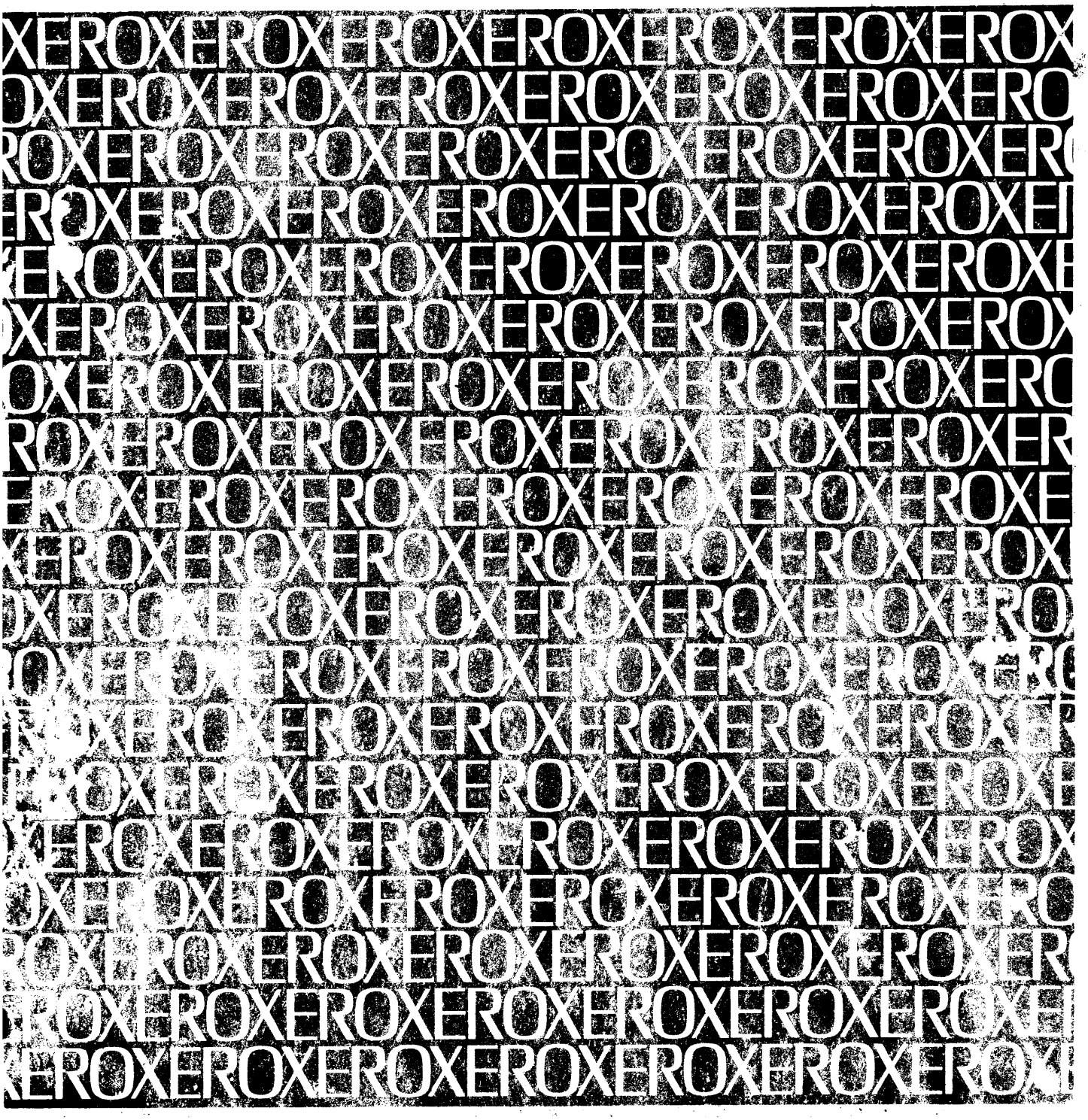
Xerox Universal Time-Sharing System (UTS)

Sigma 6/7/9 Computers

Operator Communication and Monitor Services Technical Manual



90 19 88

Xerox Universal Time-Sharing System (UTS)

Sigma 6/7/9 Computers

Operator Communication and Monitor Services Technical Manual

First Edition

90 19 88A

February 1973

Price: $5.50

Printed in U.S.A.

NOTICE

This publication documents operator communication and Monitor services of the Universal Time-Sharing System (UTS) for Sigma 6/7/9 computers. All material in this manual reflects the C01 version of UTS.

RELATED PUBLICATIONS

Title	Publication No.
UTS Overview and Index Technical Manual†	90 19 84
UTS Basic Control and Basic I/O Technical Manual	90 19 85
UTS System and Memory Management Technical Manual	90 19 86
UTS Symbiont and Job Management Technical Manual	90 19 87
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.

The specifications of the software system described in this publication are subject to change may depend on a specific configuration of equipment such as additional tape units or larger for details.

CONTENTS

UTS TECHNICAL MANUAL

Operator Communications

FUNCTIONAL OVERVIEW

The operator controls UTS through keyins, and the system communicates with the operator by typing out messages on his console. All keyins are handled by the KEYIN overlay segment, although some keyin command routines are contained in separate assemblies. For example, control is passed to the symbiont communication module (SYMCOM) for all symbiont device keyins, to the module DELPRI for delete and priority keyins, and to the display module (DISPLAY) for processing display keyins.

INTERFACES

Since the processing of most key-ins involves checking or changing of UTS tables, these constitute the principal interface between the key-in functions and other parts of UTS. The KEYIN processor itself is a separate overlay segment in the Monitor. KEYIN is a ghost job which is initiated as a result of end action processing of input from the operators console.

Examples of KEYIN processor interfaces are the !TIME and !DATE keyins which change the time and date in the table containing current time and date; the !X key-in which causes the abort event to be reported to the scheduler and the !DISPLAY key-in which interrogates tables and outputs current status information such as the amount of disc space remaining.

OPERATIONAL OVERVIEW

The processing of a key-in command is initiated by the operator pressing the control panel interrupt. This causes control to go to the control panel interrupt routine, which enqueues the output message for that console as well as an input request. The input request causes the light on the operator's console to illuminate. The operator then types his command and hits the N/L (new line) key which causes the I/O interrupt. The I/O interrupt routine checks that the interrupt is from the operator's console and that an expected input has been completed, and then passes control to T:ADDGHOST which causes the KEYIN JIT to be loaded. Next control passes to an interpretive exit instruction in the JIT (CAL1,9 1) which in turn causes the scheduler to pull the KEYIN ghost environment and execute instructions in registers 13-15 which branch to T:OV which associates the KEYIN overlay with the KEYIN ghost (i.e. loads and transfers control to KEYIN processor).

Table HA-2 Index of KEYINs

Key-in	Function	Processing Routines
ABORT, id	Abort user or job	KIABORT
D month, day, year	Enter date	KIDATE
DATE month, day, year	Enter date	
DELETE id,[yyndd]	Delete symbiont file from system	KIDEL *
DISPLAY[option]	Send system information to operator	KIDIS *
E, id	Error (terminate) job step-go on to next job step	KIERROR
ERROR, id	Error (terminate job step-go on to next job step	
GJOB name	Initiate ghost job	KIGJOB
INT id	Transfer control to user's console interrupt routine	ENTINT
MOUNT ndd$_1$[, ndd$_2$] [, reel#] [Public / Lock / , id]	Inform Monitor that tape or disk pack is mounted	KIMOUNT
OFF	Allow no more users to log on	KIGDOWN
ONB n	Set maximum number of batch users	KIGBUP
ON n	Set maximum number of on-line users	KIGUP
PRIORITY id, yyndd, priority	Change user priority	KIPRI *
REQUEST ndd	Prepare to dismount tape from unit ndd	KIREQ
REQUEST 7T	Request a 7-track tape unit	
REQUEST 9T	Request a 9-track tape unit	
REQUEST MT	Request any available tape unit	
S	Search for input symbiont files to run	KISTART
SCRATCH ndd$_1$[, ndd$_2$] reel #	Informs Monitor that scratch unit has been made available	KISCRTH
SEND message	Broadcast message to on-line users	KISEND
START	Search for input symbiont files to run	KISTART
Syyndd, option	Initiate symbiont action (both local and remote batch)	DEVKEYIN *
T hour, minute	Enter time	KITIME
TIME hour, minute	Enter time	
X, id	Abort user or job	KIABORT
yyndd, action	Initiate action indicated in response to device message	DEVKEYIN
ZAP	Abort all on-line users and active batch jobs	KIFDOWN

* These routines are not in KEYIN module.

Table HA-2 (cont'd) Index of KEYINS

Key-in	Function	Processing Routine
ANSS[CRATCH] ndd$_1$[. ndd$_2$] [, BLP] [, reel#]	ANS scratch tape	KIANSS
ANSM[OUNT] ndd$_1$[. ndd$_2$] [, BLP]	ANS tape is mounted	KIANSM
OVER ndd [, reel#]	Override output tape rejection	KIANSO
RBSEND [&RBndd / WSN] message	Send message to a RBT	KRBSEND
RBBDCST message	Send message to all RBT's	KRBBCST
RBX	Disconnects RBT's and ignores new connections	KRBX
RBS	Allow new RBT connections	KRBS
RBDI[SC] [&RBndd / WSN]	Disconnect the specified RBT	KRBDISC
RBSW[ITCH] WSN, [CP / LP], user	Switch output files	KRBSWIT

Table HA-1

Each time a KEYIN message is processed an eight word context block is obtained from TSTACK, in order to initialize following KEYIN parameters:

KIPL - KEYIN Parameter List

Word:	
0	KDLM
1	KCCP
2	KFLAGS
3	KBUF
4	KFL
5	KPLB
6	
7	

where

KDLM are delimiters. First byte represents number of delimiters; last three bytes contain the byte address of all possible delimiters for the field being processed.

KCCP is the current character position (the byte index of next character of the key-in).

KFLAGS is a blank active flag. If zero, (leading) blanks are to be skipped.

KBUF is the buffer address (word address of the key-in buffer).

KFL is the field length (number of characters in the field currently being processed).

KPLB is the field buffer (contents of current field, left justified with trailing blanks).

ID

KEYIN1 - Keyin Main Control

PURPOSE

Perform preliminary processing of messages which are input from the operators console, and pass control to the appropriate routine for specific processing of the KEYIN command.

USAGE

Called from T:OV routine which loads and transfers control to the KEYIN overlay as a result of end action processing of operator console input.

INPUT

KEYINBUF Pointer to KEYIN message.

OUTPUT

Output is a function of which KEYIN command is being processed and is described in the command routine writeup.

INTERACTION

KEYIN is a ghost job which is initiated as a result of:

1. Interrupt from the computer console
2. Interrupt handler in IOQ outputs a prompt (!) and issues a READ M:OC with end action.
3. The operator inputs his message and hits new line.
4. The end action initiates the KEYIN ghost by calling T:ADDGHOST which causes the KEYIN JIT to be loaded.
5. An interpretive exit (CAL1, 9 1) in the KEYIN JIT causes the scheduler to pull the KEYIN ghost environment and execute instructions in registers 13-15 which passes control to T:OV to load and transfer to the KEYIN overlay.

SUBROUTINES

GKIFLD	Obtain first KEYIN field.
KEYERR	KEYIN error exit routine.
KEYINR	KEYIN normal exit routine.
WDTBLSRH	Search KEYIN name table.

KIABORT	Process "ABORT" or "X" KEYIN
KIDATE	Process "DATE" or "D" KEYIN
KIERROR	Process "ERROR" or "E" KEYIN
KISTART	Process "START" or "S" KEYIN
KITIME	Process "TIME" or "T" KEYIN
KMOUNT	Process "MOUNT" KEYIN
KISCRTH	Process "SCRATCH" KEYIN
KIGBUP	Process "ONB" KEYIN
KIGUP	Process "ON" KEYIN
KIGDOWN	Process "OFF" KEYIN
KIFDOWN	Process "ZAP" KEYIN
KISEND	Process "SEND" KEYIN
KIGJOB	Process "GJOB" KEYIN
KIDEL	Process "DELETE" KEYIN
KIPRI	Process "PRIORITY" KEYIN
[illegible]Q	Process "REQUEST" KEYIN
KIDIS	Process "DISPLAY" KEYIN
DEVKEYIN	Process a device KEYIN if a search of the KEYIN name table fails to find a command processor key.
KIANSS	Process "ANSSCRATCH" KEYIN
KIANSM	Process "ANSMOUNT" KEYIN
KIANSO	Process "OVER" KEYIN
KRBSEND	Process "RBSEND" KEYIN
KRBBCST	Process "RBBDCST" KEYIN
KRBX	Process "RBX" KEYIN
KRBS	Process "RBS" KEYIN
KRBDISC	Process "RBDISC" KEYIN
KRBSWIT	Process "RBSWITCH" KEYIN

DESCRIPTION

First an eight word context block is obtained from TSTACK after which the KEYIN parameter list (TABLE HA-1) is initialized.

If the first character in the KEYIN buffer is a new line, control passes to the normal exit routine (KEYINR); otherwise GKIFLD is called to obtain the command field from the KEYIN buffer. If no error is returned, WDTBLSRH is called to compare the first 4 characters of the command field with the list of command keys in the KEYIN name table (KITBL). If a match is found control passes to the KEYIN jump table (KIJMPTBL) indexed by the found key which in turn branches to the specified command routine. If the command key is not found in the KEYIN name table control is transferred to DEVKEYIN to determine whether the message is a device keyin. If it is not a device keyin, the error routine KEYERR is called. Normal returns from command routines go through KEYINR; error returns go through KEYERR.

ID

WDTBLSRH - Search Keyin Name Table

PURPOSE

To search a specified word table for a match on a specified key.

USAGE

BAL, SR4 WDTBLSRH

Input parameters: R1 = key word
R2 = ADDR of word table to search
R3 = Length of word table
R4 = Return ADDR if search fails

Output parameter: R1 = Index of found key

INTERACTION

WDTBLSRH is called from the keyin main control routine (KEYIN1) to search the keyin command name table and from the process device routine (DEVKEYIN) to search the keyin device option table.

DESCRIPTION

This routine searches a word table for a specified key and returns, if found, with the index of the searched for key in R1. If the key is not found in the table specified, the not found return address in R4 is taken.

ID

KEYERR, KEYERR1, KEYINR - Keyin exit routines

PURPOSE

Entry at KEYERR or KEYERR1 causes the message "EH?" or "LATER", respectively, to be output on the OC device. All three entries result in releasing the parameter list space from TSTACK, restoring registers 5-11 from TSTACK, and exiting to the step routine T:DELUS to delete the keyin ghost.

USAGE

B KEYERR	Entered to send "LATER" before exit
B KEYERR1	Entered to send "EH?" before exit
B KEYINR	Normal exit

DATA BASE

OCFLG	to turn off keyin in progress flag

INTERACTION

OCQUEUE	Called to output later or "EH?" message to the operator's console.
CTRIG	Called to initialize keyin for retry.
T:DELUS	Exit to step to delete the keyin ghost.

DESCRIPTION

If a message to the operators console is specified (i.e. entry at KEYERR or KEYERR1), the output message code (LATER or EH) is put in R1. Next R7 is set to zero to indicate a canned message is to be output and the operator's console queue routine is called to queue the message for output.

Next CTRIG is called to re-initialize the keyin ghost job for retry and control passes to the normal exit entry (KEYINR). KEYINR releases the parameter space from TSTACK and clears the keyin in progress flag (OCFLG) after which control passes to T:DELUS to delete the current keyin ghost job.

UTS TECHNICAL MANUAL

ID

KIABORT, KIERROR, ENTINT - Abort, error or interrupt a job (system id)

PURPOSE

KIABORT — To process "ABORT" or "X" keyin (abort an entire job or log off an on-line user).

KIERROR — To process "ERROR" or "E" keyin (terminate a job step).

ENTINT — To process "INT" keyin (simulate break to transfer control to batch user's interrupt routine).

USAGE

B KIABORT
B KIERROR
B ENTINT

Input parameters: R7 = Keyin parameter list
SR1 = Delimiter after command field

ROUTINES

GKIFLD — Called to obtain the id field from the keyin buffer.

CVSYSID — Called to convert the id from EBCDIC to HEXIDECIMAL.

GETUSER# — Called to obtain the active user number of the id.

T:RUE — Called to report the abort, error or break event

KEYERR — Keyin error exit routine

KEYINR — Keyin normal exit routine

DATA BASE

LPART — Number of batch partitions
PLH:SID — Indexed by partition number to verify active batch user.

INTERACTION

Entered from keyin main control (KEYIN1) via the keyin jump table (KIJMPTBL, indexed by the abort, X or error, E or INT command keys in the keyin name table (KITBL).

DESCRIPTION

When the entry is KIABORT or KIERROR, J:CCBUF is set non-zero. ENTINT sets J:CCBOF zero so that special processing can take place for the INT keyin. The KIABORT, KIERROR, and ENTINT entries initialize the event code to abort (E:ABRT), error (E:ERR), or break (E:CBK) respectively, after which control passes to a common path that checks for a legal delimiter (, or blank) in SR1. If illegal, control is passed to KEYERR for retry. Otherwise the get next field routine (GHIFLD) is called to obtain the job id.

CVSYSID is then called to convert the id from EBCDIC to hexidecimal. If the conversion is unsuccessful (illegal id) control is passed to KEYERR for retry. If the INT keyin is being processed (J:CCBUF=0) PLH:SID is searched (indexed by LPART) to determine if the ID given in the keyin is an active batch user or ghost job. If not exit is to KEYINR. When it is control follows the normal path of KIABORT and KIERROR. GETUSER# is called to obtain the user number corresponding to the id. If the id is not an active user, control is passed to KEYERR for retry. Otherwise, T:RUE is called to report the abort or error event to the scheduler after which control passes to the normal exit routine (KEYINR).

ID

KIDATE - Date

PURPOSE

Process "DATE" or "D" keyin

USAGE

B KIDATE
Input parameters: R7 = Keyin parameter list
SR1 = Delimiter after date command field

SUBROUTINES

GDTKIVAL	Called to obtain and check the validity of the month, day, and year keyin values.
KEYERR	Error exit routine if an illegal month, day or year value was input, or if the delimiter before the date field is not a blank.
KEYINR	Normal exit routine

DATA BASE

Date is set to EBCDIC MMDD; DATE+1 is set to EBCDIC bbYY

INTERACTION

Entered from keyin main control (KEYIN1) via the keyin jump table (KIJMPTBL), indexed by the "DATE" or "D" command keys in the keyin name table (KITBL).

DESCRIPTION

First a check is made for a legal delimiter in SR1. If not a blank, control is passed to the error exit routine (KEYERR) for retry. Otherwise the get date/time routine (GDTKIVAL is called three times to obtain the month, day and year keyin values. If these values are not legal or if a comma delimiter is not found after the month and day field, control is passed to KEYERR. Otherwise, the two-digit EBCDIC values of month and day are stored in the system date entry, and the two digit EBCDIC value of the year is stored (right justified) in DATE+1. Control is then passed to the normal exit routine (KEYINR).

ID

KIDEL - DELETE Keyin

PURPOSE

To process the keyin to delete the input or output symbiont file created by or for a symbiont device (YYNDD) if specified, or to delete all input and output files associated with the job id if no symbiont device is specified.

USAGE

B KIDEL

Input parameters R7 = Keyin parameter list
SR1 = Delimiter after the "DELETE" command field.

ROUTINES

DPSCAN	Called to obtain the system id and DCT index from the keyin message.
SRCHF	Called to search the symbiont file directory and delete the item keywords for specified entries.
KEYERR	Keyin error exit routine.
KEYINR	Keyin normal exit routine.
MBS	called by SRCHF to unlink an input Q entry
ADD	Called by SRSCHF to release all serial links, zero resource requirements and insert job in abort Q.

INTERACTION

KIDEL is contained in the module DELPRI. It is logically a part of keyin and is entered from keyin main control (KEYIN1) via the keyin jump table (KIJMPTBL), indexed by the "DEL" command key in the keyin name table (KITBL).

DESCRIPTION

First DPSCAN is called to obtain the system id (and DCT index for the device code if specified). Upon return the last delimiter in the message is checked for a new line. If not, control passes to KEYERR. Otherwise, the id of the user (and DCTX of the device code if specified) is passed to the search symbiont file routine (SRCHF) where one or all entries in the symbiont file directory belonging to the specified id are partially deleted, depending on the presence of a device code field that specifies a particular file to be deleted. (The actual file release is initiated by SRCHF activating the output symbiont, which reads and releases the files with partially deleted entries in the symbiont file directory.) For input Q entries SRCHF calls MBS to unlink the entry from its tables then it calls ADD to release all links and set all resource requirements to zero placing the job in the abort Q. When CCI processes the job it will be aborted with a message informing the user that the operator aborted the job.

ID

KIDIS - Display main control routine

PURPOSE

To perform preliminary error checks and initialization for processing display keyin commands.

USAGE

B KDIS

Input parameters: R7 = Keyin parameter list
SR1 = Delimiter after display command field.

OUTPUT

Output messages are a function of the display option specified, and are included in the option routine writeups.

ROUTINES

GKIFLD	Called to obtain display option field.
NXKICHR	Called to obtain next character if the delimiter after the option field is not a new line.
GMB	Called to obtain a monitor buffer to establish a message area
KEYERR	Entered to output "LATER" at the operator's console if a monitor buffer is not available.
KDSNOP	Entered if no option field is specified.
KDS04	Entered to process "DISC" option
KDSTAP	Entered to process "TAPES" option
KDS07	Entered to process "JOB" option
CVSYSID	Called to verify the option field as a valid id if none of the above display options are found.
KDSYID	Entered to process "ID" option.
KDSVOL	Called to process "VOLUME" option
ALLPARTS	Called when no option specified

INTERACTION

The display routine is physically a part of the keyin overlay segment, but is contained in a separate assembly named DISPLAY. It is entered from keyin main control (KEYIN1)

via the keyin jump table (KIBJPTBL) indexed by the display command key in the keyin name table (KITBL).

DESCRIPTION

When control is passed to the DISPLAY routine it calls subroutine BLKIT which calls the get monitor buffer routine (GMB) to establish a message buffer. If a buffer is not available it keeps calling GMB until successful. The buffer is then initialized and control is passed to the option processor KIDIS1 with the buffer address in registers R14, R6 and R3 is initialized to zero. If the delimiter after the display command is not a blank or new line, control passes to KEYERR. If the delimiter is a new line, no option is specified, and the logic to obtain and verify the option field is bypassed; otherwise GKIFLD is called to obtain the option field. If the field length returned is invalid or if the delimiter is not a new line or blank, control passes to KEYERR. If the delimiter is a blank, the get next keyin character routine (NXKICHR) is called until a non-blank is returned, and if not a new line, control passes to the release monitor buffer routine (RMB) with the exit address of KEYERR.

After the pre-processing described above is complete the display main control routine analyzes the display option field and passes control to the appropriate routine to process the specified option.

ID

KDS04 - DISC option control

PURPOSE

To process "DISC" display option.

USAGE

Entered from Display main control routine if the display option field is "DISC".
Input parameters: R6 = Buffer address containing message area.

OUTPUT

The following message is output at the operators console:

(NL) USER = (#GRANULES) DC(#GRANULES) DP SYMBIONT = (#GRANULES) (NL)

DATA BASE

GRANRAD
GRANPACK } counts in tables updated by ALLOCAT
GRANSYM

ROUTINE

CONVERT	Called to convert HEX count to EBCDIC
KDS8CHM	Called to put message segments into message buffer.
TYPEIT	Called to output message to operators console
RMB	Called to release monitor buffer

DESCRIPTION

GRANRAD, GRANPACK and GRANSYM are accessed to obtain the corresponding type count. The count is then converted from HEX to EBCDIC by routine CONVERT, KDS8CHM is used to place the counts and the EBCDIC type name into the message buffer. When all types have been inserted into the ubffer, TYPEIT is called to output the message to the operators console. Then control is transferred to the release monitor buffer routine (RMB) with an exit address to KEYINR

ID

TYPEIT - OUTPUT DISPLAY message, get and initialize a monitor buffer.

PURPOSE

Calls JTYPEA to output the message at the operators console.

USAGE

Used by all Display option routines. BAL, 15 TYPEIT

Input parometers:

R3 = number of bytes to output
R6 = address of monitor buffer

Output:

R3 = initialized to zero
R6 = address of monitor buffer
R14 = address of monitor buffer

ROUTINES

JTYPEA - Called to output the message in the monitor buffer to the operators console. At end action, the buffer is released.

DESCRIPTION

R3→R14, R6→R13 and calls JTYPEA. When the message is queued, return to entry BLKIT which gets and initializes a monitor buffer. If a call is made by BAL, 15 BLKIT a monitor buffer is initialized and return to the caller with the output as described above.

ID

KDS07 - Job DISPLAY option

PURPOSE

Display the id of all batch jobs currently in execution.

USAGE

Branched to from the display main control routine if "JOB" is specified in the option field.

Input parameter: R6 = Buffer containing message area
R3 = 0 = Count of characters in buffer

OUTPUT

The following message is output at the operator's console for each job currently in execution:

(NL) ID = (JOB ID) ACCT = (USER ACCOUNT) PART = (PARTITION NUMBER JOB IS EXECUTING IN)

DATA BASE

PLH:SID - To obtain the id of the job currently executing in this partition.

LPART - number of batch partitions

PLD:ACT - To obtain the account of the job currently executing in this partition.

ROUTINES

RMB - Called to release a monitor buffer

IDTOBCD - Called to convert a HEX id to EBCDIC (HALF word in-full word out)

XTOBCD - Called to convert a HEX# to EBCDIC (byte in-half word out)

TYPEIT - Called to output the above message to the operators console and return with a new buffer.

DESCRIPTION

Each partition is checked to see if there is a job currently executing. If so, PLH:SID and PLD:ACT are indexed by the partition number to fetch the id and account for the user of that partition. The id is converted to EBCDIC by calling IDTOBCD, the partition number is converted by calling XTOBCD. With the information in the buffer TYPEIT is called to output the message to the operators console and get a new buffer. This is repeated until all jobs have been displayed when an exit is made to the release monito buffer routine with an exit address of KEYINR.

ID

KDSNOP - No DISPLAY options

PURPOSE

To process Display keyins where no option field is present.

USAGE

Branched to from Display main control routine if the option field is not present.
Input parameters: R6 = Address of buffer containing message area.
R3 = 0=Count of characters in buffer

OUTPUT

The following message is output at the operator's console for each file in the symbiont file directory:

(NL) SYYNDD, ID, PRIORITY (NL) - for output files
(NL) ID P ACCOUNT SP 7T AT CO TIME OA VALID PARTITIONS - for input files

ROUTINES

KDSFSRCH	-	Called to obtain first/next entry (item keyword) from the symbiont file directory
KDSEYNDP	-	Called to construct the device code message segment "(NL)SYYNDD" in the message buffer and return the priority in EBCDIC in D2
ALLPARTS	-	Called to display the input Q
IDTOBCD	-	Called to convert HEX ID to EBCDIC
TYPEIT	-	Called to output message to operators console and obtain a new buffer

DESCRIPTION

First KDSNOP sets J:CCBUF to zero to differentiate between a DISPLAY ID. This cell is checked before transfer to ALLPARTS to verify the type of display requested. Next KDSFSRCH is called to get the first/next item keyword from the symbiont file

directory. Next KDSEYNDP is called to obtain the device code from DCT1 and the priority from byte 1 of the item keyword, and to construct the message segment "(NL)SYYNDD, " in the message buffer and to construct the priority segment "P(NL)" right justified in D2. Upon return the priority segment is put in the message buffer, and IDTOBCD is called to convert the id in bytes 2 and 3 of the item keyword to EBCDIC. Upon return the id message segment is put in the message buffer. Next TYPEIT is called to print the buffer, get a new monitor buffer and return to KDSNOPA to get the next entry. When all of the output files in SYMFILE have been listed control is passed to ALLPARTS to display the input Q. (See HA.02.04 for the descriptive operation of ALLPARTS).

ID

KDSYID - ID DISPLAY option

PURPOSE

Entered to process "ID" display option

USAGE

Branched to from the Display main control routine if the option field is determined to be a valid id.

