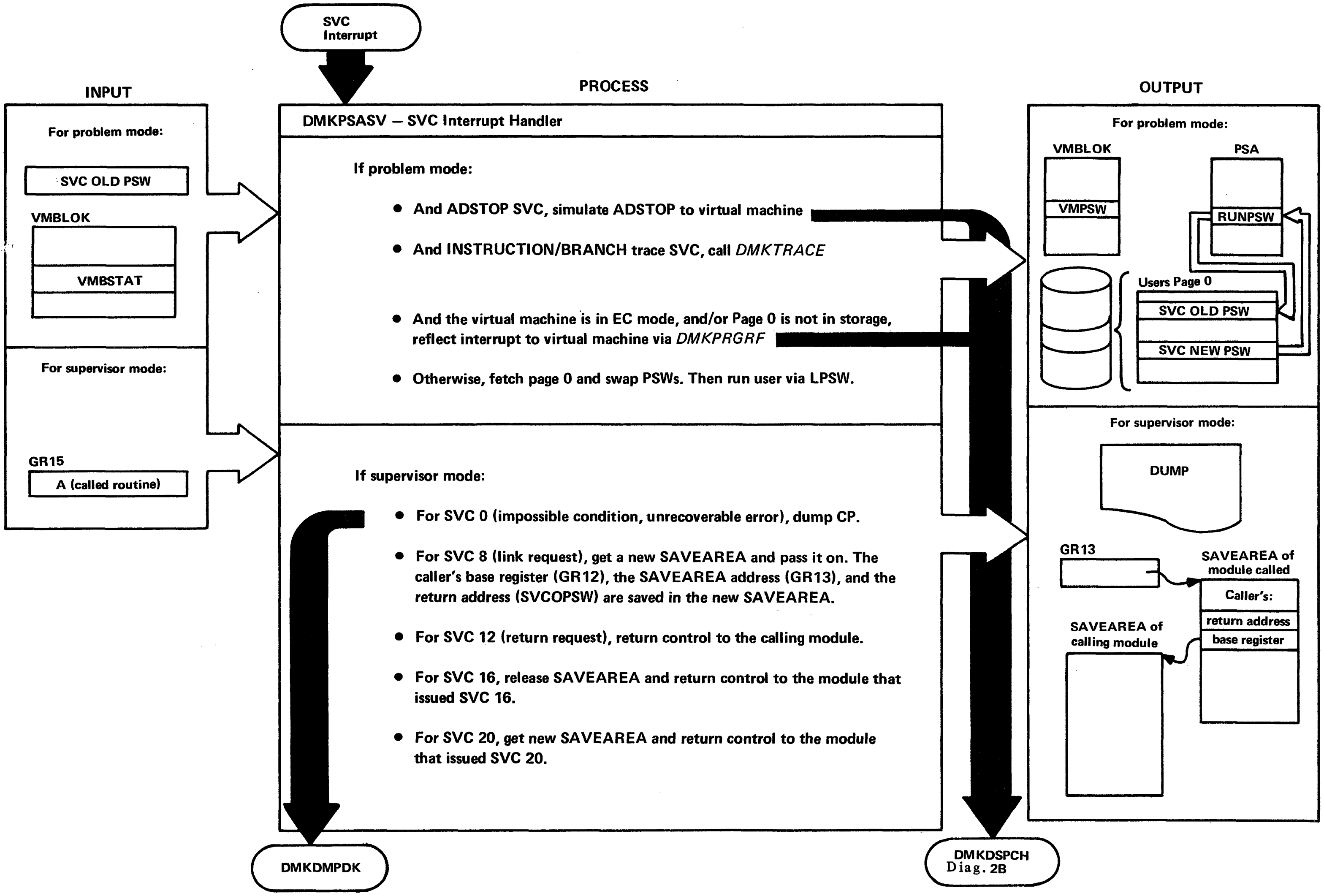
SVC 12: Return request (transfer control from called routine to calling routine). The SVCRET routine is invoked. If the routine which issued the SVC 12 is in the pageable module DMKPTRUL, then DMKPGSUL is called to unlock the page. SVCRET then restores registers 12 and 13 (addressability and save area address saved by SVCLINK), places the user's return address (also saved in the area) back into the SVCOPSW, and returns control to the calling routine by loading the SVCOPSW.

SVC 16: Release current save area from the active chain (remove linkage pointers to the calling routine). The SVCRLSE routine releases the current save area by placing the address of the next higher save area in register 13 and returns control to the current routine by loading the SVCOPSW. This SVC is used by second level interrupt handlers to bypass returning to the first level handler under specific circumstances. The base address field (register 12) in the save area being released is examined to determine if the bypassed routine is in a pageable module. If so, DMKPTRUL is called to unlock the page.

SVC 20: Obtain a new save area. The SVCGET routine places the address of the next available save area in register 13 and the address of the previous save area in the save area pointer field of the current save area.

There are 35 SAVEAREAs initially set up by DMKCPINT for use by the SVC linkage handlers. If the supply of available save areas drops to zero, the linkage handlers calls DMKFREE to obtain storage for additional save areas.

Diag. 1B1. SVC Interrupt Handler



DMKPSASV – SVC Interrupt Handler

If problem mode:

- And ADSTOP SVC, simulate ADSTOP to virtual machine
- And INSTRUCTION/BRANCH trace SVC, call *DMKTRACE*
- And the virtual machine is in EC mode, and/or Page 0 is not in storage, reflect interrupt to virtual machine via *DMKPRGRF*
- Otherwise, fetch page 0 and swap PSWs. Then run user via LPSW.

If supervisor mode:

- For SVC 0 (impossible condition, unrecoverable error), dump CP.
- For SVC 8 (link request), get a new SAVEAREA and pass it on. The caller's base register (GR12), the SAVEAREA address (GR13), and the return address (SVCOPSW) are saved in the new SAVEAREA.
- For SVC 12 (return request), return control to the calling module.
- For SVC 16, release SAVEAREA and return control to the module that issued SVC 16.
- For SVC 20, get new SAVEAREA and return control to the module that issued SVC 20.

EXTERNAL INTERRUPTIONS

When an external interruption occurs, the external interruption handler (DMKPSAEX) is entered.

TIMER INTERRUPT

If DMKPSAEX is entered because of a timer interrupt, the machine mode must be determined. If the machine was in WAIT state, control is transferred to DMKDSPCH which becomes idle until another interrupt occurs. If the machine is in problem mode, the address of the current user's VMBLOK is obtained from RUNUSER. The user's current PSW (VMPSW) is updated from the external interruption old PSW, the address of the current VMBLOK is placed in register 11, and control is transferred to DMKDSPCH. For additional information about timers see the section "Virtual Timer Maintenance".

EXTERNAL INTERRUPT

If DMKPSAEX is entered because of the operation of the console interrupt button (INTERRUPT), the following steps are taken:

1. The current system operator's VMBLOK (DMKSYSOP) is referenced.

2. His virtual machine is disconnected.

The operator can now logon from another terminal. The operation of the console interrupt button is used to implement an alternate operator's console. For a description of the processing of the EXTERNAL command refer to module DMKCPB.

PROGRAM INTERRUPTIONS

When a program interruption occurs, the program interruption handler (DMKPRGIN) is entered. Program interruptions can result from:

- Normal paging requests.
- A paging request by a virtual machine in EC mode (virtual relocation).
- Privileged instructions.
- Program errors.

DMKPRGIN determines the cause of the interruption by examining the interruption code.

NORMAL PAGING REQUESTS

If the program interrupt is caused by a normal paging request (it is not from a virtual machine that is running in EC mode with translation on), DMKPRGIN determines whether a segmentation error (a segment of the program occurred; if so, an invalid address interruption code is set, and the interruption is reflected to the user's virtual machine supervisor. If a segmentation error has not occurred, the user's current PSW is updated from the program old PSW (PROPSW), the address of the current VMBLOK is placed in register 11, and DMKPTRAN is called to obtain the required page. When the paging operation is completed, control is returned to DMKDSPCH.

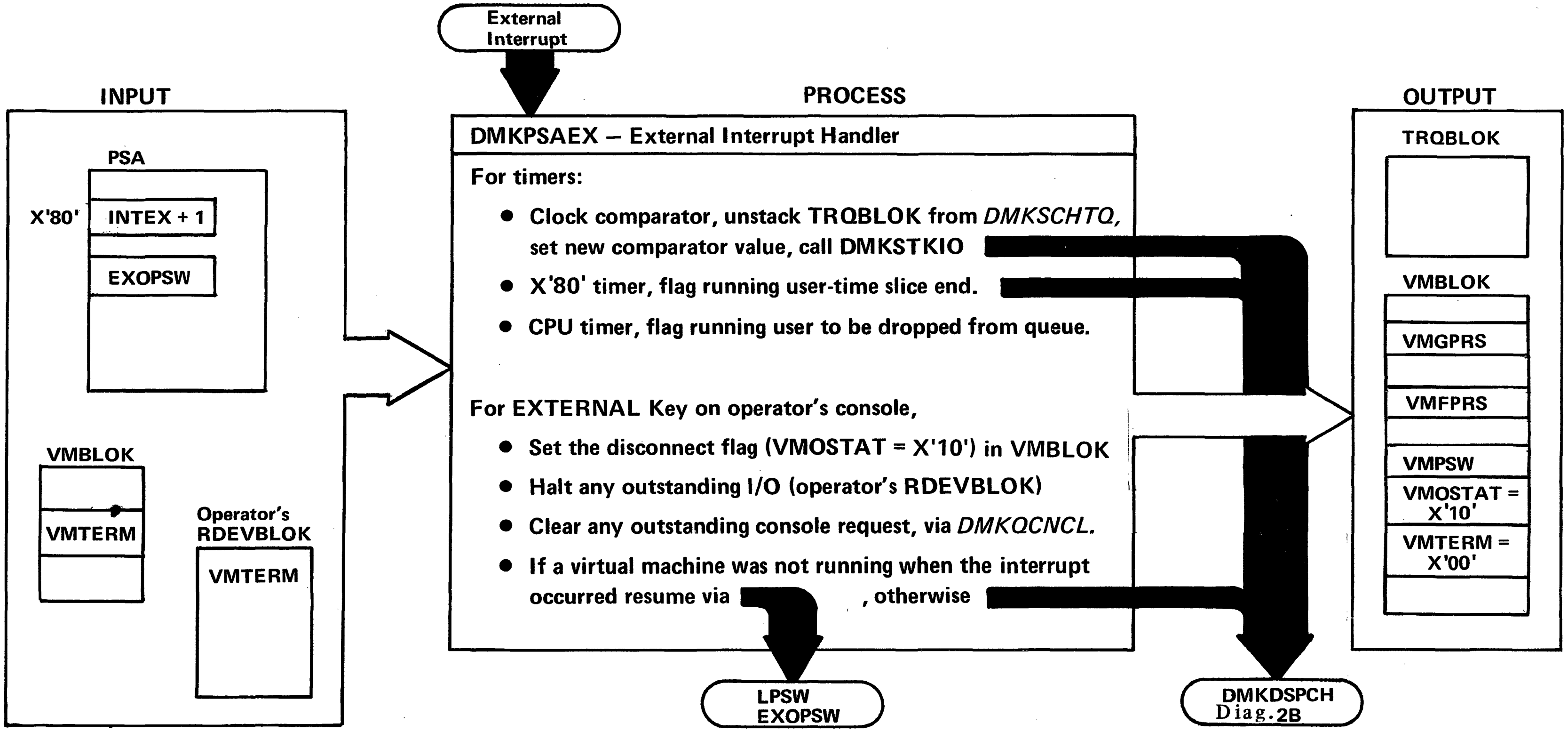
The functions of paging are divided into three categories: the management of virtual storage, the management of real storage, and the management of auxiliary storage (DASD paging devices).

Virtual Storage Management

When operating in the relocate environment provided by CP, each user's virtual storage space is described by two sets of tables.

- One set, the segment and page tables, describes the location and availability of any of the user's virtual pages that may be resident in real storage. Locations in these tables are indexable by virtual address, and the entries contain index values that reference corresponding real storage addresses. In

Diag. 1B2. External Interrupt Handler



DMKPSAEX – External Interrupt Handler

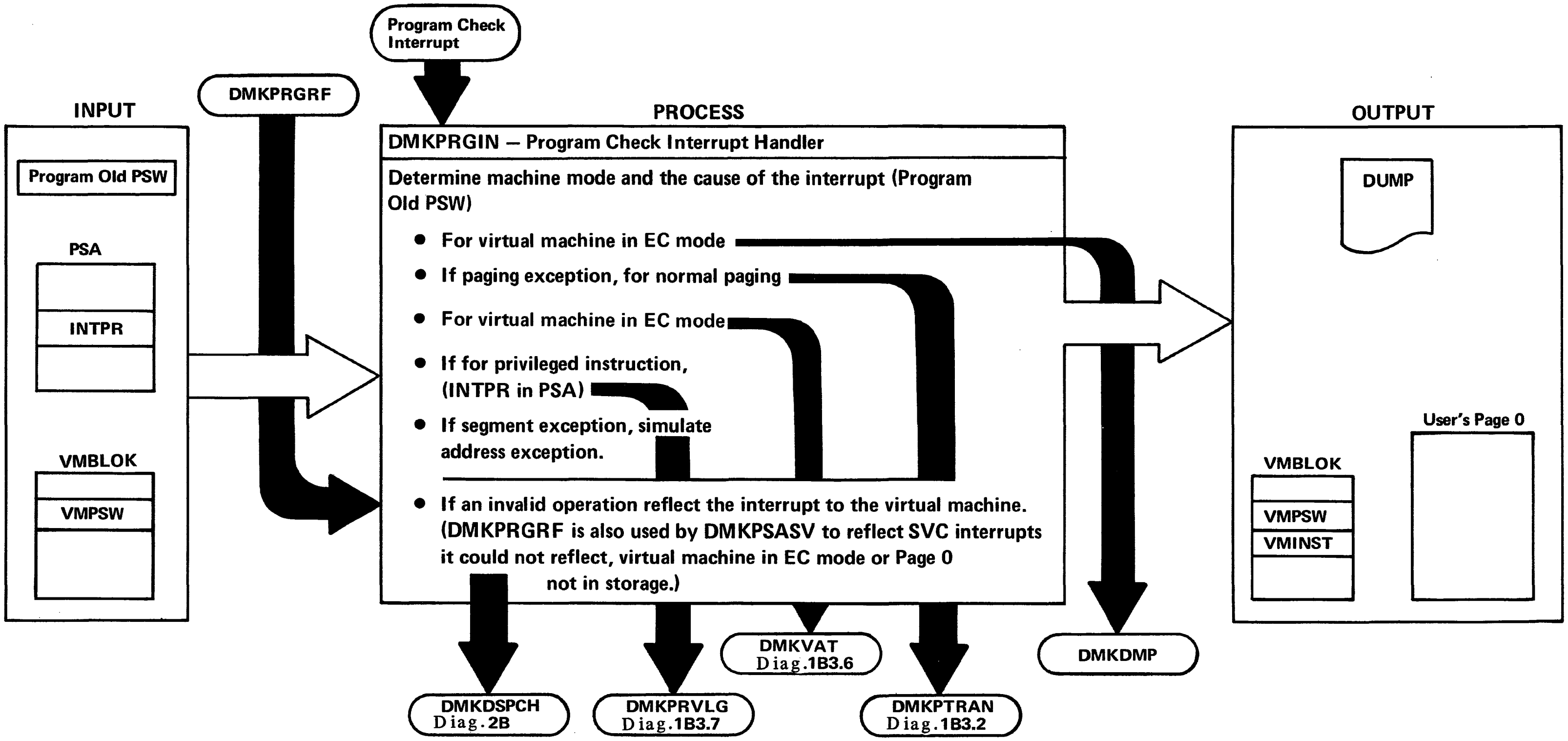
For timers:

- Clock comparator, unstack TRQBLOK from *DMKSCHTQ,* set new comparator value, call DMKSTKIO
- X'80' timer, flag running user-time slice end.
- CPU timer, flag running user to be dropped from queue.

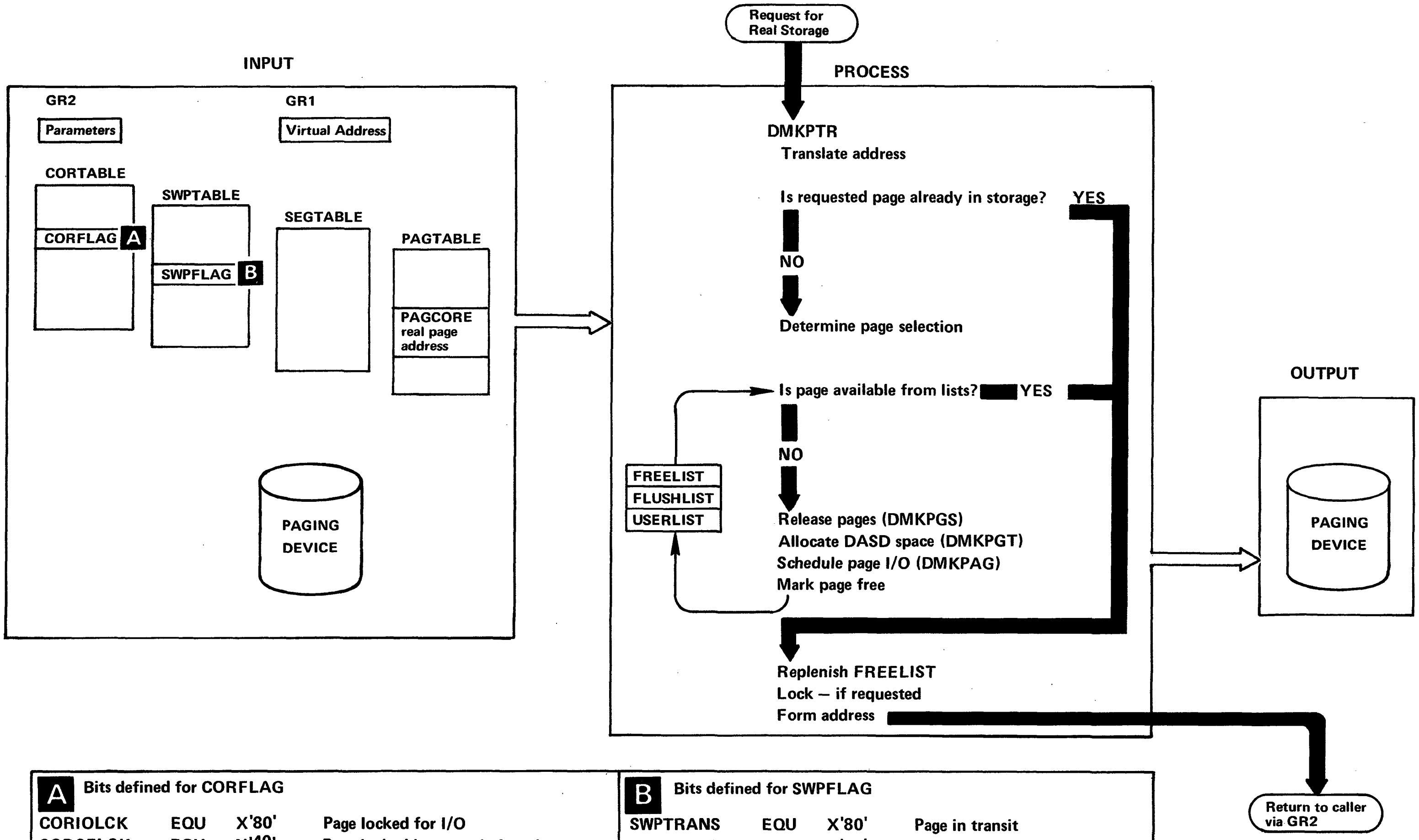
For EXTERNAL Key on operator's console,

- Set the disconnect flag (VMOSTAT = X'10') in VMBLOK
- Halt any outstanding I/O (operator's RDEVBLOK)
- Clear any outstanding console request, via *DMKQCNCL.*
- If a virtual machine was not running when the interrupt occurred resume via , otherwise

Diag. 1B3. Program Interrupt Handler


Program Check Interrupt
DMKPRGRF
INPUT
Program Old PSW
PSA
INTPR
VMBLOK
VMPSW
PROCESS
DMKPRGIN — Program Check Interrupt Handler
Determine machine mode and the cause of the interrupt (Program Old PSW)
● For virtual machine in EC mode
● If paging exception, for normal paging
● For virtual machine in EC mode
● If for privileged instruction, (INTPR in PSA)
● If segment exception, simulate address exception.
● If an invalid operation reflect the interrupt to the virtual machine. (DMKPRGRF is also used by DMKPSASV to reflect SVC interrupts it could not reflect, virtual machine in EC mode or Page 0 not in storage.)
DMKDSPCH Diag.2B
DMKPRVLG Diag.1B3.7
DMKVAT Diag.1B3.6
DMKPTRAN Diag.1B3.2
DMKDMP
OUTPUT
DUMP
VMBLOK
VMPSW
VMINST
User's Page 0

Diag. 1B3.0. Paging Overview



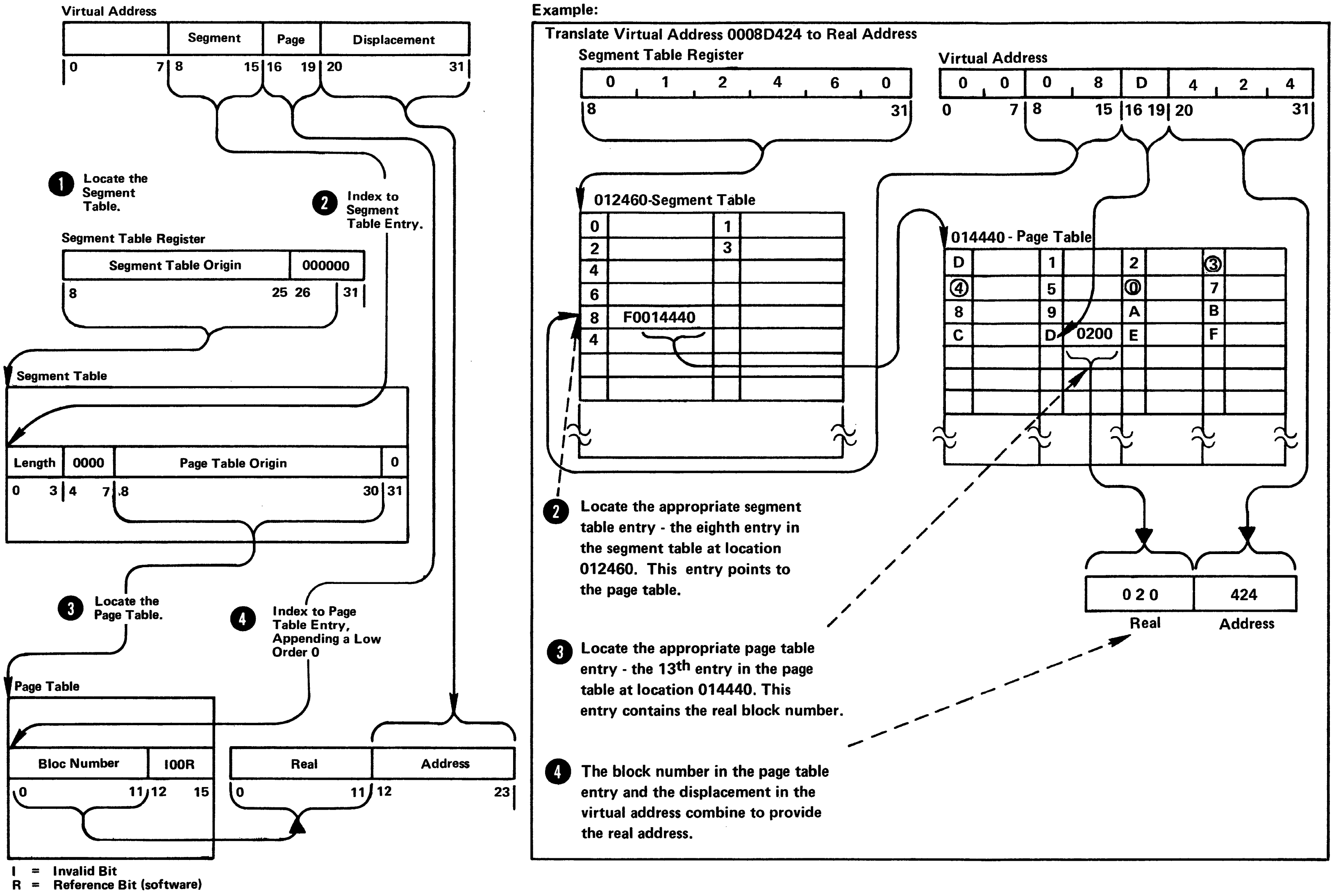
A Bits defined for CORFLAG

CORIOLCK	EQU	X'80'	Page locked for I/O
CORCFLCK	EQU	X'40'	Page locked by console function
CORFLUSH	EQU	X'20'	Page is in flush list
CORFREE	EQU	X'10'	Page is in free list
CORSHARE	EQU	X'08'	Page is shared
CORRSV	EQU	X'04'	Page is reserved
CORDISA	EQU	X'01'	Page disabled – not available

B Bits defined for SWPFLAG

SWPTRANS	EQU	X'80'	Page in transit
SWPRECMP	EQU	X'40'	Page permanently assigned
SWPALLOC	EQU	X'20'	Page enqueued for allocation
SWPSHR	EQU	X'10'	Page shared
SWPREF1	EQU	X'08'	1st half page referenced
SWPCHG1	EQU	X'04'	1st half page changed
SWPREF2	EQU	X'02'	2nd half page referenced
SWPCHG2	EQU	X'01'	2nd half page changed

Diag. 1B3.1. Virtual to Real Address Translation



2 Locate the appropriate segment table entry - the eighth entry in the segment table at location 012460. This entry points to the page table.

3 Locate the appropriate page table entry - the 13th entry in the page table at location 014440. This entry contains the real block number.

4 The block number in the page table entry and the displacement in the virtual address combine to provide the real address.

addition, each table entry contains an indication of whether the corresponding virtual page is available to the user in real storage. These tables are referenced directly by the DAT feature when the user's program is running.

- The second set of tables is a map of the locations of the user's pages on the DASD devices that comprise the system's paging or auxiliary storage. The DASD addresses in these tables can either represent the source of a page of virtual storage (the location to which a page may be moved, if necessary) or a dummy address, indicating that the given page has not yet been referenced, and thus has a value of binary zeros.

The tables are arranged in a format indexable by virtual storage address. In addition to containing the address of a page, each entry contains flags and status bytes that indicate such information as:

- The storage protection keys to be assigned to the page when it is made resident.

- Whether the page is currently on its on its way into or out of the system (in transit), etc.

These tables, called swap tables, are not referenced directly by the hardware as are the page and segment routines tables, but are used by paging management to locate user pages that are needed to execute a program.

Virtual storage management is done by the technique known as demand paging. This means that a page of virtual storage is not 'paged in' from its DASD auxiliary store slot until it is needed for execution. CP does not determine the pages required by a user before he is run. A demand for a page can be made either implicitly or explicitly.

- An implicit demand is made when a user program attempts to reference a page that is not available in real main storage. This attempt causes a program interrupt with the interrupt code indicating a page or segment exception. Upon recognition of this condition, control is passed to the paging manager to obtain a page of real main storage and to bring in the desired page.

- An explicit request for virtual storage can be made by the control program (for example, in the course of translating a user's channel program). If, in the process of translation, the control program encounters a CCW that addresses a page that is not resident in real storage, a call is made to the paging manager to make the referenced page resident.

While the requested page is being fetched, the requesting user is unable to run; however, it may be possible to run other tasks in the system, and the control program runs these while that page is being paged in. When the requested page is resident, the user can be run and is dispatched in his turn.

In addition to obtaining pages by demand, users implicitly or explicitly release pages of their virtual storage space. Part of the space may be explicitly released from both real and virtual storage via a diagnose instruction which indicates to the control program those pages that are to be released. An entire virtual storage is released when a user IPLs a new operating system or logs out from the system.

The VM/370 control program itself also has virtual storage associated with it. This space is used to contain the control program (some parts of which need not always be resident in real storage), and is also used for virtual storage buffers for spooling and system directory operations. Although the control program makes use of virtual storage space for its execution, it does not run in relocate mode itself. Thus, nonresident modules must be completely relocatable.

Real Storage Management

It is the function of real storage management to efficiently allocate the system's page frames of real storage to satisfy the demands for virtual pages made by the system's users. Efficiency of allocation involves a trade-off; the paging manager utilizes only enough CPU time allocating to ensure that:

1. The set of virtual storage pages which are resident represent those pages that are most likely to be used.

2. The number of cycles available to execute user programs is sufficient.

Inefficiency in the first area causes a condition known as thrashing, which means that highly used pages are not allowed to remain resident long enough for useful work to be performed by or on them. Thrashing could be aggravated by the paging manager's page selection algorithm or by a dispatcher that attempts to run more tasks than the system can handle (the sum of their storage requirements exceeds the real paging space available in the system). Thus, the paging manager must keep statistics on system and user paging activity and make these statistics available to the dispatcher so that a potential thrashing condition may be detected and prevented.

Inefficiency in the second area causes an unacceptable ratio of control program overhead to user program time, and in extreme case may cause the control program to utilize excessive CPU time. In order to understand how allocation is determined by the VM/370 control program, it is first necessary to describe the way in which the inventory of real storage page frames is described to the system.

Each page frame (4096 byte blocks) of real storage in the system is in one of two basic states: not-pageable or pageable. A not-pageable page must remain resident in real storage for some finite period of time; thus, the page frame cannot be taken from its current owner in order to give it to someone else. Pages can be not-pageable either permanently or temporarily, depending on their use.

- Temporary locks usually occur when an I/O operation has been initiated that is moving data either to or from the page, and the page must be kept in real storage until the operation has completed.

- A page can also be temporarily not-pageable if it contains a nonresident control program routine that is active.

In addition, a page can be not-pageable through use of the LOCK command. Pages locked in this fashion are permanently resident until they are explicitly unlocked by the UNLOCK COMMAND¬ Pages that are usually considered permanently not-pageable are those that contain the resident portion of the control program and those that contain the system's free storage area in which control blocks, I/O buffers, etc. are built.

CORTABLE: The data area that is used by the page management routines to control and allocate real storage is the CORTABLE. Each page frame of real storage has a corresponding entry in the CORTABLE, and since the table entries are fixed length and contiguous, the entry for any given real page frame may be located directly by indexing into the table. Each entry contains pointers that indicate both the status and ownership of the real page which it represents. Some pointers are used to link page table and swap table entries to the real page (and thus establish ownership), while others are used to link the entry into one of several lists that the paging routines use to indicate the page's status and availability for paging. A given CORTABLE entry may appear on one of three lists if its real page is available for paging; however, if the page is locked or in transit, its entry is not in any list and is not referenced when available page frames are being searched for swap candidates. The lists are known as the FREELIST, the FLUSHLST, and the USERLIST, and they represent various levels of page availability.

- The FREELIST contains page frames that are immediately available for assignment to a requesting user. The virtual storage pages for which they were last used have either been released by their owners or they have been paged out to auxiliary storage. Requests for real storage are always satisfied from the FREELIST. If the list has been depleted, the requestor waits until a new page frame becomes available as the result of a virtual storage release or a swap-out.

- The FLUSHLIST contains page frames that belong to those users that have been dropped from an active DISPATCHing queue. The FLUSHLST is the first place that the page frame selection routine looks to find a page to swap out or to assign to the FREELIST for a user who requires real storage space.

- The USERLIST contains the CORETABLE entries for all other pageable pages in the system that belong to active users.

Requests For Real Storage Pages

Requests for real storage fall into two general categories; those that are requesting space for a page of virtual storage, and those (such as requests for CP work space) that need the real page for their own use. The former, more general case is discussed first, since the latter case is a subset of the first.

The main page manager routine, DMKPTRAN, maps a request for a specific user's virtual storage address into a page of real storage. This requires that:

- The virtual page be read in.
- The necessary tables be updated to show the proper status of the page.

DMKPTRAN requires that the caller supply only the virtual address to be translated and any options that apply to the page to be located. Most calls are made via the TRANS macro, which sets up the necessary parameters, determines if the required page is resident, and calls DMKPTRAN if it is not.

When DMKPTRAN receives control, it first tests to see if the requested page is resident. This is done via the LRA hardware translation feature. If the page is resident, the routine locks the page if requested and exits to the caller. If the LRA indicates that the page is unavailable, it is still possible that the required page is resident. This occurs if the page has been placed on the FREELIST but has not been assigned to another user. When the page swap routine removes a page from a user, the unavailable bit is set in the corresponding page table entry; however, the real main storage index for the page is left unchanged. The page table entry is set to zero only when the corresponding page is actually assigned to another user. Thus, if DMKPTRAN finds the page unavailable, a further test is made on the page table entry to see if the page can be reclaimed. If the entry is not zero (aside from the unavailable bit), the CCRTABLE entry for the page is removed from the FREELIST and the page is returned to the calling user.

If the page table entry corresponding to the virtual page requested is zero, the required page is not in real storage and must be paged in. However, it is possible that the page is already on its way into main storage. This condition is indicated by a flag in the SWPTABLE entry for the virtual page. The DMKPAGIO routine maintains a queue of CPEXBLOKs to be dispatched when the pending page I/O is complete. The CPEXBLOK for the page in transit is located and a new CPEXBLOK, representing the current request, is chained to it.

Before exiting to wait for the paging operation to complete, DMKPTRAN checks to see if the deferred return (DEFER option) has been specified. If it has not, DMKPTRAN returns to the caller. If the DEFER option has been requested, DMKPTRAN exits to the dispatcher to wait for page I/O completion. When the requested page has been read into real storage, the list of CPEXBLOKs are unstacked FIFO to satisfy all requests for the page that arrived while it was in transit.

If a page is not in transit, a page frame of real storage must be allocated to fill the request. Before the allocation routine is called, a test is made to see if the caller wishes the return to his routine or to be delayed until after the requested page is available. If the DEFER option is not requested, DMKPTRAN returns to the caller after first building and stacking a CPEXBLOK that allows processing of the page request to be continued the next time the dispatcher (DMKDSPCH) is entered.

DMKPTRAN next calls the FREELIST manager (DMKPTRFR) to obtain the address of the next available CORTABLE entry. DMKPTRFR maintains a FIFO list of the CORTABLE entries for those page frames that are immediately available for assignment. As DMKPTRFR releases these page frames, a check is made to see if the number of entries on the FREELIST has fallen below a dynamically maintained minimum value. If it has, the page selection routine (SELECT) is called to find a suitable page for placement in the FREELIST. The number maintained as the FREELIST threshold has a value equal to the number of users in queue1 plus the number of users in queue2 plus 1.

The FREELIST is replenished directly by users releasing virtual storage space. The page-out routine DMKPGSPD calls DMKPTRFT to place released pages directly on the FREELIST. However, most replenishment is done via the page selection routine, SELECT. SELECT is called by DMKPTRFR when the FREELIST count falls below the current minimum, or when a user page is reclaimed from the FREELIST. In either case, the selection algorithm attempts to find a page to swap to auxiliary storage. The highest priority candidates for a swap are those pages whose CORTABLE entries appear on the FLUSHLST. SELECT attempts to take a flushed page before it takes a page from an active user. If such a page is found, it is checked to see if it has been changed since page-in. If not, it is placed in the FREELIST by DMKPTRFT; otherwise, it is scheduled for a swap-out by dequeueing the CORTABLE entry from the FLUSHLST, constructing a CPEXBLOK for dispatching after I/O completion, and exiting to DMKPAGIO via a GOTO. After the paging I/O is complete, the entry is placed on the FREELIST via a call to DMKPTRFT.

If the FLUSHLST is exhausted, SELECT must take a page from an active user by examining the pages represented by the entries in the USERLIST to locate the least recently used user page. This list is scanned from top to bottom, and each page is tested to see if its hardware referenced bits have been set. If a page has been referenced, its bits are reset and it is queued to the end of the USERLIST. This process is continued until either an unreferenced page is found or the list is exhausted. An unreferenced page is immediately selected. However, if the list is exhausted, it is rescanned from the top. An unreferenced page is always found; in the worst case it is the first one tested on the USERLIST at initial entry. However, if this occurs, it indicates that the rate of entry to SELECT is too low to permit differentiation between high and low usage pages.

Once a page has been selected and its page-out is scheduled, control is returned to DMKPTRFR, which then passes control back to DMKPTRAN with the address of the CORTABLE entry that was allocated. In most cases, page-outs are completely overlapped with page-ins. Approximately one half of all page-ins require a corresponding page-out.

Once a real page has been assigned, DMKPTRAN checks to see if a page-in is required. It usually is, and the DASD address of the virtual storage page must be obtained from the user's swap table entry and the I/O operation scheduled. However, if the page has not yet been referenced (as indicated by a DASD address of zero), the real main storage page is set to zero. After the page-in operation has been queued, DMKPTRAN exits to the paging I/O scheduler (DMKPAGIO) which initiates the paging operation and exits to the dispatcher (DMKDSPCH) to await the interrupt.

After the required page has been read in or set to zero, DMKPTRAN queues the appropriate CORTABLE entry to the end of the USERLIST, where it eventually is available for page selection. After developing the real storage address that corresponds to the requested virtual address, DMKPTRAN tests to see if the caller has requested that the page be locked. If LOCK is requested, the CORTABLE entry is de-queued from the USERLIST and is not available for selection. A resident page can also be locked by removing it from the USERLIST. In addition, a LOCK count is maintained in the CORTABLE entry so that when all locks have been satisfied the page can again be made available for paging (see PAGUNLOK).

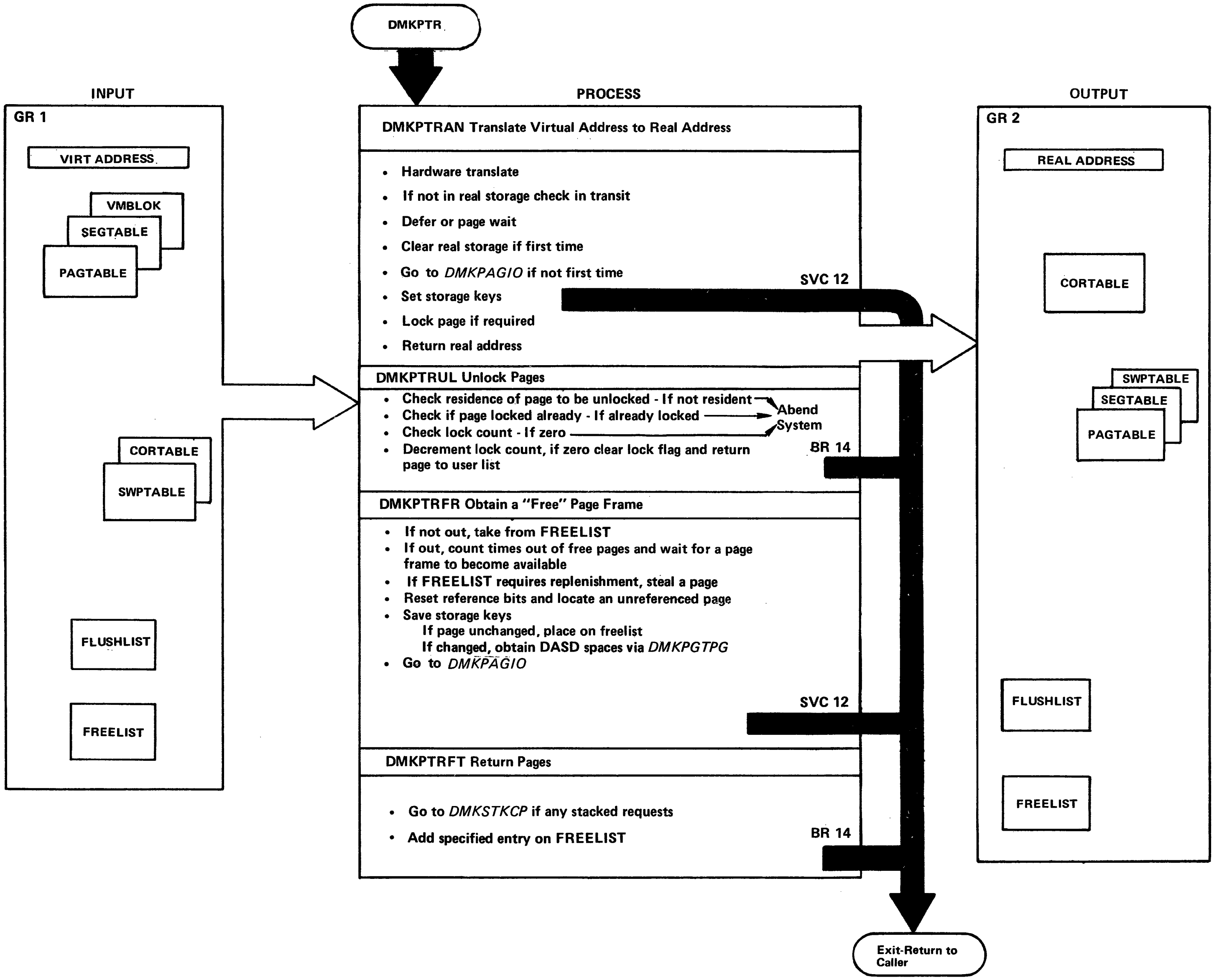
Some requests for main storage pages are handled differently than the general case of virtual-to-real storage mapping. In particular, it may be necessary for CP to obtain additional free storage for control blocks, I/O lists, buffers, etc. This is handled by the free storage manager, which makes a direct call to DMKPTRFR to obtain the needed storage. Usually this storage is immediately available (due to the page buffering technique previously described). However, if the FREELIST is exhausted, the request for free storage is recognized as a high priority call and queued first on the list of those waiting for free pages.

The real storage manager (DMKPTR) accumulates paging statistics which are used by the scheduler (DMKSCH) to project user storage requirements. A count of page-reads and page-writes is kept in each user's VMBLOK; the corresponding total counts for the system are kept in DMKPSA. A running total of the number of pages a user has resident, at each instance of page-read, is kept in the VMBLOK. A count of the number of times a user enters page-wait, because a page has been stolen from him, is also kept in the VMBLOK. The section entitled "Controlling the Depth of Multiprogramming" under the heading "Dispatcher/Scheduler" describes the use to which the scheduler puts these counts.

<u>VM/370 Virtual=Real Function</u>: The VM/370 Virtual=Real function involves the mapping in a one-for-one correspondence of a virtual machine storage area with an equivalent real storage area. For instance, virtual

page 1 is in real page frame 1 and virtual page 20 is in real page frame 20. Virtual page 0, since it cannot occupy real page 0, is relocated to be at the end of the virtual storage space.

Diag. 1B3.2 Paging, Provide Real Storage Area



DMKPTRAN Translate Virtual Address to Real Address

- Hardware translate
- If not in real storage check in transit
- Defer or page wait
- Clear real storage if first time
- Go to *DMKPAGIO* if not first time
- Set storage keys
- Lock page if required
- Return real address

DMKPTRUL Unlock Pages

- Check residence of page to be unlocked - If not resident
- Check if page locked already - If already locked
- Check lock count - If zero
- Decrement lock count, if zero clear lock flag and return page to user list

DMKPTRFR Obtain a "Free" Page Frame

- If not out, take from FREELIST
- If out, count times out of free pages and wait for a page frame to become available
- If FREELIST requires replenishment, steal a page
- Reset reference bits and locate an unreferenced page
- Save storage keys
  - If page unchanged, place on freelist
  - If changed, obtain DASD spaces via *DMKPGTPG*
- Go to *DMKPAGIO*

DMKPTRFT Return Pages

- Go to *DMKSTKCP* if any stacked requests
- Add specified entry on FREELIST

The VM/370 control program nucleus is altered at system generation to support the Virtual=Real function. Users with Virtual=Real (specially identified in the directory) can then log on and use the space reserved for this function. That space can be used by only one virtual machine at a time. Two virtual machines with the Virtual=Real capability cannot occupy the same space at the same time.

The Virtual=Real function is primarily used so that the virtual machine may bypass the control program's CCW translation. This is possible because I/O from a virtual machine occupying a Virtual=Real space contains a list of CCWs whose data addresses reflect the real storage addresses. The restriction in this situation is that the virtual machine does not perform I/O into page 0 since this would perform a data transfer into real page 0. At the same time, it is assumed, and cannot be checked, that the virtual machine will also not attempt to do I/O beyond the bounds of its virtual addressing space. To do so would cause the destruction of either the VM/370 control program nucleus, which resides beyond the virtual machine space, or another user's page.

The bypassing of CCW translation for the virtual machine occupying the Virtual=Real space is only invoked after the virtual machine has executed the SET NOTRANS ON function. This function can only be issued by the virtual machine occupying the Virtual=Real space. The function initiates the bypass of CCW translation. This functicn is automatically turned off if the virtual machine performs an explicit reset, or an implied reset by performing a virtual IPL. During virtual machine IPL, it is required that I/O be performed into page 0. For this reason, normal virtual IPL simulation assumes CCW translation in effect in order to accomplish the full simulation. Once the IPL sequence has completed, the CCW translation function can be bypassed by issuing the SET NOTRANS ON command.

When the virtual machine demands a page through normal use of the control program's page tables, the paging routine recognizes the Virtual=Real capability. It then assigns the virtual page to the equivalent real page frame and does not perform a paging operation, since all these pages are resident and are never swapped out.

Note: The virtual machine running with Virtual=Real is still run in System/370 relocate mode.

Virtual 270X lines and sense operations from the virtual machine do not use the Virtual=Real feature. These invoke CCW translation for the virtual enable/disable lines and the transfer of the sense bytes.

The UNLOCK command has an operand called VIRT=REAL and essentially releases the Virtual=Real area for normal system paging use. Once the area has been released, it can only be reclaimed by an IPL of the VM/370 System. The size of the Virtual=Real area is an installation specification that is part of the special nucleus generaticn procedure that is outlined in the *VM/370 Planning and System Generation Guide*. The size of the area must be large enough to contain the entire addressing space of whatever virtual machine wishes to occupy that space. A virtual machine can use a smaller space than is provided but cannot use a larger space without regenerating the VM/370 control program nucleus.

DASD Storage Management

Any user virtual storage pages that have been referenced but are not resident in real storage must be kept on the DASD paging device. DASD page space is assigned only when the page is selected for a page-out. Certain DASD pages may also be marked read-only. Thus, the DASD address slot initially associated with the page should be considered to be the source of the page only. If the page is changed after it has been read into real storage, a new slot must be obtained when it is paged out. Examples of read-only pages are those which contain portions of pageable saved systems and pages which are part of a system SPOOL file. Slots can be reassigned when DMKPTRAN finds that it must swap a page out to a movable head DASD device. In this case, the old slot is released and the new slot is obtained.

SLOT ALLOCATION: If a new slot is required, the DMKPGT is called to supply the address of an available slot. DMKPGT maintains a chain of cylinder allocation maps for each cylinder that has been assigned for either virtual storage or spool file paging. The allocation chains for spooling are kept separately from those used for paging so that they can be checkpointed in case of a system failure. However, in other respects they are the same. The allocation blocks for a given volume are chained from the RDEVBLOK for the device on which the volume is mounted. The chains of cylinder and slot allocation blccks are initialized by DMKCPI. Each block on an allocation chain represents one cylinder of space assigned to paging, and contains a bit map indicating which slots have been allocated and which are available. Each block also has a pointer to the next allocation block on the chain, a cylinder number, and a record count. DMKPGT searches this list sequentially until an available slot is found; its CASD address is then determined and passed back to the calling routine. If DMKPGT cannot find a cylinder with a de-allocated slot, it enters the cylinder allocation phase described next. When an available cylinder is found, it constructs a page allocation block for this cylinder and allocates a page to the caller.

<u>CYLINDER ALLCCATION</u>: DMKPGT controls the paging and spooling I/O load of the system by allocating cylinders evenly across all available channels and devices. In order for a device to be considered available for the allocation of paging and spcoling space:

- Its volume serial number must appear in the system's owned list.

- It must have at least one cylinder of temporary space marked as available in the cylinder allocation block which is located on cylinder 0, head 0, record 3.

At system initialization time, CPINIT reads in the allocation records for each volume and constructs the chains of device allocation blocks from which DMKPGT allocates the cylinders. In managing the cylinder allocation, DMKPGT takes three factors into consideration: device type, device address, and possible status as a preferred paging device.

A request for a cylinder of virtual storage page space is satisfied by allocating on a preferred paging device, provided that one exists on the system and that it has page space available. Preferred paging devices are specified by the installation at system generation time, and generally should be devices on which excessive seek times does not occur. A typical preferred paging device would be the IBM 2305 Fixed Head Storage Facility. If the 2305 is assigned as a preferred device, it is possible to allocate some of its space for other high priority data files without excessively degrading paging. An example of such usage would be for high activity read-only saved system pages that are not shared in real storage, and high activity system residence disks.

It is also possible to designate moveable head DASD devices such as the 3330 and 2314/2319 Direct Access Storage Facilities as preferred paging devices. The module(s) so designated should not be required to seek outside of a relatively narrow cylinder band around the center of the paging areas. It is advisable to share the access arm of a moveable head preferred paging device with only the lowest usage data files.

If one or more preferred devices are defined on the system, CP allocates all of the page space available on these before it allocates on any other available owned volumes. Within the class of preferred devices, space is allocated first on the fastest devices, and among these on a round robin basis across channels and devices. Allocation on nonpreferred devices is spread out in the same manner. Cylinders for spooling space are not allocated from preferred devices. Allocation on a given device is done from the relative center of the volume outward, a cylinder at a time in a zig-zag fashion in an attempt to minimize seek times.

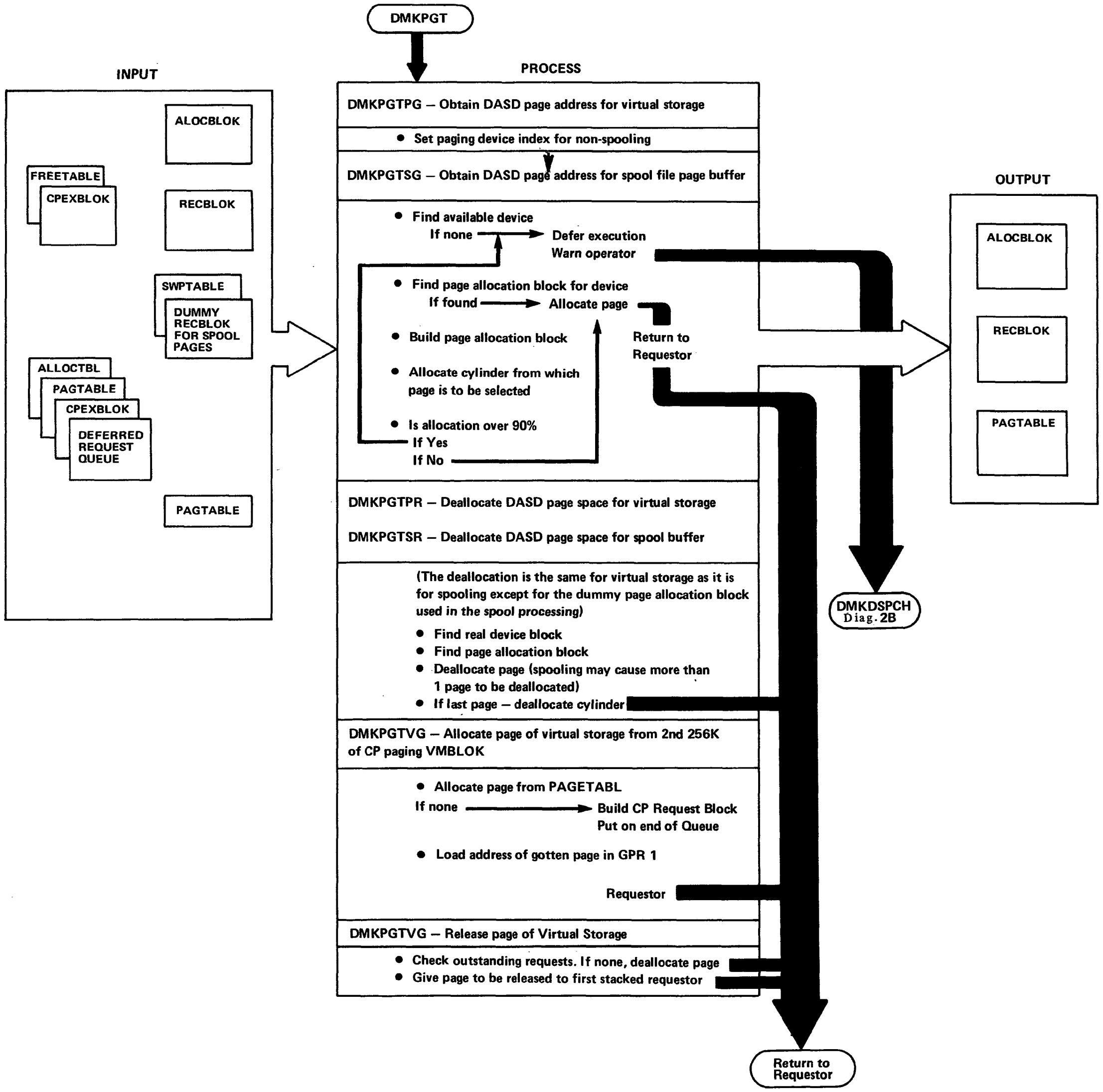
When a request to allocate a slot for virtual storage paging is received by DMKPGTGT and the slot must be allocated on a moveable-head (2314/2319 or 3330) device, a cylinder and slot is selected in the following manner:

1. An attempt is first made to allocate a slot on the cylinder at which the arm on the selected device is currently positioned.

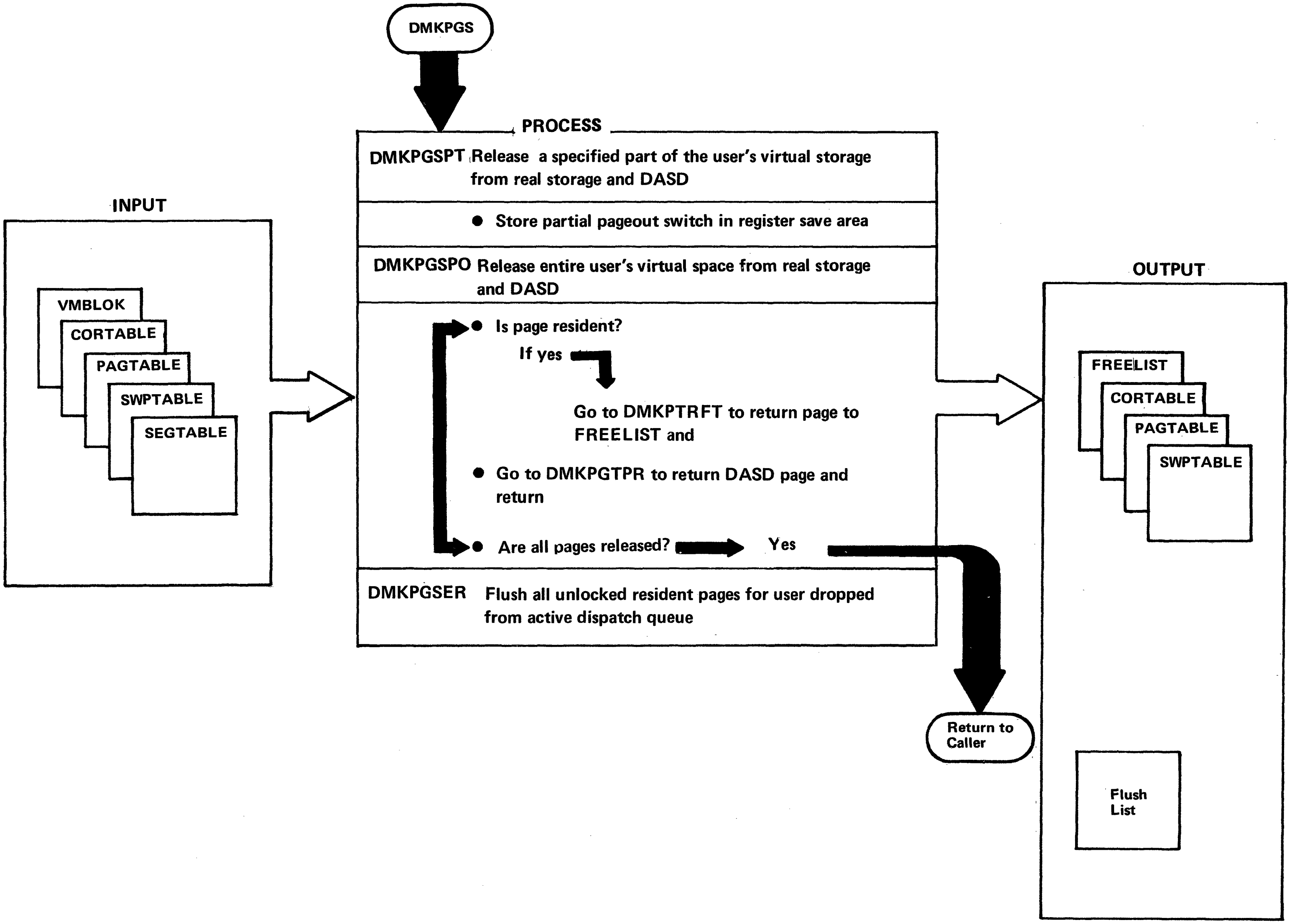
2. If slots are not available on the current cylinder, an attempt is made to allocate on a cylinder for which paging I/O has been queued.

3. If the above conditions cannot be met, allocation is done as close to the center of the volume as is possible.

Diag. 1B3.3. Paging, Allocate DASD Space


DMKPGT
INPUT
ALOCBLOK
FREETABLE
CPEXBLOK
RECBLOK
SWPTABLE
DUMMY RECBLOK FOR SPOOL PAGES
ALLOCTBL
PAGTABLE
CPEXBLOK
DEFERRED REQUEST QUEUE
PAGTABLE
PROCESS
DMKPGTPG – Obtain DASD page address for virtual storage
• Set paging device index for non-spooling
DMKPGTSG – Obtain DASD page address for spool file page buffer
• Find available device
If none → Defer execution
Warn operator
• Find page allocation block for device
If found → Allocate page
Return to Requestor
• Build page allocation block
• Allocate cylinder from which page is to be selected
• Is allocation over 90%
If Yes
If No
DMKPGTPR – Deallocate DASD page space for virtual storage
DMKPGTSR – Deallocate DASD page space for spool buffer
(The deallocation is the same for virtual storage as it is for spooling except for the dummy page allocation block used in the spool processing)
• Find real device block
• Find page allocation block
• Deallocate page (spooling may cause more than 1 page to be deallocated)
• If last page – deallocate cylinder
DMKPGTVG – Allocate page of virtual storage from 2nd 256K of CP paging VMBLOK
• Allocate page from PAGETABL
If none → Build CP Request Block
Put on end of Queue
• Load address of gotten page in GPR 1
Requestor
DMKPGTVG – Release page of Virtual Storage
• Check outstanding requests. If none, deallocate page
• Give page to be released to first stacked requestor
OUTPUT
ALOCBLOK
RECBLOK
PAGTABLE
DMKDSPCH
Diag. 2B
Return to Requestor

Diag. 1B3.4. Release Virtual Machine Pages



DMKPGSPT Release a specified part of the user's virtual storage from real storage and DASD

- Store partial pageout switch in register save area

DMKPGSPO Release entire user's virtual space from real storage and DASD

- Is page resident?

  If yes

  Go to DMKPTRFT to return page to FREELIST and

- Go to DMKPGTPR to return DASD page and return

- Are all pages released? Yes

DMKPGSER Flush all unlocked resident pages for user dropped from active dispatch queue

Before DMKIOSQR is called, the queue of IOBLOKs currently scheduled on the device is examined. If paging I/O has already been scheduled on a device, the paging channel programs are slot sorted and chained together with TICs.

Paging I/O

All input/output requests for virtual storage and spooling pages are handled by DMKPAGIO. DMKPAGIO constructs the necessary task blocks and channel programs, expands the compressed slot addresses, and maintains a queue of CPEXBLOKs for pages to be moved. Once the I/O scheduled by DMKPAGIO completes, it unchains the CPEXBLOKs that have been queued and calls DMKSTKCP to stack them for execution. DMKPAGIO is entered via a GOTO from:

- DMKPTRAN to read and write virtual storage pages
- DMKRPA to read and write virtual storage spool buffers

In any case, all that need by passed to DMKPAGIO is the address of the CORTABLE entry for the page that is to be moved, the address of a SWPTABLE entry for the slot, a read or write operation code, and the address of a CPEXBLOK that is to be stacked for dispatching after the I/O associated with the page has completed. DMKPAGIO obtains an IOBLOK and builds a channel program to do the necessary I/O, and uses the device code that is part of the page address to index into the system's OWNDLIST and locate the real device to which the I/O request should be directed. If the device is capable of rotational position sensing, the required sector is computed and a Set Sector command is inserted into the channel program. The real SIO supervisor DMKIOSQR is then called to schedule the operation on the proper device.

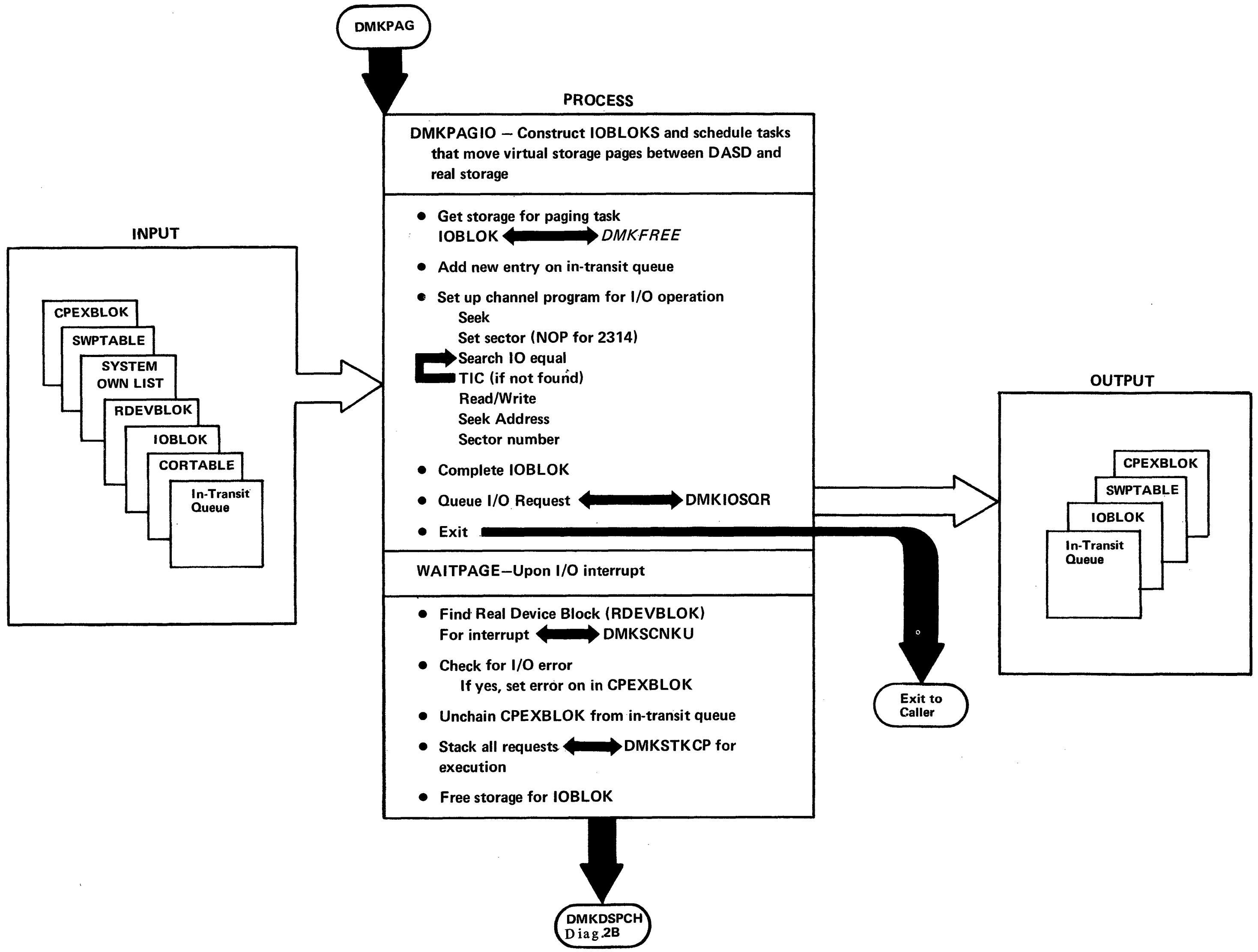
When the interrupt for the paging operation is processed by the primary I/O interrupt handler, the IOBLOK that controls the operation is unstacked to the interrupt return address, WAITPAGE, in DMKPAGIO. WAITPAGE then unchains the CPEXBLOKs that are queued to DMKPAGQ, and then stacks the queued CPEXBLOKs,via calls to DMKSTKCP, in the order in which they were received. The address of the real page is filled in to the appropriate page table entry and the pointers denoting the ownership of the real page are filled into the CORTABLE entry by the processing routines in DMKPTRAN. If a fatal I/O error occurred for the page, the CPEXBLOKs associated with it are flagged, and the dispatcher DMKSDPCH sets a nonzero condition code when it activates the pending task. The error recovery followed depends on the operation being performed. Paging I/O errors associated with spooling operations are discussed in the sections on "Virtual and Real Spooling", while errors associated with virtual storage paging operations are discussed later in section "Virtual Storage Paging Error Recovery".

DMKPAGIO maintains its own subpool of preformatted paging IOBLOKs. As I/O operations complete, their IOBLOKs are added to a list of available blocks; as new blocks are needed, they are taken from this list. If the list is empty, DMKFREE is called to obtain storage for a new block. DMKPAGIO also periodically calculates system paging overhead. After 200 pages have been moved (read or written), the elapsed time for the 200 pages is computed, and the paging rate is calculated in pages per second. The recent paging load, expressed as the percentage of time that more than one half of the system's pages were idle due to page-wait, is averaged with the previous load and re-projected as the expected load for the next interval.

Virtual Storage Paging Error Recovery

Errors encountered during virtual storage (as opposed to spooling) paging operation can generally be classified as either soft or hard errors. Soft errors allow the system to continue operation without delay or degradation. Hard errors can cause noticeable effects such as the abnormal termination of user tasks (ABEND) and response degradation. Errors that are successfully

Diag. 1B3.5. Page-in, Page-out


DMKPAG
PROCESS
DMKPAGIO – Construct IOBLOKS and schedule tasks that move virtual storage pages between DASD and real storage
• Get storage for paging task
IOBLOK ⟷ DMKFREE
• Add new entry on in-transit queue
• Set up channel program for I/O operation
Seek
Set sector (NOP for 2314)
Search IO equal
TIC (if not found)
Read/Write
Seek Address
Sector number
• Complete IOBLOK
• Queue I/O Request ⟷ DMKIOSQR
• Exit
WAITPAGE–Upon I/O interrupt
• Find Real Device Block (RDEVBLOK)
For interrupt ⟷ DMKSCNKU
• Check for I/O error
If yes, set error on in CPEXBLOK
• Unchain CPEXBLOK from in-transit queue
• Stack all requests ⟷ DMKSTKCP for execution
• Free storage for IOBLOK
INPUT
CPEXBLOK
SWPTABLE
SYSTEM OWN LIST
RDEVBLOK
IOBLOK
CORTABLE
In-Transit Queue
OUTPUT
CPEXBLOK
SWPTABLE
IOBLOK
In-Transit Queue
Exit to Caller
DMKDSPCH
Diag.2B

retried or corrected are known only to the I/O supervisor and the I/C error retry and recording routines; they appear to the second level interrupt handlers (such as WAITPAGE) as if the original operation completed normally.

SOFT ERROR RECOVERY: An I/O error which occurs on a page swap-out is considered to be a soft error. DMKPTRAN calls DMKPGTPG to assign a different DASD page slot and the page is re-queued for output. The slot which caused the error is not de-allocated, and thus is not assigned to another user. All other uncorrectable paging errors are considered hard in that they may more drastically affect system performance.

HARD ERRCR RECOVERY: Hard paging errors occur on either I/O errors for page reads or upon the condition of exhausting the system's spooling and paging space. Recovery attempted on hard errors depends upon the nature of the task for which the read was being done. If the operation was an attempt to place a page of a user's virtual storage into real storage, the operation of that particular virtual machine is terminated by setting the page frame in error to zero and placing the virtual machine in console function mode. The user and operator are informed of the condition, and the page frame causing the error is not de-allocated, thereby insuring that it will not be allocated to another user.

The control program functions which call DMKPTRAN (such as spooling, pageable control program calls, and system directory management) have the option of requesting that unrecoverable errors be returned to the caller. In this case, the CP task may attempt some recovery to keep the entire system from terminating (ABEND). In general, every attempt is made to at least allow the operator to bring the system to orderly shut-down if continued operation is impossible.

Proper installation planning should make the occurrence of a space exhaustion error an exception. An unusually heavy user load and a backed-up spooling file could cause this to happen. The operator is warned when 90% of the temporary (paging/spooling) space in the system is exhausted. He should take immediate steps to alleviate the shortage. Possible remedies that exist include preventing more users from logging on and requesting users to stop output spooling operations. More drastic measures might include the purging of low priority spool files. If the system's paging space is completely exhausted, the operation of virtual machines progressively slows as more and more users have paging requests that cannot be satisfied and operator intervention is required.

VIRTUAL RELOCATION

CP provides the virtual machine the capability of using the Dynamic Address Translation of the real System/370. Programming simulation and hardware features are combined to allow usage of all of the available features in the real hardware, (that is, 2K or 4K pages, 64K or 1M segments).

For clarification, some term definitions follow:

First-level storage: The physical storage of the real CPU, in which CP resides.

Second-level storage: The virtual storage available to any virtual machine, maintained by CP.

Third-level storage: The virtual storage space defined by the system operating in second-level storage, under control of page and segment tables which reside in second-level storage.

Page and segment tables: Logical mapping between first-level and second-level storage.

Virtual page and segment tables: Logical mapping between second-level and third-level storage.

Shadow page and segment tables: Logical mapping between first-level storage and third-level storage.

A standard, non-relocating virtual machine in CP is provided with a single control register, control register zero that can be used for:

- Extended masking of external interrupts.
- Special interrupt traps for SSM.
- Enabling of virtual block multiplexing.

A virtual machine that is allowed to use the extended control feature of System/370 is provided with a full complement of 16 control registers, allowing virtual monitor calls, PER, extended channel masking, and dynamic address translation.

An extension to the normal virtual-machine VMBLOK is built at the time that an extended control virtual machine logs onto CP. This ECBLOK contains the 16 virtual control registers, 2 shadow control registers, and several words of information for maintenance of the shadow tables, virtual CPU timer, virtual TOD clock comparator, and virtual PER event data. The majority of the processing for virtual address translation is performed by the module DMKVAT, with additional routines in DMKPRG, DMKPRV, DMKDSP, DMKCDB, DMKLOG, DMKUSO, and DMKPTR. The simulation of the relocation-control instructions (that is, LCTL, STCTL, PTLB, RRB, and LARA) is performed by DMKPRV. These instructions, with the exception of LCTL and STCTL, are not available to virtual machines which are not allowed the extended-control mode.

When an extended control virtual machine is first active, it has only the real page and segment tables provided for it by CP and operates entirely in second-level storage. DMKPRV examines each PSW loaded via LPSW to determine when the virtual machine enters or leaves extended control or translate mode, setting the appropriate flag bits in the VMBLOK. Flag bits are also set whenever the virtual machine modifies control registers 0 or 1, the registers that control the dynamic address translation feature. DMKDSP also examines PSWs that are loaded as the result of interrupts to determine any changes in the virtual machine's operating mode. The virtual machine can load or store any of the control registers, enter or leave extended control mode, take interrupts, etc., without invoking the address translation feature.

If the virtual machine, already in extended control mode, turns on the translate bit in the EC mode PSW, then the routine DMKVATMD is called to examine the virtual control registers and build the required shadow tables. (Shadow tables are required since the real DAT hardware is capable of only a first-level storage mapping.) DMKVATMD examines virtual control registers 0 and 1 to determine if they contain valid information for use in constructing the shadow tables. Control register zero specifies the size of the page and segment the virtual machine is using in the virtual page and segment tables. The shadow tables constructed by DMKVATMD are always in the same format as the virtual tables.

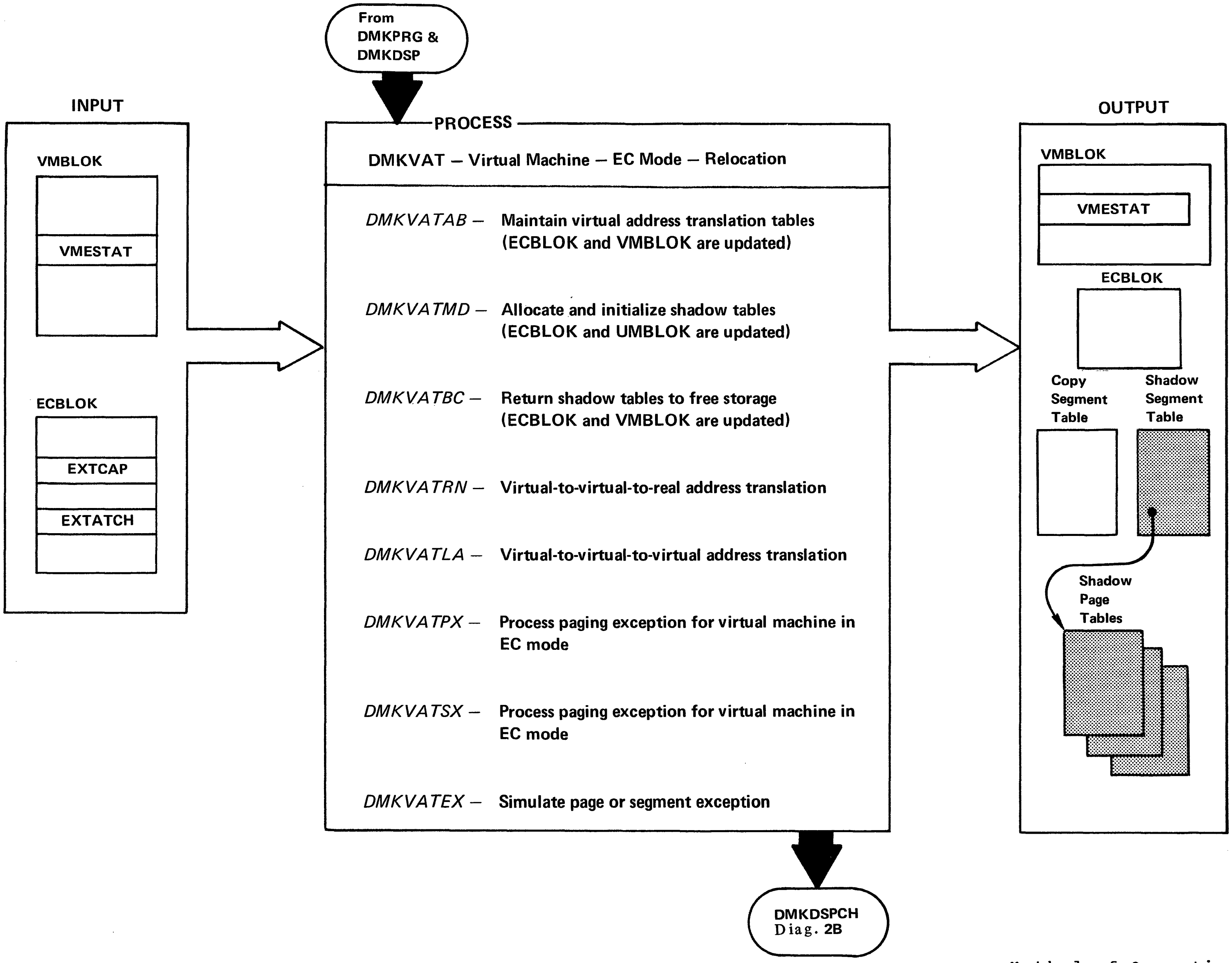
First, the virtual segment table is copied intact from second-level storage into first-level storage for speed of access when handling relocation interrupts. Another segment table of the same size, the shadow segment table, is constructed in first-level storage and initialized to indicate that all segments are unavailable. Flags are maintained in the VMBLOK to indicate that the shadow tables exist. DMKVATMD also constructs the shadow control registers 0 and 1. Shadow control register 0 contains the external interrupt mask bits used by CP, mixed with the hardware controls and enabling bits from virtual control register 0. Shadow control register 1 contains the segment table origin address of the shadow segment table.

When the virtual machine is operating in virtual translate mode, CP loads the shadow control registers into the real control registers and dispatches the user. The immediate result of attempting to execute an instruction is a segment exception, intercepted by DMKPRG and passed to DMKVATSX. DMKVATSX examines the copy, in first-level storage, of the virtual segment table in second-level storage. If the copy segment table indicates the segment is not available, the corresponding entry in the virtual segment table is examined and if necessary, the copy segment table is updated. If the virtual segment is not available, the segment exception interrupt is reflected to the virtual machine. If the virtual segment is marked available, then DMKVATSX:

- Allocates one full segment of shadow page table, in the format specified by virtual control register 0.
- Sets all of the page table entries to page not in storage.
- Marks the segment available in the shadow segment table.
- Redispatches the virtual machine via DMKDSP.

Once again, the immediate result is an interrupt, which this time is a paging exception and control is passed to DMKVATPX. DMKVATPX references the virtual page table

Diag. 1B3.6. Virtual Relocation



DMKVAT — Virtual Machine — EC Mode — Relocation

DMKVATAB — Maintain virtual address translation tables (ECBLOK and VMBLOK are updated)

DMKVATMD — Allocate and initialize shadow tables (ECBLOK and UMBLOK are updated)

DMKVATBC — Return shadow tables to free storage (ECBLOK and VMBLOK are updated)

DMKVATRN — Virtual-to-virtual-to-real address translation

DMKVATLA — Virtual-to-virtual-to-virtual address translation

DMKVATPX — Process paging exception for virtual machine in EC mode

DMKVATSX — Process paging exception for virtual machine in EC mode

DMKVATEX — Simulate page or segment exception in second-level storage through the copy segment table to determine if the virtual page is available. If the virtual page is not available, the paging interrupt is reflected to the virtual machine. However, if the virtual page is marked in storage, the virtual page table entry is used to determine which page of second-level storage is being referenced by the third-level storage address provided. DMKVATPX next determines if that page of second-level storage is resident in first-level storage at that time. If so, the appropriate entry in the shadow page table is filled in and marked in storage. If not, the required page is brought into first level storage via DMKPTRAN and the shadow page table filled in as above.

As the virtual machine continues execution, more shadow tables are filled in or allocated as the third-level storage locations are referenced. Whenever a new segment is referenced, another segment of shadow page tables is allocated. Whenever a new page is referenced, the appropriate shadow pagetable entry is validated, etc. No changes are made in the shadow tables if the virtual machine leaves translate mode (usually via an interrupt), unless it also leaves extended control mode. Dropping out of EC mode is the signal for CP to release all of the shadow page and segment tables and the copy of the virtual segment table.

There are some situations that require invalidating all of the shadow tables constructed by CP or even releasing and reallocating them. Whenever DMKPTR swaps out a page that belongs to a virtual relocating machine, it sets a bit in the VMBLOK indicating that all of the shadow page tables must be invalidated. Invalidation of all of the tables is required since CP does not know which third-level-storage pages map into the second-level page which is being swapped out. The actual invalidation is handled by DMKVATAB, called from DMKDSP when the virtual machine is on the verge of being dispatched.

The other situations which cause shadow-table invalidation arise from the simulation of privileged instructions in DMKPRV. Flags are set in the VMBLOK whenever the virtual machine loads either control register 0 or 1, and DMKPRV calls DMKVATAB to perform whatever maintenance is required. When control register 1 is loaded by the virtual machine, DMKVATAB must recopy the virtual segment table into first-level storage and invalidate the entire shadow segment table. When control register 0 is loaded, DMKVATAB examines the relocation-architecture control bits to determine if they have changed, (such that the format of the virtual page and segment tables no longer matches that of the shadow tables). If the format has not changed, the shadow tables are left intact; otherwise, all of the shadow tables and the copy segment table must be returned to free storage and another set, in the new format, must be allocated and initialized. The same actions can result from modifying the control registers via the CP console functions, in which case DMKVATAB is called from DMKCDB. The privileged operation, PTLB also causes the virtual segment tables to be recopied and all of the shadow page tables to be invalidated. since the shadow tables are the logical equivalent of the translation look-aside buffer.

DMKPRV provides virtual interrogation of the reference and change bits in the virtual storage keys, which involve the privileged instructions ISK, SSK, and RRB. The privileged instruction LRA is simulated via DMKVATLA, which searches the virtual page and segment tables to translate a third-level storage address to a second-level storage address, returning a condition-code indicator to DMKPRV, or forcing an interrupt if the tables are incorrectly formatted.

Most error situations that occur in the virtual machine are handled by means of the extended program interrupts associated with the real address translation hardware. Whenever a virtual relocating machine loads control registers 0 or 1 with an invalid value, DMKVAT releases all of the shadow tables and the copy segment table exactly as if the hardware controls had changed. The shadow control registers are set valid, with the shadow segment table re-allocated at a minimum size and all segments marked unavailable. Flag bits are set in the VMBLOK to indicate that the shadow tables are artificially valid, and DMKVATSX reflects a translation specification exception to the virtual machine as soon as it is dispatched. While it is possible for the virtual machine to enter an interrupt loop (if the new PSW is also a translate-mode PSW), the cited process prevents the occurrence of a disabled-loop within CP, which would result if the virtual machine is never dispatched.

PRIVILEGED INSTRUCTIONS

If the program interruption is caused by the virtual machine issuing a privileged instruction, DMKPRVLG obtains the address of the privileged instruction and determines the type of operation requested.

I/O Privileged Instructions

DMKPRVLG transfers control to the virtual I/O executive program (DMKVIOEX).

Non-I/O Privileged Instructions

DMKPRVLG simulates valid non-I/O privileged instructions and returns control to DMKDSPCH. For invalid privileged instructions, the routine sets an invalid interruption code and reflects the interruption to the virtual machine. For the privileged instructions SCK, SCKC, STCKC, SPT, and STPT that affect the TOD clock, CPU timer, and TOD clock comparator, control is transferred to DMKTMR by DMKPRVLG. Others that are simulated are LPSW, SSM, SSK, ISK, and diagnose.

System/370 EC mode privileged simulation includes the following:

Code	Definition
SCK	Set clock
SCKC	Set clock comparator
STCKC	Store clock comparator
SPT	Set CPU timer
STPT	Store CPU timer
STNSM	Store and AND system mask
STOSM	Store and OR system mask
STIDP	Store CPU identification
STIDC	Store channel identification
LCTL	Load control
STCTL	Store control
LRA	Load real address
RRB	Reset reference bit
PTLB	Purge table look-aside buffer

DIAGNOSE Interface (DMKHVC)

The diagnose command is used for communication between a virtual machine and the VM/370 control program. In VM/370, the machine-coded format for the diagnose command is:

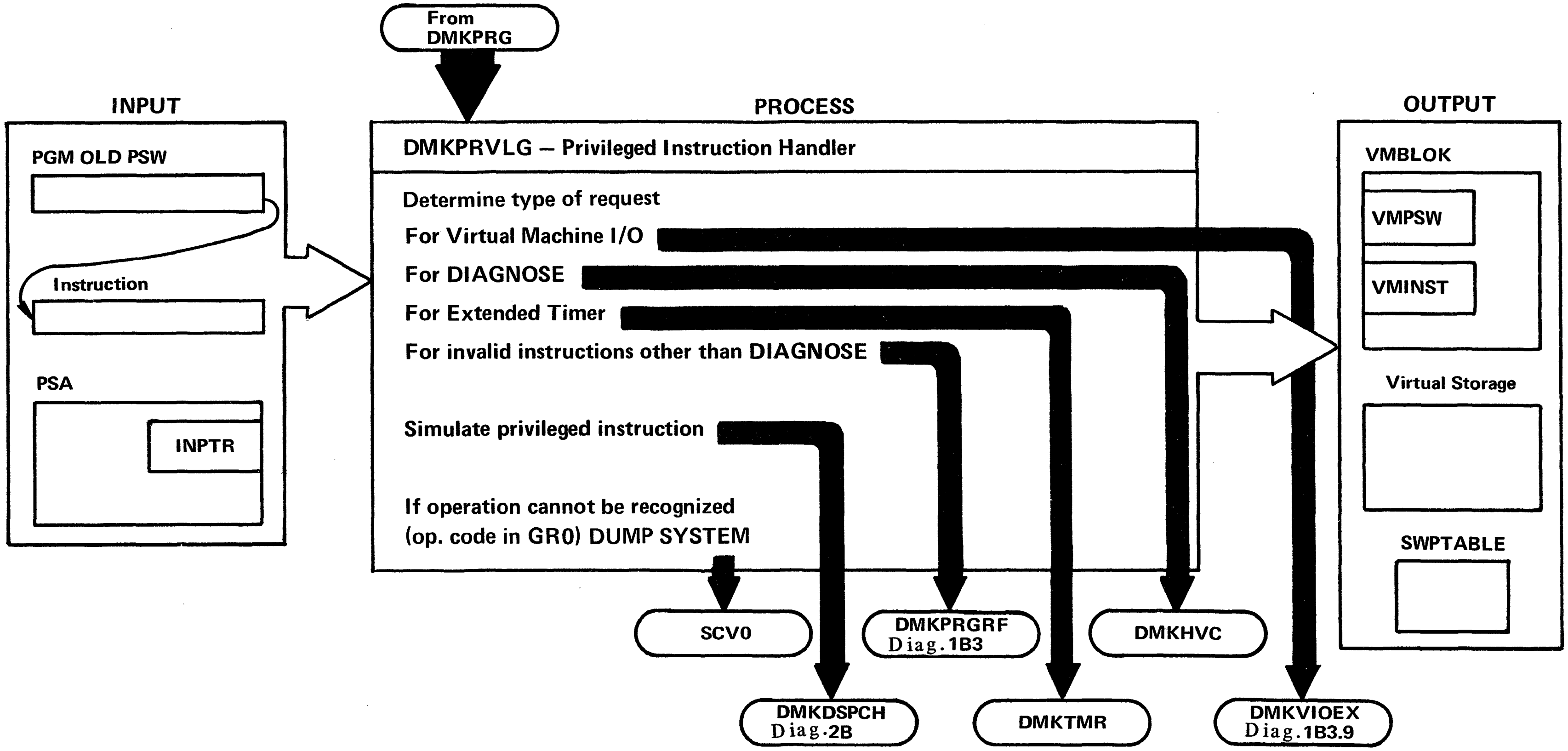
```
Bits  0     7 8    11 12   15 16     31
     ┌───────────────────────────────┐
     │  83  │  rx  │  ry  │  code    │
     └───────────────────────────────┘
```

83 is the Diagnose operation code.

rx is a user specified register number.

ry is a user specified register number.

Diag. 1B3.7. Privileged Instruction Simulation



Code is a hexadecimal value that is used to select a particular VM/370 control program function. The codes and their associated functions are:

Code	Class	Function
0004	C,E	Examine data from real storage
0008	G	Execute VM/370 control program console function
000C	G	Pseudo-timer facility
0010	G	Release virtual storage pages
0014	G	Manipulate input spool files
0018	G	Standard DASD I/O
001C	F	Clear I/O and machine check recording
0020	G	General virtual I/O without interrupts
0024	G	Virtual device type information
002C	C,E,F	Return DASD start of LOGREC area
0030	C,E,F	Read one page of LOGREC data
0034	C,F	Read system-dump spool file
0038	C,E	Read system symbol table
003C	A,B,C	Dynamically update system user directory
004C	Any	Generate accounting cards for virtual user

Notes: Rules for diagnose codes:

X'00' through X'FC'	Reserved for IBM use.
X'100' through X'1FC'	Reserved for users

The diagnose code must always be a multiple of 4.

DIAGNOSE CODE 4: Examine real storage, can only be issued by users with privilege class C or E.

rx contains the virtual address of a list of CP (real) addresses.

ry (cannot be register 15) contains a count of entries in the list.

ry+1 contains the virtual address of the result field that holds the values retrieved from the VM/370 control program locations.

DIAGNOSE CODE 8: Virtual console function, allows a virtual machine to perform the VM/370 control program console functions.

rx contains the address (virtual) of the control program console function command and parameters.

ry contains the length of the associated console function input, up to 132 characters.

The following illustrates the virtual console function:

```
         LA     R6,CPFUNC
         LA     R10,CPFUNCL
         DC     X'83',X'6A',XL2'0008'
         .
         .
         .
CPFUNC   DC     C'QUERY FILES'
CPFUNCL EQU     *-CPFUNC
```

The output of the console function is to the user's terminal, and then execution continues. Any valid and authorized console function can be executed in this manner.

A completion code is returned to the user as a value in the register specified in ry. The error code = the message number of the error message issued.

DIAGNOSE CODE C: Pseudo timer.

rx contains the virtual address of a 32-byte data area that does not cross a page boundary, into which the following data is stored:

```
Bytes 0       7 8      15 16    23 24     31
      ┌────────┬────────┬────────┬─────────┐
      │        │        │        │         │
      │MM/DD/YY│HH:MM:SS│Virt CPU│Total CPU│
      │        │        │        │         │
      └────────┴────────┴────────┴─────────┘
```

Virtual and Total CPU time used is returned as a doubleword logical value in microseconds.

DIAGNOSE CODE 10: Release pages.

rx contains the virtual address of the first page to be released.

ry contains the virtual address of the last page to be released.

Any of the virtual pages in real or auxiliary storage are released.

DIAGNOSE CODE 14: Input spool file manipulation.

rx contains either a buffer address, a copy count, or a spool-file identifier, dependent on the value of the function subcode in ry+1.

ry (cannot be register 15) contains the virtual address of a spool-input card reader.

ry+1 contains a function hexadecimal code interpreted by DMKDRDER, as follows:

Code	Function
0000	Read next spool buffer (data record)
0004	Read next print SFBLOK
0008	Read next punch SFBLOK
000C	Select a file for processing
0010	Repeat active file nn times
0014	Restart active file at beginning
0018	Backspace one record

ry+1 on return, may contain error codes which further define a returned condition code of 3. See Figure 1 for Condition Code analysis.

The file manipulation is performed by DMKDRDER.

DIAGNOSE CODE 18: Disk I/O.

rx contains the device address of the disk.

ry points to a CCW chain to read or write a limited number of disk records.

Each read or write must specify no more than 2048 bytes (usually 800 is used), and the CCW chain is of a standard form, as shown below. For a 3330, a SET SECTOR command would precede each SRCH command.

Register 15 contains the number of reads or writes in the CCW chain (the number is two in the following example for a typical CCW string (to read or write two 800-byte records):

```
    SEEK,A,CC,6
    SRCH,A+2,CC,5
    TIC,*-8,0,0
    RD or WRT,DATA,CC+SILI,800
    SEEK HEAD,B,CC,6 (Omitted if HEAD No. unchanged)
    SRCH,B+2,CC,5
    TIC,*-8,0,0
    RD or WRT,DATA+800,SILI,800
A   SEEK and SRCH arguments for first RD/WRT
B   SEEK and SRCH arguments for second RD/WRT
```

DIAGNOSE CODE 1C: Clear I/O recording, can only be issued by a privilege class F user. This code calls the DMKIOEFM routine to clear the I/O error recording data on disk.

rx contains the code value 1, 2, or 3 to clear and reformat the I/O error recording, M/C recording, or both I/O and M/C recording, respectively.

ry is ignored.

DIAGNOSE CODE 20: General I/O without interrupts.

rx contains a virtual device address.

ry contains the address of the string of CCWs to be executed.

The CCW string is processed via DMKCCWTR through DMKGENIO, providing full virtual I/O in a synchronous fashion (self-modifying CCW strings are not permitted, however) to any virtual device specified. Control returns to the virtual machine only after completion of the operation or detection of a fatal error condition. Condition codes and error codes in ry are returned to the virtual system.

DIAGNOSE CODE 24: Virtual device type information.

rx contains a virtual device address.

ry, which cannot be register 15, and ry+1 contain the following upon return:

Code	ry+1	Meaning
0		Data Transfer successful
1		End of file
2		File not found
3	4	Device address invalid
3	8	Device type invalid
3	12	Device busy
3	16	Fatal Paging I/O Error

Figure 1. Condition Code Analysis for Diagnose Codes 14 and 34

Bits	0 7	8 15	16 23	24 31
ry	VDEVTYPC	VDEVTYPE	VDEVSTAT	VDEVFLAG
ry+1	RDEVTYPC	RDEVTYPE	RDEVMDL	RDEVFIR

A condition code of one or three indicates that the virtual device address specified is either invalid (that is, too large), or the device does not exist. Condition code 2 indicates the real device does not exist.

DIAGNOSE CODE 2C: Return DASD start of LOGREC area (Privilege class C, E, or F only).

rx on return contains the DASD location, in VM/370 control program internal format, of the first record of the system I/O and machine check error recording area.

ry is ignored.

DIAGNOSE CODE 30: Read one page of LOGREC data (Privilege class C, E, or F only).

rx contains the DASD location, in the VM/370 control program internal format, of the desired record.

ry contains the virtual address of a page-size buffer to receive the data.

The page of data is provided to the virtual machine via DMKRPAGT.

cc = 0 Successful read, data available.
     1 End of cylinder, no data.
     2 Invalid cylinder, outside recording area

DIAGNOSE CODE 34: Read system dump spool file (privilege class C or E only).

rx contains the virtual address of a page-size buffer to accept the requested data.

ry (cannot be register 15) contains the virtual device address of a spool-input card reader.

ry+1 on return, may contain error codes which further define a returned condition code of 3. See Figure 1 for Condition Code analysis.

The system chain of spool input files is searched for a dump file belonging to the user issuing the diagnose command by DMKDRDMP. The first (or next) record from the dump file is provided to the virtual machine via DMKRPAGT and the condition code is set to zero. The dump file is closed via VM/370 console function CLOSE.

DIAGNOSE CODE 38: Read system symbol table.

rx contains the start address of the page buffer that is to contain the symbol table.

ry is ignored.

The system symbol table (DMKSYM) is read into storage at the location specified by rx by DMKDRDSY.

DIAGNOSE CODE 3C: Dynamically update the system user directory.

rx contains the first 4 bytes of the volume serial label.

ry, the first 2 bytes of the register specified (ry) contain the last 2 bytes of the volume serial label.

The directory if dynamically updated by DMKUDRDS.

DIAGNOSE CODE 4C: Generate accounting cards for the virtual user. This code can be issued only by a user with the account option (ACCT) in his directory.

rx contains the virtual address of a 24-byte parameter list identifying the "charge to" user; the address must be aligned on a doubleword boundary. If rx contains zeros, the accounting card will be punched with the identification of the user issuing the diagnose instruction.

ry contains a function hexadecimal code interpreted by DMKHVC as follows:

Code	Function
0000	The parameter list contains only a userid.
0004	The parameter list contains a userid and account number.
0008	The parameter list conatins a userid and distribution number.
000C	The parameter list contains a userid, account number, and distribution number.

The following condition codes are returned to the user by DMKHVC:

cc=0	Successful operation
1	User does not have account option privileges
2	Invalid userid in the parameter list
3	Invalid function hexadecimal code in ry or an error occurred in trying to read in the User Machine Block (UMACBLOK)

DMKHVC checks that the user has the account option and if not, returns a condition code of 1. If the user has the options, control is passed to DMKCPV to generate the card. DMKCPV passes control to DMKACO to complete the "charge to" information; either from the User Accounting Block (ACCTBLOK), if a pointer to it exists, or from the user's VMBLOK. DMKCPV then punches the card and passes control back to DMKHVC to release the storage for the ACCTBLOK, if one exists. DMKHVC then checks the parameter list address for the following conditions:

- If zero, control is returned to the user with a condition code of zero.
- If invalid, an addressing exception is generated.
- If not aligned on a doubleword boundary, a specification exception is generated.

For a parameter list address that is non-zero and valid, the userid in the parameter list is checked against the directory list and if not found, control is returned to the user with a condition code of two. If the function hexadecimal code is invalid, control is returned to the user with a condition code of three. If both userid and function hexadecimal code are valid, the User Accounting Block (ACCTBLOK) is built and the userid, account number, and distribution number are moved to the block from the parameter list or the User Machine Block belonging to the userid in the parameter list. Control is then passed to the user with a condition code of zero.

Virtual Timer Maintenance

The System/370 with EC mode provides the system user (both real and virtual) with four timing facilities. They are:

1. The interval timer at main storage location X'50'.
2. The time-of-day clock.
3. The time-of-day clock comparator.
4. The CPU timer.

REAL TIMING FACILITIES: Before describing how CP maintains these timers for virtual machines, it is necessary to review how VM/370 uses the timing facilities of the real machine.

1. The location X'50' interval timer is used only for time-slicing. The value placed in the timer is the maximum length of time that the dispatched user is allowed to execute.
2. The time-of-day clock is used as a time stamp for messages and enables the scheduler to compute

elapsed in-queue time for the dispatching priority calculation.

3. The time-of-day clock comparator facility is used by CP to schedule timer driven events for both control program functions and for virtual machines. A stack of comparator requests is maintained and as clock comparator interrupts occur, the timer request blocks are stacked for the dispatcher via calls to DMKSTKIO.

4. The CPU timer facility performs three functions:

   - Accumulation of CP overhead
   - Detection of in-queue time slice end
   - Virtual CPU timer simulation

   The accumulation of CP overhead is accomplished as follows. The VMTTIME field in the VMBLOK contains the total CP overhead incurred by the virtual machine; it is initialized to the maximum sized doubleword integer, X'7FFFFFFF FFFFFFFF'. Whenever CP is to perform a service for a virtual machine, GPR 11 is loaded with the address of the VMBLOK and the current value in VMTTIME is placed in the CPU timer. When CP is finished with the service for that virtual machine the CPU timer, which has been decremented by the amount of CPU time used, is stored back into VMTTIME. GPR11 is then loaded with a new VMBLOK pointer and the CPU timer is set from the new VMTTIME field. The amount of CP overhead for a given virtual machine at any point in time is the difference between the maximum integer and the current value in the VMTTIME field.

   Since VMTTIME only accounts for supervisor state overhead, detection of in-queue time slice end is performed by the CPU timer when the virtual machine is dispatched in the problem state. The VMTMOUTQ field in the VMBLOK is intialized to the amount of problem state time that the virtual machine will be allowed to accumulate before being dropped from a queue. This initial value is set by the scheduler (DMKSCH) when the virtual machine is added to a queue and its value depends on the queue entered (interactive or non-interactive) and on the CPU model. For example, the initial value of VMTMOUTQ for a user entering Q1 (interactive) on a model 145 is 300 milliseconds, while for the same user entering Q2 (non-interactive) it is 2 seconds. Each time the user is dispatched, the value in VMTMOUTQ is entered into the CPU timer; whenever the user is interrupted, the decremented CPU timer is stored into VMTMOUTQ prior to being set from the new VMTTIME. When the problem state time slice has been exhausted; a CPU timer interrupt occurs, the VMQSEND flag bit is set in the VMBLOK, and the scheduler drops the user from the queue. At each queue drop, the problem time used in-queue (the difference between VMTMOUTQ and the initial value) is added to the total problem time field (VMVTIME) in the VMBLOK.

   Virtual CPU timer simulation is handled for EC mode virtual machines if the value in their virtual CPU timer is less than that in VMTMOUTQ. In this case, the VMBLOK is flagged as "tracking CPU timer" and a CPU timer interrupt is interpreted as a virtual timer interrupt rather than as an in-queue time slice end.

VIRTUAL TIMING FACILITIES: Virtual location X'50' timers are updated by the elapsed CPU time each time the dispatcher has been entered after a running user has been interrupted. The size of the update is the difference between the value of the timer at dispatch (saved in QUANTUM at location X'54') and the value of the timer at the time of the interrupt (saved in QUANTUMR at location X'4C').

Virtual clock comparator requests are handled by the virtual timer maintenance routine DMKTMR. They are inserted into the general comparator request stack and the virtual machine is posted when the interrupt goes off.

Requests to set the virtual CPU timer place the new value into the ECBLOK. Requests to store it update the ECBLOK field by the virtual CPU time used since the last entry to dispatch and pass the value to the user. Requests to set the time of day clock are ignored.

A real interval or CPU timer is one which runs when the user is executing or is in a self-imposed wait state (that is, the wait bit is on in his virtual PSW). A real timer does not run if the user is in a CP pseudo-wait (for example, page wait or I/O wait) or if he can be run but is not being dispatched due to other user interaction. Real timers provide accurate interrupts to programs that depend on measurement of elapsed CPU and/or wait time. They do not accurately measure wall time -- the TOD clock must be used for this function.

An EC mode virtual machine with the Real Timer option has both a real interval timer and a real CPU timer. Real timer requests for waiting machines are maintained in the clock comparator stack. CPU timer requests are added to TOD clock value at the time that they are issued. Interval timer requests must have their units converted. In addition, if the virtual CPU timer contains a large negative value, then a real timer request is scheduled to occur when the virtual time turns positive, so that the pending timer interrupt can be unflagged. Comparator requests for real timer interrupts are inserted into the stack whenever a user enters a self-imposed wait. They are removed either when the user resumes execution or when he is forced (or places himself) into a pseudo wait.

Virtual I/O Requests

The function of the virtual I/O interface maintained by the control program is to provide to the software operating in the user's virtual machine the condition codes, CSW status information, and interrupts necessary to make it appear to the user software that it is in fact running on a real System/370. The virtual I/O interface consists of:

- A virtual I/O configuration of each active user represented by a set of I/O control blocks that are maintained in the Control Program's free storage. This configuration is built at LOGON time from information contained in the user's directory file, and can be changed by the user or the system operator.
- A set of routines that maintain in these blocks, the status of the virtual I/O configuration.
- Other system components to simulate/translate the channel programs provided by the user to initiate I/O on units in the real system's configuration.

VIRTUAL I/O CONTROL BLOCKS: The base for locating the I/O block structure is the user's Virtual Machine Block (VMBLOK). The VMBLOK contains a pointer to the start of three control block tables, and a table of 16 channel indexes. The control block tables contain one block for each of the virtual channels, control units, and devices that are defined for the user's virtual machine. The entries in the channel index table (VMCHTBL) contain the pointers to each channel defined for the user in the table of Virtual Channel Blocks (VCHBLOKs). Each VCHBLOK contains a table of pointers that point to the Virtual Control Unit Blocks (VCUBLOKs) for the control units attached to that virtual channel. Each VCUBLOK contains pointers to the Virtual Device Blocks (VDEVBLOKs) attached to the control unit. See Diag. 1B3.8 for an overview of the virtual I/O control blocks.

Thus, if given the unit address of any component in the form ccu, the appropriate control blocks representing each component in the subchannel path to the given unit is located via the indexing scheme.

VCHBLOK: There is one VCHBLOK for each virtual channel connected to the user's virtual CPU. Each VCHBLOK contains the channel address and flag indicating the channel type (selector, byte multiplexer or block multiplexer). The status of the channel and its attached units are represented by several status and mask bytes, as follows:

1. A status byte (VCHSTAT) indicates whether the channel is busy or has a channel class interrupt pending.
2. A halfword unit address identifies the unit causing the channel-class interrupt (if it is present).
3. A halfword mask (VCHCUINT) contains a bit map of the attached control units that have interrupt status pending. Following these status flags and masks is the table of indexes pointing to the attached VCUBLOKs; index entries representing addresses at which no control unit is attached have a value of -1.

VCUBLOK: There is one VCUBLOK for each control unit in the virtual configuration. These blocks are arranged in a table, and each contains, in addition to its base address, status flags similar to those in the VCHBLOK and a table of indexes to attached VDEVBLOKs. The status flags defined for the VCUBLOK differ from those for the VCHBLOK in that they can contain status for the control unit and also for a subchannel.

For example, if the VCUBLOK representing a 2803 Tape Control Unit is attached to a virtual selector channel, both the VCHBLOK and the VCUBLOK are marked busy. However, if the VCUBLOK is attached to a virtual byte multiplexer channel and is for a control unit on a selector subchannel of the multiplexer, the busy status of the channel is reflected in the VCUBLOK only. Thus the virtual multiplexer appears nonbusy to operations on other, nonshared subchannels.

VDEVBLOK: There is one VDEVBLOK in the configuration for each virtual device defined by the user. Each VDEVBLOK contains the device portion of the unit address, device status, and the virtual CSW for the last interrupt taken by the device. In addition, the VDEVBLOK contains device type specific information that allows the I/O translation and simulation routines to interpret the channel programs presented by the user. This information is not used by the I/O interface.

Since all virtual machines are run in the problem state, any attempt to issue a SIO instruction results in a program interrupt that indicates a privileged operation exception. This interrupt is handled by CP's first level program interrupt handler, DMKPRGIN. It determines if the virtual machine was in virtual supervisor state (problem state bit in the VIRTUAL PSW is zero). If so, the instruction causing the interrupt is saved in the VMBLOK for the virtual machine and control is transferred to the privileged instruction simulator, DMKPRVLG, via a GOTO.

Diag. 1B3.8 Virtual I/O Control Blocks

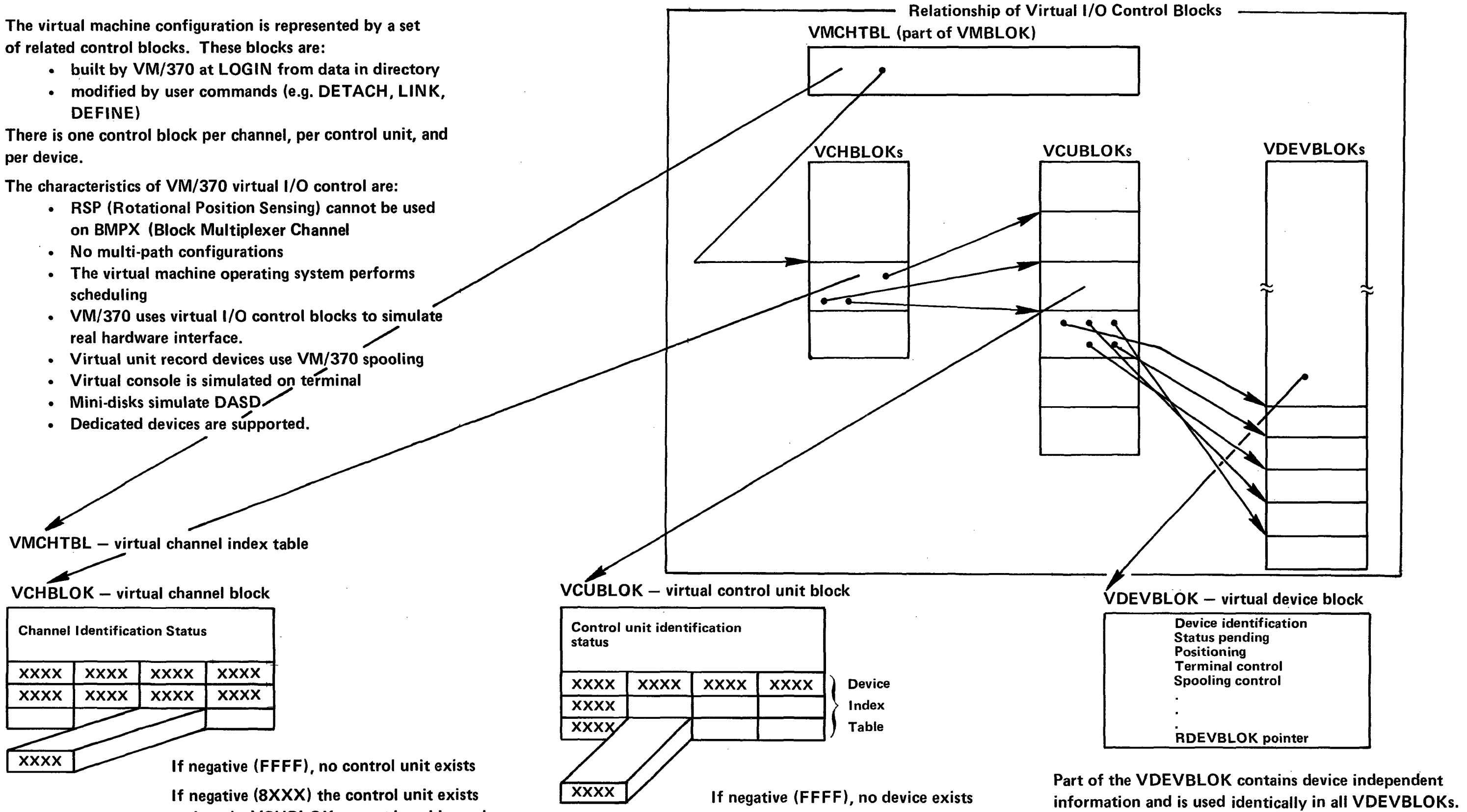
The virtual machine configuration is represented by a set of related control blocks. These blocks are:

- built by VM/370 at LOGIN from data in directory
- modified by user commands (e.g. DETACH, LINK, DEFINE)

There is one control block per channel, per control unit, and per device.

The characteristics of VM/370 virtual I/O control are:

- RSP (Rotational Position Sensing) cannot be used on BMPX (Block Multiplexer Channel
- No multi-path configurations
- The virtual machine operating system performs scheduling
- VM/370 uses virtual I/O control blocks to simulate real hardware interface.
- Virtual unit record devices use VM/370 spooling
- Virtual console is simulated on terminal
- Mini-disks simulate DASD
- Dedicated devices are supported.


Relationship of Virtual I/O Control Blocks
VMCHTBL (part of VMBLOK)
VCHBLOKs
VCUBLOKs
VDEVBLOKs
VMCHTBL – virtual channel index table
VCHBLOK – virtual channel block
Channel Identification Status
XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX
XXXX
VCUBLOK – virtual control unit block
Control unit identification status
XXXX XXXX XXXX XXXX
XXXX
XXXX
XXXX
Device Index Table
VDEVBLOK – virtual device block
Device identification
Status pending
Positioning
Terminal control
Spooling control
.
.
.
RDEVBLOK pointer

If negative (FFFF), no control unit exists

If negative (8XXX) the control unit exists but the VCUBLOK cannot be addressed by the virtual machine because the control unit is detached.

If positive, the value is on index to the VCUBLOK.

If negative (FFFF), no device exists

If negative (8XXX) the device exists but the VDEVBLOK cannot be addressed by the virtual machine because the device is detached.

If positive, the value is an index to the VDEVBLOK.

Part of the VDEVBLOK contains device independent information and is used identically in all VDEVBLOKs. However, some fields of the VDEVBLOKs have multiple uses, depending on the device type.

DMKPRVLG determines if the privileged operation affects the virtual I/O configuration. DMKPRVLG simulates non-I/O privileged instructions (such as LPSW) itself. If the instruction's operation code is from X'9C to X'9F', control is transferred to DMKVIOEX.

After clearing the condition code in the user's VMBLOK, DMKSCNVU is then called to locate the Virtual I/O blocks representing the components (channel, control unit and device) addressed by the instruction. DMKVIOEX then branches to handle the request based on the operation requested.

VIRTUAL SIO (See Figure 2): With a SIO, the condition code returned from DMKSCNVU is tested to verify that all addressed components were located. If they were not, then a condition code of 3 (unit not available) is reflected in the VPSW and control is returned to the dispatcher. Otherwise, the addresses of the appropriate virtual I/O control blccks are saved, and DMKVIOEX tests the status of the addressed I/O units by scanning the VCHBLOKs, VCUBLOKS, and VDEVBLOKs to locate the block that contains the status of the addressed subchannel. The subchannel status is indicated in:

- The VCHBLOK for a selector or block multiplexer channel

- The VCUBLOK for a shared selector subchannel on a byte multiplexer.

- The VDEVBLOK for a ncnshared subchannel on a byte multiplexer.

When the block containing the status is found, the status is tested. If the subchannel is busy or has an interrupt pending, condition code 2 is reflected. Otherwise, the subchannel is available and the device and the control unit are tested for interrupt pending or busy. If either is found, condition code 1 is reflected and the proper CSW status is stored in the user's virtual page zero. If all components in the subchannel path are free, DMKVIOEX proceeds to simulate the SIO by locating and loading the contents of the user's CAW from his virtual location X'48' and testing the device type of the unit addressed.

The device type is determined by referencing the VDEVBLCK. If the device class code indicates a terminal or ccnsole, control is passed to the virtual console executive DMKVCNEX via a GOTO. DMKVCNEX interprets and simulates the entire channel program, moving the necessary data to or from the user's virtual storage and reflecting the proper interrupts and status bytes. When DMKVCNEX has finished, it passes control directly to the dispatcher DMKDSPCH.

If DMKVIOEX determines that the referenced device is a spooled unit-record device, it passes control to DMKVSPEX for additional processing and upon return it passes control to DMKDSPCH.

If the referenced device is not a terminal nor a spooling device, the SIC is translated and executed directly on the real system's I/O device. DMKVIOEX calls DMKFREE to obtain free storage and then it ccnstructs an IOBLOK in the storage obtained. The ICBLOK serves as an identifier of the I/O task to be performed. It contains a pointer to the channel program to be executed and the address of the routine that is tc handle any interrupts associated with the operation.

DMKVIOEX stores the contents of the user's CAW in IOBCAW and sets the interrupt return address (IOBIRA) tc the virtual interrupt return address (DMKVIOIN) in DMKVIO. The CCW translation routine (DMKCCWTR) is then called to locate and bring into real main storage all user pages associated with the channel program, including those containing data and CCWs.

1. The CCWs are translated.

2. A corresponding real channel program is constructed.

3. The data pages are locked into real storage.

4. DMKCCWTR returns control to DMKVIOEX. DMKVIOEX places the user in a pseudo-wait state, IOWAIT, and calls the real I/O scheduler DMKIOSQV to schedule the I/O on the real configuration.

DMKIOSQV queues the request for operation on the real channel, control unit, and device corresponding to the one addressed by the user. When the real SIO is issued, DMKIOS takes the user out of IOWAIT and reflects the condition code for the SIO if it is zero. If it is not zero, the operation is further analyzed by DMKVIOIN. In any case, DMKIOSQV returns control to DMKVIOEX, which passes control to DMKDSPCH.

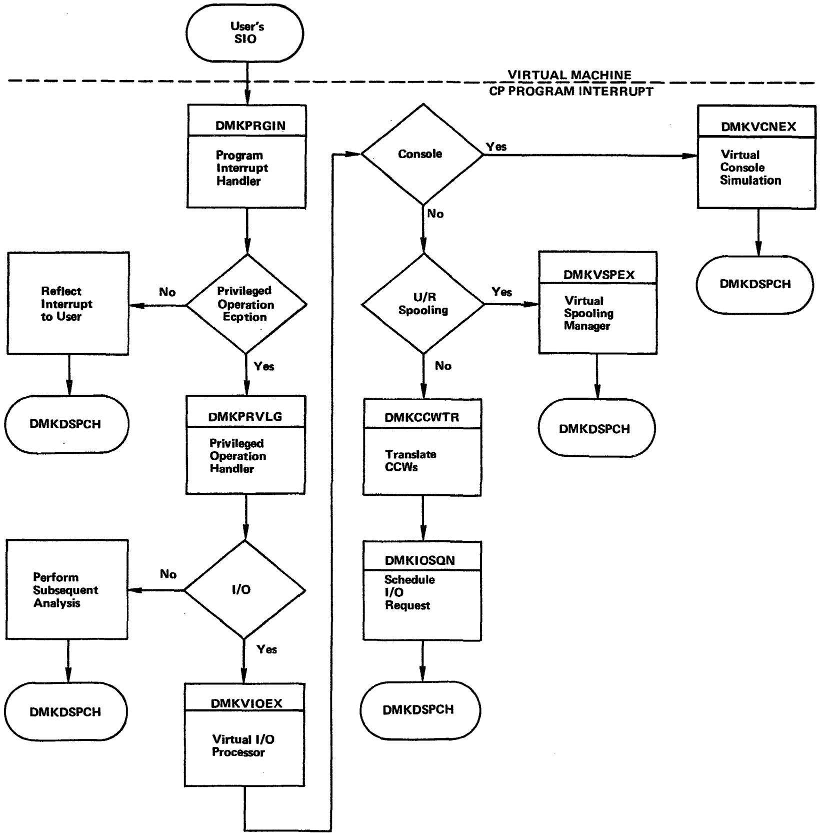


Figure 2. Overview of a Virtual SIO

OTHER VIRTUAL I/O INSTRUCTIONS: Other privileged I/O instructions are handled directly by DMKVIOEX. The general method used is to scan the virtual channel, control unit, and device blocks in the same manner as for the SIO and to reflect the proper status and condition to the user. In some cases (TIO), the status of the addressed components are altered after the status is presented.

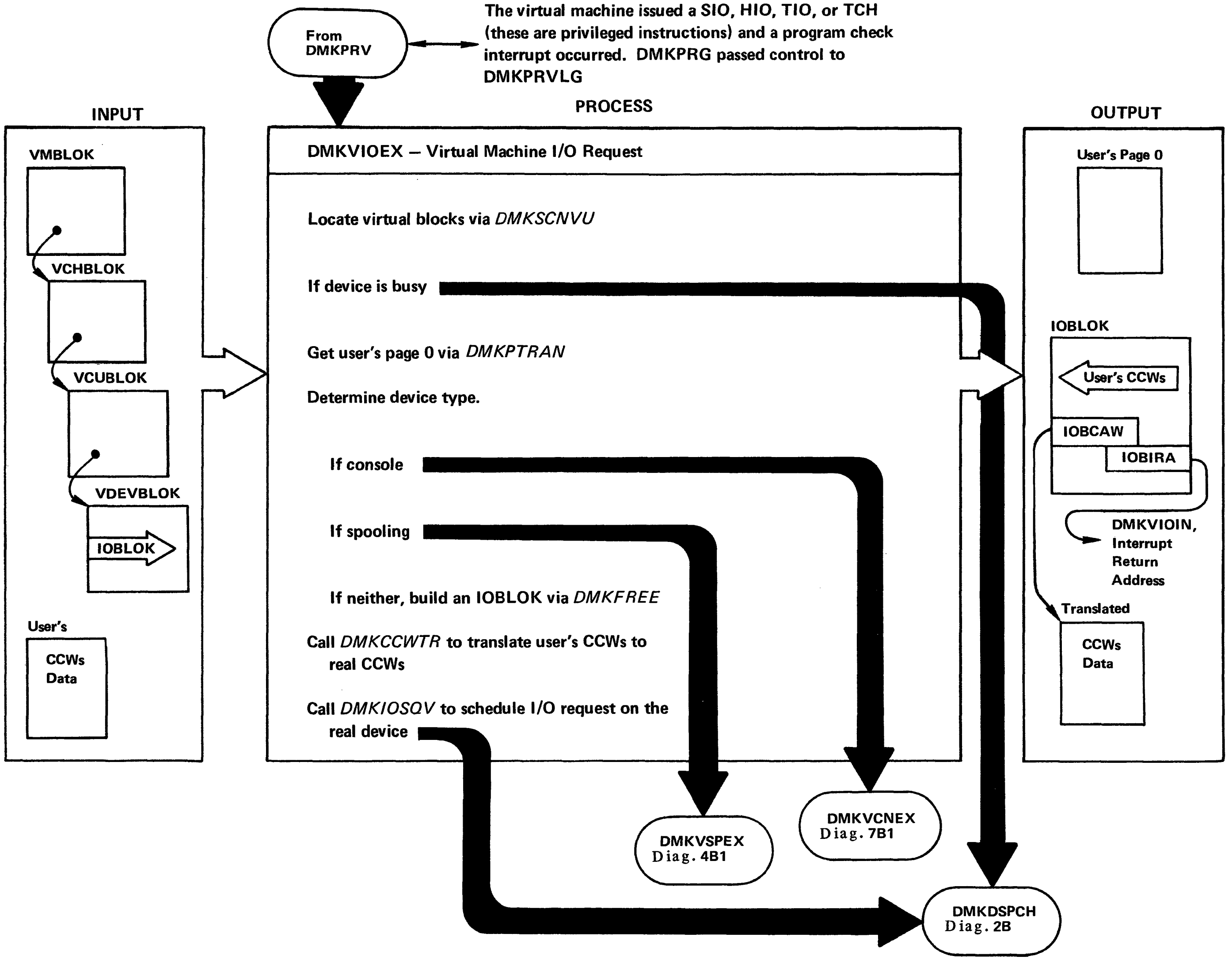
If the operation active on the virtual device is actually in progress in the real equipment, the simulation of a HIO or HDV is somewhat more involved, since it requires the actual execution of the instruction. In this case, the active operation is halted and the resultant condition code/status is returned to the user.

VIRTUAL CHANNEL-TO-CHANNEL ADAPTER: The virtual channel-to-channel adapter (CTCA) is simulation that permits data transfer and control communication between two selector channels, either on two distinct processors or two channels on a single processor. Data transfer is accomplished via synchronized complementary I/O commands (for example, read/write, write/read) issued to both parts of the CTCA. Each part of the CTCA is identical and the operation of the unit is completely symmetrical. The CTCA occupies an entire control-unit slot on each of the two channels attached. The low-order four bits of the unit address (device address) are ignored completely and are not available for use.

The VM/370 control program support for virtual CTCA includes all status, sense data, and interrupt presentation logic necessary to simulate the operation of the real CTCA. Data transfer, command byte exchange, sense data, and status data presentation for the virtual CTCA is accomplished via storage-to-storage operations (MVCL, etc.). No real I/O operations (excluding paging I/O) nor I/O interrupts are involved. Unit errors or control errors cannot occur.

VIRTUAL SELECTOR CHANNEL I/O REQUESTS: The CCW translator, DMKCCWTR, is called by the virtual machine I/O executive program (DMKVIOEX) when an I/O task block has been created and a list of virtual CCWs associated with a user's SIO request must be translated into real CCWs.

Diag. 1B3.9. Virtual I/O Request


From DMKPRV
The virtual machine issued a SIO, HIO, TIO, or TCH (these are privileged instructions) and a program check interrupt occurred. DMKPRG passed control to DMKPRVLG
INPUT
VMBLOK
VCHBLOK
VCUBLOK
VDEVBLOK
IOBLOK
User's
CCWs
Data
PROCESS
DMKVIOEX – Virtual Machine I/O Request
Locate virtual blocks via *DMKSCNVU*
If device is busy
Get user's page 0 via *DMKPTRAN*
Determine device type.
If console
If spooling
If neither, build an IOBLOK via *DMKFREE*
Call *DMKCCWTR* to translate user's CCWs to real CCWs
Call *DMKIOSQV* to schedule I/O request on the real device
OUTPUT
User's Page 0
IOBLOK
User's CCWs
IOBCAW
IOBIRA
DMKVIOIN, Interrupt Return Address
Translated
CCWs
Data
DMKVSPEX Diag. 4B1
DMKVCNEX Diag. 7B1
DMKDSPCH Diag. 2B

When the I/O operation from a self-modifying channel program is completed, DMKUNTIS is called by DMKIOS. When retranslation of OS ISAM CCWs is required, the self-modifying channel program checking portion of DMKCCWTR calls DMKISMTR.

DMKCCWTR operates in two phases:

- A scan and a translate phase.

- A TIC-scan phase, if the ISAM option was chosen.

A self-modifying channel program checking function is also included.

The scan and translate phase analyzes the virtual CCW list. Some channel commands require additional doublewords for control information (for example, seek addresses). Additional control words are also allotted (in pairs) if the data area specified by a virtual CCW crosses 4096-byte page boundaries, or if the virtual CCW includes an IDA (Indirect Data Address) flag.

Space is obtained from DMKFREE for the real CCW list, and the translation phase then translates the virtual CCW list into a real CCW list. TIC commands that cannot be immediately translated are flagged for later processing by the TIC-scan phase. A read or write command that specifies data crossing 4096-byte boundaries is revised to include an IDA flag that points to an Indirect Data Address List (IDAL) and a pair of words for each 4096-byte page, in which each word handles a data-transfer of 2048 bytes (or less). The real CCW is flagged as having a CP-generated IDA. DMKPTRAN is called (via the TRANS macro) to lock each 4096-byte page.

If the real CCW string does not fit in the allocated free storage block, a new block is obtained. The old block is transferred and adjusted before being released. The translation continues with the new block. The process is repeated as needed to contain the real CCW string.

Virtual CCWs having and IDA flag set are converted to use translated addresses for each IDAW (Indirect Data Address Word) in the virtual IDAL. DMKPTRAN is called for each IDAW. The CCW flagged as having a user (but not CP) generated IDA.

The TIC-scan phase scans the real CCW list for flagged (untranslated) TIC commands and creates a new virtual CCW list fcr the untranslated commands. Scan-translate phase processing is then repeated. When all virtual CCWs are translated, the virtual CAW in the IOBLOK task block is replaced by the real CAW (that is, a pointer to the real CCW list created by DMKCCWTR), and DMKCCWTR returns control to DMKVIOEX. The user protection key is preserved.

<u>OS ISAM Handling by DMKISMTR</u>: Because many of the OS PCP, MFT, and MVT ISAM channel programs are self-modifying, special handling is required by the VM/370 control program to allow virtual machines to use this access method. The particular CCWs that require special handling have the following general format:

```
    0        2        4        6        8
   ┌───────────────────────────────────┐
A  |      READDATA  C+7  10 bytes      |
   |--------|--------|--------|--------|
B  |               TIC to E            |
   |--------|--------|--------|--------|
C  |        |        |        |        |
   |--------|--------|--------|--------|
D  |        |        |        |        |
   |--------|--------|--------|--------|
E  |         SEEK: SEEK head on D      |
   |--------|--------|--------|--------|
F  |            SEARCH on D+2          |
   └───────────────────────────────────┘
```

The CCW at A reads 10 bytes of data, the last byte of which forms the command code of the CCW at E. In addition, the data read in forms the seek and search arguments for the CCWs at E and F. After the CCW string is translated by the VM/370 control program it usually is in the following format:

```
     0        2        4        6        8
     +-----------------------------------+
  1  |      READDATA  C+7  10 bytes      |
     |--------|--------|--------|--------|
  2  |             TIC to 3              |
     +-----------------------------------+

     +-----------------------------------+
  3  |       SEEK: SEEK head on 6        |
     |--------|--------|--------|--------|
  4  |            SEARCH on D+2          |
     |--------|--------|--------|--------|
  5  |        |      etc.       |        |
     |--------|--------|--------|--------|
     |        |        |    ISAM word    |
     +-----------------------------------+
```

In order to accomplish an efficient and non-timing dependent translated operation for OS ISAM, the virtual CCW string is modified in the following manner.

DMKISMTR is called by DMKCCWTR if, during normal translation, a CCW of the type at 1 is encountered. The scan program locates the TIC at 2 by searching the translated CCW strings. The TIC at 2 locates the seek at 3.

The virtual address of the virtual seek CCW at E is located from the RCWTASK header. Three doublewords of free storage are obtained and the address of the block is saved in the ISAM control word at 5. The three doublewords are used to save the following information from the translated CCW strings and from the users virtual storage.

Before	After
address of TIC at 2	first word of TIC at 2
address of SEEK at E	first word of SEEK at E
first word of CCW at F	address of CCW at 4

The TIC at 2 is altered to TIC to the virtual CCW at E. The CCW address field at E is translated to reference D. The four bytes at F are modified to a TIC to the CCWs starting at 4. The completed CCW string has the following format:

```
     0        2        4        6        8
     +-----------------------------------+
  1  |      READDATA  C+7  10 bytes      |
     |--------|--------|--------|--------|
  2  |             TIC to E              |
     +-----------------------------------+

     +-----------------------------------+
  3  |        |    not used     |        |
     |--------|--------|--------|--------|
  4  |        |  SEARCH on D+2  |        |
     |--------|--------|--------|--------|
  5  |        |      etc.       |        |
     |--------|--------|--------|--------|
  6  |        |        |    ISAM WORD    |
     +-----------------------------------+
```

TRANSLATED CCWs

```
     0        2        4        6        8
     +--------|--------|--------|--------+
  A  |      READDATA  C+7  10 bytes      |
     |--------|--------|--------|--------|
  B  |             TIC to E              |
     |--------|--------|--------|--------|
  C  |        |        |        |        |
     |--------|--------|--------|--------|
  D  |        |        |        |        |
     |--------|--------|--------|--------|
  E  |       SEEK: SEEK head on D        |
     |--------|--------|--------|--------|
  F  |             TIC to 4              |
     +-----------------------------------+
```

This interrupt return address in the IOBLOK is set to DMKUNTIS. DMKUNTIS restores the data to its original format from the three doubleword extension and releases the block. Normal I/O handling is resumed by DMKVIO and DMKUNT.

I/O SUPERVISOR

The module DMKIOS handles the I/O requirements of all system devices except for the low-speed lines that serve as user logon consoles. Scheduling and interrupt supervision for these devices is essentially a synchronous process and does not require the queuing and restart services of DMKIOS; it is therefore handled by the module DMKCNS.

REAL I/O CONTROL BLOCKS

In order to control the activity of the I/O devices of the system and schedule I/O requests upon them, I/O control uses several types of control blocks. These blocks can be separated into two basic types:

- Static blocks that describe the components of the I/O system.
- The dynamic blocks that represent active and pending requests for I/O operations.

The I/O components of the real system are described by one control block for each channel, control unit, and device available to the control program. Units present but not represented by control blocks are not available for either user initiated or control program initiated operations.

RCHBLOK: For each channel attached to the system there exists a Real Channel Control Block (RCHBLOK) which contains:

- The channel portion of the address of its attached units,
- Status flags reflecting its availability for scheduling.
- A two-way queue anchor pointing to the list of I/O requests waiting for its services.

In addition, each RCHBLOK contains 32 half-word indexes, arranged in ascending address order, that represent the displacement into the Real Control Unit table of the control blocks for the control units attached to the channel. The 32 entries are required because the control unit address may be made up of 5 bits from the unit address. To locate the control block for a given unit, it is only necssary to:

- Index into the table in the RCHBLOK a displacement equal to twice the control unit address.
- Load the index value.
- Add the value to the base address of the Real Control Unit Table.

RCUBLOK: The Control Unit Table is composed of Real Control Unit Blocks (RCUBLOK), one for each Control unit on the system. These blocks are similar to the RCHBLOK in that they contain the control unit portion of the address and status flags, and a pointer to a queue of I/O requests. In addition the RCUBLOK contains a pointer to the RCHBLOK for the channel to which it is attached. The RCUBLOK contains a table of 16 halfword entries that represent the displacment into the Real Device Table of its attached devices. This table is referenced in the same manner as the table in the RCHBLOK.

RDEVBLOK: Each device in the system is represented by a Real Device Control Block (RDEVBLOK), contains the device portion of the unit address and status flags similar to those in RCHBLOK and RCUBLOK. There is also a pointer for those operations that are waiting for the device to become available. Fields that appear in the RDEVBLOK and not in the other blocks include a pointer to the I/O request that is currently active on the device, SIO counts, and a pointer to error and sense information. The RDEVBLOK contains a pointer to the RCUBLOK for the control unit to which it is attached and fields of device dependent information which do not affect the operation of I/O control.

Diag. 1B4.0 Real I/O Control Blocks

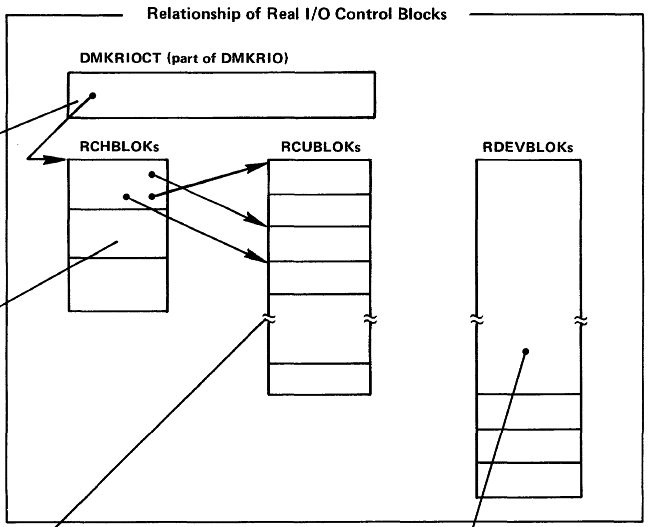
The real machine configuration is represented by a set of related control blocks. These blocks are:

- part of the VM/370 nucleus
- built from macros in assembly of DMKRID
- loaded at system IPL and initialized then for operation.

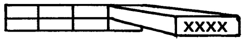
There is one control block per channel, per control unit, and per device.

The characteristics of VM/370 real I/O control are:

- Block multiplexing (BMPX) with RPS (Rotational Position Sensing) is used.
- Multi-path scheduling (2 channel switching) is not used.
- All I/O operations are handled by VM/370 scheduling and interrupt handling.


Relationship of Real I/O Control Blocks
DMKRIOCT (part of DMKRIO)
RCHBLOKs
RCUBLOKs
RDEVBLOKs

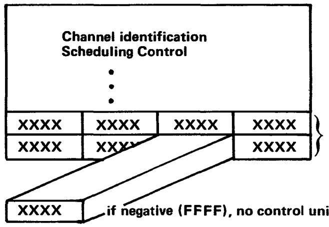
DMKRIOCT – Real Channel Table
See Appendix X for a complete description



—negative value (FFFF) indicates no channel exists

—positive value is an index to the RCHBLOK

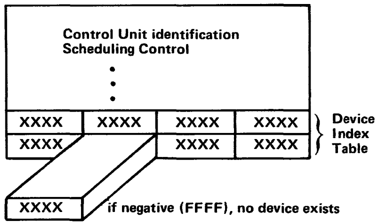
RCHBLOK – Real Channel Block



if negative (FFFF), no control unit exists

if positive, that value is an index to tne RCUBLOK

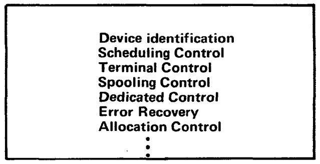
RCUBLOK – Real Control Unit Block



if negative (FFFF), no device exists

if positive, that value is an index to RDEVBLOK

RDEVBLOK – Real Device Block



Part of the RDEVBLOK pertains to functions that are device independent; that part of the RDEVBLOK is used in the same way for all devices. However, some of the fields in the RDEVBLOK have multiple uses, depending on the device type and function.

IOBLOK: I/O requests that are active in the system are represented by IOBLOKs. There is one IOBLOK for each operation (that is, channel program) to be executed. The IOBLOK is constructed by the requesting task and contains such information as:

- The identity of the requestor
- The address of the channel program to be executed
- The address to which control is to be returned upon completion of the operation

In addition, the IOBLOK contains status flags that indicate the current state of the operation (such as, whether or not an error has occurred, if an Error Recovery Procedure (ERP) is in control, and the condition returned frcm the SIO) and the CSW associated with the interrupt that signals the end of the operation. Since IOBLOKS are queued off various I/O control blocks, they also contain forward and backward queue pointers. DMKIOS builds in them the real device address of the unit on which the operation is started.

In general, the IOBLOK representing a given operation progresses through the system by being queued, in turn, from device, control unit, and channel blocks until a path is at last free to the device. A SIO is then issued. After the operation is complete, the IOBLOK is dequeued from the RDEVBLOK and stacked on a queue maintained in the dispatcher, DMKDSP. Each time the dispatcher is entered, the entries on the queue are unstacked and control is passed to the point specified in the Interrupt Return Address (IOBIRA). After I/O control stacks the IOBLOK for the given task, it attempts to restart all of the components that have been freed by the completion of the operation.

I/O COMPONENT STATES

The I/O components represented by the control blocks described in the section "Real I/O Control Blocks" are in one of four states and the state is indicated by the flag bits in the block status byte. If the component is not DISABLED, it is either BUSY, SCHEDULED, or AVAILABLE.

If the DISABLED bit is on, the component has been taken offline by the operator or the system and is at least temporarily unavailable. A request to use a disabled component causes the IOBLOK to be stacked with an indication of condition code 3 on the SIO and the real SIO is not performed.

A component is BUSY if it is transferring data (in the case of a channel or control unit), or if it is in physical motion (in the case of a device). If a component is BUSY, the IOBLOK for the request is queued from the control block representing that component.

A component is SCHEDULED if it is not BUSY but will become BUSY after a higher level component in the subchannel path becomes available and an operation is started. For example, if a request is made to read from a tape drive and the drive and control unit are available, but the channel is BUSY, the IOBLOK for that request is queued from the RCHBLOK for the BUSY channel and the RCUBLOK and RDEVBLOK of the drive and control unit are marked SCHEDULED. Future requests to that drive are queued from the RDEVBLOK for the SCHEDULED device. When the channel completes the operation, the next pending operation is dequeued and started; the SCHEDULED control unit and device are then marked BUSY.

The IOBLOKs for various I/O requests indicate the status of that request by a combination of the status bits in the IOBLOK and the queue in which the block resides. In general, an IOBLOK is queued from the control block of the highest level componenent (taken from device up to channel) in the subchannel path that is not available. Once the I/O operation is started, the IOBLOK is chained from the active IOBLOK pointer (RDEVAIOB) in the Real Device Control Block. Flags in the IOBLOK status fields may also indicate that a unit check has occurred, that a sense is in progress, or that a fatal I/O error (unrecoverable) has been recognized by ERP. After I/O control releases control of the IOBLOK, it is stacked on the queue of IOBLOKs and CPEXBLOKs anchored at DMKDSPRQ in the dispatcher and control is passed to the second level interrupt handler whose address is stored in IOBIRA.

I/O INTERRUPTS

I/O interrupts are usually either synchronous or asynchronous. Asynchronous interrupts indicate the change in status of an I/O component from the not-ready to ready state or busy to not-busy state. In either case, if the affected component has any pending requests queued from its control block, they are restarted and whether or not the given interrupt is processed any further depends upon the status of the interrupting component. Channel available and control unit end type interrupts restart the interrupting component. An asynchronous device end is passed to the user if the device is dedicated; otherwise, the device is restarted.

An interrupt is considered to be synchronous if the interrupting device has a nonzero pointer to an active IOBLOK. In this case the processing that occurs is as follows:

- If a unit check has occurred, a SENSE is scheduled, and when the SENSE is completed, the appropriate ERP is called.

- If an ERP is currently in control of the task (indicated by a flag in the IOBLOK), return the IOBLOK to the appropriate ERP.

- If the operation is incomplete (for example, channel end is received without device end), the IOBLOK is copied and the copy is stacked but the original IOBLOK remains attached to RDEVAIOB to receive the final interrupt; then, the control unit and the channel is restarted.

- If the operation is complete (that is, the device is available), the IOBLOK is unhooked from the device and stacked, and the device, control unit and channel are restored.

The restart operation usually dequeues the next IOBLOK that is queued to the restarted component and queues it to the next higher component in the subchannel path. When the channel level is reached, a SIO is issued and exit is taken to the dispatcher after handling any non-zero condition codes as previously described.

<u>DASD Error Recovery, ERP (DMKDAS)</u>

Error recovery is attempted for VM/370 control program initiated I/O operations to its supported devices and for user-initiated operations to control program supported devices which use a diagnose interface. The primary control blocks used for error recovery are the RDEVBLOK, the IOBLOK and the IOERBLOK. In addition, auxiliary storage is sometimes used for recovery channel programs and sense buffers.

The initial error is first detected by the I/O interrupt handler which performs a SENSE operation if a unit check occurs. Unit check errors are then passed to an appropriate ERP. If a channel check is encountered, the channel check interrupt handler determines whether or not retry is possible and pass control to an ERP through the I/O interrupt handler. DASD errors are processed as described below.

CHANNEL ERRORS

- Channel control check is treated as seek check. It is retried 10 times.

- Interface control check is treated as seek check. It is retried 10 times.

- Channel data check is treated as data check. It is retried 10 times.

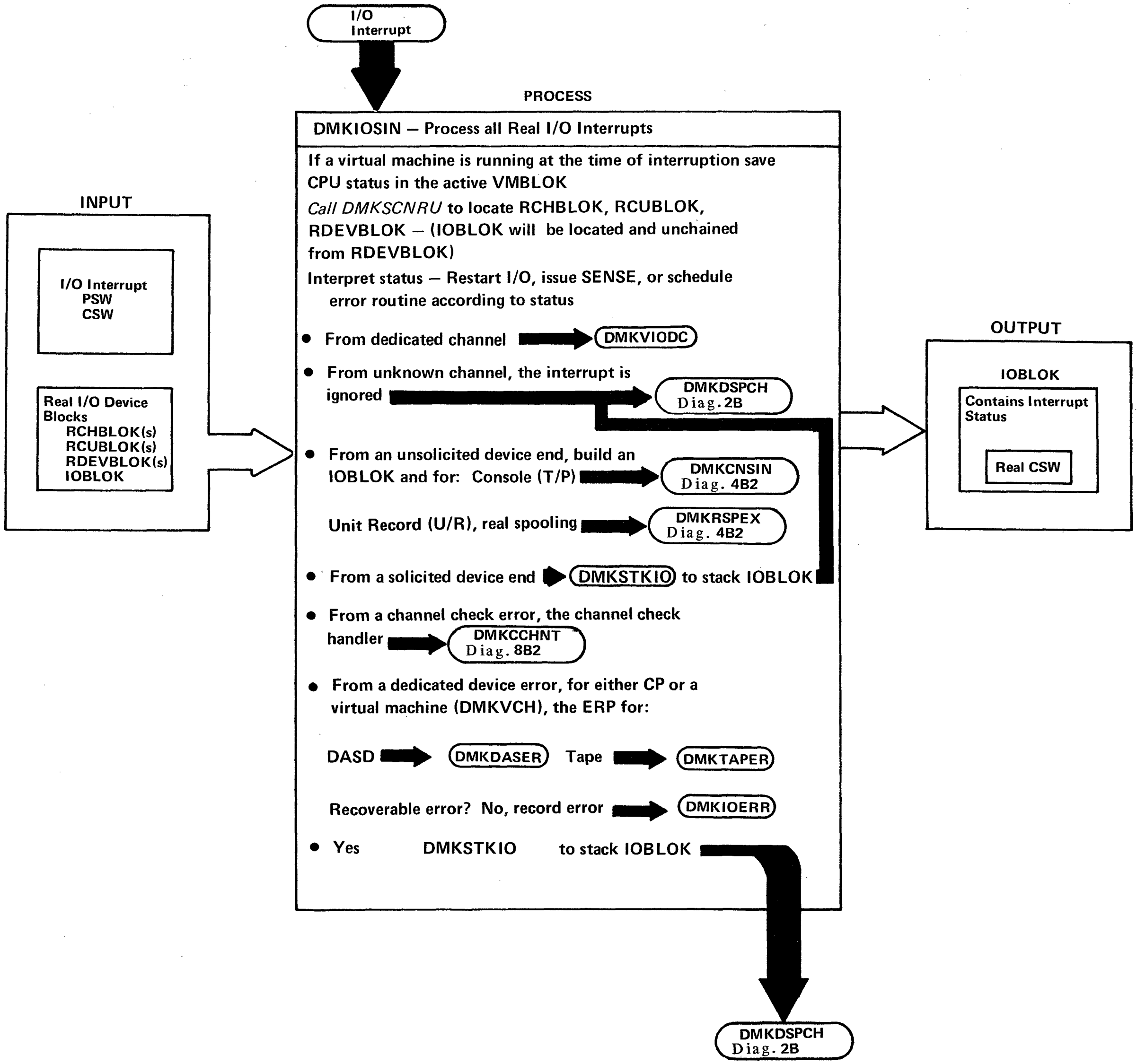
UNIT CHECK ERRORS

<u>Equipment check</u>: Retry the operation once.

<u>No record found and missing address marker</u>: Recalibrate and retry the channel program 10 times.

<u>No record found</u>: Execute a READ HOME ADDRESS and check home address against seek address. If they are the same, consider the error permanent. If they are not equal recalibrate and retry the channel program 10 times.

Diag. 1B4.1. I/O Interrupt Handler



DMKIOSIN – Process all Real I/O Interrupts

If a virtual machine is running at the time of interruption save CPU status in the active VMBLOK

Call DMKSCNRU to locate RCHBLOK, RCUBLOK, RDEVBLOK – (IOBLOK will be located and unchained from RDEVBLOK)

Interpret status – Restart I/O, issue SENSE, or schedule error routine according to status

- From dedicated channel → DMKVIODC
- From unknown channel, the interrupt is ignored → DMKDSPCH Diag. 2B
- From an unsolicited device end, build an IOBLOK and for: Console (T/P) → DMKCNSIN Diag. 4B2

  Unit Record (U/R), real spooling → DMKRSPEX Diag. 4B2
- From a solicited device end → DMKSTKIO to stack IOBLOK
- From a channel check error, the channel check handler → DMKCCHNT Diag. 8B2
- From a dedicated device error, for either CP or a virtual machine (DMKVCH), the ERP for:

  DASD → DMKDASER Tape → DMKTAPER

  Recoverable error? No, record error → DMKIOERR
- Yes DMKSTKIO to stack IOBLOK

Seek check: Retry the operation 10 times.

Intervention required: Issue a message to console and wait for solicited device end. This procedure will be repeated once.

Bus out check: Retry the operaticn 10 times.

Data checks: Retry the operation 256 times, with a recalibrate being executed every 16th try.

Overrun: Retry the operation 10 times.

Missing address marker: Retry the operation 10 times.

Command reject: Retry the operation once.

Chaining check: Retry the operation 10 times.

Environmental data present: Issue a buffer unload command and retry the operation.

Track condition check: This error should not occur. The VM/370 control program dces not use alternate tracks in its paging or spooling management. When a disk pack is formatted, any track that is marginal is marked as permanently allocated and, therefore, made unavailable for use by the VM/370 control program.

The error recovery routine keeps track of the number of retries in the IOBRCNT field of the IOBLOK. This count is used to determine if a retry limit has been exeeded for a particular error. On initial entry from DMKIOS for an error condition, the count is zero. Each time a retry is attempted the count is increased by one.

The ERP preserves the original error CSW and sense information by placing a pointer to the original IOERBLOK in the RDEVBLOK. Additional IOERBLOKs, which are received from DMKIOS on failing restart attempts are discarded. The original IOERBLOK is thus preserved for recording purposes.

If the specified number cf retries fails to correct the error situation, the operator is notified and control is returned to DMKIOS. DMKIOS is notified of the permanent error by posting the IOBLOK (IOBSTAT=IOBFATAL). The error is recorded by DMKIOS via DMKIOERR.

If the error is corrected by a restart, the temporary or transient error is not recorded. Control is returned to DMKIOS with the error flag off.

Before returning control to DMKIOS on either a permanent error of a successful recovery, the ERP frees all auxiliary storage gotten for recovery CCWs, buffers, and IOERBLOKS.

The DMKIOS interface with the ERP uses the IOBSTAT and IOBFLAG fields of the IOBLOK to determine action required when the ERP returns to DMKIOS.

When retry is to be attempted the ERP turns on the restart bit of the IOBFLAG field. The ERP bit of IOBSTAT field is also turned on to indicate to DMKIOS that the ERP wants control back when the task has finished. This enables the ERP to receive control even if the retry was successful and allows the freeing of all storage gotten for CCWs and temporary buffers. The IOBRCAW is set to the recovery CCW string address.

In handling an intervention required situation, the ERP sends a message to the operator and then waits for the device end to arrive. This is accomplished by a return to DMKIOS with the ERP bit in the IOBSTAT field set on and the IOBSTRT bit in the IOBFLAG field set off. When the device end interrupt arrives, the original channel program which was interrupted is then started.

The ERP flags of the IOERBLOK are also used to indicate when special recovery is being attempted. For example, a READ HOME ADDRESS command when a no record found error occurs.

The other two indications are self explanatory and are explained in Figure 3.

Field			Action to be
IOBSTAT	IOBFLAG	IOBSTAT	Performed
IOBERP	IOBRSTRT	IOBFATAL	by DMKIOS
1	0	0	Return control when solicited device end arrives
1	1	0	Restart using IOBRCAW
0	0	1	Permanent I/O Error
0	0	0	Retry successful

Figure 3. Summary of IOB Indicators

If the error is uncorrectable or intervention is required, the ERP calls DMKMSW for operator awareness. The specific message is identified in the MSGPARM field of the IOERBLOK.

Tape Error Recovery, ERP (DMKTAP)

Error recovery is attempted for user-initiated tape I/O operations to VM/370 control program supported devices that use the diagnose interface. The primary control blocks used for error recovery are the RDEVLOK, the IOBLOK, and the IOERBLOK. In addition, auxiliary storage is used for recovery channel programs (repositioning and erase).

The interrupt handler, DMKIOS, performs a SENSE operation when a unit check occurs. Tape errors are then passed to this DMKTAP. The sense information associated with a unit check is contained in the IOERBLOK. If a channel check is encountered, the channel check interrupt handler determines if retry is possible and passes control to the ERP through the I/O Interrupt Handler.

When an error is encountered and ERP receives control, DMKTAP determines if this the first entry into the ERP for this task. The IOBRCNT (IOB error count) field of the IOB is zero. On this first entry, the pointer to the IOERELOK is placed in the RDEVIOER field of the RDEVBLOK This preserves the original error CSW and sense information for recording. Thereafter, IOERBLOKS are discarded before a retry is attempted or a permanent error is passed to IOS.

The ERP looks for two other specific conditions. If the error count field is not zero, entry must be due to a recovery attempt. Thus, it may be a solicited device end to correct an intervention required condition or a retry attempt for either tape repositioning or channel program re-execution.

The ERP keeps track of the number of retries in the ICBRCNT field of the IOBLOK to determine if a retry limit has been exceeded for a particular error. If the specified number of retries fails to correct the error, the error is recorded and DMKIOS is notified of the permanent error by turning on a status flag in the IOBLOK (IOBSTAT=IOBFATAL).

If the error is corrected by DMKTAP, the temporary error is not recorded and control is returned to DMKIOS with error flags all off. When repositioning is required to attempt recovery, additional flags (EPPFLAGS) are contained in the IOERBLOK to indicate paths for specific errors (that is, data check on write must reposition, erase, and then reissue original channel program).

All error recovery is started the same except for intervention required errors. The IOBFLAG is turned on to indicate RESTART (IOBFLAG=IOBRSTRT), and the IOBRCAW (IOBLOK Restart CAW) is filled with the restart channel address word. In addition, an IOBSTAT flag is turned on to indicate that the ERP is in control so that control can be returned to ERP during all tape error recovery (IOBSTAT=IOBERP). In the case of an intervention required error, the ERP sends a message to the operator, and then returns to DMKIOS with indications that tell DMKIOS the ERP is waiting for a device end on this device. This is done by clearing the restart flag and returning to DMKIOS with only the IOBERP flag on.

When ERP has determined a permanent error situation or successfully recovered from an error, all auxiliary storage gotten for recovery CCWs, buffers, and IOERBLOKs is freed before a return is made to DMKIOS (see Figure 3 for a summary of the IOB indicators).

If the error is uncorrectable or operator intervention is necessary, the ERP calls the message writer to write the specific message.

Virtual I/O Interrupts

When an I/O interrupt is received (see Figure 4), the IOBLOK is stacked for dispatching and control is passed to the address specified in the IOBIRA (Interrupt Return Address) field. For operations requested by DMKVIOEX, the return address is DMKVIONT (Virtual Interrupt Return Address). When DMKVIONT receives control from the dispatcher, it loads the virtual address of the unit with which the interrupt is associated from the IOBLOK and calls DMKSCNVU to locate the virtual device control blocks. DMKVIONT then tests the IOBLOK status field to determine the cause for the interrupt. If the block has been unstacked due to an interrupt, the field is zero. If the operation was not started, it contains the condition code from the real SIO.

Note: The VIRA should nct see a real condition code 2 as the result of a SIO, since channel busy conditions are detected and reflected before any real I/O operation is attempted.

A condition code 3 is reflected to the user and exit is taken to the dispatcher. For condition code 1, the CSW status field in the IOBLOK is examined to determine the cause for the CSW stored condition. The status is reflected to the user and various components of the virtual configuration may be freed, if the status so indicates. For example, if the CSW status indicated both channel end and device end, the operation was immediate and has completed. Thus, the CCW string (real) may be released and all virtual components marked available.

The CSW status status returned for a virtual interrupt must be tested in the same manner, with the additional requirement that the status be saved in the affected virtual I/O control blocks and that the CSW be saved in the VDEVCSW field for the device causing the interrupt. If the unit check bit is on in the status field, the sense information saved in the associated IOERBLOK (pointed to by the IOBLOK) must be retained so that a sense initiated by the virtual machine receives the proper information.

In any case, when an interrupt is received for a virtual device, a bit in the interrupt mask, VCUDVINT, for the device's control unit is set to one. The bit that is set is the one corresponding to the relative address of the interrupting device on the control unit. For example, if device 235 interrupts, the fifth bit in the VCUDVINT mask in the VCUBLOK for control unit 30 on channel 2 is flagged. Similarly, the bit in the VCHCUINT in the affected VCHBLOK is also set; in this case, bit 3 in VCHBLOK for channel 2. If the interrupt is a channel class interrupt (PCI or CE), the address of the interrupting unit (235) is stored in the VCHCEDEV field in the VCHBLOK. The final interrupt flag is set in the VMPEND field in the VMBLOK for the interrupted user; the bit set corresponds to the address of the interrupting channel. The next time, the user is dispatched and becomes enabled for I/O.

Scheduling I/O Requests

A task that requests an I/O operation must specify the device on which the operation is to take place and must provide an IOBLOK that describes the operation. Upon entry to DMKIOS, Register 10 must point to the IOBLOK. The IOBLOK must contain at least a pointer to the channel program to be started in IOBCAW and the address to which the dispatcher is to pass control in IOBIRA. In addition, the flags and status fields should be set to zero. If the operation is a VM/370 control program function such as spooling or paging, the entry point DMKIOSQR is called. If the requestor is the virtual I/O executive (OMKVIDEX) attempting to start a user operation, the entry point DMKIOSQV is called and some additional housekeeping is done. In either case, an attempt is made to find an available subchannel path from the device to its control unit and channel. If a component in the path is BUSY or SCHEDULED, the IOBLOK for the request is queued to the control block of the component.

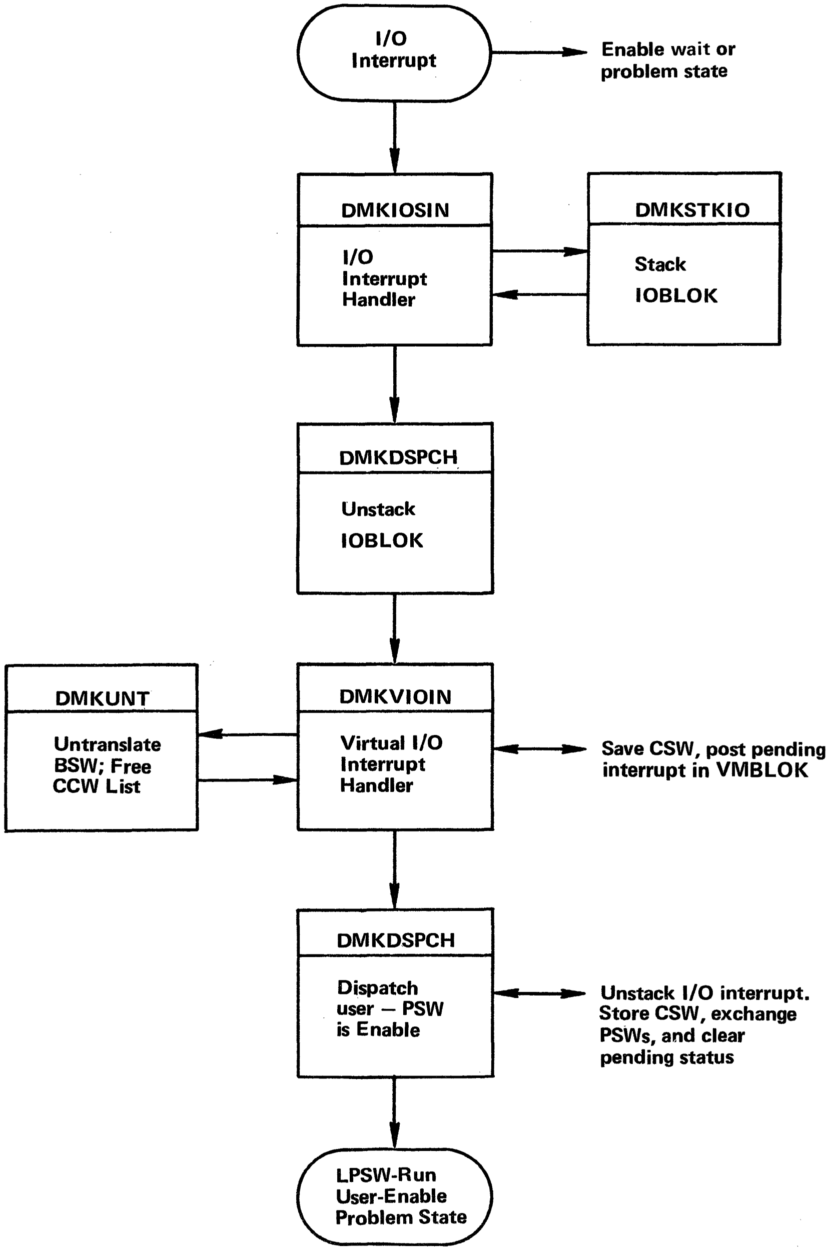
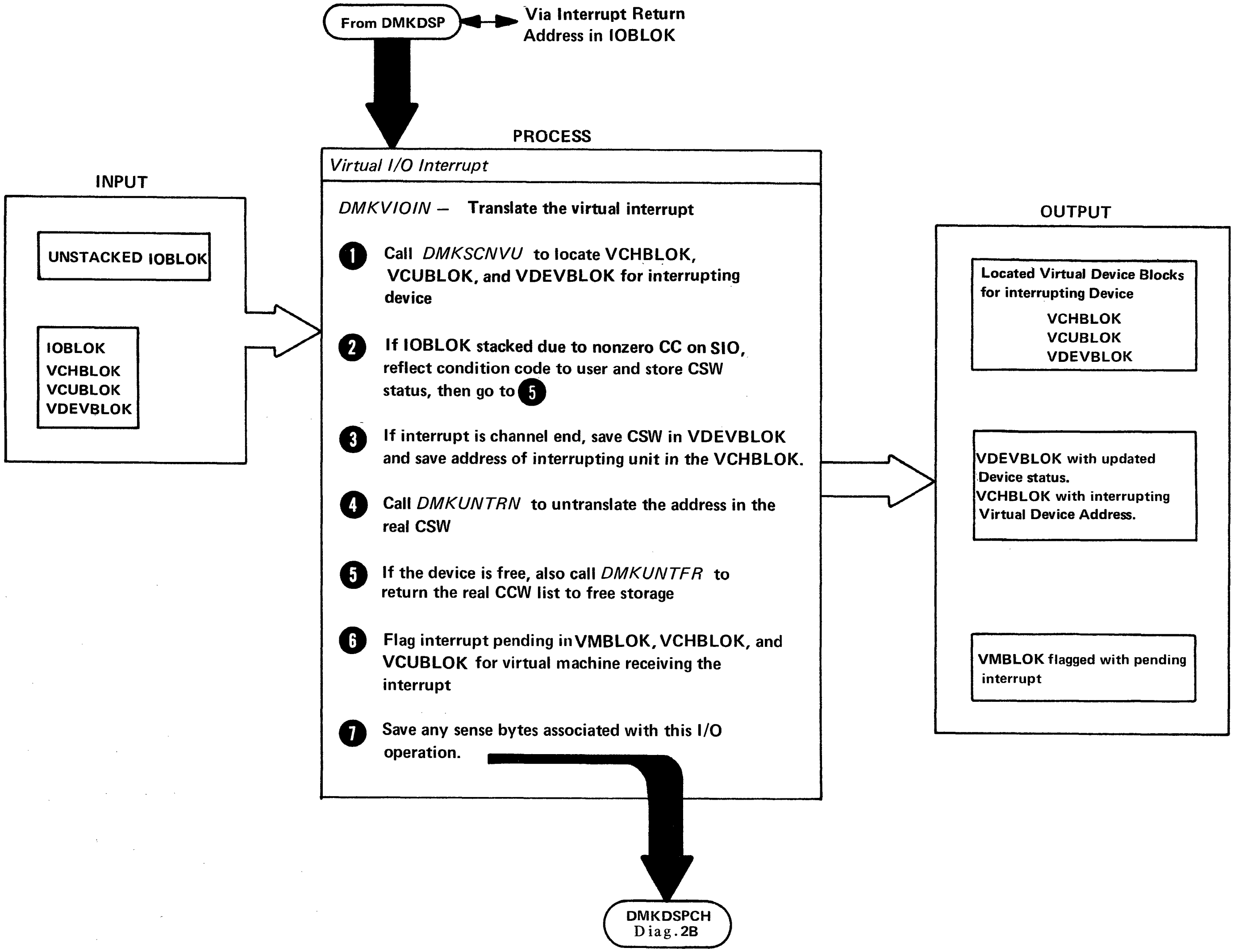

I/O Interrupt
Enable wait or problem state
DMKIOSIN
I/O Interrupt Handler
DMKSTKIO
Stack IOBLOK
DMKDSPCH
Unstack IOBLOK
DMKUNT
Untranslate BSW; Free CCW List
DMKVIOIN
Virtual I/O Interrupt Handler
Save CSW, post pending interrupt in VMBLOK
DMKDSPCH
Dispatch user — PSW is Enable
Unstack I/O interrupt. Store CSW, exchange PSWs, and clear pending status
LPSW-Run User-Enable Problem State

Figure 4. Overview of a Virtual I/O Interrupt

Diag. 1B4.2. Virtual I/O Interrupt



From DMKDSP
Via Interrupt Return Address in IOBLOK
INPUT
UNSTACKED IOBLOK
IOBLOK
VCHBLOK
VCUBLOK
VDEVBLOK
PROCESS
OUTPUT
Located Virtual Device Blocks for interrupting Device
VCHBLOK
VCUBLOK
VDEVBLOK
VDEVBLOK with updated Device status.
VCHBLOK with interrupting Virtual Device Address.
VMBLOK flagged with pending interrupt
DMKDSPCH
Diag.2B

Virtual I/O Interrupt

DMKVIOIN – Translate the virtual interrupt

1. Call *DMKSCNVU* to locate VCHBLOK, VCUBLOK, and VDEVBLOK for interrupting device
2. If IOBLOK stacked due to nonzero CC on SIO, reflect condition code to user and store CSW status, then go to 5
3. If interrupt is channel end, save CSW in VDEVBLOK and save address of interrupting unit in the VCHBLOK.
4. Call *DMKUNTRN* to untranslate the address in the real CSW
5. If the device is free, also call *DMKUNTFR* to return the real CCW list to free storage
6. Flag interrupt pending in VMBLOK, VCHBLOK, and VCUBLOK for virtual machine receiving the interrupt
7. Save any sense bytes associated with this I/O operation.

Requests are usually queued first-in first out (FIFO), except those:

- To moveable head DASD devices that are queued in order of seek address.

- That release the affected component after initiation (SEEKS and other control commands) which are queued last-in first out (LIFO) from the control block.

Regardless of whether or not the operation has been successfully started, the caller requesting the I/O operation receives control back from DMKIOS. If a free path to the device is found, the unit address is constructed and an SIO is issued. If the resulting condition code is zero, control is returned to the caller; otherwise, the code is stored in the requestor's IOBLOK along with any pertinent CSW status, the IOBLCK is stacked, any components that become available are restarted, and control is returned to the caller.

Ordered Seek Queueing: Requests to start I/O on system devices are normally handled FIFO. However, requests to moveable head DASD devices are queued on the device in ascending order by seek address. This ordered seek queuing is performed to minimize intercylinder seek times and to improve the overall throughput of the I/O system.

The VM/370 control program assumes that very few virtual machines will do chained seeks; hence, the first logical address represents where the arm will be positioned upon completion of the I/O operation. Ordered seek queueing is based on the relocated real cylinder. DMKIOS uses the cylinder location supplied in IOBCYL for ordered seek queuing. This field is initialized by the calling VM/370 control program routine for paging and spooling or by the CCW translator for virtual I/O. The CCW translator DMKCCW supplies the IOBCYL value in the following manner.

- Read IPL record, relocated to virtual cylinder 0

- Recalibrate, issue a real recalibrate and then seek to virtual cylinder 0

- Channel seeks, relocate to the virtual cylinder

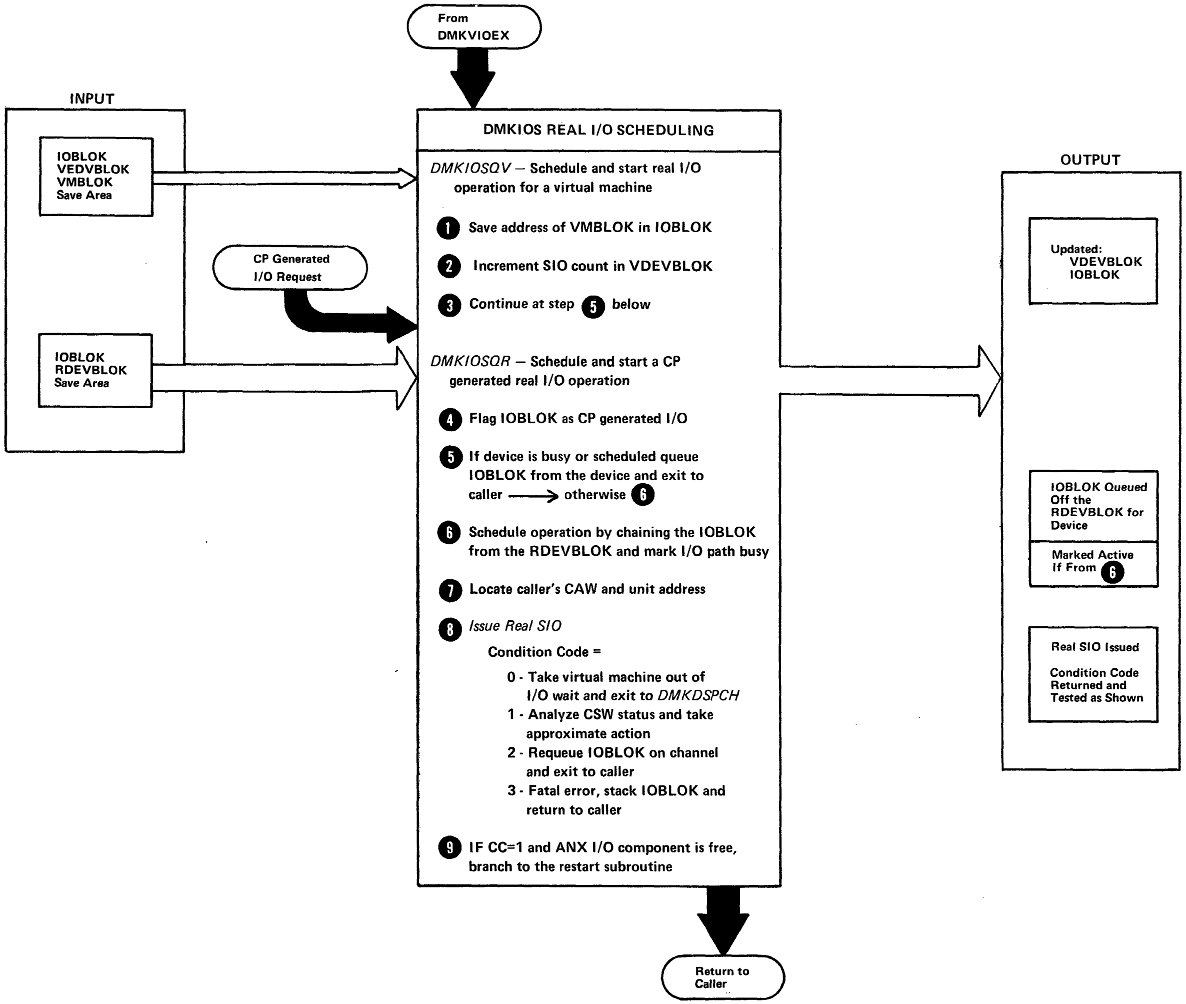
The IOBLOK queueing subroutine of DMKIOS recognizes that a request is being queued on a moveable head DASD device by means of the device class and type fields of RDEVBLOK. Instead of adding the IOBLOK to the end of the queue on the RDEVBLOK, the queueing routine sorts the block into the queue based on the cylinder number for the request. The cylinder number for any request to a DASD device is recorded in the field IOBCYL. The queue of IOBLOKs on a real device block is sorted in ascending order by seek address, unless the entire device is dedicated to a given user. In this case, DMKIOS does not automatically schedule the device, and no more than one request can be outstanding at any one time.

When an outstanding I/O request for a device has completed, DMKIOS attempts to restart the device by dequeuing and starting the next IOBLOK queued on the device. For non-DASD devices, this is the first IOBLOK queued. However, for moveable head DASD devices, the queued requests are dequeued in either ascending or descending order, depending on the current position (recorded in RDEVCYL) and the direction of motion of the arm. If the arm is seeking up (that is, toward the higher cylinder numbers), the queue of IOBLOKs is scanned from the first block toward the last until an IOBLOK is found with an IOBCYL value equal to or greater than the value in RDEVCYL, or until the end of the queue is reached. At this point, the device is flagged as seeking down and the queue is scanned from last to first until an IOBLOK with an IOBCYL value equal to or less than RDEVCYL is found. When IOBLOK is found, it is dequeued and started. The direction of motion is remembered in an RDEVFLAG bit and the next request is dequeued in the down direction until the head of the queue is reached.

Because the queue itself is a two-way chained list, no special handling for null or unity set lists is required, and the ordered seek algorithm returns to FIFO queueing.

Dedicated Channel Support: One of the facilities of the VM/370 control program allows a virtual machine to control one or more channels on a dedicated basis. The channels are attached to the virtual machine by using the privileged ATTACH CHANNEL command. A virtual machine can have one or more dedicated channels. In addition, channels can be split between virtual machines but a dedicated channel cannot be shared between two virtual machines. For instance, channel 1

Diag. 1B4.3. I/O Scheduling



DMKIOS REAL I/O SCHEDULING

DMKIOSQV – Schedule and start real I/O operation for a virtual machine

1. Save address of VMBLOK in IOBLOK
2. Increment SIO count in VDEVBLOK
3. Continue at step 5 below

DMKIOSQR – Schedule and start a CP generated real I/O operation

4. Flag IOBLOK as CP generated I/O
5. If device is busy or scheduled queue IOBLOK from the device and exit to caller ——> otherwise 6
6. Schedule operation by chaining the IOBLOK from the RDEVBLOK and mark I/O path busy
7. Locate caller's CAW and unit address
8. *Issue Real SIO*

   Condition Code =

   0 - Take virtual machine out of I/O wait and exit to *DMKDSPCH*
   1 - Analyze CSW status and take approximate action
   2 - Requeue IOBLOK on channel and exit to caller
   3 - Fatal error, stack IOBLOK and return to caller
9. IF CC=1 and ANX I/O component is free, branch to the restart subroutine

could be dedicated to virtual machine A, and channel 2 could be dedicated to virtual machine B, or they could be both dedicated to virtual machine A or B.

With a dedicated channel, all virtual machine device addresses must be identical to the real machine device addresses. For instance, virtual device 130 must be on real device 130, and virtual device 132 must be on real device 132. With dedicated channels, the VM/370 control program does not perform any virtual device address mapping. With a dedicated channel in effect, a virtual machine I/O operation to one of the dedicated devices on that channel results in the control program performing the operation directly on that device and reflecting the true condition code back to the virtual machine. None of the I/O operations are passed through control program's normal channel scheduling since the channel is completely dedicated to the virtual machine and any conditions in the channel are a direct result of that virtual machine's operation of that channel.

It is expected than any I/O new PSW for a virtual machine operating system has all channels masked off. Thus, when the VM/370 control program receives a hardware interrupt from a dedicated channel it immediately disables all further interrupts on that channel. The interrupt is then reflected to the virtual machine. The real channel stays disabled until the virtual machine issues an instruction to enable that channel. At that time, the VM/370 control program performs a hardware function to enable the real channel.

By using the dedicated channel feature, a virtual machine bypasses the VM/370 control program overhead associated with channel scheduling and virtual machine interrupt stacking. The channel scheduling is bypassed by performing the I/O operation directly and the interrupt stacking is bypassed by disabling the channel and having the hardware perform the true interrupt stacking.

The VM/370 control program error recording and channel recovery procedures are still in effect for dedicated channels. The dedicated channel support can be used in conjunction with the Virtual=Real feature for any virtual machine that is occupying the Virtual=Real storage space.

DISPATCHER/SCHEDULER

The module that selects dispatchable users from the population is DMKSCH, the Scheduler. The module that tests and alters the resources of the CPU is DMKDSP, the Dispatcher. The auxiliary routine that assists the Scheduler and Dispatcher is the request stack maintenance routine, DMKSTK.

In order to make decisions on both dispatching and scheduling, the control program classes all users into various categories, and recognizes user machines as being in one of several states. The user categories recognized are classed as being either interactive or non-interactive.

- An interactive user is one whose use of the system is punctuated by regular and frequent terminal I/O, and does not execute long CPU loops. A user becomes eligible to enter interactive status whenever a channel program for virtual console I/O has completed, or whenever I/O for a dedicated or dialed virtual telecommunications line has completed.
- A non-interactive user is one who has violated an interactive criterion, or one who has entered an idle wait state by entering console function mode (equivalent to stopped state), or by loading a wait state PSW that is not enabled for any busy channel. The control program schedules interactive users ahead of non-interactive users. Non-interactive users are subdivided into several classes. Normal non-interactive users are scheduled via a priority scheme described below. A user is allowed to execute for a specified time period and he is then placed in a list of those users who are waiting.

In order to give preference to certain classes of users, a priority scheduling scheme allows users to be scheduled with a priority class. The priority is a number assigned by the directory; however, the number may be altered by the system operator.

USER DISPATCHING LISTS AND MACHINE STATES

In order to efficiently manage the large inventory of potential users that are logged on to the system, the control program defines several states that a virtual machine may occupy. The scheduler can move a virtual machine from one state to another; however, a virtual machine may exist in only one state at any given instant. The control program can then make scheduling and dispatching decisions by looking only at the subset of users that are in the appropriate state. To facilitate this search, it also maintains lists of users in certain executable states.

A user's virtual machine may be in one of the following states:

State	Meaning
1	Interactive and dispatchable (in queue1, in DISPATCH list)
2	Interactive and not dispatchable (in queue1, not in DISPATCH list)
3	Interactive and eligible for queue1, but queue1 is full (waiting for queue1, in ELIGIBLE list)
4	In wait state with terminal read or write active
5	Non-interactive and dispatchable (in queue2, in DISPATCH list)
6	Non-interactive and not dispatchable (in queue2, not in DISPATCH list)
7	Non-interactive and eligible for queue2, but queue2 is full (waiting for queue2, in ELIGIBLE list)
8	Idle - waiting for asynchronous I/O or external interrupt, or stopped (in Console Function Mode)

Two lists of users are maintained by the scheduler:

- The DISPATCH list
- The ELIGIBLE list

Entries on the DISPATCH list are the VMBLOKS for those users in states 1 and 5, and represent the users that can be run at any given time. The DISPATCH list is sorted by dispatching priority, which is the ratio of CPU time to wait time over the life of the current user task. A task is defined as that execution which takes place between terminal reads or entry to enabled wait (that is, movement from state 4 or 8 to state 1) and is re-projected for a user each time he is dropped from a queue. Users entering state 1 always have a priority of 0.

The ELIGIBLE list is composed of those users in states 3 and 7; these users are potentially executable but due to the current load on the system they are not allowed to compete for the CPU. As soon as a user in the DISPATCH list is dropped from queue, the highest priority user(s) in the ELIGIBLE list is added to the DISPATCH list, subject to the restriction that his projected working set must not exceed the remaining system capacity. The ELIGIBLE list has two components; a section composed of those virtual machines waiting for Q1 (interactive) and a section composed of those virtual machines waiting for Q2 (non-interactive). Each section of the list is sorted by scheduling priority, which is determined at the time the virtual machine is added to the ELIGIBLE list, as follows:

1. The virtual machine's projected working set size, calculated the last time it was dropped from a queue, is expressed as a percentage of the amount of main storage available for paging. This percentage, usually between 0 and 100, is multiplied by the Paging Bias Factor (stored at DMKSCHPB).

2. The virtual machine's user priority (the priority set by the directory or the class A "SET PRIORITY" command) is multiplied by the User Bias Factor (stored at DMKSCHUB), and is added to the Paging Bias calculated in step 1.

3. The sum of Paging and User Bias is divided by the sum of the Bias Factors to obtain a weighted average.

4. A base priority is obtained by storing the TOD clock and using the high order word, which increments by 1 approximately once per second. This word is then modified by shifting it left or

right based on the Priority Delay Factor (stored at DMKSCHPD). If DMKSCHPD is positive, it indicates a right shift, thereby increasing the delay interval of the base priority; while a negative value indicates a left shift.

5. The weighted average obtained in step 3 is then logically added to the adjusted base obtained in step 4.

6. If the virtual machine is entering Q2 for the first time after being dropped from Q1, the Interactive Bias Factor (stored at DMKSCHIB) is subtracted from the priority obtained in step 5. If the virtual machine is entering Q1, or if it was last dropped from Q2, the Interactive Bias is not applied.

7. The result of steps 1 thorough 6 is the scheduling or eligible list priority, and is stored in the VMEPRIOR field of the VMBLOK.

The VMBLOK is then sorted into the appropriate section of the ELIGIBLE list in ascending value of VMEPRIOR. The effects of the various biases and the delay factor are illustrated by the following examples.

1. Assume that two virtual machines are to be added to the ELIGIBLE list for Q2. The Paging Bias Factor is 1, the User Bias Factor is 1, and the Priority Delay Factor is 0. Virtual machine "A" has a projected working set size of 80 percent of available storage and a user priority of 50. Virtual machine "B" has a projected working set size of 20 percent of available storage and also has a user priority of 50. The biases are obtained as follows:

User	Paging Bias		User Bias				Weighted Bias
A	80 X 1	+	50 X 1	=	130/2	=	65
B	20 X 1	+	50 X 1	=	70/2	=	35

If "A" is added to the eligible list at base time 0, its eligible list priority witll be 65. If the Priority Delay Factor is 0, "B" will be added <u>ahead</u> of "A" provided that "B" is eligible for entry to the list within the next (65-35) 30 seconds. If the Priority Delay Factor is set to +1, the base will be incremented once every two seconds. Therefore, although the bias difference is still 30, the delay time is now 60 seconds.

2. In order to force "A" to be given a weighted bias equal to "B," a priority differential is calculated as follows:

$$\frac{80 + A}{2} = \frac{20 + B}{2} \; ; \; A = B - 60$$

Therefore, for the biases to be equal, "A" must have a priority of 60 less than "B." For example, if "A" is given a priority of 10 and "B" is given a priority of 70, the biases would compute as follows:

User	Paging Bias		User Bias				Weighted Bias
A	80 X 1	+	10 X 1	=	90/2	=	45
B	20 X 1	+	70 X 1	=	90/2	=	45

3. The large difference in priorities could be lessened by <u>increasing</u> the User Bias Factor. If the User Bias Factor is set to 3 instead of 1, the calculated priority differential is as follows:

$$\frac{80 + 3A}{4} = \frac{20 + 3B}{4} \; ; \; 3(B - A) = 60 \; ; \; A = B - 20$$

Now, "A" requires a priority of only 20 less then "B" to achieve parity. For example:

User	Paging Bias		User Bias				Weighted Bias
A	80 X 1	+	30 X 3	=	170/4	=	42
B	20 X 1	+	50 X 3	=	170/4	=	42

The above examples illustrate the following general points about the use of the bias factors, the delay factor, and the user priority value:

1. The Paging and User Bias Factors are a measure of the relative importance of the bias value. A high

user bias will allow greater discrimination via the assigned priority; while a high paging bias makes storage requirement the primary scheduling parameter.

2. The <u>user priority value</u>, in the directory, is the means by which the paging priority may be overriden, and the means through which selected users will obtain improved performance.

3. The Priority Delay Factor is the measure of the impact which the paging and user biases are to have. The greater the delay value, the greater is the maximum delay that can be experienced by a given user.

4. The Interactive Bias Factor is a tool that enhances command response to conversational commands which require disk I/O, and which may be partially executed in Q2.

If the Paging Bias Factor is non-zero, the net effect of the priority scheme is to discriminate against users who require large amounts of real storage. This discrimination results in a higher level of multiprogramming and increased CPU utilization; however, it must be traded off against poorer throughput for large users. The distributed Scheduler is <u>not</u> biased; the bias factors are as follows:

Paging Bias Factor	(DMKSCHPB)	= 0
User Bias Factor	(DMKSCHUB)	= 1
Priority Delay Factor	(DMKSCHPD)	= 0
Interactive Bias Factor	(DMKSCHIB)	= 0

Thus, the basic VM/370 Scheduler will schedule virtual machines FIFO within user priority; the same algorithm provided with the basic Release 1.0 system.

Figure 5 is a graphic breakdown of the user states, showing the relationship between interactive and non-interactive states, in-queue and not-in-queue states, and in-list and not-in-list states.

	In-Queue		Not-in-Queue	
	DISPATCH List	No List	ELIGIBLE List	No List
Interactive	1	2	3	4
Non-Inter.	5	6	7	8

Figure 5. User Dispatching States

Figure 6 shows the possible user-state changes and the reasons for them; any changes not described are not possible.

CONTROLLING THE DEPTH OF MULTIPROGRAMMING

In order to control the number of users allowed in queue, the scheduler monitors the paging activity of all users and of the system as a whole. A decision as to whether or not to move a potential user from the eligible to the dispatch list is based upon whether or not that user's projected working set will exceed the system's remaining capacity. Individual user's working sets are calculated and projected at queue drop time according to one of the following formulas:

$P=(A+P)/2$

If $(LP-LA) * (P-A) < 0$

-- or --

$P=A$

If $(LP-LA) * (P-A) \geq 0$

<u>Note</u>: See the <u>Key</u> for the meaning of the symbols.

The working set is added to the current system load, which consists of the sum of the working sets for all users currently in a queue. The sum is compared to the system maximum, which is equal to the number of dynamically assignable pages in the system. If the user's projected working set will not push the system load over the maximum, he is placed in the queue and added to the dispatchable list.

Status Change		Reason for Status Change
From	To	
1	2	PAGEWAIT, SIO-WAIT, or enabled wait for any busy channel
1	4	Enabled wait for interactive terminal read or write
1	5	Exceeds in-queue time slice
1	7	Same as 1 to 5 except that queue2 is full
1	8	Wait without active I/O, disabled wait, or hit ATTN
2	1	Wait condition complete
2	5,7	Wait completes, but in-queue time slice exceeded
3	1	Another user drops from queue1 and now there is room
4	1	Terminal I/O completes while user is waiting
4	3	Terminal I/O completes, but queue1 is full
5	1	Terminal I/O completes while user is active in queue2
5	4	User puts up terminal read or write and enters wait
5	6	PAGEWAIT, SIO-WAIT, or enabled wait for busy channel
5	7	Dropped from queue2 due to in-queue time-slice end
5	8	Wait without active I/O, disabled wait, or hit ATTN
6	5	Wait condition completes
7	5	Room is found in queue2
8	5,7	Asynchronous I/O or External Interrupt, or BEGIN

Figure 6. User Status Changes

Key:
A = Actual working set at queue drop time

LA = Last actual working set

LP = Last projected working set

P = Current projected working set

The actual working set, A, is determined at queue drop time by the following formula:

$$A = \left[\begin{array}{c} \left(\sum_{i=1}^{N} PR_i \right) \Big/ N + \text{Steals} \\ \\ \text{— or —} \\ \\ \text{Pages referenced} \end{array} \right] \text{ whichever is greater}$$

where:

N = Number of page reads while in queue.

PR_i = Number of pages resident at the ith page read.

Steals = Number of times page wait was entered due to a stolen page.

The number of referenced pages is determined by scanning the user's page tables for software referenced bits. These bits are set by DMKPTRAN when the page is taken from the user by the control program. Thus the actual working set is generally the average number of pages resident at each page read. However, this estimate is sensitive to the overall system paging activity for the following reasons:

1. If there is no paging load on the system, there will be one page read for each resident page, and no steals; the working set will therefore tend to be equal to about one half of the resident page total.

2. As paging activity increases, and the working set locality shifts, the working set will tend to increase toward the average number of resident pages.

3. If paging activity becomes excessive, the number of page steals will increase to the extent that the working set will expand to the maximum of the total number of pages referenced while in the queue.

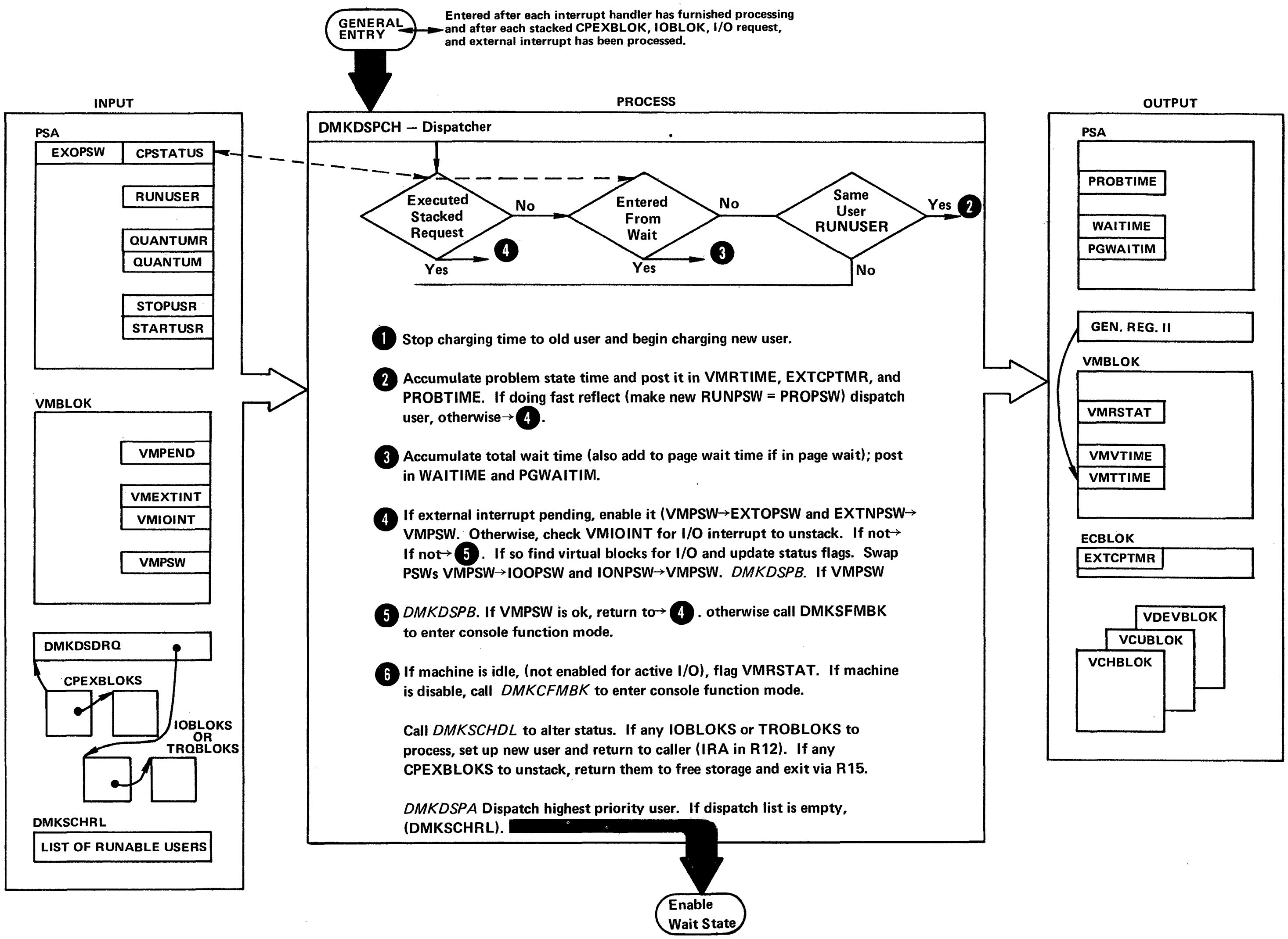
In summary, the scheduler selects the subset of logged-on users that are allowed to compete for the resources of the CPU, with the constraint that a new user is not added to the active subset if his projected main storage requirement, added to that of the other active users, causes the current capacity of the system to be exceeded. Selection within scheduling priority simply means that a executable user of high priority is always added to the active subset (to a queue) before a executable user of lower priority. If the paging bias mechanism is activated by setting the Paging Bias Factor to a non-zero value, scheduler selection will be in favor of smaller users; otherwise, selection is round robin within priority. Once the active subset (the set of in-queue users) has been selected, the dispatcher allocates resources of the CPU among them.

The list of executable users in a queue is sorted by dispatching (as opposed to scheduling) priority. The dispatching priority is a running average of a given user's CPU time/wait-time ratio. Thus, users who are most likely to go into wait state, based on past performance, are dispatched ahead of those whose demands on the CPU are more extensive. This simple ratio priority is normally altered if a user is identified as compute bound by means of the fact that he has executed for at least 50 ms. without entering the wait state. In this case, he is placed at the bottom of the dispatchable list. On the other hand, users identified as interactive by virtue of the frequency their requests for terminal I/O are placed at the top of the dispatchable list.

FAVORED EXECUTION OPTIONS

When the resources of the CPU (and real storage) are being allocated, the dispatching and scheduling functions are implemented in such a manner that options exist that allow an installation to designate certain users (virtual machines) that are to receive preferential treatment.

Diag. 2B. Dispatcher



1 Stop charging time to old user and begin charging new user.

2 Accumulate problem state time and post it in VMRTIME, EXTCPTMR, and PROBTIME. If doing fast reflect (make new RUNPSW = PROPSW) dispatch user, otherwise→4.

3 Accumulate total wait time (also add to page wait time if in page wait); post in WAITIME and PGWAITIM.

4 If external interrupt pending, enable it (VMPSW→EXTOPSW and EXTNPSW→ VMPSW. Otherwise, check VMIOINT for I/O interrupt to unstack. If not→ If not→5. If so find virtual blocks for I/O and update status flags. Swap PSWs VMPSW→IOOPSW and IONPSW→VMPSW. *DMKDSPB.* If VMPSW

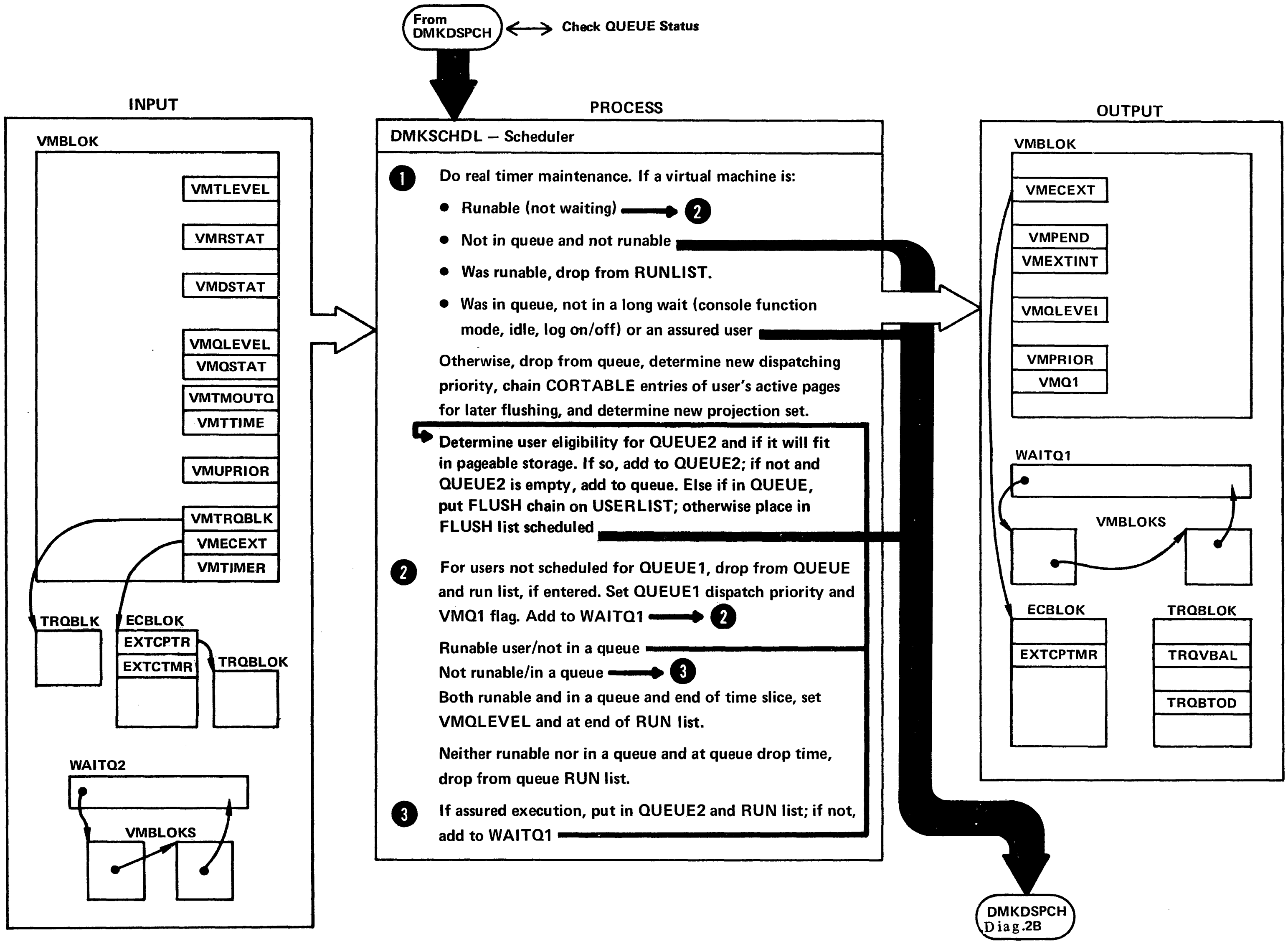
5 *DMKDSPB.* If VMPSW is ok, return to→4. otherwise call DMKSFMBK to enter console function mode.

6 If machine is idle, (not enabled for active I/O), flag VMRSTAT. If machine is disable, call *DMKCFMBK* to enter console function mode.

Call *DMKSCHDL* to alter status. If any IOBLOKS or TROBLOKS to process, set up new user and return to caller (IRA in R12). If any CPEXBLOKS to unstack, return them to free storage and exit via R15.

DMKDSPA Dispatch highest priority user. If dispatch list is empty, (DMKSCHRL).

Diag. 3B. Scheduler



From DMKDSPCH <-> Check QUEUE Status

DMKSCHDL – Scheduler

1 Do real timer maintenance. If a virtual machine is:

- Runable (not waiting) → 2
- Not in queue and not runable
- Was runable, drop from RUNLIST.
- Was in queue, not in a long wait (console function mode, idle, log on/off) or an assured user

Otherwise, drop from queue, determine new dispatching priority, chain CORTABLE entries of user's active pages for later flushing, and determine new projection set.

Determine user eligibility for QUEUE2 and if it will fit in pageable storage. If so, add to QUEUE2; if not and QUEUE2 is empty, add to queue. Else if in QUEUE, put FLUSH chain on USERLIST; otherwise place in FLUSH list scheduled

2 For users not scheduled for QUEUE1, drop from QUEUE and run list, if entered. Set QUEUE1 dispatch priority and VMQ1 flag. Add to WAITQ1 → 2

Runable user/not in a queue

Not runable/in a queue → 3

Both runable and in a queue and end of time slice, set VMQLEVEL and at end of RUN list.

Neither runable nor in a queue and at queue drop time, drop from queue RUN list.

3 If assured execution, put in QUEUE2 and RUN list; if not, add to WAITQ1

The favored execution options allow an installation to modify the algorithms described above and force the system to devote more of its resources to a given user than would ordinarily be the case. The options provided are:

1. The basic favored execution option.

2. The favored execution percentage.

The basic favored execution option means that the user so designated is never to be dropped from the active (in queue) subset by the scheduler. When the user is executable, he is to be placed in the dispatchable list at his normal priority position. However, any active user represents either an explicit or implicit commitment of main storage. An explicit storage commitment can be specified by either the Virtual=Real option or the reserved page option. An implicit commitment exists if neither of these options are specified, and the scheduler recomputes the user's projected work-set at what it would normally have been at queue-drop time. Multiple users can have the basic favored execution option set. However, if their combined main storage requirements exceed the sytem's capacity, performance can suffer due to thrashing.

The basic favored execution option removes the primary source of elapsed time stretch-out in a loaded time-sharing environment. However, if the favored task is highly compute bound and must compete for the CPU with many other tasks of the same type, an installation can define the CPU allocation to be made. In this case, the favored execution percentage option can be selected for one virtual machine. This option specifies that the selected user, in addition to remaining in queue, receives a given minimum percentage of the total CPU time, if he can use it. The percentage is assured in the following manner:

1. The in-queue time slice is multiplied by the requested percentage and added to the user's current total CPU time usage.

2. The favored user, when he is executable, is then always placed at the top of the dispatchable list until he has obtained his guarantee.

3. If the user obtains his guarantee before the interval has elapsed, he is placed in the dispatchable list according to his caluculated dispatching priority.

4. In any case, at the end of the in-queue time slice the percentage is recomputed and the process repeated.

These options can impact the response time of interactive users and only one favored percentage user is allowed at any given time.

DISPATCHING/SCHEDULING SUPPORT ROUTINES

Most of the routines in the CP nucleus are reentrant and multiple control program or user tasks can make use of one routine at the same time. However, there are certain areas where requests for a resource must be serialized (as in paging) or delayed while previous requests are serviced (as in requests to schedule I/O).

The CP Request Stack

The routine handling the request obtains a CPEXBLOK from free storage and stores the caller's registers in it; when the requested resource is free, the CPEXBLOK is stacked for the dispatcher via a call to the Request Stack Manager (DMKSTKCP). The dispatcher unstacks the block and exits to the requesting routine the next time it is entered. I/O requests are stacked in the same manner, except that the stacking vehicle is the IOBLOK, and return is passed to the address specified in the interrupt return address (IOBIRA). In either case, it should be noted that the dispatcher always unstacks and gives control to any stacked IOBLOKs and CPEXBLOKs prior to dispatching a user. This guarantees that control program information needed by a user (such as page availability) is always as up-to-date as possible.

CP INTERNAL TRACE TABLE

CP provides an internal trace table where events that occur in the real machine may be recorded. The size of the trace table depends on the amount of real storage available at IPL time. For each 256K bytes (or part thereof) of real storage available at IPL time, one page (4096 bytes) is allocated to the CP trace table. The storage thus allocated is contiguous and each entry is 16 bytes long. The first byte of each trace table entry, the identification code, identifies the event being recorded. Events that are traced are:

- External interrupts
- SVC interrupts
- Program interrupts
- I/O interrupts
- Free storage requests
- Release of free storage
- Entry into dispatch
- Queue drop
- Run user requests
- Start I/O
- Unstack I/O interrupts
- Storing a virtual CSW
- Test I/O
- Halt device

The main initialization routine, DMKCPI, allocates storage to the CP trace table and activates internal tracing. If you do not wish to record events in the trace table, the class A or E command MONITOR STOP can be issued to suppress recording. The pages allocated to the trace table are not released and recording can be restarted at any time by issuing the MONITOR START command. If the VM/370 system should abnormally terminate and automatically restart, the tracing of events on the real machine will be active. After a VM/370 IPL (manual or automatic), CP internal tracing is always active.

The first event traced is placed in the lowest trace table address. Each subsequent event is recorded in the next available trace table entry. Once the trace table is full, events are recorded at the lowest address (overlaying the data previously recorded there). Tracing continues with each new entry replacing an entry from a previous cycle.

The trace table can be used to determine the events that preceded a CP system failure. An ABEND dump contains the CP internal trace table along with the pointers to it. The address of the start of the trace table, TRACSTRT, is at location X'0C'. The address of the byte following the end of the trace table, TRACEND, is at location X'10'. The address of the next available trace table entry, TRACCURR, is at location X'14'. The trace table entry for the last event completed is obtained by subtracting 16 bytes (X'10') from the address stored in TRACCURR.

There are fourteen possible types of trace table entries, each uniquely identified by the value of the first byte. Figure 7 describes the format of each type and identifies the CP module that records the event.

Type of Event	Module	Identification Code (hexadecimal)	Format of Trace Table Entry
External interrupt	DMKPSA	01	0 X'01' 1 X'0000000000' 6 Interrupt Code 8 External Old PSW 15
SVC interrupt	DMKPSA	02	0 X'02' 1 GR 15 4 Instruction Length Code 6 Interrupt Code 8 SVC Old PSW 15
Program interrupt	DMKPRG	03	0 X'03' 1 X'000000' 4 Instruction Length Code 6 Interrupt Code 8 Program Old PSW 15
Not used		04	
I/O interrupt	DMKIOS	05	0 X'05' 1 X'00' 2 Device Address 4 I/O Old PSW + 4 8 CSW 15
Free Storage (FREE)	DMKFRE	06	0 X'06 1 GR 11 at entry 4 GR 0 at entry 8 GR 1 at exit 12 GR 14 15
Return storage (FRET)	DMKFRE	07	0 X'07' 1 GR 11 at entry 4 GR 0 at entry 8 GR 1 at entry 12 GR 14 15
Enter scheduler	DMKSCH	08	0 X'08' 1 Contents of VMRSTAT, VMDSTAT, and VMOSTAT 4 Address of VMBLOK 8 Value of CPU Timer 15
Queue drop	DMKSCH	09	0 X'09' 1 Address of VMBLOK 4 Old Priority 6 New Priority 8 Number of Resident Pages 10 Projected Working Set 12 Number of Referenced Pages 14 Current Page load (PSA) 15
Run user	DMKDSP	0A	0 X'0A' 1 X'000000' 4 RUNUSER value from PSA 8 RUNPSW value from PSA 15
Start I/O	DMKIOS	0B	0 X'0B' 1 Condition Code 2 Device Address 4 Address of IOBLOK 8 CAW 12 For CC = 1, CSW + 4 otherwise this field is not used 15
Unstack I/O interrupt	DMKDSP	0C	0 X'0C' 1 X'00' 2 Virtual Device Address 4 Address of VMBLOK 8 Virtual CSW 15
Virtual CSW store	DMKVIO	0D	0 X'0D' 1 Instruction Operation Code 2 Virtual Device Address 4 Address of VMBLOK 8 Virtual CSW 15
Test I/O	DMKIOS	0E	0 X'0E' 1 Condition Code 2 Device Address 4 Address of IOBLOK 8 CAW 12 For CC = 1, CSW + 4 otherwise this field is not used 15
Halt Device	DMKIOS	0F	0 X'0F' 1 Condition Code 2 Device Address 4 Address of IOBLOK 8 CAW 12 For CC = 1, CSW + 4 otherwise this field is not used 15

Figure 7. CP Trace Table Entries

SPOOLING

The spooling support in the VM/370 control program performs three functions.

First, to simulate the operation of the virtual unit record devices that are attached to each user's virtual machine configuration. The simulation is done in such a way that it appears to the program in the virtual machine that it is controlling a real unit record device. This support involves the interception and interpretation of user SIOs, the movement of data to and from the user's virtual storage space, and the reflection of the necessary interrupt codes and ending conditions in PSW's, CSW's and sense bytes. This support is provided by the Virtual Spooling Executive.

Second, to operate the real unit record equipment attached the system that is used to transcribe user output spool files from input from the real card readers. This function is provided by the Real Spooling Executive.

Third, to provide an interface among the users, the system operator, and the spooling system so that the location, format, priority and utilization of the systems spooling data and resources can be controlled.

SPOOL DATA FORMAT

Spool Buffer Format

The buffers used for collecting and writing spool data are all one page (4096 bytes) in length, and contain both the data to be transcribed and all CCWs necessary for operating the unit record devices that perform the transcription. The data is provided in the exact format required with no compression except that trailing blanks are suppressed. The first two doublewords of each buffer contain linkage information described below, followed by the data and CCWs.

Each spool logical record (card or print line) is stored as one data moving CCW (READ or WRITE), a TIC to the following CCW, and the full data record. Space is left at the end of each buffer so that a SENSE command can be inserted in order to force concurrent channel end and device end. For card punch channel programs there is an additional back chain field that points to the card previously punched so that error recovery for punch equipment checks can back up one card. The only exception to the format of Read/Write-TIC-Data is in buffers of files directed to the printer. In this case, immediate operation code CCWs (skips and spaces) are followed immediately by the next CCW.

Spool File Format

In addition to the data and CCWs contained in each SPOOL buffer, the first two doublewords contain forward and backward links to the next and previous buffers in the file. This two-way linkage allows the file to be backspaced/restarted from any point at any time. Also, it means that if I/O errors are encountered while reading one buffer, the file is put in system hold status. If purged, all buffers except those in error are released. The two-way chain allows this control of the file while preventing fragmentation by allowing pages to be assigned and released individually regardless of their ownership.

Each SPOOL file in the system is controlled by a Spool File Control Block (SFBLOK) that is resident in storage. While the file is open, these blocks are chained from the devices (either real or virtual) that are processing the file, and from device type file anchors after the file is closed. There is one file chain each for printer, reader, and punch files. Each SFBLOK contains information about the file that describes its owner and originator (these can be different for transfered files), the file name and type, and the class and number of copies for output files. All of these attributes can be examined and most can be changed by the file's owner or the system operator. The SFBLOK also contains information such as the starting and ending buffer addresses for the file, the record size, certain file status flags, etc.

SPOOL BUFFER MANAGEMENT

Real/Virtual Storage Management

Buffers used for the temporary storage of spool data on its way between DASD secondary storage and the user's virtual machine are allocated from a pool of virtual storage space that belongs to CP. This pool consists of the second 256K of virtual storage associated with the VMBLOK that controls CP's paging activities. This pool can be enlarged as a system generation option. Virtual storage buffers are allocated in one page increments by DMKPGT at the time the spool file is opened for either input or output. If no virtual storage space is available, the user is placed in a wait state until a buffer is freed by another user closing a file. This places limit on the number of concurrent spooling operations permitted by the system because spooling operates as a high priority task.

Real main storage is not allocated for a spooling buffer until a virtual machine actually issues a SIO that attempts to transfer data between the buffer and the user's virtual storage space. At this time, a page of real main storage is allocated to the buffer via the main storage paging manager. The buffer is locked in main storage (that is, is unavailable to be paged out) only for the amount of time necessary to transfer the data. After the data transfer is complete, the buffer is treated as a normal page of virtual storage, and can be selected to be paged out. This ensures that low usage spool files do not have buffers in real main storage, while the buffers for high usage files should remain resident. The Virtual Spooling Executive is insensitve to the location of the spool buffer in real storage, since all references to the data therein are accomplished through the dynamic address translation feature of the CPU.

DASD Space Allocation

While a spool buffer is active, it resides in in real main storage or on the paging device. After it has been filled with data from the virtual machine or a real input reader, it is written to a page of secondary DASD storage. The allocation of pages on the spooling disk(s) is managed by DMKPGT which is used to handle requests for both pages of virtual storage and semipermanent spool file residence. DMKPGT maintains separate allocation block chains for virtual storage and spooling pages. Each block contains control information and a bit map used to allocate pages on a single cylinder. If none of the cylinders allocated have any available pages, DMKPGT enters its cylinder allocation routine.

DMKPGT attempts to even out the spooling/paging I/O load by allocating cylinders in round robin fashion across channels and devices. In order to minimize seek times on a given device, an attempt is made to allocate cylinders as close to the relative center of the spooling/paging area as is possible.

Paging Device Support: All actual input/output for the page buffers on any device is controlled by the Paging I/O Executive DMKPAGIO and is discussed in that section in this publication.

VIRTUAL SPOOLING MANAGER (DMKVSP)

The two functions of the virtual spooling manager are to simulate the operation of all spooled unit-record devices attached to the user's virtual machine, and to read and write the spool files associated with those devices. The following virtual devices are supported for spooling, with the exceptions noted:

- The IBM 2540 Card Reader/Punch, except for punch feed read and column binary
- The IBM 1403 Printer Models 2 and N1 (132 positions)
- The IBM 3211 Printer (150 print positions)
- The IBM 3505 Card Reader (except for mark senses reading)
- The IBM 3525 Punch (except for the card read, print, and data protect features).

The following consoles and terminals are also supported for spooling when entered into the directory as the virtual system console:

- IBM 1052 Printer-Keyboard, Model 7 (via the 2150 Console)
- IBM 3210 Console Printer-Keyboard, Models 1 and 2
- IBM 3215 Console Printer-Keyboard, Model 1

It is assumed that all virtual printers have the universal character set feature. No checking is done on the spooled printer data. However, any UCS buffer commands issued by the virtual machine (load UCS buffer, block data checks, etc.) are ignored. It is up to the user and the installation to ensure that the output is directed to the proper real printer via use of the output CLASS feature described below. For the 3211 printer, Forms Control Buffer (FCB) commands are accepted and simulated by means of a virtual FCB maintained by the executive The use of the virtual FCB is the only way to simulate end-of-form conditions reflected by the detection of a channel 9 or 12 punch. When the spooled file is directed to a real 3211 or 1403, the operator is responsible for loading the FCB or mounting the proper carriage tape.

If any of the unsupported unit-record features are required, they may be used by attaching the real device directly to the user's virtual machine. Thus, a 3505 reader could be used for the most part as a spooling input reader, but attached directly to a batch virtual machine when it is necessary to read mark sense cards.

Output File Processing

DMKVSP receives control from the Virtual I/O Executive DMKVIO when the user issues a SIO to a spooled unit record device. DMKVIO does not pass control until it has been determined that the device is available (that is, non-busy and with no interrupts pending). DMKVSP first determines if the device is currently processing a file. If it is, processing continues. If this is the first command issued by the given device, a new output file must be opened. An open subroutine is called to build the control blocks necessary to manage the file and to obtain virtual storage and DASD buffer space. Control is then returned to DMKVSP.

DMSVSP then analyzes and interprets the channel program associated with the user's SIO. Each CCW is tested for validity of command, address, flags, alignment, protection, etc., and if the CCW is valid, the user's data is moved from his own virtual storage space to the buffer in the spooling virtual storage. When this buffer is full, it is written to a page of DASD secondary storage and a new buffer is obtained. The interpretation of the usei's channel program continues until there are no more CCWs or until an error condition is detected which prohibits further processing. In either case, the device is marked as having the proper interrupts pending, a CSW is contructed, and DMKVSP exits to the main dispatcher. In contrast to nonspooled I/O, the user has remained in a pseudo-wait (IOWAIT) for the time it took to interpret the entire channel program.

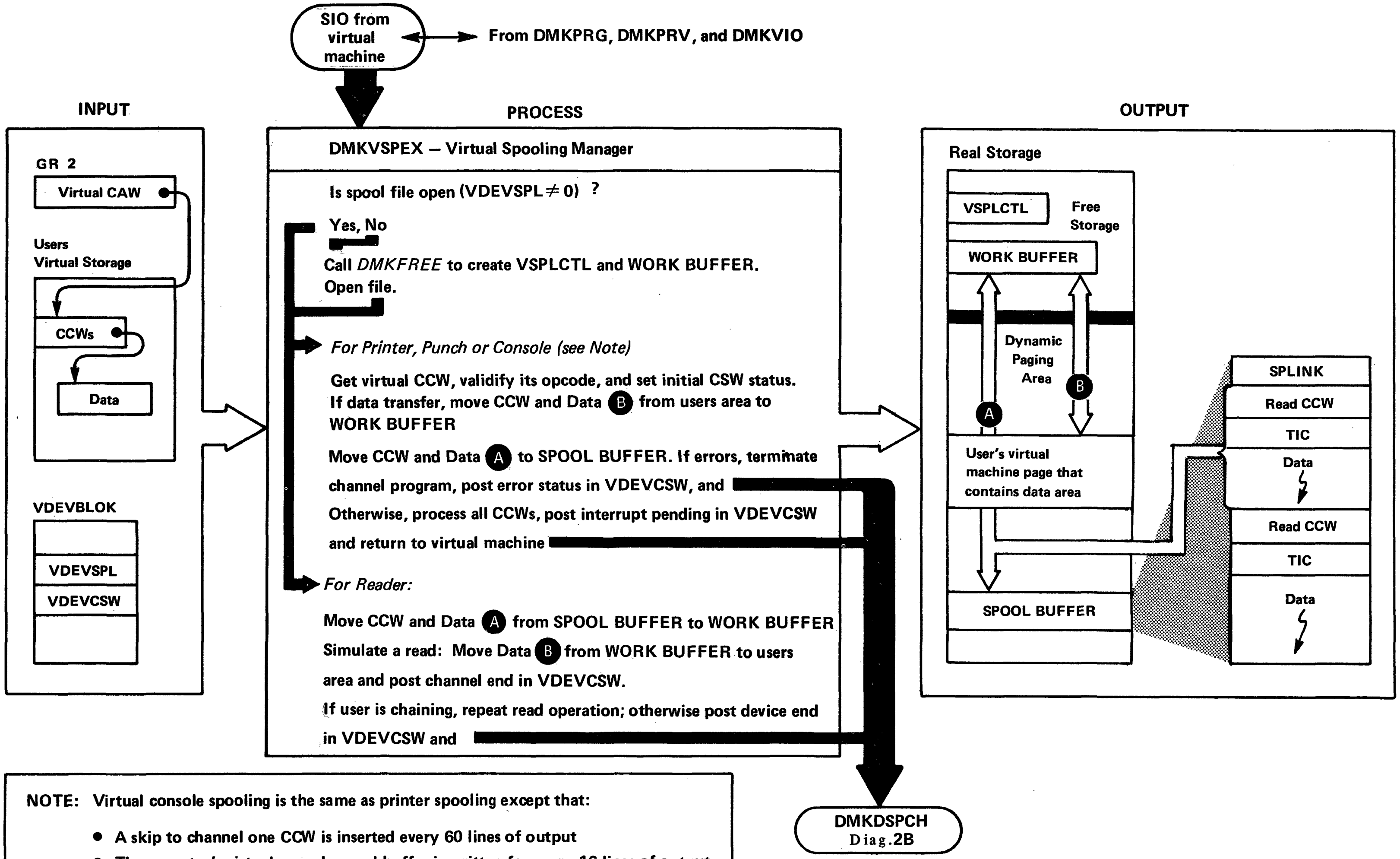
The output file can be logically closed by the user either by issuing an invalid CCW command code, or via the CP console function CLOSE. In either case, the device is cleared of pending interrupts, the file chains are completed, and the file is either queued for output on a real device of the proper type (printer or punch), or, if XFER is in effect, is queued for input to another user.

Input File Processing

Input file processing is similar to output file processing, except for the open and close functions, and the analysis of CCW commands and the direction of data movement. Many common routines are utilized to locate and verify CCWs, obtain buffer space, and to move the spooling data.

The difference in the open function is that instead of creating a new file, it is necessary to locate a reader file that already exists in the system. To do this, the open subroutine scans the SFBLOKs chained from the anchor READERS in order to find a file with an owner userid that matches that of the caller. If a file is not found, a unit check/intervention required condition is reflected to the user; otherwise, its SFBLOK is chained to the control block for the reader and the channel program is interpreted in the same manner as for an output file.

Diag. 4B1. Virtual Spooling Manager



DMKVSPEX – Virtual Spooling Manager

Is spool file open (VDEVSPL ≠ 0) ?

Yes, No

Call *DMKFREE* to create VSPLCTL and WORK BUFFER. Open file.

For Printer, Punch or Console (see Note)

Get virtual CCW, validify its opcode, and set initial CSW status. If data transfer, move CCW and Data B from users area to WORK BUFFER

Move CCW and Data A to SPOOL BUFFER. If errors, terminate channel program, post error status in VDEVCSW, and

Otherwise, process all CCWs, post interrupt pending in VDEVCSW and return to virtual machine

For Reader:

Move CCW and Data A from SPOOL BUFFER to WORK BUFFER

Simulate a read: Move Data B from WORK BUFFER to users area and post channel end in VDEVCSW.

If user is chaining, repeat read operation; otherwise post device end in VDEVCSW and

NOTE: Virtual console spooling is the same as printer spooling except that:

- A skip to channel one CCW is inserted every 60 lines of output
- The operator's virtual console spool buffer is written for every 16 lines of output
- The virtual spool buffer is written to the allocated spool device when the first CCW is placed in the virtual buffer. The buffer is kept in a pseudo closed state so that checkpoint saves the buffer in the event of a system failure.

After the input file is exahusted, a unit exception is reflected to the user machine, unless the user has requested either continuous spooling or that an EOF not be reflected. With continuous spooling, the unit exception is not reflected until the last file for that user is processed. If NOEOF is specified, the simulation terminates with a unit check/intervention required condition (similar to what happens if the EOF button on a real reader is not pushed).

In either case, the input file is then deleted from the system, unless the user has specifically requested that his input files be saved. If the file is saved, it can be re-read any number of times.

Virtual Console Spooling

Support of the virtual console input and output is provided as an option of the VM/370 spooling capabilities. This support fulfills the following requirements:

- Provides hardcopy support for CMS Batch virtual machines.
- Allows DISCONNECTED virtual machines to spool virtual console output to disk instead of losing the output.
- Improves the performance of virtual machines that currently produce a large amount of console output.

Whenever a SIO IS ISSUED TO A VIRTUAL MACHINE CONSOLE| THE Virtual Console Manager (DMKVCN) determines if the spooling option is active. If it is, control is passed to the Virtual Spooling Manager at DMKVSPBP to insert the data into a spool file buffer. While console spooling utilizes, basically, the same code as printer spooling, the following exceptions are made:

- A skip to channel one CCW is inserted every 60 lines of output.
- The operator's virtual console spool buffer is written out every 16 lines of output.
- The virtual space buffer is written out to the allocated spool device when the first CCW is placed in that virtual buffer. The linkage area of the virtual spool buffer takes the form of a CLOSE file to allow checkpoint (DMKCKP) to recover the active spool file in the event of a shutdown due to system failure. The data in the virtual buffer, not yet written out to the spool device will not be recovered.

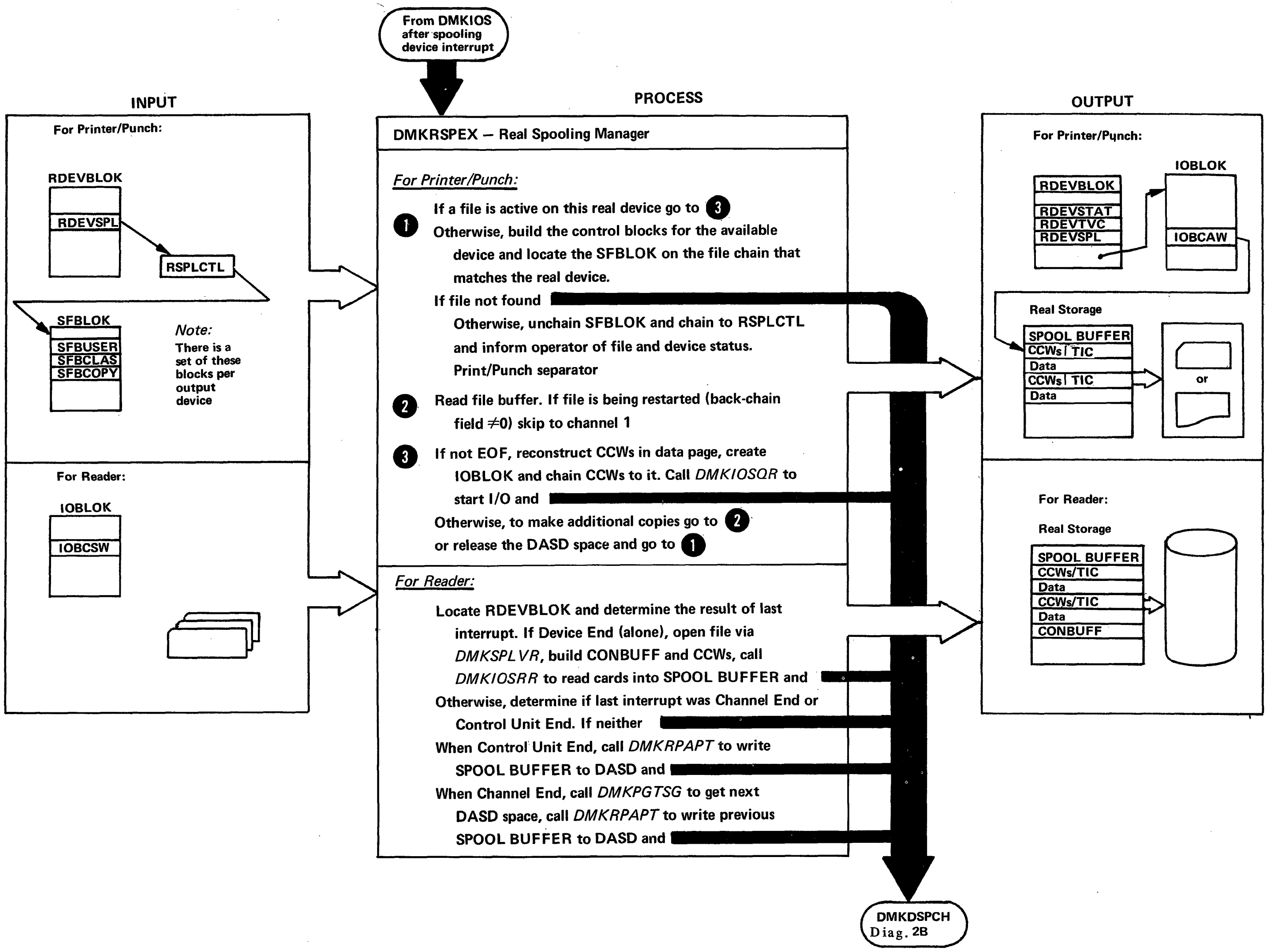
To maintain a pseudo closed file status for console spool files, DMKSPL now assigns spool identifications to all output spool files where they are first queued.

A virtual system reset, device reset, or IPL <u>will not</u> close the virtual console spool file. The LOGOFF, FORCE, or DETACH of virtual console commands <u>will</u> close the virtual console spool file. The SHUTDOWN command will close the operator's console spool file. If the SHUTDOWN command is issued by a Class A user other than the operator, the console spool file for both the user and operator will be closed.

REAL SPOOLING MANAGER (DMKRSP)

The real spooling manager operates the real unit record devices that are attached to the system and that are used to transcribe input data into reader spool files and user output spool files onto the real printers and punches. The executive optimizes the use of main storage and the CPU rather than running the system unit record devices at their rated speeds. DASD input files are not double buffered and under periods of peak load, input and output devices tend to run in bursts. However, command chaining is used for all unit record channel programs so that the devices are running at their maximum speed with a minimum of interruptions.

Diag. 4B2. Real Spooling Manager



DMKRSPEX — Real Spooling Manager

For Printer/Punch:

1. If a file is active on this real device go to 3
   Otherwise, build the control blocks for the available device and locate the SFBLOK on the file chain that matches the real device.
   If file not found
   Otherwise, unchain SFBLOK and chain to RSPLCTL and inform operator of file and device status.
   Print/Punch separator
2. Read file buffer. If file is being restarted (back-chain field ≠0) skip to channel 1
3. If not EOF, reconstruct CCWs in data page, create IOBLOK and chain CCWs to it. Call *DMKIOSQR* to start I/O and
   Otherwise, to make additional copies go to 2
   or release the DASD space and go to 1

For Reader:

Locate RDEVBLOK and determine the result of last interrupt. If Device End (alone), open file via *DMKSPLVR*, build CONBUFF and CCWs, call *DMKIOSRR* to read cards into SPOOL BUFFER and

Otherwise, determine if last interrupt was Channel End or Control Unit End. If neither

When Control Unit End, call *DMKRPAPT* to write SPOOL BUFFER to DASD and

When Channel End, call *DMKPGTSG* to get next DASD space, call *DMKRPAPT* to write previous SPOOL BUFFER to DASD and

Output File Processing

Both the input and output functions of DMKRSP are interrupt driven. Thus, DMKRSP does not process unless an internally or externally generated not-ready to ready device end interrupt occurs. External interrupts are generated by the hardware in the normal manner, while internal, "psuedo interrupts," are generated by the software when an output file has been queued on the real printers or punches file chain, or when the operator issues a START command to a drained device.

Upon receipt of the initial device end for a printer or punch, DMKRSP searches the appropriate file chain for the SFBLOK of a file whose class matches that of the device that was made ready. When the SFBLOK is located (provided the file is not in a HOLD status), it is unchained from the output queue and chained to the real device block that services the file. A page of real main storage is then obtained for use as a buffer, and the output separator routine (DMKSEP) is called to print output identifier pages. When DMKSEP returns control to DMKRSP, the first buffer of the file is paged into real main storage, and the CCWs in the channel program that it contains are adjusted so that their data addresses correspond to the real addresses at which the data resides. The real SIO supervisor (DMKIOSQR) is then called to start the channel program, and DMKRSP exits to the dispatcher (DMKDSPCH) to await the interrupt.

When the channel end/device end interrupt for the completed buffer is unstacked to DMKRSP, the forward chain file link field is used to locate the next buffer. This buffer is paged-in, and the process is repeated until the final buffer is processed. At this point, the number of copies requested for the file is decremented. If the number of copies is 0, processing is terminated and the file is deleted from the system; otherwise, the process is repeated as many times as is necessary.

When file processing is complete, a scan of the appropriate output queue is again made, and if a file is found it is processed. If the queue is empty, or if a file with a matching class is not found, an exit is taken to DMKDSPCH to wait for another ready interrupt.

Output file processing can be modified by either the system operator via the spooling support command or as a result of system errors. The operator commands allow a given file to be backspaced or restarted, and the files of individual users or the whole system to be held and released for output in a very flexible manner. I/O errors also affect the spooling system, and a description of how they are processed is in the section "Error Recovery."

Input File Processing

Reader file processing is initiated by the receipt of a device end interrupt from a spooling card reader. No explicit operator command is required to start the processing of an input file. When the device end is unstacked to DMKRSP, an open subroutine is called to build the necessary control blocks and to obtain the virtual, real, and DASD buffer space required for the file. A channel program to read 41 cards is built in the buffer, and DMKIOSQR is called to start the reader.

When the interrupt for the first buffer is unstacked, the first card is checked for its validity as a userid card. The minimum information that this card must contain is the userid of the owner of the input file. It may appear anywhere on the card, with the restriction that it must be the first information punched. Optional information on the userid card can include a file name and type and/or the class of the virtual card reader to which the file is to be directed. If the userid is valid, the file processing continues; otherwise, the operator receives an error message and processing is terminated.

After each file buffer is read, it is written onto disk by the paging I/O routines in the same manner that virtual output files are handled. When a unit exception signaling physical end of file is received from the reader, the file is closed by writing the final buffer to disk and completing and queueing the SFBLOK to the readers file chain. If the owner of the file is currently logged in, he is given a message indicating that a file has been read and the appropriate card reader is posted with a device end interrupt.

Accounting Card Processing

Various routines in CP accumulate, format, and punch account cards that contain system usage information for certain users. These routines format the information into an 80-column card image preceded by a punch CCW and call DMKACOAQ to queue the card for real output. DMKACOAQ calls DMKACOPU to punch the card on a real punch if one is available; otherwise, the card is queued in main storage until a punch is free. When a punch finishes processing its last file, a test is made to see if any accounting cards have been queued. If they have, DMKACOPU is called to process them.

In addition to the cards generated by CP to account for a virtual machine's use of system resources, the user may request cards to be punched in order to account for the use of virtual machine resources by jobs running under his userid. In order to do so, the user must have specified the account option (ACCT) when initially entered into the directory.

In order to punch an accounting card, the user must issue a X'004C' diagnose instruction with a pointer to a parameter list containing the "charge to" information. If the pointer is zero, the accounting card will be punched and will contain the user's own identification taken from his VMBLOK.

When the user accounting option is being utilized, the user must keep in mind that each additional accounting record requested is occupying real storage space. Degradation of system performance will occur if available storage becomes filled with accounting data.

SPOOLING COMMAND SYSTEM

The spooling command system provides an interface between the user, the system operator, and the spooling system itself. There are three types of spooling commands.

- Those that affect virtual devices
- Those that affect real devices
- Those that affect SPOOL files that are queued within the system

The commands that affect virtual devices are generally available to all system users, and a user can only affect the status of devices that are attached to his own virtual machine. Commands that affect the status of the real system's spooling devices are restricted to use by the system operator. Commands that affect closed spool files that are awaiting processing are generally available to all users, with some additional capabilities assigned to the system operator. For example, a user may alter the characteristics only of those files that have an owner's userid that matches his own, whereas the system operator may change any SPOOL file in the system.

File States and Attributes

Each spool file in the system has a number of attributes that are assigned to it, either explicitly or by default, at the time that it is created. These attributes and their values are as follows:

- Filename and type can be 24 character fields. Either or both can be replaced by a user-supplied value.
- Spoolid number is a system-assigned number between 1 and 9999. It is automatically assigned when the file is created (input) or closed (output), and is unique within the system. The file's owner, the device type, and the id number are specified. Usually, the userid defaults to the identification of the user issuing the given command. Since the identification number rather than the file name and type is used as an identifier, duplicate user-assigned names do not present an identification problem.
- The number of logical records (cards or print lines) in the file is an integer between 1 and 16 million. For printer files, the record count also includes any immediate operation code space or skip CCWs.

- The Originating User is the identification of the files creator, if the file has been internally transfered from the originator's printer or punch to the new owner's card reader.

- The number of copies requested for an output file is a number between 1 and 99. Unless altered by the user or operator, it defaults to 1.

- The device type is used by DIAGNOSE for a file transferred to a reader to determine the virtual type of output device.

In addition to those attributes, a file that is queued for real output or virtual input always has a class associated with it. A class is a single alphameric character from A through Z or from 0 to 9. It is used to control both the real or virtual device on which the file will be printed, punched, or read, and the relative priority and sequence of output on the device. While each file is assigned a single class, each real spooling output device be assigned from one to four classes. The device then processes only files that have a class attribute that corresponds to one of its own, and will process these files in the order that its own classes are specified.

For example, if a printer is assigned the classes A, D, 2, it processes any printer file with a class of A before it searches the printer output queue for a file with class D. All class D files are printed before class 2 files.

The output class for a file is assigned at the time the file is created and is the class that is associated with the virtual device that created it. While each real spooling device can have up to four classes, each virtual spooling device can have only one. When a user logs onto to the system, the class associated with a device is the one defined in his directory entry for that device. However, he can alter this class at any time via the spool command. As files are created and closed by a device, they take on the device's output class. After they are closed and are awaiting output, their class can be changed via a CHANGE command issued either by the file's owner or the system operator. The system operator can alter the system generated output class(es) of a real output device via the START command.

Output files transferred to a user's virtual reader can also be controlled by class. If the receiving user has several readers, the input to each can be limited to files of a certain class. In addition, the ORDER command allows sequencing of input files by class as well as spoolid number.

Output priorities can also be managed by altering the hold status of a file. Individual users can alter the hold status with the CHANGE command, while the system operator can change (hold or free) the files of specific individual users.

Virtual Device Spooling Commands

These commands affect the status of a user's virtual spooling devices:

Command	Meaning
CLOSE	Terminates spooling operations on a specified device. It clears the device of any pending interrupt conditions, and for output files completes and queues the file for real output. Optional parameters allow the user to specify a filename and type, and to override for the given file any standard class, hold/nohold or copy parameters set into the output device by the spool command.
SPOOL	Establishes the file attributes that apply to files created on, or read by, the given device. It establishes the CLASS that will be in effect, whether: files are to be automatically held, input files are to be saved or purged after reading, and output files are to be directed to the real system printers and punches or are to be transferred to a user's virtual reader.

Real Device Spooling Commands

The operator can use these commands to control the activity of the real spooling devices:

Command	Meaning
BACKSPAC	Backspaces an active spooling device for either a specified number of pages (printers only) or to the beginning of the file (printers or punches).
DRAIN	Stops the operation of a specified output or input device after it has finished processing the file on which it is currently working. A printer must be drained prior to the issuance of the LCADBUF command. Unit record devices are normally drained prior to system shutdown.
START	Restart a device after it has been drained. Optional parameters allow the operator to specify the spooling output class for the output device, and if output separator records will be created.
FLUSH	Immediately halt the output on the specified device and either flush that copy of the file from the system, or put it into the system hold status for future processing.
REPEAT	Supplement the number of copies requested by the user for the file when it was created. The operator can specify a number from 1 to 99 that is added to the number specified by the user.
LOADBUF	Load the Universal Character Set Buffer of the FCB of the specified printer with the specified image. If requested, the system verifies the loading by printing its contents on the affected printer.
SPACE	Force the output on the specified printer to be single spaced, regardless of the skipping or spacing commands specified by the file's creator.

Spool File Management Commands: The spooling commands are used to alter the attributes and status of closed spool files that are queued and awaiting processing. When a command applies to an individual file, the device type (RDR, PUN, PRT) and the spoolid number must be provided in order to identify the file. It should be noted that in most commands requiring a spoolid, the keyword class followed by a valid spool class or the keyword ALL are acceptable substitutes for the spoolid number. This causes the command to be executed for all files of the given class or device type. The userid is assumed to be the identification of the user issuing the command, except that the system operator must explicitly supply the identification of the user whose files he wishes to affect or he must specify the keyword SYSTEM which gives access to all files (valid for CHANGE, PURGE, ORDER, and TRANSFER commands also).

Command	Meaning
CHANGE	Change the filename and type, the number of copies, and the class of the specified file. Any of the above attributes of a file can be determined via the QUERY command.
HOLD	Place, via the system operator, the specified file in a hold status. The file will not be printed or punched until it is released by the system operator. The operator can hold any user files by device type.
FREE	Opposite of the HOLD command. Allows a file or group of files that were previously held to become available for processing. However, the user cannot reset a hold set by the operator via the HOLD command.
PURGE	Removes unwanted spool files from the system before they are printed or punched.
ORDER	Reorder the input files in a virtual card reader. It can order files by identification number, by class, or by any mixture of the two.
TRANSFER	Transfer a virtual input to another user's virtual reader without any processing. The TRANSFER command causes a changing in the owning userid field in the file's SFBLOK.

SPOOLING ERROR RECOVERY

Unit Record I/O Errors

I/O errors on real spooling unit record devices are handled by a transient routine that is called by DMKIOS after it has sensed the unit check associated with the error on a spooling device. If appropriate, a restart CAW is calculated and DMKIOS is requested to retry the operation, in some cases waiting for a device end that signals that the failing device has been made ready after manual corrective measures have been taken. If after retrying the operation the error is unrecoverable, DMKIOS is informed that a fatal error has occurred. DMKIOS then unstacks the interrupt, flagged as a fatal error, and passes control to real spooling executive. The routines that handle unstacked interrupts in real spooling executive only see operations that have been completed correctly or those that are fatal errors. If a fatal error is unstacked, the recovery mechanism depends on the operation in progress.

For fatal reader errors, processing of the current file is terminated and any portion of the file that has been read and stored on disk is purged. The file's owner is not informed of the presence of a fractional part of the file in the system.

For fatal printer or punch errors, the SFBLOK for the partially completed file is re-queued to the appropriate output list and processing can be resumed by another available printer or punch, or can be deferred until the failing device is repaired.

In any case, the failing device is marked logically offline, and no attempt is made by the system to use it until the operator varies it back online via the VARY command.

DASD Errors During Spooling

DASD I/O errors for page writes are transparent to the user. A new page for the buffer is assigned, the file linkage pointers are adjusted, and the buffer is rewritten. The failing page is not de-allocated and no subsequent request for page space granted access to the failing page. If an unrecoverable error is encountered while reading a page, processing depends on the routine that is reading the file. If the processing is being done for a virtual reader, the user is informed of the error and a unit check/intervention required condition is reflected to the reader. If the processing is being done for a real printer or punch, the failing buffer is put into the system hold status, and processing continues with the next file. In either case, the DASD page is not de-allocated and it is not available for the use of other tasks.

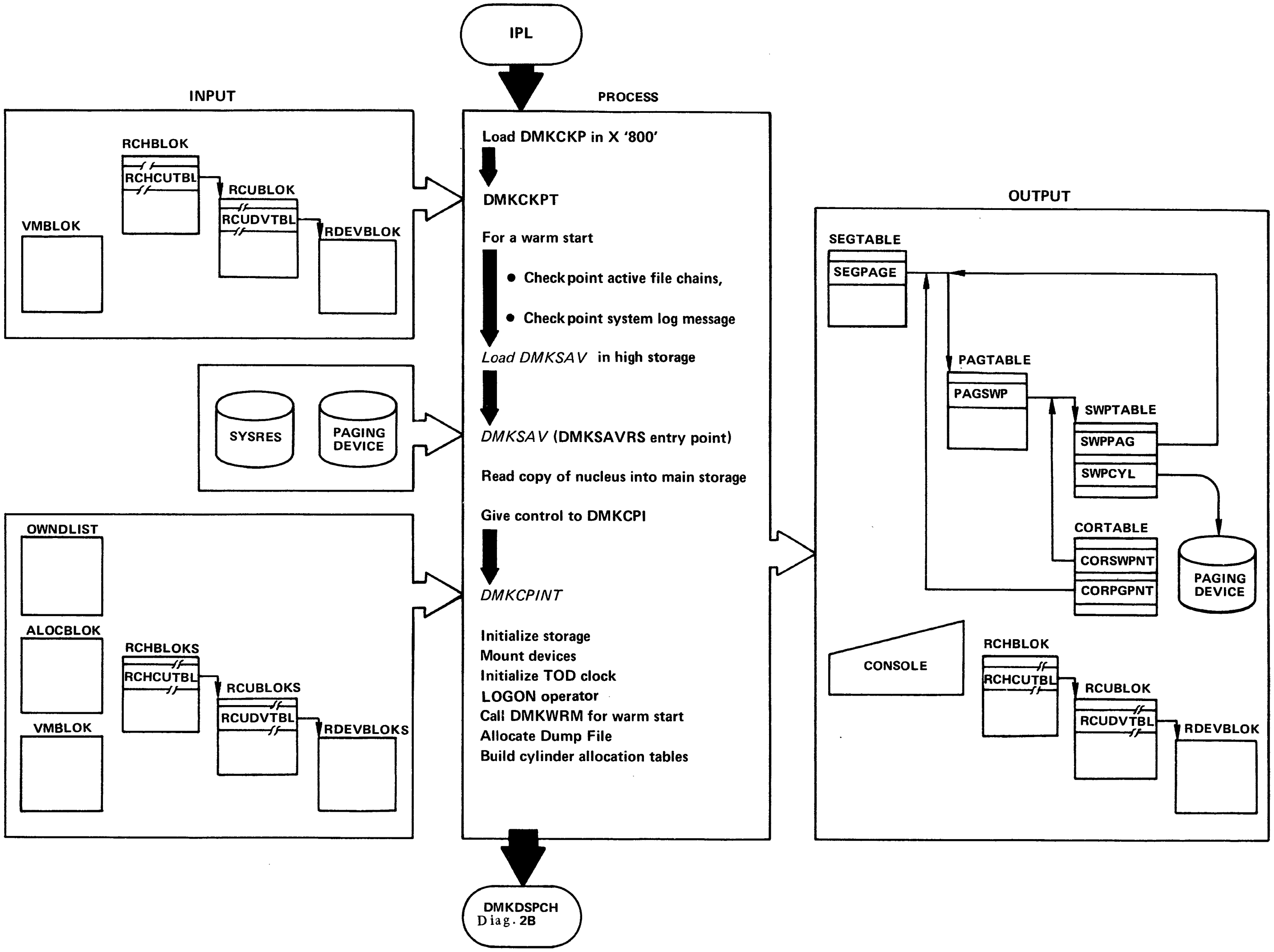
DASD Spool Space Exhausted

If the space allocated for paging and spooling on the system's DASD volumes is exhausted and more is requested by a virtual spooling function, the user receives a message and a unit check intervention required condition is reflected to the virtual output device that is requesting the space, the output file is automatically closed and it is available for future processing. The user can clear the unit check and retry the operation periodically in the hope that space is free or completely restart later from the beginning of the job. If the task requesting the space is the real spooling reader task, the operator receives an error message and the partially complete file is purged. Any time the spooling space is exhausted, the operator is warned by a console message and alarm. However, the system attempts to continue normal operation.

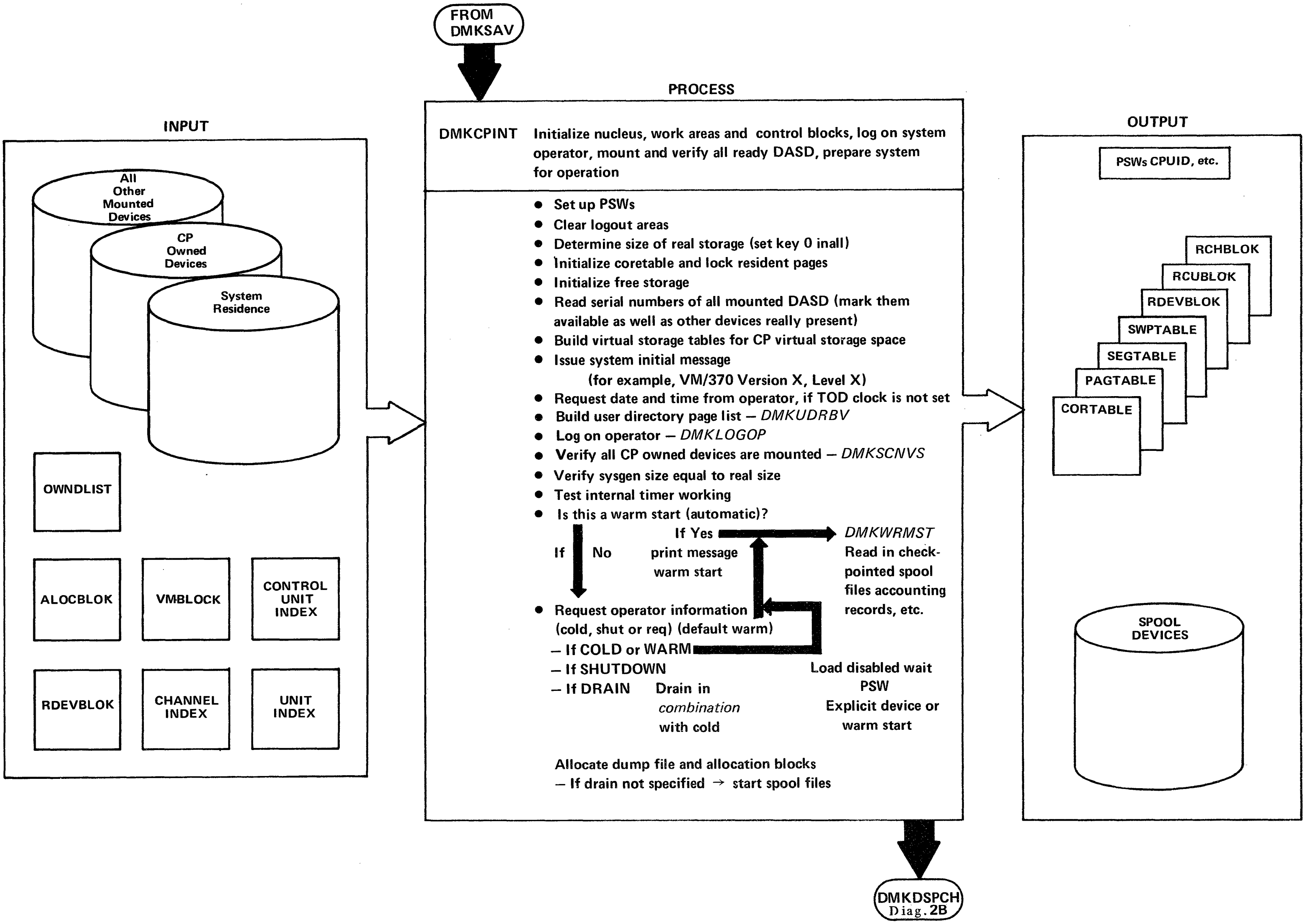
CP INITIALIZATION

System initialization starts when the operator selects the DASD device address of the VM/370 control program System's Residence Volume (SYSRES) and presses the IPL button. The System/370 hardware reads 24 bytes from record 1 of cylinder 0 on SYSRES into location 0 of main storage. This record consists of an initial PSW and a channel program. The channel program is used to read the module DMKCKP into location X'800' and give it

Diag. 5B. CP Initialization



Diag. 5B1. CP IPL



DMKCPINT Initialize nucleus, work areas and control blocks, log on system operator, mount and verify all ready DASD, prepare system for operation

- Set up PSWs
- Clear logout areas
- Determine size of real storage (set key 0 inall)
- Initialize coretable and lock resident pages
- Initialize free storage
- Read serial numbers of all mounted DASD (mark them available as well as other devices really present)
- Build virtual storage tables for CP virtual storage space
- Issue system initial message (for example, VM/370 Version X, Level X)
- Request date and time from operator, if TOD clock is not set
- Build user directory page list – *DMKUDRBV*
- Log on operator – *DMKLOGOP*
- Verify all CP owned devices are mounted – *DMKSCNVS*
- Verify sysgen size equal to real size
- Test internal timer working
- Is this a warm start (automatic)?

If No

If Yes print message warm start → *DMKWRMST* Read in check-pointed spool files accounting records, etc.

- Request operator information (cold, shut or req) (default warm)

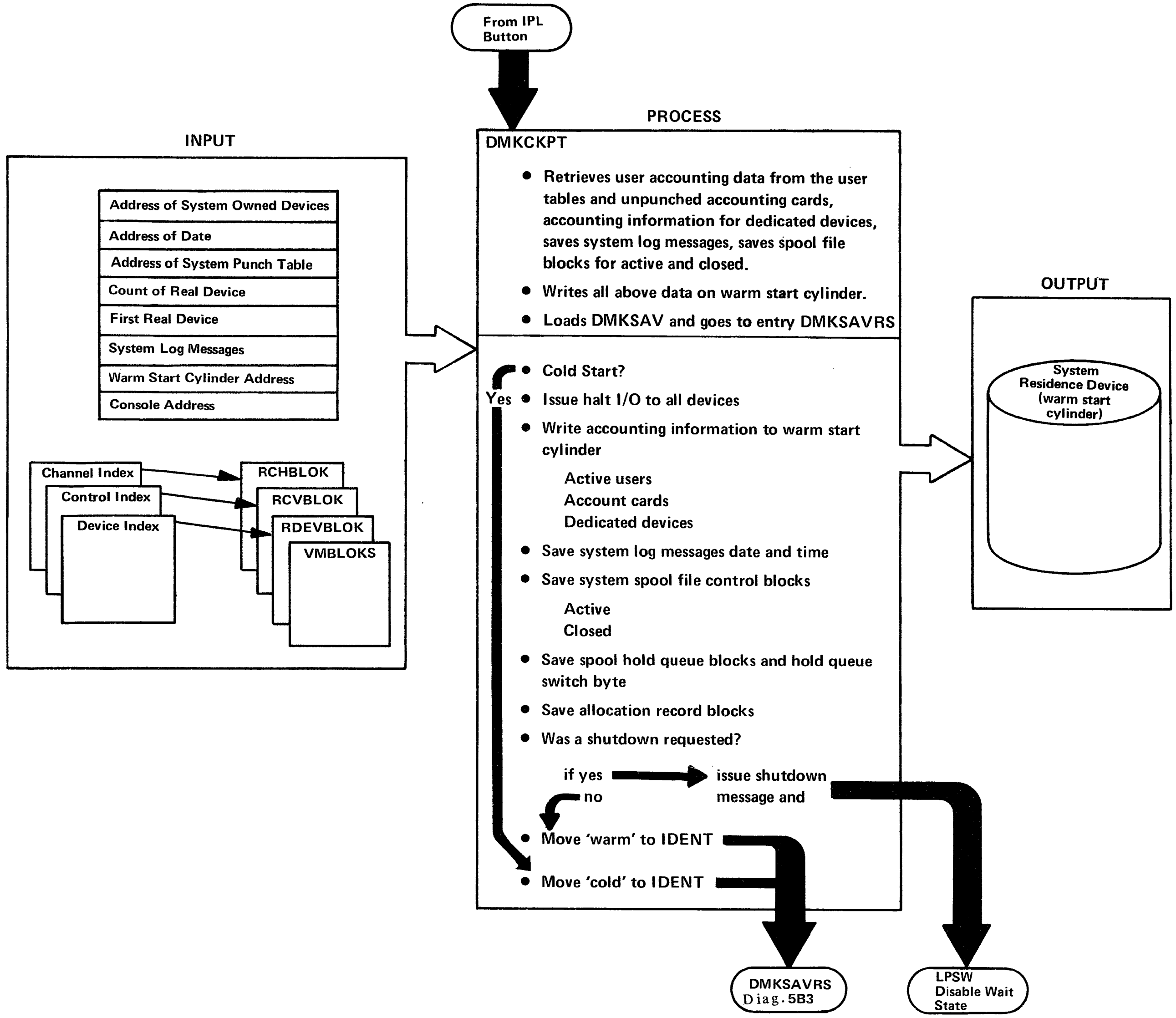
– If COLD or WARM

– If SHUTDOWN Load disabled wait PSW

– If DRAIN Drain in *combination* with cold Explicit device or warm start

Allocate dump file and allocation blocks
– If drain not specified → start spool files

Diag. 5B2. Check Point



DMKCKPT

- Retrieves user accounting data from the user tables and unpunched accounting cards, accounting information for dedicated devices, saves system log messages, saves spool file blocks for active and closed.
- Writes all above data on warm start cylinder.
- Loads DMKSAV and goes to entry DMKSAVRS

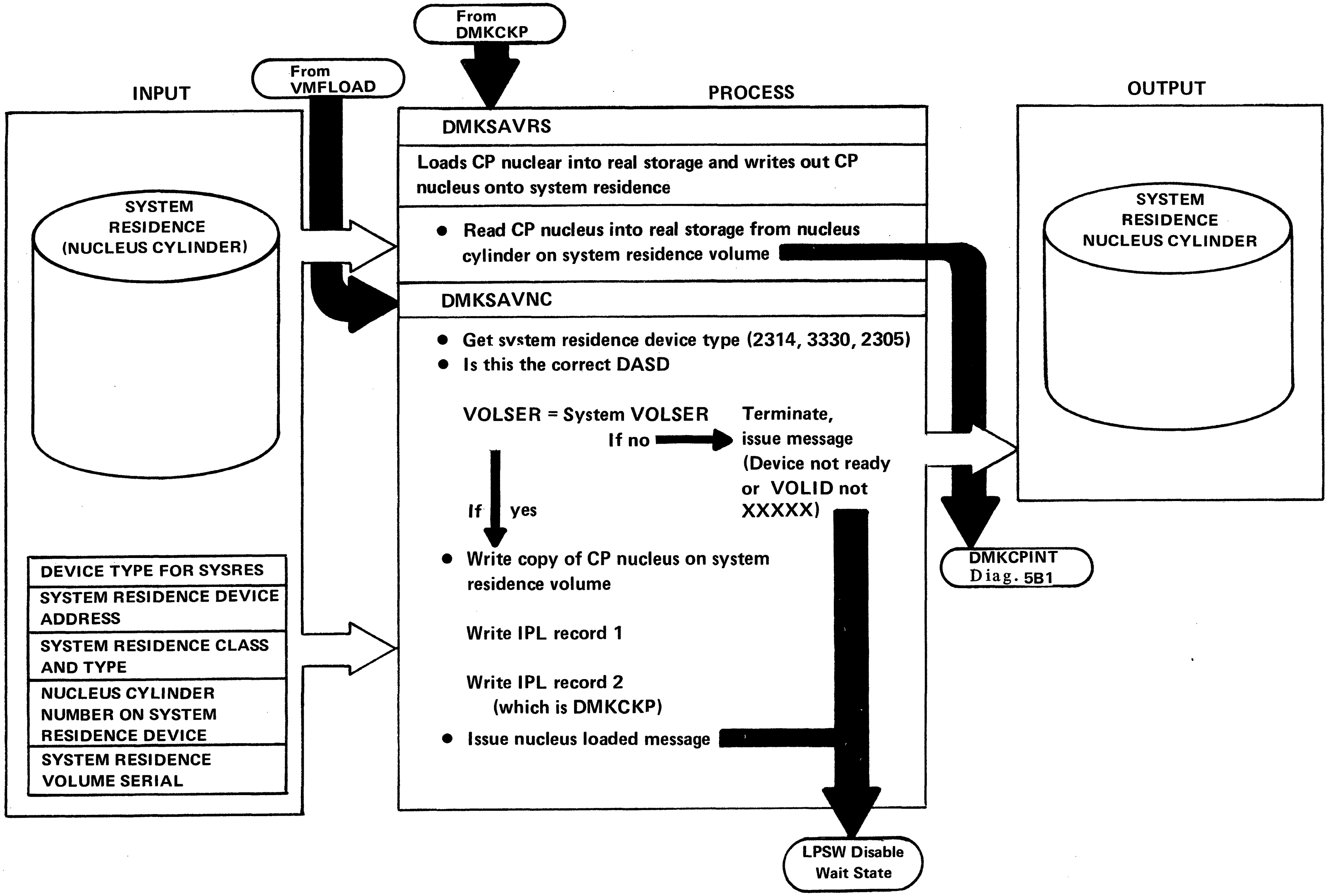
- Cold Start?
- Issue halt I/O to all devices
- Write accounting information to warm start cylinder

  Active users
  Account cards
  Dedicated devices
- Save system log messages date and time
- Save system spool file control blocks

  Active
  Closed
- Save spool hold queue blocks and hold queue switch byte
- Save allocation record blocks
- Was a shutdown requested?

  if yes → issue shutdown message and
  no
- Move 'warm' to IDENT
- Move 'cold' to IDENT

Diag. 5B3. Save System



DMKSAVRS

Loads CP nuclear into real storage and writes out CP nucleus onto system residence

- Read CP nucleus into real storage from nucleus cylinder on system residence volume

DMKSAVNC

- Get svstem residence device type (2314, 3330, 2305)
- Is this the correct DASD

VOLSER = System VOLSER

Terminate, issue message (Device not ready or VOLID not XXXXX)

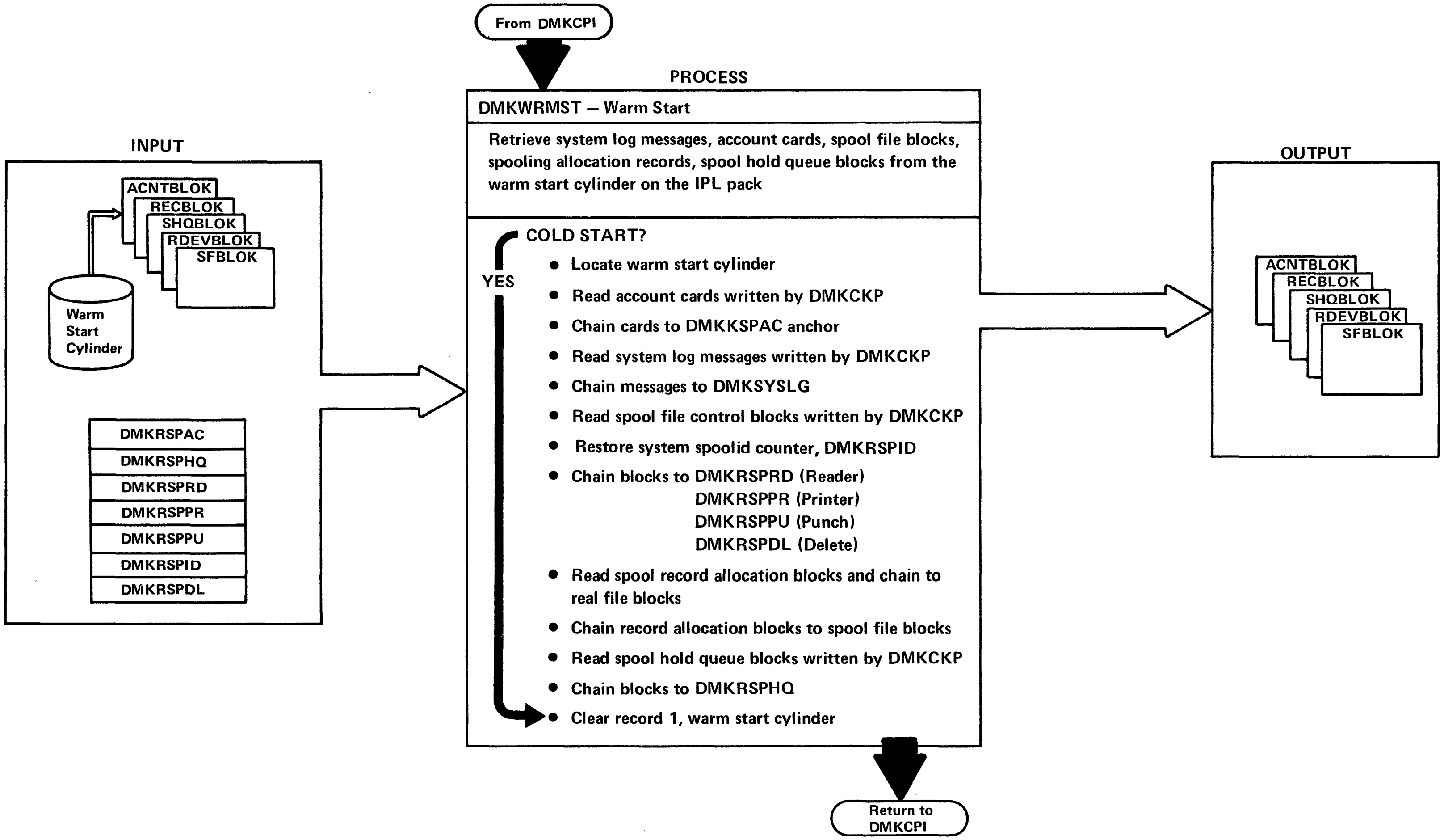
- Write copy of CP nucleus on system residence volume

Write IPL record 1

Write IPL record 2
(which is DMKCKP)

- Issue nucleus loaded message

Diag. 5B4. Warm Start



DMKWRMST — Warm Start

Retrieve system log messages, account cards, spool file blocks, spooling allocation records, spool hold queue blocks from the warm start cylinder on the IPL pack

COLD START?

- Locate warm start cylinder
- Read account cards written by DMKCKP
- Chain cards to DMKKSPAC anchor
- Read system log messages written by DMKCKP
- Chain messages to DMKSYSLG
- Read spool file control blocks written by DMKCKP
- Restore system spoolid counter, DMKRSPID
- Chain blocks to DMKRSPRD (Reader)
  DMKRSPPR (Printer)
  DMKRSPPU (Punch)
  DMKRSPDL (Delete)
- Read spool record allocation blocks and chain to real file blocks
- Chain record allocation blocks to spool file blocks
- Read spool hold queue blocks written by DMKCKP
- Chain blocks to DMKRSPHQ
- Clear record 1, warm start cylinder

control. DMKCKP checks location CPID in module DMKPSA. If this location contains the value CPCP or WARM, then DMKCKP checkpoints the active file chains and saves the system log messages and accounting informaticn; otherwise, a cold start is performed and the checkpointing is not done.

If location CPID contained the value CPCP, then checkpoint loads a wait state PSW at this time.

If location CPID does not contain the value CPCP, then DMKCP loads DMKSAV and passes control to it at entry point DMKSAVRS. DMKSAV reloads a page image copy of the CP nucleus into real storage starting at page 0. When DMKSAV is finished, control is transferred to DMKCPI. DMKCPI performs the main initialization function. This includes calling DMKWRM to perform the warm start function. When DMKCPI has finished it, it passes control to DMKDSPCH. DMKDSPCH loads a wait state PSW to wait for work.

FREE STORAGE MANAGEMENT

DMKFRE is responsible for the management of free storage, and is used within the control program for obtaining free storage for I/O tasks, CCW strings, various I/O buffers, etc. It is used, in fact, for practically all such applications except real channel, control-unit, and device-blocks, and the CORTABLE.

Block sizes of 30 doublewords or less, constituting about 99 per cent of all calls for free storage, are grouped into 10 subpool sizes (3 doublewords each), and are handled by LIFO (push-down stack) logic. Blocks of greater than 30 doublewords are strung off a chained list in the classic manner.

Subpool blocks are generally obtained, when none are available, from the first larger sized block at the end of available free storage with the smaller sizes. Large blocks, on the other hand, are obtained from the high-numbered end of the last larger block. This procedure tends to keep the volatile small subpool blocks separated from the large blocks, some of which stay in storage for much longer periods of time; thus, undue fragmenting of available storage is avoided.

DMKFRE initially starts without any subpool blocks; they are obtained from DMKFREE and returned to DMKFRET on a demand basis.

The various cases of calls to DMKFREE for obtaining free storage, or to DMKFRET for returning it, for subpool sizes and large sizes, are handled as follows:

Call to DMKFREE for a Subpool Size

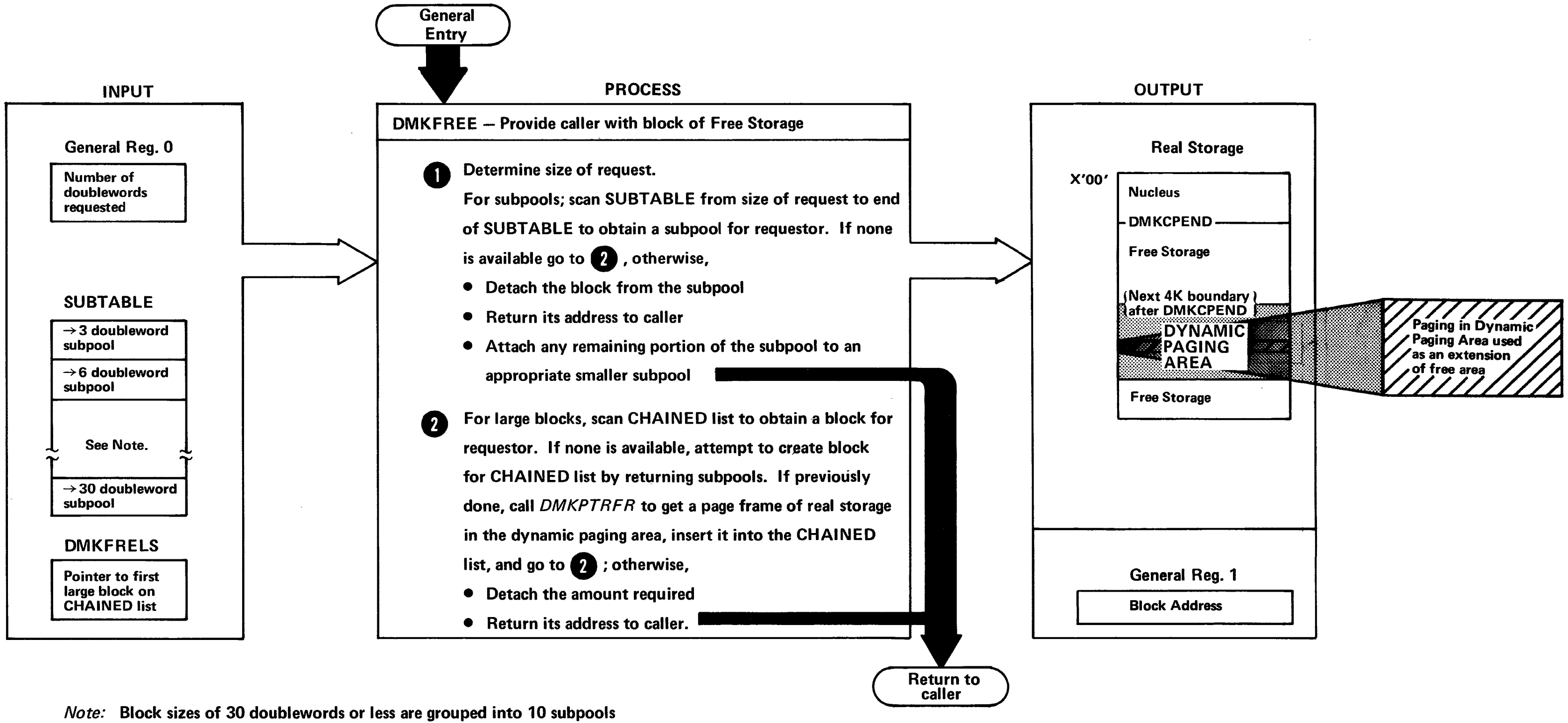
Subpool Available: If a call for a subpool size is made and a block of the suitable size is available, the block found is detached from the chain, the chain patched to the next subpool block of the same size (if any), and the given block returned to the caller.

Subpool Not Available: If there is no suitable block when a call to DMKFREE is made for a subpool size, a check is made to see if any larger subpool block can profitably be split up into the size requested and another subpool size. If this is feasible, the larger block is detached from its subpool and split. The requested block is returned to the caller, and the remaining block is attached to its subpool: Otherwise, the chained list of free storage is searched for a block of equal or larger size. The first block of larger or equal storage is used to satisfy the call (an equal-size block taking priority), except that blocks within the dynamic paging area are avoided if at all possible. If no equal or larger block is found, all the subpool blocks currently not in use are returned to the main free storage chain, and then the free storage chain is again searched for a block big enough to satisfy the call. If there still is no block big enough to satisfy the request, then DMKPTRFR is called to obtain another page of storage from the dynamic paging area, and the process is repeated to obtain the needed block.

Call to DMKFREE for a Large Block

If a call to DMKFREE is made for a block larger than 30 doublewords, then the chained list of free storage is searched fcr a block of equal or larger size. If an

Diag. 6B1. Free (Provide) a Block of Storage



General Entry

INPUT

General Reg. 0

Number of doublewords requested

SUBTABLE

→3 doubleword subpool

→6 doubleword subpool

See Note.

→30 doubleword subpool

DMKFRELS

Pointer to first large block on CHAINED list

PROCESS

DMKFREE – Provide caller with block of Free Storage

1. Determine size of request.
For subpools; scan SUBTABLE from size of request to end of SUBTABLE to obtain a subpool for requestor. If none is available go to 2, otherwise,
- Detach the block from the subpool
- Return its address to caller
- Attach any remaining portion of the subpool to an appropriate smaller subpool

2. For large blocks, scan CHAINED list to obtain a block for requestor. If none is available, attempt to create block for CHAINED list by returning subpools. If previously done, call *DMKPTRFR* to get a page frame of real storage in the dynamic paging area, insert it into the CHAINED list, and go to 2; otherwise,
- Detach the amount required
- Return its address to caller.

Return to caller

OUTPUT

Real Storage

X'00'

Nucleus

DMKCPEND

Free Storage

(Next 4K boundary after DMKCPEND)

DYNAMIC PAGING AREA

Free Storage

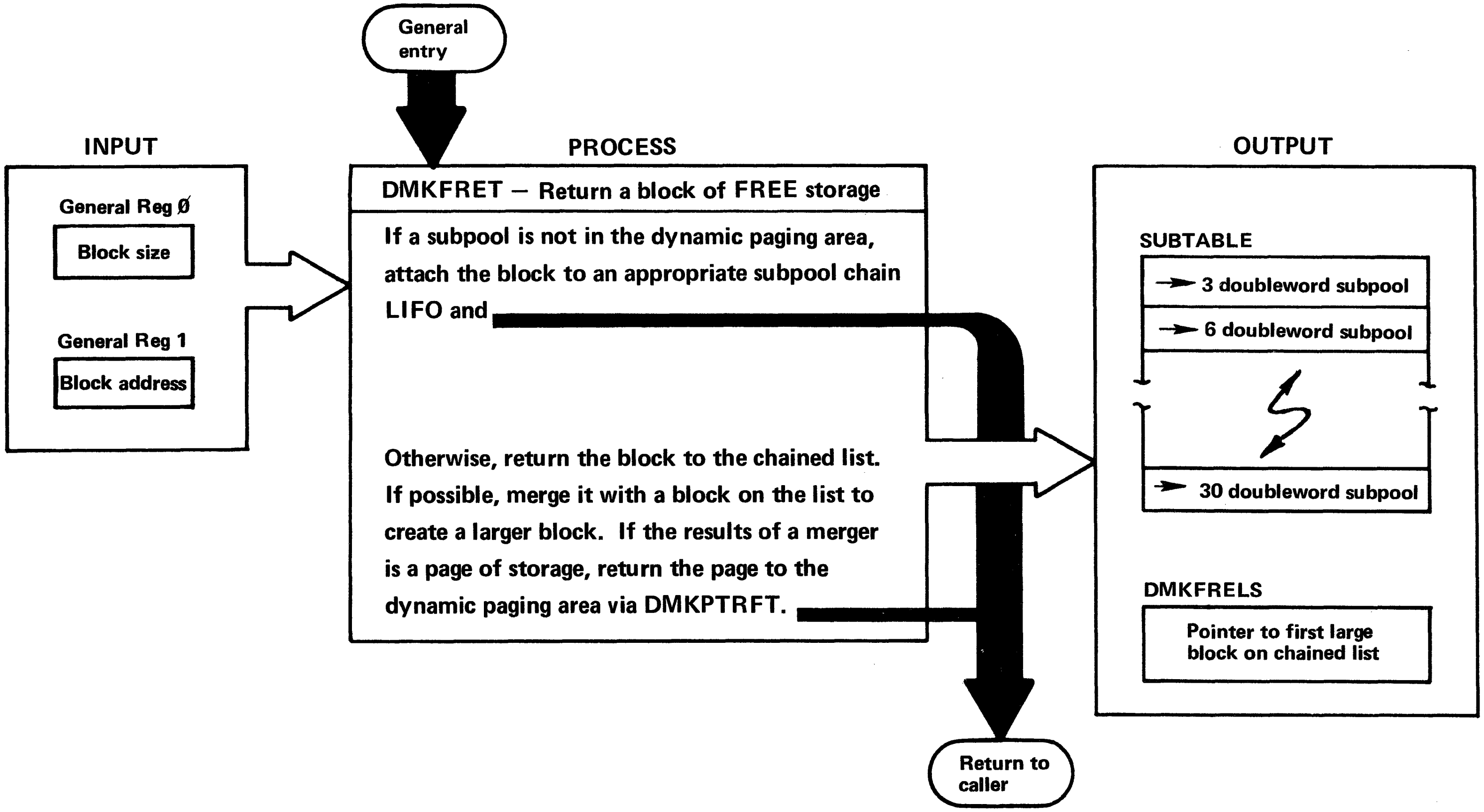
Paging in Dynamic Paging Area used as an extension of free area

General Reg. 1

Block Address

Note: Block sizes of 30 doublewords or less are grouped into 10 subpools (3,6 . . . 30 doublewords, multiples of 3 doublewords); the subpools are processed LIFO. The SUBTABLE contains pointers to the subpools. Block sizes greater than 30 doublewords chained together by a CHAINED list. DMKFRELS points to the first block on the list.

Diag. 6B2. Return a Block of Storage



DMKFRET — Return a block of FREE storage

If a subpool is not in the dynamic paging area, attach the block to an appropriate subpool chain LIFO and

Otherwise, return the block to the chained list. If possible, merge it with a block on the list to create a larger block. If the results of a merger is a page of storage, return the page to the dynamic paging area via DMKPTRFT.

equal size block is found it is detached from the chain and given to the caller. If at least one larger block is found, the desired block size is split off the high numbered end of the last larger block found, and given to the caller. If no equal or larger block is found, DMKPTRFR is called to obtain another page of storage from the dynamic paging area, and the above process is repeated (as necessary) to obtain the needed block.

Call to DMKFRET for a Subpool Size

If a subpool size block is given back via a call to DMKFRET, the block is attached to the appropriate subpool chain on a LIFO (push-down stack) basis, and return is made to the caller. If, however, the block was in a page within the dynamic paging area, the block is returned to the regular free storage chain instead.

Call to DMKFRET for a Large Block

If a block larger than 30 doublewords is returned via DMKFRET, it is merged appropriately into the regular free storage chain. Then, unless the block was returned by DMKFRETR (see the section, "Initialization", a check is made to see if the area given back (after all merging has been done) is a page within the dynamic paging area. If so, it is returned to the dynamic paging area via DMKPTRFT for subsequent use.

Initialization

The number of pages allocated to free storage depends upon the number of storage boxes upon which the VM/370 control program is running, and is initialized by DMKCPINT (usually 6 pages per 256K). DMKFRETR is called by DMKCPINT to merge available blocks of storage into the regular free storage chain regardless of their size.

CONSOLE FUNCTIONS

DMKCFM analyzes VM/370 control program commands and pass control to the appropriate routine to handle the command. DMKCFM can be entered via the attention key at the user's terminal or directly from a virtual machine.

When a console interrupt occurs via the attention key at the user's terminal, DMKIOSIN calls DMKCNSIN to handle the unsolicited interrupt, then DMKCNSIN calls DMKCFMBK.

DMKCFMBK first calls DMKFREE to obtain storage for an 18 doubleword input buffer. Next, DMKQCNWT is called to send the message CP to the terminal to inform the user that he has entered console function mode. DMKQCNRD is then called to read the console function request.

DMKCFMEN is the entry point for commands coming directly from the virtual machine. DMKPRGIN enters here when a DIAGNOSE instruction with a code of 8 is detected. The address of an 18 doubleword input buffer is passed in register 1; therefore, a read to the terminal is not needed.

After either the read to the terminal or entry from the virtual machine, DMKSCNFD is called to find the command type. On return from DMKSCNFD, register 1 points to the start of the command and register 0 contains the length of the command. The entered command is matched against a list of valid commands. The list contains a 16-byte entry for each command. Each entry contains 8 bytes for the name, 2 bytes for class mask, 2 bytes for an abbreviation count, and 4 bytes containing the routine address. If the entered command matches an entry in the list, it is then checked to ensure that a valid abbreviation for the command has been used. If this test is not successful, DMKSCN continues to scan the list for a valid command. Should the abbreviation be valid, a check is then made to determine if this user is of the proper class to use the command entered. If this is successful, DMKCFM then calls the appropriate routine to process the command.

After the command has been processed, control is returned to DMKCFM. There are three possible returns. On a normal return, the input buffer is scanned to see if there are any more commands. If none exist, DMKCFM returns to the virtual machine (if entered via

DIAGNOSE) or calls DMKQCNRD to read the next command from the terminal. On a return plus 4, the VMCFWAIT bit is turned off to allow the virtual machine to run. DMKFRET is called to return the input buffer storage. Then control returns to either the virtual machine, if entered via a diagnose or to DMKDSPCH, if entered via the attention key. On a return plus 8, the operation is the same as plus 4 except the VMCFWAIT bit is left on.

SYSTEM/USER INTERFACE

Attaching a User to the System

After CP has been initialized, the communication lines are enabled by DMKCPVEN. Then an individual user is attached to the system using the following steps:

1. Terminal Identification

When the Control Program receives the initial interrupt from a terminal on an enabled line (normally initiated by a user dialing in on a data-set), the DMKCNSID routine is entered. DMKCNSID determines the terminal device type, stores this information in the terminal device block, writes the online message and puts the terminal line in a state to receive an attention.

2. Attention from User

After the online message has been typed at the user's terminal, and he has pressed the Attention key, DMKCNSIN (the console-interrupt routine) calls DMKBLDVM to build a skeleton VMBLOK for the user. At this time, the USERID is LOGONxxx, where xxx is the terminal real device address, and a flag is set to indicate that the user has not yet completed the LOGON process.

Then DMKCNSIN calls DMKCFMBK, which types a single blank at the terminal, issues a read to the terminal, for the user to enter his first command (normally LOGON or DIAL).

3. First Command from User

After the first command has been entered by the user, DMKCNSIN further determines the type of terminal. If the terminal is a 2741, DMKTRMID is called to identify it as either a 2741P (PTTC/EBCD) or a 2741C (Correspondence) terminal. If successful, the correct device type and translate tables for input and output are set; if not, flags are set to indicate the terminal is not yet identified.

Then control is returned to DMKCFMBK, which determines if the first command is valid (for example, LOGON, MSG, or DIAL). If the first command is not valid, a restart message is given, and the read to the terminal posted again for the first command. If the first command was LOGON (or its abbreviation), DMKLOGON is called to complete the process of attaching the user to the system.

4. LOGON of User

Operations performed by DMKLOGON include the following:

- Ensuring that the maximum number of users allowed on the system is not being exceeded.

- Obtaining the userid from the command line, and checking for a possible password and other optional parameters.

- Checking the userid and password (entered separately if not on the LOGON command line) against entries in CP's directory of users.

- Ensuring that the user is not logged on at another terminal (an error condition), or reconnecting the user if he was running, but in the disconnect mode.

- Obtaining pertinent information on the user's virtual machine from the User Machine Block portion of the directory.

- Storing the correct userid (replacing the LOGONxxx userid used up until now), virtual storage size, and other vital information in the user's VMBLOK.

- Allocating and initializing segment, page, and swap tables (necessary for handling of the user's virtual storage).

- Allocating an extended VMBLOK (ECBLOK) if the user's virtual machine has the capability of running in the extended control mode.

- Allocating and initializing virtual device blocks, control unit blocks, and channel blocks, using information from the User Device Blocks portion of the directory.

- Establishing links (as feasible) to all DASD devices included in the user's directory, the accessibility of any disk being determined by the user access mode in the user's directory, and whether any other user(s) are presently linked to the disk, in read-mode and/or write-mode.

- Initializing all other virtual device blocks as appropriate, such as reader, punch, printer, and terminal.

- Mapping all virtual devices to real devices.

- Performing appropriate accounting.

- Informing the user of the date-time of the most recent revision to the system log message (LOGMSG), and of the presence of any outstanding spooled files in his virtual reader, printer, or punch.

- Sending a ready message to the user with the date-time (and weekday), and a message to the system operator indicating the user has logged on.

If the user has a device address or a named system in his user directory and he has not suppressed its initialization via an option on the LOGON command line, then that device or named system is then loaded (via IPL) at the conclusion of the LOGON process. Otherwise, when the LOGON functions are complete, the user is placed in the console function mode with a read on his terminal, ready for the entry of his first desired command.

Under the latter condition of no automatic IPL, the user can IPL an alternate nucleus by using the STOP option in the IPL command. This option will cause the normal IPL procedure to halt execution, prior to loading the initial PSW, and issue a diagnose code 8 placing the user in CP console function mode. A hexadecimal character entered in location X'08' will change the nucleus name. A hexadecimal character entered in location X'09' will change the apparent storage size. The BEGIN command allows the IPL procedure to continue.

User I/O Reconfiguration

Three commands are available to alter the I/O configuration of a user's virtual machine after he has logged on to the system. Two of the commands are available to the user, while the third is restricted to the system operator, since it affects the status of real devices attached to the system. The ATTACH and DETACH commands are contained in DMKVDB and DEFINE in DMKDEF. Both pageable modules are called by the system command scanner (DMKCFM) after their format and privilege classes have been validated. These commands access the same control-block building subroutines in the module DMKVDS that are used by the LOGON processor DMKLOG.

Attaching a Real Device: The system operator can dedicate a real device of any type to a single user by issuing the ATTACH command. The device attached is available only to the given user, and all I/O requests to it are handled via CCW translation. If the device is a DASD, cylinder relocation does not occur when seek addresses or home addresses are referenced. The I/O Supervisor does not queue operations on the device, nor automatically restart it nor do ordered seek queueing. Nonsharable devices such as tape drives must be attached to a user in order to be accessed by a virtual machine. A user can also have a dedicated card read/punch or printer. However, this is usually not necessary because of the unit record spooling facilities of CP. Unit record input or output on a dedicated (attached) device is not spooled by CP. The unit attached may be given a different virtual address than its real address; however, the user may not already have a virtual device at the attached address. A real device cannot be attached (1) if it is currently dedicated to another user, (2) if it contains mini-disks that are in use by other users, or (3) if it is a system owned volume that is in use for spooling or paging.

Defining a Virtual Device: A system user can DEFINE a new virtual device that does not require the dedication of a corresponding real device. Devices that can be defined are consoles, spooled readers, punches and printers, dialable TP lines, virtual channel-to-channel adapters, pseudo timers, and temporary disks. With DEFINE, the user can change any existing virtual device address whether it corresponds to a shared or dedicated real device or no real device unit.

Temporary disks are dynamically obtained cylinders of DASD storage space. They are available to the user for as long as they are part of his virtual configuration, but the data on them is destroyed after the user detaches the area. For all other purposes, however, they appear to be a standard disk.

Detaching a Virtual Device: A virtual device can be removed from a users configuration prior to logout via the DETACH command. A user can detach any of his own devices, and the system operator can detach a real device from a user. In this case, the user is informed of the operator's action. A real device can be detached only if it is dedicated to a single user or is attached to the system and is not in use when the DETACH is issued.

VIRTUAL CONSOLE SIMULATION

DMKVCN receives control from the virtual machine I/O executive, DMKVIO. When control is received, the device is available with no interrupts pending. A console control block, VCONCTL, that is obtained from storage and chained from the virtual device control block, VDEVBLOCK, by DMKLOG is accessed for use during the interpretation of the virtual console I/O sequence. The user's CAW is examined for validity. If it is valid, the TRANS macro is issued to fetch the first user CCW. This CCW is moved to the VCONCTL block for analysis.

The CCW is analyzed to determine if it is a read, a write, a control, a sense, a TIC, or an invalid operation. Based upon the analysis, the appropriate processing routine in DMKVCN is invoked.

The Read Simulation Routine: Obtains a buffer for input data from FREE storage. The location of the buffer is remembered in the VCONCTL block. The DMKQCNRD routine is called to schedule and perform an actual read to the corresponding real device representing the user's virtual console. If SET LINEDIT ON is specified, the buffer data is edited and translated to EBCDIC. When the read is completed, the data is moved to the specified user address obtained from the address portion of the virtual CCW. If command chaining is specified, processing returns to fetch and analyze the next CCW. If command chaining is not specified, the virtual CSW is constructed in the VDEVBLOK and an interrupt is flagged as sending in the VMBLOK.

The Write Simulation Routine: Obtains a buffer for construction of the output message from free storage. The users data is located from the virtual CCW address in the VCONCTL block and moved to the data buffer. The DMKQCNWT routine is called to write the data in the buffer and provide the necessary length, translation, and format functions. Control is received back at the DMKVCN module upon completion of the writing. At this point, the virtual CCW is re-examined. If command chaining is specified, processing continues to fetch and analyze the next CCW. If command chaining is not specified, the virtual CSW is constructed in the VDEVBLOK and an interrupt is flagged as pending in the VMBLOK.

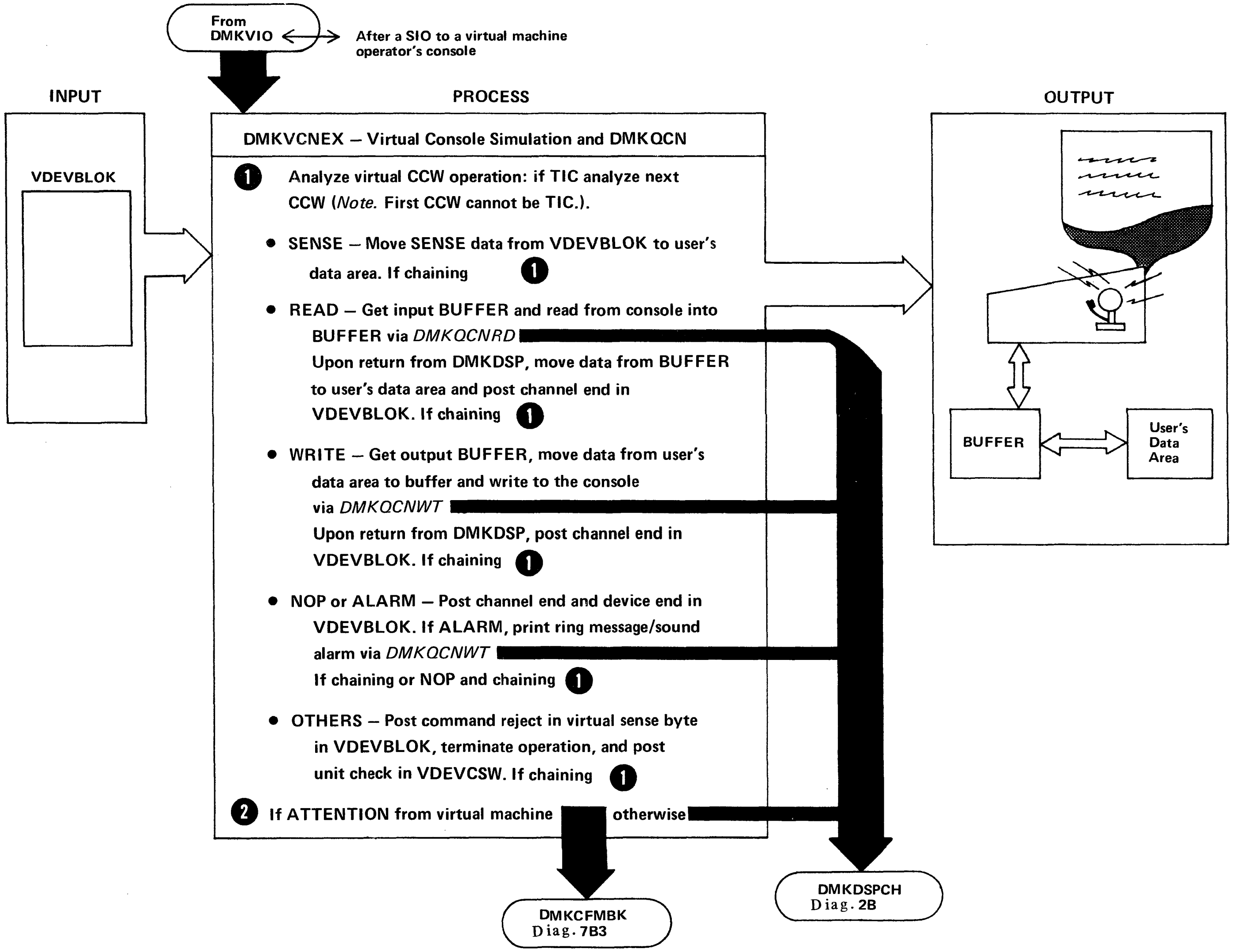
The Control Simulation Routine: Is used for the NOP and ALARM operations. A NOP operation requires no data transfer or I/O operation. An ALARM operation has no equivalent on low speed teleprocessing equipment; thus, a message indicating the alarm operation is constructed. DMKQCNWT is called to output the constructed message. If the command is chained, processing continues (for NOP or ALARM) to fetch the next CCW and analyze it. If command chaining is not specified and this is not the first CCW, a virtual CSW is constructed in the VDEVBLOK and an interrupt is flagged as pending in the VMBLOK. If this is the first (and only) CCW, then a condition code of 1 is presented with channel end and device end in the virtual CSW.

A Virtual Sense Operation: Is similar to a control operation, because no actual I/O operation is performed. However, there is data transfer. The sense data from the VDEVBLOK is moved to the virtual storage location specified in the virtual CCW address. If the command is chained, processing continues to fetch the next CCW and analyze it. Otherwise, an interrupt is flagged as pending in the VMBLOK.

A Virtual TIC: Fetches the virtual CCW addressed by the TIC address and analysis of the fetched CCW continues. If the fetched CCW is itself a TIC, or if the TIC is the first CCW, a channel program check condition is reflected to the virtual machine as an interrupt or as a CSW stored condition respectively.

Any other operation is considered invalid. Command reject status is posted in the virtual sense byte and the operation is terminated with unit check status presented in the virtual CSW.

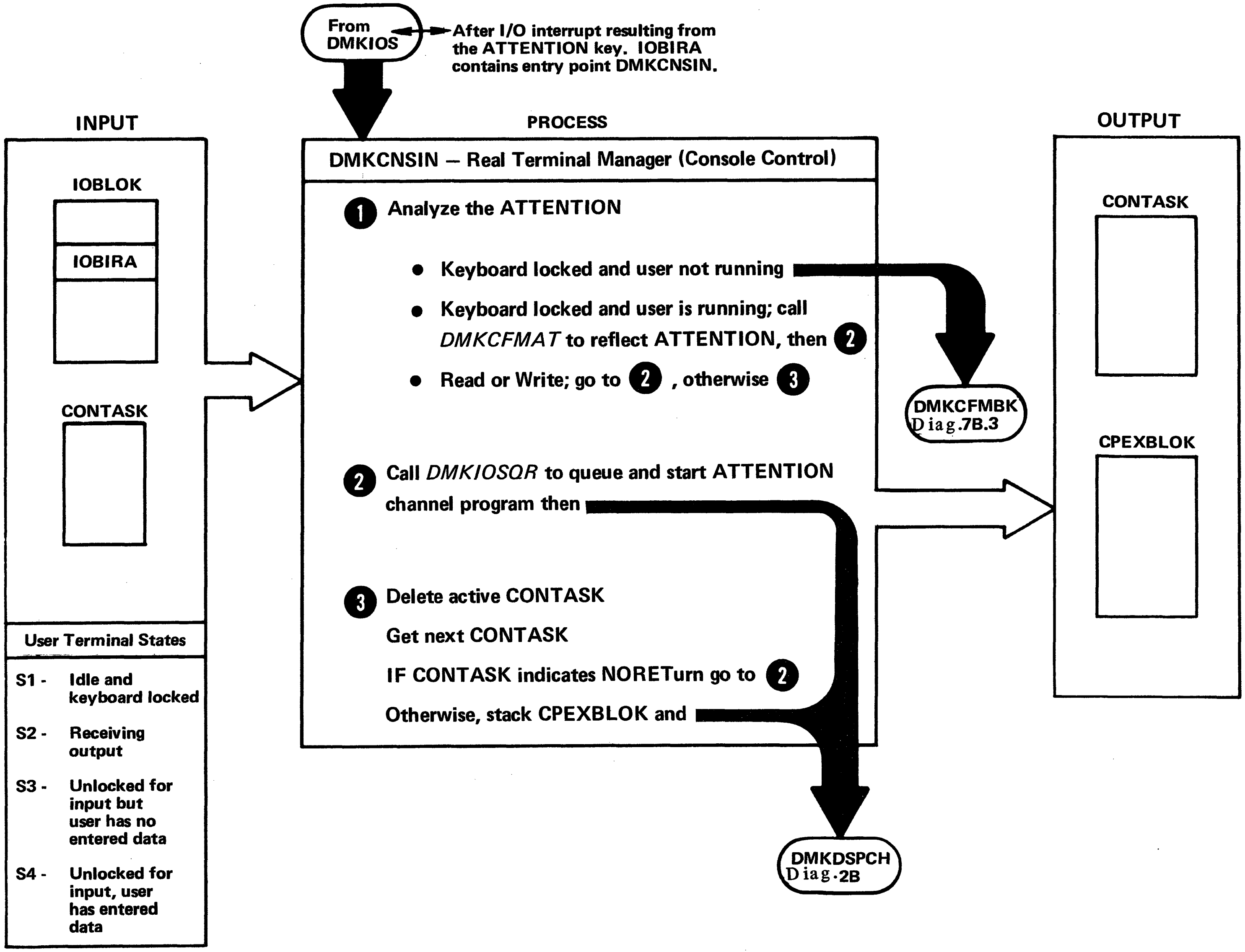
Diag. 7B1. Virtual Console Simulation, Real Terminal Operation



DMKVCNEX – Virtual Console Simulation and DMKQCN

1. Analyze virtual CCW operation: if TIC analyze next CCW (*Note.* First CCW cannot be TIC.).
   - SENSE – Move SENSE data from VDEVBLOK to user's data area. If chaining 1
   - READ – Get input BUFFER and read from console into BUFFER via *DMKQCNRD*
     Upon return from DMKDSP, move data from BUFFER to user's data area and post channel end in VDEVBLOK. If chaining 1
   - WRITE – Get output BUFFER, move data from user's data area to buffer and write to the console via *DMKQCNWT*
     Upon return from DMKDSP, post channel end in VDEVBLOK. If chaining 1
   - NOP or ALARM – Post channel end and device end in VDEVBLOK. If ALARM, print ring message/sound alarm via *DMKQCNWT*
     If chaining or NOP and chaining 1
   - OTHERS – Post command reject in virtual sense byte in VDEVBLOK, terminate operation, and post unit check in VDEVCSW. If chaining 1
2. If ATTENTION from virtual machine otherwise

Diag. 7B2. Console Function Control



After I/O interrupt resulting from the ATTENTION key. IOBIRA contains entry point DMKCNSIN.

DMKCNSIN – Real Terminal Manager (Console Control)

1 Analyze the ATTENTION

- Keyboard locked and user not running
- Keyboard locked and user is running; call *DMKCFMAT* to reflect ATTENTION, then 2
- Read or Write; go to 2 , otherwise 3

2 Call *DMKIOSQR* to queue and start ATTENTION channel program then

3 Delete active CONTASK

Get next CONTASK

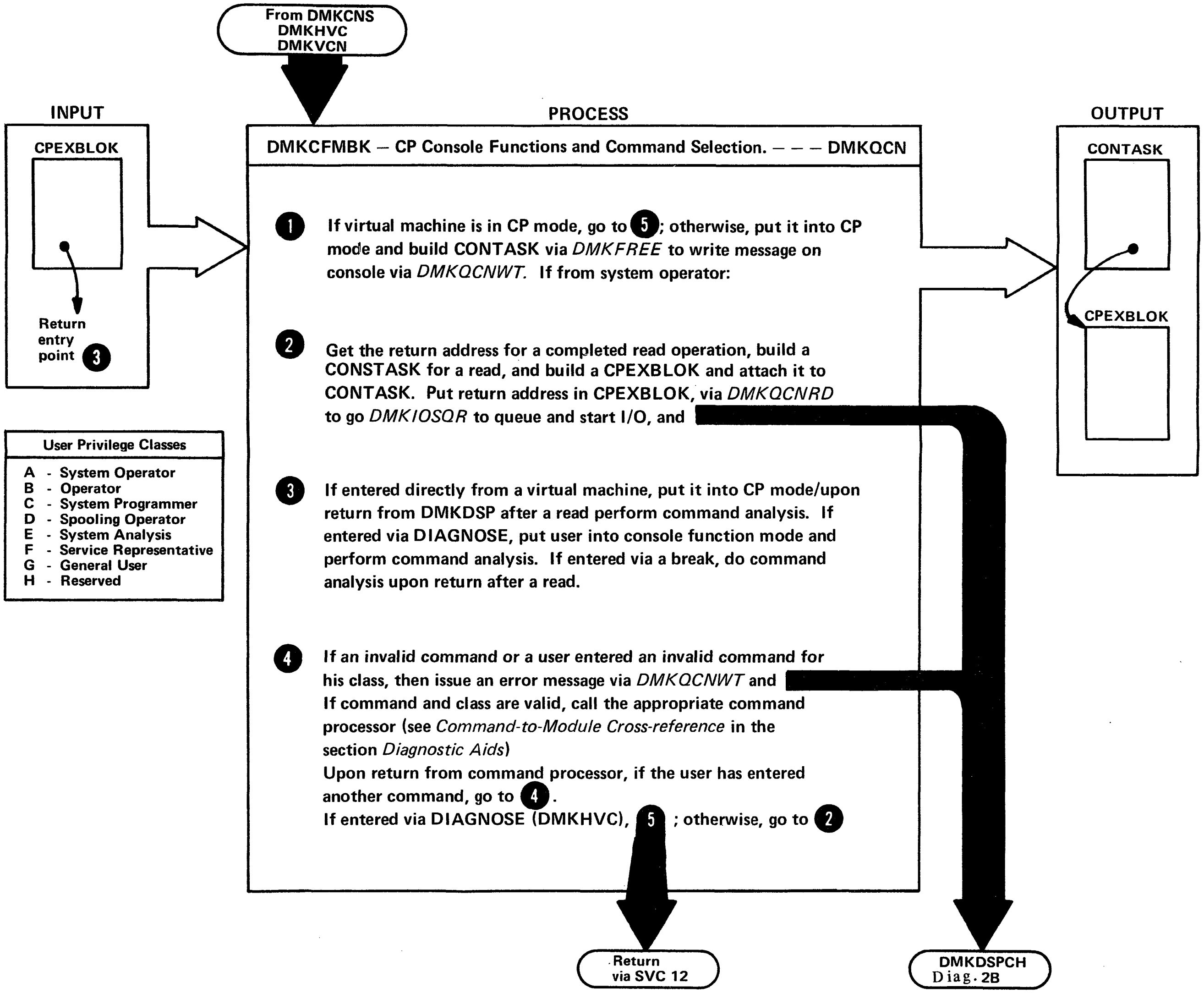
IF CONTASK indicates NORETurn go to 2

Otherwise, stack CPEXBLOK and

User Terminal States

S1 -	Idle and keyboard locked
S2 -	Receiving output
S3 -	Unlocked for input but user has no entered data
S4 -	Unlocked for input, user has entered data

Diag. 7B3. Function Call Control, Command Selection



DMKCFMBK – CP Console Functions and Command Selection. – – – DMKQCN

1 If virtual machine is in CP mode, go to 5; otherwise, put it into CP mode and build CONTASK via *DMKFREE* to write message on console via *DMKQCNWT.* If from system operator:

2 Get the return address for a completed read operation, build a CONSTASK for a read, and build a CPEXBLOK and attach it to CONTASK. Put return address in CPEXBLOK, via *DMKQCNRD* to go *DMKIOSQR* to queue and start I/O, and

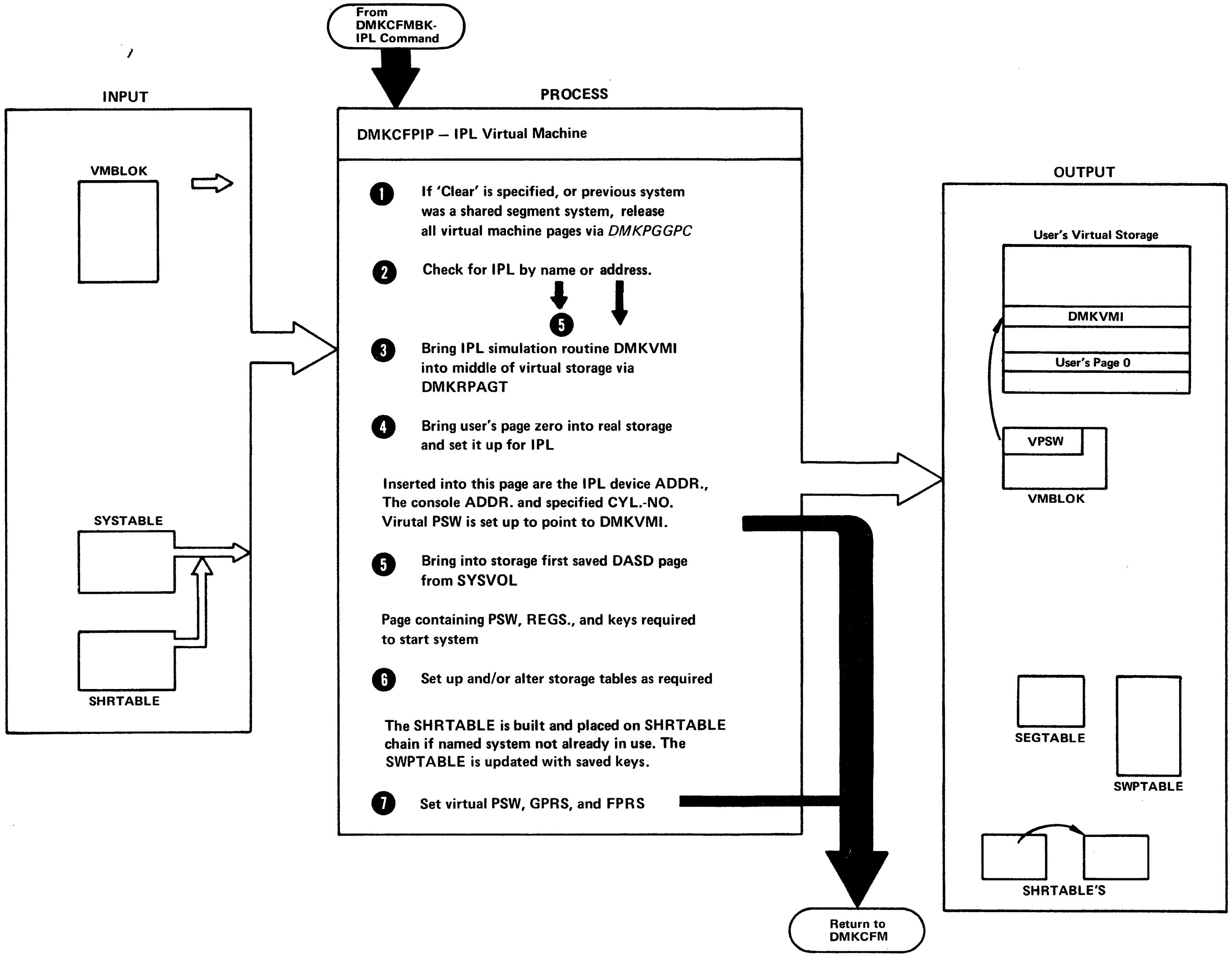
3 If entered directly from a virtual machine, put it into CP mode/upon return from DMKDSP after a read perform command analysis. If entered via DIAGNOSE, put user into console function mode and perform command analysis. If entered via a break, do command analysis upon return after a read.

4 If an invalid command or a user entered an invalid command for his class, then issue an error message via *DMKQCNWT* and
If command and class are valid, call the appropriate command processor (see *Command-to-Module Cross-reference* in the section *Diagnostic Aids*)
Upon return from command processor, if the user has entered another command, go to 4.
If entered via DIAGNOSE (DMKHVC), 5 ; otherwise, go to 2

User Privilege Classes

A	-	System Operator
B	-	Operator
C	-	System Programmer
D	-	Spooling Operator
E	-	System Analysis
F	-	Service Representative
G	-	General User
H	-	Reserved

| Diag. 7B3.1. Virtual Machine IPL



DMKCFPIP – IPL Virtual Machine

1 If 'Clear' is specified, or previous system was a shared segment system, release all virtual machine pages via *DMKPGGPC*

2 Check for IPL by name or address.

5

3 Bring IPL simulation routine DMKVMI into middle of virtual storage via DMKRPAGT

4 Bring user's page zero into real storage and set it up for IPL

Inserted into this page are the IPL device ADDR., The console ADDR. and specified CYL.-NO. Virutal PSW is set up to point to DMKVMI.

5 Bring into storage first saved DASD page from SYSVOL

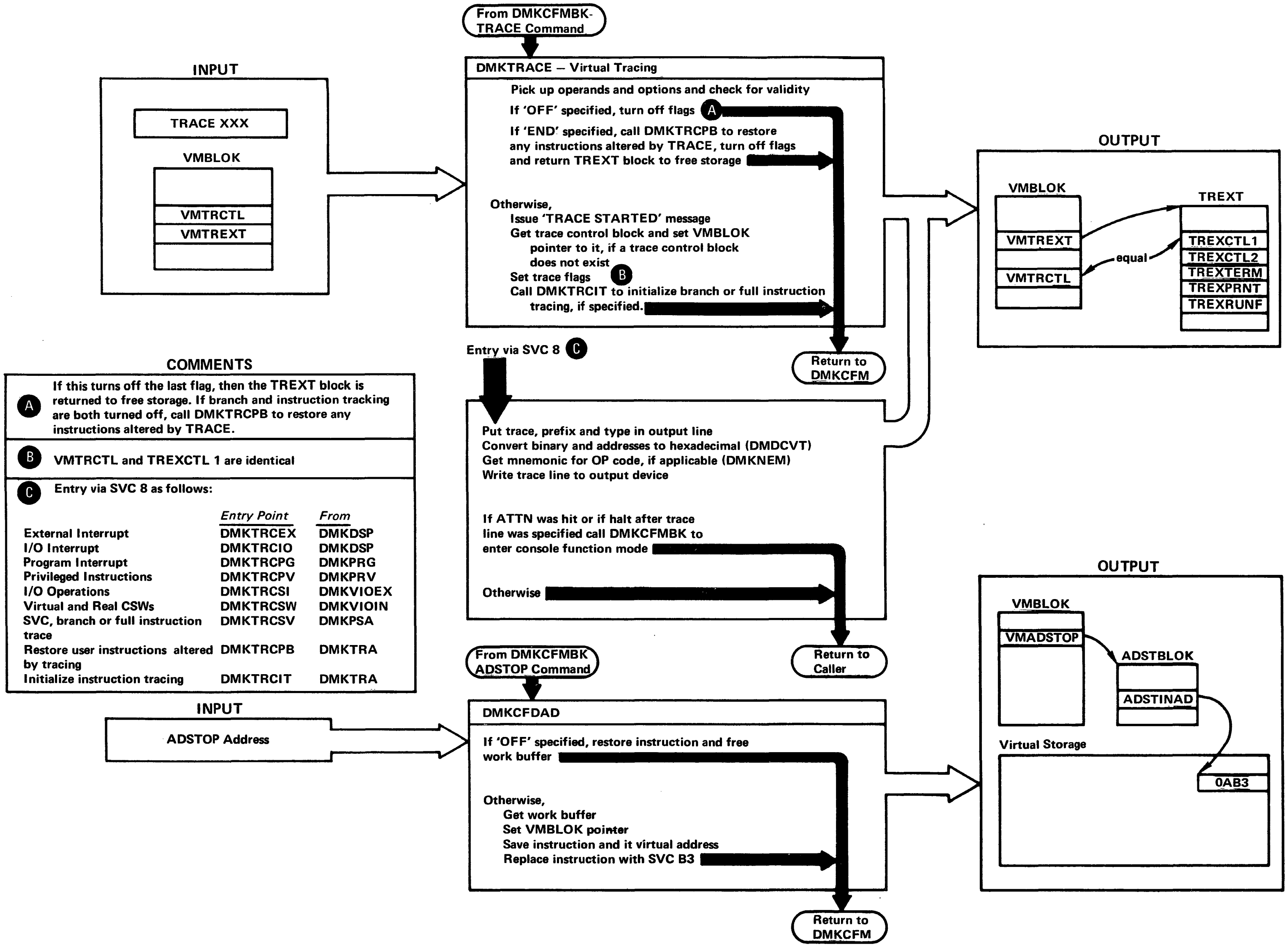
Page containing PSW, REGS., and keys required to start system

6 Set up and/or alter storage tables as required

The SHRTABLE is built and placed on SHRTABLE chain if named system not already in use. The SWPTABLE is updated with saved keys.

7 Set virtual PSW, GPRS, and FPRS

Diag. 7B3.2. Virtual Tracing



From DMKCFMBK-TRACE Command

INPUT

TRACE XXX

VMBLOK

VMTRCTL

VMTREXT

DMKTRACE – Virtual Tracing

Pick up operands and options and check for validity

If 'OFF' specified, turn off flags (A)

If 'END' specified, call DMKTRCPB to restore any instructions altered by TRACE, turn off flags and return TREXT block to free storage

Otherwise,
Issue 'TRACE STARTED' message
Get trace control block and set VMBLOK pointer to it, if a trace control block does not exist
Set trace flags (B)
Call DMKTRCIT to initialize branch or full instruction tracing, if specified.

Return to DMKCFM

OUTPUT

VMBLOK

VMTREXT

VMTRCTL

TREXT

TREXCTL1

TREXCTL2

TREXTERM

TREXPRNT

TREXRUNF

equal

Entry via SVC 8 (C)

Put trace, prefix and type in output line
Convert binary and addresses to hexadecimal (DMDCVT)
Get mnemonic for OP code, if applicable (DMKNEM)
Write trace line to output device

If ATTN was hit or if halt after trace line was specified call DMKCFMBK to enter console function mode

Otherwise

Return to Caller

COMMENTS

(A) If this turns off the last flag, then the TREXT block is returned to free storage. If branch and instruction tracking are both turned off, call DMKTRCPB to restore any instructions altered by TRACE.

(B) VMTRCTL and TREXCTL 1 are identical

(C) Entry via SVC 8 as follows:

	Entry Point	*From*
External Interrupt	DMKTRCEX	DMKDSP
I/O Interrupt	DMKTRCIO	DMKDSP
Program Interrupt	DMKTRCPG	DMKPRG
Privileged Instructions	DMKTRCPV	DMKPRV
I/O Operations	DMKTRCSI	DMKVIOEX
Virtual and Real CSWs	DMKTRCSW	DMKVIOIN
SVC, branch or full instruction trace	DMKTRCSV	DMKPSA
Restore user instructions altered by tracing	DMKTRCPB	DMKTRA
Initialize instruction tracing	DMKTRCIT	DMKTRA

From DMKCFMBK ADSTOP Command

INPUT

ADSTOP Address

DMKCFDAD

If 'OFF' specified, restore instruction and free work buffer

Otherwise,
Get work buffer
Set VMBLOK pointer
Save instruction and it virtual address
Replace instruction with SVC B3

Return to DMKCFM

OUTPUT

VMBLOK

VMADSTOP

ADSTBLOK

ADSTINAD

Virtual Storage

0AB3

DISCONNECTING A USER: A user may permanently or temporarily disconnect himself from the system by a console command, or he may be forcibly disconnected by the operator or the system. In any case, the routines that handle the termination process are in the pageable module DMKUSO.

Permanent Disconnect: The user may voluntarily exit from the system via the LOGOFF (or LOGOUT) command. This command terminates all virtual machine operation, releases all storage occupied by control blocks and user virtual storage pages, and disconnects the teleprocessing line connection to the user's terminal. If the user specifies the HOLD option with LOGOFF, all of the above occurs, except the teleprocessing line remains enabled. This option is especially useful for dialed connections that will be reused immediately by another user.

The user can be forced off the system by the system operator via the FORCE command. This has the same effect as a user-initiated logoff, except that the user is informed that the operator has logged him off. A user may also be logged off the system:

- If the time for a read of a system password expires (28 seconds).

- If he makes a connection to the system but does not logon within a given period.

- If he is running disconnected (without an active terminal) and his virtual machine attempts a terminal read or enters a disabled wait state.

The LOGOFF command is processed by the DMKUSOLG and DMKUSOFF subroutines. DMKUSOFF is also called directly by DMKDSP to force the logoff of a disconnected user as previously described.

Temporary Disconnect: A user may temporarily disconnect his terminal from his virtual machine while allowing the virtual machine to continue to run via the DISCONN command. This command flags the virtual machine as being disconnected and releases the user's terminal and teleprocessing line. If the HOLD option was specified in the DISCONN command, CP allows the line to remain enabled, and another user can use the terminal to LOGON. The disconnected virtual machine continues to be dispatched until it either attempts to execute a terminal read to the disconnected console or it enters a disabled wait state. At this time, the dispatcher (DMKDSP) calls the routine DMKUSOFF directly to force the machine out of the system. While the machine is disconnected from its virtual console (real terminal) any terminal output is lost; in addition, CP may apply a disconnected penalty to the machines scheduling priority, in order to bias the system in favor of interactive users.

A user may also be disconnected by the system operator. If the disconnected user logs on to the system while his disconnected machine is still running, he is reconnected and can continue to interact with the system in the usual manner.

| The DISCONN command is processed by the DMKUSO subroutine.

RECOVERY MANAGEMENT SUPPORT (RMS)

The Machine Check Handler (MCH) minimizes the lost computing time due to machine malfunction. MCH does this by attempting to correct the malfunction immediately, and by producing machine check records and messages to assist the service representatives in determining the cause of the problem.

The Channel Check Handler (CCH) aids the Input/Output Supervisor (DMKIOS) in the recovery from channel errors. CCH provides the device dependent Error Recovery Programs (ERPs) with the information needed to retry a channel operation which has failed.

This support is standard and model independent on the external level (from the user's point of view there are no considerations, at system generation time, for model dependencies).

SYSTEM INITIALIZATION FOR RMS

DMKIOEFL is called by DMKCPI to initialize the error recording at cold start and warm start time. DMKIOEFL will give control to DMKIOG to initialize the MCH area. A store CPU ID (STIDP) instruction is performed to determine if VM/370 is running in a virtual machine environment, or running standalone on the real machine. If VM/370 is running in a virtual machine the version code will be set to a hexadecimal 'FF' by DMKPRV. If the version code returned is hexadecimal 'FF,' the RMS functions will not be initialized beyond putting the wait bit on in the machine check new PSW (virtual). The logic of this is that machine check interrupts and channel errors (other than Channel Data Checks) will not be reflected to any virtual machine. VM/370 running on the real machine will make the determination as to whether the virtual machine should be terminated.

If the version code is not X'FF,' DMKIOG determines what channels are on line by performing a Store Channel ID (STIDC) instruction and saves the channel type for each channel on line. The maximum machine check extended logout length (MCEL) indicated by the Store CPU ID (STIDP) instruction is added to the length of the MCH record header, fixed logout length and damage assessment data field. DMKIOG will then call DMKFRE to obtain the necessary storage to be allocated for the MCH record area and the CP executing block (CPEXBLOK). DMKIOG saves the pointers for the Machine Check Record and the CPEXBLOK in DMKMCH. DMKIOG obtains the storage for the I/O extended logout area and initializes the logout area and the ECSW to ones. The I/O extended logout pointer is saved at location 172 and control register 15 is initialized with the address of the extended logout area. The length of the CCH record and the online channel types are saved in DMKCCH. It should be noted that the ability of a CPU to produce an extended logout or I/O extended logout and the length of the logouts are both model and channel dependent. If VM/370 is being initialized on a Model 165 II or 168, the 2860, 2870, and 2880 standalone channel modules are loaded and locked by the paging supervisor and the pointers are saved in DMKCCH. If VM/370 is being initialized on any other model, the integrated channel support is assumed; this support is part of the Channel Control Subroutine of DMKCCH. Before returning to DMKIOE the MCH/CCH recording cylinder for error recording is initialized. DMKIOE passes control back to DMKCPI and control register 14 is initialized with the proper mask to record machine checks.

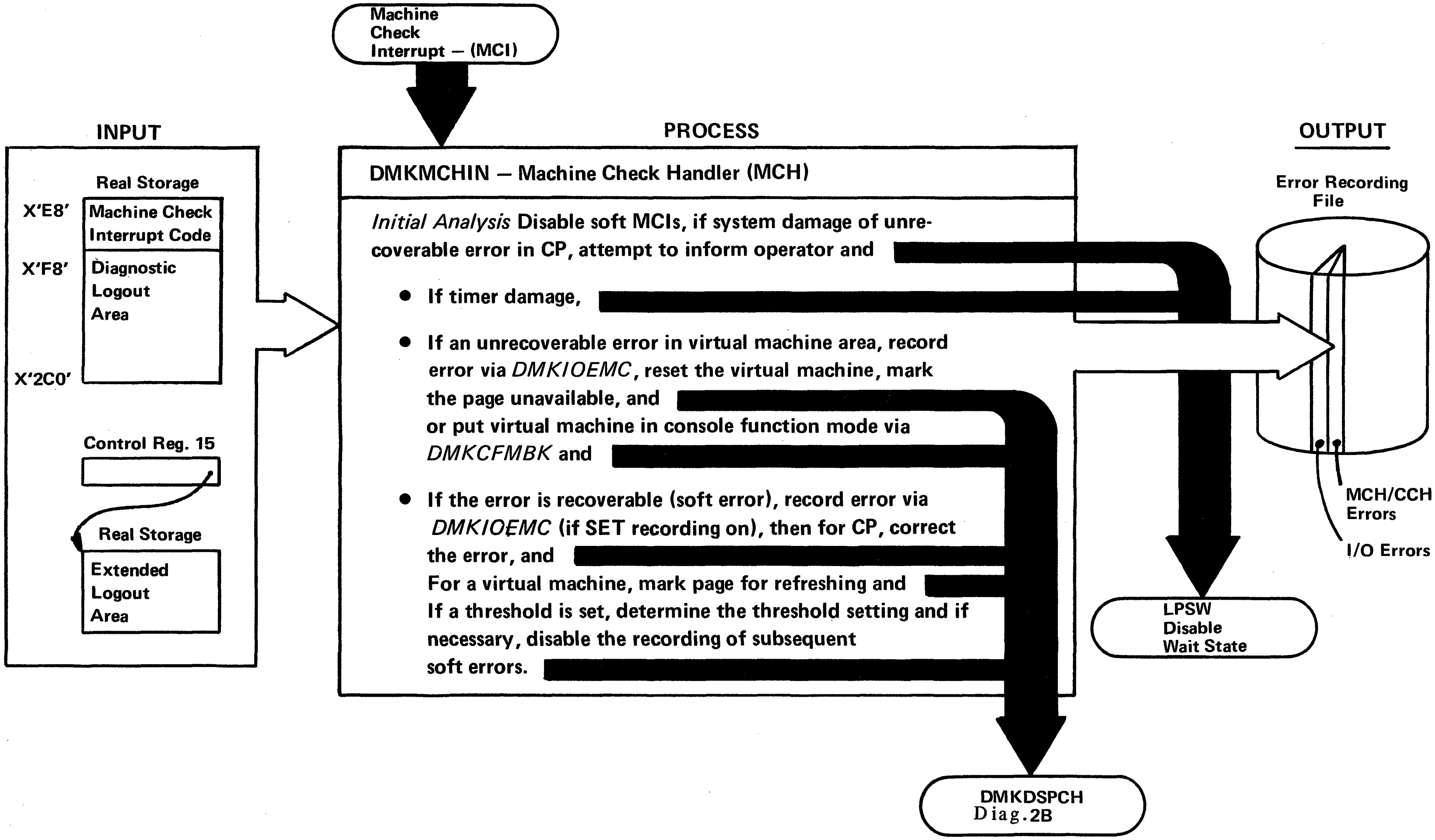
OVERVIEW OF MACHINE CHECK HANDLER

A machine malfunction can originate from the CPU, real storage or control storage. When any of these fails to work properly, an attempt to correct the malfunction is made by the CPU.

Whenever the malfunction is corrected, the Machine Check Handler (MCH) is notified by a machine check interrupt and the CPU logs out fields of information in real storage, detailing the cause and nature of the error. The model independent data is stored in the fixed logout area and the model dependent data is stored in the extended logout area. The Machine Check Handler uses these fields to analyze the error, format an error record, and write the record out on the error recording cylinder of SYSRES.

If the machine fails to recover from the malfunction through its own recovery facilities, the Machine Check Handler is notified by a machine check interrupt and an

Diag. 8B1. Machine Check Handler (MCH)



DMKMCHIN – Machine Check Handler (MCH)

Initial Analysis Disable soft MCIs, if system damage of unrecoverable error in CP, attempt to inform operator and

- If timer damage,
- If an unrecoverable error in virtual machine area, record error via *DMKIOEMC*, reset the virtual machine, mark the page unavailable, and or put virtual machine in console function mode via *DMKCFMBK* and
- If the error is recoverable (soft error), record error via *DMKIOEMC* (if SET recording on), then for CP, correct the error, and For a virtual machine, mark page for refreshing and If a threshold is set, determine the threshold setting and if necessary, disable the recording of subsequent soft errors.

interruption code, noting that the recovery attempt was unsuccessful, is inserted in the fixed logout area. The Machine Check Handler then analyzes the data and attempts to keep the system as fully operational as possible.

Recovery from machine malfunctions can be divided into four categories: functional recovery, system recovery, system-supported restart and system repair. These levels of error recovery are discussed in their order of acceptability, functional recovery being most acceptable and system repair being least acceptable:

FUNCTIONAL RECOVERY: Functional recovery is recovery from a machine check without adverse effect on the system or the interrupted user. This type of recovery can be made by the CPU Retry, the ECC facility, or the Machine Check Handler. The CPU Retry and ECC error correcting facilities are discussed separately in this section since they are significant in the total error recovery scheme. Functional recovery by MCH is made by correcting Storage Protect Feature (SPF) Keys and intermittent errors in real storage.

SYSTEM RECOVERY: System recovery is attempted when functional recovery is impossible. System recovery is the continuation of system operations at the expense of the interruped user, who is terminated. System recovery can only take place if the user in question is not critical to continued system operation. An error in a system routine which is considered to be critical to system operation precludes functional recovery and would require a system-supported restart.

SYSTEM-SUPPORTED RESTART: When the machine check occurs in a critical routine, the primary system operator is notified that the system cannot continue to operate. An automatic reload of the system occurs. This type of recovery is tried when functional and system recovery have failed or could not be tried.

SYSTEM REPAIR: System repair is recovery that requires the services of maintenance personnel and takes place at the discretion of the operator. Usually, the operator has tried to recover by system-supported restart one or more times with no success. An example of this type of error is when a hard error occurs so frequently that system-supported restart is not successful.

SYSTEM/370 RECOVERY FEATURES

The operation of the Machine Check Handler depends on certain automatic recovery actions taken by the hardware and on logout information given to it by the hardware.

CPU Retry

CPU errors are automatically retried by microprogram routines. These routines save source data before it is altered by the operation. When the error is detected, a microprogram returns the CPU to the beginning of the operation, or to a point where the operation was executing correctly, and the operation is repeated. After several unsuccessful retries, the error is considered permanent.

ECC Validity Checking

ECC checks the validity of data from real and control storage, automatically correcting single-bit errors. It also detects multiple-bit errors but does not correct them. Data enters and leaves storage through a storage adapter unit. This unit checks each double word for correct parity in each byte. If a single-bit error is detected, it is corrected. The corrected double word is then sent back into real or control storage and on to the CPU. When a multiple-bit error is detected, a machine check interruption occurs, and the error location is placed in the fixed logout area. MCH gains control and attempts to recover from the error.

Control Registers

Two control registers are used by MCH for loading and storing control information (see Figure 8). Control register 14 contains mask bits which specify whether certain conditions can cause machine check interruptions and mask bits which control conditions under which an extended logout can occur. Control register 15 contains the address of the extended logout area.

Word	Bits	Name of Field	
14	0	Check-Stop Control	Mch-Chk Handling
14	1	Synch. MCEL Ctrl.	Mch-Chk Handling
14	2	I/O Extended Logout Ctrl.	Chan-Chk Handling
14	4	Recovery Report Mask	Mch-Chk Handling
14	5	Degradation Report Mask	Mch-Chk Handling
14	6	External Damage Report Mask	Mch-Chk Handling
14	7	Warning Mask	Mch-Chk Handling
14	8	Asynch. MCEL Control	Mch-Chk Handling
14	9	Asynch. Fixed Log Ctrl.	Mch-Chk Handling
15	8-28	MCEL Address	Mch-Chk Handling

Figure 8. Control Register Assignments for RMS.

Machine Check Handler Subroutines

VM/370 Machine Check Handler (DMKMCH) consists of the following functions:

1. Initial Analysis Subroutine
2. Main Storage Analysis Subroutine
3. SPF Analysis Subroutine
4. Recovery Facility Mode Switching
5. Operator Communication Subroutine
6. Virtual User Termination Subroutine
7. Soft Recording Subroutine
8. Buffer Error Subroutine
9. Term Subroutine

INITIAL ANALYSIS SUBROUTINE: The Initial Analysis Subroutine of DMKMCH receives control via a machine check interruption. To minimize the possibility of losing logout information by recursive machine check interrupts, the machine check new PSW gives control to DMKMCH with the system disabled for further interruptions. There is always a danger that a machine malfunction may occur immediately after DMKMCH is entered and the system is disabled for interruption. Disabling all interruptions is only a temporary measure to give the Initial Analysis Subroutine time to make the following emergency provisions:

1. It disables for soft machine check interruptions. Soft recording will not be enabled until the error is recorded.

2. It saves the contents of the fixed and extended logout areas in the machine check record.

3. It alters the machine check new PSW to point to the Term Subroutine. The Term Subroutine is designed to handle second machine check errors.

4. It enables for hard machine check interrupts.

5. If a virtual user was running when the interrupt occurred, the running status (GPRs, FPRs, PSW, M.C. old PSW, CRs, etc.) is saved in the user's VMBLOK.

6. It initially examines the machine check data for the following types of errors:

   MCIC=ZERO
   PSW invalid
   System damage
   Timing facilities damage

   The occurrence of any of these errors is considered uncorrectable by DMKMCH; the primary system operator is informed, the error is formatted and recorded, and the system is shutdown followed by an automatic restart function.

7. If the instruction processing damage bit is on, it tests for the following types of malfunctions:

   - Multiple-Bit Error in Main Storage -- Control is given to the Main Storage Analysis Subroutine.

   - SPF Key Error -- Control is given to the SPF Analysis Subroutine.

   - Retry failed -- If the CPU was in supervisor state the error is considered uncorrectable and the VM/370 system is terminated. If the CPU was in problem state, the virtual user is reset or terminated and the system continues operation.

8. If the CPU Retry or ECC was successful on a soft error, control is given to the Soft Recording Subroutine to format the record, write it out on the error recording cylinder, and to update the count of soft error occurrences.

9. If external damage was reported, control is given to the Soft Recording Subroutine to format the record and write it out on the error recording cylinder.

<u>MAIN STORAGE ANALYSIS SUBROUTINE</u>: The Main Storage Analysis Subroutine is given control when it is determined that the machine check interrupt was caused by a multiple-bit storage error. An initial function is performed to point the machine check new PSW to an internal subroutine to indicate a solid machine check, in the event of a machine check interrupt while exercising main storage.

Damaged storage areas associated with any portion of the CP nucleus itself cannot be refreshed; multiple-bit storage errors in CP cause the VM/370 system to be terminated. An automatic restart will reinitialize VM/370.

If the damage is not in the CP nucleus, main storage is exercised to determine if the failure is solid or intermittent. If the failure is considered solid, the 4K page frame is marked unavailable for use by the system. If the failure is considered intermittent, the page frame is marked invalid. The change bits associated with the damaged page frame are checked to determine if the page had been altered by the virtual machine. If no alteration had occurred, VM/370 assigns a new page frame to the virtual machine and a backup copy of the page is brought into storage the next time the page is referenced. If the page had been altered VM/370 resets or terminates the virtual machine, clears its virtual storage, and sends an appropriate message to the user. Normal system operation continues for all other users.

<u>STORAGE PROTECT FEATURE (SPF) ANALYSIS SUBROUTINE</u>: The SPF Analysis Subroutine is given control when it is determined that the machine check interrupt was caused by an SPF error. An initial function is performed to point the machine check new PSW to an internal subroutine in the event of a machine check interrrupt during testing and validation. The SPF Analysis routine then determines if the error was associated with a failure in the virtual user storage or in the storage associated with CP itself.

An SPF error associated with VM/370 is a potentially catastrophic failure. Namely, VM/370 always runs with a PSW key of zero, which means that the SPF Key in memory is not checked for an out of parity condition. The SPF Analysis Subroutine exercises all sixteen keys in the failing storage 2K page frame. If an SPF machine check occurs in exercising the sixteen keys five times each, then the error is considered solid and the operating system is terminated with a system shutdown. The system is automatically restarted and the VM/370 is reinitialized. If an SPF machine check does not occur, the machine check is considered intermittent. The zero key is restored to the failing storage 2K page frame and this is done transparent to the virtual users.

If an SPF machine check occurs which is associated with a virtual user, the SPF Analysis subroutine exercises all sixteen keys in the failing storage 2K page frame. If an SPF machine check does not occur, then the machine check is considered intermittent and the SWPTABLE for the page associated with the failing storage address is located. The storage key for the failing 2K storage page frame is retrieved from the

SWPTABLE and the change and reference bits are masked on in the storage key. The storage key is then stored into the affected failing storage 2K page frame. If an SPF machine check occurs in exercising the sixteen keys five times each, then the machine check is considered solid and the following actions are taken. (1) The user is selectively reset or terminated by the virtual user termination subroutine. (2) The 4K page frame associated with the failing address is removed as an available system resource. This is accomplished by locating the CORTABLE for the defective page and altering the CORFPNT and CORPBPNT pointers to make the page unavailable to the system. The CORDISA bit in this CORTABLE is set on to identify the reason for the status of this page in a system dump.

<u>RECOVERY FACILITY MODE SWITCHING</u>: The Recovery Facility Mode Switching subroutine (DMKMCHMS) allows the service representative to change the mode that CPU retry and ECC recording are operating in. This subroutine receives control when a user with privilege class 'F' issues some form of the SET MODE command. A check is initially made to determine if this is VM/370 running under VM/370. If this is the case, the request is ignored and control is returned to the calling routine. The format of the MODE command is as follows:

```
   SET MODE {RETRY|MAIN} {QUIET|RECORD}
```

RETRY and MAIN imply CPU retry and main storage respectively.

QUIET causes the specified facility to be placed in quiet mode. RECORD causes the count of soft errors to be reset to zero and the specified facility to be placed in record mode.

<u>OPERATOR COMMUNICATION SUBROUTINE</u>: The Operator Communciation subroutine is invoked when the integrity of the system has degraded to a point where automatic shutdown and reload of the system has been tried and was unsuccessful, or could not be attempted due to the severity of the hardware failure. A check is first made to determine if the system operator is logged on as a user, next a check is made to determine if the system operator is disconnected. If either of these checks is not affirmative a message cannot be issued directly to the system operator. A LPSW is performed to place the CPU in a disabled wait state with a recognizable wait state code in the CPU Instruction Counter.

<u>VIRTUAL USER TERMINATION SUBROUTINE</u>: The virtual user termination subroutine is used to selectively reset or terminate a virtual user whose operation has been interrupted by an uncorrectable machine check. First, the user is marked non-dispatchable to prevent the damaged user from running before reset or termination is performed. The machine check record is formatted and DMKIOEMC is called to record the error. Then the user is notified by a call to DMKQCNWT that a machine check has occurred and that his operation will be terminated. The primary system operator is notified of the virtual user termination via a message issued by a call to DMKQCNWT. If the user is running in the virtual=real area, DMKUSO is called to log the user off the system and to return the storage previously allocated to the user and to clear any outstanding user I/O Requests. The hold option of LOGOFF is invoked to allow a user on a dial facility to retain the connection and thus permit LOGON without re-establishing the line connection. However, if the user is running in the virtual area, and DMKCFM is then called to put the virtual user in console function mode, the user must re-initialize the system to commence operation.

<u>SOFT RECORDING SUBROUTINE</u>: The soft recording subroutine performs <u>two</u> basic functions:

1. Format a machine check record and call DMKIOEMC to record the error on the error recording cylinder.

2. Maintain the threshold for CPU RETRY and ECC errors and switch from recording to quiet mode when the threshold value is exceeded. In order to accomplish this, a counter is maintained by DMKMCH for successful CPU retry and corrected ECC events.

<u>CPU Retry Recording Mode</u>: Recording mode (bit 4 of Control Register 14 set to one) is the initialized state, and normal operating state of VM/370 for CPU Retry errors. Recording mode may also be entered by use of the CP SET command. When 12 soft machine checks

have occurred the soft recording subroutine switches the CPU from recording mode to quiet mode. For the purpose of model-independent implementation this is accomplished by setting bit 4 of Control Register 14 to zero. Since in QUIET mode no soft machine check interruptions occur, a switch from quiet mode to recording mode can be made by issuing the SET MODE RETRY|MAIN RECORD command. While in recording mode corrected CPU RETRY|MAIN reports are formatted and recorded on the VM/370 error recording cylinder, but the primary systems operator is not informed of these incidents.

CPU Retry Quiet Mode: Quiet mode (bit 4 of Control Register 14 set to 0) can be entered in one of two ways: (1) when 12 soft machine checks have occurred, or (2) when the SET MODE RETRY QUIET command is executed by a class 'F' user. In this mode, both CPU retry and ECC reporting are disabled. The CPU will remain in quiet mode until the next system IPL (warmstart or cold start) occurs or a SET MODE RETRY|MAIN RECORD command is executed by a class 'F' user.

ECC Reporting Modes: To achieve model independent support, RMS does not set a specific mode for ECC reporting. The mode in which ECC reporting is initialized depends upon the hardware design for each specific CPU model type. For the IBM System/370 Models 135, 145, 158, and 168 the hardware initialized state (therefore the normal operational state for VM/370) is QUIET mode. For the IBM System/370 Models 155 II, and 165 II the hardware initialized state (therefore the normal operational state for VM/370) is RECORD mode. An automatic restart incident due to a VM/370 failure does not RESET the ECC reporting mode in effect at the time of failure.

The change from RECORD to QUIET mode for ECC reporting can be initiated in either of the following ways; (1) by issuing the SET MODE {MAIN|RETRY} QUIET command, or (2) automatically whenever 12 soft machine checks have occurred. For the purpose of model independent implementation this will be accomplished by setting bit 4 of Control Register 14 to zero.

The change from QUIET to RECORD mode for ECC reporting can be accomplished by use of the SET MODE MAIN RECORD command. This recording mode option is for use by maintenance personnel only. It should be noted that CPU RETRY is placed in recording mode if it is not in that state when the SET MODE MAIN RECORD command is issued.

While in RECORDING mode, corrected ECC reports are formatted and recorded on the error recording cylinder, but the primary systems operator is not informed of these incidents.

BUFFER ERROR SUBROUTINE: On CPU models equipped with a high speed buffer (155 II, 158, 165 II, 168) or a Data Look Aside Table (DLAT) (165 II, 168) the deletion of buffer blocks due to hardware failure is reported via a DEGRADATION REPORT machine check interrupt. MCH enables itself for degradation report machine check interrupts at system initialization by setting bit 5 of Control Register 14 to 1. If a machine check interrupt occurs which indicates high speed buffer or DLAT damage, MCH formats the record and calls DMKIOEMC to record it on the error recording cylinder, informs the primary systems operator of the failure, and returns control to the system to continue normal operation.

TERM SUBROUTINE: The Term Subroutine is given control in the event of a hard machine check interrupt while DMKMCH is in the process of handling a machine check interrupt. Note that soft error reporting is disabled for the entire time that MCH is processing an error.

An analysis is performed of the machine check interrupt code of the first error to determine if it was a soft error, and if it was, the first error is recorded, the system status is restored and control is restored to the point where the first error occurred. If the first error was a hard error, the Operator Communication Subroutine is given control to issue a message directly to the system operator, and to terminate CP operation.

OVERVIEW OF CHANNEL CHECK HANDLER

The Channel Check Handler (CCH) aids the I/O Supervisor in recovering from channel errors and informs the operator or service representative of the occurrence of channel errors.

CCH receives control from the I/O Supervisor when a channel data check, channel control check, or interface control check occurs. CCH produces an I/O Error Block (IOERBLOK) for the error recovery program and a record to be written on the error recording cylinder for the system operator or service representative. The operator or service representative may obtain a copy of the record by using the CPEREP programs. A message about the channel error is issued each time a record is written on the error recording cylinder.

When the Input/Output Supervisor program detects a channel error during routine status examination following an SIO, TIO, HIO, or an I/O interruption it passes control to the Channel Check Handler (DMKCCH). DMKCCH analyzes the channel logout information and constructs an IOERBLOK, if the error is a channel control or interface control check, and an ECSW will be constructed and placed in the IOERBLOK. The IOERBLOK provides information for the device dependent error recovery procedures. DMKCCH also constructs a record to be recorded on the error recording cylinder. Normally, CMKCCH returns control to the I/O Supervisor after constructing an IOERBLOK and a record. However, if DMKCCH determines that system integrity has been damaged (system reset or invalid unit address, etc.) then CP operation will be terminated. The action taken by DMKCCH for CP termination will be to issue a message directly to the system operator and place the CPU in a disabled wait state with a recognizable wait code in the CPU instruction counter.

Recovery will not be initiated for channel errors associated with I/O events inititated by a virtual user, however these will cause termination of the user after he has been notified of the failure. The error will be recorded by DMKIOECC on the error recording cylinder.

Normally, when DMKCCH returns control to the I/O supervisor, the error recovery program for the device which experienced the error is scheduled. When the ERP receives control, it prepares to retry the operation if analysis of the IOERBLOK indicates that retry is possible. Depending on the device type and error condition, the ERP will either effect recovery or mark the event fatal and return control to the I/O Supervisor. The I/O Supervisor will call the recording routine DMKIOE to record the channel error.

The primary system operator will be notified of the failure, and DMKIOE will return control to the system and normal processing will continue.

Channel Control Subroutine

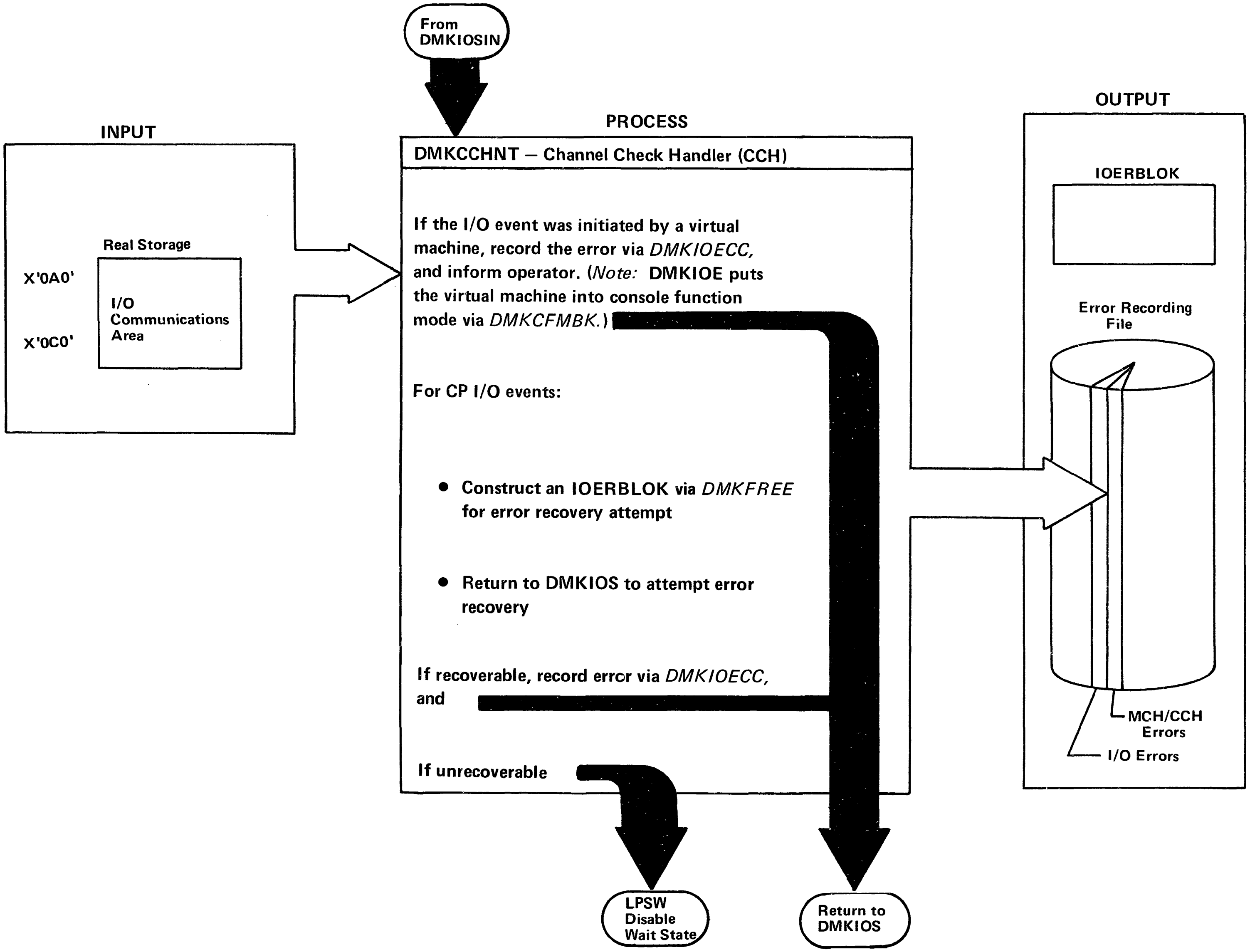
Control is passed to the Channel Control Subroutine of DMKCCH after a SIO with failing status stored, or an I/O interrupt due to a channel control check, interface control check, or channel data check.

If "logout pending" is indicated in the CSW, the CP termination flag is set. The existence of real device blocks (RCHBLOK, RCUBLOK, RDEVBLOK), for the failing device address, is determined by a call to DMKSCNRU and an indicator is set if they do exist. An indicator is also set if the IOBLOK for the failing device address exists. A call to DMKFREE obtains storage space for the channel check record and the channel control subroutine builds the record. If the indicators show that the real device blocks and the IOBLOK exist, a call to DMKFREE obtains storage space and the channel control subroutine builds the I/O error block (IOERBLOK); if these blocks do not exist, the IOERBLOK is not built. The IOERBLOK is used for two purposes:

1. The device dependent Error Recording Program (ERP) uses the IOERBLOK to attempt recovery on CP initiated I/O events. If the I/O events that resulted in a channel check are associated with a virtual user, the I/O fatal flag is set in the IOBLOK and the user's virtual machine is reset, cleared, and put into console function mode with a read up on the line. The length and address of the channel check record is placed in the IOERBLOK and the IOERBLOK is chained off the IOBLOK.

2. DMKIOECC uses the IOERBLOK to record the channel check record on the error recording cylinder.

Diag. 8B2. Channel Check Handler (CCH)



DMKCCHNT – Channel Check Handler (CCH)

If the I/O event was initiated by a virtual machine, record the error via *DMKIOECC,* and inform operator. (*Note:* DMKIOE puts the virtual machine into console function mode via *DMKCFMBK.*)

For CP I/O events:

- Construct an IOERBLOK via *DMKFREE* for error recovery attempt
- Return to DMKIOS to attempt error recovery

If recoverable, record errcr via *DMKIOECC,* and

If unrecoverable

The channel control subroutine gives control to a channel dependent error analysis routine to build or save the extended channel status word (ECSW). When the Channel Control Subroutine regains control, eight active addresses are saved in the channel check record.

If the CP termination flag is set, the I/O extended logout data from the channel check record is restored to main storage for use by SEREP. If the system operator is both logged on as a user and connected to the system, a message (DMKCCH603W) is sent to him advising him of the channel error. A LPSW is then executed to place the CPU in a disabled wait state with a wait state code of 002 in the CPU instruction counter.

If the CP termination flag is not set, a check is made to determine if an IOERBLOK was built by the channel control subroutine.

If an IOERBLOK was not built, DMKIOECC is called to record the channel check record on the error recording cylinder. The system operator is then sent a message (DMKCCH601I or DMKCCH602I) informing him of the error and control is then returned to DMKIOS to continue system operation.

If an IOERBLOK was built, control is returned to DMKIOS which calls the appropriate ERP. Whether or not recovery is successful, DMKIOS eventually calls DMKIOE to record the channel check record. DMKIOE examines the status of the error CSW in the IOERBLOK to determine is it was a channel error; if so, it finds the length and pointer to the channel check record and records the error on the error recording cylinder. If this was not a channel error, DMKIOE continues normal processing.

Individual Routines

A separate channel error analysis routine is provided for each type of channel for which DMKCCH can be used. The purpose of these routines and the Channel Control Subroutine is to analyze the channel logout to determine the extent of damage and to create a sequence and termination code to be placed in the ECSW in the IOERBLOK. At system initialization time the correct model dependent channel recovery routine is loaded and the storage necessary to support the routine is allocated. The model dependent error analysis subroutines and routines and their functions are as follows:

<u>INTEGRATED CHANNELS (Models 135, 145, 155 II, 158)</u>: Since all of these systems have integrated channels one common subroutine is used to handle all of these CPU types. This subroutine:

- Indicates CP termination if the ECSW is not complete, the channel has been reset, or reset codes are invalid
- Moves the ECSW to the IOERBLOK
- Moves the hardware stored unit address and the I/O extended logout to the channel check record
- Sets the I/O extended logout area and ECSW area to ones
- Returns control to the Channel Control Subroutine

<u>2860 CHANNEL</u> (<u>Models 165 II</u>, <u>168</u>): The 2860 logout area is checked to determine if a complete logout exists; if not, CP termination is necessary.

A check is made in the logout area for validity of the CSW fields and bits are set in the channel check record's ECSW field to indicate bad fields.

The channel logout is then checked and sequence codes are set based on the presence of a channel control check, or an interface control check. If a channel control check is present, the codes set are determined through parity. The count determines if parity is good and sets a resultant condition code.

The logout area is examined to ensure that the unit address has valid parity and is the same address passed by DMKIOS. If so, the "unit address valid" bit in the ECSW is set. If the unit address is not valid the "unit address valid" bit is reset to indicate the invalid condition.

The ECSW field in the channel check record is moved to the IOERBLOK, if one exists.

After completing the ECSW the 2680 routine moves the 2860 I/O extended logout into the channel check record, set the I/O extended logout area to ones, and returns to the Channel Control Subroutine.

2870 CHANNEL (Models 165 II, 168): If the channel failed to logout completely, at least part of the logout area is all ones. If a full word of ones is found, a CP termination condition exists.

A check is made in the logout area for valid CSW fields, and bits are set in the channel check record's ECSW field to indicate bad fields.

The termination and sequence codes are set depending on the presence of an interface control check or channel control check. If a channel control check is present, the codes set are determined through parity, count, and/or data transfer checks. For the 2870, parity can be determined directly from the channel logout.

The logout area is also examined to ensure valid parity in the unit address and to ensure that the address is the same as that passed to DMKCCH by DMKIOS. If so, the "unit address valid" bit in the ECSW is set.

The 3rd word of the logout area is also analyzed for type II errors. If one of these type II errors is found, a CP termination condition exists.

The ECSW field in the channel check record is moved to the IOERBLOK, if one exists.

Before returning to the Channel Control Subroutine, the 2870 routine moves the 2870 I/O extended logout into the channel check record and sets the I/O extended logout area to ones.

2880 CHANNEL (Models 165 II and 168): This routine will analyze 9 words of the 28 word logout.

The 2880 Analysis routine handles channel data checks, interface control checks, and channel control checks.

Termination code 3 (system reset) is not set in the ECSW because the 2880 channel does not issue system reset to the devices. Retry codes of zero to five are possible.

Note: There are several catastrophic conditions under which the CP termination flag can be set, in the 2880 analysis routine. They are:

1. The channel did not complete the logout.
2. The CSW is not reliable.
3. The unit address in the I/O interrupt device address field is not correct.

Only a channel check record is needed if the channel has recognized an internal error and has recovered from it without any damage. No recovery action is necessary in these cases.

If the channel address in the I/O interrupt device address field does not match the channel address in the logout, a CP termination condition exists.

If the channel was doing a scan and the unit control word had a parity check a CP termination condition exists. If there was no parity check, there was no damage during the scan and only a channel check record is required.

Depending on the sequence the channel has entered, the termination and sequence codes are set; command address, unit address, and unit status validity is determined; and the sequence code is set valid. The ECSW field in the channel check record is moved into the IOERBLOK, if one exists.

Before returning to the Channel Control Subroutine, the 2880 routine will move the I/O extended logout into the channel check record and set the I/O extended logout area to ones.

ERROR RECORDING AND RECOVERY

The error recording facility is made up of three modules. One module (DMKIOE) is resident and the other two (DMKIOF and DMKIOG) are pageable.

The error recording routines records: unit checks, machine checks, channel checks, and hardware environmental counter·sense data on the error recording cylinders of the system resident device in a format suitable for subsequent processing by the CPEREP program. The recorder also initializes the error recording cylinders at IPL time if they are in an unrecognizable format.

When the recorder is entered from DMKIOS, it is entered at DMKIOERR. This entry is used for unit checks and channel data checks. A test is made of the failing CSW (located in the IOERBLOK) to see if the error was a channel error. If it was, control is passed to routine for recording channel checks.

The IOERBLOK sense data, IOBLOK flags, and VMBLOK user class are examined to determine if the error should be recorded. See the section "Errors Recorded" for those that are recorded.

Writing the Record

After an error record is formatted, it is added to the error recording cylinder using DMKRPAGT and DMKRPAPT. The error recording cylinders have page sized records (4096 bytes). Each page contains a header (8 bytes) which signifies cylinder and page number of the page (4 bytes), next available space for recording within page (2 bytes), a page in-use indicator (1 byte), and a flag byte. Each record within the page is recorded with a 4-byte length prefix.

If an error record is too large to be added into a page, a new page is retrieved, updated with record, and placed back on the error recording cylinder with the paging routines.

Two cylinders are used for error recording: one cylinder is used exclusively for recording the I/O errors and the other cylinder for recording MCH/CCH errors. The cylinders that are used for error recording are specified by the user at system generation time. If either error recording cylinder becomes 90 per cent full, a message is issued to the operator using DMKQCNWT to warn him of the condition. If either cylinder becomes full, another message is issued to inform the operator and recording is stopped on that cylinder. Recording continues on the cylinder that is not full.

If a channel check error is to be recorded, the recorder is entered at DMKIOERR or DMKIOECC. The channel check handler determines the entry. A channel check error record is formatted.

A machine check enters at DMKIOEMC. Pointers are passed from the machine check handler in registers 6 and 7 to locate a buffer where the machine check record and length are saved. A machine check error record is recorded with the saved machine check logout and additional information. The machine check error record is written onto the error recording cylinder by using the paging routines.

Hardware environmental counter records are formed using routine DMKIOEEV. This routine is scheduled by DMKIOS after control is returned from the ERP. Sense data information is stored in the IOERBLOK by the ERP. The record formed is called a nonstandard record.

Errors Recorded

In addition to recording environmental data, the following types of errors are recorded for DASD virtual machines that are not Class F.

- Bus-out check
- Overrun check
- Seek check
- Track overrun check
- Missing address marker check
- Equipment check
- Permanent data check
- Unrecoverable error for a control program initiated channel program.

The following 3420 errors are recorded if the user is not Class F:

- Bus-out check
- Overrun check
- C compare check
- Write TRIG VRC check
- Feed thru check
- Vel/RESTART check
- Velocity change check
- Equipment check

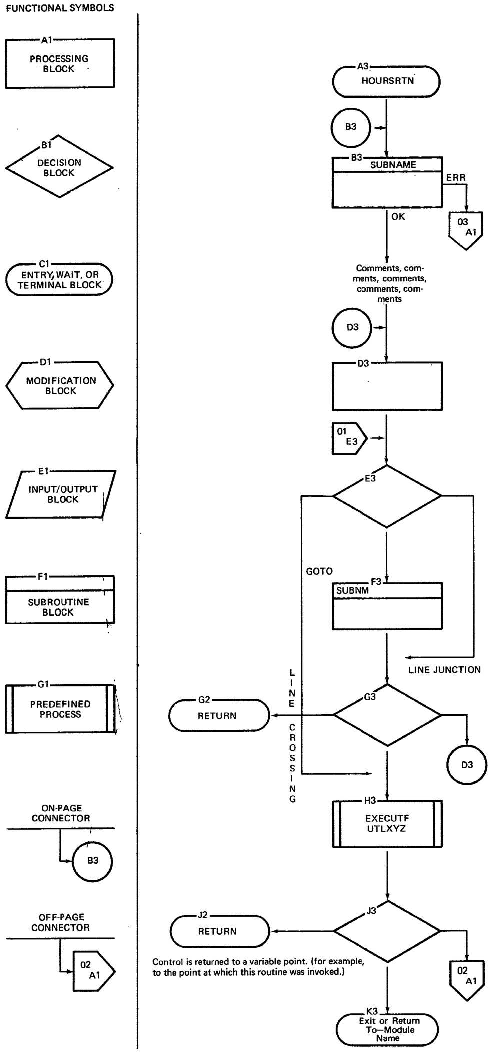
CLEAR and FORMAT Recording Area: DMKIOEFM is called by the CPEREP program via a DIAGNOSE instruction. DMKIOEFM is invoked to reset the specified error recording cylinders (if CLEARALL, CLEARIO, or CLEARMC was specified). The clear is performed by resetting each page-header space-available field. A pointer in storage is then updated to point to the first page on the error recording cylinder available for recording MCH and CCH records and the first page available on the other error recording cylinder for recording outboard errors. Control is then returned to the calling routine.

Finding First Recording Cylinder at IPL Time: DMKIOEFL is called by DMKCPI to find the first available page that can be used for error recording. The paging routines, DMKRPAPT and DMKRPAGT, are used to read the error recording cylinder's pages (4096 byte records). As each page record is read it is examined to see if this record is the last recorded. If so, a pointer in storage is saved so recording can continue on that page record. Control is then returned to the caller. If either error recording cylinder is in an unrecognizable format, that cylinder is automatically reformatted by CP.

PROGRAM ORGANIZATION

This section contains the flowcharts for all processing modules. The modules are in alphabetical order. To determine the pertinent information about a module, see the Directory entries DMKACO to DMKWRM.

FLOWCHARTS



The terminal block is used to show entry and exit points of a routine or subroutine.Block A3 shows an entry point named HOURSRTN.

This instruction at this location calls a subroutine called SUBNAME. Upon return from the subroutine, if an error was detected (ERR), then processing resumes at the block A1, part 3 of this series of charts. If an error was not detected (OK), then normal processing resumes.

Comments are included to provide additional information about the logic being displayed.

On-page entry connector one or more branches to this block appear on this part of the flowchart.

Off-page entry connector. A branch to this block appears on another part(s) of this flowchart.

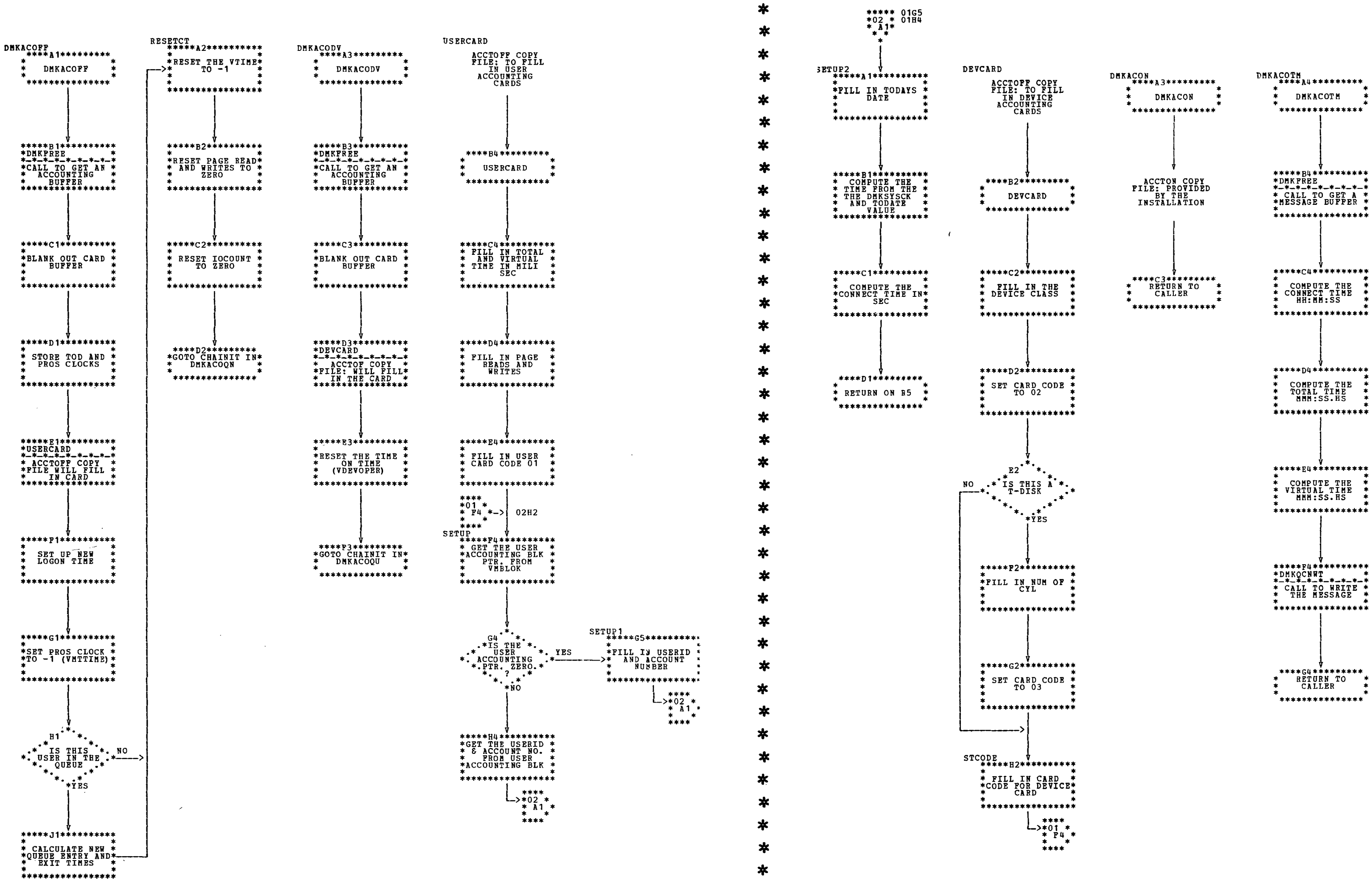
The instruction at location GOTO calls either a subroutine within this module or another module that is used like a subroutine. If the name in the subroutine block begins with DMK, then the call is to another module (see the "Module/Entry Point Directory" to determine the flowchart location of the called module). If the name does not begin with DMK, then the subroutine is in the flowchart series for this module (see the "Subroutine Directory" to determine the flowchart-part location of the called subroutine).

