Rank Xerox

Universal Time-Sharing System (UTS)

Sigma 6/7/9 Computers

Basic Control and Basic I/O Technical Manual

90 19 98A

Universal Time-Sharing System (UTS)

Sigma 6/7/9 Computers

Basic Control and Basic I/O
Technical Manual

First Edition

90 19 85A

February 1973

Price: $5.25

NOTICE

This publication documents the basic control and basic I/O routines that operate under the Universal Time-Sharing System (UTS) for Sigma 6/7/9 computers. With the exception of Section DA (Device I/O subsection), all material in this manual reflects the C01 version of the UTS operating system. Section DA reflects the B01 version of UTS.

RELATED PUBLICATIONS

Title	Publication No.
UTS Overview and Index Technical Manual[†]	90 19 84
UTS System and Memory Management Technical Manual	90 19 86
UTS Symbiont and Job Management Technical Manual	90 19 87
UTS Operator Communication and Monitor Services Technical Manual	90 19 88
UTS File Management Technical Manual[†]	90 19 89
UTS Reliability and Maintainability Technical Manual	90 19 90
UTS Interrupt Driven Tasks Technical Manual[†]	90 19 91
UTS Initialization and Recovery Technical Manual	90 19 92
UTS Command Processors Technical Manual	90 19 93
UTS System Processors Technical Manual	90 19 94
UTS Data Bases Technical Manual	90 19 95

[†]Not published as of the publication date given on the title page of this manual. Refer to the PAL Manual for current availability.

CONTENTS

UTS TECHNICAL MANUAL

ID

BASIC CONTROL - TRAPS AND INTERRUPTS

PURPOSE

The primary function of the Monitor trap routines is to establish a means by which a user program may communicate with the Monitor and vice-versa. For example, the user may request the Monitor (via CAL instructions) to perform such operations as building files, retrieving data, setting interrupts, loading program segments, and providing debugging diagnostics. In addition to servicing these requests, the Monitor may communicate to the user that he is attempting to execute non-allowed operations, or perform unimplemented instructions, and the like. Trap and interrupt routines also are activated by hardware error conditions which may result in a user abort or system recovery.

The function of the interrupt routines is to provide service to the monitor itself for processes which are not user associated, e.g., I/O interrupt processing, symbiont activity, polling of COC lines, etc. The modules discussed in this chapter provide the means by which this two-way communication is effected. Basically the mechanism is one of analyzing and servicing the hardware traps and interrupts when they occur. (This section discusses only the processing of "internal" interrupts, i.e., clock interrupts, I/O interrupts, etc. The use and processing of external interrupt (e.g. X'60' and X'61') is discussed in COCINIT, section DC).

OVERVIEW

A trap or an interrupt occurs when conditions at the hardware level cause what may be considered an unconditional hardware "branch". A number of conditions may cause this branch to occur; e.g., an attempt to execute an unimplemented instruction, to reference a nonexistent memory location, a hardware error, or a value of zero in a clock interrupt counter. In addition, four instructions (CAL1, CAL2, CAL3, and CAL4) cause a trap condition and thus the hardware branch when encountered during the execution of a program. Hardware errors (Section CD) also result in trap conditions and cause this unconditional hardware "branch".

When the branch takes place, control is transferred to one of the pre-defined memory locations X'40' through X'5D', referred to collectively as the trap and interrupt locations. Each of these locations contains an instruction stored there at system initialization by INITIAL. The execution of these instructions is the means by which communication is established between the Monitor and a user program or between the Monitor and the hardware or operator. The UTS modules involved in establishing the communication are

TABLES, ENTRY, CALPROC, ALTCP, IOQ, PFSR, CLOCK4, S9TRAPS.

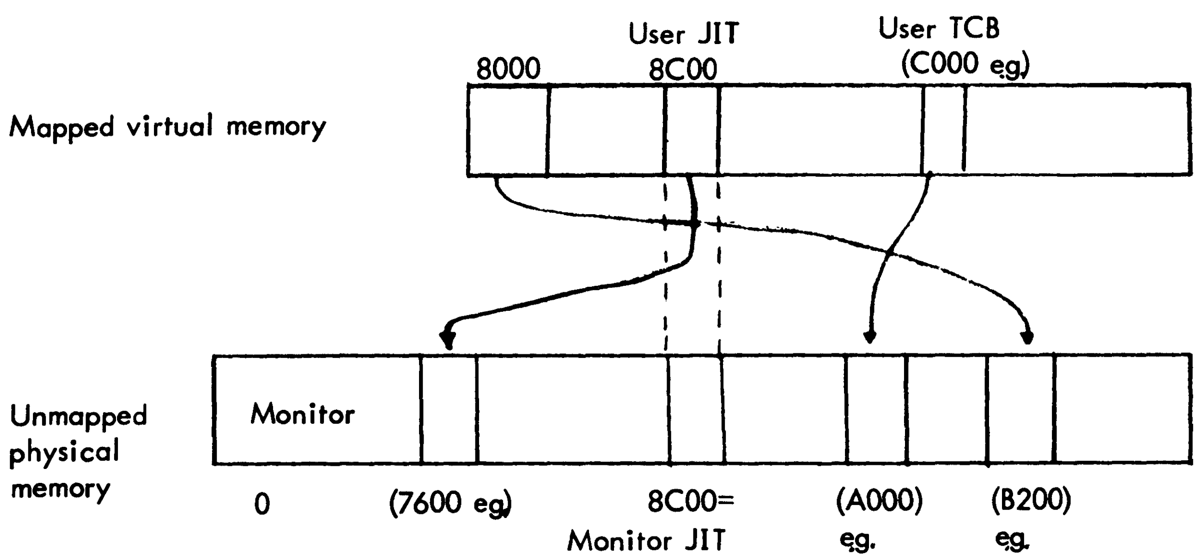
When writing his program, the UTS user requests Monitor services by coding a Monitor procedure within his program.

When a procedure call is encountered while the program is being assembled or compiled, the processor responds by retrieving a symbolic calling sequence from the procedure library, modifying it according to the parameters specified by the user, and inserting this symbolic code into the program. Typically this symbolic code begins with a CAL1 instruction and continues with a variable number of words containing the user's parameter information, referred to collectively as the Function Parameter Table (FPT). When a CAL1 instruction is encountered at execution time, the hardware branches to trap location X'48'. The instruction at X'48' is an exchange program status doubleword (XPSD) which, when executed, transfers control to a subroutine in ENTRY. ENTRY saves a 19 word environment (2 word PSD, odd word, 16 registers) and branches to CALPROC where decoding of the CAL begins.

The XPSDs in the interrupt locations do not transfer control to ENTRY but go directly to the appropriate interrupt routine.

TEMPSTACKS

In UTS there are three levels of tempstacks involved in CAL and trap processing; the monitor tempstack (unmapped JIT), the users monitor tempstack (mapped JIT) and the users tempstack (mapped user TCB). The users tempstack enters the picture only if an illegal trap occurs for which the user has requested trap control.



UTS monitor routines reference the monitor tempstack via the stack pointer doubleword (SPD)* named TSTACK. The SPD and tempstack are located in JIT. A crucial design feature of UTS is that user JITs have a fixed virtual address which is the same as the real physical address of the monitor JIT. Thus all monitor routines simply reference TSTACK and the setting of the map bit in the current program status doubleword determines which JIT is affected. When a trap occurs, the 19 word environment is pushed into the stack in the JIT in use at the time of the trap. Interrupts always push into the unmapped JIT even if the process interrupted is mapped.

THE USER TEMPSTACK

If the user has specified that he wants to process traps when they occur, the Monitor saves the user's PSD, general registers, and the location of the trap in the user tempstack before giving control to the user program. These 19 words of information are saved on a doubleword boundary in the user tempstack.

The address of the user's tempstack and its size are saved in the first two words of the Task Control Block (TCB). A description of how the Monitor uses these TCB entries to save the PSD, registers and trap location is given in the discussion of ALTCP, subroutine STKTOTMP.

SPECIAL CAL1 PROCESSES

For a CAL1, 1 which is executed itself, i.e. not executed as the result of execution of an EXU instruction, a special accelerated path of code is provided in the ENTRY module starting at symbolic location CAL1IN2. This code performs the jobs of placing the PSD and registers into the stack, switching the clock to overhead, and establishing the FPT and DCB addresses before entering the CALPROC module at symbolic location CAL1IN3.

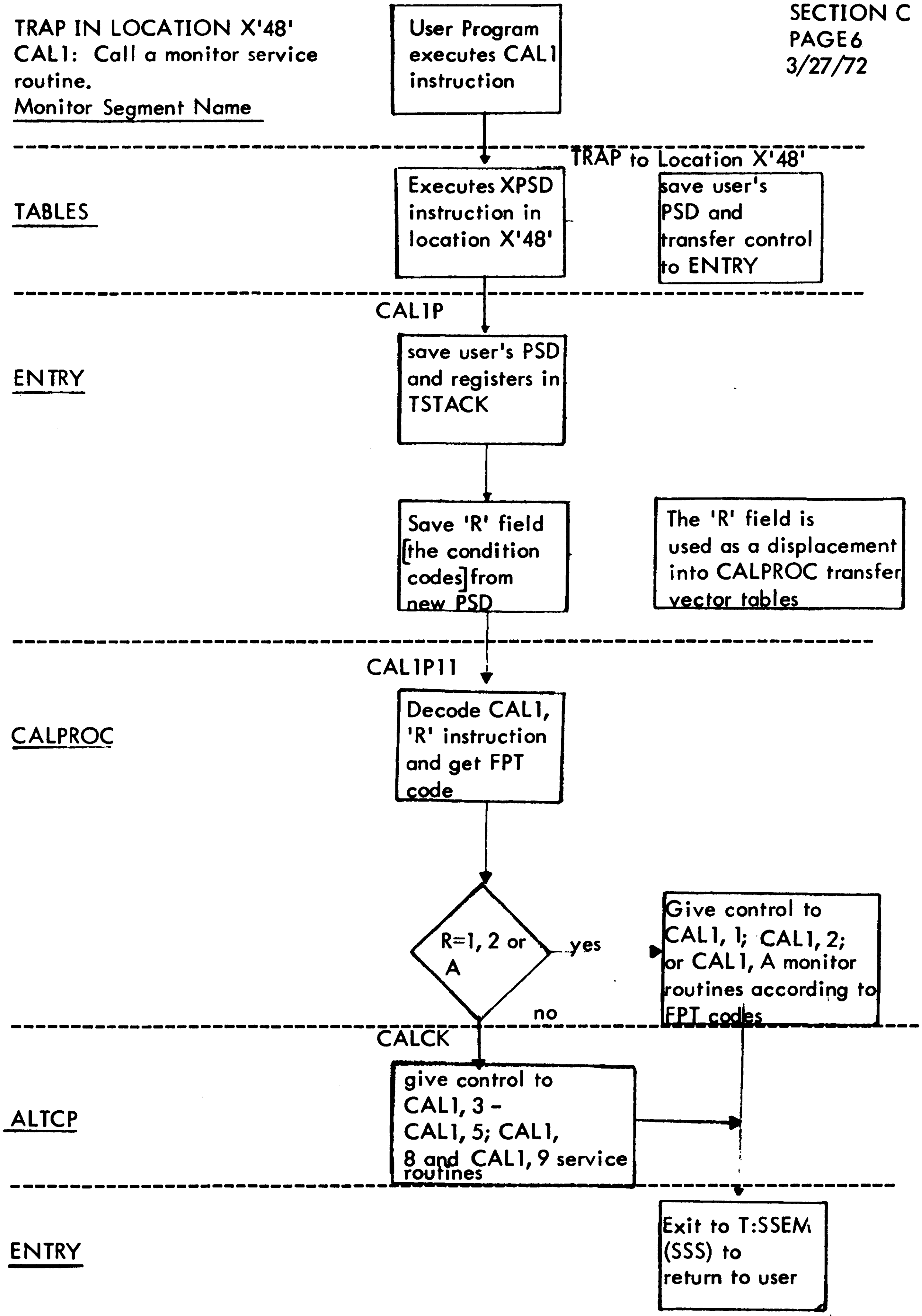
---

*See XDS Sigma 7 Computer Reference Manual for a more detailed description of a stack-pointer doubleword.

TRAPS	XPSD's assembled in INITIAL, stored in 40-61 at system initialization	XPSD double doublewords in TABLES	Trap interrupt entry points. Routine which contains the entry point is named in parentheses
40 Non-allowed op	XPSD NOPPSD	NOPPSD	NOPPGM
41 Unimplemented Instr.	XPSD UNIMPPSD	UNIMPPSD	UNIMP
42 Stack Trap	XPSD STKLPSD	STKLPSD	STKOVF
43 Fixed Overflow	XPSD FIXOVPSD	FIXOVPSD	FIXFLT
44 Floating Point Fault	XPSD FLTFPSD	FLTFPSD	FLTFLT (ENTRY) ALTCP
45 Decimal Fault	XPSD DECPSD	DECPSD	DECFLT
46 Watchdog Timer	XPSD WDOGPSD	WDOGPSD	WDOGPGM (TABLES)
48 CAL1	XPSD CAL1PSD		CAL1P (ENTRY) CALPROC CAL1, 1 or 2 SERVICE
49 CAL2	XPSD CAL2PSD		CAL2XXX ALTCP MODULE
4A CAL3	XPSD CAL3PSD		CAL3XXX
4B CAL4	XPSD CAL4PSD		CAL4XXX
4C SIGMA9 Parity Error	XPSD PARERRPSD	PARERRPSD*	PARITYER (S9TRAPS)
4D SIGMA9 Instruction	XPSD INSTXPSD	INSTXPSD*	INSTXCPT (S9TRAPS)
INTERRUPTS			
50 Poweron	XPSD POWERON	POWERON both in	BEGINON (PFSR)
51 Poweroff	XPSD POWEROFF	POWEROFF PFSR	BEGINOFF
52 Clock 1 pulse	MTW, 0 0		
53 Clock 2 pulse	MTW, -1 M:RCLOCK2		(SSDAT)
54 Clock 3 pulse	MTW, -1 TINC		(PMDAT)
55 Clock 4 pulse	MTW, 1 J:DGLTAT		(JIT)
56 Memory Parity	XPSD PERPSD	PERPSD	MEMPAR - (TABLES)
57 SIGMA9 Memory Fault	XPSD MEMFTPSD	MEMFTPSD*	MEMFAULT(S9TRAPS)
58 Clock 1 counter zero	XPSD CLK1PSD	CLK1PSD	(Point of interrupt)
59 Clock 2 counter zero	XPSD CLK2PSD	CLK2PSD	(Point of interrupt)
5A Clock 3 counter zero	XPSD CLK3PSD	CLK3PSD	CLOCK1-(CLOCK4)
5B Clock 4 counter zero	XPSD CLK4PSD	CLK4PSD	CLK4 -(SSS)
5C I/O	XPSD IOPSD	IOPSD	IOINT (IOQ)
5D Console interrupt	XPSD OCPSD	OCPSD	OCINT
60 COC input	XPSD COCIN1	COCIN1	COCIP (COC)
61 COC output	XPSD COCOUT1	COCOUT1	COCOP

* PSD Contained in S9DATA Module generated by SYSGEN PASS2

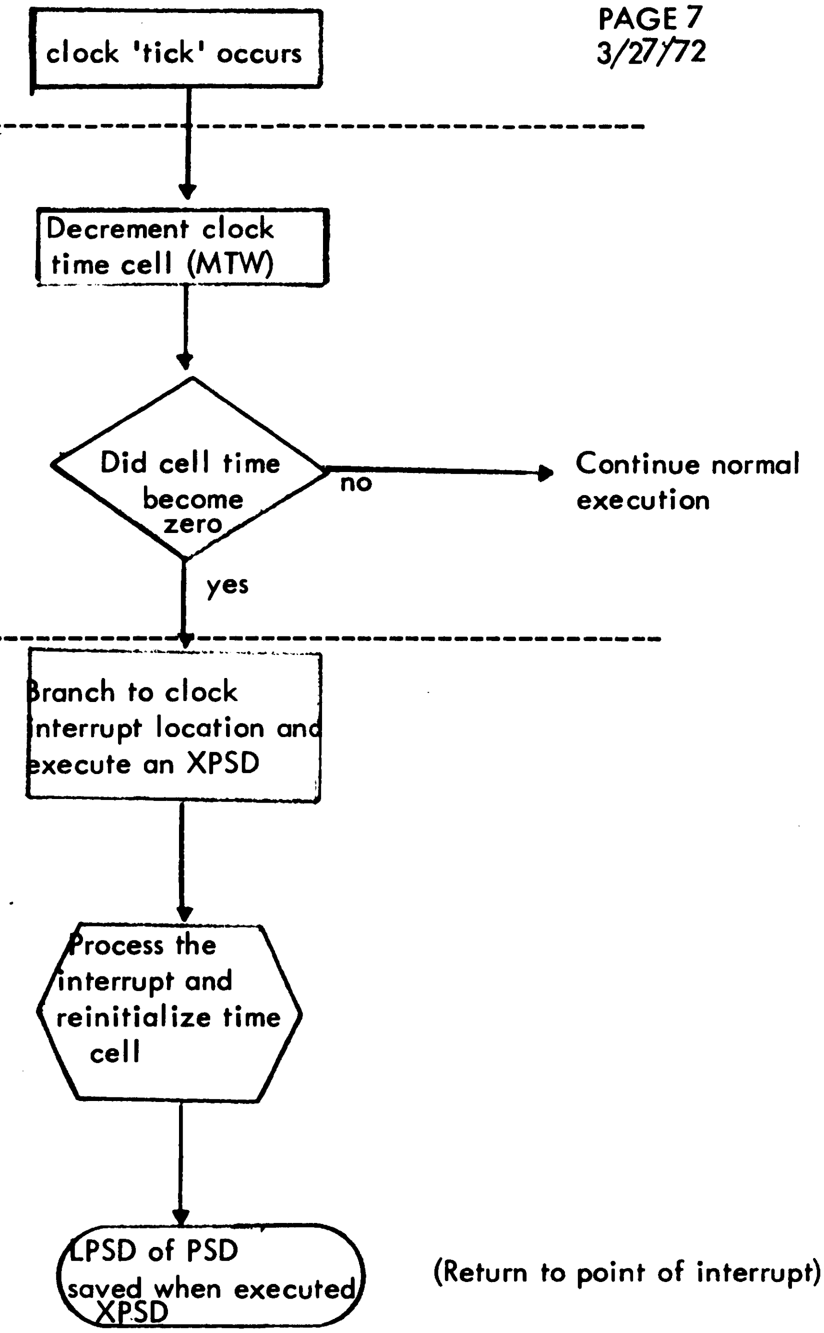
TRAP IN LOCATION X'48'
CAL1: Call a monitor service routine.
Monitor Segment Name



A An external event triggers the execution

B Execute: Modify and test instruction. If the contents of the location modified (time cell) becomes zero, the hardware will 'branch' to the interrupt location connected to the time cell (C)

C



ID

ENTRY

PURPOSE

This module contains the subroutines for entry to and exit from the Monitor when processing CAL's and traps (except hardware error traps). Since it is part of the root of the Monitor, it is always in core.

The subroutines for entry to the Monitor perform the functions of saving the current environment (PSD and registers) and providing basic decoding routines whereby control is transferred to the appropriate Monitor service or fault routines. Later, when a given function has been processed, control is returned to this module which then provides an exit route from the Monitor.

SUBROUTINES

ENTMAP - This procedure is invoked at the beginning of each entry point in ENTRY except for the stack overflow and clock 1 and 2 entries. It saves the trap condition codes, sets the map bit in the current PSD according to the map bit in the PSD at the time of the trap, pushes the 19 word environment and changes the scheduling clock, clock 4, to count in the service time counter, J:OVHTIM.

MAPUNMAP - determines if trap PSD was mapped or unmapped. It sets the current PSD map bit according to the trap PSD and checks if there is room in the stack for a 19 word environment. If not, branch to recovery entry point, RECOVER. Otherwise, push 7 registers and exit.

ENTSUB- pushes the remaining 12 locations of the 19 word environment. It then does a store double of the trap PSD into the first doubleword of the 19 word environment in the stack. Finally, the clock 4 pulse location is modified to tick into the service time counter, J:OVHTIM, in JIT.

EXECUTION TRAP ENTRIES

NOPPGM, FIXOVF, FLTFLT, DECFLT

These entries go through the ENTMAP procedure, load register 3 with a one bit mask according to the type of trap and load register 0 with the physical address of the trap XPSD (e.g., X'40' for NOPPGM). Exit is to 40TRAP, the execution trap processing routine in ALTCP.

STKOVF

The stack overflow entry is basically the same as the above trap entries except that special checks must be performed to determine which stack is involved. If the stack is a user stack, then STKOVF proceeds as above. If the stack is a monitor stack (mapped or unmapped) special action must be taken to prevent the monitor from looping. If the PSD at the time of the trap was master mode/unmapped then it was the monitor tempstack and exit is to RECOVER (software check 1C). If master/mapped the users monitor tempstack is arbitrarily initialized to look empty and exit is to RECOVER where the user will be aborted.

UNIMP

The unimplemented instruction trap entry does an ENTMAP procedure, stores an error code of 5 in the error subcode field ERO in JIT and aborts the user with a code of X'A4' via T:ABORTM in STEP.

CAL2XXX, CAL3XXX, CAL4XXX

The CAL2, CAL3, CAL4 instructions are treated as execution traps. The ENTMAP procedure is executed, an error code of X'B2' is loaded in register 14 and control passes to CALBAD in ALTCP.

CAL TRAP ENTRY

CAL1P

The only legal CAL in UTS is CAL1. This entry point does an ENTMAP procedure and transfers control to CAL1P11 in CALPROC.

MONITOR EXIT FROM CAL PROCESSING

TRAPEXIT

This is the common exit routine for CAL service modules of the monitor. It increments by 1 the instruction address portion of the PSD which was saved in the users monitor tempstack at CAL entry. It then exits to the execution scheduler (SSS) at T:SSEM which schedules the current, or some other, user for execution.

UNUSED CLOCK ENTRIES

CLK1XXX, CLK2XXX

UTS does not make use of clocks 1 and 2. If an installation should have these clocks and if the counter zero interrupts should be armed and triggered, the entry points here will execute an LPSD back to the point of the interrupt.

ID

CALPROC - CAL1 Dispatcher

PURPOSE

The function of CALPROC is to perform the initial decoding of CAL1, 1 and CAL1, 2 (I/O related) CAL's and transfer to the appropriate service module. All other CAL1's are processed by ALTCP (CAL1, 3-9). CALPROC also contains a common exit point for most I/O CAL's, IOSPRTN, which determines if an abnormal or error condition occurred during the CAL. If yes, IOSPRTN stores information in the users registers and modifies the PSD in the users monitor tempstack to enter the user at an error or abnormal address. CAL2, CAL3 and CAL4 are illegal traps in UTS and are handled at entry point CALBAD in ALTCP.

USAGE

B CAL1P11 from CAL1P in ENTRY or CAL11N3 for accelerated CAL1s.

INPUT REGISTERS:

(R0)= address of CAL1 instruction which caused the trap
(R3)= condition codes and floating control after execution of the CAL1 instruction in Byte 3, i.e., the register field of the CAL in bits 24-27.

OUTPUT REGISTERS:

If not CAL1, 1

(R6)= contents of the effective address of the CAL - usually the first word of the FPT
(R7)= address of the second word (word 1) of the FPT
(R8)= Byte 3 of R8 contains byte 0 of the FPT, i.e., FPT code and optional indirect bit.
(R11)= address of common, non-I/O CAL exit, TRAPEXIT in ENTRY.
If CAL1, 1 (I/O CAL's),
(R6)= DCB address specified directly or indirectly in the first word (word) of the FPT
(R7)= address of second word of FPT
(R8)= FPT code (optional indirect bit zeroed)
(R11)= address of common I/O CAL exit, IOSPRTN in CALPROC

INTERACTIONS

T:OV (REMEMBER) The procedure REMEMBER is defined in System UTS (Section UD) used in assembling UTS monitor routines. The procedure consists only of a "BAL, 14 T:REMEMBER", an entry point in the monitor/shared processor overlay associating routine T:OV (Section EC).

RECORD – a diagnostic recording routine. It records information in a wrap-around buffer. What information is recorded is based on a code input in R1 (Section LF).

DATA BASES

C11TV-	CAL1, 1 transfer vector, word table, contains instructions
C11CDS-	CAL1, 1 codes, byte table, contains FPT codes

These two tables are organized in parallel. The instructions in C11TV are either "LI, 15 module address" or "B module address" and serve as a transfer vector for I/O CAL's other than device type. The use of the tables is described below under CHECKCAL under SUBROUTINES.

DEVCDS-	CAL1, 1 device codes, byte table, containd device FPT codes The use of this table is described under CHECKCAL below.
C12TV-	CAL1, 2 transfer vector, word table, contains instructions
C12CDS-	CAL1, 2 code, byte table, contains FPT codes

These two tables are similar to C11TV and C11CDS except that they are for CAL1, 2 traps.

SUBROUTINES

ANLZB- analyzes the instruction in R1 and returns its effective address in R0.

CHECKCAL- The function of this routine is to search the specified byte table for the specified number of entries against the code value in SR1 (R8). If the code is found in the table, the instruction in the same entry of the specified parallel table is executed and return is to the link address (provided the instruction executed is not a branch). If the code is not found and the CAL is not a CAL1, 1, exit is to the illegal trap entry CALBAD in ALTCP. If it is a CAL1, 1, checking continues against the device CAL type FPT codes. If found, a REMEMBER procedure is executed to record the current overlay and control is transferred to the device CAL processing module, IOD. If not found, control is transferred to CALBAD.

INPUT REGISTERS:

(R1) = number of bytes to search
(R2) = address of table of codes to search
(SR1) = code value being searched for
(D1) = address of transfer vector table and also link register

Exits: There are four ways CHECKCAL can be exited.

1) Executing a branch instruction in a transfer vector table.
2) Executing a "LI, 15 module address" instruction in the transfer vector and exiting to the link address.
3) Unconditional branch to C11TV if a device FPT code is found and,
4) Unconditional branch to CALBAD if the FPT code is not in the table.

CVREG - This routine performs the conversion of R0 mentioned under GTWD.

GTWD - The purpose of this routine is to load R1 with the contents of the address pointed to by R0. If (R0) is a register (0<(R0)<15), the location in the users monitor tempstack that contains the contents of the register is loaded into R0 by subroutine CVREG.

ISEXU - This routine checks if the contents of R1 is an EXU instruction. If yes, it exits to the link address; if no, it exits to link address plus one.

DESCRIPTION

At entry the total system CAL count (C:CAL) and the total CAL count for the current user (J:CALCNT) are incremented. Preliminary decoding of the CAL is performed leaving the R-field of the CAL, the first word of the FPT, address of FPT plus one, the FPT code and the non-I/O exit address (TRAPEXIT) in registers. The CAL is recorded in the diagnostic wrap around buffer via RECORD. A switch is then executed on the R-field of the CAL. If it is not a CAL, 1 or CAL1, 2 control goes to ALTCP for dispatching. If it is a CAL1, 2 the code is checked and control is transferred to the appropriate service module.

If it is a CAL1, 1 the FPT code byte is checked for the indirect bit. If set, the DCB address is fetched indirectly through the first word of the FPT. The DCB address is checked for validity by comparing the specified DCB address against the chained DCB table which starts at ADCBTBL in JIT. If the specified DCB address is not found, the user is aborted with a code of X'AF'. If the DCB is M:UC, only read, write, and device operations are allowed. If another operation is specified, no error is returned, but the request is ignored. Next the specified FPT code is checked against the table of legal CAL1, 1 FPT codes by the routine CHECKCAL. Before entering CHECKCAL, R15 is loaded with the entry address of IOD (device CAL processor) and R11 is loaded with the common I/O CAL exit address IOSPRTN. Immediately following the BAL to CHECKCAL is a call on T:REMEMBER in T:OV which remembers the current overlay and exit address (R11) in the overlay tempstack and which causes the current overlay to be reassociated upon exit from processing the incoming CAL. Following the BAL to T:REMEMBER is a "B *R15". The effect of this sequence is to cause a "REMEMBER" for those FPT codes that have a "LI, R15 module entry" in the parallel transfer vector table. Those which have a "B module entry" go directly to the routine from CHECKCAL.

The common I/O exit point, IOSPRTN, checks (R8 $\neq$ 0) if return is to be made to the users error or abnormal address. If not, exit to TRAPEXIT in ENTRY which causes control eventually to return to the user at CAL plus one. If control is to go to the user's error/abnormal entry, check if run status abort bits are set (J:RNST). If yes, exit to TRAPEXIT (SSS will catch the abort bits on the way out of the CAL at T:SSEM). If no, set up the users registers 8 (address of CAL plus one) and 10 (error code and DCB address) and modify PSD in the users monitor tempstack to point to the error or abnormal address specified in the DCB or in the FPT. Exit is then to TRAPEXIT1 in ENTRY which stores the PSD back into the stack and exits to T:SSEM which leads ultimately back to the user.

ID

ALTCP - Alternate CAL processor and trap handler

PURPOSE

To dispatch CAL1, 3 -CAL1, 9 requests to the appropriate service module. It also processes traps 40-46, illegal CAL traps 49-4B, and undefined CAL1 traps.

OVERVIEW

This module performs two logically distinct functions. One is alternate (to CAL1, 1 and CAL1, 2) CAL processing (entry CALCK); the other is trap handling (entries 40 TRAP, BADCAL and CALBAD). There will be two instances below of each section devoted, respectively, to alternate CAL and trap processing.

USAGE

Alternate CAL processor CALCK:

B	CALCK from CALPROC
(R3)=	R-field of the CAL1 (e.g., if CAL1, 8 then (R3)=8)
(R6)=	Contents of 1st word of the FPT pointed to by the CAL
(R7)=	Address of FPT+1
(R8)=	Byte 0 of the 1st word of the FPT, i.e., the FPT code and optional indirect list
(R11)=	address of common CAL exit point, TRAPEXIT, in ENTRY

Trap processor 40TRAP:

B	40TRAP from ENTRY	
(R0)=	address of trap location (X'40'-X'46)	
(R2)=	condition codes and floating controls after the trap in Byte 0	
(R3)=	a 1 bit mask corresponding to the type of trap (trap location)	
	X'80' - illegal CAL (X'49'-X'4B', CAL2-CAL4) or undefined CAL (invalid R-field or FPT code on a CAL1)	
	X'20'	Non-allowed operation trap (X'40')
	8	Stack limit trap (X'42')
	4	Floating point fault trap (X'44')
	2	Decimal fault trap (X'45')
	1	Fixed point arithmetic fault trap (X'43')
(R4)=	address in users or monitor TSTACK which (when doubleword accessed) points to the trap PSD, i.e. the address can be odd or even.	

Undefined and illegal trap entry BADCAL and CALBAD:

B	BADCAL	This entry is for undefined CAL's and simply loads R14 with a monitor error code X'AE' and falls into CALBAD (from CALPROC and ALTCP)
B	CALBAD	From ENTRY
(R14)=	monitor error code X'AE' - undefined CAL1 X'B2' - illegal CAL2-CAL4	

INTERACTIONS

Alternate CAL processor:

CVREG	A subroutine in CALPROC is used to compute the true memory address of the register address in R0, i.e., the address in TSTACK of the users register specified in R0.
RECOVER	The system recovery routine (Section LD)

Trap Processor:

RECOVER	The system recovery routine (Section LD)
T:REG	"Report event and give up control" entry in the execution scheduler (SSS, Section EA). Used here to re-associate a debugger when a user traps who has a core library and debugger associated.
T:PAC	Memory management set processor access routine (in MM, Section GA). Used here to set the access register to allow the debugger (DELTA) to store into its context page.
T:ABORTM	"Monitor is aborting the user" entry point in the job step control routine STEP, Section EB.
T:SSEM	"mapped exit from monitor to user" entry point in the execution scheduler (SSS).
T:UTSXTS	A subroutine in the execution scheduler SSS which moves a 20 or 21 word environment from the users monitor temp stack, TSTACK, to the user temp stack in the users TCB.

DESCRIPTION

Alternate CAL processor

The FPT code byte in byte 3of R8 is checked to see if the indirect bit is set. If it is, R6 is loaded indirectly through word 0 of the FPT via CVREG. The routine then switches on R3, which contains the R field of the CAL, to individual decoding subroutines for each of the defined CALs, CAL1, 3-4-6-8-9.

If the R field is greater than 9 or equal to 0, 5 or 7, exit is to the undefined CAL entry, BADCAL, in ALTCP (described below). If the value is 1 or 2, then either CALPROC is unable to detect CAL1, 1 or CAL1, 2 because it is clobbered or control has transferred to CALCK (ALTCP) from an unexpected source. In either case recovery is called for (software check code X'7C'*).

CAL1,9

The effective address of CAL1, 9s determines which service module has been requested. The defined values are 1-6. Since the preliminary CAL decoding in CALPROC has computed the address of FPT plus one (in R7) as if it were a register, R7 actually points to a location in TSTACK which corresponds to "effective address of CAL plus one". The actual code is recomputed from that value. A switch on the effective address is then executed which leads to BADCAL for undefined codes.

CAL1,3

This routine processes debug CAL's. The FPT code is verified and loaded into R0 via CHECKCAL. The shared monitor overlay, DEBUGSEG is invoked via the procedure OVERLAY which BAL's to the module T:OV (section EC). All overlays which have more than 1 entry assume that R0 contains an index into a transfer vector of entry points. Thus the FPT code is a transfer index.

CAL1,4

This code performs a simple validity check on the FPT code and branches to the appropriate entry point in UCAL (UTS specific CAL processor).

CAL1,6

First a validity check is performed on the FPT code. Then the users privilege level in his JIT (JB:PRIV) is checked. If it is X'A0' or above, access to the service module requested is allowed. If less than X'A0' and the TEL or CCI in control flag (TIC) in the users flag (UH:FLG) is set, access is allowed. If TIC is not set, return is to the user at CAL plus one with CC1 set to indicate the error.

CAL1,8

This routine performs a validity check on the FPT code via CHECKCAL and exits to the appropriate service module.

* At the time of publication this "screech" code had never been seen.