Input parameters: R6 = Buffer containing message area.
R3 = 0 = Count of characters in buffer

OUTPUT

The following message is output at the operator's console for each file belonging to the specified id:

(NL) SYYNDD, (PRIORITY)(NL) - Output Files
(NL) ID P ACCOUNT SP 7T 9T CO TIME OA VALID PARTITIONS - INPUT FILE

ROUTINES

KDNXT - Called to find input file with this ID

KDSFSRCH - Called to get the first/next symbiont file key word in the symbiont file directory.

FINDSID - Called to find if an input file with this ID exists.

KDSEYNDP - Called to construct the device code message segment "(NL)SYYNDD, " in the message buffer and the priority segment "(P)(NL)" right justified in D2.

TYPEIT - Called to output message to operators console and obtain a new buffer

DESCRIPTION

First KDSYID stores the ID in J:CCBUF. This cell is checked before transfer to KDNXT to verify the type of display requested. KDSFSRCH is called to get the first/next item keyword from the symbiont file directory. If the system id (bytes 2 and 3 of keyword) match the id in the display option field, KDSEYNDD is called to obtain the device code, and set up the message segment "(NL) SYNDD", in the message buffer. The priority segment returned in D2 is put in the message buffer and TYPEIT is called to print the

buffer, get a new buffer and return to KDSYID1 to continue checking for entries with this ID. When all entries in SYMFILE have been checked, control is transferred to KDNXT. KDNXT calls FINDSID to determine if an input file with this ID exists. If an entry exists in the input Q, a call is made to HEADER to print the header, then to NORUN to get the corresponding information for this entry. After all entries have been printed control is transferred to the release monitor buffer routine (RMB) with an exit address to KEYINR.

ID

KDSTAP - TAPES DISPLAY option KDSVOL - Display all tapes and packs

PURPOSE

To process display "TAPES" option

USAGE

Branched to from Display main control routine if the option field is "TAPES".

Input parameters: R6 = Address of buffer message area.
R3 = 0 = Current character count

OUTPUT

The following message is output at the operators console for each magnetic tape in the system:

(NL)YYNDD { EMPTY / SCRATCH / USER / SYSTEM } (REEL#) { IN USE {ID} / AVAILABLE } (NL)

(The reel number and status message segments apply only to user or scratch tapes)

DATA BASE

RMB - Called to release monitor buffer
AVRTBL - To check tape status (i.e. empty, system, user or scratch).
DCT - To obtain the DCT index of entries with the device type assigned to magnetic tape.
AVRID - To obtain the id of the USER when a tape is in use

UBAT - Called to check if user# is on-line or batch. If batch return with ID
KDSEYNDP - Called to construct the device code message segment "(NL)SYYNDD", in the message buffer and the priority message segment "P(NL)" (right justified) in D2.
4 BYTE - Called to convert HEX to BCD, put 4 Bytes in Buffer
KDS8FRTO - Called to put a message segment in the message buffer.
TYPEIT - Called to output the message in the buffer, return with a new buffer to KDSTAPA.
ANSFLGS - To determine if an ANS tape is on a certain drive.
AVRFNMT - To obtain the file name for an ANS tape.

DESCRIPTION

First R2 is initialized to the device index of the first magnetic tape in the device control table (BATAPE). If no entries are present, the buffer is released and control passes to KEYERR. Otherwise, KDSEYND is called to construct the device code segment "(NL)SYYNDD," upon return the segment is modified to "(NL)YYNDD" and put in the message buffer.

Next the status (i.e. empty, system, user or scratch) is determined from the relevant AVR table entry and the appropriate message segment constructed and put in the message buffer by calling KDS8FRTO.

If the status was user or scratch the reel number and state (i.e. in use or available) are obtained from the AVR table and the appropriate message segment constructed and put in the message buffer by calling KDS8FRTO, or 4BYTE. If the drive is ANS the serial number is dehashed via SIXBACK before placement into the message buffer. Also, for ANS drives, the AVRFNMT is examined to obtain the filename and the julian date which are placed into the message buffer.

After each message is constructed, TYPEIT is called to print the message on the operators console, get a new buffer and return. When all tape entries have been checked, the release monitor buffer routine (RMB) is called to release the buffer with an ext address of KEYINR.

KDSVOL

This entry sets the search to the entire length of the AVRTBL. KDSTAP searches only the tape entries in the AVRTBL.

ID

KDS08 - USER DISPLAY Option

PURPOSE

Operator KEYIN to display all system users' ID number.

USAGE

Branched to from the Display main control routine if the option field is 'USER'.

DATA BASE

Input comes from system tables, SMUIS and UB:US, which are resident in the UTS monitor.

SMUIS — is a constant created at sysgen time and equals total number of users in system. Content of SMUIS is used for indexing.

UB:US — is a byte table containing the user's state. Zero (0) means no user. SW equals X'E', which means the user is asleep or ghost.

OUTPUT

Output of user system ids is received on OC device in the following format:

BATCH:	IC,19, 18, 0F, 05, 02	no leading zero, all
ONLINE:	28, 27, 26, 25, 24, 23, 2I, 20, 1F, 1E, 1D, 1B, 17; 16, 14, 13, 10, 0C, 0B, 0A, 09, 08, 07, 03	entries are 4 char pos. blank filled
ASLEEP:	12, 0D, 06, 04	

INPUT

R6 = BUFFER ADDRESS
R3 = 0 = Current Characters in buffer

ROUTINES

RMB - Called to release the monitor buffer
KDS8CHM - Called to store 8 BYTES in buffer
UBAT - Called to check if user number is online or batch. If batch return with ID.

4 BYTE - routine to convert 2BYTE HEX value to 4 BYTE EBCDIC, strip off zeros filling in blanks and calling KDS8FRTO to place word in buffer.

TYPEIT - Called to output the buffer on the operators console and return with a new buffer.

DESCRIPTION

The value SMUIS is used as an index into UB:US to pick up each active user. When a user is found, his type is determined and, if that is the type being output it is stored in the buffer. When the buffer is full (10 id's) or the search has completed for this type the buffer is output by TYPE IT. One search is made for batch, online, and asleep. At the end of the third pass, control is transferred to the release monitor buffer routine with an exit address of KEYINR.

ID

ALLPARTS

PURPOSE

List all input Q entries as described in "HEADER"

USAGE

B ALLPARTS

INPUT: R6 = address of message buffer
R3 = current character position in message buffer

DATA BASE

BB:HPRI	-	Head of priority chain - (byte table indexed by priority)
BB:LINK	-	Link chain (byte link table for input queue)
S:BFIS	-	Count of batch files in system

ROUTINE

HEADER	-	called to output header line on operators console
NORUN	-	called to put information about a user in the message buffer
TYPEIT	-	called to output the message buffer on operators console
RMB	-	called to release the monitor buffer

INTERACTION

ALLPARTS is entered to process the !Q keyin from the Display option processing routine KIDISI. It will also be entered from KDSNOP after all output symbiont files have been listed.

DESCRIPTION

When ALLPARTS in entered S:BFIS is tested for being non-zero and if not exit is to the release monitor buffer routine (RMB) with an exit address of KEYINR. If non-zero a call is made to HEADER to output the header line on the operators console. BB:HPRI is indexed by priority to find a non-empty priority chain. The entry at the head of the chain is then processed with a call to NORUN to retrieve and store the users data in the message buffer. Then TYPEIT is called to output the message on the operators console. Upon return from TYPEIT, the link from the head of the chain is used as an index into BB:LINK to get the next entry in this priority chain. If there is an entry it is processed as above and the chain (BB:LINK) is searched and each entry processed until no more entries are found. Then the head of the next priority chain is tested and processed as above. This procedure is done for each priority chain. When the last chain has been processed control is transferred to the release monitor buffer routine (RMB) with an exit address of KEYINR.

ID

KIFDOWN - Force the system to quiescence

PURPOSE

Process "ZAP" keyin and abort all jobs and users.

USAGE

B KIFDOWN

Input parameters: R7 = Keyin parameter list

OUTPUT

Number of on-line users allowed (S:QUAIS) is set to zero.
Number of batch users allowed (S:BUAIS) is set to zero.

DATA BASE

UB:US is checked for active users

ROUTINES

T:RUE - Called to report the E:ABRT event to the scheduler for all active on-line users.

KIGDOWN - Exit from KIFDOWN which clears S:QUAIS and passes control to KEYINR.

INTERACTION

KIFDOWN is entered from keyin main control (KEYIN1) via the keyin jump table (KIJMPTBL), indexed by the "ZAP" command key in the keyin name table (KITBL).

DESCRIPTION

Each user status entry (UB:US) in the user table is checked for active status. If an active user is not batch or current user (i.e. keyin ghost or ALLOCAT), the scheduler routine T:RUE is called with the E:ABRT event in R6 and the user number to abort in R5. After aborting all current terminal jobs KIFDOWN exits to KIGDOWN which zeros the number of on-line users allowed (S:OUAIS) to prevent any further on-line and batch users from logging on. Flag GOODNGT is set negative. This flag is used to indicate that when all processing has terminated the resident HGP's are to be written to the RAD.

ID

KIGDOWN - Gentle down the system

PURPOSE

To process "OFF" keyin to prevent new users from logging on.

USAGE

B KIGDOWN

Input parameters: R7 = keyin parameter list
SR1 = delimiter after "OFF" command field

DATA BASE

S:OUAIS - The number of on-line users allowed is set to zero.
S:BUAIS - The number of batch users allowed is set to zero.

ROUTINES

KEYINR - Keyin normal exit return

INTERACTION

KIGDOWN is entered from keyin main control (KEYIN1) via the keyin jump table (KIJMPTBL), indexed by the "OFF" command key in the keyin name table (KITBL).

DESCRIPTION

KIGDOWN causes a gentle down by setting the number of batch (S:BUAIS) and online (S:OUAIS) users allowed to zero which prevents any new users from logging on.

UTS TECHNICAL MANUAL

ID

KIGUP - Gentle up

PURPOSE

To process the "ON" keyin to allow users to log on.

USAGE

B KIGUP

Input parameters: R7 = Keyin parameter list
SR1 = Delimiter after "ON" command field.

DATA BASE

SMUIS - The maximum number of users allowed in the system.
S:BUAIS - Number of batch users allowed in the system
S:GUAIS - Number of ghost jobs allowed in the system

ROUTINES

GKIFLD - Called to obtain the number of users specified in the keyin field.

INTERACTION

KIGUP is entered from keyin main control (KEYIN1) via the keyin jump table (KIJMPTBL), indexed by the "ON" command key in the keyin name table (KITBL).

DESCRIPTION

J:CCBUF is set to zero to indicate the keyin being processed is attempting to adjust the number of online users. If the delimiter after the "ON" command is not a blank or if the get field routine returns an error when called to obtain the number of users allowed, control passes to KEYERR. Otherwise this number is added to the number of batch jobs and ghost jobs allowed. If this total is $\leq$ SMUIS it is added to S:OUAIS and exit is to KEYINR. If the total is greater exit is to KEYERR with no modification.

ID

KIGJOB - Initiate ghost job

PURPOSE

To process a "GJOB" keyin for ghost job initialization.

USAGE

B KIGJOB

Input parameters: R7 = Keyin parameter list
SR1 = Delimiter after "GJOB" command field.

ROUTINES

GKIFLD	-	Called to obtain the ghost name.
KEYERR	-	Keyin error exit routine.
T:GJOBSTART	-	Routine in UCAL called to verify the ghost name and to either wake up the ghost if already initialized or return the JIT ADDR if not already initialized.
T:ADDGHOST	-	Routine in SSS called to initialize and schedule the ghost job specified.

INTERACTION

Entered from keyin main control (KEYIN1) via the keyin jump table (KIJMPTBL), indexed by the "GJOB" command key in the keyin name table (KITBL).

DESCRIPTION

If the delimiter is not a blank, control passes to KEYERR for retry. Otherwise the ghost name field is obtained by calling the GKIFLD routine. If an error is returned or the field length is greater than eight, or the ghost name is "KEYIN", control passes to KEYERR. Otherwise T:GJOBSTART is called to verify the name in the ghost job name table and then to either wake the ghost if asleep (i.e. ghost job already initialized) or to return the disc ADDR of the ghost JIT if not asleep. If an error is returned control passes to KEYERR. If the jobwas asleep control passes to the normal exit routine (KEYINR). Otherwise T:ADDGHOST is called with the JIT ADDR of the ghost to be scheduled in R15, after which control passes to KEYINR.

ID

KIMOUNT, KISCRTH, KIANSS, KIANSM, KIANSO

PURPOSE

To process the various MOUNT, SCRATCH, ANSSMOUNT, ANSSCRATCH, and OVER, keyins for labeled, unlabeled, and ANS tapes as applicable. The MOUNT keyin is also applicable to disk packs.

USAGE

B	KIMOUNT
B	KISCRTH
B	KIANSS
B	KIANSM
B	KIANSO

Input Parameters: R7 = input parameter list
SR1 = terminating delimiter of initial field

DATA BASE

DCT16	to verify the device specified as magnetic tape or disk pack.
AVRTBL	current serial number and status of drive.
AVRID	to check who is using the device and how it is being used.
SOLICIT	set when request is made on a device, reset after keyin on that device.
AVRNOU	non-zero implies disk drive is in use.
ANSFLGS	to check status of an ANS drive.
AVRFNMT	the file name table for ANS tapes - unit switching must change.

S:BSPIN	indicates drive is allocated to batch - reset by PUBLIC.
S:OSPIN	indicates drive is allocated to on-line - reset by PUBLIC.
SL:SP	counts of spindles in use. PUBLIC & LOCK must update batch, on-line and ghost counts.

ROUTINES

NDD	verfies the drive specified in the keyin.
GKIFIELD	obtains the next field in the keyin.
DEVCK	called by GKIFIELD to find the specified drive in the DCT table.
KEYERR	error exit routine.
KEYINR	normal exit routine.
CVSYSID	call to verify and convert the job id from EBCDIC to hexadecimal.
SRCHAVR	called to search AVRTBL for matching reel numbers and set up AVR tables for device.
SETNEW	called to set up AVR table entries for device.

DESCRIPTION

There are several logical functions which must be performed for these keyins. They are:

1. Set the flags to indicate the type of keyin.
2. Find the specified drive of the keyin.
3. Check the legality of the keyin.
4. Obtain the next field of the keyin (if any).
5. Obtain the next field of the keyin (if any).
6. Check the availability of the drive.
7. Setting of the AVR-associated tables to reflect the keyin.

NOTE: The AVRTBL slot refers to the relative position of a drive in the parallel AVR-associated tables.

1. Setting the flags to indicate the type of keyin

If entered at KIANSS, KIANSO, or KIMOUNT, the register SR2 is loaded with the respective flags. If entered at KIANSO (OVER keyin), an exit is to KEYERR if it is not an ANS system. If entered at KISCRTH the latter part of keyin command is checked to verify a scratch keyin. Common processing for all the keyins is then passed to the location KIMOUNTZ to find the specified drive of the keyin.

2. Finding the specified drive of the keyin

The routine NDD is called to analyze the next field of the keyin. It must be three characters and be the same name as a DCT16 field. This gives the DCT index of the specified drive. If a unit switch is not specified, control passes to NOUNTSW to check the legality of the keyin (next section).

A legal unit switch must be on a tape unit, the keyin must have been solicited and it must not be during an OVER keyin. The reel # in the originally requested AVRTBL slot is cleared, and AVR:TPOS is set to the AVR indes of the new slot. The AVRID and AVRFNMT (ANS systems) are transferred from the old slot to the new slot. The SOLICIT byte of the old slot is cleared and control is passed to check the legality of the keyin.

3. Check the legality of the keyin

The ANSFLGS are examined for consistency of this keyin response to the existing environment. An OVER keyin is legal only if the system is semi-protected and the error flags are set. A MOUNT or SCRATCH is legal if SOLICIT = 0 or if SOLICIT $\neq$ 0 and the request was non-ANS. The ANSFLGS and ANSSCRATCH are legal if SOLICIT = 0 or if the request was ANS. The ANSFLGS are set to indicate the type of keyin received, and AVR:SCH id set if this is a SCRATCH keyin. If AVR:VER is set and SOLICIT = 0, an error is reported (if SOLICIT = 0, then verification is really occuring on the disk pack).

4. Obtain the next field of the keyin (if any)

If no field is present, control is passed to NOIDPUB to perform a few checks and if possible, to wake-up the user (see below). The next field is checked for the BLP option. If present, the ANSFLGS are set and a reel # is obtained. A reel # specified in a keyin is illegal under the following conditions.

a. OVER keyin and its an ANS volume.
b. ANSMOUNT keyin.
c. ANSSCRATCH in a protected system.
d. An ANS reel # is not equal to 6 characters or a labeled tape reel # is not equal to 4 characters.

5. Obtain the next field of the keyin (if any).

If no field is present, control is passed to the location PREMOUNT to check the availability of the drive. The next field is checked; PUBLIC control goes to the location PUBLK; LOCK goes to the location LOCK; or the system id register is set after CVSYSID is called to verify that a legal hexadecimal number was entered. In the case where an id is specified, control is passed to check the availability of the drive.

The PUBLIC option is illegal under the following conditions:

a. The specified unit is not a disk pack.
b. The unit is in exclusive use (AVRID≠0) and not locked (AVRID = -1 is LOCKED).
c. AVR:PUB is already set.
d. The reel # already in the AVRTBL doesn't match that of the keyin and the drive is in use (AVRNOU≠0).

To make the drive public, a user is uncharged for the unit. This consists of re-setting S:BSPIN or S:OSPIN and decrementing SL:SP for batch or on-line respectively. The ghost count in SL:SP is incremented, the public bit in the flags register is set and AVRNOU is incremented. Control is then passed to enter the AVRTBL entry without further checks.

The LOCK option results in an error if the drive is in exclusive use or the reel numbers of the existing AVRTBL entry and the keyin do not match. If the drive is public, AVRNOU is decremented and if not in use anymore (AVRNOU=0), the SL:SP ghost slot is decremented. Then for both public and private slots, AVRID is set to A-1 and AVR:PUB is zeroed. Control is then passed to enter the AVRTBL entry without further checks.

6. Check the availability of the drive

If SOLICIT $\neq 0$ and an ANSMOUNT, ANSSCRATCH or OVER keyin; control is passed to enter the AVRTBL entry without further checks (the OVER keyin also clears ANSFLGS:ERR). If SOLICIT$\neq$0 and it is not an ANS tape keyin, the availability checks are also bypassed. A drive is considered available if AVR:AVR$\neq$0 and AVR:PUB$\neq$0. It's now time to set up the AVR table entries. If the reel # of the AVRTBL matches that of the keyin, then control goes to NOIDPUB; otherwise, all flags in the AVRTBL, except AVR:SCR, are cleared and entry is made via SRCHAVR.

7. Setting the AVR-associated tables to reflect the keyin

There are two crucial routines residing in the AVR module which initialize an AVRTBL entry and unblock the user requesting the reel (if any); SRCHAVR and SETNEW.

SRCHAVR is entered when the reel number specified in the keyin does not match what already is there.

SETNEW is entered by all paths going to NOIDPUB. These paths have been noted in previous paragraphs. NOIDPUB performs the following, if no reel # is in the AVRTBL slot or if a scratch request was made (a -1 in the reel # slot), then the error exit is taken. Otherwise, SETNEW is called to enter the AVRTBL entry without further checks.

ID

KIPRI - Keyin priority

PURPOSE

To change the priority of a file in the symbiont file directory, or batch input Q

USAGE

B KIPRI

Input parameters: R7 = Keyed parameter list
SR1 = Delimiter after the "PRIORITY" command field.

ROUTINES

DPSCAN - Called to obtain the system id and device index from the keyin message.

GKIFLD - Called to obtain priority keyin field.

SRCHF - Called to change the priority field in the item keyword for specified entries in the symbiont file directory.

KEYERR - Keyin error exit routine.
KEYINR - Keyin normal exit routine.
MBS - Called by SRCHF to remove the entry from its present priority chain, and called again to insert it into the new priority chain.

INTERACTION

The KIPRI routine is physically a part of the keyin overlay segment but is contained in a separate assembly named DELPRI. It is entered from keyin main control (KEYIN1) via the keyin jump table (KIJMPTBL) indexed by the priority command key in the keyin name table (KITBL).

DESCRIPTION

First DPSCAN is called to obtain the system id (and DCTX of the device code if specified). Upon return the delimiter of the last field, i.e. id (or device code if specified), is checked and if not a comma, control passes to KEYERR. Otherwise GKIFLD is called to obtain the priority. If the priority field is not 0-F, control passes to KEYERR.

Otherwise, for output files the priority is combined with the id (and DCTX if specified) and SRCHF is called to change the priority field in the item keyword of the entries in the symbiont file directory with the specified id (and DCTX if specified). For input files when SRCHF finds the ID valid calls MBS to remove the entry from its present priority chain, and calls MBS again to insert it into the new priority chain.

ID

KIREQ - Request a tape drive or private pack

PURPOSE

To process the REQUEST commands to release a specified drive and to allocate a drive for mounting a new tape.

USAGE

B KIREQ

Input parameters: R7 = Address of keyin parameter list
SR1 = Delimiter after the "REQUEST" command field.

OUTPUT

If the appropriate device is available, a message is made up to output the appropriate instruction:

nnd - If the unit is ready

nnd DISMOUNT SCRATCH reel number - If a scratch tape is on the unit.

nnd DISMOUNT AND SAVE reel number - If the tape on the unit is to be saved.

ROUTINES

GKIFLD	-	Called to obtain the device code (ndd) or type (7T, 9T, MT) field.
KEYERR	-	Called if an error is detected in the device code or type field.
HEXCK	-	Called to verify and convert the device code to a hexidecimal digit.
KEYERR1	-	Called to output "LATER" if the device is not available.
TYPEIT	-	Called to print the message on the operators console get and initialize a new buffer.

INTERACTION

Entered from keyin main control (KEYIN1) via the keyin jump table (KIJMPTBL), indexed by the request command key in the keyin name table (KITBL).

DESCRIPTION

If the delimiter after the request command is not a blank, control passes to KEYERR for retry. Otherwise GKIFLD is called to obtain the next keyin field. If an error condition is returned, control passes to KEYERR for retry. If the field length is two, the request is of the form request "7T/9T/MT" and the type is determined. If the type is invalid KEYERR is called. Otherwise the AVR table is searched to obtain the first available unit of the type requested. If no unit is available, control passes to the keyin exit routine KEYERR1 which initiates a "LATER' message on the operator's console. If a unit is available, the channel unit is fetched from the device control table. The appropriate message is then constructed and the routine TYPEIT is called to print the message, after which control passes to the release monitor buffer routine with an exit address of KEYINR.

If the field length returned by GKIFLD is not two, the keyin is of the form request "ndd" and the device check routine (DEVCK) is called to verify and convert the device code to hexidecimal digits. If an error is returned o if the device type is not a magnetic tape, control passes to KEYERR for retry. Otherwise the AVR table entry for the device specified is checked and if the unit is not available, control passes to KEYERR1 to output a "LATER" message to the operator's console. If available, a message is constructed and output as described above.

ID

KISEND - Send message

PURPOSE

Process the send keyin to send a broadcast message or a single user message.

USAGE

B KISEND

Input parameters: R7 = Keyin parameter list
SR1 = Delimiter

OUTPUT

If the keyin is of the form SEND "MESSAGE", the message is put in the COC administrative buffer (COCMESS) and included as part of each user's page header output. If or the form SEND, id "MESSAGE", the message is output directly to the specified user.

ROUTINES

KEYERR	-	Keyin error exit routine.
KEYINR	-	Keyin normal exit routine
MOVER	-	Called to move send "MESSAGE" to COC page header administrative buffer (COCMESS).
GKIFLD	-	Called to obtain the id field of the user to receive the message.
CVSYSID	-	Called to convert the id from EBCDIC to legal hexidecimal digits.
GETUSER#	-	Called to find the active user number for the specified user id.
COCPUTBL	-	Called to release COC input buffer(s) if the message to send interrupts a read operation.
COCWR	-	Called to output a SEND, id "MESSAGE" to the terminal specified by id.
COCWD	-	Called to output a line cancel, when necessary, prior to calling COCWR.

DATA BASE

COCMESS	-	Administrative message buffer is used if the message is for all on-line users.
LB:UN	-	Searched for the line corresponding to the user number of the user to receive the message.
STATE	-	Line state modified if necessary to perform write.

INTERACTION

KISEND is entered from keyin main control (KEYIN1) via the keyin jump table (KIJMPTBL), indexed by the send command key in the keyin name table (KITBL).

DESCRIPTION

If the delimiter after the send command field is a new line, the number of characters in the administrative buffer is set to zero (byte 0 of COCMESS) to indicate no message. If the delimiter is a blank the keyin is of the form SEND "MESSAGE" and the message field is searched for a new line character in order to determine the number of characters to send. If zero characters are in the message field, control is passed to the keyin error exit routine for retry. If the message length is greater than 55 (max administrative message), the count is truncated to 55 and stored in byte 0 of COCMESS. Next the move character routine (MOVER) is called to move the message from the keyin buffer to the administrative message buffer (starting at COCMESS+1), after which control passes to the keyin exit routine (KEYINR).

If the delimiter after the send command field is a comma, the keyin is of the form SEND, id "MESSAGE" and the get keyin field routine (GKIFLD) is called to obtain the id of the on-line user to receive the message. Next the convert system id routine (CVSYSID) is called to convert the id from EBCDIC to legal hexidecimal digits. After which the get user number routine (GETUSER#) is called to find the active user number for the id. If any of the above routines return an error, control passes to KEYERR. Otherwise, the user numbers (LB:UN) in the COC line table are searched to find the line to receive the message.

Before sending the message to the COC write routine (COCWR) special processing on the relevant line is performed depending on the current line state:

Inactive or output (INAC OR OUT) - None

Switch to input after output complete (SI) - The state is changed to output and a flag is set to set the state to SI after calling COCWR.

Input or input complete OIN or IC) - This causes the line state to be set to inactive and a flag set to set the state to SI after calling COCWR. The current input line is cancelled by calling the release buffer routine (COCPUTBL) and a underscore (backarrow on some terminals) is echoed to the terminal, by calling COCWD, to indicate the input is cancelled. Next a flag is set in the COC line table that will cause a new line at the terminal after the underscore. The line count and carriage position entries in the COC line are then updated to reflect the start of a new line.

After the above operations are performed the number of characters to send is computed by searching for a new line character in the message field. If the count is zero control passes to KEYERR for retry. Otherwise the DCB ADDR, line number, byte address of message field, and number of characters in field are set up in R1, R6, R7 and SR1 respectively, and the COC write routine (COCWR) is entered directly. Upon return the flag to set the line state to SI is checked and line state changed if specified, after which control passes to the keyin exit routine (KEYINR).

ID

KISTART - Start batch scheduling

PURPOSE

Process "START" or "S" keyin to initiate scheduling of batch jobs from the input symbiont queue.

USAGE

B KISTART

Input parameters: R7 = Address of keyin parameter list
SR1 = Delimiter after the "START" command field.

ROUTINES

T:BTSCHED - Called to schedule batch job
KEYINR - Normal exit routine

INTERACTION

Entered from keyin main control (KEYIN1) via the keyin jump table (KIJMPTBL), indexed by the "START" or "S" key in the keyin name table (KITBL).

DESCRIPTION

Control is given to the scheduler routine T:BTSCHED which checks the input symbiont files for batch jobs and changes the status of each job found to be ready to run until the maximum batch users allowed (SL:BUAIS) is reached. Next control passes to the normal exit routine (KEYINR)

ID

KITIME - Time

PURPOSE

Process "TIME" or "T" keyin

USAGE

B KIDATE

Input parameters: R7 = Keyin parameter list
SR1 = Delimiter after the "TIME" or "T" command field

SUBROUTINES

GDTKIVAL	-	Called to obtain and check validity of hour and minute keyin values.
KEYERR	-	Error exit routine if an illegal hour or minute value is input or if the delimiter after the "TIME" or "T" field is not a blank.
KEYINR	-	Normal exit routine.

DATA BASE

TIME is set to EBCDIC HHMM

INTERACTION

Entered from keyin main control routine (KEYIN1) via the keyin jump table (KIJMPTBL), indexed by the time or T key in the keyin name table (KITBL).

DESCRIPTION

First a check is made for a legal delimiter in SR1. If not a blank, control is passed to the error exit routine (KEYERR) for retry. Otherwise the get date/time routine (GDTKIVAL) is called two times to obtain the hour and minute keyin values. If these values are not legal or if a comma delimiter is not found after the hour, control is passed to KEYERR. Otherwise the two digit EBCDIC values of hour and minutes are stored in the system time entry and control is passed to the normal exit routine (KEYINR).

