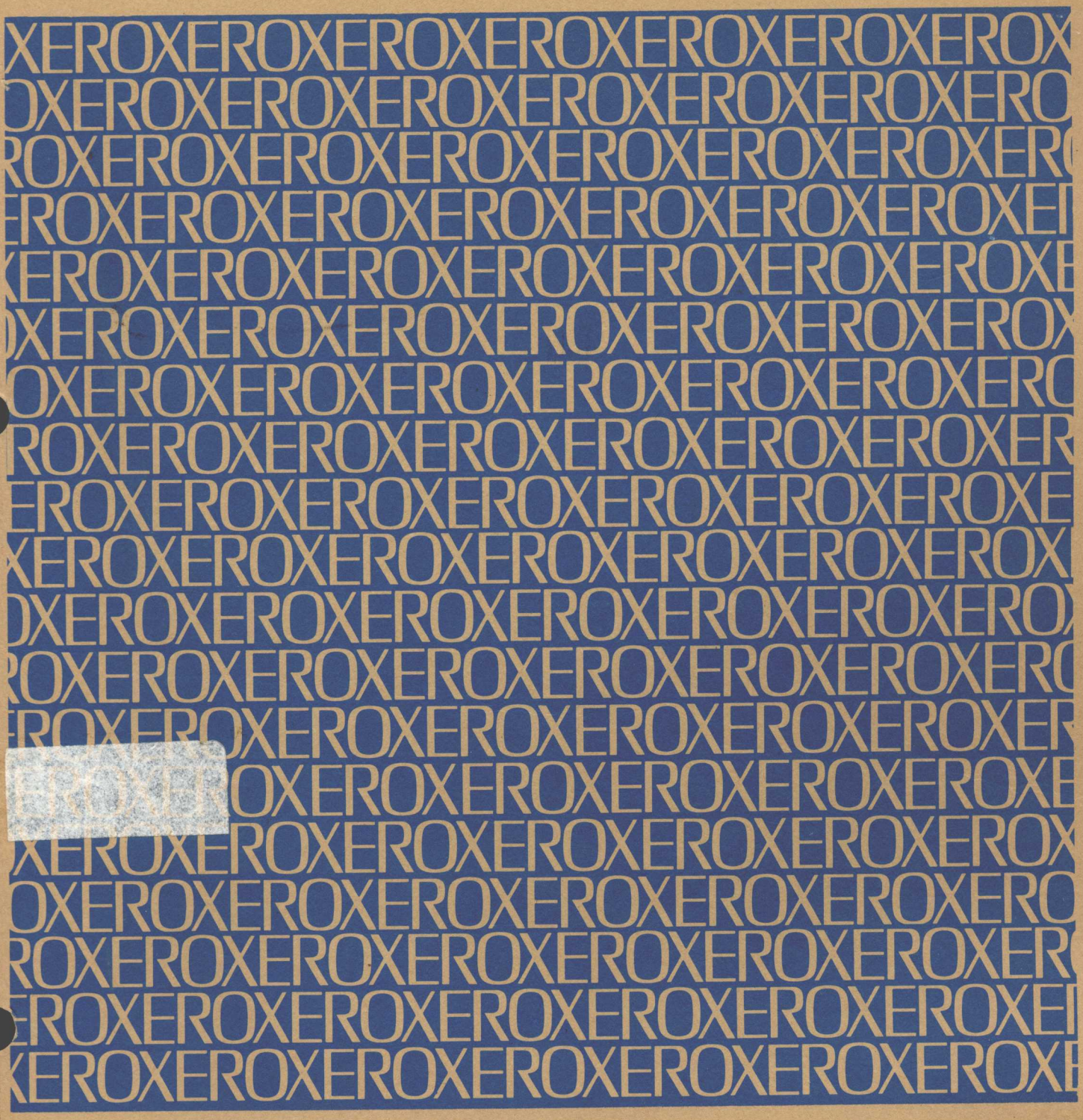
A1

Xerox Universal Time-Sharing System (UTS)

Sigma 6/7/9 Computers

System Processors
Technical Manual



90 19 94A

Xerox Corporation
701 South Aviation Boulevard
Segundo, California 90245
3 679-4511

XEROX

Xerox Universal Time-Sharing System (UTS)

Sigma 6/7/9 Computers

System Processors

Technical Manual

FIRST EDITION

90 19 94A

February, 1973

Price: $4.75

Printed in U.S.A.

NOTICE

This publication documents the system processors 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 Operator Communication and Monitor Services Technical Manual	90 19 88
UTS File Management Technical Manual[†]	90 19 89
UTS Reliability and Maintainability Technical Manual	90 19 90
UTS Interrupt Driven Tasks Technical Manual[†]	90 19 91
UTS Initialization and Recovery Technical Manual	90 19 92
UTS Command Processors Technical Manual	90 19 93
UTS Data Bases Technical Manual	90 19 95

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

CONTENTS

RATES – Accounting Rates Structures ____ 1
- Purpose ____ 1
- Usage ____ 1
- Interactions ____ 1
- Output ____ 1
- Subroutines ____ 2
- Description ____ 2

SUPER – Authorize Users ____ 4
- Purpose ____ 4
- Usage ____ 4
- Output ____ 4
- Interaction ____ 4
- Data Bases ____ 5
- Subroutines ____ 5
- Error Messages ____ 6
- Description ____ 9
  - Initialization Section ____ 9
  - Command Interpretation Section ____ 9

UTSPM – UTS Performance Monitor ____ 12
- Purpose ____ 12
- Reference ____ 12
- Usage ____ 12
- Relationship of UTSPM and SUMMARY ____ 13
- Interactions ____ 14
- Data Bases ____ 14
- Data Tables ____ 16
- Subroutines ____ 16
- Adding an Item to a Display Group ____ 18
- Record Format for History File ____ 18
- Record Format for SNAPSHOT and SSNAPSHOT Files ____ 19
- Error Handling ____ 20
- Restrictions ____ 20
- Command Handlers ____ 20

SUMMARY – UTS Performance Summary ____ 21
- Purpose ____ 21
- Reference ____ 21
- Program Subroutines ____ 21
- Using M:SI, M:CO, M:DO ____ 24
- Data Tables ____ 24

LINK – UTS On-Line One-Pass Loader ____ 26
- Purpose ____ 26
- Restrictions ____ 26
- Description ____ 26
- Data Bases ____ 28
  - Stacks ____ 28
  - Symbol Tables ____ 34
- Output ____ 35
  - Load Module File ____ 36
    - HEAD ____ 36
    - TREE ____ 37
- Virtual Core Layout During Load ____ 38
- Virtual Core Layout During Run Time ____ 39
- Flowchart ____ 40

Subroutines 47
ADVNEFLE -Advance to Next Input File 47
BINTOHEX - Convert Binary to Hex (EBCDIC) 48
BIN2BCD - Convert Binary to Decimal (EBCDIC) 49
CHKDECLD - Search for Declaration 50
CHKFREFD - Search Forward Reference Stack 51
CHK2EOF - Check for Two Consecutive End of Files 52
COREABN - Abnormal Return Opening a Core Library 53
ENDECL - Enter New Entry into Declaration Stack 54
ENNAM - Search and Enter New Entries in the REF/DEF Stack 55
ENSYMTBL - Put an Entry in Symbol Table 56
EVEXP - Evaluate Designated Expression 57
EXPRIN - Read and Build an Expression 58
EXPTTB - Put Expression into Expression Stack 59
GBYTE - Get a ROM Control Byte 60
INCLOC - Increment Load Location Counter 61
CHKLOC - Check Load Location Counter Limits 61
INSERTLB - Insert Core Library Name 63
INTRNAME - Print Symbol Table Name 64
LINKERR - LINK Command Error 65
LINKERR2 - Print 'CANT FIND – RETYPE' Message 66
MMDA - Get Memory for Data Area 67
MMDCB - Get Memory for DCB Area 68
MMPP - Get Memory for Pure Procedure Area 69
MMSYMBTB - Get Memory for Symbol Tables and Core Library REF/DEF Stack 70
MMTBASE - Get Memory for the Stacks 71
PRESSTK - Remove Space from Stacks 72
PRINTQ - Print Eh ? @ n Message 73
RAMRABN - Abnormal Return Reading ASSIGN/MERGE Record 74
READBILI - Read a ROM Card 75
REFSATY - Satisfy REFs with Zero 76
RELSYM - Release Memory from Symbol Table Area 77
RFDFCHN - Define a REF 78
SCAN - LINK/RUN Command Scan 79
STKOVF - Stack Overflow Handler 80
SYSLBABN - Abnormal Return Opening SYS Library 81
2BNUM - Get 2 ROM Control Bytes 82
3BNUM - Get 3 ROM Control Bytes 83
12BNUM - Get 1 or 2 Bytes of ROM Input 84

BATCH – Terminal Batch Entry Subsystem 85
Purpose 85
Usage 85
Input 85
Output 85
Interactions 85
Errors 86
Restrictions 87
Data Bases 88
Subroutines 88
Description 88
Flowchart 90

DEFCOM – Create Load Module Containing DEFs Only 91
Purpose 91
Usage 91
Input 91
Output 91
Interaction 91
Data Bases 92
Subroutines 92

Errors 92
Restrictions 92
Description 92

SYMCON – Symbol Table Control 94
Purpose 94
Usage 94
SYMCON Commands 96
LIST 96
DELETE 96
KEEP 96
CHANGE 96
BUILD 96
DISCARD 96
END 96
Output 97
Interaction 97
Data Bases 98
Subroutines 98
Error Messages 99
Restrictions 102
Description 102
DELETE 103
KEEP 104
CHANGE 104
LIST 104
BUILD 104
DISCARD 106
END 106
Flowchart 107

ERRMWR – Error Message File Writer 108
Purpose 108
Usage 108
Input 108
Output 108
Interactions 108
Errors 109
Description 109

MAILBOX – System Messages to Users 110
Purpose 110
Usage 110
Input 110
Output 110
Interaction 111
Data Base 111
Subroutines 111
Restrictions 111
Description 112

SYSTEM UTS
Purpose 113
Usage 113
Content 113
UTSPROC 113
S69PROC 114
ANSPROC 114
MONPROC 114

Description ____ 114
BUMP, PUSH,PULL – Modify TSTACK ____ 114
ENABLE, DISABLE – Reset and Set Interrupt Inhibit Bits ____ 115
HD, BD – Generate Halfword or Byte Addresses ____ 115
OVERLAY, OVERTO – Call Monitor Overlay and Enter it ____ 115
REMEMBER – Record Current Overlay Number and Contents of Register 11 ____ 115
DESTRUCT – Generate Call Sequence to T:SELFDESTRUCT to Delete User's Current Monitor Overlay ____ 116
MAP, UNMAP – Turn Map On or Off ____ 116
OB, OBAL, OBSR4 – Implement Segloading ____ 116
SCREECH – Load Software Check Code Number and Call RECOVER ____ 116
SET, RSET – Set and Reset User Flags in UH:FLG ____ 116
T:PUSHE – Save an Interrupted Environment ____ 117
WORTAB, HAFTAB, BYTAB – Generate Word, Halfword, and Byte Tables ____ 117
LOAD, STORE, MODTST, COMPARE – Generate Byte or Halfword Type Instructions to Access Data ____ 117
LDMAP – Load Memory Map ____ 118
BIL, BOL – Branch if Effective Word was Within or Outside Limits ____ 118
ANSB, ANSNB, ANSNOP, ANSBAL – Generate Conditional Branches Depending on Whether System is ANS or Not ____ 118
ERRABNCD – Produce Data Word for Generating Error Messages ____ 118-1
DAYS – Generate a Word Table of Julian Calendar Dates of First Day of Each Month ____ 118-1
SXP – Convert EBCDIC Character String Into a 32-Bit Binary Quantity ____ 118-1
CFU, DCB, Master Index – Define Major Fields by an EQU Directive ____ 118-2
CLEAR – Zero Registers 8 and 10 ____ 118-2
LIF – Generate a Table as Part of Branch Vector Mechanism ____ 118-2

SYSTEM BPM (For UTS) ____ 119
Purpose ____ 119
Reference ____ 119
Usage ____ 119
Error Messages ____ 119

ID

RATES - accounting rates structures

PURPOSE

The function of the RATES processor is to create and maintain the :RATE file. The :RATE file contains the charge rate structures, which determine how much each user is assessed for various services (e.g. CPU TIME). A charge rate structure specifies how many charge units a user will be assessed for each unit of each type of service.

USAGE

The RATES subsystem can be accessed only by a user with the name LBE and the account :SYS. It is called on-line from TEL.

!RATES (RET)

Complete description of usage can be found in UTS System Management Guide, Chapter 4.

INTERACTIONS - (CALs and FPTs are handwritten)

M:PC	Set prompt character to - (for command) or = (for value).
M:WAIT	Dismiss program for specified interval (=1 second). Used to dismiss program while waiting for the :RATE file to become not busy.
M:INT	Set break control.
M:TRNT	Return from break control.
M:READ	Read input from terminal through M:UC.
M:WRITE	Write output to terminal through M:UC.
M:OPEN	Open :RATE file.
M:READ	Read :RATE file through M:EI.
M:WRITE	Write :RATE file through M:EI.
M:CLOSE	Close :RATE file.

OUTPUT

The primary output of the program is a modified :RATE file. The structure and the default values of the :RATE file are described in section IC. If the :RATE file does not exist when RATES is entered, the charge rate structures in the :RATE file are

assigned their default values.

The format of the printed output resulting from the PRINT command and the error messages are shown in the UTS System Management Guide.

SUBROUTINES

SINPUT:	Input text from terminal
SOUTPUT:	Output TEXTC string plus new line to terminal
SOUTPUP:	Output TEXTC string to terminal
SOPEN:	Open :RATE file. The :RATE file is created if it does not already exist
STOBIN:	Convert decimal to binary
STODEC:	Convert binary to decimal

DESCRIPTION

1. Break Control (BBREAK)
   Breaks come here once break control has been set at the end of Initialization. When a break occurs the monitor pushes the user's registers and PSD into the user's TCB stack.

   If an END command has already been issued, an M:TRNT is issued which pulls the TCB stack and returns control to where it was interrupted. In other words the break is ignored as far as the user is concerned.

   Otherwise the address of the main command loop (MLOOP) is inserted into the first word of the PSD in the TCB stack before the M:TRNT is issued. This results in control being transferred to the main command loop.

2. Initialization (ISTART)
   Program execution begins here. The Initialization:
   a. issues the identification message (RATES HERE),
   b. checks to see if the user has the correct name (LBE) and account (:SYS), and
   c. reads in the :RATE file.

3. Main Command Loop (MLOOP)
   The main command loop identifies the command and transfers control to the appropriate command handler.

4. BUILD Command Handler (CBUILD)
   Code common with the PRINT command checks to see that the last character is

a digit 0-7. If it is, the first address (in the :RATE file buffer) of the specified charge rate structure is calculated and control then goes to logic for PRINTing (CPRINT1) or BUILDing (CBUILD1).

Values are prompted for one at a time. Each value that is typed in is converted to binary and stored in the charge rate structure. If a lone carriage return is typed in for a value, the value is left unchanged.

When there are no more values to prompt for, control goes to the main command loop (MLOOP).

5. PRINT Command Handler (CPRINT)
First, the common code described in the BUILD command handler is executed. The sidings and values for the specified charge rate structure are output one at a time. When there are no more, control goes to the main command loop (MLOOP).

6. End Command Handler (CEND)
A flag is set so that breaks will be ignored. The :RATE file is written out to the RAD and control goes to the Finalization.

7. Finalization (FEXIT, FEXITE)
On a normal exit (FEXIT), the finalization prints the message 'RATES PROCESSING TERMINATED' and exits to the terminal executive.

On an I/O error (FEXITE), the finalization first issues the messages 'UNEXPECTED I/O ERROR' and 'NO UPDATING TOOK PLACE'.

ID

SUPER - Authorize Users

PURPOSE

The main purpose of SUPER is to create, modify, remove, and list the records of the file :USERS, the logon file of authorized UTS users. SUPER is also used to create, delete, and list the records of the file :RBLOG, the logon file of authorized work stations for remote batch.

USAGE

SUPER is accessible on-line and in a batch job stream. The detailed description of the usage can be found in the UTS System Management Guide, Chapter 3.

OUTPUT

SUPER creates records in the :USERS file and in the :RBLOG file. Section VN.01 describes the :USERS file record format and contents; Section VN.04 describes the :RBLOG file record format and contents. SUPER reads and writes the :USERS file through the F:EI DCB; it reads and writes the :RBLOG file through the M:EO DCB.

On-line, SUPER writes messages and LIST output to the terminal through the M:UC DCB. In batch jobs, SUPER writes messages and LIST output through the M:LO DCB.

INTERACTION

SUPER is a processor which is called by TEL and CCI. SUPER uses the following monitor services:

M:OPEN	To open the :USERS file, the :RBLOG file, and for batch, the M:LO device.
M:READ	To read records from the :USERS file or the :RBLOG file, input from the terminals, and for batch jobs, input from the M:SI device.
M:WRITE	To write records to the :USERS file or to the :RBLOG file, and to output text to the terminal and the M:LO device.
M:DELREC	To delete a record from the :USERS file or the :RBLOG file.
M:CLOSE	To close the :USERS file, the :RBLOG file, and the M:LO device.
M:EXIT	To return control from SUPER to TEL or to CCI.
M:WAIT	To put SUPER to sleep for one-second intervals while the :USERS file is busy and SUPER wishes to access it.
M:DEVICE	To advance the M:LO device to a new page.

DATA BASES

SUPER uses three internal tables. The first is a byte table labeled CMNDS used in determining what command was input. The second is a word table labeled OPTIONS used in deciding which options were selected. The third is a byte table called TERMS which contains characters SUPER regards as illegal when incorporated into a name or an account field of a CREATE, MODIFY, DEFAULT, REMOVE, or WORKSTATION command, or in the value field of a PASSWORD option for a CREATE or MODIFY command.

SUPER reads the JIT words J:JIT, J:ACCN, and to determine whether the job was called from on-line or batch, and to verify the calling user is authorized to use SUPER (i.e., is logged on as :SYS, LBE).

Section VN.01 describes the record format of the :USERS file, which SUPER interrogates and updates. Section VN.04 describes the record format of the :RBLOG file.

SUBROUTINES

SUPER uses the following subroutines:

ADDBCERR	Updates batch command error counter.
ADDBUERR	Updates batch :USERS file I/O error counter.
BCD2BIN	Converts an EBCDIC number to a binary number.
BIN2BCD	Converts a binary number to its printable decimal EBCDIC value.
BIN2HEX	Converts a binary number to its printable hexadecimal EBCDIC value.
BLK2	Suppresses (blanks) leading zeroes in the number resulting from a BIN2BCD subroutine conversion.
CMOPTS	Handles the options for the CREATE, DEFAULT, MODIFY, and WORKSTATION commands. CMOPTS prompts for and reads the options, determines which options were selected, and stores the input option values in the appropriate fields in the :USERS or :RBLOG record buffer, INBUFF.
HEX2BIN	Converts a hexadecimal EBCDIC number to its binary value.
NAMEACT	Scans the command buffer for the name and account fields of the CREATE, MODIFY, and REMOVE commands. After obtaining the name and account, NAMEACT forms a key which SUPER uses to access the associated records in the :USERS file
OPEN	Opens the :USERS or :RBLOG file.
PRNTRT	The print routine for the LIST command. PRNTRT retrieves the selected LIST option fields from the :USERS file record(s), forms them for printout, and outputs them in line format to the terminal or M:LO device via the PRTREC subroutine.

PRTREC	Writes a line of text for the LIST command output to the terminal or M:LO device.
SCAN	Extracts the next field from the input command. Field terminators are determined by the contents of table TERMS. Leading and trailing blanks are suppressed as well as serving as terminators.
TEXTSUB	Prepares messages in TEXT format for output to the terminal or M:LO device.
TXCSUB	Prepares messages in TEXTC format for output to the terminal or M:LO device.

ERRORS

The following table lists the error messages and describes the circumstances under which SUPER outputs them. On-line jobs receive the messages at terminals, batch jobs on the M:LO device.

Message	Description
ACCOUNT, NAME?	SUPER scanned to the end of the command buffer without detecting an account and/or name field for a CREATE, MODIFY, or REMOVE command.
CALL ACCOUNT?	SUPER did not detect a delimiter (space, carriage return, period, nor semicolon) for a CALL lmn value.
WHO?	Request entered to MODIFY, REMOVE, or LIST the record(s) of a user(s) not in the :USERS file.
MODIFY?	Request entered to CREATE a user (record) which already exists in the :USERS file. SUPER issues this message, then (prompts and) awaits input of options to modify the existing record. Null option input leaves the record as it was.
LOST AN OPTION: ADVISE LISTING USER	This message should never be issued; however, should it be, it indicates a SUPER program failure, or a loss of bits in the system. SUPER outputs the message and continues processing any remaining options in the buffer.
ERROR ON M:SI DEVICE, SUPER EXITING	Abnormal return other than EOF or EOD on the batch M:SI device.

Message	Description
NOT MAIN COMMAND	Batch SUPER expected to read a command beginning in column one, but column one of the card was null.
ABNORMAL RETURN ON :USERS FILE -- value, value	Abnormal return other than "busy", "wrong key", or "file nonexistent" for I/O access of :USERS file. First value is hex abnormal code; second value is hex abnormal subcode.
ERROR RETURN ON :USERS FILE -- value, value	Error return other than "wrong key" for I/O access of :USERS file. First value is hex error code; second value is hex error subcode.
SORRY YOU ARE NOT ALLOWED TO ACCESS SUPER	User trying to access SUPER is not logged on as :SYS, LBE.
=value?	The given option value is too long, exceeds system limit for the option, or contains an illegal character. Input the option again.
value?	SUPER does not recognize the given command operator or option operator, or does recognize a too lengthy CALL field value. Input the information again.
THE :USERS FILE DOES NOT EXIST	This message should never be issued, however, should it be, it indicates SUPER, in accessing the :USERS file, received an abnormal return indicating :USERS does not exist. SUPER outputs the message and exits to executive level (TEL or CCI). Call SUPER again.
INVALID WORK STATION ID	The specified work station ID is not an alphanumeric string of 1-8 characters with at least one alphabetic character.
OPTION VALID ONLY FOR WORK STATION COMMAND	A work station option has been used on a CREATE, MODIFY, or DEFAULT command.