One-page exit connector control branches to block D3 on this part of the flowchart.

This block refers to a routine or program that is documented in some other publication.

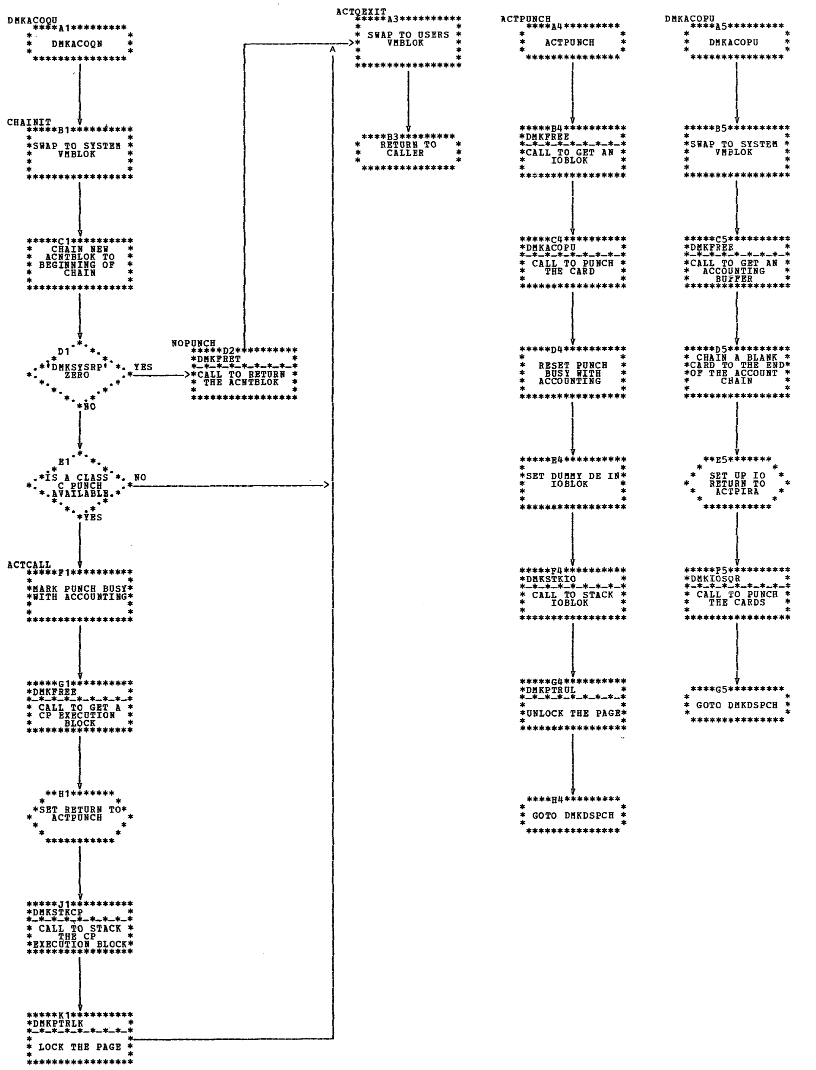
Off-page exit connector control branches to block A1 on part 2 of this series of flowchart.

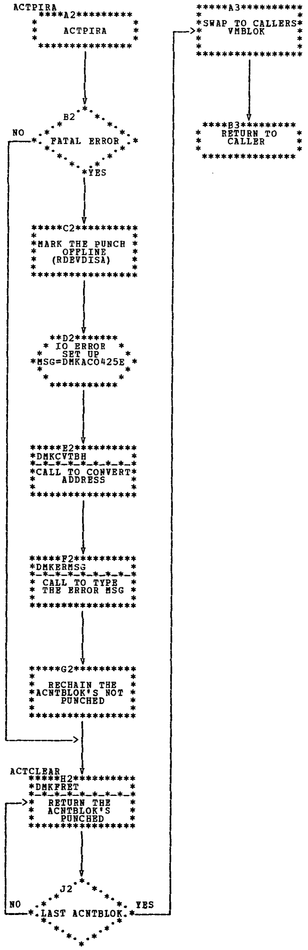
Control branches to an entry point on another flowchart.

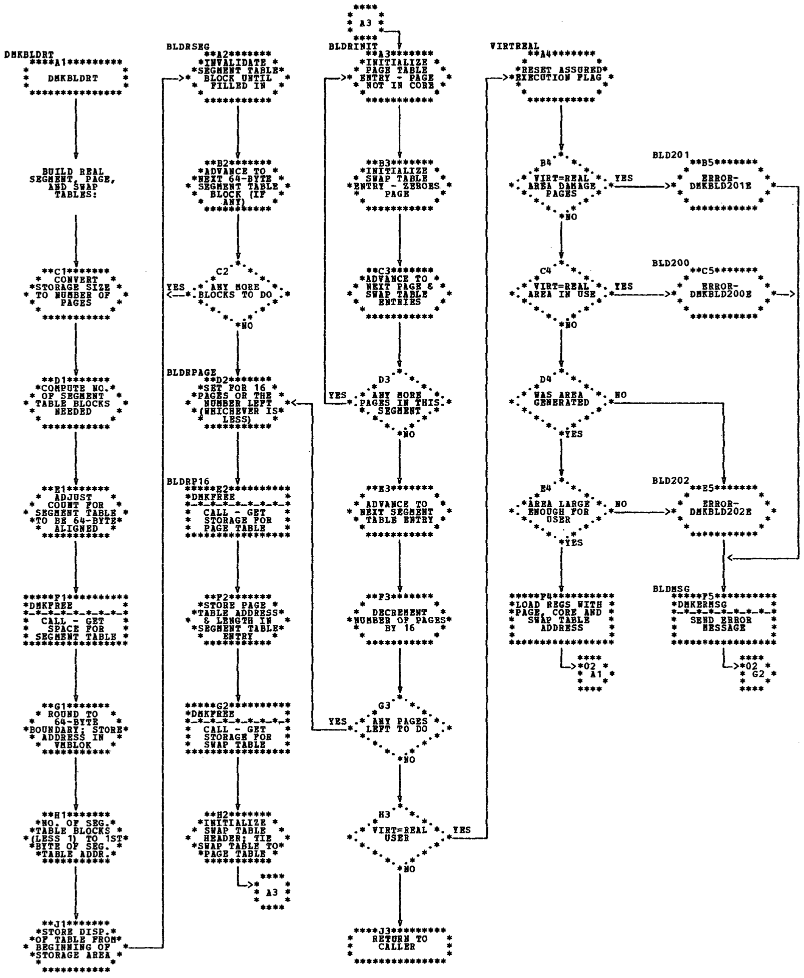

DMKACOFF
A1
DMKACOFF
B1
DMKFREE
CALL TO GET AN ACCOUNTING BUFFER
C1
BLANK OUT CARD BUFFER
D1
STORE TOD AND PROS CLOCKS
E1
USERCARD
ACCTOFF COPY FILE WILL FILL IN CARD
F1
SET UP NEW LOGON TIME
G1
SET PROS CLOCK TO -1 (VMTTIME)
H1
IS THIS USER IN THE QUEUE
NO
YES
J1
CALCULATE NEW QUEUE ENTRY AND EXIT TIMES
RESETCT
A2
RESET THE VTIME TO -1
B2
RESET PAGE READ AND WRITES TO ZERO
C2
RESET IOCOUNT TO ZERO
D2
GOTO CHAINIT IN DMKACOQN
DMKACODV
A3
DMKACODV
B3
DMKFREE
CALL TO GET AN ACCOUNTING BUFFER
C3
BLANK OUT CARD BUFFER
D3
DEVCARD
ACCTOF COPY FILE: WILL FILL IN THE CARD
E3
RESET THE TIME ON TIME (VDEVOPER)
F3
GOTO CHAINIT IN DMKACOQU
USERCARD
ACCTOFF COPY FILE: TO FILL IN USER ACCOUNTING CARDS
B4
USERCARD
C4
FILL IN TOTAL AND VIRTUAL TIME IN MILI SEC
D4
FILL IN PAGE READS AND WRITES
E4
FILL IN USER CARD CODE 01
01 F4
02H2
SETUP
F4
GET THE USER ACCOUNTING BLK PTR. FROM VMBLOK
G4
IS THE USER ACCOUNTING PTR. ZERO. ?
YES
NO
SETUP1
G5
FILL IN USERID AND ACCOUNT NUMBER
02 A1
H4
GET THE USERID & ACCOUNT NO. FROM USER ACCOUNTING BLK
02 A1
02 A1
01G5
01H4
SETUP2
A1
FILL IN TODAYS DATE
B1
COMPUTE THE TIME FROM THE THE DMKSYSCK AND TODATE VALUE
C1
COMPUTE THE CONNECT TIME IN SEC
D1
RETURN ON R5
DEVCARD
ACCTOFF COPY FILE: TO FILL IN DEVICE ACCOUNTING CARDS
B2
DEVCARD
C2
FILL IN THE DEVICE CLASS
D2
SET CARD CODE TO 02
E2
IS THIS A T-DISK
NO
YES
F2
FILL IN NUM OF CYL
G2
SET CARD CODE TO 03
STCODE
H2
FILL IN CARD CODE FOR DEVICE CARD
01 F4
DMKACON
A3
DMKACON
ACCTON COPY FILE: PROVIDED BY THE INSTALLATION
C3
RETURN TO CALLER
DMKACOTM
A4
DMKACOTM
B4
DMKFREE
CALL TO GET A MESSAGE BUFFER
C4
COMPUTE THE CONNECT TIME HH:MM:SS
D4
COMPUTE THE TOTAL TIME MMM:SS.HS
E4
COMPUTE THE VIRTUAL TIME MMM:SS.HS
F4
DMKQCNWT
CALL TO WRITE THE MESSAGE
G4
RETURN TO CALLER

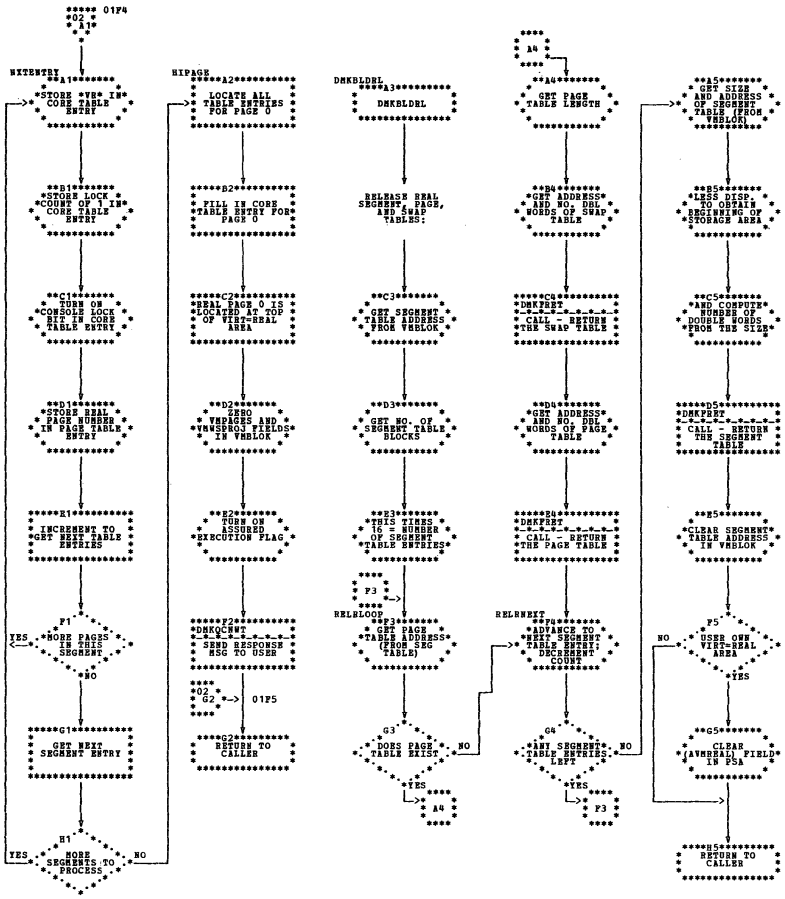
| DMKACO -- Accounting Routines (Parts 1 and 2 of 4)

| DMKACO -- Accounting Routines (Parts 3 and 4 of 4)



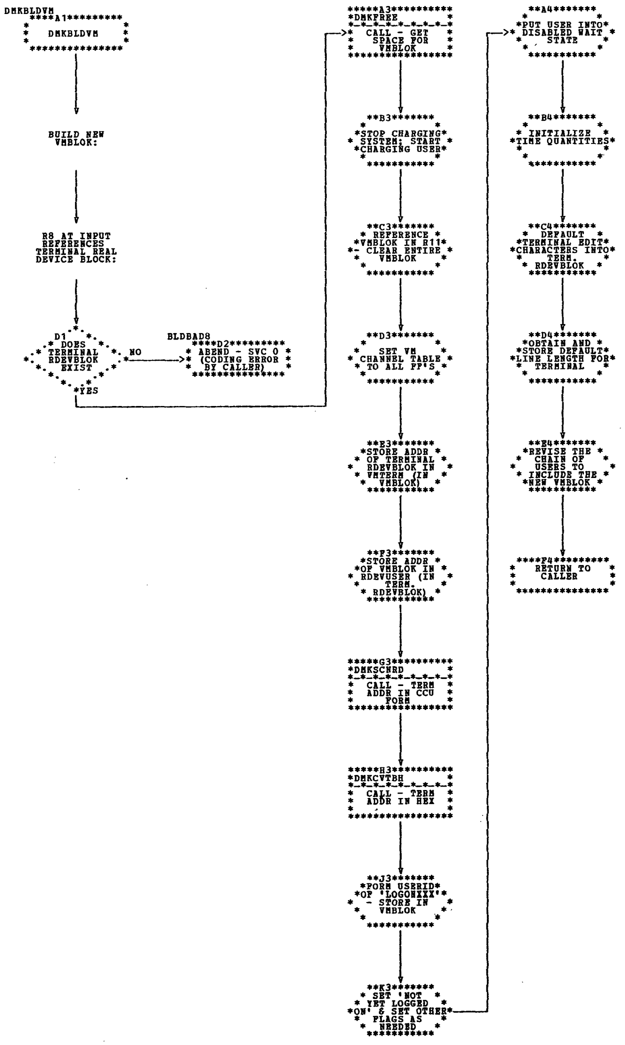


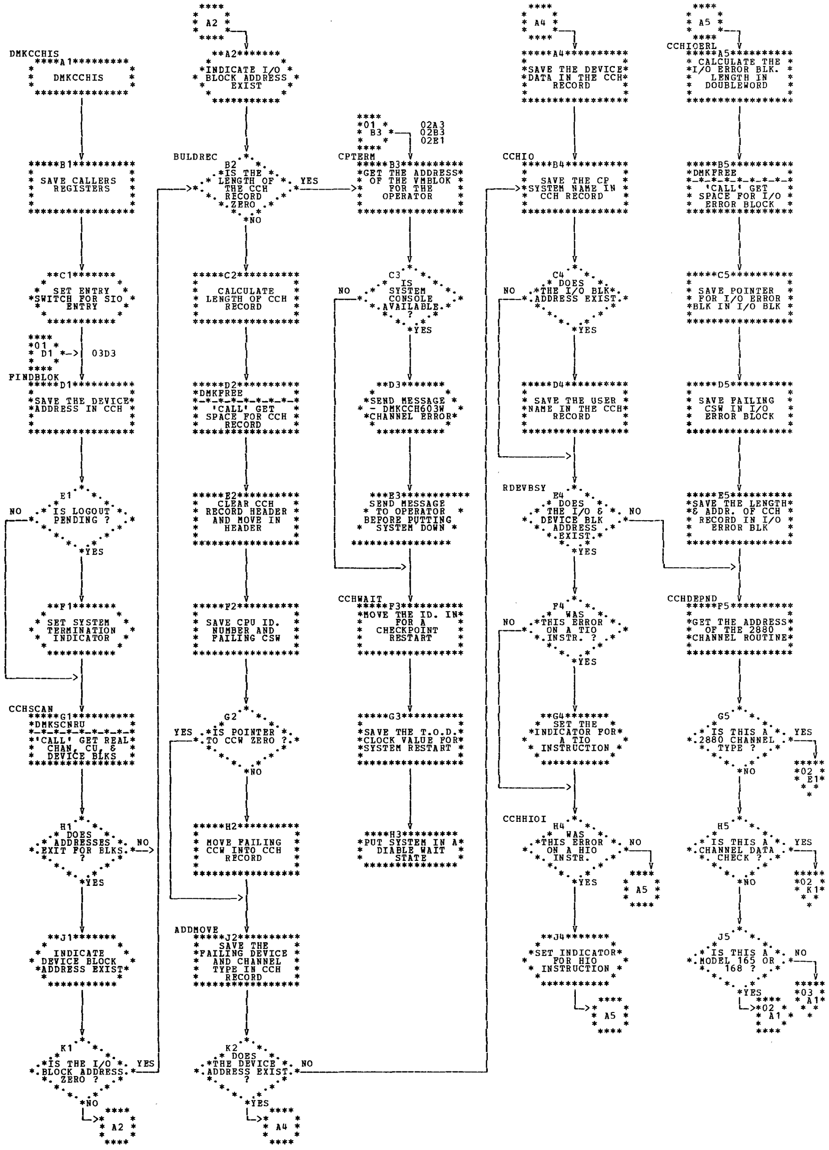

DMKBLDRT
A1 DMKBLDRT
BUILD REAL SEGMENT, PAGE, AND SWAP TABLES:
C1 CONVERT STORAGE SIZE TO NUMBER OF PAGES
D1 COMPUTE NO. OF SEGMENT TABLE BLOCKS NEEDED
E1 ADJUST COUNT FOR SEGMENT TABLE TO BE 64-BYTE ALIGNED
F1 DMKFREE CALL - GET SPACE FOR SEGMENT TABLE
G1 ROUND TO 64-BYTE BOUNDARY; STORE ADDRESS IN VMBLOK
H1 NO. OF SEG. TABLE BLOCKS (LESS 1) TO 1ST BYTE OF SEG. TABLE ADDR.
J1 STORE DISP. OF TABLE FROM BEGINNING OF STORAGE AREA
BLDRSEG
A2 INVALIDATE SEGMENT TABLE BLOCK UNTIL FILLED IN
B2 ADVANCE TO NEXT 64-BYTE SEGMENT TABLE BLOCK (IF ANY)
C2 ANY MORE BLOCKS TO DO
YES
NO
BLDRPAGE
D2 SET FOR 16 PAGES OR THE NUMBER LEFT (WHICHEVER IS LESS)
BLDRP16
E2 DMKFREE CALL - GET STORAGE FOR PAGE TABLE
F2 STORE PAGE TABLE ADDRESS & LENGTH IN SEGMENT TABLE ENTRY
G2 DMKFREE CALL - GET STORAGE FOR SWAP TABLE
H2 INITIALIZE SWAP TABLE HEADER; TIE SWAP TABLE TO PAGE TABLE
A3
BLDRINIT
A3 INITIALIZE PAGE TABLE ENTRY - PAGE NOT IN CORE
B3 INITIALIZE SWAP TABLE ENTRY - ZEROES PAGE
C3 ADVANCE TO NEXT PAGE & SWAP TABLE ENTRIES
D3 ANY MORE PAGES IN THIS SEGMENT
YES
NO
E3 ADVANCE TO NEXT SEGMENT TABLE ENTRY
F3 DECREMENT NUMBER OF PAGES BY 16
G3 ANY PAGES LEFT TO DO
YES
NO
H3 VIRT=REAL USER
YES
NO
J3 RETURN TO CALLER
VIRTREAL
A4 RESET ASSURED EXECUTION FLAG
B4 VIRT=REAL AREA DAMAGE PAGES
YES
NO
C4 VIRT=REAL AREA IN USE
YES
NO
D4 WAS AREA GENERATED
NO
YES
E4 AREA LARGE ENOUGH FOR USER
NO
YES
F4 LOAD REGS WITH PAGE, CORE AND SWAP TABLE ADDRESS
02 A1
BLD201
B5 ERROR-DMKBLD201E
BLD200
C5 ERROR-DMKBLD200E
BLD202
E5 ERROR-DMKBLD202E
BLDMSG
F5 DMKERMSG SEND ERROR MESSAGE
02 G2

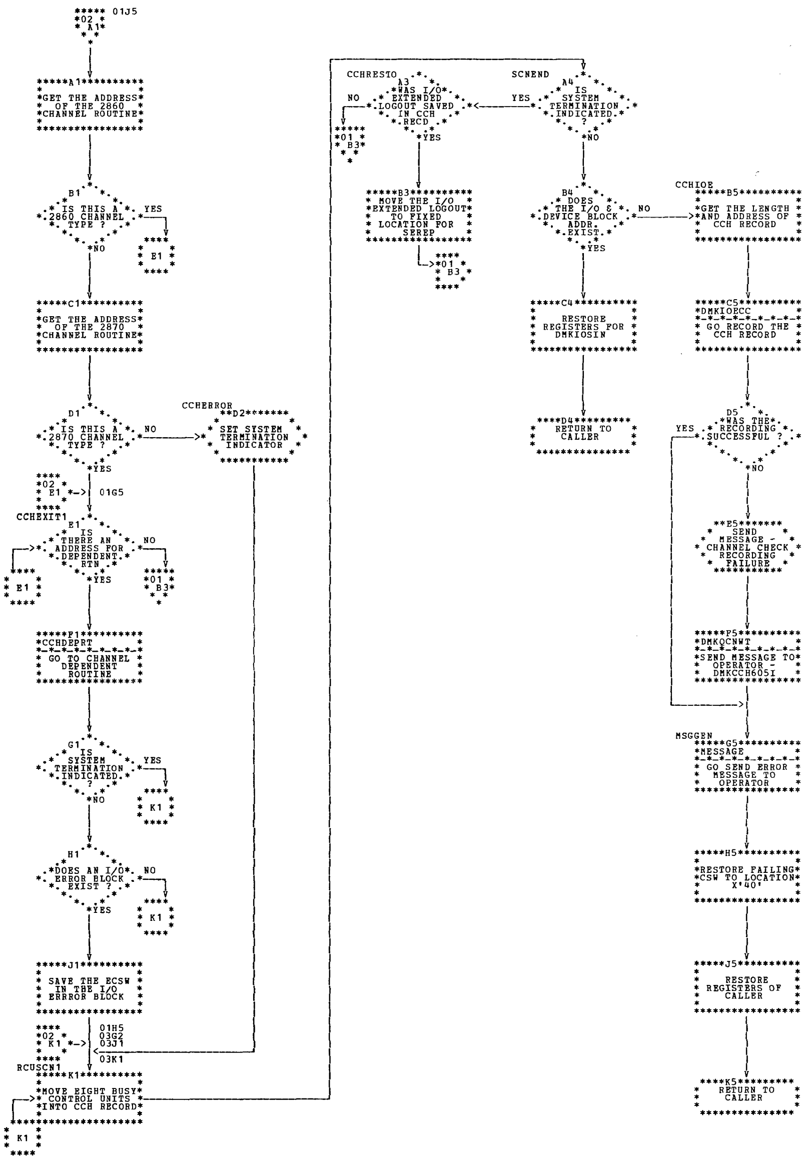

02 A1 01F4
NXTENTRY
A1 STORE *VR* IN CORE TABLE ENTRY
B1 STORE LOCK COUNT OF 1 IN CORE TABLE ENTRY
C1 TURN ON CONSOLE LOCK BIT IN CORE TABLE ENTRY
D1 STORE REAL PAGE NUMBER IN PAGE TABLE ENTRY
E1 INCREMENT TO GET NEXT TABLE ENTRIES
F1 MORE PAGES IN THIS SEGMENT
YES
NO
G1 GET NEXT SEGMENT ENTRY
H1 MORE SEGMENTS TO PROCESS
YES
NO
HIPAGE
A2 LOCATE ALL TABLE ENTRIES FOR PAGE 0
B2 FILL IN CORE TABLE ENTRY FOR PAGE 0
C2 REAL PAGE 0 IS LOCATED AT TOP OF VIRT=REAL AREA
D2 ZERO VMPAGES AND VMWSPROJ FIELDS IN VMBLOK
E2 TURN ON ASSURED EXECUTION FLAG
F2 DMKQCNWT SEND RESPONSE MSG TO USER
02 G2 01F5
G2 RETURN TO CALLER
DMKBLDRL
A3 DMKBLDRL
RELEASE REAL SEGMENT, PAGE, AND SWAP TABLES:
C3 GET SEGMENT TABLE ADDRESS FROM VMBLOK
D3 GET NO. OF SEGMENT TABLE BLOCKS
E3 THIS TIMES 16 = NUMBER OF SEGMENT TABLE ENTRIES
F3
RELRLOOP
F3 GET PAGE TABLE ADDRESS (FROM SEG TABLE)
G3 DOES PAGE TABLE EXIST
NO
YES
A4
A4
A4 GET PAGE TABLE LENGTH
B4 GET ADDRESS AND NO. DBL WORDS OF SWAP TABLE
C4 DMKFRET CALL - RETURN THE SWAP TABLE
D4 GET ADDRESS AND NO. DBL WORDS OF PAGE TABLE
E4 DMKFRET CALL - RETURN THE PAGE TABLE
RELRNEXT
F4 ADVANCE TO NEXT SEGMENT TABLE ENTRY; DECREMENT COUNT
G4 ANY SEGMENT TABLE ENTRIES LEFT
NO
YES
F3
A5 GET SIZE AND ADDRESS OF SEGMENT TABLE (FROM VMBLOK)
B5 LESS DISP. TO OBTAIN BEGINNING OF STORAGE AREA
C5 AND COMPUTE NUMBER OF DOUBLE WORDS FROM THE SIZE
D5 DMKFRET CALL - RETURN THE SEGMENT TABLE
E5 CLEAR SEGMENT TABLE ADDRESS IN VMBLOK
F5 USER OWN VIRT=REAL AREA
NO
YES
G5 CLEAR (AVMREAL) FIELD IN PSA
H5 RETURN TO CALLER

DMKBLD -- Build/Release REAL Storage Tables; Build VMBLOK (Parts 1 and 2 of 3)

DMKELD -- Build/Release Real Storage Tables; Build VMBLOK (Part 3 of 3)

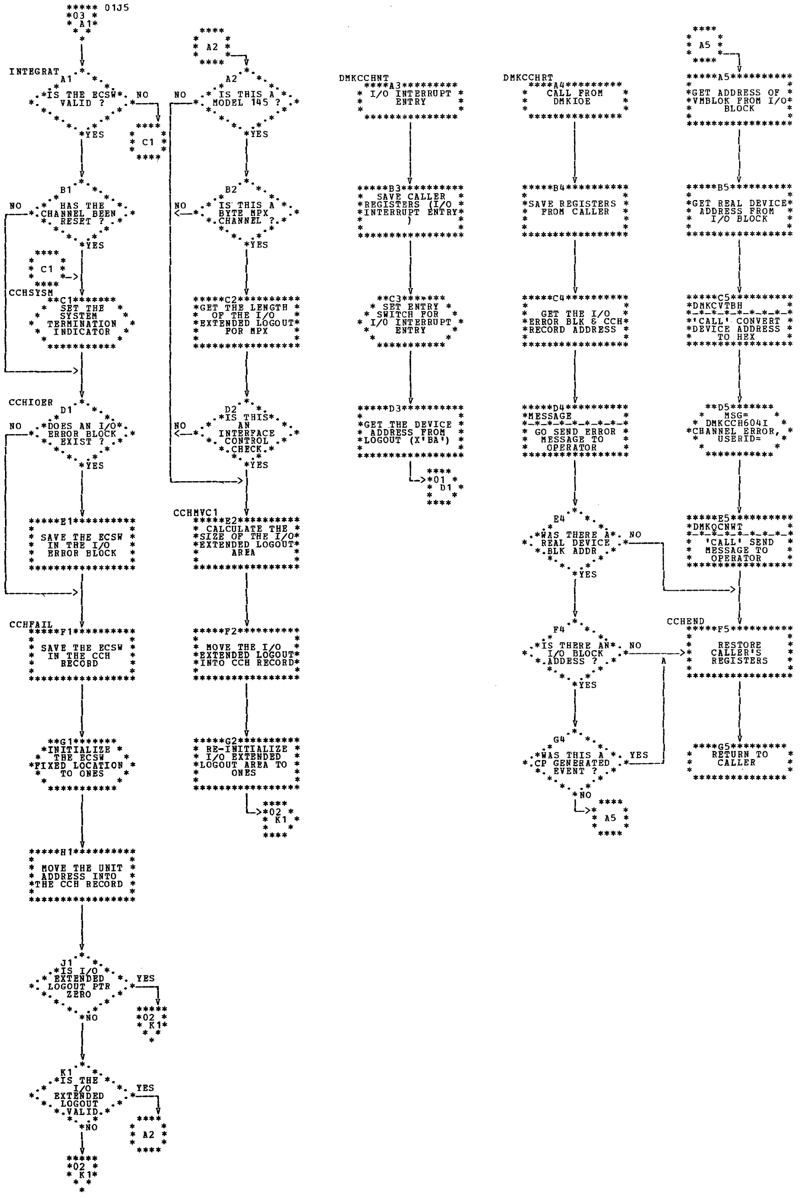

DMKBLDVM
A1
DMKBLDVM
BUILD NEW VMBLOK:
R8 AT INPUT REFERENCES TERMINAL REAL DEVICE BLOCK:
D1
DOES TERMINAL RDEVBLOK EXIST
NO
YES
BLDBAD8
D2
ABEND - SVC 0 (CODING ERROR BY CALLER)
A3
DMKFREE
CALL - GET SPACE FOR VMBLOK
B3
STOP CHARGING SYSTEM; START CHARGING USER
C3
REFERENCE VMBLOK IN R11 - CLEAR ENTIRE VMBLOK
D3
SET VM CHANNEL TABLE TO ALL FF'S
E3
STORE ADDR OF TERMINAL RDEVBLOK IN VMTERM (IN VMBLOK)
F3
STORE ADDR OF VMBLOK IN RDEVUSER (IN TERM. RDEVBLOK)
G3
DMKSCNRD
CALL - TERM ADDR IN CCU FORM
H3
DMKCVTBH
CALL - TERM ADDR IN HEX
J3
FORM USERID OF 'LOGONXXX' - STORE IN VMBLOK
K3
SET 'NOT YET LOGGED ON' & SET OTHER FLAGS AS NEEDED
A4
PUT USER INTO DISABLED WAIT STATE
B4
INITIALIZE TIME QUANTITIES
C4
DEFAULT TERMINAL EDIT CHARACTERS INTO TERM. RDEVBLOK
D4
OBTAIN AND STORE DEFAULT LINE LENGTH FOR TERMINAL
E4
REVISE THE CHAIN OF USERS TO INCLUDE THE NEW VMBLOK
F4
RETURN TO CALLER

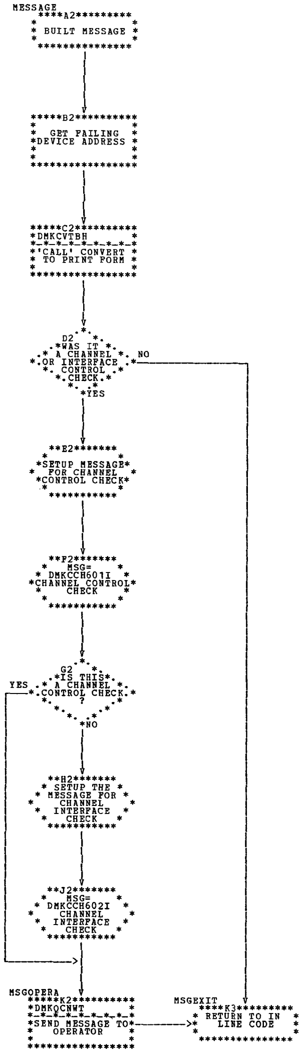

DMKCCHIS
A1 DMKCCHIS
B1 SAVE CALLERS REGISTERS
C1 SET ENTRY SWITCH FOR SIO ENTRY
01 D1 03D3
FINDBLOK
D1 SAVE THE DEVICE ADDRESS IN CCH
E1 IS LOGOUT PENDING ?
NO
YES
F1 SET SYSTEM TERMINATION INDICATOR
CCHSCAN
G1 DMKSCNRU 'CALL' GET REAL CHAN, CU, & DEVICE BLKS
H1 DOES ADDRESSES EXIT FOR BLKS ?
NO
YES
J1 INDICATE DEVICE BLOCK ADDRESS EXIST
K1 IS THE I/O BLOCK ADDRESS ZERO ?
YES
NO
A2
A2 INDICATE I/O BLOCK ADDRESS EXIST
BULDREC
B2 IS THE LENGTH OF THE CCH RECORD ZERO ?
YES
NO
C2 CALCULATE LENGTH OF CCH RECORD
D2 DMKFREE 'CALL' GET SPACE FOR CCH RECORD
E2 CLEAR CCH RECORD HEADER AND MOVE IN HEADER
F2 SAVE CPU ID. NUMBER AND FAILING CSW
G2 IS POINTER TO CCW ZERO ?
YES
NO
H2 MOVE FAILING CCW INTO CCH RECORD
ADDMOVE
J2 SAVE THE FAILING DEVICE AND CHANNEL TYPE IN CCH RECORD
K2 DOES THE DEVICE ADDRESS EXIST ?
NO
YES
A4
01 B3 02A3 02B3 02E1
CPTERM
B3 GET THE ADDRESS OF THE VMBLOK FOR THE OPERATOR
C3 IS SYSTEM CONSOLE AVAILABLE ?
NO
YES
D3 SEND MESSAGE - DMKCCH603W CHANNEL ERROR
E3 SEND MESSAGE TO OPERATOR BEFORE PUTTING SYSTEM DOWN
CCHWAIT
F3 MOVE THE ID. IN FOR A CHECKPOINT RESTART
G3 SAVE THE T.O.D. CLOCK VALUE FOR SYSTEM RESTART
H3 PUT SYSTEM IN A DIABLE WAIT STATE
A4
A4 SAVE THE DEVICE DATA IN THE CCH RECORD
CCHIO
B4 SAVE THE CP SYSTEM NAME IN CCH RECORD
C4 DOES THE I/O BLK ADDRESS EXIST ?
NO
YES
D4 SAVE THE USER NAME IN THE CCH RECORD
RDEVBSY
E4 DOES THE I/O & DEVICE BLK ADDRESS EXIST ?
NO
YES
F4 WAS THIS ERROR ON A TIO INSTR. ?
NO
YES
G4 SET THE INDICATOR FOR A TIO INSTRUCTION
CCHHIOI
H4 WAS THIS ERROR ON A HIO INSTR. ?
NO
YES
A5
J4 SET INDICATOR FOR HIO INSTRUCTION
A5
A5
CCHIOERL
A5 CALCULATE THE I/O ERROR BLK. LENGTH IN DOUBLEWORD
B5 DMKFREE 'CALL' GET SPACE FOR I/O ERROR BLOCK
C5 SAVE POINTER BLK IN I/O BLK
D5 SAVE FAILING CSW IN I/O ERROR BLOCK
E5 SAVE THE LENGTH & ADDR. OF CCH RECORD IN I/O ERROR BLK
CCHDEPND
F5 GET THE ADDRESS OF THE 2880 CHANNEL ROUTINE
G5 IS THIS A 2880 CHANNEL TYPE ?
YES
02 E1
NO
H5 IS THIS A CHANNEL DATA CHECK ?
YES
02 K1
NO
J5 IS THIS A MODEL 165 OR 168 ?
NO
03 A1
YES
02 A1

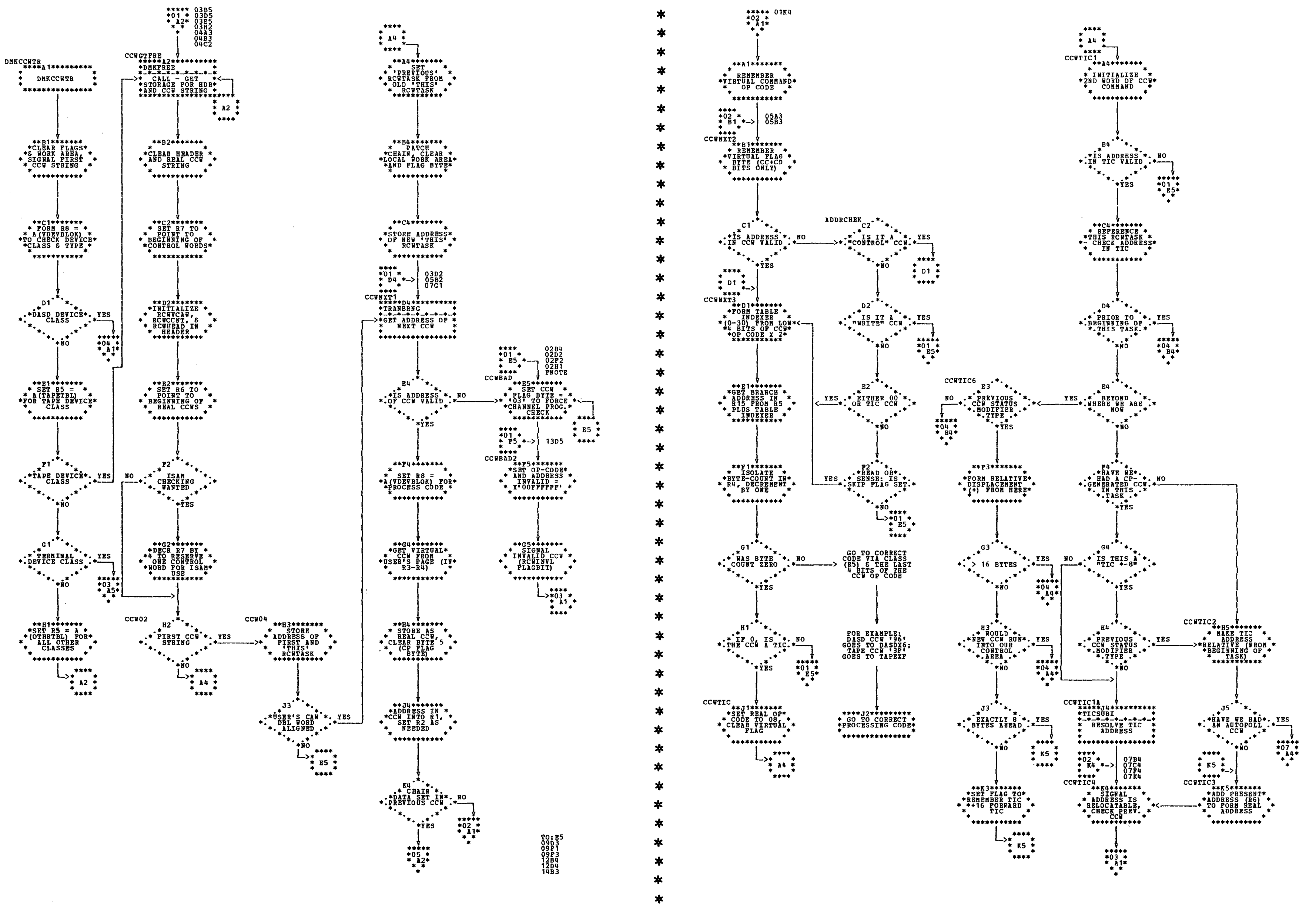

01J5
02 A1
A1 GET THE ADDRESS OF THE 2860 CHANNEL ROUTINE
B1 IS THIS A 2860 CHANNEL TYPE ?
YES
E1
NO
C1 GET THE ADDRESS OF THE 2870 CHANNEL ROUTINE
D1 IS THIS A 2870 CHANNEL TYPE ?
NO
YES
CCHERROR
D2 SET SYSTEM TERMINATION INDICATOR
02 E1 01G5
CCHEXIT1
E1 IS THERE AN ADDRESS FOR DEPENDENT RTN ?
NO
01 B3
YES
E1
F1 CCHDEPRT GO TO CHANNEL DEPENDENT ROUTINE
G1 IS SYSTEM TERMINATION INDICATED ?
YES
K1
NO
H1 DOES AN I/O ERROR BLOCK EXIST ?
NO
K1
YES
J1 SAVE THE ECSW IN THE I/O ERROR BLOCK
02 K1 01H5 03G2 03J1 03K1
RCUSCN1
K1 MOVE EIGHT BUSY CONTROL UNITS INTO CCH RECORD
K1
CCHRESTO
A3 WAS I/O EXTENDED LOGOUT SAVED IN CCH RECD
NO
01 B3
YES
B3 MOVE THE I/O EXTENDED LOGOUT TO FIXED LOCATION FOR SEREP
01 B3
SCNEND
A4 IS SYSTEM TERMINATION INDICATED ?
YES
NO
B4 DOES THE I/O & DEVICE BLOCK ADDR. EXIST ?
NO
YES
C4 RESTORE REGISTERS FOR DMKIOSIN
D4 RETURN TO CALLER
CCHIOE
B5 GET THE LENGTH AND ADDRESS OF CCH RECORD
C5 DMKIOECC GO RECORD THE CCH RECORD
D5 WAS THE RECORDING SUCCESSFUL ?
YES
NO
E5 SEND MESSAGE - CHANNEL CHECK RECORDING FAILURE
F5 DMKQCNWT SEND MESSAGE TO OPERATOR - DMKCCH605I
MSGGEN
G5 MESSAGE GO SEND ERROR MESSAGE TO OPERATOR
H5 RESTORE FAILING CSW TO LOCATION X'40'
J5 RESTORE REGISTERS OF CALLER
K5 RETURN TO CALLER

| DMKCCH -- Channel Check Handler (Parts 1 and 2 of 4)

| DMKCCH -- Channel Check Handler (Parts 3 and 4 of 4)

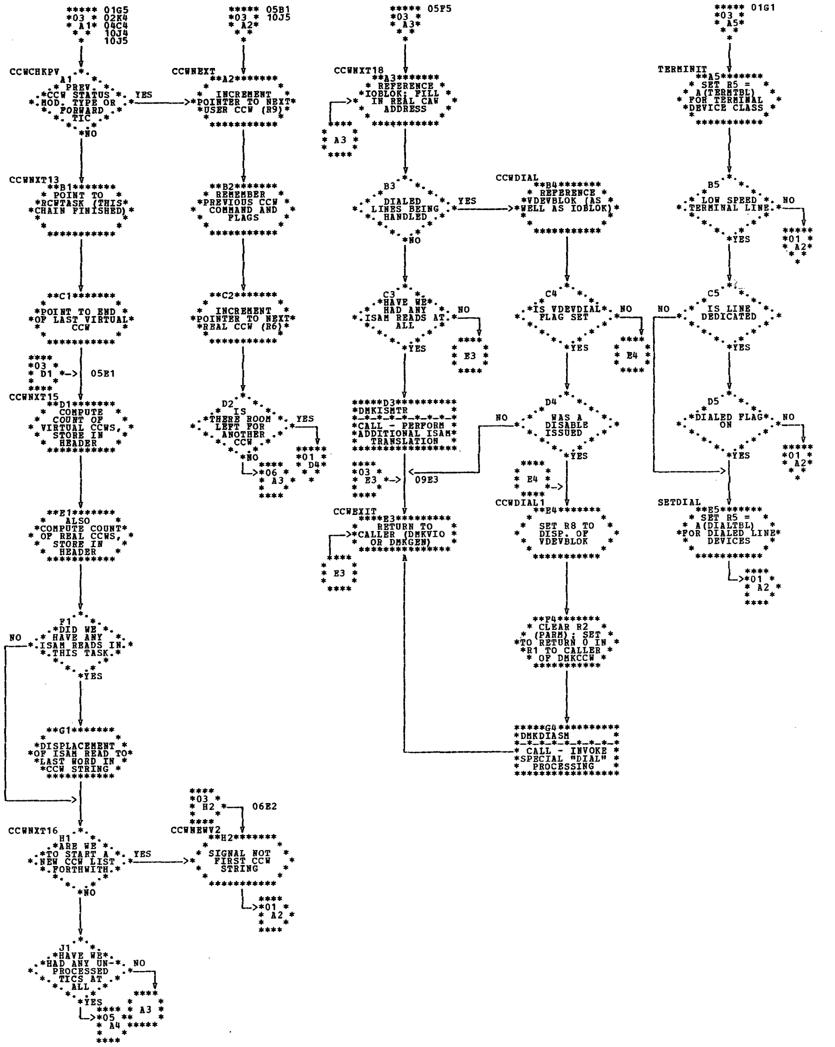

01J5
03 A1
INTEGRAT
A1
IS THE ECSW VALID ?
NO
YES
C1
B1
HAS THE CHANNEL BEEN RESET ?
NO
YES
C1
CCHSYSM
C1
SET THE SYSTEM TERMINATION INDICATOR
CCHIOER
D1
DOES AN I/O ERROR BLOCK EXIST ?
NO
YES
E1
SAVE THE ECSW IN THE I/O ERROR BLOCK
CCHFAIL
F1
SAVE THE ECSW IN THE CCH RECORD
G1
INITIALIZE THE ECSW FIXED LOCATION TO ONES
H1
MOVE THE UNIT ADDRESS INTO THE CCH RECORD
J1
IS I/O EXTENDED LOGOUT PTR ZERO
YES
02 K1
NO
K1
IS THE I/O EXTENDED LOGOUT VALID.
YES
A2
NO
02 K1
A2
A2
IS THIS A MODEL 145 ?
NO
YES
B2
IS THIS A BYTE MPX CHANNEL ?
NO
YES
C2
GET THE LENGTH OF THE I/O EXTENDED LOGOUT FOR MPX
D2
IS THIS AN INTERFACE CONTROL CHECK.
NO
YES
CCHMVC1
E2
CALCULATE THE SIZE OF THE I/O EXTENDED LOGOUT AREA
F2
MOVE THE I/O EXTENDED LOGOUT INTO CCH RECORD
G2
RE-INITIALIZE I/O EXTENDED LOGOUT AREA TO ONES
02 K1
DMKCCHNT
A3
I/O INTERRUPT ENTRY
B3
SAVE CALLER REGISTERS (I/O INTERRUPT ENTRY)
C3
SET ENTRY SWITCH FOR I/O INTERRUPT ENTRY
D3
GET THE DEVICE ADDRESS FROM LOGOUT (X'BA')
01 D1
DMKCCHRT
A4
CALL FROM DMKIOE
B4
SAVE REGISTERS FROM CALLER
C4
GET THE I/O ERROR BLK & CCH RECORD ADDRESS
D4
MESSAGE
GO SEND ERROR MESSAGE TO OPERATOR
E4
WAS THERE A REAL DEVICE BLK ADDR
NO
YES
F4
IS THERE AN I/O BLOCK ADDESS ?
NO
A
YES
G4
WAS THIS A CP GENERATED EVENT ?
YES
NO
A5
A5
A5
GET ADDRESS OF VMBLOK FROM I/O BLOCK
B5
GET REAL DEVICE ADDRESS FROM I/O BLOCK
C5
DMKCVTBH
'CALL' CONVERT DEVICE ADDRESS TO HEX
D5
MSG= DMKCCH604I CHANNEL ERROR, USERID=
E5
DMKQCNWT
'CALL' SEND MESSAGE TO OPERATOR
CCHEND
F5
RESTORE CALLER'S REGISTERS
G5
RETURN TO CALLER

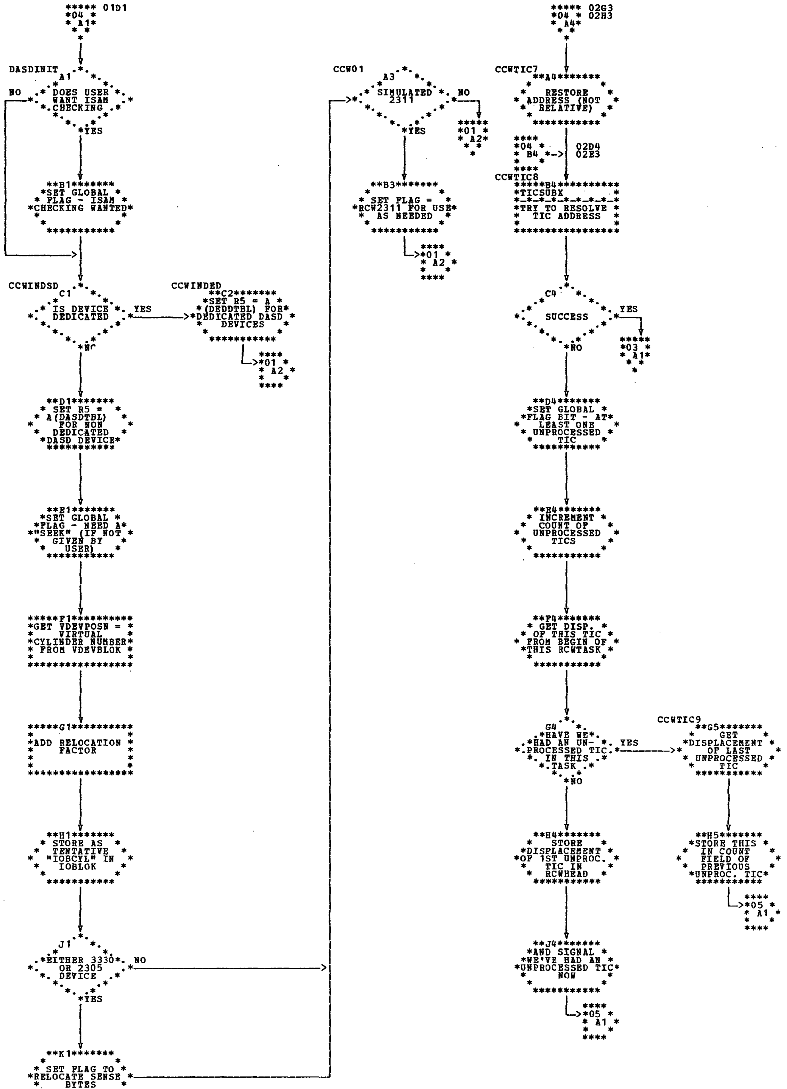

MESSAGE
A2
BUILT MESSAGE
B2
GET FAILING DEVICE ADDRESS
C2
DMKCVTBH
'CALL' CONVERT TO PRINT FORM
D2
WAS IT A CHANNEL OR INTERFACE CONTROL CHECK.
NO
YES
E2
SETUP MESSAGE FOR CHANNEL CONTROL CHECK
F2
MSG= DMKCCH601I CHANNEL CONTROL CHECK
G2
IS THIS A CHANNEL CONTROL CHECK. ?
YES
NO
H2
SETUP THE MESSAGE FOR CHANNEL INTERFACE CHECK
J2
MSG= DMKCCH602I CHANNEL INTERFACE CHECK
MSGOPERA
K2
DMKQCNWT
SEND MESSAGE TO OPERATOR
MSGEXIT
K3
RETURN TO IN LINE CODE

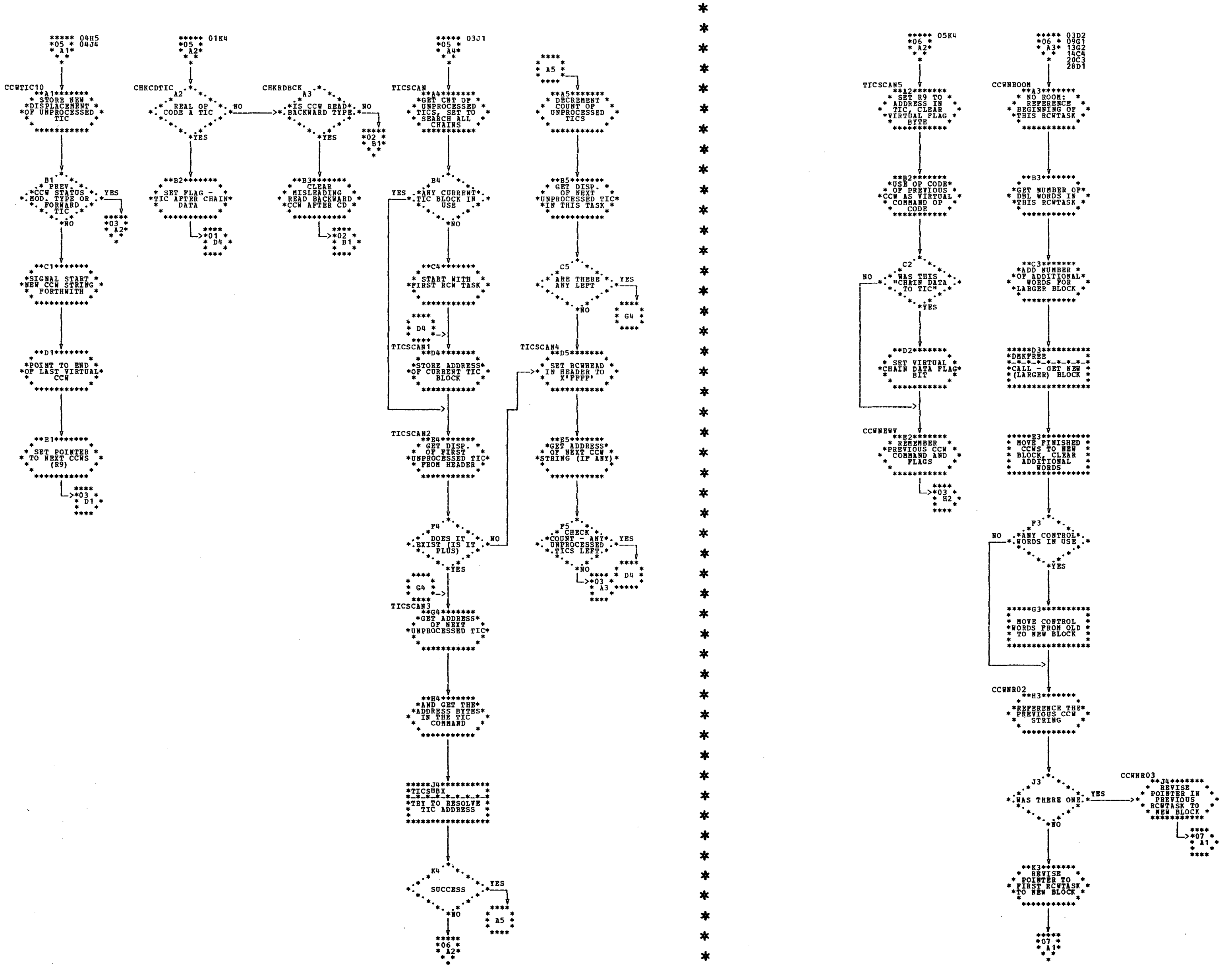

DMKCCWTR
A1
DMKCCWTR
B1
CLEAR FLAGS & WORK AREA, SIGNAL FIRST CCW STRING
C1
FORM R8 = A(VDEVBLOK) TO CHECK DEVICE CLASS & TYPE
D1
DASD DEVICE CLASS
YES
NO
04 A1
E1
SET R5 = A(TAPETBL) FOR TAPE DEVICE CLASS
F1
TAPE DEVICE CLASS
YES
NO
G1
TERMINAL DEVICE CLASS
YES
NO
03 A5
H1
SET R5 = A (OTHRTBL) FOR ALL OTHER CLASSES
A2
01 A2
03B5
03D5
03E5
03H2
04A3
04B3
04C2
CCWGTFRE
A2
DMKFREE
CALL - GET STORAGE FOR HDR AND CCW STRING
A2
B2
CLEAR HEADER AND REAL CCW STRING
C2
SET R7 TO POINT TO BEGINNING OF CONTROL WORDS
D2
INITIALIZE RCWVCAW, RCWCCNT, & RCWHEAD IN HEADER
E2
SET R6 TO POINT TO BEGINNING OF REAL CCWS
F2
ISAM CHECKING WANTED
NO
YES
G2
DECR R7 BY 4 TO RESERVE ONE CONTROL WORD FOR ISAM USE
CCW02
H2
FIRST CCW STRING
YES
NO
A4
CCW04
H3
STORE ADDRESS OF FIRST AND 'THIS' RCWTASK
J3
USER'S CAW DBL WORD ALIGNED
YES
NO
E5
A4
A4
SET 'PREVIOUS' RCWTASK FROM OLD 'THIS' RCWTASK
B4
PATCH CHAIN, CLEAR LOCAL WORK AREA AND FLAG BYTE
C4
STORE ADDRESS OF NEW 'THIS' RCWTASK
01 D4
03D2
05B2
07G1
CCWNXT1
D4
TRANBRNG
GET ADDRESS OF NEXT CCW
E4
IS ADDRESS OF CCW VALID
NO
YES
F4
SET R8 = A(VDEVBLOK) FOR PROCESS CODE
G4
GET VIRTUAL CCW FROM USER'S PAGE (IN R3-R4)
H4
STORE AS REAL CCW, CLEAR BYTE 5 (CP FLAG BYTE)
J4
ADDRESS IN CCW INTO R1, SET R2 AS NEEDED
K4
CHAIN DATA SET IN PREVIOUS CCW
NO
YES
02 A1
05 A2
01 E5
02B4
02D2
02F2
02H1
FNOTE
CCWBAD
E5
SET CCW FLAG BYTE = '03' TO FORCE CHANNEL PROG. CHECK
E5
01 F5
13D5
CCWBAD2
F5
SET OP-CODE AND ADDRESS INVALID = X'00FFFFFF'
G5
SIGNAL INVALID CCW (RCWINVL FLAGBIT)
03 A1
TO: E5
09D3
09F1
09F3
12B4
12D4
14B3
02 A1
01K4
A1
REMEMBER VIRTUAL COMMAND OP CODE
02 B1
05A3
05B3
CCWNXT2
B1
REMEMBER VIRTUAL FLAG BYTE (CC+CD BITS ONLY)
C1
IS ADDRESS IN CCW VALID
NO
YES
D1
CCWNXT3
D1
FORM TABLE INDEXER (0-30) FROM LOW 4 BITS OF CCW OP CODE X 2
E1
GET BRANCH ADDRESS IN R15 FROM R5 PLUS TABLE INDEXER
F1
ISOLATE BYTE-COUNT IN R4, DECREMENT BY ONE
G1
WAS BYTE COUNT ZERO
NO
YES
H1
IF 0, IS THE CCW A TIC.
NO
YES
01 E5
CCWTIC
J1
SET REAL OP CODE TO 08, CLEAR VIRTUAL FLAG
A4
ADDRCHEK
C2
IS IT A "CONTROL" CCW.
YES
NO
D1
D2
IS IT A "WRITE" CCW
YES
NO
01 E5
E2
EITHER 00 OR TIC CCW
YES
NO
F2
READ OR SENSE: IS SKIP FLAG SET.
YES
NO
01 E5
GO TO CORRECT CODE VIA CLASS (R5) & THE LAST 4 BITS OF THE CCW OP CODE
FOR EXAMPLE: DASD CCW '96' GOES TO DASDX6; TAPE CCW '3F' GOES TO TAPEXF
J2
GO TO CORRECT PROCESSING CODE
CCWTIC6
E3
PREVIOUS CCW STATUS MODIFIER TYPE
NO
YES
04 B4
F3
FORM RELATIVE DISPLACEMENT (+) FROM HERE
G3
> 16 BYTES
YES
NO
04 A4
H3
WOULD NEW CCW RUN INTO OUR CONTROL AREA
YES
NO
04 A4
J3
EXACTLY 8 BYTES AHEAD
YES
NO
K5
K3
SET FLAG TO REMEMBER TIC *+16 FORWARD TIC
K5
A4
CCWTIC1
A4
INITIALIZE 2ND WORD OF CCW COMMAND
B4
IS ADDRESS IN TIC VALID
NO
YES
01 E5
C4
REFERENCE THIS RCWTASK - CHECK ADDRESS IN TIC
D4
PRIOR TO BEGINNING OF THIS TASK.
YES
NO
04 B4
E4
BEYOND WHERE WE ARE NOW
YES
NO
F4
HAVE WE HAD A CP-GENERATED CCW. IN THIS TASK
NO
YES
G4
IS THIS A "TIC *-8"
NO
YES
H4
PREVIOUS CCW STATUS MODIFIER TYPE
YES
NO
CCWTIC1A
J4
TICSUBI
RESOLVE TIC ADDRESS
02 K4
07B4
07C4
07F4
07K4
CCWTIC4
K4
SIGNAL ADDRESS IS RELOCATABLE, CHECK PREV. CCW
03 A1
CCWTIC2
H5
MAKE TIC ADDRESS RELATIVE (FROM BEGINNING OF TASK)
J5
HAVE WE HAD AN AUTOPOLL CCW
YES
NO
07 A4
K5
CCWTIC3
K5
ADD PRESENT ADDRESS (R6) TO FORM REAL ADDRESS

| DMKCCW -- Translate a Virtual CCW List to a Real List (Parts 1 and 2 of 31)

| DMKCCW -- Translate a Virtual CCW List to a Real List (Parts 3 and 4 of 31)

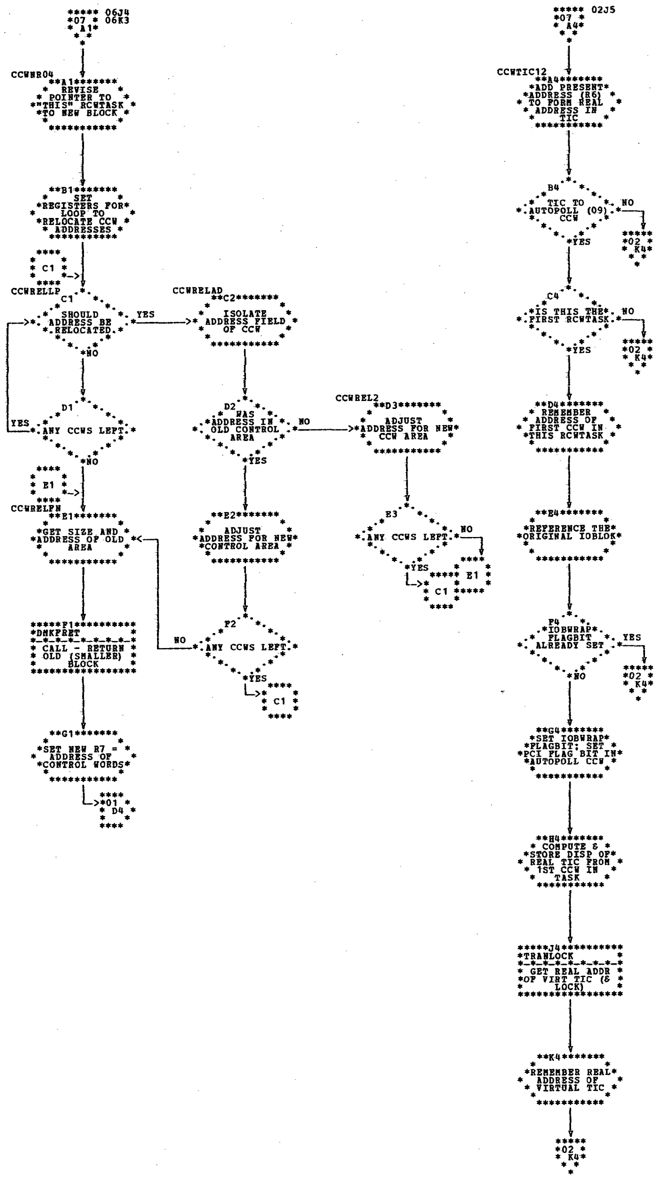

03 A1 01G5 02K4 04C4 10J4 10J5
CCWCHKPV A1 PREV. CCW STATUS MOD. TYPE OR FORWARD TIC
YES
NO
03 A2 05B1 10J5
CCWNEXT A2 INCREMENT POINTER TO NEXT USER CCW (R9)
03 A3 05F5
CCWNXT18 A3 REFERENCE IOBLOK; FILL IN REAL CAW ADDRESS
A3
03 A5 01G1
TERMINIT A5 SET R5 = A(TERMTBL) FOR TERMINAL DEVICE CLASS
CCWNXT13 B1 POINT TO RCWTASK (THIS CHAIN FINISHED)
B2 REMEMBER PREVIOUS CCW COMMAND AND FLAGS
B3 DIALED LINES BEING HANDLED
YES
NO
CCWDIAL B4 REFERENCE VDEVBLOK (AS WELL AS IOBLOK)
B5 LOW SPEED TERMINAL LINE
NO
01 A2
YES
C1 POINT TO END OF LAST VIRTUAL CCW
C2 INCREMENT POINTER TO NEXT REAL CCW (R6)
C3 HAVE WE HAD ANY ISAM READS AT ALL
NO
E3
YES
C4 IS VDEVDIAL FLAG SET
NO
E4
YES
C5 IS LINE DEDICATED
NO
YES
03 D1 05E1
CCWNXT15 D1 COMPUTE COUNT OF VIRTUAL CCWS, STORE IN HEADER
D2 IS THERE ROOM LEFT FOR ANOTHER CCW
YES
01 D4
NO
06 A3
D3 DMKISMTR CALL - PERFORM ADDITIONAL ISAM TRANSLATION
03 E3 09E3
D4 WAS A DISABLE ISSUED
NO
YES
E4
D5 DIALED FLAG ON
NO
01 A2
YES
E1 ALSO COMPUTE COUNT OF REAL CCWS, STORE IN HEADER
CCWEXIT E3 RETURN TO CALLER (DMKVIO OR DMKGEN)
E3
A
CCWDIAL1 E4 SET R8 TO DISP. OF VDEVBLOK
SETDIAL E5 SET R5 = A(DIALTBL) FOR DIALED LINE DEVICES
01 A2
F1 DID WE HAVE ANY ISAM READS IN THIS TASK
NO
YES
F4 CLEAR R2 (PARM); SET TO RETURN 0 IN R1 TO CALLER OF DMKCCW
G1 DISPLACEMENT OF ISAM READ TO LAST WORD IN CCW STRING
G4 DMKDIASM CALL - INVOKE SPECIAL "DIAL" PROCESSING
03 H2 06E2
CCWNEWV2 H2 SIGNAL NOT FIRST CCW STRING
01 A2
CCWNXT16 H1 ARE WE TO START A NEW CCW LIST FORTHWITH
YES
NO
J1 HAVE WE HAD ANY UN-PROCESSED TICS AT ALL
NO
A3
YES
05 A4

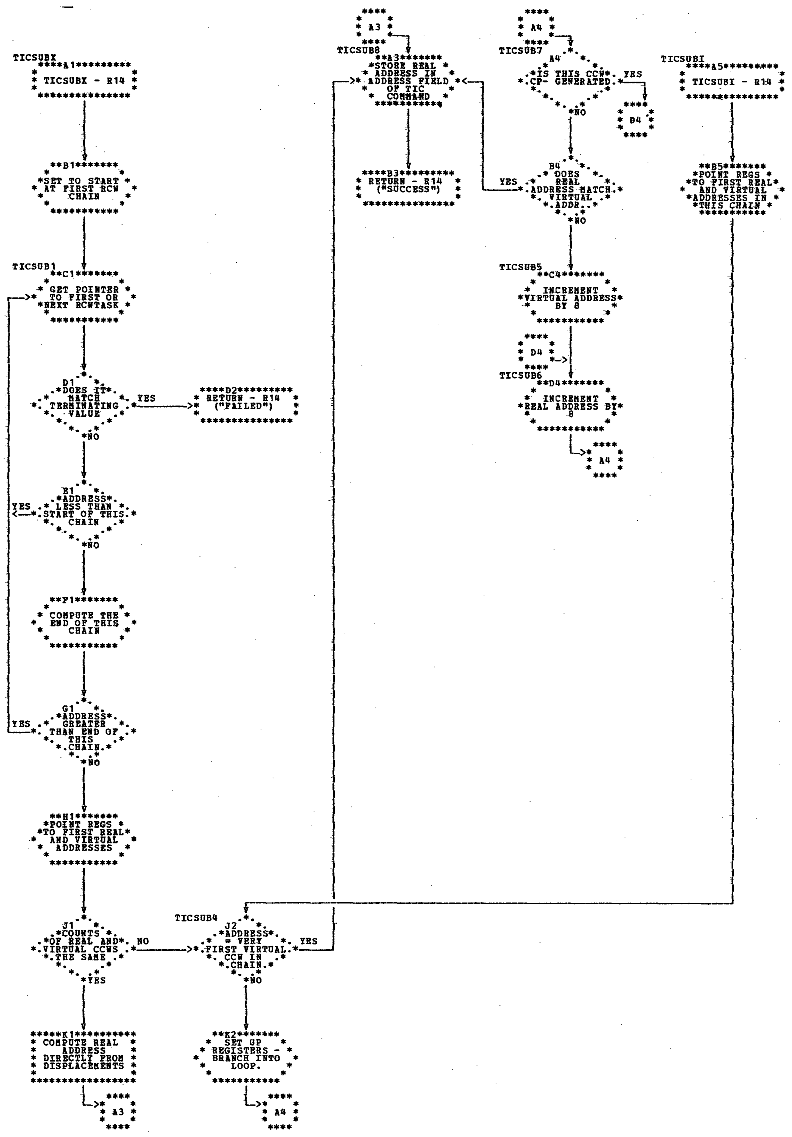

04 A1 01D1
DASDINIT A1 DOES USER WANT ISAM CHECKING
NO
YES
B1 SET GLOBAL FLAG - ISAM CHECKING WANTED
CCWINDSD C1 IS DEVICE DEDICATED
YES
NC
CCWINDED C2 SET R5 = A (DEDDTBL) FOR DEDICATED DASD DEVICES
01 A2
D1 SET R5 = A(DASDTBL) FOR NON DEDICATED DASD DEVICE
E1 SET GLOBAL FLAG - NEED A "SEEK" (IF NOT GIVEN BY USER)
F1 GET VDEVPOSN = VIRTUAL CYLINDER NUMBER FROM VDEVBLOK
G1 ADD RELOCATION FACTOR
H1 STORE AS TENTATIVE "IOBCYL" IN IOBLOK
J1 EITHER 3330 OR 2305 DEVICE
NO
YES
K1 SET FLAG TO RELOCATE SENSE BYTES
CCW01 A3 SIMULATED 2311
NO
01 A2
YES
B3 SET FLAG = RCW2311 FOR USE AS NEEDED
01 A2
04 A4 02G3 02H3
CCWTIC7 A4 RESTORE ADDRESS (NOT RELATIVE)
04 B4 02D4 02E3
CCWTIC8 B4 TICSUBX TRY TO RESOLVE TIC ADDRESS
C4 SUCCESS
YES
03 A1
NO
D4 SET GLOBAL FLAG BIT - AT LEAST ONE UNPROCESSED TIC
E4 INCREMENT COUNT OF UNPROCESSED TICS
F4 GET DISP. OF THIS TIC FROM BEGIN OF THIS RCWTASK
G4 HAVE WE HAD AN UN-PROCESSED TIC IN THIS TASK
YES
NO
CCWTIC9 G5 GET DISPLACEMENT OF LAST UNPROCESSED TIC
H4 STORE DISPLACEMENT OF 1ST UNPROC. TIC IN RCWHEAD
H5 STORE THIS IN COUNT FIELD OF PREVIOUS UNPROC. TIC
05 A1
J4 AND SIGNAL WE'VE HAD AN UNPROCESSED TIC NOW
05 A1

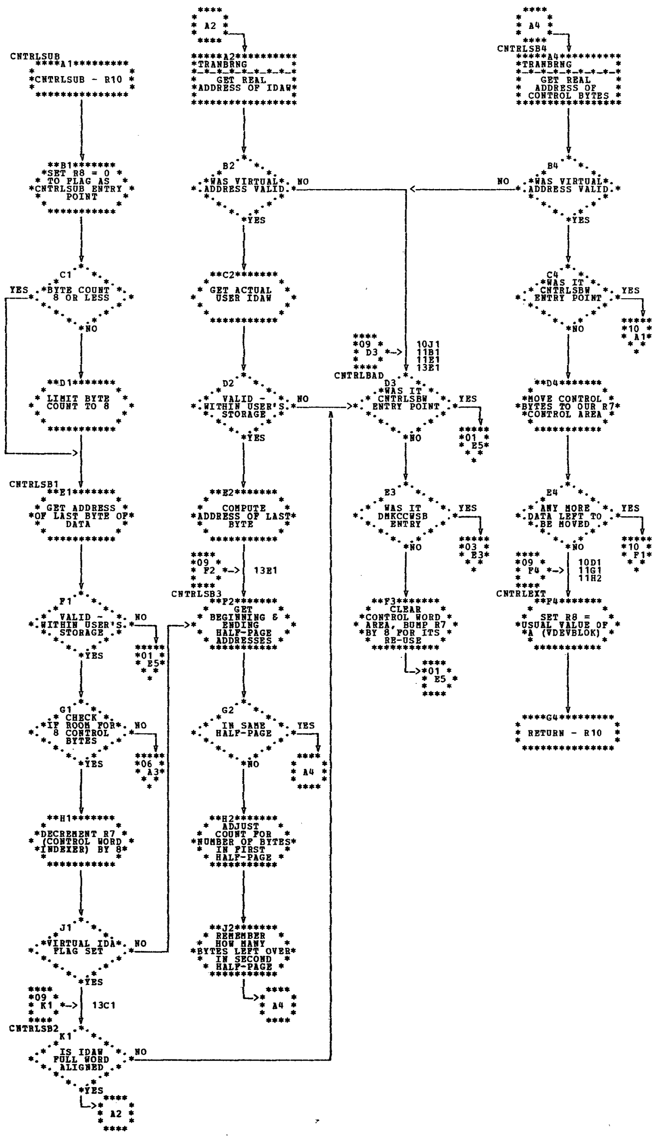

05 A1 04H5 04J4
CCWTIC10
A1 STORE NEW DISPLACEMENT OF UNPROCESSED TIC
B1 PREV. CCW STATUS MOD. TYPE OR FORWARD TIC
YES
03 A2
NO
C1 SIGNAL START NEW CCW STRING FORTHWITH
D1 POINT TO END OF LAST VIRTUAL CCW
E1 SET POINTER TO NEXT CCWS (R9)
03 D1
05 A2 01K4
CHKCDTIC
A2 REAL OP CODE A TIC
NO
YES
B2 SET FLAG - TIC AFTER CHAIN DATA
01 D4
CHKRDBCK
A3 IS CCW READ BACKWARD TYPE
NO
02 B1
YES
B3 CLEAR MISLEADING READ BACKWARD CCW AFTER CD
02 B1
05 A4 03J1
TICSCAN
A4 GET CNT OF UNPROCESSED TICS, SET TO SEARCH ALL CHAINS
B4 ANY CURRENT TIC BLOCK IN USE
YES
NO
C4 START WITH FIRST RCW TASK
D4
TICSCAN1
D4 STORE ADDRESS OF CURRENT TIC BLOCK
TICSCAN2
E4 GET DISP. OF FIRST UNPROCESSED TIC FROM HEADER
F4 DOES IT EXIST (IS IT PLUS)
NO
YES
G4
TICSCAN3
G4 GET ADDRESS OF NEXT UNPROCESSED TIC
H4 AND GET THE ADDRESS BYTES IN THE TIC COMMAND
J4 TICSUBX
TRY TO RESOLVE TIC ADDRESS
K4 SUCCESS
YES
A5
NO
06 A2
A5
A5 DECREMENT COUNT OF UNPROCESSED TICS
B5 GET DISP. OF NEXT UNPROCESSED TIC IN THIS TASK
C5 ARE THERE ANY LEFT
YES
G4
NO
TICSCAN4
D5 SET RCWHEAD IN HEADER TO X'FFFF'
E5 GET ADDRESS OF NEXT CCW STRING (IF ANY)
F5 CHECK COUNT - ANY UNPROCESSED TICS LEFT
YES
D4
NO
03 A3
06 A2 05K4
TICSCAN5
A2 SET R9 TO ADDRESS IN TIC, CLEAR VIRTUAL FLAG BYTE
B2 USE OP CODE OF PREVIOUS CCW AS VIRTUAL COMMAND OP CODE
C2 WAS THIS "CHAIN DATA TO TIC"
NO
YES
D2 SET VIRTUAL CHAIN DATA FLAG BIT
CCWNEWV
E2 REMEMBER PREVIOUS CCW COMMAND AND FLAGS
03 H2
06 A3 03D2 09G1 13G2 14C4 20C3 28D1
CCWNROOM
A3 NO ROOM: REFERENCE BEGINNING OF THIS RCWTASK
B3 GET NUMBER OF DBL WORDS IN THIS RCWTASK
C3 ADD NUMBER OF ADDITIONAL WORDS FOR LARGER BLOCK
D3 DMKFREE
CALL - GET NEW (LARGER) BLOCK
E3 MOVE FINISHED CCWS TO NEW BLOCK, CLEAR ADDITIONAL WORDS
F3 ANY CONTROL WORDS IN USE
NO
YES
G3 MOVE CONTROL WORDS FROM OLD TO NEW BLOCK
CCWNR02
H3 REFERENCE THE PREVIOUS CCW STRING
J3 WAS THERE ONE
YES
CCWNR03
J4 REVISE POINTER IN PREVIOUS RCWTASK TO NEW BLOCK
07 A1
NO
K3 REVISE POINTER TO FIRST RCWTASK TO NEW BLOCK
07 A1

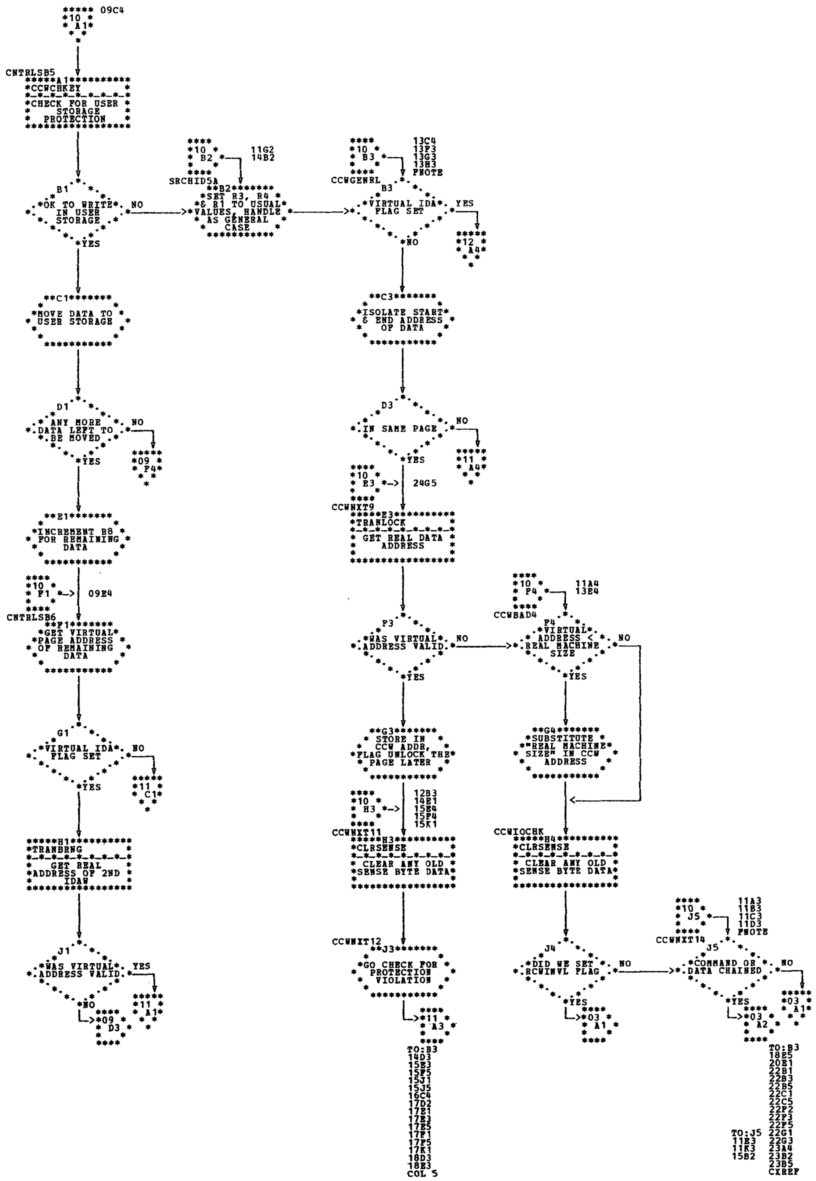
| DMKCCW -- Translate a Virtual CCW List to a Real List (Parts 5 and 6 of 31)

| DMKCCW -- Translate a Virtual CCW List to a Real List (Parts 7 and 8 of 31)


07 A1 06J4 06K3
CCWNR04
A1 REVISE POINTER TO "THIS" RCWTASK TO NEW BLOCK
B1 SET REGISTERS FOR LOOP TO RELOCATE CCW ADDRESSES
C1
CCWRELLP
C1 SHOULD ADDRESS BE RELOCATED
YES
NO
CCWRELAD
C2 ISOLATE ADDRESS FIELD OF CCW
D1 ANY CCWS LEFT
YES
NO
D2 WAS ADDRESS IN OLD CONTROL AREA
NO
YES
CCWREL2
D3 ADJUST ADDRESS FOR NEW CCW AREA
E1
CCWRELFN
E1 GET SIZE AND ADDRESS OF OLD AREA
E2 ADJUST ADDRESS FOR NEW CONTROL AREA
E3 ANY CCWS LEFT
NO
YES
E1
C1
F1 DMKFRET CALL - RETURN OLD (SMALLER) BLOCK
F2 ANY CCWS LEFT
NO
YES
C1
G1 SET NEW R7 = ADDRESS OF CONTROL WORDS
01 D4
07 A4 02J5
CCWTIC12
A4 ADD PRESENT ADDRESS (R6) TO FORM REAL ADDRESS IN TIC
B4 TIC TO AUTOPOLL (09) CCW
NO
02 K4
YES
C4 IS THIS THE FIRST RCWTASK
NO
02 K4
YES
D4 REMEMBER ADDRESS OF FIRST CCW IN THIS RCWTASK
E4 REFERENCE THE ORIGINAL IOBLOK
F4 IOBWRAP FLAGBIT ALREADY SET
YES
02 K4
NO
G4 SET IOBWRAP FLAGBIT; SET PCI FLAG BIT IN AUTOPOLL CCW
H4 COMPUTE & STORE DISP OF REAL TIC FROM 1ST CCW IN TASK
J4 TRANLOCK GET REAL ADDR OF VIRT TIC (& LOCK)
K4 REMEMBER REAL ADDRESS OF VIRTUAL TIC
02 K4

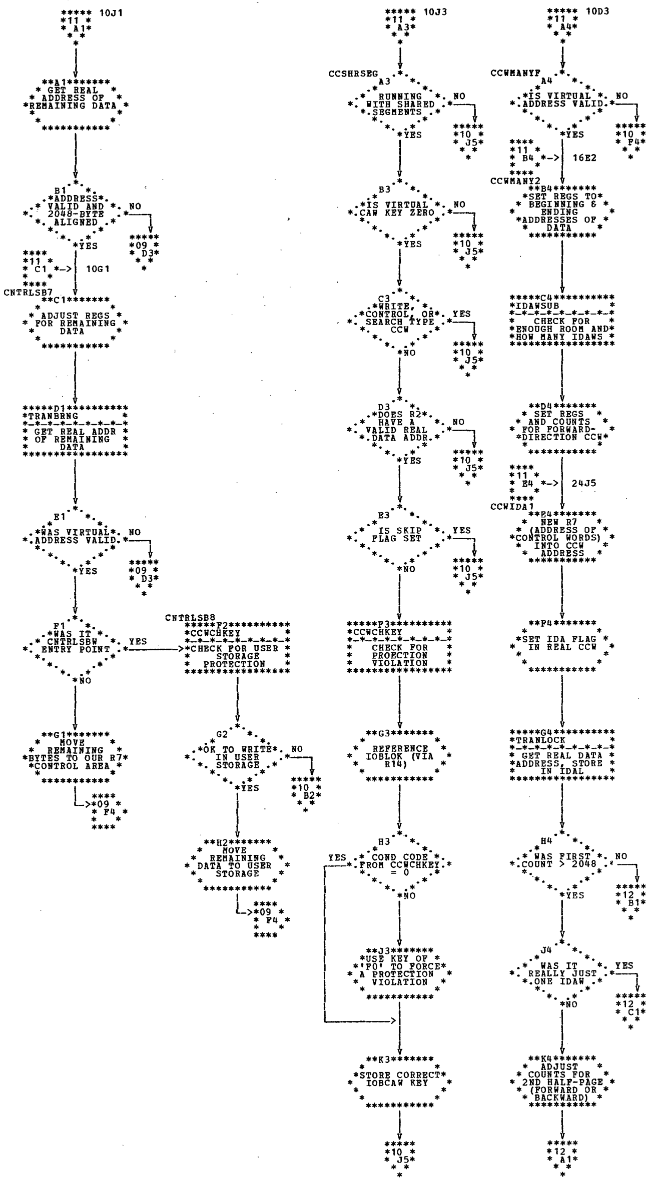

TICSUBX
A1 TICSUBX - R14
B1 SET TO START AT FIRST RCW CHAIN
TICSUB1
C1 GET POINTER TO FIRST OR NEXT RCWTASK
D1 DOES IT MATCH TERMINATING VALUE
YES
D2 RETURN - R14 ("FAILED")
NO
E1 ADDRESS LESS THAN START OF THIS CHAIN
YES
NO
F1 COMPUTE THE END OF THIS CHAIN
G1 ADDRESS GREATER THAN END OF THIS CHAIN
YES
NO
H1 POINT REGS TO FIRST REAL AND VIRTUAL ADDRESSES
J1 COUNTS OF REAL AND VIRTUAL CCWS THE SAME
NO
YES
TICSUB4
J2 ADDRESS = VERY FIRST VIRTUAL CCW IN CHAIN
YES
NO
K1 COMPUTE REAL ADDRESS DIRECTLY FROM DISPLACEMENTS
A3
K2 SET UP REGISTERS - BRANCH INTO LOOP.
A4
A3
TICSUB8
A3 STORE REAL ADDRESS IN ADDRESS FIELD OF TIC COMMAND
B3 RETURN - R14 ("SUCCESS")
A4
TICSUB7
A4 IS THIS CCW CP- GENERATED
YES
D4
NO
B4 DOES REAL ADDRESS MATCH VIRTUAL ADDR.
YES
NO
TICSUB5
C4 INCREMENT VIRTUAL ADDRESS BY 8
D4
TICSUB6
D4 INCREMENT REAL ADDRESS BY 8
A4
TICSUBI
A5 TICSUBI - R14
B5 POINT REGS TO FIRST REAL AND VIRTUAL ADDRESSES IN THIS CHAIN

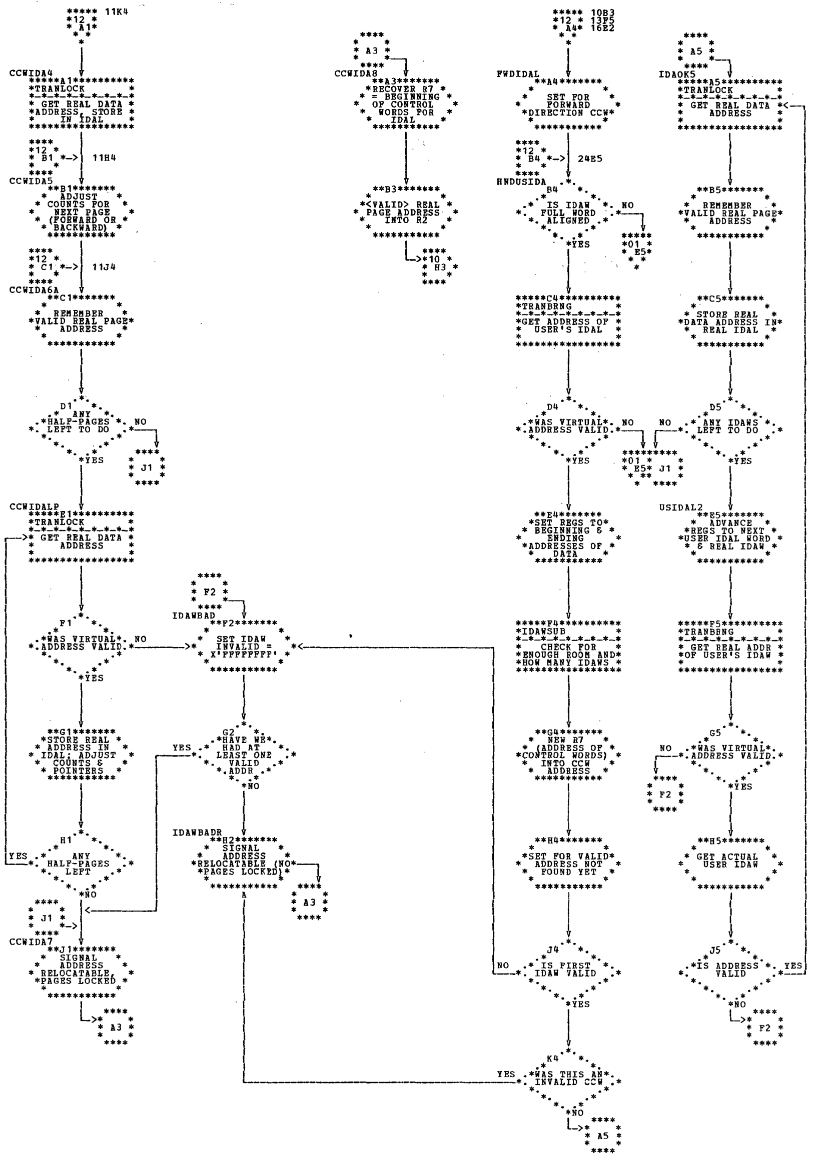

CNTRLSUB
A1
CNTRLSUB - R10
B1
SET R8 = 0 TO FLAG AS CNTRLSUB ENTRY POINT
C1
BYTE COUNT 8 OR LESS
YES
NO
D1
LIMIT BYTE COUNT TO 8
CNTRLSB1
E1
GET ADDRESS OF LAST BYTE OF DATA
F1
VALID - WITHIN USER'S STORAGE
NO
01 E5
YES
G1
CHECK IF ROOM FOR 8 CONTROL BYTES
NO
06 A3
YES
H1
DECREMENT R7 (CONTROL WORD INDEXER) BY 8
J1
VIRTUAL IDA FLAG SET
NO
YES
09 K1
13C1
CNTRLSB2
K1
IS IDAW FULL WORD ALIGNED
NO
YES
A2
A2
A2
TRANBRNG
GET REAL ADDRESS OF IDAW
B2
WAS VIRTUAL ADDRESS VALID
NO
YES
C2
GET ACTUAL USER IDAW
D2
VALID - WITHIN USER'S STORAGE
NO
YES
A
E2
COMPUTE ADDRESS OF LAST BYTE
09 F2
13E1
CNTRLSB3
F2
GET BEGINNING & ENDING HALF-PAGE ADDRESSES
G2
IN SAME HALF-PAGE
YES
A4
NO
H2
ADJUST COUNT FOR NUMBER OF BYTES IN FIRST HALF-PAGE
J2
REMEMBER HOW MANY BYTES LEFT OVER IN SECOND HALF-PAGE
A4
09 D3
10J1
11B1
11E1
13E1
CNTRLBAD
D3
WAS IT CNTRLSBW ENTRY POINT
YES
01 E5
NO
E3
WAS IT DMKCCWSB ENTRY
YES
03 E3
NO
F3
CLEAR CONTROL WORD AREA, BUMP R7 BY 8 FOR ITS RE-USE
01 E5
A4
CNTRLSB4
A4
TRANBRNG
GET REAL ADDRESS OF CONTROL BYTES
B4
WAS VIRTUAL ADDRESS VALID
NO
YES
C4
WAS IT CNTRLSBW ENTRY POINT
YES
10 A1
NO
D4
MOVE CONTROL BYTES TO OUR R7 CONTROL AREA
E4
ANY MORE DATA LEFT TO BE MOVED
YES
10 F1
NO
09 F4
10D1
11G1
11H2
CNTRLEXT
F4
SET R8 = USUAL VALUE OF A (VDEVBLOK)
G4
RETURN - R10

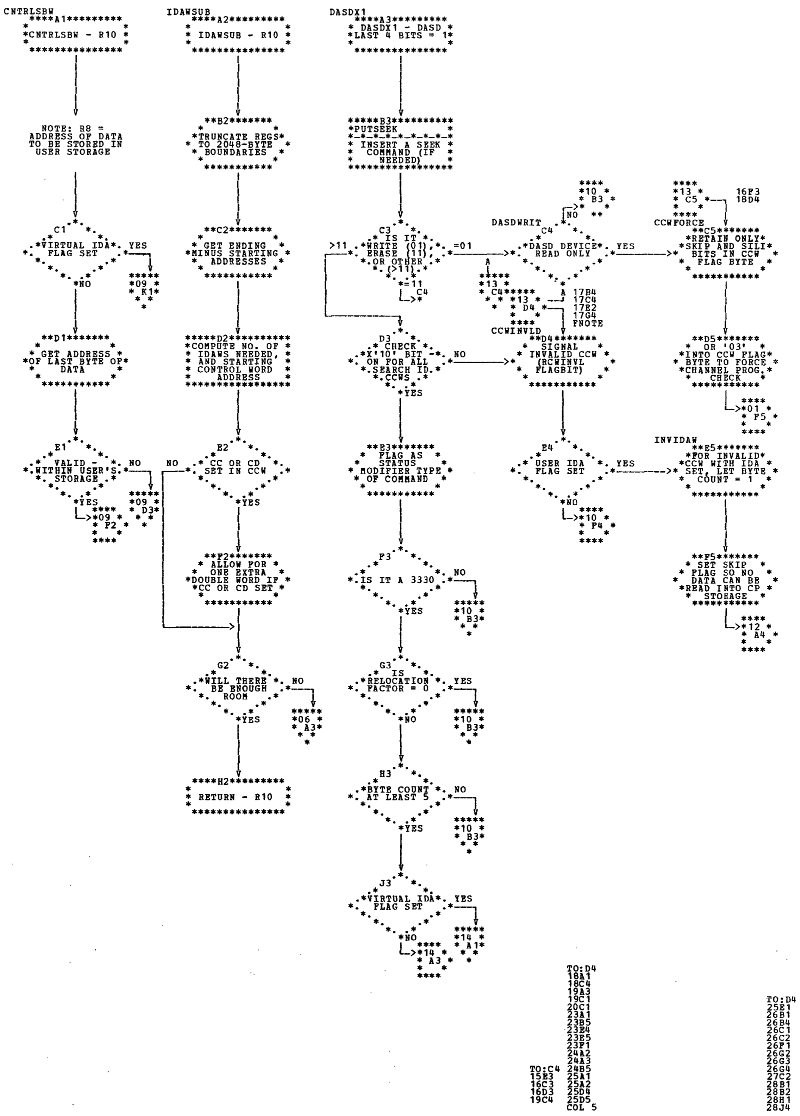


| DMKCCW -- Translate a Virtual CCW List to a Real List (Parts 9 and 10 of 31)

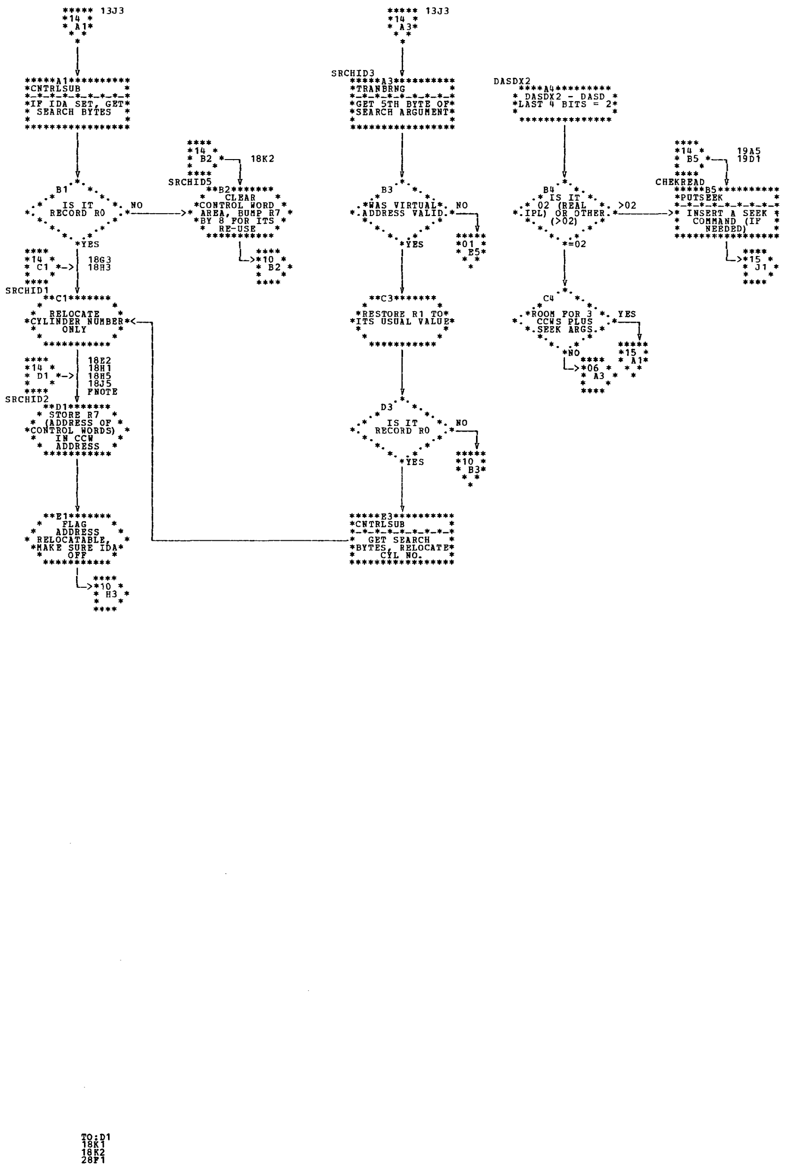
| DMKCCW -- Translate a Virtual CCW List to a Real List (Parts 11 and 12 of 31)


10J1
11 A1
A1 GET REAL ADDRESS OF REMAINING DATA
B1 ADDRESS VALID AND 2048-BYTE ALIGNED
NO
09 D3
YES
11 C1
10G1
CNTRLSB7
C1 ADJUST REGS FOR REMAINING DATA
D1 TRANBRNG GET REAL ADDR OF REMAINING DATA
E1 WAS VIRTUAL ADDRESS VALID
NO
09 D3
YES
F1 WAS IT CNTRLSBW ENTRY POINT
YES
NO
CNTRLSB8
F2 CCWCHKEY CHECK FOR USER STORAGE PROTECTION
G1 MOVE REMAINING BYTES TO OUR R7 CONTROL AREA
09 F4
G2 OK TO WRITE IN USER STORAGE
NO
10 B2
YES
H2 MOVE REMAINING DATA TO USER STORAGE
09 F4
10J3
11 A3
CCSHRSEG
A3 RUNNING WITH SHARED SEGMENTS
NO
10 J5
YES
B3 IS VIRTUAL CAW KEY ZERO
NO
10 J5
YES
C3 WRITE, CONTROL, OR SEARCH TYPE CCW
YES
10 J5
NO
D3 DOES R2 HAVE A VALID REAL DATA ADDR.
NO
10 J5
YES
E3 IS SKIP FLAG SET
YES
10 J5
NO
F3 CCWCHKEY CHECK FOR PROECTION VIOLATION
G3 REFERENCE IOBLOK (VIA R14)
H3 COND CODE FROM CCWCHKEY = 0
YES
NO
J3 USE KEY OF 'F0' TO FORCE A PROTECTION VIOLATION
K3 STORE CORRECT IOBCAW KEY
10 J5
10D3
11 A4
CCWMANYF
A4 IS VIRTUAL ADDRESS VALID
NO
10 F4
YES
11 B4
16E2
CCWMANY2
B4 SET REGS TO BEGINNING & ENDING ADDRESSES OF DATA
C4 IDAWSUB CHECK FOR ENOUGH ROOM AND HOW MANY IDAWS
D4 SET REGS AND COUNTS FOR FORWARD-DIRECTION CCW
11 E4
24J5
CCWIDA1
E4 NEW R7 (ADDRESS OF CONTROL WORDS) INTO CCW ADDRESS
F4 SET IDA FLAG IN REAL CCW
G4 TRANLOCK GET REAL DATA ADDRESS, STORE IN IDAL
H4 WAS FIRST COUNT > 2048
NO
12 B1
YES
J4 WAS IT REALLY JUST ONE IDAW
YES
12 C1
NO
K4 ADJUST COUNTS FOR 2ND HALF-PAGE (FORWARD OR BACKWARD)
12 A1


11K4
12 A1
CCWIDA4
A1 TRANLOCK GET REAL DATA ADDRESS, STORE IN IDAL
12 B1
11H4
CCWIDA5
B1 ADJUST COUNTS FOR NEXT PAGE (FORWARD OR BACKWARD)
12 C1
11J4
CCWIDA6A
C1 REMEMBER VALID REAL PAGE ADDRESS
D1 ANY HALF-PAGES LEFT TO DO
NO
J1
YES
CCWIDALP
E1 TRANLOCK GET REAL DATA ADDRESS
F1 WAS VIRTUAL ADDRESS VALID
NO
YES
G1 STORE REAL ADDRESS IN IDAL; ADJUST COUNTS & POINTERS
H1 ANY HALF-PAGES LEFT
YES
NO
J1
CCWIDA7
J1 SIGNAL ADDRESS RELOCATABLE, PAGES LOCKED
A3
F2
IDAWBAD
F2 SET IDAW INVALID = X'FFFFFFFF'
G2 HAVE WE HAD AT LEAST ONE VALID ADDR
YES
NO
IDAWBADR
H2 SIGNAL ADDRESS RELOCATABLE (NO PAGES LOCKED)
A3
A3
CCWIDA8
A3 RECOVER R7 = BEGINNING OF CONTROL WORDS FOR IDAL
B3 <VALID> REAL PAGE ADDRESS INTO R2
10 H3
10B3
13F5
16E2
12 A4
FWDIDAL
A4 SET FOR FORWARD DIRECTION CCW
12 B4
24E5
HNDUSIDA
B4 IS IDAW FULL WORD ALIGNED
NO
01 E5
YES
C4 TRANBRNG GET ADDRESS OF USER'S IDAL
D4 WAS VIRTUAL ADDRESS VALID
NO
01 E5
YES
E4 SET REGS TO BEGINNING & ENDING ADDRESSES OF DATA
F4 IDAWSUB CHECK FOR ENOUGH ROOM AND HOW MANY IDAWS
G4 NEW R7 (ADDRESS OF CONTROL WORDS) INTO CCW ADDRESS
H4 SET FOR VALID ADDRESS NOT FOUND YET
J4 IS FIRST IDAW VALID
NO
YES
K4 WAS THIS AN INVALID CCW
YES
NO
A5
A5
IDAOK5
A5 TRANLOCK GET REAL DATA ADDRESS
B5 REMEMBER VALID REAL PAGE ADDRESS
C5 STORE REAL DATA ADDRESS IN REAL IDAL
D5 ANY IDAWS LEFT TO DO
NO
J1
YES
USIDAL2
E5 ADVANCE REGS TO NEXT USER IDAL WORD & REAL IDAW
F5 TRANBRNG GET REAL ADDR OF USER'S IDAW
G5 WAS VIRTUAL ADDRESS VALID
NO
F2
YES
H5 GET ACTUAL USER IDAW
J5 IS ADDRESS VALID
YES
NO
F2



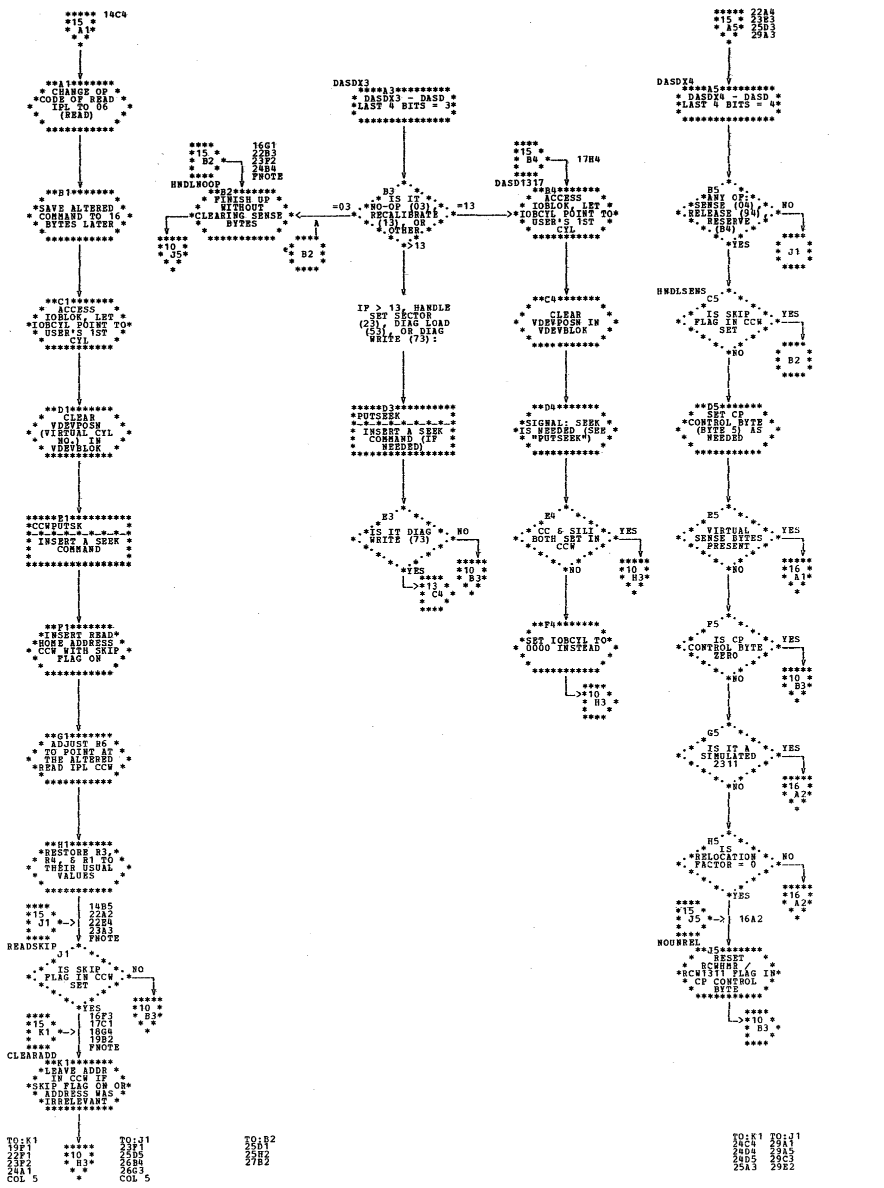
CNTRLSBW
A1
CNTRLSBW - R10
NOTE: R8 = ADDRESS OF DATA TO BE STORED IN USER STORAGE
C1
VIRTUAL IDA FLAG SET
YES
09 K1
NO
D1
GET ADDRESS OF LAST BYTE OF DATA
E1
VALID - WITHIN USER'S STORAGE
NO
09 D3
YES
09 F2
IDAWSUB
A2
IDAWSUB - R10
B2
TRUNCATE REGS TO 2048-BYTE BOUNDARIES
C2
GET ENDING MINUS STARTING ADDRESSES
D2
COMPUTE NO. OF IDAWS NEEDED, AND STARTING CONTROL WORD ADDRESS
E2
CC OR CD SET IN CCW
NO
YES
F2
ALLOW FOR ONE EXTRA DOUBLE WORD IF CC OR CD SET
G2
WILL THERE BE ENOUGH ROOM
NO
06 A3
YES
H2
RETURN - R10
DASDX1
A3
DASDX1 - DASD LAST 4 BITS = 1
B3
PUTSEEK
INSERT A SEEK COMMAND (IF NEEDED)
C3
IS IT WRITE (01), ERASE (11), OR OTHER (>11)
>11
=01
=11
C4
D3
CHECK X'10' BIT ON FOR ALL SEARCH ID. CCWS
NO
YES
E3
FLAG AS STATUS MODIFIER TYPE OF COMMAND
F3
IS IT A 3330
NO
10 B3
YES
G3
IS RELOCATION FACTOR = 0
YES
10 B3
NO
H3
BYTE COUNT AT LEAST 5
NO
10 B3
YES
J3
VIRTUAL IDA FLAG SET
YES
14 A1
NO
14 A3
10 B3
NO
DASDWRIT
C4
DASD DEVICE READ ONLY
YES
A
13 C4
A
13 D4
17B4
17C4
17E2
17G4
FNOTE
CCWINVLD
D4
SIGNAL INVALID CCW (RCWINVL FLAGBIT)
E4
USER IDA FLAG SET
YES
NO
10 F4
13 C5
16F3
18D4
CCWFORCE
C5
RETAIN ONLY SKIP AND SILI BITS IN CCW FLAG BYTE
D5
OR '03' INTO CCW FLAG BYTE TO FORCE CHANNEL PROG. CHECK
01 F5
INVIDAW
E5
FOR INVALID CCW WITH IDA SET, LET BYTE COUNT = 1
F5
SET SKIP FLAG SO NO DATA CAN BE READ INTO CP STORAGE
12 A4
TO:D4
18A1
18C4
19A3
19C1
20C1
23A1
23B5
23E4
23E5
23F1
24A2
24A3
24B5
25A1
25A2
25D4
25D5
COL 5
TO:C4
15E3
16C3
16D3
19C4
TO:D4
25E1
26B1
26B4
26C1
26C2
26F1
26G2
26G3
26G4
27C2
28B1
28B2
28H1
28J4

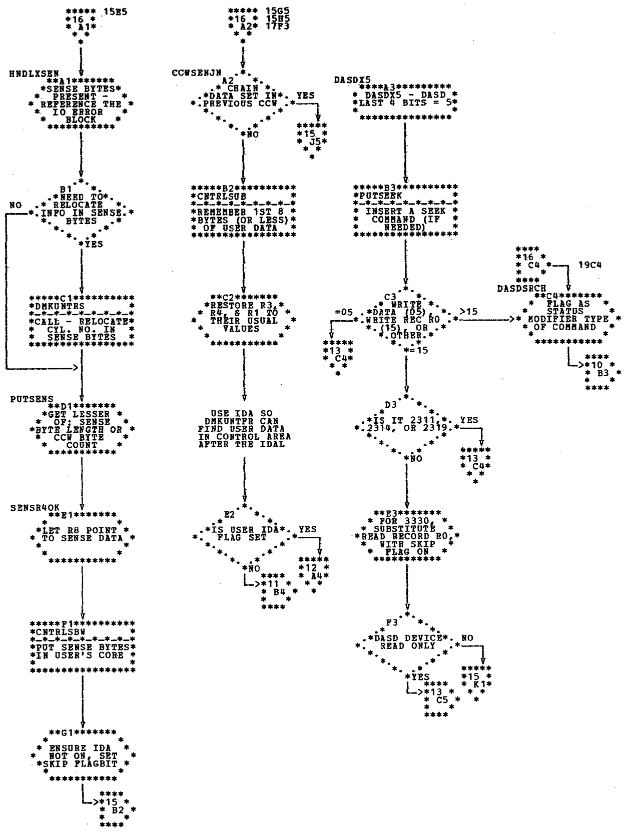


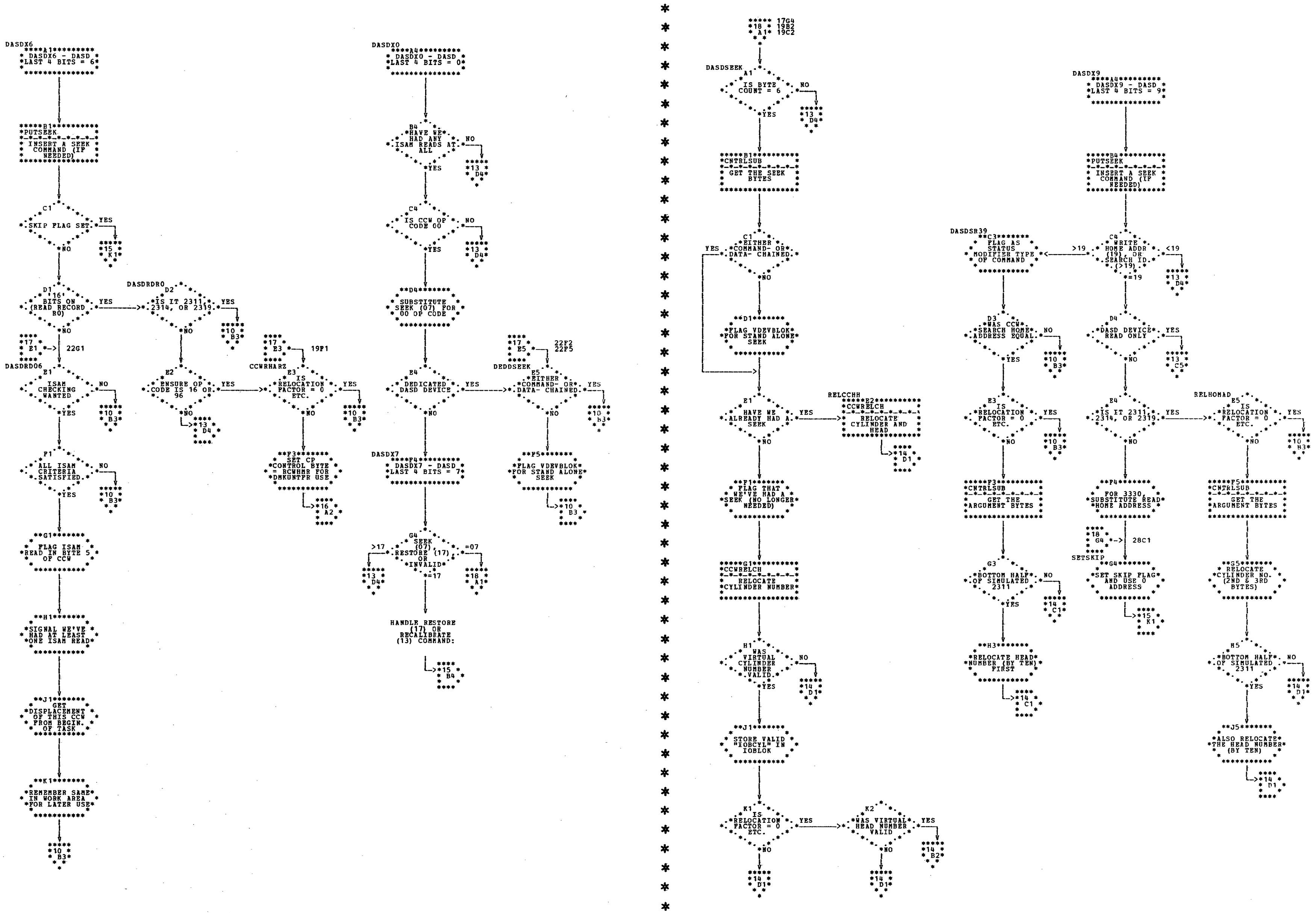
14 A1
13J3
A1
CNTRLSUB
IF IDA SET, GET SEARCH BYTES
B1
IS IT RECORD R0
NO
YES
14 C1
18G3
18H3
SRCHID1
C1
RELOCATE CYLINDER NUMBER ONLY
14 D1
18E2
18H1
18H5
18J5
FNOTE
SRCHID2
D1
STORE R7 (ADDRESS OF CONTROL WORDS) IN CCW ADDRESS
E1
FLAG ADDRESS RELOCATABLE, MAKE SURE IDA OFF
10 H3
14 B2
18K2
SRCHID5
B2
CLEAR CONTROL WORD AREA, BUMP R7 BY 8 FOR ITS RE-USE
10 B2
14 A3
13J3
SRCHID3
A3
TRANBRNG
GET 5TH BYTE OF SEARCH ARGUMENT
B3
WAS VIRTUAL ADDRESS VALID
NO
01 E5
YES
C3
RESTORE R1 TO ITS USUAL VALUE
D3
IS IT RECORD R0
NO
10 B3
YES
E3
CNTRLSUB
GET SEARCH BYTES, RELOCATE CYL NO.
DASDX2
A4
DASDX2 - DASD LAST 4 BITS = 2
B4
IS IT 02 (REAL IPL) OR OTHER (>02)
>02
=02
C4
ROOM FOR 3 CCWS PLUS SEEK ARGS.
YES
15 A1
NO
06 A3
14 B5
19A5
19D1
CHEKREAD
B5
PUTSEEK
INSERT A SEEK COMMAND (IF NEEDED)
15 J1
TO:D1
18K1
18K2
28F1

| DMKCCW -- Translate a Virtual CCW List to a Real List (Parts 13 and 14 of 31)

| DMKCCW -- Translate a Virtual CCW List of a Real List (Parts 15 and 16 of 31)

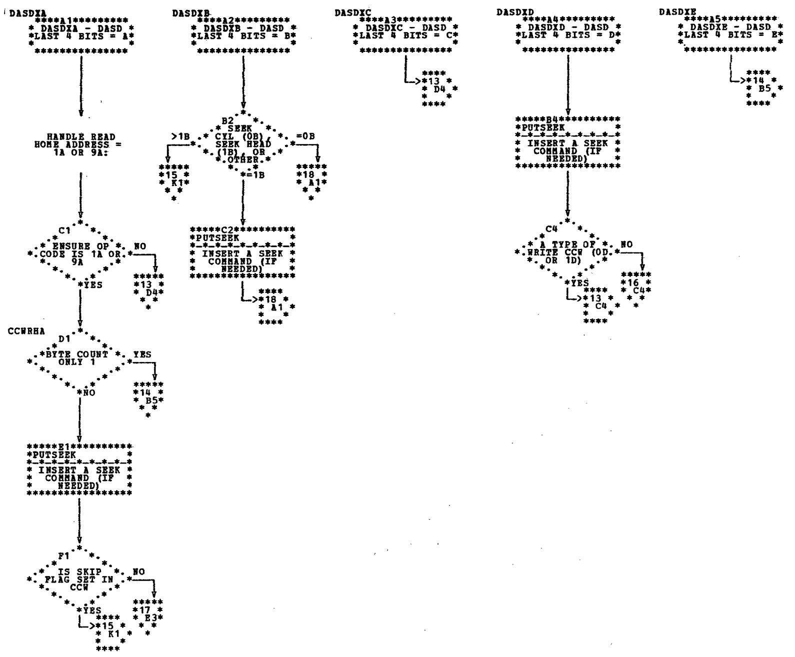

15 A1 14C4
A1 CHANGE OP CODE OF READ IPL TO 06 (READ)
B1 SAVE ALTERED COMMAND TO 16 BYTES LATER
C1 ACCESS IOBLOK, LET IOBCYL POINT TO USER'S 1ST CYL
D1 CLEAR VDEVPOSN (VIRTUAL CYL NO.) IN VDEVBLOK
E1 CCWPUTSK INSERT A SEEK COMMAND
F1 INSERT READ HOME ADDRESS CCW WITH SKIP FLAG ON
G1 ADJUST R6 TO POINT AT THE ALTERED READ IPL CCW
H1 RESTORE R3, R4, & R1 TO THEIR USUAL VALUES
15 J1 14B5 22A2 22E4 23A3 FNOTE
READSKIP
J1 IS SKIP FLAG IN CCW SET
NO
10 B3
YES
15 K1 16F3 17C1 18G4 19B2 FNOTE
CLEARADD
K1 LEAVE ADDR IN CCW IF SKIP FLAG ON OR ADDRESS WAS IRRELEVANT
10 H3
TO:K1 19F1 22F1 23F2 24A1 COL 5
TO:J1 23F1 25D5 26B4 26G3 COL 5
15 B2 16G1 22B3 23F2 24B4 FNOTE
HNDLNOOP
B2 FINISH UP WITHOUT CLEARING SENSE BYTES
10 J5
A
B2
TO:B2 25D1 25H2 27B2
DASDX3
A3 DASDX3 - DASD LAST 4 BITS = 3
=03
B3 IS IT NO-OP (03), RECALIBRATE (13), OR OTHER.
=13
>13
IF > 13, HANDLE SET SECTOR (23), DIAG LOAD (53), OR DIAG WRITE (73):
D3 PUTSEEK INSERT A SEEK COMMAND (IF NEEDED)
E3 IS IT DIAG WRITE (73)
NO
10 B3
YES
13 C4
15 B4 17H4
DASD1317
B4 ACCESS IOBLOK, LET IOBCYL POINT TO USER'S 1ST CYL
C4 CLEAR VDEVPOSN IN VDEVBLOK
D4 SIGNAL: SEEK IS NEEDED (SEE "PUTSEEK")
E4 CC & SILI BOTH SET IN CCW
YES
10 H3
NO
F4 SET IOBCYL TO 0000 INSTEAD
10 H3
15 A5 22A4 23E3 25D3 29A3
DASDX4
A5 DASDX4 - DASD LAST 4 BITS = 4
B5 ANY OF: SENSE (04), RELEASE (94), RESERVE (B4)
NO
J1
YES
HNDLSENS
C5 IS SKIP FLAG IN CCW SET
YES
B2
NO
D5 SET CP CONTROL BYTE (BYTE 5) AS NEEDED
E5 VIRTUAL SENSE BYTES PRESENT
YES
16 A1
NO
F5 IS CP CONTROL BYTE ZERO
YES
10 B3
NO
G5 IS IT A SIMULATED 2311
YES
16 A2
NO
H5 IS RELOCATION FACTOR = 0
NO
16 A2
YES
15 J5 16A2
NOUNREL
J5 RESET RCWHMR / RCW1311 FLAG IN CP CONTROL BYTE
10 B3
TO:K1 24C4 24D4 24D5 25A3
TO:J1 29A1 29A5 29C3 29E2

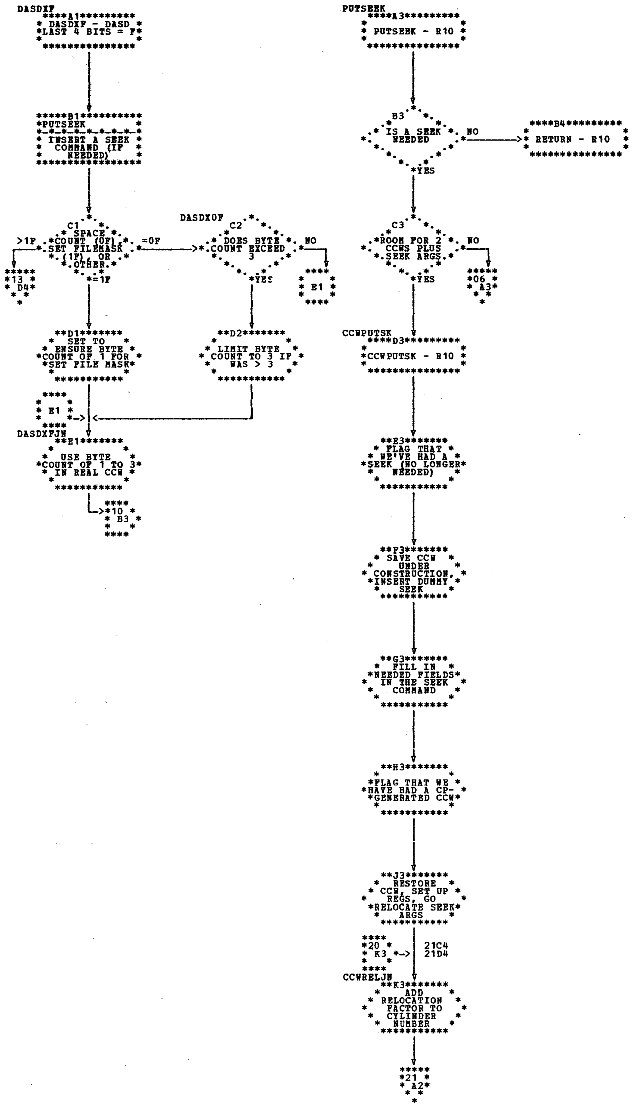

16 A1 15E5
HNDLXSEN
A1 SENSE BYTES PRESENT - REFERENCE THE IO ERROR BLOCK
B1 NEED TO RELOCATE INFO IN SENSE BYTES
NO
YES
C1 DMKUNTRS CALL - RELOCATE CYL. NO. IN SENSE BYTES
PUTSENS
D1 GET LESSER OF: SENSE BYTE LENGTH OR CCW BYTE COUNT
SENSR4OK
E1 LET R8 POINT TO SENSE DATA
F1 CNTRLSBW PUT SENSE BYTES IN USER'S CORE
G1 ENSURE IDA NOT ON, SET SKIP FLAGBIT
15 B2
16 A2 15G5 15H5 17F3
CCWSENJN
A2 CHAIN DATA SET IN PREVIOUS CCW
YES
15 J5
NO
B2 CNTRLSUB REMEMBER 1ST 8 BYTES (OR LESS) OF USER DATA
C2 RESTORE R3, R4, & R1 TO THEIR USUAL VALUES
USE IDA SO DMKUNTFR CAN FIND USER DATA IN CONTROL AREA AFTER THE IDAL
E2 IS USER IDA FLAG SET
YES
12 A4
NO
11 B4
DASDX5
A3 DASDX5 - DASD LAST 4 BITS = 5
B3 PUTSEEK INSERT A SEEK COMMAND (IF NEEDED)
C3 WRITE DATA (05), WRITE REC R0 (15), OR OTHER.
=05
13 C4
>15
=15
D3 IS IT 2311, 2314, OR 2319.
YES
13 C4
NO
E3 FOR 3330, SUBSTITUTE READ RECORD R0, WITH SKIP FLAG ON
F3 DASD DEVICE READ ONLY
NO
15 K1
YES
13 C5
16 C4 19C4
DASDSRCH
C4 FLAG AS STATUS MODIFIER TYPE OF COMMAND
10 B3

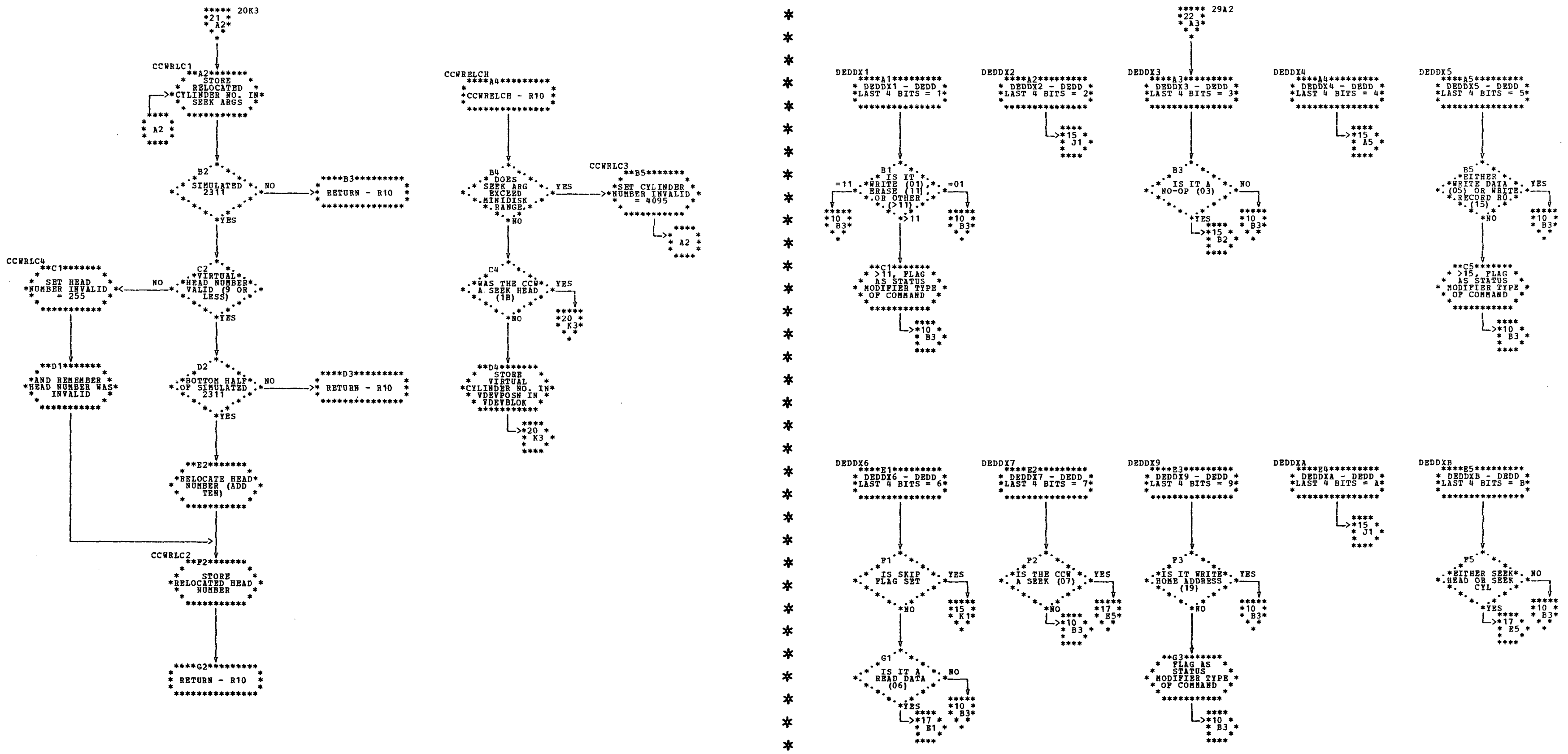

DASDX6
A1 DASDX6 - DASD LAST 4 BITS = 6
B1 PUTSEEK INSERT A SEEK COMMAND (IF NEEDED)
C1 SKIP FLAG SET
YES
15 K1
NO
D1 '16' BITS ON (READ RECORD R0)
YES
NO
17 E1
22G1
DASDRDR0
D2 IS IT 2311, 2314, OR 2319
YES
10 B3
NO
DASDRD06
E1 ISAM CHECKING WANTED
NO
10 B3
YES
E2 ENSURE OP CODE IS 16 OR 96
YES
NO
13 D4
17 E3
19F1
CCWRHARZ
E3 IS RELOCATION FACTOR = 0 ETC.
YES
10 B3
NO
F1 ALL ISAM CRITERIA SATISFIED
NO
10 B3
YES
F3 SET CP CONTROL BYTE = RCWHMR FOR DMKUNTFR USE
16 A2
G1 FLAG ISAM READ IN BYTE 5 OF CCW
H1 SIGNAL WE'VE HAD AT LEAST ONE ISAM READ
J1 GET DISPLACEMENT OF THIS CCW FROM BEGIN. OF TASK
K1 REMEMBER SAME IN WORK AREA FOR LATER USE
10 B3
DASDX0
A4 DASDX0 - DASD LAST 4 BITS = 0
B4 HAVE WE HAD ANY ISAM READS AT ALL
NO
13 D4
YES
C4 IS CCW OP CODE 00
NO
13 D4
YES
D4 SUBSTITUTE SEEK (07) FOR 00 OP CODE
E4 DEDICATED DASD DEVICE
YES
NO
17 E5
22F2
22F5
DEDDSEEK
E5 EITHER COMMAND- OR DATA- CHAINED
YES
10 B3
NO
F5 FLAG VDEVBLOK FOR STAND ALONE SEEK
10 B3
DASDX7
F4 DASDX7 - DASD LAST 4 BITS = 7
G4 SEEK (07), RESTORE (17), OR INVALID
>17
13 D4
=07
18 A1
=17
HANDLE RESTORE (17) OR RECALIBRATE (13) COMMAND:
15 B4
18 A1
17G4
19B2
19C2
DASDSEEK
A1 IS BYTE COUNT = 6
NO
13 D4
YES
B1 CNTRLSUB GET THE SEEK BYTES
C1 EITHER COMMAND- OR DATA- CHAINED
YES
NO
D1 FLAG VDEVBLOK FOR STAND ALONE SEEK
E1 HAVE WE ALREADY HAD A SEEK
YES
RELCCHH
B2 CCWRELCH RELOCATE CYLINDER AND HEAD
14 D1
NO
F1 FLAG THAT WE'VE HAD A SEEK (NO LONGER NEEDED)
G1 CCWRELCH RELOCATE CYLINDER NUMBER
H1 WAS VIRTUAL CYLINDER NUMBER VALID
NO
14 D1
YES
J1 STORE VALID "IOBCYL" IN IOBLOK
K1 IS RELOCATION FACTOR = 0 ETC.
YES
NO
14 D1
K2 WAS VIRTUAL HEAD NUMBER VALID
YES
14 B2
NO
14 D1
DASDX9
A4 DASDX9 - DASD LAST 4 BITS = 9
B4 PUTSEEK INSERT A SEEK COMMAND (IF NEEDED)
C4 WRITE HOME ADDR (19), OR SEARCH ID. (>19)
>19
<19
13 D4
=19
DASDSR39
C3 FLAG AS STATUS MODIFIER TYPE OF COMMAND
D3 WAS CCW SEARCH HOME ADDRESS EQUAL
NO
10 B3
YES
E3 IS RELOCATION FACTOR = 0 ETC.
YES
10 B3
NO
F3 CNTRLSUB GET THE ARGUMENT BYTES
G3 BOTTOM HALF OF SIMULATED 2311
NO
14 C1
YES
H3 RELOCATE HEAD NUMBER (BY TEN) FIRST
14 C1
D4 DASD DEVICE READ ONLY
YES
13 C5
NO
E4 IS IT 2311, 2314, OR 2319
YES
NO
F4 FOR 3330, SUBSTITUTE READ HOME ADDRESS
18 G4
28C1
SETSKIP
G4 SET SKIP FLAG AND USE 0 ADDRESS
15 K1
RELHOMAD
E5 IS RELOCATION FACTOR = 0 ETC.
YES
10 B3
NO
F5 CNTRLSUB GET THE ARGUMENT BYTES
G5 RELOCATE CYLINDER NO. (2ND & 3RD BYTES)
H5 BOTTOM HALF OF SIMULATED 2311
NO
14 D1
YES
J5 ALSO RELOCATE THE HEAD NUMBER (BY TEN)
14 D1

| DMKCCW -- Translate a Virtual CCW List to a Real List (Parts 17 and 18 of 31)

| DMKCCW -- Translate a Virtual CCW List to a Real List (Parts 19 and 20 of 31)

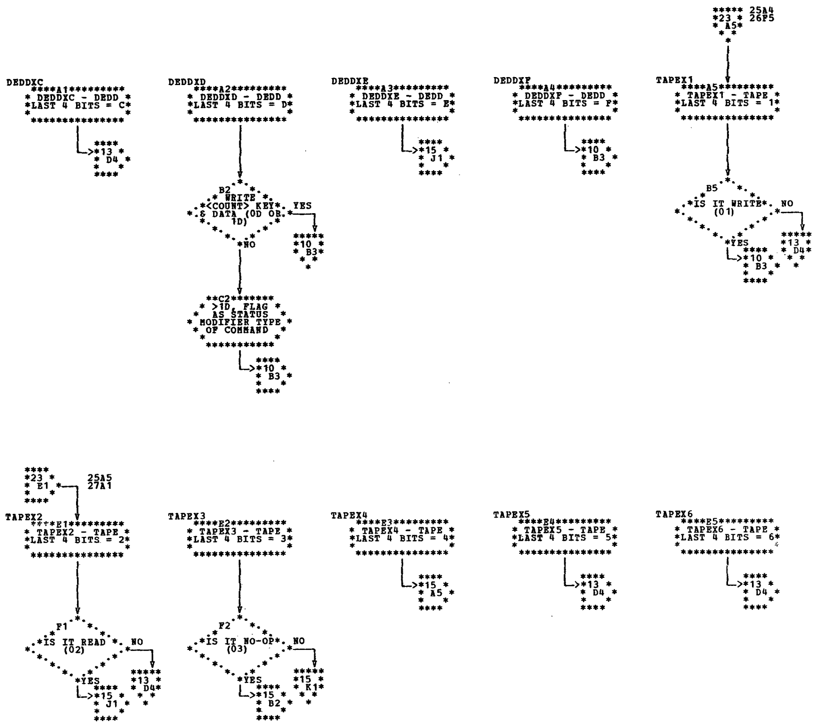


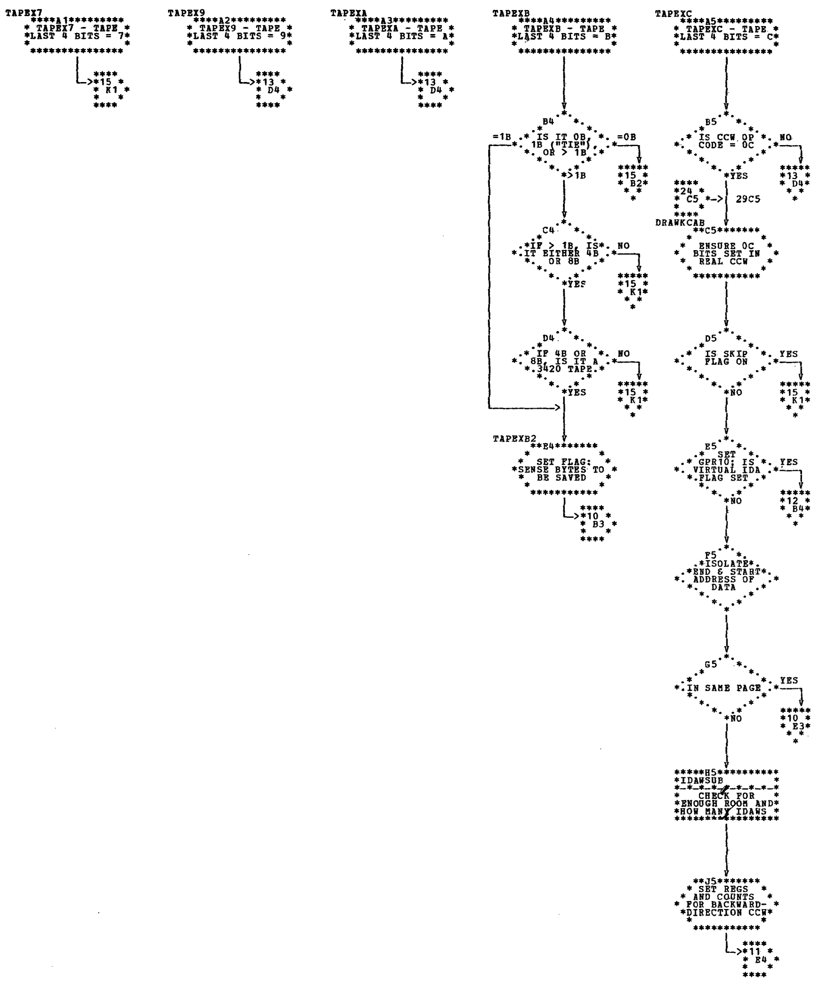


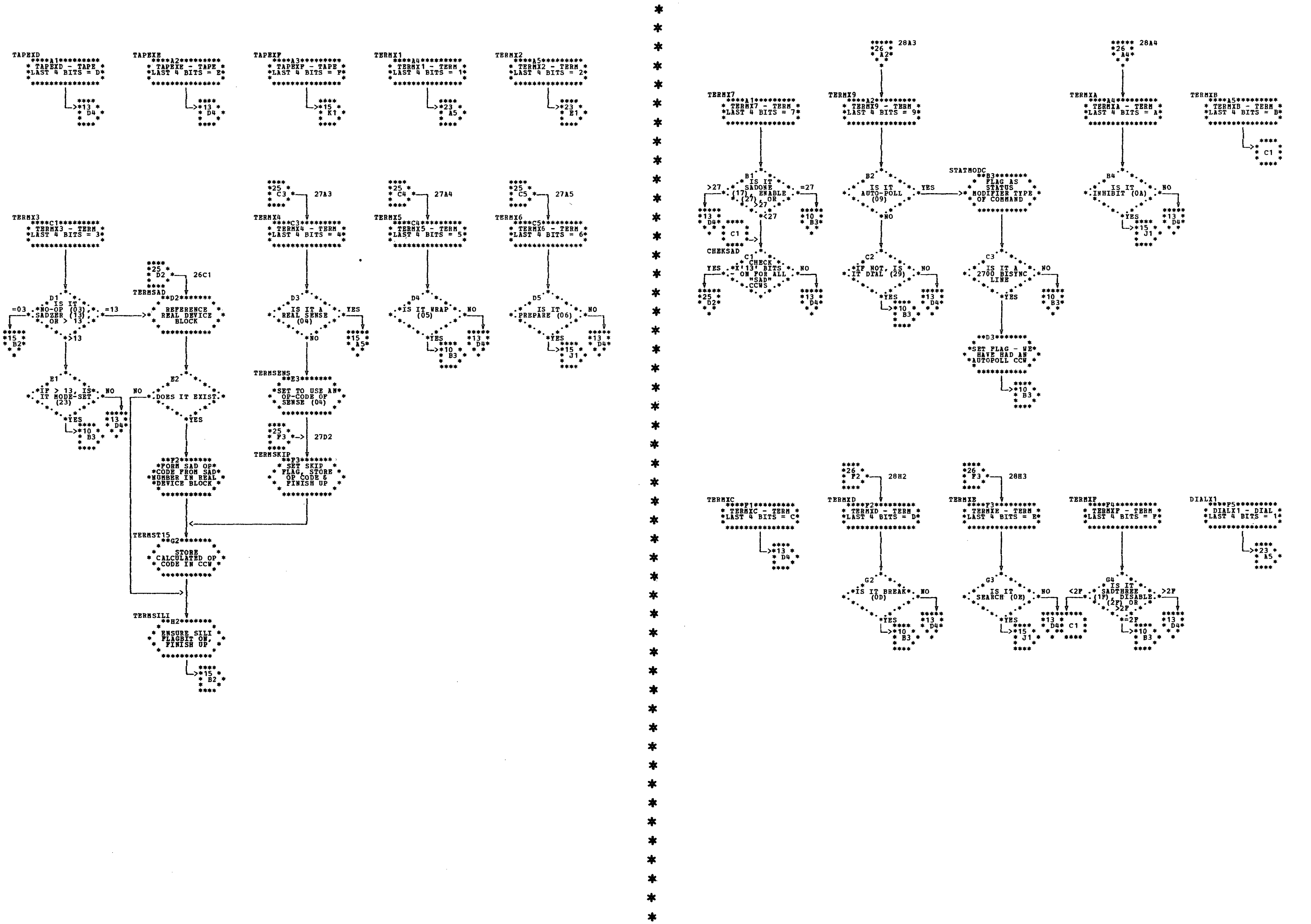

20K3
21 A2
CCWRLC1
A2 STORE RELOCATED CYLINDER NO. IN SEEK ARGS
A2
B2 SIMULATED 2311
NO
YES
B3 RETURN - R10
CCWRLC4
C1 SET HEAD NUMBER INVALID = 255
C2 VIRTUAL HEAD NUMBER VALID (9 OR LESS)
NO
YES
D1 AND REMEMBER HEAD NUMBER WAS INVALID
D2 BOTTOM HALF OF SIMULATED 2311
NO
YES
D3 RETURN - R10
E2 RELOCATE HEAD NUMBER (ADD TEN)
CCWRLC2
F2 STORE RELOCATED HEAD NUMBER
G2 RETURN - R10
CCWRELCH
A4 CCWRELCH - R10
B4 DOES SEEK ARG EXCEED MINIDISK RANGE
YES
NO
CCWRLC3
B5 SET CYLINDER NUMBER INVALID = 4095
A2
C4 WAS THE CCW A SEEK HEAD (1B)
YES
NO
20 K3
D4 STORE VIRTUAL CYLINDER NO. IN VDEVPOSN IN VDEVBLOK
20 K3
29A2
22 A3
DEDDX1
A1 DEDDX1 - DEDD LAST 4 BITS = 1
B1 IS IT WRITE (01), ERASE (11), OR OTHER (>11)
=11
=01
>11
10 B3
10 B3
C1 >11, FLAG AS STATUS MODIFIER TYPE OF COMMAND
10 B3
DEDDX2
A2 DEDDX2 - DEDD LAST 4 BITS = 2
15 J1
DEDDX3
A3 DEDDX3 - DEDD LAST 4 BITS = 3
B3 IS IT A NO-OP (03)
NO
YES
15 B2
10 B3
DEDDX4
A4 DEDDX4 - DEDD LAST 4 BITS = 4
15 A5
DEDDX5
A5 DEDDX5 - DEDD LAST 4 BITS = 5
B5 EITHER WRITE DATA (05) OR WRITE RECORD R0 (15)
YES
NO
10 B3
C5 >15, FLAG AS STATUS MODIFIER TYPE OF COMMAND
10 B3
DEDDX6
E1 DEDDX6 - DEDD LAST 4 BITS = 6
F1 IS SKIP FLAG SET
YES
NO
15 K1
G1 IS IT A READ DATA (06)
NO
YES
17 E1
10 B3
DEDDX7
E2 DEDDX7 - DEDD LAST 4 BITS = 7
F2 IS THE CCW A SEEK (07)
YES
NO
10 B3
17 E5
DEDDX9
E3 DEDDX9 - DEDD LAST 4 BITS = 9
F3 IS IT WRITE HOME ADDRESS (19)
YES
NO
10 B3
G3 FLAG AS STATUS MODIFIER TYPE OF COMMAND
10 B3
DEDDXA
E4 DEDDXA - DEDD LAST 4 BITS = A
15 J1
DEDDXB
E5 DEDDXB - DEDD LAST 4 BITS = B
F5 EITHER SEEK HEAD OR SEEK CYL
NO
YES
17 E5
10 B3

| DMKCCW -- Translate a Virtual CCW List to a Real List (Parts 21 and 22 of 31)

| DMKCCW -- Translate a Virtual CCW List to a Real List (Parts 23 and 24 of 31)

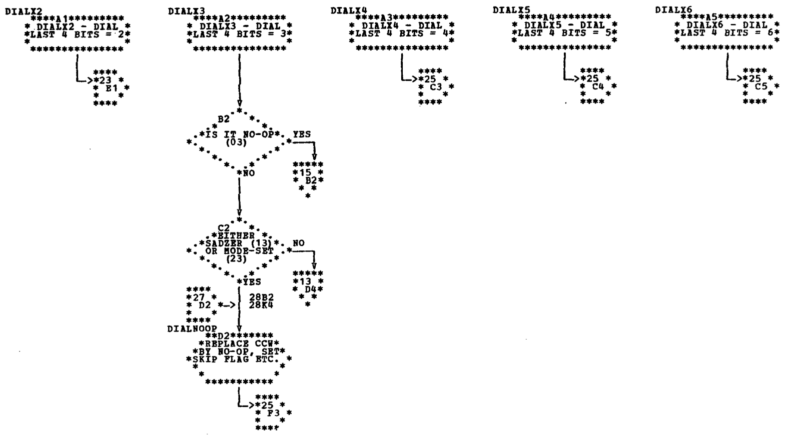


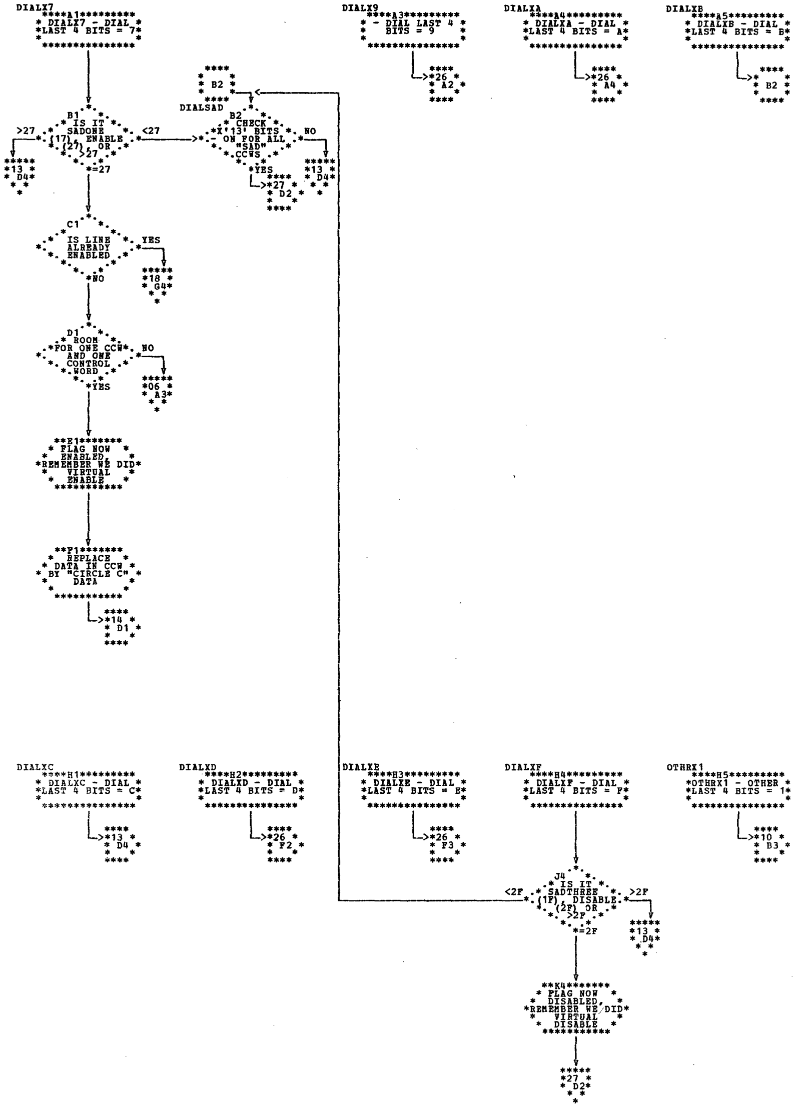


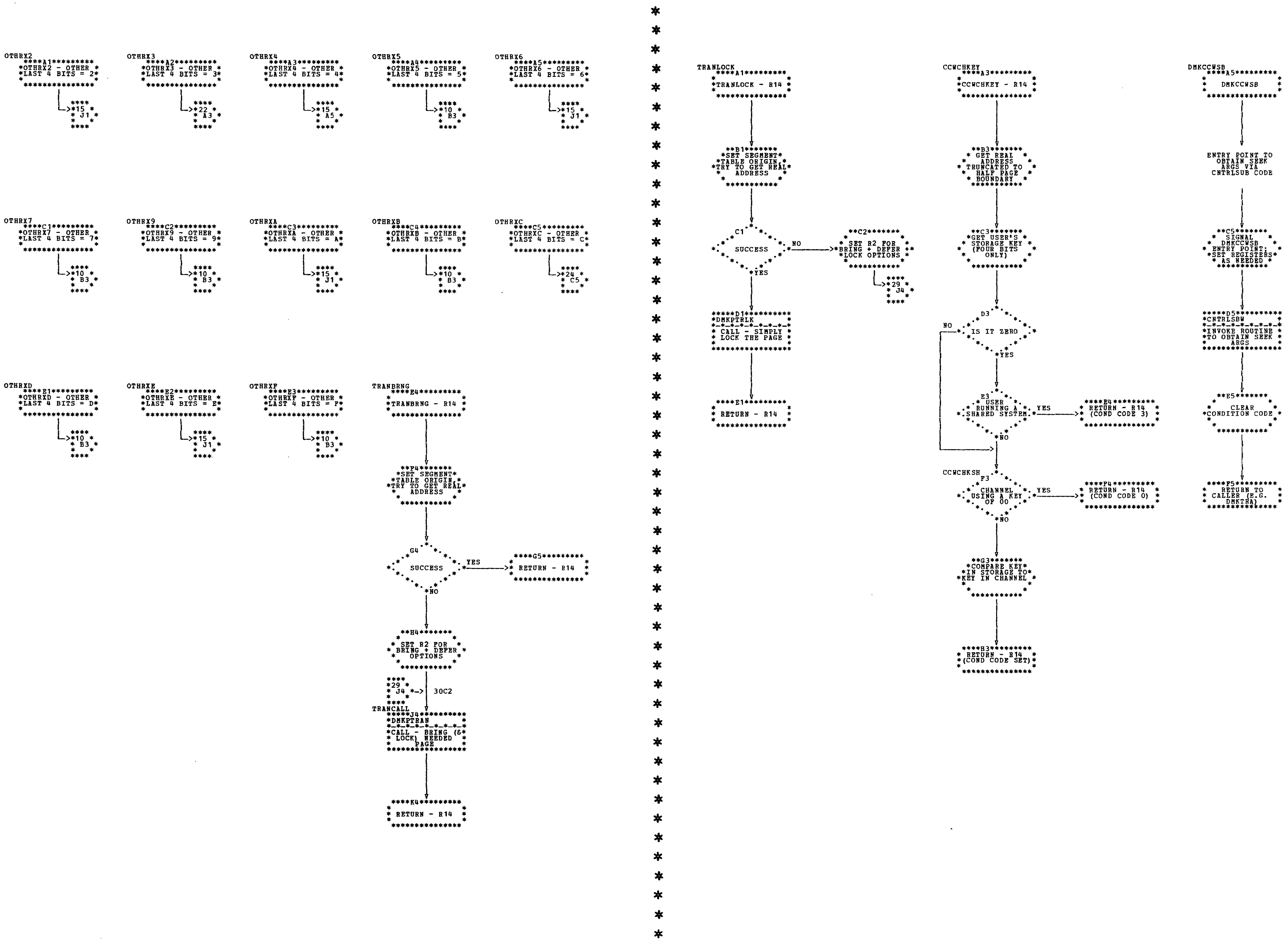


| DMKCCW -- Translate a Virtual CCW List to a Real List (Parts 25 and 26 of 31)

| DMKCCW -- Translate a Virtual CCW List to a Real List (Parts 27 and 28 of 31)

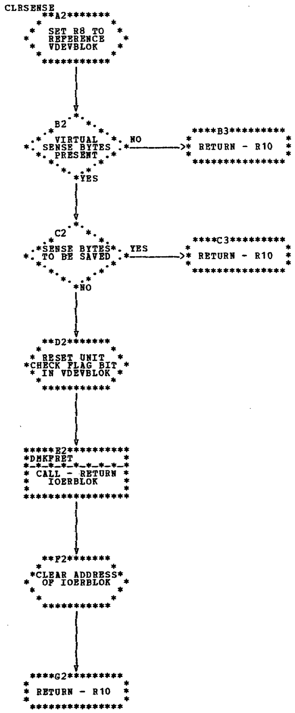

DIALX2
A1
DIALX2 - DIAL LAST 4 BITS = 2
23 E1
DIALX3
A2
DIALX3 - DIAL LAST 4 BITS = 3
B2
IS IT NO-OP (03)
YES
15 B2
NO
C2
EITHER SADZER (13) OR MODE-SET (23)
NO
13 D4
YES
27 D2
28B2
28K4
DIALNOOP
D2
REPLACE CCW BY NO-OP, SET SKIP FLAG ETC.
25 F3
DIALX4
A3
DIALX4 - DIAL LAST 4 BITS = 4
25 C3
DIALX5
A4
DIALX5 - DIAL LAST 4 BITS = 5
25 C4
DIALX6
A5
DIALX6 - DIAL LAST 4 BITS = 6
25 C5

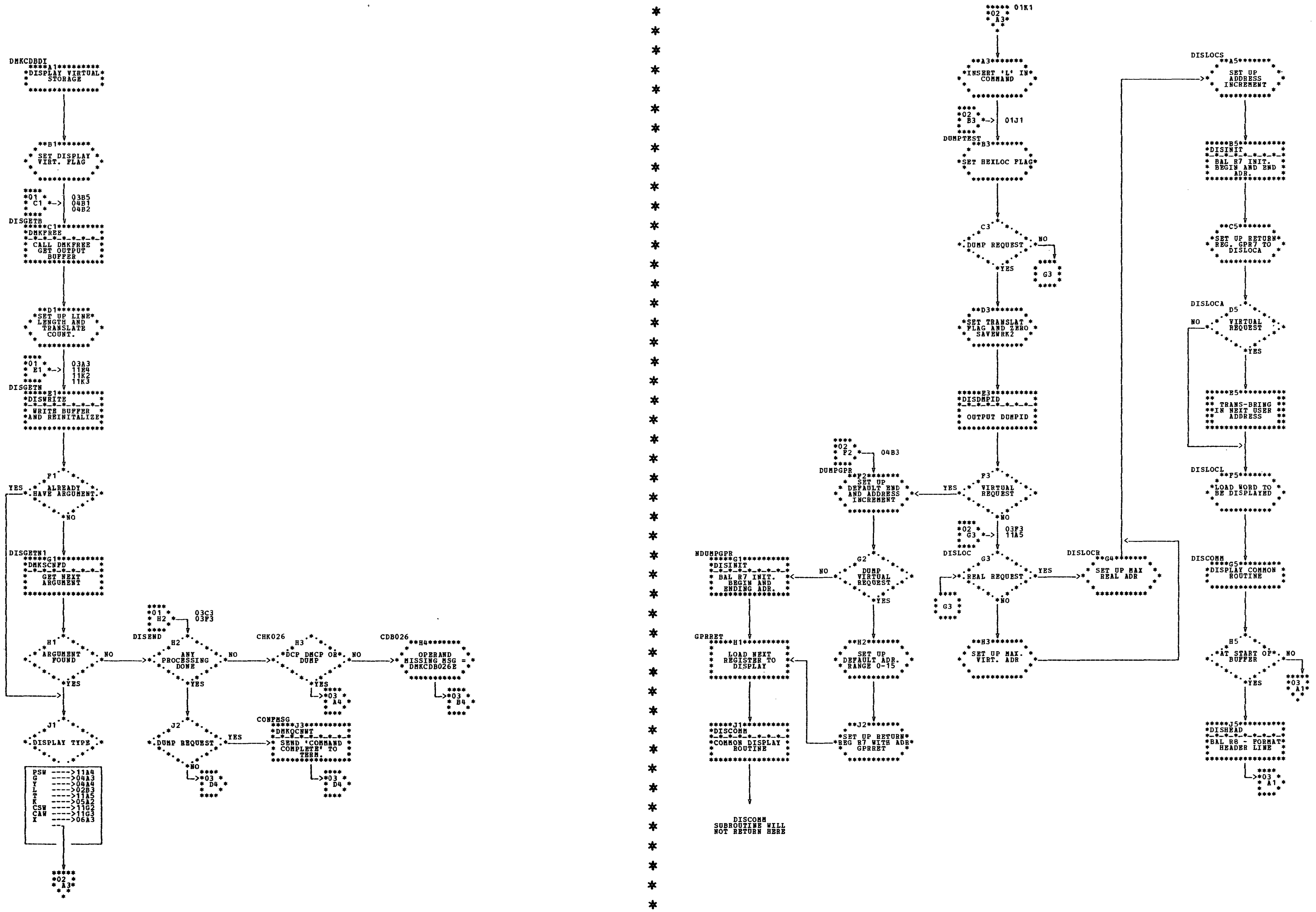

DIALX7
A1
DIALX7 - DIAL LAST 4 BITS = 7
B1
IS IT SADONE (17), ENABLE (27), OR >27
>27
13 D4
<27
=27
B2
DIALSAD
B2
CHECK X'13' BITS ON FOR ALL "SAD" CCWS
NO
13 D4
YES
27 D2
C1
IS LINE ALREADY ENABLED
YES
18 G4
NO
D1
ROOM FOR ONE CCW AND ONE CONTROL WORD
NO
06 A3
YES
E1
FLAG NOW ENABLED, REMEMBER WE DID VIRTUAL ENABLE
F1
REPLACE DATA IN CCW BY "CIRCLE C" DATA
14 D1
DIALX9
A3
- DIAL LAST 4 BITS = 9
26 A2
DIALXA
A4
DIALXA - DIAL LAST 4 BITS = A
26 A4
DIALXB
A5
DIALXB - DIAL LAST 4 BITS = B
B2
DIALXC
H1
DIALXC - DIAL LAST 4 BITS = C
13 D4
DIALXD
H2
DIALXD - DIAL LAST 4 BITS = D
26 F2
DIALXE
H3
DIALXE - DIAL LAST 4 BITS = E
26 F3
DIALXF
H4
DIALXF - DIAL LAST 4 BITS = F
J4
IS IT SADTHREE (1F), DISABLE (2F) OR >2F
<2F
>2F
13 D4
=2F
K4
FLAG NOW DISABLED, REMEMBER WE DID VIRTUAL DISABLE
27 D2
OTHRX1
H5
OTHRX1 - OTHER LAST 4 BITS = 1
10 B3


OTHRX2
A1
OTHRX2 - OTHER LAST 4 BITS = 2
15 J1
OTHRX3
A2
OTHRX3 - OTHER LAST 4 BITS = 3
22 A3
OTHRX4
A3
OTHRX4 - OTHER LAST 4 BITS = 4
15 A5
OTHRX5
A4
OTHRX5 - OTHER LAST 4 BITS = 5
10 B3
OTHRX6
A5
OTHRX6 - OTHER LAST 4 BITS = 6
15 J1
OTHRX7
C1
OTHRX7 - OTHER LAST 4 BITS = 7
10 B3
OTHRX9
C2
OTHRX9 - OTHER LAST 4 BITS = 9
10 B3
OTHRXA
C3
OTHRXA - OTHER LAST 4 BITS = A
15 J1
OTHRXB
C4
OTHRXB - OTHER LAST 4 BITS = B
10 B3
OTHRXC
C5
OTHRXC - OTHER LAST 4 BITS = C
24 C5
OTHRXD
E1
OTHRXD - OTHER LAST 4 BITS = D
10 B3
OTHRXE
E2
OTHRXE - OTHER LAST 4 BITS = E
15 J1
OTHRXF
E3
OTHRXF - OTHER LAST 4 BITS = F
10 B3
TRANBRNG
E4
TRANBRNG - R14
F4
SET SEGMENT TABLE ORIGIN, TRY TO GET REAL ADDRESS
G4
SUCCESS
YES
G5
RETURN - R14
NO
H4
SET R2 FOR BRING + DEFER OPTIONS
29 J4
30C2
TRANCALL
J4
DMKPTRAN
CALL - BRING (& LOCK) NEEDED PAGE
K4
RETURN - R14
TRANLOCK
A1
TRANLOCK - R14
B1
SET SEGMENT TABLE ORIGIN, TRY TO GET REAL ADDRESS
C1
SUCCESS
NO
C2
SET R2 FOR BRING + DEFER LOCK OPTIONS
29 J4
YES
D1
DMKPTRLK
CALL - SIMPLY LOCK THE PAGE
E1
RETURN - R14
CCWCHKEY
A3
CCWCHKEY - R14
B3
GET REAL ADDRESS TRUNCATED TO HALF PAGE BOUNDARY
C3
GET USER'S STORAGE KEY (FOUR BITS ONLY)
D3
IS IT ZERO
NO
YES
E3
USER RUNNING A SHARED SYSTEM
YES
E4
RETURN - R14 (COND CODE 3)
NO
CCWCHKSH
F3
CHANNEL USING A KEY OF 00
YES
F4
RETURN - R14 (COND CODE 0)
NO
G3
COMPARE KEY IN STORAGE TO KEY IN CHANNEL
H3
RETURN - R14 (COND CODE SET)
DMKCCWSB
A5
DMKCCWSB
ENTRY POINT TO OBTAIN SEEK ARGS VIA CNTRLSUB CODE
C5
SIGNAL DMKCCWSB ENTRY POINT; SET REGISTERS AS NEEDED
D5
CNTRLSBW
INVOKE ROUTINE TO OBTAIN SEEK ARGS
E5
CLEAR CONDITION CODE
F5
RETURN TO CALLER (E.G. DMKTRA)

| DMKCCW -- Translate a Virtual CCW List to a Real List (Parts 29 and 30 of 31)

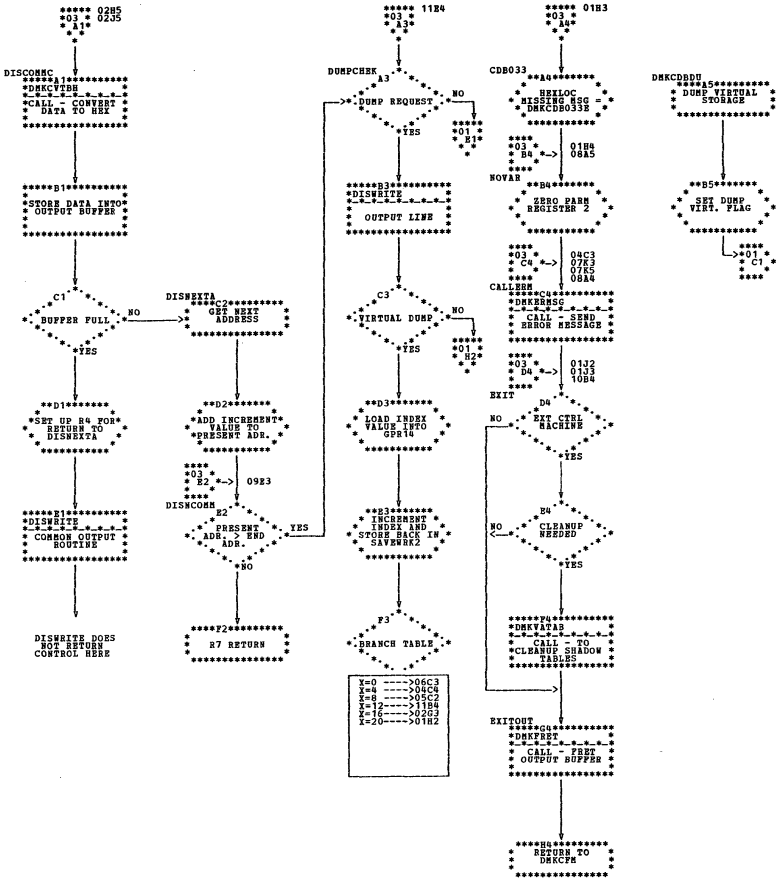
DMKCCW -- Translate a Virtual CCW List to a Real List (Part 31 of 31)

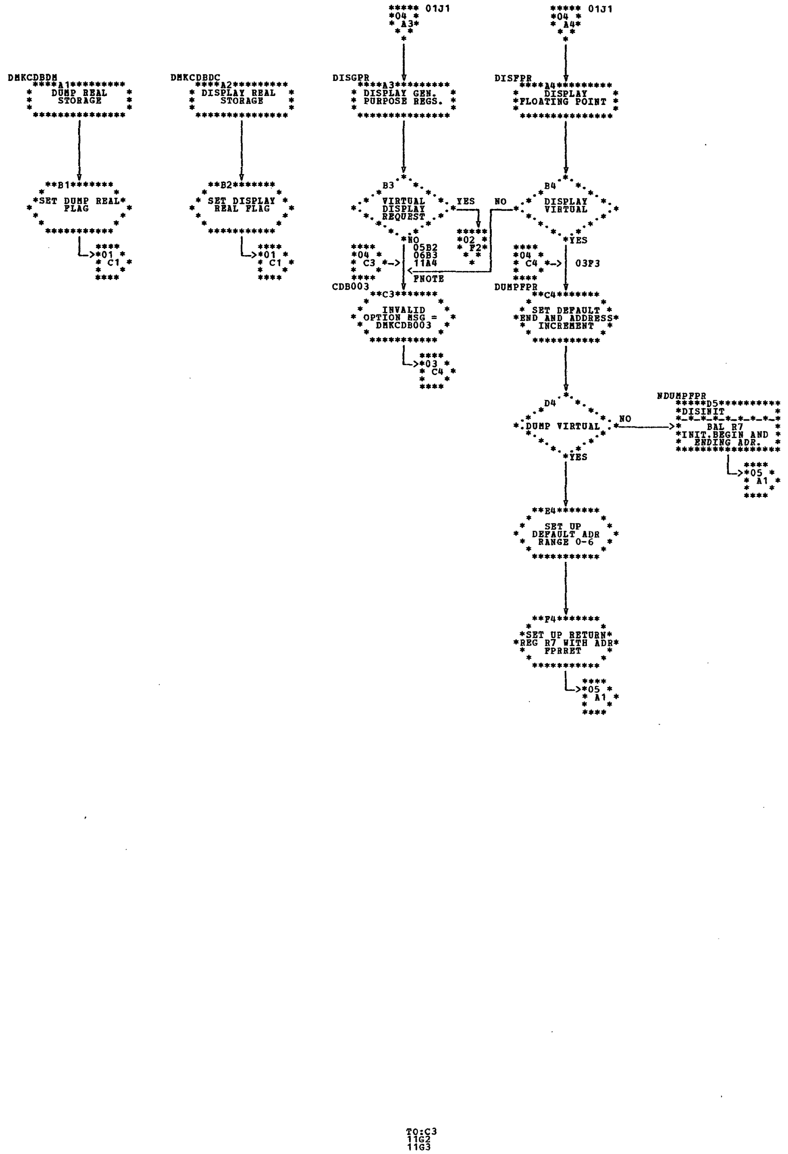


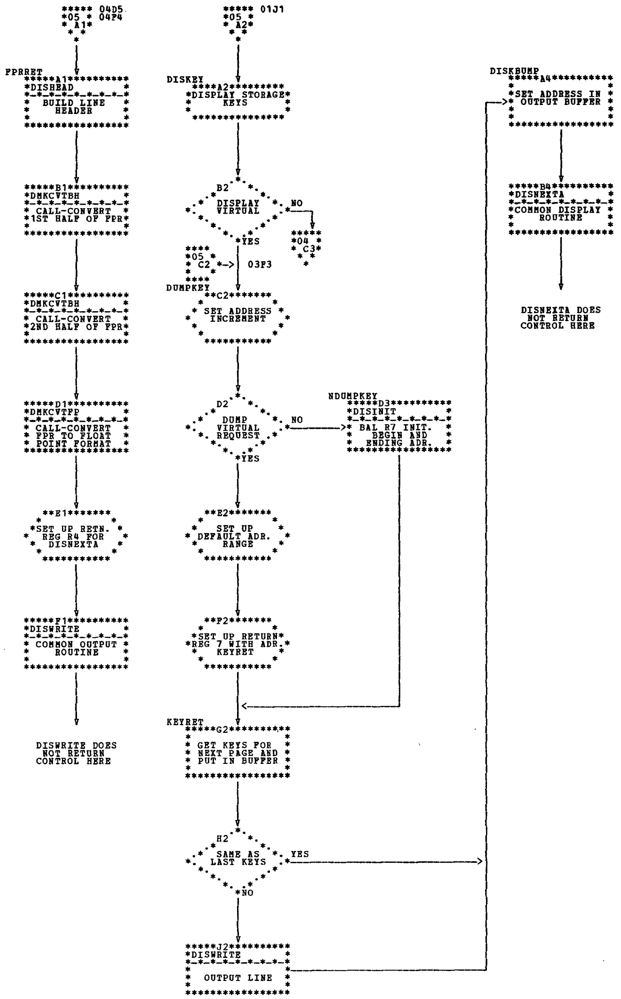


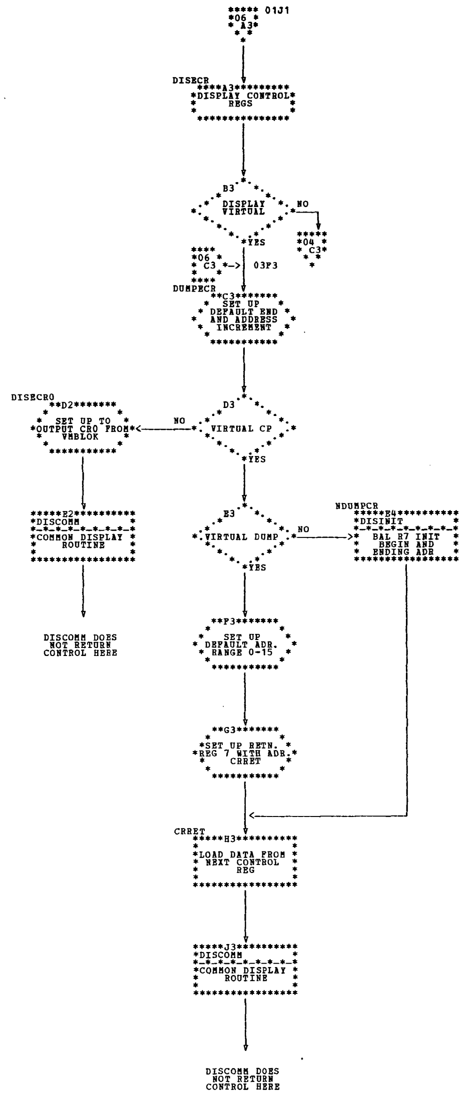
DMKCDB -- Process DCP, DISPLAY, DMCP and DUMP commands (Parts 1 and 2 of 11)

DMKCDB -- Process DCP, DISPLAY, DMCP and DUMP Commands (Parts 3 and 4 of 11)



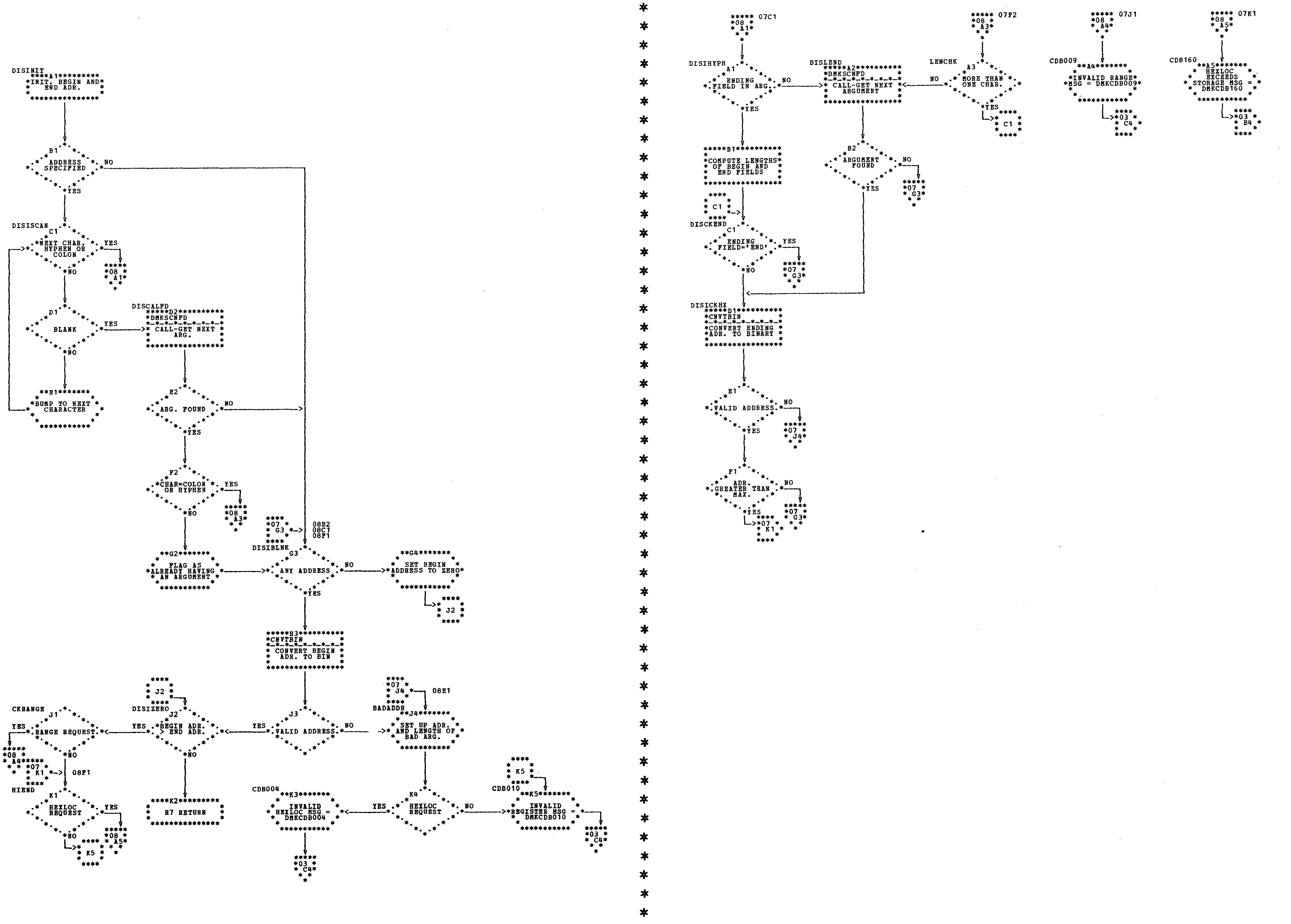


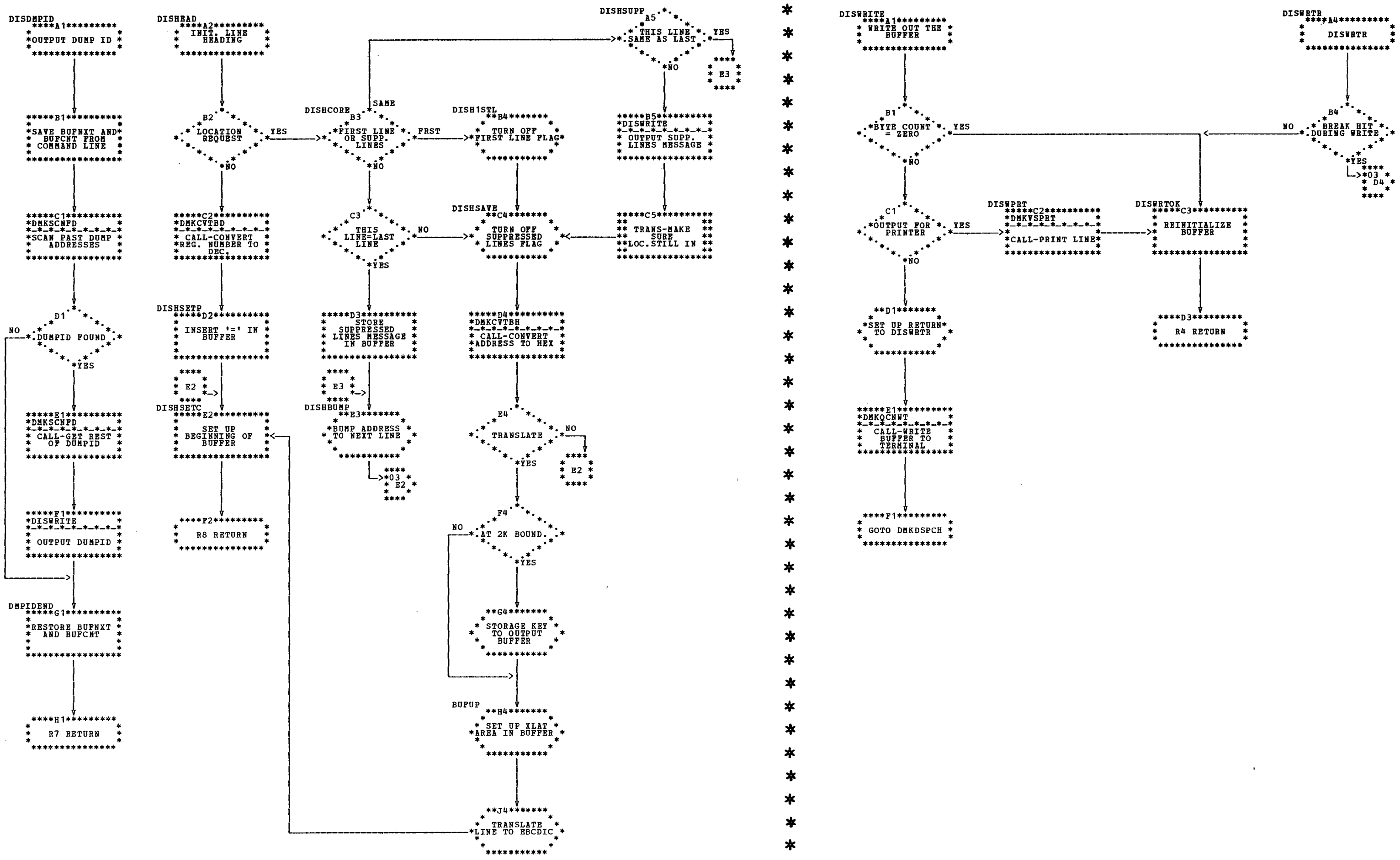

05 A1
04D5
04F4
05 A2
01J1
FPRRET
A1
DISHEAD
BUILD LINE HEADER
B1
DMKCVTBH
CALL-CONVERT 1ST HALF OF FPR
C1
DMKCVTBH
CALL-CONVERT 2ND HALF OF FPR
D1
DMKCVTFP
CALL-CONVERT FPR TO FLOAT POINT FORMAT
E1
SET UP RETN. REG R4 FOR DISNEXTA
F1
DISWRITE
COMMON OUTPUT ROUTINE
DISWRITE DOES NOT RETURN CONTROL HERE
DISKEY
A2
DISPLAY STORAGE KEYS
B2
DISPLAY VIRTUAL
NO
04 C3
YES
05 C2
03F3
DUMPKEY
C2
SET ADDRESS INCREMENT
D2
DUMP VIRTUAL REQUEST
NO
YES
NDUMPKEY
D3
DISINIT
BAL R7 INIT. BEGIN AND ENDING ADR.
E2
SET UP DEFAULT ADR. RANGE
F2
SET UP RETURN REG 7 WITH ADR. KEYRET
KEYRET
G2
GET KEYS FOR NEXT PAGE AND PUT IN BUFFER
H2
SAME AS LAST KEYS
YES
NO
J2
DISWRITE
OUTPUT LINE
DISKBUMP
A4
SET ADDRESS IN OUTPUT BUFFER
B4
DISNEXTA
COMMON DISPLAY ROUTINE
DISNEXTA DOES NOT RETURN CONTROL HERE


06 A3
01J1
DISECR
A3
DISPLAY CONTROL REGS
B3
DISPLAY VIRTUAL
NO
04 C3
YES
06 C3
03F3
DUMPECR
C3
SET UP DEFAULT END AND ADDRESS INCREMENT
D3
VIRTUAL CP
NO
YES
DISECR0
D2
SET UP TO OUTPUT CR0 FROM VMBLOK
E2
DISCOMM
COMMON DISPLAY ROUTINE
DISCOMM DOES NOT RETURN CONTROL HERE
E3
VIRTUAL DUMP
NO
YES
NDUMPCR
E4
DISINIT
BAL R7 INIT BEGIN AND ENDING ADR
F3
SET UP DEFAULT ADR. RANGE 0-15
G3
SET UP RETN. REG 7 WITH ADR. CRRET
CRRET
H3
LOAD DATA FROM NEXT CONTROL REG
J3
DISCOMM
COMMON DISPLAY ROUTINE
DISCOMM DOES NOT RETURN CONTROL HERE

DMKCDB -- Process DCP, DISPLAY, DMCP and DUMP Commands (Parts 5 and 6 of 11)

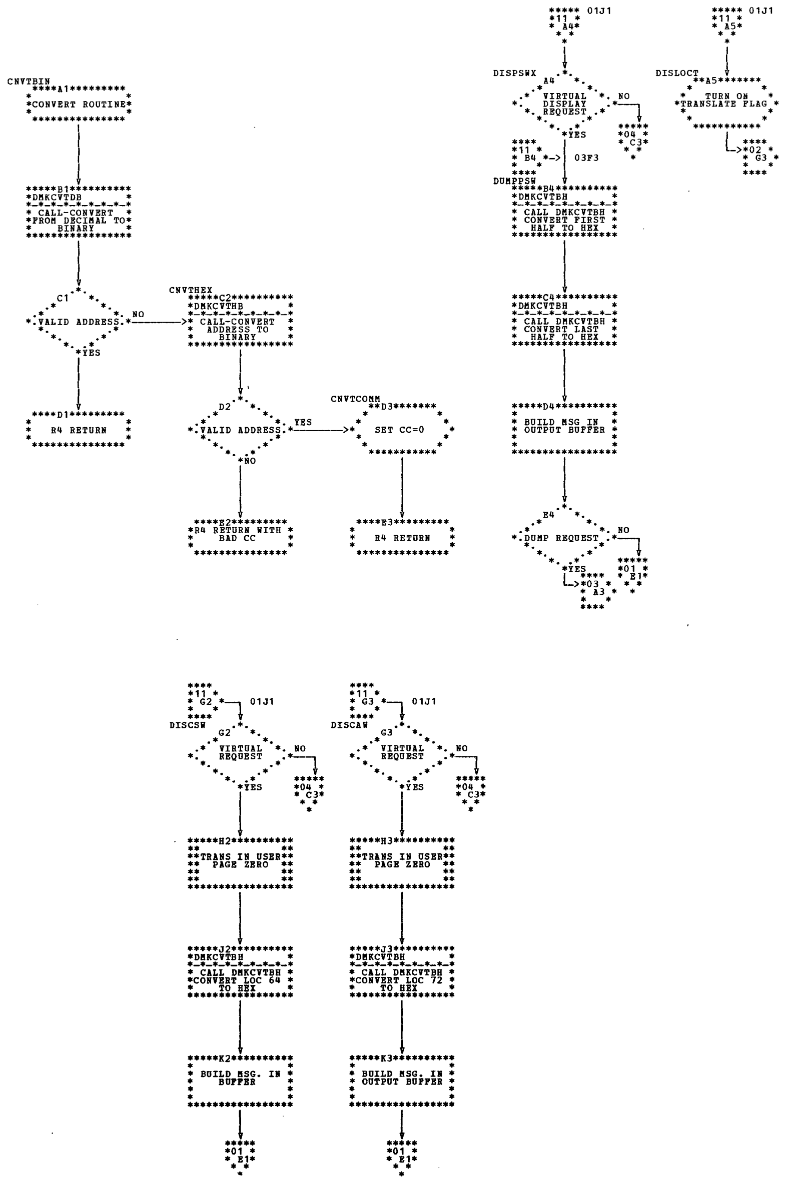
DMKCDB -- Process DCP, DISPLAY, DMCP and DUMP Commands (Parts 7 and 8 for 11)

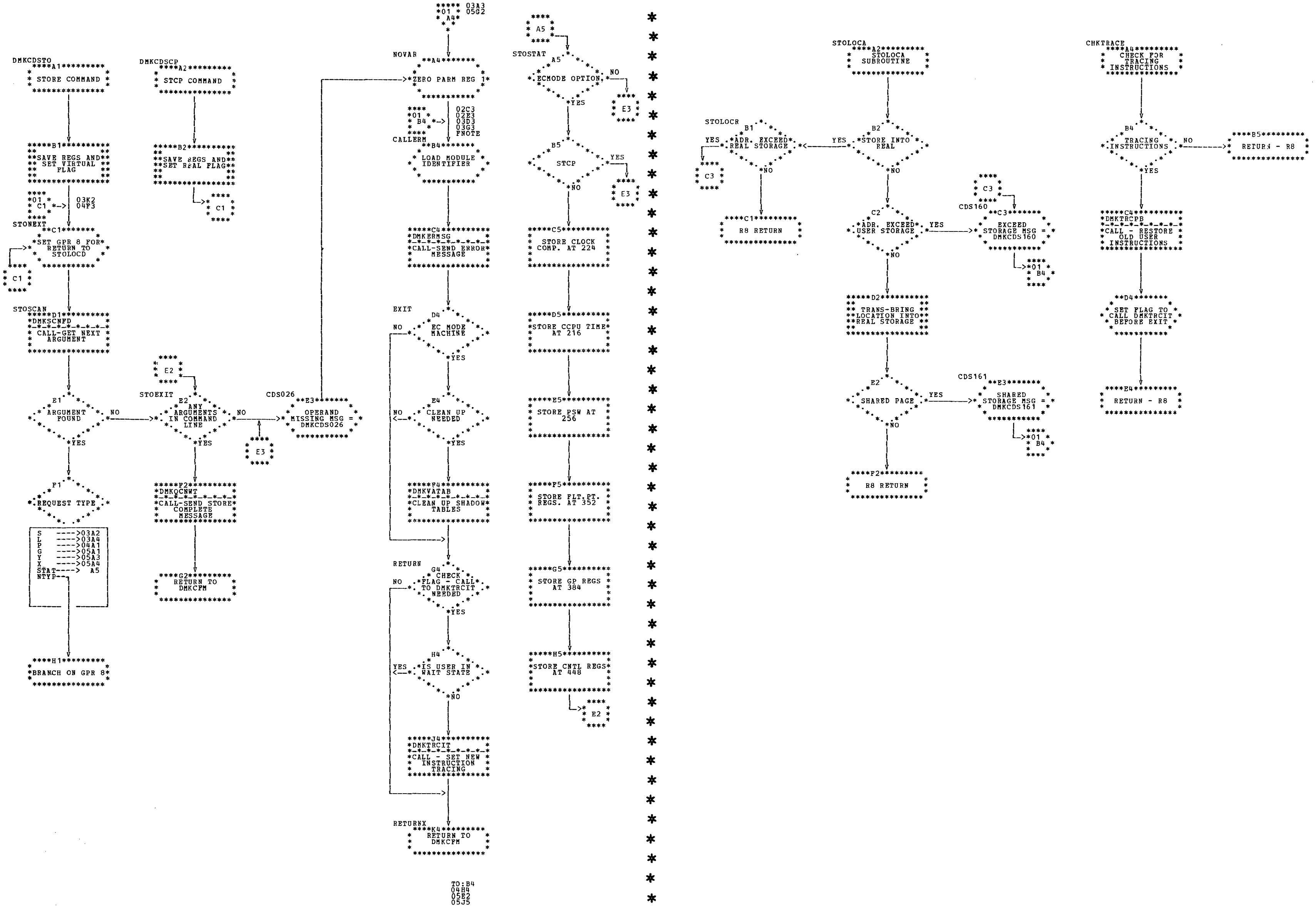




DMKCDB -- Process DCP, DISPLAY, DMCP and DUMP Commands (Parts 9 and 10 of 11)

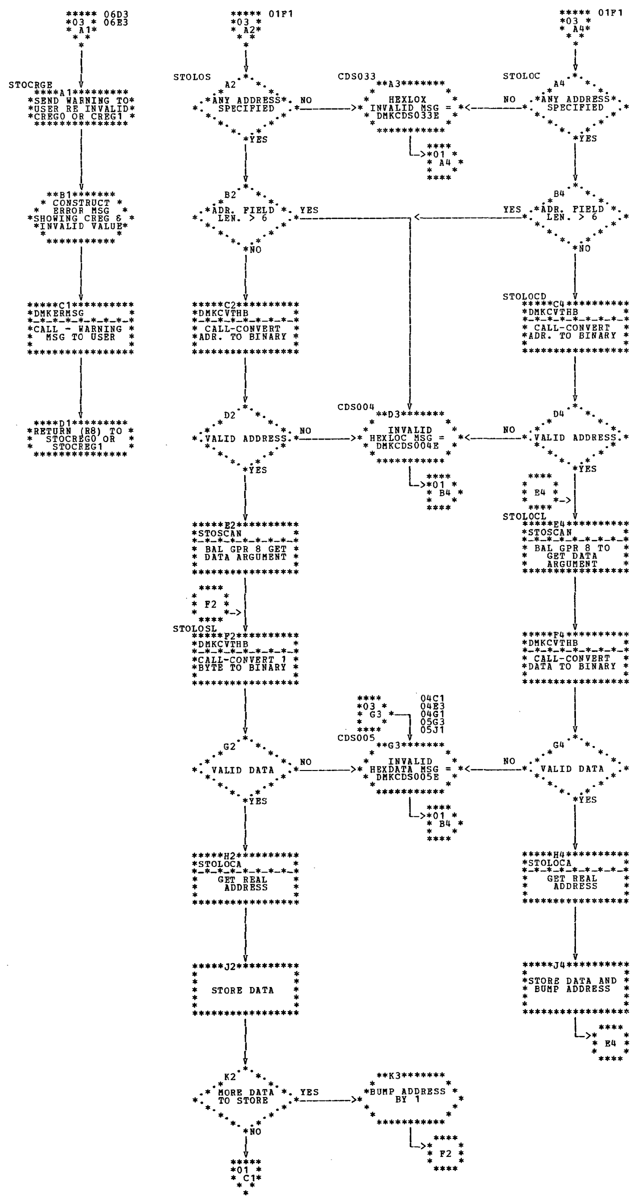
DMKCDB -- Process DCP, DISPLAY, DMCP and DUMP Commands (Part 11 of 11)

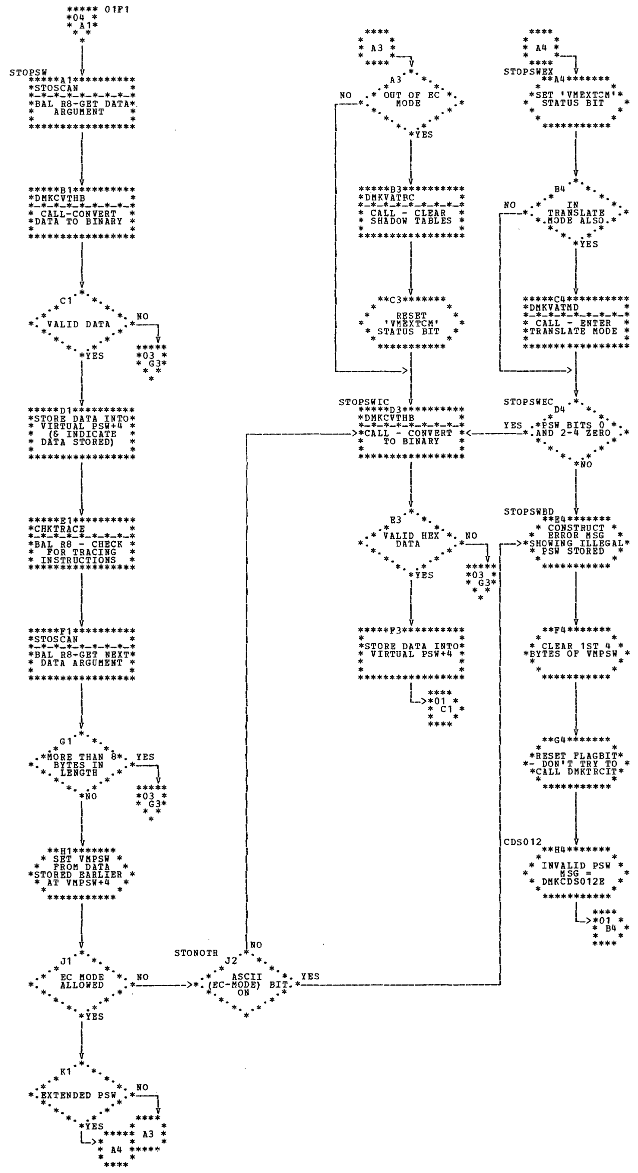


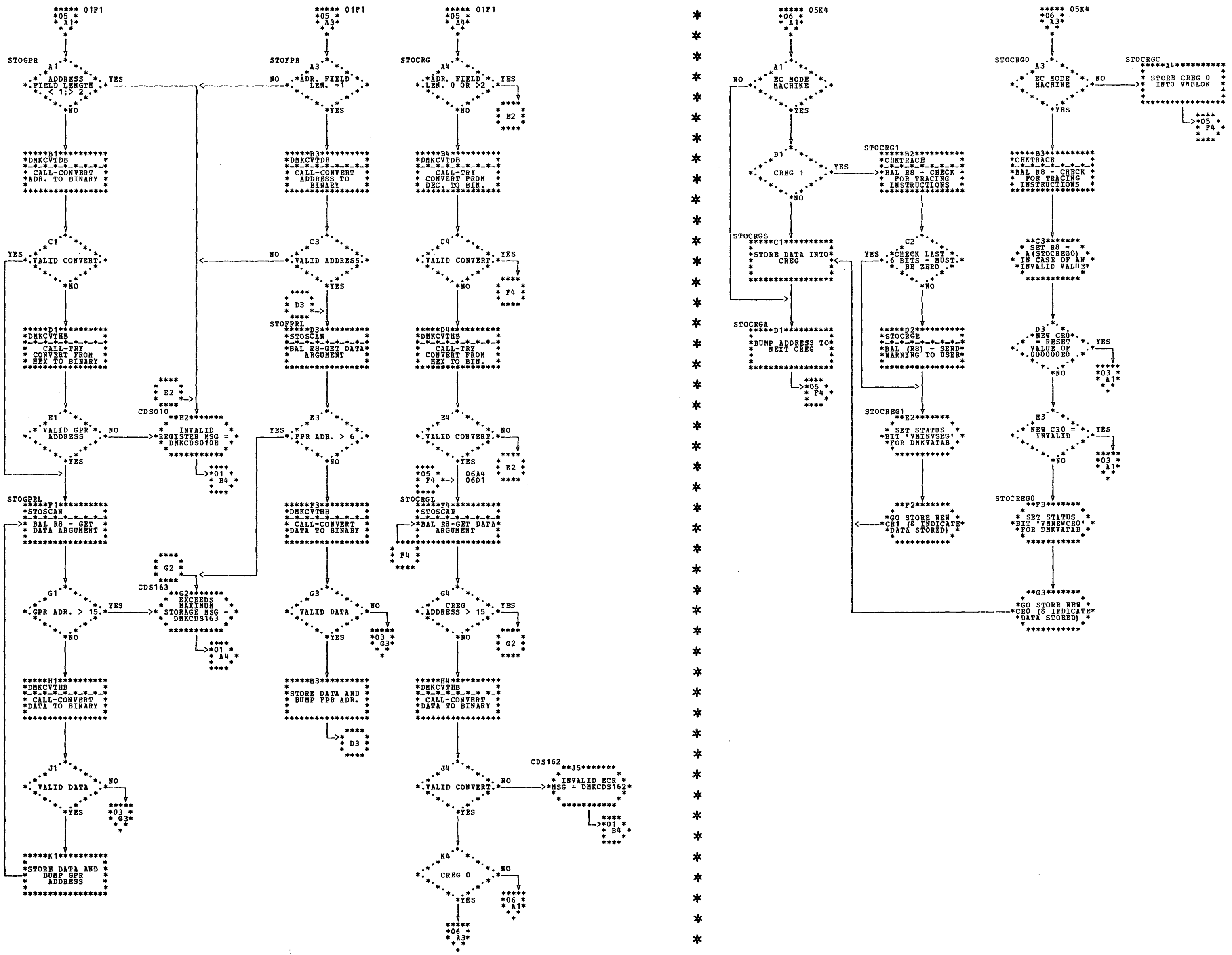


| DMKCDS -- Process STCP and STORE Commands (Parts 1 and 2 of 6)

DMKCDS -- Process STCP and STORE Commands (Parts 3 and 4 of 6)

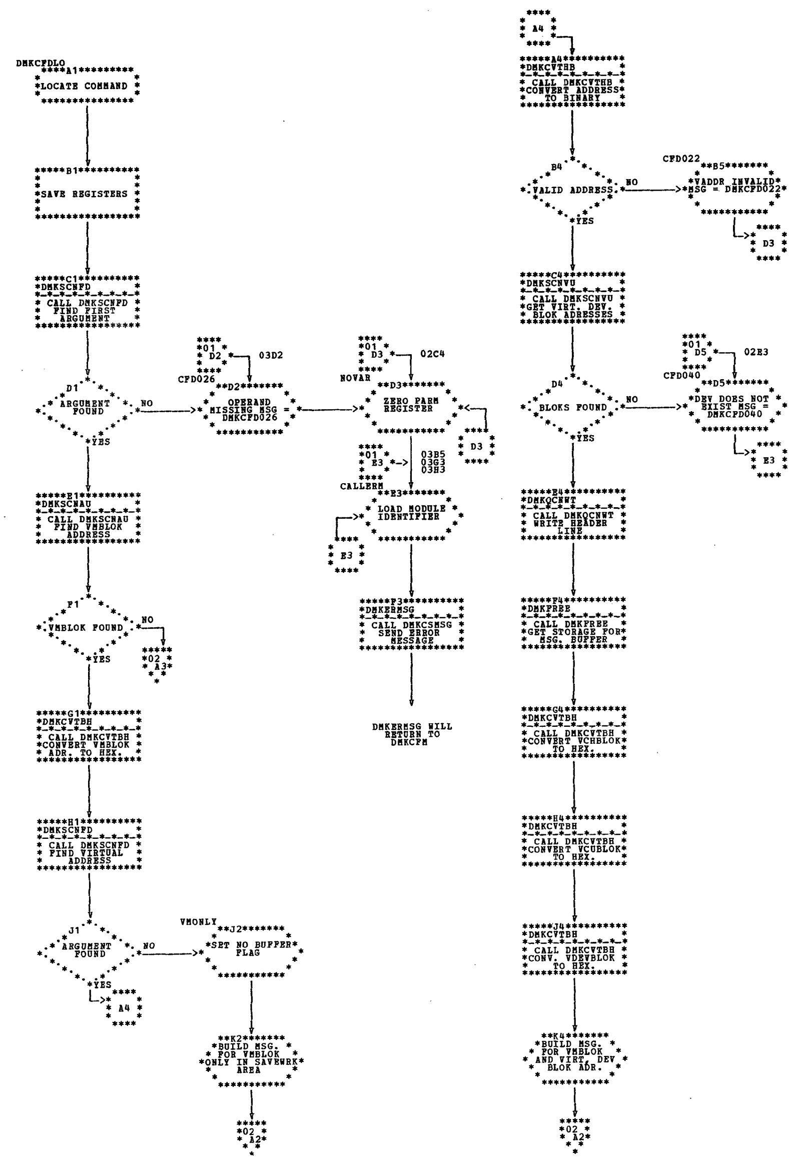

03 A1 06D3 06E3
STOCRGE
A1 SEND WARNING TO USER RE INVALID CREG0 OR CREG1
B1 CONSTRUCT ERROR MSG SHOWING CREG & INVALID VALUE
C1 DMKERMSG CALL - WARNING MSG TO USER
D1 RETURN (R8) TO STOCREG0 OR STOCREG1
03 A2 01F1
STOLOS
A2 ANY ADDRESS SPECIFIED
NO
YES
CDS033
A3 HEXLOX INVALID MSG = DMKCDS033E
01 A4
03 A4 01F1
STOLOC
A4 ANY ADDRESS SPECIFIED
NO
YES
B2 ADR. FIELD LEN. > 6
YES
NO
B4 ADR. FIELD LEN. > 6
YES
NO
C2 DMKCVTHB CALL-CONVERT ADR. TO BINARY
STOLOCD
C4 DMKCVTHB CALL-CONVERT ADR. TO BINARY
D2 VALID ADDRESS
NO
YES
CDS004
D3 INVALID HEXLOC MSG = DMKCDS004E
01 B4
D4 VALID ADDRESS
NO
YES
E2 STOSCAN BAL GPR 8 GET DATA ARGUMENT
E4
STOLOCL
E4 STOSCAN BAL GPR 8 TO GET DATA ARGUMENT
F2
STOLOSL
F2 DMKCVTHB CALL-CONVERT 1 BYTE TO BINARY
F4 DMKCVTHB CALL-CONVERT DATA TO BINARY
03 G3 04C1 04E3 04G1 05G3 05J1
CDS005
G2 VALID DATA
NO
YES
G3 INVALID HEXDATA MSG = DMKCDS005E
01 B4
G4 VALID DATA
NO
YES
H2 STOLOCA GET REAL ADDRESS
H4 STOLOCA GET REAL ADDRESS
J2 STORE DATA
J4 STORE DATA AND BUMP ADDRESS
E4
K2 MORE DATA TO STORE
YES
NO
K3 BUMP ADDRESS BY 1
F2
01 C1

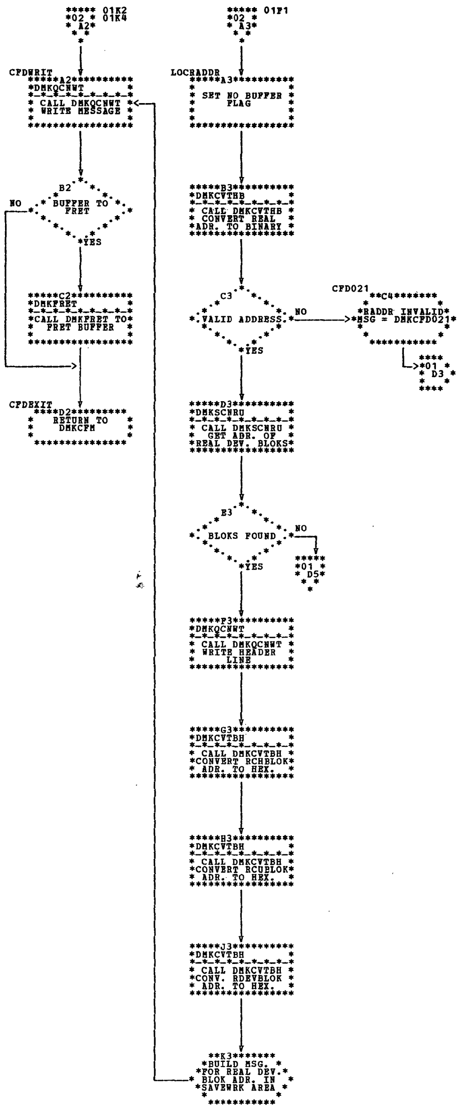

04 A1 01F1
STOPSW
A1 STOSCAN BAL R8-GET DATA ARGUMENT
B1 DMKCVTHB CALL-CONVERT DATA TO BINARY
C1 VALID DATA
NO
YES
03 G3
D1 STORE DATA INTO VIRTUAL PSW+4 (& INDICATE DATA STORED)
E1 CHKTRACE BAL R8 - CHECK FOR TRACING INSTRUCTIONS
F1 STOSCAN BAL R8-GET NEXT DATA ARGUMENT
G1 MORE THAN 8 BYTES IN LENGTH
YES
NO
03 G3
H1 SET VMPSW FROM DATA STORED EARLIER AT VMPSW+4
J1 EC MODE ALLOWED
NO
YES
STONOTR
J2 ASCII (EC-MODE) BIT ON
NO
YES
K1 EXTENDED PSW
NO
YES
A3
A4
A3
A3 OUT OF EC MODE
NO
YES
B3 DMKVATBC CALL - CLEAR SHADOW TABLES
C3 RESET 'VMEXTCM' STATUS BIT
STOPSWIC
D3 DMKCVTHB CALL - CONVERT TO BINARY
E3 VALID HEX DATA
NO
YES
03 G3
F3 STORE DATA INTO VIRTUAL PSW+4
01 C1
A4
STOPSWEX
A4 SET 'VMEXTCM' STATUS BIT
B4 IN TRANSLATE MODE ALSO
NO
YES
C4 DMKVATMD CALL - ENTER TRANSLATE MODE
STOPSWEC
D4 PSW BITS 0 AND 2-4 ZERO
YES
NO
STOPSWBD
E4 CONSTRUCT ERROR MSG SHOWING ILLEGAL PSW STORED
F4 CLEAR 1ST 4 BYTES OF VMPSW
G4 RESET FLAGBIT- DON'T TRY TO CALL DMKTRCIT
CDS012
H4 INVALID PSW MSG = DMKCDS012E
01 B4

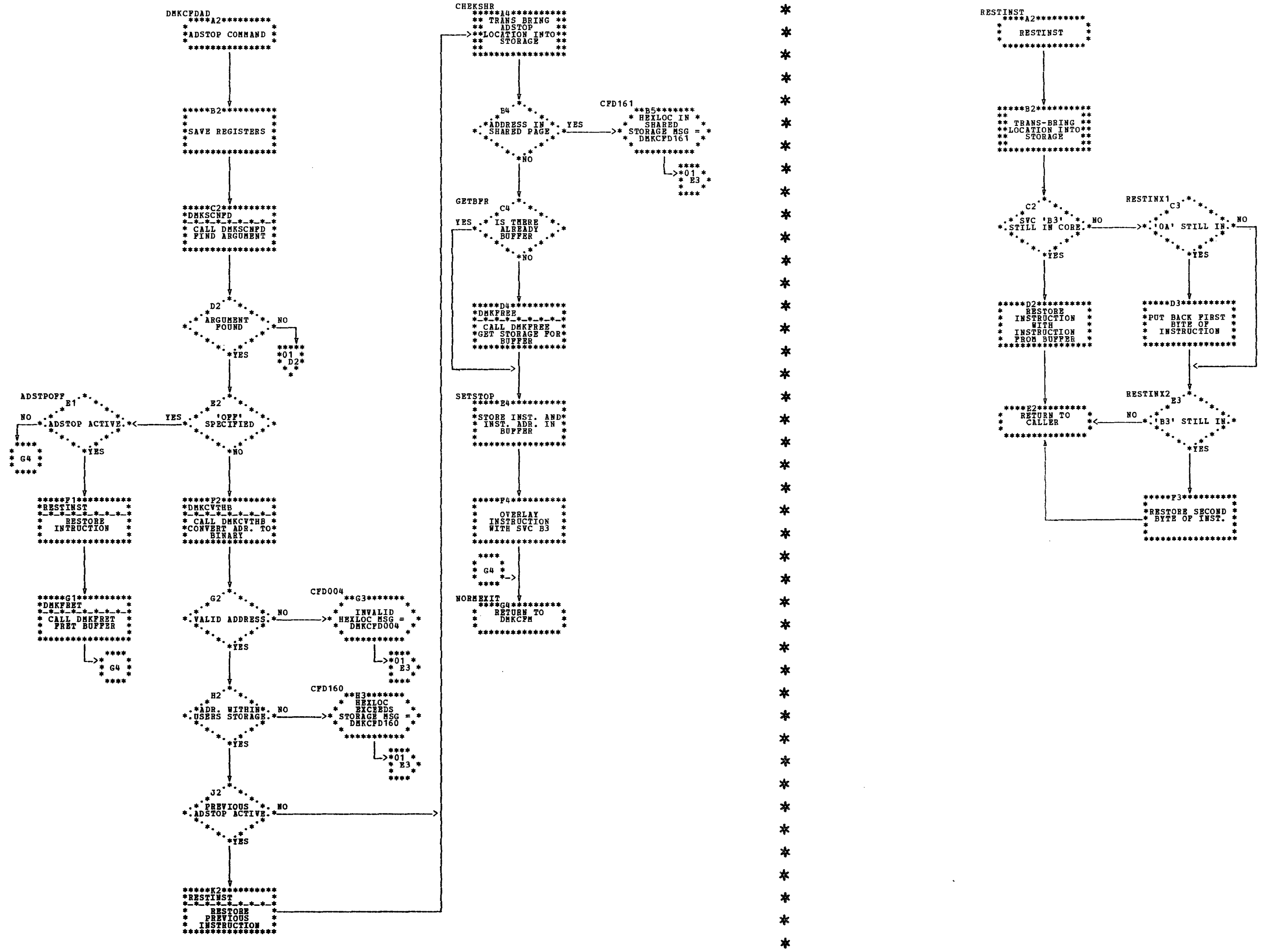

01F1
05 A1
STOGPR
A1 ADDRESS FIELD LENGTH < 1;> 2
YES
NO
B1 DMKCVTDB CALL-CONVERT ADR. TO BINARY
C1 VALID CONVERT
D1 DMKCVTHB CALL-TRY CONVERT FROM HEX TO BINARY
E1 VALID GPR ADDRESS
CDS010
E2 INVALID REGISTER MSG = DMKCDS010E
01 B4
STOGPRL
F1 STOSCAN BAL R8 - GET DATA ARGUMENT
G1 GPR ADR. > 15
CDS163
G2 EXCEEDS MAXIMUM STORAGE MSG = DMKCDS163
01 A4
H1 DMKCVTHB CALL-CONVERT DATA TO BINARY
J1 VALID DATA
03 G3
K1 STORE DATA AND BUMP GPR ADDRESS
05 A3
STOFPR
A3 ADR. FIELD LEN. =1
B3 DMKCVTDB CALL-CONVERT ADDRESS TO BINARY
C3 VALID ADDRESS
D3
STOFPRL
D3 STOSCAN BAL R8-GET DATA ARGUMENT
E3 FPR ADR. > 6
F3 DMKCVTHB CALL-CONVERT DATA TO BINARY
G3 VALID DATA
H3 STORE DATA AND BUMP FPR ADR.
05 A4
STOCRG
A4 ADR. FIELD LEN. 0 OR >2
E2
B4 DMKCVTDB CALL-TRY CONVERT FROM DEC. TO BIN.
C4 VALID CONVERT
F4
D4 DMKCVTHB CALL-TRY CONVERT FROM HEX TO BIN.
E4 VALID CONVERT
05 F4
06A4 06D1
STOCRGL
F4 STOSCAN BAL R8-GET DATA ARGUMENT
G4 CREG ADDRESS > 15
G2
H4 DMKCVTHB CALL-CONVERT DATA TO BINARY
J4 VALID CONVERT
CDS162
J5 INVALID ECR MSG = DMKCDS162
K4 CREG 0
06 A1
06 E3
05K4
06 A1
A1 EC MODE MACHINE
B1 CREG 1
STOCRG1
B2 CHKTRACE BAL R8 - CHECK FOR TRACING INSTRUCTIONS
STOCRGS
C1 STORE DATA INTO CREG
C2 CHECK LAST 6 BITS - MUST BE ZERO
STOCRGA
D1 BUMP ADDRESS TO NEXT CREG
05 F4
D2 STOCRGE BAL (R8) - SEND WARNING TO USER
STOCREG1
E2 SET STATUS BIT 'VMINVSEG' FOR DMKVATAB
F2 GO STORE NEW CR1 (& INDICATE DATA STORED)
06 A3
STOCRG0
A3 EC MODE MACHINE
STOCRGC
A4 STORE CREG 0 INTO VMBLOK
05 F4
B3 CHKTRACE BAL R8 - CHECK FOR TRACING INSTRUCTIONS
C3 SET R8 = A(STOCREG0) IN CASE OF AN INVALID VALUE
D3 NEW CR0 = RESET VALUE OF 000000E0
03 A1
E3 NEW CR0 = INVALID
STOCREG0
F3 SET STATUS BIT 'VMNEWCR0' FOR DMKVATAB
G3 GO STORE NEW CR0 (& INDICATE DATA STORED)

| DMKCDS -- Process STCP and STORE Commands (Parts 5 and 6 of 6)

DMKCFD -- Process ADSTOP and LOCATE Commands (Parts 1 and 2 of 4)

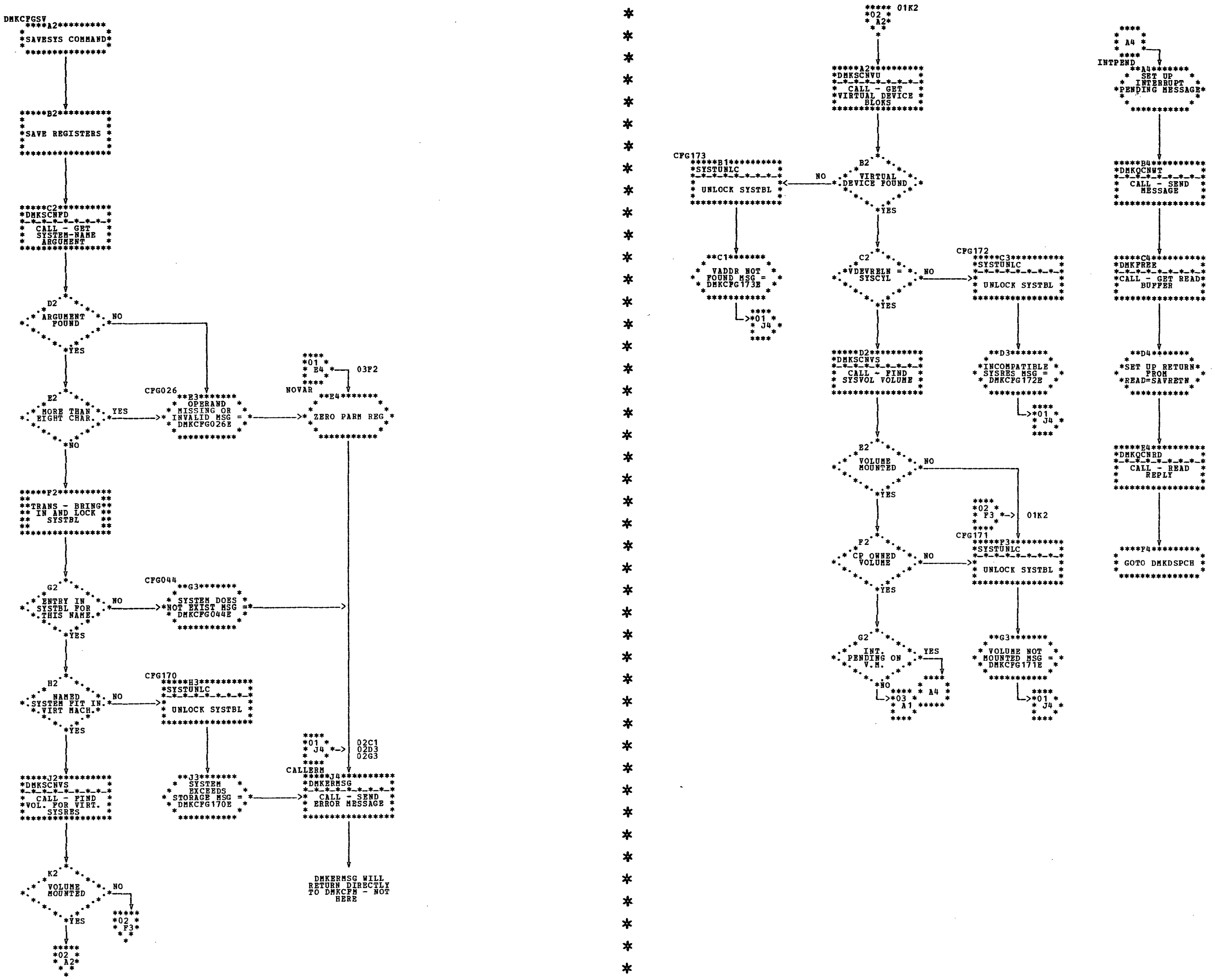

DMKCFDLO
A1
LOCATE COMMAND
B1
SAVE REGISTERS
C1
DMKSCNFD
CALL DMKSCNFD FIND FIRST ARGUMENT
D1
ARGUMENT FOUND
NO
YES
01 D2
03D2
CFD026
D2
OPERAND MISSING MSG = DMKCFD026
01 D3
02C4
NOVAR
D3
ZERO PARM REGISTER
D3
01 E3
03B5
03G3
03H3
CALLERM
E3
LOAD MODULE IDENTIFIER
E3
F3
DMKERMSG
CALL DMKCSMSG SEND ERROR MESSAGE
DMKERMSG WILL RETURN TO DMKCFM
E1
DMKSCNAU
CALL DMKSCNAU FIND VMBLOK ADDRESS
F1
VMBLOK FOUND
NO
YES
02 A3
G1
DMKCVTBH
CALL DMKCVTBH CONVERT VMBLOK ADR. TO HEX.
H1
DMKSCNFD
CALL DMKSCNFD FIND VIRTUAL ADDRESS
J1
ARGUMENT FOUND
NO
YES
A4
VMONLY
J2
SET NO BUFFER FLAG
K2
BUILD MSG. FOR VMBLOK ONLY IN SAVEWRK AREA
02 A2
A4
A4
DMKCVTHB
CALL DMKCVTHB CONVERT ADDRESS TO BINARY
B4
VALID ADDRESS
NO
YES
CFD022
B5
VADDR INVALID MSG = DMKCFD022
D3
C4
DMKSCNVU
CALL DMKSCNVU GET VIRT. DEV. BLOK ADRESSES
01 D5
02E3
CFD040
D5
DEV DOES NOT EXIST MSG = DMKCFD040
E3
D4
BLOKS FOUND
NO
YES
E4
DMKQCNWT
CALL DMKQCNWT WRITE HEADER LINE
F4
DMKFREE
CALL DMKFREE GET STORAGE FOR MSG. BUFFER
G4
DMKCVTBH
CALL DMKCVTBH CONVERT VCHBLOK TO HEX.
H4
DMKCVTBH
CALL DMKCVTBH CONVERT VCUBLOK TO HEX.
J4
DMKCVTBH
CALL DMKCVTBH CONV. VDEVBLOK TO HEX.
K4
BUILD MSG. FOR VMBLOK AND VIRT. DEV BLOK ADR.
02 A2

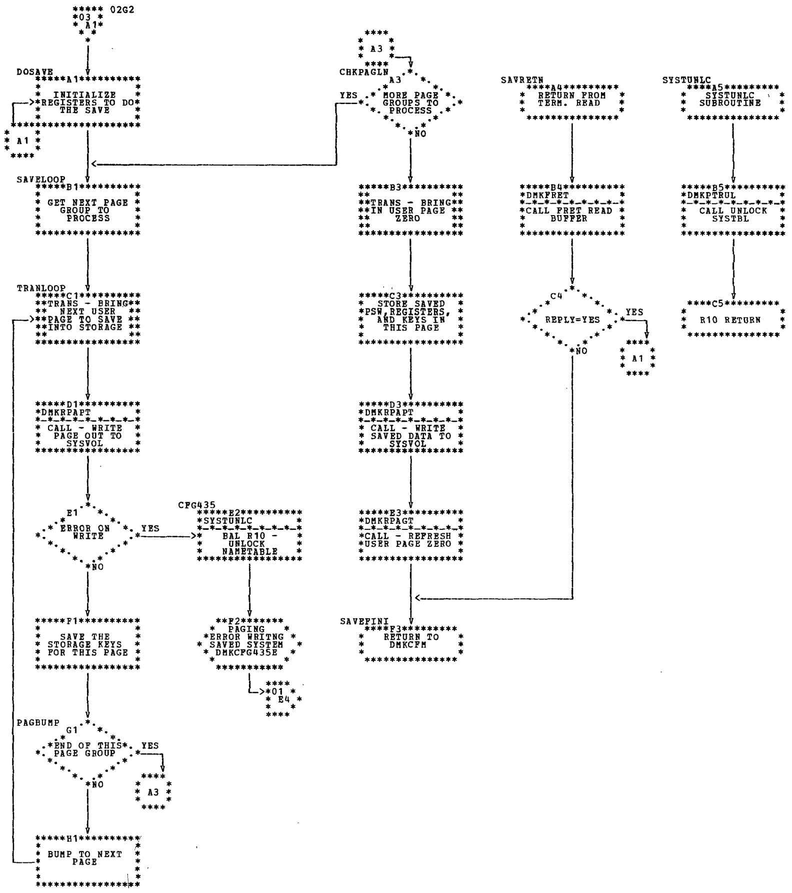

02 A2
01K2
01K4
CFDWRIT
A2
DMKQCNWT
CALL DMKQCNWT WRITE MESSAGE
B2
BUFFER TO FRET
NO
YES
C2
DMKFRET
CALL DMKFRET TO FRET BUFFER
CFDEXIT
D2
RETURN TO DMKCFM
02 A3
01F1
LOCRADDR
A3
SET NO BUFFER FLAG
B3
DMKCVTHB
CALL DMKCVTHB CONVERT REAL ADR. TO BINARY
C3
VALID ADDRESS
NO
YES
CFD021
C4
RADDR INVALID MSG = DMKCFD021
01 D3
D3
DMKSCNRU
CALL DMKSCNRU GET ADR. OF REAL DEV. BLOKS
E3
BLOKS FOUND
NO
YES
01 D5
F3
DMKQCNWT
CALL DMKQCNWT WRITE HEADER LINE
G3
DMKCVTBH
CALL DMKCVTBH CONVERT RCHBLOK ADR. TO HEX.
H3
DMKCVTBH
CALL DMKCVTBH CONVERT RCUBLOK ADR. TO HEX.
J3
DMKCVTBH
CALL DMKCVTBH CONV. RDEVBLOK ADR. TO HEX.
K3
BUILD MSG. FOR REAL DEV. BLOK ADR. IN SAVEWRK AREA



DMKCFD -- Process ADSTOP and LOCATE Commands (Parts 3 and 4 of 4)

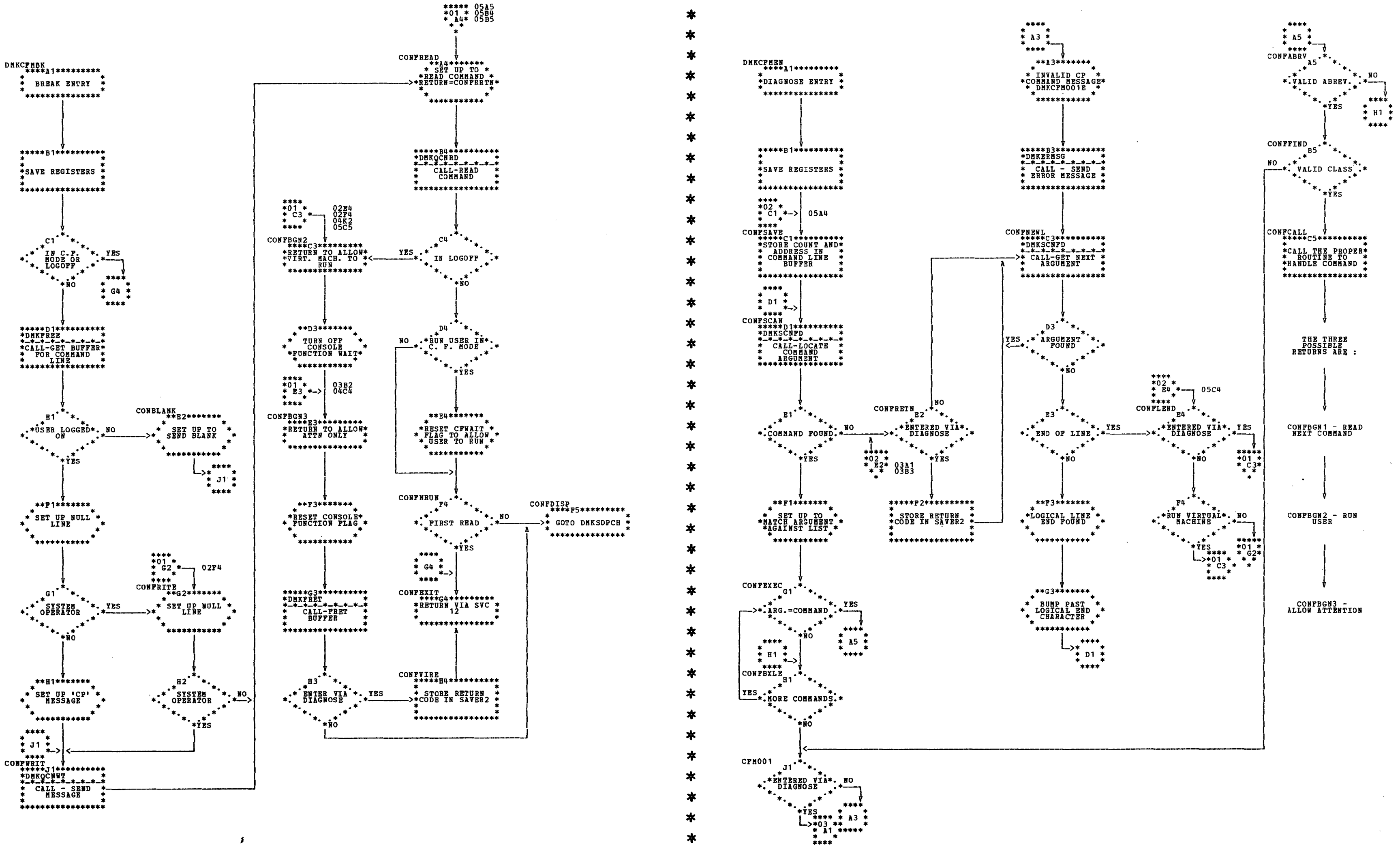
| DMKCFG -- Process SAVESYS Command (Parts 1 and 2 of 3)

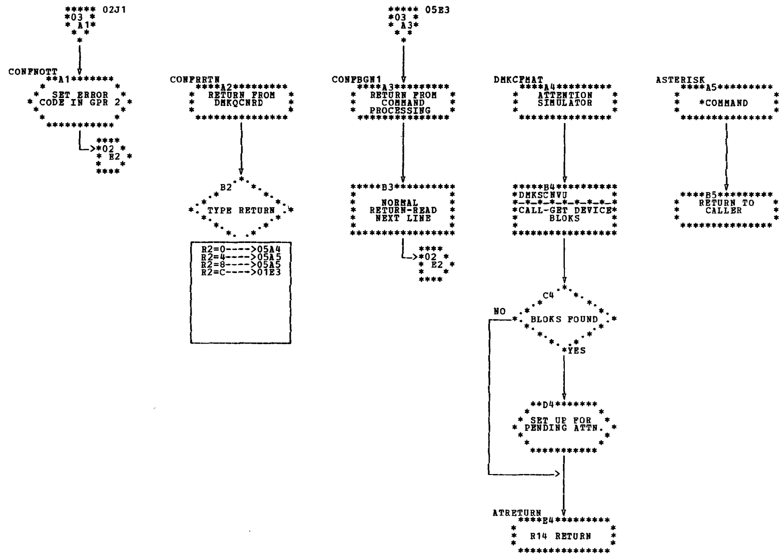


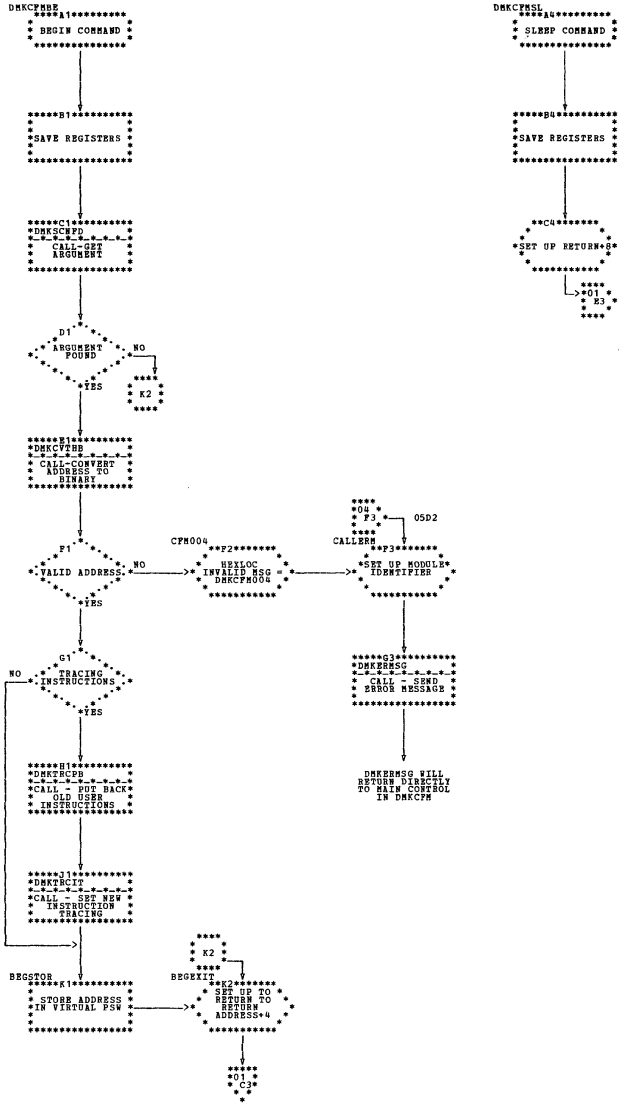

02G2
03
A1
DOSAVE
A1
INITIALIZE REGISTERS TO DO THE SAVE
A1
SAVELOOP
B1
GET NEXT PAGE GROUP TO PROCESS
TRANLOOP
C1
TRANS - BRING NEXT USER PAGE TO SAVE INTO STORAGE
D1
DMKRPAPT
CALL - WRITE PAGE OUT TO SYSVOL
E1
ERROR ON WRITE
YES
NO
CFG435
E2
SYSTUNLC
BAL R10 - UNLOCK NAMETABLE
F2
PAGING ERROR WRITNG SAVED SYSTEM DMKCFG435E
01
E4
F1
SAVE THE STORAGE KEYS FOR THIS PAGE
PAGBUMP
G1
END OF THIS PAGE GROUP
YES
A3
NO
H1
BUMP TO NEXT PAGE
A3
CHKPAGLN
A3
MORE PAGE GROUPS TO PROCESS
YES
NO
B3
TRANS - BRING IN USER PAGE ZERO
C3
STORE SAVED PSW, REGISTERS, AND KEYS IN THIS PAGE
D3
DMKRPAPT
CALL - WRITE SAVED DATA TO SYSVOL
E3
DMKRPAGT
CALL - REFRESH USER PAGE ZERO
SAVEFINI
F3
RETURN TO DMKCFM
SAVRETN
A4
RETURN FROM TERM. READ
B4
DMKFRET
CALL FRET READ BUFFER
C4
REPLY=YES
YES
A1
NO
SYSTUNLC
A5
SYSTUNLC SUBROUTINE
B5
DMKPTRUL
CALL UNLOCK SYSTBL
C5
R10 RETURN

| DMKCFG -- Process SAVESYS Command (Part 3 of 3)

| DMKCFM -- Main Console Function Routine (Parts 1 and 2 of 5)

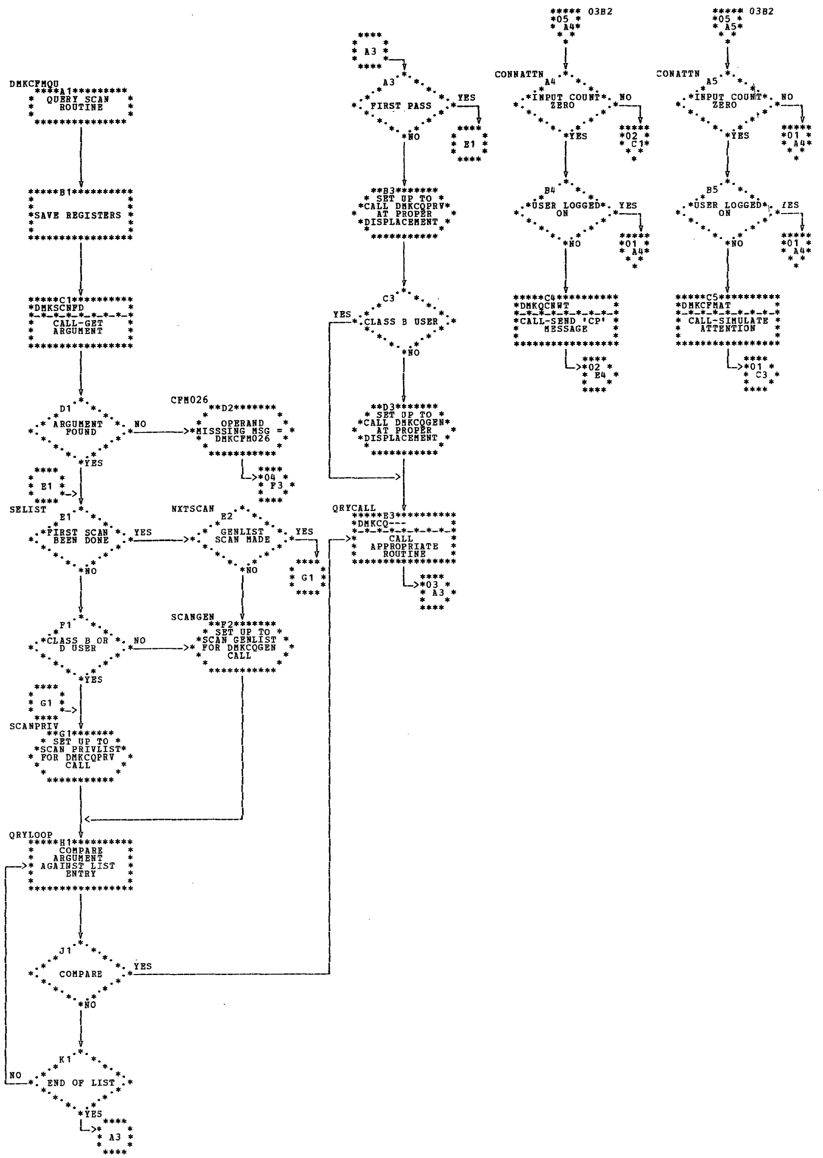


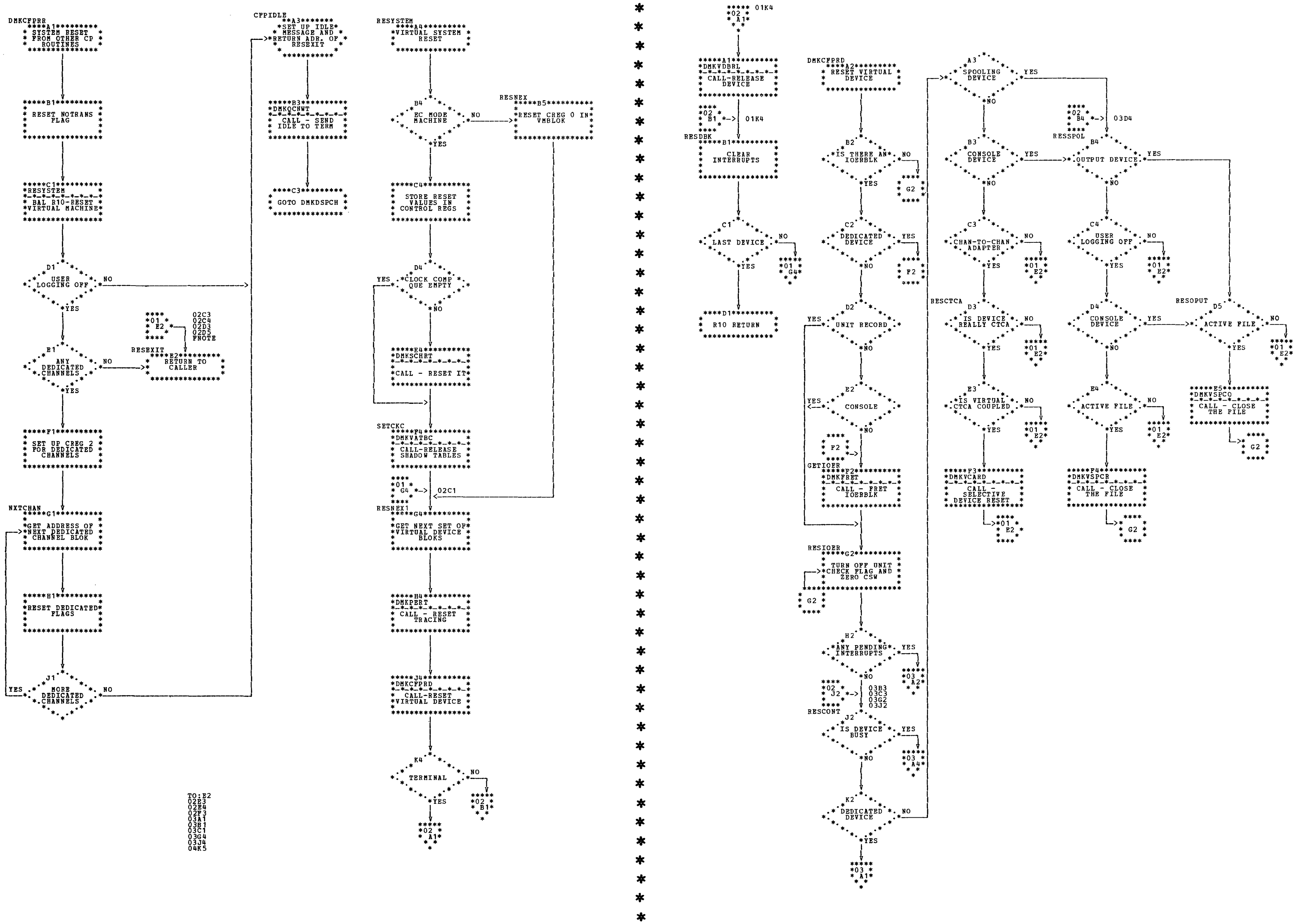




| DMKCFM -- Main Console Function Routine (Parts 3 and 4 of 5)

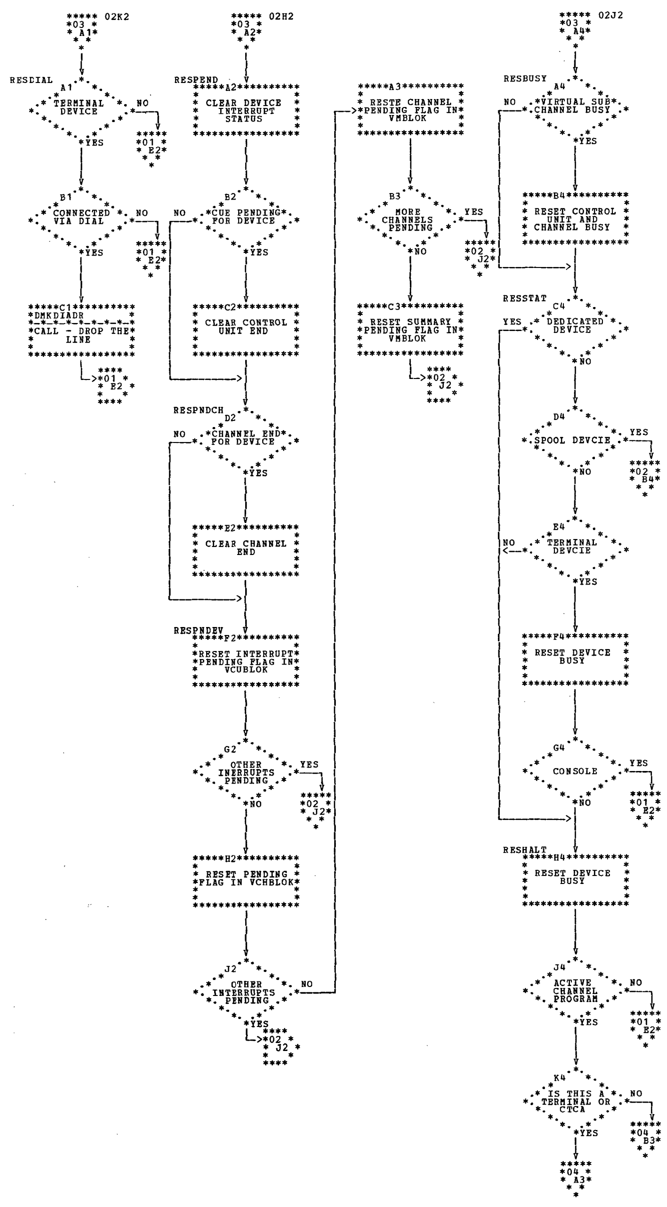
| DMKCFM -- Main Console Function Routine (Part 5 of 5)

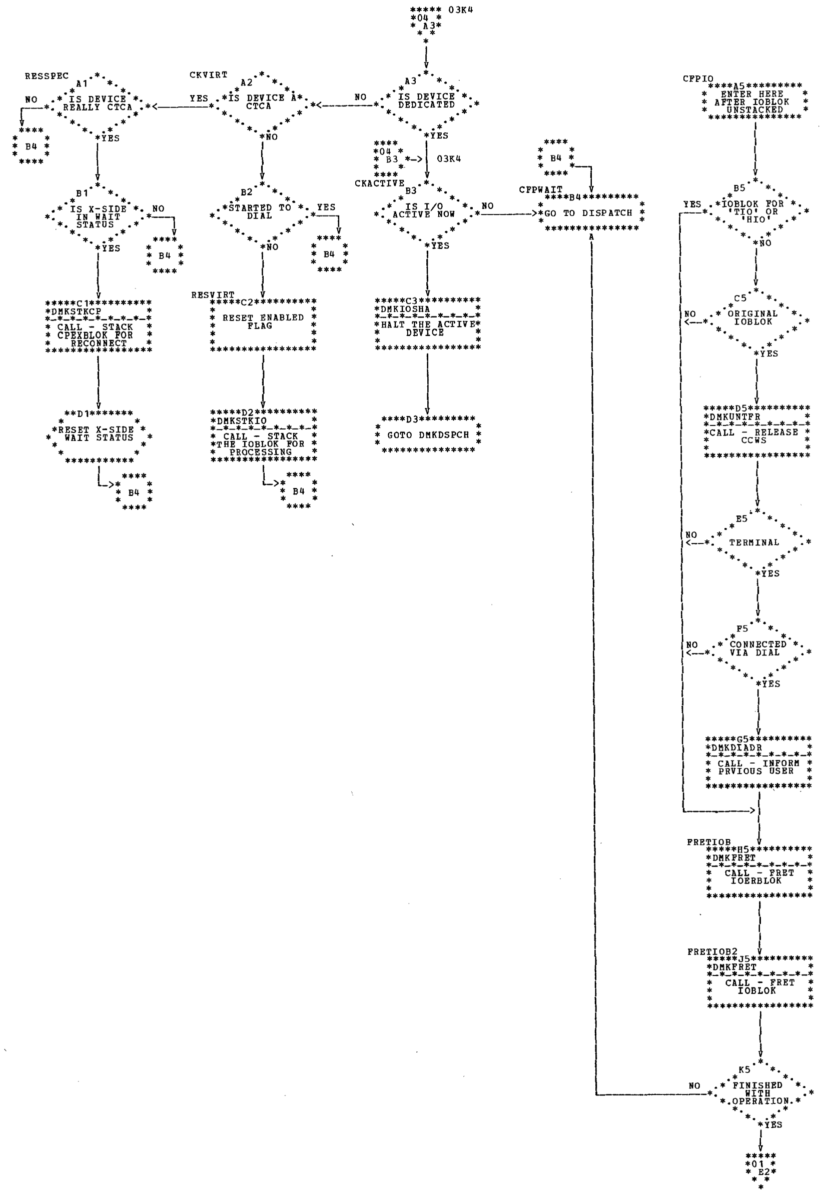


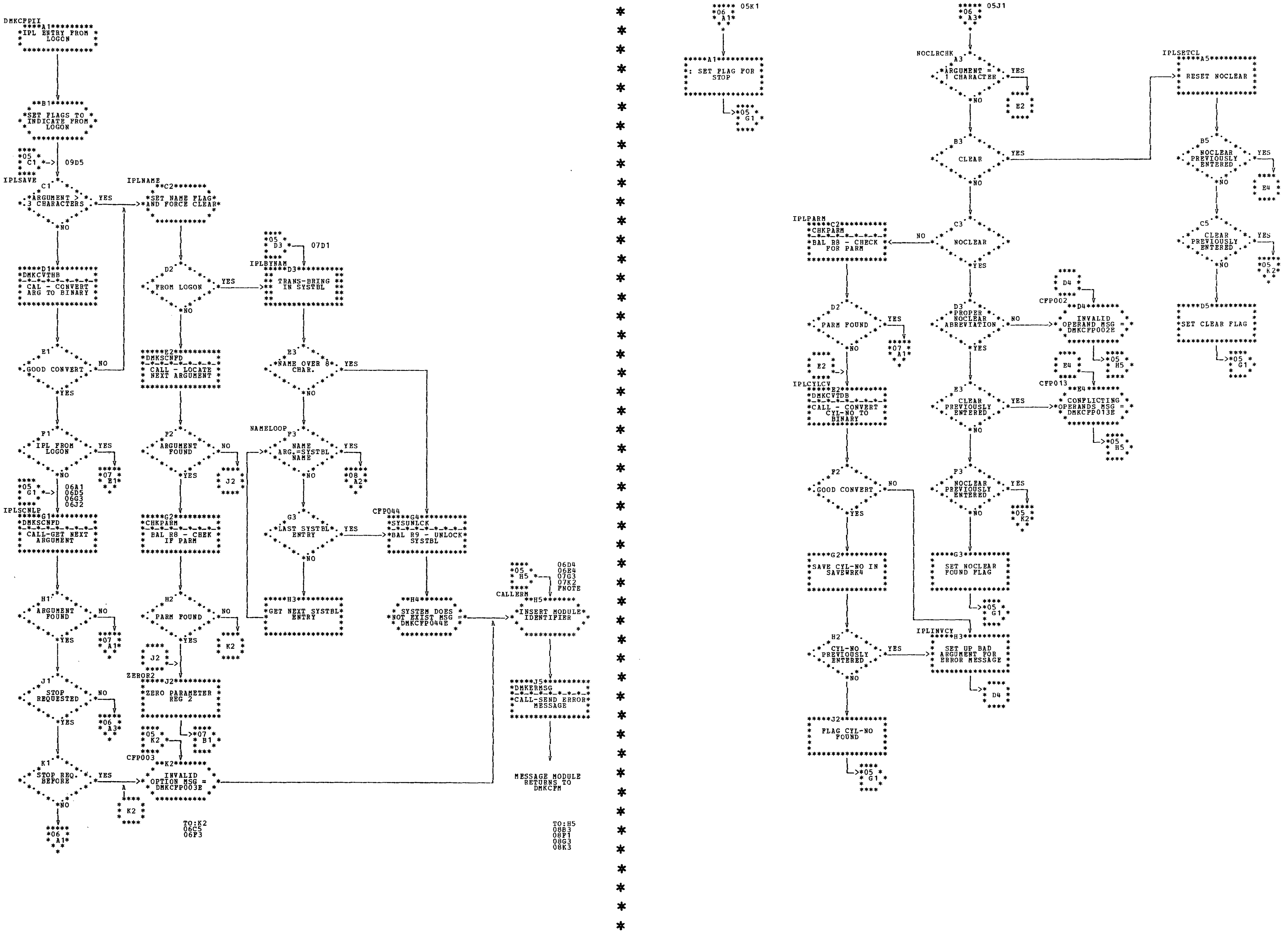


| DMKCFP -- Simulate CPU Console to Virtual Machine (Parts 1 and 2 of 10)

| DMKCFP -- Simulate CPU Console to Virtual Machine (Parts 3 and 4 of 10)

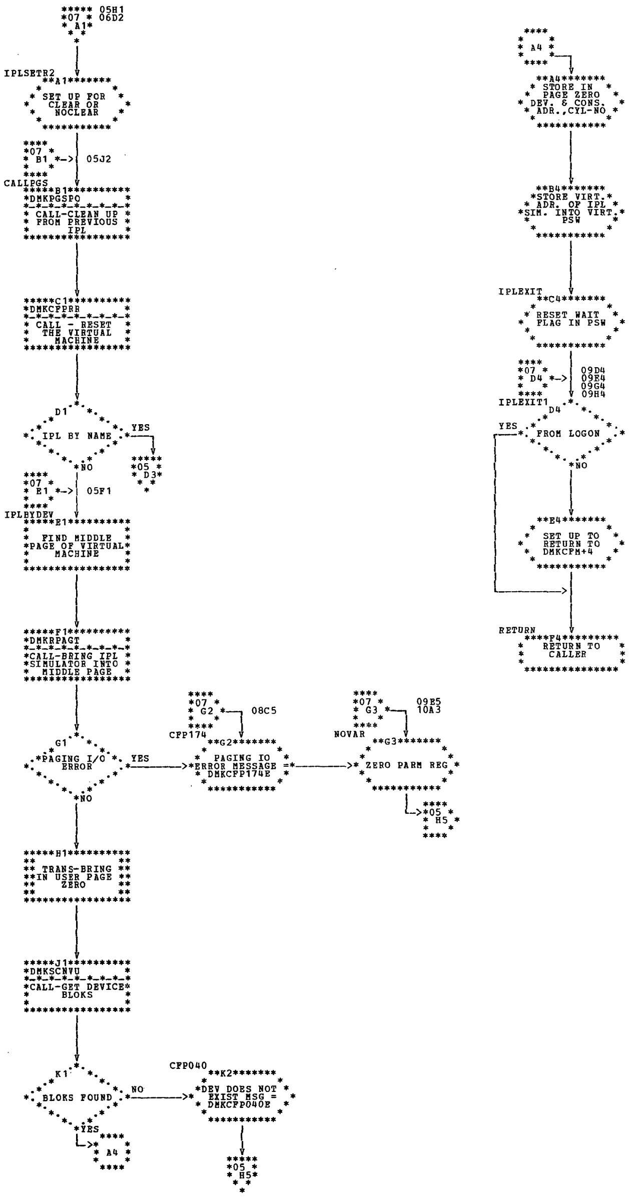


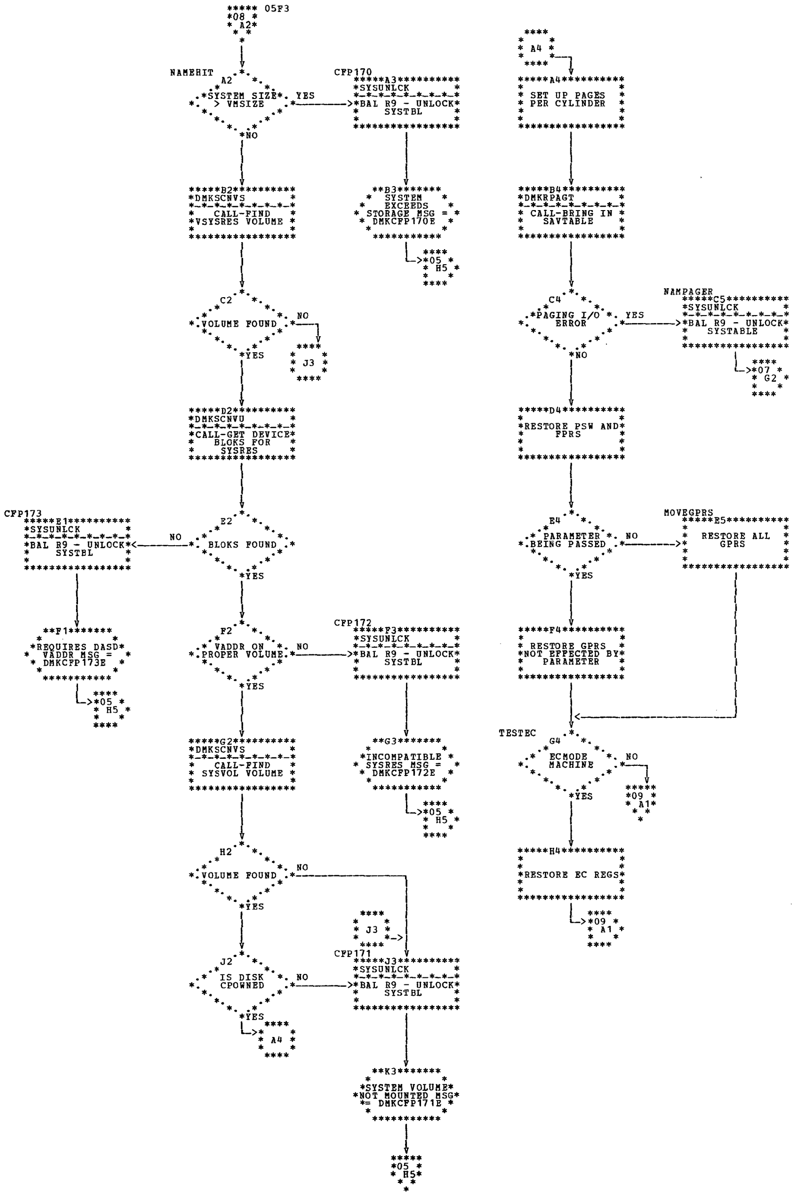


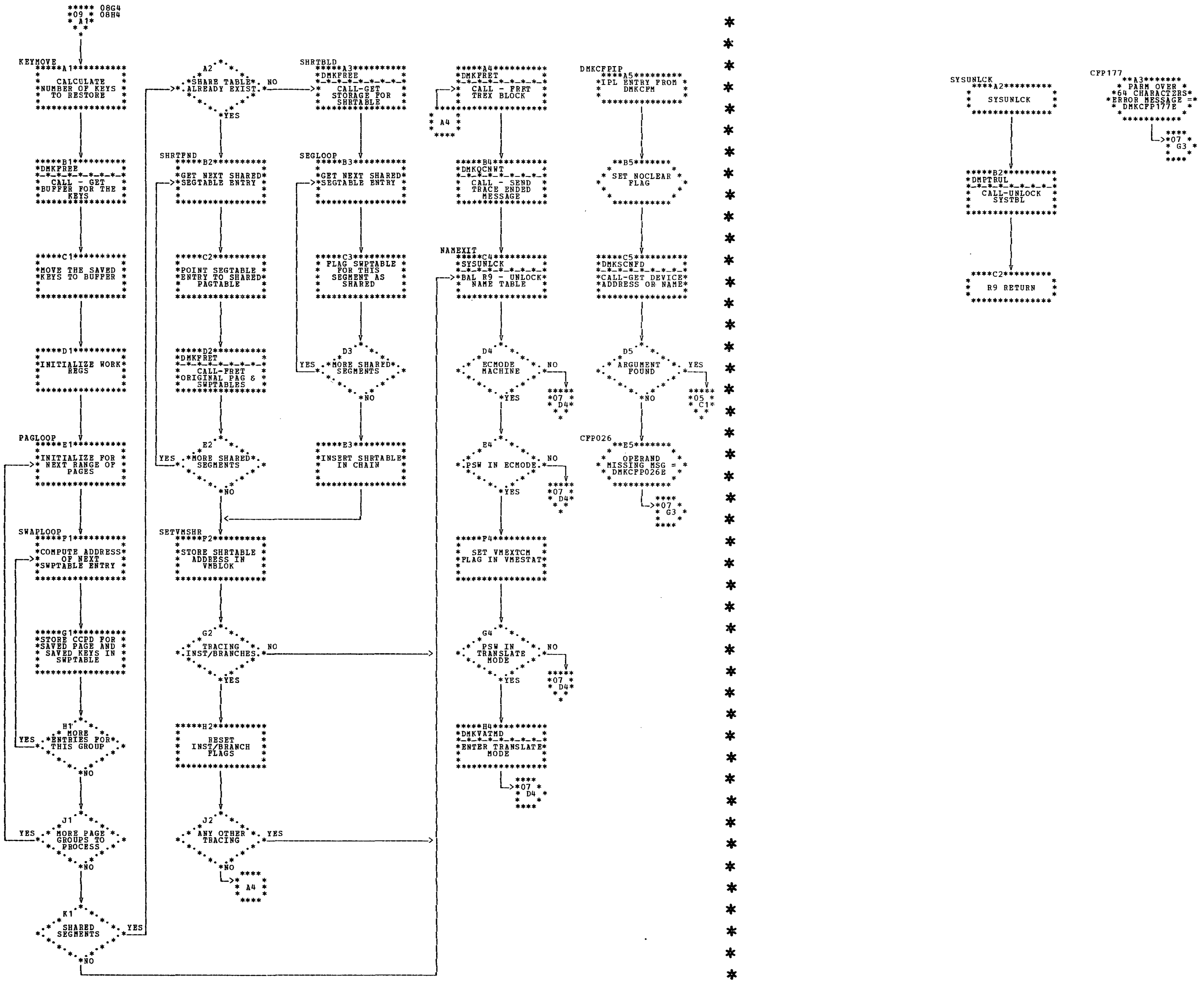


| DMKCFP -- Simulate CPU Console to Virtual Machine (Parts 5 and 6 of 10)

| DMKCFP -- Simulate CPU Console to Virtual Machine (Parts 7 and 8 of 10)


07 A1 05H1 06D2
IPLSETR2 A1 SET UP FOR CLEAR OR NOCLEAR
07 B1 05J2
CALLPGS B1 DMKPGSPO CALL-CLEAN UP FROM PREVIOUS IPL
C1 DMKCFPRR CALL - RESET THE VIRTUAL MACHINE
D1 IPL BY NAME
YES
05 D3
NO
07 E1 05F1
IPLBYDEV E1 FIND MIDDLE PAGE OF VIRTUAL MACHINE
F1 DMKRPAGT CALL-BRING IPL SIMULATOR INTO MIDDLE PAGE
G1 PAGING I/O ERROR
YES
NO
07 G2 08C5
CFP174 G2 PAGING IO ERROR MESSAGE = DMKCFP174E
07 G3 09E5 10A3
NOVAR G3 ZERO PARM REG
05 H5
H1 TRANS-BRING IN USER PAGE ZERO
J1 DMKSCNVU CALL-GET DEVICE BLOKS
K1 BLOKS FOUND
NO
YES
A4
CFP040 K2 DEV DOES NOT EXIST MSG = DMKCFP040E
05 H5
A4
A4 STORE IN PAGE ZERO DEV. & CONS. ADR.,CYL-NO
B4 STORE VIRT. ADR. OF IPL SIM. INTO VIRT. PSW
IPLEXIT C4 RESET WAIT FLAG IN PSW
07 D4 09D4 09E4 09G4 09H4
IPLEXIT1 D4 FROM LOGON
YES
NO
E4 SET UP TO RETURN TO DMKCFM+4
RETURN F4 RETURN TO CALLER

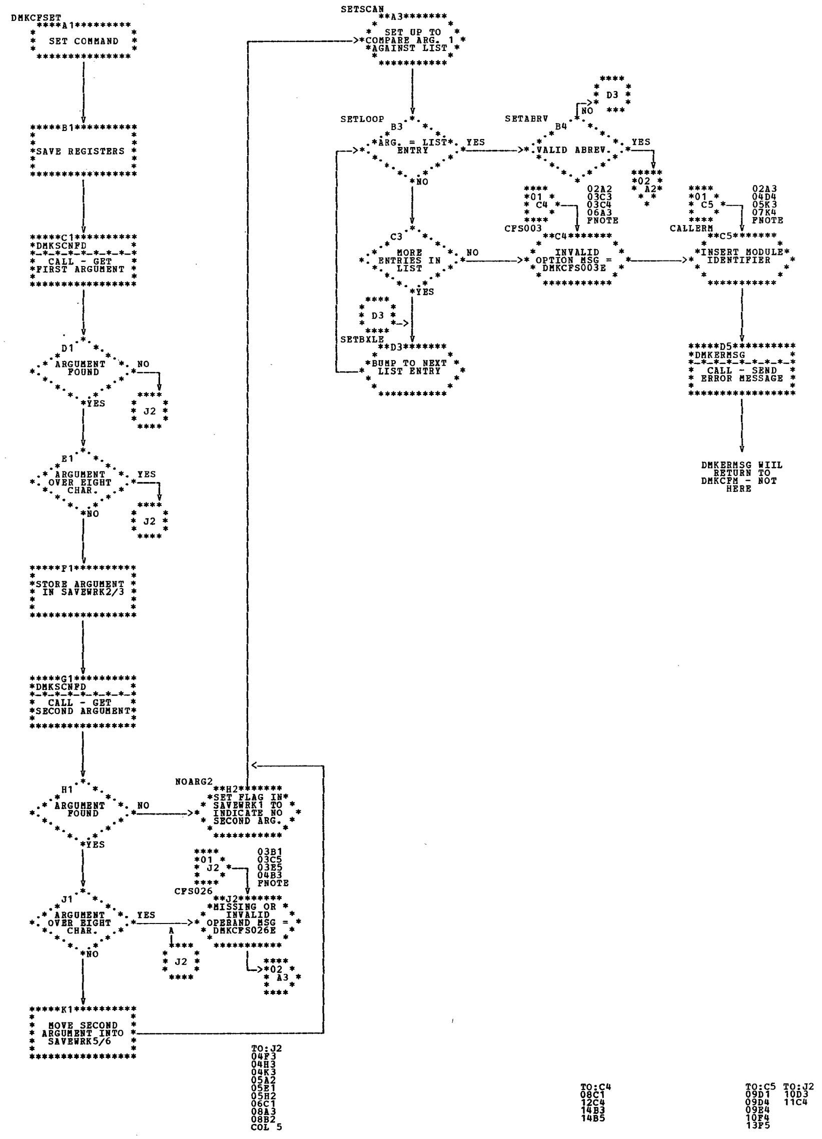

08 A2 05F3
NAMEHIT A2 SYSTEM SIZE > VMSIZE
YES
NO
CFP170 A3 SYSUNLCK BAL R9 - UNLOCK SYSTBL
B2 DMKSCNVS CALL-FIND VSYSRES VOLUME
B3 SYSTEM EXCEEDS STORAGE MSG = DMKCFP170E
05 H5
C2 VOLUME FOUND
NO
J3
YES
D2 DMKSCNVU CALL-GET DEVICE BLOKS FOR SYSRES
CFP173 E1 SYSUNLCK BAL R9 - UNLOCK SYSTBL
NO
E2 BLOKS FOUND
YES
F1 REQUIRES DASD VADDR MSG = DMKCFP173E
05 H5
F2 VADDR ON PROPER VOLUME
NO
YES
CFP172 F3 SYSUNLCK BAL R9 - UNLOCK SYSTBL
G2 DMKSCNVS CALL-FIND SYSVOL VOLUME
G3 INCOMPATIBLE SYSRES MSG = DMKCFP172E
05 H5
H2 VOLUME FOUND
NO
YES
J3
CFP171 J3 SYSUNLCK BAL R9 - UNLOCK SYSTBL
J2 IS DISK CPOWNED
NO
YES
A4
K3 SYSTEM VOLUME NOT MOUNTED MSG = DMKCFP171E
05 H5
A4
A4 SET UP PAGES PER CYLINDER
B4 DMKRPAGT CALL-BRING IN SAVTABLE
C4 PAGING I/O ERROR
YES
NO
NAMPAGER C5 SYSUNLCK BAL R9 - UNLOCK SYSTABLE
07 G2
D4 RESTORE PSW AND FPRS
E4 PARAMETER BEING PASSED
NO
YES
MOVEGPRS E5 RESTORE ALL GPRS
F4 RESTORE GPRS NOT EFFECTED BY PARAMETER
TESTEC G4 ECMODE MACHINE
NO
09 A1
YES
H4 RESTORE EC REGS
09 A1

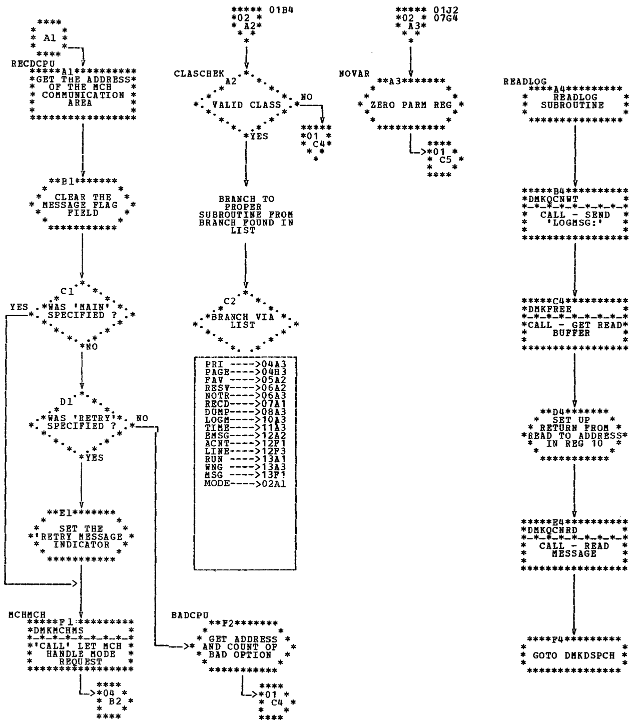


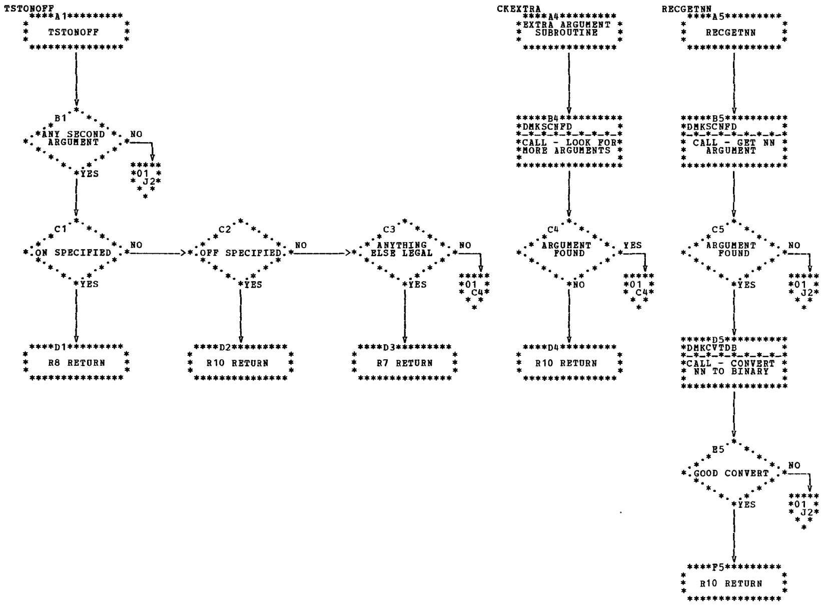
09 A1 08G4 08H4
KEYMOVE A1 CALCULATE NUMBER OF KEYS TO RESTORE
B1 DMKFREE CALL - GET BUFFER FOR THE KEYS
C1 MOVE THE SAVED KEYS TO BUFFER
D1 INITIALIZE WORK REGS
PAGLOOP E1 INITIALIZE FOR NEXT RANGE OF PAGES
SWAPLOOP F1 COMPUTE ADDRESS OF NEXT SWPTABLE ENTRY
G1 STORE CCPD FOR SAVED PAGE AND SAVED KEYS IN SWPTABLE
H1 MORE ENTRIES FOR THIS GROUP
J1 MORE PAGE GROUPS TO PROCESS
K1 SHARED SEGMENTS
A2 SHARE TABLE ALREADY EXIST
SHRTFND B2 GET NEXT SHARED SEGTABLE ENTRY
C2 POINT SEGTABLE ENTRY TO SHARED PAGTABLE
D2 DMKFRET CALL-FRET ORIGINAL PAG & SWPTABLES
E2 MORE SHARED SEGMENTS
SETVMSHR F2 STORE SHRTABLE ADDRESS IN VMBLOK
G2 TRACING INST/BRANCHES
H2 RESET INST/BRANCH FLAGS
J2 ANY OTHER TRACING
A4
SHRTBLD A3 DMKFREE CALL-GET STORAGE FOR SHRTABLE
SEGLOOP B3 GET NEXT SHARED SEGTABLE ENTRY
C3 FLAG SWPTABLE FOR THIS SEGMENT AS SHARED
D3 MORE SHARED SEGMENTS
E3 INSERT SHRTABLE IN CHAIN
A4 DMKFRET CALL - FRET TREX BLOCK
A4
B4 DMKQCNWT CALL - SEND TRACE ENDED MESSAGE
NAMEXIT C4 SYSUNLCK BAL R9 - UNLOCK NAME TABLE
D4 ECMODE MACHINE
07 D4
E4 PSW IN ECMODE
07 D4
F4 SET VMEXTCM FLAG IN VMESTAT
G4 PSW IN TRANSLATE MODE
07 D4
H4 DMKVATMD ENTER TRANSLATE MODE
07 D4
DMKCFPIP A5 IPL ENTRY FROM DMKCPM
B5 SET NOCLEAR FLAG
C5 DMKSCNFD CALL-GET DEVICE ADDRESS OR NAME
D5 ARGUMENT FOUND
05 C1
CFP026 E5 OPERAND MISSING MSG = DMKCFP026E
07 G3
YES
NO
SYSUNLCK A2 SYSUNLCK
B2 DMPTRUL CALL-UNLOCK SYSTBL
C2 R9 RETURN
CFP177 A3 PARM OVER 64 CHARACTERS ERROR MESSAGE = DMKCFP177E
07 G3

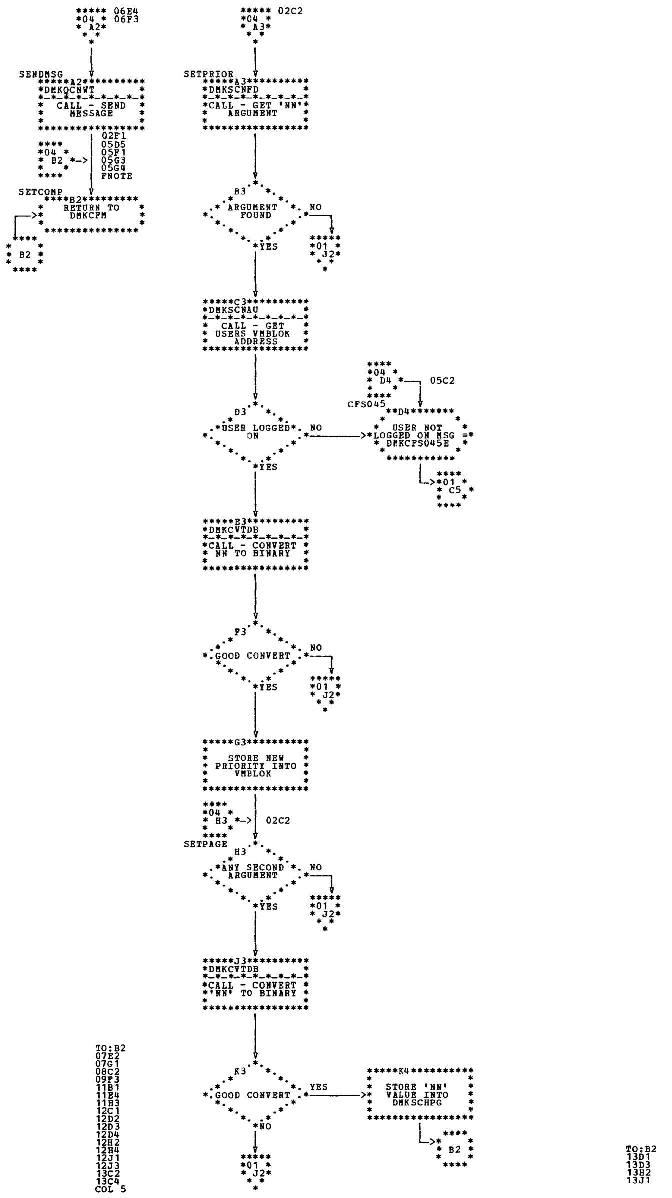
| DMKCFP -- Simulate CPU Console to Virtual Machine (Parts 9 and 10 of 10)

| DMKCFS -- Process SET Command (Parts 1 and 2 of 14)



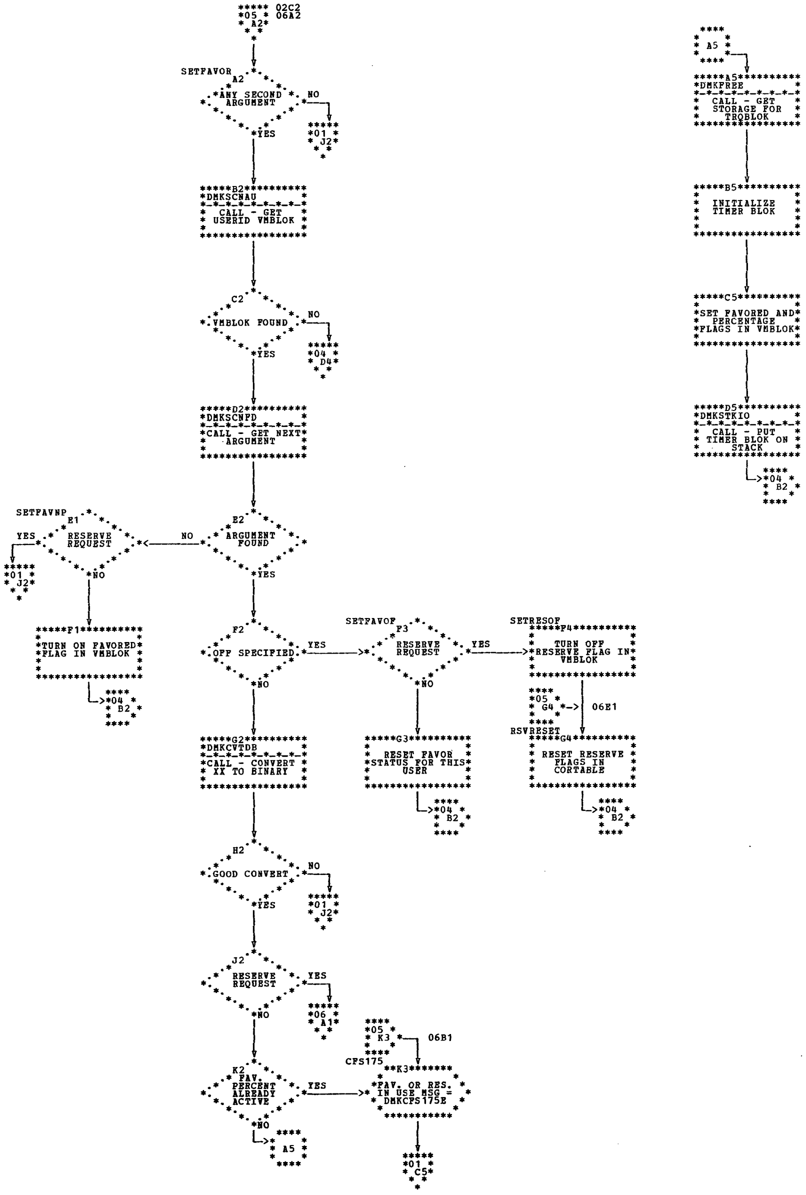


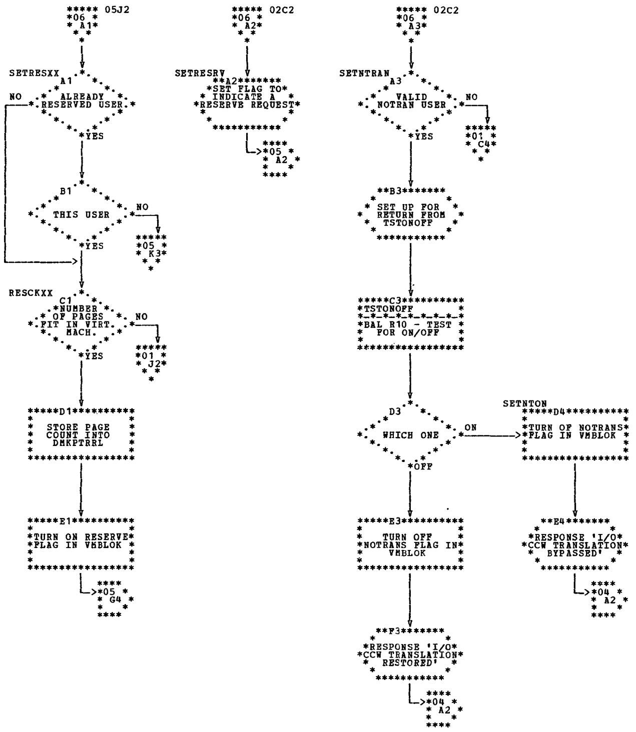


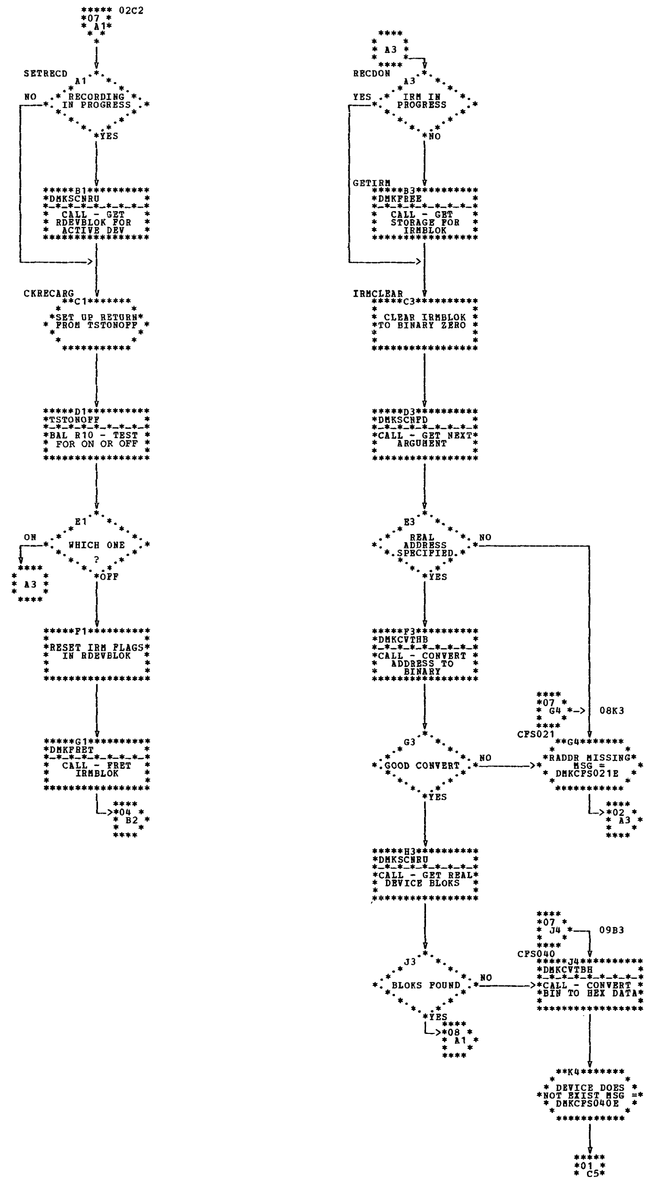


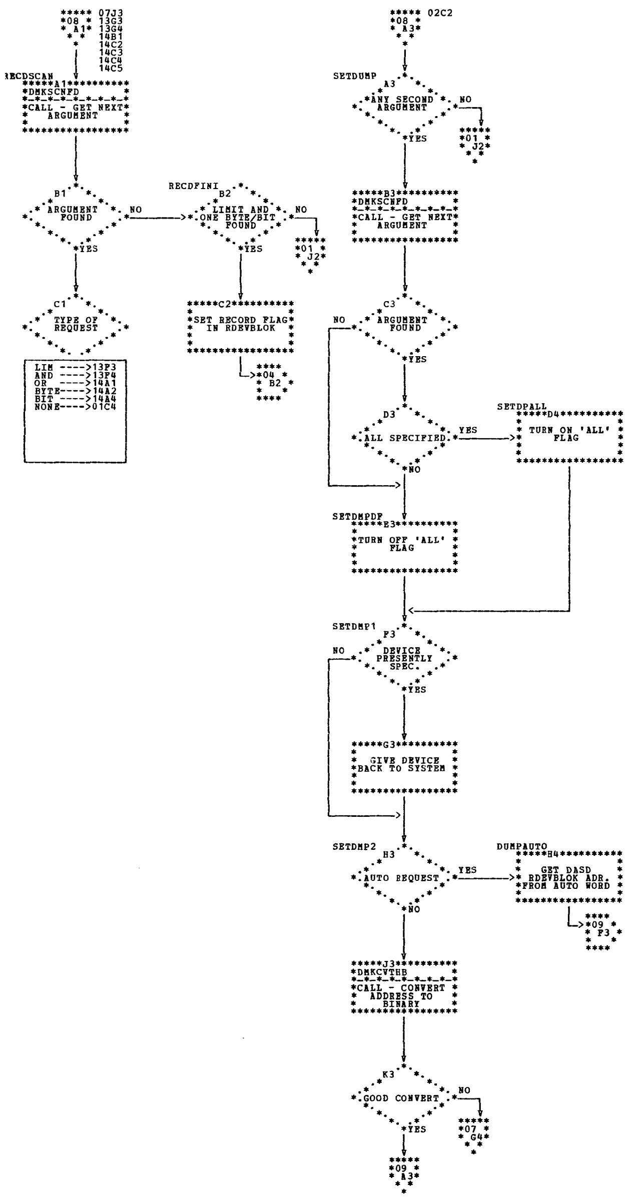
| DMKCFS -- Process SET Command (Parts 3 and 4 of 14)

DMKCFS -- Process SET Command (Parts 5 and 6 of 14)



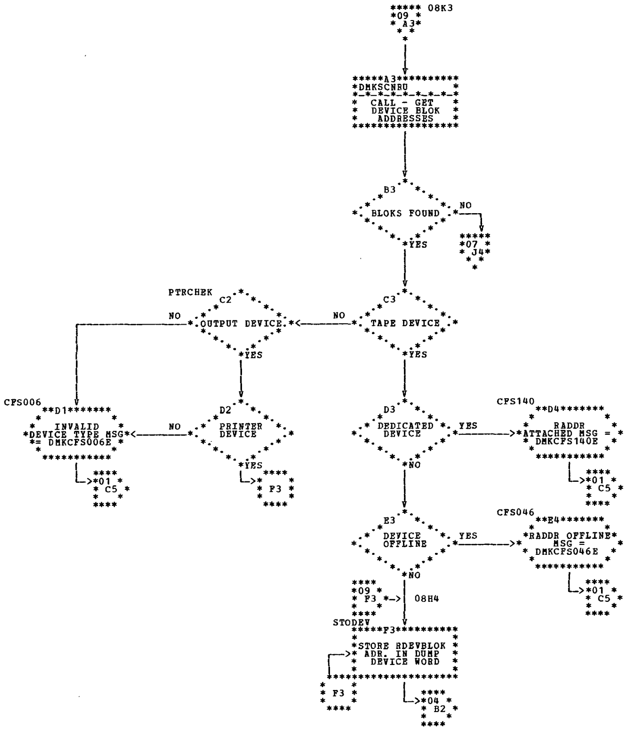

06 A1 05J2
SETRESXX A1
ALREADY RESERVED USER
NO
YES
B1
THIS USER
NO
05 K3
YES
RESCKXX C1
NUMBER OF PAGES FIT IN VIRT. MACH.
NO
01 J2
YES
D1
STORE PAGE COUNT INTO DMKPTRRL
E1
TURN ON RESERVE FLAG IN VMBLOK
05 G4
06 A2 02C2
SETRESRV A2
SET FLAG TO INDICATE A RESERVE REQUEST
05 A2
06 A3 02C2
SETNTRAN A3
VALID NOTRAN USER
NO
01 C4
YES
B3
SET UP FOR RETURN FROM TSTONOFF
C3 TSTONOFF
BAL R10 - TEST FOR ON/OFF
D3
WHICH ONE
ON
OFF
SETNTON D4
TURN OF NOTRANS FLAG IN VMBLOK
E3
TURN OFF NOTRANS FLAG IN VMBLOK
E4
RESPONSE 'I/O CCW TRANSLATION BYPASSED'
04 A2
F3
RESPONSE 'I/O CCW TRANSLATION RESTORED'
04 A2

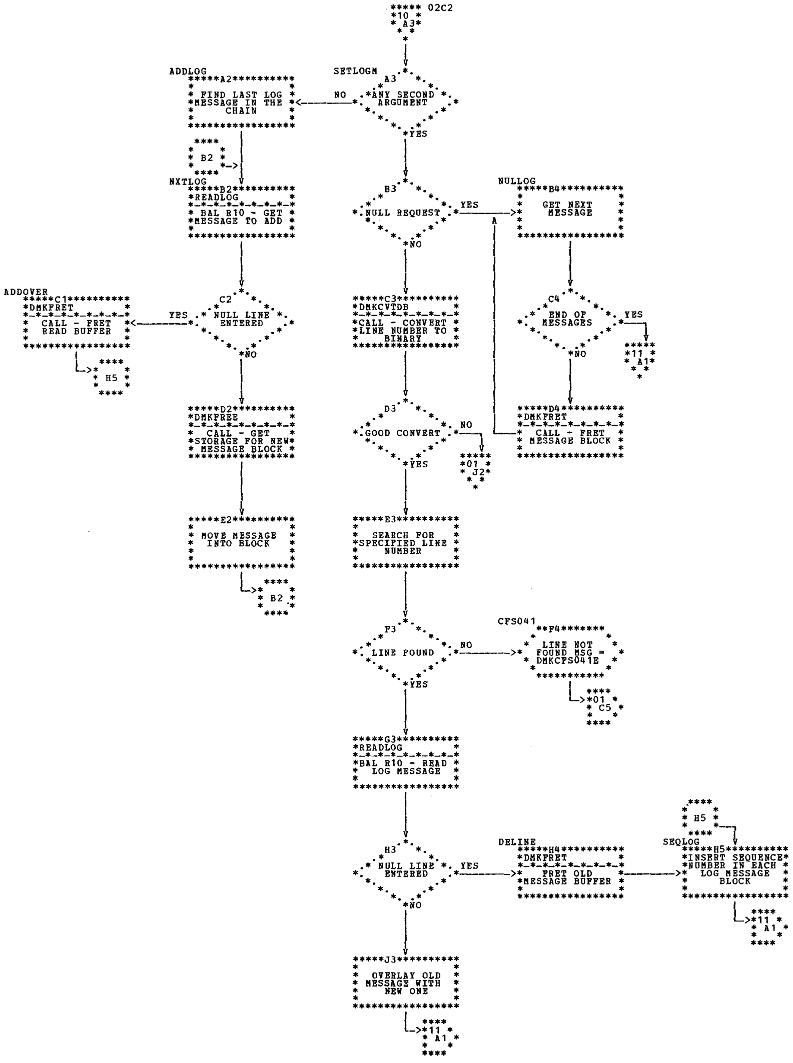


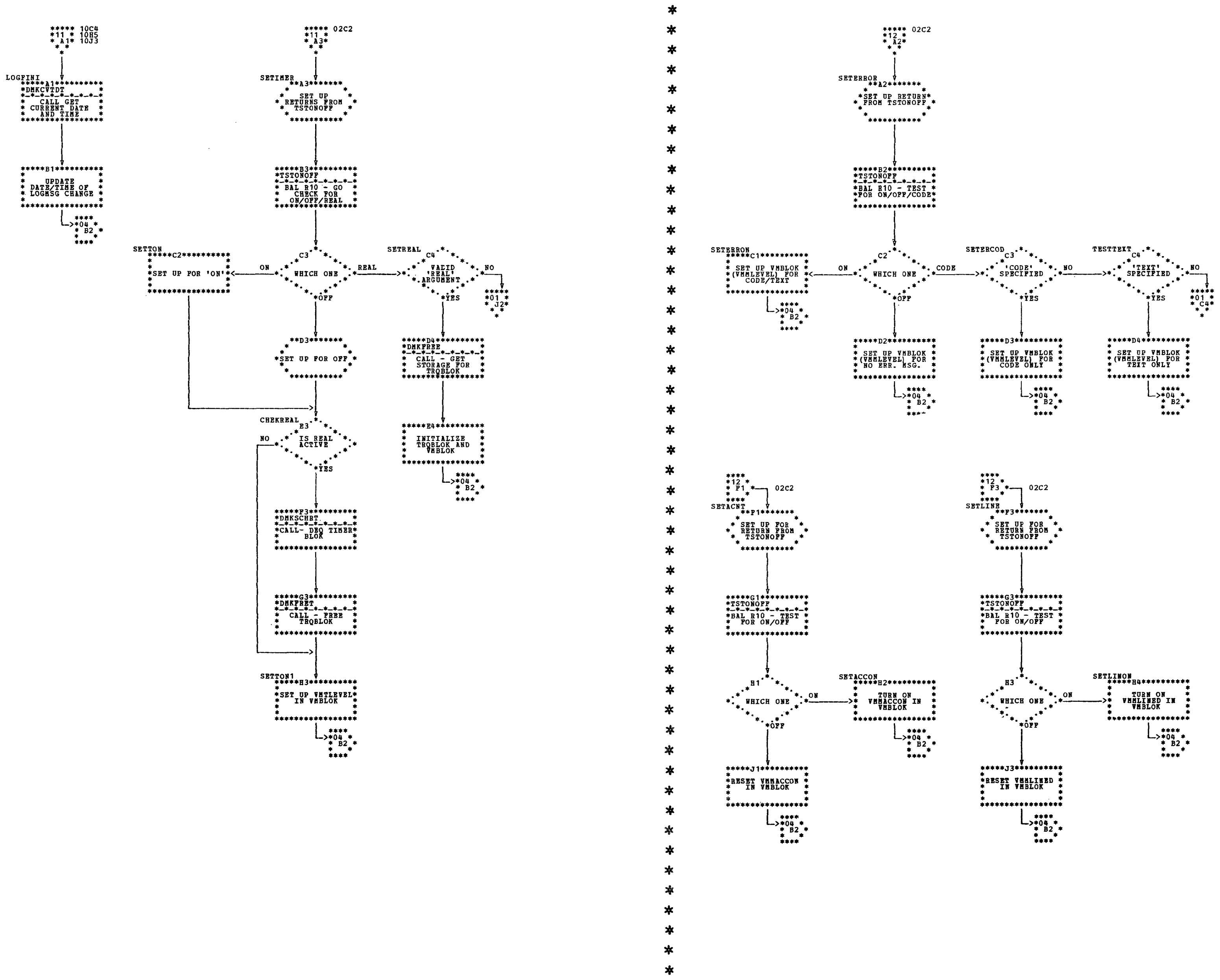


| DMKCFS -- Process SET Command (Parts 7 and 8 of 14)

DMKCFS -- Process SET Command (Parts 9 and 10 of 14)

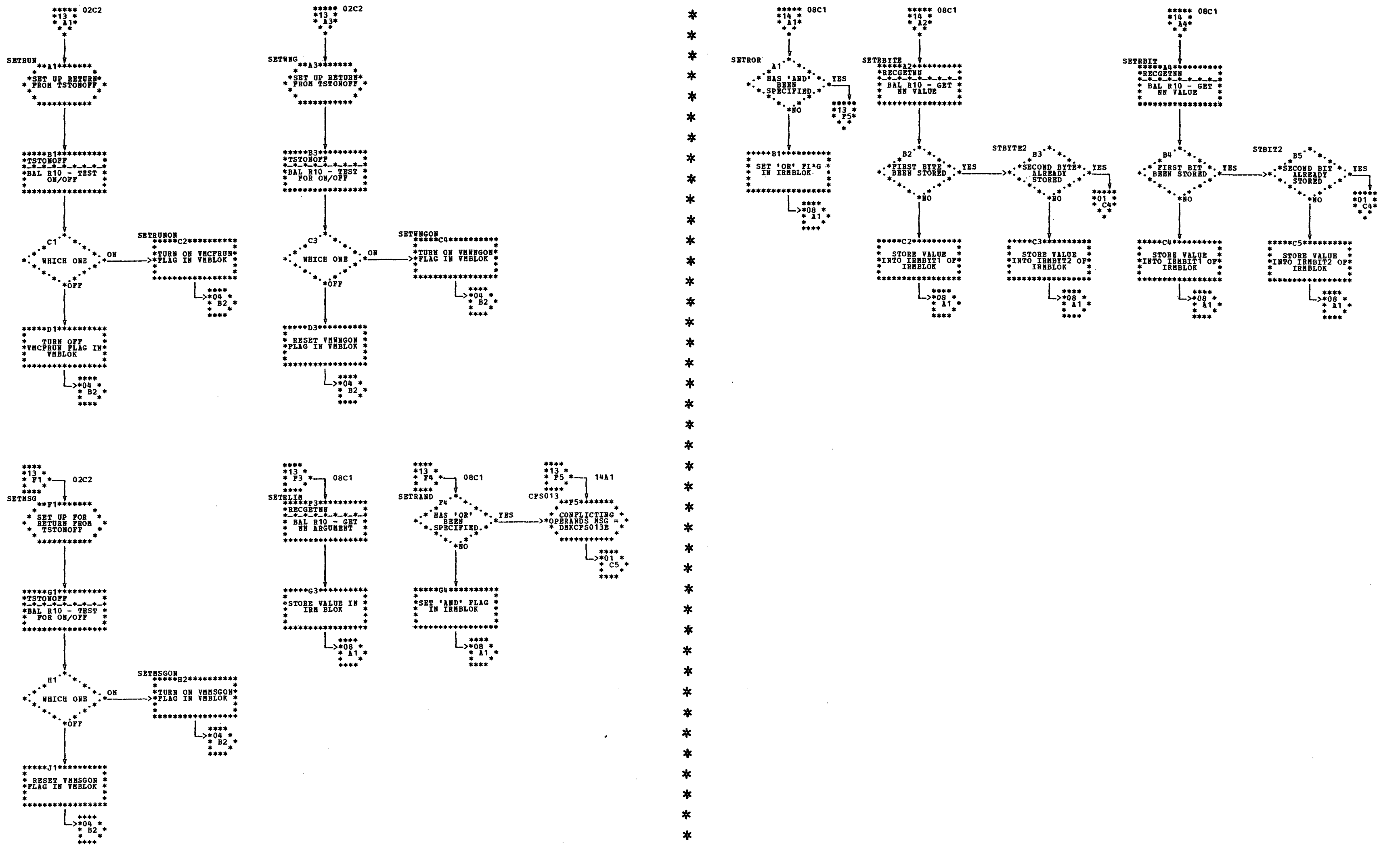


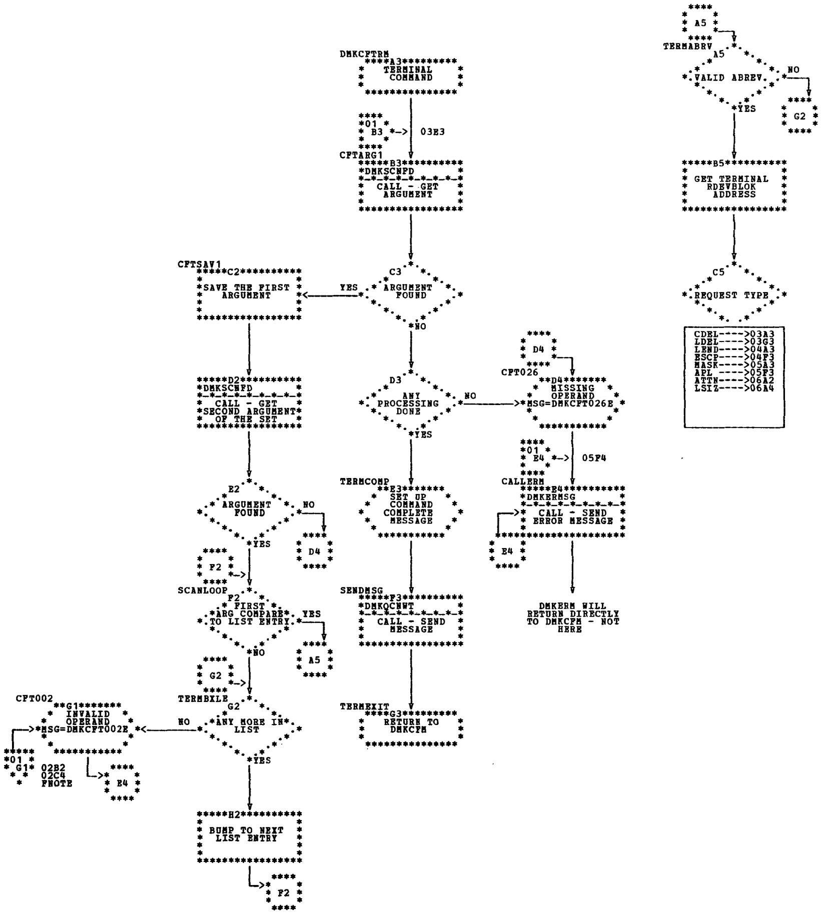




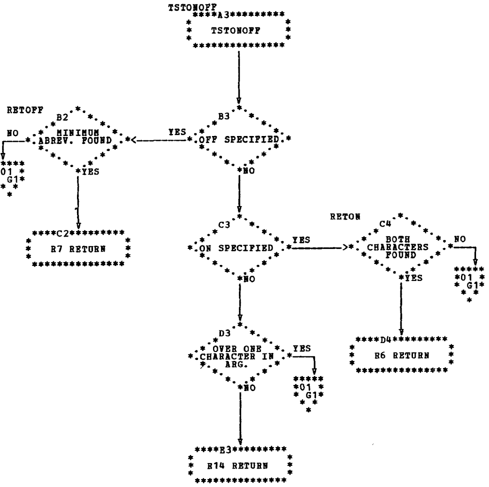
DMKCFS -- Process SET Command (Parts 11 and 12 of 14)

DMKCFS -- Process SET Command (Parts 13 and 14 of 14)

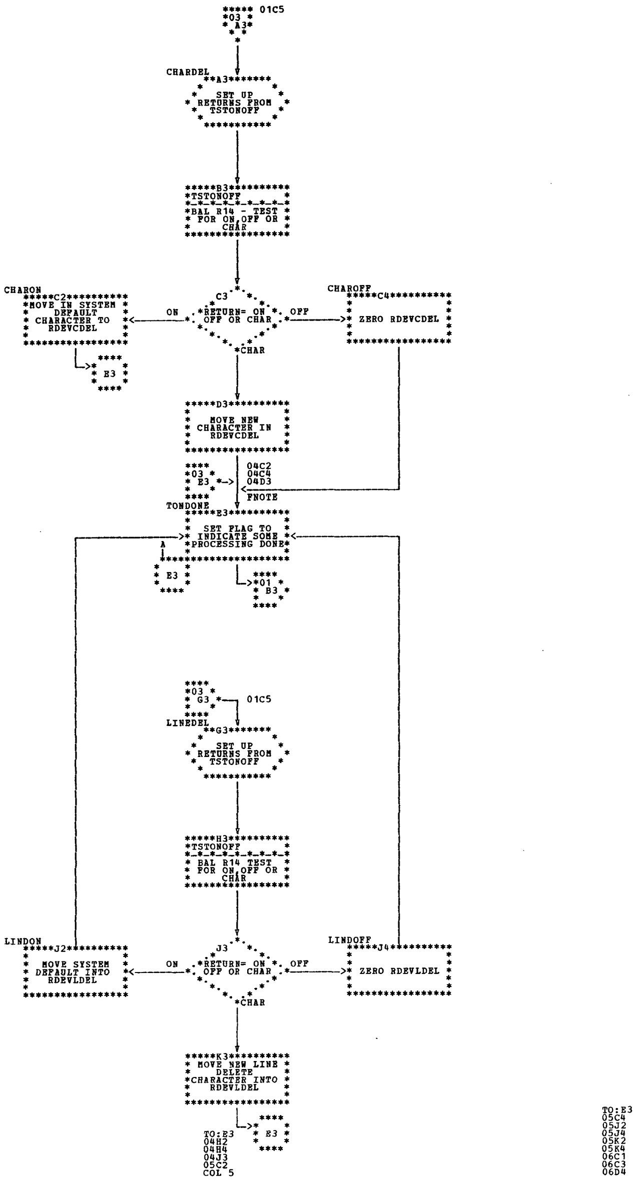


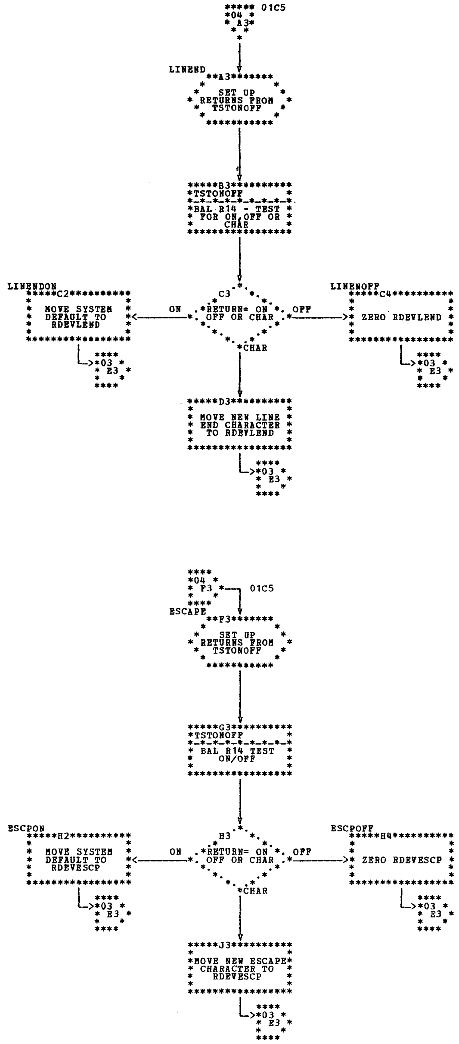


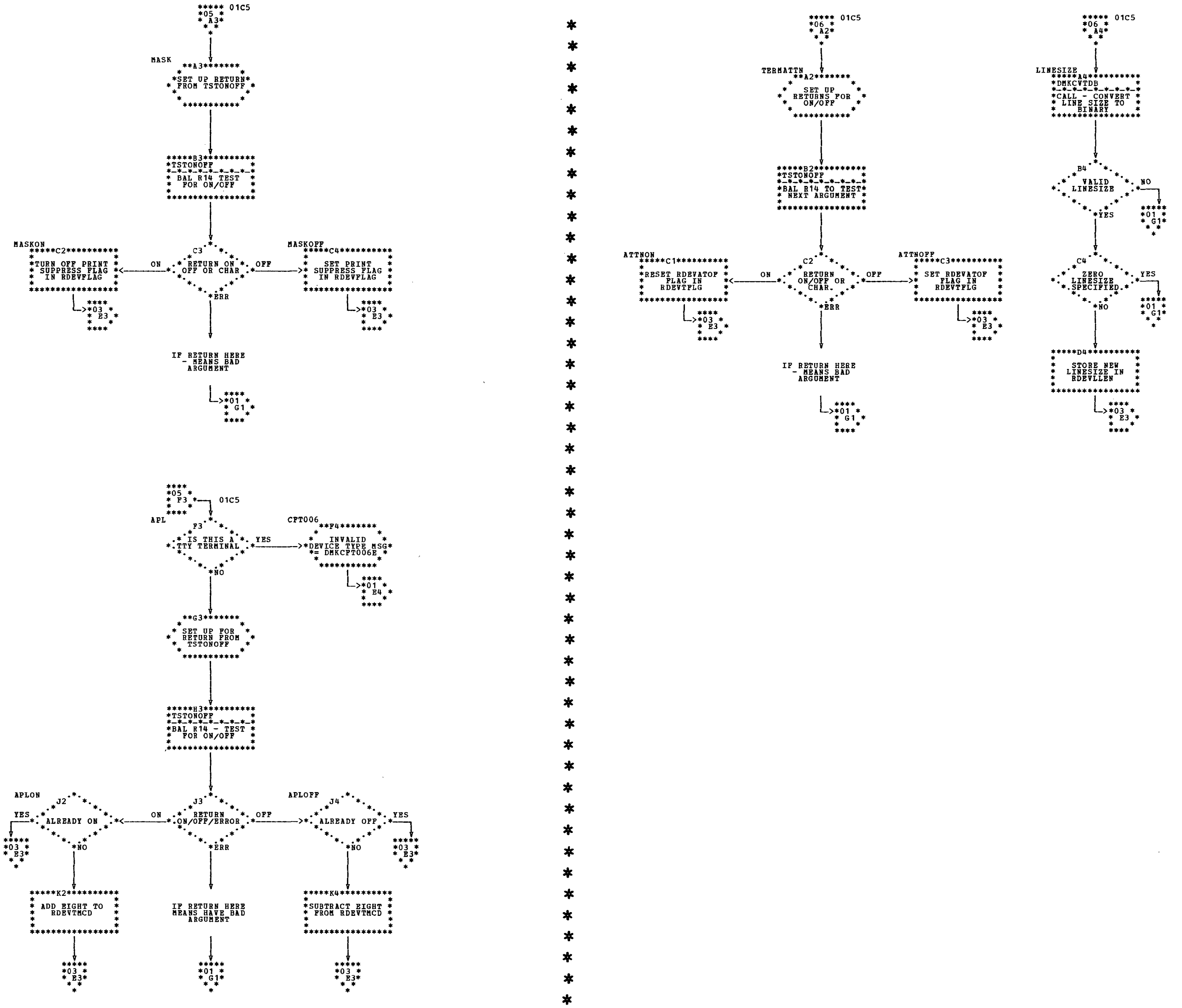
DMKCFT -- Process TERMINAL Command (Parts 1 and 2 of 6)



DMKCFT -- Process TERMINAL Command (Parts 3 and 4 of 6)

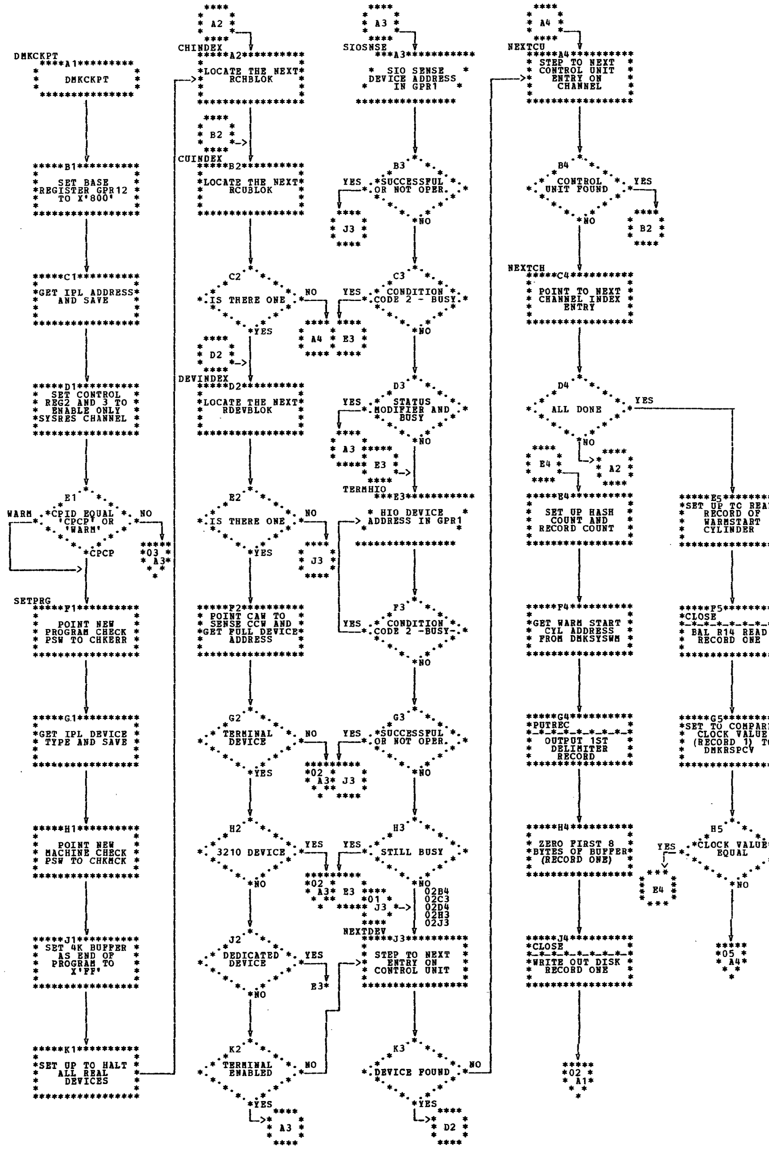

01C5
03
A3
CHARDEL
A3
SET UP RETURNS FROM TSTONOFF
B3
TSTONOFF
BAL R14 - TEST FOR ON,OFF OR CHAR
CHARON
C2
MOVE IN SYSTEM DEFAULT CHARACTER TO RDEVCDEL
E3
C3
RETURN= ON OFF OR CHAR
ON
OFF
CHAR
CHAROFF
C4
ZERO RDEVCDEL
D3
MOVE NEW CHARACTER IN RDEVCDEL
03
E3
04C2
04C4
04D3
FNOTE
TONDONE
E3
SET FLAG TO INDICATE SOME PROCESSING DONE
A
E3
01
B3
03
G3
01C5
LINEDEL
G3
SET UP RETURNS FROM TSTONOFF
H3
TSTONOFF
BAL R14 TEST FOR ON,OFF OR CHAR
LINDON
J2
MOVE SYSTEM DEFAULT INTO RDEVLDEL
J3
RETURN= ON OFF OR CHAR
ON
OFF
CHAR
LINDOFF
J4
ZERO RDEVLDEL
K3
MOVE NEW LINE DELETE CHARACTER INTO RDEVLDEL
E3
TO:E3
04H2
04H4
04J3
05C2
COL 5
TO:E3
05C4
05J2
05J4
05K2
05K4
06C1
06C3
06D4

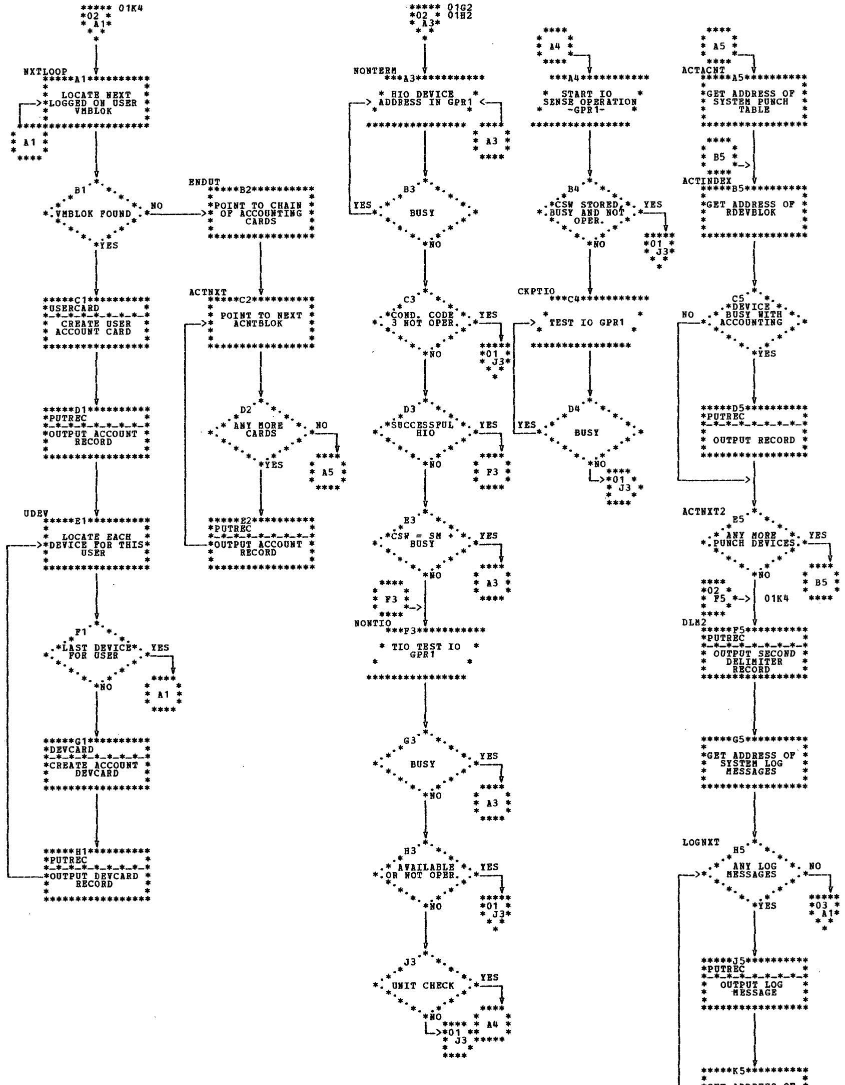


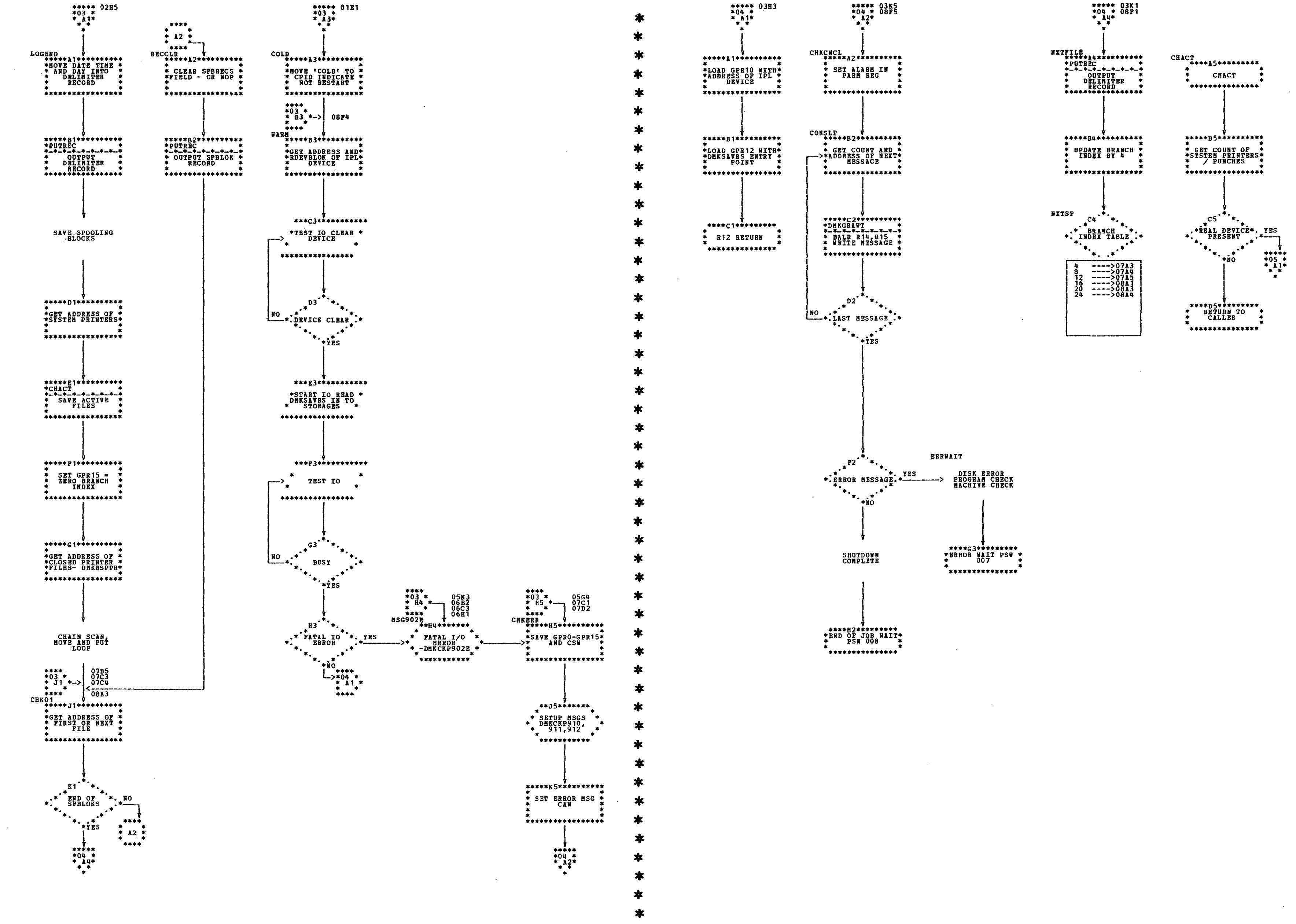

01C5
05 A3
MASK
A3
SET UP RETURN FROM TSTONOFF
B3
TSTONOFF
BAL R14 TEST FOR ON/OFF
C3
RETURN ON OFF OR CHAR
ON
OFF
ERR
MASKON
C2
TURN OFF PRINT SUPPRESS FLAG IN RDEVFLAG
03 E3
MASKOFF
C4
SET PRINT SUPPRESS FLAG IN RDEVFLAG
03 E3
IF RETURN HERE - MEANS BAD ARGUMENT
01 G1
05 F3
01C5
APL
F3
IS THIS A TTY TERMINAL
YES
NO
CPT006
F4
INVALID DEVICE TYPE MSG = DMKCPT006E
01 E4
G3
SET UP FOR RETURN FROM TSTONOFF
H3
TSTONOFF
BAL R14 - TEST FOR ON/OFF
J3
RETURN ON/OFF/ERROR
ON
OFF
ERR
APLON
J2
ALREADY ON
YES
03 E3
NO
K2
ADD EIGHT TO RDEVTMCD
03 E3
IF RETURN HERE MEANS HAVE BAD ARGUMENT
01 G1
APLOFF
J4
ALREADY OFF
YES
03 E3
NO
K4
SUBTRACT EIGHT FROM RDEVTMCD
03 E3
01C5
06 A2
TERMATTN
A2
SET UP RETURNS FOR ON/OFF
B2
TSTONOFF
BAL R14 TO TEST NEXT ARGUMENT
C2
RETURN ON/OFF OR CHAR.
ON
OFF
ERR
ATTNON
C1
RESET RDEVATOF FLAG IN RDEVTFLG
03 E3
ATTNOFF
C3
SET RDEVATOF FLAG IN RDEVTFLG
03 E3
IF RETURN HERE - MEANS BAD ARGUMENT
01 G1
01C5
06 A4
LINESIZE
A4
DMKCVTDB
CALL - CONVERT LINE SIZE TO BINARY
B4
VALID LINESIZE
NO
01 G1
YES
C4
ZERO LINESIZE SPECIFIED
YES
01 G1
NO
D4
STORE NEW LINESIZE IN RDEVLLEN
03 E3

DMKCFT -- Process TERMINAL Command (Parts 5 and 6 of 6)

| DMKCKP -- Checkpoint Processor (Parts 1 and 2 of 8)

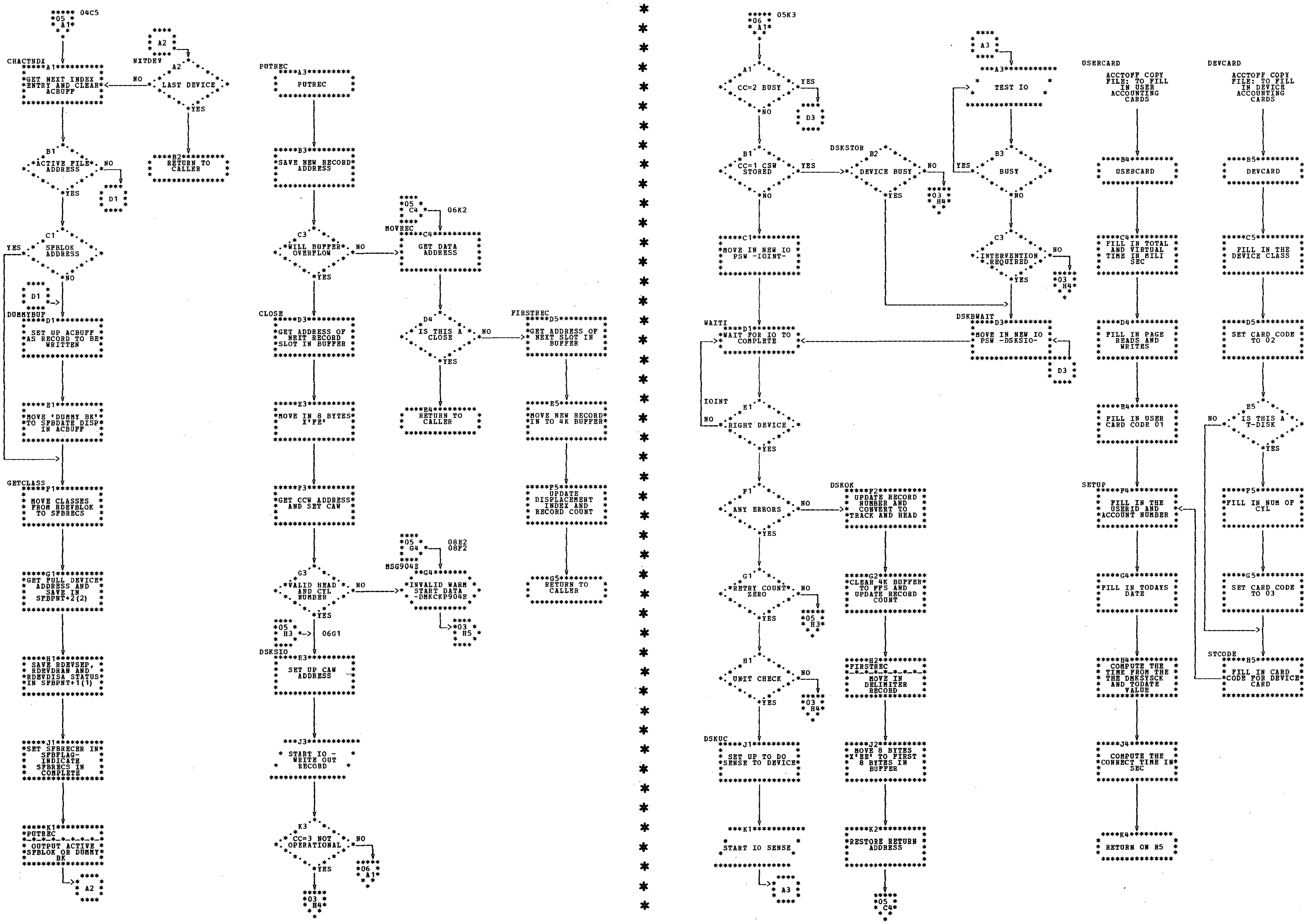

DMKCKPT
A1 DMKCKPT
B1 SET BASE REGISTER GPR12 TO X'800'
C1 GET IPL ADDRESS AND SAVE
D1 SET CONTROL REG2 AND 3 TO ENABLE ONLY SYSRES CHANNEL
WARM
E1 CPID EQUAL 'CPCP' OR 'WARM'
NO
03 A3
CPCP
SETPRG
F1 POINT NEW PROGRAM CHECK PSW TO CHKERR
G1 GET IPL DEVICE TYPE AND SAVE
H1 POINT NEW MACHINE CHECK PSW TO CHKMCK
J1 SET 4K BUFFER AS END OF PROGRAM TO X'FF'
K1 SET UP TO HALT ALL REAL DEVICES
A2
CHINDEX
A2 LOCATE THE NEXT RCHBLOK
B2
CUINDEX
B2 LOCATE THE NEXT RCUBLOK
C2 IS THERE ONE
NO
YES
A4
D2
DEVINDEX
D2 LOCATE THE NEXT RDEVBLOK
E2 IS THERE ONE
NO
YES
J3
F2 POINT CAW TO SENSE CCW AND GET FULL DEVICE ADDRESS
G2 TERMINAL DEVICE
NO
YES
02 A3
H2 3210 DEVICE
YES
NO
02 A3
J2 DEDICATED DEVICE
YES
NO
E3
K2 TERMINAL ENABLED
NO
YES
A3
A3
SIOSNSE
A3 SIO SENSE DEVICE ADDRESS IN GPR1
B3 SUCCESSFUL OR NOT OPER.
YES
NO
J3
C3 CONDITION CODE 2 - BUSY
YES
NO
E3
D3 STATUS MODIFIER AND BUSY
YES
NO
A3
E3
TERMHIO
E3 HIO DEVICE ADDRESS IN GPR1
F3 CONDITION CODE 2 -BUSY
YES
NO
G3 SUCCESSFUL OR NOT OPER.
YES
NO
J3
H3 STILL BUSY
YES
NO
E3
01 J3
02B4 02C3 02D4 02H3 02J3
NEXTDEV
J3 STEP TO NEXT ENTRY ON CONTROL UNIT
K3 DEVICE FOUND
NO
YES
D2
A4
NEXTCU
A4 STEP TO NEXT CONTROL UNIT ENTRY ON CHANNEL
B4 CONTROL UNIT FOUND
YES
NO
B2
NEXTCH
C4 POINT TO NEXT CHANNEL INDEX ENTRY
D4 ALL DONE
YES
NO
A2
E4
E4 SET UP HASH COUNT AND RECORD COUNT
F4 GET WARM START CYL ADDRESS FROM DMKSYSWM
G4 PUTREC OUTPUT 1ST DELIMITER RECORD
H4 ZERO FIRST 8 BYTES OF BUFFER (RECORD ONE)
J4 CLOSE WRITE OUT DISK RECORD ONE
02 A1
E5 SET UP TO READ RECORD OF WARMSTART CYLINDER
F5 CLOSE BAL R14 READ RECORD ONE
G5 SET TO COMPARE CLOCK VALUE (RECORD 1) TO DMKRSPCV
H5 CLOCK VALUE EQUAL
YES
E4
NO
05 A4

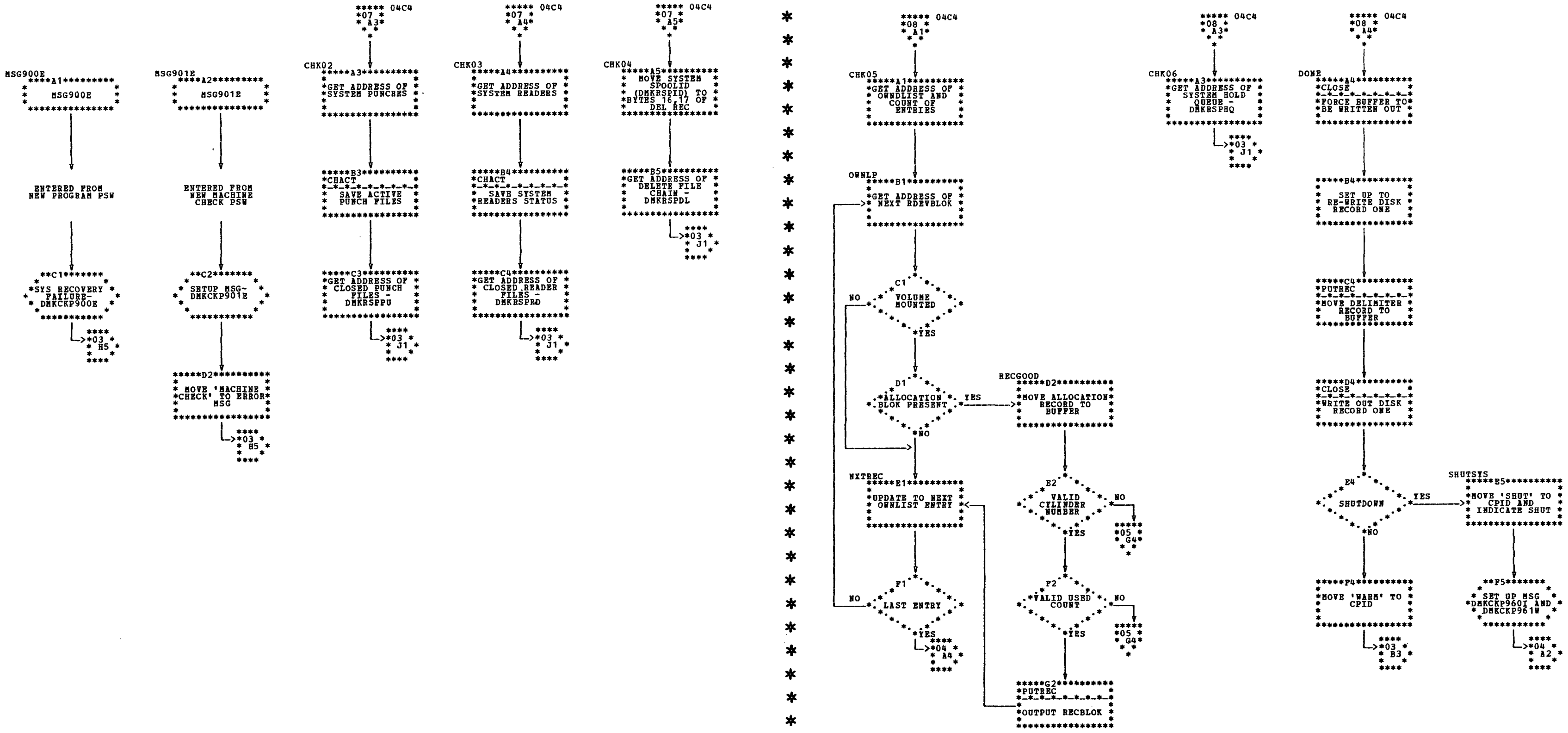

02 A1 01K4
NXTLOOP
A1 LOCATE NEXT LOGGED ON USER VMBLOK
A1
B1 VMBLOK FOUND
NO
YES
UDEV
C1 USERCARD CREATE USER ACCOUNT CARD
D1 PUTREC OUTPUT ACCOUNT RECORD
E1 LOCATE EACH DEVICE FOR THIS USER
F1 LAST DEVICE FOR USER
YES
NO
A1
G1 DEVCARD CREATE ACCOUNT DEVCARD
H1 PUTREC OUTPUT DEVCARD RECORD
ENDUT
B2 POINT TO CHAIN OF ACCOUNTING CARDS
ACTNXT
C2 POINT TO NEXT ACNTBLOK
D2 ANY MORE CARDS
NO
YES
A5
E2 PUTREC OUTPUT ACCOUNT RECORD
02 A3 01G2 01H2
NONTERM
A3 HIO DEVICE ADDRESS IN GPR1
A3
B3 BUSY
YES
NO
C3 COND. CODE 3 NOT OPER.
YES
NO
01 J3
D3 SUCCESSFUL HIO
YES
NO
F3
E3 CSW = SM BUSY
YES
NO
A3
F3
NONTIO
F3 TIO TEST IO GPR1
G3 BUSY
YES
NO
A3
H3 AVAILABLE OR NOT OPER.
YES
NO
01 J3
J3 UNIT CHECK
YES
NO
A4
01 J3
A4
A4 START IO SENSE OPERATION -GPR1-
B4 CSW STORED, BUSY AND NOT OPER.
YES
NO
01 J3
CKPTIO
C4 TEST IO GPR1
D4 BUSY
YES
NO
01 J3
A5
ACTACNT
A5 GET ADDRESS OF SYSTEM PUNCH TABLE
B5
ACTINDEX
B5 GET ADDRESS OF RDEVBLOK
C5 DEVICE BUSY WITH ACCOUNTING
NO
YES
D5 PUTREC OUTPUT RECORD
ACTNXT2
E5 ANY MORE PUNCH DEVICES
YES
NO
B5
02 F5 01K4
DLM2
F5 PUTREC OUTPUT SECOND DELIMITER RECORD
G5 GET ADDRESS OF SYSTEM LOG MESSAGES
LOGNXT
H5 ANY LOG MESSAGES
NO
YES
03 A1
J5 PUTREC OUTPUT LOG MESSAGE
K5 GET ADDRESS OF NEXT LOG MESSAGE



| DMKCKP -- Checkpoint Processor (Parts 3 and 4 of 8)

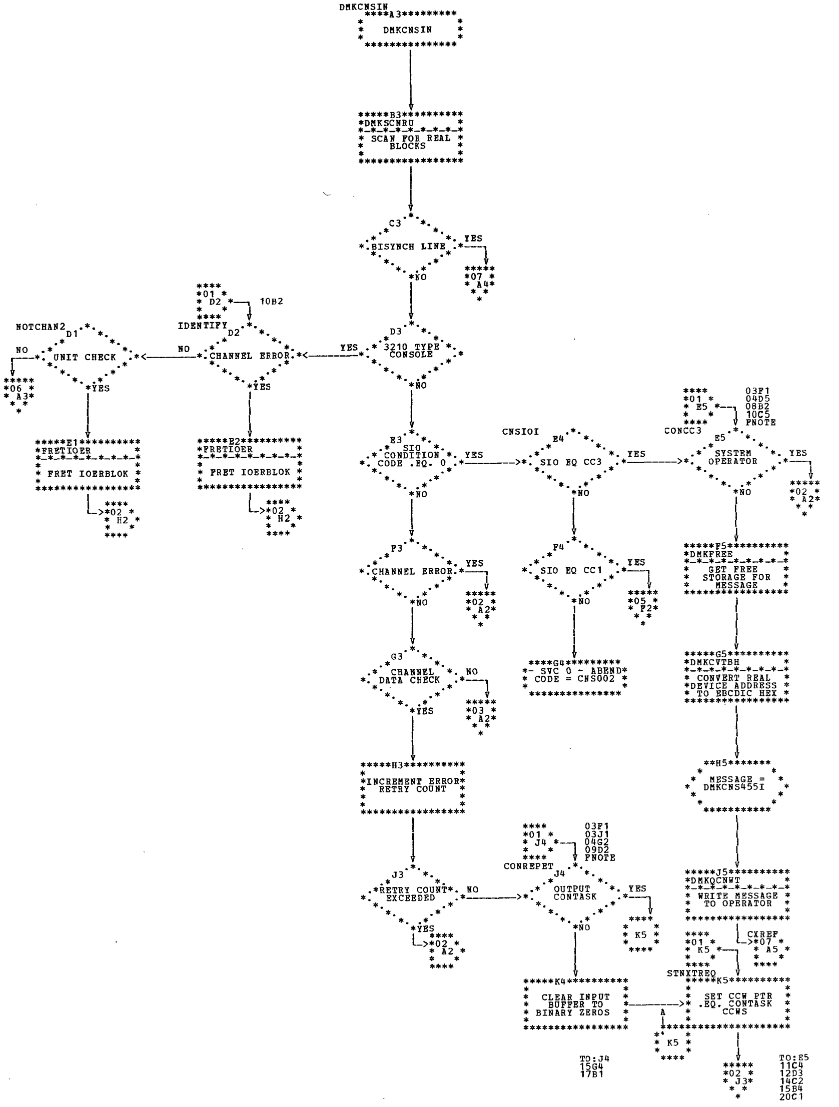
DMKCKP -- Checkpoint Processor (Parts 5 and 6 of 8)

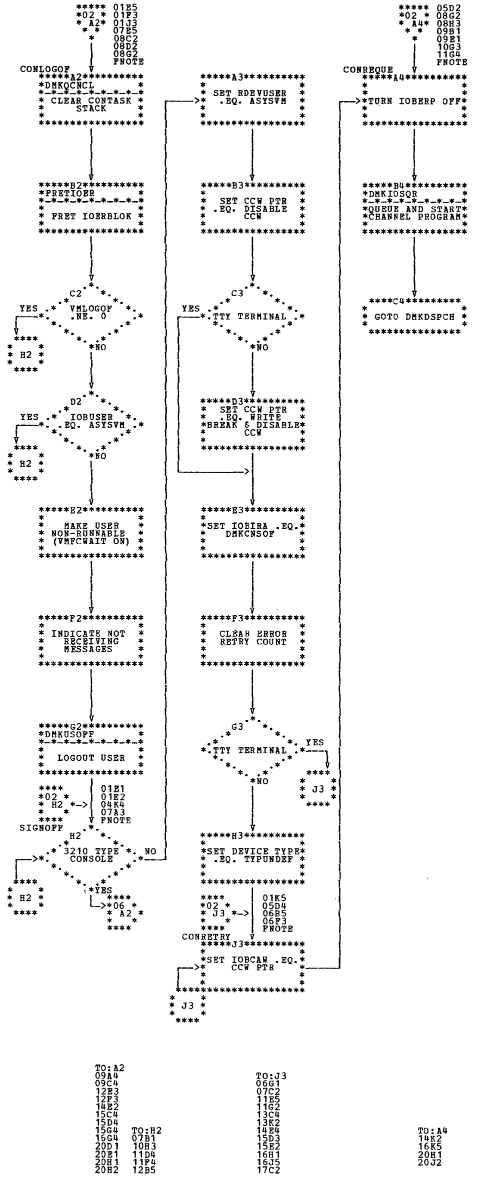

05 A1 04C5
CHACTNDX
A1 GET NEXT INDEX ENTRY AND CLEAR ACBUFF
A2
NXTDEV
A2 LAST DEVICE
NO
YES
B2 RETURN TO CALLER
B1 ACTIVE FILE ADDRESS
NO
D1
YES
C1 SFBLOK ADDRESS
YES
NO
D1
DUMMYBUF
D1 SET UP ACBUFF AS RECORD TO BE WRITTEN
E1 MOVE 'DUMMY BK' TO SFBDATE DISP IN ACBUFF
GETCLASS
F1 MOVE CLASSES FROM RDEVBLOK TO SFBRECS
G1 GET FULL DEVICE ADDRESS AND SAVE IN SFBPNT+2(2)
H1 SAVE RDEVSEP, RDEVDRAN AND RDEVDISA STATUS IN SFBPNT+1(1)
J1 SET SFBRECER IN SFBFLAG- INDICATE SFBRECS IN COMPLETE
K1 PUTREC
OUTPUT ACTIVE SFBLOK OR DUMMY BK
A2
PUTREC
A3 PUTREC
B3 SAVE NEW RECORD ADDRESS
C3 WILL BUFFER OVERFLOW
NO
YES
05 C4 06K2
MOVREC
C4 GET DATA ADDRESS
D4 IS THIS A CLOSE
NO
YES
E4 RETURN TO CALLER
FIRSTREC
D5 GET ADDRESS OF NEXT SLOT IN BUFFER
E5 MOVE NEW RECORD IN TO 4K BUFFER
F5 UPDATE DISPLACEMENT INDEX AND RECORD COUNT
G5 RETURN TO CALLER
CLOSE
D3 GET ADDRESS OF NEXT RECORD SLOT IN BUFFER
E3 MOVE IN 8 BYTES X'FE'
F3 GET CCW ADDRESS AND SET CAW
G3 VALID HEAD AND CYL NUMBER
NO
YES
05 G4 08E2 08F2
MSG904E
G4 INVALID WARM START DATA -DMKCKP904E
03 H5
05 H3 06G1
DSKSIO
H3 SET UP CAW ADDRESS
J3 START IO - WRITE OUT RECORD
K3 CC=3 NOT OPERATIONAL
NO
06 A1
YES
03 H4
06 A1 05K3
A1 CC=2 BUSY
YES
D3
NO
B1 CC=1 CSW STORED
YES
NO
DSKSTOR
B2 DEVICE BUSY
NO
03 H4
YES
C1 MOVE IN NEW IO PSW -IOINT-
WAITI
D1 WAIT FOR IO TO COMPLETE
IOINT
E1 RIGHT DEVICE
NO
YES
F1 ANY ERRORS
NO
YES
G1 RETRY COUNT ZERO
NO
05 H3
YES
H1 UNIT CHECK
NO
03 H4
YES
DSKUC
J1 SET UP TO DO SENSE TO DEVICE
K1 START IO SENSE
A3
DSKOK
F2 UPDATE RECORD NUMBER AND CONVERT TO TRACK AND HEAD
G2 CLEAR 4K BUFFER TO FFS AND UPDATE RECORD COUNT
H2 FIRSTREC
MOVE IN DELIMITER RECORD
J2 MOVE 8 BYTES X'EE' TO FIRST 8 BYTES IN BUFFER
K2 RESTORE RETURN ADDRESS
05 C4
A3
A3 TEST IO
B3 BUSY
YES
NO
C3 INTERVENTION REQUIRED
NO
03 H4
YES
DSKBWAIT
D3 MOVE IN NEW IO PSW -DSKSIO-
D3
USERCARD
ACCTOFF COPY FILE: TO FILL IN USER ACCOUNTING CARDS
B4 USERCARD
C4 FILL IN TOTAL AND VIRTUAL TIME IN MILI SEC
D4 FILL IN PAGE READS AND WRITES
E4 FILL IN USER CARD CODE 01
SETUP
F4 FILL IN THE USERID AND ACCOUNT NUMBER
G4 FILL IN TODAYS DATE
H4 COMPUTE THE TIME FROM THE THE DMKSYSCK AND TODATE VALUE
J4 COMPUTE THE CONNECT TIME IN SEC
K4 RETURN ON R5
DEVCARD
ACCTOFF COPY FILE: TO FILL IN DEVICE ACCOUNTING CARDS
B5 DEVCARD
C5 FILL IN THE DEVICE CLASS
D5 SET CARD CODE TO 02
E5 IS THIS A T-DISK
NO
YES
F5 FILL IN NUM OF CYL
G5 SET CARD CODE TO 03
STCODE
H5 FILL IN CARD CODE FOR DEVICE CARD

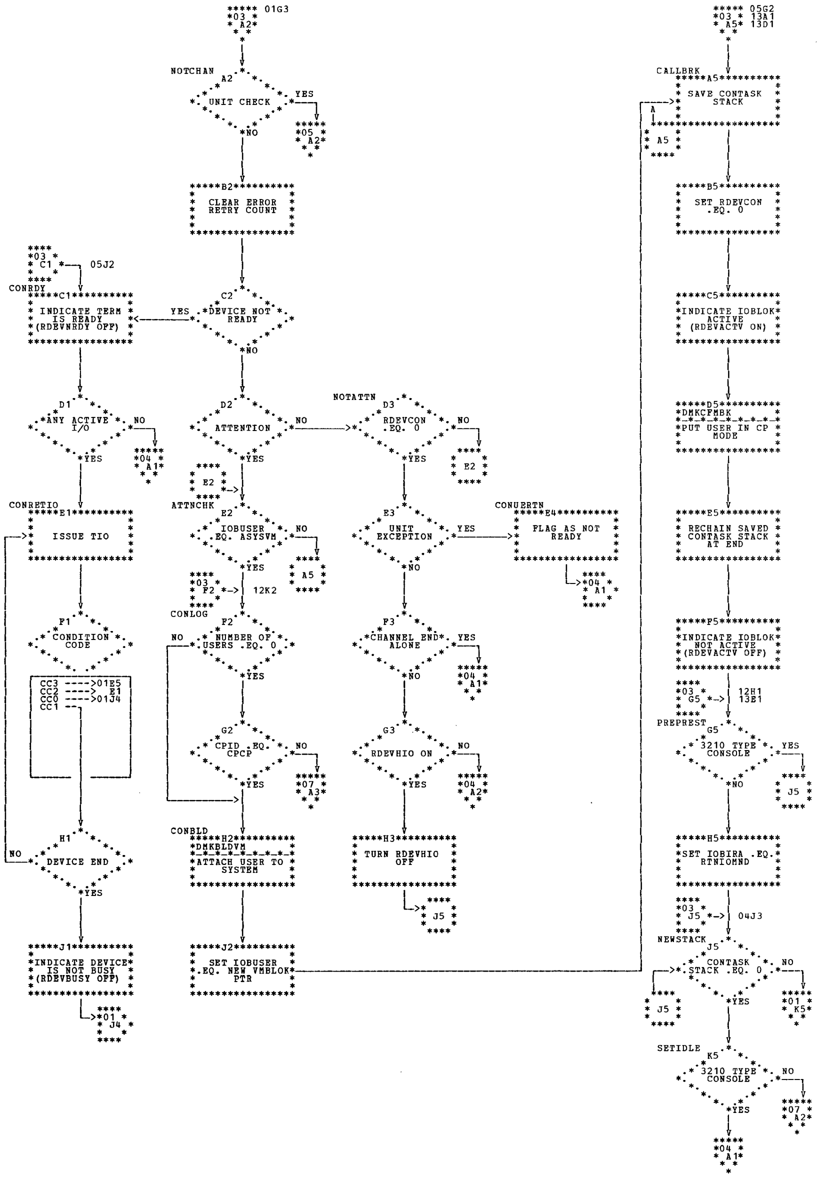

MSG900E
A1
MSG900E
ENTERED FROM NEW PROGRAM PSW
C1
SYS RECOVERY FAILURE- DMKCKP900E
03 H5
MSG901E
A2
MSG901E
ENTERED FROM NEW MACHINE CHECK PSW
C2
SETUP MSG- DMKCKP901E
D2
MOVE 'MACHINE CHECK' TO ERROR MSG
03 H5
04C4
07 A3
CHK02
A3
GET ADDRESS OF SYSTEM PUNCHES
B3
CHACT
SAVE ACTIVE PUNCH FILES
C3
GET ADDRESS OF CLOSED PUNCH FILES - DMKRSPPU
03 J1
04C4
07 A4
CHK03
A4
GET ADDRESS OF SYSTEM READERS
B4
CHACT
SAVE SYSTEM READERS STATUS
C4
GET ADDRESS OF CLOSED READER FILES - DMKRSPRD
03 J1
04C4
07 A5
CHK04
A5
MOVE SYSTEM SPOOLID (DMKRSPID) TO BYTES 16,17 OF DEL REC
B5
GET ADDRESS OF DELETE FILE CHAIN - DMKRSPDL
03 J1
04C4
08 A1
CHK05
A1
GET ADDRESS OF OWNDLIST AND COUNT OF ENTRIES
OWNLP
B1
GET ADDRESS OF NEXT RDEVBLOK
C1
VOLUME MOUNTED
NO
YES
D1
ALLOCATION BLOK PRESENT
YES
NO
RECGOOD
D2
MOVE ALLOCATION RECORD TO BUFFER
NXTREC
E1
UPDATE TO NEXT OWNLIST ENTRY
F1
LAST ENTRY
NO
YES
04 A4
E2
VALID CYLINDER NUMBER
NO
05 G4
YES
F2
VALID USED COUNT
NO
05 G4
YES
G2
PUTREC
OUTPUT RECBLOK
04C4
08 A3
CHK06
A3
GET ADDRESS OF SYSTEM HOLD QUEUE - DMKRSPHQ
03 J1
04C4
08 A4
DONE
A4
CLOSE
FORCE BUFFER TO BE WRITTEN OUT
B4
SET UP TO RE-WRITE DISK RECORD ONE
C4
PUTREC
MOVE DELIMITER RECORD TO BUFFER
D4
CLOSE
WRITE OUT DISK RECORD ONE
E4
SHUTDOWN
YES
NO
SHUTSYS
B5
MOVE 'SHUT' TO CPID AND INDICATE SHUT
F5
SET UP MSG DMKCKP960I AND DMKCKP961W
04 A2
F4
MOVE 'WARM' TO CPID
03 B3

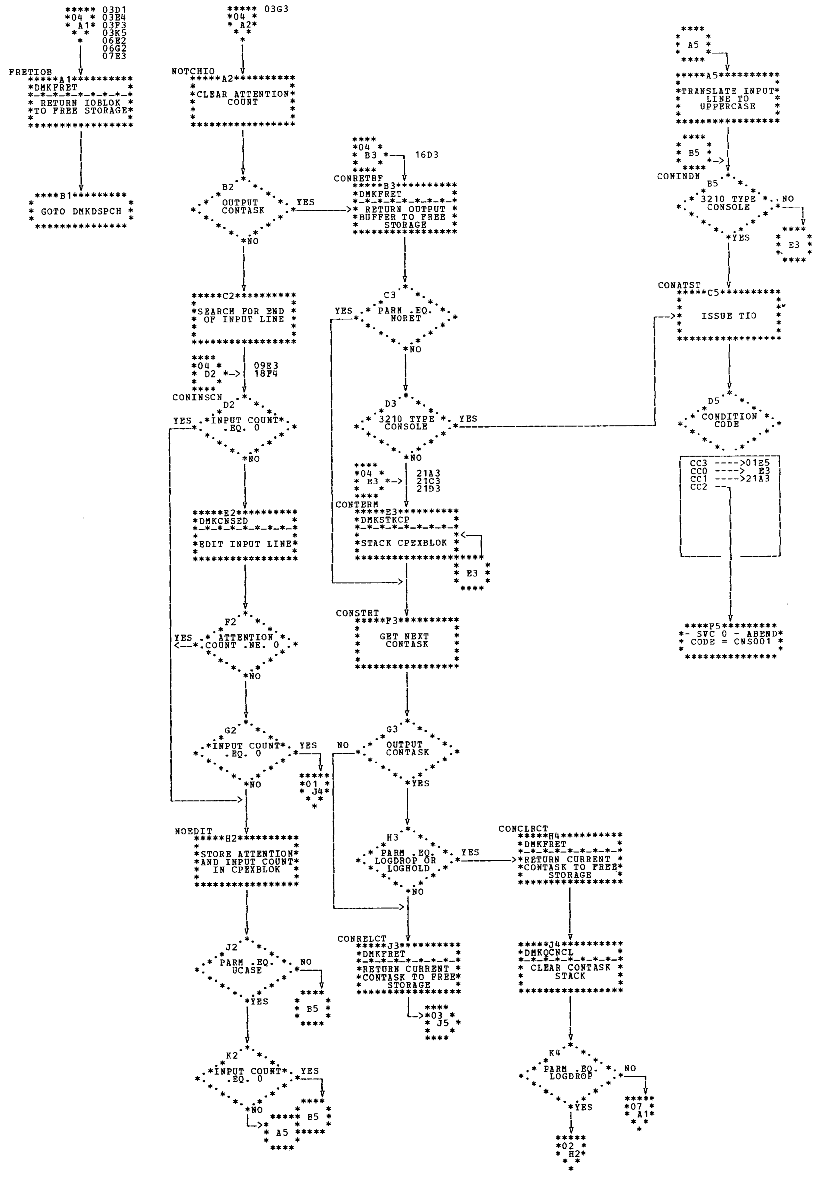
DMKCKP -- Checkpoint Processor (Parts 7 and 8 of 8)

| DMKCNS -- Real Terminal (Console) Manager (Parts 1 and 2 of 23)