ID

DEVKEYIN - Device keyin

PURPOSE

To handle device-related key-ins by passing key-ins starting with an "S" to SKEYIN and error-checking all others for the form yyndd, I where I is the processing action indicator (letter C, E, R, or L).

USAGE

B *R4 Where R4 is the not found return address set up by KEYIN1 before calling WDTBLSRH to search for a command key in the keyin name table.

Input parameters: R7 = Keyin parameter list
R1 = First four characters of keyin.

ROUTINES

IOREC - Called to continue activity on a device.

IORECE - Called to error the device.

IORECR - Called to retry activity on a device.

IORECL - Called to lockout all activity on a device.

IORECL - Called to lockout all activity on a device.

SKEYIN - Called to pre-process symbiont keyin.

DEVCK - Called to verify and convert the device code field to hexidecimal digits.

GKIFLD - Called to obtain the device indicator field.

WDTBLSRH - Called to search the device option table for the option specified in the indicator field.

KEYERR - Keyin error exit routine

KEYINR - Keyin normal exit routine

INTERACTION

DEVKEYIN is entered from the word table search routine (WDTBLSRH) if a search of the keyin name table for a command key fails.

DESCRIPTION

DEVKEYIN expects the first keyin field to be of the form yyndd or syyndd (i.e. an active or symbiont device keyin).

The first character of the keyin is checked for the letter "S" which causes a transfer to the symbiont processor routine (SKEYIN). Otherwise the device check routine is called to verify and convert the device code field to hexidecimal digits. If the check fails or device code field length is not five, or if the delimiter after the device code field is not a comma, control passes to the keyin error exit routine (KEYERR) for retry. Otherwise the get keyin field routine (GKIFLD) is called to obtain the command character. The command character (KEY) and pointer to the device command table are set up in R1 and R2 and the word table search routine (WDTBLSRH) is called to search for the device option (command) specified. If not found control passes to KEYERR for retry. Otherwise control passes to the following routines via the device option JUMPTBL indexed by the character command key found.

Device Indicator	Branch Address	
C	IORECC	Continue activity on device.
E	IORECE	Error the device
R	IORECR	Re-read
L	I ORECL	Lockout activity on the device

ID

SKEYIN - Symbiont keyin

PURPOSE

To pre-process symbiont keyins and, if valid, pass control to the symbiont communication processor (SYMCOM).

USAGE

B SKEYIN

Input parameters: R7 = Keyin parameter list
SR1 = Delimiter after the device code keyin field.

Output parameters: D1 = Command character
R2 = DCT index of device to process

ROUTINES

DEVCK - Called to verify and convert the device code field to hexidecimal digits.

KEYERR - Keyin error exit routine.

NXKICHR - Called to get the next character in the keyin buffer.

SYMCOM - Called to process symbiont keyin.

KEYINR - Keyin exit routine.

INTERACTION

SKEYIN is called from DEVKEYIN and KISCRTH to pre-process symbiont keyins.

DESCRIPTION

First the device check routine (DEVCK) is called to verify and convert the device code into hexidecimal digits. If DEVCK returns an error condition or the delimiter after the device code is not a comma, control passes to the keyin error exit routine (KEYERR). Otherwise the get next keyin character routine (NXKICHR) is called to obtain the command character from the keyin buffer. If an error condition is returned, control passes to KEYERR. Otherwise the symbiont communications processor (SYMCOM) is called, and upon return control is passed to the keyin exit routine (KEYINR).

UTS TECHNICAL MANUAL

ID

KIGBUP - Process ONB keyin

PURPOSE

To adjust the number of batch users allowed to run concurrently.

USAGE

B KIGBUP

INPUT:

R7 = Keyin parameter list
SR1 = Delimiter after "ONB" command field

DATA BASE

SMUIS - Maximum number of users allowed in the system

S:OUIAS - Number of online users allowed in the system

S:GUAIS - Number of ghost jobs allowed in the system

ROUTINES

GKIFLD - Called to obtain the number of users specified in the keyin

INTERACTION

KIGBUP is entered from keyin main control (KEYINL) via the keyin jump table (K1JMPTBL) indexed by the "ONB" command key in the keyin name table (K1TBL).

DESCRIPTION

J:CCBUF is set non-zero to indicate that the keyin being processed is attempting to adjust the number of batch users.

If the delimiter after the "ONB" command is not a blank or if the GKIFLD routine returns an error when called to obtain the number of users allowed, control passes to KEYERR. Otherwise this number is added to the number of online users and ghost jobs allowed. This total is compared against SMUIS, which is the total number of users allowed in the system. If it is $\leq$ to SMUIS, S:BUAIS is modified to the new number, and exit is to KEYINR. If the number is greater, exit is to KEYERR.

UTS TECHNICAL MANUAL

ID

KRBBCST - KRBSEND

PURPOSE

To send a message to remote batch terminals

ENTRY

B	KRBBCST	-To send a message to all RBT's
B	KRBSEND	-To send a message to a specified RBT

DATA BASE

RBLIMS	-	Defines the subset of system devices reserved for RBT's
RBD:WSN	-	Used to locate the DCT index from a specified work station name
RBB:ID	-	Tested to see if the RBT is connected
DCT16	-	To obtain the DCT index for a specified device

ROUTINES

GMB	-	Obtains a monitor buffer for the RBT message
DEVCK	-	Checks if ndd is a legal device
SGCQ	-	In the SACT module. Upon entering the routine, registers 12-14 contain information which is put into a remote batch communication buffer and the ghost job is awakened to handle this information.
KRBDCT	-	To obtain the DCT index of the device during a RBSEND keyin.

DESCRIPTION

A unique bit is set in register 12 to reflect the function of that entry point. If entry is at KRBSEND, KRBDCT examines the keyin and obtains the DCT index, utilizing DCT16 or RBD:WSN as appropriate.

Common code is then entered to obtain a monitor buffer, move the message into the buffer and call SGCQ to transfer the information to the remote batch ghost. Exit is through KEYINR.

ID

KRBSWIT

PURPOSE

To switch output files to a remote batch terminal

ENTRY

B KRBSWIT

DATA BASE

KPLB - The keyin communication buffer which contains the keyin

ROUTINES

CVSYSID - Converts the specified system id to a hexidecimal number
SGCQ2 - In the SACT module. Builds remote batch communication buffers and awakens the remote batch ghost job.

DESCRIPTION

The keyin portion of this task is concerned with setting up the necessary registers for the routine SGCQ2. These are:

R12 - The ghost function code for switching files
R13 - The high order byte is the device types to be affected and the low order bytes are the SYSID
R14-R15 - The work station name

ID

KRBX - KRBS - KRBDISC

PURPOSE

The alter various flags which control the connecting or disconnecting of remote batch terminals.

ENTRY

B KRBX
B KRBS
B KRBDISC

DATA BASE

RB:XFLG	-	OFFBIT set here to disconnect terminals
RB:FLAG	-	Examined for the status of the RBT and changed accordingly
RB:LIMS	-	Defines subset of system devices reserved for RBT's.

ROUTINES

KRBDCT	-	Obtains the DCT index of the RBT

DESCRIPTION

RB:FLAG and RB:XFLG are examined for the status of the RBT and are set or reset to accommodate the keyin. The ghost job will then perform the described function.

ID

CVSYSID - Convert system id

PURPOSE

To convert a field of up to four EBCDIC characters into hexadecimal, presumably as a system id value.

USAGE

BAL, SR4 CVSYSID

Input parameters: R7 = Keyin parameter list (character string to convert in the field buffer (KPIR) entry in the keyin parameter list).

Output parameters: R2 = Converted hexadecimal value, right justified. Condition code = 0 if no error. Condition code = 1 if error.

ROUTINES

HEXCK - Called to check for legal EBCDIC character and convert the character to hexidecimal.

INTERACTION

Called from keyin command routines to convert the id field of a keyin from EBCDIC to hexidecimal.

DESCRIPTION

This routine takes an EBCDIC field less than or equal to four characters and converts it to hex. Conversion proceeds via shifting of registers and calls to HEXCK where the actual conversion and validity checking is performed for one character per call. If the field length is greater than 4 characters, or if the conversion results in an invalid hexadecimal digit (i.e. other than 0-9 or A-F), CVSYSID exits with an error condition code set. Otherwise, the condition code is set to zero and the converted value is returned in R2.

ID

DEVCK - Check yyndd, translate to DCT index.

PURPOSE

To translate on EBCDIC yyndd string into the Device Control Table (DCT) index, provided all legality checks are satisfied. The yy legality checks are bypassed if yy is replaced with true zero.

USAGE

BAL, SR4 DEVCK

Input parameters: R2 = First 4 digits of device code (yynd).
R3 = Last digit of device code (d___).

Output parameters: R2 = DCT index
Condition code = 0 if legal device code; otherwise, condition code = 1.

ROUTINES

HEXCK - Called to convert device code (EBCDIC) to legal hexidecimal digits

INTERACTION

DEVCK is entered from KIMOUNT, KISRCH, KREQ, SKEYIN and DEVKEYIN to verify the device code field of the keyin.

DESCRIPTION

First the channel and device number (ndd) is converted to hexidecimal digits by two calls to HEXCK. If either of the above do not convert into legal hexidecimal digits or if the channel number is not in the range 0-7, control is returned to the calling program with an error condition set. Otherwise, the DCT index for the resulting device address is found by a sequential search of the DCT. The TYPMNE (type name) table is searched for a match on yy if yy is not true zero. The resulting index is checked against the type of device in DCT4 to verify that ndd is a yy device. If not, an error condition code is returned to the caller. Otherwise the condition code is set to zero and the device index is returned in R2.

ID

DPSCAN - Obtain SYSID and DCTX for user number.

PURPOSE

To obtain the system id and DCT index of the device code (if specified) for processing the delete or priority keyin commands.

USAGE

BAL, SR4 DPSCAN

Input parameters: R7 = Keyin parameter list
SR1 = Delimiter after delete or priority keyin command field.

Output parameters: SR2 = System id
SR2(byte 0) = DCTX of device code field (yyndd) of the symbiont device by which or for which the file was created.

ROUTINES

GKIFLD	-	Called to obtain the system id and device code fields from the keyin buffer.
CVSYSID	-	Called to convert the system id from EBCDIC to legal hexidecimal digits.
NXKICHR	-	Called to obtain the next character in the keyin buffer.
DEVCK	-	Called to verify and convert the device code to hexidecimal, and to return the device code in R2.

INTERACTION

DPSCAN is called from KIPRI and KIDEL to obtain the system id (and the DCTX of the device if specified).

DESCRIPTION

If the delimiter after the delete or priority command field is not a blank, control passes to KEYERR. Otherwise GKIFLD is called to obtain the system id, after which CVSYSID is called to verify and convert the id to hexidecimal digits. If either of the above routines return an error, control passes to KEYERR. Otherwise the id is put in SR2 for return, and the next non-blank character is obtained from the keyin buffer by calling the get next character routine (NXKICHR). If the non-blank character is a new line, control

is returned to the caller. If the non-blank character is not a new line or comma, control passes to KEYERR. Otherwise a file created by or for a particular device is to be deleted and the device code field is obtained by calling GKIFLD. If an error is returned or the field length is not five, control passes KEYERR. Otherwise DEVCK is called to verify and convert the device code to hexidecimal. If an error is returned, control passes to KEYERR. Otherwise the DCTX of the device specified is put in byte 0 of SR2 and control returned to the caller.

ID

GDTKIVAL - Get date/time keyin values

PURPOSE

To obtain and check the validity of the year, month, day, hour, or minute field keyins and to insert a leading EBCDIC zero if the field is a single digit.

USAGE

BAL, SR4 GDTKIVAL
Illegal field return
Normal return

Input parameters: R7 = Keyin parameter list
R0 = Maximum valid value of field to obtain

Output parameter: R2 = Two digit EBCDIC value of field obtained

ROUTINES

GKIFLD - Called to obtain the year, month, day, hour, and minute keyin fields.

INTERACTION

Called from date and time command routines to obtain relevant keyin values.

DESCRIPTION

First GKIFLD is called to obtain the relevant keyin field and if an error results from the following checks control is passed to the error return address, SR4 (+0):

1. Error return from GKIFLD.
2. Field length of keyin value greater than two.
3. Value of keyin field exceeds the legal value input in R0.

If the above checks are OK, the keyin value is put in R2 and control is passed to the normal return address, SR4 (+1). If the value of the field is only one character, a zero character is inserted for the first character in the field before control is returned.

ID

GKIFLD - Get field from keyin buffer.

PURPOSE

To move the next field from the keyin input buffer to the field buffer (KPLB) and initialize the field length (KFL).

USAGE

BAL, SR4 GKIFLD

Input parameters: R7 = ADDR or keyin parameter list.

Output parameters: CC1 = 1 if 12 < field length < 1.
(SR1) = delimiter character if field is terminated by a delimiter character.
KFL = field length.
KPLB = content of field, left justified with trailing blanks.

ROUTINES

NXKICHR - Called to obtain next character in keyin buffer.

INTERACTION

Called by the keyin main control routine (KEYIN1) to obtain the first field in the keyin buffer, and from the keyin command routines as additional fields are needed to process the keyin command.

DESCRIPTION

This routine gets the next keyin field from the keyin buffer and stores it (left justified) in KPLB, a three word buffer in the KEYIN parameter list. The field may be from 1 to 12 characters long, preceded by one or more blanks which are ignored. The field is delimited by a parenthesis (left or right), a comma, a period, a blank, a carriage return, or by having reached character position 81. Characters are obtained one at a time by calling the Get Next Character routine (NXKICHR) until a delimiter is found or the character count exceeds twelve. Leading blanks are surpressed by setting the blank flag (KFLAGS) to zero before the first character in the field is obtained and then to non-zero after the first non-blank and all remaining characters obtained from the field. Before exiting GKIFLD sets the condition code if the field length is either zero or greater than 12, otherwise the condition code is set to zero.

ID

HEXCK - Convert EBCDIC to Hex

PURPOSE

To convert a EBCDIC character to a legal hexadecimal digit.

USAGE

BAL, D4 HEXCK

Input parameters: R2 = Character to be converted

Output parameters: R2 = Converted character
Condition code = 0 if no error
Condition code = 1 if error

INTERACTION

Called from CVSYSID to convert and check system id and from DEVCK to convert and check a logical device address.

DESCRIPTION

Characters other than 0-9 or A-F cause CC1 to be set to 1. Otherwise, character is converted to a hexidecimal digit and CC1 is set to 0.

ID

KDS8CHM, KDS8FRTO - Move characters to message buffer.

PURPOSE

To move up to 8 characters to operator console message buffer.

USAGE

BAL, SR2 KDS8FRTO (or KDS8CHM)

Input parameters: R3 = Relative byte address in message buffer for first character to move.
R2 = Address of 8 character text to move to message buffer if entered at KDS8FRTO.
D3, D4 = Characters to move to message buffer if entered at KDS8CHM.
R6 = Address of message buffer.

INTERACTION

Called from all of the display option routines to put segments of the output message into the operator console buffer as they are constructed.

DESCRIPTION

This routine fetches a string of eight EBCDIC characters (terminated with a byte of zeros if less than eight) from R14 and R15 and stores it into the buffer whose word address is in R6, starting at the byte index in R3. An alternative entry to KDS8CHM is KDSFRTO which fetches the characters starting at the word whose address is in R2.

ID

KDSEYNDD - Construct device code message

PURPOSE

To construct the device code message segment "(NL) SYYNDD," in the output buffer and to return the priority segment "P(NL)" right justified in D2.

USAGE

BAL, SR2 KDSEYNDP

Input parameters: R6 = Buffer containing DCB and message area.
R2 = DCTX

Output parameters: "(NL)SYYNDD," segment of message is put in output buffer.
D2 = (priority)(NL) right justified.

DATA BASE

DCT1 - To obtain the physical device address (NDD) of the DCT index input in R2.

TYPNME - To obtain the device name (YY).

INTERACTION

Called from KDSNOP, KDSID and KDSTAP to obtain the parameters described under usage.

DESCRIPTION

This routine sets the eight EBCDIC characters "(NL) S y y n d d ," into the first two words of the buffer whose word address is in R6. The characters are derived from the DCT whose index is in R2. First the physical device address (NDD) is obtained from DCT1. Next the physical channel number (N) is converted to a channel letter (A-F). The channel and device number (DD) are then converted to EBCDIC followed by a comma. The device name (YY) is obtained from the TYPMNE table, and the segment "(NL)SYYNDD, is put in the message buffer. The priority field is then obtained from byte 1 of the key word, converted to EBCDIC, prefixed by a new line, and returned right justified in D2.

ID

KDSFSRCH

PURPOSE

To obtain first/next valid entry in the symbiont file directory.

USAGE

BAL, SR2 KDSFSRCH

Input parameters: R6 = Address of buffer containing DCB and message area.

Output parameters: D1 = First/next keyword in symbiont file directory.

DATA BASE

SYMFILE - Searched for next valid entry

ROUTINES

RMB - Called to release buffer containing DCB and message area if a keyword is not found.

KEYINR - Exit routine if a keyword not found.

INTERACTION

KDSFSRCH is called from KDSNOP and KDSID to obtain the first/next keyword in the symbiont file directory.

DESCRIPTION

If more entries are left to process, the directory index (in KDSDCBX+10) is incremented by one and the next item keyword is obtained. If the item word is empty the next entry is fetched until a valid entry is obtained or all entries are searched. If a keyword is found the DCTX is obtained from byte 0 and returned in R2. The item keyword is returned in D1. If a keyword is not found RMB is entered, to release the buffer containing the DCB and message area, after which control passes to KEYINR.

ID

HXKICHR - Get next input character

PURPOSE

To obtain the next character of key-in input, conditionally skipping blanks, and to flag delimiters.

USAGE

BAL, SR4 NXKICHR

Input parameters: R7 = ADDR of keyin parameter list

Output parameters: SR1 = Character
CC1 = 1 if character is a delimiter

INTERACTION

Called from get field from keyin buffer (GKIFLD), and process symbiont keyin (SKEYIN) routines to obtain the next character from the keyin input buffer.

DESCRIPTION

This routine gets the next character in the keyin input buffer (KBUF) and increments the current character position (KCCP). If character is blank and KFLAGS is zero, the next character if fetched, until a non-blank is found. If the character is a carriage return or is one of the characters in the delimiter list, ie ( ) , . / , or if the current character pointer is greater than 80, the delimiter flag (condition code) is set to 1. Otherwise the condition code is set to zero to indicate a valid character.

ID

OPLBTST - Operational Label Test

PURPOSE

To find operational label pointer, or to indicate illegality of label.

USAGE

BAL, SR4 OPLBTST

Input: R2 = Operational label

Output: If found:
(R1) = OPLBT1 pointer or zero if label = NO
CC1 = 0

If not found:
CC1 = 1

DATA BASE

OPLBT1 - Searched for match with input label in R2.

INTERACTION

Called from KISYST to check the validity and return the pointer to the label in R1.

DESCRIPTION

OPLBT1 is searched for match with the label specified in R2, using R1 as index. If search fails, R2 is checked for NO. If all compares fail, CC1 is set to 1. Otherwise CC1 is set to zero and the pointer to the label found is returned in R1.

UTS TECHNICAL MANUAL

ID

UBAT - Determine if user is batch, if so return ID

ENTER	EXIT
BAL, 11 UBAT	* 11 if online, or not found * 11 + 1 when found

INPUT

R4 = user number

OUTPUT

D2 = ID if found

DATA BASE

SMUIS - maximum number of users in system

UH:FLG - halfword table containing user flags

PLB:USR - byte table with entries corresponding to entries in PLH:S1D.

PLH:S1D - half word table which has the ID for batch user numbers.

DESCRIPTION

The value in R4 is compared against SMUIS and if greater exits *11. If not greater, the entry in UH:FLG indexed by the value in R4 is checked for having the batch bit set. If not set, exit is *11. If set, the value in R4 is compared against the value in PLB:USR as indexed by the partition number the PLH:S1D as indexed by the partition number the user is running in is returned in D2 at *11 + 1.

UTS TECHNICAL MANUAL

ID

FOTO40 - replace leading zeros with blanks.

ENTER	EXIT
BAL, 11 FOTO40	* 11

INPUT

D3 = WORD TO STRIP

OUTPUT

D3 = WORD with leading zeros (EBCDIC) replaced with blanks.

USES

D1, D3, D4

UTS TECHNICAL MANUAL

ID

4 BYTE

PURPOSE

Convert 2 bytes HEX to 4 bytes EBCDIC, replace leading zeros with blanks, put 4 byte EBCDIC in buffer

USAGE

BAL, 0 4BYTE

INPUT

D2 = HEX number, right-justified

R6 = buffer address

R3 = current character position in buffer.

ROUTINE

IDTOBCD - convert 2 byte HEX to 4 byte EBCDIC

FOTO40 - replace leading zeros with blanks
KDS8CHM - store into buffer

EXIT

*0
R6 = buffer address
R3 = current character position + 1

SECTION HA.03.14A

ID

2 BYTE
Same as 4 BYTE except hex byte is right justified, and 2 BYTE EBCDIC is stored in buffer.

UTS TECHNICAL MANUAL

ID

HEADER

PURPOSE

Output header line for input Q data

USAGE

BAL, 10 HEADER

INPUT: R6 = address of message buffer

uses: R3, R4, R5, R15

ROUTINES

TYPEIT - Called to output message to operator's console

EXIT

R6 = address of message buffer

R14 = address of message buffer

R 3 = 0 = Current character count

DESCRIPTION

A canned header is moved into the message buffer and TYPEIT is called to output the header line to the operator's console.

HEADER LINE

ID	P	ACCOUNT	SP	7T	9T	CO	TIME	OA	VALID PARTITIONS
4	1	8	3	3	3	3	5	2	16

where:

ID = SYSTEM ID FOR JOB

P = PRIORITY

ACCOUNT = ACCOUNT JOB WILL RUN UNDER

SP = THE NUMBER OF SPINDLES REQUESTED ON THE RESOURCE CARD

7T = THE NUMBER OF 7 TRACK DRIVES REQUESTED ON THE RESOURCE CARD

9T = THE NUMBER OF 9 TRACK DRIVES REQUESTED ON THE RESOURCE CARD

CO = THE AMOUNT OF CORE REQUESTED ON THE RESOURCE CARD

O = THE JOB IS ORDER DEPENDENT. INDICATED BY A1 set

A = THE JOB IS ACCOUNT DEPENDENT. INDICATED BY A1 set

VALID PARTITIONS = ALL PARTITIONS THE JOB CAN EXECUTE IN. INDICATED BY A1 SET.

NOTE: For O, A and VALID PARTITIONS, either a one or a period will be output for each indicator position.

UTS TECHNICAL MANUAL

ID

NORUN

PURPOSE

To get and store data (as described in HEADER) about a specific input Q entry

USAGE

BAL, 8 NORUN

INPUT: R 1 = LINK INDEX
R 2 = PRIORITY
R 6 = Address of message buffer
R3 = Current character position in the buffer

DATA BASE

BH:SID - SYSTEM ID
BD:ACCT - account
BW:RES - resource requirements
BH:TIME - specified (or default) time
BW:SDA - disc address checked for order or account dependent

indexed by link

ROUTINES

4 BYTE - convert 2 byte HEX to 4 byte EBCDIC and store in buffer
2 BYTE - convert 1 byte HEX to 2 byte EBCDIC and store in buffer
PARTS - store partition indicators in message buffer

DESCRIPTION

Information from the tables listed under DATABASE and retrieved into the message buffer spaced as indicated in "HEADER". Parts is used to access BH:PART and store a 1 or a period in the message buffer to indicate which partitions the job can run in.

UTS TECHNICAL MANUAL

ID

PARTS optional entry point PARTS 11

PURPOSE

Determine which partitions an input Q entry can execute in and place this information in the message buffer.

USAGE

BAL, 15 PARTS

INPUT: R 1 = LINK index
R 3 = current character position in the message buffer
R 6 = address of message buffer

DATA BASE

BH:PART - half word partition table indicator

DESCRIPTION

The entry for this user is retrived from BH:PART by using the LINK index. Then each bit in the halfword is converted to an EBCDIC PERIOD or ONE, and is stored in the message buffer.

ID

UTS CAL Processors

PURPOSE

The purpose of these routines is to process new UTS specific CAL's.

OVERVIEW

UTS is based on version E00 of BPM and offers all the BPM CAL's except those relating to checkpoint/restart and real-time. Most of the code that processes these CAL's has been taken intact from BPM. In addition to these BPM services, UTS offers a set of new services. Memory management CAL's are described in Section G. The new CAL's, the module containing the routine and the section containing the description are listed below.

Name		Module	Section
Adjust DCB CAL	Open prime, merge assignments	OPN	
T:AMRDWT	Read/Write Assign-Merge	UCAL	IA
T:CHTBL	M:CT - Change COC translate tables	UCAL	IA
T:GHOST	Ghost Abort Message	UCAL	IA
T:INITJOB	Initiate Ghost Job	UCAL	IA
T:WAIT	M:WAIT - Sleep	UCAL	IA
T:WAKEUP	Wakeup	UCAL	IA
T:SAVEGET	Save CAL, Get CAL	UCAL	IA
T:SELFDESTRUCT	Monitor Overlay Exit	UCAL	IA
T:STPMT	M:PC - Set Prompt	UCAL	IA
T:SYS	M:SYS - Master Mode	RDERLOG	IA
T:SYSLOAD	M:DISPLAY - System Load	UCAL	IA
T:WTERLOG	Write Error Log	RDERLOG	IA
T:ASSOCIATE	Associate DELTA/library	UCAL	IA
T:DISASSOCIATE	Disassociate DELTA/library	UCAL	IA

ID

T:AMRDWT - Assign/Merge Read/Write CAL Processor

PURPOSE

The purpose of this routine is to read or write the assign-merge record for the current user. This record is used to carry DCB assignment information from job step to job step.

USAGE

B T:AMRDWT - from IORT
R6 - contains the virutal address of a closed DCB through which the A/M record is accessed.
R8 - contains read (X'2D') or write (X'2E') A/M function code.

INPUT

J:AMR - on the first write of the A/M record a granule is obtained and its address is stored in J:AMR.

INTERACTION

MSR01EXIT - error exit from I/O CAL's. R10 contains the error code.
GBG - Get background granule entry in BUFGRAN.
T:IACU - Interrogate users access entry in MM. Returns the users access to the specified condition codes.
QUEUE - Queue I/O through a DCB entry to IOQ.
PULLALLEXIT - entry in IORT which pulls seven words from TSTACK into R5-R11 and exits on R11.
T:ABORTM - exit in STEP if error on read or write of A/M record. R14 contains the error code X'A9'.
RBG - Release background granule entry in BUFGRAN.

SUBROUTINES

BUFCHK - check if size specified in the DCB is less than 513 words. If not, error exit with code X'4A'.
FLGCHK - checks if the current processor (doing the A/M Operation) has special JIT access or is master mode. If either condition is true, exit to link + 1. If neither, error exit to the link address.

ERRORS

All error conditions result in an exit to MSR01EXIT (ERR1) with a code in R10. The codes are:

X'06' - read operation and J:AMR = 0, i.e., the A/M record has not yet been created.
X'56' - no granule available for initial creation of A/M record.
X'4A' - size specified in DCB greater than 512 or a read by a non-special processor into a non-data page (access not equal to zero).
X'14' - write request by a non-special processor (special JIT access not set in UH:FLG and not master mode.

DESCRIPTION

The maximum size of an A/M record is 512 words. If the caller requested a read or write of greater than 512 words, he is given an error return with code X'4A'. If an A/M record does not yet exist for this user (J:AMR = 0) and the request is a read, the caller is given an error code X'06'. If the operation is a write (i.e., the first write), a disc granule is acquired via GBG in BUFGRAN. If there are no granules available, return is with an error code of X'56'. If one is available its address is stored in J:AMR and initialization of the DCB proceeds. If an A/M record did exist (J:AMR $\neq$ 0), control transfers directly to DCB initialization (SET10). If the operation is a read and the page specified is not a data page, control goes to FLAGCHK to check for master mode or special JIT access. If either of those conditions hold, the read is permitted into the non-data page. Otherwise, an error code of X'4A' is given. If the operation is a write, check for special JIT access or master mode and disallow the request if neither condition holds. In that case the error code is X'14'. Finally the function count in the DCB is incremented (decremented by IOQ at end action) and the I/O request is enqueued via QUEUE in IOQ. Exit is to PULLALLEXIT in IORT which pulls 7 words into R5-R11 and exits *R11. R11 contains the I/O CAL exit address IOSPRTN in CALPROC which results in control returning to CAL plus one. If however, there was an error when reading or writing the A/M record, exit is to T:ABORTM in STEP after J:AMR is zeroed and the granule released.