DESCRIPTION

Trap Processor

The trap handler has two entry points: 40TRAP for traps to locations X'40' - X'45' and BADCAL for undefined and illegal CAL traps. When a trap occurs, the action taken depends on the following five decision points:

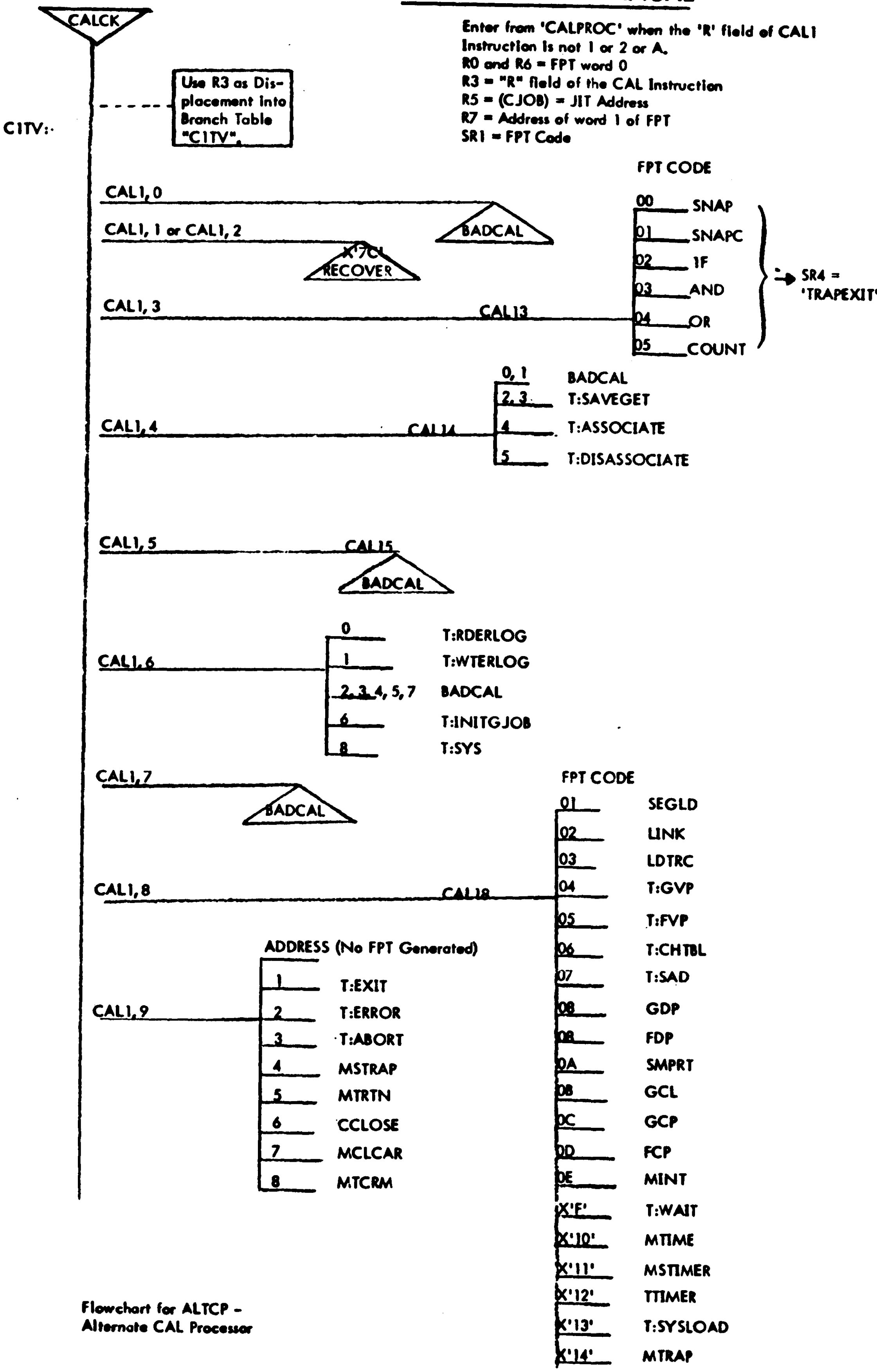
1. Trap occurred in master mode
2. Trap occurred in TEL or CCI
3. Trap occurred in DELTA
4. DELTA is associated with trapping user
5. User has trap control of the particular trap

Correspondingly the following action is taken:

1. If a trap occurs in master mode (in the monitor) or unmapped, system recovery is invoked (section LD).
2. If TEL or CCI traps, recovery is invoked.
3. If the trap occurred while DELTA was in control, further special checks are performed to determine if DELTA was attempting to modify the user's pure procedure. If it was, the trap handler executes the store for DELTA, sets the "pure procedure swap" bit in the users flags (UH:FLG) and exits via T:SSEM to the trapped instruction plus one. The "pure procedure swap" bit is set to insure that the now modified procedure portion of the users program is swapped out the next time he is selected for outswap.
4. If DELTA is associated with a user who traps, control will be ultimately transferred to DELTA's trap entry. A special check must be made first to determine if the user was running with a shared core library. The reason for this is that core libraries and DELTA both reside in the reserved special shared processor area of virtual memory. Only one can be in the user's map at any given time. If a core library was associated the processor use count (PB:UC) is decremented for the library and incremented for DELTA. The "ready to run" flag in UH:FLG is reset for this user to force a swap after the associate processor event is reported via T:REG. The effect of this is to get DELTA into the user's map. If DELTA is in core an I/O-less swap results. An associate processor event (E:AP) is then reported via T:REG (SSS). When SSS returns to ALTCP the access protection registers are set up for DELTA via T:PAC in MM. DELTA's stack and trap entry addresses are loaded in R1 and R2 and control falls in to common code for giving trap control to the user (TRAP40).

5. If none of the above conditions hold, the trap control flags (J:USENT) in the user's JIT are checked to see if the user has requested control of the current trap.

If no, the user is aborted via T:ABORTM in STEP with an error code of X'A4' and a subcode which uniquely identifies the trap (table B-5, UTS Reference Manual). If yes, a check is made to see if the user program was loaded with a TCB (J:TCB ≠ 0). If not, the contents of user's R0 are taken as a TCB address. R2 is loaded with the user's trap entry point from J:USENT and control goes to the stack transfer code at TRAP40. At TRAP40 the user's environment is transferred from the mapped monitor stack, TSTACK, to the user's TCB pointed to by R1 via T:UTSXTS in SSS. If the user's TCB can't be used because the stack pointer doubleword or the stack are not in a data page or the stack is not big enough, the user is aborted via T:ABORTM with an error code of X'A3'. After a successful transfer the trap location (X'40' - X'46') is stored in the last word of the user's TCB stack. The stack transfer left the trap environment in TSTACK. The condition codes immediately after the trap XPSD and the user's trap entry address are stored in the PSD in the trap environment. The address of the TCB is stored in the R0 register in TSTACK and the address of the trap environment in the TCB stack is stored in the R1 register in TSTACK. Exit to the user level trap control routine is via T:SSEM in SSS which will ultimately pull the modified trap environment from TSTACK.



Flowchart for ALTCP - Alternate CAL Processor

UTS TECHNICAL MANUAL

ID

TABLES, S9TRAPS - Error trap handlers

PURPOSE

The purpose of the hardware error trap handlers in TABLES and S9TRAPS is to process the following traps and interrupts on Sigma 7 and Sigma 9 computers.

Watchdog Timer Runout Trap, X'46'
Sigma 7 Memory Parity Interrupt, X'56'
Sigma 9 Memory Fault Interrupt, X'57'
Instruction Exception Trap, X'4D'
Parity Error Trap, X'4C'

USAGE

The trap handlers are entered as the result of a trap or interrupt to one of the designated memory locations. An XPSD in that location transfers control to a unique entry point for each handler. The handlers are entered in Master, unmapped mode with interrupts inhibited, and use register block zero.

The service routines for these traps and interrupt perform error correction and recovery based on the condition of the operating system, the user environment and the type of error. Whenever possible, the service routines attempt to localize a problem to a particular user and avoid entering the system RECOVERY program. The general steps taken to service a hardware error include correcting the error, if possible, logging the error in the system error log and choosing an appropriate return. The chocie of return is based on the conditions at the time of the trap or interrupt and the type of error. The possible choices, beginning with the most favorable, are:

1. Return to the point of the trap or interrupt and attempt to re-execute that instruction or continue with the next instruction in sequence.
2. Abort the user's current job step.
3. Abort the user's job. The current terminal user will be logged off.
4. Call the system RECOVERY routines.

INTERACTION

The following monitor subroutines contained in other modules are used.

MSROCTY
T:ABORTM
T:DELUS
RECOVER

DATA BASES

JB:CMAP	is a byte table in JIT which contains the physical page number corresponding to each virtual page.
J:JAC	is a two bit table contained in JIT which contains the access protection codes for each virtual page.
HIGH	is the page number of the last page of physical memory.
DCTSIZ	is the number of entries in each of the monitor DCT tables.
DCT1	is a table of unit numbers of devices attached to the system.
DCT5	is a table containing a set of flags for each device.
S:CUN	is the system identification number of the current user.
UB:JIT	is a table containing the physical address of each users JIT.
JBUP	is a word in JIT containing the beginning user page.

DESCRIPTION

Introduction

An XPSD instruction in the appropriate interrupt or trap location addresses a PSD pair in TABLES which contains, as the new PSD, the address of the proper handler routine. The XPSD instructions are coded with a register field value of X'A', which causes subjective addressing to be used and the register pointer control to be loaded from the new PSD. Each of the routines described in this section is entered in the Master mode, unmapped and with register pointer control equal zero.

On Sigma 9 computers the Parity Error, Instruction Exception and Watchdog Timer Runout traps all set the Processor Detected Fault flag. The routines to handle these traps call a common trap entry subroutine, RESETPDF, to accomplish the following:

1. Save all registers in TSTACK
2. Save the trapped PSD in registers 12 and 13.
3. Reset the instruction address of the PSD, used for entry to the trap routine, to an alternate trap handler.
4. Reset the PDF flag so that a subsequent PDF error cannot occur and cause a CPU "hang-up".

The alternate trap handler recognizes when a subsequent trap has occurred at an expected place, (i.e., at an LMS instruction), and allows the program to continue. This method permits other types of traps that set PDF to occur and to be processed correctly without interfering with the current trap. For example, a Watchdog Timer Runout that occurs while processing a Data Bus Check will not interfere and proper recovery will be made from each trap.

SIGMA9 PARITY ERROR TRAP

The Parity Error Trap routine used on Sigma 9 computers is composed of three sections to process the three types of Sigma 9 parity errors. The main entry point, PARITYER, in the S9TRAPS module, receives control from the XPSD instruction in location X'4C' and immediately calls the trap entry subroutine S9RSTPSD. Following this call a branch is made to one of the three sections described below, based on the value of the Trap Condition Code (TCC).

To process Data Bus Check errors the program forms an Error Log entry containing the trapped instruction and the Real Effective Address computed for that instruction. To compute the Real Effective Address the program restores the registers to their values at the time of the trap and executes an Analyze instruction addressing the trapped instruction. If the trapped program was in Mapped mode an LPSD is first executed to enter the Mapped Mode and an LRA instruction is used to compute the Real Effective address obtained by the Analyze instruction. After the Error Log entry is formed a call is made to ERRLOG to record it in the Error Log buffer.

To complete the processing of the Data Bus Check error the routine performs an exit sequence which is also used when exiting from the Map Check error section of the program. This code resets the instruction address of the PSD, (PARERPSD) used for entry to the trap routine, by replacing the alternate trap routine address with the address of PARITYER. Following this is a call to RAFTST, which tests the Register Altered bit in the trapped PSD. If the Register Altered bit was not set, RAFTST returns and the registers are restored and control returned to the trapped instruction. If the Register Altered bit is set, and if the trapped program was in Master Mode, the program branches directly to RECOVER with an error code of X'23'. Otherwise, if the trapped program was in Slave mode, the users job step is aborted and the message;

job-id PARITY ERROR - STEP ABORTED

is typed on the operator's console. The RAFTST subroutine is also used by the Map Register Check and Watchdog Timer Runout trap routines.

If the Parity Error was a Map Register Check error the program branches to MAPERR. This section performs a search for Memory Map errors by exeucting an LRA instruction addressing each of the 256 possible virtual pages. An entry is made in the Error Log for each error found during the search and a correction attempted. However, if the faulty Map Register corresponds to the virtual page number of the users JIT page a correction cannot be done and the user is logged off the system. The message;

job-id PARITY ERROR - USER LOGGED OFF

is typed on the operator's console.

If the faulty Map Register does not correspond to the virtual page number of the users JIT the program reloads the Memory Map Registers and Access Protection Codes for the faulty page. To do this, the bad page number is used to compute the address of the word in the users JB:CMAP and J:JAC tables which contains the physical page address and access control codes to be restored. This address is then used in the appropriate MMC instructions to reload the Memory Map and Access Protection Codes. The Memory Map Register is tested again following the correction and, if the correction was successful the search is continued. If the Map Register error was not corrected the program branches to RECOVER with a code of X'23'.

After all the Map Registers have been tested the program performs the exit sequence described above for the Data Bus Check errors. That is, the PSD is reset, the Register Altered bit is tested and the appropriate return is taken.

If no errors are found during the search a null Error Log entry is made and the program returns using the exit sequence described.

When the Parity Error Trap was caused by a Memory Parity error the program branches to S9MEMERR in the TABLES module. This section performs a search of all memory locations using an indexed Load Word instruction. When an error condition is detected a Parity Error Trap occurs and a correction subroutine is called. This subroutine uses the LMS instruction to load the three memory status registers for the bank in which the error occurred and then clears them. The contents of the memory status registers are placed in an Error Log entry exactly as they are obtained from the memory and the entry is added to the Error Log by a call to ERRLOG. The memory examination is continued until all memory banks have been tested and an entry made in the Error Log for each bank in which an error is found. In addition to the memory status, the first Error Log entry also includes a table of device addresses indicating what devices were busy when the Parity Error occurred. Up to six busy devices can be noted in the table. If the Memory Parity error was caused by a Loop check or Overtemperature condition the program branches to RECOVER with an error code of X'27'. Otherwise, the word which had the bad parity is stored back into memory in an attempt to correct the bad parity. The bad word is loaded once again to test it. If the bad parity was not corrected, the alternate trap routine will be called by a second Parity Error trap and will branch to RECOVER with an error code of X'23'.

It the parity error is corrected a call is made to SETMXERR. This routine searches the current users JB:CMAP table for a physical page number which corresponds to the physical page containing the faulty location. If the search is successful a virtual page number is obtained and compared with the Beginning Users Page (BUP) number. If the search is unsuccessful the physical page does not belong to the current user. For each case an error code is set and compared with a maximum error code. If the new code is larger it is stored as the maximum error code. The possible error code values and their meanings are:

0	The error was in the users pages.
1	The error was in the monitor context pages.
2	The error was in the monitor or was not owned by the current user.
>2	The error was reported during a memory access by an IOP.

Following the call to SETMXERR the program returns to the memory test loop. After all memory locations have been tested the Memory Fault Interrupt is cleared and the PSD used for entering the trap routine is reset with the address of PARITYER. The final step in processing the Memory Parity error is to analyze the maximum error code saved during the bank testing. If the code is greater than two, indicating that the error was detected by the memory during an IOP access, the registers are restored and control is returned to the user. If the error code is equal to two then the faulty location was found in the monitor and the program branches to RECOVER with an error code of X'28'. If the maximum error code is one the user is logged off or, if the code is zero, the users job step is aborted.

In the case where no memory errors can be found an Error Log entry with status words of zero is logged and the exit procedure described above is performed with the maximum error code set to zero.

SIGMA9 MEMORY FAULT INTERRUPT

The Simga 9 Memory Fault Interrupt (MFI) is triggered when a fault is detected by the memory as the result of an IOP or CPU access. If the memory access was by a CPU, and the fault is not a Loop Check or Overtemperature error, a Parity Error Trap is also triggered. In this case the Parity Error trap inhibits the MFI until the PDF flag is reset. Upon entering the Memory Fault Interrupt service routine a check is made of the instruction address of the interrupt and if the interrupt occurred immediately following the LPSD used to reset the PDF flag the registers saved in the temp stack are removed and control returned to the Parity Error Trap program. If the Memory Fault Interrupt did not occur after the LPSD instruction it was caused by a Loop Check or Overtemperature error, (which do not generate Pairty Error Traps) or by a memory access by an IOP. If the interrupt was caused by a Loop Check or Overtemperature error the program will branch to RECOVER with an error code of X'27' after logging the error. If the interrupt resulted from an IOP access to memory the interrupt service routine initializes the Parity Error Trap PSD to the alternate trap program, sets the maximum error code to a value greater than two and loads the interrupt PSD into registers 12 and 13. The program then branches to the Memory Parity Error section of the Parity Error Trap program.

SIGMA7 MEMORY PARITY INTERRUPT

When a Memory Parity Error Interrupt occurs on a Sigma 7 computer an entry is made to the interrupt service routine at MEMPAR. The registers are saved and the interrupt PSD is loaded into registers 12, 13. The Memory Parity Interrupt is cleared, armed and enabled and the Memory Fault Indicators are read with a RD instruction.

The address of an alternate interrupt service routine is stored into the interrupt service PSD and portions of an Error Log entry are initialized. The program then branches to S9MEMERR to perform the memory search described above. Upon completion of this loop the Error Log entry is recorded by a call to ERRLOG. The Error Log entry is set with the device addresses of any devices that were busy at the time of the interrupt. The interrupt service PSD is then reset with the address of MEMPAR and the program branches to the exit portion of the Sigma 9 Memory Parity Error Trap routine to determine the kind of exit based on where the bad memory location occurred.

If a parity error occurs while exeucting the memory test loop a Memory Parity Interrupt is triggered and the alternate interrupt service routine is called. This alternate routine stores up to two bad memory location addresses in the error log entry. It also calls SETMXERR for each bad location, no matter how many. After reseting the Memory Fault indicators and clearing the interrupt level the alternate interrupt service routine returns to the point of the interrupt in the test loop.

WATCHDOG TIMER RUN OUT TRAP

The Watchdog Timer Runout Trap routine is designed to operate on either Sigma 9 or Sigma 7 computers. The program is entered at WDOGPGM and calls the trap entry subroutine RESET PDF. Following this call is a test to determine if the trapped program was in the mapped mode. If so, the registers saved while executing in the unmapped mode are retrieved, and an LPSD is executed to enter the mapped mode to facilitate analysis of the trapped instruction.

The Real Page Address of the trap is determined from information in JB:CMAP and it and the TCC are stored in the first word of an Error Log entry. The instruction that caused the trap is obtained and stored in the Error Log entry. The ERRLOG program is then called to log the trap information.

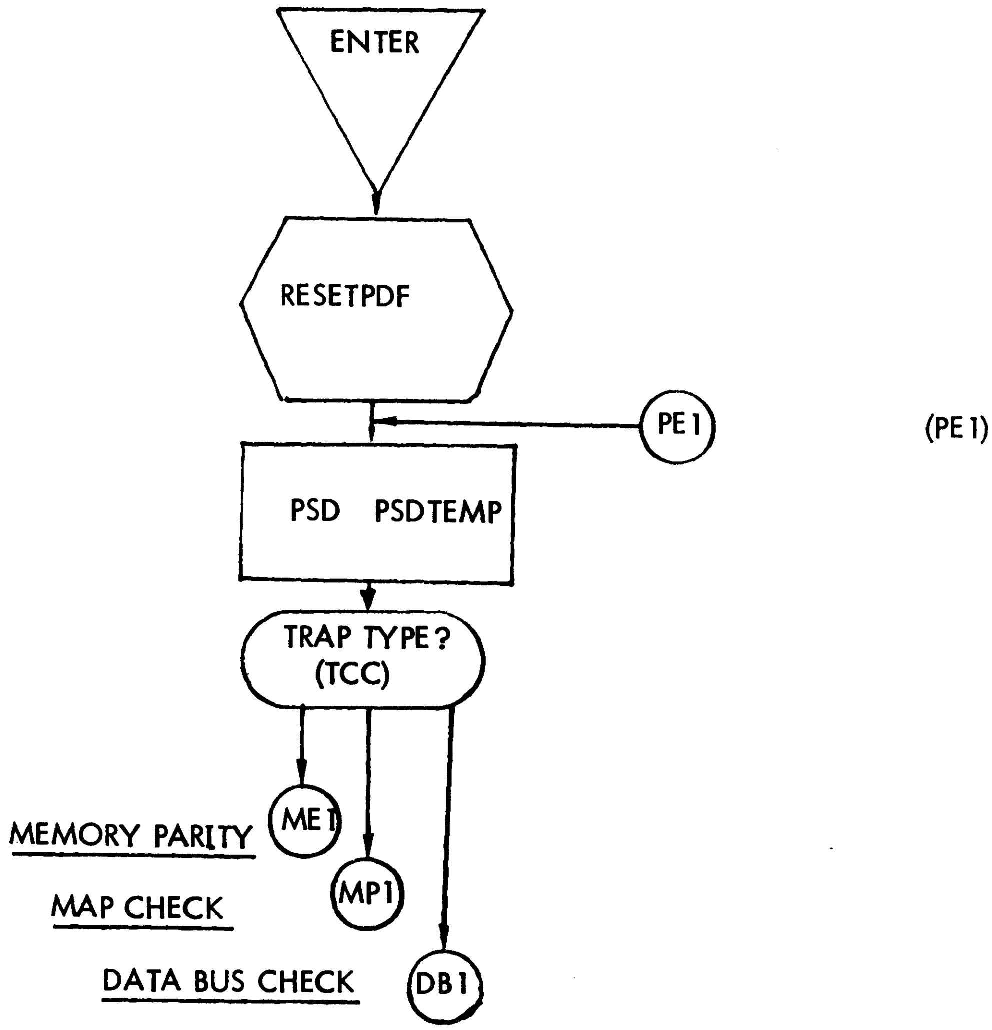
If the trapped instruction was indirectly addressed its effective address is obtained and bits 17 - 19 of this address are saved for analysis later in case the instruction was a Read Direct or Write Direct. If the computer is a Sigma 9 and the time out was in phase one, in which case the instruction completed correctly, the program returns to the user. If the time out was not during phase one the Register Altered bit is tested by a call to RAFTST. If the Register Altered bit is set the user's job step is aborted or, if in Master Mode, a RECOVER exit is taken with an error code of X'1C'. If the Register Altered bit is not set tests are made to determine if the trapped instruction was a Read Direct or Write Direct. If it was, the mode code (bits 17 - 19 saved above) is tested. If the mode code is 0 or 1 the instruction is retried, but if it is 2 or greater the instruction is skipped by incrementing the trapped PSD. Control is then returned to the user.

If the computer is a Sigma 7 the phase tests are not performed but instead the trapped instruction is tested to determine if it is a PSM or PLM. If it is, the program branches to RECOVER with a code of X'1C'. If the instruction is not a PSM or PLM the tests for Read Direct and Write Direct are performed as described above.

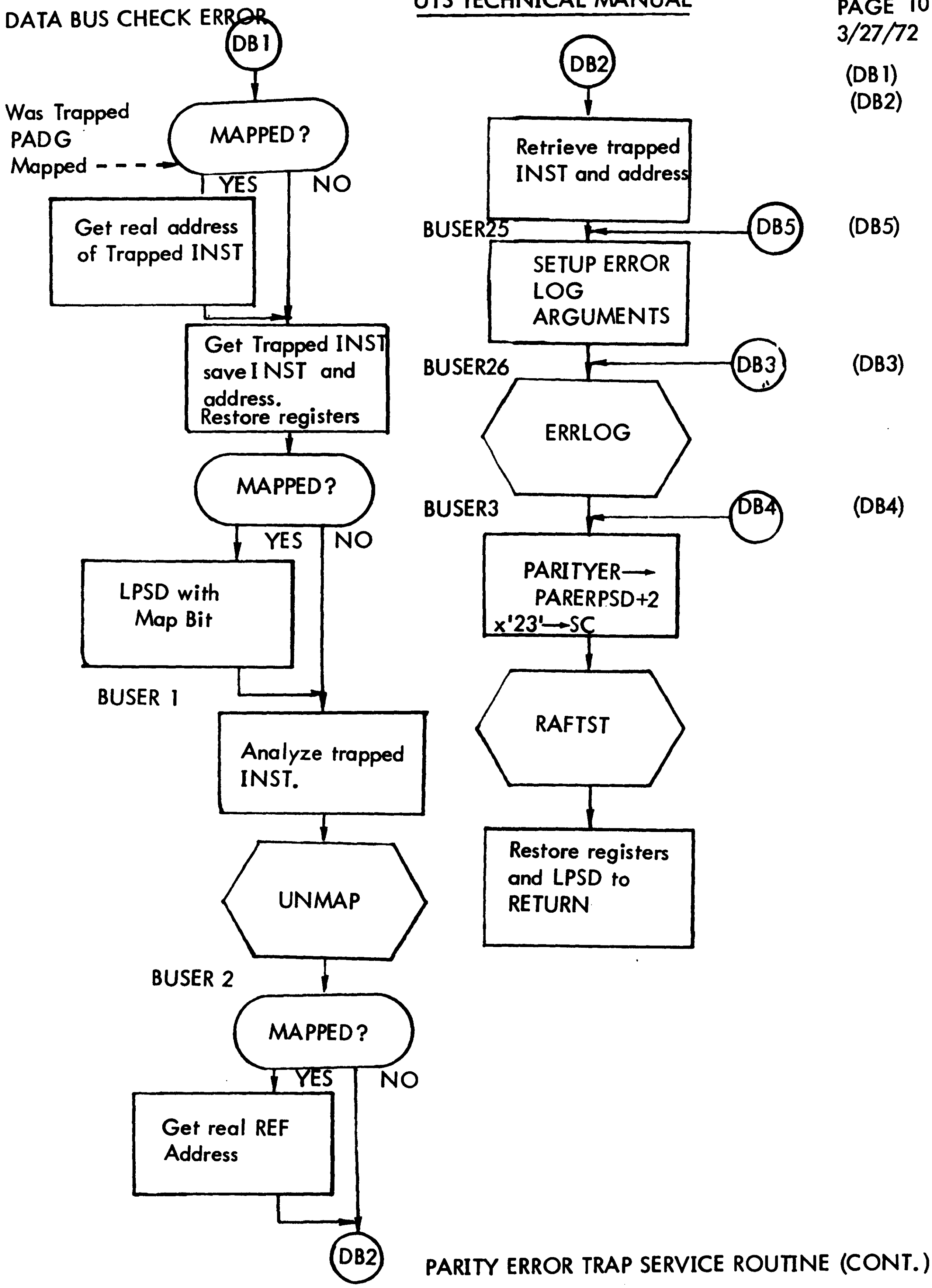
SIGMA9 INSTRUCTION EXCEPTION TRAP

The Instruction Exception Trap is processed by the section of code beginning at INSTXCPT. The usual call to RESETPDF is made to initialize and reset the PDF. If the trap occurred as the result of an invalid register designation and the program was in Slave Mode the trapped PSD is incremented by one to skip the Load instruction, and control is returned to the user. If the program was in Master mode or the reason for the trap was not invalid register designation an Error Log entry is made followed by a branch to RECOVER with an error code of X'24'.

PARITYER



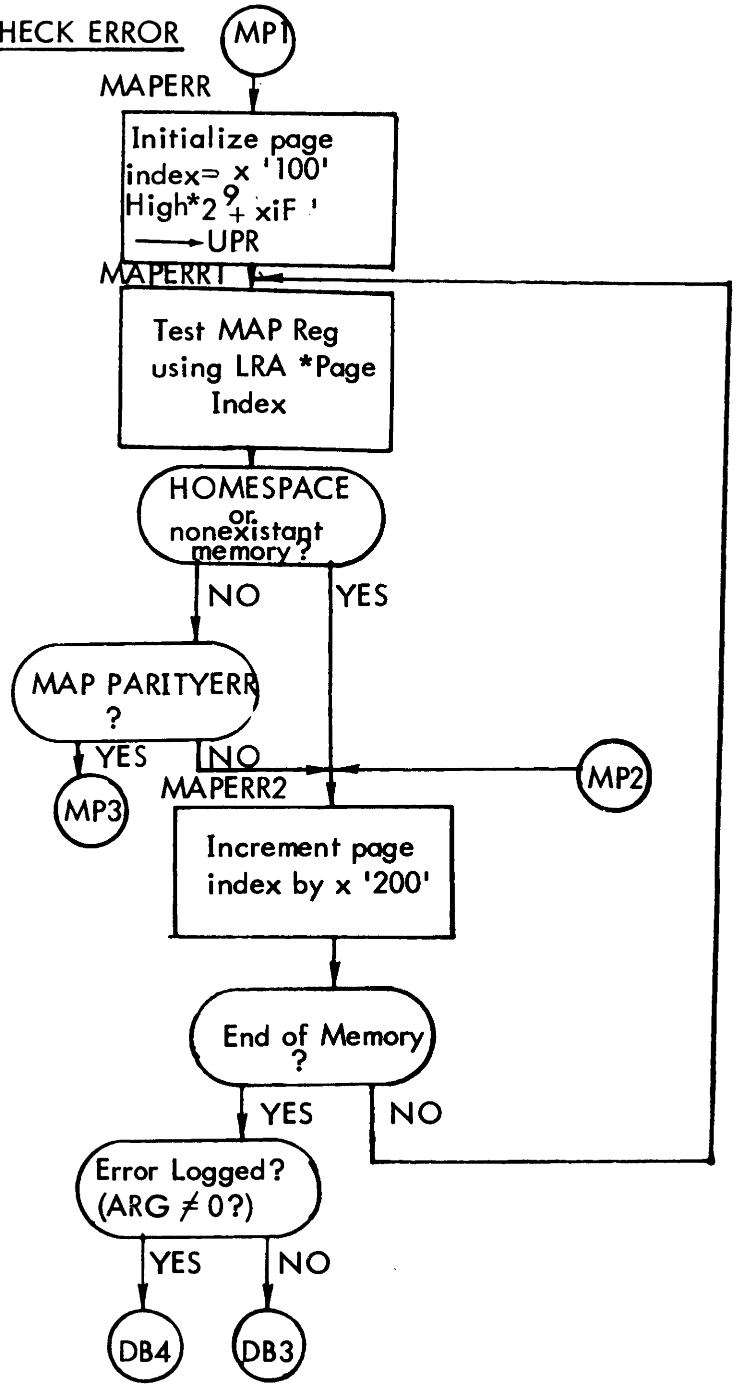
PARITY ERROR TRAP SERVICE ROUTINE



PARITY ERROR TRAP SERVICE ROUTINE (CONT.)

MAP CHECK ERROR

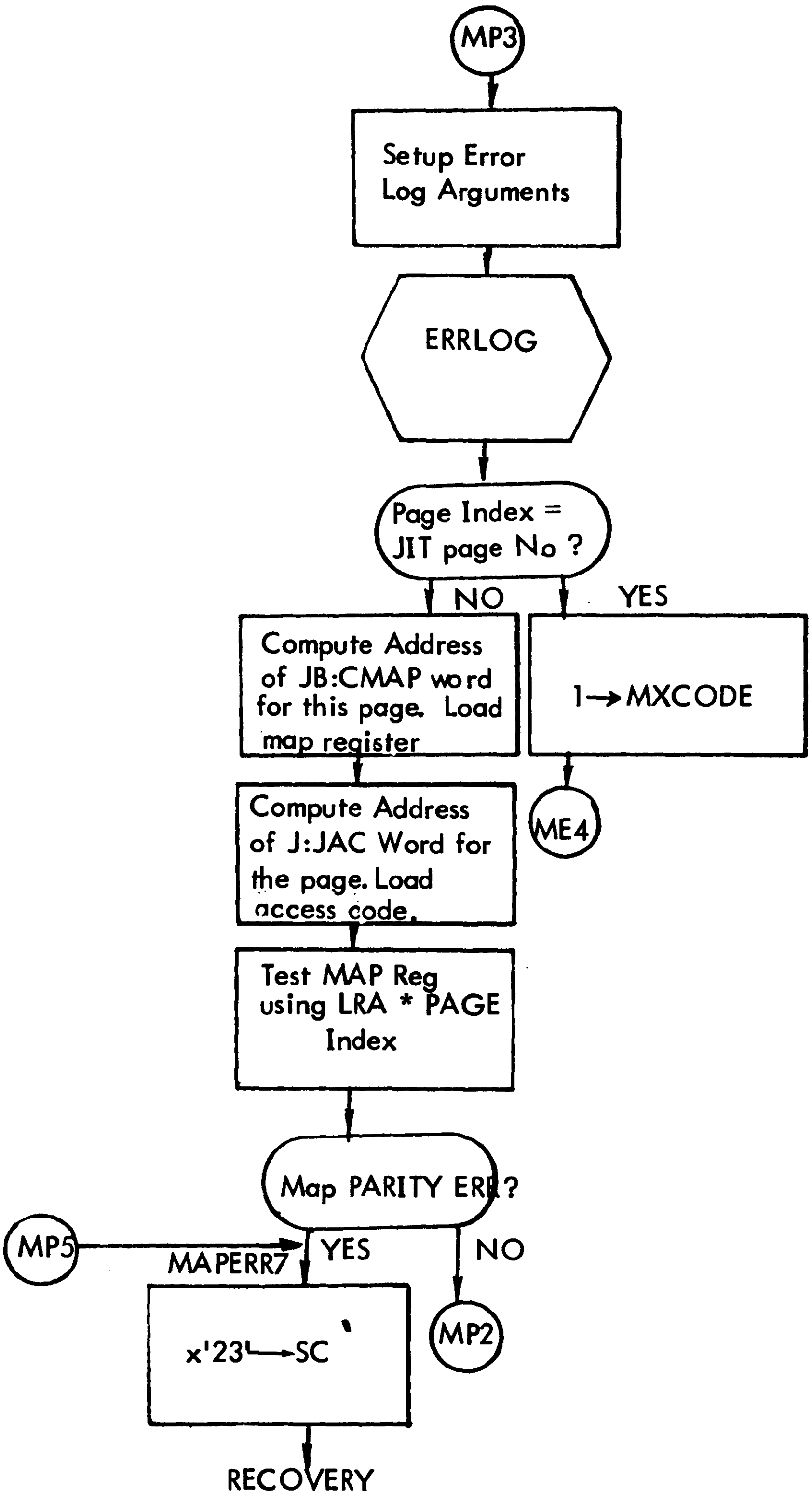
(MP1)



(MP2)

PARITY ERROR TRAP SERVICE ROUTINE (CONT.)

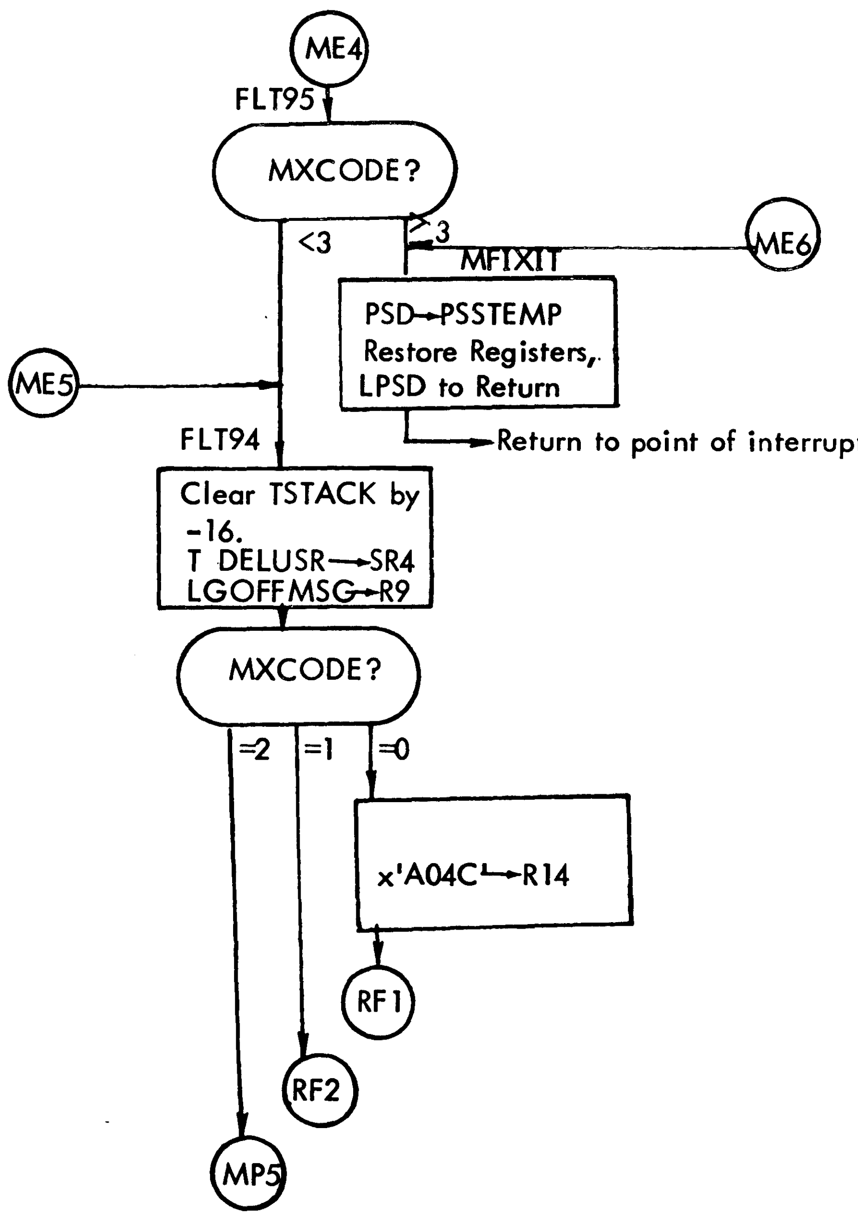
(MP3)



(MP5)

PARITY ERROR TRAP SERVICE ROUTINE (CONT.)