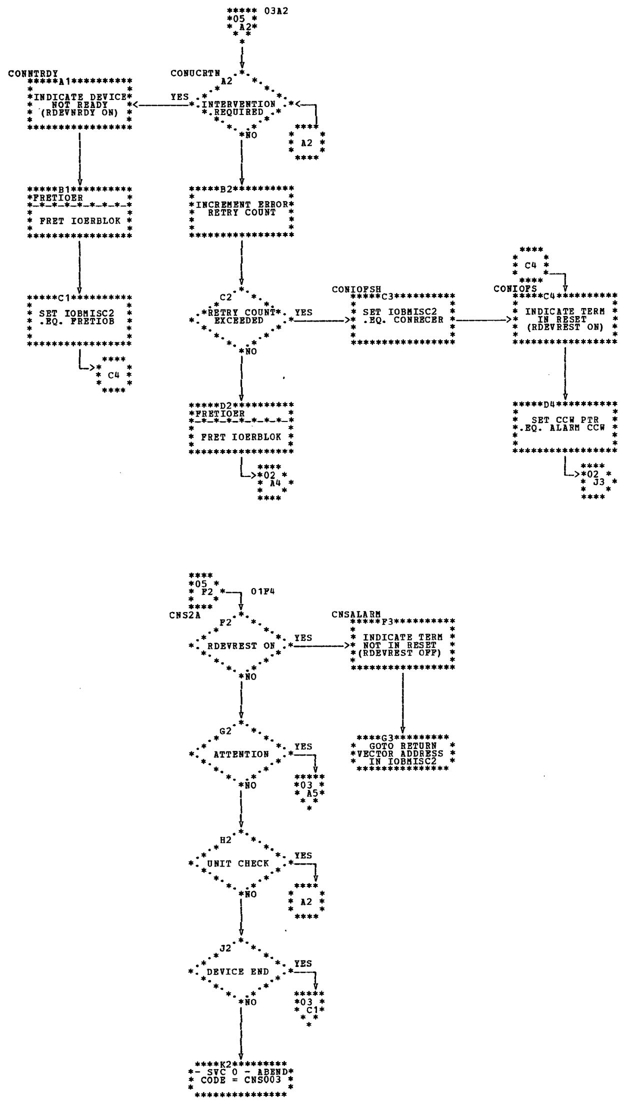


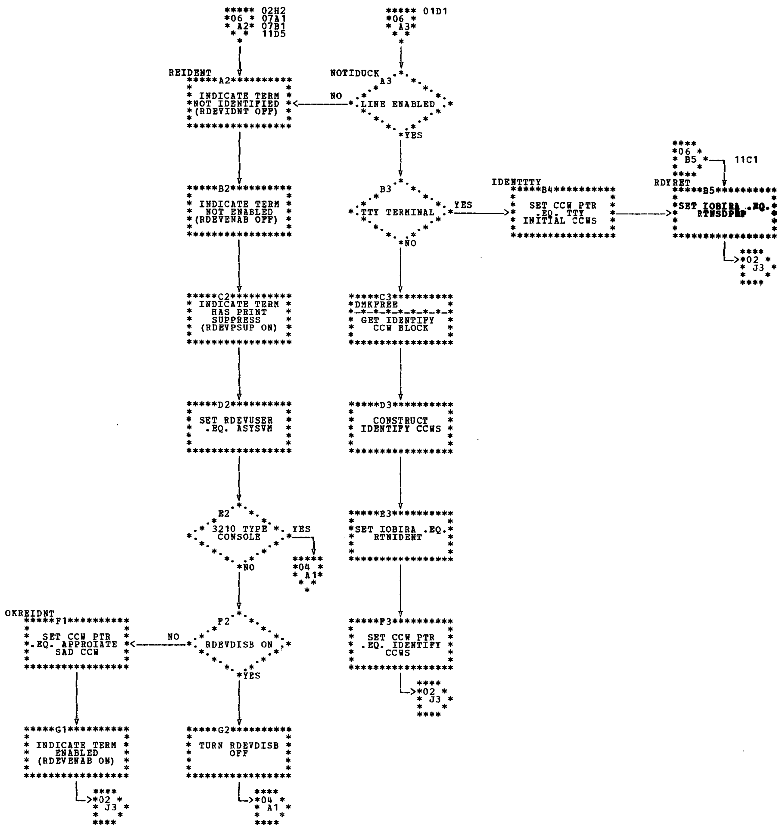


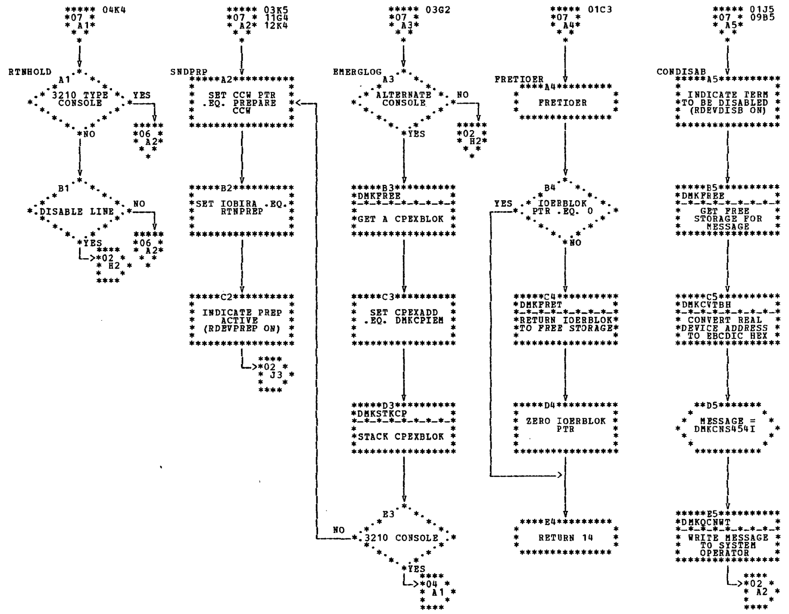


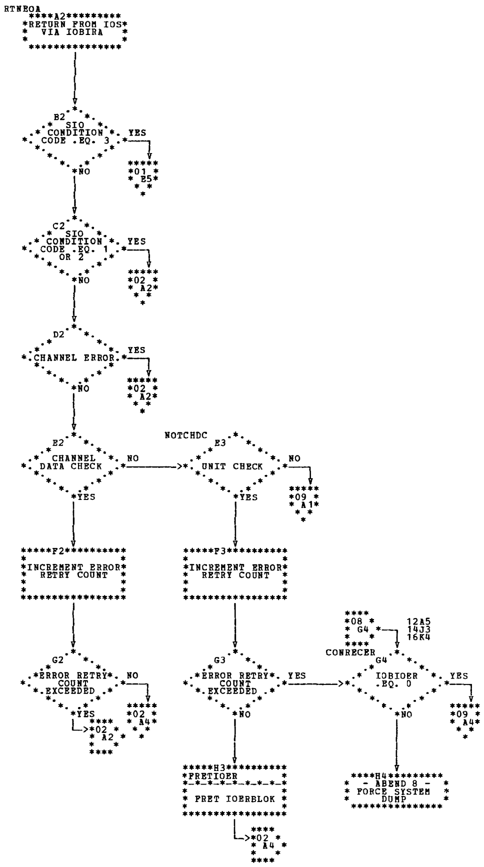
| DMKCNS -- Real Terminal (Console) Manager (Parts 3 and 4 of 23)

| DMKCNS -- Real Terminal (Console) Manager (Parts 5 and 6 of 23)


05 A2 03A2
CONNTRDY
A1
INDICATE DEVICE NOT READY (RDEVNRDY ON)
CONUCRTN
A2
INTERVENTION REQUIRED
YES
NO
A2
B1
FRETIOER
FRET IOERBLOK
B2
INCREMENT ERROR RETRY COUNT
C1
SET IOBMISC2 .EQ. FRETIOB
C4
C2
RETRY COUNT EXCEEDED
YES
NO
CONIOFSH
C3
SET IOBMISC2 .EQ. CONRECER
C4
CONIOFS
C4
INDICATE TERM IN RESET (RDEVREST ON)
D2
FRETIOER
FRET IOERBLOK
02 A4
D4
SET CCW PTR .EQ. ALARM CCW
02 J3
05 F2 01F4
CNS2A
F2
RDEVREST ON
YES
NO
CNSALARM
F3
INDICATE TERM NOT IN RESET (RDEVREST OFF)
G3
GOTO RETURN VECTOR ADDRESS IN IOBMISC2
G2
ATTENTION
YES
NO
03 A5
H2
UNIT CHECK
YES
NO
A2
J2
DEVICE END
YES
NO
03 C1
K2
- SVC 0 - ABEND CODE = CNS003

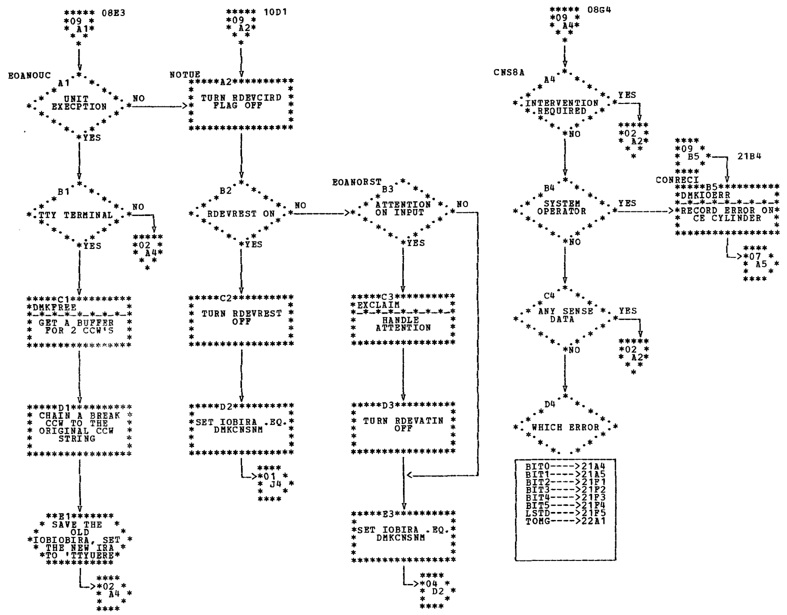


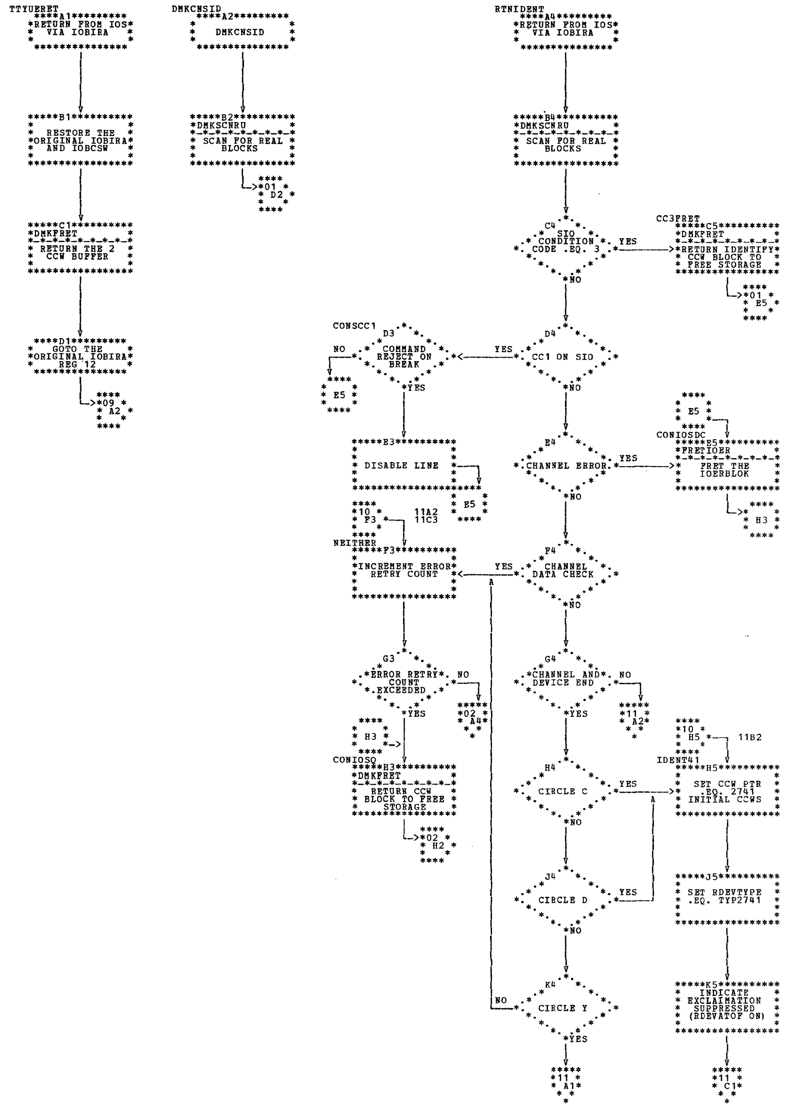


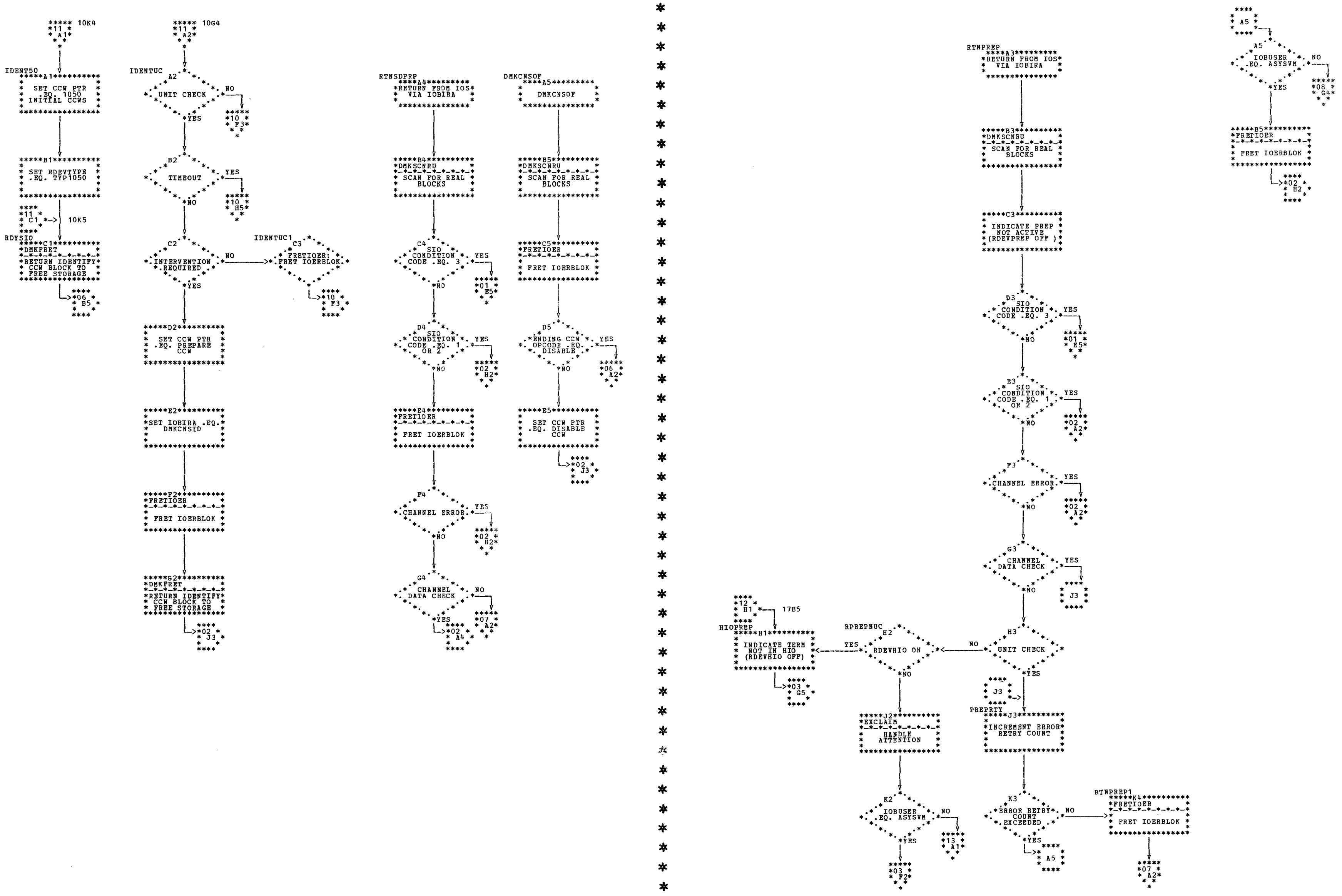


| DMKCNS -- Real Terminal (Console) Manager (Parts 7 and 8 of 23)

| DMKCNS -- Real Terminal (Console) Manager (Parts 9 and 10 of 23)

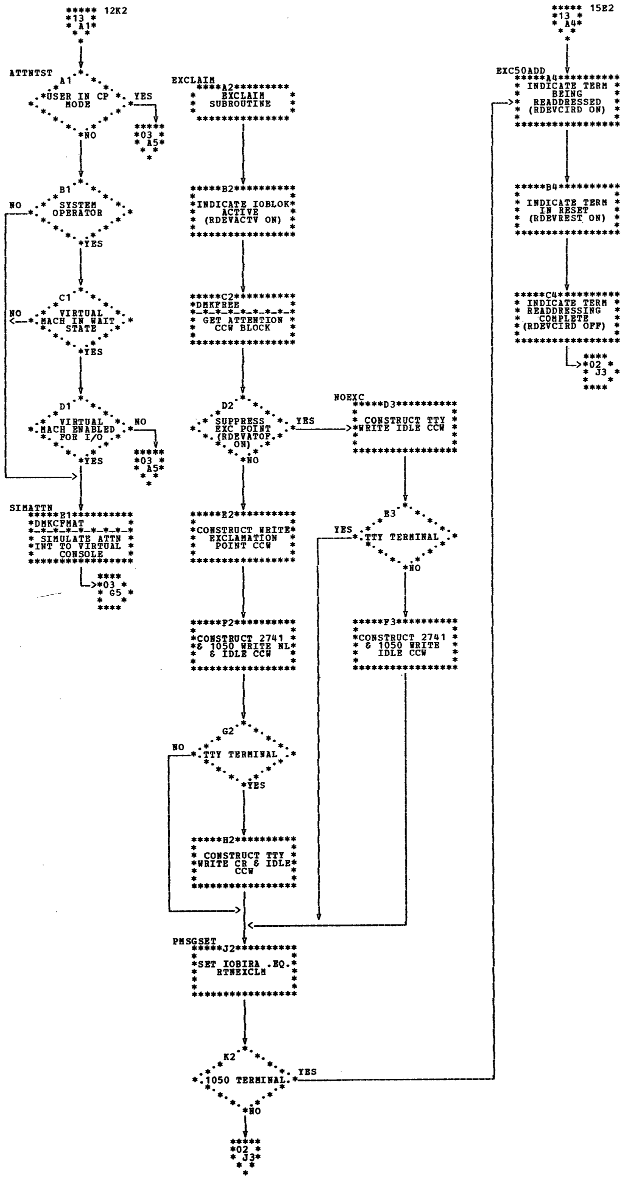




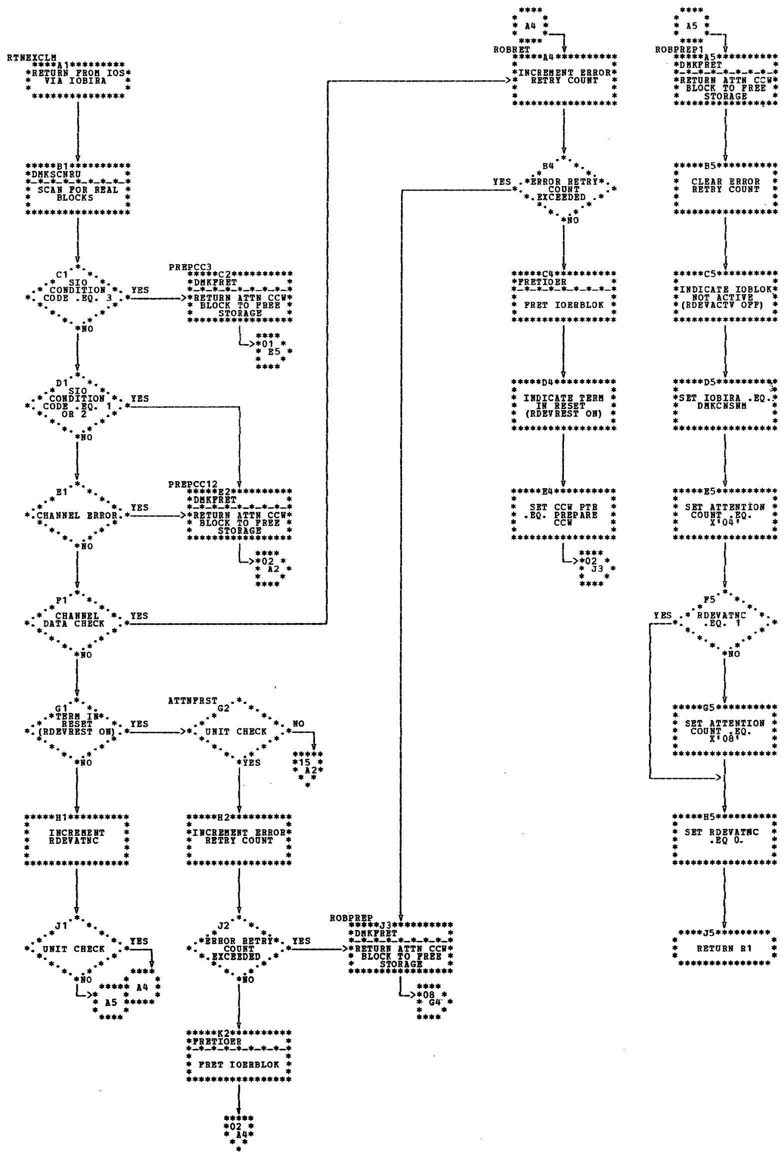


| DMKCNS -- Real Terminal (Console) Manager (Parts 11 and 12 of 23)

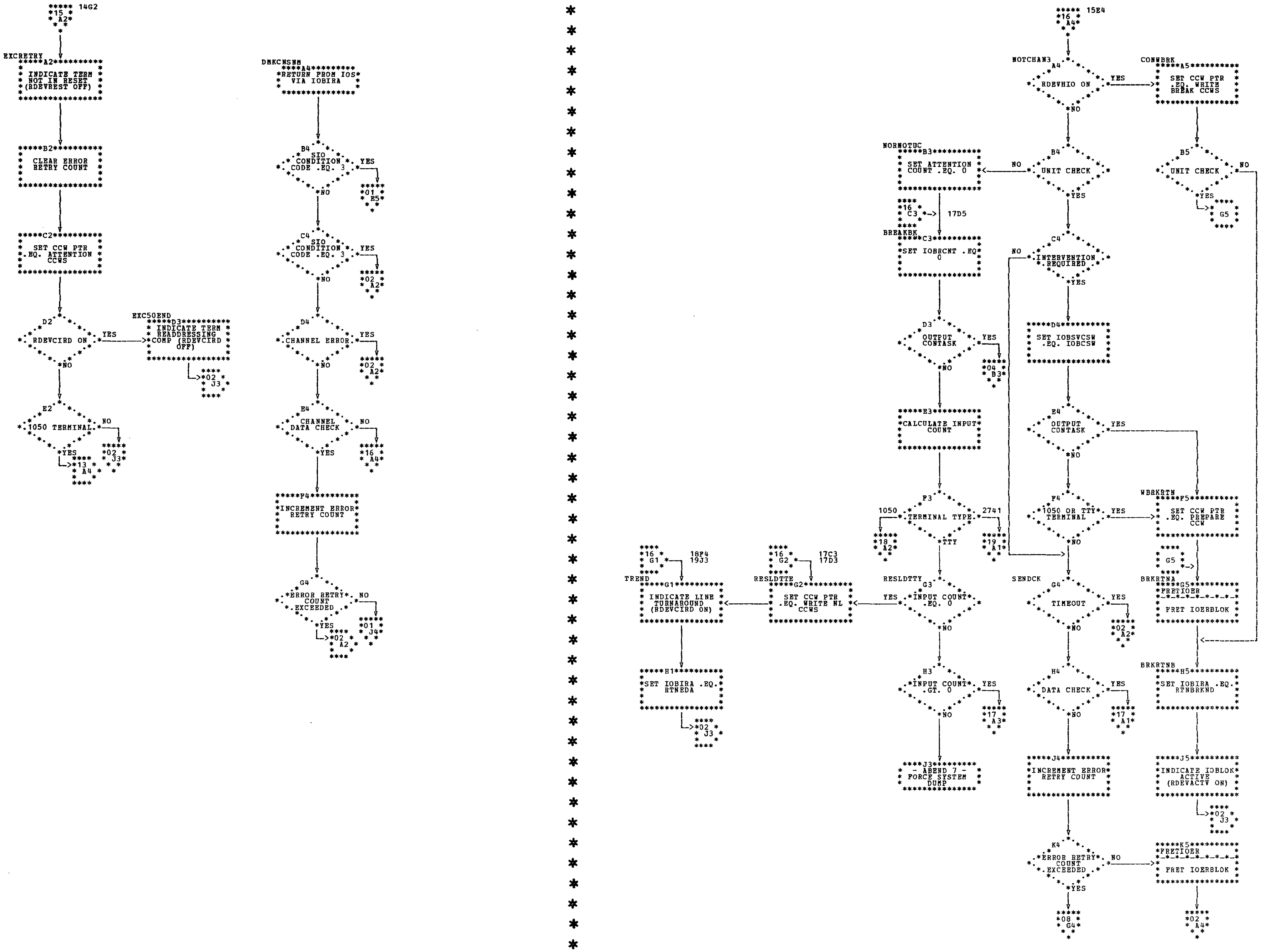
| DMKCNS -- Real Terminal (Console) Manager (Parts 13 and 14 of 23)



13 A1 12K2
ATTNTST A1 USER IN CP MODE
YES
03 A5
NO
B1 SYSTEM OPERATOR
NO
YES
C1 VIRTUAL MACH IN WAIT STATE
NO
YES
D1 VIRTUAL MACH ENABLED FOR I/O
NO
03 A5
YES
SIMATTN E1 DMKCFMAT SIMULATE ATTN INT TO VIRTUAL CONSOLE
03 G5
EXCLAIM A2 EXCLAIM SUBROUTINE
B2 INDICATE IOBLOK ACTIVE (RDEVACTV ON)
C2 DMKFREE GET ATTENTION CCW BLOCK
D2 SUPPRESS EXC POINT (RDEVATOF ON)
YES
NO
NOEXC D3 CONSTRUCT TTY WRITE IDLE CCW
E2 CONSTRUCT WRITE EXCLAMATION POINT CCW
E3 TTY TERMINAL
YES
NO
F2 CONSTRUCT 2741 & 1050 WRITE NL & IDLE CCW
F3 CONSTRUCT 2741 & 1050 WRITE IDLE CCW
G2 TTY TERMINAL
NO
YES
H2 CONSTRUCT TTY WRITE CR & IDLE CCW
PMSGSET J2 SET IOBIRA .EQ. RTNEXCLM
K2 1050 TERMINAL
YES
NO
02 J3
13 A4 15E2
EXC50ADD A4 INDICATE TERM BEING READDRESSED (RDEVCIRD ON)
B4 INDICATE TERM IN RESET (RDEVREST ON)
C4 INDICATE TERM READDRESSING COMPLETE (RDEVCIRD OFF)
02 J3

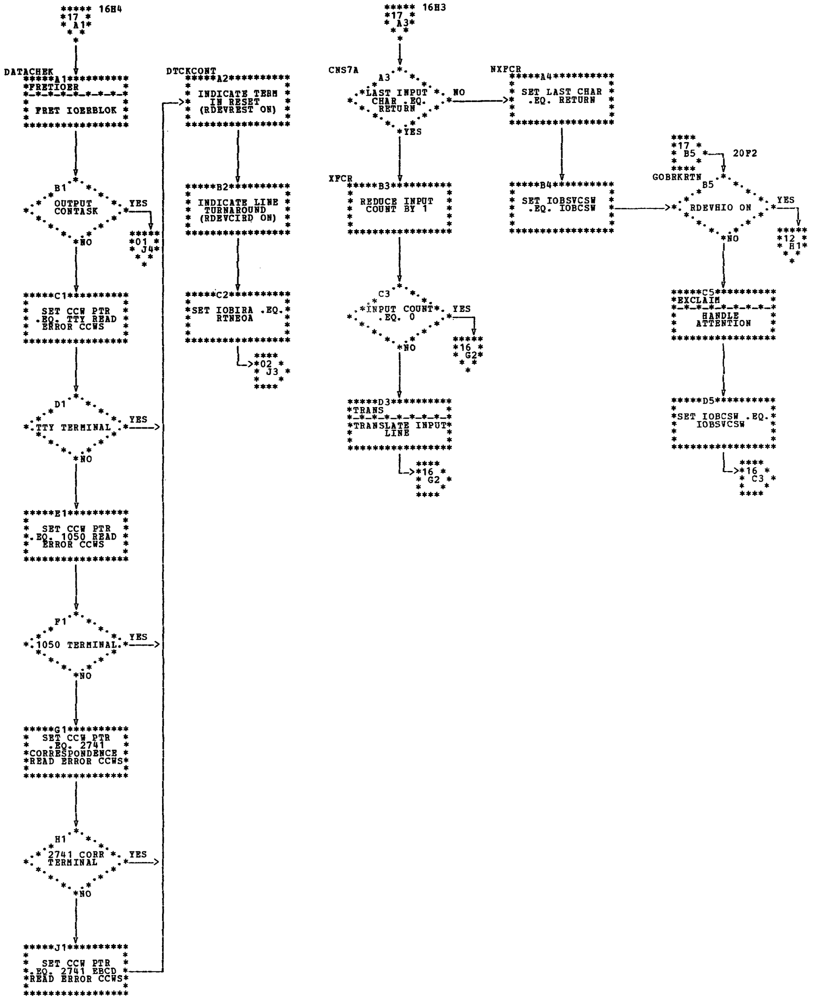


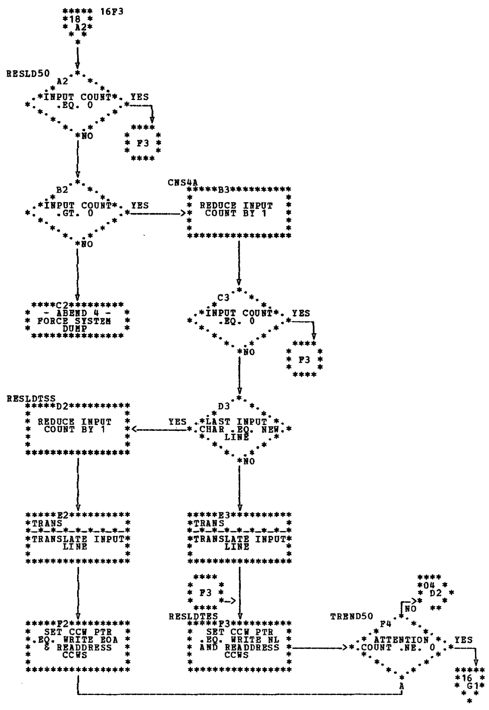
RTNEXCLM A1 RETURN FROM IOS VIA IOBIRA
B1 DMKSCNRU SCAN FOR REAL BLOCKS
C1 SIO CONDITION CODE .EQ. 3
YES
NO
PREPCC3 C2 DMKFRET RETURN ATTN CCW BLOCK TO FREE STORAGE
01 E5
D1 SIO CONDITION CODE .EQ. 1 OR 2
YES
NO
E1 CHANNEL ERROR
YES
NO
PREPCC12 E2 DMKFRET RETURN ATTN CCW BLOCK TO FREE STORAGE
02 A2
F1 CHANNEL DATA CHECK
YES
NO
G1 TERM IN RESET (RDEVREST ON)
YES
NO
ATTNFRST G2 UNIT CHECK
NO
15 A2
YES
H1 INCREMENT RDEVATNC
H2 INCREMENT ERROR RETRY COUNT
J1 UNIT CHECK
YES
A4
NO
A5
J2 ERROR RETRY COUNT EXCEEDED
YES
NO
ROBPREP J3 DMKFRET RETURN ATTN CCW BLOCK TO FREE STORAGE
08 G4
K2 FRETIOER FRET IOERBLOK
02 A4
A4
ROBRET A4 INCREMENT ERROR RETRY COUNT
B4 ERROR RETRY COUNT EXCEEDED
YES
NO
C4 FRETIOER FRET IOERBLOK
D4 INDICATE TERM IN RESET (RDEVREST ON)
E4 SET CCW PTR .EQ. PREPARE CCW
02 J3
A5
ROBPREP1 A5 DMKFRET RETURN ATTN CCW BLOCK TO FREE STORAGE
B5 CLEAR ERROR RETRY COUNT
C5 INDICATE IOBLOK NOT ACTIVE (RDEVACTV OFF)
D5 SET IOBIRA .EQ. DMKCNSNM
E5 SET ATTENTION COUNT .EQ. X'04'
F5 RDEVATNC .EQ. 1
YES
NO
G5 SET ATTENTION COUNT .EQ. X'08'
H5 SET RDEVATNC .EQ 0.
J5 RETURN R1

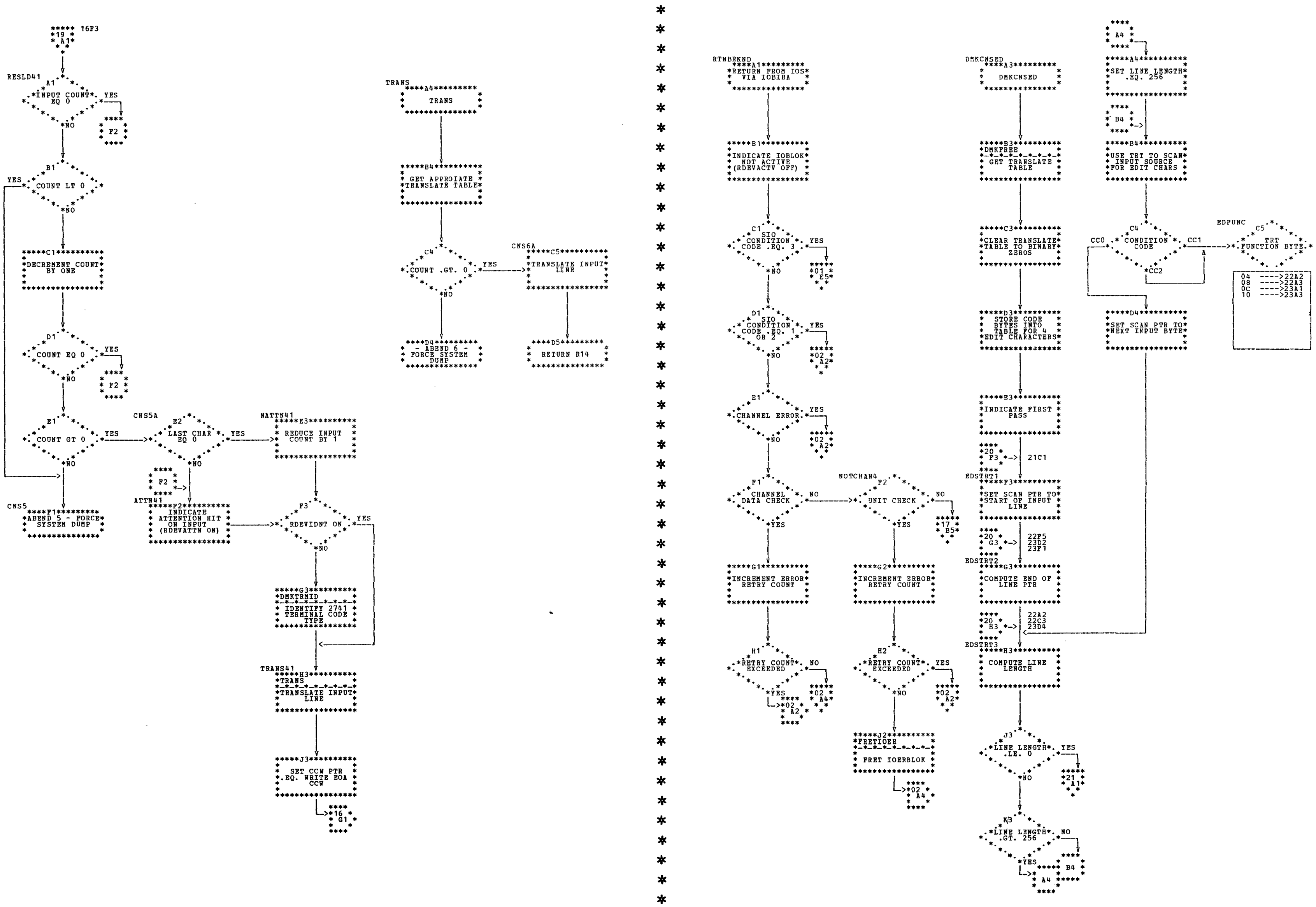


| DMKCNS -- Real Terminal (Console) Manager (Parts 15 and 16 of 23)

DMKCNS -- Real Terminal (Console) Manager (Parts 17 and 18 of 23)

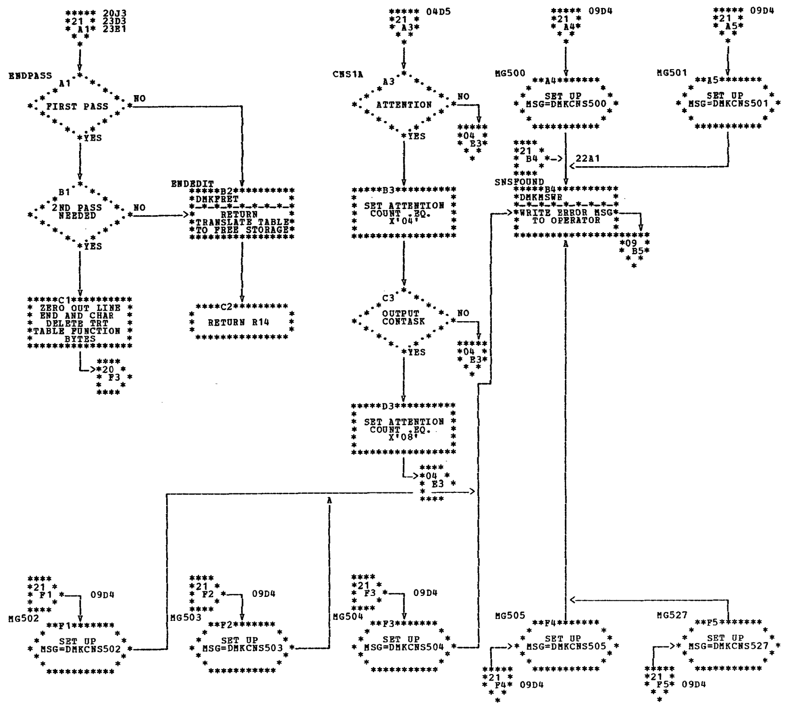


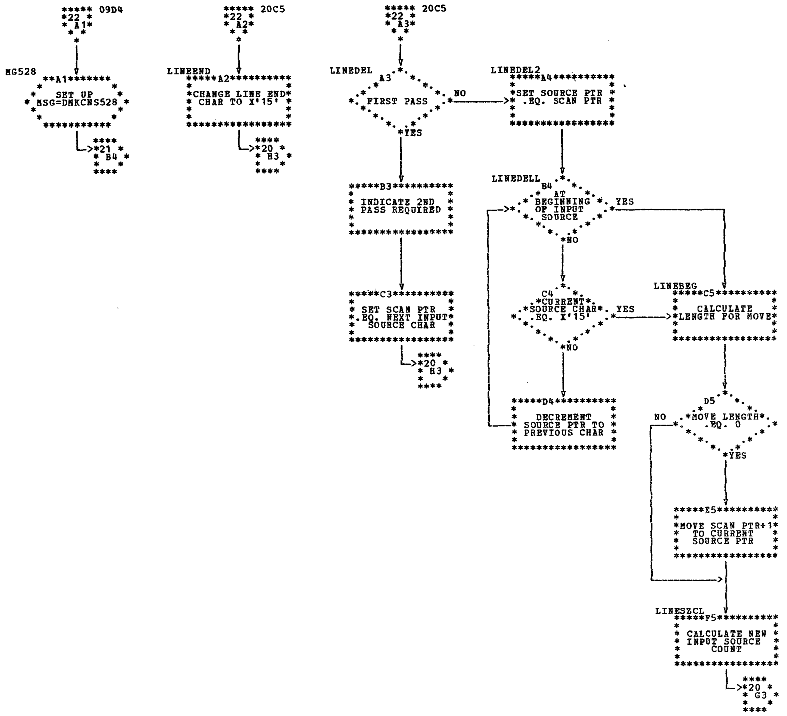


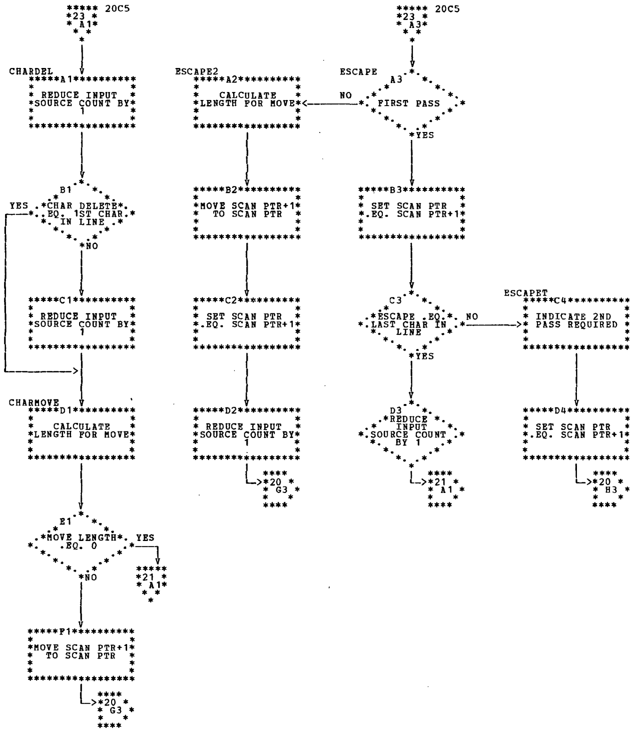


DMKCNS -- Real Terminal (Console) Manager (Parts 19 and 20 of 23)

| DMKCNS -- Real Terminal (Ccnsole) Manager (Parts 21 and 22 of 23)

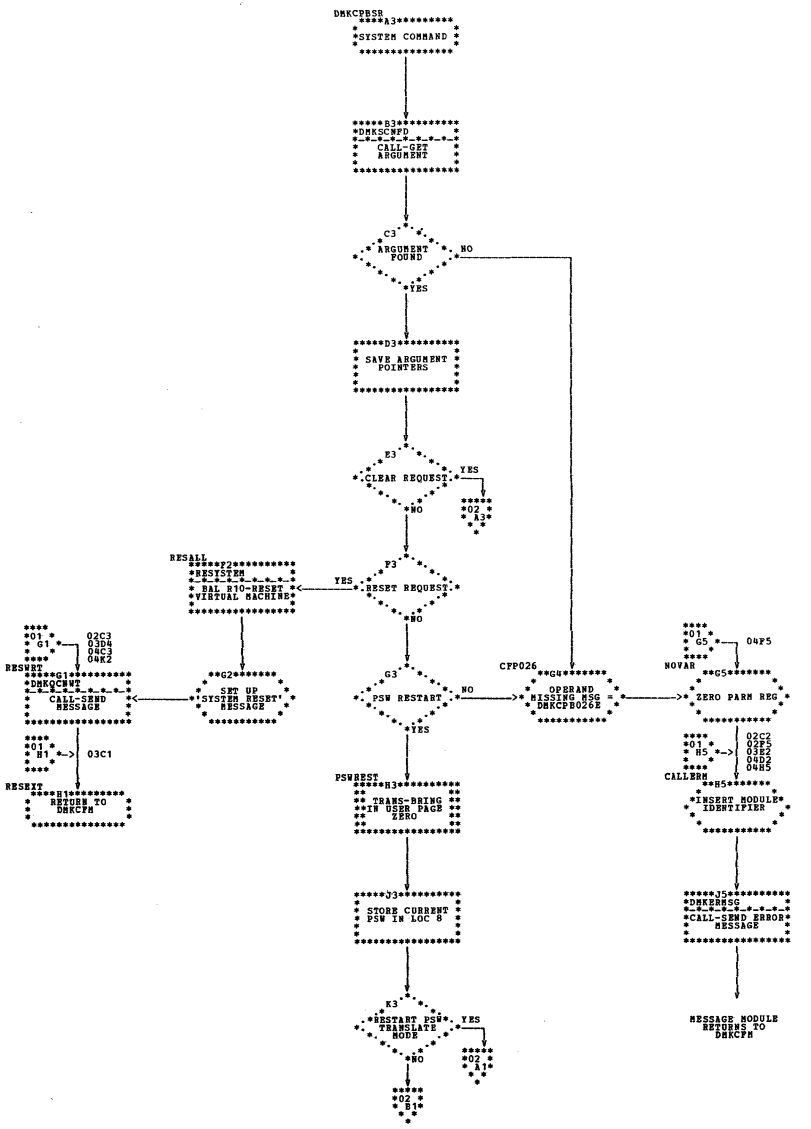


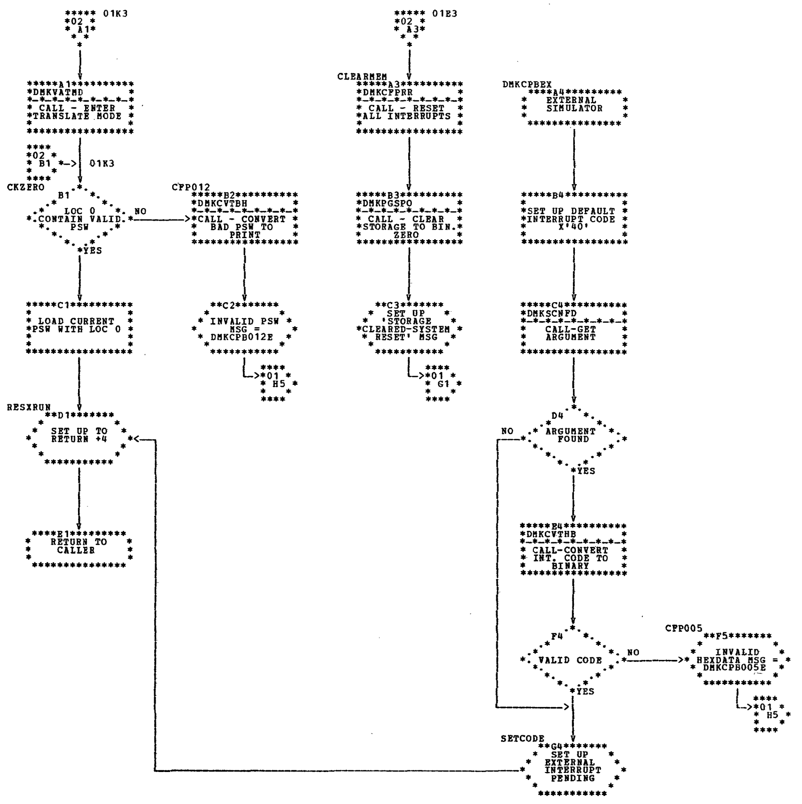

22 A1 09D4
MG528
A1 SET UP MSG=DMKCNS528
21 B4
22 A2 20C5
LINEEND
A2 CHANGE LINE END CHAR TO X'15'
20 H3
22 A3 20C5
LINEDEL
A3 FIRST PASS
NO
YES
LINEDEL2
A4 SET SOURCE PTR .EQ. SCAN PTR
B3 INDICATE 2ND PASS REQUIRED
C3 SET SCAN PTR .EQ. NEXT INPUT SOURCE CHAR
20 H3
LINEDELL
B4 AT BEGINNING OF INPUT SOURCE
YES
NO
C4 CURRENT SOURCE CHAR .EQ. X'15'
YES
NO
D4 DECREMENT SOURCE PTR TO PREVIOUS CHAR
LINEBEG
C5 CALCULATE LENGTH FOR MOVE
D5 MOVE LENGTH .EQ. 0
NO
YES
E5 MOVE SCAN PTR+1 TO CURRENT SOURCE PTR
LINESZCL
F5 CALCULATE NEW INPUT SOURCE COUNT
20 G3

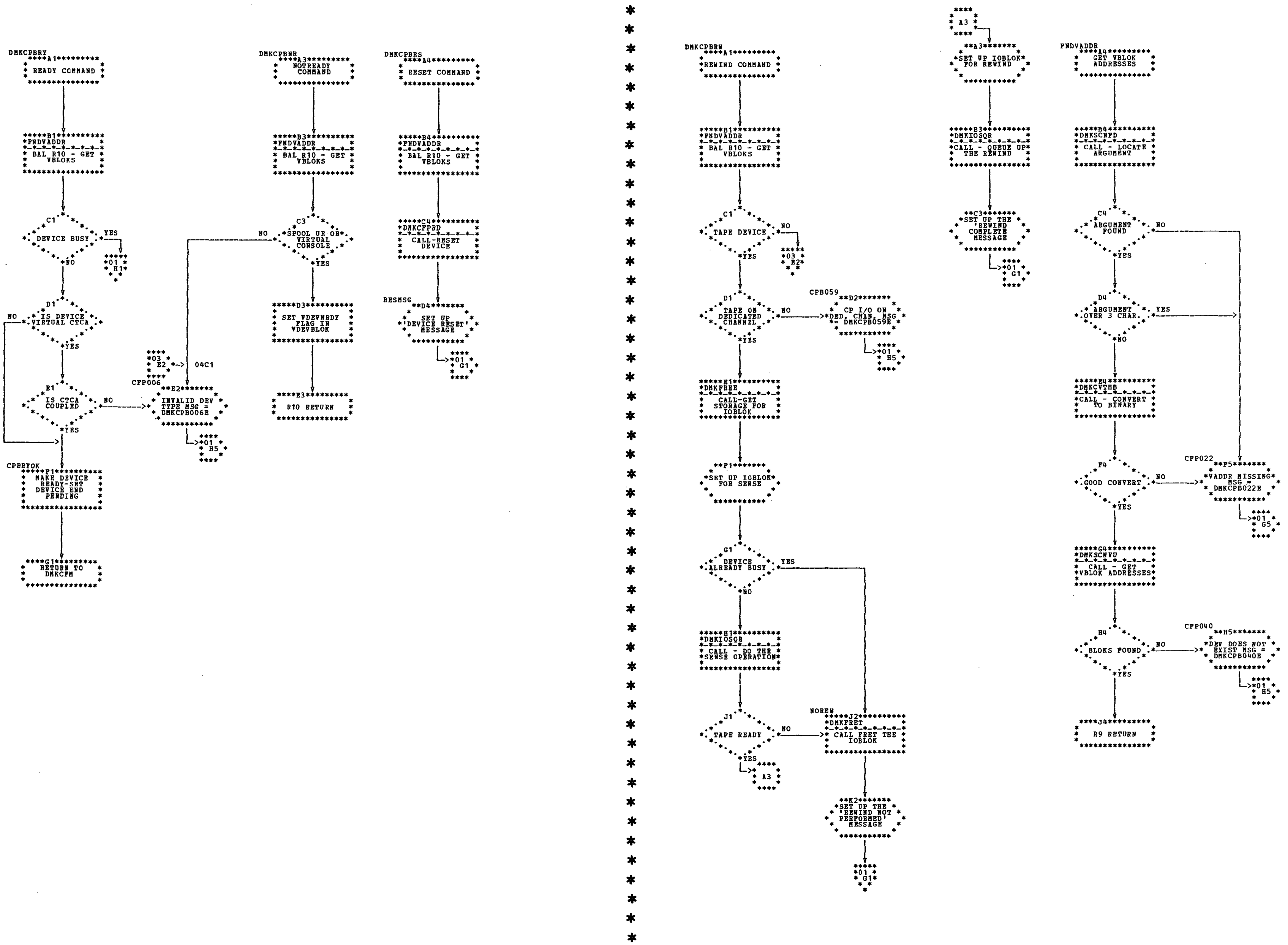

20C5
23
A1
CHARDEL
A1
REDUCE INPUT SOURCE COUNT BY 1
B1
CHAR DELETE .EQ. 1ST CHAR IN LINE
YES
NO
C1
REDUCE INPUT SOURCE COUNT BY 1
CHARMOVE
D1
CALCULATE LENGTH FOR MOVE
E1
MOVE LENGTH .EQ. 0
YES
21
A1
NO
F1
MOVE SCAN PTR+1 TO SCAN PTR
20
G3
ESCAPE2
A2
CALCULATE LENGTH FOR MOVE
B2
MOVE SCAN PTR+1 TO SCAN PTR
C2
SET SCAN PTR .EQ. SCAN PTR+1
D2
REDUCE INPUT SOURCE COUNT BY 1
20
G3
20C5
23
A3
ESCAPE
A3
FIRST PASS
NO
YES
B3
SET SCAN PTR .EQ. SCAN PTR+1
C3
ESCAPE .EQ. LAST CHAR IN LINE
NO
YES
D3
REDUCE INPUT SOURCE COUNT BY 1
21
A1
ESCAPET
C4
INDICATE 2ND PASS REQUIRED
D4
SET SCAN PTR .EQ. SCAN PTR+1
20
H3

DMKCNS -- Real Terminal (Console) Manager (Part 23 of 23)

| DMKCPB -- Process EXTERNAL, NOTREADY, READY, REWIND, and SYSTEM Commands (Parts 1 and 2 of 4)

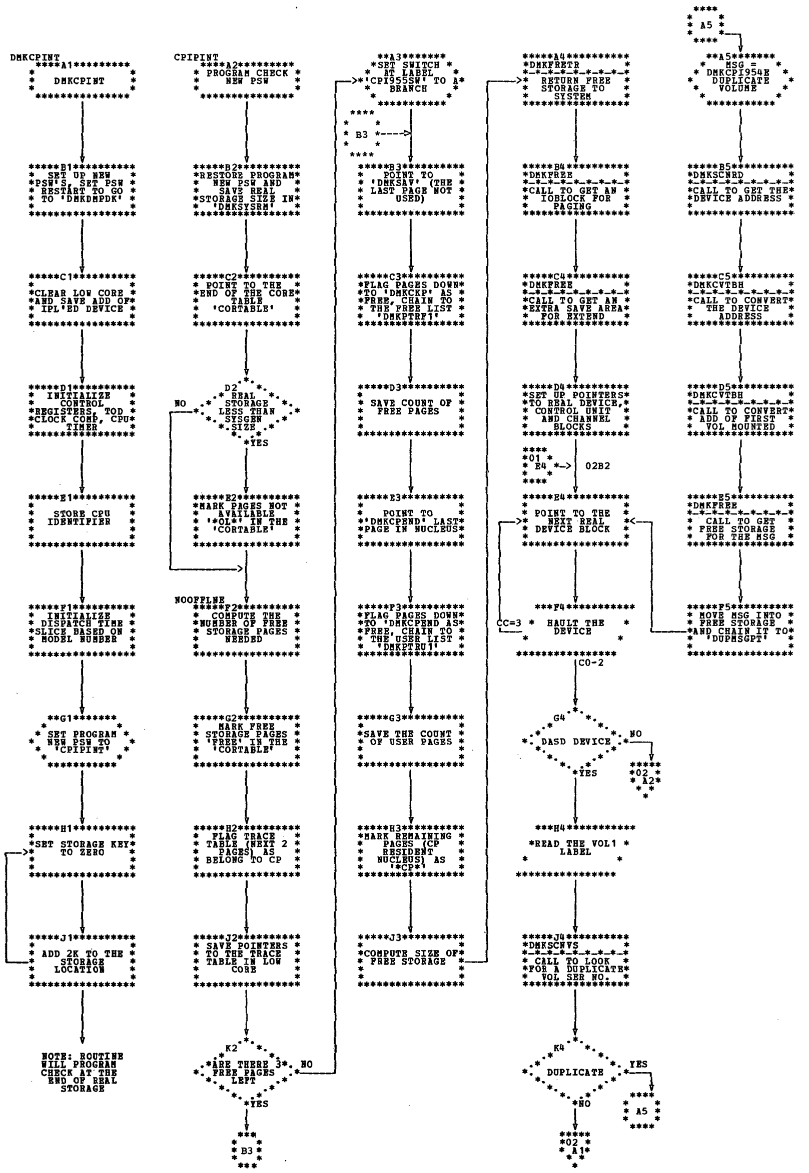


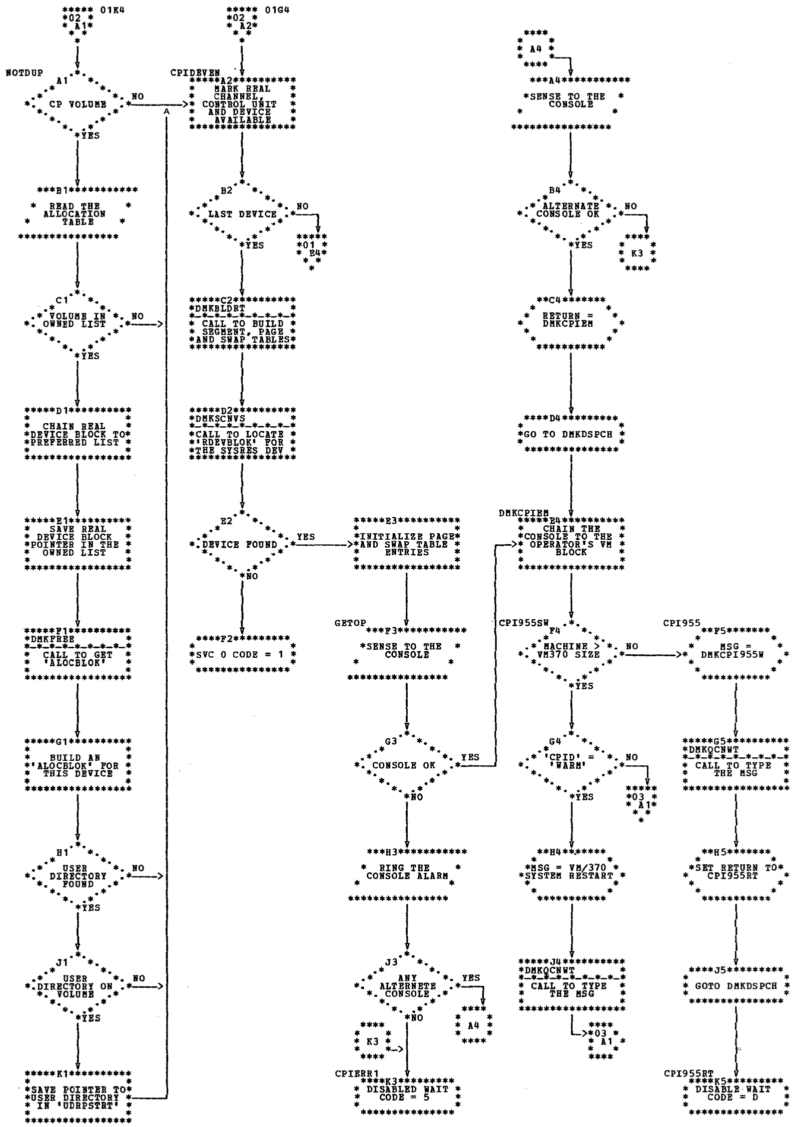

01K3
02 A1
A1
DMKVATMD
CALL - ENTER TRANSLATE MODE
02 B1
01K3
CKZERO
B1
LOC 0 CONTAIN VALID PSW
NO
YES
CFP012
B2
DMKCVTBH
CALL - CONVERT BAD PSW TO PRINT
C2
INVALID PSW MSG = DMKCPB012E
01 H5
C1
LOAD CURRENT PSW WITH LOC 0
RESXRUN
D1
SET UP TO RETURN +4
E1
RETURN TO CALLER
01E3
02 A3
CLEARMEM
A3
DMKCPPRR
CALL - RESET ALL INTERRUPTS
B3
DMKPGSPO
CALL - CLEAR STORAGE TO BIN. ZERO
C3
SET UP 'STORAGE CLEARED-SYSTEM RESET' MSG
01 G1
DMKCPBEX
A4
EXTERNAL SIMULATOR
B4
SET UP DEFAULT INTERRUPT CODE X'40'
C4
DMKSCNFD
CALL-GET ARGUMENT
D4
ARGUMENT FOUND
NO
YES
E4
DMKCVTHB
CALL-CONVERT INT. CODE TO BINARY
F4
VALID CODE
NO
YES
CFP005
F5
INVALID HEXDATA MSG = DMKCPB005E
01 H5
SETCODE
G4
SET UP EXTERNAL INTERRUPT PENDING

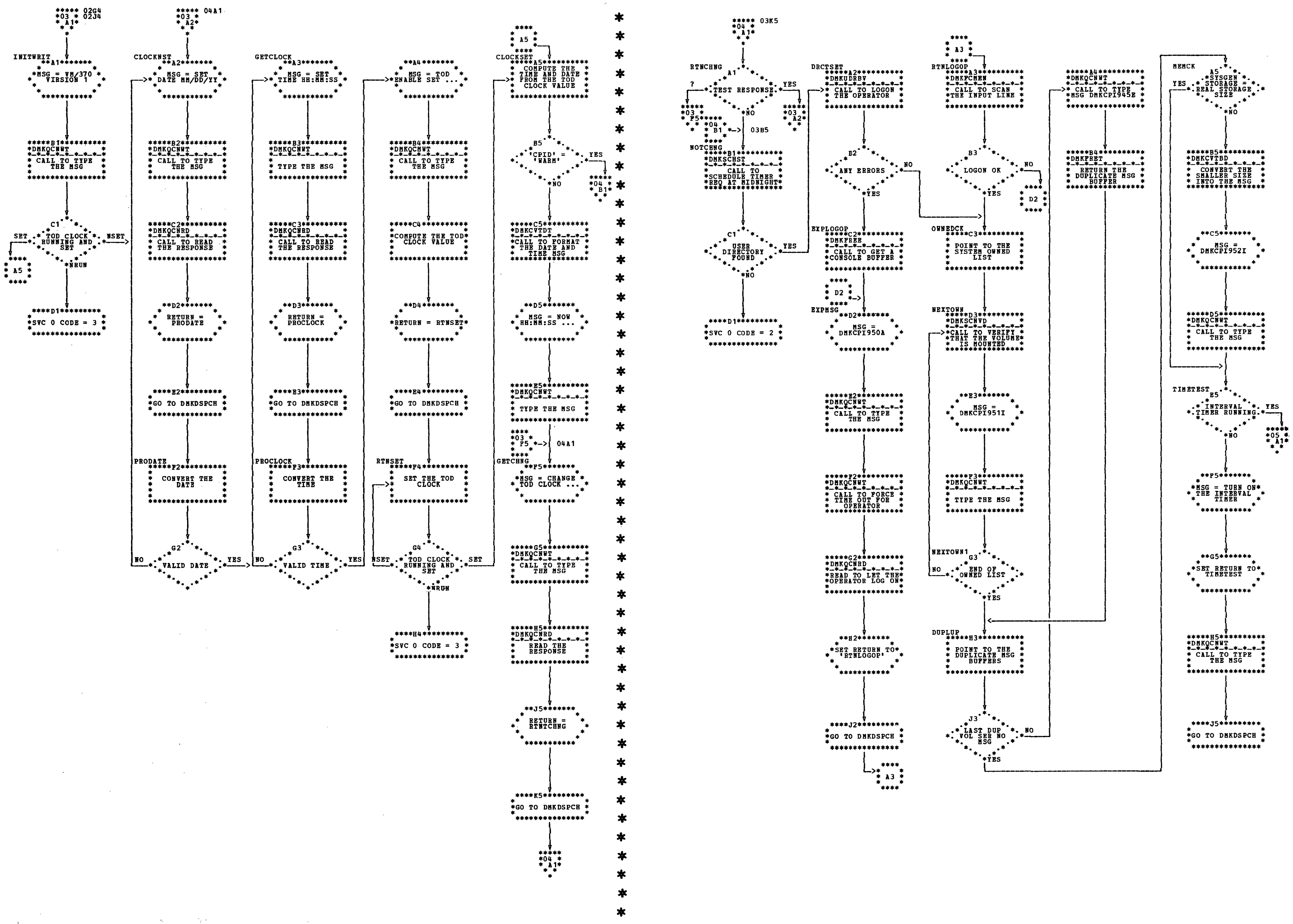

DMKCPBRY
A1 READY COMMAND
B1 FNDVADDR BAL R10 - GET VBLOKS
C1 DEVICE BUSY
YES
01 H1
NO
D1 IS DEVICE VIRTUAL CTCA
NO
YES
E1 IS CTCA COUPLED
NO
YES
CPBRYOK
F1 MAKE DEVICE READY-SET DEVICE END PENDING
G1 RETURN TO DMKCFM
03 E2
04C1
CFP006
E2 INVALID DEV TYPE MSG = DMKCPB006E
01 H5
DMKCPBNR
A3 NOTREADY COMMAND
B3 FNDVADDR BAL R10 - GET VBLOKS
C3 SPOOL UR OR VIRTUAL CONSOLE
NO
YES
D3 SET VDEVNRDY FLAG IN VDEVBLOK
E3 R10 RETURN
DMKCPBRS
A4 RESET COMMAND
B4 FNDVADDR BAL R10 - GET VBLOKS
C4 DMKCPPRD CALL-RESET DEVICE
RESMSG
D4 SET UP 'DEVICE RESET' MESSAGE
01 G1
DMKCPBRW
A1 REWIND COMMAND
B1 FNDVADDR BAL R10 - GET VBLOKS
C1 TAPE DEVICE
NO
03 E2
YES
D1 TAPE ON DEDICATED CHANNEL
NO
CPB059
D2 CP I/O ON DED. CHAN. MSG = DMKCPB059E
01 H5
YES
E1 DMKFREE CALL-GET STORAGE FOR IOBLOK
F1 SET UP IOBLOK FOR SENSE
G1 DEVICE ALREADY BUSY
YES
NO
H1 DMKIOSQR CALL - DO THE SENSE OPERATION
J1 TAPE READY
NO
YES
A3
NOREW
J2 DMKFRET CALL FRET THE IOBLOK
K2 SET UP THE 'REWIND NOT PERFORMED' MESSAGE
01 G1
A3
A3 SET UP IOBLOK FOR REWIND
B3 DMKIOSQR CALL - QUEUE UP THE REWIND
C3 SET UP THE 'REWIND COMPLETE MESSAGE
01 G1
FNDVADDR
A4 GET VBLOK ADDRESSES
B4 DMKSCNFD CALL - LOCATE ARGUMENT
C4 ARGUMENT FOUND
NO
YES
D4 ARGUMENT OVER 3 CHAR.
YES
NO
E4 DMKCVTHB CALL - CONVERT TO BINARY
F4 GOOD CONVERT
NO
CFP022
F5 VADDR MISSING MSG = DMKCPB022E
01 G5
YES
G4 DMKSCNVU CALL - GET VBLOK ADDRESSES
H4 BLOKS FOUND
NO
CFP040
H5 DEV DOES NOT EXIST MSG = DMKCPB040E
01 H5
YES
J4 R9 RETURN

| DMKCPB -- Process EXTERNAL, NOTREADY, READY, REWIND, and SYSTEM Commands (Parts 3 and 4 of 4)

| DMKCPI -- Control Program Initialization (Parts 1 and 2 of 6)

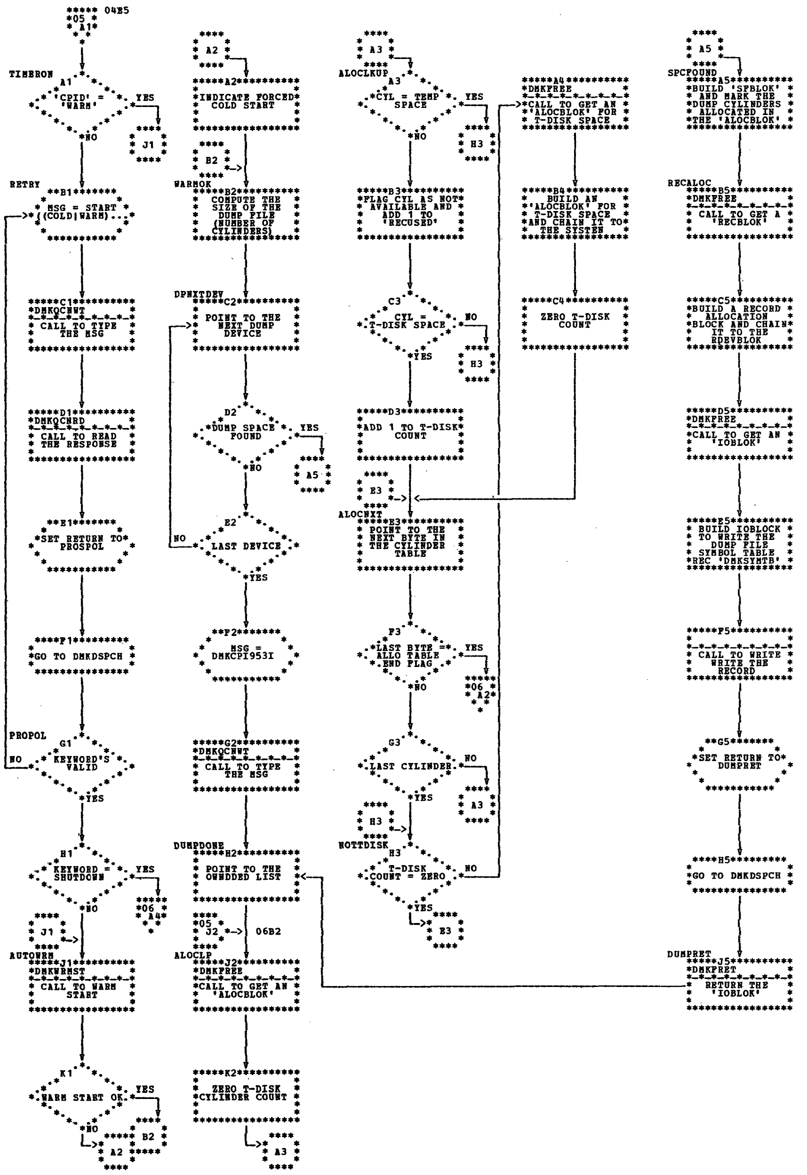

DMKCPINT
A1
DMKCPINT
B1
SET UP NEW PSW'S, SET PSW RESTART TO GO TO 'DMKDMPDK'
C1
CLEAR LOW CORE AND SAVE ADD OF IPL'ED DEVICE
D1
INITIALIZE CONTROL REGISTERS, TOD CLOCK COMP, CPU TIMER
E1
STORE CPU IDENTIFIER
F1
INITIALIZE DISPATCH TIME SLICE BASED ON MODEL NUMBER
G1
SET PROGRAM NEW PSW TO 'CPIPINT'
H1
SET STORAGE KEY TO ZERO
J1
ADD 2K TO THE STORAGE LOCATION
NOTE: ROUTINE WILL PROGRAM CHECK AT THE END OF REAL STORAGE
CPIPINT
A2
PROGRAM CHECK NEW PSW
B2
RESTORE PROGRAM NEW PSW AND SAVE REAL STORAGE SIZE IN 'DMKSYSRM'
C2
POINT TO THE END OF THE CORE TABLE 'CORTABLE'
D2
REAL STORAGE LESS THAN SYSGEN SIZE
NO
YES
E2
MARK PAGES NOT AVAILABLE '*OL*' IN THE 'CORTABLE'
NOOFFLNE
F2
COMPUTE THE NUMBER OF FREE STORAGE PAGES NEEDED
G2
MARK FREE STORAGE PAGES 'FREE' IN THE 'CORTABLE'
H2
FLAG TRACE TABLE (NEXT 2 PAGES) AS BELONG TO CP
J2
SAVE POINTERS TO THE TRACE TABLE IN LOW CORE
K2
ARE THERE 3 FREE PAGES LEFT
NO
YES
B3
A3
SET SWITCH AT LABEL 'CPI955SW' TO A BRANCH
B3
B3
POINT TO 'DMKSAV' (THE LAST PAGE NOT USED)
C3
FLAG PAGES DOWN TO 'DMKCKP' AS FREE, CHAIN TO THE FREE LIST 'DMKPTRF1'
D3
SAVE COUNT OF FREE PAGES
E3
POINT TO 'DMKCPEND' LAST PAGE IN NUCLEUS
F3
FLAG PAGES DOWN TO 'DMKCPEND AS FREE, CHAIN TO THE USER LIST 'DMKPTRU1'
G3
SAVE THE COUNT OF USER PAGES
H3
MARK REMAINING PAGES (CP RESIDENT NUCLEUS) AS '*CP*'
J3
COMPUTE SIZE OF FREE STORAGE
A4
DMKFRETR
RETURN FREE STORAGE TO SYSTEM
B4
DMKFREE
CALL TO GET AN IOBLOCK FOR PAGING
C4
DMKFREE
CALL TO GET AN EXTRA SAVE AREA FOR EXTEND
D4
SET UP POINTERS TO REAL DEVICE, CONTROL UNIT AND CHANNEL BLOCKS
01 E4
02B2
E4
POINT TO THE NEXT REAL DEVICE BLOCK
F4
CC=3
HAULT THE DEVICE
CO-2
G4
DASD DEVICE
NO
02 A2
YES
H4
READ THE VOL1 LABEL
J4
DMKSCNVS
CALL TO LOOK FOR A DUPLICATE VOL SER NO.
K4
DUPLICATE
YES
A5
NO
02 A1
A5
A5
MSG = DMKCPI954E DUPLICATE VOLUME
B5
DMKSCNRD
CALL TO GET THE DEVICE ADDRESS
C5
DMKCVTBH
CALL TO CONVERT THE DEVICE ADDRESS
D5
DMKCVTBH
CALL TO CONVERT ADD OF FIRST VOL MOUNTED
E5
DMKFREE
CALL TO GET FREE STORAGE FOR THE MSG
F5
MOVE MSG INTO FREE STORAGE AND CHAIN IT TO 'DUPMSGPT'

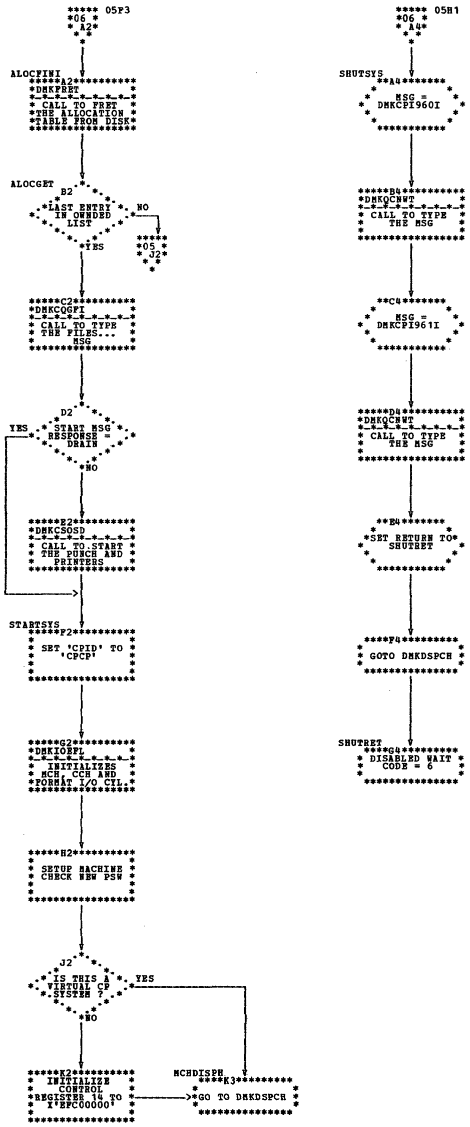


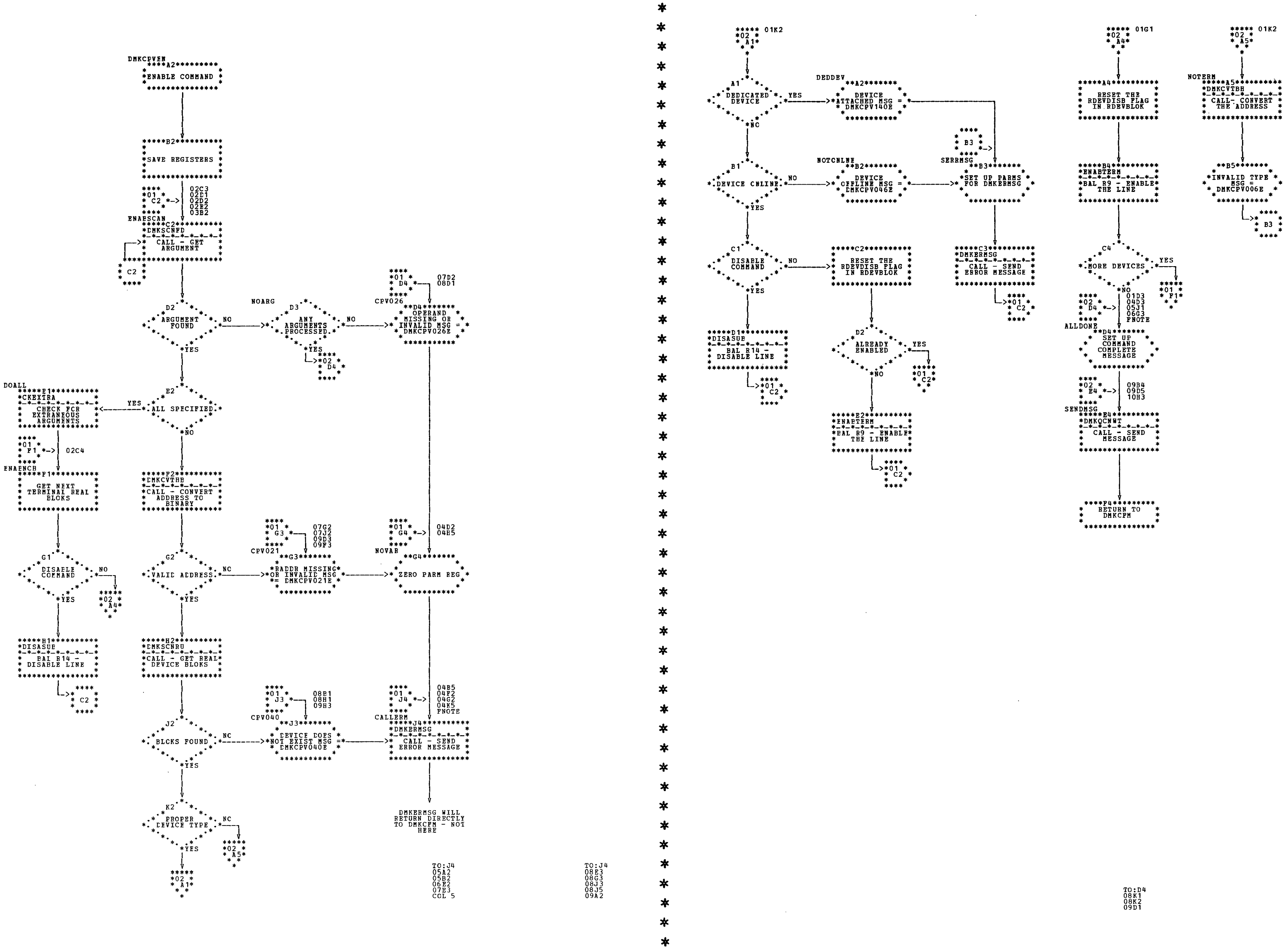

02G4
02J4
03 A1
INITWRIT
A1 MSG = VM/370 VERSION 1
B1 DMKQCNWT CALL TO TYPE THE MSG
C1 TOD CLOCK RUNNING AND SET
SET
NSET
NRUN
A5
D1 SVC 0 CODE = 3
04A1
03 A2
CLOCKNST
A2 MSG = SET DATE MM/DD/YY
B2 DMKQCNWT CALL TO TYPE THE MSG
C2 DMKQCNRD CALL TO READ THE RESPONSE
D2 RETURN = PRODATE
E2 GO TO DMKDSPCH
PRODATE
F2 CONVERT THE DATE
G2 VALID DATE
NO
YES
GETCLOCK
A3 MSG = SET TIME HH:MM:SS
B3 DMKQCNWT TYPE THE MSG
C3 DMKQCNRD CALL TO READ THE RESPONSE
D3 RETURN = PROCLOCK
E3 GO TO DMKDSPCH
PROCLOCK
F3 CONVERT THE TIME
G3 VALID TIME
NO
YES
A4 MSG = TOD ENABLE SET ...
B4 DMKQCNWT CALL TO TYPE THE MSG
C4 COMPUTE THE TOD CLOCK VALUE
D4 RETURN = RTNSET
E4 GO TO DMKDSPCH
RTNSET
F4 SET THE TOD CLOCK
G4 TOD CLOCK RUNNING AND SET
NSET
SET
NRUN
H4 SVC 0 CODE = 3
A5
CLOCKSET
A5 COMPUTE THE TIME AND DATE FROM THE TOD CLOCK VALUE
B5 'CPID' = 'WARM'
YES
NO
04 B1
C5 DMKCVTDT CALL TO FORMAT THE DATE AND TIME MSG
D5 MSG = NOW HH:MM:SS ...
E5 DMKQCNWT TYPE THE MSG
03 F5
04A1
GETCHNG
F5 MSG = CHANGE TOD CLOCK ...
G5 DMKQCNWT CALL TO TYPE THE MSG
H5 DMKQCNRD READ THE RESPONSE
J5 RETURN = RTNTCHNG
K5 GO TO DMKDSPCH
04 A1
03K5
04 A1
RTNCHNG
A1 TEST RESPONSE
?
YES
NO
03 F5
04 B1
03B5
03 A2
NOTCHNG
B1 DMKSCHST CALL TO SCHEDULE TIMER REQ AT MIDNIGHT
C1 USER DIRECTORY FOUND
YES
NO
D1 SVC 0 CODE = 2
DRCTSET
A2 DMKUDRBV CALL TO LOGON THE OPERATOR
B2 ANY ERRORS
NO
YES
EXPLOGOP
C2 DMKFREE CALL TO GET A CONSOLE BUFFER
D2
EXPMSG
D2 MSG = DMKCPI950A
E2 DMKQCNWT CALL TO TYPE THE MSG
F2 DMKQCNWT CALL TO FORCE TIME OUT FOR OPERATOR
G2 DMKQCNRD READ TO LET THE OPERATOR LOG ON
H2 SET RETURN TO 'RTNLOGOP'
J2 GO TO DMKDSPCH
A3
A3
RTNLOGOP
A3 DMKFCMEN CALL TO SCAN THE INPUT LINE
B3 LOGON OK
NO
YES
D2
OWNEDCK
C3 POINT TO THE SYSTEM OWNED LIST
NEXTOWN
D3 DMKSCNVD CALL TO VERIFY THAT THE VOLUME IS MOUNTED
E3 MSG = DMKCPI951I
F3 DMKQCNWT TYPE THE MSG
NEXTOWN1
G3 END OF OWNED LIST
NO
YES
DUPLUP
H3 POINT TO THE DUPLICATE MSG BUFFERS
J3 LAST DUP VOL SER NO MSG
NO
YES
A4 DMKQCNWT CALL TO TYPE MSG DMKCPI945E
B4 DMKFRET RETURN THE DUPLICATE MSG BUFFER
MEMCK
A5 SYSGEN STORAGE = REAL STORAGE SIZE
YES
NO
B5 DMKCVTBD CONVERT THE SMALLER SIZE INTO THE MSG
C5 MSG = DMKCPI952I
D5 DMKQCNWT CALL TO TYPE THE MSG
TIMETEST
E5 INTERVAL TIMER RUNNING
YES
NO
05 A1
F5 MSG = TURN ON THE INTERVAL TIMER
G5 SET RETURN TO TIMETEST
H5 DMKQCNWT CALL TO TYPE THE MSG
J5 GO TO DMKDSPCH

| DMKCPI -- Control Program Initialization (Parts 3 and 4 of 6)

| DMKCPI -- Control Program Initialization (Parts 5 and 6 of 6)

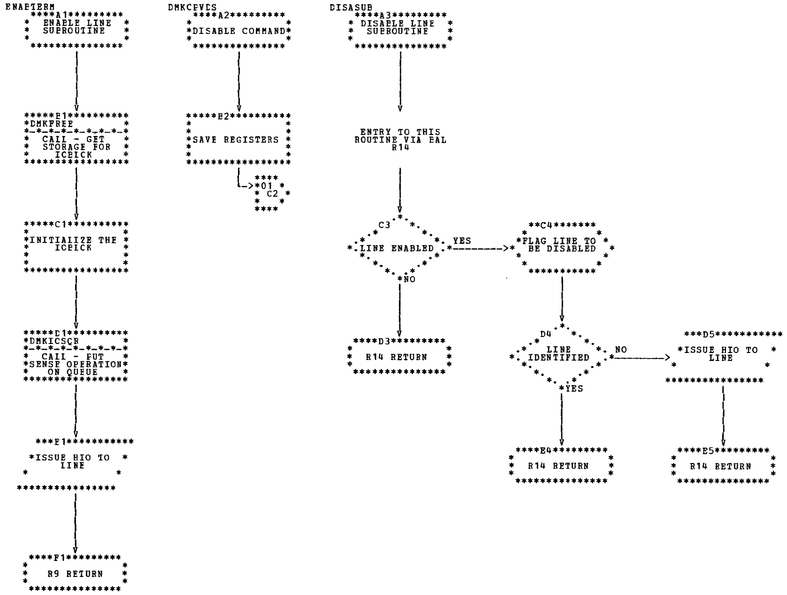


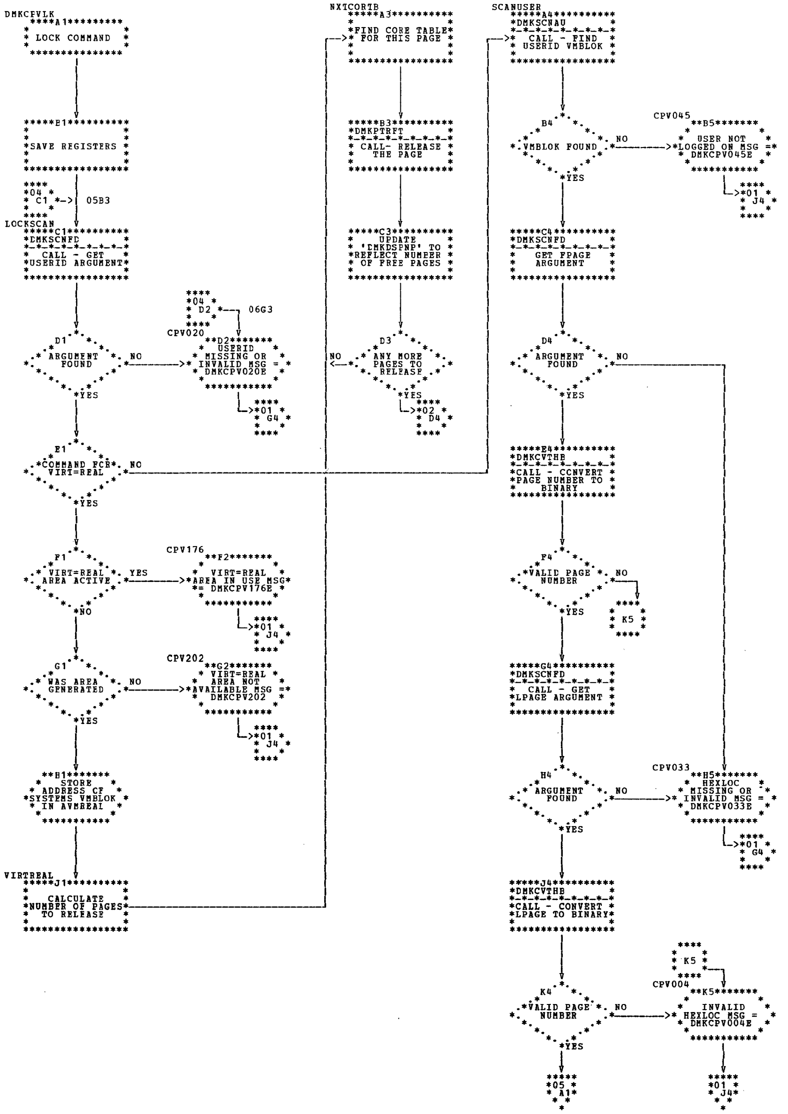


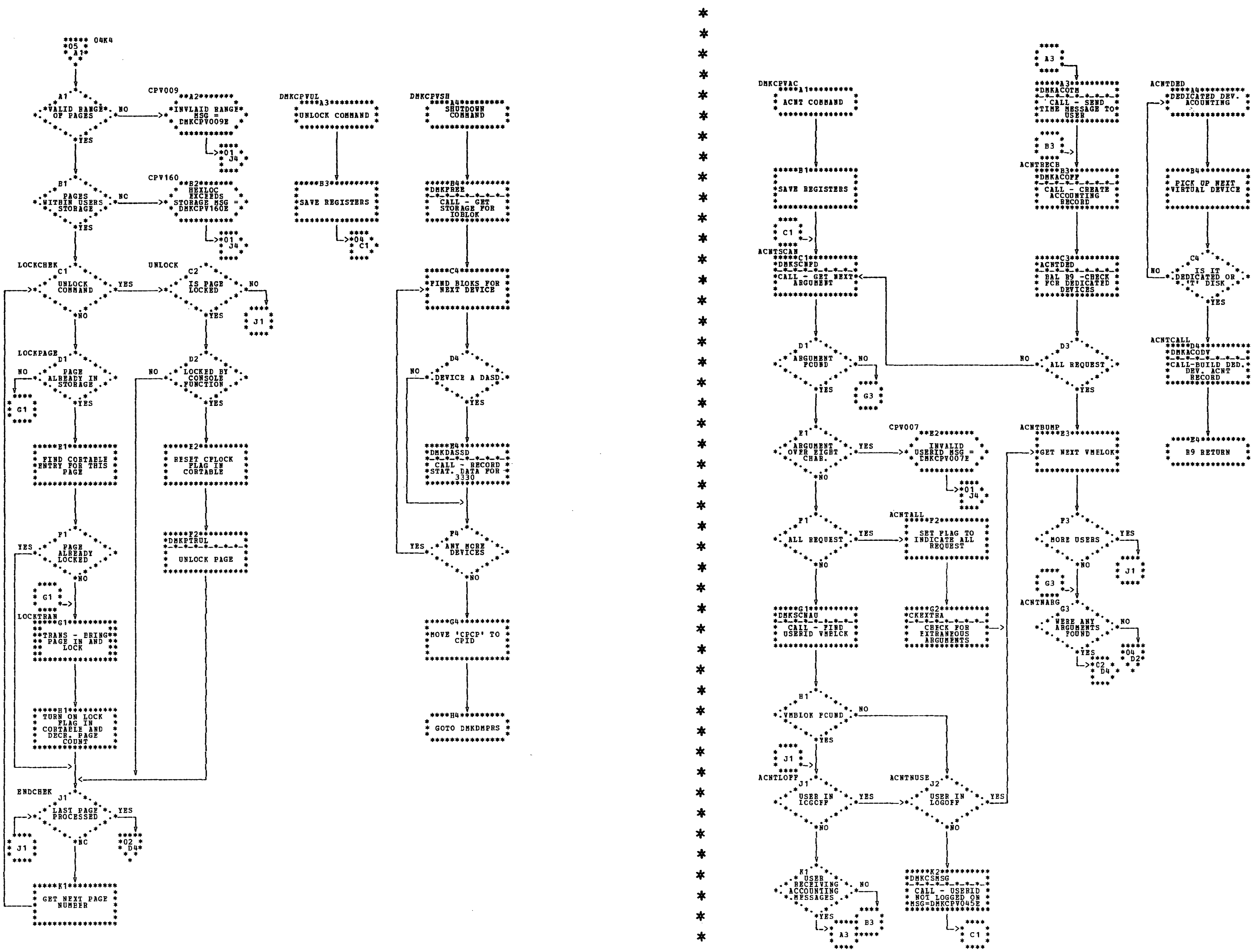


| DMKCPV -- Process DISAELE, ENAELE, HALT, SHUTDOWN, UNLOCK, and VARY Commands (Parts 1 and 2 of 10)

| DMKCPV -- Process DISABLE, ENABLE, HALT, SHUTDOWN, UNLOCK, and VARY Commands (Parts 3 and 4 of 10)

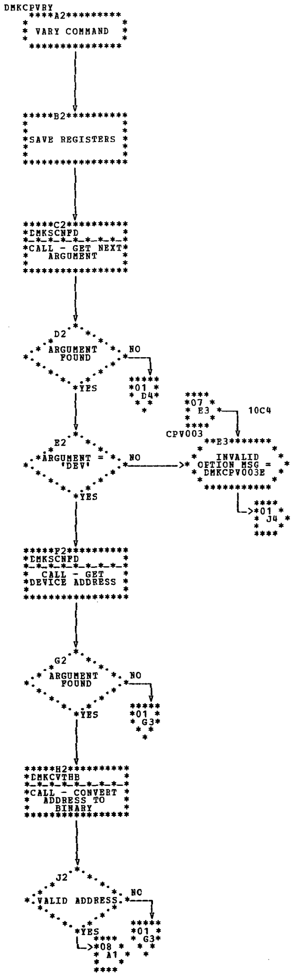


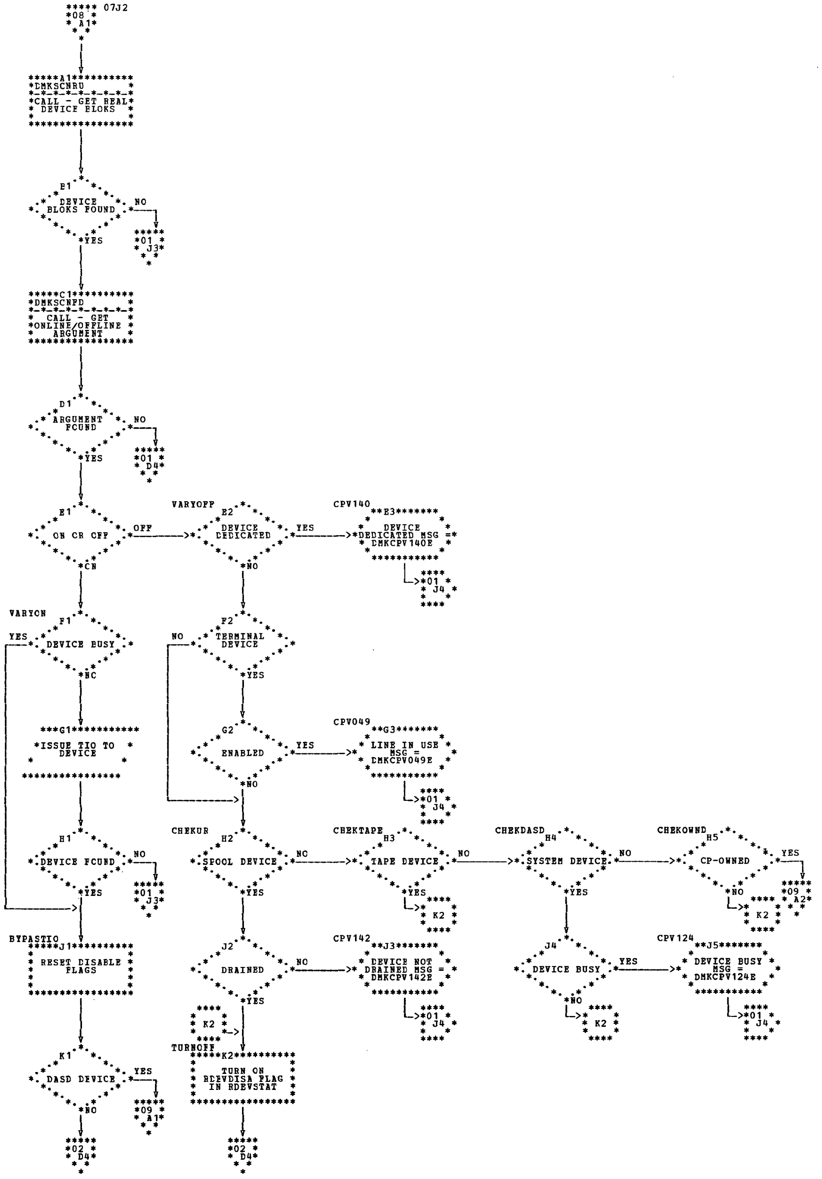

DMKCPVLK
A1
LOCK COMMAND
B1
SAVE REGISTERS
04 C1
05B3
LOCKSCAN
C1
DMKSCNFD
CALL - GET USERID ARGUMENT
04 D2
06G3
D1
ARGUMENT FOUND
NO
YES
CPV020
D2
USERID MISSING OR INVALID MSG = DMKCPV020E
01 G4
E1
COMMAND FOR VIRT=REAL
NO
YES
F1
VIRT=REAL AREA ACTIVE
YES
NO
CPV176
F2
VIRT=REAL AREA IN USE MSG = DMKCPV176E
01 J4
G1
WAS AREA GENERATED
NO
YES
CPV202
G2
VIRT=REAL AREA NOT AVAILABLE MSG = DMKCPV202
01 J4
H1
STORE ADDRESS OF SYSTEMS VMBLOK IN AVMREAL
VIRTREAL
J1
CALCULATE NUMBER OF PAGES TO RELEASE
NXTCORTB
A3
FIND CORE TABLE FOR THIS PAGE
B3
DMKPTRFT
CALL- RELEASE THE PAGE
C3
UPDATE 'DMKDSPNP' TO REFLECT NUMBER OF FREE PAGES
D3
ANY MORE PAGES TO RELEASE
NO
YES
02 D4
SCANUSER
A4
DMKSCNAU
CALL - FIND USERID VMBLOK
B4
VMBLOK FOUND
NO
YES
CPV045
B5
USER NOT LOGGED ON MSG = DMKCPV045E
01 J4
C4
DMKSCNFD
GET FPAGE ARGUMENT
D4
ARGUMENT FOUND
NO
YES
E4
DMKCVTHB
CALL - CONVERT PAGE NUMBER TO BINARY
F4
VALID PAGE NUMBER
NO
YES
K5
G4
DMKSCNFD
CALL - GET LPAGE ARGUMENT
H4
ARGUMENT FOUND
NO
YES
CPV033
H5
HEXLOC MISSING OR INVALID MSG = DMKCPV033E
01 G4
J4
DMKCVTHB
CALL - CONVERT LPAGE TO BINARY
K4
VALID PAGE NUMBER
NO
YES
K5
CPV004
K5
INVALID HEXLOC MSG = DMKCPV004E
05 A1
01 J4

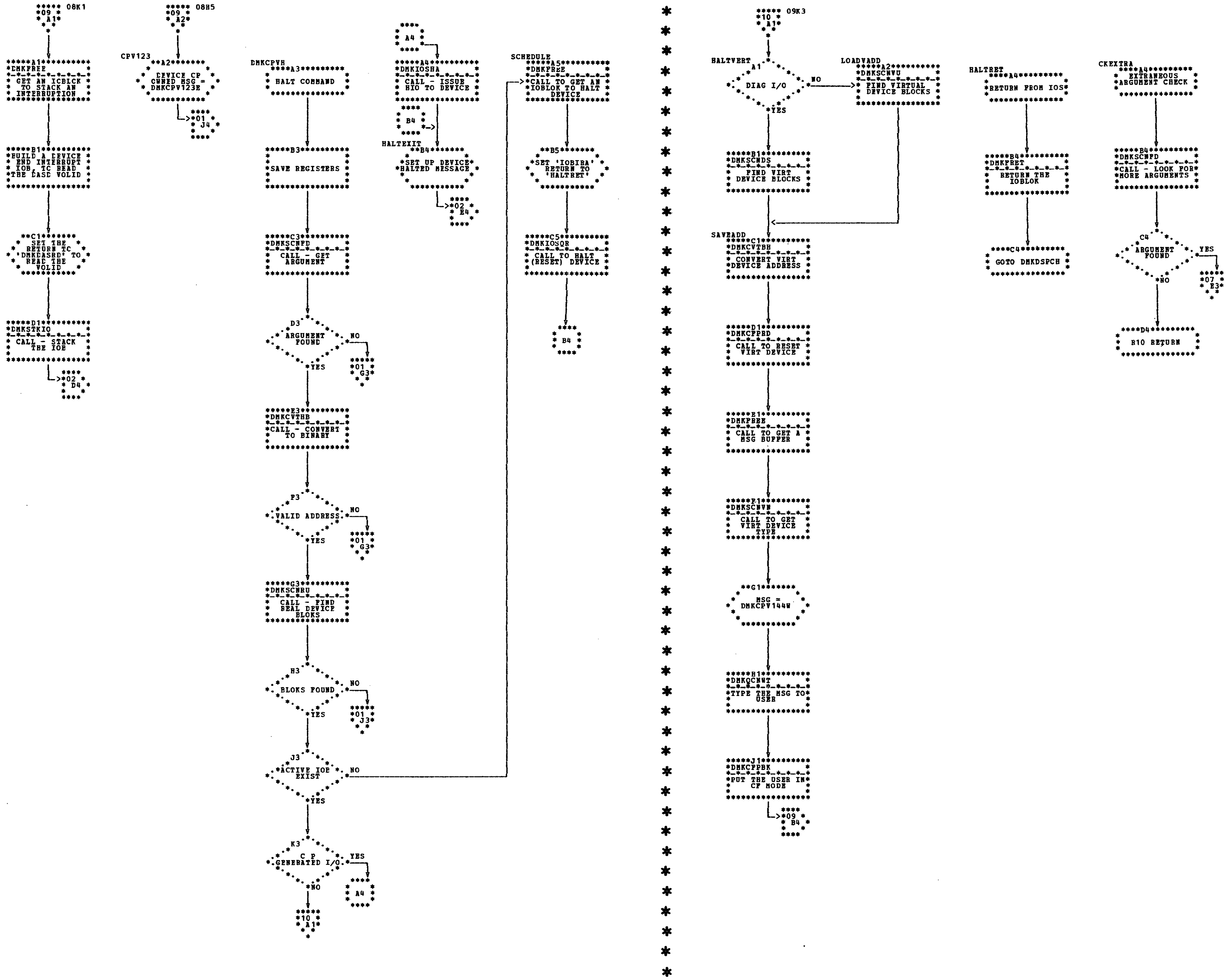

04K4
05 A1
A1 VALID RANGE OF PAGES
NO
YES
CPV009
A2 INVLAID RANGE MSG = DMKCPV009E
01 J4
B1 PAGES WITHIN USERS STORAGE
NC
CPV160
B2 HEXLOC EXCEEDS STORAGE MSG = DMKCPV160E
01 J4
LOCKCHEK
C1 UNLOCK COMMAND
YES
NO
UNLOCK
C2 IS PAGE LOCKED
J1
LOCKPAGE
D1 PAGE ALREADY IN STORAGE
G1
D2 LOCKED BY CONSOLE FUNCTION
E1 FIND CORTABLE ENTRY FOR THIS PAGE
E2 RESET CFLOCK FLAG IN CORTABLE
F1 PAGE ALREADY LOCKED
F2 DMKPTRUL UNLOCK PAGE
G1
LOCKTRAN
G1 TRANS - BRING PAGE IN AND LOCK
H1 TURN ON LOCK FLAG IN CORTABLE AND DECR. PAGE COUNT
ENDCHEK
J1 LAST PAGE PROCESSED
J1
NC
02 D4
K1 GET NEXT PAGE NUMBER
DMKCPVUL
A3 UNLOCK COMMAND
B3 SAVE REGISTERS
04 C1
DMKCPVSH
A4 SHUTDOWN COMMAND
B4 DMKFREE CALL - GET STORAGE FOR IOBLOK
C4 FIND BLOKS FOR NEXT DEVICE
D4 DEVICE A DASD
E4 DMKDASSD CALL - RECORD STAT. DATA FOR 3330
F4 ANY MORE DEVICES
G4 MOVE 'CPCP' TO CPID
H4 GOTO DMKDMPRS
DMKCPVAC
A1 ACNT COMMAND
B1 SAVE REGISTERS
C1
ACNTSCAN
C1 DMKSCNFD CALL - GET NEXT ARGUMENT
D1 ARGUMENT FOUND
G3
E1 ARGUMENT OVER EIGHT CHAR.
CPV007
E2 INVALID USERID MSG = DMKCPV007E
01 J4
F1 ALL REQUEST
ACNTALL
F2 SET FLAG TO INDICATE ALL REQUEST
G1 DMKSCNAU CALL - FIND USERID VMBLCK
G2 CKEXTRA CHECK FOR EXTRANEOUS ARGUMENTS
H1 VMBLOK FOUND
J1
ACNTLOFF
J1 USER IN LOGOFF
ACNTNUSE
J2 USER IN LOGOFF
K1 USER RECEIVING ACCOUNTING MESSAGES
B3
A3
K2 DMKCSMSG CALL - USERID NOT LOGGED ON MSG=DMKCPV045E
C1
A3
A3 DMKACOTM CALL - SEND TIME MESSAGE TO USER
B3
ACNTRECB
B3 DMKACOFF CALL - CREATE ACCOUNTING RECORD
C3 ACNTDED BAL R9 -CHECK FOR DEDICATED DEVICES
D3 ALL REQUEST
ACNTBUMP
E3 GET NEXT VMBLOK
F3 MORE USERS
J1
G3
ACNTNARG
G3 WERE ANY ARGUMENTS FOUND
04 D2
C2 D4
ACNTDED
A4 DEDICATED DEV. ACOUNTING
B4 PICK UP NEXT VIRTUAL DEVICE
C4 IS IT DEDICATED OR 'T' DISK
ACNTCALL
D4 DMKACODV CALL-BUILD DED. DEV. ACNT RECORD
E4 R9 RETURN

| DMKCPV -- Process DISABLE, ENABLE, HALT, SHUTDOWN, UNLCCK, and VARY Commands (Parts 5 and 6 of 10)

| DMKCPV -- Process DISABLE, ENABLE, HALT, SHUTDOWN, UNLOCK, and VARY Commands (Parts 7 and 8 of 10)

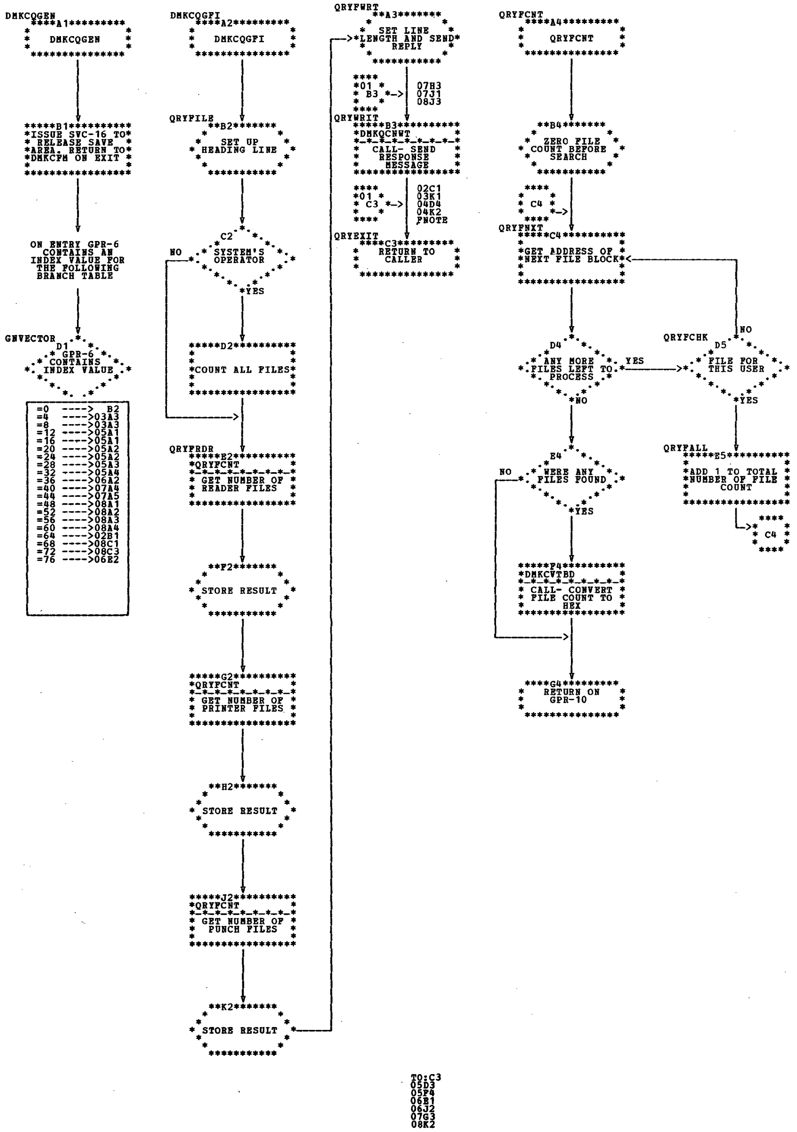


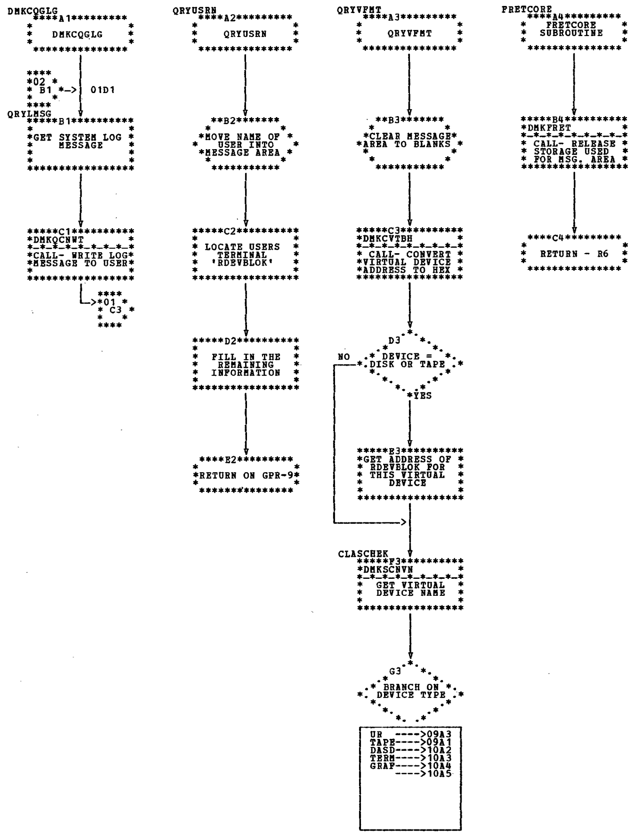


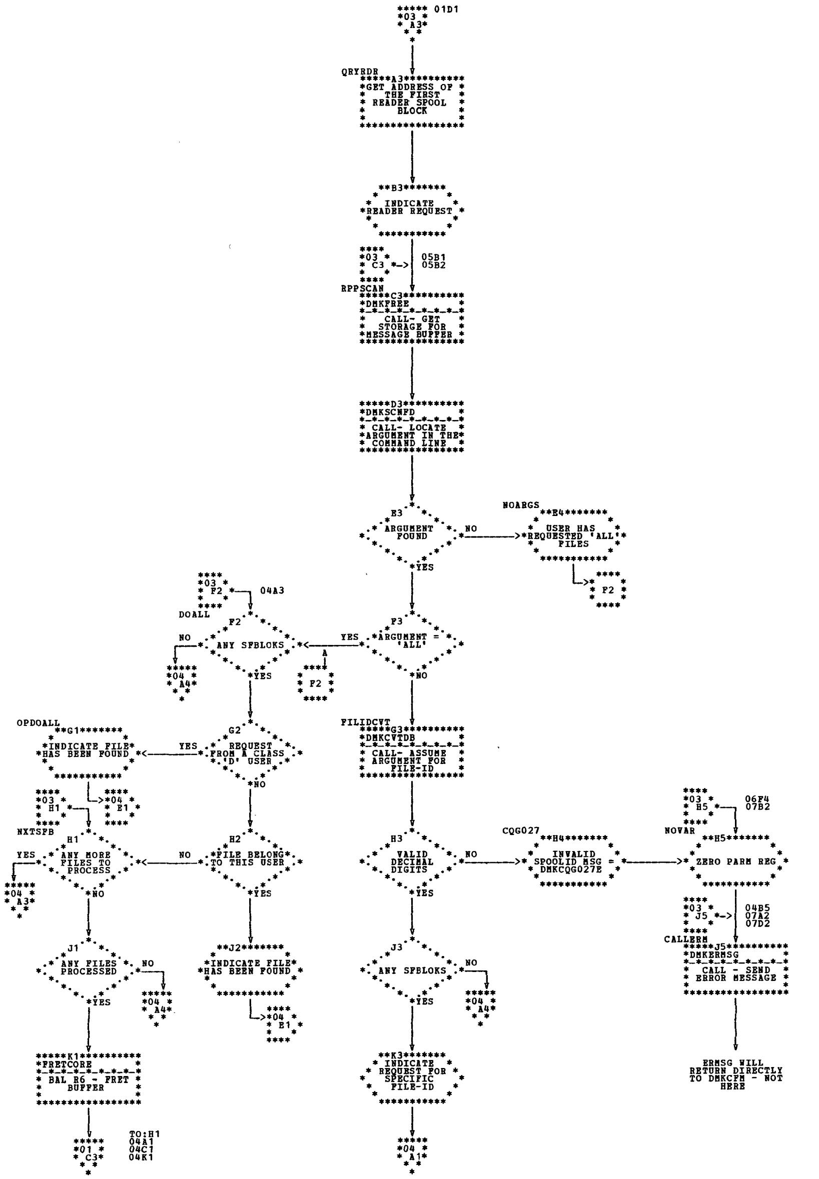


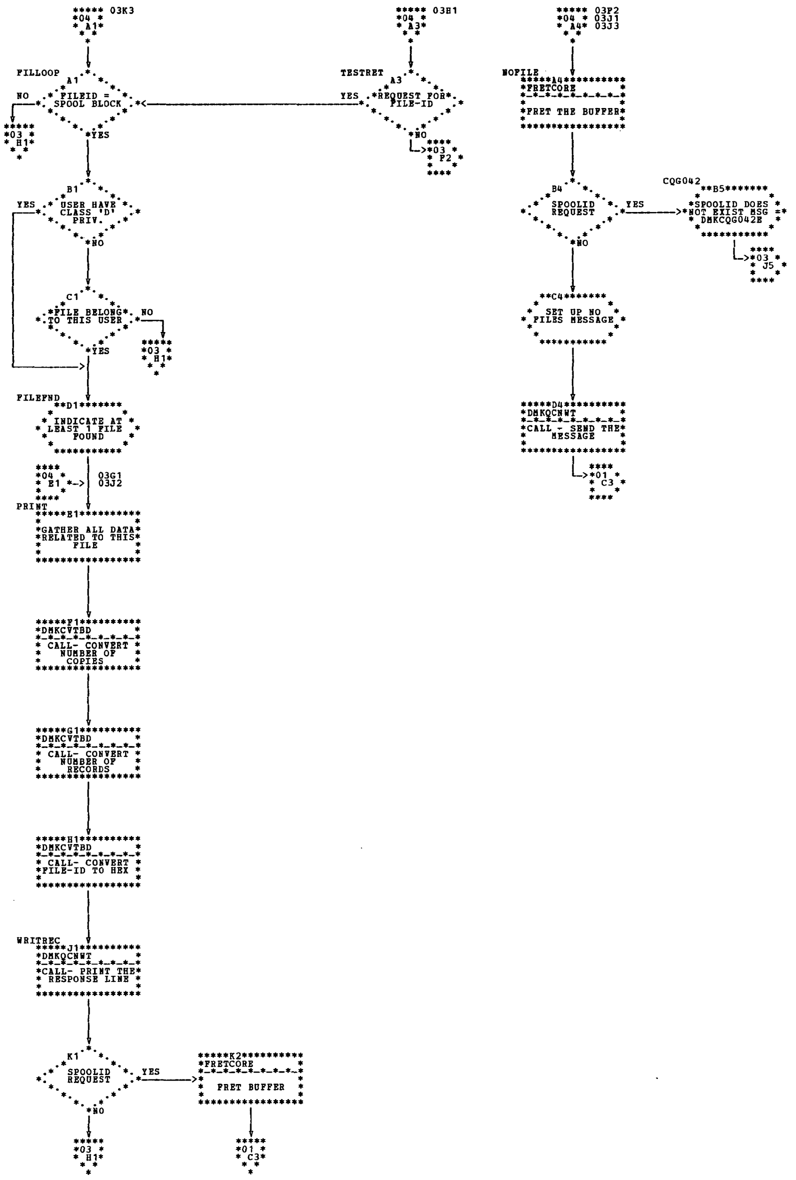
| DMKCPV -- Process DISABLE, ENABLE, HALT, SHUTDOWN, UNLOCK, and VARY Commands (Parts 9 and 10 of 10)

DMKCQG -- Process QUERY Command (Parts 1 and 2 of 10)



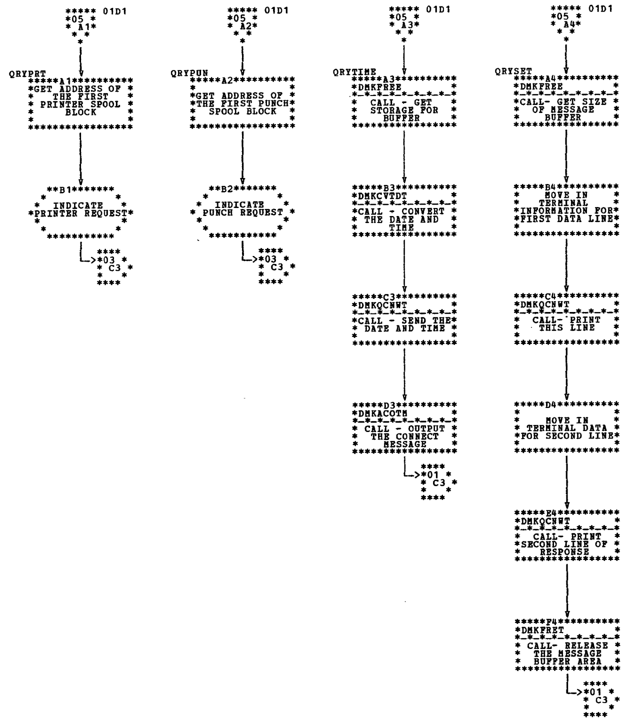


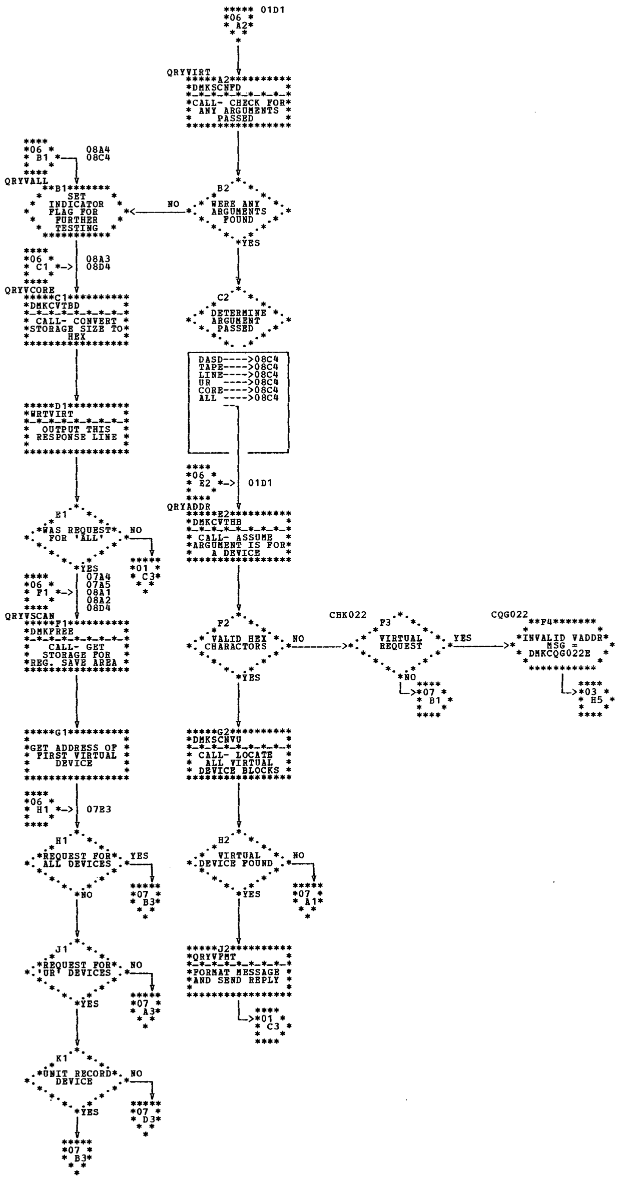

03 A3 01D1
QRYRDR
A3 GET ADDRESS OF THE FIRST READER SPOOL BLOCK
B3 INDICATE READER REQUEST
03 C3 05B1 05B2
RPPSCAN
C3 DMKFREE CALL- GET STORAGE FOR MESSAGE BUFFER
D3 DMKSCNFD CALL- LOCATE ARGUMENT IN THE COMMAND LINE
E3 ARGUMENT FOUND
NO
YES
NOARGS
E4 USER HAS REQUESTED 'ALL' FILES
F2
03 F2 04A3
DOALL
F2 ANY SFBLOKS
NO
YES
04 A4
A
F2
F3 ARGUMENT = 'ALL'
YES
NO
OPDOALL
G1 INDICATE FILE HAS BEEN FOUND
G2 REQUEST FROM A CLASS 'D' USER
YES
NO
04 E1
03 H1
NXTSFB
H1 ANY MORE FILES TO PROCESS
YES
NO
04 A3
H2 FILE BELONG TO THIS USER
NO
YES
J1 ANY FILES PROCESSED
NO
YES
04 A4
J2 INDICATE FILE HAS BEEN FOUND
04 E1
K1 FRETCORE BAL R6 - FRET BUFFER
01 C3
TO:H1 04A1 04C1 04K1
FILIDCVT
G3 DMKCVTDB CALL- ASSUME ARGUMENT FOR FILE-ID
H3 VALID DECIMAL DIGITS
NO
YES
CQG027
H4 INVALID SPOOLID MSG = DMKCQG027E
03 H5 06F4 07B2
NOVAR
H5 ZERO PARM REG
03 J5 04B5 07A2 07D2
CALLERM
J5 DMKERMSG CALL - SEND ERROR MESSAGE
ERMSG WILL RETURN DIRECTLY TO DMKCFM - NOT HERE
J3 ANY SFBLOKS
NO
YES
04 A4
K3 INDICATE REQUEST FOR SPECIFIC FILE-ID
04 A1

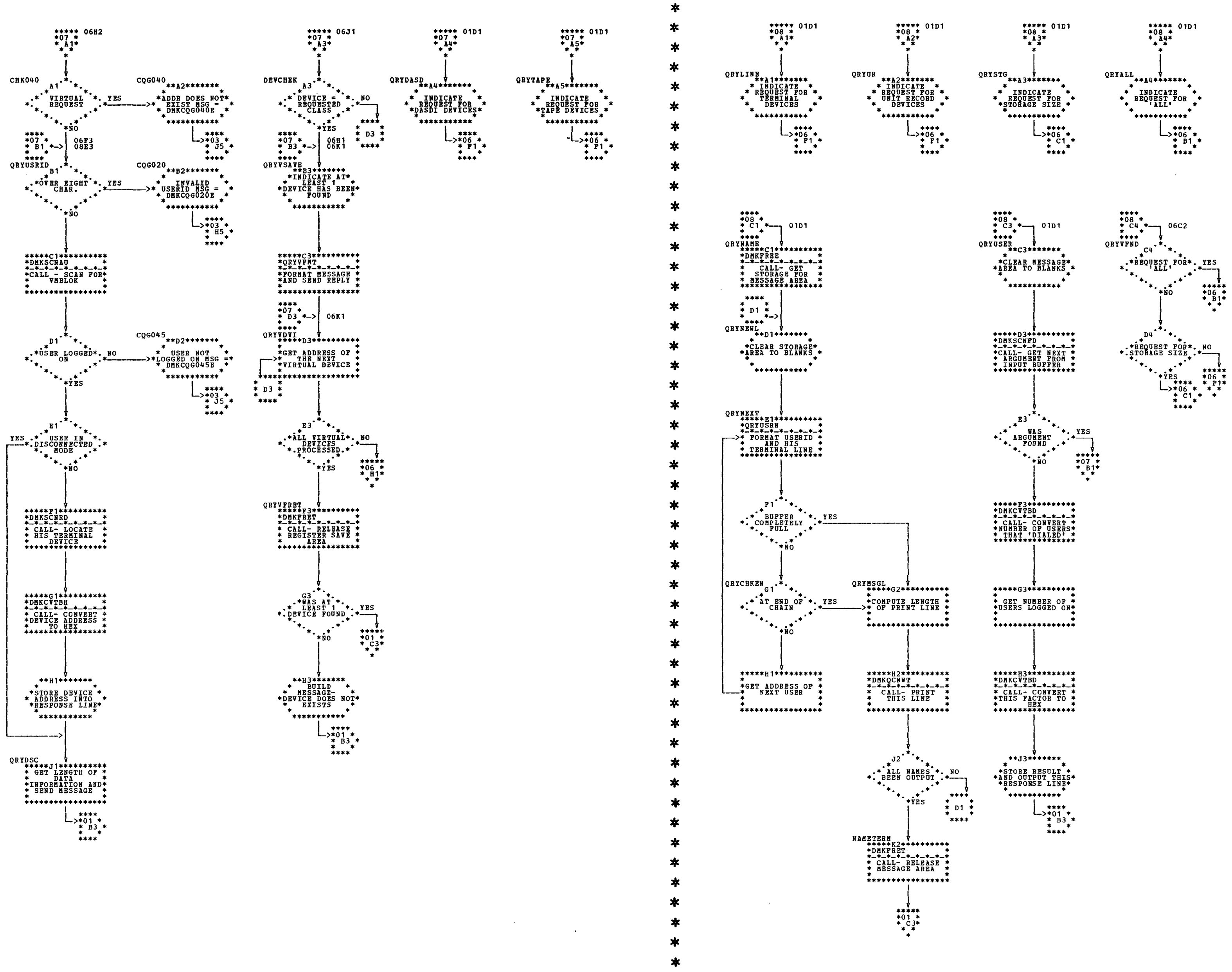

04 A1 03K3
FILLOOP
A1 FILEID = SPOOL BLOCK
NO
YES
03 H1
04 A3 03H1
TESTRET
A3 REQUEST FOR FILE-ID
YES
NO
03 F2
B1 USER HAVE CLASS 'D' PRIV.
YES
NO
C1 FILE BELONG TO THIS USER
NO
YES
03 H1
FILEFND
D1 INDICATE AT LEAST 1 FILE FOUND
04 E1 03G1 03J2
PRINT
E1 GATHER ALL DATA RELATED TO THIS FILE
F1 DMKCVTBD CALL- CONVERT NUMBER OF COPIES
G1 DMKCVTBD CALL- CONVERT NUMBER OF RECORDS
H1 DMKCVTBD CALL- CONVERT FILE-ID TO HEX
WRITREC
J1 DMKQCNWT CALL- PRINT THE RESPONSE LINE
K1 SPOOLID REQUEST
YES
NO
03 H1
K2 FRETCORE FRET BUFFER
01 C3
04 A4 03F2 03J1 03J3
NOFILE
A4 FRETCORE FRET THE BUFFER
B4 SPOOLID REQUEST
YES
NO
CQG042
B5 SPOOLID DOES NOT EXIST MSG = DMKCQG042E
03 J5
C4 SET UP NO FILES MESSAGE
D4 DMKQCNWT CALL - SEND THE MESSAGE
01 C3

DMKCQG -- Process QUERY Command (Parts 3 and 4 of 10)

DMKCQG -- Process QUERY Command (Parts 5 and 6 of 10)

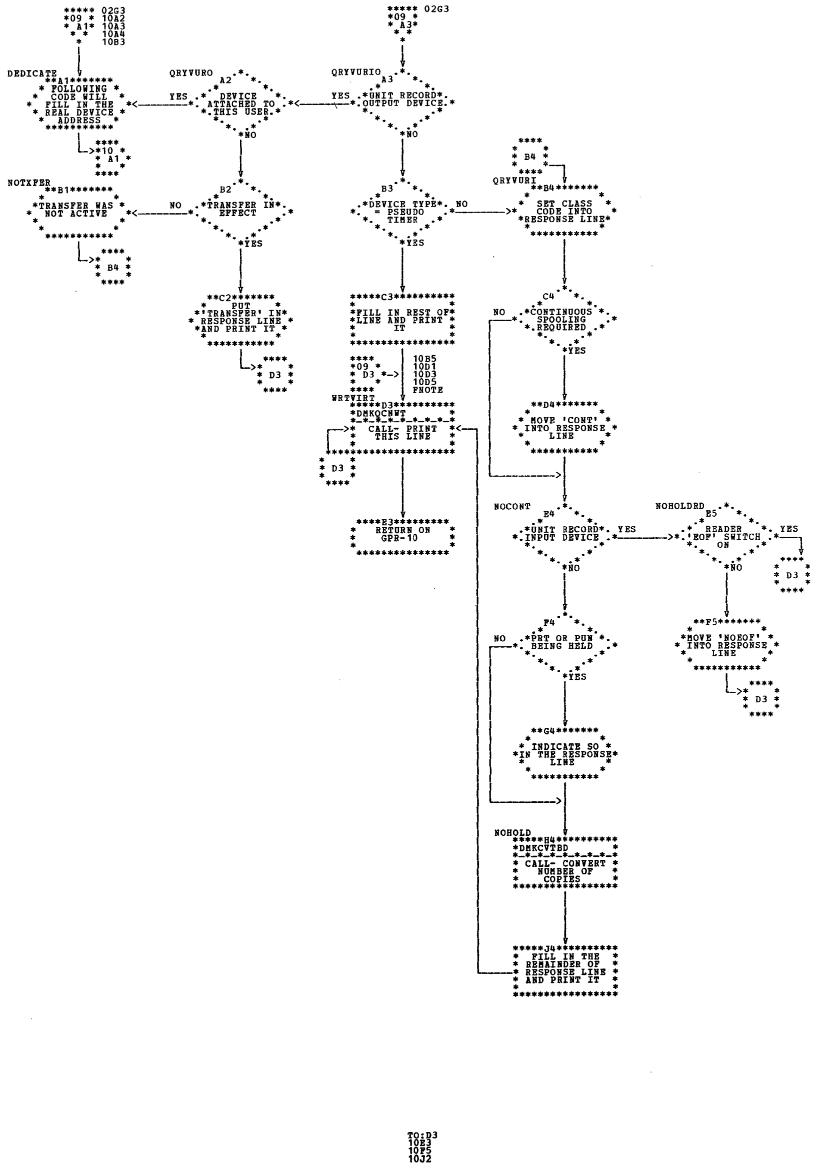

01D1
05 A1
QRYPRT
A1
GET ADDRESS OF THE FIRST PRINTER SPOOL BLOCK
B1
INDICATE PRINTER REQUEST
03 C3
01D1
05 A2
QRYPUN
A2
GET ADDRESS OF THE FIRST PUNCH SPOOL BLOCK
B2
INDICATE PUNCH REQUEST
03 C3
01D1
05 A3
QRYTIME
A3
DMKFREE
CALL - GET STORAGE FOR BUFFER
B3
DMKCVTDT
CALL - CONVERT THE DATE AND TIME
C3
DMKQCNWT
CALL - SEND THE DATE AND TIME
D3
DMKACOTM
CALL - OUTPUT THE CONNECT MESSAGE
01 C3
01D1
05 A4
QRYSET
A4
DMKFREE
CALL- GET SIZE OF MESSAGE BUFFER
B4
MOVE IN TERMINAL INFORMATION FOR FIRST DATA LINE
C4
DMKQCNWT
CALL- PRINT THIS LINE
D4
MOVE IN TERMINAL DATA FOR SECOND LINE
E4
DMKQCNWT
CALL- PRINT SECOND LINE OF RESPONSE
F4
DMKFRET
CALL- RELEASE THE MESSAGE BUFFER AREA
01 C3

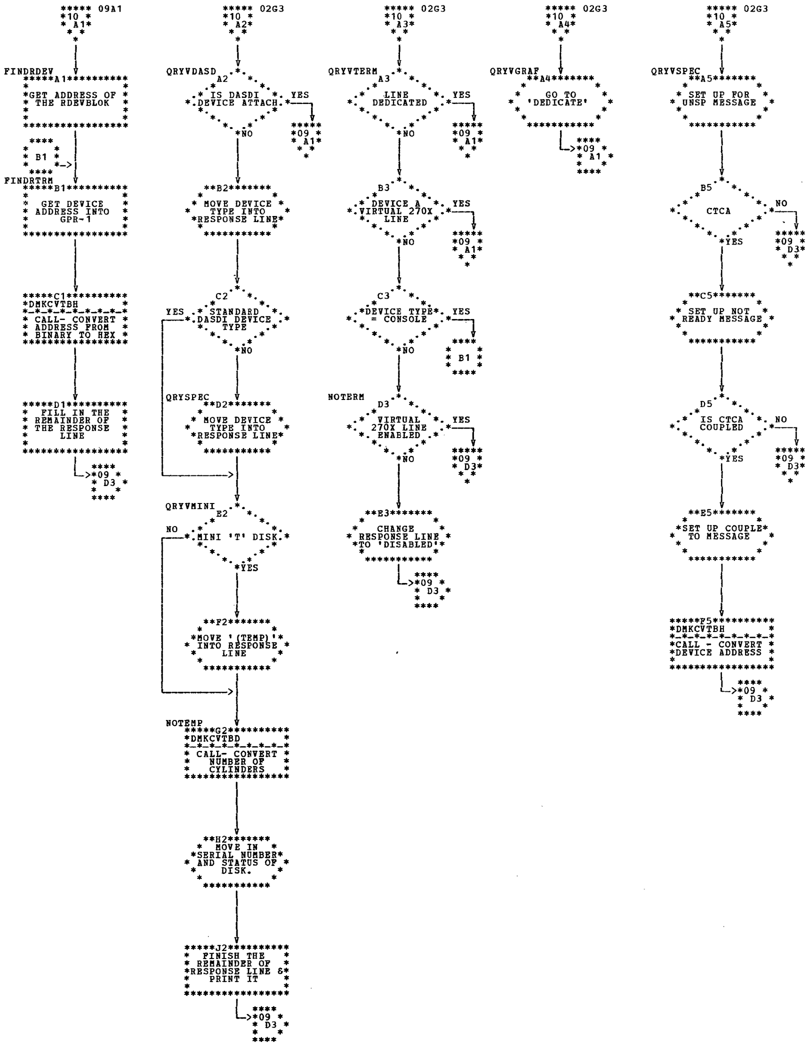


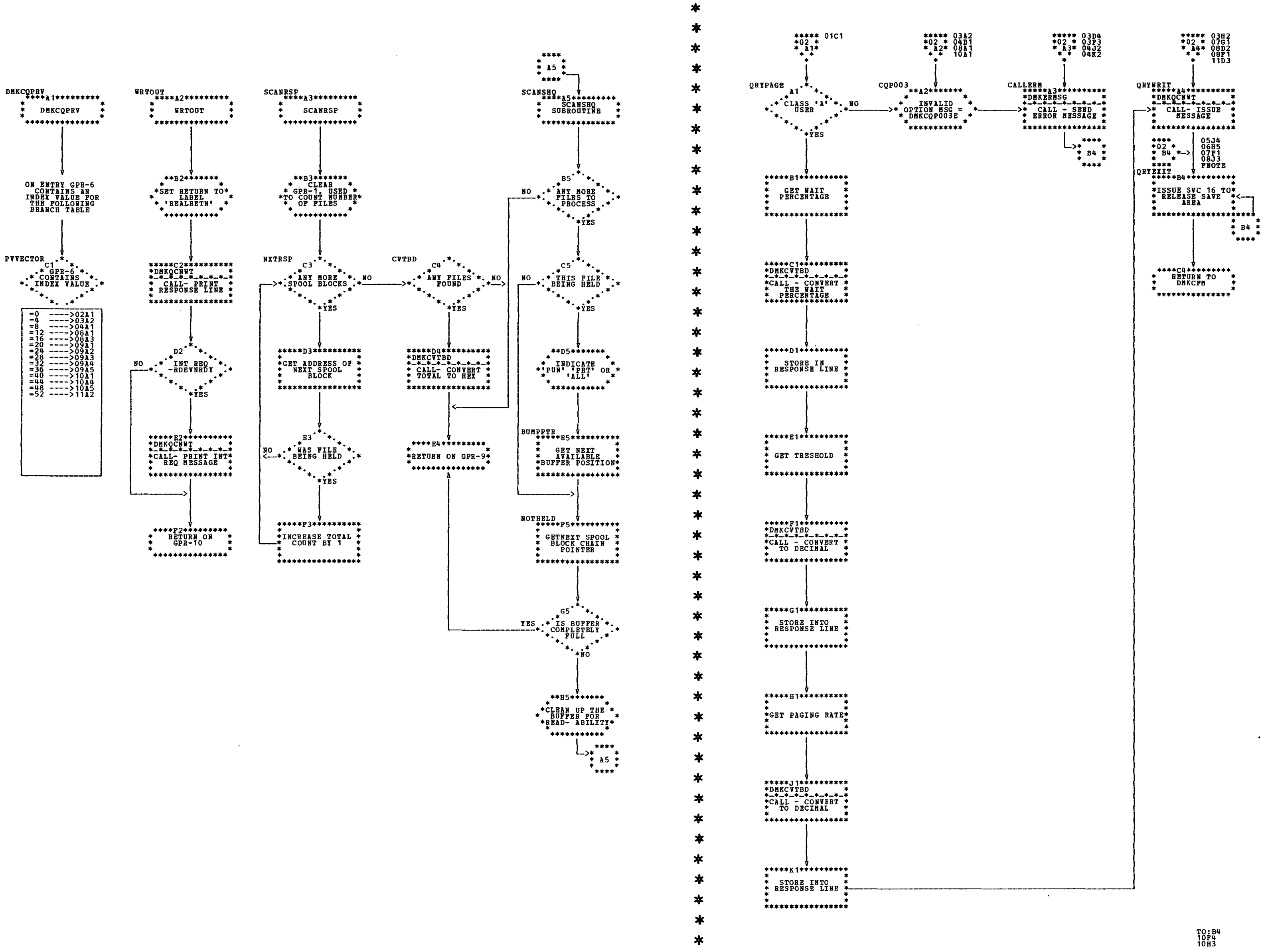

06H2
07 A1
CHK040
A1 VIRTUAL REQUEST
YES
NO
CQG040
A2 ADDR DOES NOT EXIST MSG = DMKCQG040E
03 J5
07 B1
06F3
08E3
QRYUSRID
B1 OVER EIGHT CHAR.
CQG020
B2 INVALID USERID MSG = DMKCQG020E
03 H5
C1 DMKSCNAU CALL - SCAN FOR VMBLOK
D1 USER LOGGED ON
CQG045
D2 USER NOT LOGGED ON MSG = DMKCQG045E
03 J5
E1 USER IN DISCONNECTED MODE
F1 DMKSCNRD CALL- LOCATE HIS TERMINAL DEVICE
G1 DMKCVTBH CALL- CONVERT DEVICE ADDRESS TO HEX
H1 STORE DEVICE ADDRESS INTO RESPONSE LINE
QRYDSC
J1 GET LENGTH OF DATA INFORMATION AND SEND MESSAGE
01 B3
06J1
07 A3
DEVCHEK
A3 DEVICE = REQUESTED CLASS
D3
07 B3
06H1
06K1
QRYVSAVE
B3 INDICATE AT LEAST 1 DEVICE HAS BEEN FOUND
C3 QRYVFMT FORMAT MESSAGE AND SEND REPLY
07 D3
06K1
QRYVDVI
D3 GET ADDRESS OF THE NEXT VIRTUAL DEVICE
D3
E3 ALL VIRTUAL DEVICES PROCESSED
06 H1
QRYVFRET
F3 DMKFRET CALL- RELEASE REGISTER SAVE AREA
G3 WAS AT LEAST 1 DEVICE FOUND
01 C3
H3 BUILD MESSAGE- DEVICE DOES NOT EXISTS
01 B3
01D1
07 A4
QRYDASD
A4 INDICATE REQUEST FOR DASDI DEVICES
06 F1
01D1
07 A5
QRYTAPE
A5 INDICATE REQUEST FOR TAPE DEVICES
06 F1
01D1
08 A1
QRYLINE
A1 INDICATE REQUEST FOR TERMINAL DEVICES
06 F1
01D1
08 A2
QRYUR
A2 INDICATE REQUEST FOR UNIT RECORD DEVICES
06 F1
01D1
08 A3
QRYSTG
A3 INDICATE REQUEST FOR STORAGE SIZE
06 C1
01D1
08 A4
QRYALL
A4 INDICATE REQUEST FOR 'ALL'
06 B1
08 C1
01D1
QRYNAME
C1 DMKFREE CALL- GET STORAGE FOR MESSAGE AREA
D1
QRYNEWL
D1 CLEAR STORAGE AREA TO BLANKS
QRYNEXT
E1 QRYUSRN FORMAT USERID AND HIS TERMINAL LINE
F1 BUFFER COMPLETELY FULL
QRYCHKEN
G1 AT END OF CHAIN
H1 GET ADDRESS OF NEXT USER
QRYMSGL
G2 COMPUTE LENGTH OF PRINT LINE
H2 DMKQCNWT CALL- PRINT THIS LINE
J2 ALL NAMES BEEN OUTPUT
D1
NAMETERM
K2 DMKFRET CALL- RELEASE MESSAGE AREA
01 C3
08 C3
01D1
QRYUSER
C3 CLEAR MESSAGE AREA TO BLANKS
D3 DMKSCNFD CALL- GET NEXT ARGUMENT FROM INPUT BUFFER
E3 WAS ARGUMENT FOUND
07 B1
F3 DMKCVTBD CALL- CONVERT NUMBER OF USERS THAT 'DIALED'
G3 GET NUMBER OF USERS LOGGED ON
H3 DMKCVTBD CALL- CONVERT THIS FACTOR TO HEX
J3 STORE RESULT AND OUTPUT THIS RESPONSE LINE
01 B3
08 C4
06C2
QRYVFND
C4 REQUEST FOR 'ALL'
06 B1
D4 REQUEST FOR STORAGE SIZE
06 F1
06 C1

DMKCQG -- Process QUERY Command (Parts 7 and 8 cf 10)

DMKCQG -- Process QUERY Command (Parts 9 and 10 of 10)

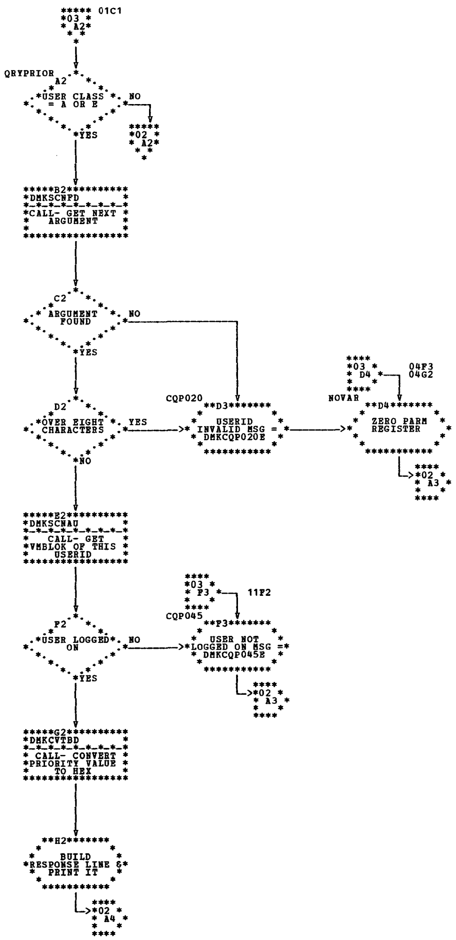

09 A1 02G3 10A2 10A3 10A4 10B3
DEDICATE A1 FOLLOWING CODE WILL FILL IN THE REAL DEVICE ADDRESS
10 A1
QRYVURO A2 DEVICE ATTACHED TO THIS USER.
YES
NO
QRYVURIO A3 UNIT RECORD OUTPUT DEVICE.
09 A3 02G3
NOTXFER B1 TRANSFER WAS NOT ACTIVE
B4
B2 TRANSFER IN EFFECT
C2 PUT 'TRANSFER' IN RESPONSE LINE AND PRINT IT
D3
B3 DEVICE TYPE = PSEUDO TIMER
C3 FILL IN REST OF LINE AND PRINT IT
09 D3 10B5 10D1 10D3 10D5 FNOTE
WRTVIRT D3 DMKQCNWT CALL- PRINT THIS LINE
E3 RETURN ON GPR-10
QRYVURI B4 SET CLASS CODE INTO RESPONSE LINE
C4 CONTINUOUS SPOOLING REQUIRED
D4 MOVE 'CONT' INTO RESPONSE LINE
NOCONT E4 UNIT RECORD INPUT DEVICE
NOHOLDRD E5 READER 'EOF' SWITCH ON
F5 MOVE 'NOEOF' INTO RESPONSE LINE
F4 PRT OR PUN BEING HELD
G4 INDICATE SO IN THE RESPONSE LINE
NOHOLD H4 DMKCVTBD CALL- CONVERT NUMBER OF COPIES
J4 FILL IN THE REMAINDER OF RESPONSE LINE AND PRINT IT
TO:D3 10E3 10F5 10J2

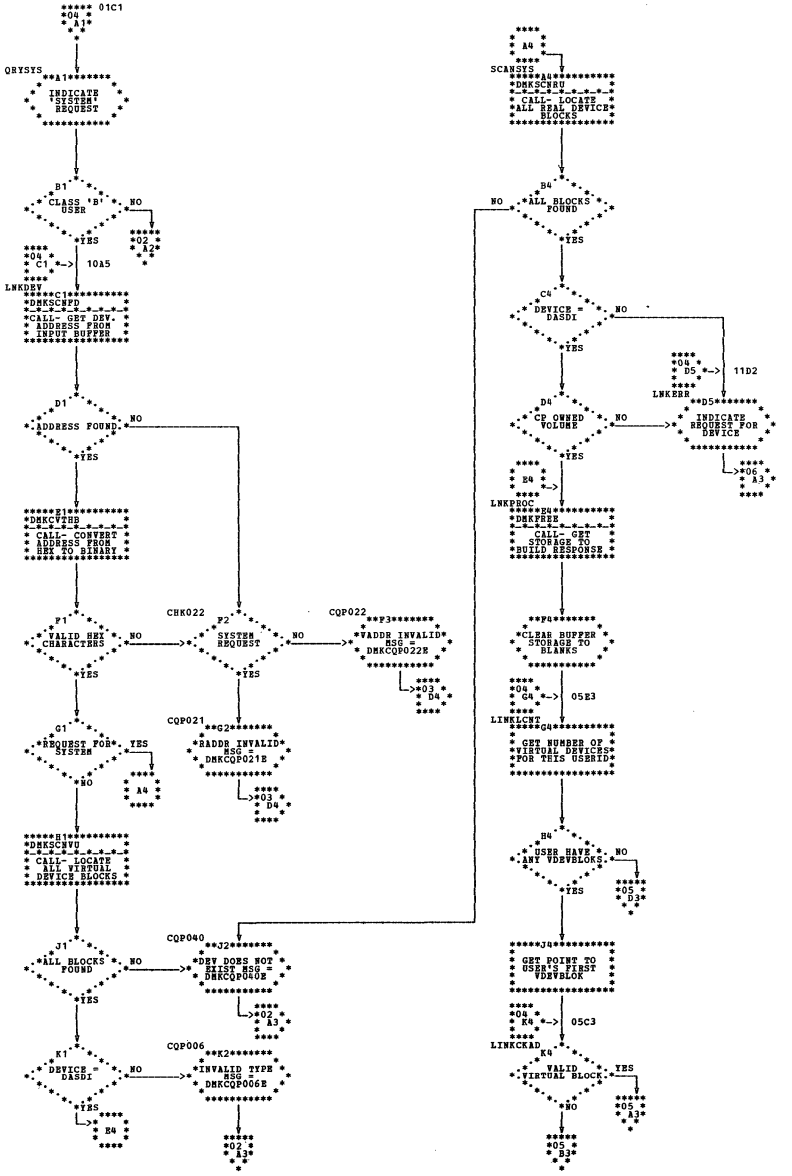

10 A1 09A1
FINDRDEV A1 GET ADDRESS OF THE RDEVBLOK
B1
FINDRTRM B1 GET DEVICE ADDRESS INTO GPR-1
C1 DMKCVTBH CALL- CONVERT ADDRESS FROM BINARY TO HEX
D1 FILL IN THE REMAINDER OF THE RESPONSE LINE
09 D3
10 A2 02G3
QRYVDASD A2 IS DASDI DEVICE ATTACH.
YES
NO
09 A1
B2 MOVE DEVICE TYPE INTO RESPONSE LINE
C2 STANDARD DASDI DEVICE TYPE
QRYSPEC D2 MOVE DEVICE TYPE INTO RESPONSE LINE
QRYVMINI E2 MINI 'T' DISK.
F2 MOVE '(TEMP)' INTO RESPONSE LINE
NOTEMP G2 DMKCVTBD CALL- CONVERT NUMBER OF CYLINDERS
H2 MOVE IN SERIAL NUMBER AND STATUS OF DISK.
J2 FINISH THE REMAINDER OF RESPONSE LINE & PRINT IT
10 A3 02G3
QRYVTERM A3 LINE DEDICATED
B3 DEVICE A VIRTUAL 270X LINE
C3 DEVICE TYPE = CONSOLE
NOTERM D3 VIRTUAL 270X LINE ENABLED
E3 CHANGE RESPONSE LINE TO 'DISABLED'
10 A4 02G3
QRYVGRAF A4 GO TO 'DEDICATE'
10 A5 02G3
QRYVSPEC A5 SET UP FOR UNSP MESSAGE
B5 CTCA
C5 SET UP NOT READY MESSAGE
D5 IS CTCA COUPLED
E5 SET UP COUPLE TO MESSAGE
F5 DMKCVTBH CALL - CONVERT DEVICE ADDRESS

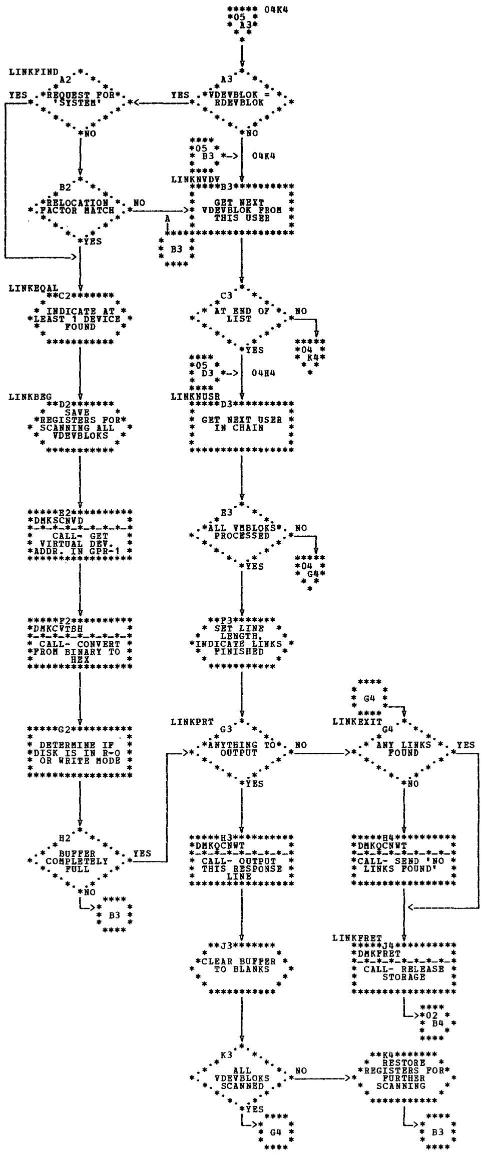


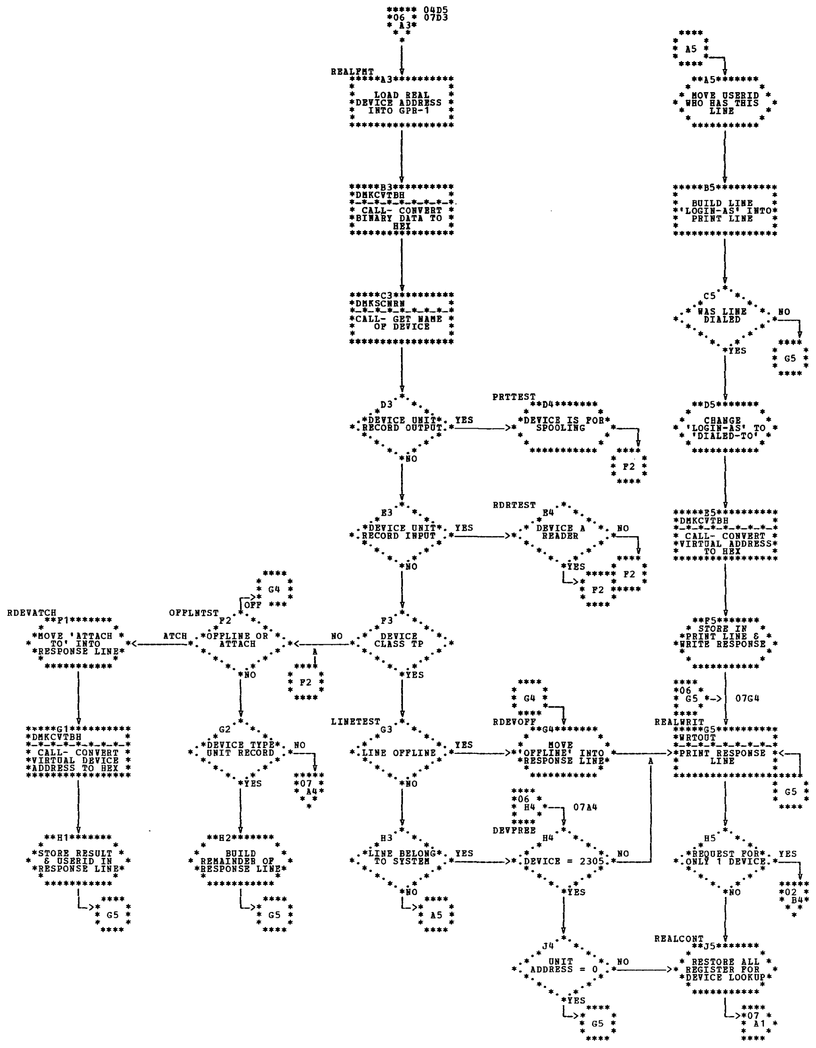
DMKCQP -- Process QUERY Command (Parts 1 and 2 of 11)

DMKCQP -- Process QUERY Command (Parts 3 and 4 cf 11)



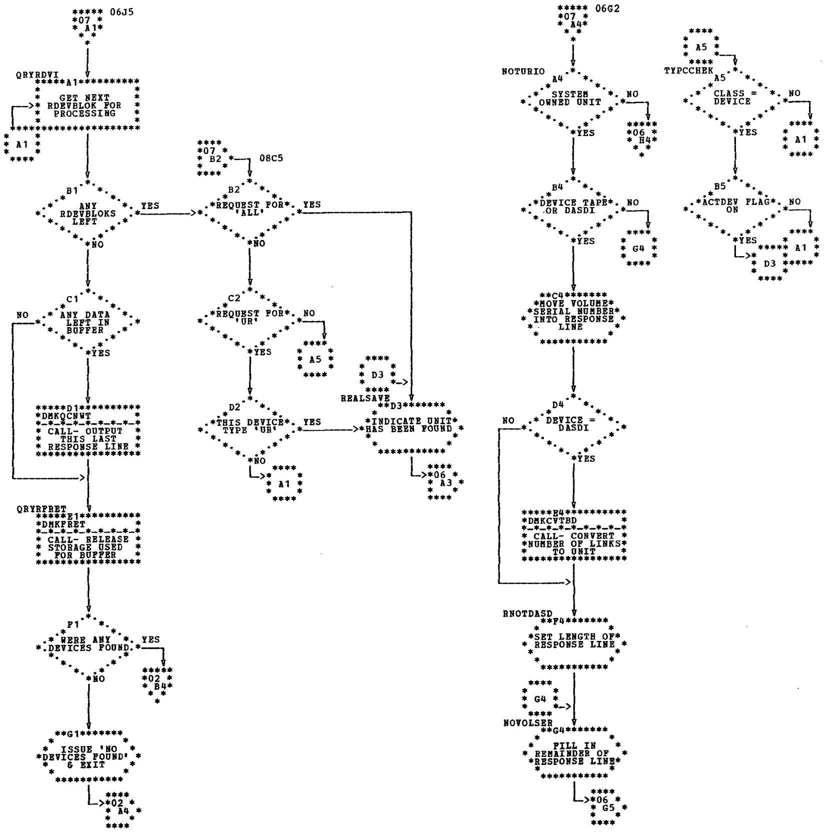


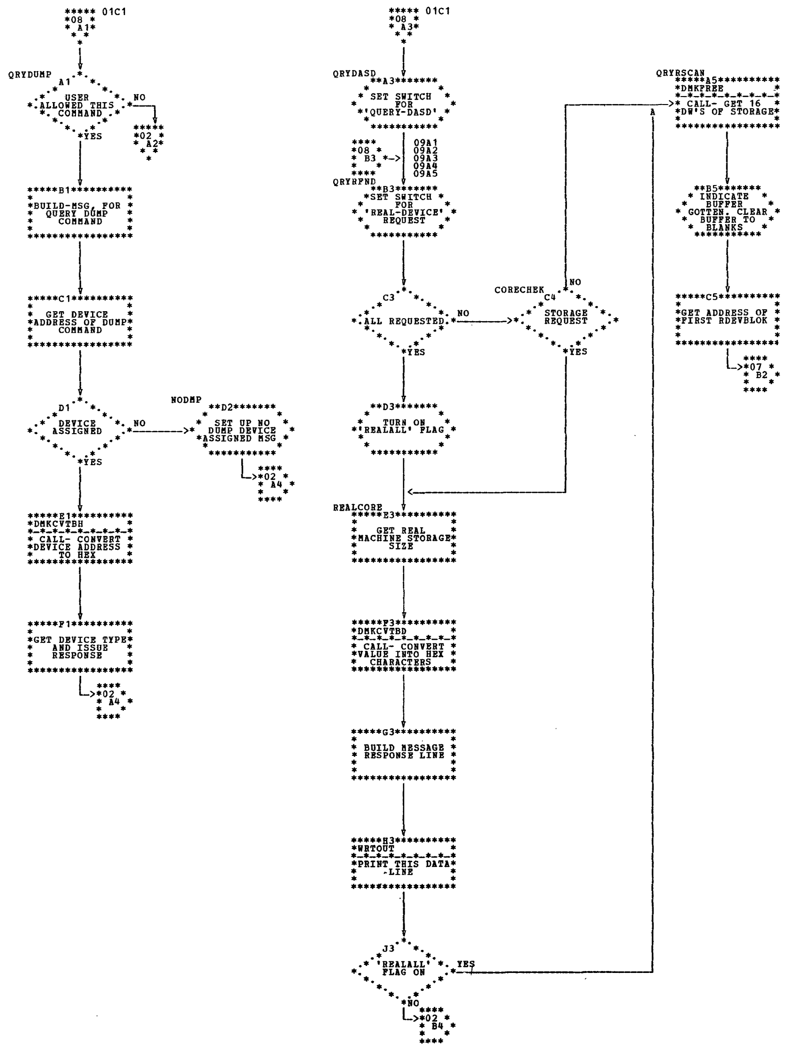


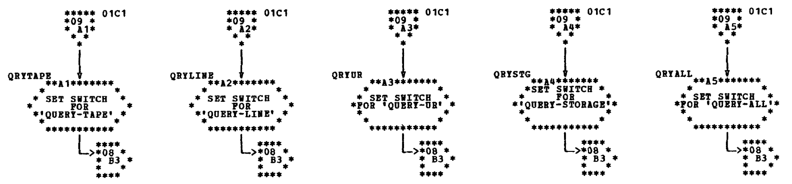


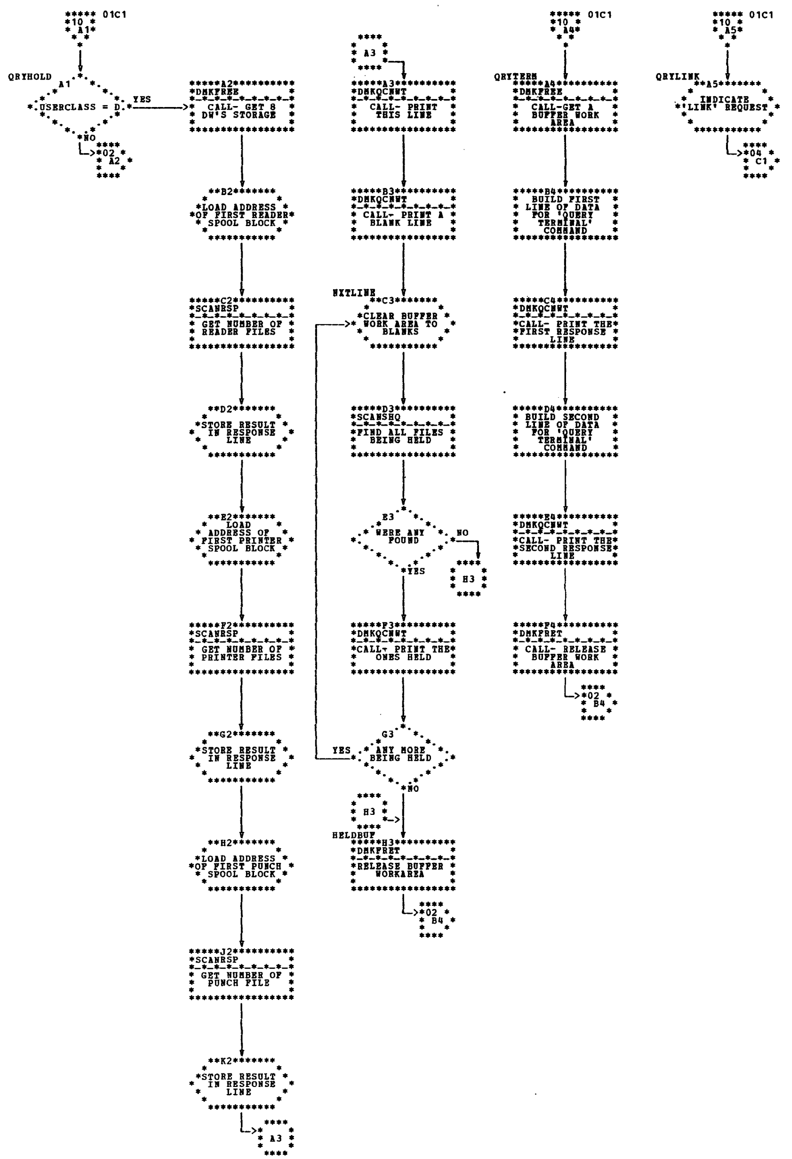
DMKCQP -- Process CUERY Command (Parts 5 and 6 cf 11)

DMKCQP -- Process QUERY Command (Parts 7 and 8 of 11)


07 A1 06J5
QRYRDVL A1 GET NEXT RDEVBLOK FOR PROCESSING
B1 ANY RDEVBLOKS LEFT
YES
NO
C1 ANY DATA LEFT IN BUFFER
NO
YES
D1 DMKQCNWT CALL- OUTPUT THIS LAST RESPONSE LINE
QRYRFRET E1 DMKFRET CALL- RELEASE STORAGE USED FOR BUFFER
F1 WERE ANY DEVICES FOUND
YES 02 B4
NO
G1 ISSUE 'NO DEVICES FOUND' & EXIT
02 A4
07 B2 08C5
B2 REQUEST FOR 'ALL'
YES
NO
C2 REQUEST FOR 'UR'
NO A5
YES
D2 THIS DEVICE TYPE 'UR'
YES
NO A1
D3
REALSAVE D3 INDICATE UNIT HAS BEEN FOUND
06 A3
07 A4 06G2
NOTURIO A4 SYSTEM OWNED UNIT
NO 06 H4
YES
B4 DEVICE TAPE OR DASDI
NO G4
YES
C4 MOVE VOLUME SERIAL NUMBER INTO RESPONSE LINE
D4 DEVICE = DASDI
NO
YES
E4 DMKCVTBD CALL- CONVERT NUMBER OF LINKS TO UNIT
RNOTDASD F4 SET LENGTH OF RESPONSE LINE
G4
NOVOLSER G4 FILL IN REMAINDER OF RESPONSE LINE
06 G5
A5
TYPCCHEK A5 CLASS = DEVICE
NO A1
YES
B5 ACTDEV FLAG ON
NO A1
YES D3

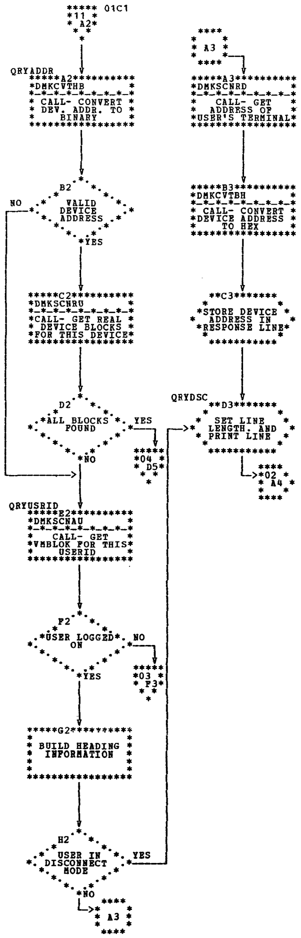

08 A1 01C1
QRYDUMP A1 USER ALLOWED THIS COMMAND
NO 02 A2
YES
B1 BUILD-MSG FOR QUERY DUMP COMMAND
C1 GET DEVICE ADDRESS OF DUMP COMMAND
D1 DEVICE ASSIGNED
NO
YES
NODMP D2 SET UP NO DUMP DEVICE ASSIGNED MSG
02 A4
E1 DMKCVTBD CALL- CONVERT DEVICE ADDRESS TO HEX
F1 GET DEVICE TYPE AND ISSUE RESPONSE
02 A4
08 A3 01C1
QRYDASD A3 SET SWITCH FOR 'QUERY-DASD'
08 B3 09A1 09A2 09A3 09A4 09A5
QRYRFND B3 SET SWITCH FOR 'REAL-DEVICE' REQUEST
C3 ALL REQUESTED
NO
YES
CORECHEK C4 STORAGE REQUEST
NO
YES
D3 TURN ON 'REALALL' FLAG
REALCORE E3 GET REAL MACHINE STORAGE SIZE
F3 DMKCVTBD CALL- CONVERT VALUE INTO HEX CHARACTERS
G3 BUILD MESSAGE RESPONSE LINE
H3 DMKQCNWT PRINT THIS DATA LINE
J3 'REALALL' FLAG ON
YES
NO 02 B4
A
QRYRSCAN A5 DMKFREE CALL- GET 16 DW'S OF STORAGE
B5 INDICATE BUFFER GOTTEN. CLEAR BUFFER TO BLANKS
C5 GET ADDRESS OF FIRST RDEVBLOK
07 B2

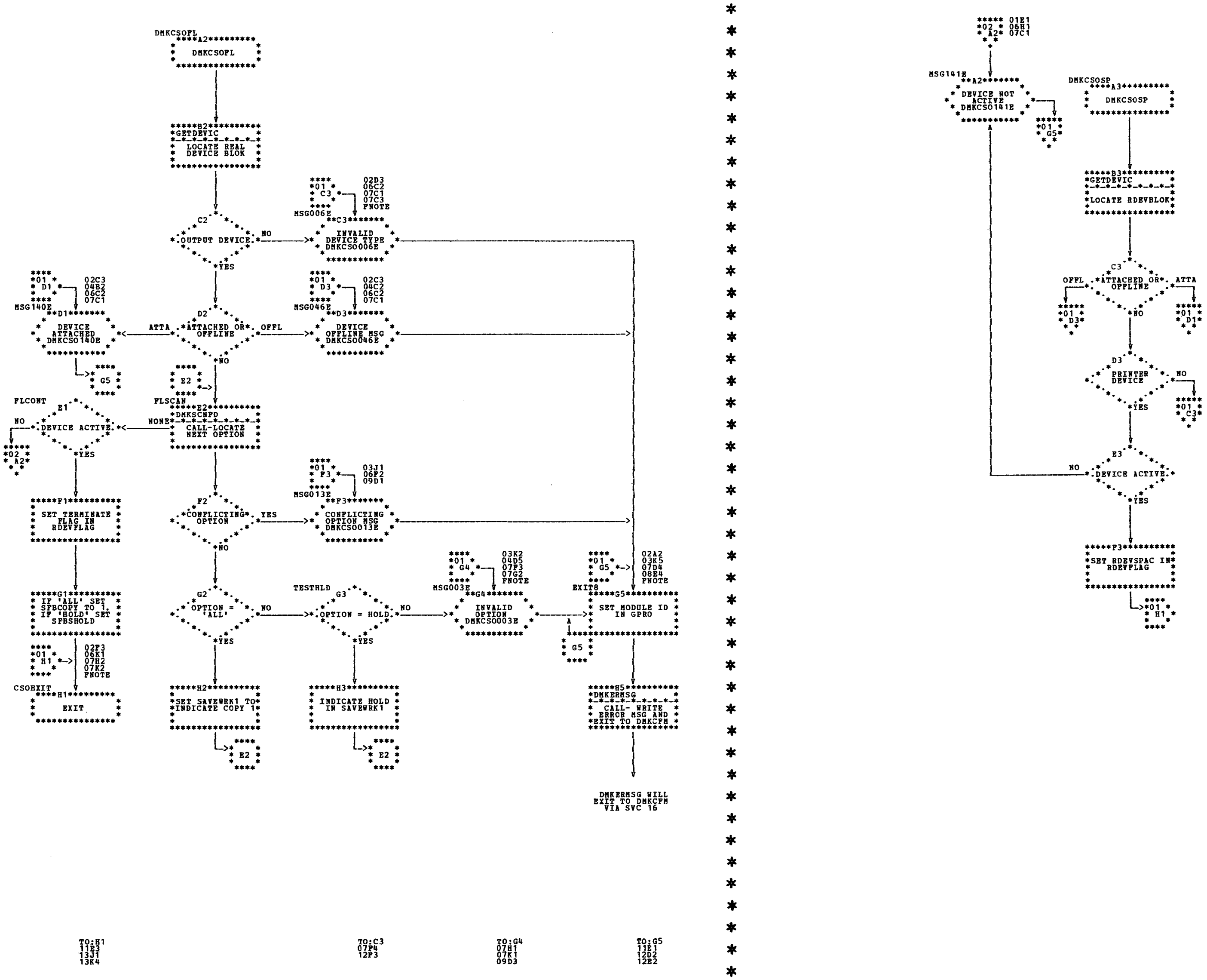

01C1
09 A1
QRYTAPE
A1
SET SWITCH FOR 'QUERY-TAPE'
08 B3
01C1
09 A2
QRYLINE
A2
SET SWITCH FOR 'QUERY-LINE'
08 B3
01C1
09 A3
QRYUR
A3
SET SWITCH FOR 'QUERY-UR'
08 B3
01C1
09 A4
QRYSTG
A4
SET SWITCH FOR 'QUERY-STORAGE'
08 B3
01C1
09 A5
QRYALL
A5
SET SWITCH FOR 'QUERY-ALL'
08 B3


01C1
10 A1
QRYHOLD
A1
USERCLASS = D.
YES
NO
02 A2
A2
DMKFREE
CALL- GET 8 DW'S STORAGE
B2
LOAD ADDRESS OF FIRST READER SPOOL BLOCK
C2
SCANRSP
GET NUMBER OF READER FILES
D2
STORE RESULT IN RESPONSE LINE
E2
LOAD ADDRESS OF FIRST PRINTER SPOOL BLOCK
F2
SCANRSP
GET NUMBER OF PRINTER FILES
G2
STORE RESULT IN RESPONSE LINE
H2
LOAD ADDRESS OF FIRST PUNCH SPOOL BLOCK
J2
SCANRSP
GET NUMBER OF PUNCH FILE
K2
STORE RESULT IN RESPONSE LINE
A3
A3
A3
DMKQCNWT
CALL- PRINT THIS LINE
B3
DMKQCNWT
CALL- PRINT A BLANK LINE
NXTLINE
C3
CLEAR BUFFER WORK AREA TO BLANKS
D3
SCANSHQ
FIND ALL FILES BEING HELD
E3
WERE ANY FOUND
NO
H3
YES
F3
DMKQCNWT
CALL- PRINT THE ONES HELD
G3
ANY MORE BEING HELD
YES
NO
H3
HELDBUF
H3
DMKFRET
RELEASE BUFFER WORKAREA
02 B4
01C1
10 A4
QRYTERM
A4
DMKFREE
CALL-GET A BUFFER WORK AREA
B4
BUILD FIRST LINE OF DATA FOR 'QUERY TERMINAL' COMMAND
C4
DMKQCNWT
CALL- PRINT THE FIRST RESPONSE LINE
D4
BUILD SECOND LINE OF DATA FOR 'QUERY TERMINAL' COMMAND
E4
DMKQCNWT
CALL- PRINT THE SECOND RESPONSE LINE
F4
DMKFRET
CALL- RELEASE BUFFER WORK AREA
02 B4
01C1
10 A5
QRYLINK
A5
INDICATE 'LINK' REQUEST
04 C1

DMKCQP -- Process QUERY Command (Parts 9 and 10 of 11)

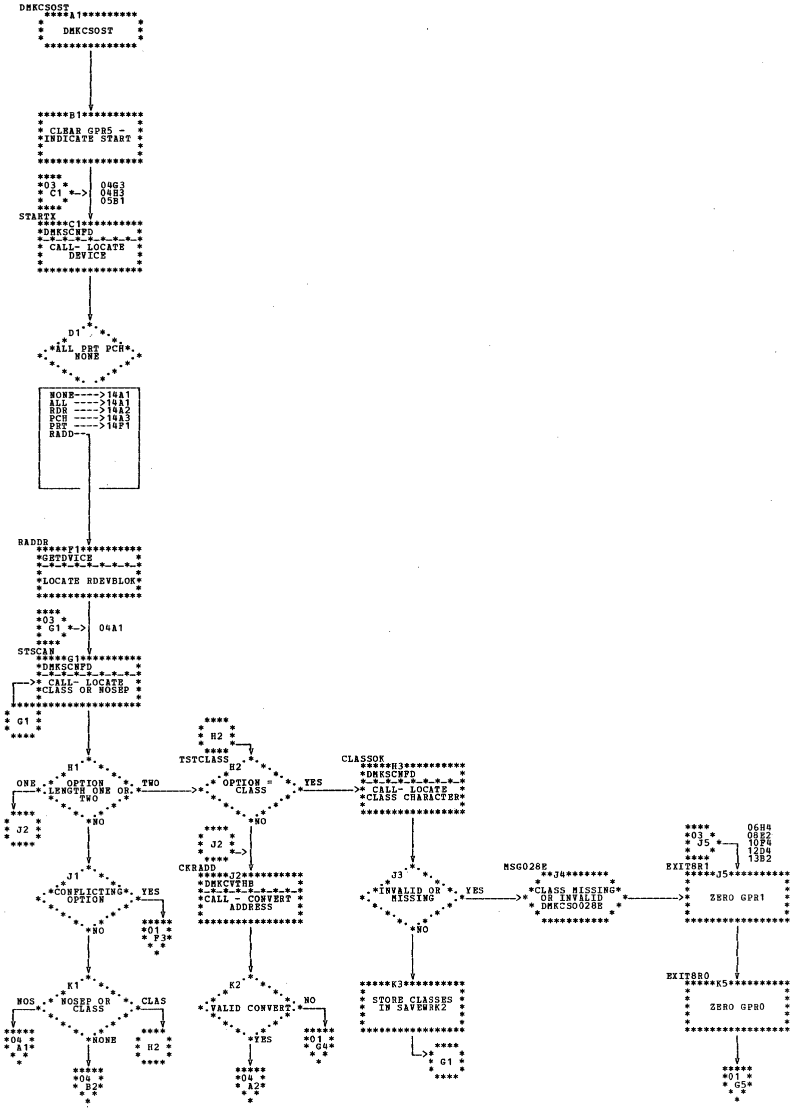
DMKCQP -- Process QUERY Command (Part 11 of 11)

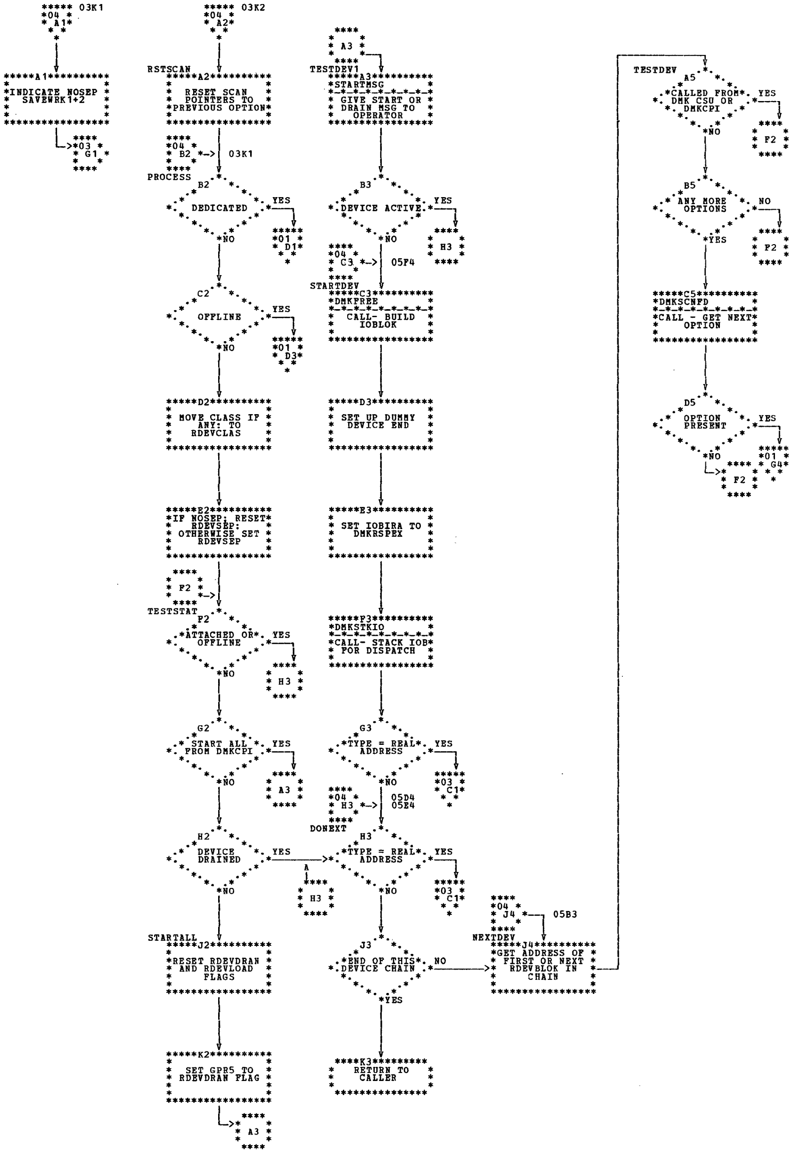

01C1
11
A2
QRYADDR
A2
DMKCVTHB
CALL- CONVERT DEV. ADDR. TO BINARY
A3
A3
DMKSCNRD
CALL- GET ADDRESS OF USER'S TERMINAL
B2
VALID DEVICE ADDRESS
NO
YES
B3
DMKCVTBH
CALL- CONVERT DEVICE ADDRESS TO HEX
C2
DMKSCNRU
CALL- GET REAL DEVICE BLOCKS FOR THIS DEVICE
C3
STORE DEVICE ADDRESS IN RESPONSE LINE
D2
ALL BLOCKS FOUND
YES
04
D5
NO
QRYDSC
D3
SET LINE LENGTH. AND PRINT LINE
02
A4
QRYUSRID
E2
DMKSCNAU
CALL- GET VMBLOK FOR THIS USERID
F2
USER LOGGED ON
NO
03
F3
YES
G2
BUILD HEADING INFORMATION
H2
USER IN DISCONNECT MODE
YES
NO
A3

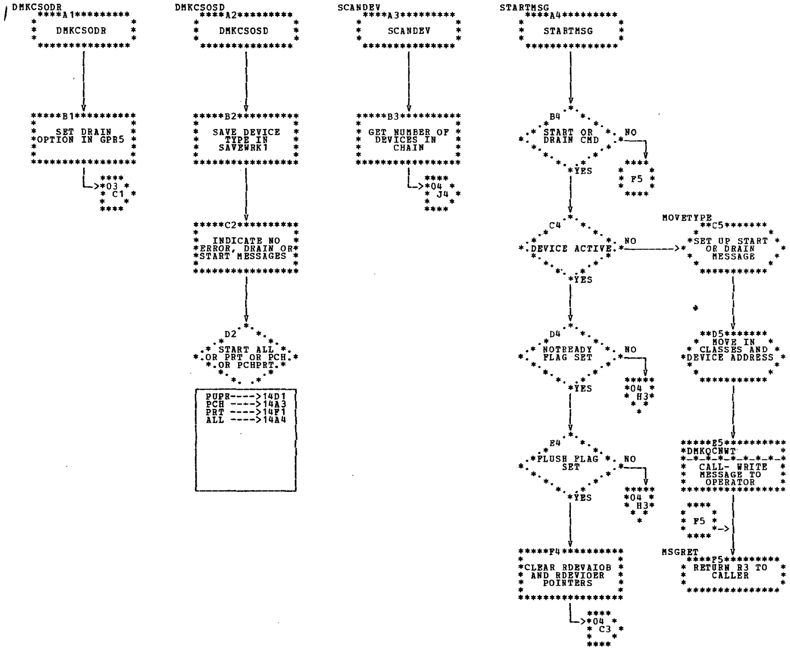


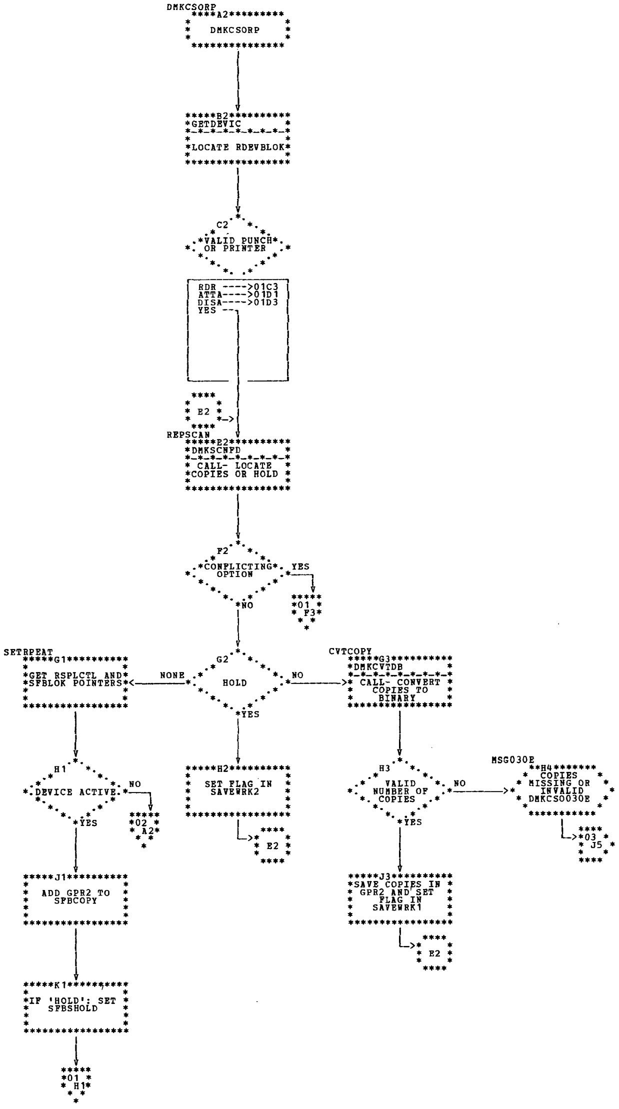
| DMKCSC -- Process Spooling Commands (Parts 1 and 2 of 14)

| DMKCSC -- Process Spooling Commands (Parts 3 and 4 of 14)



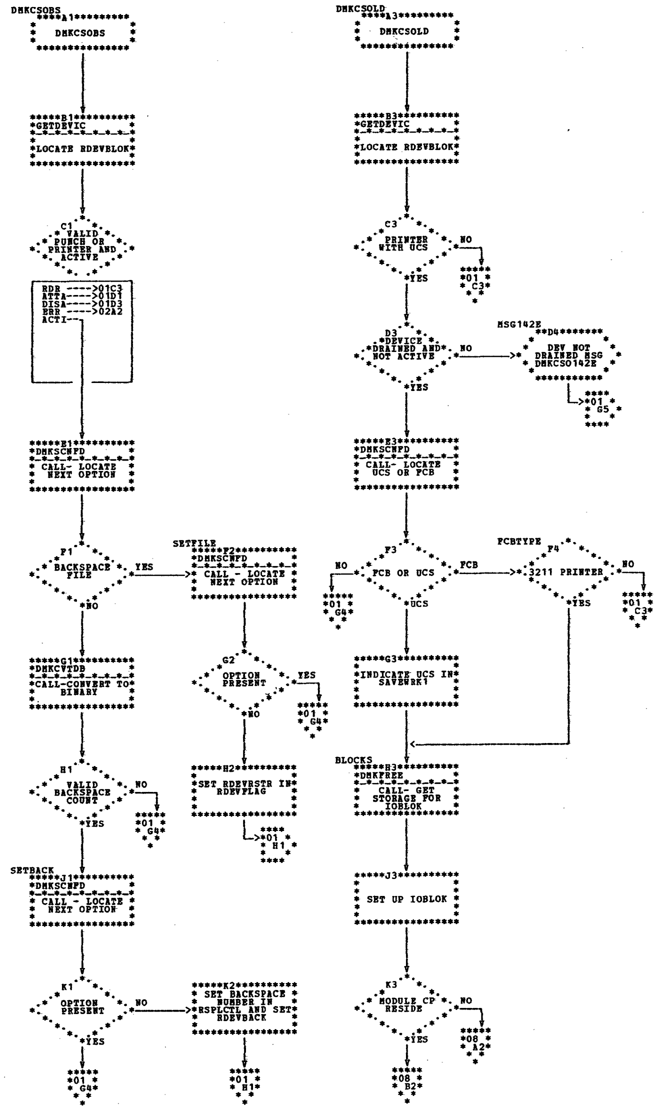


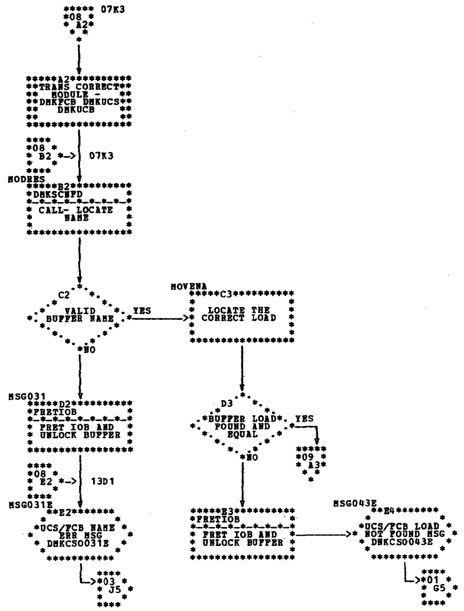


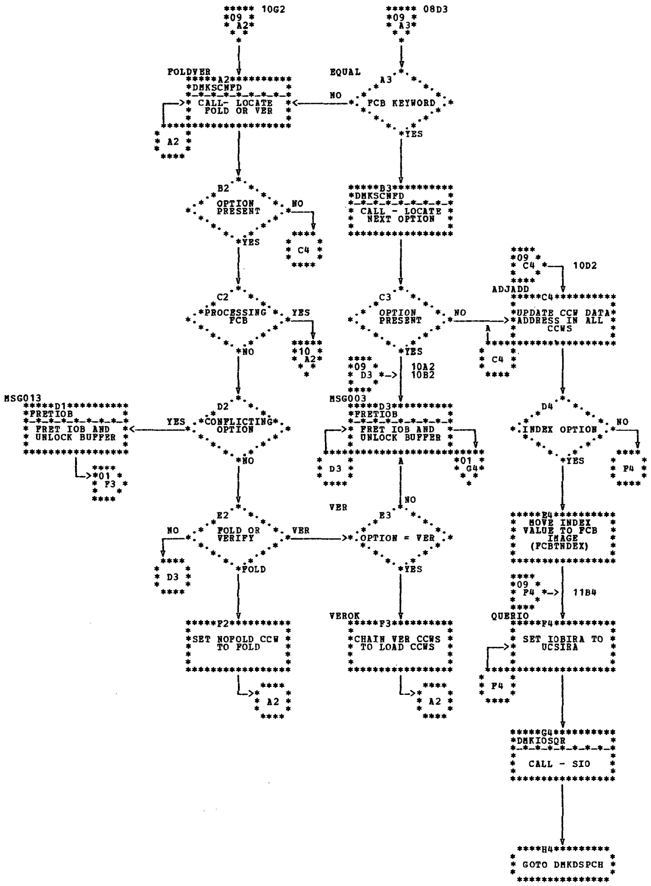


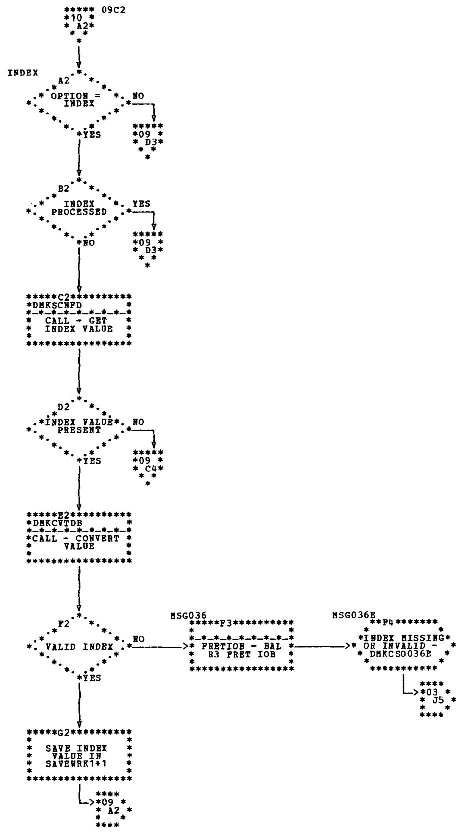
| DMKCSC -- Process Spooling Commands (Parts 5 and 6 of 14)

| DMKCSO -- Process Spooling Commands (Parts 7 and 8 of 14)



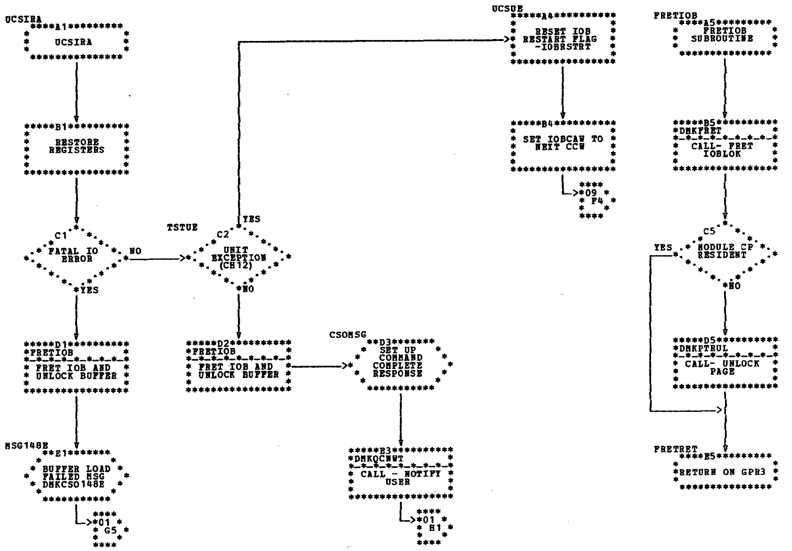

08 A2
07K3
A2
TRANS CORRECT MODULE - DMKFCB DMKUCS DMKUCB
08 B2
07K3
MODRES
B2
DMKSCNFD
CALL- LOCATE NAME
C2
VALID BUFFER NAME
YES
MOVENA
C3
LOCATE THE CORRECT LOAD
NO
MSG031
D2
FRETIOB
FRET IOB AND UNLOCK BUFFER
D3
BUFFER LOAD FOUND AND EQUAL
YES
09 A3
NO
08 E2
13D1
MSG031E
E2
UCS/FCB NAME ERR MSG DMKCSO031E
03 J5
E3
FRETIOB
FRET IOB AND UNLOCK BUFFER
MSG043E
E4
UCS/FCB LOAD NOT FOUND MSG DMKCSO043E
01 G5

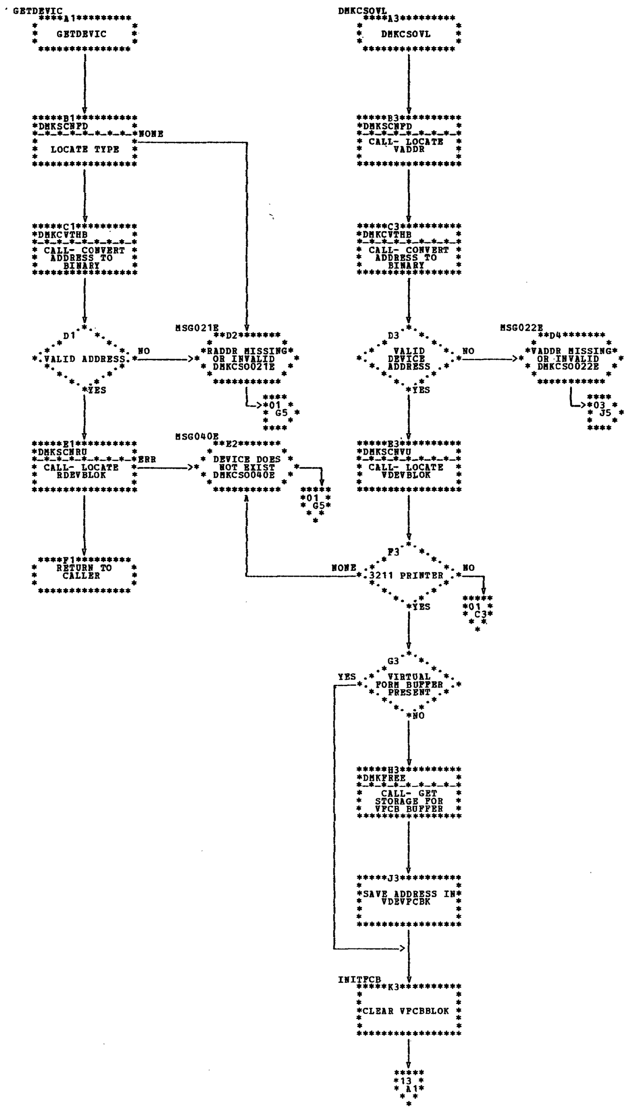


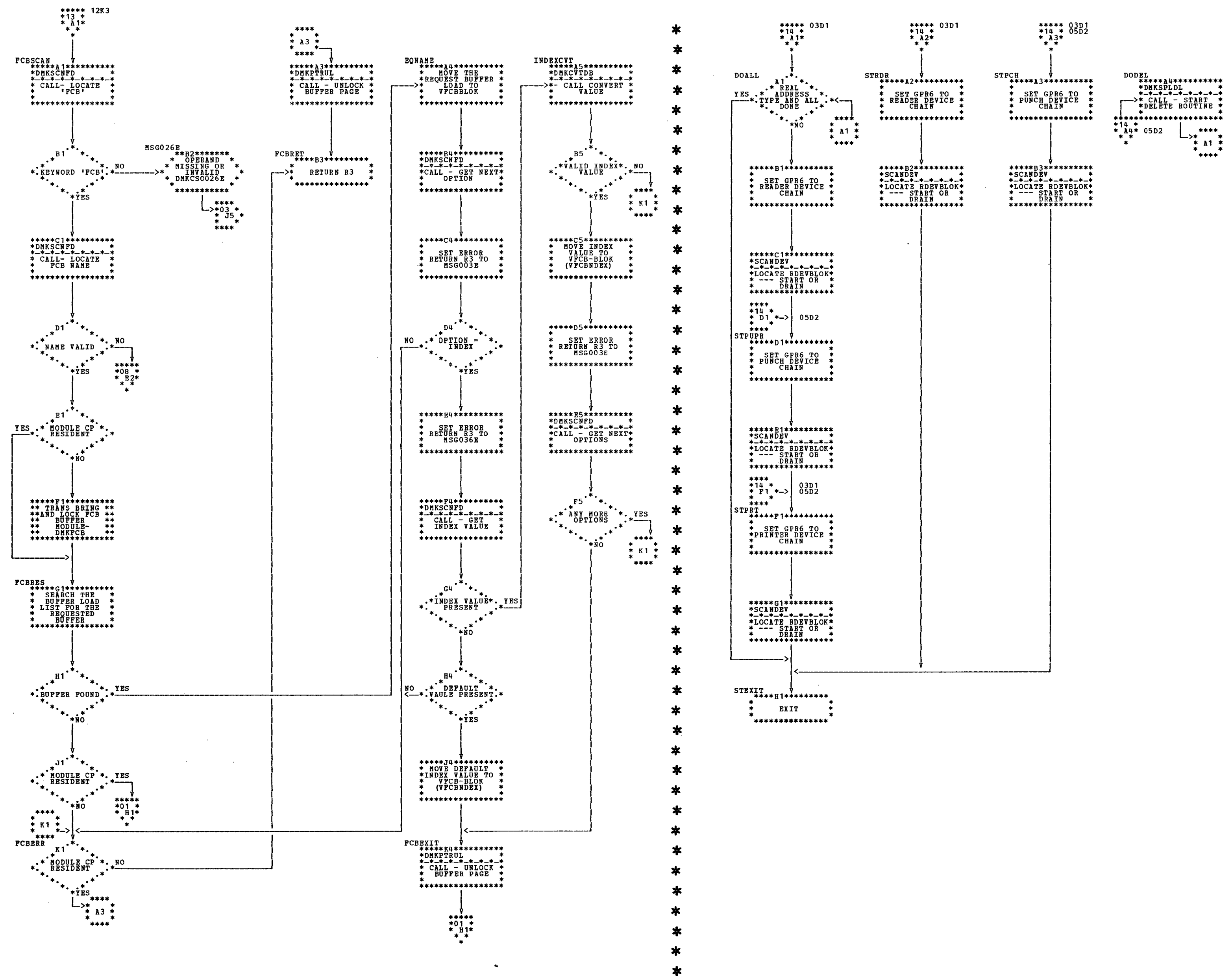


| DMKCSO -- Process Spooling Commands (Parts 9 and 10 of 14)

| DMKCSC -- Process Spooling Commands (Parts 11 and 12 cf 14)

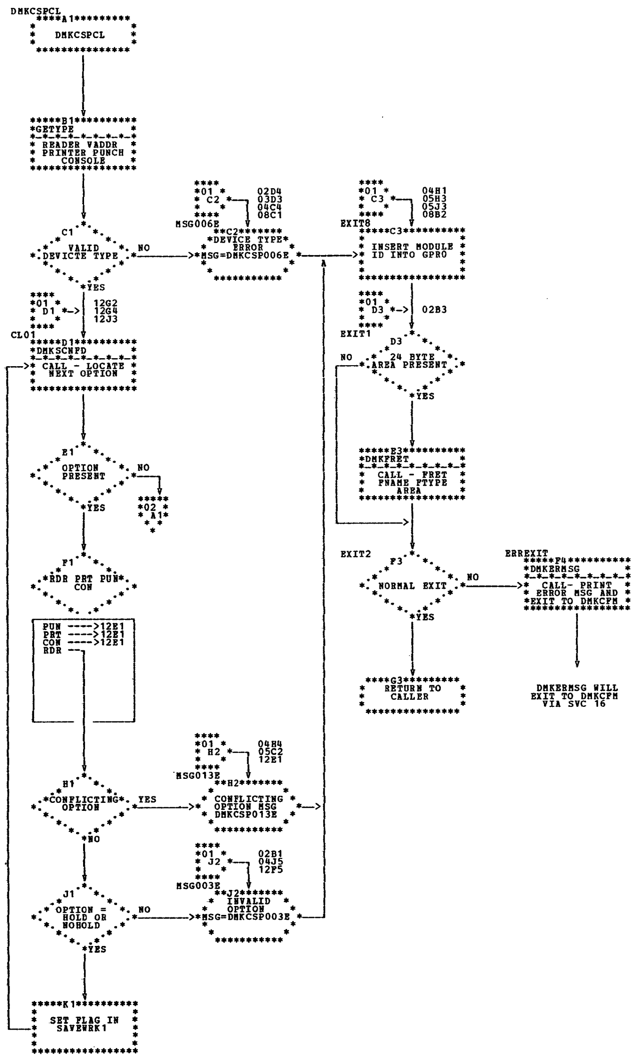


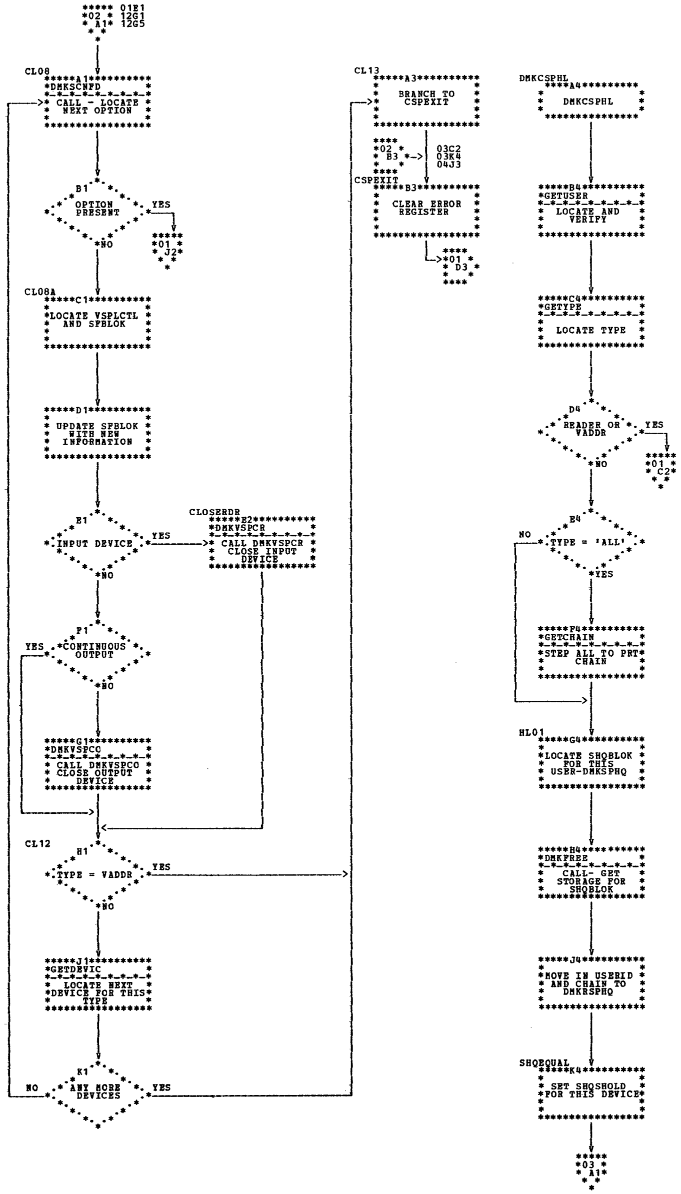


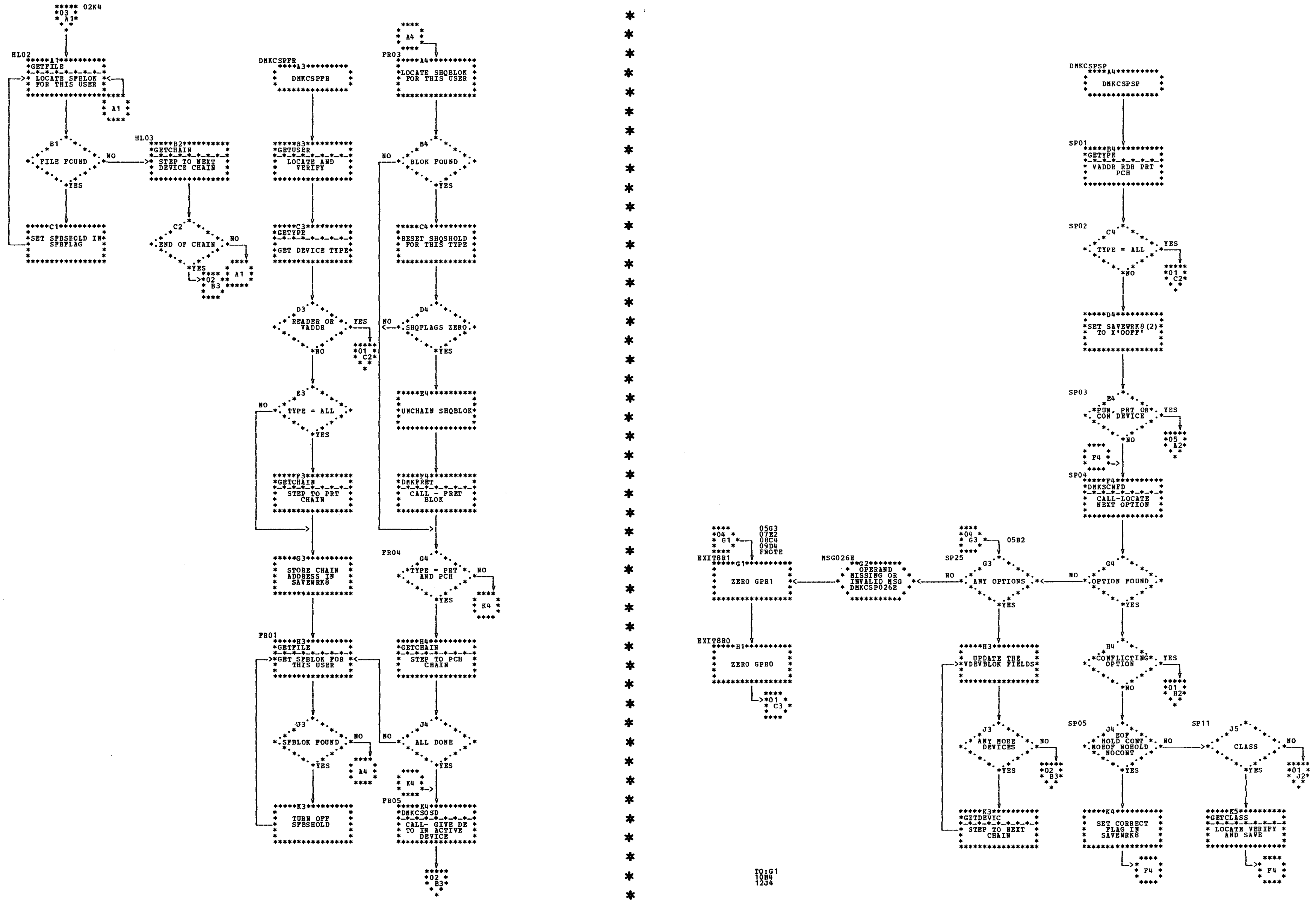

13 A1 12K3
FCBSCAN
A1 DMKSCNFD CALL- LOCATE 'FCB'
B1 KEYWORD 'FCB'
NO
YES
MSG026E
B2 OPERAND MISSING OR INVALID DMKCSO026E
03 J5
C1 DMKSCNFD CALL- LOCATE FCB NAME
D1 NAME VALID
NO
08 E2
YES
E1 MODULE CP RESIDENT
YES
NO
F1 TRANS BRING AND LOCK FCB BUFFER MODULE- DMKFCB
FCBRES
G1 SEARCH THE BUFFER LOAD LIST FOR THE REQUESTED BUFFER
H1 BUFFER FOUND
YES
NO
J1 MODULE CP RESIDENT
YES
01 H1
NO
K1
FCBERR
K1 MODULE CP RESIDENT
NO
YES
A3
A3
A3 DMKPTRUL CALL - UNLOCK BUFFER PAGE
FCBRET
B3 RETURN R3
EQNAME
A4 MOVE THE REQUEST BUFFER LOAD TO VFCBBLOK
B4 DMKSCNFD CALL - GET NEXT OPTION
C4 SET ERROR RETURN R3 TO MSG003E
D4 OPTION = INDEX
NO
YES
E4 SET ERROR RETURN R3 TO MSG036E
F4 DMKSCNFD CALL - GET INDEX VALUE
G4 INDEX VALUE PRESENT
YES
NO
H4 DEFAULT VAULE PRESENT
NO
YES
J4 MOVE DEFAULT INDEX VALUE TO VFCB-BLOK (VFCBNDEX)
FCBEXIT
K4 DMKPTRUL CALL - UNLOCK BUFFER PAGE
01 H1
INDEXCVT
A5 DMKCVTDB - CALL CONVERT VALUE
B5 VALID INDEX VALUE
NO
K1
YES
C5 MOVE INDEX VALUE TO VFCB-BLOK (VFCBNDEX)
D5 SET ERROR RETURN R3 TO MSG003E
E5 DMKSCNFD CALL - GET NEXT OPTIONS
F5 ANY MORE OPTIONS
YES
K1
NO
14 A1 03D1
DOALL
A1 REAL ADDRESS TYPE AND ALL DONE
YES
NO
A1
B1 SET GPR6 TO READER DEVICE CHAIN
C1 SCANDEV LOCATE RDEVBLOK --- START OR DRAIN
14 D1 05D2
STPUPR
D1 SET GPR6 TO PUNCH DEVICE CHAIN
E1 SCANDEV LOCATE RDEVBLOK --- START OR DRAIN
14 F1 03D1 05D2
STPRT
F1 SET GPR6 TO PRINTER DEVICE CHAIN
G1 SCANDEV LOCATE RDEVBLOK --- START OR DRAIN
STEXIT
H1 EXIT
14 A2 03D1
STRDR
A2 SET GPR6 TO READER DEVICE CHAIN
B2 SCANDEV LOCATE RDEVBLOK --- START OR DRAIN
14 A3 03D1 05D2
STPCH
A3 SET GPR6 TO PUNCH DEVICE CHAIN
B3 SCANDEV LOCATE RDEVBLOK --- START OR DRAIN
DODEL
A4 DMKSPLDL CALL - START DELETE ROUTINE
14 A4 05D2
A1

DMKCSO -- Process Spooling Commands (Parts 13 and 14 of 14)

| DMKCSP -- Process Spooling Commands (Parts 1 and 2 of 12)

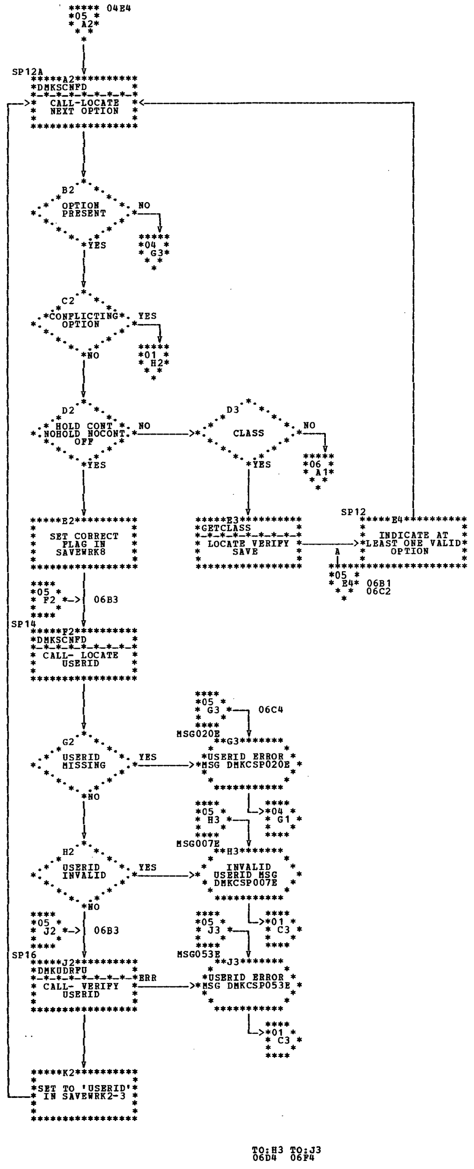


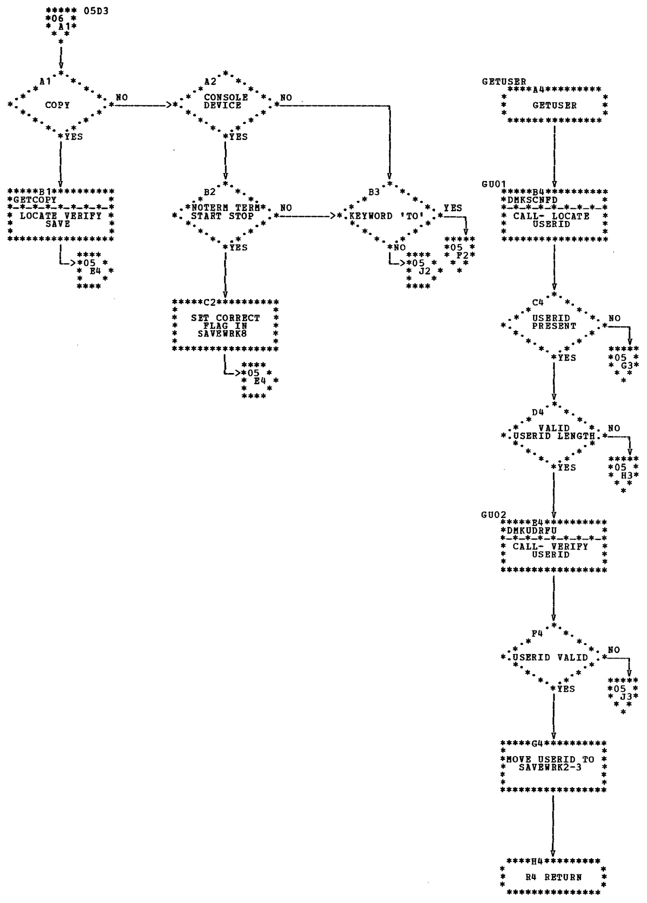


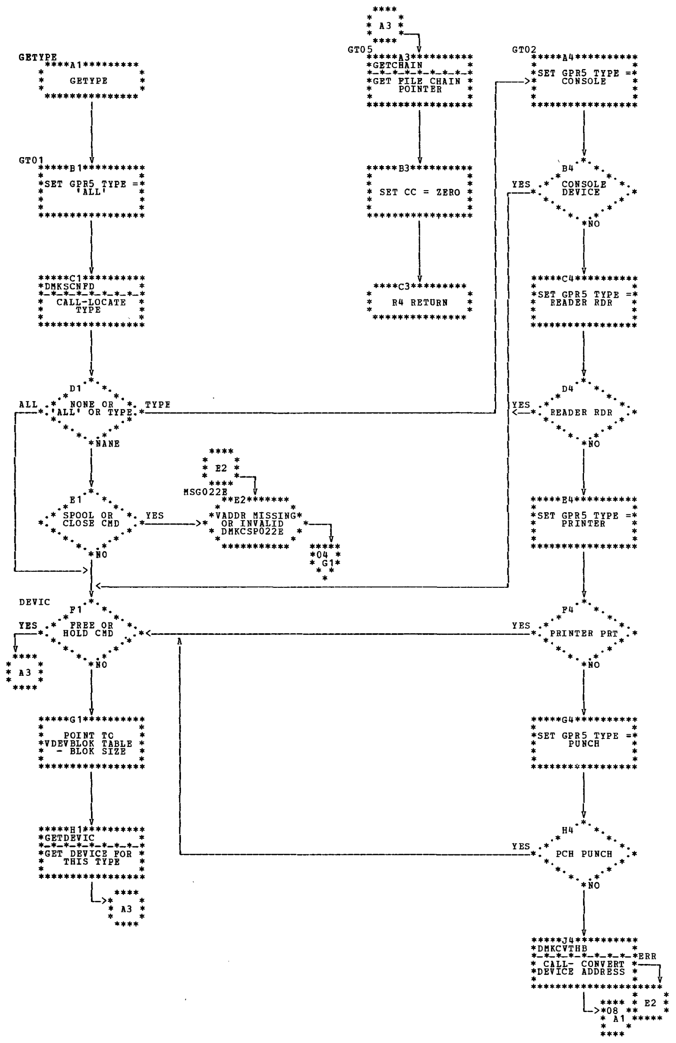


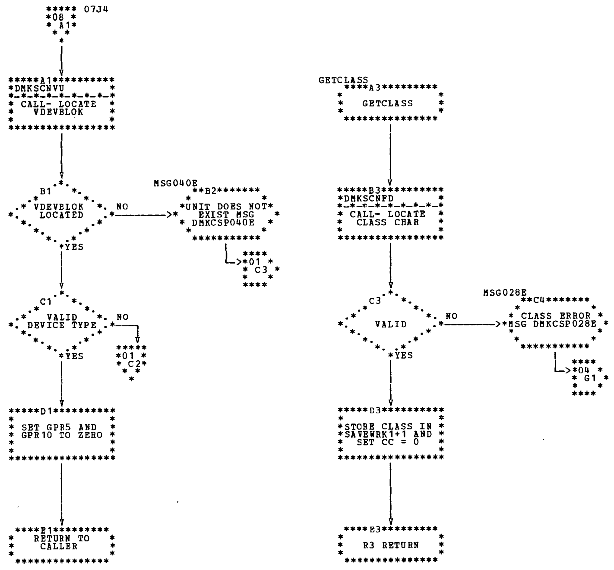
| DMKCSP -- Process Spooling Commands (Parts 3 and 4 of 12)

| DMKCSP -- Process Spooling Commands (Parts 5 and 6 of 12)


05 A2 04E4
SP12A
A2 DMKSCNFD CALL-LOCATE NEXT OPTION
B2 OPTION PRESENT
NO 04 G3
YES
C2 CONFLICTING OPTION
YES 01 H2
NO
D2 HOLD CONT NOHOLD NOCONT OFF
NO
D3 CLASS
NO 06 A1
YES
E2 SET CORRECT FLAG IN SAVEWRK8
E3 GETCLASS LOCATE VERIFY SAVE
SP12
E4 INDICATE AT LEAST ONE VALID OPTION
A
05 E4 06B1 06C2
05 F2 06B3
SP14
F2 DMKSCNFD CALL- LOCATE USERID
05 G3 06C4
MSG020E
G2 USERID MISSING
YES
G3 USERID ERROR MSG DMKCSP020E
04 G1
NO
05 H3
MSG007E
H2 USERID INVALID
YES
H3 INVALID USERID MSG DMKCSP007E
01 C3
NO
05 J2 06B3
05 J3
MSG053E
SP16
J2 DMKUDRFU CALL- VERIFY USERID
ERR
J3 USERID ERROR MSG DMKCSP053E
01 C3
K2 SET TO 'USERID' IN SAVEWRK2-3
TO:H3 06D4
TO:J3 06F4

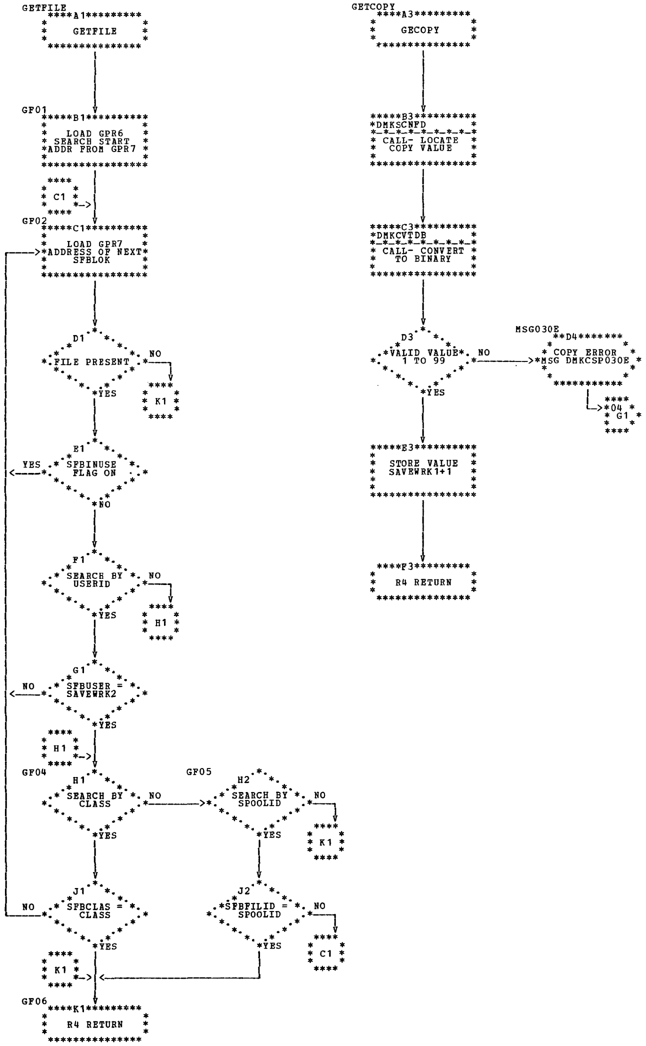


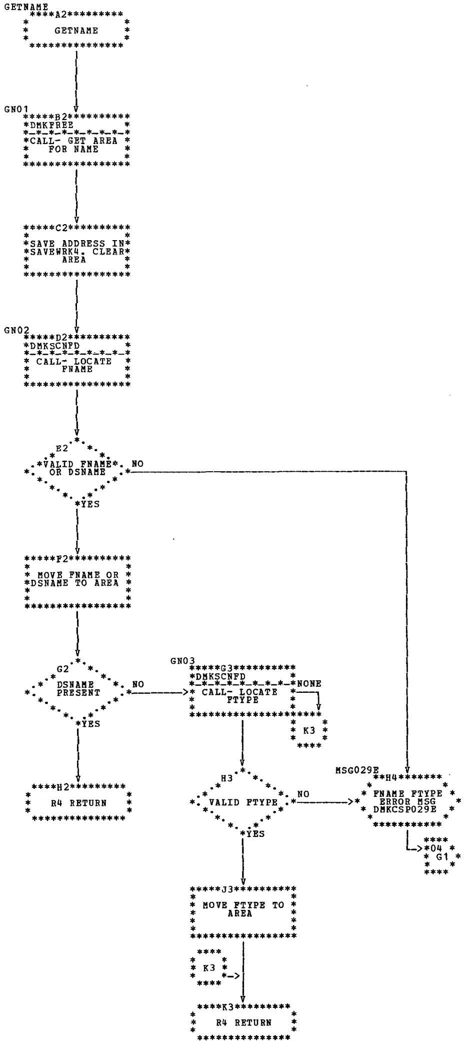


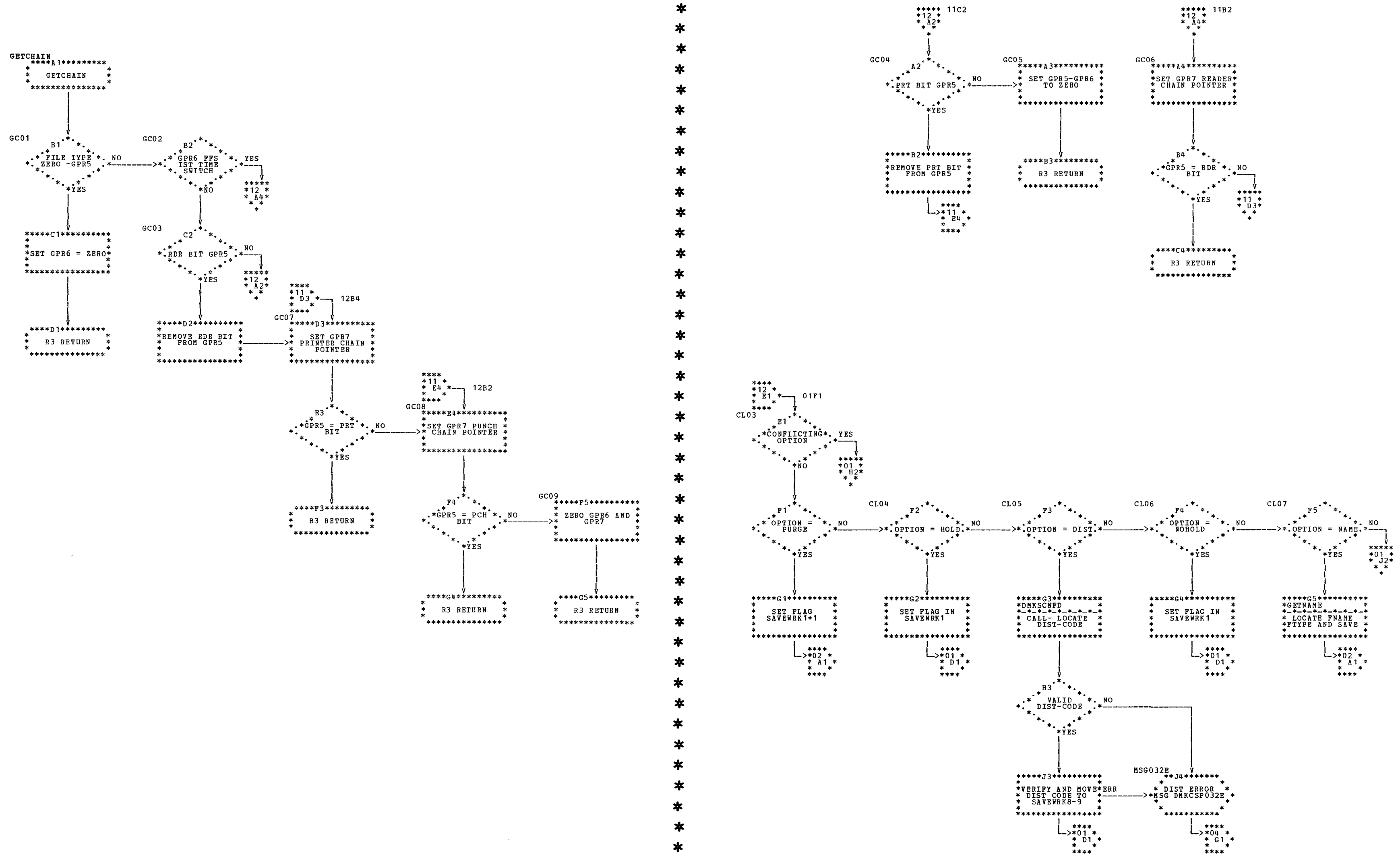


| DMKCSP -- Process Spooling Commands (Parts 7 and 8 of 12)

| DMKCSP -- Process Spooling Commands (Parts 9 and 10 of 12)

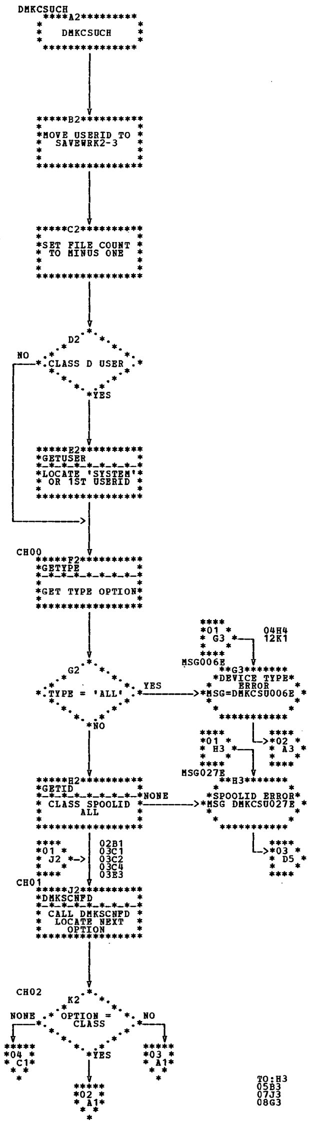


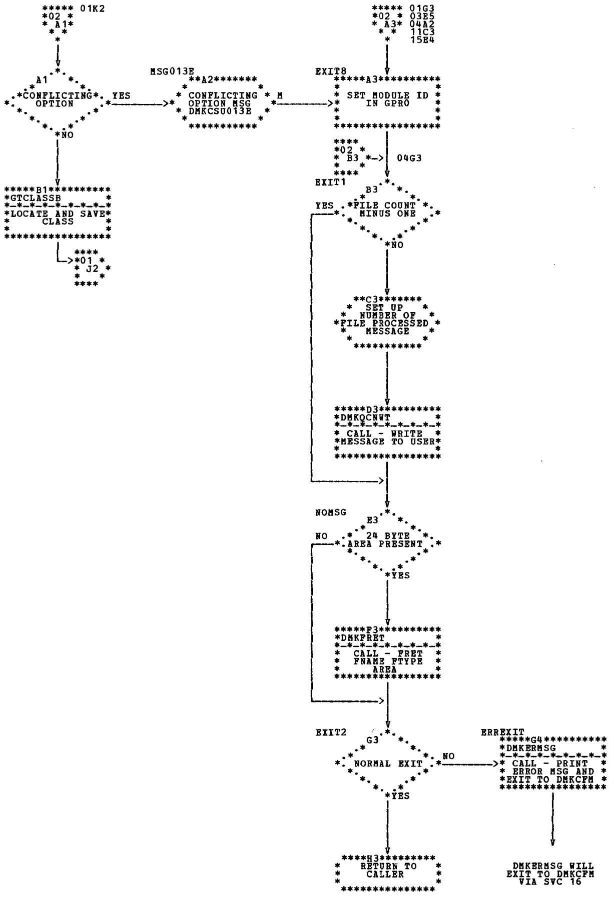


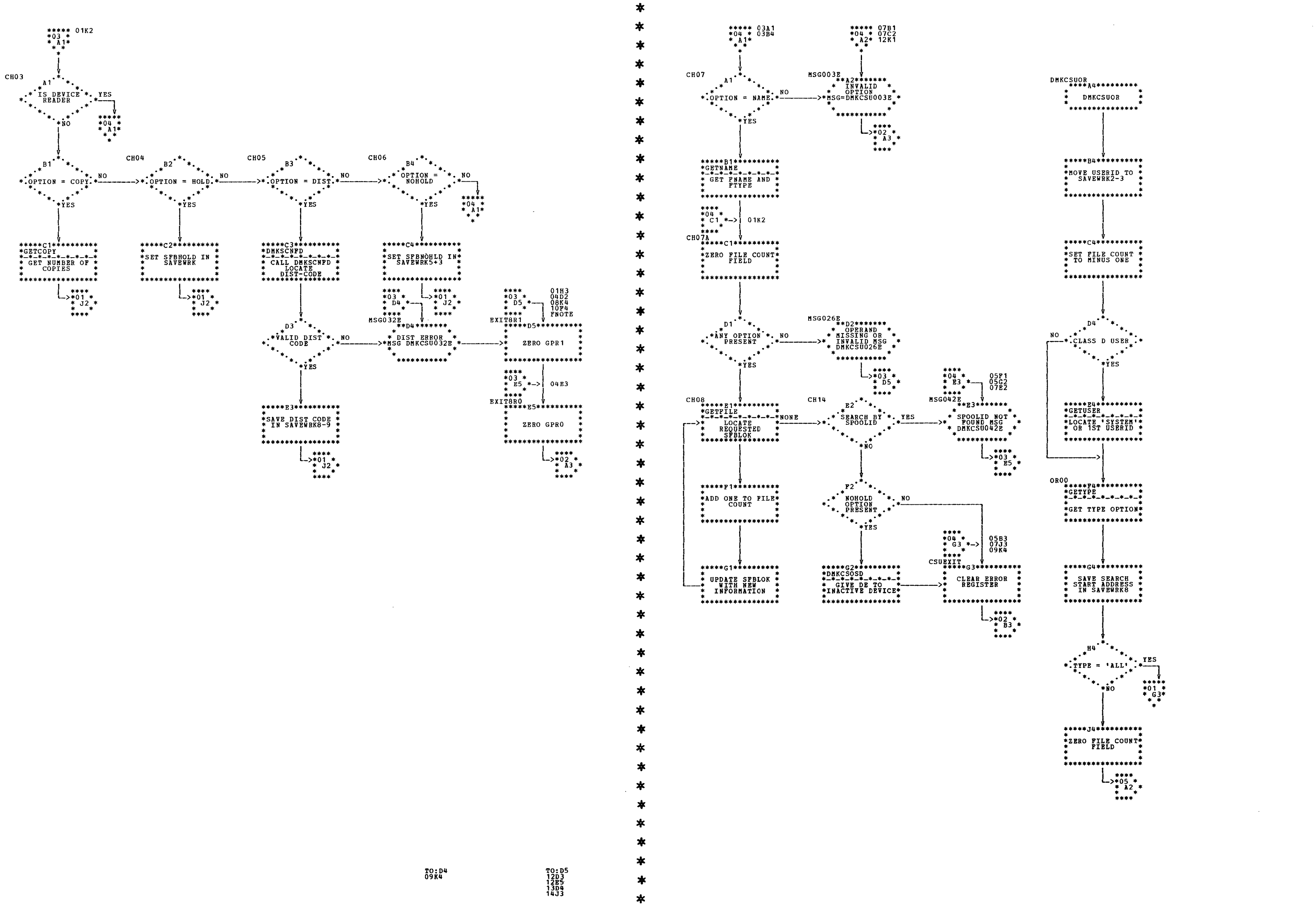


| DMKCSP -- Process Spooling Commands (Parts 11 and 12 of 12)

DMKCSU -- Process Spooling Commands (Parts 1 and 2 of 17)

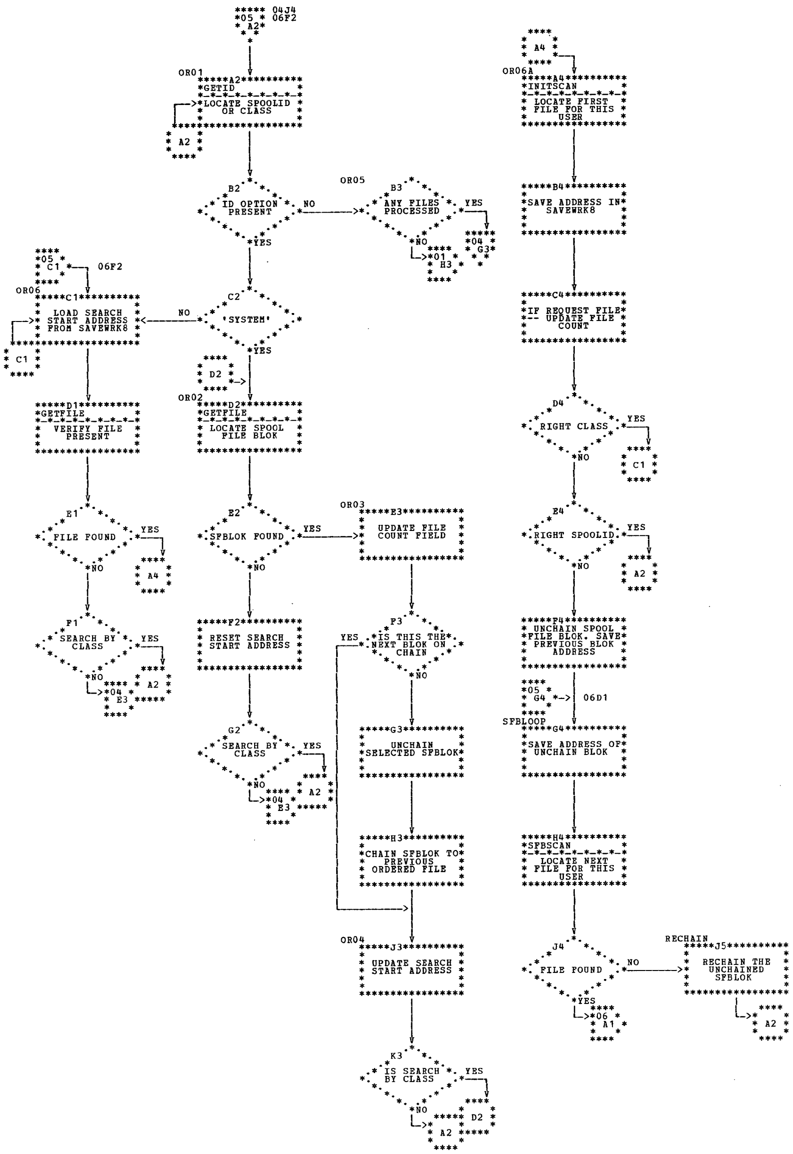


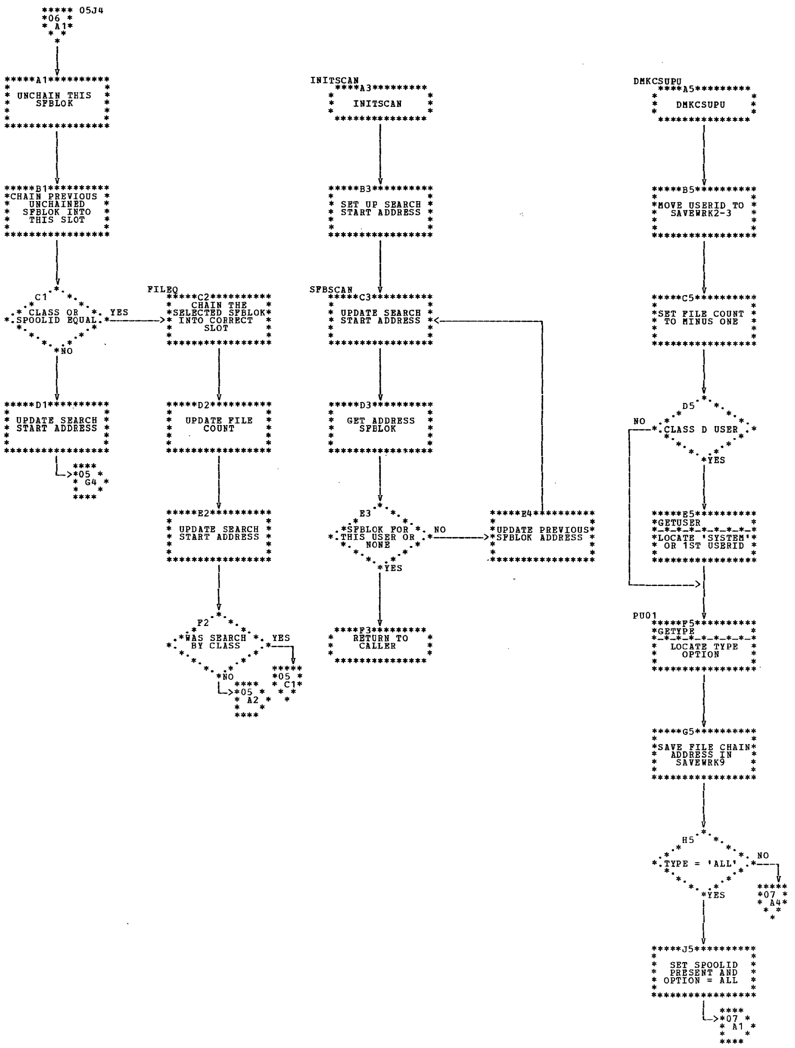


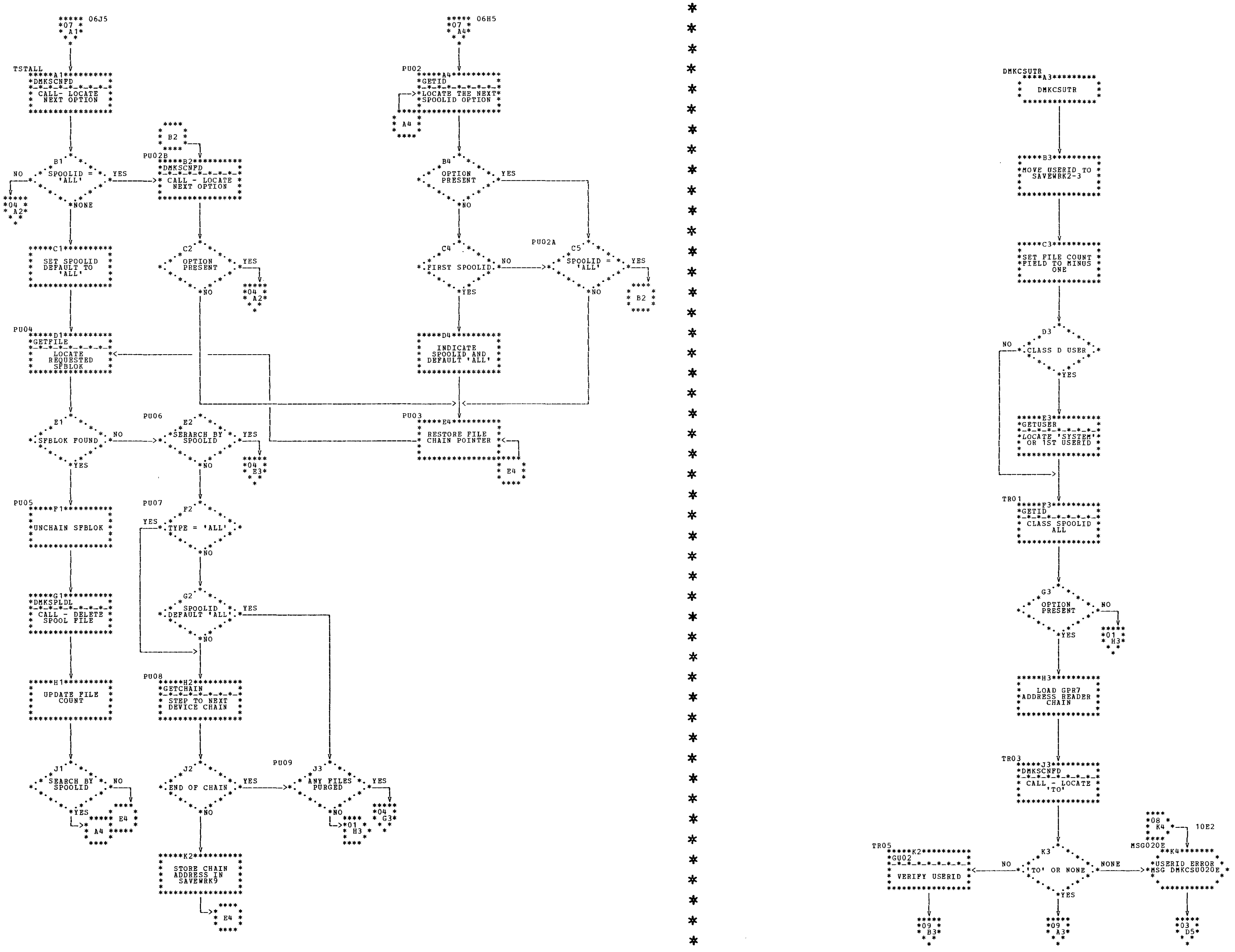


DMKCSU -- Process Spooling Commands (Parts 3 and 4 of 17)

DMKCSU -- Process Spooling Commands (Parts 5 and 6 of 17)

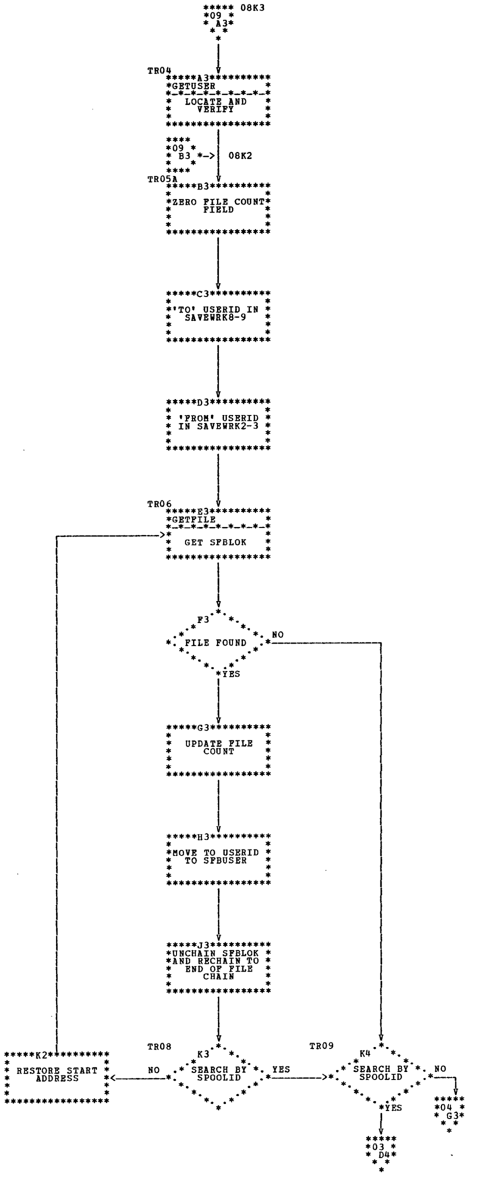


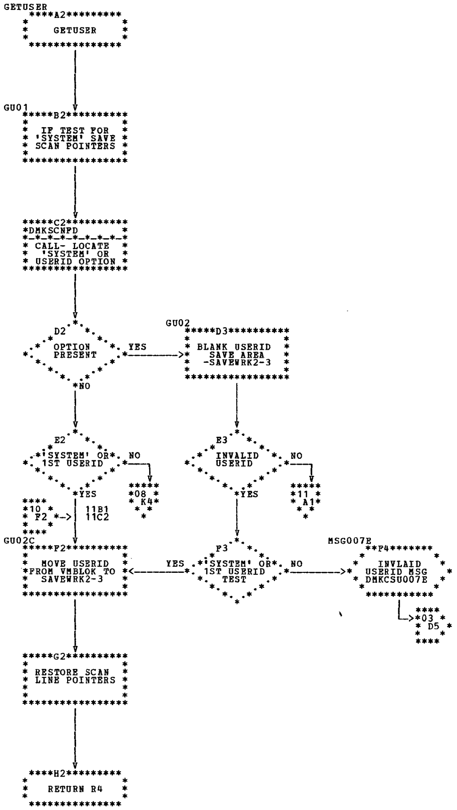

06 A1
05J4
A1
UNCHAIN THIS SFBLOK
INITSCAN
A3
INITSCAN
DMKCSUPU
A5
DMKCSUPU
B1
CHAIN PREVIOUS UNCHAINED SFBLOK INTO THIS SLOT
B3
SET UP SEARCH START ADDRESS
B5
MOVE USERID TO SAVEWRK2-3
C1
CLASS OR SPOOLID EQUAL
YES
NO
FILEQ
C2
CHAIN THE SELECTED SFBLOK INTO CORRECT SLOT
SFBSCAN
C3
UPDATE SEARCH START ADDRESS
C5
SET FILE COUNT TO MINUS ONE
D1
UPDATE SEARCH START ADDRESS
05 G4
D2
UPDATE FILE COUNT
D3
GET ADDRESS SFBLOK
D5
CLASS D USER
NO
YES
E2
UPDATE SEARCH START ADDRESS
E3
SFBLOK FOR THIS USER OR NONE
NO
YES
E4
UPDATE PREVIOUS SFBLOK ADDRESS
E5
GETUSER
LOCATE 'SYSTEM' OR 1ST USERID
F2
WAS SEARCH BY CLASS
YES
05 C1
NO
05 A2
F3
RETURN TO CALLER
PU01
F5
GETYPE
LOCATE TYPE OPTION
G5
SAVE FILE CHAIN ADDRESS IN SAVEWRK9
H5
TYPE = 'ALL'
NO
07 A4
YES
J5
SET SPOOLID PRESENT AND OPTION = ALL
07 A1

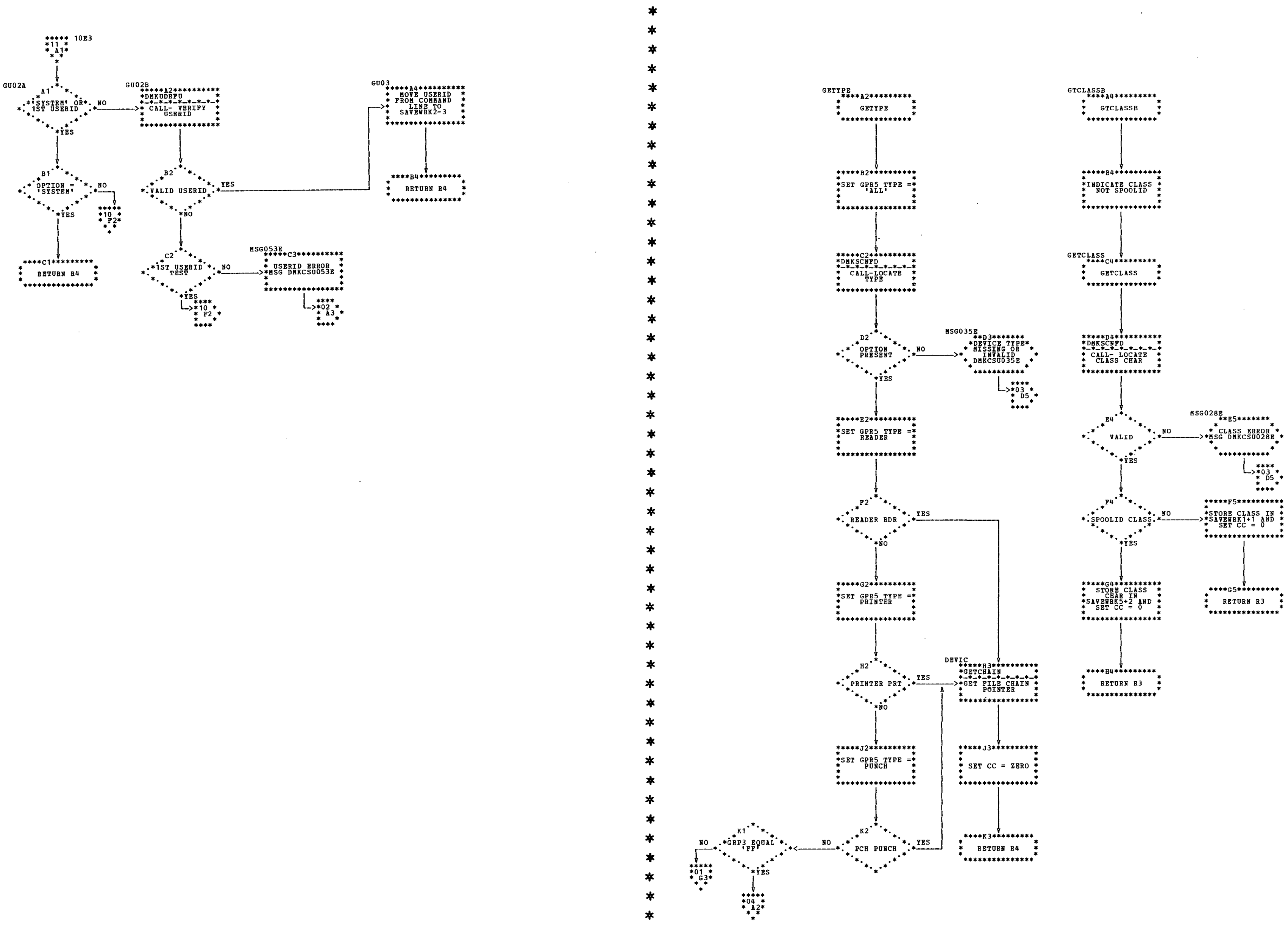


DMKCSU -- Process Spooling Commands (Parts 7 and 8 of 17)

DMKCSU -- Process Spooling Commands (Parts 9 and 10 cf 17)

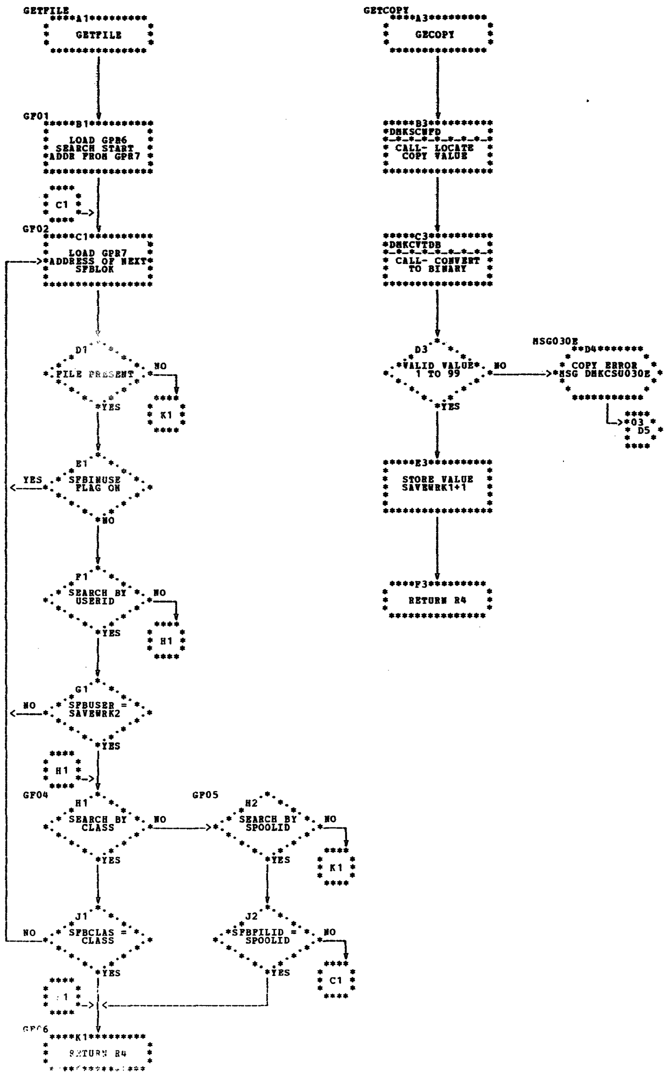


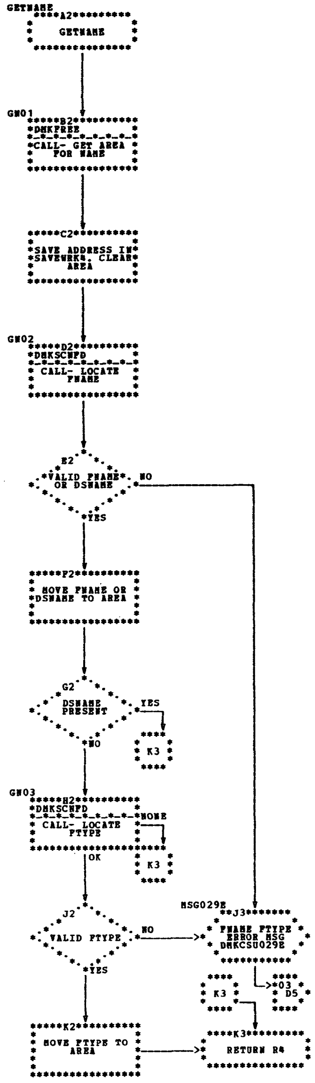

GETUSER
A2
GETUSER
GU01
B2
IF TEST FOR 'SYSTEM' SAVE SCAN POINTERS
C2
DMKSCNFD
CALL- LOCATE 'SYSTEM' OR USERID OPTION
D2
OPTION PRESENT
YES
NO
GU02
D3
BLANK USERID SAVE AREA -SAVEWRK2-3
E2
'SYSTEM' OR 1ST USERID
NO
YES
08 K4
E3
INVALID USERID
NO
YES
11 A1
10 F2
11B1
11C2
GU02C
F2
MOVE USERID FROM VMBLOK TO SAVEWRK2-3
F3
'SYSTEM' OR 1ST USERID TEST
YES
NO
MSG007E
F4
INVLAID USERID MSG DMKCSU007E
03 D5
G2
RESTORE SCAN LINE POINTERS
H2
RETURN R4

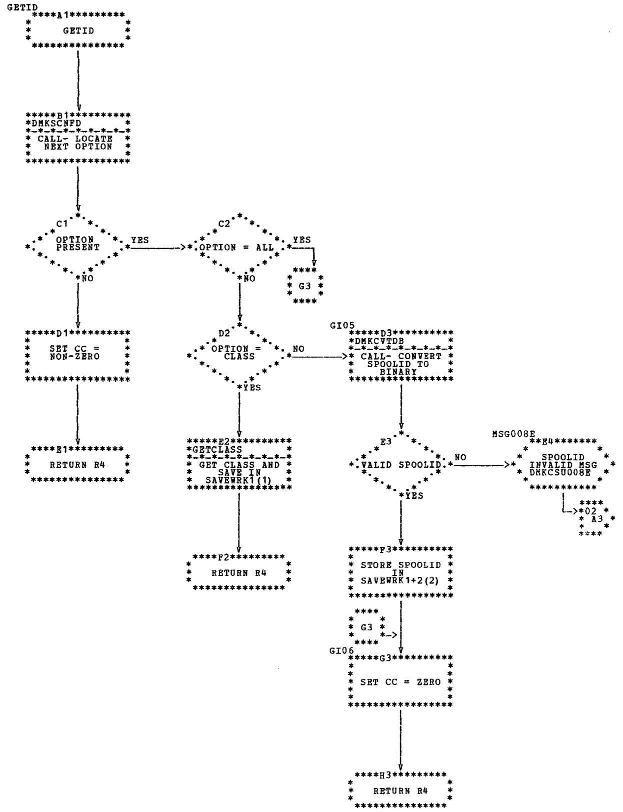


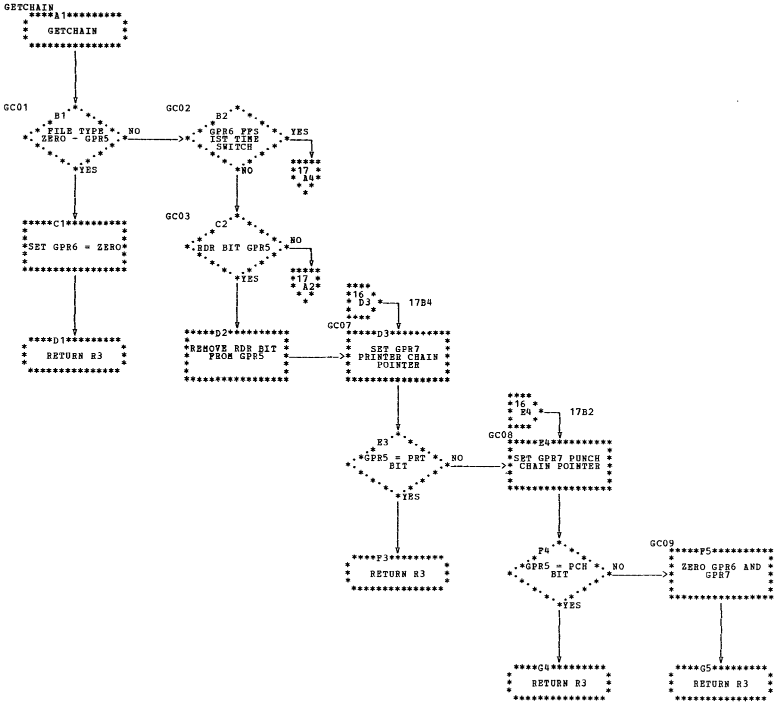
DMKCSU -- Process Spooling Commands (Parts 11 and 12 of 17)

DMKCSU -- Process Spooling Commands (Parts 13 and 14 of 17)



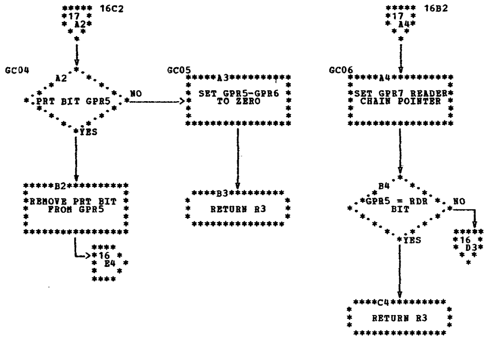


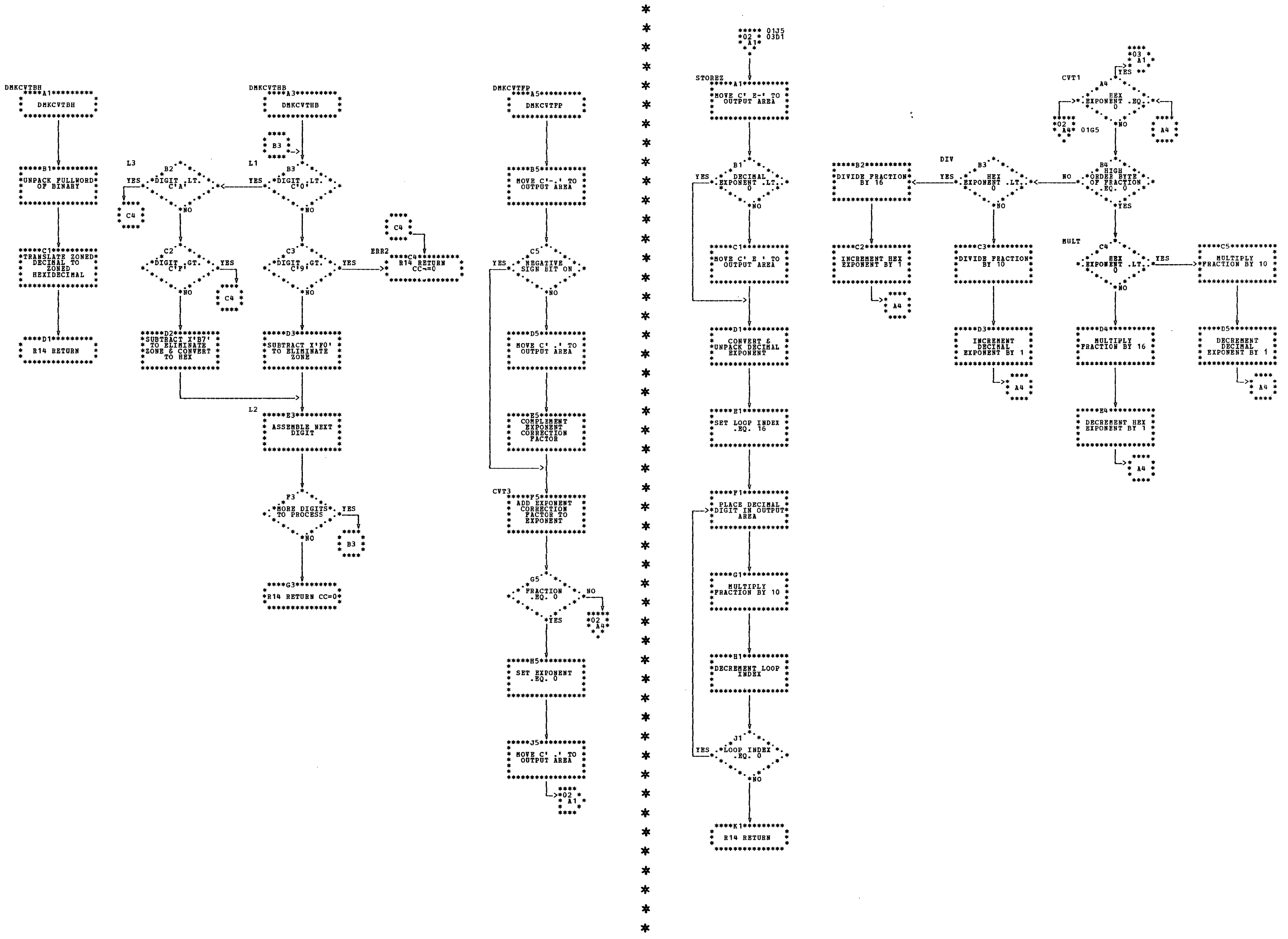




DMKCSU -- Process Spooling COommands (Parts 15 and 16 of 17)

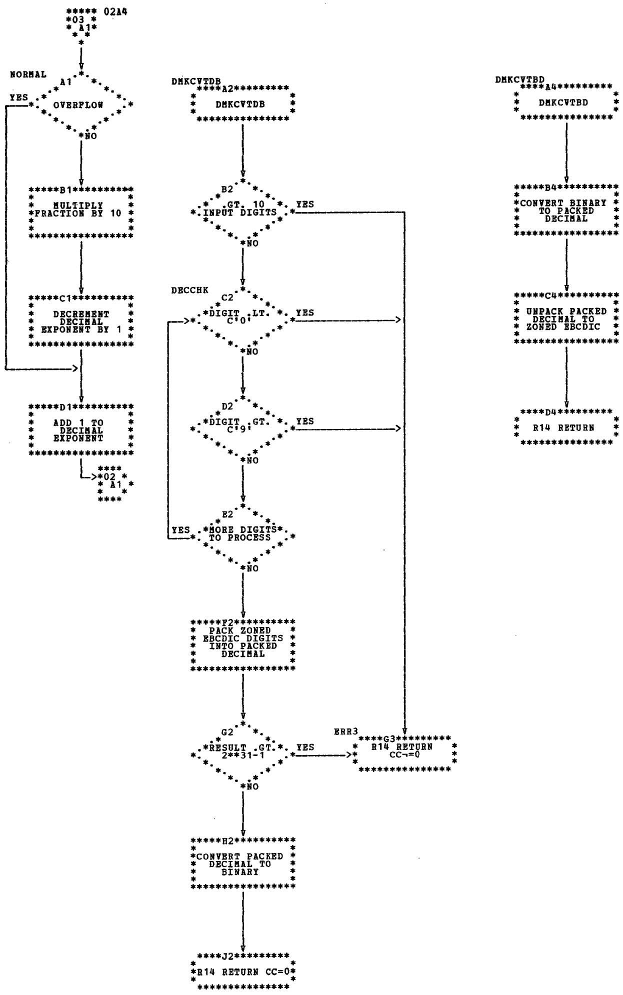
DMKCSU -- Process Spooling Commands (Part 17 of 17)

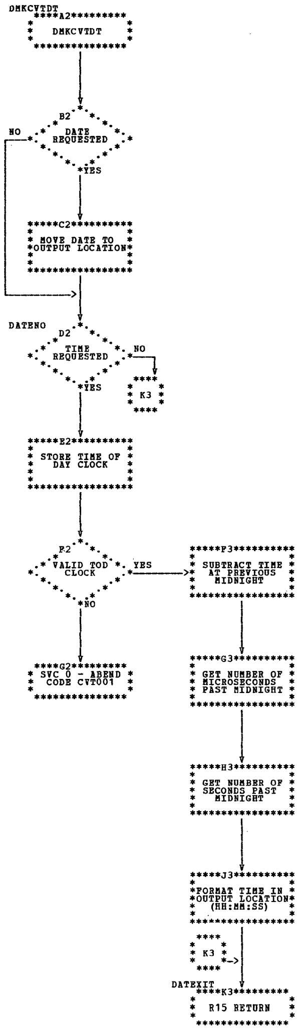


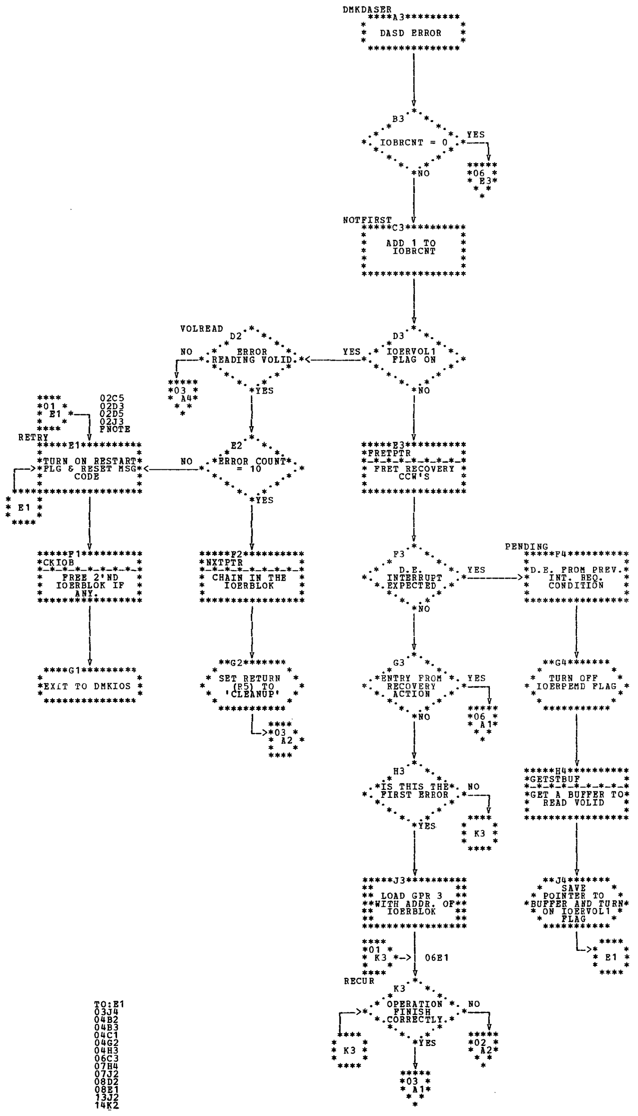


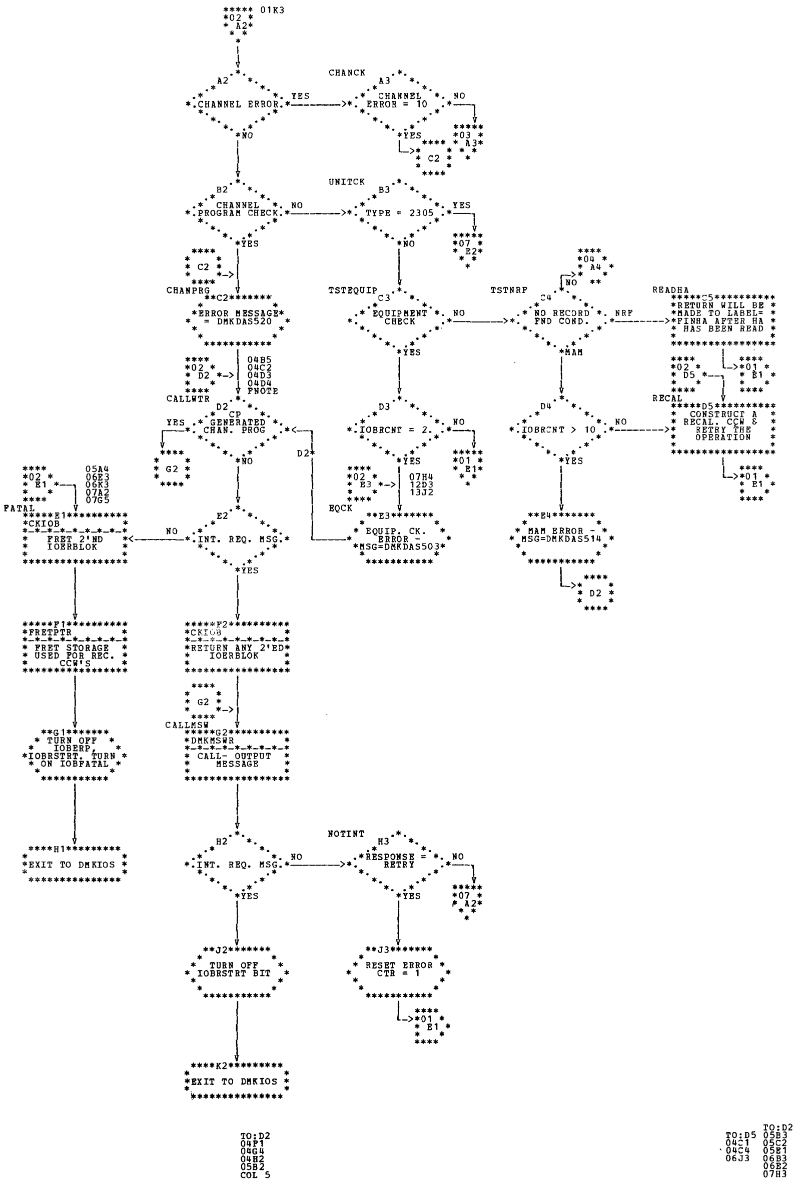
DMKCVT -- Conversion Routines (Parts 1 and 2 of 4)

DMKCVT -- Conversion Routines (Parts 3 and 4 of 4)


02A4
03 A1
NORMAL
A1 OVERFLOW
YES
NO
B1 MULTIPLY FRACTION BY 10
C1 DECREMENT DECIMAL EXPONENT BY 1
D1 ADD 1 TO DECIMAL EXPONENT
02 A1
DMKCVTDB
A2 DMKCVTDB
B2 .GT. 10 INPUT DIGITS
YES
NO
DECCHK
C2 DIGIT .LT. C'0'
YES
NO
D2 DIGIT .GT. C'9'
YES
NO
E2 MORE DIGITS TO PROCESS
YES
NO
F2 PACK ZONED EBCDIC DIGITS INTO PACKED DECIMAL
G2 RESULT .GT. 2**31-1
YES
NO
ERR3
G3 R14 RETURN CC¬=0
H2 CONVERT PACKED DECIMAL TO BINARY
J2 R14 RETURN CC=0
DMKCVTBD
A4 DMKCVTBD
B4 CONVERT BINARY TO PACKED DECIMAL
C4 UNPACK PACKED DECIMAL TO ZONED EBCDIC
D4 R14 RETURN

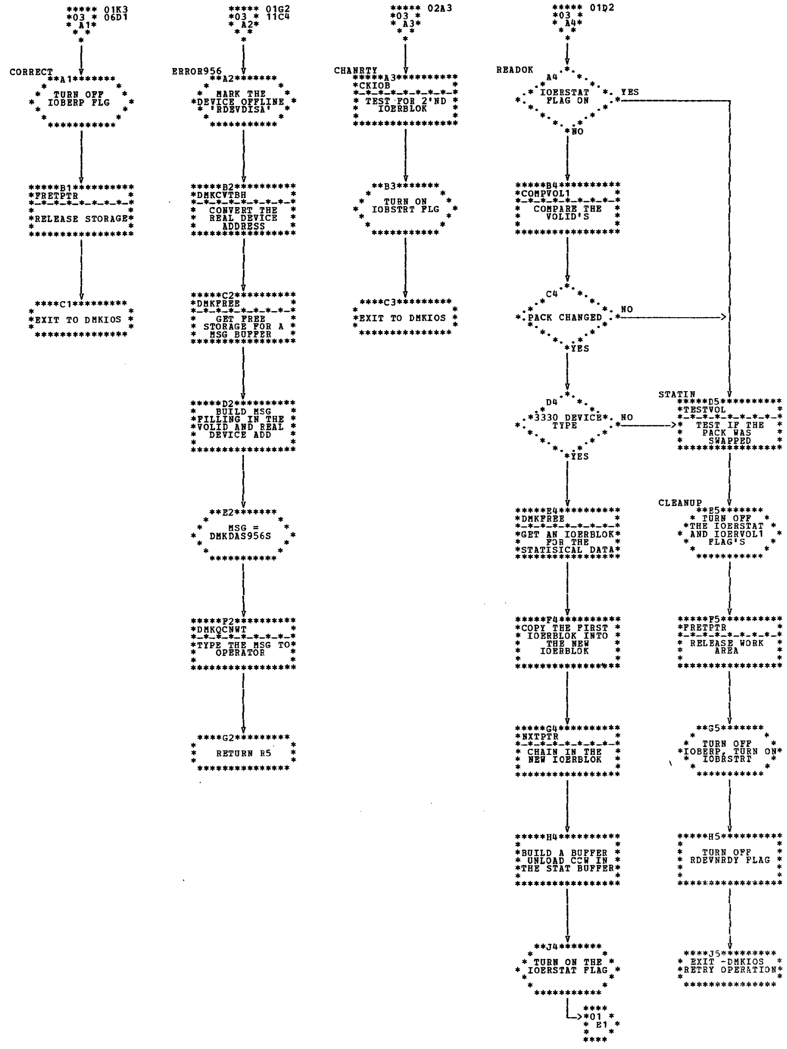

DMKCVTDT
A2 DMKCVTDT
B2 DATE REQUESTED
NO
YES
C2 MOVE DATE TO OUTPUT LOCATION
DATENO
D2 TIME REQUESTED
NO
K3
YES
E2 STORE TIME OF DAY CLOCK
F2 VALID TOD CLOCK
YES
NO
F3 SUBTRACT TIME AT PREVIOUS MIDNIGHT
G2 SVC 0 - ABEND CODE CVT001
G3 GET NUMBER OF MICROSECONDS PAST MIDNIGHT
H3 GET NUMBER OF SECONDS PAST MIDNIGHT
J3 FORMAT TIME IN OUTPUT LOCATION (HH:MM:SS)
K3
DATEXIT
K3 R15 RETURN

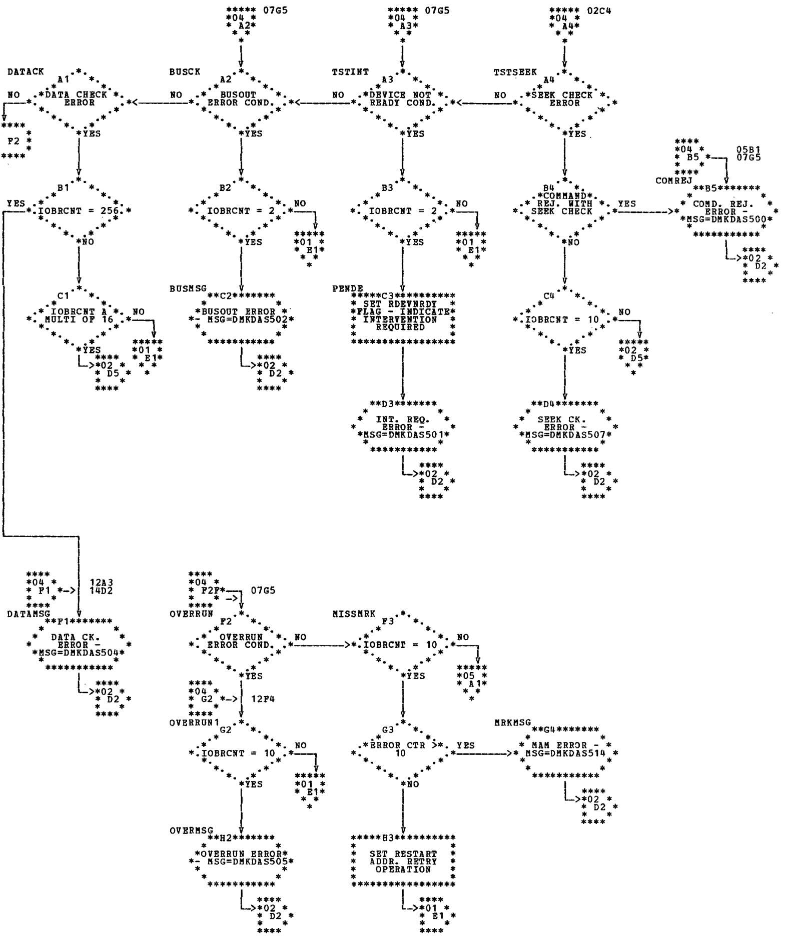


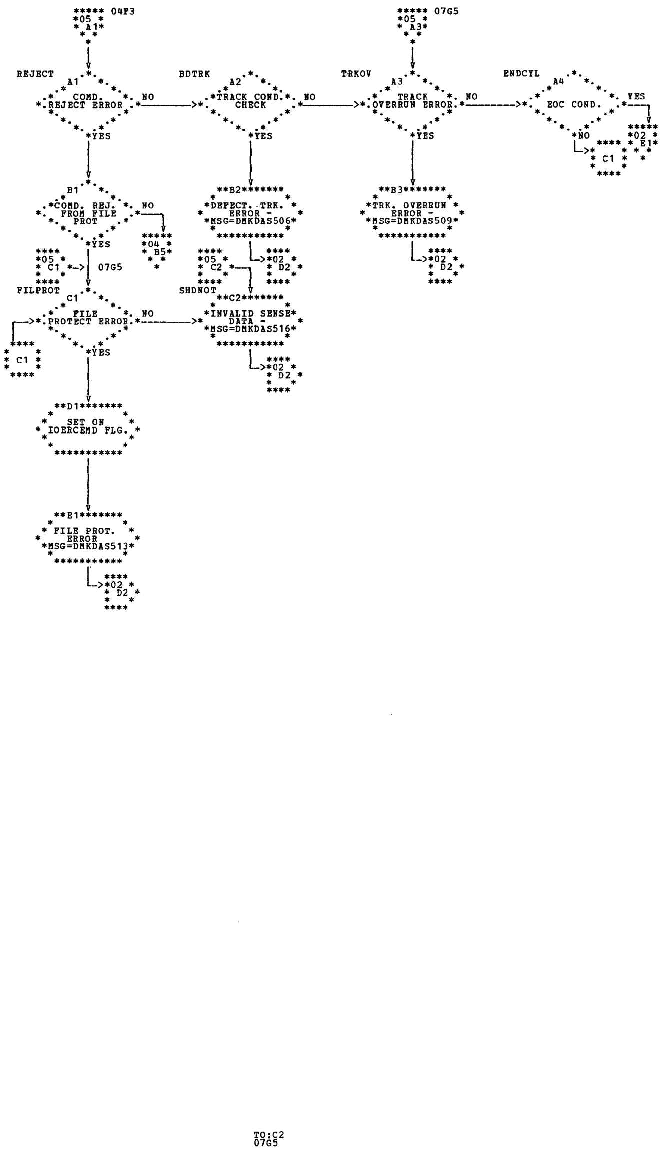


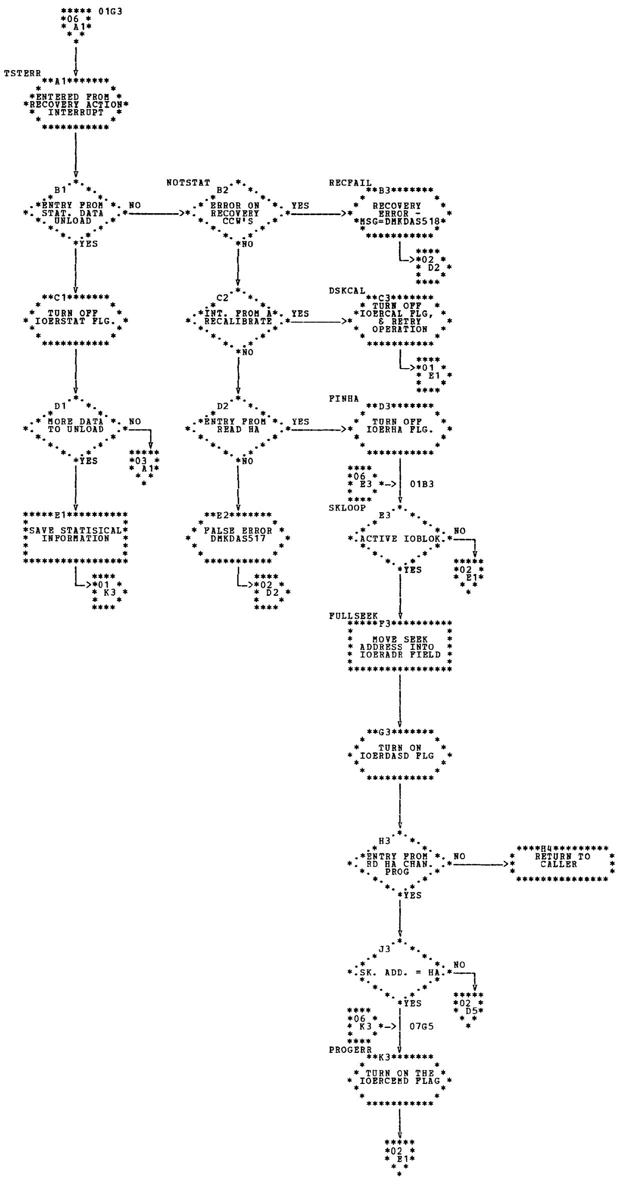
| DMKDAS -- DASD Error RECCVERY Procedures (Parts 1 and 2 of 14)

| DMKDAS -- DASD Error Reccvery Procedures (Parts 3 and 4 of 14)


03 A1 01K3 06D1
CORRECT
A1 TURN OFF IOBERP FLG
B1 FRETPTR RELEASE STORAGE
C1 EXIT TO DMKIOS
03 A2 01G2 11C4
ERROR956
A2 MARK THE DEVICE OFFLINE 'RDEVDISA'
B2 DMKCVTBH CONVERT THE REAL DEVICE ADDRESS
C2 DMKFREE GET FREE STORAGE FOR A MSG BUFFER
D2 BUILD MSG FILLING IN THE VOLID AND REAL DEVICE ADD
E2 MSG = DMKDAS956S
F2 DMKQCNWT TYPE THE MSG TO OPERATOR
G2 RETURN R5
03 A3 02A3
CHANRTY
A3 CKIOB TEST FOR 2'ND IOERBLOK
B3 TURN ON IOBSTRT FLG
C3 EXIT TO DMKIOS
03 A4 01D2
READOK
A4 IOERSTAT FLAG ON
YES
NO
B4 COMPVOL1 COMPARE THE VOLID'S
C4 PACK CHANGED
NO
YES
D4 3330 DEVICE TYPE
NO
YES
E4 DMKFREE GET AN IOERBLOK FOR THE STATISICAL DATA
F4 COPY THE FIRST IOERBLOK INTO THE NEW IOERBLOK
G4 NXTPTR CHAIN IN THE NEW IOERBLOK
H4 BUILD A BUFFER UNLOAD CCW IN THE STAT BUFFER
J4 TURN ON THE IOERSTAT FLAG
01 E1
STATIN
D5 TESTVOL TEST IF THE PACK WAS SWAPPED
CLEANUP
E5 TURN OFF THE IOERSTAT AND IOERVOL1 FLAG'S
F5 FRETPTR RELEASE WORK AREA
G5 TURN OFF IOBERP, TURN ON IOBRSTRT
H5 TURN OFF RDEVNRDY FLAG
J5 EXIT -DMKIOS RETRY OPERATION

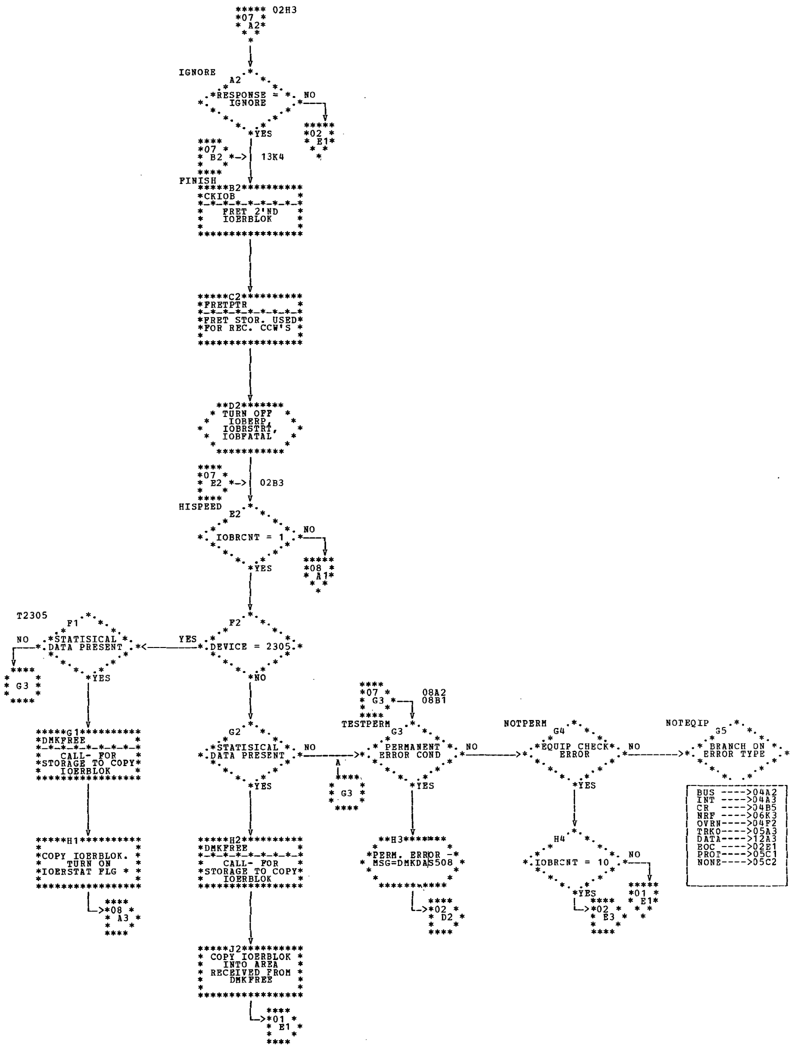




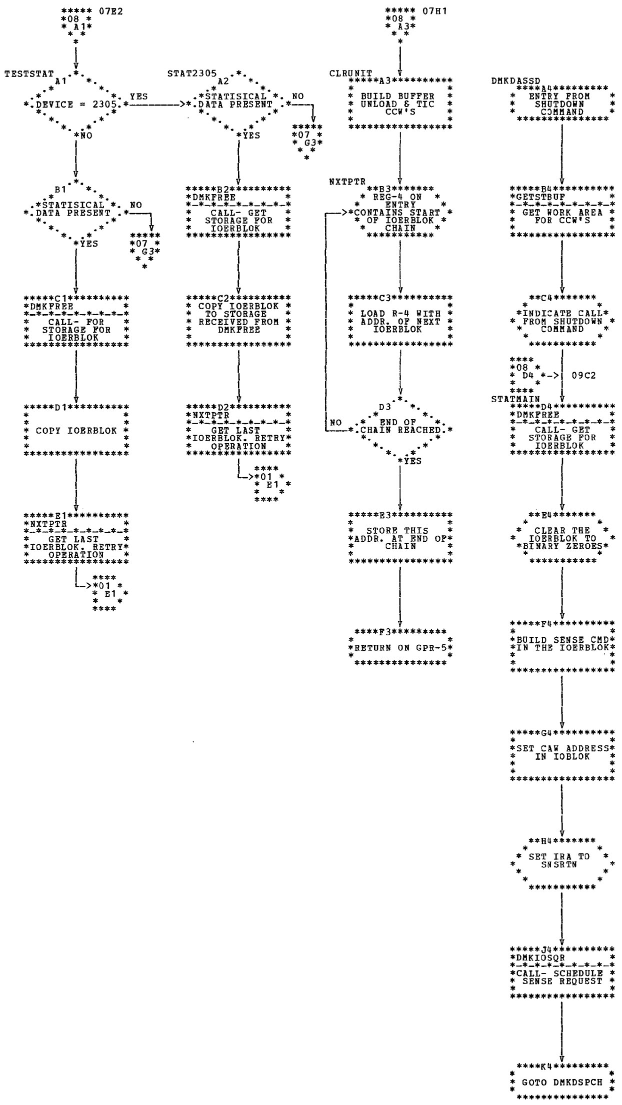


| DMKDAS -- DASD Error Recovery Procedures (Parts 5 and 6 of 14)

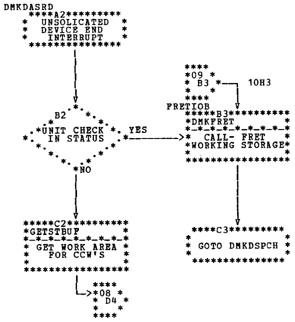
| DMKDAS -- DASD Error Reccvery Procedures (Parts 7 and 8 of 14)

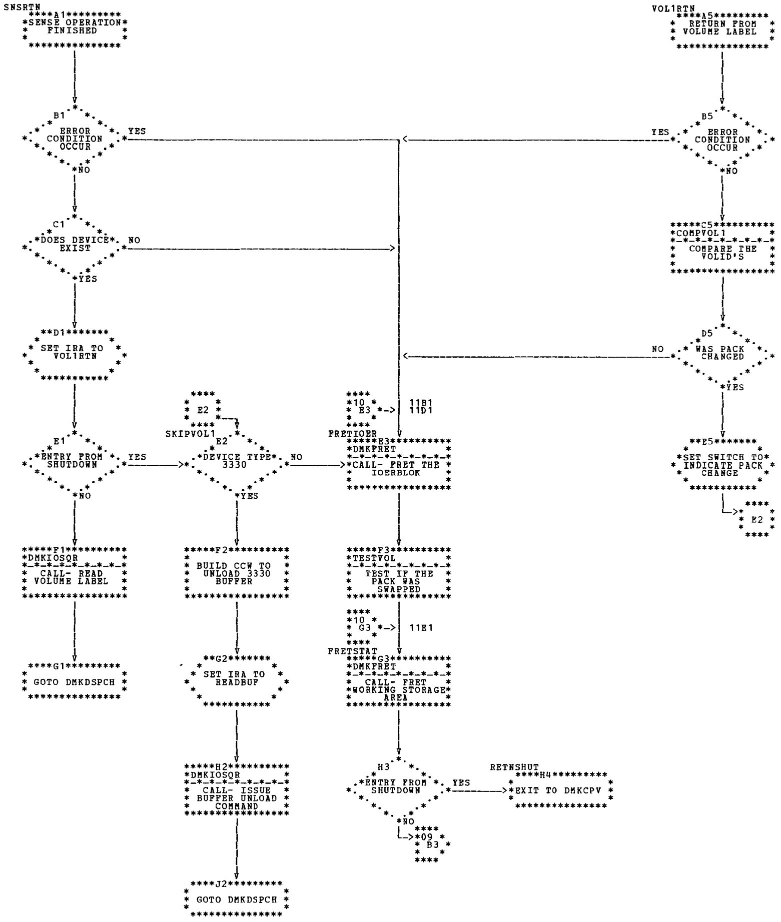


07 A2 02H3
IGNORE A2 RESPONSE = IGNORE
NO 02 E1
YES
07 B2 13K4
FINISH B2 CKIOB FRET 2'ND IOERBLOK
C2 FRETPTR FRET STOR. USED FOR REC. CCW'S
D2 TURN OFF IOBERP, IOBRSTRT, IOBFATAL
07 E2 02B3
HISPEED E2 IOBRCNT = 1
NO 08 A1
YES
T2305 F1 STATISICAL DATA PRESENT
NO G3
YES
F2 DEVICE = 2305
YES
NO
G1 DMKFREE CALL- FOR STORAGE TO COPY IOERBLOK
H1 COPY IOERBLOK. TURN ON IOERSTAT FLG
08 A3
G2 STATISICAL DATA PRESENT
NO
YES
H2 DMKFREE CALL- FOR STORAGE TO COPY IOERBLOK
J2 COPY IOERBLOK INTO AREA RECEIVED FROM DMKFREE
01 E1
07 G3 08A2 08B1
A
G3
TESTPERM G3 PERMANENT ERROR COND
NO
YES
H3 PERM. ERROR - MSG=DMKDAS508
02 D2
NOTPERM G4 EQUIP CHECK ERROR
NO
YES
H4 IOBRCNT = 10
NO 01 E1
YES 02 E3
NOTEQIP G5 BRANCH ON ERROR TYPE
BUS ---->04A2
INT ---->04A3
CR ---->04B5
NRF ---->06K3
OVRN---->04F2
TRKO---->05A3
DATA---->12A3
EOC ---->02E1
PROT---->05C1
NONE---->05C2



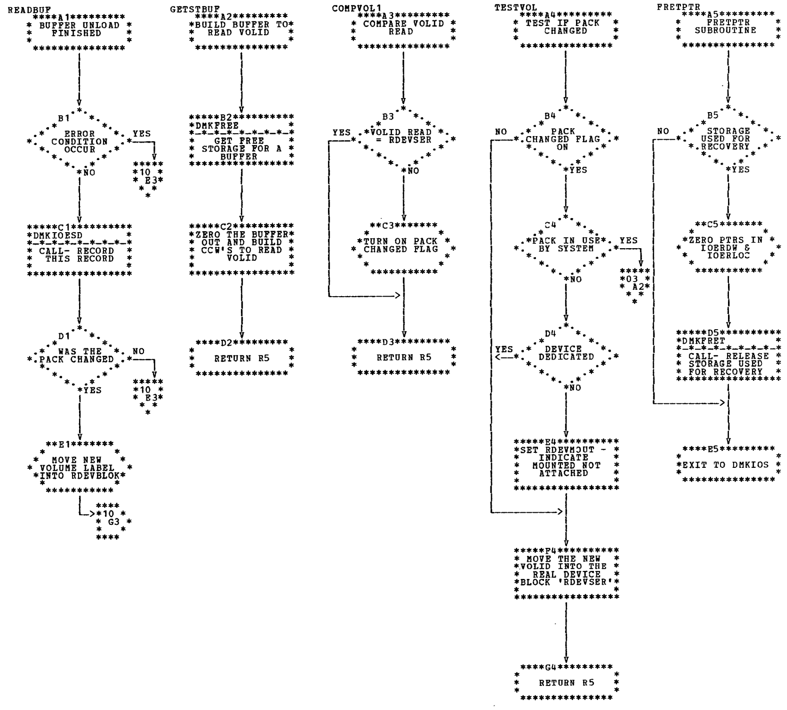
08 A1 07E2
TESTSTAT A1 DEVICE = 2305
YES
NO
B1 STATISICAL DATA PRESENT
NO 07 G3
YES
C1 DMKFREE CALL- FOR STORAGE FOR IOERBLOK
D1 COPY IOERBLOK
E1 NXTPTR GET LAST IOERBLOK. RETRY OPERATION
01 E1
STAT2305 A2 STATISICAL DATA PRESENT
NO 07 G3
YES
B2 DMKFREE CALL- GET STORAGE FOR IOERBLOK
C2 COPY IOERBLOK TO STORAGE RECEIVED FROM DMKFREE
D2 NXTPTR GET LAST IOERBLOK. RETRY OPERATION
01 E1
08 A3 07H1
CLRUNIT A3 BUILD BUFFER UNLOAD & TIC CCW'S
NXTPTR B3 REG-4 ON ENTRY CONTAINS START OF IOERBLOK CHAIN
C3 LOAD R-4 WITH ADDR. OF NEXT IOERBLOK
D3 END OF CHAIN REACHED
NO
YES
E3 STORE THIS ADDR. AT END OF CHAIN
F3 RETURN ON GPR-5
DMKDASSD A4 ENTRY FROM SHUTDOWN COMMAND
B4 GETSTBUF GET WORK AREA FOR CCW'S
C4 INDICATE CALL FROM SHUTDOWN COMMAND
08 D4 09C2
STATMAIN D4 DMKFREE CALL- GET STORAGE FOR IOERBLOK
E4 CLEAR THE IOERBLOK TO BINARY ZEROES
F4 BUILD SENSE CMD IN THE IOERBLOK
G4 SET CAW ADDRESS IN IOBLOK
H4 SET IRA TO SNSRTN
J4 DMKIOSQR CALL- SCHEDULE SENSE REQUEST
K4 GOTO DMKDSPCH

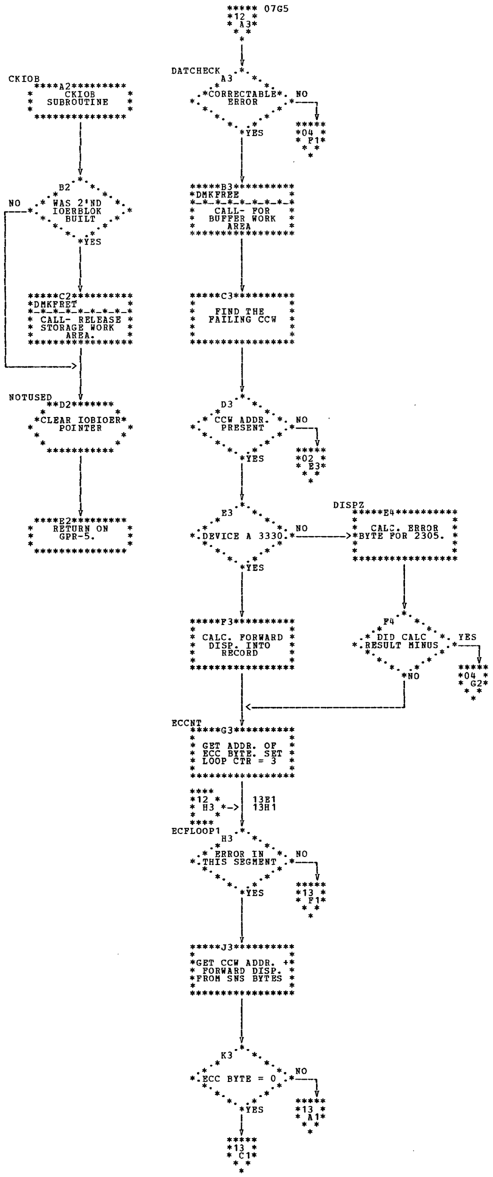


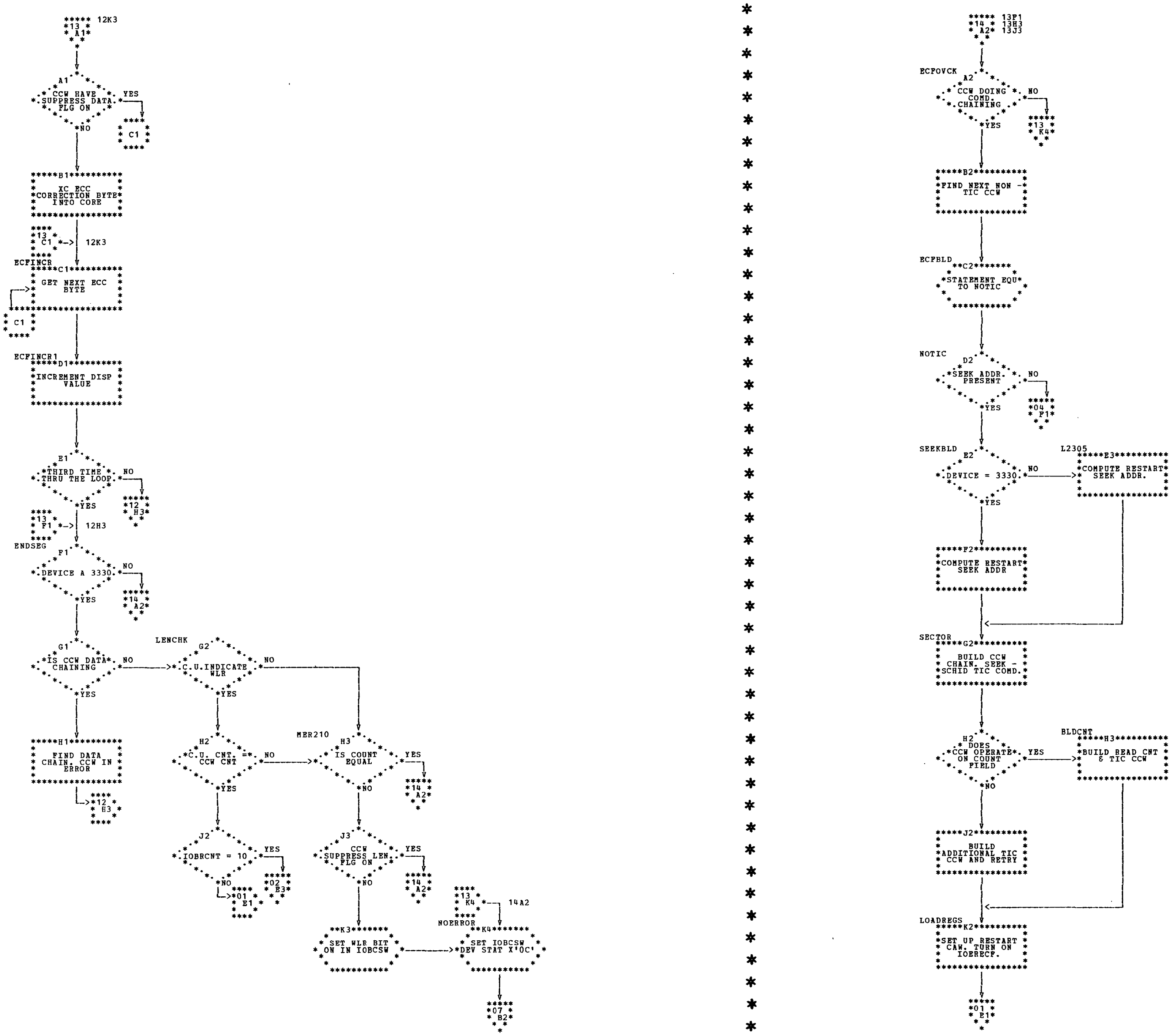


| DMKDAS -- DASD Error Recovery Procedures (Parts 9 and 10 of 14)

| DMKDAS -- DASD Error Reccvery Procedures (Parts 11 and 12 of 14)

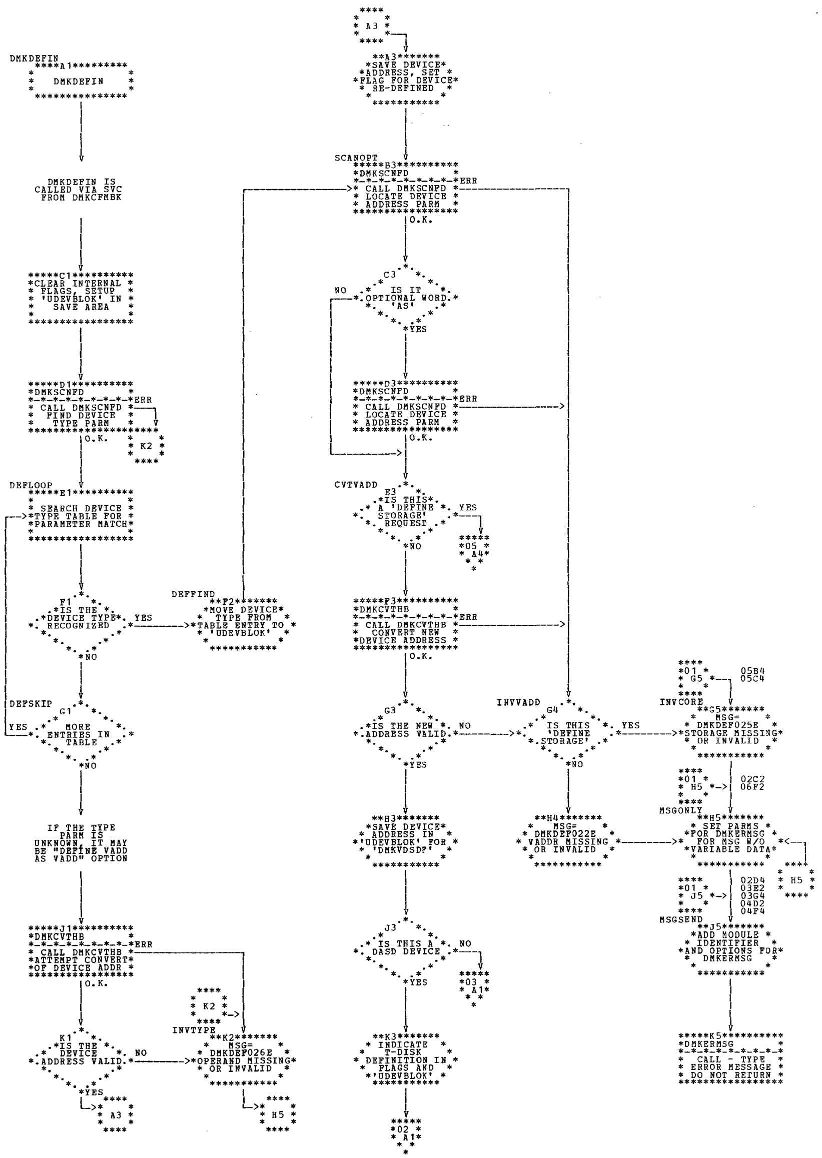

READBUF
A1 BUFFER UNLOAD FINISHED
B1 ERROR CONDITION OCCUR
YES
10 E3
NO
C1 DMKIOESD
CALL- RECORD THIS RECORD
D1 WAS THE PACK CHANGED
NO
10 E3
YES
E1 MOVE NEW VOLUME LABEL INTO RDEVBLOK
10 G3
GETSTBUF
A2 BUILD BUFFER TO READ VOLID
B2 DMKFREE
GET FREE STORAGE FOR A BUFFER
C2 ZERO THE BUFFER OUT AND BUILD CCW'S TO READ VOLID
D2 RETURN R5
COMPVOL1
A3 COMPARE VOLID READ
B3 VOLID READ = RDEVSER
YES
NO
C3 TURN ON PACK CHANGED FLAG
D3 RETURN R5
TESTVOL
A4 TEST IF PACK CHANGED
B4 PACK CHANGED FLAG ON
NO
YES
C4 PACK IN USE BY SYSTEM
YES
03 A2
NO
D4 DEVICE DEDICATED
YES
NO
E4 SET RDEVMOUT - INDICATE MOUNTED NOT ATTACHED
F4 MOVE THE NEW VOLID INTO THE REAL DEVICE BLOCK 'RDEVSER'
G4 RETURN R5
FRETPTR
A5 FRETPTR SUBROUTINE
B5 STORAGE USED FOR RECOVERY
NO
YES
C5 ZERO PTRS IN IOERDW & IOERLOC
D5 DMKFRET
CALL- RELEASE STORAGE USED FOR RECOVERY
E5 EXIT TO DMKIOS

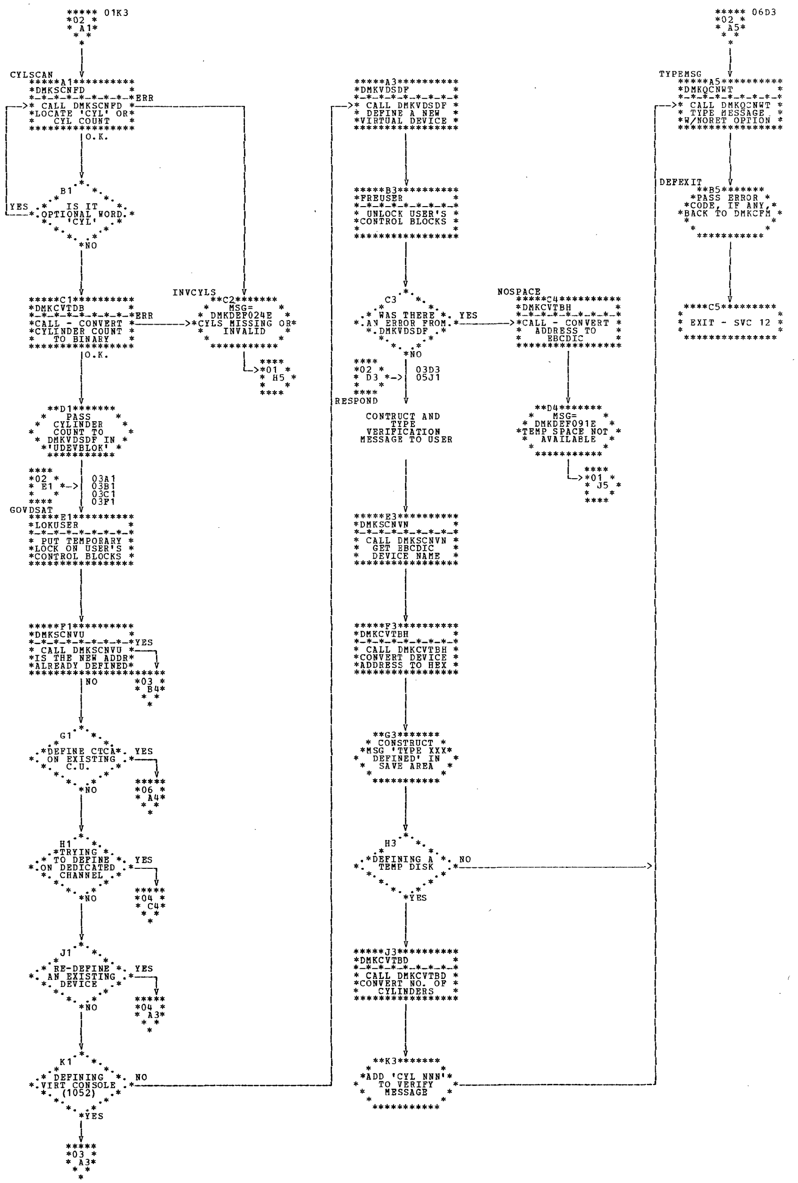


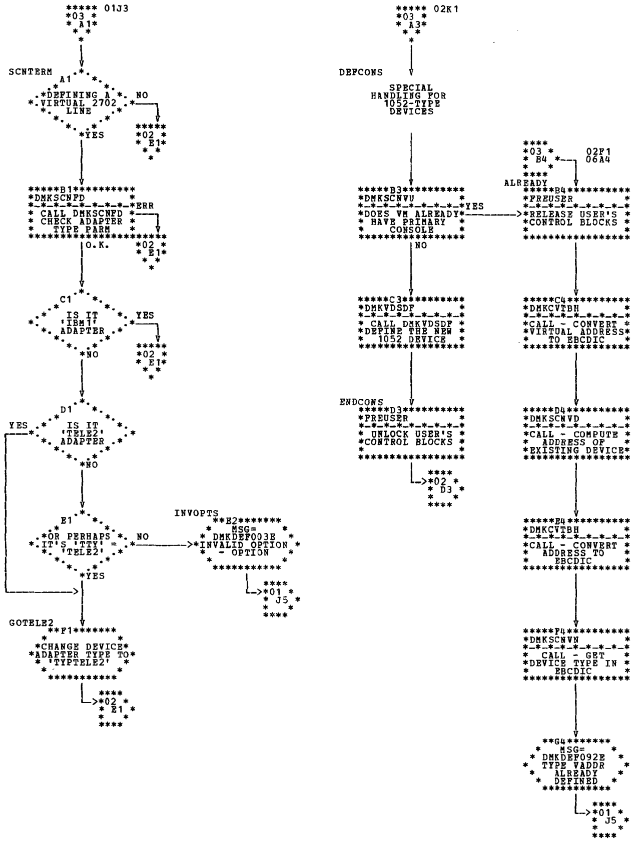

13 A1 12K3
A1 CCW HAVE SUPPRESS DATA. FLG ON
YES
NO
C1
B1 XC ECC CORRECTION BYTE INTO CORE
13 C1 12K3
ECFINCR
C1 GET NEXT ECC BYTE
C1
ECFINCR1
D1 INCREMENT DISP VALUE
E1 THIRD TIME THRU THE LOOP
NO
12 H3
YES
13 F1 12H3
ENDSEG
F1 DEVICE A 3330
NO
14 A2
YES
G1 IS CCW DATA CHAINING
NO
YES
H1 FIND DATA CHAIN. CCW IN ERROR
12 H3
LENCHK
G2 C.U.INDICATE WLR
NO
YES
H2 C.U. CNT. = CCW CNT
NO
YES
J2 IOBRCNT = 10
YES
02 B3
NO
01 E1
MER210
H3 IS COUNT EQUAL
YES
14 A2
NO
J3 CCW SUPPRESS LEN. FLG ON
YES
14 A2
NO
K3 SET WLR BIT ON IN IOBCSW
13 K4 14A2
NOERROR
K4 SET IOBCSW DEV STAT X'0C'
07 B2
14 A2 13F1 13H3 13J3
ECFOVCK
A2 CCW DOING COMD. CHAINING
NO
13 K4
YES
B2 FIND NEXT NON - TIC CCW
ECFBLD
C2 STATEMENT EQU TO NOTIC
NOTIC
D2 SEEK ADDR. PRESENT
NO
04 F1
YES
SEEKBLD
E2 DEVICE = 3330
NO
L2305
E3 COMPUTE RESTART SEEK ADDR.
YES
F2 COMPUTE RESTART SEEK ADDR
SECTOR
G2 BUILD CCW CHAIN. SEEK - SCHID TIC COMD.
H2 DOES CCW OPERATE ON COUNT FIELD
YES
BLDCNT
H3 BUILD READ CNT & TIC CCW
NO
J2 BUILD ADDITIONAL TIC CCW AND RETRY
LOADREGS
K2 SET UP RESTART CAW. TURN ON IOERECF.
01 E1

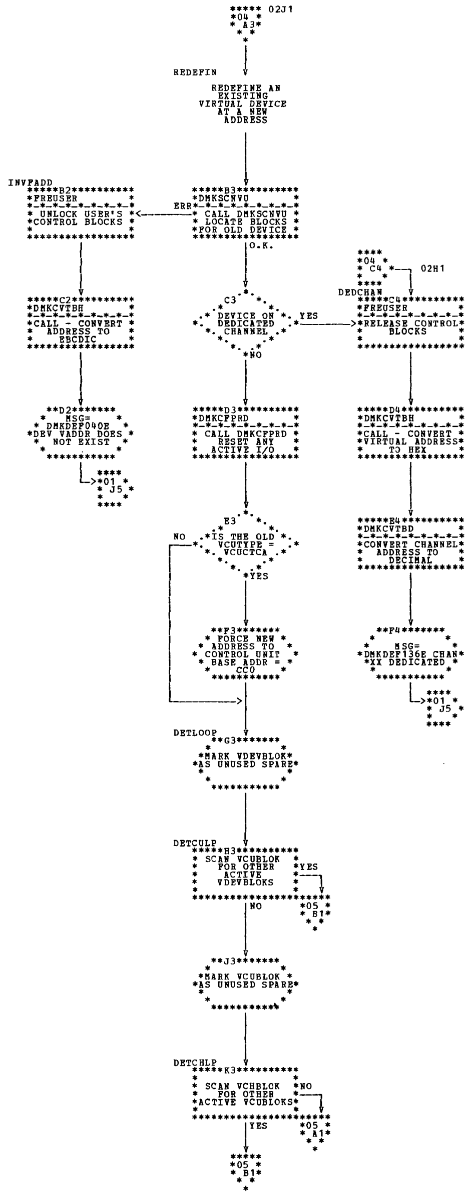
| DMKDAS -- DASD Error Recovery Procedures (Parts 13 and 14 of 14)

| DMKDEF -- Process DEFINE Command; Define Virtual Storage or a Device (Parts 1 and 2 of 7)


DMKDEFIN
A1
DMKDEFIN
DMKDEFIN IS CALLED VIA SVC FROM DMKCFMBK
C1
CLEAR INTERNAL FLAGS, SETUP 'UDEVBLOK' IN SAVE AREA
D1
DMKSCNFD
CALL DMKSCNFD FIND DEVICE TYPE PARM
ERR
O.K.
K2
DEFLOOP
E1
SEARCH DEVICE TYPE TABLE FOR PARAMETER MATCH
F1
IS THE DEVICE TYPE RECOGNIZED
YES
NO
DEFFIND
F2
MOVE DEVICE TYPE FROM TABLE ENTRY TO 'UDEVBLOK'
DEFSKIP
G1
MORE ENTRIES IN TABLE
YES
NO
IF THE TYPE PARM IS UNKNOWN, IT MAY BE "DEFINE VADD AS VADD" OPTION
J1
DMKCVTHB
CALL DMKCVTHB ATTEMPT CONVERT OF DEVICE ADDR
ERR
O.K.
K1
IS THE DEVICE ADDRESS VALID
NO
YES
A3
K2
INVTYPE
K2
MSG= DMKDEF026E OPERAND MISSING OR INVALID
H5
A3
A3
SAVE DEVICE ADDRESS, SET FLAG FOR DEVICE RE-DEFINED
SCANOPT
B3
DMKSCNFD
CALL DMKSCNFD LOCATE DEVICE ADDRESS PARM
ERR
O.K.
C3
IS IT OPTIONAL WORD 'AS'
NO
YES
D3
DMKSCNFD
CALL DMKSCNFD LOCATE DEVICE ADDRESS PARM
ERR
O.K.
CVTVADD
E3
IS THIS A 'DEFINE STORAGE' REQUEST
YES
05 A4
NO
F3
DMKCVTHB
CALL DMKCVTHB CONVERT NEW DEVICE ADDRESS
ERR
O.K.
G3
IS THE NEW ADDRESS VALID
NO
YES
H3
SAVE DEVICE ADDRESS IN 'UDEVBLOK' FOR 'DMKVDSDF'
J3
IS THIS A DASD DEVICE
NO
03 A1
YES
K3
INDICATE T-DISK DEFINITION IN FLAGS AND 'UDEVBLOK'
02 A1
INVVADD
G4
IS THIS 'DEFINE STORAGE'
YES
NO
H4
MSG= DMKDEF022E VADDR MISSING OR INVALID
01 G5
05B4
05C4
INVCORE
G5
MSG= DMKDEF025E STORAGE MISSING OR INVALID
01 H5
02C2
06F2
MSGONLY
H5
SET PARMS FOR DMKERMSG FOR MSG W/O VARIABLE DATA
H5
01 J5
02D4
03E2
03G4
04D2
04F4
MSGSEND
J5
ADD MODULE IDENTIFIER AND OPTIONS FOR DMKERMSG
K5
DMKERMSG
CALL - TYPE ERROR MESSAGE DO NOT RETURN

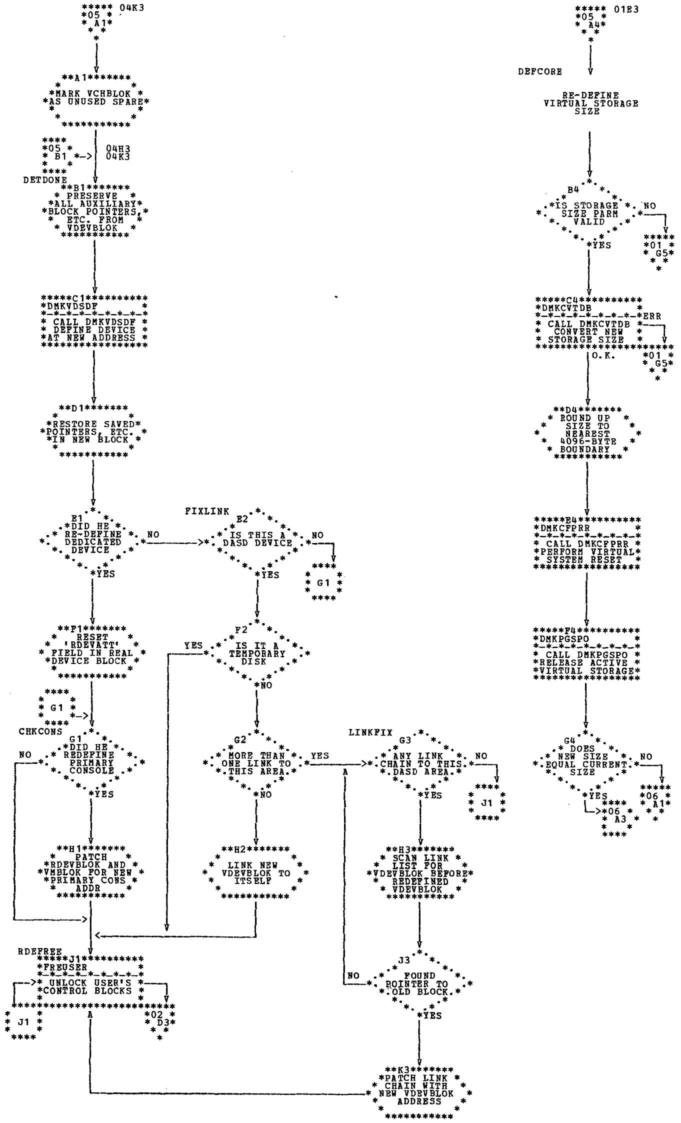

02 A1
01K3
CYLSCAN
A1
DMKSCNFD
CALL DMKSCNFD LOCATE 'CYL' OR CYL COUNT
ERR
O.K.
B1
IS IT OPTIONAL WORD 'CYL'
YES
NO
C1
DMKCVTDB
CALL - CONVERT CYLINDER COUNT TO BINARY
ERR
O.K.
INVCYLS
C2
MSG= DMKDEF024E CYLS MISSING OR INVALID
01 H5
D1
PASS CYLINDER COUNT TO DMKVDSDF IN 'UDEVBLOK'
02 E1
03A1
03B1
03C1
03F1
GOVDSAT
E1
LOKUSER
PUT TEMPORARY LOCK ON USER'S CONTROL BLOCKS
F1
DMKSCNVU
CALL DMKSCNVU IS THE NEW ADDR ALREADY DEFINED
YES
03 B4
NO
G1
DEFINE CTCA ON EXISTING C.U.
YES
06 A4
NO
H1
TRYING TO DEFINE ON DEDICATED CHANNEL
YES
04 C4
NO
J1
RE-DEFINE AN EXISTING DEVICE
YES
04 A3
NO
K1
DEFINING VIRT CONSOLE (1052)
NO
YES
03 A3
A3
DMKVDSDF
CALL DMKVDSDF DEFINE A NEW VIRTUAL DEVICE
B3
FREUSER
UNLOCK USER'S CONTROL BLOCKS
C3
WAS THERE AN ERROR FROM DMKVDSDF
YES
NO
02 D3
03D3
05J1
RESPOND
CONTRUCT AND TYPE VERIFICATION MESSAGE TO USER
E3
DMKSCNVN
CALL DMKSCNVN GET EBCDIC DEVICE NAME
F3
DMKCVTBH
CALL DMKCVTBH CONVERT DEVICE ADDRESS TO HEX
G3
CONSTRUCT MSG 'TYPE XXX DEFINED' IN SAVE AREA
H3
DEFINING A TEMP DISK
NO
YES
J3
DMKCVTBD
CALL DMKCVTBD CONVERT NO. OF CYLINDERS
K3
ADD 'CYL NNN' TO VERIFY MESSAGE
NOSPACE
C4
DMKCVTBH
CALL - CONVERT ADDRESS TO EBCDIC
D4
MSG= DMKDEF091E TEMP SPACE NOT AVAILABLE
01 J5
02 A5
06D3
TYPEMSG
A5
DMKQCNWT
CALL DMKQCNWT TYPE MESSAGE W/NORET OPTION
DEFEXIT
B5
PASS ERROR CODE, IF ANY, BACK TO DMKCFM
C5
EXIT - SVC 12

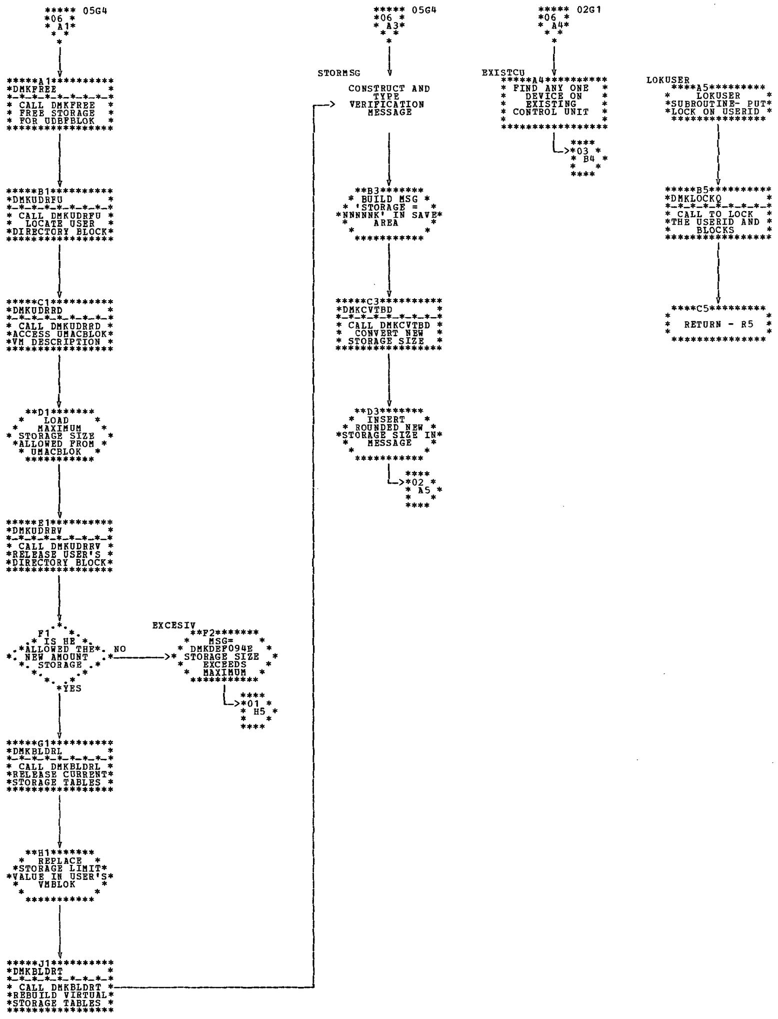

01J3
03 A1
SCNTERM
A1
DEFINING A VIRTUAL 2702 LINE
NO
02 E1
YES
B1
DMKSCNFD
CALL DMKSCNFD CHECK ADAPTER TYPE PARM
ERR
02 E1
O.K.
C1
IS IT 'IBM1' ADAPTER
YES
02 E1
NO
D1
IS IT 'TELE2' ADAPTER
YES
NO
E1
OR PERHAPS IT'S 'TTY' = 'TELE2'
NO
INVOPTS
E2
MSG= DMKDEF003E INVALID OPTION - OPTION
01 J5
YES
GOTELE2
F1
CHANGE DEVICE ADAPTER TYPE TO 'TYPTELE2'
02 E1
02K1
03 A3
DEFCONS
SPECIAL HANDLING FOR 1052-TYPE DEVICES
B3
DMKSCNVU
DOES VM ALREADY HAVE PRIMARY CONSOLE
YES
NO
03 B4
02F1
06A4
ALREADY
B4
FREUSER
RELEASE USER'S CONTROL BLOCKS
C3
DMKVDSDF
CALL DMKVDSDF DEFINE THE NEW 1052 DEVICE
C4
DMKCVTBH
CALL - CONVERT VIRTUAL ADDRESS TO EBCDIC
ENDCONS
D3
FREUSER
UNLOCK USER'S CONTROL BLOCKS
02 D3
D4
DMKSCNVD
CALL - COMPUTE ADDRESS OF EXISTING DEVICE
E4
DMKCVTBH
CALL - CONVERT ADDRESS TO EBCDIC
F4
DMKSCNVN
CALL - GET DEVICE TYPE IN EBCDIC
G4
MSG= DMKDEF092E TYPE VADDR ALREADY DEFINED
01 J5

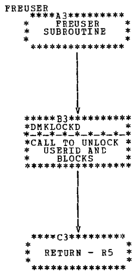

02J1
04 A3
REDEFIN
REDEFINE AN EXISTING VIRTUAL DEVICE AT A NEW ADDRESS
INVFADD
B2
FREUSER
UNLOCK USER'S CONTROL BLOCKS
B3
DMKSCNVU
ERR
CALL DMKSCNVU LOCATE BLOCKS FOR OLD DEVICE
O.K.
04 C4
02H1
DEDCHAN
C2
DMKCVTBH
CALL - CONVERT ADDRESS TO EBCDIC
C3
DEVICE ON DEDICATED CHANNEL
YES
NO
C4
FREUSER
RELEASE CONTROL BLOCKS
D2
MSG= DMKDEF040E DEV VADDR DOES NOT EXIST
01 J5
D3
DMKCFPRD
CALL DMKCFPRD RESET ANY ACTIVE I/O
D4
DMKCVTBH
CALL - CONVERT VIRTUAL ADDRESS TO HEX
E3
IS THE OLD VCUTYPE = VCUCTCA
NO
YES
E4
DMKCVTBD
CONVERT CHANNEL ADDRESS TO DECIMAL
F3
FORCE NEW ADDRESS TO CONTROL UNIT BASE ADDR = CC0
F4
MSG= DMKDEF136E CHAN XX DEDICATED
01 J5
DETLOOP
G3
MARK VDEVBLOK AS UNUSED SPARE
DETCULP
H3
SCAN VCUBLOK FOR OTHER ACTIVE VDEVBLOKS
YES
05 B1
NO
J3
MARK VCUBLOK AS UNUSED SPARE
DETCHLP
K3
SCAN VCHBLOK FOR OTHER ACTIVE VCUBLOKS
NO
05 A1
YES
05 B1

DMKDEF -- Process DEFINE Command; Define Virtual Storage or a Device (Parts 3 and 4 of 7)

| DMKDEF -- Process DEFINE Command; Define Virtual Storage or a Device (Parts 5 and 6 of 7)

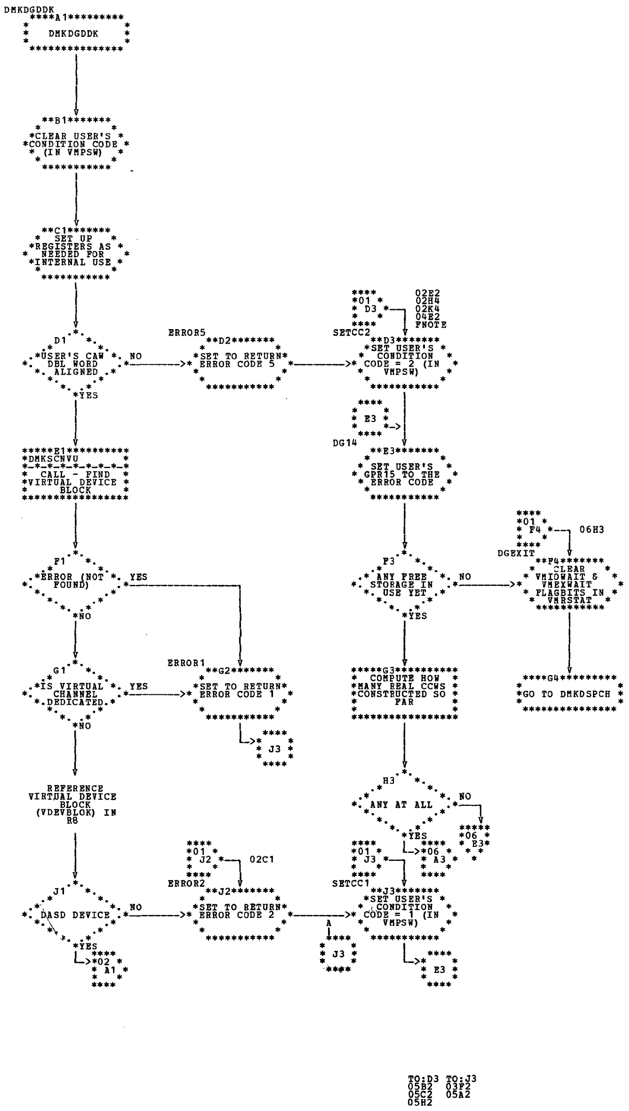

05 A1 04K3
A1 MARK VCHBLOK AS UNUSED SPARE
05 B1 04H3 04K3
DETDONE
B1 PRESERVE ALL AUXILIARY BLOCK POINTERS, ETC. FROM VDEVBLOK
C1 DMKVDSDF CALL DMKVDSDF DEFINE DEVICE AT NEW ADDRESS
D1 RESTORE SAVED POINTERS, ETC. IN NEW BLOCK
E1 DID HE RE-DEFINE DEDICATED DEVICE
NO
YES
FIXLINK
E2 IS THIS A DASD DEVICE
NO
YES
G1
F1 RESET 'RDEVATT' FIELD IN REAL DEVICE BLOCK
F2 IS IT A TEMPORARY DISK
YES
NO
G1
CHKCONS
G1 DID HE REDEFINE PRIMARY CONSOLE
NO
YES
G2 MORE THAN ONE LINK TO THIS AREA.
YES
NO
LINKFIX
G3 ANY LINK CHAIN TO THIS. DASD AREA.
NO
YES
J1
H1 PATCH RDEVBLOK AND VMBLOK FOR NEW PRIMARY CONS ADDR
H2 LINK NEW VDEVBLOK TO ITSELF
H3 SCAN LINK LIST FOR VDEVBLOK BEFORE REDEFINED VDEVBLOK
RDEFREE
J1 FREUSER UNLOCK USER'S CONTROL BLOCKS
J1
A
02 D3
J3 FOUND POINTER TO OLD BLOCK.
NO
YES
A
K3 PATCH LINK CHAIN WITH NEW VDEVBLOK ADDRESS
05 A4 01E3
DEFCORE
RE-DEFINE VIRTUAL STORAGE SIZE
B4 IS STORAGE SIZE PARM VALID
NO
01 G5
YES
C4 DMKCVTDB CALL DMKCVTDB CONVERT NEW STORAGE SIZE
ERR
01 G5
O.K.
D4 ROUND UP SIZE TO NEAREST 4096-BYTE BOUNDARY
E4 DMKCFPRR CALL DMKCFPRR PERFORM VIRTUAL SYSTEM RESET
F4 DMKPGSPO CALL DMKPGSPO RELEASE ACTIVE VIRTUAL STORAGE
G4 DOES NEW SIZE EQUAL CURRENT SIZE
NO
06 A1
YES
06 A3

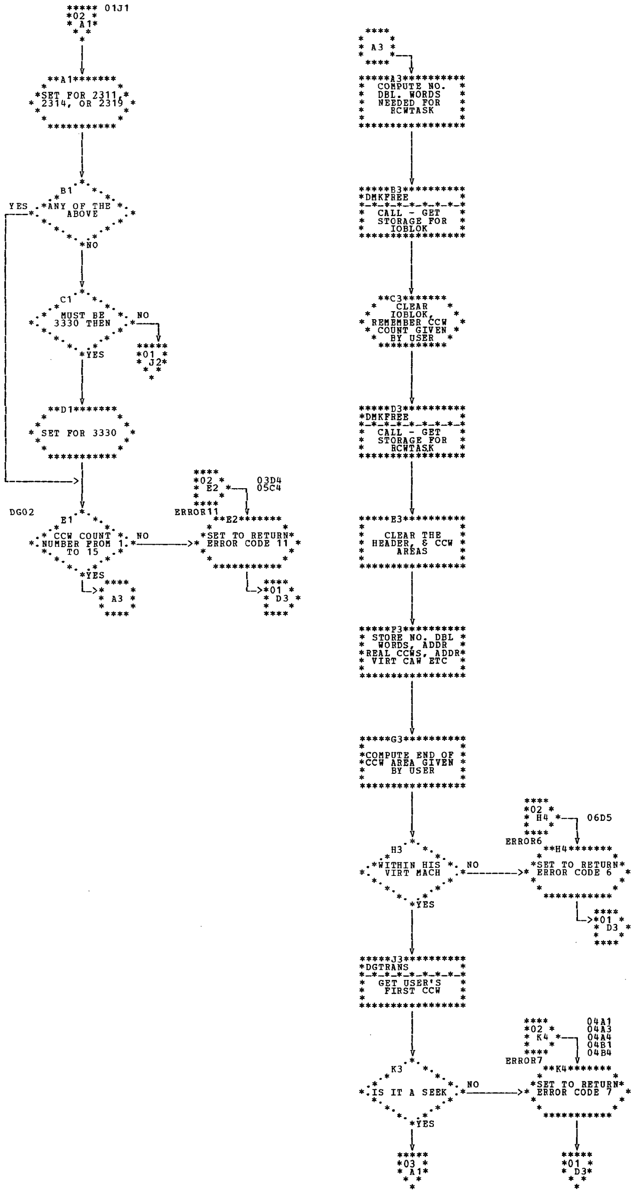

06 A1 05G4
A1 DMKFREE CALL DMKFREE FREE STORAGE FOR UDBFBLOK
B1 DMKUDRFU CALL DMKUDRFU LOCATE USER DIRECTORY BLOCK
C1 DMKUDRRD CALL DMKUDRRD ACCESS UMACBLOK VM DESCRIPTION
D1 LOAD MAXIMUM STORAGE SIZE ALLOWED FROM UMACBLOK
E1 DMKUDRRV CALL DMKUDRRV RELEASE USER'S DIRECTORY BLOCK
F1 IS HE ALLOWED THE NEW AMOUNT STORAGE
NO
YES
EXCESIV
F2 MSG= DMKDEF094E STORAGE SIZE EXCEEDS MAXIMUM
01 H5
G1 DMKBLDRL CALL DMKBLDRL RELEASE CURRENT STORAGE TABLES
H1 REPLACE STORAGE LIMIT VALUE IN USER'S VMBLOK
J1 DMKBLDRT CALL DMKBLDRT REBUILD VIRTUAL STORAGE TABLES
06 A3 05G4
STORMSG
CONSTRUCT AND TYPE VERIFICATION MESSAGE
B3 BUILD MSG 'STORAGE = NNNNNK' IN SAVE AREA
C3 DMKCVTBD CALL DMKCVTBD CONVERT NEW STORAGE SIZE
D3 INSERT ROUNDED NEW STORAGE SIZE IN MESSAGE
02 A5
06 A4 02G1
EXISTCU
A4 FIND ANY ONE DEVICE ON EXISTING CONTROL UNIT
03 B4
LOKUSER
A5 LOKUSER SUBROUTINE- PUT LOCK ON USERID
B5 DMKLOCKQ CALL TO LOCK THE USERID AND BLOCKS
C5 RETURN - R5

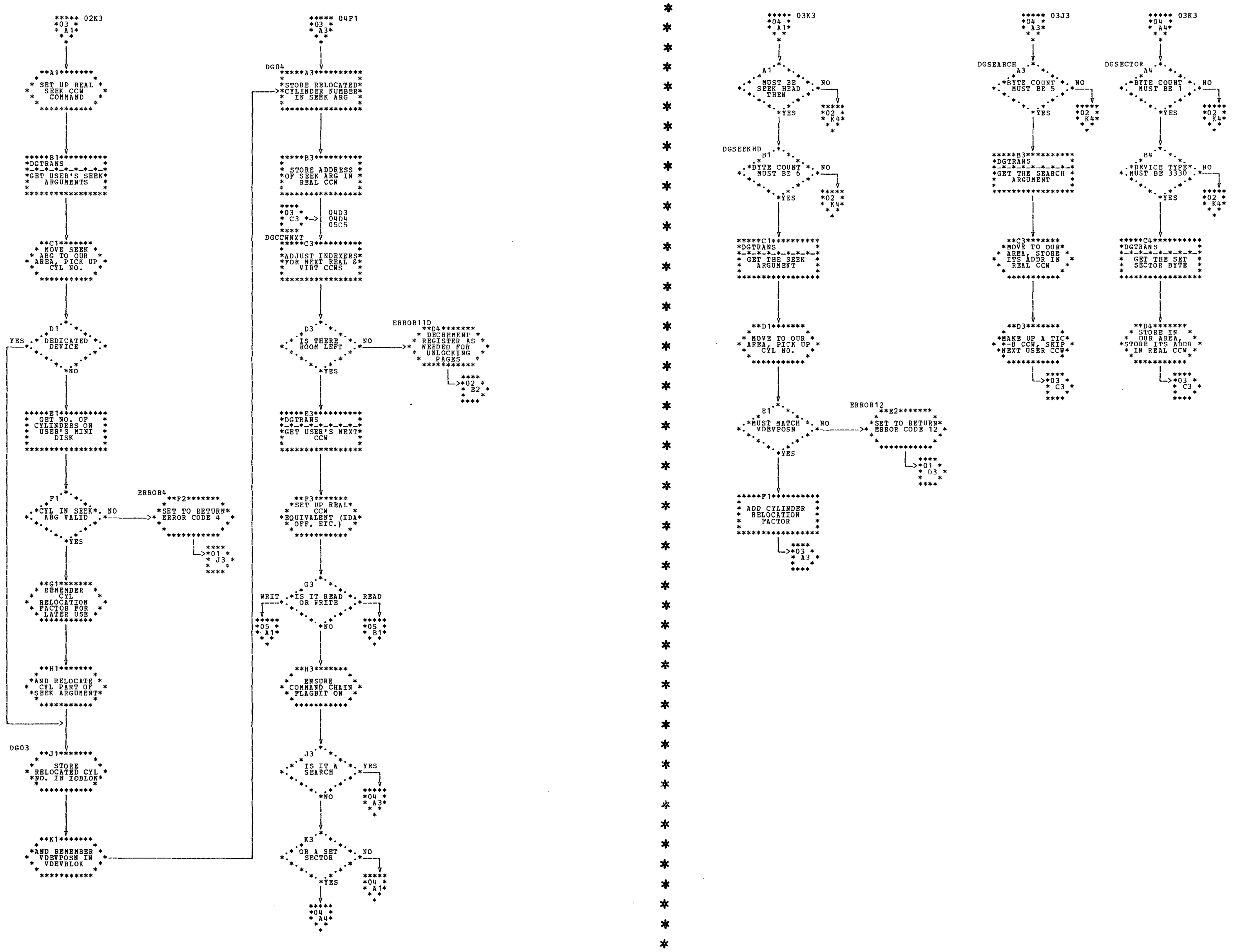


| DMKDEF -- Process DEFINE Command; Define Virtual Storage or a Device (Part 7 of 7)

DMKDGD -- Perform DASD I/O (Parts 1 and 2 of 8)

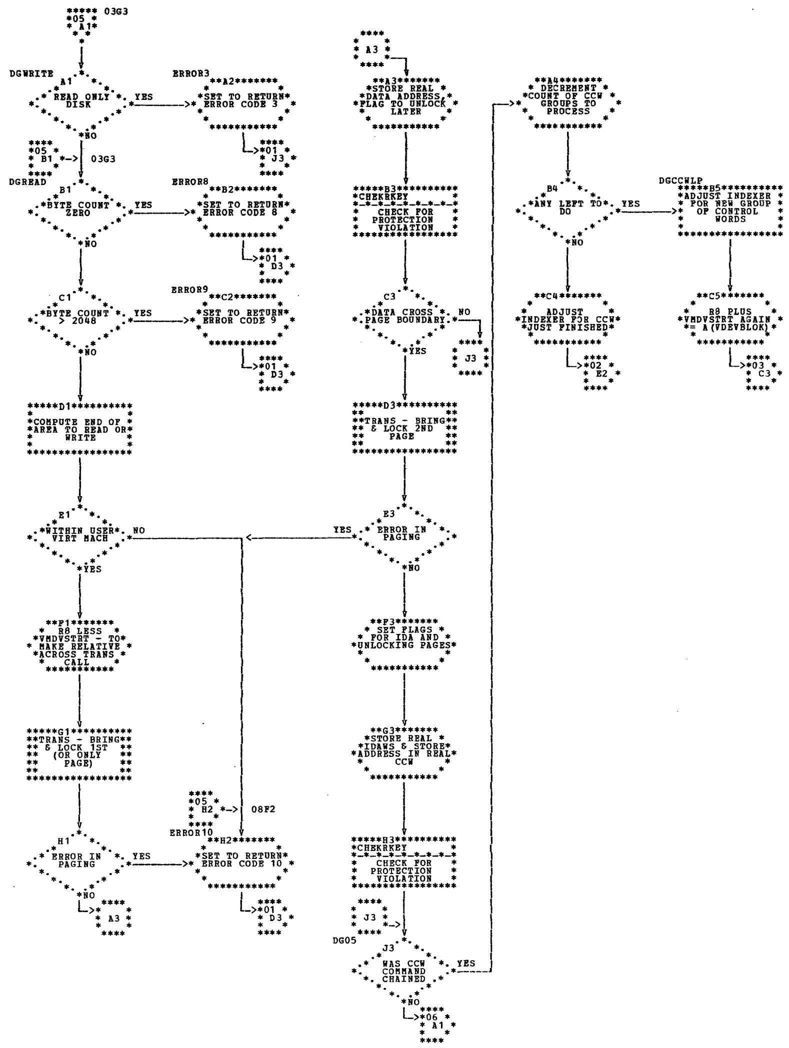

DMKDGDDK
A1 DMKDGDDK
B1 CLEAR USER'S CONDITION CODE (IN VMPSW)
C1 SET UP REGISTERS AS NEEDED FOR INTERNAL USE
D1 USER'S CAW DBL WORD ALIGNED
NO
YES
ERROR5
D2 SET TO RETURN ERROR CODE 5
01 D3
02E2
02H4
02K4
04E2
FNOTE
SETCC2
D3 SET USER'S CONDITION CODE = 2 (IN VMPSW)
E3
DG14
E3 SET USER'S GPR15 TO THE ERROR CODE
E1 DMKSCNVU CALL - FIND VIRTUAL DEVICE BLOCK
F1 ERROR (NOT FOUND)
YES
NO
F3 ANY FREE STORAGE IN USE YET
NO
YES
01 F4
06H3
DGEXIT
F4 CLEAR VMIOWAIT & VMEXWAIT FLAGBITS IN VMRSTAT
G4 GO TO DMKDSPCH
G1 IS VIRTUAL CHANNEL DEDICATED
YES
NO
ERROR1
G2 SET TO RETURN ERROR CODE 1
J3
G3 COMPUTE HOW MANY REAL CCWS CONSTRUCTED SO FAR
REFERENCE VIRTUAL DEVICE BLOCK (VDEVBLOK) IN R8
H3 ANY AT ALL
NO
YES
06 E3
06 A3
01 J2
02C1
ERROR2
J2 SET TO RETURN ERROR CODE 2
01 J3
SETCC1
J3 SET USER'S CONDITION CODE = 1 (IN VMPSW)
J1 DASD DEVICE
NO
YES
02 A1
A
J3
E3
TO:D3 TO:J3
05B2 03F2
05C2 05A2
05H2

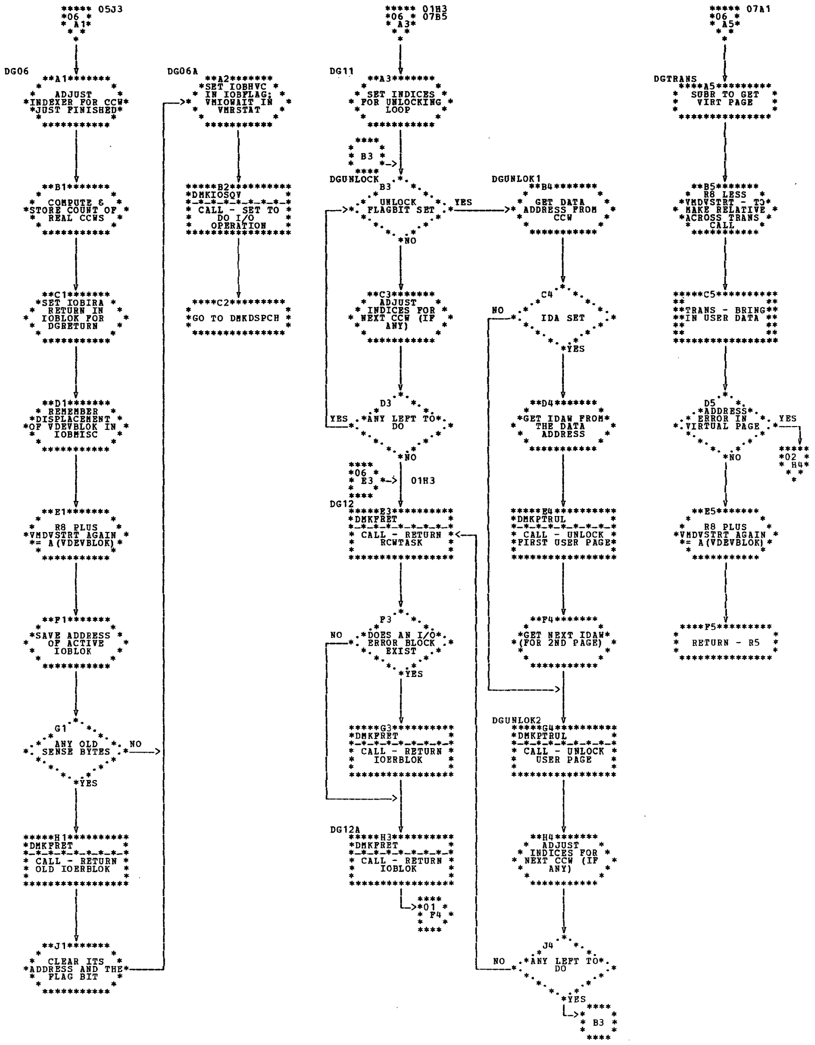

02 A1
01J1
A1 SET FOR 2311, 2314, OR 2319
B1 ANY OF THE ABOVE
YES
NO
C1 MUST BE 3330 THEN
NO
YES
01 J2
D1 SET FOR 3330
DG02
E1 CCW COUNT NUMBER FROM 1 TO 15
NO
YES
A3
02 E2
03D4
05C4
ERROR11
E2 SET TO RETURN ERROR CODE 11
01 D3
A3
A3 COMPUTE NO. DBL. WORDS NEEDED FOR RCWTASK
B3 DMKFREE CALL - GET STORAGE FOR IOBLOK
C3 CLEAR IOBLOK, REMEMBER CCW COUNT GIVEN BY USER
D3 DMKFREE CALL - GET STORAGE FOR RCWTASK
E3 CLEAR THE HEADER, & CCW AREAS
F3 STORE NO. DBL WORDS, ADDR REAL CCWS, ADDR VIRT CAW ETC
G3 COMPUTE END OF CCW AREA GIVEN BY USER
H3 WITHIN HIS VIRT MACH
NO
YES
02 H4
06D5
ERROR6
H4 SET TO RETURN ERROR CODE 6
01 D3
J3 DGTRANS GET USER'S FIRST CCW
K3 IS IT A SEEK
NO
YES
02 K4
04A1
04A3
04A4
04B1
04B4
ERROR7
K4 SET TO RETURN ERROR CODE 7
03 A1
01 D3

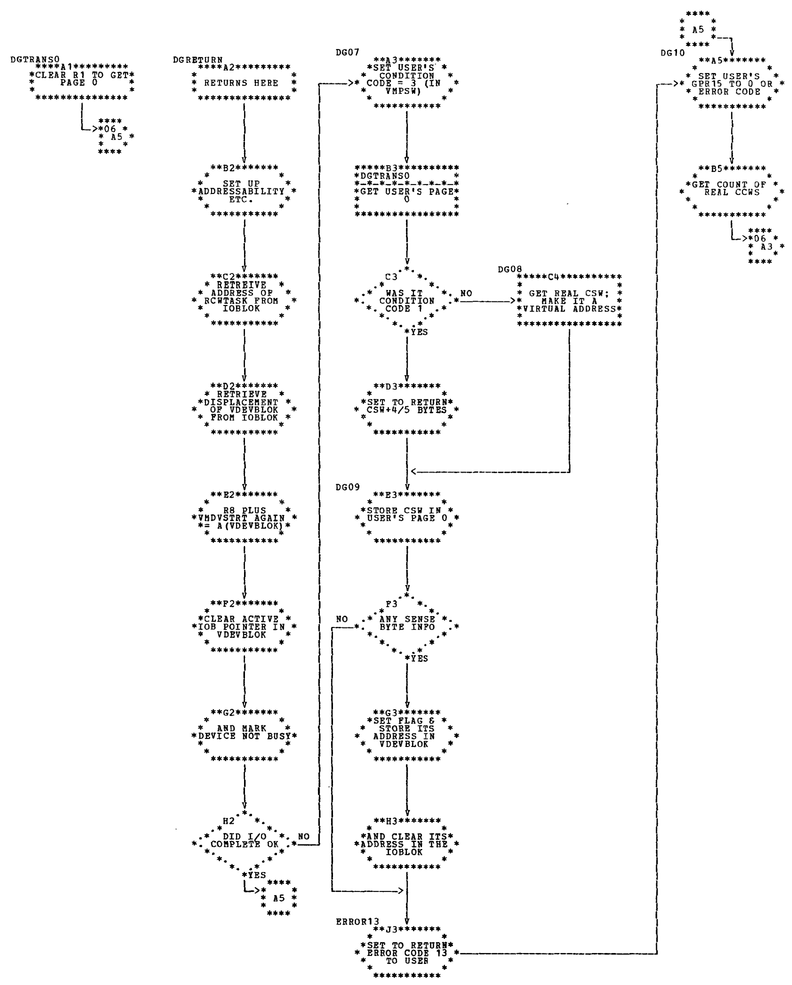

02K3
03 A1
A1 SET UP REAL SEEK CCW COMMAND
B1 DGTRANS GET USER'S SEEK ARGUMENTS
C1 MOVE SEEK ARG TO OUR AREA, PICK UP CYL NO.
D1 DEDICATED DEVICE
YES
NO
E1 GET NO. OF CYLINDERS ON USER'S MINI DISK
F1 CYL IN SEEK ARG VALID
NO
YES
ERROR4
F2 SET TO RETURN ERROR CODE 4
01 J3
G1 REMEMBER CYL RELOCATION FACTOR FOR LATER USE
H1 AND RELOCATE CYL PART OF SEEK ARGUMENT
DG03
J1 STORE RELOCATED CYL NO. IN IOBLOK
K1 AND REMEMBER VDEVPOSN IN VDEVBLOK
04F1
03 A3
DG04
A3 STORE RELOCATED CYLINDER NUMBER IN SEEK ARG
B3 STORE ADDRESS OF SEEK ARG IN REAL CCW
03 C3
04D3
04D4
05C5
DGCCWNXT
C3 ADJUST INDEXERS FOR NEXT REAL & VIRT CCWS
D3 IS THERE ROOM LEFT
NO
YES
ERROR11D
D4 DECREMENT REGISTER AS NEEDED FOR UNLOCKING PAGES
02 E2
E3 DGTRANS GET USER'S NEXT CCW
F3 SET UP REAL CCW EQUIVALENT (IDA OFF, ETC.)
G3 IS IT READ OR WRITE
WRIT
READ
NO
05 A1
05 B1
H3 ENSURE COMMAND CHAIN FLAGBIT ON
J3 IS IT A SEARCH
YES
NO
04 A3
K3 OR A SET SECTOR
NO
YES
04 A1
04 A4
03K3
04 A1
A1 MUST BE SEEK HEAD THEN
NO
YES
02 K4
DGSEEKHD
B1 BYTE COUNT MUST BE 6
NO
YES
02 K4
C1 DGTRANS GET THE SEEK ARGUMENT
D1 MOVE TO OUR AREA, PICK UP CYL NO.
E1 MUST MATCH VDEVPOSN
NO
YES
ERROR12
E2 SET TO RETURN ERROR CODE 12
01 D3
F1 ADD CYLINDER RELOCATION FACTOR
03 A3
03J3
04 A3
DGSEARCH
A3 BYTE COUNT MUST BE 5
NO
YES
02 K4
B3 DGTRANS GET THE SEARCH ARGUMENT
C3 MOVE TO OUR AREA, STORE ITS ADDR IN REAL CCW
D3 MAKE UP A TIC *-8 CCW, SKIP NEXT USER CCW
03 C3
03K3
04 A4
DGSECTOR
A4 BYTE COUNT MUST BE 1
NO
YES
02 K4
B4 DEVICE TYPE MUST BE 3330
NO
YES
02 K4
C4 DGTRANS GET THE SET SECTOR BYTE
D4 STORE IN OUR AREA, STORE ITS ADDR IN REAL CCW
03 C3