Message	Description
WORK STATION NOT PRESENT	An attempt was made to delete a non-existent work station.
UNABLE TO OPEN :RBLOG FILE	The :RBLOG file exists but cannot be accessed by the user.
NO MORE RBIDS	An attempt was made to authorize more than 255 IDs for remote batch.

At the end of the batch SUPER run, two messages summarizing the number of errors are printed. They have the following format:

n COMMAND ERROR n USER FILE I/O ERRORS

The first n indicates the number of syntax errors found in the input stream; the second n indicates the number of abnormal conditions found in accessing the :USERS file. In general, a syntax error found in an option for a main command will result in the command being executed as though the particular option were not present.

RESTRICTIONS

To use SUPER, the user logs in as :SYS,LBE. SUPER is not a shared processor; each user has his own copy of SUPER residing in his virtual memory. However, should more than one person be using SUPER and one have the :USERS file open, the others will experience a pause in response to their commands until the :USERS file is again closed. UTS users trying to log on while :USERS is open will experience a pause until the :USERS file is no longer busy.

DESCRIPTION

SUPER divides logically into the following parts:

- procedures
- messages
- data and storage areas
- option masks and command lists
- I/O parameter lists (fpt's)
- program initialization
- command interpretation
- command execution
- subroutines; I/O error and abnormal returns routines
- patch area

In the initialization section, label START, SUPER first checks the user's JIT and determines whether the calling job is on-line or batch and resets X:JIT if the job is batch. SUPER then looks again at the JIT and ASSIGN-MERGE TABLE to determine whether the user is authorized to access SUPER, i.e., whether the user is logged on as account :SYS. If the user is not logged on under this account-name, SUPER outputs a message via M:UC or M:LO saying the user may not access SUPER, and SUPER exits to the executive level TEL or CCI. Otherwise, processing continues at label RESTART.

The command interpretation section, label RESTART, determines whether there is a command in CMNDBUF. If there is not, SUPER clears the command buffer, reads a command, writes it to the M:LO device, and goes to label RST10. At RST10, SUPER determines what command was issued and branches to the command execution section to process (execute) it. Return from processing any command other than the END command is to label RESTART; the END command exits to TEL or CCI.

The command execution section has division for each of the SUPER command, i.e., major labels LIST, CREATE, MODIFY, DEFAULT, REMOVE, WORKSTATION, FAST, and END.

At label LIST, SUPER first determines what the user wants to list and accordingly sets register 15 to 1, 2, or 3 for listing the entire :USERS file, listing all the records in a single account, or listing up to three specific records in a single account. Next, at label OPTNRT SUPER prompts the on-line user to specify which options (fields in a record) he wants listed. SUPER then reads an option for both batch and on-line jobs, interprets the option and sets a flag for it. SUPER continues to (prompt and) read and interpret options until it receives a null line for on-line jobs, or a new command for batch jobs. Super then continues processing at label DETMLIST where it reads a selected record from the :USERS file into buffer INBUFF, places the selected option values in the print buffer LINE and outputs the listing one line at a time to the M:UC

or M:LO device. When SUPER has completed the listing, it branches back to label RESTART and initializes to read another command. SUPER uses subroutine PRNTRT to print each line of the listing from the LINE buffer.

At label CREATE, SUPER forms a five-word key from the name and account fields of the input CREATE command. Next SUPER initializes the buffer INBUFF where it will build the new record. SUPER forms a key from the CREATE command account and name fields and attempts to read :USERS with the key. If a record with that key does indeed exist, SUPER continues processing under MODIFY, at label MLOOP20. If the record does not exist, SUPER continues processing at CLOOP20 where it prompts the on-line user for option input. For both the on-line and batch user SUPER reads the input option value, converts the value to its prescribed format, and stores it in the appropriate field in INBUFF. When the user signifies he has completed his option input, SUPER writes the record from INBUFF to the :USERS file under the key it previously formed. SUPER closes the :USERS file and returns to label RESTART. SUPER uses subroutine NAMEACT to form the key and subroutine CMOPTS to (prompt and) read, interpret, and store option values.

At label MODIFY, SUPER forms a key from the specified name and account, and using that key it reads the requested record into INBUFF and closes the :USERS file if not in FAST mode. Next the subroutine CMOPTS, SUPER prompts the on-line user to specify which fields he wants to change. After interpreting the option input values and modifying the fields in INBUFF, SUPER reopens the :USERS file if not in FAST mode, writes the modified record in INBUFF to the file, closes the file, if not in FAST mode, and continues processing at label RESTART.

At label REMOVE, SUPER forms a key from the specified name and account and opens the :USERS file, if not in FAST mode with the key. It then deletes the record associated with the key, closes the file if not in FAST mode, and continues processing at label RESTART.

At label END, SUPER determines whether the job is on-line or batch. For on-line jobs SUPER exits immediately to TEL. Before a batch session exits to CCI, SUPER writes a line of the command error summary count and the :USERS file I/O error summary count to the M:LO device, advances to a new page, and closes the M:LOdevice. If the user was in FAST mode, the :USERS file is closed, and if the :RBLOG file is open it is also closed.

At label DEFAULT, SUPER proceeds to process option inputs via subroutine CMOPTS, and with specified values forms an image of the whole LOGON record (excluding the account and name field). This image table is hence used to simplify the usage of CREATE command, if so desired, and will remain prevalent until subsequently modified by another DEFAULT command. The DEFAULT command causes no action with regard to the :USER FILE and control is returned to label RESTART after forming the appropriate image.

At label FAST, SUPER opens the :USERS file and sets a flag for FAST mode. The :USERS file (and :RBLOG file, if needed) will remain open throughout the job.

At label WORKSTA, SUPER sets a switch to indicate a work station command has been received. Subroutine SCAN is called to scan the command. If :RBLOG is not open, it is opened and the record is read whose key is the work station ID in TEXTC format. CMOPTS is called to process the options. If the record does not exist, a record is created with the next available ID in the RBID record. If the user is not in FAST mode, the :RBLOG file is closed and processing continues at label RESTART.

ID

UTSPM - UTS Performance Monitor

PURPOSE

The UTSPM processor performs the following functions

- Display performance statistics on line.
- Create a base file as a time reference record of current statistics.
- Create a history file consisting of consecutive records of statistics created at intervals specified by the user.
- Report current statistics referencing the base file which defines the beginning of the sample interval.
- Report history file statistics by processing one or more history files and creating two snapshot files, SNAPSHOT and SSNAPSHOT.
- Format control by setting, adding and dropping print flags.

REFERENCE

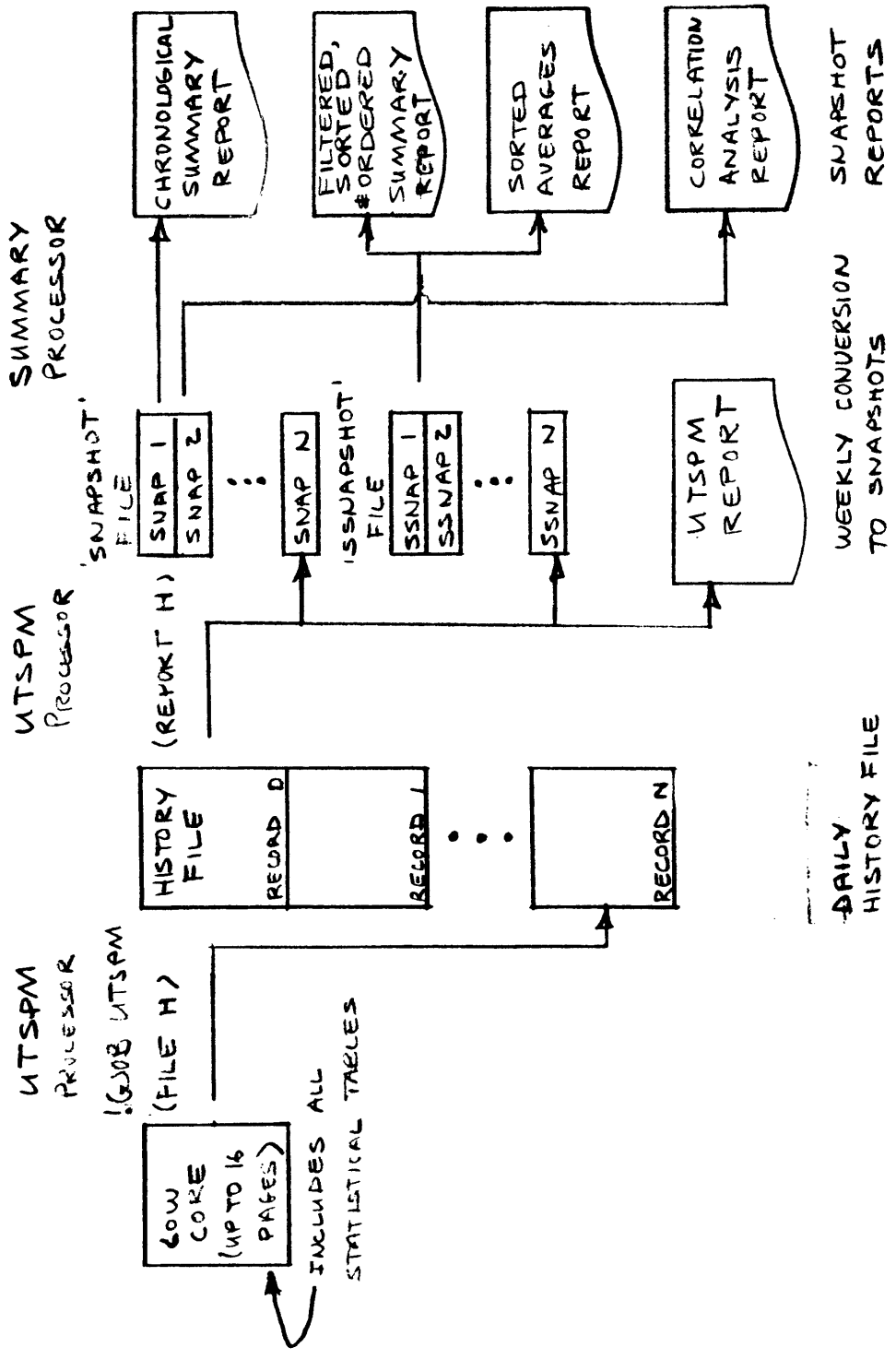
This document presupposes familiarity with the UTSPM section in the UTS/SM Reference Manual (90 16 74) which gives a complete description of UTSPM from a user point of view. The usage, input, and output are adequately described there.

USAGE

UTSPM may be executed as a batch job, an on-line processor, or as a ghost job. The input for a ghost job must be on file GHOSTSI. The output for a ghost job will be put in GHOSTLO and GHOSTDO.

A privilege of '80' is required whenever the desired function requires an access of the monitor data tables. The privilege is not required for other operations such as history file processing. The on-line use of a line printer is useful for large reports. The use of o on-line tape allows the user to copy large history files to tape weekly and to process these tapes on-line at a later time.

The M:SI DCB is used for input of commands and user responses to interactive questions. M:DO is used to correspond with the user. M:LO is used for report output. F:1 is used for history file creation and reading. F:2 and F:3 are used to create or read the 'SNAPSHOT' and the SSNAPSHOT files respectively. The following diagram illustrates the relationship of UTSPM and SUMMARY.


UTSPM PROCESSOR
(JOB UTSPM FILE H)
60W CORE (UP TO 16 PAGES)
INCLUDES ALL STATISTICAL TABLES
HISTORY FILE
RECORD 0
RECORD 1
RECORD N
DAILY HISTORY FILE
UTSPM PROCESSOR
(REPORT H)
'SNAPSHOT' FILE
SNAP 1
SNAP 2
SNAP N
'SSNAPSHOT' FILE
SSNAP 1
SSNAP 2
SSNAP N
WEEKLY CONVERSION TO SNAPSHOTS
UTSPM REPORT
SUMMARY PROCESSOR
CHRONOLOGICAL SUMMARY REPORT
FILTERED, SORTED #ORDERED SUMMARY REPORT
SORTED AVERAGES REPORT
CORRELATION ANALYSIS REPORT
SNAPSHOT REPORTS

INTERACTIONS

M:TIME	To get the time/date in TEXT for reports.
M:WAIT	To sleep while creating history files or to wait if the history file is in use by another user.
M:INT	To get break control.
M:EXIT	To return to TEL.
M:DISPLAY	To get monitor statistics as reported to user.
M:TRAP	To ignore fixed point overflow.
M:PC	To set the prompt characters.
CAL1,8	To map a monitor page.
CAL1,8	To get a virtual page to buffer the monitor data.
CAL1,9	Exit from break control.
CAL1,1	Type an interactive message.
M:READ	To read inputs from the user, to read the history file, to read the snapshot files.
M:PFIL	To position the history file to BOF or EOF.
M:PRECORD	To backspace one history file record.
M:DEVICE	Top of Form and to return the number of lines.
M:WRITE	To print all reports, write the base file record, to write history records, to write snapshot file records.
M:OPEN	To open M:SI, M:DO, M:LO to named files when run as a ghost job, to open the history file or base file to test for existence, to create the history file or base file, to update the history file, to create the snapshot files.
M:CLOSE	To close the base file, the history file, or to close the snapshot files. The SAVE options are important to assure that files are not lost.

DATA BASE (IN MONITOR)

The monitor data consists of the following areas:

1. Performance Measurement Tables

Referenced WORD Tables

C:RT90	C:CTW	C:SCB
C:SC	C:CI	C:MSB
C:TIC	C:CO	C:IDLE
C:TINC	C:RTRW	C:IDLES
C:ETMF	C:CSC	C:SIT
C:CIT	C:COS	C:SRT
C:CITI	C:SCO	C:STT
C:CAL	C:MSO	C:ST
		C:SCI

Referenced HALFWORD Tables

CH:DI1	CH:DIT
CH:DI2	CH:DTT
CH:DI3	CH:DT
CH:DOT	CH:DC
CH:DOS	CH:DLI
CH:DRT	CH:CLO

2. Referenced Processor Tables Constants

PPROCS
SMUIS
P:NAME
PBT:LOCK
EOMTIME
LNOL
BGNPMPRC
ENDPMPRC

3. Referenced User Tables

UH:FLG
UB:PCT
UB:US
UB:APR
UB:ASP
UB:DB

4. Referenced Scheduler Tables

The indices corresponding to all state queues are REF'ed when possible or EQU'ed in the program.

S:OUAIS
S:CUIS
S:BUIS
S:GUIS
SB:CQ
S:CAST

5. Referenced System Tables

SL:TB
SL:UB
SL:QUAN
SL:QMIN
SL:BB
SL:IOC
SL:IOPC
SL:OT
SL:OC
SL:OF
SL:BC
SL:BF

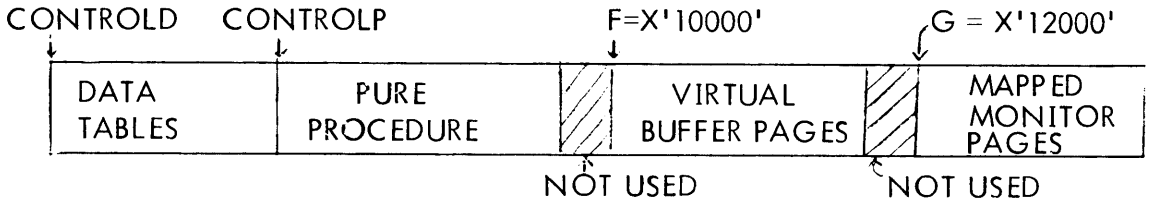
6. Monitor JIT references.

   OVHTIME
   TIMTMP

7. User JIT references

   JB:PRIV

DATA TABLES



The monitor pages are not mapped unless required by the function requested by the user. The data in the mapped monitor pages must be transferred to the virtual buffer pages within 40 msecs. for use by the program. The virtual buffer pages are used for history file writing and reading as well as for report generation. The mapped monitor pages do not require additional core since the resident monitor pages are simply mapped upward by 'G'.

All monitor constants are addressed in the program by an external reference plus the constant 'F'. The only references which are not buffered are th time references which are external references plus the constant 'G'. All external references must be satisfied by the MONSTK element file in the :SYS account.

SUBROUTINES

SDEC	Inputs an unsigned integer from the terminal and converts it to binary.
SNAME	Inputs a string (usually an item name) from the terminal.
SFIND	Finds a name in the table of item names (ATERM1 - ATERM5).
SBINOUT	Converts a number from binary to decimal.
SOUTA	Appends a string to the output buffer.
SOUT	Same as SOUTA, but also outputs the buffer.
SBLANK	Appends a specified number of blanks to the output buffer.
SEND	Appends a new line character to the output buffer and outputs the buffer.
SPD	Inserts a decimal point before the last character in the buffer.
STIME	Returns the time in seconds, rounded, since system startup.
READSI	Reads a line of input via M:SI printing the line only if M:SI is assigned to a file.
MPAGE	New page if less than 18 lines remain.

SPAGE	New page unless snapshots are being suppressed.
STIMES	Gets time in milliseconds since system startup.
LWRITE	Writes one history file record.
LREAD	Reads one history file record.
LNAME	Cracks file name syntax and fills the FPT's with appropriate data and control bytes.
DNAME	Gets time and date in TEXT and generates default file name.
OPENIN	Opens history file to test for existence.
OPENOT	Creats history file in dictionary.
OPENOT2	Creates files 'SNAPSHOT' and 'SSNAPSHOT'.
OPENUP	Opens the history file for update and positions to end of file.
CLOSE	Closes history file.
LWAIT	Used during a history file creation in order to let the program sleep between updates.
LVIR	Gets virtual page for the buffer used to read and write history file records and to buffer the monitor's data area.
LMAP	Maps the monitor's data area if needed for a particular function. This subroutine requires privilege level of 80 in the users account authorization.
GDAT	Transfers monitor data to the virtual buffer, processes the data by group to record old and new values for the statistics.

Get Fragments

LGET1	Gets values of control parameters.
LGET2	Gets values of single user variables.
LGET3	Gets values of use groups.
LGET5	Gets values for use distributions.

LOUT1 — Processes the data in the tables and prints reports for the overall and for the sample interval.

Output Fragments

LOUT1	Outputs control parameters and their sidings, two on a line.
LOUT2	Outputs single use variables and their sidings, two on a line.
LOUT3	Outputs use groups with their headings and sidings.
LOUT4	Outputs headers for use distributions.
LOUT5	Output use distributions and their sidings.

NOTE: The tables (see DATA BASE within CONTROL) used by LGET1 and LOUT1 end in 1 (e.g., ATERM1, APRIN1, AVAL1); those used by LGET2 and LOUT2 end in 2, etc.

ADDING AN ITEM TO A DISPLAY GROUP

1. Change the number of items in the group by redefining parameters D30 to D39 which are equated to constants in the first part of the program.

2. Define the computation of the numerator and the denominator of the new statistic in terms of the pseudo op codes used in tables GROUP31 to GROUP39. Generally there are group divisors computed at the end of each group and terminated by a SAVESUM pseudo op. The group divisors are used for GROUP32 to GROUP39. Only GROUP31 computes unique divisors for each parameter. These divisors are referenced uniquely by the LOUT subroutine via two tables ASUMRYN and ASUMRYO. There are two tables to enable both an overall statistic and a snapshot sample statistic to be computed.

   The snapshot statistic is computed by taking two sample points which define the beginning and end of the sample interval. Then the value of each numerator at the beginning of the sample is subtracted from the numerator at the end of the sample. (In most cases, the statistics are accumulative with time). Similarly the value of the denominators at the beginning of each sample is subtracted from the value of the denominator at the end of the snapshot interval. The computation of each numerator and divisor is executed in the GDAT subroutine in the LGET3 segment. The computation of the difference in statistics is also executed in the LGET3 segment of GDAT. This implies at least two passes to process one snapshot or one more pass through GDAT than the number of snapshots.

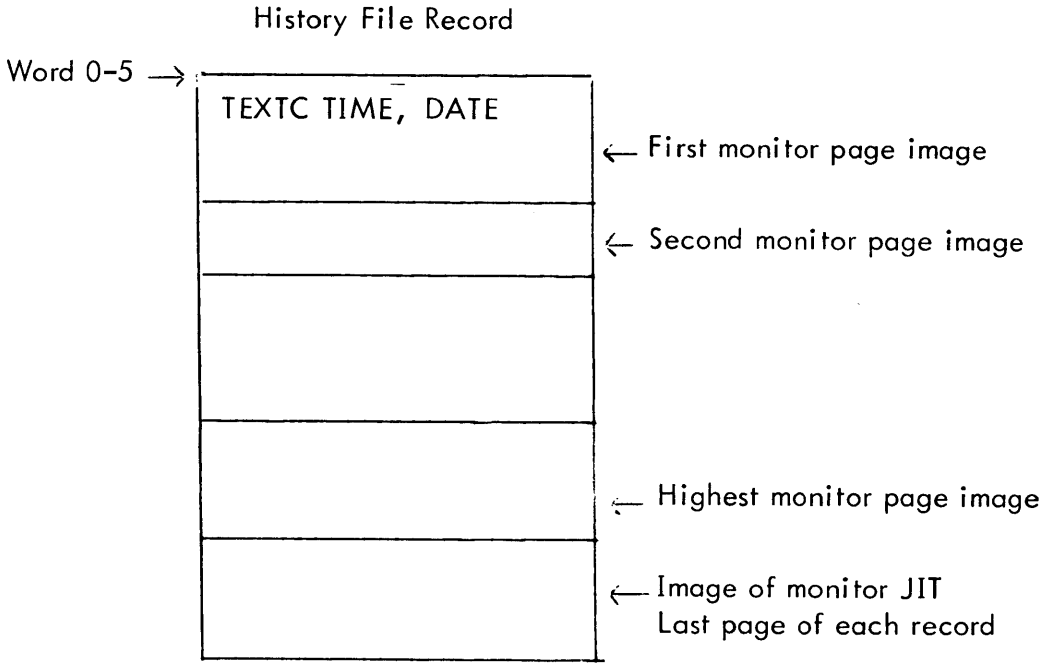
   The final computation of the printed group statistics is done in the LOUT3 segment of the LOUT subroutine. It is in LOUT that the numerators and denominators are selected for the OVERALL column and for the SAMPLE column.