UTS TECHNICAL MANUAL

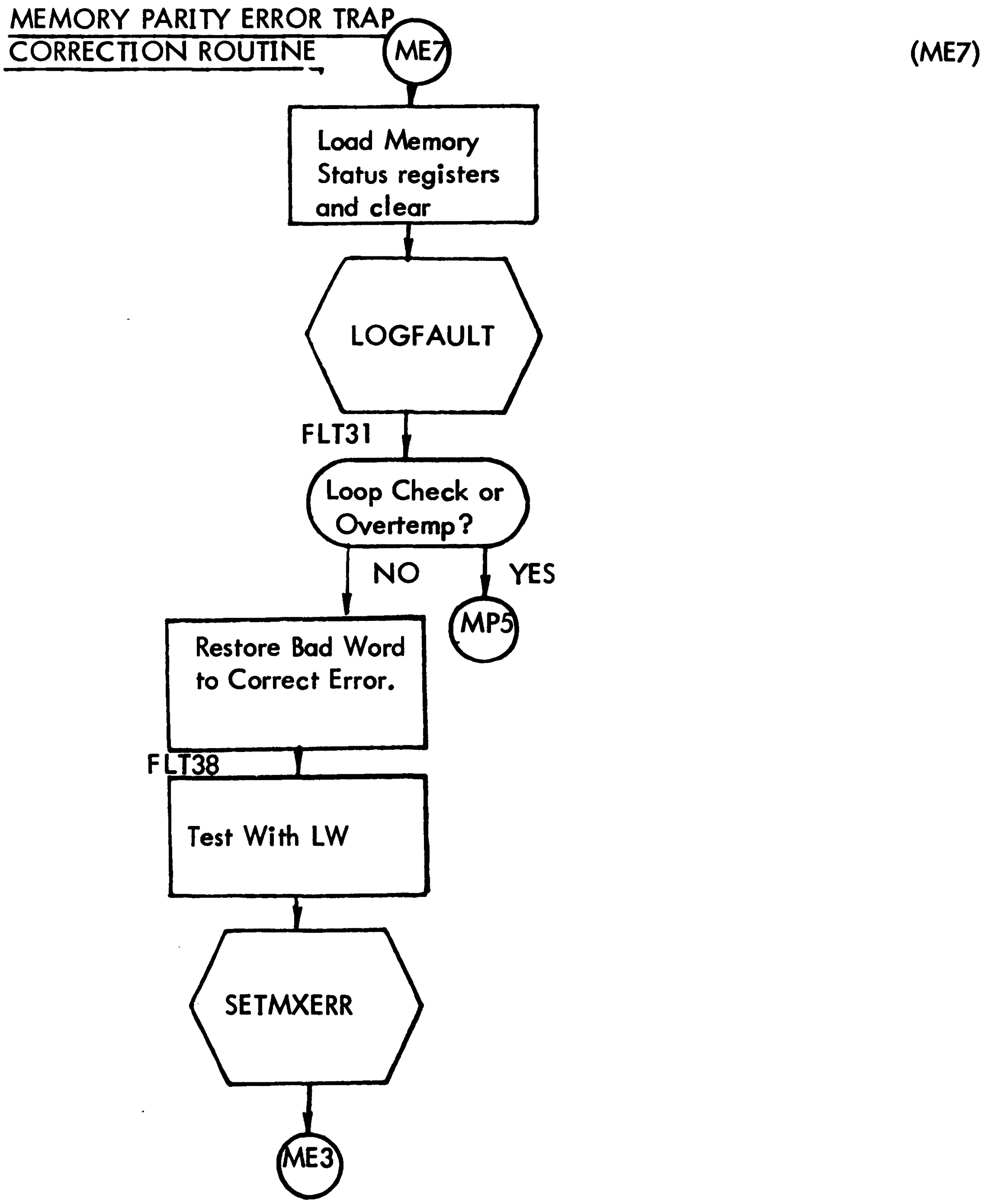


(ME4)

(ME6)

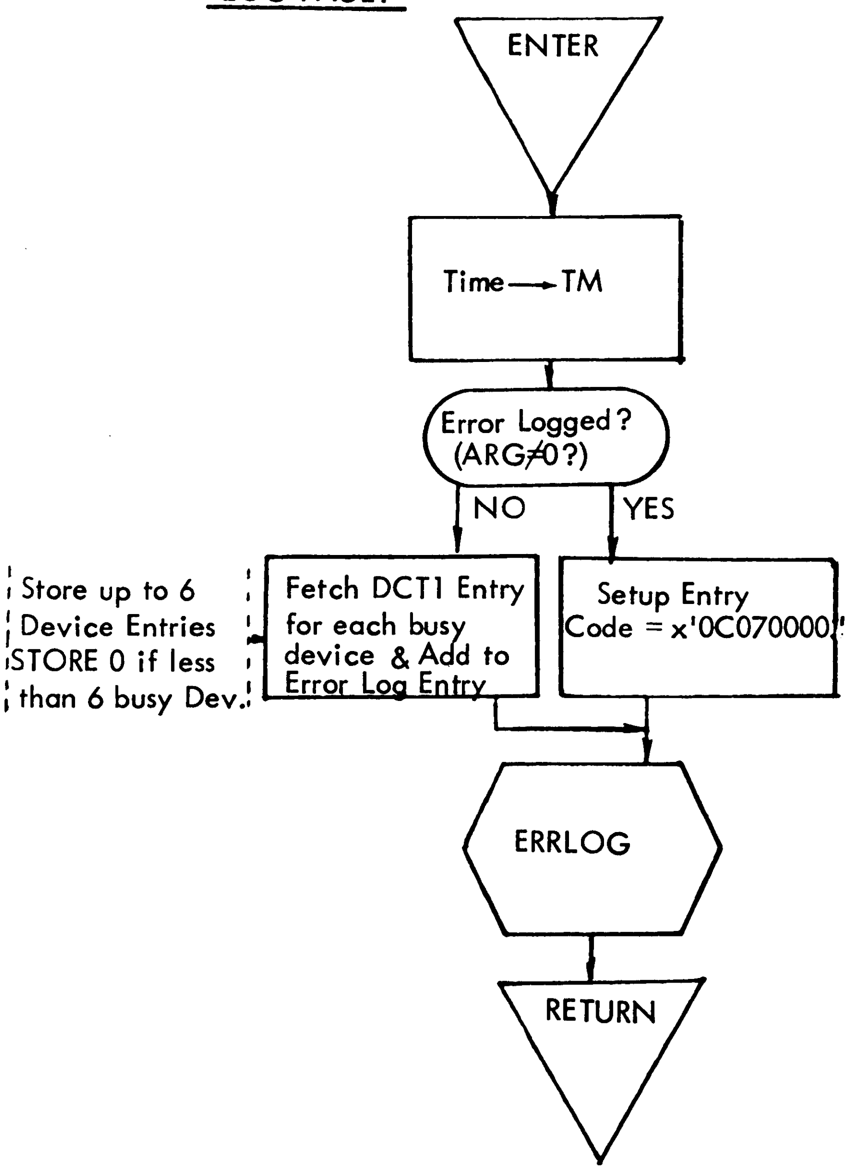
(ME5)

PARITY ERROR TRAP SERVICE ROUTINE (CONT.)



PARITY ERROR TRAP SERVICE ROUTINE (CONT.)

LOG FAULT

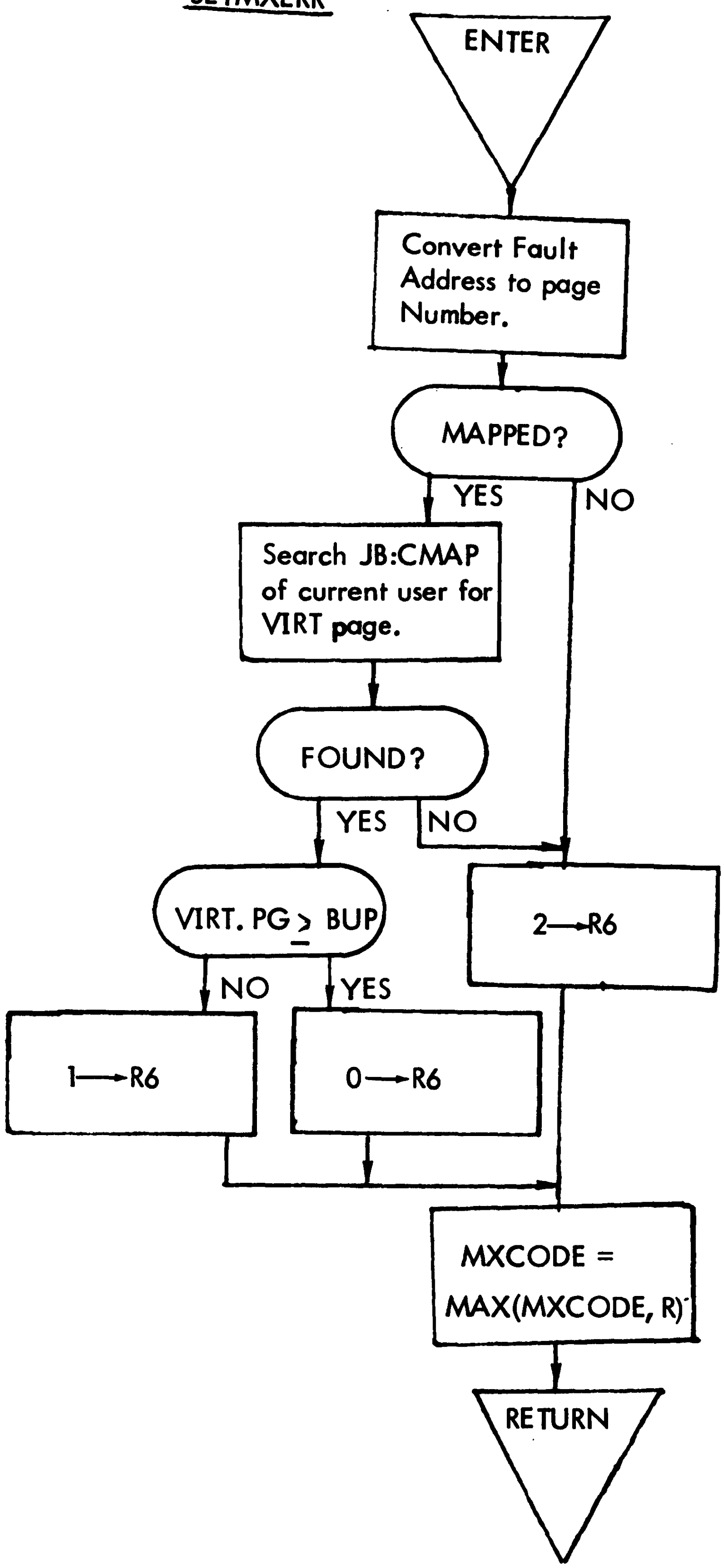


ENTER WITH:

x'070A0000' in R10
Trapped PSD in 12, 13
status in 14, 15, 0

PARITY ERROR LOGGING SUBROUTINE

SETMXERR



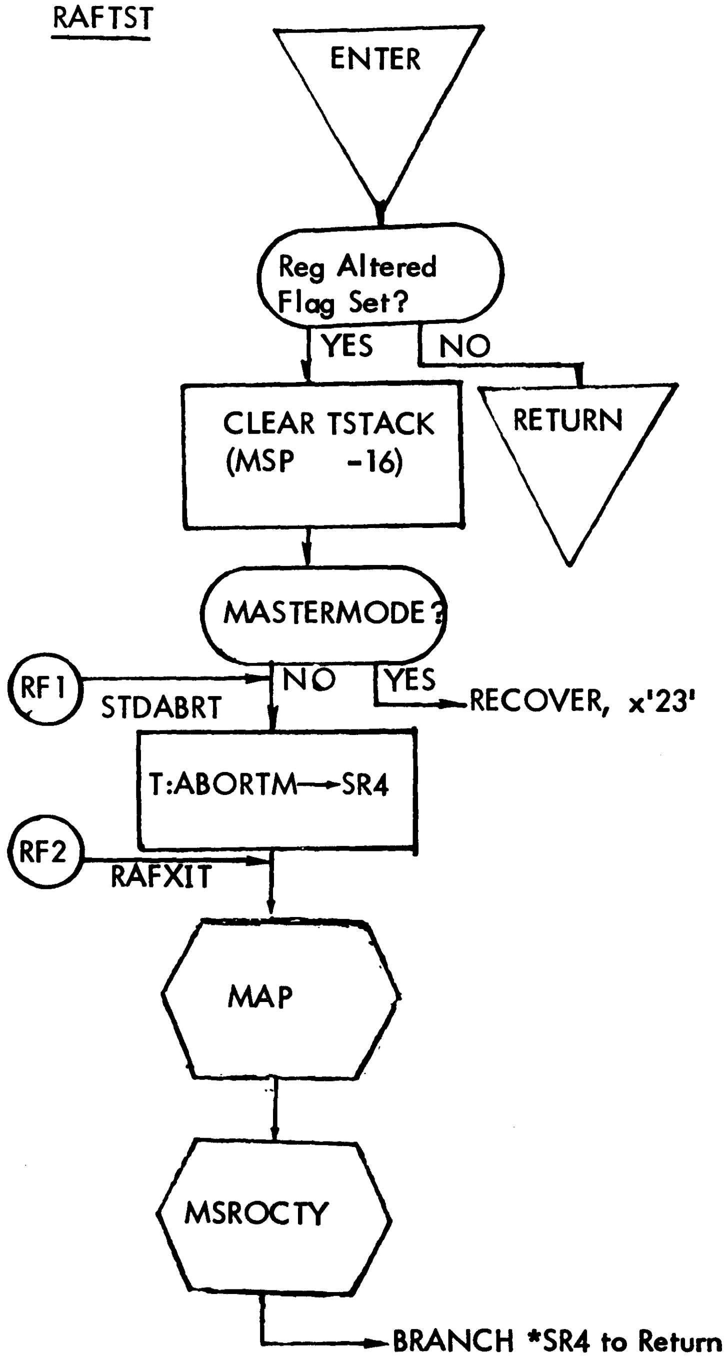
ENTER WITH:

Address of Faulty LOC in R5.
Trapped PSD in 12, 13

SET MAXIMUM ERROR LEVEL SUBROUTINE

RAFTST

ENTER WITH:
Trapped PSD in 12, 13
Message Address In 9
Step Abort Code in 14

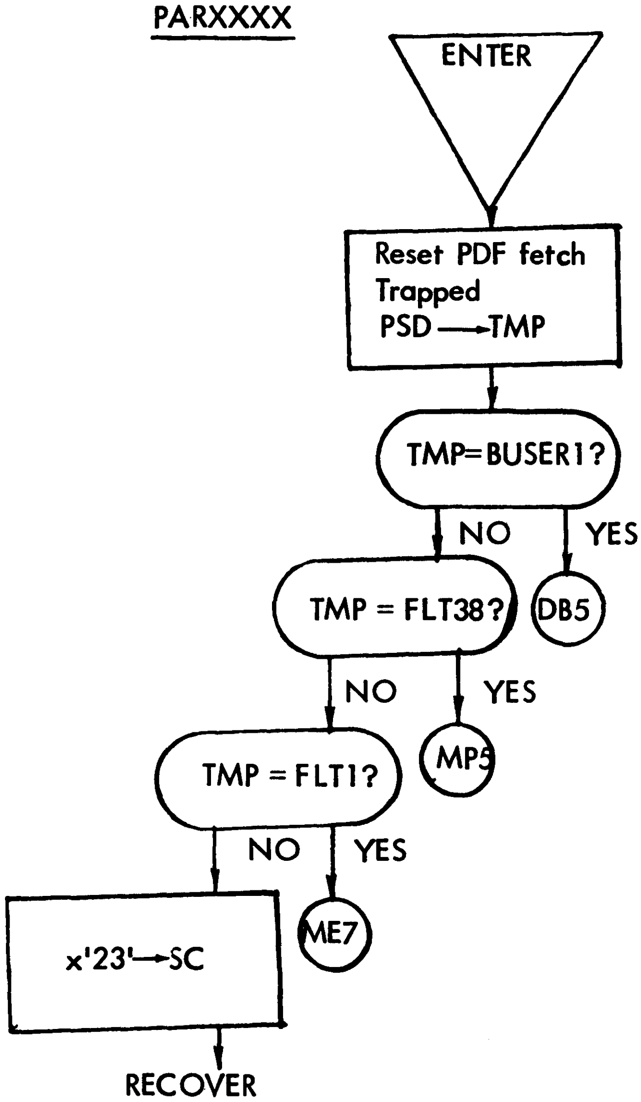


(RF1)

(RF2)

REGISTER ALTERED FLAG TEST SUBROUTINE

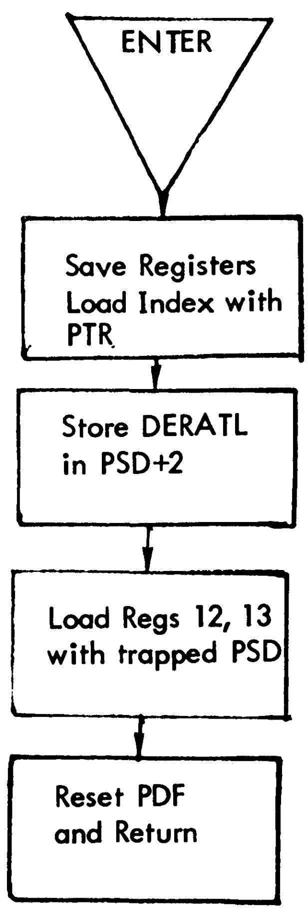
PARXXXX



PDF DOUBLE TRAP ROUTINE

UTS TECHNICAL MANUAL

RESET PDF



ENTER USING

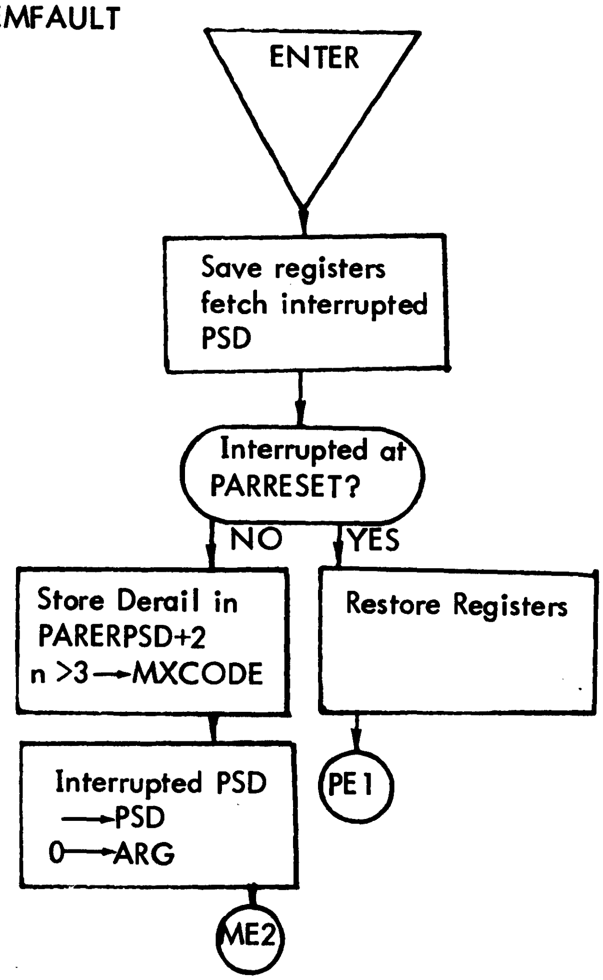
XPSD, 0 RSETPSD
DATA PTR

WHERE:

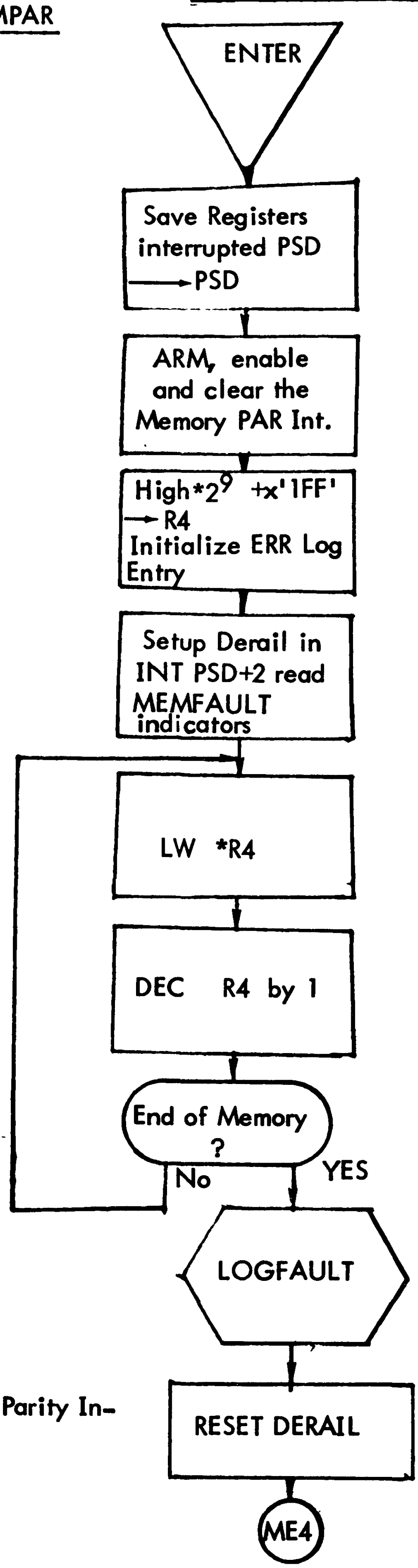
PTR DATA DFERR
DATA nnn
PSD SERVICE PROG

UTS TECHNICAL MANUAL

MEMFAULT

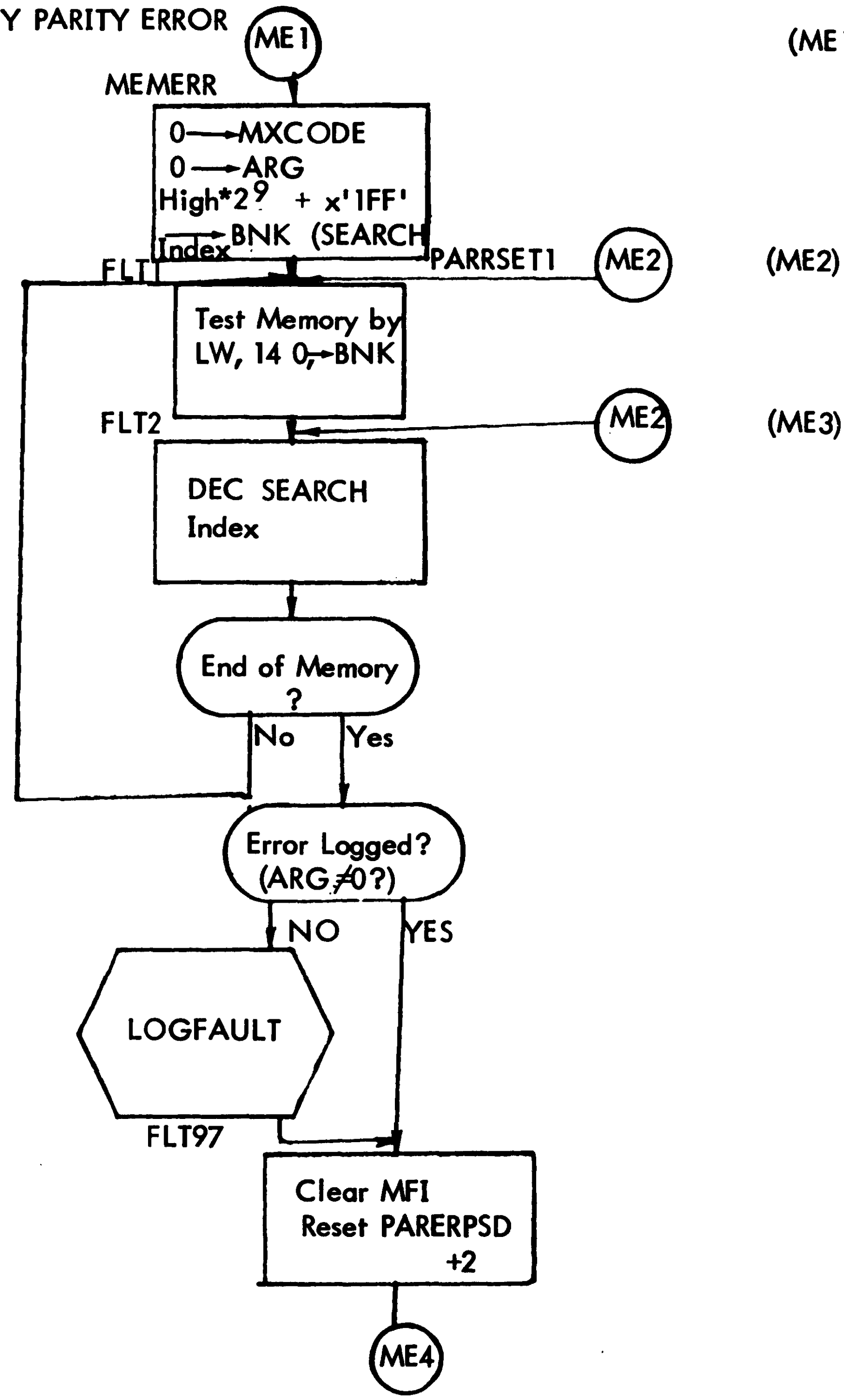


MEMPAR

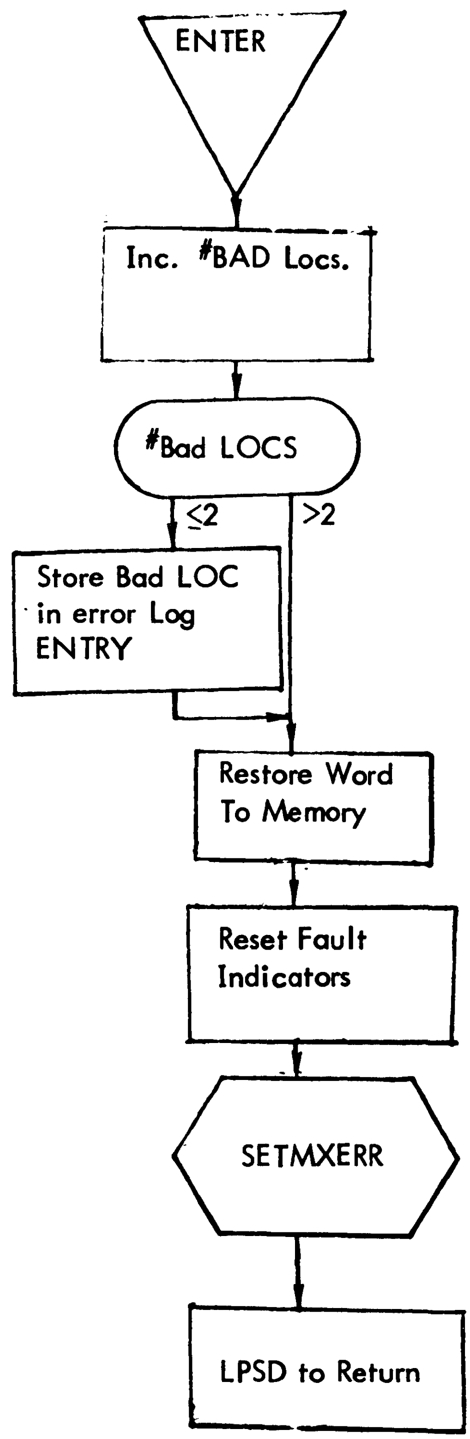


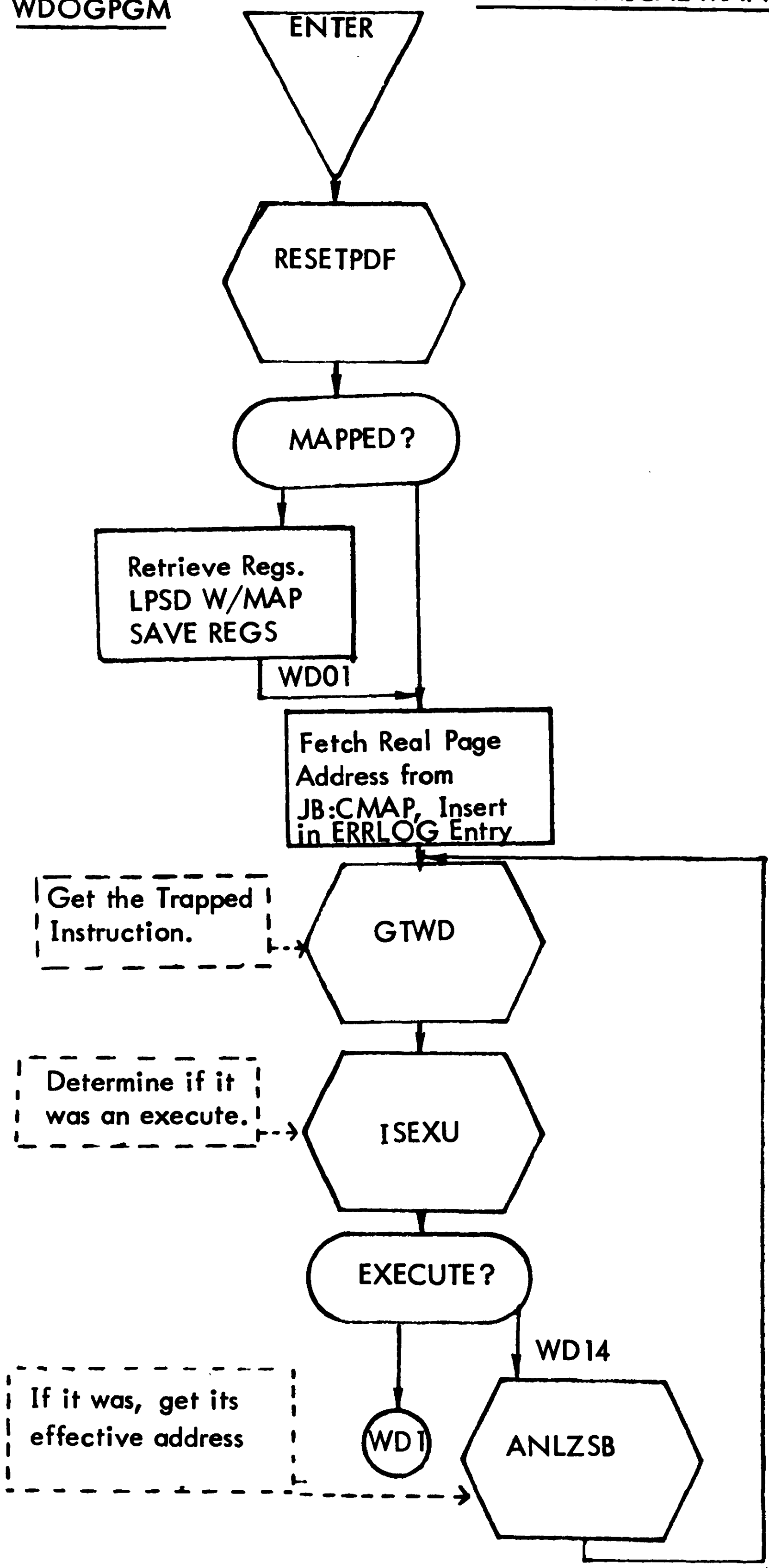
SIGMA7 Memory Parity Interrupt Service Routine

MEMORY PARITY ERROR



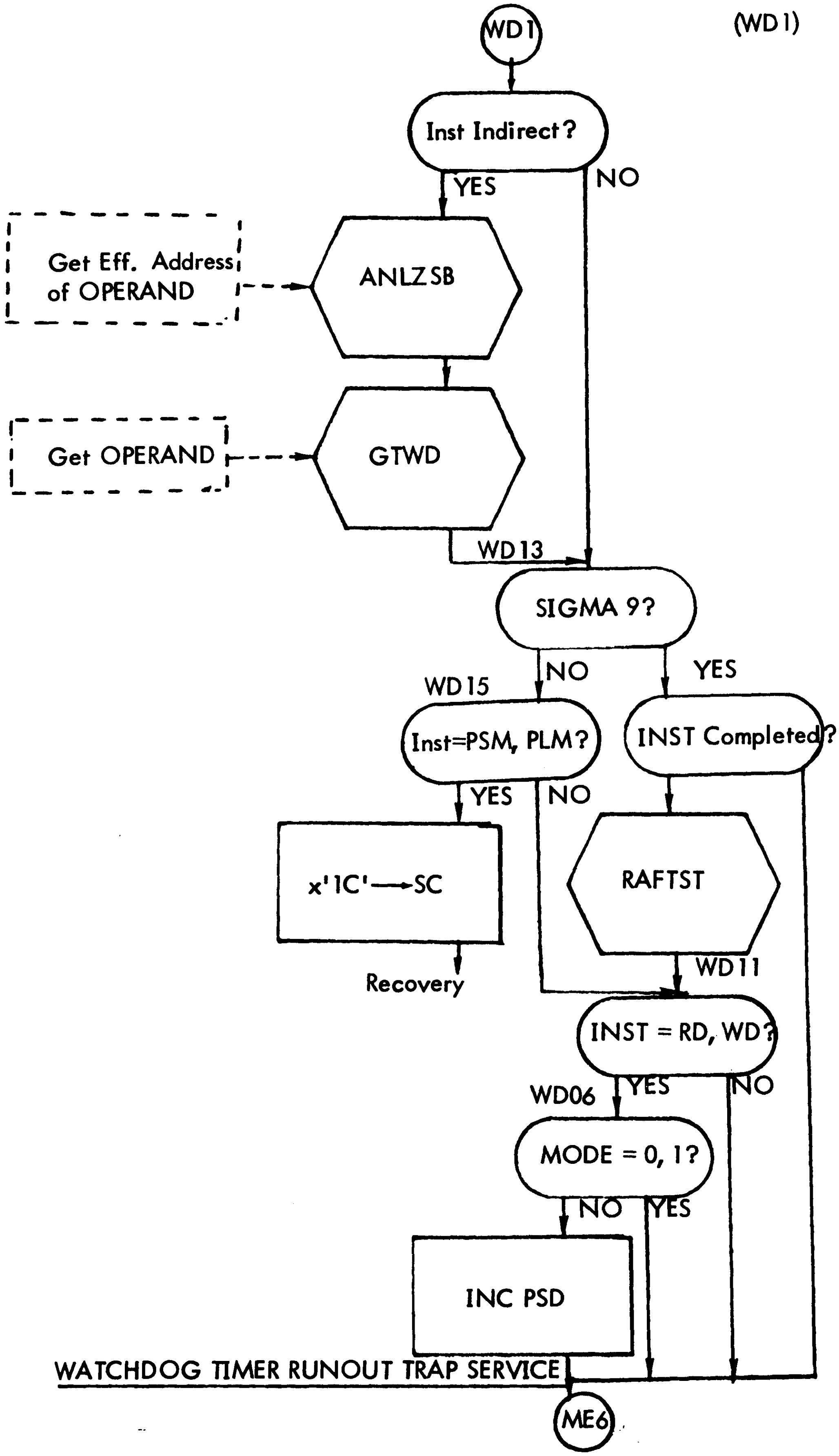
MPR50


ENTER
Inc. #BAD Locs.
#Bad LOCS
≤2
>2
Store Bad LOC in error Log ENTRY
Restore Word To Memory
Reset Fault Indicators
SETMXERR
LPSD to Return

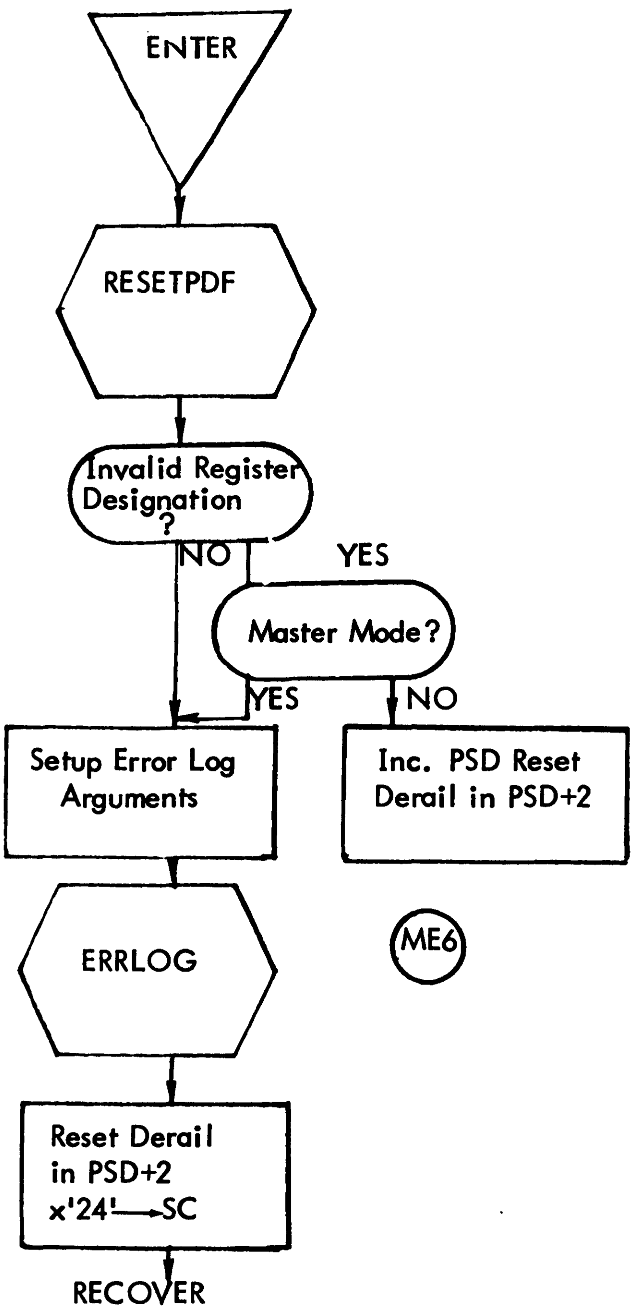


WATCHDOG TIMER RUNOUT TRAP SERVICE ROUTINE

(WD1)


WD1
Inst Indirect?
YES
NO
Get Eff. Address of OPERAND
ANLZSB
Get OPERAND
GTWD
WD13
SIGMA 9?
NO
YES
WD15
Inst=PSM, PLM?
INST Completed?
YES
NO
x'1C'→SC
RAFTST
Recovery
WD11
INST = RD, WD?
WD06
YES
NO
MODE = 0, 1?
NO
YES
INC PSD
WATCHDOG TIMER RUNOUT TRAP SERVICE
ME6

INSTXCPT



INSTRUCTION EXCEPTION TRAP SERVICE

ID

Device I/O

FUNCTIONAL OVERVIEW

The BPM Basic Input/Output System provides a simple interface between all parts of the operating system and the external peripheral devices. It stacks or "queues" the requests for service rather than waiting for each operation to complete before returning to the caller. When a request is completed the caller is notified via certain parameters in the DCB. Or the caller may specify the address of a subroutine to be executed at this time (called the "end-action" routine). It is capable of receiving requests for input at any time or from any place in the system and dispatching them in a manner which is virtually independent of other operations concurrently being executed by the system. Error recovery procedures are invoked when necessary and do not require any additional specifications from the caller.