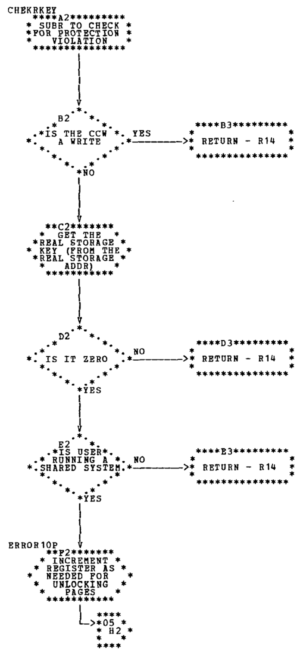
| DMKDGD -- Perform DASD I/O (Parts 3 and 4 of 8)

| DMKDGD -- Perform DASD I/O (Parts 5 and 6 of 8)


05 A1 03G3
DGWRITE A1 READ ONLY DISK
YES
NO
ERROR3 A2 SET TO RETURN ERROR CODE 3
01 J3
05 B1 03G3
DGREAD B1 BYTE COUNT ZERO
ERROR8 B2 SET TO RETURN ERROR CODE 8
01 D3
C1 BYTE COUNT > 2048
ERROR9 C2 SET TO RETURN ERROR CODE 9
01 D3
D1 COMPUTE END OF AREA TO READ OR WRITE
E1 WITHIN USER VIRT MACH
F1 R8 LESS VMDVSTRT - TO MAKE RELATIVE ACROSS TRANS CALL
G1 TRANS - BRING & LOCK 1ST (OR ONLY PAGE)
H1 ERROR IN PAGING
A3
05 H2 08F2
ERROR10 H2 SET TO RETURN ERROR CODE 10
01 D3
A3
A3 STORE REAL DATA ADDRESS, FLAG TO UNLOCK LATER
B3 CHEKRKEY CHECK FOR PROTECTION VIOLATION
C3 DATA CROSS PAGE BOUNDARY
J3
D3 TRANS - BRING & LOCK 2ND PAGE
E3 ERROR IN PAGING
F3 SET FLAGS FOR IDA AND UNLOCKING PAGES
G3 STORE REAL IDAWS & STORE ADDRESS IN REAL CCW
H3 CHEKRKEY CHECK FOR PROTECTION VIOLATION
J3
DG05 J3 WAS CCW COMMAND CHAINED
06 A1
A4 DECREMENT COUNT OF CCW GROUPS TO PROCESS
B4 ANY LEFT TO DO
C4 ADJUST INDEXER FOR CCW JUST FINISHED
02 E2
DGCCWLP B5 ADJUST INDEXER FOR NEW GROUP OF CONTROL WORDS
C5 R8 PLUS VMDVSTRT AGAIN = A(VDEVBLOK)
03 C3

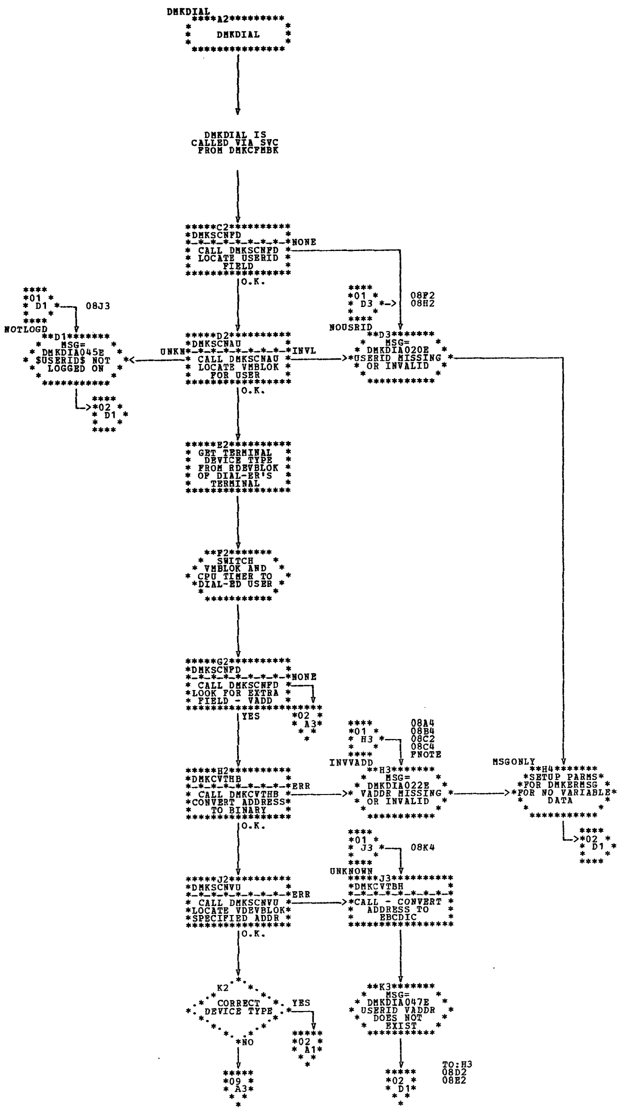

06 A1 05J3
DG06 A1 ADJUST INDEXER FOR CCW JUST FINISHED
B1 COMPUTE & STORE COUNT OF REAL CCWS
C1 SET IOBIRA RETURN IN IOBLOK FOR DGRETURN
D1 REMEMBER DISPLACEMENT OF VDEVBLOK IN IOBMISC
E1 R8 PLUS VMDVSTRT AGAIN = A(VDEVBLOK)
F1 SAVE ADDRESS OF ACTIVE IOBLOK
G1 ANY OLD SENSE BYTES
YES
NO
H1 DMKFRET CALL - RETURN OLD IOERBLOK
J1 CLEAR ITS ADDRESS AND THE FLAG BIT
DG06A A2 SET IOBHVC IN IOBFLAG; VMIOWAIT IN VMRSTAT
B2 DMKIOSQV CALL - SET TO DO I/O OPERATION
C2 GO TO DMKDSPCH
06 A3 01H3 07B5
DG11 A3 SET INDICES FOR UNLOCKING LOOP
B3
DGUNLOCK B3 UNLOCK FLAGBIT SET
C3 ADJUST INDICES FOR NEXT CCW (IF ANY)
D3 ANY LEFT TO DO
06 E3 01H3
DG12 E3 DMKFRET CALL - RETURN RCWTASK
F3 DOES AN I/O ERROR BLOCK EXIST
G3 DMKFRET CALL - RETURN IOERBLOK
DG12A H3 DMKFRET CALL - RETURN IOBLOK
01 F4
DGUNLOK1 B4 GET DATA ADDRESS FROM CCW
C4 IDA SET
D4 GET IDAW FROM THE DATA ADDRESS
E4 DMKPTRUL CALL - UNLOCK FIRST USER PAGE
F4 GET NEXT IDAW (FOR 2ND PAGE)
DGUNLOK2 G4 DMKPTRUL CALL - UNLOCK USER PAGE
H4 ADJUST INDICES FOR NEXT CCW (IF ANY)
J4 ANY LEFT TO DO
B3
06 A5 07A1
DGTRANS A5 SUBR TO GET VIRT PAGE
B5 R8 LESS VMDVSTRT - TO MAKE RELATIVE ACROSS TRANS CALL
C5 TRANS - BRING IN USER DATA
D5 ADDRESS ERROR IN VIRTUAL PAGE
02 H4
E5 R8 PLUS VMDVSTRT AGAIN = A(VDEVBLOK)
F5 RETURN - R5

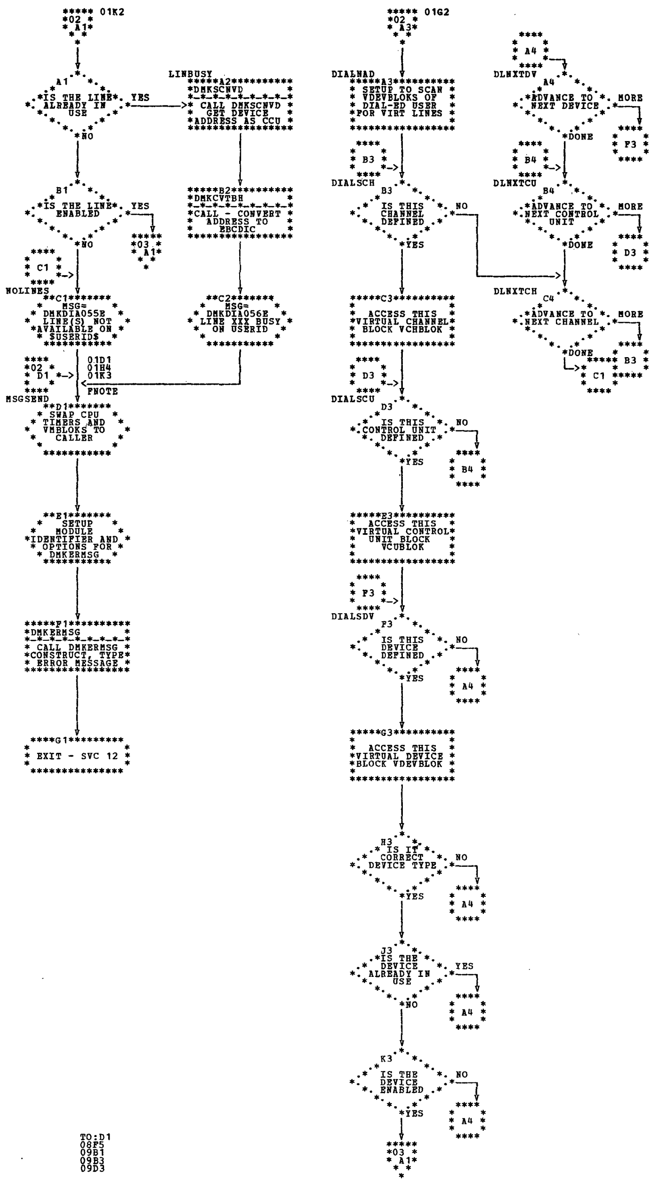

DGTRANS0
A1
CLEAR R1 TO GET PAGE 0
06 A5
DGRETURN
A2
RETURNS HERE
B2
SET UP ADDRESSABILITY ETC.
C2
RETRIEVE ADDRESS OF RCWTASK FROM IOBLOK
D2
RETRIEVE DISPLACEMENT OF VDEVBLOK FROM IOBLOK
E2
R8 PLUS VMDVSTRT AGAIN = A(VDEVBLOK)
F2
CLEAR ACTIVE IOB POINTER IN VDEVBLOK
G2
AND MARK DEVICE NOT BUSY
H2
DID I/O COMPLETE OK
NO
YES
A5
DG07
A3
SET USER'S CONDITION CODE = 3 (IN VMPSW)
B3
DGTRANS0
GET USER'S PAGE 0
C3
WAS IT CONDITION CODE 1
NO
YES
DG08
C4
GET REAL CSW; MAKE IT A VIRTUAL ADDRESS
D3
SET TO RETURN CSW+4/5 BYTES
DG09
E3
STORE CSW IN USER'S PAGE 0
F3
ANY SENSE BYTE INFO
NO
YES
G3
SET FLAG & STORE ITS ADDRESS IN VDEVBLOK
H3
AND CLEAR ITS ADDRESS IN THE IOBLOK
ERROR13
J3
SET TO RETURN ERROR CODE 13 TO USER
A5
DG10
A5
SET USER'S GPR15 TO 0 OR ERROR CODE
B5
GET COUNT OF REAL CCWS
06 A3

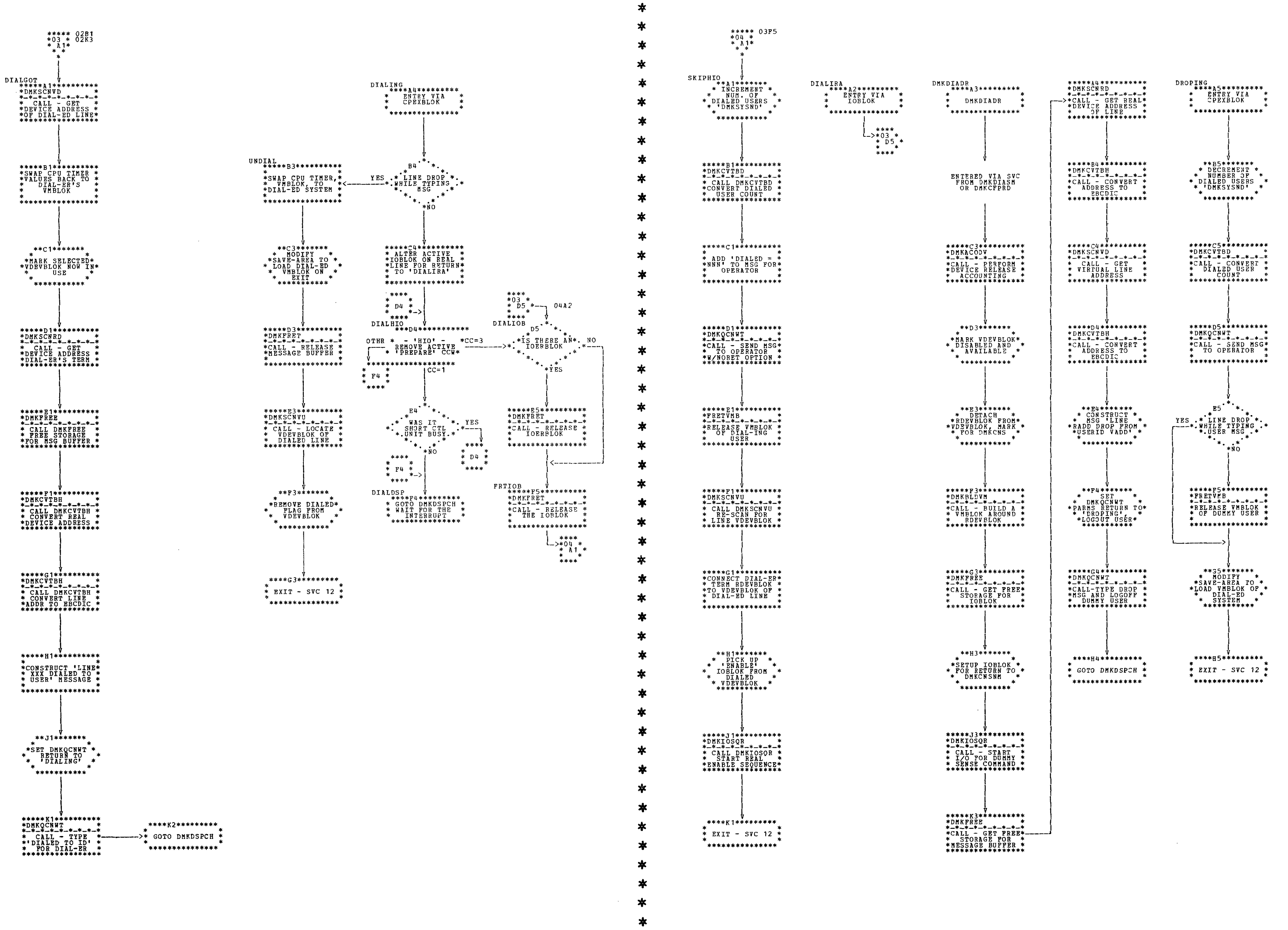

CHEKRKEY
A2
SUBR TO CHECK FOR PROTECTION VIOLATION
B2
IS THE CCW A WRITE
YES
NO
B3
RETURN - R14
C2
GET THE REAL STORAGE KEY (FROM THE REAL STORAGE ADDR)
D2
IS IT ZERO
NO
YES
D3
RETURN - R14
E2
IS USER RUNNING A SHARED SYSTEM
NO
YES
E3
RETURN - R14
ERROR10P
F2
INCREMENT REGISTER AS NEEDED FOR UNLOCKING PAGES
05 H2

| DMKDGD -- Perform DASD I/O (Parts 7 and 8 of 8)

| DMKDIA -- Process DIAL Command, Connect to a Virtual System (Parts 1 and 2 of 9)

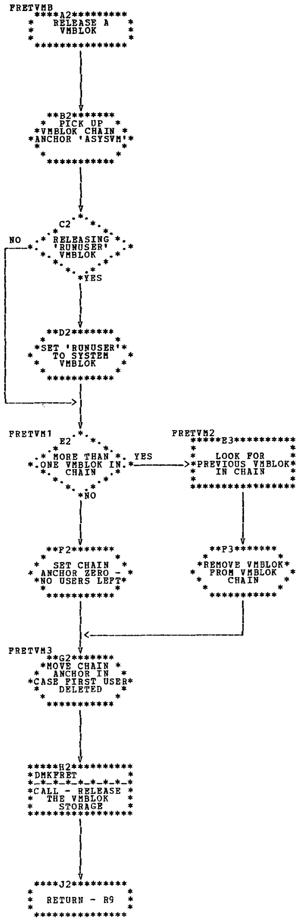

DMKDIAL
A2
DMKDIAL
DMKDIAL IS CALLED VIA SVC FROM DMKCFMBK
C2
DMKSCNFD
CALL DMKSCNFD LOCATE USERID FIELD
NONE
O.K.
01 D1
08J3
NOTLOGD
D1
MSG= DMKDIA045E $USERID$ NOT LOGGED ON
02 D1
D2
DMKSCNAU
CALL DMKSCNAU LOCATE VMBLOK FOR USER
UNKN
INVL
O.K.
01 D3
08F2
08H2
NOUSRID
D3
MSG= DMKDIA020E USERID MISSING OR INVALID
E2
GET TERMINAL DEVICE TYPE FROM RDEVBLOK OF DIAL-ER'S TERMINAL
F2
SWITCH VMBLOK AND CPU TIMER TO DIAL-ED USER
G2
DMKSCNFD
CALL DMKSCNFD LOOK FOR EXTRA FIELD - VADD
NONE
02 A3
YES
01 H3
08A4
08B4
08C2
08C4
FNOTE
INVVADD
H2
DMKCVTHB
CALL DMKCVTHB CONVERT ADDRESS TO BINARY
ERR
O.K.
H3
MSG= DMKDIA022E VADDR MISSING OR INVALID
MSGONLY
H4
SETUP PARMS FOR DMKERMSG FOR NO VARIABLE DATA
02 D1
01 J3
08K4
UNKNOWN
J2
DMKSCNVU
CALL DMKSCNVU LOCATE VDEVBLOK SPECIFIED ADDR
ERR
O.K.
J3
DMKCVTBH
CALL - CONVERT ADDRESS TO EBCDIC
K2
CORRECT DEVICE TYPE
YES
02 A1
NO
09 A3
K3
MSG= DMKDIA047E USERID VADDR DOES NOT EXIST
02 D1
TO:H3
08D2
08E2

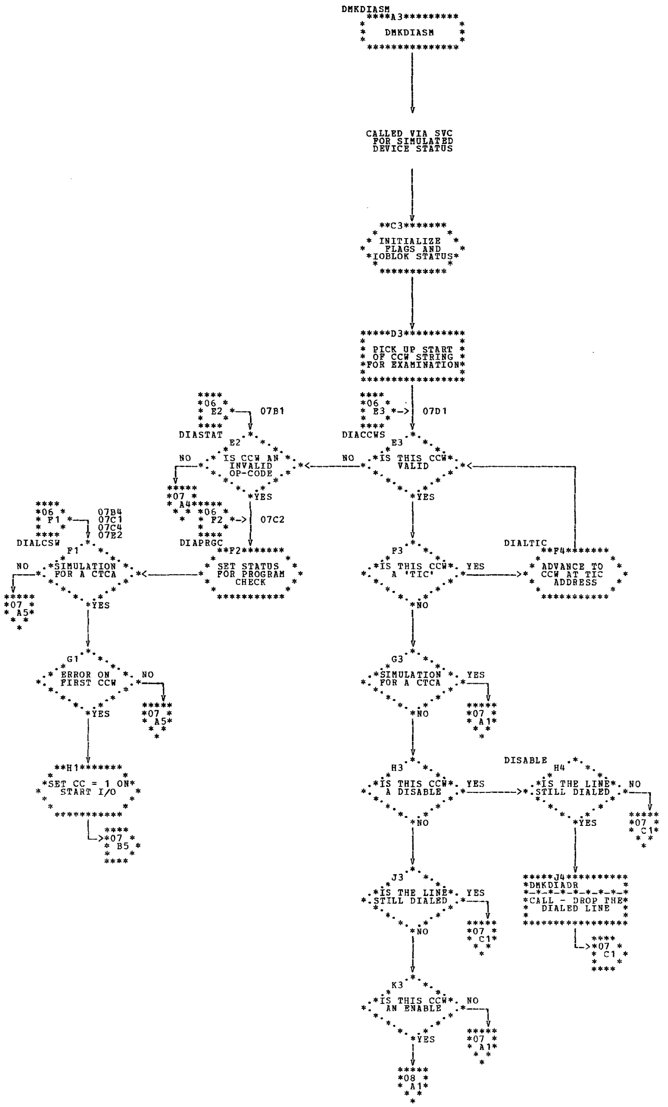

02 A1
01K2
A1
IS THE LINE ALREADY IN USE
YES
NO
LINBUSY
A2
DMKSCNVD
CALL DMKSCNVD GET DEVICE ADDRESS AS CCU
B1
IS THE LINE ENABLED
YES
03 A1
NO
B2
DMKCVTBH
CALL - CONVERT ADDRESS TO EBCDIC
C1
NOLINES
C1
MSG= DMKDIA055E LINE(S) NOT AVAILABLE ON $USERID$
C2
MSG= DMKDIA056E LINE XXX BUSY ON USERID
02 D1
01D1
01H4
01K3
FNOTE
MSGSEND
D1
SWAP CPU TIMERS AND VMBLOKS TO CALLER
E1
SETUP MODULE IDENTIFIER AND OPTIONS FOR DMKERMSG
F1
DMKERMSG
CALL DMKERMSG CONSTRUCT, TYPE ERROR MESSAGE
G1
EXIT - SVC 12
TO:D1
08F5
09B1
09B3
09D3
02 A3
01G2
DIALNAD
A3
SETUP TO SCAN VDEVBLOKS OF DIAL-ED USER FOR VIRT LINES
B3
DIALSCH
B3
IS THIS CHANNEL DEFINED
NO
YES
C3
ACCESS THIS VIRTUAL CHANNEL BLOCK VCHBLOK
D3
DIALSCU
D3
IS THIS CONTROL UNIT DEFINED
NO
B4
YES
E3
ACCESS THIS VIRTUAL CONTROL UNIT BLOCK VCUBLOK
F3
DIALSDV
F3
IS THIS DEVICE DEFINED
NO
A4
YES
G3
ACCESS THIS VIRTUAL DEVICE BLOCK VDEVBLOK
H3
IS IT CORRECT DEVICE TYPE
NO
A4
YES
J3
IS THE DEVICE ALREADY IN USE
YES
A4
NO
K3
IS THE DEVICE ENABLED
NO
A4
YES
03 A1
A4
DLNXTDV
A4
ADVANCE TO NEXT DEVICE
MORE
F3
DONE
B4
DLNXTCU
B4
ADVANCE TO NEXT CONTROL UNIT
MORE
D3
DONE
DLNXTCH
C4
ADVANCE TO NEXT CHANNEL
MORE
DONE
B3
C1

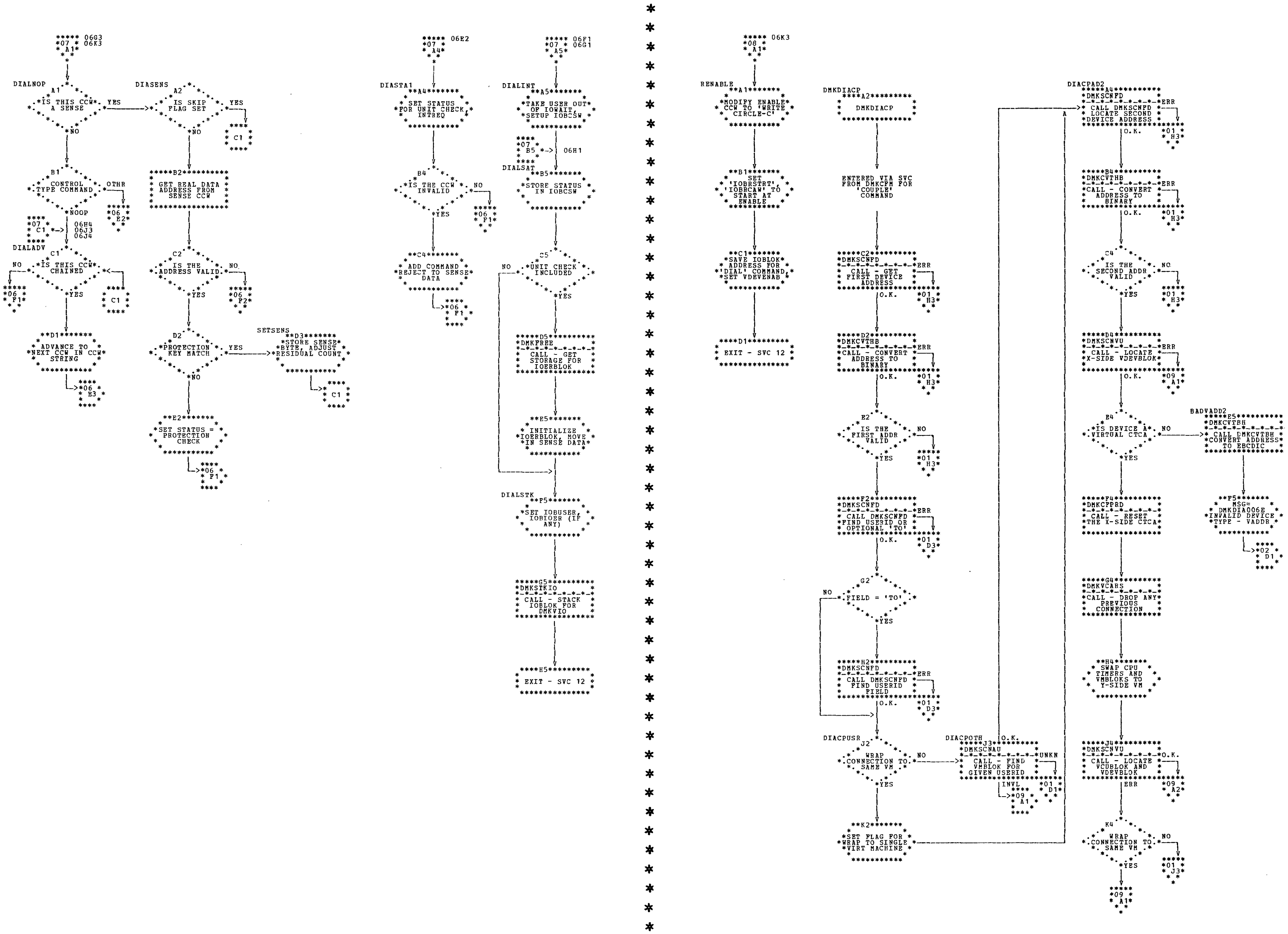

02B1 03 A1 02K3
DIALGOT
A1 DMKSCNVD CALL - GET DEVICE ADDRESS OF DIAL-ED LINE
B1 SWAP CPU TIMER VALUES BACK TO DIAL-ER'S VMBLOK
C1 MARK SELECTED VDEVBLOK NOW IN USE
D1 DMKSCNRD CALL - GET DEVICE ADDRESS DIAL-ER'S TERM
E1 DMKFREE CALL DMKFREE FREE STORAGE FOR MSG BUFFER
F1 DMKCVTBH CALL DMKCVTBH CONVERT REAL DEVICE ADDRESS
G1 DMKCVTBH CALL DMKCVTBH CONVERT LINE ADDR TO EBCDIC
H1 CONSTRUCT 'LINE XXX DIALED TO USER' MESSAGE
J1 SET DMKQCNWT RETURN TO 'DIALING'
K1 DMKQCNWT CALL - TYPE 'DIALED TO ID' FOR DIAL-ER
K2 GOTO DMKDSPCH
UNDIAL
B3 SWAP CPU TIMER, VMBLOK, TO DIAL-ED SYSTEM
C3 MODIFY SAVE-AREA TO LOAD DIAL-ED VMBLOK ON EXIT
D3 DMKFRET CALL - RELEASE MESSAGE BUFFER
E3 DMKSCNVU CALL - LOCATE VDEVBLOK OF DIALED LINE
F3 REMOVE DIALED FLAG FROM VDEVBLOK
G3 EXIT - SVC 12
DIALING
A4 ENTRY VIA CPEXBLOK
B4 LINE DROP WHILE TYPING MSG
YES
NO
C4 ALTER ACTIVE IOBLOK ON REAL LINE FOR RETURN TO 'DIALIRA'
D4
DIALHIO
D4 - 'HIO' - REMOVE ACTIVE 'PREPARE' CCW
OTHR
CC=3
CC=1
F4
E4 WAS IT SHORT CTL UNIT BUSY
YES
NO
D4
F4
DIALDSP
F4 GOTO DMKDSPCH WAIT FOR THE INTERRUPT
03 D5
04A2
DIALIOB
D5 IS THERE AN IOERBLOK
NO
YES
E5 DMKFRET CALL - RELEASE IOERBLOK
FRTIOB
F5 DMKFRET CALL - RELEASE THE IOBLOK
04 A1
03F5 04 A1
SKIPHIO
A1 INCREMENT NUM. OF DIALED USERS 'DMKSYSND'
B1 DMKCVTBD CALL DMKCVTBD CONVERT DIALED USER COUNT
C1 ADD 'DIALED = NNN' TO MSG FOR OPERATOR
D1 DMKQCNWT CALL - SEND MSG TO OPERATOR W/NORET OPTION
E1 FRETVMB RELEASE VMBLOK OF DIAL-ING USER
F1 DMKSCNVU CALL DMKSCNVU RE-SCAN FOR LINE VDEVBLOK
G1 CONNECT DIAL-ER TERM RDEVBLOK TO VDEVBLOK OF DIAL-ED LINE
H1 PICK UP 'ENABLE' IOBLOK FROM DIALED VDEVBLOK
J1 DMKIOSQR CALL DMKIOSQR START REAL ENABLE SEQUENCE
K1 EXIT - SVC 12
DIALIRA
A2 ENTRY VIA IOBLOK
03 D5
DMKDIADR
A3 DMKDIADR
ENTERED VIA SVC FROM DMKDIASM OR DMKCFPRD
C3 DMKACODV CALL - PERFORM DEVICE RELEASE ACCOUNTING
D3 MARK VDEVBLOK DISABLED AND AVAILABLE
E3 DETACH RDEVBLOK FROM VDEVBLOK, MARK FOR DMKCNS
F3 DMKBLDVM CALL - BUILD A VMBLOK AROUND RDEVBLOK
G3 DMKFREE CALL - GET FREE STORAGE FOR IOBLOK
H3 SETUP IOBLOK FOR RETURN TO DMKCNSNM
J3 DMKIOSQR CALL - START I/O FOR DUMMY SENSE COMMAND
K3 DMKFREE CALL - GET FREE STORAGE FOR MESSAGE BUFFER
A4 DMKSCNRD CALL - GET REAL DEVICE ADDRESS OF LINE
B4 DMKCVTBH CALL - CONVERT ADDRESS TO EBCDIC
C4 DMKSCNVD CALL - GET VIRTUAL LINE ADDRESS
D4 DMKCVTBH CALL - CONVERT ADDRESS TO EBCDIC
E4 CONSTRUCT MSG 'LINE RADD DROP FROM USERID VADD'
F4 SET DMKQCNWT PARMS RETURN TO 'DROPING', LOGOUT USER
G4 DMKQCNWT CALL-TYPE DROP MSG AND LOGOFF DUMMY USER
H4 GOTO DMKDSPCH
DROPING
A5 ENTRY VIA CPEXBLOK
B5 DECREMENT NUMBER OF DIALED USERS 'DMKSYSND'
C5 DMKCVTBD CALL - CONVERT DIALED USER COUNT
D5 DMKQCNWT CALL - SEND MSG TO OPERATOR
E5 LINE DROP WHILE TYPING USER MSG
YES
NO
F5 FRETVMB RELEASE VMBLOK OF DUMMY USER
G5 MODIFY SAVE-AREA TO LOAD VMBLOK OF DIAL-ED SYSTEM
H5 EXIT - SVC 12

DMKDIA -- Process DIAL Command, Connect to a Virtual System (Parts 3 and 4 of 9)

| DMKDIA -- Process DIAL Ccmmand, Connect to a Virtual System (Parts 5 and 6 of 9)


FRETVMB
A2
RELEASE A VMBLOK
B2
PICK UP VMBLOK CHAIN ANCHOR 'ASYSVM'
C2
RELEASING 'RUNUSER' VMBLOK
NO
YES
D2
SET 'RUNUSER' TO SYSTEM VMBLOK
FRETVM1
E2
MORE THAN ONE VMBLOK IN CHAIN
YES
NO
FRETVM2
E3
LOOK FOR PREVIOUS VMBLOK IN CHAIN
F2
SET CHAIN ANCHOR ZERO - NO USERS LEFT
F3
REMOVE VMBLOK FROM VMBLOK CHAIN
FRETVM3
G2
MOVE CHAIN ANCHOR IN CASE FIRST USER DELETED
H2
DMKFRET
CALL - RELEASE THE VMBLOK STORAGE
J2
RETURN - R9

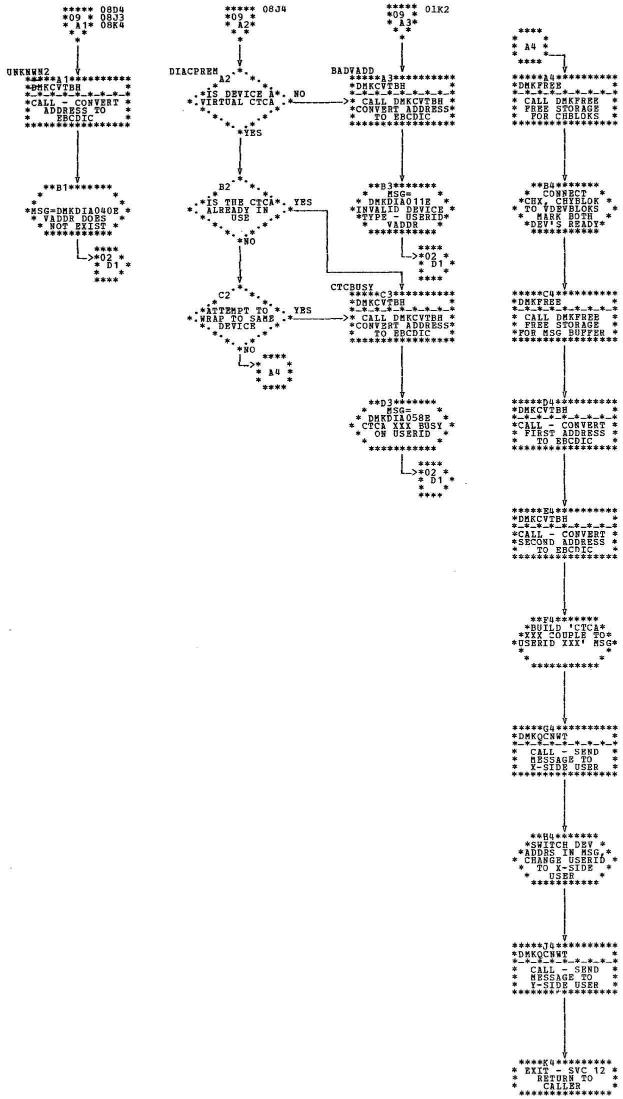


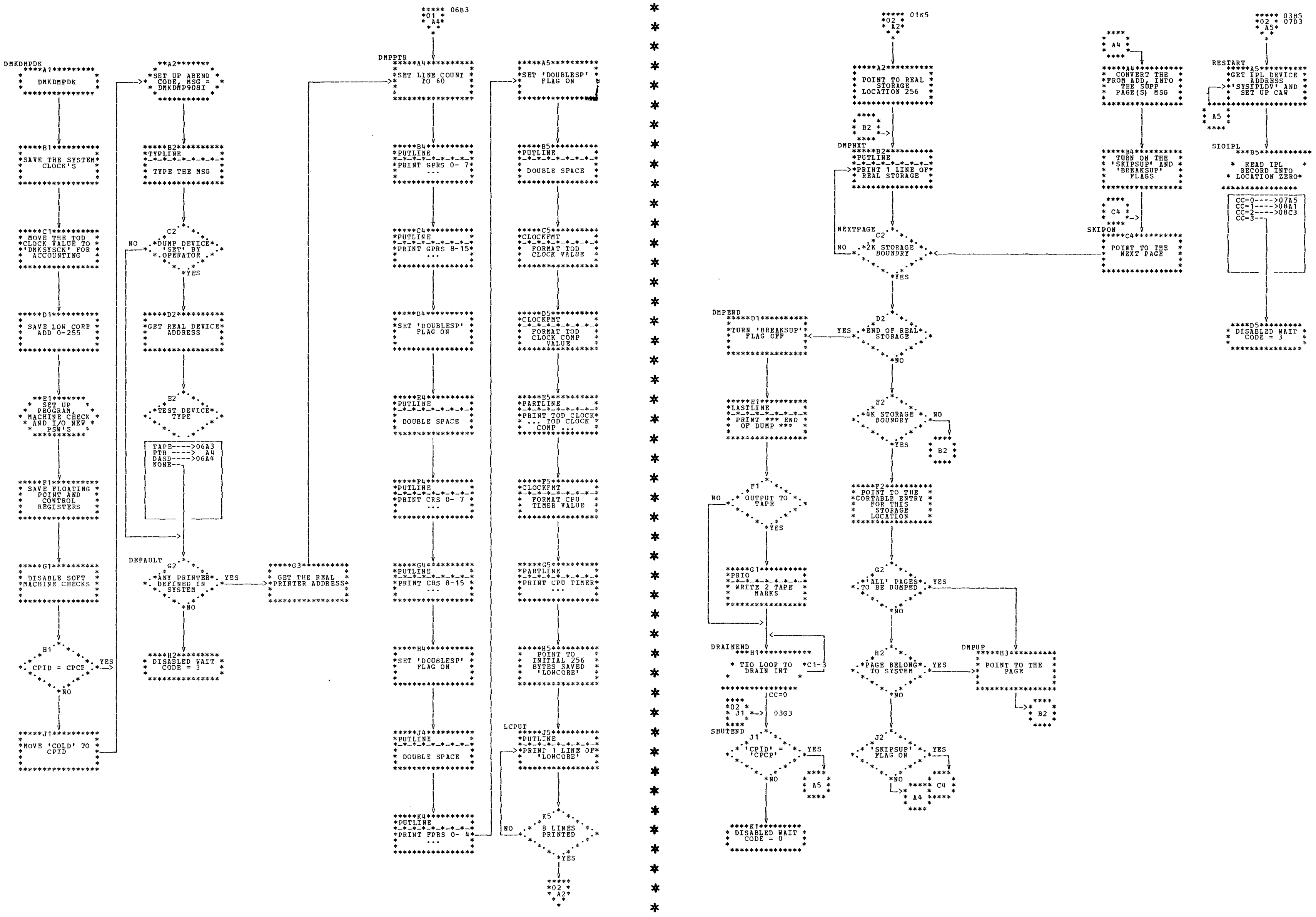


06G3
06K3
07 A1
DIALNOP
A1
IS THIS CCW A SENSE
YES
NO
DIASENS
A2
IS SKIP FLAG SET
YES
C1
NO
B1
CONTROL TYPE COMMAND
OTHR
06 E2
NOOP
07 C1
06H4
06J3
06J4
B2
GET REAL DATA ADDRESS FROM SENSE CCW
DIALADV
C1
IS THIS CCW CHAINED
NO
06 F1
YES
C1
C2
IS THE ADDRESS VALID
NO
06 F2
YES
D1
ADVANCE TO NEXT CCW IN CCW STRING
06 E3
D2
PROTECTION KEY MATCH
YES
SETSENS
D3
STORE SENSE BYTE, ADJUST RESIDUAL COUNT
C1
NO
E2
SET STATUS = PROTECTION CHECK
06 F1
06E2
07 A4
DIASTA1
A4
SET STATUS FOR UNIT CHECK, INTREQ
B4
IS THE CCW INVALID
NO
06 F1
YES
C4
ADD COMMAND REJECT TO SENSE DATA
06 F1
06F1
06G1
07 A5
DIALINT
A5
TAKE USER OUT OF IOWAIT, SETUP IOBCSW
07 B5
06H1
DIALSAT
B5
STORE STATUS IN IOBCSW
C5
UNIT CHECK INCLUDED
NO
YES
D5
DMKFREE
CALL - GET STORAGE FOR IOERBLOK
E5
INITIALIZE IOERBLOK, MOVE IN SENSE DATA
DIALSTK
F5
SET IOBUSER, IOBIOER (IF ANY)
G5
DMKSTKIO
CALL - STACK IOBLOK FOR DMKVIO
H5
EXIT - SVC 12
06K3
08 A1
RENABLE
A1
MODIFY ENABLE CCW TO 'WRITE CIRCLE-C'
B1
SET 'IOBRSTRT', 'IOBRCAW' TO START AT ENABLE
C1
SAVE IOBLOK ADDRESS FOR 'DIAL' COMMAND, SET VDEVENAB
D1
EXIT - SVC 12
DMKDIACP
A2
DMKDIACP
ENTERED VIA SVC FROM DMKCFM FOR 'COUPLE' COMMAND
C2
DMKSCNFD
CALL - GET FIRST DEVICE ADDRESS
ERR
01 H3
O.K.
D2
DMKCVTHB
CALL - CONVERT ADDRESS TO BINARY
ERR
01 H3
O.K.
E2
IS THE FIRST ADDR VALID
NO
01 H3
YES
F2
DMKSCNFD
CALL DMKSCNFD FIND USERID OR OPTIONAL 'TO'
ERR
01 D3
O.K.
G2
FIELD = 'TO'
NO
YES
H2
DMKSCNFD
CALL DMKSCNFD FIND USERID FIELD
ERR
01 D3
O.K.
DIACPUSR
J2
WRAP CONNECTION TO SAME VM
NO
YES
DIACPOTH
J3
DMKSCNAU
CALL - FIND VMBLOK FOR GIVEN USERID
O.K.
UNKN
01 D1
INVL
09 A1
K2
SET FLAG FOR WRAP TO SINGLE VIRT MACHINE
A
DIACPAD2
A4
DMKSCNFD
CALL DMKSCNFD LOCATE SECOND DEVICE ADDRESS
ERR
01 H3
O.K.
B4
DMKCVTHB
CALL - CONVERT ADDRESS TO BINARY
ERR
01 H3
O.K.
C4
IS THE SECOND ADDR VALID
NO
01 H3
YES
D4
DMKSCNVU
CALL - LOCATE X-SIDE VDEVBLOK
ERR
09 A1
O.K.
E4
IS DEVICE A VIRTUAL CTCA
NO
YES
BADVADD2
E5
DMKCVTBH
CALL DMKCVTBH CONVERT ADDRESS TO EBCDIC
F5
MSG= DMKDIA006E INVALID DEVICE TYPE - VADDR
02 D1
F4
DMKCFPRD
CALL - RESET THE X-SIDE CTCA
G4
DMKVCARS
CALL - DROP ANY PREVIOUS CONNECTION
H4
SWAP CPU TIMERS AND VMBLOKS TO Y-SIDE VM
J4
DMKSCNVU
CALL - LOCATE VCUBLOK AND VDEVBLOK
O.K.
09 A2
ERR
K4
WRAP CONNECTION TO SAME VM
NO
01 J3
YES
09 A1

| DMKDIA -- Process DIAL Command, Connect to a Virtual System (Parts 7 and 8 of 9)

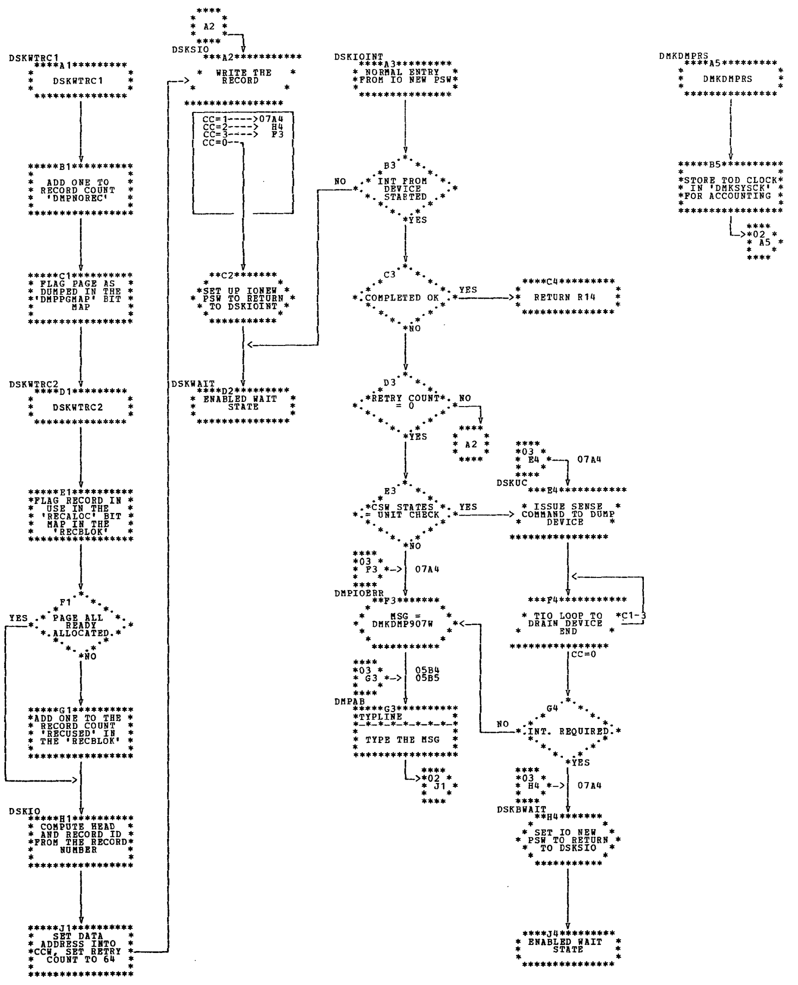
| DMKDIA -- Process DIAL Command, Connect to a Virtual System (Part 9 of 9)

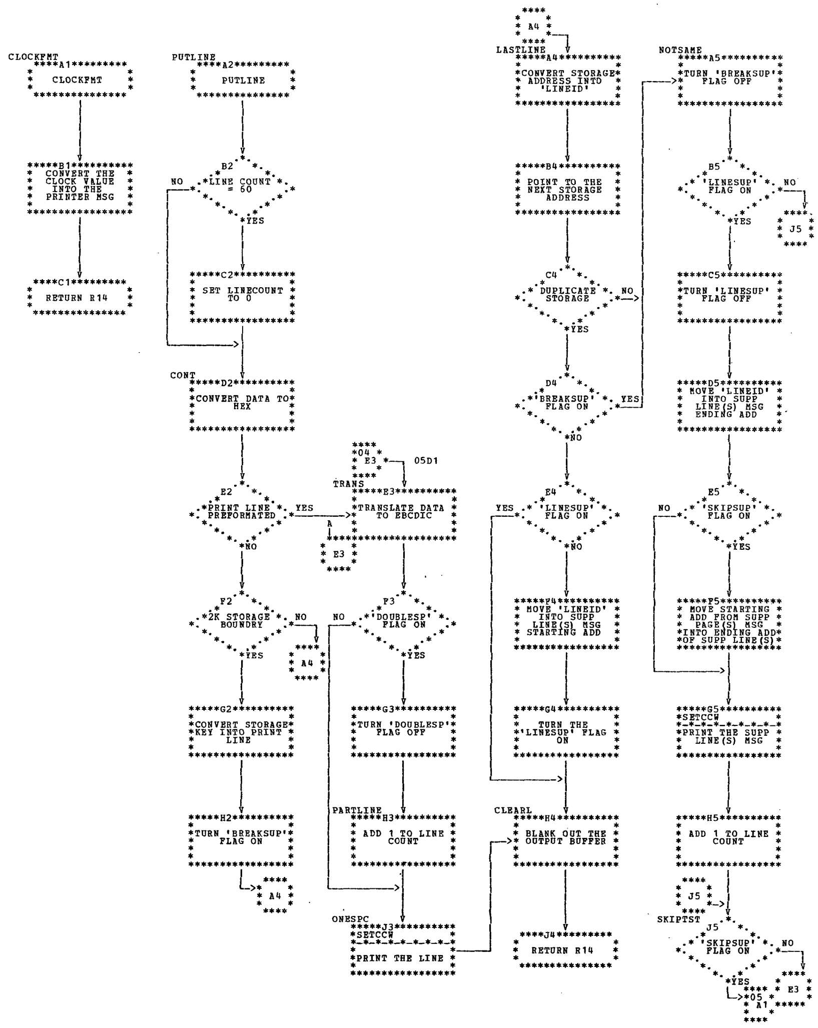

09 A1 08D4 08J3 08K4
UNKNWN2
A1 DMKCVTBH
CALL - CONVERT ADDRESS TO EBCDIC
B1
MSG=DMKDIA040E VADDR DOES NOT EXIST
02 D1
09 A2 08J4
DIACPREM
A2
IS DEVICE A VIRTUAL CTCA
NO
YES
B2
IS THE CTCA ALREADY IN USE
YES
NO
C2
ATTEMPT TO WRAP TO SAME DEVICE
YES
NO
A4
09 A3 01K2
BADVADD
A3 DMKCVTBH
CALL DMKCVTBH CONVERT ADDRESS TO EBCDIC
B3
MSG= DMKDIA011E INVALID DEVICE TYPE - USERID VADDR
02 D1
CTCBUSY
C3 DMKCVTBH
CALL DMKCVTBH CONVERT ADDRESS TO EBCDIC
D3
MSG= DMKDIA058E CTCA XXX BUSY ON USERID
02 D1
A4
A4 DMKFREE
CALL DMKFREE FREE STORAGE FOR CHBLOKS
B4
CONNECT CHX, CHYBLOK TO VDEVBLOKS MARK BOTH DEV'S READY
C4 DMKFREE
CALL DMKFREE FREE STORAGE FOR MSG BUFFER
D4 DMKCVTBH
CALL - CONVERT FIRST ADDRESS TO EBCDIC
E4 DMKCVTBH
CALL - CONVERT SECOND ADDRESS TO EBCDIC
F4
BUILD 'CTCA XXX COUPLE TO USERID XXX' MSG
G4 DMKQCNWT
CALL - SEND MESSAGE TO X-SIDE USER
H4
SWITCH DEV ADDRS IN MSG, CHANGE USERID TO X-SIDE USER
J4 DMKQCNWT
CALL - SEND MESSAGE TO Y-SIDE USER
K4
EXIT - SVC 12 RETURN TO CALLER

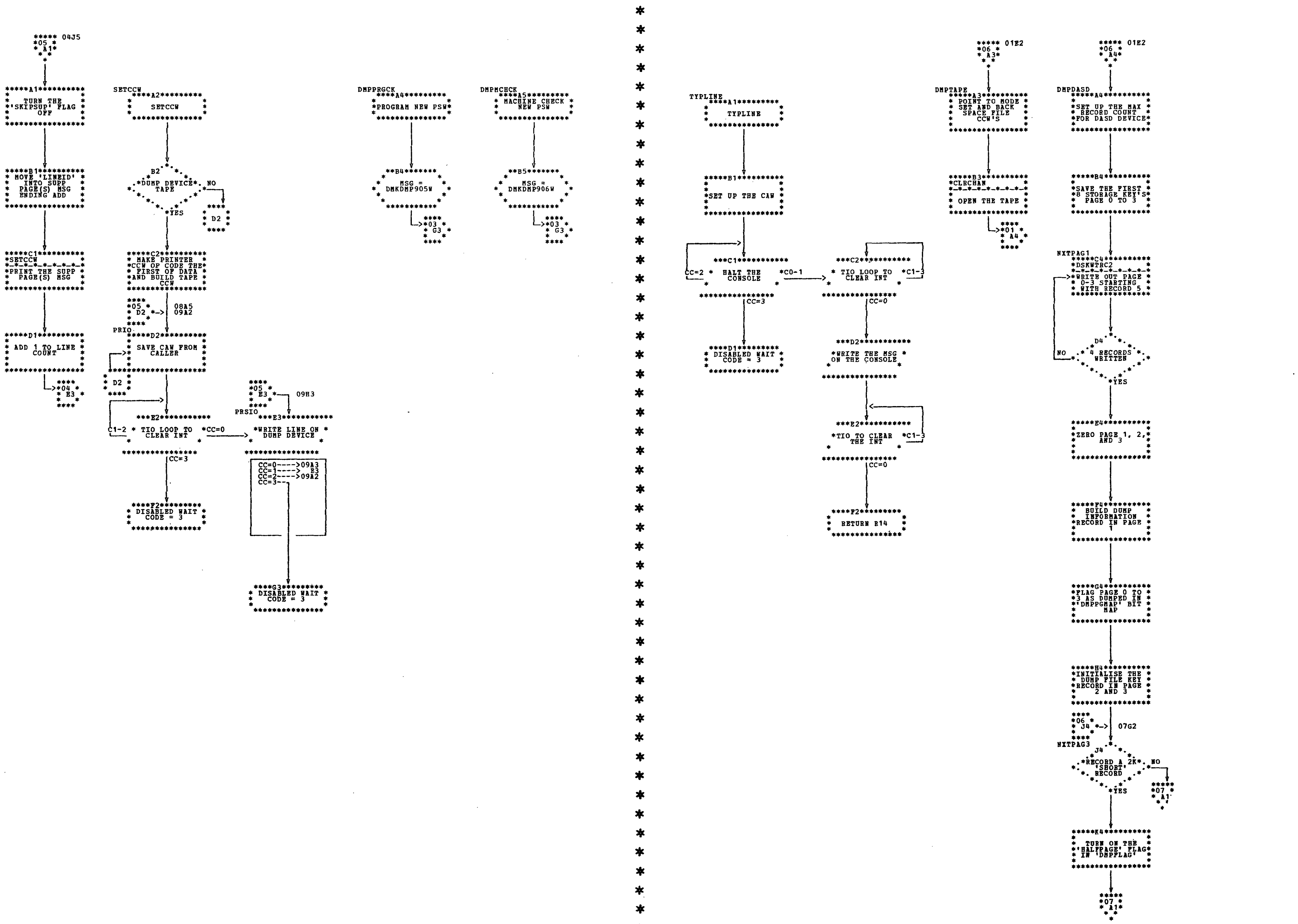


| DMKDMP -- System Dump Processor (Parts 1 and 2 cf 9)

| DMKDMP -- System Dump Prccessor (Parts 3 and 4 of 9)

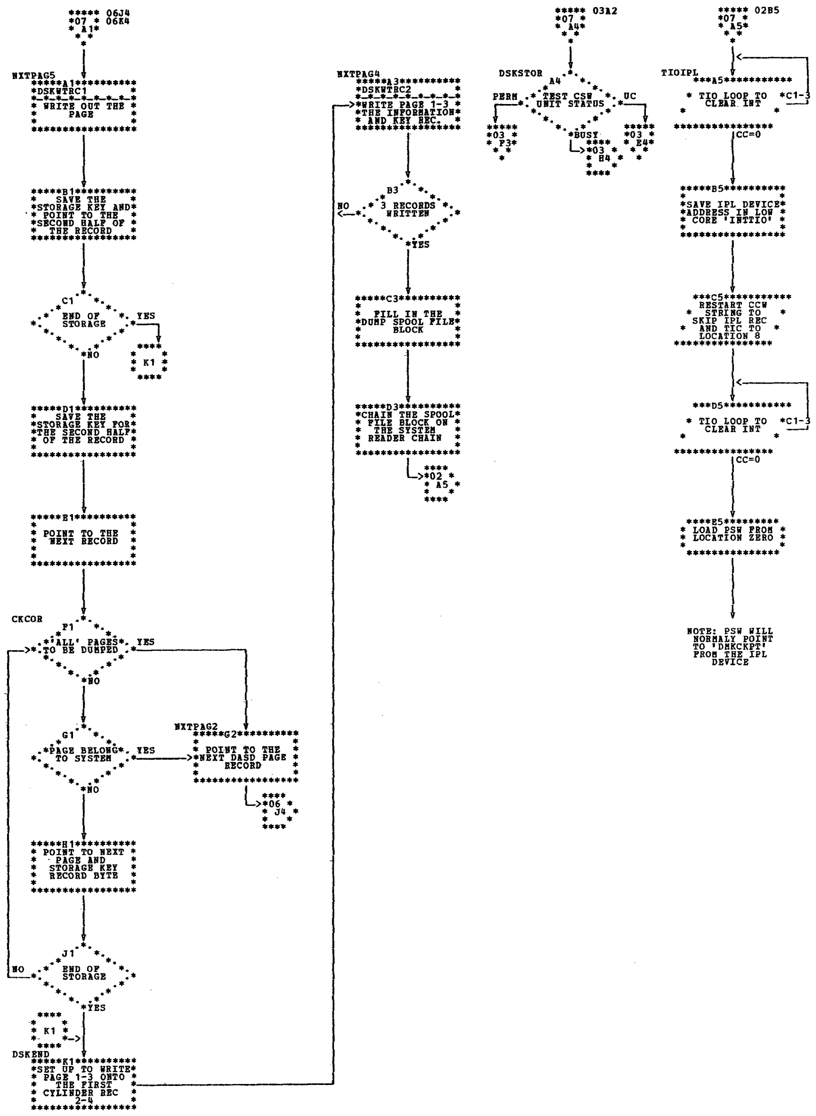

DSKWTRC1
A1
DSKWTRC1
B1
ADD ONE TO RECORD COUNT 'DMPNOREC'
C1
FLAG PAGE AS DUMPED IN THE 'DMPPGMAP' BIT MAP
DSKWTRC2
D1
DSKWTRC2
E1
FLAG RECORD IN USE IN THE 'RECALOC' BIT MAP IN THE 'RECBLOK'
F1
PAGE ALL READY ALLOCATED
YES
NO
G1
ADD ONE TO THE RECORD COUNT 'RECUSED' IN THE 'RECBLOK'
DSKIO
H1
COMPUTE HEAD AND RECORD ID FROM THE RECORD NUMBER
J1
SET DATA ADDRESS INTO CCW, SET RETRY COUNT TO 64
A2
DSKSIO
A2
WRITE THE RECORD
CC=1---->07A4
CC=2----> H4
CC=3----> F3
CC=0
C2
SET UP IONEW PSW TO RETURN TO DSKIOINT
DSKWAIT
D2
ENABLED WAIT STATE
DSKIOINT
A3
NORMAL ENTRY FROM IO NEW PSW
B3
INT FROM DEVICE STARTED
NO
YES
C3
COMPLETED OK
YES
NO
C4
RETURN R14
D3
RETRY COUNT = 0
NO
YES
A2
03
E4
07A4
DSKUC
E4
ISSUE SENSE COMMAND TO DUMP DEVICE
E3
CSW STATES = UNIT CHECK
YES
NO
03
F3
07A4
DMPIOERR
F3
MSG = DMKDMP907W
F4
TIO LOOP TO DRAIN DEVICE END
C1-3
CC=0
03
G3
05B4
05B5
DMPAB
G3
TYPLINE
TYPE THE MSG
02
J1
G4
INT. REQUIRED
NO
YES
03
H4
07A4
DSKBWAIT
H4
SET IO NEW PSW TO RETURN TO DSKSIO
J4
ENABLED WAIT STATE
DMKDMPRS
A5
DMKDMPRS
B5
STORE TOD CLOCK IN 'DMKSYSCK' FOR ACCOUNTING
02
A5

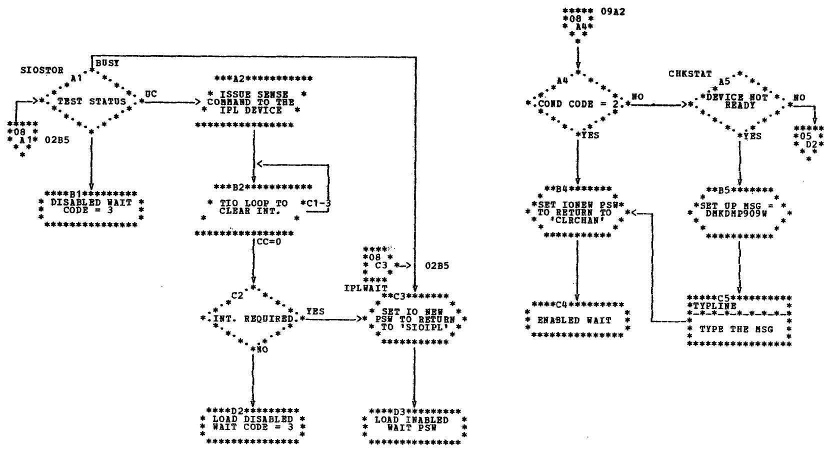


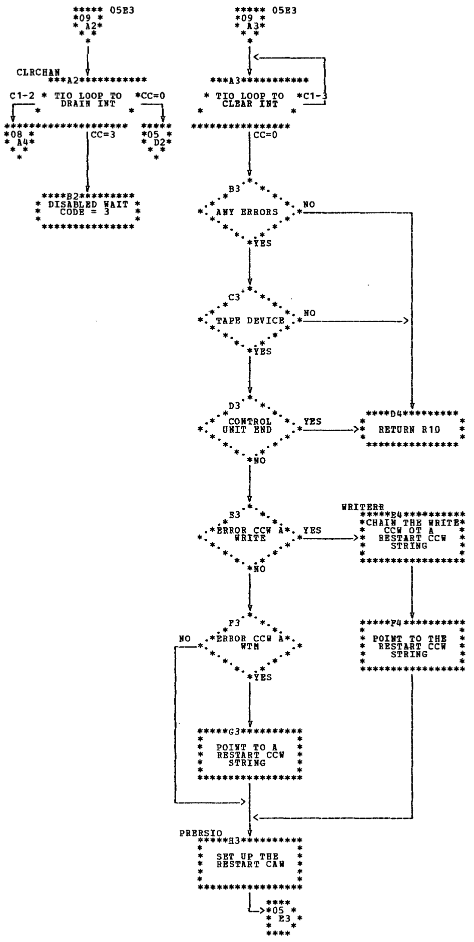

05 A1 04J5
A1 TURN THE 'SKIPSUP' FLAG OFF
B1 MOVE 'LINEID' INTO SUPP PAGE(S) MSG ENDING ADD
C1 SETCCW PRINT THE SUPP PAGE(S) MSG
D1 ADD 1 TO LINE COUNT
04 E3
SETCCW
A2 SETCCW
B2 DUMP DEVICE TAPE
NO
YES
D2
C2 MAKE PRINTER CCW OF CODE THE FIRST OF DATA AND BUILD TAPE CCW
05 D2 08A5 09A2
PRIO
D2 SAVE CAW FROM CALLER
E2 TIO LOOP TO CLEAR INT
C1-2
CC=0
CC=3
F2 DISABLED WAIT CODE = 3
05 E3 09H3
PRSIO
E3 WRITE LINE ON DUMP DEVICE
CC=0---->09A3
CC=1---->B3
CC=2---->09A2
CC=3
G3 DISABLED WAIT CODE = 3
DMPPRGCK
A4 PROGRAM NEW PSW
B4 MSG = DMKDMP905W
03 G3
DMPMCHCK
A5 MACHINE CHECK NEW PSW
B5 MSG = DMKDMP906W
03 G3
TYPLINE
A1 TYPLINE
B1 SET UP THE CAW
C1 HALT THE CONSOLE
CC=2
CC=3
C0-1
C2 TIO LOOP TO CLEAR INT
C1-3
CC=0
D1 DISABLED WAIT CODE = 3
D2 WRITE THE MSG ON THE CONSOLE
E2 TIO TO CLEAR THE INT
C1-3
CC=0
F2 RETURN R14
06 A3 01E2
DMPTAPE
A3 POINT TO MODE SET AND BACK SPACE FILE CCW'S
B3 CLRCHAN OPEN THE TAPE
01 A4
06 A4 01E2
DMPDASD
A4 SET UP THE MAX RECORD COUNT FOR DASD DEVICE
B4 SAVE THE FIRST 8 STORAGE KEY'S PAGE 0 TO 3
NXTPAG1
C4 DSKWTRC2 WRITE OUT PAGE 0-3 STARTING WITH RECORD 5
D4 4 RECORDS WRITTEN
NO
YES
E4 ZERO PAGE 1, 2, AND 3
F4 BUILD DUMP INFORMATION RECORD IN PAGE 1
G4 FLAG PAGE 0 TO 3 AS DUMPED IN 'DMPPGMAP' BIT MAP
H4 INITIALISE THE DUMP FILE KEY RECORD IN PAGE 2 AND 3
06 J4 07G2
NXTPAG3
J4 RECORD A 2K 'SHORT' RECORD
NO
YES
07 A1
K4 TURN ON THE 'HALFPAGE' FLAG IN 'DMPFLAG'
07 A1

| DMKDMP -- System Dump Processor (Parts 5 and 6 of 9)

| DMKDMP -- System Dump Processor (Parts 7 and 8 cf 9)

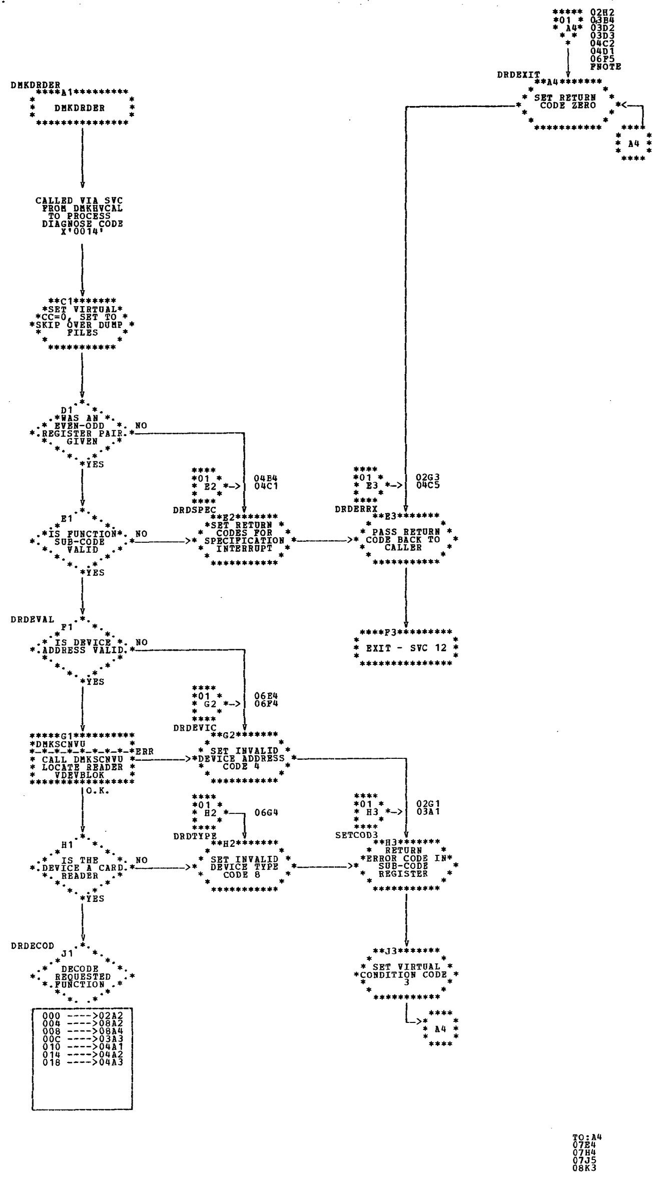

07 A1 06J4 06K4
NXTPAG5
A1
DSKWTRC1
WRITE OUT THE PAGE
B1
SAVE THE STORAGE KEY AND POINT TO THE SECOND HALF OF THE RECORD
C1
END OF STORAGE
YES
NO
K1
D1
SAVE THE STORAGE KEY FOR THE SECOND HALF OF THE RECORD
E1
POINT TO THE NEXT RECORD
CKCOR
F1
'ALL' PAGES TO BE DUMPED
YES
NO
NXTPAG2
G2
POINT TO THE NEXT DASD PAGE RECORD
06 J4
G1
PAGE BELONG TO SYSTEM
YES
NO
H1
POINT TO NEXT PAGE AND STORAGE KEY RECORD BYTE
J1
END OF STORAGE
NO
YES
K1
DSKEND
K1
SET UP TO WRITE PAGE 1-3 ONTO THE FIRST CYLINDER REC 2-4
NXTPAG4
A3
DSKWTRC2
WRITE PAGE 1-3 THE INFORMATION AND KEY REC.
B3
3 RECORDS WRITTEN
NO
YES
C3
FILL IN THE DUMP SPOOL FILE BLOCK
D3
CHAIN THE SPOOL FILE BLOCK ON THE SYSTEM READER CHAIN
02 A5
07 A4 03A2
DSKSTOR
A4
TEST CSW UNIT STATUS
PERM
UC
BUSY
03 F3
03 H4
03 E4
07 A5 02B5
TIOIPL
A5
TIO LOOP TO CLEAR INT
C1-3
CC=0
B5
SAVE IPL DEVICE ADDRESS IN LOW CORE 'INTTIO'
C5
RESTART CCW STRING TO SKIP IPL REC AND TIC TO LOCATION 8
D5
TIO LOOP TO CLEAR INT
C1-3
CC=0
E5
LOAD PSW FROM LOCATION ZERO
NOTE: PSW WILL NORMALY POINT TO 'DMKCKPT' FROM THE IPL DEVICE

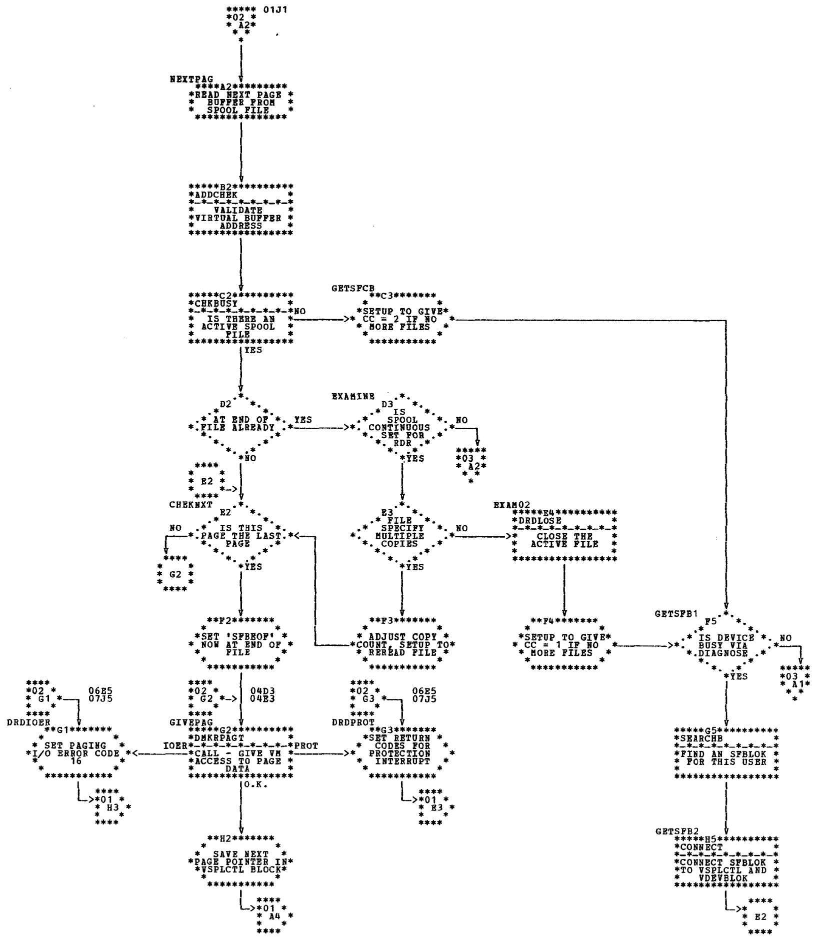


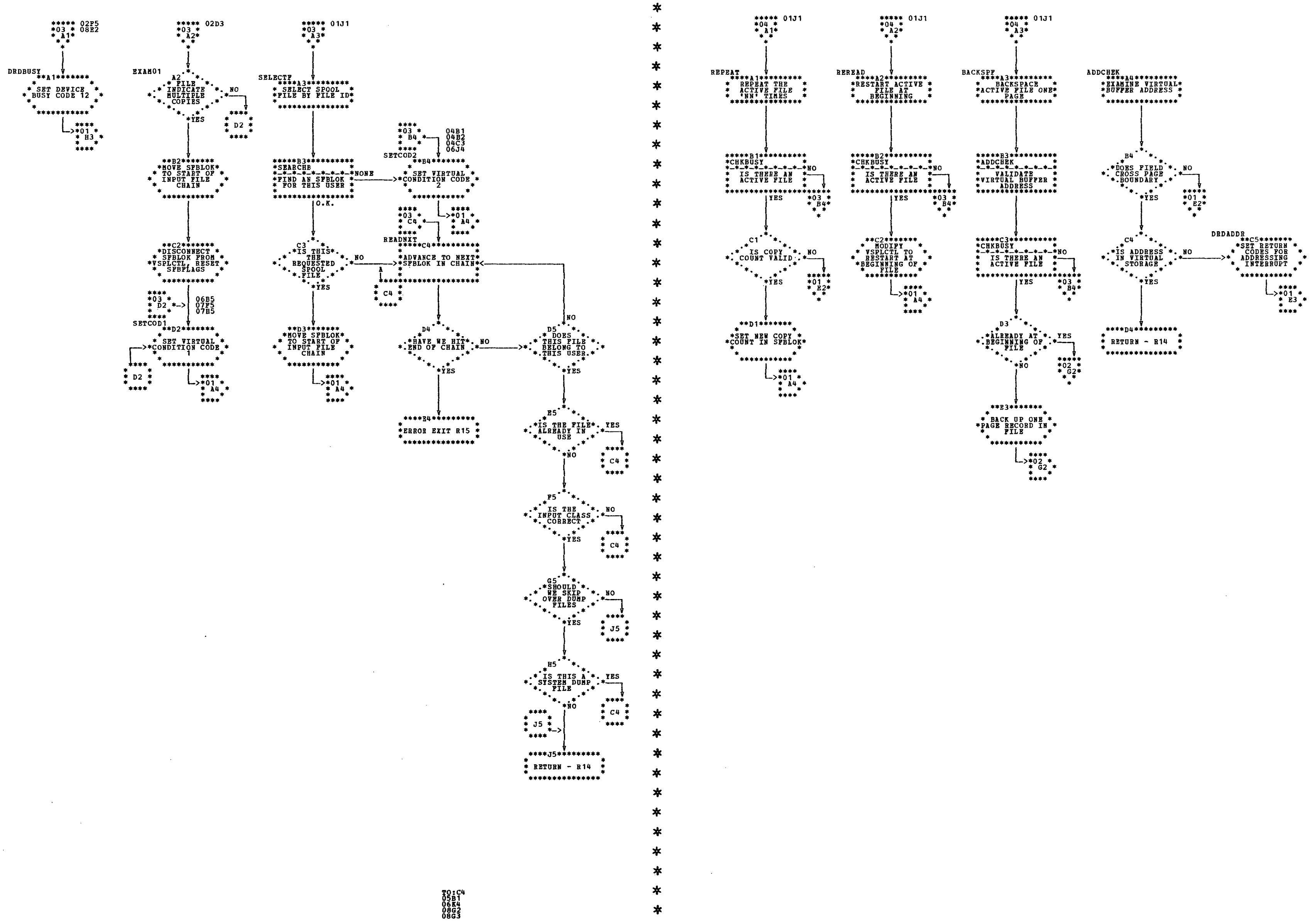


| DMKDMP -- System Dump Processor (Part 9 of 9)

DMKDRD -- Process Input Spool File (Parts 1 and 2 of 8)

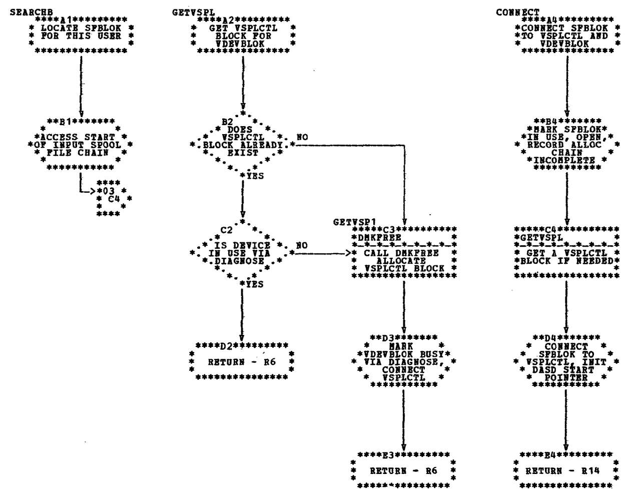

01 A4 02H2 03B4 03D2 03D3 04C2 04D1 06F5 FNOTE
DRDEXIT
A4 SET RETURN CODE ZERO
A4
DMKDRDER
A1 DMKDRDER
CALLED VIA SVC FROM DMKHVCAL TO PROCESS DIAGNOSE CODE X'0014'
C1 SET VIRTUAL CC=0, SET TO SKIP OVER DUMP FILES
D1 WAS AN EVEN-ODD REGISTER PAIR GIVEN
NO
YES
01 E2 04B4 04C1
DRDSPEC
E2 SET RETURN CODES FOR SPECIFICATION INTERRUPT
01 E3 02G3 04C5
DRDERRX
E3 PASS RETURN CODE BACK TO CALLER
E1 IS FUNCTION SUB-CODE VALID
NO
YES
F3 EXIT - SVC 12
DRDEVAL
F1 IS DEVICE ADDRESS VALID
NO
YES
01 G2 06E4 06F4
DRDEVIC
G2 SET INVALID DEVICE ADDRESS CODE 4
G1 DMKSCNVU
CALL DMKSCNVU LOCATE READER VDEVBLOK
ERR
O.K.
01 H2 06G4
DRDTYPE
H2 SET INVALID DEVICE TYPE CODE 8
01 H3 02G1 03A1
SETCOD3
H3 RETURN ERROR CODE IN SUB-CODE REGISTER
H1 IS THE DEVICE A CARD READER
NO
YES
DRDECOD
J1 DECODE REQUESTED FUNCTION
000 ---->02A2
004 ---->08A2
008 ---->08A4
00C ---->03A3
010 ---->04A1
014 ---->04A2
018 ---->04A3
J3 SET VIRTUAL CONDITION CODE 3
A4
TO:A4
07E4
07H4
07J5
08K3

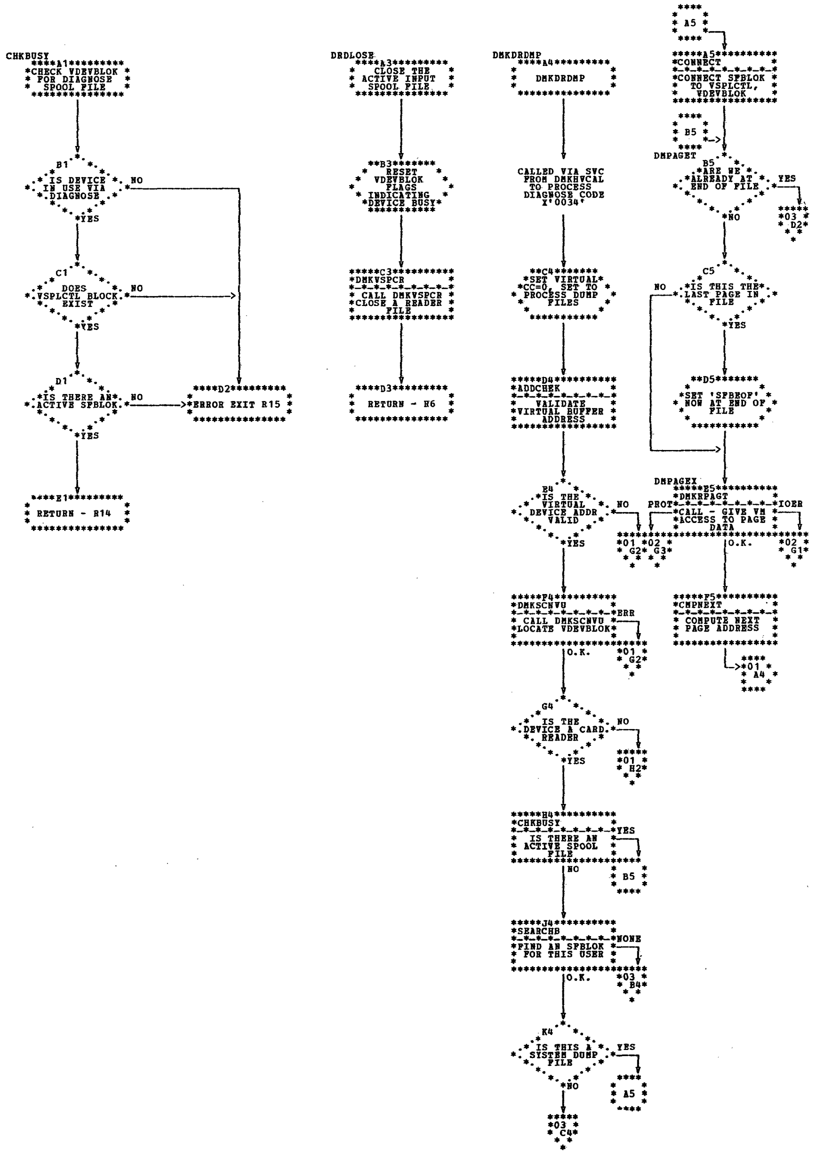

02 A2 01J1
NEXTPAG
A2 READ NEXT PAGE BUFFER FROM SPOOL FILE
B2 ADDCHEK
VALIDATE VIRTUAL BUFFER ADDRESS
C2 CHKBUSY
IS THERE AN ACTIVE SPOOL FILE
NO
YES
GETSFCB
C3 SETUP TO GIVE CC = 2 IF NO MORE FILES
D2 AT END OF FILE ALREADY
YES
NO
EXAMINE
D3 IS SPOOL CONTINUOUS SET FOR RDR
NO
YES
03 A2
E2
CHEKNXT
E2 IS THIS PAGE THE LAST PAGE
NO
YES
G2
E3 FILE SPECIFY MULTIPLE COPIES
NO
YES
EXAM02
E4 DRDLOSE
CLOSE THE ACTIVE FILE
F2 SET 'SFBEOF' NOW AT END OF FILE
F3 ADJUST COPY COUNT, SETUP TO REREAD FILE
F4 SETUP TO GIVE CC = 1 IF NO MORE FILES
GETSFB1
F5 IS DEVICE BUSY VIA DIAGNOSE
NO
YES
03 A1
02 G1 06E5 07J5
DRDIOER
G1 SET PAGING I/O ERROR CODE 16
02 G2 04D3 04E3
GIVEPAG
G2 DMKRPAGT
CALL - GIVE VM ACCESS TO PAGE DATA
IOER
PROT
O.K.
02 G3 06E5 07J5
DRDPROT
G3 SET RETURN CODES FOR PROTECTION INTERRUPT
01 H3
01 E3
G5 SEARCHB
FIND AN SFBLOK FOR THIS USER
H2 SAVE NEXT PAGE POINTER IN VSPLCTL BLOCK
01 A4
GETSFB2
H5 CONNECT
CONNECT SFBLOK TO VSPLCTL AND VDEVBLOK
E2

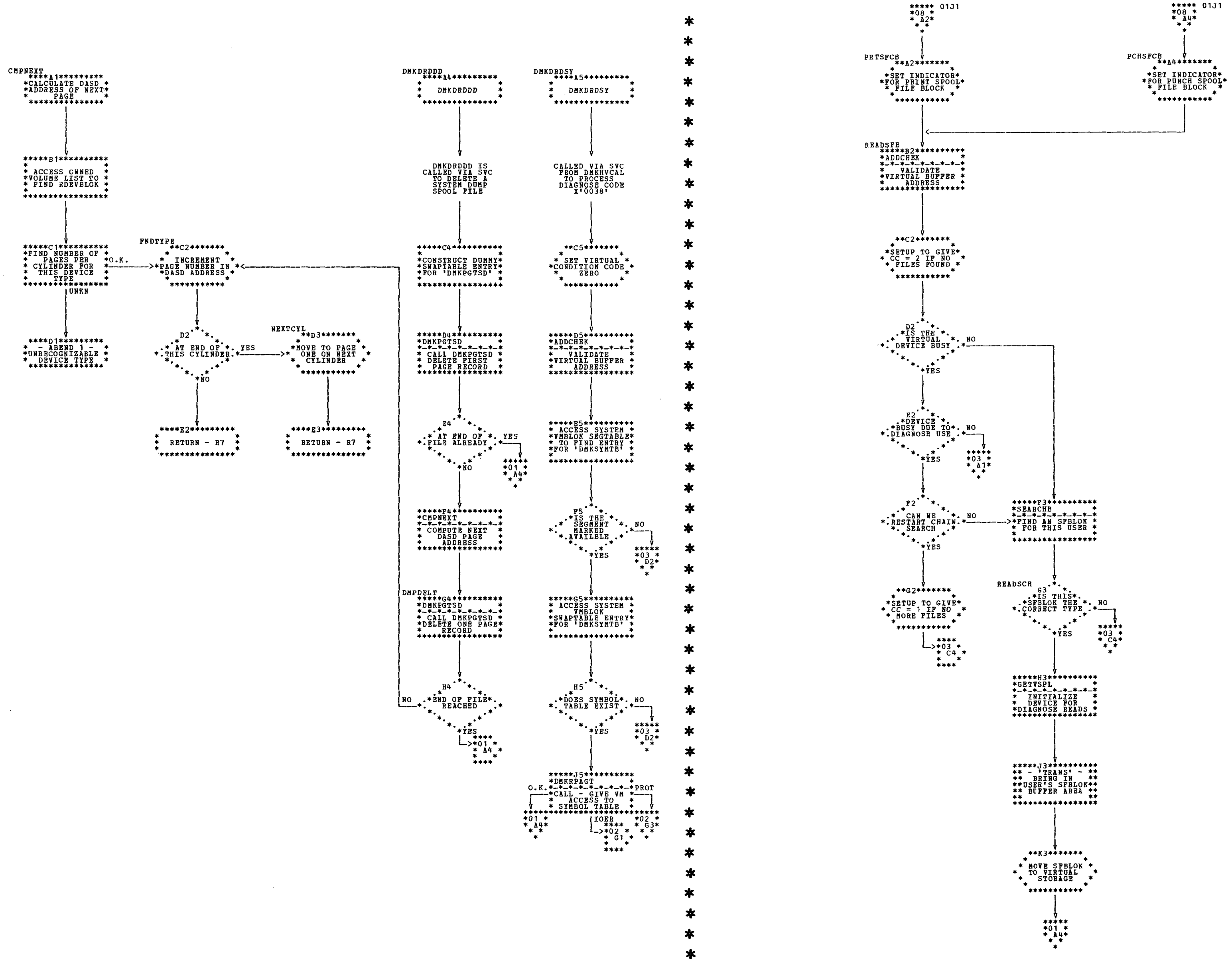

03 A1 02F5 08E2
DRDBUSY
A1
SET DEVICE BUSY CODE 12
01 H3
03 A2 02D3
EXAM01
A2
FILE INDICATE MULTIPLE COPIES
NO
D2
YES
B2
MOVE SFBLOK TO START OF INPUT FILE CHAIN
C2
DISCONNECT SFBLOK FROM VSPLCTL, RESET SFBFLAGS
03 D2
06B5
07F5
07H5
SETCOD1
D2
SET VIRTUAL CONDITION CODE 1
D2
01 A4
03 A3 01J1
SELECTF
A3
SELECT SPOOL FILE BY FILE ID
B3
SEARCHB
FIND AN SFBLOK FOR THIS USER
NONE
O.K.
C3
IS THIS THE REQUESTED SPOOL FILE
NO
YES
A
C4
D3
MOVE SFBLOK TO START OF INPUT FILE CHAIN
01 A4
03 B4
04B1
04B2
04C3
06J4
SETCOD2
B4
SET VIRTUAL CONDITION CODE 2
01 A4
03 C4
READNXT
C4
ADVANCE TO NEXT SFBLOK IN CHAIN
D4
HAVE WE HIT END OF CHAIN
NO
YES
E4
ERROR EXIT R15
D5
DOES THIS FILE BELONG TO THIS USER
NO
YES
E5
IS THE FILE ALREADY IN USE
YES
C4
NO
F5
IS THE INPUT CLASS CORRECT
NO
C4
YES
G5
SHOULD WE SKIP OVER DUMP FILES
NO
J5
YES
H5
IS THIS A SYSTEM DUMP FILE
YES
C4
NO
J5
J5
RETURN - R14
TO:C4
05B1
06K4
08G2
08G3
04 A1 01J1
REPEAT
A1
REPEAT THE ACTIVE FILE 'NN' TIMES
B1
CHKBUSY
IS THERE AN ACTIVE FILE
NO
03 B4
YES
C1
IS COPY COUNT VALID
NO
01 E2
YES
D1
SET NEW COPY COUNT IN SFBLOK
01 A4
04 A2 01J1
REREAD
A2
RESTART ACTIVE FILE AT BEGINNING
B2
CHKBUSY
IS THERE AN ACTIVE FILE
NO
03 B4
YES
C2
MODIFY VSPLCTL TO RESTART AT BEGINNING OF FILE
01 A4
04 A3 01J1
BACKSPF
A3
BACKSPACE ACTIVE FILE ONE PAGE
B3
ADDCHEK
VALIDATE VIRTUAL BUFFER ADDRESS
C3
CHKBUSY
IS THERE AN ACTIVE FILE
NO
03 B4
YES
D3
ALREADY AT BEGINNING OF FILE
YES
02 G2
NO
E3
BACK UP ONE PAGE RECORD IN FILE
02 G2
ADDCHEK
A4
EXAMINE VIRTUAL BUFFER ADDRESS
B4
DOES FIELD CROSS PAGE BOUNDARY
NO
01 E2
YES
C4
IS ADDRESS IN VIRTUAL STORAGE
NO
YES
DRDADDR
C5
SET RETURN CODES FOR ADDRESSING INTERRUPT
01 E3
D4
RETURN - R14

DMKDRD -- Process Input Spool File (Parts 3 and 4 of 8)

DMKDRD -- Process Input Spool File (Parts 5 and 6 of 8)

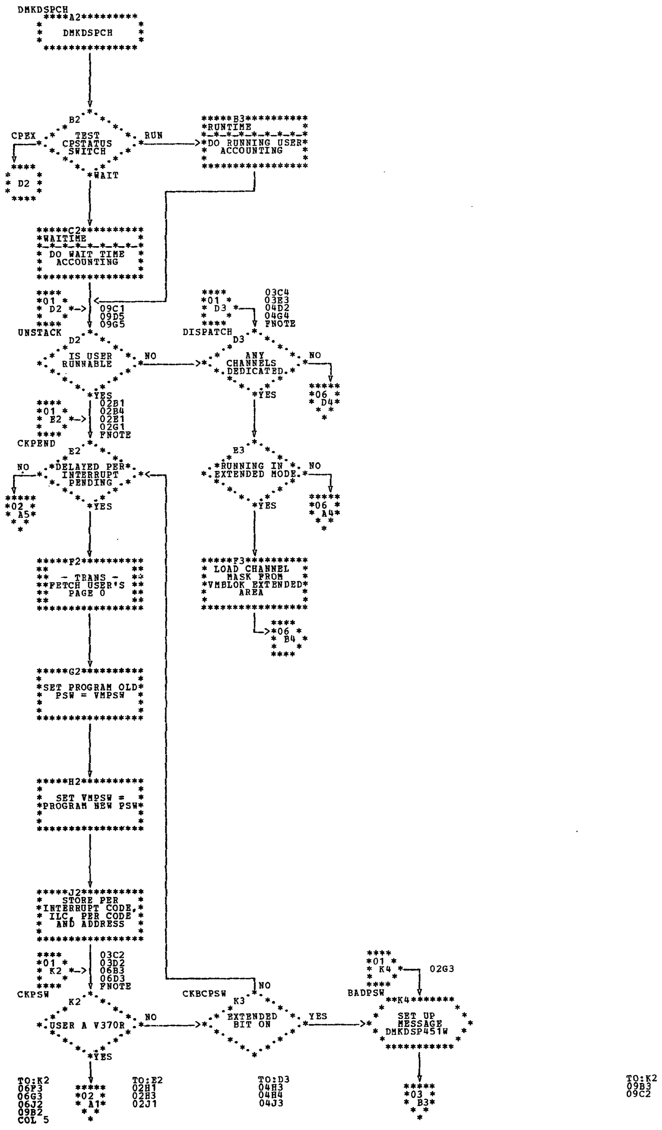

SEARCHB
A1
LOCATE SFBLOK FOR THIS USER
B1
ACCESS START OF INPUT SPOOL FILE CHAIN
03 C4
GETVSPL
A2
GET VSPLCTL BLOCK FOR VDEVBLOK
B2
DOES VSPLCTL BLOCK ALREADY EXIST
NO
YES
C2
IS DEVICE IN USE VIA DIAGNOSE
NO
YES
D2
RETURN - R6
GETVSP1
C3
DMKFREE
CALL DMKFREE ALLOCATE VSPLCTL BLOCK
D3
MARK VDEVBLOK BUSY VIA DIAGNOSE, CONNECT VSPLCTL
E3
RETURN - R6
CONNECT
A4
CONNECT SFBLOK TO VSPLCTL AND VDEVBLOK
B4
MARK SFBLOK IN USE, OPEN, RECORD ALLOC CHAIN INCOMPLETE
C4
GETVSPL
GET A VSPLCTL BLOCK IF NEEDED
D4
CONNECT SFBLOK TO VSPLCTL, INIT DASD START POINTER
E4
RETURN - R14

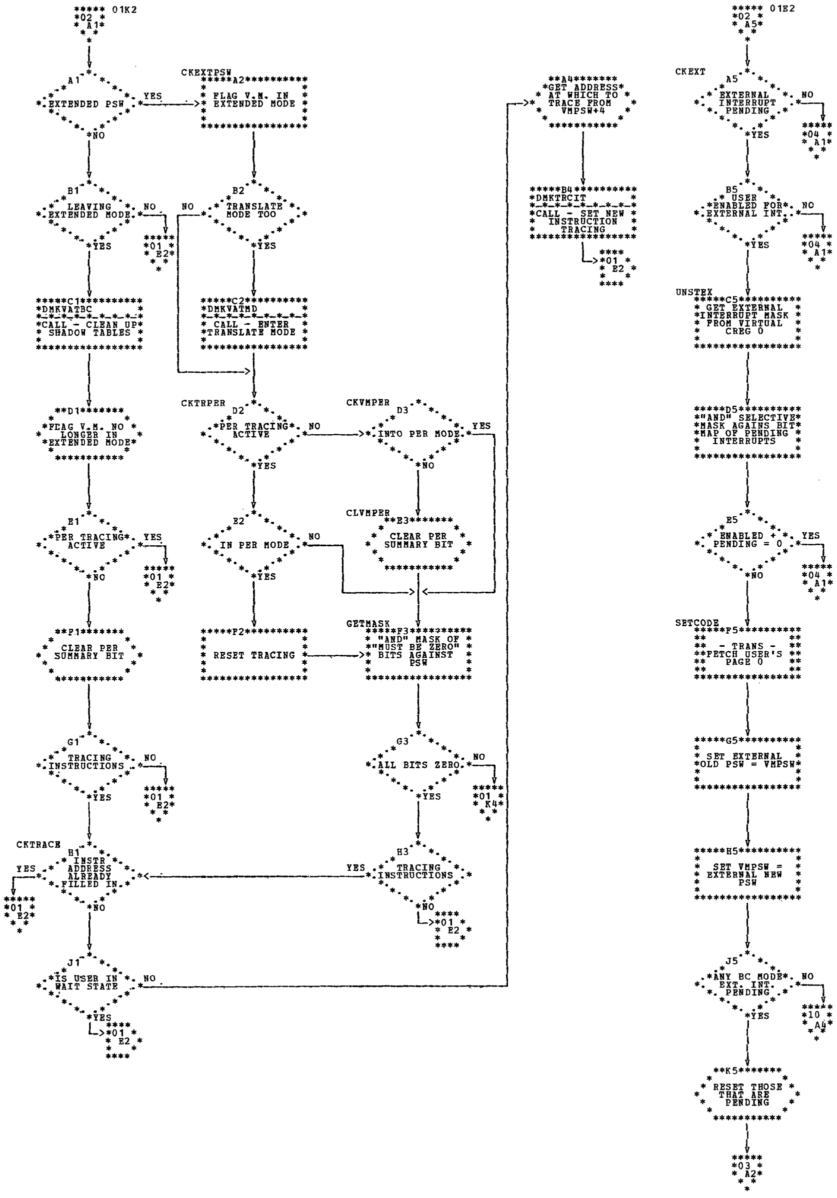

CHKBUSY
A1
CHECK VDEVBLOK FOR DIAGNOSE SPOOL FILE
B1
IS DEVICE IN USE VIA DIAGNOSE
NO
YES
C1
DOES VSPLCTL BLOCK EXIST
NO
YES
D1
IS THERE AN ACTIVE SFBLOK
NO
YES
D2
ERROR EXIT R15
E1
RETURN - R14
DRDLOSE
A3
CLOSE THE ACTIVE INPUT SPOOL FILE
B3
RESET VDEVBLOK FLAGS INDICATING DEVICE BUSY
C3
DMKVSPCR
CALL DMKVSPCR CLOSE A READER FILE
D3
RETURN - R6
DMKDRDMP
A4
DMKDRDMP
CALLED VIA SVC FROM DMKHVCAL TO PROCESS DIAGNOSE CODE X'0034'
C4
SET VIRTUAL CC=0, SET TO PROCESS DUMP FILES
D4
ADDCHEK
VALIDATE VIRTUAL BUFFER ADDRESS
E4
IS THE VIRTUAL DEVICE ADDR VALID
NO
YES
01 G2
02 G3
F4
DMKSCNVU
ERR
CALL DMKSCNVU LOCATE VDEVBLOK
O.K.
01 G2
G4
IS THE DEVICE A CARD READER
NO
YES
01 H2
H4
CHKBUSY
YES
IS THERE AN ACTIVE SPOOL FILE
NO
B5
J4
SEARCHB
NONE
FIND AN SFBLOK FOR THIS USER
O.K.
03 B4
K4
IS THIS A SYSTEM DUMP FILE
YES
NO
A5
03 C4
A5
A5
CONNECT
CONNECT SFBLOK TO VSPLCTL, VDEVBLOK
B5
DMPAGET
B5
ARE WE ALREADY AT END OF FILE
YES
NO
03 D2
C5
IS THIS THE LAST PAGE IN FILE
NO
YES
D5
SET 'SFBEOF' NOW AT END OF FILE
DMPAGEX
E5
DMKRPAGT
PROT
IOER
CALL - GIVE VM ACCESS TO PAGE DATA
O.K.
02 G1
F5
CMPNEXT
COMPUTE NEXT PAGE ADDRESS
01 A4

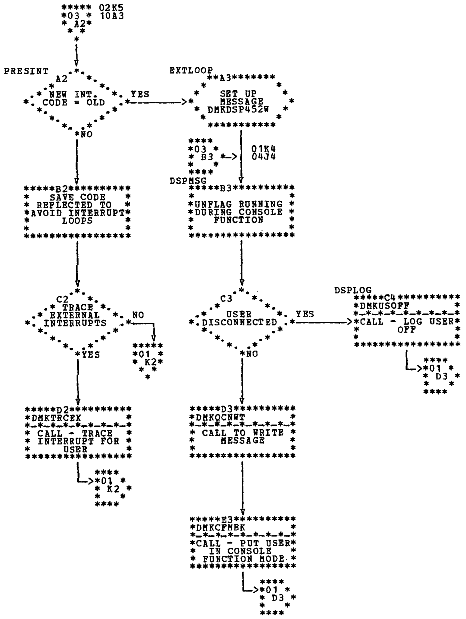

CMPNEXT
A1
CALCULATE DASD ADDRESS OF NEXT PAGE
B1
ACCESS CWNED VOLUME LIST TO FIND RDEVBLOK
C1
FIND NUMBER OF PAGES PER CYLINDER FOR THIS DEVICE TYPE
O.K.
UNKN
D1
- ABEND 1 - UNRECOGNIZABLE DEVICE TYPE
FNDTYPE
C2
INCREMENT PAGE NUMBER IN DASD ADDRESS
D2
AT END OF THIS CYLINDER
YES
NO
NEXTCYL
D3
MOVE TO PAGE ONE ON NEXT CYLINDER
E2
RETURN - R7
E3
RETURN - R7
DMKDRDDD
A4
DMKDRDDD
DMKDRDDD IS CALLED VIA SVC TO DELETE A SYSTEM DUMP SPOOL FILE
C4
CONSTRUCT DUMMY SWAPTABLE ENTRY FOR 'DMKPGTSD'
D4
DMKPGTSD
CALL DMKPGTSD DELETE FIRST PAGE RECORD
E4
AT END OF FILE ALREADY
YES
01 A4
NO
F4
CMPNEXT
COMPUTE NEXT DASD PAGE ADDRESS
DMPDELT
G4
DMKPGTSD
CALL DMKPGTSD DELETE ONE PAGE RECORD
H4
END OF FILE REACHED
NO
YES
01 A4
DMKDRDSY
A5
DMKDRDSY
CALLED VIA SVC FROM DMKHVCAL TO PROCESS DIAGNOSE CODE X'0038'
C5
SET VIRTUAL CONDITION CODE ZERO
D5
ADDCHEK
VALIDATE VIRTUAL BUFFER ADDRESS
E5
ACCESS SYSTEM VMBLOK SEGTABLE TO FIND ENTRY FOR 'DMKSYMTB'
F5
IS THE SEGMENT MARKED AVAILBLE
NO
03 D2
YES
G5
ACCESS SYSTEM VMBLOK SWAPTABLE ENTRY FOR 'DMKSYMTB'
H5
DOES SYMBOL TABLE EXIST
NO
03 D2
YES
J5
DMKRPAGT
O.K.
PROT
CALL - GIVE VM ACCESS TO SYMBOL TABLE
01 A4
IOER
02 G1
02 G3
08 A2
01J1
08 A4
01J1
PRTSFCB
A2
SET INDICATOR FOR PRINT SPOOL FILE BLOCK
PCHSFCB
A4
SET INDICATOR FOR PUNCH SPOOL FILE BLOCK
READSFB
B2
ADDCHEK
VALIDATE VIRTUAL BUFFER ADDRESS
C2
SETUP TO GIVE CC = 2 IF NO FILES FOUND
D2
IS THE VIRTUAL DEVICE BUSY
NO
YES
E2
DEVICE BUSY DUE TO DIAGNOSE USE
NO
03 A1
YES
F2
CAN WE RESTART CHAIN SEARCH
NO
YES
F3
SEARCHB
FIND AN SFBLOK FOR THIS USER
G2
SETUP TO GIVE CC = 1 IF NO MORE FILES
03 C4
READSCH
G3
IS THIS SFBLOK THE CORRECT TYPE
NO
03 C4
YES
H3
GETVSPL
INITIALIZE DEVICE FOR DIAGNOSE READS
J3
- 'TRANS' - BRING IN USER'S SFBLOK BUFFER AREA
K3
MOVE SFBLOK TO VIRTUAL STORAGE
01 A4

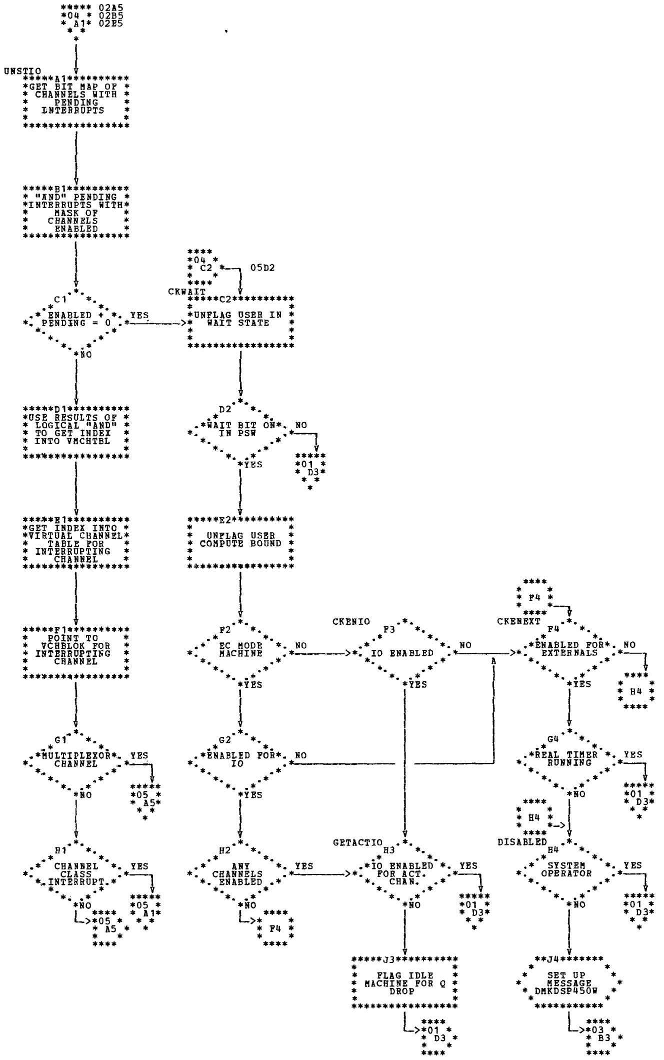
DMKDRD -- Process Input Spool File (Parts 7 and 8 of 8)

| DMKDSP -- Dispatcher (Parts 1 and 2 of 10)



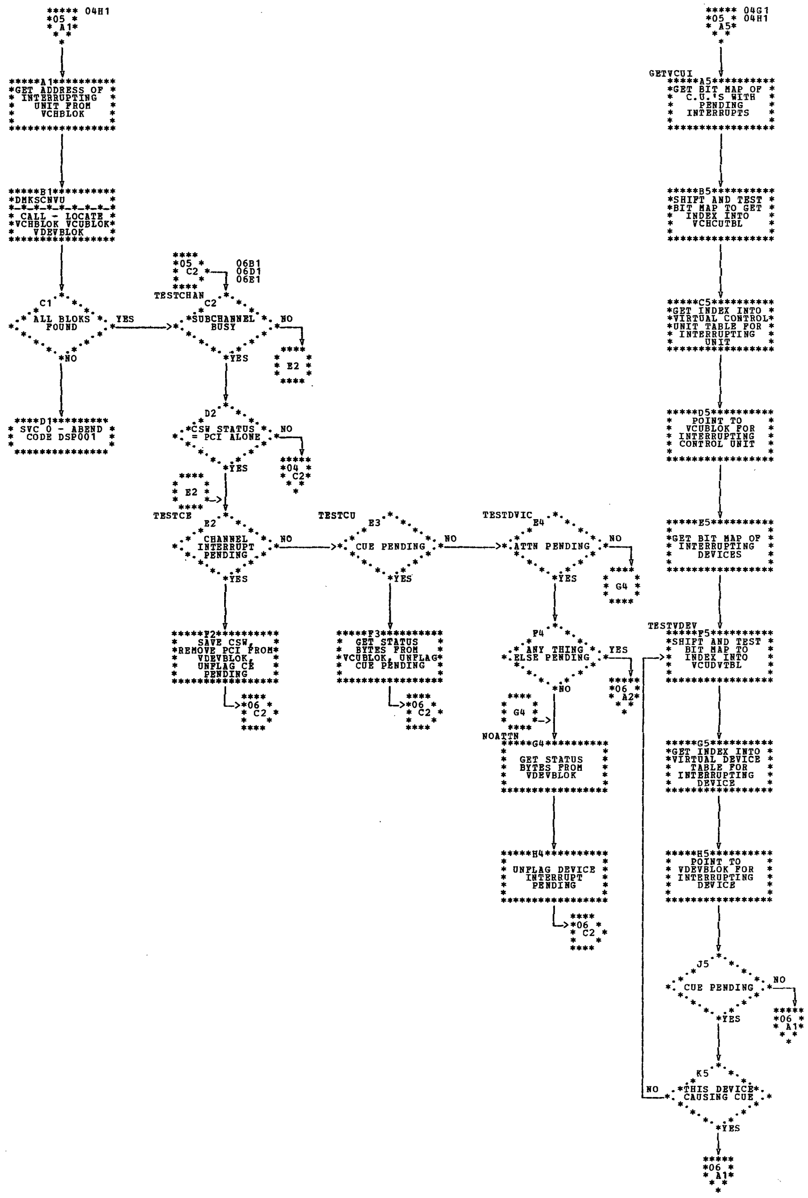


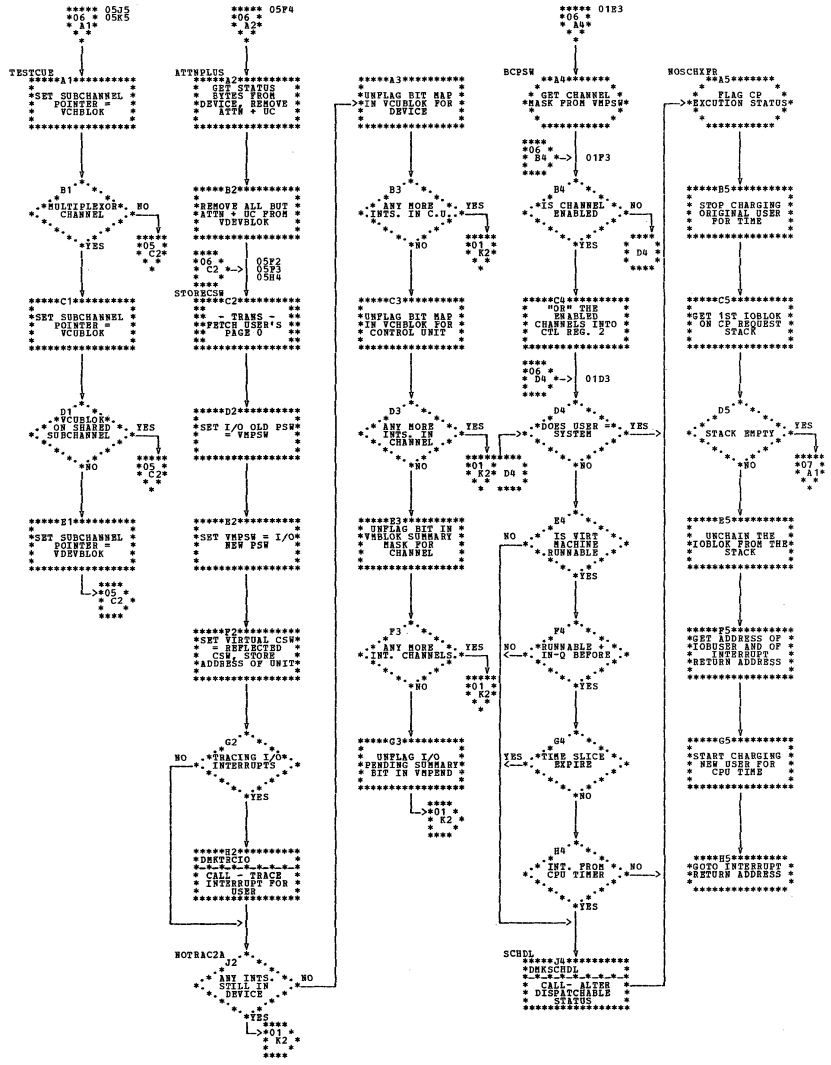


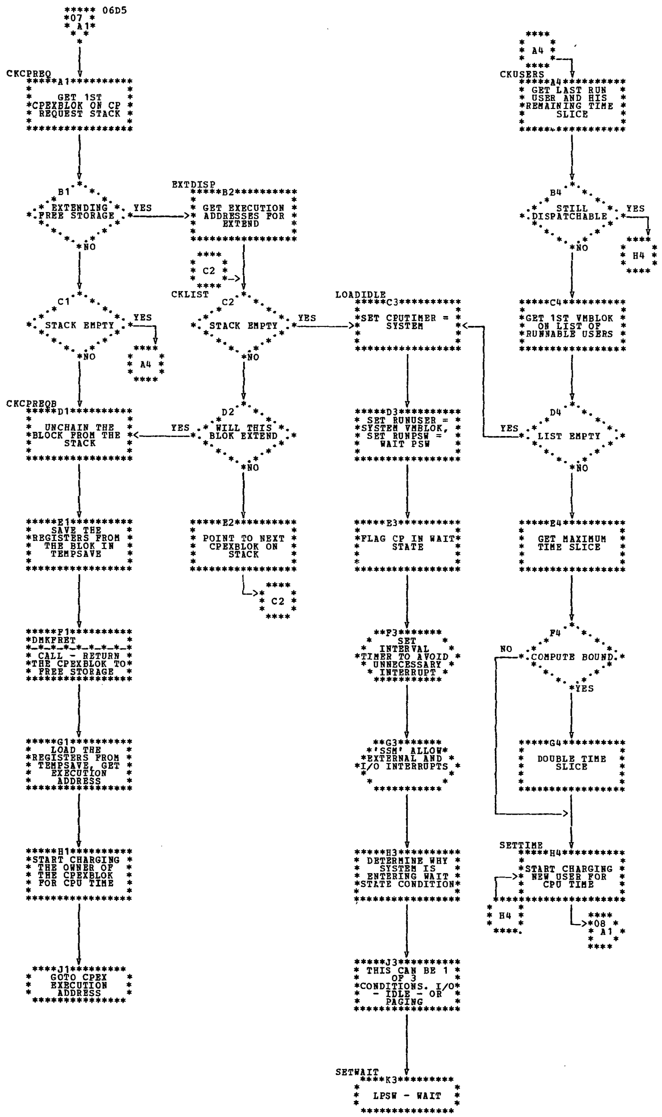


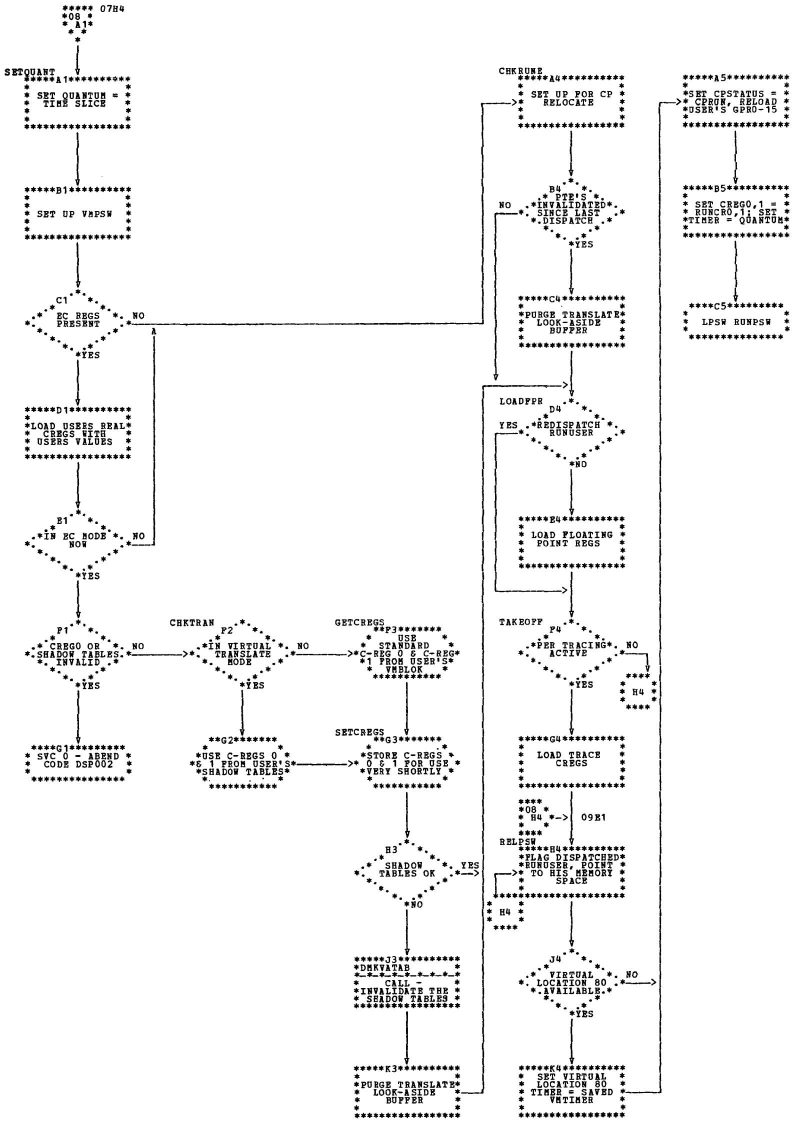
| DMKDSP -- Dispatcher (Parts 3 and 4 of 10)

| DMKDSP -- Dispatcher (Parts 5 and 6 of 10)


05 A1 04H1
A1 GET ADDRESS OF INTERRUPTING UNIT FROM VCHBLOK
B1 DMKSCNVU CALL - LOCATE VCHBLOK VCUBLOK VDEVBLOK
C1 ALL BLOKS FOUND
YES
NO
D1 SVC 0 - ABEND CODE DSP001
05 C2 06B1 06D1 06E1
TESTCHAN
C2 SUBCHANNEL BUSY
NO
E2
YES
D2 CSW STATUS = PCI ALONE
NO
04 C2
YES
E2
TESTCE
E2 CHANNEL INTERRUPT PENDING
NO
YES
F2 SAVE CSW, REMOVE PCI FROM VDEVBLOK, UNFLAG CE PENDING
06 C2
TESTCU
E3 CUE PENDING
NO
YES
F3 GET STATUS BYTES FROM VCUBLOK, UNFLAG CUE PENDING
06 C2
TESTDVIC
E4 ATTN PENDING
NO
G4
YES
F4 ANY THING ELSE PENDING
YES
06 A2
NO
G4
NOATTN
G4 GET STATUS BYTES FROM VDEVBLOK
H4 UNFLAG DEVICE INTERRUPT PENDING
06 C2
05 A5 04G1 04H1
GETVCUI
A5 GET BIT MAP OF C.U.'S WITH PENDING INTERRUPTS
B5 SHIFT AND TEST BIT MAP TO GET INDEX INTO VCHCUTBL
C5 GET INDEX INTO VIRTUAL CONTROL UNIT TABLE FOR INTERRUPTING UNIT
D5 POINT TO VCUBLOK FOR INTERRUPTING CONTROL UNIT
E5 GET BIT MAP OF INTERRUPTING DEVICES
TESTVDEV
F5 SHIFT AND TEST BIT MAP TO INDEX INTO VCUDVTBL
G5 GET INDEX INTO VIRTUAL DEVICE TABLE FOR INTERRUPTING DEVICE
H5 POINT TO VDEVBLOK FOR INTERRUPTING DEVICE
J5 CUE PENDING
NO
06 A1
YES
K5 THIS DEVICE CAUSING CUE
NO
YES
06 A1

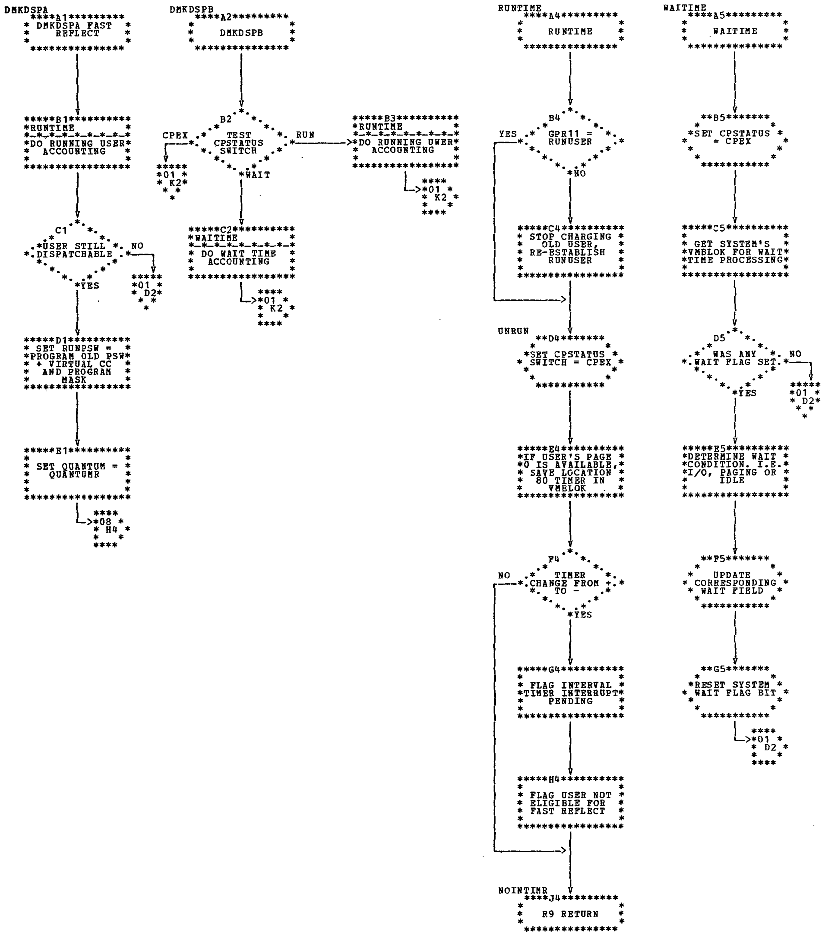

06 A1 05J5 05K5
TESTCUE
A1 SET SUBCHANNEL POINTER = VCHBLOK
B1 MULTIPLEXOR CHANNEL
NO
05 C2
YES
C1 SET SUBCHANNEL POINTER = VCUBLOK
D1 VCUBLOK ON SHARED SUBCHANNEL
YES
05 C2
NO
E1 SET SUBCHANNEL POINTER = VDEVBLOK
05 C2
06 A2 05F4
ATTNPLUS
A2 GET STATUS BYTES FROM DEVICE, REMOVE ATTN + UC
B2 REMOVE ALL BUT ATTN + UC FROM VDEVBLOK
06 C2 05F2 05F3 05H4
STORECSW
C2 - TRANS - FETCH USER'S PAGE 0
D2 SET I/O OLD PSW = VMPSW
E2 SET VMPSW = I/O NEW PSW
F2 SET VIRTUAL CSW = REFLECTED CSW, STORE ADDRESS OF UNIT
G2 TRACING I/O INTERRUPTS
NO
YES
H2 DMKTRCIO CALL - TRACE INTERRUPT FOR USER
NOTRAC2A
J2 ANY INTS. STILL IN DEVICE
NO
YES
01 K2
A3 UNFLAG BIT MAP IN VCUBLOK FOR DEVICE
B3 ANY MORE INTS. IN C.U.
YES
01 K2
NO
C3 UNFLAG BIT MAP IN VCHBLOK FOR CONTROL UNIT
D3 ANY MORE INTS. IN CHANNEL
YES
01 K2
D4
NO
E3 UNFLAG BIT IN VMBLOK SUMMARY MASK FOR CHANNEL
F3 ANY MORE INT. CHANNELS
YES
01 K2
NO
G3 UNFLAG I/O PENDING SUMMARY BIT IN VMPEND
01 K2
06 A4 01E3
BCPSW
A4 GET CHANNEL MASK FROM VMPSW
06 B4 01F3
B4 IS CHANNEL ENABLED
NO
D4
YES
C4 "OR" THE ENABLED CHANNELS INTO CTL REG. 2
06 D4 01D3
D4 DOES USER = SYSTEM
YES
NO
E4 IS VIRT MACHINE RUNNABLE
NO
YES
F4 RUNNABLE + IN-Q BEFORE
NO
YES
G4 TIME SLICE EXPIRE
YES
NO
H4 INT. FROM CPU TIMER
NO
YES
SCHDL
J4 DMKSCHDL CALL- ALTER DISPATCHABLE STATUS
NOSCHXFR
A5 FLAG CP EXCUTION STATUS
B5 STOP CHARGING ORIGINAL USER FOR TIME
C5 GET 1ST IOBLOK ON CP REQUEST STACK
D5 STACK EMPTY
YES
07 A1
NO
E5 UNCHAIN THE IOBLOK FROM THE STACK
F5 GET ADDRESS OF IOBUSER AND OF INTERRUPT RETURN ADDRESS
G5 START CHARGING NEW USER FOR CPU TIME
H5 GOTO INTERRUPT RETURN ADDRESS

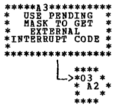

06D5
07 A1
CKCPREQ
A1 GET 1ST CPEXBLOK ON CP REQUEST STACK
B1 EXTENDING FREE STORAGE
YES
NO
EXTDISP
B2 GET EXECUTION ADDRESSES FOR EXTEND
C2
CKLIST
C1 STACK EMPTY
YES
NO
A4
C2 STACK EMPTY
YES
NO
LOADIDLE
C3 SET CPUTIMER = SYSTEM
CKCPREQB
D1 UNCHAIN THE BLOCK FROM THE STACK
D2 WILL THIS BLOK EXTEND
YES
NO
D3 SET RUNUSER = SYSTEM VMBLOK, SET RUNPSW = WAIT PSW
E1 SAVE THE REGISTERS FROM THE BLOK IN TEMPSAVE
E2 POINT TO NEXT CPEXBLOK ON STACK
C2
E3 FLAG CP IN WAIT STATE
F1 DMKFRET CALL - RETURN THE CPEXBLOK TO FREE STORAGE
F3 SET INTERVAL TIMER TO AVOID UNNECESSARY INTERRUPT
G1 LOAD THE REGISTERS FROM TEMPSAVE, GET EXECUTION ADDRESS
G3 'SSM' ALLOW EXTERNAL AND I/O INTERRUPTS
H1 START CHARGING THE OWNER OF THE CPEXBLOK FOR CPU TIME
H3 DETERMINE WHY SYSTEM IS ENTERING WAIT STATE CONDITION
J1 GOTO CPEX EXECUTION ADDRESS
J3 THIS CAN BE 1 OF 3 CONDITIONS. I/O - IDLE - OR PAGING
SETWAIT
K3 LPSW - WAIT
A4
CKUSERS
A4 GET LAST RUN USER AND HIS REMAINING TIME SLICE
B4 STILL DISPATCHABLE
YES
NO
H4
C4 GET 1ST VMBLOK ON LIST OF RUNNABLE USERS
D4 LIST EMPTY
YES
NO
E4 GET MAXIMUM TIME SLICE
F4 COMPUTE BOUND
NO
YES
G4 DOUBLE TIME SLICE
SETTIME
H4 START CHARGING NEW USER FOR CPU TIME
H4
08 A1

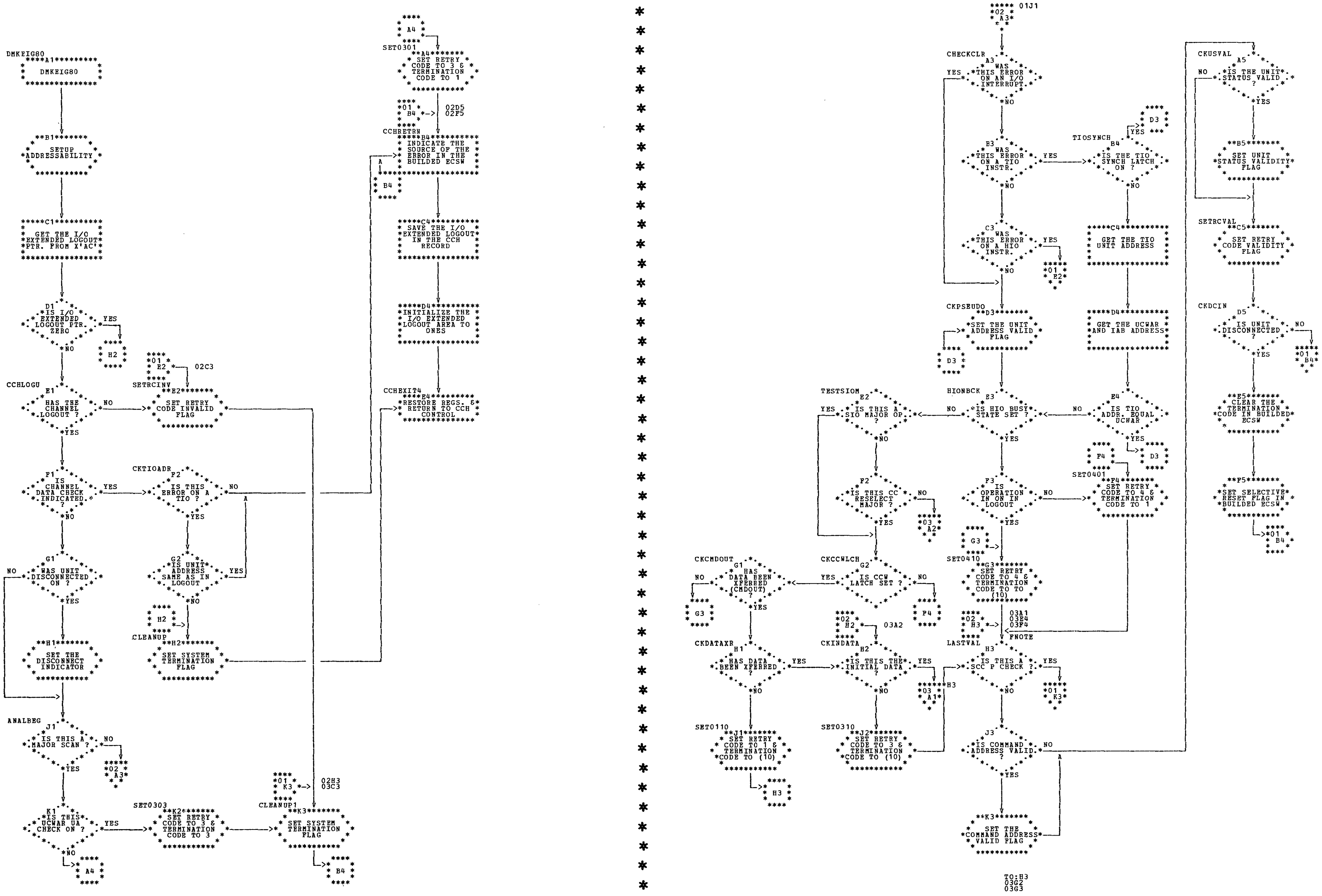

07H4
08 A1
SETQUANT
A1 SET QUANTUM = TIME SLICE
B1 SET UP VMPSW
C1 EC REGS PRESENT
NO
YES
A
D1 LOAD USERS REAL CREGS WITH USERS VALUES
E1 IN EC MODE NOW
NO
YES
F1 CREG0 OR SHADOW TABLES INVALID
NO
YES
G1 SVC 0 - ABEND CODE DSP002
CHKTRAN
F2 IN VIRTUAL TRANSLATE MODE
NO
YES
G2 USE C-REGS 0 & 1 FROM USER'S SHADOW TABLES
GETCREGS
F3 USE STANDARD C-REG 0 & C-REG 1 FROM USER'S VMBLOK
SETCREGS
G3 STORE C-REGS 0 & 1 FOR USE VERY SHORTLY
H3 SHADOW TABLES OK
YES
NO
J3 DMKVATAB CALL - INVALIDATE THE SHADOW TABLES
K3 PURGE TRANSLATE LOOK-ASIDE BUFFER
CHKRUNE
A4 SET UP FOR CP RELOCATE
B4 PTE'S INVALIDATED SINCE LAST DISPATCH
NO
YES
C4 PURGE TRANSLATE LOOK-ASIDE BUFFER
LOADFPR
D4 REDISPATCH RUNUSER
YES
NO
E4 LOAD FLOATING POINT REGS
TAKEOFF
F4 PER TRACING ACTIVE
NO
YES
H4
G4 LOAD TRACE CREGS
08 H4
09E1
RELPSW
H4 FLAG DISPATCHED RUNUSER, POINT TO HIS MEMORY SPACE
H4
J4 VIRTUAL LOCATION 80 AVAILABLE
NO
YES
K4 SET VIRTUAL LOCATION 80 TIMER = SAVED VMTIMER
A5 SET CPSTATUS = CPRUN, RELOAD USER'S GPR0-15
B5 SET CREG0,1 = RUNCR0,1; SET TIMER = QUANTUM
C5 LPSW RUNPSW

| DMKDSP -- Dispatcher (Parts 7 and 8 of 10)

| DMKDSP -- Dispatcher (Parts 9 and 10 of 10)

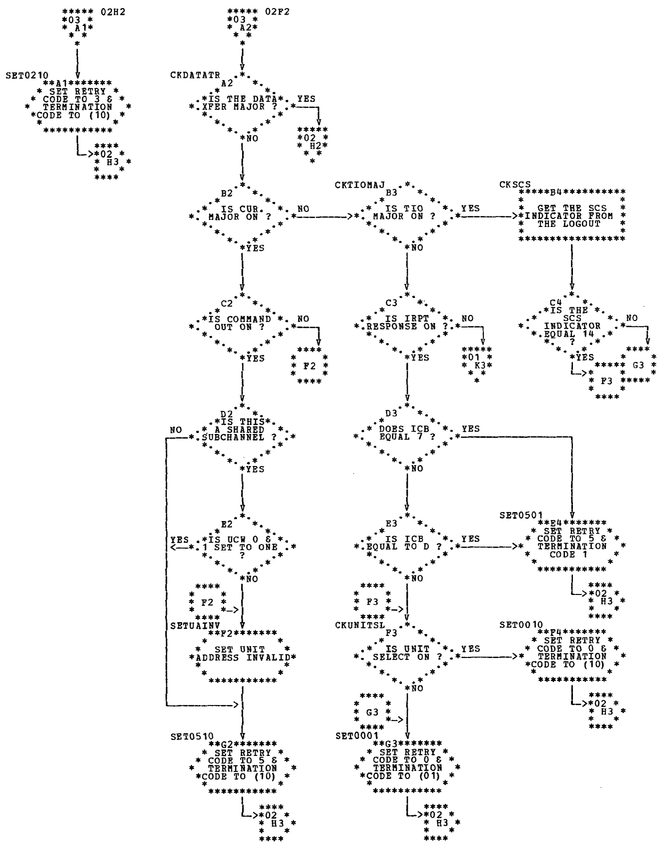

DMKDSPA
A1
DMKDSPA FAST REFLECT
B1
RUNTIME
DO RUNNING USER ACCOUNTING
C1
USER STILL DISPATCHABLE
NO
01 D2
YES
D1
SET RUNPSW = PROGRAM OLD PSW + VIRTUAL CC AND PROGRAM MASK
E1
SET QUANTUM = QUANTUMR
08 H4
DMKDSPB
A2
DMKDSPB
B2
TEST CPSTATUS SWITCH
CPEX
01 K2
RUN
WAIT
B3
RUNTIME
DO RUNNING UWER ACCOUNTING
01 K2
C2
WAITIME
DO WAIT TIME ACCOUNTING
01 K2
RUNTIME
A4
RUNTIME
B4
GPR11 = RUNUSER
YES
NO
C4
STOP CHARGING OLD USER, RE-ESTABLISH RUNUSER
UNRUN
D4
SET CPSTATUS SWITCH = CPEX
E4
IF USER'S PAGE 0 IS AVAILABLE, SAVE LOCATION 80 TIMER IN VMBLOK
F4
TIMER CHANGE FROM + TO -
NO
YES
G4
FLAG INTERVAL TIMER INTERRUPT PENDING
H4
FLAG USER NOT ELIGIBLE FOR FAST REFLECT
NOINTIMR
J4
R9 RETURN
WAITIME
A5
WAITIME
B5
SET CPSTATUS = CPEX
C5
GET SYSTEM'S VMBLOK FOR WAIT TIME PROCESSING
D5
WAS ANY WAIT FLAG SET
NO
01 D2
YES
E5
DETERMINE WAIT CONDITION. I.E. I/O, PAGING OR IDLE
F5
UPDATE CORRESPONDING WAIT FIELD
G5
RESET SYSTEM WAIT FLAG BIT
01 D2

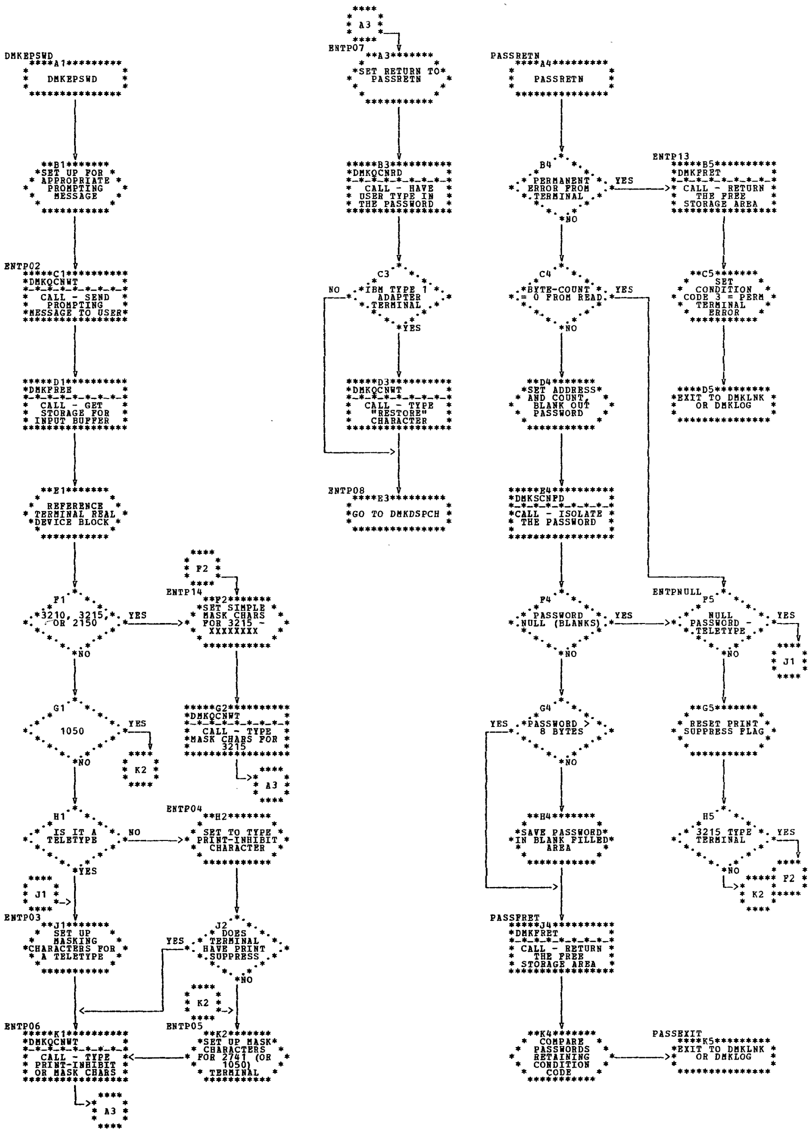



DMKEIG80
A1
DMKEIG80
B1
SETUP ADDRESSABILITY
C1
GET THE I/O EXTENDED LOGOUT PTR. FROM X'AC'
D1
IS I/O EXTENDED LOGOUT PTR. ZERO
YES
NO
H2
01 E2
02C3
CCHLOGU
E1
HAS THE CHANNEL LOGOUT ?
NO
YES
SETRCINV
E2
SET RETRY CODE INVALID FLAG
F1
IS CHANNEL DATA CHECK INDICATED ?
YES
NO
CKTIOADR
F2
IS THIS ERROR ON A TIO ?
NO
YES
A
G1
WAS UNIT DISCONNECTED ON ?
NO
YES
G2
IS UNIT ADDRESS SAME AS IN LOGOUT
YES
NO
H2
CLEANUP
H2
SET SYSTEM TERMINATION FLAG
H1
SET THE DISCONNECT INDICATOR
ANALBEG
J1
IS THIS A MAJOR SCAN ?
NO
YES
02 A3
K1
IS THIS UCWAR UA CHECK ON ?
YES
NO
A4
SET0303
K2
SET RETRY CODE TO 3 & TERMINATION CODE TO 3
01 K3
02H3
03C3
CLEANUP1
K3
SET SYSTEM TERMINATION FLAG
B4
A4
SET0301
A4
SET RETRY CODE TO 3 & TERMINATION CODE TO 1
01 B4
02D5
02F5
CCHRETRN
B4
INDICATE THE SOURCE OF THE ERROR IN THE BUILDED ECSW
A
B4
C4
SAVE THE I/O EXTENDED LOGOUT IN THE CCH RECORD
D4
INITIALIZE THE I/O EXTENDED LOGOUT AREA TO ONES
CCHEXIT4
E4
RESTORE REGS. & RETURN TO CCH CONTROL
02 A3
01J1
CHECKCLR
A3
WAS THIS ERROR ON AN I/O INTERRUPT
YES
NO
B3
WAS THIS ERROR ON A TIO INSTR.
YES
NO
TIOSYNCH
B4
IS THE TIO SYNCH LATCH ON ?
YES
D3
NO
C3
WAS THIS ERROR ON A HIO INSTR.
YES
01 E2
NO
C4
GET THE TIO UNIT ADDRESS
CKPSEUDO
D3
SET THE UNIT ADDRESS VALID FLAG
D3
D4
GET THE UCWAR AND IAB ADDRESS
TESTSIOM
E2
IS THIS A SIO MAJOR OP. ?
YES
NO
HIONBCK
E3
IS HIO BUSY STATE SET ?
NO
YES
E4
IS TIO ADDR. EQUAL UCWAR
NO
YES
D3
F4
SET0401
F4
SET RETRY CODE TO 4 & TERMINATION CODE TO 1
F2
IS THIS CC RESELECT MAJOR ?
NO
03 A2
YES
F3
IS OPERATION IN ON IN LOGOUT
NO
YES
G3
SET0410
G3
SET RETRY CODE TO 4 & TERMINATION CODE TO TO (10)
02 H3
03A1
03E4
03F4
FNOTE
CKCMDOUT
G1
HAS DATA BEEN XFERRED (CMDOUT) ?
NO
YES
G3
CKCCWLCH
G2
IS CCW LATCH SET ?
YES
NO
F4
02 H2
03A2
CKDATAXR
H1
HAS DATA BEEN XFERRED ?
YES
NO
CKINDATA
H2
IS THIS THE INITIAL DATA ?
YES
NO
03 A1
H3
LASTVAL
H3
IS THIS A SCC P CHECK ?
YES
NO
01 K3
SET0110
J1
SET RETRY CODE TO 1 & TERMINATION CODE TO (10)
H3
SET0310
J2
SET RETRY CODE TO 3 & TERMINATION CODE TO (10)
J3
IS COMMAND ADDRESS VALID ?
NO
A
YES
K3
SET THE COMMAND ADDRESS VALID FLAG
TO:H3
03G2
03G3
CKUSVAL
A5
IS THE UNIT STATUS VALID ?
NO
YES
B5
SET UNIT STATUS VALIDITY FLAG
SETRCVAL
C5
SET RETRY CODE VALIDITY FLAG
CKDCIN
D5
IS UNIT DISCONNECTED ?
NO
01 B4
YES
E5
CLEAR THE TERMINATION CODE IN BUILDED ECSW
F5
SET SELECTIVE RESET FLAG IN BUILDED ECSW
01 B4

| DMKEIG -- 2880 Channel Module (Parts 1 and 2 of 3)

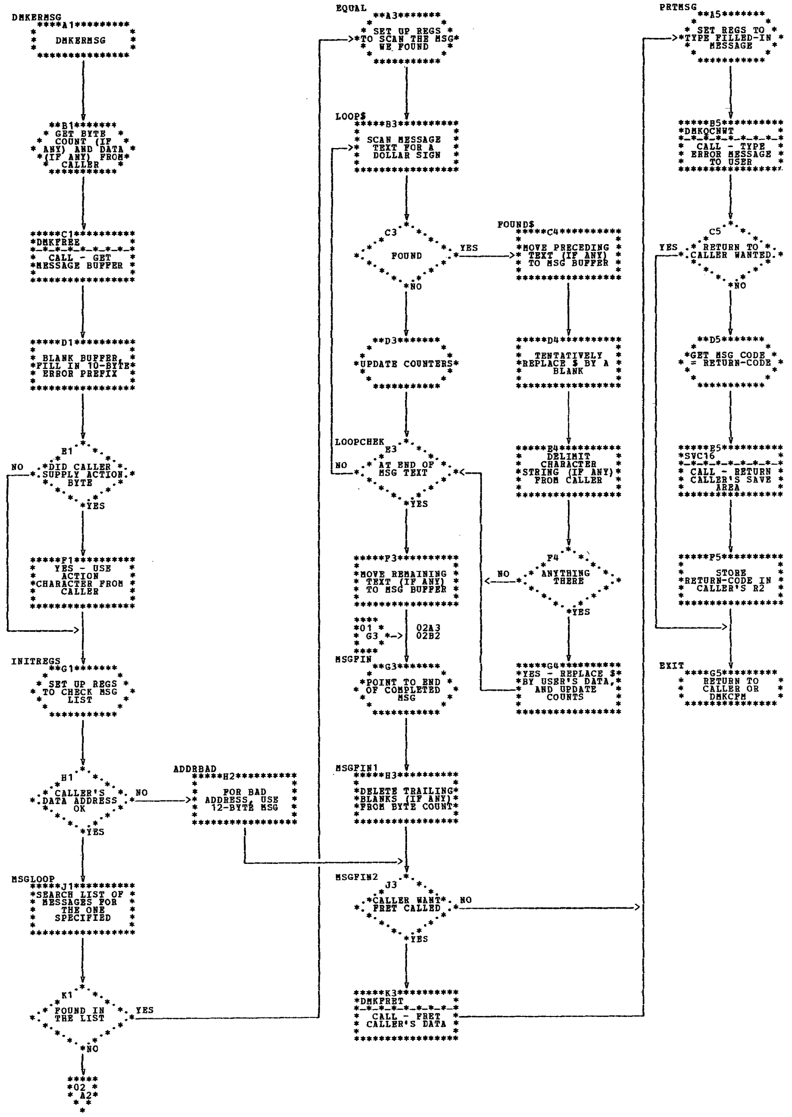
| DMKEIG -- 2880 Channel Module (Part 3 of 3)

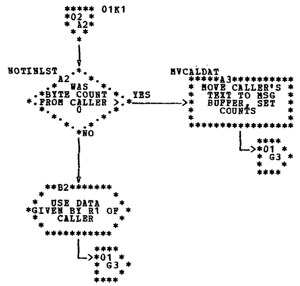


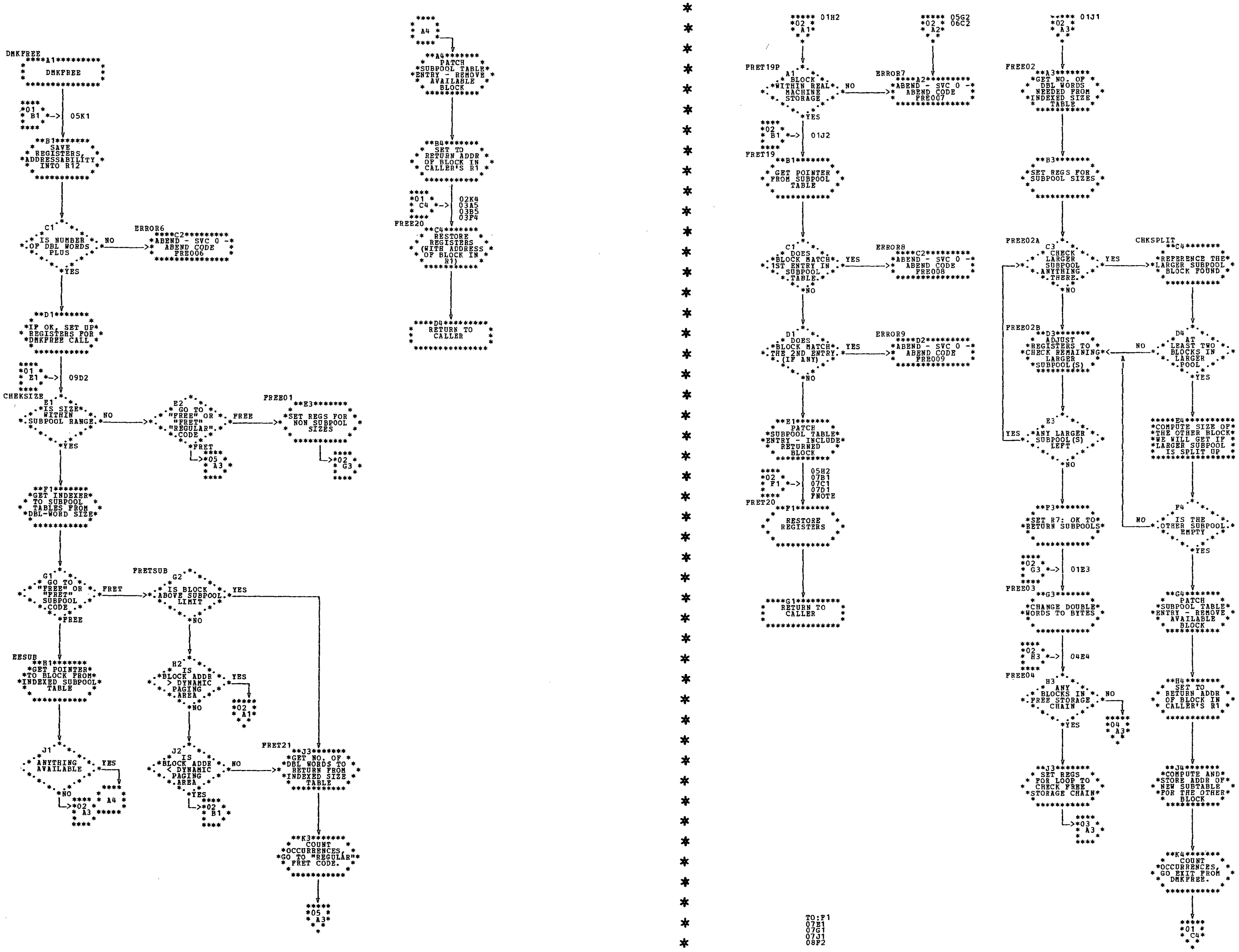

DMKEPSWD
A1
DMKEPSWD
B1
SET UP FOR APPROPRIATE PROMPTING MESSAGE
ENTP02
C1
DMKQCNWT
CALL - SEND PROMPTING MESSAGE TO USER
D1
DMKFREE
CALL - GET STORAGE FOR INPUT BUFFER
E1
REFERENCE TERMINAL REAL DEVICE BLOCK
F1
3210, 3215, OR 2150
YES
NO
G1
1050
YES
K2
NO
H1
IS IT A TELETYPE
NO
YES
J1
ENTP03
J1
SET UP MASKING CHARACTERS FOR A TELETYPE
ENTP06
K1
DMKQCNWT
CALL - TYPE PRINT-INHIBIT OR MASK CHARS
A3
F2
ENTP14
F2
SET SIMPLE MASK CHARS FOR 3215 - XXXXXXXX
G2
DMKQCNWT
CALL - TYPE MASK CHARS FOR 3215
A3
ENTP04
H2
SET TO TYPE PRINT-INHIBIT CHARACTER
J2
DOES TERMINAL HAVE PRINT SUPPRESS
YES
NO
K2
ENTP05
K2
SET UP MASK CHARACTERS FOR 2741 (OR 1050) TERMINAL
A3
ENTP07
A3
SET RETURN TO PASSRETN
B3
DMKQCNRD
CALL - HAVE USER TYPE IN THE PASSWORD
C3
IBM TYPE 1 ADAPTER TERMINAL
NO
YES
D3
DMKQCNWT
CALL - TYPE "RESTORE" CHARACTER
ENTP08
E3
GO TO DMKDSPCH
PASSRETN
A4
PASSRETN
B4
PERMANENT ERROR FROM TERMINAL
YES
NO
C4
BYTE-COUNT = 0 FROM READ
YES
NO
D4
SET ADDRESS AND COUNT, BLANK OUT PASSWORD
E4
DMKSCNFD
CALL - ISOLATE THE PASSWORD
F4
PASSWORD NULL (BLANKS)
YES
NO
G4
PASSWORD > 8 BYTES
YES
NO
H4
SAVE PASSWORD IN BLANK FILLED AREA
PASSFRET
J4
DMKFRET
CALL - RETURN THE FREE STORAGE AREA
K4
COMPARE PASSWORDS RETAINING CONDITION CODE
ENTP13
B5
DMKFRET
CALL - RETURN THE FREE STORAGE AREA
C5
SET CONDITION CODE 3 = PERM TERMINAL ERROR
D5
EXIT TO DMKLNK OR DMKLOG
ENTPNULL
F5
NULL PASSWORD - TELETYPE
YES
J1
NO
G5
RESET PRINT SUPPRESS FLAG
H5
3215 TYPE TERMINAL
YES
F2
NO
K2
PASSEXIT
K5
EXIT TO DMKLNK OR DMKLOG

DMKEPS -- Process User Password (Part 1 of 1)

DMKERM -- Message Writer (Parts 1 and 2 of 2)


DMKERMSG
A1
DMKERMSG
B1
GET BYTE COUNT (IF ANY) AND DATA (IF ANY) FROM CALLER
C1
DMKFREE
CALL - GET MESSAGE BUFFER
D1
BLANK BUFFER, FILL IN 10-BYTE ERROR PREFIX
E1
DID CALLER SUPPLY ACTION BYTE
NO
YES
F1
YES - USE ACTION CHARACTER FROM CALLER
INITREGS
G1
SET UP REGS TO CHECK MSG LIST
H1
CALLER'S DATA ADDRESS OK
NO
YES
ADDRBAD
H2
FOR BAD ADDRESS, USE 12-BYTE MSG
MSGLOOP
J1
SEARCH LIST OF MESSAGES FOR THE ONE SPECIFIED
K1
FOUND IN THE LIST
YES
NO
02 A2
EQUAL
A3
SET UP REGS TO SCAN THE MSG WE FOUND
LOOP$
B3
SCAN MESSAGE TEXT FOR A DOLLAR SIGN
C3
FOUND
YES
NO
D3
UPDATE COUNTERS
LOOPCHEK
E3
AT END OF MSG TEXT
NO
YES
F3
MOVE REMAINING TEXT (IF ANY) TO MSG BUFFER
01 G3
02A3
02B2
MSGFIN
G3
POINT TO END OF COMPLETED MSG
MSGFIN1
H3
DELETE TRAILING BLANKS (IF ANY) FROM BYTE COUNT
MSGFIN2
J3
CALLER WANT FRET CALLED
NO
YES
K3
DMKFRET
CALL - FRET CALLER'S DATA
FOUND$
C4
MOVE PRECEDING TEXT (IF ANY) TO MSG BUFFER
D4
TENTATIVELY REPLACE $ BY A BLANK
E4
DELIMIT CHARACTER STRING (IF ANY) FROM CALLER
F4
ANYTHING THERE
NO
YES
G4
YES - REPLACE $ BY USER'S DATA, AND UPDATE COUNTS
PRTMSG
A5
SET REGS TO TYPE FILLED-IN MESSAGE
B5
DMKQCNWT
CALL - TYPE ERROR MESSAGE TO USER
C5
RETURN TO CALLER WANTED
YES
NO
D5
GET MSG CODE = RETURN-CODE
E5
SVC16
CALL - RETURN CALLER'S SAVE AREA
F5
STORE RETURN-CODE IN CALLER'S R2
EXIT
G5
RETURN TO CALLER OR DMKCFM

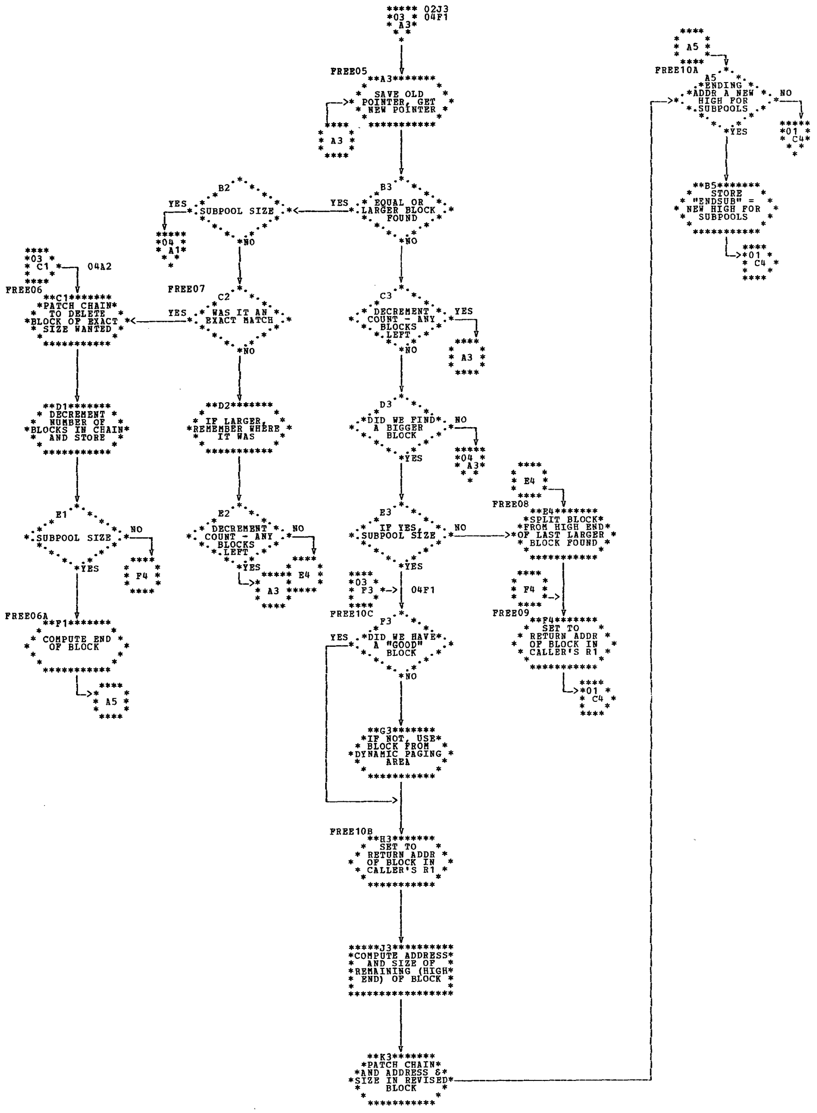

02 A2
01K1
NOTINLST
A2
WAS BYTE COUNT FROM CALLER > 0
YES
NO
MVCALDAT
A3
MOVE CALLER'S TEXT TO MSG BUFFER, SET COUNTS
01 G3
B2
USE DATA GIVEN BY R1 OF CALLER
01 G3

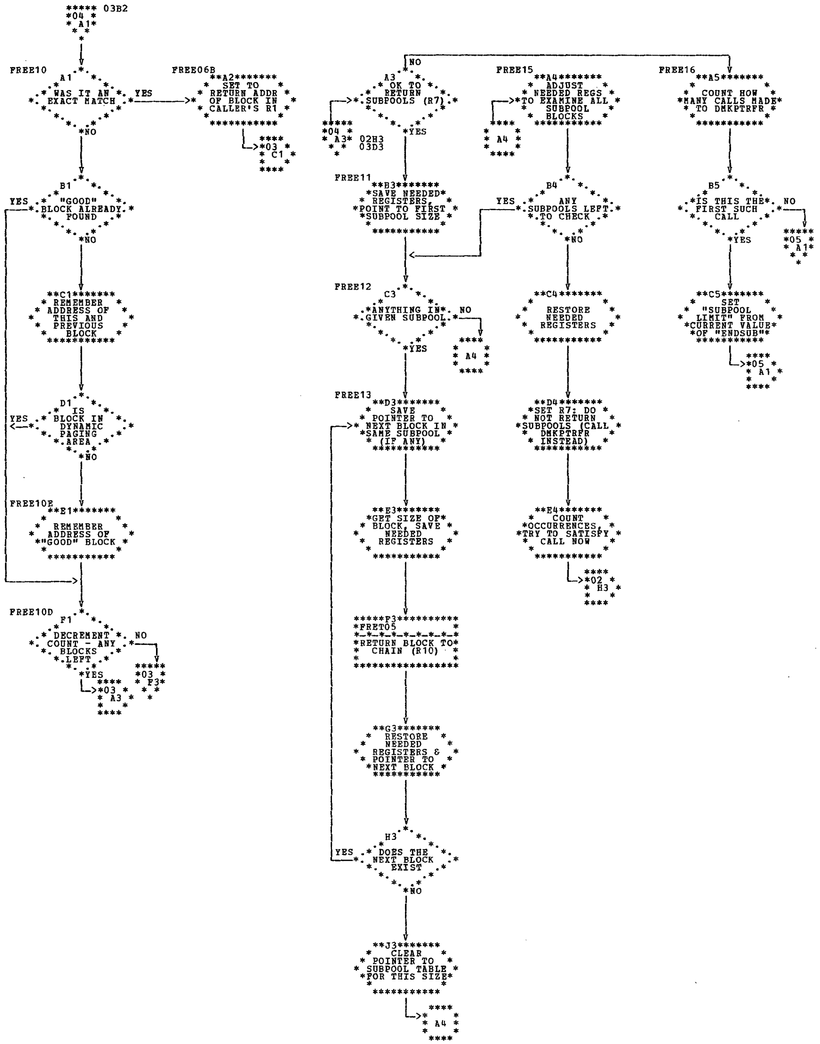


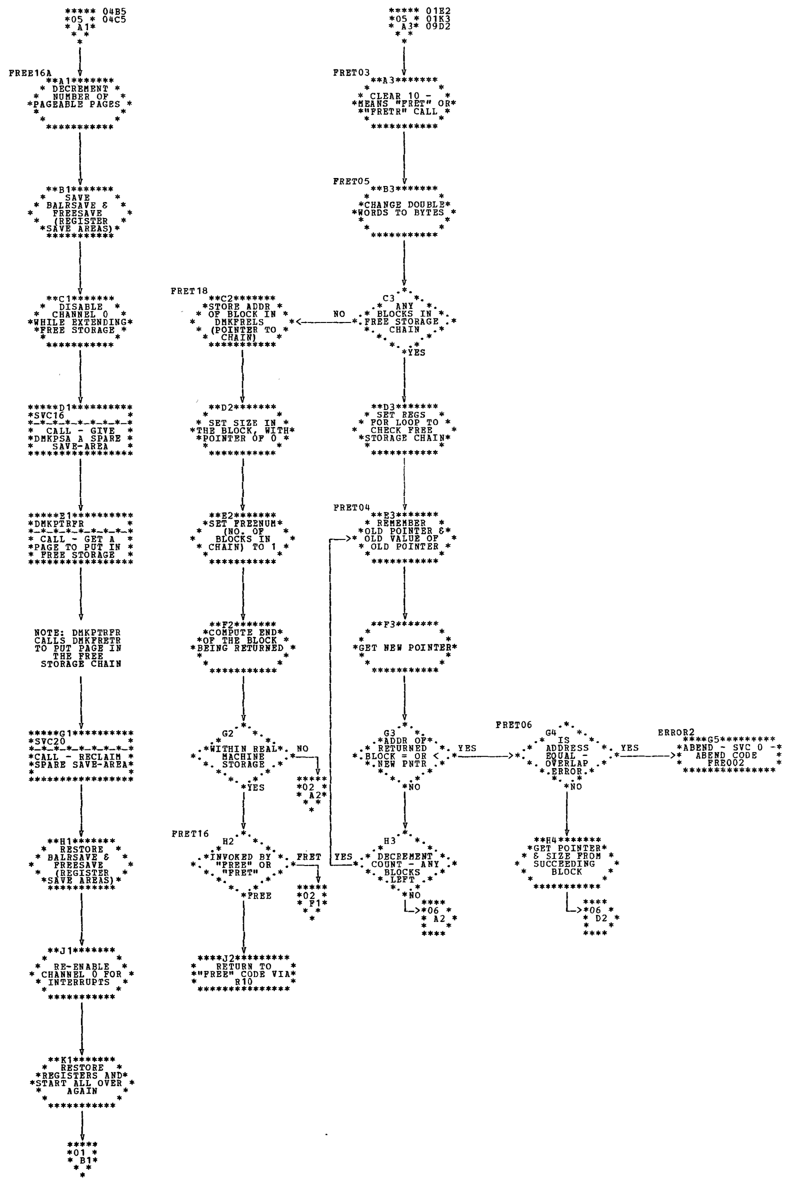
DMKFREE
A1 DMKFREE
01 B1 05K1
B1 SAVE REGISTERS, ADDRESSABILITY INTO R12
C1 IS NUMBER OF DBL WORDS PLUS
NO
YES
ERROR6
C2 ABEND - SVC 0 - ABEND CODE FRE006
D1 IF OK, SET UP REGISTERS FOR DMKFREE CALL
01 E1 09D2
CHEKSIZE
E1 IS SIZE WITHIN SUBPOOL RANGE
E2 GO TO "FREE" OR "FRET" "REGULAR" CODE
FREE
FRET
FREE01
E3 SET REGS FOR NON SUBPOOL SIZES
05 A3
02 G3
F1 GET INDEXER TO SUBPOOL TABLES FROM DBL-WORD SIZE
G1 GO TO "FREE" OR "FRET" SUBPOOL CODE
FRETSUB
G2 IS BLOCK ABOVE SUBPOOL LIMIT
EESUB
H1 GET POINTER TO BLOCK FROM INDEXED SUBPOOL TABLE
H2 IS BLOCK ADDR > DYNAMIC PAGING AREA
02 A1
J1 ANYTHING AVAILABLE
A4
02 A3
J2 IS BLOCK ADDR < DYNAMIC PAGING AREA
02 B1
FRET21
J3 GET NO. OF DBL WORDS TO RETURN FROM INDEXED SIZE TABLE
K3 COUNT OCCURRENCES, GO TO "REGULAR" FRET CODE.
05 A3
A4
A4 PATCH SUBPOOL TABLE ENTRY - REMOVE AVAILABLE BLOCK
B4 SET TO RETURN ADDR OF BLOCK IN CALLER'S R1
01 C4 02K4 03A5 03B5 03F4
FREE20
C4 RESTORE REGISTERS (WITH ADDRESS OF BLOCK IN R1)
D4 RETURN TO CALLER
01H2
02 A1
FRET19P
A1 BLOCK WITHIN REAL MACHINE STORAGE
ERROR7
A2 ABEND - SVC 0 - ABEND CODE FRE007
05G2 06C2
02 A2
02 B1 01J2
FRET19
B1 GET POINTER FROM SUBPOOL TABLE
C1 DOES BLOCK MATCH 1ST ENTRY IN SUBPOOL TABLE
ERROR8
C2 ABEND - SVC 0 - ABEND CODE FRE008
D1 DOES BLOCK MATCH THE 2ND ENTRY (IF ANY)
ERROR9
D2 ABEND - SVC 0 - ABEND CODE FRE009
E1 PATCH SUBPOOL TABLE ENTRY - INCLUDE RETURNED BLOCK
02 F1 05H2 07B1 07C1 07D1 FNOTE
FRET20
F1 RESTORE REGISTERS
G1 RETURN TO CALLER
TO:F1 07E1 07G1 07J1 08F2
01J1
02 A3
FREE02
A3 GET NO. OF DBL WORDS NEEDED FROM INDEXED SIZE TABLE
B3 SET REGS FOR SUBPOOL SIZES
FREE02A
C3 CHECK LARGER SUBPOOL ANYTHING THERE
CHKSPLIT
C4 REFERENCE THE LARGER SUBPOOL BLOCK FOUND
FREE02B
D3 ADJUST REGISTERS TO CHECK REMAINING LARGER SUBPOOL(S)
D4 AT LEAST TWO BLOCKS IN LARGER POOL
E3 ANY LARGER SUBPOOL(S) LEFT
E4 COMPUTE SIZE OF THE OTHER BLOCK WE WILL GET IF LARGER SUBPOOL IS SPLIT UP
F3 SET R7: OK TO RETURN SUBPOOLS
F4 IS THE OTHER SUBPOOL EMPTY
02 G3 01E3
FREE03
G3 CHANGE DOUBLE WORDS TO BYTES
G4 PATCH SUBPOOL TABLE ENTRY - REMOVE AVAILABLE BLOCK
02 H3 04E4
FREE04
H3 ANY BLOCKS IN FREE STORAGE CHAIN
04 A3
H4 SET TO RETURN ADDR OF BLOCK IN CALLER'S R1
J3 SET REGS FOR LOOP TO CHECK FREE STORAGE CHAIN
03 A3
J4 COMPUTE AND STORE ADDR OF NEW SUBTABLE FOR THE OTHER BLOCK
K4 COUNT OCCURRENCES, GO EXIT FROM DMKFREE.
01 C4

| DMKFRE -- Free Storage Manager (Parts 1 and 2 of 9)

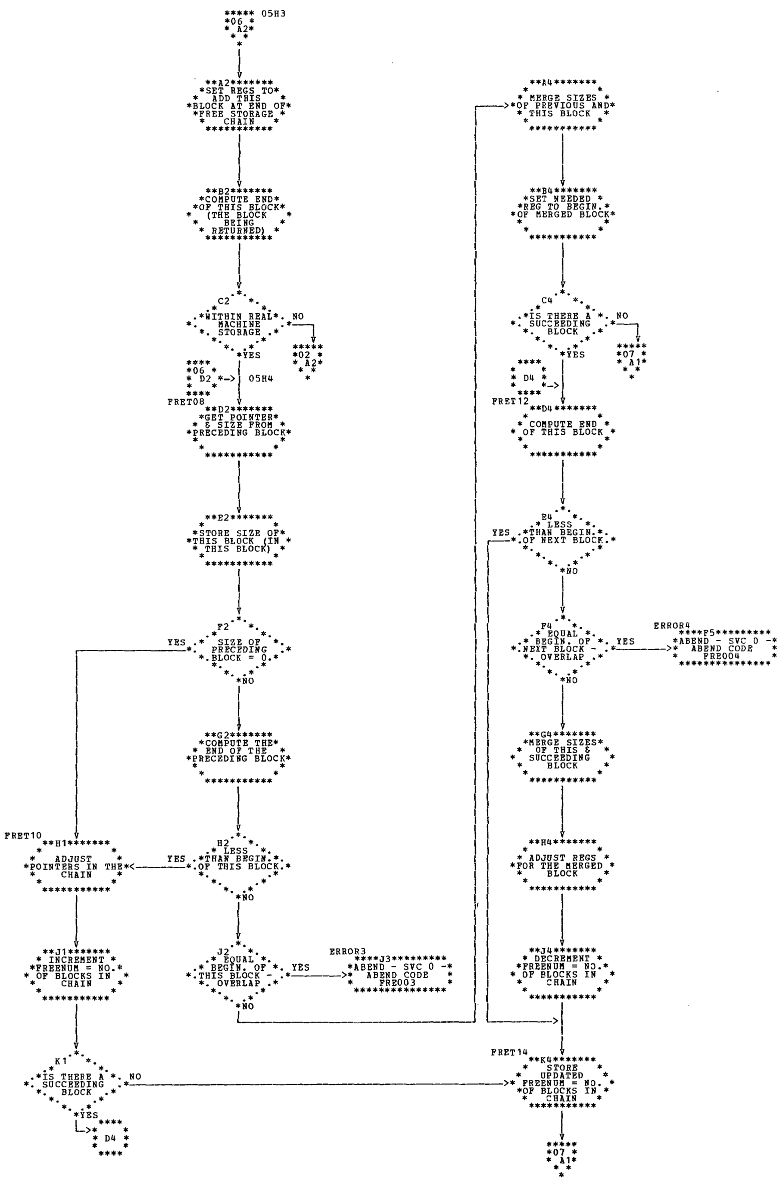
| DMKFRE -- Free Storage Manager (Parts 3 and 4 of 9)


03 A3 02J3 04F1
FREE05 A3 SAVE OLD POINTER, GET NEW POINTER
A3
B3 EQUAL OR LARGER BLOCK FOUND
YES
NO
B2 SUBPOOL SIZE
YES
04 A1
NO
03 C1 04A2
FREE06 C1 PATCH CHAIN TO DELETE BLOCK OF EXACT SIZE WANTED
FREE07 C2 WAS IT AN EXACT MATCH
C3 DECREMENT COUNT - ANY BLOCKS LEFT
A3
D1 DECREMENT NUMBER OF BLOCKS IN CHAIN AND STORE
D2 IF LARGER, REMEMBER WHERE IT WAS
D3 DID WE FIND A BIGGER BLOCK
04 A3
E1 SUBPOOL SIZE
F4
E2 DECREMENT COUNT - ANY BLOCKS LEFT
E4
A3
E3 IF YES, SUBPOOL SIZE
E4
FREE08 E4 SPLIT BLOCK FROM HIGH END OF LAST LARGER BLOCK FOUND
03 F3 04F1
FREE06A F1 COMPUTE END OF BLOCK
A5
FREE10C F3 DID WE HAVE A "GOOD" BLOCK
F4
FREE09 F4 SET TO RETURN ADDR OF BLOCK IN CALLER'S R1
01 C4
G3 IF NOT, USE BLOCK FROM DYNAMIC PAGING AREA
FREE10B H3 SET TO RETURN ADDR OF BLOCK IN CALLER'S R1
J3 COMPUTE ADDRESS AND SIZE OF REMAINING (HIGH END) OF BLOCK
K3 PATCH CHAIN AND ADDRESS & SIZE IN REVISED BLOCK
A5
FREE10A A5 ENDING ADDR A NEW HIGH FOR SUBPOOLS
01 C4
B5 STORE "ENDSUB" = NEW HIGH FOR SUBPOOLS
01 C4

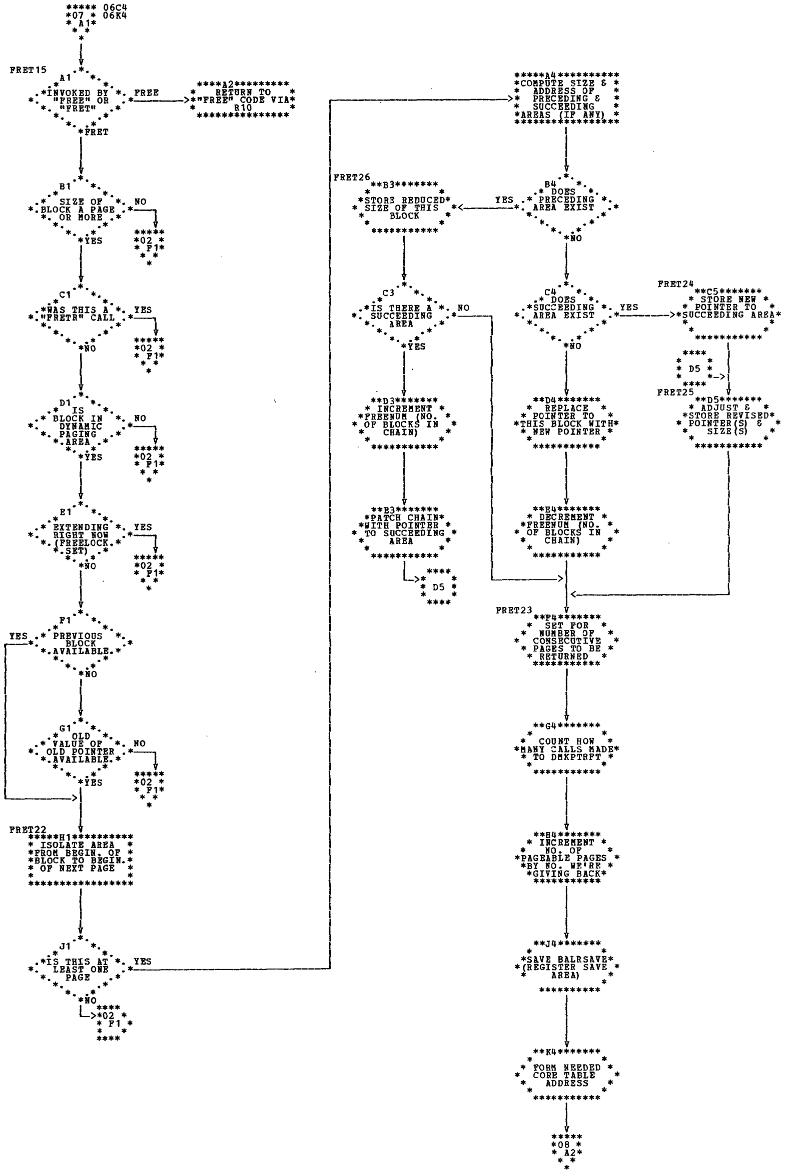

04 A1 03B2
FREE10 A1 WAS IT AN EXACT MATCH
YES
NO
FREE06B A2 SET TO RETURN ADDR OF BLOCK IN CALLER'S R1
03 C1
B1 "GOOD" BLOCK ALREADY FOUND
C1 REMEMBER ADDRESS OF THIS AND PREVIOUS BLOCK
D1 IS BLOCK IN DYNAMIC PAGING AREA
FREE10E E1 REMEMBER ADDRESS OF "GOOD" BLOCK
FREE10D F1 DECREMENT COUNT - ANY BLOCKS LEFT
03 A3
03 F3
A3 OK TO RETURN SUBPOOLS (R7)
04 A3 02H3 03D3
FREE11 B3 SAVE NEEDED REGISTERS, POINT TO FIRST SUBPOOL SIZE
FREE12 C3 ANYTHING IN GIVEN SUBPOOL
A4
FREE13 D3 SAVE POINTER TO NEXT BLOCK IN SAME SUBPOOL (IF ANY)
E3 GET SIZE OF BLOCK, SAVE NEEDED REGISTERS
F3 FRET05 RETURN BLOCK TO CHAIN (R10)
G3 RESTORE NEEDED REGISTERS & POINTER TO NEXT BLOCK
H3 DOES THE NEXT BLOCK EXIST
J3 CLEAR POINTER TO SUBPOOL TABLE FOR THIS SIZE
A4
FREE15 A4 ADJUST NEEDED REGS TO EXAMINE ALL SUBPOOL BLOCKS
A4
B4 ANY SUBPOOLS LEFT TO CHECK
C4 RESTORE NEEDED REGISTERS
D4 SET R7: DO NOT RETURN SUBPOOLS (CALL DMKPTRFR INSTEAD)
E4 COUNT OCCURRENCES, TRY TO SATISFY CALL NOW
02 H3
FREE16 A5 COUNT HOW MANY CALLS MADE TO DMKPTRFR
B5 IS THIS THE FIRST SUCH CALL
05 A1
C5 SET "SUBPOOL LIMIT" FROM CURRENT VALUE OF "ENDSUB"
05 A1

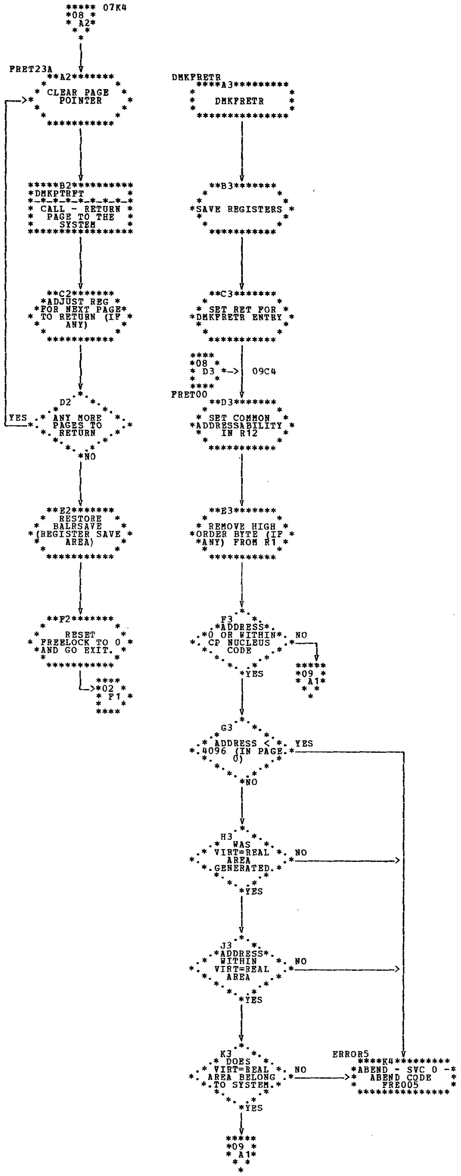

05 A1 04B5 04C5
FREE16A
A1 DECREMENT NUMBER OF PAGEABLE PAGES
B1 SAVE BALRSAVE & FREESAVE (REGISTER SAVE AREAS)
C1 DISABLE CHANNEL 0 WHILE EXTENDING FREE STORAGE
D1 SVC16 CALL - GIVE DMKPSA A SPARE SAVE-AREA
E1 DMKPTRFR CALL - GET A PAGE TO PUT IN FREE STORAGE
NOTE: DMKPTRFR CALLS DMKFRETR TO PUT PAGE IN THE FREE STORAGE CHAIN
G1 SVC20 CALL - RECLAIM SPARE SAVE-AREA
H1 RESTORE BALRSAVE & FREESAVE (REGISTER SAVE AREAS)
J1 RE-ENABLE CHANNEL 0 FOR INTERRUPTS
K1 RESTORE REGISTERS AND START ALL OVER AGAIN
01 B1
FRET18
C2 STORE ADDR OF BLOCK IN DMKFRELS (POINTER TO CHAIN)
D2 SET SIZE IN THE BLOCK, WITH POINTER OF 0
E2 SET FREENUM (NO. OF BLOCKS IN CHAIN) TO 1
F2 COMPUTE END OF THE BLOCK BEING RETURNED
G2 WITHIN REAL MACHINE STORAGE
NO
02 A2
YES
FRET16
H2 INVOKED BY "FREE" OR "FRET"
FRET
02 F1
FREE
J2 RETURN TO "FREE" CODE VIA R10
05 A3 01E2 01K3 09D2
FRET03
A3 CLEAR 10 - MEANS "FRET" OR "FRETR" CALL
FRET05
B3 CHANGE DOUBLE WORDS TO BYTES
C3 ANY BLOCKS IN FREE STORAGE CHAIN
NO
YES
D3 SET REGS FOR LOOP TO CHECK FREE STORAGE CHAIN
FRET04
E3 REMEMBER OLD POINTER & OLD VALUE OF OLD POINTER
F3 GET NEW POINTER
G3 ADDR OF RETURNED BLOCK = OR < NEW PNTR
YES
NO
H3 DECREMENT COUNT - ANY BLOCKS LEFT
YES
NO
06 A2
FRET06
G4 IS ADDRESS EQUAL - OVERLAP ERROR
YES
NO
H4 GET POINTER & SIZE FROM SUCCEEDING BLOCK
06 D2
ERROR2
G5 ABEND - SVC 0 - ABEND CODE FRE002

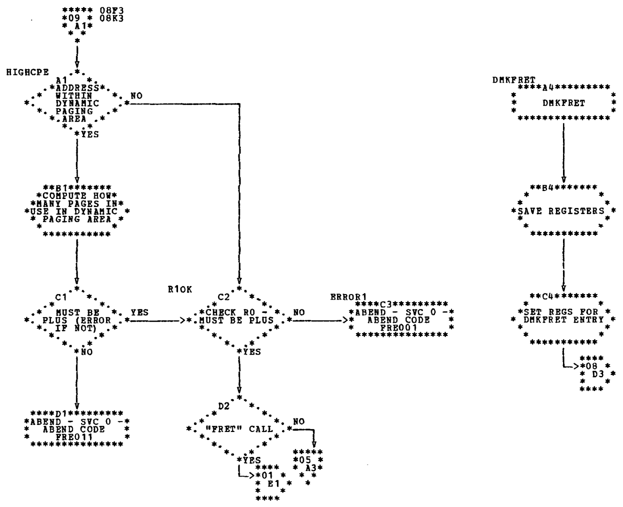
| DMKFRE -- Free Storage Manager (Parts 5 and 6 of 9)


06 A2 05H3
A2 SET REGS TO ADD THIS BLOCK AT END OF FREE STORAGE CHAIN
B2 COMPUTE END OF THIS BLOCK (THE BLOCK BEING RETURNED)
C2 WITHIN REAL MACHINE STORAGE
NO
02 A2
YES
06 D2
05H4
FRET08
D2 GET POINTER & SIZE FROM PRECEDING BLOCK
E2 STORE SIZE OF THIS BLOCK (IN THIS BLOCK)
F2 SIZE OF PRECEDING BLOCK = 0
YES
NO
G2 COMPUTE THE END OF THE PRECEDING BLOCK
H2 LESS THAN BEGIN. OF THIS BLOCK
YES
NO
J2 EQUAL BEGIN. OF THIS BLOCK - OVERLAP
YES
NO
ERROR3
J3 ABEND - SVC 0 - ABEND CODE FRE003
FRET10
H1 ADJUST POINTERS IN THE CHAIN
J1 INCREMENT FREENUM = NO. OF BLOCKS IN CHAIN
K1 IS THERE A SUCCEEDING BLOCK
NO
YES
D4
A4 MERGE SIZES OF PREVIOUS AND THIS BLOCK
B4 SET NEEDED REG TO BEGIN. OF MERGED BLOCK
C4 IS THERE A SUCCEEDING BLOCK
NO
07 A1
YES
D4
FRET12
D4 COMPUTE END OF THIS BLOCK
E4 LESS THAN BEGIN. OF NEXT BLOCK
YES
NO
F4 EQUAL BEGIN. OF NEXT BLOCK - OVERLAP
YES
NO
ERROR4
F5 ABEND - SVC 0 - ABEND CODE FRE004
G4 MERGE SIZES OF THIS & SUCCEEDING BLOCK
H4 ADJUST REGS FOR THE MERGED BLOCK
J4 DECREMENT FREENUM = NO. OF BLOCKS IN CHAIN
FRET14
K4 STORE UPDATED FREENUM = NO. OF BLOCKS IN CHAIN
07 A1

| DMKFRE -- Free Storage Manager (Parts 7 and 8 of 9)

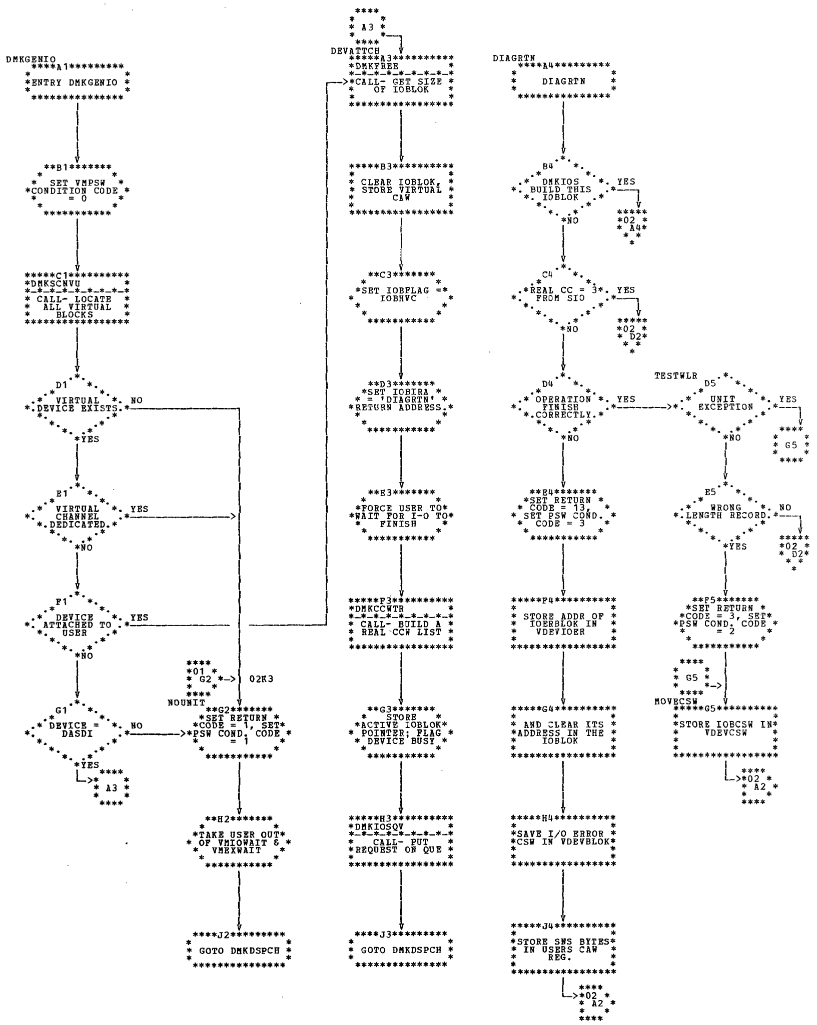

07 A1 06C4 06K4
FRET15
A1 INVOKED BY "FREE" OR "FRET"
FREE
FRET
A2 RETURN TO "FREE" CODE VIA R10
B1 SIZE OF BLOCK A PAGE OR MORE
NO
YES
02 F1
C1 WAS THIS A "FRETR" CALL
YES
NO
02 F1
D1 IS BLOCK IN DYNAMIC PAGING AREA
NO
YES
02 F1
E1 EXTENDING RIGHT NOW (FREELOCK SET)
YES
NO
02 F1
F1 PREVIOUS BLOCK AVAILABLE
YES
NO
G1 OLD VALUE OF OLD POINTER AVAILABLE
NO
YES
02 F1
FRET22
H1 ISOLATE AREA FROM BEGIN. OF BLOCK TO BEGIN. OF NEXT PAGE
J1 IS THIS AT LEAST ONE PAGE
YES
NO
02 F1
A4 COMPUTE SIZE & ADDRESS OF PRECEDING & SUCCEEDING AREAS (IF ANY)
FRET26
B3 STORE REDUCED SIZE OF THIS BLOCK
B4 DOES PRECEDING AREA EXIST
YES
NO
C3 IS THERE A SUCCEEDING AREA
NO
YES
C4 DOES SUCCEEDING AREA EXIST
YES
NO
FRET24
C5 STORE NEW POINTER TO SUCCEEDING AREA
D5
FRET25
D3 INCREMENT FREENUM (NO. OF BLOCKS IN CHAIN)
D4 REPLACE POINTER TO THIS BLOCK WITH NEW POINTER
D5 ADJUST & STORE REVISED POINTER(S) & SIZE(S)
E3 PATCH CHAIN WITH POINTER TO SUCCEEDING AREA
D5
E4 DECREMENT FREENUM (NO. OF BLOCKS IN CHAIN)
FRET23
F4 SET FOR NUMBER OF CONSECUTIVE PAGES TO BE RETURNED
G4 COUNT HOW MANY CALLS MADE TO DMKPTRFT
H4 INCREMENT NO. OF PAGEABLE PAGES BY NO. WE'RE GIVING BACK
J4 SAVE BALRSAVE (REGISTER SAVE AREA)
K4 FORM NEEDED CORE TABLE ADDRESS
08 A2

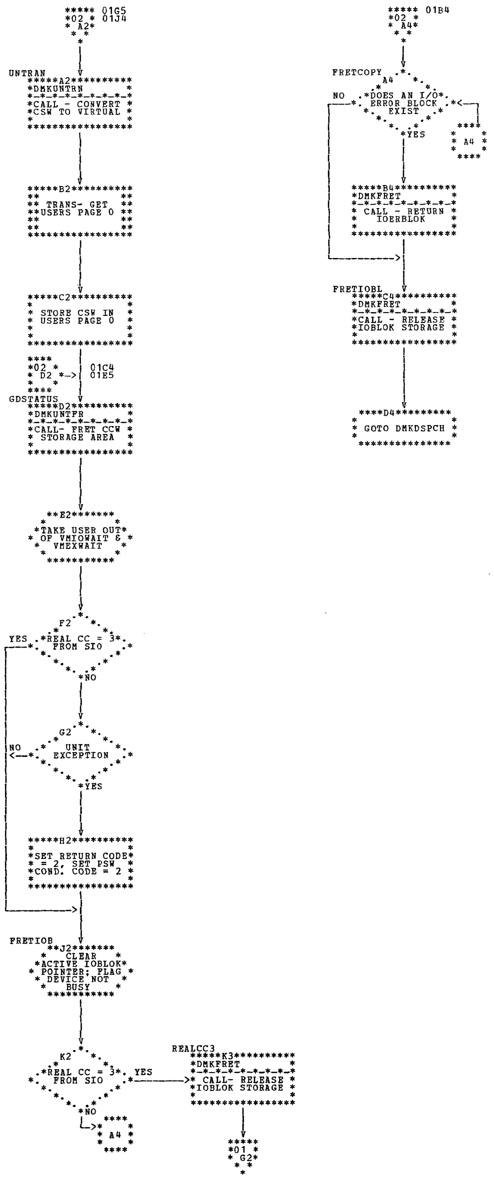

08 A2 07K4
FRET23A
A2 CLEAR PAGE POINTER
B2 DMKPTRFT
CALL - RETURN PAGE TO THE SYSTEM
C2 ADJUST REG FOR NEXT PAGE TO RETURN (IF ANY)
D2 ANY MORE PAGES TO RETURN
YES
NO
E2 RESTORE BALRSAVE (REGISTER SAVE AREA)
F2 RESET FREELOCK TO 0 AND GO EXIT.
02 F1
DMKFRETR
A3 DMKFRETR
B3 SAVE REGISTERS
C3 SET RET FOR DMKFRETR ENTRY
08 D3
09C4
FRET00
D3 SET COMMON ADDRESSABILITY IN R12
E3 REMOVE HIGH ORDER BYTE (IF ANY) FROM R1
F3 ADDRESS 0 OR WITHIN CP NUCLEUS CODE
NO
YES
09 A1
G3 ADDRESS < 4096 (IN PAGE 0)
YES
NO
H3 WAS VIRT=REAL AREA GENERATED
NO
YES
J3 ADDRESS WITHIN VIRT=REAL AREA
NO
YES
K3 DOES VIRT=REAL AREA BELONG TO SYSTEM
NO
YES
ERROR5
K4 ABEND - SVC 0 - ABEND CODE FRE005
09 A1

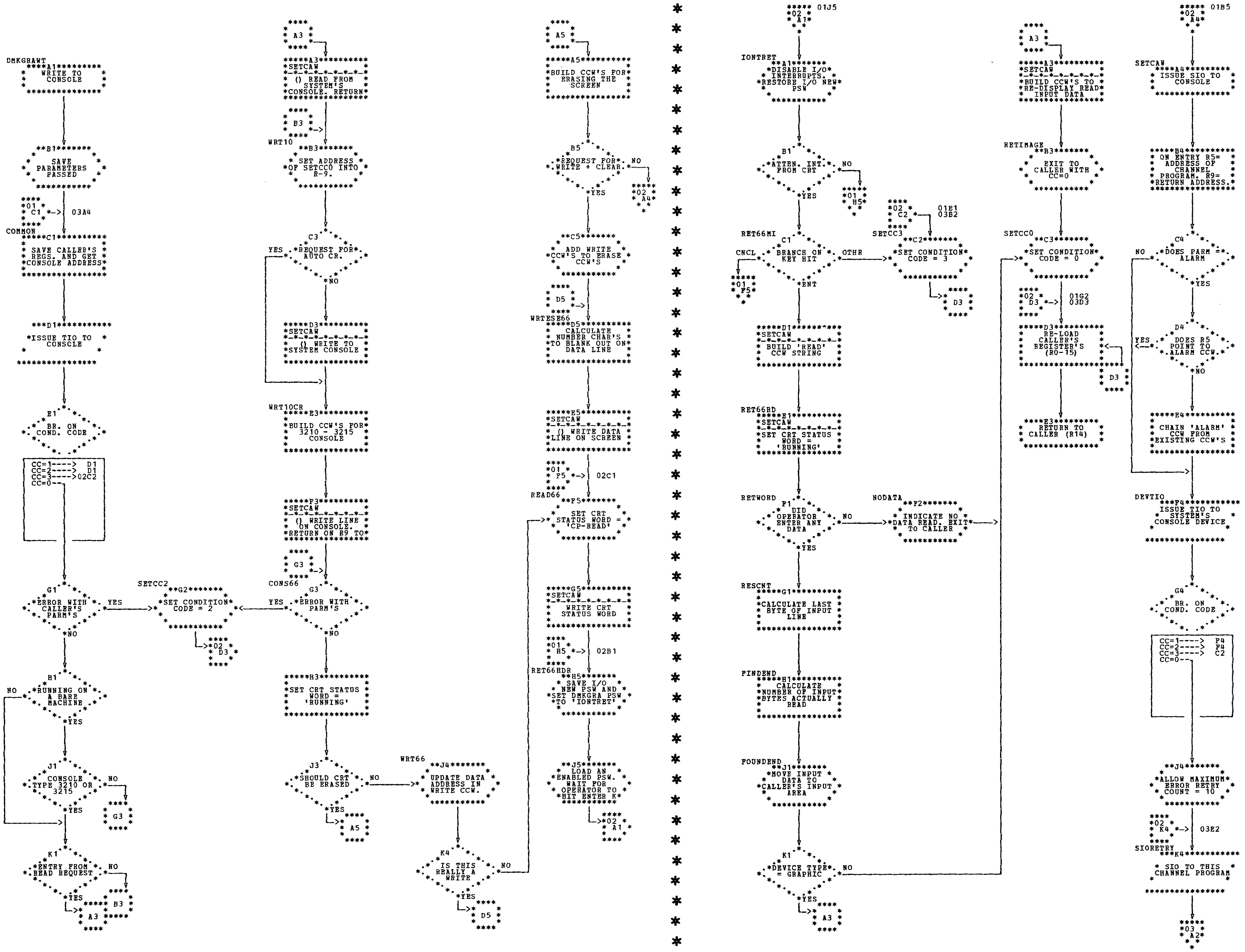

09 A1
08F3
08K3
HIGHCPE
A1
ADDRESS WITHIN DYNAMIC PAGING AREA
NO
YES
B1
COMPUTE HOW MANY PAGES IN USE IN DYNAMIC PAGING AREA
C1
MUST BE PLUS (ERROR IF NOT)
YES
NO
D1
ABEND - SVC 0 - ABEND CODE FRE011
R1OK
C2
CHECK R0 - MUST BE PLUS
NO
YES
ERROR1
C3
ABEND - SVC 0 - ABEND CODE FRE001
D2
"FRET" CALL
NO
YES
01 E1
05 A3
DMKFRET
A4
DMKFRET
B4
SAVE REGISTERS
C4
SET REGS FOR DMKFRET ENTRY
08 D3

| DMKFRE -- Free Storage Manager (Part 9 of 9)

| DMKGEN -- CMS I/O Error Recovery Interface (Parts 1 and 2 of 2)

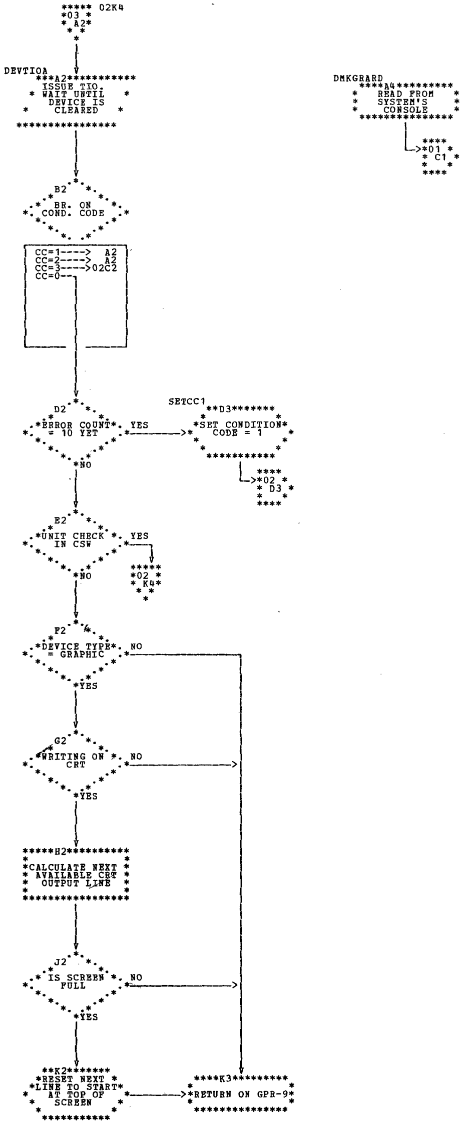

DMKGENIO
A1
ENTRY DMKGENIO
B1
SET VMPSW CONDITION CODE = 0
C1
DMKSCNVU
CALL- LOCATE ALL VIRTUAL BLOCKS
D1
VIRTUAL DEVICE EXISTS
NO
YES
E1
VIRTUAL CHANNEL DEDICATED
YES
NO
F1
DEVICE ATTACHED TO USER
YES
NO
G1
DEVICE = DASDI
NO
YES
A3
01 G2
02K3
NOUNIT
G2
SET RETURN CODE = 1, SET PSW COND. CODE = 1
H2
TAKE USER OUT OF VMIOWAIT & VMEXWAIT
J2
GOTO DMKDSPCH
A3
DEVATTCH
A3
DMKFREE
CALL- GET SIZE OF IOBLOK
B3
CLEAR IOBLOK, STORE VIRTUAL CAW
C3
SET IOBFLAG = IOBHVC
D3
SET IOBIRA = 'DIAGRTN' RETURN ADDRESS
E3
FORCE USER TO WAIT FOR I-O TO FINISH
F3
DMKCCWTR
CALL- BUILD A REAL CCW LIST
G3
STORE ACTIVE IOBLOK POINTER; FLAG DEVICE BUSY
H3
DMKIOSQV
CALL- PUT REQUEST ON QUE
J3
GOTO DMKDSPCH
DIAGRTN
A4
DIAGRTN
B4
DMKIOS BUILD THIS IOBLOK
YES
02 A4
NO
C4
REAL CC = 3 FROM SIO
YES
02 D2
NO
D4
OPERATION FINISH CORRECTLY
YES
NO
E4
SET RETURN CODE = 13, SET PSW COND. CODE = 3
F4
STORE ADDR OF IOERBLOK IN VDEVIOER
G4
AND CLEAR ITS ADDRESS IN THE IOBLOK
H4
SAVE I/O ERROR CSW IN VDEVBLOK
J4
STORE SNS BYTES IN USERS CAW REG.
02 A2
TESTWLR
D5
UNIT EXCEPTION
YES
G5
NO
E5
WRONG LENGTH RECORD
NO
02 D2
YES
F5
SET RETURN CODE = 3, SET PSW COND. CODE = 2
G5
MOVECSW
G5
STORE IOBCSW IN VDEVCSW
02 A2

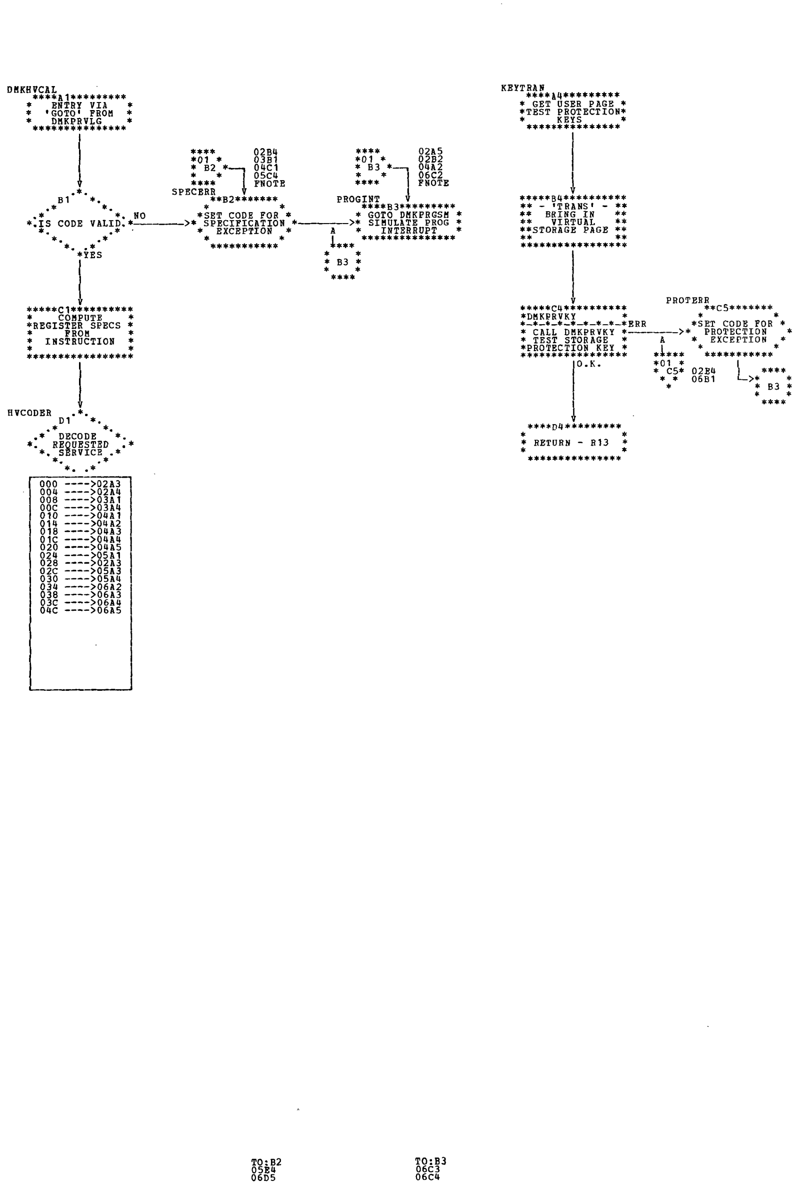

02 A2
01G5
01J4
UNTRAN
A2
DMKUNTRN
CALL - CONVERT CSW TO VIRTUAL
B2
TRANS- GET USERS PAGE 0
C2
STORE CSW IN USERS PAGE 0
02 D2
01C4
01E5
GDSTATUS
D2
DMKUNTFR
CALL- FRET CCW STORAGE AREA
E2
TAKE USER OUT OF VMIOWAIT & VMEXWAIT
F2
REAL CC = 3 FROM SIO
YES
NO
G2
UNIT EXCEPTION
NO
YES
H2
SET RETURN CODE = 2, SET PSW COND. CODE = 2
FRETIOB
J2
CLEAR ACTIVE IOBLOK POINTER; FLAG DEVICE NOT BUSY
K2
REAL CC = 3 FROM SIO
YES
NO
A4
REALCC3
K3
DMKFRET
CALL- RELEASE IOBLOK STORAGE
01 G2
02 A4
01B4
FRETCOPY
A4
DOES AN I/O ERROR BLOCK EXIST
NO
YES
A4
B4
DMKFRET
CALL - RETURN IOERBLOK
FRETIOBL
C4
DMKFRET
CALL - RELEASE IOBLOK STORAGE
D4
GOTO DMKDSPCH

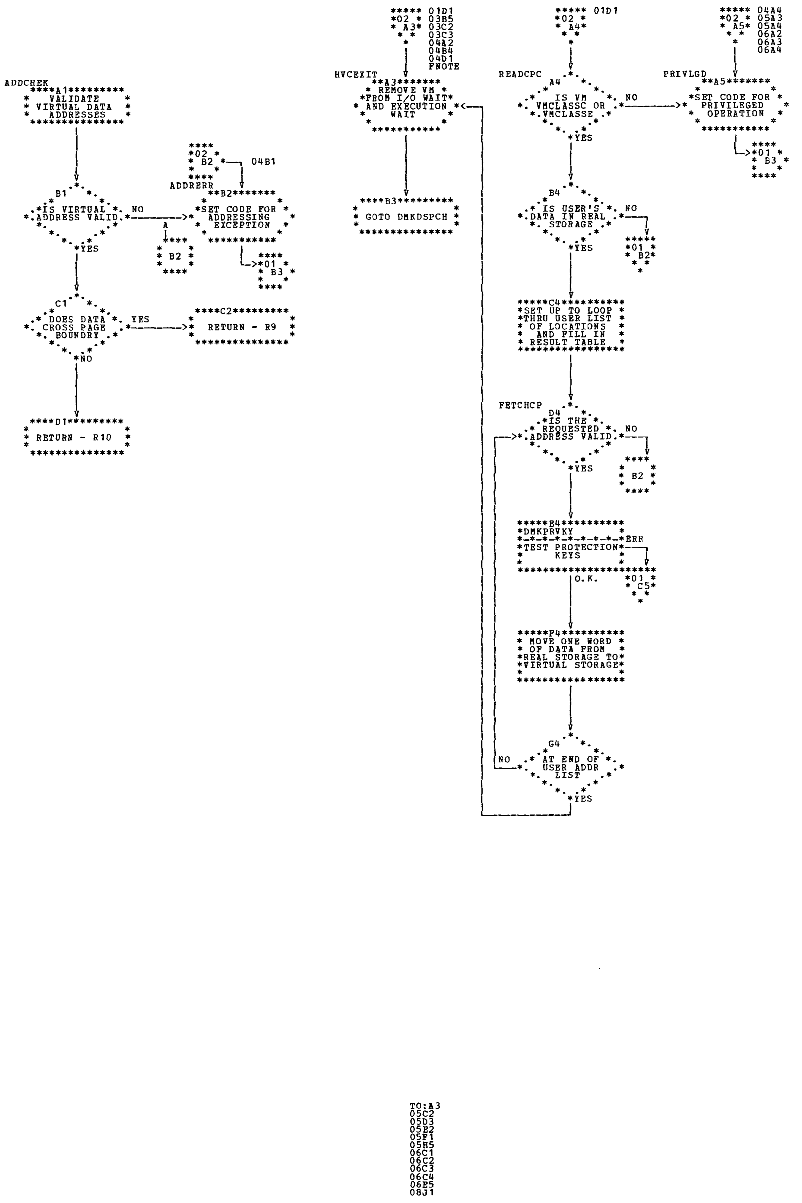

DMKGRAWT
A1 WRITE TO CONSOLE
B1 SAVE PARAMETERS PASSED
01 C1 03A4
COMMON
C1 SAVE CALLER'S REGS. AND GET CONSOLE ADDRESS
D1 ISSUE TIO TO CONSCLE
E1 BR. ON COND. CODE
CC=1----> D1
CC=2----> D1
CC=3---->02C2
CC=0
G1 ERROR WITH CALLER'S PARM'S
YES
NO
SETCC2
G2 SET CONDITION CODE = 2
02 D3
H1 RUNNING ON A BARE MACHINE
NO
YES
J1 CONSOLE TYPE 3210 OR 3215
NO
G3
YES
K1 ENTRY FROM READ REQUEST
NO
B3
YES
A3
A3
A3 SETCAW () READ FROM SYSTEM'S CONSOLE. RETURN
B3
WRT10
B3 SET ADDRESS OF SETCC0 INTO R-9.
C3 REQUEST FOR AUTO CR.
YES
NO
D3 SETCAW () WRITE TO SYSTEM CONSOLE
WRT10CR
E3 BUILD CCW'S FOR 3210 - 3215 CONSOLE
F3 SETCAW () WRITE LINE ON CONSOLE. RETURN ON R9 TO
G3
CONS66
G3 ERROR WITH PARM'S
YES
NO
H3 SET CRT STATUS WORD = 'RUNNING'
J3 SHOULD CRT BE ERASED
NO
YES
A5
WRT66
J4 UPDATE DATA ADDRESS IN WRITE CCW.
K4 IS THIS REALLY A WRITE
NO
YES
D5
A5
A5 BUILD CCW'S FOR ERASING THE SCREEN
B5 REQUEST FOR WRITE + CLEAR.
NO
02 A4
YES
C5 ADD WRITE CCW'S TO ERASE CCW'S
D5
WRTESE66
D5 CALCULATE NUMBER CHAR'S TO BLANK OUT ON DATA LINE
E5 SETCAW () WRITE DATA LINE ON SCREEN
01 F5 02C1
READ66
F5 SET CRT STATUS WORD = 'CP-READ'
G5 SETCAW WRITE CRT STATUS WORD
01 H5 02B1
RET66HDR
H5 SAVE I/O NEW PSW AND SET DMKGRA PSW TO 'IONTRET'
J5 LOAD AN ENABLED PSW. WAIT FOR OPERATOR TO HIT ENTER K
02 A1
02 A1 01J5
IONTRET
A1 DISABLE I/O INTERRUPTS. RESTORE I/O NEW PSW
B1 ATTEN. INT. FROM CRT
NO
01 H5
YES
RET66MI
C1 BRANCH ON KEY HIT
CNCL
01 F5
OTHR
ENT
02 C2 01E1 03B2
SETCC3
C2 SET CONDITION CODE = 3
D3
D1 SETCAW BUILD 'READ' CCW STRING
RET66RD
E1 SETCAW SET CRT STATUS WORD = 'RUNNING'
RETWORD
F1 DID OPERATOR ENTER ANY DATA
NO
YES
NODATA
F2 INDICATE NO DATA READ. EXIT TO CALLER
RESCNT
G1 CALCULATE LAST BYTE OF INPUT LINE
FINDEND
H1 CALCULATE NUMBER OF INPUT BYTES ACTUALLY READ
FOUNDEND
J1 MOVE INPUT DATA TO CALLER'S INPUT AREA
K1 DEVICE TYPE = GRAPHIC
NO
YES
A3
A3
A3 SETCAW BUILD CCW'S TO RE-DISPLAY READ INPUT DATA
RETIMAGE
B3 EXIT TO CALLER WITH CC=0
SETCC0
C3 SET CONDITION CODE = 0
02 D3 01G2 03D3
D3 RE-LOAD CALLER'S REGISTER'S (R0-15)
D3
E3 RETURN TO CALLER (R14)
02 A4 01B5
SETCAW
A4 ISSUE SIO TO CONSOLE
B4 ON ENTRY R5= ADDRESS OF CHANNEL PROGRAM. R9= RETURN ADDRESS.
C4 DOES PARM = ALARM
NO
YES
D4 DOES R5 POINT TO ALARM CCW.
YES
NO
E4 CHAIN 'ALARM' CCW FROM EXISTING CCW'S
DEVTIO
F4 ISSUE TIO TO SYSTEM'S CONSOLE DEVICE
G4 BR. ON COND. CODE
CC=1----> F4
CC=2----> F4
CC=3----> C2
CC=0
J4 ALLOW MAXIMUM ERROR RETRY COUNT = 10
02 K4 03E2
SIORETRY
K4 SIO TO THIS CHANNEL PROGRAM
03 A2

| DMKGRA -- System's Console Routine (Parts 1 and 2 of 3)

| DMKGRA -- System's Console Routine (Part 3 of 3)

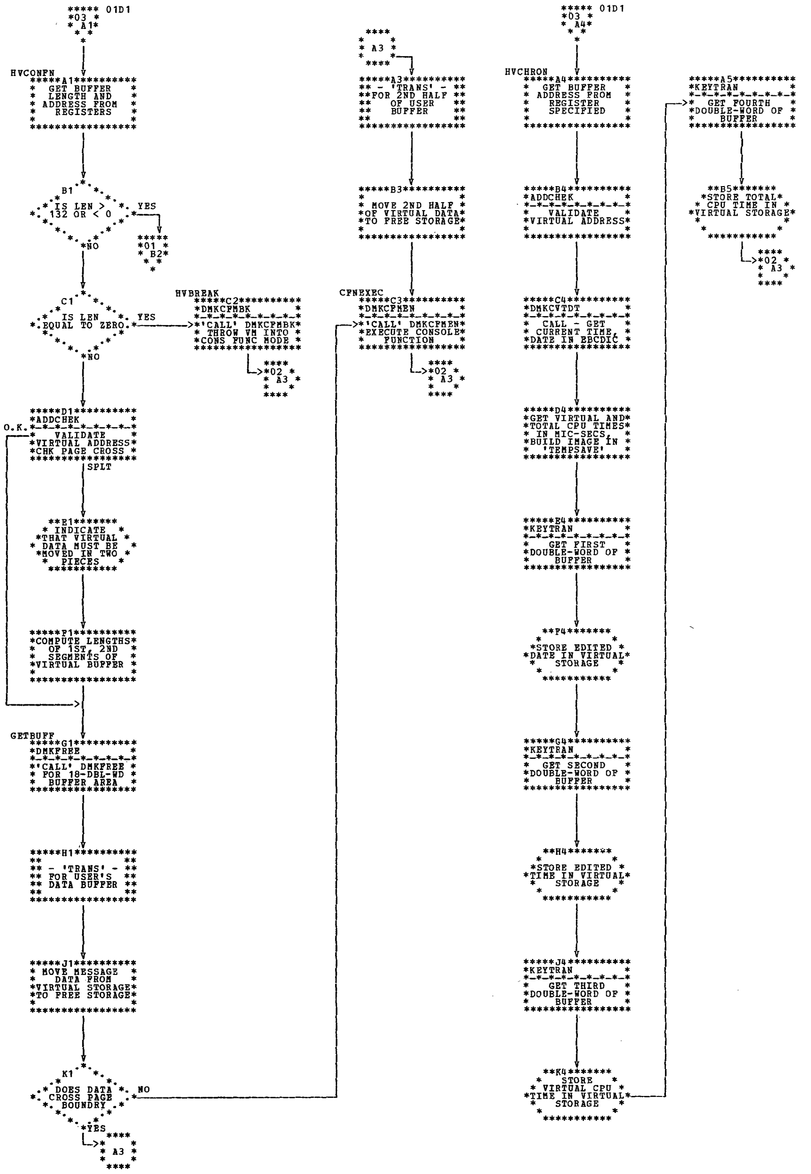


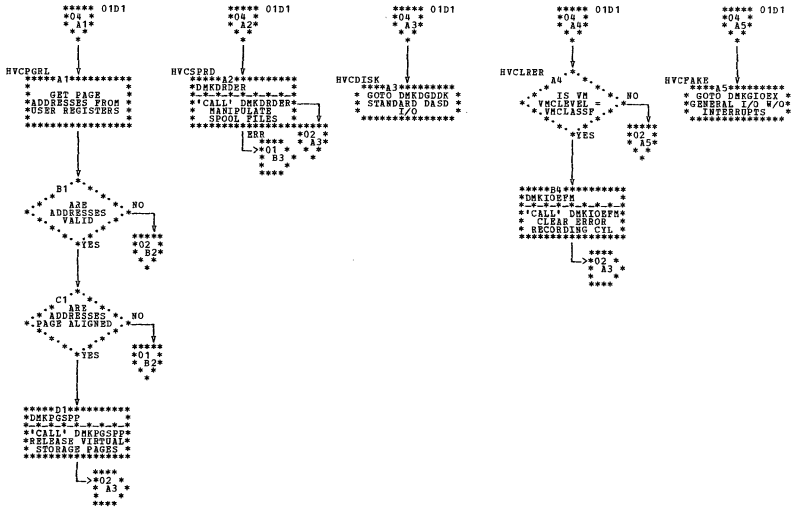


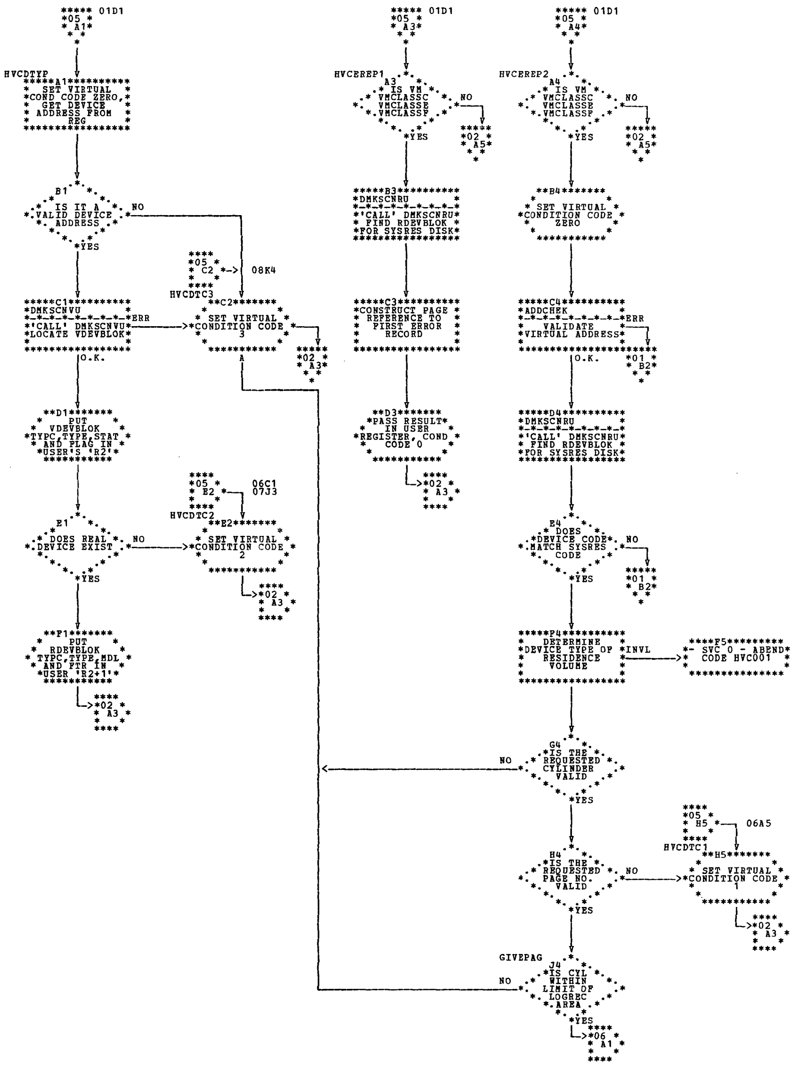

ADDCHEK
A1
VALIDATE VIRTUAL DATA ADDRESSES
02 B2
04B1
B1
IS VIRTUAL ADDRESS VALID.
NO
YES
ADDRERR
B2
SET CODE FOR ADDRESSING EXCEPTION
A
B2
01 B3
C1
DOES DATA CROSS PAGE BOUNDRY
YES
NO
C2
RETURN - R9
D1
RETURN - R10
02 A3
01D1
03B5
03C2
03C3
04A2
04B4
04D1
FNOTE
HVCEXIT
A3
REMOVE VM FROM I/O WAIT AND EXECUTION WAIT
B3
GOTO DMKDSPCH
02 A4
01D1
READCPC
A4
IS VM VMCLASSC OR VMCLASSE
NO
YES
02 A5
04A4
05A3
05A4
06A2
06A3
06A4
PRIVLGD
A5
SET CODE FOR PRIVILEGED OPERATION
01 B3
B4
IS USER'S DATA IN REAL STORAGE
NO
YES
01 B2
C4
SET UP TO LOOP THRU USER LIST OF LOCATIONS AND FILL IN RESULT TABLE
FETCHCP
D4
IS THE REQUESTED ADDRESS VALID.
NO
YES
B2
E4
DMKPRVKY
TEST PROTECTION KEYS
ERR
O.K.
01 C5
F4
MOVE ONE WORD OF DATA FROM REAL STORAGE TO VIRTUAL STORAGE
G4
AT END OF USER ADDR LIST
NO
YES
TO:A3
05C2
05D3
05E2
05F1
05H5
06C1
06C2
06C3
06C4
06E5
08J1

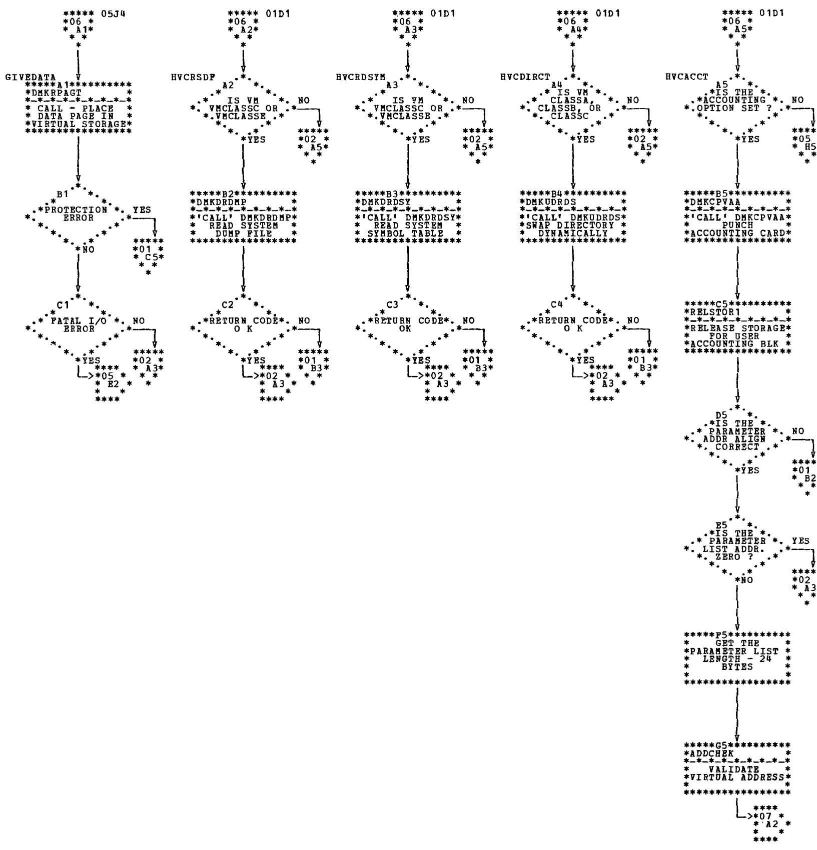
| DMKHVC -- Process DIAGNOSE Instructions (Parts 1 and 2 of 8)

| DMKHVC -- Process DIAGNOSE Instructions (Parts 3 and 4 of 8)


01D1
03 A1
HVCONFN
A1
GET BUFFER LENGTH AND ADDRESS FROM REGISTERS
B1
IS LEN > 132 OR < 0
YES
01 B2
NO
C1
IS LEN EQUAL TO ZERO
YES
HVBREAK
C2
DMKCFMBK
'CALL' DMKCFMBK THROW VM INTO CONS FUNC MODE
02 A3
NO
D1
ADDCHEK
VALIDATE VIRTUAL ADDRESS CHK PAGE CROSS
O.K.
SPLT
E1
INDICATE THAT VIRTUAL DATA MUST BE MOVED IN TWO PIECES
F1
COMPUTE LENGTHS OF 1ST, 2ND SEGMENTS OF VIRTUAL BUFFER
GETBUFF
G1
DMKFREE
'CALL' DMKFREE FOR 18-DBL-WD BUFFER AREA
H1
- 'TRANS' - FOR USER'S DATA BUFFER
J1
MOVE MESSAGE DATA FROM VIRTUAL STORAGE TO FREE STORAGE
K1
DOES DATA CROSS PAGE BOUNDRY
NO
YES
A3
A3
A3
- 'TRANS' - FOR 2ND HALF OF USER BUFFER
B3
MOVE 2ND HALF OF VIRTUAL DATA TO FREE STORAGE
CFNEXEC
C3
DMKCFMEN
'CALL' DMKCFMEN EXECUTE CONSOLE FUNCTION
02 A3
01D1
03 A4
HVCHRON
A4
GET BUFFER ADDRESS FROM REGISTER SPECIFIED
B4
ADDCHEK
VALIDATE VIRTUAL ADDRESS
C4
DMKCVTDT
CALL - GET CURRENT TIME, DATE IN EBCDIC
D4
GET VIRTUAL AND TOTAL CPU TIMES IN MIC-SECS, BUILD IMAGE IN 'TEMPSAVE'
E4
KEYTRAN
GET FIRST DOUBLE-WORD OF BUFFER
F4
STORE EDITED DATE IN VIRTUAL STORAGE
G4
KEYTRAN
GET SECOND DOUBLE-WORD OF BUFFER
H4
STORE EDITED TIME IN VIRTUAL STORAGE
J4
KEYTRAN
GET THIRD DOUBLE-WORD OF BUFFER
K4
STORE VIRTUAL CPU TIME IN VIRTUAL STORAGE
A5
KEYTRAN
GET FOURTH DOUBLE-WORD OF BUFFER
B5
STORE TOTAL CPU TIME IN VIRTUAL STORAGE
02 A3

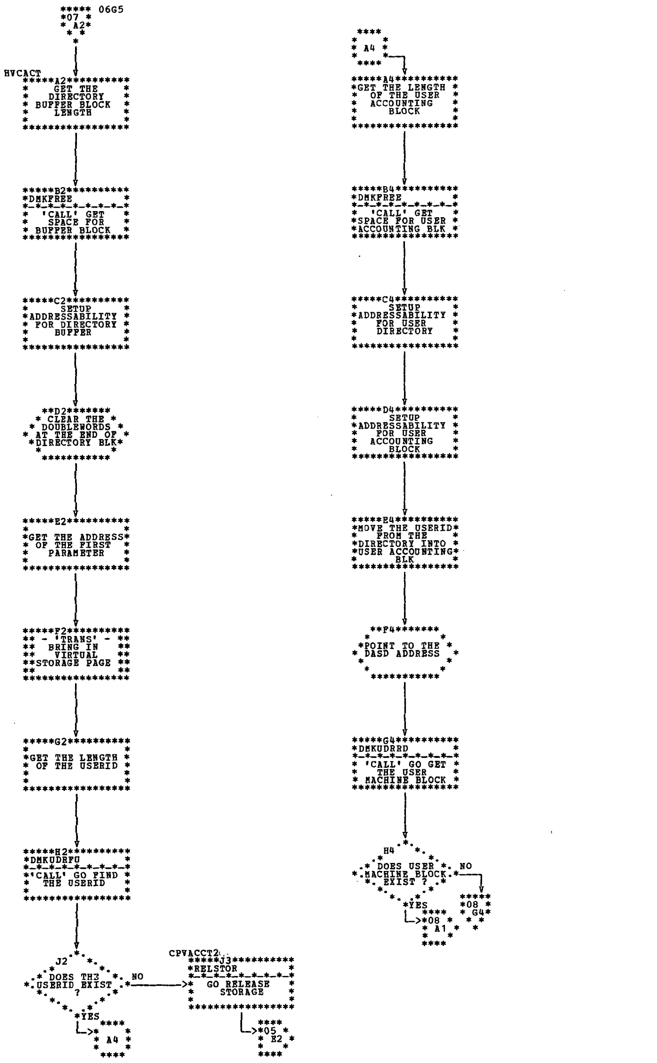

01D1
04 A1
HVCPGRL
A1
GET PAGE ADDRESSES FROM USER REGISTERS
B1
ARE ADDRESSES VALID
NO
02 B2
YES
C1
ARE ADDRESSES PAGE ALIGNED
NO
01 B2
YES
D1
DMKPGSPP
'CALL' DMKPGSPP RELEASE VIRTUAL STORAGE PAGES
02 A3
01D1
04 A2
HVCSPRD
A2
DMKDRDER
'CALL' DMKDRDER MANIPULATE SPOOL FILES
ERR
01 B3
02 A3
01D1
04 A3
HVCDISK
A3
GOTO DMKDGDDK STANDARD DASD I/O
01D1
04 A4
HVCLRER
A4
IS VM VMCLEVEL = VMCLASSF
NO
02 A5
YES
B4
DMKIOEFM
'CALL' DMKIOEFM CLEAR ERROR RECORDING CYL
02 A3
01D1
04 A5
HVCFAKE
A5
GOTO DMKGIOEX GENERAL I/O W/O INTERRUPTS

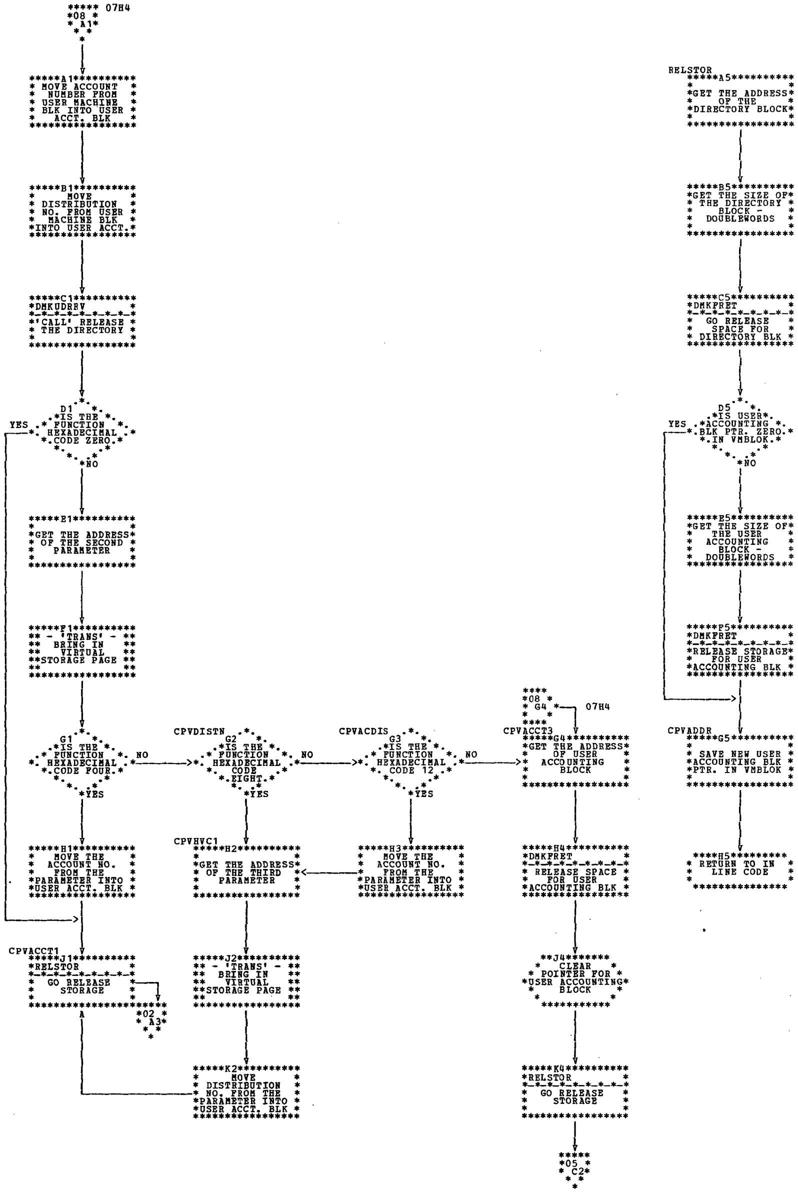

05 A1 01D1
HVCDTYP
A1 SET VIRTUAL COND CODE ZERO, GET DEVICE ADDRESS FROM REG
B1 IS IT A VALID DEVICE ADDRESS
NO
YES
C1 DMKSCNVU 'CALL' DMKSCNVU LOCATE VDEVBLOK
ERR
O.K.
05 C2 08K4
HVCDTC3
C2 SET VIRTUAL CONDITION CODE 3
A
02 A3
D1 PUT VDEVBLOK TYPC,TYPE,STAT AND FLAG IN USER'S 'R2'
05 E2 06C1 07J3
HVCDTC2
E1 DOES REAL DEVICE EXIST
E2 SET VIRTUAL CONDITION CODE 2
02 A3
F1 PUT RDEVBLOK TYPC,TYPE,MDL AND PTR IN USER 'R2+1'
02 A3
05 A3 01D1
HVCEREP1
A3 IS VM VMCLASSC VMCLASSE VMCLASSF
02 A5
B3 DMKSCNRU 'CALL' DMKSCNRU FIND RDEVBLOK FOR SYSRES DISK
C3 CONSTRUCT PAGE REFERENCE TO FIRST ERROR RECORD
D3 PASS RESULT IN USER REGISTER, COND CODE 0
02 A3
05 A4 01D1
HVCEREP2
A4 IS VM VMCLASSC VMCLASSE VMCLASSF
02 A5
B4 SET VIRTUAL CONDITION CODE ZERO
C4 ADDCHEK VALIDATE VIRTUAL ADDRESS
ERR
O.K.
01 B2
D4 DMKSCNRU 'CALL' DMKSCNRU FIND RDEVBLOK FOR SYSRES DISK
E4 DOES DEVICE CODE MATCH SYSRES CODE
01 B2
F4 DETERMINE DEVICE TYPE OF RESIDENCE VOLUME
INVL
F5 SVC 0 - ABEND CODE HVC001
G4 IS THE REQUESTED CYLINDER VALID
05 H5 06A5
HVCDTC1
H4 IS THE REQUESTED PAGE NO. VALID
H5 SET VIRTUAL CONDITION CODE 1
02 A3
GIVEPAG
J4 IS CYL WITHIN LIMIT OF LOGREC AREA
06 A1

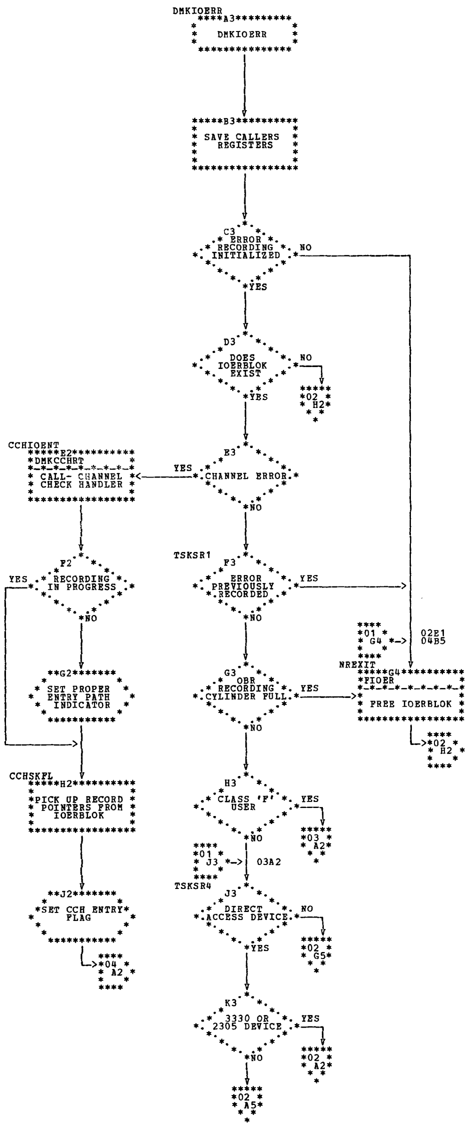

06 A1 05J4
GIVEDATA
A1 DMKRPAGT CALL - PLACE DATA PAGE IN VIRTUAL STORAGE
B1 PROTECTION ERROR
YES
NO
01 C5
C1 FATAL I/O ERROR
02 A3
05 E2
06 A2 01D1
HVCRSDF
A2 IS VM VMCLASSC OR VMCLASSE
02 A5
B2 DMKDRDMP 'CALL' DMKDRDMP READ SYSTEM DUMP FILE
C2 RETURN CODE O K
01 B3
02 A3
06 A3 01D1
HVCRDSYM
A3 IS VM VMCLASSC OR VMCLASSE
02 A5
B3 DMKDRDSY 'CALL' DMKDRDSY READ SYSTEM SYMBOL TABLE
C3 RETURN CODE OK
01 B3
02 A3
06 A4 01D1
HVCDIRCT
A4 IS VM CLASSA, CLASSB, OR CLASSC
02 A5
B4 DMKUDRDS 'CALL' DMKUDRDS SWAP DIRECTORY DYNAMICALLY
C4 RETURN CODE O K
01 B3
02 A3
06 A5 01D1
HVCACCT
A5 IS THE ACCOUNTING OPTION SET ?
05 H5
B5 DMKCPVAA 'CALL' DMKCPVAA PUNCH ACCOUNTING CARD
C5 RELSTOR1 RELEASE STORAGE FOR USER ACCOUNTING BLK
D5 IS THE PARAMETER ADDR ALIGN CORRECT
01 B2
E5 IS THE PARAMETER LIST ADDR. ZERO ?
02 A3
F5 GET THE PARAMETER LIST LENGTH - 24 BYTES
G5 ADDCHEK VALIDATE VIRTUAL ADDRESS
07 A2

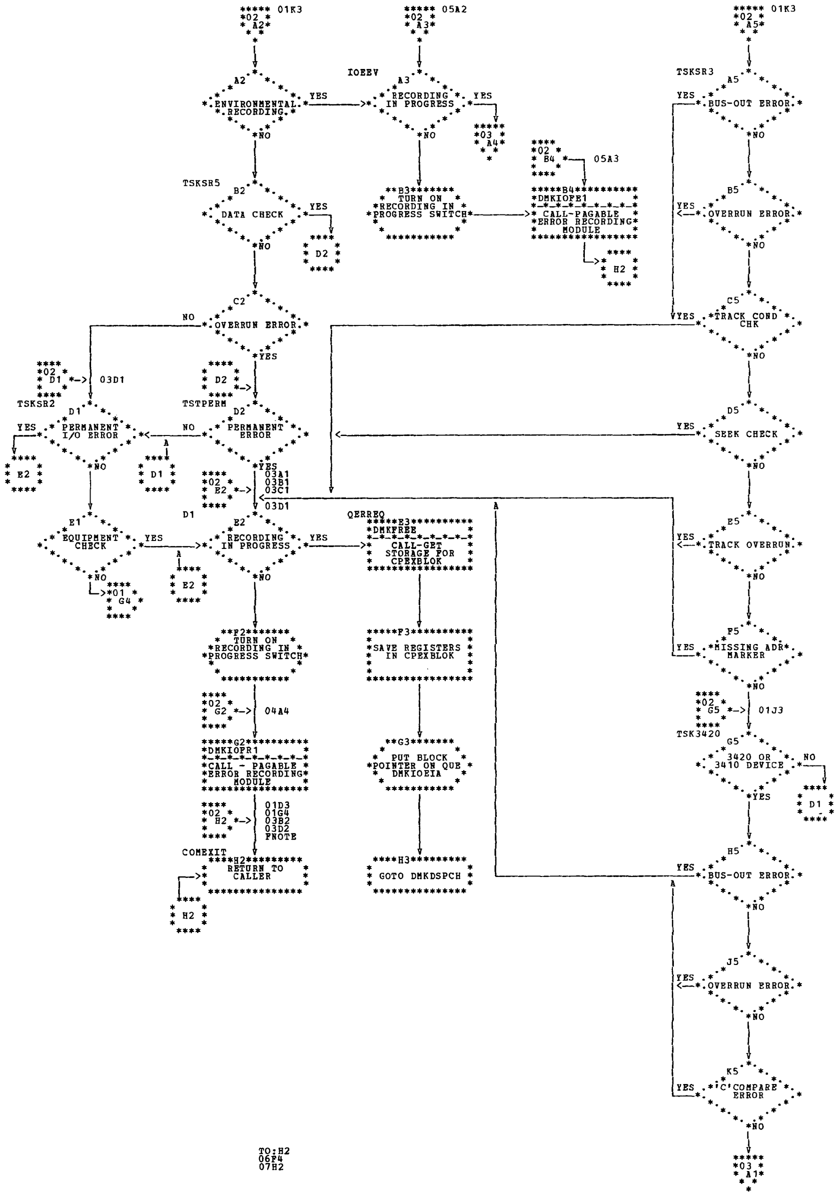
| DMKHVC -- Process DIAGNOSE Instructions (Parts 5 and 6 of 8)

DMKHVC -- Process DIAGNOSE Instructions (Parts 7 and 8 of 8)

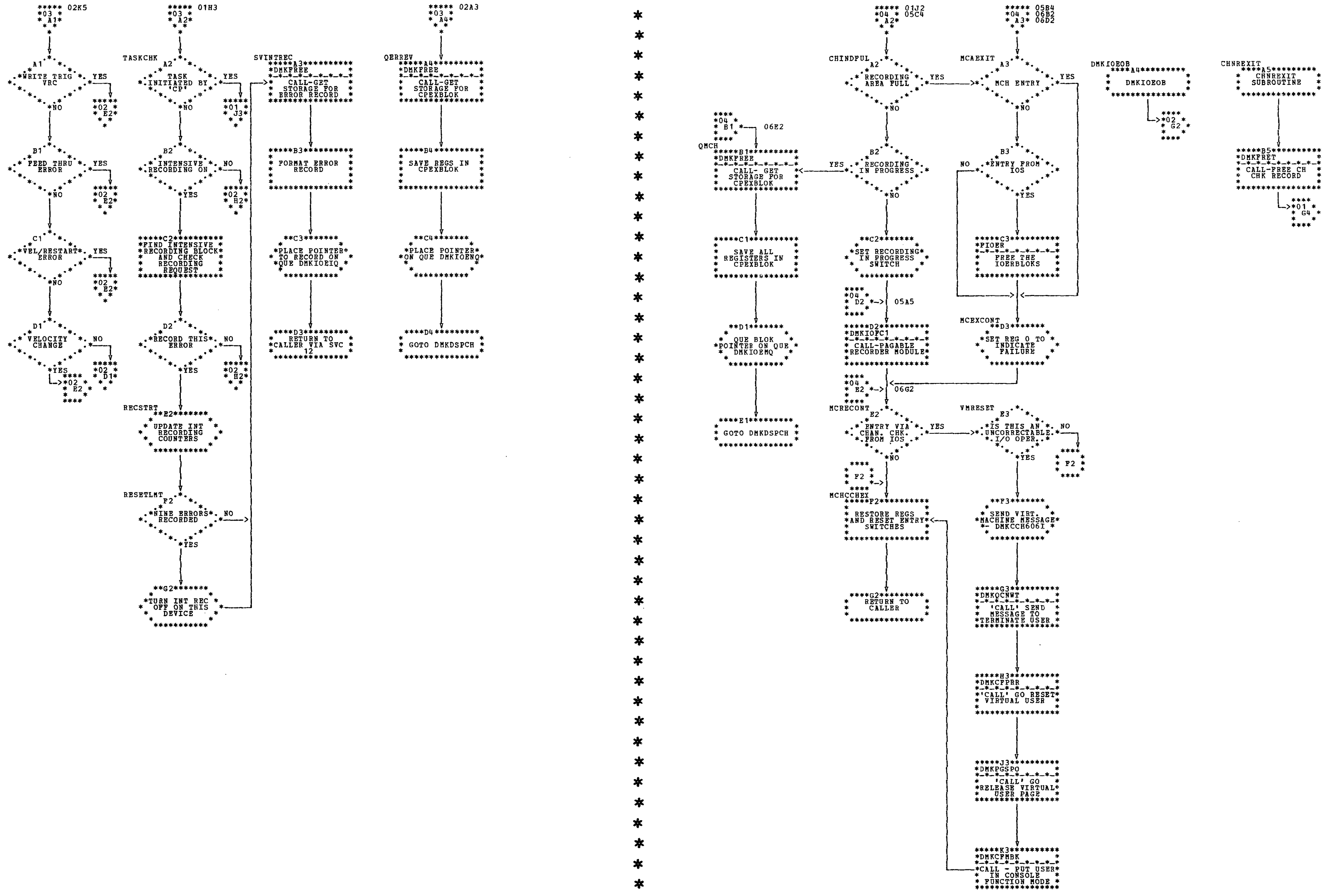

07 A2 06G5
HVCACT
A2 GET THE DIRECTORY BUFFER BLOCK LENGTH
B2 DMKFREE 'CALL' GET SPACE FOR BUFFER BLOCK
C2 SETUP ADDRESSABILITY FOR DIRECTORY BUFFER
D2 CLEAR THE DOUBLEWORDS AT THE END OF DIRECTORY BLK
E2 GET THE ADDRESS OF THE FIRST PARAMETER
F2 - 'TRANS' - BRING IN VIRTUAL STORAGE PAGE
G2 GET THE LENGTH OF THE USERID
H2 DMKUDRFU 'CALL' GO FIND THE USERID
J2 DOES TH3 USERID EXIST ?
NO
YES
A4
CPVACCT2
J3 RELSTOR GO RELEASE STORAGE
05 E2
A4
A4 GET THE LENGTH OF THE USER ACCOUNTING BLOCK
B4 DMKFREE 'CALL' GET SPACE FOR USER ACCOUNTING BLK
C4 SETUP ADDRESSABILITY FOR USER DIRECTORY
D4 SETUP ADDRESSABILITY FOR USER ACCOUNTING BLOCK
E4 MOVE THE USERID FROM THE DIRECTORY INTO USER ACCOUNTING BLK
F4 POINT TO THE DASD ADDRESS
G4 DMKUDRRD 'CALL' GO GET THE USER MACHINE BLOCK
H4 DOES USER MACHINE BLOCK EXIST ?
NO
YES
08 A1
08 G4


08 A1 07H4
A1 MOVE ACCOUNT NUMBER FROM USER MACHINE BLK INTO USER ACCT. BLK
B1 MOVE DISTRIBUTION NO. FROM USER MACHINE BLK INTO USER ACCT.
C1 DMKUDRRV 'CALL' RELEASE THE DIRECTORY
D1 IS THE FUNCTION HEXADECIMAL CODE ZERO
YES
NO
E1 GET THE ADDRESS OF THE SECOND PARAMETER
F1 - 'TRANS' - BRING IN VIRTUAL STORAGE PAGE
G1 IS THE FUNCTION HEXADECIMAL CODE FOUR
NO
YES
H1 MOVE THE ACCOUNT NO. FROM THE PARAMETER INTO USER ACCT. BLK
CPVACCT1
J1 RELSTOR GO RELEASE STORAGE
A
02 A3
CPVDISTN
G2 IS THE FUNCTION HEXADECIMAL CODE EIGHT
NO
YES
CPVHVC1
H2 GET THE ADDRESS OF THE THIRD PARAMETER
J2 - 'TRANS' - BRING IN VIRTUAL STORAGE PAGE
K2 MOVE DISTRIBUTION NO. FROM THE PARAMETER INTO USER ACCT. BLK
CPVACDIS
G3 IS THE FUNCTION HEXADECIMAL CODE 12
NO
YES
H3 MOVE THE ACCOUNT NO. FROM THE PARAMETER INTO USER ACCT. BLK
08 G4 07H4
CPVACCT3
G4 GET THE ADDRESS OF USER ACCOUNTING BLOCK
H4 DMKFRET RELEASE SPACE FOR USER ACCOUNTING BLK
J4 CLEAR POINTER FOR USER ACCOUNTING BLOCK
K4 RELSTOR GO RELEASE STORAGE
05 C2
RELSTOR
A5 GET THE ADDRESS OF THE DIRECTORY BLOCK
B5 GET THE SIZE OF THE DIRECTORY BLOCK - DOUBLEWORDS
C5 DMKFRET GO RELEASE SPACE FOR DIRECTORY BLK
D5 IS USER ACCOUNTING BLK PTR. ZERO IN VMBLOK
YES
NO
E5 GET THE SIZE OF THE USER ACCOUNTING BLOCK - DOUBLEWORDS
F5 DMKFRET RELEASE STORAGE FOR USER ACCOUNTING BLK
CPVADDR
G5 SAVE NEW USER ACCOUNTING BLK PTR. IN VMBLOK
H5 RETURN TO IN LINE CODE



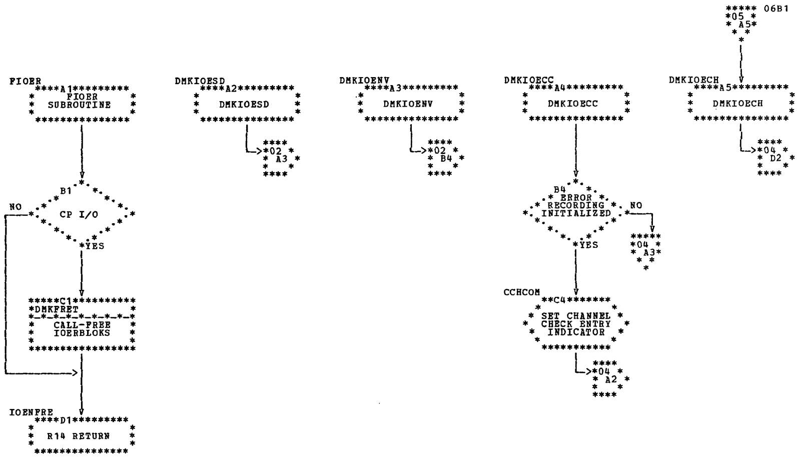


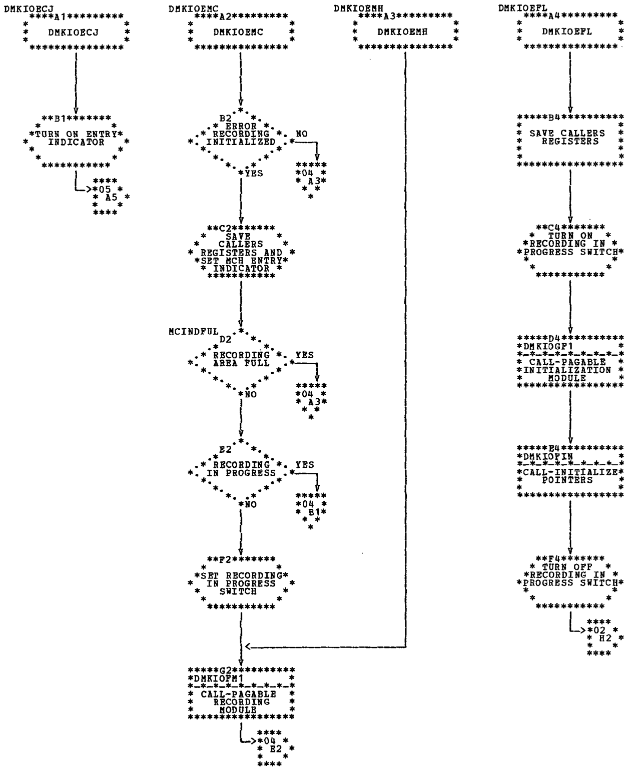
| DMKIOE -- Main Error Recording Processor (Parts 1 and 2 of 7)

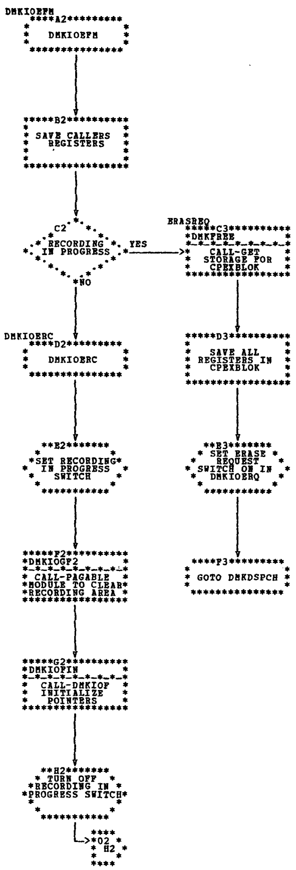

02K5
03 A1
A1 WRITE TRIG VRC
YES
02 E2
NO
B1 FEED THRU ERROR
YES
02 E2
NO
C1 VEL/RESTART ERROR
YES
02 E2
NO
D1 VELOCITY CHANGE
NO
02 D1
YES
02 E2
01H3
03 A2
TASKCHK
A2 TASK INITIATED BY 'CP'
YES
01 J3
NO
B2 INTENSIVE RECORDING ON
NO
02 H2
YES
C2 FIND INTENSIVE RECORDING BLOCK AND CHECK RECORDING REQUEST
D2 RECORD THIS ERROR
NO
02 H2
YES
RECSTRT
E2 UPDATE INT RECORDING COUNTERS
RESETLMT
F2 NINE ERRORS RECORDED
NO
YES
G2 TURN INT REC OFF ON THIS DEVICE
SVINTREC
A3 DMKFREE
CALL-GET STORAGE FOR ERROR RECORD
B3 FORMAT ERROR RECORD
C3 PLACE POINTER TO RECORD ON QUE DMKIOEIQ
D3 RETURN TO CALLER VIA SVC 12
02A3
03 A4
QERREV
A4 DMKFREE
CALL-GET STORAGE FOR CPEXBLOK
B4 SAVE REGS IN CPEXBLOK
C4 PLACE POINTER ON QUE DMKIOENQ
D4 GOTO DMKDSPCH
01J2
04 A2
05C4
CHINDFUL
A2 RECORDING AREA FULL
YES
NO
04 B1
06E2
QMCH
B1 DMKFREE
CALL- GET STORAGE FOR CPEXBLOK
B2 RECORDING IN PROGRESS
YES
NO
C1 SAVE ALL REGISTERS IN CPEXBLOK
D1 QUE BLOK POINTER ON QUE DMKIOEMQ
E1 GOTO DMKDSPCH
C2 SET RECORDING IN PROGRESS SWITCH
04 D2
05A5
D2 DMKIOPC1
CALL-PAGABLE RECORDER MODULE
04 E2
06G2
MCRECONT
E2 ENTRY VIA CHAN. CHK. FROM IOS
YES
NO
F2
MCHCCHEX
F2 RESTORE REGS AND RESET ENTRY SWITCHES
G2 RETURN TO CALLER
05B4
04 A3
06B2
06D2
MCAEXIT
A3 MCH ENTRY
YES
NO
B3 ENTRY FROM IOS
NO
YES
C3 FIOER
FREE THE IOERBLOKS
MCEXCONT
D3 SET REG 0 TO INDICATE FAILURE
VMRESET
E3 IS THIS AN UNCORRECTABLE I/O OPER.
NO
F2
YES
F3 SEND VIRT. MACHINE MESSAGE - DMKCCH606I
G3 DMKQCNWT
'CALL' SEND MESSAGE TO TERMINATE USER
H3 DMKCFPRR
'CALL' GO RESET VIRTUAL USER
J3 DMKPGSPO
'CALL' GO RELEASE VIRTUAL USER PAGE
K3 DMKCFMBK
CALL - PUT USER IN CONSOLE FUNCTION MODE
DMKIOEOB
A4 DMKIOEOB
02 G2
CHNREXIT
A5 CHNREXIT SUBROUTINE
B5 DMKFRET
CALL-FREE CH CHK RECORD
01 G4

| DMKIOE -- Main Error Recording Processor (Parts 3 and 4 of 7)

| DMKIOE -- Main Error Recording Processor (Parts 5 and 6 of 7)

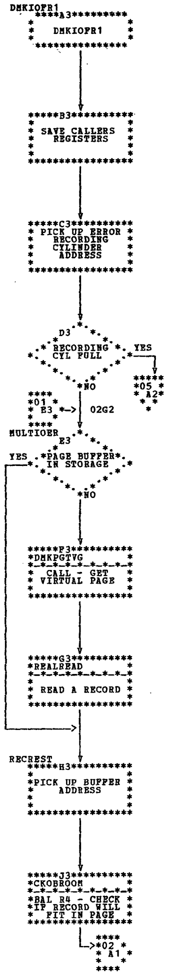


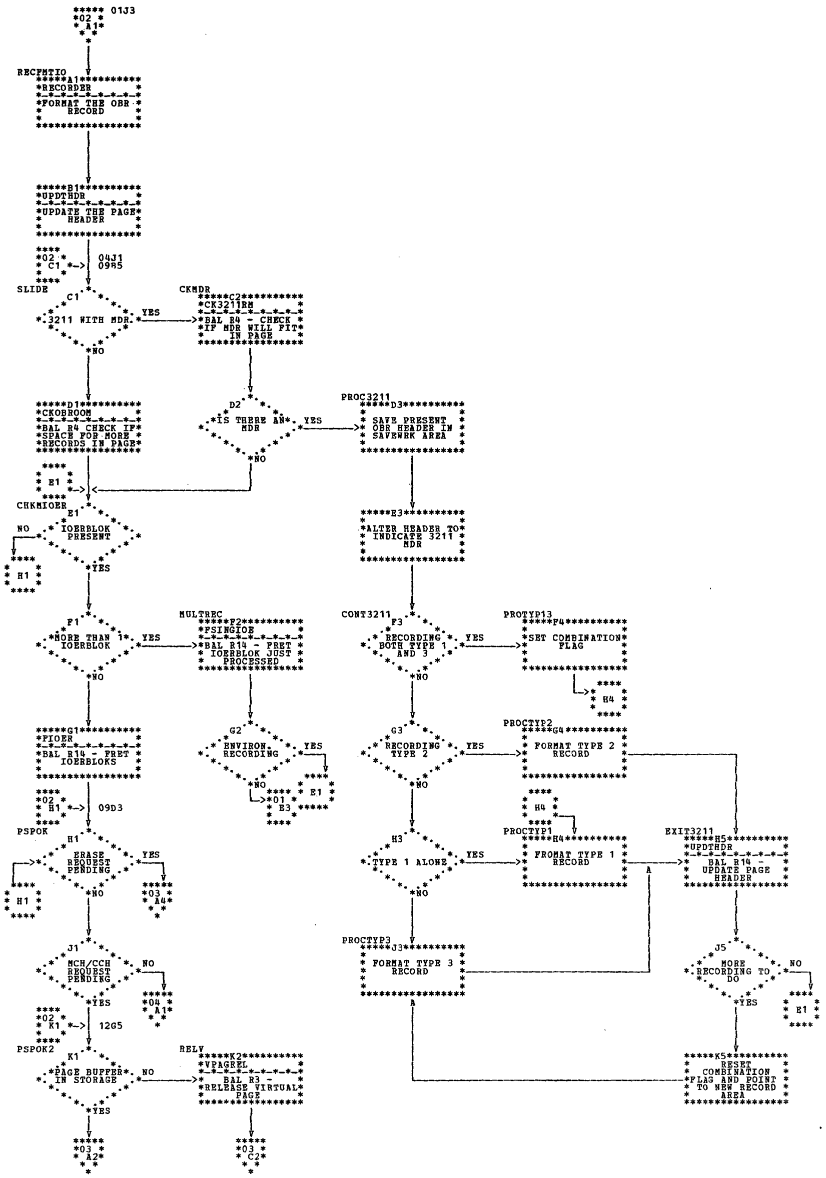


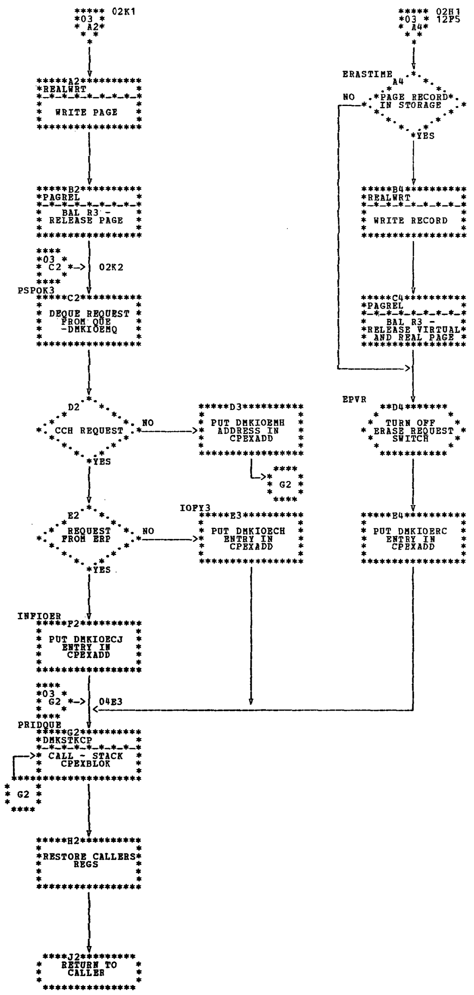

DMKIOEFM
A2
DMKIOEFM
B2
SAVE CALLERS REGISTERS
C2
RECORDING IN PROGRESS
YES
NO
ERASREQ
C3
DMKFREE
CALL-GET STORAGE FOR CPEXBLOK
D3
SAVE ALL REGISTERS IN CPEXBLOK
E3
SET ERASE REQUEST SWITCH ON IN DMKIOERQ
F3
GOTO DMKDSPCH
DMKIOERC
D2
DMKIOERC
E2
SET RECORDING IN PROGRESS SWITCH
F2
DMKIOGF2
CALL-PAGABLE MODULE TO CLEAR RECORDING AREA
G2
DMKIOFIN
CALL-DMKIOF INITIALIZE POINTERS
H2
TURN OFF RECORDING IN PROGRESS SWITCH
02
H2

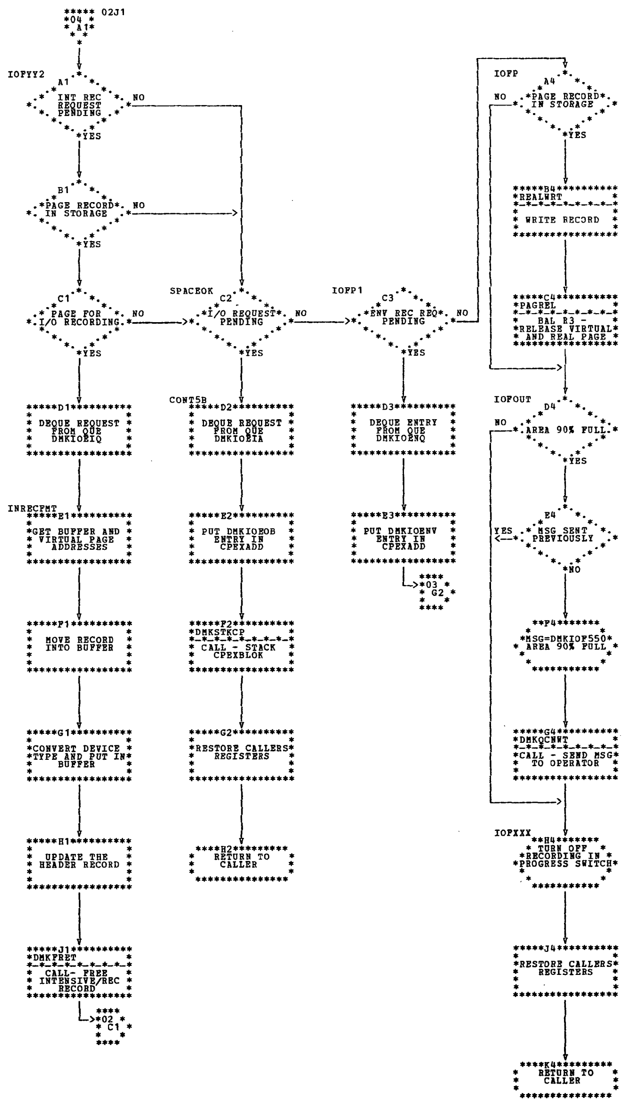
DMKIOE -- Main Error Recording Processor (Part 7 of 7)

| DMKIOF -- Error Recorder (Parts 1 and 2 of 13)



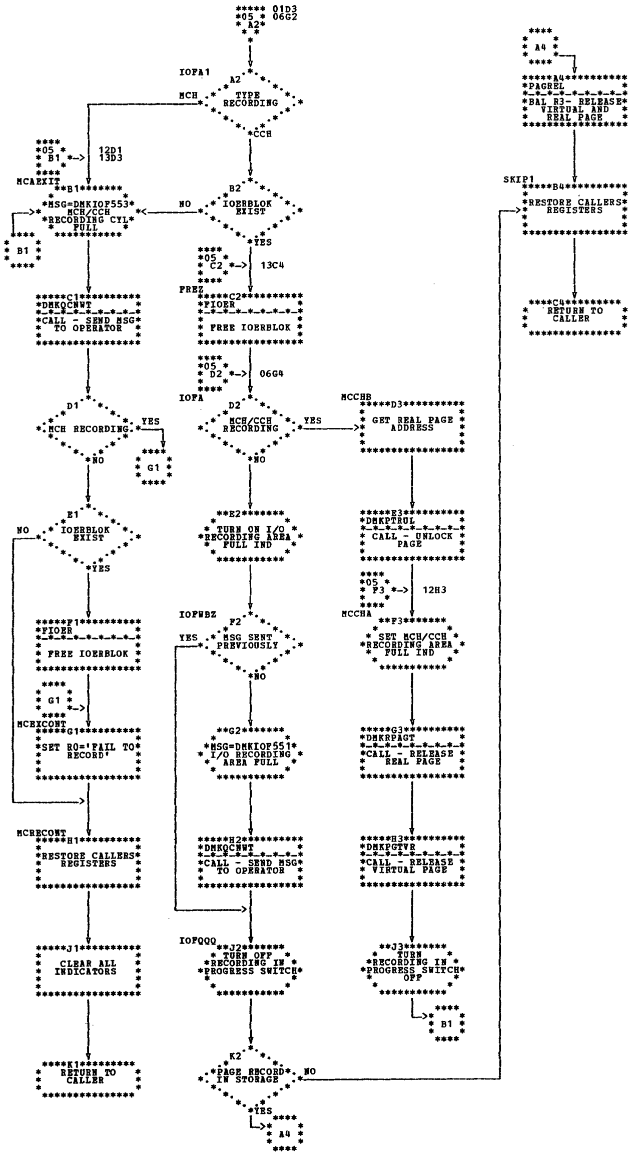


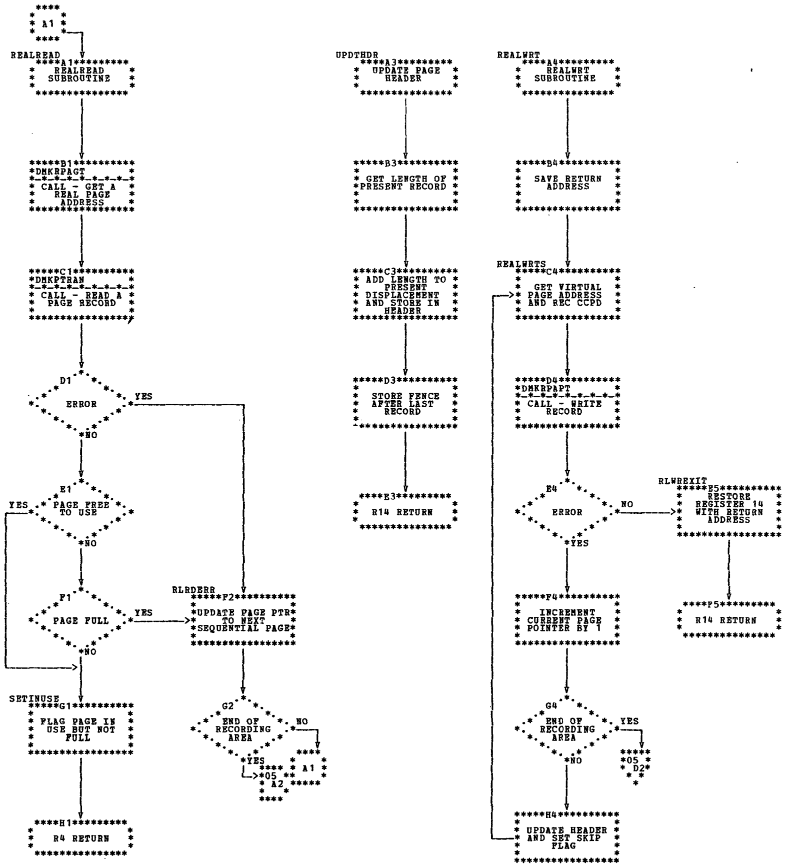


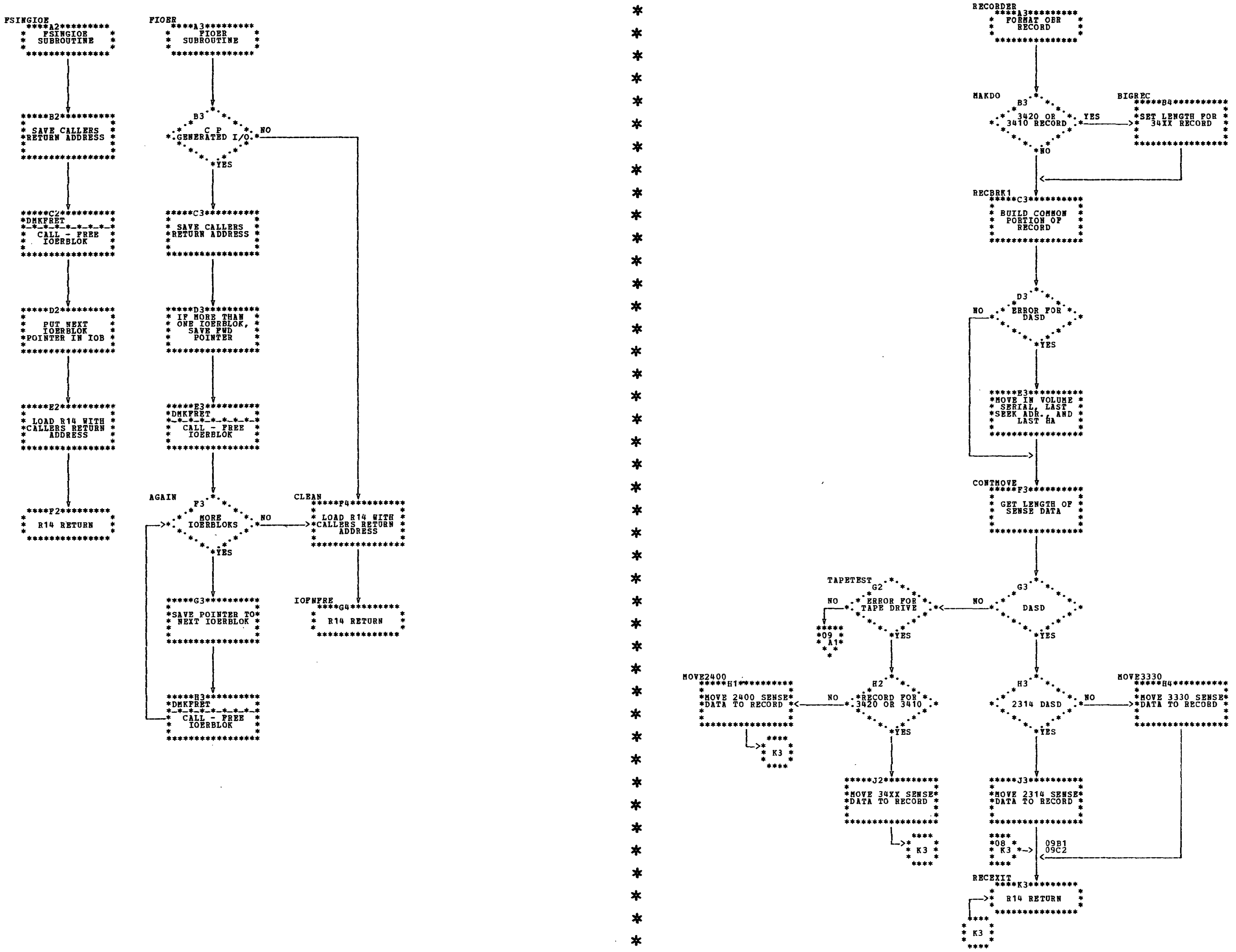


| DMKIOF -- Errcr Recorder (Parts 3 and 4 of 13)

| DMKIOF -- Error Recorder (Parts 5 and 6 of 13)

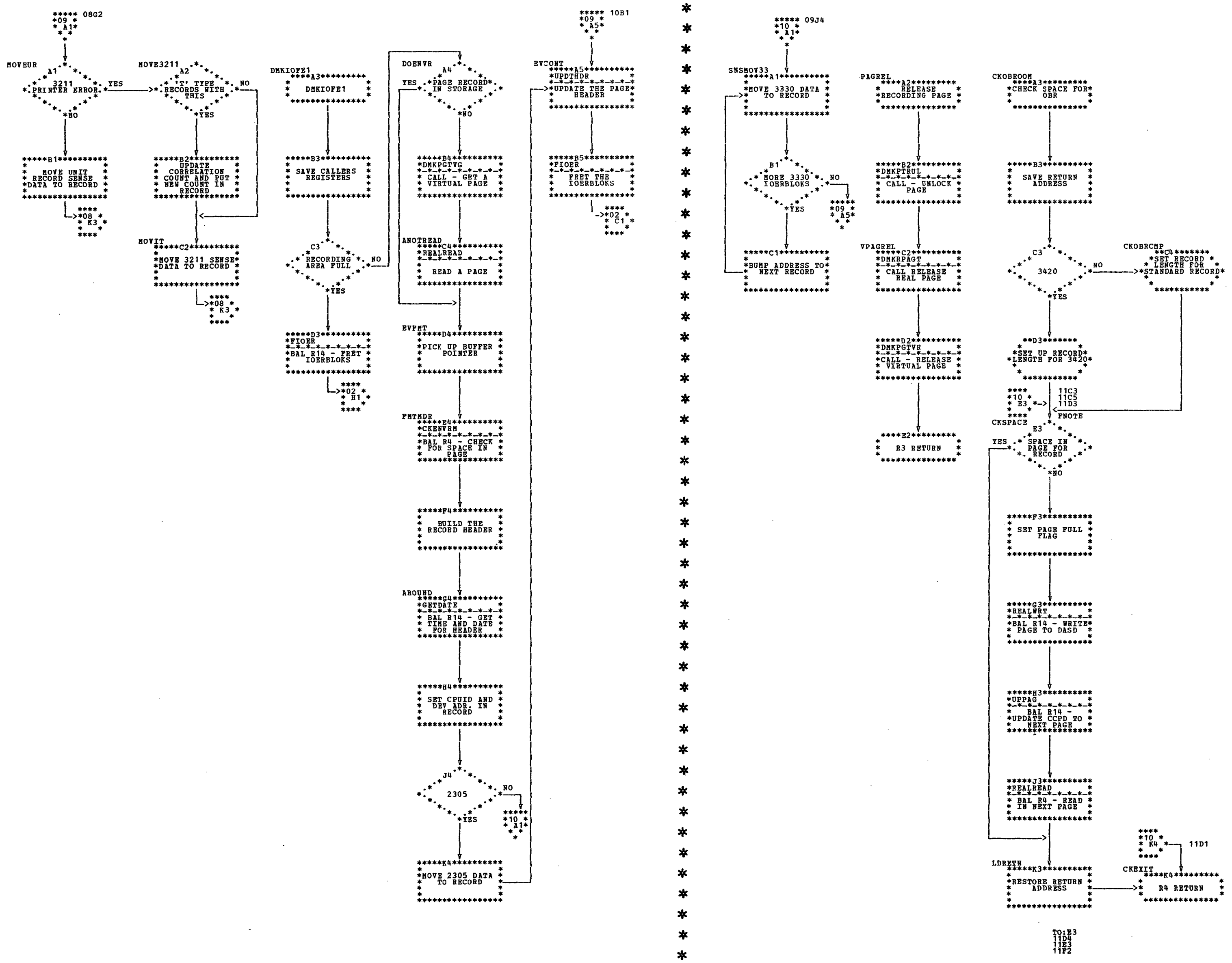

05 A2 01D3 06G2
IOFA1
A2 TYPE RECORDING
MCH
CCH
05 B1 12D1 13D3
MCAEXIT
B1 MSG=DMKIOF553 MCH/CCH RECORDING CYL FULL
B1
B2 IOERBLOK EXIST
NO
YES
05 C2 13C4
FREZ
C2 FIOER FREE IOERBLOK
C1 DMKQCNWT CALL - SEND MSG TO OPERATOR
05 D2 06G4
IOFA
D1 MCH RECORDING
YES
G1
NO
D2 MCH/CCH RECORDING
YES
NO
MCCHB
D3 GET REAL PAGE ADDRESS
E1 IOERBLOK EXIST
NO
YES
E2 TURN ON I/O RECORDING AREA FULL IND
E3 DMKPTRUL CALL - UNLOCK PAGE
05 F3 12H3
MCCHA
F3 SET MCH/CCH RECORDING AREA FULL IND
IOFWBZ
F2 MSG SENT PREVIOUSLY
YES
NO
F1 FIOER FREE IOERBLOK
G1
MCEXCONT
G1 SET R0='FAIL TO RECORD'
G2 MSG=DMKIOF551 I/O RECORDING AREA FULL
G3 DMKRPAGT CALL - RELEASE REAL PAGE
MCRECONT
H1 RESTORE CALLERS REGISTERS
H2 DMKQCNWT CALL - SEND MSG TO OPERATOR
H3 DMKPGTVR CALL - RELEASE VIRTUAL PAGE
J1 CLEAR ALL INDICATORS
IOFQQQ
J2 TURN OFF RECORDING IN PROGRESS SWITCH
J3 TURN RECORDING IN PROGRESS SWITCH OFF
B1
K1 RETURN TO CALLER
K2 PAGE RECORD IN STORAGE
NO
YES
A4
A4
A4 PAGREL BAL R3- RELEASE VIRTUAL AND REAL PAGE
SKIP1
B4 RESTORE CALLERS REGISTERS
C4 RETURN TO CALLER

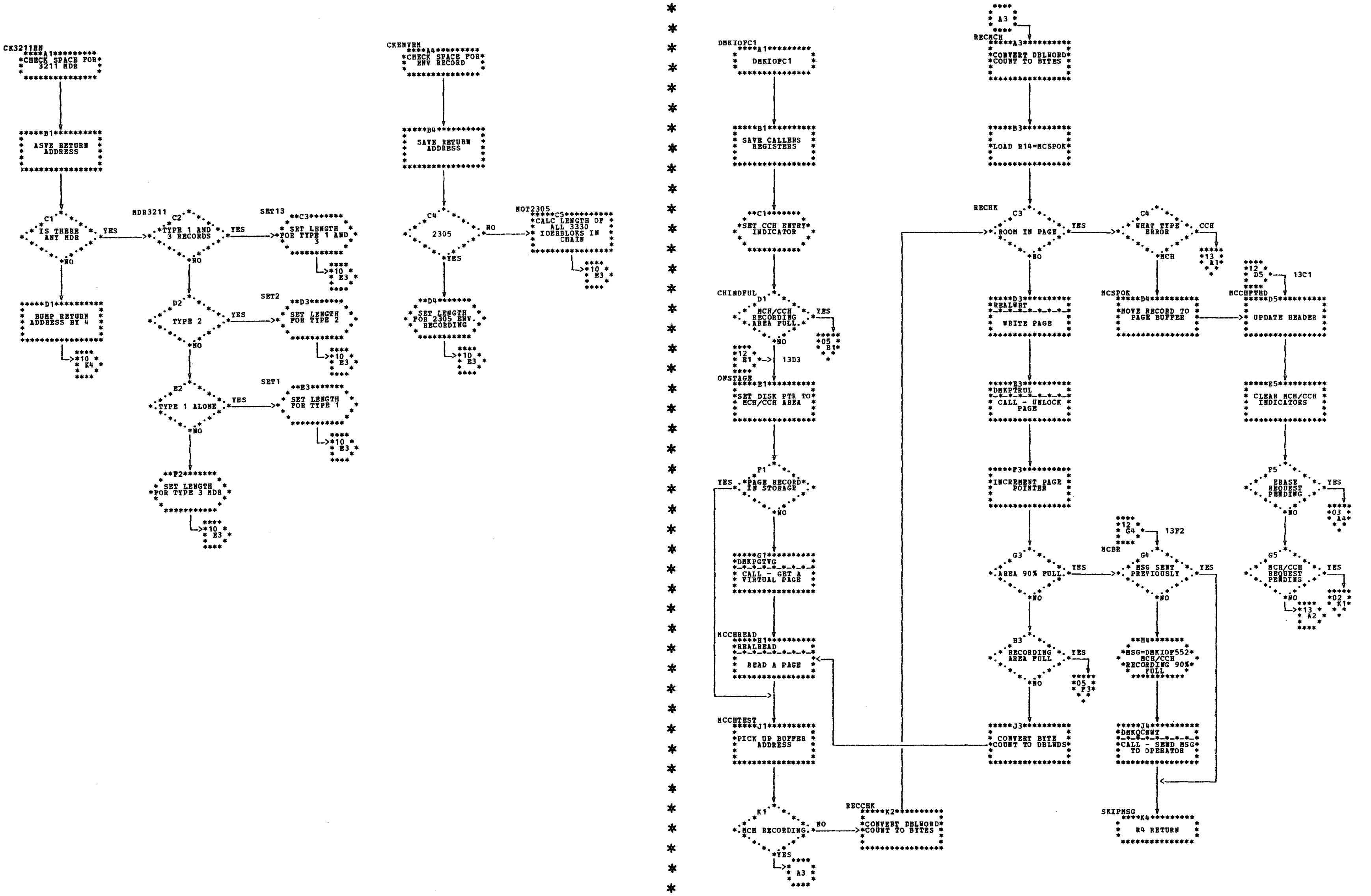

A1
REALREAD
A1 REALREAD SUBROUTINE
B1 DMKRPAGT CALL - GET A REAL PAGE ADDRESS
C1 DMKPTRAN CALL - READ A PAGE RECORD
D1 ERROR
YES
NO
E1 PAGE FREE TO USE
YES
NO
F1 PAGE FULL
YES
NO
RLRDERR
F2 UPDATE PAGE PTR TO NEXT SEQUENTIAL PAGE
G2 END OF RECORDING AREA
NO
A1
YES
05 A2
SETINUSE
G1 FLAG PAGE IN USE BUT NOT FULL
H1 R4 RETURN
UPDTHDR
A3 UPDATE PAGE HEADER
B3 GET LENGTH OF PRESENT RECORD
C3 ADD LENGTH TO PRESENT DISPLACEMENT AND STORE IN HEADER
D3 STORE FENCE AFTER LAST RECORD
E3 R14 RETURN
REALWRT
A4 REALWRT SUBROUTINE
B4 SAVE RETURN ADDRESS
REALWRTS
C4 GET VIRTUAL PAGE ADDRESS AND REC CCPD
D4 DMKRPAPT CALL - WRITE RECORD
E4 ERROR
NO
YES
RLWREXIT
E5 RESTORE REGISTER 14 WITH RETURN ADDRESS
F5 R14 RETURN
F4 INCREMENT CURRENT PAGE POINTER BY 1
G4 END OF RECORDING AREA
YES
05 D2
NO
H4 UPDATE HEADER AND SET SKIP FLAG



| DMKIOF -- Error Recorder (Parts 7 and 8 of 13)

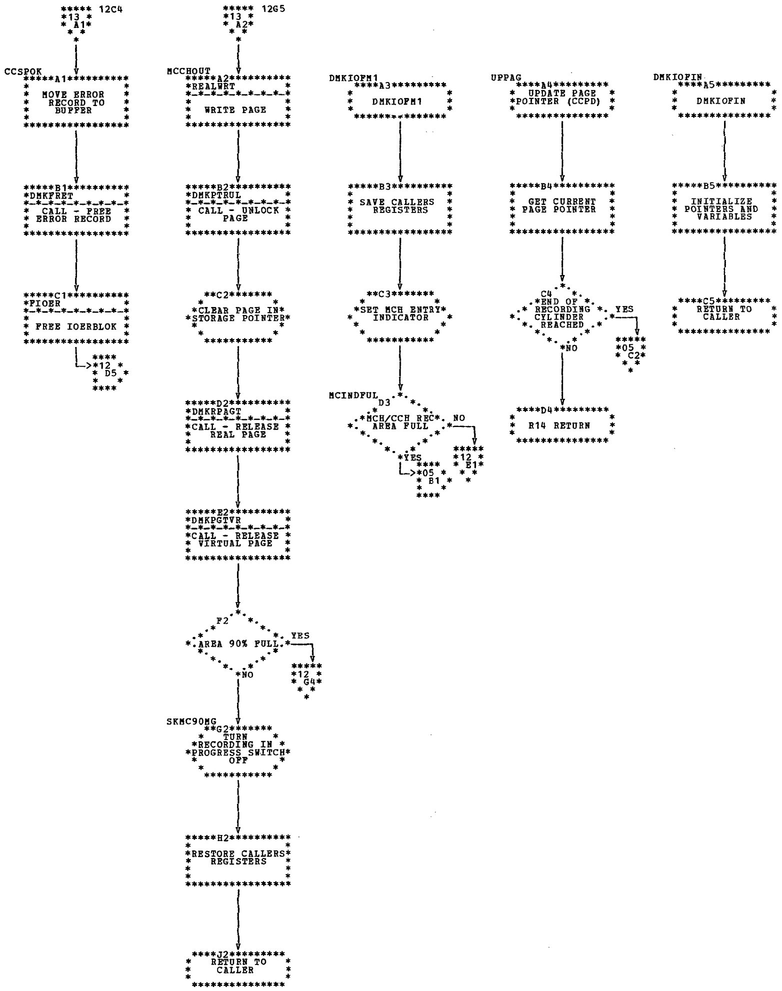
| DMKIOF -- Error Recorder (Parts 9 and 10 of 13)

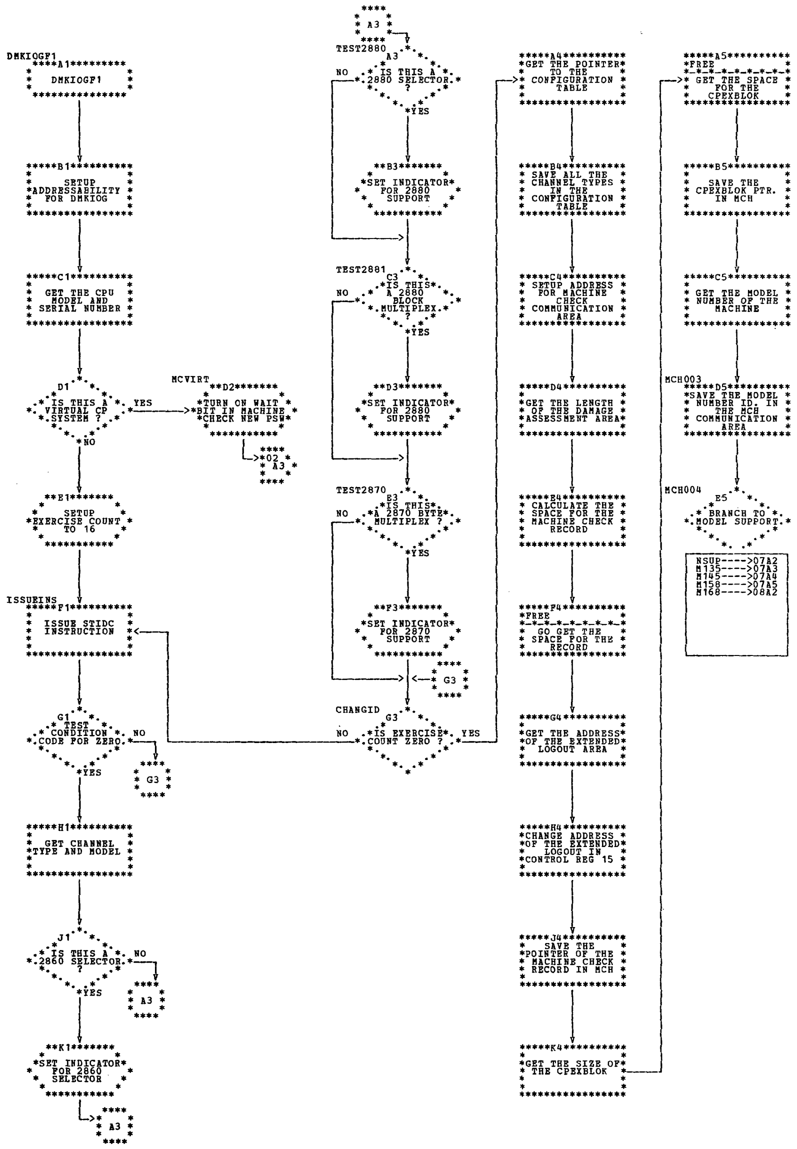


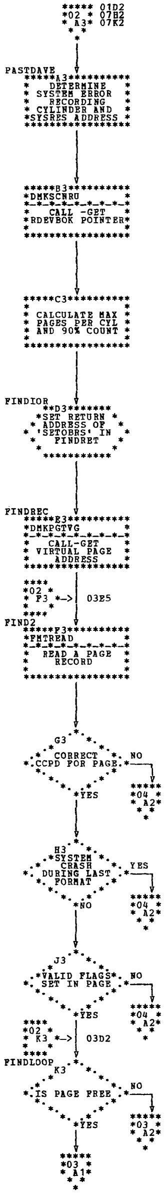


| DMKIOF -- Errcr Recorder (Parts 11 and 12 of 13)

| DMKIOF -- Error Recorder (Parts 13 of 13)

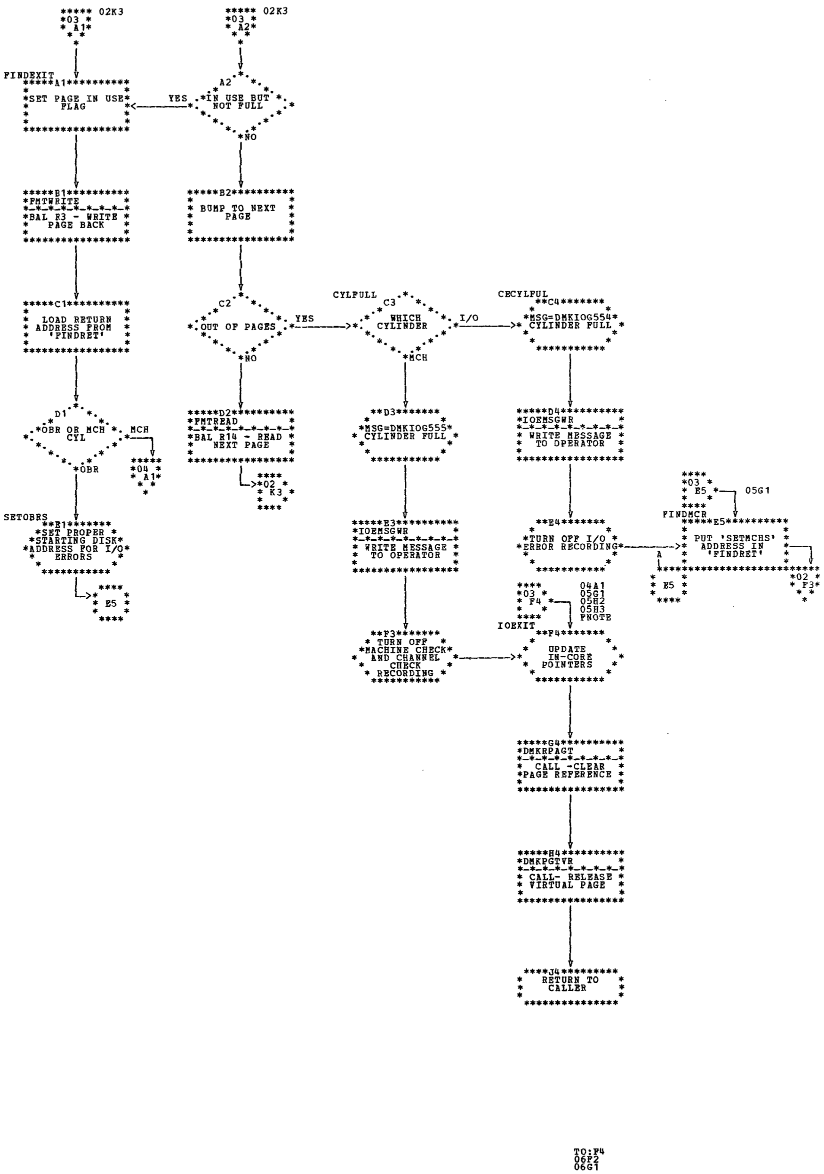


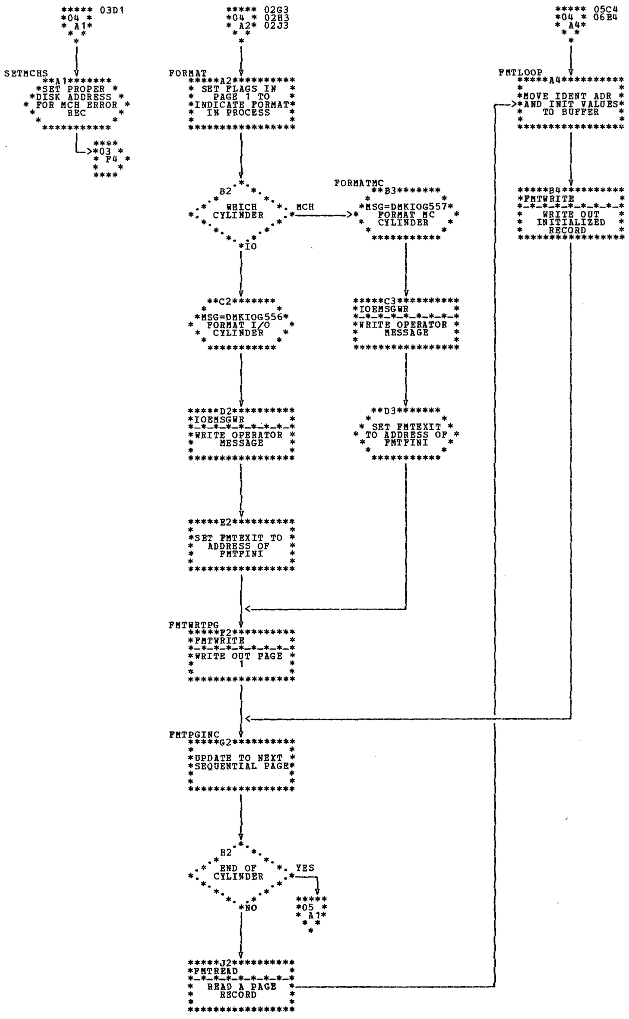

DMKIOGF1
A1 DMKIOGF1
B1 SETUP ADDRESSABILITY FOR DMKIOG
C1 GET THE CPU MODEL AND SERIAL NUMBER
D1 IS THIS A VIRTUAL CP SYSTEM ?
YES
NO
MCVIRT
D2 TURN ON WAIT BIT IN MACHINE CHECK NEW PSW
02 A3
E1 SETUP EXERCISE COUNT TO 16
ISSUEINS
F1 ISSUE STIDC INSTRUCTION
G1 TEST CONDITION CODE FOR ZERO.
NO
G3
YES
H1 GET CHANNEL TYPE AND MODEL
J1 IS THIS A 2860 SELECTOR.
NO
A3
YES
K1 SET INDICATOR FOR 2860 SELECTOR
A3
A3
TEST2880
A3 IS THIS A 2880 SELECTOR. ?
NO
YES
B3 SET INDICATOR FOR 2880 SUPPORT
TEST2881
C3 IS THIS A 2880 BLOCK MULTIPLEX. ?
NO
YES
D3 SET INDICATOR FOR 2880 SUPPORT
TEST2870
E3 IS THIS A 2870 BYTE MULTIPLEX ?
NO
YES
F3 SET INDICATOR FOR 2870 SUPPORT
G3
CHANGID
G3 IS EXERCISE COUNT ZERO ?
NO
YES
A4 GET THE POINTER TO THE CONFIGURATION TABLE
B4 SAVE ALL THE CHANNEL TYPES IN THE CONFIGURATION TABLE
C4 SETUP ADDRESS FOR MACHINE CHECK COMMUNICATION AREA
D4 GET THE LENGTH OF THE DAMAGE ASSESSMENT AREA
E4 CALCULATE THE SPACE FOR THE MACHINE CHECK RECORD
F4 FREE GO GET THE SPACE FOR THE RECORD
G4 GET THE ADDRESS OF THE EXTENDED LOGOUT AREA
H4 CHANGE ADDRESS OF THE EXTENDED LOGOUT IN CONTROL REG 15
J4 SAVE THE POINTER OF THE MACHINE CHECK RECORD IN MCH
K4 GET THE SIZE OF THE CPEXBLOK
A5 FREE GET THE SPACE FOR THE CPEXBLOK
B5 SAVE THE CPEXBLOK PTR. IN MCH
C5 GET THE MODEL NUMBER OF THE MACHINE
MCH003
D5 SAVE THE MODEL NUMBER ID. IN THE MCH COMMUNICATION AREA
MCH004
E5 BRANCH TO MODEL SUPPORT.
NSUP---->07A2
M135---->07A3
M145---->07A4
M158---->07A5
M168---->08A2

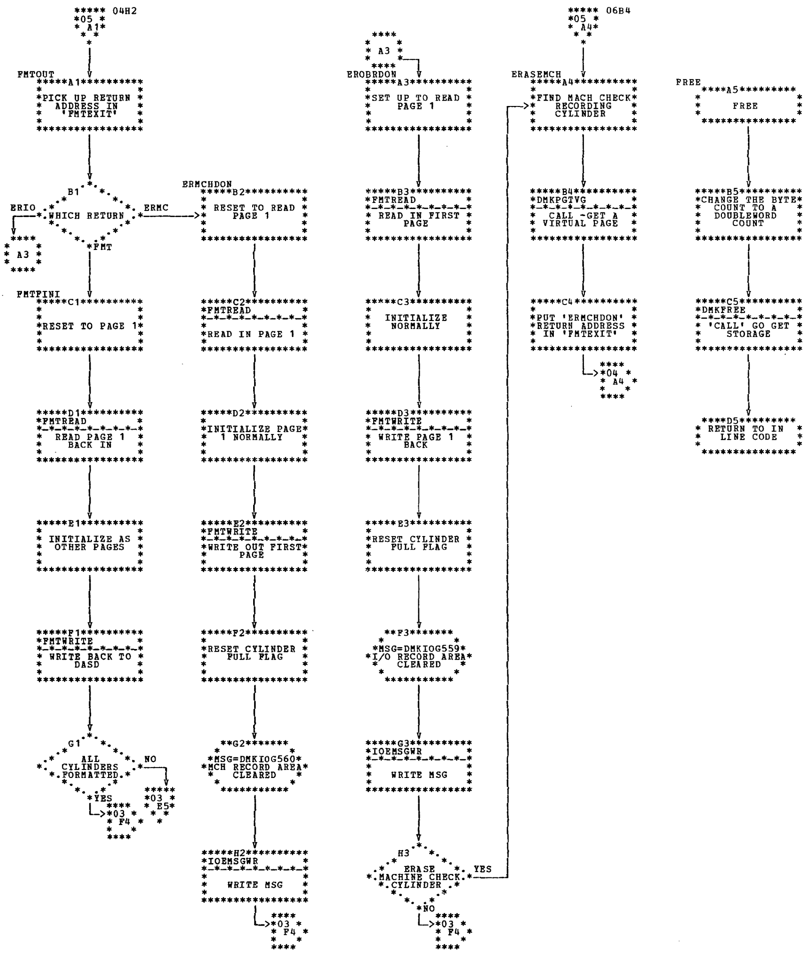

02 A3
01D2
07B2
07K2
PASTDAVE
A3 DETERMINE SYSTEM ERROR RECORDING CYLINDER AND SYSRES ADDRESS
B3 DMKSCNRU CALL -GET RDEVBOK POINTER
C3 CALCULATE MAX PAGES PER CYL AND 90% COUNT
FINDIOR
D3 SET RETURN ADDRESS OF 'SETOBRS' IN FINDRET
FINDREC
E3 DMKPGTVG CALL-GET VIRTUAL PAGE ADDRESS
02 F3
03E5
FIND2
F3 FMTREAD READ A PAGE RECORD
G3 CORRECT CCPD FOR PAGE.
NO
04 A2
YES
H3 SYSTEM CRASH DURING LAST FORMAT
YES
04 A2
NO
J3 VALID FLAGS SET IN PAGE
NO
04 A2
YES
02 K3
03D2
FINDLOOP
K3 IS PAGE FREE
NO
03 A2
YES
03 A1

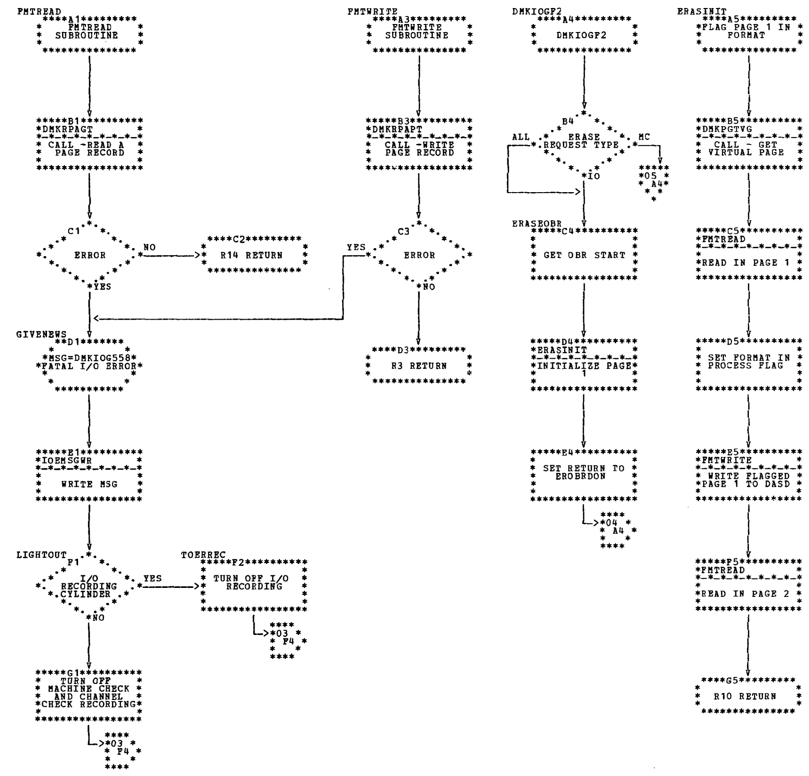
| DMKIOG -- Error Recording, Initialization and Clearing (Parts 1 and 2 of 8)

| DMKIOG -- Error Recording, Initialization and Clearing (Parts 3 and 4 of 8)


03 A1 02K3
FINDEXIT
A1 SET PAGE IN USE FLAG
03 A2 02K3
A2 IN USE BUT NOT FULL
YES
NO
B1 FMTWRITE BAL R3 - WRITE PAGE BACK
B2 BUMP TO NEXT PAGE
C1 LOAD RETURN ADDRESS FROM 'FINDRET'
C2 OUT OF PAGES
YES
NO
CYLFULL
C3 WHICH CYLINDER
I/O
MCH
CECYLFUL
C4 MSG=DMKIOG554 CYLINDER FULL
D1 OBR OR MCH CYL
MCH
04 A1
OBR
D2 FMTREAD BAL R14 - READ NEXT PAGE
02 K3
D3 MSG=DMKIOG555 CYLINDER FULL
D4 IOEMSGWR WRITE MESSAGE TO OPERATOR
SETOBRS
E1 SET PROPER STARTING DISK ADDRESS FOR I/O ERRORS
E5
E3 IOEMSGWR WRITE MESSAGE TO OPERATOR
E4 TURN OFF I/O ERROR RECORDING
03 E5 05G1
FINDMCR
E5 PUT 'SETMCHS' ADDRESS IN 'FINDRET'
A
E5
02 F3
03 F4 04A1 05G1 05H2 05H3 FNOTE
IOEXIT
F3 TURN OFF MACHINE CHECK AND CHANNEL CHECK RECORDING
F4 UPDATE IN-CORE POINTERS
G4 DMKRPAGT CALL -CLEAR PAGE REFERENCE
H4 DMKPGTVR CALL- RELEASE VIRTUAL PAGE
J4 RETURN TO CALLER
TO:F4 06F2 06G1

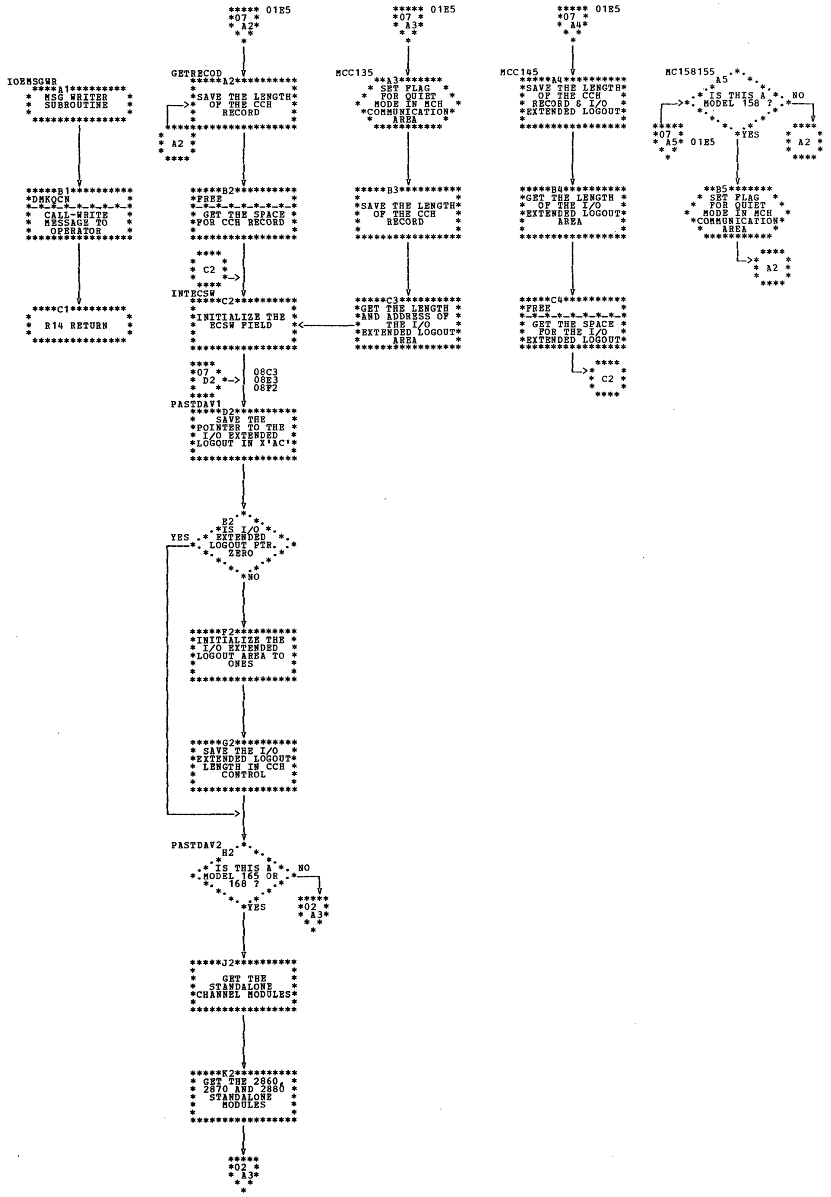

04 A1 03D1
SETMCHS
A1 SET PROPER DISK ADDRESS FOR MCH ERROR REC
03 F4
04 A2 02G3 02H3 02J3
FORMAT
A2 SET FLAGS IN PAGE 1 TO INDICATE FORMAT IN PROCESS
B2 WHICH CYLINDER
MCH
IO
FORMATMC
B3 MSG=DMKIOG557 FORMAT MC CYLINDER
C2 MSG=DMKIOG556 FORMAT I/O CYLINDER
C3 IOEMSGWR WRITE OPERATOR MESSAGE
D2 IOEMSGWR WRITE OPERATOR MESSAGE
D3 SET FMTEXIT TO ADDRESS OF FMTFINI
E2 SET FMTEXIT TO ADDRESS OF FMTFINI
FMTWRTPG
F2 FMTWRITE WRITE OUT PAGE 1
FMTPGINC
G2 UPDATE TO NEXT SEQUENTIAL PAGE
H2 END OF CYLINDER
YES
05 A1
NO
J2 FMTREAD READ A PAGE RECORD
04 A4 05C4 06E4
FMTLOOP
A4 MOVE IDENT ADR AND INIT VALUES TO BUFFER
B4 FMTWRITE WRITE OUT INITIALIZED RECORD



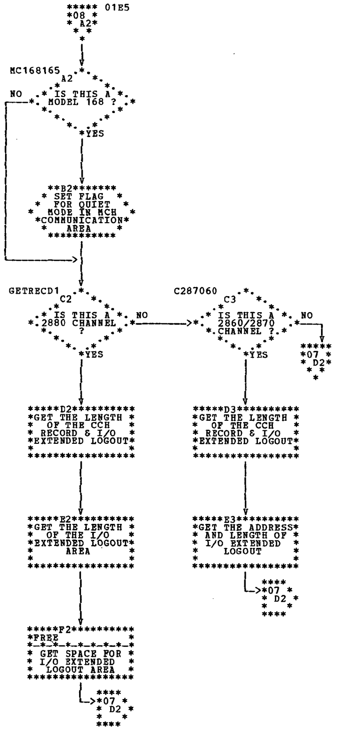


| DMKIOG -- Error Recording, Initialization and Clearing (Parts 5 and 6 of 8)

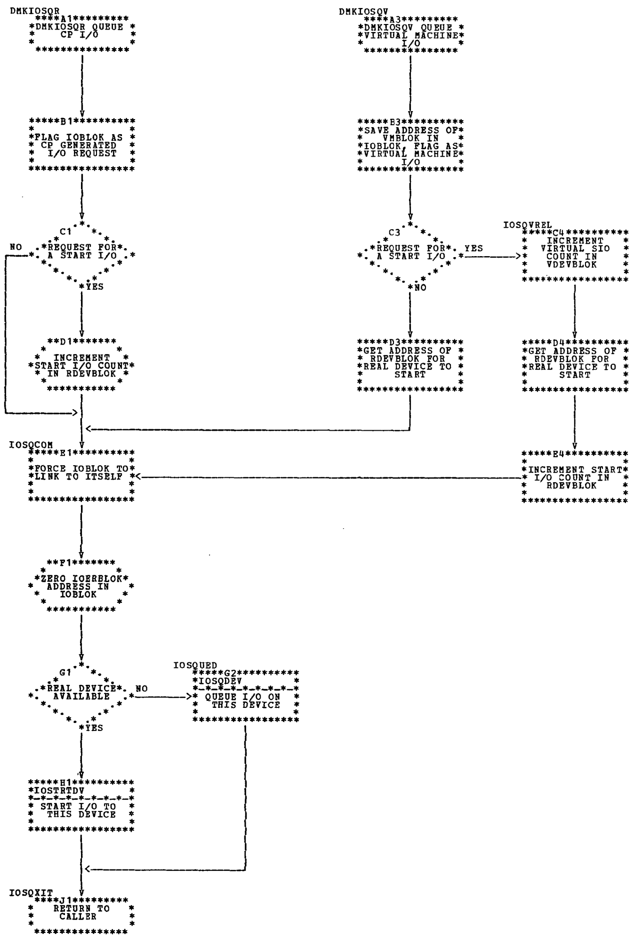
DMKIOG -- Error Recording, Initialization and Clearing (Parts 7 and 8 of 8)

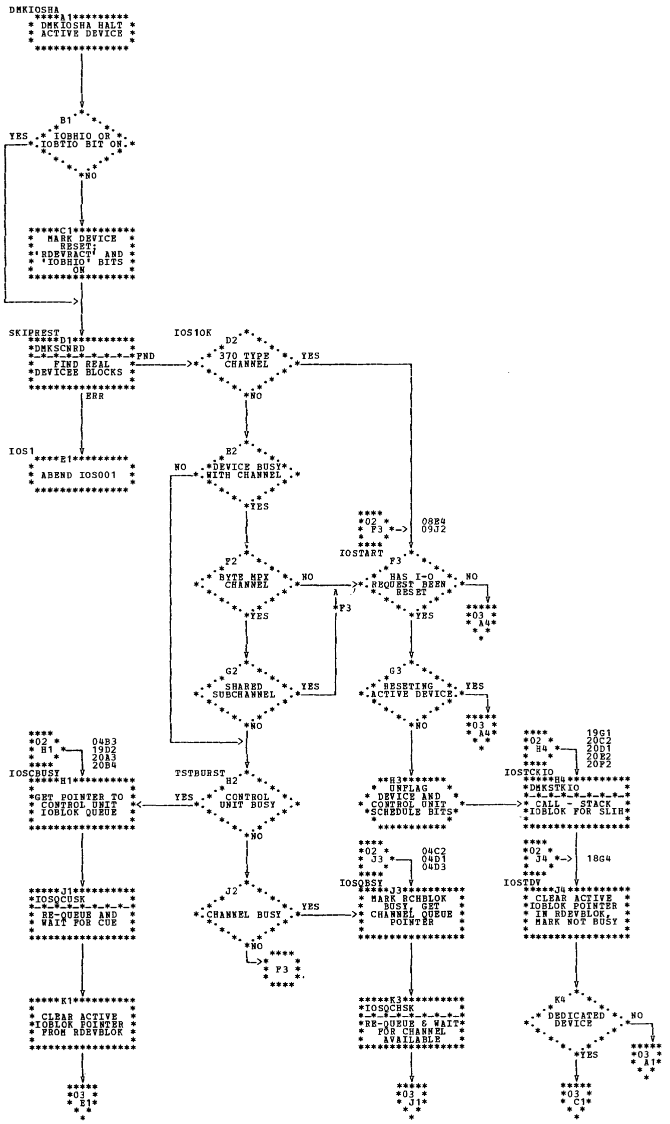


IOEMSGWR
A1 MSG WRITER SUBROUTINE
B1 DMKQCN CALL-WRITE MESSAGE TO OPERATOR
C1 R14 RETURN
07 A2 01E5
GETRECOD
A2 SAVE THE LENGTH OF THE CCH RECORD
A2
B2 FREE GET THE SPACE FOR CCH RECORD
C2
INTECSW
C2 INITIALIZE THE ECSW FIELD
07 D2 08C3 08E3 08F2
PASTDAV1
D2 SAVE THE POINTER TO THE I/O EXTENDED LOGOUT IN X'AC'
E2 IS I/O EXTENDED LOGOUT PTR. ZERO
YES
NO
F2 INITIALIZE THE I/O EXTENDED LOGOUT AREA TO ONES
G2 SAVE THE I/O EXTENDED LOGOUT LENGTH IN CCH CONTROL
PASTDAV2
H2 IS THIS A MODEL 165 OR 168 ?
NO
02 A3
YES
J2 GET THE STANDALONE CHANNEL MODULES
K2 GET THE 2860, 2870 AND 2880 STANDALONE MODULES
02 A3
07 A3 01E5
MCC135
A3 SET FLAG FOR QUIET MODE IN MCH COMMUNICATION AREA
B3 SAVE THE LENGTH OF THE CCH RECORD
C3 GET THE LENGTH AND ADDRESS OF THE I/O EXTENDED LOGOUT AREA
07 A4 01E5
MCC145
A4 SAVE THE LENGTH OF THE CCH RECORD & I/O EXTENDED LOGOUT
B4 GET THE LENGTH OF THE I/O EXTENDED LOGOUT AREA
C4 FREE GET THE SPACE FOR THE I/O EXTENDED LOGOUT
C2
MC158155
A5 IS THIS A MODEL 158 ?
NO
07 A5 01E5
A2
YES
B5 SET FLAG FOR QUIET MODE IN MCH COMMUNICATION AREA
A2

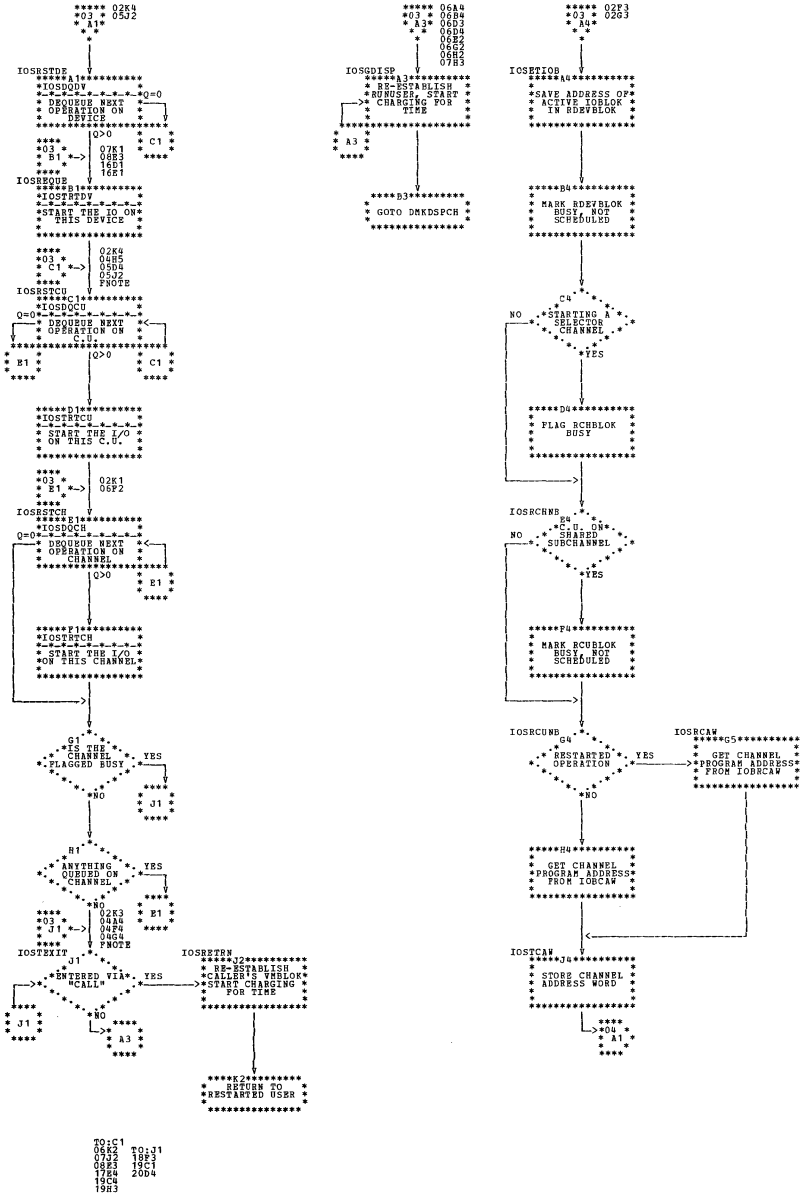


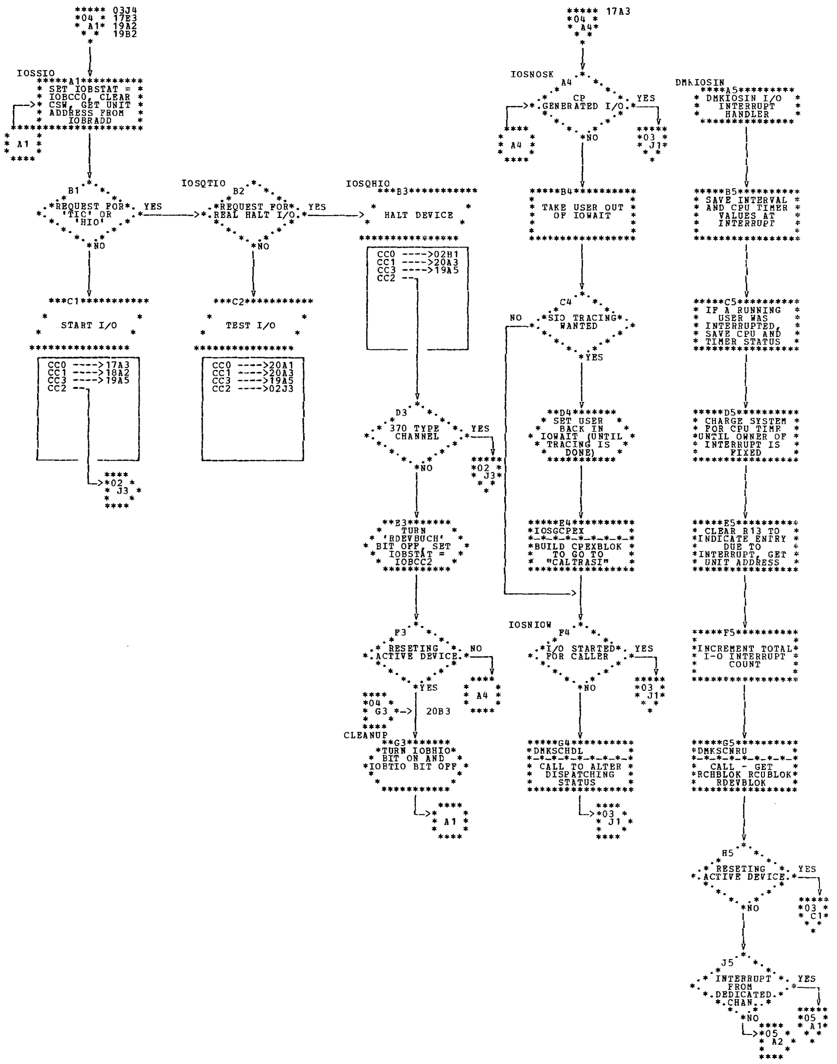
08 A2 01E5
MC168165
A2 IS THIS A MODEL 168 ?
NO
YES
B2 SET FLAG FOR QUIET MODE IN MCH COMMUNICATION AREA
GETRECD1
C2 IS THIS A 2880 CHANNEL ?
NO
YES
C287060
C3 IS THIS A 2860/2870 CHANNEL ?
NO
07 D2
YES
D2 GET THE LENGTH OF THE CCH RECORD & I/O EXTENDED LOGOUT
D3 GET THE LENGTH OF THE CCH RECORD & I/O EXTENDED LOGOUT
E2 GET THE LENGTH OF THE I/O EXTENDED LOGOUT AREA
E3 GET THE ADDRESS AND LENGTH OF I/O EXTENDED LOGOUT
07 D2
F2 FREE GET SPACE FOR I/O EXTENDED LOGOUT AREA
07 D2


DMKIOSQR
A1
DMKIOSQR QUEUE CP I/O
B1
FLAG IOBLOK AS CP GENERATED I/O REQUEST
C1
REQUEST FOR A START I/O
NO
YES
D1
INCREMENT START I/O COUNT IN RDEVBLOK
DMKIOSQV
A3
DMKIOSQV QUEUE VIRTUAL MACHINE I/O
B3
SAVE ADDRESS OF VMBLOK IN IOBLOK, FLAG AS VIRTUAL MACHINE I/O
C3
REQUEST FOR A START I/O
YES
NO
IOSQVREL
C4
INCREMENT VIRTUAL SIO COUNT IN VDEVBLOK
D3
GET ADDRESS OF RDEVBLOK FOR REAL DEVICE TO START
D4
GET ADDRESS OF RDEVBLOK FOR REAL DEVICE TO START
IOSQCOM
E1
FORCE IOBLOK TO LINK TO ITSELF
E4
INCREMENT START I/O COUNT IN RDEVBLOK
F1
ZERO IOERBLOK ADDRESS IN IOBLOK
G1
REAL DEVICE AVAILABLE
NO
YES
IOSQUED
G2
IOSQDEV
QUEUE I/O ON THIS DEVICE
H1
IOSTRTDV
START I/O TO THIS DEVICE
IOSQXIT
J1
RETURN TO CALLER



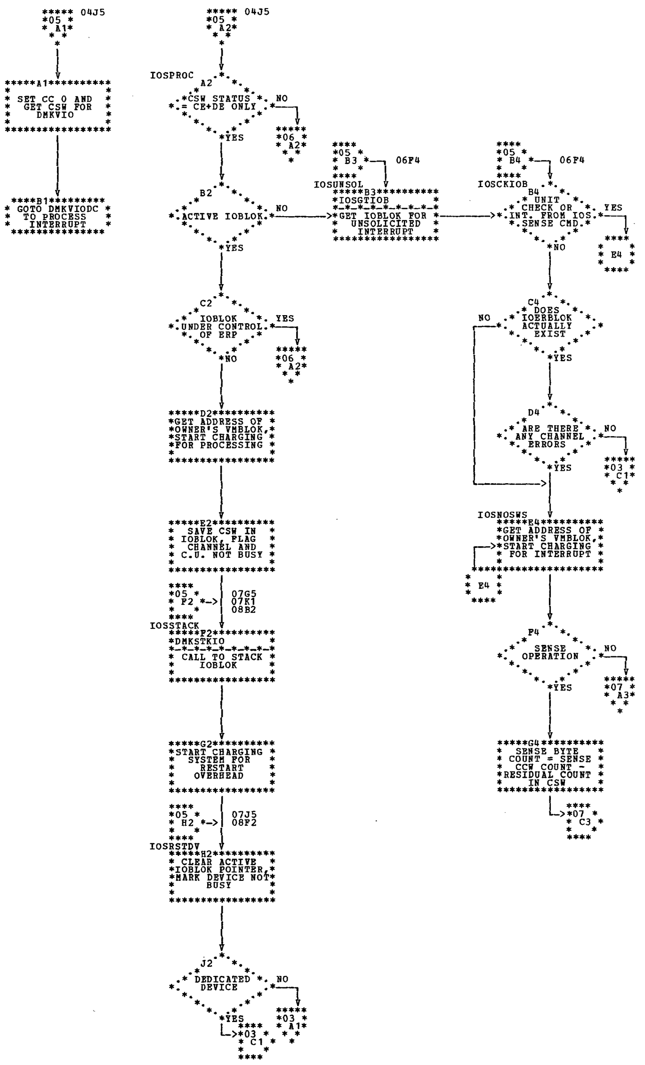
| DMKIOS -- I/O Supervisor (Parts 1 and 2 of 20)

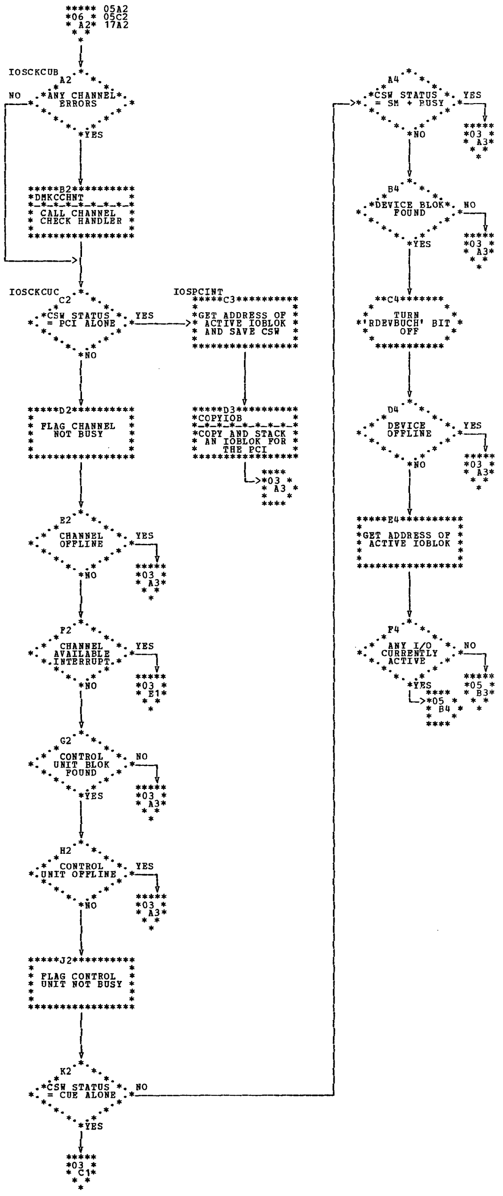

03 A1 02K4 05J2
IOSRSTDE
A1 IOSDQDV DEQUEUE NEXT OPERATION ON DEVICE
Q=0
Q>0
C1
03 B1 07K1 08E3 16D1 16E1
IOSREQUE
B1 IOSTRTDV START THE IO ON THIS DEVICE
03 C1 02K4 04H5 05D4 05J2 FNOTE
IOSRSTCU
C1 IOSDQCU DEQUEUE NEXT OPERATION ON C.U.
Q=0
Q>0
E1
C1
D1 IOSTRTCU START THE I/O ON THIS C.U.
03 E1 02K1 06F2
IOSRSTCH
E1 IOSDQCH DEQUEUE NEXT OPERATION ON CHANNEL
Q=0
Q>0
E1
F1 IOSTRTCH START THE I/O ON THIS CHANNEL
G1 IS THE CHANNEL FLAGGED BUSY
YES
NO
J1
H1 ANYTHING QUEUED ON CHANNEL
YES
NO
E1
03 J1 02K3 04A4 04F4 04G4 FNOTE
IOSTEXIT
J1 ENTERED VIA "CALL"
YES
NO
J1
A3
IOSRETRN
J2 RE-ESTABLISH CALLER'S VMBLOK START CHARGING FOR TIME
K2 RETURN TO RESTARTED USER
TO:C1 06K2 07J2 08E3 17E4 19C4 19H3
TO:J1 18F3 19C1 20D4
03 A3 06A4 06B4 06D3 06D4 06E2 06G2 06H2 07H3
IOSGDISP
A3 RE-ESTABLISH RUNUSER, START CHARGING FOR TIME
A3
B3 GOTO DMKDSPCH
03 A4 02F3 02G3
IOSETIOB
A4 SAVE ADDRESS OF ACTIVE IOBLOK IN RDEVBLOK
B4 MARK RDEVBLOK BUSY, NOT SCHEDULED
C4 STARTING A SELECTOR CHANNEL
NO
YES
D4 FLAG RCHBLOK BUSY
IOSRCHNB
E4 C.U. ON SHARED SUBCHANNEL
NO
YES
F4 MARK RCUBLOK BUSY, NOT SCHEDULED
IOSRCUNB
G4 RESTARTED OPERATION
YES
NO
IOSRCAW
G5 GET CHANNEL PROGRAM ADDRESS FROM IOBRCAW
H4 GET CHANNEL PROGRAM ADDRESS FROM IOBCAW
IOSTCAW
J4 STORE CHANNEL ADDRESS WORD
04 A1

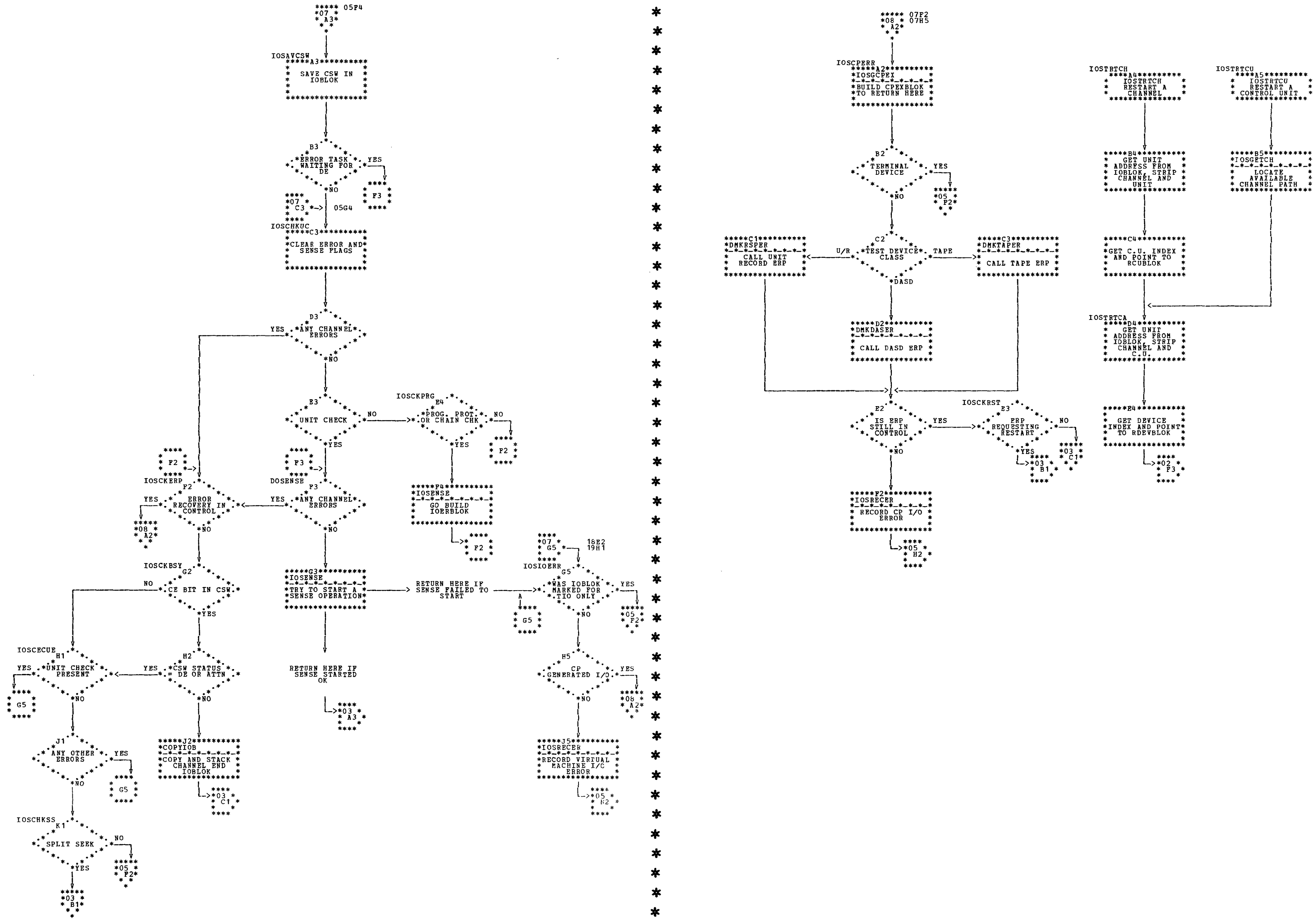

04 A1 03J4 17E3 19A2 19B2
IOSSIO
A1 SET IOBSTAT = IOBCC0, CLEAR CSW, GET UNIT ADDRESS FROM IOBRADD
A1
B1 REQUEST FOR 'TIC' OR 'HIO'
YES
NO
C1 START I/O
CC0 ---->17A3
CC1 ---->18A2
CC3 ---->19A5
CC2 --
02 J3
IOSQTIO
B2 REQUEST FOR REAL HALT I/O
YES
NO
C2 TEST I/O
CC0 ---->20A1
CC1 ---->20A3
CC3 ---->19A5
CC2 ---->02J3
IOSQHIO
B3 HALT DEVICE
CC0 ---->02H1
CC1 ---->20A3
CC3 ---->19A5
CC2 --
D3 370 TYPE CHANNEL
YES
NO
02 J3
E3 TURN 'RDEVBUCH' BIT OFF, SET IOBSTAT = IOBCC2
F3 RESETING ACTIVE DEVICE
NO
YES
A4
04 G3 20B3
CLEANUP
G3 TURN IOBHIO BIT ON AND IOBTIO BIT OFF
A1
04 A4 17A3
IOSNOSK
A4 CP GENERATED I/O
YES
NO
A4
03 J1
B4 TAKE USER OUT OF IOWAIT
C4 SIO TRACING WANTED
NO
YES
D4 SET USER BACK IN IOWAIT (UNTIL TRACING IS DONE)
E4 IOSGCPEX BUILD CPEXBLOK TO GO TO "CALTRASI"
IOSNIOW
F4 I/O STARTED FOR CALLER
YES
NO
03 J1
G4 DMKSCHDL CALL TO ALTER DISPATCHING STATUS
03 J1
DMKIOSIN
A5 DMKIOSIN I/O INTERRUPT HANDLER
B5 SAVE INTERVAL AND CPU TIMER VALUES AT INTERRUPT
C5 IF A RUNNING USER WAS INTERRUPTED, SAVE CPU AND TIMER STATUS
D5 CHARGE SYSTEM FOR CPU TIME UNTIL OWNER OF INTERRUPT IS FIXED
E5 CLEAR R13 TO INDICATE ENTRY DUE TO INTERRUPT, GET UNIT ADDRESS
F5 INCREMENT TOTAL I-O INTERRUPT COUNT
G5 DMKSCNRU CALL - GET RCHBLOK RCUBLOK RDEVBLOK
H5 RESETING ACTIVE DEVICE
YES
NO
03 C1
J5 INTERRUPT FROM DEDICATED CHAN.
YES
NO
05 A1
05 A2

| DMKIOS -- I/O Supervisor (Parts 3 and 4 of 20)

| DMKIOS -- I/O Supervisor (Parts 5 and 6 of 20)

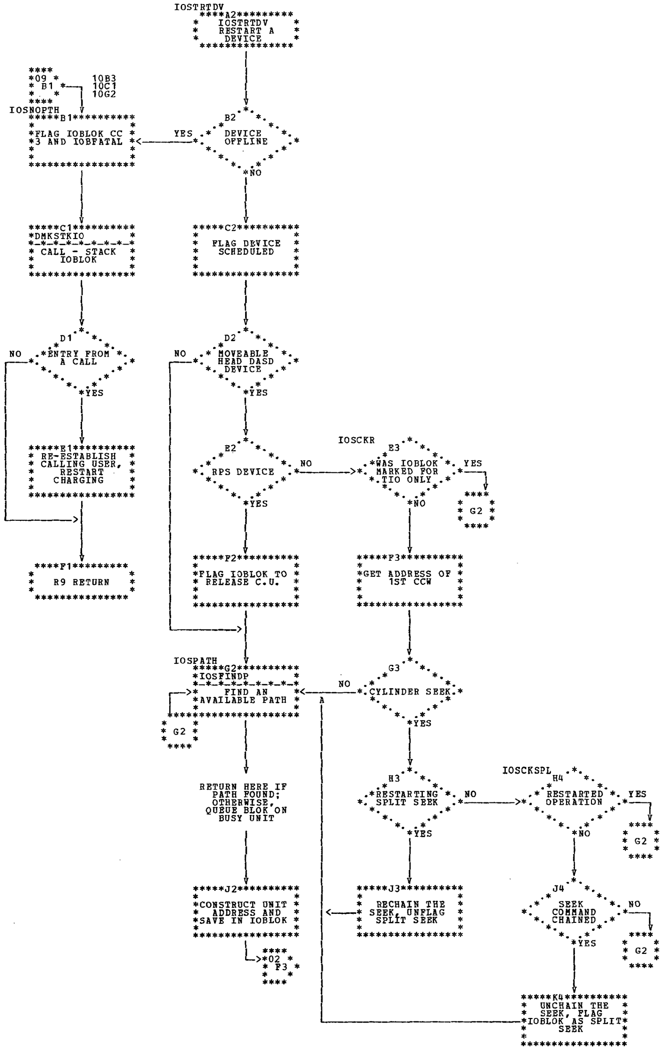

05 A1 04J5
A1
SET CC 0 AND GET CSW FOR DMKVIO
B1
GOTO DMKVIODC TO PROCESS INTERRUPT
05 A2 04J5
IOSPROC
A2
CSW STATUS = CE+DE ONLY
NO
06 A2
YES
B2
ACTIVE IOBLOK
NO
YES
05 B3 06F4
IOSUNSOL
B3
IOSGTIOB
GET IOBLOK FOR UNSOLICITED INTERRUPT
05 B4 06F4
IOSCKIOB
B4
UNIT CHECK OR INT. FROM IOS. SENSE CMD.
YES
E4
NO
C2
IOBLOK UNDER CONTROL OF ERP
YES
06 A2
NO
C4
DOES IOERBLOK ACTUALLY EXIST
NO
YES
D2
GET ADDRESS OF OWNER'S VMBLOK, START CHARGING FOR PROCESSING
D4
ARE THERE ANY CHANNEL ERRORS
NO
03 C1
YES
E2
SAVE CSW IN IOBLOK, FLAG CHANNEL AND C.U. NOT BUSY
IOSNOSWS
E4
GET ADDRESS OF OWNER'S VMBLOK, START CHARGING FOR INTERRUPT
E4
05 F2 07G5 07K1 08B2
IOSSTACK
F2
DMKSTKIO
CALL TO STACK IOBLOK
F4
SENSE OPERATION
NO
07 A3
YES
G2
START CHARGING SYSTEM FOR RESTART OVERHEAD
G4
SENSE BYTE COUNT = SENSE CCW COUNT - RESIDUAL COUNT IN CSW
07 C3
05 H2 07J5 06F2
IOSRSTDV
H2
CLEAR ACTIVE IOBLOK POINTER, MARK DEVICE NOT BUSY
J2
DEDICATED DEVICE
NO
03 A1
YES
03 C1

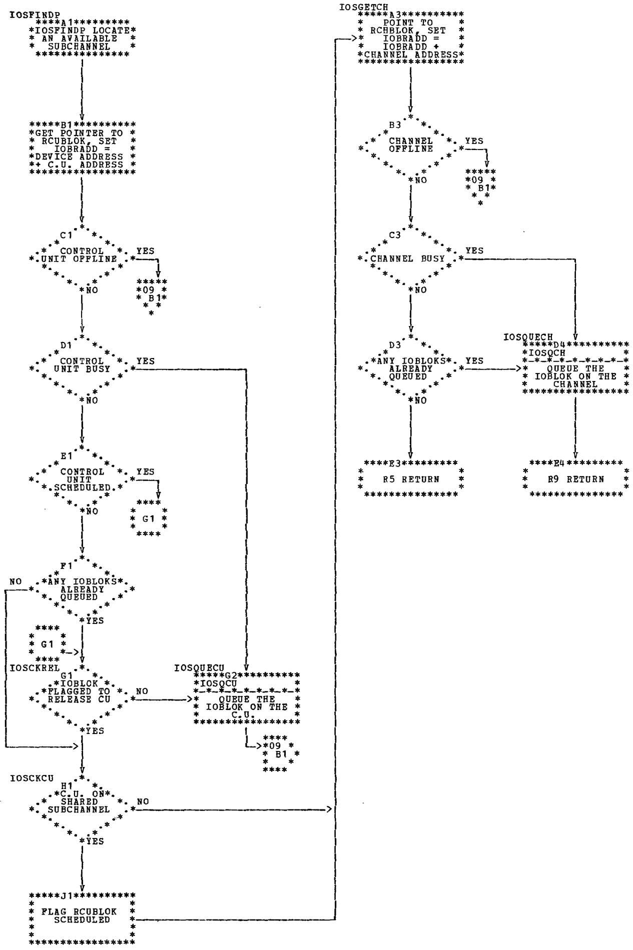


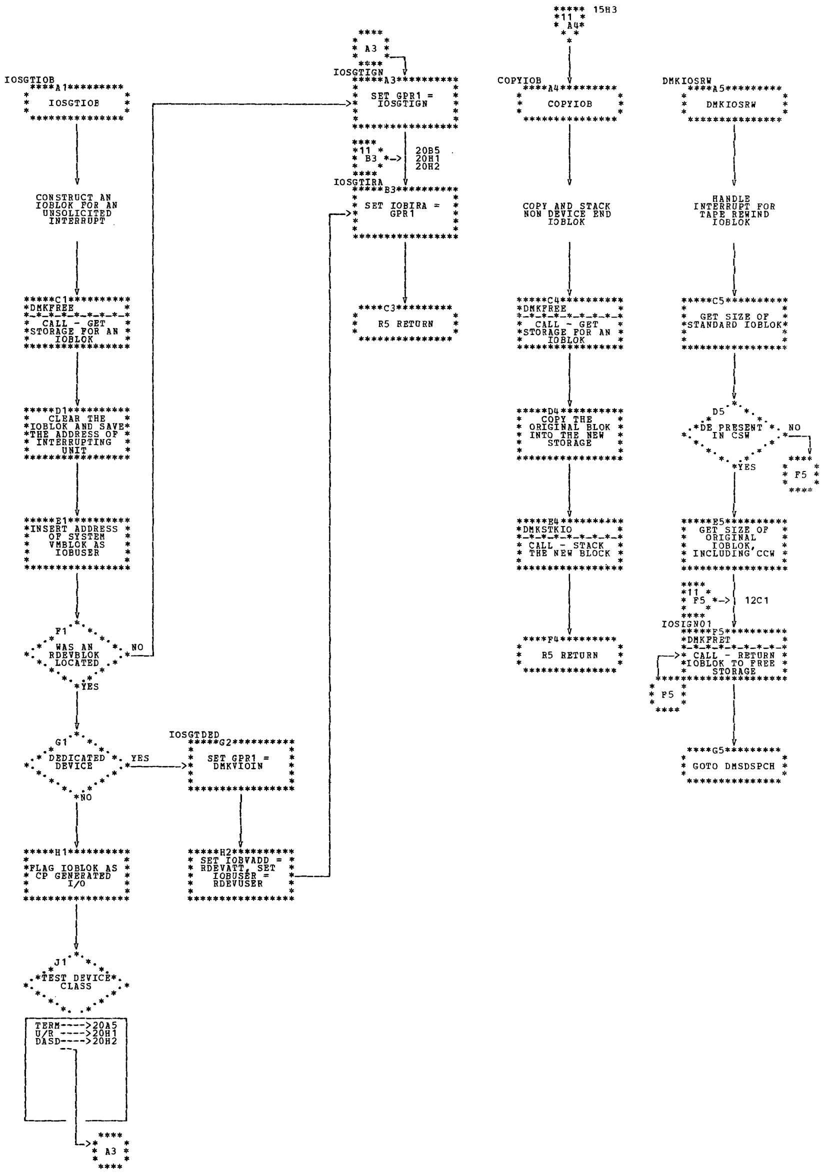

07 A3 05F4
IOSAVCSW A3
SAVE CSW IN IOBLOK
B3 ERROR TASK WAITING FOR DE
YES
F3
NO
07 C3 05G4
IOSCHKUC C3
CLEAR ERROR AND SENSE FLAGS
D3 ANY CHANNEL ERRORS
YES
NO
E3 UNIT CHECK
NO
YES
IOSCKPRG E4 PROG. PROT. OR CHAIN CHK
NO
YES
F2
F2
F3
IOSCKERP F2 ERROR RECOVERY IN CONTROL
YES
08 A2
NO
DOSENSE F3 ANY CHANNEL ERRORS
YES
NO
F4 IOSENSE
GO BUILD IOERBLOK
F2
IOSCKBSY G2 CE BIT IN CSW
NO
YES
G3 IOSENSE
TRY TO START A SENSE OPERATION
RETURN HERE IF SENSE FAILED TO START
A
G5
07 G5 18E2 19H1
IOSIOERR G5 WAS IOBLOK MARKED FOR TIO ONLY
YES
05 F2
NO
IOSCECUE H1 UNIT CHECK PRESENT
YES
G5
NO
H2 CSW STATUS = DE OR ATTN
YES
NO
RETURN HERE IF SENSE STARTED OK
03 A3
H5 CP GENERATED I/O
YES
08 A2
NO
J1 ANY OTHER ERRORS
YES
G5
NO
J2 COPYIOB
COPY AND STACK CHANNEL END IOBLOK
03 C1
J5 IOSRECER
RECORD VIRTUAL MACHINE I/C ERROR
05 H2
IOSCHKSS K1 SPLIT SEEK
NO
05 F2
YES
03 B1
08 A2 07F2 07H5
IOSCPERR A2 IOSGCPEX
BUILD CPEXBLOK TO RETURN HERE
B2 TERMINAL DEVICE
YES
05 F2
NO
C1 DMKRSPER
CALL UNIT RECORD ERP
U/R
C2 TEST DEVICE CLASS
TAPE
C3 DMKTAPER
CALL TAPE ERP
DASD
D2 DMKDASER
CALL DASD ERP
E2 IS ERP STILL IN CONTROL
YES
IOSCKRST E3 ERP REQUESTING RESTART
NO
03 C1
YES
03 B1
NO
F2 IOSRECER
RECORD CP I/O ERROR
05 H2
IOSTRTCH A4 IOSTRTCH
RESTART A CHANNEL
B4 GET UNIT ADDRESS FROM IOBLOK, STRIP CHANNEL AND UNIT
C4 GET C.U. INDEX AND POINT TO RCUBLOK
IOSTRTCA D4 GET UNIT ADDRESS FROM IOBLOK, STRIP CHANNEL AND C.U.
E4 GET DEVICE INDEX AND POINT TO RDEVBLOK
02 F3
IOSTRTCU A5 IOSTRTCU
RESTART A CONTROL UNIT
B5 IOSGETCH
LOCATE AVAILABLE CHANNEL PATH

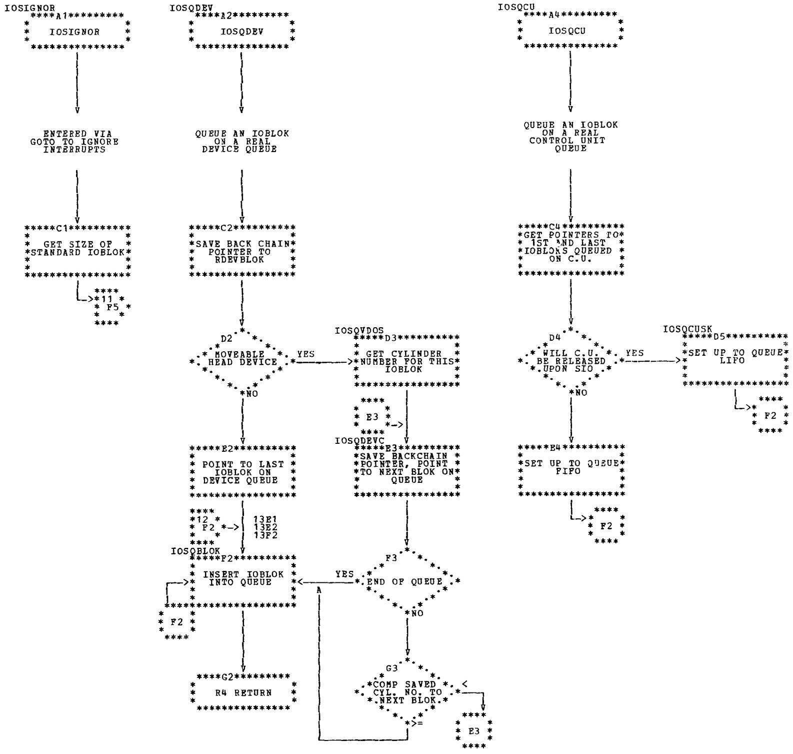
DMKIOS -- I/O Supervisor (Parts 7 and 8 of 20)

| DMKIOS -- I/O Supervisor (Parts 9 and 10 of 20)