3. The last step is to define a siding to be printed beside the new statistic by adding an entry to the text strings pointed to by ASIDE3.

4. Review data tables ADIV3, ASAM3, ADEC3, ALEN3, ADEPTH3 and ASCAL3 to be sure that format control flags and group scale factors are as desired.

RECORD FORMAT FOR HISTORY FILE

The history file is a consecutive file which contains a core image of the lower part of memory beginning with page zero and including up to 16 pages. The length of each record is determined at run time based upon the MONSTK references which were satisfied at load module time. The last page of data in each history file record is the monitor's JIT in order to satisfy the need for certain JIT references. In addition the first five words of the core image are overwritten with the TEXTC string for the time and date.



The history file record is always written from and read into the virtual pages beginning at the symbol defined as 'F', the buffered monitor area.

RECORD FORMAT FOR FILE 'SNAPSHOT' AND 'SSNAPSHOT'

The snapshot records for SNAPSHOT and for SSNAPSHOT are identical. The only difference between these files is that the SNAPSHOT file is consecutive while the SSNAPSHOT file is keyed. The key is a 15 character code generated by using an intensity group number, the number of users logged on, and the time date to insure completely unique keys.

Words 0-4	TEXTC for time and date
Words 5-8	TEXT for file name
Word 9	Length of SNAPSHOT record in bytes
Word 10-19	Lengths of Groups 30 + 39

GROUP 30 DATA, CON TROL PARAMETERS
GROUP 31 DATA, SUMMARY
GROUP 32 DATA, CPU
GROUP 33 DATA, ON-LINE
GROUP 34 DATA, USERS
GROUP 35 DATA, I/O
GROUP 36 DATA, TASK
GROUP 37 DATA, QUEUE
GROUP 38 DATA, INTERACT
GROUP 39 DATA, BATCH

12 SNAPSHOT DISTRIBUTIONS (8 WORDS EACH)

ERROR HANDLING

All error messages are described in the UTS/SM Reference Manual (90 16 74).

RESTRICTIONS

1. UTSPM must be loaded with element files MONSTK from the :SYS account to satisfy monitor references. Some unsatisfied PREF's may result due to MONSTK.

2. The user must have a privilege level of '80' to run UTSPM.

3. A special version of UTSPM may be desired in order to process older history files with a MONSTK other than the current system MONSTK.

COMMAND HANDLERS

Command	Segment
*(comment)	Treated as a null command.
END	LEXIT
PROCEED	LPRO
NUMBER	LNLS
ADD	LADD
DROP	LDROP
BUILD	LSETUP
CONTROL!	LCON
LIST	LIST
TIME	LTIME
DISPLAY	LUSE
FILE	LFILE TO LBASE
REPORT	LRFILE TO LRBASE OR LRHIST

UTS TECHNICAL MANUAL

ID

SUMMARY - UTS Performance Summary.

PURPOSE

The SUMMARY processor accesses the chronological SNAPSHOT file and keyed SSNAPSHOT files to provide the user with the following functions:

- Report one or more display groups chronologically (CHRONOLOGICAL SUMMARIES).
- Define a filter parameter list with high and low limits. (SPECIFY SORT FILTER).
- Report one or more display groups after filtering and sorting. (SORTED SUMM SUMMARIES).
- Report one or more display groups after filtering, sorting and averaging. (SORTED AVERAGES).
- Report overall statistics after filtering. This includes the mean, standard deviation, minimum, maximum and correlation co-efficients.

REFERENCE

This document presupposes familiarity with the UTSPM section and the SUMMARY section of the UTS/SM Reference Manual (90 16 74) which describes the use of UTSPM and SUMMARY from a user's point of view. The usage input and output are adequately described there. Please refer also to the UTSPM section of the Technical Manual.

PROGRAM SUBROUTINES

MAIN PROGRAM	This portion controls the overall logical flow of SUMMARY. All labeled COMMON creates dummy sections, DSECT's which are all defined the main program. The dimensions of any array must remain consistent in all occurrences in other subroutines. This applies to both Fortran subroutines and Meta-symbol subroutines.
SUBROUTINE ZERO	This subroutine sets the data in/XDATA/to zero for initialization purposes.

SUBROUTINE COVAR

This subroutine computes and accumulates minimums, maximums, squares and cross products. This is done after each filtered snapshot if the CORRELATION ANALYSIS is demanded by the user.

SUBROUTINE COVAR1

After all of the filtered snapshots have been processed the data in XDATA must be processed to compute the averages and correlation coefficients. It is in this routine that the double precision correlation matrix is needed for arithmetic precision. It should be noted that this matrix, YCOV is a symmetric diagonal matrix which is dimensioned only for non-redundant entries due to symmetry.

SUBROUTINE LIST

After the arithmetic is completed, this report is output if the user has requested a CORRELATION ANALYSIS. It is apparant that the matrix headings are dependent upon the statistical group sizes determined by UTSPM when the snapshot records are written. The sidings are determined by the IHDR array which is also dependent upon synchronization with UTSPM.

SUBROUTINE HEADER

For chronological and sorted summaries and sorted averages the headers for each statistical group are printed here. These formats are dependent upon the synchronization of group sizes with UTSPM as in SUBROUTINE LIST.

SUBROUTINE AVERAGE

The snapshots are averaged at the end of each chronological listing of a file and at the end of each Sorted Summary intensity group. The snapshots with the same number of users are also averaged when using Sorted Averages.

SUBROUTINE SCALE

When each snapshot is read, this subroutine transfers, floats and scales data from BUFFER to vector X.

SUBROUTINE ALINE

If the snapshot is to be listed this subroutine selects the group format to be used.

SUBROUTINE REPLY

The responses to all interactive questions are decoded here.

SUBROUTINE CNAME	If listing snapshots chronologically, this subroutine tests for a history file name change to allow history file averages to be printed.
SUBROUTINE CINTEN	If listing sorted summaries or sorted averages this subroutine is used to detect a change in estimated user intensity. The detection of a change causes the last intensity group averages to be listed.
SUBROUTINE PLOT	If statistical group 10 is being listed, this routine computes histogram percentages, adjusts scale to suit and prints the one line with one or more histograms.
FUNCTION ASUM	Computes percent for histograms when called by APLOT.
SUBROUTINE APLOT	Prints titles, and entire line of print for one or more histograms via the PLOT subroutine.
READT	Metasymbol routine to read a line via M:SI into COMMON /IANS/, (a dummy section). If reading M:SI from a file, the line read is echoed via M:DO for user convenience. Usually the data is interpreted by a FORTRAN DECODE statement.
PRTDO	Metasymbol routine to print a line from COMMON /IANS/, (a dummy section). The data in IANS is filled by the FORTRAN ENCODE statement.
READSN	Metasymbol routine to read a record via F:2 into a labeled common area COMMON /BUFFER/, (a dummy section).
READSS	Metasymbol routine to read a record via F:3 into a labeled common area, COMMON /BUFFER/, a dummy section.
NEWLIN	Metasymbol routine to return the number of lines remaining via COMMON /LINES/.
CPROM, SPROM	Metasymbol routines to set the prompt character for on line user convenience.

USING M:SI, M:CO, M:DO

The FORTRAN program utilizes monitor DCB's via METASYMBOL subroutines and the use of the ENCODE/DECODE statements in FORTRAN.

DATA TABLES

COMMON /BUFFER/	This integer buffer is used to read snapshot records. The format of this buffer is defined by the UTSPM processor which creates snapshot records during history file report writing (REPORT H command of UTSPM). The SCALE subroutine assumes the format of BUFFER via EQUIVALENCE statements. All statistics reported in SUMMARY are received in binary form in BUFFER. The INTEGER statement must appear in each subroutine using BUFFER so that the statistics in BUFFER will be interpreted correctly.
COMMON /ABUF/	This buffer is used to record up to 8 characters.
COMMON /BFILN/	This buffer is used to store the text for the file name of the history file being reported chronologically.
COMMON /ISTAT/	This buffer is used to return the status of a file manipulated in the METASYMBOL subroutines.
COMMON/IANS/	This buffer is used with the ENCODE/DECODE statements to pass or receive data written or read by the METASYMBOL subroutines. Usually the buffer reads from M:SI and writes to M:DO and M:LO.
COMMON /IALPHA/	This area includes two double precision buffers used to specify the statistical group headers and the SIDRs for the reports. These text strings must be synchronized with the format of the statistics as formatted by UTSPM.
COMMON /XLIMIT/	This array is used by the 'SPECIFY SORT FILTER' command to record the filter item numbers and the low and high rejection criteria. The rejection count is also kept in this array.
COMMON /XDATA/	This area includes several data vectors. The XM and YCOV vectors are both double precision vectors in order to keep enough precision for the correlation coefficient computation. XM accumulates the sum of all data for the mean computation. XMIN and XMAX

are used to record the minimums and maximums respectively for the correlation analysis.

COMMON /LINES/ — This is used to transfer the number of lines remaining from the METASYMBOL subroutine to the FORTRAN subroutine.

COMMON KDIM — This unlabeled COMMON area is used to record the dimension of the statistical groups of data as read from the BUFFER snapshot record.

DIMENSION XOLD — This vector is used to reduce the number of lines of print when listing 'Group 0', the control parameters chronologically. A new line is printed only when the parameters are changed.

DIMENSION XAVE — This vector accumulates snapshot averages for a given intensity group and a given number of users logged when using the 'SORTED AVERAGES' command.

DIMENSION YM — This vector is used to compute history file averages when using 'CHRONOLOGICAL SUMMARIES'. In addition this vector is used to compute intensity group averages for 'SORTED SUMMARIES' and with 'SORTED AVERAGES'.

DIMENSION SNTIME, SFILN, JDIM, ISNAP are all used with EQUIVALENCE statements to match the format of the snapshot files.

ID

LINK - UTS on-line one pass loader

PURPOSE

LINK is used from on-line terminals to load from multiple input files that contain either single or multiple ROMs. In addition user ROM libraries can be loaded with a Core Library and the System Library.

RESTRICTIONS

LINK can only load ROM information. It cannot load Load Modules. Dummy Sections (DSECTS) must be defined at their largest size in the initial definition. Also, multiple use of the same Dummy Section must not involve the use of instructions that use external references either Secondary or Primary. If the definitions are loaded prior to the references the preceding restriction is not necessary. A reference to M:XX must be satisfied from the JIT (J0) core library. LINK will not build an M:XX DCB nor may the user build one.

DESCRIPTION

Upon entry to LINK, memory is obtained via the Get Virtual Page CAL for the loader stacks. The initial stack configuration is assembled into LINK. The ':' prompt character is set via the Set Prompt CAL.

Each input element file name from the RUN or LINK command is translated into a Flag word and a variable length parameter list for an OPEN CAL. The Flag word has a bit for each input option.

The output file is opened to perform the 'ON', 'OVER' checking. If 'ON' is specified for an already existing file an error message is given and LINK exits.

Otherwise LINK continues and puts a DEF entry in the RFDF stack for the JIT DCB M:UC. A REF entry for M:DO is placed in the RFDF stack to insure the generation of a M:DO DCB so that snap shot debug output may be produced by the loaded program at execution time. In order to allow Fortran to use N as an I/O device indicator, the Assign Merge table is read. All F: numeric entries from the assign merge record are used to generate REF entries in the RFDF stack to cause the generation of a DCB from the F: numeric SET command in TEL.

The first input element file is opened. If this is the '$' file, the M:GO DCB is opened in INPUT mode. Otherwise the F:LINKIN DCB is opened with the fid of the first element file on the RUN of LINK command.

Each ROM control byte is processed by the appropriate routine. The core image of the program is built into each of the three program areas: Data, Pure Procedure, and DCBs.

When the Module End ROM control byte is encountered, an end of file check is made. If no end of file is found, it is assumed another ROM follows in the current file and ROM control byte processing continues.

When an end of file is reached, the current input file is closed. If Internal Symbol Table ROM control bytes were encountered and an Internal Symbol Table is wanted, it is written out as a keyed record as part of the output load module.

A check is made to determine if any more input element files are present; if so the next input element file is opened and ROM control byte processing continues. When a library input element file is encountered, LINK goes into library mode and only ROMs that satisfy Primary References are loaded.

When the last input element has been processed, LINK goes into library mode. A search of the RFDF stack is made for Primary References. If none are found, control goes to ENDLOAD. Primary References cause searching first of the Core Library and second of the System Library.

The Core Library is searched by reading its RFDF stack into the symbol table area of memory. All of the Primary and Secondary References in LINK's RFDF stack are searched for in the Core Library's RFDF stack. When a match occurs, LINK's RFDF chain is satisfied with the value from the Core Library RFDF stack. The System Library is treated like any other ROM library element file. Only those ROMs that satisfy Primary References are loaded. The Core Library is searched a second time after loading the System Library. This is to satisfy references into the Core Library from the System Library. Control goes to ENDLOAD after library searching.

At ENDLOAD LINK's RFDF stack is searched for Primary References that start with 'F:' or 'M:'. If found, these references are considered DCBs and a 48 word DCB is constructed for each reference. When the name of the DCB is recognized as one of the standard DCBs, assignment information is put into the DCB. It should be noted that the two JIT DCBs, M:UC and M:XX, are not constructed by LINK.

LINK constructs a DCB table (DCBTAB) of all the entries in the RFDF stack that start with 'F:' or 'M:' except for M:UC and M:XX.

At this point the size of the RFDF stack is saved. The additional entries added by LINK for MAP printout are not written out as part of the Load Module. The MAP is then printed.

The RFDF stack is scanned and a Global Symbol is produced and written out as part of the Load Module. The user's TCB is constructed immediately after the user's Data area and is written out as part of the Data area.

In the writeout phase of LINK, the HEAD, TREE, RFDF stack, Data area, Pure Procedure area, DCB area, and Global Symbol table are written out as keyed records in the load module.

From the LINK command, an exit (CAL1, 9 1) is executed. In the RUN command the stack memory is released, the symbol table memory is released, and HEAD+15 is set to zero before the exit CAL is executed.

DATA BASES

A. Stacks

Probably the most important thing to understand about the loader is the make-up of each of the stacks and the nature and use of the entries in them.

There are 5 stacks. The size of the stacks may vary depending on the usage. The subroutine PRESSTK collapses the stack space if possible, and the subroutine STKOVF expands the appropriate stack in the event of overflow. All subroutines using the stacks check for overflow at the completion of pushes and call STKOVF with the offending stack number as an argument for correction of the overflow problem. A certain amount of extra space is left in the temporary stack for non-recursive use.

In general, the size of the loader stacks is variable. They expand as entries are added to them. The entries in the stacks are referenced indirectly through stack pointers. This allows the actual position of the stack in core to be variable. This is used to move the stacks around to make more space available when correcting for stack overflow. Also, space (unused) may be squeezed out of stacks when trying to generate more space, when no more new space is available. That is, there may be enough space if it is distributed amoung the stacks in a more equitable manner.

T:STACK:

The temporary stack TSTACK is the simplest one to consider. The entries here are just temporary and are generally of little interest. A certain minimum fixed size is kept to allow for non-recursive use without checking for stack overflow.

EXPRSTK:

Refer to Figures 1 through 3. The expression stack is used to hold expressions which may or may not be evaluatable. In general, certain load items call for an expression to be input. When this occurs, the destination type of the expression is placed in R6 and a call made to EXPRIN (the expression input routine). This routine constructs an expression of the general form shown in Figure 1. Then EXPTTB is called to place the expression in the stack. Finally, EVEXP is called in an effort to evaluate the expression (if possible). Sometimes expressions cannot be evaluated because of forward reference problems or REF problems.

Refer to Figure 1. The word count is the total number of words in the expression. The next byte contains a pointer to the first value word. This is a number greater than 2. The next byte has a bit marked in it which gives the destination of the expression. A detailed picture of this byte is given in Figure 2. The last byte of word 0 and subsequent bytes form the "expression control string". This is somewhat like a string of operators.

These bytes determine what is to be done to the expression accumulator during evaluation. Possible operations are things like add constant, add declaration, change resolution. If the operation requires an 'argument', this argument is available as the next value word. The value word contains either the value (when defined) or a chain pointer which may point to another expression in the stack or core location which needs the value. The pointer is of the form shown in Figure 3. Zero is used to indicate the termination of a chain. Note that the relative word in the expression stack chain will never be zero. When the relative word in the expression pointer is zero, this indicates that linking in the expression stack has terminated and indicates chaining into a core chain.

The expression control string may be arbitrarily long. There is a byte which indicates "end of expression". The next word following contains the address of the destination. The subsequent words are "value words" which contain values to be used as operands by the expression control bytes. In the case of values which are not yet available to the program (REFs or forward references) the value word contains a chain pointer.

FREFSTK:
The forward reference stack FREFSTK has two word entries (see Figure 5). The first word contains the forward reference number, and the second word contains a chain origin to the expression stack. This points to one of the value words of an expression requiring this forward reference value and there is a corresponding expression control byte indicating add or subtract a forward reference. These entries remain in the forward reference stack only while the forward reference is undefined.

This is OK since forward references are not to be referred to after they have been defined. When the forward reference definition is encountered by the loader, the link to the expression stack is used to begin evaluating expressions if possible. If evaluation is possible, the link in the expression is removed and the value word now contains the value. The corresponding expression control byte is set = defined.

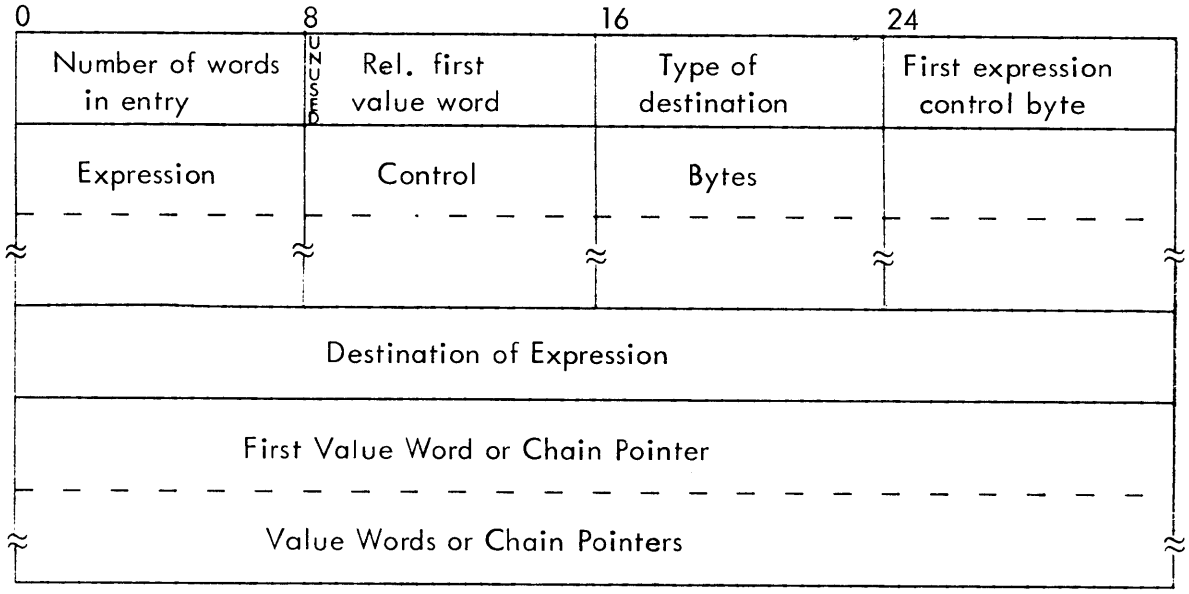
There exists the ability for holding the value of forward references. This load item is called forward-reference-and-hold. In this case bit 0 of word 1 of the entry is marked in the forward reference stack. This designates the item as forward-reference-and-hold. Word 2 then contains the value. This item remains in the FREFSTK until the module end is reached. Ordinarily, forward references are deleted from the stack when they are defined.

RFDFSTK:
The entries in the RFDFSTK are variable length items. Refer to Figure 6. The first byte (byte 0) contains the word count of the entry. The second word contains the value (if defined) or a chain origin to the expression stack. The nature of the linkage here is the same as that of the forward reference stack. The third word is used for type and resolution information in the Global Symbol Table entry. The length of the name may be up to 63 characters. The first byte of word 4 is a count of the number of bytes in the name; subsequent bytes comprise the name in EBCDIC.

There is additional information about the entry contained in byte 1. There is a bit marked to indicate the type of entry. A description of the function of each bit is shown in Figure 7.

DECLSTK:
The entries for the declaration stack (DECKSTK) are shown in Figure 4. Declarations are used to provide information about control sections and DEF/REF situations to the loader. There is a two word entry for each such declaration. Byte 0 is flagged to indicate the type of declaration. For sections, a base address and the size is required. This information is in the remainder of word 0 and word 1. The base address and size are byte resolution. If the information is not available, a pointer to the RFDFSTK is placed in word 1. The protection code is placed in the upper two bits of byte 1, word 0. These entries are retained in the stack for the duration of the current ROM.

FIGURE RA-1 EXPRSTK - Expression Stack Entry



If bit 8 of the first word is set, the entry space is unused.

Bit 0 of each expression control byte when set indicates that the associated value word is defined.