ID

T:CHTBL - Change COC translate or break set table (M:CT procedure)

PURPOSE

The purpose of this routine is to permit the user to declare to the COC handler the type of terminal he has (i.e., which translate table to use) or which break or activation character set he wishes in use at his terminal.

USAGE

B T:CHTBL - from ALTCP as a result of a CAL1,8 instruction with an FPT code of X'06'.
See UTS/TS Reference Manual, Chapter 9 for description of the FPT.
R6 - contains first word of FPT.

OUTPUT

MODE2 - COC line control flags
COCTERM - COC terminal type index

DESCRIPTION

If the caller was not an on-line user (Bit 0 of word 0 of JIT reset), exit is to CAL plus one with CC1 set. The presence bit for change break set (bit 8) in word 0 of the FPT is checked. If set, bits 14-15 are stored in MODE2 and exit is to CAL plus one with CC1 reset. If not set, bits 24-31 are checked for legality as a terminal type. If not legal, exit with CC1 set. If legal, store in COCTERM and exit with CC1 reset.

ID

T:GHOST

PURPOSE

The purpose of this routine is to output an error message on the operator's console when a ghost job errors or aborts.

USAGE

BAL, 11 TGHOST - from STEP when a ghost job is being aborted by the monitor.

INPUT

JB:PNR - GJOB index in JIT
J:ACCN - user account in JIT
J:UNAME - user name in JIT

INTERACTION

MSROCTY - Monitor type to OC routine

SUBROUTINES

DELETBLNKS - Delete trailing blanks routine. Byte offset plus one of the last byte in the buffer is in R3, word address of the buffer is in R15. R3 can be byte address plus one of the last byte if R15 contains 0. Exit is with R3 pointing at the last non-blank character plus one in the buffer. If the field is all blank, the routine returns with CC1 set.

MOVER - (MOVER1). Moves 255 or fewer bytes from source to destination buffer as specified in registers 1, 2 and 3 (count, byte address of source, byte address of destination). R3 points to next available byte upon exit. MOVER1 entry uses (R14) as a byte displacement for the source and R15 as the byte displacement for the destination buffers.

MVDATIME - This routine moves the current date and time into a buffer pointed to by the byte address in R3.

MYNAMACT - This routine moves the name and account from the current user's JIT into the buffer pointed to by the byte address in R3. Trailing blanks are deleted and a comma is inserted between the name and account.

DESCRIPTION

This routine simply formats the message

"GJOB name, account ERR"

into a buffer in JIT, J:CCBUF, and outputs it to the operator's console via MSROCTY. The system ID and two tabs are prefixed to the message by MSROCTY.

ID

T:INITJOB - Initiate ghost job CAL processor.

PURPOSE

The purpose of this routine is to initiate the specified Load module as a ghost job.

USAGE

B T:INITJOB - from ALTCP
R7 - contains the address of word 1 of the FPT. Words 1 and 2 of the FPT contains the TEXTC EBCDIC name of the load module to be initiated.

INTERACTIONS

ADD1 - entry to the execution scheduler SSS. R4 contains the user number. This routine allocates a JIT page and initiates execution of a user.

SUBROUTINES

T:GJOBSTRT - This subroutine verifies that the name specified is a ghost known to the system. If the ghost is asleep, this routine reports a wake up event on it. Otherwise it transfers to T:ADDGHOST with the JIT disc address in R15.

R0, R1 - contain the TEXTC name of the ghost to be initiated.
R10 - link register

INPUT

SB:GJOBUN - ghost job user number table.
UB:US - user state table
S:GJOBTBL - names of currently active or reserved ghost job names.

INTERACTION

T:RUE - Report user event entry to the execution is scheduler.

DESCRIPTION

T:INITJOB loads R0 and R1 with the ghost name from the FPT and transfers to T:GJOBSTRT.

T:GJOBSTRT first searches S:GJOBTBL for the name. If found, SB:GJOBUN is checked for an active copy. If active, UB:US is checked for the sleep state (SW). One of three responses occurs:

1) The name is not in the table. If a zero entry is, and a user slot is available, the name is put in S:GJOBTBL, the user slot is put in SB:GJOBUN, the user is started via ADD1 and condition codes are reset. If either of these conditions fails, CC1 is set.

2) The name is found and the user number is zero. If a user slot is available, the user slot is put in SB:GJOBUN, the user is started via ADD1, and condition codes are reset. If not, CC1 is set.

3) The name is found and the user number is non-zero. If the user state is asleep (SW), the wake-up event (E:WU) is reported to SSS and CC2 is set.
If the user state is not SW, CC1 is set.

ID

T:WAIT - M:WAIT CAL Processor

PURPOSE

The T:WAIT CAL Processor saves the requested 'sleep' interval and reports the sleep event, E:SL, to the execution scheduler.

USAGE

B T:WAIT - called by ALTCP as a result of a CAL1,8 instruction with an FPT code of X'0F'.

R6 Contains the first word (word 0) of the FPT, bits 15-31 of which are the interval count field.

INPUT

S:CUN - Current User Number

OUTPUT

U:MISC - Number of .02 minute intervals before a wake up event will be issued.

INTERACTIONS

T:REG - Report event and give up control entry to the execution scheduler, SSS.

DESCRIPTION

This routine stores the specified elapsed or "wall clock" time interval in the resident user table U:MISC. It rounds the interval up by one if the clock 3 driven counter, 1SEC, has less than half a timer interval (.02 min, or 1.2 sec) to go before being incremented again. It then reports the "sleep" event, E:SL, on the current user to execution scheduler, SSS. While the user is in the sleep state he is a high priority candidate for outswap. Exit is to CAL plus one with CC1 reset.

The following event's remove a user from the sleep state:

E:CBK - break - from COC
E:CEC - control Y - from COC
E:WU - wake up - from T:WAKEUP or T:GJOBSTRT in UCAL
E:ERR - error - from KEYIN (E, id keyin)
E:OFF (E:ABRT) - off or abort from KEYIN and COC (X, id; ZAP keyins or phone hang up)

ID

T:WAKEUP - wake sleeping users

PURPOSE

The purpose of this routine is to decrement and test the timer interval count for all sleeping users. If the interval has elapsed, the wake up event (E:WU) is posted to the execution scheduler SSS. The routine is invoked by the clock 3 interrupt processor every 1.2 seconds (.02 minutes).

USAGE

BAL, 1 T:WAKEUP - from CLOCKI the clock 3 interrupt processor.

INPUT

U:MISC - count of 1.2 sec intervals until wake up (if the user is in the sleep state).
UB:FL - Forward link table. Links all users in a given state, terminates with a value of zero.
SB:HQ - Head of queues table. Indexed by state number, contains user number of first user in the state (or queue).

OUTPUT

U:MISC - count is decremented by 1 each entry into T:WAKEUP.

INTERACTIONS

T:RUE - Report user event entry to the execution scheduler.

DESCRIPTION

The head of the queue of sleeping users is checked. If zero, there is no one sleeping. If non-zero, the count of intervals remaining until wake up is decremented. If it reaches zero, the wake up event E:WU is reported on that user. The process is repeated for each user in the sleep state.

ID

T:SAVEGET - Process SAVE and GET CAL's.

PURPOSE

The purpose of this routine is to provide an interface between user programs, TEL, or CCI and critical resident user tables. This interface is used in the limited checkpoint/restart capability called SAVE/GET in UTS. T:SAVE fetches information from user tables and stores it in the user JIT. T:GET fetches the information from the user JIT and sets up the user tables with it.

USAGE

B T:SAVEGET - from ALTCP as a result of a CAL1,4 with an FPT code of X'02' or X'03'.
R8 - contains FPT code.

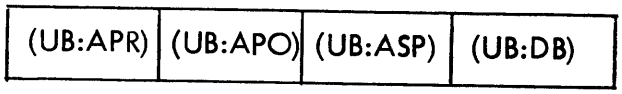
INPUT - T:SAVE

UB:APR - Associated shared processor root
UB:APO - Associated shared processor overlay
UB:ASP - Associated special shared processor
UB:DB - Associated shared debugger
UH:FLG - User's status flags

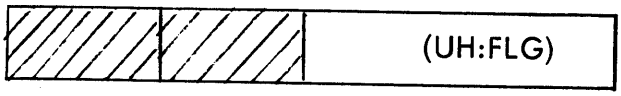
OUTPUT - T:SAVE

J:CPROCS - Bits 0-31. Word in JIT which contains shared processor numbers of saved environment.
J:CFLAGS - Bits 16-31. Current users status flags.

J:CPROCS



J:CFLAGS



PB:REP Processor user count

INPUT - T:GET

PB:PSZ - number of pages of shared processor procedure.
PB:PVA - page number of first virtual page of shared processor procedure.
J:CFLAGS - User's status flags at time of SAVE CAL.
J:CPROCS - User's shared processors at time of SAVE CAL.

OUTPUT - T:GET

UH:FLG - The DELTA associated and DELTA in control bits from the users flags at the time of the SAVE CAL are selectively stored in the user's current flags (at the time of the GET command).

PB:REP
UB:APR
UB:APO
UB:ASP
UB:DB

} See Above

INTERACTIONS - T:GET

T:SNAC - Set n pages of 01 access for shared processors (MM)

DESCRIPTION

The SAVE CAL is used to record in the current user's JIT, the shared processor and debugger numbers associated with him. TEL issues this CAL and writes out the JIT and the user's context data, and procedure when the user issues a SAVE command. When a GET command is issued, TEL reads in the named file, extracts information from J:CPROCS, J:CFLAGS and TSTACK in the JIT in the file and stores it in the current user's JIT. Then TEL issues a GET CAL which takes the information from J:CPROCS, stores it in the appropriate user tables (see OUTPUT - T:GET above), and initializes the access registers for APR and APO if they exist.

ID

T:SELFDESTRUCT

PURPOSE - DESCRIPTION

The purpose of this subroutine is to serve as an exit point for monitor overlays which should be disassociated from the users after each call. The current user's entry in the user overlay table (UB:OV) is zeroed and the use count for the overlay (PB:UC and PB:REP) are decremented by one. Exit is on R11.

ID

T:STPMT - Set prompt character.

PURPOSE

To allow the user via the M:PC procedure to establish a prompt character to be issued on each COC read request.

USAGE

B T:STPMT - from CALPROC as a result of a CAL1, 1 instruction with an FPT code of X'2C'.
R6 - contains word 0 of FPT including the EBCDIC prompt character in bits 24-31.

OUTPUT

JB:PROMPT - 1 byte in JIT.

DESCRIPTION

The EBCDIC prompt character is stored in JIT (JB:PROMPT). The ANSI equivalent is picked up from the appropriate translate table and stored in the COC table PROMPT by the COC routines. A prompt specification of binary zero resets the COC to no prompt mode.

ID

T:SYS

PURPOSE

To allow a sufficiently priviledged user to operate in master mode and to identify for the user the DCB and non-DCB entry points to IOQ.

USAGE

B T:SYS - from ALTCP as a result of a CAL1,6 with an FPT code of X'08'.

DESCRIPTION

If the user has a privilege level of X'C0' or higher, the master mode bit in his PSD in TSTACK is reset and the addresses of QUEUE, QUEUE1 and NEWQ are stored in his registers 8, 9, 10 in TSTACK. It should be noted that all versions of UTS up to and including B01 do not provide no-block or no-wait I/O. In addition end-action addresses, if specified, must be in the resident monitor area of memory which is mapped one to one virtual to physical. The reason for this is that end action occurs at the I/O interrupt level which is asynchronous to user processes and thus the contents of the map registers is indeterminate.

ID

T:SYSLOAD - Display system load (M:DISPLAY)

PURPOSE - DESCRIPTION

This routine returns execution time multiplication factor (ETMF), 90% response point and the number of active users in the caller's registers 5, 6 and 7.

USAGE

B T:SYSLOAD - from ALTCP as a result of a CAL1,8 instruction with an FPT code of of X'13'.

INPUT

C:ETM	Current value of ETMF.
C:RT90	Current 90% response point (computed by PM performance measurement routine).
S:CUIS	Current users in system.

ID

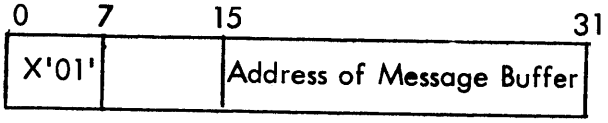
T:WTERLOG - write error log CAL processor (in RDERLOG)

PURPOSE

To permit users access to the monitor ERRLOG routine.

USAGE

B T:WTERLOG - from ALTCP as a result of a CAL1,6 with an FPT code of X'01'.
R6 - contains word 0 of the FPT



DESCRIPTION

This routine basically sets up a BAL to ERRLOG specifying the user's buffer as the message buffer. The buffer is checked via T:IACU in MM to insure that the page belongs to the user. If it doesn't or if the buffer indicates a message word count (in word 0 byte 1) greater than 10, return is with CC2 set. Otherwise return is to CAL+1 with CC1 reset.

ID

Associate and Disassociate CALs.

PURPOSE

To provide a mechanism for a user or shared processor to control the association and disassociation of shared public libraries or a debugger with his program.

OVERVIEW

The associate CAL logic, entered at T:ASSOCIATE in UCAL, obtains the name specified in the FPT and verifies it is a library or debugger, then associates the request by setting the appropriate user and processor tables and using the T:REG entry to SSS to report the associate processor event.

The disassociate CAL logic, entered at T:DISASSOCIATE in UCAL, resets the appropriate user and processor tables thus disassociating the user from the specified library or debugger.

USAGE

The FPT and associated tables of the associate and disassociate CALs are described in the UTS/BP Reference Manual, 90-17-64. Upon entry at T:ASSOCIATE or T:DISASSOCIATE, ALTCP provides:

R6 = 1st word of the PLIST
R7 = Pointer to the PLIST+1, the requested name

INPUT and OUTPUT

Tables used as described in Section V of the UTS Technical Manual. There is no input or output unique to these routines.

INTERACTION

ENTRY	MODULE	PURPOSE
T:PAC	MM	Set access on newly associated or disassociated library or debugger.
T:GNVNPI	MM	Get virtual page for debugger's initial data to be read into during association.

ENTRY	MODULE	PURPOSE
T:REG	SSS	Report E:AP to request processor via swapper mechanism.
CC0RST CC1SET CC2SET CC3SET	UCAL	Exit through TRAPEXIT with the appropriate condition codes set.

DATA BASES

Described in Section V of the Technical Manual.

SUBROUTINES

GETNAME Picks up the specified name from the FPT and attempts to match it with an entry in the P:NAME table. If unsuccessful, it returns control to the user via CC1SET.

Access:	BAL, R0 GETNAME
In:	R7 = Pointer to name in FPT
Out:	R4 = User number
	R5 = Processor number
	R8 and R9 = Processor name
	R15 = User's flags from UH:FLG

GETTYPE Given the processor number, returns the type, library or debugger. If neither, returns to the user via CC1SET.

Access:	BAL, R0 GETTYPE
In:	R5 = Processor number
Out:	R2 = 1 if Library
	2 if Debugger

ERRORS

Condition code settings on error conditions are described in the UTS/BP Reference Manual 90-17-64.

RESTRICTIONS

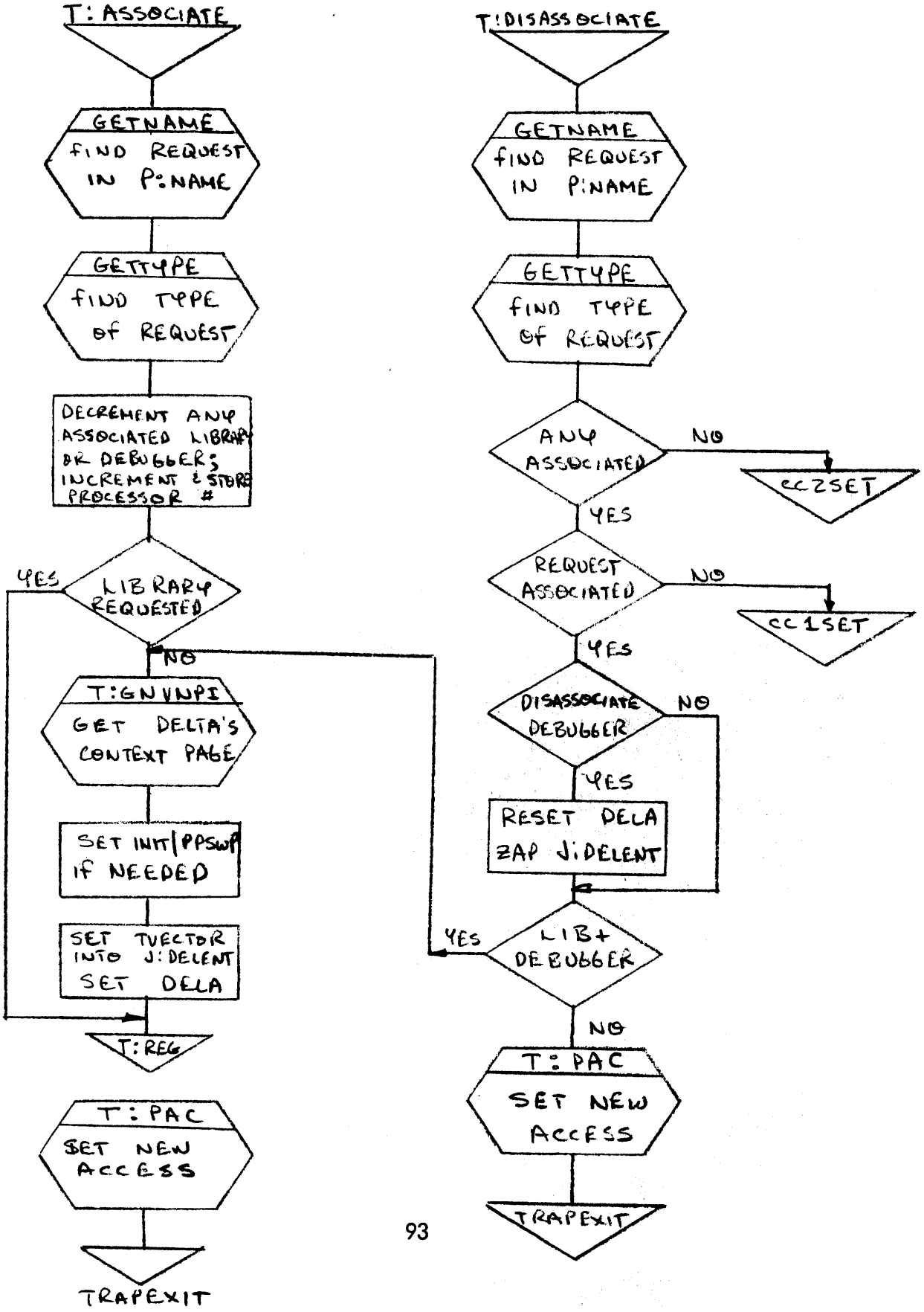
The debugger associated must be DELTA. If another debugger is written and associated, great care must be taken to insure it responds properly to the special-

casing of DELTA which is found throughout the monitor.

DESCRIPTION

T:ASSOCIATE is the entry point for the associate CAL logic which first obtains the processor number and type of the request via the subroutines GETNAME and GETTYPE. If a library or debugger is already associated, its use count in PB:UC is decremented and an abnormal condition indicated. The processor number of the request is set into UB:ASP or UB:DB, depending on type, and its use count incremented. If the request is a library, control is transferred directly to the logic which forces a swap by resetting the ready-to-run bit in UH:FLG and reports an associate processor event via T:REG. Otherwise, if the request is a debugger and a library is already associated, the library remains in core; DELTA will automatically be associated when required (trap condition, break key, etc). The DELTA virtual context page is obtained at J:EUP + 1 if not already in the user's map, and PPSWAP and INIT are set into UH:FLG to indicate to the swapper to bring in DELTA's initial data. The address of the vector supplied in the FPT is set into J:DELENT (equated to J:INTENT) and DELA set into UH:FLG. Then control is given to the logic which forces the swap. Upon return from T:REG, new access is set via T:PAC to reflect the association, and control returned to the user with appropriate condition code settings.

The logic entered at T:DISASSOCIATE first gets the requested processor number and type in exactly the same manner as T:ASSOCIATE. If no library or debugger is associated, CC2 is set and control is returned to the user via CC2SET. If the associated library or debugger is not the one requested, CC1 is set and control returned. If the request is to disassociate a debugger, DELA is reset in UH:FLG, and J:DELENT zeroed. In either case, the use count PB:UC and PB:REP of the request are decremented, then a test made for the following condition: Both a library and debugger associated, the dis-association of the library request. In this case, the library was in control, and DELTA's use count must be incremented to insure its association when required. In any case, the appropriate entry in either UB:ASP or UB:DB is zeroed, then control is passed to the T:ASSOCIATE logic if condition mentioned above is fulfilled, to properly set DELTA's processor tables and the user's flags. Otherwise, T:PAC is called to reset the access to reflect the disassociation, and control is returned to the user.


T:ASSOCIATE
GETNAME
FIND REQUEST IN P:NAME
GETTYPE
FIND TYPE OF REQUEST
DECREMENT ANY ASSOCIATED LIBRARY OR DEBUGGER; INCREMENT & STORE PROCESSOR #
YES
LIBRARY REQUESTED
NO
T:GNVNPI
GET DELTA'S CONTEXT PAGE
SET INIT|PPSWP IF NEEDED
SET TVECTOR INTO J:DELENT
SET DELA
T:REG
T:PAC
SET NEW ACCESS
TRAPEXIT
T:DISASSOCIATE
GETNAME
FIND REQUEST IN P:NAME
GETTYPE
FIND TYPE OF REQUEST
ANY ASSOCIATED
NO
CC2SET
YES
REQUEST ASSOCIATED
NO
CC1SET
YES
DISASSOCIATE DEBUGGER
NO
YES
RESET DELA
ZAP J:DELENT
YES
LIB+ DEBUGGER
NO
T:PAC
SET NEW ACCESS
TRAPEXIT

ID

PERFORMANCE - System Performance Measurement

PURPOSE

To provide performance measurement facilities in the system.

USAGE

The CONTROL processor can be called by a user to display system tables and values computed by the performance measurement (PM) package. Details regarding usage of CONTROL are available in the System Management Guide (901674) Chapter 5.

OVERVIEW and DESCRIPTION

The performance measurement package consists of two modules – PM and PMDAT. The PM module contains a number of subroutines which are BALed to by various monitor routines in the process of evaluating system performance. PMDAT contains the tables and data bases used to perform the above function.

Reference is made to the System Management Guide (901674) for key concepts in performance monitoring especially interaction time, response time, task turnaround time. Also of inportance is the concept of distribution tables and buckets. PM contains 3 'filters' which are 14 half words each, with values that progress linearly or exponentially. The linear table contains linearly increasing values from 5 to 65 in increments of 5. The exponential table has values from 1 to 10,000 in exponential increments i.e. 1, 2, 5, 10, 20, etc. The special linear has values from 1 to 13 in unit increments. A value to be distributed (e.g. inswap time) is first scaled down and then compared against the filter specified in its distribution table. Each value in the filter is compared to the value being distributed. The count is bumped in bucket 1 if the scaled data is less than the first filter table value. Count is bumped in bucket 'n', if scaled data is less than n^{th} filter value but greater than the (n-1) value.

Analysis of system performance boils down to examination of four critical parameters:

a) Response time distribution curve with 90% of the users interactive

b) Execution Time Multiplication Factor (ETMF) - A measure of the system's response to compute bound tasks. It is the amount of time that a program takes to execute, assuming that it would take one unit of time if running alone. Details of how this is evaluated are in routine T:SYSTEMLOAD.

c) Percent of unoverlapped I/O time - Measures the system's ability to schedule so that an executable user is always in core.

d) Number of users in core - Is related to the system's ability to overlap computing with swapping. System throughput and response are degraded if only enough core is available for a single user.

DATA BASES

Counter names and usage:

Most of this information resides in PMDAT and is used, but never modified by CONTROL.

Name	Description	Use
C:SIT	Sum of all interaction times	READREQ routine in PM computer interaction time and keeps a running total
C:CIT	Count of number of interactions since system startup	Incremented by 1 during READREQ routine in P7
C:CITI	Count of interactive interactions ones with CPU time < QMIN	Incremented by 1 during READREQ routine in PM
C:SRT	Sum of system response time	READREQ routine in PM makes the calculation and keeps a running total
C:TINC	Total number of 2 ms tics remaining (initial value = 30)	CURNTIM routine in PM uses it to compute current time of day. Incremented by clock 3 pulse interrupt.
C:TIC	Current block 3 counter – 2 ms tics since system startup	Accumulated by clock 3 counter zero interrupt. CURNTIM routine in PM, uses this value.
C:IDLE	Total time in idle	This is incremented by clock 4 while in the 'idle loop' in SSS
C:IDLES	Time with swap in progress and users in COM or BAT queues	Incremented by clock 4 while in the 'idle loop' in SSS

Name	Description	Use
C:IDLEW	Time with file or tape I/O and no simultaneous swap or user program execution. e.g. core is filled with I/O bound users.	Incremented by clock 4 while in "idle loop" in SSS
C:IDLESW	Time with file on tape I/O and concurrent swap I/O but no user program in execution although users appear in COM or BAT.	Incremented by clock 4 while in "idle loop" in SSS
C:STT	Total turn around time-system total and individual processor totals	READREQ routine in PM keeps a running total

Name	Description	Use
C:ST	Total think and type time – system and individual processor totals	READREQ routine in PM keeps a running total
C:SCO	Total CPU times for on-line user; system and individual processor totals	PMSC routine in PM does the summing
C:SCB	Total CPU time for batch user; system and individual processor totals	PMSC routine in PM does the summing
C:SCI	Total CPU time for interactive interactions	READREQ in PM increments , this by 1 for compute times < QMIN
C:SC	Total CPU interaction time,CPU time used by each processor also	READREQ sets up the totals
C:CTW	Count of number of terminal writes	Incremented by COCWR routine in COCC
C:CO	Count of number of characters output to all terminals	Incremented by COCOP routine (output interrupt handling routine) in COCC
C:CI	Count of number of characters input by all terminals	Incremented by COCIP (input interrupt handling) routine in COCC
C:COS	Count of number of out swaps	Incremented by OSWAPMEAS routine in PM
C:NSP	Count of number of times sufficient core was not found, using all available means	Incremented in PROCSOUT routine in SSS
C:NS	Count number of times no swap possible due to insufficient core (after swapping max number of users and unused processors)	Incremented in PROCSOUT routine in SSS

Name	Description	Use
C:RT90	90% point of response time distribution	Calculated once per minute by T:SYSTEMLOAD routine in PM and averaged with the last value.
C:RECYCLE	Number of times a recycle was scheduled due to change in events. Recycle increases overhead time. Recycle implies more than one pass through queues for either a) execution, b) inswap, c) outswap	Incremented in various SSS routines where checks are made to see if right action is being taken.
C:RTRW	Total number of I/O actions to RAD and tape	Posted in the FRONTEND routine of IOQ.
C:SCOB	Sum of on-line and batch compute times	Calculated by T:SYSTEMLOAD in PM.
C:SRT	System response time total	READREQ routine in PM keeps a running total.
C:CAL	Total number of CALs issued	CALPROC increments this.
C:CSC	Count of number of symbiont and cooperative reads and writes	COOP, INSYM and OUTSYM routines maintain running totals.
C:ETM	Execution Time Multiplication factor	Calculated once every minute by T:SYSTEMLOAD in PM.
C:MSO	Total CPU time for monitor services on-line	PMSC routine in PM keeps a running total.
C:MSB	Total CPU time for batch monitor services	PMSC routine in PM.
C:NOPROC	Number of times processor not in core	PRCAV routine in SSS.
C:NOQ	Number of times no queue entry available	GETQ routine in IOQ bumps this counter value each time a queue is not available.
C:PROCREQ	Count of number of times processor required	PRCAV routine in SSS increments

Distribution Table Description and Names

Each distribution table contains either:

a) A single distribution for all of the system

or

b) A distribution for all of the system, a distribution for each of the measured shared processors, and one for user programs or non-shared processors.