Requests are normally serviced in the order in which they are received. In a real-time system, requests are serviced by task priority. Precautions are taken to prevent any major service to lower priority requests when a higher priority task is active.

Communication with peripherals is designed to afford the most complete recovery possible from errors and device malfunctions. Operator intervention is enlisted only after all other alternatives have been exhausted.

No restrictions are placed on buffer size or location. Facilities are included for gather-write/scatter-read operations (data chaining), and provision is made to allow construction of IOP command lists outside of the Basic I/O.

The inherent differences between peripheral devices is accounted for by the insertion of device-oriented code (handler) for each type of device in the system. A well-defined handler interface allows addition of new handlers with a minimum of difficulty. Also, a number of subroutines are available which perform common handler functions.

OPERATIONAL OVERVIEW

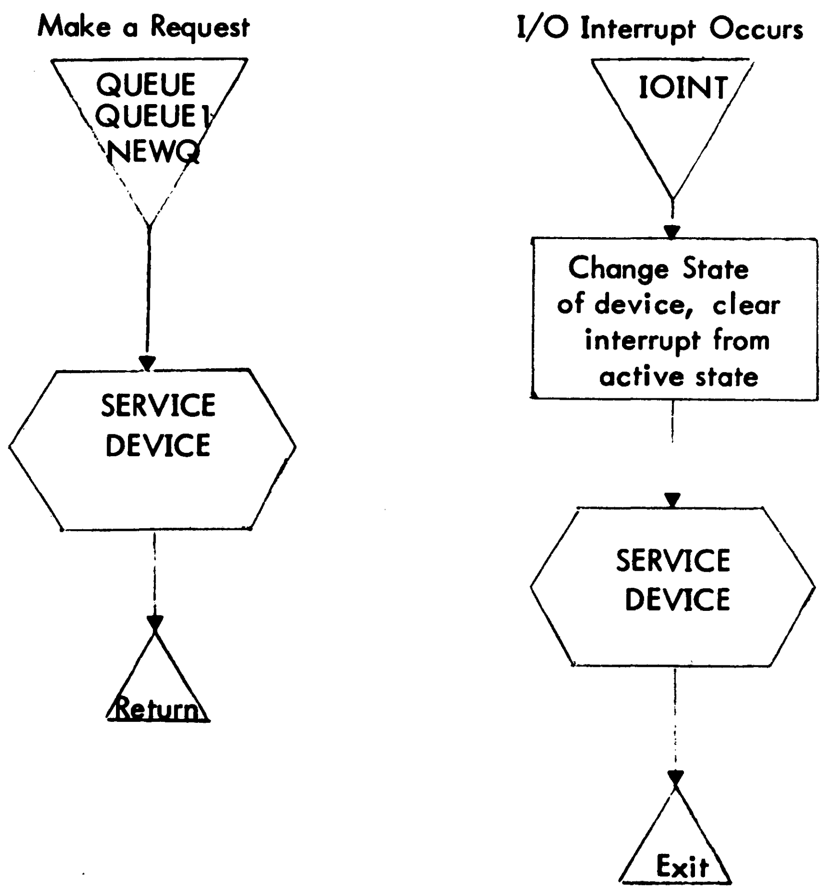
There are two major parts involved in the processing of an I/O request: start (done by STARTIO) and cleanup (done by CLEANUP). The start consists of building the IOP command list and executing the SIO instruction, while the cleanup consists of testing for errors and notifying the caller of the completion. For a given request, the time at which a start or cleanup is done is determined by the I/O scheduler (called Service Device or SERDEV).

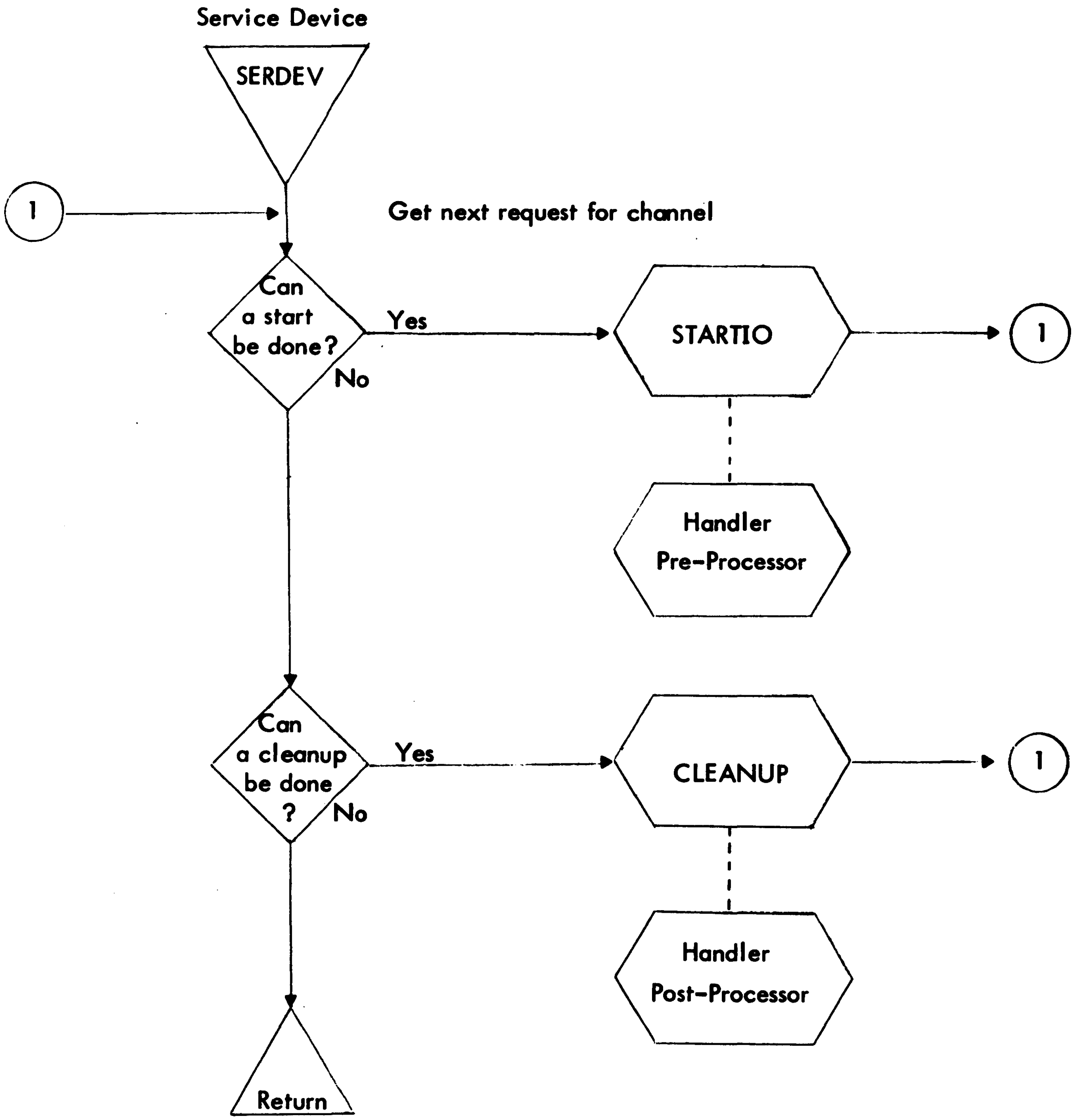
Service Device is a highly independent routine in the sense that it can be called at any time from any where in the monitor. It is called whenever there is any chance that a start or cleanup can be done for a given device. Some examples of when Service

Device is called are:

1. When a request is queued (start may be performed unless device is already busy).
2. After an I/O interrupt has occurred (cleanup may be done).
3. After a cleanup has been done (a start may be performed for the next request in the queue).

Device dependent routines are provided for building command lists and testing for errors. STARTIO calls the "handler pre-processor" to do the former, while CLEANUP calls the "handler post-processor" to do the latter. These two parts constitute the device handler for any given peripheral and are provided in separate assembly modules.


Make a Request
QUEUE
QUEUE1
NEWQ
SERVICE
DEVICE
Return
I/O Interrupt Occurs
IOINT
Change State
of device, clear
interrupt from
active state
SERVICE
DEVICE
Exit


Service Device
SERDEV
1
Get next request for channel
Can a start be done?
Yes
No
STARTIO
1
Handler Pre-Processor
Can a cleanup be done ?
Yes
No
CLEANUP
1
Handler Post-Processor
Return

PROCEDURES FOR MAKING REQUESTS

Requests for input/output may be placed in one of two ways: with all arguments contained in general registers or with most arguments residing in a DCB.

The caller may specify the address of a routine to be entered after the completion of any request (successful or not). This "end-action" routine will be entered after information pertinent to the outcome of the request has been loaded into registers or stored in the DCB.

Register formats will be indicated by listing the parameters contains therein followed by the field lengths of the respective parameters.

Register call:

BAL, R11	NEWQ	
R12	FC, PRI, NRT, DCT	(8, 8, 8, 8)
R13	D, C, -, BUF	(1, 1, 11, 19)
R14	-, SIZE	(16, 16)
R15	SEEK	(32)
RO	-, EA	(15, 17)
R1	EAI	(32)

The normal return is to BAL+2. If the device is marked down the return is to BAL+1 (not currently implemented). Registers 5 through 11 are considered non-volatile.

FC	New function code as described in DA.03.
PRI	Priority. Normally the current task priority (obtained from CJOB).
NRT	Number of recovery tries to be attempted.
DCT	Device control table index (described in Section VG).
D	Data chaining flag.
C	Command list flag.
BUF	D=0, C=0: byte address of buffer. D=1, C=0: doubleword address of data chain list. D=0, C=1: doubleword address of complete command list.
SIZE	D=0, C=0: length of buffer in bytes. D=1, C=0: number of commands in data chain list. D=0, C=1: time-out increment (see Service Device).
SEEK	Seek address for random access devices, left justified.
EA	Address of end-action routine. Zero indicates no end-action desired.
EAI	End-action information. Supplied by caller and returned at end-action time.

The caller's end-action routine is entered with interrupts enabled and all registers volatile:

```
    BAL, R11    EA
R7      -, DCT          (24, 8)
R12     TYC, -, RBC     (8, 8, 16)
R13     -, CCA          (16, 16)
R14     EAI             (32)
R15     -, BUF          (13, 19)
```

The caller must return via register 11.

TYC	Type of completion code returned by device handler (See BPM reference manual).
RBC	Remaining byte count (usually from TDV).
CCA	Current IOP command address (from TDV).

Other parameters are as described above. BUF and SIZE are the values supplied by the caller.

DCB call:

```
        BAL, R11    QUEUE     no end-action
        BAL, R11    QUEUE1    end-action
R8      FC, -, DCB      (8, 7, 17)
R9      -, EA           (15, 17)
R10     EAI             (32)
```

Registers 9 and 10 are not necessary on a call to QUEUE. For DCB calls, FC refers to the old handler function code as described in subsequent paragraph. The DCB must contain NRT, DCT, BUF, SIZE and SEEK. Registers 5 through 11 are considered non-volatile.

End-action is entered as above after the TYC and actual record size have been entered into the DCB:

```
    BAL, R11            EA
R6      -, BUF          (15, 17)
R7      -, DCT          (24, 8)
R8      FC, -, DCB      (8, 7, 17)
R14     EAI             (32)
```

In this case BUF is a word address. The other parameters are as in the call.

The old handler function code is interpreted as follows:

Bits: 0	1	2	3	4 7
0	DIR	PACK	FBCD	CODE

where

CODE = 0 - read BCD
1 - read direct BCD
2 - read binary
3 - read direct binary
4 - write BCD
5 - write direct BCD
6 - write binary (write and format)
7 - write direct binary
A - skip record forward
B - skip record reverse
C - skip file forward
D - skip file reverse
E - rewind
F - write end-of-file

(codes A–F: bits 0-3 are ignored for these codes)

FBCD = 0 - specifies no FORTRAN conversions
1 - specifies FORTRAN conversions

DIR = 0 - specifies forward direction
1 - specifies reverse direction

If the device is not 9T, 7T, or MT, only bits 5 thru 7 are meaningful.

CHANNEL CONCEPT

For the purposes of this specification let us define the term "channel" as: the highest order data path connected to one or more devices, only one of which may be transmitting data (to or from CPU memory) at any time.

Thus a magnetic tape controller connected to an MIOP is a channel. But one connected to an SIOP is not, for in this case the SIOP itself fits the definition. Other examples of channels are a card reader on a MIOP, a keyboard/printer on an MIOP or a RAD controller on an MIOP.

Input/Output requests made on the system are queued by channel. This method facilitates starting a new request on the channel when the previous one has completed. The exception to this rule is the "off-line" type of operation such as rewinding of magnetic tape or arm movement of certain moving arm devices. If this type of operation is started, an attempt is always made to start a data transfer operation as well. Thus the channel is always kept busy if possible.

SEPARATION OF PRIORITIES AND CONTROL TASK

All input/output functions are controlled with respect to time by a scheduler called "Service Device". This routine is device-oriented as far as the calling program is concerned, but in reality takes the necessary steps to keep the applicable channel operating within the constraints of priority.

This means that no request will be started whose priority is lower than that of the operating task, nor will an interrupt from a request be processed unless priority dictates. It must be realized that some overhead is suffered from the scheduler itself, but this overhead is considered to be small compared with starting a request or processing its interrupt.

Since requests on a channel are normally "chained" by the I/O interrupt, there must be a means whereby any action on a request which is deferred by priority may be resumed at a later time. This provision is the "Control Task", usually the lowest level external interrupt in the system. When action is deferred, the device code is entered into the Control Task stack and its interrupt is triggered. When it becomes active it will call the scheduler for the device in question. In a system created with no Control Task, the console interrupt will be triggered instead. The console interrupt receiver is designed to perform Control Task Functions when there is no external interrupt assigned for this purpose.

SYSTEM FLOW

As indicated above the center of I/O activity is the scheduler, Service Device. This routine starts all operations and processes their interrupts (cleanup). Thus Service Device must be called whenever certain key events occur or when other special conditions are present in the system. Figure 1 shows the downward flow of control from some of the most important areas of the I/O system.

SYSTEM TABLES

Information pertaining to requests, devices and channels is maintained in a series of parallel tables produced at System Generation Time. The format of these tables is presented in Section VG and will be referenced throughout the remainder of this specification. The first entry (index=0) in each table is reserved for special use by the system.

a) IOQ, Request Intormation
These tables contain all information necessary to perform an input/output operation. When a request is made on the system, data is transferred from the controlling DCB and/or registers into one element in each of the parallel IOQ tables. This set of elements forms a "queue entry". The entry is then linked into the channel queue below other requests of higher or the same priority.

b) DCT, Device Control
The device control tables contain fixed information about each system device (unit level) and variable information about the operation currently being performed on the device.

c) CIT, Channel Information
These tables are used primarily to define the "head" and "tail" of those entries which represent the queue for a given channel at any time. A channel queue may have more than one entry active at anytime (such as several tapes rewinding while another reads or writes).

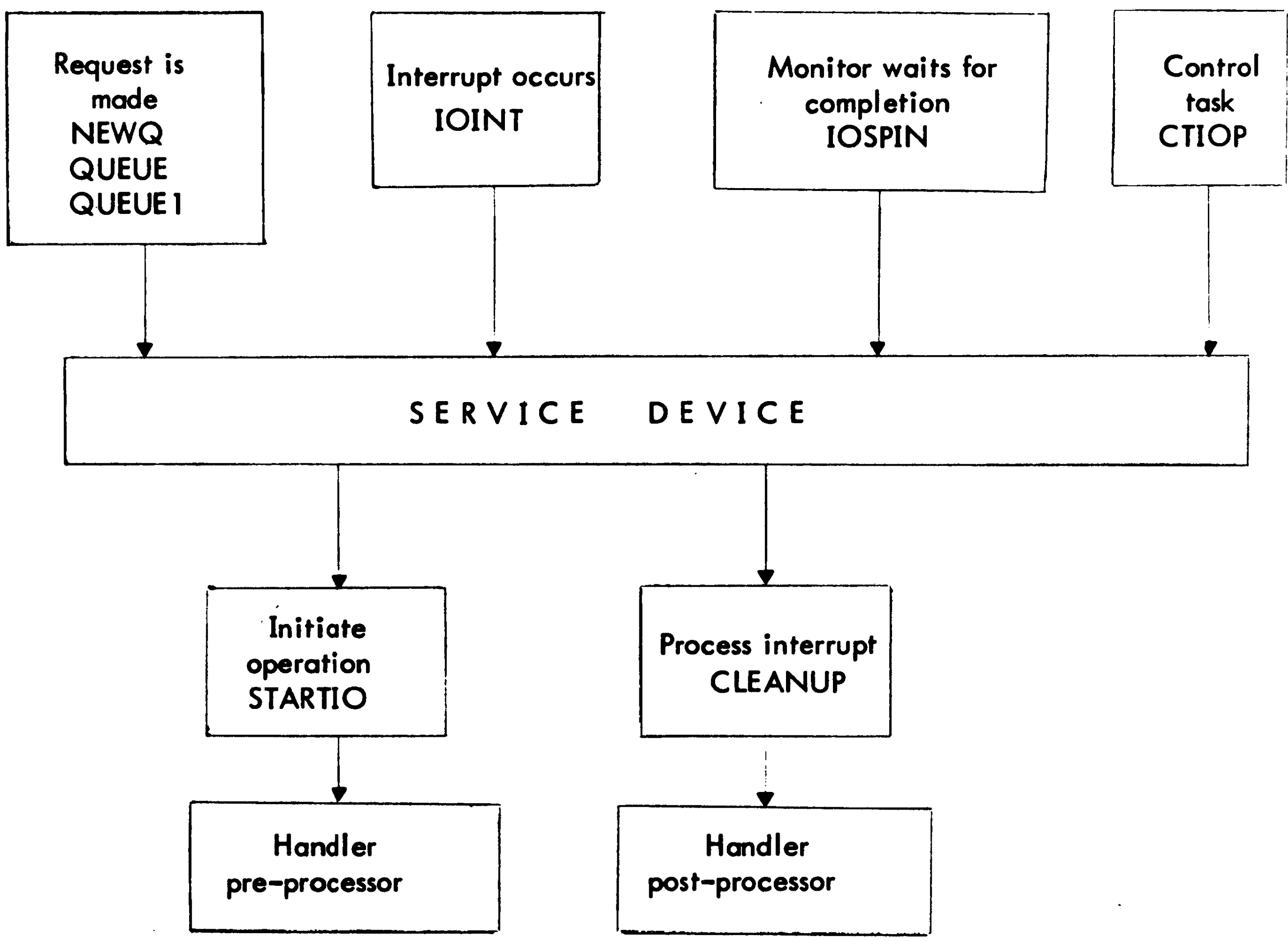


FIGURE DA-1 SYSTEM FLOW

DESCRIPTION OF ROUTINES

This section presents descriptions of the routines which comprise the I/O System. Only the most important functions of each routine are described. The listings should be consulted for more detailed information.

The handlers and related subroutines are described in later sections.

NEWQ

Purpose: to receive requests for I/O operations, register format.
Inputs: described in paragraph "Procedure for Making Requests".
Description:
The index of an entry in the IOQ tables is obtained (See GETQ) and the arguments passed in registers are properly formatted and stored into the respective tables. The queue entry is then linked (by priority) into the queue for the appropriate channel. Then Service Device is called and control is returned to the caller.

QUEUE, QUEUE1

Purpose: to receive requests for I/O operations, DCB format.
Inputs: described in paragraph "Procedures for Making Requests".
Description:
These routines are actually different entries to NEWQ. They differ only in the manner in which they build the queue entry — most of the arguments are obtained from the associated DCB. A set of byte tables is used to convert the old handler function code to a new handler function code.

GETQ

Purpose: to obtain the index of a queue entry from the pool of free entries.
Call: BAL, R11 GETQ
Inputs: none
Outputs: R3 = 0, IOQ index (24, 8)
Description:
The head of the free entry pool is contained in the byte QFREE. If QFREE is non-zero its contents are loaded into R3 and the second entry in the pool becomes the head. The free entries are linked forward by IOQ2, with the last entry having a forward link of zero. If the head is zero, Service Device is called for each device in the system until an operation completes causing an entry to be freed. This is done without regard for priority and is considered to be an emergency measure. GETQ will not exit until a free queue entry has been obtained.

There are two other constraints in GETQ. First, in a real-time system, a limit may be placed on the number of queue entries to be used by the background. If this limit is reached, all devices will be driven as above until the number of entries in use by the background is once again below the limit. Second, one queue entry is always reserved for the Operator's Console typewriter to assure that the operator is not cut off from communicating with the system.

IOSERV, IOFORCE

Purpose: to provide an entry to Service Device which does not destroy any registers.
Call: BAL, R11 IOSERV
BAL, R11 IOFORCE
Inputs: R12=0, DCT index (24, 8)
Description: (See Service Device)
IOSERV is called when normal considerations are to be given to the priority of the operations involved. If IOFORCE is called, the priority going into Service Device will be set to FF (lowest).

SERDEV (Service Device)

Purpose: to determine the state of the device and/or channel in question and to perform whatever action is possible within the constraint of priority.
Call: BAL, R2 SERDEV
Inputs: R1 = PRI, 0, DCT (8, 16, 8)
Description: (refer to flowchart)
The priority input (PRI) is normally the current task priority (from CJOB) and should be obtained by the caller just before entry. However, it may arbitrarily be set to other values under special conditions.

The scheduler, Service Device is basically device-oriented but will always attempt to "sequence" the channel (to which the device is assigned) before exiting. This means that the queue (for the channel) is examined to determine if any action for any device on the channel may be processed. In other words the scheduler will not exit until one of the following is true:

1. Queue is empty. There are no more requests for this channel at this time.
2. Channel is busy. Data is being transferred to or from a device on this channel.
3. Channel is being held. Channel status from a previous operation must be preserved.
4. There are no requests in the queue for this channel for which an operation may be started.

The fourth of these may be true even if the first three are not. Two example situations are when the devices for which there are requests in the queue are all busy (e.g. rewinding), or when the highest priority request which can be started has been deferred to the Control Task.

As can be seen in Figure DA-1 there are two major functions which must be performed for each I/O operation — start and cleanup. For a given device these must always be performed alternately. Thus a cleanup must be done for a previous operation before a new operation can be started. To elaborate on this part of the scheduler's operation, a number of device "states" will be defined, and the transitions into and out of each state will be explained.

a) Free
The device is free when it is not actively linked to any request in the queue. There is no specific condition for this which can be tested since the free state is actually the lack of any of the conditions described below.

b) Busy
A device becomes busy upon the successful execution of an SIO instruction in STARTIO. This is what has been referred to as a "start". If the SIO is not accepted by the IOP, then the device will not be busy upon exit from STARTIO.

c) Cleanup Pending
"Cleanup pending" means that some event has occurred which has made it necessary to remove the device from the busy state; normally this event is an I/O interrupt from the device. Others are the failure of an SIO or an operation halted for taking too much time. In any case it means a call must be made to the CLEANUP routine.

d) Keyin Pending
This state exists when it has been determined that no further action can be taken without a response from the system operator. The device remains in this state until the operator gives his answer, with the "PLEASE RESPOND" message periodically repeating itself on the typewriter. The transitions are cleanup pending to keyin pending, then keyin pending to free.

e) Inter-operation
This is really a special version of the free state and it means that the request to which the device is currently linked involves more than one operation (i.e. start and cleanup). And furthermore that no other request is to be linked to this device until they are all completed, regardless of priority. On a disk pack, for example, a request usually involves a seek (moving the arm), followed by a read or write. If a higher priority request were to intervene between the two operations it is likely that the read or write would be from the wrong place on the pack.

The primary function then of STARTIO in conjunction with the handler per-processor, is to change the state of the device from free to busy. And the main job of CLEANUP is to change the state from cleanup pending to free. The link between these two is the I/O interrupt (busy to cleanup pending).

When any operation is started or when an error message which is to be repeated is typed, a "time-out" is set up. A cell called IOCLOCK is continuously incremented every five seconds by the monitor's clock interrupt routine. When a time-out is initiated, the current contents of IOCLOCK plus some increment are saved in

DCT11. When Service Device is entered and the device is busy or has a keyin pending, this value is compared with the now current contents of IOCLOCK. If the time is up, the operation is terminated with an HIO instruction, or the "PLEASE RESPOND" message is repeated if a keyin was pending. If an operation is halted, the timed-out bit in DCT3 is set and the device is set waiting for cleanup.

STANDARD REGISTER SETUP

Reference will be made in later sections to a "standard register setup". This refers to the way in which some registers are generally used in Service Device, and in particular to the contents of registers at the entry to STARTIO or CLEANUP. The standard register setup is:

R1	PRI, -, DCT	(8, 16, 8)
R2	0, Link to SERDEV	(15, 17)
R3	0, IOQ index	(24, 8)
R4	0, CIT index	(24, 8)
R14	0, DAC	(16, 16)
R15	0, link	(15, 17)

The DAC in R14 is the "device activity count" used for making re-entrance tests (see STARTIO). The link in R15 is the link to STARTIO or CLEANUP.

The remaining registers are normally available in STARTIO and CLEANUP and in the handler pre-processor and post-processor, although some are used for handler communication (see STARTIO and CLEANUP).

CTEST

Purpose: to perform priority tests for Service Device
Call: BAL, R15 CTEST
Returns: BAL+1 if processing is to be deferred.
BAL+2 if processing may continue.
Description:
CTEST is called by Service Device whenever it is about to perform a start or cleanup. If the priority of the request (IOQ14) is lower than the priority being carried by Service Device (in R1), then the processing of the start or cleanup is deferred to the Control Task.

Priorities X'F0' through X'FF' are all considered background priorities, and deferments are never made when R1 is in this range.

CTRIG

Purpose: to trigger the Control Task interrupt after notifying the Control Task of some impending action.

Call: BAL, R11 CTRIG
Inputs: R8 code, -, DCT (8, 16, 8)
Description:
A Control Task stack is established at Sysgen time by the formula: number of devices plus number of tape drives plus two. This is the minimum number of entries required to prevent overflow.

CTRIG pushes the contents of R8 into the stack and triggers the Control Task interrupt or console interrupt in a non real-time system. The codes are:

0 (with DCT)	defer start or cleanup for this device.
1 (no DCT)	operator has pressed console interrupt.
2 (with DCT)	operator has pressed attention on tape drive.
3 (no DCT)	operator has completed input for an unsolicited keyin
4 (no DCT)	call Service Device for all devices which are busy or have cleanup pending (this entry is made by the system clock routine every 5 seconds).
5 (no DCT)	keyin is busy when console read is complete.

STARTIO

Purpose: to initiate all I/O operations.
Call: BAL, R15 STARTIO
Inputs: standard register setup.
Description: (flowchart included)
The primary function of the handler pre-processor is to build the IOP command list to be used for a given operation. The handler is entered by a branch to the address in DCT8 with the standard register setup. When the command list is built, the handler returns to STARTIO by a branch to IOSST, passing the following information:

R0:	doubleword address of command list.
R4(bit 0):	a flag set to indicate that the channel is not to be set busy for this operation. Usually this means that the operation does not tie up the device controller which is free to be used by another device attached to it. Examples are rewinding tape and disc pack seeks.
R4(bit 1):	a flag set to indicate that the DCB function count should be decremented at start time rather than at cleanup time. This bit is used only when bit 0 (above) is set and prevents the system from having to wait for tape rewinds before proceeding to the next job step.
R4(bit 2):	Channel is to be held.
R10:	word address of handler DOT table (see Handler Interface section).

When the handler returns to STARTIO at IOSST, all interrupts are inhibited. This is called the Disable Point (there is a similar place in CLEANUP). The inhibits are not removed until a number of critical actions have been performed. This is necessary to prevent the device from taking on an undefined software state and then having an interrupt occur. If I/O were attempted on the same device at the interrupt level the scheduler might be confused by an abnormal combination of factors.

Following the Disable Point is a "re-entrance test". This is done to determine if the device has been used by a program at a higher interrupt level. If it has, the start is aborted. The interrupt may have occurred any time between the time the scheduler decided to perform the start and the Disable Point. This concept is best illustrated with an example listing the execution of key events with respect to time:

1. Low level request is made.
2. Scheduler decides to start request.
3. Current Device Activity Count (DAC, from DCT10) is loaded into R14.
4. Scheduler calls STARTIO
5. Handler pre-processor begins building command list.
6. Interrupt occurs.
7. High level request is made by interrupt program (same device).
8. 2 through 5 above are executed (for high level request).
9. Handler returns to Disable Point.
10. Re-entrance test. R14 is compared with value in DCT10. There is no change, R14 = DCT10.
11. Device is started (SIO etc.).
12. DAC is incremented by 1.
13. Interrupt program exits.
14. Control returns to 5 at the lower level.
15. 9 and 10 are executed again, but this time R14 is one less than the contents of DCT10.
16. Start is aborted.

It would appear that at the higher level the scheduler was unaware of the activity at the lower level. This is exactly the case. Until the Disable Point is reached, no parameters in any of the tables may be changed in any way to indicate that a start is in progress. And if it is necessary to store into scratch areas, such as storing command doublewords, a re-entrance test must be made before the actual storing into core. This is to prevent storing over information prepared at a higher level.

Thus the handler pre-processor must make a re-entrance test before it stores each command doubleword into core. This is done by comparing R14 with DCT10 and aborting the start if they are unequal.

In some handlers it may be absolutely necessary to modify some table parameter before returning to IOSST. In this case the handler may extend the Disable Point backwards by inhibiting interrupts and making a re-entrance test (aborting if reentrant). The handler must leave the interrupts inhibited when branching to IOSST. An abort is accomplished by executing a: B *R15, with interrupt inhibits off.

There are three things that can happen after the Disable Point has been passed (and the start is not aborted due to re-entrance).

1. SIO is accepted and device is automatic — a successful start.
2. SIO is accepted but device is in manual mode. A message is output to the operator and repeated every 30 seconds until he starts the device. The start is otherwise successful.
3. SIO is rejected. The SIO failures bit in DCT3 is set and the device is set waiting for cleanup. When the scheduler calls CLEANUP the operator will be notified and must decide whether the operation should be retried or if it should be aborted (i.e., indicated as unrecoverable to the caller).

IOINT

Purpose: to process all I/O interrupts
Call: entered via XPSD in location X'5C'.
Description: (flowchart included)
The first portion of the I/O interrupt receiver is executed with the interrupt in the active state and is non re-entrant. (If the interrupt is from the swapping RAD, then control passes to T:SIOEA, the monitor swap end action handler.) The DCT index is determined from the AIO data by searching DCT1. If the device was not busy and AIO status bit 1 is set, then it is assumed that the interrupt was caused by the operator pressing the attention switch on a tape drive. In this case, the Control Task is notified to perform an "AVR" sequence. Otherwise the states of the device and channel are appropriately modified and the AIO and TDV status information is saved in DCT tables.

After the interrupt is cleared the scheduler is called for the device in question. If the priority in CJOB is background, then the Symbiont Activate routine (SACT) is called.

An error is reported in the System Error Log if the device was not busy and AIO status bit 1 was not set. An error is also reported if the AIO indicates no interrupt recognition.

Exit is to T:SSE, the scheduler entry point for asynchronous events.

CLEANUP

Purpose: to perform the post-interrupt processing for any I/O operation.
Call: BAL, R15 CLEANUP
Inputs: standard register setup
Description: (refer to flowchart)
CLEANUP enters the handler post-processor at the address specified in DCT9. The handler must examine the information available (in the DCT tables primarily) and

decide what action is to be taken by CLEANUP. The alternatives are:

1. Normal completion. Complete request and report completion to caller via DCB and/or end-action.
2. Operation is in error. Decrement retry count and set request not busy (in IOQ3). This prepares the request for another pass through the system (start and cleanup). If the retry count is exhausted, the request is to be completed. In any case a message is to be typed if requested.
3. There is "follow-on". The handler must perform another I/O operation in order to complete the request. The request is set not busy.
4. A keyin is required. The device is set to the keyin pending state and the request is left hanging until the operator responds (see IOREC).

The handler communicates its wishes via registers:

R10	-, CCA	(16, 16)
R11	-, RBC	(16, 16)
R12	-, flags, TYC	(16, 8, 8)
R13	0, MSG	(15, 17)

The flags are:

- Bit 16: retry. Alternative 2 above is to be taken.
- Bit 17: follow-on. Alternative 3.
- Bit 18: inter-op. If bit 16 or 17 is set, set the inter-op bit in DCT5 (see Service Device).
- Bit 19: keyin required. Alternative 4.
- Bit 20: keyin required. This is the same as bit 19 except that the response "C" is not allowed and will be taken to mean "R" (see IOREC).

MSG is the word address of a message to be typed following the device name. This is used with alternatives 1, 2, and 4 (see MSGOUT). The other parameters are described in "Procedure for Making Requests".

If the request is to be completed the subroutine REQCOM is called (see next section).

The re-entrance considerations mentioned in the section on STARTIO apply to the handler post-processor as well. The handler returns to CLEANUP at the address IOSCU, the Disable Point. The handler must make re-entrance tests whenever changing table parameters or storing into scratch areas. It may push the Disable Point back as described in STARTIO.

REQCOM

Purpose:	to perform the final cleanup of a completed request.		
Call:	BAL, R5	REQCOM	
Inputs:	R10,	-, CCA	(16, 16)
	R11,	-, RBC	(16, 16)
	R12,	-, TYC	(24, 8)

Description:
For a register call, REQCOM releases the queue entry back to the pool of free entries and executes the end-action routine.

In addition, for a DCB call, it communicates a number of parameters to the caller via the DCB:

TYC	the type of completion, if greater than the current value in the DCB, is stored.
FCN	the function count is decremented.
EGV	the EGV bit is set to 0.
ARS	the actual record size is computed by subtracting the RBC from the caller's byte count (only if request was not for a RAD or tape file).

If a Monitor Buffer was used, it is released if the following are all true:

1. Request was not to perform a position operation.
2. Request was not for an input operation.
3. Request was not for a file operation (ASN≠1).

OCINT

Purpose: to process control panel interrupts.
Call: entered via XPSD in location X'5D'.
Description: (flowchart included)
If the interrupt was caused by triggering the Control Task (non real-time system) the Control Task I/O Processor is called after the interrupt lavel has been cleared (see CTIOP).

If the operator has pressed the console interrupt switch the keyin sequence is initiated. This sequence consists of the following steps:

1. Trigger Control Task for keyin (code 1 CTRIG).
2. Control Task becomes active, makes requests to output and to input up to 72 characters from the Operator's Console, the latter with end-action.
3. End-action occurs for input. Trigger Control Task to process keyin (code 3, CTRIG).
4. Control Task becomes active, calls KEYIN overlay to process keyin.

CTIOP

Purpose: to process Control Task I/O functions.
Call: BAL, R11 CTIOP
Description:
Since the I/O and control panel interrupts are generally of higher priority than the interrupts of real-time tasks, it is necessary to take steps to prevent the loss of CPU processing time from these tasks for lower priority functions. These latter may be listed as:

1. Performing start or cleanup for requests of lower priority than the currently operating task.
2. Processing unsolicited keyins from the operator.
3. Labeled Tape recognition (initiated by operator pressing attention switch, also called AVR).
4. Periodic checking of all devices for time-out purposes.