FIGURE RA-2 Type of Destination Byte

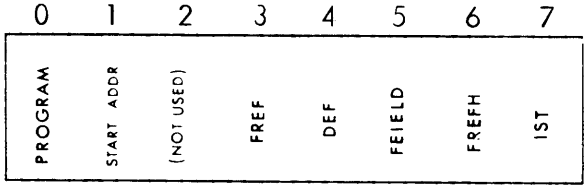
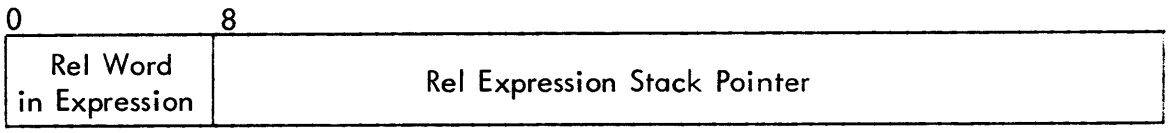


FIGURE RA-3 Value Word Chain Pointer



When relative word in expression is zero, the pointer refers to core.

FIGURE RA-4 DECLSTK - Declaration Stack

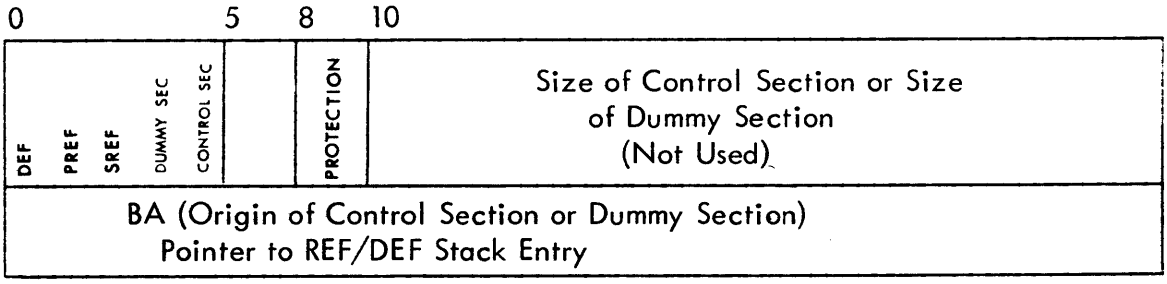


FIGURE RA-5 FREFSTK - Forward Reference Stack

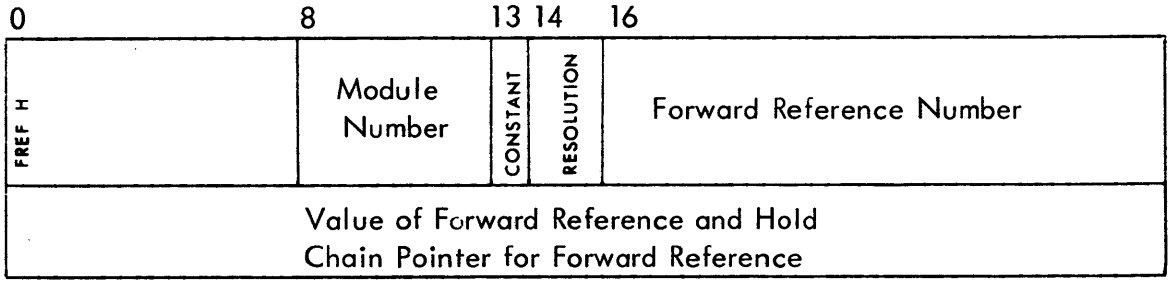


FIGURE RA-6 RFDFSTK - REF/DEF Stack

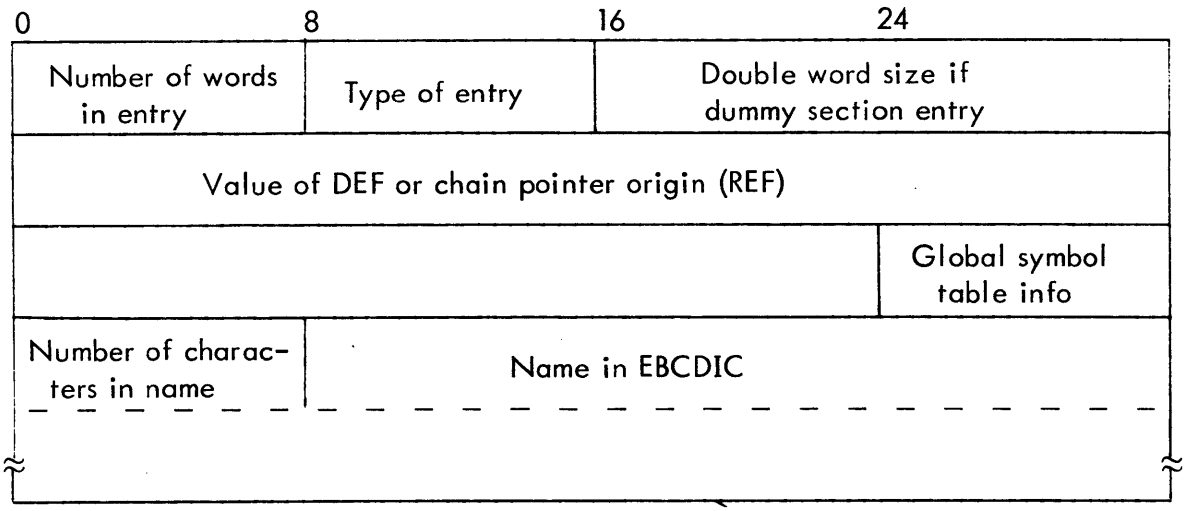


FIGURE RA-7 "Type of Entry" in RFDFSTK

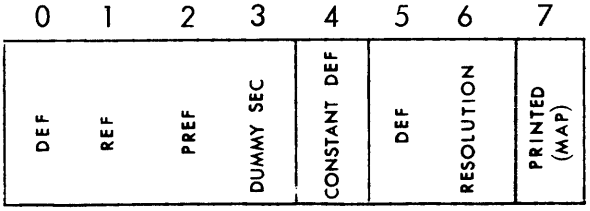
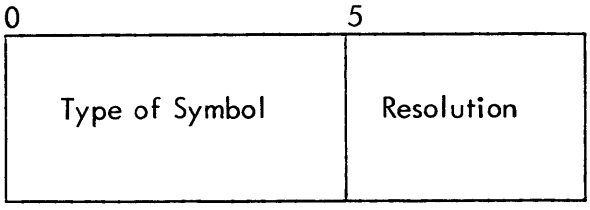


FIGURE RA-8 Global Symbol Table Information



byte 3 of third word

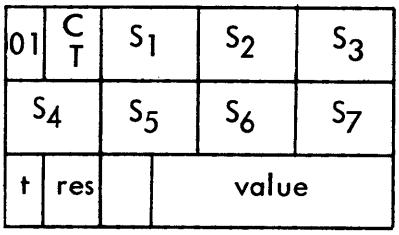
B. Symbol Tables

Global Symbols:
While performing the link process, a global symbol table is constructed. This table is a list of correspondences between symbolic identifiers (labels) used in the original source program and the values of virtual core addresses which have been assigned to them by LINK. The global symbols identify object (DEFs) within a module which may be referred to (REFed) in other modules. This table is available to DELTA, for use in debugging.

Internal Symbols:
An internal symbol table is a list of correspondences similar to the global but which applies solely within the module. Each internal symbol table constructed by LINK is associated with a specific input file and identified by its name. The internal as well as the global symbol tables are created for use by the debug processors, such as DELTA. The user has the ability under DELTA to define which set of internal symbols is to be used for specific debugging activities.

Symbol Table Format:
Both global and internal symbol tables, as formed by LINK and used by DELTA, consist of three word entries. Symbolic identifiers (labels) are limited to seven (7) characters plus count. Symbols originally longer than seven are truncated leaving the initial characters, although the original character count is retained. Symbols which are identical in their first seven characters and are of equal length occupy one position in the symbol table. The value or definition for such multiply defined symbols is the first one encountered during the linking process. Each symbol entered into the table has a type and internal resolution classification. The internal resolution types are: byte, halfword, word, doublewrod, and constant. The following are the symbol types which are supplied by the object language and maintained in the symbol table: instruction, integer, EBCDIC text short floating point, long floating point, decimal, packed decimal, and hexadecimal.

Location Symbol - code = 01



where:

CT is a six-bit field containing the character count of the original symbol.

S_i are the first seven (7) characters of the symbol. Symbols with fewer than seven characters are zero filled.

t — is a five-bit field where the values are:
- 00000 - instruction
- 00001 - integer
- 00111 - EBCDIC test (also for unpacked decimal)
- 00010 - short floating point
- 00011 - long floating point
- 00110 - hexadecimal (also for packed decimal)
- 01001 - integer array
- 01010 - short floating point array
- 01011 - long floating complex array
- 01000 - logical array
- 10000 - undefined symbol

res — is a three-bit field representing the internal resolution. The values are:
- 000 - byte
- 001 - halfword
- 010 - word
- 011 - doubleword

value — location symbols are always represented as a 19-bit byte resolution value.

Constants - code = 10

10	CT	S_1	S_2	S_3
S_4		S_5	S_6	S_7
value				

where:

CT and S_i — have the same meaning as above.

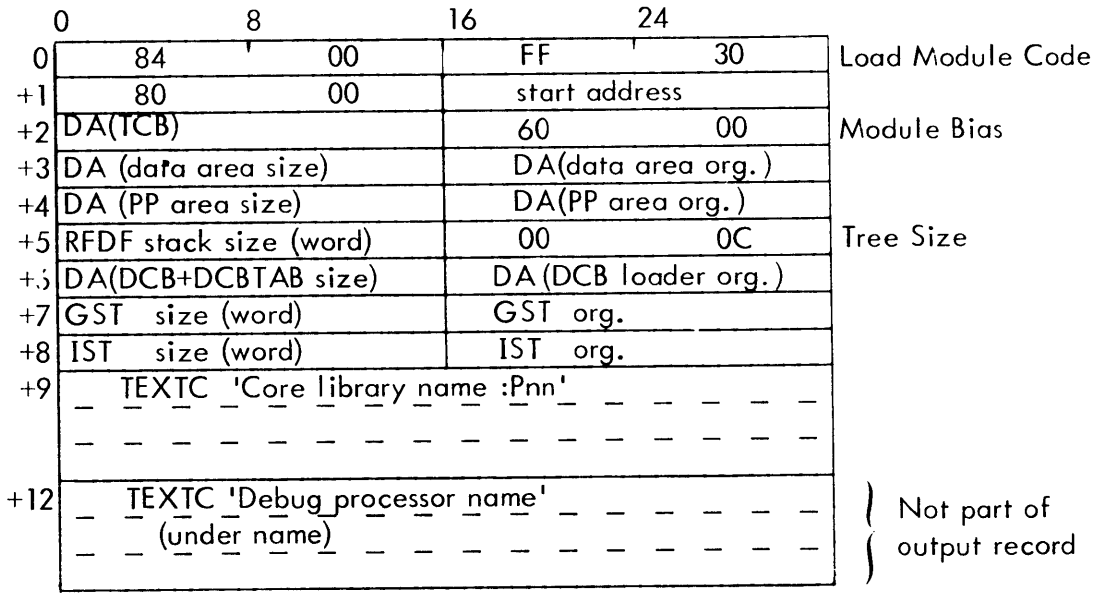
value — is the 32-bit value of the constant.

OUTPUT

The load module output from LINK is a KEYED file. There is a keyed record for the HEAD, TREE, RFDF stack, DCB area, DATA area, PURE PROCEDURE area, Internal Symbol Tables, and the Global Symbol table. The Key name for the head and tree records is HEAD and TREE respectively. The Key name for the RFDF stack, DCB, DATA, PURE PROCEDURE, and Global Symbol table is the output file name concatenated with an identifying key code. The Internal Symbol table records have Key names made up from the input file names concatenated with an identifying key code.

FIGURE RA-9 Load Module File

Keyname: HEAD



Keyname: 'Output File Name'

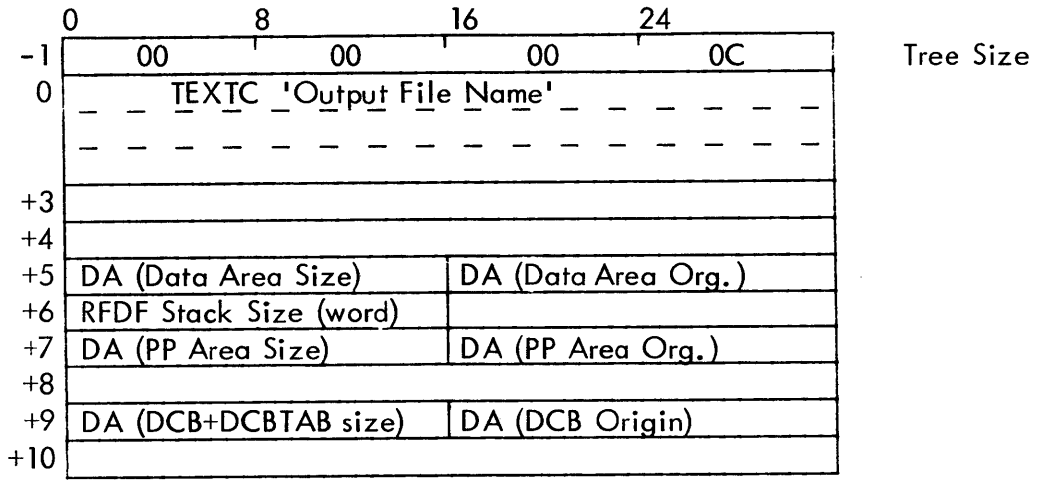
Followed by: 03 - Data area control sections
05 - Pure procedure area control sections
07 - DCB+DCBTAB
09 - Global symbol table

Keyname: 'Input File Name'

Followed by: 10 - Internal symbol table

FIGURE RA-9 Load Module File (Cont'd)