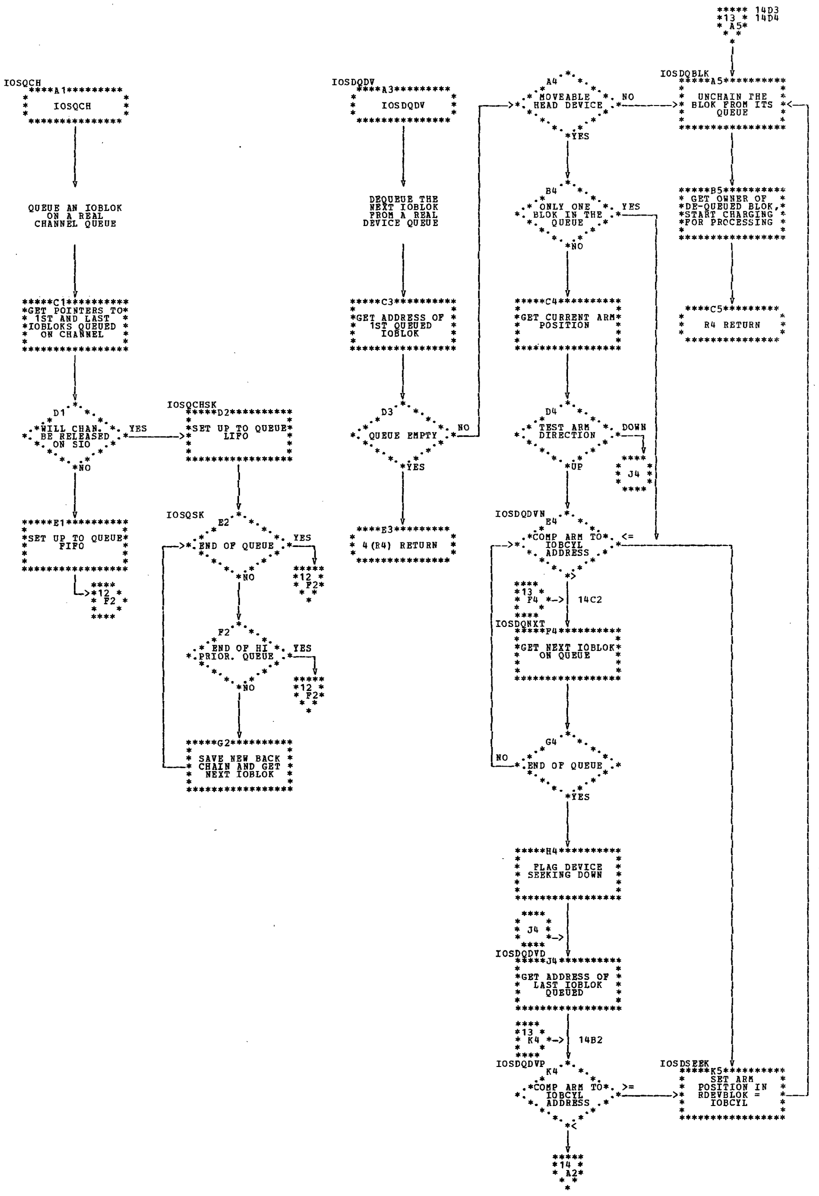

IOSFINDP
A1
IOSFINDP LOCATE AN AVAILABLE SUBCHANNEL
B1
GET POINTER TO RCUBLOK, SET IOBRADD = DEVICE ADDRESS + C.U. ADDRESS
C1
CONTROL UNIT OFFLINE
YES
NO
09 B1
D1
CONTROL UNIT BUSY
YES
NO
E1
CONTROL UNIT SCHEDULED.
YES
NO
G1
F1
ANY IOBLOKS ALREADY QUEUED
NO
YES
G1
IOSCKREL
G1
IOBLOK FLAGGED TO RELEASE CU
NO
YES
IOSQUECU
G2
IOSQCU
QUEUE THE IOBLOK ON THE C.U.
09 B1
IOSCKCU
H1
C.U. ON SHARED SUBCHANNEL
NO
YES
J1
FLAG RCUBLOK SCHEDULED
IOSGETCH
A3
POINT TO RCHBLOK, SET IOBRADD = IOBRADD + CHANNEL ADDRESS
B3
CHANNEL OFFLINE
YES
NO
09 B1
C3
CHANNEL BUSY
YES
NO
D3
ANY IOBLOKS ALREADY QUEUED
YES
NO
IOSQUECH
D4
IOSQCH
QUEUE THE IOBLOK ON THE CHANNEL
E3
R5 RETURN
E4
R9 RETURN

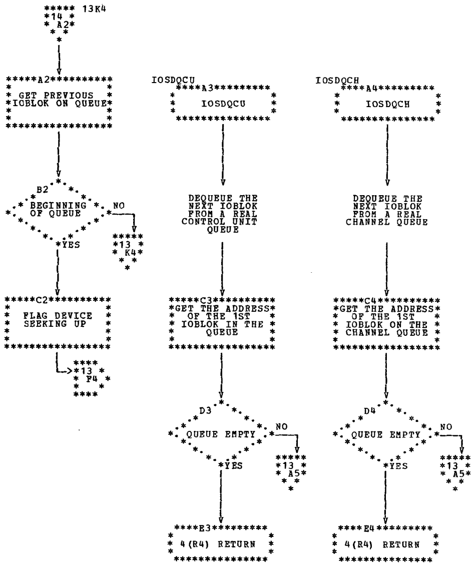

IOSGTIOB
A1
IOSGTIOB
CONSTRUCT AN IOBLOK FOR AN UNSOLICITED INTERRUPT
C1
DMKFREE
CALL - GET STORAGE FOR AN IOBLOK
D1
CLEAR THE IOBLOK AND SAVE THE ADDRESS OF INTERRUPTING UNIT
E1
INSERT ADDRESS OF SYSTEM VMBLOK AS IOBUSER
F1
WAS AN RDEVBLOK LOCATED
NO
YES
G1
DEDICATED DEVICE
YES
NO
IOSGTDED
G2
SET GPR1 = DMKVIOIN
H1
FLAG IOBLOK AS CP GENERATED I/O
H2
SET IOBVADD = RDEVATT, SET IOBUSER = RDEVUSER
J1
TEST DEVICE CLASS
TERM---->20A5
U/R ---->20H1
DASD---->20H2
A3
A3
IOSGTIGN
A3
SET GPR1 = IOSGTIGN
11
B3
20B5
20H1
20H2
IOSGTIRA
B3
SET IOBIRA = GPR1
C3
R5 RETURN
15H3
11
A4
COPYIOB
A4
COPYIOB
COPY AND STACK NON DEVICE END IOBLOK
C4
DMKFREE
CALL - GET STORAGE FOR AN IOBLOK
D4
COPY THE ORIGINAL BLOK INTO THE NEW STORAGE
E4
DMKSTKIO
CALL - STACK THE NEW BLOCK
F4
R5 RETURN
DMKIOSRW
A5
DMKIOSRW
HANDLE INTERRUPT FOR TAPE REWIND IOBLOK
C5
GET SIZE OF STANDARD IOBLOK
D5
DE PRESENT IN CSW
NO
F5
YES
E5
GET SIZE OF ORIGINAL IOBLOK, INCLUDING CCW
11
F5
12C1
IOSIGN01
F5
DMKFRET
CALL - RETURN IOBLOK TO FREE STORAGE
F5
G5
GOTO DMSDSPCH

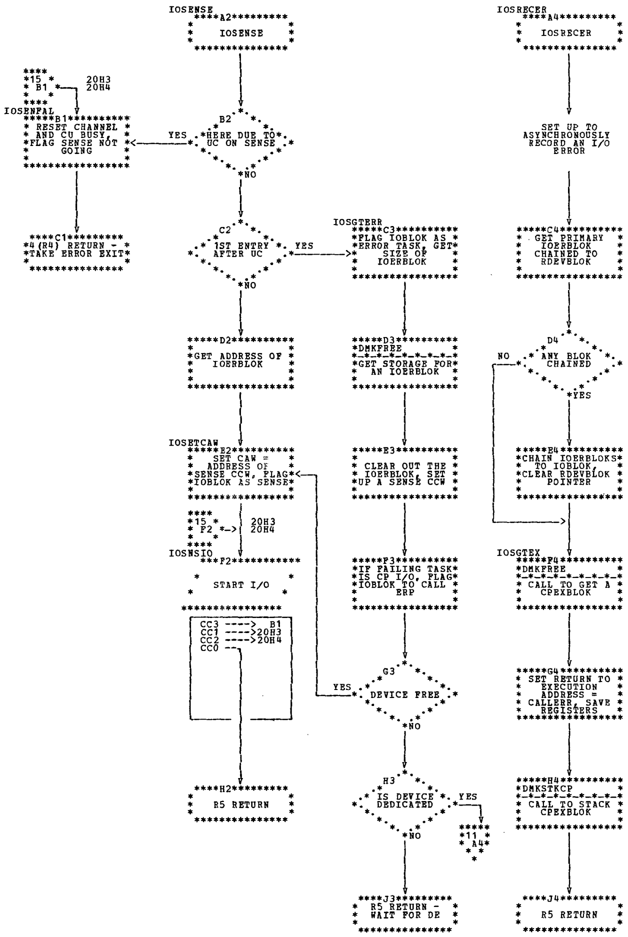

IOSIGNOR
A1
IOSIGNOR
ENTERED VIA GOTO TO IGNORE INTERRUPTS
C1
GET SIZE OF STANDARD IOBLOK
11
F5
IOSQDEV
A2
IOSQDEV
QUEUE AN IOBLOK ON A REAL DEVICE QUEUE
C2
SAVE BACK CHAIN POINTER TO RDEVBLOK
D2
MOVEABLE HEAD DEVICE
YES
NO
IOSQVDOS
D3
GET CYLINDER NUMBER FOR THIS IOBLOK
E3
IOSQDEVC
E3
SAVE BACKCHAIN POINTER, POINT TO NEXT BLOK ON QUEUE
E2
POINT TO LAST IOBLOK ON DEVICE QUEUE
12
F2
13E1
13E2
13F2
IOSQBLOK
F2
INSERT IOBLOK INTO QUEUE
F2
G2
R4 RETURN
F3
END OF QUEUE
YES
NO
A
G3
COMP SAVED CYL. NO. TO NEXT BLOK.
<
>=
E3
IOSQCU
A4
IOSQCU
QUEUE AN IOBLOK ON A REAL CONTROL UNIT QUEUE
C4
GET POINTERS TO 1ST AND LAST IOBLOKS QUEUED ON C.U.
D4
WILL C.U. BE RELEASED UPON SIO
YES
NO
IOSQCUSK
D5
SET UP TO QUEUE LIFO
F2
E4
SET UP TO QUEUE FIFO
F2

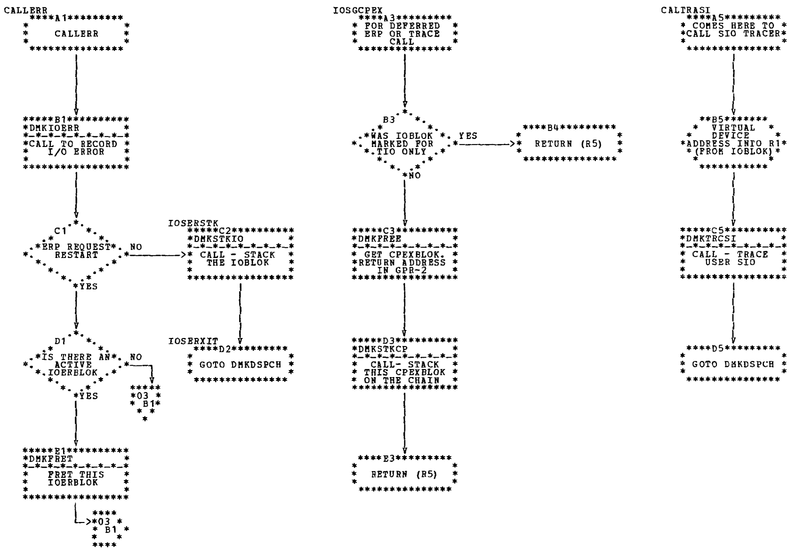
| DMKIOS -- I/O Supervisor (Parts 11 and 12 of 20)

| DMKIOS -- I/O Supervisor (Parts 13 and 14 of 20)


IOSQCH
A1
IOSQCH
QUEUE AN IOBLOK ON A REAL CHANNEL QUEUE
C1
GET POINTERS TO 1ST AND LAST IOBLOKS QUEUED ON CHANNEL
D1
WILL CHAN. BE RELEASED ON SIO
YES
NO
E1
SET UP TO QUEUE FIFO
12 F2
IOSQCHSK
D2
SET UP TO QUEUE LIFO
IOSQSK
E2
END OF QUEUE
YES
12 F2
NO
F2
END OF HI PRIOR. QUEUE
YES
12 F2
NO
G2
SAVE NEW BACK CHAIN AND GET NEXT IOBLOK
IOSDQDV
A3
IOSDQDV
DEQUEUE THE NEXT IOBLOK FROM A REAL DEVICE QUEUE
C3
GET ADDRESS OF 1ST QUEUED IOBLOK
D3
QUEUE EMPTY
NO
YES
E3
4(R4) RETURN
A4
MOVEABLE HEAD DEVICE
NO
YES
B4
ONLY ONE BLOK IN THE QUEUE
YES
NO
C4
GET CURRENT ARM POSITION
D4
TEST ARM DIRECTION
DOWN
J4
UP
IOSDQDVN
E4
COMP ARM TO IOBCYL ADDRESS
<=
>
13 F4
14C2
IOSDQNXT
F4
GET NEXT IOBLOK ON QUEUE
G4
END OF QUEUE
NO
YES
H4
FLAG DEVICE SEEKING DOWN
J4
IOSDQDVD
J4
GET ADDRESS OF LAST IOBLOK QUEUED
13 K4
14B2
IOSDQDVP
K4
COMP ARM TO IOBCYL ADDRESS
>=
<
14 A2
14D3
13 A2
14D4
IOSDQBLK
A5
UNCHAIN THE BLOK FROM ITS QUEUE
B5
GET OWNER OF DE-QUEUED BLOK, START CHARGING FOR PROCESSING
C5
R4 RETURN
IOSDSEEK
K5
SET ARM POSITION IN RDEVBLOK = IOBCYL

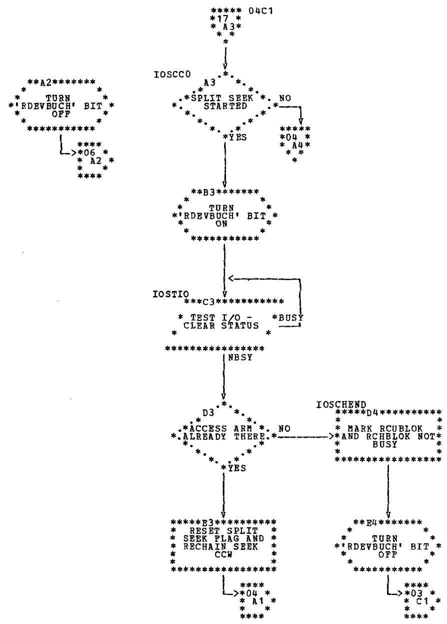

14 A2
13K4
A2
GET PREVIOUS IOBLOK ON QUEUE
B2
BEGINNING OF QUEUE
NO
13 K4
YES
C2
FLAG DEVICE SEEKING UP
13 F4
IOSDQCU
A3
IOSDQCU
DEQUEUE THE NEXT IOBLOK FROM A REAL CONTROL UNIT QUEUE
C3
GET THE ADDRESS OF THE 1ST IOBLOK IN THE QUEUE
D3
QUEUE EMPTY
NO
13 A5
YES
E3
4(R4) RETURN
IOSDQCH
A4
IOSDQCH
DEQUEUE THE NEXT IOBLOK FROM A REAL CHANNEL QUEUE
C4
GET THE ADDRESS OF THE 1ST IOBLOK ON THE CHANNEL QUEUE
D4
QUEUE EMPTY
NO
13 A5
YES
E4
4(R4) RETURN

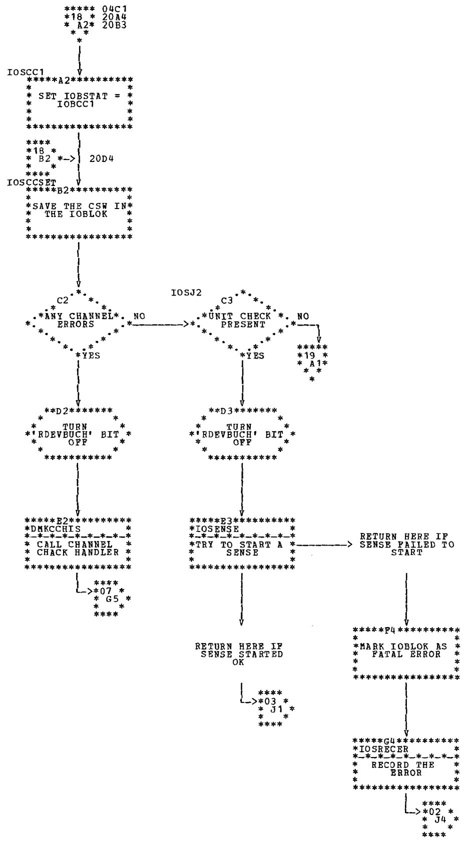

IOSENSE
A2 IOSENSE
15 B1 20H3 20H4
IOSENFAL
B1 RESET CHANNEL AND CU BUSY, FLAG SENSE NOT GOING
B2 HERE DUE TO UC ON SENSE
YES
NO
C1 4(R4) RETURN - TAKE ERROR EXIT
C2 1ST ENTRY AFTER UC
YES
NO
IOSGTERR
C3 FLAG IOBLOK AS ERROR TASK, GET SIZE OF IOERBLOK
D2 GET ADDRESS OF IOERBLOK
D3 DMKFREE GET STORAGE FOR AN IOERBLOK
IOSETCAW
E2 SET CAW = ADDRESS OF SENSE CCW, FLAG IOBLOK AS SENSE
E3 CLEAR OUT THE IOERBLOK, SET UP A SENSE CCW
15 F2 20H3 20H4
IOSNSIO
F2 START I/O
CC3 ----> B1
CC1 ---->20H3
CC2 ---->20H4
CC0
F3 IF FAILING TASK IS CP I/O, FLAG IOBLOK TO CALL ERP
G3 DEVICE FREE
YES
NO
H2 R5 RETURN
H3 IS DEVICE DEDICATED
YES
NO
11 A4
J3 R5 RETURN - WAIT FOR DE
IOSRECER
A4 IOSRECER
SET UP TO ASYNCHRONOUSLY RECORD AN I/O ERROR
C4 GET PRIMARY IOERBLOK CHAINED TO RDEVBLOK
D4 ANY BLOK CHAINED
NO
YES
E4 CHAIN IOERBLOKS TO IOBLOK, CLEAR RDEVBLOK POINTER
IOSGTEX
F4 DMKFREE CALL TO GET A CPEXBLOK
G4 SET RETURN TO EXECUTION ADDRESS = CALLERR, SAVE REGISTERS
H4 DMKSTKCP CALL TO STACK CPEXBLOK
J4 R5 RETURN

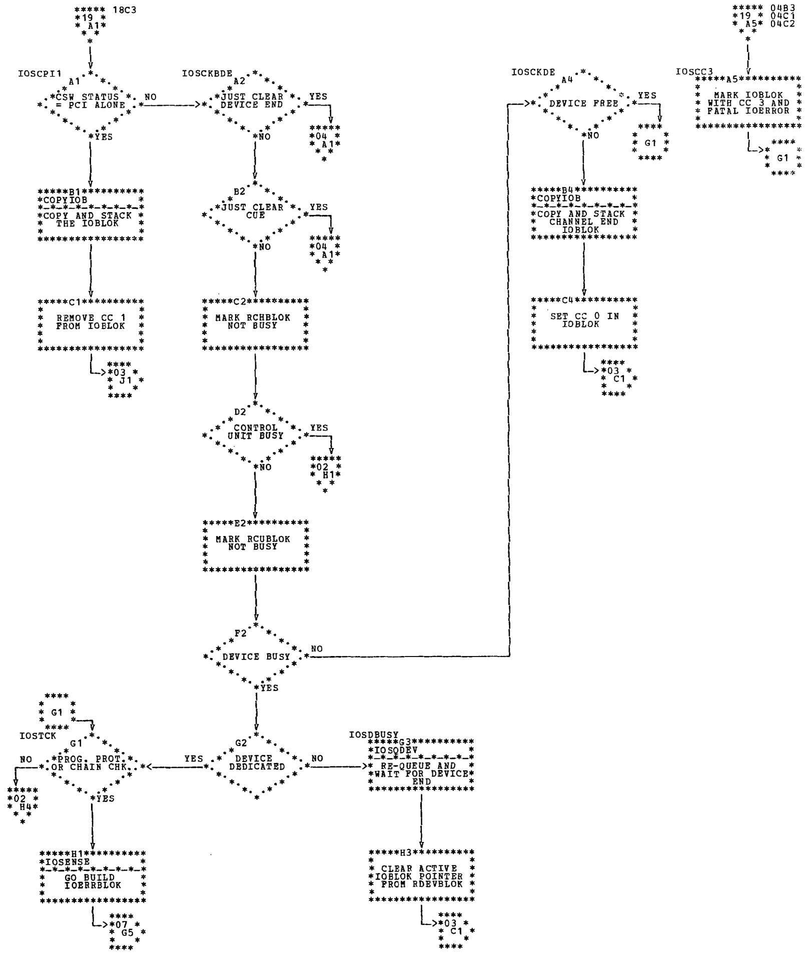

CALLERR
A1 CALLERR
B1 DMKIOERR CALL TO RECORD I/O ERROR
C1 ERP REQUEST RESTART
NO
YES
IOSERSTK
C2 DMKSTKIO CALL - STACK THE IOBLOK
D1 IS THERE AN ACTIVE IOERBLOK
NO
YES
03 B1
IOSERXIT
D2 GOTO DMKDSPCH
E1 DMKFRET FRET THIS IOERBLOK
03 B1
IOSGCPEX
A3 FOR DEFERRED ERP OR TRACE CALL
B3 WAS IOBLOK MARKED FOR TIO ONLY
YES
NO
B4 RETURN (R5)
C3 DMKFREE GET CPEXBLOK. RETURN ADDRESS IN GPR-2
D3 DMKSTKCP CALL- STACK THIS CPEXBLOK ON THE CHAIN
E3 RETURN (R5)
CALTRASI
A5 COMES HERE TO CALL SIO TRACER
B5 VIRTUAL DEVICE ADDRESS INTO R1 (FROM IOBLOK)
C5 DMKTRCSI CALL - TRACE USER SIO
D5 GOTO DMKDSPCH

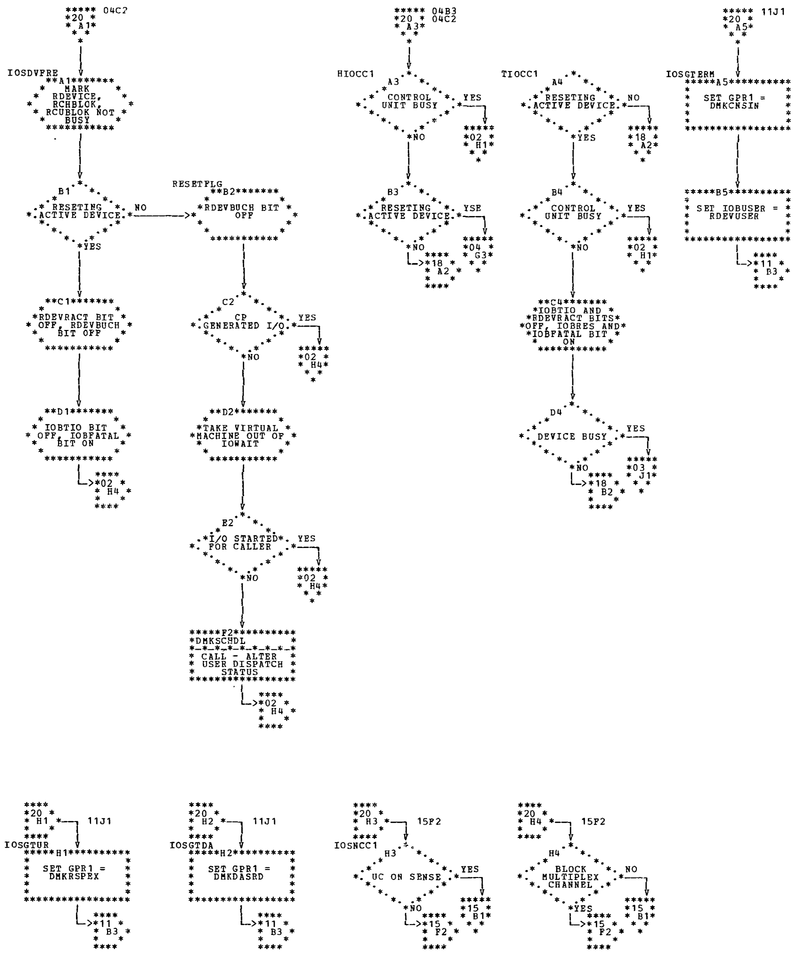
| DMKIOS -- I/O Supervisor (Parts 15 and 16 of 20)

| DMKIOS -- I/O Supervisor (Parts 17 and 18 of 20)



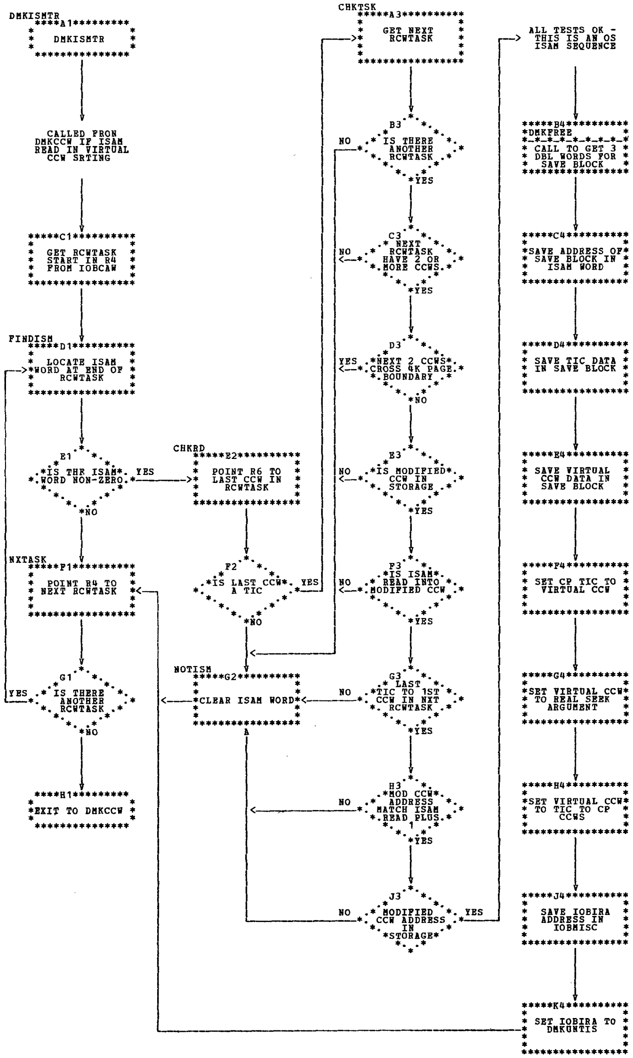


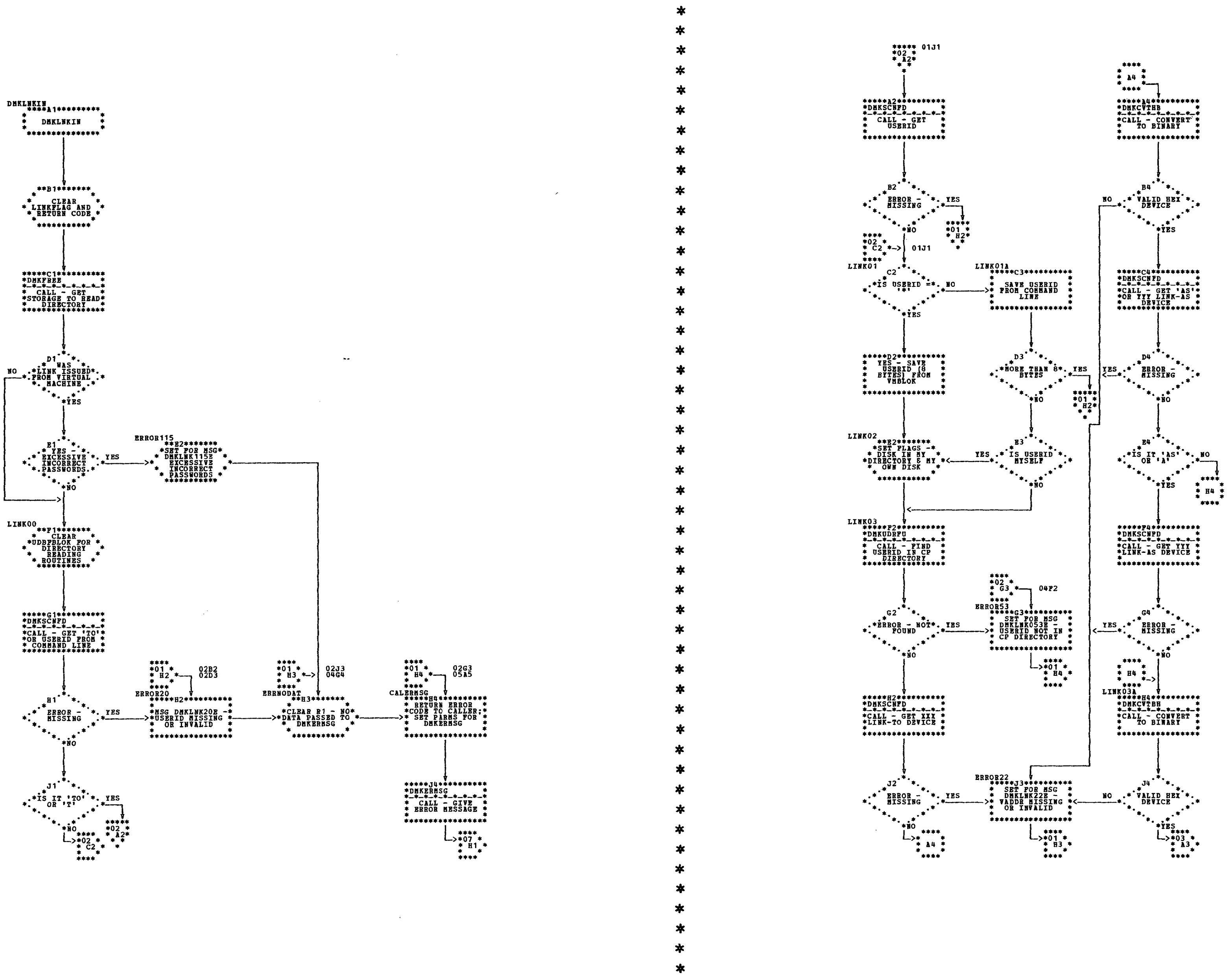




| DMKIOS -- I/O Supervisor (Parts 19 and 20 of 20)

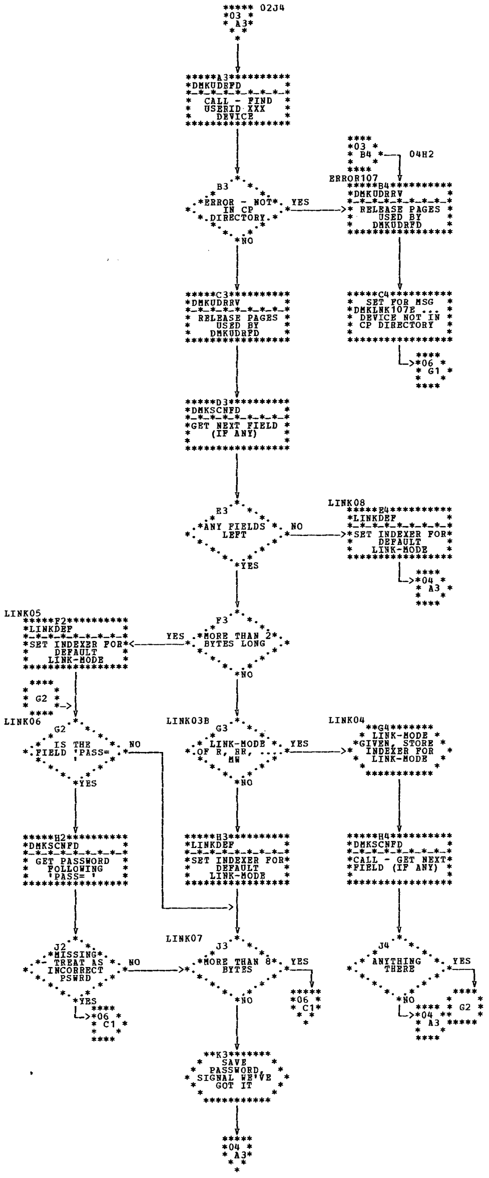
DMKISM -- Modify RCWTASK for OS ISAM Input/Output (Part 1 of 1)

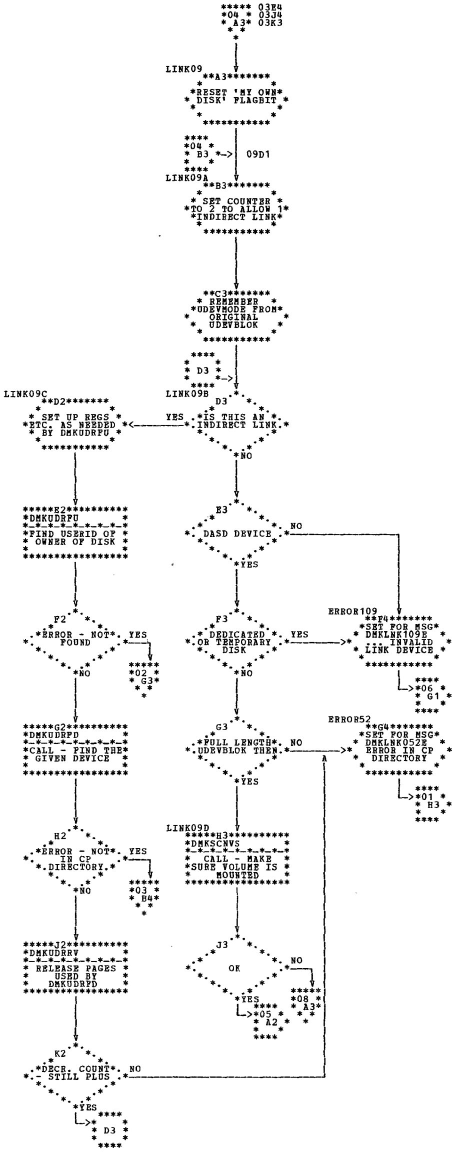

DMKISMTR
A1
DMKISMTR
CALLED FRON DMKCCW IF ISAM READ IN VIRTUAL CCW SRTING
C1
GET RCWTASK START IN R4 FROM IOBCAW
FINDISM
D1
LOCATE ISAM WORD AT END OF RCWTASK
E1
IS THR ISAM WORD NON-ZERO
YES
NO
CHKRD
E2
POINT R6 TO LAST CCW IN RCWTASK
F2
IS LAST CCW A TIC
YES
NO
NXTASK
F1
POINT R4 TO NEXT RCWTASK
G1
IS THERE ANOTHER RCWTASK
YES
NO
H1
EXIT TO DMKCCW
NOTISM
G2
CLEAR ISAM WORD
CHKTSK
A3
GET NEXT RCWTASK
B3
IS THERE ANOTHER RCWTASK
NO
YES
C3
NEXT RCWTASK HAVE 2 OR MORE CCWS
NO
YES
D3
NEXT 2 CCWS CROSS 4K PAGE BOUNDARY
YES
NO
E3
IS MODIFIED CCW IN STORAGE
NO
YES
F3
IS ISAM READ INTO MODIFIED CCW
NO
YES
G3
LAST TIC TO 1ST CCW IN NXT RCWTASK
NO
YES
H3
MOD CCW ADDRESS MATCH ISAM READ PLUS 1
NO
YES
J3
MODIFIED CCW ADDRESS IN STORAGE
NO
YES
ALL TESTS OK - THIS IS AN OS ISAM SEQUENCE
B4
DMKFREE
CALL TO GET 3 DBL WORDS FOR SAVE BLOCK
C4
SAVE ADDRESS OF SAVE BLOCK IN ISAM WORD
D4
SAVE TIC DATA IN SAVE BLOCK
E4
SAVE VIRTUAL CCW DATA IN SAVE BLOCK
F4
SET CP TIC TO VIRTUAL CCW
G4
SET VIRTUAL CCW TO REAL SEEK ARGUMENT
H4
SET VIRTUAL CCW TO TIC TO CP CCWS
J4
SAVE IOBIRA ADDRESS IN IOBMISC
K4
SET IOBIRA TO DMKUNTIS

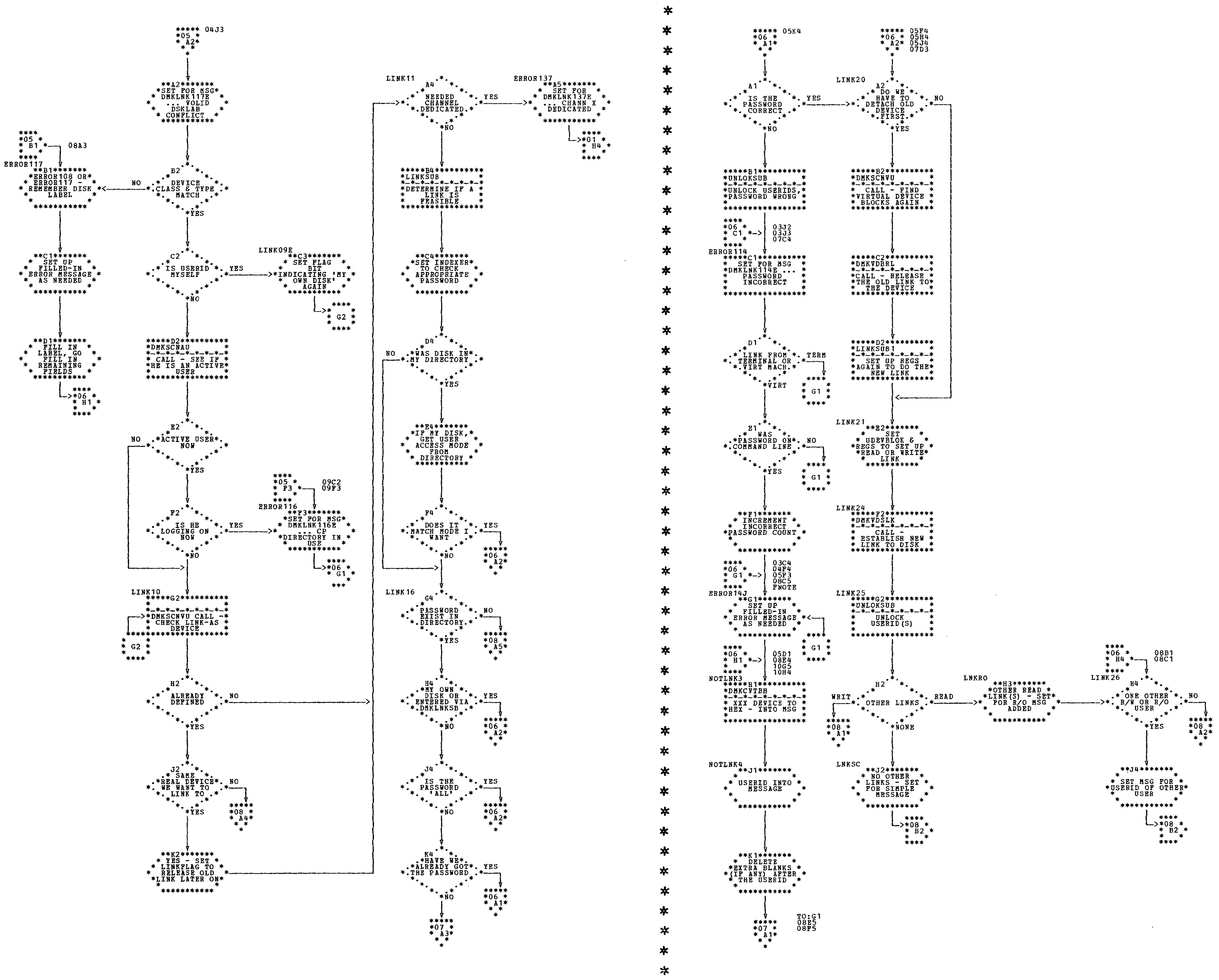

DMKLNKIN
A1
DMKLNKIN
B1
CLEAR LINKFLAG AND RETURN CODE
C1
DMKFREE
CALL - GET STORAGE TO READ DIRECTORY
D1
WAS LINK ISSUED FROM VIRTUAL MACHINE
NO
YES
E1
YES - EXCESSIVE INCORRECT PASSWORDS
YES
NO
ERROR115
E2
SET FOR MSG DMKLNK115E EXCESSIVE INCORRECT PASSWORDS
LINK00
F1
CLEAR UDBFBLOK FOR DIRECTORY READING ROUTINES
G1
DMKSCNFD
CALL - GET 'TO' OR USERID FROM COMMAND LINE
H1
ERROR - MISSING
YES
NO
01 H2
02B2
02D3
ERROR20
H2
MSG DMKLNK20E - USERID MISSING OR INVALID
01 H3
02J3
04G4
ERRNODAT
H3
CLEAR R1 - NO DATA PASSED TO DMKERMSG
01 H4
02G3
05A5
CALERMSG
H4
RETURN ERROR CODE TO CALLER; SET PARMS FOR DMKERMSG
J1
IS IT 'TO' OR 'T'
YES
NO
02 C2
02 A2
J4
DMKERMSG
CALL - GIVE ERROR MESSAGE
07 H1
02 A2
01J1
A2
DMKSCNFD
CALL - GET USERID
B2
ERROR - MISSING
YES
NO
01 H2
02 C2
01J1
LINK01
C2
IS USERID = '*'
NO
YES
LINK01A
C3
SAVE USERID FROM COMMAND LINE
D2
YES - SAVE USERID (8 BYTES) FROM VMBLOK
D3
MORE THAN 8 BYTES
YES
NO
01 H2
LINK02
E2
SET FLAGS - DISK IN MY DIRECTORY & MY OWN DISK
E3
IS USERID MYSELF
YES
NO
LINK03
F2
DMKUDRFU
CALL - FIND USERID IN CP DIRECTORY
G2
ERROR - NOT FOUND
YES
NO
02 G3
04F2
ERROR53
G3
SET FOR MSG DMKLNK053E - USERID NOT IN CP DIRECTORY
01 H4
H2
DMKSCNFD
CALL - GET XXX LINK-TO DEVICE
J2
ERROR - MISSING
YES
NO
A4
ERROR22
J3
SET FOR MSG DMKLNK22E - VADDR MISSING OR INVALID
01 H3
A4
A4
DMKCVTHB
CALL - CONVERT TO BINARY
B4
VALID HEX DEVICE
NO
YES
C4
DMKSCNFD
CALL - GET 'AS' OR YYY LINK-AS DEVICE
D4
ERROR - MISSING
YES
NO
E4
IS IT 'AS' OR 'A'
NO
YES
H4
F4
DMKSCNFD
CALL - GET YYY LINK-AS DEVICE
G4
ERROR - MISSING
YES
NO
H4
LINK03A
H4
DMKCVTBH
CALL - CONVERT TO BINARY
J4
VALID HEX DEVICE
NO
YES
03 A3

DMKLNK -- Process LINK Command; Link to a Virtual DASD (Parts 1 and 2 of 11)

DMKLNK -- Process LINK Command; Link to a Virtual DASD (Parts 3 and 4 of 11)

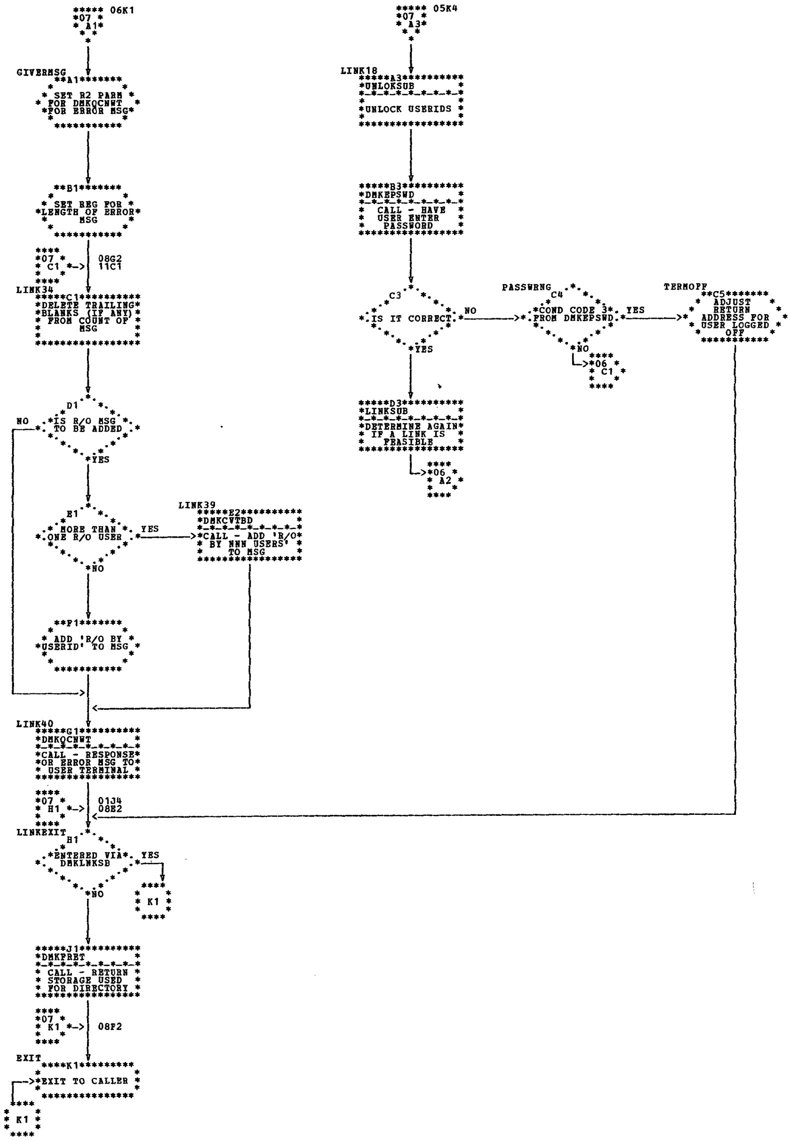

03 A3 02J4
A3 DMKUDRFD CALL - FIND USERID XXX DEVICE
B3 ERROR - NOT IN CP DIRECTORY
YES
NO
03 B4 04H2
ERROR107
B4 DMKUDRRV RELEASE PAGES USED BY DMKUDRFD
C3 DMKUDRRV RELEASE PAGES USED BY DMKUDRFD
C4 SET FOR MSG DMKLNK107E ... DEVICE NOT IN CP DIRECTORY
06 G1
D3 DMKSCNFD GET NEXT FIELD (IF ANY)
E3 ANY FIELDS LEFT
NO
YES
LINK08
E4 LINKDEF SET INDEXER FOR DEFAULT LINK-MODE
04 A3
LINK05
F2 LINKDEF SET INDEXER FOR DEFAULT LINK-MODE
F3 MORE THAN 2 BYTES LONG
YES
NO
G2
LINK06
G2 IS THE FIELD 'PASS= '
NO
YES
LINK03B
G3 LINK-MODE OF R, RR, ..., MW
YES
NO
LINK04
G4 LINK-MODE GIVEN, STORE INDEXER FOR LINK-MODE
H2 DMKSCNFD GET PASSWORD FOLLOWING 'PASS= '
H3 LINKDEF SET INDEXER FOR DEFAULT LINK-MODE
H4 DMKSCNFD CALL - GET NEXT FIELD (IF ANY)
J2 MISSING- TREAT AS INCORRECT PSWRD
NO
YES
06 C1
LINK07
J3 MORE THAN 8 BYTES
YES
NO
06 C1
J4 ANYTHING THERE
YES
NO
04 A3
G2
K3 SAVE PASSWORD, SIGNAL WE'VE GOT IT
04 A3

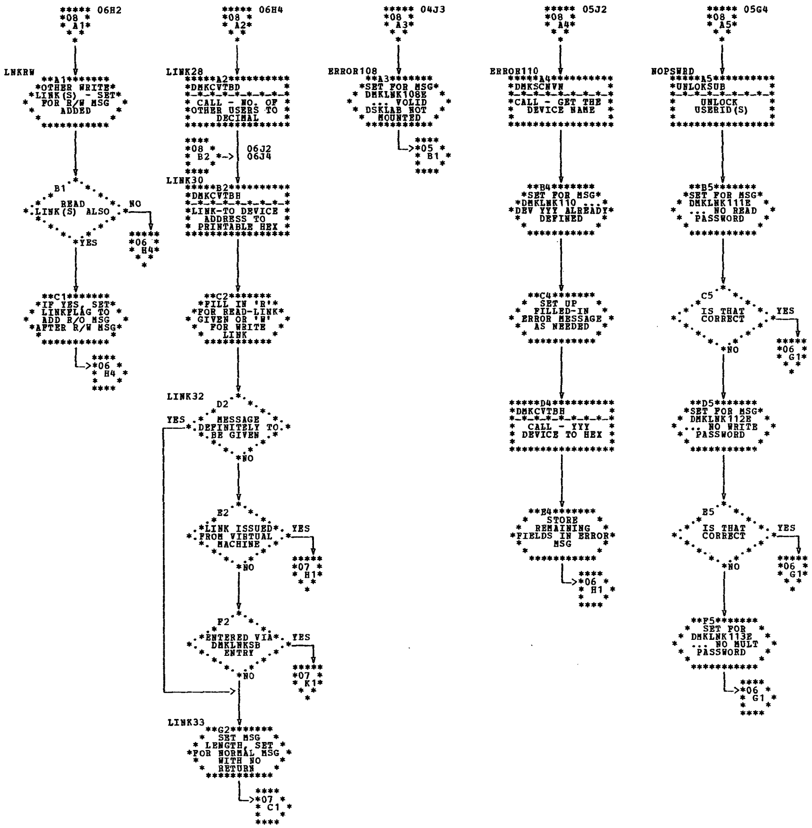

04 A3 03E4 03J4 03K3
LINK09
A3 RESET 'MY OWN DISK' FLAGBIT
04 B3 09D1
LINK09A
B3 SET COUNTER TO 2 TO ALLOW 1 INDIRECT LINK
C3 REMEMBER UDEVMODE FROM ORIGINAL UDEVBLOK
D3
LINK09C
D2 SET UP REGS ETC. AS NEEDED BY DMKUDRFU
LINK09B
D3 IS THIS AN INDIRECT LINK
YES
NO
E2 DMKUDRFU FIND USERID OF OWNER OF DISK
E3 DASD DEVICE
NO
YES
F2 ERROR - NOT FOUND
YES
NO
02 G3
F3 DEDICATED OR TEMPORARY DISK
YES
NO
ERROR109
F4 SET FOR MSG DMKLNK109E ... INVALID LINK DEVICE
06 G1
G2 DMKUDRFD CALL - FIND THE GIVEN DEVICE
G3 FULL LENGTH UDEVBLOK THEN
NO
YES
A
ERROR52
G4 SET FOR MSG DMKLNK052E ERROR IN CP DIRECTORY
01 H3
H2 ERROR - NOT IN CP DIRECTORY
YES
NO
03 B4
LINK09D
H3 DMKSCNVS CALL - MAKE SURE VOLUME IS MOUNTED
J2 DMKUDRRV RELEASE PAGES USED BY DMKUDRFD
J3 OK
NO
YES
05 A2
08 A3
K2 DECR. COUNT - STILL PLUS
NO
YES
D3

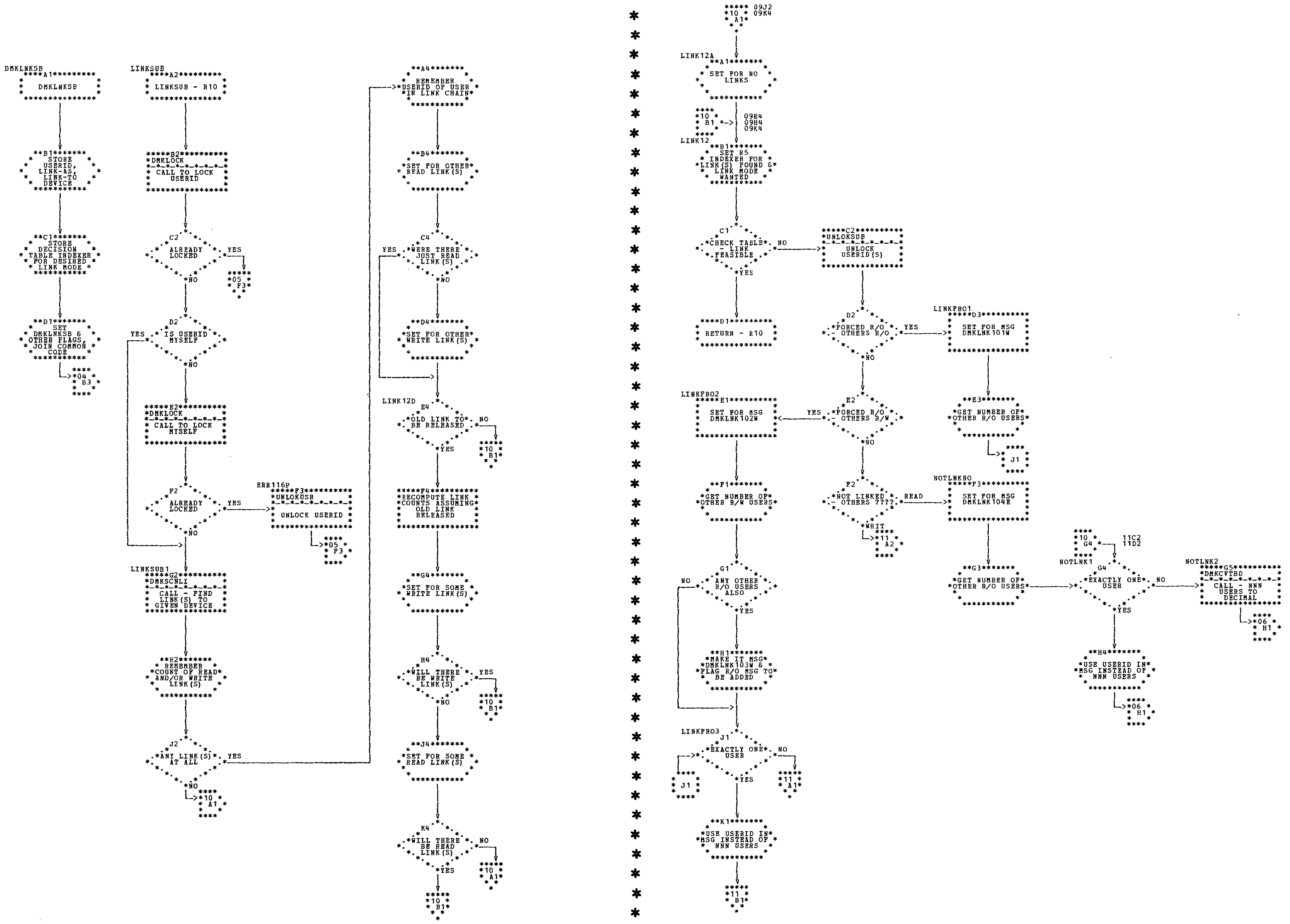

05 A2 04J3
A2 SET FOR MSG DMKLNK117E ... VOLID DSKLAB CONFLICT
05 B1 08A3
ERROR117
B1 ERROR108 OR ERROR117 - REMEMBER DISK LABEL
B2 DEVICE CLASS & TYPE MATCH
NO
YES
C1 SET UP FILLED-IN ERROR MESSAGE AS NEEDED
C2 IS USERID MYSELF
YES
NO
LINK09E
C3 SET FLAG BIT INDICATING 'MY OWN DISK' AGAIN
G2
D1 FILL IN LABEL, GO FILL IN REMAINING FIELDS
06 H1
D2 DMKSCNAU CALL - SEE IF HE IS AN ACTIVE USER
E2 ACTIVE USER NOW
NO
YES
05 F3 09C2 09F3
ERROR116
F2 IS HE LOGGING ON NOW
YES
NO
F3 SET FOR MSG DMKLNK116E ... CP DIRECTORY IN USE
06 G1
LINK10
G2 DMKSCNVU CALL - CHECK LINK-AS DEVICE
G2
H2 ALREADY DEFINED
NO
YES
J2 SAME REAL DEVICE WE WANT TO LINK TO
NO
YES
08 A4
K2 YES - SET LINKFLAG TO RELEASE OLD LINK LATER ON
LINK11
A4 NEEDED CHANNEL DEDICATED
YES
NO
ERROR137
A5 SET FOR DMKLNK137E ... CHANN X DEDICATED
01 H4
B4 LINKSUB DETERMINE IF A LINK IS FEASIBLE
C4 SET INDEXER TO CHECK APPROPRIATE PASSWORD
D4 WAS DISK IN MY DIRECTORY
NO
YES
E4 IF MY DISK, GET USER ACCESS MODE FROM DIRECTORY
F4 DOES IT MATCH MODE I WANT
YES
NO
06 A2
LINK16
G4 PASSWORD EXIST IN DIRECTORY
NO
YES
08 A5
H4 MY OWN DISK OR ENTERED VIA DMKLNKSB
YES
NO
06 A2
J4 IS THE PASSWORD 'ALL'
YES
NO
06 A2
K4 HAVE WE ALREADY GOT THE PASSWORD
YES
NO
06 A1
07 A3
06 A1 05K4
A1 IS THE PASSWORD CORRECT
YES
NO
06 A2 05F4 05H4 05J4 07D3
LINK20
A2 DO WE HAVE TO DETACH OLD DEVICE FIRST
NO
YES
B1 UNLOKSUB UNLOCK USERIDS, PASSWORD WRONG
B2 DMKSCNVU CALL - FIND VIRTUAL DEVICE BLOCKS AGAIN
06 C1 03J2 03J3 07C4
ERROR114
C1 SET FOR MSG DMKLNK114E ... PASSWORD INCORRECT
C2 DMKVDBRL CALL - RELEASE THE OLD LINK TO THE DEVICE
D1 LINK FROM TERMINAL OR VIRT MACH.
TERM
VIRT
G1
D2 LINKSUB1 SET UP REGS AGAIN TO DO THE NEW LINK
E1 WAS PASSWORD ON COMMAND LINE
NO
YES
G1
LINK21
E2 SET UDEVBLOK & REGS TO SET UP READ OR WRITE LINK
F1 INCREMENT INCORRECT PASSWORD COUNT
LINK24
F2 DMKVDSLK CALL - ESTABLISH NEW LINK TO DISK
06 G1 03C4 04F4 05F3 08C5 FNOTE
ERROR14J
G1 SET UP FILLED-IN ERROR MESSAGE AS NEEDED
G1
LINK25
G2 UNLOKSUB UNLOCK USERID(S)
06 H1 05D1 08E4 10G5 10H4
NOTLNK3
H1 DMKCVTBH XXX DEVICE TO HEX - INTO MSG
H2 OTHER LINKS
WRIT
READ
NONE
08 A1
LNKRO
H3 OTHER READ LINK(S) - SET FOR R/O MSG ADDED
06 H4 08B1 08C1
LINK26
H4 ONE OTHER R/W OR R/O USER
NO
YES
08 A2
NOTLNK4
J1 USERID INTO MESSAGE
LNKSC
J2 NO OTHER LINKS - SET FOR SIMPLE MESSAGE
08 B2
J4 SET MSG FOR USERID OF OTHER USER
08 B2
K1 DELETE EXTRA BLANKS (IF ANY) AFTER THE USERID
07 A1 TO:G1 08E5 08F5

DMKLNK -- Process LINK Ccmmand; Link to a Virtual DASD (Parts 5 and 6 of 11)

DMKLNK -- Process LINK Command; Link to a Virtual DASD (Parts 7 and 8 of 11)

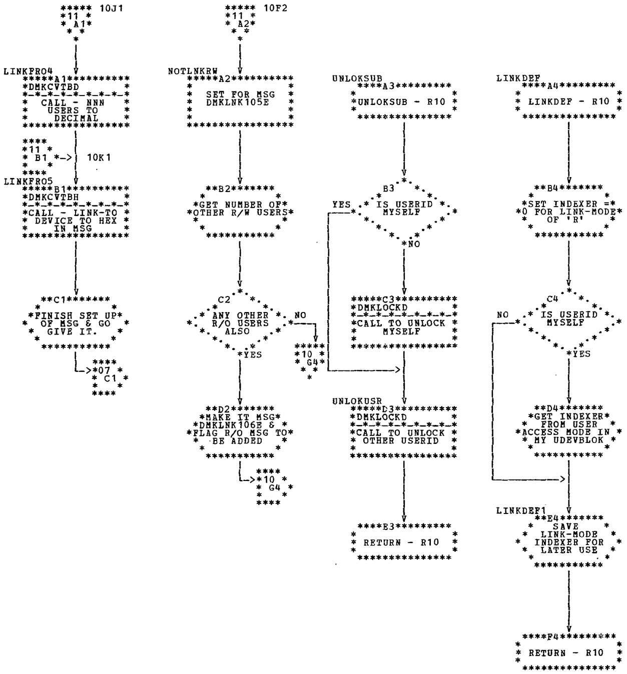

06K1
07 A1
GIVERMSG
A1
SET R2 PARM FOR DMKQCNWT FOR ERROR MSG
B1
SET REG FOR LENGTH OF ERROR MSG
07 C1
08G2
11C1
LINK34
C1
DELETE TRAILING BLANKS (IF ANY) FROM COUNT OF MSG
D1
IS R/O MSG TO BE ADDED
NO
YES
E1
MORE THAN ONE R/O USER
YES
NO
LINK39
E2
DMKCVTBD
CALL - ADD 'R/O BY NNN USERS' TO MSG
F1
ADD 'R/O BY USERID' TO MSG
LINK40
G1
DMKQCNWT
CALL - RESPONSE OR ERROR MSG TO USER TERMINAL
07 H1
01J4
08E2
LINKEXIT
H1
ENTERED VIA DMKLNKSB
YES
NO
K1
J1
DMKFRET
CALL - RETURN STORAGE USED FOR DIRECTORY
07 K1
08F2
EXIT
K1
EXIT TO CALLER
K1
05K4
07 A3
LINK18
A3
UNLOKSUB
UNLOCK USERIDS
B3
DMKEPSWD
CALL - HAVE USER ENTER PASSWORD
C3
IS IT CORRECT
NO
YES
D3
LINKSUB
DETERMINE AGAIN IF A LINK IS FEASIBLE
06 A2
PASSWRNG
C4
COND CODE 3 FROM DMKEPSWD
YES
NO
06 C1
TERMOFF
C5
ADJUST RETURN ADDRESS FOR USER LOGGED OFF

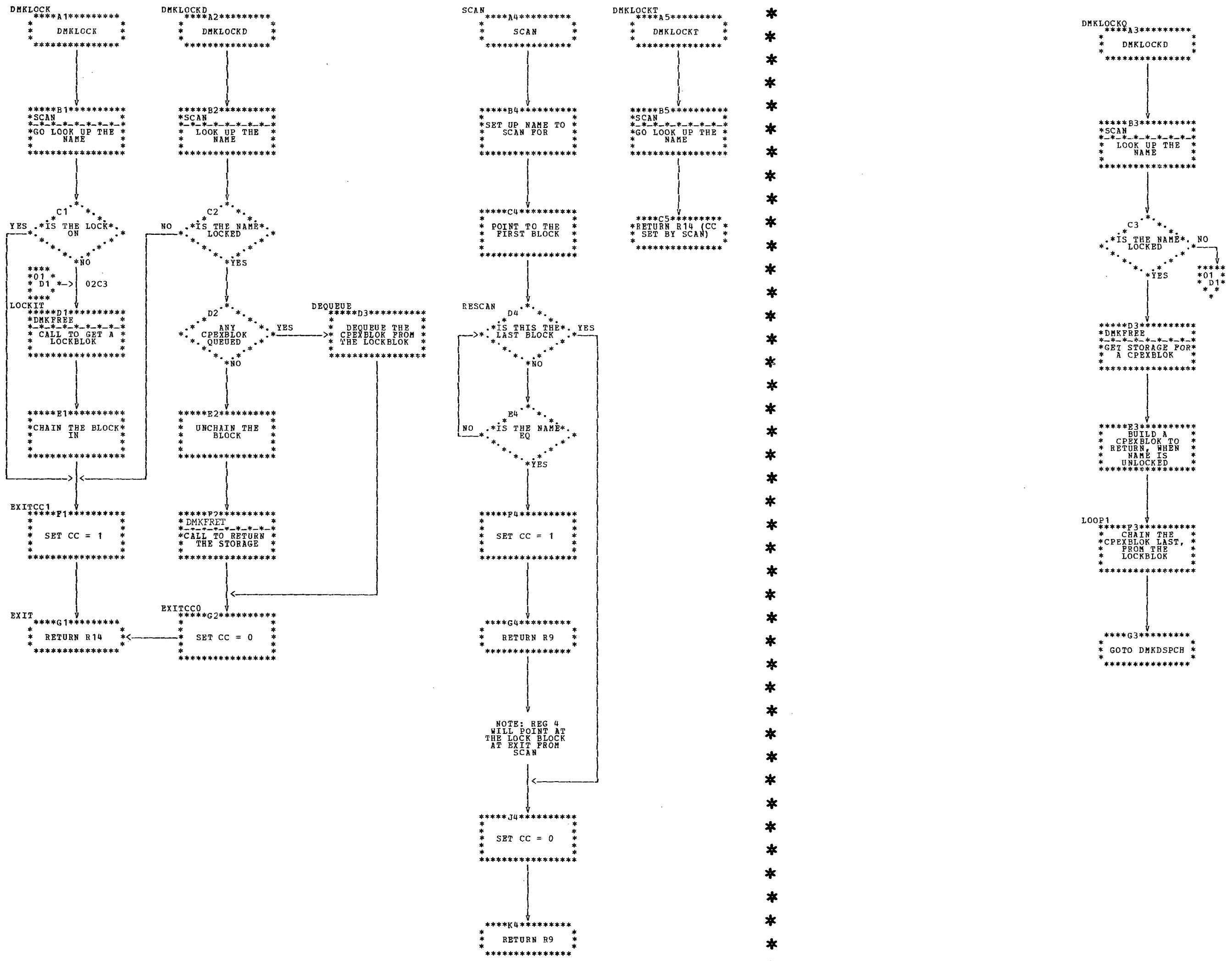

06H2
08 A1
LNKRW
A1
OTHER WRITE LINK(S) - SET FOR R/W MSG ADDED
B1
READ LINK(S) ALSO
NO
06 H4
YES
C1
IF YES, SET LINKFLAG TO ADD R/O MSG AFTER R/W MSG
06 H4
06H4
08 A2
LINK28
A2
DMKCVTBD
CALL - NO. OF OTHER USERS TO DECIMAL
08 B2
06J2
06J4
LINK30
B2
DMKCVTBH
LINK-TO DEVICE ADDRESS TO PRINTABLE HEX
C2
FILL IN 'R' FOR READ-LINK GIVEN OR 'W' FOR WRITE LINK
LINK32
D2
MESSAGE DEFINITELY TO BE GIVEN
YES
NO
E2
LINK ISSUED FROM VIRTUAL MACHINE
YES
07 H1
NO
F2
ENTERED VIA DMKLNKSB ENTRY
YES
07 K1
NO
LINK33
G2
SET MSG LENGTH, SET FOR NORMAL MSG WITH NO RETURN
07 C1
04J3
08 A3
ERROR108
A3
SET FOR MSG DMKLNK108E ... VOLID DSKLAB NOT MOUNTED
05 B1
05J2
08 A4
ERROR110
A4
DMKSCNVN
CALL - GET THE DEVICE NAME
B4
SET FOR MSG DMKLNK110 ... DEV YYY ALREADY DEFINED
C4
SET UP FILLED-IN ERROR MESSAGE AS NEEDED
D4
DMKCVTBH
CALL - YYY DEVICE TO HEX
E4
STORE REMAINING FIELDS IN ERROR MSG
06 H1
05G4
08 A5
NOPSWRD
A5
UNLOKSUB
UNLOCK USERID(S)
B5
SET FOR MSG DMKLNK111E ... NO READ PASSWORD
C5
IS THAT CORRECT
YES
06 G1
NO
D5
SET FOR MSG DMKLNK112E ... NO WRITE PASSWORD
E5
IS THAT CORRECT
YES
06 G1
NO
F5
SET FOR DMKLNK113E ... NO MULT PASSWORD
06 G1


DMKLNKSB
A1 DMKLNKSB
B1 STORE USERID, LINK-AS, LINK-TO DEVICE
C1 STORE DECISION TABLE INDEXER FOR DESIRED LINK MODE
D1 SET DMKLNKSB & OTHER FLAGS, JOIN COMMON CODE
04 B3
LINKSUB
A2 LINKSUB - R10
B2 DMKLOCK CALL TO LOCK USERID
C2 ALREADY LOCKED
YES
05 F3
NO
D2 IS USERID MYSELF
YES
NO
E2 DMKLOCK CALL TO LOCK MYSELF
F2 ALREADY LOCKED
YES
NO
ERR116P
F3 UNLOKUSR UNLOCK USERID
05 F3
LINKSUB1
G2 DMKSCNLI CALL - FIND LINK(S) TO GIVEN DEVICE
H2 REMEMBER COUNT OF READ AND/OR WRITE LINK(S)
J2 ANY LINK(S) AT ALL
YES
NO
10 A1
A4 REMEMBER USERID OF USER IN LINK CHAIN
B4 SET FOR OTHER READ LINK(S)
C4 WERE THERE JUST READ LINK(S)
YES
NO
D4 SET FOR OTHER WRITE LINK(S)
LINK12D
E4 OLD LINK TO BE RELEASED
NO
10 B1
YES
F4 RECOMPUTE LINK COUNTS ASSUMING OLD LINK RELEASED
G4 SET FOR SOME WRITE LINK(S)
H4 WILL THERE BE WRITE LINK(S)
YES
10 B1
NO
J4 SET FOR SOME READ LINK(S)
K4 WILL THERE BE READ LINK(S)
NO
10 A1
YES
10 B1
10 A1 09J2 09K4
LINK12A
A1 SET FOR NO LINKS
10 B1 09E4 09H4 09K4
LINK12
B1 SET R5 INDEXER FOR LINK(S) FOUND & LINK MODE WANTED
C1 CHECK TABLE - LINK FEASIBLE
NO
YES
C2 UNLOKSUB UNLOCK USERID(S)
D1 RETURN - R10
D2 FORCED R/O - OTHERS R/O
YES
NO
LINKFRO1
D3 SET FOR MSG DMKLNK101W
LINKFRO2
E1 SET FOR MSG DMKLNK102W
E2 FORCED R/O - OTHERS R/W
YES
NO
E3 GET NUMBER OF OTHER R/O USERS
J1
F1 GET NUMBER OF OTHER R/W USERS
F2 NOT LINKED - OTHERS ????
READ
WRIT
11 A2
NOTLNKRO
F3 SET FOR MSG DMKLNK104E
G1 ANY OTHER R/O USERS ALSO
NO
YES
H1 MAKE IT MSG DMKLNK103W & FLAG R/O MSG TO BE ADDED
LINKFRO3
J1 EXACTLY ONE USER
NO
11 A1
J1
YES
K1 USE USERID IN MSG INSTEAD OF NNN USERS
11 B1
G3 GET NUMBER OF OTHER R/O USERS
10 G4 11C2 11D2
NOTLNK1
G4 EXACTLY ONE USER
NO
YES
NOTLNK2
G5 DMKCVTBD CALL - NNN USERS TO DECIMAL
06 H1
H4 USE USERID IN MSG INSTEAD OF NNN USERS
06 H1

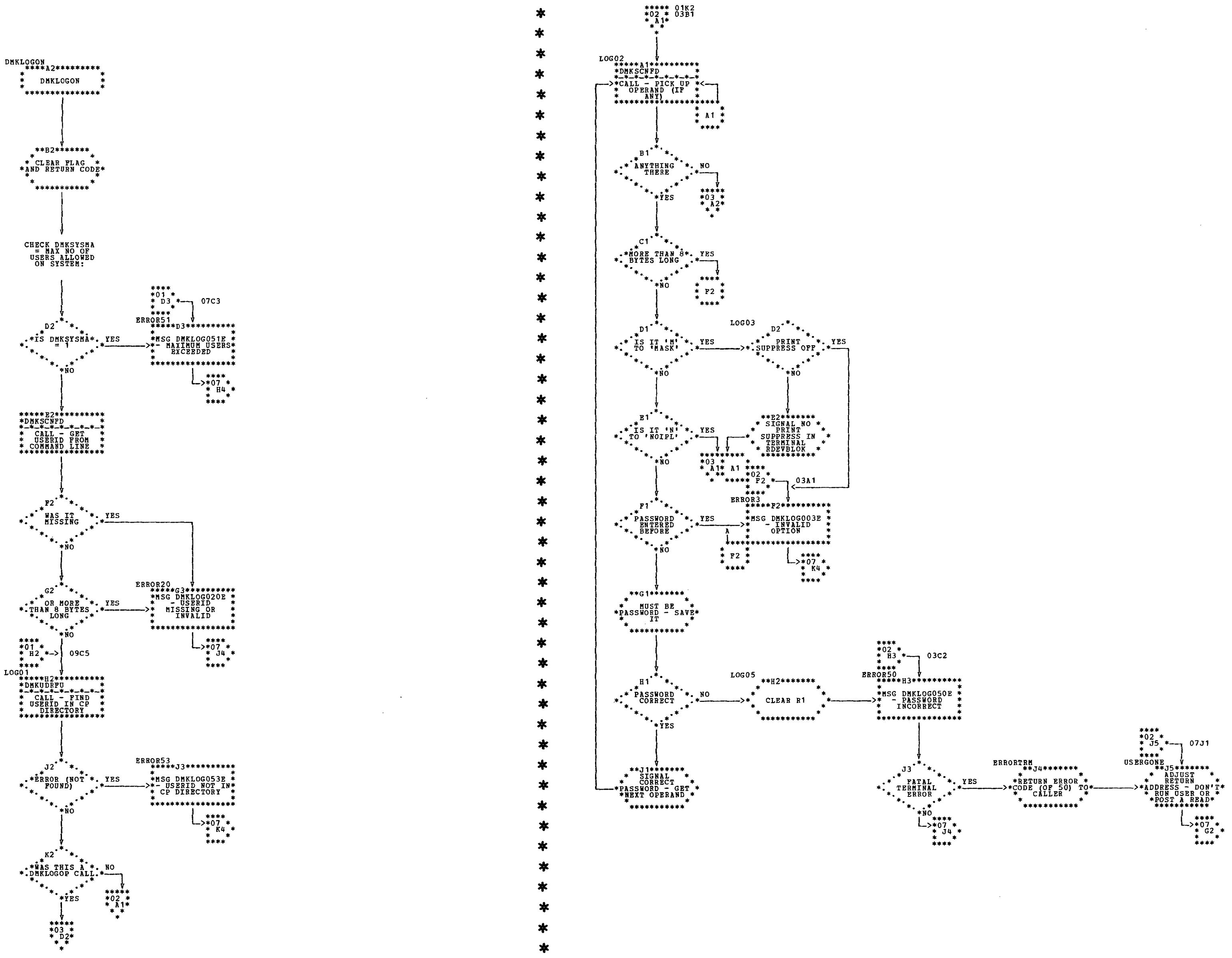
| DMKLNK -- Process LINK Command; Link to a Virtual DASD (Parts 9 and 10 of 11)

DMKLNK -- Process LINK Command; Link to a Virtual DASD (Part 11 of 11)



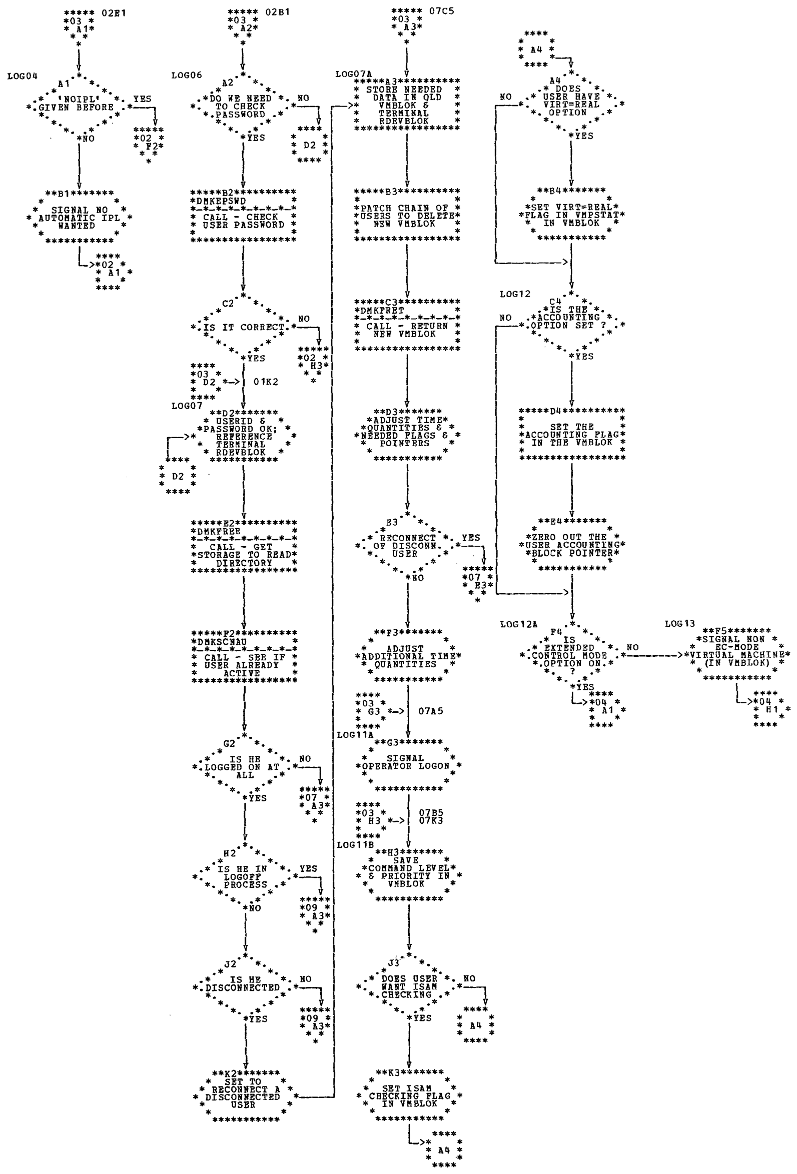


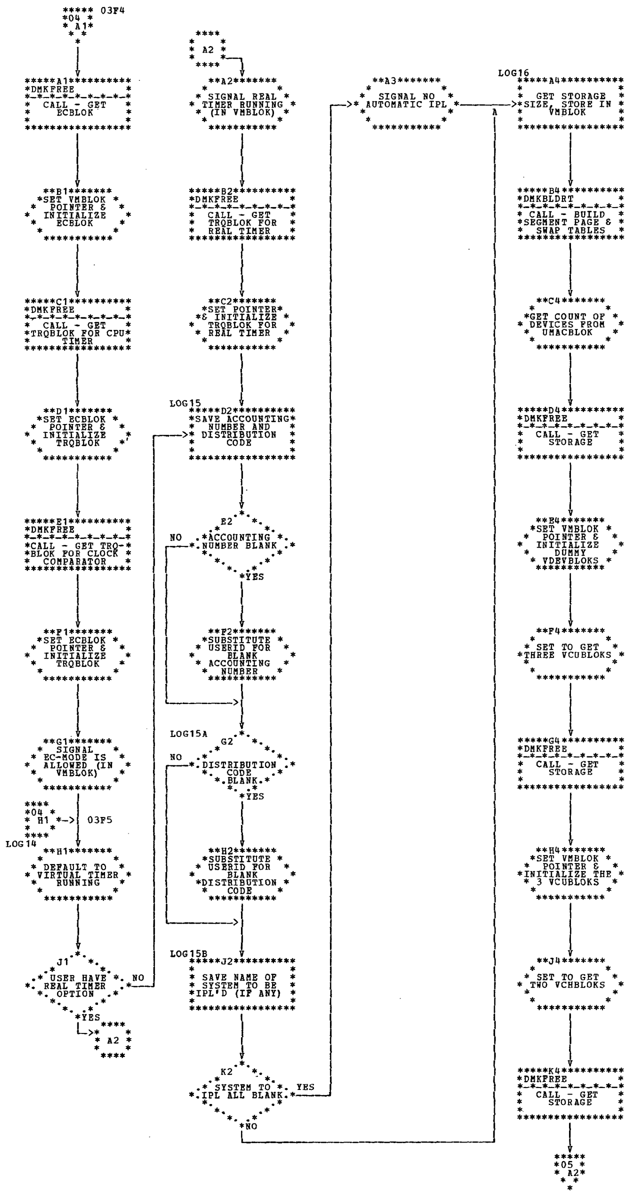
| DMKLOC -- User Lock Module (Parts 1 and 2 of 2)

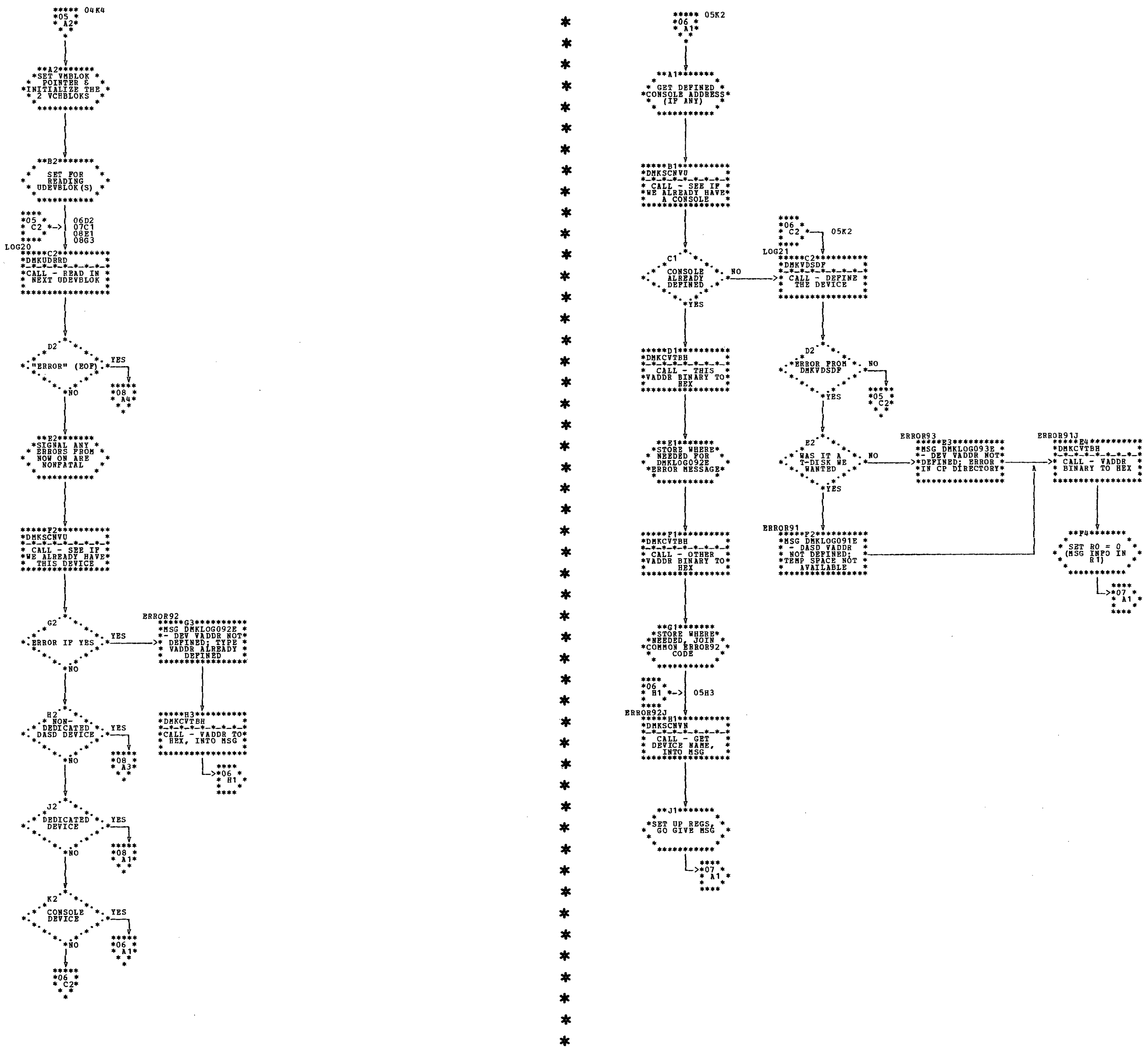

DMKLOGON
A2
DMKLOGON
B2
CLEAR FLAG AND RETURN CODE
CHECK DMKSYSMA = MAX NO OF USERS ALLOWED ON SYSTEM:
D2
IS DMKSYSMA = 1
YES
NO
01 D3
07C3
ERROR51
D3
MSG DMKLOG051E - MAXIMUM USERS EXCEEDED
07 H4
E2
DMKSCNFD
CALL - GET USERID FROM COMMAND LINE
F2
WAS IT MISSING
YES
NO
G2
OR MORE THAN 8 BYTES LONG
YES
NO
ERROR20
G3
MSG DMKLOG020E - USERID MISSING OR INVALID
07 J4
01 H2
09C5
LOG01
H2
DMKUDRFU
CALL - FIND USERID IN CP DIRECTORY
J2
ERROR (NOT FOUND)
YES
NO
ERROR53
J3
MSG DMKLOG053E - USERID NOT IN CP DIRECTORY
07 K4
K2
WAS THIS A DMKLOGOP CALL
NO
YES
02 A1
03 D2
02 A1
01K2
03B1
LOG02
A1
DMKSCNFD
CALL - PICK UP OPERAND (IF ANY)
A1
B1
ANYTHING THERE
NO
YES
03 A2
C1
MORE THAN 8 BYTES LONG
YES
NO
F2
D1
IS IT 'M' TO 'MASK'
YES
NO
LOG03
D2
PRINT SUPPRESS OFF
YES
NO
E1
IS IT 'N' TO 'NOIPL'
YES
NO
E2
SIGNAL NO PRINT SUPPRESS IN TERMINAL RDEVBLOK
03 A1
A1
02 F2
03A1
ERROR3
F1
PASSWORD ENTERED BEFORE
YES
NO
F2
MSG DMKLOG003E - INVALID OPTION
A
F2
07 K4
G1
MUST BE PASSWORD - SAVE IT
H1
PASSWORD CORRECT
NO
YES
LOG05
H2
CLEAR R1
02 H3
03C2
ERROR50
H3
MSG DMKLOG050E - PASSWORD INCORRECT
J1
SIGNAL CORRECT PASSWORD - GET NEXT OPERAND
J3
FATAL TERMINAL ERROR
YES
NO
07 J4
ERRORTRM
J4
RETURN ERROR CODE (OF 50) TO CALLER
02 J5
07J1
USERGONE
J5
ADJUST RETURN ADDRESS - DON'T RUN USER OR POST A READ
07 G2

| DMKLOG -- Process LOGON/LOGIN Command; Logon the User or Operator (Parts 1 and 2 of 9)

| DMKLOG -- Process LOGON/LOGIN Command; Logon the User or Operator (Parts 3 and 4 of 9)

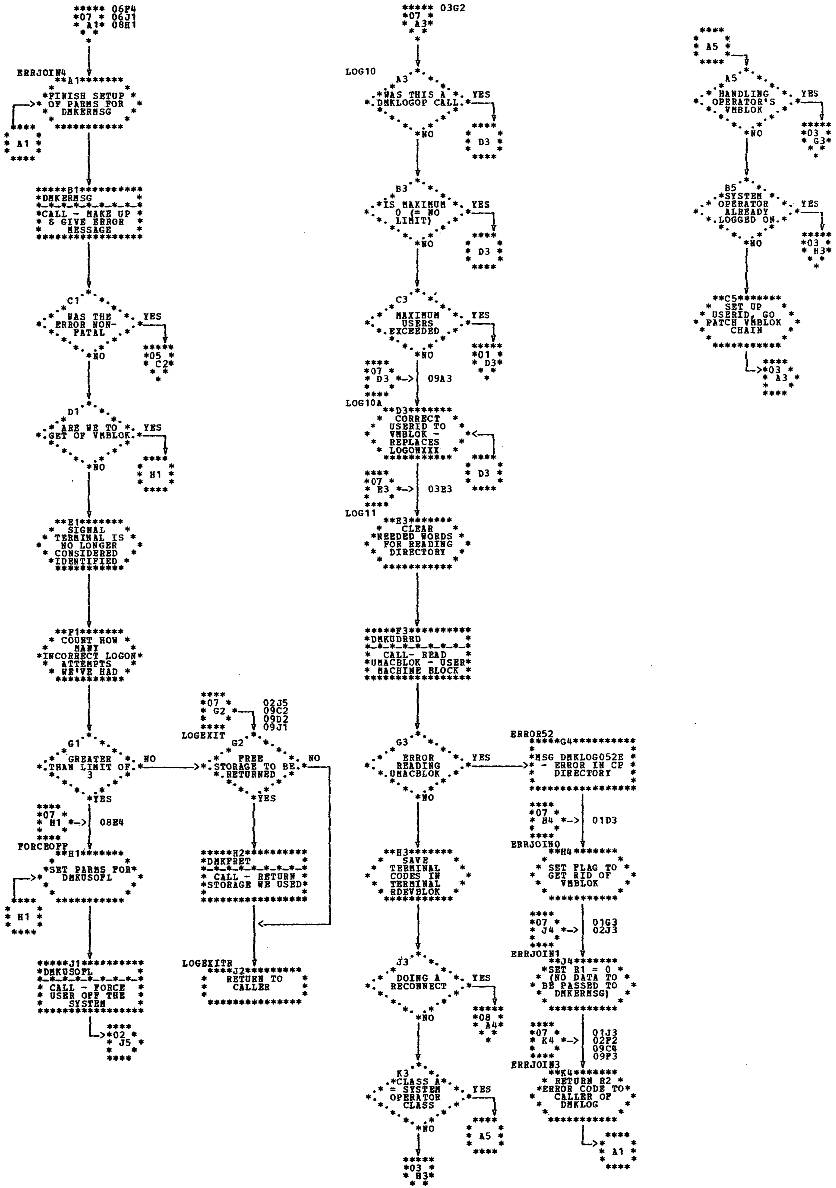

03 A1 02E1
LOG04
A1 'NOIPL' GIVEN BEFORE
YES 02 F2
NO
B1 SIGNAL NO AUTOMATIC IPL WANTED
02 A1
03 A2 02B1
LOG06
A2 DO WE NEED TO CHECK PASSWORD
NO D2
YES
B2 DMKEPSWD CALL - CHECK USER PASSWORD
C2 IS IT CORRECT
NO 02 H3
YES
03 D2 01K2
LOG07
D2 USERID & PASSWORD OK; REFERENCE TERMINAL RDEVBLOK
D2
E2 DMKFREE CALL - GET STORAGE TO READ DIRECTORY
F2 DMKSCNAU CALL - SEE IF USER ALREADY ACTIVE
G2 IS HE LOGGED ON AT ALL
NO 07 A3
YES
H2 IS HE IN LOGOFF PROCESS
YES 09 A3
NO
J2 IS HE DISCONNECTED
NO 09 A3
YES
K2 SET TO RECONNECT A DISCONNECTED USER
03 A3 07C5
LOG07A
A3 STORE NEEDED DATA IN OLD VMBLOK & TERMINAL RDEVBLOK
B3 PATCH CHAIN OF USERS TO DELETE NEW VMBLOK
C3 DMKFRET CALL - RETURN NEW VMBLOK
D3 ADJUST TIME QUANTITIES & NEEDED FLAGS & POINTERS
E3 RECONNECT OF DISCONN. USER
YES 07 E3
NO
F3 ADJUST ADDITIONAL TIME QUANTITIES
03 G3 07A5
LOG11A
G3 SIGNAL OPERATOR LOGON
03 H3 07B5 07K3
LOG11B
H3 SAVE COMMAND LEVEL & PRIORITY IN VMBLOK
J3 DOES USER WANT ISAM CHECKING
NO A4
YES
K3 SET ISAM CHECKING FLAG IN VMBLOK
A4
A4
A4 DOES USER HAVE VIRT=REAL OPTION
NO
YES
B4 SET VIRT=REAL FLAG IN VMPSTAT IN VMBLOK
LOG12
C4 IS THE ACCOUNTING OPTION SET ?
NO
YES
D4 SET THE ACCOUNTING FLAG IN THE VMBLOK
E4 ZERO OUT THE USER ACCOUNTING BLOCK POINTER
LOG12A
F4 IS EXTENDED CONTROL MODE OPTION ON. ?
NO
YES 04 A1
LOG13
F5 SIGNAL NON EC-MODE VIRTUAL MACHINE (IN VMBLOK)
04 H1

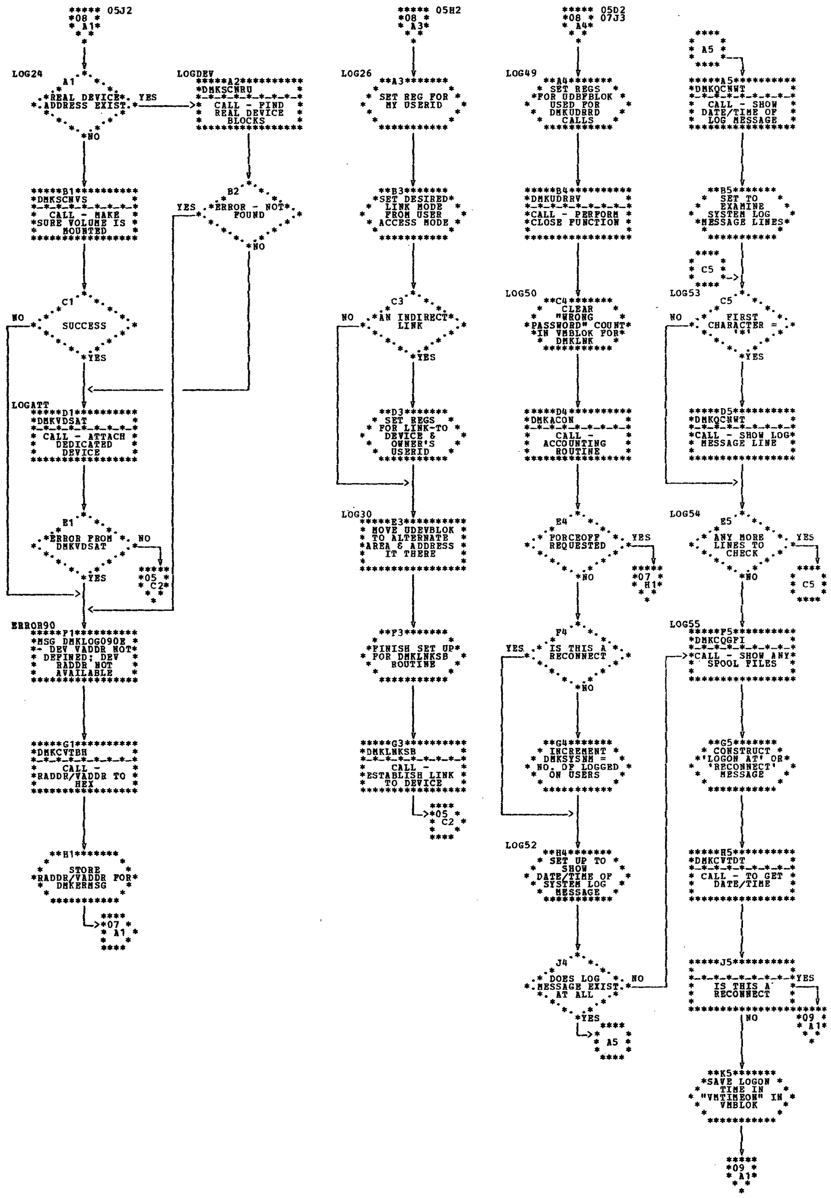

04 A1 03F4
A1 DMKFREE CALL - GET ECBLOK
B1 SET VMBLOK POINTER & INITIALIZE ECBLOK
C1 DMKFREE CALL - GET TRQBLOK FOR CPU TIMER
D1 SET ECBLOK POINTER & INITIALIZE TRQBLOK
E1 DMKFREE CALL - GET TRQ-BLOK FOR CLOCK COMPARATOR
F1 SET ECBLOK POINTER & INITIALIZE TRQBLOK
G1 SIGNAL EC-MODE IS ALLOWED (IN VMBLOK)
04 H1 03F5
LOG14
H1 DEFAULT TO VIRTUAL TIMER RUNNING
J1 USER HAVE REAL TIMER OPTION
NO
YES A2
A2
A2 SIGNAL REAL TIMER RUNNING (IN VMBLOK)
B2 DMKFREE CALL - GET TRQBLOK FOR REAL TIMER
C2 SET POINTER & INITIALIZE TRQBLOK FOR REAL TIMER
LOG15
D2 SAVE ACCOUNTING NUMBER AND DISTRIBUTION CODE
E2 ACCOUNTING NUMBER BLANK
NO
YES
F2 SUBSTITUTE USERID FOR BLANK ACCOUNTING NUMBER
LOG15A
G2 DISTRIBUTION CODE BLANK
NO
YES
H2 SUBSTITUTE USERID FOR BLANK DISTRIBUTION CODE
LOG15B
J2 SAVE NAME OF SYSTEM TO BE IPL'D (IF ANY)
K2 SYSTEM TO IPL ALL BLANK
YES
NO
A3 SIGNAL NO AUTOMATIC IPL
A
LOG16
A4 GET STORAGE SIZE, STORE IN VMBLOK
B4 DMKBLDRT CALL - BUILD SEGMENT PAGE & SWAP TABLES
C4 GET COUNT OF DEVICES FROM UMACBLOK
D4 DMKFREE CALL - GET STORAGE
E4 SET VMBLOK POINTER & INITIALIZE DUMMY VDEVBLOKS
F4 SET TO GET THREE VCUBLOKS
G4 DMKFREE CALL - GET STORAGE
H4 SET VMBLOK POINTER & INITIALIZE THE 3 VCUBLOKS
J4 SET TO GET TWO VCHBLOKS
K4 DMKFREE CALL - GET STORAGE
05 A2

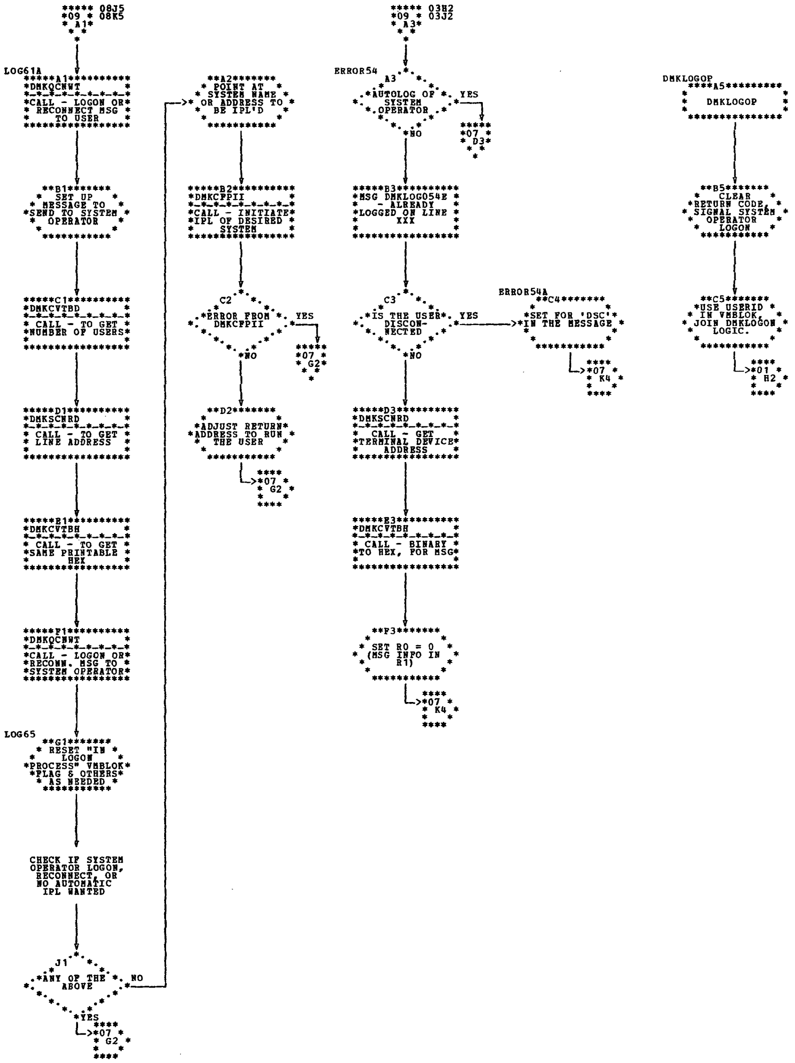

04K4
05 A2
A2 SET VMBLOK POINTER & INITIALIZE THE 2 VCHBLOKS
B2 SET FOR READING UDEVBLOK(S)
05 C2
06D2
07C1
08E1
08G3
LOG20
C2 DMKUDRRD
CALL - READ IN NEXT UDEVBLOK
D2 "ERROR" (EOF)
YES
08 A4
NO
E2 SIGNAL ANY ERRORS FROM NOW ON ARE NONFATAL
F2 DMKSCNVU
CALL - SEE IF WE ALREADY HAVE THIS DEVICE
G2 ERROR IF YES
YES
ERROR92
G3 MSG DMKLOG092E - DEV VADDR NOT DEFINED; TYPE VADDR ALREADY DEFINED
NO
H3 DMKCVTBH
CALL - VADDR TO HEX, INTO MSG
06 H1
H2 NON-DEDICATED DASD DEVICE
YES
08 A3
NO
J2 DEDICATED DEVICE
YES
08 A1
NO
K2 CONSOLE DEVICE
YES
06 A1
NO
06 C2
05K2
06 A1
A1 GET DEFINED CONSOLE ADDRESS (IF ANY)
B1 DMKSCNVU
CALL - SEE IF WE ALREADY HAVE A CONSOLE
06 C2
05K2
LOG21
C1 CONSOLE ALREADY DEFINED
NO
C2 DMKVDSDF
CALL - DEFINE THE DEVICE
YES
D1 DMKCVTBH
CALL - THIS VADDR BINARY TO HEX
D2 ERROR FROM DMKVDSDF
NO
05 C2
YES
E1 STORE WHERE NEEDED FOR DMKLOG092E ERROR MESSAGE
E2 WAS IT A T-DISK WE WANTED
NO
ERROR93
E3 MSG DMKLOG093E - DEV VADDR NOT DEFINED; ERROR IN CP DIRECTORY
A
ERROR91J
E4 DMKCVTBH
CALL - VADDR BINARY TO HEX
YES
F1 DMKCVTBH
CALL - OTHER VADDR BINARY TO HEX
ERROR91
F2 MSG DMKLOG091E - DASD VADDR NOT DEFINED; TEMP SPACE NOT AVAILABLE
F4 SET R0 = 0 (MSG INFO IN R1)
07 A1
G1 STORE WHERE NEEDED, JOIN COMMON ERROR92 CODE
06 H1
05H3
ERROR92J
H1 DMKSCNVN
CALL - GET DEVICE NAME, INTO MSG
J1 SET UP REGS, GO GIVE MSG
07 A1

| DMKLOG -- Process LOGON/LOGIN Command; Logon the User or Operator (Parts 5 and 6 of 9)

DMKLOG -- Process LOGON/LOGIN Command; Logon the User or Operator (Part 7 and 8 of 9)

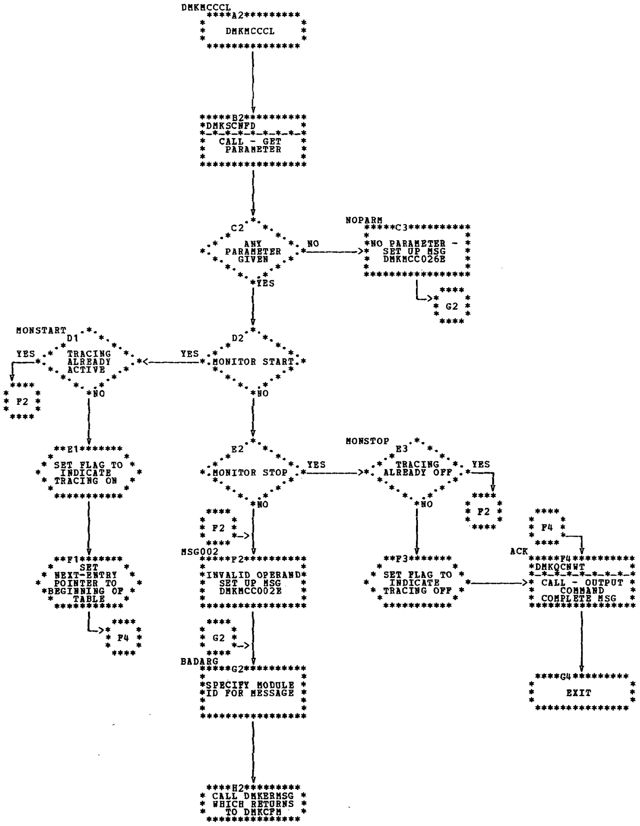

07 A1 06F4 06J1 08H1
ERRJOIN4
A1 FINISH SETUP OF PARMS FOR DMKERMSG
A1
B1 DMKERMSG CALL - MAKE UP & GIVE ERROR MESSAGE
C1 WAS THE ERROR NON-FATAL
YES
05 C2
NO
D1 ARE WE TO GET OF VMBLOK.
YES
H1
NO
E1 SIGNAL TERMINAL IS NO LONGER CONSIDERED IDENTIFIED
F1 COUNT HOW MANY INCORRECT LOGON ATTEMPTS WE'VE HAD
G1 GREATER THAN LIMIT OF 3
NO
YES
07 H1 08E4
FORCEOFF
H1 SET PARMS FOR DMKUSOFL
H1
J1 DMKUSOFL CALL - FORCE USER OFF THE SYSTEM
02 J5
07 G2 02J5 09C2 09D2 09J1
LOGEXIT
G2 FREE STORAGE TO BE RETURNED
NO
YES
H2 DMKFRET CALL - RETURN STORAGE WE USED
LOGEXITR
J2 RETURN TO CALLER
07 A3 03G2
LOG10
A3 WAS THIS A DMKLOGOP CALL.
YES
D3
NO
B3 IS MAXIMUM 0 (= NO LIMIT)
YES
D3
NO
C3 MAXIMUM USERS EXCEEDED
YES
01 D3
NO
07 D3 09A3
LOG10A
D3 CORRECT USERID TO VMBLOK - REPLACES LOGONXXX
D3
07 E3 03E3
LOG11
E3 CLEAR NEEDED WORDS FOR READING DIRECTORY
F3 DMKUDRRD CALL- READ UMACBLOK - USER MACHINE BLOCK
G3 ERROR READING UMACBLOK
YES
NO
ERROR52
G4 MSG DMKLOG052E - ERROR IN CP DIRECTORY
07 H4 01D3
ERRJOIN0
H4 SET FLAG TO GET RID OF VMBLOK
07 J4 01G3 02J3
ERRJOIN1
J4 SET R1 = 0 (NO DATA TO BE PASSED TO DMKERMSG)
07 K4 01J3 02F2 09C4 09F3
ERRJOIN3
K4 RETURN R2 ERROR CODE TO CALLER OF DMKLOG
A1
H3 SAVE TERMINAL CODES IN TERMINAL RDEVBLOK
J3 DOING A RECONNECT
YES
08 A4
NO
K3 CLASS A = SYSTEM OPERATOR CLASS
YES
A5
NO
03 H3
A5
A5 HANDLING OPERATOR'S VMBLOK
YES
03 G3
NO
B5 SYSTEM OPERATOR ALREADY LOGGED ON.
YES
03 H3
NO
C5 SET UP USERID, GO PATCH VMBLOK CHAIN
03 A3

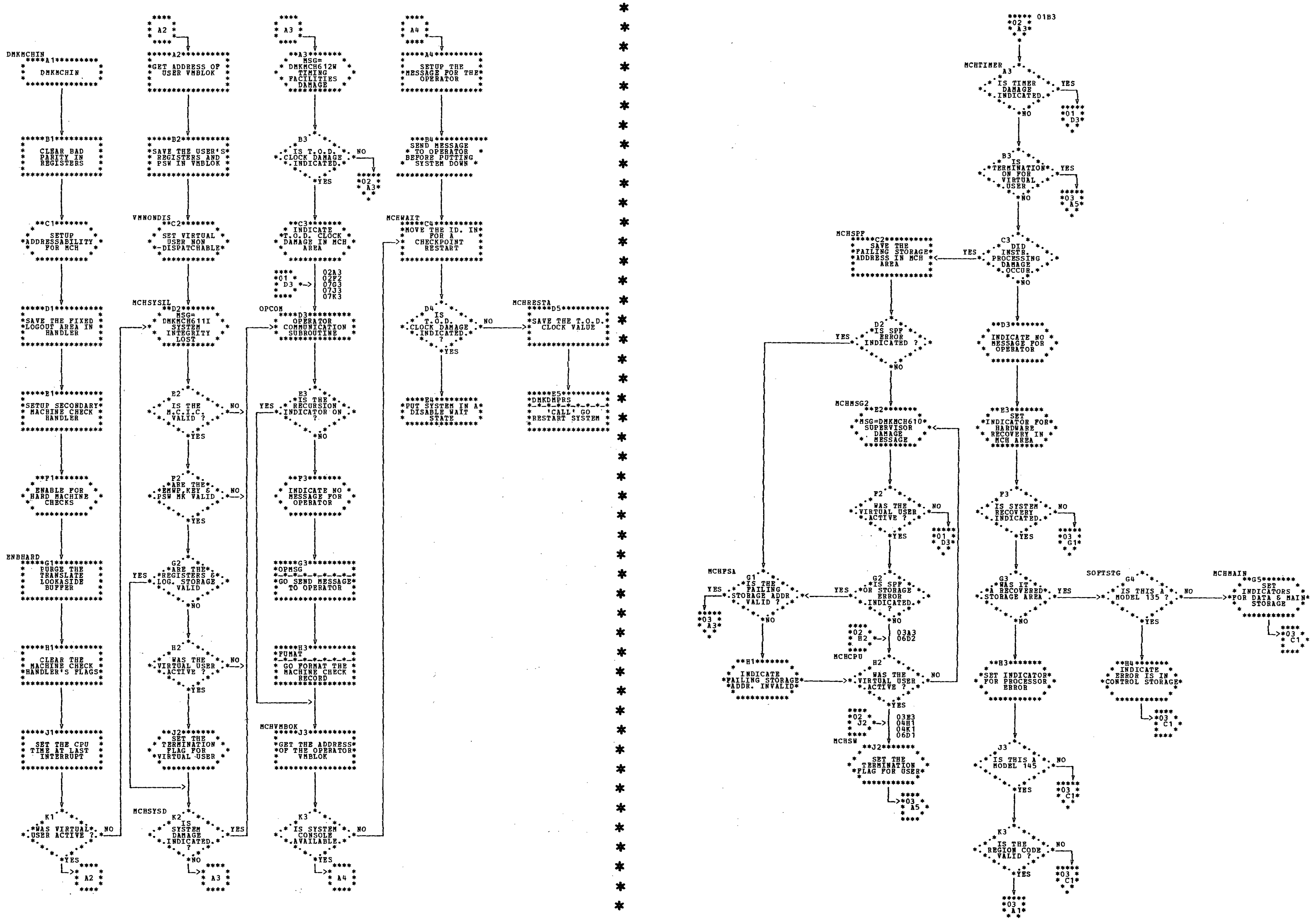

08 A1 05J2
LOG24
A1 REAL DEVICE ADDRESS EXIST.
YES
NO
LOGDEV
A2 DMKSCNRU CALL - FIND REAL DEVICE BLOCKS
B2 ERROR - NOT FOUND
YES
NO
B1 DMKSCNVS CALL - MAKE SURE VOLUME IS MOUNTED
C1 SUCCESS
NO
YES
LOGATT
D1 DMKVDSAT CALL - ATTACH DEDICATED DEVICE
E1 ERROR FROM DMKVDSAT
NO
05 C2
YES
ERROR90
F1 MSG DMKLOG090E - DEV VADDR NOT DEFINED; DEV RADDR NOT AVAILABLE
G1 DMKCVTBH CALL - RADDR/VADDR TO HEX
H1 STORE RADDR/VADDR FOR DMKERMSG
07 A1
08 A3 05H2
LOG26
A3 SET REG FOR MY USERID
B3 SET DESIRED LINK MODE FROM USER ACCESS MODE
C3 AN INDIRECT LINK
NO
YES
D3 SET REGS FOR LINK-TO DEVICE & OWNER'S USERID
LOG30
E3 MOVE UDEVBLOK TO ALTERNATE AREA & ADDRESS IT THERE
F3 FINISH SET UP FOR DMKLNKSB ROUTINE
G3 DMKLNKSB CALL - ESTABLISH LINK TO DEVICE
05 C2
08 A4 05D2 07J3
LOG49
A4 SET REGS FOR UDBFBLOK USED FOR DMKUDRRD CALLS
B4 DMKUDRRV CALL - PERFORM CLOSE FUNCTION
LOG50
C4 CLEAR "WRONG PASSWORD" COUNT IN VMBLOK FOR DMKLNK
D4 DMKACON CALL - ACCOUNTING ROUTINE
E4 FORCEOFF REQUESTED
YES
07 H1
NO
F4 IS THIS A RECONNECT
YES
NO
G4 INCREMENT DMKSYSNM = NO. OF LOGGED ON USERS
LOG52
H4 SET UP TO SHOW DATE/TIME OF SYSTEM LOG MESSAGE
J4 DOES LOG MESSAGE EXIST. AT ALL
NO
YES
A5
A5
A5 DMKQCNWT CALL - SHOW DATE/TIME OF LOG MESSAGE
B5 SET TO EXAMINE SYSTEM LOG MESSAGE LINES
C5
LOG53
C5 FIRST CHARACTER = '*'
NO
YES
D5 DMKQCNWT CALL - SHOW LOG MESSAGE LINE
LOG54
E5 ANY MORE LINES TO CHECK
YES
C5
NO
LOG55
F5 DMKCQGFI CALL - SHOW ANY SPOOL FILES
G5 CONSTRUCT 'LOGON AT' OR 'RECONNECT' MESSAGE
H5 DMKCVTDT CALL - TO GET DATE/TIME
J5 IS THIS A RECONNECT
YES
09 A1
NO
K5 SAVE LOGON TIME IN "VMTIMEON" IN VMBLOK
09 A1


09 A1 08J5 08K5
LOG61A
A1 DMKQCNWT CALL - LOGON OR RECONNECT MSG TO USER
B1 SET UP MESSAGE TO SEND TO SYSTEM OPERATOR
C1 DMKCVTBD CALL - TO GET NUMBER OF USERS
D1 DMKSCNRD CALL - TO GET LINE ADDRESS
E1 DMKCVTBH CALL - TO GET SAME PRINTABLE HEX
F1 DMKQCNWT CALL - LOGON OR RECONN. MSG TO SYSTEM OPERATOR
LOG65
G1 RESET "IN LOGON PROCESS" VMBLOK FLAG & OTHERS AS NEEDED
CHECK IF SYSTEM OPERATOR LOGON, RECONNECT, OR NO AUTOMATIC IPL WANTED
J1 ANY OF THE ABOVE
NO
YES
07 G2
A2 POINT AT SYSTEM NAME OR ADDRESS TO BE IPL'D
B2 DMKCFPII CALL - INITIATE IPL OF DESIRED SYSTEM
C2 ERROR FROM DMKCFPII
YES
07 G2
NO
D2 ADJUST RETURN ADDRESS TO RUN THE USER
07 G2
09 A3 03H2 03J2
ERROR54
A3 AUTOLOG OF SYSTEM OPERATOR
YES
07 D3
NO
B3 MSG DMKLOG054E - ALREADY LOGGED ON LINE XXX
C3 IS THE USER DISCON-NECTED
YES
NO
ERROR54A
C4 SET FOR 'DSC' IN THE MESSAGE
07 K4
D3 DMKSCNRD CALL - GET TERMINAL DEVICE ADDRESS
E3 DMKCVTBH CALL - BINARY TO HEX, FOR MSG
F3 SET R0 = 0 (MSG INFO IN R1)
07 K4
DMKLOGOP
A5 DMKLOGOP
B5 CLEAR RETURN CODE, SIGNAL SYSTEM OPERATOR LOGON
C5 USE USERID IN VMBLOK, JOIN DMKLOGON LOGIC.
01 H2

| DMKLOG -- Process LOGON/LOGIN Command; Logon the User or Operator (Part 9 of 9)

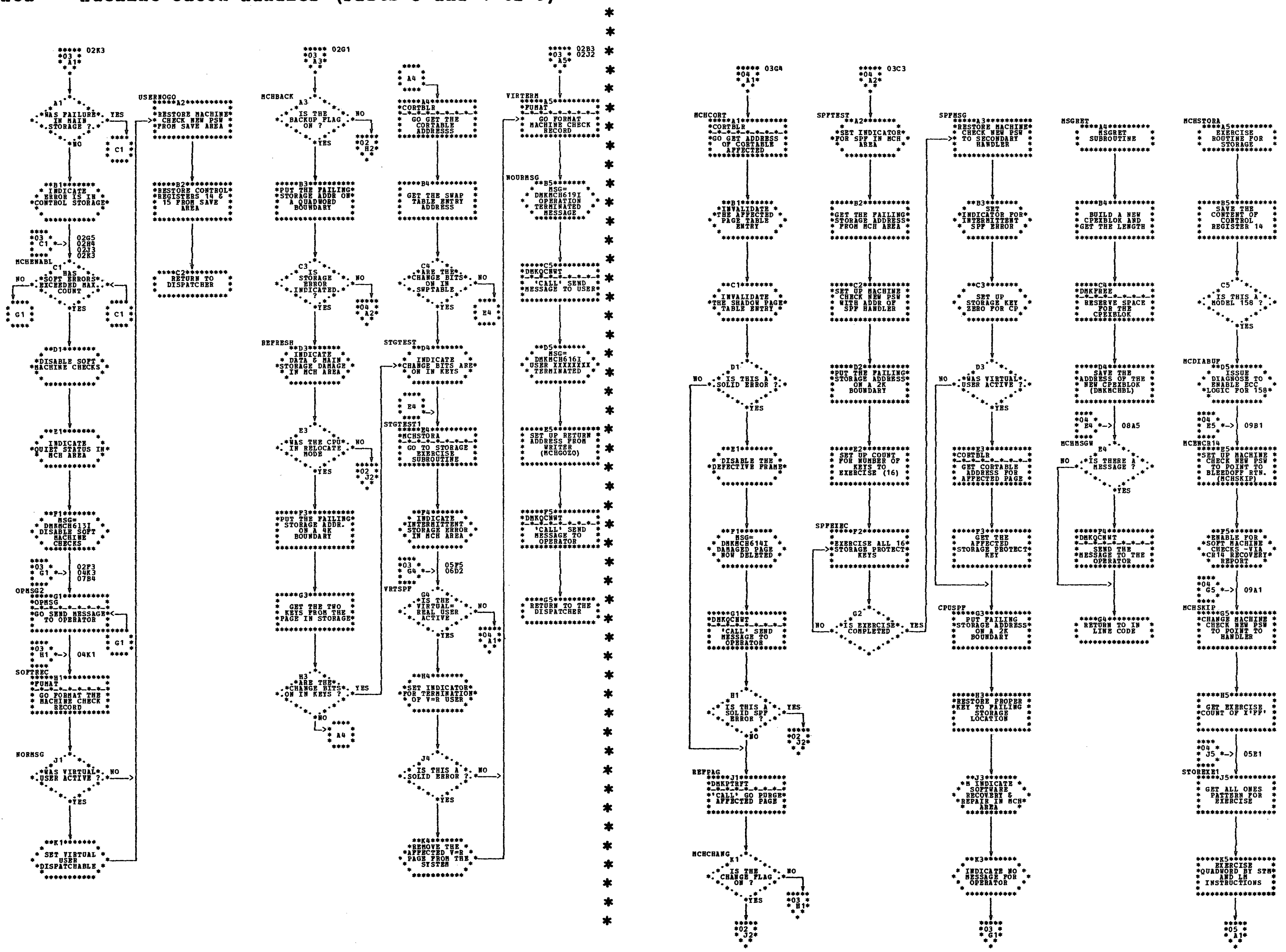
| DMKMCC -- VM Monitor Command Handler (Part 1 of 1)

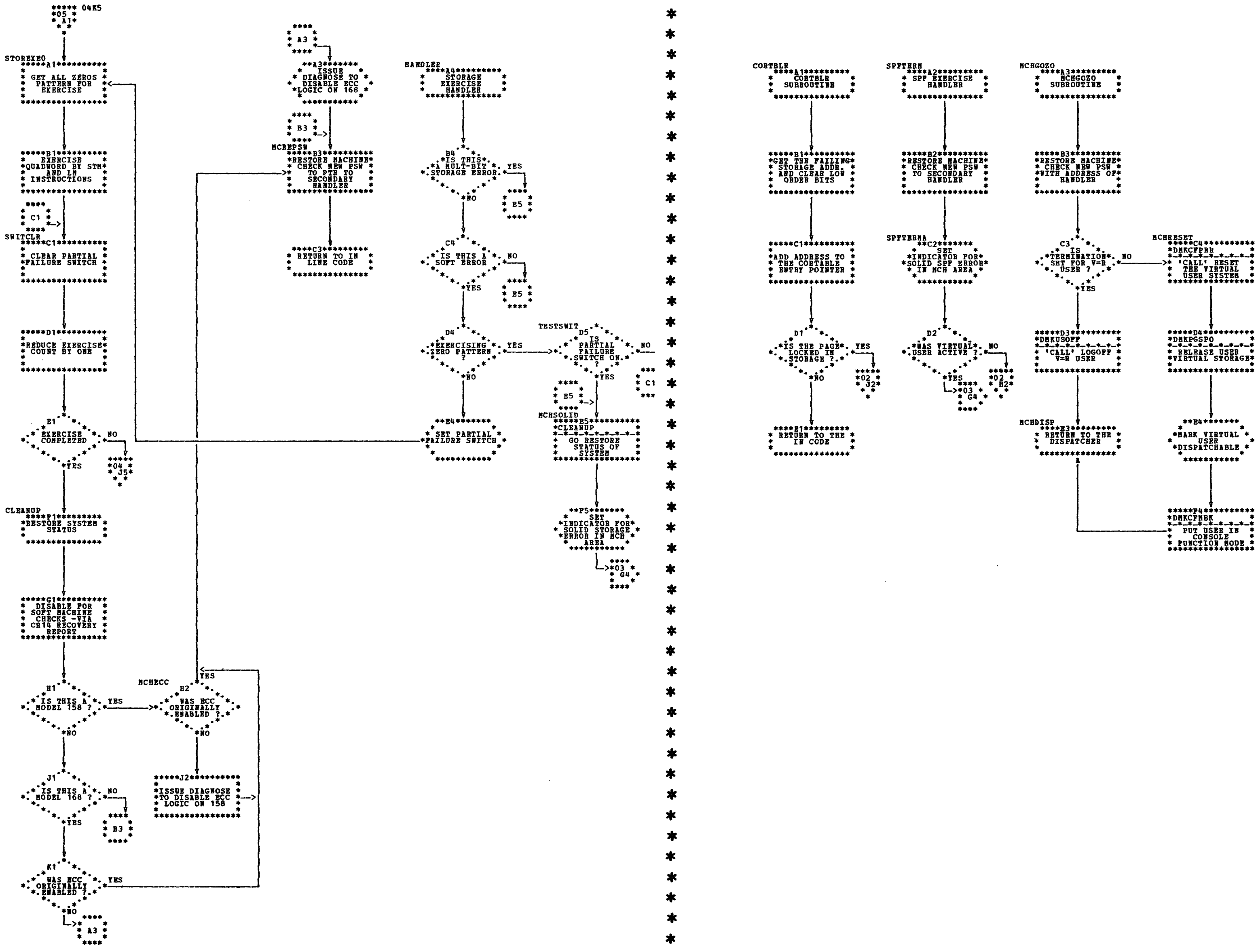

DMKMCCCL
A2
DMKMCCCL
B2
DMKSCNFD
CALL - GET PARAMETER
C2
ANY PARAMETER GIVEN
NO
NOPARM
C3
NO PARAMETER - SET UP MSG DMKMCC026E
G2
YES
D2
MONITOR START
YES
MONSTART
D1
TRACING ALREADY ACTIVE
YES
F2
NO
E1
SET FLAG TO INDICATE TRACING ON
F1
SET NEXT-ENTRY POINTER TO BEGINNING OF TABLE
F4
NO
E2
MONITOR STOP
YES
MONSTOP
E3
TRACING ALREADY OFF
YES
F2
NO
F3
SET FLAG TO INDICATE TRACING OFF
NO
F2
MSG002
F2
INVALID OPERAND SET UP MSG DMKMCC002E
G2
BADARG
G2
SPECIFY MODULE ID FOR MESSAGE
H2
CALL DMKERMSG WHICH RETURNS TO DMKCPM
F4
ACK
F4
DMKQCNWT
CALL - OUTPUT COMMAND COMPLETE MSG
G4
EXIT


DMKMCHIN
A1 DMKMCHIN
B1 CLEAR BAD PARITY IN REGISTERS
C1 SETUP ADDRESSABILITY FOR MCH
D1 SAVE THE FIXED LOGOUT AREA IN HANDLER
E1 SETUP SECONDARY MACHINE CHECK HANDLER
F1 ENABLE FOR HARD MACHINE CHECKS
ENBHARD
G1 PURGE THE TRANSLATE LOOKASIDE BUFFER
H1 CLEAR THE MACHINE CHECK HANDLER'S FLAGS
J1 SET THE CPU TIME AT LAST INTERRUPT
K1 WAS VIRTUAL USER ACTIVE ?
NO
YES
A2
A2 GET ADDRESS OF USER VMBLOK
B2 SAVE THE USER'S REGISTERS AND PSW IN VMBLOK
VMNONDIS
C2 SET VIRTUAL USER NON-DISPATCHABLE
MCHSYSIL
D2 MSG= DMKMCH611I SYSTEM INTEGRITY LOST
E2 IS THE M.C.I.C. VALID ?
NO
YES
F2 ARE THE EMWP,KEY & PSW MK VALID
NO
YES
G2 ARE THE REGISTERS & LOG. STORAGE VALID
YES
NO
H2 WAS THE VIRTUAL USER ACTIVE ?
NO
YES
J2 SET THE TERMINATION FLAG FOR VIRTUAL USER
MCHSYSD
K2 IS SYSTEM DAMAGE INDICATED. ?
YES
NO
A3
A3
A3 MSG= DMKMCH612W TIMING FACILITIES DAMAGE
B3 IS T.O.D. CLOCK DAMAGE INDICATED.
NO
02 A3
YES
C3 INDICATE T.O.D. CLOCK DAMAGE IN MCH AREA
01 D3
02A3
02F2
07G3
07J3
07K3
OPCOM
D3 OPERATOR COMMUNICATION SUBROUTINE
E3 IS THE RECURSION INDICATOR ON ?
YES
NO
F3 INDICATE NO MESSAGE FOR OPERATOR
G3 OPMSG
GO SEND MESSAGE TO OPERATOR
H3 FUMAT
GO FORMAT THE MACHINE CHECK RECORD
MCHVMBOK
J3 GET THE ADDRESS OF THE OPERATOR VMBLOK
K3 IS SYSTEM CONSOLE AVAILABLE.
NO
YES
A4
A4
A4 SETUP THE MESSAGE FOR THE OPERATOR
B4 SEND MESSAGE TO OPERATOR BEFORE PUTTING SYSTEM DOWN
MCHWAIT
C4 MOVE THE ID. IN FOR A CHECKPOINT RESTART
D4 IS T.O.D. CLOCK DAMAGE INDICATED. ?
NO
YES
E4 PUT SYSTEM IN A DISABLE WAIT STATE
MCHRESTA
D5 SAVE THE T.O.D. CLOCK VALUE
E5 DMKDMPRS
'CALL' GO RESTART SYSTEM
02 A3 01B3
MCHTIMER
A3 IS TIMER DAMAGE INDICATED.
YES
01 D3
NO
B3 IS TERMINATION ON FOR VIRTUAL USER
YES
03 A5
NO
MCHSPF
C2 SAVE THE FAILING STORAGE ADDRESS IN MCH AREA
C3 DID INSTR. PROCESSING DAMAGE OCCUR.
YES
NO
D2 IS SPF ERROR INDICATED ?
YES
NO
D3 INDICATE NO MESSAGE FOR OPERATOR
MCHMSG2
E2 MSG=DMKMCH610 SUPERVISOR DAMAGE MESSAGE
E3 SET INDICATOR FOR HARDWARE RECOVERY IN MCH AREA
F2 WAS THE VIRTUAL USER ACTIVE ?
NO
01 D3
YES
F3 IS SYSTEM RECOVERY INDICATED.
NO
03 G1
YES
MCHFSA
G1 IS THE FAILING STORAGE ADDR VALID ?
YES
03 A3
NO
G2 IS SPF OR STORAGE ERROR INDICATED. ?
YES
NO
02 H2
03A3
06D2
MCHCPU
H1 INDICATE FAILING STORAGE ADDR. INVALID
H2 WAS THE VIRTUAL USER ACTIVE ?
NO
YES
02 J2
03E3
04H1
04K1
06D1
MCHSW
J2 SET THE TERMINATION FLAG FOR USER
03 A5
G3 WAS IT A RECOVERED STORAGE AREA
YES
NO
SOFTSTG
G4 IS THIS A MODEL 135 ?
NO
YES
MCHMAIN
G5 SET INDICATORS FOR DATA & MAIN STORAGE
03 C1
H3 SET INDICATOR FOR PROCESSOR ERROR
H4 INDICATE ERROR IS IN CONTROL STORAGE
03 C1
J3 IS THIS A MODEL 145
NO
03 C1
YES
K3 IS THE REGION CODE VALID ?
NO
03 C1
YES
03 A1

| DMKMCH -- Machine Check Handler (Parts 1 and 2 of 9)

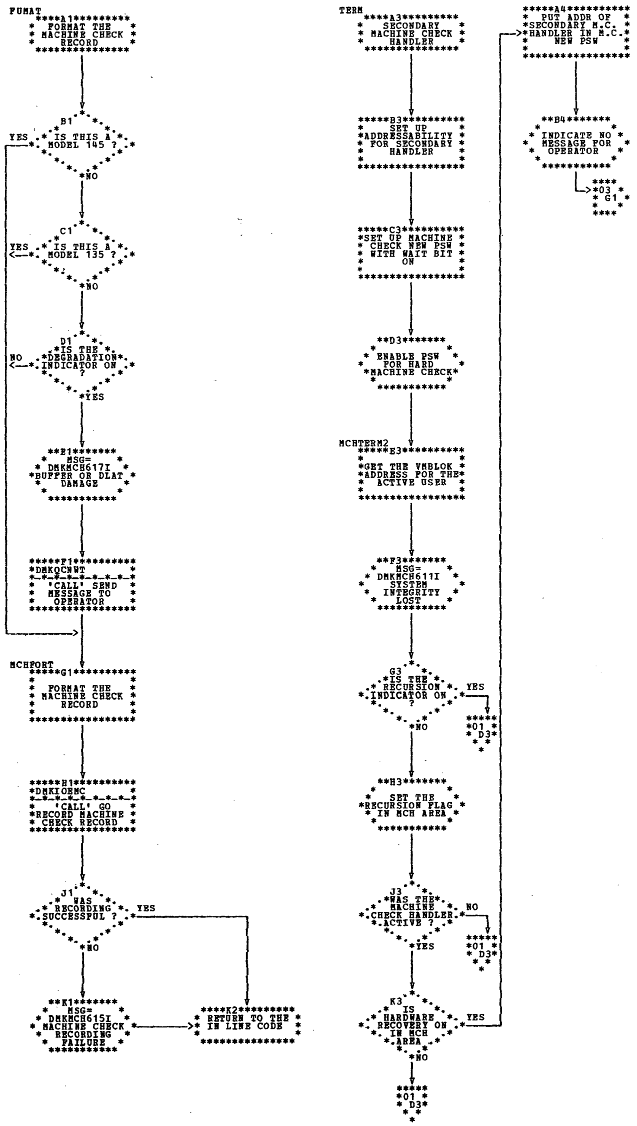
DMKMCH -- Machine Check Handler (Parts 3 and 4 of 9)

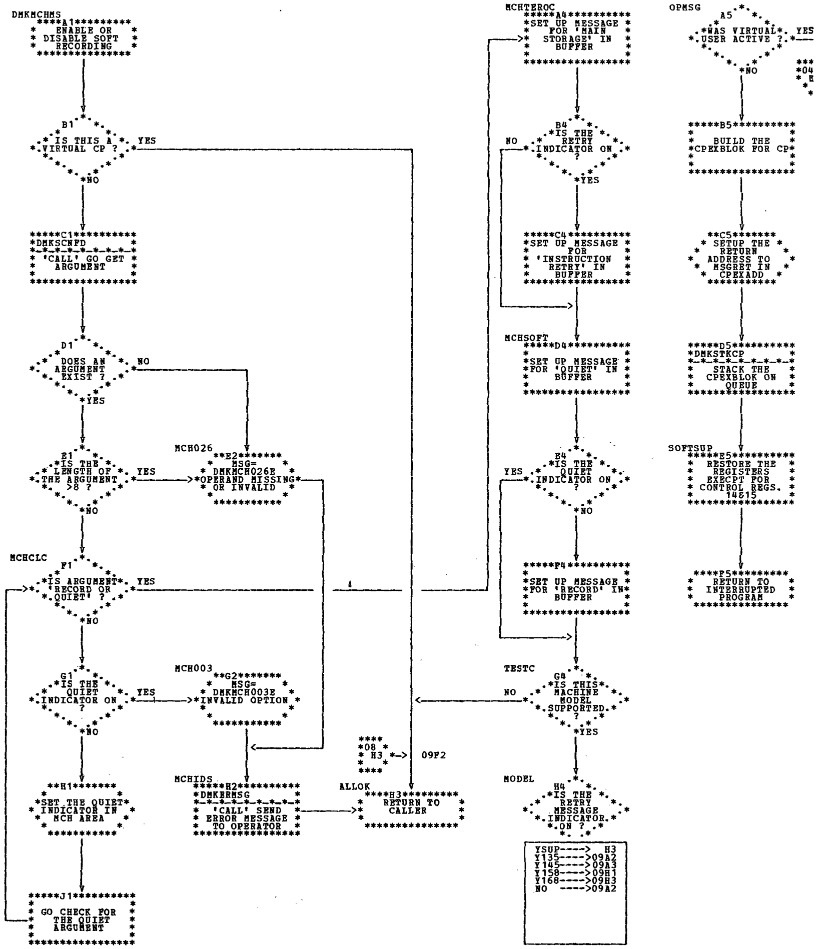

03 A1 02K3
A1 WAS FAILURE IN MAIN STORAGE ?
YES
NO
C1
B1 INDICATE ERROR IS IN CONTROL STORAGE
03 C1 02G5 02H4 02J3 02K3
MCHENABL
C1 HAS SOFT ERRORS EXCEEDED MAX. COUNT
NO
G1
C1
YES
D1 DISABLE SOFT MACHINE CHECKS
E1 INDICATE QUIET STATUS IN MCH AREA
F1 MSG= DMKMCH613I DISABLE SOFT MACHINE CHECKS
03 G1 02F3 04K3 07B4
OPMSG2
G1 OPMSG GO SEND MESSAGE TO OPERATOR
G1
03 H1 04K1
SOFTREC
H1 FUMAT GO FORMAT THE MACHINE CHECK RECORD
NORMSG
J1 WAS VIRTUAL USER ACTIVE ?
NO
YES
K1 SET VIRTUAL USER DISPATCHABLE
USERNOGO
A2 RESTORE MACHINE CHECK NEW PSW FROM SAVE AREA
B2 RESTORE CONTROL REGISTERS 14 & 15 FROM SAVE AREA
C2 RETURN TO DISPATCHER
03 A3 02G1
MCHBACK
A3 IS THE BACKUP FLAG ON ?
NO
02 H2
YES
B3 PUT THE FAILING STORAGE ADDR ON A QUADWORD BOUNDARY
C3 IS STORAGE ERROR INDICATED. ?
NO
04 A2
YES
REFRESH
D3 INDICATE DATA & MAIN STORAGE DAMAGE IN MCH AREA
E3 WAS THE CPU IN RELOCATE MODE
NO
02 J2
YES
F3 PUT THE FAILING STORAGE ADDR. ON A 4K BOUNDARY
G3 GET THE TWO KEYS FROM THE PAGE IN STORAGE
H3 ARE THE CHANGE BITS ON IN KEYS ?
YES
NO
A4
A4
A4 CORTBLR GO GET THE CORTABLE ADDRESSS
B4 GET THE SWAP TABLE ENTRY ADDRESS
C4 ARE THE CHANGE BITS ON IN SWPTABLE
NO
E4
YES
STGTEST
D4 INDICATE CHANGE BITS ARE ON IN KEYS
E4
STGTEST1
E4 MCHSTORA GO TO STORAGE EXERCISE SUBROUTINE
F4 INDICATE INTERMITTENT STORAGE ERROR IN MCH AREA
03 G4 05F5 06D2
VRTSPF
G4 IS THE VIRTUAL= REAL USER ACTIVE
NO
04 A1
YES
H4 SET INDICATOR FOR TERMINATION OF V=R USER
J4 IS THIS A SOLID ERROR ?
NO
YES
K4 REMOVE THE AFFECTED V=R PAGE FROM THE SYSTEM
03 A5 02B3 02J2
VIRTERM
A5 FUMAT GO FORMAT MACHINE CHECK RECORD
NOURMSG
B5 MSG= DMKMCH619I OPERATION TERMINATED MESSAGE
C5 DMKQCNWT 'CALL' SEND MESSAGE TO USER
D5 MSG= DMKMCH616I USER XXXXXXXX TERMINATED
E5 SET UP RETURN ADDRESS FROM WRITER (MCHGOZO)
F5 DMKQCNWT 'CALL' SEND MESSAGE TO OPERATOR
G5 RETURN TO THE DISPATCHER
04 A1 03G4
MCHCORT
A1 CORTBLR GO GET ADDRESS OF CORTABLE AFFECTED
B1 INVALIDATE THE AFFECTED PAGE TABLE ENTRY
C1 INVALIDATE THE SHADOW PAGE TABLE ENTRY
D1 IS THIS A SOLID ERROR ?
NO
YES
E1 DISABLE THE DEFECTIVE FRAME
F1 MSG= DMKMCH614I DAMAGED PAGE NOW DELETED
G1 DMKQCNWT 'CALL' SEND MESSAGE TO OPERATOR
H1 IS THIS A SOLID SPF ERROR ?
YES
02 J2
NO
REFPAG
J1 DMKPTRFT 'CALL' GO PURGE AFFECTED PAGE
MCHCHANG
K1 IS THE CHANGE FLAG ON ?
NO
03 H1
YES
02 J2
04 A2 03C3
SPFTEST
A2 SET INDICATOR FOR SPF IN MCH AREA
B2 GET THE FAILING STORAGE ADDRESS FROM MCH AREA
C2 SET UP MACHINE CHECK NEW PSW WITH ADDR OF SPF HANDLER
D2 PUT THE FAILING STORAGE ADDRESS ON A 2K BOUNDARY
E2 SET UP COUNT FOR NUMBER OF KEYS TO EXERCISE (16)
SPFEXEC
F2 EXERCISE ALL 16 STORAGE PROTECT KEYS
G2 IS EXERCISE COMPLETED
NO
YES
SPFMSG
A3 RESTORE MACHINE CHECK NEW PSW TO SECONDARY HANDLER
B3 SET INDICATOR FOR INTERMITTENT SPF ERROR
C3 SET UP STORAGE KEY ZERO FOR CP
D3 WAS VIRTUAL USER ACTIVE ?
NO
YES
E3 CORTBLR GET CORTABLE ADDRESS FOR AFFECTED PAGE
F3 GET THE AFFECTED STORAGE PROTECT KEY
CPUSPF
G3 PUT FAILING STORAGE ADDRESS ON A 2K BOUNDARY
H3 RESTORE PROPER KEY TO FAILING STORAGE LOCATION
J3 M INDICATE SOFTWARE RECOVERY & REPAIR IN MCH AREA
K3 INDICATE NO MESSAGE FOR OPERATOR
03 G1
MSGRET
A4 MSGRET SUBROUTINE
B4 BUILD A NEW CPEXBLOK AND GET THE LENGTH
C4 DMKFREE RESERVE SPACE FOR THE CPEXBLOK
D4 SAVE THE ADDRESS OF THE NEW CPEXBLOK (DMKMCHBL)
04 E4 08A5
MCHMSGW
E4 IS THERE A MESSAGE ?
NO
YES
F4 DMKQCNWT SEND THE MESSAGE TO THE OPERATOR
G4 RETURN TO IN LINE CODE
MCHSTORA
A5 EXERCISE ROUTINE FOR STORAGE
B5 SAVE THE CONTENT OF CONTROL REGISTER 14
C5 IS THIS A MODEL 158 ?
YES
MCDIABUF
D5 ISSUE DIAGNOSE TO ENABLE ECC LOGIC FOR 158
04 E5 09B1
MCENCR14
E5 SET UP MACHINE CHECK NEW PSW TO POINT TO BLEEDOFF RTN. (MCHSKIP)
F5 ENABLE FOR SOFT MACHINE CHECKS -VIA CR14 RECOVERY REPORT
04 G5 09A1
MCHSKIP
G5 CHANGE MACHINE CHECK NEW PSW TO POINT TO HANDLER
H5 GET EXERCISE COUNT OF X'FF'
04 J5 05E1
STOREXE1
J5 GET ALL ONES PATTERN FOR EXERCISE
K5 EXERCISE QUADWORD BY STM AND LM INSTRUCTIONS
05 A1

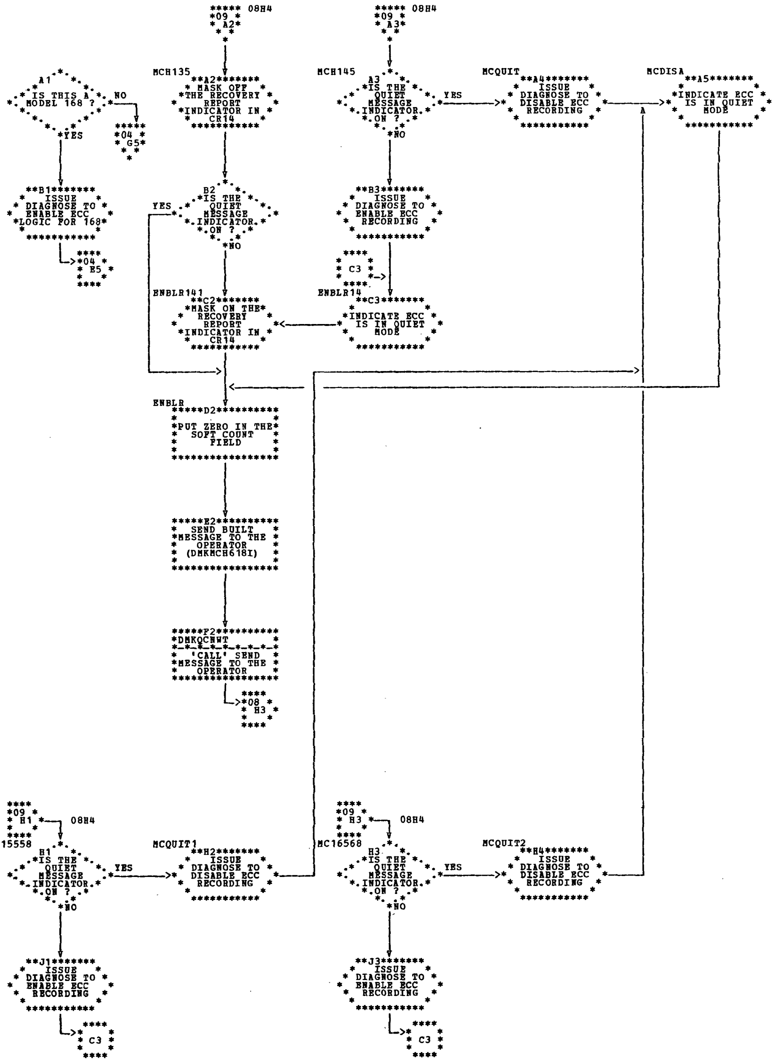

04K5
05 A1
STOREXEO
A1
GET ALL ZEROS PATTERN FOR EXERCISE
B1
EXERCISE QUADWORD BY STM AND LM INSTRUCTIONS
C1
SWITCLR
C1
CLEAR PARTIAL FAILURE SWITCH
D1
REDUCE EXERCISE COUNT BY ONE
E1
EXERCISE COMPLETED
NO
04 J5
YES
CLEANUP
F1
RESTORE SYSTEM STATUS
G1
DISABLE FOR SOFT MACHINE CHECKS -VIA CR14 RECOVERY REPORT
H1
IS THIS A MODEL 158 ?
YES
NO
MCHECC
H2
WAS ECC ORIGINALLY ENABLED ?
YES
NO
J1
IS THIS A MODEL 168 ?
NO
B3
YES
J2
ISSUE DIAGNOSE TO DISABLE ECC LOGIC ON 158
K1
WAS ECC ORIGINALLY ENABLED ?
YES
NO
A3
A3
A3
ISSUE DIAGNOSE TO DISABLE ECC LOGIC ON 168
B3
MCREPSW
B3
RESTORE MACHINE CHECK NEW PSW TO PTR TO SECONDARY HANDLER
C3
RETURN TO IN LINE CODE
HANDLER
A4
STORAGE EXERCISE HANDLER
B4
IS THIS A MULT-BIT STORAGE ERROR
YES
E5
NO
C4
IS THIS A SOFT ERROR
NO
E5
YES
D4
EXERCISING ZERO PATTERN ?
YES
NO
E4
SET PARTIAL FAILURE SWITCH
TESTSWIT
D5
IS PARTIAL FAILURE SWITCH ON. ?
NO
C1
YES
E5
MCHSOLID
E5
CLEANUP
GO RESTORE STATUS OF SYSTEM
F5
SET INDICATOR FOR SOLID STORAGE ERROR IN MCH AREA
03 G4
CORTBLR
A1
CORTBLR SUBROUTINE
B1
GET THE FAILING STORAGE ADDR. AND CLEAR LOW ORDER BITS
C1
ADD ADDRESS TO THE CORTABLE ENTRY POINTER
D1
IS THE PAGE LOCKED IN STORAGE ?
YES
02 J2
NO
E1
RETURN TO THE IN CODE
SPFTERM
A2
SPF EXERCISE HANDLER
B2
RESTORE MACHINE CHECK NEW PSW TO SECONDARY HANDLER
SPFTERMA
C2
SET INDICATOR FOR SOLID SPF ERROR IN MCH AREA
D2
WAS VIRTUAL USER ACTIVE ?
NO
02 H2
YES
03 G4
MCHGOZO
A3
MCHGOZO SUBROUTINE
B3
RESTORE MACHINE CHECK NEW PSW WITH ADDRESS OF HANDLER
C3
IS TERMINATION SET FOR V=R USER ?
NO
YES
MCHRESET
C4
DMKCFPRR
'CALL' RESET THE VIRTUAL USER SYSTEM
D3
DMKUSOFF
'CALL' LOGOFF V=R USER
D4
DMKPGSPO
RELEASE USER VIRTUAL STORAGE
MCHDISP
E3
RETURN TO THE DISPATCHER
E4
MARK VIRTUAL USER DISPATCHABLE
F4
DMKCFMBK
PUT USER IN CONSOLE FUNCTION MODE

| DMKMCH -- Machine Check Handler (Parts 5 and 6 of 9)

| DMKMCH -- Machine Check Handler (Parts 7 and 8 of 9)

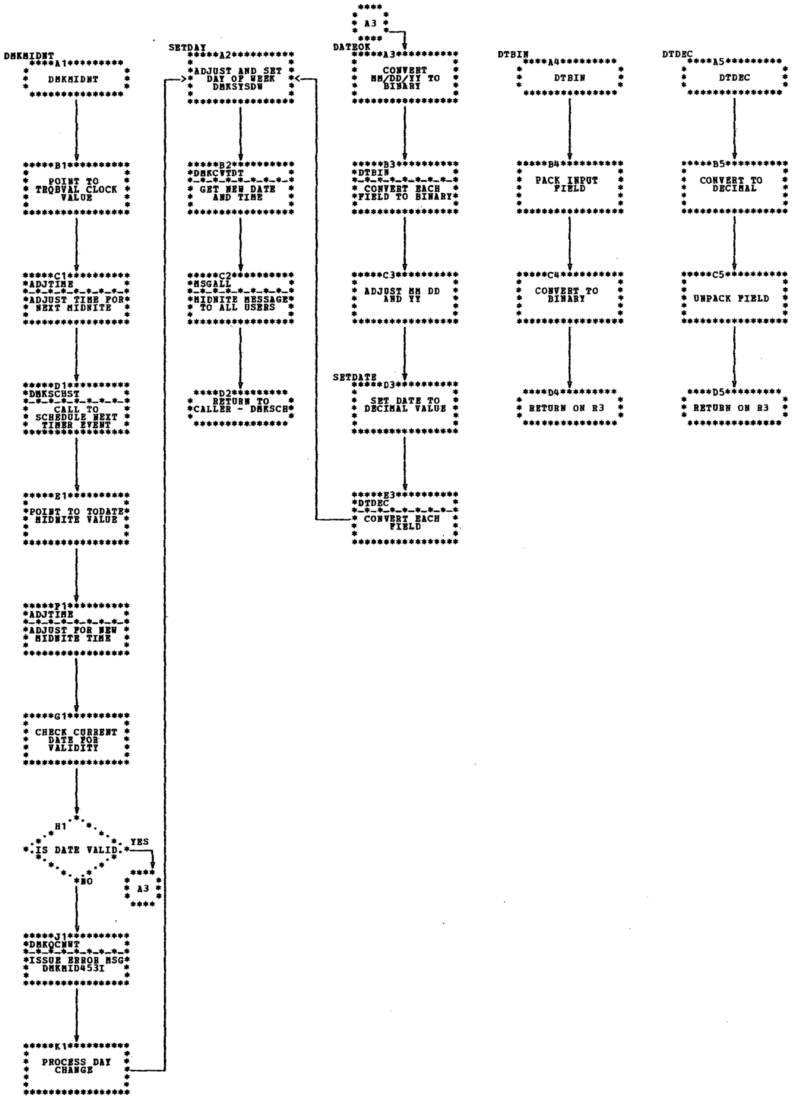

FUMAT
A1
FORMAT THE MACHINE CHECK RECORD
B1
IS THIS A MODEL 145 ?
YES
NO
C1
IS THIS A MODEL 135 ?
YES
NO
D1
IS THE DEGRADATION INDICATOR ON ?
NO
YES
E1
MSG= DMKMCH617I BUFFER OR DLAT DAMAGE
F1
DMKQCNWT
'CALL' SEND MESSAGE TO OPERATOR
MCHFORT
G1
FORMAT THE MACHINE CHECK RECORD
H1
DMKIOEMC
'CALL' GO RECORD MACHINE CHECK RECORD
J1
WAS RECORDING SUCCESSFUL ?
YES
NO
K1
MSG= DMKMCH615I MACHINE CHECK RECORDING FAILURE
K2
RETURN TO THE IN LINE CODE
TERM
A3
SECONDARY MACHINE CHECK HANDLER
B3
SET UP ADDRESSABILITY FOR SECONDARY HANDLER
C3
SET UP MACHINE CHECK NEW PSW WITH WAIT BIT ON
D3
ENABLE PSW FOR HARD MACHINE CHECK
MCHTERM2
E3
GET THE VMBLOK ADDRESS FOR THE ACTIVE USER
F3
MSG= DMKMCH611I SYSTEM INTEGRITY LOST
G3
IS THE RECURSION INDICATOR ON ?
YES
01 D3
NO
H3
SET THE RECURSION FLAG IN MCH AREA
J3
WAS THE MACHINE CHECK HANDLER ACTIVE ?
NO
01 D3
YES
K3
IS HARDWARE RECOVERY ON IN MCH AREA
YES
NO
01 D3
A4
PUT ADDR OF SECONDARY M.C. HANDLER IN M.C. NEW PSW
B4
INDICATE NO MESSAGE FOR OPERATOR
03 G1

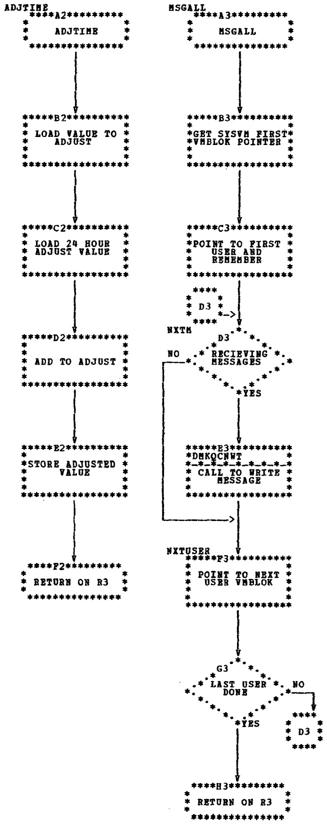

DMKMCHMS
A1
ENABLE OR DISABLE SOFT RECORDING
B1
IS THIS A VIRTUAL CP ?
YES
NO
C1
DMKSCNFD
'CALL' GO GET ARGUMENT
D1
DOES AN ARGUMENT EXIST ?
NO
YES
E1
IS THE LENGTH OF THE ARGUMENT >8 ?
YES
NO
MCH026
E2
MSG= DMKMCH026E OPERAND MISSING OR INVALID
MCHCLC
F1
IS ARGUMENT 'RECORD OR QUIET' ?
YES
NO
G1
IS THE QUIET INDICATOR ON ?
YES
NO
MCH003
G2
MSG= DMKMCH003E INVALID OPTION
H1
SET THE QUIET INDICATOR IN MCH AREA
J1
GO CHECK FOR THE QUIET ARGUMENT
MCHIDS
H2
DMKERMSG
'CALL' SEND ERROR MESSAGE TO OPERATOR
08 H3
09F2
ALLOK
H3
RETURN TO CALLER
MCHTEROC
A4
SET UP MESSAGE FOR 'MAIN STORAGE' IN BUFFER
B4
IS THE RETRY INDICATOR ON ?
NO
YES
C4
SET UP MESSAGE FOR 'INSTRUCTION RETRY' IN BUFFER
MCHSOFT
D4
SET UP MESSAGE FOR 'QUIET' IN BUFFER
E4
IS THE QUIET INDICATOR ON ?
YES
NO
F4
SET UP MESSAGE FOR 'RECORD' IN BUFFER
TESTC
G4
IS THIS MACHINE MODEL SUPPORTED ?
NO
YES
MODEL
H4
IS THE RETRY MESSAGE INDICATOR ON ?
YSUP----> H3
Y135---->09A2
Y145---->09A3
Y158---->09H1
Y168---->09H3
NO ---->09A2
OPMSG
A5
WAS VIRTUAL USER ACTIVE ?
YES
NO
04 E
B5
BUILD THE CPEXBLOK FOR CP
C5
SETUP THE RETURN ADDRESS TO MSGRET IN CPEXADD
D5
DMKSTKCP
STACK THE CPEXBLOK ON QUEUE
SOFTSUP
E5
RESTORE THE REGISTERS EXECPT FOR CONTROL REGS. 14&15
F5
RETURN TO INTERRUPTED PROGRAM

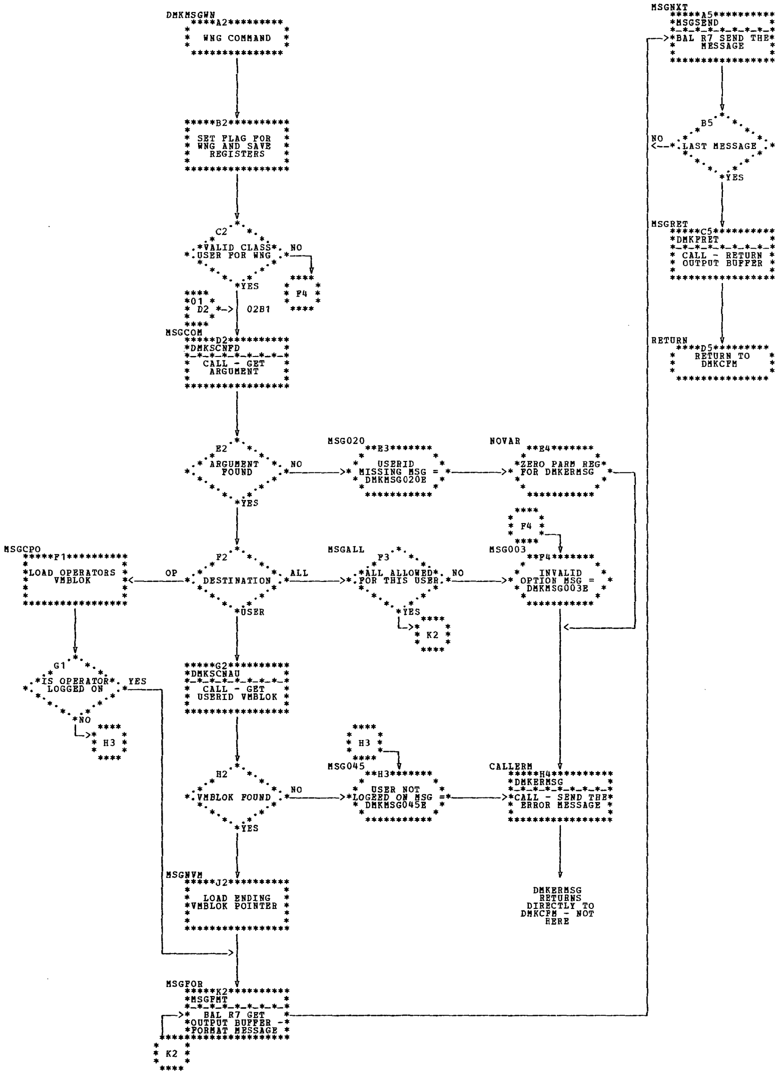

08H4
09
A2
08H4
09
A3
A1
IS THIS A MODEL 168 ?
NO
YES
04
G5
MCH135
A2
MASK OFF THE RECOVERY REPORT INDICATOR IN CR14
MCH145
A3
IS THE QUIET MESSAGE INDICATOR ON ?
YES
NO
MCQUIT
A4
ISSUE DIAGNOSE TO DISABLE ECC RECORDING
MCDISA
A5
INDICATE ECC IS IN QUIET MODE
A
B1
ISSUE DIAGNOSE TO ENABLE ECC LOGIC FOR 168
04
E5
B2
IS THE QUIET MESSAGE INDICATOR ON ?
YES
NO
B3
ISSUE DIAGNOSE TO ENABLE ECC RECORDING
C3
ENBLR141
C2
MASK ON THE RECOVERY REPORT INDICATOR IN CR14
ENBLR14
C3
INDICATE ECC IS IN QUIET MODE
ENBLR
D2
PUT ZERO IN THE SOFT COUNT FIELD
E2
SEND BUILT MESSAGE TO THE OPERATOR (DMKMCH618I)
F2
DMKQCNWT
'CALL' SEND MESSAGE TO THE OPERATOR
08
H3
09
H1
08H4
MC15558
H1
IS THE QUIET MESSAGE INDICATOR ON ?
YES
NO
MCQUIT1
H2
ISSUE DIAGNOSE TO DISABLE ECC RECORDING
09
H3
08H4
MC16568
H3
IS THE QUIET MESSAGE INDICATOR ON ?
YES
NO
MCQUIT2
H4
ISSUE DIAGNOSE TO DISABLE ECC RECORDING
J1
ISSUE DIAGNOSE TO ENABLE ECC RECORDING
C3
J3
ISSUE DIAGNOSE TO ENABLE ECC RECORDING
C3

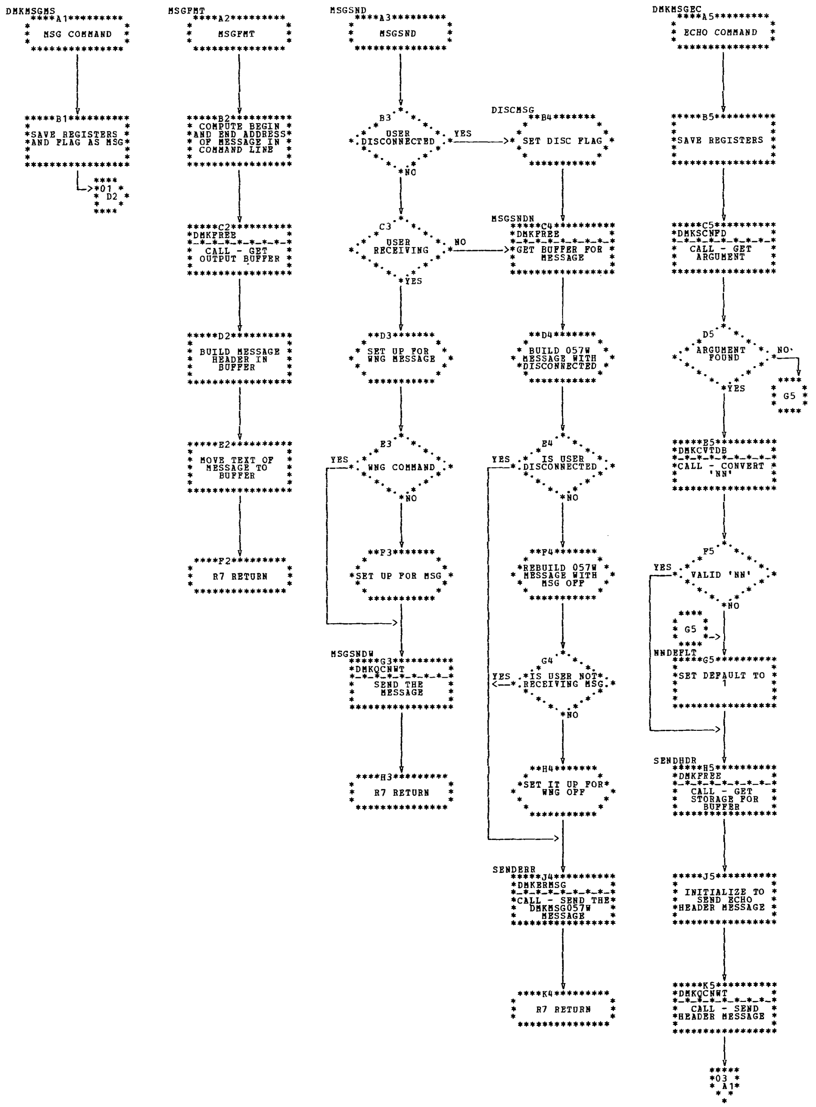
| DMKMCH -- Machine Check Handler (Part 9 of 9)

DMKMID -- Change Date at Midnight (Parts 1 and 2 of 2)


DMKMIDNT
A1
DMKMIDNT
B1
POINT TO TRQBVAL CLOCK VALUE
C1
ADJTIME
ADJUST TIME FOR NEXT MIDNITE
D1
DMKSCHST
CALL TO SCHEDULE NEXT TIMER EVENT
E1
POINT TO TODATE MIDNITE VALUE
F1
ADJTIME
ADJUST FOR NEW MIDNITE TIME
G1
CHECK CURRENT DATE FOR VALIDITY
H1
IS DATE VALID
YES
A3
NO
J1
DMKQCNWT
ISSUE ERROR MSG DMKMID453I
K1
PROCESS DAY CHANGE
SETDAY
A2
ADJUST AND SET DAY OF WEEK DMKSYSDW
B2
DMKCVTDT
GET NEW DATE AND TIME
C2
MSGALL
MIDNITE MESSAGE TO ALL USERS
D2
RETURN TO CALLER - DMKSCH
A3
DATEOK
A3
CONVERT MM/DD/YY TO BINARY
B3
DTBIN
CONVERT EACH FIELD TO BINARY
C3
ADJUST MM DD AND YY
SETDATE
D3
SET DATE TO DECIMAL VALUE
E3
DTDEC
CONVERT EACH FIELD
DTBIN
A4
DTBIN
B4
PACK INPUT FIELD
C4
CONVERT TO BINARY
D4
RETURN ON R3
DTDEC
A5
DTDEC
B5
CONVERT TO DECIMAL
C5
UNPACK FIELD
D5
RETURN ON R3

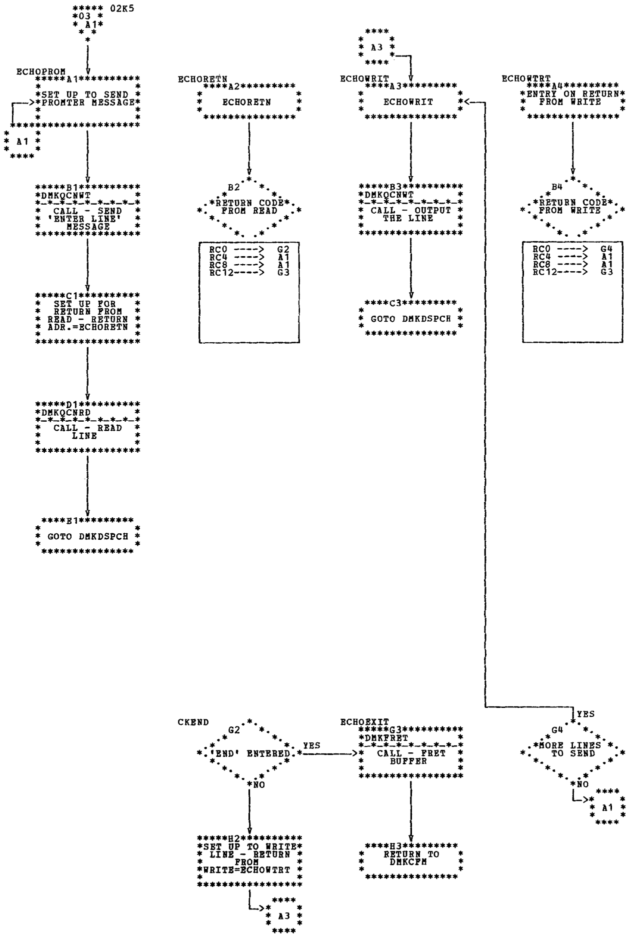


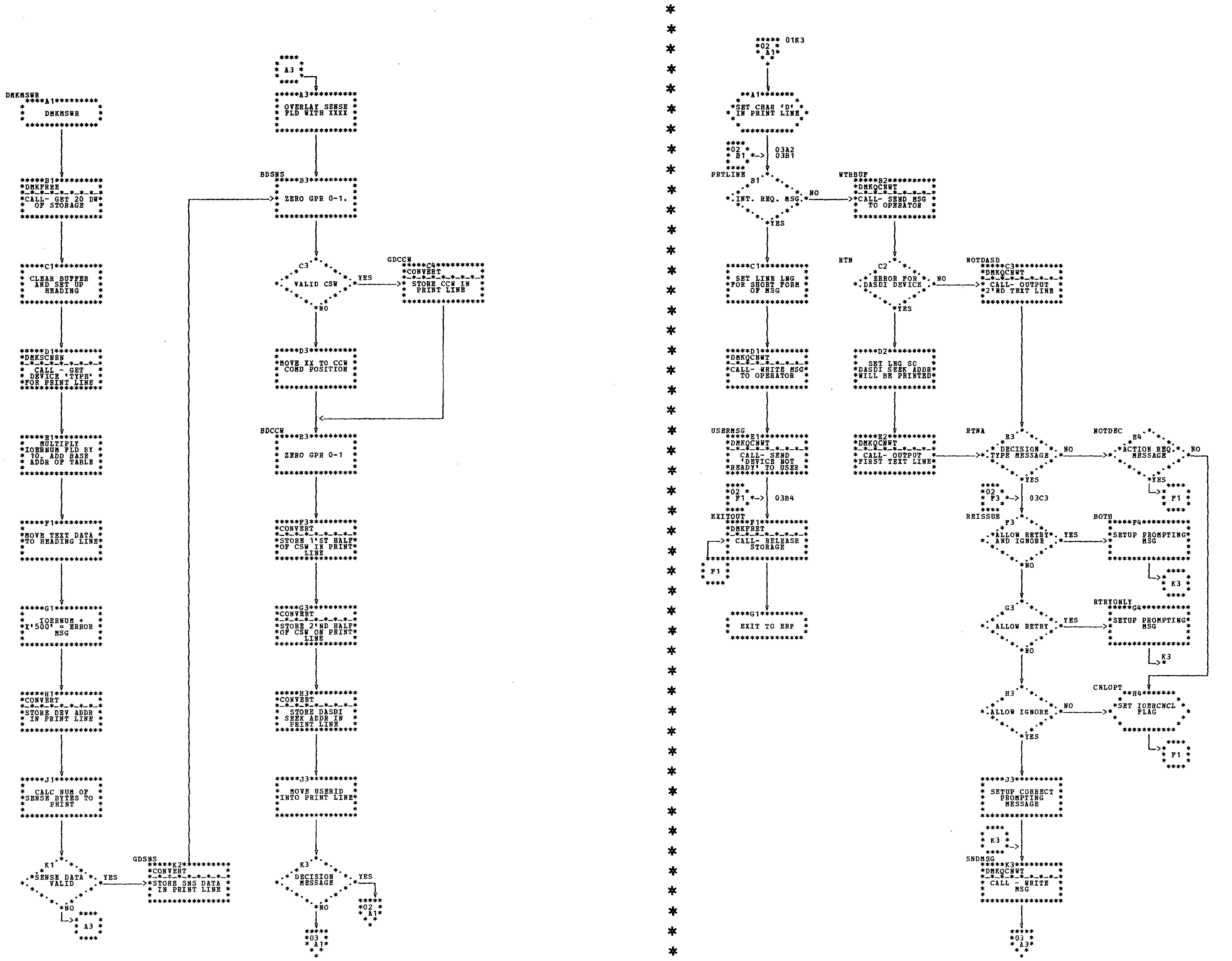




DMKMSG -- Process ECHO, MSG, and WNG Commands (Parts 1 and 2 of 3)

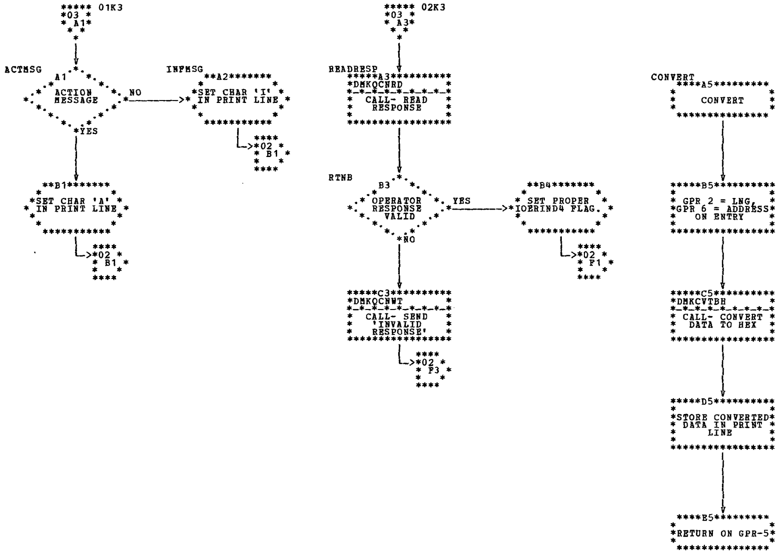
DMKMSG -- Process ECHO, MSG, and WNG Commands (Part 3 of 3)

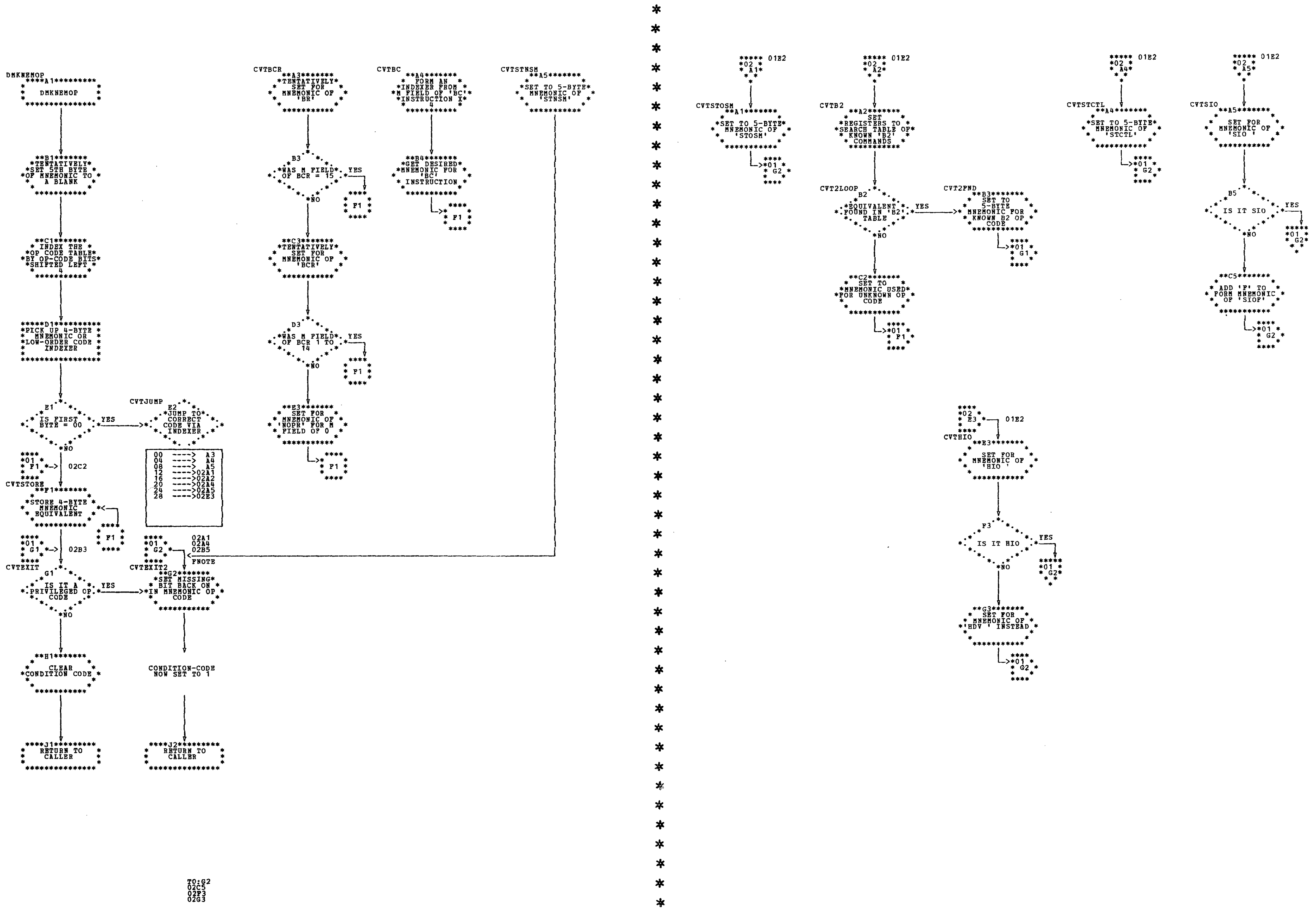



DMKMSWR
A1
DMKMSWR
B1
DMKFREE
CALL- GET 20 DW OF STORAGE
C1
CLEAR BUFFER AND SET UP HEADING
D1
DMKSCNRN
CALL - GET DEVICE 'TYPE' FOR PRINT LINE
E1
MULTIPLY IOERNUM FLD BY 10. ADD BASE ADDR OF TABLE
F1
MOVE TEXT DATA TO HEADING LINE
G1
IOERNUM + X'500' = ERROR MSG
H1
CONVERT
STORE DEV ADDR IN PRINT LINE
J1
CALC NUM OF SENSE BYTES TO PRINT
K1
SENSE DATA VALID
YES
NO
A3
GDSNS
K2
CONVERT
STORE SNS DATA IN PRINT LINE
A3
A3
OVERLAY SENSE FLD WITH XXXX
BDSNS
B3
ZERO GPR 0-1.
C3
VALID CSW
YES
NO
GDCCW
C4
CONVERT
STORE CCW IN PRINT LINE
D3
MOVE XX TO CCW COMD POSITION
BDCCW
E3
ZERO GPR 0-1
F3
CONVERT
STORE 1'ST HALF OF CSW IN PRINT LINE
G3
CONVERT
STORE 2'ND HALF OF CSW ON PRINT LINE
H3
CONVERT
STORE DASDI SEEK ADDR IN PRINT LINE
J3
MOVE USERID INTO PRINT LINE
K3
DECISION MESSAGE
YES
NO
02 A1
03 A1
02 A1
01K3
A1
SET CHAR 'D' IN PRINT LINE
02 B1
03A2
03B1
PRTLINE
B1
INT. REQ. MSG.
NO
YES
WTRBUF
B2
DMKQCNWT
CALL- SEND MSG TO OPERATOR
C1
SET LINE LNG FOR SHORT FORM OF MSG
RTN
C2
ERROR FOR DASDI DEVICE
NO
YES
NOTDASD
C3
DMKQCNWT
CALL- OUTPUT 2'ND TEXT LINE
D1
DMKQCNWT
CALL- WRITE MSG TO OPERATOR
D2
SET LNG SO DASDI SEEK ADDR WILL BE PRINTED
USERMSG
E1
DMKQCNWT
CALL- SEND 'DEVICE NOT READY' TO USER
E2
DMKQCNWT
CALL- OUTPUT FIRST TEXT LINE
RTNA
E3
DECISION TYPE MESSAGE
NO
YES
NOTDEC
E4
ACTION REQ. MESSAGE
NO
YES
F1
02 F1
03B4
EXITOUT
F1
DMKFRET
CALL- RELEASE STORAGE
F1
02 F3
03C3
REISSUE
F3
ALLOW RETRY AND IGNORE
YES
NO
BOTH
F4
SETUP PROMPTING MSG
K3
G1
EXIT TO ERP
G3
ALLOW RETRY
YES
NO
RTRYONLY
G4
SETUP PROMPTING MSG
K3
H3
ALLOW IGNORE
NO
YES
CNLOPT
H4
SET IOERCNCL FLAG
F1
J3
SETUP CORRECT PROMPTING MESSAGE
K3
SNDMSG
K3
DMKQCNWT
CALL - WRITE MSG
03 A3

DMKMSW -- ERP Message Writer (Parts 1 and 2 of 3)

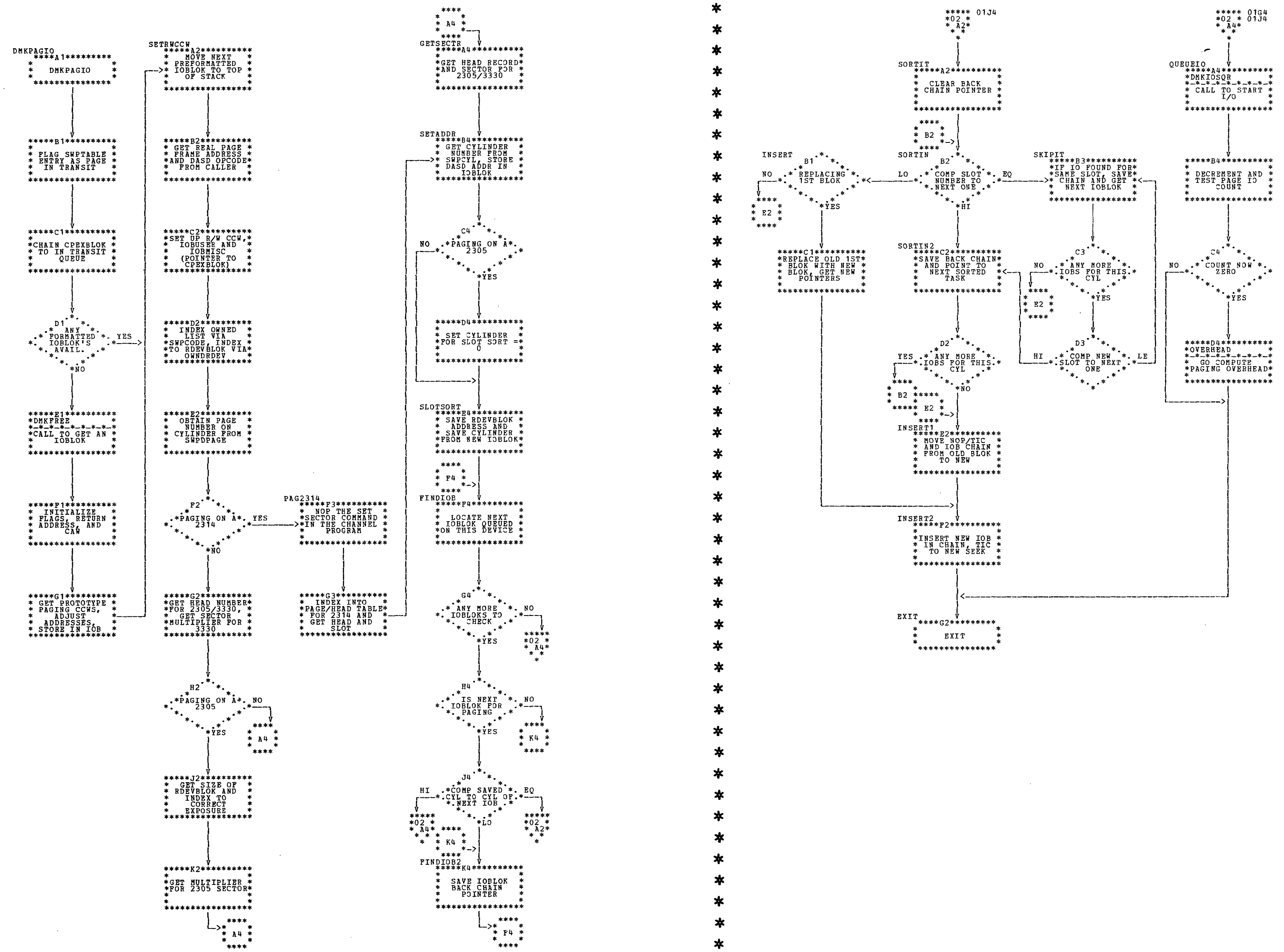
DMKMSW -- ERP Message Writer (Part 3 of 3)

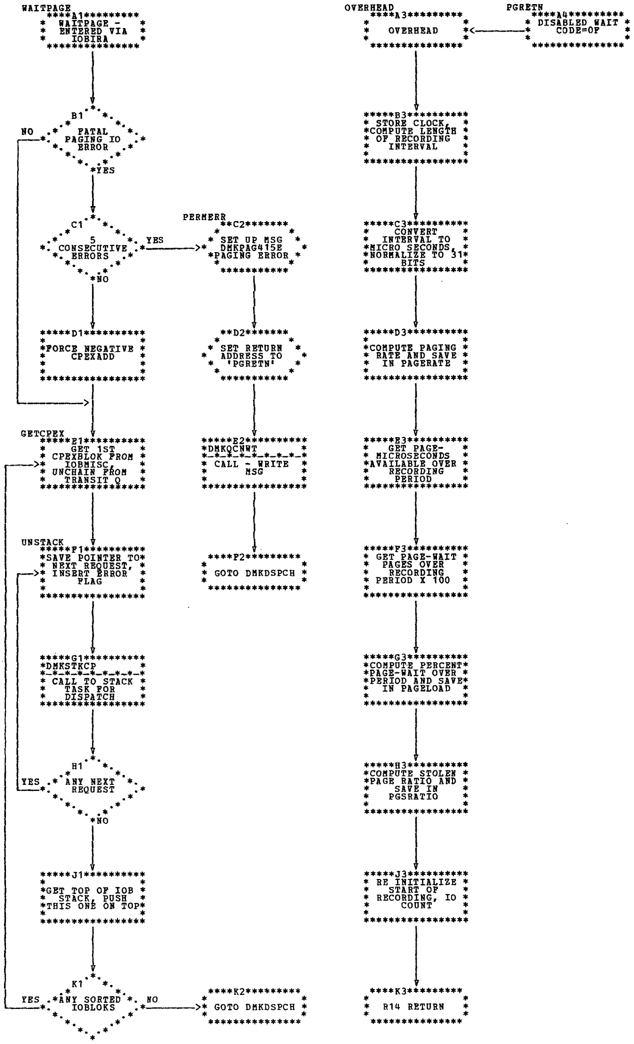

01K3
03
A1
ACTMSG
A1
ACTION MESSAGE
NO
YES
INFMSG
A2
SET CHAR 'I' IN PRINT LINE
02
B1
B1
SET CHAR 'A' IN PRINT LINE
02
B1
02K3
03
A3
READRESP
A3
DMKQCNRD
CALL- READ RESPONSE
RTNB
B3
OPERATOR RESPONSE VALID
YES
NO
B4
SET PROPER IOERIND4 FLAG.
02
F1
C3
DMKQCNWT
CALL- SEND 'INVALID RESPONSE'
02
F3
CONVERT
A5
CONVERT
B5
GPR 2 = LNG, GPR 6 = ADDRESS ON ENTRY
C5
DMKCVTBH
CALL- CONVERT DATA TO HEX
D5
STORE CONVERTED DATA IN PRINT LINE
E5
RETURN ON GPR-5


DMKNEMOP
A1
DMKNEMOP
B1
TENTATIVELY SET 5TH BYTE OF MNEMONIC TO A BLANK
C1
INDEX THE OP CODE TABLE BY OP-CODE BITS SHIFTED LEFT 4
D1
PICK UP 4-BYTE MNEMONIC OR LOW-ORDER CODE INDEXER
E1
IS FIRST BYTE = 00
YES
NO
CVTJUMP
E2
JUMP TO CORRECT CODE VIA INDEXER
00 ---> A3
04 ---> A4
08 ---> A5
12 --->02A1
16 --->02A2
20 --->02A4
24 --->02A5
28 --->02E3
01 F1
02C2
CVTSTORE
F1
STORE 4-BYTE MNEMONIC EQUIVALENT
F1
01 G1
02B3
CVTEXIT
G1
IS IT A PRIVILEGED OP CODE
YES
NO
01 G2
02A1
02A4
02B5
FNOTE
CVTEXIT2
G2
SET MISSING BIT BACK ON IN MNEMONIC OP CODE
H1
CLEAR CONDITION CODE
CONDITION-CODE NOW SET TO 1
J1
RETURN TO CALLER
J2
RETURN TO CALLER
TO:G2
02C5
02F3
02G3
CVTBCR
A3
TENTATIVELY SET FOR MNEMONIC OF 'BR'
B3
WAS M FIELD OF BCR = 15
YES
F1
NO
C3
TENTATIVELY SET FOR MNEMONIC OF 'BCR'
D3
WAS M FIELD OF BCR 1 TO 14
YES
F1
NO
E3
SET FOR MNEMONIC OF 'NOPR' FOR M FIELD OF 0
F1
CVTBC
A4
FORM AN INDEXER FROM M FIELD OF 'BC' INSTRUCTION X 4
B4
GET DESIRED MNEMONIC FOR 'BC' INSTRUCTION
F1
CVTSTNSM
A5
SET TO 5-BYTE MNEMONIC OF 'STNSM'
02 A1
01E2
CVTSTOSM
A1
SET TO 5-BYTE MNEMONIC OF 'STOSM'
01 G2
02 A2
01E2
CVTB2
A2
SET REGISTERS TO SEARCH TABLE OF KNOWN 'B2' COMMANDS
CVT2LOOP
B2
EQUIVALENT FOUND IN 'B2' TABLE
YES
NO
CVT2FND
B3
SET TO 5-BYTE MNEMONIC FOR KNOWN B2 OP CODE
01 G1
C2
SET TO MNEMONIC USED FOR UNKNOWN OP CODE
01 F1
02 A4
01E2
CVTSTCTL
A4
SET TO 5-BYTE MNEMONIC OF 'STCTL'
01 G2
02 A5
01E2
CVTSIO
A5
SET FOR MNEMONIC OF 'SIO '
B5
IS IT SIO
YES
01 G2
NO
C5
ADD 'F' TO FORM MNEMONIC OF 'SIOF'
01 G2
02 E3
01E2
CVTHIO
E3
SET FOR MNEMONIC OF 'HIO'
F3
IS IT HIO
YES
01 G2
NO
G3
SET FOR MNEMONIC OF 'HDV ' INSTEAD
01 G2

| DMKNEM -- Translate Operation Code (Parts 1 and 2 of 2)

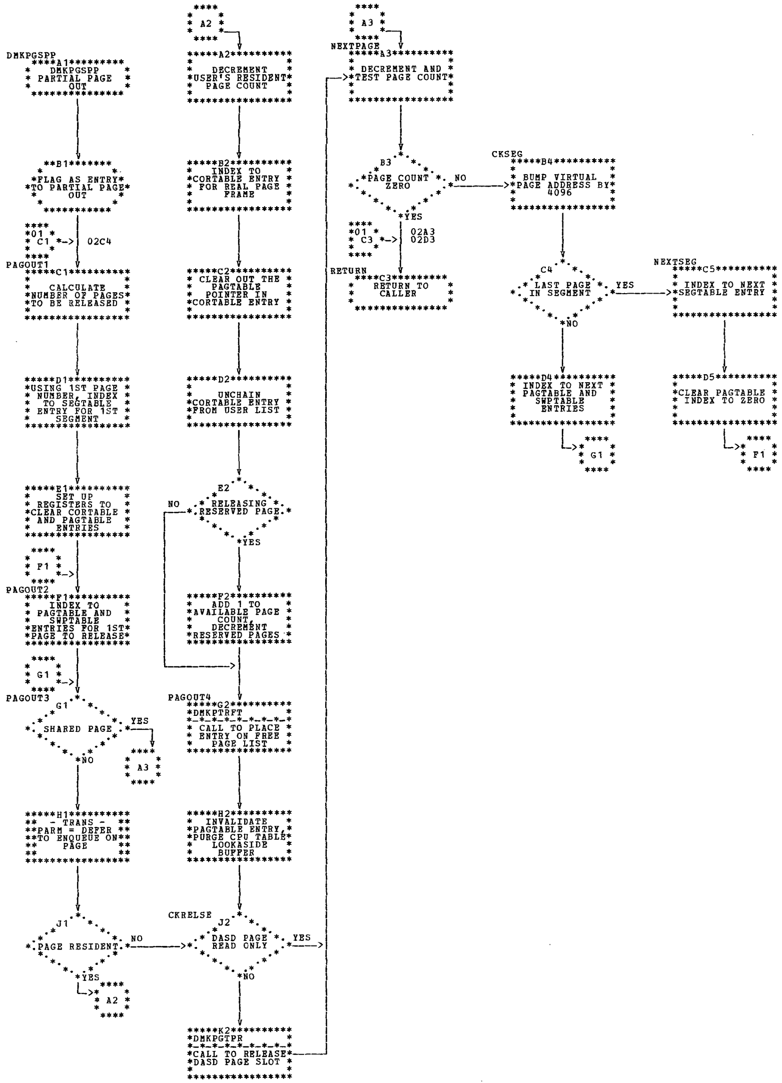
| DMKPAG -- Paging I/O Scheduler (Parts 1 and 2 of 3)

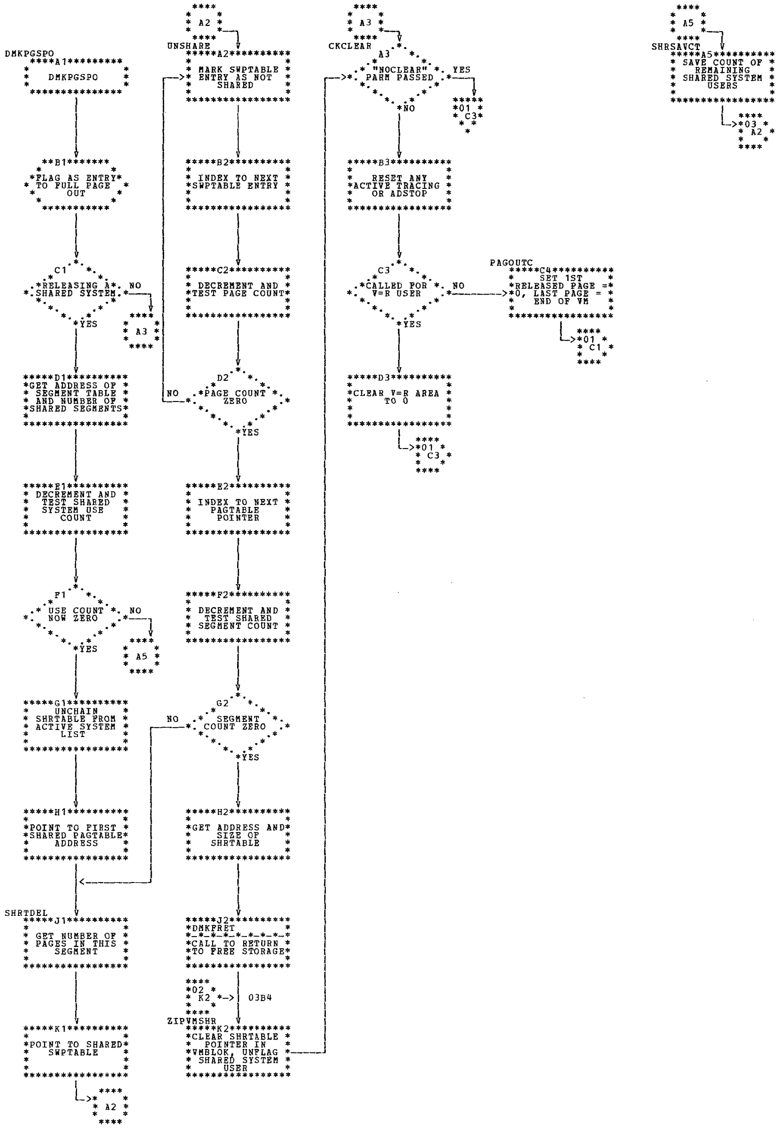

DMKPAGIO
A1 DMKPAGIO
B1 FLAG SWPTABLE ENTRY AS PAGE IN TRANSIT
C1 CHAIN CPEXBLOK TO IN TRANSIT QUEUE
D1 ANY FORMATTED IOBLOK'S AVAIL.
YES
NO
E1 DMKFREE CALL TO GET AN IOBLOK
F1 INITIALIZE FLAGS, RETURN ADDRESS, AND CAW
G1 GET PROTOTYPE PAGING CCWS, ADJUST ADDRESSES, STORE IN IOB
SETRWCCW
A2 MOVE NEXT PREFORMATTED IOBLOK TO TOP OF STACK
B2 GET REAL PAGE FRAME ADDRESS AND DASD OPCODE FROM CALLER
C2 SET UP R/W CCW, IOBUSER AND IOBMISC (POINTER TO CPEXBLOK)
D2 INDEX OWNED LIST VIA SWPCODE, INDEX TO RDEVBLOK VIA OWNDRDEV
E2 OBTAIN PAGE NUMBER ON CYLINDER FROM SWPDPAGE
F2 PAGING ON A 2314
YES
NO
G2 GET HEAD NUMBER FOR 2305/3330, GET SECTOR MULTIPLIER FOR 3330
H2 PAGING ON A 2305
NO
A4
YES
J2 GET SIZE OF RDEVBLOK AND INDEX TO CORRECT EXPOSURE
K2 GET MULTIPLIER FOR 2305 SECTOR
A4
PAG2314
F3 NOP THE SET SECTOR COMMAND IN THE CHANNEL PROGRAM
G3 INDEX INTO PAGE/HEAD TABLE FOR 2314 AND GET HEAD AND SLOT
A4
GETSECTR
A4 GET HEAD RECORD AND SECTOR FOR 2305/3330
SETADDR
B4 GET CYLINDER NUMBER FROM SWPCYL, STORE DASD ADDR IN IOBLOK
C4 PAGING ON A 2305
NO
YES
D4 SET CYLINDER FOR SLOT SORT = 0
SLOTSORT
E4 SAVE RDEVBLOK ADDRESS AND SAVE CYLINDER FROM NEW IOBLOK
F4
FINDIOB
F4 LOCATE NEXT IOBLOK QUEUED ON THIS DEVICE
G4 ANY MORE IOBLOKS TO CHECK
NO
02 A4
YES
H4 IS NEXT IOBLOK FOR PAGING
NO
K4
YES
J4 COMP SAVED CYL TO CYL OF NEXT IOB
HI
EQ
02 A4
LO
02 A2
K4
FINDIOB2
K4 SAVE IOBLOK BACK CHAIN POINTER
F4
02 A2 01J4
SORTIT
A2 CLEAR BACK CHAIN POINTER
B2
SORTIN
B2 COMP SLOT NUMBER TO NEXT ONE
LO
EQ
HI
INSERT
B1 REPLACING 1ST BLOK
NO
E2
YES
C1 REPLACE OLD 1ST BLOK WITH NEW BLOK, GET NEW POINTERS
SORTIN2
C2 SAVE BACK CHAIN AND POINT TO NEXT SORTED TASK
D2 ANY MORE IOBS FOR THIS CYL
YES
B2
NO
E2
INSERT1
E2 MOVE NOP/TIC AND IOB CHAIN FROM OLD BLOK TO NEW
INSERT2
F2 INSERT NEW IOB IN CHAIN, TIC TO NEW SEEK
EXIT
G2 EXIT
SKIPIT
B3 IF IO FOUND FOR SAME SLOT, SAVE CHAIN AND GET NEXT IOBLOK
C3 ANY MORE IOBS FOR THIS CYL
NO
E2
YES
D3 COMP NEW SLOT TO NEXT ONE
HI
LE
02 A4 01G4 01J4
QUEUEIO
A4 DMKIOSQR CALL TO START I/O
B4 DECREMENT AND TEST PAGE IO COUNT
C4 COUNT NOW ZERO
NO
YES
D4 OVERHEAD GO COMPUTE PAGING OVERHEAD

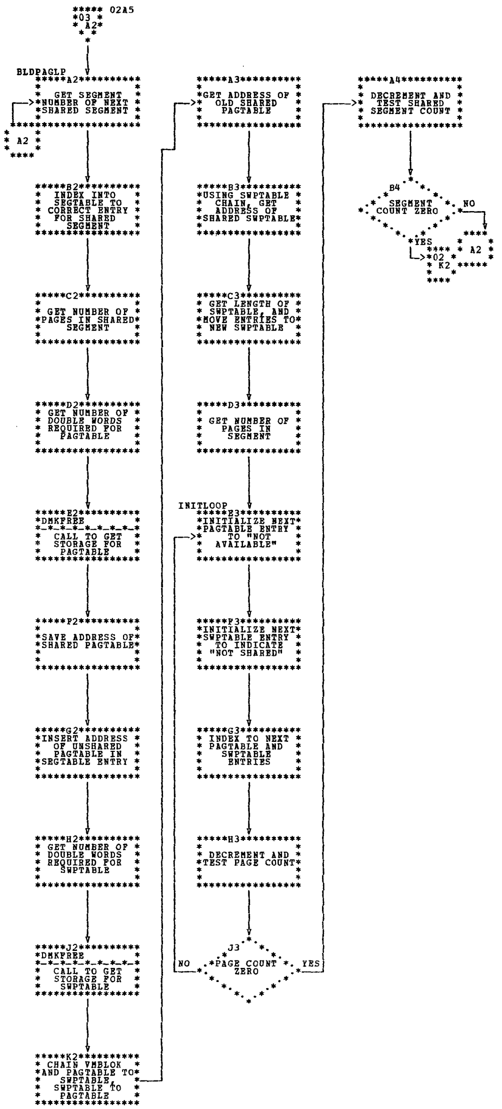

WAITPAGE
A1
WAITPAGE - ENTERED VIA IOBIRA
B1
FATAL PAGING IO ERROR
NO
YES
C1
5 CONSECUTIVE ERRORS
YES
NO
PERMERR
C2
SET UP MSG DMKPAG415E PAGING ERROR
D1
FORCE NEGATIVE CPEXADD
D2
SET RETURN ADDRESS TO 'PGRETN'
GETCPEX
E1
GET 1ST CPEXBLOK FROM IOBMISC, UNCHAIN FROM TRANSIT Q
E2
DMKQCNWT
CALL - WRITE MSG
UNSTACK
F1
SAVE POINTER TO NEXT REQUEST, INSERT ERROR FLAG
F2
GOTO DMKDSPCH
G1
DMKSTKCP
CALL TO STACK TASK FOR DISPATCH
H1
ANY NEXT REQUEST
YES
NO
J1
GET TOP OF IOB STACK, PUSH THIS ONE ON TOP
K1
ANY SORTED IOBLOKS
YES
NO
K2
GOTO DMKDSPCH
OVERHEAD
A3
OVERHEAD
PGRETN
A4
DISABLED WAIT CODE=0F
B3
STORE CLOCK, COMPUTE LENGTH OF RECORDING INTERVAL
C3
CONVERT INTERVAL TO MICRO SECONDS, NORMALIZE TO 31 BITS
D3
COMPUTE PAGING RATE AND SAVE IN PAGERATE
E3
GET PAGE-MICROSECONDS AVAILABLE OVER RECORDING PERIOD
F3
GET PAGE-WAIT PAGES OVER RECORDING PERIOD X 100
G3
COMPUTE PERCENT PAGE-WAIT OVER PERIOD AND SAVE IN PAGELOAD
H3
COMPUTE STOLEN PAGE RATIO AND SAVE IN PGSRATIO
J3
RE INITIALIZE START OF RECORDING, IO COUNT
K3
R14 RETURN

| DMKPAG -- Paging I/O Scheduler (Part 3 of 3)

DMKPGS -- Release Virtual Storage (Parts 1 and 2 of 3)

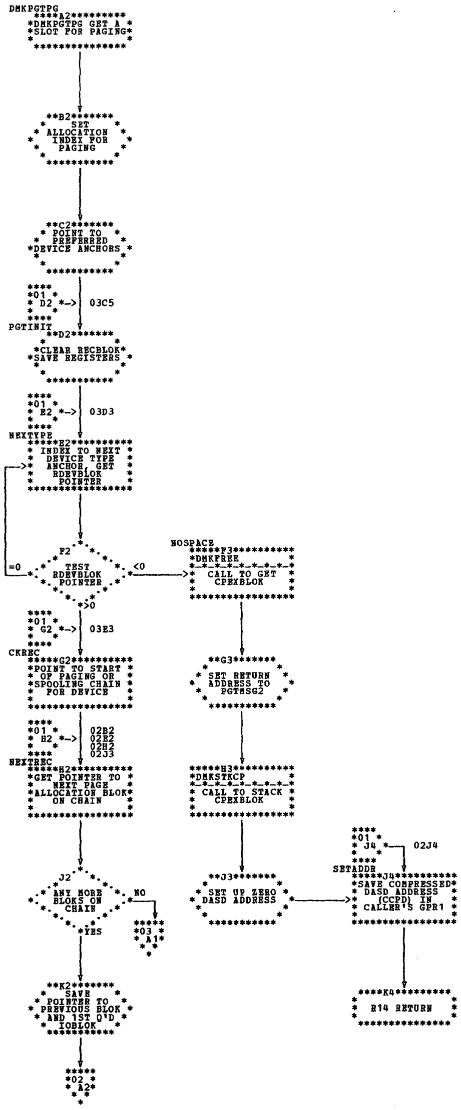

DMKPGSPP
A1
DMKPGSPP PARTIAL PAGE OUT
B1
FLAG AS ENTRY TO PARTIAL PAGE OUT
01 C1
02C4
PAGOUT1
C1
CALCULATE NUMBER OF PAGES TO BE RELEASED
D1
USING 1ST PAGE NUMBER, INDEX TO SEGTABLE ENTRY FOR 1ST SEGMENT
E1
SET UP REGISTERS TO CLEAR CORTABLE AND PAGTABLE ENTRIES
F1
PAGOUT2
F1
INDEX TO PAGTABLE AND SWPTABLE ENTRIES FOR 1ST PAGE TO RELEASE
G1
PAGOUT3
G1
SHARED PAGE
YES
NO
A3
H1
- TRANS - PARM = DEFER TO ENQUEUE ON PAGE
J1
PAGE RESIDENT
NO
YES
A2
A2
A2
DECREMENT USER'S RESIDENT PAGE COUNT
B2
INDEX TO CORTABLE ENTRY FOR REAL PAGE FRAME
C2
CLEAR OUT THE PAGTABLE POINTER IN CORTABLE ENTRY
D2
UNCHAIN CORTABLE ENTRY FROM USER LIST
E2
RELEASING RESERVED PAGE
NO
YES
F2
ADD 1 TO AVAILABLE PAGE COUNT, DECREMENT RESERVED PAGES
PAGOUT4
G2
DMKPTRFT
CALL TO PLACE ENTRY ON FREE PAGE LIST
H2
INVALIDATE PAGTABLE ENTRY, PURGE CPU TABLE LOOKASIDE BUFFER
CKRELSE
J2
DASD PAGE READ ONLY
YES
NO
K2
DMKPGTPR
CALL TO RELEASE DASD PAGE SLOT
A3
NEXTPAGE
A3
DECREMENT AND TEST PAGE COUNT
B3
PAGE COUNT ZERO
NO
YES
01 C3
02A3
02D3
RETURN
C3
RETURN TO CALLER
CKSEG
B4
BUMP VIRTUAL PAGE ADDRESS BY 4096
C4
LAST PAGE IN SEGMENT
YES
NO
D4
INDEX TO NEXT PAGTABLE AND SWPTABLE ENTRIES
G1
NEXTSEG
C5
INDEX TO NEXT SEGTABLE ENTRY
D5
CLEAR PAGTABLE INDEX TO ZERO
F1

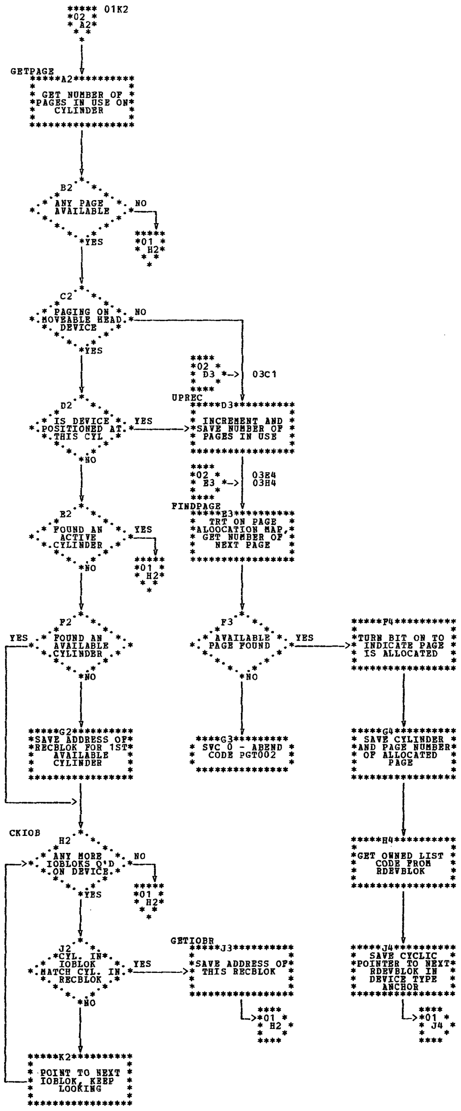

DMKPGSPO
A1
DMKPGSPO
B1
FLAG AS ENTRY TO FULL PAGE OUT
C1
RELEASING A SHARED SYSTEM
NO
A3
YES
D1
GET ADDRESS OF SEGMENT TABLE AND NUMBER OF SHARED SEGMENTS
E1
DECREMENT AND TEST SHARED SYSTEM USE COUNT
F1
USE COUNT NOW ZERO
NO
A5
YES
G1
UNCHAIN SHRTABLE FROM ACTIVE SYSTEM LIST
H1
POINT TO FIRST SHARED PAGTABLE ADDRESS
SHRTDEL
J1
GET NUMBER OF PAGES IN THIS SEGMENT
K1
POINT TO SHARED SWPTABLE
A2
A2
UNSHARE
A2
MARK SWPTABLE ENTRY AS NOT SHARED
B2
INDEX TO NEXT SWPTABLE ENTRY
C2
DECREMENT AND TEST PAGE COUNT
D2
PAGE COUNT ZERO
NO
YES
E2
INDEX TO NEXT PAGTABLE POINTER
F2
DECREMENT AND TEST SHARED SEGMENT COUNT
G2
SEGMENT COUNT ZERO
NO
YES
H2
GET ADDRESS AND SIZE OF SHRTABLE
J2
DMKFRET
CALL TO RETURN TO FREE STORAGE
02 K2
03B4
ZIPVMSHR
K2
CLEAR SHRTABLE POINTER IN VMBLOK, UNFLAG SHARED SYSTEM USER
A3
CKCLEAR
A3
"NOCLEAR" PARM PASSED
YES
01 C3
NO
B3
RESET ANY ACTIVE TRACING OR ADSTOP
C3
CALLED FOR V=R USER
NO
YES
D3
CLEAR V=R AREA TO 0
01 C3
PAGOUTC
C4
SET 1ST RELEASED PAGE =0, LAST PAGE = END OF VM
01 C1
A5
SHRSAVCT
A5
SAVE COUNT OF REMAINING SHARED SYSTEM USERS
03 A2

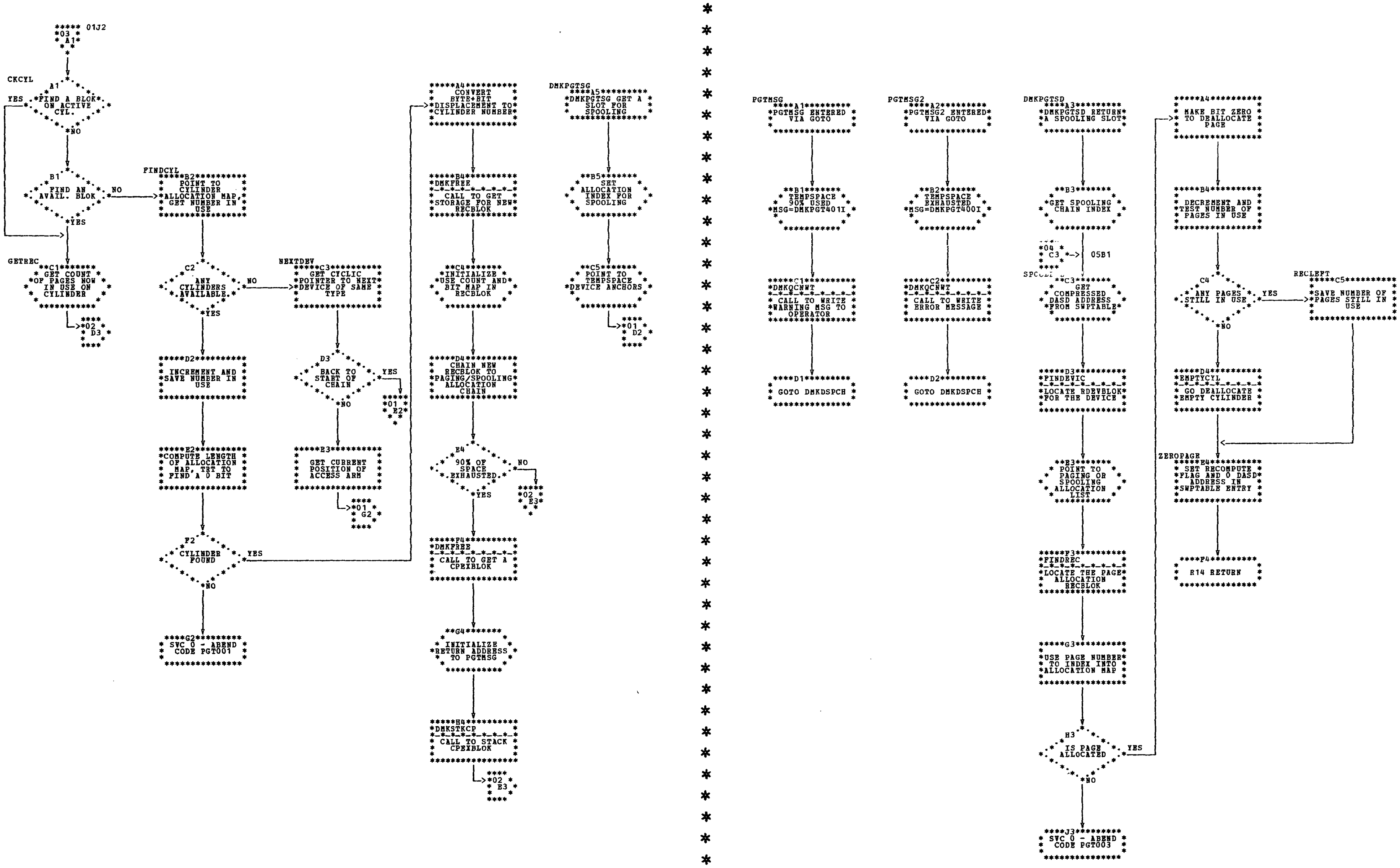

02A5
03
A2
BLDPAGLP
A2
GET SEGMENT NUMBER OF NEXT SHARED SEGMENT
A2
B2
INDEX INTO SEGTABLE TO CORRECT ENTRY FOR SHARED SEGMENT
C2
GET NUMBER OF PAGES IN SHARED SEGMENT
D2
GET NUMBER OF DOUBLE WORDS REQUIRED FOR PAGTABLE
E2
DMKFREE
CALL TO GET STORAGE FOR PAGTABLE
F2
SAVE ADDRESS OF SHARED PAGTABLE
G2
INSERT ADDRESS OF UNSHARED PAGTABLE IN SEGTABLE ENTRY
H2
GET NUMBER OF DOUBLE WORDS REQUIRED FOR SWPTABLE
J2
DMKFREE
CALL TO GET STORAGE FOR SWPTABLE
K2
CHAIN VMBLOK AND PAGTABLE TO SWPTABLE, SWPTABLE TO PAGTABLE
A3
GET ADDRESS OF OLD SHARED PAGTABLE
B3
USING SWPTABLE CHAIN, GET ADDRESS OF SHARED SWPTABLE
C3
GET LENGTH OF SWPTABLE, AND MOVE ENTRIES TO NEW SWPTABLE
D3
GET NUMBER OF PAGES IN SEGMENT
INITLOOP
E3
INITIALIZE NEXT PAGTABLE ENTRY TO "NOT AVAILABLE"
F3
INITIALIZE NEXT SWPTABLE ENTRY TO INDICATE "NOT SHARED"
G3
INDEX TO NEXT PAGTABLE AND SWPTABLE ENTRIES
H3
DECREMENT AND TEST PAGE COUNT
J3
PAGE COUNT ZERO
NO
YES
A4
DECREMENT AND TEST SHARED SEGMENT COUNT
B4
SEGMENT COUNT ZERO
NO
YES
02
K2
A2

DMKPGS -- Release Virtual Storage (Part 3 of 3)

DMKPGT -- Allocate DASD/Virtual Storage (Parts 1 and 2 of 8)

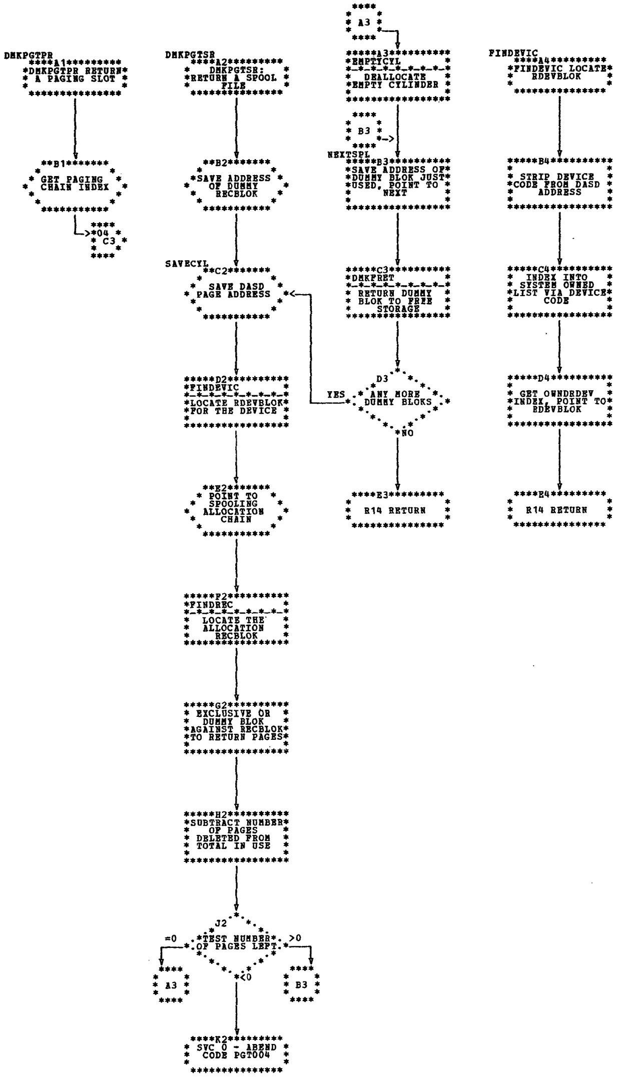

DMKPGTPG
A2
DMKPGTPG GET A SLOT FOR PAGING
B2
SET ALLOCATION INDEX FOR PAGING
C2
POINT TO PREFERRED DEVICE ANCHORS
01 D2 03C5
PGTINIT
D2
CLEAR RECBLOK SAVE REGISTERS
01 E2 03D3
NEXTYPE
E2
INDEX TO NEXT DEVICE TYPE ANCHOR, GET RDEVBLOK POINTER
F2
TEST RDEVBLOK POINTER
=0
<0
>0
NOSPACE
F3
DMKFREE
CALL TO GET CPEXBLOK
01 G2 03E3
CKREC
G2
POINT TO START OF PAGING OR SPOOLING CHAIN FOR DEVICE
G3
SET RETURN ADDRESS TO PGTMSG2
01 H2 02B2 02E2 02H2 02J3
NEXTREC
H2
GET POINTER TO NEXT PAGE ALLOCATION BLOK ON CHAIN
H3
DMKSTKCP
CALL TO STACK CPEXBLOK
01 J4 02J4
SETADDR
J2
ANY MORE BLOKS ON CHAIN
NO
03 A1
YES
J3
SET UP ZERO DASD ADDRESS
J4
SAVE COMPRESSED DASD ADDRESS (CCPD) IN CALLER'S GPR1
K2
SAVE POINTER TO PREVIOUS BLOK AND 1ST Q'D IOBLOK
K4
R14 RETURN
02 A2

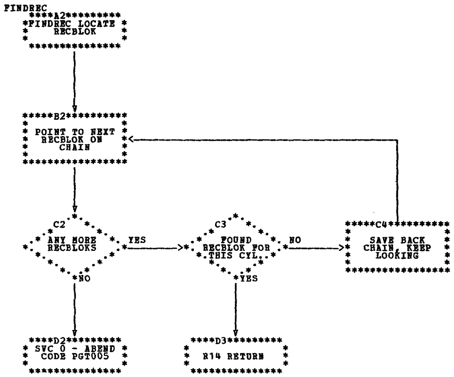

02 A2 01K2
GETPAGE
A2
GET NUMBER OF PAGES IN USE ON CYLINDER
B2
ANY PAGE AVAILABLE
NO
01 H2
YES
C2
PAGING ON MOVEABLE HEAD DEVICE
NO
YES
02 D3 03C1
UPREC
D2
IS DEVICE POSITIONED AT THIS CYL
YES
D3
INCREMENT AND SAVE NUMBER OF PAGES IN USE
NO
02 E3 03E4 03H4
FINDPAGE
E2
FOUND AN ACTIVE CYLINDER
YES
01 H2
NO
E3
TRT ON PAGE ALOOCATION MAP, GET NUMBER OF NEXT PAGE
F2
FOUND AN AVAILABLE CYLINDER
YES
NO
F3
AVAILABLE PAGE FOUND
YES
NO
F4
TURN BIT ON TO INDICATE PAGE IS ALLOCATED
G2
SAVE ADDRESS OF RECBLOK FOR 1ST AVAILABLE CYLINDER
G3
SVC 0 - ABEND CODE PGT002
G4
SAVE CYLINDER AND PAGE NUMBER OF ALLOCATED PAGE
CKIOB
H2
ANY MORE IOBLOKS Q'D ON DEVICE
NO
01 H2
YES
H4
GET OWNED LIST CODE FROM RDEVBLOK
GETIOBR
J2
CYL. IN IOBLOK MATCH CYL. IN RECBLOK
YES
J3
SAVE ADDRESS OF THIS RECBLOK
01 H2
J4
SAVE CYCLIC POINTER TO NEXT RDEVBLOK IN DEVICE TYPE ANCHOR
01 J4
NO
K2
POINT TO NEXT IOBLOK, KEEP LOOKING

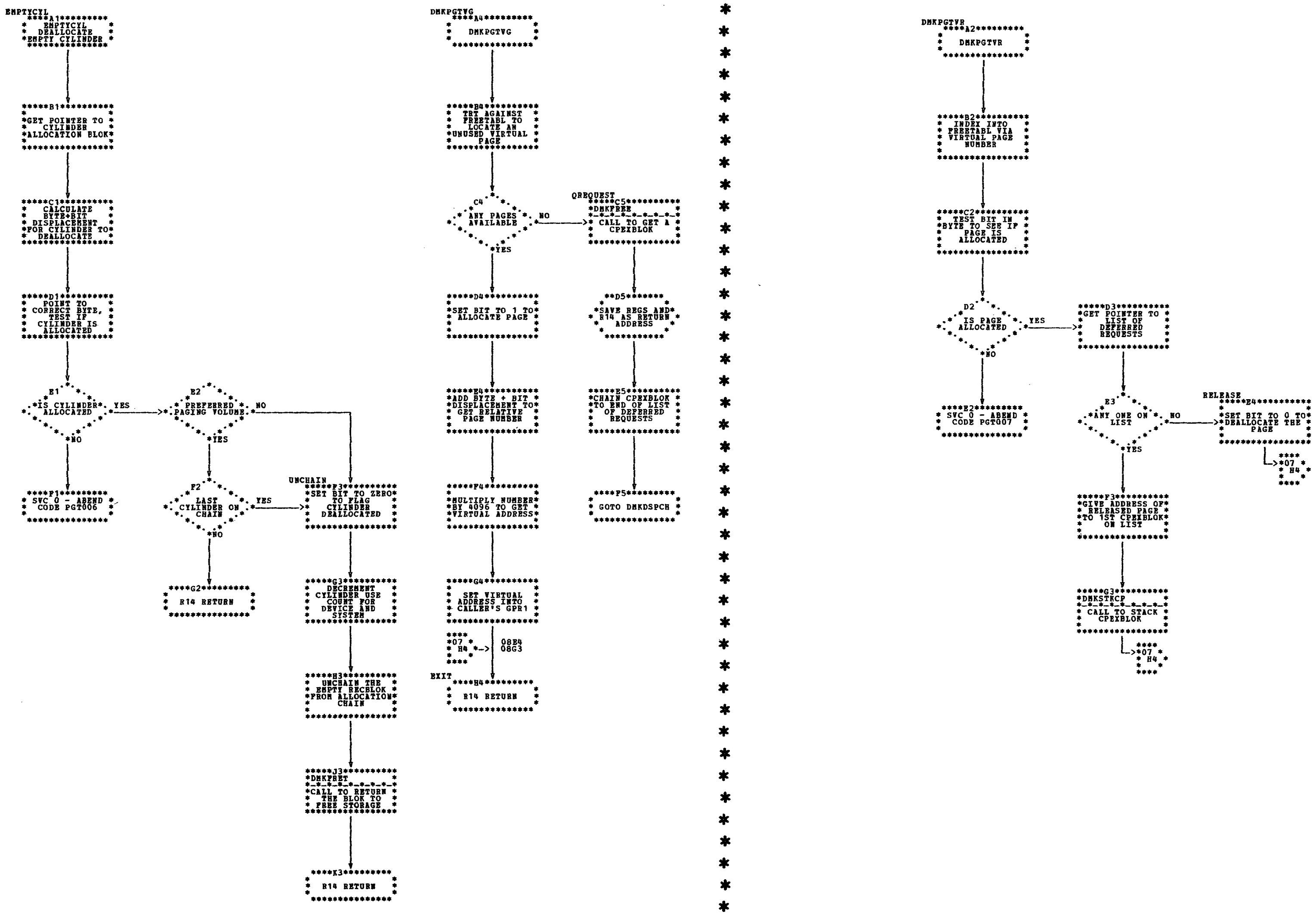

03 A1 01J2
CKCYL
A1 FIND A BLOK ON ACTIVE CYL.
YES
NO
B1 FIND AN AVAIL. BLOK
NO
YES
FINDCYL
B2 POINT TO CYLINDER ALLOCATION MAP, GET NUMBER IN USE
GETREC
C1 GET COUNT OF PAGES NOW IN USE ON CYLINDER
02 D3
C2 ANY CYLINDERS AVAILABLE.
NO
YES
NEXTDEV
C3 GET CYCLIC POINTER TO NEXT DEVICE OF SAME TYPE
D2 INCREMENT AND SAVE NUMBER IN USE
D3 BACK TO START OF CHAIN
YES
01 E2
NO
E2 COMPUTE LENGTH OF ALLOCATION MAP, TRT TO FIND A 0 BIT
E3 GET CURRENT POSITION OF ACCESS ARM
01 G2
F2 CYLINDER FOUND
YES
NO
G2 SVC 0 - ABEND CODE PGT001
A4 CONVERT BYTE+BIT DISPLACEMENT TO CYLINDER NUMBER
B4 DMKFREE CALL TO GET STORAGE FOR NEW RECBLOK
C4 INITIALIZE USE COUNT AND BIT MAP IN RECBLOK
D4 CHAIN NEW RECBLOK TO PAGING/SPOOLING ALLOCATION CHAIN
E4 90% OF SPACE EXHAUSTED.
NO
02 E3
YES
F4 DMKFREE CALL TO GET A CPEXBLOK
G4 INITIALIZE RETURN ADDRESS TO PGTMSG
H4 DMKSTKCP CALL TO STACK CPEXBLOK
02 E3
DMKPGTSG
A5 DMKPGTSG GET A SLOT FOR SPOOLING
B5 SET ALLOCATION INDEX FOR SPOOLING
C5 POINT TO TEMPSPACE DEVICE ANCHORS
01 D2
PGTMSG
A1 PGTMSG ENTERED VIA GOTO
B1 TEMPSPACE 90% USED MSG=DMKPGT401I
C1 DMKQCNWT CALL TO WRITE WARNING MSG TO OPERATOR
D1 GOTO DMKDSPCH
PGTMSG2
A2 PGTMSG2 ENTERED VIA GOTO
B2 TEMPSPACE EXHAUSTED MSG=DMKPGT400I
C2 DMKQCNWT CALL TO WRITE ERROR MESSAGE
D2 GOTO DMKDSPCH
DMKPGTSD
A3 DMKPGTSD RETURN A SPOOLING SLOT
B3 GET SPOOLING CHAIN INDEX
04 C3
05B1
SPCOLI
C3 GET COMPRESSED DASD ADDRESS FROM SWPTABLE
D3 FINDEVIC LOCATE RDEVBLOK FOR THE DEVICE
E3 POINT TO PAGING OR SPOOLING ALLOCATION LIST
F3 FINDREC LOCATE THE PAGE ALLOCATION RECBLOK
G3 USE PAGE NUMBER TO INDEX INTO ALLOCATION MAP
H3 IS PAGE ALLOCATED
YES
NO
J3 SVC 0 - ABEND CODE PGT003
A4 MAKE BIT ZERO TO DEALLOCATE PAGE
B4 DECREMENT AND TEST NUMBER OF PAGES IN USE
C4 ANY PAGES STILL IN USE
YES
NO
RECLEFT
C5 SAVE NUMBER OF PAGES STILL IN USE
D4 EMPTYCYL GO DEALLOCATE EMPTY CYLINDER
ZEROPAGE
E4 SET RECOMPUTE FLAG AND 0 DASD ADDRESS IN SWPTABLE ENTRY
F4 R14 RETURN

DMKPGT -- Allocate DASD/Virtual Storage (Parts 3 and 4 of 8)

DMKPGT -- Allccate DASD/Virtual Storage (Parts 5 and 6 of 8)

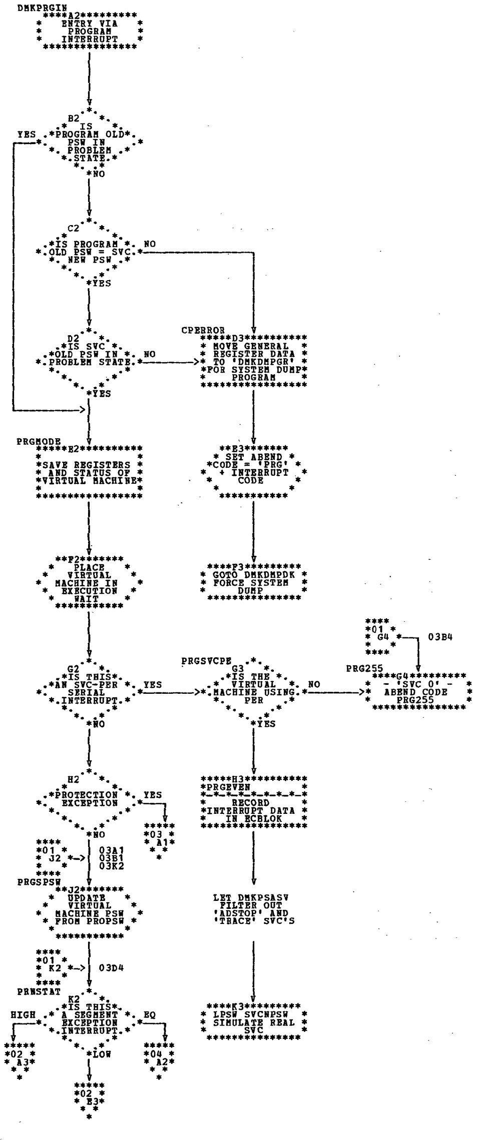

DMKPGTPR
A1
DMKPGTPR RETURN A PAGING SLOT
B1
GET PAGING CHAIN INDEX
04 C3
DMKPGTSR
A2
DMKPGTSR: RETURN A SPOOL FILE
B2
SAVE ADDRESS OF DUMMY RECBLOK
SAVECYL
C2
SAVE DASD PAGE ADDRESS
D2
FINDEVIC
LOCATE RDEVBLOK FOR THE DEVICE
E2
POINT TO SPOOLING ALLOCATION CHAIN
F2
FINDREC
LOCATE THE ALLOCATION RECBLOK
G2
EXCLUSIVE OR DUMMY BLOK AGAINST RECBLOK TO RETURN PAGES
H2
SUBTRACT NUMBER OF PAGES DELETED FROM TOTAL IN USE
J2
TEST NUMBER OF PAGES LEFT
=0
>0
<0
A3
B3
K2
SVC 0 - ABEND CODE PGT004
A3
A3
EMPTYCYL
DEALLOCATE EMPTY CYLINDER
B3
NEXTSPL
B3
SAVE ADDRESS OF DUMMY BLOK JUST USED, POINT TO NEXT
C3
DMKFRET
RETURN DUMMY BLOK TO FREE STORAGE
D3
ANY MORE DUMMY BLOKS
YES
NO
E3
R14 RETURN
FINDEVIC
A4
FINDEVIC LOCATE RDEVBLOK
B4
STRIP DEVICE CODE FROM DASD ADDRESS
C4
INDEX INTO SYSTEM OWNED LIST VIA DEVICE CODE
D4
GET OWNDRDEV INDEX, POINT TO RDEVBLOK
E4
R14 RETURN

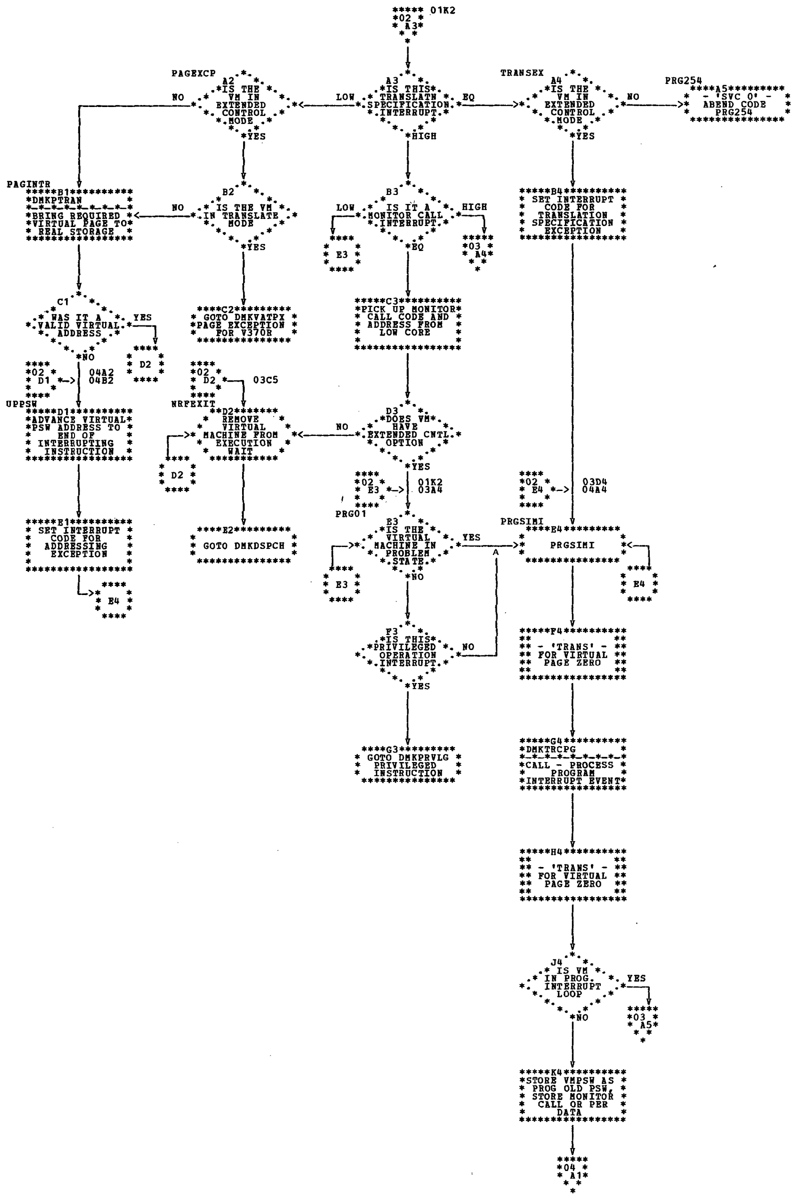

FINDREC
A2
FINDREC LOCATE RECBLOK
B2
POINT TO NEXT RECBLOK ON CHAIN
C2
ANY MORE RECBLOKS
YES
NO
C3
FOUND RECBLOK FOR THIS CYL.
NO
YES
C4
SAVE BACK CHAIN, KEEP LOOKING
D2
SVC 0 - ABEND CODE PGT005
D3
R14 RETURN

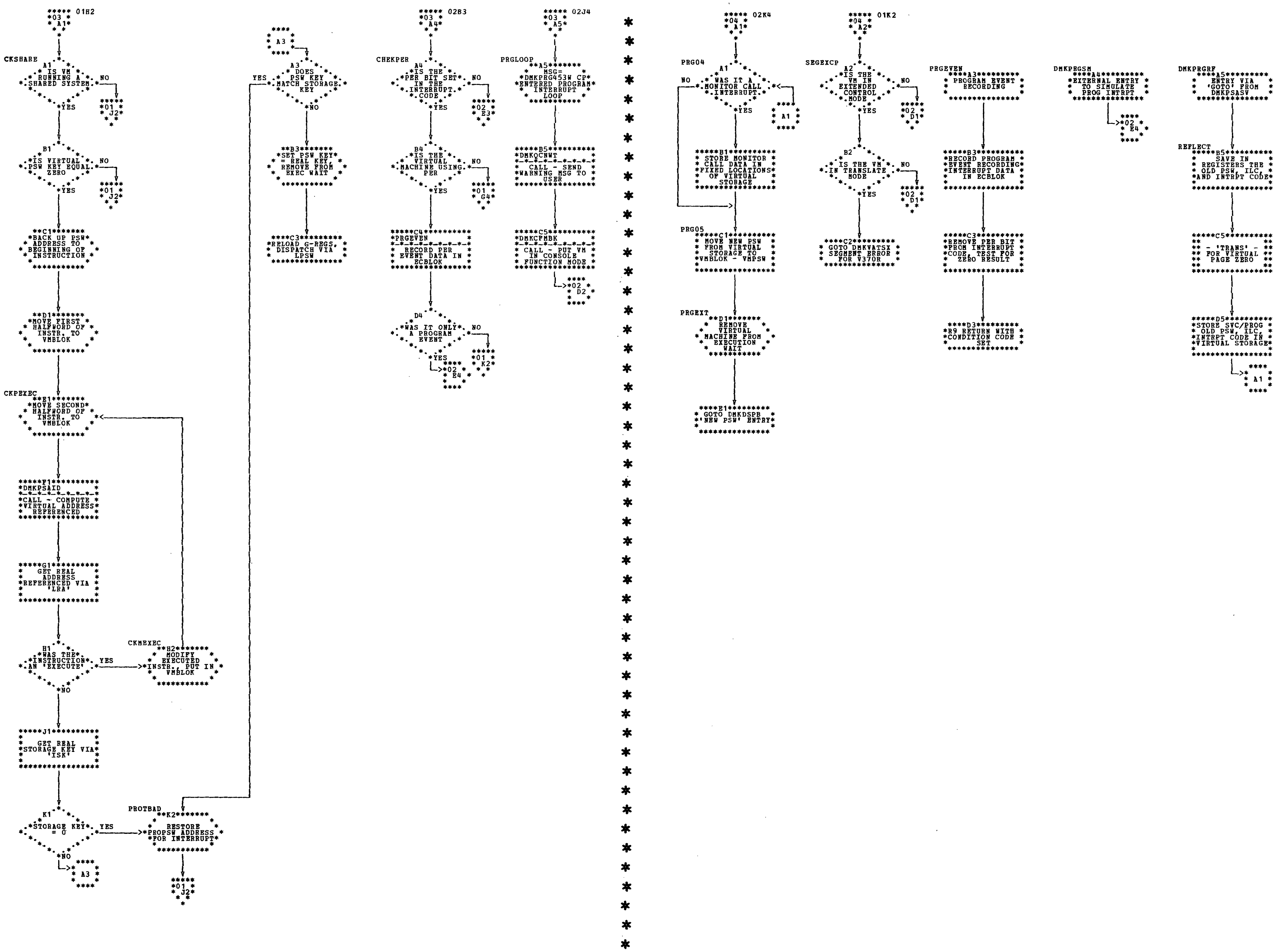

EMPTYCYL
A1
EMPTYCYL DEALLOCATE EMPTY CYLINDER
B1
GET POINTER TO CYLINDER ALLOCATION BLOK
C1
CALCULATE BYTE+BIT DISPLACEMENT FOR CYLINDER TO DEALLOCATE
D1
POINT TO CORRECT BYTE, TEST IF CYLINDER IS ALLOCATED
E1
IS CYLINDER ALLOCATED
YES
NO
F1
SVC 0 - ABEND CODE PGT006
E2
PREFERRED PAGING VOLUME
NO
YES
F2
LAST CYLINDER ON CHAIN
YES
NO
G2
R14 RETURN
UNCHAIN
F3
SET BIT TO ZERO TO FLAG CYLINDER DEALLOCATED
G3
DECREMENT CYLINDER USE COUNT FOR DEVICE AND SYSTEM
H3
UNCHAIN THE EMPTY RECBLOK FROM ALLOCATION CHAIN
J3
DMKFRET
CALL TO RETURN THE BLOK TO FREE STORAGE
K3
R14 RETURN
DMKPGTVG
A4
DMKPGTVG
B4
TRT AGAINST FREETABL TO LOCATE AN UNUSED VIRTUAL PAGE
C4
ANY PAGES AVAILABLE
NO
YES
QREQUEST
C5
DMKFREE
CALL TO GET A CPEXBLOK
D5
SAVE REGS AND R14 AS RETURN ADDRESS
E5
CHAIN CPEXBLOK TO END OF LIST OF DEFERRED REQUESTS
F5
GOTO DMKDSPCH
D4
SET BIT TO 1 TO ALLOCATE PAGE
E4
ADD BYTE + BIT DISPLACEMENT TO GET RELATIVE PAGE NUMBER
F4
MULTIPLY NUMBER BY 4096 TO GET VIRTUAL ADDRESS
G4
SET VIRTUAL ADDRESS INTO CALLER'S GPR1
07 H4
08E4
08G3
EXIT
H4
R14 RETURN
DMKPGTVR
A2
DMKPGTVR
B2
INDEX INTO FREETABL VIA VIRTUAL PAGE NUMBER
C2
TEST BIT IN BYTE TO SEE IF PAGE IS ALLOCATED
D2
IS PAGE ALLOCATED
YES
NO
E2
SVC 0 - ABEND CODE PGT007
D3
GET POINTER TO LIST OF DEFERRED REQUESTS
E3
ANY ONE ON LIST
NO
YES
RELEASE
E4
SET BIT TO 0 TO DEALLOCATE THE PAGE
07 H4
F3
GIVE ADDRESS OF RELEASED PAGE TO 1ST CPEXBLOK ON LIST
G3
DMKSTKCP
CALL TO STACK CPEXBLOK
07 H4

DMKPGT -- Allocate DASD/Virtual Storage (Parts 7 and 8 of 8)

| DMKPRG -- Program Interrupt Handler (Parts 1 and 2 of 4)

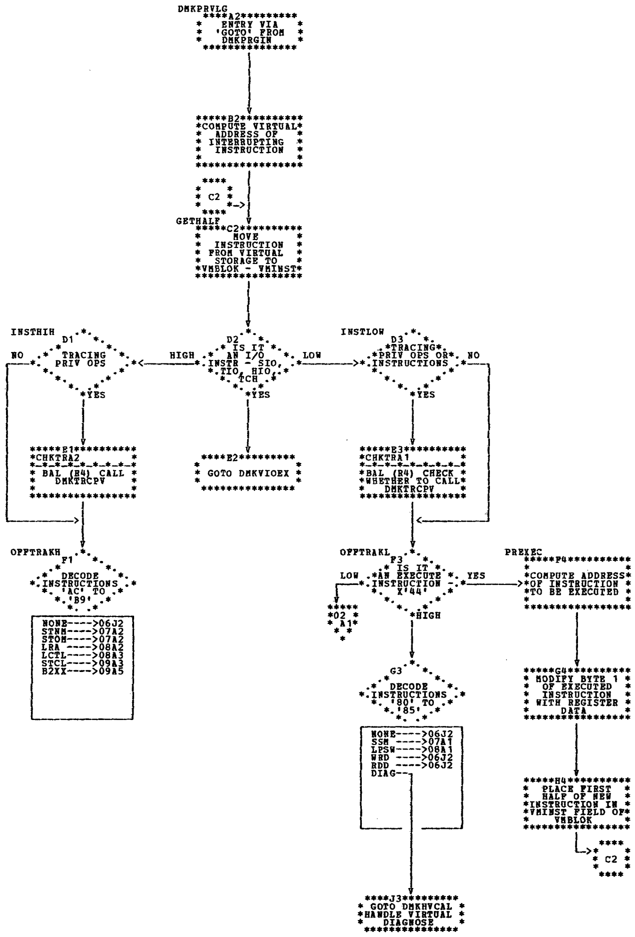

DMKPRGIN
A2 ENTRY VIA PROGRAM INTERRUPT
B2 IS PROGRAM OLD PSW IN PROBLEM STATE
YES
NO
C2 IS PROGRAM OLD PSW = SVC NEW PSW
NO
YES
D2 IS SVC OLD PSW IN PROBLEM STATE
NO
YES
CPERROR
D3 MOVE GENERAL REGISTER DATA TO 'DMKDMPGR' FOR SYSTEM DUMP PROGRAM
PRGMODE
E2 SAVE REGISTERS AND STATUS OF VIRTUAL MACHINE
E3 SET ABEND CODE = 'PRG' + INTERRUPT CODE
F2 PLACE VIRTUAL MACHINE IN EXECUTION WAIT
F3 GOTO DMKDMPDK FORCE SYSTEM DUMP
01 G4 03B4
G2 IS THIS AN SVC-PER SERIAL INTERRUPT
YES
NO
PRGSVCPE
G3 IS THE VIRTUAL MACHINE USING PER
NO
YES
PRG255
G4 - 'SVC 0' - ABEND CODE PRG255
H2 PROTECTION EXCEPTION
YES
NO
03 A1
H3 PRGEVEN RECORD INTERRUPT DATA IN ECBLOK
01 J2 03A1 03B1 03K2
PRGSPSW
J2 UPDATE VIRTUAL MACHINE PSW FROM PROPSW
LET DMKPSASV FILTER OUT 'ADSTOP' AND 'TRACE' SVC'S
01 K2 03D4
PRNSTAT
K2 IS THIS A SEGMENT EXCEPTION INTERRUPT
HIGH
EQ
LOW
02 A3
04 A2
02 E3
K3 LPSW SVCNPSW SIMULATE REAL SVC

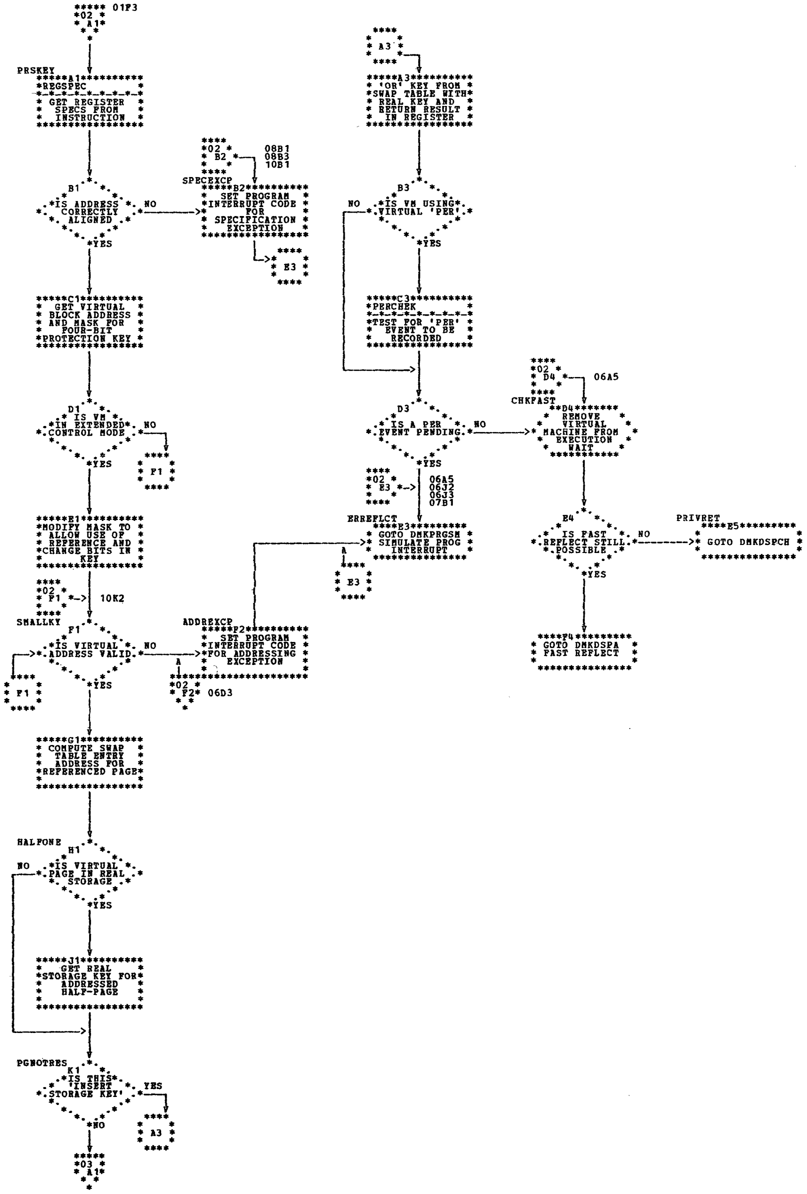

02 A3 01K2
PAGEXCP
A2 IS THE VM IN EXTENDED CONTROL MODE
NO
YES
A3 IS THIS TRANSLATN SPECIFICATION INTERRUPT
LOW
EQ
HIGH
TRANSEX
A4 IS THE VM IN EXTENDED CONTROL MODE
NO
YES
PRG254
A5 - 'SVC 0' - ABEND CODE PRG254
PAGINTR
B1 DMKPTRAN BRING REQUIRED VIRTUAL PAGE TO REAL STORAGE
B2 IS THE VM IN TRANSLATE MODE
NO
YES
B3 IS IT A MONITOR CALL INTERRUPT
LOW
HIGH
EQ
E3
03 A4
B4 SET INTERRUPT CODE FOR TRANSLATION SPECIFICATION EXCEPTION
C1 WAS IT A VALID VIRTUAL ADDRESS
YES
NO
D2
C2 GOTO DMKVATPX PAGE EXCEPTION FOR V370R
C3 PICK UP MONITOR CALL CODE AND ADDRESS FROM LOW CORE
02 D1 04A2 04B2
02 D2 03C5
UPPSW
D1 ADVANCE VIRTUAL PSW ADDRESS TO END OF INTERRUPTING INSTRUCTION
NRFEXIT
D2 REMOVE VIRTUAL MACHINE FROM EXECUTION WAIT
D2
D3 DOES VM HAVE EXTENDED CNTL OPTION
NO
YES
02 E3 01K2 03A4
02 E4 03D4 04A4
PRG01
E1 SET INTERRUPT CODE FOR ADDRESSING EXCEPTION
E4
E2 GOTO DMKDSPCH
E3 IS THE VIRTUAL MACHINE IN PROBLEM STATE
YES
NO
E3
A
PRGSIMI
E4 PRGSIMI
E4
F3 IS THIS PRIVILEGED OPERATION INTERRUPT
NO
YES
F4 - 'TRANS' - FOR VIRTUAL PAGE ZERO
G3 GOTO DMKPRVLG PRIVILEGED INSTRUCTION
G4 DMKTRCPG CALL - PROCESS PROGRAM INTERRUPT EVENT
H4 - 'TRANS' - FOR VIRTUAL PAGE ZERO
J4 IS VM IN PROG. INTERRUPT LOOP
YES
NO
03 A5
K4 STORE VMPSW AS PROG OLD PSW, STORE MONITOR CALL OR PER DATA
04 A1

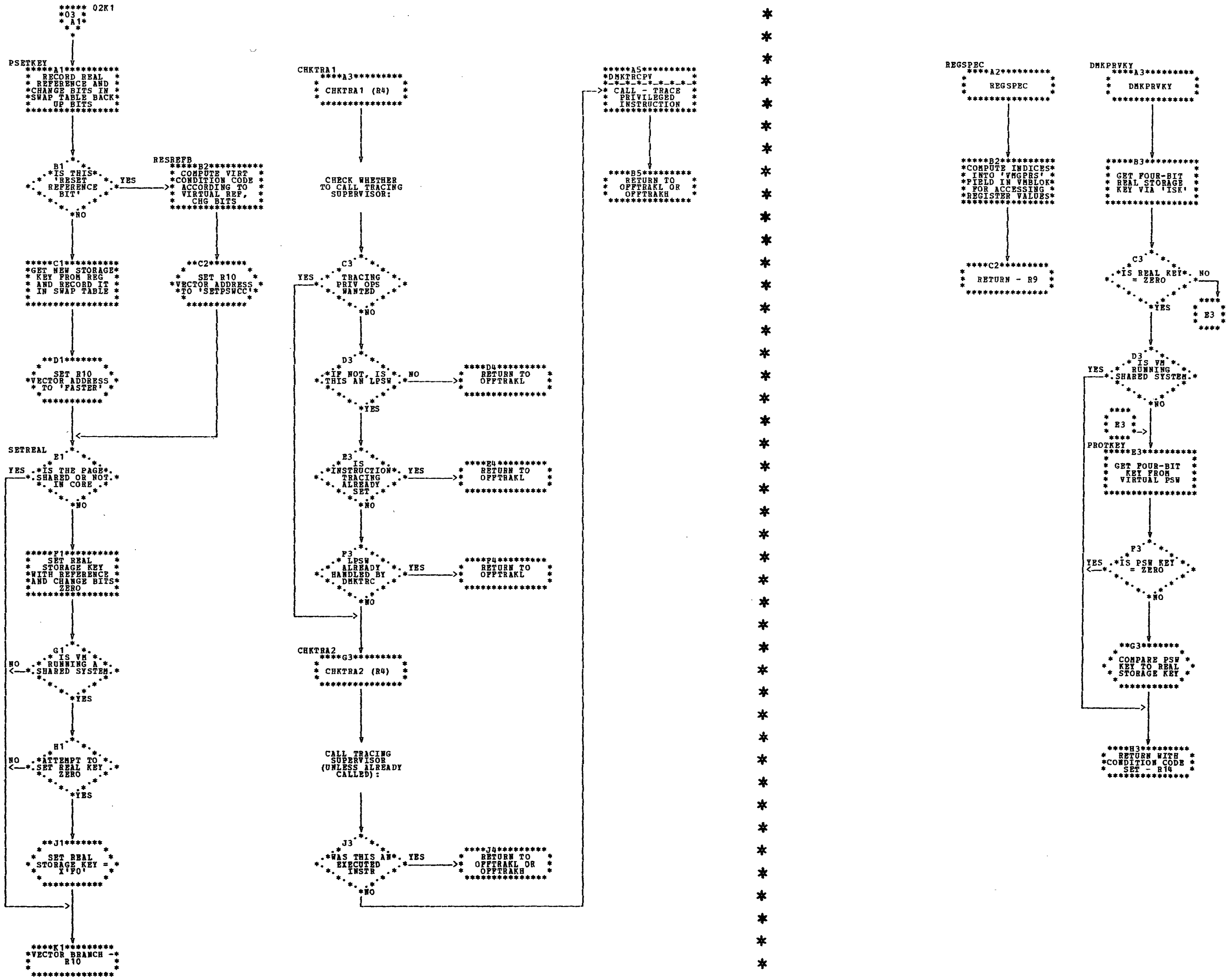

01H2
03 A1
CKSHARE
A1 IS VM RUNNING A SHARED SYSTEM
NO
01 J2
YES
B1 IS VIRTUAL PSW KEY EQUAL ZERO
NO
01 J2
YES
C1 BACK UP PSW ADDRESS TO BEGINNING OF INSTRUCTION
D1 MOVE FIRST HALFWORD OF INSTR. TO VMBLOK
CKPEXEC
E1 MOVE SECOND HALFWORD OF INSTR. TO VMBLOK
F1 DMKPSAID CALL - COMPUTE VIRTUAL ADDRESS REFERENCED
G1 GET REAL ADDRESS REFERENCED VIA 'LRA'
H1 WAS THE INSTRUCTION AN 'EXECUTE'
YES
NO
CKMEXEC
H2 MODIFY EXECUTED INSTR., PUT IN VMBLOK
J1 GET REAL STORAGE KEY VIA 'ISK'
K1 STORAGE KEY = 0
YES
NO
A3
PROTBAD
K2 RESTORE PROPSW ADDRESS FOR INTERRUPT
01 J2
A3
A3 DOES PSW KEY MATCH STORAGE KEY
YES
NO
B3 SET PSW KEY = REAL KEY, REMOVE FROM EXEC WAIT
C3 RELOAD G-REGS, DISPATCH VIA LPSW
02B3
03 A4
CHEKPER
A4 IS THE PER BIT SET IN THE INTERRUPT CODE
NO
02 E3
YES
B4 IS THE VIRTUAL MACHINE USING PER
NO
01 G4
YES
C4 PRGEVEN RECORD PER EVENT DATA IN ECBLOK
D4 WAS IT ONLY A PROGRAM EVENT
NO
01 K2
YES
02 E4
02J4
03 A5
PRGLOOP
A5 MSG= DMKPRG453W CP ENTERED PROGRAM INTERRUPT LOOP
B5 DMKQCNWT CALL - SEND WARNING MSG TO USER
C5 DMKCFMBK CALL - PUT VM IN CONSOLE FUNCTION MODE
02 D2
02K4
04 A1
PRG04
A1 WAS IT A MONITOR CALL INTERRUPT
NO
YES
A1
B1 STORE MONITOR CALL DATA IN FIXED LOCATIONS OF VIRTUAL STORAGE
PRG05
C1 MOVE NEW PSW FROM VIRTUAL STORAGE TO VMBLOK - VMPSW
PRGEXT
D1 REMOVE VIRTUAL MACHINE FROM EXECUTION WAIT
E1 GOTO DMKDSPB 'NEW PSW' ENTRY
01K2
04 A2
SEGEXCP
A2 IS THE VM IN EXTENDED CONTROL MODE
NO
02 D1
YES
B2 IS THE VM IN TRANSLATE MODE
NO
02 D1
YES
C2 GOTO DMKVATSX SEGMENT ERROR FOR V370R
PRGEVEN
A3 PROGRAM EVENT RECORDING
B3 RECORD PROGRAM EVENT RECORDING INTERRUPT DATA IN ECBLOK
C3 REMOVE PER BIT FROM INTERRUPT CODE, TEST FOR ZERO RESULT
D3 R9 RETURN WITH CONDITION CODE SET
DMKPRGSM
A4 EXTERNAL ENTRY TO SIMULATE PROG INTRPT
02 E4
DMKPRGRF
A5 ENTRY VIA 'GOTO' FROM DMKPSASV
REFLECT
B5 SAVE IN REGISTERS THE OLD PSW, ILC, AND INTRPT CODE
C5 - 'TRANS' - FOR VIRTUAL PAGE ZERO
D5 STORE SVC/PROG OLD PSW, ILC, INTRPT CODE IN VIRTUAL STORAGE
A1

DMKPRG -- Program Interrupt Handler (Parts 3 and 4 of 4)

DMKPRV -- Simulate Virtual Machine Privileged Instruction (Parts 1 and 2 of 10)

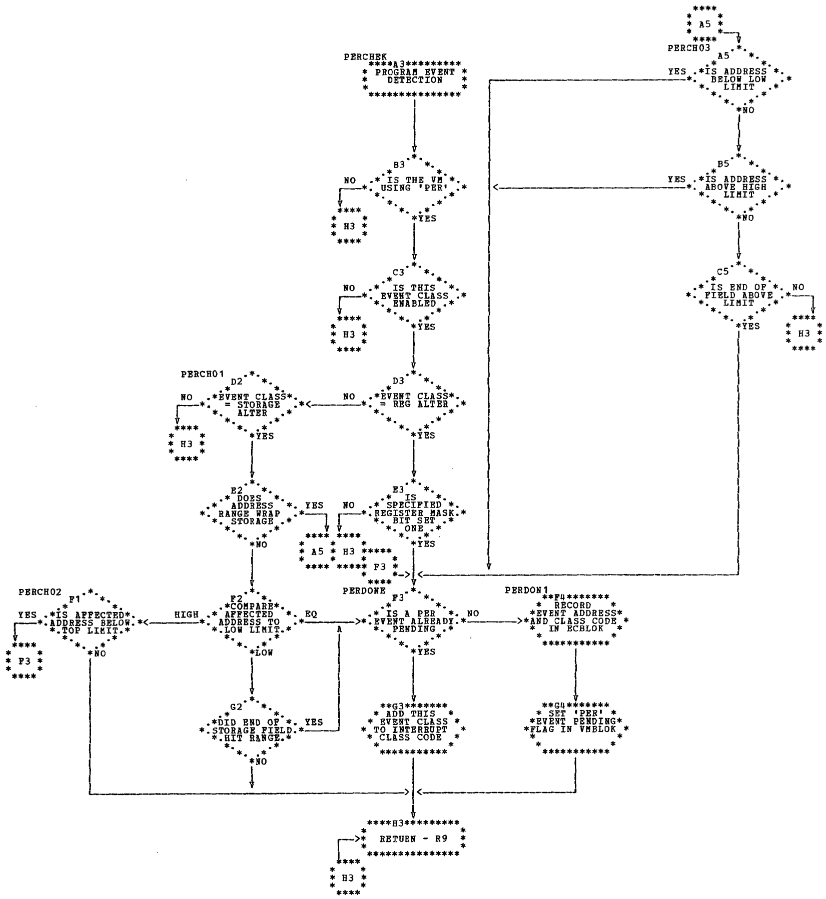

DMKPRVLG
A2 ENTRY VIA 'GOTO' FROM DMKPRGIN
B2 COMPUTE VIRTUAL ADDRESS OF INTERRUPTING INSTRUCTION
C2
GETHALF
C2 MOVE INSTRUCTION FROM VIRTUAL STORAGE TO VMBLOK - VMINST
INSTHIH
D1 TRACING PRIV OPS
NO
YES
HIGH
D2 IS IT AN I/O INSTR - SIO, TIO, HIO, TCH
LOW
YES
INSTLOW
D3 TRACING PRIV OPS OR INSTRUCTIONS
NO
YES
E1 CHKTRA2
BAL (R4) CALL DMKTRCPV
E2 GOTO DMKVIOEX
E3 CHKTRA1
BAL (R4) CHECK WHETHER TO CALL DMKTRCPV
OFFTRAKH
F1 DECODE INSTRUCTIONS 'AC' TO 'B9'
NONE---->06J2
STNM---->07A2
STOM---->07A2
LRA ---->08A2
LCTL---->08A3
STCL---->09A3
B2XX---->09A5
OFFTRAKL
F3 IS IT AN EXECUTE INSTRUCTION - X'44'
LOW
YES
HIGH
02 A1
PREXEC
F4 COMPUTE ADDRESS OF INSTRUCTION TO BE EXECUTED
G3 DECODE INSTRUCTIONS '80' TO '85'
NONE---->06J2
SSM ---->07A1
LPSW---->08A1
WRD ---->06J2
RDD ---->06J2
DIAG--
G4 MODIFY BYTE 1 OF EXECUTED INSTRUCTION WITH REGISTER DATA
H4 PLACE FIRST HALF OF NEW INSTRUCTION IN VMINST FIELD OF VMBLOK
C2
J3 GOTO DMKHVCAL HANDLE VIRTUAL DIAGNOSE

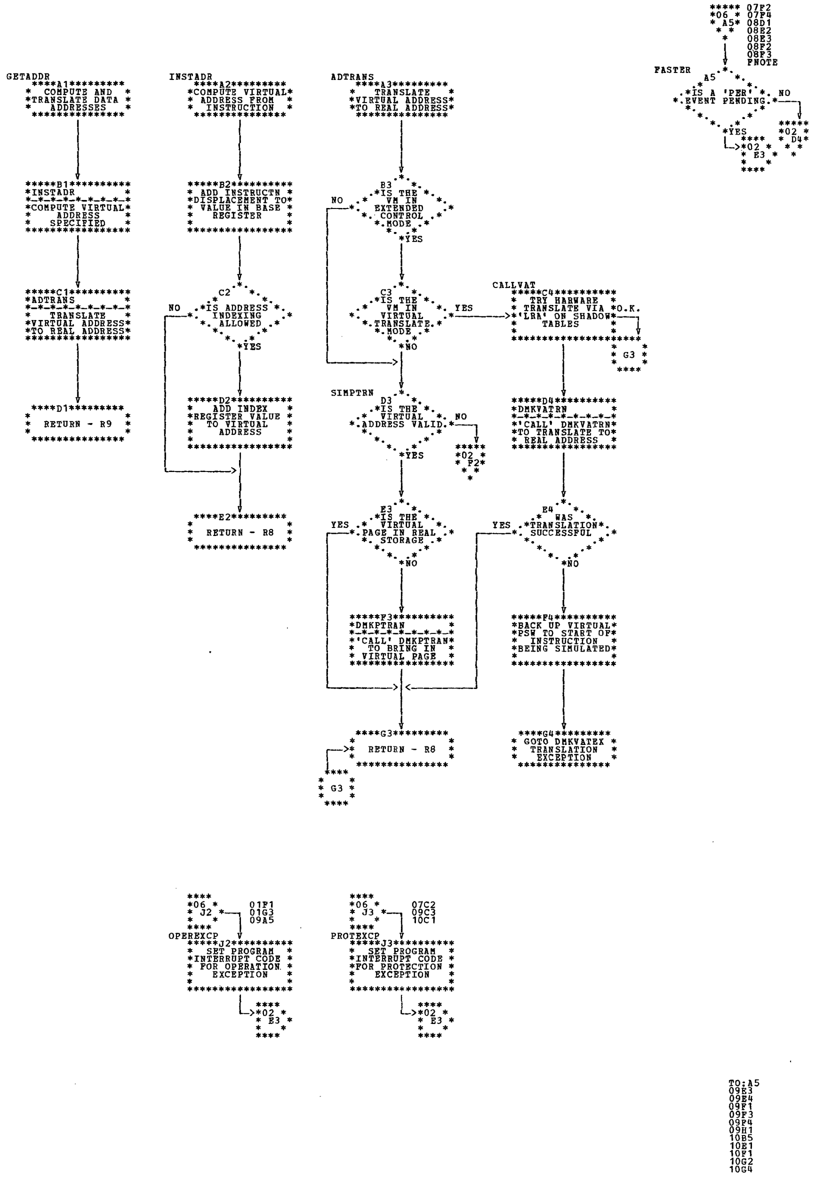

02 A1 01F3
PRSKEY
A1 REGSPEC
GET REGISTER SPECS FROM INSTRUCTION
02 B2 08B1 08B3 10B1
SPECEXCP
B1 IS ADDRESS CORRECTLY ALIGNED
NO
YES
B2 SET PROGRAM INTERRUPT CODE FOR SPECIFICATION EXCEPTION
E3
C1 GET VIRTUAL BLOCK ADDRESS AND MASK FOR FOUR-BIT PROTECTION KEY
D1 IS VM IN EXTENDED CONTROL MODE
NO
YES
F1
E1 MODIFY MASK TO ALLOW USE OF REFERENCE AND CHANGE BITS IN KEY
02 F1 10K2
SMALLKY
F1 IS VIRTUAL ADDRESS VALID
NO
YES
F1
ADDREXCP
F2 SET PROGRAM INTERRUPT CODE FOR ADDRESSING EXCEPTION
A
02 F2 06D3
G1 COMPUTE SWAP TABLE ENTRY ADDRESS FOR REFERENCED PAGE
HALFONE
H1 IS VIRTUAL PAGE IN REAL STORAGE
NO
YES
J1 GET REAL STORAGE KEY FOR ADDRESSED HALF-PAGE
PGNOTRES
K1 IS THIS 'INSERT STORAGE KEY'
YES
NO
A3
03 A1
A3
A3 'OR' KEY FROM SWAP TABLE WITH REAL KEY AND RETURN RESULT IN REGISTER
B3 IS VM USING VIRTUAL 'PER'
NO
YES
C3 PERCHEK
TEST FOR 'PER' EVENT TO BE RECORDED
D3 IS A PER EVENT PENDING
NO
YES
02 E3 06A5 06J2 06J3 07B1
ERREFLCT
E3 GOTO DMKPRGSM SIMULATE PROG INTERRUPT
A
E3
02 D4 06A5
CHKFAST
D4 REMOVE VIRTUAL MACHINE FROM EXECUTION WAIT
E4 IS FAST REFLECT STILL POSSIBLE
NO
YES
PRIVRET
E5 GOTO DMKDSPCH
F4 GOTO DMKDSPA FAST REFLECT

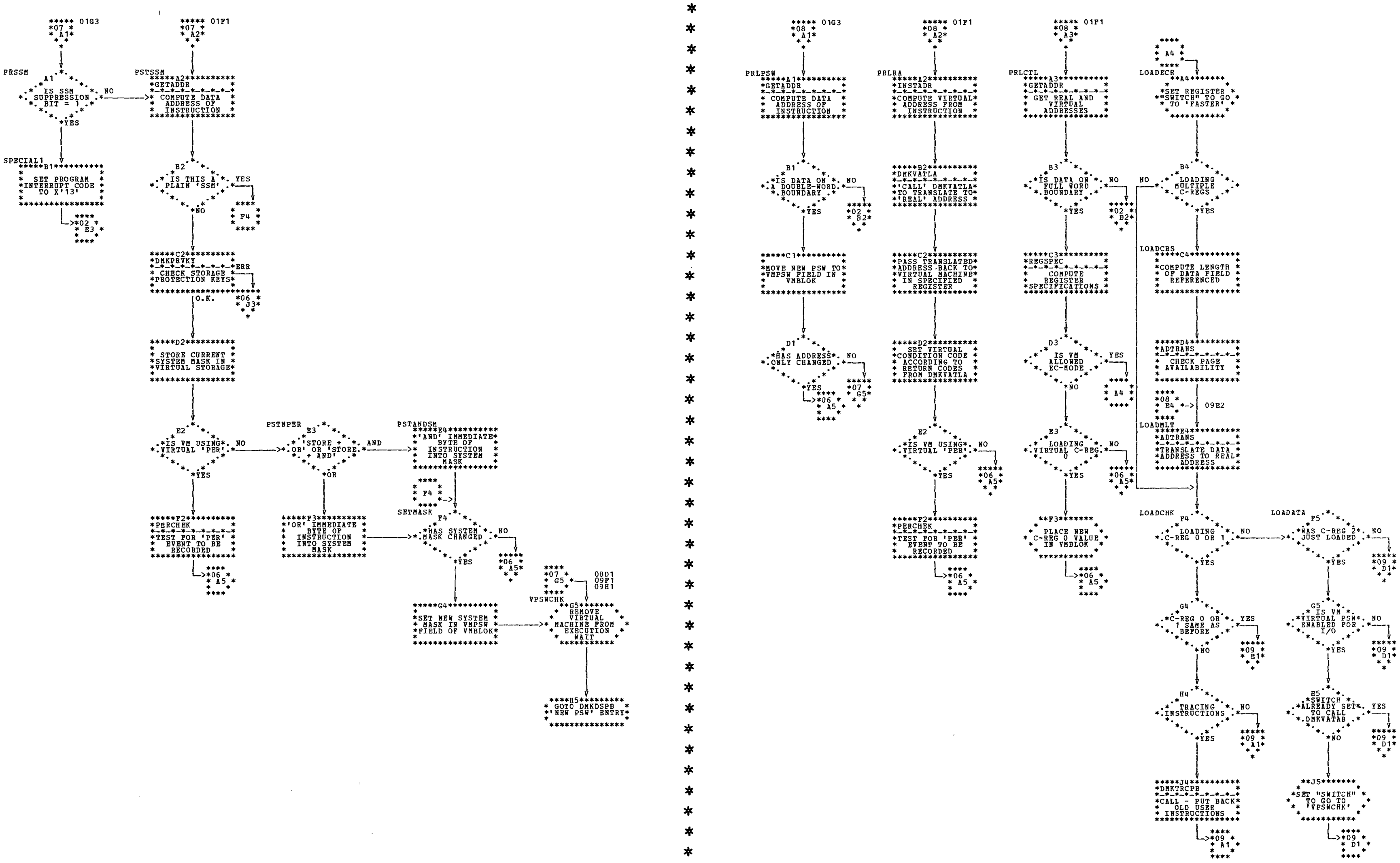

02K1
03
A1
PSETKEY
A1
RECORD REAL REFERENCE AND CHANGE BITS IN SWAP TABLE BACK UP BITS
B1
IS THIS 'RESET REFERENCE BIT'
YES
NO
RESREFB
B2
COMPUTE VIRT CONDITION CODE ACCORDING TO VIRTUAL REF, CHG BITS
C1
GET NEW STORAGE KEY FROM REG AND RECORD IT IN SWAP TABLE
C2
SET R10 VECTOR ADDRESS TO 'SETPSWCC'
D1
SET R10 VECTOR ADDRESS TO 'FASTER'
SETREAL
E1
IS THE PAGE SHARED OR NOT IN CORE
YES
NO
F1
SET REAL STORAGE KEY WITH REFERENCE AND CHANGE BITS ZERO
G1
IS VM RUNNING A SHARED SYSTEM
NO
YES
H1
ATTEMPT TO SET REAL KEY ZERO
NO
YES
J1
SET REAL STORAGE KEY = X'F0'
K1
VECTOR BRANCH - R10
CHKTRA1
A3
CHKTRA1 (R4)
CHECK WHETHER TO CALL TRACING SUPERVISOR:
C3
TRACING PRIV OPS WANTED
YES
NO
D3
IF NOT, IS THIS AN LPSW
NO
YES
D4
RETURN TO OFFTRAKL
E3
IS INSTRUCTION TRACING ALREADY SET
YES
NO
E4
RETURN TO OFFTRAKL
F3
LPSW ALREADY HANDLED BY DMKTRC
YES
NO
F4
RETURN TO OFFTRAKL
CHKTRA2
G3
CHKTRA2 (R4)
CALL TRACING SUPERVISOR (UNLESS ALREADY CALLED):
J3
WAS THIS AN EXECUTED INSTR
YES
NO
J4
RETURN TO OFFTRAKL OR OFFTRAKH
A5
DMKTRCPV
CALL - TRACE PRIVILEGED INSTRUCTION
B5
RETURN TO OFFTRAKL OR OFFTRAKH
REGSPEC
A2
REGSPEC
B2
COMPUTE INDICES INTO 'VMGPRS' FIELD IN VMBLOK FOR ACCESSING REGISTER VALUES
C2
RETURN - R9
DMKPRVKY
A3
DMKPRVKY
B3
GET FOUR-BIT REAL STORAGE KEY VIA 'ISK'
C3
IS REAL KEY = ZERO
NO
YES
E3
D3
IS VM RUNNING SHARED SYSTEM
YES
NO
E3
PROTKEY
E3
GET FOUR-BIT KEY FROM VIRTUAL PSW
F3
IS PSW KEY = ZERO
YES
NO
G3
COMPARE PSW KEY TO REAL STORAGE KEY
H3
RETURN WITH CONDITION CODE SET - R14

| DMKPRV -- Simulate Virtual Machine Privileged Instructicn (Parts 3 and 4 of 10)

| DMKPRV -- Simulate Virtual Machine Privileged Instruction (Parts 5 and 6 of 10)

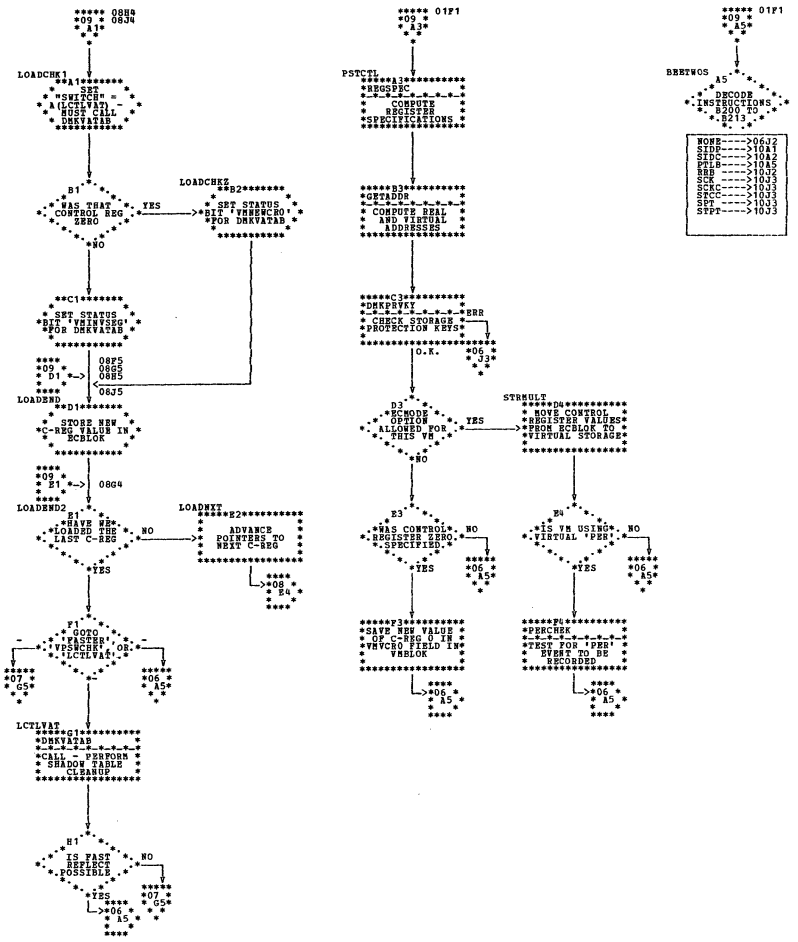

A5
PERCH03
A5 IS ADDRESS BELOW LOW LIMIT
YES
NO
B5 IS ADDRESS ABOVE HIGH LIMIT
YES
NO
C5 IS END OF FIELD ABOVE LIMIT
NO
H3
YES
PERCHEK
A3 PROGRAM EVENT DETECTION
B3 IS THE VM USING 'PER'
NO
H3
YES
C3 IS THIS EVENT CLASS ENABLED
NO
H3
YES
D3 EVENT CLASS = REG ALTER
NO
YES
PERCH01
D2 EVENT CLASS = STORAGE ALTER
NO
H3
YES
E2 DOES ADDRESS RANGE WRAP STORAGE
YES
A5
NO
E3 IS SPECIFIED REGISTER MASK BIT SET ONE
NO
H3
YES
F3
PERCH02
F1 IS AFFECTED ADDRESS BELOW TOP LIMIT
YES
F3
NO
HIGH
F2 COMPARE AFFECTED ADDRESS TO LOW LIMIT
EQ
A
LOW
PERDONE
F3 IS A PER EVENT ALREADY PENDING
NO
YES
PERDON1
F4 RECORD EVENT ADDRESS AND CLASS CODE IN ECBLOK
G2 DID END OF STORAGE FIELD HIT RANGE
YES
NO
G3 ADD THIS EVENT CLASS TO INTERRUPT CLASS CODE
G4 SET 'PER' EVENT PENDING FLAG IN VMBLOK
H3 RETURN - R9
H3

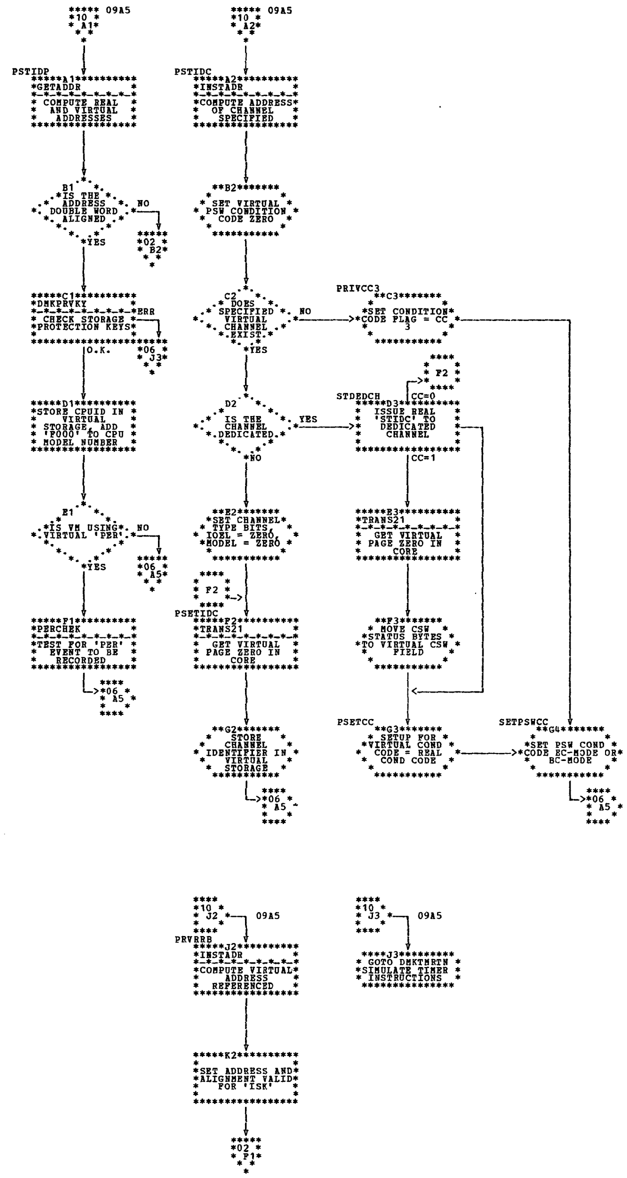

GETADDR
A1 COMPUTE AND TRANSLATE DATA ADDRESSES
B1 INSTADR
COMPUTE VIRTUAL ADDRESS SPECIFIED
C1 ADTRANS
TRANSLATE VIRTUAL ADDRESS TO REAL ADDRESS
D1 RETURN - R9
INSTADR
A2 COMPUTE VIRTUAL ADDRESS FROM INSTRUCTION
B2 ADD INSTRUCTN DISPLACEMENT TO VALUE IN BASE REGISTER
C2 IS ADDRESS INDEXING ALLOWED
NO
YES
D2 ADD INDEX REGISTER VALUE TO VIRTUAL ADDRESS
E2 RETURN - R8
ADTRANS
A3 TRANSLATE VIRTUAL ADDRESS TO REAL ADDRESS
B3 IS THE VM IN EXTENDED CONTROL MODE
NO
YES
C3 IS THE VM IN VIRTUAL TRANSLATE MODE
YES
NO
CALLVAT
C4 TRY HARWARE TRANSLATE VIA 'LRA' ON SHADOW TABLES
O.K.
G3
SIMPTRN
D3 IS THE VIRTUAL ADDRESS VALID
NO
02 F2
YES
D4 DMKVATRN
'CALL' DMKVATRN TO TRANSLATE TO REAL ADDRESS
E3 IS THE VIRTUAL PAGE IN REAL STORAGE
YES
NO
E4 WAS TRANSLATION SUCCESSFUL
YES
NO
F3 DMKPTRAN
'CALL' DMKPTRAN TO BRING IN VIRTUAL PAGE
F4 BACK UP VIRTUAL PSW TO START OF INSTRUCTION BEING SIMULATED
G3 RETURN - R8
G3
G4 GOTO DMKVATEX TRANSLATION EXCEPTION
06 A5
07F2
07F4
08D1
08E2
08E3
08F2
08F3
FNOTE
FASTER
A5 IS A 'PER' EVENT PENDING
NO
02 D4
YES
02 E3
06 J2
01F1
01G3
09A5
OPEREXCP
J2 SET PROGRAM INTERRUPT CODE FOR OPERATION EXCEPTION
02 E3
06 J3
07C2
09C3
10C1
PROTEXCP
J3 SET PROGRAM INTERRUPT CODE FOR PROTECTION EXCEPTION
02 E3
TO:A5
09E3
09E4
09F1
09F3
09F4
09H1
10B5
10E1
10F1
10G2
10G4

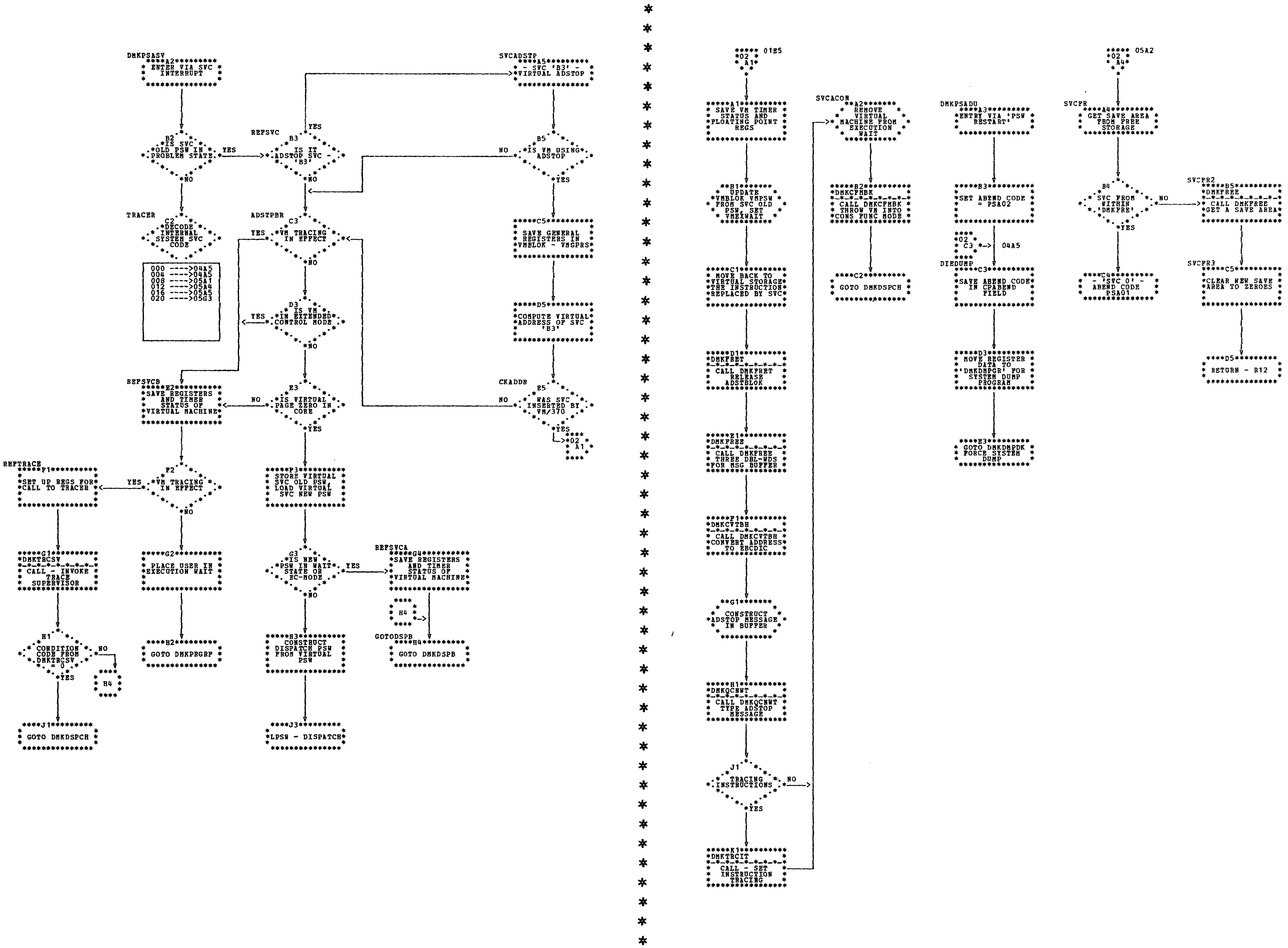

01G3
07 A1
PRSSM
A1 IS SSM SUPPRESSION BIT = 1
NO
YES
SPECIAL1
B1 SET PROGRAM INTERRUPT CODE TO X'13'
02 E3
01F1
07 A2
PSTSSM
A2 GETADDR COMPUTE DATA ADDRESS OF INSTRUCTION
B2 IS THIS A PLAIN 'SSM'
YES
F4
NO
C2 DMKPRVKY CHECK STORAGE PROTECTION KEYS
ERR
06 J3
O.K.
D2 STORE CURRENT SYSTEM MASK IN VIRTUAL STORAGE
E2 IS VM USING VIRTUAL 'PER'
NO
YES
F2 PERCHEK TEST FOR 'PER' EVENT TO BE RECORDED
06 A5
PSTNPER
E3 'STORE + OR' OR 'STORE + AND'
AND
OR
F3 'OR' IMMEDIATE BYTE OF INSTRUCTION INTO SYSTEM MASK
PSTANDSM
E4 'AND' IMMEDIATE BYTE OF INSTRUCTION INTO SYSTEM MASK
F4
SETMASK
F4 HAS SYSTEM MASK CHANGED
NO
06 A5
YES
G4 SET NEW SYSTEM MASK IN VMPSW FIELD OF VMBLOK
07 G5
08D1
09F1
09H1
VPSWCHK
G5 REMOVE VIRTUAL MACHINE FROM EXECUTION WAIT
H5 GOTO DMKDSPB 'NEW PSW' ENTRY
01G3
08 A1
PRLPSW
A1 GETADDR COMPUTE DATA ADDRESS OF INSTRUCTION
B1 IS DATA ON A DOUBLE-WORD BOUNDARY
NO
02 B2
YES
C1 MOVE NEW PSW TO VMPSW FIELD IN VMBLOK
D1 HAS ADDRESS ONLY CHANGED
NO
07 G5
YES
06 A5
01F1
08 A2
PRLRA
A2 INSTADR COMPUTE VIRTUAL ADDRESS FROM INSTRUCTION
B2 DMKVATLA 'CALL' DMKVATLA TO TRANSLATE TO 'REAL' ADDRESS
C2 PASS TRANSLATED ADDRESS BACK TO VIRTUAL MACHINE IN SPECIFIED REGISTER
D2 SET VIRTUAL CONDITION CODE ACCORDING TO RETURN CODES FROM DMKVATLA
E2 IS VM USING VIRTUAL 'PER'
NO
06 A5
YES
F2 PERCHEK TEST FOR 'PER' EVENT TO BE RECORDED
06 A5
01F1
08 A3
PRLCTL
A3 GETADDR GET REAL AND VIRTUAL ADDRESSES
B3 IS DATA ON FULL WORD BOUNDARY
NO
02 B2
YES
C3 REGSPEC COMPUTE REGISTER SPECIFICATIONS
D3 IS VM ALLOWED EC-MODE
YES
A4
NO
E3 LOADING VIRTUAL C-REG. 0
NO
06 A5
YES
F3 PLACE NEW C-REG 0 VALUE IN VMBLOK
06 A5
A4
LOADECR
A4 SET REGISTER "SWITCH" TO GO TO 'FASTER'
B4 LOADING MULTIPLE C-REGS
NO
YES
LOADCRS
C4 COMPUTE LENGTH OF DATA FIELD REFERENCED
D4 ADTRANS CHECK PAGE AVAILABILITY
08 E4
09E2
LOADMLT
E4 ADTRANS TRANSLATE DATA ADDRESS TO REAL ADDRESS
LOADCHK
F4 LOADING C-REG 0 OR 1
NO
YES
G4 C-REG 0 OR 1 SAME AS BEFORE
YES
09 E1
NO
H4 TRACING INSTRUCTIONS
NO
09 A1
YES
J4 DMKTRCPB CALL - PUT BACK OLD USER INSTRUCTIONS
09 A1
LOADATA
F5 WAS C-REG 2 JUST LOADED
NO
09 D1
YES
G5 IS VM VIRTUAL PSW ENABLED FOR I/O
NO
09 D1
YES
H5 SWITCH ALREADY SET TO CALL DMKVATAB
YES
09 D1
NO
J5 SET "SWITCH" TO GO TO 'VPSWCHK'
09 D1

| DMKPRV -- Simulate Virtual Machine Privileged Instruction (Parts 7 and 8 of 10)

| DMKPRV -- Simulate Virtual Machine Privileged Instruction (Parts 9 and 10 of 10)

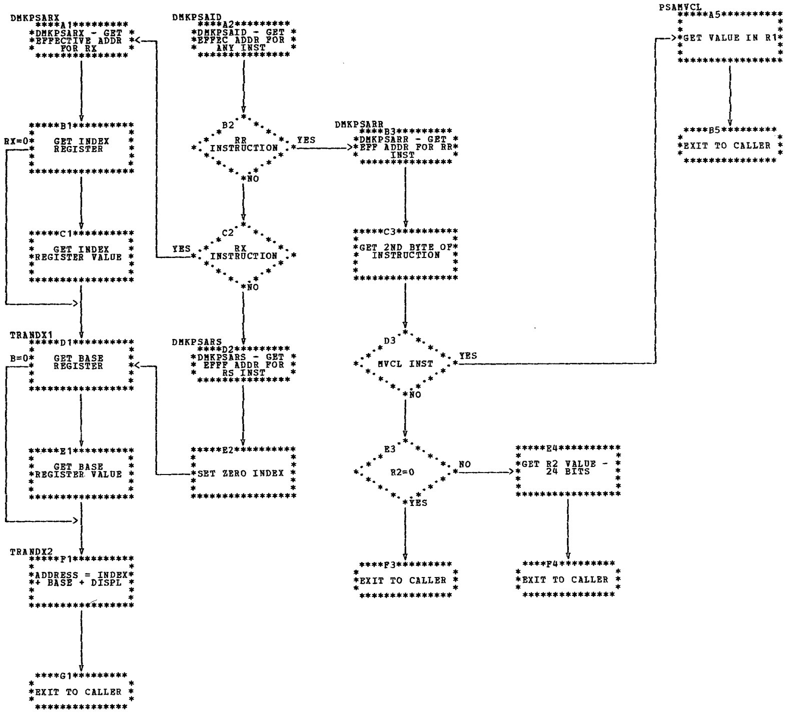

09 A1 08H4 08J4
LOADCHK1
A1 SET "SWITCH" = A(LCTLVAT) - MUST CALL DMKVATAB
B1 WAS THAT CONTROL REG ZERO
YES
NO
LOADCHKZ
B2 SET STATUS BIT 'VMNEWCRO' FOR DMKVATAB
C1 SET STATUS BIT 'VMINVSEG' FOR DMKVATAB
09 D1 08F5 08G5 08H5 08J5
LOADEND
D1 STORE NEW C-REG VALUE IN ECBLOK
09 E1 08G4
LOADEND2
E1 HAVE WE LOADED THE LAST C-REG
NO
YES
LOADNXT
E2 ADVANCE POINTERS TO NEXT C-REG
08 E4
F1 GOTO 'FASTER', 'VPSWCHK', OR 'LCTLVAT'
07 G5
06 A5
LCTLVAT
G1 DMKVATAB CALL - PERFORM SHADOW TABLE CLEANUP
H1 IS FAST REFLECT POSSIBLE
NO
YES
06 A5
07 G5
09 A3 01F1
PSTCTL
A3 REGSPEC COMPUTE REGISTER SPECIFICATIONS
B3 GETADDR COMPUTE REAL AND VIRTUAL ADDRESSES
C3 DMKPRVKY CHECK STORAGE PROTECTION KEYS
ERR
06 J3
O.K.
D3 ECMODE OPTION ALLOWED FOR THIS VM
YES
NO
STRMULT
D4 MOVE CONTROL REGISTER VALUES FROM ECBLOK TO VIRTUAL STORAGE
E3 WAS CONTROL REGISTER ZERO SPECIFIED
NO
06 A5
YES
F3 SAVE NEW VALUE OF C-REG 0 IN VMVCRO FIELD IN VMBLOK
06 A5
E4 IS VM USING VIRTUAL 'PER'
NO
06 A5
YES
F4 PERCHEK TEST FOR 'PER' EVENT TO BE RECORDED
06 A5
09 A5 01F1
BEETWOS
A5 DECODE INSTRUCTIONS B200 TO B213
NONE---->06J2
SIDP---->10A1
SIDC---->10A2
PTLB---->10A5
RRB ---->10J2
SCK ---->10J3
SCKC---->10J3
STCC---->10J3
SPT ---->10J3
STPT---->10J3

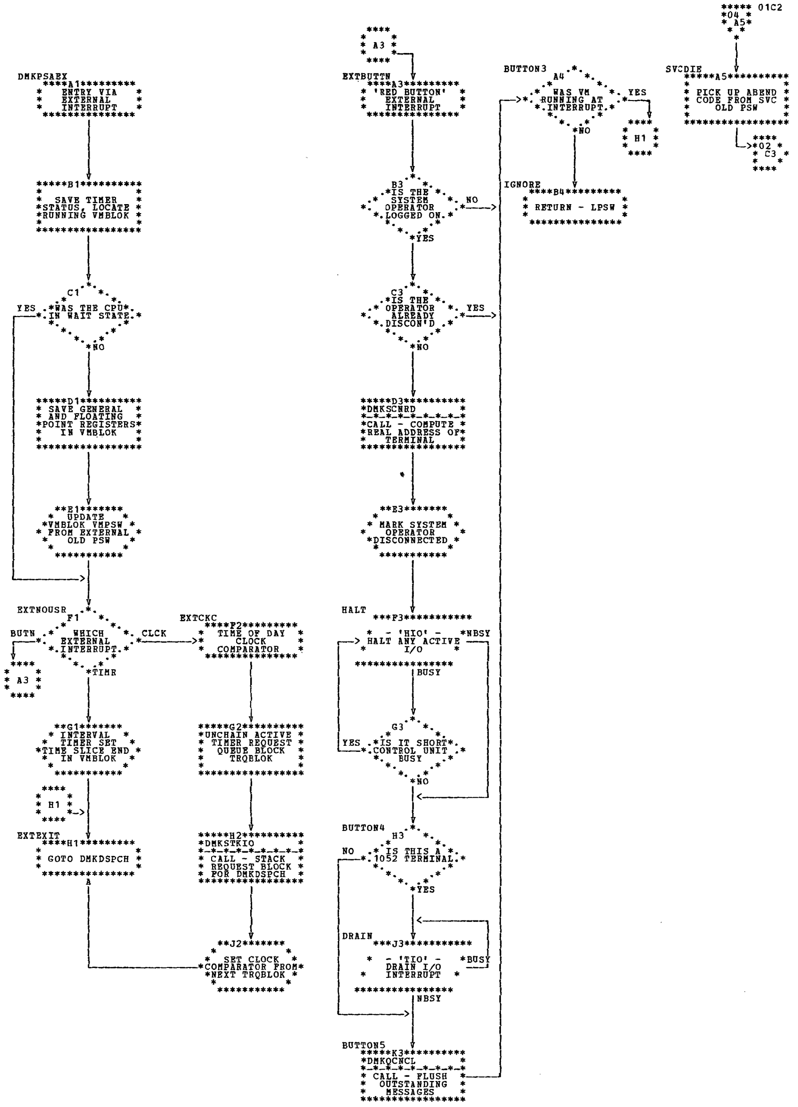

10 A1 09A5
PSTIDP
A1 GETADDR COMPUTE REAL AND VIRTUAL ADDRESSES
B1 IS THE ADDRESS DOUBLE WORD ALIGNED
NO
02 B2
YES
C1 DMKPRVKY CHECK STORAGE PROTECTION KEYS
ERR
06 J3
O.K.
D1 STORE CPUID IN VIRTUAL STORAGE, ADD 'F000' TO CPU MODEL NUMBER
E1 IS VM USING VIRTUAL 'PER'
NO
06 A5
YES
F1 PERCHEK TEST FOR 'PER' EVENT TO BE RECORDED
06 A5
10 A2 09A5
PSTIDC
A2 INSTADR COMPUTE ADDRESS OF CHANNEL SPECIFIED
B2 SET VIRTUAL PSW CONDITION CODE ZERO
C2 DOES SPECIFIED VIRTUAL CHANNEL EXIST
NO
YES
PRIVCC3
C3 SET CONDITION CODE FLAG = CC 3
D2 IS THE CHANNEL DEDICATED
YES
NO
F2
STDEDCH
CC=0
D3 ISSUE REAL 'STIDC' TO DEDICATED CHANNEL
CC=1
E2 SET CHANNEL TYPE BITS, IOEL = ZERO, MODEL = ZERO
F2
PSETIDC
F2 TRANS21 GET VIRTUAL PAGE ZERO IN CORE
G2 STORE CHANNEL IDENTIFIER IN VIRTUAL STORAGE
06 A5
E3 TRANS21 GET VIRTUAL PAGE ZERO IN CORE
F3 MOVE CSW STATUS BYTES TO VIRTUAL CSW FIELD
PSETCC
G3 SETUP FOR VIRTUAL COND CODE = REAL COND CODE
SETPSWCC
G4 SET PSW COND CODE EC-MODE OR BC-MODE
06 A5
10 A5 09A5
PRPTLB
A5 SET STATUS BITS IN VMESTAT TO FORCE INVALIDATION OF SHADOW TABLES
B5 DMKVATAB 'CALL' DMKVATAB TO CLEAN UP SHADOW TABLES
06 A5
10 J2 09A5
PRVRRB
J2 INSTADR COMPUTE VIRTUAL ADDRESS REFERENCED
K2 SET ADDRESS AND ALIGNMENT VALID FOR 'ISK'
02 F1
10 J3 09A5
J3 GOTO DMKTMRTN SIMULATE TIMER INSTRUCTIONS

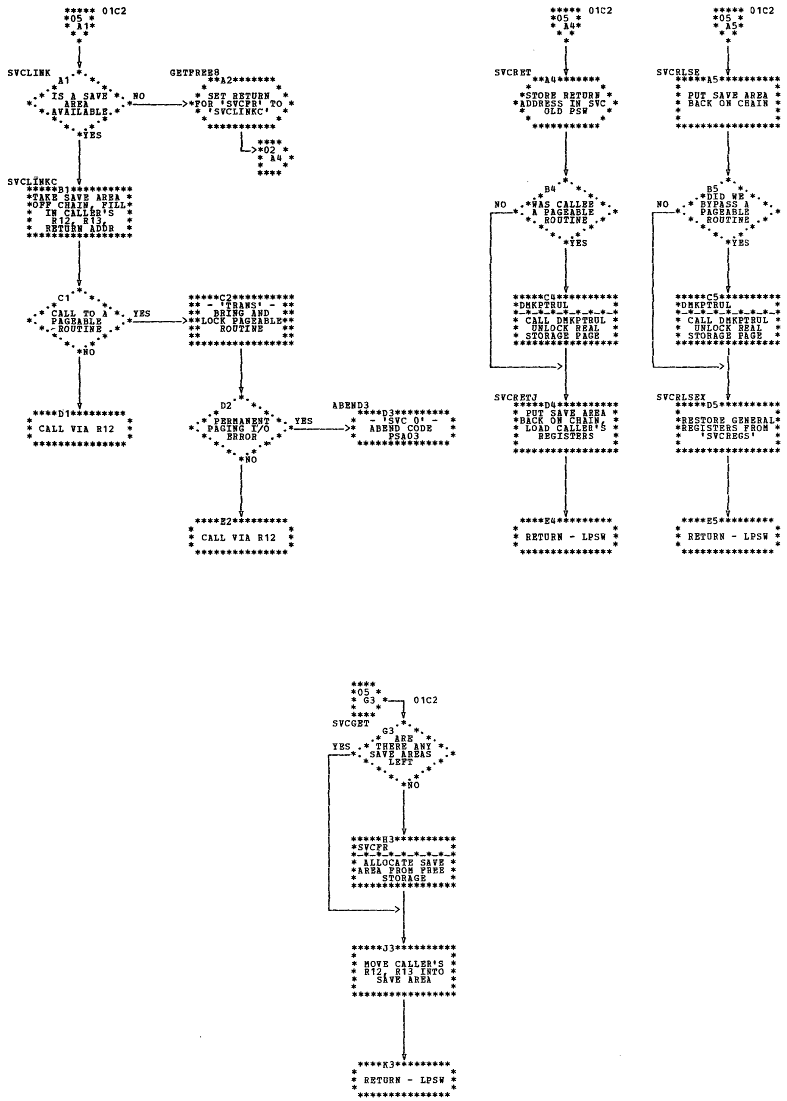

DMKPSASV
A2
ENTER VIA SVC INTERRUPT
B2
IS SVC OLD PSW IN PROBLEM STATE
YES
NO
TRACER
C2
DECODE INTERNAL SYSTEM SVC CODE
000 ---->04A5
004 ---->04A5
008 ---->05A1
012 ---->05A4
016 ---->05A5
020 ---->05G3
REFSVC
B3
IS IT ADSTOP SVC - 'B3'
YES
NO
ADSTPBR
C3
VM TRACING IN EFFECT
YES
NO
D3
IS VM IN EXTENDED CONTROL MODE
YES
NO
E3
IS VIRTUAL PAGE ZERO IN CORE
NO
YES
REFSVCB
E2
SAVE REGISTERS AND TIMER STATUS OF VIRTUAL MACHINE
SVCADSTP
A5
- SVC 'B3' - VIRTUAL ADSTOP
B5
IS VM USING ADSTOP
NO
YES
C5
SAVE GENERAL REGISTERS IN VMBLOK - VMGPRS
D5
COMPUTE VIRTUAL ADDRESS OF SVC 'B3'
CKADDR
E5
WAS SVC INSERTED BY VM/370
NO
YES
02 A1
REFTRACE
F1
SET UP REGS FOR CALL TO TRACER
F2
VM TRACING IN EFFECT
YES
NO
F3
STORE VIRTUAL SVC OLD PSW, LOAD VIRTUAL SVC NEW PSW
G1
DMKTRCSV
CALL - INVOKE TRACE SUPERVISOR
G2
PLACE USER IN EXECUTION WAIT
G3
IS NEW PSW IN WAIT STATE OR EC-MODE
YES
NO
REFSVCA
G4
SAVE REGISTERS AND TIMER STATUS OF VIRTUAL MACHINE
H4
H1
CONDITION CODE FROM DMKTRCSV = 0
NO
YES
H4
H2
GOTO DMKPRGRF
H3
CONSTRUCT DISPATCH PSW FROM VIRTUAL PSW
GOTODSPB
H4
GOTO DMKDSPB
J1
GOTO DMKDSPCH
J3
LPSW - DISPATCH
01E5
02 A1
A1
SAVE VM TIMER STATUS AND FLOATING POINT REGS
B1
UPDATE VMBLOK VMPSW FROM SVC OLD PSW, SET VMEXWAIT
C1
MOVE BACK TO VIRTUAL STORAGE THE INSTRUCTION REPLACED BY SVC
D1
DMKFRET
CALL DMKFRET RELEASE ADSTBLOK
E1
DMKFREE
CALL DMKFREE THREE DBL-WDS FOR MSG BUFFER
F1
DMKCVTBH
CALL DMKCVTBH CONVERT ADDRESS TO EBCDIC
G1
CONSTRUCT ADSTOP MESSAGE IN BUFFER
H1
DMKQCNWT
CALL DMKQCNWT TYPE ADSTOP MESSAGE
J1
TRACING INSTRUCTIONS
NO
YES
K1
DMKTRCIT
CALL - SET INSTRUCTION TRACING
SVCACON
A2
REMOVE VIRTUAL MACHINE FROM EXECUTION WAIT
B2
DMKCFMBK
CALL DMKCFMBK THROW VM INTO CONS FUNC MODE
C2
GOTO DMKDSPCH
DMKPSADU
A3
ENTRY VIA 'PSW RESTART'
B3
SET ABEND CODE = PSA02
02 C3
04A5
DIEDUMP
C3
SAVE ABEND CODE IN CPABEND FIELD
D3
MOVE REGISTER DATA TO 'DMKDMPGR' FOR SYSTEM DUMP PROGRAM
E3
GOTO DMKDMPDK FORCE SYSTEM DUMP
05A2
02 A4
SVCFR
A4
GET SAVE AREA FROM FREE STORAGE
B4
SVC FROM WITHIN 'DMKFRE'
NO
YES
C4
- 'SVC 0' - ABEND CODE PSA01
SVCFR2
B5
DMKFREE
CALL DMKFREE GET A SAVE AREA
SVCFR3
C5
CLEAR NEW SAVE AREA TO ZEROES
D5
RETURN - R12

DMKPSA -- SVC and External Interrupt Handler (Parts 1 and 2 of 5)

DMKPSA -- SVC and External Interrupt Handler (Parts 3 and 4 of 5)

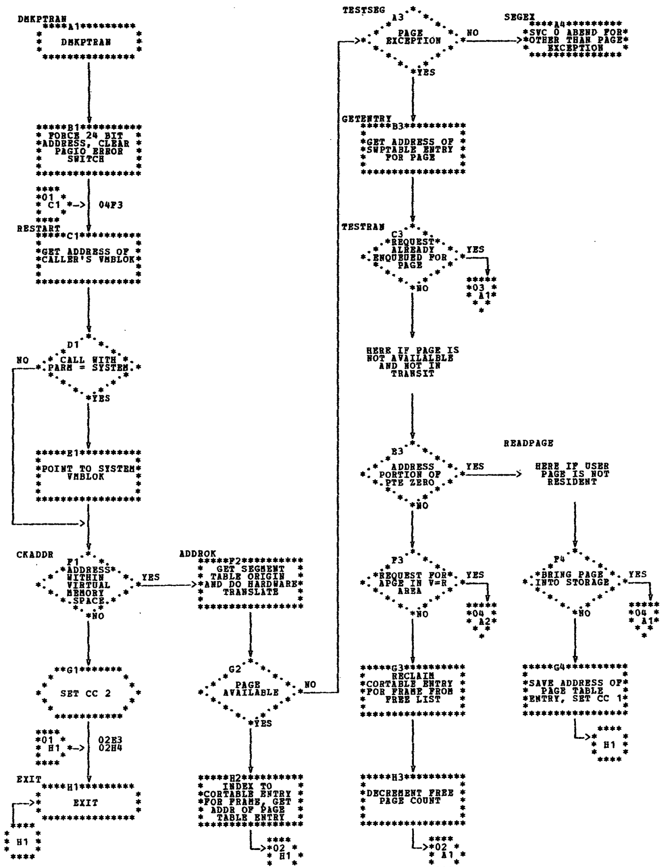


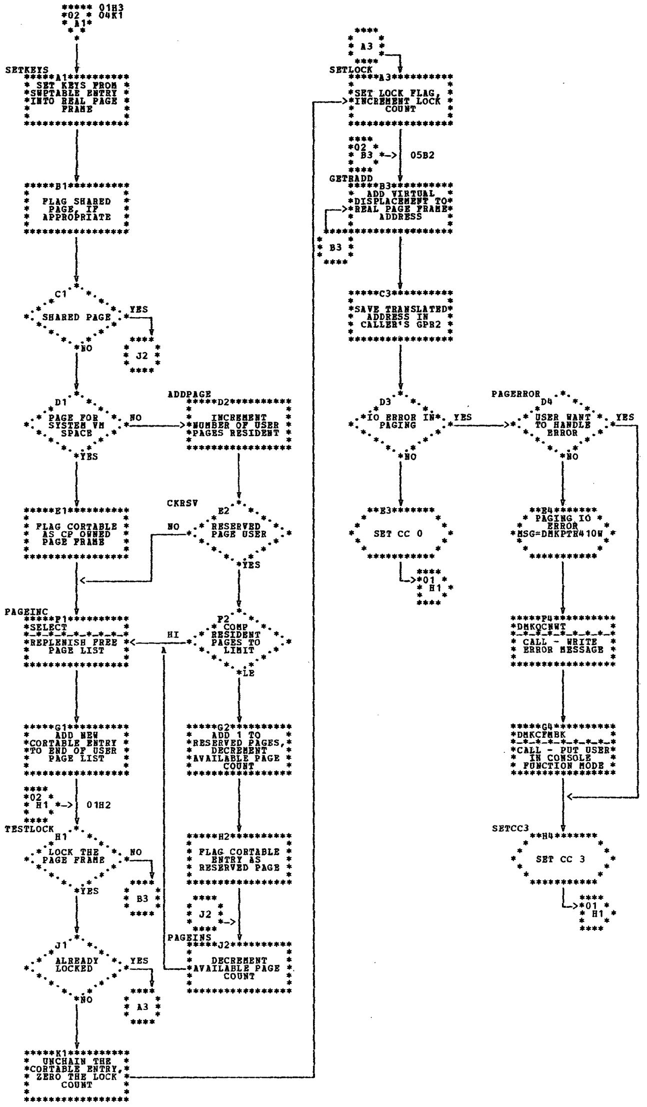

01C2
04
A5
A3
DMKPSAEX
A1
ENTRY VIA EXTERNAL INTERRUPT
EXTBUTTN
A3
'RED BUTTON' EXTERNAL INTERRUPT
BUTTON3
A4
WAS VM RUNNING AT INTERRUPT
YES
H1
NO
SVCDIE
A5
PICK UP ABEND CODE FROM SVC OLD PSW
02
C3
B1
SAVE TIMER STATUS, LOCATE RUNNING VMBLOK
B3
IS THE SYSTEM OPERATOR LOGGED ON
NO
YES
IGNORE
B4
RETURN - LPSW
C1
YES
WAS THE CPU IN WAIT STATE
NO
C3
IS THE OPERATOR ALREADY DISCON'D
YES
NO
D1
SAVE GENERAL AND FLOATING POINT REGISTERS IN VMBLOK
D3
DMKSCNRD
CALL - COMPUTE REAL ADDRESS OF TERMINAL
E1
UPDATE VMBLOK VMPSW FROM EXTERNAL OLD PSW
E3
MARK SYSTEM OPERATOR DISCONNECTED
EXTNOUSR
F1
BUTN
WHICH EXTERNAL INTERRUPT
CLCK
TIMR
A3
EXTCKC
F2
TIME OF DAY CLOCK COMPARATOR
HALT
F3
- 'HIO' - HALT ANY ACTIVE I/O
NBSY
BUSY
G1
INTERVAL TIMER SET TIME SLICE END IN VMBLOK
G2
UNCHAIN ACTIVE TIMER REQUEST QUEUE BLOCK TRQBLOK
G3
YES
IS IT SHORT CONTROL UNIT BUSY
NO
H1
EXTEXIT
H1
GOTO DMKDSPCH
A
H2
DMKSTKIO
CALL - STACK REQUEST BLOCK FOR DMKDSPCH
BUTTON4
H3
NO
IS THIS A 1052 TERMINAL
YES
J2
SET CLOCK COMPARATOR FROM NEXT TRQBLOK
DRAIN
J3
- 'TIO' - DRAIN I/O INTERRUPT
BUSY
NBSY
BUTTON5
K3
DMKQCNCL
CALL - FLUSH OUTSTANDING MESSAGES

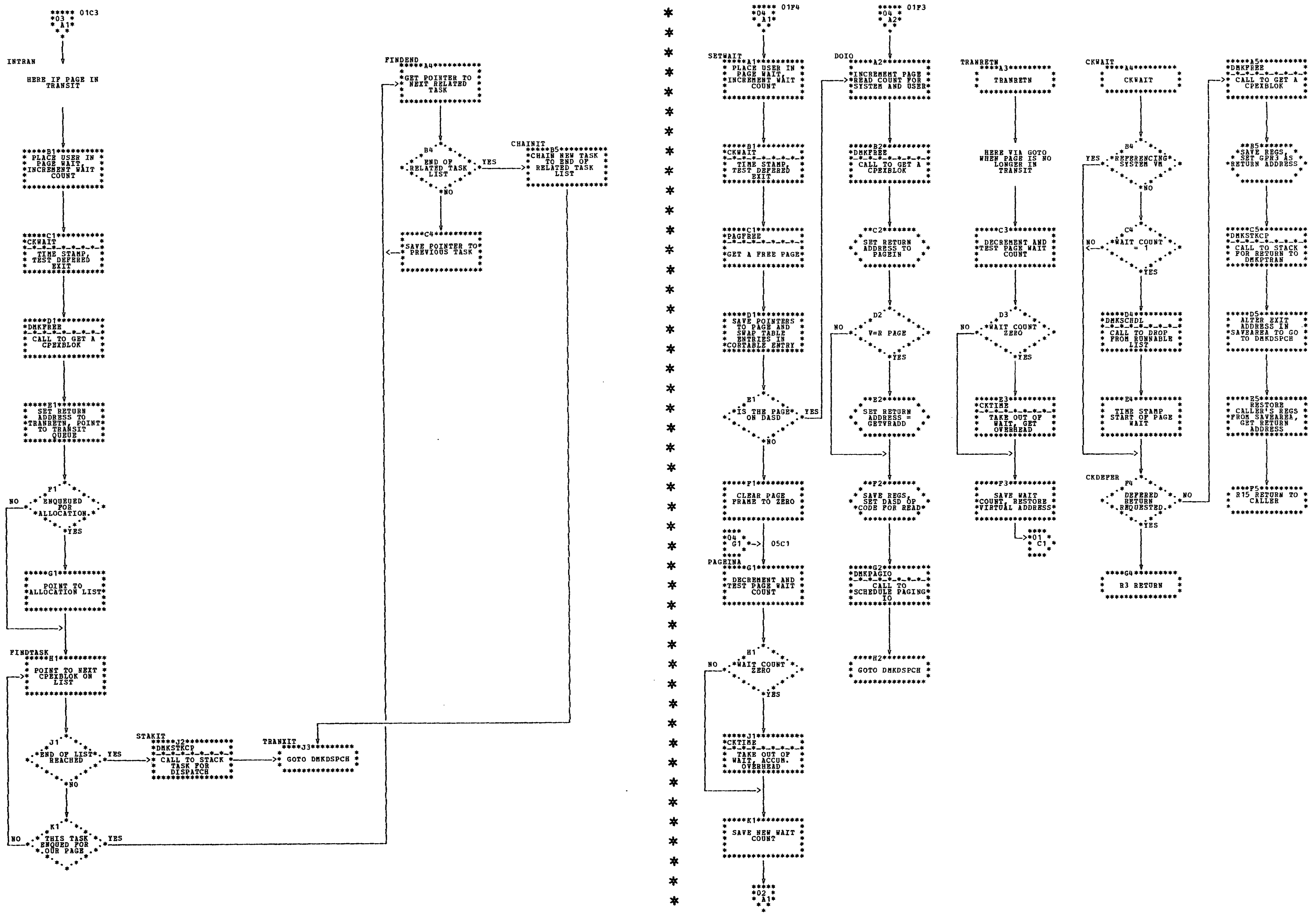

05 A1 01C2
SVCLINK
A1
IS A SAVE AREA AVAILABLE
NO
YES
GETFREE8
A2
SET RETURN FOR 'SVCPR' TO 'SVCLINKC'
02 A4
SVCLINKC
B1
TAKE SAVE AREA OFF CHAIN, FILL IN CALLER'S R12, R13, RETURN ADDR
C1
CALL TO A PAGEABLE ROUTINE
YES
NO
C2
- 'TRANS' - BRING AND LOCK PAGEABLE ROUTINE
D1
CALL VIA R12
D2
PERMANENT PAGING I/O ERROR
YES
NO
ABEND3
D3
- 'SVC 0' - ABEND CODE PSA03
E2
CALL VIA R12
05 A4 01C2
SVCRET
A4
STORE RETURN ADDRESS IN SVC OLD PSW
B4
WAS CALLEE A PAGEABLE ROUTINE
NO
YES
C4
DMKPTRUL
CALL DMKPTRUL UNLOCK REAL STORAGE PAGE
SVCRETJ
D4
PUT SAVE AREA BACK ON CHAIN, LOAD CALLER'S REGISTERS
E4
RETURN - LPSW
05 A5 01C2
SVCRLSE
A5
PUT SAVE AREA BACK ON CHAIN
B5
DID WE BYPASS A PAGEABLE ROUTINE
NO
YES
C5
DMKPTRUL
CALL DMKPTRUL UNLOCK REAL STORAGE PAGE
SVCRLSEX
D5
RESTORE GENERAL REGISTERS FROM 'SVCREGS'
E5
RETURN - LPSW
05 G3 01C2
SVCGET
G3
ARE THERE ANY SAVE AREAS LEFT
YES
NO
H3
SVCFR
ALLOCATE SAVE AREA FROM FREE STORAGE
J3
MOVE CALLER'S R12, R13 INTO SAVE AREA
K3
RETURN - LPSW

DMKPSA -- SVC and External Interrupt Handler (Part 5 of 5)

DMKPTR -- Real Storage Page Manager (Parts 1 and 2 of 13)

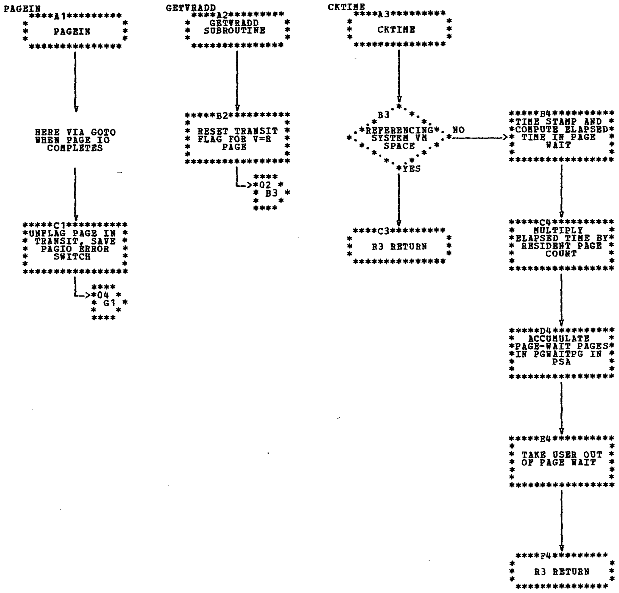

DMKPTRAN
A1
DMKPTRAN
B1
FORCE 24 BIT ADDRESS, CLEAR PAGIO ERROR SWITCH
01 C1
04F3
RESTART
C1
GET ADDRESS OF CALLER'S VMBLOK
D1
CALL WITH PARM = SYSTEM
NO
YES
E1
POINT TO SYSTEM VMBLOK
CKADDR
F1
ADDRESS WITHIN VIRTUAL MEMORY SPACE
YES
NO
G1
SET CC 2
01 H1
02E3
02H4
EXIT
H1
EXIT
H1
ADDROK
F2
GET SEGMENT TABLE ORIGIN AND DO HARDWARE TRANSLATE
G2
PAGE AVAILABLE
NO
YES
H2
INDEX TO CORTABLE ENTRY FOR FRAME, GET ADDR OF PAGE TABLE ENTRY
02 H1
TESTSEG
A3
PAGE EXCEPTION
NO
YES
SEGEX
A4
SVC 0 ABEND FOR OTHER THAN PAGE EXCEPTION
GETENTRY
B3
GET ADDRESS OF SWPTABLE ENTRY FOR PAGE
TESTRAN
C3
REQUEST ALREADY ENQUEUED FOR PAGE
YES
03 A1
NO
HERE IF PAGE IS NOT AVAILALBLE AND NOT IN TRANSIT
E3
ADDRESS PORTION OF PTE ZERO
YES
NO
READPAGE
HERE IF USER PAGE IS NOT RESIDENT
F3
REQUEST FOR APGE IN V=R AREA
YES
04 A2
NO
G3
RECLAIM CORTABLE ENTRY FOR FRAME FROM FREE LIST
H3
DECREMENT FREE PAGE COUNT
02 A1
F4
BRING PAGE INTO STORAGE
YES
04 A1
NO
G4
SAVE ADDRESS OF PAGE TABLE ENTRY, SET CC 1
H1

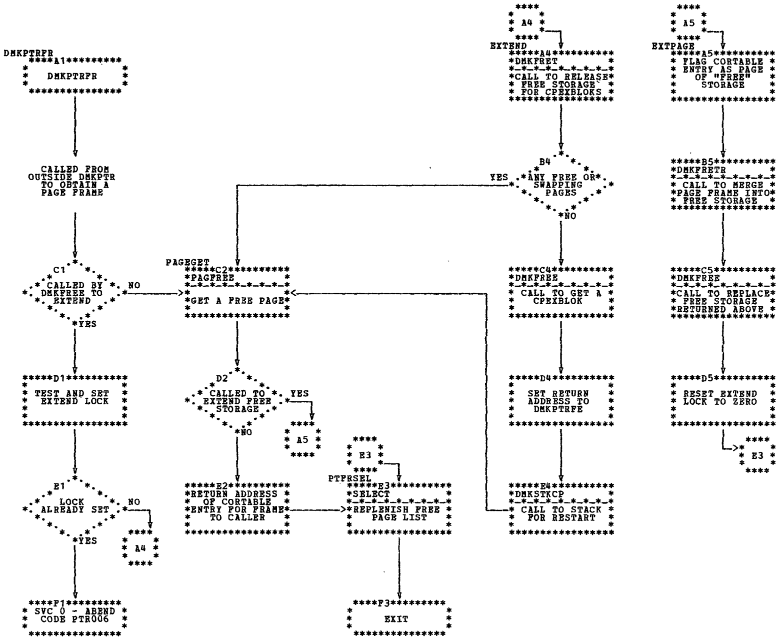

02 A1
01H3
04K1
SETKEYS
A1
SET KEYS FROM SWPTABLE ENTRY INTO REAL PAGE FRAME
B1
FLAG SHARED PAGE, IF APPROPRIATE
C1
SHARED PAGE
YES
J2
NO
D1
PAGE FOR SYSTEM VM SPACE
NO
YES
E1
FLAG CORTABLE AS CP OWNED PAGE FRAME
PAGEINC
F1
SELECT
REPLENISH FREE PAGE LIST
G1
ADD NEW CORTABLE ENTRY TO END OF USER PAGE LIST
02 H1
01H2
TESTLOCK
H1
LOCK THE PAGE FRAME
NO
B3
YES
J1
ALREADY LOCKED
YES
A3
NO
K1
UNCHAIN THE CORTABLE ENTRY, ZERO THE LOCK COUNT
ADDPAGE
D2
INCREMENT NUMBER OF USER PAGES RESIDENT
CKRSV
E2
RESERVED PAGE USER
NO
YES
F2
COMP RESIDENT PAGES TO LIMIT
HI
LE
G2
ADD 1 TO RESERVED PAGES, DECREMENT AVAILABLE PAGE COUNT
H2
FLAG CORTABLE ENTRY AS RESERVED PAGE
J2
PAGEINS
J2
DECREMENT AVAILABLE PAGE COUNT
A3
SETLOCK
A3
SET LOCK FLAG, INCREMENT LOCK COUNT
02 B3
05B2
GETRADD
B3
ADD VIRTUAL DISPLACEMENT TO REAL PAGE FRAME ADDRESS
B3
C3
SAVE TRANSLATED ADDRESS IN CALLER'S GPR2
D3
IO ERROR IN PAGING
YES
NO
E3
SET CC 0
01 H1
PAGERROR
D4
USER WANT TO HANDLE ERROR
YES
NO
E4
PAGING IO ERROR
MSG=DMKPTR410W
F4
DMKQCNWT
CALL - WRITE ERROR MESSAGE
G4
DMKCFMBK
CALL - PUT USER IN CONSOLE FUNCTION MODE
SETCC3
H4
SET CC 3
01 H1

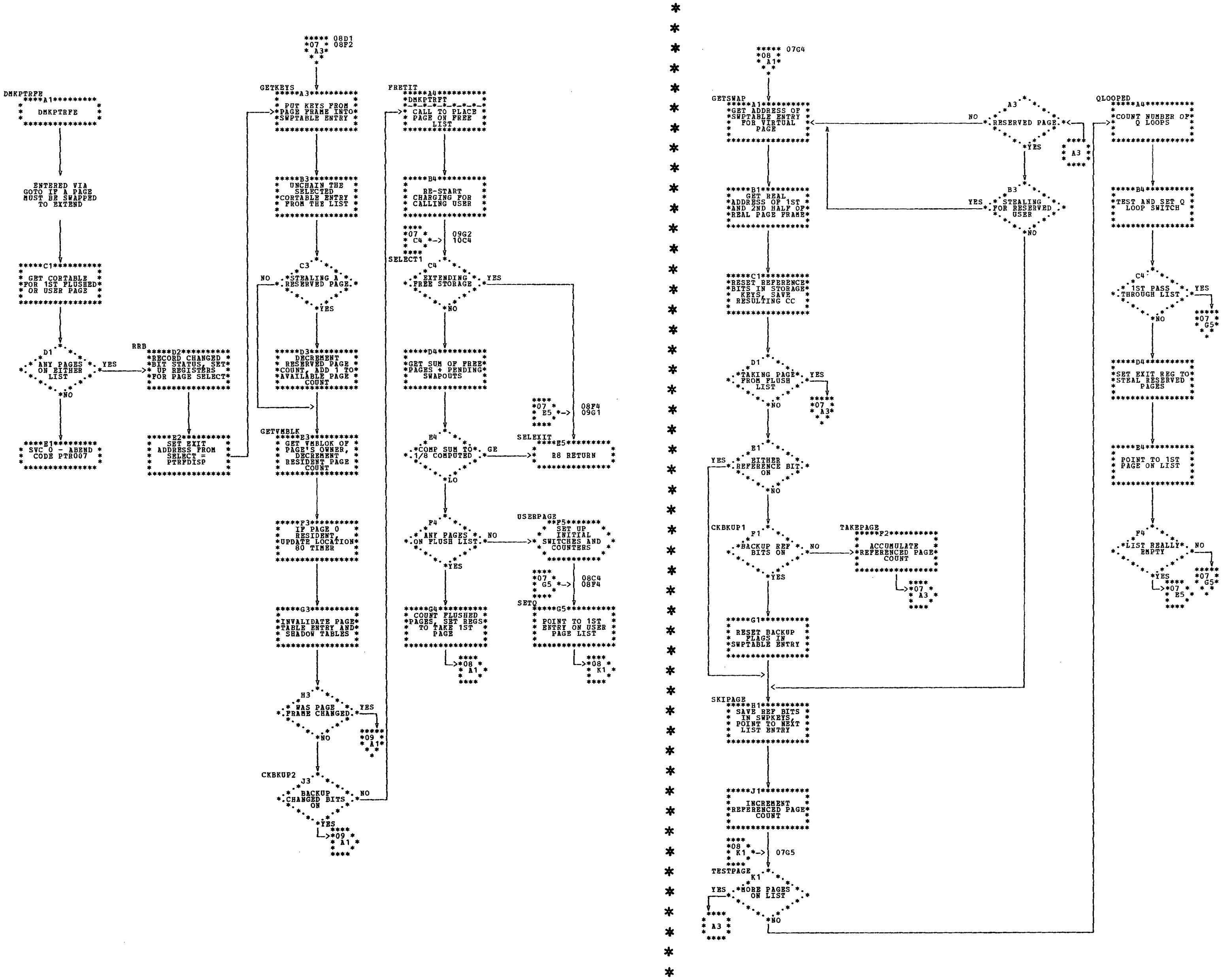

01C3
03 A1
INTRAN
HERE IF PAGE IN TRANSIT
B1 PLACE USER IN PAGE WAIT, INCREMENT WAIT COUNT
C1 CKWAIT TIME STAMP, TEST DEFERED EXIT
D1 DMKFREE CALL TO GET A CPEXBLOK
E1 SET RETURN ADDRESS TO TRANRETN, POINT TO TRANSIT QUEUE
F1 ENQUEUED FOR ALLOCATION
NO
YES
G1 POINT TO ALLOCATION LIST
FINDTASK
H1 POINT TO NEXT CPEXBLOK ON LIST
J1 END OF LIST REACHED
YES
NO
STAKIT
J2 DMKSTKCP CALL TO STACK TASK FOR DISPATCH
TRANXIT
J3 GOTO DMKDSPCH
K1 THIS TASK ENQUED FOR OUR PAGE
NO
YES
FINDEND
A4 GET POINTER TO NEXT RELATED TASK
B4 END OF RELATED TASK LIST
YES
NO
CHAINIT
B5 CHAIN NEW TASK TO END OF RELATED TASK LIST
C4 SAVE POINTER TO PREVIOUS TASK
01F4
04 A1
SETWAIT
A1 PLACE USER IN PAGE WAIT, INCREMENT WAIT COUNT
B1 CKWAIT TIME STAMP, TEST DEFERED EXIT
C1 PAGFREE GET A FREE PAGE
D1 SAVE POINTERS TO PAGE AND SWAP TABLE ENTRIES IN CORTABLE ENTRY
E1 IS THE PAGE ON DASD
YES
NO
F1 CLEAR PAGE FRAME TO ZERO
04 G1
05C1
PAGEINA
G1 DECREMENT AND TEST PAGE WAIT COUNT
H1 WAIT COUNT ZERO
NO
YES
J1 CKTIME TAKE OUT OF WAIT, ACCUM. OVERHEAD
K1 SAVE NEW WAIT COUNT
02 A1
01F3
04 A2
DOIO
A2 INCREMENT PAGE READ COUNT FOR SYSTEM AND USER
B2 DMKFREE CALL TO GET A CPEXBLOK
C2 SET RETURN ADDRESS TO PAGEIN
D2 V=R PAGE
NO
YES
E2 SET RETURN ADDRESS = GETVRADD
F2 SAVE REGS, SET DASD OP CODE FOR READ
G2 DMKPAGIO CALL TO SCHEDULE PAGING IO
H2 GOTO DMKDSPCH
TRANRETN
A3 TRANRETN
HERE VIA GOTO WHEN PAGE IS NO LONGER IN TRANSIT
C3 DECREMENT AND TEST PAGE WAIT COUNT
D3 WAIT COUNT ZERO
NO
YES
E3 CKTIME TAKE OUT OF WAIT, GET OVERHEAD
F3 SAVE WAIT COUNT, RESTORE VIRTUAL ADDRESS
01 C1
CKWAIT
A4 CKWAIT
B4 REFERENCING SYSTEM VM
YES
NO
C4 WAIT COUNT = 1
NO
YES
D4 DMKSCHDL CALL TO DROP FROM RUNNABLE LIST
E4 TIME STAMP START OF PAGE WAIT
CKDEFER
F4 DEFERED RETURN REQUESTED
NO
YES
G4 R3 RETURN
A5 DMKFREE CALL TO GET A CPEXBLOK
B5 SAVE REGS, SET GPR3 AS RETURN ADDRESS
C5 DMKSTKCP CALL TO STACK FOR RETURN TO DMKPTRAN
D5 ALTER EXIT ADDRESS IN SAVEAREA TO GO TO DMKDSPCH
E5 RESTORE CALLER'S REGS FROM SAVEAREA, GET RETURN ADDRESS
F5 R15 RETURN TO CALLER

DMKPTR -- Real Storage Page Manager (Parts 3 and 4 of 13)

DMKPTR -- Real Storage Page Manager (Parts 5 and 6 of 13)

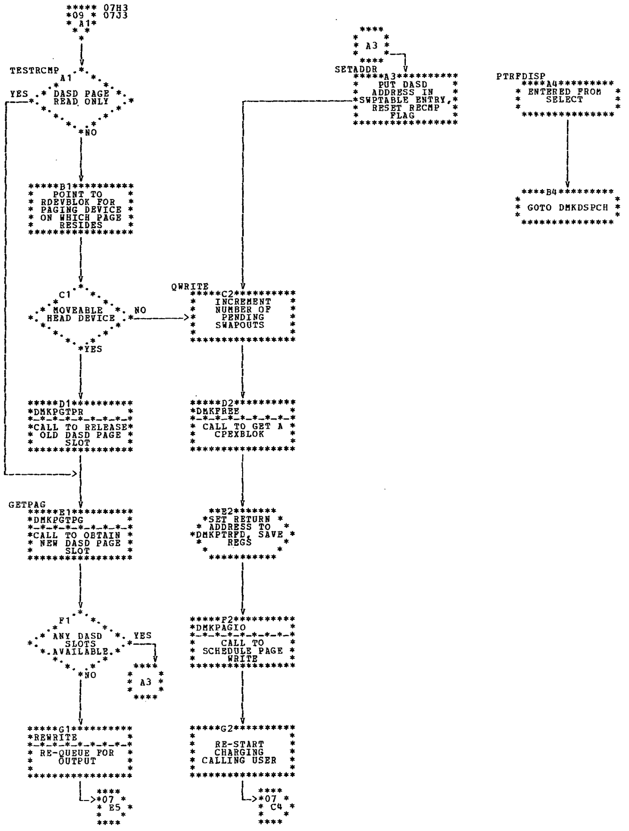

PAGEIN
A1
PAGEIN
HERE VIA GOTO WHEN PAGE IO COMPLETES
C1
UNFLAG PAGE IN TRANSIT, SAVE PAGIO ERROR SWITCH
04 G1
GETVRADD
A2
GETVRADD SUBROUTINE
B2
RESET TRANSIT FLAG FOR V=R PAGE
02 B3
CKTIME
A3
CKTIME
B3
REFERENCING SYSTEM VM SPACE
NO
YES
C3
R3 RETURN
B4
TIME STAMP AND COMPUTE ELAPSED TIME IN PAGE WAIT
C4
MULTIPLY ELAPSED TIME BY RESIDENT PAGE COUNT
D4
ACCUMULATE PAGE-WAIT PAGES IN PGWAITPG IN PSA
E4
TAKE USER OUT OF PAGE WAIT
F4
R3 RETURN

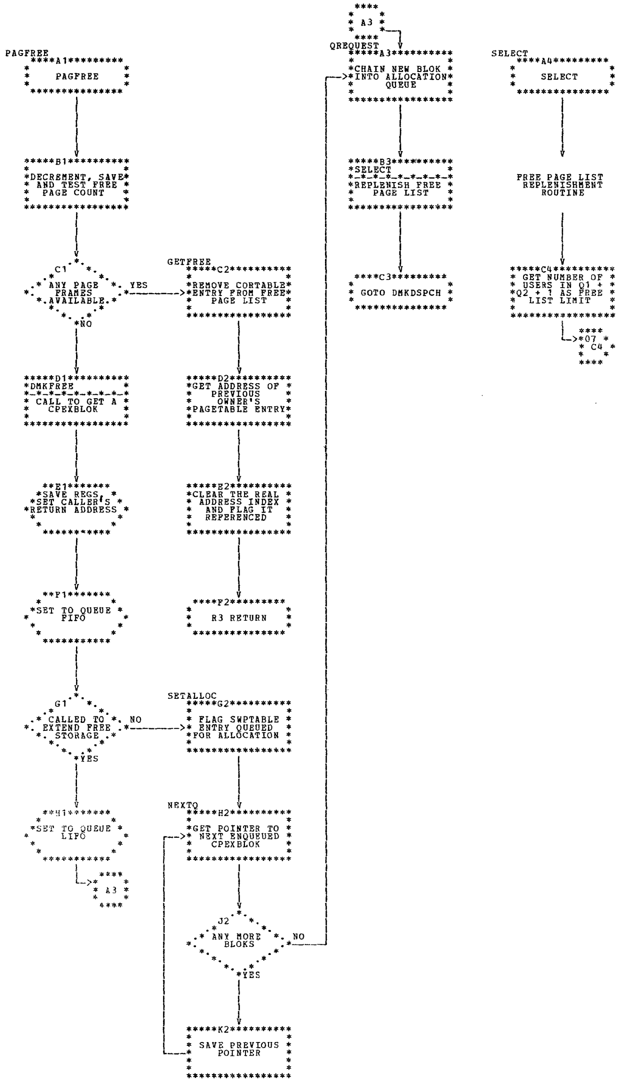

DMKPTRFR
A1
DMKPTRFR
CALLED FROM OUTSIDE DMKPTR TO OBTAIN A PAGE FRAME
C1
CALLED BY DMKFREE TO EXTEND
NO
YES
D1
TEST AND SET EXTEND LOCK
E1
LOCK ALREADY SET
NO
YES
A4
F1
SVC 0 - ABEND CODE PTR006
PAGEGET
C2
PAGFREE
GET A FREE PAGE
D2
CALLED TO EXTEND FREE STORAGE
YES
NO
A5
E2
RETURN ADDRESS OF CORTABLE ENTRY FOR FRAME TO CALLER
E3
PTFRSEL
E3
SELECT
REPLENISH FREE PAGE LIST
F3
EXIT
A4
EXTEND
A4
DMKFRET
CALL TO RELEASE FREE STORAGE FOR CPEXBLOKS
B4
ANY FREE OR SWAPPING PAGES
YES
NO
C4
DMKFREE
CALL TO GET A CPEXBLOK
D4
SET RETURN ADDRESS TO DMKPTRFE
E4
DMKSTKCP
CALL TO STACK FOR RESTART
A5
EXTPAGE
A5
FLAG CORTABLE ENTRY AS PAGE OF "FREE" STORAGE
B5
DMKFRETR
CALL TO MERGE PAGE FRAME INTO FREE STORAGE
C5
DMKFREE
CALL TO REPLACE FREE STORAGE RETURNED ABOVE
D5
RESET EXTEND LOCK TO ZERO
E3

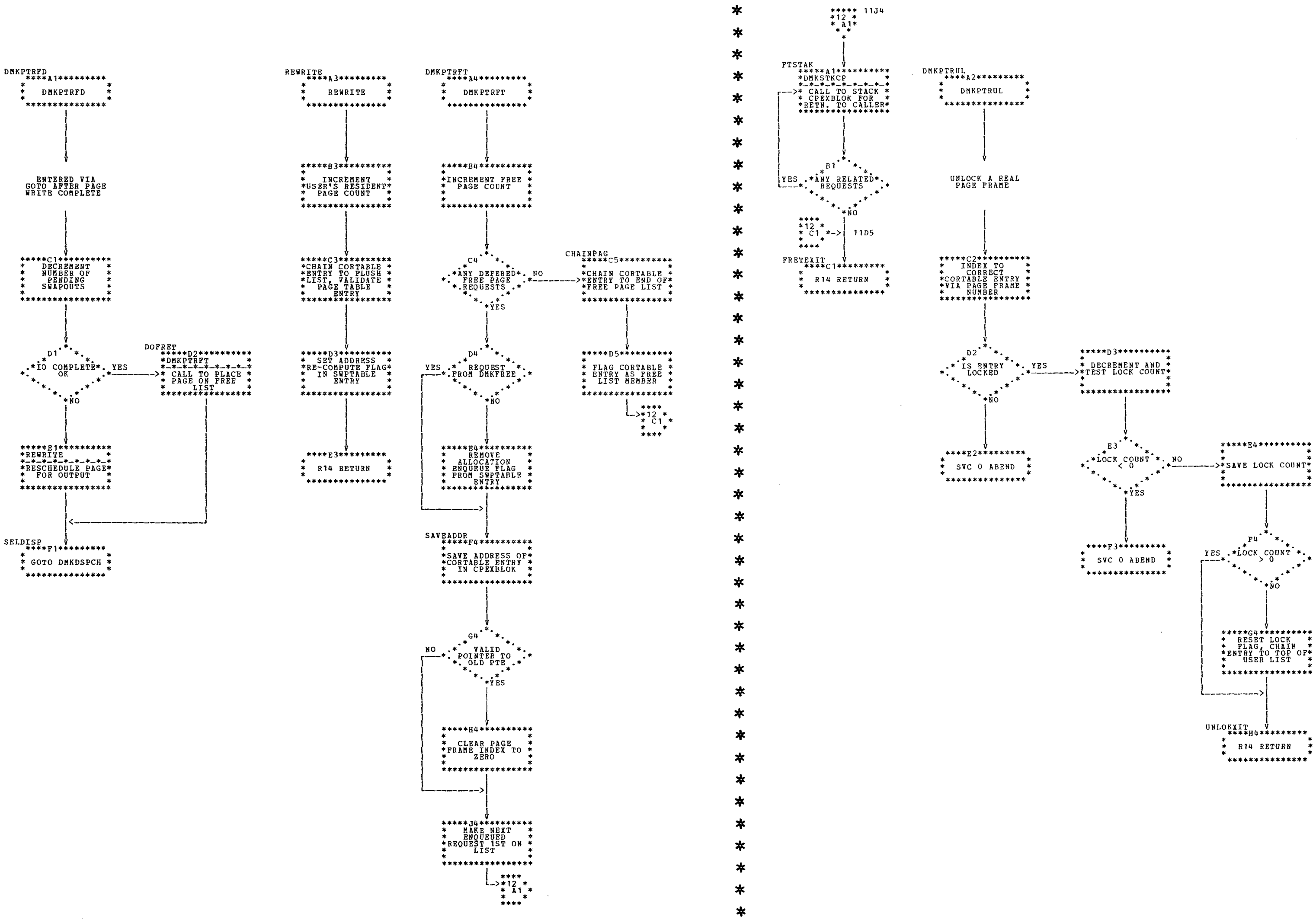

DMKPTRFE
A1
DMKPTRFE
ENTERED VIA GOTO IF A PAGE MUST BE SWAPPED TO EXTEND
C1
GET CORTABLE FOR 1ST FLUSHED OR USER PAGE
D1
ANY PAGES ON EITHER LIST
YES
NO
E1
SVC 0 - ABEND CODE PTR007
RRB
D2
RECORD CHANGED BIT STATUS, SET UP REGISTERS FOR PAGE SELECT
E2
SET EXIT ADDRESS FROM SELECT = PTRFDISP
07 A3
08D1
08F2
GETKEYS
A3
PUT KEYS FROM PAGE FRAME INTO SWPTABLE ENTRY
B3
UNCHAIN THE SELECTED CORTABLE ENTRY FROM THE LIST
C3
STEALING A RESERVED PAGE
NO
YES
D3
DECREMENT RESERVED PAGE COUNT, ADD 1 TO AVAILABLE PAGE COUNT
GETVMBLK
E3
GET VMBLOK OF PAGE'S OWNER, DECREMENT RESIDENT PAGE COUNT
F3
IF PAGE 0 RESIDENT, UPDATE LOCATION 80 TIMER
G3
INVALIDATE PAGE TABLE ENTRY AND SHADOW TABLES
H3
WAS PAGE FRAME CHANGED
YES
09 A1
NO
CKBKUP2
J3
BACKUP CHANGED BITS ON
NO
YES
09 A1
FRETIT
A4
DMKPTRFT
CALL TO PLACE PAGE ON FREE LIST
B4
RE-START CHARGING FOR CALLING USER
07 C4
09G2
10C4
SELECT1
C4
EXTENDING FREE STORAGE
YES
NO
D4
GET SUM OF FREE PAGES + PENDING SWAPOUTS
07 E5
08F4
09G1
E4
COMP SUM TO 1/8 COMPUTED
GE
LO
SELEXIT
E5
R8 RETURN
F4
ANY PAGES ON FLUSH LIST
NO
YES
USERPAGE
F5
SET UP INITIAL SWITCHES AND COUNTERS
07 G5
08C4
08F4
G4
COUNT FLUSHED PAGES, SET REGS TO TAKE 1ST PAGE
08 A1
SETQ
G5
POINT TO 1ST ENTRY ON USER PAGE LIST
08 K1
08 A1
07G4
GETSWAP
A1
GET ADDRESS OF SWPTABLE ENTRY FOR VIRTUAL PAGE
A
B1
GET REAL ADDRESS OF 1ST AND 2ND HALF OF REAL PAGE FRAME
C1
RESET REFERENCE BITS IN STORAGE KEYS, SAVE RESULTING CC
D1
TAKING PAGE FROM FLUSH LIST
YES
07 A3
NO
E1
EITHER REFERENCE BIT ON
YES
NO
CKBKUP1
F1
BACKUP REF BITS ON
NO
YES
TAKEPAGE
F2
ACCUMULATE REFERENCED PAGE COUNT
07 A3
G1
RESET BACKUP FLAGS IN SWPTABLE ENTRY
SKIPAGE
H1
SAVE REF BITS IN SWPKEYS, POINT TO NEXT LIST ENTRY
J1
INCREMENT REFERENCED PAGE COUNT
08 K1
07G5
TESTPAGE
K1
MORE PAGES ON LIST
YES
A3
NO
A3
RESERVED PAGE
NO
YES
A3
B3
STEALING FOR RESERVED USER
YES
NO
QLOOPED
A4
COUNT NUMBER OF Q LOOPS
B4
TEST AND SET Q LOOP SWITCH
C4
1ST PASS THROUGH LIST
YES
07 G5
NO
D4
SET EXIT REG TO STEAL RESERVED PAGES
E4
POINT TO 1ST PAGE ON LIST
F4
LIST REALLY EMPTY
NO
07 G5
YES
07 E5

DMKPTR -- Real Storage Page Manager (Parts 7 and 8 of 13)

DMKPTR -- Real Storage Page Manager (Parts 9 and 10 of 13)

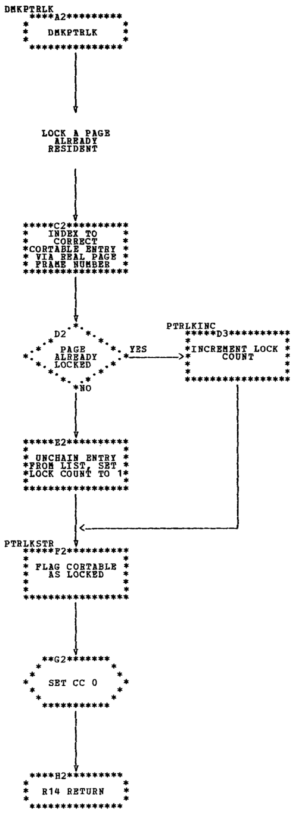

09 A1
07H3
07J3
TESTRCMP
A1
DASD PAGE READ ONLY
YES
NO
B1
POINT TO RDEVBLOK FOR PAGING DEVICE ON WHICH PAGE RESIDES
C1
MOVEABLE HEAD DEVICE
NO
YES
D1
DMKPGTPR
CALL TO RELEASE OLD DASD PAGE SLOT
GETPAG
E1
DMKPGTPG
CALL TO OBTAIN NEW DASD PAGE SLOT
F1
ANY DASD SLOTS AVAILABLE.
YES
NO
A3
G1
REWRITE
RE-QUEUE FOR OUTPUT
07 E5
A3
SETADDR
A3
PUT DASD ADDRESS IN SWPTABLE ENTRY, RESET RECMP FLAG
QWRITE
C2
INCREMENT NUMBER OF PENDING SWAPOUTS
D2
DMKFREE
CALL TO GET A CPEXBLOK
E2
SET RETURN ADDRESS TO DMKPTRFD, SAVE REGS
F2
DMKPAGIO
CALL TO SCHEDULE PAGE WRITE
G2
RE-START CHARGING CALLING USER
07 C4
PTRFDISP
A4
ENTERED FROM SELECT
B4
GOTO DMKDSPCH

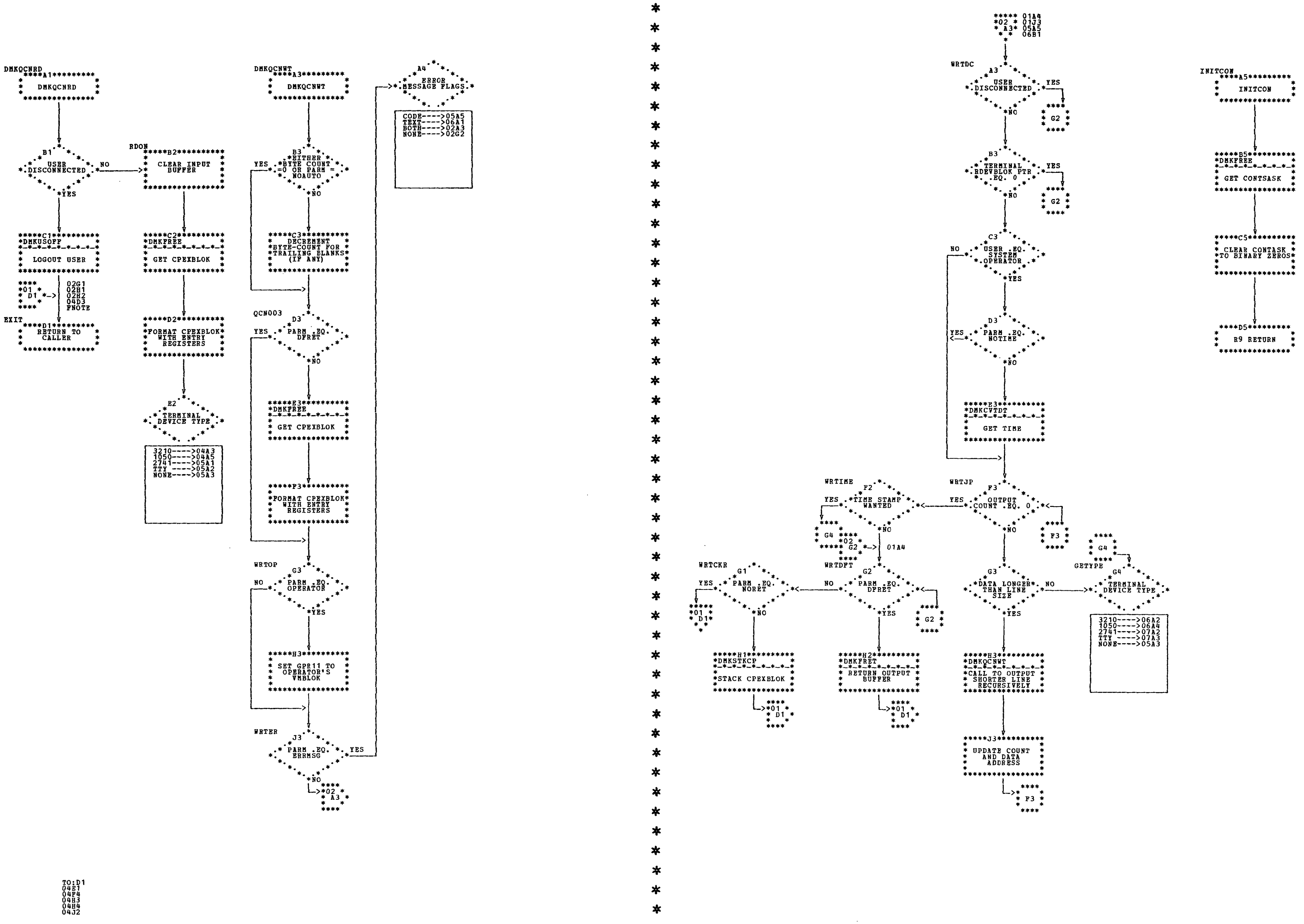

PAGFREE
A1
PAGFREE
B1
DECREMENT, SAVE AND TEST FREE PAGE COUNT
C1
ANY PAGE FRAMES AVAILABLE.
YES
NO
D1
DMKFREE
CALL TO GET A CPEXBLOK
E1
SAVE REGS, SET CALLER'S RETURN ADDRESS
F1
SET TO QUEUE FIFO
G1
CALLED TO EXTEND FREE STORAGE
NO
YES
H1
SET TO QUEUE LIFO
A3
GETFREE
C2
REMOVE CORTABLE ENTRY FROM FREE PAGE LIST
D2
GET ADDRESS OF PREVIOUS OWNER'S PAGETABLE ENTRY
E2
CLEAR THE REAL ADDRESS INDEX AND FLAG IT REFERENCED
F2
R3 RETURN
SETALLOC
G2
FLAG SWPTABLE ENTRY QUEUED FOR ALLOCATION
NEXTQ
H2
GET POINTER TO NEXT ENQUEUED CPEXBLOK
J2
ANY MORE BLOKS
NO
YES
K2
SAVE PREVIOUS POINTER
A3
QREQUEST
A3
CHAIN NEW BLOK INTO ALLOCATION QUEUE
B3
SELECT
REPLENISH FREE PAGE LIST
C3
GOTO DMKDSPCH
SELECT
A4
SELECT
FREE PAGE LIST REPLENISHMENT ROUTINE
C4
GET NUMBER OF USERS IN Q1 + Q2 + 1 AS FREE LIST LIMIT
07 C4


DMKPTRFD
A1
DMKPTRFD
ENTERED VIA GOTO AFTER PAGE WRITE COMPLETE
C1
DECREMENT NUMBER OF PENDING SWAPOUTS
D1
IO COMPLETE OK
YES
NO
DOFRET
D2
DMKPTRFT
CALL TO PLACE PAGE ON FREE LIST
E1
REWRITE
RESCHEDULE PAGE FOR OUTPUT
SELDISP
F1
GOTO DMKDSPCH
REWRITE
A3
REWRITE
B3
INCREMENT USER'S RESIDENT PAGE COUNT
C3
CHAIN CORTABLE ENTRY TO FLUSH LIST, VALIDATE PAGE TABLE ENTRY
D3
SET ADDRESS RE-COMPUTE FLAG IN SWPTABLE ENTRY
E3
R14 RETURN
DMKPTRFT
A4
DMKPTRFT
B4
INCREMENT FREE PAGE COUNT
C4
ANY DEFERED FREE PAGE REQUESTS
NO
YES
CHAINPAG
C5
CHAIN CORTABLE ENTRY TO END OF FREE PAGE LIST
D5
FLAG CORTABLE ENTRY AS FREE LIST MEMBER
12 C1
D4
REQUEST FROM DMKFREE
YES
NO
E4
REMOVE ALLOCATION ENQUEUE FLAG FROM SWPTABLE ENTRY
SAVEADDR
F4
SAVE ADDRESS OF CORTABLE ENTRY IN CPEXBLOK
G4
VALID POINTER TO OLD PTE
NO
YES
H4
CLEAR PAGE FRAME INDEX TO ZERO
J4
MAKE NEXT ENQUEUED REQUEST 1ST ON LIST
12 A1
12 A1 11J4
FTSTAK
A1
DMKSTKCP
CALL TO STACK CPEXBLOK FOR RETN. TO CALLER
B1
ANY RELATED REQUESTS
YES
NO
12 C1 11D5
FRETEXIT
C1
R14 RETURN
DMKPTRUL
A2
DMKPTRUL
UNLOCK A REAL PAGE FRAME
C2
INDEX TO CORRECT CORTABLE ENTRY VIA PAGE FRAME NUMBER
D2
IS ENTRY LOCKED
YES
NO
E2
SVC 0 ABEND
D3
DECREMENT AND TEST LOCK COUNT
E3
LOCK COUNT < 0
NO
YES
E4
SAVE LOCK COUNT
F3
SVC 0 ABEND
F4
LOCK COUNT > 0
YES
NO
G4
RESET LOCK FLAG, CHAIN ENTRY TO TOP OF USER LIST
UNLOKXIT
H4
R14 RETURN