The order of distribution in a table is determined by the entry number of the processor in P:NAME. The first distribution is for overall system response. The last distribution, in a table with more than one distribution is for user programs. The processor distributions fit in between, based on processor index. For example:

P:NAME	Index	Description
	0	Distribution for all on-line
5BASIC	1	Distribution for on-line Basic
5DELTA	2	Distribution for on-line Delta
4EDIT	3	Distribution for on-line Edit
7FORTRAN	4	Distribution for on-line Fortran
7METASYM	5	Distribution for on-line Metasymbol

The maximum number of processors that can be currently measured is 10. In addition, before a processor can be measured, its processor number should fall between limits BGNPMPRC+1 and ENDPMPRC.

The user index is ENDPMPRC+1, so users fit into the last slot.

Distribution Table Format CH:XXX

The table is 16 halfwords long.

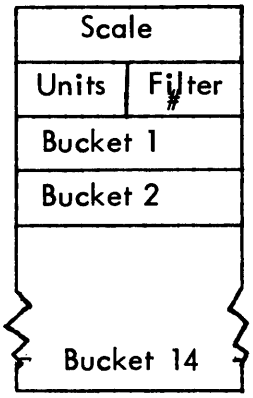
Data: 14 halfwords and 14 halfword buckets

Scale: First halfword. Specified the divisor for input data

Units: Byte 0 of second halfword
1 = seconds
2 = milliseconds
3 = count

Filter #: Byte 1 of second halfword. Specifies which filter table to use.
1 = log scale 1, 2, 5, 10, 20, 50, . . . 1K, 2K, 5K, 10K
2 = linear scale 5, 10, 15, . . . 65
3 = special linear scale 1, 2, 3, 4. . . 13

Table Name
(16 halfwords long)



Subscript 'p' indicates that a distribution is included for each shared processor, as well as for the system.

Distribution Name	Scale, Scale Factor	Unit	Distribution Significance
CH:DC_p	Log, 1	milliseconds	Compute time per interaction (by processor)
CH:DIT	Log, 500	seconds	Time per interaction
CH:DI1	Log, 1	seconds	Processor and JIT inswap time
CH:DI2	Log, 1	seconds	Inswap time for rest of the user
CH:DI3	Log, 1	seconds	Time between end of JIT inswap to end of user inswap
CH:DLI_p	Linear	characters	Number of characters per input transaction (by processor)
CH:DLO_p	Linear	characters	Number of characters per output transaction (by processor)
CH:DOS	Special linear, 1	users	Numbers of users swapped out per out swap
CH:DOT	Log, 1	seconds	Outswap time distribution
CH:DRT	Log, 1	milliseconds	Response time distribution
CH:DT_p	Log, 500	seconds	Think and type time per interaction (by processor)
CH:DTT_p	Log, 500	seconds	Task turnaround per interaction (in processor)

SUBROUTINES			Section IB.01
Name	Purpose	BAL from	
ACTIVATE	Evaluate numerator for ETMF calculation Response time calculation	SSS	IB.01.01
COMPTIM	Evaluate compute time used	READREQ in PM	IB.01.02
GETINDX	Fetch processor # or user index (the displacement into the distribution table)	Various PM routines	IB.01.03
OSWAP$MEAS ISWAP$MEAS1 ISWAP$MEAS2	Compute inswap times for Processor and JIT and for rest of user. Compute outswap time for user.	SSS	IB.01.04
PMSC	Update appropriate counters for (batch and on-line) compute times and monitor services. Evaluate denominator for ETMF calculation.	ACCT	IB.01.05
RDMSGSIZ	Distribute number of characters per input transaction.	COC	IB.01.06
READREQ	Evaluate: a) System response time b) Task turnaround time c) Think and type time d) Compute time per interaction Set up J:TIC, J:T for use by other routines e) Distribute the computed values	COCRD	IB.01.07
SWAPRCRD	Get # of users swapped out at last swap	SSS	IB.01.08

Name	Purpose	BAL from	Section
T:SYSTEMLOAD	To evaluate system load: a) Calculate the 90% point of response-time distribution b) Calculate ETMF c) Adjust batch quantum in steps of 12 1/2% to fall within minimum and maximum values and meet bias requirements	CLOCK4	IB.01.09
T:DSTRB	Distribute performance statistics. Increment appropriate system and user distribution buckets. Other routines in PM branch and link to this routine to distribute values they have evaluated and bump appropriate buckets.	Various PM routimes	IB.01.10
WTMSGSIZ	Distribute output message length. Set turnaround time if first write.	COC	IB.01.11

ID

ACTIVATE

PURPOSE

a) To evaluate the numerator for ETMF calculations.
b) To evaluate response time for interactive users.

USAGE

SSS enters ACTIVATE just before a user is about to execute. The count of high priority users ready to run is decremented and a BAL to ACTIVATE is executed to evaluate response time, if user is interactive.

Return is via R5.

INPUT

a) User number in R4
b) User state in R3
c) Time at end of message saved in EOMTIME by COC.
d) Time at which EXU state entered, recorded by ETMM in U:MISC.

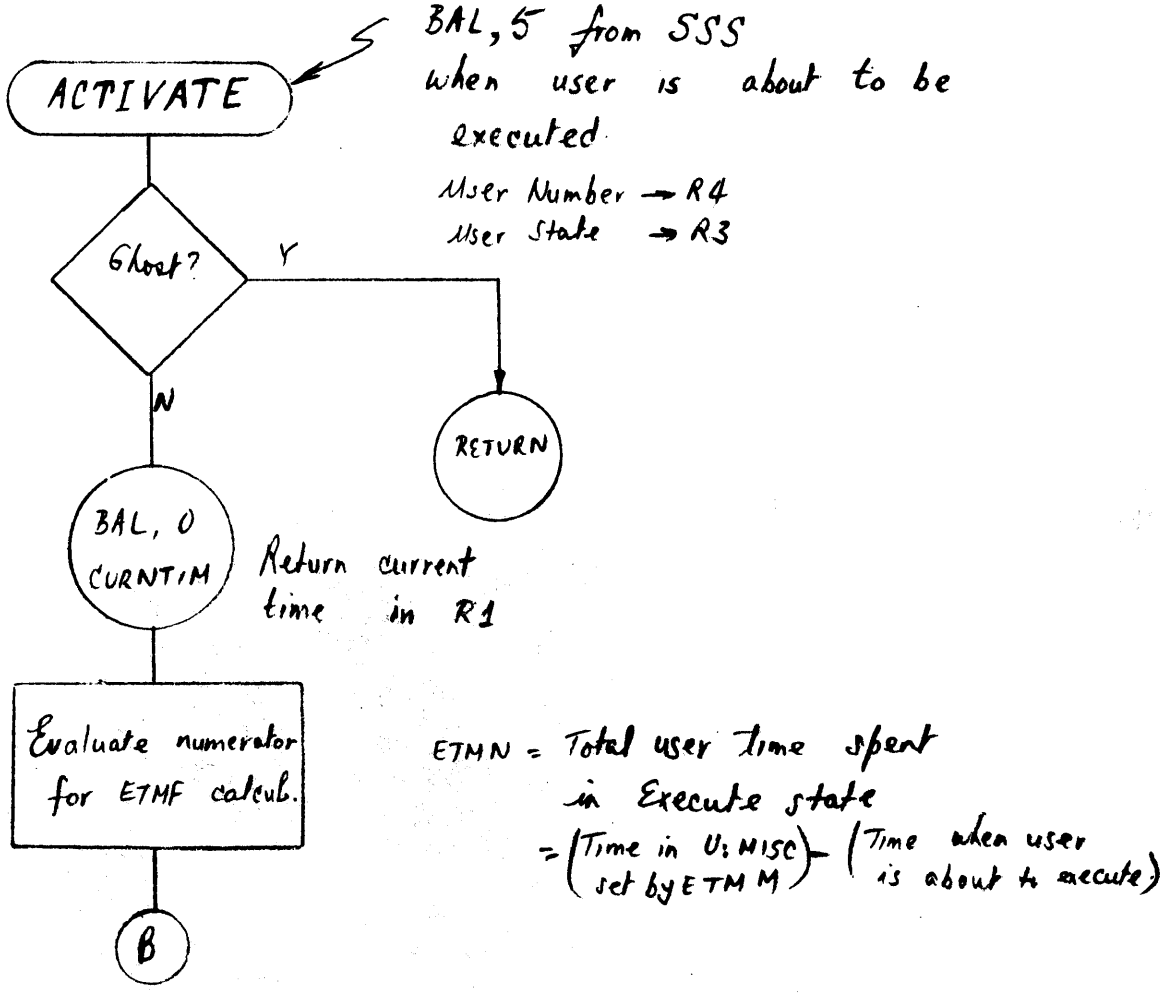
OUTPUT

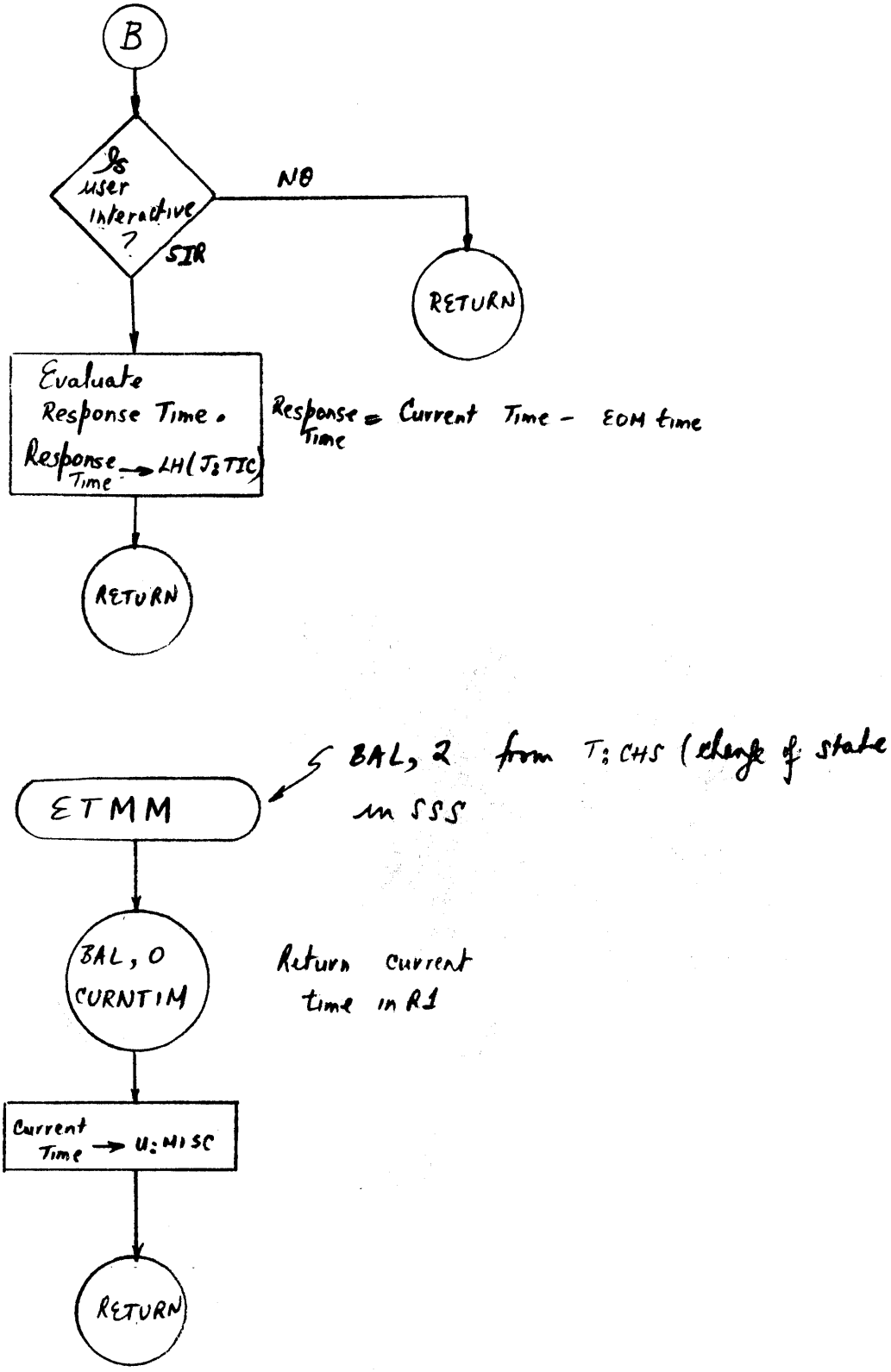
a) ETMN - Numerator for ETMF calculation
b) Response Time - Left half word of J:TIC (for interactive user)
c) Extended value for EOMTIME

DESCRIPTION

Upon entry if the job is identified as a ghost, return is immediate. If not a ghost, a BAL to CURNTIM gets the current time in R1. U:MISC contains the time set by ETMM, when there w(a change to this execute state. Hence the difference between the time when state was entered and the time at which user is ready to execute gives the time spent waiting in the execute state. This is added to the numerator for ETMF calculations.

Following this, inspection of user state indicates whether or not he is interactive. If so, response time is evaluated as (current time − EOM time). Response time is saved in the left halfword of J:TIC. The available EOMTIME value is extended five bits before returning.


ACTIVATE
BAL, 5 from SSS when user is about to be executed
User Number → R4
User State → R3
Ghost?
Y
N
RETURN
BAL, 0 CURNTIM
Return current time in R1
Evaluate numerator for ETMF calcul.
B
ETMN = Total user time spent in Execute state
= (Time in U: MISC set by ETMM) − (Time when user is about to execute)


B
Is user interactive ?
NO
SIR
RETURN
Evaluate Response Time.
Response Time → LH(J;TIC)
Response Time = Current Time - EOM time
RETURN
ETMM
BAL, 2 from T:CHS (change of state in SSS
BAL, 0 CURNTIM
Return current time in R1
Current Time → U:MISC
RETURN

ID

COMPTIM

PURPOSE

Evaluate compute time for a user

USAGE

READREQ enters this routine to evaluate the compute time used during this read.

INPUT

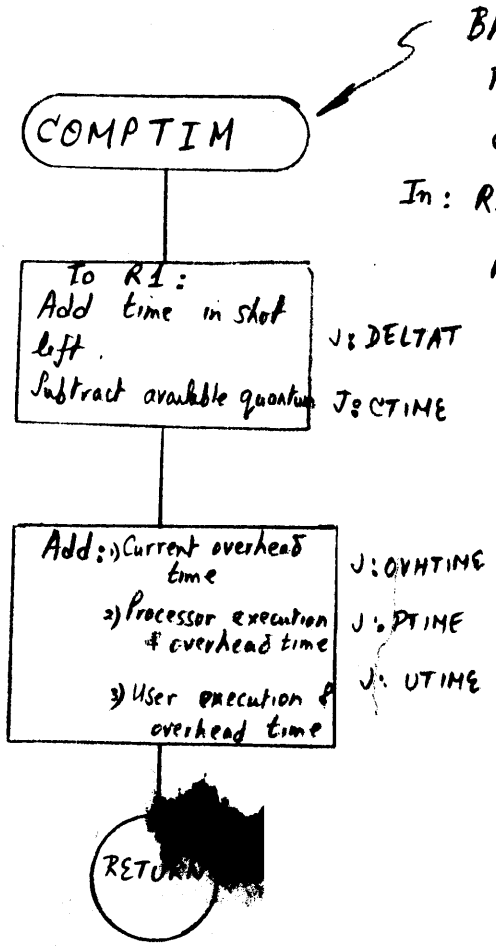
a) Compute time at previous read from J:T or 0 if this is the first read
b) J:DELTAT - time in left
c) J:CTIME - time in quantum
d) J:OVHTIM - current overhead time
e) J:PTIME - processor execution time
f) J:UTIME - user execution time
g) J:PTIME+1 - processor overhead time
h) J:UTIME+1 - user overhead time

OUTPUT

New compute time in R1

DESCRIPTION

The READREQ routine in PM fills the left halfword of J:T with the total compute time used at the end of each read request. The next time a read request is made, READREQ takes what was filled the previous time in J:T and BALs to COMPTIM. It enters COMPTIM with the complement value of previous compute time, in R1. To this COMPTIM adds processor and user overhead and execute times. It also adds the difference in time left and the time originally given him.


COMPTIM
BAL,0 from READREQ to evaluate compute time
In: R1 → -(Compute time at previous read)
or
R1 → 0 if compute time for this read
To R1:
Add time in slot left.
Subtract available quantum
J: DELTAT
J: CTIME
Add: 1) Current overhead time
2) Processor execution & overhead time
3) User execution & overhead time
J: OVHTIME
J: PTIME
J: UTIME
RETURN

ID

GETINDX

PURPOSE

Obtain the offset into the distribution table, for a user or processor.

USAGE

RDMSGSIZ, WTMSGSIZ, READREQ, PMSC execute BAL to GETINDX before a value can be distributed via T:DSTRB. The user or processor index (i.e. displacement) is necessary to determine where in a particular distribution table there is a slot for the processor or user.

INPUT

User number in R1
BGNPMPRC, ENDPMPRC – Limits on processor # for which measurements are to be made.

OUTPUT

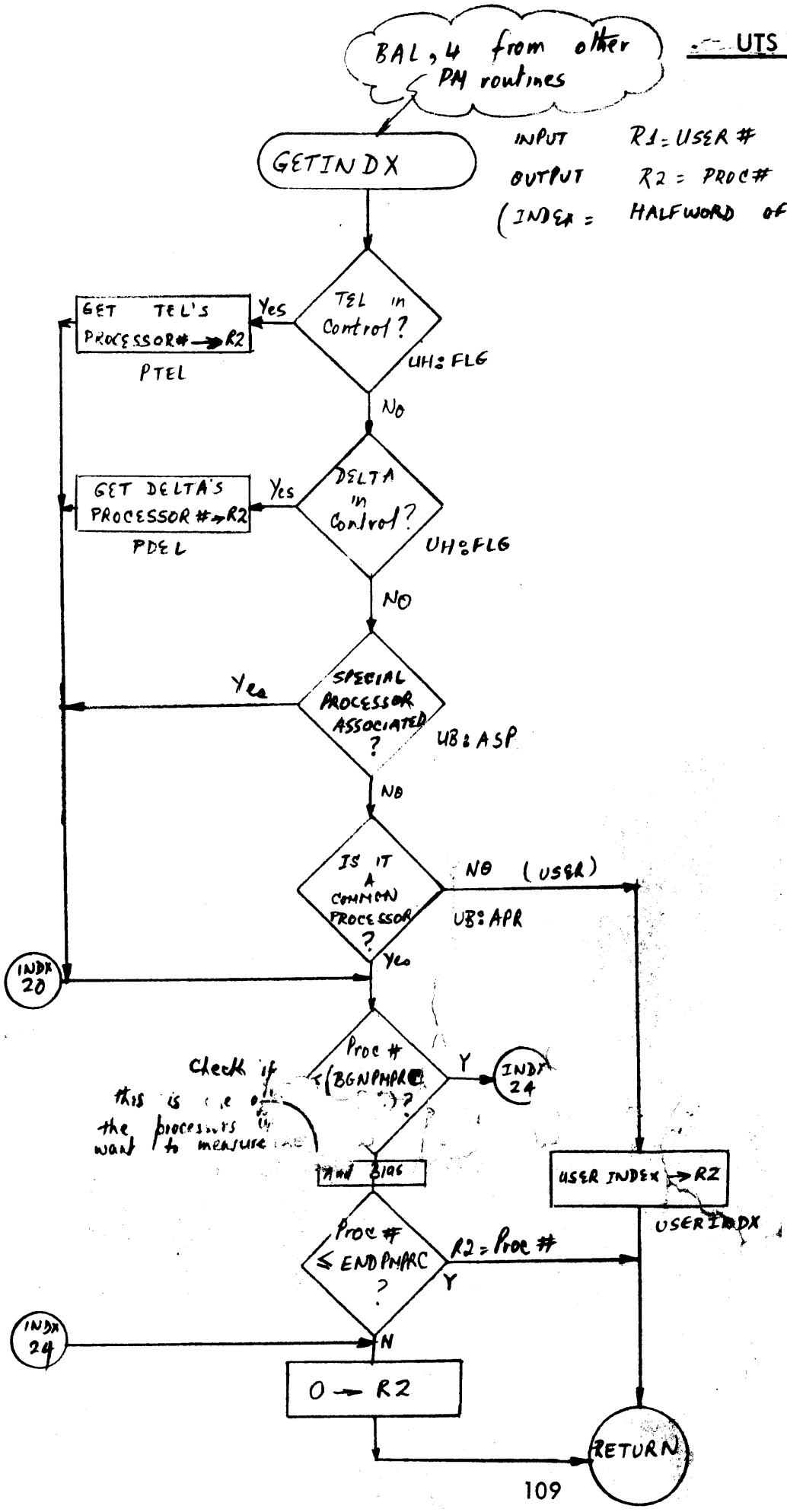
Processor or User index in R2.

DESCRIPTION

On entry, the first check is if TEL is in control. If so, TEL's processor number is obtained (PTEL).

If not, check if DELTA is in control, get DELTA's processor number PDEL. After this, checks are made to determine if it is a user being dealt with, or a processor. If it is not a processor, a user index = ENDPMPRC + 1 is entered into R2, and a routine exit performed. This index value, ensures that the distribution is made in the last distribution bucket (reserved for users).

If it is a processor, a check is made to see the [illegible]essor [illegible] wit[illegible]in the set measurement limits [illegible]NPMPRC and END[illegible]MPRC). [illegible]it do[illegible]t. Otherwise a bi[illegible] of (-BGNPMPRC+1) [illegible]ac[illegible]ed to the processor number. This g[illegible] the processor displacement in R2, before exit.


BAL, 4 from other PM routines
GETINDX
INPUT R1 = USER #
OUTPUT R2 = PROC# OR USER INDEX
(INDEX = HALFWORD OFFSET TO PROC OR USER TABLE)
TEL in Control?
UH:FLG
Yes
GET TEL'S PROCESSOR# → R2
PTEL
No
DELTA in Control?
UH:FLG
Yes
GET DELTA'S PROCESSOR# → R2
PDEL
No
SPECIAL PROCESSOR ASSOCIATED?
UB:ASP
Yes
No
IS IT A COMMON PROCESSOR?
UB:APR
No (USER)
Yes
INDX 20
Proc # < (BGNPMPRC)?
Y
INDX 24
Check if this is one of the processors to measure
and BIAS
USER INDEX → R2
USERINDX
Proc # ≤ ENDPMPRC?
R2 = Proc #
Y
INDX 24
N
0 → R2
RETURN

ID

OSWAP$MEAS, ISWAP$MEAS1, ISWAP$MEAS2

PURPOSE

Compute inswap and outswap times.

USAGE

The '0' routine computes outswap time
The 'I' routine computes inswap time

SSS enters one of the three after issuing an SIO:

a) To swap out a user – OSWAP$MEAS
b) To swap in users JIT and associated processors – ISWAP$MEAS1
c) To swap in rest of user – ISWAP$MEAS2

Return is via R4.

INPUT

R3 = Start time for I/O swap.

OUTPUT

CH:DOT, the outswap time distribution bucket
CH:DI1, DH:DI2, the inswap buckets
CH:DI3, distribution for time between MEAS1 and MEAS2
C:COS is bumped if OSWAP$MEAS is entered (this is the count of outswaps)
CH:SWAPT, time at end of swap

INTERACTION

T:DSTRB is used to [compute value] (R appropriate buckets.
CURNTIM is used t [rent time (2] con).
Both routines are re .. PM.

ROUTINES and DESCRIPTION

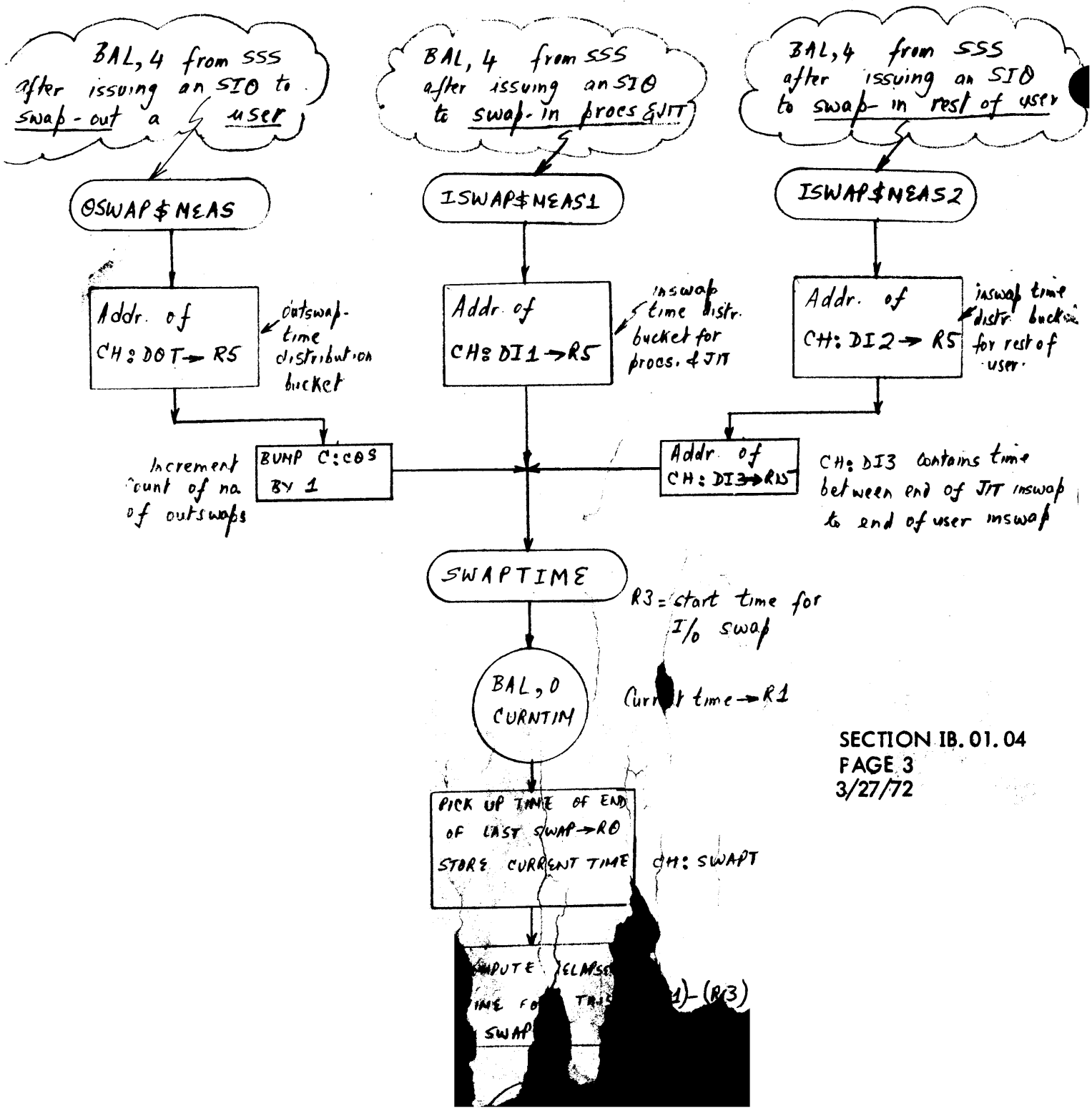
Before executing BAL to T:SIO (to issue an SIO to do a swap) SSS saves the current time in R3. One of the three MEAS routines is entered, after the swap is complete. R3 contains the time at which the swap was started (2 msec resolution).

OSWAP$MEAS - This is entered after a user is swapped out. The address of the distribution table CH:DOT is entered into R5. The count of number of outswap is increment before branching to SWAPTIME (described later).

ISWAP$MEAS1 - This is entered after swapping in JIT and processors. The address of CH:DI1 is entered in R5 before branching to SWAPTIME.

ISWAP$MEAS2 - Entered after rest of user swapped in. Enter address of CH:DI2 into R5 and CH:DI3 into R15. Then move on to SWAPTIME.