Keyname: TREE



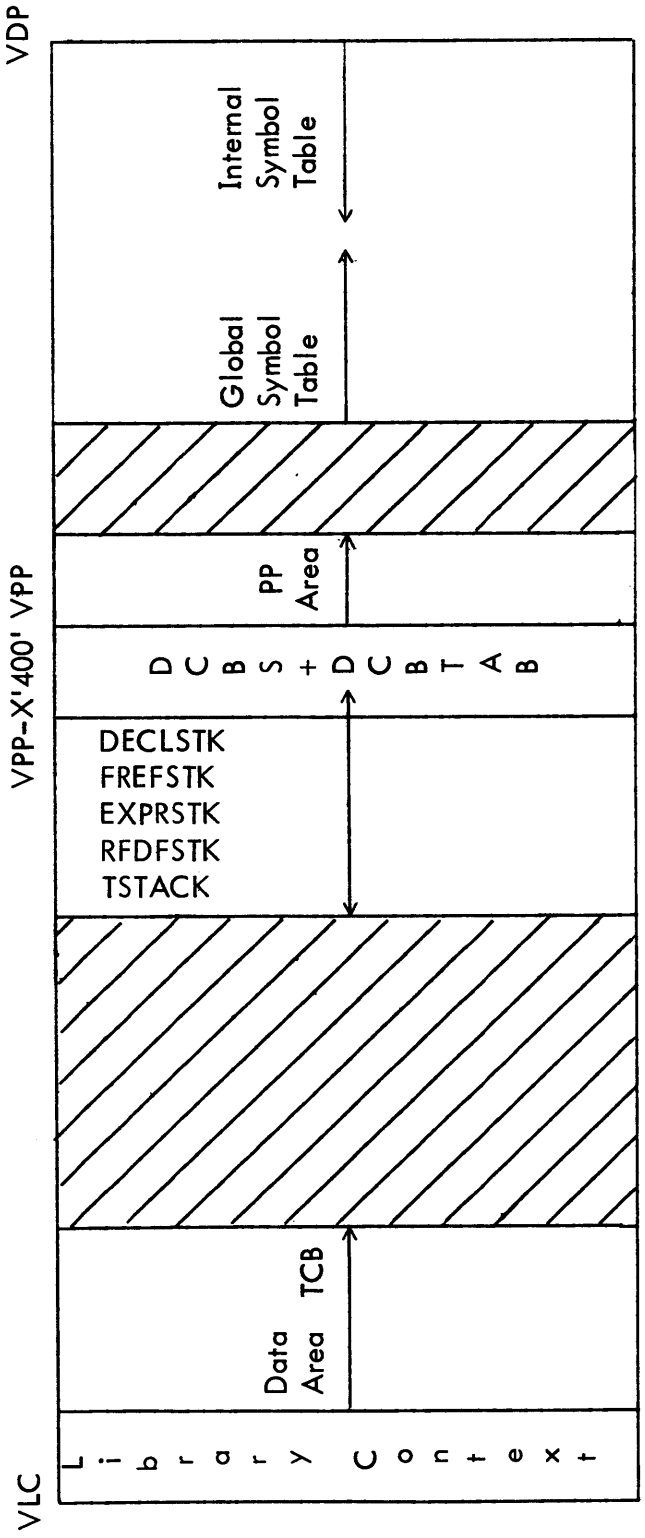


FIGURE RA-10 Virtual Core Layout During Load

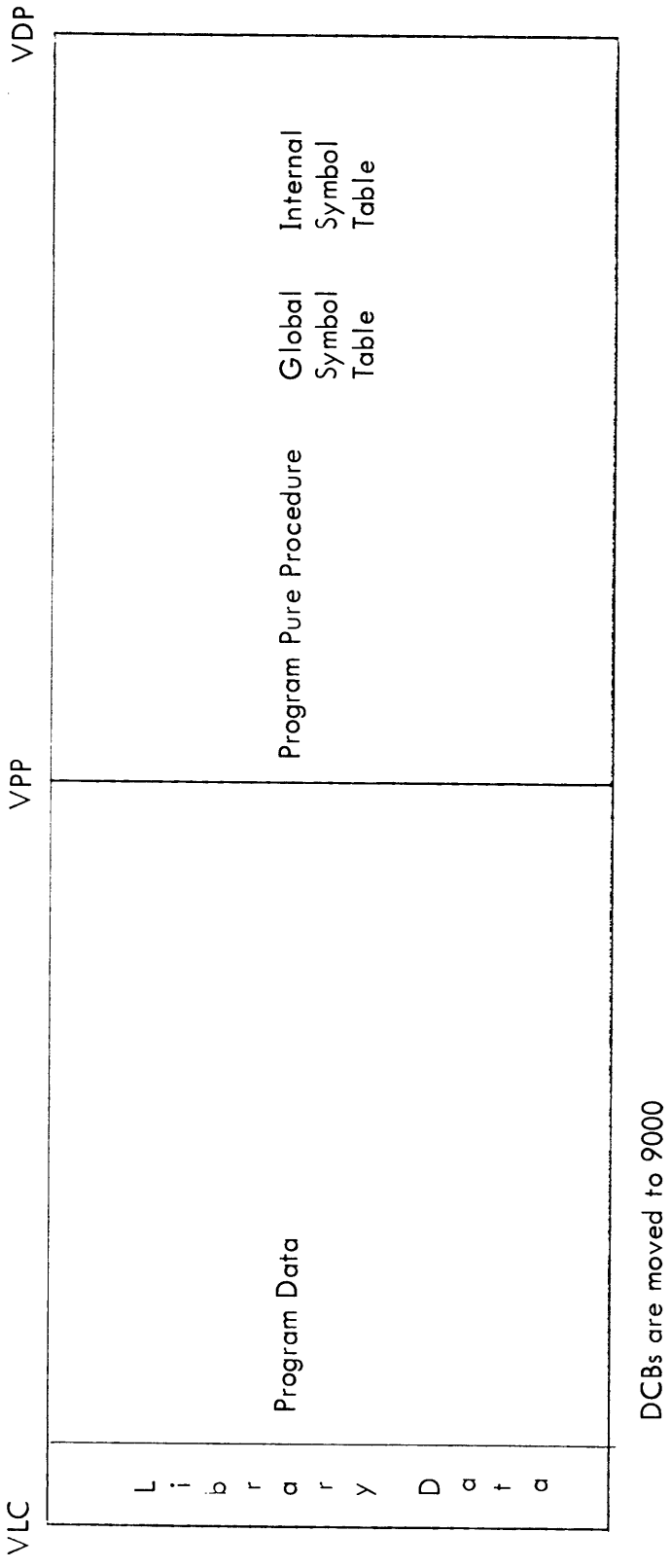
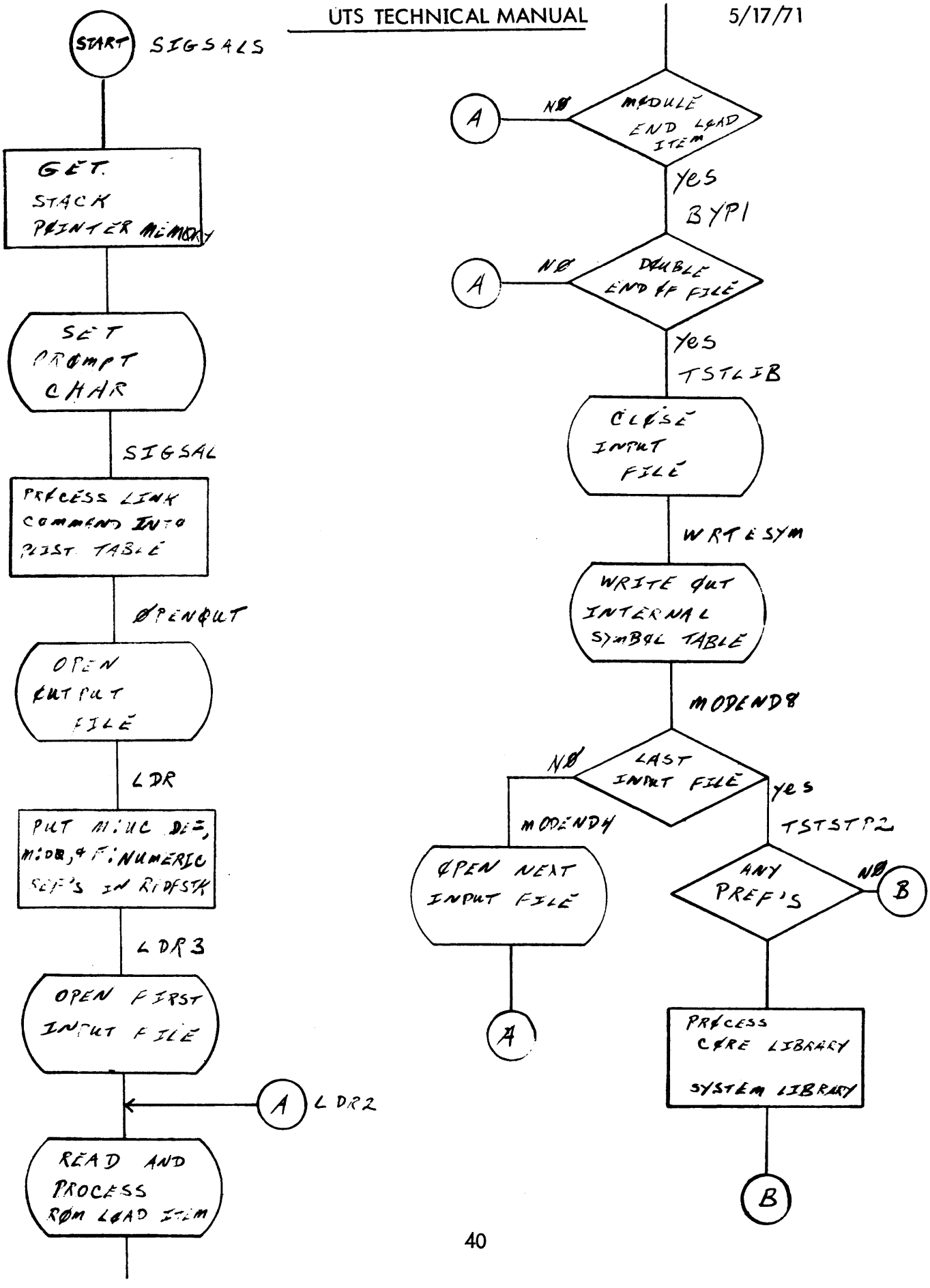
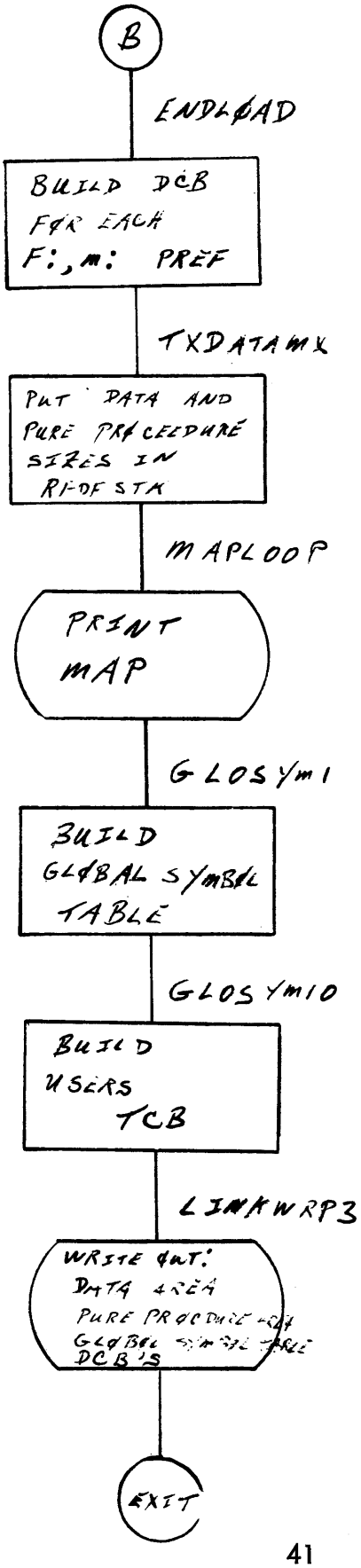
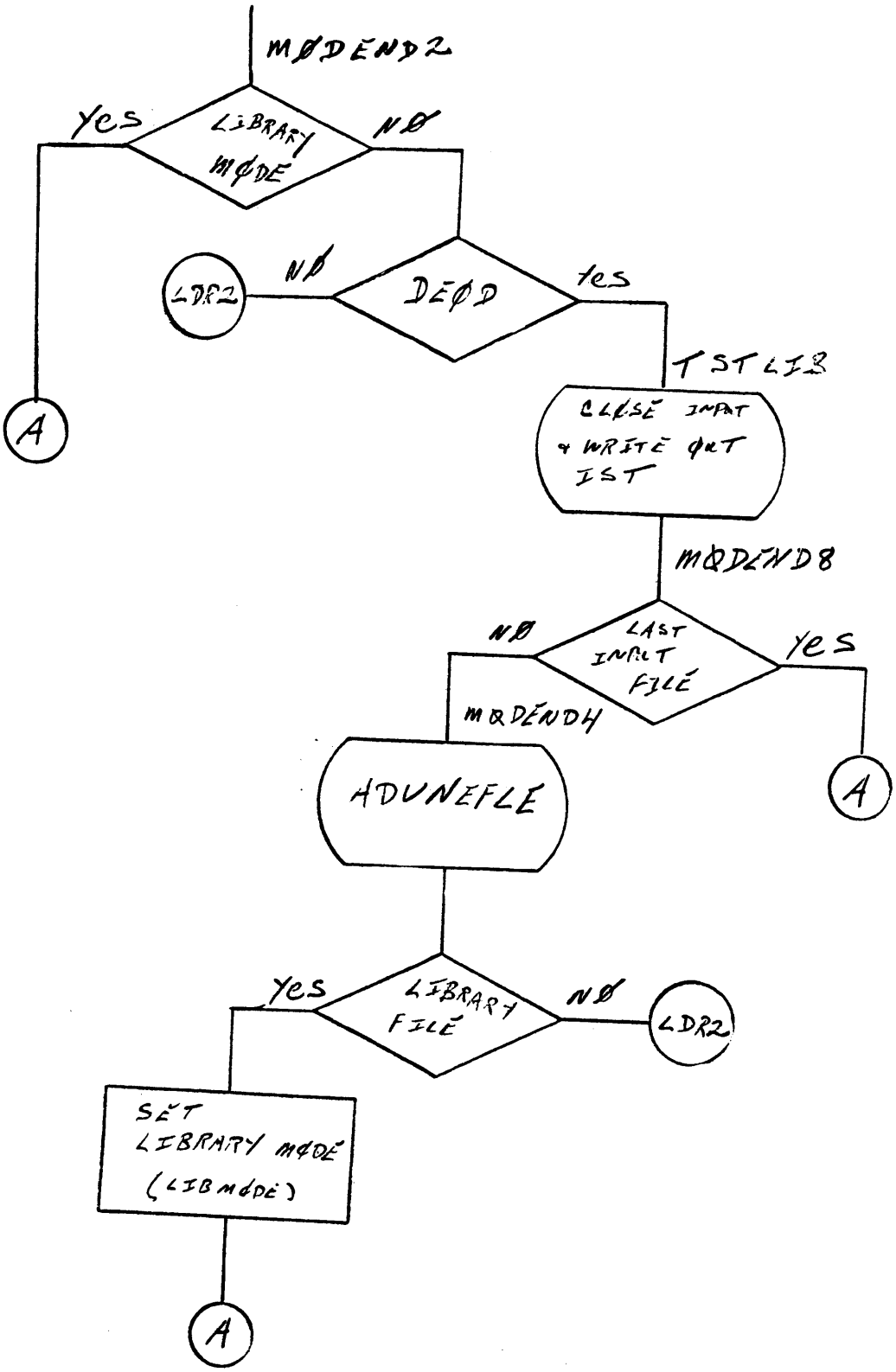
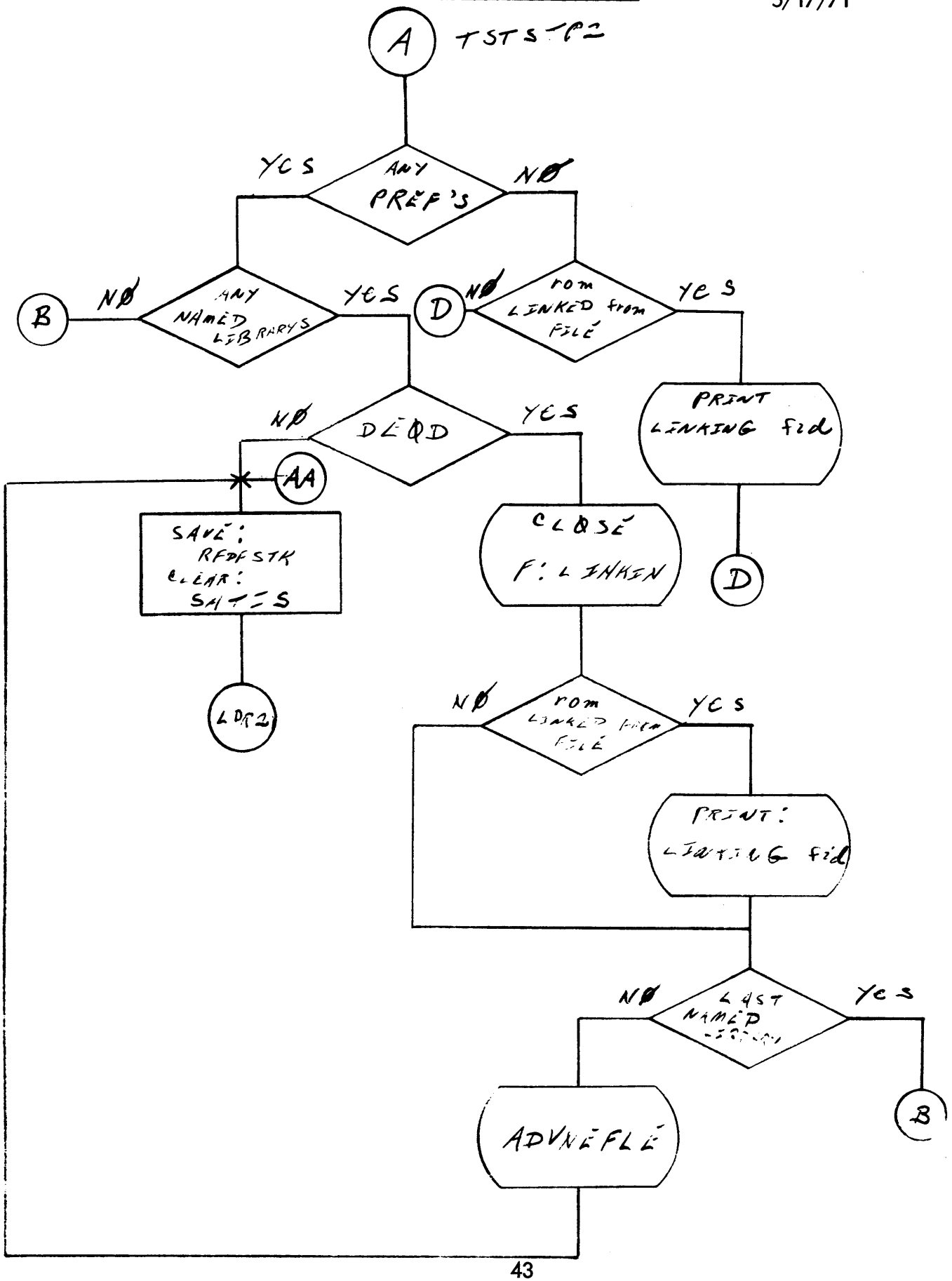


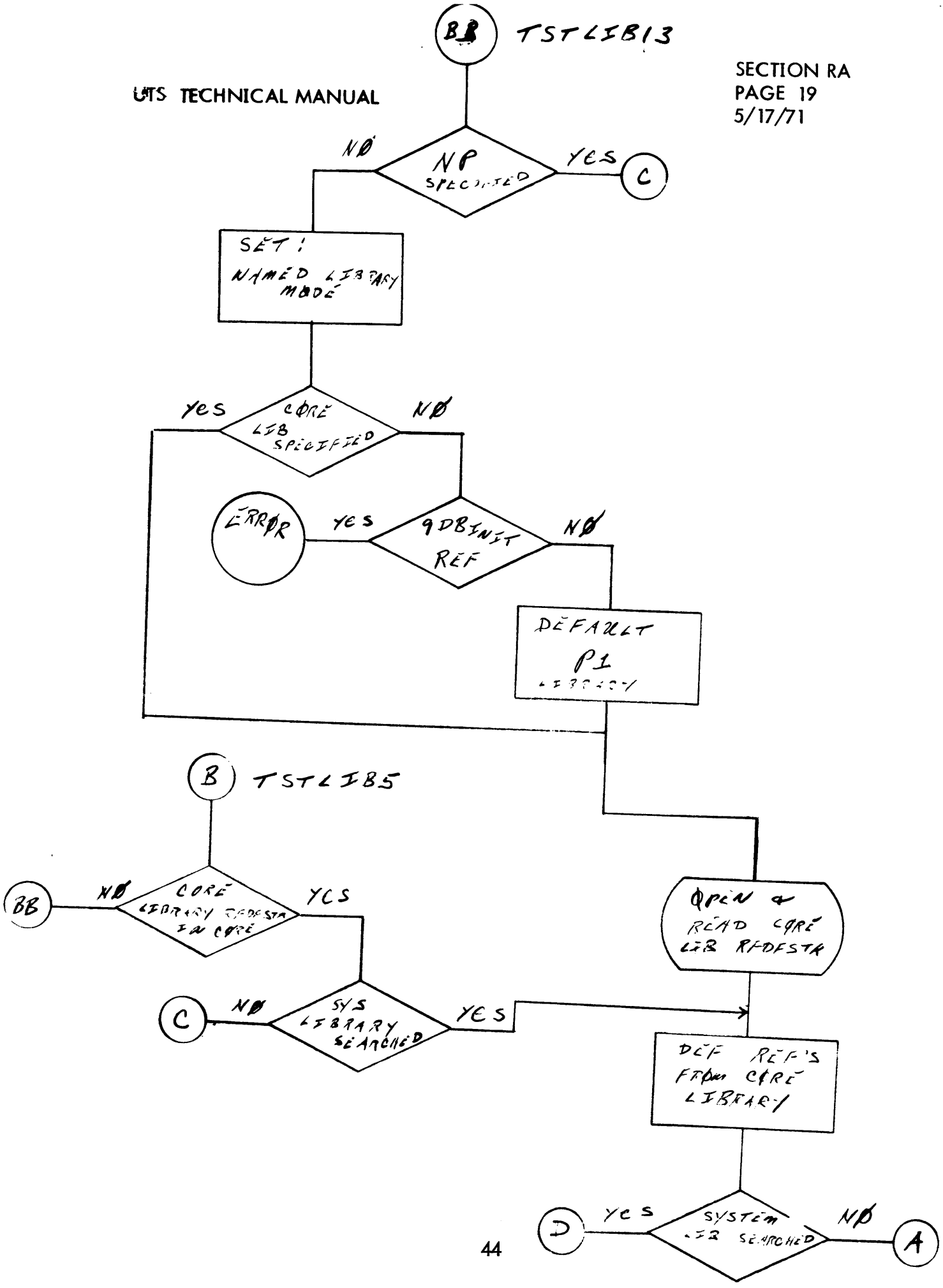
FIGURE RA-11 Run Time Virtual Core

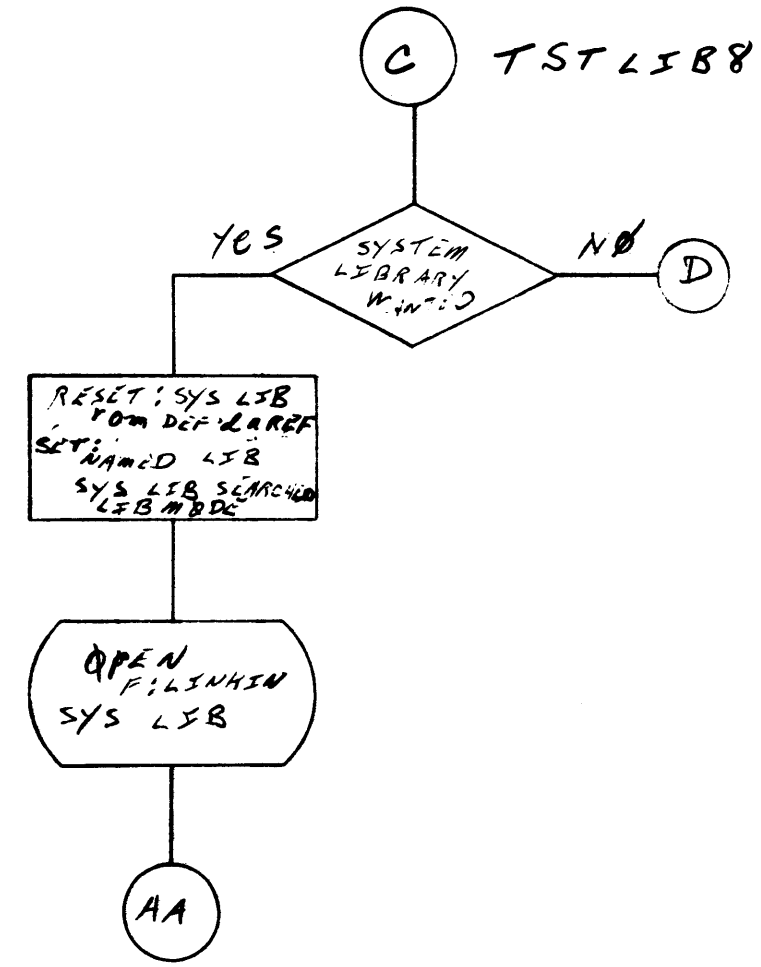

START
SIGSALS
GET STACK PRINTER MEMORY
SET PROMPT CHAR
SIGSAL
PROCESS LINK COMMAND INTO LIST TABLE
OPENOUT
OPEN OUTPUT FILE
LDR
PUT MODE, & F. NUMERIC DEF'S IN RIDFSTK
LDR3
OPEN FIRST INPUT FILE
A
LDR2
READ AND PROCESS ROM LOAD ITEM
A
NO
MODULE END LOAD ITEM
yes
BYP1
A
NO
DOUBLE END OF FILE
yes
TSTLIB
CLOSE INPUT FILE
WRTESYM
WRITE OUT INTERNAL SYMBOL TABLE
MODEND8
NO
LAST INPUT FILE
yes
MODEND4
TSTSTP2
OPEN NEXT INPUT FILE
A
ANY PREF'S
NO
B
PROCESS CORE LIBRARY SYSTEM LIBRARY
B

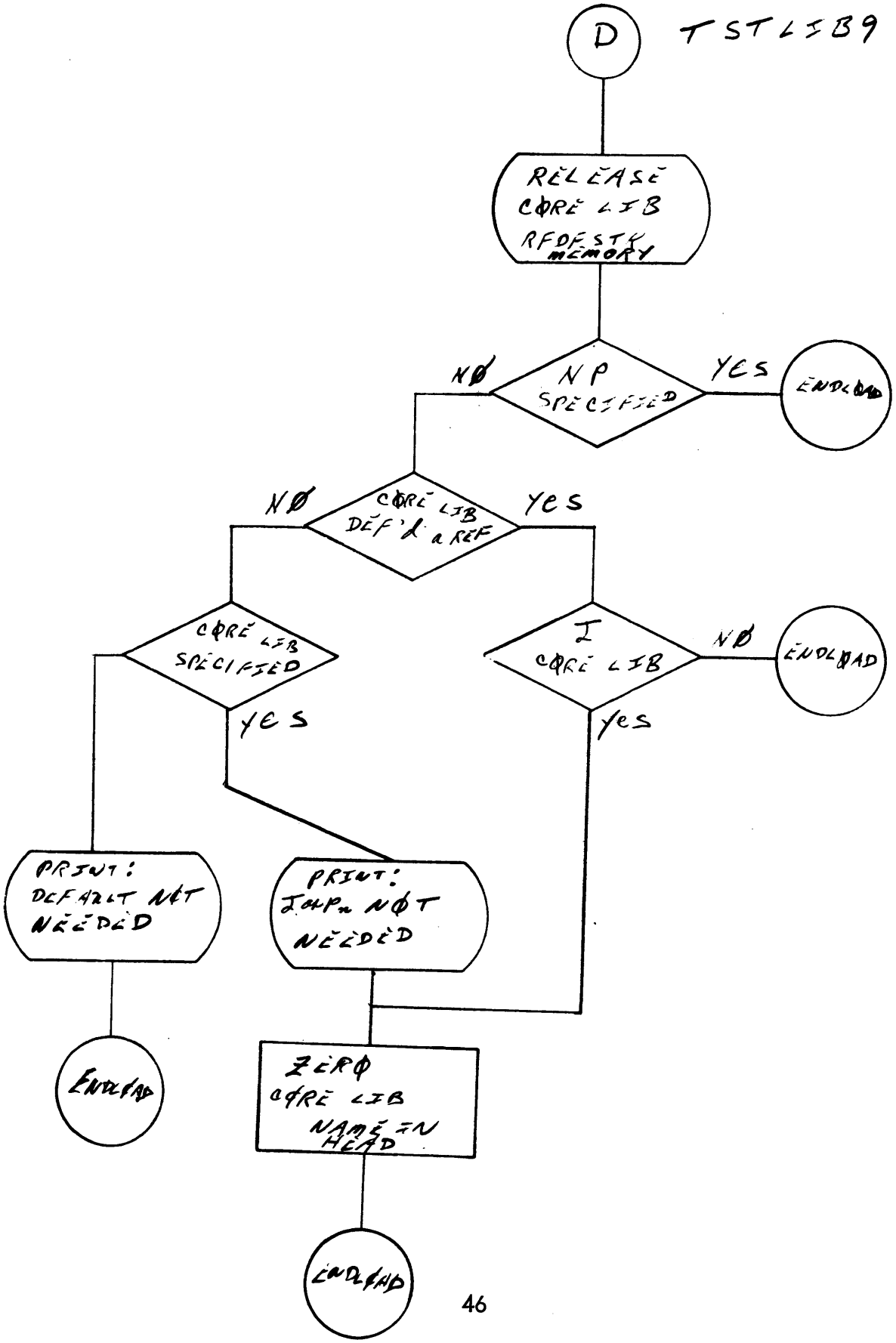

B
ENDLOAD
BUILD DCB FOR EACH F:,M: PREF
TXDATAMX
PUT DATA AND PURE PROCEDURE SIZES IN RMODSTA
MAPLOOP
PRINT MAP
GLOSYM1
BUILD GLOBAL SYMBOL TABLE
GLOSYM10
BUILD USERS TCB
LINKWRP3
WRITE OUT: DATA AREA PURE PROCEDURE AREA GLOBAL SYMBOL TABLE DCB'S
EXIT


MODEND2
LIBRARY MODE
yes
NO
LDR2
NO
DEOD
yes
TSTLIB
A
CLOSE INPT & WRITE OUT IST
MODEND8
LAST INPUT FILE
NO
yes
MODEND4
ADVNEFLE
A
LIBRARY FILE
yes
NO
LDR2
SET LIBRARY MODE (LIBMODE)
A


A
TSTSTP2
ANY PREF'S
YES
NO
B
NO
ANY NAMED LIBRARYS
YES
D
NO
rom LINKED from FILE
YES
NO
DEQD
YES
PRINT LINKING fid
AA
SAVE: RFDFSTK
CLEAR: SATES
CLOSE F: LINKIN
D
LDR2
NO
rom LINKED from FILE
YES
PRINT: LINKING fid
NO
LAST NAMED
YES
ADVNEFLE
B


BB
TSTLIB13
NO
NP SPECIFIED
YES
C
SET NAMED LIBRARY MODE
YES
CORE LIB SPECIFIED
NO
ERROR
YES
QDBINIT REF
NO
DEFAULT P1 LIBRARY
B
TSTLIB5
BB
NO
CORE LIBRARY RFDESTR IN CORE
YES
C
NO
SYS LIBRARY SEARCHED
YES
OPEN & READ CORE LIB RFDESTR
DEF REF'S FROM CORE LIBRARY
D
YES
SYSTEM LIB SEARCHED
NO
A


C
TSTLIB8
YES
SYSTEM LIBRARY
NO
D
RESET : SYS LIB
SET : NAMED LIB
OPEN
FILINKIN
SYS LIB
AA


D
TSTLIB9
RELEASE CORE LIB RFDEST MEMORY
NP SPECIFIED
NO
YES
ENDLOAD
CORE LIB DEF'D & REF
NO
YES
CORE LIB SPECIFIED
I CORE LIB
NO
ENDLOAD
YES
YES
PRINT: DEFAULT NOT NEEDED
PRINT: IOUP NOT NEEDED
ENDLOAD
ZERO CORE LIB NAME IN HEAD
ENDLOAD

ID

ADVNEFLE - Advance to next input file

PURPOSE

To advance to the next input file from the list of input files on the LINK or RUN command.