DMKPTR -- Real Storage Page Manager (Parts 11 and 12 of 13)

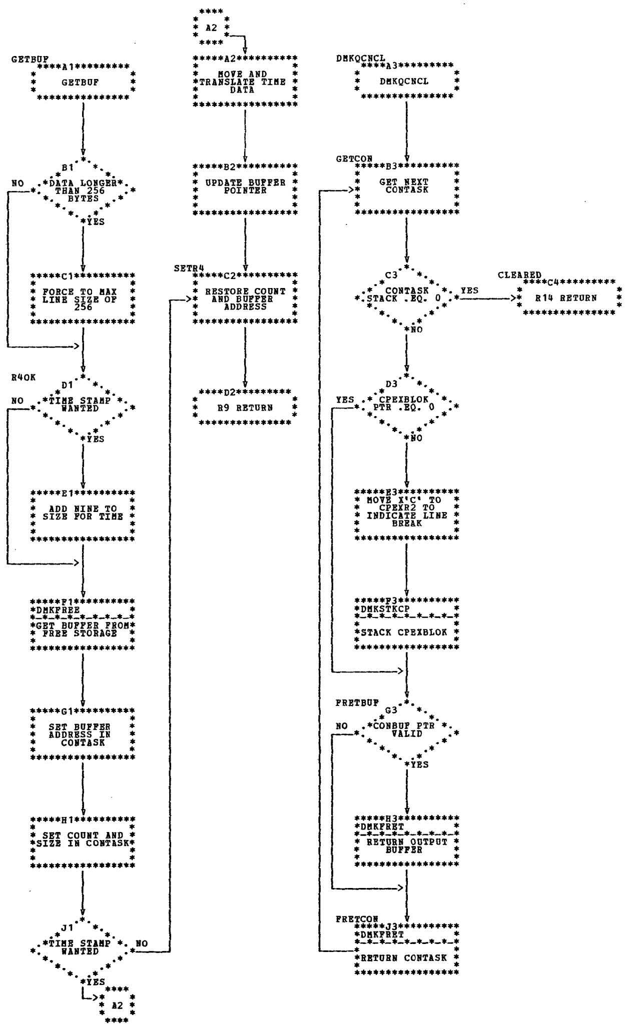
DMKPTR -- Real Storage Page Manager (Part 13 of 13)

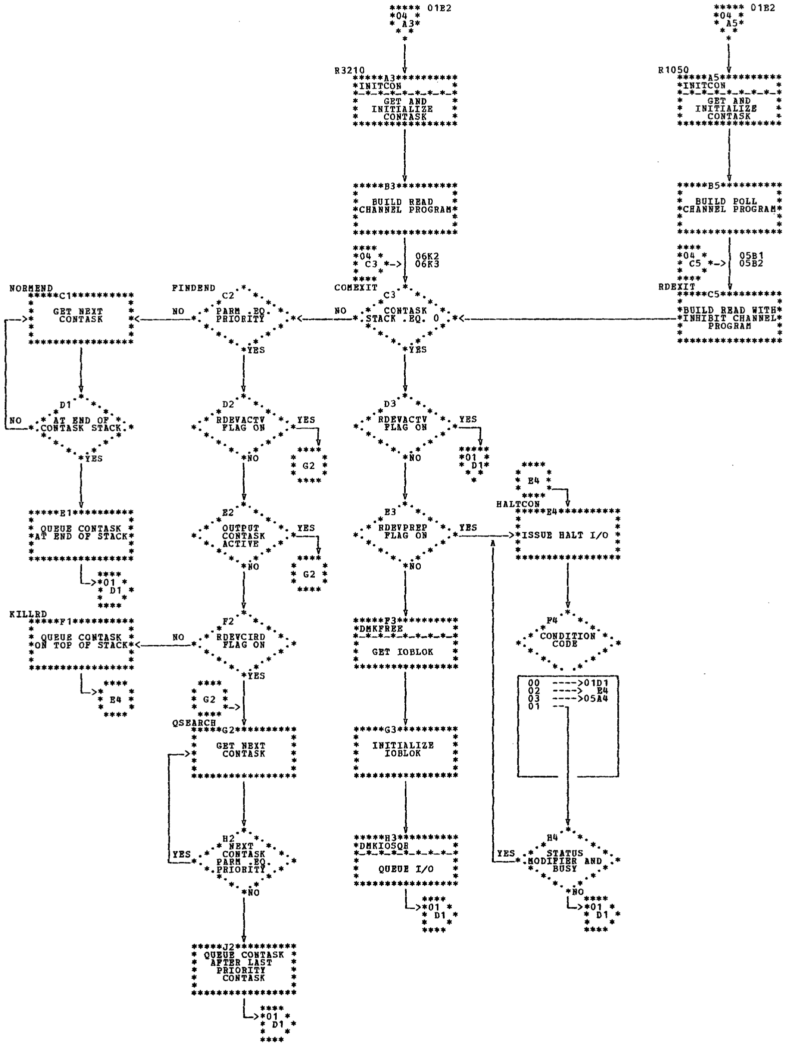


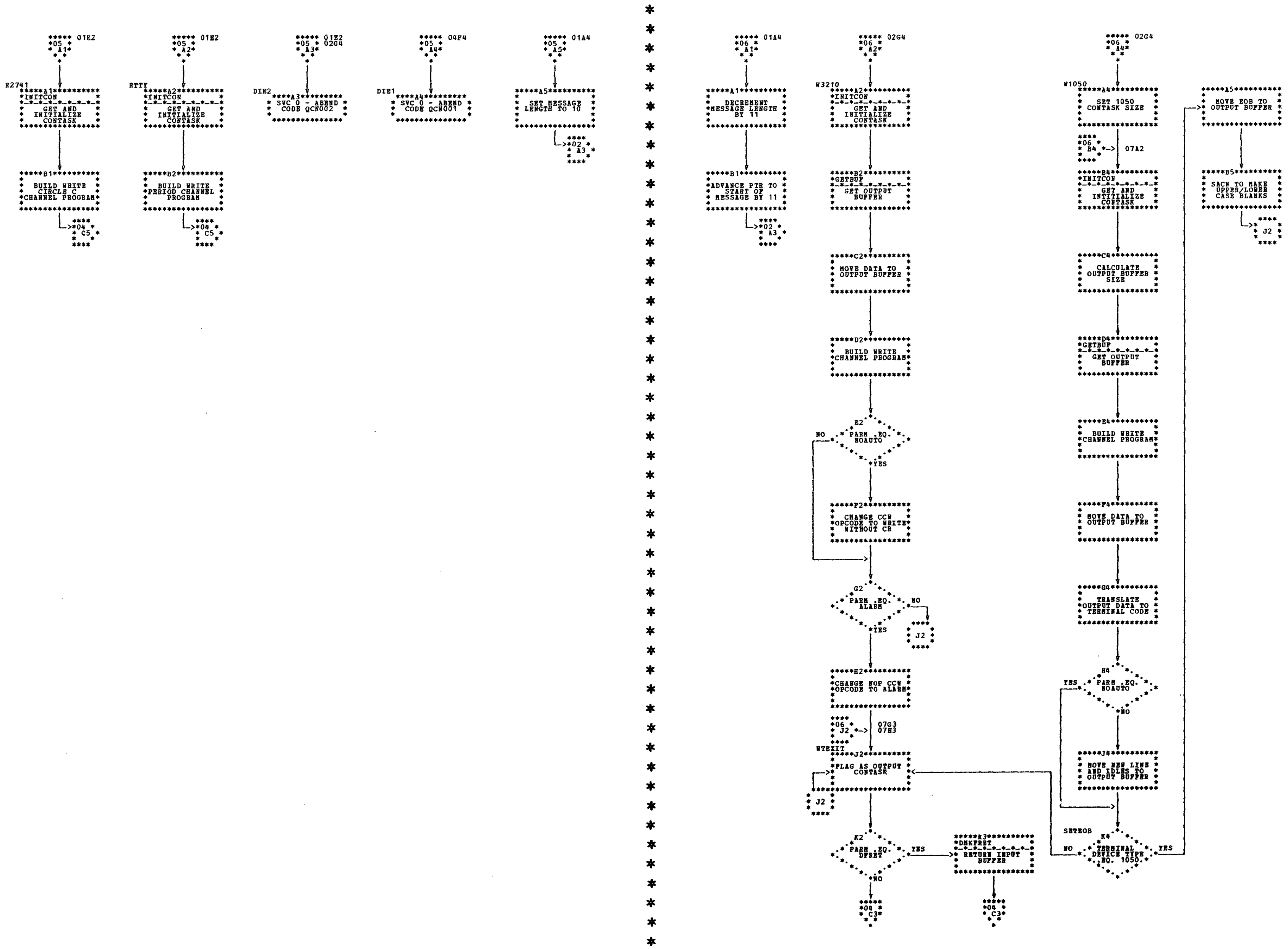


DMKQCN -- Ccnsole Message Queue Manager (Parts 1 and 2 of 7)

DMKQCN -- Console Message Queue Manager (Parts 3 and 4 of 7)

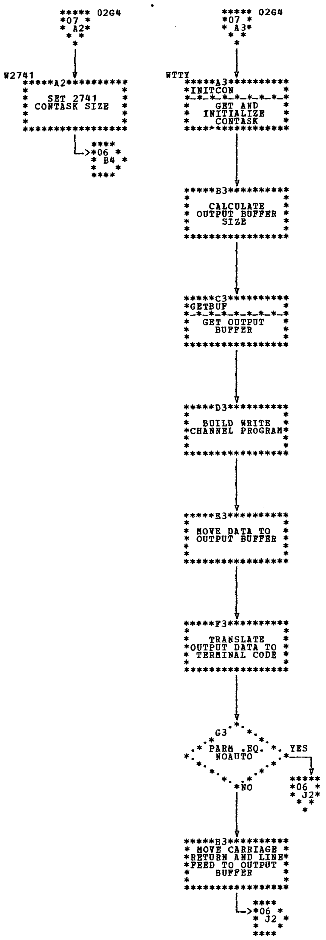


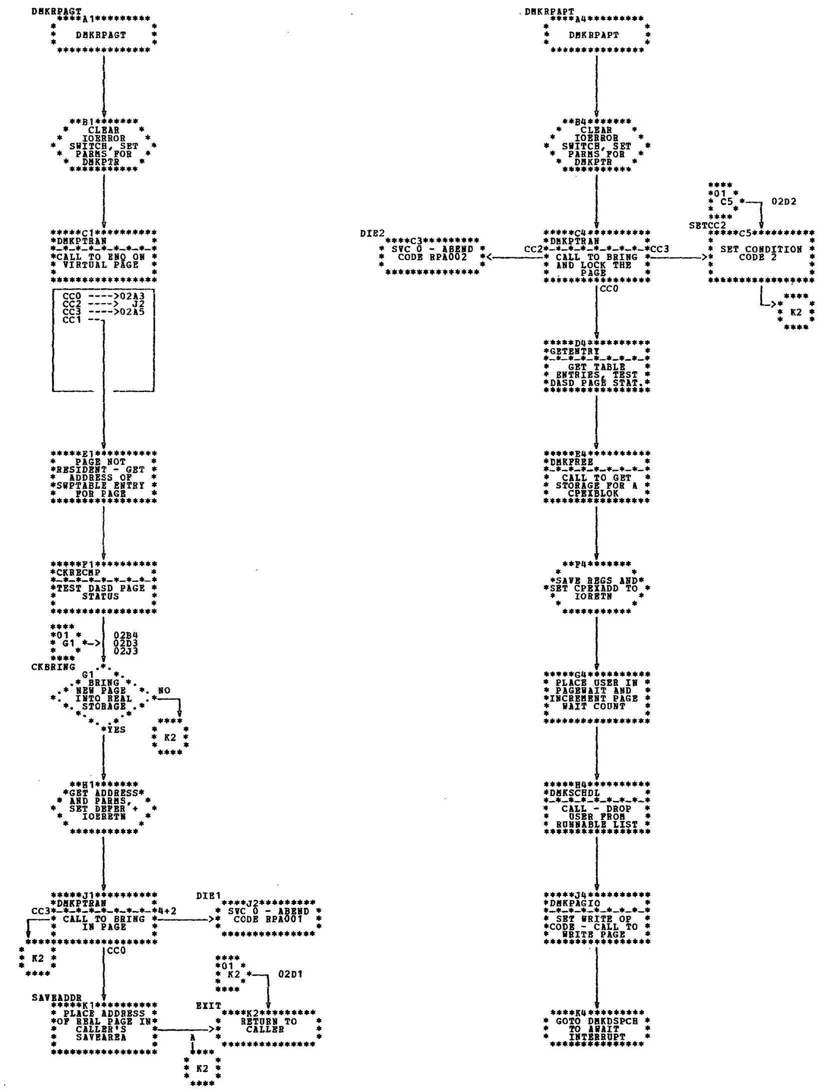


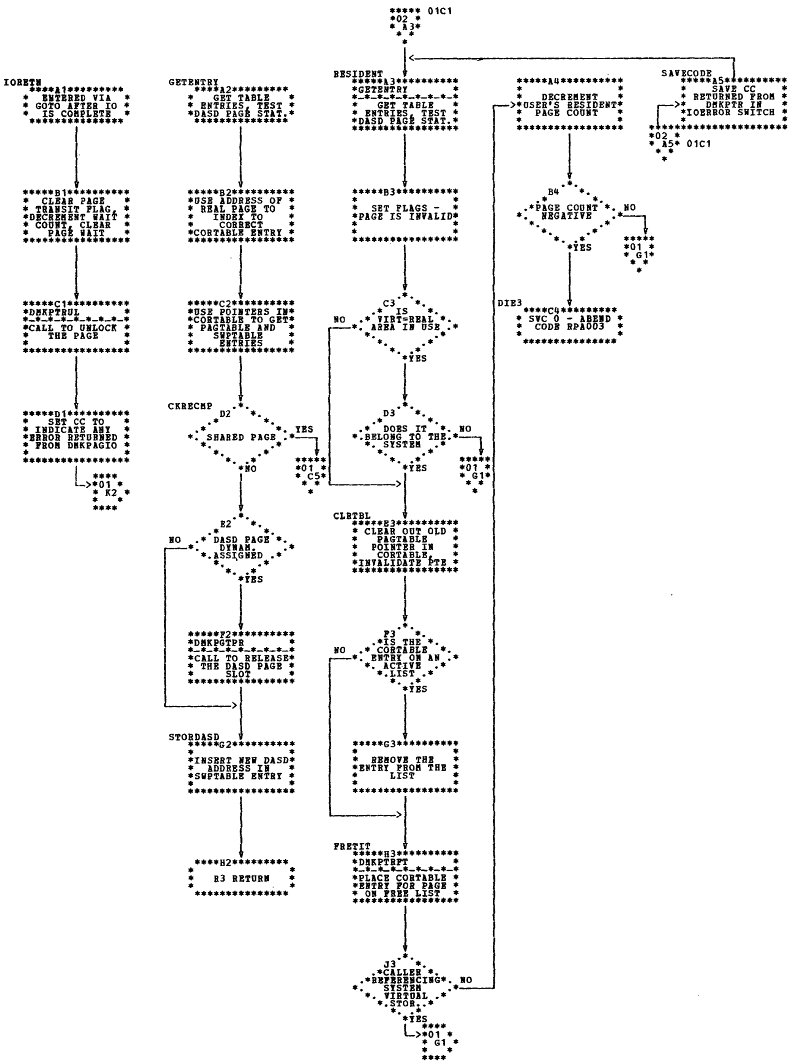

01E2
05 A1
R2741
A1
INITCON
GET AND INITIALIZE CONTASK
B1
BUILD WRITE CIRCLE C CHANNEL PROGRAM
04 C5
01E2
05 A2
RTTY
A2
INITCON
GET AND INITIALIZE CONTASK
B2
BUILD WRITE PERIOD CHANNEL PROGRAM
04 C5
01E2
02G4
05 A3
DIE2
A3
SVC 0 - ABEND CODE QCN002
04F4
05 A4
DIE1
A4
SVC 0 - ABEND CODE QCN001
01A4
05 A5
A5
SET MESSAGE LENGTH TO 10
02 A3
01A4
06 A1
A1
DECREMENT MESSAGE LENGTH BY 11
B1
ADVANCE PTR TO START OF MESSAGE BY 11
02 A3
02G4
06 A2
W3210
A2
INITCON
GET AND INITIALIZE CONTASK
B2
GETBUF
GET OUTPUT BUFFER
C2
MOVE DATA TO OUTPUT BUFFER
D2
BUILD WRITE CHANNEL PROGRAM
E2
PARM .EQ. NOAUTO
NO
YES
F2
CHANGE CCW OPCODE TO WRITE WITHOUT CR
G2
PARM .EQ. ALARM
NO
J2
YES
H2
CHANGE NOP CCW OPCODE TO ALARM
06 J2
07G3
07H3
WTEXIT
J2
FLAG AS OUTPUT CONTASK
J2
K2
PARM .EQ. DFRET
YES
NO
04 C3
K3
DMKFRET
RETURN INPUT BUFFER
04 C3
02G4
06 A4
W1050
A4
SET 1050 CONTASK SIZE
06 B4
07A2
B4
INITCON
GET AND INTITIALIZE CONTASK
C4
CALCULATE OUTPUT BUFFER SIZE
D4
GETBUF
GET OUTPUT BUFFER
E4
BUILD WRITE CHANNEL PROGRAM
F4
MOVE DATA TO OUTPUT BUFFER
G4
TRANSLATE OUTPUT DATA TO TERMINAL CODE
H4
PARM .EQ. NOAUTO
YES
NO
J4
MOVE NEW LINE AND IDLES TO OUTPUT BUFFER
SETEOB
K4
TERMINAL DEVICE TYPE .EQ. 1050
NO
YES
A5
MOVE EOB TO OUTPUT BUFFER
B5
SACN TO MAKE UPPER/LOWER CASE BLANKS
J2

DMKQCN -- Ccnsole Message Queue Manager (Parts 5 and 6 of 7)

DMKQCN -- Console Message Queue Manager (Part 7 of 7)

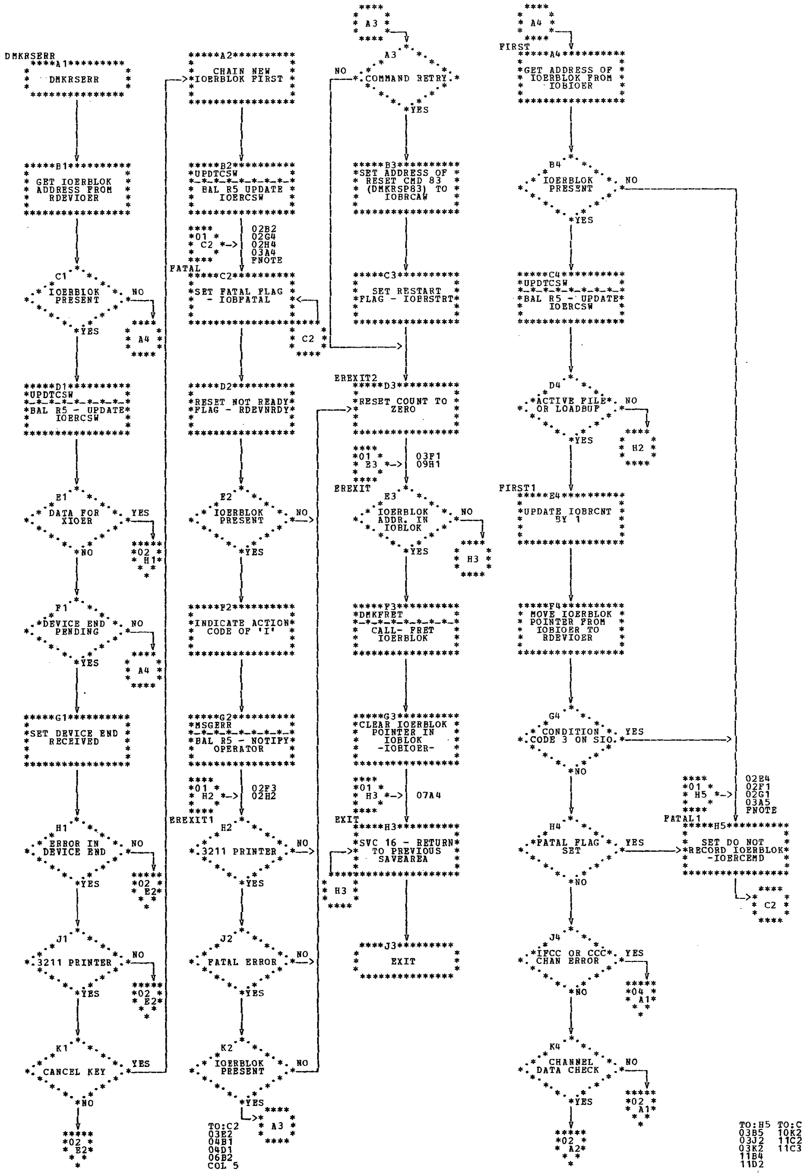


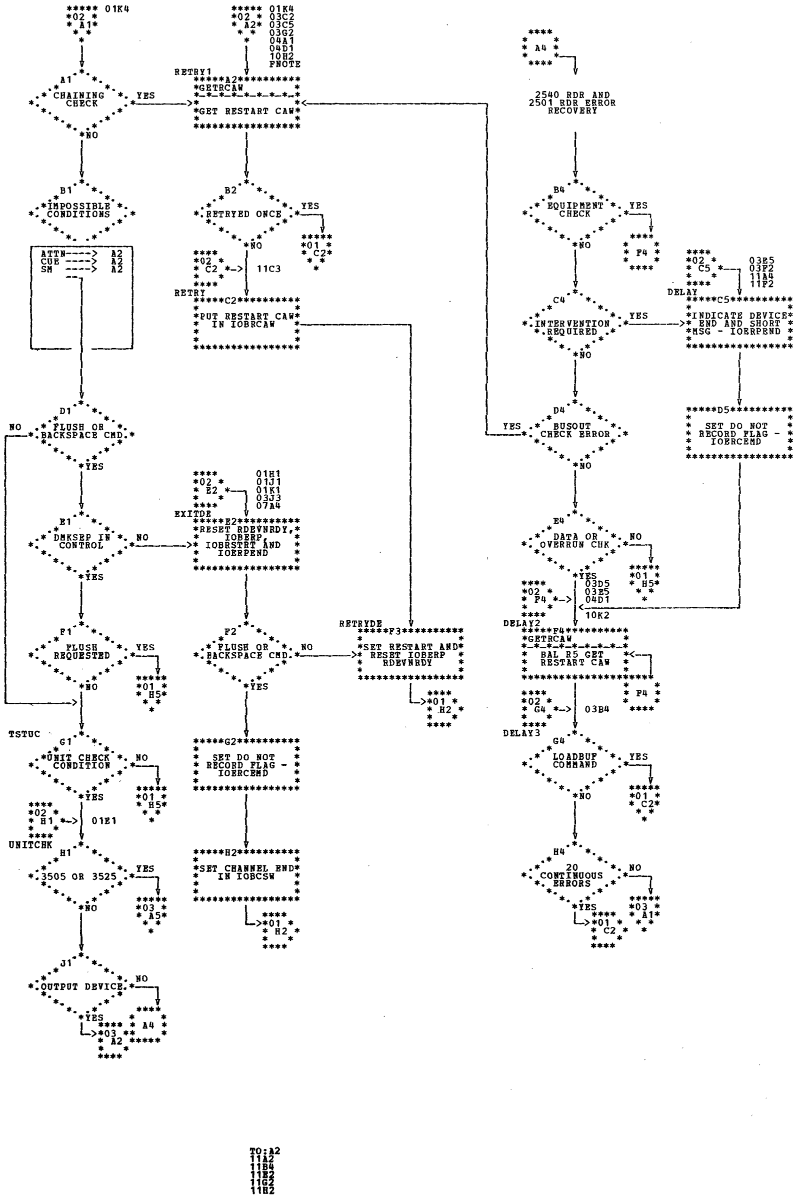

DMKRPAGT
A1
DMKRPAGT
B1
CLEAR IOERROR SWITCH, SET PARMS FOR DMKPTR
C1
DMKPTRAN
CALL TO ENQ ON VIRTUAL PAGE
CC0 ---->02A3
CC2 ----> J2
CC3 ---->02A5
CC1
E1
PAGE NOT RESIDENT - GET ADDRESS OF SWPTABLE ENTRY FOR PAGE
F1
CKRECMP
TEST DASD PAGE STATUS
01 G1
02B4
02D3
02J3
CKBRING
G1
BRING NEW PAGE INTO REAL STORAGE
NO
YES
K2
H1
GET ADDRESS AND PARMS, SET DEFER + IOERRETN
J1
DMKPTRAN
CC3
4+2
CALL TO BRING IN PAGE
K2
CC0
DIE1
J2
SVC 0 - ABEND CODE RPA001
01 K2
02D1
SAVEADDR
K1
PLACE ADDRESS OF REAL PAGE IN CALLER'S SAVEAREA
EXIT
K2
RETURN TO CALLER
A
K2
DMKRPAPT
A4
DMKRPAPT
B4
CLEAR IOERROR SWITCH, SET PARMS FOR DMKPTR
DIE2
C3
SVC 0 - ABEND CODE RPA002
C4
DMKPTRAN
CC2
CC3
CALL TO BRING AND LOCK THE PAGE
CC0
01 C5
02D2
SETCC2
C5
SET CONDITION CODE 2
K2
D4
GETENTRY
GET TABLE ENTRIES, TEST DASD PAGE STAT.
E4
DMKFREE
CALL TO GET STORAGE FOR A CPEXBLOK
F4
SAVE REGS AND SET CPEXADD TO IORETN
G4
PLACE USER IN PAGEWAIT AND INCREMENT PAGE WAIT COUNT
H4
DMKSCHDL
CALL - DROP USER FROM RUNNABLE LIST
J4
DMKPAGIO
SET WRITE OP CODE - CALL TO WRITE PAGE
K4
GOTO DMKDSPCH TO AWAIT INTERRUPT

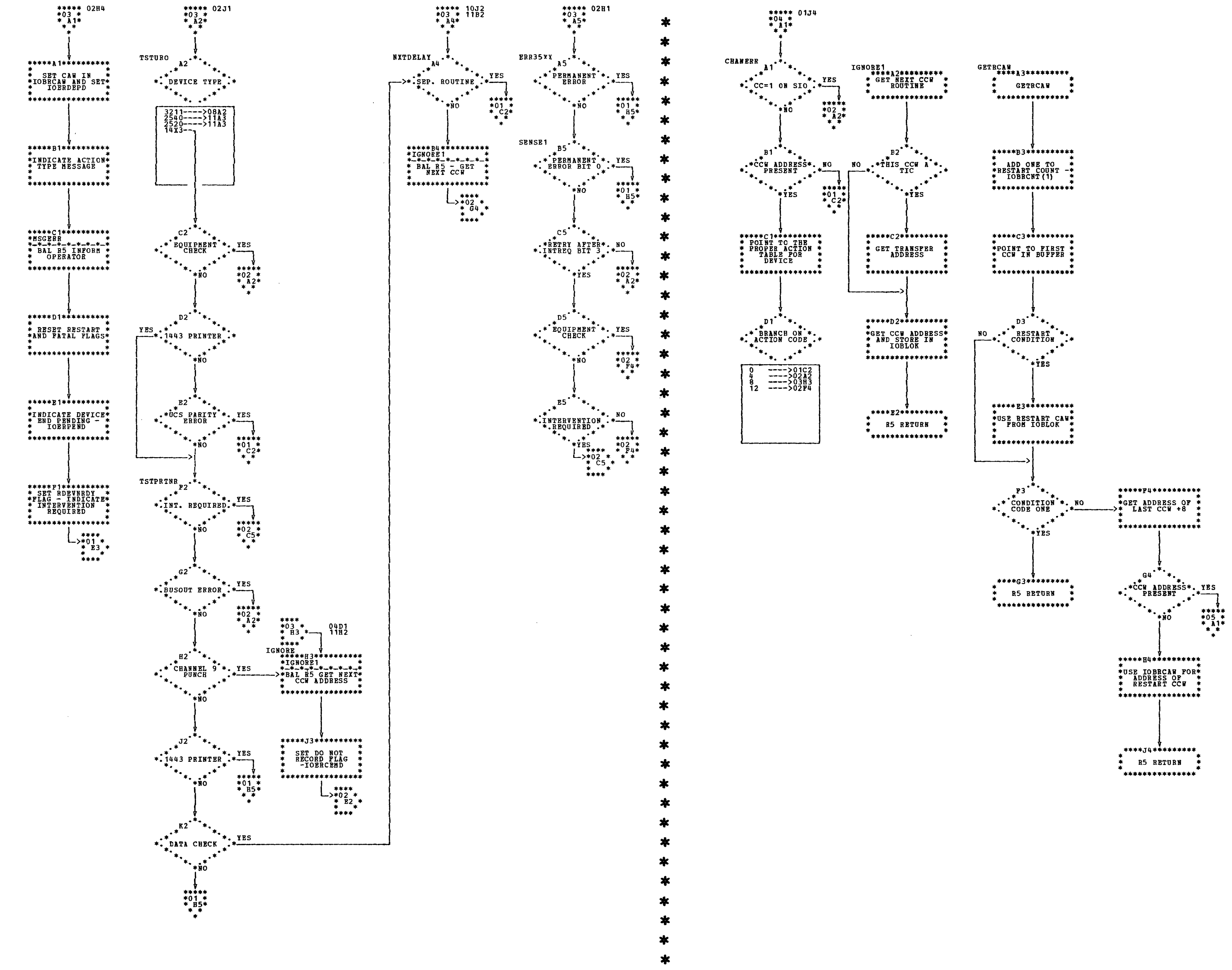

IORETN
A1
ENTERED VIA GOTO AFTER IO IS COMPLETE
B1
CLEAR PAGE TRANSIT FLAG, DECREMENT WAIT COUNT, CLEAR PAGE WAIT
C1
DMKPTRUL
CALL TO UNLOCK THE PAGE
D1
SET CC TO INDICATE ANY ERROR RETURNED FROM DMKPAGIO
01 K2
GETENTRY
A2
GET TABLE ENTRIES, TEST DASD PAGE STAT.
B2
USE ADDRESS OF REAL PAGE TO INDEX TO CORRECT CORTABLE ENTRY
C2
USE POINTERS IN CORTABLE TO GET PAGTABLE AND SWPTABLE ENTRIES
CKRECMP
D2
SHARED PAGE
YES
NO
01 C5
E2
DASD PAGE DYNAM. ASSIGNED
NO
YES
F2
DMKPGTPR
CALL TO RELEASE THE DASD PAGE SLOT
STORDASD
G2
INSERT NEW DASD ADDRESS IN SWPTABLE ENTRY
H2
R3 RETURN
02 A3
01C1
RESIDENT
A3
GETENTRY
GET TABLE ENTRIES, TEST DASD PAGE STAT.
B3
SET FLAGS - PAGE IS INVALID
C3
IS VIRT=REAL AREA IN USE
NO
YES
D3
DOES IT BELONG TO THE SYSTEM
NO
YES
01 G1
CLRTBL
E3
CLEAR OUT OLD PAGTABLE POINTER IN CORTABLE, INVALIDATE PTE
F3
IS THE CORTABLE ENTRY ON AN ACTIVE LIST
NO
YES
G3
REMOVE THE ENTRY FROM THE LIST
FRETIT
H3
DMKPTRPT
PLACE CORTABLE ENTRY FOR PAGE ON FREE LIST
J3
CALLER REFERENCING SYSTEM VIRTUAL STOR.
NO
YES
01 G1
A4
DECREMENT USER'S RESIDENT PAGE COUNT
B4
PAGE COUNT NEGATIVE
NO
YES
01 G1
DIE3
C4
SVC 0 - ABEND CODE RPA003
SAVECODE
A5
SAVE CC RETURNED FROM DMKPTR IN IOERROR SWITCH
02 A5
01C1

DMKRPA -- Real Page Access Manager (Parts 1 and 2 of 2)

| DMKRSE -- Real Spool Error Procedures (Parts 1 and 2 of 11)

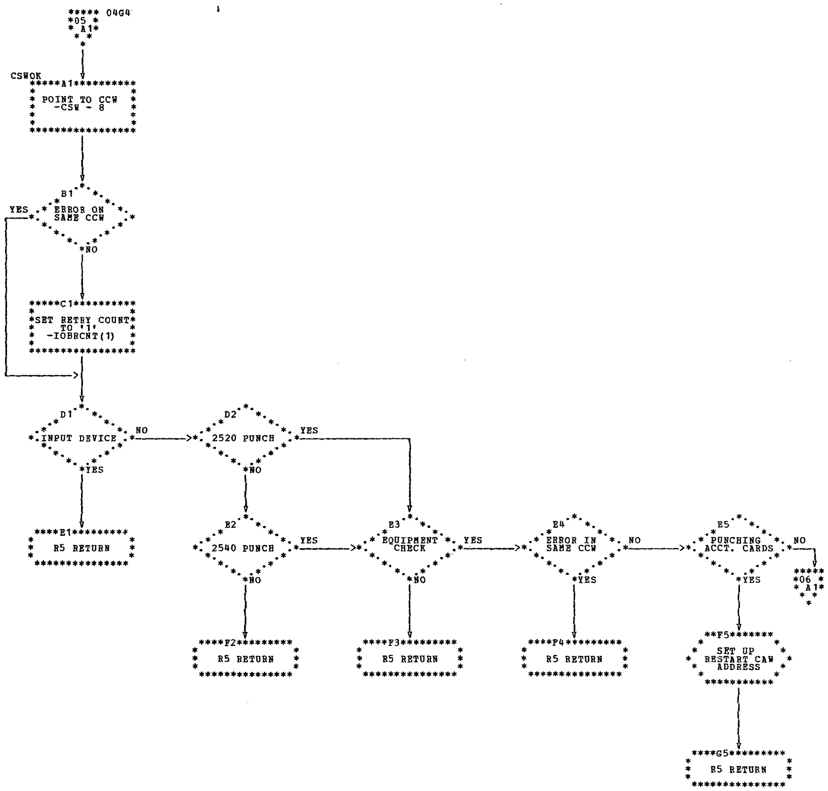

DMKRSERR
A1 DMKRSERR
B1 GET IOERBLOK ADDRESS FROM RDEVIOER
C1 IOERBLOK PRESENT
NO A4
YES
D1 UPDTCSW BAL R5 - UPDATE IOERCSW
E1 DATA FOR XIOER
YES 02 H1
NO
F1 DEVICE END PENDING
NO A4
G1 SET DEVICE END RECEIVED
H1 ERROR IN DEVICE END
NO 02 E2
J1 3211 PRINTER
NO 02 E2
K1 CANCEL KEY
YES
NO 02 E2
A2 CHAIN NEW IOERBLOK FIRST
B2 UPDTCSW BAL R5 UPDATE IOERCSW
01 C2 02B2 02G4 02H4 03A4 FNOTE
FATAL
C2 SET FATAL FLAG - IOBFATAL
C2
D2 RESET NOT READY FLAG - RDEVNRDY
E2 IOERBLOK PRESENT
NO
F2 INDICATE ACTION CODE OF 'I'
G2 MSGERR BAL R5 - NOTIFY OPERATOR
01 H2 02F3 02H2
EREXIT1
H2 3211 PRINTER
NO
J2 FATAL ERROR
NO
K2 IOERBLOK PRESENT
NO
YES A3
TO:C2 03E2 04B1 04D1 06B2 COL 5
A3
A3 COMMAND RETRY
NO
YES
B3 SET ADDRESS OF RESET CMD 83 (DMKRSP83) TO IOBRCAW
C3 SET RESTART FLAG - IOERSTRT
EREXIT2
D3 RESET COUNT TO ZERO
01 B3 03F1 09H1
EREXIT
E3 IOERBLOK ADDR. IN IOBLOK
NO H3
F3 DMKFRET CALL- FRET IOERBLOK
G3 CLEAR IOERBLOK POINTER IN IOBLOK -IOBIOER-
01 H3 07A4
EXIT
H3 SVC 16 - RETURN TO PREVIOUS SAVEAREA
H3
J3 EXIT
A4
FIRST
A4 GET ADDRESS OF IOERBLOK FROM IOBIOER
B4 IOERBLOK PRESENT
NO
C4 UPDTCSW BAL R5 - UPDATE IOERCSW
D4 ACTIVE FILE OR LOADBUF
NO H2
FIRST1
E4 UPDATE IOBRCNT BY 1
F4 MOVE IOERBLOK POINTER FROM IOBIOER TO RDEVIOER
G4 CONDITION CODE 3 ON SIO.
YES
H4 FATAL FLAG SET
YES
J4 IFCC OR CCC CHAN ERROR
YES 04 A1
K4 CHANNEL DATA CHECK
NO 02 A1
YES 02 A2
01 H5 02E4 02F1 02G1 03A5 FNOTE
FATAL1
H5 SET DO NOT RECORD IOERBLOK -IOERCEND
C2
TO:H5 03B5 03J2 03K2 11B4 11D2
TO:C2 10K2 11C2 11C3

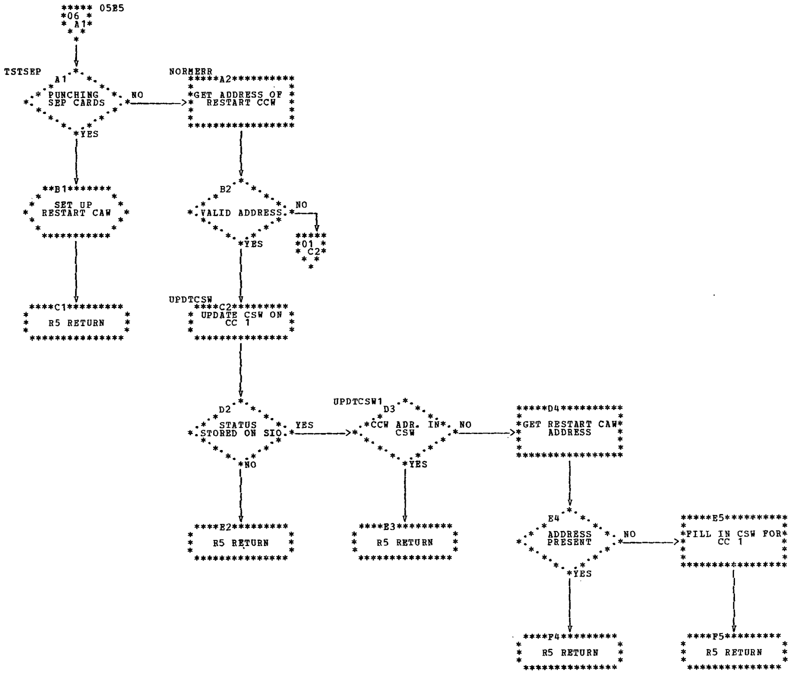

02 A1 01K4
A1 CHAINING CHECK
YES
NO
B1 IMPOSSIBLE CONDITIONS
ATTN ----> A2
CUE ----> A2
SM ----> A2
D1 FLUSH OR BACKSPACE CMD.
NO
YES
E1 DMKSEP IN CONTROL
NO
F1 FLUSH REQUESTED
YES 01 H5
TSTUC
G1 UNIT CHECK CONDITION
NO 01 H5
02 H1 01E1
UNITCHK
H1 3505 OR 3525
YES 03 A5
J1 OUTPUT DEVICE
NO A4
YES 03 A2
02 A2 01K4 03C2 03C5 03G2 04A1 04D1 10H2 FNOTE
RETRY1
A2 GETRCAW GET RESTART CAW
B2 RETRYED ONCE
YES 01 C2
02 C2 11C3
RETRY
C2 PUT RESTART CAW IN IOBRCAW
02 E2 01H1 01J1 01K1 03J3 07A4
EXITDE
E2 RESET RDEVNRDY, IOBERP, IOBRSTRT AND IOERPEND
F2 FLUSH OR BACKSPACE CMD.
NO
YES
RETRYDE
F3 SET RESTART AND RESET IOBERP RDEVNRDY
01 H2
G2 SET DO NOT RECORD FLAG - IOERCEND
H2 SET CHANNEL END IN IOBCSW
01 H2
TO:A2 11A2 11B4 11E2 11G2 11H2
A4
2540 RDR AND 2501 RDR ERROR RECOVERY
B4 EQUIPMENT CHECK
YES F4
C4 INTERVENTION REQUIRED
YES
02 C5 03E5 03F2 11A4 11F2
DELAY
C5 INDICATE DEVICE END AND SHORT MSG - IOERPEND
D4 BUSOUT CHECK ERROR
YES
D5 SET DO NOT RECORD FLAG - IOERCEND
E4 DATA OR OVERRUN CHK
NO 01 H5
02 F4 03D5 03E5 04D1 10K2
DELAY2
F4 GETRCAW BAL R5 GET RESTART CAW
F4
02 G4 03B4
DELAY3
G4 LOADBUF COMMAND
YES 01 C2
H4 20 CONTINUOUS ERRORS
NO 03 A1
YES 01 C2

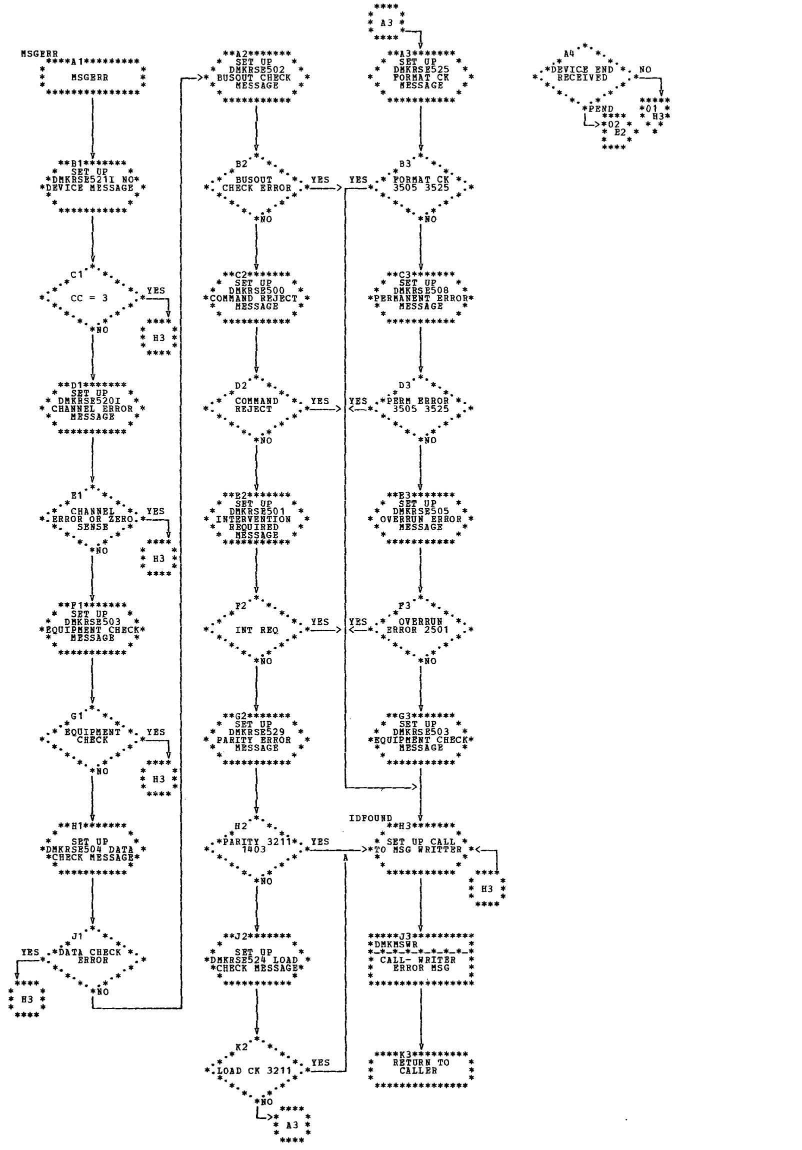


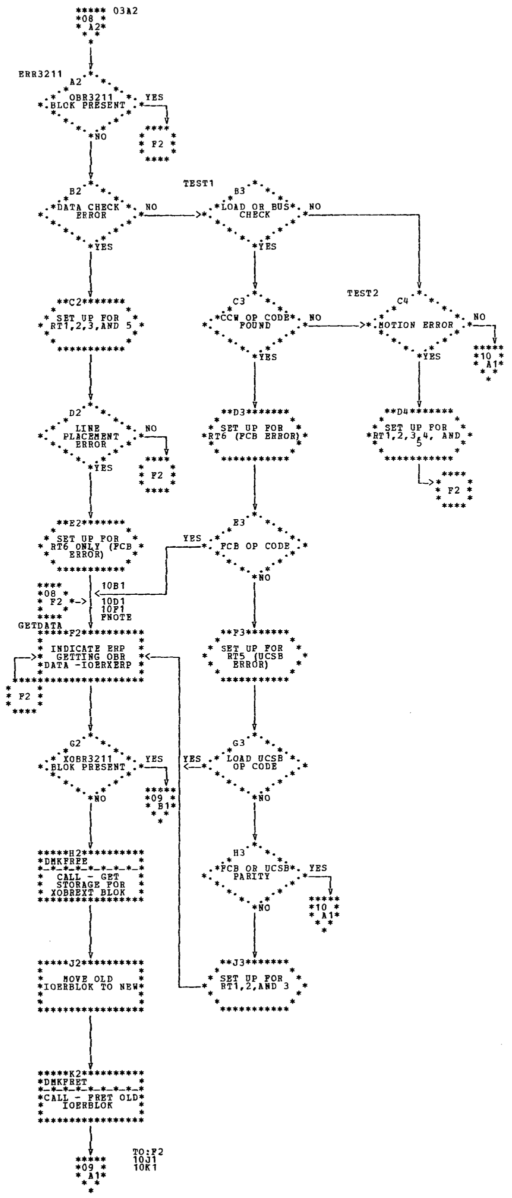
| DMKRSE -- Real Spool Error Procedures (Parts 3 and 4 of 11)

| DMKRSE -- Real Spool Error Procedures (Parts 5 and 6 of 11)



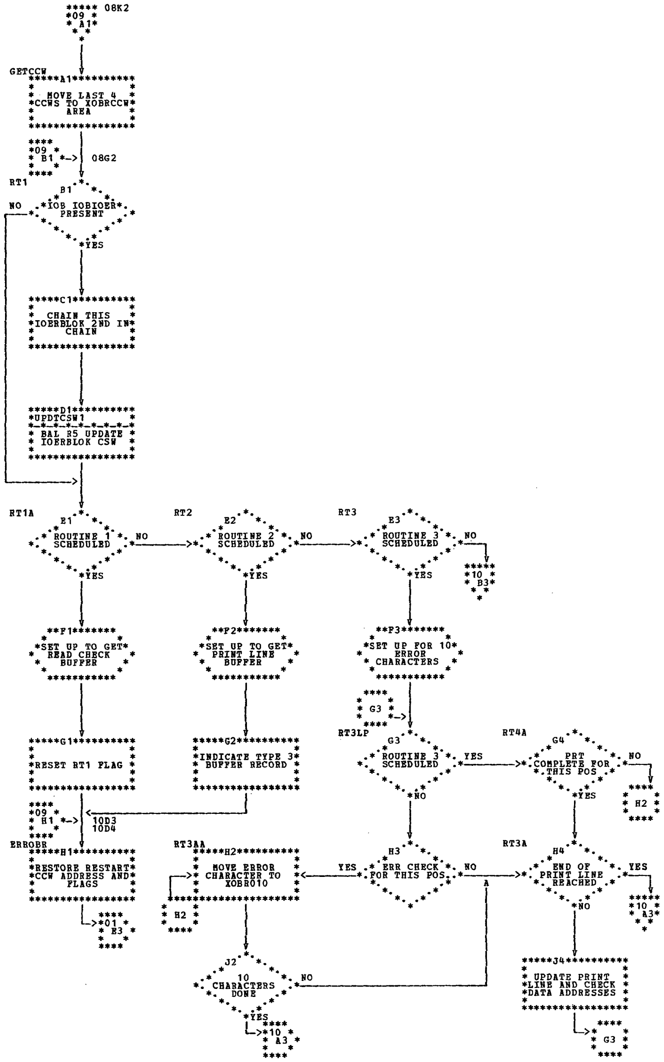


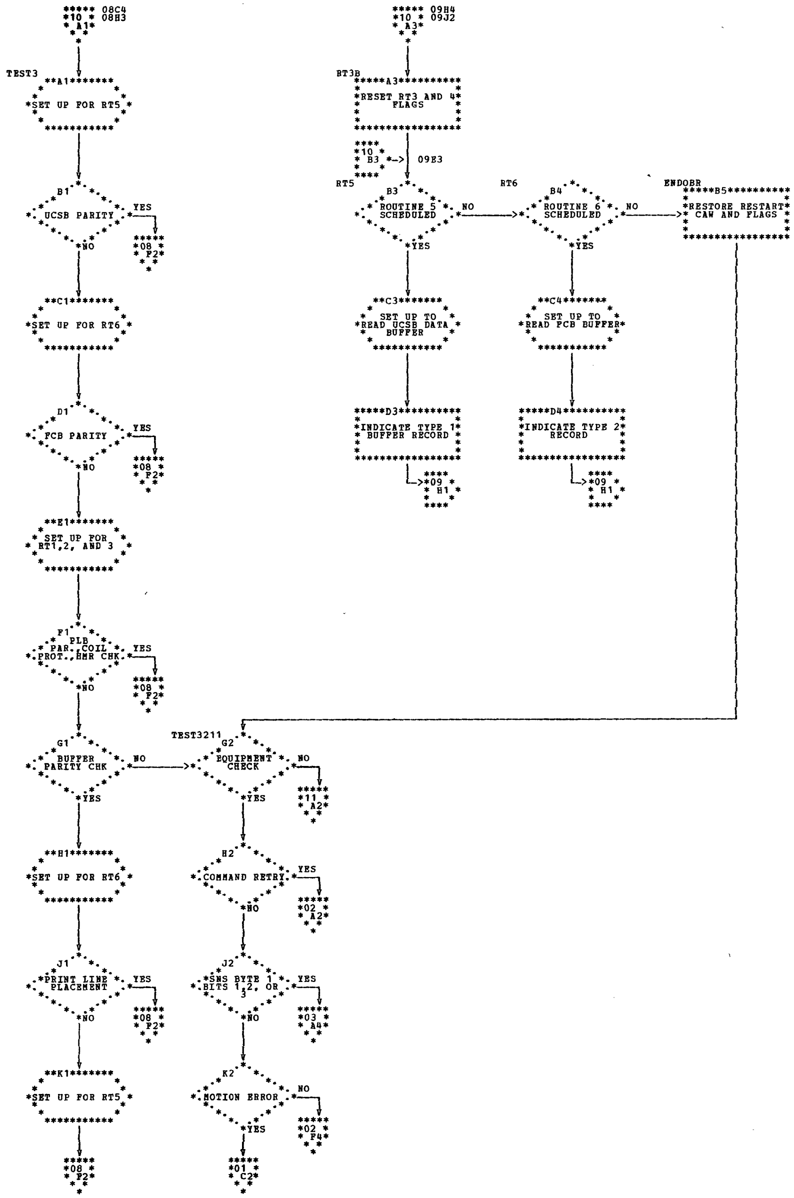

MSGERR
A1 MSGERR
B1 SET UP DMKRSE521I NO DEVICE MESSAGE
C1 CC = 3
YES
H3
NO
D1 SET UP DMKRSE520I CHANNEL ERROR MESSAGE
E1 CHANNEL ERROR OR ZERO SENSE
YES
H3
NO
F1 SET UP DMKRSE503 EQUIPMENT CHECK MESSAGE
G1 EQUIPMENT CHECK
YES
H3
NO
H1 SET UP DMKRSE504 DATA CHECK MESSAGE
J1 DATA CHECK ERROR
YES
H3
NO
A2 SET UP DMKRSE502 BUSOUT CHECK MESSAGE
B2 BUSOUT CHECK ERROR
YES
NO
C2 SET UP DMKRSE500 COMMAND REJECT MESSAGE
D2 COMMAND REJECT
YES
NO
E2 SET UP DMKRSE501 INTERVENTION REQUIRED MESSAGE
F2 INT REQ
YES
NO
G2 SET UP DMKRSE529 PARITY ERROR MESSAGE
H2 PARITY 3211 1403
YES
NO
J2 SET UP DMKRSE524 LOAD CHECK MESSAGE
K2 LOAD CK 3211
YES
NO
A3
A3 SET UP DMKRSE525 FORMAT CK MESSAGE
B3 FORMAT CK 3505 3525
YES
NO
C3 SET UP DMKRSE508 PERMANENT ERROR MESSAGE
D3 PERM ERROR 3505 3525
YES
NO
E3 SET UP DMKRSE505 OVERRUN ERROR MESSAGE
F3 OVERRUN ERROR 2501
YES
NO
G3 SET UP DMKRSE503 EQUIPMENT CHECK MESSAGE
IDFOUND
H3 SET UP CALL TO MSG WRITTER
H3
J3 DMKMSWR CALL- WRITER ERROR MSG
K3 RETURN TO CALLER
A4 DEVICE END RECEIVED
NO
01 H3
PEND
02 E2

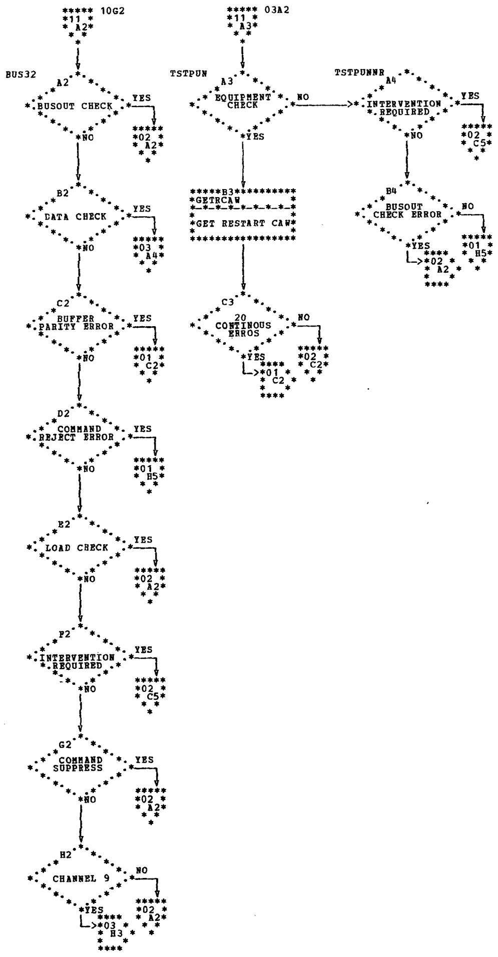


| DMKRSE -- Real Spool Error Procedures (Parts 7 and 8 of 11)

| DMKRSE -- Real Spool Error Procedures (Parts 9 and 10 of 11)

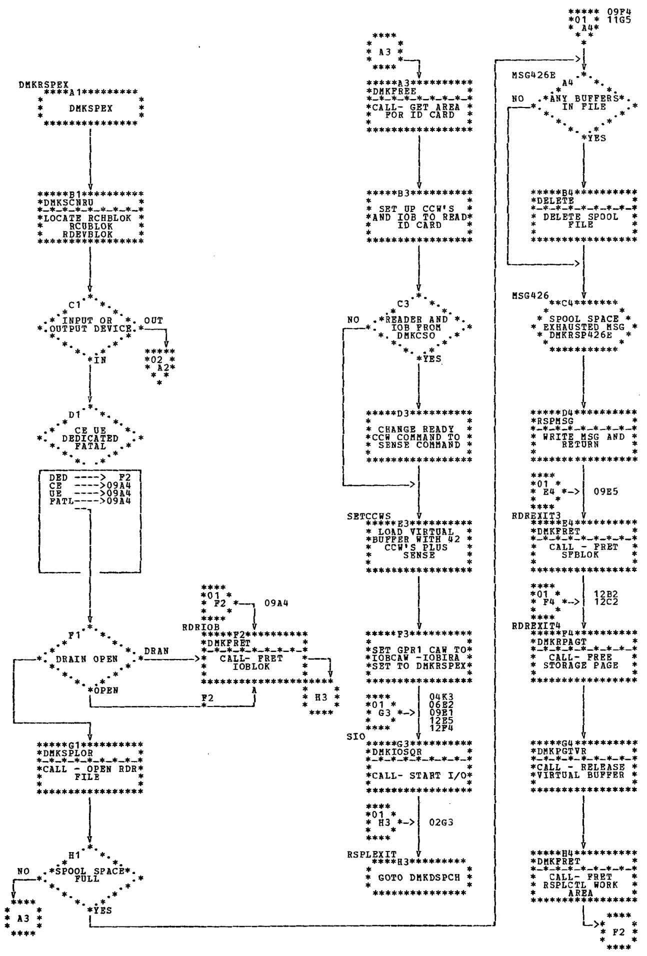

09 A1 08K2
GETCCW
A1
MOVE LAST 4 CCWS TO XOBRCCW AREA
09 B1 08G2
RT1
B1
IOB IOBIOER PRESENT
NO
YES
C1
CHAIN THIS IOERBLOK 2ND IN CHAIN
D1
UPDTCSW1
BAL R5 UPDATE IOERBLOK CSW
RT1A
E1
ROUTINE 1 SCHEDULED
NO
YES
RT2
E2
ROUTINE 2 SCHEDULED
NO
YES
RT3
E3
ROUTINE 3 SCHEDULED
NO
10 B3
YES
F1
SET UP TO GET READ CHECK BUFFER
F2
SET UP TO GET PRINT LINE BUFFER
F3
SET UP FOR 10 ERROR CHARACTERS
G1
RESET RT1 FLAG
G2
INDICATE TYPE 3 BUFFER RECORD
G3
RT3LP
G3
ROUTINE 3 SCHEDULED
YES
NO
RT4A
G4
PRT COMPLETE FOR THIS POS
NO
H2
YES
09 H1 10D3 10D4
ERROBR
H1
RESTORE RESTART CCW ADDRESS AND FLAGS
01 E3
RT3AA
H2
MOVE ERROR CHARACTER TO XOBR010
H2
H3
ERR CHECK FOR THIS POS
YES
NO
A
RT3A
H4
END OF PRINT LINE REACHED
YES
10 A3
NO
J2
10 CHARACTERS DONE
NO
YES
10 A3
J4
UPDATE PRINT LINE AND CHECK DATA ADDRESSES
G3

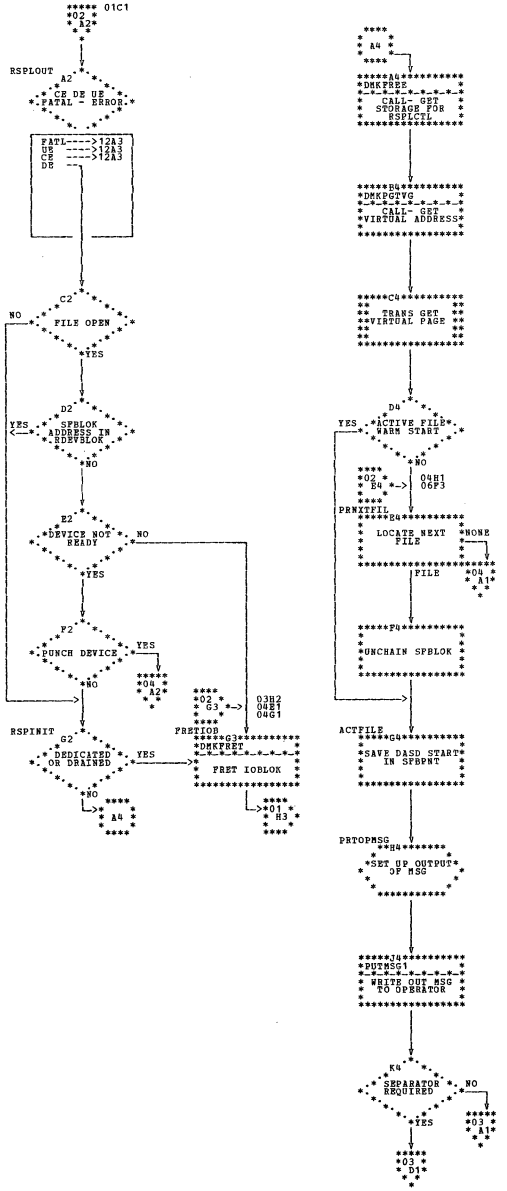


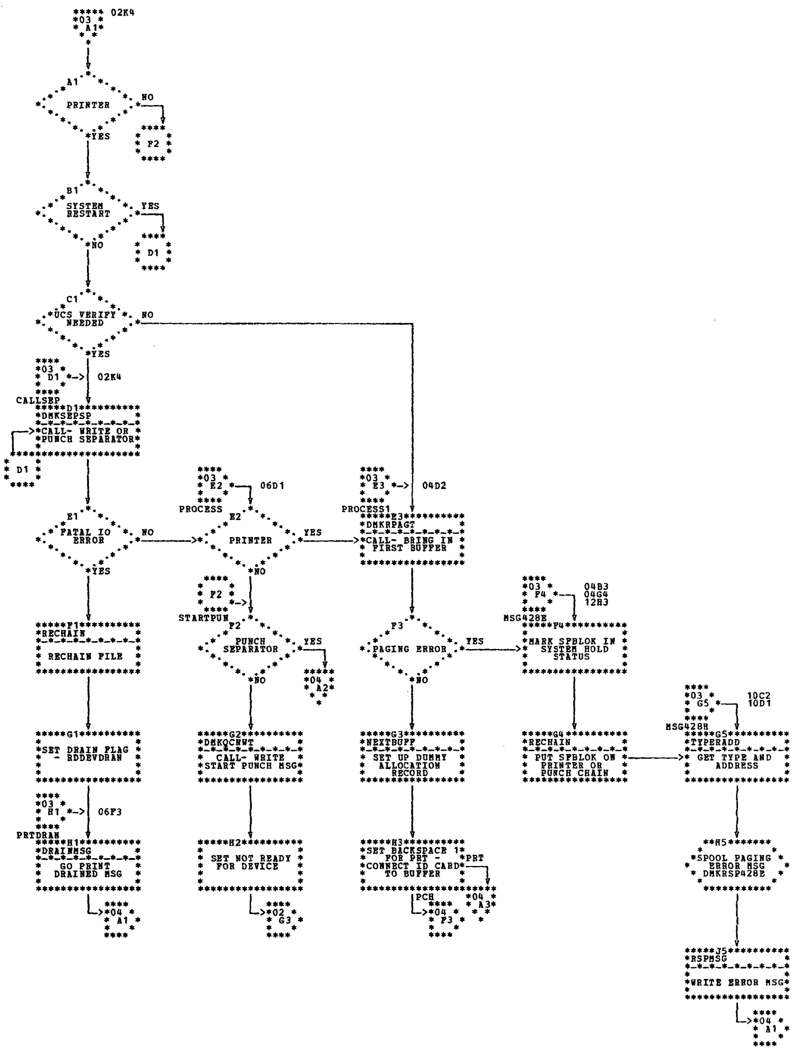


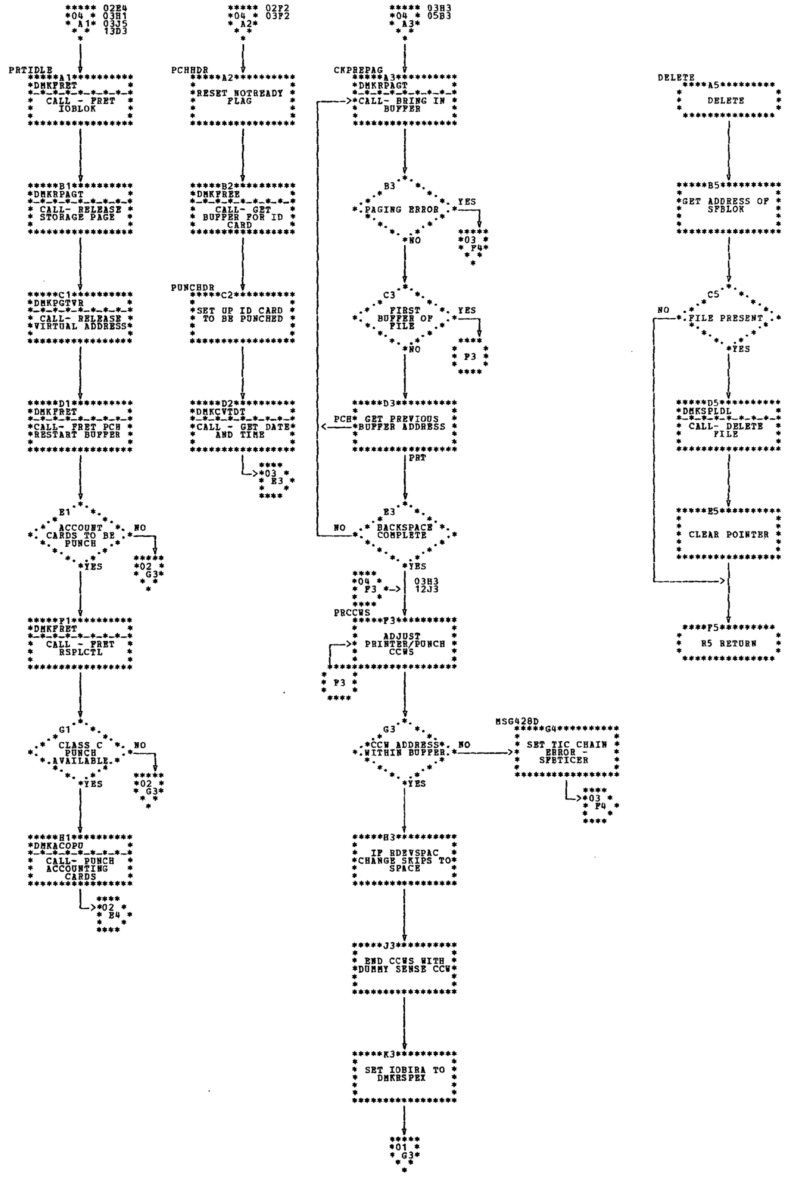
| DMKRSE -- Real Spool Error Procedures (Part 11 of 11)

DMKRSP -- Real Spooling Manager (Parts 1 and 2 of 13)



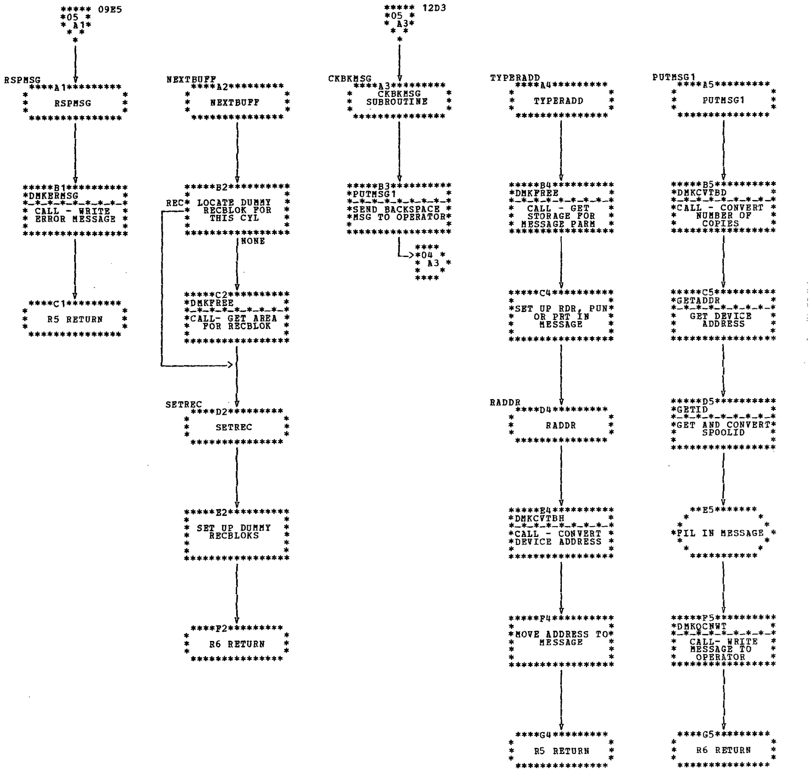


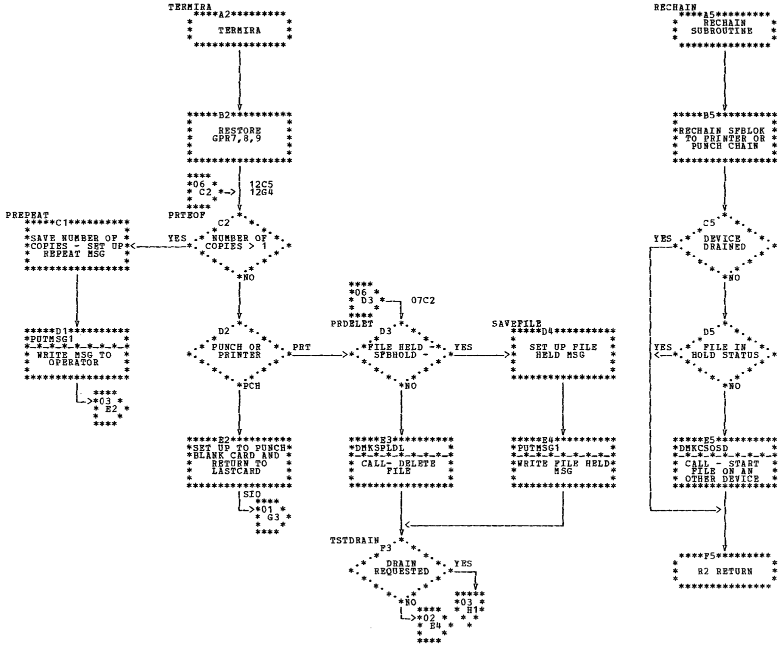


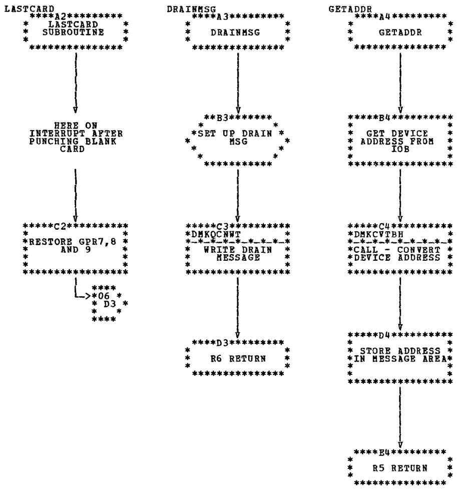


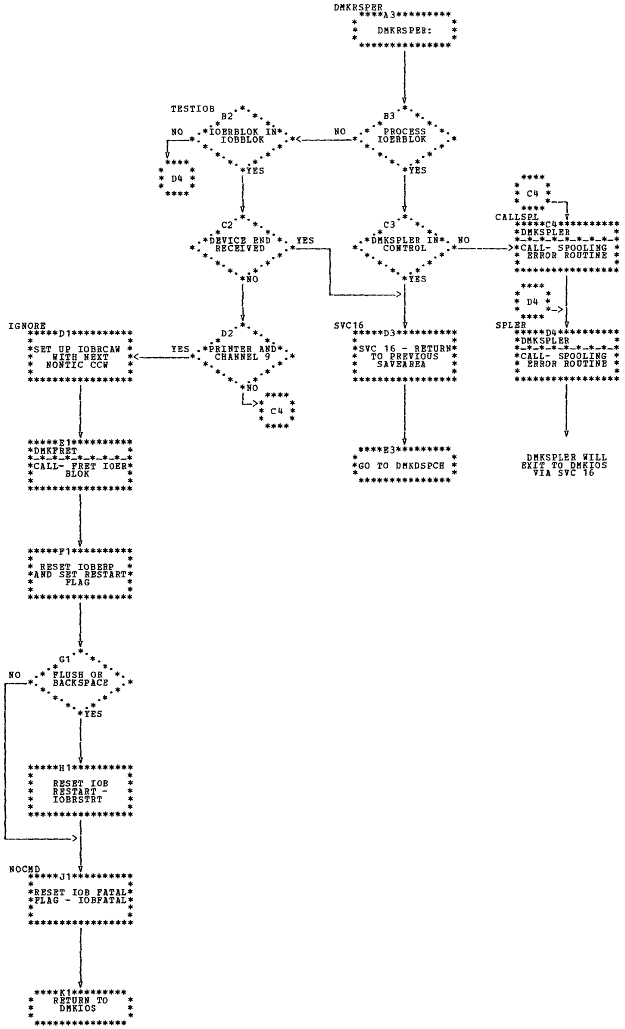
| DMKRSP -- Real Spooling Manager (Parts 3 and 4 of 13)

| DMKRSP -- Real Spooling Manager (Parts 5 and 6 of 13)



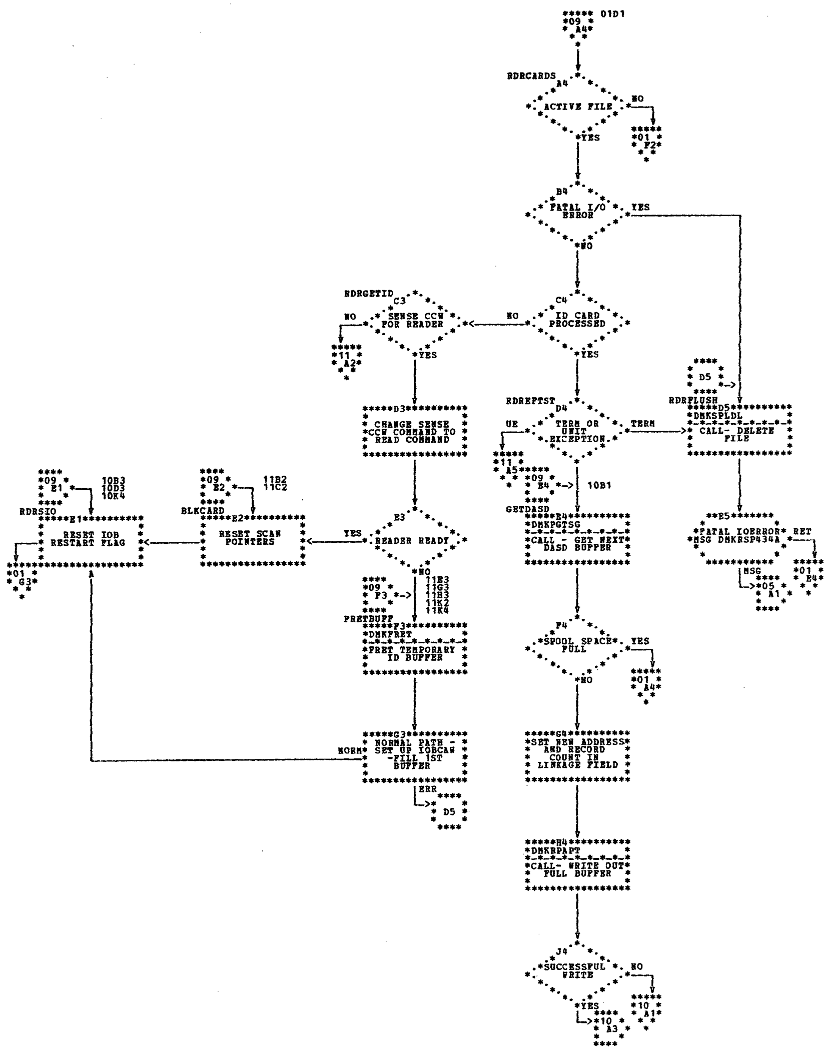


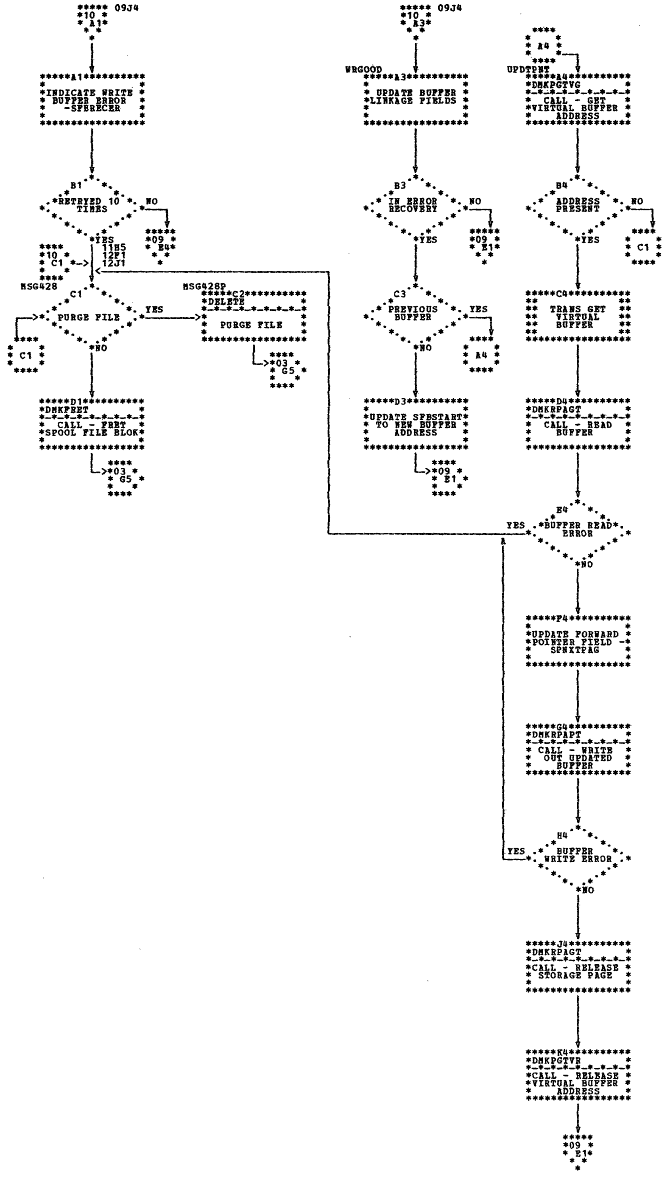


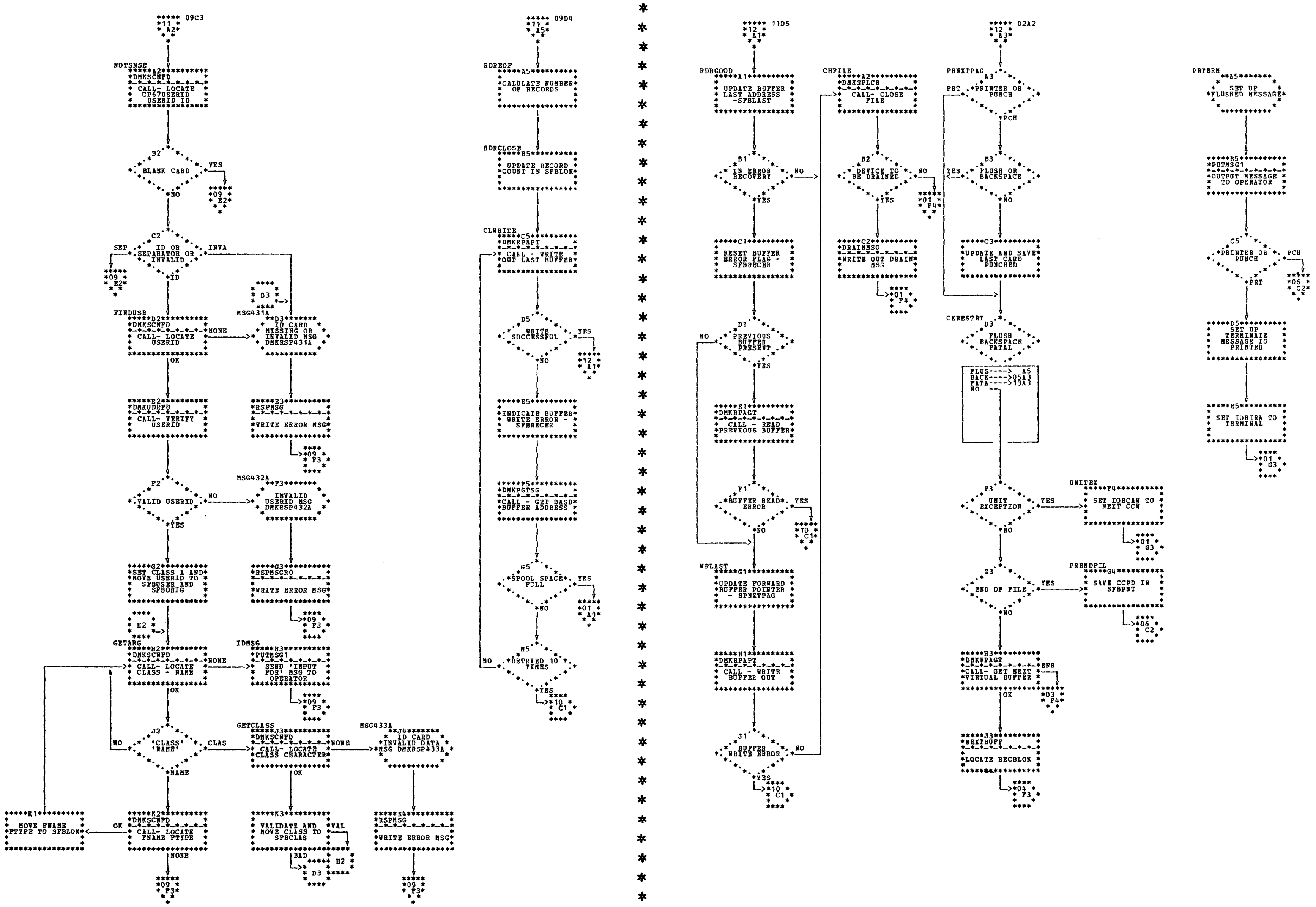


| DMKRSP -- Real Spooling Manager (Parts 7 and 8 of 13)

| DMKRSP -- Real Spooling Manager (Parts 9 and 10 of 13)

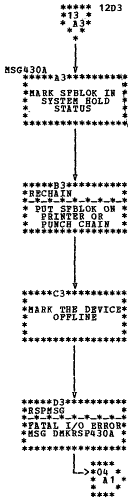


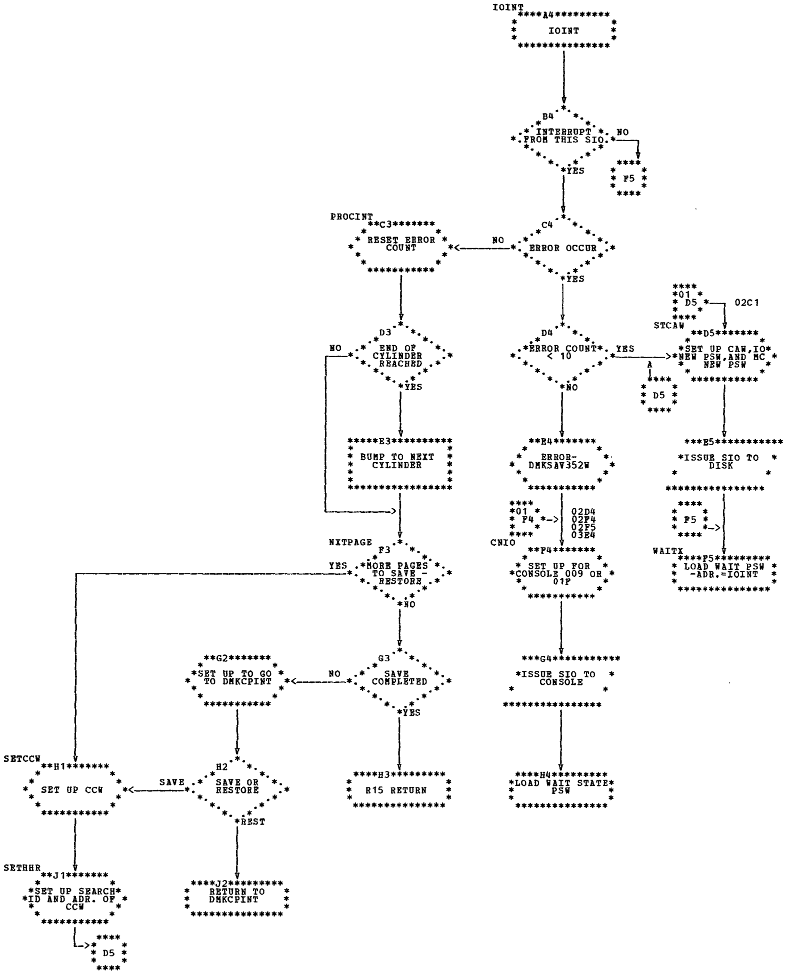


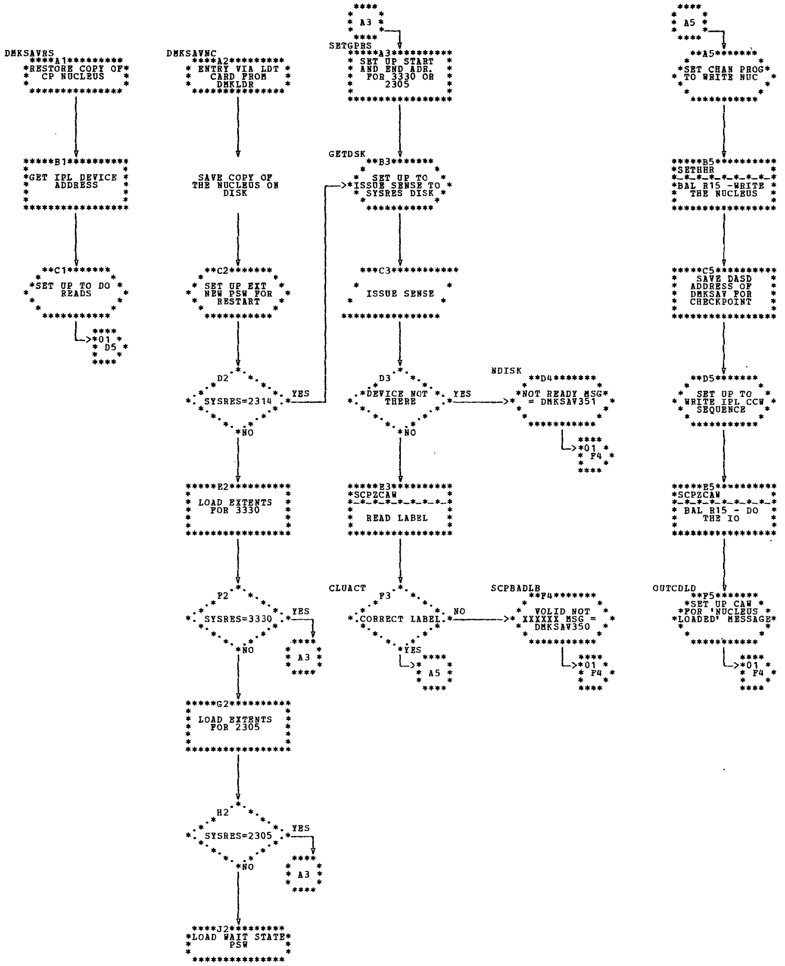


| DMKRSP -- Real Spooling Manager (Parts 11 and 12 of 13)

| DMKRSP -- Real Spooling Manager (Part 13 of 13)

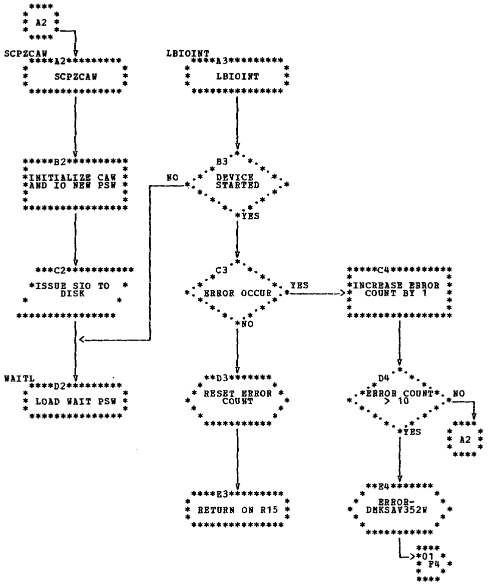


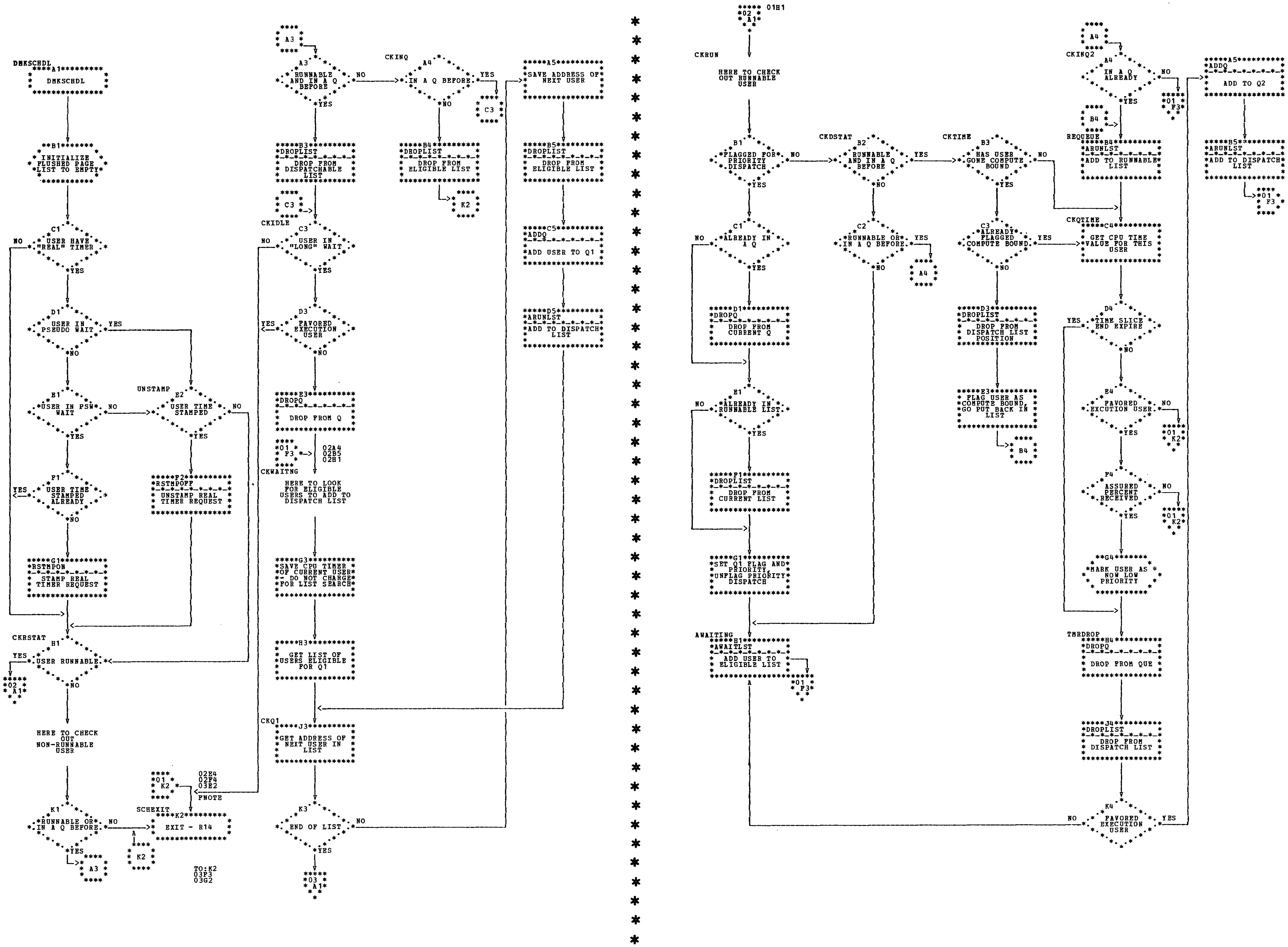



DMKSAVRS
A1 RESTORE COPY OF CP NUCLEUS
DMKSAVNC
A2 ENTRY VIA LDT CARD FROM DMKLDR
A3
SETGPRS
A3 SET UP START AND END ADR. FOR 3330 OR 2305
A5
A5 SET CHAN PROG TO WRITE NUC
B1 GET IPL DEVICE ADDRESS
SAVE COPY OF THE NUCLEUS ON DISK
GETDSK
B3 SET UP TO ISSUE SENSE TO SYSRES DISK
B5 SETHHR
BAL R15 -WRITE THE NUCLEUS
C1 SET UP TO DO READS
01 D5
C2 SET UP EXT NEW PSW FOR RESTART
C3 ISSUE SENSE
C5 SAVE DASD ADDRESS OF DMKSAV FOR CHECKPOINT
D2 SYSRES=2314
YES
NO
D3 DEVICE NOT THERE
YES
NO
NDISK
D4 NOT READY MSG = DMKSAV351
01 F4
D5 SET UP TO WRITE IPL CCW SEQUENCE
E2 LOAD EXTENTS FOR 3330
E3 SCPZCAW
READ LABEL
E5 SCPZCAW
BAL R15 - DO THE IO
F2 SYSRES=3330
YES
NO
A3
CLUACT
F3 CORRECT LABEL
NO
YES
A5
SCPBADLB
F4 VOLID NOT XXXXXX MSG = DMKSAV350
01 F4
OUTCDLD
F5 SET UP CAW FOR 'NUCLEUS LOADED' MESSAGE
01 F4
G2 LOAD EXTENTS FOR 2305
H2 SYSRES=2305
YES
NO
A3
J2 LOAD WAIT STATE PSW

DMKSAV -- Save CP Nucleus or SYSRES (Parts 1 and 2 of 3)

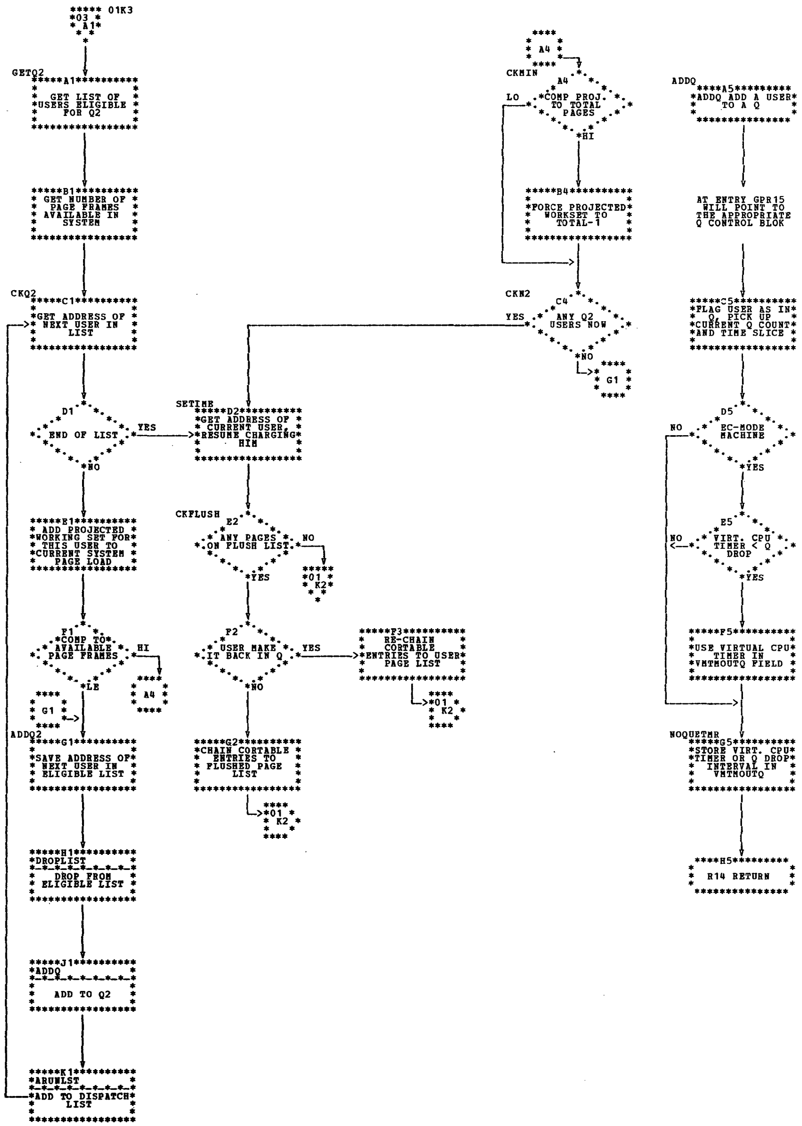
DMKSAV -- Save CP Nucleus or SYSRES (Part 3 of 3)

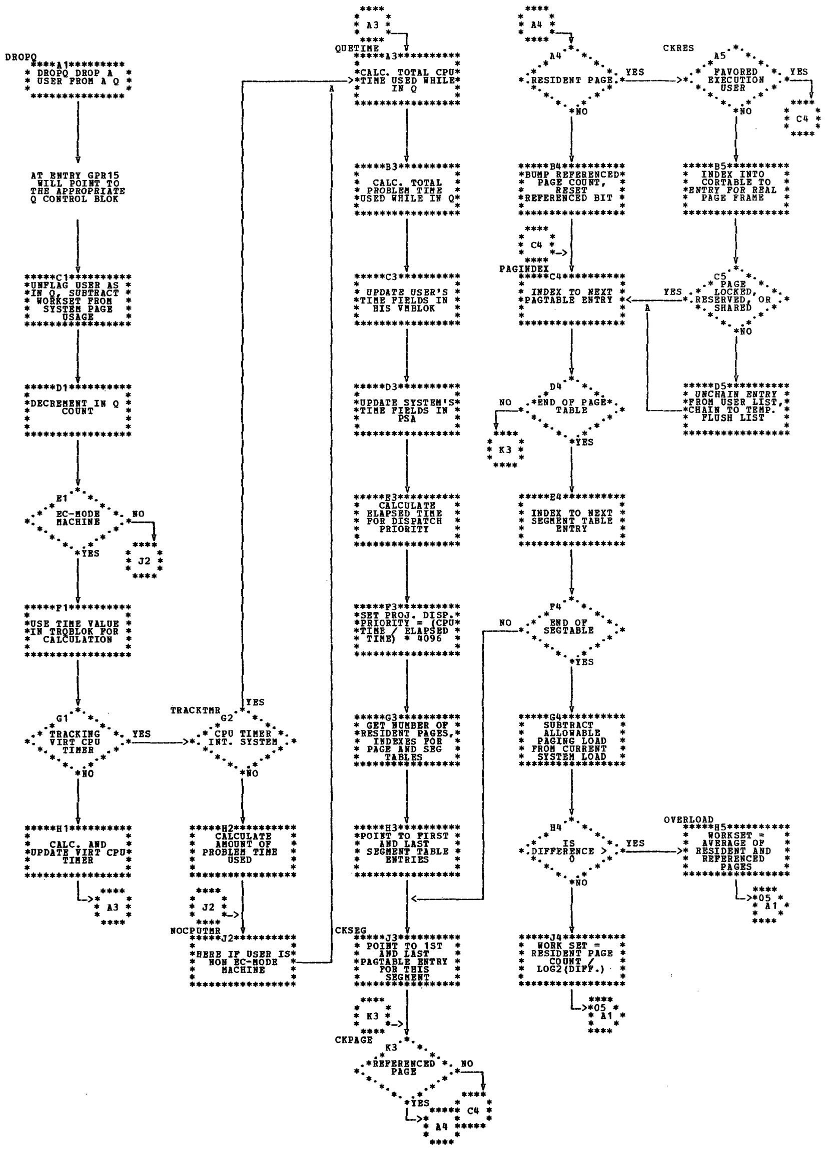


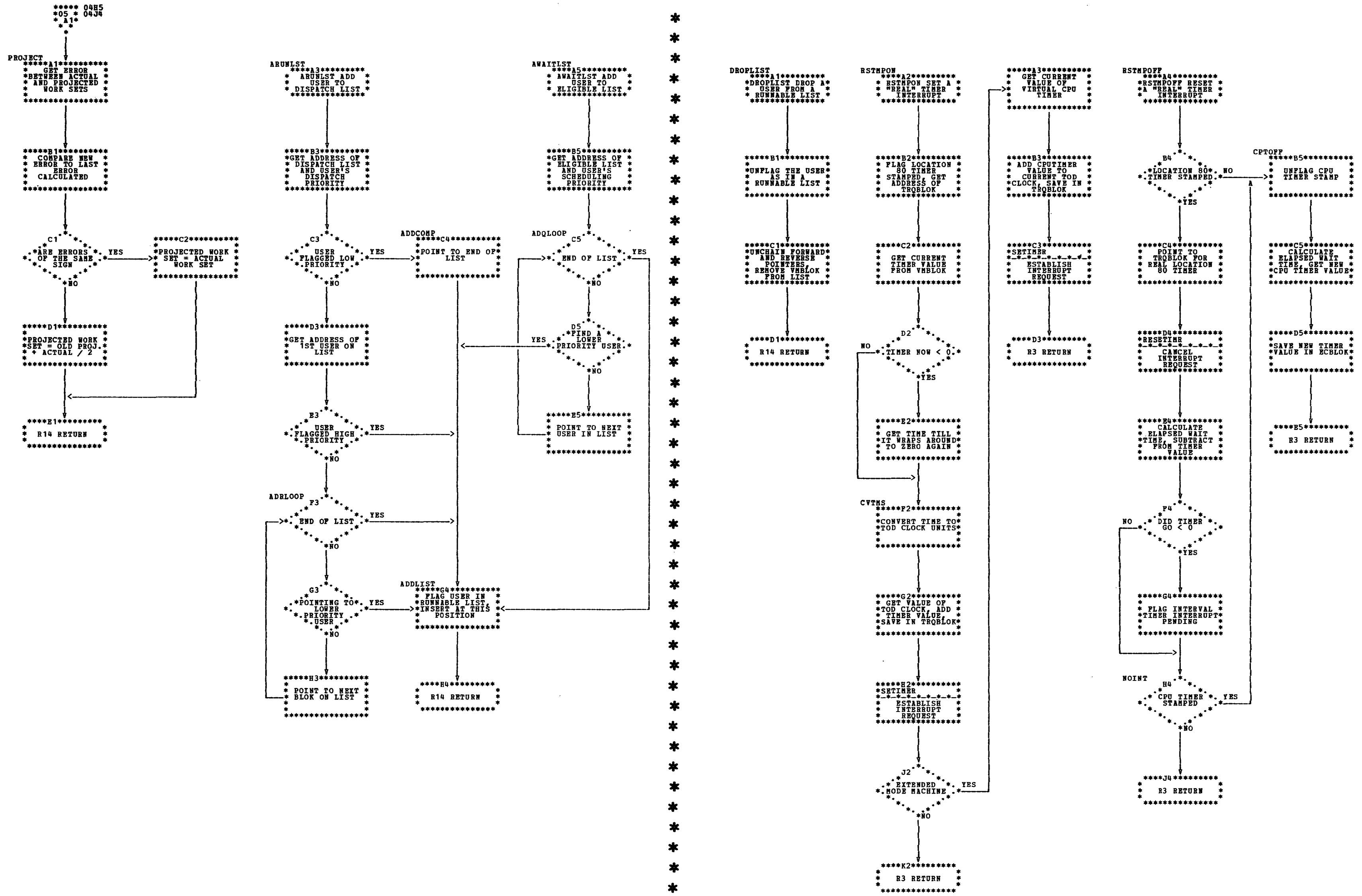


| DMKSCH -- Scheduler (Parts 1 and 2 of 8)

| DMKSCH -- Scheduler (Parts 3 and 4 of 8)

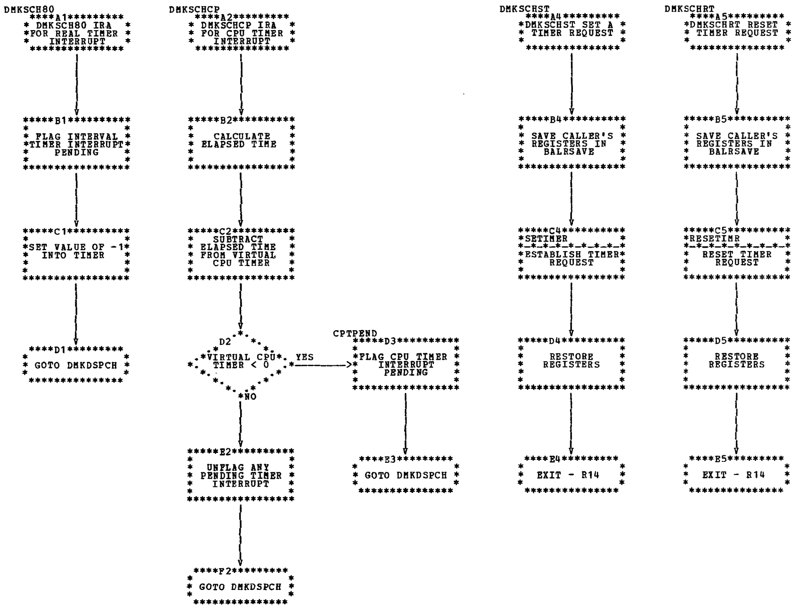

01K3
03 A1
GETQ2
A1 GET LIST OF USERS ELIGIBLE FOR Q2
B1 GET NUMBER OF PAGE FRAMES AVAILABLE IN SYSTEM
CKQ2
C1 GET ADDRESS OF NEXT USER IN LIST
D1 END OF LIST
YES
NO
E1 ADD PROJECTED WORKING SET FOR THIS USER TO CURRENT SYSTEM PAGE LOAD
F1 COMP TO AVAILABLE PAGE FRAMES
HI
A4
LE
G1
ADDQ2
G1 SAVE ADDRESS OF NEXT USER IN ELIGIBLE LIST
H1 DROPLIST
DROP FROM ELIGIBLE LIST
J1 ADDQ
ADD TO Q2
K1 ARUNLST
ADD TO DISPATCH LIST
SETIME
D2 GET ADDRESS OF CURRENT USER, RESUME CHARGING HIM
CKFLUSH
E2 ANY PAGES ON FLUSH LIST
NO
01 K2
YES
F2 USER MAKE IT BACK IN Q
YES
NO
F3 RE-CHAIN CORTABLE ENTRIES TO USER PAGE LIST
01 K2
G2 CHAIN CORTABLE ENTRIES TO FLUSHED PAGE LIST
01 K2
A4
CKMIN
A4 COMP PROJ. TO TOTAL PAGES
LO
HI
B4 FORCE PROJECTED WORKSET TO TOTAL-1
CKN2
C4 ANY Q2 USERS NOW
YES
NO
G1
ADDQ
A5 ADDQ ADD A USER TO A Q
AT ENTRY GPR15 WILL POINT TO THE APPROPRIATE Q CONTROL BLOK
C5 FLAG USER AS IN Q, PICK UP CURRENT Q COUNT AND TIME SLICE
D5 EC-MODE MACHINE
NO
YES
E5 VIRT. CPU TIMER < Q DROP
NO
YES
F5 USE VIRTUAL CPU TIMER IN VMTMOUTQ FIELD
NOQUETMR
G5 STORE VIRT. CPU TIMER OR Q DROP INTERVAL IN VMTMOUTQ
H5 R14 RETURN

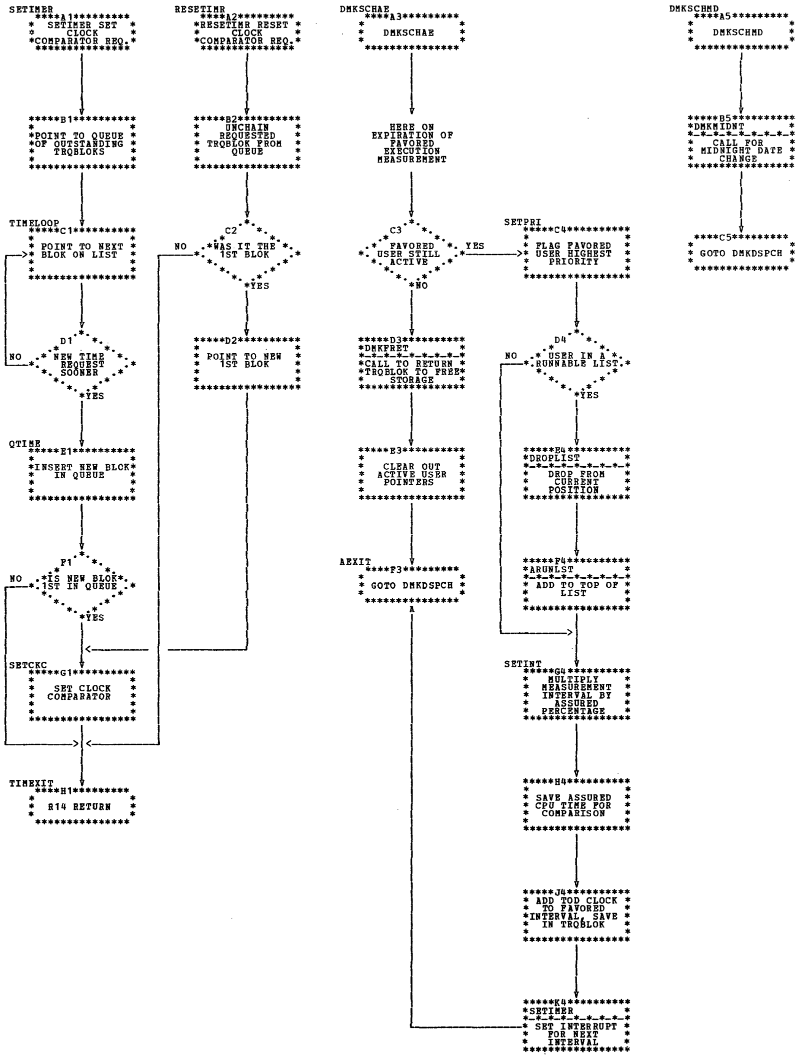

DROPQ
A1 DROPQ DROP A USER FROM A Q
AT ENTRY GPR15 WILL POINT TO THE APPROPRIATE Q CONTROL BLOK
C1 UNFLAG USER AS IN Q, SUBTRACT WORKSET FROM SYSTEM PAGE USAGE
D1 DECREMENT IN Q COUNT
E1 EC-MODE MACHINE
NO
J2
YES
F1 USE TIME VALUE IN TROBLOK FOR CALCULATION
G1 TRACKING VIRT CPU TIMER
YES
NO
H1 CALC. AND UPDATE VIRT CPU TIMER
A3
TRACKTMR
G2 CPU TIMER INT. SYSTEM
YES
NO
H2 CALCULATE AMOUNT OF PROBLEM TIME USED
J2
NOCPUTMR
J2 HERE IF USER IS NON EC-MODE MACHINE
A3
QUETIME
A3 CALC. TOTAL CPU TIME USED WHILE IN Q
A
B3 CALC. TOTAL PROBLEM TIME USED WHILE IN Q
C3 UPDATE USER'S TIME FIELDS IN HIS VMBLOK
D3 UPDATE SYSTEM'S TIME FIELDS IN PSA
E3 CALCULATE ELAPSED TIME FOR DISPATCH PRIORITY
F3 SET PROJ. DISP. PRIORITY = (CPU TIME / ELAPSED TIME) * 4096
G3 GET NUMBER OF RESIDENT PAGES, INDEXES FOR PAGE AND SEG TABLES
H3 POINT TO FIRST AND LAST SEGMENT TABLE ENTRIES
CKSEG
J3 POINT TO 1ST AND LAST PAGTABLE ENTRY FOR THIS SEGMENT
K3
CKPAGE
K3 REFERENCED PAGE
NO
C4
YES
A4
A4
A4 RESIDENT PAGE
YES
NO
B4 BUMP REFERENCED PAGE COUNT, RESET REFERENCED BIT
C4
PAGINDEX
C4 INDEX TO NEXT PAGTABLE ENTRY
D4 END OF PAGE TABLE
NO
K3
YES
E4 INDEX TO NEXT SEGMENT TABLE ENTRY
F4 END OF SEGTABLE
NO
YES
G4 SUBTRACT ALLOWABLE PAGING LOAD FROM CURRENT SYSTEM LOAD
H4 IS DIFFERENCE > 0
YES
NO
J4 WORK SET = RESIDENT PAGE COUNT / LOG2(DIFF.)
05 A1
CKRES
A5 FAVORED EXECUTION USER
YES
C4
NO
B5 INDEX INTO CORTABLE TO ENTRY FOR REAL PAGE FRAME
C5 PAGE LOCKED, RESERVED, OR SHARED
YES
NO
A
D5 UNCHAIN ENTRY FROM USER LIST, CHAIN TO TEMP. FLUSH LIST
OVERLOAD
H5 WORKSET = AVERAGE OF RESIDENT AND REFERENCED PAGES
05 A1

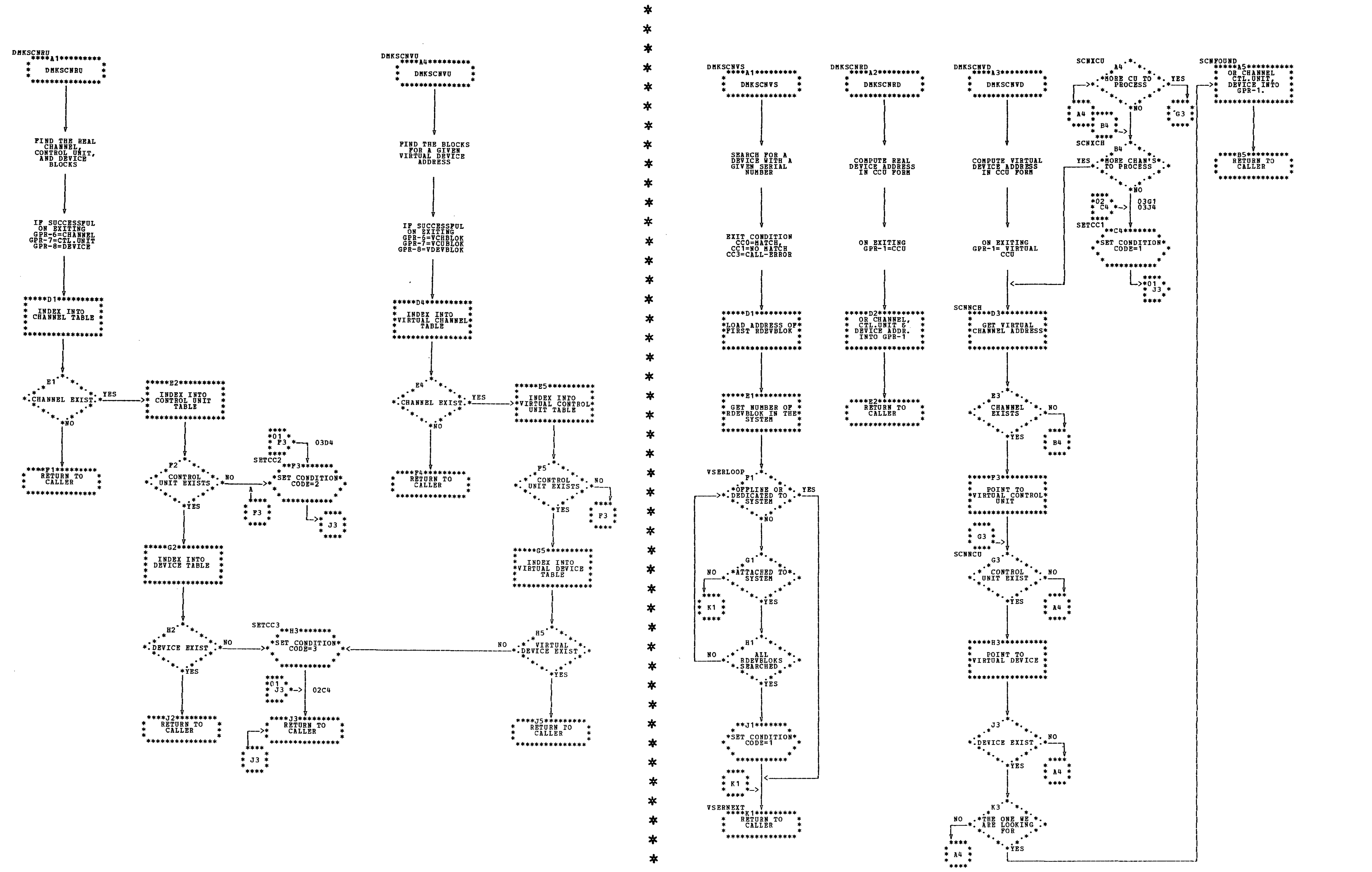

04H5
05
04J4
A1
PROJECT
A1
GET ERROR BETWEEN ACTUAL AND PROJECTED WORK SETS
B1
COMPARE NEW ERROR TO LAST ERROR CALCULATED
C1
ARE ERRORS OF THE SAME SIGN
YES
NO
C2
PROJECTED WORK SET = ACTUAL WORK SET
D1
PROJECTED WORK SET = OLD PROJ. + ACTUAL / 2
E1
R14 RETURN
ARUNLST
A3
ARUNLST ADD USER TO DISPATCH LIST
B3
GET ADDRESS OF DISPATCH LIST AND USER'S DISPATCH PRIORITY
C3
USER FLAGGED LOW PRIORITY
YES
NO
ADDCOMP
C4
POINT TO END OF LIST
D3
GET ADDRESS OF 1ST USER ON LIST
E3
USER FLAGGED HIGH PRIORITY
YES
NO
ADRLOOP
F3
END OF LIST
YES
NO
G3
POINTING TO LOWER PRIORITY USER
YES
NO
ADDLIST
G4
FLAG USER IN RUNNABLE LIST, INSERT AT THIS POSITION
H3
POINT TO NEXT BLOK ON LIST
H4
R14 RETURN
AWAITLST
A5
AWAITLST ADD USER TO ELIGIBLE LIST
B5
GET ADDRESS OF ELIGIBLE LIST AND USER'S SCHEDULING PRIORITY
ADQLOOP
C5
END OF LIST
YES
NO
D5
FIND A LOWER PRIORITY USER
YES
NO
E5
POINT TO NEXT USER IN LIST
DROPLIST
A1
DROPLIST DROP A USER FROM A RUNNABLE LIST
B1
UNFLAG THE USER AS IN A RUNNABLE LIST
C1
UNCHAIN FORWARD AND REVERSE POINTERS, REMOVE VMBLOK FROM LIST
D1
R14 RETURN
RSTMPON
A2
RSTMPON SET A "REAL" TIMER INTERRUPT
B2
FLAG LOCATION 80 TIMER STAMPED, GET ADDRESS OF TRQBLOK
C2
GET CURRENT TIMER VALUE FROM VMBLOK
D2
TIMER NOW < 0
NO
YES
E2
GET TIME TILL IT WRAPS AROUND TO ZERO AGAIN
CVTMS
F2
CONVERT TIME TO TOD CLOCK UNITS
G2
GET VALUE OF TOD CLOCK, ADD TIMER VALUE, SAVE IN TRQBLOK
H2
SETIMER
ESTABLISH INTERRUPT REQUEST
J2
EXTENDED MODE MACHINE
YES
NO
K2
R3 RETURN
A3
GET CURRENT VALUE OF VIRTUAL CPU TIMER
B3
ADD CPUTIMER VALUE TO CURRENT TOD CLOCK, SAVE IN TRQBLOK
C3
SETIMER
ESTABLISH INTERRUPT REQUEST
D3
R3 RETURN
RSTMPOFF
A4
RSTMPOFF RESET A "REAL" TIMER INTERRUPT
B4
LOCATION 80 TIMER STAMPED
NO
YES
C4
POINT TO TRQBLOK FOR REAL LOCATION 80 TIMER
D4
RESETIMR
CANCEL INTERRUPT REQUEST
E4
CALCULATE ELAPSED WAIT TIME, SUBTRACT FROM TIMER VALUE
F4
DID TIMER GO < 0
NO
YES
G4
FLAG INTERVAL TIMER INTERRUPT PENDING
NOINT
H4
CPU TIMER STAMPED
YES
NO
J4
R3 RETURN
CPTOFF
B5
UNFLAG CPU TIMER STAMP
C5
CALCULATE ELAPSED WAIT TIME, GET NEW CPU TIMER VALUE
D5
SAVE NEW TIMER VALUE IN ECBLOK
B5
R3 RETURN

| DMKSCH -- Scheduler (Parts 5 and 6 of 8)

| DMKSCH -- Scheduler (Parts 7 and 8 of 8)

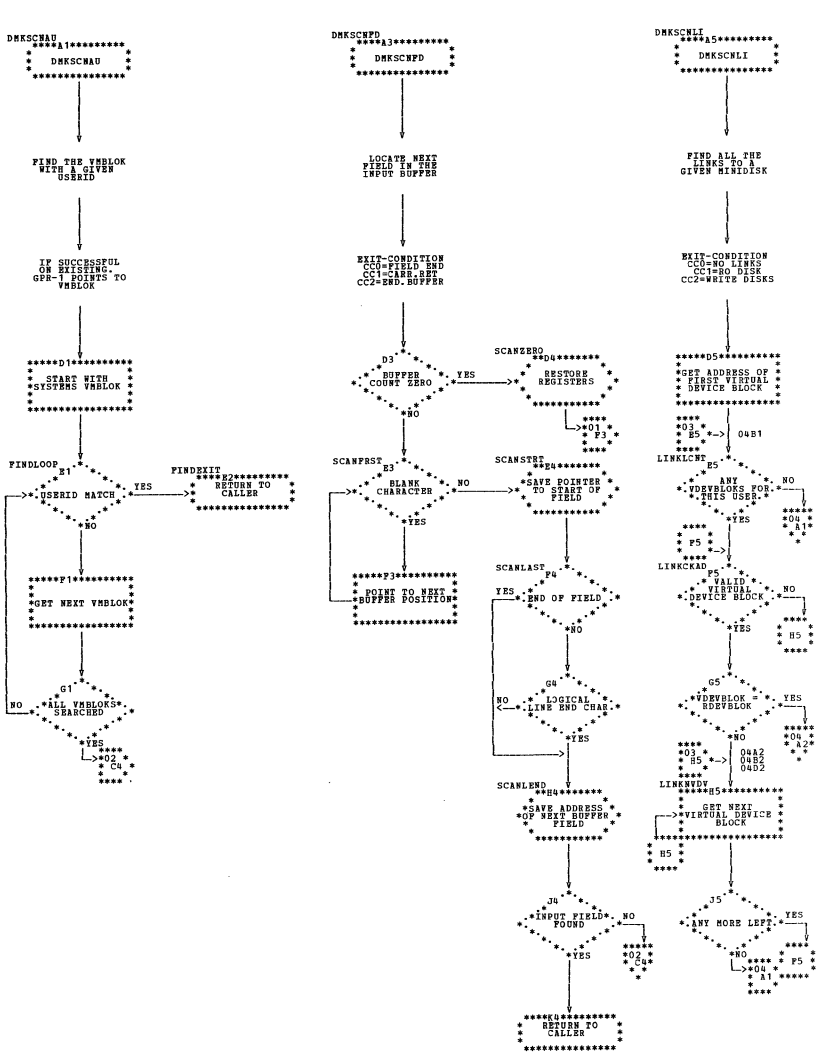

DMKSCH80
A1
DMKSCH80 IRA FOR REAL TIMER INTERRUPT
B1
FLAG INTERVAL TIMER INTERRUPT PENDING
C1
SET VALUE OF -1 INTO TIMER
D1
GOTO DMKDSPCH
DMKSCHCP
A2
DMKSCHCP IRA FOR CPU TIMER INTERRUPT
B2
CALCULATE ELAPSED TIME
C2
SUBTRACT ELAPSED TIME FROM VIRTUAL CPU TIMER
D2
VIRTUAL CPU TIMER < 0
YES
NO
E2
UNFLAG ANY PENDING TIMER INTERRUPT
F2
GOTO DMKDSPCH
CPTPEND
D3
FLAG CPU TIMER INTERRUPT PENDING
E3
GOTO DMKDSPCH
DMKSCHST
A4
DMKSCHST SET A TIMER REQUEST
B4
SAVE CALLER'S REGISTERS IN BALRSAVE
C4
SETIMER
ESTABLISH TIMER REQUEST
D4
RESTORE REGISTERS
E4
EXIT - R14
DMKSCHRT
A5
DMKSCHRT RESET TIMER REQUEST
B5
SAVE CALLER'S REGISTERS IN BALRSAVE
C5
RESETIMR
RESET TIMER REQUEST
D5
RESTORE REGISTERS
E5
EXIT - R14

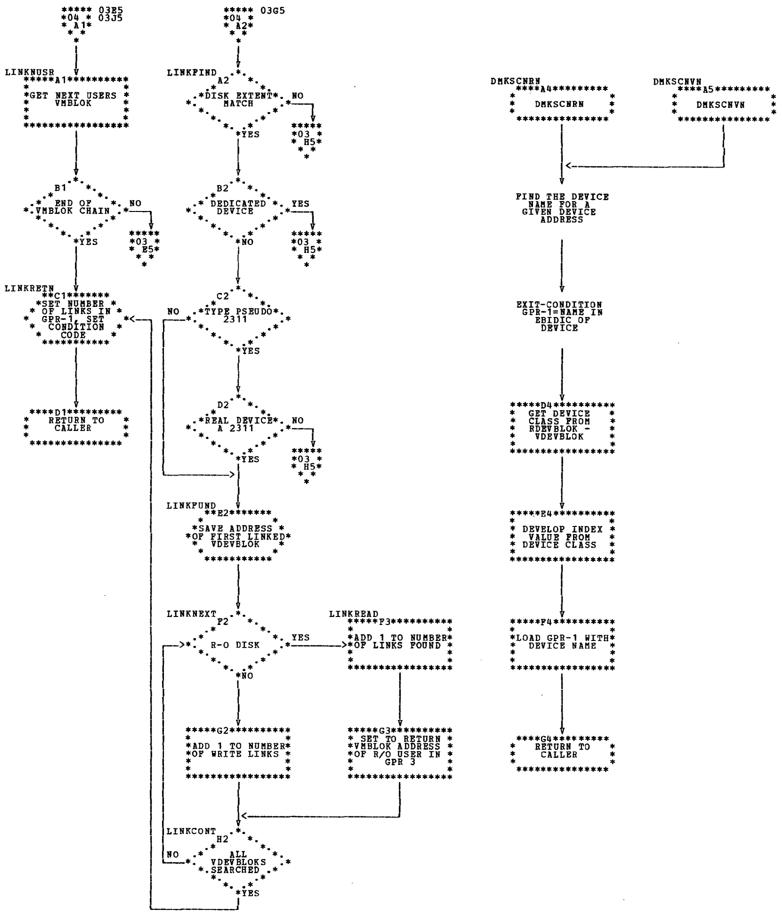

SETIMER
A1
SETIMER SET CLOCK COMPARATOR REQ.
B1
POINT TO QUEUE OF OUTSTANDING TRQBLOKS
TIMELOOP
C1
POINT TO NEXT BLOK ON LIST
D1
NEW TIME REQUEST SOONER
NO
YES
QTIME
E1
INSERT NEW BLOK IN QUEUE
F1
IS NEW BLOK 1ST IN QUEUE
NO
YES
SETCKC
G1
SET CLOCK COMPARATOR
TIMEXIT
H1
R14 RETURN
RESETIMR
A2
RESETIMR RESET CLOCK COMPARATOR REQ.
B2
UNCHAIN REQUESTED TRQBLOK FROM QUEUE
C2
WAS IT THE 1ST BLOK
NO
YES
D2
POINT TO NEW 1ST BLOK
DMKSCHAE
A3
DMKSCHAE
HERE ON EXPIRATION OF FAVORED EXECUTION MEASUREMENT
C3
FAVORED USER STILL ACTIVE
YES
NO
D3
DMKFRET
CALL TO RETURN TRQBLOK TO FREE STORAGE
E3
CLEAR OUT ACTIVE USER POINTERS
AEXIT
F3
GOTO DMKDSPCH
A
SETPRI
C4
FLAG FAVORED USER HIGHEST PRIORITY
D4
USER IN A RUNNABLE LIST
NO
YES
E4
DROPLIST
DROP FROM CURRENT POSITION
F4
ARUNLST
ADD TO TOP OF LIST
SETINT
G4
MULTIPLY MEASUREMENT INTERVAL BY ASSURED PERCENTAGE
H4
SAVE ASSURED CPU TIME FOR COMPARISON
J4
ADD TOD CLOCK TO FAVORED INTERVAL, SAVE IN TRQBLOK
K4
SETIMER
SET INTERRUPT FOR NEXT INTERVAL
DMKSCHMD
A5
DMKSCHMD
B5
DMKMIDNT
CALL FOR MIDNIGHT DATE CHANGE
C5
GOTO DMKDSPCH

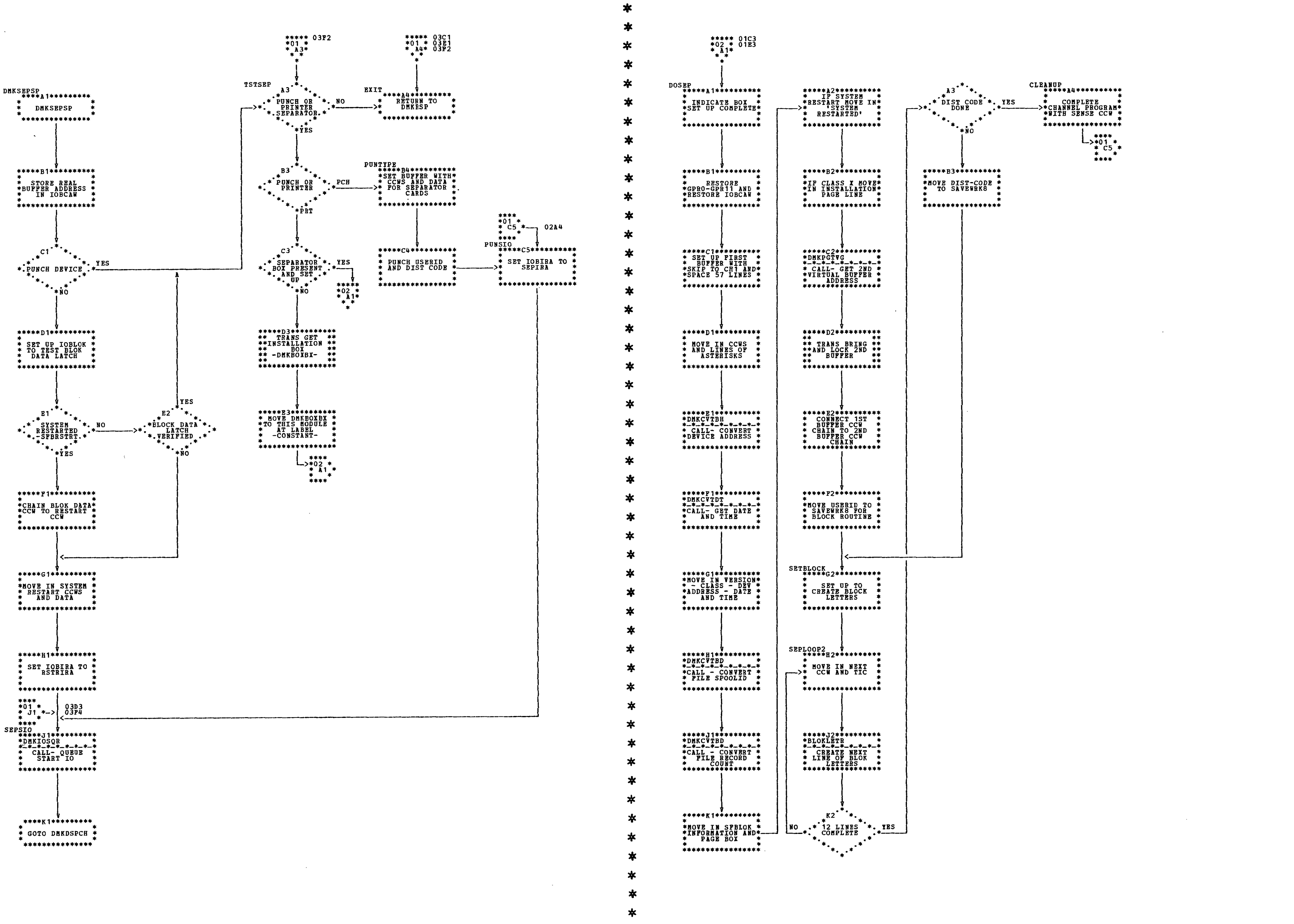


| DMKSCN -- Scan Routines (Parts 1 and 2 of 4)

DMKSCN -- Scan Routines (Parts 3 and 4 of 4)

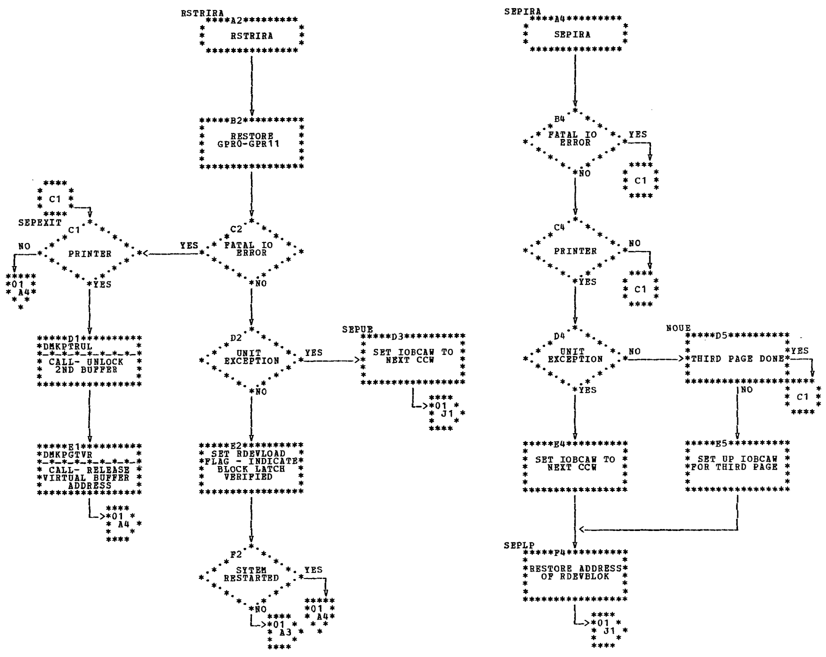

DMKSCNAU
A1
DMKSCNAU
FIND THE VMBLOK WITH A GIVEN USERID
IF SUCCESSFUL ON EXISTING, GPR-1 POINTS TO VMBLOK
D1
START WITH SYSTEMS VMBLOK
FINDLOOP
E1
USERID MATCH
YES
NO
FINDEXIT
E2
RETURN TO CALLER
F1
GET NEXT VMBLOK
G1
ALL VMBLOKS SEARCHED
NO
YES
02 C4
DMKSCNFD
A3
DMKSCNFD
LOCATE NEXT FIELD IN THE INPUT BUFFER
EXIT-CONDITION CC0=FIELD END CC1=CARR.RET CC2=END.BUFFER
D3
BUFFER COUNT ZERO
YES
NO
SCANZERO
D4
RESTORE REGISTERS
01 F3
SCANFRST
E3
BLANK CHARACTER
NO
YES
SCANSTRT
E4
SAVE POINTER TO START OF FIELD
F3
POINT TO NEXT BUFFER POSITION
SCANLAST
F4
END OF FIELD
YES
NO
G4
LOGICAL LINE END CHAR.
NO
YES
SCANLEND
H4
SAVE ADDRESS OF NEXT BUFFER FIELD
J4
INPUT FIELD FOUND
NO
YES
02 C4
K4
RETURN TO CALLER
DMKSCNLI
A5
DMKSCNLI
FIND ALL THE LINKS TO A GIVEN MINIDISK
EXIT-CONDITION CC0=NO LINKS CC1=RO DISK CC2=WRITE DISKS
D5
GET ADDRESS OF FIRST VIRTUAL DEVICE BLOCK
03 E5
04B1
LINKLCNT
E5
ANY VDEVBLOKS FOR THIS USER.
NO
YES
04 A1
F5
LINKCKAD
F5
VALID VIRTUAL DEVICE BLOCK
NO
YES
H5
G5
VDEVBLOK = RDEVBLOK
YES
NO
04 A2
03 H5
04A2
04B2
04D2
LINKNVDV
H5
GET NEXT VIRTUAL DEVICE BLOCK
H5
J5
ANY MORE LEFT
YES
NO
F5
04 A1

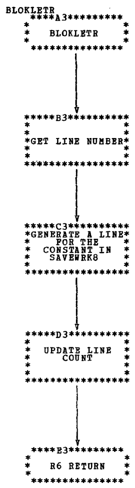

03E5
03J5
04 A1
LINKNUSR
A1
GET NEXT USERS VMBLOK
B1
END OF VMBLOK CHAIN
NO
YES
03 E5
LINKRETN
C1
SET NUMBER OF LINKS IN GPR-1, SET CONDITION CODE
D1
RETURN TO CALLER
03G5
04 A2
LINKFIND
A2
DISK EXTENT MATCH
NO
YES
03 H5
B2
DEDICATED DEVICE
YES
NO
03 H5
C2
TYPE PSEUDO 2311
NO
YES
D2
REAL DEVICE A 2311
NO
YES
03 H5
LINKFUND
E2
SAVE ADDRESS OF FIRST LINKED VDEVBLOK
LINKNEXT
F2
R-O DISK
YES
NO
LINKREAD
F3
ADD 1 TO NUMBER OF LINKS FOUND
G2
ADD 1 TO NUMBER OF WRITE LINKS
G3
SET TO RETURN VMBLOK ADDRESS OF R/O USER IN GPR 3
LINKCONT
H2
ALL VDEVBLOKS SEARCHED
NO
YES
DMKSCNRN
A4
DMKSCNRN
DMKSCNVN
A5
DMKSCNVN
FIND THE DEVICE NAME FOR A GIVEN DEVICE ADDRESS
EXIT-CONDITION GPR-1=NAME IN EBCDIC OF DEVICE
D4
GET DEVICE CLASS FROM RDEVBLOK - VDEVBLOK
E4
DEVELOP INDEX VALUE FROM DEVICE CLASS
F4
LOAD GPR-1 WITH DEVICE NAME
G4
RETURN TO CALLER

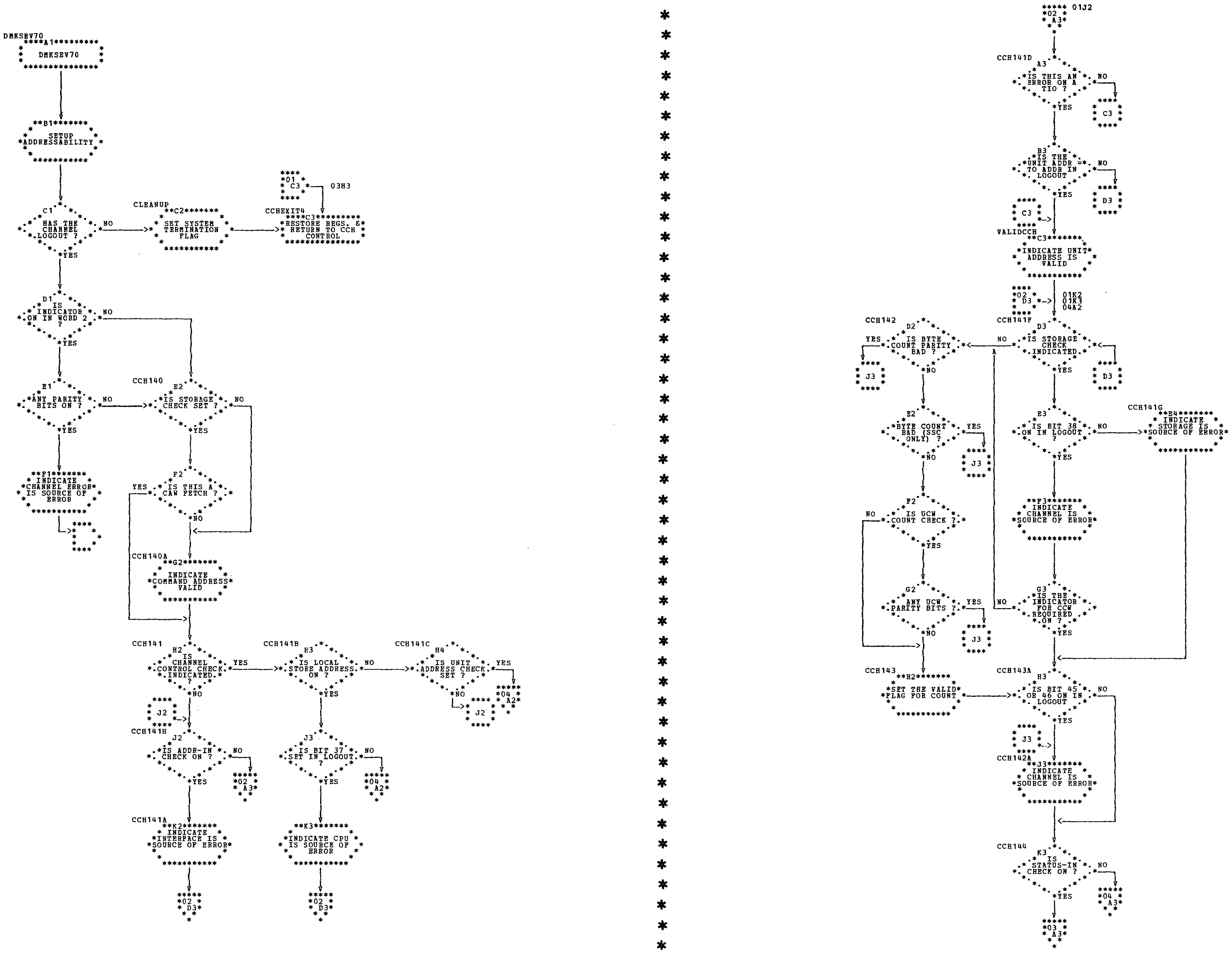

DMKSEPSP
A1
DMKSEPSP
B1
STORE REAL BUFFER ADDRESS IN IOBCAW
C1
PUNCH DEVICE
YES
NO
D1
SET UP IOBLOK TO TEST BLOK DATA LATCH
E1
SYSTEM RESTARTED -SFBRSTRT
NO
YES
E2
BLOCK DATA LATCH VERIFIED
YES
NO
F1
CHAIN BLOK DATA CCW TO RESTART CCW
G1
MOVE IN SYSTEM RESTART CCWS AND DATA
H1
SET IOBIRA TO RSTRIRA
01 J1 03D3 03F4
SEPSIO
J1
DMKIOSQR
CALL- QUEUE START IO
K1
GOTO DMKDSPCH
01 A3 03F2
TSTSEP
A3
PUNCH OR PRINTER SEPARATOR
NO
YES
B3
PUNCH OR PRINTER
PCH
PRT
C3
SEPARATOR BOX PRESENT AND SET UP
YES
NO
02 A1
D3
TRANS GET INSTALLATION BOX -DMKBOXBX-
E3
MOVE DMKBOXBX TO THIS MODULE AT LABEL -CONSTANT-
02 A1
01 A4 03C1 03E1 03F2
EXIT
A4
RETURN TO DMKRSP
PUNTYPE
B4
SET BUFFER WITH CCWS AND DATA FOR SEPARATOR CARDS
C4
PUNCH USERID AND DIST CODE
01 C5 02A4
PUNSIO
C5
SET IOBIRA TO SEPIRA
02 A1 01C3 01E3
DOSEP
A1
INDICATE BOX SET UP COMPLETE
B1
RESTORE GPR0-GPR11 AND RESTORE IOBCAW
C1
SET UP FIRST BUFFER WITH SKIP TO CH1 AND SPACE 57 LINES
D1
MOVE IN CCWS AND LINES OF ASTERISKS
E1
DMKCVTBH
CALL- CONVERT DEVICE ADDRESS
F1
DMKCVTDT
CALL- GET DATE AND TIME
G1
MOVE IN VERSION - CLASS - DEV ADDRESS - DATE AND TIME
H1
DMKCVTBD
CALL - CONVERT FILE SPOOLID
J1
DMKCVTBD
CALL - CONVERT FILE RECORD COUNT
K1
MOVE IN SFBLOK INFORMATION AND PAGE BOX
A2
IF SYSTEM RESTART MOVE IN 'SYSTEM RESTARTED'
B2
IF CLASS X MOVE IN INSTALLATION PAGE LINE
C2
DMKPGTVG
CALL- GET 2ND VIRTUAL BUFFER ADDRESS
D2
TRANS BRING AND LOCK 2ND BUFFER
E2
CONNECT 1ST BUFFER CCW CHAIN TO 2ND BUFFER CCW CHAIN
F2
MOVE USERID TO SAVEWRK8 FOR BLOCK ROUTINE
SETBLOCK
G2
SET UP TO CREATE BLOCK LETTERS
SEPLOOP2
H2
MOVE IN NEXT CCW AND TIC
J2
BLOKLETR
CREATE NEXT LINE OF BLOK LETTERS
K2
12 LINES COMPLETE
NO
YES
A3
DIST CODE DONE
YES
NO
B3
MOVE DIST-CODE TO SAVEWRK8
CLEANUP
A4
COMPLETE CHANNEL PROGRAM WITH SENSE CCW
01 C5

DMKSEP -- Print/Punch Output Separator (Parts 1 and 2 of 4)

DMKSEP -- Print/Punch Output Separator (Parts 3 and 4 of 4)

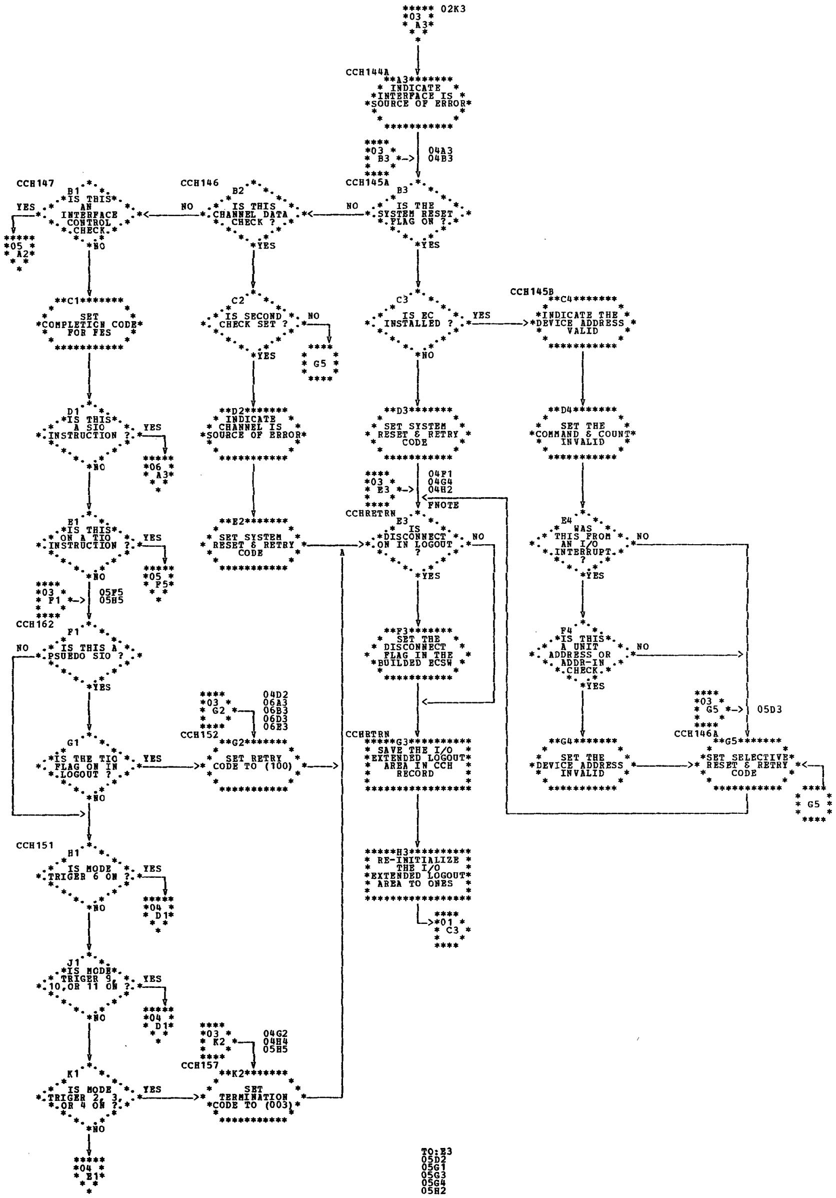


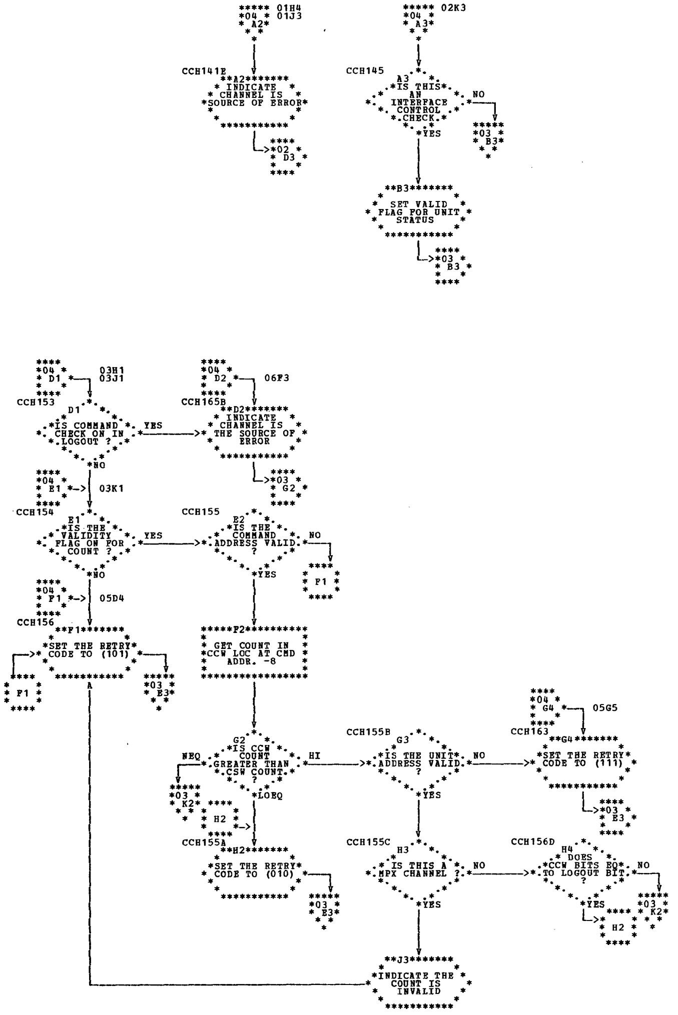


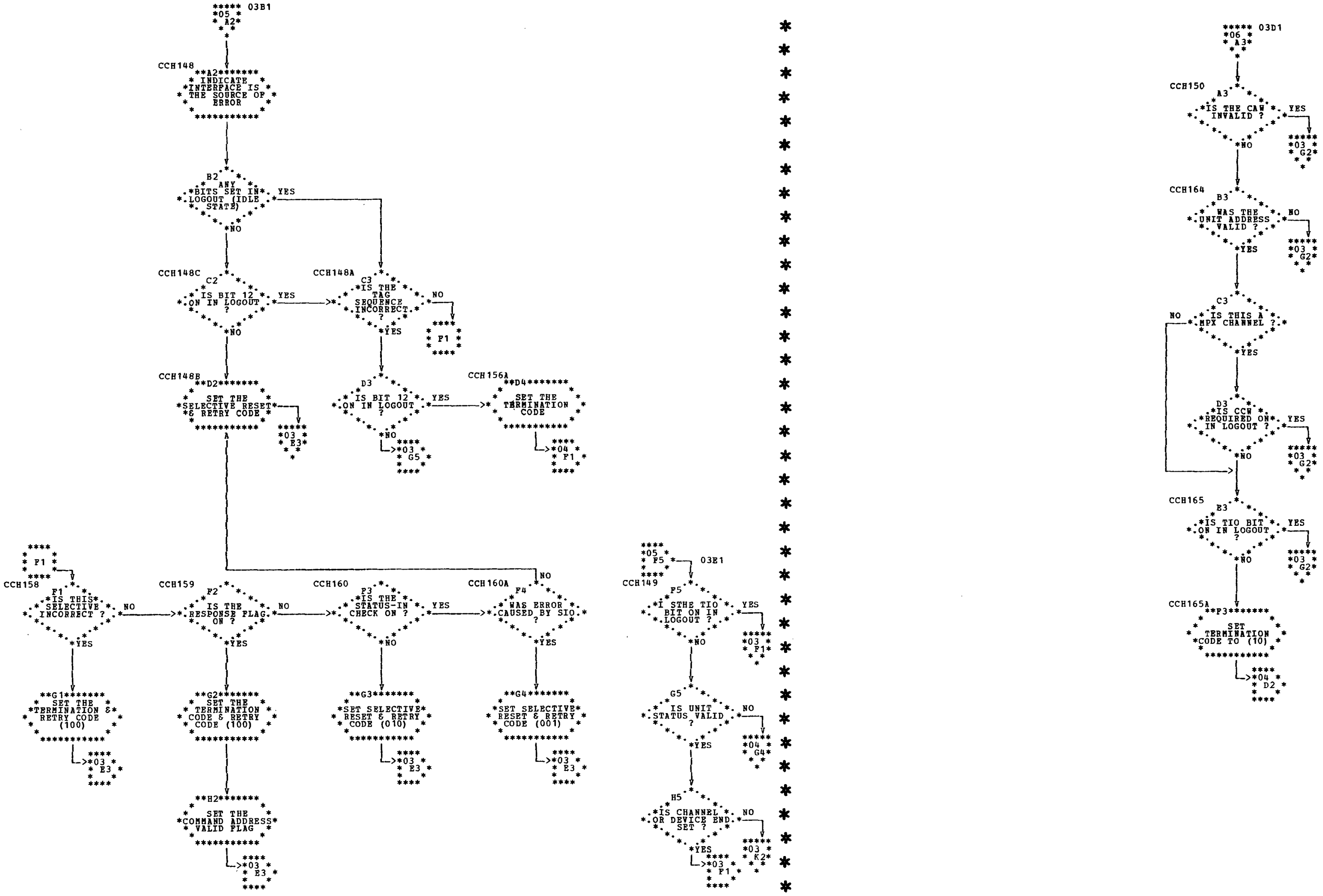

DMKSEV70
A1
DMKSEV70
B1
SETUP ADDRESSABILITY
C1
HAS THE CHANNEL LOGOUT ?
NO
YES
CLEANUP
C2
SET SYSTEM TERMINATION FLAG
01
C3
03H3
CCHEXIT4
C3
RESTORE REGS. & RETURN TO CCH CONTROL
D1
IS INDICATOR ON IN WORD 2 ?
NO
YES
E1
ANY PARITY BITS ON ?
NO
YES
CCH140
E2
IS STORAGE CHECK SET ?
NO
YES
F1
INDICATE CHANNEL ERROR IS SOURCE OF ERROR
F2
IS THIS A CAW FETCH ?
YES
NO
CCH140A
G2
INDICATE COMMAND ADDRESS VALID
CCH141
H2
IS CHANNEL CONTROL CHECK. INDICATED. ?
YES
NO
CCH141B
H3
IS LOCAL STORE ADDRESS. ON ?
NO
YES
CCH141C
H4
IS UNIT ADDRESS CHECK. SET ?
YES
NO
04
A2
J2
J2
CCH141H
J2
IS ADDR-IN CHECK ON ?
NO
YES
02
A3
J3
IS BIT 37 SET IN LOGOUT. ?
NO
YES
04
A2
CCH141A
K2
INDICATE INTERFACE IS SOURCE OF ERROR
K3
INDICATE CPU IS SOURCE OF ERROR
02
D3
02
D3
02
A3
01J2
CCH141D
A3
IS THIS AN ERROR ON A TIO ?
NO
YES
C3
B3
IS THE UNIT ADDR = TO ADDR IN LOGOUT
NO
YES
D3
C3
VALIDCCH
C3
INDICATE UNIT ADDRESS IS VALID
02
D3
01K2
01K3
04A2
CCH142
D2
IS BYTE COUNT PARITY BAD ?
YES
NO
J3
CCH141F
D3
IS STORAGE CHECK INDICATED.
NO
YES
A
D3
E2
BYTE COUNT BAD (SSC ONLY) ?
YES
NO
J3
E3
IS BIT 38 ON IN LOGOUT ?
NO
YES
CCH141G
E4
INDICATE STORAGE IS SOURCE OF ERROR
F2
IS UCW COUNT CHECK ?
NO
YES
F3
INDICATE CHANNEL IS SOURCE OF ERROR
G2
ANY UCW PARITY BITS ?
YES
NO
J3
G3
IS THE INDICATOR FOR CCW REQUIRED .ON ?
NO
YES
CCH143
H2
SET THE VALID FLAG FOR COUNT
CCH143A
H3
IS BIT 45 OR 46 ON IN LOGOUT
NO
YES
J3
CCH142A
J3
INDICATE CHANNEL IS SOURCE OF ERROR
CCH144
K3
IS STATUS-IN CHECK ON ?
NO
YES
04
A3
03
A3

| DMKSEV -- 2870 Channel Module (Parts 1 and 2 of 6)

| DMKSEV -- 2870 Channel Module (Parts 3 and 4 of 6)

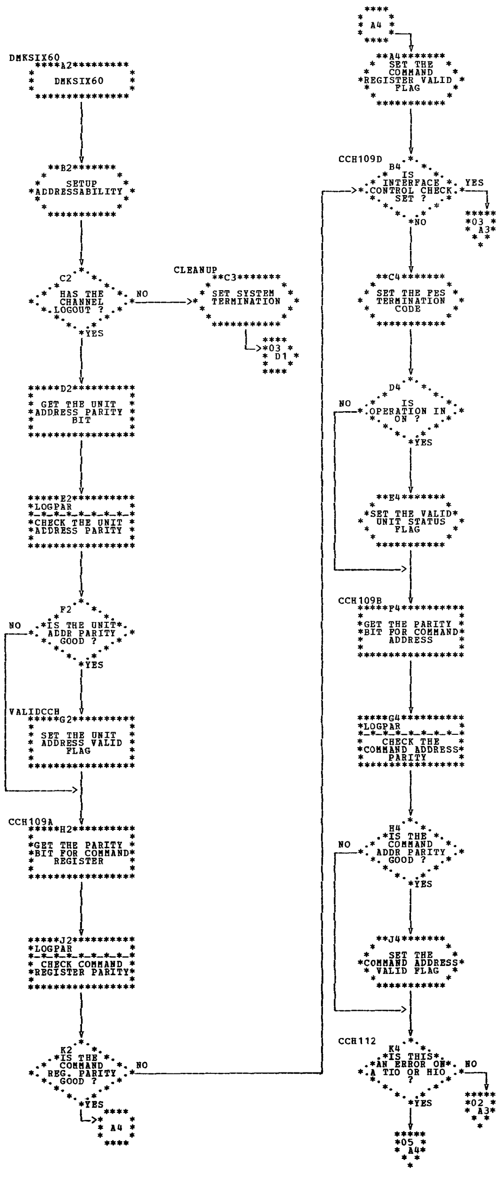

03 A3 02K3
CCH144A A3 INDICATE INTERFACE IS SOURCE OF ERROR
03 B3 04A3 04B3
CCH145A B3 IS THE SYSTEM RESET FLAG ON ? NO YES
CCH146 B2 IS THIS CHANNEL DATA CHECK ? NO YES
CCH147 B1 IS THIS AN INTERFACE CONTROL CHECK. YES NO
05 A2
C1 SET COMPLETICN CODE FOR FES
C2 IS SECOND CHECK SET ? NO YES
G5
C3 IS EC INSTALLED ? YES NO
CCH145B C4 INDICATE THE DEVICE ADDRESS VALID
D1 IS THIS A SIO INSTRUCTION ? YES NO
06 A3
D2 INDICATE CHANNEL IS SOURCE OF ERROR
D3 SET SYSTEM RESET & RETRY CODE
D4 SET THE COMMAND & COUNT INVALID
03 E3 04F1 04G4 04H2
FNOTE
CCHRETRN E3 IS DISCONNECT ON IN LOGOUT ? NO YES
E1 IS THIS ON A TIO INSTRUCTION ? YES NO
05 F5
E2 SET SYSTEM RESET & RETRY CODE
A
E4 WAS THIS FROM AN I/O INTERRUPT. ? NO YES
03 F1 05F5 05H5
CCH162 F1 IS THIS A PSUEDO SIO ? NO YES
F3 SET THE DISCONNECT FLAG IN THE BUILDED ECSW
F4 IS THIS A UNIT ADDRESS OR ADDR-IN CHECK. NO YES
03 G2 04D2 06A3 06B3 06D3 06E3
CCH152 G2 SET RETRY CODE TO (100)
G1 IS THE TIO FLAG ON IN LOGOUT ? YES NO
CCHRTRN G3 SAVE THE I/O EXTENDED LOGOUT AREA IN CCH RECORD
G4 SET THE DEVICE ADDRESS INVALID
03 G5 05D3
CCH146A G5 SET SELECTIVE RESET & RETRY CODE
G5
H3 RE-INITIALIZE THE I/O EXTENDED LOGOUT AREA TO ONES
01 C3
CCH151 H1 IS MODE TRIGER 6 ON ? YES NO
04 D1
J1 IS MODE TRIGER 9, 10,OR 11 ON ? YES NO
04 D1
03 K2 04G2 04H4 05H5
CCH157 K2 SET TERMINATION CODE TO (003)
K1 IS MODE TRIGER 2, 3, OR 4 ON ? YES NO
04 E1
TO:E3 05D2 05G1 05G3 05G4 05H2

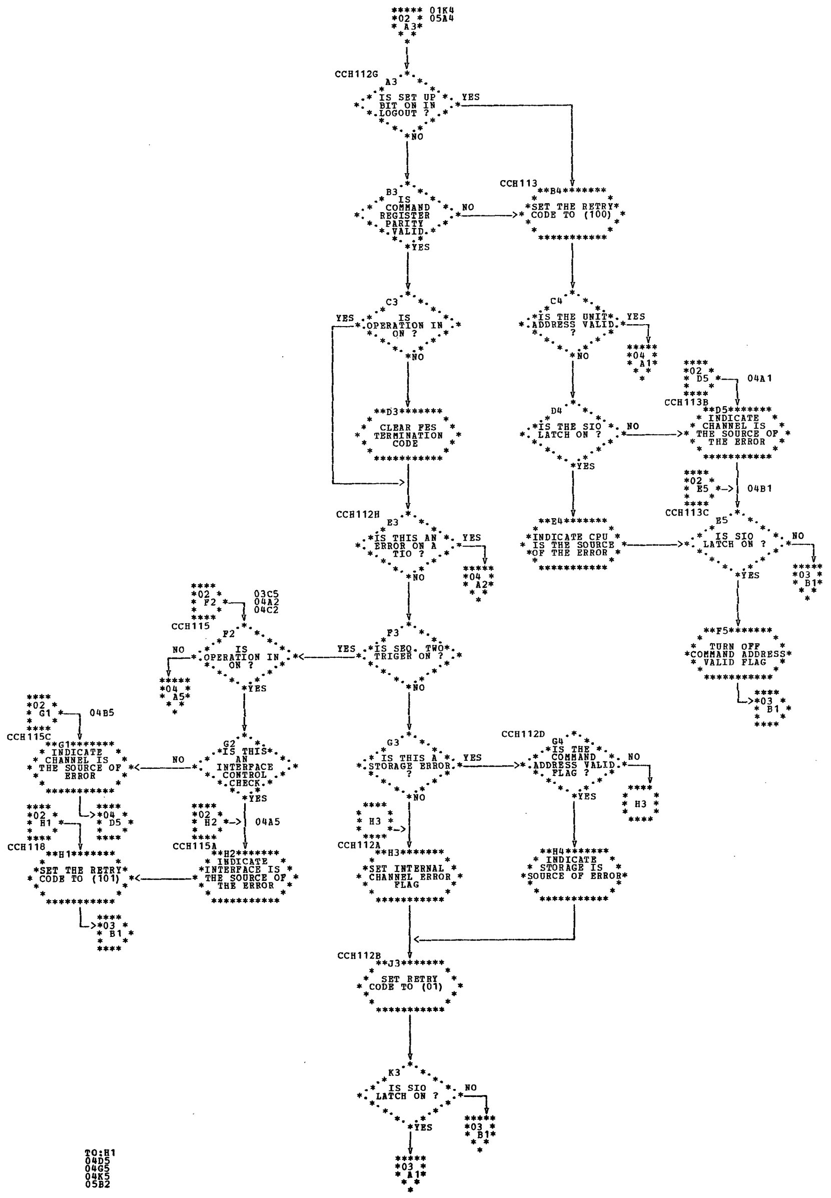

04 A2 01H4 01J3
CCH141E A2 INDICATE CHANNEL IS SOURCE OF ERROR
02 D3
04 A3 02K3
CCH145 A3 IS THIS AN INTERFACE CONTROL CHECK. NO YES
03 B3
B3 SET VALID FLAG FOR UNIT STATUS
03 B3
04 D1 03H1 03J1
CCH153 D1 IS COMMAND CHECK ON IN LOGOUT ? YES NO
04 D2 06F3
CCH165B D2 INDICATE CHANNEL IS THE SOURCE OF ERROR
03 G2
04 E1 03K1
CCH154 E1 IS THE VALIDITY FLAG ON FOR COUNT ? YES NO
CCH155 E2 IS THE COMMAND ADDRESS VALID. ? NO YES
F1
04 F1 05D4
CCH156 F1 SET THE RETRY CODE TO (101)
F1
A
03 E3
F2 GET COUNT IN CCW LOC AT CMD ADDR. -8
G2 IS CCW COUNT GREATER THAN CSW COUNT. ? NEQ HI LOEQ
03 K2
H2
CCH155B G3 IS THE UNIT ADDRESS VALID. ? NO YES
04 G4 05G5
CCH163 G4 SET THE RETRY CODE TO (111)
03 E3
CCH155A H2 SET THE RETRY CODE TO (010)
03 E3
CCH155C H3 IS THIS A MPX CHANNEL ? NO YES
CCH156D H4 DOES CCW BITS EQ TO LOGOUT BIT. ? NO YES
03 K2
H2
J3 INDICATE THE COUNT IS INVALID

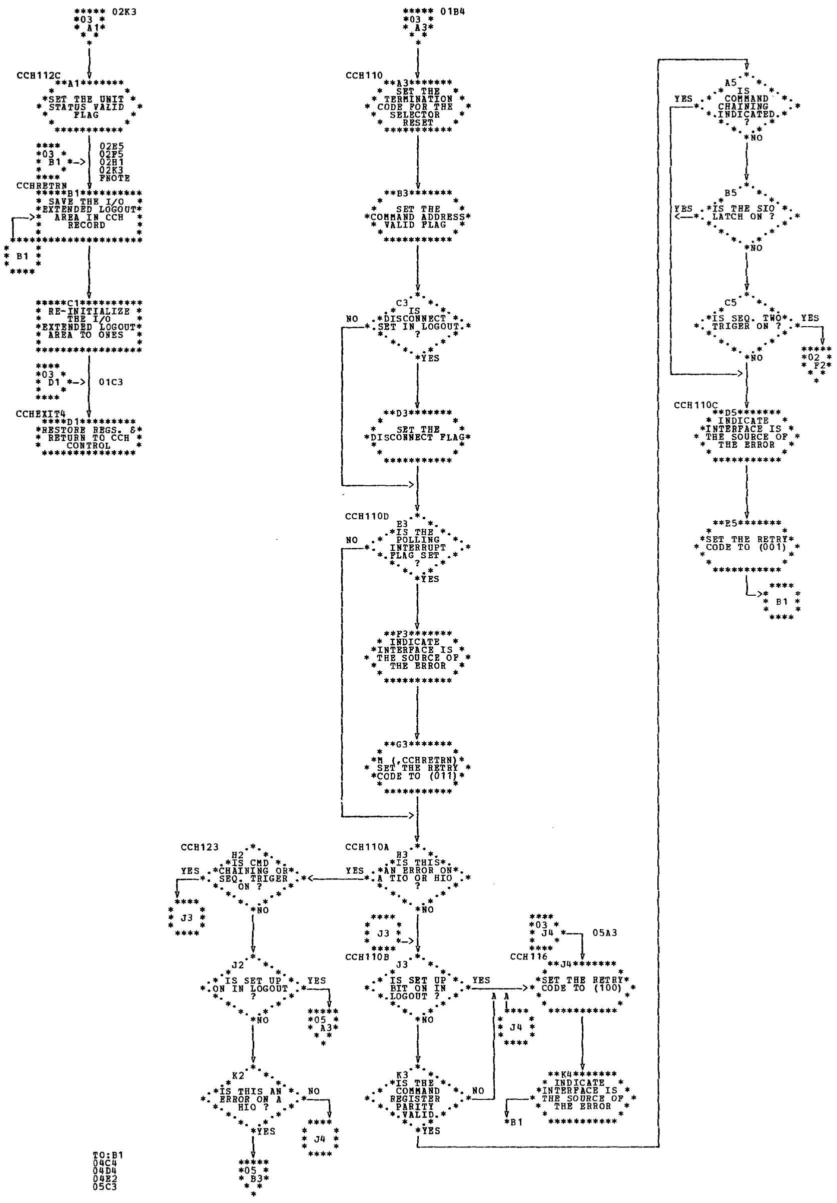

03B1
05 A2
CCH148
A2 INDICATE INTERFACE IS THE SOURCE OF ERROR
B2 ANY BITS SET IN LOGOUT (IDLE STATE)
YES
NO
CCH148C
C2 IS BIT 12 ON IN LOGOUT ?
YES
NO
CCH148A
C3 IS THE TAG SEQUENCE INCORRECT. ?
NO
F1
YES
CCH148B
D2 SET THE SELECTIVE RESET & RETRY CODE
03 E3
A
D3 IS BIT 12 ON IN LOGOUT ?
YES
NO
03 G5
CCH156A
D4 SET THE TERMINATION CODE
04 F1
F1
CCH158
F1 IS THIS SELECTIVE INCORRECT ?
NO
YES
CCH159
F2 IS THE RESPONSE FLAG ON ?
NO
YES
CCH160
F3 IS THE STATUS-IN CHECK ON ?
YES
NO
CCH160A
F4 WAS ERROR CAUSED BY SIO ?
NO
YES
G1 SET THE TERMINATION & RETRY CODE (100)
03 E3
G2 SET THE TERMINATION CODE & RETRY CODE (100)
G3 SET SELECTIVE RESET & RETRY CODE (010)
03 E3
G4 SET SELECTIVE RESET & RETRY CODE (001)
03 E3
H2 SET THE COMMAND ADDRESS VALID FLAG
03 E3
05 F5
03E1
CCH149
F5 I STHE TIO BIT ON IN LOGOUT ?
YES
03 F1
NO
G5 IS UNIT STATUS VALID ?
NO
04 G4
YES
H5 IS CHANNEL OR DEVICE END SET ?
NO
03 K2
YES
03 F1
03D1
06 A3
CCH150
A3 IS THE CAW INVALID ?
YES
03 G2
NO
CCH164
B3 WAS THE UNIT ADDRESS VALID ?
NO
03 G2
YES
C3 IS THIS A MPX CHANNEL ?
NO
YES
D3 IS CCW REQUIRED ON IN LOGOUT ?
YES
03 G2
NO
CCH165
E3 IS TIO BIT ON IN LOGOUT ?
YES
03 G2
NO
CCH165A
F3 SET TERMINATION CODE TO (10)
04 D2

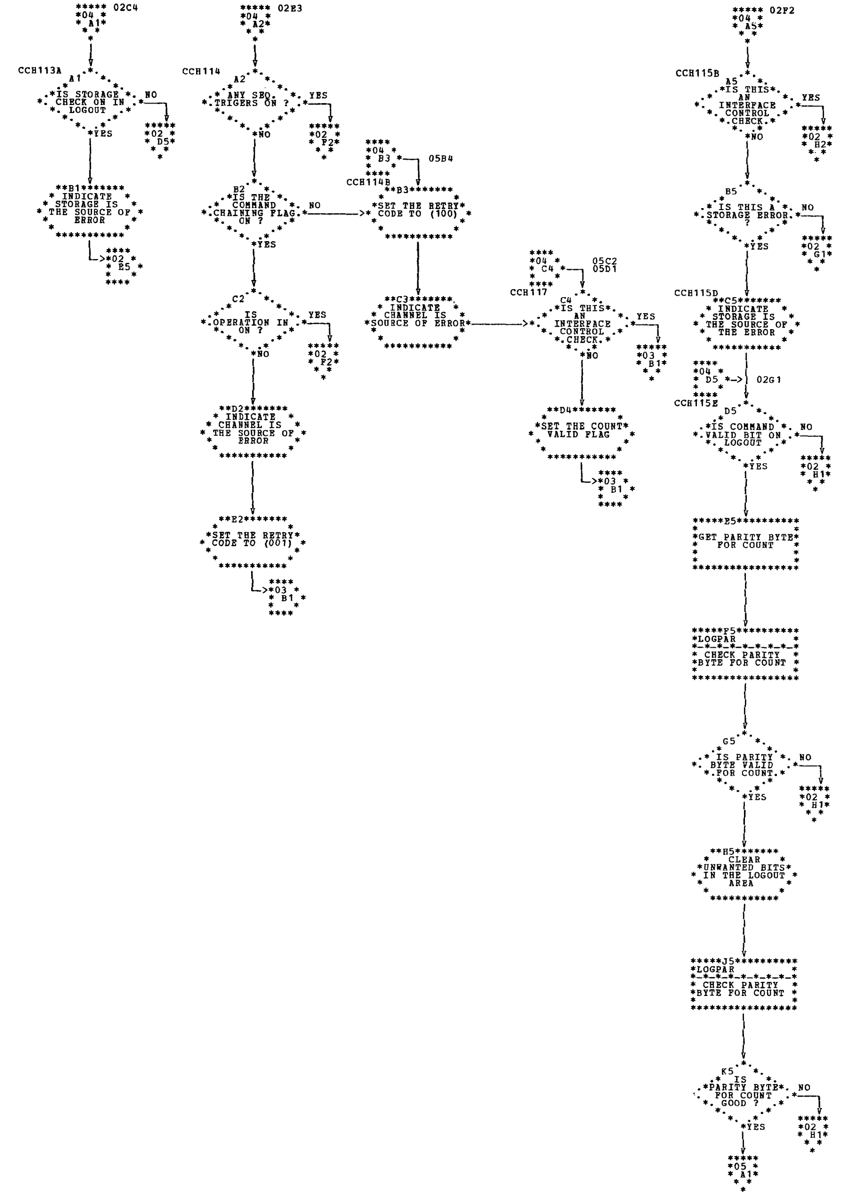
| DMKSEV -- 2870 Channel Module (Parts 5 and 6 of 6)

| DMKSIX -- 2860 Channel Module (Parts 1 and 2 of 5)


A4
A4 SET THE COMMAND REGISTER VALID FLAG
DMKSIX60
A2 DMKSIX60
B2 SETUP ADDRESSABILITY
C2 HAS THE CHANNEL LOGOUT ?
NO
YES
CLEANUP
C3 SET SYSTEM TERMINATION
03 D1
D2 GET THE UNIT ADDRESS PARITY BIT
E2 LOGPAR CHECK THE UNIT ADDRESS PARITY
F2 IS THE UNIT ADDR PARITY GOOD ?
NO
YES
VALIDCCH
G2 SET THE UNIT ADDRESS VALID FLAG
CCH109A
H2 GET THE PARITY BIT FOR COMMAND REGISTER
J2 LOGPAR CHECK COMMAND REGISTER PARITY
K2 IS THE COMMAND REG. PARITY GOOD ?
NO
YES
A4
CCH109D
B4 IS INTERFACE CONTROL CHECK SET ?
YES
03 A3
NO
C4 SET THE FES TERMINATION CODE
D4 IS OPERATION IN ON ?
NO
YES
E4 SET THE VALID UNIT STATUS FLAG
CCH109B
F4 GET THE PARITY BIT FOR COMMAND ADDRESS
G4 LOGPAR CHECK THE COMMAND ADDRESS PARITY
H4 IS THE COMMAND ADDR PARITY GOOD ?
NO
YES
J4 SET THE COMMAND ADDRESS VALID FLAG
CCH112
K4 IS THIS AN ERROR ON A TIO OR HIO ?
NO
02 A3
YES
05 A4

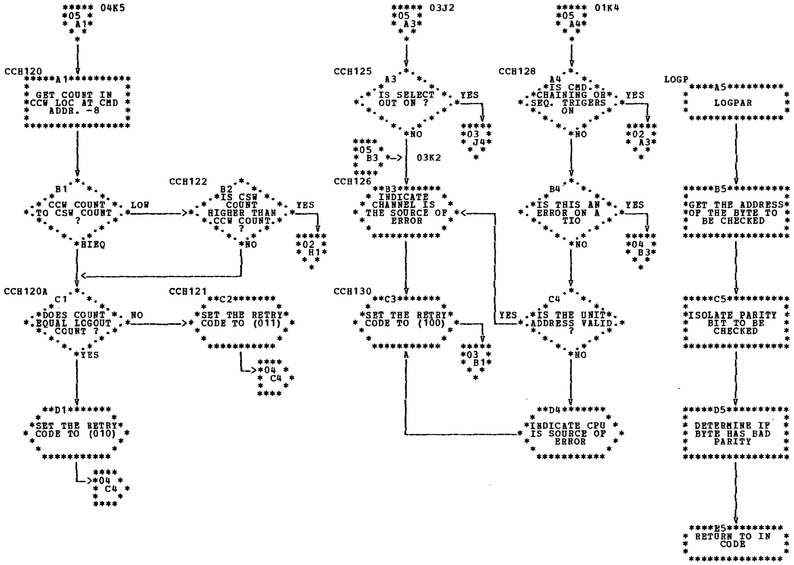

02 A3 01K4 05A4
CCH112G
A3 IS SET UP BIT ON IN LOGOUT ?
YES
NO
B3 IS COMMAND REGISTER PARITY VALID ?
NO
YES
CCH113
B4 SET THE RETRY CODE TO (100)
C3 IS OPERATION IN ON ?
YES
NO
D3 CLEAR FES TERMINATION CODE
C4 IS THE UNIT ADDRESS VALID ?
YES
04 A1
NO
D4 IS THE SIO LATCH ON ?
NO
YES
02 D5 04A1
CCH113B
D5 INDICATE CHANNEL IS THE SOURCE OF THE ERROR
02 E5 04B1
CCH113C
E4 INDICATE CPU IS THE SOURCE OF THE ERROR
E5 IS SIO LATCH ON ?
NO
03 B1
YES
F5 TURN OFF COMMAND ADDRESS VALID FLAG
03 B1
CCH112H
E3 IS THIS AN ERROR ON A TIO ?
YES
04 A2
NO
F3 IS SEQ. TWO TRIGER ON ?
YES
NO
02 F2 03C5 04A2 04C2
CCH115
F2 IS OPERATION IN ON ?
NO
04 A5
YES
G2 IS THIS AN INTERFACE CONTROL CHECK
NO
YES
02 G1 04B5
CCH115C
G1 INDICATE CHANNEL IS THE SOURCE OF ERROR
04 D5
02 H1
CCH118
H1 SET THE RETRY CODE TO (101)
03 B1
02 H2 04A5
CCH115A
H2 INDICATE INTERFACE IS THE SOURCE OF THE ERROR
G3 IS THIS A STORAGE ERROR ?
YES
NO
CCH112D
G4 IS THE COMMAND ADDRESS VALID FLAG ?
NO
H3
YES
H3
CCH112A
H3 SET INTERNAL CHANNEL ERROR FLAG
H4 INDICATE STORAGE IS SOURCE OF ERROR
CCH112B
J3 SET RETRY CODE TO (01)
K3 IS SIO LATCH ON ?
NO
03 B1
YES
03 A1
TO:H1 04D5 04G5 04K5 05B2

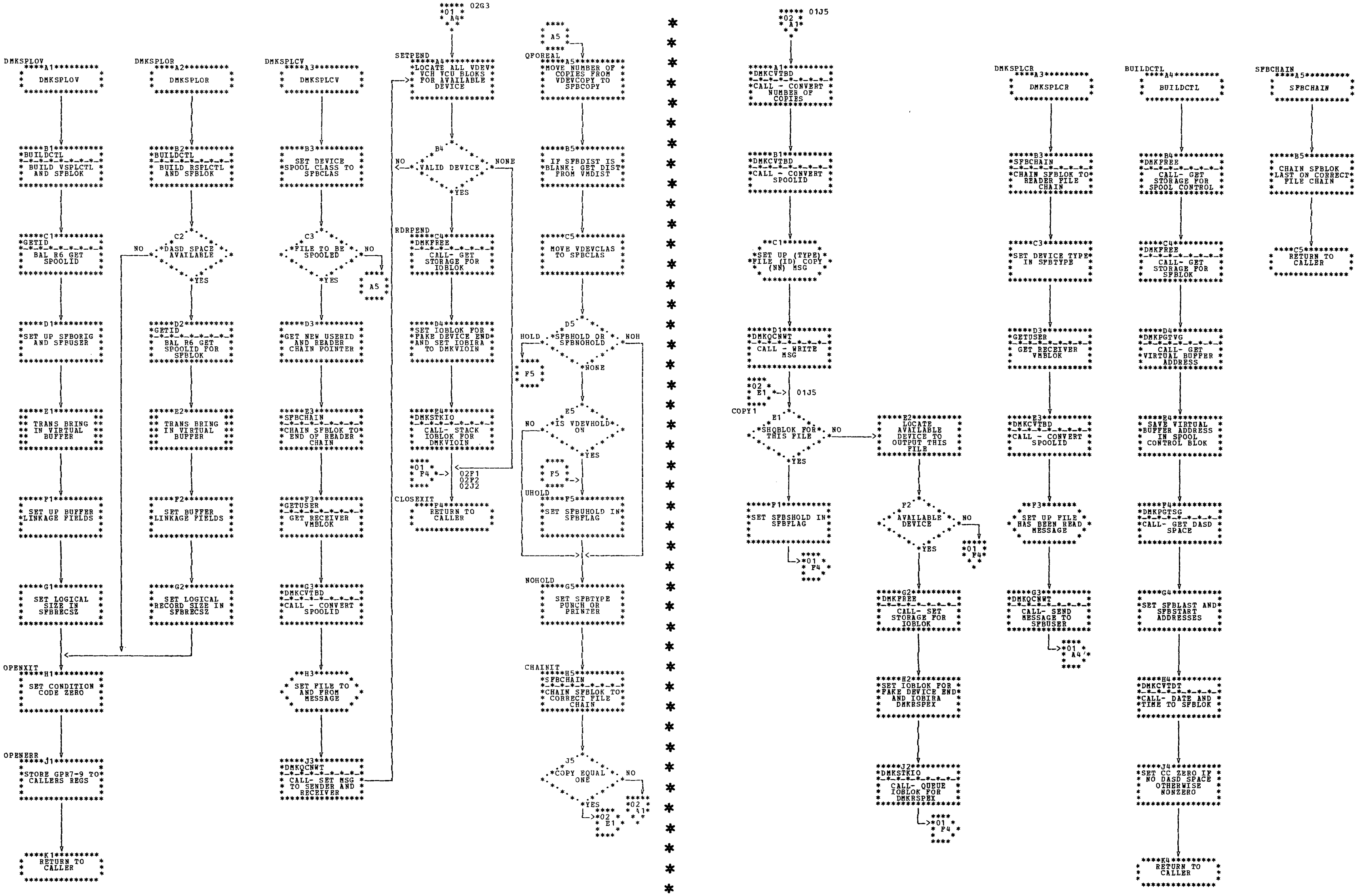

02K3
03 A1
CCH112C A1
SET THE UNIT STATUS VALID FLAG
03 B1 02E5 02F5 02B1 02K3 FNOTE
CCHRETRN B1
SAVE THE I/O EXTENDED LOGOUT AREA IN CCH RECORD
B1
C1
RE-INITIALIZE THE I/O EXTENDED LOGOUT AREA TO ONES
03 D1 01C3
CCHEXIT4 D1
RESTORE REGS. & RETURN TO CCH CONTROL
01B4
03 A3
CCH110 A3
SET THE TERMINATION CODE FOR THE SELECTOR RESET
B3
SET THE COMMAND ADDRESS VALID FLAG
C3
IS DISCONNECT SET IN LOGOUT ?
NO
YES
D3
SET THE DISCONNECT FLAG
CCH110D E3
IS THE POLLING INTERRUPT FLAG SET ?
NO
YES
F3
INDICATE INTERFACE IS THE SOURCE OF THE ERROR
G3
M (CCHRETRN) SET THE RETRY CODE TO (011)
CCH123 H2
IS CMD CHAINING OR SEQ. TRIGER ON ?
YES
J3
NO
CCH110A H3
IS THIS AN ERROR ON A TIO OR HIO ?
YES
NO
J2
IS SET UP ON IN LOGOUT ?
YES
05 A3
NO
K2
IS THIS AN ERROR ON A HIO ?
NO
J4
YES
05 B3
TO:B1 04C4 04D4 04E2 05C3
J3
CCH110B J3
IS SET UP BIT ON IN LOGOUT ?
YES
NO
K3
IS THE COMMAND REGISTER PARITY VALID.
NO
YES
03 J4 05A3
CCH116 J4
SET THE RETRY CODE TO (100)
J4
K4
INDICATE INTERFACE IS THE SOURCE OF THE ERROR
B1
A5
IS COMMAND CHAINING INDICATED.
YES
NO
B5
IS THE SIO LATCH ON ?
YES
NO
C5
IS SEQ. TWO TRIGER ON ?
YES
02 F2
NO
CCH110C D5
INDICATE INTERFACE IS THE SOURCE OF LOGOUT
E5
SET THE RETRY CODE TO (001)
B1


02C4
04 A1
CCH113A A1
IS STORAGE CHECK ON IN LOGOUT
NO
02 D5
YES
B1
INDICATE STORAGE IS THE SOURCE OF ERROR
02 E5
02E3
04 A2
CCH114 A2
ANY SEQ. TRIGERS ON ?
YES
02 F2
NO
B2
IS THE COMMAND CHAINING FLAG ON ?
NO
YES
C2
IS OPERATION IN ON ?
YES
02 F2
NO
D2
INDICATE CHANNEL IS THE SOURCE OF ERROR
E2
SET THE RETRY CODE TO (001)
03 B1
04 B3 05B4
CCH114B B3
SET THE RETRY CODE TO (100)
C3
INDICATE CHANNEL IS SOURCE OF ERROR
04 C4 05C2 05D1
CCH117 C4
IS THIS AN INTERFACE CONTROL CHECK.
YES
03 B1
NO
D4
SET THE COUNT VALID FLAG
03 B1
02F2
04 A5
CCH115B A5
IS THIS AN INTERFACE CONTROL CHECK.
YES
02 H2
NO
B5
IS THIS A STORAGE ERROR. ?
NO
02 G1
YES
CCH115D C5
INDICATE STORAGE IS THE SOURCE OF THE ERROR
04 D5 02G1
CCH115E D5
IS COMMAND VALID BIT ON . LOGOUT
NO
02 H1
YES
E5
GET PARITY BYTE FOR COUNT
F5
LOGPAR
CHECK PARITY BYTE FOR COUNT
G5
IS PARITY BYTE VALID FOR COUNT.
NO
02 H1
YES
H5
CLEAR UNWANTED BITS IN THE LOGOUT AREA
J5
LOGPAR
CHECK PARITY BYTE FOR COUNT
K5
IS PARITY BYTE FOR COUNT GOOD ?
NO
02 H1
YES
05 A1

| DMKSIX -- 2860 Channel Module (Parts 3 and 4 of 5)

| DMKSIX -- 2860 Channel Module (Part 5 of 5)

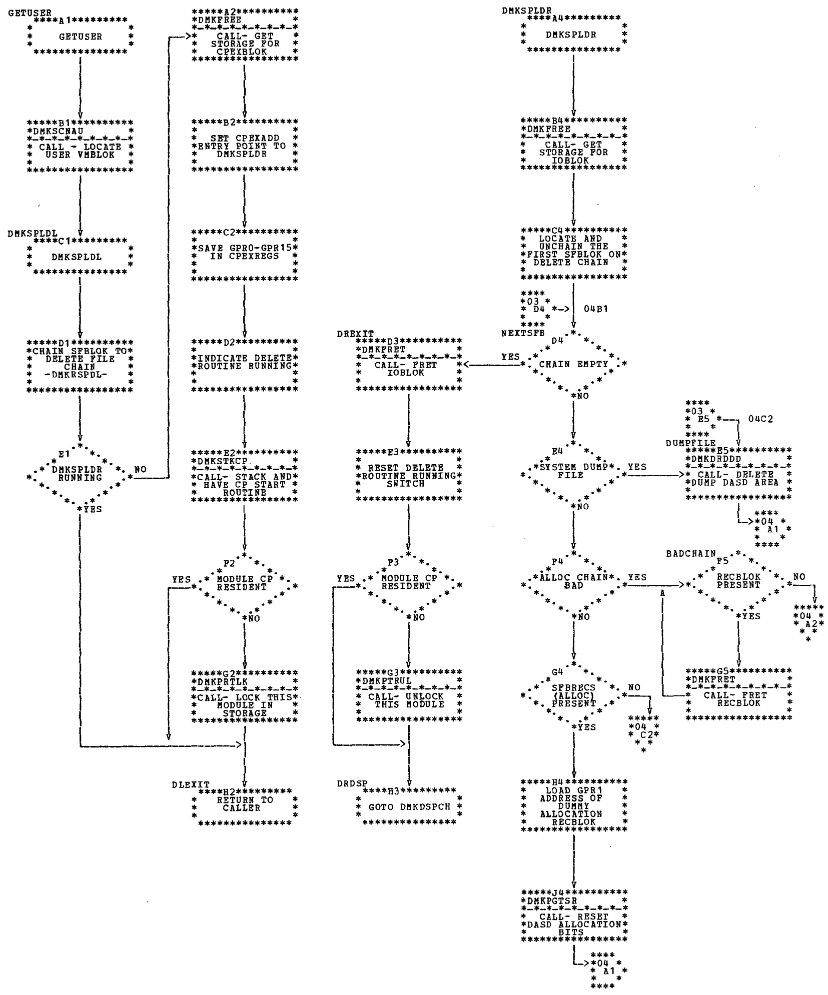

05 A1 04K5
CCH120
A1
GET COUNT IN CCW LOC AT CMD ADDR. -8
B1
CCW COUNT TO CSW COUNT ?
LOW
HIEQ
CCH122
B2
IS CSW COUNT HIGHER THAN CCW COUNT. ?
YES
NO
02 H1
CCH120A
C1
DOES COUNT EQUAL LCGOUT COUNT ?
NO
YES
CCH121
C2
SET THE RETRY CODE TO (011)
04 C4
D1
SET THE RETRY CODE TO (010)
04 C4
05 A3 03J2
CCH125
A3
IS SELECT OUT ON ?
YES
03 J4
NO
05 B3
03K2
CCH126
B3
INDICATE CHANNEL IS THE SOURCE OF ERROR
CCH130
C3
SET THE RETRY CODE TO (100)
03 B1
A
05 A4 01K4
CCH128
A4
IS CMD. CHAINING OR SEQ. TRIGERS ON
YES
02 A3
NO
B4
IS THIS AN ERROR ON A TIO
YES
04 B3
NO
C4
IS THE UNIT ADDRESS VALID. ?
YES
NO
D4
INDICATE CPU IS SOURCE OF ERROR
LOGP
A5
LOGPAR
B5
GET THE ADDRESS OF THE BYTE TO BE CHECKED
C5
ISOLATE PARITY BIT TO BE CHECKED
D5
DETERMINE IF BYTE HAS BAD PARITY
E5
RETURN TO IN CODE

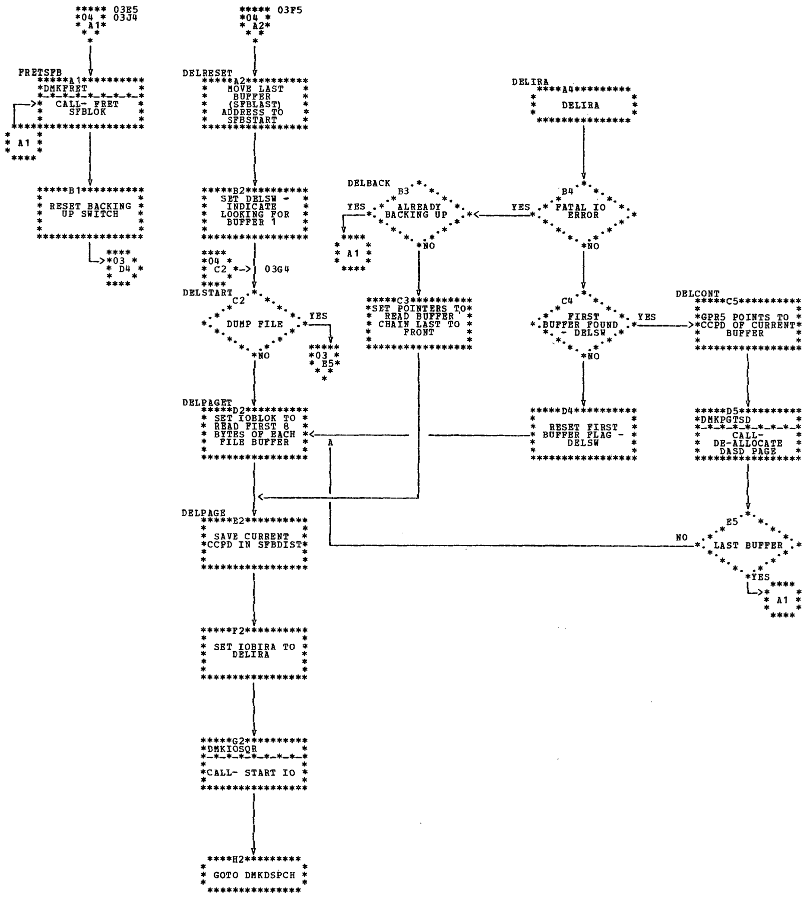


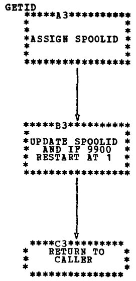
DMKSPLOV
A1 DMKSPLOV
B1 BUILDCTL BUILD VSPLCTL AND SFBLOK
C1 GETID BAL R6 GET SPOOLID
D1 SET UP SFBORIG AND SFBUSER
E1 TRANS BRING IN VIRTUAL BUFFER
F1 SET UP BUFFER LINKAGE FIELDS
G1 SET LOGICAL SIZE IN SFBRECSZ
OPENXIT
H1 SET CONDITION CODE ZERO
OPENERR
J1 STORE GPR7-9 TO CALLERS REGS
K1 RETURN TO CALLER
DMKSPLOR
A2 DMKSPLOR
B2 BUILDCTL BUILD RSPLCTL AND SFBLOK
C2 DASD SPACE AVAILABLE
NO
YES
D2 GETID BAL R6 GET SPOOLID FOR SFBLOK
E2 TRANS BRING IN VIRTUAL BUFFER
F2 SET BUFFER LINKAGE FIELDS
G2 SET LOGICAL RECORD SIZE IN SFBRECSZ
DMKSPLCV
A3 DMKSPLCV
B3 SET DEVICE SPOOL CLASS TO SFBCLAS
C3 FILE TO BE SPOOLED
NO
A5
YES
D3 GET NEW USERID AND READER CHAIN POINTER
E3 SFBCHAIN CHAIN SFBLOK TO END OF READER CHAIN
F3 GETUSER GET RECEIVER VMBLOK
G3 DMKCVTBD CALL - CONVERT SPOOLID
H3 SET FILE TO AND FROM MESSAGE
J3 DMKQCNWT CALL- SET MSG TO SENDER AND RECEIVER
01 A4 02G3
SETPEND
A4 LOCATE ALL VDEV VCH VCU BLOKS FOR AVAILABLE DEVICE
B4 VALID DEVICE
NO
NONE
YES
RDRPEND
C4 DMKFREE CALL- GET STORAGE FOR IOBLOK
D4 SET IOBLOK FOR FAKE DEVICE END AND SET IOBIRA TO DMKVIOIN
E4 DMKSTKIO CALL- STACK IOBLOK FOR DMKVIOIN
01 F4
02F1
02F2
02J2
CLOSEXIT
F4 RETURN TO CALLER
A5
QFOREAL
A5 MOVE NUMBER OF COPIES FROM VDEVCOPY TO SFBCOPY
B5 IF SFBDIST IS BLANK: GET DIST FROM VMDIST
C5 MOVE VDEVCLAS TO SFBCLAS
D5 SFBHOLD OR SFBNOHOLD
HOLD
NOH
NONE
F5
E5 IS VDEVHOLD ON
NO
YES
F5
UHOLD
F5 SET SFBUHOLD IN SFBFLAG
NOHOLD
G5 SET SFBTYPE PUNCH OR PRINTER
CHAINIT
H5 SFBCHAIN CHAIN SFBLOK TO CORRECT FILE CHAIN
J5 COPY EQUAL ONE
NO
YES
02 E1
02 A1
02 A1 01J5
A1 DMKCVTBD CALL - CONVERT NUMBER OF COPIES
B1 DMKCVTBD CALL - CONVERT SPOOLID
C1 SET UP (TYPE) FILE (ID) COPY (NN) MSG
D1 DMKQCNWT CALL - WRITE MSG
02 E1 01J5
COPY1
E1 SHOBLOK FOR THIS FILE
NO
YES
F1 SET SFBSHOLD IN SFBFLAG
01 F4
E2 LOCATE AVAILABLE DEVICE TO OUTPUT THIS FILE
F2 AVAILABLE DEVICE
NO
YES
01 F4
G2 DMKFREE CALL- SET STORAGE FOR IOBLOK
H2 SET IOBLOK FOR FAKE DEVICE END AND IOBIRA DMKRSPEX
J2 DMKSTKIO CALL- QUEUE IOBLOK FOR DMKRSPEX
01 F4
DMKSPLCR
A3 DMKSPLCR
B3 SFBCHAIN CHAIN SFBLOK TO READER FILE CHAIN
C3 SET DEVICE TYPE IN SFBTYPE
D3 GETUSER GET RECEIVER VMBLOK
E3 DMKCVTBD CALL - CONVERT SPOOLID
F3 SET UP FILE HAS BEEN READ MESSAGE
G3 DMKQCNWT CALL- SEND MESSAGE TO SFBUSER
01 A4
BUILDCTL
A4 BUILDCTL
B4 DMKFREE CALL- GET STORAGE FOR SPOOL CONTROL
C4 DMKFREE CALL- GET STORAGE FOR SFBLOK
D4 DMKPGTVG CALL- GET VIRTUAL BUFFER ADDRESS
E4 SAVE VIRTUAL BUFFER ADDRESS IN SPOOL CONTROL BLOK
F4 DMKPGTSG CALL- GET DASD SPACE
G4 SET SFBLAST AND SFBSTART ADDRESSES
H4 DMKCVTDT CALL- DATE AND TIME TO SFBLOK
J4 SET CC ZERO IF NO DASD SPACE OTHERWISE NONZERO
K4 RETURN TO CALLER
SFBCHAIN
A5 SFBCHAIN
B5 CHAIN SFBLOK LAST ON CORRECT FILE CHAIN
C5 RETURN TO CALLER

DMKSPL -- Spooling Subroutines (Parts 1 and 2 of 5)

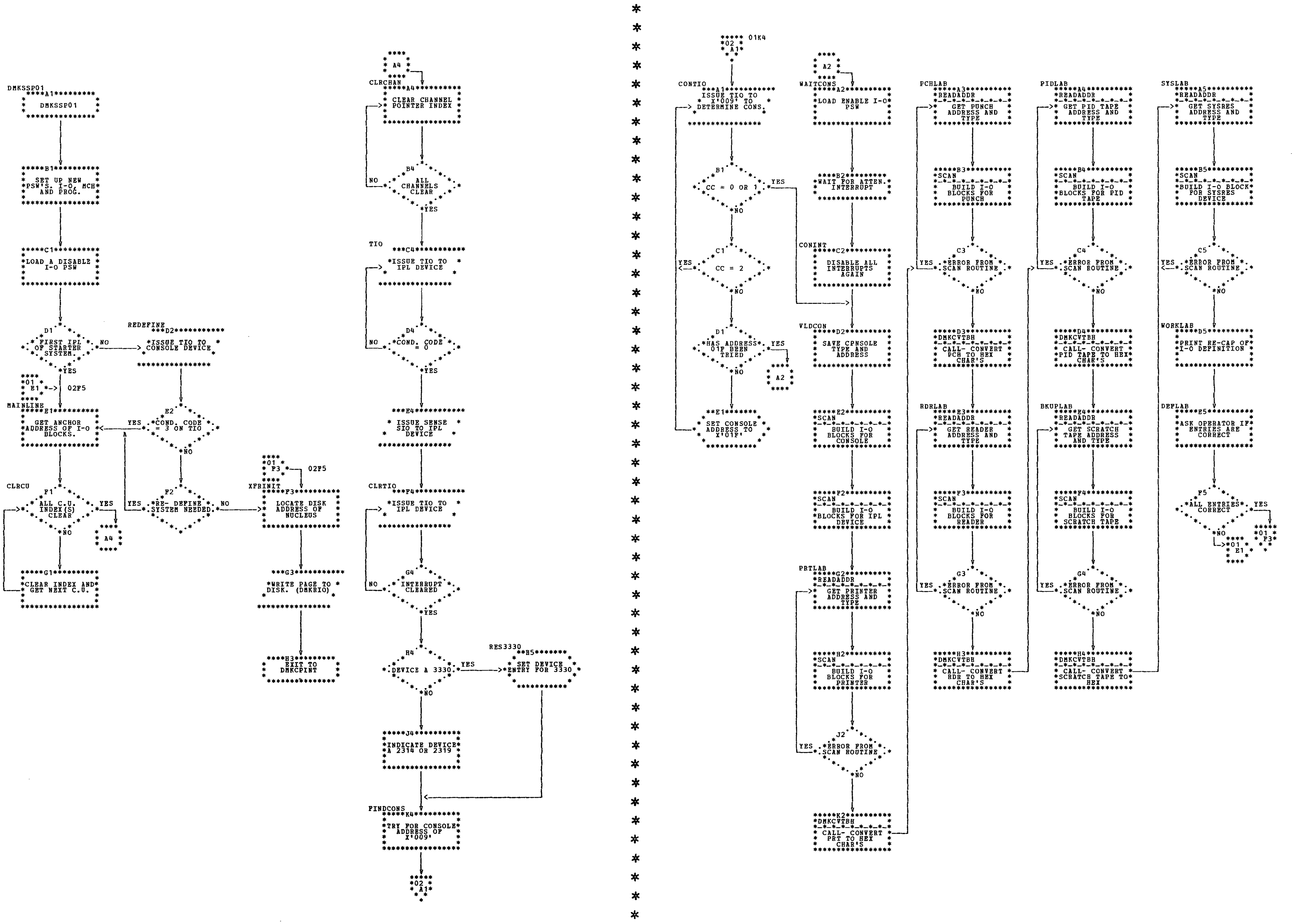
DMKSPL -- Spooling Subroutines (Parts 3 and 4 of 5)


GETUSER
A1 GETUSER
B1 DMKSCNAU CALL - LOCATE USER VMBLOK
DMKSPLDL
C1 DMKSPLDL
D1 CHAIN SFBLOK TO DELETE FILE CHAIN -DMKRSPDL-
E1 DMKSPLDR RUNNING
NO
YES
A2 DMKFREE CALL- GET STORAGE FOR CPEXBLOK
B2 SET CPEXADD ENTRY POINT TO DMKSPLDR
C2 SAVE GPR0-GPR15 IN CPEXREGS
D2 INDICATE DELETE ROUTINE RUNNING
E2 DMKSTKCP CALL- STACK AND HAVE CP START ROUTINE
F2 MODULE CP RESIDENT
YES
NO
G2 DMKPRTLK CALL- LOCK THIS MODULE IN STORAGE
DLEXIT
H2 RETURN TO CALLER
DREXIT
D3 DMKFRET CALL- FRET IOBLOK
E3 RESET DELETE ROUTINE RUNNING SWITCH
F3 MODULE CP RESIDENT
YES
NO
G3 DMKPTRUL CALL- UNLOCK THIS MODULE
DRDSP
H3 GOTO DMKDSPCH
DMKSPLDR
A4 DMKSPLDR
B4 DMKFREE CALL- GET STORAGE FOR IOBLOK
C4 LOCATE AND UNCHAIN THE FIRST SFBLOK ON DELETE CHAIN
03 D4 04B1
NEXTSFB
D4 CHAIN EMPTY
YES
NO
E4 SYSTEM DUMP FILE
YES
NO
F4 ALLOC CHAIN BAD
YES
NO
G4 SFBRECS (ALLOC) PRESENT
NO
YES
04 C2
H4 LOAD GPR1 ADDRESS OF DUMMY ALLOCATION RECBLOK
J4 DMKPGTSR CALL- RESET DASD ALLOCATION BITS
04 A1
03 E5 04C2
DUMPFILE
E5 DMKDRDDD CALL- DELETE DUMP DASD AREA
04 A1
BADCHAIN
F5 RECBLOK PRESENT
NO
YES
04 A2
A
G5 DMKFRET CALL- FRET RECBLOK


04 A1 03E5 03J4
FRETSFB
A1 DMKFRET CALL- FRET SFBLOK
A1
B1 RESET BACKING UP SWITCH
03 D4
04 A2 03F5
DELRESET
A2 MOVE LAST BUFFER (SFBLAST) ADDRESS TO SFBSTART
B2 SET DELSW - INDICATE LOOKING FOR BUFFER 1
04 C2 03G4
DELSTART
C2 DUMP FILE
YES
NO
03 E5
DELPAGET
D2 SET IOBLOK TO READ FIRST 8 BYTES OF EACH FILE BUFFER
A
DELPAGE
E2 SAVE CURRENT CCPD IN SFBDIST
F2 SET IOBIRA TO DELIRA
G2 DMKIOSQR CALL- START IO
H2 GOTO DMKDSPCH
DELBACK
B3 ALREADY BACKING UP
YES
NO
A1
C3 SET POINTERS TO READ BUFFER CHAIN LAST TO FRONT
DELIRA
A4 DELIRA
B4 FATAL IO ERROR
YES
NO
C4 FIRST BUFFER FOUND - DELSW
YES
NO
D4 RESET FIRST BUFFER FLAG - DELSW
DELCONT
C5 GPR5 POINTS TO CCPD OF CURRENT BUFFER
D5 DMKPGTSD CALL- DE-ALLOCATE DASD PAGE
E5 LAST BUFFER
NO
YES
A1

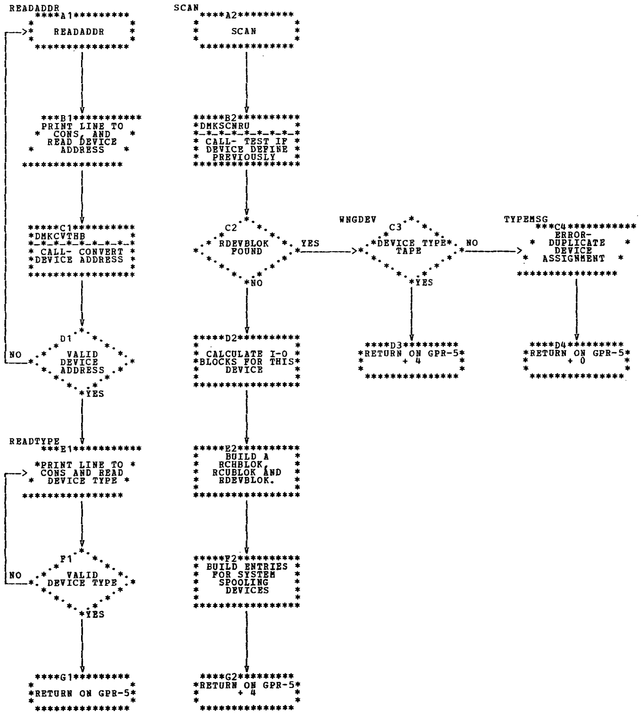


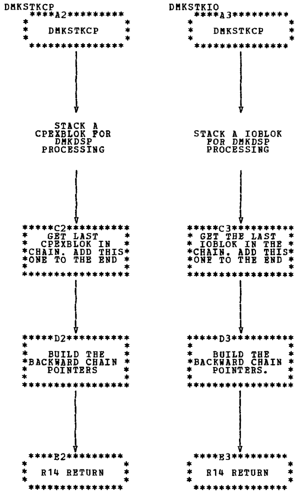
DMKSPL -- Spooling Subroutines (Part 5 of 5)


DMKSSP01
A1
DMKSSP01
B1
SET UP NEW PSW'S. I-O, MCH AND PROG.
C1
LOAD A DISABLE I-O PSW
D1
FIRST IPL OF STARTER SYSTEM.
NO
YES
REDEFINE
D2
ISSUE TIO TO CONSOLE DEVICE
01 E1
02F5
MAINLINE
E1
GET ANCHOR ADDRESS OF I-O BLOCKS.
E2
COND. CODE = 3 ON TIO
YES
NO
A
CLRCU
F1
ALL C.U. INDEX(S) CLEAR
YES
A4
NO
G1
CLEAR INDEX AND GET NEXT C.U.
F2
RE- DEFINE SYSTEM NEEDED.
YES
NO
01 F3
02F5
XFRINIT
F3
LOCATE DISK ADDRESS OF NUCLEUS
G3
WRITE PAGE TO DISK. (DMKRIO)
H3
EXIT TO DMKCPINT
A4
CLRCHAN
A4
CLEAR CHANNEL POINTER INDEX
B4
ALL CHANNELS CLEAR
NO
YES
TIO
C4
ISSUE TIO TO IPL DEVICE
D4
COND. CODE = 0
NO
YES
E4
ISSUE SENSE SIO TO IPL DEVICE
CLRTIO
F4
ISSUE TIO TO IPL DEVICE
G4
INTERRUPT CLEARED
NO
YES
H4
DEVICE A 3330
YES
NO
RES3330
H5
SET DEVICE ENTRY FOR 3330
J4
INDICATE DEVICE A 2314 OR 2319
FINDCONS
K4
TRY FOR CONSOLE ADDRESS OF X'009'
02 A1
02 A1
01K4
CONTIO
A1
ISSUE TIO TO X'009' TO DETERMINE CONS.
B1
CC = 0 OR 1
YES
NO
C1
CC = 2
YES
NO
D1
HAS ADDRESS 01F BEEN TRIED
YES
NO
A2
E1
SET CONSOLE ADDRESS TO X'01F'
A2
WAITCONS
A2
LOAD ENABLE I-O PSW
B2
WAIT FOR ATTEN. INTERRUPT
CONINT
C2
DISABLE ALL INTERRUPTS AGAIN
VLDCON
D2
SAVE CPNSOLE TYPE AND ADDRESS
E2
SCAN
BUILD I-O BLOCKS FOR CONSOLE
F2
SCAN
BUILD I-O BLOCKS FOR IPL DEVICE
PRTLAB
G2
READADDR
GET PRINTER ADDRESS AND TYPE
H2
SCAN
BUILD I-O BLOCKS FOR PRINTER
J2
ERROR FROM SCAN ROUTINE
YES
NO
K2
DMKCVTBH
CALL- CONVERT PRT TO HEX CHAR'S
PCHLAB
A3
READADDR
GET PUNCH ADDRESS AND TYPE
B3
SCAN
BUILD I-O BLOCKS FOR PUNCH
C3
ERROR FROM SCAN ROUTINE
YES
NO
D3
DMKCVTBH
CALL- CONVERT PCH TO HEX CHAR'S
RDRLAB
E3
READADDR
GET READER ADDRESS AND TYPE
F3
SCAN
BUILD I-O BLOCKS FOR READER
G3
ERROR FROM SCAN ROUTINE
YES
NO
H3
DMKCVTBH
CALL- CONVERT RDR TO HEX CHAR'S
PIDLAB
A4
READADDR
GET PID TAPE ADDRESS AND TYPE
B4
SCAN
BUILD I-O BLOCKS FOR PID TAPE
C4
ERROR FROM SCAN ROUTINE
YES
NO
D4
DMKCVTBH
CALL- CONVERT PID TAPE TO HEX CHAR'S
BKUPLAB
E4
READADDR
GET SCRATCH TAPE ADDRESS AND TYPE
F4
SCAN
BUILD I-O BLOCKS FOR SCRATCH TAPE
G4
ERROR FROM SCAN ROUTINE
YES
NO
H4
DMKCVTBH
CALL- CONVERT SCRATCH TAPE TO HEX
SYSLAB
A5
READADDR
GET SYSRES ADDRESS AND TYPE
B5
SCAN
BUILD I-O BLOCK FOR SYSRES DEVICE
C5
ERROR FROM SCAN ROUTINE
YES
NO
WORKLAB
D5
PRINT RE-CAP OF I-O DEFINITION
DEFLAB
E5
ASK OPERATOR IF ENTRIES ARE CORRECT
F5
ALL ENTRIES CORRECT
YES
NO
01 E1
01 F3

DMKSSP -- Build Real I/O Blocks for Starter System or at IPL Time (Parts 1 and 2 of 3)

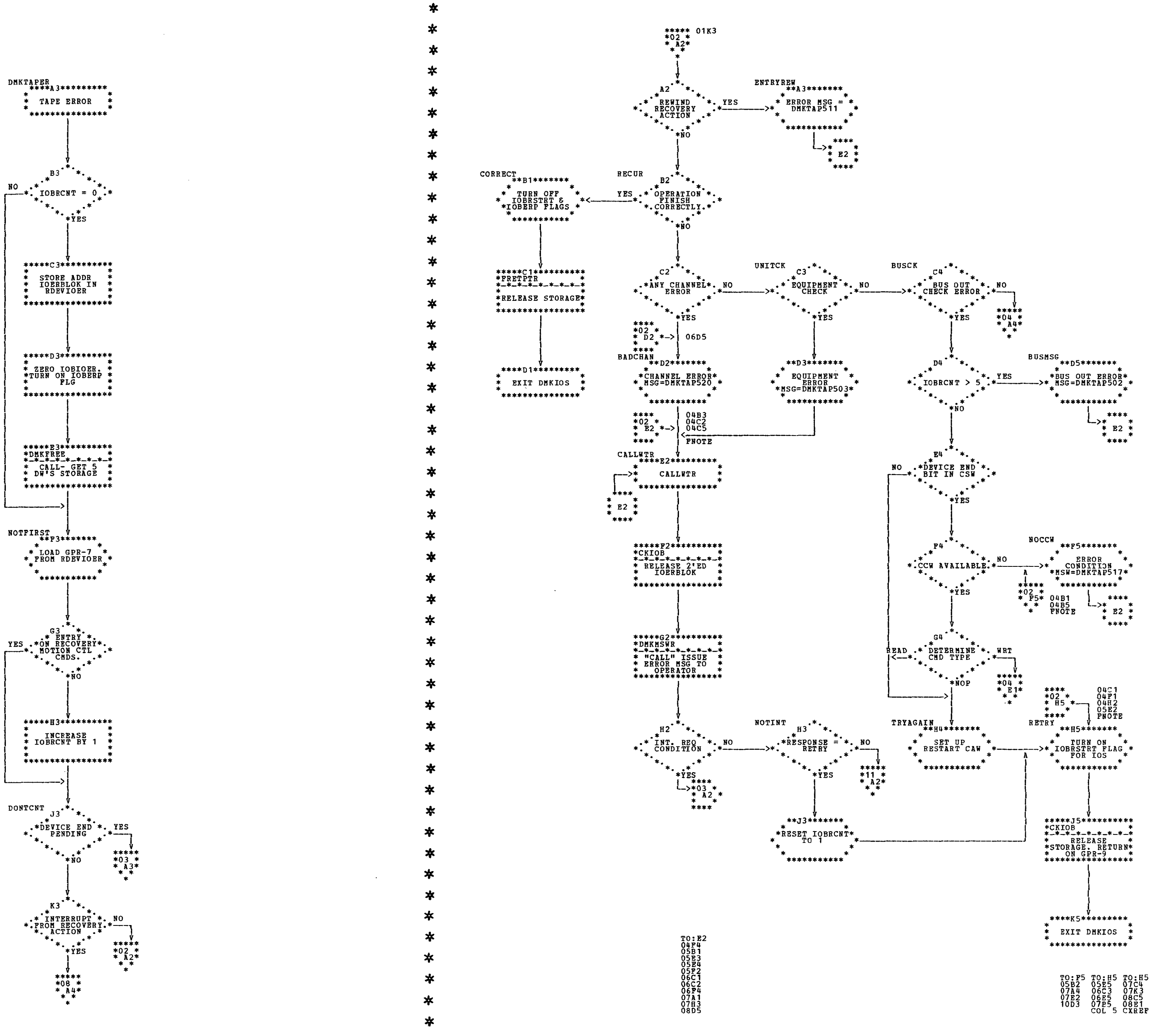
DMKSSP -- Build Real I/O Blocks for Starter System or at IPL Time (Part 3 of 3)

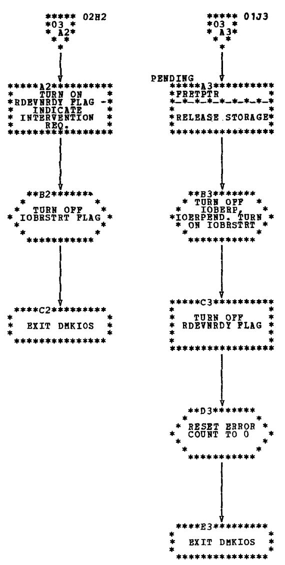

READADDR
A1
READADDR
B1
PRINT LINE TO CONS, AND READ DEVICE ADDRESS
C1
DMKCVTHB
CALL- CONVERT DEVICE ADDRESS
D1
VALID DEVICE ADDRESS
NO
YES
READTYPE
E1
PRINT LINE TO CONS AND READ DEVICE TYPE
F1
VALID DEVICE TYPE
NO
YES
G1
RETURN ON GPR-5
SCAN
A2
SCAN
B2
DMKSCNRU
CALL- TEST IF DEVICE DEFINE PREVIOUSLY
C2
RDEVBLOK FOUND
YES
NO
D2
CALCULATE I-O BLOCKS FOR THIS DEVICE
E2
BUILD A RCHBLOK, RCUBLOK AND RDEVBLOK.
F2
BUILD ENTRIES FOR SYSTEM SPOOLING DEVICES
G2
RETURN ON GPR-5 + 4
WNGDEV
C3
DEVICE TYPE TAPE
NO
YES
D3
RETURN ON GPR-5 + 4
TYPEMSG
C4
ERROR- DUPLICATE DEVICE ASSIGNMENT
D4
RETURN ON GPR-5 + 0

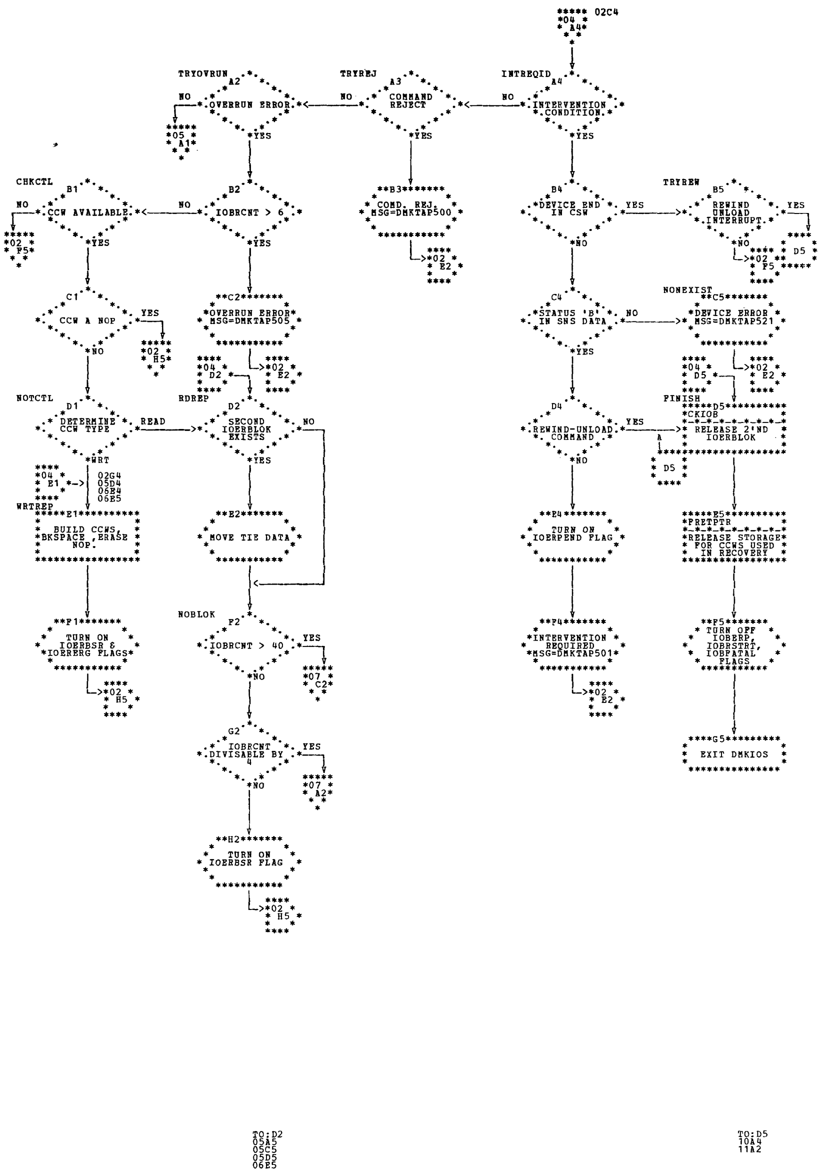

DMKSTKCP
A2
DMKSTKCP
STACK A CPEXBLOK FOR DMKDSP PROCESSING
C2
GET LAST CPEXBLOK IN CHAIN. ADD THIS ONE TO THE END
D2
BUILD THE BACKWARD CHAIN POINTERS
E2
R14 RETURN
DMKSTKIO
A3
DMKSTKCP
STACK A IOBLOK FOR DMKDSP PROCESSING
C3
GET THE LAST IOBLOK IN THE CHAIN. ADD THIS ONE TO THE END
D3
BUILD THE BACKWARD CHAIN POINTERS.
E3
R14 RETURN

DMKSTK -- Stack (queue) a CPEXBLOK or IOBLOK for Dispatching (Part 1 of 1)

| DMKTAP -- Tape Error Reccvery Procedures (Parts 1 and 2 of 11)

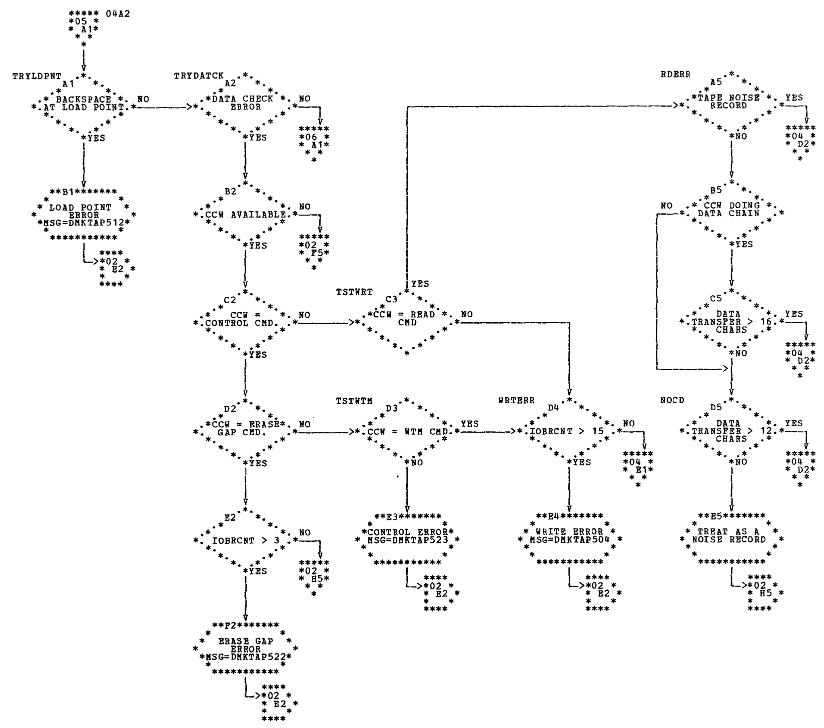


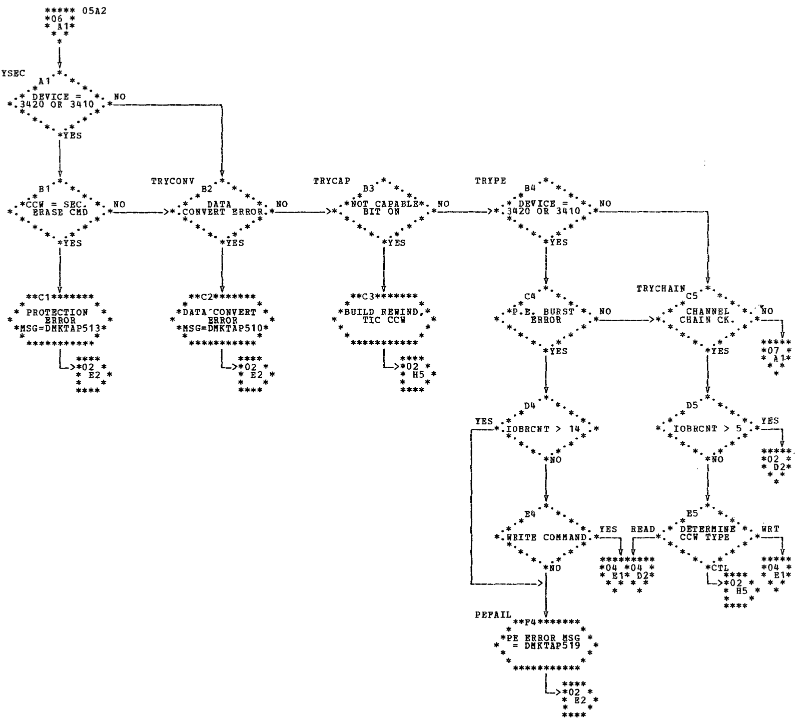


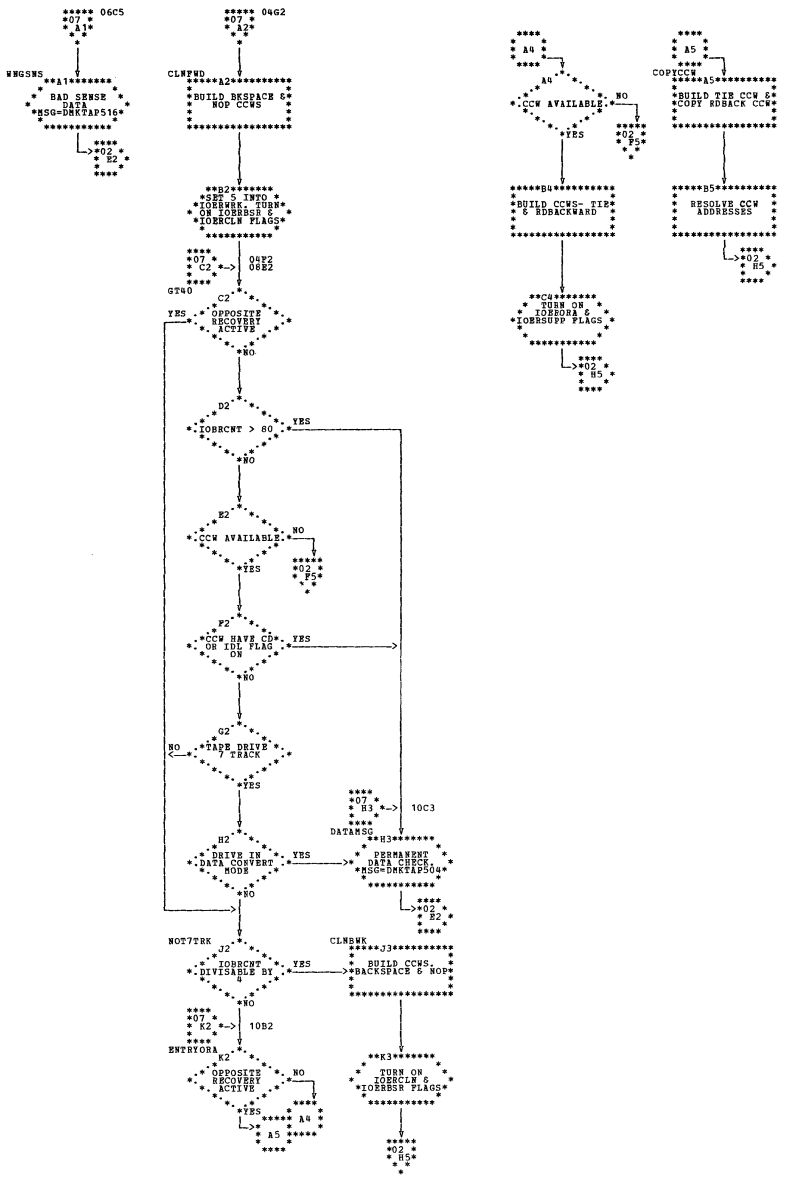


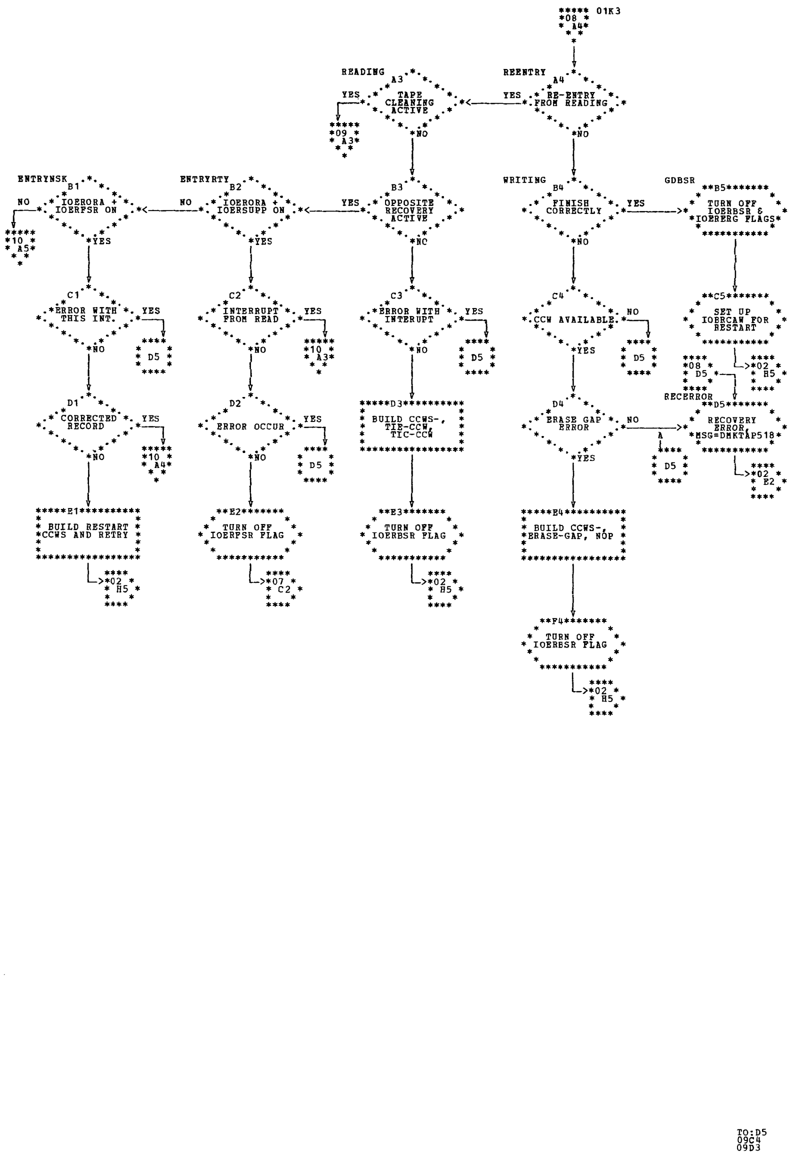
| DMKTAP -- Tape Error Recovery Procedures (Parts 3 and 4 of 11)

| DMKTAP -- Tape Error Reccvery Procedures (Parts 5 and 6 of 11)



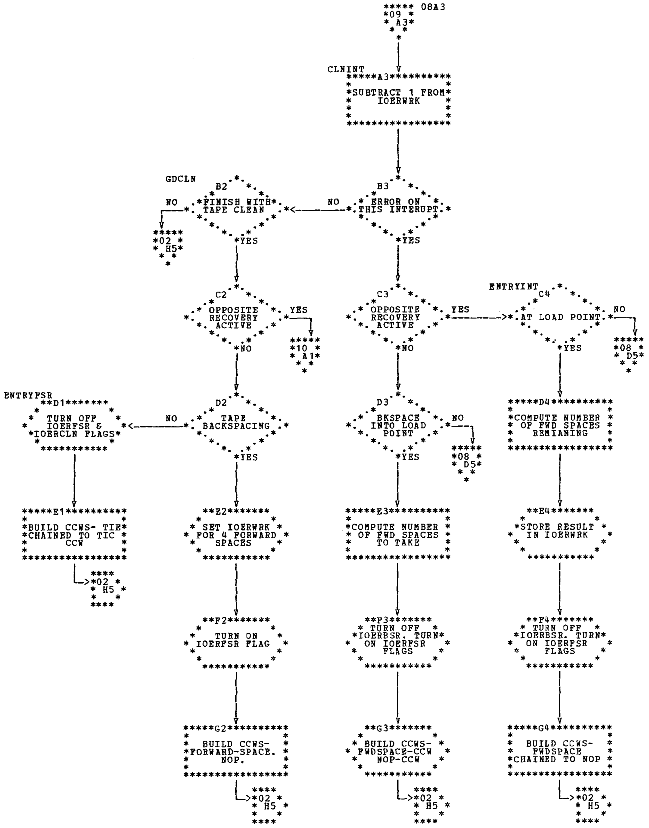


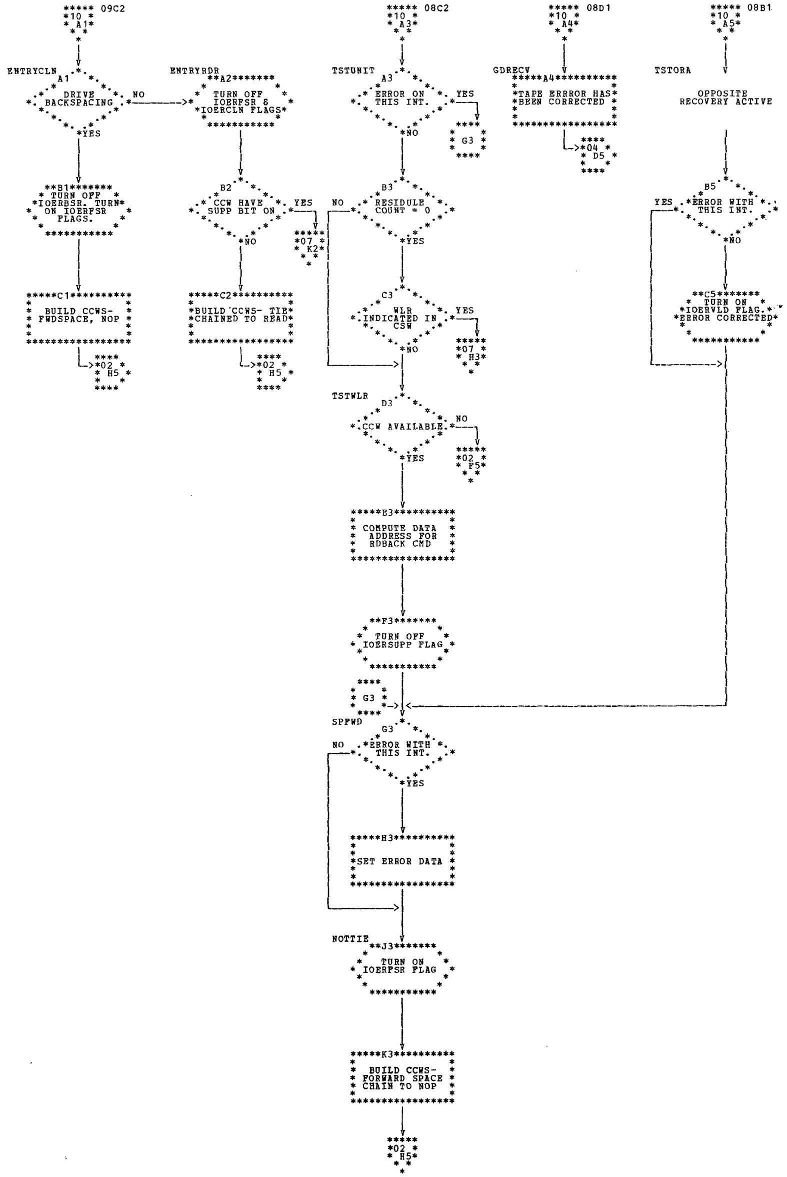

07 A1 06C5
WNGSNS A1 BAD SENSE DATA MSG=DMKTAP516
02 E2
07 A2 04G2
CLNFWD A2 BUILD BKSPACE & NOP CCWS
B2 SET 5 INTO IOERWRK. TURN ON IOERBSR & IOERCLN FLAGS
07 C2 04F2 08E2
GT40 C2 OPPOSITE RECOVERY ACTIVE
YES
NO
D2 IOBRCNT > 80
YES
NO
E2 CCW AVAILABLE
NO
02 F5
YES
F2 CCW HAVE CD OR IDL FLAG ON
YES
NO
G2 TAPE DRIVE 7 TRACK
NO
YES
H2 DRIVE IN DATA CONVERT MODE
YES
NO
07 H3 10C3
DATAMSG H3 PERMANENT DATA CHECK. MSG=DMKTAP504
02 E2
NOT7TRK J2 IOBRCNT DIVISABLE BY 4
YES
NO
CLNBWK J3 BUILD CCWS. BACKSPACE & NOP
07 K2 10B2
ENTRYORA K2 OPPOSITE RECOVERY ACTIVE
NO
YES
A4
A5
K3 TURN ON IOERCLN & IOERBSR FLAGS
02 H5
A4
A4 CCW AVAILABLE
NO
02 F5
YES
B4 BUILD CCWS- TIE & RDBACKWARD
C4 TURN ON IOERORA & IOERSUPP FLAGS
02 H5
A5
COPYCCW A5 BUILD TIE CCW & COPY RDBACK CCW
B5 RESOLVE CCW ADDRESSES
02 H5

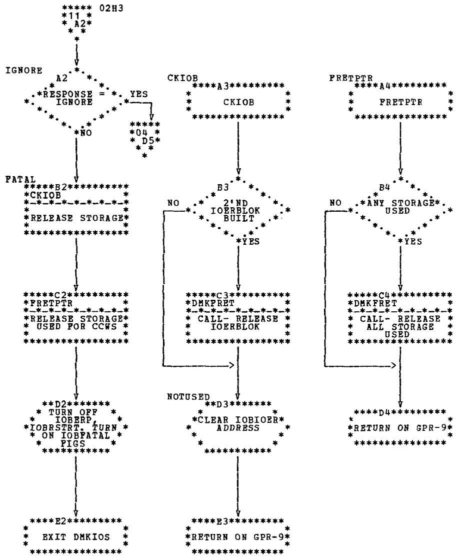

08 A4 01K3
REENTRY A4 RE-ENTRY FROM READING
YES
NO
READING A3 TAPE CLEANING ACTIVE
YES
09 A3
NO
ENTRYNSK B1 IOERORA + IOERFSR ON
NO
10 A5
YES
C1 ERROR WITH THIS INT.
YES
D5
NO
D1 CORRECTED RECORD
YES
10 A4
NO
E1 BUILD RESTART CCWS AND RETRY
02 H5
ENTRYRTY B2 IOERORA + IOERSUPP ON
NO
YES
C2 INTERRUPT FROM READ
YES
10 A3
NO
D2 ERROR OCCUR
YES
D5
NO
E2 TURN OFF IOERFSR FLAG
07 C2
B3 OPPOSITE RECOVERY ACTIVE
YES
NC
C3 ERROR WITH INTERUPT
YES
D5
NO
D3 BUILD CCWS-, TIE-CCW, TIC-CCW
E3 TURN OFF IOERBSR FLAG
02 H5
WRITING B4 FINISH CORRECTLY
YES
NO
C4 CCW AVAILABLE
NO
D5
YES
D4 ERASE GAP ERROR
NO
D5
YES
E4 BUILD CCWS-, ERASE-GAP, NOP
F4 TURN OFF IOERBSR FLAG
02 H5
GDBSR B5 TURN OFF IOERBSR & IOERERG FLAGS
C5 SET UP IOBRCAW FOR RESTART
02 H5
08 D5
RECERROR D5 RECOVERY ERROR, MSG=DMKTAP518
02 E2
TO:D5
09C4
09D3

| DMKTAP -- Tape Error Recovery Procedures (Part 7 and 8 of 10)

| DMKTAP -- Tape Error Reccvery Procedures (Parts 9 and 10 of 11)

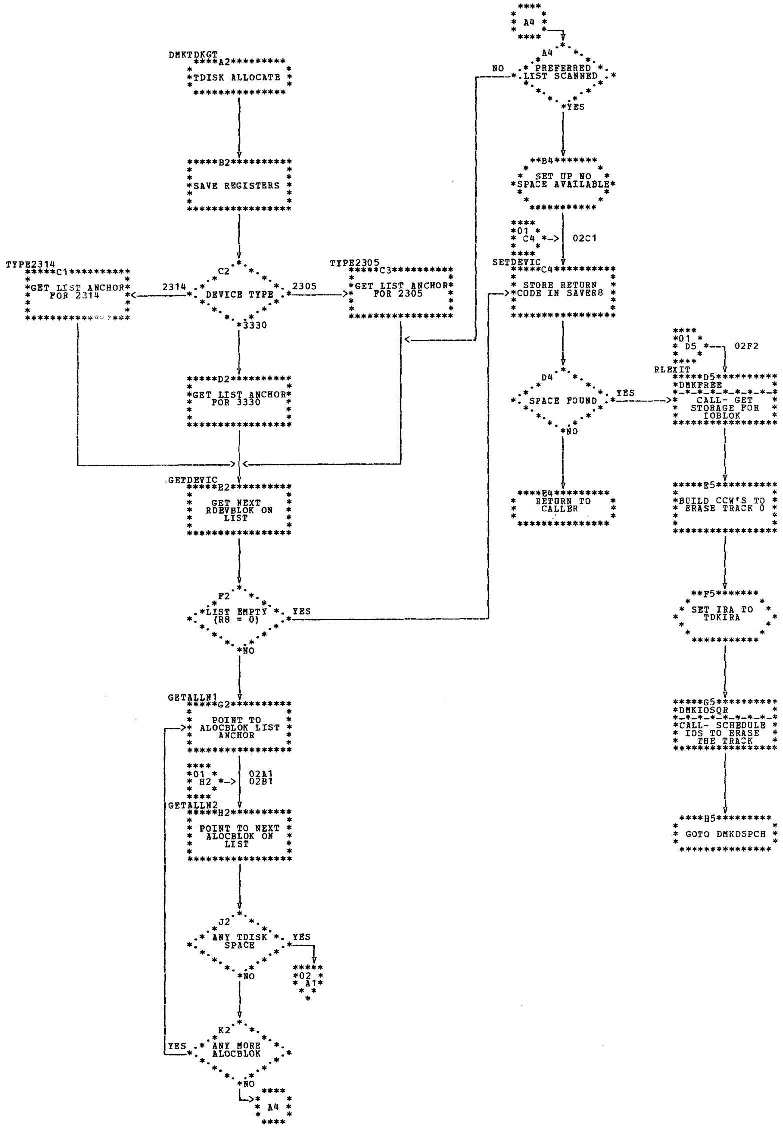


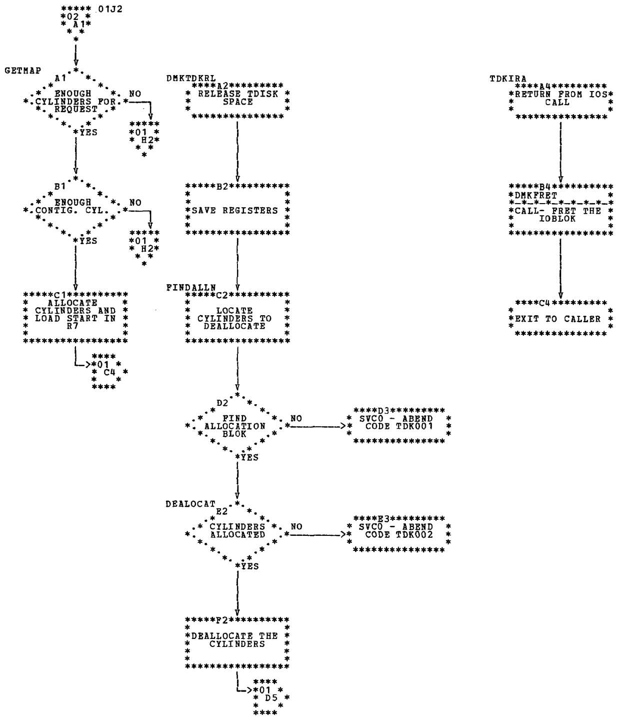


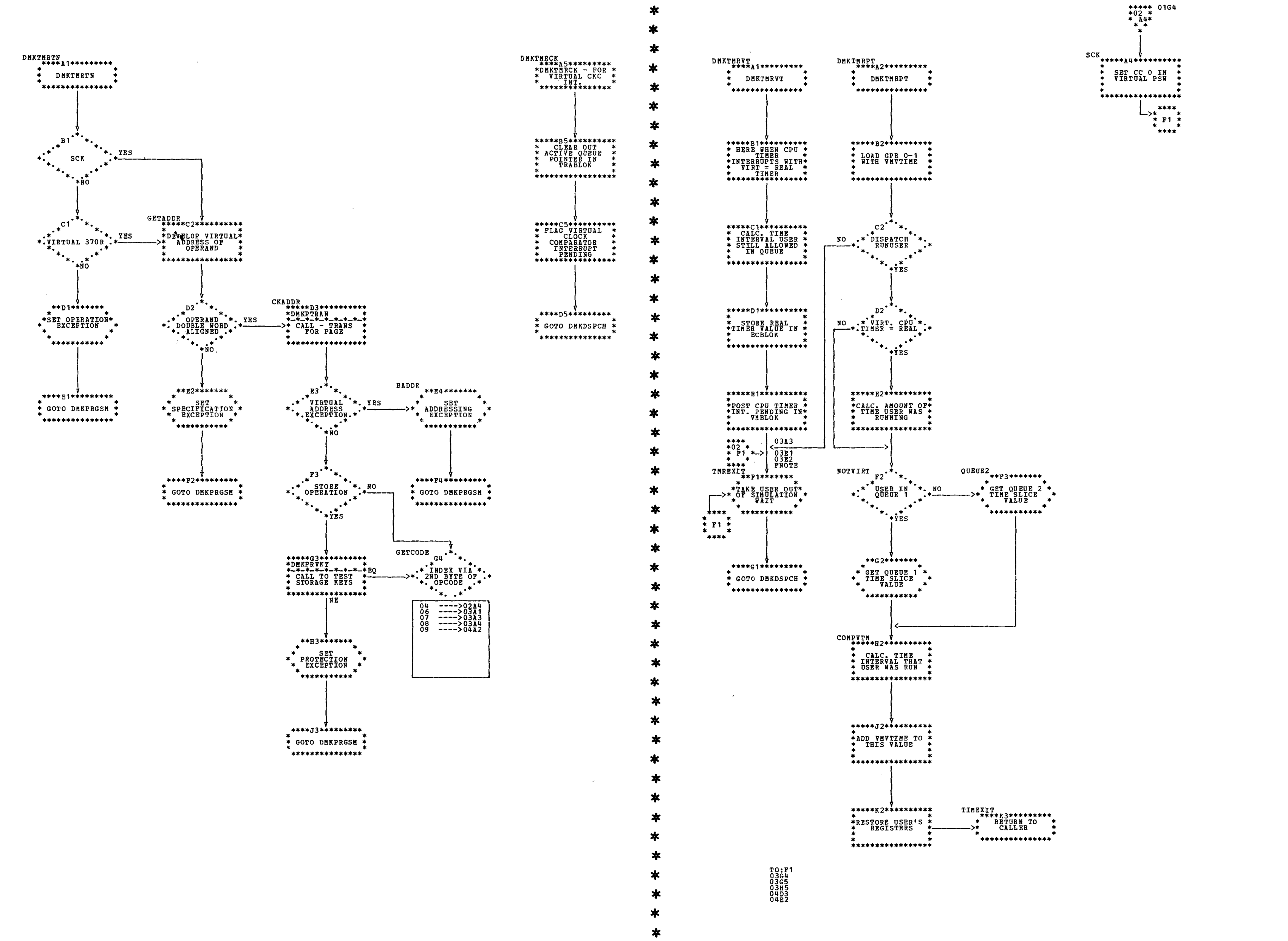


| DMKTAP -- Tape Error Recovery Procedures (Part 11 of 11)

| DMKTDK -- T-Disk Space Manager (Parts 1 and 2 of 2)

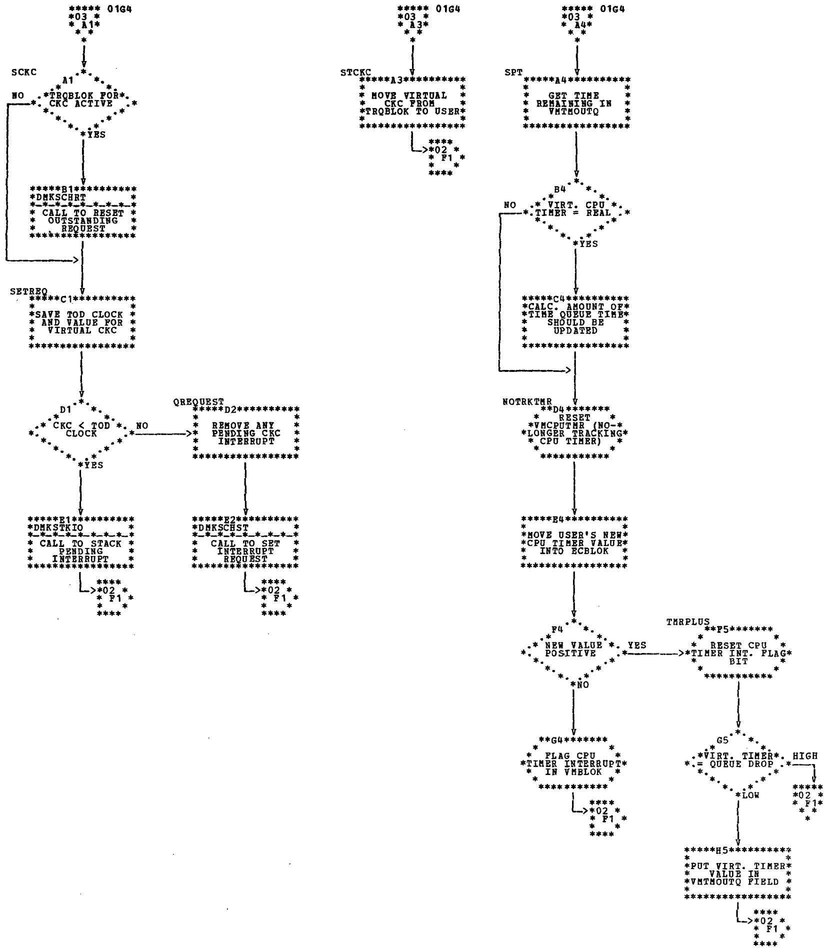

A4
A4 PREFERRED LIST SCANNED
NO
YES
DMKTDKGT
A2 TDISK ALLOCATE
B2 SAVE REGISTERS
B4 SET UP NO SPACE AVAILABLE
01 C4 02C1
TYPE2314
C1 GET LIST ANCHOR FOR 2314
2314
C2 DEVICE TYPE
2305
TYPE2305
C3 GET LIST ANCHOR FOR 2305
3330
SETDEVIC
C4 STORE RETURN CODE IN SAVER8
01 D5 02F2
RLEXIT
D2 GET LIST ANCHOR FOR 3330
D4 SPACE FOUND
YES
NO
D5 DMKFREE CALL- GET STORAGE FOR IOBLOK
GETDEVIC
E2 GET NEXT RDEVBLOK ON LIST
E4 RETURN TO CALLER
E5 BUILD CCW'S TO ERASE TRACK 0
F2 LIST EMPTY (R8 = 0)
YES
NO
F5 SET IRA TO TDKIRA
GETALLN1
G2 POINT TO ALOCBLOK LIST ANCHOR
G5 DMKIOSQR CALL- SCHEDULE IOS TO ERASE THE TRACK
01 H2 02A1 02B1
GETALLN2
H2 POINT TO NEXT ALOCBLOK ON LIST
H5 GOTO DMKDSPCH
J2 ANY TDISK SPACE
YES
02 A1
NO
K2 ANY MORE ALOCBLOK
YES
NO
A4

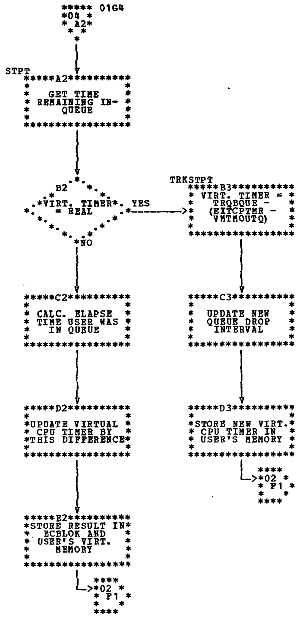


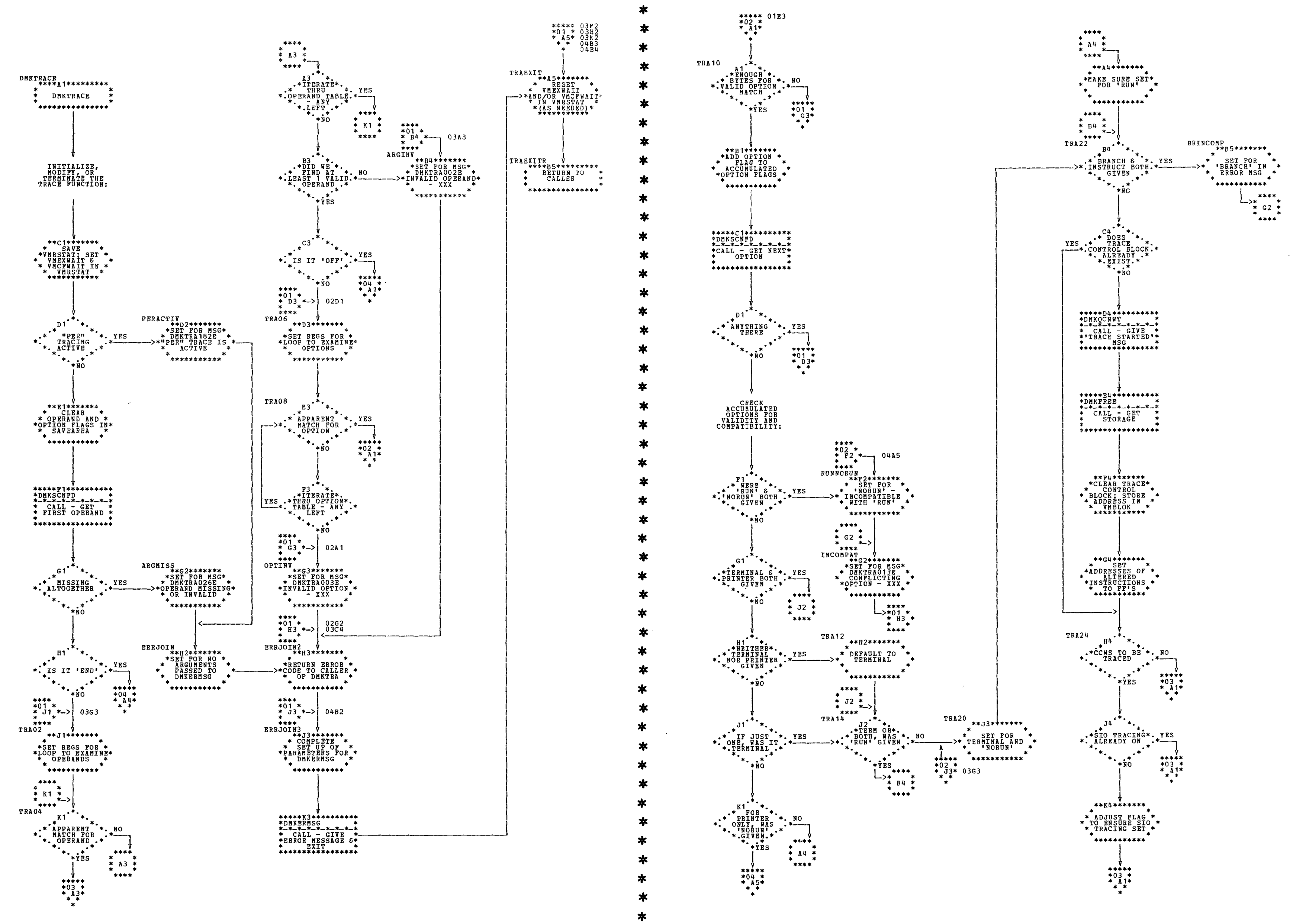


| DMKTMR -- Virtual CPU Timer and Clock Comparator Simulator (Parts 1 and 2 of 4)

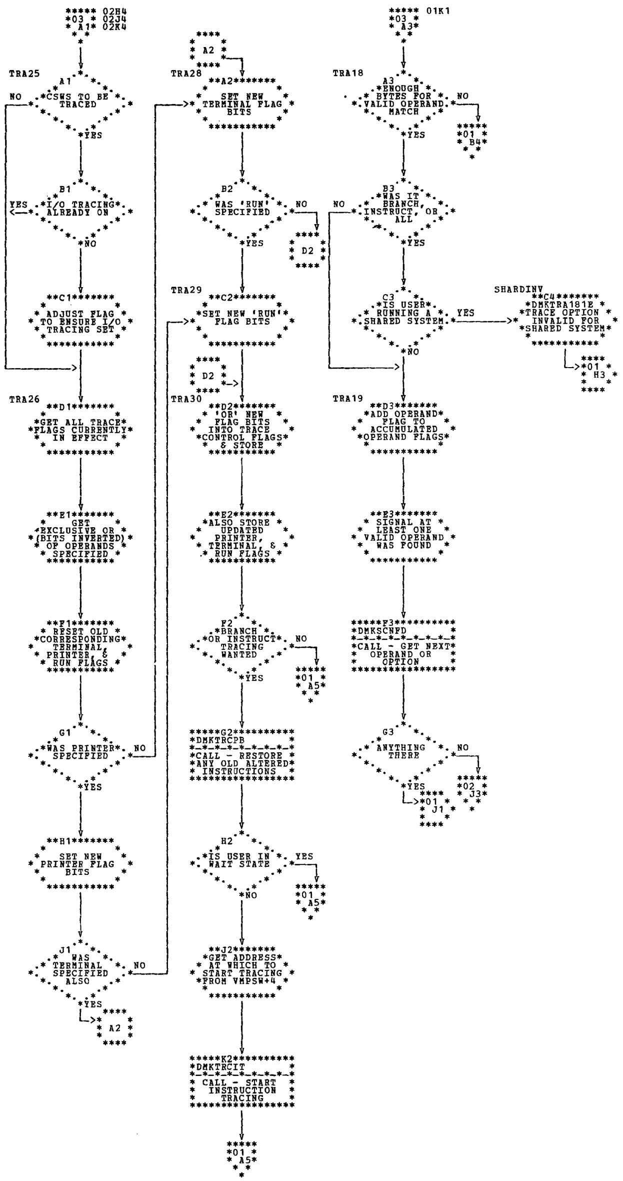
DMKTMR -- Virtual CPU Timer and Clock Comparator Simulator (Parts 3 and 4 of 4)

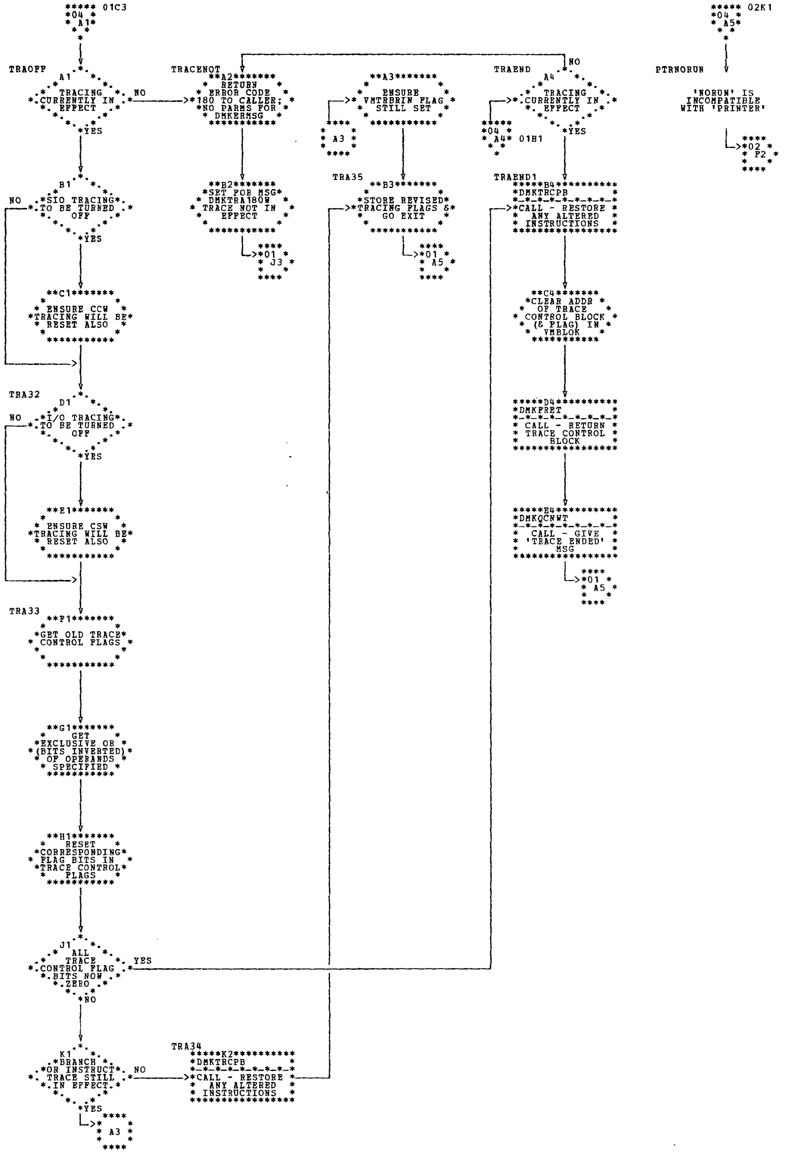

01G4
03 A1
SCKC
A1
TRQBLOK FOR CKC ACTIVE
NO
YES
B1
DMKSCHRT
CALL TO RESET OUTSTANDING REQUEST
SETREQ
C1
SAVE TOD CLOCK AND VALUE FOR VIRTUAL CKC
D1
CKC < TOD CLOCK
NO
YES
QREQUEST
D2
REMOVE ANY PENDING CKC INTERRUPT
E1
DMKSTKIO
CALL TO STACK PENDING INTERRUPT
02 F1
E2
DMKSCHST
CALL TO SET INTERRUPT REQUEST
02 F1
01G4
03 A3
STCKC
A3
MOVE VIRTUAL CKC FROM TRQBLOK TO USER
02 F1
01G4
03 A4
SPT
A4
GET TIME REMAINING IN VMTMOUTQ
B4
VIRT. CPU TIMER = REAL
NO
YES
C4
CALC. AMOUNT OF TIME QUEUE TIME SHOULD BE UPDATED
NOTRKTMR
D4
RESET VMCPUTMR (NO-LONGER TRACKING CPU TIMER)
E4
MOVE USER'S NEW CPU TIMER VALUE INTO ECBLOK
F4
NEW VALUE POSITIVE
YES
NO
TMRPLUS
F5
RESET CPU TIMER INT. FLAG BIT
G4
FLAG CPU TIMER INTERRUPT IN VMBLOK
02 F1
G5
VIRT. TIMER = QUEUE DROP
HIGH
LOW
02 F1
H5
PUT VIRT. TIMER VALUE IN VMTMOUTQ FIELD
02 F1


01G4
04 A2
STPT
A2
GET TIME REMAINING IN-QUEUE
B2
VIRT. TIMER = REAL
YES
NO
TRKSTPT
B3
VIRT. TIMER = TRQBQUE - (EXTCPTMR - VMTMOUTQ)
C2
CALC. ELAPSE TIME USER WAS IN QUEUE
C3
UPDATE NEW QUEUE DROP INTERVAL
D2
UPDATE VIRTUAL CPU TIMER BY THIS DIFFERENCE
D3
STORE NEW VIRT. CPU TIMER IN USER'S MEMORY
02 F1
E2
STORE RESULT IN ECBLOK AND USER'S VIRT. MEMORY
02 F1


DMKTRACE
A1
DMKTRACE
INITIALIZE, MODIFY, OR TERMINATE THE TRACE FUNCTION:
C1
SAVE VMRSTAT; SET VMEXWAIT & VMCFWAIT IN VMRSTAT
D1
"PER" TRACING ACTIVE
YES
NO
PERACTIV
D2
SET FOR MSG* DMKTRA182E "PER" TRACE IS ACTIVE
E1
CLEAR OPERAND AND OPTION FLAGS IN SAVEAREA
F1
DMKSCNFD
CALL - GET FIRST OPERAND
G1
MISSING ALTOGETHER
YES
NO
ARGMISS
G2
SET FOR MSG DMKTRA026E OPERAND MISSING OR INVALID
ERRJOIN
H2
SET FOR NO ARGUMENTS PASSED TO DMKERMSG
H1
IS IT 'END'
YES
04 A4
NO
01 J1
03G3
TRA02
J1
SET REGS FOR LOOP TO EXAMINE OPERANDS
K1
TRA04
K1
APPARENT MATCH FOR OPERAND
NO
A3
YES
03 A3
A3
A3
ITERATE THRU OPERAND TABLE - ANY LEFT
YES
K1
NO
B3
DID WE FIND AT LEAST 1 VALID OPERAND
NO
YES
01 B4
03A3
ARGINV
B4
SET FOR MSG DMKTRA002E INVALID OPERAND - XXX
C3
IS IT 'OFF'
YES
04 A1
NO
01 D3
02D1
TRA06
D3
SET REGS FOR LOOP TO EXAMINE OPTIONS
TRA08
E3
APPARENT MATCH FOR OPTION
YES
02 A1
NO
F3
ITERATE THRU OPTION TABLE - ANY LEFT
YES
NO
01 G3
02A1
OPTINV
G3
SET FOR MSG DMKTRA003E INVALID OPTION - XXX
01 H3
02G2
03C4
ERRJOIN2
H3
RETURN ERROR CODE TO CALLER OF DMKTRA
01 J3
04B2
ERRJOIN3
J3
COMPLETE SET UP OF PARAMETERS FOR DMKERMSG
K3
DMKERMSG
CALL - GIVE ERROR MESSAGE & EXIT
01 A5
03F2
03H2
03K2
04B3
04B4
TRAEXIT
A5
RESET VMEXWAIT AND/OR VMCFWAIT IN VMRSTAT (AS NEEDED)
TRAEXITR
B5
RETURN TO CALLER
02 A1
01E3
TRA10
A1
ENOUGH BYTES FOR VALID OPTION MATCH
NO
01 G3
YES
B1
ADD OPTION FLAG TO ACCUMULATED OPTION FLAGS
C1
DMKSCNFD
CALL - GET NEXT OPTION
D1
ANYTHING THERE
YES
01 D3
NO
CHECK ACCUMULATED OPTIONS FOR VALIDITY AND COMPATIBILITY:
F1
WERE 'RUN' & 'NORUN' BOTH GIVEN
YES
NO
02 F2
04A5
RUNNORUN
F2
SET FOR 'NORUN' - INCOMPATIBLE WITH 'RUN'
G2
INCOMPAT
G2
SET FOR MSG DMKTRA013E CONFLICTING OPTION - XXX
01 H3
G1
TERMINAL & PRINTER BOTH GIVEN
YES
J2
NO
H1
NEITHER TERMINAL NOR PRINTER GIVEN
YES
NO
TRA12
H2
DEFAULT TO TERMINAL
J2
J1
IF JUST ONE, WAS IT TERMINAL
YES
NO
TRA14
J2
TERM OR BOTH, WAS 'RUN' GIVEN
NO
YES
B4
A
02 J3
03G3
TRA20
J3
SET FOR TERMINAL AND 'NORUN'
K1
FOR PRINTER ONLY, WAS 'NORUN' GIVEN
NO
A4
YES
04 A5
A4
A4
MAKE SURE SET FOR 'RUN'
B4
TRA22
B4
BRANCH & INSTRUCT BOTH GIVEN
YES
NO
BRINCOMP
B5
SET FOR 'BRANCH' IN ERROR MSG
G2
C4
DOES TRACE CONTROL BLOCK ALREADY EXIST
YES
NO
D4
DMKQCNWT
CALL - GIVE 'TRACE STARTED' MSG
E4
DMKFREE
CALL - GET STORAGE
F4
CLEAR TRACE CONTROL BLOCK; STORE ADDRESS IN VMBLOK
G4
SET ADDRESSES OF ALTERED INSTRUCTIONS TO FF'S
TRA24
H4
CCWS TO BE TRACED
NO
03 A1
YES
J4
SIO TRACING ALREADY ON
YES
03 A1
NO
K4
ADJUST FLAG TO ENSURE SIO TRACING SET
03 A1

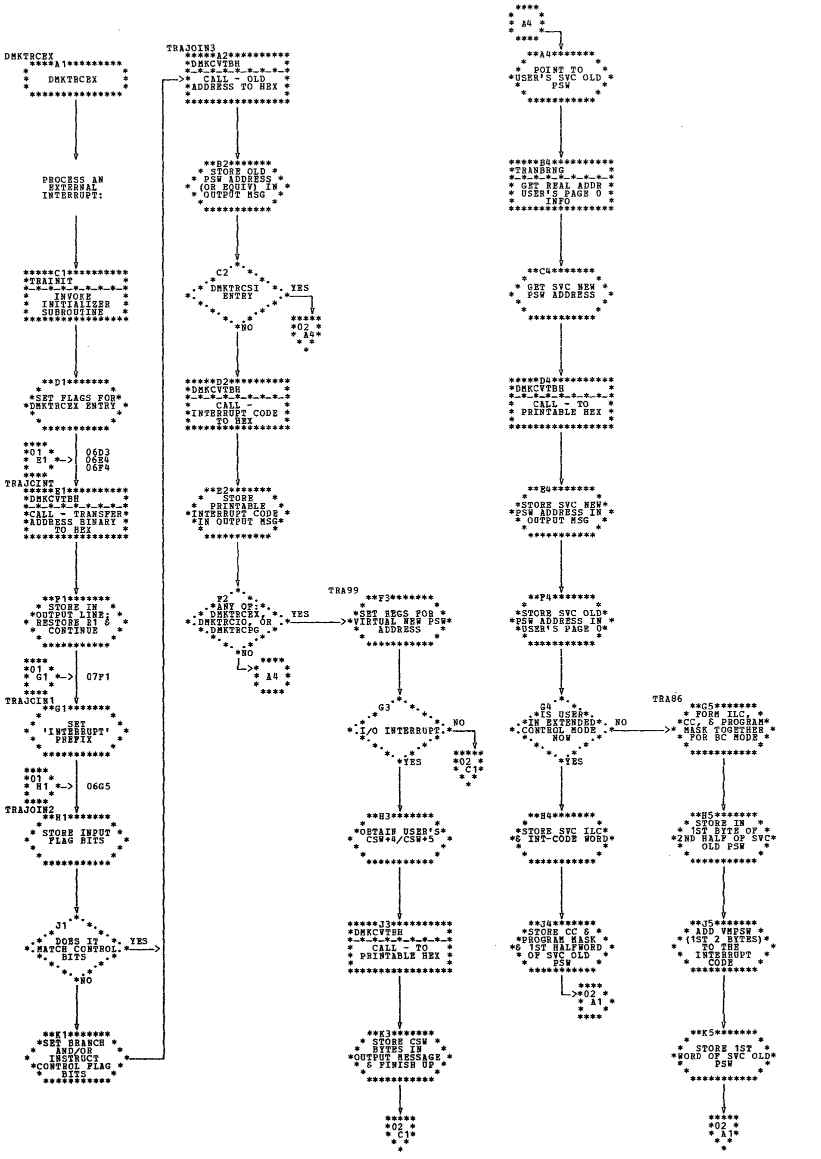
DMKTRA -- Virtual Machine TRACE Command Initialization (Parts 1 and 2 of 4)

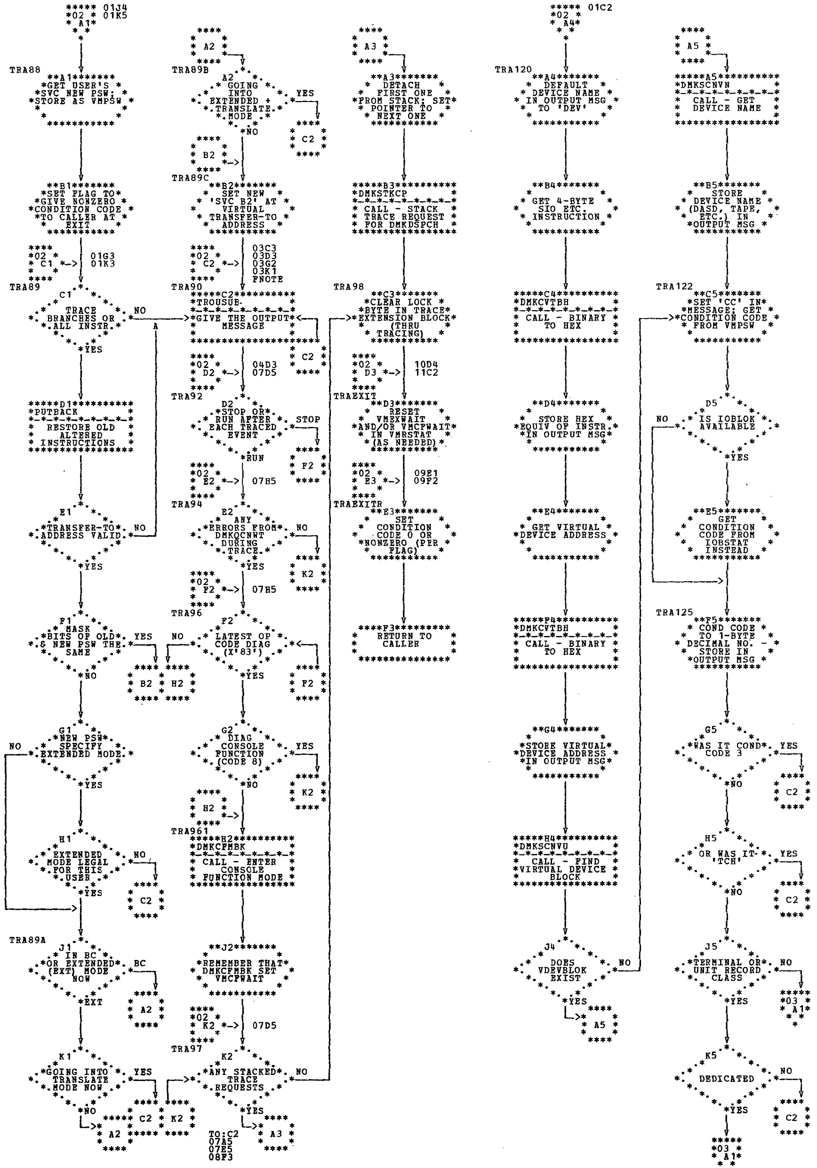

03 A1 02H4 02J4 02K4
TRA25
A1 CSWS TO BE TRACED
NO
YES
B1 I/O TRACING ALREADY ON
YES
NO
C1 ADJUST FLAG TO ENSURE I/O TRACING SET
TRA26
D1 GET ALL TRACE FLAGS CURRENTLY IN EFFECT
E1 GET EXCLUSIVE OR (BITS INVERTED) OF OPERANDS SPECIFIED
F1 RESET OLD CORRESPONDING TERMINAL, PRINTER, & RUN FLAGS
G1 WAS PRINTER SPECIFIED
NO
YES
H1 SET NEW PRINTER FLAG BITS
J1 WAS TERMINAL SPECIFIED ALSO
NO
YES
A2
A2
TRA28
A2 SET NEW TERMINAL FLAG BITS
B2 WAS 'RUN' SPECIFIED
NO
D2
YES
TRA29
C2 SET NEW 'RUN' FLAG BITS
D2
TRA30
D2 'OR' NEW FLAG BITS INTO TRACE CONTROL FLAGS & STORE
E2 ALSO STORE UPDATED PRINTER, TERMINAL, & RUN FLAGS
F2 BRANCH OR INSTRUCT TRACING WANTED
NO
01 A5
YES
G2 DMKTRCPB
CALL - RESTORE ANY OLD ALTERED INSTRUCTIONS
H2 IS USER IN WAIT STATE
YES
01 A5
NO
J2 GET ADDRESS AT WHICH TO START TRACING FROM VMPSW+4
K2 DMKTRCIT
CALL - START INSTRUCTION TRACING
01 A5
03 A3 01K1
TRA18
A3 ENOUGH BYTES FOR VALID OPERAND MATCH
NO
01 B4
YES
B3 WAS IT BRANCH, INSTRUCT, OR ALL
NO
YES
C3 IS USER RUNNING A SHARED SYSTEM
YES
NO
SHARDINV
C4 DMKTRA181E TRACE OPTION INVALID FOR SHARED SYSTEM
01 H3
TRA19
D3 ADD OPERAND FLAG TO ACCUMULATED OPERAND FLAGS
E3 SIGNAL AT LEAST ONE VALID OPERAND WAS FOUND
F3 DMKSCNFD
CALL - GET NEXT OPERAND OR OPTION
G3 ANYTHING THERE
NO
02 J3
YES
01 J1

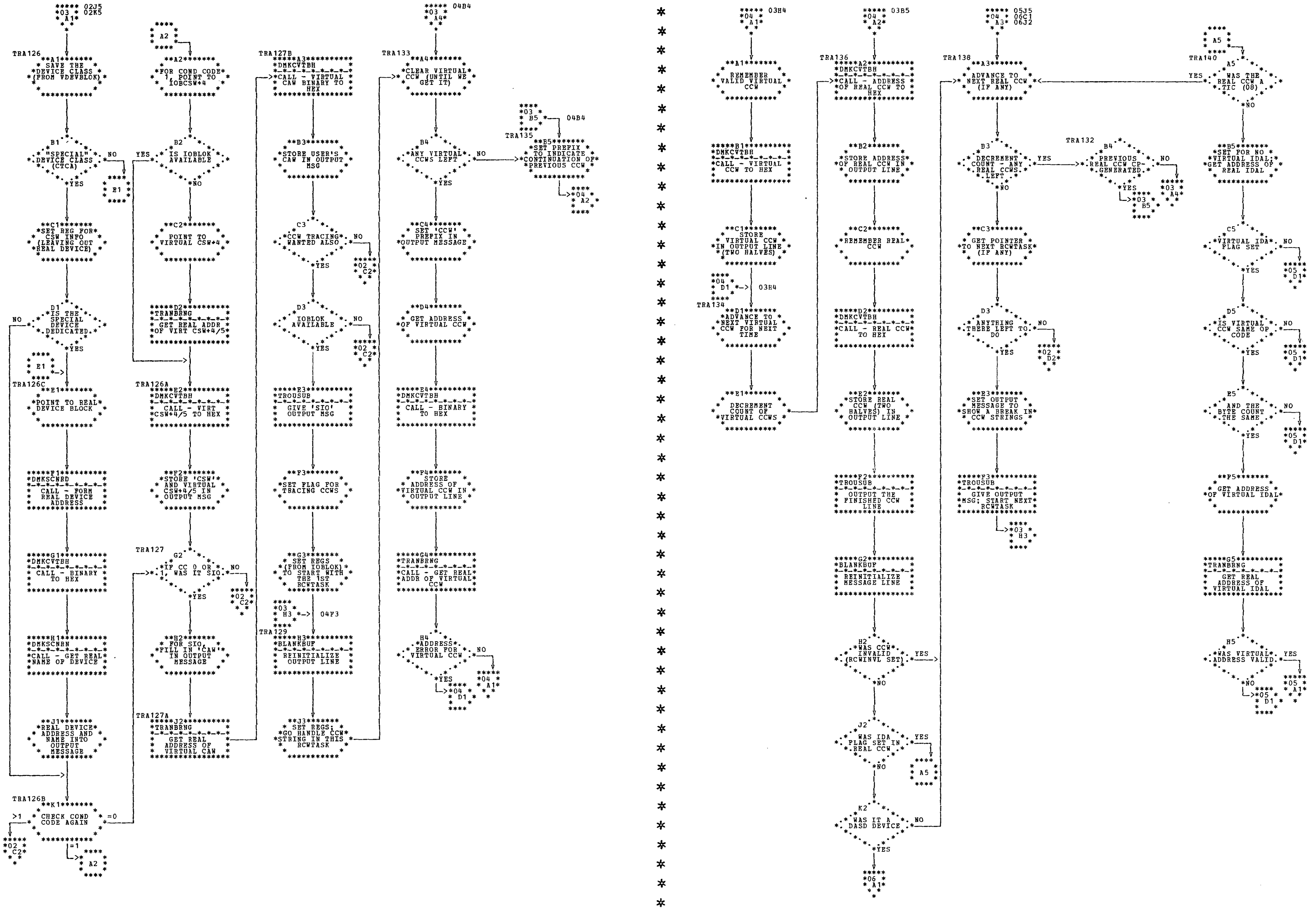

04 A1 01C3
TRAOFF
A1 TRACING CURRENTLY IN EFFECT
NO
YES
B1 SIO TRACING TO BE TURNED OFF
NO
YES
C1 ENSURE CCW TRACING WILL BE RESET ALSO
TRA32
D1 I/O TRACING TO BE TURNED OFF
NO
YES
E1 ENSURE CSW TRACING WILL BE RESET ALSO
TRA33
F1 GET OLD TRACE CONTROL FLAGS
G1 GET EXCLUSIVE OR (BITS INVERTED) OF OPERANDS SPECIFIED
H1 RESET CORRESPONDING FLAG BITS IN TRACE CONTROL FLAGS
J1 ALL TRACE CONTROL FLAG BITS NOW ZERO
YES
NO
K1 BRANCH OR INSTRUCT TRACE STILL IN EFFECT
NO
YES
A3
TRACENOT
A2 RETURN ERROR CODE 180 TO CALLER; NO PARMS FOR DMKERMSG
B2 SET FOR MSG DMKTRA180W TRACE NOT IN EFFECT
01 J3
TRA34
K2 DMKTRCPB
CALL - RESTORE ANY ALTERED INSTRUCTIONS
A3
A3 ENSURE VMTRBRIN FLAG STILL SET
TRA35
B3 STORE REVISED TRACING FLAGS & GO EXIT
01 A5
TRAEND
NO
A4 TRACING CURRENTLY IN EFFECT
04 A4 01H1
YES
TRAEND1
B4 DMKTRCPB
CALL - RESTORE ANY ALTERED INSTRUCTIONS
C4 CLEAR ADDR OF TRACE CONTROL BLOCK (& FLAG) IN VMBLOK
D4 DMKFRET
CALL - RETURN TRACE CONTROL BLOCK
E4 DMKQCNWT
CALL - GIVE 'TRACE ENDED' MSG
01 A5
04 A5 02K1
PTRNORUN
'NORUN' IS INCOMPATIBLE WITH 'PRINTER'
02 F2

| DMKTRA -- Virtual Machine TRACE Command Initializaticn (Parts 3 and 4 of 4)

| DMKTRC -- Virtual Machine Tracing Supervisor (Parts 1 and 2 of 13)

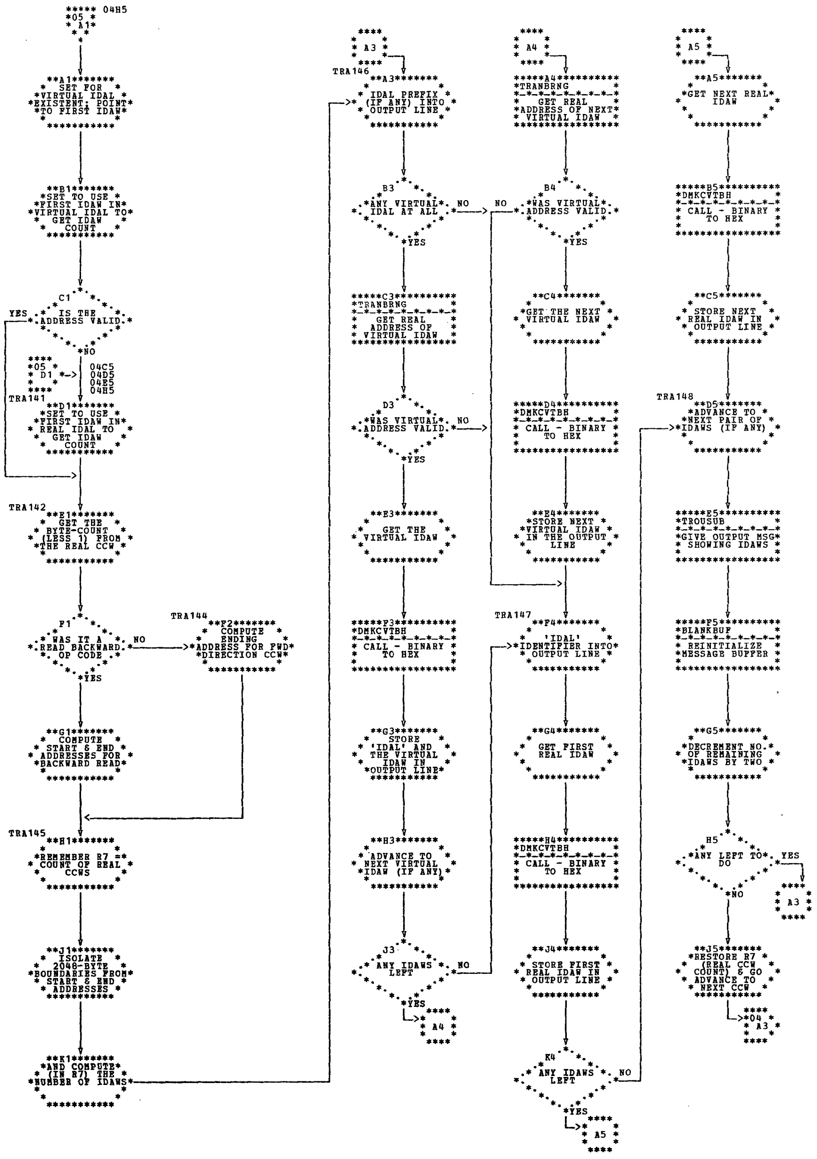

DMKTRCEX
A1
DMKTRCEX
PROCESS AN EXTERNAL INTERRUPT:
C1
TRAINIT
INVOKE INITIALIZER SUBROUTINE
D1
SET FLAGS FOR DMKTRCEX ENTRY
01 E1
06D3
06E4
06F4
TRAJCINT
E1
DMKCVTBH
CALL - TRANSFER ADDRESS BINARY TO HEX
F1
STORE IN OUTPUT LINE; RESTORE R1 & CONTINUE
01 G1
07F1
TRAJCIN1
G1
SET 'INTERRUPT' PREFIX
01 H1
06G5
TRAJOIN2
H1
STORE INPUT FLAG BITS
J1
DOES IT MATCH CONTROL BITS
YES
NO
K1
SET BRANCH AND/OR INSTRUCT CONTROL FLAG BITS
TRAJOIN3
A2
DMKCVTBH
CALL - OLD ADDRESS TO HEX
B2
STORE OLD PSW ADDRESS (OR EQUIV) IN OUTPUT MSG
C2
DMKTRCSI ENTRY
YES
02 A4
NO
D2
DMKCVTBH
CALL - INTERRUPT CODE TO HEX
E2
STORE PRINTABLE INTERRUPT CODE IN OUTPUT MSG
F2
ANY OF: DMKTRCEX, DMKTRCIO, OR DMKTRCPG
YES
NO
A4
TRA99
F3
SET REGS FOR VIRTUAL NEW PSW ADDRESS
G3
I/O INTERRUPT
NO
02 C1
YES
H3
OBTAIN USER'S CSW+4/CSW+5
J3
DMKCVTBH
CALL - TO PRINTABLE HEX
K3
STORE CSW BYTES IN OUTPUT MESSAGE & FINISH UP
02 C1
A4
A4
POINT TO USER'S SVC OLD PSW
B4
TRANBRNG
GET REAL ADDR USER'S PAGE 0 INFO
C4
GET SVC NEW PSW ADDRESS
D4
DMKCVTBH
CALL - TO PRINTABLE HEX
E4
STORE SVC NEW PSW ADDRESS IN OUTPUT MSG
F4
STORE SVC OLD PSW ADDRESS IN USER'S PAGE 0
G4
IS USER IN EXTENDED CONTROL MODE NOW
NO
YES
H4
STORE SVC ILC & INT-CODE WORD
J4
STORE CC & PROGRAM MASK & 1ST HALFWORD OF SVC OLD PSW
02 A1
TRA86
G5
FORM ILC, CC, & PROGRAM MASK TOGETHER FOR BC MODE
H5
STORE IN 1ST BYTE OF 2ND HALF OF SVC OLD PSW
J5
ADD VMPSW (1ST 2 BYTES) TO THE INTERRUPT CODE
K5
STORE 1ST WORD OF SVC OLD PSW
02 A1

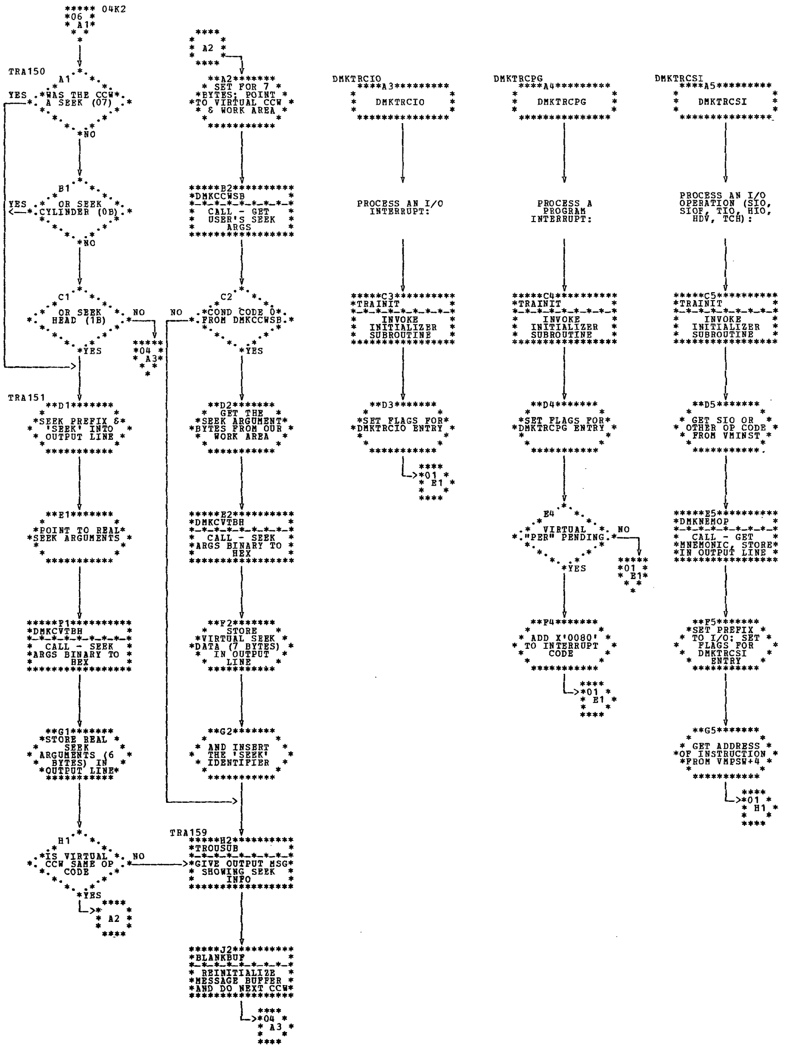

02 A1
01J4
01K5
TRA88
A1
GET USER'S SVC NEW PSW; STORE AS VMPSW
B1
SET FLAG TO GIVE NONZERO CONDITION CODE TO CALLER AT EXIT
02 C1
01G3
01K3
TRA89
C1
TRACE BRANCHES OR ALL INSTR.
NO
A
YES
D1
PUTBACK
RESTORE OLD ALTERED INSTRUCTIONS
E1
TRANSFER-TO ADDRESS VALID
NO
YES
F1
MASK BITS OF OLD & NEW PSW THE SAME
YES
B2
NO
G1
NEW PSW SPECIFY EXTENDED MODE
NO
YES
H1
EXTENDED MODE LEGAL FOR THIS USER
NO
C2
YES
TRA89A
J1
IN BC OR EXTENDED (EXT) MODE NOW
BC
A2
EXT
K1
GOING INTO TRANSLATE MODE NOW
YES
C2
NO
A2
A2
TRA89B
A2
GOING INTO EXTENDED + TRANSLATE MODE
YES
C2
NO
B2
TRA89C
B2
SET NEW 'SVC B2' AT VIRTUAL TRANSFER-TO ADDRESS
02 C2
03C3
03D3
03G2
03K1
FNOTE
TRA90
C2
TROUSUB
GIVE THE OUTPUT MESSAGE
C2
02 D2
04D3
07D5
TRA92
D2
STOP OR RUN AFTER EACH TRACED EVENT
STOP
F2
RUN
02 E2
07H5
TRA94
E2
ANY ERRORS FROM DMKQCNWT DURING TRACE
NO
K2
YES
02 F2
07H5
TRA96
F2
LATEST OP CODE DIAG (X'83')
NO
H2
F2
YES
G2
DIAG CONSOLE FUNCTION (CODE 8)
YES
K2
NO
H2
TRA961
H2
DMKCFMBK
CALL - ENTER CONSOLE FUNCTION MODE
J2
REMEMBER THAT DMKCFMBK SET VMCFWAIT
02 K2
07D5
TRA97
K2
ANY STACKED TRACE REQUESTS
NO
K2
YES
A3
TO:C2
07A5
07E5
08F3
A3
A3
DETACH FIRST ONE FROM STACK; SET POINTER TO NEXT ONE
B3
DMKSTKCP
CALL - STACK TRACE REQUEST FOR DMKDSPCH
TRA98
C3
CLEAR LOCK BYTE IN TRACE EXTENSION BLOCK (THRU TRACING)
02 D3
10D4
11C2
TRAEXIT
D3
RESET VMEXWAIT AND/OR VMCFWAIT IN VMRSTAT (AS NEEDED)
02 E3
09E1
09F2
TRAEXITR
E3
SET CONDITION CODE 0 OR NONZERO (PER FLAG)
F3
RETURN TO CALLER
02 A4
01C2
TRA120
A4
DEFAULT DEVICE NAME IN OUTPUT MSG TO 'DEV'
B4
GET 4-BYTE SIO ETC. INSTRUCTION
C4
DMKCVTBH
CALL - BINARY TO HEX
D4
STORE HEX EQUIV OF INSTR. IN OUTPUT MSG
E4
GET VIRTUAL DEVICE ADDRESS
F4
DMKCVTBH
CALL - BINARY TO HEX
G4
STORE VIRTUAL DEVICE ADDRESS IN OUTPUT MSG
H4
DMKSCNVU
CALL - FIND VIRTUAL DEVICE BLOCK
J4
DOES VDEVBLOK EXIST
NO
YES
A5
A5
A5
DMKSCNVN
CALL - GET DEVICE NAME
B5
STORE DEVICE NAME (DASD, TAPE, ETC.) IN OUTPUT MSG
TRA122
C5
SET 'CC' IN MESSAGE; GET CONDITION CODE FROM VMPSW
D5
IS IOBLOK AVAILABLE
NO
YES
E5
GET CONDITION CODE FROM IOBSTAT INSTEAD
TRA125
F5
COND CODE TO 1-BYTE DECIMAL NO. - STORE IN OUTPUT MSG
G5
WAS IT COND CODE 3
YES
C2
NO
H5
OR WAS IT 'TCH'
YES
C2
NO
J5
TERMINAL OR UNIT RECORD CLASS
NO
03 A1
YES
K5
DEDICATED
NO
C2
YES
03 A1

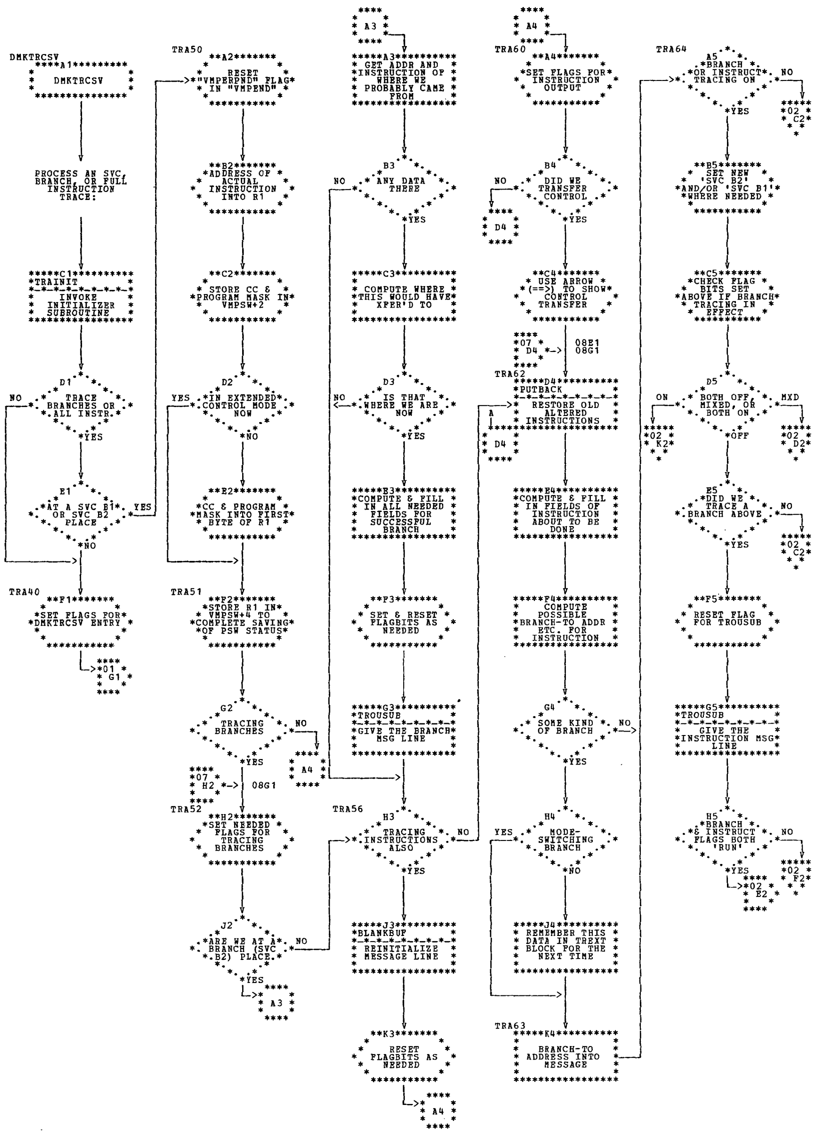

03 A1 02J5 02K5
TRA126
A1 SAVE THE DEVICE CLASS (FROM VDEVBLOK)
B1 "SPECIAL" DEVICE CLASS (CTCA)
NO
YES
E1
C1 SET REG FOR CSW INFO (LEAVING OUT REAL DEVICE)
D1 IS THE SPECIAL DEVICE DEDICATED.
NO
YES
E1
TRA126C
E1 POINT TO REAL DEVICE BLOCK
F1 DMKSCNRD CALL - FORM REAL DEVICE ADDRESS
G1 DMKCVTBH CALL - BINARY TO HEX
H1 DMKSCNRN CALL - GET REAL NAME OF DEVICE
J1 REAL DEVICE ADDRESS AND NAME INTO OUTPUT MESSAGE
TRA126B
K1 CHECK COND CODE AGAIN
>1
=0
=1
02 C2
A2
A2
A2 FOR COND CODE 1, POINT TO IOBCSW+4
B2 IS IOBLOK AVAILABLE
YES
NO
C2 POINT TO VIRTUAL CSW+4
D2 TRANBRNG GET REAL ADDR OF VIRT CSW+4/5
TRA126A
E2 DMKCVTBH CALL - VIRT CSW+4/5 TO HEX
F2 STORE 'CSW' AND VIRTUAL CSW+4/5 IN OUTPUT MSG
TRA127
G2 IF CC 0 OR 1, WAS IT SIO.
NO
YES
02 C2
H2 FOR SIO, FILL IN 'CAW' IN OUTPUT MESSAGE
TRA127A
J2 TRANBRNG GET REAL ADDRESS OF VIRTUAL CAW
TRA127B
A3 DMKCVTBH CALL - VIRTUAL CAW BINARY TO HEX
B3 STORE USER'S CAW IN OUTPUT MSG
C3 CCW TRACING WANTED ALSO
NO
YES
02 C2
D3 IOBLOK AVAILABLE
NO
YES
02 C2
E3 TROUSUB GIVE 'SIO' OUTPUT MSG
F3 SET FLAG FOR TRACING CCWS
G3 SET REGS (FROM IOBLOK) TO START WITH THE 1ST RCWTASK
03 H3 04F3
TRA129
H3 BLANKBUF REINITIALIZE OUTPUT LINE
J3 SET REGS; GO HANDLE CCW STRING IN THIS RCWTASK
03 A4 04B4
TRA133
A4 CLEAR VIRTUAL CCW (UNTIL WE GET IT)
B4 ANY VIRTUAL CCWS LEFT
NO
YES
03 B5 04B4
TRA135
B5 SET PREFIX TO INDICATE CONTINUATION OF PREVIOUS CCW
04 A2
C4 SET 'CCW' PREFIX IN OUTPUT MESSAGE
D4 GET ADDRESS OF VIRTUAL CCW
E4 DMKCVTBH CALL - BINARY TO HEX
F4 STORE ADDRESS OF VIRTUAL CCW IN OUTPUT LINE
G4 TRANBRNG CALL - GET REAL ADDR OF VIRTUAL CCW
H4 ADDRESS ERROR FOR VIRTUAL CCW
NO
YES
04 A1
04 D1
04 A1 03H4
A1 REMEMBER VALID VIRTUAL CCW
B1 DMKCVTBH CALL - VIRTUAL CCW TO HEX
C1 STORE VIRTUAL CCW IN OUTPUT LINE (TWO HALVES)
04 D1 03H4
TRA134
D1 ADVANCE TO NEXT VIRTUAL CCW FOR NEXT TIME
E1 DECREMENT COUNT OF VIRTUAL CCWS
04 A2 03B5
TRA136
A2 DMKCVTBH CALL - ADDRESS OF REAL CCW TO HEX
B2 STORE ADDRESS OF REAL CCW IN OUTPUT LINE
C2 REMEMBER REAL CCW
D2 DMKCVTBH CALL - REAL CCW TO HEX
E2 STORE REAL CCW (TWO HALVES) IN OUTPUT LINE
F2 TROUSUB OUTPUT THE FINISHED CCW LINE
G2 BLANKBUF REINITIALIZE MESSAGE LINE
H2 WAS CCW INVALID (RCWINVL SET)
YES
NO
J2 WAS IDA FLAG SET IN REAL CCW
YES
NO
A5
K2 WAS IT A DASD DEVICE
NO
YES
06 A1
04 A3 05J5 06C1 06J2
TRA138
A3 ADVANCE TO NEXT REAL CCW (IF ANY)
B3 DECREMENT COUNT - ANY REAL CCWS LEFT
YES
NO
C3 GET POINTER TO NEXT RCWTASK (IF ANY)
D3 ANYTHING THERE LEFT TO DO
NO
YES
02 D2
E3 SET OUTPUT MESSAGE TO SHOW A BREAK IN CCW STRINGS
F3 TROUSUB GIVE OUTPUT MSG; START NEXT RCWTASK
03 H3
TRA132
B4 PREVIOUS REAL CCW CP-GENERATED
NO
YES
03 A4
03 B5
A5
TRA140
A5 WAS THE REAL CCW A TIC (08)
YES
NO
B5 SET FOR NO VIRTUAL IDAL; GET ADDRESS OF REAL IDAL
C5 VIRTUAL IDA FLAG SET
NO
YES
05 D1
D5 IS VIRTUAL CCW SAME OP CODE
NO
YES
05 D1
E5 AND THE BYTE COUNT THE SAME
NO
YES
05 D1
F5 GET ADDRESS OF VIRTUAL IDAL
G5 TRANBRNG GET REAL ADDRESS OF VIRTUAL IDAL
H5 WAS VIRTUAL ADDRESS VALID.
YES
NO
05 A1
05 D1

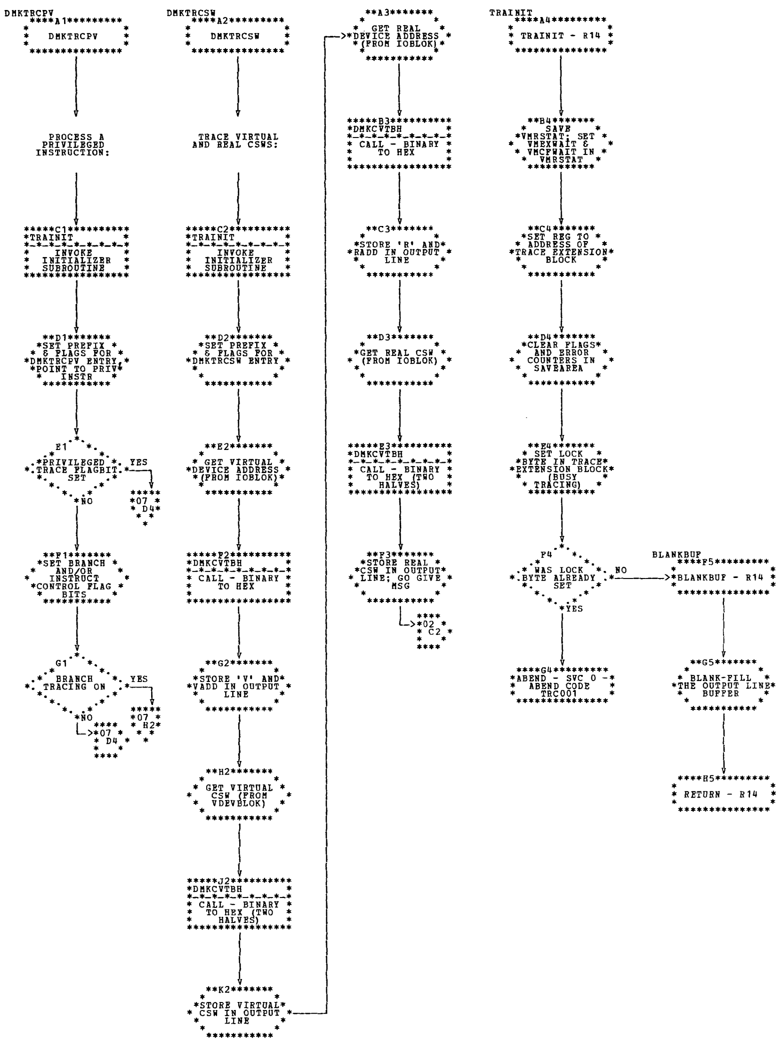
| DMKTRC -- Virtual Machine Tracing Supervisor (Parts 3 and 4 of 13)

| DMKTRC -- Virtual Machine Tracing Supervisor (Parts 5 and 6 of 13)


04H5
05 A1
A1 SET FOR VIRTUAL IDAL EXISTENT; POINT TO FIRST IDAW
B1 SET TO USE FIRST IDAW IN VIRTUAL IDAL TO GET IDAW COUNT
C1 IS THE ADDRESS VALID
YES
NO
05 D1
04C5
04D5
04E5
04H5
TRA141
D1 SET TO USE FIRST IDAW IN REAL IDAL TO GET IDAW COUNT
TRA142
E1 GET THE BYTE-COUNT (LESS 1) FROM THE REAL CCW
F1 WAS IT A READ BACKWARD OP CODE
NO
YES
TRA144
F2 COMPUTE ENDING ADDRESS FOR FWD DIRECTION CCW
G1 COMPUTE START & END ADDRESSES FOR BACKWARD READ
TRA145
H1 REMEMBER R7 = COUNT OF REAL CCWS
J1 ISOLATE 2048-BYTE BOUNDARIES FROM START & END ADDRESSES
K1 AND COMPUTE (IN R7) THE NUMBER OF IDAWS
A3
TRA146
A3 IDAL PREFIX (IF ANY) INTO OUTPUT LINE
B3 ANY VIRTUAL IDAL AT ALL
NO
YES
C3 TRANBRNG
GET REAL ADDRESS OF VIRTUAL IDAW
D3 WAS VIRTUAL ADDRESS VALID
NO
YES
E3 GET THE VIRTUAL IDAW
F3 DMKCVTBH
CALL - BINARY TO HEX
G3 STORE 'IDAL' AND THE VIRTUAL IDAW IN OUTPUT LINE
H3 ADVANCE TO NEXT VIRTUAL IDAW (IF ANY)
J3 ANY IDAWS LEFT
NO
YES
A4
A4
A4 TRANBRNG
GET REAL ADDRESS OF NEXT VIRTUAL IDAW
B4 WAS VIRTUAL ADDRESS VALID
NO
YES
C4 GET THE NEXT VIRTUAL IDAW
D4 DMKCVTBH
CALL - BINARY TO HEX
E4 STORE NEXT VIRTUAL IDAW IN THE OUTPUT LINE
TRA147
F4 'IDAL' IDENTIFIER INTO OUTPUT LINE
G4 GET FIRST REAL IDAW
H4 DMKCVTBH
CALL - BINARY TO HEX
J4 STORE FIRST REAL IDAW IN OUTPUT LINE
K4 ANY IDAWS LEFT
NO
YES
A5
A5
A5 GET NEXT REAL IDAW
B5 DMKCVTBH
CALL - BINARY TO HEX
C5 STORE NEXT REAL IDAW IN OUTPUT LINE
TRA148
D5 ADVANCE TO NEXT PAIR OF IDAWS (IF ANY)
E5 TROUSUB
GIVE OUTPUT MSG SHOWING IDAWS
F5 BLANKBUF
REINITIALIZE MESSAGE BUFFER
G5 DECREMENT NO. OF REMAINING IDAWS BY TWO
H5 ANY LEFT TO DO
YES
A3
NO
J5 RESTORE R7 (REAL CCW COUNT) & GO ADVANCE TO NEXT CCW
04 A3

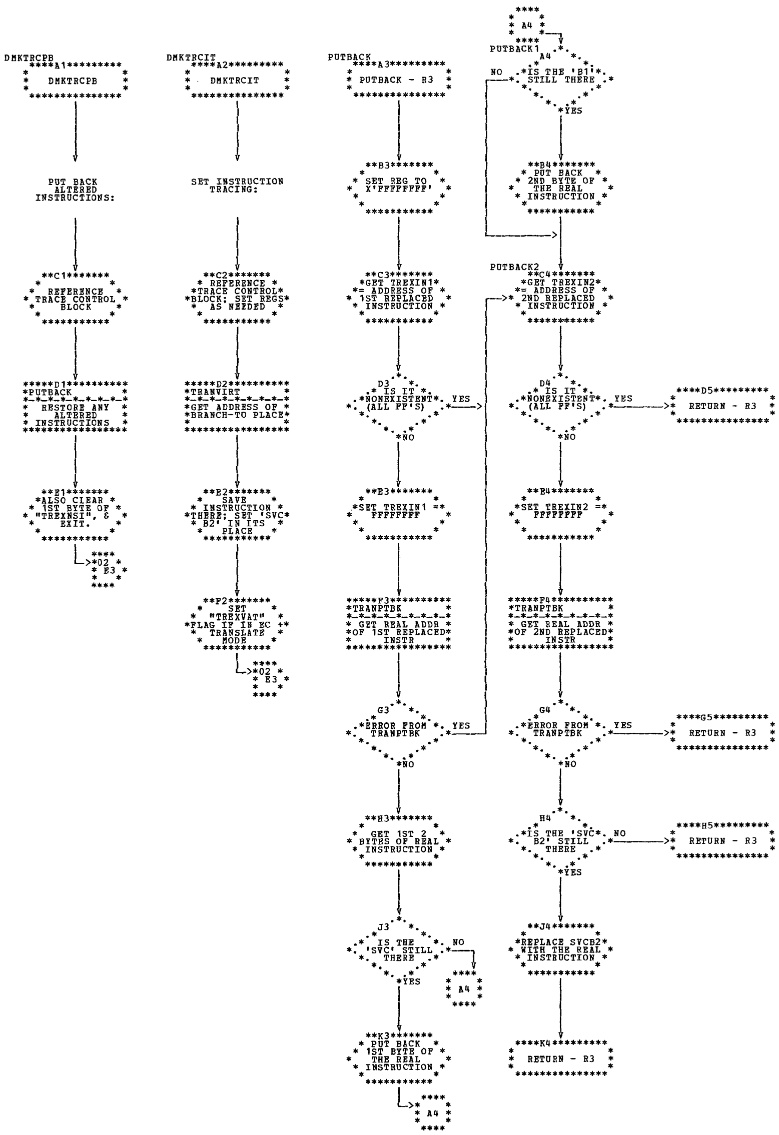

04K2
06 A1
TRA150
A1 WAS THE CCW A SEEK (07)
YES
NO
B1 OR SEEK CYLINDER (0B)
YES
NO
C1 OR SEEK HEAD (1B)
NO
04 A3
YES
TRA151
D1 SEEK PREFIX & 'SEEK' INTO OUTPUT LINE
E1 POINT TO REAL SEEK ARGUMENTS
F1 DMKCVTBH
CALL - SEEK ARGS BINARY TO HEX
G1 STORE REAL SEEK ARGUMENTS (6 BYTES) IN OUTPUT LINE
H1 IS VIRTUAL CCW SAME OP CODE
NO
YES
A2
A2
A2 SET FOR 7 BYTES; POINT TO VIRTUAL CCW & WORK AREA
B2 DMKCCWSB
CALL - GET USER'S SEEK ARGS
C2 COND CODE 0 FROM DMKCCWSB
NO
YES
D2 GET THE SEEK ARGUMENT BYTES FROM OUR WORK AREA
E2 DMKCVTBH
CALL - SEEK ARGS BINARY TO HEX
F2 STORE VIRTUAL SEEK DATA (7 BYTES) IN OUTPUT LINE
G2 AND INSERT THE 'SEEK' IDENTIFIER
TRA159
H2 TROUSUB
GIVE OUTPUT MSG SHOWING SEEK INFO
J2 BLANKBUF
REINITIALIZE MESSAGE BUFFER AND DO NEXT CCW
04 A3
DMKTRCIO
A3 DMKTRCIO
PROCESS AN I/O INTERRUPT:
C3 TRAINIT
INVOKE INITIALIZER SUBROUTINE
D3 SET FLAGS FOR DMKTRCIO ENTRY
01 E1
DMKTRCPG
A4 DMKTRCPG
PROCESS A PROGRAM INTERRUPT:
C4 TRAINIT
INVOKE INITIALIZER SUBROUTINE
D4 SET FLAGS FOR DMKTRCPG ENTRY
E4 VIRTUAL "PER" PENDING
NO
01 E1
YES
F4 ADD X'0080' TO INTERRUPT CODE
01 E1
DMKTRCSI
A5 DMKTRCSI
PROCESS AN I/O OPERATION (SIO, SIOF, TIO, HIO, HDV, TCH):
C5 TRAINIT
INVOKE INITIALIZER SUBROUTINE
D5 GET SIO OR OTHER OP CODE FROM VMINST
E5 DMKNEMOP
CALL - GET MNEMONIC, STORE IN OUTPUT LINE
F5 SET PREFIX TO I/O; SET FLAGS FOR DMKTRCSI ENTRY
G5 GET ADDRESS OF INSTRUCTION FROM VMPSW+4
01 H1

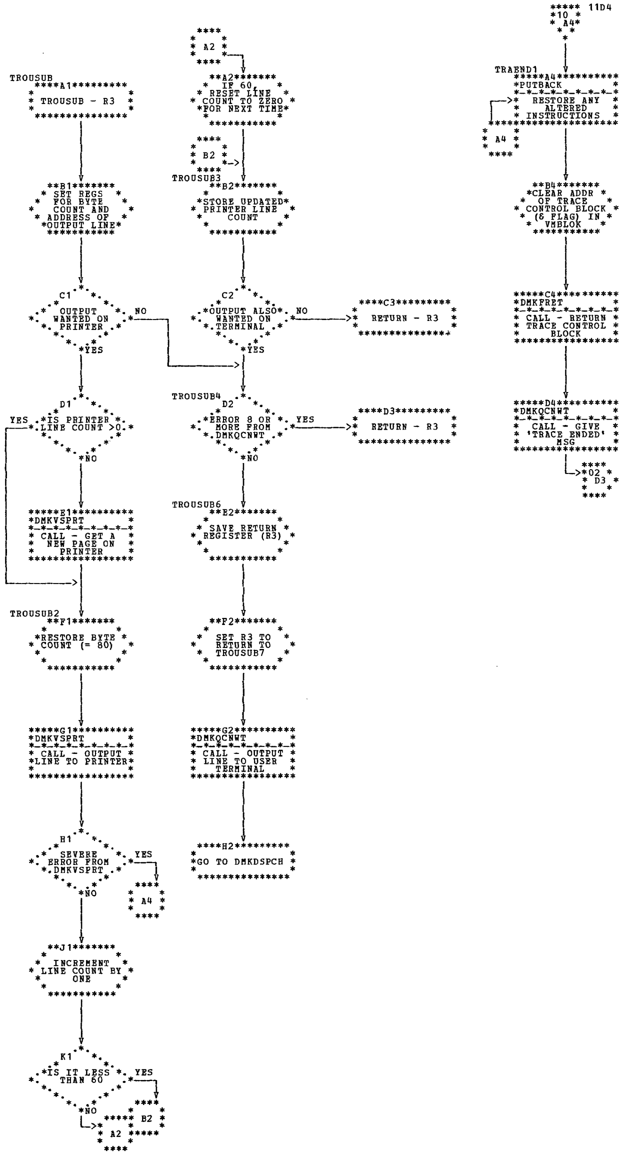

DMKTRCSV
A1
DMKTRCSV
PROCESS AN SVC, BRANCH, OR FULL INSTRUCTION TRACE:
C1
TRAINIT
INVOKE INITIALIZER SUBROUTINE
D1
TRACE BRANCHES OR ALL INSTR.
NO
YES
E1
AT A SVC B1 OR SVC B2 PLACE
YES
NO
TRA40
F1
SET FLAGS FOR DMKTRCSV ENTRY
01 G1
TRA50
A2
RESET "VMPERPND" FLAG IN "VMPEND"
B2
ADDRESS OF ACTUAL INSTRUCTION INTO R1
C2
STORE CC & PROGRAM MASK IN VMPSW+2
D2
IN EXTENDED CONTROL MODE NOW
YES
NO
E2
CC & PROGRAM MASK INTO FIRST BYTE OF R1
TRA51
F2
STORE R1 IN VMPSW+4 TO COMPLETE SAVING OF PSW STATUS
G2
TRACING BRANCHES
NO
YES
A4
07 H2
08G1
TRA52
H2
SET NEEDED FLAGS FOR TRACING BRANCHES
J2
ARE WE AT A BRANCH (SVC B2) PLACE.
NO
YES
A3
A3
A3
GET ADDR AND INSTRUCTION OF WHERE WE PROBABLY CAME FROM
B3
ANY DATA THERE
NO
YES
C3
COMPUTE WHERE THIS WOULD HAVE XFER'D TO
D3
IS THAT WHERE WE ARE NOW
NO
YES
E3
COMPUTE & FILL IN ALL NEEDED FIELDS FOR SUCCESSFUL BRANCH
F3
SET & RESET FLAGBITS AS NEEDED
G3
TROUSUB
GIVE THE BRANCH MSG LINE
TRA56
H3
TRACING INSTRUCTIONS ALSO
NO
YES
J3
BLANKBUF
REINITIALIZE MESSAGE LINE
K3
RESET FLAGBITS AS NEEDED
A4
A4
TRA60
A4
SET FLAGS FOR INSTRUCTION OUTPUT
B4
DID WE TRANSFER CONTROL
NO
D4
YES
C4
USE ARROW (==>) TO SHOW CONTROL TRANSFER
07 D4
08E1
08G1
TRA62
D4
PUTBACK
RESTORE OLD ALTERED INSTRUCTIONS
A
D4
E4
COMPUTE & FILL IN FIELDS OF INSTRUCTION ABOUT TO BE DONE
F4
COMPUTE POSSIBLE BRANCH-TO ADDR ETC. FOR INSTRUCTION
G4
SOME KIND OF BRANCH
NO
YES
H4
MODE-SWITCHING BRANCH
YES
NO
J4
REMEMBER THIS DATA IN TREXT BLOCK FOR THE NEXT TIME
TRA63
K4
BRANCH-TO ADDRESS INTO MESSAGE
TRA64
A5
BRANCH OR INSTRUCT TRACING ON
NO
02 C2
YES
B5
SET NEW 'SVC B2' AND/OR 'SVC B1' WHERE NEEDED
C5
CHECK FLAG BITS SET ABOVE IF BRANCH TRACING IN EFFECT
D5
BOTH OFF, MIXED, OR BOTH ON
ON
MXD
02 K2
OFF
02 D2
E5
DID WE TRACE A BRANCH ABOVE
NO
02 C2
YES
F5
RESET FLAG FOR TROUSUB
G5
TROUSUB
GIVE THE INSTRUCTION MSG LINE
H5
BRANCH & INSTRUCT FLAGS BOTH 'RUN'
NO
YES
02 E2
02 F2

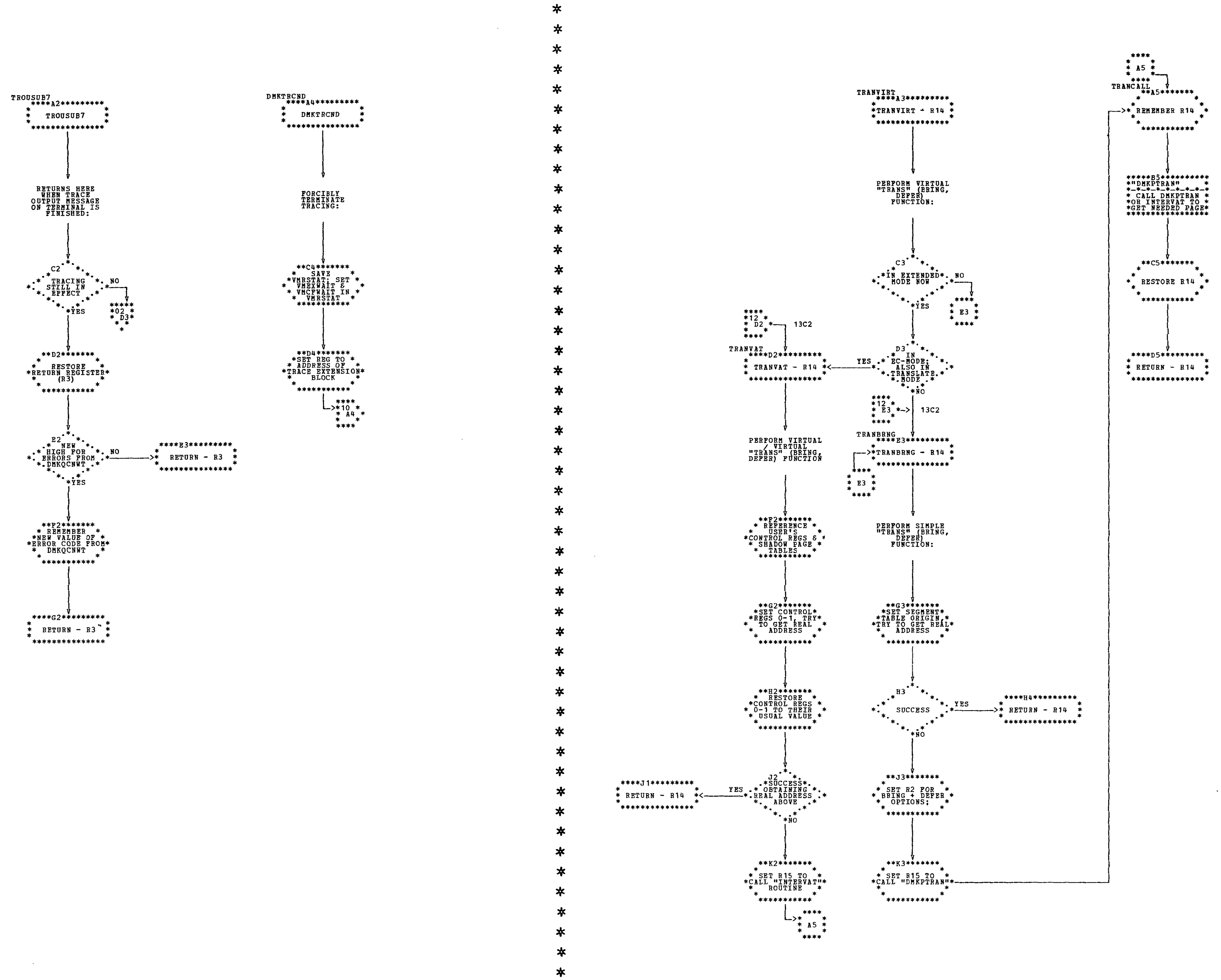

DMKTRCPV
A1
DMKTRCPV
PROCESS A PRIVILEGED INSTRUCTION:
C1
TRAINIT
INVOKE INITIALIZER SUBROUTINE
D1
SET PREFIX & FLAGS FOR DMKTRCPV ENTRY, POINT TO PRIV INSTR
E1
PRIVILEGED TRACE FLAGBIT SET
YES
07 D4
NO
F1
SET BRANCH AND/OR INSTRUCT CONTROL FLAG BITS
G1
BRANCH TRACING ON
YES
07 H2
NO
07 D4
DMKTRCSW
A2
DMKTRCSW
TRACE VIRTUAL AND REAL CSWS:
C2
TRAINIT
INVOKE INITIALIZER SUBROUTINE
D2
SET PREFIX & FLAGS FOR DMKTRCSW ENTRY
E2
GET VIRTUAL DEVICE ADDRESS (FROM IOBLOK)
F2
DMKCVTBH
CALL - BINARY TO HEX
G2
STORE 'V' AND VADD IN OUTPUT LINE
H2
GET VIRTUAL CSW (FROM VDEVBLOK)
J2
DMKCVTBH
CALL - BINARY TO HEX (TWO HALVES)
K2
STORE VIRTUAL CSW IN OUTPUT LINE
A3
GET REAL DEVICE ADDRESS (FROM IOBLOK)
B3
DMKCVTBH
CALL - BINARY TO HEX
C3
STORE 'R' AND RADD IN OUTPUT LINE
D3
GET REAL CSW (FROM IOBLOK)
E3
DMKCVTBH
CALL - BINARY TO HEX (TWO HALVES)
F3
STORE REAL CSW IN OUTPUT LINE; GO GIVE MSG
02 C2
TRAINIT
A4
TRAINIT - R14
B4
SAVE VMRSTAT; SET VMEXWAIT & VMCFWAIT IN VMRSTAT
C4
SET REG TO ADDRESS OF TRACE EXTENSION BLOCK
D4
CLEAR FLAGS AND ERROR COUNTERS IN SAVEAREA
E4
SET LOCK BYTE IN TRACE EXTENSION BLOCK (BUSY TRACING)
F4
WAS LOCK BYTE ALREADY SET
NO
YES
G4
ABEND - SVC 0 - ABEND CODE TRC001
BLANKBUF
F5
BLANKBUF - R14
G5
BLANK-FILL THE OUTPUT LINE BUFFER
H5
RETURN - R14

| DMKTRC -- Virtual Machine Tracing Supervisor (Parts 7 and 8 of 13)

| DMKTRC -- Virtual Machine Tracing Supervisor (Parts 9 and 10 of 13)

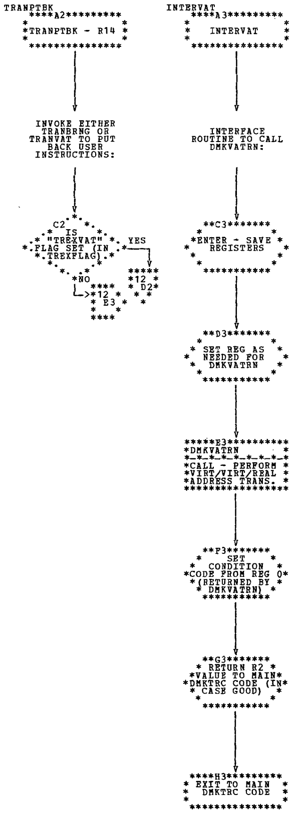

DMKTRCPB
A1
DMKTRCPB
PUT BACK ALTERED INSTRUCTIONS:
C1
REFERENCE TRACE CONTROL BLOCK
D1
PUTBACK
RESTORE ANY ALTERED INSTRUCTIONS
E1
ALSO CLEAR 1ST BYTE OF "TREXNSI", & EXIT.
02 E3
DMKTRCIT
A2
DMKTRCIT
SET INSTRUCTION TRACING:
C2
REFERENCE TRACE CONTROL BLOCK; SET REGS AS NEEDED
D2
TRANVIRT
GET ADDRESS OF BRANCH-TO PLACE
E2
SAVE INSTRUCTION THERE; SET 'SVC B2' IN ITS PLACE
F2
SET "TREXVAT" FLAG IF IN EC TRANSLATE MODE
02 E3
PUTBACK
A3
PUTBACK - R3
B3
SET REG TO X'FFFFFFFF'
C3
GET TREXIN1 = ADDRESS OF 1ST REPLACED INSTRUCTION
D3
IS IT NONEXISTENT (ALL FF'S)
YES
NO
E3
SET TREXIN1 = FFFFFFFF
F3
TRANPTBK
GET REAL ADDR OF 1ST REPLACED INSTR
G3
ERROR FROM TRANPTBK
YES
NO
H3
GET 1ST 2 BYTES OF REAL INSTRUCTION
J3
IS THE 'SVC' STILL THERE
NO
A4
YES
K3
PUT BACK 1ST BYTE OF THE REAL INSTRUCTION
A4
A4
PUTBACK1
A4
IS THE 'B1' STILL THERE
NO
YES
B4
PUT BACK 2ND BYTE OF THE REAL INSTRUCTION
PUTBACK2
C4
GET TREXIN2 = ADDRESS OF 2ND REPLACED INSTRUCTION
D4
IS IT NONEXISTENT (ALL FF'S)
YES
NO
D5
RETURN - R3
E4
SET TREXIN2 = FFFFFFFF
F4
TRANPTBK
GET REAL ADDR OF 2ND REPLACED INSTR
G4
ERROR FROM TRANPTBK
YES
NO
G5
RETURN - R3
H4
IS THE 'SVC B2' STILL THERE
NO
YES
H5
RETURN - R3
J4
REPLACE SVCB2 WITH THE REAL INSTRUCTION
K4
RETURN - R3

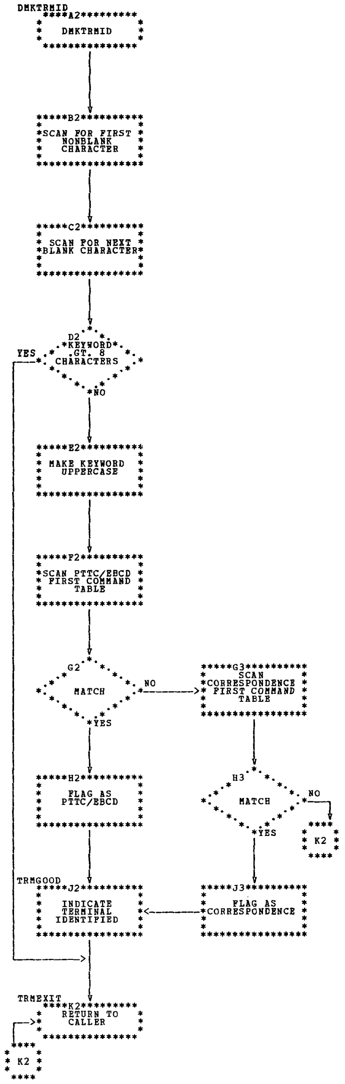

TROUSUB
A1
TROUSUB - R3
B1
SET REGS FOR BYTE COUNT AND ADDRESS OF OUTPUT LINE
C1
OUTPUT WANTED ON PRINTER
NO
YES
D1
IS PRINTER LINE COUNT >0
YES
NO
E1
DMKVSPRT
CALL - GET A NEW PAGE ON PRINTER
TROUSUB2
F1
RESTORE BYTE COUNT (= 80)
G1
DMKVSPRT
CALL - OUTPUT LINE TO PRINTER
H1
SEVERE ERROR FROM DMKVSPRT
YES
A4
NO
J1
INCREMENT LINE COUNT BY ONE
K1
IS IT LESS THAN 60
YES
B2
NO
A2
A2
A2
IF 60, RESET LINE COUNT TO ZERO FOR NEXT TIME
B2
TROUSUB3
B2
STORE UPDATED PRINTER LINE COUNT
C2
OUTPUT ALSO WANTED ON TERMINAL
NO
YES
C3
RETURN - R3
TROUSUB4
D2
ERROR 8 OR MORE FROM DMKQCNWT
YES
NO
D3
RETURN - R3
TROUSUB6
E2
SAVE RETURN REGISTER (R3)
F2
SET R3 TO RETURN TO TROUSUB7
G2
DMKQCNWT
CALL - OUTPUT LINE TO USER TERMINAL
H2
GO TO DMKDSPCH
11D4
10 A4
TRAEND1
A4
PUTBACK
RESTORE ANY ALTERED INSTRUCTIONS
A4
B4
CLEAR ADDR OF TRACE CONTROL BLOCK (& FLAG) IN VMBLOK
C4
DMKFRET
CALL - RETURN TRACE CONTROL BLOCK
D4
DMKQCNWT
CALL - GIVE 'TRACE ENDED' MSG
02 D3


TROUSUB7
A2
TROUSUB7
RETURNS HERE WHEN TRACE OUTPUT MESSAGE ON TERMINAL IS FINISHED:
C2
TRACING STILL IN EFFECT
NO
02 D3
YES
D2
RESTORE RETURN REGISTER (R3)
E2
NEW HIGH FOR ERRORS FROM DMKQCNWT
NO
E3
RETURN - R3
YES
F2
REMEMBER NEW VALUE OF ERROR CODE FROM DMKQCNWT
G2
RETURN - R3
DMKTRCND
A4
DMKTRCND
FORCIBLY TERMINATE TRACING:
C4
SAVE VMRSTAT; SET VMEXWAIT & VMCFWAIT IN VMRSTAT
D4
SET REG TO ADDRESS OF TRACE EXTENSION BLOCK
10 A4
TRANVIRT
A3
TRANVIRT - R14
PERFORM VIRTUAL "TRANS" (BRING, DEFER) FUNCTION:
C3
IN EXTENDED MODE NOW
NO
E3
YES
12 D2
13C2
TRANVAT
D2
TRANVAT - R14
YES
D3
IN EC-MODE; ALSO IN TRANSLATE MODE
NO
12 E3
13C2
TRANBRNG
E3
TRANBRNG - R14
E3
PERFORM VIRTUAL / VIRTUAL "TRANS" (BRING, DEFER) FUNCTION
F2
REFERENCE USER'S CONTROL REGS & SHADOW PAGE TABLES
G2
SET CONTROL REGS 0-1, TRY TO GET REAL ADDRESS
H2
RESTORE CONTROL REGS 0-1 TO THEIR USUAL VALUE
J1
RETURN - R14
YES
J2
SUCCESS OBTAINING REAL ADDRESS ABOVE
NO
K2
SET R15 TO CALL "INTERVAT" ROUTINE
A5
PERFORM SIMPLE "TRANS" (BRING, DEFER) FUNCTION:
G3
SET SEGMENT TABLE ORIGIN, TRY TO GET REAL ADDRESS
H3
SUCCESS
YES
H4
RETURN - R14
NO
J3
SET R2 FOR BRING + DEFER OPTIONS;
K3
SET R15 TO CALL "DMKPTRAN"
A5
TRANCALL
A5
REMEMBER R14
B5
"DMKPTRAN"
CALL DMKPTRAN OR INTERVAT TO GET NEEDED PAGE
C5
RESTORE R14
D5
RETURN - R14

| DMKTRC -- Virtual Machine Tracing Supervisor (Parts 11 and 12 of 13)

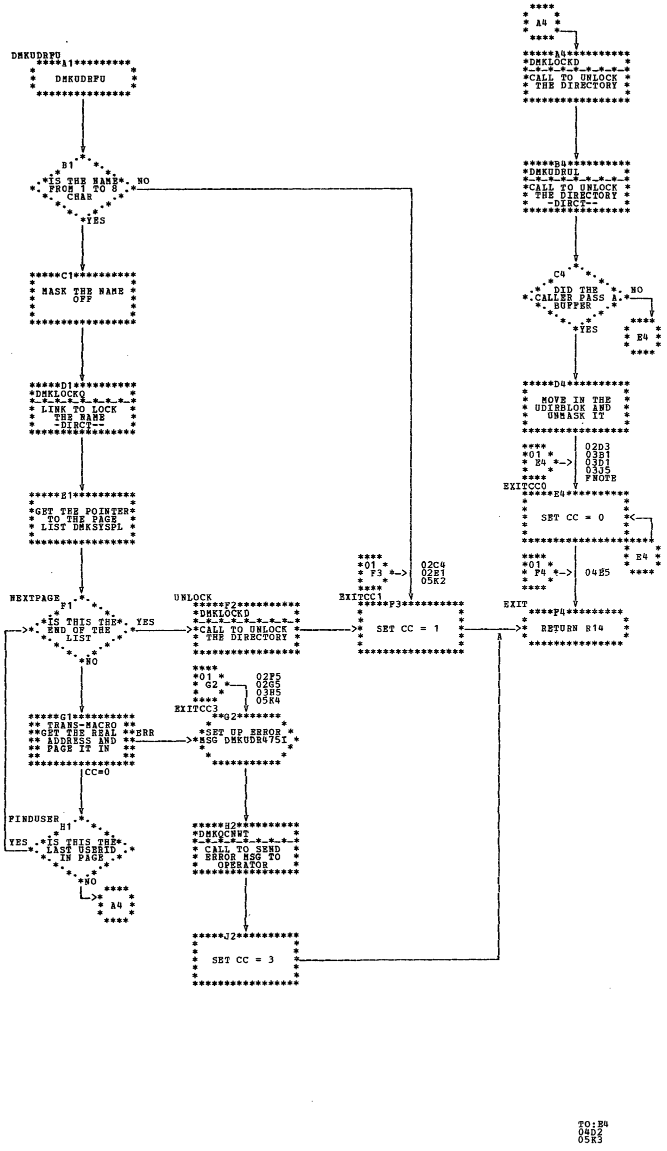
| DMKTRC -- Virtual Machine Tracing Supervisor (Part 13 of 13)

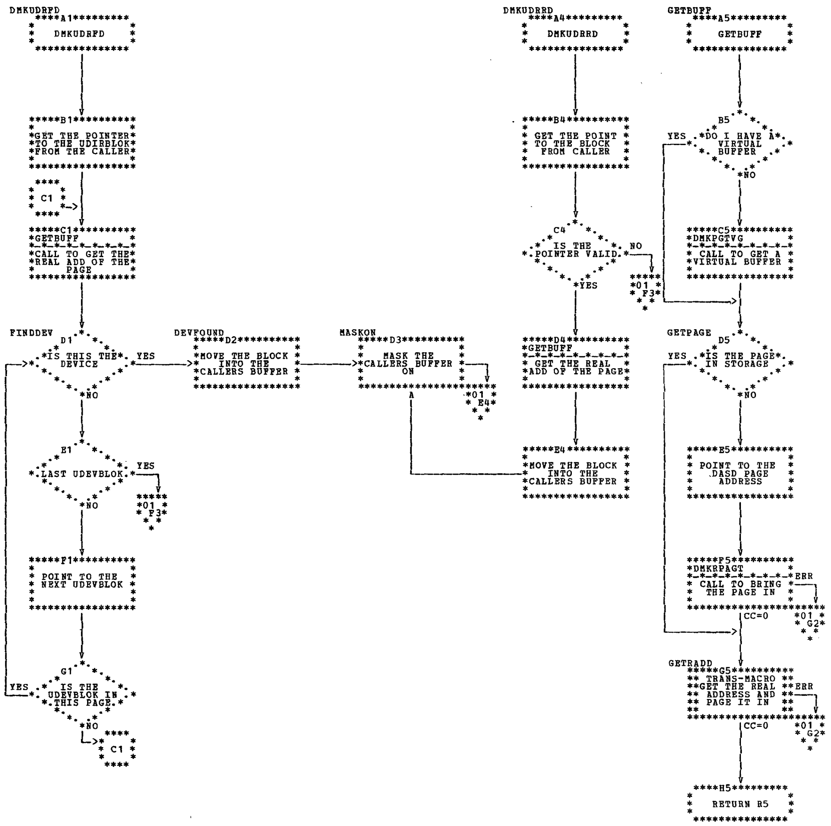

TRANPTBK
A2
TRANPTBK - R14
INVOKE EITHER TRANBRNG OR TRANVAT TO PUT BACK USER INSTRUCTIONS:
C2
IS "TREXVAT" FLAG SET (IN TREXFLAG)
YES
NO
12 D2
12 E3
INTERVAT
A3
INTERVAT
INTERFACE ROUTINE TO CALL DMKVATRN:
C3
ENTER - SAVE REGISTERS
D3
SET REG AS NEEDED FOR DMKVATRN
E3
DMKVATRN
CALL - PERFORM VIRT/VIRT/REAL ADDRESS TRANS.
F3
SET CONDITION CODE FROM REG 0 (RETURNED BY DMKVATRN)
G3
RETURN R2 VALUE TO MAIN DMKTRC CODE (IN CASE GOOD)
H3
EXIT TO MAIN DMKTRC CODE

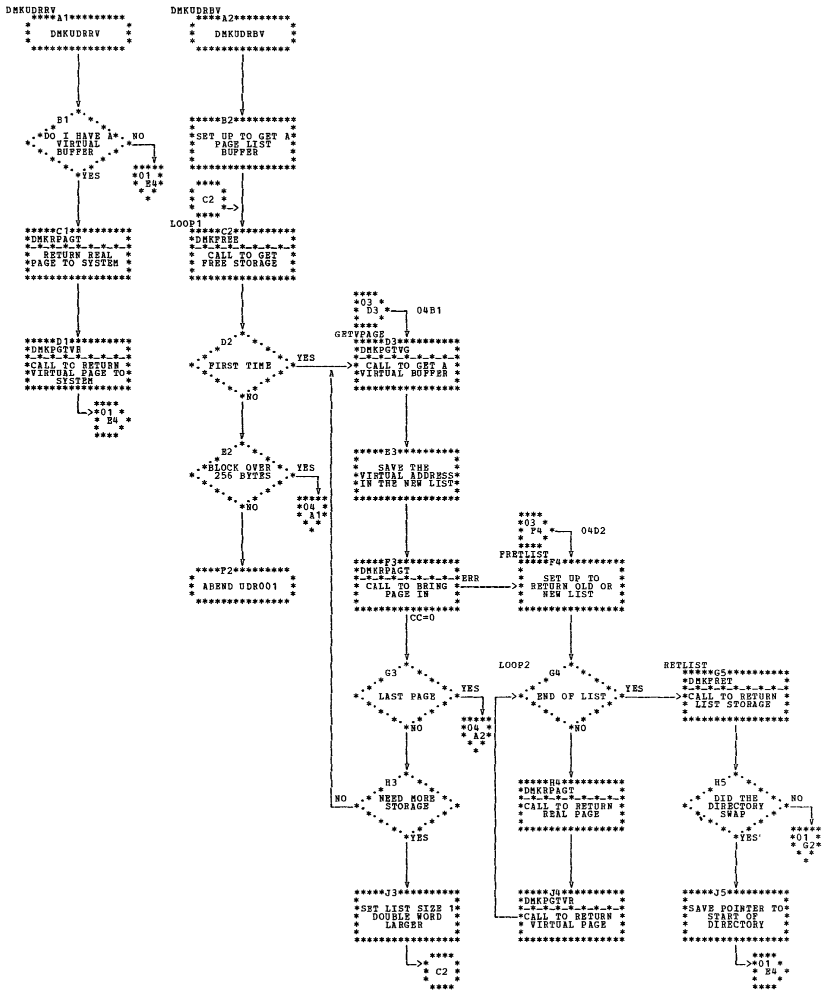

DMKTRMID
A2
DMKTRMID
B2
SCAN FOR FIRST NONBLANK CHARACTER
C2
SCAN FOR NEXT BLANK CHARACTER
D2
KEYWORD .GT. 8 CHARACTERS
YES
NO
E2
MAKE KEYWORD UPPERCASE
F2
SCAN PTTC/EBCD FIRST COMMAND TABLE
G2
MATCH
NO
YES
G3
SCAN CORRESPONDENCE FIRST COMMAND TABLE
H2
FLAG AS PTTC/EBCD
H3
MATCH
NO
YES
K2
TRMGOOD
J2
INDICATE TERMINAL IDENTIFIED
J3
FLAG AS CORRESPONDENCE
TRMEXIT
K2
RETURN TO CALLER
K2

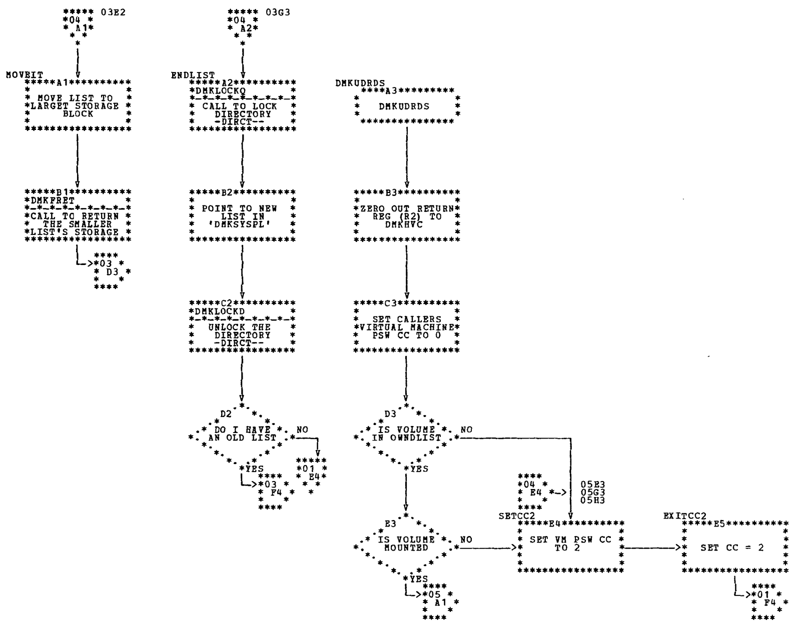
DMKTRM -- Identify Terminal (Part 1 of 1)

| DMKUDR -- User Directory Manager (Parts 1 and 2 of 5)


DMKUDRFU
A1
DMKUDRFU
B1
IS THE NAME FROM 1 TO 8 CHAR
NO
YES
C1
MASK THE NAME OFF
D1
DMKLOCKQ
LINK TO LOCK THE NAME -DIRCT--
E1
GET THE POINTER TO THE PAGE LIST DMKSYSPL
NEXTPAGE
F1
IS THIS THE END OF THE LIST
YES
NO
G1
TRANS-MACRO GET THE REAL ADDRESS AND PAGE IT IN
ERR
CC=0
FINDUSER
H1
IS THIS THE LAST USERID IN PAGE
YES
NO
A4
UNLOCK
F2
DMKLOCKD
CALL TO UNLOCK THE DIRECTORY
01 G2
02F5
02G5
03H5
05K4
EXITCC3
G2
SET UP ERROR MSG DMKUDR475I
H2
DMKQCNWT
CALL TO SEND ERROR MSG TO OPERATOR
J2
SET CC = 3
01 F3
02C4
02E1
05K2
EXITCC1
F3
SET CC = 1
A
A4
A4
DMKLOCKD
CALL TO UNLOCK THE DIRECTORY
B4
DMKUDRUL
CALL TO UNLOCK THE DIRECTORY -DIRCT--
C4
DID THE CALLER PASS A BUFFER
NO
YES
E4
D4
MOVE IN THE UDIRBLOK AND UNMASK IT
01 E4
02D3
03B1
03D1
03J5
FNOTE
EXITCC0
E4
SET CC = 0
E4
01 F4
04E5
EXIT
F4
RETURN R14
TO:E4
04D2
05K3

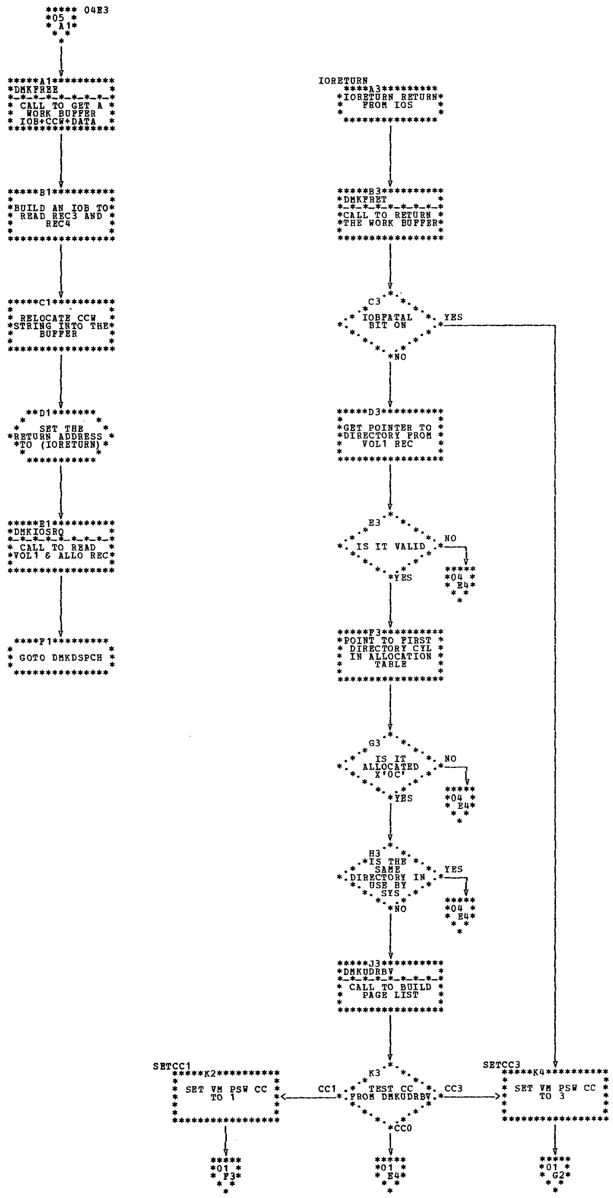

DMKUDRFD
A1
DMKUDRFD
B1
GET THE POINTER TO THE UDIRBLOK FROM THE CALLER
C1
C1
GETBUFF
CALL TO GET THE REAL ADD OF THE PAGE
FINDDEV
D1
IS THIS THE DEVICE
YES
NO
E1
LAST UDEVBLOK
YES
01 F3
NO
F1
POINT TO THE NEXT UDEVBLOK
G1
IS THE UDEVBLOK IN THIS PAGE
YES
NO
C1
DEVFOUND
D2
MOVE THE BLOCK INTO THE CALLERS BUFFER
MASKON
D3
MASK THE CALLERS BUFFER ON
A
01 E4
DMKUDRRD
A4
DMKUDRRD
B4
GET THE POINT TO THE BLOCK FROM CALLER
C4
IS THE POINTER VALID
NO
01 F3
YES
D4
GETBUFF
GET THE REAL ADD OF THE PAGE
E4
MOVE THE BLOCK INTO THE CALLERS BUFFER
GETBUFF
A5
GETBUFF
B5
DO I HAVE A VIRTUAL BUFFER
YES
NO
C5
DMKPGTVG
CALL TO GET A VIRTUAL BUFFER
GETPAGE
D5
IS THE PAGE IN STORAGE
YES
NO
E5
POINT TO THE DASD PAGE ADDRESS
F5
DMKRPAGT
CALL TO BRING THE PAGE IN
ERR
CC=0
01 G2
GETRADD
G5
TRANS-MACRO GET THE REAL ADDRESS AND PAGE IT IN
ERR
CC=0
01 G2
H5
RETURN R5