CTIOP will operate until its stack (IOCTQ) is empty, at which time it will reset bit 31 of CTFLAGS (set by CTRIG). This flag is used by the main Control Task processor (or OCINT in non real-time) to decide when to call CTIOP.

The functions performed by CTIOP are described in the sections on CTRIG and OCINT.

IOREC

Purpose: to handle operator communications for I/O devices.
Call: entered from main keyin processor.
Inputs: R7 0, DCT (24, 8)
Description:
When the I/O system requires operator assistance, it outputs the name of the device in question followed by a message indicating the problem. Messages for which a response is mandatory (via a keyin) are:

ERROR (non-automatic recovery devices only)
TIMED OUT
NOT OPERATIONAL
WRITE PROTECTED

The device name followed by PLEASE RESPOND is output periodically until a response is received. The response is in the form: yyndd, X where X may be C, E, or R. The UTS Operations Manual should be consulted for complete explanations of the messages and responses.

IOKEC resets the keyin pending flag and sets up the registers as required for entry to REQCOM. If the response is C or E it branches to KYIO1, if R it branches to KYIO2, effecting a call to REQCOM and SERDEV or just SERDEV respectively.

MSGOUT

Purpose:	to output I/O System error messages.		
Call:	BAL, R5	MSGOUT	
Inputs:	R1	-, DCT	(24, 8)
	R3	0, IOQ	(24, 8)
	R13	0, MSG	(15, 17)

Description:
Messages are output in the form: yyndd message. The message (MSG) should have a blank as its first character.

A request is made on NEWQ using the priority of the request associated with the error. The DCT index is passed in R15 (normally a seek address) and thus gets placed in IOQ12. A special function code of the typewriter handler (02) will chain the device name from DCT16 to the message and output the entirety in one operation.

OCQUEUE

Purpose:	to output typewriter messages for certain routines		
Call:	BAL, R11	OCQUEUE	
Inputs:	R1	Code	(32)
	R7	0, DCT	(24, 8)

If the DCT index in R7 is zero the message is output alone with no device name. Otherwise the message format is the same as for MSGOUT. The codes for messages now available are:

1. KEYERR
2. AVRERR
3. LATER
4. EH?
5. AVAIL
8. SYMB NOT ACTIVE
9. SYMB ACTIVE
10. SYMB NOT SUSP
11. SYMB NOT AVAIL
12. SYMB SUSPENDED
13. SYMB TERMINATED

The last group, 8 through 13, is used by the symbiont routines.

HANDLER INTERFACE

The handler has two primary functions:

1. build command list (pre-processor)
2. examine results after interrupt (post-processor)

The register inputs and outputs of these routines and the re-entrance restrictions placed on them are described in detail in the sections on STARTIO and CLEANUP.

A number of subroutines are available in the Standard Handler Package to aid any handler in performing its functions. These routines are discussed in the following paragraphs.

COMLIST

Purpose: to build a command list using information contained in a set of special tables.
Call: B COMLIST
Inputs: standard register setup plus:
R10 -, DOT (15, 17)
Description:
Three tables are used on a call to COMLIST:

1. Device Operation Table (DOT).
2. Command List Table (CLIST).
3. Dummy commands.

The DOT table is an ordered word table containing one entry for each function code allowed by the handler, beginning with zero. The first word in the DOT is usually given a label and its value is the address passed in R10 on the call. This label will subsequently be referred to as "DOT". Each word in the table is broken into four 8-bit fields as follows:

Byte 0: The offset, in bytes, from DOT (first entry in DOT table) to the first byte of a list of bytes describing the command list to be built. (CLIST table)
Byte 1: The number of 5-second increments allowed to complete the operation before it is timed-out by the scheduler.
Byte 2: a function code which becomes the current function step (IOQ5) if retry is specified by the post-processor (see CLEANUP).
Byte 3: a function code which becomes the current function step if follow-on is specified by the post-processor.

The two function codes are picked up by STARTIO and saved in DCT17; they are retrieved by CLEANUP after the return from the handler post-processor. The handler may modify the contents of DCT17 if it deems necessary, but must extend the Disable Point back so that it comes before the store into DCT17.

The CLIST table consists of strings of bytes where each byte is the double-word offset from DOT to a dummy command doubleword. The first byte of each string has a label which is referenced by byte 0 of one of the DOT entries. Each string describes a complete command list for some operation with the command doublewords replaced by bytes to save space.

The dummy commands are used to build the actual commands and are very similar in appearance:

word 0:	order, 0, address	(8, 5, 19)
word 1:	flags, 0, function, count	(8, 8, 8, 8)

COMLIST assembles the commands specified in the CLIST table according to the function specified (word 1) and stores them in order into the command list buffer designated for the device (DCT7). A re-entrance test is made before storing. Each function will be explained along with the required contents of the other parameters in the dummy command.

function 00:
Store command as is. The presence of the function byte restricts the count field, but this function is usually used for tape spacing operations and the like which have no byte count anyway.

function 01:
Build seek command. The order, flags and count must be correct for the particular device. COMLIST computes the byte address of the IOQ12 entry and stores it into the address field of the command.

function 02:
Build data transfer command. The address and count fields are obtained from IOQ8 and IOQ9 respectively. The order and flag fields are used as is.

If data chaining is specified (bit 0, IOQ8) the normal data transfer command is not built. Instead a Transfer in Channel (TIC) command is inserted which will transfer IOP control to the caller's data chain list. The doubleword address of this list is found in IOQ8, while the number of commands in it is contained in IOQ9. The byte address and count must be supplied by the caller in each command, while COMLIST supplies the order and appropriate flags (the order used is the one in the dummy command which initiated this function). Flag bit 7 (the skip flag) is left unmodified and must be supplied by the caller. This feature of the IOP can be used to skip portions of an input record or to fill portions of an output record with zeros. There is no provision for having more commands after the data transfer (i.e., it should be the last item in the CLIST table). Also the individual handler should be examined to determine if this feature is usable. Some handlers do not use COMLIST at all.

function 03:
Build device name command. COMLIST computes the byte address of the DCT16

entry for the proper device (the DCT index is found in IOQ12, see MSGOUT) and stores it into the address field of the command. The order and flags are used as is and the byte count should be 8. This command is normally followed by a data transfer command to output the message part of an I/O System error message.

function 04:
Return to handler. In this case the address portion of the dummy command specifies a program address in the handler. When this command is encountered by COMLIST it branches to the specified address, thereby enabling the handler to take some special action (i.e., perform some function not provided by COMLIST). When the handler is entered the registers contain the following information (except for the command in R8 and R9 no register should be disturbed unless it is in the "open" list below):

R6	current CLIST table offset
R7	current command list area pointer (where next command will be stored).
R8	dummy command (word 0). The address field will have been set to all zeros.
R9	dummy command (word 1). The function byte will have been set to zero.
R10	DOT address.

Open registers: R10, R5, R11, R12, R13. The remaining registers are as in the standard register setup.

After the handler has done what it will with the command, it must return by branching to one of three re-entry addresses in COMLIST:

USECOM:	store command as is and go on to next.
DELCOM:	do not use this command at all, go on to next.
DEPCOM:	a new function byte has been placed in the command — repeat the test of the function byte and act accordingly.

COMLIST is finished after it has processed a dummy command which has neither the data chain flag nor the command chain flag set in the flag field. This means that all commands but the last must have at least one of these flags set. At this point, the doubleword address in DCT7 is loaded into R0 and COMLIST branches to IOSST. Control is not returned to the handler pre-processor.

Refer to the listings of existing handlers for examples of table structure and the use of assembler features which facilitate the construction of the tables.

IOSERCK

Purpose: to test for and report common device error conditions.
Call: BAL, R9 IOSERCK

Inputs: standard register setup.
Returns: BAL+1 if error detected.
BAL+2 if no error.

Description:

IOSERCK acts in one of four ways depending upon various status information:

1. SIO failure bit in DCT3 is set (see STARTIO). The condition is logged in the System Error Log. Bits 18 and 20 in R12 are set and the address of the NOT OPERATIONAL message is put in R13 (see CLEANUP). A branch is made directly to IOSCU.

2. Timed-out bit in DCT3 is set. The same is done as for (1) except that the message is TIMED OUT and bits 19 and 20 in R12 are set.

3. Any of TDV status bits 9 through 14 are set. These bits of the Operational Status Byte are common to all devices and indicate that some sort of malfunction occurred when transmission was attempted. A device error is logged. The retry bit and TYC = 8 are set in R12; the address of the ERROR message is put into R13. Return is to BAL+1.

4. None of the above. Return is to BAL+2 with the following in registers:

R5	-, TDV status	(16, 16)
R6	-, AIO status	(16, 16)
R10	-, CCA	(16, 16)
R11	-, RBC	(16, 16)
R12	1 if normal	
	2 if lost data	

In R5 and R6 the status includes the Device Status Byte and the Operational Status Byte. Lost data means that the remaining byte count was zero and the incorrect length bit in the TDV status was set (i.e., the caller provided a buffer which was shorter than the actual record).

IOSEREC

Purpose: to log an error detected by the handler.
Call: BAL, R9 IOSEREC
Inputs: standard register setup.

Description:

For any device there may be device dependent conditions which are not detected by IOSERCK. If the handler determines that any such condition should be classified as an error, it calls IOSEREC to have the error entered into the System Error Log. The return and registers are as for (3) in IOSERCK.

RE:ENT

Purpose:	to make a reentrance test.
Call:	BAL, R0 RE:ENT
Inputs:	standard register setup.
Returns:	BAL+1 Not reentered
	B *R15 Reentered

Description:
The reentrance test consists of comparing R14 against the current Device Activity Count in DCT10 (see STARTIO). If they are equal the return is to BAL+1 with all interrupts inhibited. If not (i.e., reentrance has occurred), the start or cleanup is aborted by returning on R15.

4CHAR

Purpose:	to load the first four bytes from the caller's buffer into a register.
Call:	BAL, R5 4CHAR

Description:
Starting at the byte address in IOQ8, the first four bytes are loaded into R0. This routine is used when the caller's buffer is not necessarily on a word boundary.

HANDLER DESCRIPTIONS

Typewriter Handler

Operation: The typewriter handler accepts the following function codes:

- 0 - read with editing
- 1 - write
- 2 - write with device name
- 3 - read without editing
- 4 - read with editing and retry
- 5 - write new line character
- 6 - write with device name tabbed

The pre-processor loads R10 with the DOT address and branches to COMLIST. Only the read-with-editing function has any special post-processing. When the post-processor obtains control from CLEANUP, the last character typed is examined to see if it is an EOM (X'08'). If so, a "new-line" character is output and the typewriter is enabled for input again. This, in effect, erases what was typed previously and allows the operator to start over again. If the maximum character count is reached, the message is taken and processed as is. Finally a check is made for !EOD as the first four characters. If present, the type completion code (TYC) is set to six. None of the other functions have any special post-processing. In no case is error checking or error recovery attempted for typewriter operations.

RAD Handler

Operation: The RAD handler accepts the following function codes:

- 0 - seek-read
- 1 - seek-write
- 2 - sense
- 3 - seek-checkwrite
- 4 - seek-write, seek-checkwrite

Error recovery on the RAD generally amounts to redoing the same operation when an error has been detected. One exception is when a check-write is being performed for a write and an error is indicated. In this case the write is done over, followed by another check-write. Check-writes are performed for all writes if sense switch 1 is set on the operator's console. Special conditions checked for are write violation and illegal seek address.

9 Track Tape Handler

Operation: The 9 track tape handler accepts the following function codes.

- 0 - read
- 1 - write
- 2 - read reverse

3 - write tapemark
4 - backspace record
5 - forewardspace record
6 - backspace file
7 - forewardspace file
8 - rewind
9 - sense
10 - correctable read recovery
11 - non-correctable read recovery
12 - write recovery
13 - correctable read reverse recovery
14 - non-correctable read reverse recovery
15 - write tape mark recovery

Most operations are straightforward. A special feature allows the caller to space multiple records (forward or reverse) on one forespace or backspace call. The high-order halfword of the seek address field in the calling sequence (QUEUE or NEWQ) is used to indicate the number of records to be spaced over (should be zero or one for a single record). The spacing is always terminated when a tape mark is passed or the load point is encountered. Correctable read recovery consists of rereading the offending record using the Sense, Set Correction, Read sequence of orders. Non-correctable read recovery consists of re-reading the offending record. Write recovery is always preceded by erasing a fixed amount of tape before writing the record again.

The following is a list of special conditions detected by the handler, and resulting actions:

1. Write protect error. Operator is notified and must correct the problem (put in write ring) or abort the operation (with "E" key-in).
2. Tape mark (EOF). Type of complete is set to six.
3. Beginning of tape. Type of complete is set to three.
4. End of tape. Type of complete is set to five.

7 Track Tape Handler

Operation: The 7 track tape handler accepts the following function codes:

0 - read packed
1 - write packed
2 - read reverse packed
3 - write tape mark
4 - backspace record
5 - forewardspace record
6 - backspace file
7 - forewardspace file
8 - rewind
9 - read binary

10 - write binary
11 - read reverse binary
12 - read decimal
13 - write decimal
14 - read reverse decimal
15 - read packed recovery
16 - write packed recovery
17 - write tape mark recovery
18 - read binary recovery
19 - write binary recovery
20 - read decimal recovery
21 - write decimal recovery
22 - final backspace record for reverse read
23 - final backspace record if unrecoverable error

The 7 track tape handler uses the existing 9 track tape handler code wherever applicable. Refer to the 9 track tape handler writeup for a description of those items that are applicable to 7 track tapes (e.g. recovery, spacing multiple records, etc.).

Card Reader Handler

Operation: The card reader handler accepts the following function codes:

0 - read binary
2 - read automatic

When a call is made to read a card, the mode of the read (automatic or binary) is always determined by the mode bit in DCT5. This bit can be changed directly in DCT5 by any routine in the monitor. It is also changed by the presence of a !BIN or !BCD card. These cards are used specifically for this purpose and are not passed to the caller. The !BIN card must precede any deck of non-standard binary cards, and the !BCD card must follow this deck to return the handler to the automatic mode.

A special check for the unusual end interrupt bit in the AIO operational status byte is performed and if set, a call to IOSEREC is made to log the error (TYPE = 05) and bit 19 of register 12 is set (see CLEANUP).

If a !EOD card is read in either mode, the TYC is set to six.

Line Printer Handler

Operation: The line printer handler accepts the following function codes:

1 - write without format
3 - write with format

The pre-processor tests for the following three conditions:

1. Is the function "print with format?"

2. Is the format byte a "top of form?"
3. Is the printer at top of form now?

If the answers are all "yes", the result will be a blank page in the listing. Therefore the format byte, X'F1', is replaced with X'C0', to suppress the extra page.

If an error is detected during transmission, the recovery procedure is to re-transmit the line. If the error occurs during printing, then an operator response is required to resume printing.

A special check for the unusual end interrupt bit in the AIO operational status byte is performed and if set, a call to IOSEREC is made to log the error (TYPE = 05) and bit 20 of register 12 is set (see CLEANUP).

Paper Tape Handler (PTAP)

Operation: The paper tape handler accepts the following function codes:

0 - read automatic
1 - write BCD
2 - read count
3 - write binary
4 - read direct
5 - write direct
6 - read BCD
7 - read binary

The formatted write operations (write binary, write BCD) have two null characters (X'00') appended via a data chain operation. In the case of write binary, the output record is preceded by a one-byte indicator (X'11') and a two-byte record count.

On a read automatic operation, the indicator byte is first read into the caller's buffer (obtained from IOQ8) ignoring leading null characters. If binary is indicated, the record count is read into scratch space in the command list area, and the entire record is read into the caller's buffer. If BCD is indicated, the record is read in one byte at a time until an EOM, NL, or null character is encountered. In the case of an EOM, the follow-on code is reset to read automatic which, in effect, erases the current record and reschedules input of the next record. If the caller's buffer is not large enough to contain the entire record, the excess position is skipped and the TYC code is set to indicate lost data. Finally, a check is made for !EOD as the first four characters. If present, the TYC code is set to indicate end of data.

Card Punch Handlers

Operation: The card punch handlers accept the following function codes:

0 - punch BCD
1 - punch binary

For the high-speed card punch there are two buffers, located in the command list area pointed to by DCT7. Thus the last two card images are available at all times. This is necessary since the punch "read-checks" the last card punched while it is punching the current card. If there is a read-check error, the bad card is directed to the error stacker, where it is repunched, and the card that was being punched when the error was detected is also directed to the error stacker to be repunched while the card originally in error is once again read-checked. The net result of a read-check error recovery is a good deck in the normal stacker and two cards in the error stacker.

A transmission error on the card being punched will result in that card being repunched, with the bad card directed to the error stacker. This results in only one card appearing in the error stacker.

The low-speed card punch handler does no special processing or recovery. In particular, lost data is ignored.

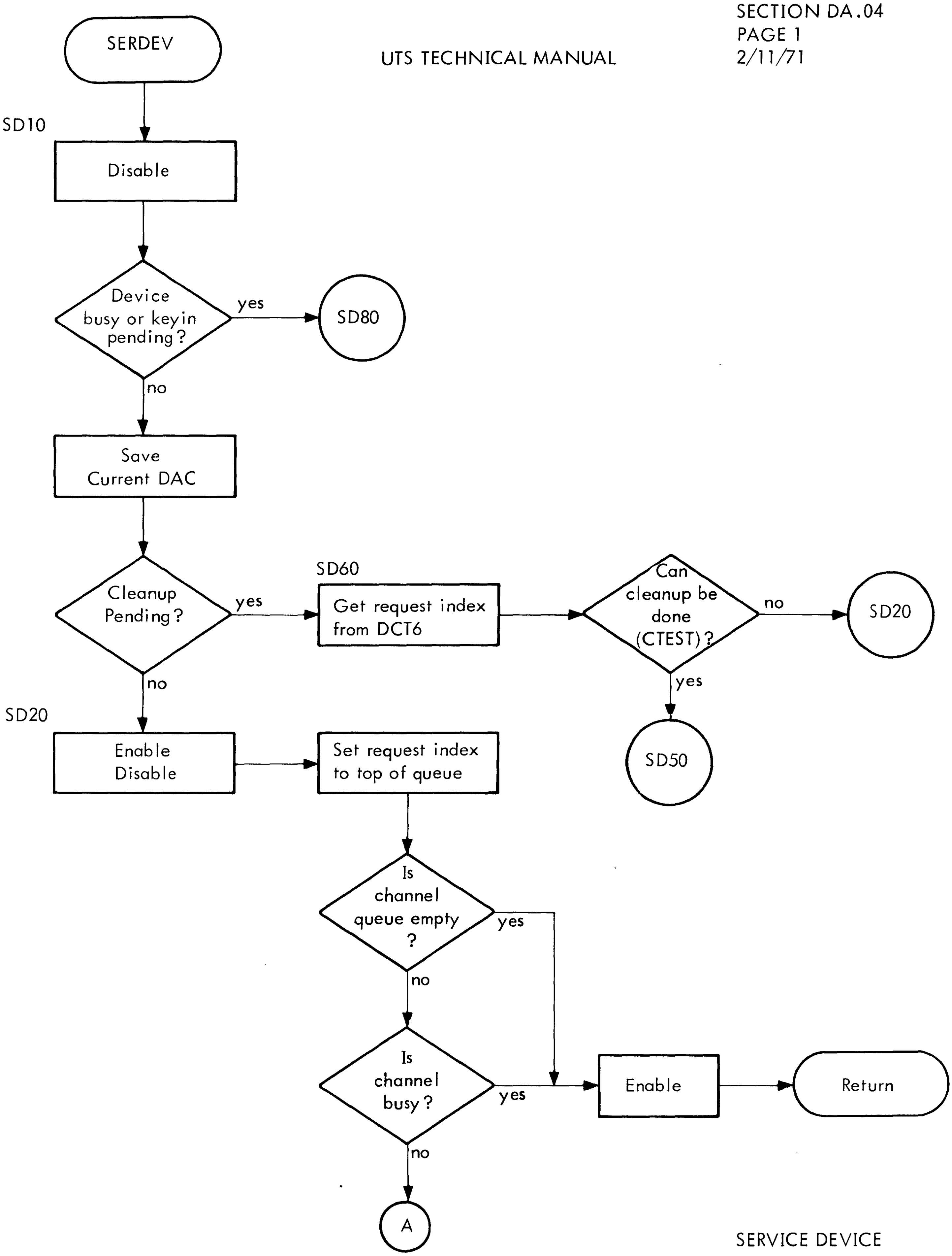
Disk Pack Handler (DPAK)

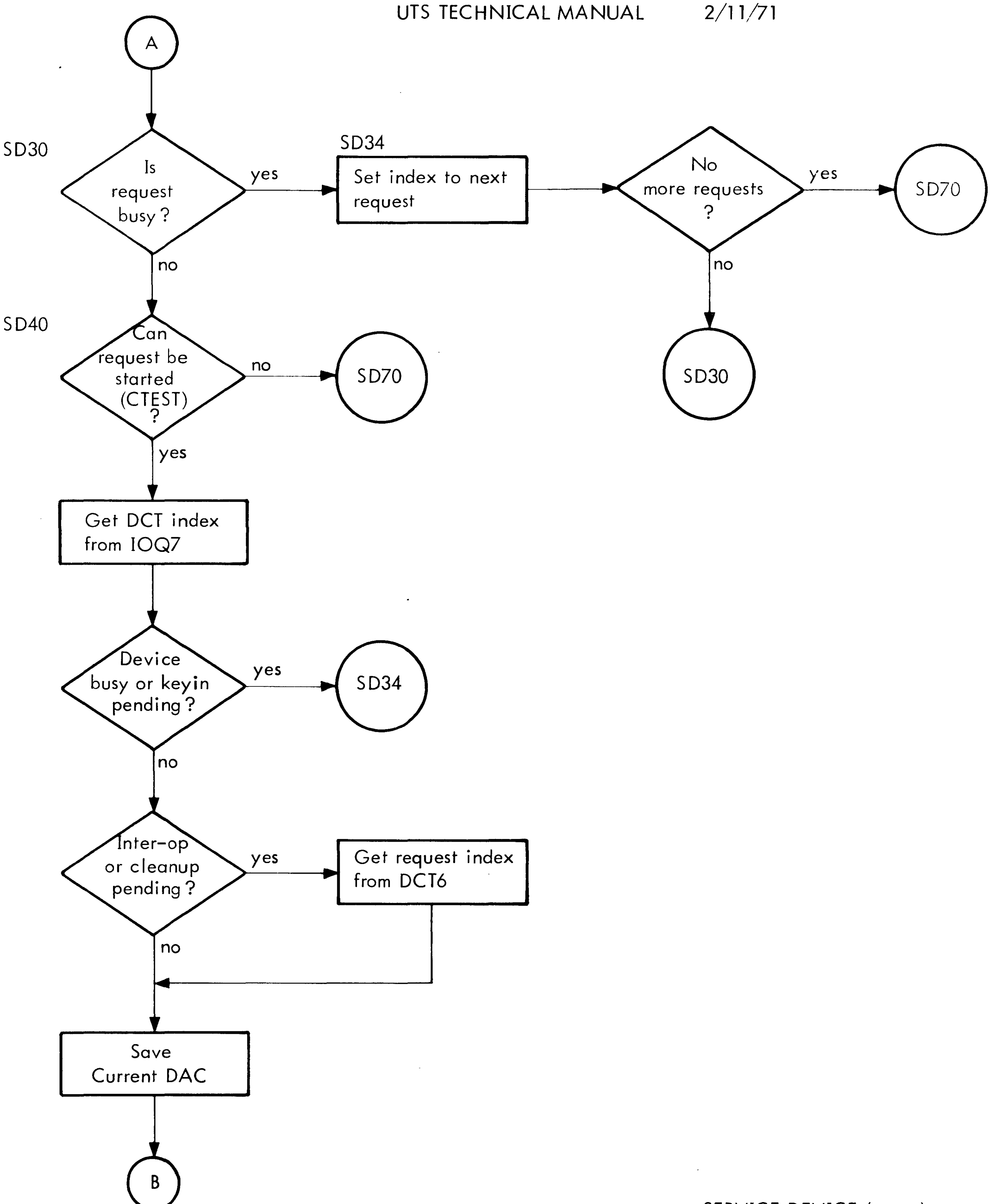
Operation: The disk pack handler uses the following function codes:

0 - seek-read
1 - seek-write
2 - sense
3 - seek-checkwrite
4 - read
5 - write
6 - checkwrite
7 - restore
8 - seek-read header
9 - read header

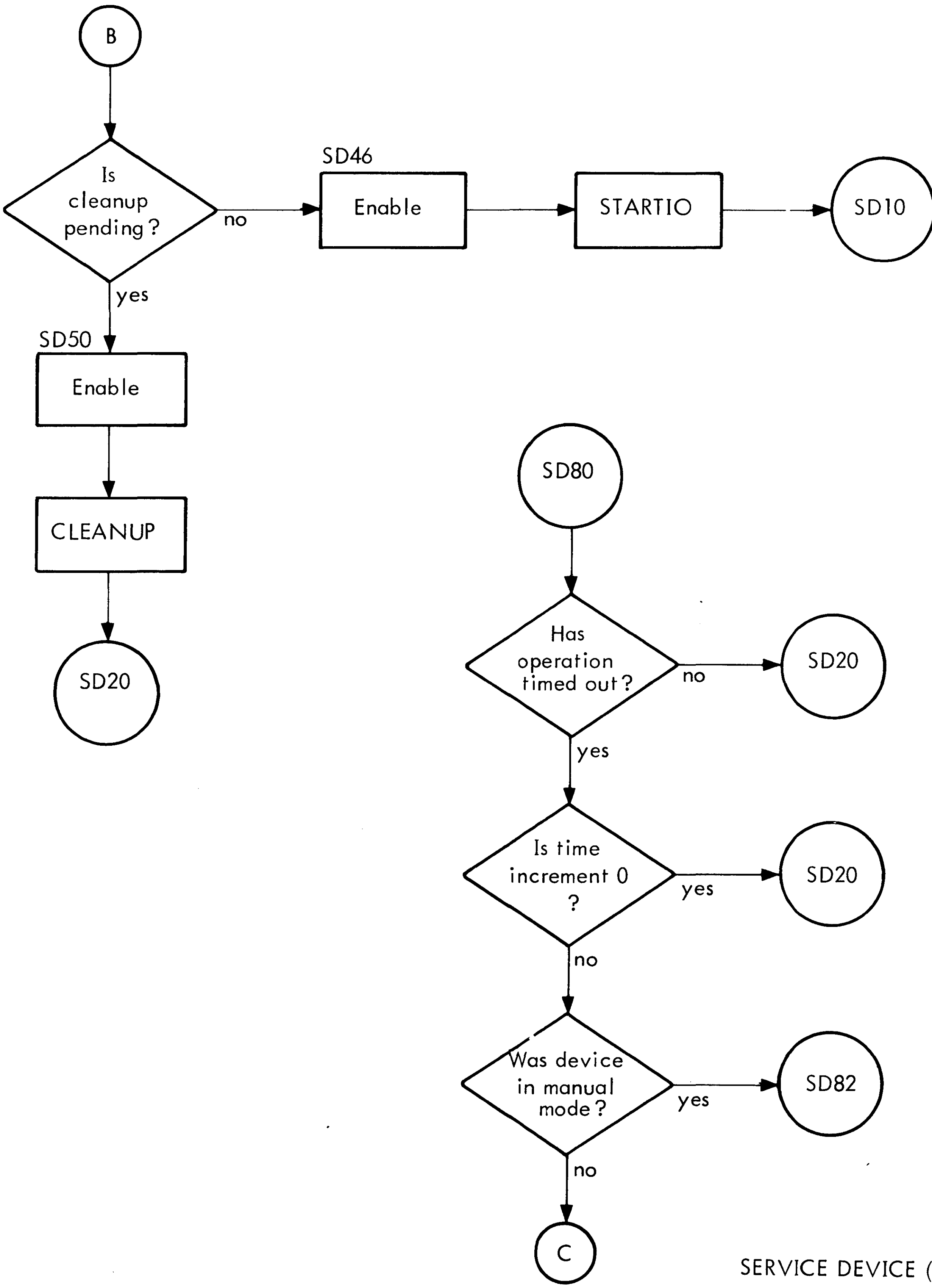
A restore carriage order is specified for follow-on in the event of an error on a seek address or a header verify or parity error associated with a data transfer order. If a flaw mark has been detected during a data transfer operation indicating a bad track, a seek-read header sequence is initiated in order to pick up the alternate track, and the caller's seek address in IOQ12 is altered. On a seek-write operation, a seek-checkwrite follow-on sequence is performed if sense switch 1 is set.

A header verify or parity error on a read header command and three successive seek/restore errors are considered fatal and the system recovery routines are invoked. (Software Check - FF).

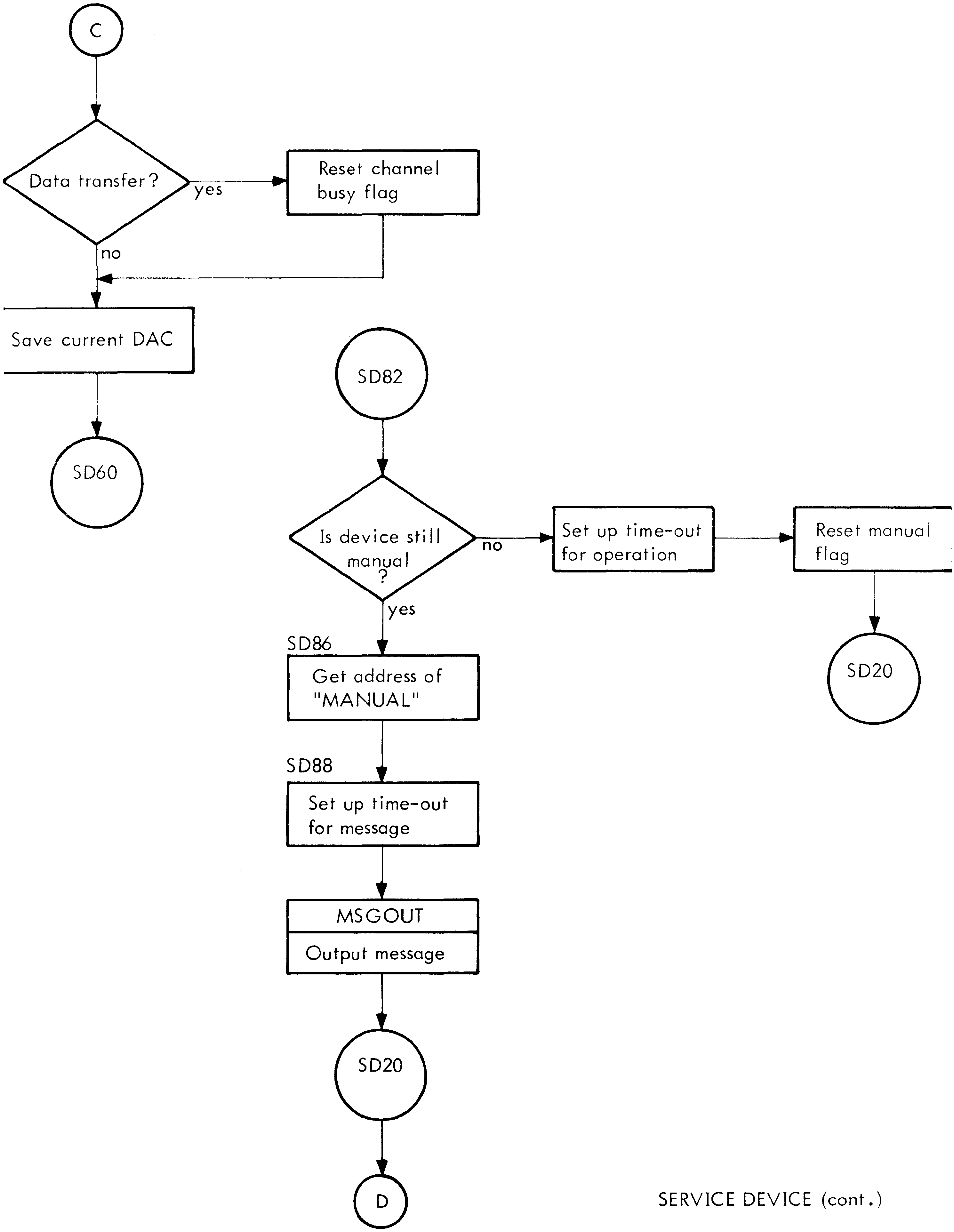

SERDEV
SD10
Disable
Device busy or keyin pending?
yes
SD80
no
Save Current DAC
Cleanup Pending?
yes
SD60
Get request index from DCT6
Can cleanup be done (CTEST)?
no
SD20
yes
SD50
no
SD20
Enable Disable
Set request index to top of queue
Is channel queue empty ?
yes
no
Is channel busy?
yes
Enable
Return
no
A



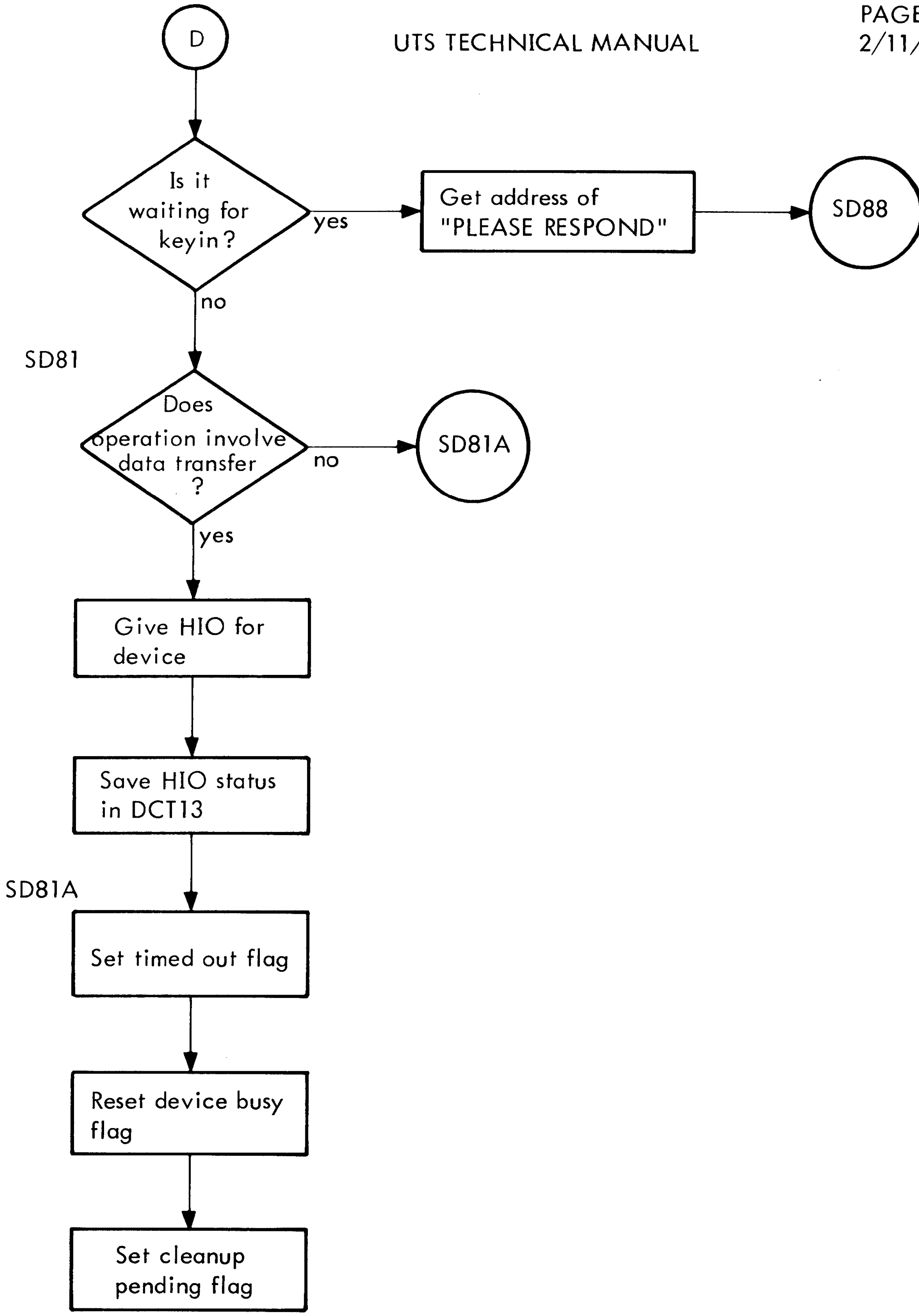
SERVICE DEVICE (cont.)