SWAPTIME - Save time at end of previous swap from CH:SWAPT into R0. Save time at end of this swap into CH:SWAPT. The time at which this swap was started is available in R3. Difference in time gives the time used for this swap (in R1). This value is distributed by BALing to T:DSTRB. If user inswap is not complete (rest of user not yet swapped in) return to SSS to complete operation. If swap is complete the time at which procs and JIT were swapped in will be in R0. The time the rest of the user got swapped in is in R1. Difference gives the interval between swaps. This is distributed into CH:DI3 via T:DSTRB.



SECTION IB.01.04
PAGE 3
3/27/72

ID

PMSC

PURPOSE

Update appropriate counters (see OUTPUT) for batch and on-line compute time used and monitor services. Calculate denominator for ETMF evaluation.

USAGE

ACCT - The accounting routine, enters here after calculating the product CPU time, times the number of pages used. PMSC will not be entered if the user has exceeded his run time limits (determined by checking J:MRT).

Return is via R10.

INPUT

R3 = Tics of compute time

R4 = { 0 if execution time
     { 1 if overhead time

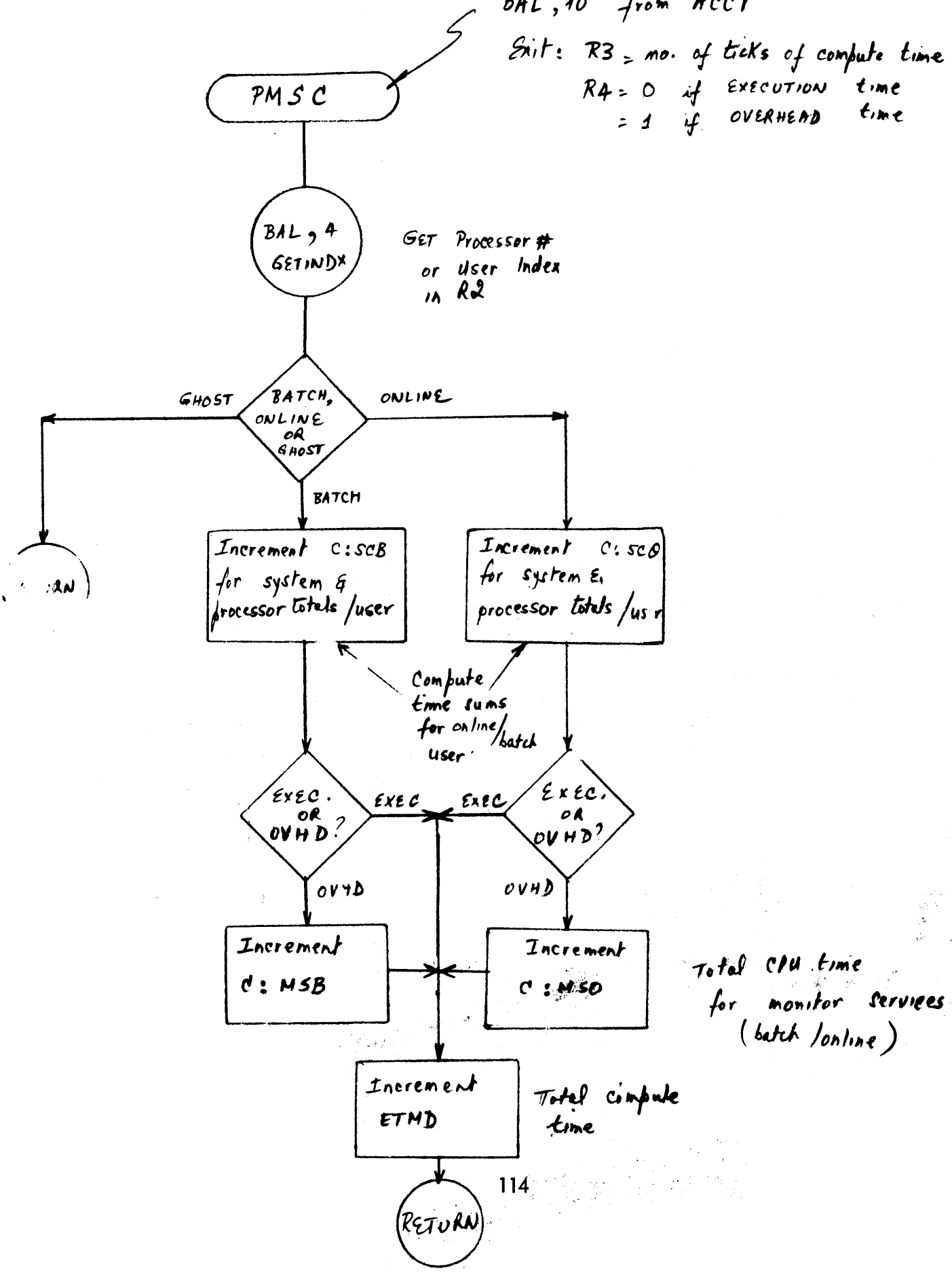
S:CUN User #
J:JIT To check if on-line or batch

OUTPUT

C:SCO, C:MSO	-	Compute time sum, monitor service time if job is on-line.
C:SCB, C:MSB	-	Compute time sum, monitor service time if job is batch.
ETMD	-	De ominator for ETMF calculation.

INTERACTION

IDX - To obtain user or processor index in R2.


BAL, 10 from ACCT
Exit: R3 = no. of ticks of compute time
R4 = 0 if EXECUTION time
= 1 if OVERHEAD time
PMSC
BAL, 4 GETINDX
GET Processor # or User Index in R2
GHOST
BATCH, ONLINE OR GHOST
ONLINE
BATCH
Increment C:SCB for system & processor totals /user
Increment C:SCB for system & processor totals /user
Compute time sums for online/batch user
EXEC. OR OVHD?
EXEC
EXEC
EXEC. OR OVHD?
OVHD
OVHD
Increment C:MSB
Increment C:MSB
Total CPU time for monitor services (batch/online)
Increment ETMD
Total compute time
RETURN

ID

RDMSGSIZ

PURPOSE

To distribute the number of characters per input transaction.

INPUT

Line number in R2
ARS (actual record size) in R10
User number from LB:UN

OUTPUT

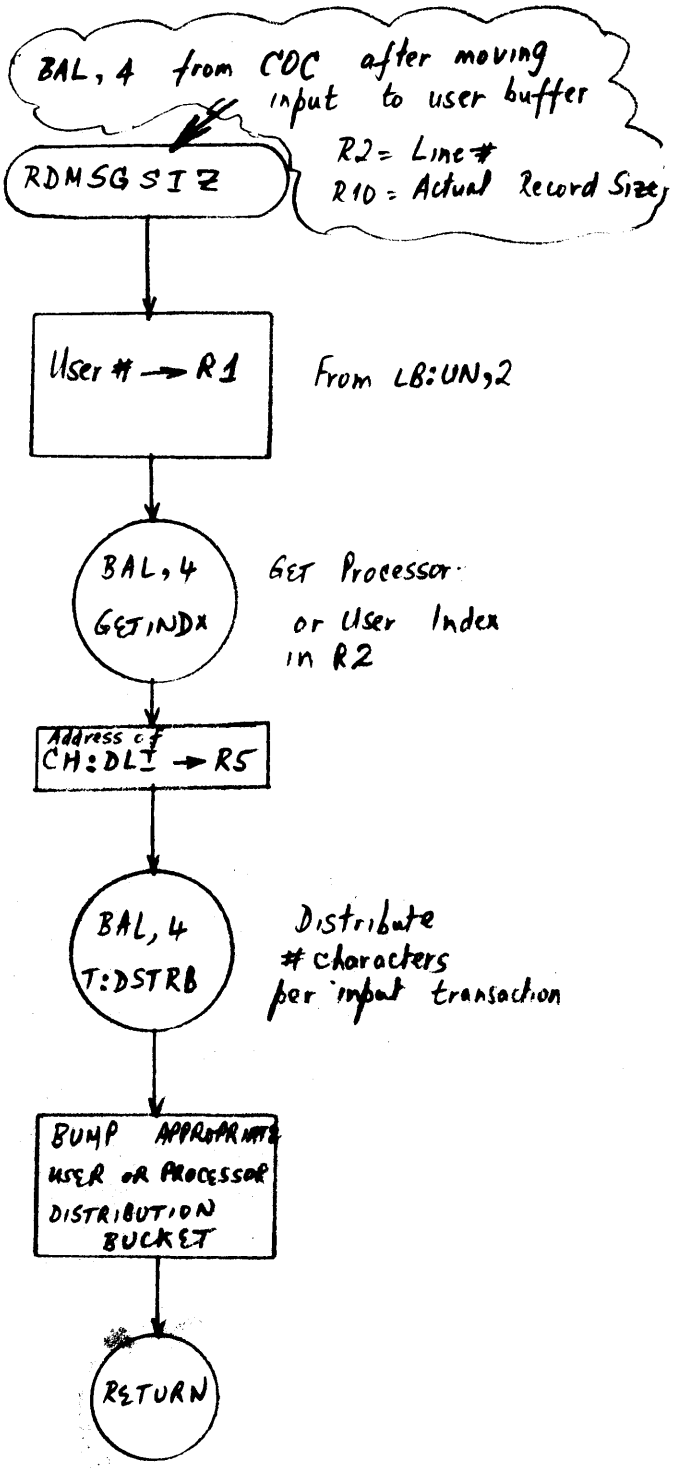
Entries into distribution table CH:DLI

INTERACTION

GETINDX	To get processor number or user index in R2.
T:DSTRB	To distribute the number of input characters.

DESCRIPTION and USAGE

COC routine moves the input to a user buffer and BAL's to this routine. The COC line number and record size are available at entry. The ARS value is moved into R1 in preparation for distribution. A BAL to T:DSTRB, counts up the appropriate bucket. Return via R4


BAL, 4 from COC after moving input to user buffer
R2 = Line #
R10 = Actual Record Size
RDMSGSIZ
User # → R1
From LB:UN,2
BAL, 4
GETINDX
Get Processor or User Index in R2
Address of CH:DLT → R5
BAL, 4
T:DSTRB
Distribute # characters per input transaction
BUMP APPROPRIATE USER OR PROCESSOR DISTRIBUTION BUCKET
RETURN

ID

READREQ

PURPOSE

1. To evaluate and distribute:
   a) System response time
   b) Task turnaround time
   c) Think and type time
   d) Compute time per interaction
2. To set up J:TIC and J:T for use by other PM routines.

USAGE

The COCRD routine is entered each time there is a terminal read request. On entry into COCRD, the first move is to record performance data by a BAL to READREQ.

INPUT

COCLN	Line number
EOMTIME	Saved by COC when input complete

OUTPUT

1) Current time of read in RH of J:T
2) Compute time at read in LH of J:T
3) C:SRT, CH:DRT - Response time sum, distribution
4) C:JIT, CH:DIT - Turnaround time per interaction sums, TT distributions
5) C:ST, CH:DT - Think and type sums, distributions
6) C:JIT, CH:DIT - Interaction time sums and distributions
7) C:SCI - CPU time for interactive interactions
8) C:CITI - Count of interactive interactions
9) C:SC - Total CPU interaction time
10) C:CIT - Count of number of interactions since startup
11) CH:DC - Compute time per interaction distributions
12) Zero in J:TIC

INTERACTIONS

CURNTIM	-	To get current time (2 msec resolution)
GETINDX	-	To get processor number or user index

T:DSTRB - To distribute any one of the computed values into the appropriate bucket.

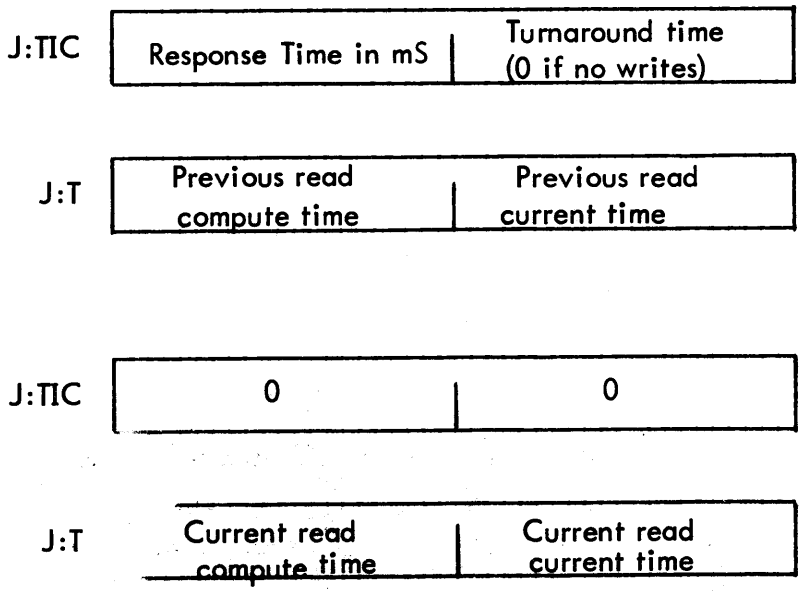
COMPTIM - To evaluate compute time used in this interaction.

DESCRIPTION

A large number of PM calculations are performed in the READREQ routine.

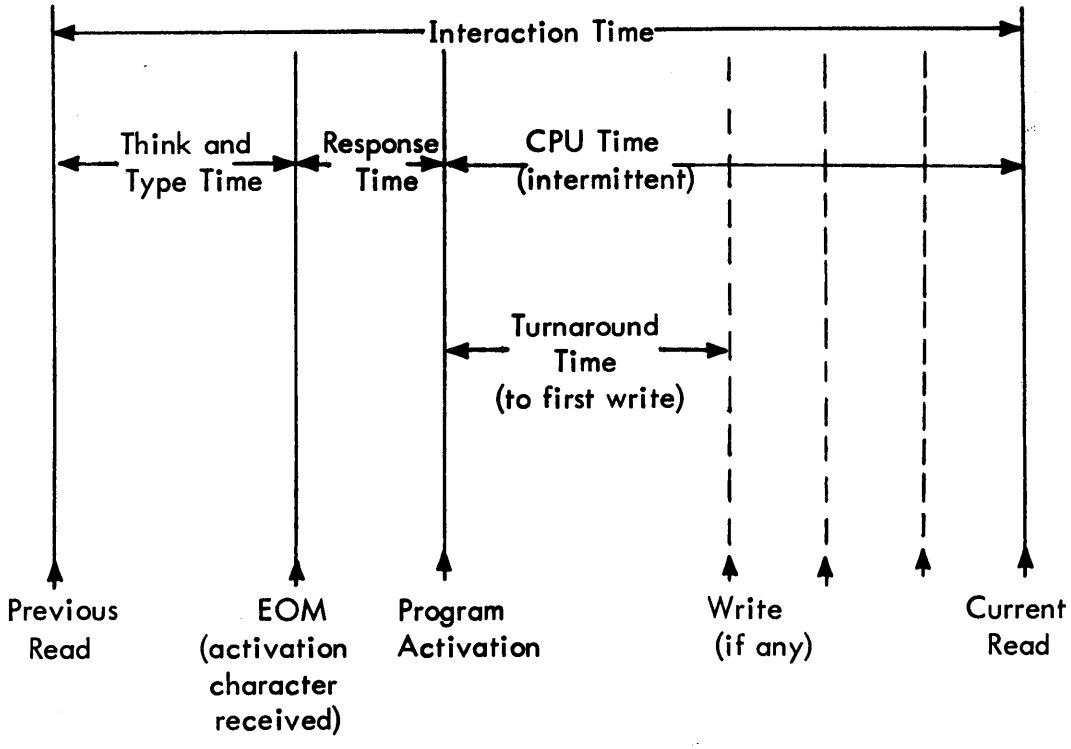
There are two JIT locations reserved specifically for performance measurement purposes. They are J:TIC and J:T. These cells appear as shown below, at entry into READREQ and at exit from the routine.

ENTRY



EXIT

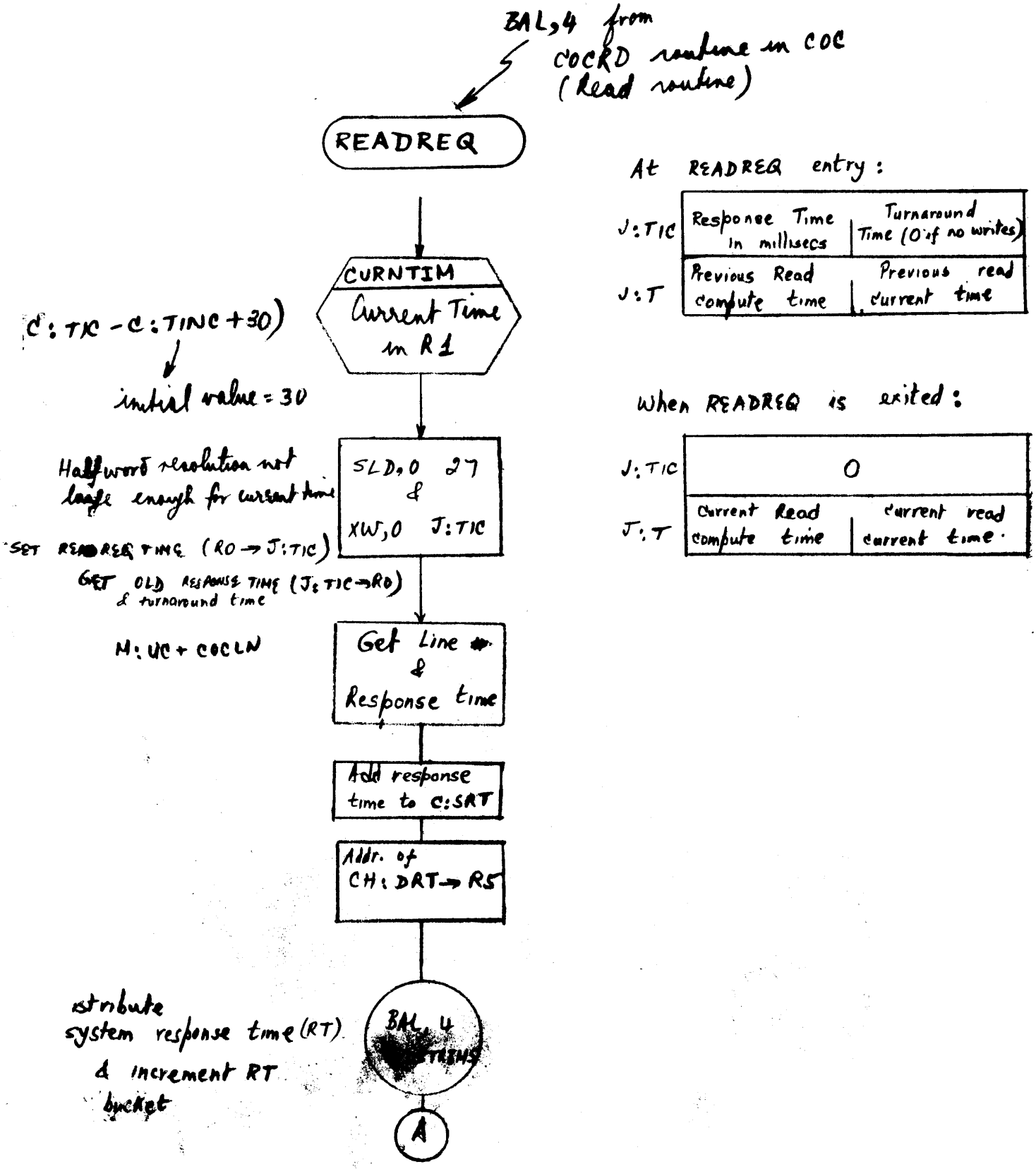
If there have been no · s, prior to this read, the RH of J:TIC is 0. The LH has response time saved p viously by the ACTIVATE routine. This value is distributed into the CH:DRT table and the system response time counter is appropriately incremented. Following this, turnaround time is evaluated if WTMSGSIZ was never entered (a write has not been performed before this read). This is easily calculated since EOMTIME was set by COC when the input message was complete. Current time of read request is obtained via BAL to CURNTIM. The difference in times gives turnaround time. The appropriate counter and bucket are incremented (C:SIT, CH:DTT)

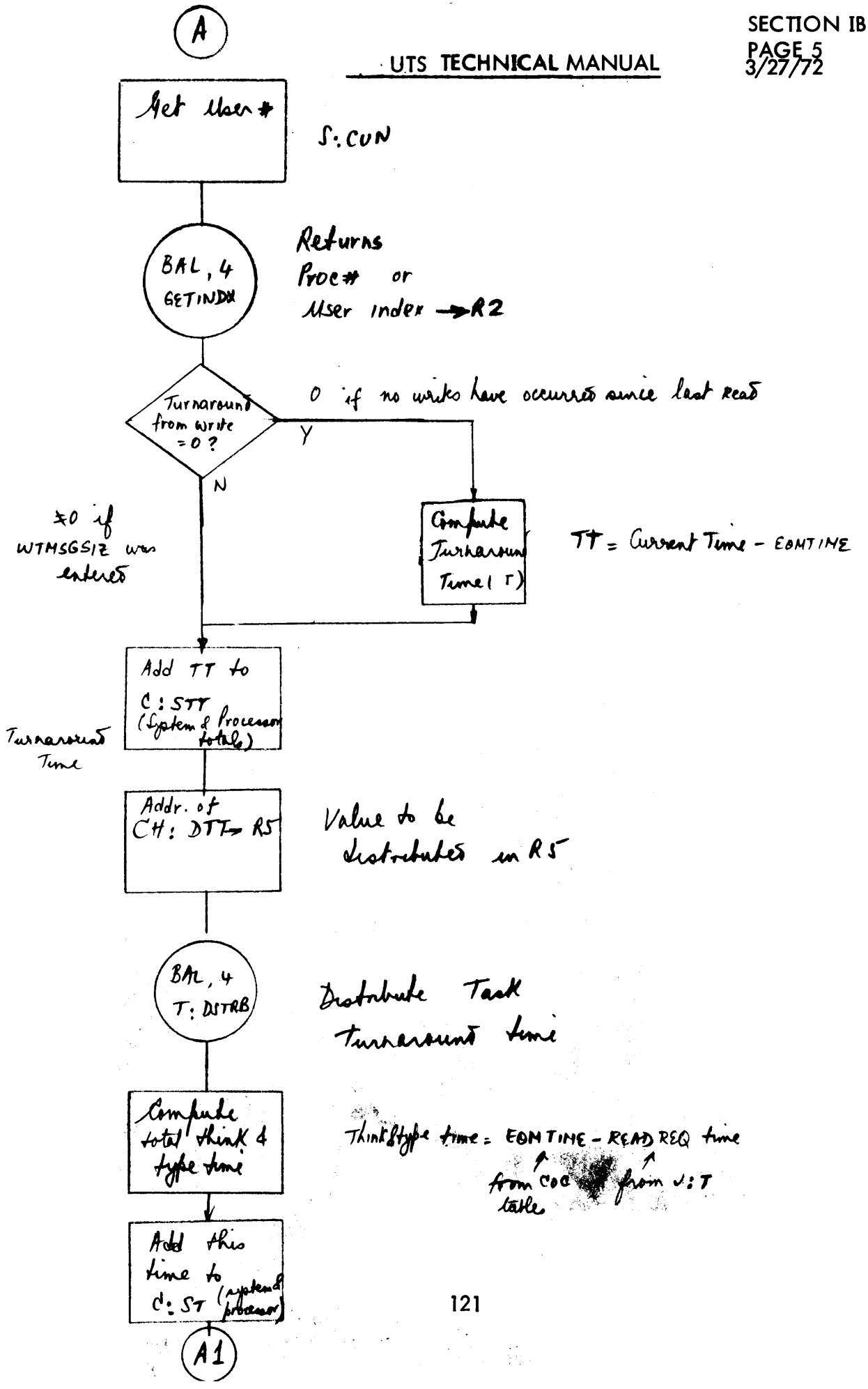


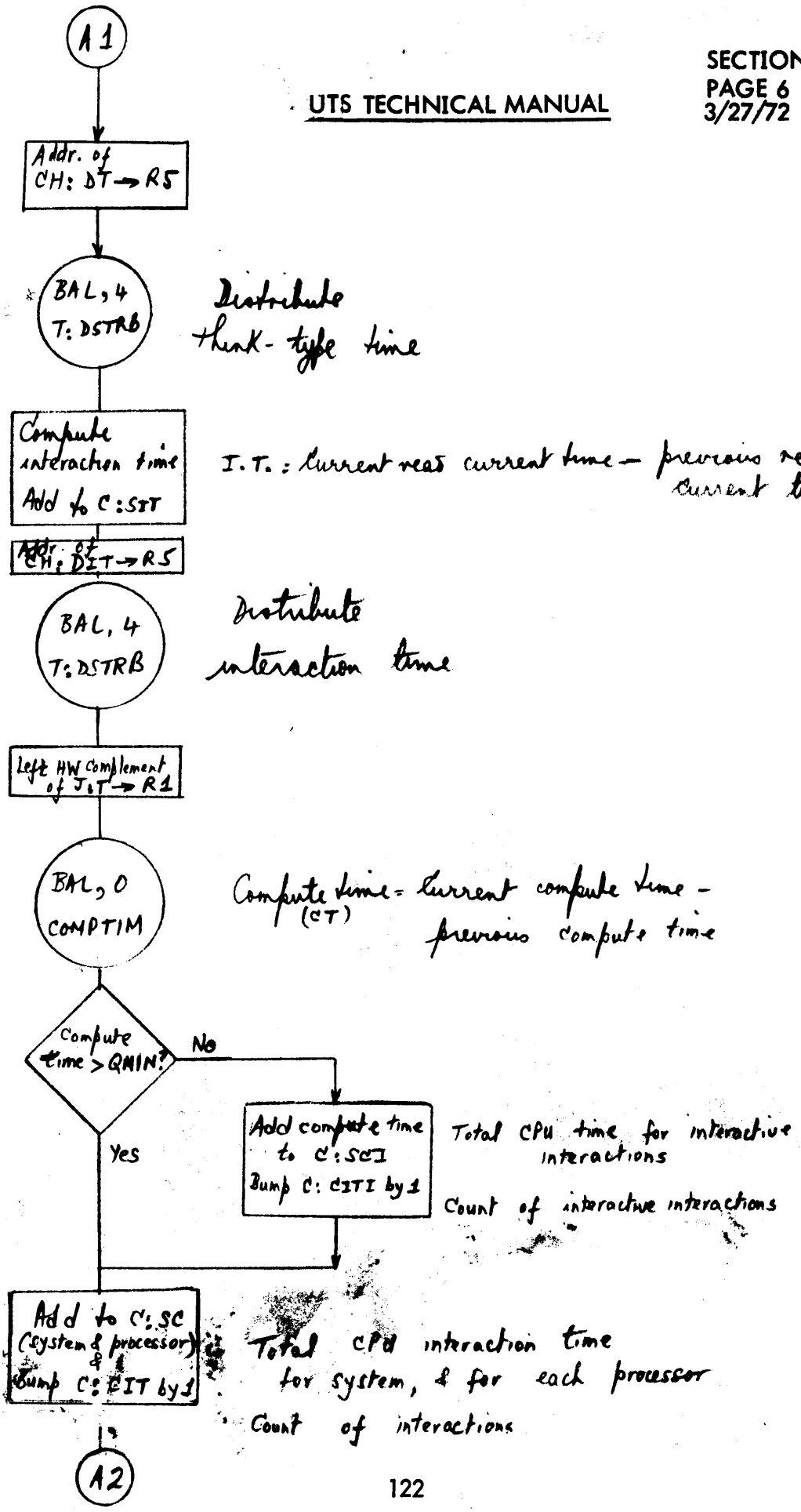
The terminal interaction concept used for performance measurements is as shown above.