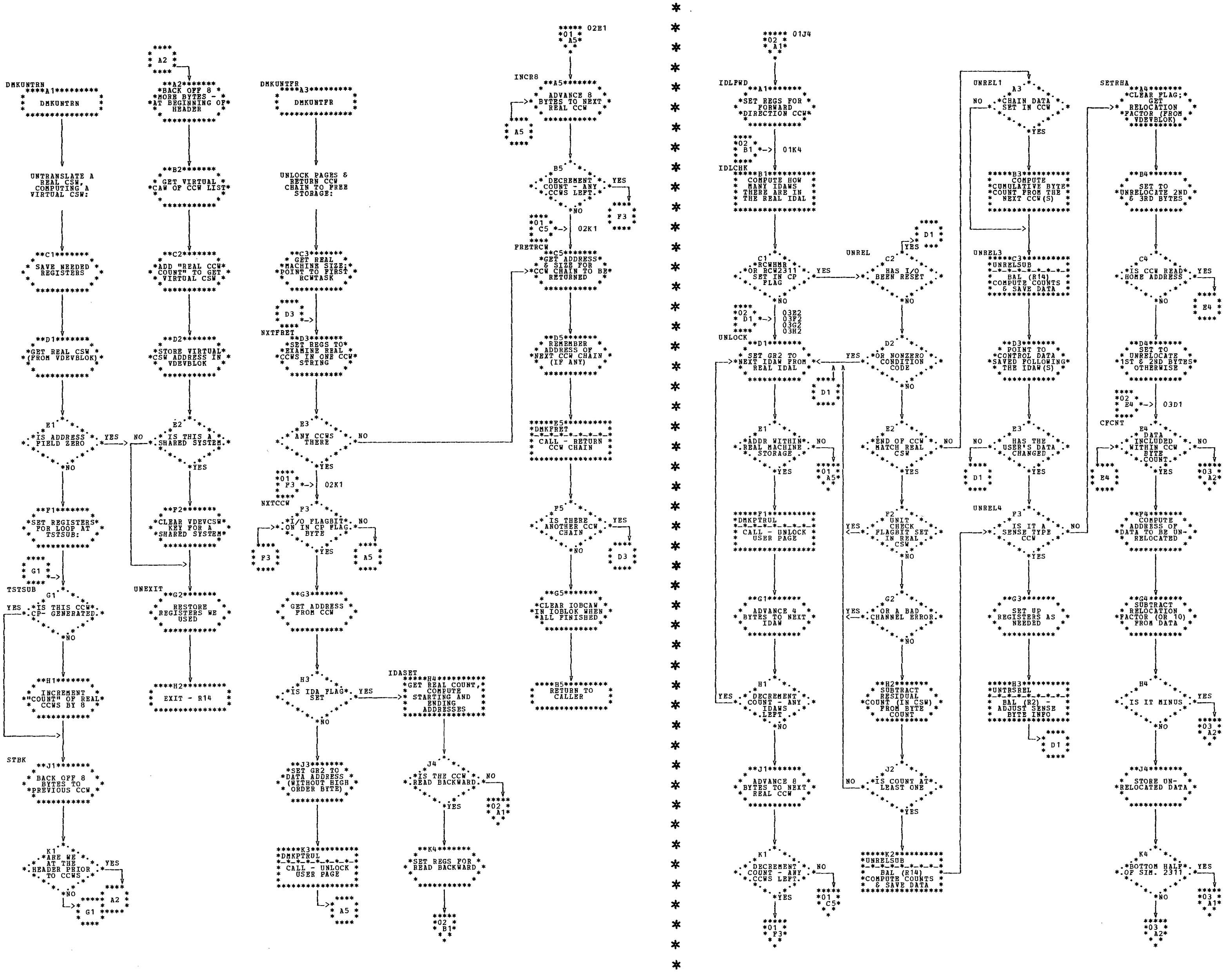




| DMKUDR -- User Directory Manager (Parts 3 and 4 of 5)

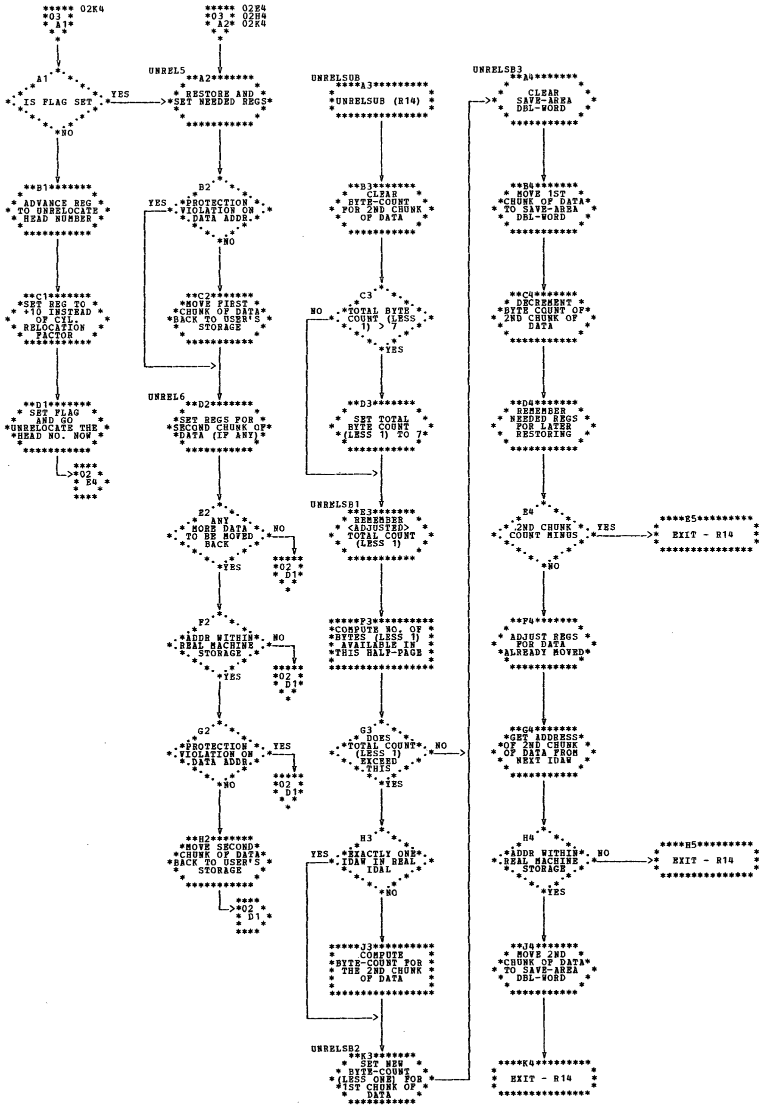
| DMKUDR -- User Directory Manager (Part 5 of 5)

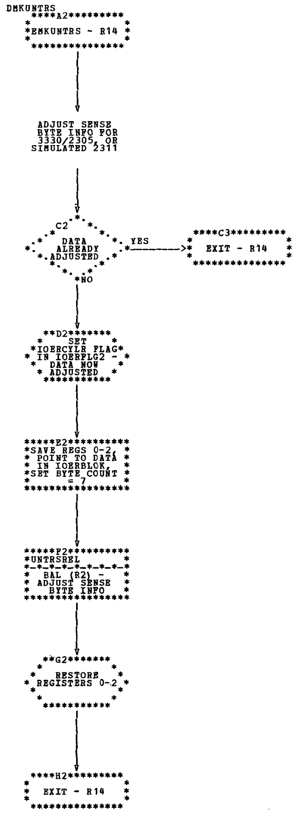

04E3
05 A1
A1 DMKFREE
CALL TO GET A WORK BUFFER IOB+CCW+DATA
B1
BUILD AN IOB TO READ REC3 AND REC4
C1
RELOCATE CCW STRING INTO THE BUFFER
D1
SET THE RETURN ADDRESS TO (IORETURN)
E1 DMKIOSRQ
CALL TO READ VOL1 & ALLO REC
F1
GOTO DMKDSPCH
IORETURN
A3
IORETURN RETURN FROM IOS
B3 DMKFRET
CALL TO RETURN THE WORK BUFFER
C3
IOBFATAL BIT ON
YES
NO
D3
GET POINTER TO DIRECTORY FROM VOL1 REC
E3
IS IT VALID
NO
04 E4
YES
F3
POINT TO FIRST DIRECTORY CYL IN ALLOCATION TABLE
G3
IS IT ALLOCATED X'0C'
NO
04 E4
YES
H3
IS THE SAME DIRECTORY IN USE BY SYS
YES
04 E4
NO
J3 DMKUDRBV
CALL TO BUILD PAGE LIST
SETCC1
K2
SET VM PSW CC TO 1
01 F3
CC1
K3
TEST CC FROM DMKUDRBV
CC3
CC0
01 E4
SETCC3
K4
SET VM PSW CC TO 3
01 G2

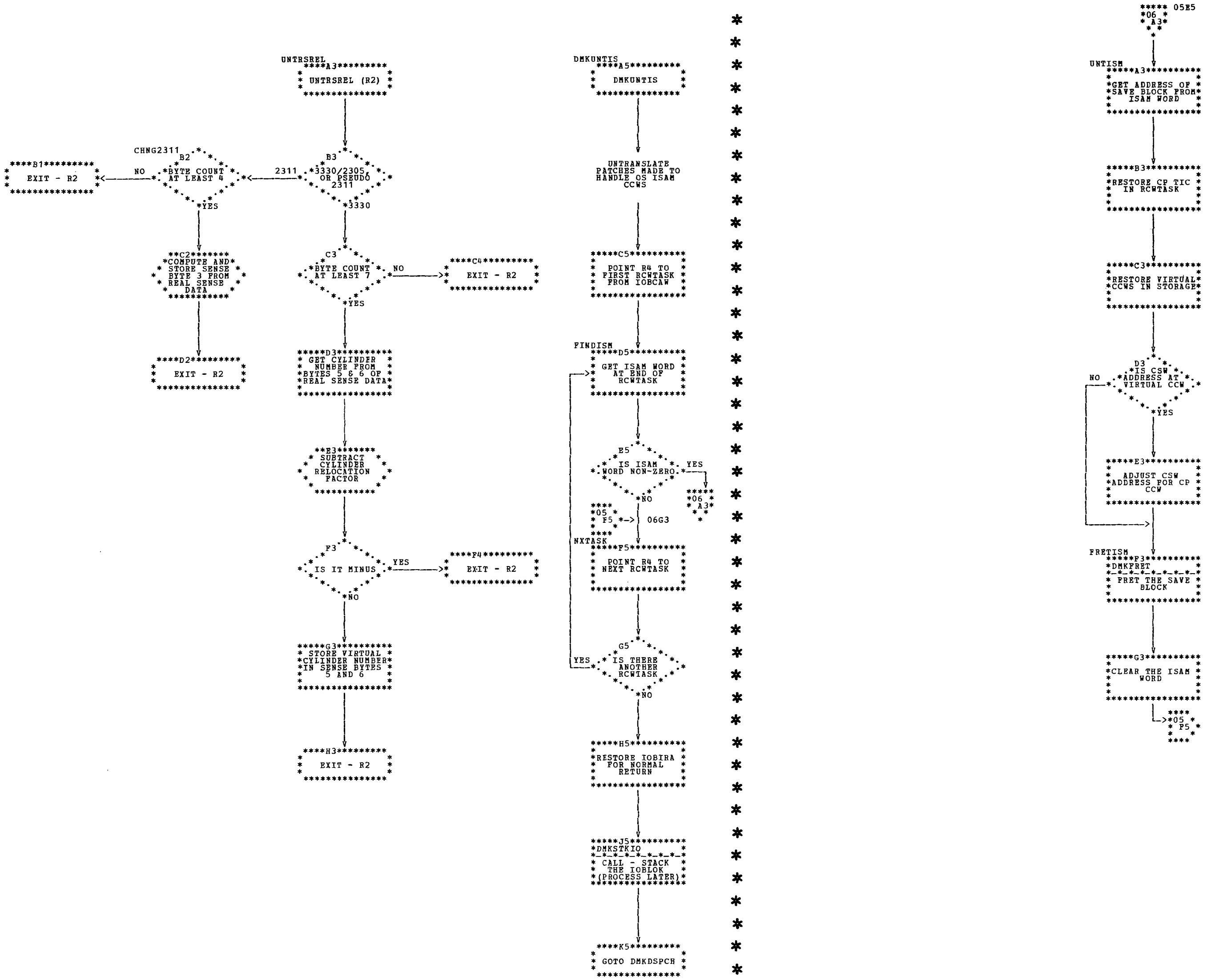

DMKUNTRN
A1 DMKUNTRN
UNTRANSLATE A REAL CSW, COMPUTING A VIRTUAL CSW:
C1 SAVE NEEDED REGISTERS
D1 GET REAL CSW (FROM VDEVBLOK)
E1 IS ADDRESS FIELD ZERO
YES
NO
F1 SET REGISTERS FOR LOOP AT TSTSUB:
G1
TSTSUB
G1 IS THIS CCW CP- GENERATED
H1 INCREMENT "COUNT" OF REAL CCWS BY 8
STBK
J1 BACK OFF 8 BYTES TO PREVIOUS CCW
K1 ARE WE AT THE HEADER PRIOR TO CCWS
A2
A2 BACK OFF 8 MORE BYTES - AT BEGINNING OF HEADER
B2 GET VIRTUAL CAW OF CCW LIST
C2 ADD "REAL CCW COUNT" TO GET VIRTUAL CSW
D2 STORE VIRTUAL CSW ADDRESS IN VDEVBLOK
E2 IS THIS A SHARED SYSTEM
F2 CLEAR VDEVCSW KEY FOR A SHARED SYSTEM
UNEXIT
G2 RESTORE REGISTERS WE USED
H2 EXIT - R14
DMKUNTFR
A3 DMKUNTFR
UNLOCK PAGES & RETURN CCW CHAIN TO FREE STORAGE:
C3 GET REAL MACHINE SIZE; POINT TO FIRST RCWTASK
D3
NXTFRET
D3 SET REGS TO EXAMINE REAL CCWS IN ONE CCW STRING
E3 ANY CCWS THERE
01 F3 02K1
NXTCCW
F3 I/O FLAGBIT ON IN CP FLAG BYTE
F3
A5
G3 GET ADDRESS FROM CCW
H3 IS IDA FLAG SET
J3 SET GR2 TO DATA ADDRESS (WITHOUT HIGH ORDER BYTE)
K3 DMKPTRUL CALL - UNLOCK USER PAGE
A5
IDASET
H4 GET REAL COUNT, COMPUTE STARTING AND ENDING ADDRESSES
J4 IS THE CCW READ BACKWARD
02 A1
K4 SET REGS FOR READ BACKWARD
02 B1
01 A5 02E1
INCR8
A5 ADVANCE 8 BYTES TO NEXT REAL CCW
A5
B5 DECREMENT COUNT - ANY CCWS LEFT
F3
01 C5 02K1
FRETRCW
C5 GET ADDRESS & SIZE FOR CCW CHAIN TO BE RETURNED
D5 REMEMBER ADDRESS OF NEXT CCW CHAIN (IF ANY)
E5 DMKFRET CALL - RETURN CCW CHAIN
F5 IS THERE ANOTHER CCW CHAIN
D3
G5 CLEAR IOBCAW IN IOBLOK WHEN ALL FINISHED
H5 RETURN TO CALLER
02 A1 01J4
IDLFWD
A1 SET REGS FOR FORWARD DIRECTION CCW
02 B1 01K4
IDLCHK
B1 COMPUTE HOW MANY IDAWS THERE ARE IN THE REAL IDAL
C1 RCWHMR OR RCW2311 SET IN CP FLAG
02 D1 03E2 03F2 03G2 03H2
UNLOCK
D1 SET GR2 TO NEXT IDAW FROM REAL IDAL
D1
E1 ADDR WITHIN REAL MACHINE STORAGE
01 A5
F1 DMKPTRUL CALL - UNLOCK USER PAGE
G1 ADVANCE 4 BYTES TO NEXT IDAW
H1 DECREMENT COUNT - ANY IDAWS LEFT
J1 ADVANCE 8 BYTES TO NEXT REAL CCW
K1 DECREMENT COUNT - ANY CCWS LEFT
01 C5
01 F3
D1
UNREL
C2 HAS I/O BEEN RESET
D2 OR NONZERO CONDITION CODE
E2 END OF CCW MATCH REAL CSW
F2 UNIT CHECK FLAGBIT SET IN REAL CSW
G2 OR A BAD CHANNEL ERROR
H2 SUBTRACT RESIDUAL COUNT (IN CSW) FROM BYTE COUNT
J2 IS COUNT AT LEAST ONE
K2 UNRELSUB BAL (R14) COMPUTE COUNTS & SAVE DATA
UNREL1
A3 CHAIN DATA SET IN CCW
B3 COMPUTE CUMULATIVE BYTE COUNT FROM THE NEXT CCW(S)
UNREL3
C3 UNRELSUB BAL (R14) COMPUTE COUNTS & SAVE DATA
D3 POINT TO CONTROL DATA SAVED FOLLOWING THE IDAW(S)
E3 HAS THE USER'S DATA CHANGED
D1
UNREL4
F3 IS IT A SENSE TYPE CCW
G3 SET UP REGISTERS AS NEEDED
H3 UNTRSREL BAL (R2) - ADJUST SENSE BYTE INFO
D1
SETRHA
A4 CLEAR FLAG; GET RELOCATION FACTOR (FROM VDEVBLOK)
B4 SET TO UNRELOCATE 2ND & 3RD BYTES
C4 IS CCW READ HOME ADDRESS
E4
D4 SET TO UNRELOCATE 1ST & 2ND BYTES OTHERWISE
02 E4 03D1
CFCNT
E4 DATA INCLUDED WITHIN CCW BYTE COUNT
E4
03 A2
F4 COMPUTE ADDRESS OF DATA TO BE UN- RELOCATED
G4 SUBTRACT RELOCATION FACTOR (OR 10) FROM DATA
H4 IS IT MINUS
03 A2
J4 STORE UN- RELOCATED DATA
K4 BOTTOM HALF OF SIM. 2311
03 A1
03 A2

DMKUNT -- Virtual I/O Subroutines (Parts 1 and 2 of 6)

DMKUNT -- Virtual I/O Subrotuines (Parts 3 and 4 of 6)

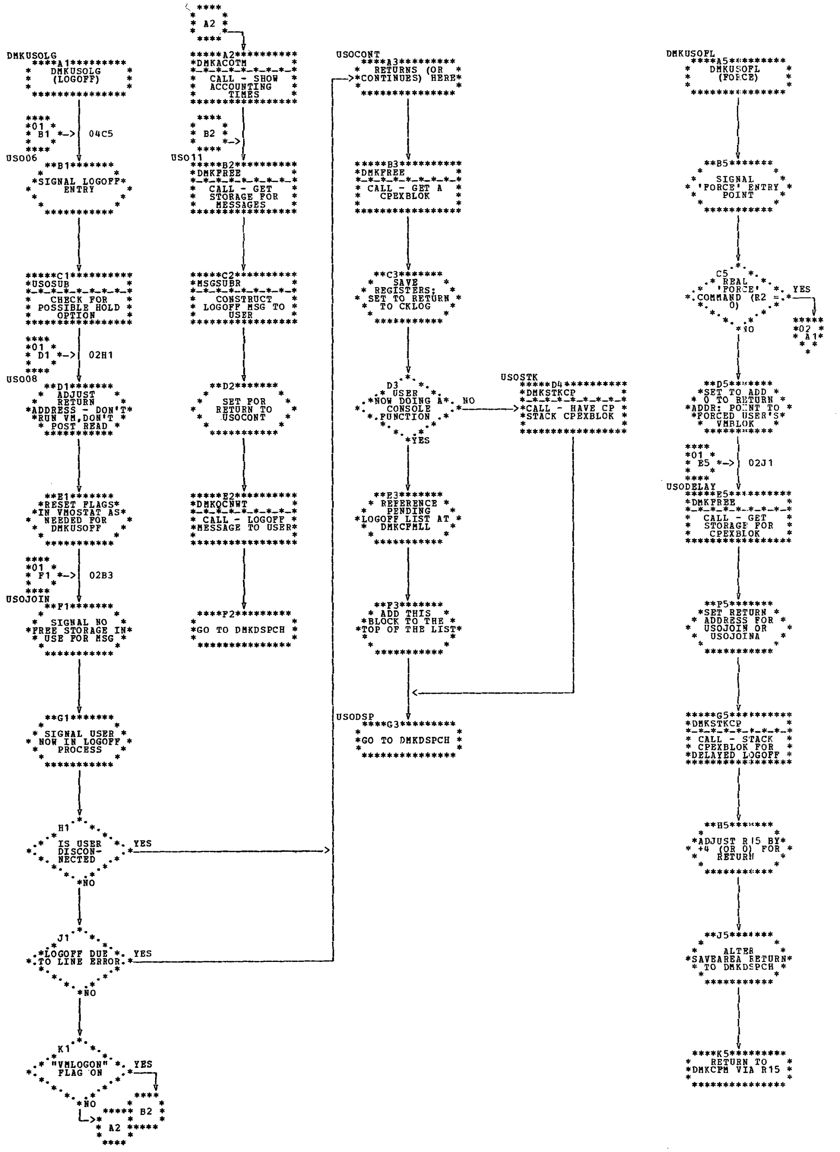

03 A1 02K4
A1 IS FLAG SET
YES
NO
B1 ADVANCE REG TO UNRELOCATE HEAD NUMBER
C1 SET REG TO +10 INSTEAD OF CYL. RELOCATION FACTOR
D1 SET FLAG AND GO UNRELOCATE THE HEAD NO. NOW
02 E4
03 A2 02E4 02H4 02K4
UNREL5
A2 RESTORE AND SET NEEDED REGS
B2 PROTECTION VIOLATION ON DATA ADDR.
YES
NO
C2 MOVE FIRST CHUNK OF DATA BACK TO USER'S STORAGE
UNREL6
D2 SET REGS FOR SECOND CHUNK OF DATA (IF ANY)
E2 ANY MORE DATA TO BE MOVED BACK
NO
02 D1
YES
F2 ADDR WITHIN REAL MACHINE STORAGE
NO
02 D1
YES
G2 PROTECTION VIOLATION ON DATA ADDR.
YES
02 D1
NO
H2 MOVE SECOND CHUNK OF DATA BACK TO USER'S STORAGE
02 D1
UNRELSUB
A3 UNRELSUB (R14)
B3 CLEAR BYTE-COUNT FOR 2ND CHUNK OF DATA
C3 TOTAL BYTE COUNT (LESS 1) > 7
NO
YES
D3 SET TOTAL BYTE COUNT (LESS 1) TO 7
UNRELSB1
E3 REMEMBER <ADJUSTED> TOTAL COUNT (LESS 1)
F3 COMPUTE NO. OF BYTES (LESS 1) AVAILABLE IN THIS HALF-PAGE
G3 DOES TOTAL COUNT (LESS 1) EXCEED THIS
NO
YES
H3 EXACTLY ONE IDAW IN REAL IDAL
YES
NO
J3 COMPUTE BYTE-COUNT FOR THE 2ND CHUNK OF DATA
UNRELSB2
K3 SET NEW BYTE-COUNT (LESS ONE) FOR 1ST CHUNK OF DATA
UNRELSB3
A4 CLEAR SAVE-AREA DBL-WORD
B4 MOVE 1ST CHUNK OF DATA TO SAVE-AREA DBL-WORD
C4 DECREMENT BYTE COUNT OF 2ND CHUNK OF DATA
D4 REMEMBER NEEDED REGS FOR LATER RESTORING
E4 2ND CHUNK COUNT MINUS
YES
E5 EXIT - R14
NO
F4 ADJUST REGS FOR DATA ALREADY MOVED
G4 GET ADDRESS OF 2ND CHUNK OF DATA FROM NEXT IDAW
H4 ADDR WITHIN REAL MACHINE STORAGE
NO
H5 EXIT - R14
YES
J4 MOVE 2ND CHUNK OF DATA TO SAVE-AREA DBL-WORD
K4 EXIT - R14

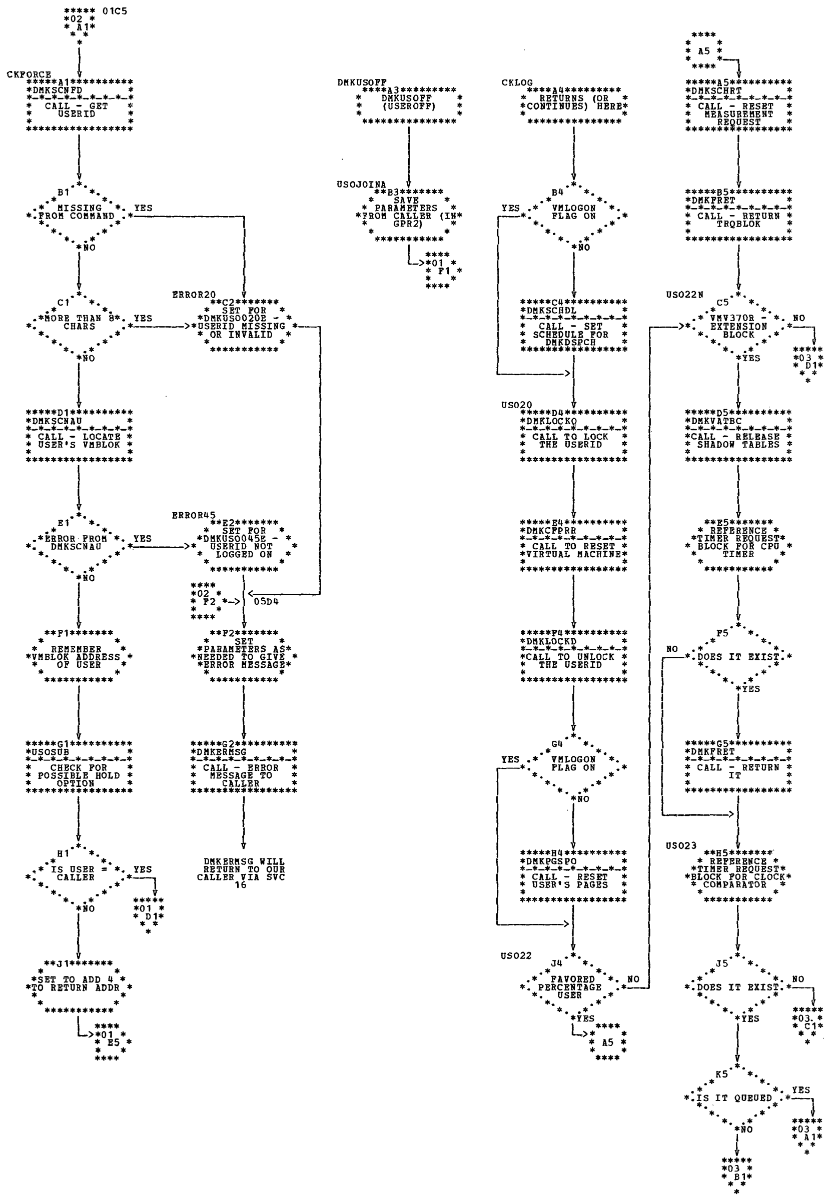

DMKUNTRS
A2 EMKUNTRS - R14
ADJUST SENSE BYTE INFO FOR 3330/2305, OR SIMULATED 2311
C2 DATA ALREADY ADJUSTED
YES
C3 EXIT - R14
NO
D2 SET IOERCYLR FLAG IN IOERFLG2 - DATA NOW ADJUSTED
E2 SAVE REGS 0-2, POINT TO DATA IN IOERBLOK, SET BYTE COUNT = 7
F2 UNTRSREL
BAL (R2) - ADJUST SENSE BYTE INFO
G2 RESTORE REGISTERS 0-2
H2 EXIT - R14

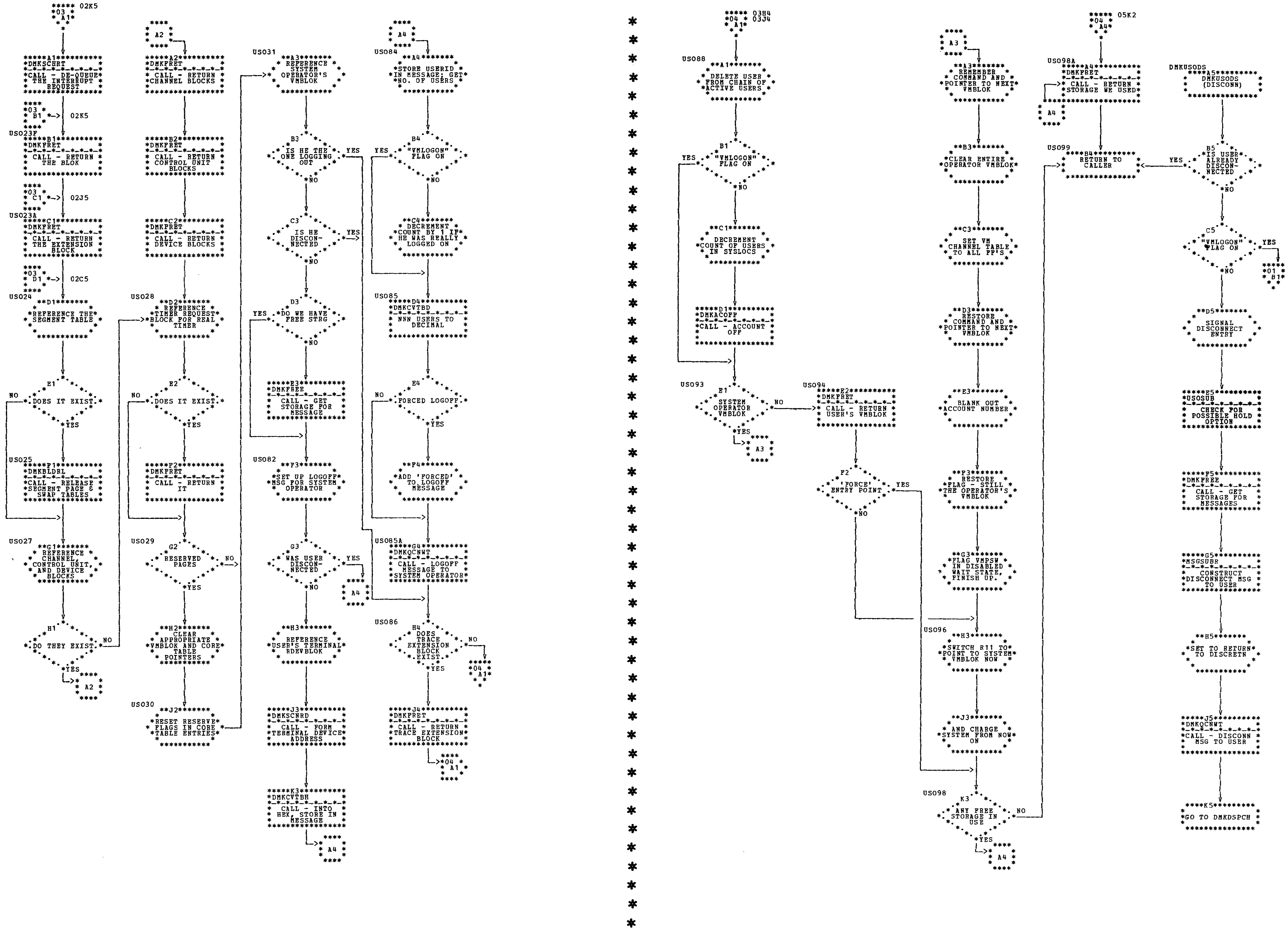

UNTRSREL
A3
UNTRSREL (R2)
CHNG2311
B2
BYTE COUNT AT LEAST 4
NO
YES
B1
EXIT - R2
B3
3330/2305, OR PSEUDO 2311
2311
3330
C2
COMPUTE AND STORE SENSE BYTE 3 FROM REAL SENSE DATA
D2
EXIT - R2
C3
BYTE COUNT AT LEAST 7
NO
YES
C4
EXIT - R2
D3
GET CYLINDER NUMBER FROM BYTES 5 & 6 OF REAL SENSE DATA
E3
SUBTRACT CYLINDER RELOCATION FACTOR
F3
IS IT MINUS
YES
NO
F4
EXIT - R2
G3
STORE VIRTUAL CYLINDER NUMBER IN SENSE BYTES 5 AND 6
H3
EXIT - R2
DMKUNTIS
A5
DMKUNTIS
UNTRANSLATE PATCHES MADE TO HANDLE OS ISAM CCWS
C5
POINT R4 TO FIRST RCWTASK FROM IOBCAW
FINDISM
D5
GET ISAM WORD AT END OF RCWTASK
E5
IS ISAM WORD NON-ZERO
YES
NO
06 A3
05 F5
06G3
NXTASK
F5
POINT R4 TO NEXT RCWTASK
G5
IS THERE ANOTHER RCWTASK
YES
NO
H5
RESTORE IOBIRA FOR NORMAL RETURN
J5
DMKSTKIO
CALL - STACK THE IOBLOK (PROCESS LATER)
K5
GOTO DMKDSPCH
06 A3
05E5
UNTISM
A3
GET ADDRESS OF SAVE BLOCK FROM ISAM WORD
B3
RESTORE CP TIC IN RCWTASK
C3
RESTORE VIRTUAL CCWS IN STORAGE
D3
IS CSW ADDRESS AT VIRTUAL CCW
NO
YES
E3
ADJUST CSW ADDRESS FOR CP CCW
FRETISM
F3
DMKFRET
FRET THE SAVE BLOCK
G3
CLEAR THE ISAM WORD
05 F5

DMKUNT -- Virtual I/O Subroutines (Parts 5 and 6 of 6)

| DMKUSO -- Process DISCON,FORCE, and LOGOFF Commands; Logoff Routine (Parts 1 and 2 of 6)

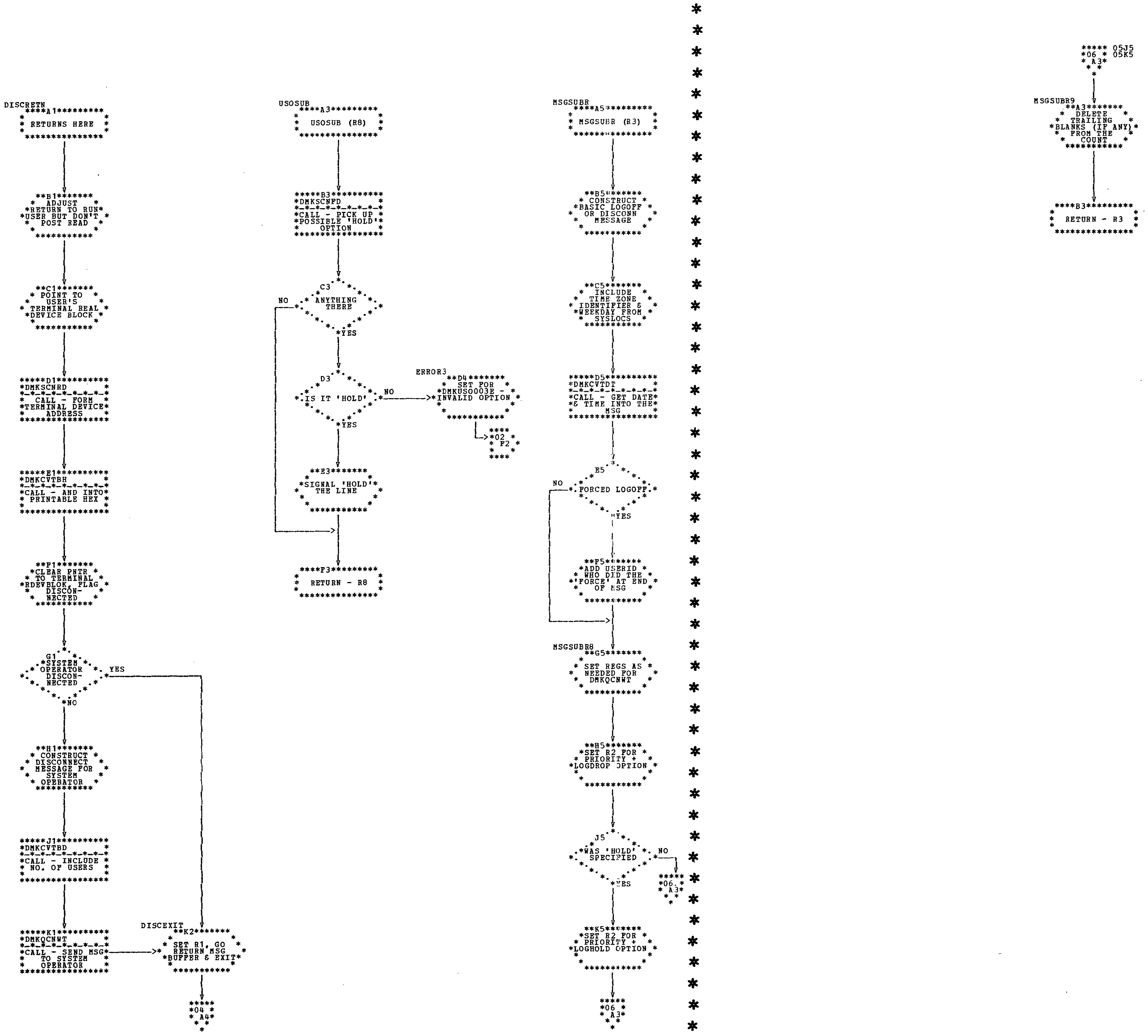

DMKUSOLG
A1 DMKUSOLG (LOGOFF)
01 B1 04C5
USO06
B1 SIGNAL LOGOFF ENTRY
C1 USOSUB CHECK FOR POSSIBLE HOLD OPTION
01 D1 02H1
USO08
D1 ADJUST RETURN ADDRESS - DON'T RUN VM,DON'T POST READ
E1 RESET FLAGS IN VMOSTAT AS NEEDED FOR DMKUSOFF
01 F1 02B3
USOJOIN
F1 SIGNAL NO FREE STORAGE IN USE FOR MSG
G1 SIGNAL USER NOW IN LOGOFF PROCESS
H1 IS USER DISCON-NECTED
YES
NO
J1 LOGOFF DUE TO LINE ERROR
YES
NO
K1 "VMLOGON" FLAG ON
YES
NO
B2
A2
A2
A2 DMKACOTM CALL - SHOW ACCOUNTING TIMES
B2
USO11
B2 DMKFREE CALL - GET STORAGE FOR MESSAGES
C2 MSGSUBR CONSTRUCT LOGOFF MSG TO USER
D2 SET FOR RETURN TO USOCONT
E2 DMKQCNWT CALL - LOGOFF MESSAGE TO USER
F2 GO TO DMKDSPCH
USOCONT
A3 RETURNS (OR CONTINUES) HERE
B3 DMKFREE CALL - GET A CPEXBLOK
C3 SAVE REGISTERS; SET TO RETURN TO CKLOG
D3 USER NOW DOING A CONSOLE FUNCTION
NO
YES
USOSTK
D4 DMKSTKCP CALL - HAVE CP STACK CPEXBLOK
E3 REFERENCE PENDING LOGOFF LIST AT DMKCFMLL
F3 ADD THIS BLOCK TO THE TOP OF THE LIST
USODSP
G3 GO TO DMKDSPCH
DMKUSOFL
A5 DMKUSOFL (FORCE)
B5 SIGNAL 'FORCE' ENTRY POINT
C5 REAL 'FORCE' COMMAND (R2 = 0)
YES
NO
02 A1
D5 SET TO ADD 0 TO RETURN ADDR: POINT TO FORCED USER'S VMBLOK
01 E5 02J1
USODELAY
E5 DMKFREE CALL - GET STORAGE FOR CPEXBLOK
F5 SET RETURN ADDRESS FOR USOJOIN OR USOJOINA
G5 DMKSTKCP CALL - STACK CPEXBLOK FOR DELAYED LOGOFF
H5 ADJUST R15 BY +4 (OR 0) FOR RETURN
J5 ALTER SAVEAREA RETURN TO DMKDSPCH
K5 RETURN TO DMKCFM VIA R15

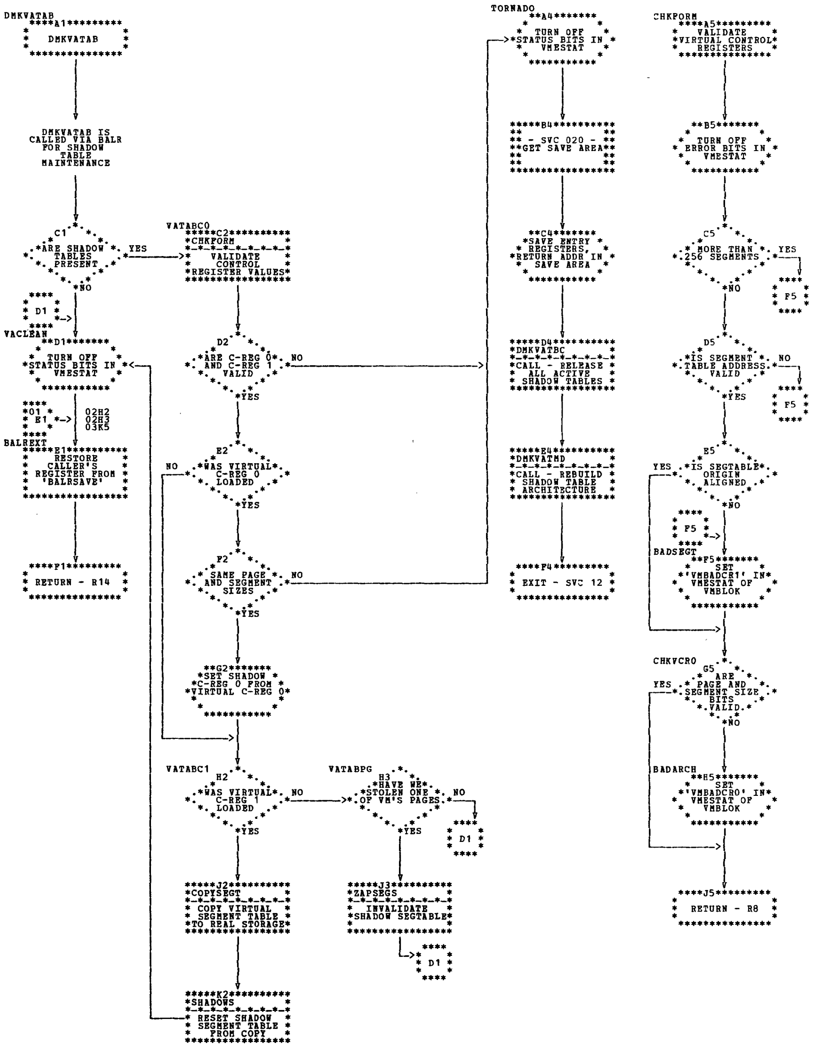

02 A1 01C5
CKFORCE
A1 DMKSCNFD CALL - GET USERID
B1 MISSING FROM COMMAND
YES
NO
C1 MORE THAN 8 CHARS
YES
NO
ERROR20
C2 SET FOR DMKUSO020E - USERID MISSING OR INVALID
D1 DMKSCNAU CALL - LOCATE USER'S VMBLOK
E1 ERROR FROM DMKSCNAU
YES
NO
ERROR45
E2 SET FOR DMKUSO045E - USERID NOT LOGGED ON
02 F2 05D4
F1 REMEMBER VMBLOK ADDRESS OF USER
F2 SET PARAMETERS AS NEEDED TO GIVE ERROR MESSAGE
G1 USOSUB CHECK FOR POSSIBLE HOLD OPTION
G2 DMKERMSG CALL - ERROR MESSAGE TO CALLER
DMKERMSG WILL RETURN TO OUR CALLER VIA SVC 16
H1 IS USER = CALLER
YES
NO
01 D1
J1 SET TO ADD 4 TO RETURN ADDR
01 E5
DMKUSOFF
A3 DMKUSOFF (USEROFF)
USOJOINA
B3 SAVE PARAMETERS FROM CALLER (IN GPR2)
01 F1
CKLOG
A4 RETURNS (OR CONTINUES) HERE
B4 VMLOGON FLAG ON
YES
NO
C4 DMKSCHDL CALL - SET SCHEDULE FOR DMKDSPCH
USO20
D4 DMKLOCKQ CALL TO LOCK THE USERID
E4 DMKCFPRR CALL TO RESET VIRTUAL MACHINE
F4 DMKLOCKD CALL TO UNLOCK THE USERID
G4 VMLOGON FLAG ON
YES
NO
H4 DMKPGSPO CALL - RESET USER'S PAGES
USO22
J4 FAVORED PERCENTAGE USER
NO
YES
A5
A5
A5 DMKSCHRT CALL - RESET MEASUREMENT REQUEST
B5 DMKFRET CALL - RETURN TRQBLOK
USO22N
C5 VMV370R - EXTENSION BLOCK
NO
YES
03 D1
D5 DMKVATBC CALL - RELEASE SHADOW TABLES
E5 REFERENCE TIMER REQUEST BLOCK FOR CPU TIMER
F5 DOES IT EXIST
NO
YES
G5 DMKFRET CALL - RETURN IT
USO23
H5 REFERENCE TIMER REQUEST BLOCK FOR CLOCK COMPARATOR
J5 DOES IT EXIST
NO
YES
03 C1
K5 IS IT QUEUED
YES
NO
03 A1
03 B1

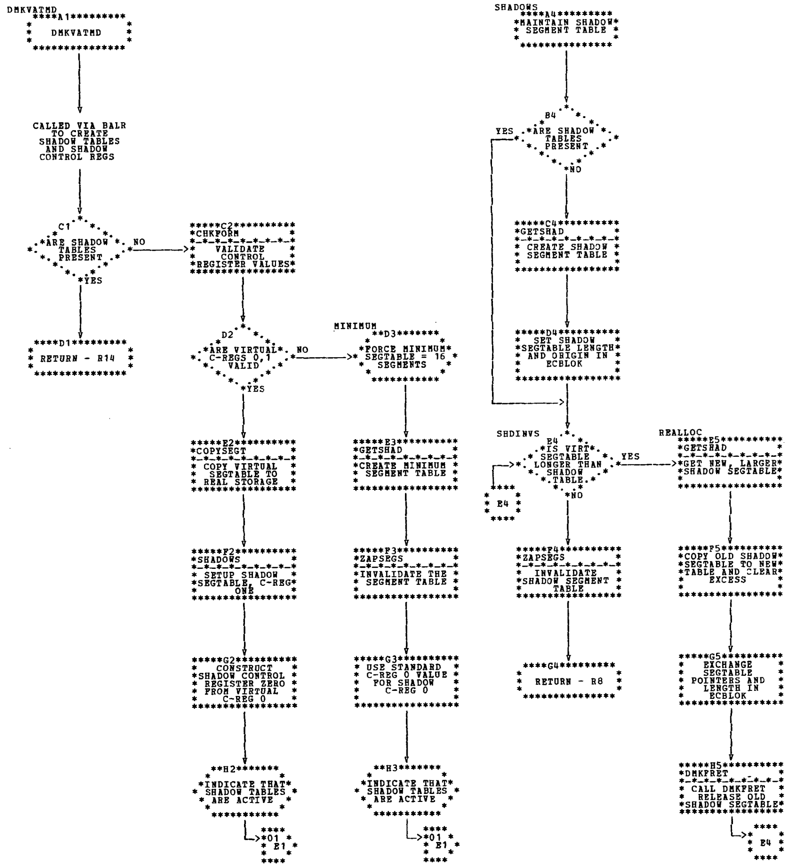

03 A1 02K5
A1 DMKSCHRT CALL - DE-QUEUE THE INTERRUPT REQUEST
03 B1 02K5
USO23F
B1 DMKFRET CALL - RETURN THE BLOK
03 C1 02J5
USO23A
C1 DMKFRET CALL - RETURN THE EXTENSION BLOCK
03 D1 02C5
USO24
D1 REFERENCE THE SEGMENT TABLE
E1 DOES IT EXIST
NO
YES
USO25
F1 DMKBLDRL CALL - RELEASE SEGMENT PAGE & SWAP TABLES
USO27
G1 REFERENCE CHANNEL, CONTROL UNIT, AND DEVICE BLOCKS
H1 DO THEY EXIST
NO
YES
A2
A2
A2 DMKFRET CALL - RETURN CHANNEL BLOCKS
B2 DMKFRET CALL - RETURN CONTROL UNIT BLOCKS
C2 DMKFRET CALL - RETURN DEVICE BLOCKS
USO28
D2 REFERENCE TIMER REQUEST BLOCK FOR REAL TIMER
E2 DOES IT EXIST
NO
YES
F2 DMKFRET CALL - RETURN IT
USO29
G2 RESERVED PAGES
NO
YES
H2 CLEAR APPROPRIATE VMBLOK AND CORE TABLE POINTERS
USO30
J2 RESET RESERVE FLAGS IN CORE TABLE ENTRIES
USO31
A3 REFERENCE SYSTEM OPERATOR'S VMBLOK
B3 IS HE THE ONE LOGGING OUT
YES
NO
C3 IS HE DISCON-NECTED
YES
NO
D3 DO WE HAVE FREE STRG
YES
NO
E3 DMKFREE CALL - GET STORAGE FOR MESSAGE
USO82
F3 SET UP LOGOFF MSG FOR SYSTEM OPERATOR
G3 WAS USER DISCON-NECTED
YES
NO
A4
H3 REFERENCE USER'S TERMINAL RDEVBLOK
J3 DMKSCNRD CALL - FORM TERMINAL DEVICE ADDRESS
K3 DMKCVTBH CALL - INTO HEX, STORE IN MESSAGE
A4
A4
USO84
A4 STORE USERID IN MESSAGE; GET NO. OF USERS
B4 "VMLOGON" FLAG ON
YES
NO
C4 DECREMENT COUNT BY 1 IF HE WAS REALLY LOGGED ON
USO85
D4 DMKCVTBD NNN USERS TO DECIMAL
E4 FORCED LOGOFF
NO
YES
F4 ADD 'FORCED' TO LOGOFF MESSAGE
USO85A
G4 DMKQCNWT CALL - LOGOFF MESSAGE TO SYSTEM OPERATOR
USO86
H4 DOES TRACE EXTENSION BLOCK EXIST
NO
YES
04 A1
J4 DMKFRET CALL - RETURN TRACE EXTENSION BLOCK
04 A1
04 A1 03H4 03J4
USO88
A1 DELETE USER FROM CHAIN OF ACTIVE USERS
B1 "VMLOGON" FLAG ON
YES
NO
C1 DECREMENT COUNT OF USERS IN SYSLOCS
D1 DMKACOFF CALL - ACCOUNT OFF
USO93
E1 SYSTEM OPERATOR VMBLOK
NO
YES
A3
USO94
E2 DMKFRET CALL - RETURN USER'S VMBLOK
F2 'FORCE' ENTRY POINT
YES
NO
A3
A3 REMEMBER COMMAND AND POINTER TO NEXT VMBLOK
B3 CLEAR ENTIRE OPERATOR VMBLOK
C3 SET VM CHANNEL TABLE TO ALL FF'S
D3 RESTORE COMMAND AND POINTER TO NEXT VMBLOK
E3 BLANK OUT ACCOUNT NUMBER
F3 RESTORE FLAG - STILL THE OPERATOR'S VMBLOK
G3 FLAG VMPSW IN DISABLED WAIT STATE, FINISH UP.
USO96
H3 SWITCH R11 TO POINT TO SYSTEM VMBLOK NOW
J3 AND CHARGE SYSTEM FROM NOW ON
USO98
K3 ANY FREE STORAGE IN USE
NO
YES
A4
04 A4 05K2
USO98A
A4 DMKFRET CALL - RETURN STORAGE WE USED
A4
USO99
B4 RETURN TO CALLER
DMKUSODS
A5 DMKUSODS (DISCONN)
B5 IS USER ALREADY DISCON-NECTED
YES
NO
C5 "VMLOGON" FLAG ON
YES
NO
01 B1
D5 SIGNAL DISCONNECT ENTRY
E5 USOSUB CHECK FOR POSSIBLE HOLD OPTION
F5 DMKFREE CALL - GET STORAGE FOR MESSAGES
G5 MSGSUBR CONSTRUCT DISCONNECT MSG TO USER
H5 SET TO RETURN TO DISCRETN
J5 DMKQCNWT CALL - DISCONN MSG TO USER
K5 GO TO DMKDSPCH

| DMKUSO -- Process DISCON, FORCE, and LOGOFF Commands; Logoff Routine (Parts 3 and 4 of 6)

| DMKUSO -- Process DISCON, FORCE, and LOGOFF Commands; Logoff Routine (Parts 5 and 6 of 6)

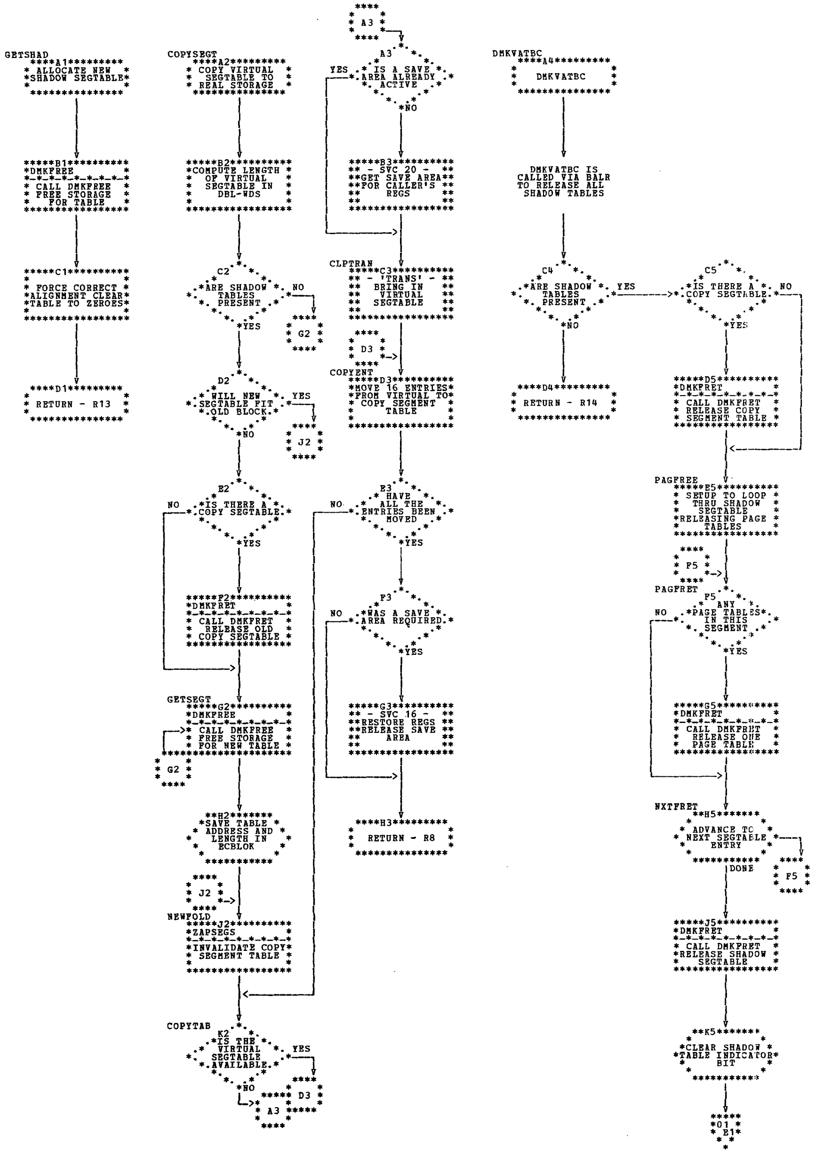

DISCRETN
A1
RETURNS HERE
B1
ADJUST RETURN TO RUN USER BUT DON'T POST READ
C1
POINT TO USER'S TERMINAL REAL DEVICE BLOCK
D1
DMKSCNRD
CALL - FORM TERMINAL DEVICE ADDRESS
E1
DMKCVTBH
CALL - AND INTO PRINTABLE HEX
F1
CLEAR PNTR TO TERMINAL RDEVBLOK, FLAG DISCON- NECTED
G1
SYSTEM OPERATOR DISCON- NECTED
YES
NO
H1
CONSTRUCT DISCONNECT MESSAGE FOR SYSTEM OPERATOR
J1
DMKCVTBD
CALL - INCLUDE NO. OF USERS
K1
DMKQCNWT
CALL - SEND MSG TO SYSTEM OPERATOR
DISCEXIT
K2
SET R1, GO RETURN MSG BUFFER & EXIT
04 A4
USOSUB
A3
USOSUB (R8)
B3
DMKSCNFD
CALL - PICK UP POSSIBLE 'HOLD' OPTION
C3
ANYTHING THERE
NO
YES
D3
IS IT 'HOLD'
NO
YES
ERROR3
D4
SET FOR DMKUSO003E - INVALID OPTION
02 F2
E3
SIGNAL 'HOLD' THE LINE
F3
RETURN - R8
MSGSUBR
A5
MSGSUBR (R3)
B5
CONSTRUCT BASIC LOGOFF OR DISCONN MESSAGE
C5
INCLUDE TIME ZONE IDENTIFIER & WEEKDAY FROM SYSLOCS
D5
DMKCVTDT
CALL - GET DATE & TIME INTO THE MSG
E5
FORCED LOGOFF
NO
YES
F5
ADD USERID WHO DID THE 'FORCE' AT END OF MSG
MSGSUBR8
G5
SET REGS AS NEEDED FOR DMKQCNWT
H5
SET R2 FOR PRIORITY + LOGDROP OPTION
J5
WAS 'HOLD' SPECIFIED
NO
06 A3
YES
K5
SET R2 FOR PRIORITY + LOGHOLD OPTION
06 A3
06 A3 05J5 05K5
MSGSUBR9
A3
DELETE TRAILING BLANKS (IF ANY) FROM THE COUNT
B3
RETURN - R3

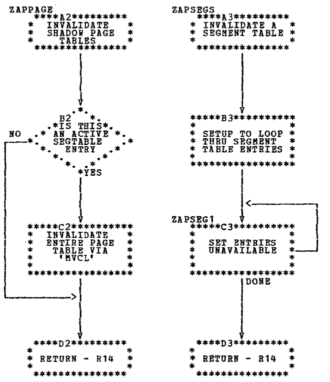

DMKVATAB
A1
DMKVATAB
DMKVATAB IS CALLED VIA BALR FOR SHADOW TABLE MAINTENANCE
C1
ARE SHADOW TABLES PRESENT
YES
NO
VATABC0
C2
CHKFORM
VALIDATE CONTROL REGISTER VALUES
D1
VACLEAN
D1
TURN OFF STATUS BITS IN VMESTAT
D2
ARE C-REG 0 AND C-REG 1 VALID
NO
YES
01 E1
02H2
02H3
03K5
BALREXT
E1
RESTORE CALLER'S REGISTER FROM 'BALRSAVE'
E2
NO
WAS VIRTUAL C-REG 0 LOADED
YES
F1
RETURN - R14
F2
SAME PAGE AND SEGMENT SIZES
NO
YES
G2
SET SHADOW C-REG 0 FROM VIRTUAL C-REG 0
VATABC1
H2
WAS VIRTUAL C-REG 1 LOADED
NO
YES
VATABPG
H3
HAVE WE STOLEN ONE OF VM'S PAGES
NO
YES
D1
J2
COPYSEGT
COPY VIRTUAL SEGMENT TABLE TO REAL STORAGE
J3
ZAPSEGS
INVALIDATE SHADOW SEGTABLE
D1
K2
SHADOWS
RESET SHADOW SEGMENT TABLE FROM COPY
TORNADO
A4
TURN OFF STATUS BITS IN VMESTAT
B4
- SVC 020 -
GET SAVE AREA
C4
SAVE ENTRY REGISTERS, RETURN ADDR IN SAVE AREA
D4
DMKVATBC
CALL - RELEASE ALL ACTIVE SHADOW TABLES
E4
DMKVATMD
CALL - REBUILD SHADOW TABLE ARCHITECTURE
F4
EXIT - SVC 12
CHKFORM
A5
VALIDATE VIRTUAL CONTROL REGISTERS
B5
TURN OFF ERROR BITS IN VMESTAT
C5
MORE THAN 256 SEGMENTS
YES
NO
F5
D5
IS SEGMENT TABLE ADDRESS VALID
NO
YES
F5
E5
YES
IS SEGTABLE ORIGIN ALIGNED
NO
F5
BADSEGT
F5
SET 'VMBADCR1' IN VMESTAT OF VMBLOK
CHKVCR0
G5
YES
ARE PAGE AND SEGMENT SIZE BITS VALID
NO
BADARCH
H5
SET 'VMBADCR0' IN VMESTAT OF VMBLOK
J5
RETURN - R8

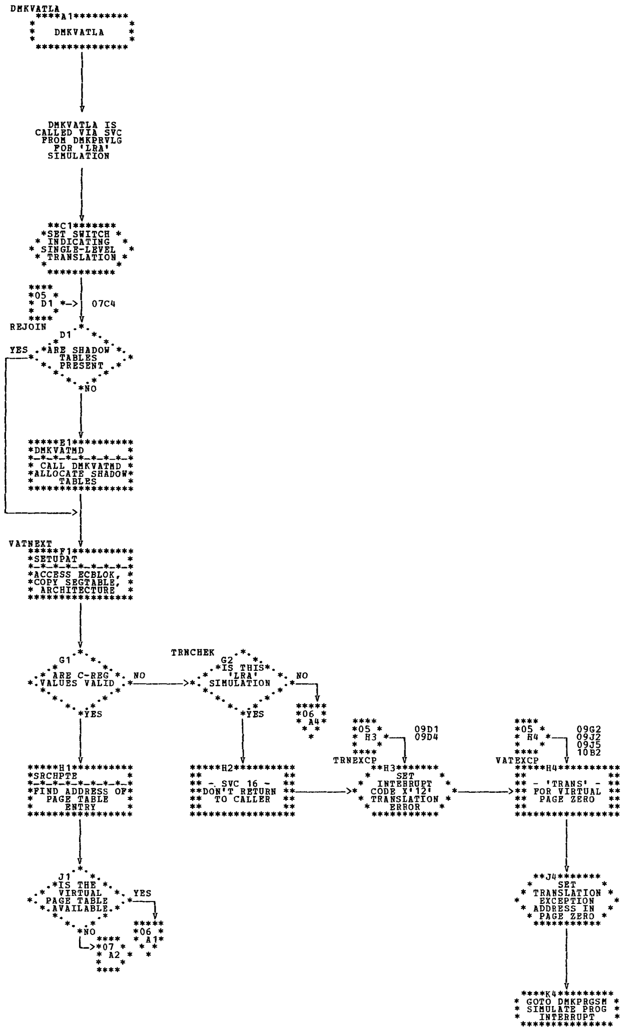

DMKVATMD
A1
DMKVATMD
CALLED VIA BALR TO CREATE SHADOW TABLES AND SHADOW CONTROL REGS
C1
ARE SHADOW TABLES PRESENT
NO
YES
C2
CHKFORM
VALIDATE CONTROL REGISTER VALUES
D1
RETURN - R14
D2
ARE VIRTUAL C-REGS 0,1 VALID
NO
YES
MINIMUM
D3
FORCE MINIMUM SEGTABLE = 16 SEGMENTS
E2
COPYSEGT
COPY VIRTUAL SEGTABLE TO REAL STORAGE
E3
GETSHAD
CREATE MINIMUM SEGMENT TABLE
F2
SHADOWS
SETUP SHADOW SEGTABLE, C-REG ONE
F3
ZAPSEGS
INVALIDATE THE SEGMENT TABLE
G2
CONSTRUCT SHADOW CONTROL REGISTER ZERO FROM VIRTUAL C-REG 0
G3
USE STANDARD C-REG 0 VALUE FOR SHADOW C-REG 0
H2
INDICATE THAT SHADOW TABLES ARE ACTIVE
01 E1
H3
INDICATE THAT SHADOW TABLES ARE ACTIVE
01 E1
SHADOWS
A4
MAINTAIN SHADOW SEGMENT TABLE
B4
YES
ARE SHADOW TABLES PRESENT
NO
C4
GETSHAD
CREATE SHADOW SEGMENT TABLE
D4
SET SHADOW SEGTABLE LENGTH AND ORIGIN IN ECBLOK
SHDINVS
E4
IS VIRT SEGTABLE LONGER THAN SHADOW TABLE
YES
NO
E4
REALLOC
E5
GETSHAD
GET NEW, LARGER SHADOW SEGTABLE
F4
ZAPSEGS
INVALIDATE SHADOW SEGMENT TABLE
F5
COPY OLD SHADOW SEGTABLE TO NEW TABLE AND CLEAR EXCESS
G4
RETURN - R8
G5
EXCHANGE SEGTABLE POINTERS AND LENGTH IN ECBLOK
H5
DMKFRET
CALL DMKFRET RELEASE OLD SHADOW SEGTABLE
E4

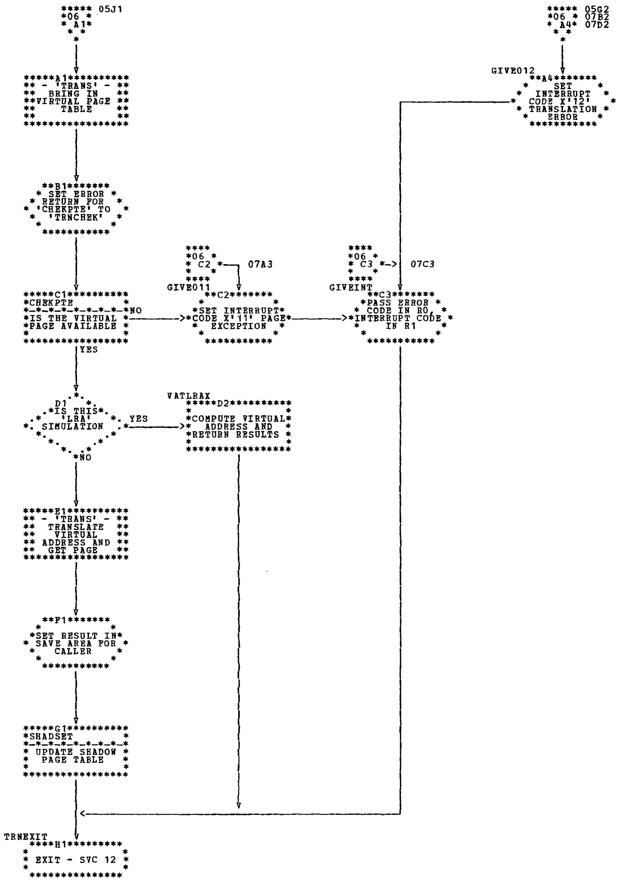
DMKVAT -- Virtual Storage Manager for EC Mode Virtual Machine that does Paging (Parts 1 and 2 of 11)

DMKVAT -- Virtual Storage Manager for EC Mode Virtual Machine that does Paging (Part 3 and 4 of 11)


GETSHAD
A1
ALLOCATE NEW SHADOW SEGTABLE
B1
DMKFREE
CALL DMKFREE FREE STORAGE FOR TABLE
C1
FORCE CORRECT ALIGNMENT CLEAR TABLE TO ZEROES
D1
RETURN - R13
COPYSEGT
A2
COPY VIRTUAL SEGTABLE TO REAL STORAGE
B2
COMPUTE LENGTH OF VIRTUAL SEGTABLE IN DBL-WDS
C2
ARE SHADOW TABLES PRESENT
NO
G2
YES
D2
WILL NEW SEGTABLE FIT OLD BLOCK
YES
J2
NO
E2
IS THERE A COPY SEGTABLE
NO
YES
F2
DMKFRET
CALL DMKFRET RELEASE OLD COPY SEGTABLE
GETSEGT
G2
DMKFREE
CALL DMKFREE FREE STORAGE FOR NEW TABLE
G2
H2
SAVE TABLE ADDRESS AND LENGTH IN ECBLOK
J2
NEWFOLD
J2
ZAPSEGS
INVALIDATE COPY SEGMENT TABLE
COPYTAB
K2
IS THE VIRTUAL SEGTABLE AVAILABLE
YES
D3
NO
A3
A3
A3
IS A SAVE AREA ALREADY ACTIVE
YES
NO
B3
- SVC 20 -
GET SAVE AREA FOR CALLER'S REGS
CLPTRAN
C3
- 'TRANS' -
BRING IN VIRTUAL SEGTABLE
D3
COPYENT
D3
MOVE 16 ENTRIES FROM VIRTUAL TO COPY SEGMENT TABLE
E3
HAVE ALL THE ENTRIES BEEN MOVED
NO
YES
F3
WAS A SAVE AREA REQUIRED
NO
YES
G3
- SVC 16 -
RESTORE REGS
RELEASE SAVE AREA
H3
RETURN - R8
DMKVATBC
A4
DMKVATBC
DMKVATBC IS CALLED VIA BALR TO RELEASE ALL SHADOW TABLES
C4
ARE SHADOW TABLES PRESENT
YES
NO
D4
RETURN - R14
C5
IS THERE A COPY SEGTABLE
NO
YES
D5
DMKFRET
CALL DMKFRET RELEASE COPY SEGMENT TABLE
PAGFREE
E5
SETUP TO LOOP THRU SHADOW SEGTABLE RELEASING PAGE TABLES
F5
PAGFRET
F5
ANY PAGE TABLES IN THIS SEGMENT
NO
YES
G5
DMKFRET
CALL DMKFRET RELEASE ONE PAGE TABLE
NXTFRET
H5
ADVANCE TO NEXT SEGTABLE ENTRY
DONE
F5
J5
DMKFRET
CALL DMKFRET RELEASE SHADOW SEGTABLE
K5
CLEAR SHADOW TABLE INDICATOR BIT
01
E1

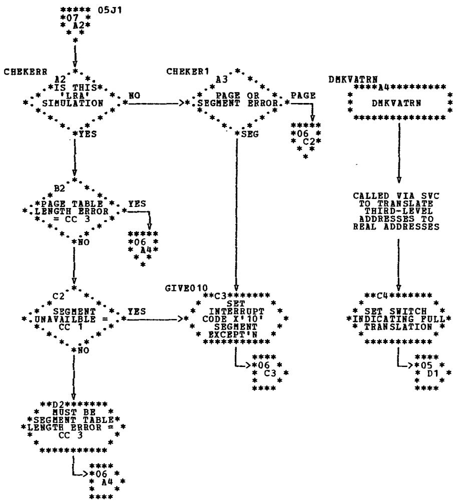

ZAPPAGE
A2
INVALIDATE SHADOW PAGE TABLES
B2
IS THIS AN ACTIVE SEGTABLE ENTRY
NO
YES
C2
INVALIDATE ENTIRE PAGE TABLE VIA 'MVCL'
D2
RETURN - R14
ZAPSEGS
A3
INVALIDATE A SEGMENT TABLE
B3
SETUP TO LOOP THRU SEGMENT TABLE ENTRIES
ZAPSEG1
C3
SET ENTRIES UNAVAILABLE
DONE
D3
RETURN - R14

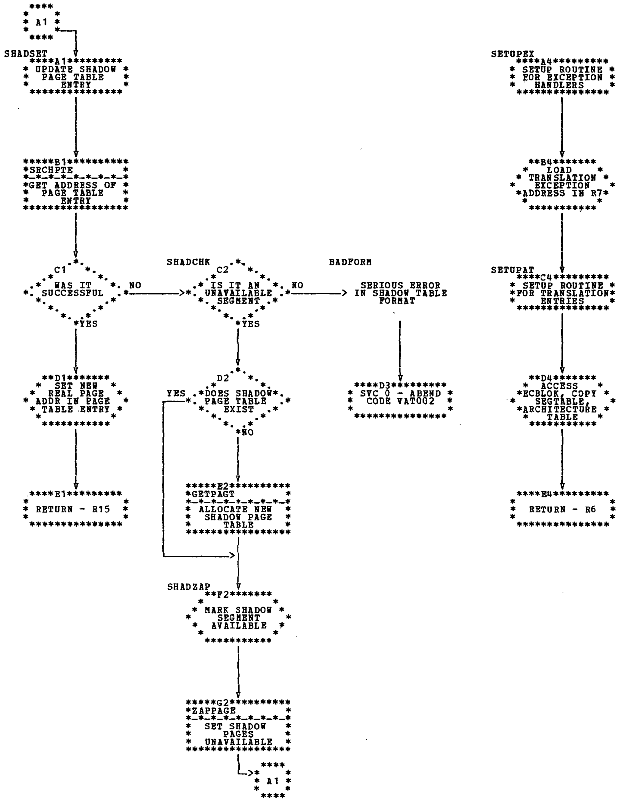

DMKVATLA
A1
DMKVATLA
DMKVATLA IS CALLED VIA SVC FROM DMKPRVLG FOR 'LRA' SIMULATION
C1
SET SWITCH INDICATING SINGLE-LEVEL TRANSLATION
05 D1
07C4
REJOIN
D1
ARE SHADOW TABLES PRESENT
YES
NO
E1
DMKVATMD
CALL DMKVATMD ALLOCATE SHADOW TABLES
VATNEXT
F1
SETUPAT
ACCESS ECBLOK, COPY SEGTABLE, ARCHITECTURE
G1
ARE C-REG VALUES VALID
NO
YES
TRNCHEK
G2
IS THIS 'LRA' SIMULATION
NO
YES
06 A4
H1
SRCHPTE
FIND ADDRESS OF PAGE TABLE ENTRY
H2
- SVC 16 - DON'T RETURN TO CALLER
05 H3
09D1
09D4
TRNEXCP
H3
SET INTERRUPT CODE X'12' TRANSLATION ERROR
05 H4
09G2
09J2
09J5
10B2
VATEXCP
H4
- 'TRANS' - FOR VIRTUAL PAGE ZERO
J1
IS THE VIRTUAL PAGE TABLE AVAILABLE
YES
NO
07 A2
06 A1
J4
SET TRANSLATION EXCEPTION ADDRESS IN PAGE ZERO
K4
GOTO DMKPRGSM SIMULATE PROG INTERRUPT

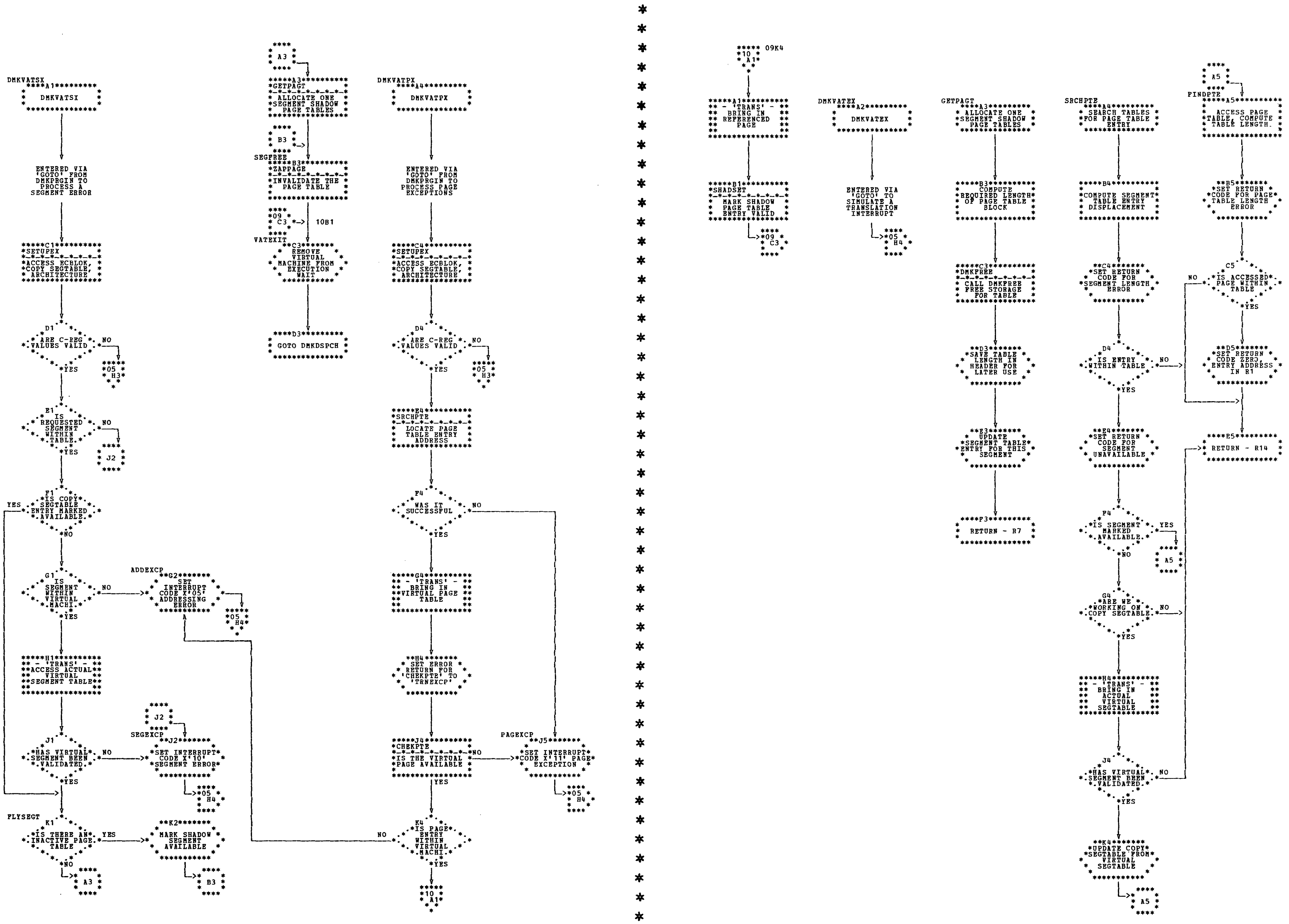

06 A1
05J1
A1
- 'TRANS' - BRING IN VIRTUAL PAGE TABLE
B1
SET ERROR RETURN FOR 'CHEKPTE' TO 'TRNCHEK'
C1
CHEKPTE
IS THE VIRTUAL PAGE AVAILABLE
NO
YES
06 C2
07A3
GIVE011
C2
SET INTERRUPT CODE X'11' PAGE EXCEPTION
06 C3
07C3
GIVEINT
C3
PASS ERROR CODE IN R0, INTERRUPT CODE IN R1
06 A4
05G2
07B2
07D2
GIVE012
A4
SET INTERRUPT CODE X'12' TRANSLATION ERROR
D1
IS THIS 'LRA' SIMULATION
YES
NO
VATLRAX
D2
COMPUTE VIRTUAL ADDRESS AND RETURN RESULTS
E1
- 'TRANS' - TRANSLATE VIRTUAL ADDRESS AND GET PAGE
F1
SET RESULT IN SAVE AREA FOR CALLER
G1
SHADSET
UPDATE SHADOW PAGE TABLE
TRNEXIT
H1
EXIT - SVC 12

| DMKVAT -- Virtual Storage Manager for EC Mode Virtual Machine that does Paging (Part 5 and 6 of 11)

| DMKVAT -- Virtual Storage Manager for EC Mode Virtual Machine that does Paging (Part 7 and 8 of 11)

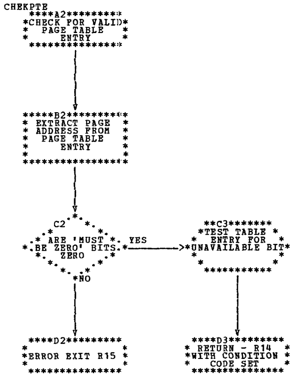

05J1
07
A2
CHEKERR
A2
IS THIS 'LRA' SIMULATION
NO
YES
CHEKER1
A3
PAGE OR SEGMENT ERROR
PAGE
06
C2
SEG
DMKVATRN
A4
DMKVATRN
CALLED VIA SVC TO TRANSLATE THIRD-LEVEL ADDRESSES TO REAL ADDRESSES
B2
PAGE TABLE LENGTH ERROR = CC 3
YES
06
A4
NO
C2
SEGMENT UNAVAILBLE = CC 1
YES
NO
GIVE010
C3
SET INTERRUPT CODE X'10' SEGMENT EXCEPT'N
06
C3
C4
SET SWITCH INDICATING FULL TRANSLATION
05
D1
D2
MUST BE SEGMENT TABLE LENGTH ERROR = CC 3
06
A4

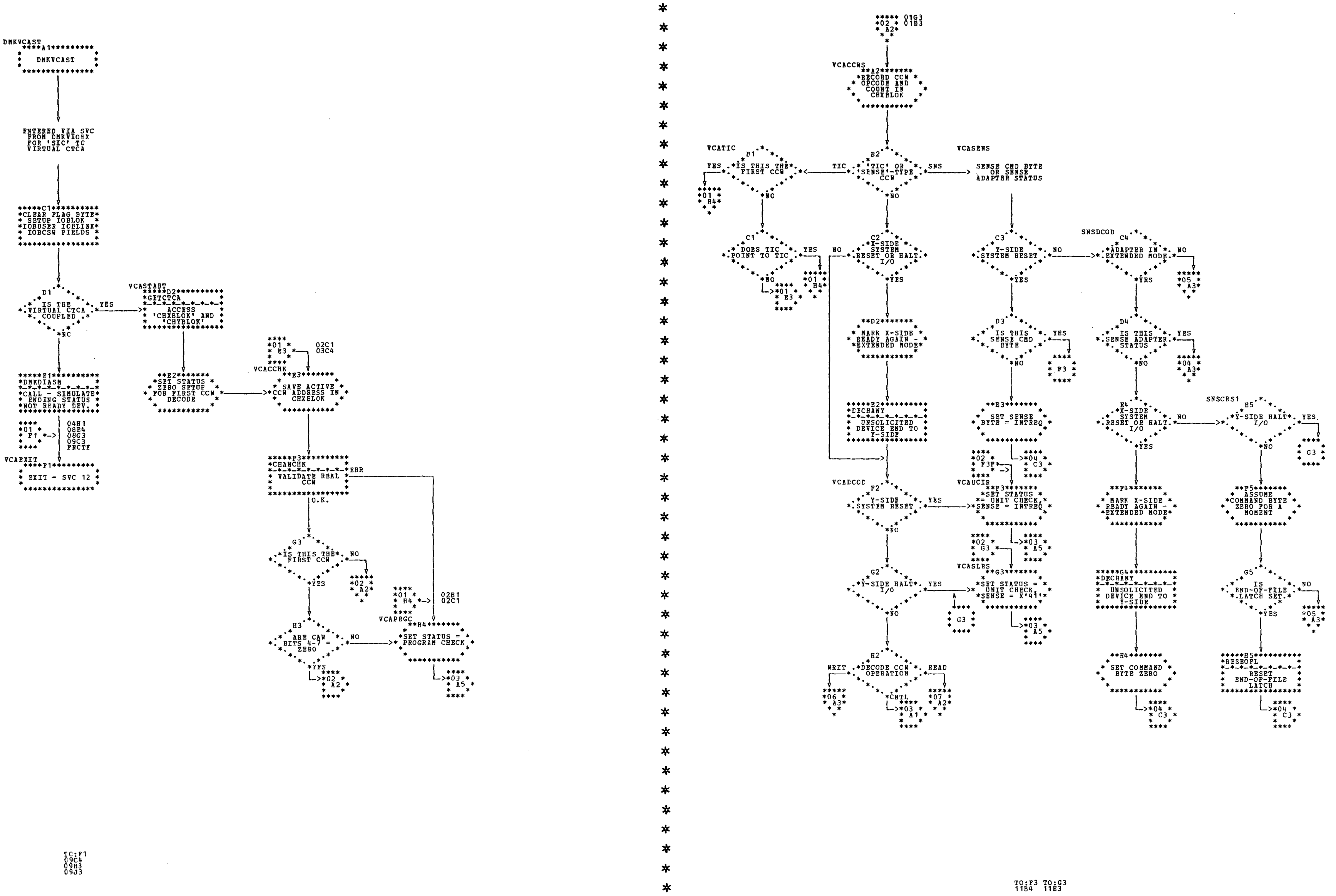

A1
SHADSET
A1
UPDATE SHADOW PAGE TABLE ENTRY
B1
SRCHPTE
GET ADDRESS OF PAGE TABLE ENTRY
C1
WAS IT SUCCESSFUL
NO
YES
SHADCHK
C2
IS IT AN UNAVAILABLE SEGMENT
NO
YES
BADFORM
SERIOUS ERROR IN SHADOW TABLE FORMAT
D1
SET NEW REAL PAGE ADDR IN PAGE TABLE ENTRY
D2
DOES SHADOW PAGE TABLE EXIST
YES
NO
D3
SVC 0 - ABEND CODE VAT002
E1
RETURN - R15
E2
GETPAGT
ALLOCATE NEW SHADOW PAGE TABLE
SHADZAP
F2
MARK SHADOW SEGMENT AVAILABLE
G2
ZAPPAGE
SET SHADOW PAGES UNAVAILABLE
A1
SETUPEX
A4
SETUP ROUTINE FOR EXCEPTION HANDLERS
B4
LOAD TRANSLATION EXCEPTION ADDRESS IN R7
SETUPAT
C4
SETUP ROUTINE FOR TRANSLATION ENTRIES
D4
ACCESS ECBLOK, COPY SEGTABLE, ARCHITECTURE TABLE
E4
RETURN - R6


DMKVATSX
A1
DMKVATSX
ENTERED VIA 'GOTO' FROM DMKPRGIN TO PROCESS A SEGMENT ERROR
C1
SETUPEX
ACCESS ECBLOK, COPY SEGTABLE, ARCHITECTURE
D1
ARE C-REG VALUES VALID
NO
YES
05 H3
E1
IS REQUESTED SEGMENT WITHIN TABLE.
NO
YES
J2
F1
IS COPY SEGTABLE ENTRY MARKED AVAILABLE.
YES
NO
G1
IS SEGMENT WITHIN VIRTUAL MACHI.
NO
YES
ADDEXCP
G2
SET INTERRUPT CODE X'05' ADDRESSING ERROR
A
05 H4
H1
- 'TRANS' -
ACCESS ACTUAL VIRTUAL SEGMENT TABLE
J1
HAS VIRTUAL SEGMENT BEEN VALIDATED.
NO
YES
J2
SEGEXCP
J2
SET INTERRUPT CODE X'10' SEGMENT ERROR
05 H4
FLYSEGT
K1
IS THERE AN INACTIVE PAGE TABLE
YES
NO
A3
K2
MARK SHADOW SEGMENT AVAILABLE
B3
A3
A3
GETPAGT
ALLOCATE ONE SEGMENT SHADOW PAGE TABLES
B3
SEGFREE
B3
ZAPPAGE
INVALIDATE THE PAGE TABLE
09 C3
10B1
VATEXIT
C3
REMOVE VIRTUAL MACHINE FROM EXECUTION WAIT
D3
GOTO DMKDSPCH
DMKVATPX
A4
DMKVATPX
ENTERED VIA 'GOTO' FROM DMKPRGIN TO PROCESS PAGE EXCEPTIONS
C4
SETUPEX
ACCESS ECBLOK, COPY SEGTABLE, ARCHITECTURE
D4
ARE C-REG VALUES VALID
NO
YES
05 H3
E4
SRCHPTE
LOCATE PAGE TABLE ENTRY ADDRESS
F4
WAS IT SUCCESSFUL
NO
YES
G4
- 'TRANS' -
BRING IN VIRTUAL PAGE TABLE
H4
SET ERROR RETURN FOR 'CHEKPTE' TO 'TRNEXCP'
J4
CHEKPTE
IS THE VIRTUAL PAGE AVAILABLE
NO
YES
PAGEXCP
J5
SET INTERRUPT CODE X'11' PAGE EXCEPTION
05 H4
K4
IS PAGE ENTRY WITHIN VIRTUAL MACHI.
NO
YES
10 A1
09K4
10 A1
A1
- 'TRANS' -
BRING IN REFERENCED PAGE
B1
SHADSET
MARK SHADOW PAGE TABLE ENTRY VALID
09 C3
DMKVATEX
A2
DMKVATEX
ENTERED VIA 'GOTO' TO SIMULATE A TRANSLATION INTERRUPT
05 H4
GETPAGT
A3
ALLOCATE ONE SEGMENT SHADOW PAGE TABLES
B3
COMPUTE REQUIRED LENGTH OF PAGE TABLE BLOCK
C3
DMKFREE
CALL DMKFREE FREE STORAGE FOR TABLE
D3
SAVE TABLE LENGTH IN HEADER FOR LATER USE
E3
UPDATE SEGMENT TABLE ENTRY FOR THIS SEGMENT
F3
RETURN - R7
SRCHPTE
A4
SEARCH TABLES FOR PAGE TABLE ENTRY
B4
COMPUTE SEGMENT TABLE ENTRY DISPLACEMENT
C4
SET RETURN CODE FOR SEGMENT LENGTH ERROR
D4
IS ENTRY WITHIN TABLE
NO
YES
E4
SET RETURN CODE FOR SEGMENT UNAVAILABLE
F4
IS SEGMENT MARKED AVAILABLE.
YES
NO
A5
G4
ARE WE WORKING ON COPY SEGTABLE.
NO
YES
H4
- 'TRANS' -
BRING IN ACTUAL VIRTUAL SEGTABLE
J4
HAS VIRTUAL SEGMENT BEEN VALIDATED.
NO
YES
K4
UPDATE COPY SEGTABLE FROM VIRTUAL SEGTABLE
A5
A5
FINDPTE
A5
ACCESS PAGE TABLE, COMPUTE TABLE LENGTH.
B5
SET RETURN CODE FOR PAGE TABLE LENGTH ERROR
C5
IS ACCESSED PAGE WITHIN TABLE
NO
YES
D5
SET RETURN CODE ZERO, ENTRY ADDRESS IN R1
E5
RETURN - R14

| DMKVAT -- Virtual Storage Manager for EC Mode Virtual Machine that dces Paging (Parts 9 and 10 of 11)

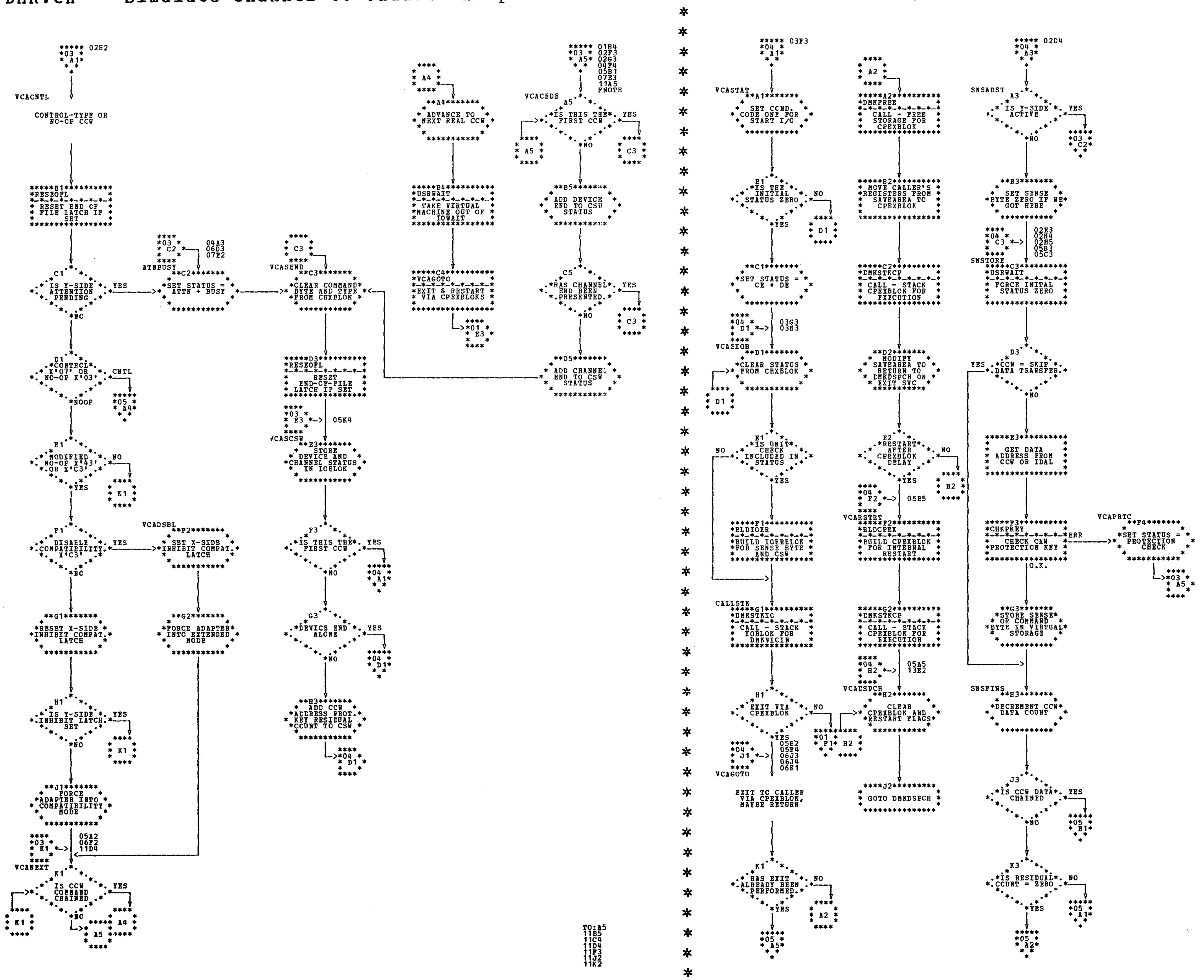
| DMKVAT -- Virtual Storage Manager for EC Mode Virtual Machine that does Paging (Part 11 of 11)

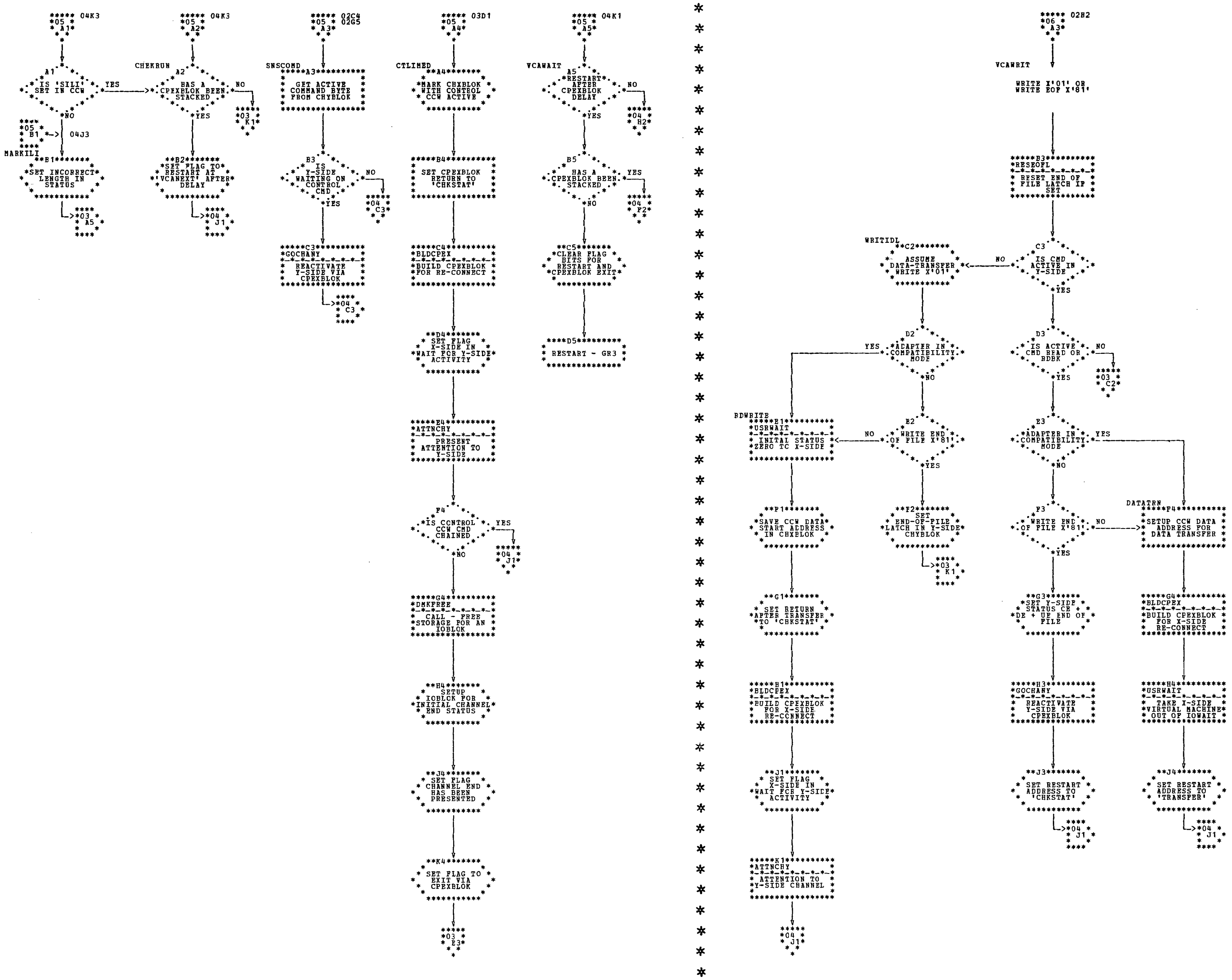

CHEKPTE
A2
CHECK FOR VALID PAGE TABLE ENTRY
B2
EXTRACT PAGE ADDRESS FROM PAGE TABLE ENTRY
C2
ARE 'MUST BE ZERO' BITS ZERO
YES
NO
C3
TEST TABLE ENTRY FOR UNAVAILABLE BIT
D2
ERROR EXIT R15
D3
RETURN - R14 WITH CONDITION CODE SET



| DMKVCA -- Simulate Channel-to-Channel Adapter between two Virtual Machines (Parts 1 and 2 of 19)

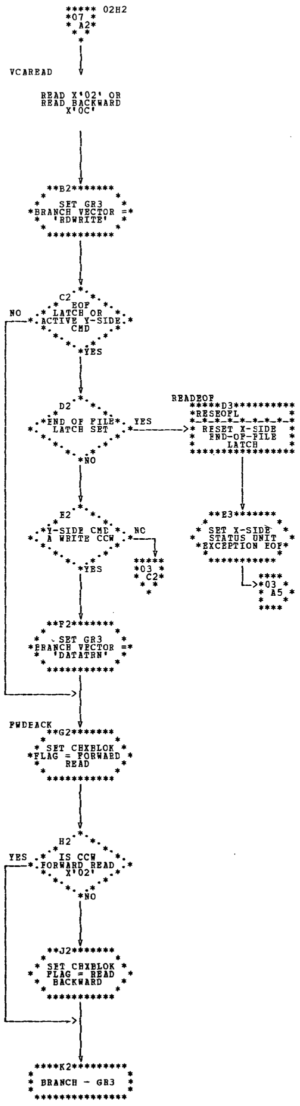
| DMKVCA -- Simulate Channel-to-Channel Adapter between two Virtual Machines (Parts 3 and 4 of 19)

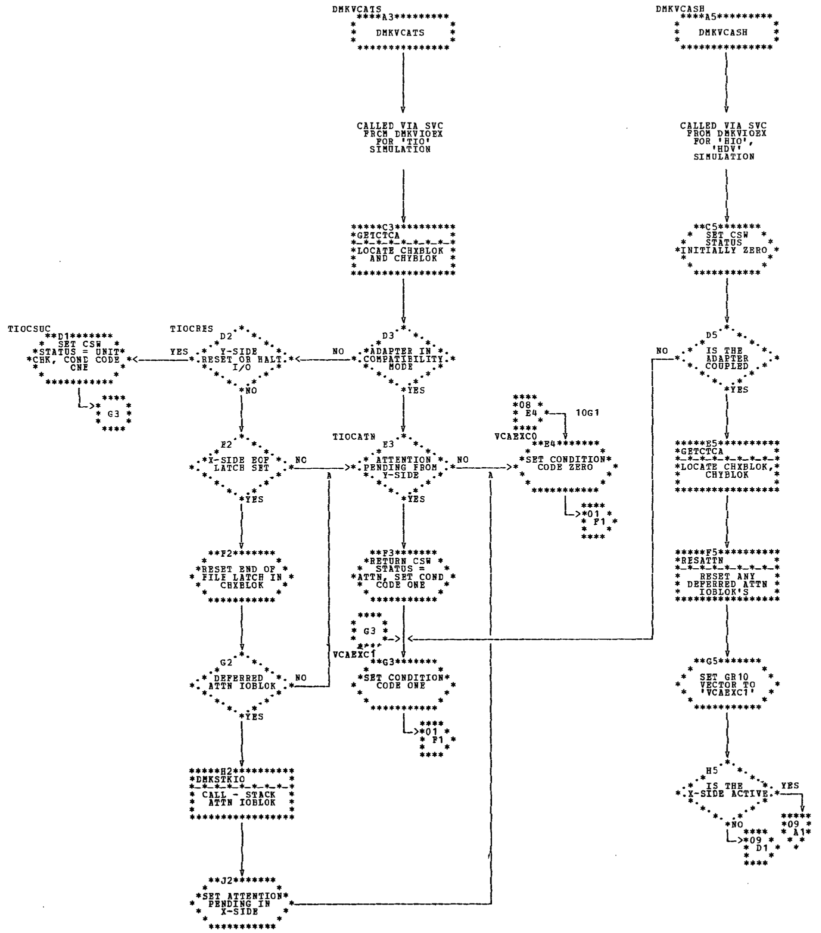

03 A1 02H2
VCACNTL
CONTROL-TYPE OR NC-OP CCW
B1 RESEOPL RESET END CF FILE LATCH IF SET
C1 IS Y-SIDE ATTENTION PENDING
YES
NC
03 C2 04A3 06D3 07E2
ATNBUSY
C2 SET STATUS = ATTN + BUSY
C3
VCASEND
C3 CLEAR COMMAND BYTE AND TYPE FROM CHXBLOK
D1 CONTRCL X'07' OR NO-OP X'03'
CNTL
NOOP
05 A4
D3 RESEOPL RESET END-OF-FILE LATCH IF SET
03 E3 05K4
VCASCSW
E3 STORE DEVICE AND CHANNEL STATUS IN IOBLOK
E1 MODIFIED NO-OP X'43' OR X'C3'
NO
K1
YES
F1 DISABLE COMPATIBILITY X'C3'
YES
VCADSBL
F2 SET X-SIDE INHIBIT COMPAT. LATCH
NC
F3 IS THIS THE FIRST CCW
YES
04 A1
NO
G1 RESET X-SIDE INHIBIT COMPAT. LATCH
G2 FORCE ADAPTER INTO EXTENDED MODE
G3 DEVICE END ALONE
YES
04 D1
NO
H1 IS Y-SIDE INHIBIT LATCH SET
YES
K1
NO
H3 ADD CCW ADDRESS PROT. KEY RESIDUAL CCUNT TO CSW
04 D1
J1 FORCE ADAPTER INTO COMPATIBILITY MODE
03 K1 05A2 06F2 11D4
VCANEXT
K1 IS CCW COMMAND CHAINED
YES
K1
NC
A5
A4
A4
A4 ADVANCE TO NEXT REAL CCW
B4 USRWAIT TAKE VIRTUAL MACHINE OUT OF IOWAIT
C4 VCAGOTO EXIT & RESTART VIA CPEXBLOKS
01 E3
03 A5 01H4 02F3 02G3 04F4 05B1 07E3 11A5 FNOTE
VCACEDE
A5 IS THIS THE FIRST CCW
YES
A5
C3
NO
B5 ADD DEVICE END TO CSW STATUS
C5 HAS CHANNEL END BEEN PRESENTED.
YES
C3
NO
D5 ADD CHANNEL END TO CSW STATUS
TO:A5 11B5 11C4 11D4 11F3 11J2 11K2
04 A1 03F3
VCASTAT
A1 SET CCND. CODE ONE FOR START I/O
B1 IS THE INITIAL STATUS ZERO
NO
D1
YES
C1 SET STATUS = CE + DE
04 D1 03G3 03H3
VCASIOB
D1 CLEAR STATUS FROM CHXBLOK
D1
E1 IS UNIT CHECK INCLUDED IN STATUS
NO
YES
F1 BLDIOER BUILD IOERBLCK FOR SENSE BYTE AND CSW
CALLSTK
G1 DMKSTKIC CALL - STACK IOBLOK FOR DMKVICIN
H1 EXIT VIA CPEXBLOK
NO
01 F1
YES
04 J1 05B2 05F4 06J3 06J4 06K1
VCAGOTO
EXIT TC CALLER VIA CPEXBLOK, MAYBE RETURN
K1 HAS EXIT ALREADY BEEN PERFORMED.
NO
A2
YES
05 A5
A2
A2 DMKFREE CALL - FREE STORAGE FOR CPEXBLOK
B2 MOVE CALLER'S REGISTERS FROM SAVEAREA TO CPEXBLOK
C2 DMKSTKCP CALL - STACK CPEXBLOK FOR EXECUTION
D2 MODIFY SAVEAREA TO RETURN TO DMKDSPCH ON EXIT SVC
E2 RESTART AFTER CPEXBLOK DELAY
NO
H2
YES
04 F2 05B5
VCARSTRT
F2 BLDCPEX BUILD CPEXBLOK FOR INTERNAL RESTART
G2 DMKSTKCP CALL - STACK CPEXBLOK FOR EXECUTION
04 H2 05A5 13B2
VCADSPCH
H2 CLEAR CPEXBLOK AND RESTART FLAGS
H2
J2 GOTO DMKDSPCH
04 A3 02D4
SNSADST
A3 IS Y-SIDE ACTIVE
YES
03 C2
NO
B3 SET SENSE BYTE ZERO IF WE GOT HERE
04 C3 02E3 02H4 02H5 05B3 05C3
SNSTORE
C3 USRWAIT FORCE INITAL STATUS ZERO
D3 CCW = SKIP DATA TRANSFER
YES
NO
E3 GET DATA ADDRESS FROM CCW OR IDAL
F3 CHKPKEY CHECK CAW PROTECTION KEY
ERR
VCAPRTC
F4 SET STATUS = PROTECTION CHECK
03 A5
O.K.
G3 STORE SENSE OR COMMAND BYTE IN VIRTUAL STORAGE
SNSFINS
H3 DECREMENT CCW DATA COUNT
J3 IS CCW DATA CHAINED
YES
05 B1
NO
K3 IS RESIDUAL CCUNT = ZERO
NO
05 A1
YES
05 A2

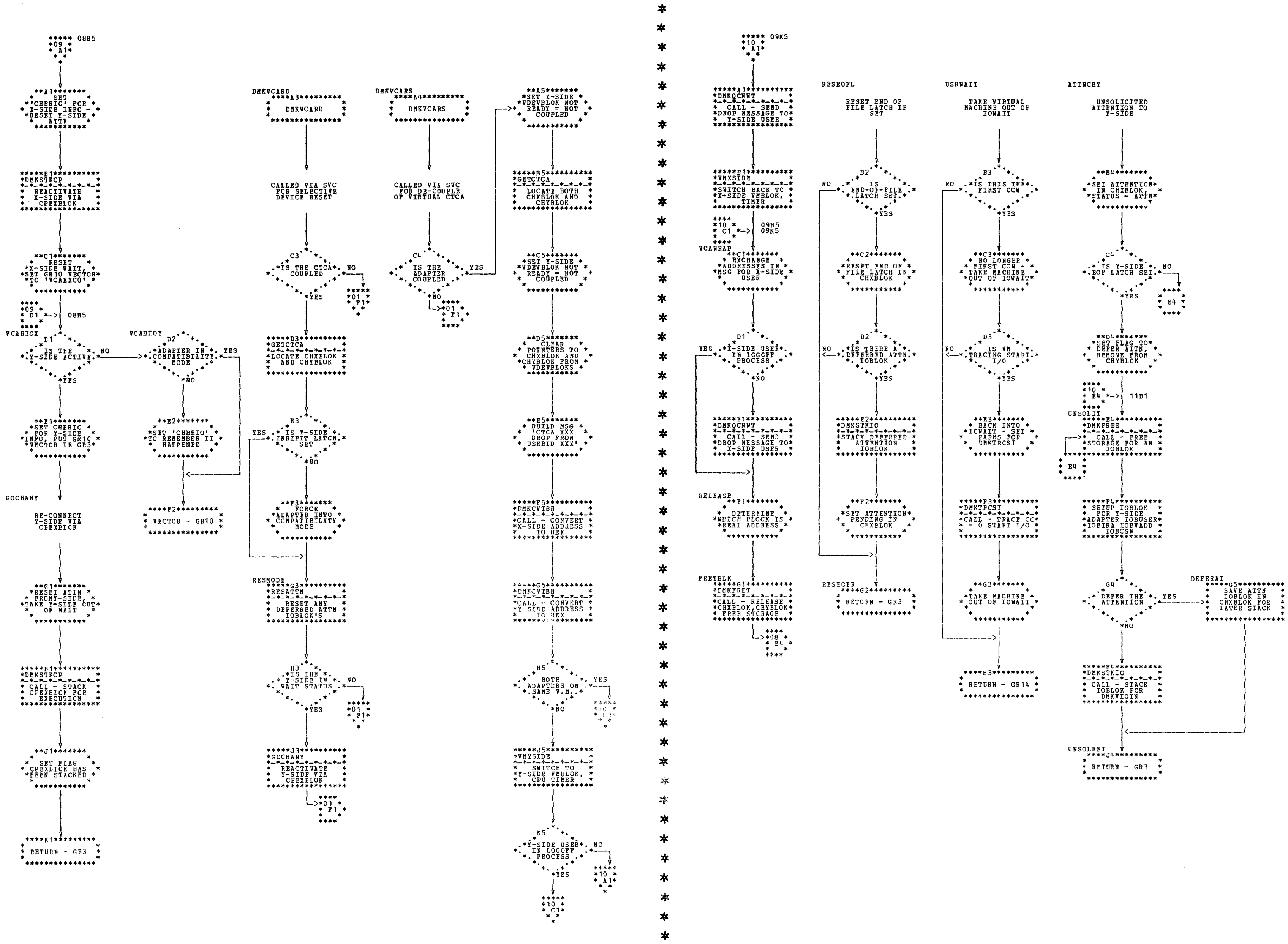

05 A1 04K3
A1 IS 'SILI' SET IN CCW
YES
NO
05 B1 04J3
MARKILI
B1 SET INCORRECT LENGTH IN STATUS
03 A5
05 A2 04K3
CHEKRUN
A2 HAS A CPEXBLOK BEEN STACKED
NO
03 K1
YES
B2 SET FLAG TO RESTART AT 'VCANEXT' AFTER DELAY
04 J1
05 A3 02C4 02G5
SNSCOMD
A3 GET ACTIVE COMMAND BYTE FROM CHYBLOK
B3 IS Y-SIDE WAITING ON CONTROL CMD
NO
04 C3
YES
C3 GOCHANY
REACTIVATE Y-SIDE VIA CPEXBLOK
04 C3
05 A4 03D1
CTLIMED
A4 MARK CHXBLOK WITH CONTROL CCW ACTIVE
B4 SET CPEXBLOK RETURN TO 'CHKSTAT'
C4 BLDCPEX
BUILD CPEXBLOK FOR RE-CONNECT
D4 SET FLAG X-SIDE IN WAIT FOR Y-SIDE ACTIVITY
E4 ATTNCHY
PRESENT ATTENTION TO Y-SIDE
F4 IS CONTROL CCW CMD CHAINED
YES
04 J1
NO
G4 DMKFREE
CALL - FREE STORAGE FOR AN IOBLOK
H4 SETUP IOBLOK FOR INITIAL CHANNEL END STATUS
J4 SET FLAG CHANNEL END HAS BEEN PRESENTED
K4 SET FLAG TO EXIT VIA CPEXBLOK
03 E3
05 A5 04K1
VCAWAIT
A5 RESTART AFTER CPEXBLOK DELAY
NO
04 H2
YES
B5 HAS A CPEXBLOK BEEN STACKED
YES
04 F2
NO
C5 CLEAR FLAG BITS FOR RESTART AND CPEXBLOK EXIT
D5 RESTART - GR3
06 A3 02B2
VCAWRIT
WRITE X'01' OR WRITE EOF X'81'
B3 RESEOFL
RESET END OF FILE LATCH IF SET
C3 IS CMD ACTIVE IN Y-SIDE
NO
YES
WRITIDL
C2 ASSUME DATA-TRANSFER WRITE X'01'
D2 ADAPTER IN COMPATIBILITY MODE
YES
NO
D3 IS ACTIVE CMD READ OR RDBK
NO
03 C2
YES
RDWRITE
E1 USRWAIT
INITAL STATUS ZERO TC X-SIDE
E2 WRITE END OF FILE X'81'
NO
YES
E3 ADAPTER IN COMPATIBILITY MODE
YES
NO
F1 SAVE CCW DATA START ADDRESS IN CHXBLOK
F2 SET END-OF-FILE LATCH IN Y-SIDE CHYBLOK
03 K1
F3 WRITE END OF FILE X'81'
NO
YES
DATATRN
F4 SETUP CCW DATA ADDRESS FOR DATA TRANSFER
G1 SET RETURN AFTER TRANSFER TO 'CHKSTAT'
G3 SET Y-SIDE STATUS CE + DE + UE END OF FILE
G4 BLDCPEX
BUILD CPEXBLOK FOR X-SIDE RE-CONNECT
H1 BLDCPEX
BUILD CPEXBLOK FOR X-SIDE RE-CCNNECT
H3 GOCHANY
REACTIVATE Y-SIDE VIA CPEXBLOK
H4 USRWAIT
TAKE X-SIDE VIRTUAL MACHINE OUT OF IOWAIT
J1 SET FLAG X-SIDE IN WAIT FCR Y-SIDE ACTIVITY
J3 SET RESTART ADDRESS TO 'CHKSTAT'
04 J1
J4 SET RESTART ADDRESS TO 'TRANSFER'
04 J1
K1 ATTNCHY
ATTENTION TO Y-SIDE CHANNEL
04 J1

| DMKVCA -- Simulate Channel-to-Channel Adapter between two Virtual Machines (Parts 5 and 6 of 19)

| DMKVCA -- Simulate Channel-to-Channel Adapter between two Virtual Machines (Parts 7 and 8 of 19)

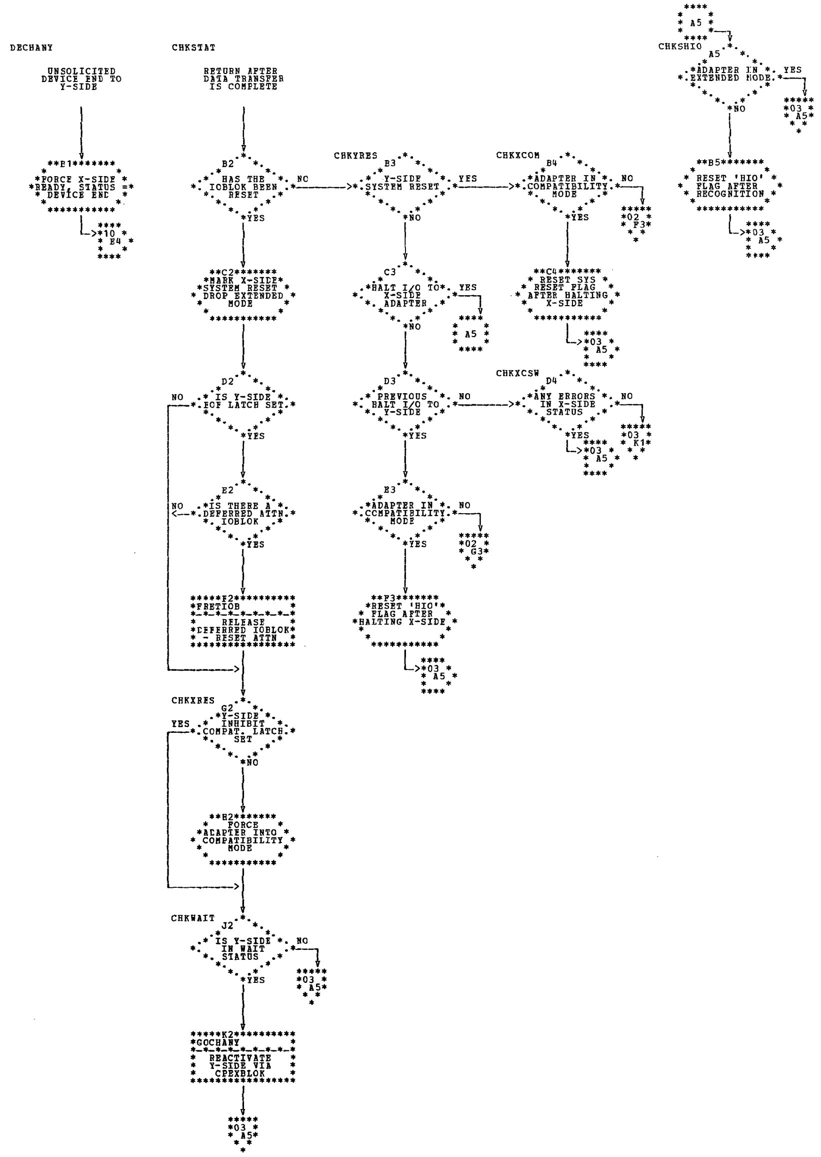

02H2
07 A2
VCAREAD
READ X'02' OR READ BACKWARD X'OC'
B2
SET GR3 BRANCH VECTOR = 'RDWRITE'
C2
EOF LATCH OR ACTIVE Y-SIDE CMD
NO
YES
D2
END OF FILE LATCH SET
YES
NO
READEOF
D3
RESEOFL
RESET X-SIDE END-OF-FILE LATCH
E3
SET X-SIDE STATUS UNIT EXCEPTION EOF
03 A5
E2
Y-SIDE CMD A WRITE CCW
NC
YES
03 C2
F2
SET GR3 BRANCH VECTOR = 'DATATRN'
FWDEACK
G2
SET CHXBLOK FLAG = FORWARD READ
H2
IS CCW FORWARD READ X'02'
YES
NO
J2
SET CHXBLOK FLAG = READ BACKWARD
K2
BRANCH - GR3

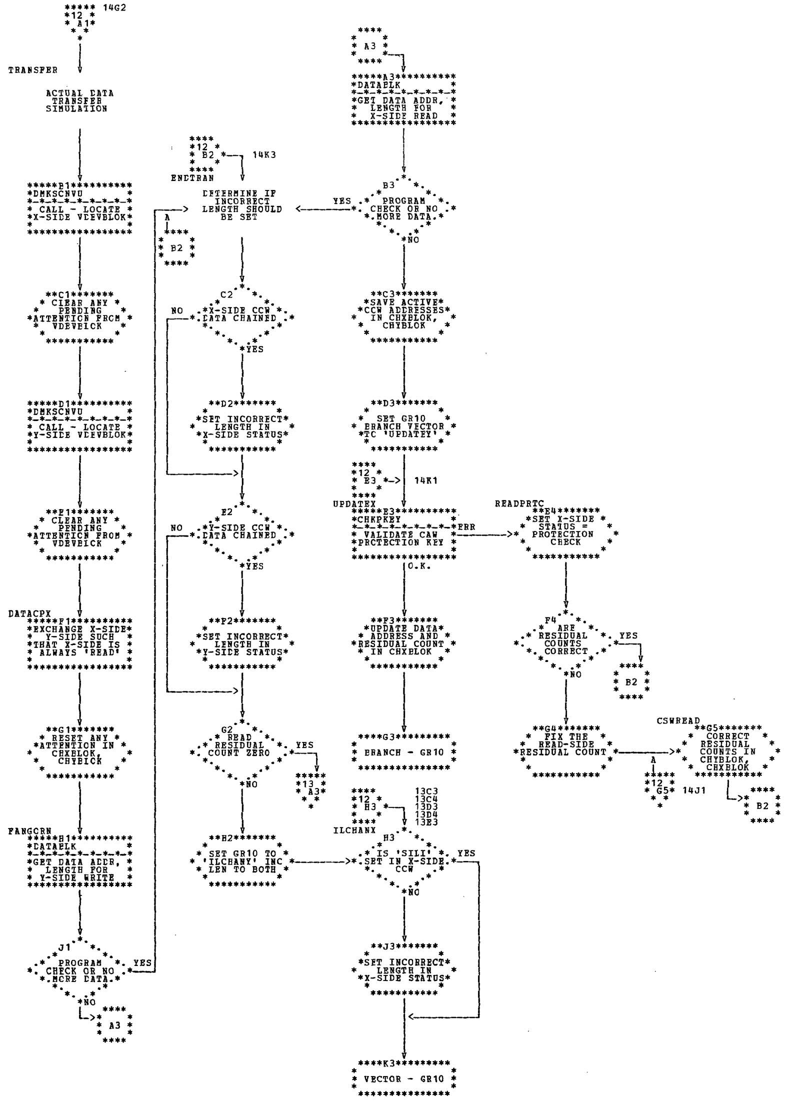

DMKVCATS
A3
DMKVCATS
CALLED VIA SVC FRCM DMKVIOEX FOR 'TIO' SIMULATION
C3
GETCTCA
LOCATE CHXBLOK AND CHYBLOK
DMKVCASH
A5
DMKVCASH
CALLED VIA SVC FROM DMKVIOEX FOR 'HIO', 'HDV' SIMULATION
C5
SET CSW STATUS INITIALLY ZERO
TIOCSUC
D1
SET CSW STATUS = UNIT CHK, COND CODE CNE
G3
TIOCRES
D2
Y-SIDE RESET OR HALT I/O
YES
NO
D3
ADAPTER IN COMPATIBILITY MODE
NO
YES
D5
IS THE ADAPTER COUPLED
NO
YES
08 E4
10G1
VCAEXC0
E4
SET CONDITION CODE ZERO
01 F1
E2
X-SIDE EOF LATCH SET
NC
YES
A
TIOCATN
E3
ATTENTION PENDING FROM Y-SIDE
NO
YES
A
E5
GETCTCA
LOCATE CHXBLOK, CHYBLOK
F2
RESET END OF FILE LATCH IN CHXBLOK
F3
RETURN CSW STATUS = ATTN, SET COND CODE ONE
F5
RESATTN
RESET ANY DEFERRED ATTN IOBLOK'S
G3
VCAEXC1
G3
SET CONDITION CODE ONE
01 F1
G2
DEFERRED ATTN IOBLOK
NO
YES
G5
SET GR10 VECTOR TO 'VCAEXC1'
H2
DMKSTKIO
CALL - STACK ATTN IOBLOK
H5
IS THE X-SIDE ACTIVE
YES
NO
09 A1
09 D1
J2
SET ATTENTION PENDING IN X-SIDE

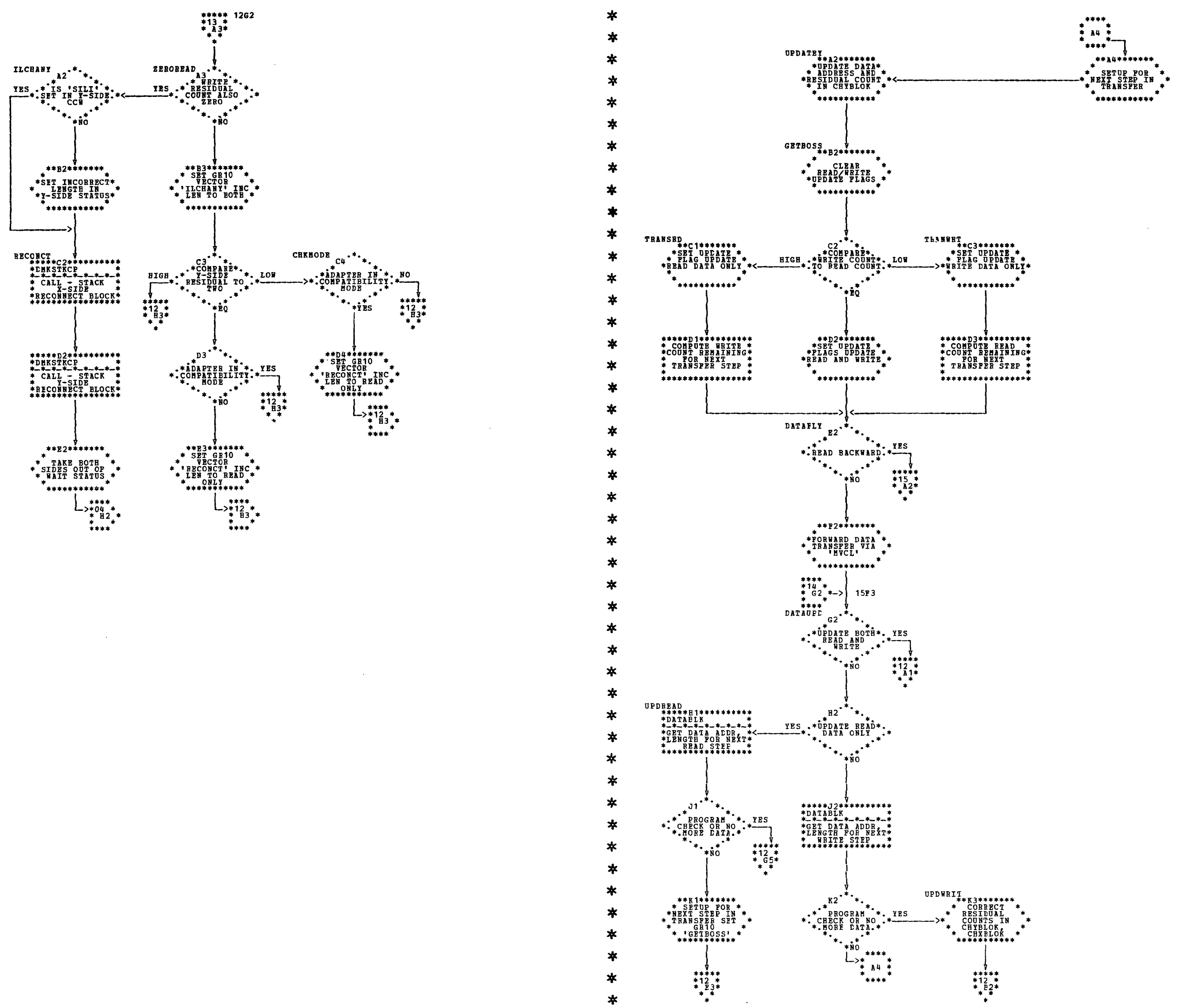

09 A1 08H5
A1 SET 'CHBHIC' FCR X-SIDE INFC - RESET Y-SIDE ATTN
B1 DMKSTKCP REACTIVATE X-SIDE VIA CPEXBLOK
C1 RESET X-SIDE WAIT, SET GR10 VECTOR TO 'VCAEXCO'
09 D1 08H5
VCAHIOX
D1 IS THE Y-SIDE ACTIVE
NO
YES
VCAHIOY
D2 ADAPTER IN COMPATIBILITY MODE
YES
NO
E1 SET CHBHIC FOR Y-SIDE INFO, PUT GR10 VECTOR IN GR3
E2 SET 'CHBHIO' TO REMEMBER IT HAPPENED
F2 VECTOR - GR10
GOCHANY
RE-CONNECT Y-SIDE VIA CPEXBICK
G1 RESET ATTN FROMY-SIDE, TAKE Y-SIDE CUT OF WAIT
H1 DMKSTKCP CALL - STACK CPEXBICK FCR EXECUTICN
J1 SET FLAG CPEXBICK HAS BEEN STACKED
K1 RETURN - GR3
DMKVCARD
A3 DMKVCARD
CALLED VIA SVC FCR SELECTIVE DEVICE RESET
C3 IS THE CTCA COUPLED
NO
01 F1
YES
D3 GETCTCA LOCATE CHXBLOK AND CHYBLOK
E3 IS Y-SIDE INHIBIT LATCH SET
YES
NO
F3 FORCE ADAPTER INTO COMPATIBILITY MODE
RESMODE
G3 RESATTN RESET ANY DEFERRED ATTN IOBLOK'S
H3 IS THE Y-SIDE IN WAIT STATUS
NO
01 F1
YES
J3 GOCHANY REACTIVATE Y-SIDE VIA CPEXBLOK
01 F1
DMKVCARS
A4 DMKVCARS
CALLED VIA SVC FOR DE-COUPLE OF VIRTUAL CTCA
C4 IS THE ADAPTER COUPLED
YES
NO
01 F1
A5 SET X-SIDE VDEVBLOK NOT READY = NOT COUPLED
B5 GETCTCA LOCATE BOTH CHXBLOK AND CHYBLOK
C5 SET Y-SIDE VDEVBLOK NOT READY = NOT COUPLED
D5 CLEAR POINTERS TO CHXBLOK AND CHYBLOK FROM VDEVBLOKS
E5 BUILD MSG 'CTCA XXX DROP FROM USERID XXX'
F5 DMKCVTBH CALL - CONVERT X-SIDE ADDRESS TO HEX
G5 DMKCVTBH CALL - CONVERT Y-SIDE ADDRESS TO HEX
H5 BOTH ADAPTERS ON SAME V.M.
YES
10 C1
NO
J5 VMYSIDE SWITCH TO Y-SIDE VMBLOK, CPU TIMER
K5 Y-SIDE USER IN LOGOFF PROCESS
NO
10 A1
YES
10 C1
10 A1 09K5
A1 DMKQCNWT CALL - SEND DROP MESSAGE TO Y-SIDE USER
B1 VMXSIDE SWITCH BACK TC X-SIDE VMBLOK, TIMER
10 C1 09H5 09K5
VCAWRAP
C1 EXCHANGE ADDRESSES IN MSG FOR X-SIDE USER
D1 X-SIDE USER IN ICGCFF PROCESS
YES
NO
E1 DMKQCNWT CALL - SEND DROP MESSAGE TO X-SIDE USER
RELEASE
F1 DETERMINE WHICH ELOCK IS REAL ADDRESS
FRETBLK
G1 DMKFRET CALL - RELEASE CHXBLOK,CHYBLOK FREE STCRAGE
08 E4
RESEOFL
RESET END OF FILE LATCH IF SET
B2 IS END-OF-FILE LATCH SET
NO
YES
C2 RESET END OF FILE LATCH IN CHXBLOK
D2 IS THERE A DEFERRED ATTN IOBLOK
NO
YES
E2 DMKSTKIO STACK DEFERRED ATTENTION IOBLOK
F2 SET ATTENTION PENDING IN CHXBLOK
RESECFR
G2 RETURN - GR3
USRWAIT
TAKE VIRTUAL MACHINE OUT OF IOWAIT
B3 IS THIS THE FIRST CCW
NO
YES
C3 NO LONGER FIRST CCW - TAKE MACHINE OUT OF IOWAIT
D3 IS VM TRACING START I/O
NO
YES
E3 BACK INTO ICWAIT - SET PARMS FOR DMKTRCSI
F3 DMKTRCSI CALL - TRACE CC = 0 START I/O
G3 TAKE MACHINE OUT OF IOWAIT
H3 RETURN - GR14
ATTNCHY
UNSOLICITED ATTENTION TO Y-SIDE
B4 SET ATTENTION IN CHXBLOK, STATUS = ATTN
C4 IS Y-SIDE EOF LATCH SET
NO
E4
YES
D4 SET FLAG TO DEFER ATTN, REMOVE FROM CHYBLOK
10 E4 11B1
UNSOLIT
E4 DMKFREE CALL - FREE STORAGE FOR AN IOBLOK
E4
F4 SETUP IOBLOK FOR Y-SIDE ADAPTER IOBUSER IOBIRA IOBVADD IOBCSW
G4 DEFER THE ATTENTION
YES
NO
DEFERAT
G5 SAVE ATTN IOBLOK IN CHXBLOK FOR LATER STACK
H4 DMKSTKIO CALL - STACK IOBLOK FOR DMKVIOIN
UNSOLRET
J4 RETURN - GR3

| DMKVCA -- Simulate Channel-to-Channel Adapter between two Virtual Machines (Parts 9 and 10 of 19)

DMKVCA -- Simulate Channel-to-Channel Adapter between two Virtual Machines (Parts 11 and 12 of 19)

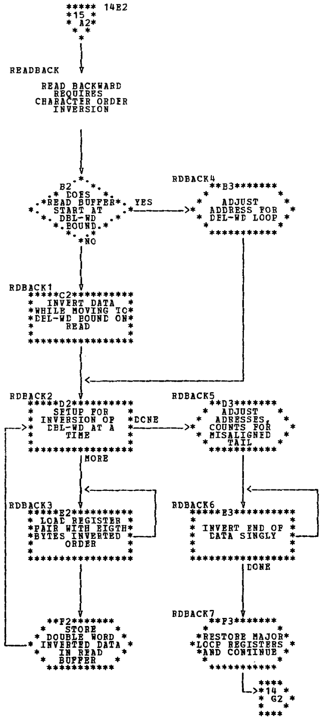

DECHANY
UNSOLICITED DEVICE END TO Y-SIDE
B1
FORCE X-SIDE READY, STATUS = DEVICE END
10 E4
CHKSTAT
RETURN AFTER DATA TRANSFER IS COMPLETE
B2
HAS THE IOBLOK BEEN RESET
NC
YES
C2
MARK X-SIDE SYSTEM RESET DROP EXTENDED MODE
D2
IS Y-SIDE EOF LATCH SET
NO
YES
E2
IS THERE A DEFERRED ATTN. IOBLOK
NO
YES
F2
FRETIOB
RELEASE DEFERRED IOBLOK - RESET ATTN
CHKXRES
G2
Y-SIDE INHIBIT COMPAT. LATCH SET
YES
NO
H2
FORCE ADAPTER INTO COMPATIBILITY MODE
CHKWAIT
J2
IS Y-SIDE IN WAIT STATUS
NO
03 A5
YES
K2
GOCHANY
REACTIVATE Y-SIDE VIA CPEXBLOK
03 A5
CHKYRES
B3
Y-SIDE SYSTEM RESET
YES
NO
C3
HALT I/O TO X-SIDE ADAPTER
YES
A5
NO
D3
PREVIOUS HALT I/O TO Y-SIDE
NO
YES
E3
ADAPTER IN COMPATIBILITY MODE
NO
02 G3
YES
F3
RESET 'HIO' FLAG AFTER HALTING X-SIDE
03 A5
CHKXCOM
B4
ADAPTER IN COMPATIBILITY MODE
NO
02 F3
YES
C4
RESET SYS RESET FLAG AFTER HALTING X-SIDE
03 A5
CHKXCSW
D4
ANY ERRORS IN X-SIDE STATUS
NO
03 K1
YES
03 A5
A5
CHKSHIO
A5
ADAPTER IN EXTENDED MODE
YES
03 A5
NO
B5
RESET 'HIO' FLAG AFTER RECOGNITION
03 A5

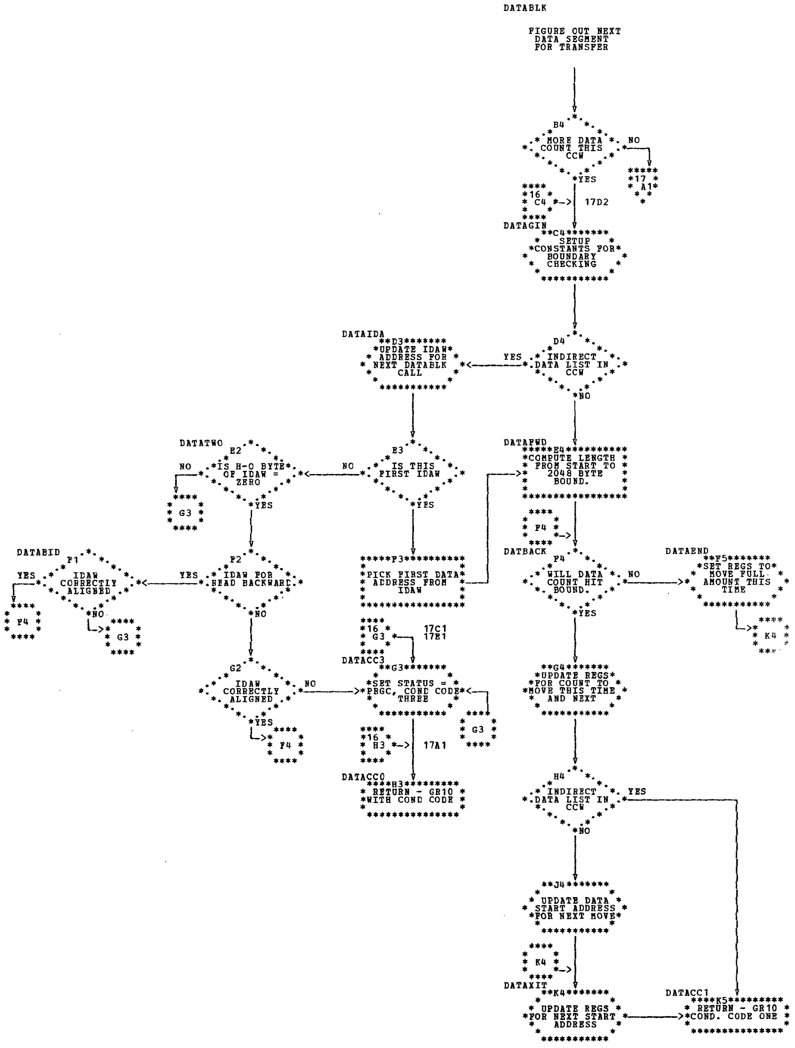

12 A1 14G2
TRANSFER
ACTUAL DATA TRANSFER SIMULATION
B1
DMKSCNVU
CALL - LOCATE X-SIDE VDEVBLOK
C1
CLEAR ANY PENDING ATTENTION FROM VDEVBLOK
D1
DMKSCNVU
CALL - LOCATE Y-SIDE VDEVBLOK
E1
CLEAR ANY PENDING ATTENTION FROM VDEVBLOK
DATACPX
F1
EXCHANGE X-SIDE Y-SIDE SUCH THAT X-SIDE IS ALWAYS 'READ'
G1
RESET ANY ATTENTION IN CHXBLOK, CHYBLOK
FANGCRN
H1
DATABLK
GET DATA ADDR, LENGTH FOR Y-SIDE WRITE
J1
PROGRAM CHECK OR NO MORE DATA
YES
NO
A3
12 B2 14K3
ENDTRAN
DETERMINE IF INCORRECT LENGTH SHOULD BE SET
A
B2
C2
X-SIDE CCW DATA CHAINED
NO
YES
D2
SET INCORRECT LENGTH IN X-SIDE STATUS
E2
Y-SIDE CCW DATA CHAINED
NO
YES
F2
SET INCORRECT LENGTH IN Y-SIDE STATUS
G2
READ RESIDUAL COUNT ZERO
YES
13 A3
NO
H2
SET GR10 TO 'ILCHANY' INC LEN TO BOTH
A3
A3
DATABLK
GET DATA ADDR, LENGTH FOR X-SIDE READ
B3
PROGRAM CHECK OR NO MORE DATA
YES
NO
C3
SAVE ACTIVE CCW ADDRESSES IN CHXBLOK, CHYBLOK
D3
SET GR10 BRANCH VECTOR TC 'UPDATEY'
12 E3 14K1
UPDATEX
E3
CHKPKEY
VALIDATE CAW PRCTECTION KEY
ERR
O.K.
F3
UPDATE DATA ADDRESS AND RESIDUAL COUNT IN CHXBLOK
G3
BRANCH - GR10
12 H3 13C3 13C4 13D3 13D4 13E3
ILCHANX
H3
IS 'SILI' SET IN X-SIDE CCW
YES
NO
J3
SET INCORRECT LENGTH IN X-SIDE STATUS
K3
VECTOR - GR10
READPRTC
E4
SET X-SIDE STATUS = PROTECTION CHECK
F4
ARE RESIDUAL COUNTS CORRECT
YES
B2
NO
G4
FIX THE READ-SIDE RESIDUAL COUNT
A
12 G5 14J1
CSWREAD
G5
CORRECT RESIDUAL COUNTS IN CHYBLOK, CHXBLOK
B2

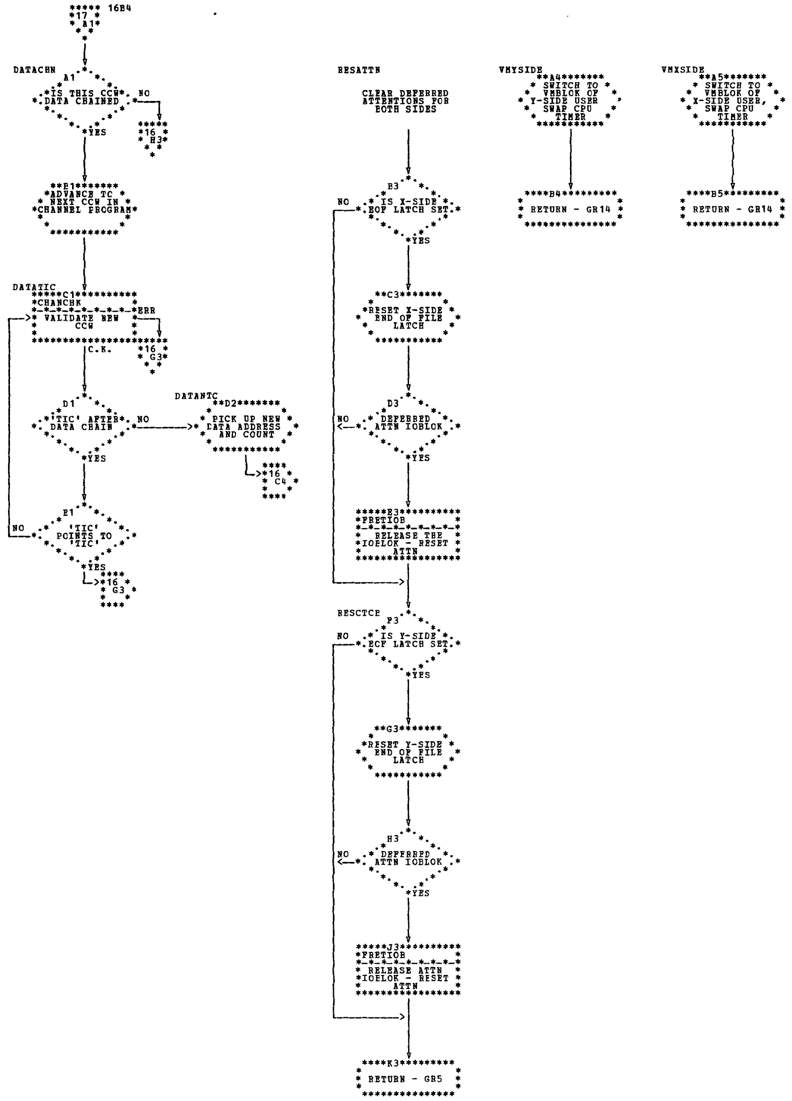

12G2
13 A3
ILCHANY A2 IS 'SILI' SET IN Y-SIDE CCW
YES
NO
ZEROREAD A3 WRITE RESIDUAL COUNT ALSO ZERO
YES
NO
B2 SET INCORRECT LENGTH IN Y-SIDE STATUS
B3 SET GR10 VECTOR 'ILCHANY' INC LEN TO BOTH
RECONCT C2 DMKSTKCP CALL - STACK X-SIDE RECONNECT BLOCK
C3 COMPARE Y-SIDE RESIDUAL TO TWO
HIGH
LOW
EQ
12 H3
CHKMODE C4 ADAPTER IN COMPATIBILITY MODE
NO
YES
12 H3
D2 DMKSTKCP CALL - STACK Y-SIDE RECONNECT BLOCK
D3 ADAPTER IN COMPATIBILITY MODE
YES
NO
12 H3
D4 SET GR10 VECTOR 'RECONCT' INC LEN TO READ ONLY
12 H3
E2 TAKE BOTH SIDES OUT OF WAIT STATUS
04 H2
E3 SET GR10 VECTOR 'RECONCT' INC LEN TO READ ONLY
12 H3
A4
UPDATEY A2 UPDATE DATA ADDRESS AND RESIDUAL COUNT IN CHYBLOK
A4 SETUP FOR NEXT STEP IN TRANSFER
GETBOSS B2 CLEAR READ/WRITE UPDATE FLAGS
TRANSRD C1 SET UPDATE FLAG UPDATE READ DATA ONLY
C2 COMPARE WRITE COUNT TO READ COUNT
HIGH
LOW
EQ
TRANWRT C3 SET UPDATE FLAG UPDATE WRITE DATA ONLY
D1 COMPUTE WRITE COUNT REMAINING FOR NEXT TRANSFER STEP
D2 SET UPDATE FLAGS UPDATE READ AND WRITE
D3 COMPUTE READ COUNT REMAINING FOR NEXT TRANSFER STEP
DATAFLY E2 READ BACKWARD
YES
NO
15 A2
F2 FORWARD DATA TRANSFER VIA 'MVCL'
14 G2
15F3
DATAUPD G2 UPDATE BOTH READ AND WRITE
YES
NO
12 A1
UPDREAD H1 DATABLK GET DATA ADDR, LENGTH FOR NEXT READ STEP
H2 UPDATE READ DATA ONLY
YES
NO
J1 PROGRAM CHECK OR NO MORE DATA
YES
NO
12 G5
J2 DATABLK GET DATA ADDR, LENGTH FOR NEXT WRITE STEP
K1 SETUP FOR NEXT STEP IN TRANSFER SET GR10 'GETBOSS'
12 B3
K2 PROGRAM CHECK OR NO MORE DATA
YES
NO
A4
UPDWRIT K3 CORRECT RESIDUAL COUNTS IN CHYBLOK, CHXBLOK
12 B2

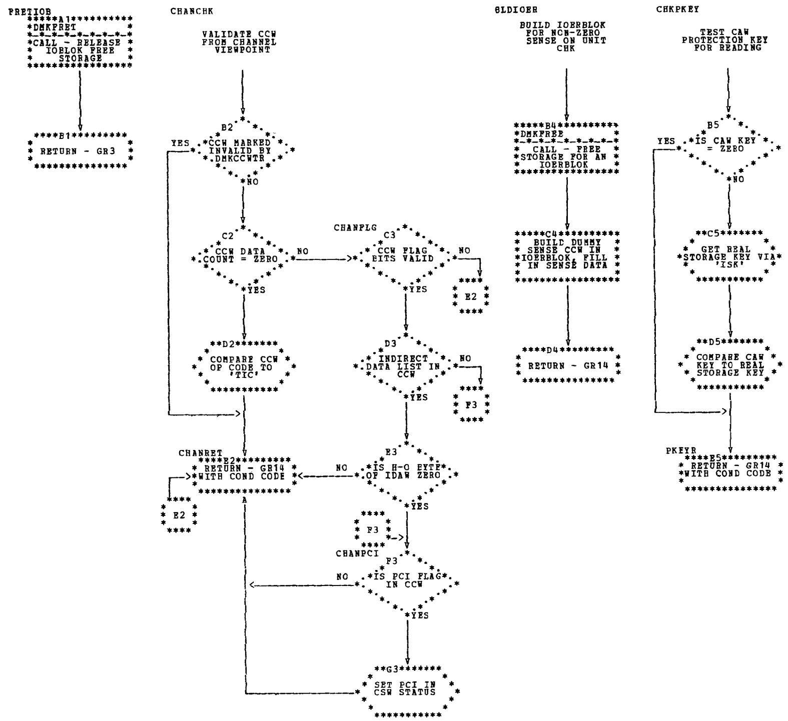
| DMKVCA -- Simulate Channel-to-Channel Adapter between two Virtual Machines (Parts 13 and 14 of 19)

DMKVCA -- Simulate Channel-to-Channel Adapter between two Virtual Machines (Parts 15 and 16 of 19)


15 A2 14E2
READBACK
READ BACKWARD REQUIRES CHARACTER ORDER INVERSION
B2 DOES READ BUFFER START AT DBL-WD BOUND.
YES
NO
RDBACK4 B3 ADJUST ADDRESS FOR DEL-WD LOOP
RDBACK1 C2 INVERT DATA WHILE MOVING TO DEL-WD BOUND ON READ
RDBACK2 D2 SETUP FOR INVERSION OF DBL-WD AT A TIME
DONE
MORE
RDBACK5 D3 ADJUST ADRESSES, COUNTS FOR MISALIGNED TAIL
RDBACK3 E2 LOAD REGISTER PAIR WITH EIGTH BYTES INVERTED ORDER
RDBACK6 E3 INVERT END OF DATA SINGLY
DONE
F2 STORE DOUBLE WORD INVERTED DATA IN READ BUFFER
RDBACK7 F3 RESTORE MAJOR LOCP REGISTERS AND CONTINUE
14 G2

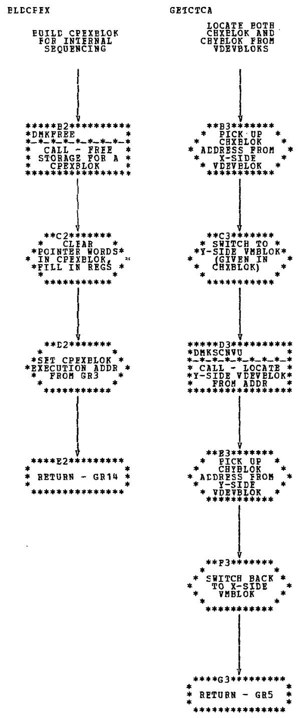

DATABLK
FIGURE OUT NEXT DATA SEGMENT FOR TRANSFER
B4 MORE DATA COUNT THIS CCW
NO
17 A1
YES
16 C4
17D2
DATAGIN C4 SETUP CONSTANTS FOR BOUNDARY CHECKING
D4 INDIRECT DATA LIST IN CCW
YES
NO
DATAIDA D3 UPDATE IDAW ADDRESS FOR NEXT DATABLK CALL
E3 IS THIS FIRST IDAW
NO
YES
DATATWO E2 IS H-O BYTE OF IDAW = ZERO
NO
G3
YES
DATAFWD E4 COMPUTE LENGTH FROM START TO 2048 BYTE BOUND.
F3 PICK FIRST DATA ADDRESS FROM IDAW
DATABID F1 IDAW CORRECTLY ALIGNED
YES
F4
NO
G3
F2 IDAW FOR READ BACKWARD
YES
NO
F4
DATBACK F4 WILL DATA COUNT HIT BOUND.
NO
YES
DATAEND F5 SET REGS TO MOVE FULL AMOUNT THIS TIME
K4
16 G3
17C1
17E1
DATACC3 G3 SET STATUS = PRGC, COND CODE THREE
G2 IDAW CORRECTLY ALIGNED
NO
YES
F4
G3
16 H3
17A1
DATACC0 H3 RETURN - GR10 WITH COND CODE
G4 UPDATE REGS FOR COUNT TO MOVE THIS TIME AND NEXT
H4 INDIRECT DATA LIST IN CCW
YES
NO
J4 UPDATE DATA START ADDRESS FOR NEXT MOVE
K4
DATAXIT K4 UPDATE REGS FOR NEXT START ADDRESS
DATACC1 K5 RETURN - GR10 COND. CODE ONE

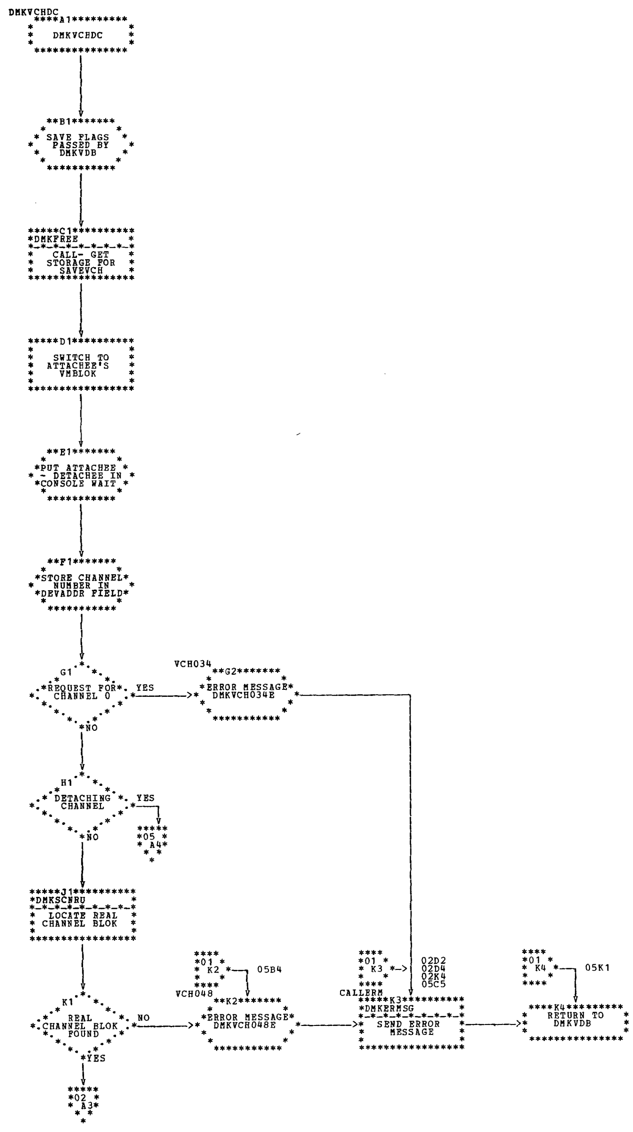

16B4
17 A1
DATACHN
A1 IS THIS CCW DATA CHAINED
NO
16 H3
YES
B1 ADVANCE TC NEXT CCW IN CHANNEL PROGRAM
DATATIC
C1 CHANCHK VALIDATE NEW CCW
ERR
16 G3
C.K.
D1 'TIC' AFTER DATA CHAIN
NO
DATANTC
D2 PICK UP NEW DATA ADDRESS AND COUNT
16 C4
YES
E1 'TIC' POINTS TO 'TIC'
NO
YES
16 G3
RESATTN
CLEAR DEFERRED ATTENTIONS FOR BOTH SIDES
B3 IS X-SIDE EOF LATCH SET
NO
YES
C3 RESET X-SIDE END OF FILE LATCH
D3 DEFERRED ATTN IOBLOK
NO
YES
E3 FRETIOB RELEASE THE IOBLOK - RESET ATTN
RESCTCE
F3 IS Y-SIDE ECF LATCH SET
NO
YES
G3 RESET Y-SIDE END OF FILE LATCH
H3 DEFERRED ATTN IOBLOK
NO
YES
J3 FRETIOB RELEASE ATTN IOBLOK - RESET ATTN
K3 RETURN - GR5
VMYSIDE
A4 SWITCH TO VMBLOK OF Y-SIDE USER SWAP CPU TIMER
B4 RETURN - GR14
VMXSIDE
A5 SWITCH TO VMBLOK OF X-SIDE USER, SWAP CPU TIMER
B5 RETURN - GR14

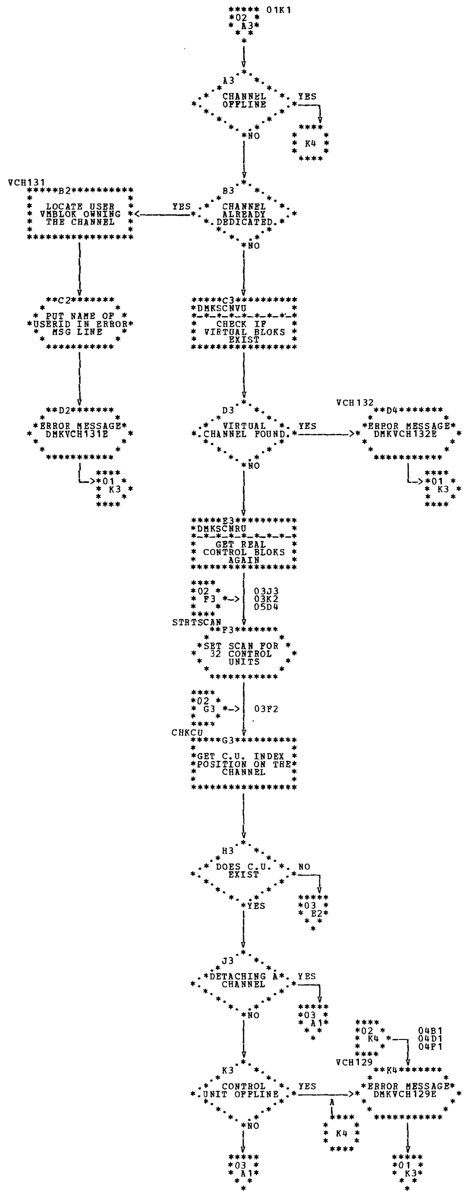

FRETIOB
A1 DMKFRET CALL - RELEASE IOBLOK FREE STORAGE
B1 RETURN - GR3
CHANCHK
VALIDATE CCW FROM CHANNEL VIEWPOINT
B2 CCW MARKED INVALID BY DMKCCWTR
YES
NO
C2 CCW DATA COUNT = ZERO
NO
YES
D2 COMPARE CCW OP CODE TO 'TIC'
CHANFLG
C3 CCW FLAG BITS VALID
NO
E2
YES
D3 INDIRECT DATA LIST IN CCW
NO
F3
YES
CHANRET
E2 RETURN - GR14 WITH COND CODE
E2
E3 IS H-O BYTE OF IDAW ZERO
NO
YES
F3
CHANPCI
F3 IS PCI FLAG IN CCW
NO
YES
G3 SET PCI IN CSW STATUS
BLDIOER
BUILD IOERBLOK FOR NON-ZERO SENSE ON UNIT CHK
B4 DMKFREE CALL - FREE STORAGE FOR AN IOERBLOK
C4 BUILD DUMMY SENSE CCW IN IOERBLOK, FILL IN SENSE DATA
D4 RETURN - GR14
CHKPKEY
TEST CAW PROTECTION KEY FOR READING
B5 IS CAW KEY = ZERO
YES
NO
C5 GET REAL STORAGE KEY VIA 'ISK'
D5 COMPARE CAW KEY TO REAL STORAGE KEY
PKEYR
E5 RETURN - GR14 WITH COND CODE

| DMKVCA -- Simulate Channel-to-Channel Adapter between two Virtual Machines (Parts 17 and 18 of 19)

| DMKVCA -- Simulate Channel-to-Channel Adapter between two Virtual Machines (Part 19 of 19)

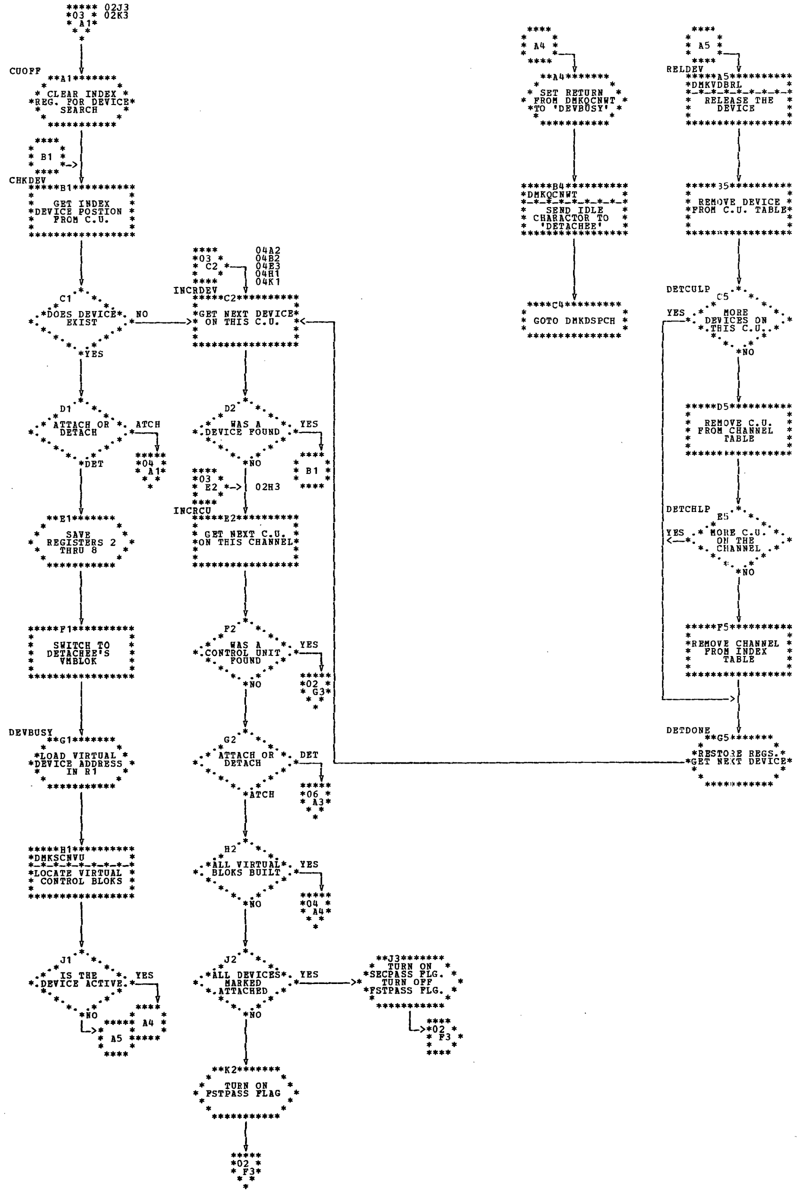

BLDCPEX
BUILD CPEXBLOK FOR INTERNAL SEQUENCING
B2
DMKFREE
CALL - FREE STORAGE FOR A CPEXBLOK
C2
CLEAR POINTER WORDS IN CPEXBLOK, FILL IN REGS
D2
SET CPEXBLOK EXECUTION ADDR FROM GR3
E2
RETURN - GR14
GETCTCA
LOCATE BOTH CHXBLOK AND CHYBLOK FROM VDEVBLOKS
B3
PICK UP CHXBLOK ADDRESS FROM X-SIDE VDEVBLOK
C3
SWITCH TO Y-SIDE VMBLOK (GIVEN IN CHXBLOK)
D3
DMKSCNVU
CALL - LOCATE Y-SIDE VDEVBLOK FROM ADDR
E3
PICK UP CHYBLOK ADDRESS FROM Y-SIDE VDEVBLOK
F3
SWITCH BACK TO X-SIDE VMBLOK
G3
RETURN - GR5

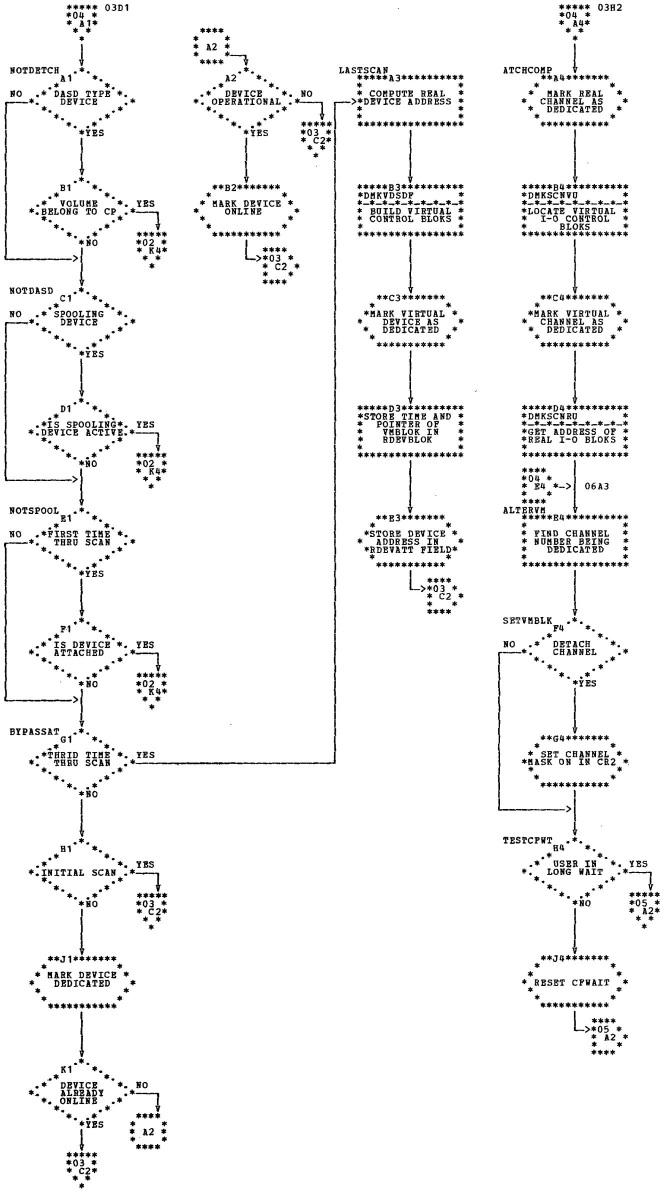


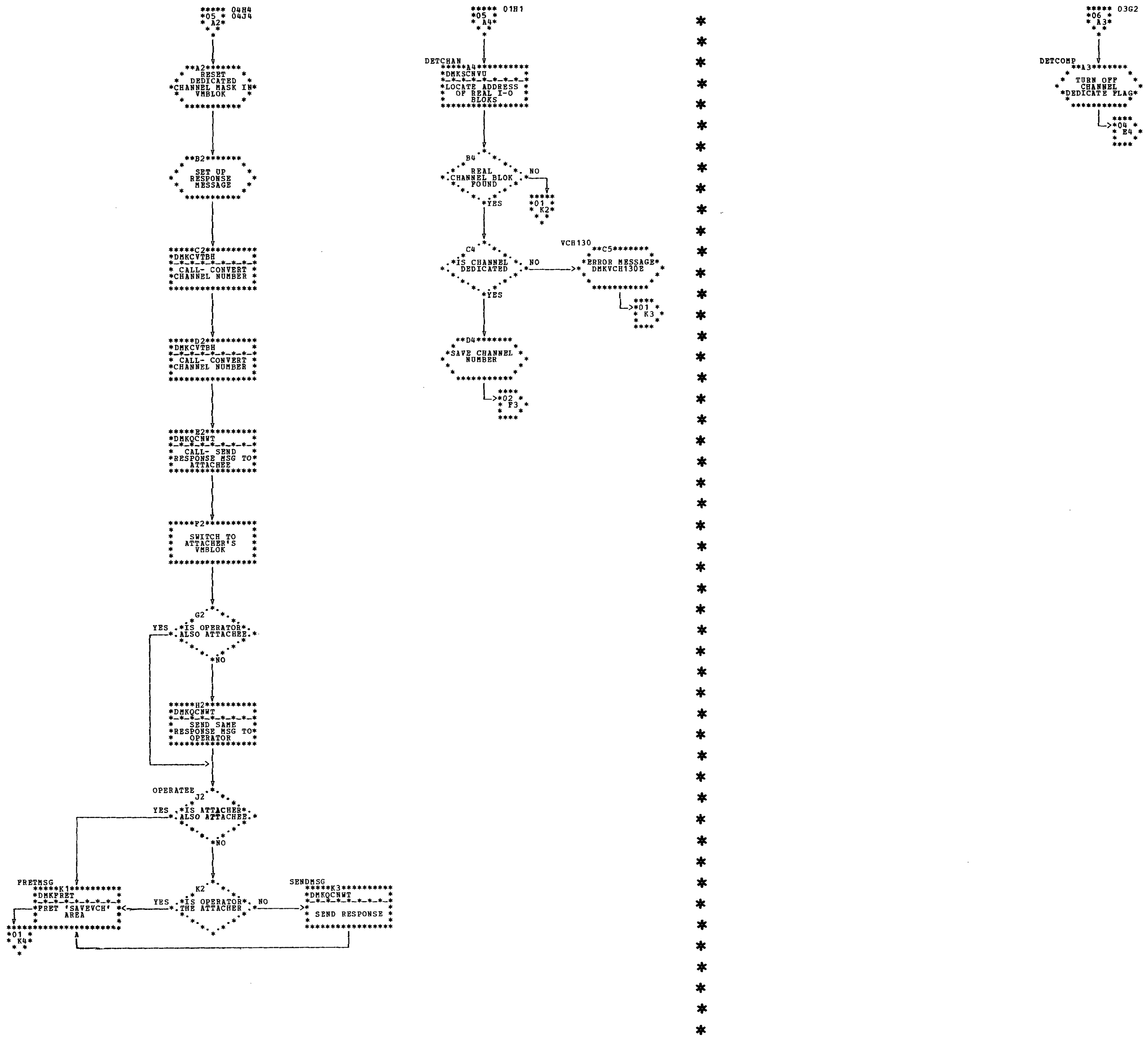


| DMKVCH -- Process ATTACH and DETACH Channel Commands (Parts 1 and 2 of 6)

| DMKVCH -- Process ATTACH and DETACH Channel Commands (Parts 3 and 4 of 6)

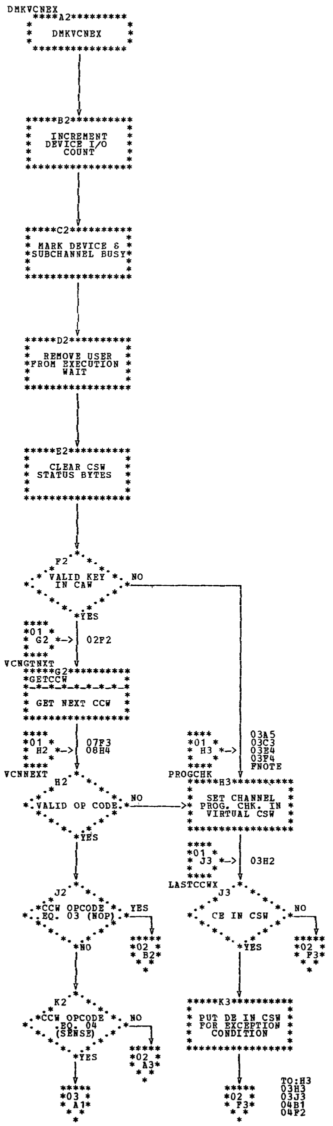


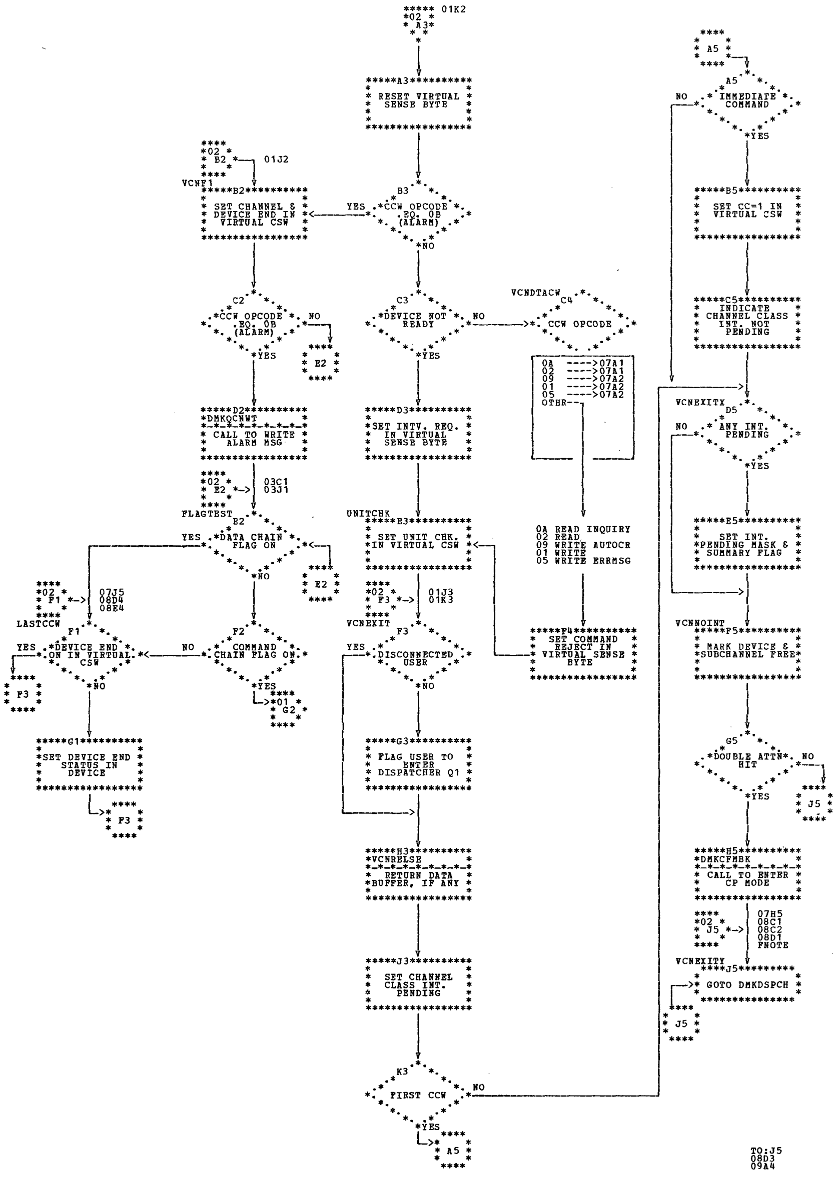


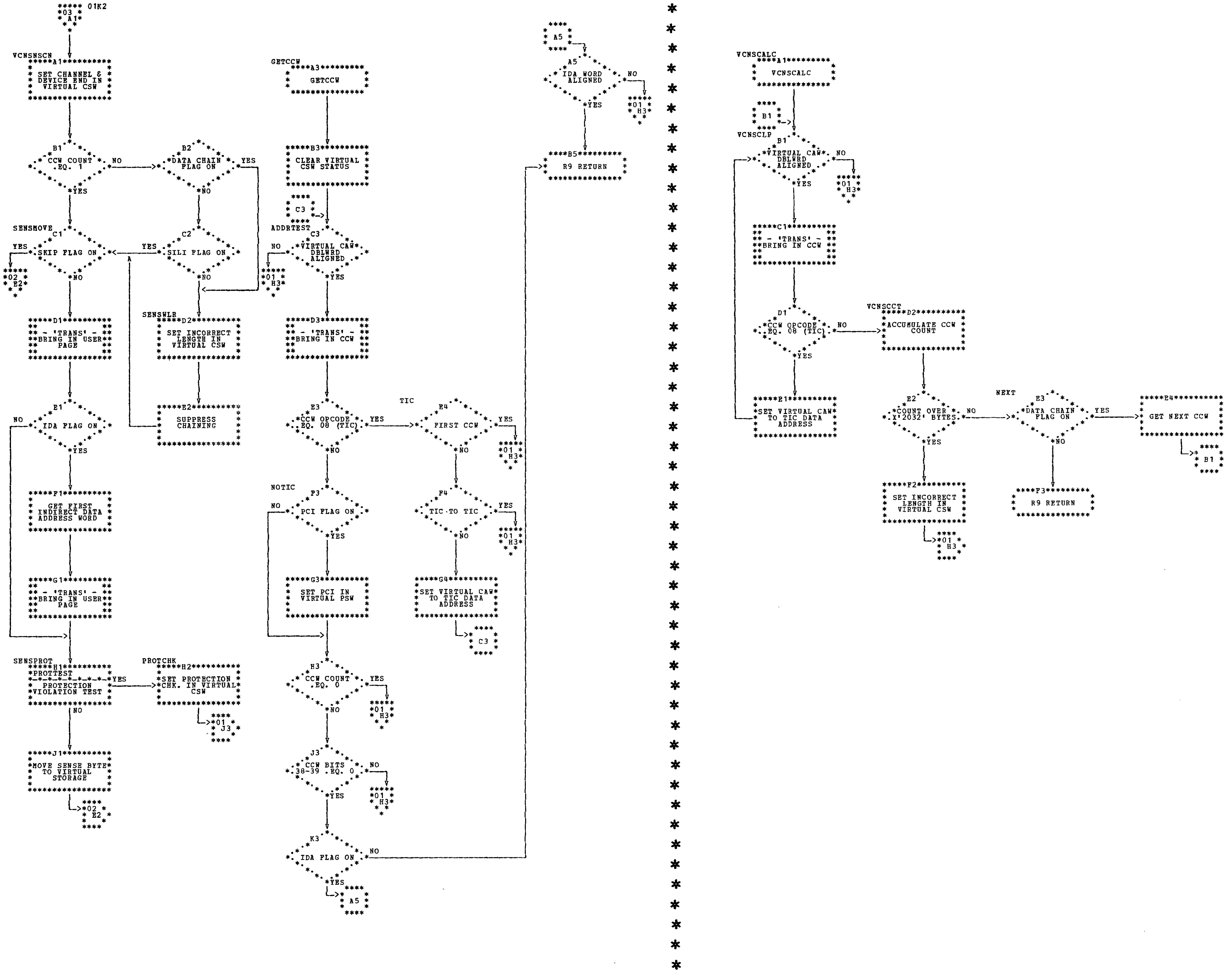


| DMKVCH -- Process ATTACH and DETACH Channel Commands (Parts 5 and 6 of 6)

| DMKVCN -- Console I/O Simulator (Parts 1 and 2 of 9)

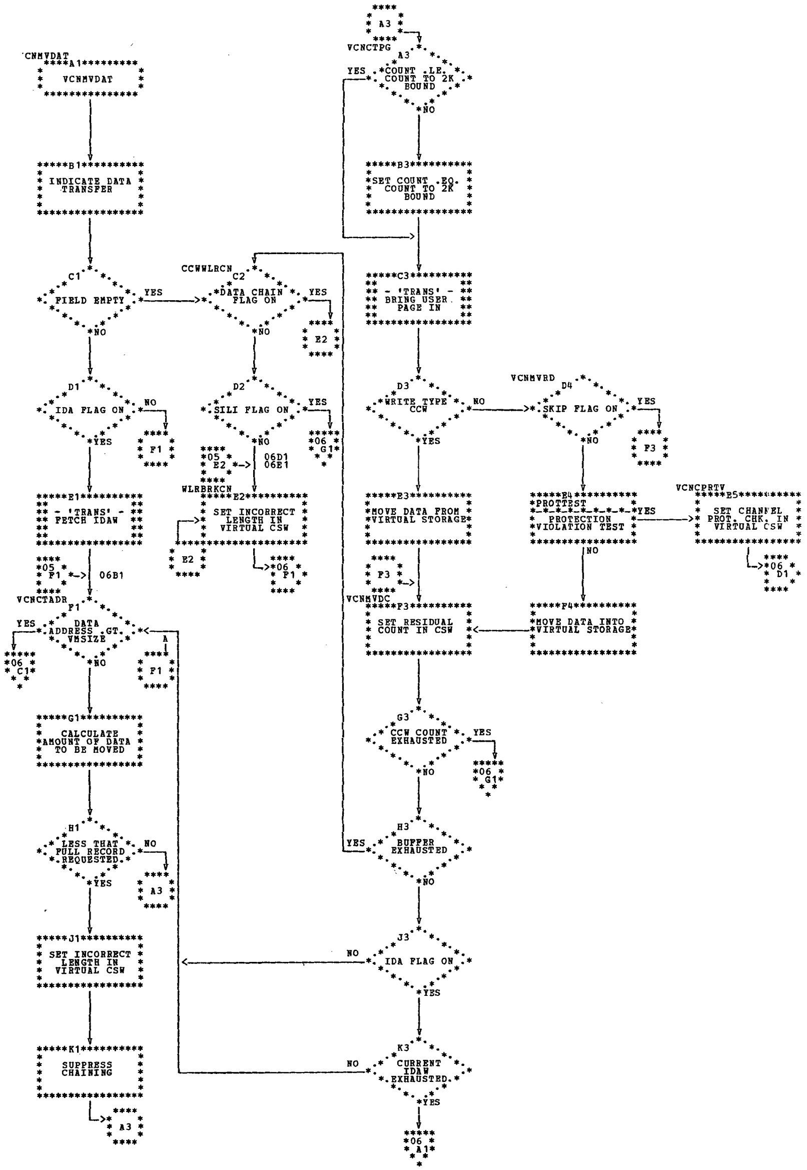


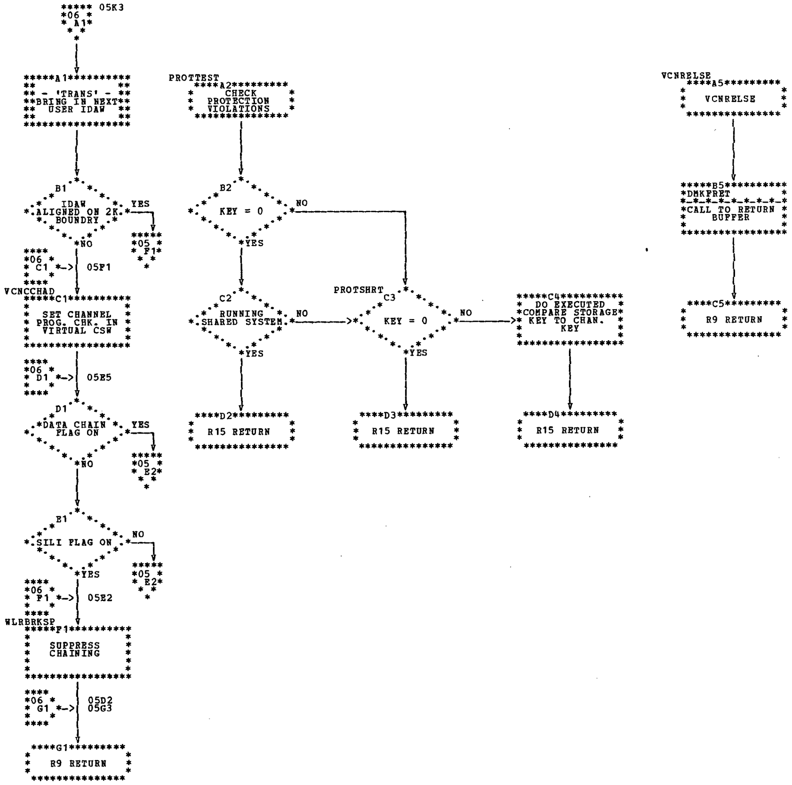


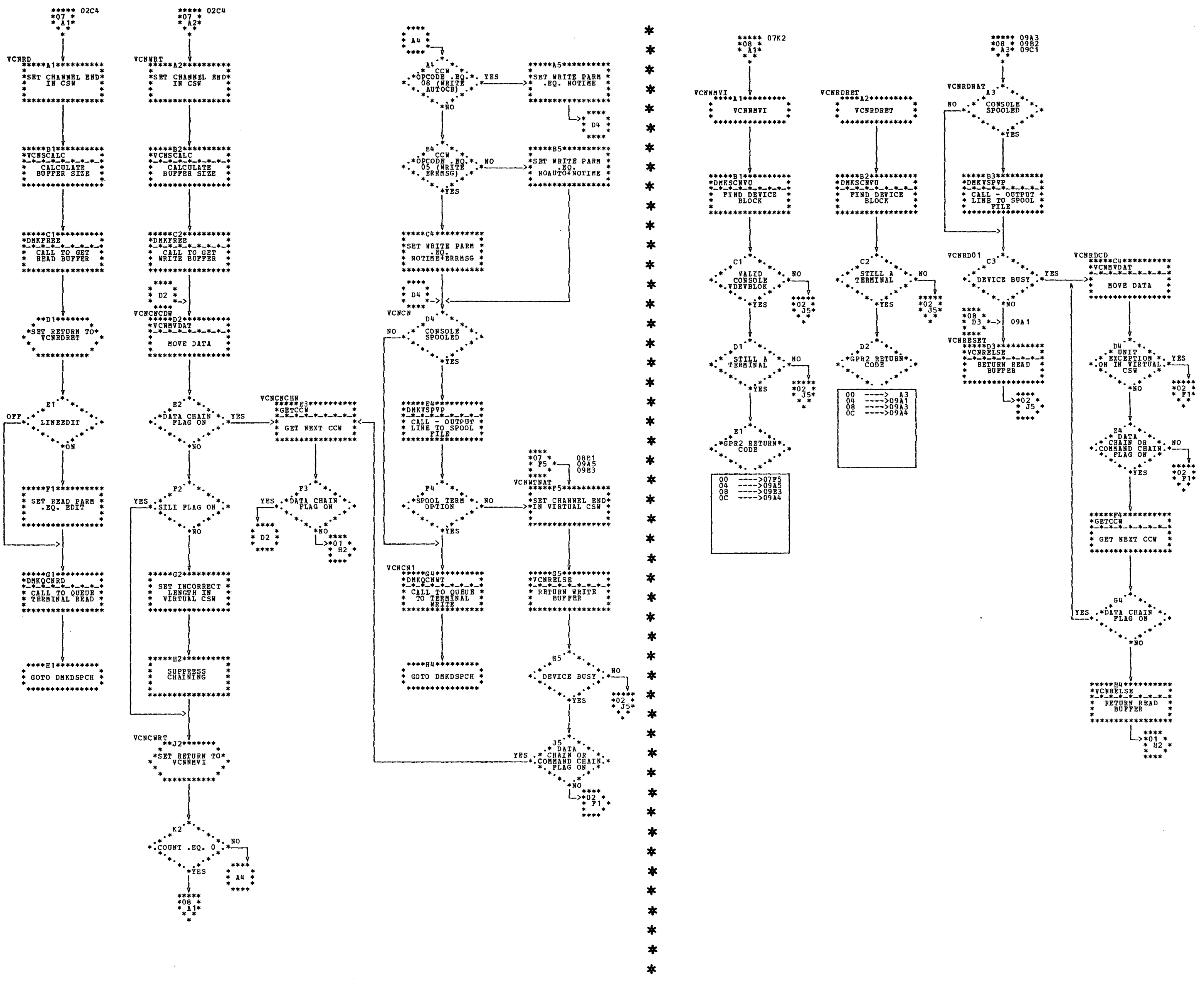

03 A1 01K2
VCNSNSCN A1
SET CHANNEL & DEVICE END IN VIRTUAL CSW
B1 CCW COUNT EQ. 1
NO
YES
B2 DATA CHAIN FLAG ON
YES
NO
SENSMOVE C1 SKIP FLAG ON
YES
02 B2
NO
C2 SILI FLAG ON
YES
A
D1 - 'TRANS' - BRING IN USER PAGE
SENSWLR D2 SET INCORRECT LENGTH IN VIRTUAL CSW
E2 SUPPRESS CHAINING
E1 IDA FLAG ON
NO
YES
F1 GET FIRST INDIRECT DATA ADDRESS WORD
G1 - 'TRANS' - BRING IN USER PAGE
SENSPROT H1 PROTTEST PROTECTION VIOLATION TEST
YES
NO
PROTCHK H2 SET PROTECTION CHK. IN VIRTUAL CSW
01 J3
J1 MOVE SENSE BYTE TO VIRTUAL STORAGE
02 E2
GETCCW A3 GETCCW
B3 CLEAR VIRTUAL CSW STATUS
C3
ADDRTEST C3 VIRTUAL CAW DBLWRD ALIGNED
NO
01 H3
YES
D3 - 'TRANS' - BRING IN CCW
E3 CCW OPCODE EQ. 08 (TIC)
YES
NO
TIC E4 FIRST CCW
YES
01 H3
NO
F4 TIC TO TIC
YES
01 H3
NO
G4 SET VIRTUAL CAW TO TIC DATA ADDRESS
C3
NOTIC F3 PCI FLAG ON
NO
YES
G3 SET PCI IN VIRTUAL PSW
H3 CCW COUNT EQ. 0
YES
01 H3
NO
J3 CCW BITS 38-39 EQ. 0
NO
01 H3
YES
K3 IDA FLAG ON
NO
YES
A5
A5
A5 IDA WORD ALIGNED
NO
01 H3
YES
B5 R9 RETURN
VCNSCALC A1 VCNSCALC
B1
VCNSCLP B1 VIRTUAL CAW DBLWRD ALIGNED
NO
01 H3
YES
C1 - 'TRANS' - BRING IN CCW
D1 CCW OPCODE EQ. 08 (TIC)
NO
YES
VCNSCCT D2 ACCUMULATE CCW COUNT
E1 SET VIRTUAL CAW TO TIC DATA ADDRESS
E2 COUNT OVER X'2032' BYTES
NO
YES
F2 SET INCORRECT LENGTH IN VIRTUAL CSW
01 H3
NEXT E3 DATA CHAIN FLAG ON
YES
NO
E4 GET NEXT CCW
B1
F3 R9 RETURN

| DMKVCN -- Console I/O Simulator (Parts 3 and 4 of 9)

| DMKVCN -- Console I/O Simulator (Parts 5 and 6 of 9)

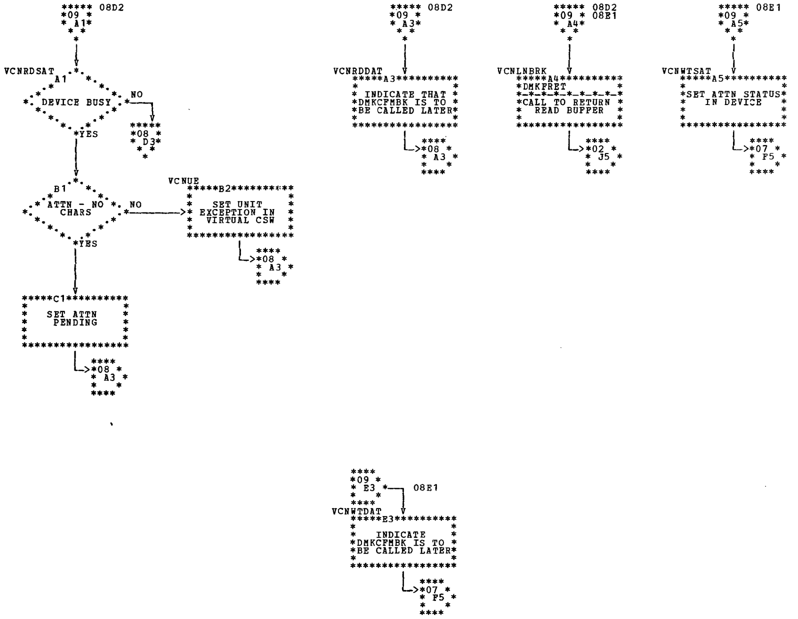


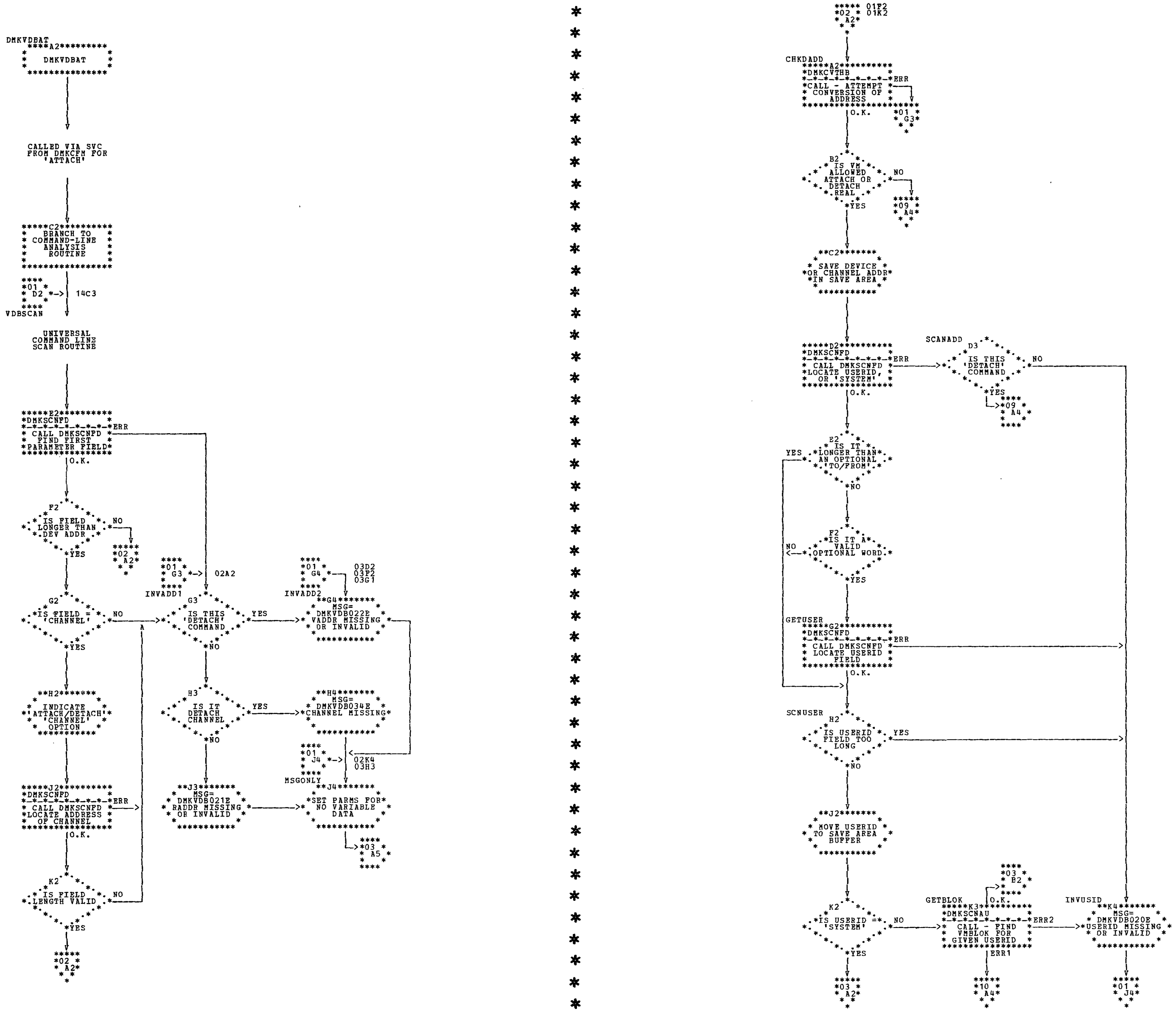




| DMKVCN -- Console I/O Simulator (Parts 7 and 8 of 9)

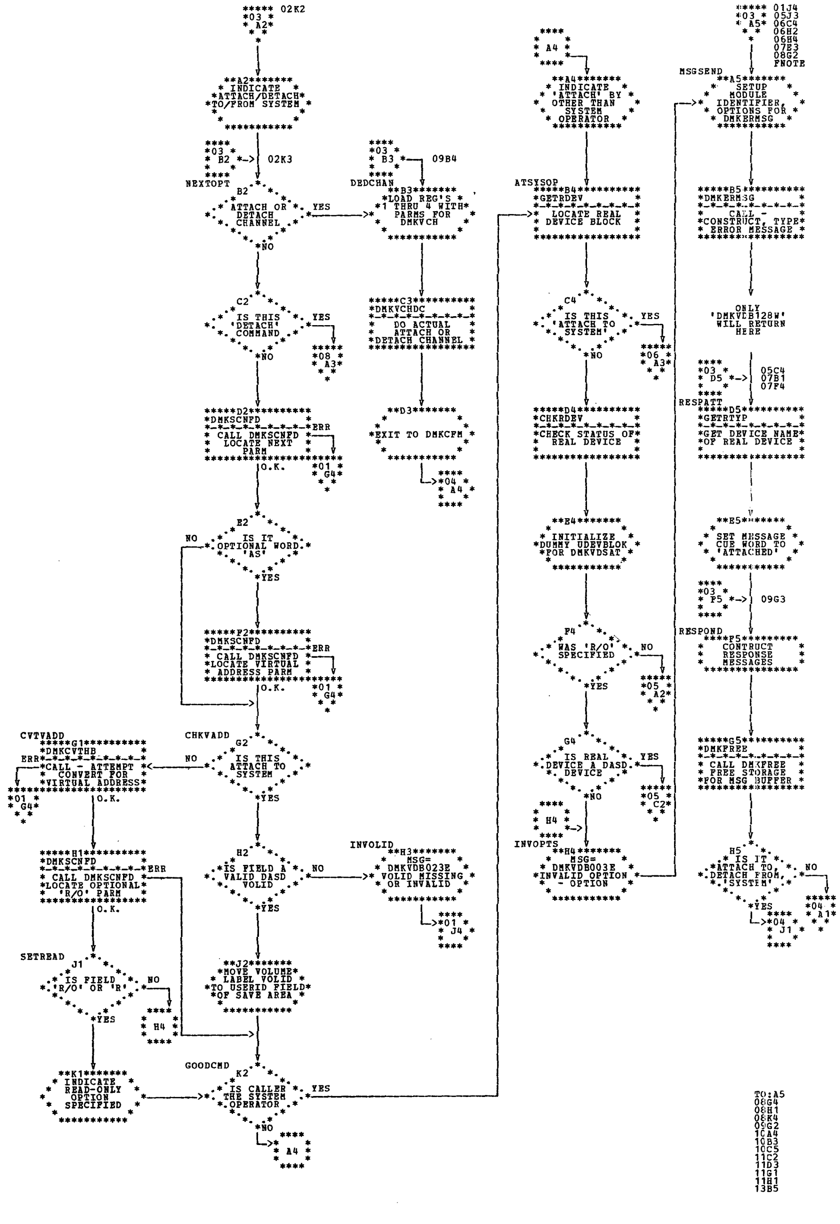
| DMKVCN -- Console I/O Simulator (Part 9 of 9)

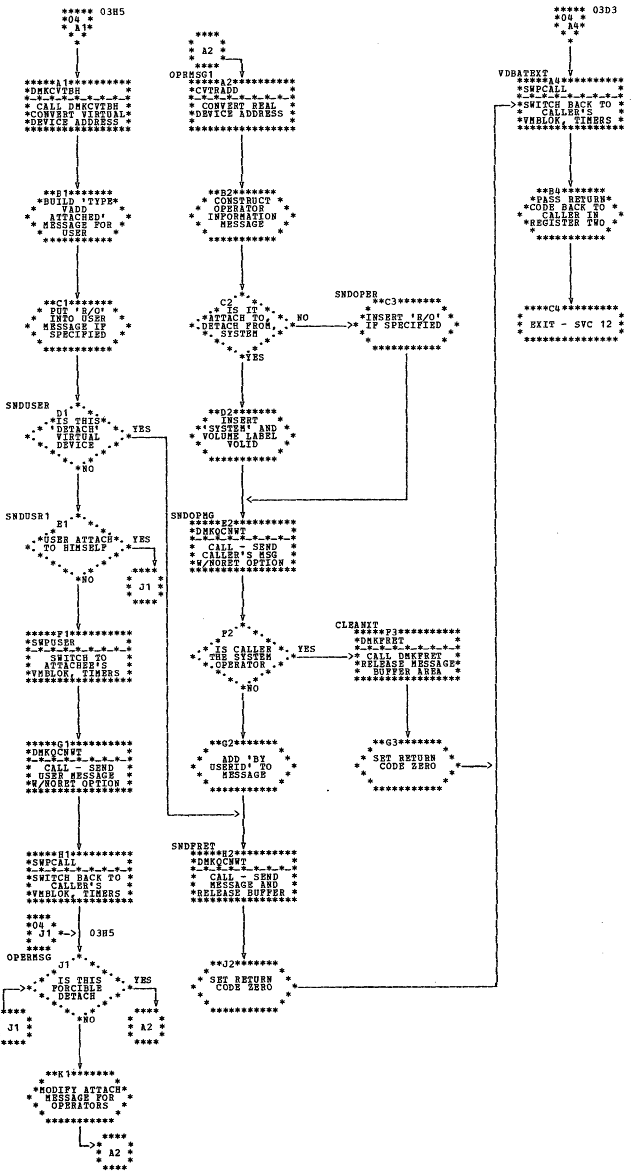


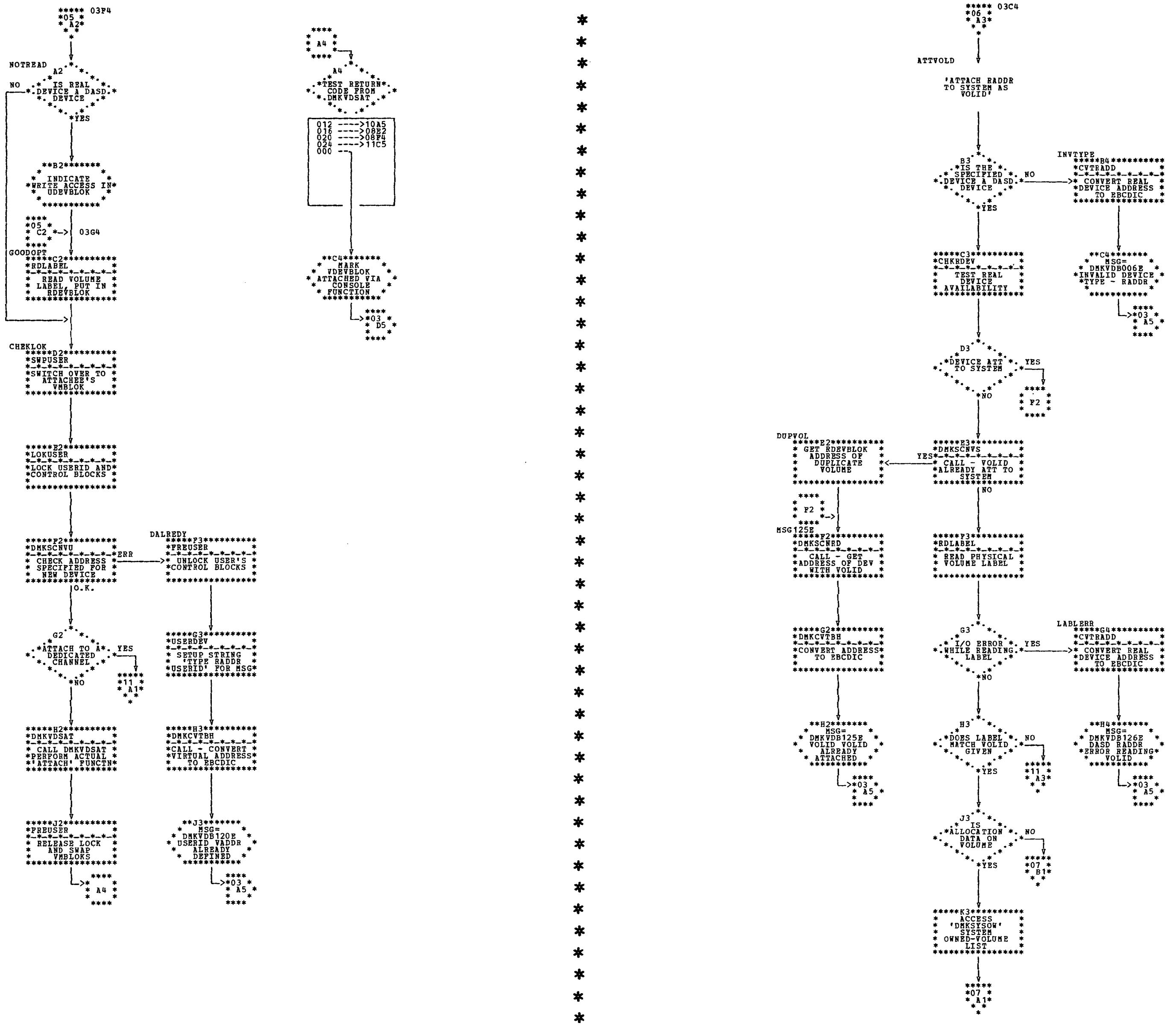

DMKVDBAT
A2
DMKVDBAT
CALLED VIA SVC FROM DMKCFM FOR 'ATTACH'
C2
BRANCH TO COMMAND-LINE ANALYSIS ROUTINE
01 D2
14C3
VDBSCAN
UNIVERSAL COMMAND LINE SCAN ROUTINE
E2
DMKSCNFD
ERR
CALL DMKSCNFD FIND FIRST PARAMETER FIELD
O.K.
F2
IS FIELD LONGER THAN DEV ADDR
NO
YES
02 A2
01 G3
02A2
INVADD1
G2
IS FIELD = 'CHANNEL'
NO
A
G3
IS THIS 'DETACH' COMMAND
YES
NO
01 G4
03D2
03F2
03G1
INVADD2
G4
MSG= DMKVDB022E VADDR MISSING OR INVALID
H2
INDICATE 'ATTACH/DETACH' 'CHANNEL' OPTION
H3
IS IT DETACH CHANNEL
YES
NO
H4
MSG= DMKVDB034E CHANNEL MISSING
01 J4
02K4
03H3
MSGONLY
J2
DMKSCNFD
ERR
CALL DMKSCNFD LOCATE ADDRESS OF CHANNEL
O.K.
J3
MSG= DMKVDB021E RADDR MISSING OR INVALID
J4
SET PARMS FOR NO VARIABLE DATA
03 A5
K2
IS FIELD LENGTH VALID
NO
YES
02 A2
02 A2
01F2
01K2
CHKDADD
A2
DMKCVTHB
ERR
CALL - ATTEMPT CONVERSION OF ADDRESS
O.K.
01 G3
B2
IS VM ALLOWED ATTACH OR DETACH REAL
NO
YES
09 A4
C2
SAVE DEVICE OR CHANNEL ADDR IN SAVE AREA
D2
DMKSCNFD
ERR
CALL DMKSCNFD LOCATE USERID, OR 'SYSTEM'
O.K.
SCANADD
D3
IS THIS 'DETACH' COMMAND
NO
YES
09 A4
E2
IS IT LONGER THAN AN OPTIONAL 'TO/FROM'
YES
NO
F2
IS IT A VALID OPTIONAL WORD
NO
YES
GETUSER
G2
DMKSCNFD
ERR
CALL DMKSCNFD LOCATE USERID FIELD
O.K.
SCNUSER
H2
IS USERID FIELD TOO LONG
YES
NO
J2
MOVE USERID TO SAVE AREA BUFFER
03 B2
GETBLOK
O.K.
K2
IS USERID = 'SYSTEM'
NO
YES
K3
DMKSCNAU
ERR2
CALL - FIND VMBLOK FOR GIVEN USERID
ERR1
INVUSID
K4
MSG= DMKVDB020E USERID MISSING OR INVALID
03 A2
10 A4
01 J4

| DMKVDB -- Process ATTACH and DETACH Commands (Parts 1 and 2 of 18)

| DMKVDB -- Process ATTACH and DETACH Commands (Parts 3 and 4 of 18)

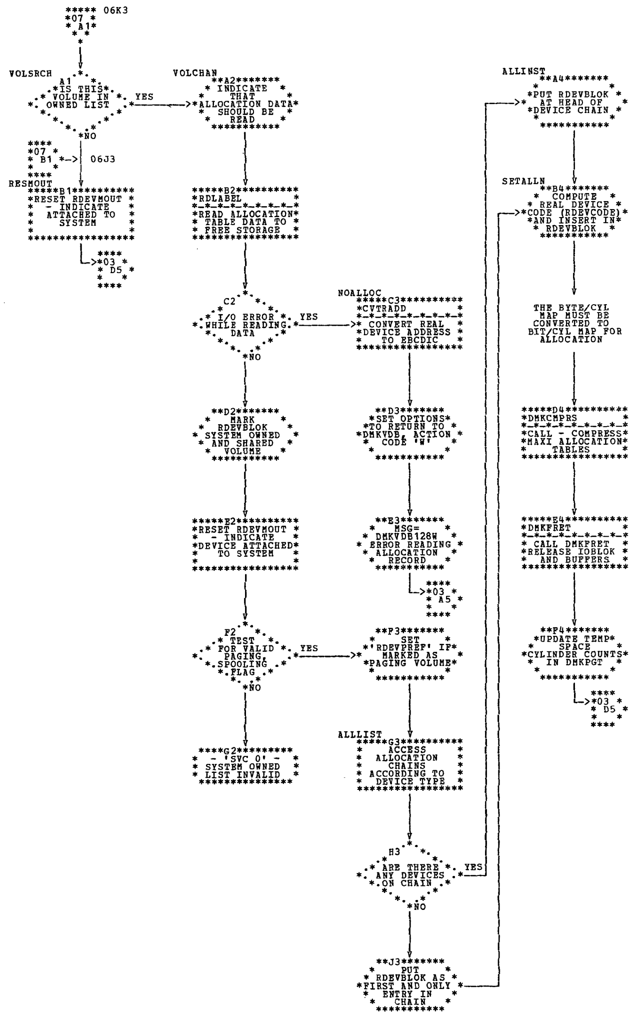

03 A2 02K2
A2 INDICATE ATTACH/DETACH TO/FROM SYSTEM
03 B2 02K3
NEXTOPT
B2 ATTACH OR DETACH CHANNEL
YES
NO
03 B3 09B4
DEDCHAN
B3 LOAD REG'S 1 THRU 4 WITH PARMS FOR DMKVCH
C2 IS THIS 'DETACH' COMMAND
YES
08 A3
NO
C3 DMKVCHDC DO ACTUAL ATTACH OR DETACH CHANNEL
D2 DMKSCNFD CALL DMKSCNFD LOCATE NEXT PARM
ERR
01 G4
O.K.
D3 EXIT TO DMKCFM
04 A4
E2 IS IT OPTIONAL WORD 'AS'
NO
YES
F2 DMKSCNFD CALL DMKSCNFD LOCATE VIRTUAL ADDRESS PARM
ERR
01 G4
O.K.
CVTVADD
G1 DMKCVTHB CALL - ATTEMPT CONVERT FOR VIRTUAL ADDRESS
ERR
01 G4
O.K.
CHKVADD
G2 IS THIS ATTACH TO SYSTEM
NO
YES
H1 DMKSCNFD CALL DMKSCNFD LOCATE OPTIONAL 'R/O' PARM
ERR
O.K.
H2 IS FIELD A VALID DASD VOLID
NO
YES
INVOLID
H3 MSG= DMKVDB023E VOLID MISSING OR INVALID
01 J4
SETREAD
J1 IS FIELD 'R/O' OR 'R'
NO
H4
YES
J2 MOVE VOLUME LABEL VOLID TO USERID FIELD OF SAVE AREA
K1 INDICATE READ-ONLY OPTION SPECIFIED
GOODCMD
K2 IS CALLER THE SYSTEM OPERATOR
YES
NO
A4
A4
A4 INDICATE 'ATTACH' BY OTHER THAN SYSTEM OPERATOR
ATSYSOP
B4 GETRDEV LOCATE REAL DEVICE BLOCK
C4 IS THIS 'ATTACH TO SYSTEM'
YES
06 A3
NO
D4 CHKRDEV CHECK STATUS OF REAL DEVICE
E4 INITIALIZE DUMMY UDEVBLOK FOR DMKVDSAT
F4 WAS 'R/O' SPECIFIED
NO
05 A2
YES
G4 IS REAL DEVICE A DASD DEVICE
YES
05 C2
NO
H4
INVOPTS
H4 MSG= DMKVDB003E INVALID OPTION - OPTION
03 A5 01J4 05J3 06C4 06H2 06H4 07E3 08G2 FNOTE
MSGSEND
A5 SETUP MODULE IDENTIFIER, OPTIONS FOR DMKERMSG
B5 DMKERMSG CALL - CONSTRUCT, TYPE ERROR MESSAGE
ONLY 'DMKVDB128W' WILL RETURN HERE
03 D5 05C4 07B1 07F4
RESPATT
D5 GETRTYP GET DEVICE NAME OF REAL DEVICE
E5 SET MESSAGE CUE WORD TO 'ATTACHED'
03 F5 09G3
RESPOND
F5 CONTRUCT RESPONSE MESSAGES
G5 DMKFREE CALL DMKFREE FREE STORAGE FOR MSG BUFFER
H5 IS IT ATTACH TO, DETACH FROM, 'SYSTEM'
NO
04 A1
YES
04 J1
TO:A5 08G4 08H1 08K4 09G2 10A4 10B3 10C5 11C2 11D3 11G1 11H1 13B5

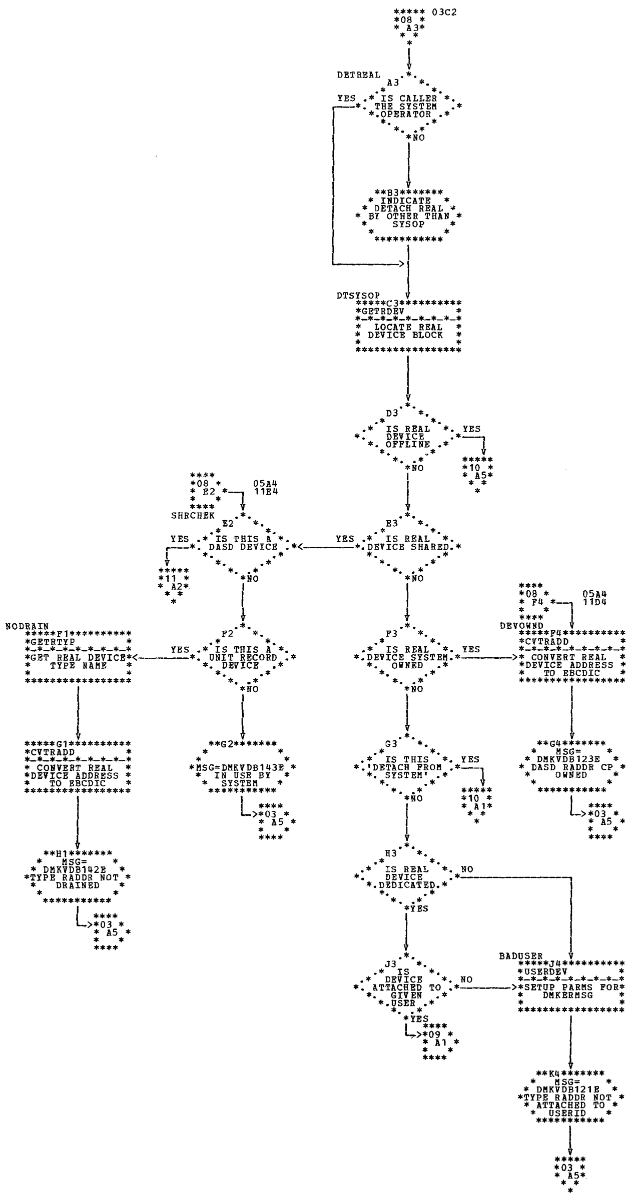

04 A1 03H5
A1 DMKCVTBH CALL DMKCVTBH CONVERT VIRTUAL DEVICE ADDRESS
B1 BUILD 'TYPE VADD ATTACHED' MESSAGE FOR USER
C1 PUT 'R/O' INTO USER MESSAGE IF SPECIFIED
SNDUSER
D1 IS THIS 'DETACH' VIRTUAL DEVICE
YES
NO
SNDUSR1
E1 USER ATTACH TO HIMSELF
YES
J1
NO
F1 SWPUSER SWITCH TO ATTACHEE'S VMBLOK, TIMERS
G1 DMKQCNWT CALL - SEND USER MESSAGE W/NORET OPTION
H1 SWPCALL SWITCH BACK TO CALLER'S VMBLOK, TIMERS
04 J1 03H5
OPERMSG
J1 IS THIS FORCIBLE DETACH
YES
J1
NO
A2
K1 MODIFY ATTACH MESSAGE FOR OPERATORS
A2
A2
OPRMSG1
A2 CVTRADD CONVERT REAL DEVICE ADDRESS
B2 CONSTRUCT OPERATOR INFORMATION MESSAGE
C2 IS IT ATTACH TO, DETACH FROM, SYSTEM
NO
YES
SNDOPER
C3 INSERT 'R/O' IF SPECIFIED
D2 INSERT 'SYSTEM' AND VOLUME LABEL VOLID
SNDOPMG
E2 DMKQCNWT CALL - SEND CALLER'S MSG W/NORET OPTION
F2 IS CALLER THE SYSTEM OPERATOR
YES
NO
CLEANXT
F3 DMKFRET CALL DMKFRET RELEASE MESSAGE BUFFER AREA
G2 ADD 'BY USERID' TO MESSAGE
G3 SET RETURN CODE ZERO
SNDFRET
H2 DMKQCNWT CALL - SEND MESSAGE AND RELEASE BUFFER
J2 SET RETURN CODE ZERO
04 A4 03D3
VDBATEXT
A4 SWPCALL SWITCH BACK TO CALLER'S VMBLOK, TIMERS
B4 PASS RETURN CODE BACK TO CALLER IN REGISTER TWO
C4 EXIT - SVC 12

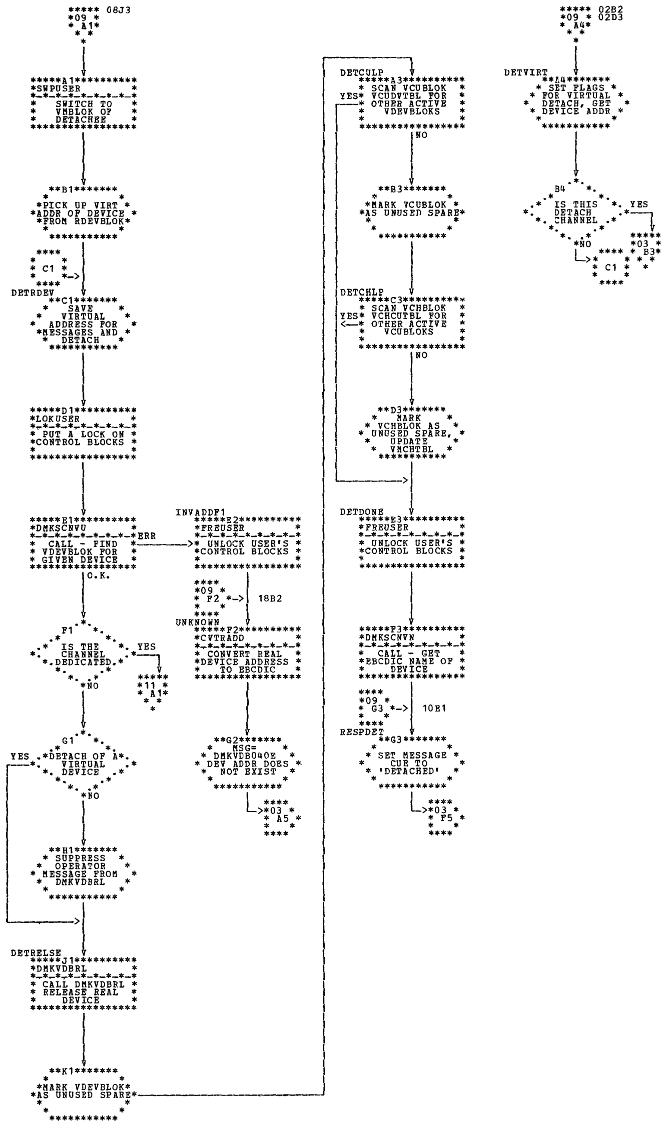

03F4
05 A2
NOTREAD
A2
IS REAL DEVICE A DASD DEVICE
NO
YES
B2
INDICATE WRITE ACCESS IN UDEVBLOK
05 C2
03G4
GOODOPT
C2
RDLABEL
READ VOLUME LABEL, PUT IN RDEVBLOK
CHEKLOK
D2
SWPUSER
SWITCH OVER TO ATTACHEE'S VMBLOK
E2
LOKUSER
LOCK USERID AND CONTROL BLOCKS
F2
DMKSCNVU
CHECK ADDRESS SPECIFIED FOR NEW DEVICE
ERR
O.K.
DALREDY
F3
FREUSER
UNLOCK USER'S CONTROL BLOCKS
G2
ATTACH TO A DEDICATED CHANNEL
YES
NO
11 A1
G3
USERDEV
SETUP STRING 'TYPE RADDR USERID' FOR MSG
H2
DMKVDSAT
CALL DMKVDSAT PERFORM ACTUAL 'ATTACH' FUNCTN
H3
DMKCVTBH
CALL - CONVERT VIRTUAL ADDRESS TO EBCDIC
J2
FREUSER
RELEASE LOCK AND SWAP VMBLOKS
A4
J3
MSG= DMKVDB120E USERID VADDR ALREADY DEFINED
03 A5
A4
A4
TEST RETURN CODE FROM DMKVDSAT
012 ---->10A5
016 ---->08E2
020 ---->08F4
024 ---->11C5
000
C4
MARK VDEVBLOK ATTACHED VIA CONSOLE FUNCTION
03 D5
03C4
06 A3
ATTVOLD
'ATTACH RADDR TO SYSTEM AS VOLID'
B3
IS THE SPECIFIED DEVICE A DASD DEVICE
NO
YES
INVTYPE
B4
CVTRADD
CONVERT REAL DEVICE ADDRESS TO EBCDIC
C3
CHKRDEV
TEST REAL DEVICE AVAILABILITY
C4
MSG= DMKVDB006E INVALID DEVICE TYPE - RADDR
03 A5
D3
DEVICE ATT TO SYSTEM
YES
F2
NO
DUPVOL
E2
GET RDEVBLOK ADDRESS OF DUPLICATE VOLUME
E3
DMKSCNVS
CALL - VOLID ALREADY ATT TO SYSTEM
YES
NO
F2
MSG125E
F2
DMKSCNRD
CALL - GET ADDRESS OF DEV WITH VOLID
F3
RDLABEL
READ PHYSICAL VOLUME LABEL
G2
DMKCVTBH
CONVERT ADDRESS TO EBCDIC
G3
I/O ERROR WHILE READING LABEL
YES
NO
LABLERR
G4
CVTRADD
CONVERT REAL DEVICE ADDRESS TO EBCDIC
H2
MSG= DMKVDB125E VOLID VOLID ALREADY ATTACHED
03 A5
H3
DOES LABEL MATCH VOLID GIVEN
NO
YES
11 A3
H4
MSG= DMKVDB126E DASD RADDR ERROR READING VOLID
03 A5
J3
IS ALLOCATION DATA ON VOLUME
NO
YES
07 B1
K3
ACCESS 'DMKSYSOW' SYSTEM OWNED-VOLUME LIST
07 A1

| DMKVDB -- Process ATTACH and DETACH Commands (Parts 5 and 6 of 18)

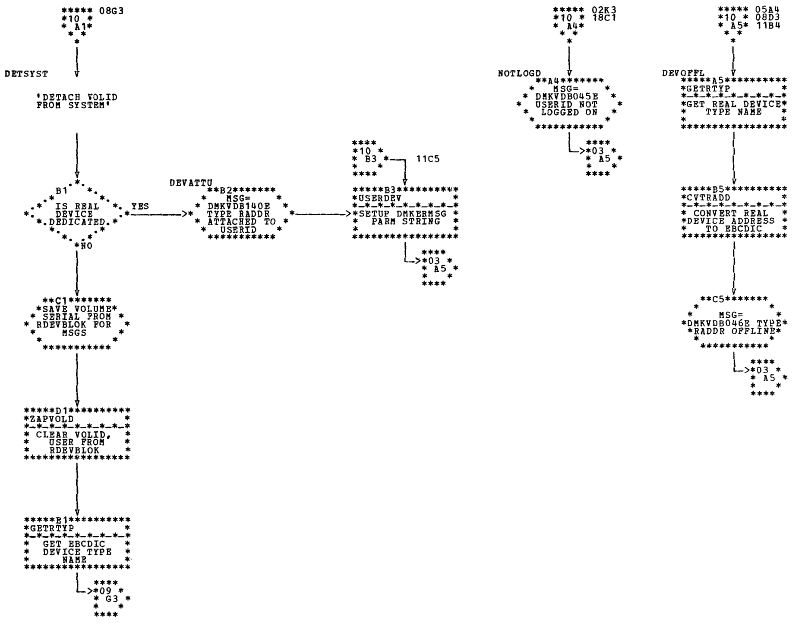
| DMKVDB -- Process ATTACH and DETACH Commands (Parts 7 and 8 of 18)


07 A1 06K3
VOLSRCH
A1 IS THIS VOLUME IN OWNED LIST
YES
NO
VOLCHAN
A2 INDICATE THAT ALLOCATION DATA SHOULD BE READ
07 B1 06J3
RESMOUT
B1 RESET RDEVMOUT - INDICATE ATTACHED TO SYSTEM
03 D5
B2 RDLABEL
READ ALLOCATION TABLE DATA TO FREE STORAGE
C2 I/O ERROR WHILE READING DATA
YES
NO
NOALLOC
C3 CVTRADD
CONVERT REAL DEVICE ADDRESS TO EBCDIC
D2 MARK RDEVBLOK SYSTEM OWNED AND SHARED VOLUME
D3 SET OPTIONS TO RETURN TO DMKVDB, ACTION CODE 'W'
E2 RESET RDEVMOUT - INDICATE DEVICE ATTACHED TO SYSTEM
E3 MSG= DMKVDB128W ERROR READING ALLOCATION RECORD
03 A5
F2 TEST FOR VALID PAGING, SPOOLING FLAG
YES
NO
F3 SET 'RDEVPREF' IF MARKED AS PAGING VOLUME
G2 - 'SVC 0' - SYSTEM OWNED LIST INVALID
ALLLIST
G3 ACCESS ALLOCATION CHAINS ACCORDING TO DEVICE TYPE
H3 ARE THERE ANY DEVICES ON CHAIN
YES
NO
J3 PUT RDEVBLOK AS FIRST AND ONLY ENTRY IN CHAIN
ALLINST
A4 PUT RDEVBLOK AT HEAD OF DEVICE CHAIN
SETALLN
B4 COMPUTE REAL DEVICE CODE (RDEVCODE) AND INSERT IN RDEVBLOK
THE BYTE/CYL MAP MUST BE CONVERTED TO BIT/CYL MAP FOR ALLOCATION
D4 DMKCMPRS
CALL - COMPRESS MAXI ALLOCATION TABLES
E4 DMKFRET
CALL DMKFRET RELEASE IOBLOK AND BUFFERS
F4 UPDATE TEMP SPACE CYLINDER COUNTS IN DMKPGT
03 D5


08 A3 03C2
DETREAL
A3 IS CALLER THE SYSTEM OPERATOR
YES
NO
B3 INDICATE DETACH REAL BY OTHER THAN SYSOP
DTSYSOP
C3 GETRDEV
LOCATE REAL DEVICE BLOCK
D3 IS REAL DEVICE OFFLINE
YES
10 A5
NO
08 E2 05A4 11E4
SHRCHEK
E2 IS THIS A DASD DEVICE
YES
11 A2
NO
E3 IS REAL DEVICE SHARED
YES
NO
NODRAIN
F1 GETRTYP
GET REAL DEVICE TYPE NAME
F2 IS THIS A UNIT RECORD DEVICE
YES
NO
F3 IS REAL DEVICE SYSTEM OWNED
YES
NO
08 F4 05A4 11D4
DEVOWND
F4 CVTRADD
CONVERT REAL DEVICE ADDRESS TO EBCDIC
G1 CVTRADD
CONVERT REAL DEVICE ADDRESS TO EBCDIC
G2 MSG=DMKVDB143E IN USE BY SYSTEM
03 A5
G3 IS THIS 'DETACH FROM SYSTEM'
YES
10 A1
NO
G4 MSG= DMKVDB123E DASD RADDR CP OWNED
03 A5
H1 MSG= DMKVDB142E TYPE RADDR NOT DRAINED
03 A5
H3 IS REAL DEVICE DEDICATED
NO
YES
J3 IS DEVICE ATTACHED TO GIVEN USER
NO
YES
09 A1
BADUSER
J4 USERDEV
SETUP PARMS FOR DMKERMSG
K4 MSG= DMKVDB121E TYPE RADDR NOT ATTACHED TO USERID
03 A5



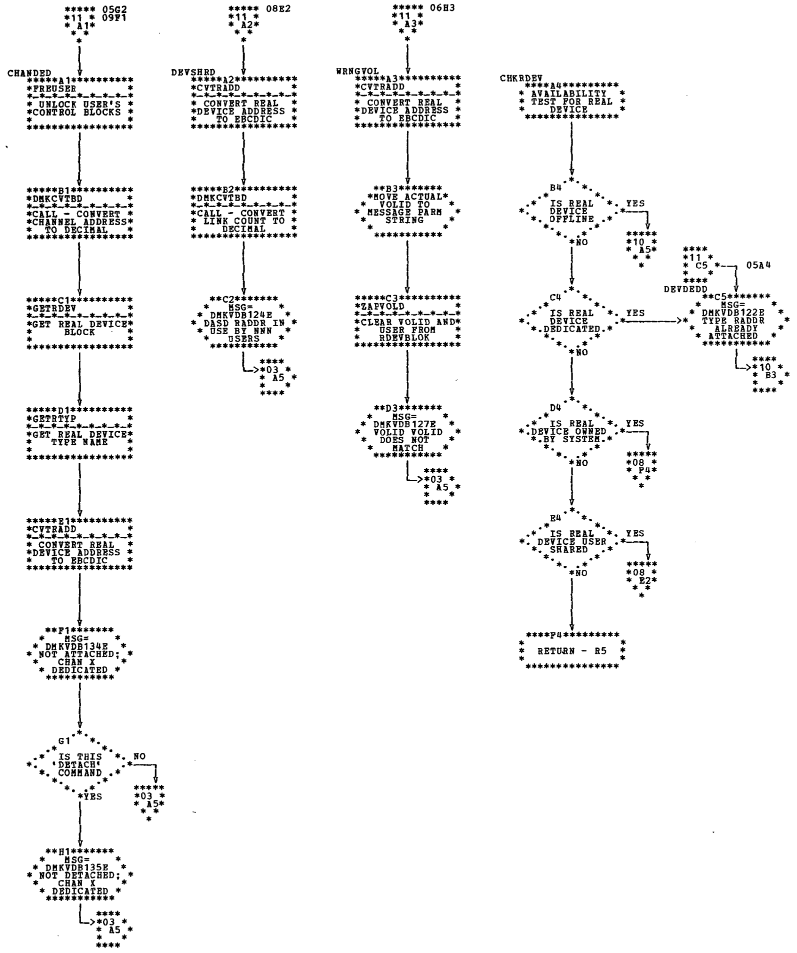
*09 * A1* 08J3
A1 SWPUSER SWITCH TO VMBLOK OF DETACHEE
B1 PICK UP VIRT ADDR OF DEVICE FROM RDEVBLOK
C1
DETRDEV
C1 SAVE VIRTUAL ADDRESS FOR MESSAGES AND DETACH
D1 LOKUSER PUT A LOCK ON CONTROL BLOCKS
E1 DMKSCNVU CALL - FIND VDEVBLOK FOR GIVEN DEVICE
ERR
O.K.
INVADDF1
E2 FREUSER UNLOCK USER'S CONTROL BLOCKS
*09 * F2 18B2
UNKNOWN
F2 CVTRADD CONVERT REAL DEVICE ADDRESS TO EBCDIC
G2 MSG= DMKVDB040E DEV ADDR DOES NOT EXIST
*03 * A5
F1 IS THE CHANNEL DEDICATED
YES
NO
*11 * A1
G1 DETACH OF A VIRTUAL DEVICE
YES
NO
H1 SUPPRESS OPERATOR MESSAGE FROM DMKVDBRL
DETRELSE
J1 DMKVDBRL CALL DMKVDBRL RELEASE REAL DEVICE
K1 MARK VDEVBLOK AS UNUSED SPARE
DETCULP
A3 SCAN VCUBLOK VCUDVTBL FOR OTHER ACTIVE VDEVBLOKS
YES
NO
B3 MARK VCUBLOK AS UNUSED SPARE
DETCHLP
C3 SCAN VCHBLOK VCHCUTBL FOR OTHER ACTIVE VCUBLOKS
YES
NO
D3 MARK VCHBLOK AS UNUSED SPARE, UPDATE VMCHTBL
DETDONE
E3 FREUSER UNLOCK USER'S CONTROL BLOCKS
F3 DMKSCNVN CALL - GET EBCDIC NAME OF DEVICE
*09 * G3 10E1
RESPDET
G3 SET MESSAGE CUE TO 'DETACHED'
*03 * F5
*09 * A4* 02B2 02D3
DETVIRT
A4 SET FLAGS FOR VIRTUAL DETACH, GET DEVICE ADDR
B4 IS THIS DETACH CHANNEL
YES
NO
*03 * B3
C1



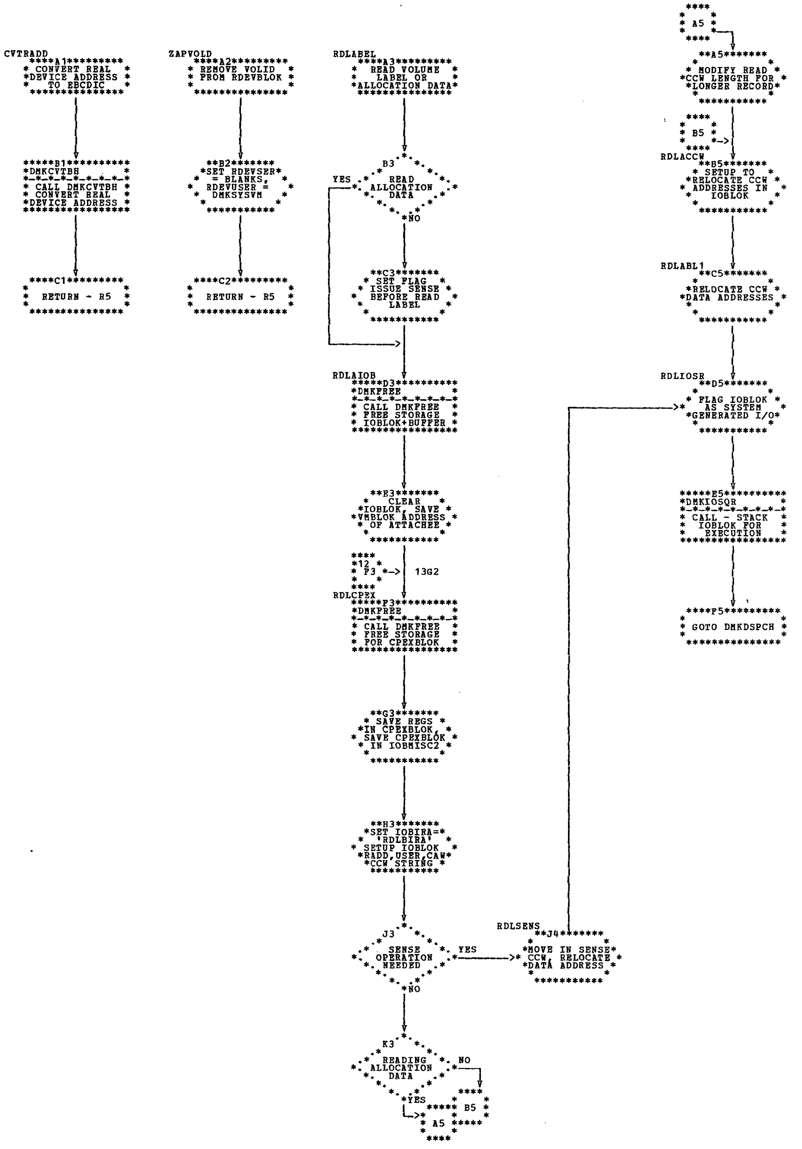
*10 * A1* 08G3
DETSYST
'DETACH VOLID FROM SYSTEM'
B1 IS REAL DEVICE DEDICATED
YES
NO
DEVATTU
B2 MSG= DMKVDB140E TYPE RADDR ATTACHED TO USERID
*10 * B3 11C5
B3 USERDEV SETUP DMKERMSG PARM STRING
*03 * A5
C1 SAVE VOLUME SERIAL FROM RDEVBLOK FOR MSGS
D1 ZAPVOLD CLEAR VOLID, USER FROM RDEVBLOK
E1 GETRTYP GET EBCDIC DEVICE TYPE NAME
*09 * G3
*10 * A4* 02K3 18C1
NOTLOGD
A4 MSG= DMKVDB045E USERID NOT LOGGED ON
*03 * A5
*10 * A5* 05A4 08D3 11B4
DEVOFFL
A5 GETRTYP GET REAL DEVICE TYPE NAME
B5 CVTRADD CONVERT REAL DEVICE ADDRESS TO EBCDIC
C5 MSG= DMKVDB046E TYPE RADDR OFFLINE
*03 * A5

| DMKVDB -- Process ATTACH and DETACH Commands (Parts 9 and 10 of 18)

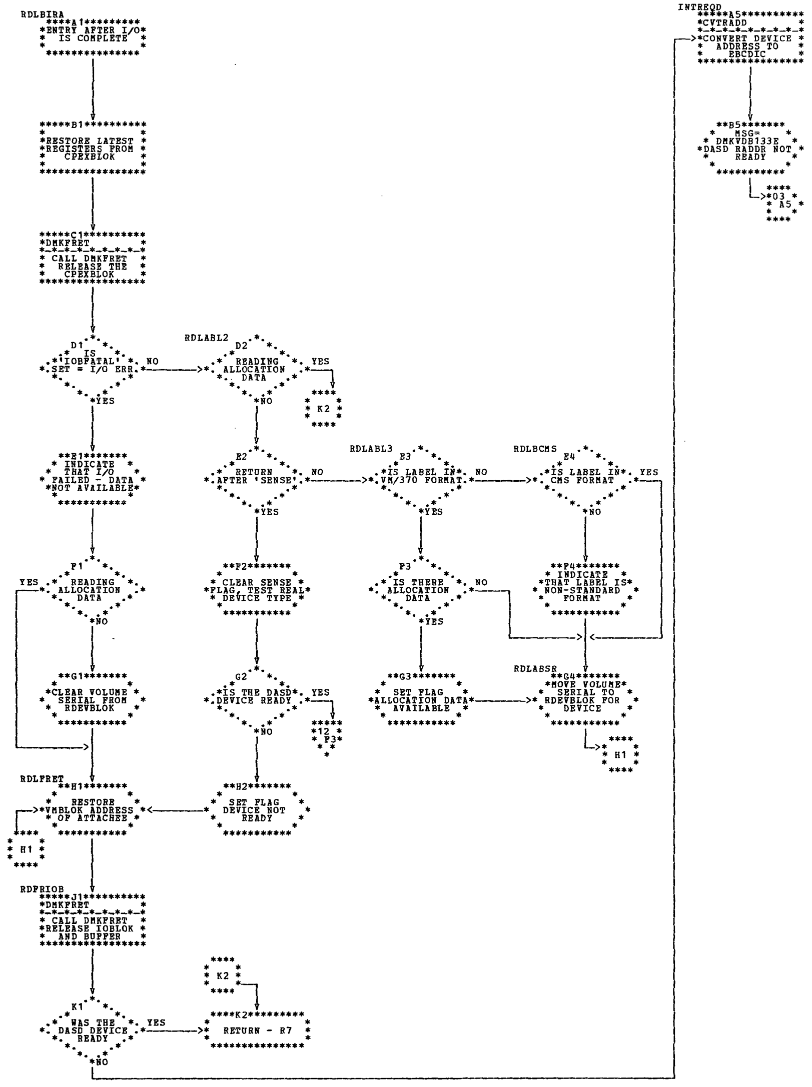
| DMKVDB -- Process ATTACH and DETACH Commands (Parts 11 and 12 of 18)

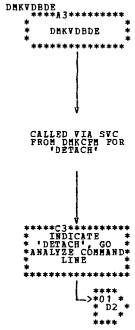


11 A1 05G2 09F1
CHANDED A1 FREUSER UNLOCK USER'S CONTROL BLOCKS
B1 DMKCVTBD CALL - CONVERT CHANNEL ADDRESS TO DECIMAL
C1 GETRDEV GET REAL DEVICE BLOCK
D1 GETRTYP GET REAL DEVICE TYPE NAME
E1 CVTRADD CONVERT REAL DEVICE ADDRESS TO EBCDIC
F1 MSG= DMKVDB134E NOT ATTACHED; CHAN X DEDICATED
G1 IS THIS 'DETACH' COMMAND
NO
03 A5
YES
H1 MSG= DMKVDB135E NOT DETACHED; CHAN X DEDICATED
03 A5
11 A2 08E2
DEVSHRD A2 CVTRADD CONVERT REAL DEVICE ADDRESS TO EBCDIC
B2 DMKCVTBD CALL - CONVERT LINK COUNT TO DECIMAL
C2 MSG= DMKVDB124E DASD RADDR IN USE BY NNN USERS
03 A5
11 A3 06H3
WRNGVOL A3 CVTRADD CONVERT REAL DEVICE ADDRESS TO EBCDIC
B3 MOVE ACTUAL VOLID TO MESSAGE PARM STRING
C3 ZAPVOLD CLEAR VOLID AND USER FROM RDEVBLOK
D3 MSG= DMKVDB127E VOLID VOLID DOES NOT MATCH
03 A5
CHKRDEV A4 AVAILABILITY TEST FOR REAL DEVICE
B4 IS REAL DEVICE OFFLINE
YES
10 A5
NO
C4 IS REAL DEVICE DEDICATED
YES
NO
11 C5 05A4
DEVDEDD C5 MSG= DMKVDB122E TYPE RADDR ALREADY ATTACHED
10 B3
D4 IS REAL DEVICE OWNED BY SYSTEM
YES
08 F4
NO
E4 IS REAL DEVICE USER SHARED
YES
08 E2
NO
F4 RETURN - R5



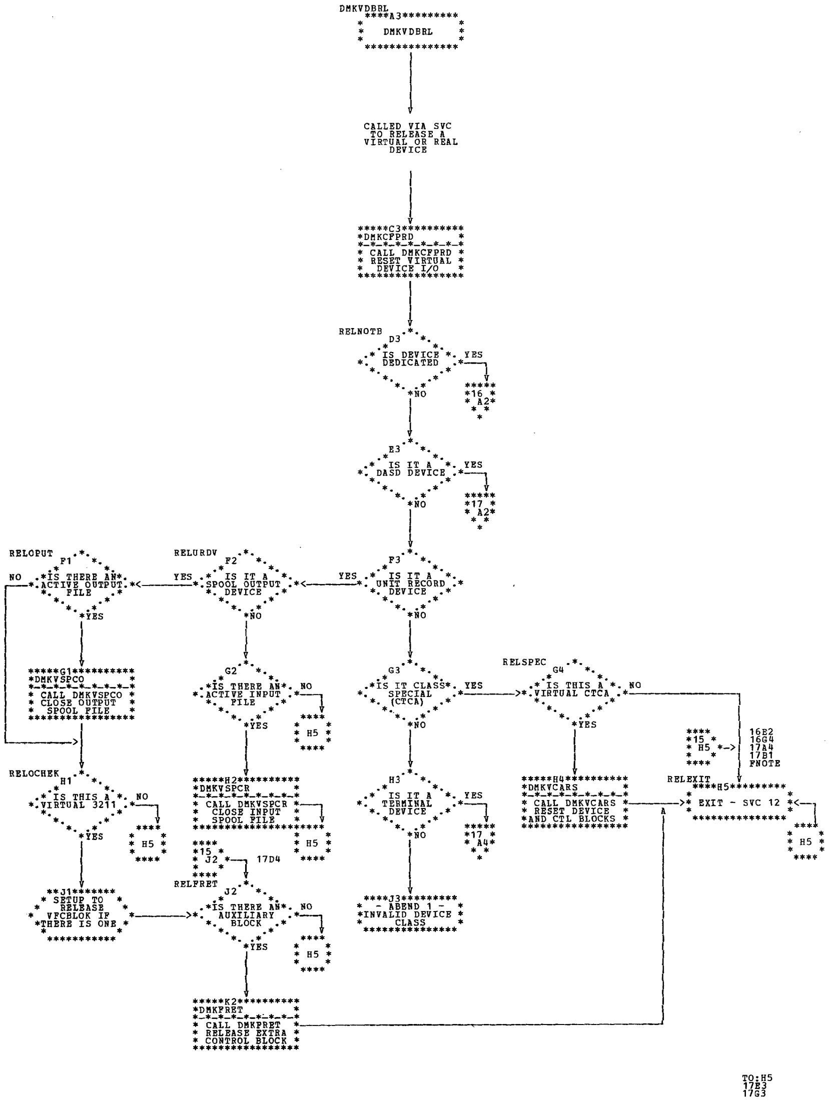
CVTRADD A1 CONVERT REAL DEVICE ADDRESS TO EBCDIC
B1 DMKCVTBH CALL DMKCVTBH CONVERT REAL DEVICE ADDRESS
C1 RETURN - R5
ZAPVOLD A2 REMOVE VOLID FROM RDEVBLOK
B2 SET RDEVSER = BLANKS, RDEVUSER = DMKSYSVM
C2 RETURN - R5
RDLABEL A3 READ VOLUME LABEL OR ALLOCATION DATA
B3 READ ALLOCATION DATA
YES
NO
C3 SET FLAG ISSUE SENSE BEFORE READ LABEL
RDLAIOB D3 DMKFREE CALL DMKFREE FREE STORAGE IOBLOK+BUFFER
E3 CLEAR IOBLOK, SAVE VMBLOK ADDRESS OF ATTACHEE
12 F3 13G2
RDLCPEX F3 DMKFREE CALL DMKFREE FREE STORAGE FOR CPEXBLOK
G3 SAVE REGS IN CPEXBLOK, SAVE CPEXBLOK IN IOBMISC2
H3 SET IOBIRA= 'RDLBIRA' SETUP IOBLOK RADD,USER,CAW CCW STRING
J3 SENSE OPERATION NEEDED
YES
NO
RDLSENS J4 MOVE IN SENSE CCW, RELOCATE DATA ADDRESS
K3 READING ALLOCATION DATA
NO
YES
B5
A5
A5
A5 MODIFY READ CCW LENGTH FOR LONGER RECORD
B5
RDLACCW B5 SETUP TO RELOCATE CCW ADDRESSES IN IOBLOK
RDLABL1 C5 RELOCATE CCW DATA ADDRESSES
RDLIOSR D5 FLAG IOBLOK AS SYSTEM GENERATED I/O
E5 DMKIOSQR CALL - STACK IOBLOK FOR EXECUTION
F5 GOTO DMKDSPCH

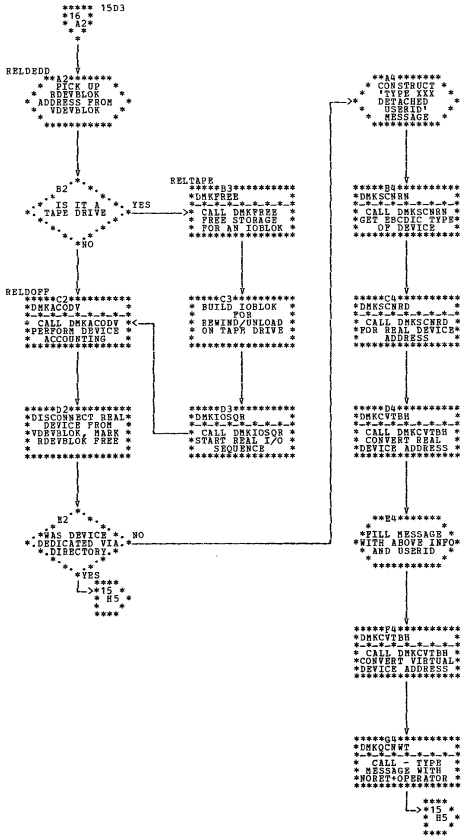

RDLBIRA
A1
ENTRY AFTER I/O IS COMPLETE
B1
RESTORE LATEST REGISTERS FROM CPEXBLOK
C1
DMKFRET
CALL DMKFRET RELEASE THE CPEXBLOK
D1
IS 'IOBFATAL' SET = I/O ERR
NO
YES
RDLABL2
D2
READING ALLOCATION DATA
YES
NO
K2
E1
INDICATE THAT I/O FAILED - DATA NOT AVAILABLE
E2
RETURN AFTER 'SENSE'
NO
YES
RDLABL3
E3
IS LABEL IN VM/370 FORMAT
NO
YES
RDLBCMS
E4
IS LABEL IN CMS FORMAT
YES
NO
F1
READING ALLOCATION DATA
YES
NO
F2
CLEAR SENSE FLAG, TEST REAL DEVICE TYPE
F3
IS THERE ALLOCATION DATA
NO
YES
F4
INDICATE THAT LABEL IS NON-STANDARD FORMAT
G1
CLEAR VOLUME SERIAL FROM RDEVBLOK
G2
IS THE DASD DEVICE READY
YES
NO
12
F3
G3
SET FLAG ALLOCATION DATA AVAILABLE
RDLABSR
G4
MOVE VOLUME SERIAL TO RDEVBLOK FOR DEVICE
H1
RDLFRET
H1
RESTORE VMBLOK ADDRESS OF ATTACHEE
H1
H2
SET FLAG DEVICE NOT READY
RDFRIOB
J1
DMKFRET
CALL DMKFRET RELEASE IOBLOK AND BUFFER
K2
K1
WAS THE DASD DEVICE READY
YES
NO
K2
RETURN - R7
INTREQD
A5
CVTRADD
CONVERT DEVICE ADDRESS TO EBCDIC
B5
MSG= DMKVDB133E DASD RADDR NOT READY
03
A5

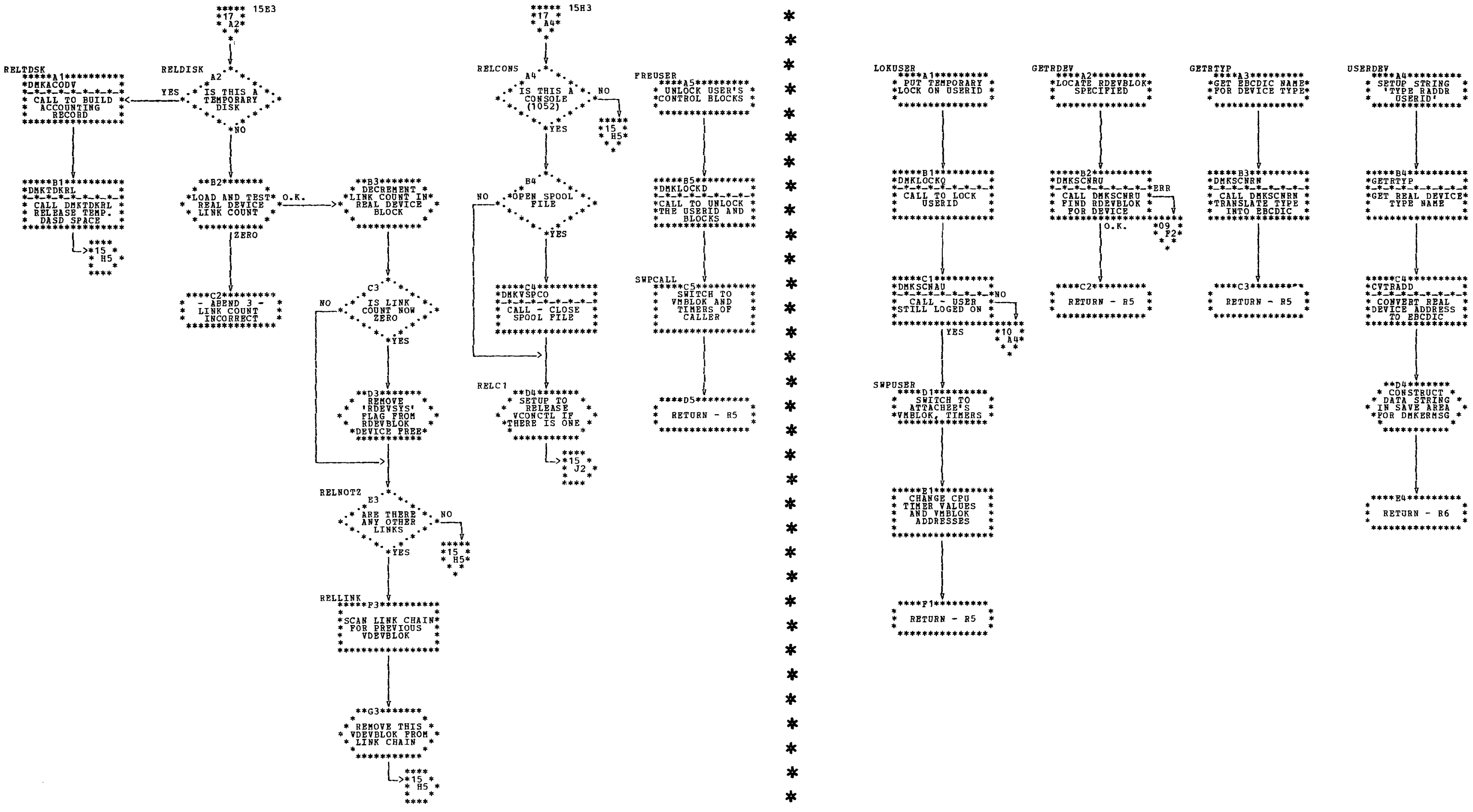

DMKVDBDE
A3
DMKVDBDE
CALLED VIA SVC FROM DMKCFM FOR 'DETACH'
C3
INDICATE 'DETACH', GO ANALYZE COMMAND LINE
01
D2

| DMKVDB -- Process ATTACH and DETACH Commands (Parts 13 and 14 of 18)

| DMKVDB -- Process ATTACH and DETACH Commands (Parts 15 and 16 of 18)

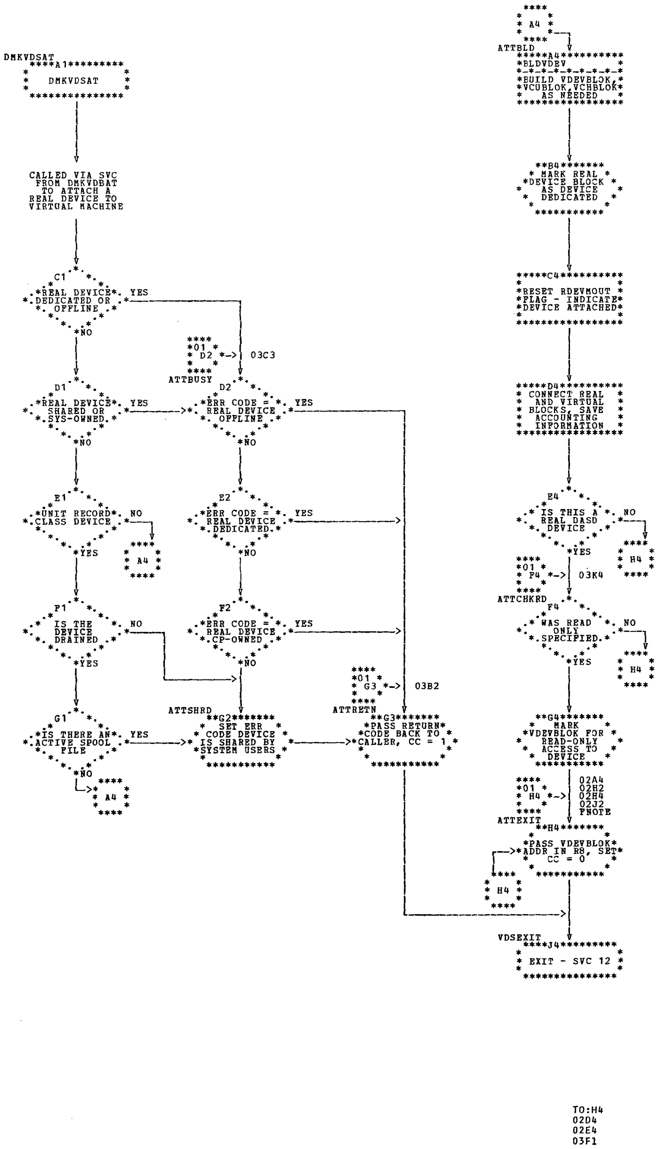

DMKVDBRL
A3
DMKVDBRL
CALLED VIA SVC TO RELEASE A VIRTUAL OR REAL DEVICE
C3
DMKCFPRD
CALL DMKCFPRD RESET VIRTUAL DEVICE I/O
RELNOTB
D3
IS DEVICE DEDICATED
YES
NO
16 A2
E3
IS IT A DASD DEVICE
YES
NO
17 A2
RELOPUT
F1
IS THERE AN ACTIVE OUTPUT FILE
NO
YES
RELURDV
F2
IS IT A SPOOL OUTPUT DEVICE
YES
NO
F3
IS IT A UNIT RECORD DEVICE
YES
NO
G1
DMKVSPCO
CALL DMKVSPCO CLOSE OUTPUT SPOOL FILE
G2
IS THERE AN ACTIVE INPUT FILE
NO
YES
H5
G3
IS IT CLASS SPECIAL (CTCA)
YES
NO
RELSPEC
G4
IS THIS A VIRTUAL CTCA
NO
YES
15 H5
16E2
16G4
17A4
17B1
FNOTE
RELOCHEK
H1
IS THIS A VIRTUAL 3211
NO
YES
H5
H2
DMKVSPCR
CALL DMKVSPCR CLOSE INPUT SPOOL FILE
H5
15 J2
17D4
H3
IS IT A TERMINAL DEVICE
YES
NO
17 A4
H4
DMKVCARS
CALL DMKVCARS RESET DEVICE AND CTL BLOCKS
RELEXIT
H5
EXIT - SVC 12
A
H5
J1
SETUP TO RELEASE VFCBLOK IF THERE IS ONE
RELFRET
J2
IS THERE AN AUXILIARY BLOCK
NO
YES
H5
J3
- ABEND 1 - INVALID DEVICE CLASS
K2
DMKFRET
CALL DMKFRET RELEASE EXTRA CONTROL BLOCK
TO:H5
17E3
17G3

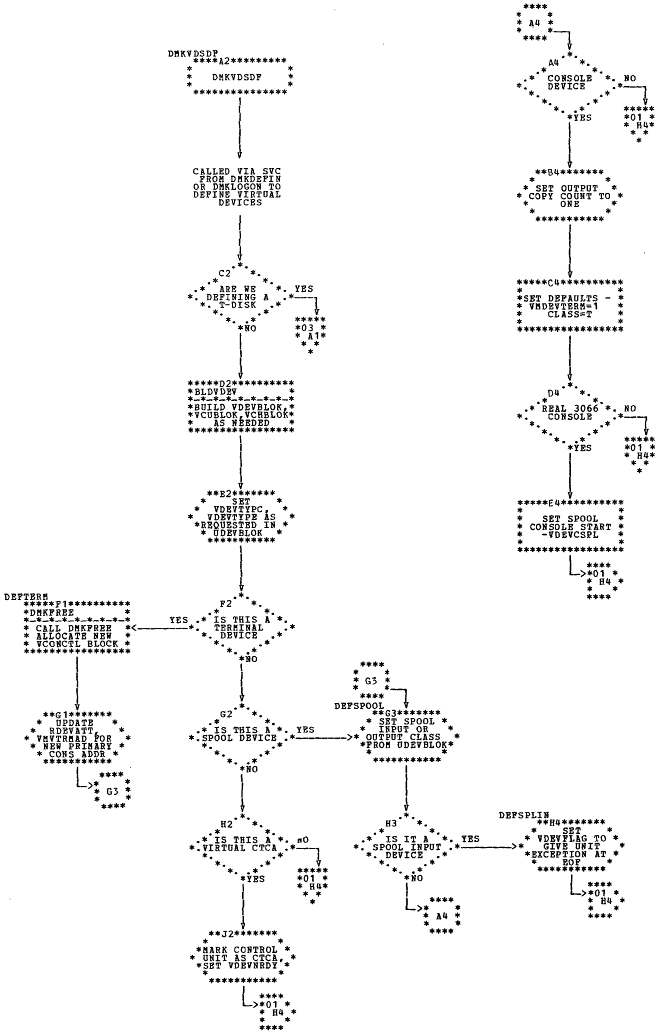

15D3
16 A2
RELDEDD
A2
PICK UP RDEVBLOK ADDRESS FROM VDEVBLOK
B2
IS IT A TAPE DRIVE
YES
NO
RELTAPE
B3
DMKFREE
CALL DMKFREE FREE STORAGE FOR AN IOBLOK
RELDOFF
C2
DMKACODV
CALL DMKACODV PERFORM DEVICE ACCOUNTING
C3
BUILD IOBLOK FOR REWIND/UNLOAD ON TAPE DRIVE
D2
DISCONNECT REAL DEVICE FROM VDEVBLOK, MARK RDEVBLOK FREE
D3
DMKIOSQR
CALL DMKIOSQR START REAL I/O SEQUENCE
E2
WAS DEVICE DEDICATED VIA DIRECTORY
NO
YES
15 H5
A4
CONSTRUCT 'TYPE XXX DETACHED USERID' MESSAGE
B4
DMKSCNRN
CALL DMKSCNRN GET EBCDIC TYPE OF DEVICE
C4
DMKSCNRD
CALL DMKSCNRD FOR REAL DEVICE ADDRESS
D4
DMKCVTBH
CALL DMKCVTBH CONVERT REAL DEVICE ADDRESS
E4
FILL MESSAGE WITH ABOVE INFO AND USERID
F4
DMKCVTBH
CALL DMKCVTBH CONVERT VIRTUAL DEVICE ADDRESS
G4
DMKQCNWT
CALL - TYPE MESSAGE WITH NORET+OPERATOR
15 H5

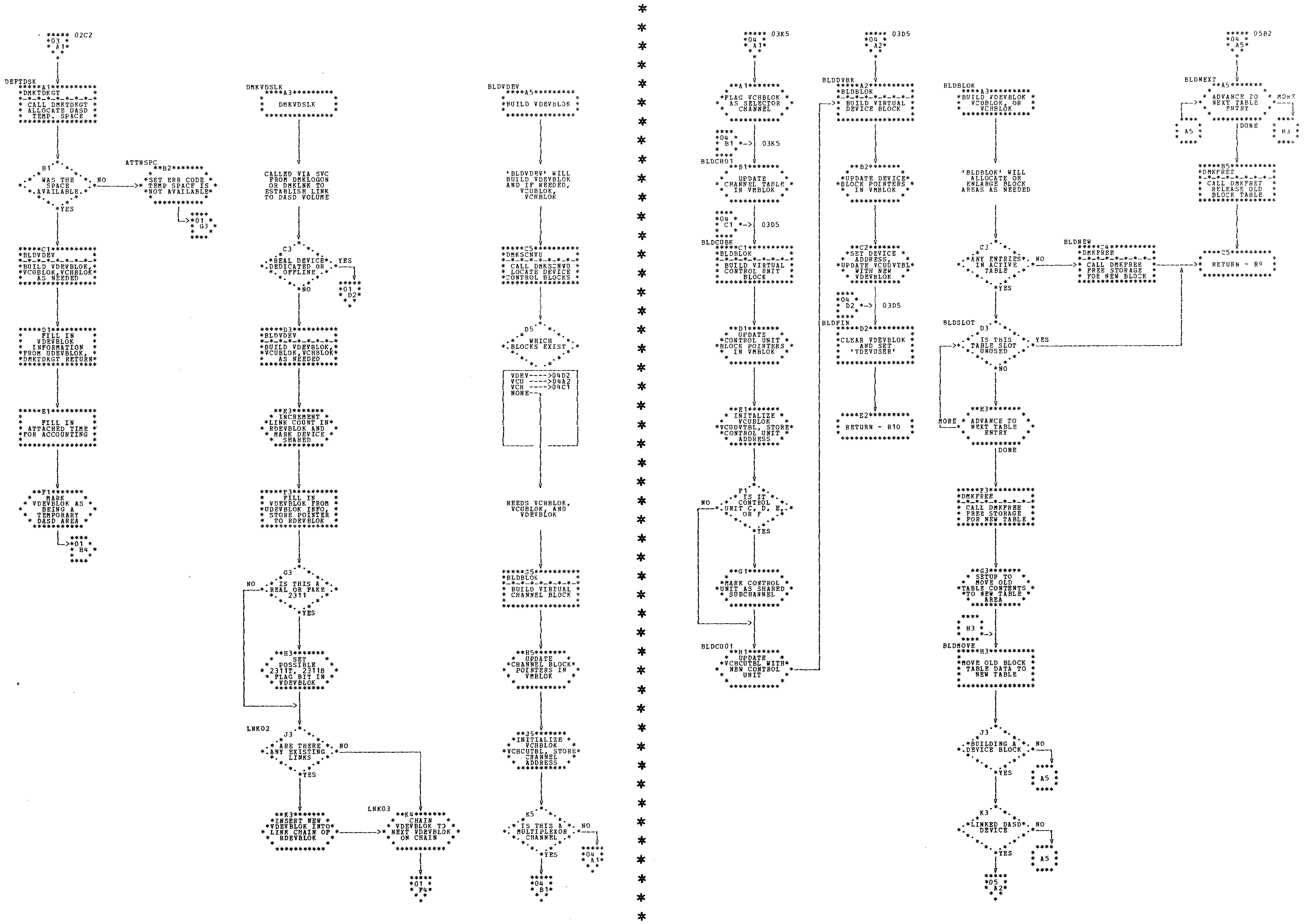

15E3
17 A2
RELTDSK
A1
DMKACODV
CALL TO BUILD ACCOUNTING RECORD
B1
DMKTDKRL
CALL DMKTDKRL RELEASE TEMP. DASD SPACE
15 H5
RELDISK
A2
IS THIS A TEMPORARY DISK
YES
NO
B2
LOAD AND TEST REAL DEVICE LINK COUNT
O.K.
ZERO
C2
- ABEND 3 - LINK COUNT INCORRECT
B3
DECREMENT LINK COUNT IN REAL DEVICE BLOCK
C3
IS LINK COUNT NOW ZERO
NO
YES
D3
REMOVE 'RDEVSYS' FLAG FROM RDEVBLOK DEVICE FREE
RELNOTZ
E3
ARE THERE ANY OTHER LINKS
NO
15 H5
YES
RELLINK
F3
SCAN LINK CHAIN FOR PREVIOUS VDEVBLOK
G3
REMOVE THIS VDEVBLOK FROM LINK CHAIN
15 H5
15H3
17 A4
RELCONS
A4
IS THIS A CONSOLE (1052)
NO
15 H5
YES
B4
OPEN SPOOL FILE
NO
YES
C4
DMKVSPCO
CALL - CLOSE SPOOL FILE
RELC1
D4
SETUP TO RELEASE VCONCTL IF THERE IS ONE
15 J2
FREUSER
A5
UNLOCK USER'S CONTROL BLOCKS
B5
DMKLOCKD
CALL TO UNLOCK THE USERID AND BLOCKS
SWPCALL
C5
SWITCH TO VMBLOK AND TIMERS OF CALLER
D5
RETURN - R5
LOKUSER
A1
PUT TEMPORARY LOCK ON USERID
B1
DMKLOCKQ
CALL TO LOCK USERID
C1
DMKSCNAU
CALL - USER STILL LOGED ON
NO
10 A4
YES
SWPUSER
D1
SWITCH TO ATTACHEE'S VMBLOK, TIMERS
E1
CHANGE CPU TIMER VALUES AND VMBLOK ADDRESSES
F1
RETURN - R5
GETRDEV
A2
LOCATE RDEVBLOK SPECIFIED
B2
DMKSCNRU
CALL DMKSCNRU FIND RDEVBLOK FOR DEVICE
ERR
09 F2
O.K.
C2
RETURN - R5
GETRTYP
A3
GET EBCDIC NAME FOR DEVICE TYPE
B3
DMKSCNRN
CALL DMKSCNRN TRANSLATE TYPE INTO EBCDIC
C3
RETURN - R5
USERDEV
A4
SETUP STRING 'TYPE RADDR USERID'
B4
GETRTYP
GET REAL DEVICE TYPE NAME
C4
CVTRADD
CONVERT REAL DEVICE ADDRESS TO EBCDIC
D4
CONSTRUCT DATA STRING IN SAVE AREA FOR DMKERMSG
E4
RETURN - R6

| DMKVDB -- Process ATTACH and DETACH Commands (Parts 17 and 18 of 18)

| DMKVDS -- ATTACH, DEFINE, and LINK Virtual Device Subroutines (Parts 1 and 2 of 5)

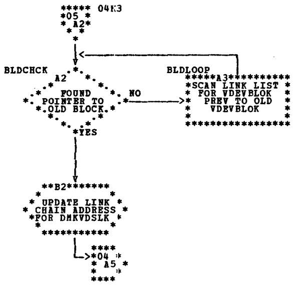

DMKVDSAT
A1
DMKVDSAT
CALLED VIA SVC FROM DMKVDBAT TO ATTACH A REAL DEVICE TO VIRTUAL MACHINE
C1
REAL DEVICE DEDICATED OR OFFLINE
YES
NO
01 D2
03C3
ATTBUSY
D1
REAL DEVICE SHARED OR SYS-OWNED
YES
NO
D2
ERR CODE = REAL DEVICE OFFLINE
YES
NO
E1
UNIT RECORD CLASS DEVICE
NO
A4
YES
E2
ERR CODE = REAL DEVICE DEDICATED
YES
NO
F1
IS THE DEVICE DRAINED
NO
YES
F2
ERR CODE = REAL DEVICE CP-OWNED
YES
NO
01 G3
03B2
G1
IS THERE AN ACTIVE SPOOL FILE
YES
NO
A4
ATTSHRD
G2
SET ERR CODE DEVICE IS SHARED BY SYSTEM USERS
ATTRETN
G3
PASS RETURN CODE BACK TO CALLER, CC = 1
A4
ATTBLD
A4
BLDVDEV
BUILD VDEVBLOK, VCUBLOK, VCHBLOK AS NEEDED
B4
MARK REAL DEVICE BLOCK AS DEVICE DEDICATED
C4
RESET RDEVMOUT FLAG - INDICATE DEVICE ATTACHED
D4
CONNECT REAL AND VIRTUAL BLOCKS, SAVE ACCOUNTING INFORMATION
E4
IS THIS A REAL DASD DEVICE
NO
H4
YES
01 F4
03K4
ATTCHKRD
F4
WAS READ ONLY SPECIFIED
NO
H4
YES
G4
MARK VDEVBLOK FOR READ-ONLY ACCESS TO DEVICE
01 H4
02A4
02H2
02H4
02J2
FNOTE
ATTEXIT
H4
PASS VDEVBLOK ADDR IN R8, SET CC = 0
H4
VDSEXIT
J4
EXIT - SVC 12
TO:H4
02D4
02E4
03F1

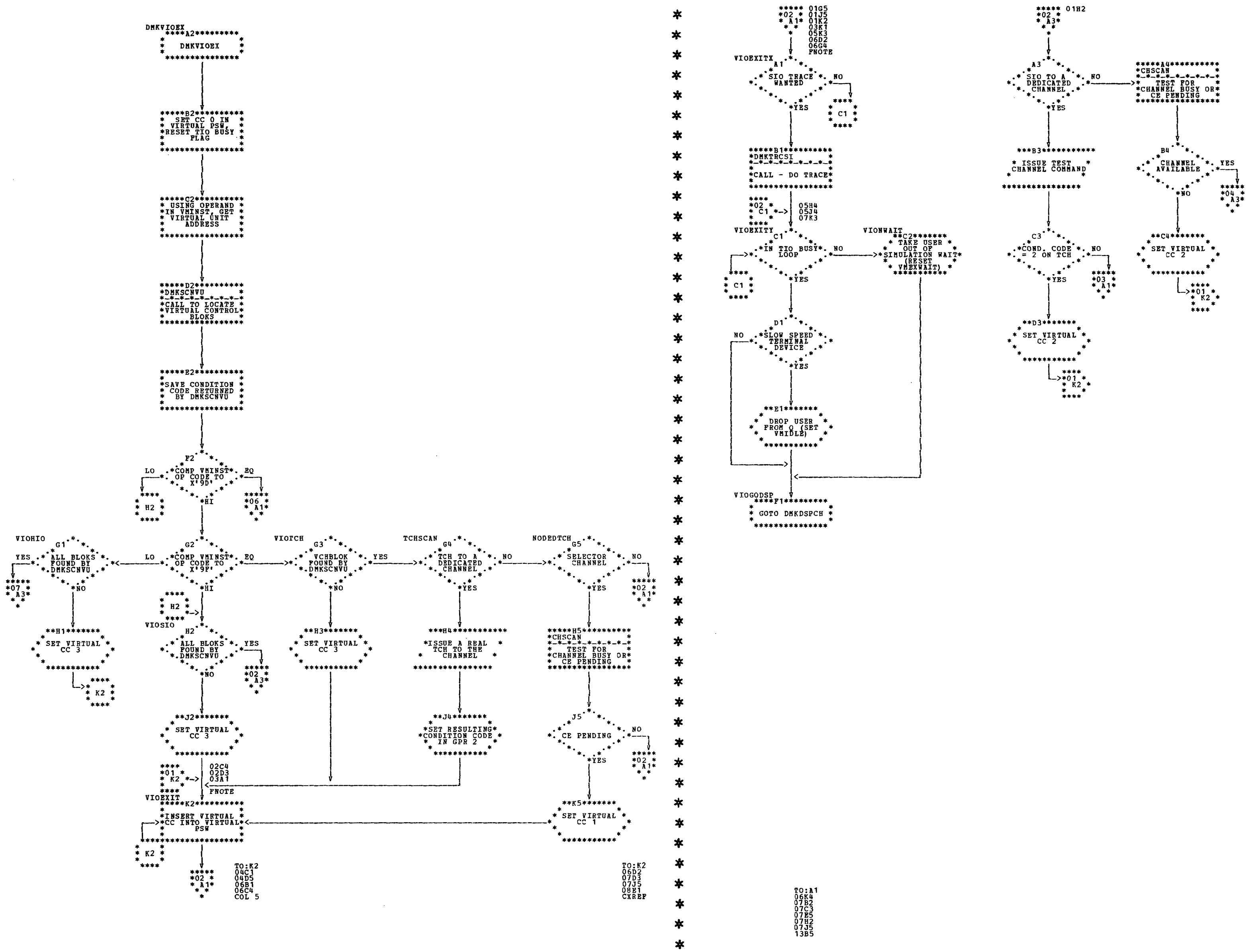

DMKVDSDF
A2
DMKVDSDF
CALLED VIA SVC FROM DMKDEFIN OR DMKLOGON TO DEFINE VIRTUAL DEVICES
C2
ARE WE DEFINING A T-DISK
YES
03 A1
NO
D2
BLDVDEV
BUILD VDEVBLOK, VCUBLOK, VCHBLOK AS NEEDED
E2
SET VDEVTYPC, VDEVTYPE AS REQUESTED IN UDEVBLOK
DEFTERM
F1
DMKFREE
CALL DMKFREE ALLOCATE NEW VCONCTL BLOCK
YES
F2
IS THIS A TERMINAL DEVICE
NO
G1
UPDATE RDEVATT, VMVTRMAD FOR NEW PRIMARY CONS ADDR
G3
G2
IS THIS A SPOOL DEVICE
YES
NO
G3
DEFSPOOL
G3
SET SPOOL INPUT OR OUTPUT CLASS FROM UDEVBLOK
H2
IS THIS A VIRTUAL CTCA
nO
01 H4
YES
H3
IS IT A SPOOL INPUT DEVICE
YES
NO
A4
DEFSPLIN
H4
SET VDEVFLAG TO GIVE UNIT EXCEPTION AT EOF
01 H4
J2
MARK CONTROL UNIT AS CTCA, SET VDEVNRDY
01 H4
A4
A4
CONSOLE DEVICE
NO
01 H4
YES
B4
SET OUTPUT COPY COUNT TO ONE
C4
SET DEFAULTS - VMDEVTERM=1 CLASS=T
D4
REAL 3066 CONSOLE
NO
01 H4
YES
E4
SET SPOOL CONSOLE START -VDEVCSPL
01 H4


02C2
03
A1
DEFTDSK
A1
DMKTDKGT
CALL DMKTDKGT ALLOCATE DASD TEMP. SPACE
B1
WAS THE SPACE AVAILABLE
NO
YES
ATTNSPC
B2
SET ERR CODE TEMP SPACE IS NOT AVAILABLE
01
G3
C1
BLDVDEV
BUILD VDEVBLOK, VCUBLOK,VCHBLOK AS NEEDED
D1
FILL IN VDEVBLOK INFORMATION FROM UDEVBLOK, DMKTDKGT RETURN
E1
FILL IN ATTACHED TIME FOR ACCOUNTING
F1
MARK VDEVBLOK AS BEING A TEMPORARY DASD AREA
01
H4
DMKVDSLK
A3
DMKVDSLK
CALLED VIA SVC FROM DMKLOGON OR DMKLNK TO ESTABLISH LINK TO DASD VOLUME
C3
REAL DEVICE DEDICATED OR OFFLINE
YES
NO
01
D2
D3
BLDVDEV
BUILD VDEVBLOK, VCUBLOK,VCHBLOK AS NEEDED
E3
INCREMENT LINK COUNT IN RDEVBLOK AND MARK DEVICE SHARED
F3
FILL IN VDEVBLOK FROM UDEVBLOK INFO, STORE POINTER TO RDEVBLOK
G3
IS THIS A REAL OR FAKE 2311
NO
YES
H3
SET POSSIBLE 2311T, 2311B FLAG BIT IN VDEVBLOK
LNK02
J3
ARE THERE ANY EXISTING LINKS
NO
YES
K3
INSERT NEW VDEVBLOK INTO LINK CHAIN OF RDEVBLOK
LNK03
K4
CHAIN VDEVBLOK TO NEXT VDEVBLOK ON CHAIN
01
F4
BLDVDEV
A5
BUILD VDEVBLOK
'BLDVDEV' WILL BUILD VDEVBLOK AND IF NEEDED, VCUBLOK, VCHBLOK
C5
DMKSCNVU
CALL DMKSCNVU LOCATE DEVICE CONTROL BLOCKS
D5
WHICH BLOCKS EXIST
VDEV---->04D2
VCU ---->04A2
VCH ---->04C1
NONE--
NEEDS VCHBLOK, VCUBLOK, AND VDEVBLOK
G5
BLDBLOK
BUILD VIRTUAL CHANNEL BLOCK
H5
UPDATE CHANNEL BLOCK POINTERS IN VMBLOK
J5
INITIALIZE VCHBLOK VCHCUTBL, STORE CHANNEL ADDRESS
K5
IS THIS A MULTIPLEXOR CHANNEL
NO
YES
04
A1
04
B1
03K5
04
A1
A1
FLAG VCHBLOK AS SELECTOR CHANNEL
04
B1
03K5
BLDCH01
B1
UPDATE CHANNEL TABLE IN VMBLOK
04
C1
03D5
BLDCUBK
C1
BLDBLOK
BUILD VIRTUAL CONTROL UNIT BLOCK
D1
UPDATE CONTROL UNIT BLOCK POINTERS IN VMBLOK
E1
INITALIZE VCUBLOK VCUDVTBL, STORE CONTROL UNIT ADDRESS
F1
IS IT CONTROL UNIT C, D, E, OR F
NO
YES
G1
MARK CONTROL UNIT AS SHARED SUBCHANNEL
BLDCU01
H1
UPDATE VCHCUTBL WITH NEW CONTROL UNIT
03D5
04
A2
BLDDVBK
A2
BLDBLOK
BUILD VIRTUAL DEVICE BLOCK
B2
UPDATE DEVICE BLOCK POINTERS IN VMBLOK
C2
SET DEVICE ADDRESS, UPDATE VCUDVTBL WITH NEW VDEVBLOK
04
D2
03D5
BLDFIN
D2
CLEAR VDEVBLOK AND SET 'VDEVUSER'
E2
RETURN - R10
BLDBLOK
A3
BUILD VDEVBLOK VCUBLOK, OR VCHBLOK
'BLDBLOK' WILL ALLOCATE OR ENLARGE BLOCK AREAS AS NEEDED
C3
ANY ENTRIES IN ACTIVE TABLE
NO
YES
BLDNEW
C4
DMKFREE
CALL DMKFREE FREE STORAGE FOR NEW BLOCK
A
BLDSLOT
D3
IS THIS TABLE SLOT UNUSED
YES
NO
E3
MORE
ADVANCE TO NEXT TABLE ENTRY
DONE
F3
DMKFREE
CALL DMKFREE FREE STORAGE FOR NEW TABLE
G3
SETUP TO MOVE OLD TABLE CONTENTS TO NEW TABLE AREA
H3
BLDMOVE
H3
MOVE OLD BLOCK TABLE DATA TO NEW TABLE
J3
BUILDING A DEVICE BLOCK
NO
YES
A5
K3
LINKED DASD DEVICE
NO
YES
A5
05
A2
05B2
04
A5
BLDNEXT
A5
ADVANCE TO NEXT TABLE ENTRY
MORE
DONE
A5
H3
B5
DMKFRET
CALL DMKFRET RELEASE OLD BLOCK TABLE
C5
RETURN - R9

| DMKVDS -- ATTACH, DEFINE, and LINK Virtual Device Sutrcutines (Parts 3 and 4 of 5)

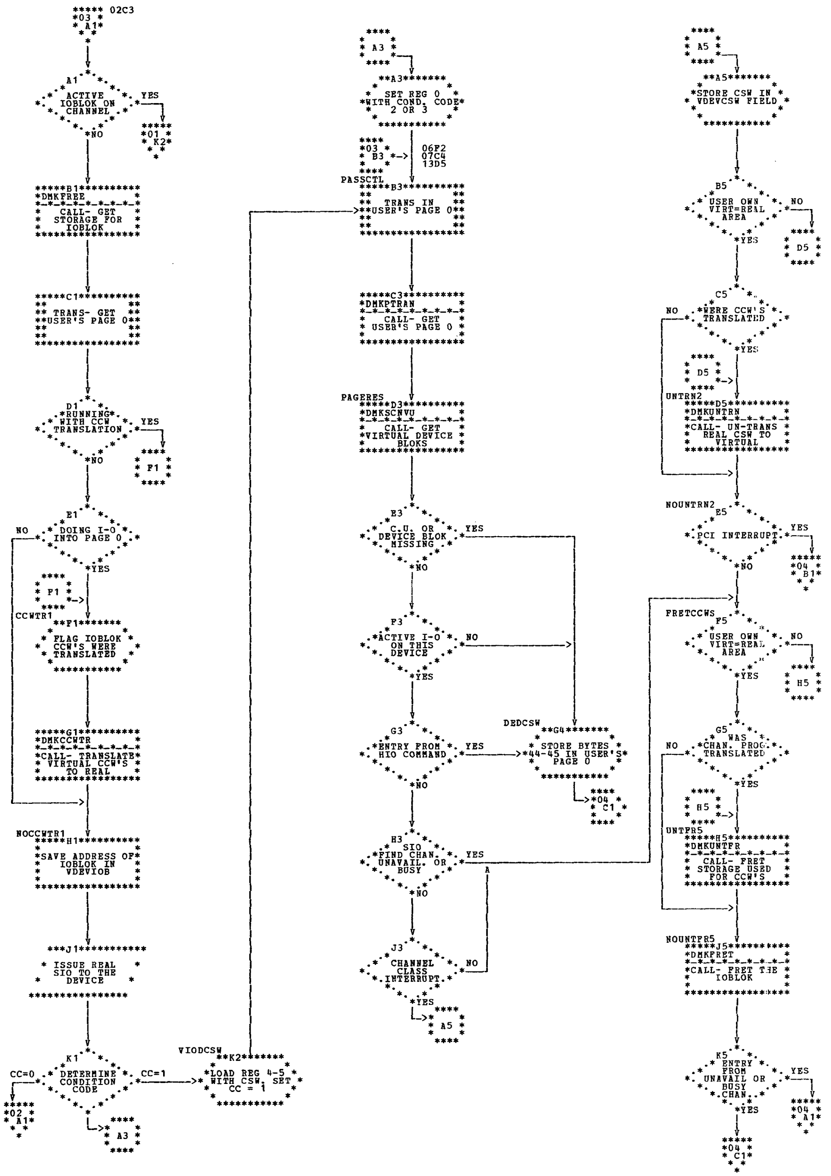
| DMKVDS -- ATTACH, DEFINE, and LINK Virtual Device Subroutines (Part 5 of 5)

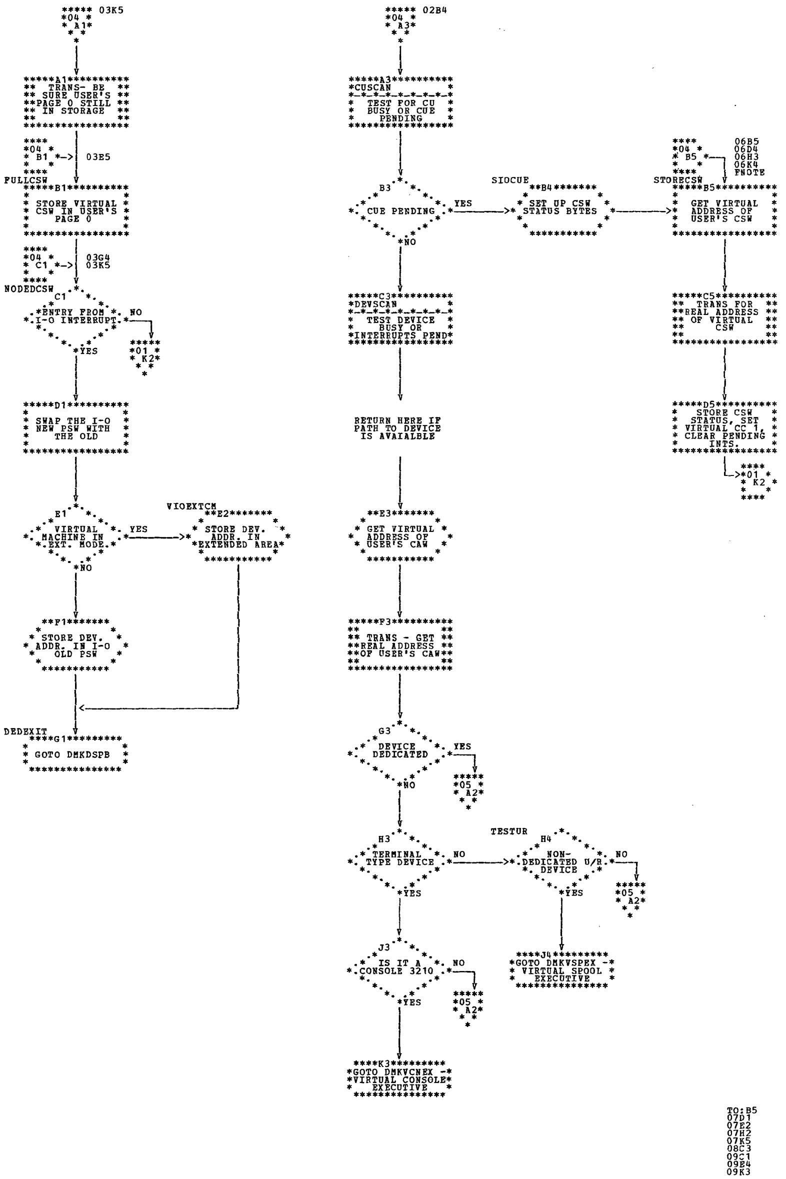

05 A2
04K3
BLDCHCK A2
FOUND POINTER TO OLD BLOCK
NO
YES
BLDLOOP A3
SCAN LINK LIST FOR VDEVBLOK PREV TO OLD VDEVBLOK
B2
UPDATE LINK CHAIN ADDRESS FOR DMKVDSLK
04 A5

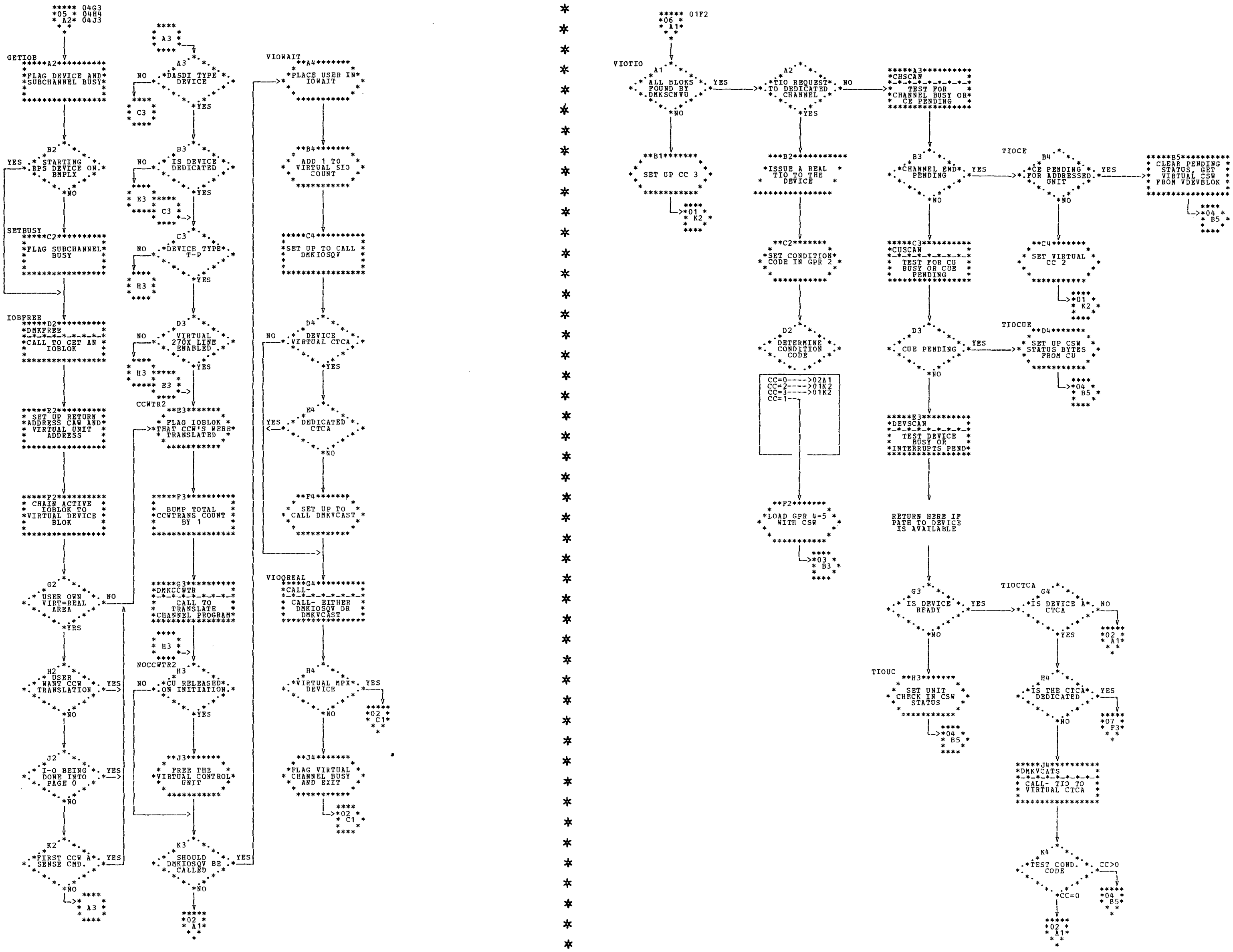


| DMKVIO -- Virtual I/O Manager (Parts 1 and 2 of 14)

| DMKVIO -- Virtual I/O Manager (Parts 3 and 4 of 14)

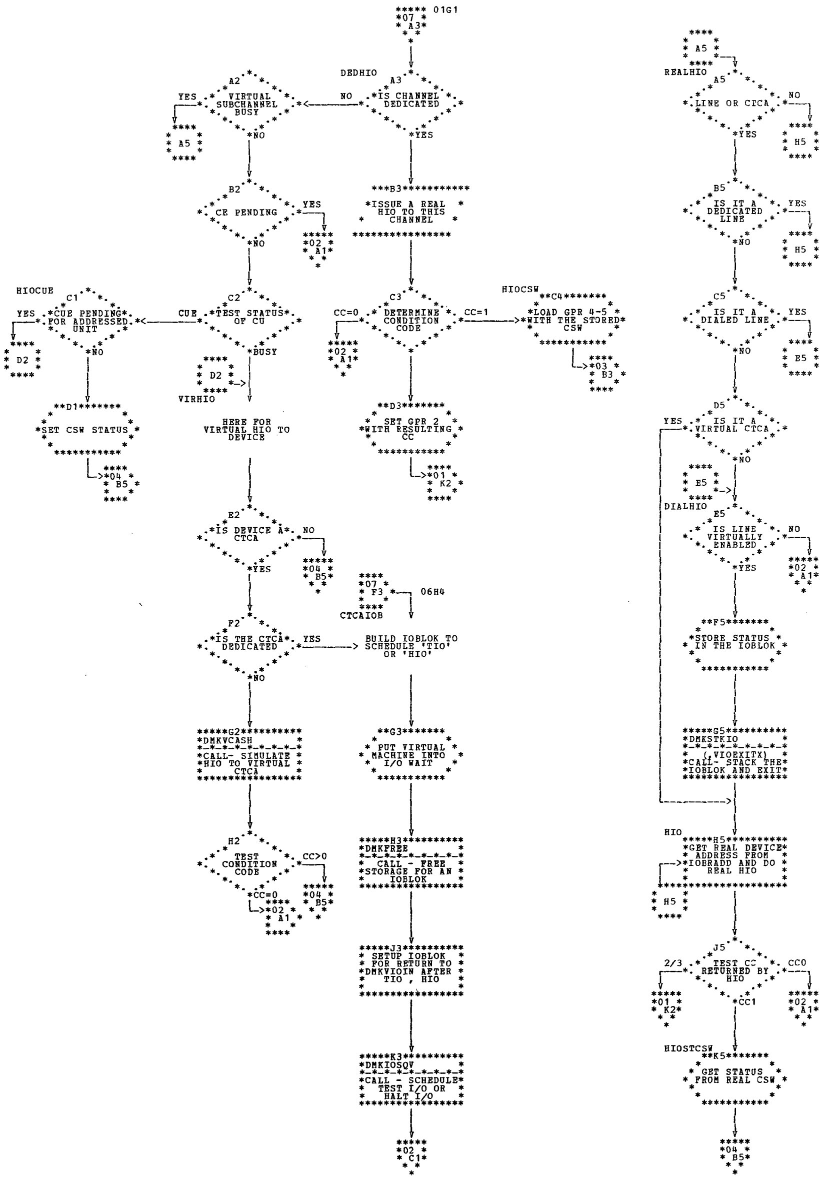

03 A1 02C3
A1 ACTIVE IOBLOK ON CHANNEL
YES
NO
01 K2
B1 DMKFREE CALL- GET STORAGE FOR IOBLOK
C1 TRANS- GET USER'S PAGE 0
D1 RUNNING WITH CCW TRANSLATION
YES
NO
F1
E1 DOING I-O INTO PAGE 0
NO
YES
F1
CCWTR1
F1 FLAG IOBLOK CCW'S WERE TRANSLATED
G1 DMKCCWTR CALL- TRANSLATE VIRTUAL CCW'S TO REAL
NOCCWTR1
H1 SAVE ADDRESS OF IOBLOK IN VDEVIOB
J1 ISSUE REAL SIO TO THE DEVICE
K1 DETERMINE CONDITION CODE
CC=0
CC=1
02 A1
A3
VIODCSW
K2 LOAD REG 4-5 WITH CSW, SET CC = 1
A3
A3 SET REG 0 WITH COND. CODE 2 OR 3
03 B3
06F2
07C4
13D5
PASSCTL
B3 TRANS IN USER'S PAGE 0
C3 DMKPTRAN CALL- GET USER'S PAGE 0
PAGERES
D3 DMKSCNVU CALL- GET VIRTUAL DEVICE BLOKS
E3 C.U. OR DEVICE BLOK MISSING
YES
NO
F3 ACTIVE I-O ON THIS DEVICE
NO
YES
G3 ENTRY FROM HIO COMMAND
YES
NO
DEDCSW
G4 STORE BYTES 44-45 IN USER'S PAGE 0
04 C1
H3 SIO FIND CHAN. UNAVAIL. OR BUSY
YES
NO
A
J3 CHANNEL CLASS INTERRUPT
NO
YES
A5
A5
A5 STORE CSW IN VDEVCSW FIELD
B5 USER OWN VIRT=REAL AREA
NO
YES
D5
C5 WERE CCW'S TRANSLATED
NO
YES
D5
UNTRN2
D5 DMKUNTRN CALL- UN-TRANS REAL CSW TO VIRTUAL
NOUNTRN2
E5 PCI INTERRUPT
YES
NO
04 B1
FRETCCWS
F5 USER OWN VIRT=REAL AREA
NO
YES
H5
G5 WAS CHAN. PROG. TRANSLATED
NO
YES
H5
UNTFR5
H5 DMKUNTFR CALL- FRET STORAGE USED FOR CCW'S
NOUNTFR5
J5 DMKFRET CALL- FRET THE IOBLOK
K5 ENTRY FROM UNAVAIL OR BUSY CHAN.
YES
04 A1
YES
04 C1

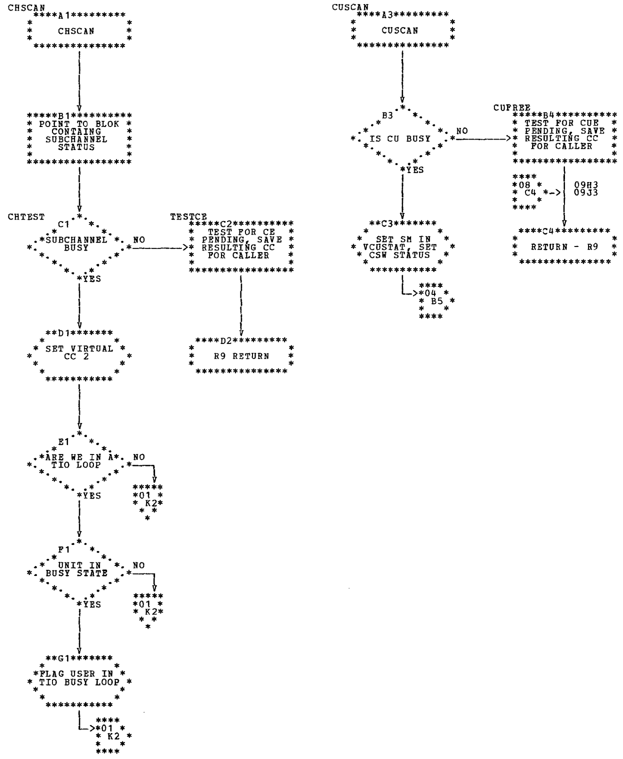

04 A1 03K5
A1 TRANS- BE SURE USER'S PAGE 0 STILL IN STORAGE
04 B1
03E5
FULLCSW
B1 STORE VIRTUAL CSW IN USER'S PAGE 0
04 C1
03G4
03K5
NODEDCSW
C1 ENTRY FROM I-O INTERRUPT
NO
01 K2
YES
D1 SWAP THE I-O NEW PSW WITH THE OLD
E1 VIRTUAL MACHINE IN EXT. MODE
YES
NO
VIOEXTCM
E2 STORE DEV. ADDR. IN EXTENDED AREA
F1 STORE DEV. ADDR. IN I-O OLD PSW
DEDEXIT
G1 GOTO DMKDSPB
04 A3 02B4
A3 CUSCAN TEST FOR CU BUSY OR CUE PENDING
B3 CUE PENDING
YES
NO
SIOCUE
B4 SET UP CSW STATUS BYTES
04 B5
06B5
06D4
06H3
06K4
FNOTE
STORECSW
B5 GET VIRTUAL ADDRESS OF USER'S CSW
C3 DEVSCAN TEST DEVICE BUSY OR INTERRUPTS PEND
C5 TRANS FOR REAL ADDRESS OF VIRTUAL CSW
D5 STORE CSW STATUS, SET VIRTUAL CC 1, CLEAR PENDING INTS.
01 K2
RETURN HERE IF PATH TO DEVICE IS AVAIALBLE
E3 GET VIRTUAL ADDRESS OF USER'S CAW
F3 TRANS - GET REAL ADDRESS OF USER'S CAW
G3 DEVICE DEDICATED
YES
NO
05 A2
H3 TERMINAL TYPE DEVICE
NO
YES
TESTUR
H4 NON-DEDICATED U/R DEVICE
NO
YES
05 A2
J3 IS IT A CONSOLE 3210
NO
YES
05 A2
J4 GOTO DMKVSPEX - VIRTUAL SPOOL EXECUTIVE
K3 GOTO DMKVCNEX - VIRTUAL CONSOLE EXECUTIVE
TO: B5
07D1
07E2
07H2
07K5
08C3
09C1
09E4
09K3

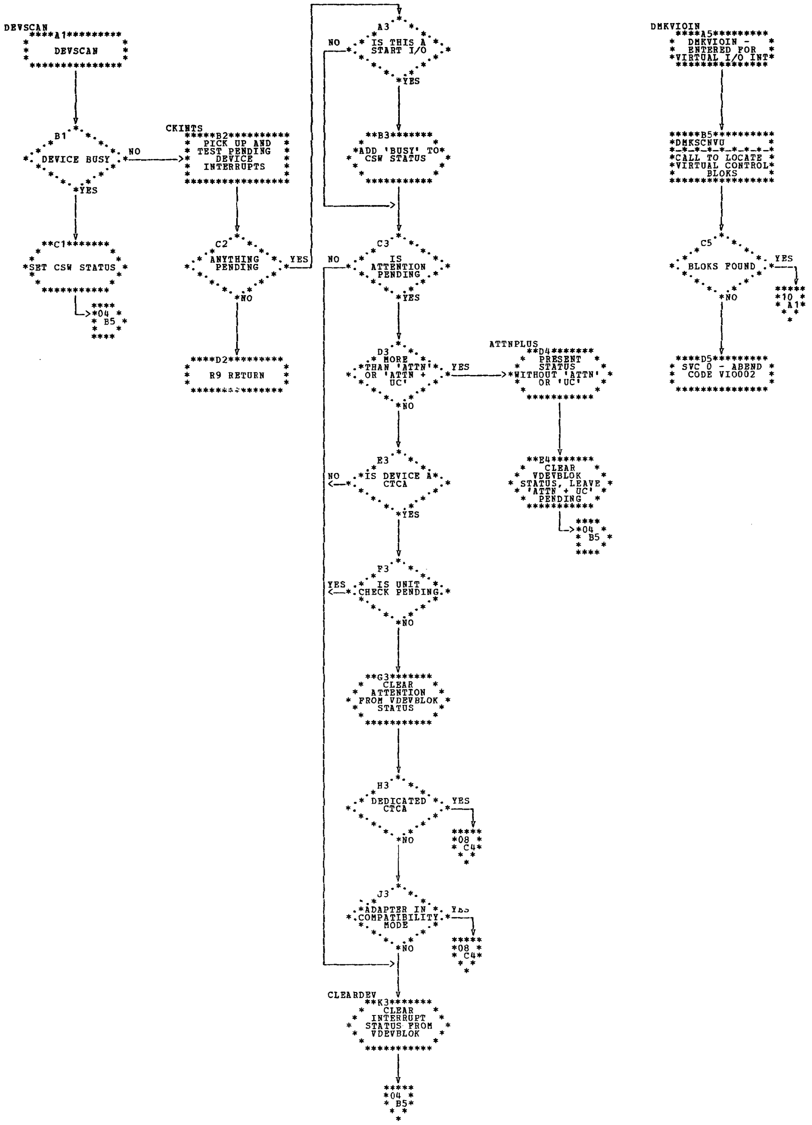

05 A2 04G3 04H4 04J3
GETIOB
A2 FLAG DEVICE AND SUBCHANNEL BUSY
B2 STARTING RPS DEVICE ON BMPLX
YES
NO
SETBUSY
C2 FLAG SUBCHANNEL BUSY
IOBFREE
D2 DMKFREE CALL TO GET AN IOBLOK
E2 SET UP RETURN ADDRESS CAW AND VIRTUAL UNIT ADDRESS
F2 CHAIN ACTIVE IOBLOK TO VIRTUAL DEVICE BLOK
G2 USER OWN VIRT=REAL AREA
NO
YES
A
H2 USER WANT CCW TRANSLATION
YES
NO
J2 I-O BEING DONE INTO PAGE 0
YES
NO
K2 FIRST CCW A SENSE CMD.
YES
NO
A3
A3
A3 DASDI TYPE DEVICE
NO
C3
YES
B3 IS DEVICE DEDICATED
NO
E3
C3
YES
C3 DEVICE TYPE T-P
NO
H3
YES
D3 VIRTUAL 270X LINE ENABLED
NO
H3
E3
YES
CCWTR2
E3 FLAG IOBLOK THAT CCW'S WERE TRANSLATED
F3 BUMP TOTAL CCWTRANS COUNT BY 1
G3 DMKCCWTR CALL TO TRANSLATE CHANNEL PROGRAM
H3
NOCCWTR2
H3 CU RELEASED ON INITIATION
NO
YES
J3 FREE THE VIRTUAL CONTROL UNIT
K3 SHOULD DMKIOSQV BE CALLED
YES
NO
02 A1
VIOWAIT
A4 PLACE USER IN IOWAIT
B4 ADD 1 TO VIRTUAL SIO COUNT
C4 SET UP TO CALL DMKIOSQV
D4 DEVICE VIRTUAL CTCA
NO
YES
E4 DEDICATED CTCA
YES
NO
F4 SET UP TO CALL DMKVCAST
VIOQREAL
G4 CALL- CALL- EITHER DMKIOSQV OR DMKVCAST
H4 VIRTUAL MPX DEVICE
YES
02 C1
NO
J4 FLAG VIRTUAL CHANNEL BUSY AND EXIT
02 C1
06 A1 01F2
VIOTIO
A1 ALL BLOKS FOUND BY DMKSCNVU
YES
NO
B1 SET UP CC 3
01 K2
A2 TIO REQUEST TO DEDICATED CHANNEL
NO
YES
B2 ISSUE A REAL TIO TO THE DEVICE
C2 SET CONDITION CODE IN GPR 2
D2 DETERMINE CONDITION CODE
CC=0---->02A1
CC=2---->01K2
CC=3---->01K2
CC=1--
F2 LOAD GPR 4-5 WITH CSW
03 B3
A3 CHSCAN TEST FOR CHANNEL BUSY OR CE PENDING
B3 CHANNEL END PENDING
YES
NO
TIOCE
B4 CE PENDING FOR ADDRESSED UNIT
YES
NO
B5 CLEAR PENDING STATUS, GET VIRTUAL CSW FROM VDEVBLOK
04 B5
C3 CUSCAN TEST FOR CU BUSY OR CUE PENDING
C4 SET VIRTUAL CC 2
01 K2
D3 CUE PENDING
YES
NO
TIOCUE
D4 SET UP CSW STATUS BYTES FROM CU
04 B5
E3 DEVSCAN TEST DEVICE BUSY OR INTERRUPTS PEND
RETURN HERE IF PATH TO DEVICE IS AVAILABLE
G3 IS DEVICE READY
YES
NO
TIOUC
H3 SET UNIT CHECK IN CSW STATUS
04 B5
TIOCTCA
G4 IS DEVICE A CTCA
NO
02 A1
YES
H4 IS THE CTCA DEDICATED
YES
07 F3
NO
J4 DMKVCATS CALL- TIO TO VIRTUAL CTCA
K4 TEST COND. CODE
CC>0
04 B5
CC=0
02 A1

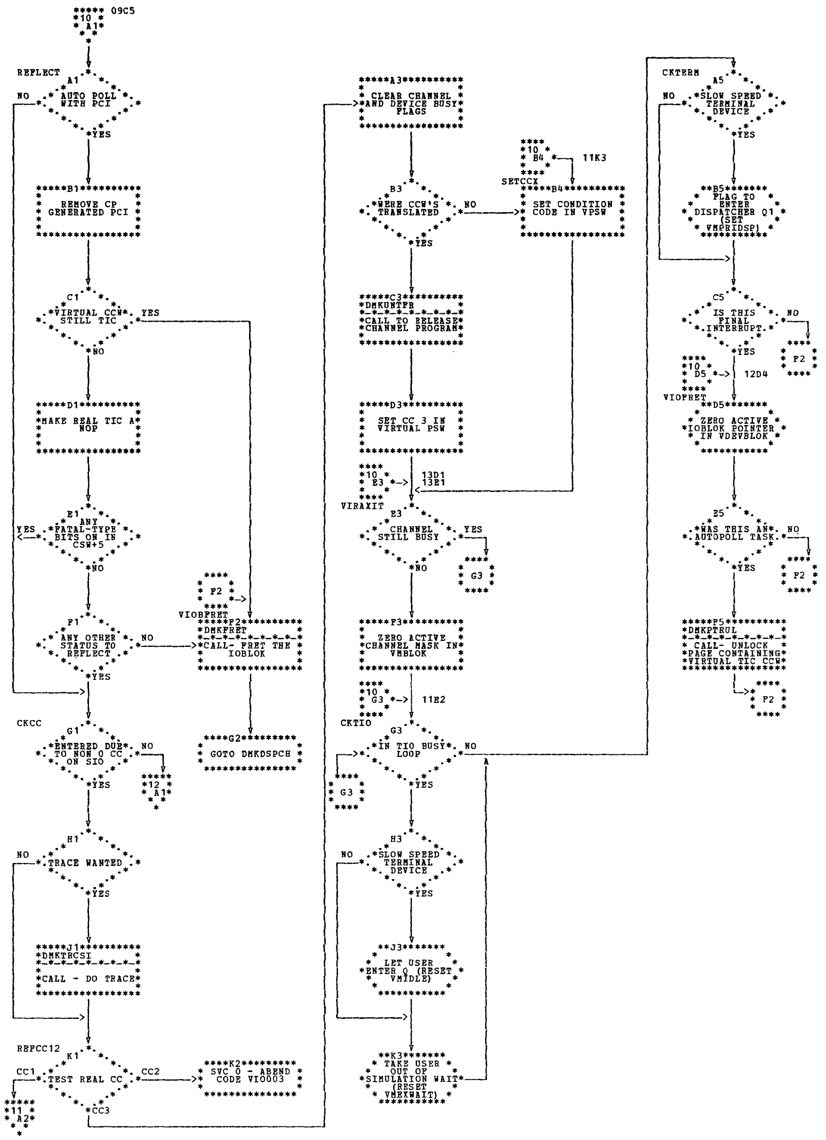
| DMKVIO -- Virtual I/O Manager (Parts 5 and 6 of 14)

| DMKVIC -- Virtual I/O Manager (Parts 7 and 8 of 14)


01G1
07 A3
DEDHIO
A3 IS CHANNEL DEDICATED
NO
YES
A2 VIRTUAL SUBCHANNEL BUSY
YES
A5
NO
B2 CE PENDING
YES
02 A1
NO
C2 TEST STATUS OF CU
CUE
HIOCUE
C1 CUE PENDING FOR ADDRESSED UNIT
YES
D2
NO
D1 SET CSW STATUS
04 B5
BUSY
D2
VIRHIO
HERE FOR VIRTUAL HIO TO DEVICE
E2 IS DEVICE A CTCA
NO
04 B5
YES
F2 IS THE CTCA DEDICATED
YES
NO
G2 DMKVCASH CALL- SIMULATE HIO TO VIRTUAL CTCA
H2 TEST CONDITION CODE
CC>0
04 B5
CC=0
02 A1
B3 ISSUE A REAL HIO TO THIS CHANNEL
C3 DETERMINE CONDITION CODE
CC=0
02 A1
CC=1
HIOCSW
C4 LOAD GPR 4-5 WITH THE STORED CSW
03 B3
D3 SET GPR 2 WITH RESULTING CC
01 K2
07 F3
06H4
CTCAIOB
BUILD IOBLOK TO SCHEDULE 'TIO' OR 'HIO'
G3 PUT VIRTUAL MACHINE INTO I/O WAIT
H3 DMKFREE CALL - FREE STORAGE FOR AN IOBLOK
J3 SETUP IOBLOK FOR RETURN TO DMKVIOIN AFTER TIO , HIO
K3 DMKIOSQV CALL - SCHEDULE TEST I/O OR HALT I/O
02 C1
A5
REALHIO
A5 LINE OR CTCA
NO
H5
YES
B5 IS IT A DEDICATED LINE
YES
H5
NO
C5 IS IT A DIALED LINE
YES
E5
NO
D5 IS IT A VIRTUAL CTCA
YES
NO
E5
DIALHIO
E5 IS LINE VIRTUALLY ENABLED
NO
02 A1
YES
F5 STORE STATUS IN THE IOBLOK
G5 DMKSTKIO (VIOEXITX) CALL- STACK THE IOBLOK AND EXIT
HIO
H5 GET REAL DEVICE ADDRESS FROM IOBRADD AND DO REAL HIO
H5
J5 TEST CC RETURNED BY HIO
2/3
01 K2
CC0
02 A1
CC1
HIOSTCSW
K5 GET STATUS FROM REAL CSW
04 B5

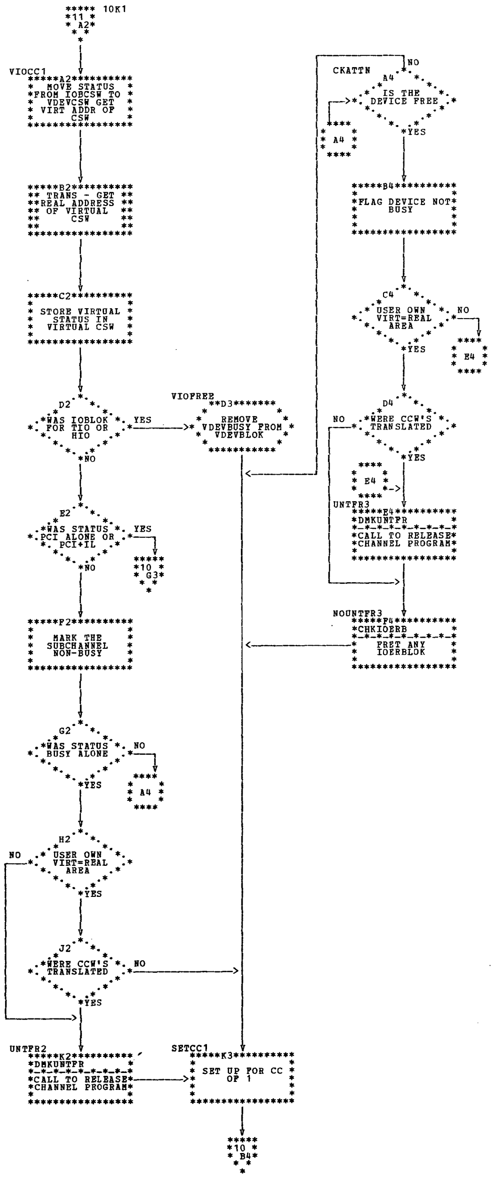


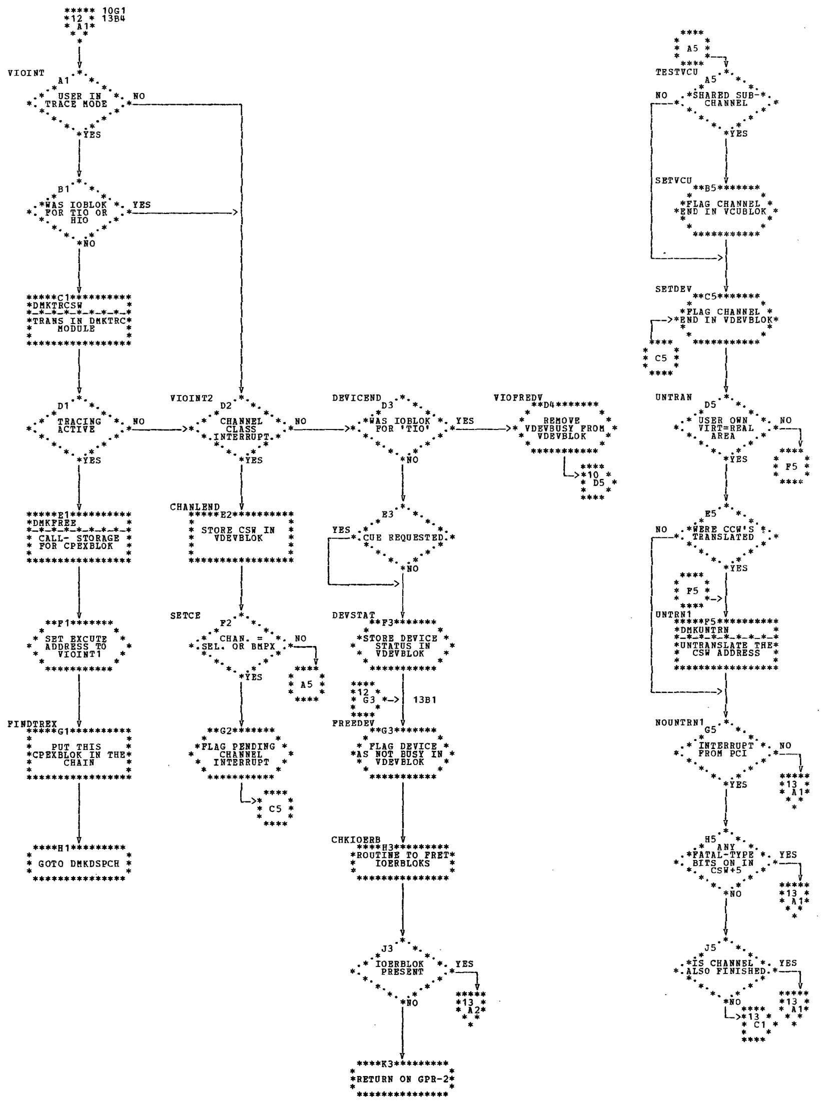

DEVSCAN
A1
DEVSCAN
B1
DEVICE BUSY
NO
YES
C1
SET CSW STATUS
04 B5
CKINTS
B2
PICK UP AND TEST PENDING DEVICE INTERRUPTS
C2
ANYTHING PENDING
YES
NO
D2
R9 RETURN
A3
IS THIS A START I/O
NO
YES
B3
ADD 'BUSY' TO CSW STATUS
C3
IS ATTENTION PENDING
NO
YES
D3
MORE THAN 'ATTN' OR 'ATTN + UC'
YES
NO
ATTNPLUS
D4
PRESENT STATUS WITHOUT 'ATTN' OR 'UC'
E4
CLEAR VDEVBLOK STATUS, LEAVE 'ATTN + UC' PENDING
04 B5
E3
IS DEVICE A CTCA
NO
YES
F3
IS UNIT CHECK PENDING
YES
NO
G3
CLEAR ATTENTION FROM VDEVBLOK STATUS
H3
DEDICATED CTCA
YES
08 C4
NO
J3
ADAPTER IN COMPATIBILITY MODE
YES
08 C4
NO
CLEARDEV
K3
CLEAR INTERRUPT STATUS FROM VDEVBLOK
04 B5
DMKVIOIN
A5
DMKVIOIN - ENTERED FOR VIRTUAL I/O INT
B5
DMKSCNVU
CALL TO LOCATE VIRTUAL CONTROL BLOKS
C5
BLOKS FOUND
YES
10 A1
NO
D5
SVC 0 - ABEND CODE VIO002

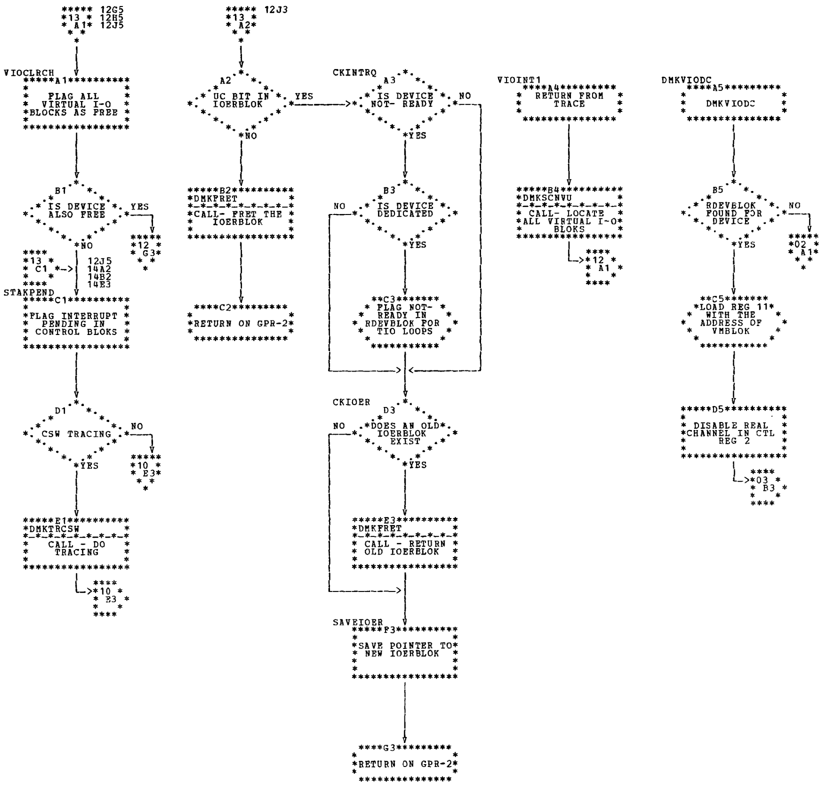


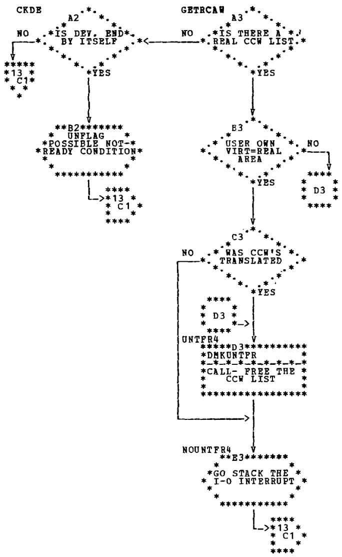
| DMKVIC -- Virtual I/O Manager (Parts 9 and 10 of 14)

| DMKVIO -- Virtual I/O Manager (Parts 11 and 12 of 14)



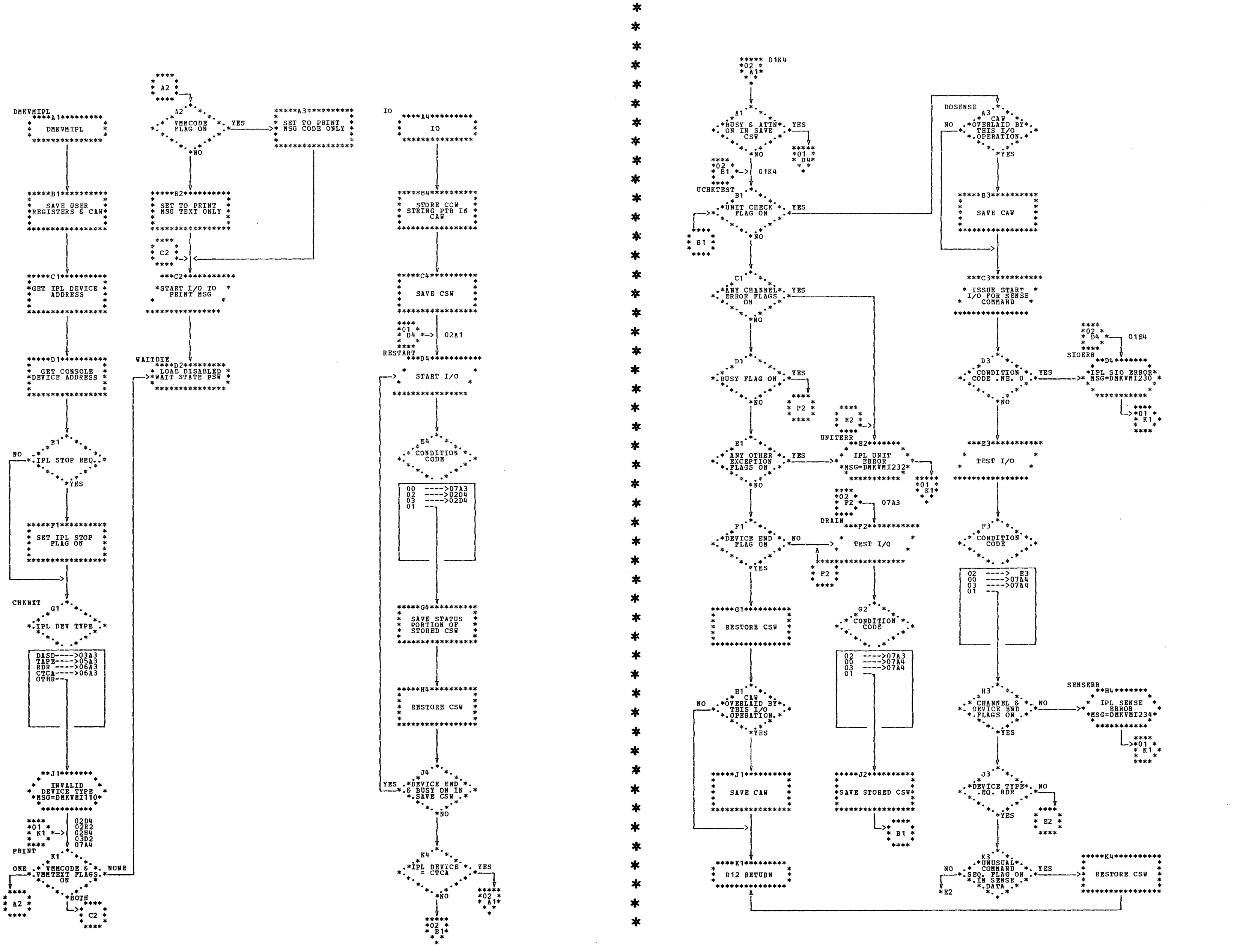


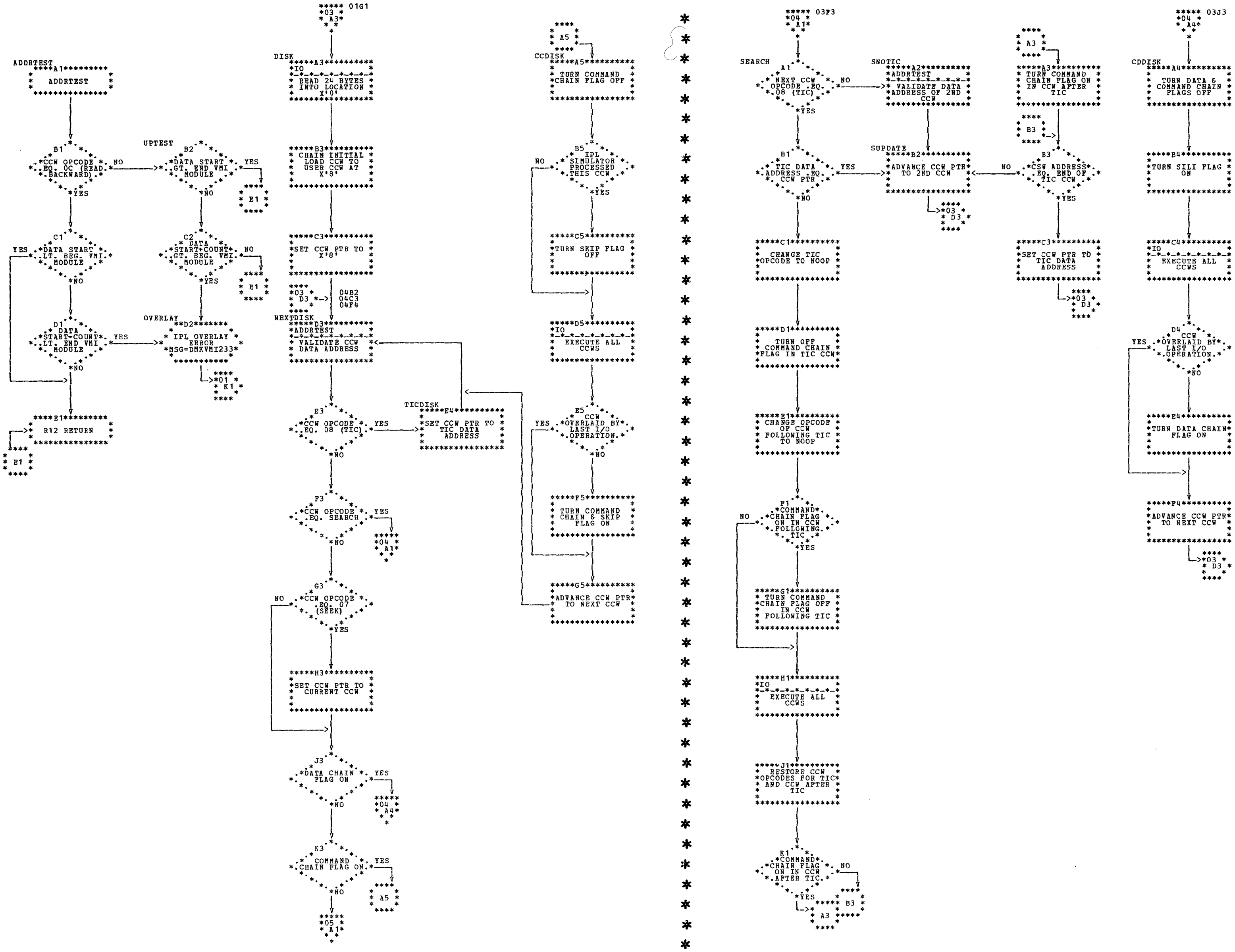

13 A1 12G5 12H5 12J5
VIOCLRCH A1
FLAG ALL VIRTUAL I-O BLOCKS AS FREE
B1 IS DEVICE ALSO FREE
YES
12 G3
NO
13 C1 12J5 14A2 14B2 14E3
STAKPEND C1
FLAG INTERRUPT PENDING IN CONTROL BLOKS
D1 CSW TRACING
NO
10 E3
YES
E1 DMKTRCSW
CALL - DO TRACING
10 E3
13 A2 12J3
A2 UC BIT IN IOERBLOK
YES
NO
B2 DMKFRET
CALL- FRET THE IOERBLOK
C2 RETURN ON GPR-2
CKINTRQ A3 IS DEVICE NOT- READY
NO
YES
B3 IS DEVICE DEDICATED
NO
YES
C3 FLAG NOT- READY IN RDEVBLOK FOR TIO LOOPS
CKIOER D3 DOES AN OLD IOERBLOK EXIST
NO
YES
E3 DMKFRET
CALL - RETURN OLD IOERBLOK
SAVEIOER F3
SAVE POINTER TO NEW IOERBLOK
G3 RETURN ON GPR-2
VIOINT1 A4
RETURN FROM TRACE
B4 DMKSCNVU
CALL- LOCATE ALL VIRTUAL I-O BLOKS
12 A1
DMKVIODC A5
DMKVIODC
B5 RDEVBLOK FOUND FOR DEVICE
NO
02 A1
YES
C5 LOAD REG 11 WITH THE ADDRESS OF VMBLOK
D5 DISABLE REAL CHANNEL IN CTL REG 2
03 B3


CKDE A2 IS DEV. END BY ITSELF
NO
13 C1
YES
B2 UNFLAG POSSIBLE NOT- READY CONDITION
13 C1
GETRCAW A3 IS THERE A REAL CCW LIST
NO
YES
B3 USER OWN VIRT=REAL AREA
NO
D3
YES
C3 WAS CCW'S TRANSLATED
NO
YES
D3
UNTFR4 D3
DMKUNTFR
CALL- FREE THE CCW LIST
NOUNTFR4 E3
GO STACK THE I-O INTERRUPT
13 C1

DMKVIO -- Virtual I/O Manager (Parts 13 and 14 of 14)

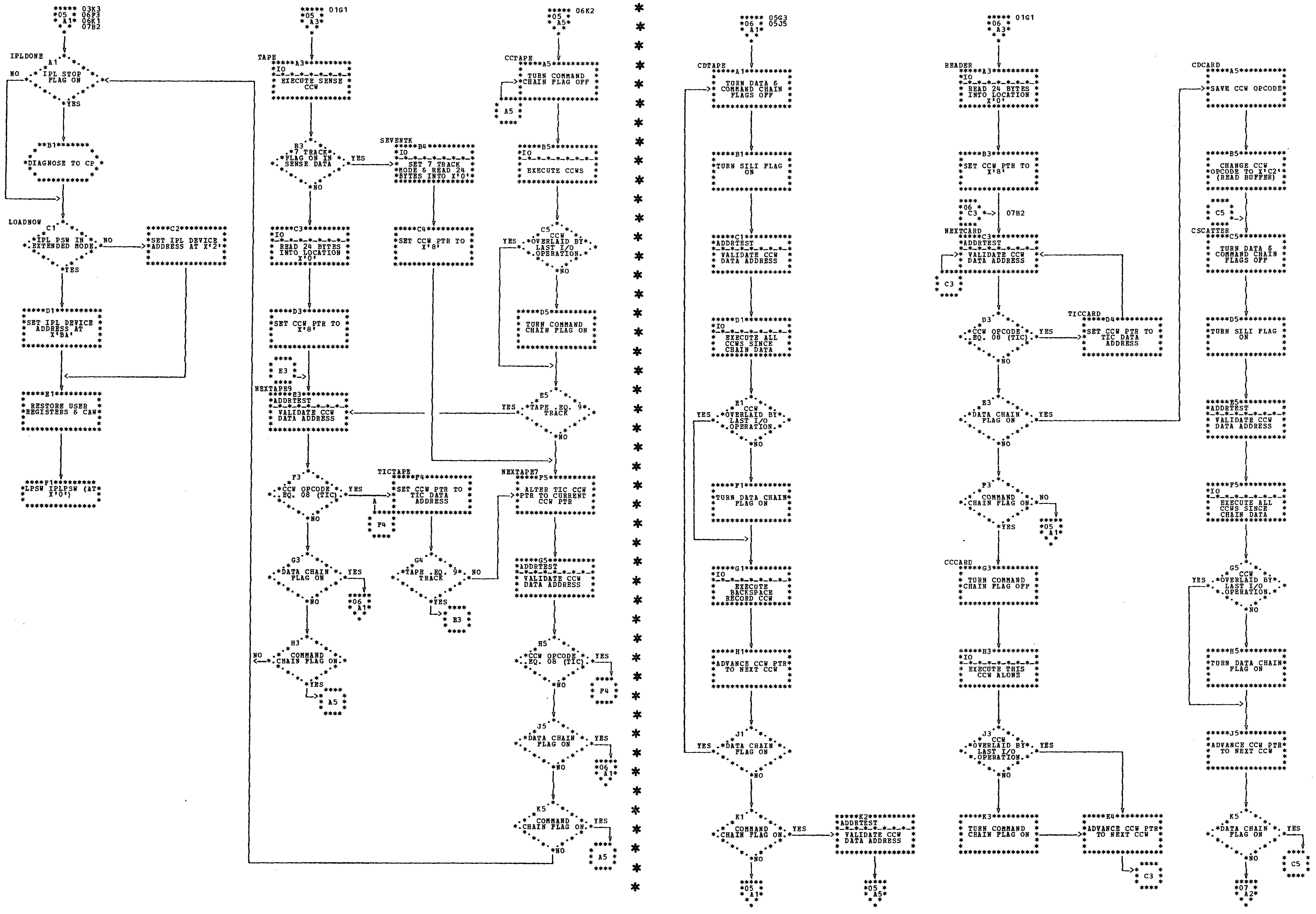
| DMKVMI -- Simulate IPL to Virtual Machine (Parts 1 and 2 of 7)

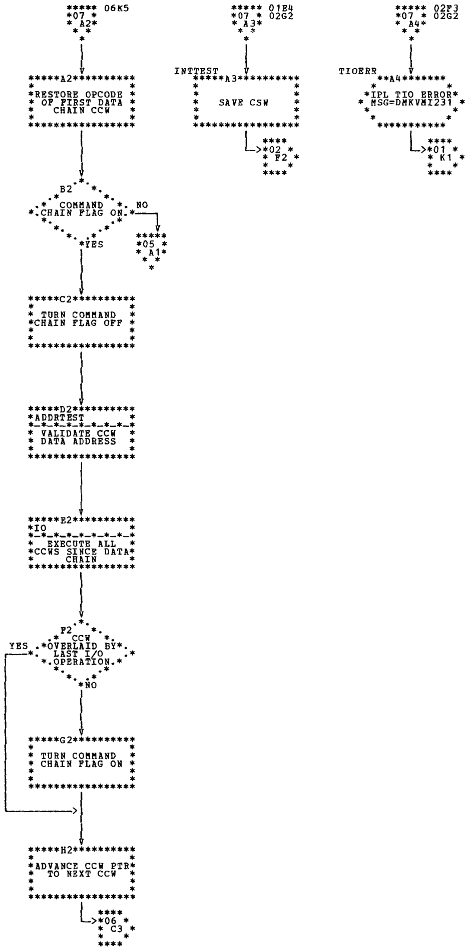



03 A3 01G1
ADDRTEST A1
ADDRTEST
B1 CCW OPCODE .EQ. 0C (READ BACKWARD)
NO
YES
UPTEST B2 DATA START .GT. END VMI MODULE
YES
E1
NO
C1 DATA START .LT. BEG. VMI MODULE
YES
NO
C2 DATA START+COUNT .GT. BEG. VMI MODULE
NO
E1
YES
D1 DATA START+COUNT .LT. END VMI MODULE
YES
NO
OVERLAY D2 IPL OVERLAY ERROR MSG=DMKVMI233
01 K1
E1 R12 RETURN
E1
DISK A3 IO READ 24 BYTES INTO LOCATION X'0'
B3 CHAIN INITIAL LOAD CCW TO USER CCW AT X'8'
C3 SET CCW PTR TO X'8'
03 D3 04B2 04C3 04F4
NEXTDISK D3 ADDRTEST VALIDATE CCW DATA ADDRESS
E3 CCW OPCODE .EQ. 08 (TIC)
YES
NO
TICDISK E4 SET CCW PTR TO TIC DATA ADDRESS
F3 CCW OPCODE .EQ. SEARCH
YES
04 A1
NO
G3 CCW OPCODE .EQ. 07 (SEEK)
NO
YES
H3 SET CCW PTR TO CURRENT CCW
J3 DATA CHAIN FLAG ON
YES
04 A4
NO
K3 COMMAND CHAIN FLAG ON
YES
A5
NO
05 A1
A5
CCDISK A5 TURN COMMAND CHAIN FLAG OFF
B5 IPL SIMULATOR PROCESSED THIS CCW
NO
YES
C5 TURN SKIP FLAG OFF
D5 IO EXECUTE ALL CCWS
E5 CCW OVERLAID BY LAST I/O OPERATION
YES
NO
F5 TURN COMMAND CHAIN & SKIP FLAG ON
G5 ADVANCE CCW PTR TO NEXT CCW
04 A1 03F3
SEARCH A1 NEXT CCW OPCODE .EQ. 08 (TIC)
NO
YES
SNOTIC A2 ADDRTEST VALIDATE DATA ADDRESS OF 2ND CCW
B1 TIC DATA ADDRESS .EQ. CCW PTR
YES
NO
SUPDATE B2 ADVANCE CCW PTR TO 2ND CCW
03 D3
C1 CHANGE TIC OPCODE TO NOOP
D1 TURN OFF COMMAND CHAIN FLAG IN TIC CCW
E1 CHANGE OPCODE OF CCW FOLLOWING TIC TO NOOP
F1 COMMAND CHAIN FLAG ON IN CCW FOLLOWING TIC
NO
YES
G1 TURN COMMAND CHAIN FLAG OFF IN CCW FOLLOWING TIC
H1 IO EXECUTE ALL CCWS
J1 RESTORE CCW OPCODES FOR TIC AND CCW AFTER TIC
K1 COMMAND CHAIN FLAG ON IN CCW AFTER TIC
NO
B3
YES
A3
A3
A3 TURN COMMAND CHAIN FLAG ON IN CCW AFTER TIC
B3
B3 CSW ADDRESS .EQ. END OF TIC CCW
NO
YES
C3 SET CCW PTR TO TIC DATA ADDRESS
03 D3
04 A4 03J3
CDDISK A4 TURN DATA & COMMAND CHAIN FLAGS OFF
B4 TURN SILI FLAG ON
C4 IO EXECUTE ALL CCWS
D4 CCW OVERLAID BY LAST I/O OPERATION
YES
NO
E4 TURN DATA CHAIN FLAG ON
F4 ADVANCE CCW PTR TO NEXT CCW
03 D3

| DMKVMI -- Simulate IPL to Virtual Machine (Parts 3 and 4 of 7)

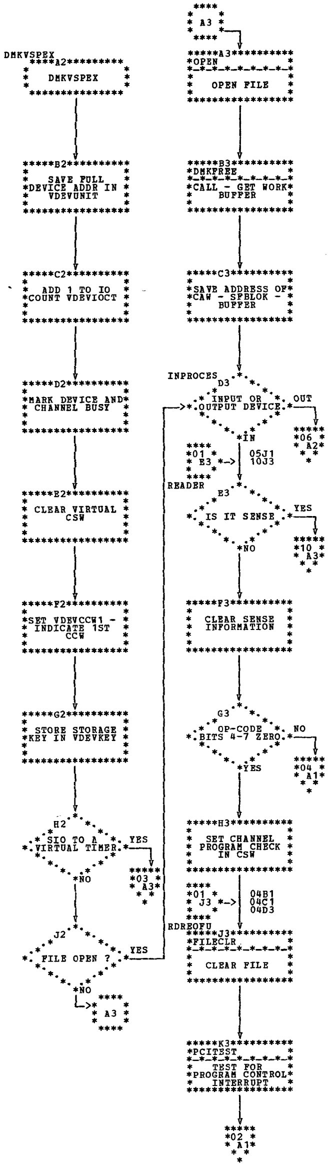
| DMKVMI -- Simulate IPL to Virtual Machine (Parts 5 and 6 of 7)

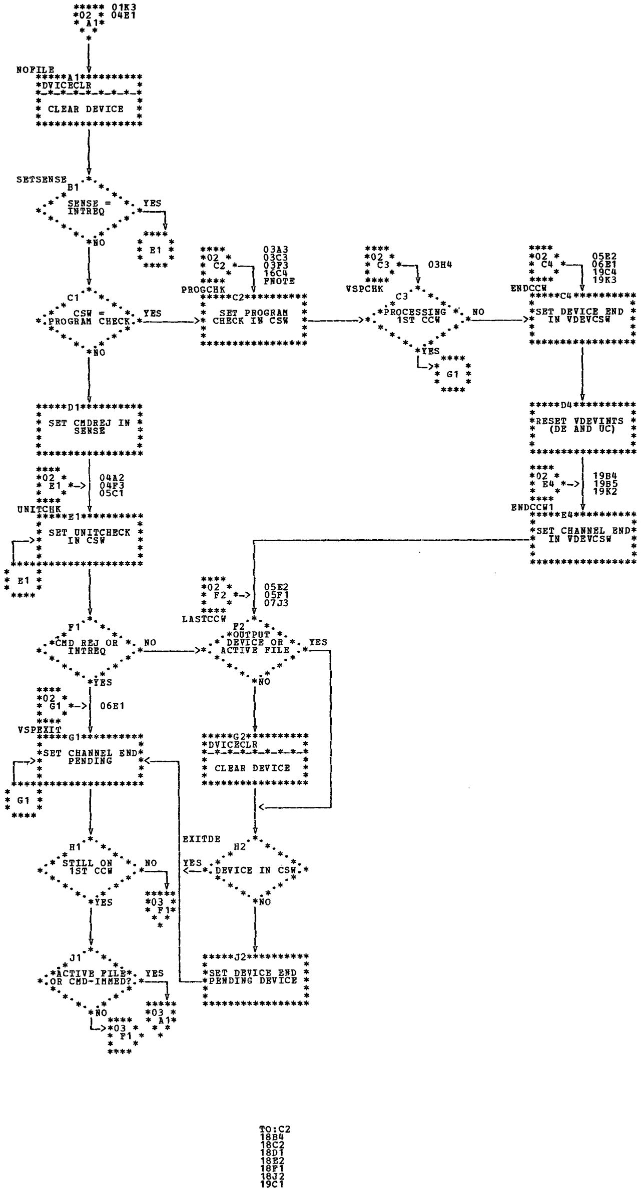


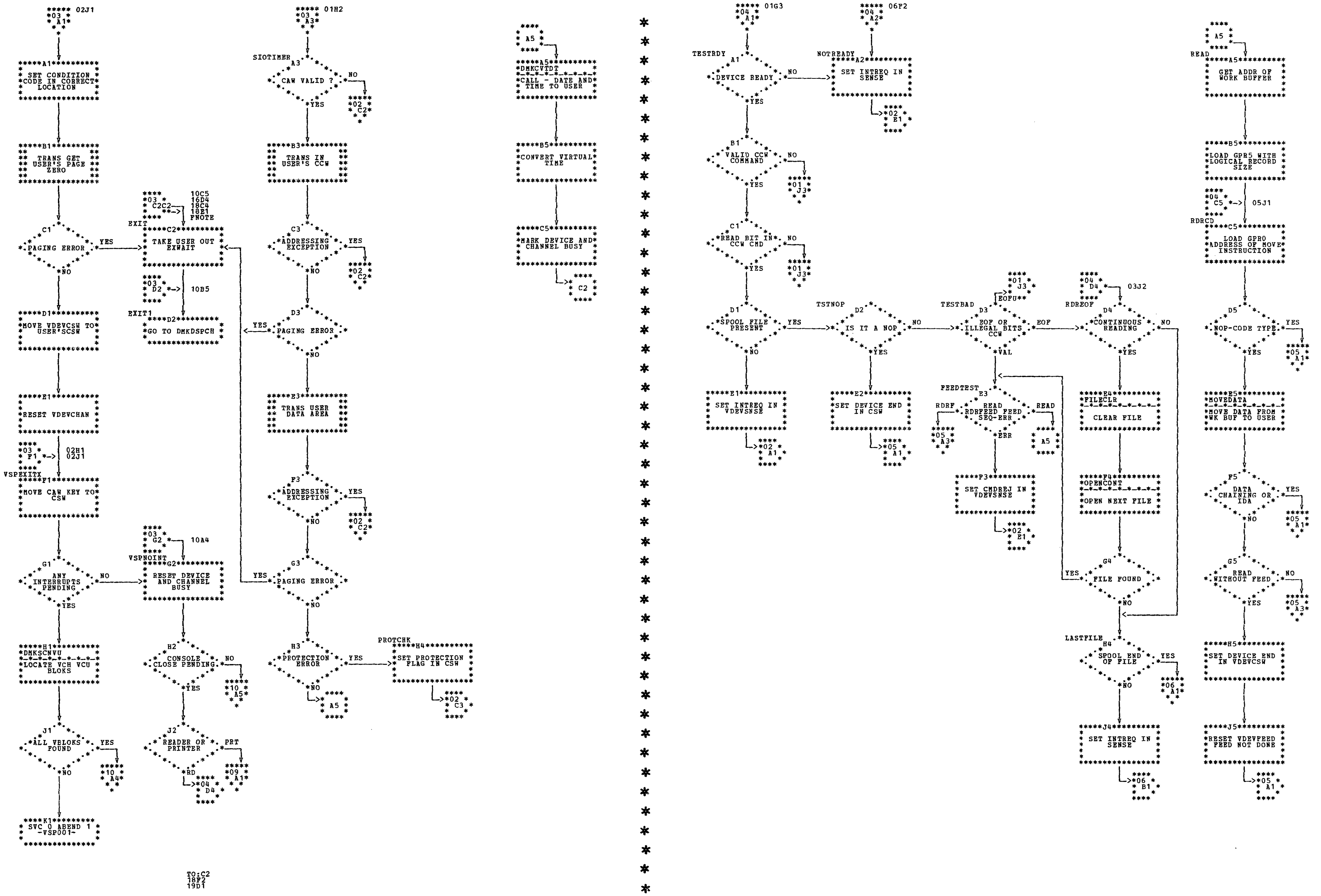

06K5
07
A2
01E4
02G2
07
A3
02F3
02G2
07
A4
A2
RESTORE OPCODE OF FIRST DATA CHAIN CCW
INTTEST
A3
SAVE CSW
02
F2
TIOERR
A4
IPL TIO ERROR MSG=DMKVMI231
01
K1
B2
COMMAND CHAIN FLAG ON
NO
05
A1
YES
C2
TURN COMMAND CHAIN FLAG OFF
D2
ADDRTEST
VALIDATE CCW DATA ADDRESS
E2
IO
EXECUTE ALL CCWS SINCE DATA CHAIN
F2
CCW OVERLAID BY LAST I/O OPERATION
YES
NO
G2
TURN COMMAND CHAIN FLAG ON
H2
ADVANCE CCW PTR TO NEXT CCW
06
C3

| DMKVMI -- Simulate IPL to Virtual Machine (Part 7 of 7)

| DMKVSP -- Virtual Spooling Manager (Parts 1 and 2 of 19)

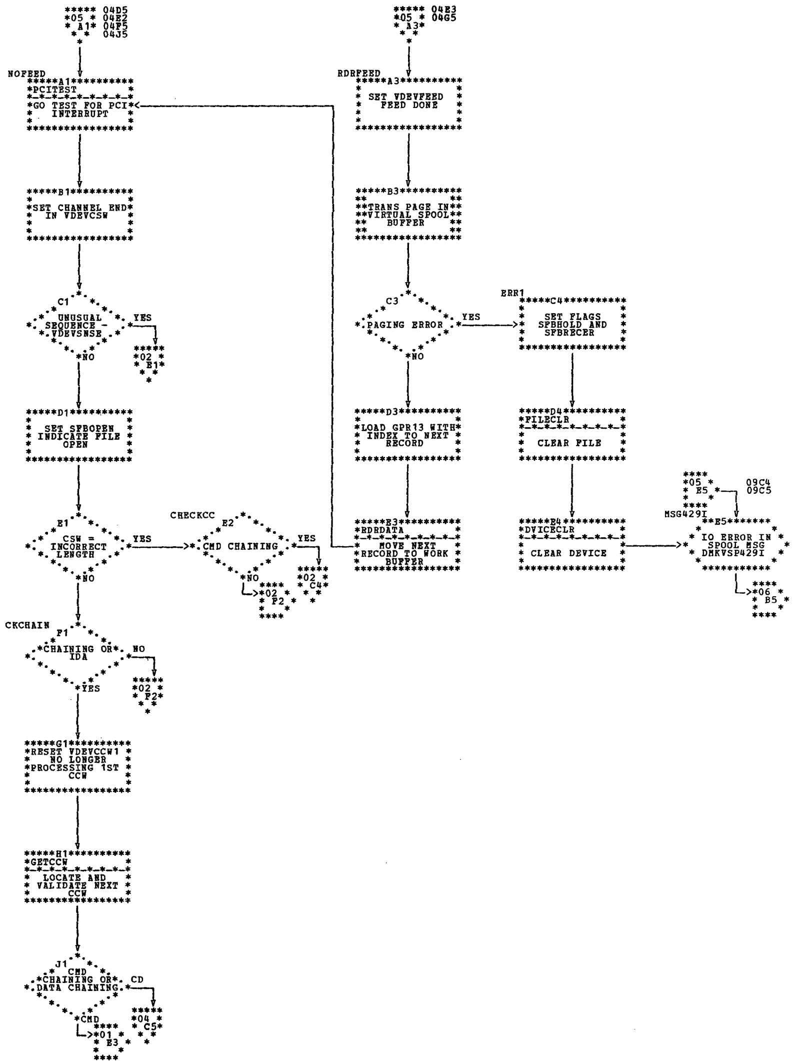


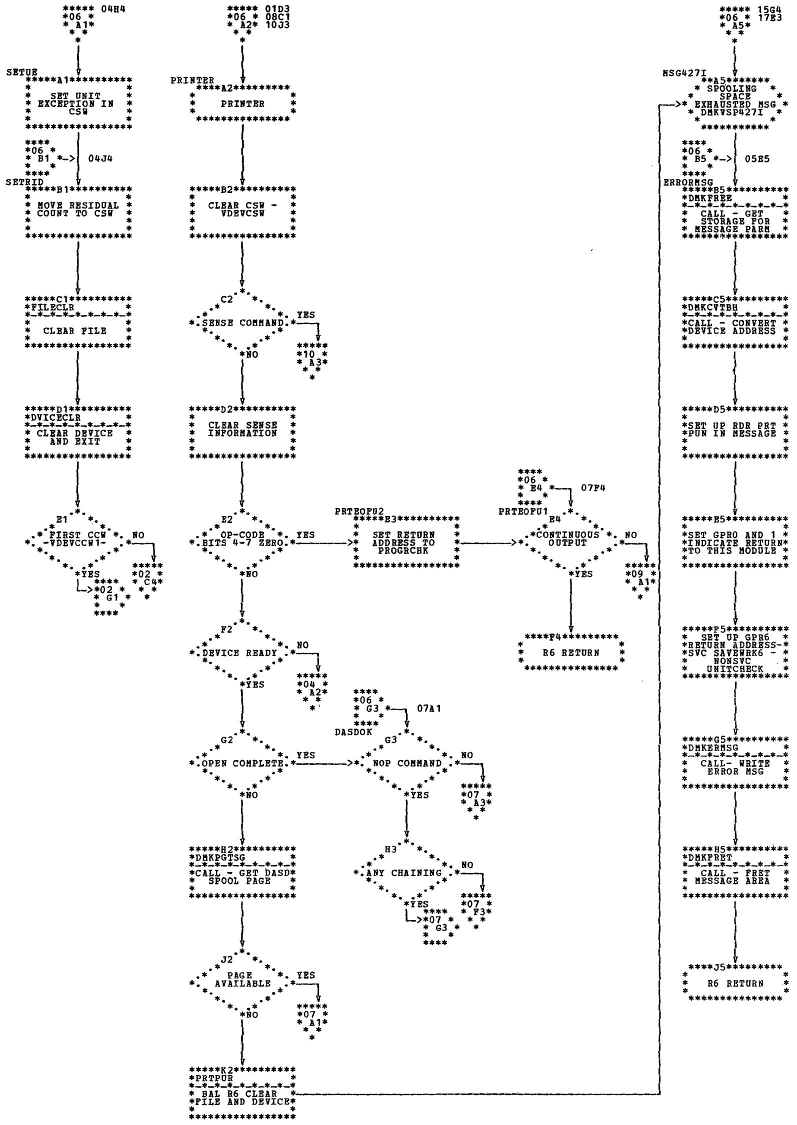


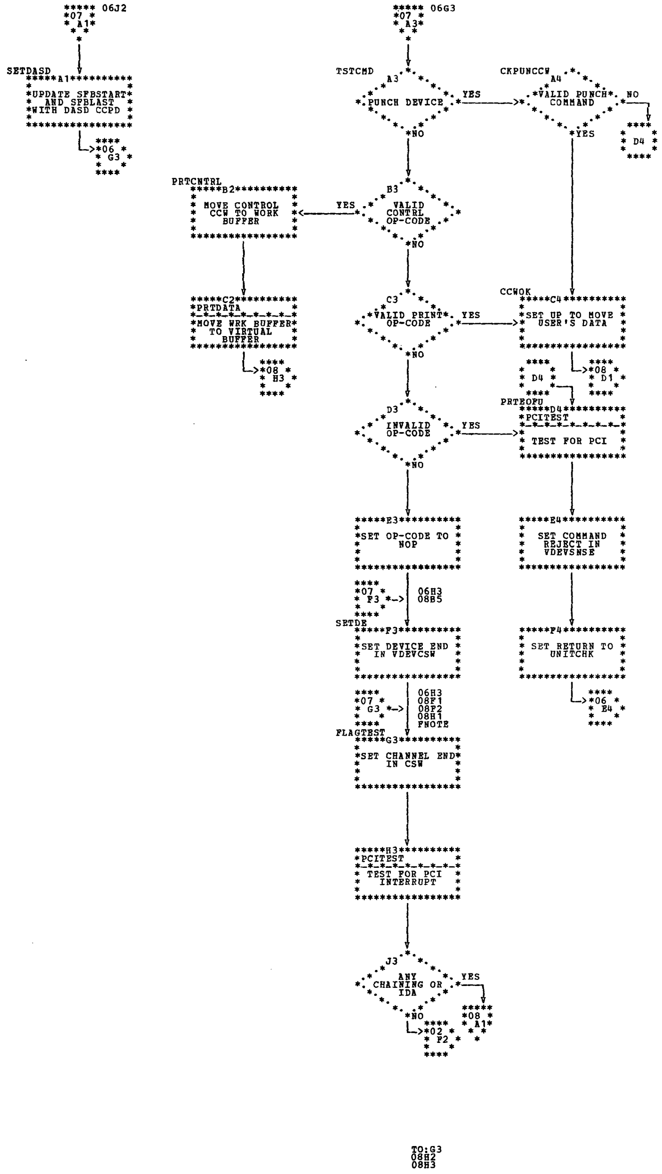


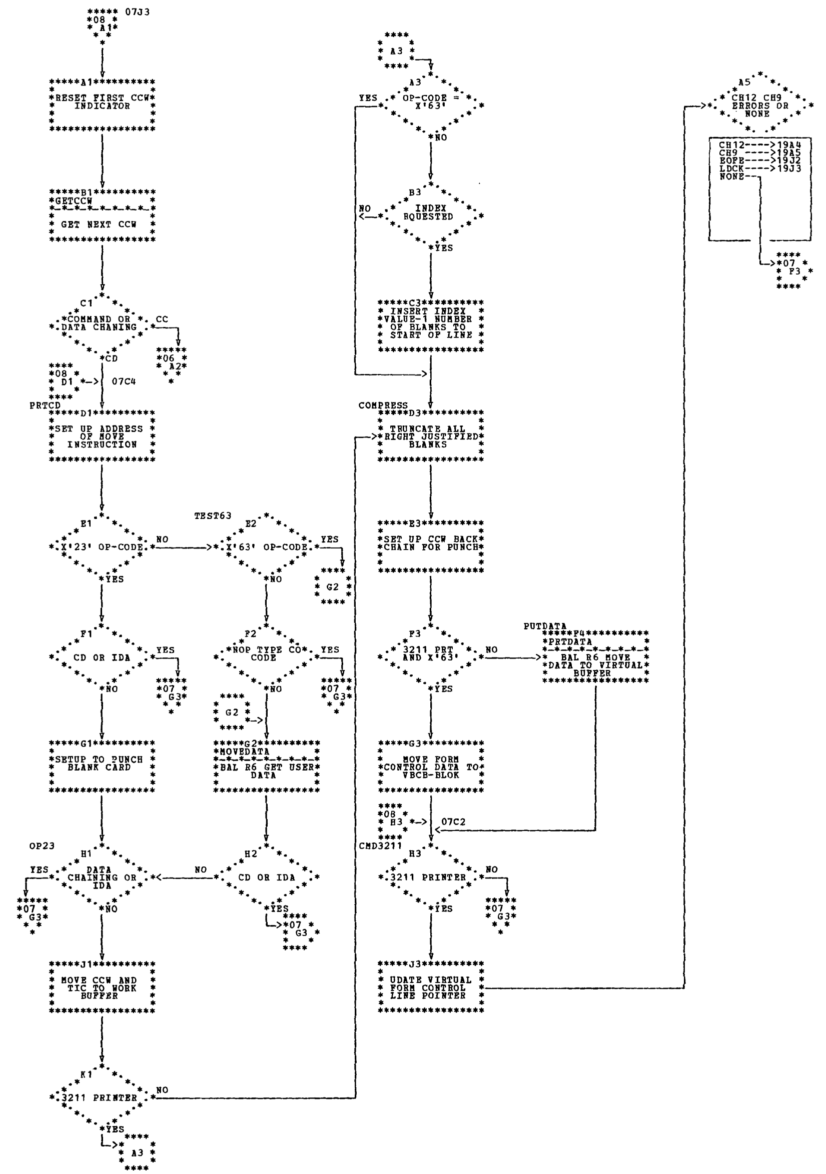
| DMKVSP -- Virtual Spooling Manager (Parts 3 and 4 of 19)

| DMKVSP -- Virtual Spooling Manager (Parts 5 and 6 of 19)