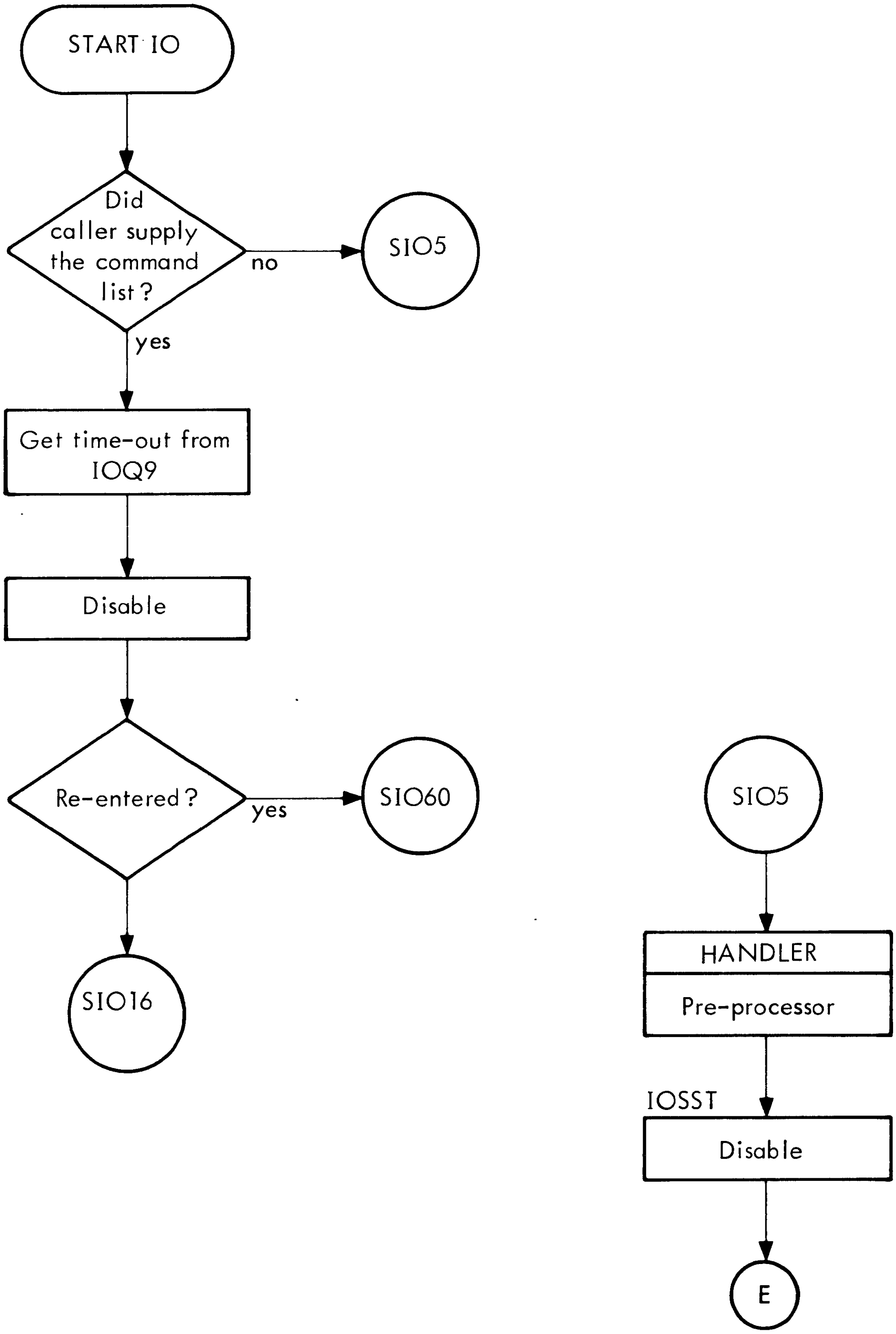
SERVICE DEVICE (cont.)



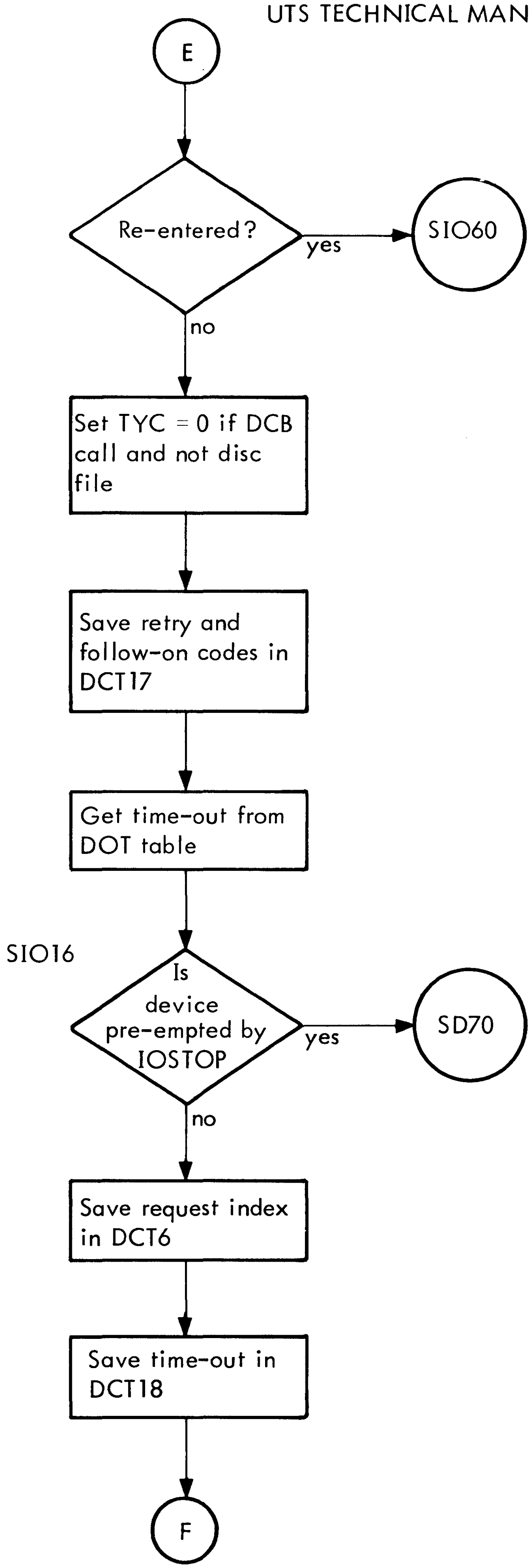
SERVICE DEVICE (cont.)



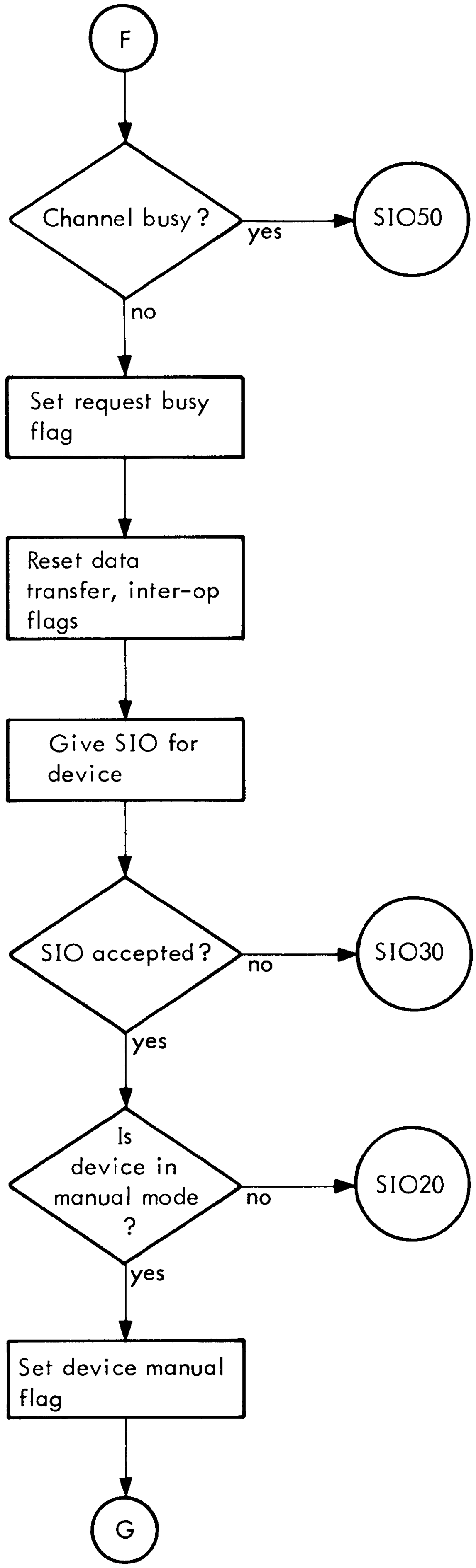
SERVICE DEVICE (cont.)



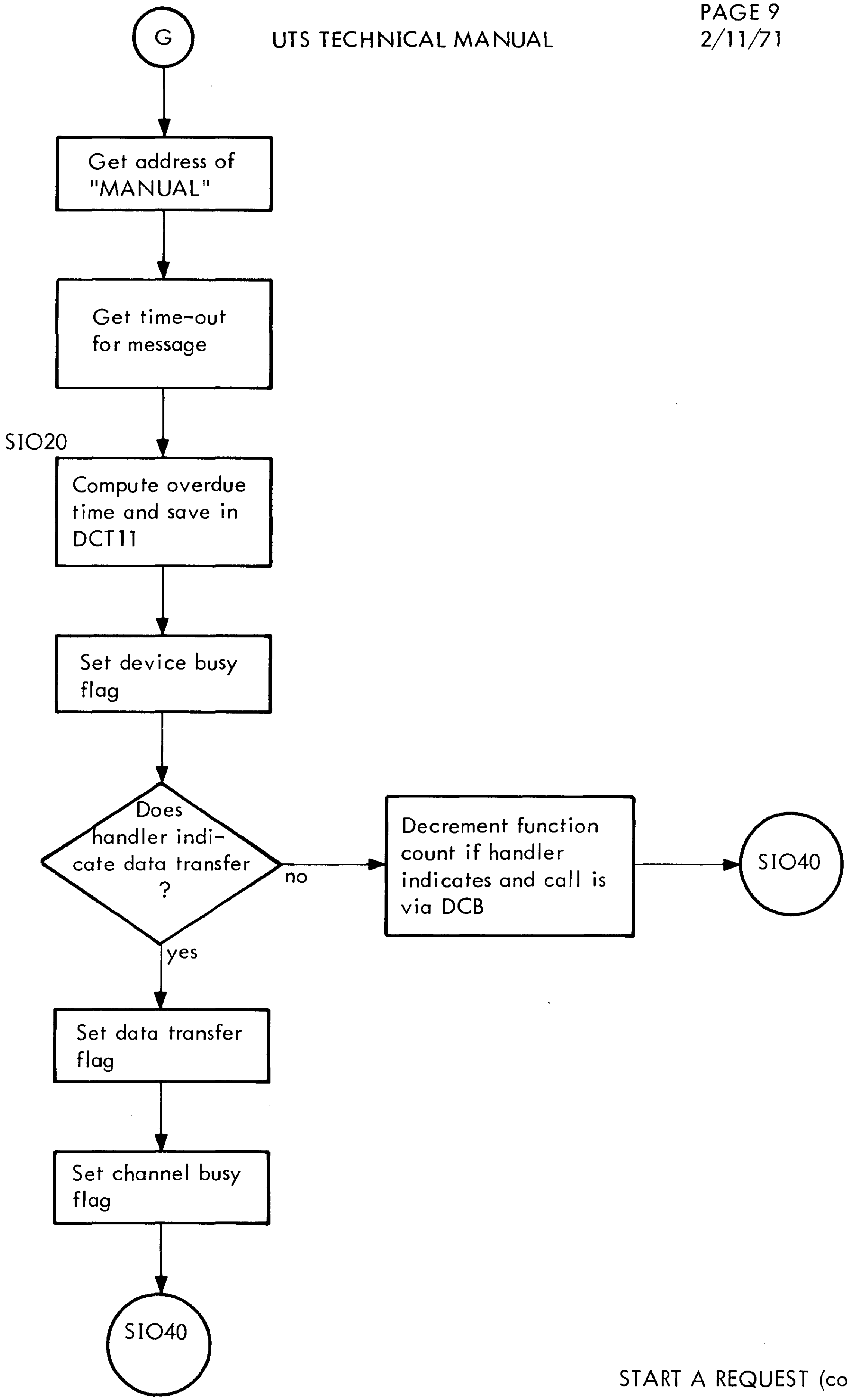
START A REQUEST



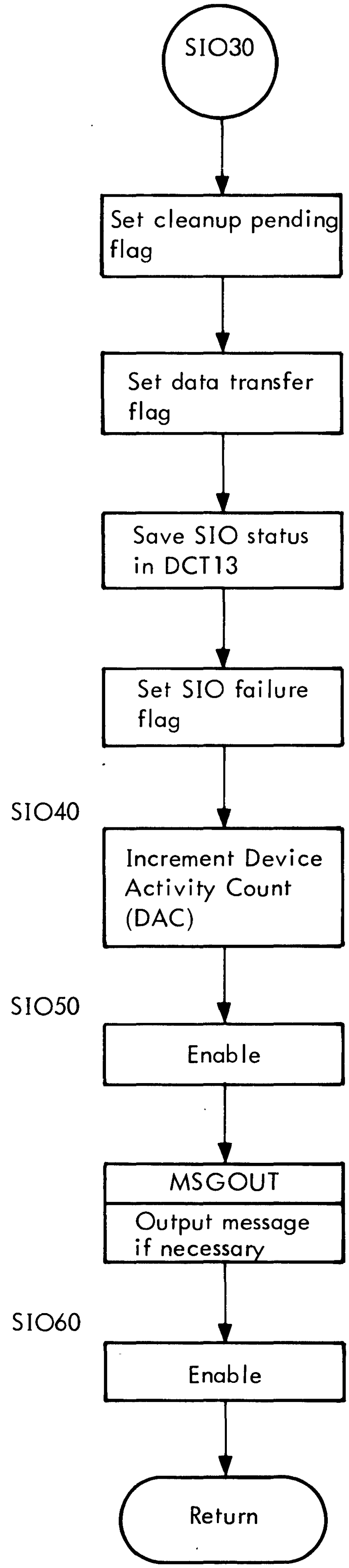
START A REQUEST (cont.)



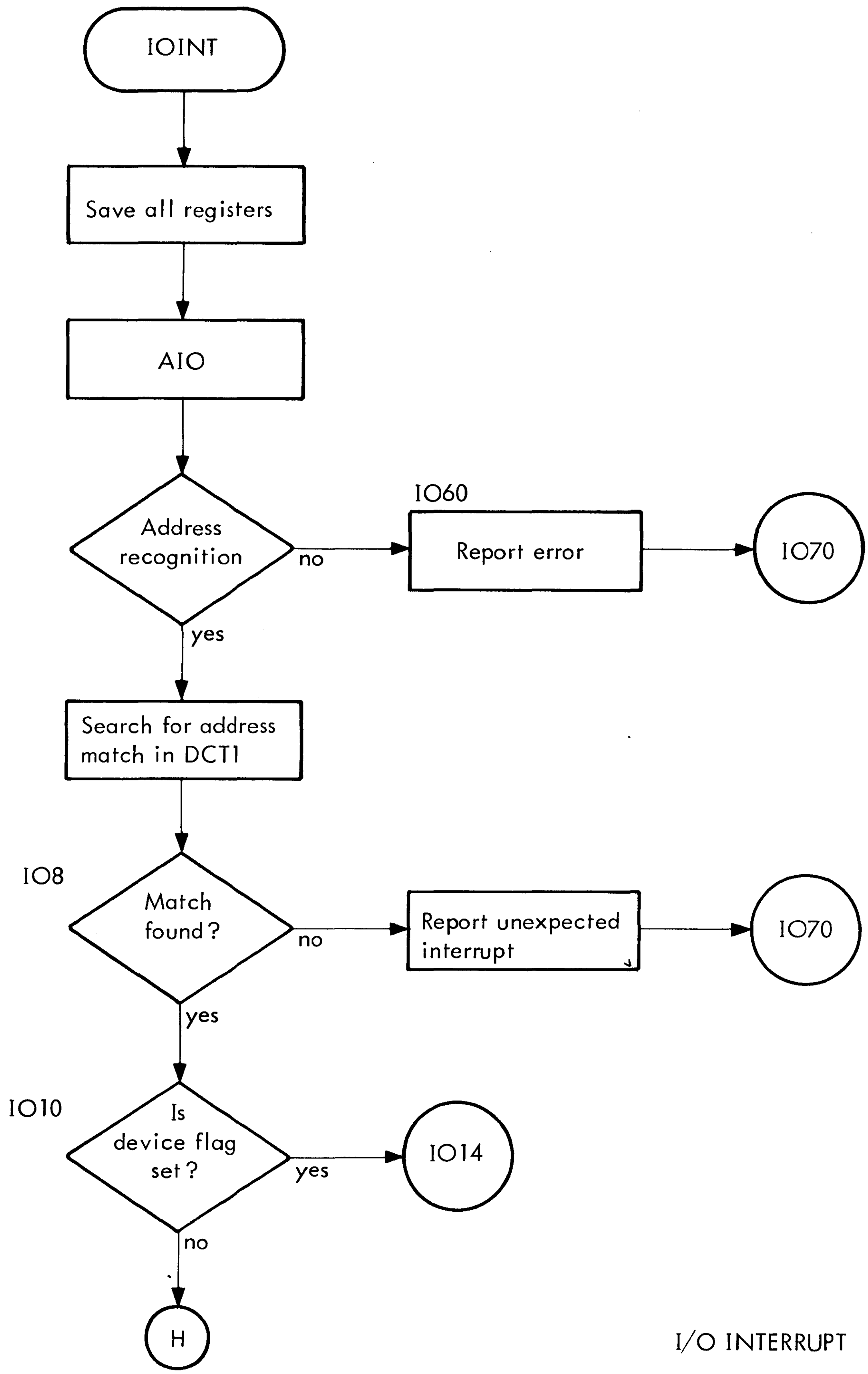
START A REQUEST (cont.)



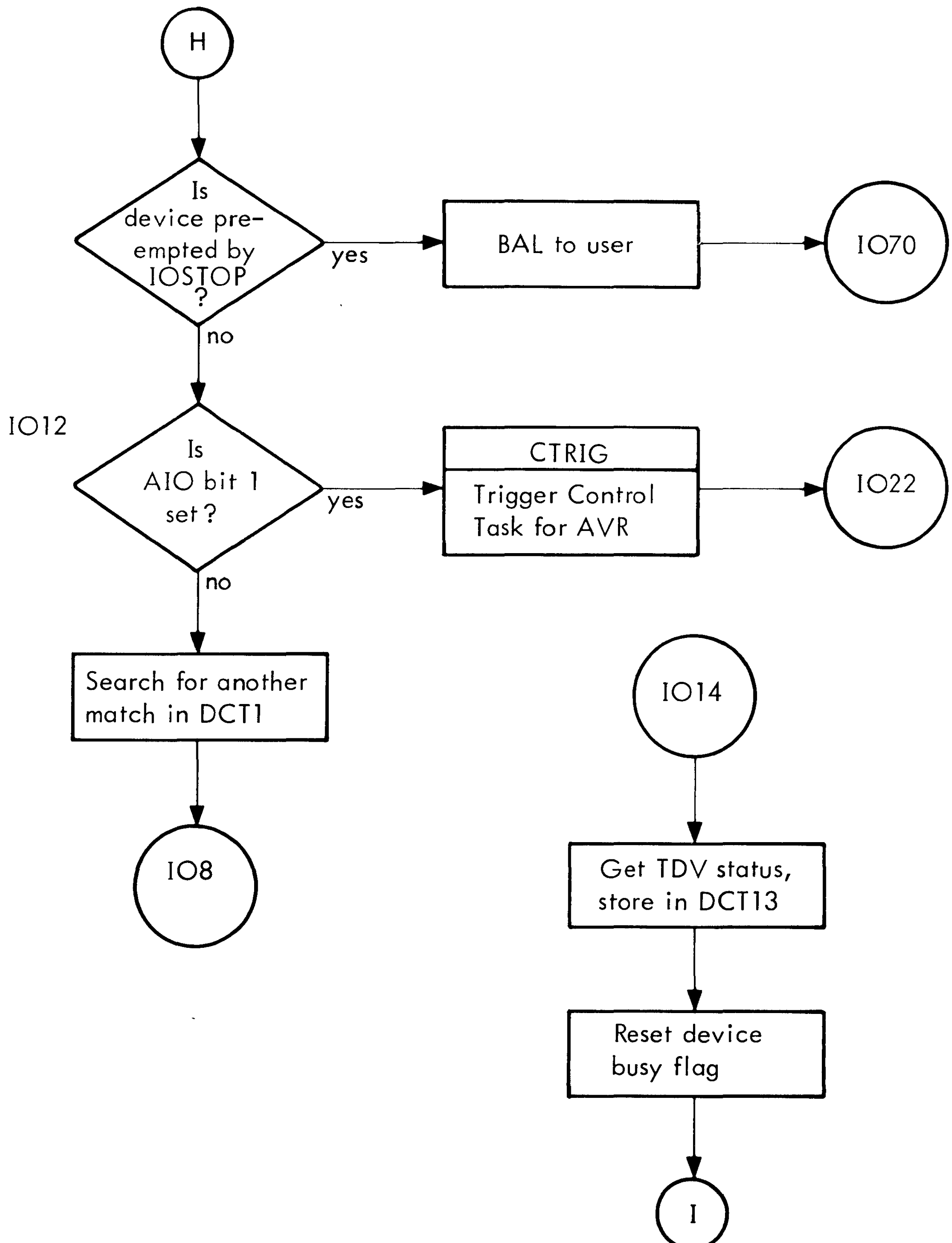
START A REQUEST (cont.)



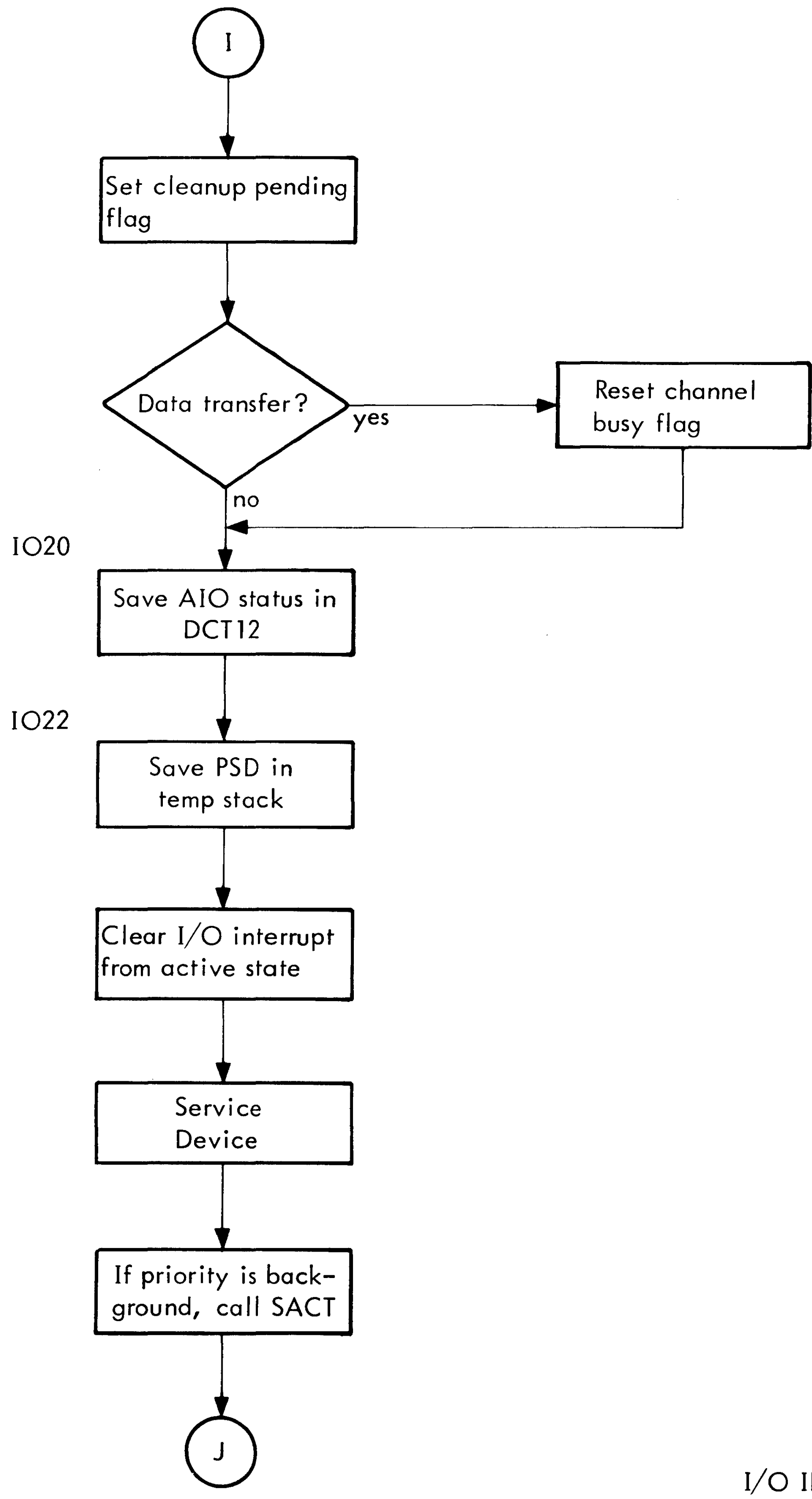
START A REQUEST (cont.)



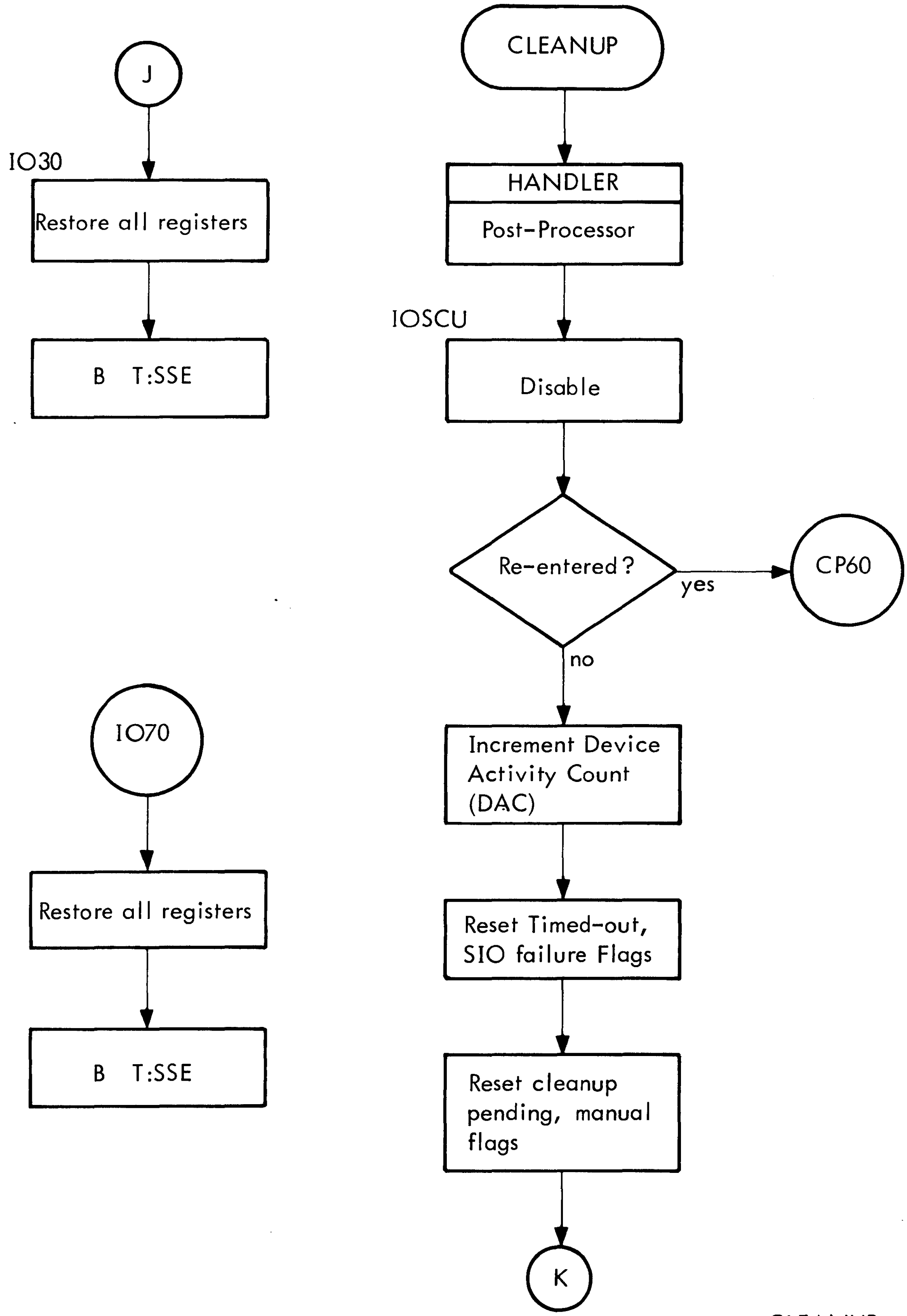
I/O INTERRUPT



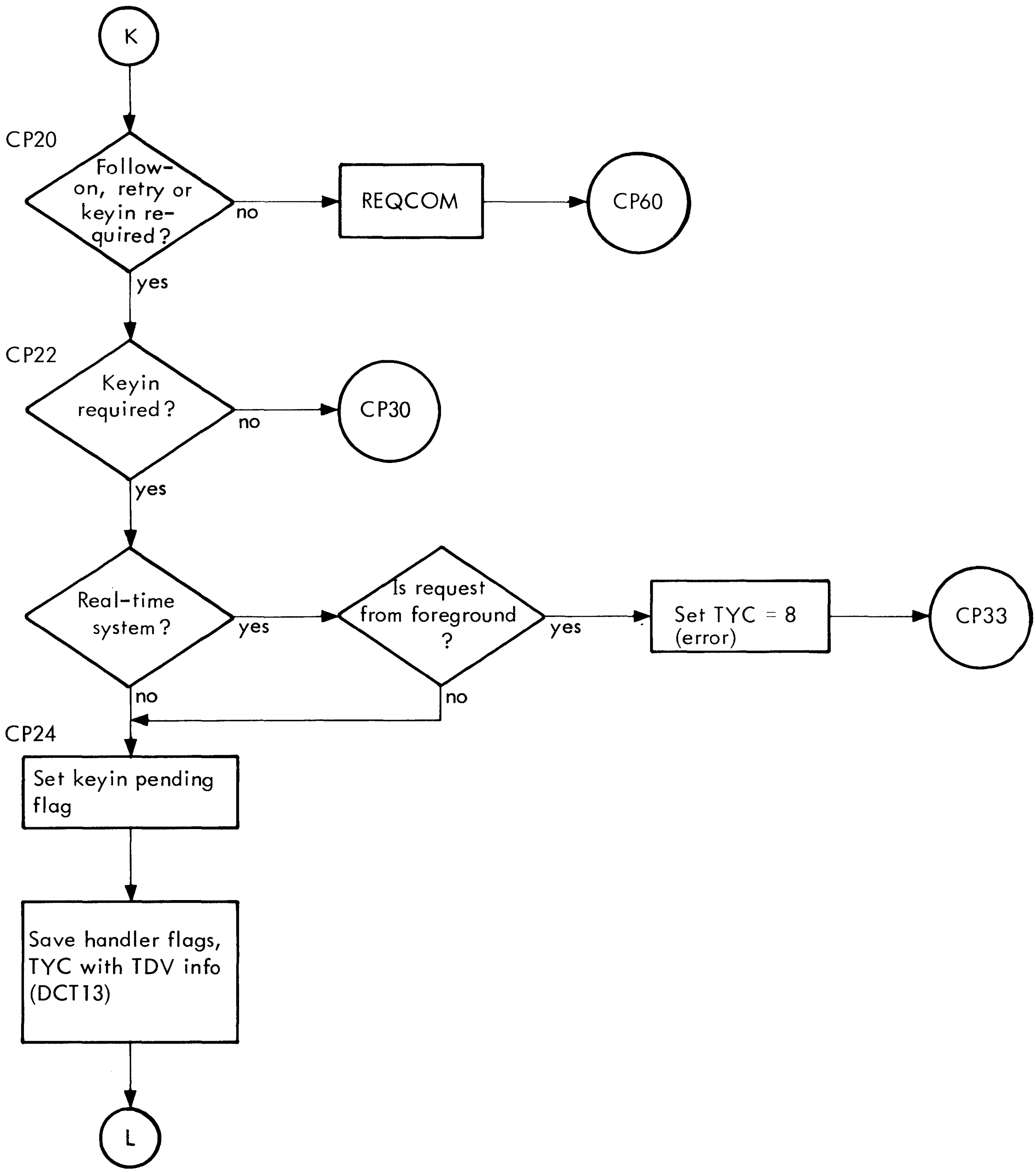
I/O INTERRUPT (cont.)



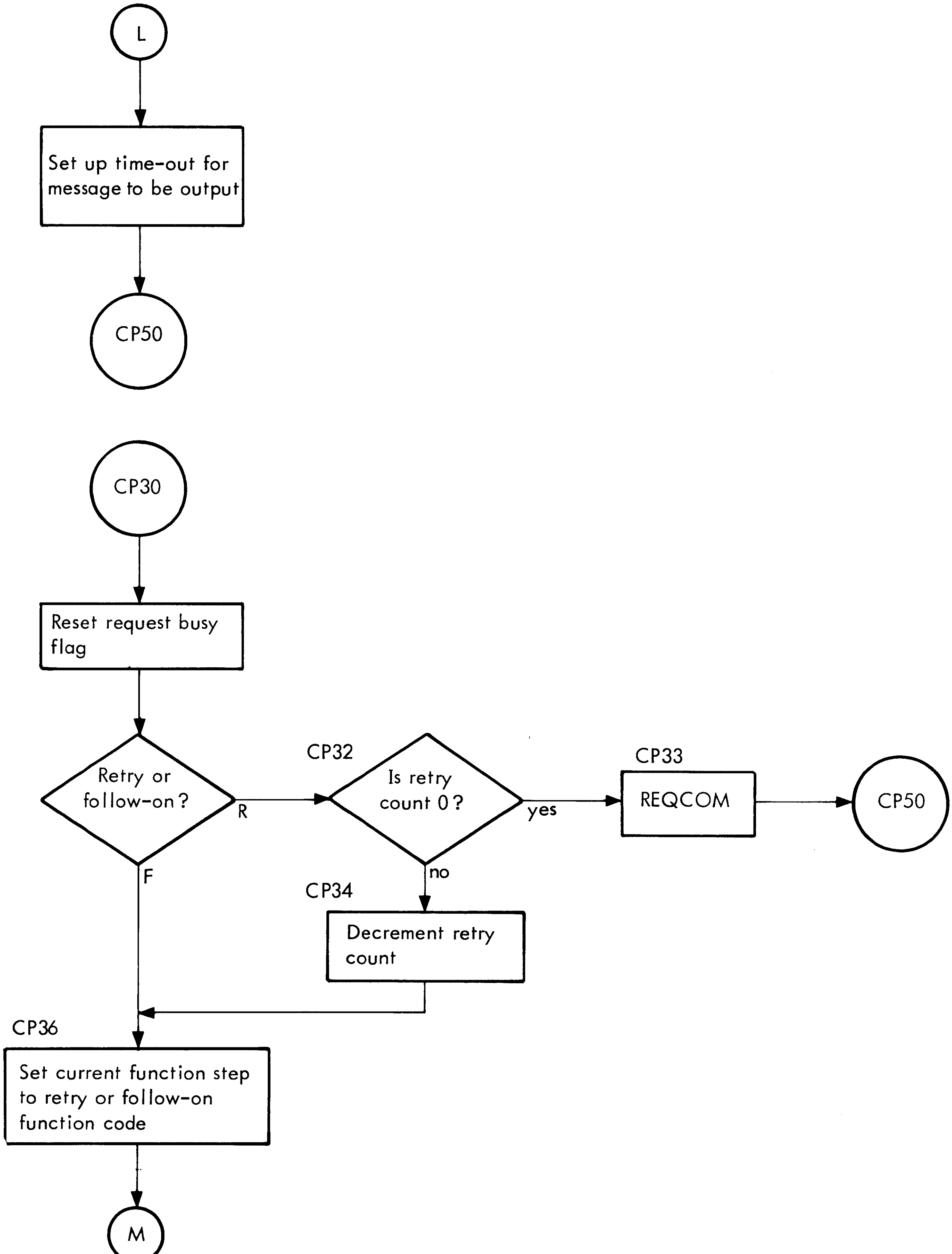
I/O INTERRUPT (cont.)