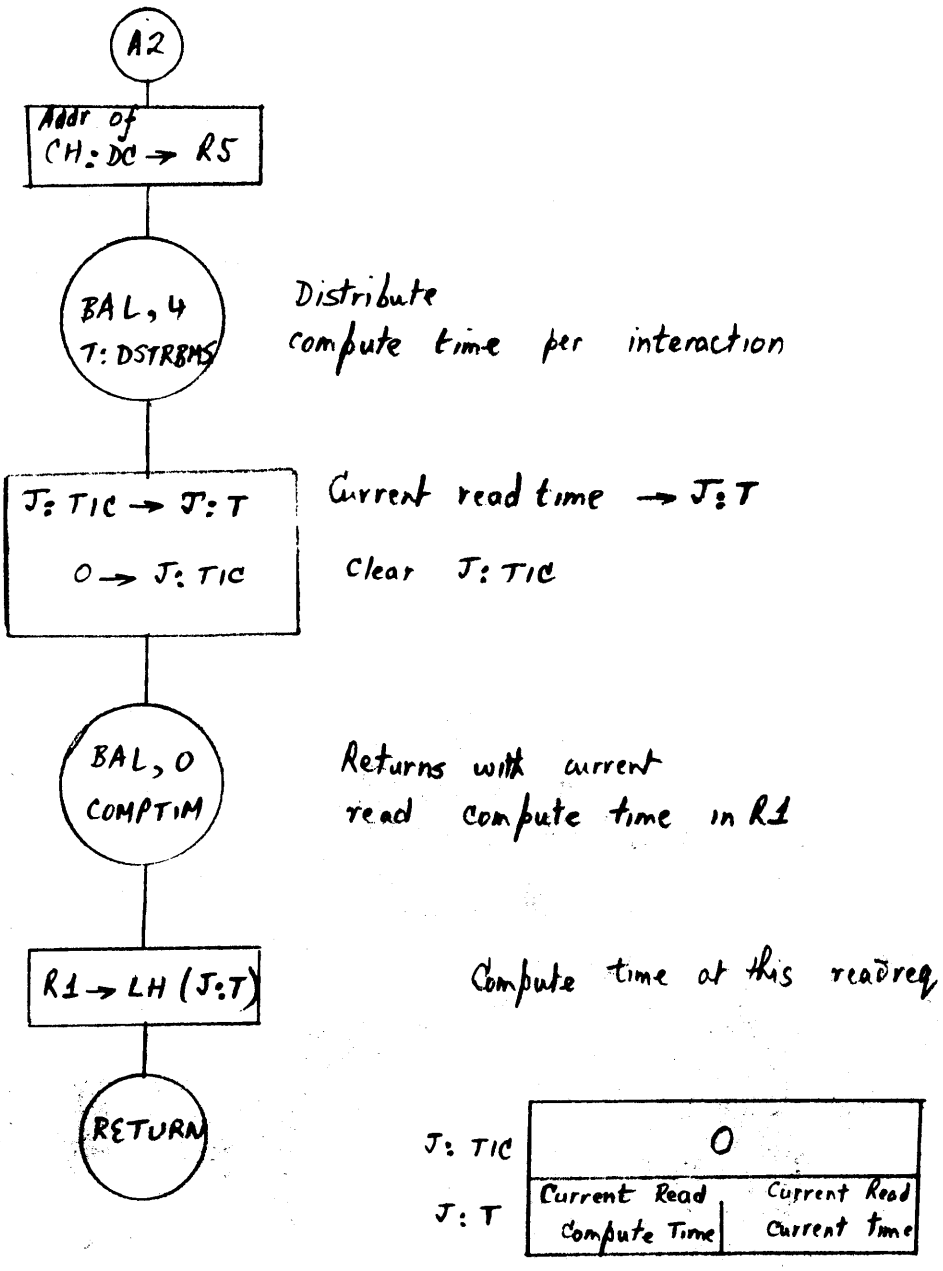
Think and type time is now evaluated. It is the difference between EOMTIME and time of previous read (saved in LH of J:T). The appropriate counter and bucket are incremented (C:ST, CH:DT).

Interaction time is the difference between current read current time (held in R1 by a BAL to CURNTIM at entry to READREQ and moved into J:TIC by an XW operation) and the previous read time which is in the RH of J:T. Appropriate counter and bucket are incremented (C:SIT, CH:DIT). Interaction compute time is evaluated by a BAL to COMPTIM with the complement of previous read total compute time (LH of J:T) in R1. If compute time is less than QMIN, it is an interactive interaction so C:SCI and C:CITI are incremented. C:SC, C:CIT are incremented irrespective of whether compute time is less than or greater than QMIN. Compute time per interaction is now distributed and the CH:DT bucket incremented. Before exiting COMPTIM is called to get the current read total compute time. This is saved in LH of J:T. The RH of J:T is plugged with the current time that was fetched when READREQ was entered this time. J:TIC is cleared and an exit performed through R4.


BAL,4 from
COCRD routine in COC
(Read routine)
READREQ
CURNTIM
Current Time in R1
C:TIC - C:TINC + 30)
initial value = 30
Halfword resolution not large enough for current time
SLD,0 27
&
XW,0 J:TIC
SET READREQ TIME (R0 → J:TIC)
GET OLD RESPONSE TIME (J:TIC → R0)
& turnaround time
M:UC + COCLN
Get Line #
&
Response time
Add response time to C:SRT
Addr. of CH:DRT → R5
Distribute system response time (RT)
& increment RT bucket
BAL,4
A
At READREQ entry:
J:TIC
Response Time in millisecs
Turnaround Time (0 if no writes)
J:T
Previous Read compute time
Previous read current time
When READREQ is exited:
J:TIC
0
J:T
Current Read compute time
current read current time.


A
Get User #
S:CUN
BAL, 4
GETINDX
Returns
Proc# or
User index → R2
Turnaround from write = 0 ?
0 if no writes have occurred since last read
Y
N
≠0 if WTMSGSIZ was entered
Compute Turnaround Time (TT)
TT = Current Time - EOMTIME
Add TT to C:STT (System & Processor totals)
Turnaround Time
Addr. of CH:DTT → R5
Value to be distributed in R5
BAL, 4
T:DSTRB
Distribute Task Turnaround time
Compute total think & type time
Think&type time = EOM TIME - READ REQ time
from COC table
from U:T
Add this time to C:ST (system & processor)
A1


A1
Addr. of CH: DT → R5
BAL, 4 T: DSTRB
Distribute think-type time
Compute interaction time Add to C:SIT
I.T.: Current read current time – previous read current time
Addr. of CH: DIT → R5
BAL, 4 T: DSTRB
Distribute interaction time
Left HW complement of JOT → R1
BAL, 0 COMPTIM
Compute time (CT) = Current compute time – previous compute time
Compute time > QMIN?
No
Yes
Add compute time to C:SCI
Bump C:CITI by 1
Total CPU time for interactive interactions
Count of interactive interactions
Add to C:SC (system & processor) & Bump C:CIT by 1
Total CPU interaction time for system, & for each processor
Count of interactions
A2


A2
Addr of CH:DC → R5
BAL,4 T:DSTRBMS
Distribute compute time per interaction
J:TIC → J:T
0 → J:TIC
Current read time → J:T
Clear J:TIC
BAL,0 COMPTIM
Returns with current read compute time in R1
R1 → LH (J:T)
Compute time at this readreq
RETURN
J:TIC
J:T
0
Current Read Compute Time
Current Read Current time

ID

SWAPRCRD

PURPOSE

Distribute the number of users swapped out.

USAGE

SSS executes a BAL to SWAPRCRD to record the number of users swapped out. Return via R4.

INPUT

SB:OSN Number of users swapped out

OUTPUT

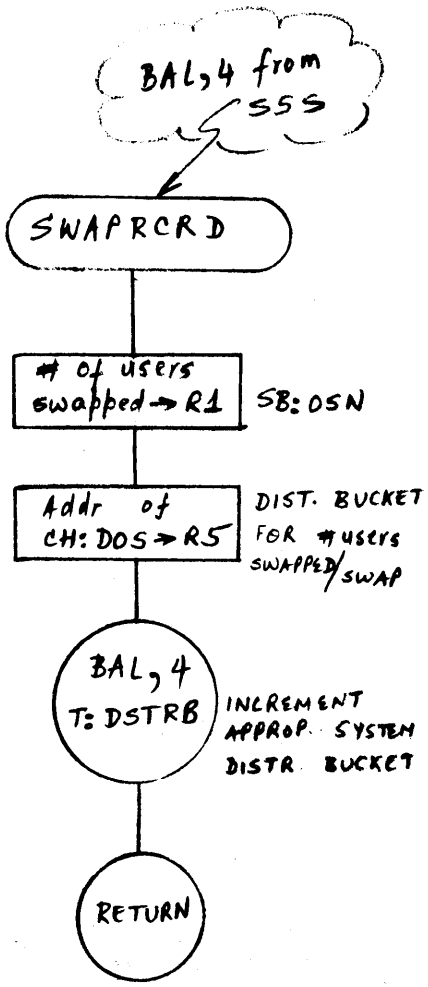
CH:DOS Distribution bucket

INTERACTION

T:DSTRB To distribute the SB:OSN value into appropriate buckets

DESCRIPTION

The number of users swapped out is transferred to R1 from SB:OSN. The address of the distribution bucket is entered in R5 and a BAL to T:DSTRB executed to distribute the value in R1.


BAL,4 from SSS
SWAPRCRD
# of users swapped → R1
SB: OSN
Addr of CH: DOS → R5
DIST. BUCKET FOR # users SWAPPED/SWAP
BAL,4 T: DSTRB
INCREMENT APPROP. SYSTEM DISTR. BUCKET
RETURN

ID

T:SYSTEMLOAD

PURPOSE

a. Evaluate response time distribution
b. Evaluate ETMF
c. Adjust batch and on-line quanta based on system limits and system usage.
d. Compute batch and on-line compute time averages.

USAGE

This routine is entered every minute by the CLOCK4 routine. The system load is returned as output. Return is via R11.

INPUT

ETMN, ETMD - Numerator, Denominator for ETMF calculation

S:BQUAN, SL:BB, SL:QMIN, SL:QUAN - Batch quantum and system limits

UH:FLG - To check if job is batch or on-line

SB:HQ, UB:FL - Swap in queue and FLINKS

S:CUN - Current user number

OUTPUT

a. $ETMF = 1 + \frac{\text{Total user time spent in SB:EXU state}}{\text{Total compute time used}}$

b. 90% point of response time distribution for interactions

c. DRTHEN (response time distribution) is updated

d. Batch and on-line total compute times saved for time period (currently set at 15 minutes) in C:SCOB.

e. Batch quantum is set based on current on-line/batch usage of system.

DESCRIPTION

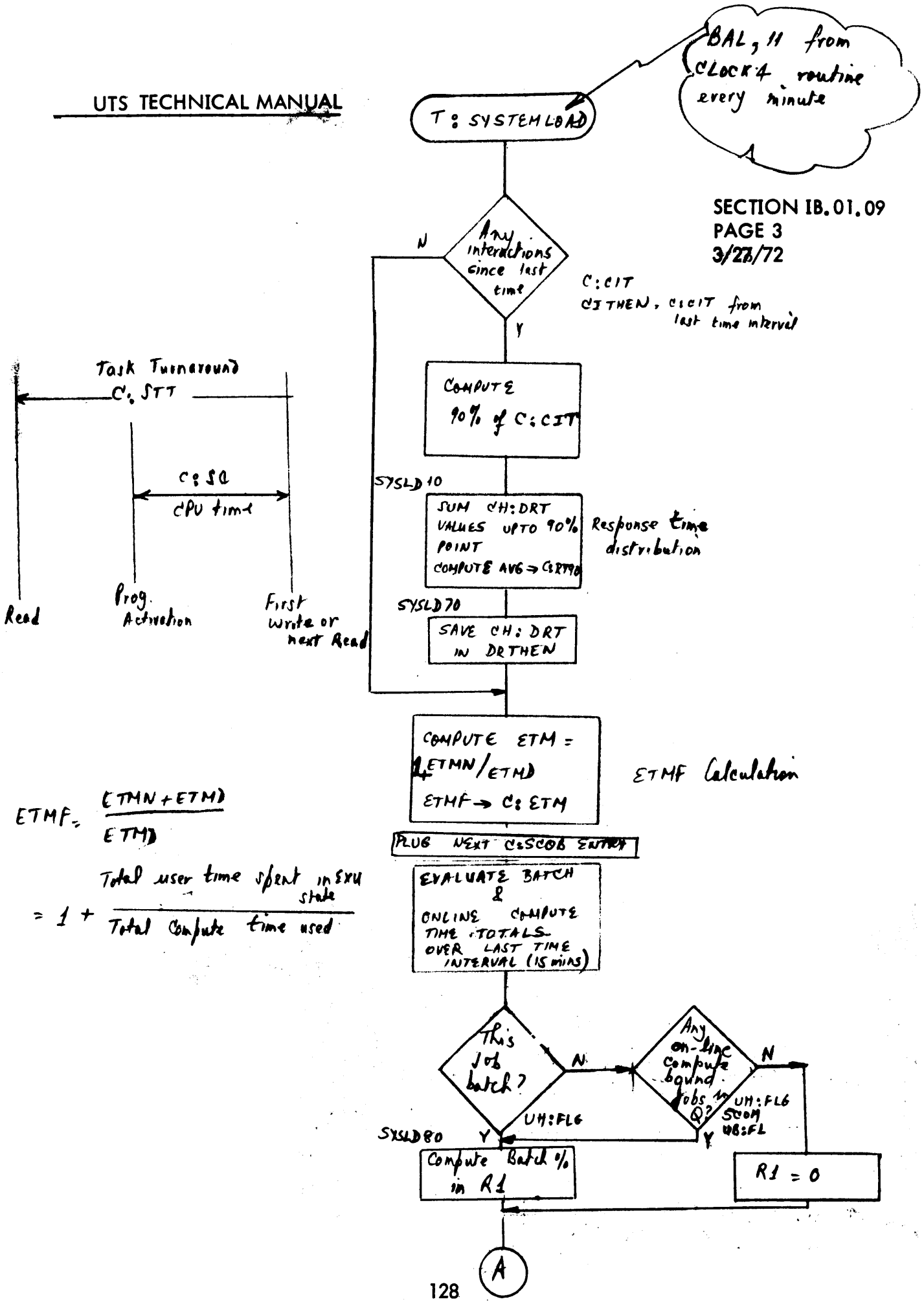
This routine is entered every minute by the CLOCK4 routine to evaluate the system load. The total number of interactions is accumulated in C:CIT. 90% of this value is evaluated and response time distribution buckets in CH:DRT summed until the 90% point is reached. The value in CH:DRT at the end of the minute is saved in DRTHEN. The 90% point of

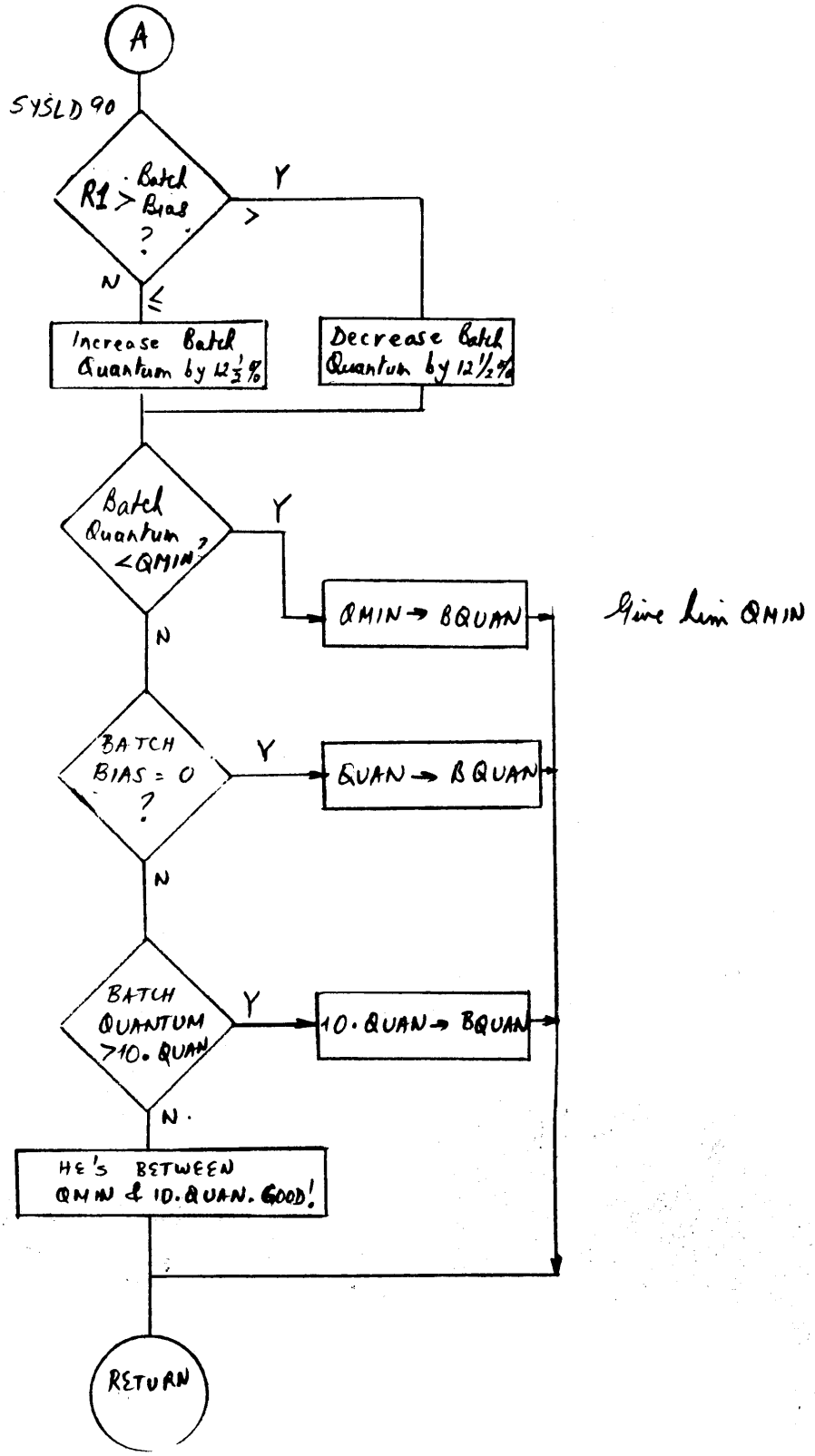
response-time distribution is saved in C:RT90. The number of interactions for the previous minute is held in CITHEN.

Following this ETMF is computed using and zeroing ETMN (calculated in ACTIVATE) and ETMD (calculated in PMSC).

$$ETMF = \frac{ETMN + ETMD}{ETMD}$$ The value is saved in C:ETM

C:SCOB is a 16 word table, with each cell containing a total of on-line and batch compute times(except the first 2 HWDs) computed 1 to 15 minutes ago. The first word has the index to the cell calculated 15 minutes ago and is decremented as each calculation is made (replaced by 15 when 0). Thus C:SCOB and C:SCO and C:SCB give compute sums over the last 15 minutes (this time period can be modified by changing BBINT). The swapper state queues are checked for any on-line compute bound users. If none, batch quantum can be stepped up. If there are on-line compute bound users, the batch % is calculated and compared against SL:BB. If it is less, S:BQUAN is added to the current batch quantum and a check made to see if it lies between QMIN and (10 x QUAN). If greater than 10 x QUAN, the batch quantum is set at 10 x QUAN. If less than QMIN set it at QMIN. If there are no on-line compute bound users and batch bias (SL:BB) is set to zero, batch quantum is set at SL:QUAN.


BAL, 11 from CLOCK 4 routine every minute
T: SYSTEMLOAD
Any interactions since last time
N
Y
C:CIT
CITHEN, C:CIT from last time interval
COMPUTE 90% of C:CIT
SYSLD10
SUM CH:DRT VALUES UPTO 90% POINT
COMPUTE AVG → CRT90
Response time distribution
SYSLD70
SAVE CH:DRT IN DRTHEN
COMPUTE ETM = 1+ETMN/ETMD
ETMF → C:ETM
ETMF Calculation
PLUG NEXT C:SCOB ENTRY
EVALUATE BATCH & ONLINE COMPUTE TIME TOTALS OVER LAST TIME INTERVAL (15 MINS)
This Job batch?
N
Y
UH:FLG
Any on-line compute bound jobs in Q?
N
Y
UH:FLG
SYSLD80
Compute Batch % in R1
R1 = 0
A
Task Turnaround
C: STT
C: SQ
CPU time
Read
Prog. Activation
First write or next Read
ETMF = (ETMN + ETMD) / ETMD
= 1 + Total user time spent in EXU state / Total Compute time used


A
SYSLD 90
R1 > Batch Bias ?
Y
>
N
≤
Increase Batch Quantum by 12½%
Decrease Batch Quantum by 12½%
Batch Quantum < QMIN ?
Y
QMIN → BQUAN
Give Lim QMIN
N
BATCH BIAS = 0 ?
Y
QUAN → BQUAN
N
BATCH QUANTUM > 10. QUAN
Y
10. QUAN → BQUAN
N.
HE'S BETWEEN QMIN & 10. QUAN. GOOD!
RETURN

ID

T:DSTRB

PURPOSE

To distribute an input parameter (e.g. response time) into the appropriate distribution table and increment appropriate system bucket.

USAGE

Any routine which calculates values relevant to system performance evaluation calls to this routine to distribute the value. Before performing the BAL, the address of the table into which the value will be distributed should be contained in R5. The value itself should be in R1. Return is via R4.

INPUT

R1 Value to be distributed
R5 Address of distribution bucket (CH:XXX)
R2 Processor number or user index into tables.

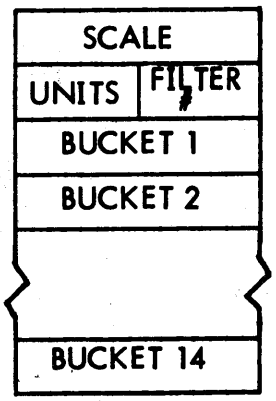
OUTPUT

Distribution bucket is incremented
R5: Halfword address of bucket to be incremented for processor or user distribution (or zero if R2 = 0)

DESCRIPTION

A distribution table is shown below. It is 16 halfwords long.

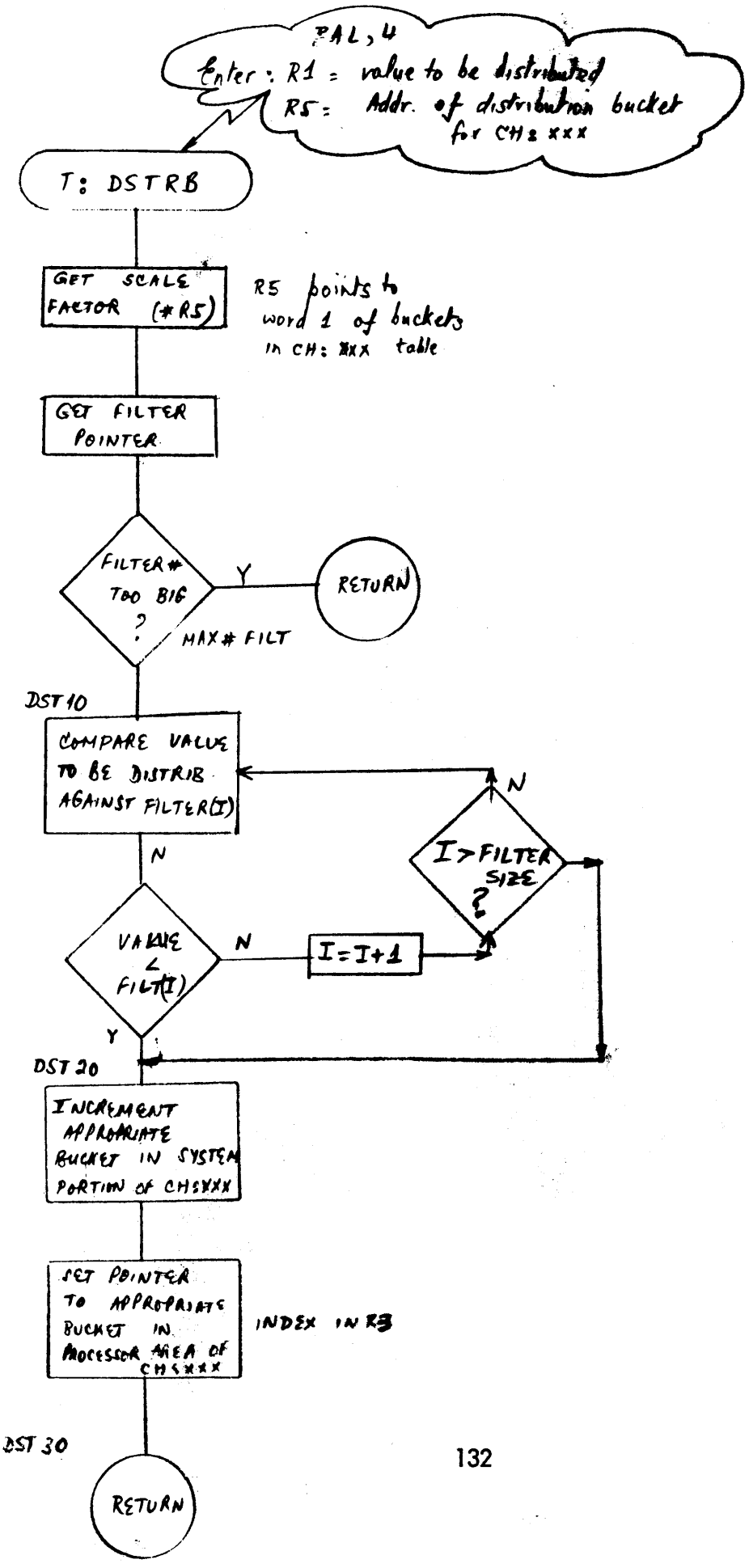
CH:XXX Table



The values of the scale units and filter used for each particular table are generated in PMDAT. On entry, R5 points to scale factor which is picked up and used as input divisor. After this the filter number is accessed and if valid the correct filter is accessed. The values of the scale units and filter used for each particular table are generated in PMDAT. On entry, R5 points to scale factor which is picked up and used as input divisor. After this the filter number is accessed and if valid the correct filter is accessed. The value being distributed is compared successively against each filter element in the filter until it is less than the element value. The count is bumped in bucket 1 if the scaled data is less than the first filter element value.

Count is bumped in bucket n if it is less than the n^{th} filter element value.

The first distribution is for system performance. If distribution is included for each processor as well, there will be more than one such table. The index to these processor distribution tables is calculated in GETINDX. The processor number is biased by a value of BGNPMPRC + 1. Only those processors whose numbers lie between BGNPMPRC and ENDPMPRC (which are values generated at SYSGEN) will be measured (non-zero processor index). All others will come out of GETINDX with an index value of 0. The last distribution, if more than one distribution is present is for user. The user index is set at ENDPMPRC + 1 by PM.


PAL, 4
Enter: R1 = value to be distributed
R5 = Addr. of distribution bucket for CH: xxx
T: DSTRB
GET SCALE FACTOR (*R5)
R5 points to word 1 of buckets in CH: xxx table
GET FILTER POINTER
FILTER # TOO BIG ?
Y
RETURN
MAX # FILT
DST 10
COMPARE VALUE TO BE DISTRIB AGAINST FILTER(I)
N
I > FILTER SIZE ?
N
VALUE < FILT(I)
N
I=I+1
Y
DST 20
INCREMENT APPROPRIATE BUCKET IN SYSTEM PORTION OF CH:XXX
SET POINTER TO APPROPRIATE BUCKET IN PROCESSOR AREA OF CH:XXX
INDEX IN R3
DST 30
RETURN

ID

WTMSGSIZ

PURPOSE

1) Count up the CH:DLO bucket (output message length)
2) Set turnaround time if this is the first write after a read.

USAGE

COC BALs here to distribute the length of output messages sent out.

INPUT

Line number in R2.
Count of output characters when request was made R5.
Current count of output characters in COCOC (COC tables).
EOMTIME used to calculate turnaround time for first write.