05 A1 04D5 04E2 04F5 04J5
NOFEED
A1 PCITEST
GO TEST FOR PCI INTERRUPT
B1 SET CHANNEL END IN VDEVCSW
C1 UNUSUAL SEQUENCE - VDEVSNSE
YES
02 E1
NO
D1 SET SFBOPEN INDICATE FILE OPEN
E1 CSW = INCORRECT LENGTH
YES
NO
CHECKCC
E2 CMD CHAINING
YES
02 C4
NO
02 F2
CKCHAIN
F1 CHAINING OR IDA
NO
02 F2
YES
G1 RESET VDEVCCW1 NO LONGER PROCESSING 1ST CCW
H1 GETCCW
LOCATE AND VALIDATE NEXT CCW
J1 CMD CHAINING OR DATA CHAINING
CD
04 C5
CMD
01 E3
05 A3 04E3 04G5
RDRFEED
A3 SET VDEVFEED FEED DONE
B3 TRANS PAGE IN VIRTUAL SPOOL BUFFER
C3 PAGING ERROR
YES
NO
D3 LOAD GPR13 WITH INDEX TO NEXT RECORD
E3 RDRDATA
MOVE NEXT RECORD TO WORK BUFFER
ERR1
C4 SET FLAGS SFBHOLD AND SFBRECER
D4 FILECLR
CLEAR FILE
E4 DVICECLR
CLEAR DEVICE
05 E5 09C4 09C5
MSG429I
E5 IO ERROR IN SPOOL MSG DMKVSP429I
06 B5

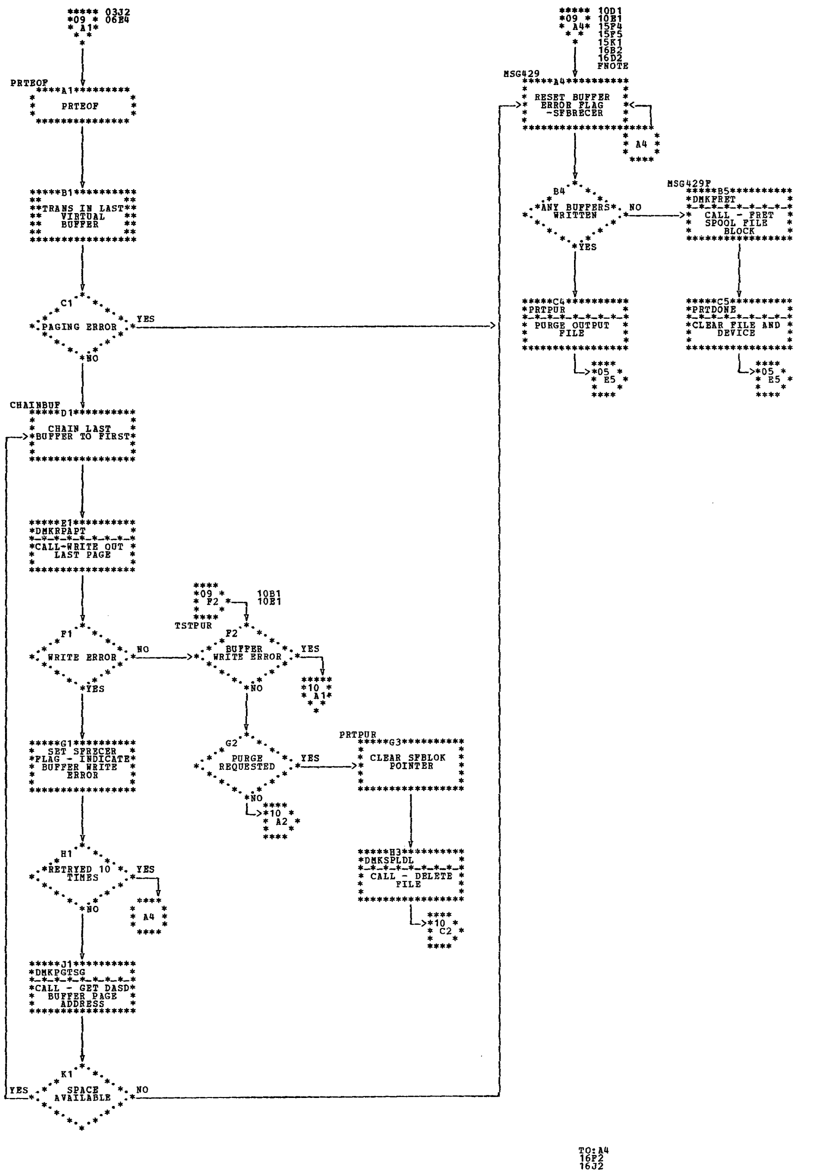


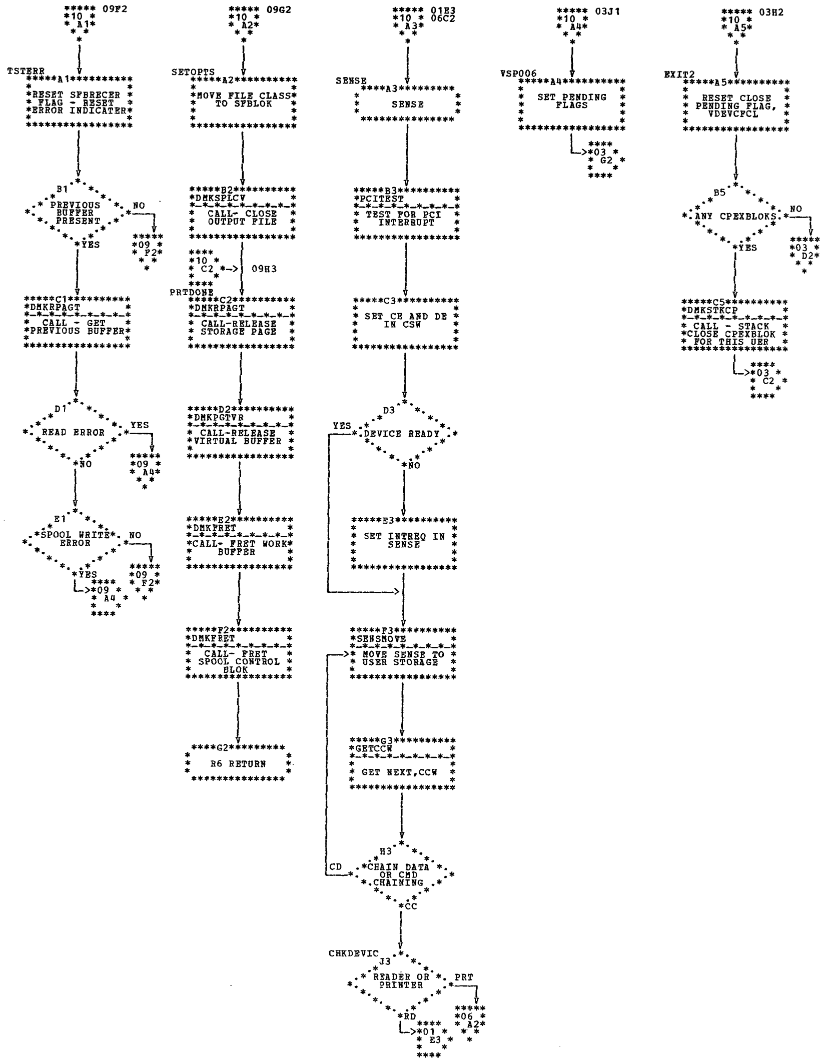


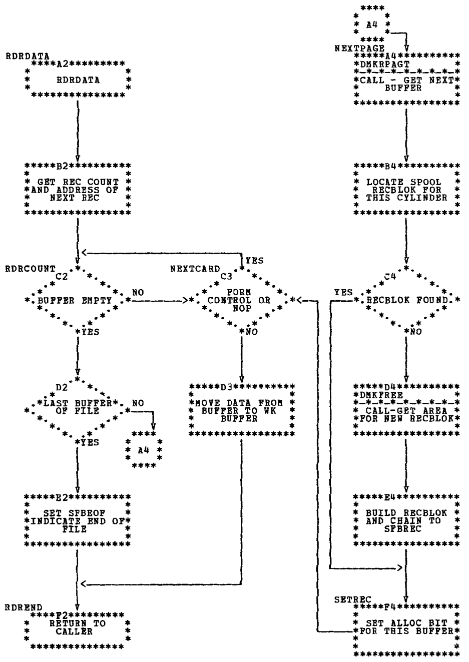


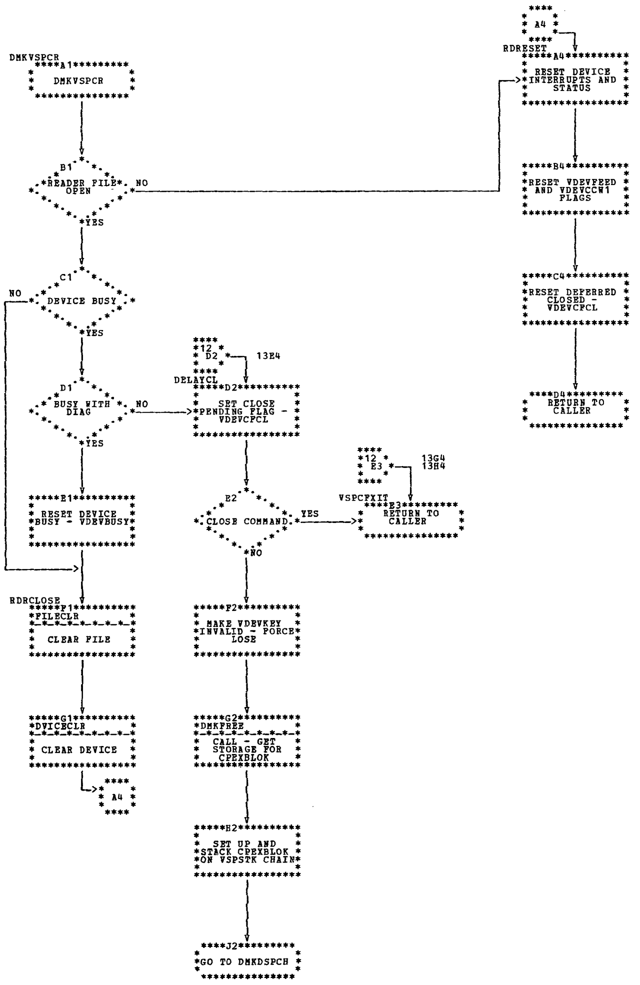
| DMKVSP -- Virtual Spooling Manager (Parts 7 and 8 of 19)

| DMKVSP -- Virtual Spooling Manager (Parts 9 and 10 of 19)


09 A1 03J2 06E4
PRTEOF
A1 PRTEOF
B1 TRANS IN LAST VIRTUAL BUFFER
C1 PAGING ERROR
YES
NO
CHAINBUF
D1 CHAIN LAST BUFFER TO FIRST
E1 DMKRPAPT CALL-WRITE OUT LAST PAGE
F1 WRITE ERROR
NO
YES
09 F2 10B1 10E1
TSTPUR
F2 BUFFER WRITE ERROR
YES
10 A1
NO
G1 SET SFRECER FLAG - INDICATE BUFFER WRITE ERROR
G2 PURGE REQUESTED
YES
NO
10 A2
PRTPUR
G3 CLEAR SFBLOK POINTER
H1 RETRYED 10 TIMES
YES
A4
NO
H3 DMKSPLDL CALL - DELETE FILE
10 C2
J1 DMKPGTSG CALL - GET DASD BUFFER PAGE ADDRESS
K1 SPACE AVAILABLE
YES
NO
09 A4 10D1 10E1 15F4 15F5 15K1 16B2 16D2 FNOTE
MSG429
A4 RESET BUFFER ERROR FLAG -SFBRECER
A4
B4 ANY BUFFERS WRITTEN
NO
YES
MSG429F
B5 DMKFRET CALL - FRET SPOOL FILE BLOCK
C4 PRTPUR PURGE OUTPUT FILE
05 E5
C5 PRTDONE CLEAR FILE AND DEVICE
05 E5
TO: A4 16F2 16J2

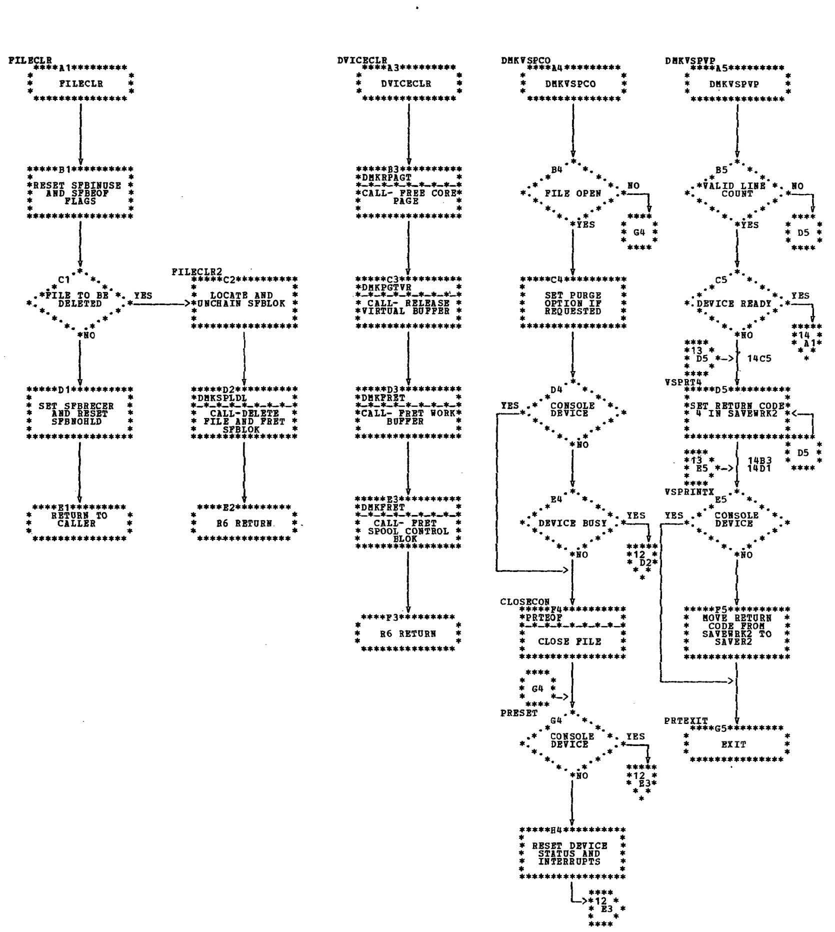

10 A1 09F2
TSTERR
A1 RESET SFBRECER FLAG - RESET ERROR INDICATER
B1 PREVIOUS BUFFER PRESENT
NO
09 F2
YES
C1 DMKRPAGT CALL - GET PREVIOUS BUFFER
D1 READ ERROR
YES
09 A4
NO
E1 SPOOL WRITE ERROR
NO
09 F2
YES
09 A4
10 A2 09G2
SETOPTS
A2 MOVE FILE CLASS TO SFBLOK
B2 DMKSPLCV CALL- CLOSE OUTPUT FILE
10 C2 09H3
PRTDONE
C2 DMKRPAGT CALL-RELEASE STORAGE PAGE
D2 DMKPGTVR CALL-RELEASE VIRTUAL BUFFER
E2 DMKFRET CALL- FRET WORK BUFFER
F2 DMKFRET CALL- FRET SPOOL CONTROL BLOK
G2 R6 RETURN
10 A3 01E3 06C2
SENSE
A3 SENSE
B3 PCITEST TEST FOR PCI INTERRUPT
C3 SET CE AND DE IN CSW
D3 DEVICE READY
YES
NO
E3 SET INTREQ IN SENSE
F3 SENSMOVE MOVE SENSE TO USER STORAGE
G3 GETCCW GET NEXT,CCW
H3 CHAIN DATA OR CMD CHAINING
CD
CC
CHKDEVIC
J3 READER OR PRINTER
PRT
06 A2
RD
01 E3
10 A4 03J1
VSP006
A4 SET PENDING FLAGS
03 G2
10 A5 03H2
EXIT2
A5 RESET CLOSE PENDING FLAG, VDEVCPCL
B5 ANY CPEXBLOKS
NO
03 D2
YES
C5 DMKSTKCP CALL - STACK CLOSE CPEXBLOK FOR THIS UER
03 C2

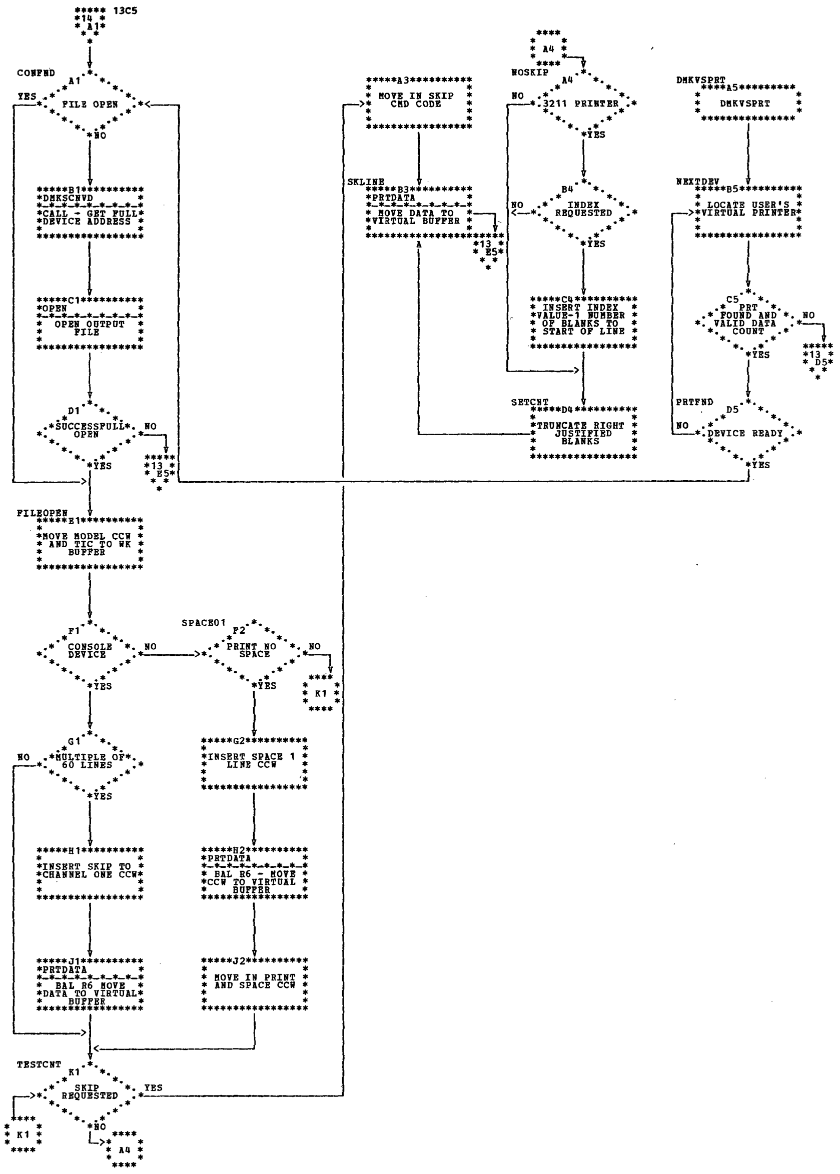

RDRDATA
A2
RDRDATA
B2
GET REC COUNT AND ADDRESS OF NEXT REC
RDRCOUNT
C2
BUFFER EMPTY
NO
YES
D2
LAST BUFFER OF FILE
NO
YES
A4
E2
SET SFBEOF INDICATE END OF FILE
RDREND
F2
RETURN TO CALLER
NEXTCARD
C3
FORM CONTROL OR NOP
YES
NO
D3
MOVE DATA FROM BUFFER TO WK BUFFER
A4
NEXTPAGE
A4
DMKRPAGT
CALL - GET NEXT BUFFER
B4
LOCATE SPOOL RECBLOK FOR THIS CYLINDER
C4
RECBLOK FOUND
YES
NO
D4
DMKFREE
CALL-GET AREA FOR NEW RECBLOK
E4
BUILD RECBLOK AND CHAIN TO SFBREC
SETREC
F4
SET ALLOC BIT FOR THIS BUFFER

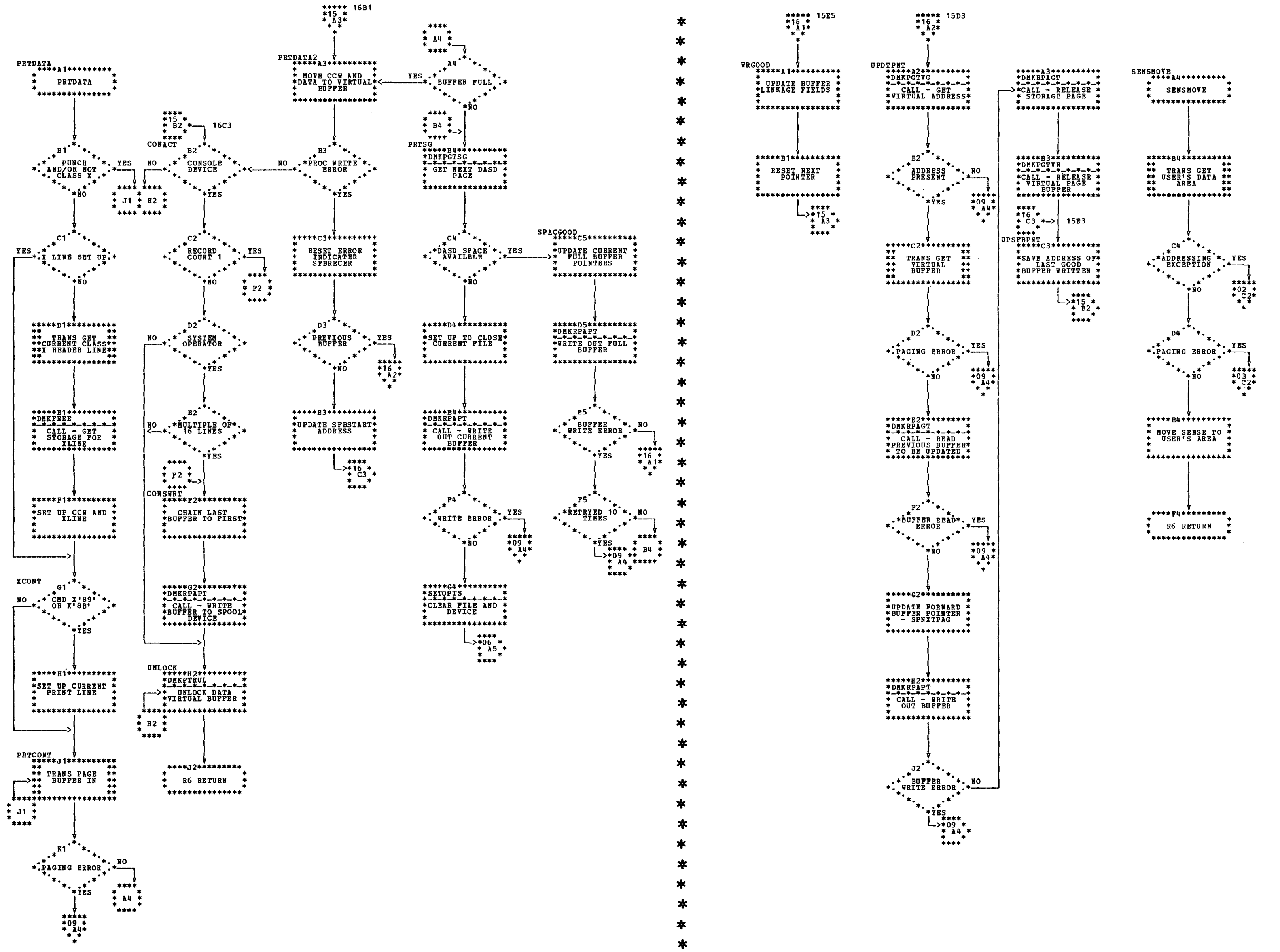


| DMKVSP -- Virtual Spooling Manager (Parts 11 and 12 of 19)

| DMKVSP -- Virtual Spooling Manager (Parts 13 and 14 of 19)

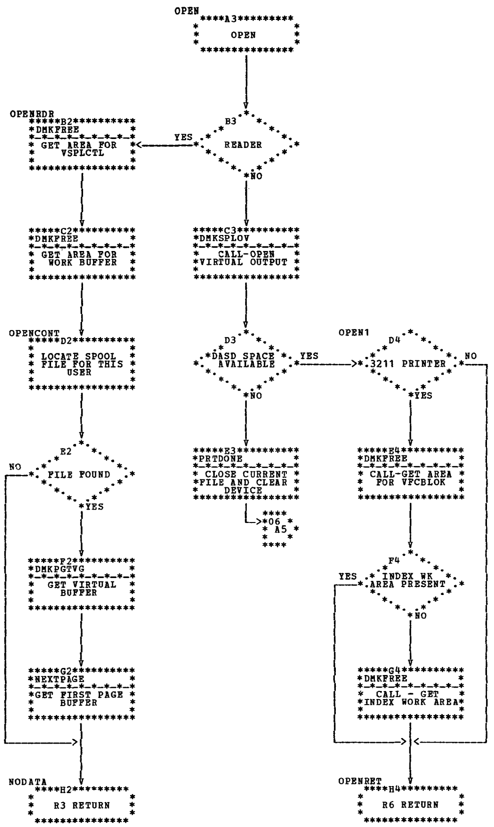

FILECLR
A1 FILECLR
B1 RESET SFBINUSE AND SFBEOF FLAGS
C1 FILE TO BE DELETED
YES
NO
FILECLR2
C2 LOCATE AND UNCHAIN SFBLOK
D1 SET SFBRECER AND RESET SFBNOHLD
D2 DMKSPLDL CALL-DELETE FILE AND FRET SFBLOK
E1 RETURN TO CALLER
E2 R6 RETURN
DVICECLR
A3 DVICECLR
B3 DMKRPAGT CALL- FREE CORE PAGE
C3 DMKPGTVR CALL- RELEASE VIRTUAL BUFFER
D3 DMKFRET CALL- FRET WORK BUFFER
E3 DMKFRET CALL- FRET SPOOL CONTROL BLOK
F3 R6 RETURN
DMKVSPCO
A4 DMKVSPCO
B4 FILE OPEN
NO
G4
YES
C4 SET PURGE OPTION IF REQUESTED
D4 CONSOLE DEVICE
YES
NO
E4 DEVICE BUSY
YES
12 D2
NO
CLOSECON
F4 PRTEOF CLOSE FILE
G4
PRESET
G4 CONSOLE DEVICE
YES
12 E3
NO
H4 RESET DEVICE STATUS AND INTERRUPTS
12 E3
DMKVSPVP
A5 DMKVSPVP
B5 VALID LINE COUNT
NO
D5
YES
C5 DEVICE READY
YES
14 A1
NO
13 D5
14C5
VSPRT4
D5 SET RETURN CODE 4 IN SAVEWRK2
D5
13 E5
14B3
14D1
VSPRINTX
E5 CONSOLE DEVICE
YES
NO
F5 MOVE RETURN CODE FROM SAVEWRK2 TO SAVER2
PRTEXIT
G5 EXIT

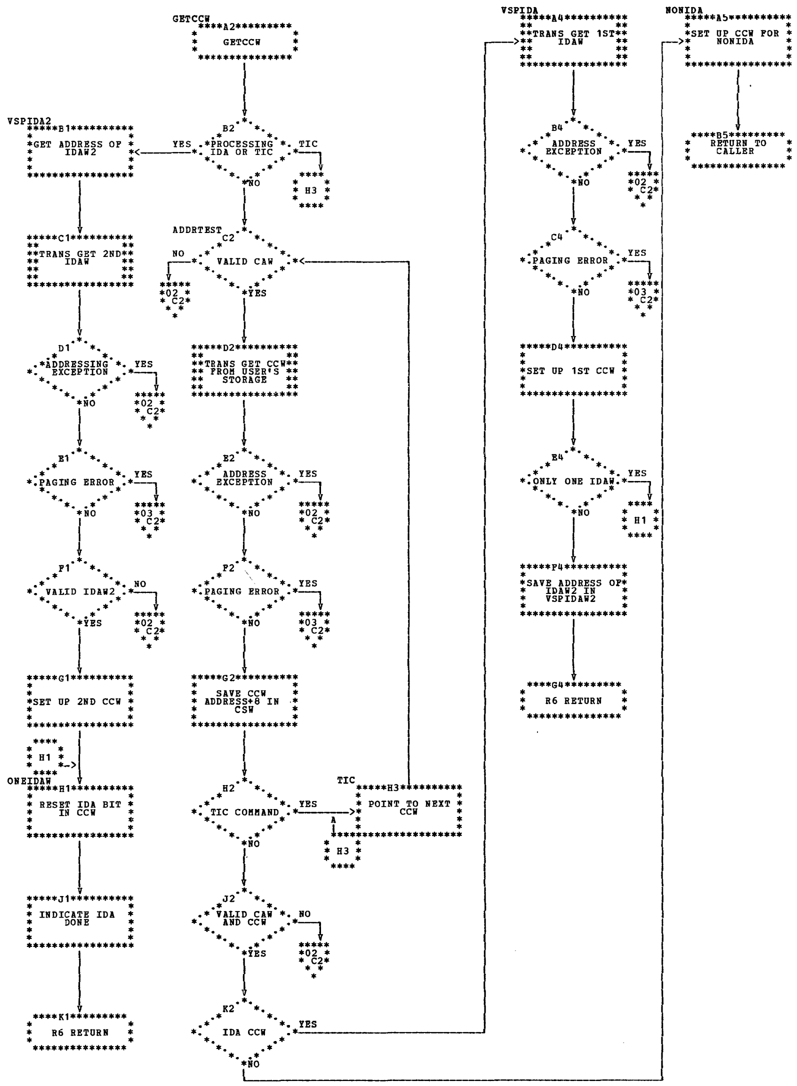

14 A1
13C5
CONFRD
A1 FILE OPEN
YES
NO
B1 DMKSCNVD CALL - GET FULL DEVICE ADDRESS
C1 OPEN OPEN OUTPUT FILE
D1 SUCCESSFULL OPEN
NO
13 E5
YES
FILEOPEN
E1 MOVE MODEL CCW AND TIC TO WK BUFFER
F1 CONSOLE DEVICE
NO
YES
SPACE01
F2 PRINT NO SPACE
NO
K1
YES
G1 MULTIPLE OF 60 LINES
NO
YES
G2 INSERT SPACE 1 LINE CCW
H1 INSERT SKIP TO CHANNEL ONE CCW
H2 PRTDATA BAL R6 - MOVE CCW TO VIRTUAL BUFFER
J1 PRTDATA BAL R6 MOVE DATA TO VIRTUAL BUFFER
J2 MOVE IN PRINT AND SPACE CCW
TESTCNT
K1 SKIP REQUESTED
YES
NO
K1
A4
A3 MOVE IN SKIP CMD CODE
SKLINE
B3 PRTDATA MOVE DATA TO VIRTUAL BUFFER
A
13 E5
A4
NOSKIP
A4 3211 PRINTER
NO
YES
B4 INDEX REQUESTED
NO
YES
C4 INSERT INDEX VALUE-1 NUMBER OF BLANKS TO START OF LINE
SETCNT
D4 TRUNCATE RIGHT JUSTIFIED BLANKS
DMKVSPRT
A5 DMKVSPRT
NEXTDEV
B5 LOCATE USER'S VIRTUAL PRINTER
C5 PRT FOUND AND VALID DATA COUNT
NO
13 D5
YES
PRTFND
D5 DEVICE READY
NO
YES

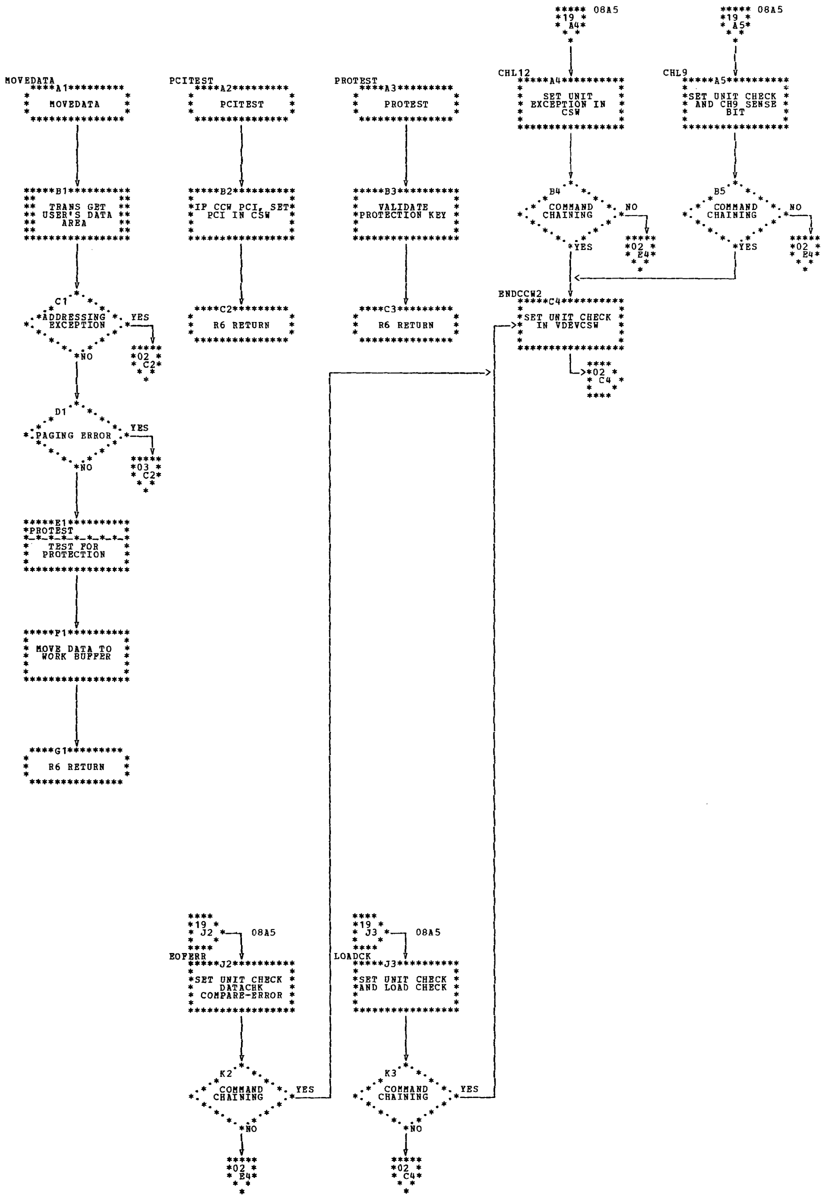

PRTDATA
PRTDATA
PUNCH AND/OR NOT CLASS X
YES
NO
J1
H2
X LINE SET UP
TRANS GET CURRENT CLASS X HEADER LINE
DMKFREE CALL - GET STORAGE FOR XLINE
SET UP CCW AND XLINE
XCONT
CMD X'89' OR X'8B'
SET UP CURRENT PRINT LINE
PRTCONT
TRANS PAGE BUFFER IN
PAGING ERROR
09 A4
A4
15 B2 16C3
CONACT
CONSOLE DEVICE
RECORD COUNT 1
F2
SYSTEM OPERATOR
MULTIPLE OF 16 LINES
CONSWRT
CHAIN LAST BUFFER TO FIRST
DMKRPAPT CALL - WRITE BUFFER TO SPOOL DEVICE
UNLOCK
DMKPTRUL UNLOCK DATA VIRTUAL BUFFER
R6 RETURN
15 A3 16B1
PRTDATA2
MOVE CCW AND DATA TO VIRTUAL BUFFER
PROC WRITE ERROR
RESET ERROR INDICATER SFBRECER
PREVIOUS BUFFER
16 A2
UPDATE SFBSTART ADDRESS
16 C3
BUFFER FULL
PRTSG
DMKPGTSG GET NEXT DASD PAGE
DASD SPACE AVAILBLE
SET UP TO CLOSE CURRENT FILE
DMKRPAPT CALL - WRITE OUT CURRENT BUFFER
WRITE ERROR
09 A4
SETOPTS CLEAR FILE AND DEVICE
06 A5
SPACGOOD
UPDATE CURRENT FULL BUFFER POINTERS
DMKRPAPT WRITE OUT FULL BUFFER
BUFFER WRITE ERROR
16 A1
RETRYED 10 TIMES
B4
16 A1 15E5
WRGOOD
UPDATE BUFFER LINKAGE FIELDS
RESET NEXT POINTER
15 A3
16 A2 15D3
UPDTPNT
DMKPGTVG CALL - GET VIRTUAL ADDRESS
ADDRESS PRESENT
TRANS GET VIRTUAL BUFFER
PAGING ERROR
DMKRPAGT CALL - READ PREVIOUS BUFFER TO BE UPDATED
BUFFER READ ERROR
UPDATE FORWARD BUFFER POINTER - SPNXTPAG
DMKRPAPT CALL - WRITE OUT BUFFER
BUFFER WRITE ERROR
DMKRPAGT CALL - RELEASE STORAGE PAGE
DMKPGTVR CALL - RELEASE VIRTUAL PAGE BUFFER
16 C3 15E3
UPSFBPNT
SAVE ADDRESS OF LAST GOOD BUFFER WRITTEN
15 B2
SENSMOVE
SENSMOVE
TRANS GET USER'S DATA AREA
ADDRESSING EXCEPTION
02 C2
PAGING ERROR
03 C2
MOVE SENSE TO USER'S AREA
R6 RETURN

| DMKVSP -- Virtual Spooling Manager (Parts 15 and 16 of 19)

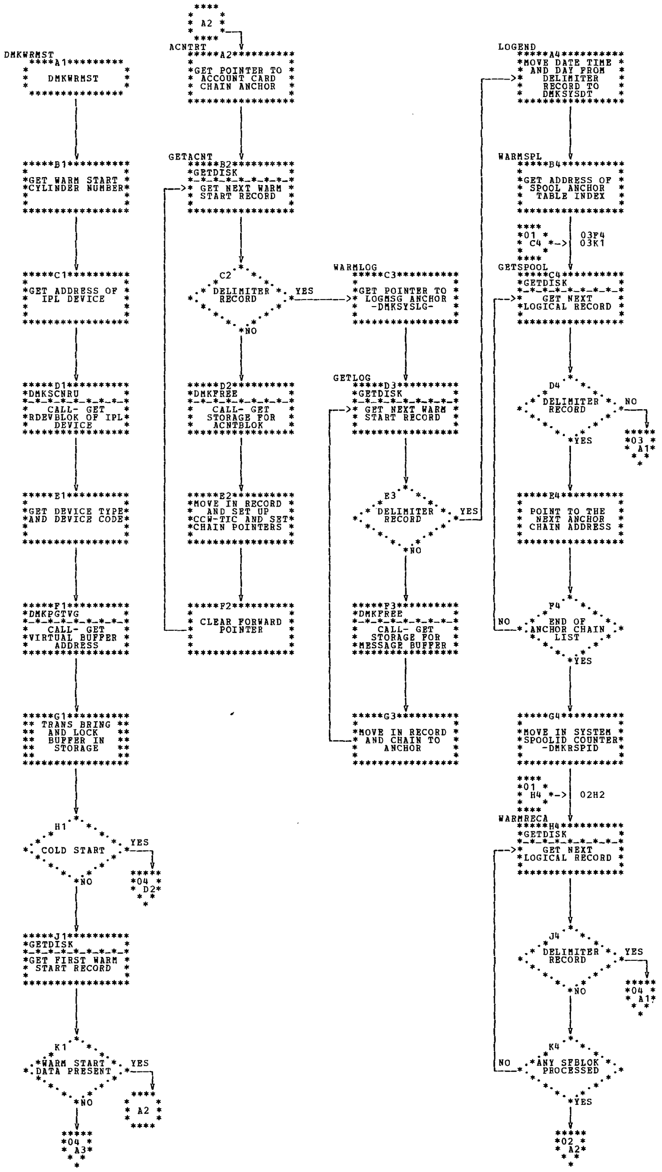
DMKVSP -- Virtual Spooling Manager (Parts 17 and 18 of 19)

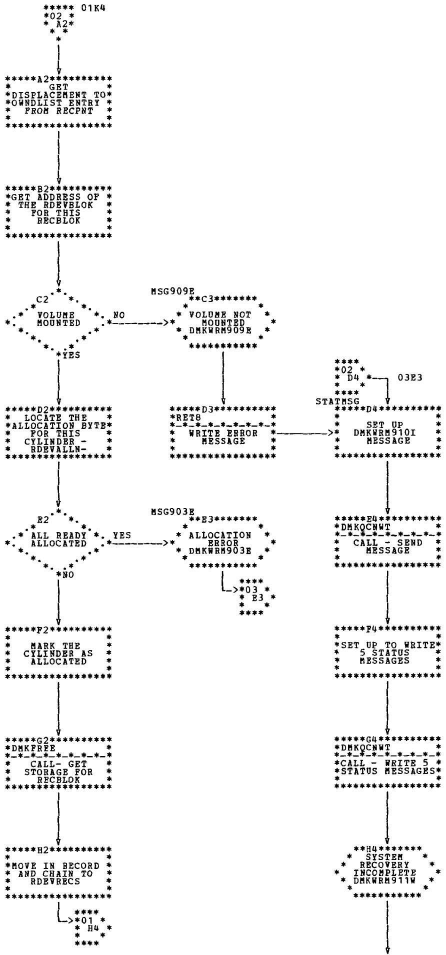






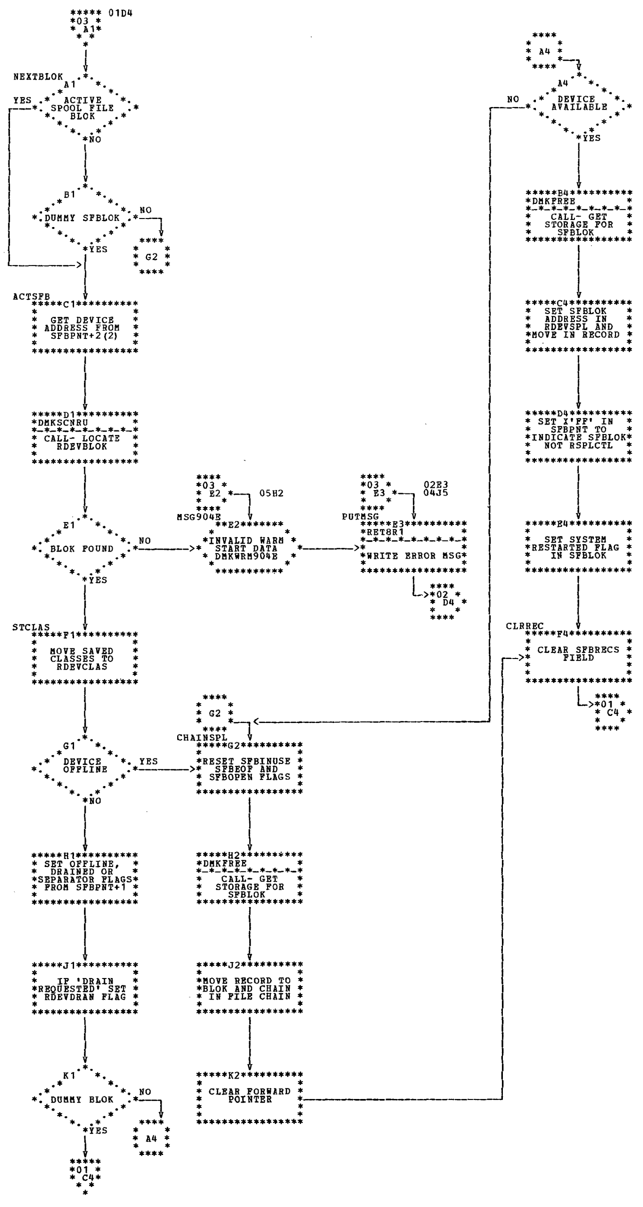
| DMKVSP -- Virtual Spooling Manager (Part 19 of 19)

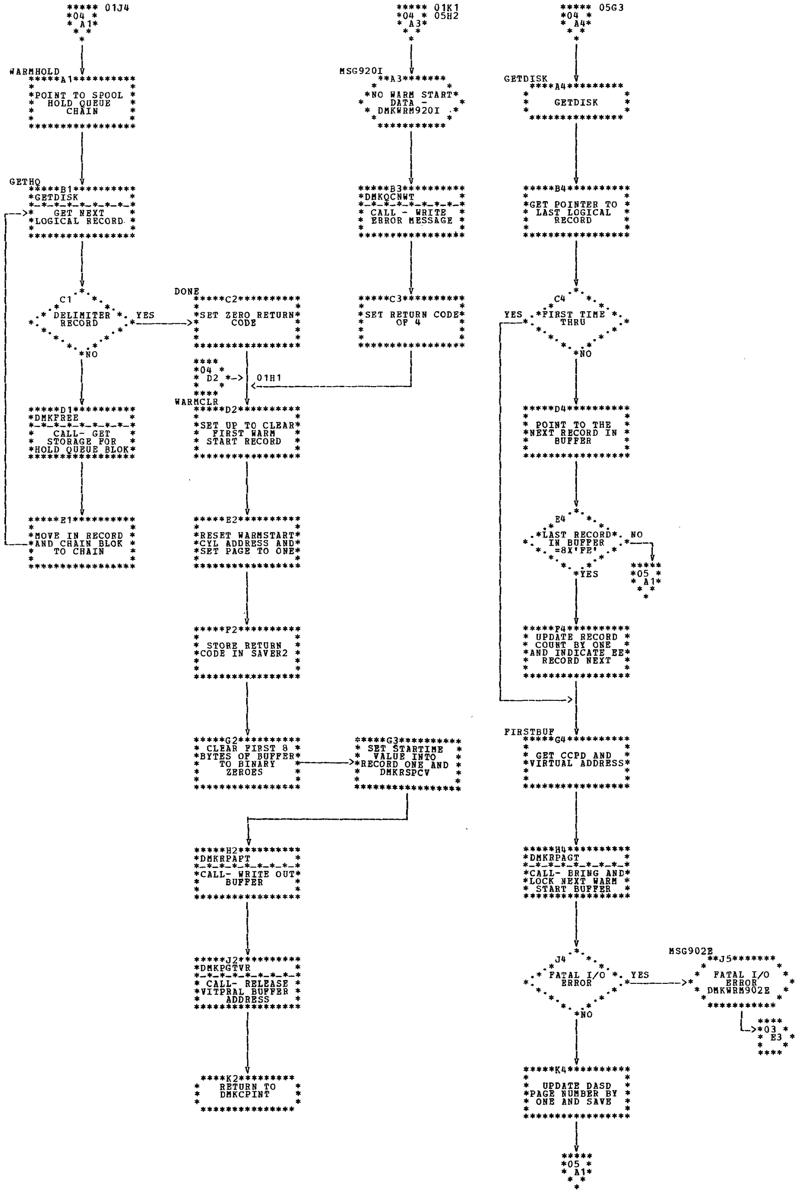

DMKWRMST
A1
DMKWRMST
B1
GET WARM START CYLINDER NUMBER
C1
GET ADDRESS OF IPL DEVICE
D1
DMKSCNRU
CALL- GET RDEVBLOK OF IPL DEVICE
E1
GET DEVICE TYPE AND DEVICE CODE
F1
DMKPGTVG
CALL- GET VIRTUAL BUFFER ADDRESS
G1
TRANS BRING AND LOCK BUFFER IN STORAGE
H1
COLD START
YES
04 D2
NO
J1
GETDISK
GET FIRST WARM START RECORD
K1
WARM START DATA PRESENT
YES
A2
NO
04 A3
A2
ACNTRT
A2
GET POINTER TO ACCOUNT CARD CHAIN ANCHOR
GETACNT
B2
GETDISK
GET NEXT WARM START RECORD
C2
DELIMITER RECORD
YES
NO
D2
DMKFREE
CALL- GET STORAGE FOR ACNTBLOK
E2
MOVE IN RECORD AND SET UP CCW-TIC AND SET CHAIN POINTERS
F2
CLEAR FORWARD POINTER
WARMLOG
C3
GET POINTER TO LOGMSG ANCHOR -DMKSYSLG-
GETLOG
D3
GETDISK
GET NEXT WARM START RECORD
E3
DELIMITER RECORD
YES
NO
F3
DMKFREE
CALL- GET STORAGE FOR MESSAGE BUFFER
G3
MOVE IN RECORD AND CHAIN TO ANCHOR
LOGEND
A4
MOVE DATE TIME AND DAY FROM DELIMITER RECORD TO DMKSYSDT
WARMSPL
B4
GET ADDRESS OF SPOOL ANCHOR TABLE INDEX
01 C4
03F4
03K1
GETSPOOL
C4
GETDISK
GET NEXT LOGICAL RECORD
D4
DELIMITER RECORD
NO
03 A1
YES
E4
POINT TO THE NEXT ANCHOR CHAIN ADDRESS
F4
END OF ANCHOR CHAIN LIST
NO
YES
G4
MOVE IN SYSTEM SPOOLID COUNTER -DMKRSPID
01 H4
02H2
WARMRECA
H4
GETDISK
GET NEXT LOGICAL RECORD
J4
DELIMITER RECORD
YES
04 A1
NO
K4
ANY SFBLOK PROCESSED
NO
YES
02 A2

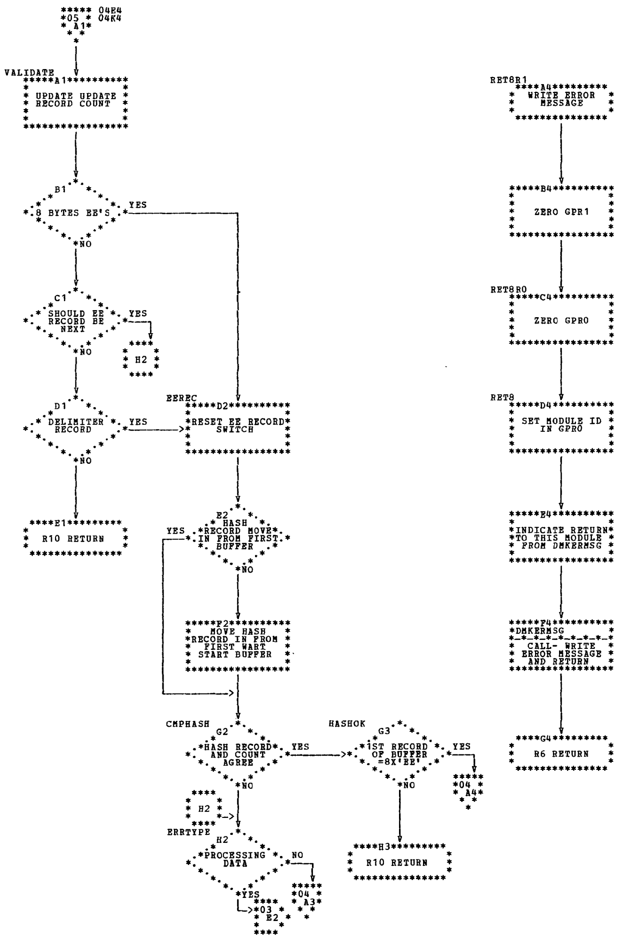

02 A2
01K4
A2
GET DISPLACEMENT TO OWNDLIST ENTRY FROM RECPNT
B2
GET ADDRESS OF THE RDEVBLOK FOR THIS RECBLOK
C2
VOLUME MOUNTED
NO
YES
MSG909E
C3
VOLUME NOT MOUNTED DMKWRM909E
D2
LOCATE THE ALLOCATION BYTE FOR THIS CYLINDER - RDEVALLN-
D3
RET8
WRITE ERROR MESSAGE
02 D4
03E3
STATMSG
D4
SET UP DMKWRM910I MESSAGE
E2
ALL READY ALLOCATED
YES
NO
MSG903E
E3
ALLOCATION ERROR DMKWRM903E
03 E3
E4
DMKQCNWT
CALL - SEND MESSAGE
F2
MARK THE CYLINDER AS ALLOCATED
F4
SET UP TO WRITE 5 STATUS MESSAGES
G2
DMKFREE
CALL- GET STORAGE FOR RECBLOK
G4
DMKQCNWT
CALL - WRITE 5 STATUS MESSAGES
H2
MOVE IN RECORD AND CHAIN TO RDEVRECS
01 H4
H4
SYSTEM RECOVERY INCOMPLETE DMKWRM911W
J4
DMKQCNWT
CALL - WRITE MESSAGE DMKWRM911W
K4
ERROR WAIT PSW 009

DMKWRM -- Warm Start - (Parts 1 and 2 of 5)

DMKWRM -- Warm Start (Parts 3 and 4 of 5)


01D4
03 A1
NEXTBLOK
A1
ACTIVE SPOOL FILE BLOK
YES
NO
B1
DUMMY SFBLOK
NO
G2
YES
ACTSFB
C1
GET DEVICE ADDRESS FROM SFBPNT+2(2)
D1
DMKSCNRU
CALL- LOCATE RDEVBLOK
E1
BLOK FOUND
NO
YES
03 E2
05H2
MSG904E
E2
INVALID WARM START DATA DMKWRM904E
03 E3
02E3
04J5
PUTMSG
E3
RET8R1
WRITE ERROR MSG
02 D4
STCLAS
F1
MOVE SAVED CLASSES TO RDEVCLAS
G1
DEVICE OFFLINE
YES
NO
G2
CHAINSPL
G2
RESET SFBINUSE SFBEOF AND SFBOPEN FLAGS
H1
SET OFFLINE, DRAINED OR SEPARATOR FLAGS FROM SFBPNT+1
H2
DMKFREE
CALL- GET STORAGE FOR SFBLOK
J1
IF 'DRAIN REQUESTED' SET RDEVDRAN FLAG
J2
MOVE RECORD TO BLOK AND CHAIN IN FILE CHAIN
K1
DUMMY BLOK
NO
A4
YES
01 C4
K2
CLEAR FORWARD POINTER
A4
A4
DEVICE AVAILABLE
NO
YES
B4
DMKFREE
CALL- GET STORAGE FOR SFBLOK
C4
SET SFBLOK ADDRESS IN RDEVSPL AND MOVE IN RECORD
D4
SET X'FF' IN SFBPNT TO INDICATE SFBLOK NOT RSPLCTL
E4
SET SYSTEM RESTARTED FLAG IN SFBLOK
CLRREC
F4
CLEAR SFBRECS FIELD
01 C4


01J4
04 A1
WARMHOLD
A1
POINT TO SPOOL HOLD QUEUE CHAIN
GETHQ
B1
GETDISK
GET NEXT LOGICAL RECORD
C1
DELIMITER RECORD
YES
NO
D1
DMKFREE
CALL- GET STORAGE FOR HOLD QUEUE BLOK
E1
MOVE IN RECORD AND CHAIN BLOK TO CHAIN
DONE
C2
SET ZERO RETURN CODE
04 D2
01H1
WARMCLR
D2
SET UP TO CLEAR FIRST WARM START RECORD
E2
RESET WARMSTART CYL ADDRESS AND SET PAGE TO ONE
F2
STORE RETURN CODE IN SAVER2
G2
CLEAR FIRST 8 BYTES OF BUFFER TO BINARY ZEROES
G3
SET STARTIME VALUE INTO RECORD ONE AND DMKRSPCV
H2
DMKRPAPT
CALL- WRITE OUT BUFFER
J2
DMKPGTVR
CALL- RELEASE VITPRAL BUFFER ADDRESS
K2
RETURN TO DMKCPINT
01K1
05H2
04 A3
MSG920I
A3
NO WARM START DATA - DMKWRM920I
B3
DMKQCNWT
CALL - WRITE ERROR MESSAGE
C3
SET RETURN CODE OF 4
05G3
04 A4
GETDISK
A4
GETDISK
B4
GET POINTER TO LAST LOGICAL RECORD
C4
FIRST TIME THRU
YES
NO
D4
POINT TO THE NEXT RECORD IN BUFFER
E4
LAST RECORD IN BUFFER =8X'FE'
NO
05 A1
YES
F4
UPDATE RECORD COUNT BY ONE AND INDICATE EE RECORD NEXT
FIRSTBUF
G4
GET CCPD AND VIRTUAL ADDRESS
H4
DMKRPAGT
CALL- BRING AND LOCK NEXT WARM START BUFFER
J4
FATAL I/O ERROR
YES
NO
MSG902E
J5
FATAL I/O ERROR DMKWRM902E
03 E3
K4
UPDATE DASD PAGE NUMBER BY ONE AND SAVE
05 A1



DMKWRM -- Warm Start (Part 5 of 5)

DIRECTORIES

MODULE/ENTRY POINT DIRECTORY

The directory contains much of the CP cross-reference information for the manual.

Name contains 1 of 2 possible items:

Modules – Six alphabetic characters that begin with DMK.

Entry Points – Seven or eight alphameric characters that begin with DMK. The fourth, fifth, and sixth characters of the entry point identify the module.

Comments Contains:
For modules, the function and a list of the data area that are used.

For entry points, the function.

MOD indicates:
Method of Operation Diagram that the entry point is in.

Chart indicates:
Page of the flowchart for a module that the entry point is.

Calls To indicates:
For modules the entry points in other modules that this module calls (Calls To).

Called By indicates:
For entry points, the modules that call it (Called By).

Name	Comments	MOD	Chart	Calls To	Called By
DMKACO	Accounting Routines			DMKCVTBH	
	Data Areas Used			DMKDSPCH	
	ACNTBLOK			DMKERMSG	
	IOBLOK			DMKFREE	
	RDEVBLOK			DMKFRET	
	SAVEAREA			DMKIOSQR	
	SYSLOCS			DMKPTRUL	
				DMKQCNWT	
				DMKRSPEX	

Name	Comments	MOD	Chart	Calls To	Called By
				DMKSTKCP	
				DMKSTKIO	
DMKACODV	Create account card for VDEVBLOK		1		DMKCPV
					DMKDIA
					DMKUSO
					DMKVDB
DMKACOFF	Create account card for VMBLOK		1		DMKCPV
DMKACON	User entry point		2		DMKLOG
					DMKUSO
DMKACOPU	Punch queued account cards		3		DMKRSP
DMKACOQU	Queue request to punch cards		3		
DMKACOTM	Process use time message		2		DMKCPV
					DMKCQG
					DMKUSO
DMKBLD	Process Blocks			DMKCVTBH	
	Data Areas Used			DMKERMSG	
	PGTABLE			DMKFREE	
	RDEVBLOK			DMKFRET	
	SEGTABLE			DMKSCNRD	
	SWPTABLE				
	VMBLOK				
DMKBLDRL	Release real segment, page, and		2		DMKDEF
	swap tables.				DMKUSO
DMKBLDRT	Build real segment, page, and		1		DMKCNS
	swap tables.				DMKCPI
					DMKDEF
					DMKLOG
DMKBLDVM	Build a new VMBLOK		3		DMKCNS
					DMKDIA
DMKCCH	Channel Check Handler			DMKCVTBH	
	Data Area Used:			DMKFREE	
	CCHREC			DMKIOECC	
	IOBLOK			DMKQCNWT	
	IOERBLOK			DMKSCNRU	
	PSA				
	RCHBLOK				
	RCUBLOK				
	RDEVBLOK				
	SAVEAREA				
	VMBLOK				
DMKCCHIS	Channel check and CSW stored				
	after SIO.	8B2	1		DMKIOS
DMKCCHNT	Channel check from I/O interrupt		3		DMKIOS
DMKCCHRT	Print error message.		3		DMKIOE

Name	Comments	MOD	Chart	Calls To	Called By
DMKCCW	Translate CCWs			DMKDIASM	
	Data Areas Used:			DMKFREE	
	IOBLOK			DMKFRET	
	RDEVBLOK			DMKISMTR	
	SAVEAREA			DMKPTRAN	
	VDEVBLOK			DMKPTRLK	
	VMBLOK			DMKUNTRS	
DMKCCWSB	Get SEEK arguments.		30		DMKTRC
DMKCCWTR	Translate user's CCWs.		1		DMKGEN
					DMKVIO
DMKCDB	Process DISPLAY, DCP, DUMP, and			DMKCVTBD	
	DMCP commands.			DMKCVTBH	
	Data Areas Used			DMKCVTDB	
	BUFFER			DMKCVTFP	
	ECBLOK			DMKCVTHB	
	PGTABLE			DMKDSPCH	
	SAVEAREA			DMKERMSG	
	VMBLOK			DMKFREE	
				DMKFRET	
				DMKPTRAN	
				DMKQCNWT	
				DMKSCNFD	
				DMKVATAB	
				DMKVATBC	
				DMKVATMD	
				DMKVSPRT	
DMKCDBDC	Display real storage				
	(DCP Command).		4		DMKCFM
DMKCDBDI	Display virtual storage				
	(DISPLAY command).		1		DMKCFM
DMKCDBDM	Dump real storage in spooled				
	printer (DMCP command).		4		DMKCFM
DMKCDBDU	Dump virtual storage on				
	spooled printer (DUMP command).		3		DMKCFM
DMKCDS	Process STORE and STCP				
	commands.			DMKCVTBH	
	Data Areas Used:			DMKCVTDB	
	ECBLOK			DMKCVTHB	
	SAVEAREA			DMKERMSG	
	TRQBLOK			DMKPTRAN	
	VMBLOK			DMKQCNWT	
				DMKSCNFD	
				DMKVATAB	
				DMKVATBC	

Name	Comments	MOD	Chart	Calls To	Called By
				DMKVATMD	
DMKCDSCP	Store into real storage (STCP command).		1		DMKCFM
DMKCDSTO	Store into virtual storage (STORE command).		1		DMKCFM
DMKCFD	Process LOCATE and ADSTOP commands. Data Areas Used: ECBLOK RCHBLOK RDEVBLOK SAVEAREA VCHBLOK VCUBLOK VDEVBLOK VMBLOK			DMKCVTBH DMKCVTHB DMKERMSG DMKFREE DMKFRET DMKQCNWT DMKSCNAU DMKSCNFD DMKSCNRU DMKSCNVU	
DMKCFDAD	Stop virtual machine at specified address (ADSTOP command).		3		DMKCFM
DMKCFDLO	Display address of real device blocks, or VMBLOK and/or virtual device blocks (LOCATE command).		1		DMKCFM
DMKCFG	Command Processor Data Areas Used: ECBLOK PSA RDEVBLOK SAVEAREA SAVTABLE SYSTABLE VDEVBLOK VMBLOK			DMKCVTBH DMKERMSG DMKFREE DMKFRET DMKPTRAN DMKPTRUL DMKQCNRD DMKQCNWT DMKRPAPT DMKSCNFD DMKSCNVS DMKSCNVU	
DMKCFGSV	SAVESYS command processor.		1		DMKCFM
DMKCFM	Process SLEEP, BEGIN and QUERY command and direct all other CP commands to current module. Data Areas Used: BUFFER SAVEAREA VMBLOK			DMKCDBDC DMKCDBDI DMKCDBDM DMKCDBDU DMKCDSCP DMKCDSTO DMKCFDAD DMKCFDLO DMKCFMBE	

Name	Comments	MOD	Chart	Calls To	Called By
DMKCFM (cont)				DMKCFMQU	
				DMKCFMSL	
				DMKCFPEX	
				DMKCFPIP	
				DMKCFPNR	
				DMKCFPRS	
				DMKCFPRW	
				DMKCFPRY	
				DMKCFPSR	
				DMKCFSET	
				DMKCFTRM	
				DMKCPVAC	
				DMKCPVDS	
				DMKCPVEN	
				DMKCPVH	
				DMKCPVLK	
				DMKCPVRY	
				DMKCPVSH	
				DMKCPVSV	
				DMKCPVUL	
				DMKCQGEN	
				DMKCQPRV	
				DMKCSOBS	
				DMKCSODR	
				DMKCSOFL	
				DMKCSOLD	
				DMKCSORP	
				DMKCSOSP	
				DMKCSOST	
				DMKCSOVL	
				DMKCSPCL	
				DMKCSPFR	
				DMKCSPHL	
				DMKCSPSP	
				DMKCSUCH	
				DMKCSUOR	
				DMKCSUPU	
				DMKCSUTR	
				DMKCVTHB	
				DMKDEFIN	
				DMKDIAL	
				DMKDSPCH	
				DMKERMSG	
				DMKFREE	
				DMKFRET	
				DMKLNKIN	

Name	Comments	MOD	Chart	Calls To	Called By
DMKCFM (cont)				DMKLOGON DMKMCCCL DMKMSGEC DMKMSGMS DMKMSGWN DMKQCNRD DMKQCNWT DMKSCNFD DMKSCNVU DMKSTKCP DMKTRACE DMKUSODS DMKUSOFL DMKUSOLG DMKVDBAT DMKVDBDE	
DMKCFMAT	Simulate attention interrupt to virtual machine.		3		DMKCNS
DMKCFMBE	BEGIN command processor		4		DMKCFM
DMKCFMBK	Attention interrupt twice from a terminal.	7B3	1		DMKCNS DMKDSP DMKHVC DMKIOE DMKMCH DMKPRG DMKPSA DMKPTR DMKTRC DMKVCN
DMKCFMEN	DIAGNOSE code 8.		2		DMKCPI DMKHVC DMKLNK
DMKCFMQU	QUERY command processor (initial)		5		DMKCFM
DMKCFMSL	SLEEP command processor.		4		DMKCFM
DMKCFP	Simulate the operators console for the virtual machine. Data Areas Used IOBLOK RDEVBLOK SAVEAREA VCHBLOK VCUBLOK VDEVBLOK			DMKCVTBH DMKCVTDB DMKCVTHB DMKDSPCH DMKDIADR DMKERMSG DMKFREE DMKFRET DMKPGSPO	

Name	Comments	MOD	Chart	Calls To	Called By
	VMBLOK			DMKPTRAN DMKPTRUL DMKRPAGT DMKQCNWT DMKSCHRT DMKSCNFD DMKSCNVS DMKSCNVU DMKSTKCP DMKSTKIO DMKUNTFR DMKVATBC DMKVCARD DMKVDBRL DMKVSPCO DMKVSPCR	
DMKCFPII	IPL from LOGON		5		DMKLOG
DMKCFPIP	IPL command processor.	7B3.1	9		DMKCFM
DMKCFPRD	Reset a virtual device.		2		DMKDEF DMKVDB
DMKCFPRR	Process system resets from other routines		1		DMKDEF DMKMCH DMKUSO
DMKCFS	Process SET command Data Areas Used: CORTABLE IRMBLOK SAVEAREA RDEVBLOK VMBLOK			DMKCVTBH DMKCVTDB DMKCVTDT DMKCVTHB DMKERMSG DMKFREE DMKFRET DMKMCHMS DMKQCNRD DMKQCNWT DMKSCHRT DMKSCNAU DMKSCNFD DMKSCNRU DMKSTKIO	
DMKCFSET	SET command processor		1		DMKCFM
DMKCFT	Process user's terminal options. Data Areas Used:			DMKCVTDB DMKERMSG	

Name	Comments	MOD	Chart	Calls To	Called By
	RDEVBLOK			DMKQCNWT	
	SAVEAREA			DMKSCNFD	
	SYSLOCS				
	VMBLOK				
DMKCFTRM	TERMINAL command processor.		1		DMKCFM
DMKCKP	Save pertinent data			DMKSAVRS	
	when check point occurs				
	Data Areas Used:				
	RCHBLOK				
	RCUBLOK				
	RDEVBLCK				
	SAVEAREA				
	VMBLOK				
DMKCKPT	Check Point program.		1		DMKSAV
DMKCNS	Real Console Terminal Manager				
DMKCNSED			20		
DMKCNSID			9		
DMKCNSIN			1		DMKIOS
DMKCNSNM			15		
DMKCNSOF			11		DMKCPV
					DMKIOS
					DMKQCN
DMKCPB	Simulate the operator's console			DMKCFPRD	
	for the virtual machine.			DMKCFPRR	
	Data Areas Used:			DMKCVTBH	
	IOBLOK			DMKCVTHB	
	RDEVBLOK			DMKERMSG	
	SAVEAREA			DMKFREE	
	VCHBLOK			DMKFRET	
	VCUBLOK			DMKIOSQR	
	VDEVBLOK			DMKPGSPO	
	VMBLOK			DMKPTRAN	
				DMKQCNWT	
				DMKSCNFD	
				DMKSCNVU	
				DMKVATBC	
				DMKVATMD	
DMKCPBEX	Process the EXTERNAL command.		2		DMKCFM
DMKCPBNR	Process the NOTREADY command.		3		DMKCFM
DMKCPBRS	Process the RESET command.		3		DMKCFM
DMKCPBRW	Process the REWIND command.		4		DMKCFM
DMKCPBRY	Process the READY command.		3		DMKCFM
DMKCPBSR	Process the SYSTEM command.		1		DMKCFM
DMKCPI	Prepare VM/370 for operation			DMKBLDRT	

Name	Comments	MOD	Chart	Calls To	Called By
DMKCPI	Data Areas Used:			DMKCFMEN	
(cont)	ALOCBLOK			DMKCQGFI	
	CORTABLE			DMKCSOSD	
	PAGTABLE			DMKCVTBD	
	RDEVBLOK			DMKCVTBH	
	SEGTABLE			DMKCVTDT	
	SWPTABLE			DMKDSPCH	
	VMBLOK			DMKFRETR	
				DMKIOEFL	
				DMKIOSQR	
				DMKLOGOP	
				DMKQCNRD	
				DMKSCHST	
				DMKSCNRD	
				DMKSCNVS	
				DMKWRMST	
DMKCPIEM	Saves necessary data		2		DMKCNS
	for automatic re-IPL.				
DMKCPINT	Start initialization of the		1		DMKSAV
	VM/370 control program.	5B1			
DMKCPV	Command Processor			DMKACODV	
	Data Areas Used:			DMKACOFF	
	BUFFER			DMKACOTM	
	CORTABLE			DMKCVTHB	
	IOBLOK			DMKCVTBD	
	RDEVBLOK			DMKCVTBH	
	SAVEAREA			DMKDMPRS	
				DMKERMSG	
				DMKFREE	
				DMKFRET	
				DMKIOSQR	
				DMKPTRAN	
				DMKPTRUL	
				DMKQCNRD	
				DMKQCNWT	
				DMKRPAPT	
				DMKSCNAU	
				DMKSCNFD	
				DMKSCNRU	
				DMKSCNVS	
				DMKSCNVU	

Name	Comments	MOD	Chart	Calls To	Called By
DMKCPVAC	ACNT command processor.		6		DMKCFM
DMKCPVDS	DISABLE command processor.		3		DMKCFM
DMKCPVEN	ENABLE command processor.		1		DMKCFM
DMKCPVH	HALT command processor.		12		DMKCFM
DMKCPVLK	LOCK command processor.		4		DMKCFM
DMKCPVRY	VARY command processor.		7		DMKCFM
DMKCPVSH	SHUTDOWN command processor.		5		DMKCFM
DMKCPVUL	UNLOCK command processor.		5		DMKCFM
DMKCQG	Process QUERY command.			DMKACOTM	
	Data Areas Used:			DMKCVTBD	
	BUFFER			DMKCVTBH	
	SAVEAREA			DMKCVTDB	
	SFBLOK			DMKCVTDT	
	VCHBLOK			DMKCVTHB	
	VCUBLOK			DMKERMSG	
	VDEVBLOK			DMKFREE	
	VMBLOK			DMKFRET	
				DMKQCNWT	
				DMKSCNAU	
				DMKSCNFD	
				DMKSCNRD	
				DMKSCNVN	
				DMKSCNVU	
DMKCQGEN	QUERY command processor for Class G users.		1		DMKCFM
DMKCQGFI	QUERY command processor for Class G users.		1		DMKCPI
DMKCQGLG	QUERY command processor for Class G users.		2		DMKLOG DMKLOG
DMKCQP	Process QUERY command			DMKCVTBD	
	Data Areas Used:			DMKCVTBH	
	RCHBLOK			DMKCVTHB	
	RCUBLOK			DMKERMSG	
	RDEVBLOK			DMKFREE	
	SAVEAREA			DMKFRET	
	VCHBLOK			DMKQCNWT	
	VCUBLOK			DMKSCNAU	
	VDEVBLOK			DMKSCNFD	
	VMBLOK			DMKSCNRD	
				DMKSCNRN	
				DMKSCNRU	
				DMKSCNVD	
				DMKSCNVU	

Name	Comments	MOD	Chart	Calls To	Called By
DMKCQPRV	QUERY command processor for Classes B, E, and G users.		1		DMKCFM
DMKCSO	Process real spooling commands for real unit record devices. Data Areas Used: IOBLOK RDEVBLOK SAVEAREA SAVEWRK2 SFBLOK VMBLOK			DMKCVTBH DMKCVTDB DMKCVTHB DMKDSPCH DMKERMSG DMKFREE DMKFRET DMKIOSQR DMKPTRAN DMKPTRUL DMKQCNWT DMKRSPEX DMKSCNFD DMKSCNRU DMKSCNVU DMKSPLDL DMKSTKIO	
DMKCSOBS	BACKSPACE command processor.		7		DMKCFM
DMKCSODR	DRAIN command processor.		5		DMKCFM
DMKCSOFL	FLUSH command processor.		1		DMKCFM
DMKCSOLD	LOADBUF command processor.		7		DMKCFM
DMKCSORP	REPEAT command processor.		6		DMKCFM
DMKCSOSD	Start entry point for warm start.		5		DMKCPI DMKCSP DMKCSU
DMKCSOSP	SPACE command processor.		2		DMKCFM
DMKCSOST	START command processor.		3		DMKCFM
DMKCSOVL	LOAD virtual Forms Control Buffer.		12		DMKCFM
DMKCSP	Process Class D and G spooling commands. Data Areas Used: SAVEAREA SFBLOK VDEVBLOK VMBLOK			DMKDCOSD DMKCVTDB DMKCVTHB DMKERMSG DMKFREE DMKFRET DMKQCNWT DMKSCNFD DMKSCNVU DMKSPLDC DMKUDRFU DMKVSPCO DMKVSPCR	

Name	Comments	MOD	Chart	Calls To	Called By
DMKCSPCL	CLOSE command processor.		1		DMKCFM
DMKCSPFR	FREE command processor.		3		DMKCFM
DMKCSPHL	HOLD command processor.		3		DMKCFM
DMKCSPSP	SPOOL command processor.		4		DMKCFM
DMKCSU	Process class D and G spooling			DMKCSOSD	
	commands.			DMKCVTBD	
	<u>Data Areas Used</u>:			DMKCVTDB	
	SAVEAREA			DMKERMSG	
	SFBLOK			DMKFREE	
	VDEVBLOK			DMKFRET	
	VMBLOK			DMKQCNWT	
				DMKSCNFD	
				DMKCSNAU	
				DMKSPLDL	
				DMKUDRFU	
DMKCSUCH	CHANGE command processor.		1		DMKCFM
DMKCSUOR	ORDER command processor.		4		DMKCFM
DMKCSUPU	PURGE command processor.		6		DMKCFM
DMKCSUTR	TRANSFER command processor.		8		DMKCFM
DMKCVT	Convert Routines			None	
	<u>Data Areas Used</u>:				
	BALRSAVE				
	TEMPSAVE				
DMKCVTBD	Convert binary to EBCDIC decimal.		3		DMKCDB
					DMKCPI
					DMKCPV
					DMKCQG
					DMKCQP
					DMKCSU
					DMKDEF
					DMKDIA
					DMKLNK
					DMKLOG
					DMKRSP
					DMKSEP
					DMKSPL
					DMKUSO
					DMKVDB
DMKCVTBH	Convert binary to EBCDIC		1		DMKACO
	hexadecimal.				DMKBLD
					DMKCCH
					DMKCDB
					DMKCDS
					DMKCFD

Name	Comments	MOD	Chart	Calls To	Called By
					DMKCFP DMKCFS DMKCQG DMKCQP DMKCPI DMKCPV DMKCSO DMKDEF DMKDIA DMKLNK DMKLOG DMKMSW DMKPSA DMKRSP DMKSCH DMKSEP DMKSPL DMKTRC DMKUSO DMKVCA DMKVDB
DMKCVTDB	Convert EBCDIC decimal to binary.		3		DMKCDB DMKCDS DMKCFP DMKCFS DMKCFT DMKCQG DMKCSO DMKCSP DMKCSU DMKDEF DMKMSG
DMKCVTDT	Convert data and time to EBCDIC.		4		DMKCFS DMKCPI DMKCQG DMKHVC DMKIOF DMKLOG DMKMID DMKQCN DMKRSP DMKSEP DMKSPL DMKUSO DMKVSP

Name	Comments	MOD	Chart	Calls To	Called By
DMKCVTFP	Convert floating-point hexadecimal to binary.		1		DMKCDB
DMKCVTHB	Convert EBCDIC hexadecimal to binary.		1		DMKCDB DMKCDS DMKCFD DMKCFM DMKCFP DMKCFS DMKCPV DMKCQG DMKCQP DMKCSO DMKCSP DMKDEF DMKDIA DMKLNK DMKVDB
DMKDAS	DASD ERP Data Areas Used: IOBLOK IOERBLOK RDEVBLOK SAVEAREA VMBLOK			DMKCVTBH DMKFREE DMKFRET DMKIOESD DMKIOSQR DMKMSWR DMKQCNWT DMKSCNRU	
DMKDASER	Retry the failing DASD channel program.		1		DMKIOS
DMKDASRD	Process unsolicited Device End interrupts.		9		DMKIOS
DMKDASSD	Collect 3330 Statistical Data		8		DMKCPV
DMKDEF	Define a virtual device or storage. Data Areas Used: VDEVBLOK VMBLOK SAVEAREA			DMKBLDRL DMKBLDRT DMKCFPRD DMKCFPRR DMKCVTBD DMKCVTBH DMKCVTDB DMKCVTHB DMKERMSG DMKFREE DMKPGSPO DMKQCNWT DMKSCNFD	

Name	Comments	MOD	Chart	Calls To	Called By
				DMKSCNVD	
				DMKSCNVN	
				DMKSCNVU	
				DMKUDRFU	
				DMKUDRLK	
				DMKUDRRD	
				DMKUDRRV	
				DMKUDRUL	
				DMKVDSDF	
DMKDEFIN	DEFINE command processor.		1		DMKCFM
DMKDGD	DASD I/O			DMKFREE	
	Data Areas Used:			DMKFRET	
	IOBLOK			DMKIOSQV	
	SAVEAREA			DMKPTRUL	
	VDEVBLCK			DMKSCNVU	
	VMBLOK				
DMKDGDDK	Perform disk I/O.		1		DMKHVC
DMKDIA	Connect terminal to			DMKACODV	
	virtual 270X or convert			DMKBLDVM	
	virtual channel to			DMKCVTBD	
	channel adapters.			DMKCVTEH	
	Data Areas Used:			DMKCVTHB	
	SAVEAREA			DMKDSPCH	
	VCHBLOK			DMKERMSG	
	VCUBLOK			DMKFREE	
	VDEVBLOK			DMKFRET	
	VMBLCK			DMKIOSQR	
				DMKQCNWT	
				DMKSCNAU	
				DMKCNSFD	
				DMKSCNRD	
				DMKSCNVD	
				DMKSCNVU	
				DMKSTKIO	
DMKDIACP	COUPLE command processor.		8		
DMKDIADR	Drop dialed line to virtual				DMKCCW
	System		5		DMKCFD
DMKDIAL	DIAL command processor.		1		DMKCFM
DMKDIASM	Simulate status for undialed lines		6		DMKVCA
DMKDMP	Dump system and re-IPL				
DMKDMPDK	WRITE dump to output device.		1		DMKPRG
					DMKPSA

Name	Comments	MOD	Chart	Calls To	Called By
DMKDMPRS	Re-IPL system.		3		DMKCPV
					DMKCPV
DMKDRD	Process spool files			DMKFREE	
	<u>Data Areas Used</u>:			DMKPGTSD	
	SAVEAREA			DMKQRVKY	
	SFBLOK			DMKPTRAN	
	SWPTABLE			DMKRPAGT	
	UDEVBLCK			DMKSCNVU	
	VSPCTL			DMKVSPCR	
	VMBLOK				
DMKDRDDD	Delete system dump spool file.		7		DMKSPL
DMKDRDER	Diagnose interface to input spool files.		1		DMKHVC
DMKDRDMP	Diagnose read of system dump spool files.		6		DMKHVC
DMKDRDSY	Diagnose read of system symbol table.		7		DMKHVC
DMKDSP	Dispatcher			DMKCFMBK	
	<u>Data Areas Used</u>:			DMKQCNWT	
	CPEXBLOK			DMKSCHDL	
	ECBLOK			DMKSCNVU	
	ICBLOK			DMKTRCEX	
	VCHBLOK			DMKTRCIO	
	VCUBLOK			DMKTRCIT	
	VDEVBLOK			DMKUSOFF	
	VMBLCK			DMKVATAB	
				DMKVATBC	
				DMKVATMD	
DMKDSPA	Fast user re-dispatch.		9		DMKPRV
DMKDSPB	Process new virtual PSW and dispatch.		9		DMKPRG
					DMKPRU
					DMKPSA
					DMKVIO
DMKDSPCH	Update timers and dispatch users.	2B	1		DMKACO
					DMKCDB
					DMKCFM
					DMKCFD
					DMKCPI
					DMKCSO
					DMKDIA
					DMKEPS
					DMKGEN
					DMKHVC
					DMKIOE
					DMKIOS

Name	Comments	MOD	Chart	Calls To	Called By
DMKDSPCH					DMKMCH
(cont)					DMKPAG
					DMKPGT
					DMKPRG
					DMKPRV
					DMKPSA
					DMKPTR
					DMKRPA
					DMKTRC
					DMKTMR
					DMKUSO
					DMKVCA
					DMKVIO
					DMKVSP
					DMKWRM
DMKEIG	2880 Channel logout analysis.				
	<u>Data</u> <u>Areas</u> <u>Used</u>:				
	CCHREC				
DMKEIG80	2880 Channel logout analysis.		1		DMKCCH
DMKEPSWD	Entry point in DMKLNK.				DMKLOG
DMKERM	Message writer			DMKFREE	
	<u>Data</u> <u>Areas</u> <u>Used</u>:			DMKFRET	
	SAVEAREA			DMKQCNWT	
	VMBLOK				
DMKERMSG	Message writer.		1		DMKACO
					DMKBLD
					DMKCDB
					DMKCDS
					DMKCFD
					DMKCFM
					DMKCFP
					DMKCFS
					DMKCFD
					DMKCPV
					DMKCQG
					DMKCQP
					DMKCSO
					DMKCSP
					DMKCSU
					DMKDEF
					DMKDIA

Name	Comments	MOD	Chart	Calls To	Called By
DMKERMSG (cont)					DMKLNK DMKLOG DMKMSG DMKRSP DMKTRA DMKUSO DMKVCH DMKVDB DMKVSP DMKWRM
DMKFRE	Free storage manager. Data Areas Used: FREESAVE			DMKPTRFR DMKPTRFT	
DMKFREE	Get space from free storage.	6B1	1		(General
DMKFRET	Return space to free storage.	6B2	9		entries)
DMKFRETR	Return space to free storage; do not release pages.		8		DMKCPI DMKPTR
DMKGEN	Process CMS I/O error recovery. Data Areas Used: IOBLOK IOERBLOK RDEVBLOK SAVEAREA VDEVBLOK VMBLOK			DMKCCWTR DMKDSPCH DMKFREE DMKFRET DMKIOSQV DMKSCNVU DMKUNTFR DMKUNTRN	
DMKGENIO	Process CMS I/O error recovery.		1		DMKHVC
DMKGRA	Read or write to the primary system console. Data Areas Used: None.				
DMKGRARD	Read from primary system console.		1		
DMKGRAWT	Write to primary system console.		3		DMKCKP DMKRSP
DMKHVC	Process DIAGNOSE Data Areas Used: RDEVBLOK VDEVBLOK VMBLOK			DMKCFMBK DMKCFMEN DMKCVTDT DMKDGDDK DMKDRDER DMKDRDMP DMKDRDSY DMKDSPCH	

Name	Comments	MOD	Chart	Calls To	Called By
DMKHVC (cont)				DMKFREE DMKGENIO DMKIOEFM DMKPGSPP DMKPRGSM DMKPRVKY DMKPTRAN DMKRPAGT DMKSCNRU DMKSCNVU DMKUDRDS	
DMKHVCAL	Process a diagnose instruction from a virtual machine.		1		DMKPRV

Name	Comments	MOD	Chart	Calls To	Called By
DMKIOE	Initiate errcr recording. <u>Data Areas Used</u>: IDBLOK ICERBLOK IRMBLOK RDEVELCK SAVEAREA VMBLOK			DMKCCHRT DMKCFMBK DMKDSPCH DMKFREE DMKFRET DMKIOFC1 DMKIOFE1 DMKIOFIN DMKIOFM1 DMKIOFR1 DMKIOGF1 DMKIOGF2 DMKQCNWT	
DMKICECC	Channel error from SIO in DMKICS, CC=1.		5		DMKCCH
DMKIOECH	Stack channel check recording from DMKCCH.		5		DMKIOF
DMKIOECJ	Stack channel check recording from ERP.		6		DMKIOF
DMKIOEFL	Locate starting page record for recording.		6		DMKCPI
DMKIOEFM	Clear and format recording area on disk.		7		DMKHVC
DMKIOEMC	Machine check recording.		6		DMKMCH
DMKICEMH	Stack machine check recording.		6		DMKIOF
DMKIOENV	Stack environmental recording		5		DMKIOF
DMKICEOB	Stack OBR recording.		4		DMKIOF
DMKICERC	Stack erase request.		7		DMKIOF
DMKIOERR	Schedule recording for unit check, channel data check, and hardware environmental counts.		1		DMKIOS
DMKIOESD	Record 3330 data		5		DMKDAS
DMKIOF	Perform error recording <u>Data Areas Used</u>: CPDEVCOD ICBIOK RDEVBLOK SAVEAREA VMBLOK			DMKCVTDT DMKIOECH DMKIOECJ DMKIOEMH DMKIOENV DMKIOEOB DMKIOERC DMKFRET DMKPGTVG DMKPGTVR DMKPTRAN DMKPTRUL	

Name	Comments	MOD	Chart	Calls To	Called By
				DMKQCNWT DMKRPAGT DMKRPAPT DMKSTKCP	
DMKIOFC1	Records channel check error from SIO in DMKIOS, CC=1.		12		DMKIOE
DMKIOFE1	Records 3330 and 2305 environmental counters.		9		DMKIOE
DMKIOFIN	Initialize pointers to available recording pages.		13		DMKIOE
DMKIOFM1	Record machine checks.		13		DMKIOE
DMKIOFR1	Record outboard I/O errors.		1		DMKIOE
DMKIOG	Initializaticn/cleanup error recording routines. Data Areas Used: RDEVBLOK SAVEAREA VMBLOK			DMKFREE DMKPGTVG DMKPGTVR DMKQCNWT DMKRPAGT DMKRPAPT DMKSCNRU	
DMKIOGF1	Initialize RMS functions.		1		DMKIOE
DMKIOGF2	Erase recording area on disk.		4		DMKIOE
DMKIOS	I/O Supervisor Data Areas Used: CPEXBLOK IOBLOK RCHBLOK RCUBLOK RDEVBLOK SAVEAREA TEMPSAVE VDEVBLOK VMBLOK			DMKCCHIS DMKCCHNT DMKCNSIN DMKDASER DMKDSPCH DMKFREE DMKFRET DMKIOERR DMKRSPER DMKSCHDL DMKSCNRU DMKSTKCP DMKSTKIO DMKTAPER DMKTRCSI DMKVIODC	
DMKIOSHA	Halt an active device and drain all interrupts.		1		DMKCFP DMKCPV
DMKIOSIN	Process an I/O interrupt.	1B4.1	3		
DMKIOSQR	Schedule control program generated I/O.	1B4.3	1		DMKACO DMKCFP DMKCNS

Name	Comments	MOD	Chart	Calls To	Called By
DMKIOSQR (cont)					DMKCPI DMKCPV DMKCSO DMKDAS DMKDIA DMKPAG DMKPRG DMKQCN DMKRSP DMKSEP DMKSPL DMKUDR DMKVDB
DMKIOSQV	Schedule virtual machine I/O	1B4.3	1		DMKDGD DMKGEN DMKVIO
DMKIOSRW	Process IOBLOK used for rewind.		11		DMKCFP DMKVDB
DMKISM	Process ISAM CCWs Data Areas Used: IOBLOK RCWTASK SAVEAREA VMBLOK			DMKFREE	
DMKISMTR	Find and modify ISAM CCWs		1		DMKCCW
DMKLNK	Link to a virtual device. Data Areas Used: BUFFER RDEVBLOK SAVEAREA UDEVBLOK UDIRBLOK VDEVBLOK VMBLOK			DMKCVTBD DMKCVTBH DMKCVTHB DMKEPSWD DMKERMSG DMKFREE DMKFRET DMKQCNRD DMKQCNWT DMKSCNAU DMKSCNFD DMKSCNLI DMKSCNVN DMKSCNVS DMKSCNVU DMKUDRFD DMKUDRFU	

Name	Comments	MOD	Chart	Calls To	Called By
DMKLNK (cont)				DMKUDRRV DMKVDBRL DMKVDSLK	
DMKEPSWD	Enter a password. (Part of LINK)		1		DMKLOG
DMKLNKIN	LINK command processor.		1		DMKCFM
DMKLNKSB	LINK subroutines.		9		DMKLOG
DMKLOC	Lock system resource by unique name. Data Areas Used: CPEXBLOK			DMKFREE DMKFRET	
DMKLOCK	Lock a name.		1		DMKLNK
DMKLOCKD	Dequeue a locked name.		1		DMKDEF DMKLNK DMKUDR DMKUSO DMKVDB
DMKLOCKQ	Queue or lock a name.		2		DMKDEF DMKUDR DMKUSO DMKVDB
DMKLOCKT	Test if a name is locked.		1		
DMKLOG	Logon a user or operator. Data Areas Used: BUFFER RDEVBLOK SAVEAREA UDEVBLOK UDIRBLOK UMACBLOK VCHBLOK VCUBLOK VDEVBLOK VMBLOK			DMKACON DMKBLDRT DMKCFPII DMKCQGFI DMKCVTBD DMKCVTBH DMKCVTDT DMKEPSWD DMKERMSG DMKFREE DMKFRET DMKLNKSB DMKQCNWT DMKSCHCP DMKSCH80 DMKSCNAU DMKSCNFD DMKSCNRD DMKSCNRU DMKSCNVN DMKSCNVU DMKTMRCK	

Name	Comments	MOD	Chart	Calls To	Called By
DMKLOG (cont)				DMKUDRFU	
				DMKUDRRD	
				DMKUDRRV	
				DMKUSOFL	
				DMKVDSAT	
				DMKVDSDF	
DMKLOGON	LOGON a user.		1		DMKCFM
DMKLOGOP	LOGON the operator.		9		DMKCPI
DMKMCC	Monitor Command Handler			DMKERMSG	
	Data Areas Used:			DMKFREE	
	CORTABLE			DMKPTRFR	
	MONCOMM			DMKPTRFT	
	PSA			DMKQCNWT	
	VMBLOK			DMKSCNFD	
DMKMCCCL	Process the MONITOR Command		1		DMKCFM
DMKMCH	Machine Check Handler			DMKCFMBK	
	Data Areas Used:			DMKCFPRR	
	CORTABLE			DMKDMPRS	
	MCHAREA			DMKDSPCH	
	MCRECORD			DMKFREE	
	PAGTABLE			DMKIOEMC	
	PSA			DMKPGSPO	
	SEGTABLE			DMKPTRFT	
	SWPTABLE			DMKQCNWT	
	VMBLOK			DMKSCNFD	
				DMKSTKCP	
				DMKUSOFF	
DMKMCHIN	Process a machine check interrupt.	8B1	1		DMKCPI
DMKMCHMS	Enable or disable soft recording from SET command	8B1	8		DMKCFS
DMKMID	Date Change			DMKCVTDT	
				DMKQCNWT	
				DMKSCHST	
DMKMIDNT	Change system date at midnight.		1		DMKSCH

Name	Comments	MOD	Chart	Calls To	Called By
DMKMSG	Message Handler.			DMKCVTDB	
	Data Areas Used:			DMKERMSG	
	BUFFER			DMKFREE	
	RDEVBLOK			DMKFRET	
	SAVEAREA			DMKQCNRD	
	VDEVBLOK			DMKQCNWT	
	VMBLOK			DMKSCNAU	
				DMKSCNFD	
DMKMSGEC	ECHO command processor.		2		DMKCFM
DMKMSGMS	MSG command processor.		2		DMKCFM
DMKMSGWN	WNG command processor.		1		DMKCFM
DMKMSW	ERP Message Writer			DMKCVTBH	
	Data Areas Used:			DMKFREE	
	IOBLOK			DMKFRET	
	IOERBLOK			DMKQCNRD	
	RDEVBLOK			DMKQCNWT	
	SAVEAREA			DMKSCNRN	
	VMBLOK				
DMKMSWR	ERP message writer.		1		DMKCNS
					DMKDAS
					DMKSPL
					DMKTAP
DMKNEM	Translate op-codes.				
	Data Areas Used:				
	SAVEAREA				
DMKNEMOP	Translate op-codes.		1		DMKTRC
DMKPAG	Perform paging I/O			DMKDSPCH	
	Data Areas Used:			DMKFREE	
	CORTABLE			DMKIOSQR	
	CPEXBLOK			DMKSTKCP	
	IOBLOK				
	RDEVBLOK				
	SAVEAREA				
	SWPTABLE				
	VMBLOK				
DMKPAGIO	Process requests for paging I/O.	1B3.5	1		DMKPTR
					DMKRPA

Name	Comments	MOD	Chart	Calls To	Called By
DMKPGS	Release virtual storage. Data Areas Used: CORTABLE PAGTABLE SAVAREA SEGTABLE SHRTABLE SWPTABLE VMBLOK			DMKFREE DMKFRET DMKPGTPR DMKPTRAN DMKPTRFT	
DMKPGSPO	Release user's entire virtual storage.	1B3.4	2		DMKCFP DMKDEF DMKMCH DMKUSO
DMKPGSPP	Release a specified area of virtual storage.	1B3.4	1		DMKHVC
DMKPGT	DASD storage management. Data Areas Used: ALOCBLOK CPEXBLOK PAGTABLE RDEVBLOK RECBLOK SAVEAREA SWPTABLE VMBLOK			DMKDSPCH DMKFREE DMKFRET DMKQCNWT DMKSTKCP	
DMKPGTPG	Allocate DASD storage for paging.	1B3.3	1		DMKPTR
DMKPGTPR	Release DASD storage used for paging.	1B3.3	5		DMKPGS DMKPTR DMKRPA

Name	Comments	MOD	Chart	Calls To	Called By
DMKPGTPR (cont)					DMKSPL DMKVSP
DMKPGTSD	Release DASD storage used for spooling.		4		DMKDRD DMKSPL
DMKPGTSG	Allocate DASD storage for spooling.		3		DMKRSP DMKSPL DMKVSP
DMKPGTSR	Release DASD storage used for a complete spool file.	1B3.3	5		DMKSPL
DMKPGTVG	Allocate system virtual storage.	1B3.3	7		DMKIOF DMKIOG DMKLOG DMKRSP DMKSEP DMKSPL DMKUDR DMKVSP DMKWRM
DMKPGTVR	Release system virtual storage.		8		DMKIOF DMKIOG DMKRSP DMKSEP DMKUDR DMKVSP DMKWRM
DMKPRG	Interrupt Handler Data Areas Used: TEMPSAVE VMBLOK			DMKCFMBK DMKDMPDK DMKDSPB DMKDSPCH DMKPRVLG DMKPSAID DMKPTRAN DMKQCNWT DMKTRCPG DMKVATPX DMKVATSX	
DMKPRGIN	Hardware program interrupt.	1B3	1		DMKCPI
DMKPRGRF	Reflect SVC interrupt to virtual machine.	1B3	4		DMKPSA

Name	Comments	MOD	Chart	Calls To	Called By
DMKPRGSM	Simulate virtual program interrupt.		4		DMKHVC DMKPRV DMKTMR DMKVAT
DMKPRV	Simulate privileged operations. Data Areas Used: ECBLOK VMBLOK			DMKDSPA DMKDSPB DMKDSPCH DMKHVCAL DMKPRGSM DMKPTRAN DMKTMRTN DMKTRCPV DMKVATAB DMKVATEX DMKVATLA DMKVATRN DMKVIOEX	
DMKPRVKY	Process virtual storage keys.		4		DMKDRD DMKHVC DMKTMR DMKTRC
DMKPRVLG	Simulate a privileged operation.	1B3.7	1		DMKPRG
DMKPSA	Interrupt Handler Data Areas Used: SAVEAREA SYSLOCS			DMKCFMBK DMKCVTBH DMKDMPDK DMKDSPB DMKFREE DMKPRGRF DMKPTRAN DMKPTRUL DMKQCNCL DMKSCNRD DMKSTKIO DMKTRCSV	
DMKPSADU	Force an SVC 0 type of dump.		2		DMKCPI
DMKPSAEX	External interrupt handler.	1B2	4		DMKCPI
DMKPSAID	Get virtual address for any instruction.		3		DMKPRG

Name	Comments	MOD	Chart	Calls To	Called By
DMKPSARR	Get virtual address for RR instruction.		3		DMKTRC
DMKPSARS	Get virtual address for RS,SI, or SS instruction.		3		DMKTRC
DMKPSARX	Get virtual address for RX instruction.		3		DMKTRC
DMKPSASV	SVC interrupt handler.	1B1	1		DMKCPI
DMKPTR	Real Storage Manager Data Areas Used: CORTABLE PAGTABLE SAVEAREA SEGTABLE SWPTABLE VMBLOK			DMKCFMBK DMKDSPCH DMKFREE DMKFRET DMKFRETR DMKPAGIO DMKPGTPG DMKPGTPR DMKPTRFT DMKQCNWT DMKSCHDL DMKSTKCP	
DMKPTRAN	Translate user virtual storage address to a real storage address.	1B3.2	1		DMKCCW DMKCDB DMKCDS DMKCPV DMKCSO DMKDRD DMKHVC DMKIOF DMKPGS DMKPRG DMKPRV DMKPSA DMKRPA DMKSPL DMKTMR DMKTRC DMKVCA DMKVIO DMKVSP
DMKPTRFD	Release page after writing		11		
DMKPTRFE	Page must be swapped to extend		7		
DMKPTRFR	Get a page of real storage.	1B3.2	6		DMKCPI DMKFRE

Name	Comments	DGM	Count	Calls	Called By
DMKPTRFT	Release a page of real storage	1B3.2	11		[illegible] [illegible] DMKPGT [illegible]
DMKPTRLK	Lock a page of real storage.		13		DMKCCH DMKPSA DMKSPL
DMKPTRUL	Unlock a page of real storage.	1B3.2	12		DMKACO DMKCFP DMKCPV DMKCSO DMKDGD DMKIOF DMKPSA DMKRPA DMKSEP DMKSPL DMKUNT DMKVCA DMKVSP
DMKQCN	Console queue manager Data Areas Used BUFFER CPEXBLOK CONTASK IOBLOK RDEVBLCK SAVEAREA VMBLOK			DMKCVTDT DMKFREE DMKFRET DMKIOSQR DMKSTKCP DMKUSOFF	
DMKQCNCL	Clear message queue.		3		DMKCFS DMKPSA
DMKQCNRD	Queue console read request.		1		DMKCFM DMKCFS DMKCPI DMKCPV DMKEPS DMKLNK DMKMSG DMKMSW
DMKQCNWT	Queue console write request.		1		DMKACO DMKCCH DMKCDB DMKCDS DMKCFD

Name	Comments	MOD	Chart	Calls To	Called By
DMKQCNWT					DMKCFM
(cont)					DMKCFP
					DMKCFS
					DMKCFT
					DMKCPI
					DMKCPV
					DMKCQG
					DMKCQP
					DMKCSO
					DMKCSP
					DMKCSU
					DMKDEF
					DMKDIA
					DMKDSP
					DMKEPS
					DMKERM
					DMKIOE
					DMKIOF
					DMKIOG
					DMKLNK
					DMKLOG
					DMKMCH
					DMKMSG
					DMKMSW
					DMKPGT
					DMKPRG
					DMKPSA
					DMKPTR
					DMKRSP
					DMKSPL
					DMKTRA
					DMKTRC
					DMKUSO
					DMKVCA
					DMKVCH
					DMKVCN
					DMKVDB
					DMKWRM
DMKRPA	Virtual storage mapping.			DMKDSPCH	
	Data Areas Used:			DMKFREE	
	CORTABLE			DMKPAGIO	
	PAGTABLE			DMKPGTPR	
	SAVEAREA			DMKPTRAN	
	SWPTABLE			DMKPTRFT	
	VMBLOK			DMKPTRUL	
				DMKSCHDL	

Name	Comments	MOD	Chart	Calls To	Called By
DMKRPAGT	Page-in from DASD to user's virtual storage.		1		DMKCFP DMKDRD DMKHVC DMKIOF DMKIOG DMKRSP DMKUDR DMKVSP DMKWRM
DMKRPAPT	Page out to DASD from User's virtual storage.		1		DMKCVP DMKIOF DMKIOG DMKRSP DMKVSP DMKWRM
DMKRSE	Real U/R device I/O error handler. __Data Areas Used:__ IOBLOK IOERBLOK RDEVBLOK SAVEAREA VMBLOK			DMKFRET DMKMSWR	
DMKRSERR	Real spool error processing.		1		DMKRSP
DMKRSP	Real spooling manager __Data Areas Used:__ IOBLOK RDEVBLOK RSPLCTL SFBLOK VMBLOK			DMKACOPU DMKCVTBD DMKCVTBH DMKCVTDT DMKERMSG DMKFREE DMKFRET DMKIOSQR DMKPGTSG DMKPGTVG DMKPGTVR DMKQCNWT DMKRPAGT DMKRPAPT DMKRSERR DMKSCNFD DMKSCNRU DMKSEPSP	

Name	Comments	MOD	Chart	Calls To	Called By
DMKRSP (cont)				DMKSPLCR DMKSPLDL DMKSPLOR DMKUDRFU	
DMKRSPER	Processing spooling errors (ERP).		8		DMKIOS
DMKRSPEX	Process spooling operations for real UR devices.	4B2	1		DMKACO DMKCSO DMKIOS DMKSPL
DMKSAV	Save CP nucleus or SYSRES			DMKCPINT	

Name	Comments	MOD	Chart	Calls To	Called By
DMKSAVNC	Write a page image of the control program's nucleus onto DASD.	5B3	2		
DMKSAVRS	Restore a page image copy of the control program's nucleus from DASD into main storage.	5B3	2		DMKCKP
DMKSCH	Scheduler Data Areas Used: CORTABLE TEMPSAVE TRQBLOK VMBLOK			DMKCVTBH DMKFRET DMKMIDNT	
DMKSCHAE	Interrupt from expiration of execution interval.		8		DMKCFS
DMKSCHCP	Interrupt from real CPU timer.		7		DMKLOG
DMKSCHDL	Alters a user's dispatching states.	3B	1		DMKDSP DMKIOS DMKPTR DMKRPA DMKUSO
DMKSCHMD	Interrupt from midnight date change.		8		DMKCPI
DMKSCHRT	Reset a clock comparator interrupt.		7		DMKCFP DMKCFS DMKMID DMKTMR DMKUSO
DMKSCHST	Establish a clock comparator interrupt.		7		DMKCPI DMKMID DMKTMR
DMKSCH80	Interrupt from real timer at storage address 80.		6		DMKCFS DMKLOG
DMKSCN	Scan Routine Data Areas Used: BALRSAVE BUFFER RCHBLOK RCUBLOK VCHBLOK VCUBLOK VDEVBLOK VMBLOK			None	

Name	Comments	MOD	Chart	Calls To	Called By
DMKSCNAU	Find the specified VMBLOK.		3		DMKCFD
					DMKCFS
					DMKCPV
					DMKCQG
					DMKCQP
					DMKCSU
					DMKDIA
					DMKLNK
					DMKLOG
					DMKMSG
					DMKSPL
					DMKUSO
					DMKVDB
					DMKVSP
DMKSCNFD	Find next field in input buffer.		3		DMKCDB
					DMKCDS
					DMKCFD
					DMKCFM
					DMKCFP
					DMKCFS
					DMKCFT
					DMKCPV
					DMKCQG
					DMKCQP
					DMKCSO
					DMKCSP
					DMKCSU
					DMKDEF
					DMKDIA
					DMKEPS
					DMKLNK
					DMKLOG
					DMKMSG
					DMKRSP
					DMKTRC
					DMKUSO
					DMKVDB
DMKSCNLI	Find all links of a minidisk.		3		DMKLNK
DMKSCNRD	Determine real device address.		2		DMKBLD
					DMKCPI
					DMKCQG
					DMKCQP

Name	Comments	MOD	Chart	Calls To	Called By
					DMKDIA
					DMKLOG
					DMKPSA
					DMKTRA
					DMKUSO
					DMKVDB
DMKSCNRN	Find device name for a given real device address.		4		DMKCFD
					DMKCFS
					DMKCNS
					DMKCPV
					DMKCQP
					DMKCSO
					DMKHVC
					DMKIOG
					DMKIOS
					DMKLOG
					DMKMSW
					DMKRSP
					DMKTRC
					DMKVDB
					DMKVCH
					DMKWRM
DMKSCNRU	Find blocks for a specified real device.				
			1		DMKCCH
					DMKCFD
					DMKCFS
					DMKCPV
					DMKCQP
					DMKCSO
					DMKHVC
					DMKIOG
					DMKIOS
					DMKLOG
					DMKRSP
					DMKVCH
					DMKVDB
					DMKWRM
DMKSCNVD	Determine virtual device address.				DMKCQP
					[illegible]
					[illegible]
					[illegible]
					[illegible]

Name	Comments	MOD	Chart	Calls To	Called By
DMKSCNVN	Find device name for a given virtual device address.		4		DMKCQG DMKDEF DMKLNK DMKLOG DMKTRC DMKVDB
DMKSCNVS	Find device for specified serial number.		2		DMKCFP DMKCPI DMKCPV DMKLNK DMKLOG DMKVDB
DMKSCNVU	Find blocks for a specified virtual device address.		1		DMKCFD DMKCFM DMKCFP DMKCPV DMKCQG DMKCQP DMKCSO DMKCSP DMKDEF DMKDGD DMKDIA DMKDRD DMKDSP DMKGEN DMKHVC DMKLNK DMKLOG DMKTRC DMKVCA DMKVCH DMKVDB DMKVDG DMKVIO DMKVSP
DMKSEP	Print/punch output separator <u>Data Areas Used:</u> IOBLOK DMKBOX RDEVBLOK SAVEAREA SFBLOK			DMKCVTBD DMKCVTBH DMKCVTDT DMKDSPCH DMKIOSQR DMKPGTVG DMKPGTVR DMKPTRUL	

Name	Comments	MOD	Chart	Calls To	Called By
DMKSEPSP	Print and punch the respective separators on real spooling devices.		1		DMKRSP
DMKSEV	2870 Channel logout analysis. Data Areas Used: CCHREC				
DMKSEV70	2870 Channel logout analysis.		1		DMKCCH
DMKSIX	2860 Channel logout analysis. Data Areas Used: CCHREC				
DMKSIX60	2860 Channel logout analysis.		1		DMKCCH
DMKSPL	Spool file manager Data Areas Used: CPEXBLOK IOBLOK IOERBLOK OWNDLIST RDEVBLOK RECBLOK RSPLCTL SAVEAREA SFBLOK VCONCTL VCUBLOK VDEVBLOK VMBLOK VSPLCTL			DMKCVTBD DMKCVTDT DMKDRDDD DMKDSPCH DMKFREE DMKFRET DMKIOSQR DMKMSWR DMKPGTSD DMKPGTSG DMKPGTSR DMKPGTVG DMKPTRAN DMKPTRLK DMKPTRUL DMKQCNWT DMKSCNAU DMKSTKCP DMKSTKIO	
DMKSPLCR	Close real reader file.		2		DMKRSP
DMKSPLCV	Close virtual printer or punch file.		1		DMKVSP
DMKSPLDL	Delete spool file buffers.		3		DMKCSO DMKCSP DMKCSU DMKRSP DMKVSP
DMKSPLDR	Delete spool file blocks.		3		
DMKSPLOR	Open real reader file.		1		DMKRSP

Name	Comments	MOD	Chart	Calls To	Called By
DMKSPLOV	Open virtual printer or punch file.		1		DMKVSP
DMKSSP	System Initialization/ configuration. Data Areas Used: RCHBLOK RCUBLOK RDEVBLOK			DMKCPINT DMKCVTBH DMKCVTHB DMKSCNRU	
DMKSSP01	Build real I/O blocks for minimum system.		1		
DMKSTK	Stack I/O Data Areas Used: CPEXBLOK IOBLOK				
DMKSTKCP	Stack a CPEXBLOK		1		DMKACO DMKCFM DMKCNS DMKIOF DMKIOS DMKMCH DMKPAG DMKPGT DMKPTR DMKQCN DMKSPL DMKTRC DMKUSO DMKVCA
DMKSTKIO	Stack an IOBLOK.		1		DMKACO DMKCCW DMKCFP DMKCFS DMKCSO DMKIOS DMKPSA DMKSPL DMKTMR DMKVIO
DMKTAP	Tape ERP Data Areas Used: IOBLOK IOERBLOK			DMKFREE DMKFRET DMKMSW	

Name	Comments	MOD	Chart	Calls To	Called By
DMKTAP (cont)	RDEVBLOK SAVEAREA VMBLOK				
DMKTAPER	Magnetic tape ERP.		1		DMKIOS
DMKTDK	T-disk space manager Data Areas Used: ALOCBLOK RDEVBLOK SAVEAREA				
DMKTDKGT	Get T disk space.		1		DMKVDS
DMKTDKRL	Release T disk space.		2		DMKVDB

Name	Comments	MOD	Chart	Calls To	Called By
DMKTMR	Simulate CPU timer and TOD clock Data Areas Used: ECBLOK TRQBLOK VMBLOK			DMKDSPCH DMKPRGSM DMKPRVKY DMKPTRAN DMKSCHRT DMKSCHST DMKSTKIO	
DMKTMRCK	Simulate virtual clock comparator interrupt.		1		DMKLOG
DMKTMRTN	Simulate timer instruction.		1		DMKPRV
DMKTRA	TRACE command processor. Data Areas Used: TREXT			DMKERMSG DMKFREE DMKFRET DMKQCNWT DMKSCNFD DMKTRCIT DMKTRCPB	
DMKTRACE	TRACE command processor.	7B3.2	1		DMKCFM
DMKTRC	TRACE command routines. Data Areas Used: TREXT			DMKCCWSB DMKCFMBK DMKCVTBH DMKDSPCH DMKFREE DMKFRET DMKNEMOP DMKPRVKY DMKPSARR DMKPSARS DMKPSARX DMKPTRAN DMKQCNWT DMKSCNRD DMKSCNRN DMKSCNVN DMKSCNVU DMKSTKCP DMKVATRN DMKVSPRT	

Name	Comments	MOD	Chart	Calls To	Called By
DMKTRCEX	Process an external interrupt.	7B3.2	1		DMKDSP
DMKTRCIO	Process an I/O interrupt.	7B3.2	6		DMKDSP
DMKTRCIT	Set "SVC B2" for instruction tracing.		9		DMKDSP DMKTRA
DMKTRCND	Forcibly end tracing.		11		
DMKTRCPB	Put back user instructions altered by tracing.		9		DMKCFM DMKTRA
DMKTRCPG	Process a program interrupt.	7B3.2	6		DMKPRG
DMKTRCPV	Process a privileged instruction interrupt.	7B3.2	8		DMKPRV
DMKTRCSI	Process an I/O operation (SIO, TIO, HIO, TCH).	7B3.2	6		DMKIOS DMKVIO
DMKTRCSV	Process an SVC, Branch, or full instruction TRACE.	7B3.2	7		DMKPSA
DMKTRCSW	TRACE virtual and real CSWs.		8		DMKVIO
DMKTRM	Identify type of terminal Data Areas Used: RDEVBLOK SAVEAREA VMBLOK				
DMKTRMID	Determine type of 2741.		1		DMKCNS
DMKUDR	Directory Manager Data Areas Used: IOBLOK OWNDLIST RDEVBLOK SAVEAREA SYSLOCS UDEVBLCK UDLBLOK UDIRBLCK UMACBLOK VMBLOK			DMKFREE DMKFRET DMKIOSQR DMKPGTVG DMKPGTVR DMKPTRAN DMKPTRUL DMKRPAGT	
DMKUDRBV	Build a list of virtual page buffers for each UDIRBLOK page on disk.		4		DMKCPI
DMKUDRDS	Swap active user directory to newly created user directory.		5		DMKHVC

Name	Comments	MOD	Chart	Calls To	Called By
DMKUDRFD	Put specified UDEVBLOK into user's buffer.		3		DMKLNK DMKLOG
DMKUDRFU	Put UDIRBLOK into user's buffer.		2		DMKCSP DMKCSU DMKDEF DMKLNK DMKLOG DMKRSP
DMKUDRLK	Lock user directory.		1		DMKDEF DMKLNK DMKLOG DMKVDB
DMKUDRLT	Test user directory lock.		1		
DMKUDRRD	Put next udevblok into user's buffer.		3		DMKDEF DMKLOG
DMKUDRRV	Release a virtual page used for a buffer.		4		DMKDEF DMKLNK DMKLOG
DMKUDRUL	Unlock user directory.		1		DMKDEF DMKLNK DMKLOG DMKVDB
DMKUNT	Untranslate CCWs and CSWs Data Areas Used: IOBLOK RCWTASK RDEVBLOK SAVEAREA VCHBLOK VDEVBLCK VMBLOK			DMKFRET DMKPTRUL DMKSTKIO	
DMKUNTFR	Release pages and free storage used for CCW chain.		1		DMKCFP DMKGEN DMKVIO
DMKUNTIS	Untranslate ISAM CCWs		5		
DMKUNTRN	Translate a real CSW to a virtual CSW.		1		DMKGEN DMKVIO
DMKUNTRS	Relocate sense byte information.		4		DMKCCW

Name	Comments	MOD	Chart	Calls To	Called By
DMKUSO	Process user termination Data Areas Used: RDEVBLOK SAVEAREA SYSLOCS VMBLOK			DMKACODV DMKACOFF DMKACOTM DMKBLDRL DMKCFPRR DMKCVTBD DMKCVTBH DMKCVTDT DMKDSPCH DMKERMSG DMKFRET DMKPGSPO DMKQCNWT DMKSCHDL DMKSCHRT DMKSCNAU DMKSCNFD DMKSCNRD DMKSTKCP DMKVATBC	
DMKUSODS	DISCONN (disconnect) command processor.		4		DMKCFM
DMKUSOFF	Logoff a user.		4		DMKCCH DMKCDS DMKDSP DMKLOG DMKMCH DMKQCN
DMKUSOFL	FCRCE command processor.		3		DMKCFM DMKLOG
DMKUSCLG	LCGCFF command processor.		1		DMKCFM
DMKVAT	Storage management for EC mode virtual machine. Data Areas Used: BALRSAVE ECBLOK VMBLCK			DMKDSPCH DMKFREE DMKFRET DMKPRGSM DMKPTRAN	
DMKVATAB	Maintain virtual address translation.	1B3.6	1		DMKCDB DMKCDS DMKDSP DMKPRV

Name	Comments	MOD	Chart	Calls To	Called By
DMKVATBC	Return shadow tables to free storage.		3		DMKCDB DMKCDS DMKCFP DMKDSP DMKUSO
DMKVATEX	Simulate page or segment exception.		10		DMKPRV
DMKVATLA	Virtual (shadow) - virtual to virtual address translation.		5		DMKPRV
DMKVATMD	Allocate and initialization shadow table.		2		DMKCDB DMKCDS DMKCFP DMKDSP
DMKVATPX	Process paging exception for a virtual machine that performs paging.		9		DMKPRG
DMKVATRN	Virtual (shadow) - virtual to real address translation.		7		DMKPRV DMKTRC
DMKVATSX	Process segment exception for a virtual machine that performs paging.		9		DMKPRG
DMKVCA	Simulate I/O for virtual channel to channel. _Data Areas Used_: CHXBLOK CHYBLOK IOBLOK SAVEAREA VCHBLOK VCUBLOK VDEVBLOK VMBLOK			DMKCVTBH DMKDIASM DMKDSPCH DMKFREE DMKFRET DMKPTRAN DMKPTRUL DMKQCNWT DMKSCNVU DMKSTKCP	
DMKVCARD	Reset device without de-coupling.		9		DMKCFP
DMKVCARS	Reset device and drop CTCA.		9		DMKVDB
DMKVCASH	Simulate virtual HIO.		8		DMKVIO
DMKVCAST	Simulate virtual SIO.		1		DMKVIO
DMKVCATS	Simulate virtual TIO.		8		DMKVIO
DMKVCH	Process real I/O corrections _Data Areas Used_: RCHBLOK RCUBLOK RDEVBLOK			DMKFREE DMKFRET DMKQCNWT DMKSCNRU DMKSCNVU	

Name	Comments	MOD	Chart	Calls To	Called By
	SAVEAREA			DMKVDBRL	
	VCHBLOK			DMKVDSDF	
	VCUBLOK				
	VDEVBLOK				
	VMBLOK				
DMKVCHDC	ATTACH and DETACH (real devices				
	and channels) command processor.		1		DMKVDB
DMKVCN	Simulate user SIOs to			DMKCFMBK	
	virtual console.			DMKDSPCH	
	Data Areas Used:			DMKFREE	
	VCHBLOK			DMKFRET	
	VCONCTL			DMKQCNRD	
	VCUBLOK			DMKQCNWT	
	VDEVBLCK				
	VMBLOK				
DMKVCNEX	Simulate all SIOs to a virtual				
	console.	7B1	1		DMKVIO
DMKVDB	Process virtual I/O connections			DMKACODV	
	Data Areas Used:			DMKCFPRD	
	BUFFER			DMKCVTBD	
	IOBLCK			DMKCVTBH	
	RDEVBLOK			DMKCVTHB	
	SAVEAREA			DMKERMSG	
	VCHBLOK			DMKFREE	
	VCUBLOK			DMKFRET	
	VDEVBLOK			DMKIOSQR	
	VMBLCK			DMKQCNWT	
				DMKSCNAU	
				DMKSCNFD	
				DMKSCNRD	
				DMKSCNRN	
				DMKSCNRU	
				DMKSCNVN	
				DMKSCNVS	
				DMKSCNVU	
				DMKTDKRL	
				DMKUDRLK	
				DMKUDRUL	
				DMKVCARS	
				DMKVCHDC	
				DMKVDBRL	
				DMKVDSAT	
				DMKVSPCO	
				DMKVSPCR	

Name	Comments	MOD	Chart	Calls To	Called By
DMKVDBAT	ATTACH (virtual devices or channels to a virtual machine) command processor.		1		DMKCFM
DMKVDBDE	DETACH (virtual devices or channels from virtual machine) command processor.		13		DMKCFM
DMKVDBRL	Release a device from a virtual machine.		14		DMKCFP DMKLNK DMKVCH
DMKVDS	Virtual device interface Data Area Used: RDEVBLOK SAVEAREA VDEVBLOK VCHBLOK VCUBLOK VDEVBLOK VMBLOK			DMKFREE DMKFRET DMKSCNVU DMKTDKGT	
DMKVDSAT	Attach a virtual device.		1		DMKLOG DMKVDB
DMKVDSDF	Define a virtual device.		2		DMKDEF DMKLOG DMKVCH
DMKVDSLK	Link to a virtual DASD device.		3		DMKLNK DMKLOG
DMKVIO	Virtual I/O manager Data Areas Used: IOBLOK VCHBLOK VCUBLOK VDEVBLOK VMBLCK			DMKCCWTR DMKDSPB DMKDSPCH DMKFREE DMKFRET DMKIOSQV DMKPTRAN DMKSCNVU DMKSTKIO DMKTRCSI DMKTRCSW DMKUNTFR DMKUNTRN DMKVCASH DMKVCAST DMKVCATS DMKVCNEX DMKVSPEX	
DMKVIODC	Dedicated Channel Interrupt		13		DMKIOS

Name	Comments	MOD	Chart	Calls To	Called By
DMKVIOEX	Simulate a SIO, TIO, HIO, or TCH.	1B3.9	1		DMKPRV
DMKVIOIN	Translate a virtual I/O interrupt.	1B4.2	9		DMKIOS DMKSPL
DMKVMI	IPL virtual machine			None	
DMKVMIPL	Simulate IPL to a virtual machine.		1		DMKCFP
DMKVSP	Virtual spooling manager __Data Areas Used__: SAVEAREA SFBLOK VCHBLOK VCUBLOK VDEVBLOK VFCBLOK VSPLCTL VMBLOK			DMKCVTBH DMKCVTDT DMKERMSG DMKFREE DMKFRET DMKPGTSG DMKPGTVG DMKPGTVR DMKPTRAN DMKPTRUL DMKQCNWT DMKRPAGT DMKRPAPT DMKSCNVD DMKSCNVU DMKSPLCV DMKSPLDL DMKSPLOV	
DMKVSPCO	Close spooled printers and punches.		11		DMKCDB DMKCFD DMKCSP DMKVDB
DMKVSPCR	Stop and clear all pending status for spooled card reader.		10		DMKCFP DMKCSP DMKDRD DMKTRA DMKVDB

Name	Comments	MOD	Chart	Calls To	Called By
DMKVSPEX	Simulate SIO to a spooled unit record device.		1		DMKVIO
DMKVSPRT	Put a CP generated line on the User's spooled printer		14		DMKCPB DMKTRC
DMKVSPVP	Simulate SIO to a spooled virtual console.		13		DMKVCN
DMKWRM	Warm Start			DMKDSPCH DMKERMSG DMKFREE DMKFRET DMKPGTVG DMKPGTVR DMKQCNWT DMKRPAGT DMKRPAPT DMKSCNRU	
DMKWRMST	Warm start.	5B1.4	1		DMKCPI

SUBROUTINE DIRECTORY

Subroutine	Module	Chart
ACNTDED	DMKCPV	6
ACTPIRA	DMKACO	3
ACTPUNCH	DMKACO	3
ADDCHEK	DMKDRD	4
ADDCHEK	DMKHVC	2
ADDQ	DMKSCH	3
ADDRTEST	DMKVMI	3
ADJTIME	DMKMID	2
ADTRANS	DMKPRV	5
ARUNLST	DMKSCH	5
ASTERISK	DMKCFM	3
AWAITLST	DMKSCH	5
BLANKBUF	DMKTRC	8
BLDBLOK	DMKVDS	4
BLDVDEV	DMKVDS	3
BLOKLETR	DMKSEP	4
BUILDCTL	DMKSPL	2
CALLERR	DMKIOS	15
CCWCHKEY	DMKCCW	30
CCWFETCH	DMKVCA	3
CCWPUTSK	DMKCCW	20
CCWRELCH	DMKCCW	21
CFPIO	DMKCFP	6
CHACT	DMKCKP	4
CHEKPTE	DMKVAT	11
CHEKRKEY	DMKDGD	8
CHKBUSY	DMKDRD	6
CHKFORM	DMKVAT	1
CHKRDEV	DMKVDB	10
CHNREXIT	DMKIOE	4
CHSCAN	DMKVIO	8
CKBKMSG	DMKRSP	5
CKEXTRA	DMKCFS	3
CKEXTRA	DMKCPV	11
CKIOB	DMKDAS	9
CKIOB	DMKTAP	10
CKRECMP	DMKRPA	2
CKTIME	DMKPTR	4
CKWAIT	DMKPTR	4
CLOCKFMT	DMKDMP	4
CLOSE	DMKCKP	5
CLRCHAN	DMKDMP	8
CMPNEXT	DMKDRD	7

Subroutine	Module	Chart
CNTRLSBW	DMKCCW	13
CNTRLSUB	DMKCCW	9
CNVTBIN	DMKCDB	11
CONFRRTN	DMKCFM	3
CONNECT	DMKDRD	5
CONVERT	DMKMSW	3
COPYIOB	DMKIOS	10
COPYSEGT	DMKVAT	3
CORTBLR	DMKMCH	6
CPIPINT	DMKCPI	1
CUSCAN	DMKVIO	8
CVTRADD	DMKVDB	11
DASDXA	DMKCCW	19
DASDXB	DMKCCW	19
DASDXC	DMKCCW	19
DASDXD	DMKCCW	19
DASDXE	DMKCCW	19
DASDXF	DMKCCW	20
DASDX0	DMKCCW	17
DASDX1	DMKCCW	13
DASDX2	DMKCCW	14
DASDX3	DMKCCW	15
DASDX4	DMKCCW	15
DASDX5	DMKCCW	16
DASDX6	DMKCCW	17
DASDX7	DMKCCW	17
DASDX9	DMKCCW	18
DEDDXA	DMKCCW	22
DEDDXB	DMKCCW	22
DEDDXC	DMKCCW	23
DEDDXD	DMKCCW	23
DEDDXE	DMKCCW	23
DEDDXF	DMKCCW	23
DEDDX1	DMKCCW	22
DEDDX2	DMKCCW	22
DEDDX3	DMKCCW	22
DEDDX4	DMKCCW	22
DEDDX5	DMKCCW	22
DEDDX6	DMKCCW	22
DEDDX7	DMKCCW	22
DEDDX9	DMKCCW	22
DELETE	DMKRSP	4
DELIRA	DMKSPL	9
DEVCARD	DMKACO	1
DEVCARD	DMKCKP	6
DEVSCAN	DMKVIO	9
DGRETURN	DMKDGD	7

Subroutine	Module	Chart
DGTRANS	DMKDGD	6
DGTRANSO	DMKDGD	6
DIAGRTN	DMKGEN	1
DIALING	DMKDIA	4
DIALIRA	DMKDIA	4
DIALXA	DMKCCW	28
DIALXB	DMKCCW	28
DIALXC	DMKCCW	28
DIALXD	DMKCCW	28
DIALXE	DMKCCW	28
DIALXF	DMKCCW	28
DIALX1	DMKCCW	26
DIALX2	DMKCCW	27
DIALX3	DMKCCW	27
DIALX4	DMKCCW	27
DIALX5	DMKCCW	27
DIALX6	DMKCCW	27
DIALX7	DMKCCW	28
DIALX9	DMKCCW	28
DISASUB	DMKCPV	3
DISCOMM	DMKCDB	2
DISCRETN	DMKUSO	5
DISDMPID	DMKCDB	9
DISHEAD	DMKCDB	9
DISINIT	DMKCDB	7
DISNEXTA	DMKCDB	3
DISWRITE	DMKCDB	10
DISWRTR	DMKCDB	10
DMPMCHCK	DMKDMP	5
DMPPRGCK	DMKDMP	5
DRAINMSG	DMKRSP	7
DROPQ	DMKSCH	4
DRDLCSE	DMKDRD	6
DROPING	DMKDIA	5
DROPLIST	DMKSCH	5
DSKIOINT	DMKDMP	3
DSKWRTC1	DMKDMP	3
DSKWTRC2	DMKDMP	3
DTBIN	DMKMID	1
DTDEC	DMKMID	1
DVICECLR	DMKVSP	11
ECHORETN	DMKMSG	3
ECHOWRIT	DMKMSG	3
ECHOWTRT	DMKMSG	3
EMPTYCYL	DMKPGT	7
ENABTERM	DMKCPV	3
EXCLAIM	DMKCNS	13
FILECLR	DMKVSP	10
FINDEVIC	DMKPGT	5
FINDREC	DMKPGT	6
FIOER	DMKIOE	5
FIOER	DMKIOF	6
FIRSTREC	DMKCKP	5
FMTREAD	DMKIOG	4
FMTWRITE	DMKIOG	4
FNDVADDR	DMKCFP	7
FNDVADDR	DMKCPB	3,4
FRETCORE	DMKCQG	2
FRETIOB	DMKCNS	4
FRETIOB	DMKCSO	10
FRETIOER	DMKCNS	7
FRETPTR	DMKDAS	9
FRETPTR	DMKTAP	10
FRETVMB	DMKDIA	5
FRET05	DMKFRE	3
FREUSER	DMKDEF	6
FREUSER	DMKVDB	16
FSINGIOE	DMKIOF	5
FUMAT	DMKMCH	6
GETADDR	DMKPRV	5
GETADDR	DMKRSP	7
GETBUF	DMKQCN	3
GETBUFF	DMKUDR	3
GETCCW	DMKVCN	3
GETCCW	DMKVSP	15
GETCHAIN	DMKCSP	10
GETCHAIN	DMKCSU	16
GETCLASS	DMKCSP	7
GETCLASS	DMKCSU	12
GETCOPY	DMKCSP	8
GETCOPY	DMKCSU	13
GETDEVIC	DMKCSO	11
GETDEVIC	DMKCSP	8
GETDISK	DMKWRM	4
GETENTRY	DMKRPA	2
GETFILE	DMKCSP	8
GETFILE	DMKCSU	13
GETID	DMKCSP	10
GETID	DMKCSU	15
GETNAME	DMKCSP	9
GETNAME	DMKCSU	14
GETPAGT	DMKVAT	10
GETRCAW	DMKSPL	6
GETRCAW	DMKVIO	13

Subroutine	Module	Chart
GETRDEV	DMKVDB	17
GETRTYP	DMKVDB	17
GETSHAD	DMKVAT	3
GETUSER	DMKCSP	6
GETUSER	DMKCSU	10
GETUSER	DMKSPL	3
GETVRADD	DMKPTR	5
GETVSPL	DMKDRD	5
GETYPE	DMKCSP	6
GETYPE	DMKCSU	12
GTCLASSB	DMKCSP	7
GTCLASSB	DMKCSU	12
GU02	DMKCSU	10
IDAWSUB	DMKCCW	13
INITCON	DMKQCN	2
INITSCAN	DMKCSU	6
INSTADR	DMKPRV	4
IO	DMKVMI	1
IOEMSGWR	DMKIOG	4
IOINT	DMKSAV	1
IORETN	DMKRPA	2
IORETURN	DMKUDR	6
IOSDQCH	DMKIOS	13
IOSDQCU	DMKIOS	13
IOSDQDV	DMKIOS	12
IOSENSE	DMKIOS	14
IOSFINDP	DMKIOS	9
IOSGTIOB	DMKIOS	10
IOSIGNOR	DMKIOS	11
IOSQCH	DMKIOS	12
IOSQCU	DMKIOS	11
IOSQDEV	DMKIOS	11
IOSRECER	DMKIOS	14
IOSTRTCH	DMKIOS	7
IOSTRTCU	DMKIOS	7
IOSTRTDV	DMKIOS	8
KEYTRAN	DMKHVC	1
LASTLINE	DMKDMP	4
LBIONT	DMKSAV	3
LINESIZE	DMKCFT	1
LINKDEF	DMKLNK	11
LINKSUB	DMKLNK	9
LOKUSER	DMKDEF	6
LOKUSER	DMKVDB	17

Subroutine	Module	Chart
MCHGOZO	DMKMCH	9
MCHHSKP	DMKMCH	8
MOVEDATA	DMKVSP	16
MSGALL	DMKMID	2
MSGERR	DMKSPL	7
MSGFMT	DMKMSG	2
MSGRET	DMKMCH	5
MSGSND	DMKMSG	2
MSGSUBR	DMKUSO	5
MSG900E	DMKCKP	7
MSG901E	DMKCKP	7
NEXTBUFF	DMKRSP	5
NXTFCR	DMKCNS	17
NXTPTR	DMKDAS	7
OPEN	DMKVSP	14
OPENCONT	DMKVSP	14
OTHRXA	DMKCCW	29
OTHRXB	DMKCCW	29
OTHRXC	DMKCCW	29
OTHRXD	DMKCCW	29
OTHRXE	DMKCCW	29
OTHRXF	DMKCCW	29
OTHRX1	DMKCCW	28
OTHRX2	DMKCCW	29
OTHRX3	DMKCCW	29
OTHRX4	DMKCCW	29
OTHRX5	DMKCCW	29
OTHRX6	DMKCCW	29
OTHRX7	DMKCCW	29
OTHRX9	DMKCCW	29
OVERHEAD	DMKPAG	3
PAGFREE	DMKPTR	10
PAGIN	DMKPTR	5
PARTLINE	DMKDMP	4
PASSRETN	DMKEPS	1
PCITEST	DMKVSP	16
PERCHEK	DMKPRV	4
PGTMSG	DMKPGT	4
PGTMSG2	DMKMSG	4
PRGEVEN	DMKPRG	4
PRIO	DMKDMP	5
PROTEST	DMKVSP	16
PROTTEST	DMKVCN	6
PRTDATA	DMKVSP	12
PRTDONE	DMKVSP	8

Subroutine	Module	Chart
PRTEOF	DMKVSP	7
PSETKEY	DMKPRV	3
PTRFDISP	DMKPTR	9
PUTBACK	DMKTRC	9
PUTLINE	DMKDMP	4
PUTMSG1	DMKRSP	5
PUTREC	DMKCKP	5
PUTSEEK	DMKCCW	20
QRYFCNT	DMKCQG	1
QRYUSRN	DMKCQG	2
QRYVFMT	DMLCQG	2
RDLABEL	DMKVDB	11
RDLBIRA	DMKVDB	12
RDRDATA	DMKVSP	9
RDRID	DMKSPL	2
READADDR	DMKSSP	3
READBUF	DMKDAS	9
READLOG	DMKCFS	2
REALREAD	DMKIOF	5
REALRETN	DMKCQP	2
REALWRT	DMKIOF	5
RECGETNN	DMKCFS	3
RECHAIN	DMKRSP	6
REGSPEC	DMKPRV	3
RESETIMR	DMKSCH	8
RESTINST	DMKCFD	4
RESYSTEM	DMKCFP	4
RESYSTEM	DMKCPB	1
RET8	DMKWRM	5
RET8R1	DMKWRM	5
REWRITE	DMKPTR	11
RSPMSG	DMKRSP	5
RSTMPOFF	DMKSCH	6
RSTMPON	DMKSCH	5
RSTRIRA	DMKSEP	3
RTNBRKND	DMKCNS	20
RTNEOA	DMKCNS	8
RTNEXCLM	DMKCNS	14
RTNIDENT	DMKCNS	10
RTNPREP	DMKCNS	12
RTNSDPRP	DMKCNS	11
RUNTIME	DMKDSP	8
SAVRETN	DMKCFG	3
SAVRETN	DMKCPV	11
SCAN	DMKSSP	3

Subroutine	Module	Chart
SCAN	DMKUDR	1
SCANDEV	DMKCSO	5
SCANRSP	DMKCQP	1
SCANSHQ	DMKCQP	1
SCPZCAW	DMKSAV	3
SEARCHB	DMKDRD	5
SELECT	DMKPTR	10
SENSE	DMKVSP	8
SENSMOVE	DMKVSP	13
SEPIRA	DMKSEP	3
SETCCW	DMKDMP	5
SETHHR	DMKSAV	1
SETIMER	DMKSCH	7
SETOPTS	DMKVSP	7
SETUPEX	DMKVAT	8
SFBCHAIN	DMKSPL	2
SFBSCAN	DMKCSU	6
SHADOWS	DMKVAT	2
SHADSET	DMKVAT	8
SNSRTN	DMKDAS	8
SRCHPTE	DMKVAT	10
STARTMSG	DMKCSO	5
STOLOCA	DMKCDS	2
STOSCAN	DMKCDS	1
STSTERR	DMKMCH	6
SVCFR	DMKPSA	2
SVCGET	DMKPSA	5
SVCRLSE	DMKPSA	5
SWPCALL	DMKVDB	16
SWPUSER	DMKVDB	17
SYSTUNLC	DMKCFG	3
SYSUNLCK	DMKCFP	12
TAPERADD	DMKRSP	5
TAPEXA	DMKCCW	24
TAPEXB	DMKCCW	24
TAPEXC	DMKCCW	24
TAPEXD	DMKCCW	25
TAPEXE	DMKCCW	25
TAPEXF	DMKCCW	25
TAPEX1	DMKCCW	23
TAPEX2	DMKCCW	23
TAPEX3	DMKCCW	23
TAPEX4	DMKCCW	23
TAPEX5	DMKCCW	23
TAPEX6	DMKCCW	23
TAPEX7	DMKCCW	24
TAPEX9	DMKCCW	24

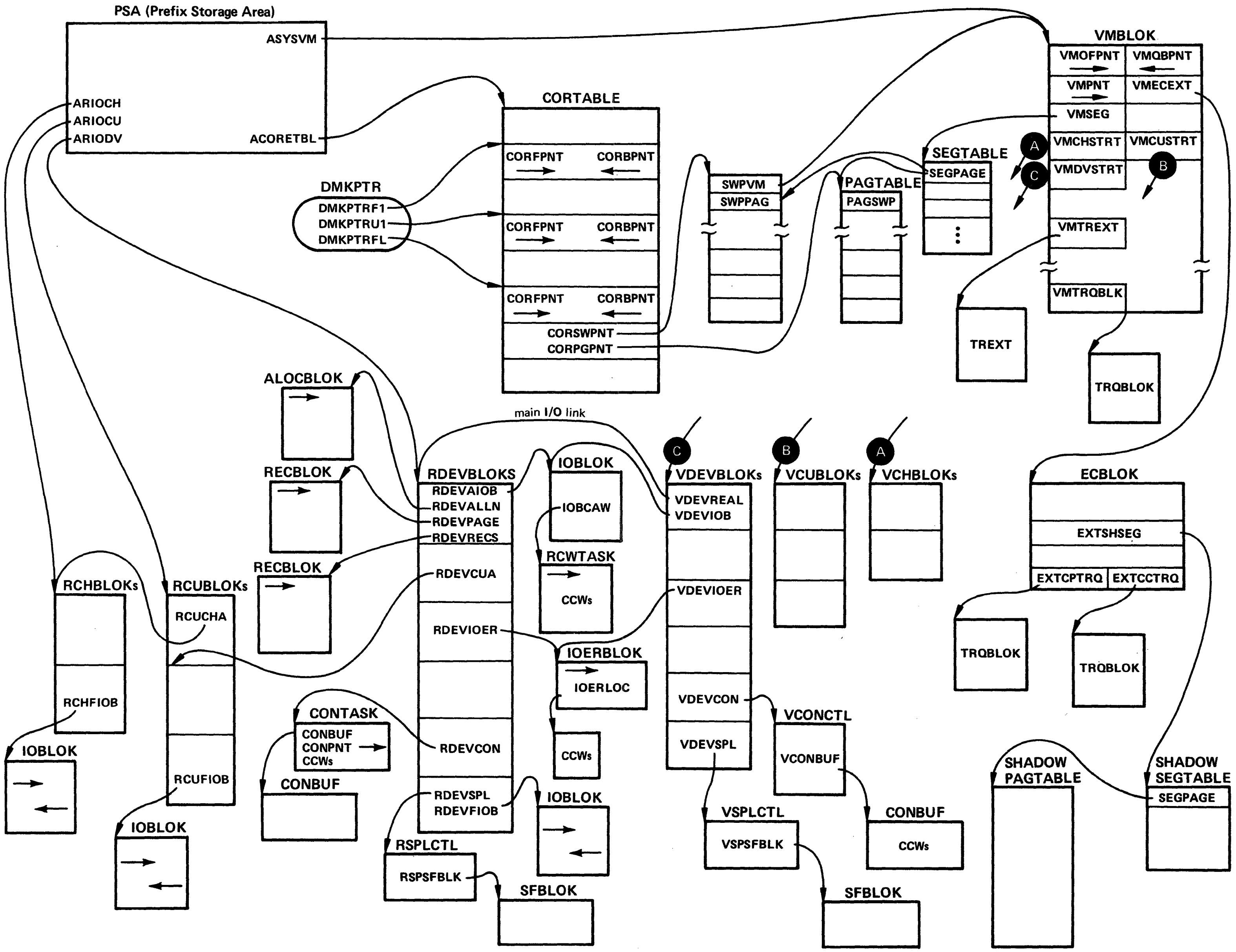
Subroutine	Module	Chart
TDKIRA	DMKTDK	2
TERM	DMKMCH	6
TERMIRA	DMKRSP	6
TERMXA	DMKCCW	26
TERMXB	DMKCCW	26
TERMXC	DMKCCW	26
TERMXD	DMKCCW	26
TERMXE	DMKCCW	26
TERMXF	DMKCCW	26
TERMX1	DMKCCW	25
TERMX2	DMKCCW	25
TERMX3	DMKCCW	25
TERMX4	DMKCCW	25
TERMX5	DMKCCW	25
TERMX6	DMKCCW	25
TERMX7	DMKCCW	26
TERMX9	DMKCCW	26
TICSUBI	DMKCCW	8
TICSUBX	DMKCCW	8
TICSUB1	DMKCCW	8
TRAINIT	DMKTRC	8
TRANBRNG	DMKCCW	29
TRANBRNG	DMKTRC	12
TRANLOCK	DMKCCW	30
TRANRETN	DMKPTR	4
TRANS	DMKCNS	19
TROUSUB	DMKTRC	10
TROUSUB7	DMKTRC	11

Subroutine	Module	Chart
TSTONOFF	DMKCFS	3
TSTONOFF	DMKCFT	2
TSTSEP	DMKSEP	3
TYPLINE	DMKDMP	6
UNLOKSUB	DMKLNK	11
UNRELSUB	DMKUNT	3
UNTRSREL	DMKUNT	5
USERCARD	DMKACO	1
USERCARD	DMKCKP	6
USERDEV	DMKVDB	17
USOSUB	DMKUSO	5
VCNMVDAT	DMKVCN	4
VCNRDRET	DMKVCN	8
VCNRELSE	DMKVCN	6
VCNSCALC	DMKVCN	4
VDBSCAN	DMKVDB	1
VIOINT1	DMKVIO	12
VOL1RTN	DMKDAS	9
WAITIME	DMKDSP	9
WAITPAGE	DMKPAG	2
WRTOUT	DMKCQP	1
WRTVIRT	DMKCQG	9
ZAPPAGE	DMKVAT	4
ZAPSEGS	DMKVAT	4
ZAPVOLD	DMKVDB	11

DATA AREAS -- CONTROL BLOCKS

To determine the modules that reference or alter a data area or a field in a data area, refer to the alphamerical Label Cross-reference list that is contained in the microfiche for VM/370.

Diag. 9A0. CP Control Block Relationships



ACCTBLOK - USER ACCOUNTING BLOCK

```
    ┌──────────────────────────────────────────┐
  0 |                ACCTUSER                  |
    |──────────────────────────────────────────|
  8 |                ACCTACNO                  |
    |──────────────────────────────────────────|
 10 |                ACCTDIST                  |
    └──────────────────────────────────────────┘
```

Displacement		Field			
Hex	Dec	Name			Field Description, Contents, Meaning
0	0	ACCTUSER	DS	CL8	Virtual machine identification
8	8	ACCTACNO	DS	CL8	Virtual machine accouting number
10	16	ACCTDIST	DS	CL8	Virtual machine distribution number
		ACCTLENG	EQU	(*-ACCTBLOK)/8	Size of ACCTBLOK in doublewords (X'03')

ACNTBLOK - ACCOUNTING CARD BUFFER

```
    ┌───────────────────────────────────────────┐
  0 |                  ACNTCCW                  |
    |───────────────────────────────────────────|
  8 |     ACNTNEXT         |                    |
    |──────────────────────┘                    |
    |                                           |
    |     ACNTDATA  (See Format for User Cards) |
    |                                           |
    |                      ┌────────────────────|
 58 |                      |     ACNTBACK       |
    └───────────────────────────────────────────┘
```

Displacement		Field			
Hex	Dec	Name			Field Description, Contents, Meaning
0	0	ACNTCCW	DS	D	Punch CCW for accounting card
8	8	ACNTNEXT	DS	F	Address of next ACNTBLOK in chain
C	12	ACNTDATA	DS	CL80	Accounting information
5C	92	ACNTBACK	DS	F	Address of previous ACNTBLOK in chain
		ACNTSIZE	EQU	(*-ACNTBLOK)/8	Size of ACNTBLOK in doublewords (X'0C')

Format for User Cards

```
     ┌────────────────────────────────────────────┐
  C  |                 ACNTUSER                    |
     |────────────────────────────────────────────|
 14  |                 ACNTNUM                     |
     |────────────────────────────────────────────|
 1C  |                 ACNTSTOP                    |
     |                    ┌───────────────────────|
     |                    |       ACNTCONT         |
     |────────────────────────────────────────────|
 2C  |     ACNTTIME       |       ACNTVTIM         |
     |────────────────────────────────────────────|
 34  |     ACNTPGRD       |       ACNTPGWT         |
     |────────────────────────────────────────────|
 3C  |     ACNTIOCT       |       ACNTPNCH         |
     |────────────────────────────────────────────|
 44  |     ACNTLINS       |       ACNTCRDS         |
     |────────────────────────────────────────────|
 4C  |                 ACNTRSV1                    |
     |────────────────────────────────────────────|
 54  |     ACNTRSV2                 | ACNTCODE     |
     └────────────────────────────────────────────┘
```

Displacement Hex	Dec	Field Name			Field Description, Contents, Meaning
			ORG	ACNTDATA	
C	12	ACNTUSER	DS	CL8	Virtual machine identification
14	20	ACNTNUM	DS	CL8	Virtual machine accounting number
1C	28	ACNTSTOP	DS	CL12	Date and time of accounting MMDDYYHHSS
28	40	ACNTCONT	DS	1F	Number of seconds connected
2C	44	ACNTTIME	DS	1F	Milliseconds of CPU time used
30	48	ACNTVTIM	DS	1F	Milliseconds of virtual CPU time used
34	52	ACNTPGRD	DS	1F	Total page reads
38	56	ACNTPGWT	DS	1F	Total page writes
3C	60	ACNTIOCT	DS	1F	Virtual SIO count for non-spooled I/O
40	64	ACNTPNCH	DS	1F	Virtual card count - spooled punch
44	68	ACNTLINS	DS	1F	Virtual line count - spooled printer
48	72	ACNTCRDS	DS	1F	Virtual card count - spooled reader
4C	76	ACNTRSV1	DS	FL8	Reserved for IBM use
54	84	ACNTRSV2	DS	HL6	Reserved for IBM use
5A	90	ACNTCODE	DS	1H	Accounting card identification code
		Card code for ACNTCODE			
			DC	C'x1'	User virtual machine accounting card
			DC	C'x2'	User dedicated device accounting card
			DC	C'x3'	User temporary disk space accounting card

where x = C if the card was punched for the userid in the User Accounting Block
= 0 if the card was punched for the user requesting the card

ORG ACNTTIME

2C	44	ACNTDEVC	DS	XL4	Device code (CTFM) See DEVTYPE copy file
30	48	ACNTNCYL	DS	1H	Number of cylinders of T-disk space

ALOCBLOK - DASD CYLINDER ALLOCATION BLOCK

```
  ┌─────────────────────────────────────┐
0 |      ALOCPNT      |ALOCUSED |ALOCMAX |
  |─────────────────────────────────────|
8 |               ALOCMAP               |
  └─────────────────────────────────────┘
```

Displacement		Field			
Hex	Dec	Name			Field Description, Contents, Meaning
0	0	ALOCPNT	DS	1F	Pointer to next ALOCBLOK on chain
4	4	ALOCUSED	DS	1H	Number of cylinders currently in use
6	6	ALOCMAX	DS	1H	Maximum number of cylinders available
8	8	ALOCMAP	DS	0F	Cylinder allocation bit map

Bits defined in ALOCMAP:
0 = Cylinder is available
1 = Cylinder has been assigned

Note: The size of the ALOCMAP is variable and depends on the number of cylinders on the device. Generally, the size of the ALOCBLOK is determined by the following formula:

ALOCSIZE(doublewords) = ((((ALOCMAX+7)/8)+7)/8)+1

where:

ALOCMAX	for 2314	=	203
	for 3330	=	404
	for 2305-1	=	48
	for 2305-2	=	96

Any bits in the map that represent cylinders not present on the device are set to one.

For TDISK allocation blocks

ORG ALOCUSED

4	4	ALOCCYL1	DS	1H	First cylinder of TDISK area
6	6	ALOCCYL2	DS	1H	Last cylinder of TDISK area

Bytes defined in ALOCMAP
X'00' = Cylinder is available
X'AA' = Cylinder has been allocated

__Note__: The size of the TDISK ALOCMAP is variable and depends on the number of cylinders in the range ALOCCYL1 to ALOCCYL2. Generally, the size of a given block is determined by the following formula:

ALOCSIZE(doublewords) = ((ALOCCYL2—ALOCCYL1+8)/8)+1

Bytes for cylinders that are not available are marked allocated.

__BUFFER__ — __CONSOLE FUNCTION INPUT BUFFER__

0	BUFFER	
88	BUFNXT	BUFCNT

Displacement		Field			
Hex	Dec	Name			Field Description, Contents, Meaning
0	0	BUFIN	DS	CL136	Input line
88	136	BUFNXT	DS	1F	Pointer to next byte in BUFFER
8C	140	BUFCNT	DS	1F	Count of characters in input line
88	136	BUFINLTH	EQU	L'BUFIN	Input BUFFER size in bytes
		BUFSIZE	EQU	(*—BUFFER)/8	BUFFER size in doublewords (X'12')

CCHREC - CHANNEL CHECK HANDLER RECORD

```
    ┌────────────────────────────────────────────┐
 0  |C*1 |C*2 |C*3 |C*4 | CCSW2REV|C*5 |C*6 |
    |────────────────────────────────────────────|
 8  |               CCDATE                       |
    |────────────────────────────────────────────|
10  |               CCCPUID                      |
    |────────────────────────────────────────────|
18  |               CCPROGID                     |
    |────────────────────────────────────────────|
20  |                                            |
    =               FAILADD                      =
    |                                            |
    |────────────────────────────────────────────|
30  |               FAILCCW                      |
    |────────────────────────────────────────────|
38  |               FAILCSW                      |
    |────────────────────────────────────────────|
40  |    FAILECSW          |     CCDEVTYP        |
    |────────────────────────────────────────────|
48  | CCHANID | CCHCUA     |        CCHMP        |
    |────────────────────────────────────────────|
50  |                                            |
    <                                            >
    <               CCHLOG                       >
    <                                            >
    |                                            |
    └────────────────────────────────────────────┘
```

Displacement Hex	Dec	Field Name			Field Description, Contents, Meaning
0	0	CCRECTYP	DS	1X	C*1 Record type
1	1	CCOPSYS	DS	1X	C*2 Operating system
2	2	CCSW1	DS	1X	C*3 Switch one
3	3	CCSW2	DS	1X	C*4 Switch two
4	4	CCSW2REV	DS	2X	Unused
6	6	CCRECNT	DS	1X	C*5 Record count
7	7	CCRECNT1	DS	1X	C*6 Unused
8	8	CCDATE	DS	1D	Date and time
10	16	CCCPUID	DS	1D	CPUID
18	24	CCPROGID	DS	1D	USERID
20	32	FAILADD	DS	8H	Active I/O units
30	48	FAILCCW	DS	1D	Failing CCW
38	56	FAILCSW	DS	1D	Failing CSW
40	64	FAILECSW	DS	0F	Failing ECSW
40	64	IGPRGFLG	DS	CL1	Program flag bits

```
                 Bits defined in IGPRGFLG
                 CCHSIOB  EQU   X'80'          SIO bit
                 CCHINTB  EQU   X'40'          Interrupt bit
                 CCHSNSB  EQU   X'04'          Sense data stored bit
                 CCHCNTB  EQU   X'02'          Count valid bit
                 CCHNRYB  EQU   X'01'          No retry bit

41   65          IGBLAME  DS    CL1            Probable source of error
                 Bits defined in IGBLAME
                 CCHCPU   EQU   X'80'          CPU is source of error
                 CCHCHNL  EQU   X'40'          Channel is source of error
                 CCHSCUB  EQU   X'20'          Storage control unit is source of error
                 CCHSTG   EQU   X'10'          Storage is source of error
                 CCHINTFC EQU   X'08'          I/O interface is source of error

42   66          IGVALIDB DS    CL1            Validity indicator bits
                 Bits defined in IGVALIDB
                 CCHRCV   EQU   X'10'          Retry code valid
                 CCHUSV   EQU   X'08'          Selective reset
                 CCHCMDV  EQU   X'04'          Command address valid
                 CCHCAV   EQU   X'02'          Channel address valid
                 CCHDAV   EQU   X'01'          Device address valid

43   67          IGTERMSQ DS    CL1            Termination/sequence code bits
                 Bits defined in IGTERMSQ
                 COMPSYS  EQU   X'C0'          System reset
                 COMPSEL  EQU   X'80'          Selective reset
                 COMPFES  EQU   X'40'          Forced ending sequence
                 COMPID   EQU   X'00'          Interface disconnect
                 CCHDI    EQU   X'08'          Disconnect in sequence code bits

                 Sequence code bits
                 RTCODE0  EQU   X'00'          Retry
                 RTCODE1  EQU   X'01'                Code
                 RTCODE2  EQU   X'02'                    Values
                 RTCODE3  EQU   X'03'                            For
                 RTCODE4  EQU   X'04'                               The
                 RTCODE5  EQU   X'05'                                      Constructed
                 RTCODE6  EQU   X'06'                                                 ECSW
                 RTCODE7  EQU   X'07'

44   68          CCDEVTYP DS    1F             CP device type
48   72          CCHANID  DS    XL2            Channel ID
4A   74          CCHCUA   DS    1H             Actual failing address
4C   76          CCHMP    DS    1F             MP information
50   80          CCHLOG80 DS    0CL112         2880 channel - 112 bytes
50   80          CCHLOG70 DS    0CL24          2870 channel - 24 bytes
50   80          CCHLOG60 DS    0CL24          2860 channel - 24 bytes

                 CCHSIZE1 EQU   (*-CCHREC)/8   Size in doublewords (X'0A')
```

```
50   80          CUAADDR  DS    CL4            Unit address stored by integrated channel

                 CCHSIZE  EQU   (*-CCHREC)/8   Size in doublewords

54   84          CCHLOG45 DS    0CL96          145 integrated channel - 96 bytes
54   84          CCHLOG35 DS    0CL24          135 integrated channel - 24 bytes
```

CHXBLOK AND CHYBLOK - VIRTUAL CHANNEL-TO-CHANNEL ADAPTER CONTROL BLOCKS

```
    ┌───────────────────────────────────────┐
  0 |      CHXOTHR       |      CHYOTHR      |
    |───────────────────────────────────────|
  8 | X*1| X*2| X*3| X*4| Y*1| Y*2| Y*3| Y*4|
    |───────────────────────────────────────|
 10 |      CHXNCCW       |      CHYNCCW      |
    |───────────────────────────────────────|
 18 |      CHXRCNT       |      CHYRCNT      |
    |───────────────────────────────────────|
 20 | CHXSTAT | CHXYADD | CHYSTAT | CHXYADD |
    |───────────────────────────────────────|
 28 |      CHXIDAW       |      CHYIDAW      |
    |───────────────────────────────────────|
 30 |      CHXCNCT       |      CHYCNCT      |
    |───────────────────────────────────────|
 38 |      CHXWRK1       |      CHYWRK1      |
    └───────────────────────────────────────┘
```

```
Displacement     Field
Hex  Dec         Name                     Field Description, Contents, Meaning
────────────     ────                     ─────────────────────────────────────
                 CHXBLOK                  X-side channel adapter block
  0    0         CHXOTHR  DS    2F        VMBLOK address of Y-side adapter user
  8    8         CHXFLAG  DS    1X        X*1  Internal processing flags
                 Bits defined in CHXFLAG and CHYFLAG:
                 CHBMNOP  EQU   X'80'     Modified NOP issued (also in CMDT)
                 CHBM370  EQU   X'40'     CTCA operating in System/370 mode
                 CHBATTN  EQU   X'20'     Attention pending from Y-side
                 CHBREST  EQU   X'10'     CTCA has been reset X-side and Y-side
                 CHBEOFL  EQU   X'08'     Force EOF to next READ

  9    9         CHXCMDB  DS    1X        X*2  Active CCW command byte buffer
  A   10         CHXCMDT  DS    1X        X*3  Active CCW command type (RD, WR, etc.)
                 Bits defined in CHXCMDT and CHYCMDT:
                 CHBCTNL  EQU   X'40'     Control, other than NOP
                 CHBRDBK  EQU   X'20'     Read backward
                 CHBWEOF  EQU   X'10'     Write EOF
                 CHBSCMD  EQU   X'08'     Sense command byte
                 CHBSADS  EQU   X'04'     Sense adapter status
                 CHBREAD  EQU   X'02'     Read
                 CHBWRIT  EQU   X'01'     Write

  B   11         CHXPKEY  DS    1X        X*4  Virtual CAW protection key
  C   12                  DS    4X
 10   16         CHXNCCW  DS    2F        Next CCW fetch address (real)
 18   24         CHXRCNT  DS    2F        Remaining CCW data count
 20   32            CHXSTAT  DS    1H            Device status accumulation field
 22   34            CHXYADD  DS    1H            Virtual address of Y-side adapter
 24   36                     DS    2H
 28   40            CHXIDAW  DS    2F            Active indirect-data-list word
 30   48            CHXCNCT  DS    2F            CPEXBLOK for channel reconnect
 38   56            CHXWRK1  DS    2F            Work area word

|                   CHBSIZE  EQU   (*-CHXBLOK)/8 Total block size in doublewords (X'08')

                    CHYBLOK  DSECT ,             Y-side channel adapter block
  0    0            CHYOTHR  DS    2F            VMBLOK address of X-side adapter user
  8    8            CHYFLAG  DS    1X            Y*1  Internal processing flags
                    Bits defined in CHXFLAG and CHYFLAG:
                    CHBMNOP  EQU   X'80'         Modified NOP issued (also in CMDT)
                    CHBM370  EQU   X'40'         CTCA operating in System/370 mode
                    CHBATTN  EQU   X'20'         Attention pending from Y-side
                    CHBREST  EQU   X'10'         CTCA has been reset X-side and Y-side
                    CHBEOFL  EQU   X'08'         Force EOF to next READ

  9    9            CHYCMDB  DS    1X            Y*2  Active CCW command byte buffer
  A   10            CHYCMDT  DS    1X            Y*3  Active CCW command byte
                    Bits defined in CHXCMDT and CHYCMDT:
                    CHBCNTL  EQU   X'40'         Control, other than NOP
                    CHBRDBK  EQU   X'20'         Read backward
                    CHBWEOF  EQU   X'10'         Write EOF
                    CHBSCMD  EQU   X'08'         Sense command byte
                    CHBSADS  EQU   X'04'         Sense adapter status
                    CHBREAD  EQU   X'02'         Read
                    CHBWRIT  EQU   X'01'         Write

  B   11            CHYPKEY  DS    1X            Y*4  Virtual CAW protection key
  C   12                     DS    4X
 10   16            CHYNCCW  DS    2F            Next CCW fetch address
 18   24            CHYRCNT  DS    2F            Remaining CCW data count
 20   32            CHYSTAT  DS    1H            Device status accumulation field
 22   34            CHYXADD  DS    1H            Virtual address of X-side adapter
 24   36                     DS    2H
 28   40            CHYIDAW  DS    2F            Active indirect-data-list word
 30   48            CHYCNCT  DS    2F            CPEXBLOK for channel reconnect
 38   56            CHYWRK1  DS    2F            Work area word
```

Note: As indicated in the illustrated block, the CHXBLOK and CHYBLOK are interleaved with a 4-byte displacement. The X-side VDEVBLOK points to the +0 slot, the Y-side VDEVBLOK points to the +4 slot; however, once the virtual connection is made, either side can be the X-side or Y-side since this interleaved arrangement makes the control block references completely symmetrical. The dual DSECT definition allows the active adapter (defined to be the X-side, arbitrarily) to reference both adapter sides concurrently without knowing which is at +0 or +4.

CONTASK - CONSOLE I/O

```
     ┌─────────────────────────────────────────┐
   0 |    CONPNT          |    CONRETN         |
     |─────────────────────────────────────────|
   8 | CONFLGS |C*1 |C*2 |    CONBUF          |
     |─────────────────────────────────────────|
  10 |                                         |
     |                 CONCCW                  |
     |                                         |
     └─────────────────────────────────────────┘
```

Displacement Hex	Dec	Field Name			Field Description, Contents, Meaning
0	0	CONPNT	DS	1F	Pointer to next CONTASK
4	4	CONRETN	DS	1F	Pointer to CPEXBLOK for return
8	8	CONFLGS	DS	1H	CONTASK flags
A	10	CONTSKSZ	DS	1X	C*1 - CONTASK size in doublewords
B	11	CONBUFSZ	DS	1X	C*2 - BUFFER size in doublewords
C	12	CONBUF	DS	1F	Address of data BUFFER
10	16	CONCCW	DS	1D	One or more CCWs for console I/O
		CONTSIZE	EQU	(CONCCW-CONTASK)/8	CONTASK size in doublewords (X'03')
			ORG	CONFLGS	
8	8	CONSTAT	DS	4B	CONTASK Status
		Bits Defined in CONSTAT			
		CONOUTPT	EQU	X'80'	Output CONTASK
		CONBUFVD	EQU	X'40'	CONBUF contains a valid Free Storage Buffer
			ORG	CONFLGS	
8	8	CONPARM	DS	1H	QUECONS parameter flags
			ORG	CONCCW	
10	16	CONADDR	DS	1F	CCW data address
14	20	CONFLAG	DS	1X	CCW flag bits
15	21	CONRSV3	DS	1X	Reserved for IBM use
16	22	CONCNT	DS	1H	CCW byte count
			ORG	CONADDR	
10	16	CONCOMND	DS	1X	CCW command code

CORTABLE - STCRAGE ALLOCATION TAELE

```
   ┌───────────────────────────────────────┐
 0 |     CORFPNT        |      CORBPNT     |
   |───────────────────────────────────────|
 8 |C*1 |  CORSWPNT     |     CORPGPNT     |
   |───────────────────────────────────────|
   |                                       |
   |                                       |
   |                                       |
   |───────────────────────────────────────|
   |                    |                  |
   |───────────────────────────────────────|
   |                    |                  |
   └───────────────────────────────────────┘
```

```
Displacement      Field
Hex  Dec          Name                          Field Description, Contents, Meaning
------------      -----                         ---------------------------------------

  0    0          CORFPNT  DS     1F            Pointer to next CORETABLE entry in queue
  4    4          CORBPNT  DS     1F            Pointer to previous CORETABLE entry in queue
  8    8          CORSWPNT DS     1F            Pointer to SWAPTABLE for page
  C   12          CORPGPNT DS     1F            Pointer to PAGTABLE for page

                           ORG    CORSWPNT
  8    8          CORFLAG  DS     1X            C*1 - CORTABLE entry status flags
                  Bits Defined in CORFLAG
                  CORIOLCK EQU    X'80'         Page locked for I/O, CORLCNT greater than 0
                  CORCFLCK EQU    X'40'         Page locked by console function
                  CORFLUSH EQU    X'20'         Page is in FLUSH list
                  CORFREE  EQU    X'10'         Page is in FREE list
                  CORSHARE EQU    X'08'         Page is shared
                  CORRSV   EQU    X'04'         Page is reserved
                  CORCP    EQU    X'02'         Page belongs to CP
                  CORDISA  EQU    X'01'         Page disabled, not available

                  Entry definition if page is locked
                           ORG    CORBPNT
  4    4          CORLCNT  DS     1F            Page lock count for CORIOLCK

                  Entry definition if page is in transit
                           ORG    CORFLAG
  8    8          CORCODE  DS     1X            C*1 - DASD op-code for PAGEIO
```

CPEXBLOK - CP EXECUTE BLCCK

```
      ┌─────────────────────────────────────┐
    0 │   CPEXFPNT        │    CPEXBPNT     │
      │─────────────────────────────────────│
    8 │   CPEXMISC        │    CPEXADD      │
      │─────────────────────────────────────│
   10 │                                     │
      │            CPEXREGS                 │
      │                                     │
      └─────────────────────────────────────┘
```

Displacement		Field			
Hex	Dec	Name			Field Description, Contents, Meaning
0	0	CPEXFPNT	DS	1F	Pointer to next CPEXBLOK
4	4	CPEXBPNT	DS	1F	Pointer to previous CPEXBLOK
8	8	CPEXMISC	DS	1F	Use varies with stacker
C	12	CPEXADD	DS	1F	Execute address
10	16	CPEXREGS	DS	16F	Execute registers
		CPEXSIZE	EQU	(*-CPEXBLOK)/8	Size in doublewords (X'0A')
		For CPEXREGS Area			
			ORG	CPEXREGS	
10	16	CPEXR0	DS	1F	
14	20	CPEXR1	DS	1F	
18	24	CPEXR2	DS	1F	
1C	28	CPEXR3	DS	1F	
20	32	CPEXR4	DS	1F	
24	36	CPEXR5	DS	1F	
28	40	CPEXR6	DS	1F	
2C	44	CPEXR7	DS	1F	
30	48	CPEXR8	DS	1F	
34	52	CPEXR9	DS	1F	
38	56	CPEXR10	DS	1F	
3C	60	CPEXR11	DS	1F	
40	64	CPEXR12	DS	1F	
44	68	CPEXR13	DS	1F	
48	72	CPEXR14	DS	1F	
4C	76	CPEXR15	DS	1F	

DMPINREC - DUMP FILE INFORMATION RECORD

```
     ┌─────────────────────────────────────────┐
   0 |                DMPGPRS                  |
     |─────────────────────────────────────────|
  40 |                DMPCRS                   |
     |─────────────────────────────────────────|
  80 |                DMPFPRS                  |
     |─────────────────────────────────────────|
  A0 |                DMPTODCK                 |
     |─────────────────────────────────────────|
  A8 |                DMPCPUTM                 |
     |─────────────────────────────────────────|
  B0 |                DMPCKCOM                 |
     |─────────────────────────────────────────|
  B8 |S*1 |S*2 | DMPRSV2 |      DMPSYSRV       |
     |─────────────────────────────────────────|
  C0 |                DMPLCORE                 |
     |─────────────────────────────────────────|
 1C0 |                DMPPGMAP                 |
     └─────────────────────────────────────────┘
```

Displacement Hex	 Dec	Field Name			Field Description, Contents, Meaning
0	0	DMPGPRS	DS	16F	16 General Registers
40	64	DMPCRS	DS	16F	16 Control Registers
80	128	DMPFPRS	DS	4D	4 Floating Point Registers (if Floating-Point Feature is installed.)
A0	160	DMPTODCK	DS	1D	Time-of-day clock
A8	168	DMPCPUTM	DS	1D	CPU timer
B0	176	DMPCKCOM	DS	1D	Time-of-day clock comparator
B8	184	DMPFLAG	DS	1X	S*1 - flag byte
		Bits Defined in DMPFLAG			
		HALFPAGE	EQU	X'80'	Last record in DUMP file = 2K
B9	185	DMPRSV1	DS	1X	S*2 - reserved for IBM use
BA	186	DMPRSV2	DS	1H	Reserved for IBM use
BC	188	DMPSYSRV	DS	1F	System generated storage size
C0	192	DMPLCORE	DS	256X	Locations 0-256 of storage memory
1C0	448	DMPPGMAP	DS	4096B	Bit map indicating which pages appear in the DUMP file (each bit represents a 4K block)

DMPKYREC - DUMP FILE KEY RECORD

```
    ┌──────────────────────────────────────────┐
  0 │S*1 │                                     │
    │────┘          DMPKEYS                    │
    │                                          │
    └──────────────────────────────────────────┘
```

Displacement		Field			
Hex	Dec	Name			Field Description, Contents, Meaning
0	0	DMPKEYS	DS	4096X	Main storage keys
			ORG	DMPKEYS	
0	0	DMPKEY	DS	1X	S*1 - storage key for each 2K block

DMPTBREC - DUMP FILE SYMBOL TABLE RECORD

```
  0 ┌──────────────────────────────────────────────┐
    │                  DMPSYMNM                    │
    ├──────────────────────────────────────────────┤
  8 │      DMPSYMVA          │                     │
    ├────────────────────────┘                     │
    │                  DMPSYMEN                    │
    └──────────────────────────────────────────────┘
```

Displacement		Field			
Hex	Dec	Name			Field Description, Contents, Meaning
0	0	DMPSYMEN	DS	341XL12	Symbol table entries
			ORG	DMPSYMEN	
0	0	DMPSYMNM	DS	CL8	CSECT or entry point name
8	8	DMPSYMVA	DS	A	Location in main storage of this symbol

ECBLOK - EXTENSION TO VMBLOK FOR VIRTUAL MACHINE WITH RELOCATE WITH RELOCATE

```
    ┌─────────────────────────────────────────┐
  0 |    EXTCR0          |     EXTCR1         |
    |-----------------------------------------|
  8 |    EXTCR2          |     EXTCR3         |
    |-----------------------------------------|
 10 |    EXTCR4          |     EXTCR5         |
    |-----------------------------------------|
 18 |    EXTCR6          |     EXTCR7         |
    |-----------------------------------------|
 20 |    EXTCR8          |     EXTCR9         |
    |-----------------------------------------|
 28 |    EXTCR10         |     EXTCR11        |
    |-----------------------------------------|
 30 |    EXTCR12         |     EXTCR13        |
    |-----------------------------------------|
 38 |    EXTCR14         |     EXTCR15        |
    |-----------------------------------------|
 40 |    EXTSHCR0        |     EXTSHCR1       |
    |-----------------------------------------|
 48 |EXTSHLEN |EXTCPLEN |     EXTCOPY        |
    |-----------------------------------------|
 50 |    EXTSHSEG        |EXTSEGLN |EXTARCH  |
    |-----------------------------------------|
 58 |    EXTPERAD        |EXTPERCD |EXTRSV1  |
    |-----------------------------------------|
 60 |               EXTCPTMR                  |
    |-----------------------------------------|
 68 |    EXTCPTRQ        |     EXTCCTRQ       |
    └─────────────────────────────────────────┘
```

Displacement Hex	 Dec	Field Name			Field Description, Contents, Meaning
0	0	EXTCR0	DS	1F	Virtual control register 0; architecture controls
4	4	EXTCR1	DS	1F	Virtual control register 1; segment table pointer
8	8	EXTCR2	DS	1F	Virtual control register 2
C	12	EXTCR3	DS	1F	...thru register 15
10	16	EXTCR4	DS	1F	
14	20	EXTCR5	DS	1F	
18	24	EXTCR6	DS	1F	
1C	28	EXTCR7	DS	1F	
20	32	EXTCR8	DS	1F	
24	36	EXTCR9	DS	1F	
28	40	EXTCR10	DS	1F	
2C	44	EXTCR11	DS	1F	
30	48	EXTCR12	DS	1F	

```
34   52        EXTCR13  DS     1F
38   56        EXTCR14  DS     1F
3C   60        EXTCR15  DS     1F
40   64        EXTSHCR0 DS     1F              Shadow control register 0
44   68        EXTSHCR1 DS     1F              Shadow control register 1
48   72        EXTSHLEN DS     1H              Length of shadow SEGTABLE in bytes
4A   74        EXTCPLEN DS     1H              Length of copy SEGTABLE in bytes
4C   76        EXTCOPY  DS     1F              Pointer to copy segment table
50   80        EXTSHSEG DS     1F              Real address of shadow SEGTABLE
54   84        EXTSEGLN DS     1H              Length of shadow SEGTABLE in doublewords
56   86        EXTARCH  DS     1H              Architecture control index
58   88        EXTPERAD DS     1F              PER interrupt address
5C   92        EXTPERCD DS     1H              PER interrupt code to be reflected
5E   94        EXTRSV1  DS     1H              Reserved for IBM use
60   96        EXTCPTMR DS     1D              Virtual CPU timer
68  104        EXTCPTRQ DS     1F              Address of TRQBLOK for CPU timer
6C  108        EXTCCTRQ DS     1F              Address of TRQBLOK for clock comparator

               EXTSIZE  EQU    (*-ECBLOK)/8    ECBLOK size in doublewords (X'0E')
```

IOBLOK - I/O CONTROL BLOCK

```
    ┌--------------------------------------------┐
  0 |IOBRADD  |I*1 |I*2 |      IOBLINK           |
    |--------------------------------------------|
  8 |   IOBFPNT         |      IOBBPNT           |
    |--------------------------------------------|
 10 |IOBCYL   |IOBVADD  |      IOBMISC           |
    |--------------------------------------------|
 18 |   IOBUSER         |      IOBIRA            |
    |--------------------------------------------|
 20 |   IOBCAW          |      IOBRCAW           |
    |--------------------------------------------|
 28 |              IOBCSW                        |
    |--------------------------------------------|
 30 |   IOBIOER         |      IOBMISC2          |
    |--------------------------------------------|
 38 |I*3 |V*1 |   V*2   |          V*3           |
    └--------------------------------------------┘
```

Displacement Hex	 Dec	Field Name			Field Description, Contents, Meaning
0	0	IOBRADD	DS	1H	Real device address for SIO
2	2	IOBFLAG	DS	1X	I*1 - IOBLOK flags
		Bits Defined in IOBFLAG			
		IOBCP	EQU	X'80'	CP generated I/O operation
		IOBRSTRT	EQU	X'40'	Restarted operation - IOBRCAW
		IOBSPLT	EQU	X'20'	DASD - CP split seek operation
		IOBPAG	EQU	X'10'	IOBLOK created for paging I/O
		IOBRELCU	EQU	X'08'	Control unit released at initiation
		IOBERP	EQU	X'04'	I/O task is under control of ERP
		IOBRES	EQU	X'02'	I/O task has been reset
		IOBHVC	EQU	X'01'	I/O initiated via DIAGNOSE instruction
3	3	IOBSTAT	DS	1X	I*2 - IOBLOK status
		Bits Defined in IOBSTAT			
		IOBFATAL	EQU	X'80'	Uncorrectable error in this I/O operation
		IOBUC	EQU	X'40'	Unit check status
		IOBSNSIO	EQU	X'20'	Sense operation (IOBSNSE)
		IOBREQUE	EQU	X'10'	Restarted operation (IOBCAW)
		IOBWRAP	EQU	X'08'	I/O task for autopoll wrap list
		IOBCC0	EQU	X'00'	Processing I/O interrupt
		IOBCC1	EQU	X'01'	Processing CC 1, CSW stored
		IOBCC2	EQU	X'02'	Processing CC 2, channel busy
		IOBCC3	EQU	X'03'	Processing CC 3, not available
4	4	IOBLINK	DS	1F	Reserved for IBM use

```
     8    8         IOBFPNT  DS    1F            Pointer to next IOBLOK in queue
     C   12         IOBBPNT  DS    1F            Pointer to previous IOBLOK in queue

|                   IOBMSIZE EQU   (*-IOBLOK)/8  Multiple path IOBLOK size in dbl. wds (X'02')

    10   16         IOBCYL   DS    1H            DASD - seek cylinder for this IOBLOK
|   12   18         IOBVADD  DS    1H            Virtual device address
    14   20         IOBMISC  DS    1F            Use varies according to caller
    18   24         IOBUSER  DS    1F            Pointer to VMBLOK of user
    1C   28         IOBIRA   DS    1F            ICBLOK interrupt return address
    20   32         IOBCAW   DS    1F            Pointer to CCW chain
    24   36         IOBRCAW  DS    1F            Pointer to restart CCW chain
    28   40         IOBCSW   DS    1D            Real CSW for I/O operation
    30   48         IOBIOER  DS    1F            Pointer to IOERBLOK with sense
    34   52         IOBMISC2 DS    1F            Use varies according to caller
|   38   56         ICBSPEC  DS    1X            I*3 - IOBLOK special requests
|                   Bits Defined in IOBSPEC
|                   IOBTIO   EQU   X'80'         IOBLOK request for a 'TIO'
|                   IOBHIO   EQU   X'40'         IOBLOK request for a 'HIO'

|   39   57         IOBSV1   DS    1X            V*1 reserved for IBM use
|   3A   58         IOBSV2   DS    XL2           V*2 reserved for IBM use
|   3C   60         IOBSV3   DS    1F            V*3 reserved for IBM use

                    IOBSIZE   EQU   (*-IOBLOK)/8  IOBLOK size in doublewords (X'08')

                    For CP IOBLCKs
                             ORG   IOBVADD
    12   18         IOBRCNT  DS    1H            Retry count

|
```

IOERBLOK - I/O ERROR INFORMATION BLOCK

```
               ┌───────────────────────────────────────┐
             0 │    IOERPNT         │    IOERLOC        │
               │───────────────────────────────────────│
             8 │ IOERDW  │   IOERMSG    │I*1 │I*2 │W*1 │
               │───────────────────────────────────────│
            10 │              IOERADR                  │
               │───────────────────────────────────────│
            18 │              IOERCSW                  │
               │───────────────────────────────────────│
            20 │              IOERCCW        │ IOERLEN │
               │───────────────────────────────────────│
            28 │ IOEREXT │         IOERSV1             │
               │───────────────────────────────────────│
            30 │              IOERDATA                 │
               │───────────────────────────────────────│
|           48 │ Additional sense data area for devices│
|              │ that return more than 24 sense bytes. │
|              │ See XOBR3211 which follows.           │
               └───────────────────────────────────────┘
```

Displacement Hex	Dec	Field Name			Field Description, Contents, Meaning
0	0	IOERPNT	DS	1F	Pointer to next IOERBLOK
4	4	IOERLOC	DS	1F	Address of CCWs used in recovery
8	8	IOERDW	DS	1H	Size in doublewords of storage to construct CCWs
A	10	IOERMSG	DS	XL3	Communications with ERP and message writer
			ORG	IOERMSG	
A	10	IOERNUM	DS	1X	Message number for message writer
B	11	IOERIND3	DS	1X	Indicators for message writer
		Bits Defined in IOERIND3			
		IOERIGN	EQU	X'80'	Allow IGNORE response
		IOERETRY	EQU	X'40'	Allow RETRY response
		IOERCAN	EQU	X'20'	Allow CANCEL response
		IOEREC	EQU	X'10'	Error occurred during recovery action
		IOERDASD	EQU	X'08'	Home address is present
		IOERDEC	EQU	X'04'	Operator decision is necessary
		IOERINFO	EQU	X'02'	Informational message
		IOERACT	EQU	X'01'	Operator action is required
C	12	IOERIND4	DS	1X	Indicators for message writer
		Bits Defined in IOERIND4			
		IOERIGNR	EQU	X'80'	Operator responded IGNORE
		IOERSTRT	EQU	X'40'	Operator responded RETRY

```
 D   13         IOERFLG1 DS    1X             I*1 - IOERFLG1 field
                Bits Defined in IOERFLG1
                IOERPEND EQU   X'80'          Pending device end interrupt from interrupt request
                IOERCLN  EQU   X'40'          Tape cleaning in progress
                IOERERP  EQU   X'40'          Spooling - error routine in control
                IOERFSR  EQU   X'20'          Forward space record being executed
                IOERDEPD EQU   X'20'          Spooling - waiting for device end
                IOERBSR  EQU   X'10'          Backspace record being executed
                IOERDERD EQU   X'10'          Spooling - device end received
                IOERERG  EQU   X'08'          Erase gap command in progress
                IOERORA  EQU   X'04'          Opposite recovery action in progress
                IOERSUPP EQU   X'02'          CCW has suppress data transfer bit on
                IOERVLD  EQU   X'01'          Read opposite recovery successful

 E   14         IOERFLG2 DS    1X             I*2 - IOERFLG2 field
                Bits Defined in IOERFLG2
                IOERSTAT EQU   X'80'          Statistical data being unloaded
                IOERHA   EQU   X'40'          DASD home address being read
                IOERCAL  EQU   X'20'          Stand alone recalibrate being executed
                IOERECF  EQU   X'10'          Error correction function
                IOERREW  EQU   X'08'          Tape rewind being executed
                IOERCYLR EQU   X'04'          Cylinder (in sense byte) has been relocated
                IOERCEMD EQU   X'02'          Intensive recording mode

 F   15         IOERWRK  DS    1X             W*1 - Miscellaneous work area
10   16         IOERADR  DS    1D             Home address for DASD devices
18   24         IOERCSW  DS    1D             CSW associated with error
20   32         IOERCCW  DS    1D             Sense CCW used to sense the real device

                         ORG   IOERCCW+6
26   38         IOERLEN  DS    1H             Number of sense bytes present
28   40         IOEREXT  DS    1H             Size of extended sense area in doublewords
2A   42         IOERSV1  DS    XL6            Reserved for IBM use
30   48         IOERDATA DS    3D             Sense bytes associated with error

|               IOERSIZE EQU   (*-IOERBLOK)/8 IOERBLOK size in doublewords (X'09')
```

IOERBLOK DSECT CCNTINUE

XOBR3211 - EXTENDED OUTBOARD RECORDING BLOCK

```
         ┌──────────────────────────────────────────┐
    48   |                 XOBRCCW1                  |
         |──────────────────────────────────────────|
    50   |                 XOBRCCW2                  |
         |──────────────────────────────────────────|
    58   |                 XOBRCCW3                  |
         |──────────────────────────────────────────|
    60   |                 XOBRCCW4                  |
         |──────────────────────────────────────────|
    68   | X*1| X*2| XOBRMIS1|       XOBRMIS2        |
         |──────────────────────────────────────────|
    70   |                                           |
         |                 XOBR512                   |
         |                                           |
         |──────────────────────────────────────────|
   270   |                                           |
         |                 XOBR180                   |
         |                                           |
         |──────────────────────────────────────────|
   328   |                 XOBR010                   |
         |          ┌───────────────────────────────|
         |          |           XOBRVS1             |
         └──────────────────────────────────────────┘
```

Displacement Hex	Dec	Field Name			Field Description, Contents, Meaning
48	72	XOBRCCW1	DS	1D	CCW used to read OBR information
50	80	XOBRCCW2	DS	1D	CCW used to read OBR information
58	88	XOBRCCW3	DS	1D	CCW used to read OBR information
60	96	XOBRCCW4	DS	1D	CCW used to read OBR information
68	104	XOBRFLAG	DS	1X	X*1 - XOBRFLAG field
		Bits Defined in XOBRFLAG			
		XOBRT1	EQU	X'80'	T1 Buffer type information present
		XOBRT2	EQU	X'40'	T2 Buffer type information present
		XOBRT3	EQU	X'20'	T3 Buffer type information present
69	105	XOBRSTAT	DS	1X	X*2 - XOBRSTAT field
		Bits defined in XOBRSTAT			
		XOBRRT1	EQU	X'80'	Perform routine 1 in error module
		XOBRRT2	EQU	X'40'	Perform routine 2 in error module
		XOBRRT3	EQU	X'20'	Perform routine 3 in error module
		XOBRRT4	EQU	X'10'	Perform routine 4 in error module
		XOBRRT5	EQU	X'08'	Perform routine 5 in error module

```
|                     XOBRRT6  EQU   X'04'          Perform routine 6 in error module
|                     XOBRRT7  EQU   X'02'          Perform routine 7 in error module
|                     XOBRRT8  EQU   X'01'          Perform routine 8 in error module

|  6A   106           XOBRMIS1 DS    1H             Used by the error routine
|  6C   108           XOBRMIS2 DS    1F             Used by the error routine
|  70   112           XOBR512  DS    CL512          Space for USCB data
| 270   624           XOBR180  DS    CL184          Space for FCB data

|                              ORG   XOBR180
| 270   624           XOBR150  DS    CL150          Space for PLB check data
|                              ORG
| 328   808           XCBR010  DS    CL10           Space for first ten error characters
| 332   818           XOBRSV1  DS    CL6            Reserved for IBM use

|                     XOBRSIZE EQU   (*-IOERBLOK)/8 Size of IOER and XOBR in double words (X'67')
|                     XOBREXT  EQU   (*-XOBRCCW1)/8 Size of XOBR3211 in double words (X'5E')
```

IRMBLOCK - INTENSIVE ERROR RECORDING MODE BLOCK

```
  ┌──────────────────────────────────────────────┐
0 │       IRMFWPTR        │ IRMRLADD │  IRMLMT   │
  │──────────────────────────────────────────────│
8 │ I*1 │ I*2│ I*3│ I*4 │ IRMLMTCT │ I*5│ I*6  │
  └──────────────────────────────────────────────┘
```

Displacement Hex	 Dec	Field Name			Field Description, Contents, Meaning
0	0	IRMFWPTR	DS	1F	Reserved for IBM use
4	4	IRMRLADD	DS	1H	Device address
6	6	IRMLMT	DS	1H	Limit count - every 'nth' record is requested.
8	8	IRMBYT1	DS	1X	I*1 - first sense byte specified
9	9	IRMBIT1	DS	1X	I*2 - sense bit within first sense byte
A	10	IRMBYT2	DS	1X	I*3 - second sense byte specified
B	11	IRMBIT2	DS	1X	I*4 - sense bit within second sense byte
C	12	IRMLMTCT	DS	1H	Temporary summary count for limit detection
E	13	IRMMAXCT	DS	1X	I*5 - count of recordings made for this request
F	15	IRMFLG	DS	1X	I*6 - flag byte
		Bits Defined in IRMFLG			
		IRMAND	EQU	X'80'	AND condition specified
		IRMOR	EQU	X'40'	OR condition specified
		IRMSIZE	EQU	(*-IRMBLOK)/8	IRMBLOK size in doublewords (X'02')

MCHAREA - MACHINE CHECK SAVE AREA

```
     ┌─────────────────────────────────────────────────┐
   0 |MCDAMLEN |       MCHRESEV                        |
     |─────────────────────────────────────────────────|
   8 |M*1  |M*2  |M*3  |M*4  |M*5  |M*6  |M*7  |M*8  |
     |─────────────────────────────────────────────────|
  10 |                                                 |
     =                 MCHLSUM                         =
     |                                                 |
     |─────────────────────────────────────────────────|
  38 |N*1  |N*2  |N*3  |N*4  |N*5  |N*6  |N*7  |N*8  |
     |─────────────────────────────────────────────────|
  40 |   MCHFSAR              |      MCHFSAV           |
     |─────────────────────────────────────────────────|
  48 |   MCHFSEAV             |      MCHPDARI          |
     |─────────────────────────────────────────────────|
  50 |L*1  |L*2  |L*3  |L*4  |CPULIMIT  |MCHRES1       |
     |─────────────────────────────────────────────────|
  58 |   BUFDIA55             |      BUF55DIA          |
     |─────────────────────────────────────────────────|
  60 |   BUFENA55             |      BUF55ENA          |
     |─────────────────────────────────────────────────|
  68 |   ECCDIS55             |      ECC55DIS          |
     |─────────────────────────────────────────────────|
  70 |   ECCENA55             |      ECC55ENA          |
     |─────────────────────────────────────────────────|
  78 |                 BUFDIA65                        |
     |─────────────────────────────────────────────────|
  80 |                 BUFENA65                        |
     |─────────────────────────────────────────────────|
  88 |                 ECCDIS65                        |
     |─────────────────────────────────────────────────|
  90 |                 ECCENA65                        |
     └─────────────────────────────────────────────────┘
```

Displacement Hex	Dec	Field Name	Field Description, Contents, Meaning
0	0	MCDAMASS DS 0D	Damage assessment
0	0	MCDAMLEN DS 1H	Length of the damage assessment field
2	2	MCHRESEV DS XL6	Reserved for IBM use
8	8	MCHDAMFL DS 0BL8	Damage assessment data
8	8	MCHFLAG0 DS 1X	M*1 System status
		Bits defined in MCHFLAG0	
		MCH0HDWR EQU X'80'	Hardware recovery
		MCH0SFTR EQU X'40'	Software recovery
		MCH0USAD EQU X'20'	User aborted

```
                   MCHOTERM EQU   X'08'            Operating system termination
                   MCHOQUIT EQU   X'04'            Quiet mode in effect

  9    9           MCHFLAG1 DS    1X               M*2 Damage area
                   Bits defined in MCHFLAG1
                   MCH1MAIN EQU   X'80'            Main storage
                   MCH1BUFF EQU   X'40'            Buffer
                   MCH1COST EQU   X'20'            Control storage
                   MCH1PROC EQU   X'08'            Processor
                   MCH1TODC EQU   X'02'            Time-of-day clock
                   MCH1SYSD EQU   X'01'            System damage

  A   10           MCHFLAG2 DS    1X               M*3 Damage area (continued)
  B   11           MCHFLAG3 DS    1X               M*4 Error type
                   Bits defined in MCHFLAG3
                   MCH3INTE EQU   X'80'            Intermittent
                   MCH3SOLD EQU   X'40'            Solid
                   MCH3DATA EQU   X'20'            Data
                   MCH3PROT EQU   X'10'            Protect

  C   12           MCHFLAG4 DS    1X               M*5 RMS Action data
                   Bits defined in MCHFLAG4
                   MCH4TOLO EQU   X'80'            Time out loop
                   MCH4REPA EQU   X'40'            Repair
                   MCH4STRE EQU   X'20'            Storage reconfigure
                   MCH4BURE EQU   X'10'            Buffer reconfigure

  D   13           MCHFLAG5  DS    1X              M*6 RMS Information status
                   Bits defined in MCHFLAG5
                   MCH5INLG EQU   X'80'            Invalid logout
                   MCH5INMC EQU   X'40'            Invalid machine check interrupt code
                   MCH5IFSA EQU   X'20'            Invalid failing storage address

  E   14           MCHFLAG6 DS    1X               M*7 RMS wait state suffix
  F   15           MCHFLAG7 DS    1X               M*8 RMS information status
                   Bits defined in MCHFLAG7
                   MCH7SMCR EQU   X'80'            Second machine check recursion
                   MCH7VRTM EQU   X'40'            Terminate the virtual user
                   MCH7OPSW EQU   X'10'            M.C. old PSW in problem state
                   MCH7VEQR EQU   X'08'            Terminate the Virtual equal Real user

 10   16           MCHLSUM  DS    1X               Summary
 38   56           MCHPDAR  DS    0BL8
 38   56           MCHPDAR0 DS    1X               N*1 Action taken
 39   57           MCHPDAR1 DS    1X               Failure type
                   Bits defined in MCHPDAR1
                   MCHP1SDE EQU   X'80'            Solid storage data error
                   MCHP1IDE EQU   X'40'            Intermittent storage data error
                   MCHP1SKE EQU   X'20'            Solid SPF key error
                   MCHP1IKE EQU   X'10'            Intermittent SPF key error
```

```
3A   58        MCHPDAR2 DS    1X           N*3 Operating system status
3B   59        MCHPDAR3 DS    1X           N*4 Location of failure
3C   60        MCHPDAR4 DS    1X           N*5 Location of failure
3D   61        MCHPDAR5 DS    1X           N*6 Requested operator awareness
3E   62        MCHPDAR6 DS    1X           Footprint
               Bits defined in MCHPDAR6
               MCHP6CBA EQU   X'80'        Change bit active

3F   63        MCHPDAR7 DS    1X           Footprints
               Bits defined in MCHPDAR7
               MCH7STCK EQU   X'80'        Interfaces for STACK routine
               MCH7GSTR EQU   X'40'        Interfaces for GETMAIN routine
               MCH7PURG EQU   X'20'        Interfaces for PURGE routine
               MCH7LOGO EQU   X'10'        Interfaces for V=R LOGOFF routine
               MCH7EXIT EQU   X'08'        Interfaces for exit to CP
               MCH7RSRE EQU   X'04'        Interfaces for RELEASE and RESET routines
               MCH7IOEM EQU   X'02'        Interfaces for the recorder

40   64        MCHFSAR  DS    1F           Failing location real address
44   68        MCHFSAV  DS    1F           Instruction address at failure
48   72        MCHFSEAV DS    1F           End of the failing location
4C   76        MCHPDARI DS    1F           End of failing storage address - virtual

               MCHLEN1  EQU   *-MCDAMASS   Length of damage assessment area
               MCHLEN   EQU   *-MCHRESEV   Length of area to be cleared

50   80        MCHMODEL DS    1X           L*1 The model number for the machine
               Bits defined in MCHMODEL
               NOMODEL  EQU   X'00'        No support for machine
               MODEL135 EQU   X'04'        ID number for the 135 machine
               MODEL145 EQU   X'08'        ID number for the 145 machine
               MODEL155 EQU   X'0C'        ID number for the 155 machine
               MODEL158 EQU   X'0C'        ID number for the 158 machine
               MODEL165 EQU   X'10'        ID number for the 165 machine
               MODEL168 EQU   X'10'        ID number for the 168 machine

51   81        SWITCH   DS    1X           L*2 Main storage exercise switch
52   82        MODEFLAG DS    1X           L*3 Flag field for MODE command
               Bits defined in MODEFLAG
               MODEQUIT EQU   X'80'        ECC is in QUIET mode

53   83        MODFLAG1 DS    1X           L*4 Flag field for message indicator in MODE command
               Bits defined in MODFLAG1
               MOD1RETY EQU   X'80'        Message indicator for RETRY message
               MOD1QUIT EQU   X'40'        Message indicator for QUIET message

54   84        CPULIMIT DS    1H           The count field for soft error
56   86        MCHRES1  DS    1H           Reserved for IBM use
58   88                 DS    0D
```

```
58   88         BUFDIA55 DC    X'0100D100'     Disable buffer for Model 155
5C   92         BUF55DIA DS    1F              Reserved for IBM use
60   96         BUFENA55 DC    X'0200D100'     Enable buffer for Model 155
64  100         BUF55ENA DS    1F              Reserved for IBM use
68  104         ECCDIS55 DC    X'0300D100'     Disable ECC for Model 155
6C  108         ECC55DIS DS    1F              Reserved for IBM use
70  112         ECCENA55 DC    X'0400D100'     Enable ECC for Model 155
74  116         ECC55ENA DS    1F              Reserved for IBM use
78  120         BUFDIA65 DC    X'0300000000000000'  Disable buffer for Model 165
80  128         BUFENA65 DC    X'0300002000000000'  Enable buffer for Model 165
88  136         ECCDIS65 DC    X'0200000003000000'  Disable ECC for Model 165
90  144         ECCENA65 DC    X'0200000000000000'  Enable ECC for Model 165

                MCHFIX   EQU   280+48          The length of the fixed logout and header record for machine
                                                 check handler
                MCHLEN2  EQU   *-MCDAMASS      The communication area length
```

MCRECORD - MACHINE CHECK HANDLER RECORD

```
      ┌────────────────────────────────────────┐
    0 |M*1 |M*2 |M*3 |M*4| MCSWITCH|M*5 |M*6 |
      |────────────────────────────────────────|
    8 |              MCDATE                    |
      |────────────────────────────────────────|
   10 |              MCCPUID                   |
      |────────────────────────────────────────|
   18 |              MCPROGID                  |
      |────────────────────────────────────────|
   20 |              MCJOBID                   |
      |────────────────────────────────────────|
   28 |              MCOLDPW                   |
      |────────────────────────────────────────|
   30 |                                        |
      =              MCFXDLOG                  =
      =                                        =
      |                                        |
      |────────────────────────────────────────|
  148 |                                        |
      =                                        =
      =             MCEXTLOG                   =
      =             (variable length)          =
      |                                        |
      |────────────────────────────────────────|
      |                                        |
      =             MCHDAMAG                   =
      |                                        |
      └────────────────────────────────────────┘
```

Displacement Hex	 Dec	Field Name			Field Description, Contents, Meaning
0	0	MCREC	DS	0D	
0	0	MCRECTYP	DS	1X	M*1 Machine check record type
1	1	MCOPSYS	DS	1X	M*2 Operating system
2	2	MCSWONE	DS	1X	M*3 Record independent switch
3	3	MCSWTWO	DS	1X	M*4 Record dependent switch
4	4	MCSWITCH	DS	2X	Unused switches
6	6	MCRECCNT	DS	1X	M*5 Record count
7	7	MCRECCC	DS	1X	M*6 Spare
8	8	MCDATE	DS	XL8	Date and time
10	16	MCCPUID	DS	XL8	CPU identification
18	24	MCPROGID	DS	XL8	Program identity
20	32	MCJOBID	DS	XL8	Job identity (unused)
28	40	MCOLDPW	DS	XL8	Machine check old PSW
30	48	MCFXDLOG	DS	35D	Machine check fixed logout

```
                        FXDLGLH  EQU   (*-MCFXDLOG)

148  328                MCEXTLOG EQU   *              Machine check extended logout (the extended logout length is
                                                        variable length -- machine dependent)
                        MCHDAMAG EQU   *              The damage assessment area (80 bytes)
```

OWNDLIST - CP OWNED VOLUMES LIST

```
  ┌─────────────────────────────────────┐
0 │             OWNDVSER       │OWNDRDEV │
  └─────────────────────────────────────┘
```

Displacement Hex	 Dec	Field Name			Field Description, Contents, Meaning
0	0	OWNDVSER	DS	CL6	Volume serial number
6	6	OWNDRDEV	DS	1H	Displacement of RDEVBLOK for the volume
			ORG	OWNDRDEV	
6	6	OWNDPREF	DS	1X	Allocation preference

PAGTABLE - PAGE TABLE

```
  ┌─────────────────────────────────────┐
0 │    PAGRSV1         │     PAGSWP      │
  │─────────────────────────────────────┘
8 │    PAGCORE         │
  └────────────────────┘
```

Displacement Hex	 Dec	Field Name			Field Description, Contents, Meaning
0	0	PAGRSV1	DS	1F	Reserved to align PAGCORE on a doubleword
4	4	PAGSWP	DS	1F	Pointer to SWPTABLE
8	8	PAGCORE	DS	1H	Real page address
		Bits Defined in PAGCORE+1			
		PAGINVAL	EQU	X'08'	PAGTABLE entry invalid
		PAGREF	EQU	X'01'	Page has been referenced

PSA - PREFIX STORAGE AREA (LOW STORAGE LOCATIONS)

```
              Page 0, Machine Usage
      ------------------------------------------
  0  |   IPLPSW            |    IPLCCW1         |
     |------------------------------------------|
 10  |   IPLCCW2           |    EXOPSW          |
     |------------------------------------------|
 20  |   SVCOPSW           |    PROPSW          |
     |------------------------------------------|
 30  |   MCOPSW            |    IOOPSW          |
     |------------------------------------------|
 40  |   CSW               |   CAW    |QUANTUMR |
     |------------------------------------------|
 50  | TIMER   |QUANTUM    |    EXNPSW          |
     |------------------------------------------|
 60  |   SVCNPSW           |    PRNPSW          |
     |------------------------------------------|
 70  |   MCNPSW            |    IONPSW          |
     |------------------------------------------|
 80  |                CPULOG                    |
     |------------------------------------------|
100  |                FXDLOG                    |
     |------------------------------------------|
160  |                 FPRLOG                   |
     |------------------------------------------|
180  |                 GRLOG                    |
     |------------------------------------------|
1C0  |                 CRLOG                    |
     |------------------------------------------|
200  |                 TEMPSAVE                 |
     |------------------------------------------|
240  |                 BALRSAVE                 |
     |------------------------------------------|
280  |                 FREESAVE                 |
     |------------------------------------------|
2C0  |                 FREEWORK                 |
     |------------------------------------------|
2F0  |   DATE              |    TODATE          |
      ------------------------------------------
```

```
      ------------------------------------------
300  |   STARTIME         |     CPUID          |
     |------------------------------------------|
310  |   IDLEWAIT         |     PAGEWAIT       |
     |------------------------------------------|
320  |   IONTWAIT         |     PROBTIME       |
     |------------------------------------------|
330  |   RUNPSW           |RUNUSER  |DSPLPSW   |
     |------------------------------------------|
340  |RUNCR0    |RUNCR1   |CPSTAT   |CPRESTRT  |
     |------------------------------------------|
350  |PGREAD    |PGWRITE  |     PGWAITIM       |
     |------------------------------------------|
360  |   PGWAITPG         |PSASVCCT |P*1 |P*2  |
     |------------------------------------------|
370  |CPID      |CPABEND  |P*3 |P*4 |ASYSVM    |
     |------------------------------------------|
380  |ARSPPR    |ARSPPU   |ARSPRD   |ARIOPU    |
     |------------------------------------------|
390  |ARIOPR    |ARIORD   |PSARSV6  |ARSPAC    |
     |------------------------------------------|
3A0  |AVMREAL   |ASYSABND |ASYSLC   |ASYSOP    |
     |------------------------------------------|
3B0  |ARIOCT    |ARIOCH   |ARIOCU   |ARIODV    |
     |------------------------------------------|
3C0  |ARIOCC    |ARIOUC   |ARIODC   |ACORETBL  |
     |------------------------------------------|
3D0  |APAGCP    |CPCREG0  |CPCREG8  |PSARSV9   |
     |------------------------------------------|
3E0  |PSARSV10  |PSARSV11 |ADMKFVR  |XVRINST   |
     |------------------------------------------|
3F0  |PAGECUR   |MONNEXT  |PAGEND   |PAGENXT   |
     |------------------------------------------|
400  |TRACEFLG  |PSARSV12 |                    |
     |--------------------                     |
     |                                          |
     |               PSARSV15                   |
     |------------------------------------------|
430  |INSTWRD1  |INSTWRD2 |INSTWRD3 |INSTWRD4  |
      ------------------------------------------
```

Displacement		Field			
Hex	Dec	Name			Field Description, Contents, Meaning
0	0	IPLPSW	DS	1D	IPL start PSW
8	8	IPLCCW1	DS	1D	IPL CCW
10	16	IPLCCW2	DS	1D	IPL CCW
			ORG	IPLCCW1	
8	8	PSARSV3	DS	1F	Reserved for IBM use
C	12	TRACSTRT	DS	1F	Pointer to start of trace table
10	16	TRACEND	DS	1F	Pointer to end of trace table
14	20	TRACCURR	DS	1F	Pointer to next available trace table entry
18	24	EXOPSW	DS	1D	External old PSW
20	32	SVCOPSW	DS	1D	SVC old PSW
28	40	PROPSW	DS	1D	Program old PSW
30	48	MCOPSW	DS	1D	Machine check old PSW
38	56	IOOPSW	DS	1D	I/O old PSW
40	64	CSW	DS	1D	Channel status word
48	72	CAW	DS	1F	Channel address word
4C	76	QUANTUMR	DS	1F	Interval timer value at last interrupt
50	80	TIMER	DS	1F	13 microsecond interval timer
54	84	QUANTUM	DS	1F	Interval timer value at last dispatch
58	88	EXNPSW	DS	1D	External new PSW
60	96	SVCNPSW	DS	1D	SVC new PSW
68	104	PRNPSW	DS	1D	Program new PSW
70	112	MCNPSW	DS	1D	Machine check new PSW
78	120	IONPSW	DS	1D	I/O new PSW
80	128	CPULOG	DS	16D	CPU and storage logout area
			ORG	CPULOG	
80	128		DS	1F	Reserved for IBM use
84	132	INTEXF	DS	1F	External interrupt code (fullword)
86	134	INTEX	EQU	INTEXF+2	External interrupt code (halfword)
88	136	INTSVCL	DS	1H	SVC instruction length code (ILC)
8A	138	INTSVC	DS	1H	SVC interrupt code
8C	140	INTPRL	DS	1H	Program instruction length code (ILC)
8E	142	INTPR	DS	1H	Program interrupt code
90	144	TREXADD	DS	1F	Translation exception address
94	148	MONCLASS	DS	1H	Monitor class
96	150	PERCODE	DS	1H	PER interrupt code
98	152	PERADD	DS	1F	PER interrupt address
9C	156	MONCODE	DS	1F	Monitor code
A0	160		DS	1D	Reserved for IBM use
A8	168	CHANID	DS	1F	Channel identification
AC	172	IOELPNTR	DS	1F	I/O extended logout (IOEL) pointer
B0	176	ECSWLOG	DS	1F	Limited channel logout (ECSW)
B4	180		DS	1F	Reserved for IBM use
B8	184	INTKFLIN	DS	1F	I/O interrupt key, flags, interface address
BA	186	INTTIO	EQU	INTKFLIN+2	I/O interrupt device address (halfword)
BC	188		DS	11F	Reserved for IBM use

```
 E8  232       INTMC    DS    1D         Machine check interrupt code
 F0  240                DS    1D         Reserved for IBM use
 F8  248       FAILSTAD DS    1F         Failing storage address
 FC  252       REGNCODE DS    1F         Region code

100  256       FXDLOG   DS    12D        Fixed logout area
160  352       FPRLOG   DS    4D         Floating-point register logout area
180  384       GRLCG    DS    16F        General register lcgout area
1C0  448       CRLOG    DS    16F        Control register logout area
200  512       CPUSAGE  DS    0H         End of machine usage, start of CP usage
                        ORG   CPUSAGE
200  512       TEMPSAVE DS    16F        Temporary save area
                        ORG   TEMPSAVE
200  512       TEMPR0   DS    1F
204  516       TEMPR1   DS    1F
208  520       TEMPR2   DS    1F
20C  524       TEMPR3   DS    1F
210  528       TEMPR4   DS    1F
214  532       TEMPR5   DS    1F
218  536       TEMPR6   DS    1F
21C  540       TEMPR7   DS    1F
220  544       TEMPR8   DS    1F
224  548       TEMPR9   DS    1F
228  552       TEMPR10  DS    1F
22C  556       TEMPR11  DS    1F
230  560       TEMPR12  DS    1F
234  564       TEMPR13  DS    1F
238  568       TEMPR14  DS    1F
23C  572       TEMPR15  DS    1F

240  576       BALRSAVE DS    16F        BALR linkage save area
                        ORG   BALRSAVE
240  576       BALR0    DS    1F
244  580       BALR1    DS    1F
248  584       BALR2    DS    1F
24C  588       BALR3    DS    1F
250  592       BALR4    DS    1F
254  596       BALR5    DS    1F
258  600       BALR6    DS    1F
25C  604       BALR7    DS    1F
260  608       BALR8    DS    1F
264  612       BALR9    DS    1F
268  616       BALR10   DS    1F
26C  620       BALR11   DS    1F
270  624       BALR12   DS    1F
274  628       BALR13   DS    1F
278  632       BALR14   DS    1F
27C  636       BALR15   DS    1F

280  640       FREESAVE DS    16F        DMKFRE save area
```

```
                         ORG   FREESAVE
280  640       FREER0    DS    1F
284  644       FREER1    DS    1F
288  648       FREER2    DS    1F
28C  652       FREER3    DS    1F
290  656       FREER4    DS    1F
294  660       FREER5    DS    1F
298  664       FREER6    DS    1F
29C  668       FREER7    DS    1F
2A0  672       FREER8    DS    1F
2A4  676       FREER9    DS    1F
2A8  680       FREER10   DS    1F
2AC  684       FREER11   DS    1F
2B0  688       FREER12   DS    1F
2B4  692       FREER13   DS    1F
2B8  696       FREER14   DS    1F
2BC  700       FREER15   DS    1F

2C0  704       FREEWORK  DS    12F             DMKFRE work area
2F0  752       DATE      DS    CL8             Date - mm/dd/yy - edited EBCDIC
2F8  760       TODATE    DS    1D              TOD clock at 00.00.00 today - local time
300  768       STARTIME  DS    1D              Date and time started - TOD clock value
308  776       CPUID     DS    1D              CPU identification
                         ORG   CPUID
308  776       CPUVERSN  DS    1X              Version code
309  777       CPUSER    DS    3X              CPU serial number - packed unsigned
30C  780       CPUMODEL  DS    2X              CPU model number
30E  782       CPUMCELL  DS    1H              MAXIMUM length in bytes of MCEL

310  784       IDLEWAIT  DC    X'7FFFFFFFFFFFF000'  Total system idle wait time
318  792       PAGEWAIT  DC    X'7FFFFFFFFFFFF000'  Total system page wait time
320  800       IONTWAIT  DC    X'7FFFFFFFFFFFF000'  Total system I/O wait time
328  808       PROBTIME  DC    X'7FFFFFFFFFFFF000'  Total system problem state time
330  816       RUNPSW    DS    1D              PSW last loaded by Dispatcher
338  824       RUNUSER   DS    1F              Address of dispatched VMBLOK
33C  828       DSPLPSW   DS    1F              Load PSW instruction used to dispatch
340  832       RUNCR0    DS    1F              Control register zero at dispatch
344  836       RUNCR1    DS    1F              Control register one at dispatch
348  840       CPSTAT    DS    1F              CP running status
                         ORG   CPSTAT
348  840       CPSTATUS  DS    1X              CP running status
               Bits defined in CPSTATUS
               CPWAIT    EQU   X'80'           CP in wait state
               CPRUN     EQU   X'40'           CP running user in RUNUSER
               CPEX      EQU   X'20'           CP executing stacked request
               CPFVRUN   EQU   X'10'           Reserved for IBM use

34C  844       CPRESTRT  DS    1F              Restart address if external interrupt marks page invalid
350  848       PGREAD    DS    1F              Total number of page reads
354  852       PGWRITE   DS    1F              Total number of page writes
```

```
358  856          PGWAITIM DS    1D             Time spent in page wait (TOD units)
360  864          PGWAITPG DS    1D             Time spent in page wait, x pages waiting
368  872          PSASVCCT DS    1F             Total number of user SVCs
36C  876          PAGELOAD DS    1H             P*1 - Page wait percent, last measurement
36E  878          PAGERATE DS    1H             P*2 - Paging rate, pages per second
370  880          PSENDCLR DS    0F             End of area cleared by DMKCPINT
                  CPID     DS    1F             CP running identifier
374  884          CPABEND  DS    1F             CP ABEND code
378  888          PSTARTSV DS    0F             Start of save/restored code
                  SYSIPLDV DS    1H             P*3 - device address of system IPL device
37A  890          PGSRATIO DC    H'0'           P*4 - Page steals/total replenished
37C  892          ASYSVM   DC    V(DMKSYSVM)    Address of system VMBLOK
380  896          ARSPPR   DC    V(DMKRSPPR)    Address of system printer file chain
384  900          ARSPPU   DC    V(DMKRSPPU)    Address of system punch file chain
388  904          ARSPRD   DC    V(DMKRSPRD)    Address of system reader file chain
38C  908          ARIOPU   DC    V(DMKRIOPU)    Address of system punch table
390  912          ARIOPR   DC    V(DMKRIOPR)    Address of system printer table
394  916          ARIORD   DC    V(DMKRIORD)    Address of system reader table
398  920          PSARSV6  DS    1F             Reserved for IBM use
39C  924          ARSPAC   DC    V(DMKRSPAC)    Address of system accounting chain
3A0  928          AVMREAL  DC    A(0)           VMBLOK address of VIRTUAL=REAL user
3A4  932          ASYSABND DC    A(0)           Address of system ABEND printer
3A8  936          ASYSLC   DC    V(DMKSYSLC)    Address of SYSLOCS information
3AC  940          ASYSOP   DC    V(DMKSYSOP)    Address of system operator VMBLOK
3B0  944          ARIOCT   DC    V(DMKRIOCT)    Address of real channel index table
3B4  948          ARIOCH   DC    V(DMKRIOCH)    Address of first RCHBLOK
3B8  952          ARIOCU   DC    V(DMKRIOCU)    Address of first RCUBLOK
3BC  956          ARIODV   DC    V(DMKRIODV)    Address of first RDEVBLOK
3C0  960          ARIOCC   DC    V(DMKRIOCC)    Address of count of real system channels
3C4  964          ARIOUC   DC    V(DMKRIOUC)    Address of count of real system control units
3C8  968          ARIODC   DC    V(DMKRIODC)    Address of count of real system devices
3CC  972          ACORETBL DC    V(DMKSYSCS)    Address of system core table
3D0  976          APAGCP   DC    A(X'FFFFFF')   Address of first pageable program
3D4  980          CPCREG0  DC    X'808008C0'    CP architecture control and external mask
3D8  984          CPCREG8  DC    F'0'           Monitor call enable mask
3DC  988          PSARSV9  DS    1F             Reserved for IBM use
3E0  992          PSARSV10 DS    1F             Reserved for IBM use
3E4  996          PSARSV11 DS    1F             Reserved for IBM use
3E8 1000          ADMKFVR  DC    F'0'           Reserved for IBM use
3EC 1004          XVRINST  DC    F'0'           Reserved for IBM use
3F0 1008          PAGECUR  DS    1F             Reserved for IBM use
3F4 1012          MONNEXT  DS    1F             Reserved for IBM use
3F8 1016          PAGEND   DS    1F             Reserved for IBM use
3FC 1020          PAGENXT  DS    1F             Reserved for IBM use
400 1024          TRACEFLG DS    1F             Trace table flags
404 1028          PSARSV12 DS    1F             Reserved for IBM use
408 1032          PSARSV15 DS    5D             Reserved for IBM use
430 1072          INSTWRD1 DC    F'0'           Reserved for installation use
434 1076          INSTWRD2 DC    F'0'           Reserved for installation use
438 1080          INSTWRD3 DC    F'0'           Reserved for installation use
```

```
43C 1084        INSTWRD4 DC    F'0'           Reserved for installation use

                Pool of frequently used constants:

440 1088        ZEROES   DC   6D'0'
470 1136        BLANKS   DC    8X'40'
478 1144        FFS      DC    8X'FF'         ALSO = -1

440 1088        F0       EQU   ZEROES
480 1152        F1       DC    F'1'
484 1156        F2       DC    F'2'
```

```
488 1160          F3        DC     F'3'
48C 1164          F4        DC     F'4'
490 1168          F5        DC     F'5'
494 1172          F6        DC     F'6'
498 1176          F7        DC     F'7'
49C 1180          F8        DC     F'8'
4A0 1184          F9        DC     F'9'
4A4 1188          F10       DC     F'10'
4A8 1192          F15       DC     F'15'              ALSO = X'0000000F'
4AC 1196          F16       DC     F'16'
4B0 1200          F20       DC     F'20'
4B4 1204          F24       DC     F'24'
4B8 1208          F60       DC     F'60'              ALSO = X'0000003C'
4BC 1212          F240      DC     F'240'             ALSO = X'000000F0' = C'0'
4C0 1216          F255      DC     F'255'             ALSO = X'000000FF'
4C4 1220          F256      DC     F'256'             ALSO = X'00000100'
4C8 1224          F4095     DC     F'4095'            ALSO = X'00000FFF'
4CC 1228          F4096     DC     F'4096'            ALSO = X'00001000'
4D0 1232          APTRLK    DC     V(DMKPTRLK)
4D4 1236          NOADD     DC     X'FF000000'
4D8 1240          X40FFS    DC     X'40FFFFFF'
4DC 1244          XRIGHT24  DC     X'00FFFFFF'
4E0 1248          XPAGNUM   DC     X'00FFF000'
4E4 1252          XRIGHT16  DC     X'0000FFFF'
4E8 1256          AFREE     DC     V(DMKFREE)
4EC 1260          AFRET     DC     V(DMKFRET)
4F0 1264          AQCNWT    DC     V(DMKQCNWT)
4F4 1268          ADSPCH    DC     V(DMKDSPCH)
4F8 1272          APTRAN    DC     V(DMKPTRAN)
4FC 1276          X2048BND  DC     X'00FFF800'

500 1280          PSAEND    DS     0D                 End of page 0 usage.
```

RCHBLOK - REAL CHANNEL BLOCK

```
    ┌─────────────────────────────────────────────┐
 0  |RCHADD    |RCHLOCK   |R*1  |R*2  |RCHRSV1    |
    |─────────────────────────────────────────────|
 8  |      RCHFIOB        |      RCHLIOB          |
    |─────────────────────────────────────────────|
10  |R*3  |R*4  |R*5  |R*6  |      RCHRSV2        |
    |─────────────────────────────────────────────|
18  |     RCHQUED         |      RCHOPER          |
    |─────────────────────────────────────────────|
20  |                RCHCUTBL                     |
    └─────────────────────────────────────────────┘
```

Displacement Hex	 Dec	Field Name			Field Description, Contents, Meaning
0	0	RCHADD	DS	1H	Channel address
2	2	RCHLOCK	DS	1H	Channel lock
4	4	RCHSTAT	DS	1X	R*1 - channel status
		Bits Defined in RCHSTAT			
		RCHBUSY	EQU	X'80'	Channel busy
		RCHSCED	EQU	X'40'	IOB scheduled on channel
		RCHDISA	EQU	X'20'	Channel disabled
		RCHDED	EQU	X'01'	Channel dedicated
5	5	RCHTYPE	DS	1X	R*2 - Channel type
		Bits Defined in RCHTYPE			
		RCHSEL	EQU	X'80'	Selector channel
		RCHBMX	EQU	X'40'	Block-multiplexer channel
		RCHIFA	EQU	X'81'	Selector-type integrated file adapter
6	6	RCHRSV1	DS	1H	Reserved for IBM use
8	8	RCHFIOB	DS	1F	Pointer to first IOBLOK queued
C	12	RCHLIOB	DS	1F	Pointer to last IOBLOK queued
10	16	RCHDTCK	DS	1X	R*3 - channel data check count
11	17	RCHCCCK	DS	1X	R*4 - channel control check count
12	18	RCHIFCC	DS	1X	R*5 - interface control check count
13	19	RCHCHCK	DS	1X	R*6 - channel chaining check count
14	20	RCHRSV2	DS	1F	Reserved for IBM use
18	24	RCHQUED	DS	1F	IOBLOK queued on channel time
1C	28	RCHOPER	DS	1F	ICBLOK operational on channel time
20	32	RCHCUTBL	DS	32H	Control units attached - RCUSTART index
		RCHSIZE	EQU	(*-RCHBLOK)/8	RCHBLOK size in doublewords (X'0C')

RCUBLOK - REAL CONTROL UNIT BLOCK

```
    ┌───────────────────────────────────────────┐
 0  |RCUADD    |RCULOCK   |R*1 |R*2 |RCURSV1   |
    |───────────────────────────────────────────|
 8  |    RCUFIOB           |     RCULIOB        |
    |───────────────────────────────────────────|
10  |    RCUCHA            |     RCUCHB         |
    |───────────────────────────────────────────|
18  |    RCUQUED           |     RCUOPER        |
    |───────────────────────────────────────────|
20  |               RCUDVTBL                    |
    └───────────────────────────────────────────┘
```

Displacement Hex	Dec	Field Name			Field Description, Contents, Meaning
0	0	RCUADD	DS	1H	Control unit address
2	2	RCULOCK	DS	1H	Control unit lock
4	4	RCUSTAT	DS	1X	R*1 - control unit status
		Bits Defined in RCUSTAT			
		RCUBUSY	EQU	X'80'	Control unit busy
		RCUSCED	EQU	X'40'	IOB scheduled on control unit
		RCUDISA	EQU	X'20'	Control unit disabled
		RCUDED	EQU	X'01'	Control unit dedicated
5	5	RCUTYPE	DS	1X	R*2 - control unit type
		Bits Defined in RCUTYPE			
		RCUSHRD	EQU	X'80'	This control unit can attach to only 1 subchannel
		RCU2701	EQU	X'01'	TCU is a 2701
		RCU2702	EQU	X'02'	TCU is a 2702
		RCU2703	EQU	X'03'	TCU is a 2703
6	6	RCURSV1	DS	1H	Reserved for future use
8	8	RCUFIOB	DS	1F	Pointer to first IOBLOK queued
C	12	RCULIOB	DS	1F	Pointer to last IOBLOK queued
10	16	RCUCHA	DS	1F	Pointer to RCHBLOK - interface A
14	20	RCUCHB	DS	1F	Pointer to RCHBLOK - interface B
18	24	RCUQUED	DS	1F	IOBLOK queued on control unit time
1C	28	RCUOPER	DS	1F	IOBLOK operational on control unit time
20	32	RCUDVTBL	DS	16H	Devices attached - RDVSTART index
		RCUSIZE	EQU	(*-RCUBLOK)/8	RCUBLOK size in doublewords (X'08')

RCWTASK - TRANSLATED VIRTUAL I/O CCW

```
    ┌─────────────────────────────────────────┐
  0 │     RCWPNT          │     RCWVCAW        │
    │─────────────────────────────────────────│
  8 │RCWVCNT  │RCWRCNT  │RCWHEAD  │RCWCCNT    │
    │─────────────────────────────────────────│
 10 │                 RCWCCW                    │
    └─────────────────────────────────────────┘
```

Displacement Hex	 Dec	Field Name			Field Description, Contents, Meaning
0	0	RCWPNT	DS	1F	Pointer to next RCWTASK
4	4	RCWVCAW	DS	1F	Virtual address of CCW chain
8	8	RCWVCNT	DS	1H	Virtual CCW count
A	10	RCWRCNT	DS	1H	Real CCW count
C	12	RCWHEAD	DS	1H	RCWTASK header mark X'FFFF'
E	14	RCWCCNT	DS	1H	RCWTASK size in doublewords
10	16	RCWCCW	DS	1D	One or more CCWs for device I/O
			ORG	RCWCCW	
10	16	RCWADDR	DS	1F	CCW data address
14	20	RCWFLAG	DS	1X	CCW flag bits
15	21	RCWCTL	DS	1X	CCW CP control bits
		Bits Defined in RCWCTL			
		RCWIO	EQU	X'80'	I/O data page locked
		RCWGEN	EQU	X'40'	CP generated CCW
		RCWHMR	EQU	X'20'	DMKUNT to relocate home address/record R0
		RCWREL	EQU	X'10'	CCW address relocatable if CCWs moved
		RCWISAM	EQU	X'08'	ISAM modifying CCW
		RCW2311	EQU	X'04'	TYP2311T-B pseudo 2311 on 2314
		RCWIDA	EQU	X'02'	CP generated indirect data address
16	22	RCWCNT	DS	1H	CCW byte count
			ORG	RCWADDR	
10	16	RCWCOMND	DS	1X	CCW command code

RDEVBLOK - REAL DEVICE BLOCK

```
    ┌───────────────────────────────────────────────┐
  0 |RDEVADD   |RDEVLOCK  |R*1  |R*2  |R*3  |R*4  |
    |-----------------------------------------------|
  8 |     RDEVFIOB         |      RDEVLIOB          |
    |-----------------------------------------------|
 10 |     RDEVCUA          |      RDEVCUB           |
    |-----------------------------------------------|
 18 |                RDEVQUED                       |
    |-----------------------------------------------|
 20 |     RDEVIOCT         |      RDEVAIOB          |
    |-----------------------------------------------|
 28 |     RDEVUSER         |RDEVATT    |RDEVCYL    |
    |-----------------------------------------------|
 30 |             RDEVSER              |RDEVLNKS    |
    |-----------------------------------------------|
 38 | R   | D   | E   | V   | T   | C   | T   | L   |
    |-----------------------------------------------|
 40 |     RDEVTMAT         |R*5V|R*6  |R*7  |R*8   |
    |-----------------------------------------------|
 48 |     RDEVICER         |       RDEVCTRS         |
    └───────────────────────────────────────────────┘
```

Displacement Hex	 Dec	Field Name	Field Description, Contents, Meaning
0	0	RDEVADD DS 1H	Device address
2	2	RDEVLOCK DS 1H	Device lock
4	4	RDEVSTAT DS 1X	R*1 - Device status
		Bits Defined in RDEVSTAT	
		RDEVBUSY EQU X'80'	Device busy
		RDEVSCED EQU X'40'	IOB scheduled on device
		RDEVDISA EQU X'20'	Device disabled (offline)
		RDEVRSVD EQU X'10'	Device reserved
		RDEVIRM EQU X'08'	Device in intensive error recording mode
		RDEVNRDY EQU X'04'	Device intervention required
		RDEVDED EQU X'01'	Dedicated device (attached to a user)
5	5	RDEVFLAG DS 1X	R*2 - device flags, device dependent
		Bits Defined in RDEVFLAG	
		RDEVSKUP EQU X'80'	DASD - ascending order seek queuing
		RDEVPREF EQU X'40'	DASD - volume preferred for paging
		RDEVSYS EQU X'20'	DASD - volume attached to system
		RDEVOWN EQU X'10'	DASD - CP owned volume
		RDEVMOUT EQU X'08'	DASD - volume mounted, not attached
		RDEVPSUP EQU X'80'	CONSOLE - terminal has print suppress
		RDEVPREP EQU X'40'	CONSOLE - terminal executing prepare command

```
                  RDEVACTV EQU   X'20'          Console - IOBLOK pending; queue request
                  RDEVIDNT EQU   X'10'          Console - 2741 terminal code identified
                  RDEVENAB EQU   X'08'          Console - device is enabled
                  RDEVHIO  EQU   X'04'          Console - next interrupt from a halt I/O
                  RDEVDISB EQU   X'02'          Console - device is to be disabled
                  RDEVDRAN EQU   X'80'          Spooling - device output drained
                  RDEVTERM EQU   X'40'          Spooling - device output terminated
                  RDEVACNT EQU   X'20'          Spooling - device busy with accounting
                  RDEVSPAC EQU   X'10'          Spooling - force printer to single space
                  RDEVRSTR EQU   X'08'          Spooling - restart current file
                  RDEVBACK EQU   X'04'          Spooling - backspace the current file
                  RDEVSEP  EQU   X'02'          Spooling - print/punch job separator
|                 RDEVLOAD EQU   X'01'          Spooling - UCS buffer verified

   6    6         RDEVTYPC DS    1X             R*3 - device type class (See Appendix C)
   7    7         RDEVTYPE DS    1X             R*4 - device type (See Appendix C)
   8    8         RDEVFIOB DS    1F             Pointer to first IOBLOK queued
   C   12         RDEVLIOB DS    1F             Pointer to last IOBLOK queued
  10   16         RDEVCUA  DS    1F             Pointer to RCUBLOK - interface A
  14   20         RDEVCUB  DS    1F             Pointer to RCUBLOK - interface B
  18   24         RDEVQUED DS    1D             IOBLOK queued time - TOD clock units
  20   32         RDEVIOCT DS    1F             Device I/O count
  24   36         RDEVAIOB DS    1F             Active IOBLOK
  28   40         RDEVUSER DS    1F             Pointer to VMBLOK of dedicated user
  2C   44         RDEVATT  DS    1H             Attached virtual address
  2E   46         RDEVCYL  DS    1H             DASD - current cylinder location
| 30   48         RDEVSER  DS    CL6            Device volume serial number
  36   54         RDEVLNKS DS    1H             DASD - number of links to this disk
  38   56         RDEVTCTL DS    8X             Terminal control bytes
  40   64         RDEVTMAT DS    1F             Device attached time - TOD clock word 0
| 44   68         RDEVRSV1 DS    1X             R*5 - Reserved for IBM use
| 45   69         RDEVSTA2 DS    1X             R*6 - Device status (2nd byte)
|                 Bits Defined in RDEVSTA2
|                 RDEVRACT EQU   X'80'          Active device is being reset
|                 RDEVBUCH EQU   X'40'          Device is busy with the channel

| 46   70         RDEVMDL  DS    1X             R*7 - device model number
| 47   71         RDEVFTR  DS    1X             R*8 - device feature code
  48   72         RDEVIOER DS    1F             Pointer to IOERBLOK for last CP error
  4C   76         RDEVCTRS DS    1F             Pointer to error counter control blok

                  RDEVSIZE EQU   (*-RDEVBLOK)/8 RDEVBLOK size in doublewords (X'0A')

                  For CP owned devices
                           ORG   RDEVUSER
  28   40         RDEVALLN DS    1F             Anchor for ALOCBLOK chain for this device
  2C   44         RDEVCODE DS    1H             Device code - SYSOWNED index
                           ORG   RDEVTCTL
  38   56         RDEVPAGE DS    1F             Anchor for RECBLOK chain for paging
  3C   60         RDEVRECS DS    1F             Anchor for RECBLOK chain for spooling
```

```
   40   64            RDEVPNT  DS    1F             Pointer to next RDEVBLOK for allocation

|                     For slotted 2301 paging devices
|                              ORG   RDEVRECS
|  3C   60            RDEVDCTL DS    1F             Pointer to DRUMTABL control block

                      For spooling unit record devices
                               ORG   RDEVQUED
   18   24            RDEVSPL  DS    1F             Pointer to active RSPLCTL block
   1C   28            RDEVCLAS DS    4C             Device class(es)

                      For terminal devices
                               ORG   RDEVQUED
   18   24            RDEVCON  DS    1F             Pointer to CONTASK list
   1C   28            RDEVAIRA DS    1F             Attention interrupt return address
                               ORG   RDEVTCTL
   38   56            RDEVLEND DS    1C             Device line end symbol
   39   57            RDEVLDEL DS    1C             Device line delete symbol
   3A   58            RDEVCDEL DS    1C             Device character delete symbol
   3B   59            RDEVESCP DS    1C             Device character escape symbol
   3C   60            RDEVLLEN DS    1X             Device line length
   3D   61            RDEVATUC DS    1X             Device attention count
   3E   62            RDEVTFLG DS    1X             Additional terminal flags
                      Bits Defined in RDEVTFLG
                      RDEVATIN EQU   X'80'          Attention signalled on input
                      RDEVREST EQU   X'40'          Terminal in process of being reset
                      RDEVATOF EQU   X'20'          Do not type exclamation point or CR
                      RDEVCIRD EQU   X'10'          Write a circle D to a terminal

   3F   63            RDEVRSV3 DS    1X             Reserved for IBM use

                               ORG   RDEVMDL
|  46   70            RDEVTMCD DS    1X             Terminal code
|                     Bits Defined in RDEVTMCD
                      RDEVPTTC EQU   X'00'          PTTC/EBCD
                      RDEVCORR EQU   X'04'          Correspondence
                      RDEVAPLP EQU   X'08'          APL PTTC/EBCD
                      RDEVAPLC EQU   X'0C'          APL Correspondence
                      RDEVUSC8 EQU   X'10'          UASCII-8 level

   47   71            RDEVSADN DS    1X             Terminal set-address number
```

RECBLOK - DASD PAGE (SLOT) ALLOCATION BLOCK

```
    ┌────────────────────────────────────────────┐
  0 |        RECPNT        | RECCYL  |R*1 |R*2 |
    |────────────────────────────────────────────|
  8 |                  RECMAP                    |
    └────────────────────────────────────────────┘
```

Displacement Hex	Dec	Field Name			Field Description, Contents, Meaning
0	0	RECPNT	DS	1F	Pointer to next RECBLOK on chain
4	4	RECCYL	DS	1H	Cylinder address for pages in this block
6	6	RECUSED	DS	1X	R*1 - Number of pages currently in use
7	7	RECMAX	DS	1X	R*2 - Maximum number of pages available
8	8	RECMAP	DS	1D	Page allocation bit map

Bits Defined in RECMAP
- 0 - Page is available
- 1 - Page has been assigned

RECSIZE EQU (*-RECBLOK)/8 RECBLOK size in doublewords (X'02')

__Note__: Although the size of RECMAP is fixed, the maximum number of pages available on a cylinder is device dependent. For any pages that are not physically present on a cylinder, their corresponding bits are set to one.

RSPLCTL - REAL SPOOL CONTROL BLOCK

```
    ┌──────────────────────────────────────┐
  0 │    RSPRSTRT        │    RSPDPAGE      │
    ├──────────────────────────────────────┤
  8 │    RSPVPAGE        │    RSPRPAGE      │
    ├──────────────────────────────────────┤
 10 │    RSPMISC         │    RSPSFBLK      │
    └──────────────────────────────────────┘
```

Displacement Hex	Dec	Field Name			Field Description, Contents, Meaning
0	0	RSPRSTRT	DS	1F	Restart CAW - CCW address
4	4	RSPDPAGE	DS	1F	DASD location (DCHR) of current page buffer
8	8	RSPVPAGE	DS	1F	Virtual address of page buffer
C	12	RSPRPAGE	DS	1F	Real address of page buffer
10	16	RSPMISC	DS	1F	Use varies according to caller
14	20	RSPSFBLK	DS	1F	Pointer to SFBLOK for file
		RSPSIZE	EQU	(*-RSPLCTL)/8	Size in doublewords (X'03')

SAVEAREA

0	SAVERETN	SAVER12
8	SAVER13	SAVEWRK1
10	SAVEREGS	
40	SAVEWRK2	SAVEWRK3
48	SAVEWRK4	SAVEWRK5
50	SAVEWRK6	SAVEWRK7
58	SAVEWRK8	SAVEWRK9

Displacement Hex	 Dec	Field Name			Field Description, Contents, Meaning
0	0	SAVERETN	DS	1F	Active SAVEAREA (caller's return address)
			ORG	SAVERETN	
0	0	SAVENEXT	DS	1F	Inactive SAVEAREA (next SAVEAREA address)
4	4	SAVER12	DS	1F	Caller's base (R12)
8	8	SAVER13	DS	1F	Caller's SAVEAREA (R13)
C	12	SAVEWRK1	DS	1F	Callee's workarea
10	16	SAVEREGS	DS	12F	Caller's registers (R0 TO R11)
			ORG	SAVEREGS	
10	16	SAVER0	DS	1F	
14	20	SAVER1	DS	1F	
18	24	SAVER2	DS	1F	
1C	28	SAVER3	DS	1F	
20	32	SAVER4	DS	1F	
24	36	SAVER5	DS	1F	
28	40	SAVER6	DS	1F	
2C	44	SAVER7	DS	1F	
30	48	SAVER8	DS	1F	
34	52	SAVER9	DS	1F	
38	56	SAVER10	DS	1F	
3C	60	SAVER11	DS	1F	
40	64	SAVEWRK2	DS	1F	Callee's workarea (8 words)
44	68	SAVEWRK3	DS	1F	
48	72	SAVEWRK4	DS	1F	
4C	76	SAVEWRK5	DS	1F	

```
54   84          SAVEWRK7 DS     1F
58   88          SAVEWRK8 DS     1F
5C   92          SAVEWRK9 DS     1F

                 SAVESIZE EQU    (*-SAVEAREA)/8 size in doublewords (X'0C')
```

SAVTABLE - FIRST PAGE ON SAVED SYSTEM DASD

```
    ┌────────────────────────────────────────┐
  0 |                SAVPSW                  |
    |────────────────────────────────────────|
  8 |                SAVGREGS                |
    |────────────────────────────────────────|
 48 |                SAVFPRES                |
    |────────────────────────────────────────|
 68 |                SAVCREGS                |
    |────────────────────────────────────────|
 A8 |                SAVKEYS                 |
    └────────────────────────────────────────┘
```

Displacement Hex	 Dec	Field Name			Field Description, Contents, Meaning
0	0	SAVPSW	DS	1D	PSW of virtual machine at SAVSYS table
8	8	SAVGREGS	DS	16F	General registers
48	72	SAVFPRES	DS	4D	Floating-point registers
68	104	SAVCREGS	DS	16F	Control registers
A8	168	SAVKEYS	DS	1H	Two byte entry for each saved page containing storage keys for each page

SEGTABLE - SEGMENT TABLE

```
    ┌────────────────────────────────────────┐
  0 | S*1                                    |
    | SEGPAGE  (variable length)             |
    |                                        |
    └────────────────────────────────────────┘
```

Displacement Hex	 Dec	Field Name			Field Description, Contents, Meaning
0	0	SEGPAGE	DS	1F	Pointer to page table - PAGTABLE
		Page table length			
			ORG	SEGPAGE	
0	0	SEGPLEN	DS	1X	S*1 - page table length (pages - 1) (in left half of byte)

SFBLOK - SPOOL FILE BLOCK

```
   ┌─────────────────────────────────────────┐
 0 |    SFBPNT          |    SFBSTART        |
   |─────────────────────────────────────────|
 8 |               SFBUSER                   |
   |─────────────────────────────────────────|
10 |               SFBORIG                   |
   |─────────────────────────────────────────|
18 |    SFBRECNO        |SFBRECSZ |SFBFILID  |
   |─────────────────────────────────────────|
20 |S*1 |S*2 |SFBMISC1 |    SFBRECS          |
   |─────────────────────────────────────────|
28 |               SFBFNAME                  |
   |─────────────────────────────────────────|
34 |               SFBFTYPE                  |
   |─────────────────────────────────────────|
40 |               SFBDATE                   |
   |─────────────────────────────────────────|
48 |               SFBTIME                   |
   |─────────────────────────────────────────|
50 |    SFBLAST         |SFBCOPY  |S*3 |S*4  |
   |─────────────────────────────────────────|
58 |                SFBDIST                  |
   └─────────────────────────────────────────┘
```

Displacement Hex	 Dec	Field Name			Field Description, Contents, Meaning
0	0	SFBPNT	DS	1F	Pointer to next SFBLOK
4	4	SFBSTART	DS	1F	DASD location (DCHR) of last page buffer
8	8	SFBUSER	DS	CL8	VMUSER identification of file owner
10	16	SFBORIG	DS	CL8	VMUSER identification of file origin
18	24	SFBRECNO	DS	1F	Number of data records in file
1C	28	SFBRECSZ	DS	1H	Logical record size - excluding CCW's
1E	30	SFBFILID	DS	1H	Binary system file number
20	32	SFBFLAG	DS	1X	S*1 - SFBLOK control bits
		Bits Defined in SFBFLAG			
		SFBINUSE	EQU	X'80'	File being processed
		SFBRECOK	EQU	X'40'	Allocation records complete
		SFBUHOLD	EQU	X'20'	File in user hold status
		SFBDUMP	EQU	X'10'	File is a CP system dump
		SFBOPEN	EQU	X'08'	Input file has been opened
		SFBSHOLD	EQU	X'04'	File in system hold status
		SFBEOF	EQU	X'02'	Input file has reached EOF
		SFBRECER	EQU	X'01'	SFBREC chain incomplete
21	33	SFBTYPE	DS	1X	S*2 - device type for output

X'41' = printer

```
|  22   34          SFBMISC1 DS    1H             Use varies according to caller       0
   24   36          SFBRECS  DS    1F             Pointer to RECBLOKS for active file  0
   28   40          SFBFNAME DS    CL12           File name
   34   52          SFBFTYPE DS    CL12           File type
   40   64          SFBDATE  DS    CL8            Creation date of spool file  MM/DD/YY
   48   72          SFBTIME  DS    CL8            Creation time of spool file  HH:MM:SS
   50   80          SFBLAST  DS    1F             DASD location (DCHR) of last page buffer   [illegible]
   54   84          SFBCOPY  DS    1H             Number of copies requested
   56   86          SFBCLAS  DS    1C             S*3 - Spool output class
   57   87          SFBFLAG2 DS    1X             S*4 - SFBLOK flag byte two
                    Bits Defined in SFBFLAG2
                    SFBHOLD  EQU   X'80'          Save input file, or hold output file
                    SFBNOHID EQU   X'40'          Delete input file, or do not hold ouput file
                    SFBHOLD and SFBNHOLD          Override options in VDEVBLOK
                    SFBREQUE EQU   X'20'          Re-queue spool file
                    SFBRSTRT EQU   X'10'          Restart in progress
                    SFBTICER EQU   X'08'          Buffer TIC error
|                   SFBPURGE EQU   X'04'          Purge open spool file

   58   88          SFBDIST  DS    CL8            Distribution code

                    SFBSIZE  EQU   (*-SFBLOK)/8   Size in doublewords (X'0C')
```

SHQBLCK - SPOOL HOLD QUEUE BLOCK

```
           ----------------------------------------
        0 |       SHQPNT       | V*1| V*2| SPARE   |
          |--------------------------------------- |
        8 |                SHQUSER                 |
           ----------------------------------------
```

```
Displacement        Field
Hex  Dec            Name                          Field Description, Contents, Meaning
------------        -----                         ------------------------------------
  0    0            SHQPNT   DS    1F             Address of next SHQBLOK
  4    4            SHQFLAGS DS    0CL4           Length
  4    4            SHQUHOLD DS    1X             V*1 user 'USER HOLD' flag byte
  5    5            SHQSHOLD DS    1X             V*2 user 'SYSTEM HOLD' flag byte
  6    6            SHQSPARE DS    2X             Spare
  8    8            SHQUSER  DS    CL8            VMUSER identification of file owner

                    SHQBSIZE EQU   (*-SHQBLOK)/8  Size in doublewords (X'02')

                    Bits Defined in SHQUHOLD and SHQSHOLD
                     TYPPRT  is used for printer type. (See Appendix C for DEVTYPES)
                     TYPPUN  is used for punch type. (See Appendix C for DEVTYPES)
```

SHRTABLE - NAMED-SHARED SEGMENT SYSTEMS

```
   ┌──────────────────────────────────────────┐
 0 |      SHRFPNT        |      SHRBPNT        |
   |──────────────────────────────────────────|
 8 |                 SHRNAME                  |
   |──────────────────────────────────────────|
10 | SHRTSIZE| SHRUSECT|     SHRSEGCT        |
   |──────────────────────────────────────────|
18 |      SHRSEGNM       |     SHRPAGE         |
   └──────────────────────────────────────────┘
```

Displacement Hex	Dec	Field Name			Field Description, Contents, Meaning
0	0	SHRFPNT	DS	1F	Pointer to next SHRTABLE
4	4	SHRBPNT	DS	1F	Pointer to previous SHRTABLE
8	8	SHRNAME	DS	CL8	Name of saved system
10	16	SHRTSIZE	DS	1H	Size of SHRTABLE in doublewords
12	18	SHRUSECT	DS	1H	Number of users IPLed to this name
14	20	SHRSEGCT	DS	1F	Number of shared segments
18	24	SHRSEGNM	DS	1F	Contains shared segment numbers. Up to four segment numbers per word.
1C	28	SHRPAGE	DS	1F	Pointers to each of the shared SEGTABLES. There is one word for each shared segment. The entry is the same as S*1 SEGPAGE in the SEGTABLE.

SPLINK - SPOOL PAGE BUFFER LINKAGE

```
   ┌──────────────────────────────────────────┐
 0 |     SPNXTPAG        |     SPPREPAG        |
   |──────────────────────────────────────────|
 8 |     SPRMISC         |     SPRECNUM        |
   |──────────────────────────────────────────|
10 |          Spool Buffer Data               |
   └──────────────────────────────────────────┘
```

Displacement Hex	Dec	Field Name			Field Description, Contents, Meaning
0	0	SPNXTPAG	DS	1F	DASD location (DCHR) of next page buffer
4	4	SPPREPAG	DS	1F	DASD location (DCHR) of previous page buffer
8	8	SPRMISC	DS	1F	Use varies according to caller
C	12	SPRECNUM	DS	1F	Number of data records in buffer
		SPSIZE	EQU	(*-SPLINK)	Size in bytes (X'10')

SWPTABLE - SWAP TABLE FOR VIRTUAL MACHINE PAGING

```
  ┌────────────────────────────────────────┐
0 |    SWPVM          |    SWPPAG          |
  |────────────────────────────────────────|
8 |S*1 |S*2 |S*3 |S*4 | SWPCYL   |S*6 |S*7 |
  └────────────────────────────────────────┘
```

Displacement		Field			
Hex	Dec	Name			Field Descripticn, Contents, Meaning
0	0	SWPVM	DS	1F	Pcinter to VMBLOK
4	4	SWPPAG	DS	1F	Pointer to PAGTABLE
8	8	SWPFLAG	DS	1X	S*1 - SWPTABLE flag bits
		Bits Defined in SWPFLAG			
		SWPTRANS	EQU	X'80'	Page in transit
		SWPRECMP	EQU	X'40'	Page permanently assigned
		SWPALLCC	EQU	X'20'	Page enqueued for allocation
		SWPSHR	EQU	X'10'	Page shared
		SWPREF1	EQU	X'08'	First half page referenced
		SWPCHG1	EQU	X'04'	First half page changed
		SWPREF2	EQU	X'02'	Seccnd half page referenced
		SWPCHG2	EQU	X'01'	Second half page changed
9	9	SWPVPAGE	DS	1X	S*2 - virtual page number
A	10	SWPKEY1	DS	1X	S*3 - virtual storage key
B	11	SWPKEY2	DS	1X	S*4 - virtual storage key
C	12	SWPCYL	DS	1H	DASD cylinder address
E	14	SWPDPAGE	DS	1X	S*6 - Page number on cylinder
F	15	SWPCODE	DS	1X	S*7 - RDEVBLOK device code

Note: For each SWPTABLE there is only one doubleword that consists of SWPVM and SWPPAG followed by 16 entries (one for each PAGTABLE entry) that consist of S*1, S*2, S*3, S*4, SWPCYL, S*6 and S*7. Thus, the total size of the SWPTABLE is 17 doublewords.

SYSLOCS - SYSTEM LOW STORAGE INFORMATION

```
    ┌───────────────────────────────────────────────┐
  0 |       DMKSYSDT      |      DMKSYSTM           |
    |-----------------------------------------------|
 10 |            DMKSYSLW            | DMKSYSLG     |
    |-----------------------------------------------|
 20 |DMKSYSNM  |DMKSYSMA|DMKSYSMU| DMKSYSND        |
    |-----------------------------------------------|
 30 |DMKSYSLB  |DMKSYSUD|DMKSYSPL|                  |
    |-----------------------------------------------|
 40 |            DMKSYSDW            |# |¢ |@ |"   |
    |-----------------------------------------------|
 50 | S*1 |               |DMKSYSCK|                |
    └───────────────────────────────────────────────┘
```

Displacement Hex	Dec	Field Name			Field Description, Contents, Meaning
0	0	DMKSYSDT	DC	CL8'MM/DD/YY'	Date of system log message
8	8	DMKSYSTM	DC	CL8'HH:MM:SS'	Time of system log message
10	16	DMKSYSLW	DC	X'00',X'00',CL10' '	Weekday of system log message
1C	28	DMKSYSLG	DC	A(0)	Pointer to first log message block
20	32	DMKSYSNM	DC	F'0'	Current number of users on the system
24	36	DMKSYSMA	DC	F'0'	Maximum number of users allowed on
28	40	DMKSYSMU	DC	F'0'	Maximum number of users on the system
2C	44	DMKSYSND	DC	F'0'	Number of dialed users on the system
30	48	DMKSYSLB	DC	A(0)	Pointer to user directory lock block
34	52	DMKSYSUD	DC	A(0)	Pointer to start of user directory on SYSRES
38	56	DMKSYSPL	DC	A(0)	Pointer to a list of virtual page buffers
3C	60		DC	A(0)	Reserved for IBM use
40	64	DMKSYSDW	DC	X'00',X'00',CL10' '	Day-of-week in Hex. and EBCDIC
4C	76	DMKSYSLE	DC	X'7B'	# default line-end (pound-sign)
4D	77	DMKSYSLD	DC	X'4A'	¢ default line-delete (cent-sign)
4E	78	DMKSYSCD	DC	X'7C'	@ default character-delete (at-sign)
4F	79	DMKSYSES	DC	X'7F'	" default edit escape (double-quote-mark)
50	80	DMKSYSLL	DC	AL1(130,129,72)	S*1 Default line lengths for 3210 & 3215 - 2741 & 1050 - TTY terminals
53	83		DC	XL5'0'	Reserved for IBM use
58	88	DMKSYSCK	DC	D'0'	Time-of-day clock value last stored by accounting, DUMP or machine check

SYSTABLE - NAMED SYSTEM TABLE

```
     ----------------------------------------
  0 |       SYSPNT       |      SYSSIZE      |
    |----------------------------------------|
  8 |               SYSNAME                  |
    |----------------------------------------|
 10 |          VSYSRES            | SYSVADDR |
    |----------------------------------------|
 18 |          SYSVOL             | SYSCYL   |
    |----------------------------------------|
 20 |     SYSSTART       |      SYSPAGCT     |
    |----------------------------------------|
 28 | SYSPAGLN|        SYSPAGNM              |
    |---------                               |
    |----------------------------------------|
 30 | SYSSEGLN|        SYSHRSEG              |
    |---------                               |
    |                                        |
     ----------------------------------------
```

Displacement		Field			
Hex	Dec	Name			Field Description, Contents, Meaning
0	0	SYSPNT	DS	1F	Chain pointer to next entry
4	4	SYSSIZE	DS	1F	Minimum storage size needed to run system
8	8	SYSNAME	DS	CL8	System name
10	16	VSYSRES	DS	CL6	Volume serial of DASD containing user's system
16	22	SYSVADDR	DS	1H	Virtual address of VSYSRES
18	24	SYSVOL	DS	CL6	Volume serial of DASD containing saved pages
1E	30	SYSCYL	DS	1H	Cylinder on VSYSRES of user's system same as VDEVRELN
20	32	SYSSTART	DS	1F	CCPD of first page on SYSVOL
24	36	SYSPAGCT	DS	1F	Total number of pages saved
28	40	SYSPAGLN	DS	1H	Number of entries in SYSPAGNM
2C	44	SYSPAGNM	DS	1F	One full word entry for each range of pages to be saved
30	48	SYSSEGLN	DS	1H	Numbers of entries in SYSHRSEG
32	50	SYSHRSEG	DS	1X	One byte for each segment to be shared

TREXT - VIRTUAL MACHINE TRACING EXTENSION TO VMBLOK

```
               ┌─────────────────────────────────────────────┐
             0 │      TREXIN1          │       TREXIN2       │
               │─────────────────────────────────────────────│
|            8 │ TREXSVC1 │ TREXSVC2   │ S*1 │ S*2 │ TREXLOCK │
               │─────────────────────────────────────────────│
            10 │      TREXPERA         │ TREXPERC  │ TREXLCNT │
               │─────────────────────────────────────────────│
            18 │      TREXANSI         │       TREXCR9       │
               │─────────────────────────────────────────────│
            20 │      TREXCR10         │       TREXCR11      │
               │─────────────────────────────────────────────│
            28 │                 TREXBUFF                    │
               └─────────────────────────────────────────────┘
```

```
  Displacement       Field
  Hex  Dec           Name                         Field Descripticn, Contents, Meaning
  ------------       -----                        ---------------------------------------
   0     0           TREXIN1  DS     1F           First address - replaced instruction
   4     4           TREXIN2  DS     1F           Second address - replaced instruction
   8     8           TREXSVC1 DS     1H           Displaced halfword - instruction one
   A    10           TREXSVC2 DS     1H           Displaced halfword - instruction two

                              ORG    TREXIN1
   0     0           TREXPSW  DS     1D           Old PSW for pending SVC interrupt
   8     8           TREXINTL DS     1H           Instruction length code
   A    10           TREXINTC DS     1H           Interruption code for pending interrupt
   C    12           TREXFLAG DS     1X           S*1 - tracing control flags
                     Bits Defined in TREXFLAG
                     TREXRUN  EQU    X'80'        Prevent CFWAIT between events
|                    TREXVAT  EQU    X'40'        Call DMKVATRN to put back virtual instruction

   D    13           TREXOUT  DS     1X           S*2 - trace output controls
                     Bits Defined in TREXOUT
                     TREXPRT  EQU    X'80'        Output to the virtual printer
                     TREXCON  EQU    X'40'        Output to user terminal

|  E    14           TREXLOCK DS     1H           Indicates tracing when set
| 10    16           TREXPERA DS     1F           PER event address cn interrupt
| 14    20           TREXPERC DS     1H           PER code bits from hardware event
  16    22           TREXLCNT DS     1H           Printed output line count
  18    24           TREXANSI DS     1A           Address of next (or last) sequential instruction
| 1C    28           TREXCR9  DS     0F           Shadow control registers for PER trace
|                    TREXPER  DS     XL2            PER control field
| 1E    30           TREXPREG DS     1H             PER register mask field
  20    32           TREXCR10 DS     1F           Address range start value
| 24    36           TREXCR11 DS     1F           Address range ending value
```

```
   28   40        TREXBUFF DS    10D             Console/printer output buffer (80 bytes)

                  TREXSIZE EQU   (*-TREXT)/8     TREXT size in doublewords (X'0F')

                           ORG   TREXPERA        Re-Definition for TRACE use
   10   16        TREXNSI  DS    6X              Actual next (or last) sequential instruction

|                          ORG   TREXCR9         Re-definition for TRACE use
   1C   28        TREXCTL  DS    0H              Halfword holding tracing control bits:
                  TREXCTL1 DS    1X              First byte = same as VMTRCTL in VMBLOK
   1D   29        TREXCTL2 DS    1X              Second byte = remaining control bits
                  Bits Defined in TREXCTL2:
                  TREXCCW  EQU   X'80'           TRACE virtual and real CCWs
                  TREXCSW  EQU   X'40'           TRACE virtual and real CSWs
                  TREXBRAN EQU   X'20'           TRACE successful branches
                  TREXINST EQU   X'10'           TRACE all instructions

   1E   30        TREXPRNT DS    1H              Printer flagbits corresponding to TREXCTL
   20   32        TREXTERM DS    1H              Terminal flagbits corresponding to TREXCTL
   22   43        TREXRUNF DS    1H              Run/norun flagbits corresponding to TREXCTL
|  24   45        TREXPNTR DS    1F              Pointer to 1st stacked TRACE request, if any
```

TRQBLOK - TIMER REQUEST BLOCK

```
        ┌────────────────────────────────────────┐
     0  |                TRQBVAL                 |
        |────────────────────────────────────────|
     8  |   TRQBFPNT          |      TRQBBPNT    |
        |────────────────────────────────────────|
    10  |                TRQBTOD                 |
        |────────────────────────────────────────|
    18  |   TRQBUSER          |      TRQBIRA     |
        └────────────────────────────────────────┘
```

Displacement Hex	Dec	Field Name			Field Description, Contents, Meaning
0	0	TRQBVAL	DS	1D	TCD clock comparator value for interrupt
8	8	TRQBFPNT	DS	1F	Pointer to next TRQBLOK
C	12	TRQBBPNT	DS	1F	Pointer to previous TRQBLOK
10	16	TRQBTOD	DS	1D	TCD clock value when TRQBLOK is queued
18	24	TRQBUSER	DS	1F	Address of VMBLOK for user
1C	28	TRQBIRA	DS	1F	Interrupt return address
		TRQBSIZE	EQU	(*-TRQBLOK)/8	Size in doublewords (X'04')

UDBFBLOK - USER DIRECTORY BUFFER BLOCK

```
      ┌─────────────────────────────────────────┐
    0 │                UDBFWORK                  │
      ├─────────────────────────────────────────┤
   30 │    UDBFVADD        │     UDBFDASD        │
      └─────────────────────────────────────────┘
```

Displacement		Field			
Hex	Dec	Name			Field Description, Contents, Meaning
0	0	UDBFWORK	DS	6D	Buffer work space used by the caller
30	48	UDBFVADD	DS	1F	Virtual address of the last directory page
34	52	UDBFDASD	DS	1F	DASD address of the last directory page
		UDBFSIZE	EQU	(*-UDBFBLOK)/8	UDBFBLOK size in doublewords (X'07')

UDEVBLOK - USER DEVICE BLOCK

```
      ┌─────────────────────────────────────────┐
    0 │UDEVADD  │UDEVDISP │      UDEVDASD        │
      ├─────────────────────────────────────────┤
    8 │U*1 │U*2 │U*3 │U*4 │U*5 │U*6 │UDEVNCYL    │
      ├─────────────────────────────────────────┤
   10 │UDEVRELN │           UDEVVSER             │
      ├─────────────────────────────────────────┤
   18 │                UDEVPASR                  │
      ├─────────────────────────────────────────┤
   20 │                UDEVPASW                  │
      ├─────────────────────────────────────────┤
   28 │                UDEVPASM                  │
      └─────────────────────────────────────────┘
```

Displacement		Field			
Hex	Dec	Name			Field Description, Contents, Meaning
0	0	UDEVADD	DS	1H	Virtual device address
2	2	UDEVDISP	DS	1H	Displacement of the next UDEVBLOK
4	4	UDEVDASD	DS	1F	DASD address of the next UDEVBLOK
8	8	UDEVSTAT	DS	1X	U*1 - status information
		Bits Defined in UDEVSTAT			
		UDEVDED	EQU	X'80'	Device to be dedicated to this user
		UDEVTDSK	EQU	X'40'	TDISK to be allocated
		UDEVLONG	EQU	X'20'	Device block is full length (6 doublewords)
		UDEVLKDV	EQU	X'10'	Device is to be linked (at LOGON time)
		UDEVSPOO	EQU	X'08'	Device is a spool device

9	9	UDEVMODE	DS	1X	U*2 – access mode information
		Bits Defined in UDEVMODE			
		UDEVLR	EQU	X'80'	Read links allowed
		UDEVLW	EQU	X'40'	Write links allowed
		UDEVLM	EQU	X'20'	Multiple–write links allowed
		UDEVR	EQU	0	Device to be in R link mode for owner
		UDEVRR	EQU	4	Device to be in RR link mode for owner
		UDEVW	EQU	8	Device to be in W link mode for owner
		UDEVWR	EQU	12	Device to be in WR link mode for owner
		UDEVM	EQU	16	Device to be in M link mode for owner
		UDEVMR	EQU	20	Device to be in MR link mode for owner
		UDEVMW	EQU	24	Device to be in MW link mode for owner
A	10	UDEVTYPC	DS	1C	U*3 – device class
B	11	UDEVTYPE	DS	1C	U*4 – device type
C	12	UDEVFTR	DS	1C	U*5 – device feature mode
D	13	UDEVMDL	DS	1C	U*6 – device model number
E	14	UDEVNCYL	DS	1H	Virtual DASD size
10	16	UDEVRELN	DS	1H	Virtual DASD cylinder relocation
12	18	UDEVVSER	DS	6C	Volume serial number
18	24	UDEVPASR	DS	1D	Password for read access
20	32	UDEVPASW	DS	1D	Password for write access
28	40	UDEVPASM	DS	1D	Password for multiple access
		UDEVSIZE	EQU	(*–UDEVBLOK)/8	UDEVBLOK size in doublewords
			ORG	UDEVMDL	
D	13	UDEVCLAS	DS	1C	User device block (Short)
E	14	UDEVLINK	DS	1H	U*6 – unit spool output class
10	16	UDEVLKID	DS	1D	User link to USERID

UDIRBLOK – USER DIRECTORY BLOCK

```
    ┌─────────────────────────────────────────┐
 0  |UDIRRSV1  |UDIRDISP |      UDIRDASD       |
    |─────────────────────────────────────────|
 8  |                UDIRUSER                 |
    |─────────────────────────────────────────|
10  |                UDIRPASS                 |
    └─────────────────────────────────────────┘
```

Displacement Hex	Dec	Field Name			Field Description, Contents, Meaning
0	0	UDIRRSV1	DS	1H	Reserved for IBM use
2	2	UDIRDISP	DS	1H	Displacement of the user UMACBLOK

4	4	UDIRDASD DS	1F	DASD address of the user UMACBLOK
8	8	UDIRUSER DS	1D	USERID
10	16	UDIRPASS DS	1D	User password
		UDIRSIZE EQU	(*-UDIRBLOK)/8	UDIRBLOK size in doublewords (X'03')

UDLKBLOK - USER DIRECTORY LOCK BLOCK

```
    ┌─────────────────────────────────────┐
  0 |      UDLKNEXT      |     UDLKRSV1   |
    |─────────────────────────────────────|
  8 |              UDLKNAME               |
    └─────────────────────────────────────┘
```

Displacement Hex	 Dec	Field Name		 Field Description, Contents, Meaning
0	0	UDLKNEXT DS	1F	Pointer to the next lock block
4	4	UDLKRSV1 DS	1F	Reserved for IBM use
8	8	UDLKNAME DS	1D	The name locked
		UDLKSIZE EQU	(*-UDLKBLOK)/8	UDLKBLOK size in doublewords (X'02')

UMACBLOK - USER MACHINE BLOCK

```
     ┌───────────────────────────────────────┐
   0 |UMACDVCT |UMACDISP |     UMACDASD      |
     |───────────────────────────────────────|
   8 |U*1 |U*2 |U*3 |U*4 |U*5 |U*6 |U*7 |U*8 |
     |───────────────────────────────────────|
  10 |   UMACCORE        |     UMACMCOR      |
     |───────────────────────────────────────|
  18 |               UMACACCT                |
     |───────────────────────────────────────|
  20 |               UMACDIST                |
     |───────────────────────────────────────|
  28 |               UMACIPL                 |
     └───────────────────────────────────────┘
```

Displacement Hex	 Dec	Field Name		 Field Description, Contents, Meaning
0	0	UMACDVCT DS	1H	Number of devices
2	2	UMACDISP DS	1H	Displacement of the user's first UDEVBLOK

```
  4    4          UMACDASD DS    1F              DASD address of the user's first UDEVBLOK
  8    8          UMACCLEV DS    1C              U*1 - command level
                  Bits Defined in UMACCLEV
                  UMACCLA  EQU   X'80'           Class A functions
                  UMACCLB  EQU   X'40'           Class B functions
                  UMACCLC  EQU   X'20'           Class C functions
                  UMACCLD  EQU   X'10'           Class D functions
                  UMACCLE  EQU   X'08'           Class E functions
                  UMACCLF  EQU   X'04'           Class F functions
                  UMACCLG  EQU   X'02'           Class G functions
                  UMACCLH  EQU   X'01'           Class H functions

  9    9          UMACPRIR DS    1X              U*2 - priority
  A   10          UMACOPT  DS    1X              U*3 - virtual machine options
                  Bits Defined in UMACOPT
                  UMACISAM EQU   X'80'           ISAM CCW checking option
                  UMACECOP EQU   X'40'           Extended control mode option
                  UMACRT   EQU   X'20'           Real timer option
                  UMACVROP EQU   X'10'           Virtual = Real storage option
                  UMACACC  EQU   X'08'           Accounting card option

  B   11          UMACRSV1 DS    1C              U*4 - Reserved for IBM use
  C   12          UMACLEND DS    1C              U*5 - Terminal line end symbol
  D   13          UMACLDEL DS    1C              U*6 - Terminal line delete symbol
  E   14          UMACCDEL DS    1C              U*7 - Terminal character delete symbol
  F   15          UMACES   DS    1C              U*8 - Edit escape symbol
 10   16          UMACCORE DS    1F              Virtual storage size in bytes
 14   20          UMACMCOR DS    1F              Maximum virtual storage size in bytes
 18   24          UMACACCT DS    1D              Accounting information
 20   32          UMACDIST DS    1D              User machine distribution information
 28   40          UMACIPL  DS    1D              Name of system to be IPLed at LOGON time

                  UMACSIZE EQU   (*-UMACBLOK)/8 UMACBLOK size in doublewords (X'06')
```

VCHBLOK - VIRTUAL CHANNEL BLOCK

```
   ┌───────────────────────────────────────────┐
 0 |VCHADD   |VCHCUINT |VCHCEDEV |V*1 |V*2 |
   |───────────────────────────────────────────|
 8 |                  VCHCUTBL                 |
   └───────────────────────────────────────────┘
```

```
Displacement      Field
Hex  Dec          Name                                Field Descripticn, Contents, Meaning
------------      -----                               ---------------------------------------
  0    0          VCHADD   DS     1H                  Virtual channel address
  2    2          VCHCUINT DS     1H                  VCUBLOCK with interrupt - bit map
  4    4          VCHCEDEV DS     1H                  Virtual device address with channel class interrupt
  6    6          VCHSTAT  DS     1X                  V*1 - virtual channel status
                  Bits Defined in VCHSTAT
                  VCHBUSY  EQU    X'80'               Virtual channel busy
                  VCHCEPND EQU    X'40'               Virtual channel class interrupt pending
|                 VCHDED   EQU    X'01'               Virtual channel dedicated

  7    7          VCHTYPE  DS     1X                  V*2 - Virtual channel type
                  Bits Defined in VCHTYPE
                  VCHSEL   EQU    X'80'               Virtual selector channel
                  VCHBMX   EQU    X'40'               Virtual block multiplexor

  8    8          VCHCUTBL DS     16H                 Contrcl units attached - VMCUSTRT index

                  VCHSIZE  EQU    (*-VCHBLOK)/8       VCHBLOK size in doublewords (X'05')
```

VCONCTL - VIRTUAL CONSOLE CONTROL BLOCK

```
     ┌─────────────────────────────────────────┐
   0 │     VCONCAW         │     VCONBUF       │
     │─────────────────────────────────────────│
   8 │                 VCONCCW                 │
     │─────────────────────────────────────────│
  10 │V*1 │V*2 │V*3 │V*4 │     VCONIDAP        │
     └─────────────────────────────────────────┘
```

Displacement Hex	Dec	Field Name			Field Description, Contents, Meaning
0	0	VCONCAW	DS	1F	Virtual address of user CCW
4	4	VCONBUF	DS	1F	Pointer to data buffer
8	8	VCONCCW	DS	1D	Current user CCW
10	16	VCONRSV1	DS	1X	V*1 - Reserved for IBM use
11	17	VCONBFSZ	DS	1X	V*2 - Data buffer size in doublewords
12	18	VCONRSV2	DS	1X	V*3 - reserved for IBM use
13	19	VCONRSV3	DS	1X	V*4 - reserved for IBM use
14	20	VCONIDAP	DS	1F	For IDA pointer to current IDAW
		VCONSIZE	EQU	(*-VCONCTL)/8	VCONCTL size in doublewords (X'03')
			ORG	VCONCCW	
8	8	VCONADDR	DS	1F	CCW data address
C	12	VCONFLAG	DS	1X	CCW flag bits
D	13	VCONRSV4	DS	1X	Reserved for IBM use
E	14	VCONCNT	DS	1H	CCW byte count
			ORG	VCONADDR	
8	8	VCONCOMD	DS	1X	CCW command code

VCUBLOK - VIRTUAL CONTROL UNIT BLOCK

```
   ┌──────────────────────────────────────────┐
 0 |VCUADD    |VCUDVINT |VCUINTS  |V*1 |V*2 |
   |──────────────────────────────────────────|
 8 |                VCUDVTBL                  |
   └──────────────────────────────────────────┘
```

Displacement		Field			
Hex	Dec	Name			Field Description, Contents, Meaning
0	0	VCUADD	DS	1H	Virtual control unit address
2	2	VCUDVINT	DS	1H	VDEVBLOK with interrupt - bit map
4	4	VCUINTS	DS	1H	Virtual control unit interrupt status
6	6	VCUSTAT	DS	1X	V*1 - virtual control unit status
		Bits Defined in VCUSTAT			
		VCUCHBSY	EQU	X'80'	Virtual subchannel busy
		VCUCEPND	EQU	X'40'	Interrupt pending in sub-channel
		VCUBUSY	EQU	X'20'	Virtual control unit busy
		VCUPEND	EQU	X'10'	Virtual control unit interrupt pending
		VCUCUEPN	EQU	X'08'	Virtual control unit end pending
7	7	VCUTYPE	DS	1X	V*2 - virtual control unit type
		Bits Defined in VCUTYPE			
		VCUSHRD	EQU	X'80'	Virtual control unit on shared subchannel
		VCUCTCA	EQU	X'40'	Virtual control unit is a channel-to-channel adapter
8	8	VCUDVTBL	DS	16H	Devices attached - VMDVSTRT index
		VCUSIZE	EQU	(*-VCUBLOK)/8	VCUBLOK size in doublewords (X'05')

VDEVBLOK – VIRTUAL DEVICE BLOCK

```
    ┌────────────────────────────────────────┐
  0 |VDEVADD  |VDEVINTS |V*1 |V*2 |V*3 |V*4 |
    |────────────────────────────────────────|
  8 |              VDEVCSW                   |
    |────────────────────────────────────────|
 10 |VDEVRELN |VDEVBND  |     VDEVPOSN       |
    |────────────────────────────────────────|
 18 | VDEVQUED          | VDEVOPER           |
    |────────────────────────────────────────|
 20 | VDEVLINK          | VDEVEREAL          |
    |────────────────────────────────────────|
 28 |   VDEVIOCT        |   VDEVUSER         |
    |────────────────────────────────────────|
 30 |   VDEVIOER        |   VDEVIOB          |
    └────────────────────────────────────────┘
```

Displacement Hex	Dec	Field Name			Field Description, Contents, Meaning
0	0	VDEVADD	DS	1H	Virtual Device Address
2	2	VDEVINTS	DS	1H	Virtual Device Interrupt Status
4	4	VDEVTYPC	DS	1X	V*1 – virtual device type class
5	5	VDEVTYPE	DS	1X	V*2 – virtual device type
6	6	VDEVSTAT	DS	1X	V*3 – virtual device status
		Bits Defined in VDEVSTAT			
		VDEVCHBS	EQU	X'80'	Virtual subchannel busy
		VDEVCHAN	EQU	X'40'	Virtual channel interrupt pending
		VDEVBUSY	EQU	X'20'	Virtual device busy
		VDEVPEND	EQU	X'10'	Virtual device interrupt pending
		VDEVCUE	EQU	X'08'	Virtual control unit end
		VDEVNRDY	EQU	X'04'	Virtual device not ready
		VDEVCATT	EQU	X'02'	Virtual device attached by console function
		VDEVDED	EQU	X'01'	VDEVREAL is dedicated device RDEVBLOK
7	7	VDEVFLAG	DS	1X	V*4 – virtual device flags
		Bits Defined in VDEVFLAG			
		VDEVRDO	EQU	X'80'	DASD – read-only
		VDEVENAB	EQU	X'80'	Virtual 270X – line enabled
		VDEVTDSK	EQU	X'40'	DASD – TDISK space allocated by CP
		VDEVDIAL	EQU	X'40'	VIRTUAL 270x – line connected
		VDEVCSPL	EQU	X'40'	Console – activity spooled
		VDEV231T	EQU	X'20'	DASD – 2311 simulated on top half of 2314
		VDEV231B	EQU	X'10'	DASD 2311 simulated on bottom half of 2314
		VDEVCCW1	EQU	X'10'	Console and spooling – processing first CCW
		VDEVSAS	EQU	X'08'	DASD – Executing stand-alone seek
		VDEVRSRL	EQU	X'02'	Reserve/Release are valid CCW operation codes
		VDEVUC	EQU	X'01'	Virtual device sense bytes present
8	8	VDEVCSW	DS	1D	Virtual channel status word

```
10    16          VDEVRELN DS    1H               Virtual DASD cylinder relocation
12    18          VDEVBND  DS    1H               Virtual DASD size (in cylinders)
14    20          VDEVPOSN DS    1F               Virtual DASD seek position
18    24          VDEVQUED DS    1F               Virtual SIO to real SIO queued time
1C    28          VDEVOPER DS    1F               Device operational time
20    32          VDEVLINK DS    1F               Link to virtual shared devices
24    36          VDEVREAL DS    1F               Pointer to real device RDEVBLOK
28    40          VDEVIOCT DS    1F               Virtual Device I/O count
2C    44          VDEVUSER DS    1F               Pointer to VMVLOK of VDEVBLOK owner
30    48          VDEVIOER DS    1F               Pointer to IOERBLOK for last error
34    52          VDEVIOB  DS    1F               Pointer to active IOBLOK

                  VDEVSIZE EQU   (*-VDEVBLOK)/8 VDEVBLOK size in doublewords (X'07')
                  For spooling/console devices
                           ORG   VDEVRELN
10    16          VDEVXUSR DS    CL8              Transfered to VMUSER
18    24          VDEVCON  DS    1F               Pointer to VCONCTL console control
1C    28          VDEVSPL  DS    1F               Pointer to VSPLCTL spool control
20    32          VDEVCLAS DS    1C               Spool - output class
21    33          VDEVKEY  DS    1X               Storage key in user's CAW
22    34          VDEVUNIT DS    1H               Spool - output directed device address
24    36          VDEVCOPY DS    1H               Number of copies requested
26    38          VDEVCFLG DS    1X               Console - virtual console flags
                  Bits Defined in VDEVCFLG
                  VDEVATTN EQU   X'80'            User pressed  Attention more than once
                  VDEVTIC  EQU   X'40'            Last CCW processed was a TIC
                  VDEVTRAN EQU   X'20'            Data transfer occurred during this channel program
                  VDEVVCF  EQU   X'10'            Virtual console function in progress
                  VDEVAUCR EQU   X'08'            Auto carriage return on first read

27    39          VDEVSFLG DS    1X               Spool - virtual spool flags
                  Bits Defined in VDEVSFLAG
                  VDEVFEED EQU   X'80'            Spool reader - last command was a feed
                  VDEVXFER EQU   X'80'            Spool output - transfered to VDEVXUSR
                  VDEVCONT EQU   X'40'            Spool input - continuous reading
                  VDEVCP   EQU   X'40'            Spool output - continuous printing
                  VDEVHOLD EQU   X'20'            Hold output - save input
                  VDEVEOF  EQU   X'08'            Spool input - set unit exception at EOF
                  VDEVTERM EQU   X'08'            Terminal output required for spooled console
                  VDEVCFCL EQU   X'04'            Device closed by console function
                  VDEVPURG EQU   X'02'            Spool output - purge file at close
                  VDEVDIAG EQU   X'02'            Spool input - device opened by DIAGNOSE
                  VDEVSVC  EQU   X'01'            Spool output - DMKVSP entered via SVC

                           ORG   VDEVIOER
30    48          VDEVSNSE DS    1F               Sense bytes for spool device
34    52          VDEVFCBK DS    1F               Address of forms control blok (VFCBBLOK)

                           ORG   VDEVLINK
20    32          VDEVTMAT DS    1F               TDISK attached time (TOD clock word 0)
```

VFCBBLOK - VIRTUAL FORM CONTROL BUFFER BLOCK

```
   ┌─────────────────────────────────────────────────┐
 0 | VFCBCNT  |V*1  |V*2 |      VFCBWORK             |
   |─────────────────────────────────────────────────|
 8 |VFCBSPAR  |V*3  |                                |
   |────────────────┘                                |
   |          VFCBLOAD BUFFER AREA                   |
   └─────────────────────────────────────────────────┘
```

Displacement		Field			
Hex	Dec	Name			Field Description, Contents, Meaning
0	0	VFCBCNT	DS	1H	Current pointer to carriage column
2	2	VFCBFLAG	DS	1X	V*1 working flag byte
		Bits Defined in VCFCBFLAG			
		VFCBEOF	EQU	X'80'	End of forms passed once
		VFCBCMD	EQU	X'40'	forms control given
3	3	VFCBCHL	DS	1X	V*2 Channel number or space count
4	4	VFCBWORK	DS	1F	Work area
8	8	VFCBSPAR	DS	2X	Spare
A	10	VCFBNDEX	DS	1X	V*3 Index byte value
B	11	VFCBLOAD	DS	CL181	Form control buffer area
		VFCBSIZE	EQU	(*-VFCBBLOK)/8	Size in doublewords (X'18')

VMBLOK - VIRTUAL MACHINE CONTROL BLOCK

```
     ______________________________________________________________
  0 |     VMQFPNT               |     VMQBPNT                      |
    |--------------------------------------------------------------|
  8 |     VMPNT                 |     VMECEXT                      |
    |--------------------------------------------------------------|
 10 |     VMSEG                 |     VMSIZE                       |
    |--------------------------------------------------------------|
 18 |     VMCHSTRT              |     VMCUSTRT                     |
    |--------------------------------------------------------------|
 20 |     VMDVSTRT              |     VMTERM                       |
    |--------------------------------------------------------------|
 28 |VMCHCNT   |VMCUCNT   |VMDVCNT   |VMIOACTV                     |
    |--------------------------------------------------------------|
 30 |                      VMCHTBL                                 |
    |--------------------------------------------------------------|
 50 |V*3  |V*4  |V*5  |V*6  |V*7  |V*8  |V*9  |V*10                |
    |--------------------------------------------------------------|
 58 |V*11|V*12|V*13|V*14|VMEXTINT  |VMIOINT                        |
    |--------------------------------------------------------------|
 60 |VMSLOCK   |VMLLOCK   |     VMTIMER                            |
    |--------------------------------------------------------------|
 68 |                     VMVTIME                                  |
    |--------------------------------------------------------------|
 70 |                     VMTMOUTQ                                 |
    |--------------------------------------------------------------|
 78 |                     VMTTIME                                  |
    |--------------------------------------------------------------|
 80 |                     VMTMINQ                                  |
    |--------------------------------------------------------------|
 88 |                     VMTODINQ                                 |
    |--------------------------------------------------------------|
 90 |                  VMINST                 |VMUPRIOR            |
    |--------------------------------------------------------------|
 98 |     VMTREXT               |     VMADSTOP                     |
    |______________________________________________________________|
```

```
      ______________________________________________________________
  A0 |                          VMPSW                               |
     |--------------------------------------------------------------|
  A8 |                          VMGPRS                              |
     |--------------------------------------------------------------|
  E8 |                          VMFPRS                              |
     |--------------------------------------------------------------|
 108 |                          VMUSER                              |
     |--------------------------------------------------------------|
 110 |                          VMACNT                              |
     |--------------------------------------------------------------|
 118 |                          VMDIST                              |
     |--------------------------------------------------------------|
 120 |      VMPGREAD             |     VMPGWRIT                     |
     |--------------------------------------------------------------|
 128 |VMWCNT    |VMSEGDSP |     VMSHRSYS                            |
     |--------------------------------------------------------------|
 130 |      VMIOCNT              |     VMPNCH                       |
     |--------------------------------------------------------------|
 138 |      VMLINS               |     VMCRDS                       |
     |--------------------------------------------------------------|
 140 |                          VMCOMND                             |
     |--------------------------------------------------------------|
 148 |      VMTIMEON             |VMDEDCH   |VMQPRIOR               |
     |--------------------------------------------------------------|
 150 |VMPSWDCT |VMVTRMAD |VMPAGES   |VMWSPROJ                       |
     |--------------------------------------------------------------|
 158 |      VMTRQBLK             |VMPRGIL   |VMSTEALS               |
     |--------------------------------------------------------------|
 160 |VMPDRUM   |VMPDISK   |       VMACOUNT                          |
     |--------------------------------------------------------------|
 168 |       VMRDINQ             |       VMPGRINQ                   |
     |--------------------------------------------------------------|
 170 |       VMEPRIOR            |       VMRSV4                     |
     |--------------------------------------------------------------|
 178 |       VMRSV5              |       VMRSV6                     |
     |--------------------------------------------------------------|
 180 |      VMUSER1              |     VMUSER2                      |
     |--------------------------------------------------------------|
 188 |      VMUSER3              |     VMUSER4                      |
     |______________________________________________________________|
```

Displacement Hex	 Dec	Field Name			Field Description, Contents, Meaning
0	0	VMQFPNT	DS	1F	Pointer to next VMBLOK in queue
4	4	VMQBPNT	DS	1F	Pointer to previous VMBLOK in queue
8	8	VMPNT	DS	1F	Pointer (CYCLIC) to next VMBLOK

```
   C   12        VMECEXT  DS    1F            VMBLOK extended control pointer - ECBLOK
   C   12        VMVCR0   EQU   VMECEXT       Virtual control register 0 for non-EC mode machine
  10   16        VMSEG    DS    1F            Pointer to VMSEGTBL
  14   20        VMSIZE   DS    1F            Virtual storage size - bytes
  18   24        VMCHSTRT DS    1F            Pointer to VCHBLOK table
  1C   28        VMCUSTRT DS    1F            Pointer to VCUBLOK table
  20   32        VMDVSTRT DS    1F            Pointer to VDEVBLOK table
  24   36        VMTERM   DS    1F            Pointer to RDEVBLOK for user terminal
  28   40        VMCHCNT  DS    1H            Virtual channel count
  2A   42        VMCUCNT  DS    1H            Virtual control unit count
  2C   44        VMDVCNT  DS    1H            Virtual device count
  2E   46        VMIOACTV DS    1H            Active channel mask
  30   48        VMCHTBL  DS    16H           Channels attached - VMCHSTRT index
  50   80        VMRSTAT  DS    1X            V*3 - virtual machine running status
                 Bits Defined in VMRSTAT
                 VMCFWAIT EQU   X'80'         Waiting - Executing console function
                 VMPGWAIT EQU   X'40'         Waiting - paging operation(s)
                 VMIOWAIT EQU   X'20'         Waiting - Scheduled IOBLOK start
                 VMPSWAIT EQU   X'10'         Waiting - virtual PSW wait state
                 VMEXWAIT EQU   X'08'         Waiting - Instruction simulation
                 VMLOGON  EQU   X'04'         User not logged on
                 VMLOGOFF EQU   X'02'         User logging off
                 VMIDLE   EQU   X'01'         Virtual machine in idle wait state
                 VMCPWAIT EQU   VMCFWAIT+VMPGWAIT+VMIOWAIT+VMEXWAIT+VMLOGOFF+VMLOGON
                 VMNORUN  EQU   VMCPWAIT+VMPSWAIT
                 VMLONGWT EQU   VMCFWAIT+VMLOGON+VMLOGOFF+VMIDLE

  51   81        VMDSTAT  DS    1X            V*4 - virtual machine dispatching status
                 Bits Defined in VMDSTAT
                 VMDSP    EQU   X'80'         Virtual machine is dispatched runuser
                 VMTSEND  EQU   X'40'         Virtual machine is compute bound
                 VMTIO    EQU   X'10'         Virtual Machine is in TIO/SIO busy loop
                 VMRUN    EQU   X'08'         Virtual machine runnable
                 VMINQ    EQU   X'04'         Virtual machine in a queue
|                VMELIG   EQU   X'02'         Reserved for IBM use

  52   82        VMOSTAT  DS    1X            V*5 - virtual machine operating status
                 Bits Defined in VMOSTAT
                 VMSYSOP  EQU   X'80'         Virtual machine is system operator
                 VMSHR    EQU   X'40'         Virtual machine running shared system
                 VMDISC   EQU   X'10'         Virtual machine console disconnected
                 VMCFRUN  EQU   X'08'         Virtual machine running in CF mode
                 VMVIRCF  EQU   X'04'         Virtual machine executing virtual CF
                 VMCF     EQU   X'02'         Virtual machine executing CF
                 VMKILL   EQU   X'01'         Virtual machine is to be logged off

  53   83        VMQSTAT  DS    1X            V*6 - Virtual machine queueing status
                 Bits Defined in VMQSTAT
                 VMPRIDSP EQU   X'80'         Virtual eligible for Queue1
```

```
54   84          VMPSTAT  DS    1X              V*7 - virtual machine processing status
                 Bits Defined in VMPSTAT
                 VMISAM   EQU   X'80'           Virtual machine has ISAM CCW checking
                 VMV370R  EQU   X'40'           Virtual machine can use extended format
                 VMRPAGE  EQU   X'20'           Virtual machine can reserve pages
                 VMREAL   EQU   X'10'           Virtual machine has V=R option
                 VMNOTRAN EQU   X'08'           No CCW translation for V=R user
                 VMPNMCS  EQU   X'04'           Reserved for IBM use
                 VMACCOUN EQU   X'02'           Virtual machine may punch account cards
```

```
  55   85          VMESTAT  DS    1X             V*8 - Virtual machine control status
                   Bits Defined in VMESTAT
                   VMSHADT  EQU   X'80'          Shadcw tables are present
                   VMPERCM  EQU   X'40'          Virtual/CP PER active
|                  VMBADCR0 EQU   X'20'          Virtual control register 0 is invalid
|                  VMBADCR1 EQU   X'10'          Virtual control register 1 is invalid
                   VMEXTCM  EQU   X'08'          Virtual machine in extended control mode
                   VMNEWCR0 EQU   X'04'          Virtual control register 0 has changed
                   VMINVSEG EQU   X'02'          All shadow tables invalid
                   VMINVPAG EQU   X'01'          Shadcw page tables invalid

  56   86          VMTRCTL  DS    1X             V*9 - virtual machine tracing control
                   Bits Defined in VMTRCTL
                   VMTRPER  EQU   X'80'          Virtual PER tracing active
                   VMTRSVC  EQU   X'40'          TRACE user SVC instructions
                   VMTRPRG  EQU   X'20'          TRACE virtual program interrupts
                   VMTRIO   EQU   X'10'          TRACE virtual I/O interrupts
                   VMTREX   EQU   X'08'          TRACE external interrupts
                   VMTRPRV  EQU   X'04'          TRACE user privileged instructions
                   VMTRSIO  EQU   X'02'          TRACE virtual I/O instructions
                   VMTRBRIN EQU   X'01'          Trace successful branches or all instructions
                   VMTRINT  EQU   VMTRSVC+VMTRPRG+VMTRIO+VMTREX  Trace all user interrupts

  57   87          VMMLEVEL DS    1X             V*10- message level
                   Bits Defined in VMMLEVEL
                   VMMSGON  EQU   X'80'          Receiving messages
                   VMWNGON  EQU   X'40'          Receiving warnings
                   VMMCODE  EQU   X'20'          Receiving error message codes
                   VMMTEXT  EQU   X'10'          Receiving texts of error messages
                   VMMLINED EQU   X'08'          Line editing on
                   VMMACCCN EQU   X'04'          Receiving accounting information

| 58   88          VMQLEVEL DS    1X             V*11- queue level
                   Bits Defined in VMQLEVEL
                   VMQ1     EQU   X'80'          Virtual machine is interactive
                   VMCOMP   EQU   X'40'          Virtual machine is compute bound
                   VMHIPRI  EQU   X'20'          Virtual machine is highest priority
                   VMLOPRI  EQU   X'10'          Virtual machine is lowest priority
                   VMAEX    EQU   X'08'          Virtual machine is assured execution
                   VMAEXP   EQU   X'04'          Virtual machine is assured percentage
                   VMDROP1  EQU   X'02'          Virtual machine just dropped from Q1

  59   89          VMCLEVEL DS    1X             V*12- command level
                   Bits Defined in VMCLEVEL
                   VMCLASSA EQU   X'80'          Class A functions
                   VMCLASSB EQU   X'40'          Class B functions
                   VMCLASSC EQU   X'20'          Class C functions
                   VMCLASSD EQU   X'10'          Class D functions
                   VMCLASSE EQU   X'08'          Calss E functions
```

```
                     VMCLASSF EQU   X'04'         Class F functions
                     VMCLASSG EQU   X'02'         Class G functions
                     VMCLASSH EQU   X'01'         Class H functions

    5A   90          VMTLEVEL DS    1X            V*13- timer level
                     Bits Defined in VMTLEVEL
                     VMTON    EQU   X'80'         Virtual timer running
                     VMRON    EQU   X'40'         Virtual real timer running
                     VMSTMPI  EQU   X'08'         Virtual interval timer request queued
                     VMSTMPT  EQU   X'04'         Virtual CPU timer request queued

    5B   91          VMPEND   DS    1X            V*14- Interrupt pending summary flag
                     Bits Defined in VMPEND
|                    VMPERPND EQU   X'40'         Virtual PER interrupt pending
|                    VMPRGPND EQU   X'20'         Virtual program interrupt deferred
                     VMSVCPND EQU   X'10'         Virtual SVC interrupt deferred
                     VMIOPND  EQU   X'02'         Virtual I/O interrupt pending
                     VMEXTPND EQU   X'01'         Virtual external interrupt pending

    5C   92          VMEXTINT DS    1H            External interrupt pending flags
                     Bits Defined in VMEXTINT
                     VMCKCINT EQU   X'08'         Clock comparator interrupt pending
                     VMCPTINT EQU   X'04'         CPU timer interrupt pending
                     Bits Defined in VMEXTINT+1
                     VMINTINT EQU   X'80'         Interval timer interrupt pending
                     VMKEYINT EQU   X'40'         RED button interrupt pending
                     VMSIGINT EQU   X'2F'         External signals pending

    5E   93          VMIOINT  DS    1H            I/O interrupt pending flags
    60   96          VMSLOCK  DS    1H            Short lock - reserved for IBM use
    62   98          VMLLOCK  DS    1H            Long lock - reserved for IBM use
|   64  100          VMTIMER  DS    1F            Virtual timer value - X'50'
    68  104          VMVTIME  DS    1D            Virtual CPU time used - 2s complement
    70  112          VMTMOUTQ DS    1D            VMTTIME for exit from queue -2s complement
    78  120          VMTTIME  DS    1D            Total CPU time used - 2s complement
    80  128          VMTMINQ  DS    1D            VMTTIME value at entry to queue
    88  136          VMTODINQ DS    1D            TOD clock time stamp at queue entry
    90  144          VMINST   DS    XL6           Virtual machine privileged or tracing instruction
    96  150          VMUPRIOR DS    1H            User priority from directory
    98  152          VMTREXT  DS    1F            Address of extended trace control block
    9C  156          VMADSTOP DS    1F            Address of address stop control block
    A0  160          VMPSW    DS    1D            Virtual machine PSW
    A8  168          VMGPRS   DS    16F           Virtual machine general registers
    E8  232          VMFPRS   DS    4D            Virtual machine floating point registers
   108  264          VMUSER   DS    CL8           Virtual machine identification
   110  272          VMACNT   DS    CL8           Virtual machine accounting number
   118  280          VMDIST   DS    CL8           Virtual machine distribution code
   120  288          VMPGREAD DS    1F            Total page reads
   124  292          VMPGWRIT DS    1F            Total page writes
   128  296          VMWCNT   DS    1H            Page wait count
```

```
12A  298         VMSEGDSP DS   1H            Displacement of virtual machine SEGTABLE from start of block
12C  300         VMSHRSYS DS   1F            Pointer to shared system table
130  304         VMIOCNT  DS   1F            Virtual SIO count for non-spooled I/O
134  308         VMPNCH   DS   1F            Virtual card count - spooled punch
138  312         VMLINS   DS   1F            Virtual line count - spooled printer
13C  316         VMCRDS   DS   1F            Virtual card count - spooled reader
140  320         VMCOMND  DS   CL8           Last CP command executed
148  328         VMTIMEON DS   1F            LOGON time -TOD clock word 0
14C  332         VMDEDCH  DS   1H            Dedicated channel mask
14E  334         VMQPRIOR DS   1H            Priority in dispatching queue
150  336         VMPSWDCT DS   1H            Count of incorrect passwords entered
152  338         VMVTRMAD DS   1H            Virtual terminal device address
154  340         VMPAGES  DS   1H            Number of pages currently resident
156  342         VMWSPROJ DS   1H            Projected working set size
158  344         VMTRQBLK DS   1F            Address of TRQBLOK for real timer
15C  348         VMPRGIL  DS   1H            ILC for pending program interrupt
15E  350         VMSTEALS DS   1H            Number of waits for stolen pages
160  352         VMPDRUM  DS   1H            Reserved for IBM use
162  354         VMPDISK  DS   1H            Reserved for IBM use
164  356         VMACOUNT DS   1F            Address of user ACCTBLOK
168  360         VMRDINQ  DS   1F            Page read total (VMPGREAD) at Q entry
16C  364         VMPGRINQ DS   1F            Sum of VMPAGES count at each page read
170  368         VMEPRIOR DS   1F            Eligible list priority
174  372         VMRSV4   DS   1F            Reserved for IBM use
178  376         VMRSV5   DS   1F            Reserved for IBM use
17C  380         VMRSV6   DS   1F            Reserved for IBM use
180  384         VMUSER1  DS   1F            Reserved for installation use
184  388         VMUSER2  DS   1F            Reserved for installation use
188  392         VMUSER3  DS   1F            Reserved for installation use
18C  396         VMUSER4  DS   1F            Reserved for installation use

                 VMBSIZE  EQU  (*-VMBLOK)/8  VMBLOK size in doublewords (X'32')
```

VSPLCTL - VIRTUAL SPOOL CONTROL BLOCK

```
    ┌──────────────────────────────────────────────┐
  0 |   VSPCAW           |    VSPDPAGE             |
    |──────────────────────────────────────────────|
  8 |   VSPVPAGE         |    VSPRECNO             |
    |──────────────────────────────────────────────|
 10 |VSPNEXT  |VSPIDACT  |    VSPSFBLK             |
    |──────────────────────────────────────────────|
 18 |                 VSPCCW                       |
    |──────────────────────────────────────────────|
 20 |   VSPBUFBK         |     VSPMISC             |
    |──────────────────────────────────────────────|
 28 |V*1 |   VSPIDAL     |     VSPIDAW2            |
    └──────────────────────────────────────────────┘
```

Displacement Hex	Dec	Field Name			Field Description, Contents, Meaning
0	0	VSPCAW	DS	1F	Virtual address of user CCW
4	4	VSPDPAGE	DS	1F	DASD location (DCHR) of current page buffer
8	8	VSPVPAGE	DS	1F	Virtual address of page buffer
C	12	VSPRECNO	DS	1F	Records remaining in current buffer
10	16	VSPNEXT	DS	1H	DISP. in buffer of next record start
12	18	VSPIDACT	DS	1H	Data byte count of IDA CCW
14	20	VSPSFBLK	DS	1F	Pointer to SFBLOK for file
18	24	VSPCCW	DS	1D	Current user CCW
20	32	VSPBUFBK	DS	1F	Address of a buffer area
24	36	VSPMISC	DS	1F	Use varies according to caller
28	40	VSPIDASW	DS	1X	V*1 IDA work flag
29	41	VSPIDAL	DS	3X	Address of indirect data list
2C	44	VSPIDAW2	DS	1F	Contains IDAW2
		VSPSIZE	EQU	(*-VSPLCTL)/8	Size in doublewords (X'06')
		VSPBUFSZ	EQU	(200)/8	Size in doublewords (X'19')

COMMAND-TO-MODULE CROSS-REFERENCE

Command	Entry Point	Messages
ACNT	DMKCPVAC	DMKCPV003E
		DMKCPV007E
		DMKCPV020E
		DMKCPV045E
ADSTOP	DMKCFDAD	DMKCFD004E
		DMKCFD026E
		DMKCFD160E
		DMKCFD161E
ATTACH (channel)	DMKVCHDC	DMKVCH034E
		DMKVCH048E
		DMKVCH129E
		DMKVCH131E
		DMKVCH132E
	DMKVDBAT	DMKVDB020E
		DMKVDB045E
ATTACH	DMKVDBAT	DMKVDB003E
		DMKVDB006E
		DMKVDB020E
		DMKVDB021E
		DMKVDB022E
		DMKVDB023E
		DMKVDB034E
		DMKVDB040E
		DMKVDB045E
		DMKVDB046E
		DMKVDB120E
		DMKVDB122E
		DMKVDB123E
		DMKVDB124E
		DMKVDB125E
		DMKVDB126E
		DMKVDB127E
		DMKVDB128E
		DMKVDB133E
		DMKVDB134E
		DMKVDB142E
		DMKVDB143E
BACKSPACE	DMKCSOBS	DMKCSO003E
		DMKCSO006E
		DMKCSO021E
		DMKCSO040E
		DMKCSO046E
		DMKCSO140E
		DMKCSO141E
BEGIN	DMKCFMBE	DMKCFM004E
CHANGE	DMKCSUCH	DMKCSU003E
		DMKCSU006E
		DMKCSU008E
		DMKCSU013E
		DMKCSU026E
		DMKCSU027E
		DMKCSU028E
		DMKCSU029E
		DMKCSU030E
		DMKCSU032E
		DMKCSU035E
		DMKCSU042E
CLOSE	DMKCSPCL	DMKCSP003E
		DMKCSP006E
		DMKCSP013E
		DMKCSP022E
		DMKCSP029E
		DMKCSP032E
		DMKCSP040E
COUPLE	DMKDIACP	DMKDIA006E
		DMKDIA011E
		DMKDIA020E
		DMKDIA022E
		DMKDIA040E
		DMKDIA045E
		DMKDIA047E
		DMKDIA058E
DCP	DMKCDBDC	DMKCDB003E

		DMKCDB004E
		DMKCDB009E
		DMKCDB010E
		DMKCDB026E
		DMKCDB033E
		DMKCDB160E
DEFINE	DMKDEFIN	DMKDEF003E
		DMKDEF022E
		DMKDEF024E
		DMKDEF025E
		DMKDEF026E
		DMKDEF040E
		DMKDEF091E
		DMKDEF092E
		DMKDEF094E
		DMKDEF136E
DETACH (channel)	DMKVCHDC	DMKVCH034E
		DMKVCH048E
		DMKVCH130E
	DMKVDBDE	DMKVDB020E
		DMKVDB034E
DETACH	DMKVDBDE	DMKVDB006E
		DMKVDB020E
		DMKVDB021E
		DMKVDB022E
		DMKVDB040E
		DMKVDB045E
		DMKVDB046E
		DMKVDB121E
		DMKVDB123E
		DMKVDB124E
		DMKVDB135E
		DMKVDB140E
DIAL	DMKDIAL	DMKDIA011E
		DMKDIA020E
		DMKDIA022E
		DMKDIA045E
		DMKDIA047E
		DMKDIA055E
		DMKDIA056E
DISABLE	DMKCPVDS	DMKCPV003E
		DMKCPV006E
		DMKCPV021E
		DMKCPV026E
		DMKCPV040E
		DMKCPV046E
		DMKCPV140E
DISCONN	DMKUSODS	DMKUSO003E
DISPLAY	DMKCDBDI	DMKCDB003E
		DMKCDB004E
		DMKCDB009E
		DMKCDB010E
		DMKCDB026E
		DMKCDB160E
DMCP	DMKCDBDM	DMKCDB003E
		DMKCDB004E
		DMKCDB009E
		DMKCDB033E
		DMKCDB160E
DRAIN	DMKCSODR	DMKCSO003E
		DMKCSO006E
		DMKCSO021E
		DMKCSO040E
		DMKCSO046E
		DMKCSO140E
DUMP	DMKCDBDU	DMKCDB003E
		DMKCDB004E
		DMKCDB009E
		DMKCDB033E
		DMKCDB160E
ECHO	DMKMSGEC	none.
ENABLE	DMKCPVEN	DMKCPV003E
		DMKCPV006E
		DMKCPV021E
		DMKCPV026E
		DMKCPV040E
		DMKCPV046E
		DMKCPV140E
EXTERNAL	DMKCPBEX	DMKCPB005E
FLUSH	DMKCSOFL	DMKCSO003E
		DMKCSO006E
		DMKCSO013E
		DMKCSO021E
		DMKCSO040E
		DMKCSO046E
		DMKCSO140E

		DMKCSO141E
FORCE	DMKUSOFL	DMKUSO003E
		DMKUSO020E
		DMKUSO045E
FREE	DMKCSPFR	DMKCSP006E
		DMKCSP007E
		DMKCSP020E
		DMKCSP053E
HALT	DMKCPVH	DMKCPV021E
		DMKCPV040E
		DMKCPV144W
HOLD	DMKCSPHL	DMKCSP006E
		DMKCSP007E
		DMKCSP020E
		DMKCSP053E
IPL	DMKCFPIP	DMKCFP002E
		DMKCFP003E
		DMKCFP013E
		DMKCFP022E
		DMKCFP026E
		DMKCFP040E
		DMKCFP044E
		DMKCFP170E
		DMKCFP171E
		DMKCFP172E
		DMKCFP173E
		DMKCFP174E
		DMKCFP177E
		DMKVMI230E
		DMKVMI231E
		DMKVMI232E
		DMKVMI233E
		DMKVMI234E
LINK	DMKLNKIN	DMKLNK020E
		DMKLNK022E
		DMKLNK052E
		DMKLNK053E
		DMKLNK101W
		DMKLNK102W
		DMKLNK103W
		DMKLNK104E
		DMKLNK105E
		DMKLNK106E
		DMKLNK107E
		DMKLNK108E
		DMKLNK109E
		DMKLNK110E
		DMKLNK111E
		DMKLNK112E
		DMKLNK113E
		DMKLNK114E
		DMKLNK115E
		DMKLNK116E
		DMKLNK117E
		DMKLNK137E
LOADBUF	DMKCSOLD	DMKCSO003E
		DMKCSO006E
		DMKCSO013E
		DMKCSO021E
		DMKCSO026E
		DMKCSO031E
		DMKCSO036E
		DMKCSO040E
		DMKCSO043E
		DMKCSO046E
		DMKCSO140E
		DMKCSO142E
		DMKCSO148E
LOADVFCB	DMKCSOVL	DMKCSO006E
		DMKCSO022E
		DMKCSO026E
		DMKCSO031E
		DMKCSO036E
		DMKCSO040E
		DMKCSO043E
LOCATE	DMKCFDLO	DMKCFD021E
		DMKCFD022E
		DMKCFD026E
		DMKCFD040E
LOCK	CMKCPVLK	DMKCPV004E
		DMKCPV009E
		DMKCPV020E
		DMKCPV033E
		DMKCPV045E
		DMKCPV160E
LOGOFF (LOGOUT)	DMKUSOLG	DMKUSO003E

LOGON (LOGIN, operator)	DMKLOGOP	DMKLOG003E
		DMKLOG020E
LOGON (LOGIN, user)	DMKLOGON	DMKLOG050E
		DMKLOG051E
		DMKLOG052E
		DMKLOG053E
		DMKLOG054E
		DMKLOG090E
		DMKLOG091E
		DMKLOG092E
		DMKLOG093E
MONITOR	DMKMCCCL	DMKMCC002E
		DMKMCC026E
MSG	DMKMSGMS	DMKMSG003E
		DMKMSG020E
		DMKMSG045E
		DMKMSG057W
NOTREADY	DMKCPBNR	DMKCPB006E
		DMKCPB022E
		DMKCPB040E
ORDER	DMKCSUOR	DMKCSU003E
		DMKCSU006E
		DMKCSU008E
		DMKCSU026E
		DMKCSU027E
		DMKCSU028E
		DMKCSU035E
		DMKCSU042E
PURGE	DMKCSUPU	DMKCSU003E
		DMKCSU006E
		DMKCSU008E
		DMKCSU026E
		DMKCSU028E
		DMKCSU035E
		DMKCSU042E
QUERY (initialize)	DMKCFMQU	DMKCFM026E
(Class G)	DMKCQGEN	DMKCQG020E
		DMKCQG022E
		DMKCQG027E
		DMKCQG040E
		DMKCQG042E
		DMKCQG045E
(Class B,E,G)	DMKCQPRV	DMKCQP003E
		DMKCQP006E
		DMKCQP020E
		DMKCQP021E
		DMKCQP022E
		DMKCQP040E
		DMKCQP045E
READY	DMKCPBRY	DMKCPB006E
		DMKCPB022E
		DMKCPB040E
REPEAT	DMKCSORP	DMKCSO003E
		DMKCSO006E
		DMKCSO013E
		DMKCSO021E
		DMKCSO030E
		DMKCSO040E
		DMKCSO046E
		DMKCSO140E
		DMKCSO141E
RESET	DMKCPBRS	DMKCPB022E
		DMKCPB040E
REWIND	DMKCPBRW	DMKCPB006E
		DMKCPB022E
		DMKCPB040E
		DMKCPB059E
SAVESYS	DMKCFGSV	DMKCFG026E
		DMKCFG044E
		DMKCFG170E
		DMKCFG171E
		DMKCFG172E
		DMKCFG173E
		DMKCFG435E
SET	DMKCFSET	DMKCFS003E
		DMKCFS006E
		DMKCFS013E
		DMKCFS021E
		DMKCFS026E
		DMKCFS040E
		DMKCFS041E
		DMKCFS045E
		DMKCFS046E
		DMKCFS140E
		DMKCFS175E
	DMKMCHMS	DMKMCH003E
		DMKMCH026E

SHUTDOWN	DMKCPVSH	none.
SLEEP	DMKCFMSL	none.
SPACE	DMKCSOSP	DMKCSO006E DMKCSO021E DMKCSO040E DMKCSO046E DMKCSO140E DMKCSO141E
SPOOL	DMKCSPSP	DMKCSP003E DMKCSP006E DMKCSP007E DMKCSP013E DMKCSP020E DMKCSP022E DMKCSP026E DMKCSP028E DMKCSP030E DMKCSP040E DMKCSP053E
START	DMKCSOSD	DMKCSO003E DMKCSO006E DMKCSO013E DMKCSO021E DMKCSO028E DMKCSO040E DMKCSO046E DMKCSO140E
STCP	DMKCDSCP	DMKCDS004E DMKCDS005E DMKCDS026E DMKCDS033E DMKCDS160E DMKCDS162E
STORE	DMKCDSTO	DMKCDS004E DMKCDS005E DMKCDS010E DMKCDS012E DMKCDS026E DMKCDS033E DMKCDS160E DMKCDS161E DMKCDS162E DMKCDS163E
SYSTEM	DMKCPBSR	DMKCPB012E DMKCPB026E
TERMINAL	DMKCFTRM	DMKCFT002E DMKCFT006E DMKCFT026E
TRACE	DMKTRA	DMKTRA002E DMKTRA003E DMKTRA013E DMKTRA026E DMKTRA180E DMKTRA181E
TRANSFER	DMKCSUTR	DMKCSU003E DMKCSU007E DMKCSU008E DMKCSU020E DMKCSU026E DMKCSU027E DMKCSU028E DMKCSU042E DMKCSU053E
UNLOCK	DMKCPVUL	DMKCPV004E DMKCPV009E DMKCPV020E DMKCPV033E DMKCPV045E DMKCPV160E DMKCPV176E DMKCPV202E
VARY	DMKCPVRY	DMKCPV003E DMKCPV021E DMKCPV026E DMKCPV040E DMKCPV049E DMKCPV123E DMKCPV124E DMKCPV140E DMKCPV142E
WNG	DMKMSGWN	DMKMSG003E DMKMSG020E DMKMSG045E DMKMSG057W

MESSAGE-TO-FLOWCHART CROSS-REFERENCE

DMKAC0425A	4
DMKBLD200E	1
DMKBLD201E	1
DMKBLD202E	1
DMKCCH601I	4
DMKCCH602I	4
DMKCCH603W	1
DMKCCH604I	3
DMKCCH605I	2
DMKCCH606I	4 (in module DMKIOE)
DMKCDB003E	4
DMKCDB004E	7
DMKCDB009E	8
DMKCDB010E	7
DMKCDB026E	1
DMKCDB033E	3
DMKCDB160E	8
DMKCDS004E	3
DMKCDS005E	3
DMKCDS010E	5
DMKCDS012E	4
DMKCDS026E	1
DMKCDS033E	3
DMKCDS160E	2
DMKCDS161E	2
DMKCDS162E	5
DMKCDS162W	6
DMKCDS163E	5
DMKCFD004E	3
DMKCFD021E	2
DMKCFD022E	1
DMKCFD026E	1
DMKCFD040E	1
DMKCFD160E	3
DMKCFD161E	3
DMKCFG026E	1
DMKCFG044E	1
DMKCFG170E	1
DMKCFG171E	2
DMKCFG172E	2
DMKCFG173E	2
DMKCFG435E	3

DMKCFM001E	2
DMKCFM004E	4
DMKCFM026E	5
DMKCFP002E	6
DMKCFP003E	5
DMKCFP013E	6
DMKCFP026E	9
DMKCFP040E	7
DMKCFP044E	5
DMKCFP170E	8
DMKCFP171E	8
DMKCFP172E	8
DMKCFP173E	8
DMKCFP174E	7
DMKCFP177E	10
DMKCFS003E	1
DMKCFS006E	9
DMKCFS013E	13
DMKCFS021E	7
DMKCFS026E	1
DMKCFS040E	7
DMKCFS041E	10
DMKCFS045E	4
DMKCFS046E	9
DMKCFS140E	9
DMKCFS175E	5
DMKCFT002E	1
DMKCFT006E	5
DMKCFT026E	1
DMKCKP900E	7
DMKCKP901E	7
DMKCKP902E	3
DMKCKP904E	5
DMKCKP910I	3
DMKCKP911W	3
DMKCKP912W	3
DMKCKP960I	8
DMKCKP961W	8
DMKCNS454I	7
DMKCNS455I	1
DMKCNS500I	21
DMKCNS501I	21
DMKCNS502I	21
DMKCNS503I	21

DMKCNS504I	21
DMKCNS505I	21
DMKCNS527I	21
DMKCNS528I	22

DMKCPB005E	2
DMKCPB006E	3
DMKCPB012E	2
DMKCPB022E	4
DMKCPB026E	1
DMKCPB040E	4
DMKCPB059E	4

DMKCPI950A	4
DMKCPI951I	4
DMKCPI952I	4
DMKCPI953I	5
DMKCPI954E	1
DMKCPI955W	2
DMKCPI960I	6
DMKCPI961W	6

DMKCPV003E	7
DMKCPV004E	4
DMKCPV006E	2
DMKCPV007E	6
DMKCPV009E	5
DMKCPV020E	4
DMKCPV021E	1
DMKCPV026E	1
DMKCPV033E	4
DMKCPV040E	1
DMKCPV045E	4
DMKCPV046E	2
DMKCPV049E	8
DMKCPV123E	9
DMKCPV124E	8
DMKCPV140E	2,8
DMKCPV142E	8
DMKCPV144W	10
DMKCPV160E	5
DMKCPV176E	4
DMKCPV202E	4

DMKCQG020E	7
DMKCQG022E	6
DMKCQG027E	3
DMKCQG040E	7
DMKCQG042E	4
DMKCQG045E	7

DMKCQP003E	2
DMKCQP006E	4
DMKCQP020E	3
DMKCQP021E	4
DMKCQP022E	4
DMKCQP040E	4
DMKCQP045E	3

DMKCSO003E	1
DMKCSO006E	1
DMKCSO013E	1
DMKCSO021E	12
DMKCSO022E	12
DMKCSO026E	13
DMKCSO028E	3
DMKCSO030E	6
DMKCSO031E	8
DMKCSO036E	10
DMKCSO040E	12
DMKCSO043E	8
DMKCSO046E	1
DMKCSO140E	1
DMKCSO141E	2
DMKCSO142E	7
DMKCSO148E	11

DMKCSP003E	1
DMKCSP006E	1
DMKCSP007E	5
DMKCSP013E	1
DMKCSP020E	5
DMKCSP022E	7
DMKCSP026E	4
DMKCSP028E	8
DMKCSP029E	10
DMKCSP030E	9
DMKCSP032E	12
DMKCSP040E	8
DMKCSP053E	5

DMKCSU003E	4
DMKCSU006E	1
DMKCSU007E	10
DMKCSU008E	15
DMKCSU013E	2
DMKCSU020E	8
DMKCSU026E	4
DMKCSU027E	1
DMKCSU028E	12
DMKCSU029E	14

DMKCSU030E	13
DMKCSU032E	3
DMKCSU035E	12
DMKCSU042E	4
DMKCSU053E	11
DMKDAS500I	4
DMKDAS501A	4
DMKDAS502D	4
DMKDAS503I	2
DMKDAS504D	4
DMKDAS505D	4
DMKDAS506I	5
DMKDAS507D	4
DMKDAS508I	7
DMKDAS509I	5
DMKDAS513I	5
DMKDAS514D	2,4
DMKDAS516I	5
DMKDAS517I	6
DMKDAS518I	6
DMKDAS520I	2
DMKDAS956A	3
DMKDEF003E	3
DMKDEF022E	1
DMKDEF024E	2
DMKDEF025E	1
DMKDEF026E	1
DMKDEF040E	4
DMKDEF091E	2
DMKDEF092E	3
DMKDEF094E	6
DMKDEF136E	2
DMKDIA006E	8
DMKDIA011E	3
DMKDIA020E	1
DMKDIA022E	1
DMKDIA040E	9
DMKDIA045E	1
DMKDIA047E	1
DMKDIA055E	2
DMKDIA056E	2
DMKDIA058E	9
DMKDMP905W	5
DMKDMP906W	5
DMKDMP907W	3
DMKDMP908I	1
DMKDMP909W	8
DMKDSP450W	4
DMKDSP451W	1
DMKDSP452W	3
DMKIOF550E	4
DMKIOF551E	5
DMKIOF552E	12
DMKIOF553E	5
DMKIOG554E	3
DMKIOG555E	3
DMKIOG556I	4
DMKIOG557I	4
DMKIOG558I	6
DMKIOG559I	5
DMKIOG560I	5
DMKLNK020E	1
DMKLNK022E	2
DMKLNK052E	4
DMKLNK053E	2
DMKLNK101W	10
DMKLNK102W	10
DMKLNK103W	10
DMKLNK104E	10
DMKLNK105E	11
DMKLNK106E	11
DMKLNK107E	3
DMKLNK108E	8
DMKLNK109E	4
DMKLNK110E	8
DMKLNK111E	8
DMKLNK112E	8
DMKLNK113E	8
DMKLNK114E	6
DMKLNK115E	1
DMKLNK116E	5
DMKLNK117E	5
DMKLNK137E	5
DMKLOG003E	2
DMKLOG020E	1
DMKLOG050E	2
DMKLOG051E	1
DMKLOG052E	7
DMKLOG053E	1
DMKLOG054E	9
DMKLOG090E	8

DMKLOG091E	6
DMKLOG092E	5
DMKLOG093E	6
DMKMCC002E	1
DMKMCC026E	1
DMKMCH003E	8
DMKMCH026E	8
DMKMCH610I	2
DMKMCH611I	1,7
DMKMCH612W	1
DMKMCH613I	3
DMKMCH614I	4
DMKMCH615I	7
DMKMCH616I	3
DMKMCH617I	7
DMKMCH618I	8
DMKMCH619I	3
DMKMID453I	1
DMKMSG003E	1
DMKMSG020E	1
DMKMSG045E	1
DMKMSG057W	2
DMKPAG415E	3
DMKPGT400I	4
DMKPGT401I	4
DMKPRG453W	3
DMKPTR410W	2
DMKRSE500I	7
DMKRSE501A	7
DMKRSE501I	7
DMKRSE502I	7
DMKRSE503A	7
DMKRSE503I	7
DMKRSE504A	7
DMKRSE504I	7
DMKRSE505A	7
DMKRSE508I	7
DMKRSE520A	7
DMKRSE520I	7
DMKRSE521I	7
DMKRSE524I	7

DMKRSE525I	7
DMKRSE529I	7
DMKRSP426E	1
DMKRSP428E	3
DMKRSP430A	13
DMKRSP431A	11
DMKRSP432A	11
DMKRSP433A	11
DMKRSP434A	9
DMKSAV350W	2
DMKSAV351W	2
DMKSAV352W	3
DMKTAP500I	4
DMKTAP501A	4
DMKTAP502D	2
DMKTAP503I	2
DMKTAP504D	5,7
DMKTAP505D	4
DMKTAP510I	6
DMKTAP511I	2
DMKTAP512I	5
DMKTAP513I	6
DMKTAP516I	7
DMKTAP517I	2
DMKTAP518I	8
DMKTAP519I	6
DMKTAP520I	2
DMKTAP521I	4
DMKTAP522I	5
DMKTAP523I	5
DMKTRA002E	1
DMKTRA003E	1
DMKTRA013E	2
DMKTRA026E	1
DMKTRA180W	4
DMKTRA181E	3
DMKTRA182E	1
DMKUDR475I	2
DMKUSO003E	5
DMKUSO020E	2
DMKUSO045E	2
DMKVCH034E	1
DMKVCH048E	1

DMKVCH129E	2
DMKVCH130E	5
DMKVCH131E	2
DMKVCH132E	2
DMKVDB003E	3
DMKVDB006E	6
DMKVDB020E	2
DMKVDB021E	1
DMKVDB022E	1
DMKVDB023E	3
DMKVDB034E	1
DMKVDB040E	9
DMKVDB045E	10
DMKVDB046E	10
DMKVDB120E	5
DMKVDB121E	8
DMKVDB122E	11
DMKVDB123E	8
DMKVDB124E	11
DMKVDB125E	6
DMKVDB126E	6
DMKVDB127E	11
DMKVDB128E	7
DMKVDB133E	13
DMKVDB134E	11
DMKVDB135E	11
DMKVDB140E	10
DMKVDB142E	8
DMKVDB143E	8
DMKVMI110E	1
DMKVMI230E	2
DMKVMI231E	7
DMKVMI232E	2
DMKVMI233E	3
DMKVMI234E	2
DMKVSP427I	6
DMKVSP429I	5
DMKWRM902E	4
DMKWRM903E	2
DMKWRM904E	3
DMKWRM909E	2
DMKWRM910I	2
DMKWRM911W	2
DMKWRM920I	4

CP WAIT STATE CODES

The wait state code is found in the right half of the program status word (PSW) when the CPU is in wait state. A wait state is produced by one cf the following modules:

DMKCCH
DMKCKP
DMKCPI
DMKDMP
DMKMCH
DMKSAV
DMKWRM

When a wait state occurs, the PSW is displayed at the operator's console in the following format:

```
    xxyyyyyyzzzzzwww
```

Where:

xxyyyyyy is the left half of the program status word.

This half may be either:

03yyyyyy Valid wait condition. The system is waiting for work.

00yyyyyy System wait caused by an error condition.

zzzzzwww is the right half of the program status word. The wait state code, www, indicates the error condition.

Wait Codes	Explanation
001	The machine check handler has encountered an irrecoverable failure. Probable hardware error.
002	The channel check handler has encountered an irrecoverable failure. Probable hardware error.
003	A system failure has occurred before a valid warm start was performed.
004	This wait state code is loaded by DMKDMP when a console, or an output device is not operational, or when a console or output device produce an inexplicable error status. Probable hardware error.
005	DMKCPI could not find an operational primary or alternate console. Probable hardware error.
006	This is a normal wait when a system shutdown is completed.
007	A program check, a machine check, or a permanent I/O error was encountered by the checkpoint program.
008	Checkpoint and system shutdown are complete.
009	An error condition has occurred that prevents a warm start.
00A	A machine check occurred while DMKSAV was attempting to save or restore a page image copy of the nucleus on a SYSRES device. Probable hardware error.
00B	A machine check occurred before initiation was complete.
00C	An attempt was made to IPL from a disk that did not contain a system. Thus, the wait code 00C enter on disk by the Format program is encountered.
00D	The size defined during system generation is greater than the real machine size, or a hardware error has occurred which inhibits VM/370 from using the required storage.
00F	Hardware errors are being received on VM/370 paging device(s). This wait state is preceded by message DMKPAG415E - CONTINUOUS PAGING ERRORS FROM DASDxxx.

ABEND CODES

Code	Reason
BLD001	An invalid pointer to the RDEVBLOK was found in register 8 when DMKBLDVM was called to build a new VMBLOK.
CFM001	No stacked CPEXBLOK was found for a user with a pending LOGOFF flag set.
CNS001	Condition code 2 was returned by a TIO instruction to a logged on user communication line or console.
CNS002	Condition code 2 or 3 was returned from a SIO instruction to a logged on user communication line or console.
CNS003	Condition code 1 was returned from a SIO instruction to a logged on user communication line or console, accompanied by CSW status other than Attention or Unit Check.
CNS004	The input data count, less idle and control characters, for a read from a 2741 is less than 0.
CNS005	The input data count for a read from a 2741 is equal to 0.
CNS006	The data count at entry to the code translation routine is less than or equal to 0.
CNS007	The input data count less idle and pad characters is less than zero for a non-IBM terminal.
CNS008	The IOBIOER field of the IOBLOK contains an invalid pointer to an IOERBLOK.
CPI001	The RDEVBLOK for the DASD on which the SYSRES volume is mounted cannot be located. The SYSRES volume is specified in the SYSRES macro in the module DMKSYS.
CPI002	A valid system directory file could not be located.
CPI003	The system TOD clock is not operational.
CVT001	The system TOD clock is in error or is not operational.
DRD001	The device code index in the compressed DASD address for the system dump file points to a RDEVLBOK for an invalid DASD. The valid DASDs are 2305, 3330, or 2314/2319.
DSP001	During I/O Interrupt Unstack and Reflection, DMKSCNVU could not locate all of the virtual control blocks for the interrupting unit.
DSP002	The dispatcher (DMKDSP) is attempting to dispatch a virtual relocate user whose shadow segment tables or virtual extended control register 0 are invalid.
DSP003	The dispatcher has sensed that the interval timer did not decrement properly.
DSP004	A virtual device was detached while an I/O interrupt was being traced and its VDEVBLOK cannot be found.
FRE001	The size of the block being returned (via register 0) is less than or equal to 0.
FRE002	The address of the free storage block being returned matches the address of a block in the free storage chain.
FRE003	The address of the free storage block being returned overlaps the next lower block on the free storage chain.
FRE004	The address of the free storage block being returned overlaps the next higher block on the free storage chain.
FRE005	A module is attempting to release storage in the resident CP nucleus.
FRE006	A module is requesting a block of storage whose size (in register 0) is less than or equal to zero.

FRE007 A module is attempting to release a block of storage whose address exceeds the size of real storage.

FRE008 The address of the free storage block being returned matches the address of the first block in the subpool for that size.

FRE009 The address of the free storage block being returned matches the address of the second block in the subpool for that size.

FRE010 A program is attempting to extend free storage while storage is in the process of being extended.

FRE011 A CP module has attempted to return a block of storage that is in the user dynamic paging area.

HVC001 The user pointed to by register 11 issued a diagnose while attempting to format the I/O Error or Channel Check/Machine Check recording areas: the SYSRES device type is unrecognizable.

IOS001 The caller is attempting to reset an active IOBLOK that contains an invalid unit address.

IOS002 DMKIOS is attempting to restart an IOBLOK from the RCHBLOK queue, but that IOBLOK contains an invalid unit address.

IOS003 DMKIOS is attempting to remove an IOBLOK from a queue, but that IOBLOK is on more than one queue.

PGT001 The number of cylinders in use stored in the allocation block (ALOCBLOK) is less than the maximum but DMKPGT was unable to find available cylinders.

PGT002 The count of pages in use in a page allocation block (RECBLOK) is less than the maximum but DMKPGT was unable to find available pages.

PGT003 The DASD page slot being released is not marked allocated.

PGT004 The dummy RECBLOK indicating the spooling DASD pages on the cylinder that are to be released contains a page count greater than the number of pages allocated on the cylinder.

PGT005 A module is attempting to release a DASD page slot on a cylinder for which no page allocation block (RECBLOK) exists.

PGT006 The last DASD page slot in a RECBLOK has been deallocated but the bit representing the cylinder in the cylinder allocation block (ALOCBLOK) is not set to one, indicate that the cylinder was not allocated.

PGT007 A module is attempting to release a page of virtual storage being used by CP that has not been marked allocated.

PRG001 Program check (operation) in the control program.

PRG002 Program check (privileged operation) in the control program.

PRG003 Program check (execute) in the control program.

PRG004 Program check (protection) in the control program.

PRG005 Program check (addressing) in the control program.

PRG006 Program check (specification) in the control program.

PRG007 Program check (data) in the control program.

PRG008 Program check (fixed-point overflow) in the control program.

PRG009 Program check (fixed point divide) in the control program.

PRG010 Program check (decimal overflow) in the control program.

PRG011 Program check (decimal divide) in the control program.

PRG012 Program check (exponential overflow) in the control program.

PRG013 Program check (exponential underflow) in the control program.

PRG014 Program check (significance) in the control program.

PRG015 Program check (floating-point divide) in the control program.

PRG254 A translation specification exception has been received for a virtual machine that is not in Extended Control Mode.

PRG255 A PER interrupt has been received for a virtual machine that is running with PER disabled int its virtual PSW.

PSA001 Free storage is not available for the save areas.

PSA002 The System Restart key in the CPU console was depressed.

PSA003 Fatal DASD I/O error on paging device.

PTR001 A segment exception or a translation specification exception has occurred while executing a LRA (Load Real Address) instruction in DMKPTR.

PTR002 A program is attempting to unlock a page frame whose address exceeds the size of real storage.

PTR003 A program is attempting to unlock a real storage page frame whose CORTABLE entry is not flagged as locked.

PTR004 The lock count in the CORTABLE entry for the page frame being unlocked has been decremented to a value that is less than 0.

PTR006 Request to extend storage while extending.

PTR007 No storage available for extend.

PTR008 The CORTABLE entry on the free list points to a page currently active.

PTR009 The adjusted count of resident shared pages has fallen below zero.

PTR010 The adjusted count of resident reserved pages has fallen below zero.

QCN001 A HIO instruction attempting to halt a prepare/read to a logged on user terminal received a condition code of 3.

QCN002 A CP routine has attempted to initiate a read or write to a device whose RDEVTYPE field is invalid.

RPA001 The virtual address supplied to DMKRPAGT is outside of the virtual storage being referenced.

RPA002 The virtual address supplied to DMKPRAPT is outside of the virtual storage being referenced.

RPA003 User page count is negative.

SCH001 The adjusted count of users in the in-queue (interactive plus non-interactive) has fallen below zero.

TDK001 A program is attempting to deallocate a cylinder of T-disk space for which no cylinder allocation block (ALOCBLOK) exists.

TDK002 A program is attempting to deallocate a cylinder or cylinders of T-disk space that are not marked allocated.

TRC001 An erroneous call to TRACE was detected.

UDR001 The user directory module is looping, trying to read all of the UDIRBLOK page buffers from the directory device. Or, a directory containing over 10,816 users was loaded.

VAT001 A hardware page exception occurred, but the translation tables indicate either the segment is not available or the page table does not exist.

VDB001 The VDEVBLOK for the virtual device being released contains an unrecognizable device type.

VDB002 The 'Sysownd' list is in an invalid format.

VDB003 DASD link chain is invalid.

VIO001 The VMINST field in the user's VMBLOK issuing privileged I/O operation does not contain a recognizable I/O operation code.

VIO002 DMKSCNVU was unable to locate all of the virtual I/O control blocks for the virtual unit address associated with the interrupt previously unstacked.

VIO003 DMKIOS has returned an IOBLOK indicating a condition code of 2 was received from the Start I/O for the operation.

VSP001 DMKSCNVU was unable to locate all of the virtual I/O control blocks for the channel program that was previously executed. The virtual I/O configuration was destroyed.

APPENDIX A: VM/370 MODULE FORMAT

Every module for VM/370 is formatted in the following manner:

```
MOD      TITIE Card
         ISEQ 73, 80  Validate source seq.
```

Module Prologue

The prologue contains a heading for each of the topics listed in the order that they appear below, even if the topic does not apply to the given module. Topic headings start in column three and are followed by one blank line. The text beneath each heading should start in column ten. The required topics are as follows:

MODULE NAME - The actual name of the module (that is, the label of the START or first CSECT card)

FUNCTION - A brief (one or two sentence) description of the purpose of the module

ATTRIBUTES - A list of things such as whether or not the module is reentrant or serially reusable , resident or pageable, an how it is called (via an SVC, BALR, or GOTO).

ENTRY POINTS - A list of the name of each entry point, followed by a short explanation of the reason that the entry point is called. Also included is a list of all fields that are referenced or modified by an external routine.

ENTRY CONDITIONS - State any registers that must be loaded by the caller and what values they contain. If general registers contain parameters, state their symbolic values and their meanings.

EXIT CONDITIONS, NORMAL AND ERROR - State any of the caller's registers that are modified and what values they contain, if a meaningful condition code is set, and indicate if the module does not return to its caller.

CALLS TO OTHER ROUTINES - A list of any external routines or modules that are called. Include any exits to DISPATCH via a GOTO.

EXTERNAL REFERENCES - A list of any tables, control blocks or values that are referenced in this module but are defined elsewhere.

TABLES/WORKEAREAS - A list of any temporary work or scratch areas. For example, the use of the BALRSAVE area in PSA or the SAVEWRK areas in the standard SAVEAREA for any purpose other than normal register saving and restoring.

REGISTER USAGE - A list of the usage for each general register that has a consistent purpose through the module. Also list any unused registers and any registers used only for scratch or intermediate values.

NOTES - Include any comments not relevant in another section such as descriptions or unusual coding techniques, formulas, or release dependencies.

OPERATION - Include here a brief description, in general terms, of the logical steps performed by the module. The steps may be numbered so that branches and loops may be easily described.

The prologue is used for most of the system's modules. However, in some cases a module consists of a collection of relatively unrelated subroutines that have been grouped together but contain little common code. In this case, the prologue page contains only the MODULE NAME section and a list of the subroutines that the module contains. Preceding each subroutine, there is a prologue exactly as described above, except that the heading MODULE NAME is replaced by the heading SUBROUTINE NAME.

If CSECTS are large (more than 4096 bytes) it is preferable that code and referenced data reside in the same page. The proper use of LTORG statements will accomplish this.

```
         EJECT

label    CSECT

         ENTRY Statements

         EXTRN Statements

         USING Statements

         Source Code

         Constants   (See Note)

         LTORG   (See Note)

         Working Storage (See Note)

         System DSECTS and EQUATES

         END
```

MOD is the 3 character module name without the component code.

Label is the formal module name consisting of the component code and the Module Name.

Note: Where possible, these are in the same storage page with the instructions.

APPENDIX B: VM/370 CODING CONVENTIONS

1. FORMAT:

col. 1 - labels
col. 10 - op code
col. 16 - operands
col. 31, 36, 41, etc. - comments (See Item 2.)

2. COMMENT:

Approximately 75 per cent of the source code contains comments. Sections of code performing distinct functions are separated from each other by a comment section.

3. CONSTANTS:

Constants follow the executable code and precede the copy files and/or macros which contain dsects or system equates. Constants are defined in a section followed by a section containing initialized working storage, followed by working storage. Each of these sections are identified by a comment. Where possible for a module that is greater than a page, constants and working storage are within the same page in which they are referenced.

4. No program modifies its own instructions during execution.

5. No program uses its own unlabeled instructions as data.

6. REGISTER USAGE: For CP, in general

Register		Use
6	-	RCHBLOK, VCHBLOK
7	-	RCUBLOK, VCUBLOK
8	-	RDEVBLOK,VDEVBLCK
10	-	IOBLOK
11	-	VMBLOK
12	-	Base register for modules called via SVC
13	-	SAVEAREA for modules called via SVC
14	-	Return linkage for modules called via BALR
15	-	Base address for modules called via BALR

For Virtual to Real address translation:
1 - Virtual Address
2 - Real Address

7. When describing an area of storage in mainline code, a copy file, or a macro, DSECT is issued containing DS instructions.

8. Meaningful names are used instead of self-defining terms for example 5,X'02',C'I') to represent a quantity (for example absolute address, offset, length, register, etc.). All labels, displacements, and values are symbolic. All bits should be symbolic and defined by EQU. For example:

```
VMSTATUS   EQU   X'02'
```

To set a bit, use:

```
OI    BYTE,BIT
```

Where BYTE = name of field, BIT is an EQU symbol.

To reset a bit, use:

```
NI    BYTE,255-BIT
```

To set multiple bits, use:

```
IO    BYTE,BIT1+BIT2
```

etc....

All registers are referred to as:

```
R0, R1, ......., R15.
```

All lengths of fields or blocks are symbolic, i.e. length of VMBLOK is:

```
VMBLOKSZ  EQU  *-VMBLCK
```

9. Avoid absolute relative addressing in branches and data references, (that is, location counter value (*) or symbolic label plus or minus a self-defining term used to form either a displacement or offset).

10. When using a single operation to reference multiple values, specify each value referenced, for example:

```
LM  R2,R4,CONT    SET R2=CON1
                  SET R3=CON2
                  SET R4=CON3
.
.
.
CON1    DC    F'1'
CON2    DC    F'2'
CON3    DC    F'3'
```

11. Do not use PRINT NOGEN.

12. Module Names: Control Section Names and External References are as follows:

    Control Secticn or Module Name
    The first three letters of the name are the assigned component code.

    Example: DMK

    The next three letters of the Module Name identify the module and must be unique.

    Example: DSP

    This three letter unique module identifier is the label of the TITLE card.

    Each entry point or external reference must be prefixed by the six letter unique identifier of the module.

    Example: DMKDSPCH

13. TITLE Card:

    DSP TITLE 'DMKDSP VM/370 DISPATCHER VERSION 1 LEVEL 0'

14. PTF Card Example:

    CP/CMS: PUNCH 'xxxxxxxx APPLIED'

    Where xxxxxxxx = APAR Number Response

15. Error messages:

    There should not be any insertions into the message at execution time and the length of the message should be resolved by the assembler. If insertions must be made, the message must be assembled as different DC statements, and the insert positions are to be individually labeled.

16. For all RX instructions use ',' to specify the base register when indexing is not being used, i.e.

```
L    R2,AB(,R4)
```

APPENDIX C. CP EQUATE SYMBOLS

CP DEVICE CLASSES, TYPES, MODELS AND FEATURES

```
CLASTERM EQU   X'80'          Terminal Device Class
TYP2700  EQU   X'40'          2700 Bisync line
TYP2955  EQU   TYP2700        2955 Communications Line
TYPTELE2 EQU   X'20'          Telegraph Terminal Control Type II
TYPTTY   EQU   X'20'          Teletype Terminal
TYPIBM1  EQU   X'10'          IBM Terminal Control Type I
TYP2741  EQU   X'18'          2741 Communications Terminal
TYP1050  EQU   X'14'          1050 Communications Terminal
TYPUNDEF EQU   X'1C'          Terminal device type is undefined
TYP3210  EQU   X'00'          3210 Console
TYP3215  EQU   TYP3210        3215 Console
TYP2150  EQU   TYP3210        2150 Console
TYP1052  EQU   TYP3210        1052 Console

CLASGRAF EQU   X'40'          Graphics Device Class
TYP2250  EQU   X'80'          2250 Display Unit
TYP2260  EQU   X'40'          2260 Display Station
TYP2265  EQU   X'20'          2265 Display Station
TYP3066  EQU   X'10'          3066 Console
TYP1053  EQU   X'08'          1053 Printer
TYP3277  EQU   X'04'          3277 Display Station
TYP3284  EQU   X'02'          3284 Printer
TYP3286  EQU   TYP3284        3286 Printer

CLASURI  EQU   X'20'          Unit Record Input Device Class
TYPRDR   EQU   X'80'          Card Reader
TYP2501  EQU   X'81'          2501 Card Reader
TYP2540R EQU   X'82'          2540 Card Reader
TYP3505  EQU   X'84'          3505 Card Reader
TYP1442R EQU   X'88'          1442 Card Reader/Punch
TYP2520R EQU   X'90'          2520 Card Reader/Punch
TYPTIMER EQU   X'40'          Timer
TYPTR    EQU   X'20'          Tape Reader
TYP2495  EQU   X'21'          2495 Magnetic Tape Cartridge Reader
TYP2671  EQU   X'22'          2671 Paper Tape Reader
TYP1017  EQU   X'24'          1017 Paper Tape Reader

CLASURO  EQU   X'10'          Unit Record Output Device Class
TYPPUN   EQU   X'80'          Card Punch Device
TYP2540P EQU   X'82'          2540 Card Punch
TYP3525  EQU   X'84'          3525 Card Punch
TYP1442P EQU   X'88'          1442 Card Punch
TYP2520P EQU   X'90'          2520 Card Punch
```

```
  TYPPRT   EQU   X'40'        Printer
  TYP1403  EQU   X'41'        1403 Printer
  TYP3211  EQU   X'42'        3211 Printer
  TYP1443  EQU   X'44'        1443 Printer
  TYPTP    EQU   X'20'        Tape Punch
  TYP1018  EQU   X'24'        1018 Paper Tape Punch
  FTRUCS   EQU   X'01'        UCS Feature

  CLASTAPE EQU   X'08'        Magnetic Tape Device Class Units
  TYP2401  EQU   X'80'        2401 Tape Drive
  TYP2415  EQU   X'40'        2415 Tape Drive
  TYP2420  EQU   X'20'        2420 Tape Drive
  TYP3410  EQU   X'08'        3410 Tape Drive
  TYP3420  EQU   X'10'        3420 Tape Drive
  FTR7TRK  EQU   X'80'        7-Track Feature
  FTRDLDNS EQU   X'40'        Dual Density Feature
  FTRTRANS EQU   X'20'        Translate Feature
  FTRDCONV EQU   X'10'        Data Conversion Feature

  CLASDASD EQU   X'04'        Direct Access Storage Device Class
  TYP2311  EQU   X'80'        2311 Disk Storage Drive
  TYP2314  EQU   X'40'        2314 Disk Storage Facility
  TYP2319  EQU   TYP2314      2319 Disk Storage Facility
  TYP2321  EQU   X'20'        2321 Data Cell Drive
  TYP3330  EQU   X'10'        3330 Disk Storage Facility
  TYP2301  EQU   X'08'        2301 Parallel Drum
  TYP2303  EQU   X'04'        2303 Serial Drum
  TYP2305  EQU   X'02'        2305 Fixed Head Storage Device
  FTR2311T EQU   X'20'        (= VDEV231T)  Top half of 2314 used as 2311
  FTR2311B EQU   X'10'        (= VDEV231B)  Bottom Half of 2314 used as 2311

| FTRRSRL  EQU   X'02'        Reserve/Release are valid CCW op codes
|                             (control unit has a 2-channel switch)


  CLASSPEC EQU   X'02'        Special Devices
  TYPCTCA  EQU   X'80'        Channel to Channel Adapter
```

MACHINE USAGE

```
Bits defined in standard extended PSW
EXTMODE  EQU   X'08'          Bit 12 - Extended Mode
MCHEK    EQU   X'04'          Bit 13 - Machine check enabled
WAIT     EQU   X'02'          Bit 14 - Wait state
PROBMODE EQU   X'01'          Bit 15 - Problem state

Bits defined in extended PSW
PERMODE  EQU   X'40'          Bit 01 - PER enabled
MODE31   EQU   X'08'          Bit 04 - 31 bit mode addressing
TRANMODE EQU   X'04'          Bit 05 - Translate mode
IOMASK   EQU   X'02'          Bit 06 - Summary I/O Mask
EXTMASK  EQU   X'01'          Bit 07 - Summary external mask

Bits defined in channel status word - CSW
ATTN     EQU   X'80'          Bit 32 - Attention
SM       EQU   X'40'          Bit 33 - Status modifier
CUE      EQU   X'20'          Bit 34 - Control unit end
BUSY     EQU   X'10'          Bit 35 - Busy
CE       EQU   X'08'          Bit 36 - Channel end
DE       EQU   X'04'          Bit 37 - Device end
UC       EQU   X'02'          Bit 38 - Unit check
UE       EQU   X'01'          Bit 39 - Unit exception

PCI      EQU   X'80'          Bit 40 - Program-control interrupt
IL       EQU   X'40'          Bit 41 - Incorrect length
PRGC     EQU   X'20'          Bit 42 - Program check
PRTC     EQU   X'10'          Bit 43 - Protection check
CDC      EQU   X'08'          Bit 44 - Channel data check
CCC      EQU   X'04'          Bit 45 - Channel control check
IFCC     EQU   X'02'          Bit 46 - Interface control check
CHC      EQU   X'01'          Bit 47 - Chaining check

Bits defined in channel command word - CCW
CD       EQU   X'80'          Bit 32 - Chain data
CC       EQU   X'40'          Bit 33 - Command chain
SILI     EQU   X'20'          Bit 34 - Suppress incorrect length indication
SKIP     EQU   X'10'          Bit 35 - Suppress data transfer
PCIF     EQU   X'08'          Bit 36 - Program-control interrupt fetch
IDA      EQU   X'04'          Bit 37 - Indirect data address

Bits defined in sense byte 0 -- Common to  most devices
CMDREJ   EQU   X'80'          Bit 0 - Command reject
INTREQ   EQU   X'40'          Bit 1 - Intervention required
BUSOUT   EQU   X'20'          Bit 2 - BUS out
EQCHK    EQU   X'10'          Bit 3 - Equipment check
DATACHK  EQU   X'08'          Bit 4 - Data check
```

EXTENDED CONTROL REGISTERS

```
Bits defined in Control register 0
                   Byte 0
BLKMPX   EQU    X'80'          Bit 00 - Enable block multiplexing
SSMSUPP  EQU    X'40'          Bit 01 - Enable SSM suppression
                   Byte 1
PAGE4K   EQU    X'80'          Bit 08 - Use 4K pages
PAGE2K   EQU    X'40'          Bit 09 - Use 2K pages
SEG1M    EQU    X'08'          Bit 12 - Use 1M segments
                   Byte 2
CKCMASK  EQU    X'08'          Bit 20 - Mask on clock comparator intercept
CPTMASK  EQU    X'04'          Bit 21 - Mask on CPU timer intercept
                   Byte 3
INTMASK  EQU    X'80'          Bit 24 - Mask on interval timer intercept
KEYMASK  EQU    X'40'          Bit 25 - Mask on operator key intercept
SIGMASK  EQU    X'20'          Bit 26 - Mask on external signals 2-7

Bits defined in Control register 9
                   Byte 0
PERSUBR  EQU    X'80'          Bit 00 - Monitor successful branches
PERIFET  EQU    X'40'          Bit 01 - Monitor instruction fetches
PERSALT  EQU    X'20'          Bit 02 - Monitor storage alteration
PERGPRS  EQU    X'10'          Bit 03 - Monitor register alteration

Bits defined in Control register 14
                   Byte 0
HARDSTOP EQU    X'80'          Bit 00 - Check stop control
SYNCLOG  EQU    X'40'          Bit 01 - Synchronous logout control
IOLOG    EQU    X'20'          Bit 02 - I/O logout control
RECOVRPT EQU    X'08'          Bit 04 - Recovery report mask
CONFGRPT EQU    X'04'          Bit 05 - Configuration report mask
DAMAGRPT EQU    X'02'          Bit 06 - External damage report mask
WARNGRPT EQU    X'01'          Bit 07 - Warning condition report mask
                   Byte 1
ASYNELOG EQU    X'80'          Bit 08 - Asynchronous extended logout control
ASYNFLOG EQU    X'40'          Bit 09 - Asynchronous fixed logout control
```

CONTROL PROGRAM USAGE

```
Bits defined for TRANS macro
BRING    EQU   X'80'          Bring requested page
DEFER    EQU   X'40'          Defer execution until page in  storage
LOCK     EQU   X'20'          Lock page for I/O operation
IOERETN  EQU   X'10'          Return I/O errors to caller
SYSTEM   EQU   X'08'          Call to DMKPTRAN for system virtual machine space

Bits defined for terminal I/O
ERRMSG   EQU   X'04'*256      Control program error message
PRIORITY EQU   X'02'*256      Queue and start this message immediately
VMGENIO  EQU   X'01'*256      Virtual machine generated I/O request
LOGDROP  EQU   X'80'          LOGOUT and drop line after output message
LOGHOLD  EQU   X'40'          LOGOUT & hold line after output message
NORET    EQU   X'20'          Return immediately after call
DFRET    EQU   X'10'          FRET Buffer after write
NOAUTO   EQU   X'08'          No automatic carriage return
EDIT     EQU   X'08'          Edit input for corrections
ALARM    EQU   X'04'          Sound the alarm
UCASE    EQU   X'04'          Translate input to upper case
OPERATOR EQU   X'02'          Message for operator
NOTIME   EQU   X'01'          Do not time stamp message
```

APPENDIX D. DASD RECORD FORMATS

RECORD 0, 8 BYTES (PAGE BIT MAP)

Used to flag pages that are in use or have bad recording area. Devices that do no use all 64 bits (64 pages per cylinder) have the unused bits turned on.

Examples

RECORD 0 CYLINDER 0 Only

32 Pages/cylinder 2314,2319

```
*|E0 00 00 00 FF FF FF FF|
   |
11100000
```

57 pages/cylinder 3330

```
 |E0 00 00 00 00 00 00 7F|
```

24 pages/cylinder 2305

```
 |E0 00 00 FF FF FF FF FF|
```

* The first three pages of cylinder 0 are always flagged in use, since they are used by CP. On all other cylinders, the first byte hex '00' unless the disk area is flagged bad. Record 0 of all tracks cther than track 0 is initialized to hex '0C'.

All Page Records, 4096 Bytes Each

2314 and 2319	32 pages/cylinder
3330 series	57 pages/cylinder
2305	24 pages/cylinder

Cylinder 0 contains less pages because this area is used by CP.

RECORD 1 (24 BYTES)

IPL record - Puts system into wait state if storage device is IPLed.

```
|00020000 0000000C 03000000 20000000 00000000 00000000|
```

RECORD 2, 4096 BYTES

Check point record - this is the CHECKPOINT program load at CP IPL time to retrieve and save control information for a warm start.

RECORD 3

4 byte key of VOL1
80 byte data record

```
   Key
┌───────┐
│ VOL1  │
└───────┘
```

Record

```
Bytes ┌────────┬────────┬────────┬────────┬────────┐
 1-20 │E5D6D3F1 xx────────>xxF000 00000005 00C0C000│
      │                                            │
21-40 │0040───────────────────────────────────>40│
      │                                            │
41-60 │4000──────>00C3D7 F3F7F040 40404040 40──>40│
      │                                            │
61-80 │40─────────────────────────────────────>40│
      └────────┴────────┴────────┴────────┴────────┘
```

Where:
xx->xx is a 6 byte label

Bytes 13-16 is a pointer to the VTOC

Bytes 46-50 identify the system

Bytes 52-55 is a pointer to the active directory

RECORD 4

1024 bytes Track 0 Cylinder 0

Allocation byte map - used to identify Cylinder 1 usage. Each byte identifies one cylinder.

```
 0                           ┌─> all 0<─┐
┌─────────┬──────────────────┬─+──────────+┐
│00000100 040200───>FF 0000────>0000│
└─────────┴───────────+─┴────────────────┘
                      *
```

* FF dfines the last cylinder + 1 that can be allocated. This varies depending on the device.

00 = temporary
01 = permanent
02 = T-disk
04 = directory

RECORD 5

44 bytes key Track 0, Cylinder 0 96 bytes data area

Format 4 OS DSCB type label - used to be compatible with OS.

```
┌────────────┐    ┌──────────────────────┐
│ 04────>04│    │   FORMAT 4 LABEL     │
└────────────┘    └──────────────────────┘
    44 Key            96 Byte Data
```

RECORD 6

44 bytes key Track 0, Cylinder 0 96 bytes data area

Format 5 OS DSCB type lable for ccmpatibility with OS.

```
┌───┬──┬──┬──┬───┐    ┌─────────────────────┐
│ 05│05│05│05│00 │    │ CS FORMAT 5 LABEL  │
└───┴──┴──┴──┴───┘    └─────────────────────┘
  44 Byte Key             96 Byte Data Area
```

RECORD F3

4096 bytes - 1 page, track 0 or track 1

F3 Record is reserved for CPsytem use. Referred to as filler record.

RECORD F4

1624 bytes, Track 1 (2314, 2319 cnly)

F4 used only on 2314 and 2319 devices to align Record 4 in proper position on track.

RECORD 4

824 bytes track 1, cylinder 0 =2314, 2319 only)

First segment of Record 4 to be used for paging.

2314 RECORD LAYOUT

CYLINDER 0, TRACK 0

```
 R0   R1   R2   Key  R3    R4   Key    R5   Key    R6

┌────┬──┬─────┬─┬─────┬─────┬──┬─────┬──┬─────┬─────┐
│Page│I │Check│V│VOL1 │Alloc│  │Format│  │Format│     │
│Bit │P │Point│O│Label│Byte │  │ 4    │  │ 5    │     │
│Map │L │     │L│     │Map  │  │      │  │      │     │
│    │  │     │1│     │     │  │      │  │      │     │
│  8 │24│ 4096│4│  80 │ 1024│44│   96 │44│  96  │     │
└────┴──┴─────┴─┴─────┴─────┴──┴──────┴──┴──────┴─────┘
```

Cylinder 0, Track 1

```
   R0          RF3        RF4        R4
┌───────┐   ┌───────┐  ┌───────┐  ┌───────┐
│       │   │1 PAGE │  │FILLER │  │       │
│       │───│       │──│       │──│       │
│   8   │   │ 4096  │  │ 1624  │  │824    │
└───────┘   └───────┘  └───────┘  └───────┘
```

ALL CYLINDERS EXCEPT 0, TRACK 0

```
   R0          R1          R2
┌────────┐   ┌────────┐   ┌─────────┐
│Page    │   │        │   │         │
│Bit Map │───│        │───│         │
│   8    │   │  4096  │   │  2472   │
└────────┘   └────────┘   └─────────┘
```

These records appear as above formats if cylinder is 0.

Track 1

```
   R0          R2           R3          R4
┌───────┐   ┌───────┐   ┌───────┐   ┌───────┐
│       │───│       │───│       │───│       │
│   8   │   │ 1624  │   │ 4096  │   │  824  │
└───────┘   └───────┘   └───────┘   └───────┘
```

Track 2

```
   R0          R4          R5
┌───────┐   ┌───────┐   ┌───────┐
│       │───│       │───│       │
│   8   │   │ 3272  │   │ 3296  │
└───────┘   └───────┘   └───────┘
```

Track 3

```
   R0          R5          R6          R7
┌───────┐   ┌───────┐   ┌───────┐   ┌───────┐
│       │───│       │───│       │───│       │
│   8   │   │  800  │   │ 4096  │   │1648   │
└───────┘   └───────┘   └───────┘   └───────┘
```

Track 4

```
   R0          R7          R8
┌───────┐   ┌───────┐   ┌───────┐
│       │───│       │───│       │
│   8   │   │ 2448  │   │ 4096  │
└───────┘   └───────┘   └───────┘
```

Note: Track 0 to 4 are repeated for tracks 5 to 9 =R9_R16), 10 to 14, =R17_R24), and 15 to 19 =R25_R32). The last record is R32.

3330 SERIES RECORD LAYOUT

CYLINDER 0, TRACK 0

R0	R1	R2	Key	R3	R4	Key	R5	Key	R6	RF3	
Page Bit Map 8	I P L 24	Check Point 4096	V O L 1 4	VOL1 Label 80	Byte Map 1024	44	Format 4 96	44	Format 5 96	1 Page 4096	

ANY CYLINDER EXCEPT 0

Track 0

```
     R0            R1             R2              R3
┌─────────┐   ┌─────────┐   ┌──────────┐   ┌─────────┐
│Page     │   │         │   │          │   │         │
│Bit  Map │───│         │───│          │───│         │
│         │   │         │   │          │   │         │
│    8    │   │  4096   │   │  4096    │   │  4096   │
└─────────┘   └─────────┘   └──────────┘   └─────────┘
                               │
                               │
                               │
                               V
```

Track 18

```
     R0           R55             R56            R57
┌─────────┐   ┌─────────┐   ┌─────────┐   ┌─────────┐
│         │───│         │───│         │───│         │
│         │   │         │   │         │   │         │
│    8    │   │  4096   │   │  4096   │   │  4096   │
└─────────┘   └─────────┘   └─────────┘   └─────────┘
```

2305 MODEL 1 and MODEL 2

CYLINDER 0, TRACK 0

```
  R0   R1   R2   Key  R3    R4   Key    R5   Key    R6    RF3
┌────┬───┬─────┬─┬─────┬────┬───┬───────┬───┬───────┬────┬──┐
|Page|I  |Check|V|VOL1 |Byte|   |Format |   |Format | 1  |  |
|Bit |P  |Point|O|Label|Map |   | 4     |   | 5     |Page|  |
|Map |L  |     |L|     |    |   |       |   |       |    |  |
|    |   |     |1|     |    |   |       |   |       |    |  |
|  8 |24 | 4096|4|  80 |1024|44 |   96  |44 |   96  |4096|  |
└────┴───┴─────┴─┴─────┴────┴───┴───────┴───┴───────┴────┴──┘
```

ANY CYLINDER EXCEPT 0

Track 0

```
    R0          R1           R2          R3
┌────────┐  ┌────────┐  ┌────────┐  ┌───────┐
|Page    |  |        |  |        |  |       |
|Bit Map |──|        |──|        |──|       |
|        |  |        |  |        |  |       |
|   8    |  | 4096   |  | 4096   |  | 4096  |
└────────┘  └────────┘  └────────┘  └───────┘
                      |
Track 7               |
                      |
    R0         R22    V     R23         R24
┌────────┐  ┌────────┐  ┌──────────┐  ┌────────┐
|        |──|        |──|          |──|        |
|        |  |        |  |          |  |        |
|  8     |  | 4096   |  | 4096     |  | 4096   |
└────────┘  └────────┘  └──────────┘  └────────┘
```

Ken Jones

IBM Technical Newsletter

This Newsletter No.	SN20-2624
Date	August 15, 1973
Base Publication No.	SY20-0880-1
File No.	S370-37 (VM/370 Release 1 PLC 9)
Previous Newsletters	None

IBM Virtual Machine Facility/370: Control Program (CP) Program Logic

This Technical Newsletter, a part of Release 1 PLC 9 of IBM Virtual Machine Facility/370 provides replacement pages for your publication. These replacement pages remain in effect for subsequent VM/370 releases unless specifically altered. Pages to be removed and/or inserted are listed below.

Title page,2
Preface
Summary of
  Amendments
Contents
15-22
25,26
33,34,34.1
37-40
47-52,52.1-.2
57,58
69-84,84.1-.5
97,98,98.1
105-114,114.1-.2
117-118
121,122
145,146
155-164
169-172
185-190
195,196
213-220
239-242
259-264,264.1-.2
272.1-272.2 (new)
273-276,276.1
277-278
287-290,290.1
291-300
305-306,306.1
307-310,310.1-.2
311-316
329-332
351,352,352.1-.6
361-364
369-376,376.1
387,388,388.1
397-400
403-406
423-440
449-462,462.1
463-464
467-472
475-476
483,484,484.1
487-492,492.1
495,496,496.1
501,502,502.1
511,512
517-522,522.1-.4
539-540,540.1-.6
543-544,544.1
565-568
571-576,576.1
579-590
593-595

Summary of Amendments

This Technical Newsletter contains changes that reflect VM/370 support for:

- The IBM System/370 Model 168
- The IBM 2860 Selector Channel
- The IBM 2870 Multiplexor Cannel
- The IBM 2880 Block Multiplexor Channel
- The IBM 2305 Fixed Head Storage, Model 1
- User Accounting Option
- Virtual Console Spooling

See the Summary of Amendments page for further details.

Note: Please file this cover letter at the back of your publication to provide a record of changes.

IBM Corporation, VM/370 Publications, 24 New England Executive Park, Burlington, Massachusetts 01803

Printed in U.S.A.

Trim Along This Line

READER'S COMMENTS

Title: IBM Virtual Machine
Facility/370:
Control Program (CP)
Program Logic

Order No. SY20-0880-1

Please check or fill in the items; adding explanations/comments in the space provided.

Which of the following terms best describes your job?

□ Programmer
□ Manager
□ Operator
□ Instructor
□ Systems Analyst
□ Engineer
□ Mathematician
□ Student/Trainee
□ Customer Engineer
□ Systems Engineer
□ Sales Representative
□ Other (explain below)

Does your installation subscribe to the SL/SS? □ Yes □ No

How did you use this publication?
□ As an introduction
□ As a reference manual
□ For another purpose (explain) ________________________
□ As a text (student)
□ As a text (instructor)

Did you find the material easy to read and understand? □ Yes □ No (explain below)

Did you find the material organized for convenient use? □ Yes □ No (explain below)

Specific criticisms (explain below)
Clarifications on pages ________________________
Additions on pages ________________________
Deletions on pages ________________________
Errors on pages ________________________

Explanations and other comments:

Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

SY20-0880-1

Trim Along This Line

YOUR COMMENTS PLEASE . . .

This manual is one of a series which serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your comments on the back of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

FOLD FOLD

FIRST CLASS
PERMIT NO. 172
BURLINGTON, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

IBM CORPORATION

VM/370 Publications
24 New England Executive Park
Burlington, Massachusetts 01803

FOLD FOLD

IBM VM/370: CP Program Logic Printed in U.S.A. SY20-0880-1

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

SY20-0880-1

IBM VM/370: CP Program Logic   Printed in U.S.A.   SY20-0880-1

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)