J
IO30
Restore all registers
B T:SSE
IO70
Restore all registers
B T:SSE
CLEANUP
HANDLER
Post-Processor
IOSCU
Disable
Re-entered?
yes
CP60
no
Increment Device Activity Count (DAC)
Reset Timed-out, SIO failure Flags
Reset cleanup pending, manual flags
K

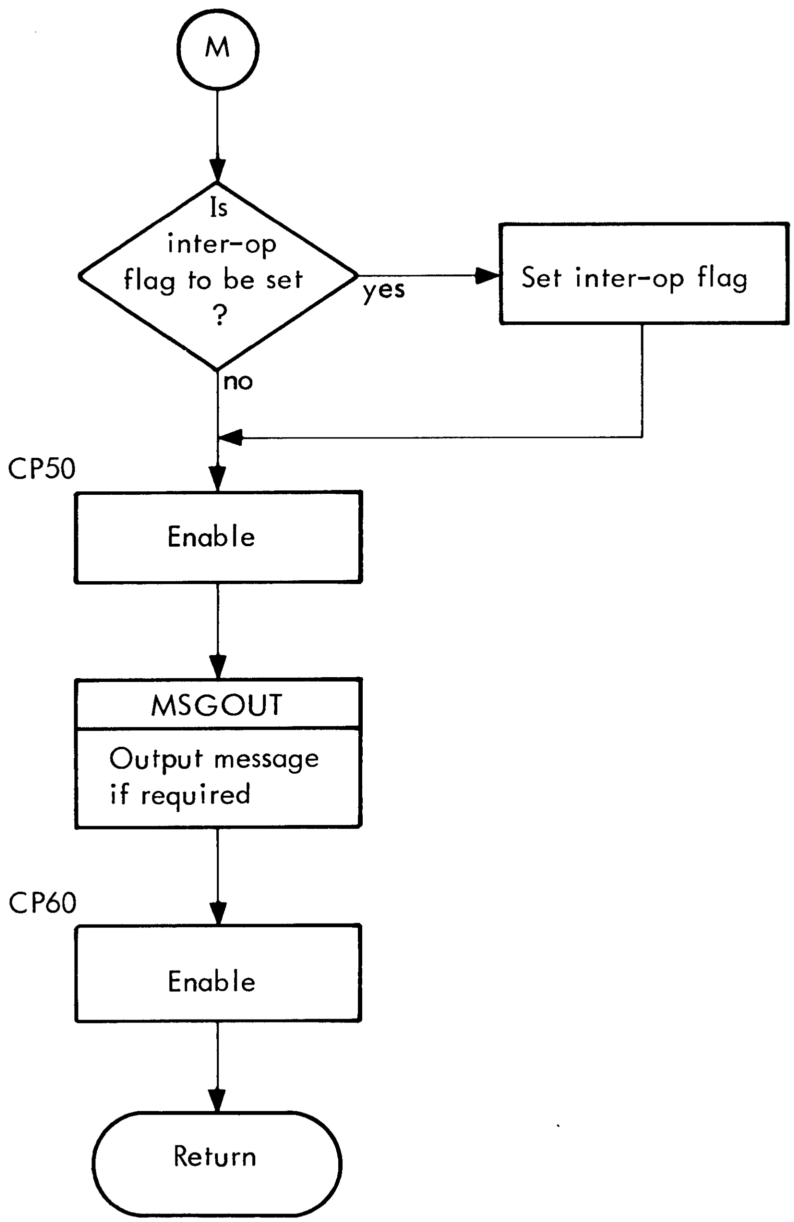
CLEANUP



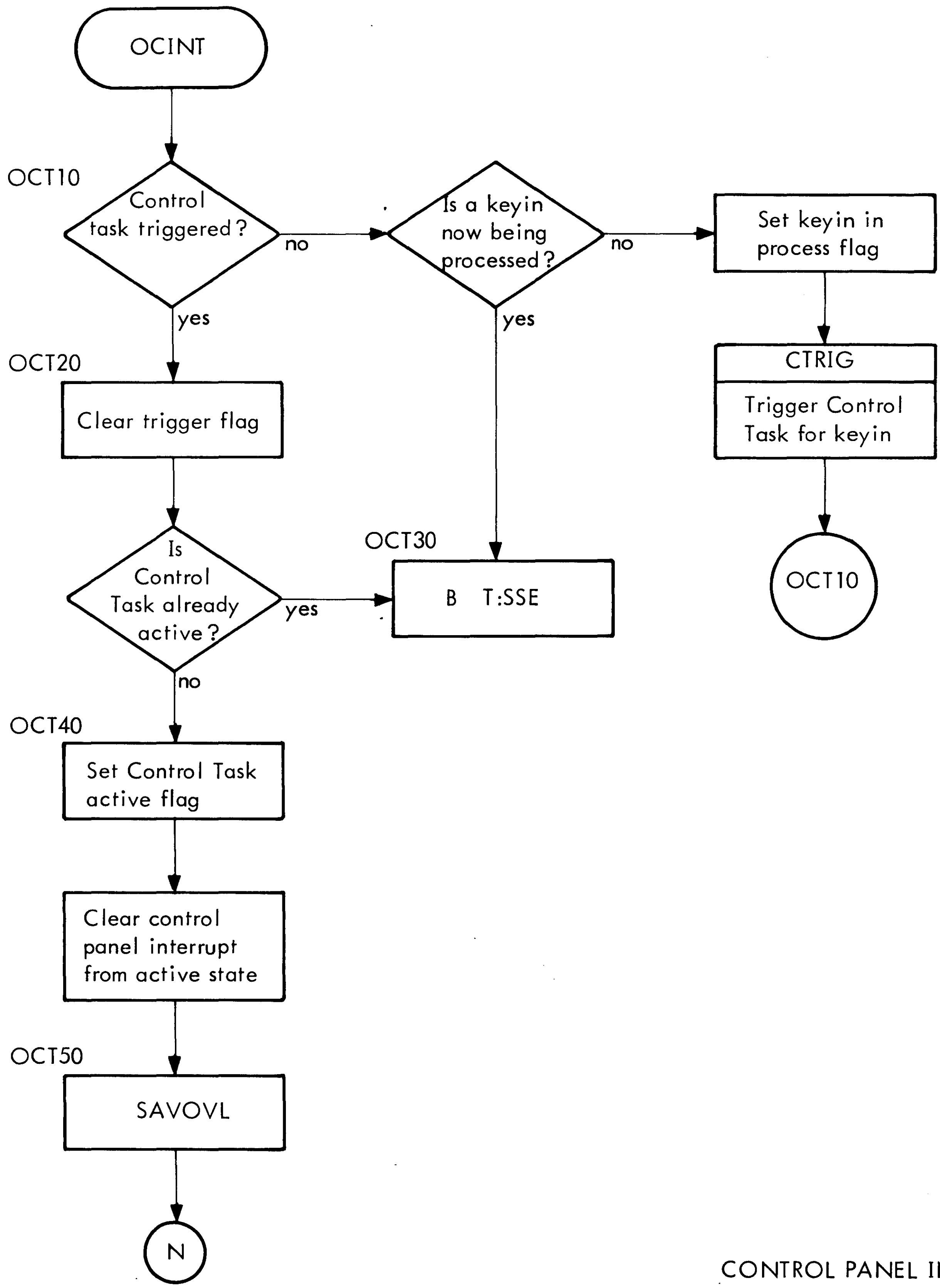
CLEANUP (cont.)



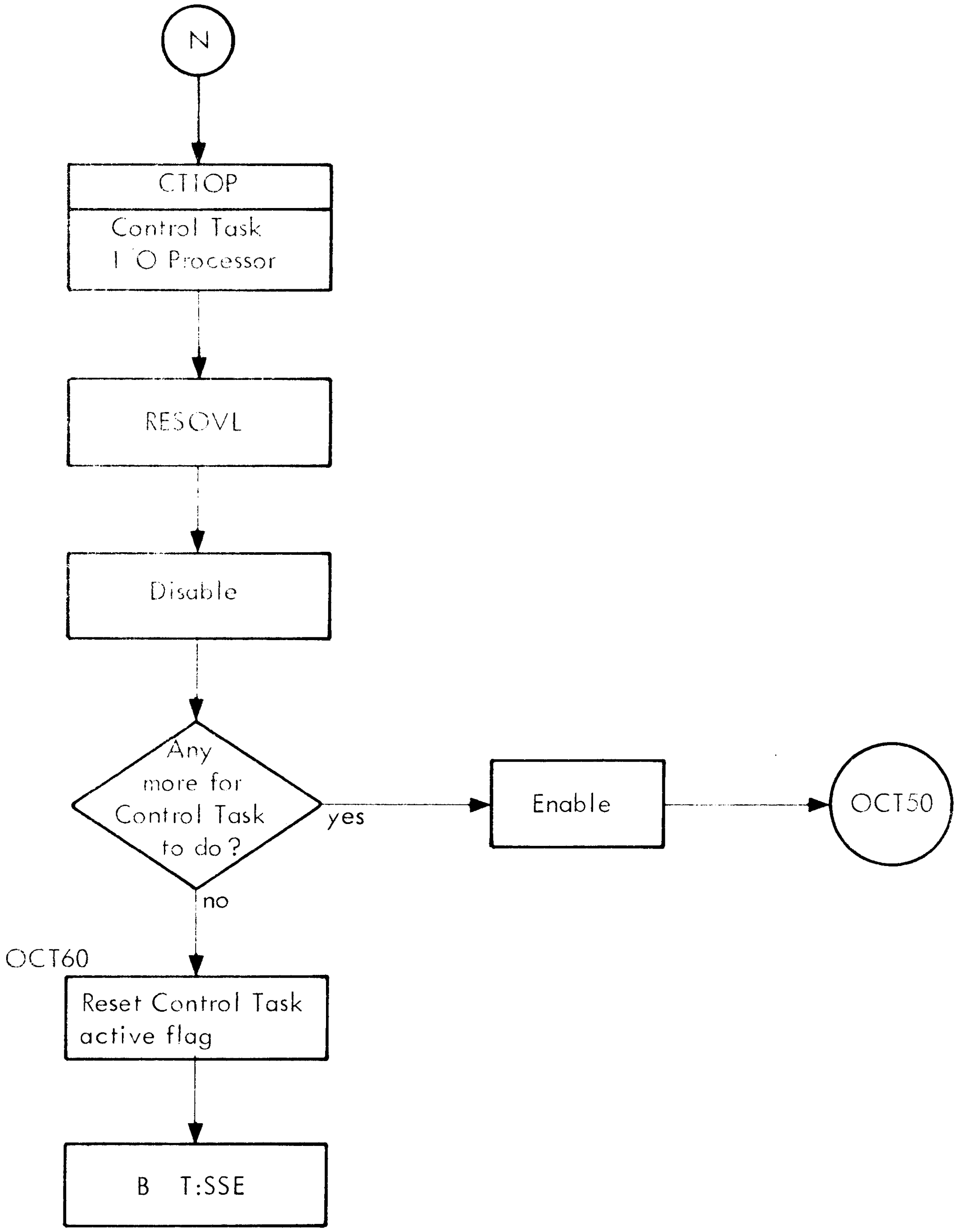
CLEANUP (Cont)



CLEANUP (cont.)



CONTROL PANEL INTERRUPT



CONTROL PANEL INTERRUPT (cont

ID

Swapping RAD I/O - T:SIO

PURPOSE

When the swapper has set up a command chain, for which swapping RAD I/O must be performed, it calls upon T:SIO. TSIO calls upon the I/O system (IOQ) to do the actual I/O and interrupt processing. The I/O system returns to TSIO for end action.

OVERVIEW

TSIO performs error checks on the CL chain, sets up information in registers and calls upon NEWQ to queue up the request. When the interrupt occurs and processing is complete, the I/O system transfers control to the end action rotuine in TSIO. If an error occurred, the I/O system entered a record in the error log file, output a message to the operator's console and passed information about the error to the end action routine. The end action routine will retry the call N times, and if that fails it will set a user flag indicating the error and continue. If the I/O was successful, TSIO returns to the SWAPPER still on end action. However, if the function performed was a write, the I/O system is called upon to do a check write. If the function was reading a user, then TSIO performs a software read check before returning to the SWAPPER.

USAGE

```
T:SIO   BAL,11    T:SIO
```

R6 = Address of beginning of command list.
R5 = Address of end of command list.
R7 = Function code; 2 for read and 1 for write

ERRORS

The screech codes reported by T:SIO are as follows:

0A — Read or write orders in command list are not consistently one or the other but a mixture or in analyzing N read errors order is invalid.

0B — Didn't find seek or sense order in command list when one or the other was expected.

OC Physical page number from byte address in IOCD with read or write order is not between values contained in LOW and HIGH.

OD Termination of command list doesn't agree with command list ending address input T:SIO or termination IOCD doesn't have flags of X'1E'.

OE No I/O is needed as indicated by input beginning address of command list address being equal to input ending address.

OF The function input parameter is not read or write.

93 N write errors occurred and the offending command list can't be found

94 Discovered invalid order trying to continue write checking the rest of command list after N errors occurred.

95 N read errors occurred and there is an invalid address pointing to the offending command list.

96 N errors occurred trying to read a processor.

If a hardware error occurs, IOQ types a message, logs the error and returns to TSIO. After N errors occur, one of three flags is set in a user flag table (UH:FLG2) and TSIO continues, i.e., returns to the SWAPPER. Prior to execution of the user if one of these three flags is set, the error is logged and appropriate action taken. If the flag (bit 13) indicates that a write or write check failed on any page of the user or a read or read check failed and it wasn't in the user's context area (JIT, DCBs, etc.) then the message "SYSTEM SWAPPING ERROR" is output to the user and execution continues as usual. If however the error was in reading or read checking the user's context (bit 14) or user's JIT (bit 15), then the user is deleted.

INTERACTION

T:SSE Control is returned to the system following an interrupt.

RECOVER Is called as a result of failing consistency checks and unrecoverable I/O errors.

T:SEXIT Control is returned to the system to wait for I/O completion.

DOWTCK Is a software switch, normally set, requesting write checking.

DORDCK Is a software switch, normally set, requesting read checking.

SUBROUTINES

SET$REG sets the arguments into registers that are required for the call to NEWQ. Input to SET$REG is the doubleword command list address in register 0 and the DCT index in register 14.

DESCRIPTION

If software checking is required as indicated by sense switch 4 being set, T:SIO ripples through the complete chain of command lists checking for errors. Each command list entry, consisting of 4 words i.e. 2 IOCDs, must have an IOCD with a seek order followed by an IOCD with a read or write order. In one command list there must be only reads or writes but not both. Each 4 word entry must have termination flags of X'4C' in the second IOCD, or be followed by another 4 word entry with a seek order in the 1st IOCD, or be followed by a transfer In Channel IOCD. Each TIC IOCD must be followed by an IOCD containing a seek or a sense order. The command list must be terminated by an IOCD with a sense order or with X'4C' flags and this termination point must agree with the address of the end of the command list specified as input to T:SIO. All physical page numbers contained in the byte addresses of IOCDs with read or write orders must be within the range of physical pages, not containing the monitor, used by the system, as defined (the range) by the values contained in locations LOW and HIGH. If any errors are found, T:SIO transfers to RECOVER with a screech code indicating the error.

If there are no errors, the number of retries is initialized and SET$REG is called to set up the arguments in registers for NEWQ. NEWQ is called upon to queue up the request. When it returns, T:SEXIT is executed.

T:SEXIT pulls a return address from the stack and transfers control to that location. When the swap scheduler was entered, the address of the caller was pushed into the stack. The first time T:SEXIT is called, it will return to that caller. When the I/O system has finished processing a swapper interrupt, it transfers control to end action in TSIO. This end action routine pushes into the stack the return location of the I/O system. End action transfers to the swapper, the swapper calls TSIO again and finally T:SEXIT gets executed again, which finally pulls and returns to the I/O system which returns to the point of interrupt. (See diagram DB-1)

When the I/O system finishes the I/O and processes the interrupt, it transfers to T:SIOEA, the end action routine in TSIO, with information about any errors. T:SIOEA pushes the return address into the stack.

If the I/O system detected any errors, TSIO retries (by calling NEWQ) N times. If these retries are all unsuccessful, a user flag is set as indicated in the error section and TSIO returns to the swapper. If the function was a write check, retry consists of re-writting and then retrying the write check. If software read checking fails, retry consists of rereading.

All successful writes are write checked if DOWTCK is set. No matter how many CLs are in chain, it is executed at one time if the function is read or write. Write checking requires the chain to be partitioned and I/O initiated separately for each part. The AJIT and JIT are write checked first. When this is completed, the JIT can be altered by setting write check orders in the user's CL. If there is another user's JIT CL following, it can be done at the same time. So the routine ripples through the chain, changing write orders to write checks, until it finds a TIC from a JIT CL to a user CL, at which point it resets the chaining flag and sets the interrupt flag. After this I/O is completed, it continues where it left off until it finds the next user CL, and so on, until everything written has been checked. An unsuccessful write check results in only that section just checked, being rewritten and the rechecked.

When the function was reading a user (not processors, JIT or initial data and DCBs), a software read check is performed if DORDCK is set. Comparison is made to insure that halfword identifiers in the user's page start with the value saved in JIT and are consecutive. The halfword destroyed by an identifier is saved in the command chain for each page before it is swapped out and restored during this read check.

When all requested I/O has been completed, TSIO returns to the swapper.

Diagram DB-1: Relationship of SWAPPER, TSIO and IOQ. Illustration shows swapping out 1 user with one error and swapping in a user (JIT in core) without error.

(PUSH 11 Branch to)

SWAPPER

set up
out swap

TSIO

set up for IOQ

IOQ
start I/O

Pull return (routine T:SEXIT in TSIO)
(return to caller of swap)

INTERRUPT ①

end action

process interrupt

push return
suppose error:
set up for IOQ (retry)

pull return (return to
interrupted at ①)

INTERRUPT ②

end action

process interrupt

push return
suppose I/O ok:
return to swapper

set up in swap

set up for IOQ

start I/O

pull return (return to
interrupt at ②) INTERRUPT ③

process interrupt

end action

push return suppose
I/O OK

finish swap, pull return
(return to interrupt and at ③)

ID

COC - Terminal I/O

INTRODUCTION

The UTS COC routines provide I/O operation between typewriter-like user terminals and user programs issuing requests for read and write operations. Connections or communications exist between the COC routines and 1) user or processor programs through CAL instructions which are requests for read, write, format control, and other actions (in this capacity the COC routines are treated as the I/O "handler" for the 7611 communications hardware); 2) the external interrupts from the 7611 which report receipt of input characters and completion of transmission of output characters; and 3) the UTS scheduler to which the significant events of the terminal I/O are reported and to which control is given up for user scheduling when the crucial events occur.

Operations of these routines from the point of view of the user at the terminal and from the view of the user program are described in the UTS System Management Reference Manual, Chapter 6 and the UTS Reference Manual, Chapter 8.

The major functions provided by these routines are:

1) Terminal I/O, Read and Write operations
2) Demultiplexing input characters
3) Buffering of input and output messages into 14-character linked blocks
4) Translation between internal EBCDIC characters and the external code appropriate to the terminal
5) Generation and checking of parity for each character for those terminals requiring it.
6) Recognition of end-of-message characters
7) Echoing CR for LF and LF for CR
8) Reporting of significant I/O events to the scheduler
9) Line-delete and character-delete editing commands
10) Echoplexing for nonlocal printing terminals
11) Sending of user "prompt" characters for each read CAL
12) Tab simulation
13) Splitting long lines to fit on the platen
14) Vertical format control on first output message character
15) Formatting and issuing of page headings

ORGANIZATION

The COC routines may be divided into three groups:

1) The read and write routines which service explicit user CAL instructions to ship messages to and from user terminals. These routines operate in the user map; the scheduler guarantees that the entire user's program is in core during their brief execution. These routines include the control program COC, the read routine COCRD, and the write routine COCWR. A flow chart of the read and write routines is given in Section DC.02.

2) The input and output interrupt routines which service external interrupts from the COC hardware. These routines operate unmapped; the user's program is not required to be in core, and the output routine makes use of an extra register block for faster operation. For proper operation the input external interrupt must be of higher hardware priority than the output interrupt. The input interrupt routine is COCIP and the output interrupt routine is COCOP. Flow charts for these routines are given in Section DC.02.

3) 'Hybrid' routines such as COCMU, COCSENDI, and COCECHO which operate mapped or unmapped as they are called from both read and write routines and interrupt routines.

DATA BASES

The Line Tables:

The COC routines maintain information about each line in a series of tables which are indexed by the COC line number. This control information amounts to 23 bytes per line and contains:

a. MODE, MODE2, MODE3, and COCTERM are bytes which record the operating mode of the line (echoplex, tab simulate, space insertion, paper tape, parity checking, break set characters, etc.), and the type of terminal connected.

b. Bytes containing counts of characters remaining for output, COCOC, and of the maximum number of characters allowed in an input message, RSZ, and of the current size of an input message, ARSZ.

c. In order to simulate physical tab stops the current carriage position is maintained at all times in CPOS and the position at the start of a read in CPI. A halfword, TL, contains the relative address of a buffer containing the tab stop positions to be used during user typing.

d. COCOI, COCOR, COCII, and COCIR record the current insertion and removal points of input and output buffers for the line.

e. BUFCNT records the number of buffers currently occupied by the line.

f. Counts are also kept in JIT of the number of lines on the current page and of the current page number.

COC LINE TABLES

Label	Size (Bytes)	
LB:UN	1	User number associated with line.
COCTERM	1	Terminal Type (implies translation, etc.)
MODE	1	Various
MODE2	1	Line
MODE3	1	Descriptors (see below)
COCOI	2	Byte pointer to current insertion point into output stream for the line. Used by write routine.
COCOR	2	Byte pointer to current removal point from output stream for the line. 0→no buffer. Used by output interrupt.
COCOC	1	Current number of characters pending output including current character being output. 0 → inactive. 1→last character being output and thus COCOI and COCOR are meaningless.
COCII	2	Byte pointer to current insertion point into input stream for the line. 0→no buffer. Used by input interrupt routine.
COCIR	2	Byte pointer to current removal point from input stream for the line. Used by user read routine
RSZ	1	Size of record requested by user if a read is pending
ARSZ	1	Current size of record being read (and echoed) while read pending.

Label	Size (Bytes)	
CPOS	1	Carriage position. Indicates the current column number at which the terminal is (logically) positioned.
CPI	1	Initial carriage position for a read.
BUFCNT	1	Current number of buffers in use by the line. Used for inforcement of maximum number of buffers allocated to a line.
TL	2	Pointer to tab buffer in use by the line while a read is pending. A value of 0 indicates no tabs in effect for the read. Byte 1 of tab buffer is reserved for BS edit.
EOMTIME	2	Contains zero if user read is on going while input has been read ahead. Contains the time remaining before the user will be timed out while a read is pending. Contains the time that the current read request was satisfied.
	23 bytes/line	

Obtaining Terminal Line Table Information

A CAL will be available that provides a requesting program a snapshot of the line table information associated with the user's terminal.

The format of the CAL is:

    CAL1,8   FPT

where FPT contains X'06400000'

The following information will be returned in registers 8 and 9.

Register 8	Register 9
byte 0 = COCTERM	byte 0 , CPOS
byte 1 = MODE	byte 1 = COCOC
byte 2 = MODE2	byte 2 = BUFCNT
byte 3 = MODE3	byte 3 = LB:UN

UTS TECHNICAL MANUAL

VALUES IN COC LINE TABLES

MODE: Bit meanings are:

80	Echoplex (full duplex) Mode (esc E)	1 → Echoplex
40	TTY - Escape Sequence Pending, 2741 -	1 → ESC received
	2741 - EOA Pending	1 → Pending
20	Transparent Mode (DRC)	1 → Transparent
10	Reading Pending (0 → Read Ahead)	1 → Read pending
08	Tab Simulation Active (esc T)	1 → Tab simulate
04	Restrict Code to Upper Case (esc U)	1 → Restricted
02, 01	Break Count	

MODE2: Bit meanings are:

80	Line Reported Off	1 → Line reported off
40	Full Duplex Paper Tape Mode (X ON)	1 → X ON
20	Space Insertion (esc S)	1 → Space insert
10	2741 Line	1 → 2741
08	Shift to Lower Case (esc (esc))	1 → Shifted to lower case
04	Check Parity Mode	1 → Check Parity
02, 01	Break Set	

MODE3: Bit meanings are:

80	Tab Relative to Beginning of Input (esc C)	1 → Rel Tabbing
40	Half Duplex Paper Tape Mode (esc P)	1 → Half Duplex Paper Tape
20	Backspace Edit Flag (2741)	1 → B. S. Edit mode
10	2741 Keyboard Locked	1 → Locked
08	Lost Input (insufficient buffers)	1 → Input Lost
04, 02, 01	Number of lines upspaced during input	

Defaults are:

TTY - Echoplex, Tab Simulate, Space Insert, all else off
2741 - EOA pending, 2741 line, check parity, keyboard locked all else off.

UTS TECHNICAL MANUAL

COCTERM

0	Model 33 TTY
1	Model 35 TTY
2	Model 37 TTY
3	XOS Model 7015
4-5	EBCD Standard 2741
6-7	EBCD APL 2741
8-9	Selectric Standard 2741
10-11	Selectric APL 2741

Buffers:

Input from and output to user terminals is buffered in resident core using linked chains of four-word buffers containing 14 characters and a relative link in the first half-word. For output, the message is translated to external form and placed in as many buffers as are required. The output interrupt routine sends the characters one at a time from the buffers, releasing those that are empty. On input, a buffer is not assigned until the first input character is received.

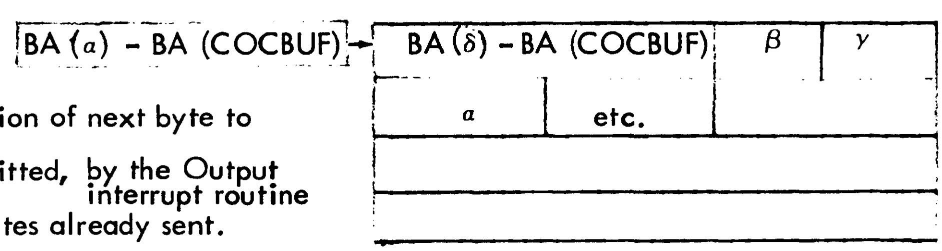
Buffers as shown below are four words long and chained together by relative pointers to the buffer pool carried in the first halfword of each buffer. A zero link terminates the the chain. Fourteen characters are placed in the remaining space in each buffer.

A chain of free buffers are retained and pointed to by COCHPB. The free buffers are chained through the first word of each buffer. Zero signifies the last buffer in the free chain.

Lines with input and/or output characters in the line tables have buffers linked as follows:

Output:

COCOR



α ≈ Location of next byte to be transmitted, by the Output interrupt routine

β, γ = Bytes already sent.

UTS TECHNICAL MANUAL

Last buffer

COCOI $\boxed{\text{BA }(\epsilon)\text{ - BA(COCBUF)}}$ →

0		δ	etc.
			ε

ε is position of last character placed in buffer by COCSEND1

Input:

COCIR $\boxed{\text{BA }(\alpha)\text{ - BA (COCBUF)}}$ →

DA (δ) - BA (COCBUF)		β	γ
α	etc.		

α = Location of next byte to be moved to user as a result of read.

B, Y = Bytes that have already been moved.

Last Buffer

COCII $\boxed{\text{BA }(\epsilon)\text{ - BA (COCRUF)}}$ →

0		δ	etc.
			ε

ε is position of last character received and placed in buffer by Input Interrupt, routine.

Error Counts:

COCIPC	Count of characters received with parity error.
COCIPL	Line # for last parity error or untranslatable character received.
COCOEC = COCBLC	Count of input and output interrupts from lines not valid (out of COC table range).
COCOEL = COCBLN	Line # for last invalid interrupt.

Executive message:

COCMESS	Administrative message buffer for page heading. (16 words)

Translate Tables:

Associated with each type of terminal is a pair of translation tables which give the correspondences between internal and external character codes. Special translation codes trigger input functions such as character and line delete, tab simulation, echoes for carriage return and line feed, and other special operations.

A single pair of translate tables which handles all ASCII coded terminals (types 0-3) is provided in the standard UTS system. Additional translate tables are provided via SYSGEN option for 2741.

The general format of translation tables is as follows:

a. Input (table indexed by device code yields EBCDIC code)

   1) TTY - 128 bytes in length (parity stripped before translation)

   2) 2741 (each code set) - 2 tables each 64 bytes in length (parity stripped before translation); first table for lower case, second table for upper case.

b. Output (table indexed by EBCDIC code yields device code less parity). All are 256 bytes in length.

Device Code is formatted as follows:

0	device code-parity	
1	Type	Special Code
0	1 2	3 4 5 6 7

If bit 0 =0, remainder is device code minus parity. If 2741, bit 1= 1 for Upper Case Character, bit 1 = 0 for Lower Case Character

If bit 0 = 1 remainder is further qualified by Type:

Type = 0 → Special Code Significance is:

0 - Form Feed
1 - HT (Tab)
2 - CR and LF must be sent (TTY only)
3 - NL must be sent (2741 only)
4 - ESCF
5 - ESCX
6 - Destructive Rubout (Rubout, BS ATTN)
7 - Retype
8 - Local Carriage Return
9 - "⌉" (7015), "[" (TTY 33-37)
A - "⌐" (7015), "]" (TTY 33-37)
B - 2741 Backspace (2741)
C - Local LF (TTY)
D - LF must be sent
E - Parity Error

UTS TECHNICAL MANUAL

Type = 1 ⟶ Character is Delta Activation Character and bits 2-7 are EBCDIC code to use for true output translation.

Type = 2 ⟶ This is a mode setting operation (e.g., ESC E). Bits 5-7 determine the bit within a flag byte to be affected (0 ⟶ bit 7, 1 ⟶ bit 6, ... 7 ⟶ bit 0).

Type = 3 ⟶ Ordinary Activation Character and bits 2-7 are EBCDIC code to use for true output translation.

The exact action taken on all input and output characters is contained in this section under CONTROL FUNCTIONS.

Sample Translate Tables for TTY and the selectric standard 2741 terminals follows:

```
2960                                *
2961                                *
2962                                *       TTY AND K/D  INPUT TRANSLATE TABLE --ASCII TO EBCDIC
2963                                *
2964                                *
2965          FR                    TTYOUT    EQU      KDOUT
2966          01 00729              TTYIN     EQU      $
2967          01 00729              KDIN      EQU      $
2968                                *                  EBCDIC EQUIVAVENT OF ..... ASCII CHARACTERS
2969                                *     0
2970                                *
2971                                *
2972     01 00729     00010203 A     DATA,8 X'0001020304090607'    NUL,  SOH,  STX,  ETX,  EOT,  ENQ,  ACK,  BEL
                      04090607
2973     01 0072B     0805150B A     DATA,8 X'0805150B0C0D0E0F'     BS,   HT, NL(LF), VT,   FF,   CR,   SO,   SI
                      0C0D0E0F
2974                                *     1
2975     01 0072D     103C123D A     DATA,8 X'103C123D140A1617'   DLE,DC1(XON),DC2,DC3(XOFF),DC4, NAK, SYN,  ETB
                      140A1617
2976     01 0072F     32191A30 A     DATA,8 X'32191A301C1D1E1F'  CAN(CTL-X),EM(CTL-Y),SUB,ESC,FS, GS,  RS ,  US
                      1C1D1E1F
2977                                *     2
2978     01 00731     405A7F7B A     DATA,8 X'405A7F7B5B6C507D'  BLANK,EXCL MK,QUOT MK, #,   $,    %,   &,    '
                      5B6C507D
2979     01 00733     4D5D5C4E A     DATA,8 X'4D5D5C4E6B604B61'     (,    ),    *,    +,    ,,    -,    .,    /
                      6B604B61
2980                                *     3
2981     01 00735     F0F1F2F3 A     DATA,8 X'F0F1F2F3F4F5F6F7'     0,    1,    2,    3,    4,    5,    6,    7
                      F4F5F6F7
2982     01 00737     F8F97A5E A     DATA,8 X'F8F97A5E4C7E6E6F'     8,    9,    :,    ;,    <,    =,    >,QUEST MK
                      4C7E6E6F
```

112

```
 2984                                *    4
 2985   01 00739   7CC1C2C3 A        DATA,8 X'7CC1C2C3C4C5C6C7'     @,   A,   B,   C,   D,   E,   F,   G
                   C4C5C6C7
 2986   01 0073B   C8C9D1D2 A        DATA,8 X'C8C9D1D2D3D4D5D6'     H,   I,   J,   K,   L,   M,   N,   O
                   D3D4D5D6
 2987                                *
 2988                                *    5
 2989   01 0073D   D7D8D9E2 A        DATA,8 X'D7D8D9E2E3E4E5E6'     P,   Q,   R,   S,   T,   U,   V,   W
                   E3E4E5E6
 2990   01 0073F   E7E8E94F A        DATA,8 X'E7E8E94FB15F6A6D'     X,Y,Z,7015'OR',BK/,7015'NOT', ARROW - UP,BACKN
                   B15F6A6D
 2991                                * FOR TTY'S OTHER THAN 7015, ASCII '5B' & '5D'(LEFT & RIGHT BRACKETS)
 2992                                * ARE TRANSLATED RESPECTIVELY INTO - 'B4' & 'B5'
 2993                                *
 2994                                *
 2995                                *    6
 2996   01 00741   4A818283 A        DATA,8 X'4A81828384858687'   CENTS,LC'A',LC'B',LC'C',LC'D',LC'E',LC'F',LC'G'
                   84858687
 2997   01 00743   88899192 A        DATA,8 X'8889919293949596'   LC'H',LC'I',LC'J',LC'K',LC'L',LC'M',LC'N',LC'O'
                   93949596
 2998                                *    7
 2999   01 00745   979899A2 A        DATA,8 X'979899A2A3A4A5A6'   LC'P',LC'Q',LC'R',LC'S',LC'T',LC'U',LC'V',LC'W'
                   A3A4A5A6
 3000   01 00747   A7A8A9B2 A        DATA,8 X'A7A8A9B24FB35FFF'   LC'X',LC'Y',LC'Z',BRACE(, OR,BRACE), NOT, RUB
                   4FB35FFF

 3002   01 00749   311B1B80 A     ALTMODES DATA     X'311B1B80'        ACTIVE RUB, ALT-ESC, ALT-ESC, RUBOUT
```

113

```
                                        *
                                        *       TTY AND K/D  OUTPUT TRANSLATE TABLE -- EBCDIC TO ASCII
                                        *
           01 0074A                    KDOUT    EQU      *                    K/D OUTPUT TRANSLATE TABLE        KD000600
                                        *   00
01 0074A    00010203 A     DATA,8  X'00010203EA810607'   NUL,  SOH,  STX,  ETX, *EOT,  *HT,  ACQ,  BEL
            EA810607
01 0074C    08051508 A     DATA,8 X'0805150B80820E0F'    BS,  ENQ,  NAK,   VT,  *FF,  *CR,   SO,   SI
            80820E0F
                                        *   01
01 0074E    10111213 A     DATA,8  X'1011121314821617'  DLE,  XON,  DC2, XOFF,  DC4,*NL(LF),SYN,  ETB
            14821617
01 00750    18EC8E1B A     DATA,8 X'18EC8E1BE1E2E3E4'   CAN,  *EM, *SUB, *ESC,  *FS,  *GS,  *RS,  *US
            E1E2E3E4
                                        *   02
01 00752    8D1C1D1E A     DATA,8 X'8D1C1D1E1F292F5E'   *LF,   FS,   GS,   RS,   US,    ),   /,UP-ARROWN
            1F292F5E
01 00754    3D0D0408 A     DATA,8  X'3D0D040819090A23'   =,   CR,  EOT,   BS,   EM,   HT,   LF, ALARM
            19090A23
                                        *    3
01 00756    84868506 A     DATA,8  X'848685D6D2C3CBD3'  ESC*F,*RUB,ESC*X,ESC&P,ESC&U,ESC&T,ESC&T,ESC&T
            D2C3CBD3
01 00758    D5D7D78C A     DATA,8 X'D5D7D78CCEC68788'   ESC&S,ESC&E,ESC&C,ESC*LF,*XON,*XOFF,ESC*R,ESC*CRN
            CEC68788
                                        *    4
01 0075A    20232323 A     DATA,8  X'2023232323232323'  BLANK, SUB , SUB , SUB , SUB , SUB , SUB , SUB
            23232323
01 0075C    2323602E A     DATA,8  X'2323602E3C282B89'   SUB , SUB ,CENTS,    .,    <,    (,    +, *OR
            3C282B89
                                        *    5
01 0075E    26232323 A     DATA,8  X'2623232323232323'     &,  SUB , SUB , SUB , SUB , SUB , SUB , SUB
            23232323
01 00760    23232124 A     DATA,8  X'232321242AA53B8A'   SUB , SUB ,EXCL MK, $,     *,    *),    ;, *NOT
            2AA53B8A
                                        *    6
01 00762    2DA62323 A     DATA,8  X'2DA6232323232323'     -,    */,  SUB , SUB , SUB , SUB , SUB , SUB
            23232323
```