USAGE

```
      BAL, SR4        ADVNEFLE
Secondary entry point
      BAL, SR4        FPTLOAD2
to print name of current input file
```

OUTPUT

PLISTBUF	-	Pointer within current file entry
PLISTORG	-	Pointer to current file flag word
FIRSTRD	-	First read SWT (rewind if EOD)
OPNIN	-	Open FPT for F:LINKIN DCB

INTERACTION

CAL1, 1	Set DCB (remove abnormal address)
CAL1, 1	Open DCB (M:GO, F:LINKIN)
CAL1, 1	Print message

SUBROUTINES

LINKERR2	-	Print 'CANT FIND - RETYPE' message

ID

BINTOHEX - Convert binary to Hex (EBCDIC)

PURPOSE

Convert a word of binary to printable (EBCDIC) Hexadecimal.

USAGE

BAL, SR4 BINTOHEX
R7 - Binary number to convert
Return
D3, D4 - Printable (EBCDIC) Hexadecimal number.
Leading blanks supplied for zeros.

ID

BIN2BCD - Convert binary to decimal (EBCDIC)

PURPOSE

Convert a word of binary to printable (EBCDIC) decimal.

USAGE

BAL, SR4 BIN2BCD
R7 - Binary number to convert
Return
D3, D4 - Printable (EBCDIC) decimal number.
Leading blanks supplied for zeros.

ID

CHKDECLD - Search for Declaration

PURPOSE

Check the Declaration for definition or add to the REF/DEF stack chain. If the Declaration is defined, the value is returned.

USAGE

BAL, SR4 CHKDECLD
Secondary entry point
B HASVAL
BUF2 byte 0 - Displacement into BUF2 of current value word.
R5 - Expression control byte.
R7 - Declaration number.
Return
SR3 - number of addresses in TSTACK
R2 - address of REF/DEF stack pointer word
R7 - value or pointer for value word of Expression
TSTACK - address of REF/DEF stack word that points to Expression stack value word
R5 - Expression control byte - set defined if Declaration has a value.

INPUT

DECLBAS - Base address of Declaration stack
RFDFBAS - Base address of REF/DEF stack
EXPRSTK - Expression stack pointer double word

ID

CHKFREFD - Search Forward Reference stack

PURPOSE

Search the Forward Reference stack and add to the Forward Reference chain or return the value of a Forward-Reference-and-hold item, or add an entry.

USAGE

BAL, SR4 CHKFREFD

BUF2 byte 0 - Displacement into BUF2 of current value word.
R5 - Expression control byte
R7 - Forward Reference number

Return
SR3 - number of addresses in TSTACK
R2 - address of Forward Reference stack pointer word
R7 - value or pointer for value word of Expression
TSTACK - address of FREF word that points to Expression stack value word
R5 - Expression Control byte - set defined if Forward-Reference-and-hold

INPUT

MODNUM	-	current module number
FREFBAS	-	Forward Reference stack base
FREFN	-	Forward Reference stack number
EXPRSTK	-	Expression stack pointer double word

OUTPUT

FREFSTK - Forward Reference stack pointer double word

SUBROUTINES

STKOVF	-	Expand designated stack
HASVAL	-	Forward-Reference-and-hold exit

ID

CHK2EOF - check for two consecutive end of files.

PURPOSE

To determine when the end of an input element file has been reached.

USAGE

BAL, SR4 CHK2EOF

Return

+1 end of input element file
+2 further data found on the file

OUTPUT

CURBYTE - current ROM byte

SUBROUTINES

READBILI - read a ROM card.

ID

COREABN - Abnormal return opening a core library.

PURPOSE

Get more memory for the core library buffer in the case of a 07 abnormal. Exit LINK after an error message in the case of any other abnormal.

USAGE

Abnormal return from a core library open or read CAL1, 1.
SR3 - Abnormal code from CAL1, 1.

INTERACTION

CAL1, 1 Print I/O error reading core library
CAL1, 9 1 Exit CAL

OUTPUT

SYMBOLTB	-	address of core library DEF stack.
RDCORELB+3	-	address of core library DEF stack buffer
RDCORELB+4	-	size of core library DEF stack buffer

SUBROUTINES

MMSYMBTB - Get memory for core library DEF stack.

ID

ENDECL - Enter new entry into Declaration stack.

PURPOSE

Place a new entry into the Declaration stack and check for stack overflow.

USAGE

BAL, SR4 ENDECL
R5, R6 - contain Declaration stack entry.

INPUT

DECLN - Declaration stack number.

OUTPUT

DECLSTK - Declaration stack pointer double word.

SUBROUTINES

STKOVF - Expand designated stack.

ID

ENNAM - Search and enter new entries in the REF/DEF stack.

PURPOSE

Search the REF/DEF stack for a matching entry; if none found, add a new REF/DEF stack entry.

USAGE

```
BAL, SR4        ENNAM
Return
    R6 - pointer to REF/DEF stack entry
Secondary entry point
BAL, SR4        ENNAM11
    BUF2 - new REF/DEF stack entry.
    Put new entry from BUF2 into the REF/DEF stack.
Return
    R6 - pointer to REF/DEF stack entry
```

INPUT

RFDFBAS	-	Base address of REF/DEF stack.
RFDFN	-	REF/DEF stack number.

OUTPUT

BUF2	-	REF/DEF stack entry.
RFDFSTK	-	Stack pointer doubleword for REF/DEF stack.

SUBROUTINES

GBYTE	-	Get a ROM control byte
STKOVF	-	Expand designated stack

ID

ENSYMTBL - Put an entry in symbol table.

PURPOSE

Put an internal symbol or global symbol table entry into the correct symbol table.

USAGE

BAL, SR4 ENSYMTBL
BUF2 - Contains symbol table entry.

INPUT

PLISTORG - pointer to current file
VDP - top of available memory +1

OUTPUT

SYMBOLTB - current symbol table origin

INTERACTION

CAL1, 1 Print 'IDDEF' message

SUBROUTINES

INTRNAME - print symbol table name
MMSYMBTB - obtain memory for symbol tables

ID

EVEXP - Evaluate designated expression.

PURPOSE

Evaluate the designated expression in the Expression stack. If the expression can be evaluated, satisfy all chains requiring the expression result.

USAGE

```
BAL, SR4        EVEXP
    R6 - point within expression stack to designated expression
Return
    +1   normal return (error return for ORIGIN)
    +2   normal return for ORIGIN
    R7   value of expression for ORIGIN
```

INPUT

TEMPN - Temp stack number
EXPRBAS - Base address of expression stack

OUTPUT

EXPRSTK - Expression stack pointer double word

SUBROUTINES

STKOVF - Expand designated stack.
RFDFCHN - Define a REF

ID

EXPRIN - Read and build an expression

PURPOSE

Read the expression control bytes from the ROM cards and build an expression in BUF2.

USAGE

BAL, SR4 EXPRIN
R6 - Destination type (zero no destination - origin)
R7 - Destination address
If no destination, return as follows:
+1 error return - expression not defined
+2 R7 - value of expression

INPUT

BUF2SIZ - Size of BUF2

OUTPUT

BUF2 - Build expression area

SUBROUTINES

EVEXP - Evaluate the expression
GBYTE - Get a ROM byte
CHKFREFD - Search Forward Reference stack
12BNUM - Get a one or two byte declaration number
CHKDECLD - Search for declaration
3BNUM - Get 3 bytes of ROM data

ID

EXPTTB - Put expression into expression stack

PURPOSE

Move an expression from BUF2 into the expression stack. Expand the expression stack if necessary.

USAGE

BAL, SR4 EXPTTB
BUF2 - The expression to go into the expression stack.
Return
R6 - pointer within expression stack to the expression

INPUT

EXPRBAS	-	Base address of expression stack
EXPRN	-	Expression stack number

OUTPUT

EXPRSTK	-	Expression stack pointer double word.

SUBROUTINES

STKOVF - Expand designated stack.

ID

GBYTE - Get a ROM control byte.

PURPOSE

Read a control byte from the ROM card.
Sequence check and checksum each ROM card.

USAGE

BAL, SR4 GBYTE
Return
R5 - ROM control byte in low order byte

OUTPUT

CURBYTE	-	current ROM byte
RCDSIZE	-	current logical ROM record size
LASTCARD	-	last ROM card indicator
SEQNUM	-	ROM card sequence number

INPUT

BUF - ROM card buffer

SUBROUTINES

READBILI - Read a ROM card

ERRORS

Internal error symbol

(PA7)	-	Illegal data format
(PA5)	-	Sequence error
(PA6)	-	Checksum error

ID

INCLOC - Increment load location counter
CHKLOC - Check load location counter limits

PURPOSE

To increment the load location counter and then check the location counter to make sure it is within the DCB area, Data area, or Pure Procedure area.

USAGE

BAL, SR4 INCLOC
R6 - Increment value or bytes.
To only check load location limits (i.e. after ORIGIN evaluation)
BAL, SR4 CHKLOC

INPUT

VPP	-	Pure Procedure origin.
MINSYMTL	-	Origin of largest internal symbol table
SYMBOLTB	-	Origin of current symbol table
BACORE+1	-	Address + 1 of memory obtained for Pure Procedure area
MXPP	-	Maximum load address of Pure Procedure area
VLC	-	Data origin
BASES+1	-	Lowest stack address
BACOREDA+1	-	Address + 1 of memory obtained for Data area.
DECLBAS	-	Base address of Declaration stack
VDCB	-	Origin of DCB area
BADCB	-	Load location counter of DCB area
DCBBIAS	-	Difference between DCB Execution address and load address
MAXLOC	-	Maximum load address of Data area

OUTPUT

LOC - Current load location counter
VIRTUAL - Address for Get Virtual Page CAL.

SUBROUTINES

PRESSTK - Remove all space from stacks.

INTERACTION

CAL1,8 - Get Virtual Page

ERRORS

Internal error symbol

(PA8) - current load location counter out of legal range.

(PA4) - stacks overflow into Data area.

ID

INSERTLB - Insert core library name

PURPOSE

Insert the core library name into the OPEN and READ PLISTs in order to obtain the proper core library RFDFSTK. Also insert the core library name into the HEAD record.

USAGE

BAL, SR4 INSERTLB
R3 - Upper half word must be zero,
lower half word - EBCDIC of core library
J0, J1, . . . , J9, P0, P1, . . . , P9

OUTPUT

OPNCORL1	-	File name for OPEN core library RFDF stack
PNSIZE	-	File name for OPEN core library HEAD
COREKEY	-	Key name for READ core library RFDF stack
CORENAME	-	Core library procedure name in HEAD record
PLISTBUF	-	Pointer to next file Flag word

ID

INTRNAME - Print symbol table name

PURPOSE

Print the name of a symbol table entry as part of the error or warning messages IDDEF and IUSAT

USAGE

BAL, R7 INTRNAME
BUF2 - Symbol table entry

INTERACTION

CAL1, 1 - print the symbol table name

RESTRICTIONS

The symbol table entry in BUF2 is not restored after printing the name.

ID

LINKERR - LINK command error

PURPOSE

Entry is made when an error has been detected during the scan of the LINK or RUN command.

USAGE

```
B       LINKERR
    R2 - character position within LINK or RUN command where error was detected.
Return
    CAL1,9  1       EXIT back to TEL
```

SUBROUTINES

PRINTQ - Prints Eh ? @ n message on user's terminal.

ID

LINKERR2 - Print 'CANT FIND - RETYPE' message

PURPOSE

This routine is entered when an input file can not be opened.

USAGE

Abnormal return on the CAL1, 1 open ROM input.

INTERACTION

CAL1, 1 Print message on user's terminal.
CAL1, 1 Read new file NAME. ACCOUNT. PASSWORD from user's terminal.

SUBROUTINES

SCAN	-	Scan routine for commands
PRINTQ	-	Print Eh ? @ n message
FPTLOAD2	-	Print current input file name

ID

MMDA - Get memory for Data area

PURPOSE

To obtain the correct Virtual page(s) to load Data.

USAGE

BAL, SR4 MMDA

INPUT

BACOREDA - Load Location Counter for the Data area.

OUTPUT

BACOREDA+1 - address + 1 of the memory obtained for Data area.
VIRTUAL - address for Get Virtual Page CAL.

INTERACTION

CAL1, 8 - Get Virtual Page

ID

MMDCB - Get memory for DCB area

PURPOSE

To obtain the correct Virtual Page(s) to load DCBs.

USAGE

BAL, SR4 MMDCB
Return
SR1 - address of first word in previous page if new page obtained; otherwise current page.

INPUT

DCBBIAS	-	Difference between DCB execution address and load address
BADCB	-	Load location counter of DCB area
VPP	-	Pure Procedure origin

OUTPUT

BADCB+1	-	address + 1 of memory obtained for DCB area
VIRTUAL	-	address for Get Virtual Page CAL

INTERACTION

CAL1, 8 - Get Virtual Page

ERRORS

Internal error symbol
(PA9) - more than 2 pages for DCBs have been requested

ID

MMPP - Get memory for Pure Procedure area

PURPOSE

To obtain the correct virtual page(s) to load Pure Procedure.

USAGE

BAL, SR4 MMPP

INPUT

BACORE - Load location counter for the Pure Procedure area

OUTPUT

BACORE+1 - address+1 of the memory obtained for Pure Procedure area.
VIRTUAL - address for Get Virtual Page CAL.

INTERACTION

CAL1,8 - Get Virtual Page

ID

MMSYMBTB - Get memory for symbol tables and core library REF/DEF stack

PURPOSE

To obtain the correct virtual page(s) for the Internal and Global Symbol tables. Also obtain memory for the core library REF/DEF stack.

USAGE

BAL, R0 MMSYMBTB

INPUT

SYMBOLTB - current symbol table origin

OUTPUT

SYMBASE - lowest address+3 of memory obtained for tables.
VIRTUAL - address for Get Virtual page CAL.

INTERACTION

CAL1, 8 Get Virtual Page

ID

MMTBASE - Get memory for the stacks

PURPOSE

To obtain the correct virtual page(s) for the internal load stacks.

USAGE

BAL, R0 MMTBASE

INPUT

BASES+1 - Lowest stack address

OUTPUT

STKBASE - lowest address of memory obtained for stacks
VIRTUAL - address for Get Virtual Page CAL.

INTERACTION

CAL1, 8 - Get Virtual Page

ID

PRESSTK - Remove space from stacks

PURPOSE

Remove all space from all the stacks. The only exception is the stack TSTACK. A minimum space is retained because TSTACK is used with no overflow logic.

USAGE

```
BAL, R0      PRESSTK
```

INPUT

NSTKS - number of stacks
TEMPN - temp stack number

OUTPUT

BASE - Table of stack pointer double words
BASES - Table of stack base addresses

ID

PRINTQ - Print Eh ? @ n message

PURPOSE

To print one line Eh ? @ n message on user's terminal.

USAGE

```
BAL, SR2        PRINTQ
     R2 - Value of n
```

INTERACTION

CAL1, 1 print a line on user's terminal

SUBROUTINES

BIN2BCD - Convert binary number to EBCDIC

ID

RAMRABN - Abnormal return reading ASSIGN MERGE record.

PURPOSE

To print 'I/O ERROR READING ASSIGN MERGE RECORD' message

USAGE

Abnormal return from the ASSIGN MERGE read CAL1, 1.

INTERACTION

CAL1, 1 - print 'I/O ERROR READING ASSIGN MERGE RECORD'
CAL1, 9 1 EXIT CAL

ID

READBILI - Read a ROM card

PURPOSE

Read a ROM card either through F:LINKIN DCB or the M:GO DCB.

USAGE

BAL, SR4 READBILI

INPUT

PLISTORG - Pointer to current file Flag word
M:GO - Input DCB
F:LINKIN - Input DCB

OUTPUT

BUF - ROM card buffer
FIRSTRD - First read SWT (rewind if EOD)

ERRORS

Internal error symbol
(PA7) - ILLEGAL DATA FORMAT

ID

REFSATY - Satisfy REFS with zero

PURPOSE

After the MAP has been printed all remaining unsatisfied REFs are satisfied with zero. This removes the core chain addresses from the program thus cleaning up unsatisfied references to zero.

INPUT

RFDFBAS - Base address of REF/DEF stack
RFDFSTK - Stack pointer double word for the REF/DEF stack

SUBROUTINES

RFDFCHN - Define a REF

ID

RELSYM - Release memory from symbol table area

PURPOSE

To release the page(s) from the Internal and Global symbol table area. Also release memory for the core library REF/DEF stack area.

USAGE

BAL, SR4 RELSYM

INPUT

VDP - Top of available memory + 1.

OUTPUT

SYMBASE - Lowest address + 3 of memory obtained for tables.
VIRTUAL - Address for Release Virtual Page CAL.

INTERACTION

CAL1, 8 - Release Virtual Page

ID

RFDFCHN - Define a REF

PURPOSE

Satisfy a REF chain through both the expression stack and program.

USAGE

BAL, SR4 RFDFCHN
SR1 - Value of definition
SR3 - Pointer into REF/DEF stack of entry to define
Top entry of TSTACK - resolution of value in SR1

INPUT

RFDFBAS - Base address of REF/DEF stack.

SUBROUTINES

EVEXP - Evaluate designated expression

ID

SCAN - LINK/RUN command scan

PURPOSE

Scan command buffer until terminator or end of command is reached.

USAGE

BAL, SR4 SCAN

INPUT

R0 - Address of command buffer
R1 - Remaining size of input command (ARS)
R2 - Byte displacement into command of next field
R3 - Address where field is to be moved

OUTPUT

R6 - Termination character in EBCDIC
R7 - Number of characters in field – exclusive of terminator.
R1-R2 - Updated

DATA BASES

TERMS - Termination character set used by LINK/RUN commands.

ERRORS

LINKERR - a) no command characters remain
b) field length too long (greater than 10 characters)

ID

STKOVF - Stack overflow handler

PURPOSE

Expand the designated stack. If Data area overlaps stacks call PRESSTK to remove all excess space from all the stacks then expand designated stack.

USAGE

BAL, R4 STKOVF
R1 - Stack number of stack to expand

INPUT

ADONSIZ - Table of add-on stack sizes
MAXLOC - Maximum load address of Data area

OUTPUT

BASE - Table of stack pointer double words
BASES - Table of stack base addresses

SUBROUTINES

MMTBASE - Get memory for the stacks
PRESSTK - Remove all space from stacks

ERRORS

PA4 - Stacks overflow into Data area

ID

SYSLBABN - Abnormal return opening SYS library

PURPOSE

To print 'I/O ERROR LINKING SYSTEM LIBRARY' message

USAGE

Abnormal return from the system library (:BLIB. :SYS) open CAL1, 1

INTERACTION

CAL1, 1 print 'I/O ERROR LINKING SYSTEM LIBRARY'
CAL1, 9 1 EXIT CAL

ID

2BNUM - Get 2 ROM control bytes

PURPOSE

Read and pack into one register 2 bytes from the ROM input.