OUTPUT

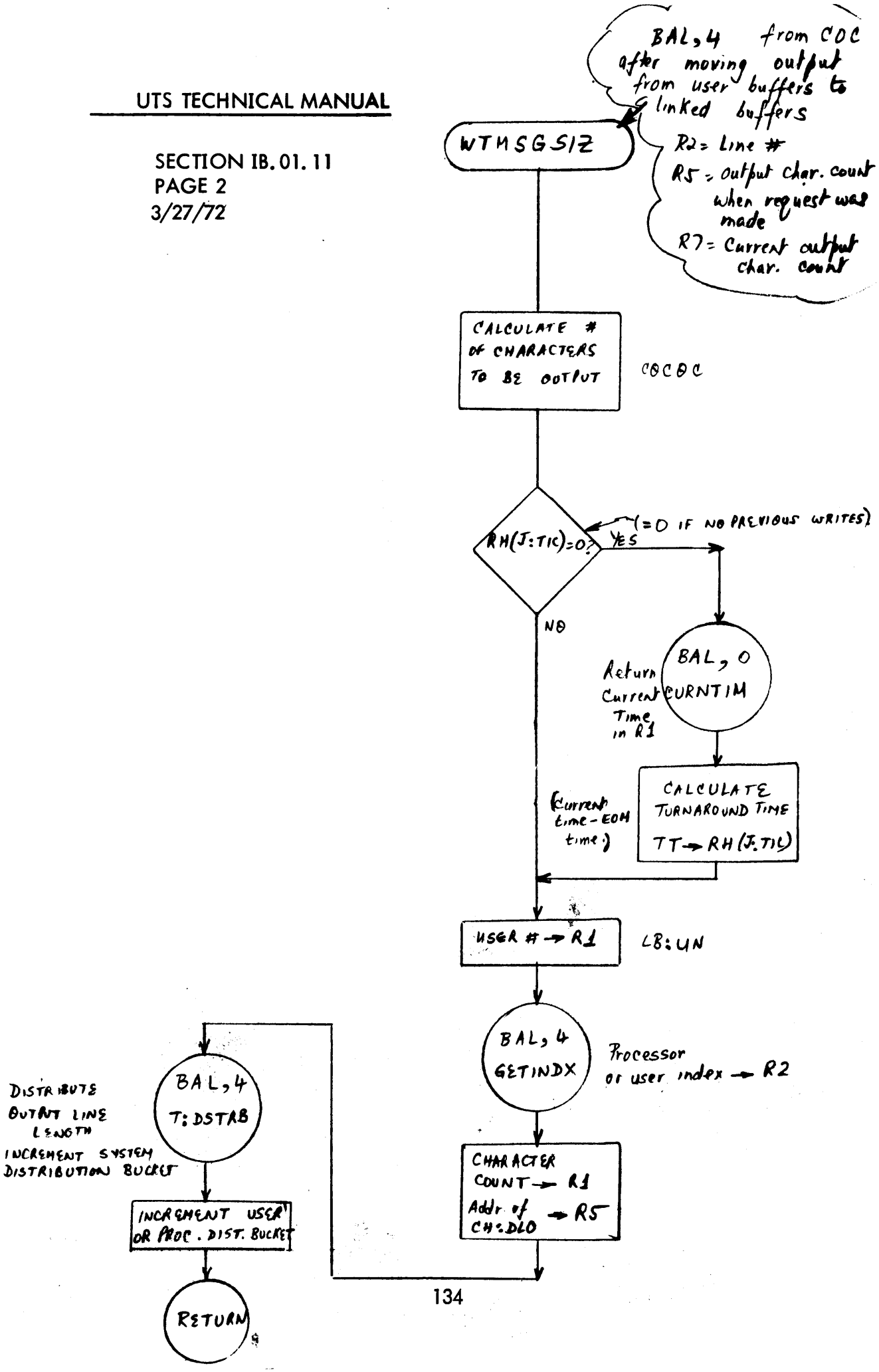
Turnaround time in J:TIC if this is first write after read.
CH:DLO distribution bucket is incremented.

INTERACTION

GETINDX	-	Processor or user displacement into CH:DLO
T:DSTRB	-	To distribute the character count

DESCRIPTION

Things start moving when an output message is generated and handed to COC to perform a write. COC moves the message into linked buffers and BALs to WTMSGSIZ to distribute the output length. It enters with the COC line number in R2. The count of output characters when the write request was made is in R5. On entry an initial adjustment of character count is made, to keep track of some characters which may not have got into COCOC. Then the right halfword of J:TIC is tested for zero. If zero, this is the first write, and turnaround time is computed: current time at write (CURNTIM) – time at end of message (EOMTIME). This is stored into the right halfword of J:TIC. If this is not the first write J:TIC is not changed. The address of CH:DLO is put into R5 and the output character count distributed.


BAL,4 from COC after moving output from user buffers to linked buffers
R2 = Line #
R5 = output char. count when request was made
R7 = Current output char. count
WTMSGSIZ
CALCULATE # OF CHARACTERS TO BE OUTPUT
COCOC
RH(J:TIC)=0?
(=0 IF NO PREVIOUS WRITES)
YES
NO
BAL,0 CURNTIM
Return Current Time in R1
CALCULATE TURNAROUND TIME TT → RH(J.TIC)
(Current time - EOM time)
USER # → R1
LB:UN
BAL,4 GETINDX
Processor or user index → R2
CHARACTER COUNT → R1
Addr of CH:DLO → R5
BAL,4 T:DSTAB
DISTRIBUTE OUTPUT LINE LENGTH INCREMENT SYSTEM DISTRIBUTION BUCKET
INCREMENT USER OR PROC. DIST. BUCKET
RETURN

ID

Accounting

PURPOSE

The UTS monitor accounting procedures record the amount of resources provided to a user of the system during a single job or on-line session.

OVERVIEW

When UTS recognizes a user ready to log on, the swapper sets the initial system limits into his Job Information Table the first time it is brought into core. These limits were provided by the :BLIMIT and :OLIMIT options during PASS2 of Sysgen.

After receiving an account/name combination from the user, LOGON (or LOGRT in CCI for batch users) reads the :USERS file with the account/name as a key; a successful read validates the user. Non-zero limits in the record from the :USERS file, supplied by SUPER, replace corresponding system default limits in the JIT. If the job is batch, CCI may obtain limits from the !LIMIT card to replace those previously established.

While the user's job runs, UTS records the resources used while insuring none of the specified maximums is exceeded; the job is aborted if any is.

Upon completion of the job, or when logging off, LOGON (for on-line and batch users) writes the information accumulated in the JIT to a sequential record in the :ACCTLG file, to be used as raw data for the installation's accounting routines.

REFERENCES

This document concerns itself mostly with where and how accounting information is obtained within the UTS Monitor, without much attention paid to the processors which take part in the accounting process, specifically LOGON, SUPER, CCI, RATES. The P and Q sections of the Technical Manual are recommended for additional information.

USAGE

Varies, documented by routine.

INPUT

Documented by routine.

OUTPUT

:ACCTLG file in the :SYS account
:USERS file in the :SYS account
JIT cells documented by routine.

INTERACTION

Varies, documented by routine

DATA BASES

Job Information Table, Section VA.
Administrative Data Bases, Section VN.

ERRORS

All aborts of a program for exceeding a system limit may be said to come from accounting. In each situation, the appropriate bit is set in J:ASSIGN in the JIT to indicate the type of max exceeded, and a X'04' set into byte 0 of J:RNST. The user will be aborted when a non-zero run status is recognized by the monitor.

DESCRIPTION

UTS accounting may be separated into three general areas: Time, file space and simple accumulation. Of these, time is the only area which contains a routine whose sole function is accounting; the other areas consist of short portions of code in many routines.

UTS processor interaction with the accounting process occurs in two areas: During the logon process when limits are established, and during logoff when the accounting summary is written to the :ACCTLG file.

Time Accounting

The accounting of time in UTS is dependent upon the hardware modification to CLOCK4 to allow it to tick through the map whenever the map bit is set in the current PSD. Therefore, CLOCK4 ticks into one of two cells of the current user's JIT:

J:DELTAT	During user execution, and
J:CVHTIM	During monitor services.

Unmapped, CLOCK4 ticks into the corresponding cells in the monitor JIT, which was placed by the initialization process at the physical location X'8C00', which is the user JIT's virtual address. These cells are always ≥ 0 to avoid the CLOCK4 interrupt in the unmapped monitor.

Before control is transferred to a user by the scheduler in the SSS routine T:SE, the address of either J:OVHTIM or J:DELTAT is stored into the address field of X'55', the CLOCK4 location. Which address is determined by the setting of the slave bit of the PSD in the user's environment: J:DELTAT if slave, J:OVHTIM if master.

When a slave mode user program issues a CAL, ENTRY switches the address field of X'55' to J:OVHTIM to accumulate monitor overhead. This cell is always $\geq$ to zero so the timer zero interrupt will not occur while in the monitor. Upon completion of the CAL processing, control is passed to T:SSEM in the scheduler which calls the routine ACCT, entering at T:ACCTOV, to transfer the accumulated monitor overhead time to the appropriate cells in the JIT. T:SSEM falls into T:SSE, which sets J:DELTAT into X'55', then determines whether a user other than the current user is to run next. If no other user is selected, control is returned to the current user at the CAL+1. Otherwise T:ACCTEX in ACCT is called to transfer the time difference between J:CTIME and J:DELTAT (the amount of time used when slave mode) to the appropriate JIT cells.

Subroutines (Time Accounting):

T:ACCTOV

Overhead time, accumulated from the time ENTRY changed the clock, is retrieved from J:OVHTIM which is then zeroed. R4 is loaded with 1 to indicate overhead accounting, and T:ACCT is called. Upon return from T:ACCT, the address of J:DELTAT is set into X'55' and control returned to the calling routine.

Calling sequence:
BAL, 11 T:ACCTOV

T:ACCTEX

The positive difference between J:CTIME, the amount of time allocated to the user, and J:DELTAT, the amount remaining, is obtained. R4 is set to zero to indicate execution time accounting and T:ACCT is called. Upon returning from T:ACCT, control is returned to the calling routine.

Calling sequence:
BAL, 11 T:ACCTEX

T:ACCT

The address of the appropriate accounting statistics to update is determined by byte 1 of J:RNST. If a user is running, the address is J:UTIME, otherwise J:PTIME is used.

J:PTIME	Processor Execution
+1	Processor Overhead
+2	CPU Time * Core size
J:UTIME	User Execution
+1	User Overhead
+2	CPU Time * Core size

R4 contains the index, 0 for execution time and 1 for overhead, and R3 contain the positive time in 2 msc tics to be accumulated; R3 is added to the appropriate cell.

The negative of the time is added to J:UTIMER for the M:STIMER routine, and also to J:MRT if the latter is non-zero. (A zero J:MRT indicates the user is not to be aborted for max time). If J:MRT becomes negative thereby,

0 ⟶ J:MRT
X'80' ⟶ J:ASSIGN
X'4' ⟶ J:RNST (Byte 0)

To indicate the user is to be aborted for exceeding his maximum run time. If a command processor is not in control (SJAC not set in UH:FLG) control is transferred to T:TELDELCCI in STEP to abort the user. Otherwise T:ACCT proceeds normally.

If maximum time has not been exceeded, or a command processor is in control, the positive overhead time (if overhead is being accounted) is added to J:CTIME and J:DELTAT. Then the positive time in either case is multiplied by the number of pages currently required to contain the user (from UB:PCT), and the product stored into the appropriate cell of J:UTIME or J:PTIME. Control is then transferred to PMSC in the performance measurement routines which will return to the original caller of T:ACCT.

in: R3 Positive time in 2msc tics
R4 0 for execution time
1 for overhead time

Calling sequence:

```
BAL,10 T:ACCT
```

Granule Accounting

Monitor routines which perform granule accounting are CLS, DLT, OPN, and WRTF. The following JIT cells are maintained by granule accounting:

PRDCRM	Permanent RAD Space Remaining
PRDPRM	Permanent PACK Space Remaining
TMDCRM	Temporary RAD Space Remaining
TMDPRM	Temporary PACK Space Remaining
TMPDCPK	Peak Temporary RAD Space Used
TMPDPPK	Peak Temporary PACK Space Used

Subroutines (Granule Accounting):

CLS

Gets granules for additions to directories. CLS decrements PRDCRM/PRDPRM for each granule it gets. No maximum test is made for granules obtained by CLS. CLS also releases random file granules, incrementing either PRDCRM/PRDPRM, or TMDCRM/TMDPRM.

DLT

Releases granules. After each granule is released, PRDCRM/PRDPRM is incremented for files for which the CFU FUN is IN or INOUT. For files with FUN of OUT or OUTIN, and SAVE previously specified in the DCB, PRDCRM/PRDPRM is incremented and TMDCRM/TMDPRM decremented, as the file was assumed permanent when written. If SAVE was not specified, TMDCRM/TMDPRM is incremented as one might expect.

OPN

Gets granules for random files. At OPNRAND the number of granules required is compared with either PRDCRM/PRDPRM or TMDCRM/TMDPRM depending on SAVE being specified or not in FIL1 of the DCB. If greater, an I/O error 57 results. Otherwise the number of granules required is obtained and the number subtracted from PRDCRM/PRDPRM or TMDCRM/TMDPRM. In the case of temporary files, TMPDCPK/TMPDPPK is updated accordingly.

WRTF

Gets granules. WRTF decrements PRDCRM/PRDPRM for each granule of permanent file storage it writes, permanent being defined by the FUN in the CFU being IN or INOUT, or SAVE specified in the DCB. WRTF cecrements TMDCRM/TMDPRM for temporary file granules, in which case TMPDCPK/TMPDPPK is updated appropriately.

After the counters are decremented a check is made to see if the counter has turned negative; if so, the appropriate bit is set into J:ASSIGN in the JIT to indicate the type of maximum exceeded (X'10' if permanent, X'20' if temporary) and X'04' set into byte 0 of J:RNST. The user will be aborted at IOSPRTN in CALPROC while exiting the CAL.

Simple Accumulation

The following JIT cells are maintained by the monitor as follows:

J:JIT + CIC:	Initially zero, incremented in IORT following each successful read of the C device.
+ CPO:	Initially zero, incremented in IORT each time a card image is punched.
+ MPO:	Established during LOGON, compared with CPO in IORT each time CPO is incremented to check for max.
+ CPPO:	Initially zero, incremented in WRTD for each page output by a processor, as determined by PUF in J:RNST.
+ MPPO:	Established during LOGON, compared with CPPO in IORT at PMAX to check for max.
+ CUPO:	Initially zero, incremented in WRTD for each page output by a user program.
+ MUPO:	Established during LOGON, compared with CUPO in IORT at PMAX to check for max.
+ CDPO:	Initially zero, incremented in WRTD for each page output through M:DO.
+ MDPO:	Established during LOGON, compared with CDPO in IORT at PMAX to check for max.
JB:JIT+ BANSVT:	Initially zero, incremented in OPNTP after a tape is successfully mounted.
+ BACNST:	Initially zero, incremented in OPNTP before a scratch tape is mounted.
+ BAMNST:	Established during LOGON, compared with BACNST in OPNTP after BACNST is incremented.
J:CALCNT:	Initially zero, incremented each time CALPROC is entered.
J:INTER:	Initially zero, incremented in COC each time a read is completed from a user's console.
J:JIT + NDRW:	Initially zero, incremented in IOQ for each I/O operation to RAD/PACK.
+ NDTW:	Initially zero, incremented in IOQ for each non-RAD I/O operation.

Setting Initial Limits

Limits in the user JIT are initially set up during SWAPIN in SS when the JIT is brought into core for the first time. They are established as follows:

Batch	On-line		JIT Cell
SL:BTIME	SL:OTIME (* 30000)	=	MRT
CL:BLO	SL:OLO	=	MPPO
SL:BPO	SL:OPO	=	MPO
SL:BDO	SL:ODO	=	MDPO
SL:BUO	SL:OUO	=	MUPO
SL:BT	SL:OT	=	MNST
SL:BPS	SL:OPS	=	PRDCRM/PRDPRM
SL:BTS	SL:OTS	=	TMDCRM/TMDPRM

CCI and LOGON (Section PA and PC) alter PRDCRM/PRDPRM during the logon process. The record from the :USERS file contains the SUPER-specified granule limits and the amount used by that account/name to date. If limit is zero, the JIT default (from SL:OPS or SL:BPS) is used. The algorithm used for adjusting the max disc available is:

LIMIT - Accumulated Usage ⟶ Space Remaining

JB:JIT + BAMNST is also obtained from the :USERS file record at this time.

Upon processing the !LIMIT card, CCI will override JIT maximums as follows:

Option	JIT Cell
TIME	MRT
LO	MPPO
PO	MPO
DO	MDPO
UO	MUPO
TSTORE	TMDCRM/TMDPRM
PSTORE	PRDCRM/PRDPRM
SCRATCH	MNST

Production of the :ACCTLG Record

The routine ACCTSUM produces the accounting record for the :ACCTLG file during LOGOFF. At LOGON time of each job, items such as the date, time, permanent remaining RAD and DBK Pack space are saved in the Assign-Merge Table. This information is (manipulated, if necessary) then inserted into the accounting record. The dynamic items in the record are mostly obtained from the JIT. They are as follows:

1. Start date and time: From Assign-Merge Table
   Finish date and time: Via M:TIME CAL at LOGOFF
2. Processor Execution Time: J:PTIME
3. Processor Service Time: J:PTIME+1
4. User Execution Time: J:UTIME
5. User Service Time: J:UTIME+1
6. CPU Time * Core: (J:UTIME+2) + (J:PTIME+2)
7. Number of cards read: J:JIT+CIC
8. Number of cards punched: J:JIT+CPO
9. Number of processor pages printed: J:JIT+CPPO
10. Number of user pages printed: J:JIT+CUPO
11. Number of diagnostic pages printed: J:JIT+CDPO
12. I/O CALs: J:CALCNT
13. Physical tape accesses: J:JIT+TPACCESS
14. Physical RAD accesses: J:JIT+DCACCESS
15. Physical DISK accesses: J:JIT+ DPACCESS
16. Tapes Mounted: JB:TMTS
17. Tape drives allocated: sum of JB:M9T and JB:M9T
18. Number of save tapes used: J:JIT+NSVT
19. Number of scratch tapes used: J:JIT + CNST
20. Packs mounted: JB:PMTS
21. Number of spindles allocated: JB:MSP
22. PEAK core size: JB:PEAK
23. PEAK temporary RAD granules used: J:JIT+TMPDCPK
24. PEAK temporary DISK PACK granules used: J:JIT+TMPDPPK
25. Permanent RAD granules used: PRDCRM at LOGON-PRDCRM at LOGOFF
26. Permanent DISK PACK granules used: PRDPRM at LOGON - PRDPRM at LOGOFF
    a positive value indicates the number of granules used.
    a negative number indicates the number of granules released.

ACCTSUM also updates the granule accounting in the record in the :USERS file in the same manner.

ID

LNKTRC - Load and Link

PURPOSE

The purpose of LNKTRC is to process the load and link (LDLNK) and the load and transfer control (LDTRC) CALs. LDLNK saves the user's program and transfers to a specified program. LDTRC restores and returns to the saved program. Common pages are not released and may be used to transmit data between the programs.

OVERVIEW

In order to release the blocking buffers from the DCBs back to the pool, M:TRUNC CALs are executed specifying each of the user's DCBs. Tests are made to insure that it is valid to LDLNK from the calling program. It then creates a 'n' Star file into which it writes a record of JIT information, a record of DCBs, and a record consisting of the rest of the user's program. It releases core and builds an exit environment in the temp stack consisting of the FPT information (name, etc) of the load module being called. It then exits to T:ASP which brings the load module in and executes it.

LDTRC releases blocking buffers (M:TRUNC) and insures that it is valid to LDTRC from the calling program. It then closes the DCBs and releases core. If the requested name is a saved 'n' Star file, it is restored in memory and the file released. It sets the return PSD in the stack. It associates a core library, if necessary, and if a shared processor (APR) must be restored, it sets up an environment and drives to T:ASP. Otherwise it returns to the LDLNK CAL+1.

At exit time, STEP recognizes from the N in the low byte of J:RNST that LDLNKs were performed. LEXIT, the load and link exit cleanup routine, is called. It releases core, reads those Star files that still exist and closes their DCBs and releases the Star files. It then returns to STEP.

USAGE

For LDLNK and LDTRC

register 6	=	lst word of FPT
7	=	address of word 1 of FPT
8	=	op code from first byte of FPT (02 for LDLNK and 03 for LDTRC)

For LEXIT

register 8	=	*, N (1, 31) where N is one less than the number of load and link files created (from J:RNST).

OUTPUT

LDLNK: Register 8 contains name of the file the program was saved under when LDLNKing to another program.

DATA BASES

The first record of the 'n' Star file which contains JIT information is as follows:

word	0	J:INTENT
	1	J:TIMENT
	2	J:UTIMER
	3	J:USENT
	4	J:TCB
	5	J:TREE
	6	PSD
	7	PSD
	8	# of DCB pages
	9	J:PLL
	10	J:PUL
	11	J:DLL
	12	J:DUL
	13	J:DDLL
	14	J:DDUL
	15	APR, APO, ASP, DB
	16	User's Flags
	17	# of DCB pages
	18	J:PLL
	19	pure procedure size (PUL-PLL+1)
	20	J:DLL
	21	data size (DUL-DLL+1)
	22	JB:TDP
	23	dynamic data size (TDP-DDLL)
	24	program size (TDP-DLL)

Normally the second record contains the DCBs and the third contains the rest of the user's program. Exceptions to this are 1) there may be no DCB record, and 2) a shared processor is associated in which case, if they are present, data is in one record and his dynamic data is in the next.

SUBROUTINES

VALID determines whether LDLNK or LDTRC functions are allowed. The following are the restrictions tested for:

1) A debugger must not be associated (UB:DB=0).

2) If a shared processor is associated it must be a core library (bit 3 of $P:SA_{(UB:ASP)}$).

3) Data and pure procedure parts of the program must be able to be saved with one write. This is tested for by insuring that:

   a) If pure procedure count, JB:PCP = 0 (insures that program not created by overlay loader) then there must be a shared processor associated, i.e. UB:APR $\neq$ 0 or UB:ASP $\neq$ 0.

   b) If pure procedure count = 0, then dynamic data must be above pure procedure (DDLL-1 = PUL) indicating that the program was created by the overlay loader and can be saved.

   c) Program must not have executed virtual page CALs in the dynamic data area (DDUL-BCP)+(TDP-DDLL)=PCDD to perform LDLNK.

   d) Program must not have executed virtual page CALs in data area (JB:CMAP from DLL to DUL must not contain FPMC, free page map constants).

MOVEFPT gets a blocking buffer from J:FPOOL, sets bit 2 of J:CFLGS to indicate the buffer was obtained, and moves the user's environment and the CAL's FPT to the buffer beginning at word X'100' in the buffer.

RELBUF insures that a blocking buffer was obtained (bit 2 of J:CFLGS) and that it is valid, i.e., in the context area. It insures that the stack is empty and then moves the user's environment back to the stack. The buffer is released to J:FPOOL.

CLOSE closes all of the user's DCBs.

CLOSEREL closes and releases all of the user's DCBs.

TRUNK truncates all of the user's DCBs. In the general loop used by this and the above 2 routines: register 2 = -1 indicates truncate, = 0 indicates close and = 1 indicates close and release.

PROCFLG sets into regisrer 6 the numbers of the shared processors associated with the user (the contents of the user's UB:APR, UB:APO, UB:DB and UB:ASP) and into register 7 the contents of UB:FLG.

RELCORE releases any associated processors by decrementing their usage count and zeroing the appropriate user tables (UB:ASP, UB:APR, UB:APO, UB:DB). In the case of roots and overlays of shared processors (UB:APR and UB:APO) the free page map constant (FPMC) is set into the user's Map image in JB:CMAP. It then releases all user core except Common core, zeros J:DCBLINK, and sets page number of the beginning of the user's program, BPU, into JB:TDP, J:DLL, J:PLL, J:DDLL and BUP-1 to J:DUL and J:PUL.

ABORT is the routine that handles errors by exiting to T:ABORTM in STEP. It releases the blocking buffer to J:FPOOL if one is present. LEXABORT is an entry point for LEXIT for use when I/O errors occur. It does not release the blocking buffer.

ERRORS

00B501 Occurs when LNKTRC can't open the requested file due to insufficient information.
00B503 Occurs when LNKTRC tries to open a non-existent file.
00B514 I/O error indicating LNKTRC denied access to file.
00B546 Insufficient information to open the file.
00B561 Not permitted to LDLNK or LDTRC while running under a debugger (condition 1 in VALID subroutine).
00B562 Special processor other than a core library is associated.
00B563 Program created by LINK (Condition 3a and 3b in VALID).
00B564 Program doesn't own all virtual core from data through dynamic data (Condition 3c and 3d in VALID).
00B565 DCB address in DCB name table not valid (TRUNK).
00B566 Couldn't find blocking buffer (MOVEFPT) or trying to release invalid one (RELBUF).
00B567 Trying to perform exit cleanup when LDLNKs were never performed (LEXIT).
00B568 Illegal information encountered in transfer file.

DESCRIPTION

LDLNK (entry point is LNK0) calls TRUNK which truncates the user's DCBs, calls MOVEFPT which gets a blocking buffer and moves the user's FPT into it, and calls VALID which insures that it is valid to perform a LDLNK. It closes M:XX, opens it sequential to the next 'n' Star file and increments n in the low order byte of J:RNST. It sets up the JIT information record, as specified in the Data Base section, and writes it to the file. It then writes the DCBs if any. If there is no shared processor (UB:APR) associated, all virtual memory is obtained between data, procedure, and dynamic data to make it contiguous, then the rest of the user's program (data, pure procedure and dynamic) except for common is written. If a shared processor is associated, it writes data if any and then dynamic data if any. The file is closed and RELCORE releases user processors and core. The exit routine, L10LNK, is called to set up an exit environment for T:ASP in the STEP module which gets the requested program.

LDTRC (entry point is LDT0) also first calls on TRUNK, MOVEFPT and VALID and closes M:XX. It then calls on CLOSE to close the user's DCBs and RELCORE to release processors and user core. If the requested name is not in the form of a Star file with a valid 'n', LDTRC immediately calls the exit routine, L10LNK, to set up the file name, etc. in an exit environment and go to T:ASP which in turn gets the program and goes to it.

If the name requests an 'n' Star file, it is opened. The JIT information record is read

into the blocking buffer and the first 6 values in it and the program area limits are restored to JIT after checking the values obtained to make sure they are reasonable. If there is a DCB record, a page(s) is obtained and the DCBs read. If the JIT information indicates no shared processor, APR, to associate, the rest of the pages are obtained and the last record read in. Otherwise, the data pages are obtained, the data record read, the dynamic data pages obtained and that record read. Before pure procedure is read (if no APR), the count of the number of pure procedure pages (JB:PCP) is zeroed. After it is read the "pure procedure must be swapped", PPSWP, flag in UH:FLG is set and the count restored. This keeps the swapper out of trouble. JB:TDP, top dynamic page, is restored from the JIT information. The file is closed and released and the PSD moved from the blocking buffer to the user's environment. If a core library must be restored, its number is set in UB:ASP, the Ready-to-Run flag reset in UH:FLG and an associate processor event, E:AP, reported to T:REG. The system returns when the processor has been associated. If there is no APR to reassociate, RELBUF is called to release the blocking buffer, J:TCB is set into register 0 of the user's environment in TSTACK, the slave bit is ORed into the user's environment to maintain security, J:DCBLINK is set up and a n exit CAL executed. If an APR must be restored, the registers are set up, the associated processor overlay APO, number is set into byte one of J:CFLGS, bit 0 in J:CFLGS is set, and the exit routine L12 is called which sets up TSTACK and goes to T:ASP. Bit 0 of J:CFLGS indicates to STEP the few special operations it must do (or ignore) for LDTRC such as not setting up the start address in the PSD since we are returning to the location already set by LDTRC.

L10LNK and L12 are the 2 entry points for the routine that LDLNK and LDTRC use to exit. L10LNK sets bit 1 of J:CFLGS to indicate to STEP that CDLNK or LDTRC is going to a new program rather than restoring an old one and sets the name, account, and password into registers. L10LNK then transfers to L12. L12 calls RELBUF to release the blocking buffer. A second environment is set in TSTACK, the registers (containing name etc.) are set into the stack and a self-destruct (T:SELF DESTRUCT with register 11 = T:ASP) to T:ASP accomplished.

LEXIT is called by STEP when 'n' is not zero in J:RNST at job step. It closes and releases any LDLNK files and closes the DCBs in them. Upon entry, LEXIT first checks that 'n' is not zero. All user pages are released except context and two DCB pages are obtained if not already present. A blocking buffer is obtained from J:FPOOL and the M:XX DCB closed. An 'n' Star file (first where n = 0) is opened. If the file was not previously released by performing a LDTRC, the JIT information record is read into the blocking buffer to obtain the number of pages in the DCB record. The DCB record is read, if present, and the DCBs closed by calling CLOSE. This 'n' Star file is closed and released and this process continued until all files have been read, DCBs closed and 'n' Star files released. The buffer is returned to J:FPOOL, the DCB pages released and control returned to STEP through T:SELFDESTRUCT.