114

3

```
G00   13:19  MAR 21, '72                           ASCII TRANSLATION TABLE                              154
  3028   01 00764    2323A72C A      DATA,8  X'2323A72C255F3E3F'   SUB , SUB ,+UP-ARROW, ,,  %,BK-ARROW, >,QUEST MK
                     255F3E3F
  3029                             *     7
  3030   01 00766    23232323 A      DATA,8  X'2323232323232323'   SUB , SUB , SUB , SUB , SUB , SUB , SUB , SUB
                     23232323
  3031   01 00768    23233A23 A      DATA,8  X'23233A234027A822'   SUB , SUB ,   :,    #,    @,    ',   +",!
                     4027A822
  3032                             *     8
  3033   01 0076A    23616263 A      DATA,8  X'2361626364656667'   SUB ,LC'A',LC'B',LC'C',LC'D',LC'E',LC'F',LC'G'
                     64656667
  3034   01 0076C    68692323 A      DATA,8  X'6869232323232323'  LC'H',LC'I', SUB , SUB , SUB , SUB , SUB , SUB
                     23232323
  3035                             *     9
  3036   01 0076E    236A6B6C A      DATA,8  X'236A6B6C6D6E6F70'   SUB ,LC'J',LC'K',LC'L',LC'M',LC'N',LC'O',LC'P'
                     6D6E6F70
  3037   01 00770    71722323 A      DATA,8  X'7172232323232323'  LC'Q',LC'R', SUB , SUB , SUB , SUB , SUB , SUB
                     23232323
  3038                             *     A
  3039   01 00772    23237374 A      DATA,8  X'2323737475767778'   SUB , SUB ,LC'S',LC'T',LC'U',LC'V',LC'W',LC'X'
                     75767778
  3040   01 00774    797A2323 A      DATA,8  X'797A232323232323'  LC'Y',LC'Z', SUB , SUB , SUB , SUB , SUB , SUB
                     23232323
  3041                             *     B
  3042   01 00776    235C7B7D A      DATA,8  X'235C7B7D5B5D2323'   SUB ,BK'/',(BRACE,BRACE),(BRACK,BRACK),SUB,SUB
                     5B5D2323
  3043   01 00778    23232323 A      DATA,8  X'232323232323EF23'    SUB, SUB,  SUB,  SUB,  SUB,SUB,+LOSTDATA,SUB
                     2323EF23
  3044                             *     C
  3045   01 0077A    20414243 A      DATA,8  X'2041424344454647'  SPACE,   A,    B,    C,    D,    E,    F,    G
                     44454647
  3046   01 0077C    48492323 A      DATA,8  X'4849232323232323'     H,    I,  SUB , SUB , SUB , SUB , SUB , SUB
                     23232323
  3047                             *     D
  3048   01 0077E    234A4B4C A      DATA,8  X'234A4B4C4D4E4F50'   SUB ,   J,    K,    L,    M,    N,    O,    P
                     4D4E4F50
  3049   01 00780    51522323 A      DATA,8  X'5152232323232323'     Q,    R,  SUB , SUB , SUB , SUB , SUB , SUB
                     23232323
```

4

```
                                     *    E
01 00782   2D235354 A         DATA,8  X'2D23535455565758'   - , SUB,   S,   T,   U,   V,   W,   X
           55565758
01 00784   595A2323 A         DATA,8  X'595A232323232323'   Y,   Z,  SUB , SUB , SUB , SUB , SUB , SUB
           23232323
                                     *    F
01 00786   30313233 A         DATA,8  X'3031323334353637'   0,   1,   2,   3,   4,   5,   6,   7
           34353637
01 00788   38392323 A         DATA,8  X'383923232323237F'   8,   9,  SUB , SUB , SUB , SUB , SUB , DEL
           2323237F
                                     *
                                     *
                                     *
                                     * THE SYMBOL *, +, AND &, WHICH PRECEED OR ARE IMBEDDED IN COMMENTARY SYMBOLS
                                     * INDICATE CATAGORIES OF CHARACTERS WHICH REQUIRE SPECIAL HANDLING.
                                     *  THE SPECIAL CATAGORIES ARE:
                                     *
                                     *          *    . . .    UNIQUE ACTION IS GENERALLY REQUIRED.
                                     *
                                     *          +    . . .    THE CHARACTER WILL NORMALLY ACTIVATE, OR
                                     *                        IT IS A DELTA ACTIVATION CHARACTER.
                                     *
                                     *          &    . . .    CHANGE APPROPRIATE MODE IN LINE TABLE.
                                     *
                                     *
                                     *
                                     * END OF K/D OUTPUT TRANSLATE TABLE                               KD000950
                                     *                                                                 KD000960
           0000078A                  CBCMAINSIZE EQU   2*(ABSVAL(DA($-1))+1)
01 0078A                                      USECT CBCECHBB
           0000078A                  CBC CODE SIZE EQU CBCMAINSIZE+2*(ABSVAL(DA($-1))+1)
                                              END                                               KD000970
```

CONTROL SECTION SUMMARY: 01 0078A   PT 0   02 00000   PT 0

5

116

```
                                  *         OUTPUT TRANSLATION TABLE  . . .  EBCDIC TO SELECTRIC STANDARD
            01 00020              SSTD      EQU       $
                                  *
                                  *    0
   01 00020     3D3E0E0D A                  DATA      X'3D3E0E0D'            NUL-->IL,SOH->PRE,STX-->BY,ETX->RES
   01 00021     EA813D3D A                  DATA      X'EA813D3D'              'EOT' ,  'HT'  ,ACR-->IL,BEL-->IL
   01 00022     8B3D3D3D A                  DATA      X'8B3D3D3D'              'BS'  ,ENQ-->IL,NAK-->IL,VT--->IL
   01 00023     80833D3D A                  DATA      X'80833D3D'              'FF'  ,CR->'NL',SO--->IL,SI--->IL
                                  *    1
   01 00024     3D3D4C2C A                  DATA      X'3D3D4C2C'            DLE-->IL,DC1-->IL,DC2-->PN,DC3-->RS
   01 00025     4F833D5E A                  DATA      X'4F833D5E'            DC4-->PF,  'NL'  ,SYN-->IL,ETB->EOB
   01 00026     8B3D8E3D A                  DATA      X'8B3D8E3D'            CAN-->BS,EM--->IL,SUB->.NE.,ESC->IL
   01 00027     3D3D3DE4 A                  DATA      X'3D3D3DE4'            FS--->IL,GS--->IL,RS--->IL,SP ATTN
                                  *    2
   01 00028     8D3D3D3D A                  DATA      X'8D3D3D3D'            INDX->LF,FS--->IL,GS--->IL,RS--->IL
   01 00029     3D3D0758 A                  DATA      X'3D3D0758'            US--->IL,   SYN  ,   '/'  ,CRCMFLX
   01 0002A     136D3D5D A                  DATA      X'136D3D5D'               '*'  ,  'NL'  ,EOT->IL ,  'BS'
   01 0002B     642F6E70 A                  DATA      X'642F6E70'               ')'  ,  'HT'  ,   'LF' , BE->'#'
                                  *    3
   01 0002C     8486853D A                  DATA      X'8486853D'             F ATTN , BS ATTN, X ATTN ,   SYN
   01 0002D     D2C3C8D3 A                  DATA      X'D2C3C8D3'             U ATTN , ( ATTN , ) ATTN , T ATTN
   01 0002E     D53DD7D5 A                  DATA      X'D53DD7D5'             S ATTN ,   SYN  , C ATTN , O ATTN
   01 0002F     3D3D8788 A                  DATA      X'3D3D8788'               SYN  ,   SYN  , R ATTN , N ATTN
                                  *    4
   01 00030     40707670 A                  DATA      X'40707670'              SPACE ,   SUB  ,APL'DCD',   SUB
   01 00031     6A4A7070 A                  DATA      X'6A4A7070'            APL'MIN',APL'EPS',   SUB  ,   SUB
   01 00032     66595811 A                  DATA      X'66595811'           APL'DLTA',APL'NEGN',  CENTS ,   '.'
   01 00033     51745341 A                  DATA      X'51745341'            '<'=>'.',   '('  ,   '+'  ,OR--->DE!
                                  *    5
   01 00034     68707046 A                  DATA      X'68707046'               '&'  ,   SUB  ,   SUB  ,  QUAD
   01 00035     70524570 A                  DATA      X'70524570'               SUB  , ENCODE ,CIRCULAR,   SUB
   01 00036     70700144 A                  DATA      X'70700144'               SUB  ,   SUB  ,EXCL. MK,   '$'
   01 00037     78AC2B60 A                  DATA      X'78AC2B60'               '*'  ,   ')'  ,   ';'  ,NOT->'+-
                                  *    6
   01 00038     37A66570 A                  DATA      X'37A66570'               '-'  ,   '/'  ,MAXIMUM,   SUB
   01 00039     72707562 A                  DATA      X'72707562'           APL'D-ARW',   SUB  ,APL'BMG',APL'SPST
   01 0003A     7070A73B A                  DATA      X'7070A73B'               SUB  ,   SUB  , CRCMFLX,   ','
```

117

```
  2                                      PCC      0
  1*        00000000             2741ARUB SET      0
  3                                      DEF      SSTD,SSTDLC,SSTDUC
  4         01 00000             SSTDLC   EQU      $        SELECTRIC STANDARD LOWER CASE TO EBCDIC
  5                              *
  6                              *   0
  7   01 00000    405AA391 A              DATA     X'405AA391'       SPACE ,EXCL. MK,  LC'T' ,  LC'J'
  8   01 00001    F4969361 A              DATA     X'F4969361'        '4'  ,  LC'O' ,  LC'L' ,   '/'
  9   01 00002    F57D8597 A              DATA     X'F57D8597'        '5'  ,   '''  ,  LC'E' ,  LC'P'
 10   01 00003    12030214 A              DATA     X'12030214'     PN-->DC2,RES-->ETX,BY-->STX,PF-->DC4
 11                              *   1
 12   01 00004    F24B957E A              DATA     X'F24B957E'        '2'  ,   '.'  ,  LC'N' ,   '='
 13   01 00005    A9000000 A              DATA     X'A9000000'      LC'Z' , UNUSED , UNUSED , UNUSED
 14   01 00006    F6899298 A              DATA     X'F6899298'        '6'  ,  LC'I' ,  LK'K' ,  LC'Q'
 15   01 00007    00081700 A              DATA     X'00081700'     UC->N.A.,  'BS'  ,EOB->ETB,LC->N.A.
 16                              *   2
 17   01 00008    F194A787 A              DATA     X'F194A787'        '1'  ,  LC'M' ,  LC'X' ,  LC'G'
 18   01 00009    F0A288A8 A              DATA     X'F0A288A8'        '0'  ,  LC'S' ,  LC'H' ,  LC'Y'
 19   01 0000A    F799845E A              DATA     X'F799845E'        '7'  ,  LC'R' ,  LC'D' ,   ';'
 20   01 0000B    130D2005 A              DATA     X'130D2005'     RS-->DC3,NL--->CR,LF->INDX,  'HT'
 21                              *   3
 22   01 0000C    F3A5A486 A              DATA     X'F3A5A486'        '3'  ,  LC'V' ,  LC'U' ,  LC'F'
 23   01 0000D    F9A68260 A              DATA     X'F9A68260'        '9'  ,  LC'W' ,  LC'B' ,   '-'
 24   01 0000E    F881836B A              DATA     X'F881836B'        '8'  ,  LC'A' ,  LC'C' ,   ','
 25   01 0000F    04160100 A              DATA     X'04160100'      'EOT' ,IL-->SYN,PRE->SOH,DEL->IGN
```

118

```
27         01 00010                  SSTDUC    EQU       $       SELECTRIC STANDARD UPPER CASE TO EBCDIC
28                                   *
29                                   *    0
30   01 00010    404FE3D1 A                    DATA      X'404FE3D1'        SPACE ,DGR-->OR,   'T'  ,   'J'
31   01 00011    5BD6D36F A                    DATA      X'5BD6D36F'          '$'  ,   'O'  ,   'L'  ,QUEST MK
32   01 00012    6C7FC5D7 A                    DATA      X'6C7FC5D7'          '%'  ,  QUOTE ,   'E'      'P'
33   01 00013    12030214 A                    DATA      X'12030214'      PN-->DC2,RES->ETX,BY-->STX,PF-->DC4
34                                   *    1
 1*  01 00014    7C4CD54E A                    DATA      X'7C4CD54E'          '@'  ,   '<'  ,   'N'  ,   '+'
36   01 00015    E9000000 A                    DATA      X'E9000000'          'Z'  , UNUSED , UNUSED , UNUSED
 1*  01 00016    6AC9D2D8 A                    DATA      X'6AC9D2D8'        CENTS ,   'I'  ,   'K'  ,   'Q'
 2*        00000000                            DO        2741ARUB=1
 3*                               *S*          DATA      X'00181700'      UC-->N/A,BS-->CAN,EOB->ETB,LC-->N/A
 4*                                            ELSE
38   01 00017    00081700 A                    DATA      X'00081700'      UC->N.A.,  'BS'   ,EOB->ETB,LC->N.A.
 1*                                            FIN
39                                   *    2
40   01 00018    5FD4E7C7 A                    DATA      X'5FD4E7C7'     '+-'->NOT,   'M'  ,   'X'  ,   'G'
41   01 00019    5DE2C8E8 A                    TEXT      ')SHY'               ')'  ,   'S'  ,   'H'  ,   'Y'
42   01 0001A    50D9C47A A                    TEXT      '&RD:'               '&   ,   'R'  ,   'D'  ,   ':'
43   01 0001B    13152005 A                    DATA      X'13152005'      RS-->DC3,  'NL'  ,LF->INDX,  'HT'
44                                   *    3
45   01 0001C    7BE5E4C6 A                    TEXT      '#VUF'               '#'  ,   'V'  ,   'U'  ,   'F'
46   01 0001D    4DE6C26D A                    DATA      X'4DE6C26D'          '('  ,   'W'  ,   'B'  ,UNDERLINE
 1*  01 0001E    5CC1C36E A                    DATA      X'5CC1C36E'          '*'  ,   'A'  ,   'C'  ,   '>'
48   01 0001F    04160100 A                    DATA      X'04160100'        'EOT' ,IL-->SYN,PRE->SOH,DEL->IGN
49                                   *
```

119

```
  1*   01 0003B     48777B47 A              DATA      X'48777B47'          '%'  ,UNDRLINE,'>'=>','QUEST MK
 88                                   *  7
 89    01 0003C     64605070 A              DATA      X'64605070'      APL'AND' ,APL QUOT,APL OVER,  SUB
 90    01 0003D     44705870 A              DATA      X'44705870'      APL'LE' ,   SUB  , APL'GE',   SUB
  1*   01 0003E     70746B70 A              DATA      X'70746B70'         SUB  ,APL'OR' ,   ':'  ,   '#'
  2*   01 0003F     5009A849 A              DATA      X'5009A849'         '@'  ,   '''  ,   '='  ,QUOTE MK
 93                                   *  8
 94    01 00040     7039363A A              DATA      X'7039363A'         SUB  ,  LC'A' ,  LC'B' ,  LC'C'
 95    01 00041     2A0A3323 A              DATA      X'2A0A3323'       LC'D' ,  LC'E' ,  LC'F' ,  LC'G'
  1*   01 00042     26197070 A              DATA      X'26197070'       LC'H' ,  LC'I' ,   SUB  ,   SUB
 97    01 00043     70707070 A              DATA      X'70707070'         SUB  ,   SUB  ,   SUB  ,   SUB
 98                                   *  9
 99    01 00044     70031A06 A              DATA      X'70031A06'         SUB  ,  LC'J' ,  LC'K' ,  LC'L'
100    01 00045     2112050B A              DATA      X'2112050B'       LC'M' ,  LC'N' ,  LC'O' ,  LC'P'
101    01 00046     1B297070 A              DATA      X'1B297070'       LC'Q' ,  LC'R' ,   SUB  ,   SUB
102    01 00047     70707070 A              DATA      X'70707070'         SUB  ,   SUB  ,   SUB  ,   SUB
103                                   *  A
104    01 00048     70702502 A              DATA      X'70702502'         SUB  ,   SUB  ,  LC'S' ,  LC'T'
105    01 00049     32313522 A              DATA      X'32313522'       LC'U' ,  LC'V' ,  LC'W' ,  LC'X'
106    01 0004A     27147070 A              DATA      X'27147070'       LC'Y' ,  LC'Z' ,   SUB  ,   SUB
107    01 0004B     70707070 A              DATA      X'70707070'         SUB  ,   SUB  ,   SUB  ,   SUB
108                                   *  B
  1*   01 0004C     70077464 A              DATA      X'70077464'         SUB  ,BK SLASH, (BRACE , BRACE)
  2*   01 0004D     74647070 A              DATA      X'74647070'      (BRACK , BRACK) ,   SUB  ,   SUB
111    01 0004E     70707070 A              DATA      X'70707070'         SUB  ,   SUB  ,   SUB  ,   SUB
112    01 0004F     7070EF70 A              DATA      X'7070EF70'         SUB  ,   SUB  ,LOSTDATA,   SUB
113                                   *  C
  1*   01 00050     4079767A A              DATA      X'4079767A'       SPACE  ,   'A'  ,   'B'  ,   'C'
115    01 00051     6A4A7363 A              DATA      X'6A4A7363'         'D'  ,   'E'  ,   'F'  ,   'G'
  1*   01 00052     66597070 A              DATA      X'66597070'         'H'  ,   'I'  ,   SUB  ,   SUB
117    01 00053     70707070 A              DATA      X'70707070'         SUB  ,   SUB  ,   SUB  ,   SUB
118                                   *  D
119    01 00054     70435A46 A              DATA      X'70435A46'         SUB  ,   'J'  ,   'K'  ,   'L'
120    01 00055     6152454B A              DATA      X'6152454B'         'M'  ,   'N'  ,   'O'  ,   'P'
121    01 00056     5B697070 A              DATA      X'5B697070'         'Q'  ,   'R'  ,   SUB  ,   SUB
122    01 00057     70707070 A              DATA      X'70707070'         SUB  ,   SUB  ,   SUB  ,   SUB
123                                   *  E
```

```
  1*   01 00058   37706542 A          DATA   X'37706542'     '-'  ,   SUB  ,  'S'  ,  'T'
125    01 00059   72717562 A          DATA   X'72717562'     'U'  ,   'V'  ,  'W'  ,  'X'
126    01 0005A   67547070 A          DATA   X'67547070'     'Y'  ,   'Z'  ,  SUB  ,  SUB
127    01 0005B   70707070 A          DATA   X'70707070'     SUB  ,   SUB  ,  SUB  ,  SUB
128                              *   F
129    01 0005C   24201030 A          DATA   X'24201030'     '0'  ,   '1'  ,  '2'  ,  '3'
130    01 0005D   04081828 A          DATA   X'04081828'     '4'  ,   '5'  ,  '6'  ,  '7'
131    01 0005F   38341353 A          DATA   X'38341353'     '8'  ,   '9'  ,APL MULT, APL DIV
132    01 0005F   4101707F A          DATA   X'4101707F'  APL'ARROW',APL'B-ARROW',SUB, 'DEL'
133                                   END

CONTROL SECTION SUMMARY: 01 00060   PT 0
```

UTS TECHNICAL MANUAL

Control Functions

Terminology

a. Input Char(s) - The graphic characters typed at the keyboard to invoke the action. If d fferent invocation are available on the 2741 than on TTY, the 2741 is given on a second line.

b. Carriage Position - The (best estimate of the) physical position of the carriage on the dev ce. This is maintained for three purposes: insertion of local carriage returns, tabulation control, and insertion of idle characters on 2741's for timing carriage returns. CPI indicates the position at the beginning of the input message.

c. Record size - The number of characters transmitted to the user program as the result of the Input.

d. EBCDIC code - The input code passed to the user program by the COC Handler for a read request.

e. Echo - The resultant graphics appearing on the terminal printer as a result of the input (if not echoplex part of the graphic is due to local printing).

f. Activation - The condition under which the Input causes the outstanding M:READ to be satisfied. The codes used have the following meanings:

1) Always Activate

2) Never Activate

3) Activate if special activation 1 or 2 (See below), or Record Size

4) Activate if special activation 1 or Record Size

5) Activate if DELTA reading or special activation 1 or record size.

6) Activate only if Record Size reaches requested size.

UTS TECHNICAL MANUAL

Special activation 1 will activate for the special graphics and teletype control characters defined below.

Special activation 2 will activate for the teletype control characters defined below and for EOT activation on 27415.

The special graphics characters are:

]] [} { \ " = ' @ # : ? > _ % , ^

/ - ;) * $! & 1 +) < . ¢

The teletype control characters are:

SOH, STX, ETX, HT, ACK, BEL, BS, ENQ, NAK,
VT, SO, SI, DLE, DC2, DC4, SYN, ETB, CAN

g. Special Action - Any special action taken as a result of the input. Where a toggle is indicated, the default is listed as the second action.

h. Immediate or Deferred - Indicates whether special action is taken when the character is received or is deferred until echo time.

CONTROL FUNCTIONS

Input Char(s) (First Set for TTY, Second Set for 2741)	Carriage Position	Record Size	EBCDIC Code	Echo	Activation	Special Action	(I)mmediate or (D)eferred
break B ATTN, ATTN (if no input)	0	N. A.	N. A.	CR LF NL	N. A.		I
ESCY, Y^c, ESC ESC Y ATTN	0	N. A.	N. A	CR LF NL	N. A.	Escape to TEL	I
ESC Q None	+2	+0	N. A.	!!	2	None	I
X^c X ATTN	CPI	0	N. A.	CR LF X NL	2	Delete all input and output	I
ESC X None	CPI	0	N. A.	X CR LF	2	Delete current input line	D
Rubout, ESC Rubout BS ATTN	+1 -1	-1	N. A.	\\ nothing (also see *2)	2	Delete previous character (also see *2)	D
None BS	-1	+1	08	*2	3	*2	D
ESC P none	+2	+0	N. A.	P \\	2	Set or Reset Half Duplex Paper Tape Mode	D

Input Char(s) (First Set for TTY, Second Set for 2741	Carriage Position	Record Size	EBCDIC Code	Echo	Activation	Special Action	(I)mmediate or (D)eferred
ESC C C ATTN	+2 +1	+0	N. A.	C \ C	2	Set or Reset Tab Relative Mode	D
ESC CR, ESC LF N ATTN	0	+0	N. A.	CR LF N NL	2	Issue Local Carriage Return	D
none O ATTN	+1	+0	N. A.	O	2	Set or Reset Overstrike Edit Mode	D
X ON none	+0	+0	N. A.		2	Set Full on Half Duplex Paper Tape Mode	D
X OFF none	+0	+0	N. A.		2	Reset Full or Half Duplex Paper Tape Mode	D
ESC F F ATTN	0	+1	OD	F\CR LF F NL	1	Report End-of-File	D
L^c, ESC L L ATTN	0	+1	OC	None L	1	Force Form to Top of Next Page	D

Input Char(s) (First Set for TTY, Second Set for 2741	Carriage Position	Record Size	EBCDIC Code	Echo	Activation	Special Action	(I)mmediate or (D)eferred
Non-Printing Control Characters	+0	+1	XDS EBCDIC	Input Code is echoed	3		D
Special Graphics (Non-Alphanumerics)	+1	+1	XDS EBCDIC	Input Code is echoed	4		D
Upper and Lower Case Alphabet	+1	+1	*4	*4	6		D
Numerics	+1	+1	F0-F9	0-9	6		D

Input Char(s) (First Set for TTY, Second Set for 2741	Carriage Position	Record Size	EBCDIC Code	Echo	Activation	Special Action	(I)mmediate or (D)eferred
CR NL	0	+1	OD	CR LF NL	1		D
LF Upper Case NL	0	+1	15	CR LF NL	1		D
FS (L^{cs}) none	+0	+1	1C		1		D
GS (M^{cs}) none	+0	+1	1D		1		D
RS (N^{cs}) none	+0	+1	1E		1		D
US (O^{cs}) SPACE-ATTN	+0	+1	1F		1		D
I^c, HT, ESC I Tab	*3	*3	*3	*3	5	*3	D
/ , =,), or ↑ Must be followed by ATTN	+1	+1	XDS EBCDIC	/ , = ,), or ↑	5		D
[none	+1	+1	B4 for TTY33 - 37 [for TTY 33-37 4F for 7015 1 for 7015		4		D
] none	+1	+1	] for TTY33-37 B5 for TTY 33-37 5F for 7015 ¬ for 7015		4		D

CONTROL FUNCTIONS

Input Char(s) (First Set for TTY, Second Set for 2741	Carriage Position	Record Size	EBCDIC Code	Echo	Activation	Special Action	(I)mmediate or (D)eferred
ESC U U ATTN	+2 +1	+0	N. A.	U \ U	2	Set or Reset Restrict Alphabetics to Upper Case Mode	D
ESC ((ATTN	+2 +1	+0	N. A.	(\ (	2	Interpret Alphabetics Normally	D
ESC)) ATTN	+2 +1	+0	N. A.	) \ )	2	Interpret Upper Case Alphabetics as Lower	D
none Upper Case Shift	+0	+0	N. A.		2	Select Upper Case Half of Keyboard	D
none Lower Case Shift	+0	+0	N. A.		2	Select Lower Case Half of Keyboard	D
ESC T T ATTN	+2 +1	+0	N. A.	T \ T	2	Reset or Set Tabs Simulation Mode	D output only
ESC S S ATTN	+2 +1	+0	N. A.	S \ S	2	Reset or Set Space Insertion Mode	D
ESC E none	+2	+0	N. A.	E \	2	Reset or Set Echoplex Mode	D
ESC R R ATTN	CPI + Current Record Size	+0	N. A.	R \ R +Re-typing of the input line	2	Retype the effective Current Input Line	D

NOTES

*1 - The break signal causes one of several actions to take place in the following hierarchy:

a. If four consecutive breaks have been received without other intervening Input, treat as Y^c.

b. If an M:INT has been issued by the running program, honor it.

c. If DELTA is in control, go to DELTA.

d. Escape to TEL.

*2 - If Overstrike Edit Mode (O ATTN) is in effect, BS is preempted as an editing character. BS ATTN also takes on special meaning as does SPACE under certain circumstances. In the Overstrike Edit Mode normal input is identical to that when the mode is OFF. However, the BS character is merely treated as a cursor positioner. After (one or more) BS characters has been received the following rules apply:

a. The size of the record does not change (except by BS ATTN or X ATTN)

b. SPACE is treated as a forward cursor positioner.

c. Normal Characters are stored over the character at the current cursor position.

d. BS ATTN is treated as a SPACE to replace the current character (i.e., the character at the carriage position before the BS) and two spaces are echoed to position the cursor properly.

e. All attention sequences are honored but also cause the cursor to move 1 position.

f. Normal rules continue to apply when the cursor reaches the position it had before the first BS.

g. Any record delimiter causes the record to be accepted as it currently exists.

h. Tab characters are treated as specified in *3 below, i.e., it is a tab character to be stored or n SPACES for cursor positioning depending on the state of the space insertion (ESC S) switch.

*3 - The tab character causes a variety of actions (upon output, echoing, and the resultant input record) depending upon the device type, the state of the Tab Relative Mode (ESC C), the Echoing Mode (ESC E), the Tab Simulation Mode (ESC T), and the Space Insertion Mode (ESC S).

The Tab Relative Mode is meaningless for output. For input the mode specifies that tabs are to be considered relative to the beginning of the input record. The tab stops (if present) are thus adjusted for each operation by the amount of the initial carriage position. In further discussion Tab Stops are defined as the effective tab stops after adjustment.

The remaining discussion is presented in tabular form with the following parameters defined:

CPOS - Current Carriage Position
CPI - Carriage Position of the Beginning of an input message
ARSZ - Number of characters accumulated in current input message
TRSZ - Difference between Size of input message if space insertion were on and ARSZ.

When no tab specifications are present, a value of one greater than current carriage position is assumed, but if physical tabbing is involved the carriage is assumed to move 10 positions. The following table illustrates the results of a tab character when received as a function of affecting modes of operation:

Tabs	ESCC	ESCT	ESCS	Devices	Tab Stop (TS)	Echo	ARS Z	TRS Z	CPOS
N	0	0	1	TTY 33, 7015	N. A.	b̸	b̸	N. A.	CPOS+1
N	0	0	0	TTY 33, 7015	N. A.	b̸	HT	N. A.	CPOS+1
N	1	0	0	TTY 35, 37	N. A.	HT	HT	N. A.	CPOS+1
N	1	0	1	TTY 35, 37	N. A.	HT	b̸	N. A.	CPOS+1
N	1	1	0	TTY 35, 37	N. A.	b̸	HT	N. A.	CPOS+1
N	1	1	1	TTY 35, 37	N. A.	b̸	b̸	N. A.	CPOS+1
N	0	0	0	TTY 35, 37, 2741	N. A.	nil	HT	N. A.	CPOS+1
N	0	0	1	TTY 35, 37, 2741	N. A.	nil	b̸	N. A.	CPOS+1
Y	0	0	0	TTY 33, 7015	Stop after	b̸	HT	TS-CPI-ARSZ-1	CPOS+1
Y	0	0	1	TTY 33, 7015	CPI+ARS Z +TRS Z	b̸	(TS-CPI- ARS Z) b̸	0	CPOS+1
Y	0	1	0	TTY 33, 7015		(TS-CPOS) b̸	HT	TS-CPI-ARSZ-1	TS
Y	0	1	1	TTY 33, 7015		(TS-CPOS) b̸	(TS-CPI- ARS Z)b̸	0	TS
Y	0	0	0	TTY 35, 37, 2741	Stop after	nil	HT	TS-CPI-ARSZ-1	TS
Y	0	0	1	TTY 35, 37, 2741	CPI+ARS Z +TRS Z	nil	(TS-CPI- ARS Z)b̸	0	TS
Y	1	0	0	TTY 35, 37	Stop after	HT	HT	TS-CPI-ARSZ-1	TS
Y	1	0	1	TTY 35, 37	CPI+ARS Z +TRS Z	HT	(TS-CPI- ARS Z)b̸	0	
Y	1	1	0	TTY 35, 37		(TS-CPOS) b̸	HT	TS-CPI-ARS Z-1	TS
Y	1	1	1	TTY 35, 37		(TS-CPOS) b̸	(TS-CPI- ARS Z)b̸	0	TS
N	1	0	0	2741	Not Allowed				

*4 - Upon receiving an upper or lower case Alphabetic character (after device shifts are accounted for, of course) two possible transformations take place. First, if ESC) has been received, Upper Case Alphabetics are transformed to Lower Case. Then, if 'Restrict Alphabetics to Upper Case' is in effect (ESC U), all Lower Case Alphabetics are transformed to Upper Case.

Output Action

When an M:WRITE is executed, presenting a record to the COC handler, the following actions take place (unless DRC and BIN is specified):

a. If the DCB has VFC specified, the first character is examined. Then:

1) If the character is X'F1' a new page is issued.

2) If the character is X'CX', X upspaces are issued. If the bottom margin is reached a new page is issued and no further upspacing is done.

3) If the character is X'60' or X'E0' this fact is memorized. These characters specify 'inhibit upspace'.

4) If not 1, 2, or 3 the first character is ignored.

b. If the record contains more than three trailing blanks, all are suppressed. However, if the entire record consists of blanks, a single blank will be output.

c. The characters remaining are translated and sent to the terminal except where special action is indicated. The following characters invoke special action:

Null	00	Terminate Character Processing
HT	05	See Below
FF	0C	A new page is issued
CR	0D	CR and NL are issued to TT 's
LF	15	CR and NL are issued to TTY's
CR	0D	NL followed by appropriate number of idles
LF	15	are sent to 2741's

UTS TECHNICAL MANUAL

LF (specific)	20	Line feed only is issued
[	B4	[on TTY's ! on 7015
]	B5	] on TTY's ¬ on 7015
!	4F	! on 7015 [on TTY's
¬	5F	¬ on 7015] on TTY's

Lower/case alphabetics send upper/case alphabetics on TTY33, 7015, and some 2741 terminals.

HT causes the following actions:

Tabs	ESCT	Device	Transmitted	CPOS
N	0	TTY 33, 7015	b̸	CPOS+1
Y	0	TTY 33, 7015	b̸	CPOS+1
Y	1	TTY 33, 7015	b̸ to next stop	Next Stop
N	0	TTY 35, 37, 2741	HT	CPOS+10
N	1	TTY 35, 37, 2741	b̸	CPOS+1
Y	0	TTY 35, 37, 2741	HT	Next Stop
Y	1	TTY 35, 37, 2741	b̸'s to Next Stop	Next Stop

d. After all characters are processed (or Null is encountered), the calling DCB is checked. IF M:UC or if the line terminates with CR, LF, SYN, or specific LF (x '20') no further action takes place. Otherwise a CR, LF is sent to the terminal unless the format control character was X'60' or X'E0', in which case, a CR only is sent (inhibit upspace.)

e. In the course of the output, line length control and pagination control are maintained.

If DRC and BIN is specified (indicating transparent text), the record as presented by the user is transmitted exactly with no special functions performed and no translation.

SIZE AND TIMING

Approximately 2000 words of memory are required for COC handling routines, and are allocated as follows:

1. Input and output interrupt routines take up 500 instructions.

2. Read/write routines are comprised of 500 instructions.

3. Activation detection and echoing routines contain 400 instructions.

4. Get/put buffering routines have 200 instructions.

5. Line detection and intialization routines have 200 instructions.

6. The teletype translation table requires 65 words of memory.

7. Miscellaneous tables and constants comprise the remaining 135 words.

Additional storage is required for each communication line in the system; 23 bytes for control information and eight words (average) for buffering input and output messages.

IBM 2741-type terminal translation tables are available via SYSGEN parameters for EBCD and standard code sets.

Four translation tables are ava lable for 2741-like terminals, allowing translation of EBCD and Selectric (r) code sets with either standard or APL keyboards. Each translation table adds 96 words to storage requirements if incorporated in a system.

Assembly parameters have been defined to allow conditional assembling of the procedure concerning 2741 terminal logic, page headings, performance monitoring, and buffer security checking. Assembling out all of these will reduce core requirements by 760 words.

UTS TECHNICAL MANUAL

Approximate execution times in microseconds are:

Write processing - per write	250
additional per character	140
Read processing - per read	580
additional per character	220
Input interrupt processing - per character	110
Output interrupt processing - per character	80
Buffering routines - per 14 characters buffered	110

Assuming an average write size of 40 characters and an average read size of ten characters, the per character execution time will be approximately 235 μsec on output and 399 μs on input. Average terminal I/O rates of one character input and four characters output per second per user result in an overhead burden of 13.4% of a SIGMA 7CPU per one hundred users.

UTS TECHNICAL MANUAL

ID

COC - Control Routine

PURPOSE

Provide common entry and exit for terminal I/O CAL1 processing.

USAGE

Effective, BAL, 11 COC: Actually a branch to COC from the I/O scheduler which was originally called via R11.

INPUT PARAMETER:

R8 FCN, DCB address
FCN - function code in byte 0
0 - read BCD
1 - read direct BCD
2 - read BIN
3 - read direct BIN (transparent)
4 - write BCD
5 - write direct BCD
6 - write BIN
7 - write direct BIN (transparent)

SUBROUTINES

COCWR	called if the function code is a write operation.
COCRD	called if the function code is a read operation.
WTMSGSIZ	called to record performance data.

INTERACTION

COC	called from the I/O scheduler (IOQ) for terminal I/O.
SETTYC	called to set up the type of completion code returned from COCWR or COCRD in the user's DCB.

DESCRIPTION

The byte count is extracted from the DCB (BLK field) as is the buffer address (QBUF field) which is then converted to a byte address with the HBTD field of word 0 of the DCB added. The line number is extracted from the M:UC DCB.

Control is passed to COCRD or COCWR dependent upon a valid value for FCN. If FCN is invalid, then control is returned to the caller (R11) after setting the TYC field of the DCB to 3.

Upon return from COCRD or COCWR, SR1 contains the ARS value which is then stored in that field of the DCB. D1 contains the TYC value which is put in the DCB via a call to SETTYC before returning to the original CAL1 caller.