SUBROUTINES

GBYTE - Get a ROM control byte.

USAGE

BAL, SR4 2BNUM
Return
R7 - 2 bytes from ROM input in low order bytes.

ID

3BNUM - Get 3 ROM control bytes

PURPOSE

Read and pack into one register 3 bytes from the ROM input.

USAGE

BAL, SR4 3BNUM
Return
R7 -3bytes from ROM input in low order bytes.

SUBROUTINES

2BNUM - Get 2 ROM control bytes

ID

12BNUM - Get 1 or 2 bytes of ROM input.

PURPOSE

Read 1 or 2 bytes of ROM input depending on the number of declarations in the declaration stack.

USAGE

BAL, SR4 12BNUM
Return
R7 - 1 or 2 bytes from ROM input in low order bytes.

INPUT

DECLSTK - Declaration stack pointer doubleword.

SUBROUTINES

2BNUM - Get 2 ROM control bytes.

ID

!BATCH - Terminal Batch Entry Subsystem
Allan Ramacher

PURPOSE

The terminal batch entry subsystem controls the insertion of terminal jobs into the batch job queue of a symbiont UTS system.

USAGE

The calling sequence is a TEL command:

!BATCH fid ,fid ,fid .. fid_n

fid = file name . account .password

If the BATCH command has no argument field, TEL responds:

BATCH WHAT?

Otherwise, control is transferred to the BATCH subsystem with the TEL or CCI command line in J:CCBUF, the command ARS (Actual Record Size) in byte 0 of J:PUF and the index to the command argument field in byte 1 of J:PUF.

INPUT

The input for this processor is the edit file(s) specified in the command line. The job must conform to the requirements of any other batch job as it is indistinguishable from card reader batch job after its insertion in the input symbiont job stream. More than one job is permissible in a given file.

OUTPUT

The input symbiont buffer created by batch is described in Section VI.04.

INTERACTIONS

Monitor services used:

CAL1,8	GETPAGE	get 1 page of dynamic storage
CAL1,1	OPEN	open edit file
CAL1,1	READFILE	read edit file
CAL1,1	CMDMSG	echo card image that has an error
CAL1,1	BATCH	submit a batch job
CAL1,2	ADDFMSG	type job insertion message to OC
CAL1,1	OUTLIST	type ID message to terminal

CAL1,1	JOBSTAT	request job status
CAL1,1	MSG	type job status message to terminal
CAL1,1	CLOSFILE	close edit file
CAL1,8	FRGEPAGE	release page dynamic storage
CAL1,9	1	return to monitor

ERRORS

All error conditions except command rejected result in the jobs being ignored and not entered into the job queue.

Error conditions fall into 4 broad categories:

Command errors
Job errors
System errors
JOBENT errors

Command errrors and corresponding diagnostics:

EH?@n
Structure of Command did not conform to requirements (at character n).

Job errors and corresponding diagnostics:

MISSING JOB COMMAND
The first record of the terminal job must be a JOB Control command.

ILLEGAL ACCOUNT
The account on the JOB Control Command must match the user's logon account.

ILLEGAL NAME
The name on the JOB Control Command must also match the user's logon name.

ILLEGAL PRIORITY
The terminal batch job priority may not exceed the on-line priority.

BLANK NOT ALLOWED IN XACCT FIELD
The extended acctg. field on the JOB Control Command contains blanks.

XACCT FIELD NOT TERM. By RT. PAREN.
The extended acctg. field on the JOB Control Command is not terminated by a right parenthesis.

COMMAND REJECTED
The file contains a BIN or FIN Control Command.

DATA LOST
Batch expects card images and longer records are truncated.

System errors and corresponding diagnostics:

FILE READ ERROR
A file read error occurred and job must be restarted.

JOBENT abnormal codes (Corresponding diagnostics found in error message file);

3F3B
3F3C
3F3D
3F3E
3F3F

RESTRICTIONS

System Restrictions:
System must be symbiont.

User Restrictions:
A JOB Control Command must be the first record present and specify an account and name which match the user's logon account and name. (Found in J:ACCN and J:UNAME, respectively.)

The priority specified must be legal, i.e., not higher than allowed the user (found in bits 8-11 J:ABC, currently 7), and a hex number (1-F). Optionally, the specification field of the JOB Command maybe left blank, in which case BATCH will supply default entries.

DATA BASES

Internal tables include:

CCS	Control Command options

External data bases (JIT) See Section VA

J:CCBUF	Command line buffer
J:ABC	Maximum on-line priority (bits 8-11)
J:ACCN	user account
J:PUF	command ARS and argument index (byte zero and one, respectively)

SUBROUTINES

BATCHCAL	flag symbiont buffer complete and issue BATCH CAL
SYNTAX	report character position of syntax error
SCAN	return next argument field of a command
HEX2PRNT	convert hexadecimal to EBCDIC equivalent
BIN2BCD	convert binary to EBCDIC equivalent

DESCRIPTION

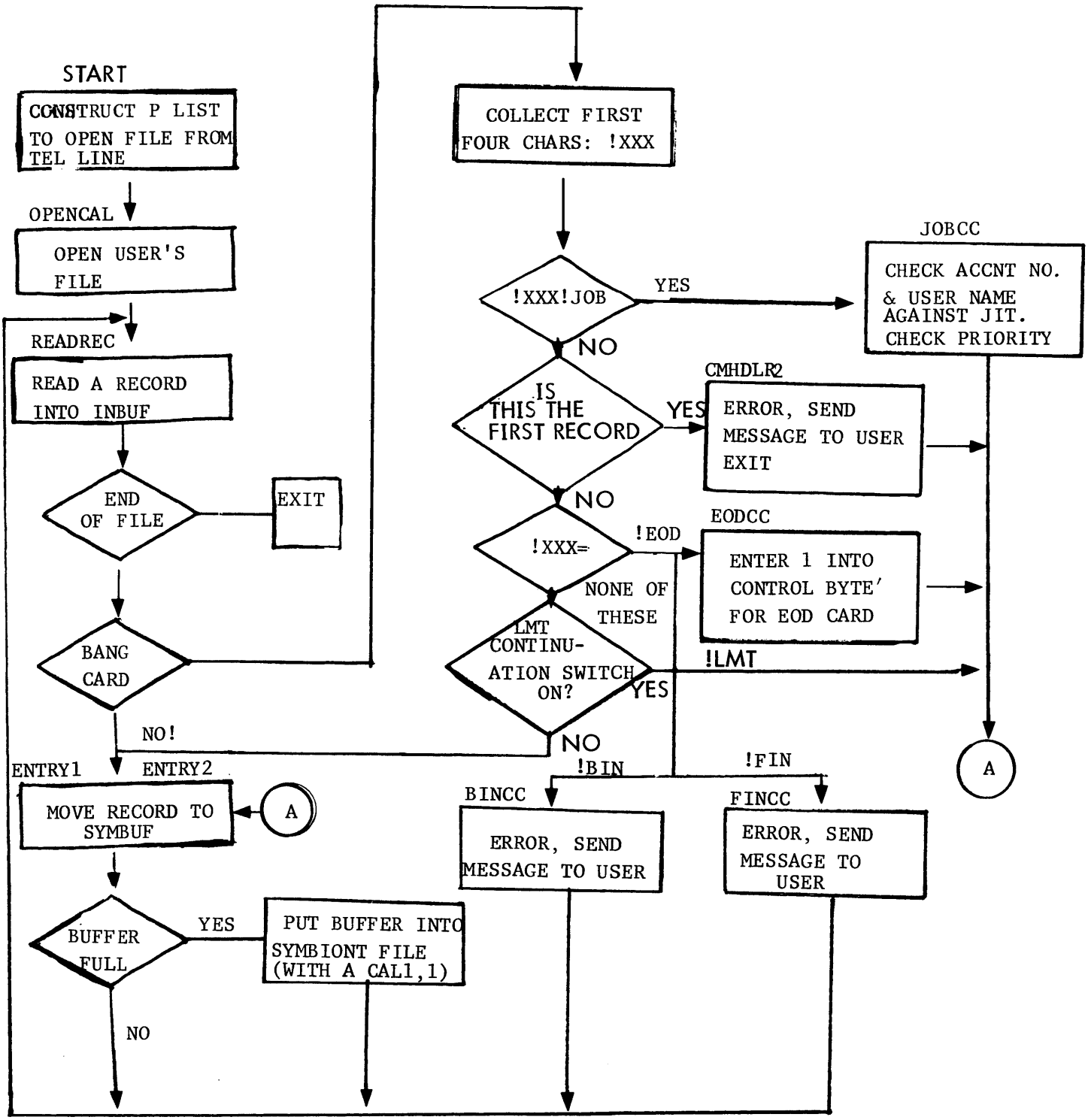
BATCH opens and reads the file(s) specified in the command line argument field. JOB Control Commands are scanned and illegal commands result in error notification. BIN and FIN control commands are rejected with notification. Each legal record is formatted into an input symbiont block. When the data block is full, the block is flagged as continued and is submitted by a Monitor service procedure (M:JOB) to the input symbiont area of the RAD. Subsequent JOB Control Commands or end of file signify job completion (the continuation/completion flag is in the M:JOB PLIST). At that point, M:JOB returns the job ID in SR1 and it is output through M:SL along with the time, date and relative queue position of the job.

Example:

ID = 041C SUBMITTED 12:14 MAR 03, '70

WAITING: 49 to RUN

UTS TECHNICAL MANUAL



ID

DEFCOM - Create load module containing DEFs only.
Frances Soskins

PURPOSE

The purpose of DEFCOM is to make a new load module from an old one. The new load module consists only of the HEAD and TREE records (modified to reflect the new sizes and locations of the new load module segments), and a modified REF/DEF stack. The input load modules are usually core libraries.

The new load module thus produced allows all the REFs of a ROM to be satisfied without having to load a whole core library load module, while making a load module of the ROM. Hence, DEFCOM makes the DEFs and their associated values in one load module available to another without the necessity of loading an entire core library.

USAGE

Before DEFCOM can be called, the old load module must be assigned to the element input DCB (M:EI) and the element output DCB (M:EO) must be designated for the output load module. Then DEFCOM is called by the following control command:

```
!DEFCOM
```

It is unlikely that DEFCOM will be used on-line; however, it can be used with SET commands and the !DEFCOM command.

INPUT

The only input to DEFCOM consists of the load module assigned to the M:EI DCB, usually a core library.

OUTPUT

The output from DEFCOM consists of a load module containing only the HEAD and TREE records and the REF/DEF stack, modified as explained under program DESCRIPTION below.

INTERACTION

DEFCOM makes use of the following monitor services:

M:OPEN	Used to open the M:EO DCB, preparing for keyed, direct, saved output.
M:READ	Used to read the HEAD and TREE records and REF/DEF stack of the load module assigned to the M:EI DCB.

M:GP Used to get page(s) for the reading in of the HEAD and TREE records and REF/DEF stack.

M:WRITE Used to write the HEAD and TREE records and the REF/DEF stack forming the new load module. This is, of course, via the M:EO DCB.

M:CLOSE Used to close both the M:EI and M:EO DCB's.

DEFCOM uses only the M:EI and M:EO DCBs.

DATA BASES

External:
DEFCOM uses the HEAD and TREE tables and the REF/DEF stack produced by LOAD, usually for a core library.

Internal:
DEFCOM uses the following data:

TXHEAD	The key for the HEAD table.
TXTREE	The key for the TREE table.
TX:PNDAT	The words, in TEXTC format, :PnDATA and several arithmetic constants for masking.

SUBROUTINES

There are no subroutines in DEFCOM.

ERRORS

DEFCOM writes no error (or any other) messages, nor are there any error returns. Any I/O errors will be processed as indicated in the corresponding DCB and not by DEFCOM. No error checks are made within DEFCOM of any kind.

RESTRICTIONS

None.

DESCRIPTION

DEFCOM reads the HEAD table through the M:EI DCB and obtains from there the size of the TREE table. The TREE table is then read and the size of the REF/DEF stack obtained from there. In order to flag the new load module as a DEFCOM load module, the first byte of the HEAD table is set to hex '83'. A dynamic page is then obtained (M:GP) to start the REF/DEF stack buffer. If a core library is being created, a :PN DEF entry is put at the top of the REF/DEF stack buffer and the size of the stack in the TREE table image is increased to include this first entry. The maximum REF/DEF stack size in the HEAD table image is also modified if it is not already large enough to include the additional entry. In order to determine whether or not a core library is

being created, the account field in the output DCB (M:EO + 27) is checked for ":SYS" and the file name (M:EO + 23 and 24) is compared with "PnDATA'.

The segment name and size are copied from the TREE table into a buffer which is used as a key to read the REF/DEF stack. The TREE table image is altered now to now to show that the sizes of control sections with protection types 00, 01, and 10, and the size of the expression stack are all zero. Additional pages are obtained to contain the entire REF/DEF stack and the REF/DEF stack is read in, starting at the first available page, and using the segment name as the key.

Each entry of the REF/DEF stack is examined to determine if it is a CSECT (in which case the evaluated bit is set and the size is set to 0), or an 03:PN entry (in which case the core library data size is put, in byte resolution, into the low order bits of the eighth word of the HEAD table image.) All other entries are unmodified.

DEFCOM then writes out (M:WRITE, via M:EO), the new, modified HEAD table, TREE table, and REF/DEF stack. The M:EO and M:EI DCBs are closed, (M:EO is closed with a SAVE instruction), and DEFCOM exits to the Monitor.

ID

SYMCON - Symbol Table Control

PURPOSE

The purpose of SYMCON is to provide a user with a means to build a DELTA-format global symbol table, modify and/or list the external symbols of a load module. This load module must have been created by the overlay loader. SYMCON reads the REF/DEF and expression stacks and the TREE record of a load module and allows a user to change external symbol names, to list and delete external symbols, and to build and discard a global symbol table.

Three of the more common uses of SYMCON are to reconcile any possible conflicts of external symbol names between two load modules which have been or will be combined into one, to reduce the number of external symbols in load modules which cannot be combined because their tables of control information are too large, and to provide the load module with a global symbol table in DELTA symbol table format for use by DELTA during module execution.

USAGE

Before SYMCON can be called via batch, the load module file must be assigned to the element input DCB as a Keyed file. Then SYMCON is called by the following control command:

```
!SYMCON
```

When SYMCON is used as an on-line processor, the load module must be assigned using a SET command and called by typing in SYMCON.

As an on-line processor SYMCON does the following:

1. types out 'SYMCON HERE' initially.
2. prompts all command inputs with a '*' character.
3. understands the following characters as command terminators.

Hex	Character	
OC	FF	Form Feed
OD	CR	Carrier Return
15	NL/LF	New Line/Line Feed
1C	FS	Field Separator
1D	GS	Group Separator
1E	RS	Record Separator
1F	US	Unit Separator

4. responds to the 'END' command by writing out the updated load module on M:EI and returning control to the monitor (via M:EXIT).

5. suppresses certain output to M:LO - namely, the copy of the command line, and various page-ejects. All diagnostic messages, as well as the map produced by the 'LIST' command are directed to M:LO.

There are seven SYMCON commands: LIST, DELETE, KEEP, CHANGE, BUILD, DISCARD and END. The functions of these commands are:

1) LIST - produce a load map.
2) DELETE - delete the specified symbols.
3) KEEP - delete all symbols except those specified.
4) CHANGE - rename a symbol.
5) BUILD [(LIB)] - build a Delta - format global symbol table with inclusion of library DEFS being optional.
6) DISCARD - discard a Delta-format global symbol table.
7) END - exit from SYMCON when running on-line.

The formats of these commands are given in Table SE-1. Blanks may be used within commands but may not be imbedded within a command verb or symbol. A command is terminated by the end of the input record or by a period. In batch mode, a SYMCON command may be continued from record to record (card to card) by the use of a semicolon, in which case the continuation record begins with the first character.

TABLE SE-1 SYMCON COMMANDS

Command	Description
LIST	Lists the external symbols of the load module in the same format as the load map. (The order may be different than that produced by a load map, and there may be some additional control sections listed corresponding to items (such as DCBs) obtained from the library.
DELETE name [, name]... [, name]	Deletes the specified symbols. Any DEF symbol in the module load map may be deleted unless it enters the definition of an item that is not completely defined and is either a DEF symbol or a forward reference. Forward references do not appear in the load map.
KEEP name [, name ... [, name	Deletes all symbols except: 1. DEFs listed in this command 2. DEFs that help define symbols listed in this command. 3. DEFs defined in terms of unsatisfied references (and used)
CHANGE $name_1/name_2$ [, $name_1/name_2$]... [, $name_1/name_2$]	Renames symbols. This command may be used to operate on any item with a name (DEF, SREF, PREF, DSEC). $Name_1$ is the name of the symbol to be changed and must be in the module. $Name_2$ is the name to be given to the symbol identified by $name_1$ and must not already exist in the module.
BUILD [,(LIB)]	Builds a global symbol table which contains all of the names and values of DEFS in the load module's REF/DEF stack. Library DEFs are included if the (LIB) option is specified. Names will be necessarily truncated to DELTA's seven character format. If two or more names are found which are identical in the first seven characters and which qualify for inclusion in the symbol table, only the first is included.
DISCARD	Deletes an existing global symbol table from the load module.
END	Terminates SYMCON when used as an on-line processor.

OUTPUT

Printed output from SYMCON (LIST command and all informational and error messages) goes to the system LO device.

SYMCON reads the TREE record and the REF/DEF and expression stacks, processes each command independently, then rewrites these same records providing no major errors are encountered. The old load module is always overwritten unless an abort occurred, in which case it is not. The execution of each command is independent of any other command. Thus, the configuration of the load module after the execution of one command is what is seen by the next command.

Any alteration of the REF/DEF stack by the 'DELETE', 'KEEP' or 'CHANGE' commands is not reflected in the global symbol table, if one already exists, until another 'BUILD' command is executed.

In addition to the updated load module and a load map, SYMCON also outputs the informational message, THESE SYMBOLS WERE DELETED name, name...name. This message includes all deleted symbols including deletions caused by other deletions.

INTERACTION

SYMCON uses the following monitor services:

M:SETDCB	Used to set an error address in the user's DCB M:EI for SYMCON processing.
M:READ	Used to read the TREE table and the REF/DEF and expression stacks of the load module and to read the commands to SYMCON.
M:DEVICE	Used to space on the M:LO DCB.
M:WRITE	Used to write the error and informational messages, the load map (from a LIST command), and the updated load module.
M:EXIT	Used to return to the Monitor after normal SYMCON completion.
M:XXX	Used to return to the Monitor after an unrecoverable error so that the job is aborted.
M:GP	Used to get memory pages for reading in the load module.
M:TRAP	Used to ignore fixed point arithmetic overflow.
M:OPEN	Used to verify that the specified load module exists.
M:DELREC	Used to delete the global symbol table record from the load module.
M:CLOSE	Used to close the M:LO DCB after printing messages.

SYMCON uses the DCBs M:SI, M:EI, and M:LO.

DATA BASES

External:
SYMCON uses the JIT, HEAD and TREE table and the REF/DEF and expression stacks produced by LOAD.

Internal:
SYMCON includes the following data:

SYMTAB	A 64-word table containing bytes which are picked up to determine if an input character is an illegal character, a legal symbol character, a blank, a comma, a slash, a command terminator or a semicolon.
HEAD	The HEAD key for the HEAD record of the load module.
TREE	The TREE key for the TREE record of the load module.
HEXBCD9	A 16-character table used for character conversion of hex numbers by direct table lookup.

and, in TEXTC format, 16 error and informational messages.

SUBROUTINES

SYMCON contains the following subroutines and modules:

CBINT	This routine interprets expression stack control bytes. It skips terms which do not point to the REF/DEF stack.
CHARNX	This is character scanner for input commands. It gets the next character and its type and puts them in registers for use by SYMNEX (below). The table SYMTAB (described under DATA BASES, above) is used to determine the character's type.
EXNEXT	This routine sets a register to the next item in the expression stack or to the start of the expression stack, depending on a particular register's contents at entry, and initializes the control byte interpreter (CBINT, above).
HEXBCD	This routine converts a hex value to EBCDIC for output by table lookup. Leading zeroes are suppressed. This routine uses the HEXBCD9 table described under the section on DATA BASES.
PRINT	This routine is used to print (using M:WRITE with the M:LO DCB) a symbol and a message on two lines.
PRINT1	This routine is used to print a message only on one line, as in PRINT, above.
RDNEXT	This routine sets a register to the next item in the REF/DEF stack or to the start of the REF/DEF stack, depending on a particular register's contents at entry.

RDSRCH — This routine locates a particular symbol in the REF/DEF stack and returns with the address of the symbol in a given register or returns so as to indicate the symbol was not in the stack. It calls TEXCOM to compare two names to find a match.

RESCOM — This routine determines the resolution of an item in the REF/DEF stack and stores the value and byte displacement upon returning.

SPACE — This routine upspaces a given number of lines by using the PRINTI routine to print blank lines (M:LO DCB).

SYMNEX — This routine scans the next symbol from the input command buffer and uses CHARNX, above, to obtain the character and type, which are returned in registers.

TEXCOM — This routine compares two TEXTC names. The inputs are the addresses of the two names. The return from TEXCOM indicates whether the two names were or were not equal. TEXCOM is called from RDSRCH.

ERRORS

The error messages in Table SE-2 are produced by SYMCON. Those flagged with asterisks are printed through the M:LO DCB just prior to an M:XXX (Monitor, abort this job) exit. The other messages cause no unrecoverable problems and the job continues.

TABLE SE-2 SYMCON ERROR MESSAGES

Messages	Description
name ALREADY IN STACK, CHANGE NOT MADE	An attempt was made to change the name of an item to a name currently used by another item.
name APPEARS AS TYPE OTHER THAN DEF NO ACTION	The symbol was a PREF, SREF, or DSEC and could not be deleted.
*COMMAND CONTAINS ILLEGAL CHARACTER	The command contained a character not in the character set defined for METASYMBOL.
*INCORRECT SYNTAX	Command syntax was incorrect.
*INCOMPLETE COMMAND LOAD MODULE UNCHANGED	This message indicates that a continuation was specified (with a semicolon), but the end of file was encountered when an attempt was made to read another card.
NO DELETIONS RESULTED FROM THIS COMMAND	None of the symbols listed caused any deletions. The load module is unchanged.
name NOT FOUND IN REF/DEF STACK	The identified symbol did not exist as an external symbol in the load module.
*REQUIRED CORE SPACE NOT AVAILABLE	This message indicates that the M:GP procedure failed to supply enough operating space for the processor.
THESE SYMBOLS WERE DELETED name name... name	This message includes all deleted symbols, including deletions caused by other deletions.

Message	Description
name USED IN UNEVAULATED EXPRESSION, NOT DELETED	This message indicates that the symbol was used to define an item that depended on an external reference. The item may have been a DEF, a forward reference, or a core location of the object code.
*INPUT M:EI FILE NOT STANDARD LOAD MODULE	This message indicates that the file read (HEAD and TREE record) was not that of a LOAD-created keyed load module. SYMCON operates only on files which are such load modules.
DELTA SYMBOL TABLE ALREADY IN LOAD MODULE, NO ACTION TAKEN	This message indicates that the global symbol table word of the load module's head record is non-zero.
NO SYMBOLS FOR DELTA SYMBOL TABLE, TABLE NOT BUILT	This message indicates that after scanning the REF/DEF stack, there are no symbols which qualify for inclusion in the global symbol table.
NO DELTA SYMBOL TABLE TO DISCARD, NO ACTION TAKEN	This message indicates that the global symbol table word of the load module's head record is zero.
ILLEGAL OPTION	An option other than LIB was specified on the BUILD command.
OVERLAY PROGRAM, DELTA SYMBOL TABLE BUILT FOR ROOT ONLY	This messages informs the user that a global symbol table was built for only the root of an overlay program.

RESTRICTIONS

SYMCON language and terminal conventions are essentially the same as those for TEL. It might be valuable to reiterate here that SYMCON operates properly only on load modules and only as described in the paragraph on USAGE, above. DEFs are the only symbols that can be deleted.

DESCRIPTION

SYMCON first uses M:SETDCB to set the M:EI DCB error address to the address "WRONG" (below), an error processing part of this program. The TREE record of the load module is read (M:READ) into a 12-word buffer to determine whether the file was keyed correctly as a load module. The HEAD record of the load module is read in and word 0 (byte 0) is checked to verify that the load module was formed by the Overlay Loader. If the file was not keyed correctly or was not formed by the overlay loader, SYMCON braches to WRONG where the message "INPUT M:EI FILE NOT STANDARD LOAD MODULE" is printed via (PRINT1) and the monitor abort exit is taken. Otherwise, it is assumed to be a proper load module and the size of the REF/DEF stack is obtained from the TREE record. This size is rounded, another page is added and this value comprises the PLIST to M:GP which is called to obtain memory pages for reading in the REF/DEF stack. If enough space was not available, the message "REQUIRED CORE SPACE NOT AVAILABLE" is printed (via PRINT1) and the M:XXX abort exit is taken.

Otherwise, the REF/DEF stack is read from the M:EI DCB (via M:READ) into the specified area, and a count of the number of items in it is made (using RDNEXT) and saved. The expression stack size from the TREE record is saved. The sum of this value and the number of items in the REF/DEF stack, rounded to the next highest page, constitutes the PLIST to M:GP, called to get memory for the expression stack. If the expression stack is not void, it is read (via M:READ), the expression stack size is added to the start address, and this sum is saved. The M:DEVICE Cal is used to place the M:LO device at the top of the form, if user is not on-line and SYMCON branches to COMPROC which reads (M:READ from M:SI) one control record, prints the command image (using PRINT1), and determines whether a DELETE, KEEP, CHANGE, BUILD, DISCARD, END, or LIST or other command was read by calling SYMNEX to scan the command verb. The symbol scan flag and the command start flag are cleared, and control is transferred to DELETE, KEEP, CHANGE, BUILD, DISCARD, FINISH, or LIST below, if the command was one of these, or back to COMPROC if it was not. If an END command, end of file or a bad control record was encountered, control passes to "FINISH".

If FINISH is entered because a bad control record was read, the message "INCOMPLETE COMMAND, LOAD MODULE UNCHANGED" is printed (using PRINT1) and the Monitor abort exit M:XXX is taken. Otherwise, the new expression stack size is stored into the TREE record image in SYMCON's buffer, and the new expression stack is overwritten on the old one (using M:WRITE). Then the new REF/DEF stack size is entered in the TREE record image, and the new REF/DEF stack is overwritten (M:WRITE), followed by the overwrite of the TREE record. Then the normal exit to the Monitor (M:EXIT) is taken.

DELETE

A symbol is obtained from the command buffer (via SYMNEX). If there are no more (or none), a check is made to see if any deletions have been made. If not, the message "NO DELETIONS RESULTED FROM THIS COMMAND" is output (via PRINT1) and control is returned to COMPROC. If deletions have been made, the message "THESE SYMBOLS WERE DELETED..." is printed (via PRINT1), and the names of all deleted DEFs follow. (A) Then new REF/DEF and expression stacks are built, according to whatever deletions were made (RDNEXT, EXNEXT, RDSTRT, CBINT, EXCNT, and EXSTRT are used to go through the stacks and create new ones), the sizes of the new stacks are saved, and a branch to COMPROC is made.

If there is a symbol, RDSRCH is branched to determine if this symbol is in the REF/DEF stack. If it is not, the message "symbol NOT FOUND IN REF/DEF STACK" is output (PRINT), and processing continues at the beginning of DELETE again. If the symbol was found but is not a DEF, the message "symbol APPEARS AS TYPE OTHER THAN DEF, NO ACTION" is output (PRINT) and processing continues at the beginning of DELETE once again.

If the symbol is a valid one to delete, it is entered into the deleted-symbols list (if not already there), and the expression stack is scanned for this symbol (using EXNEXT, and CBINT). If the symbol cannot be deleted (it does enter into the definition of an item that is not completely defined), the message "symbol USED IN UNEVALUATED EXPRESSION, NOT DELETED" is output (PRINT), the symbol is removed from the list of deleted symbols, and processing resumes at the start of DELETE again. Otherwise, the symbol is saved (and delete at completion of this command) and control goes to the beginning of DELETE again.

KEEP — The REF/DEF stack is searched (using RDNEXT) and all DEFs and forward references items are flagged. (B) A symbol is obtained from the command buffer (via SYMNEX), and if there are no more (or none), the expression stack is gone through (using EXNEXT) to see if the flagged items can be deleted. Those items which can be deleted are left flagged, and the flags of those which cannot, are cleared. Then a branch into the DELETE portion of SYMCON is taken, if any deletions have been made, at point (A), above. If a symbol was obtained, it is checked (with RDSRCH) to see if it is in the REF/DEF stack and if it is not, the message "symbol NOT FOUND IN REF/DEF STACK" is output (PRINT), and processing resumes at (B), above. If the symbol is found, its flag (to-be-deleted flag) is reset, and the next symbol is picked up at (B), above.

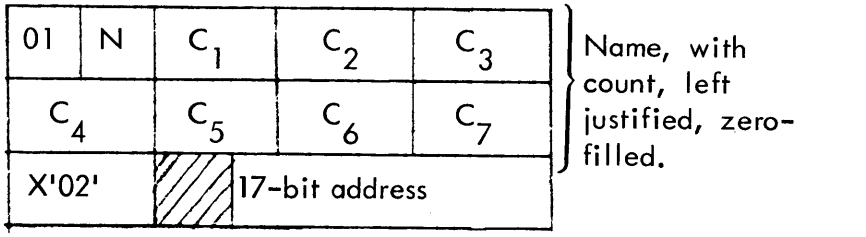
CHANGE — First, a correlation table is built from the REF/DEF stack using RDNEXT to get items one at a time. (C) Then a symbol is obtained from the command buffer (using SYMNEX) and checked for correct syntax (the name1/name2 format described in the section INPUT). If the delimiter in the command is not correct (a slash between two names, comma or period after), the message "INCORRECT SYNTAX" is output (PRINT1) and the Monitor abort exit is taken. RDSRCH is used to see if the first named symbol is in the REF/DEF stack. If not, the message "symbol NOT FOUND IN REF/DEF STACK" is output (PRINT), and scanning is restarted at (C), above, skipping to the next pair. Then RDSRCH is used to determine if the second specified name is in the REF/DEF stack; and if it is, the message "symbol ALREADY IN STACK, CHANGE NOT MADE" is output (PRINT) and the processing continues at (C). If both names are all right, the change is made and the REF/DEF stack is altered and repacked, if necessary. After the last pair of names has been read and processed as above, the expression stack is then adjusted accordingly, and a branch to COMPROC is taken.

LIST — The M:LO DCB is advanced to the top of a new page (using M:DEVICE). The REF/DEF stack is processed from top to bottom using RDNEXT and each DEF, SREF, PREF, DSEC, and CSEC and its name and its resolution (from RESCOM) or byte displacement are output (PRINT1), one to a line, like a LOAD-produced load map. At the completion of the REF/DEF stack, the M:LO DCB is advanced to the top of a new page (M:DEVICE) and LIST branches to COMPROC.

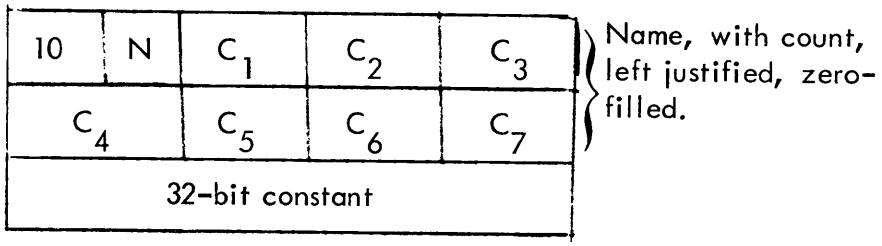
BUILD [(LIB)] — A check is made to determine whether a global symbol table already exists. If one doesn't, the remainder of the command image is scanned for the library option (LIB) and a flag is set accordingly to indicate

including or excluding of library DEFs. Two sweeps of the REF/DEF stack are made: The first of which is to establish the size of the global symbol table by counting the number of all DEFs (if LIB option is specified) or just non-library DEFs. (This may result in an over-estimate since no attempt is made in this sweep to exclude duplicate DEFs). Three words are reserved for each item, and the resulting number of pages are obtained for the global symbol table. In the second sweep of the REF/DEF stack, all library DEFs (if LIB option specified) or non-library DEFs are re-formatted into a Delta symbol table item, and included in the global symbol table if it's name is not a duplicate (to seven characters) if one already in the global symbol table. These items are three words long each, and divided into two categories, according to whether the value is a constant or relocatable address. The formats of these items are as follows:

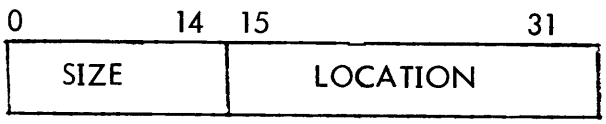
Location item:



Constant item:



The load modules head record (HEAD+7) is updated to reflect the size and location of the global symbol table

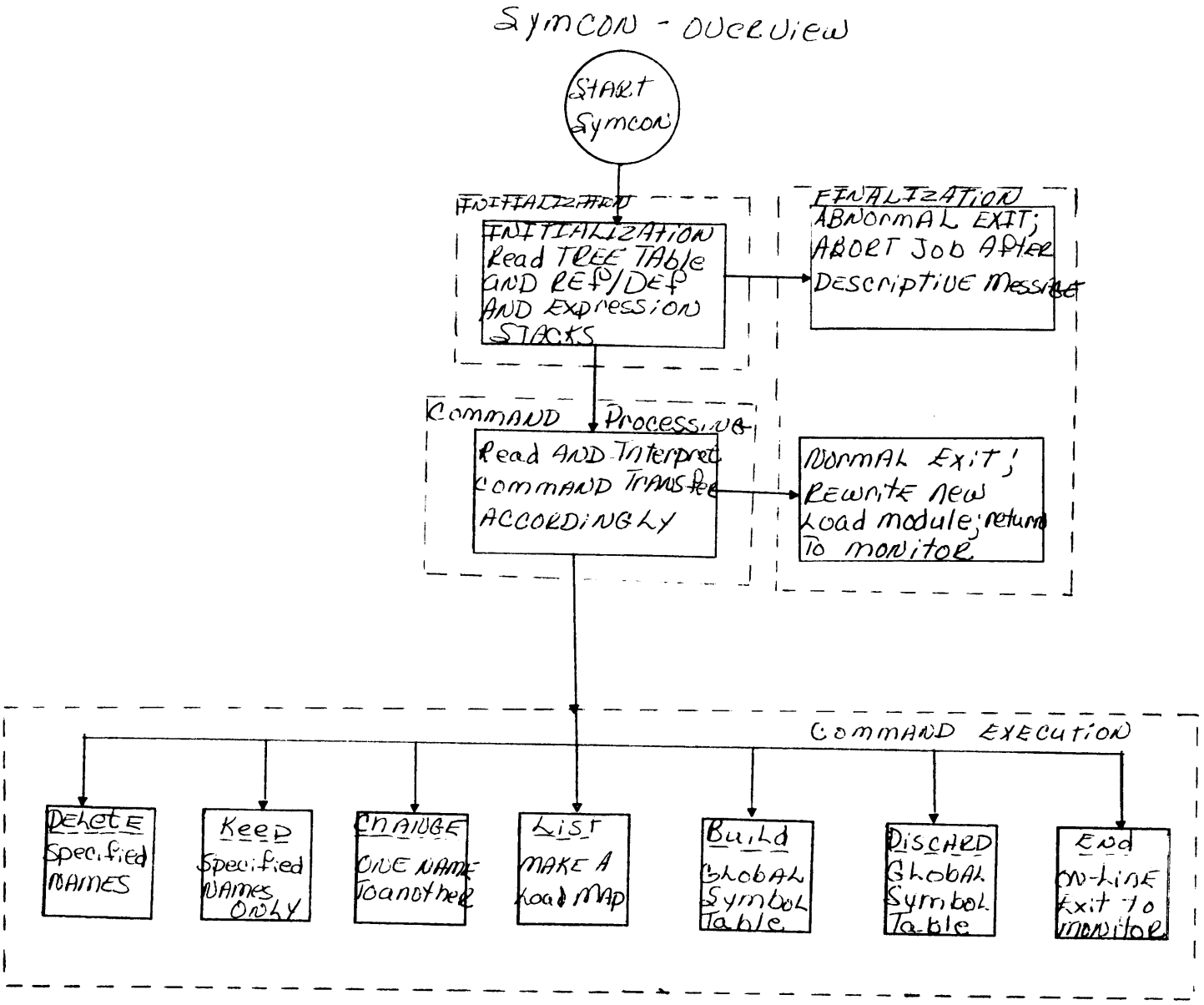


The record key for the global symbol table is formed by appending a X'09' to the root name found in the tree.

After all DEFs are processed, the head and global symbol table records are written out. If the global symbol table built was for an overlay program the user is informed that global symbol table was built for the root only.

DISCARD — The global symbol table is deleted by deleting the global symbol table record (via M:DELETE) and setting the global symbol table head record to zero.

END — A branch is made to the termination routine which sets up the key name for the expression stack record, updates the expression and REF/DEF stack size in the tree (TREE + 7, 9 respectively), writes out the expression and REF/DEF stacks and exits (M:EXIT).


SymCON - Overview
START SymCON
INITIALIZATION
INITIALIZATION Read TREE TABLE AND REF/DEF AND EXPRESSION STACKS
FINALIZATION
ABNORMAL EXIT; ABORT JOB AFTER DESCRIPTIVE MESSAGE
COMMAND PROCESSING
Read AND Interpret COMMAND TRANSFER ACCORDINGLY
NORMAL EXIT; REWRITE NEW LOAD MODULE; RETURN TO MONITOR
COMMAND EXECUTION
DELETE Specified NAMES
KEEP Specified NAMES ONLY
CHANGE ONE NAME TO ANOTHER
LIST MAKE A LOAD MAP
BUILD GLOBAL Symbol Table
DISCARD GLOBAL Symbol Table
END ON-LINE Exit TO MONITOR

ID

ERRMWR - Error Message File Writer

PURPOSE

ERRMWR creates and modifies the error message file ERRMSG (Section VM).

USAGE

! ERRMWR

INPUT

Input is read through the M:EI DCB and consists of error message cards and optional control cards as described in Chapter 11 of the UTS System Management Reference Manual (90 16 74).

OUTPUT

Output is the ERRMSG file containing a keyed record corresponding to each error message card. Trailing blanks are excluded and a new line character (X'15') is appended.

The key is one word long and has the form:

03	Binary values of columns - 1-6

Each error message card is listed through the M:LO DCB.

INTERACTIONS

M:DEVICE (HEADER)	establish a heading for list of error messages
M:DEVICE (PAGE)	page eject
M:OPEN	open or create ERRMSG
M:READ	read (through M:EI) deck of error message cards.
M:WRITE	write (through M:EO) a keyed record corresponding to each error message card
M:CLOSE	close ERRMSG
M:WRITE	write (through M:DO) error notification if card illegal
M:WRITE	write (through M:LO) list of error messages

ERRORS

Error notifications are printed immediately below the offending error message card image and the card is ignored.

*** INVALID CONTROL	a control card with an invalid code was detected.
*** INVALID HEX CODE	card column 1-6 was punched with a non-hex digit (except blanks).
*** ILLEGAL CONTINUATION	a second continuation was attempted.
*** UNABLE TO OPEN FILE 'ERRMSG': JOB ABORTED	ERRMSG was unavailable after 10 attempts to open.

DESCRIPTION

ERRMWR attempts to open the ERMMSG file in 'INOUT' mode. If the file does not exist, the mode is changed to 'OUT' and the file is created (in the user's account). If the file is busy, ERRMWR is put to sleep for one second and another attempt is made. Ten successive failures result in error notification and job abortion.

Once the file is open, a page eject and heading are sent to the LO device unless LO is assigned to the terminal, in which case a prompt character (>) is issued. Each error message card is read and printed unless LO is not assigned to the printer, in which case printing is suppressed.

Columns 1-6 are interpreted as a hex number and converted to a three byte key. A record corresponding to the text of error message is written (with NEWKEY overriden) into ERRMSG.

Processing is terminated by a !EOD control command, end of file return or a carriage return. At this point ERRMSG is closed and saved.

ID

MAILBOX - System Messages to Users

PURPOSE

The mailbox subroutine may be called by any processor with JIT access to place a message in a file named MAILBOX in any specified account. Log-on detects the presence of that file and notifies the logging on user that he has messages waiting.

USAGE

```
LI, D4      flag
LI, SR3     FPT
BAL, SR4    MAILBOX
```

If flag is zero the message is sent to the line printer only. If flag is nonzero, the message is sent to MAILBOX and the line printer.

FPT is the address of a two-word parameter list. The first word (FPT) contains the address of the TEXTC messages to be sent. The message length may not exceed 40 characters. The second word (FPT+1) contains the address of the two-word account number of the recipient.

All registers are saved and restored.

INPUT

The calling user's JIT is examined for the sender's account number which is included in the MAILBOX message.

OUTPUT

The message is formatted in J:CCBUF of JIT and then is added as a keyed record to MAILBOX (if D4 is nonzero) and is sent to the printer by M:PRINT. The key is the current julian date-time.

The message to the MAILBOX contains the sender's account number. The message to the line printer contains the recipient's account.

The message format is

hh:mm MON dd BY account Message text ....

hh:mm is the time in hours and minutes (24-hour clock).
MON dd is the month and day.
BY account is either sender or recipient as just described.
Message text is the message as sent.

INTERACTION

M:TIME is called to determine the current date-time.
M:PRINT sends the message to the line printer.
M:OPEN, M:WRITE, M:CLOSE add the message to the MAILBOX file.
M:WAIT CAL is issued if the MAILBOX is busy.

SUBROUTINES

Julian is called to format the date/time to be used for a key.

RESTRICTIONS

The calling processor must have C0 privelege to build a file in another account. The caller must insure that a Super Close is finally issued. The message text may not exceed 40 characters.

DESCRIPTION

MAILBOX calls M:TIME and Julian and formats the message in the buffer. If D4 is nonzero, the MAILBOX file in the recipient account is opened in update mode. If it does not exist, it is opened in output mode. If it is busy, the M:WAIT CAL is executed to wait for one minute and try again. If the file is still busy after two minutes, the MAILBOX writing is skipped. When the file is opened, the record is written with a key of the current julian date. If the key already exists, the key is incremented in the rightmost bit of the second word and the M:WRITE is executed again.

After the record is written in the MAILBOX file or if D4 is zero, the recipient's account is placed in the message and the message is sent to the printer via M:PRINT.

The JIT account and the registers are restored and exit is taken.

ID

SYSTEM UTS

PURPOSE

Under a single cover, SYSTEM UTS provides a set of frequently-used procs that have been found useful in UTS system development. Except for SYSTEM BPM, which is used in some processors, SYSTEM UTS is the only Meta-Symbol system proc used in UTS.

USAGE

SYSTEM UTS contains four subsets of procs that are selectively assembled by defining symbols prior to the call or SYSTEM UTS. The defining symbols and their default values are given below. A value of 1 for a symbol causes the corresponding set of procs to be assembled. A value of 0 causes it to be skipped.

UTSPROC	default value is 1.
S69PROC	default value is equal to the value assigned to UTSPROC.
ANSPROC	default value is 0.
MONPROC	default value is 0.

Note that the value of S69PROC is dependent on UTSPROC. As an example, the INITIAL module contains code that conflicts with that generated by the UTSPROC portion of SYSTEM UTS but it requires the proc generated by S69PROC. Thus, INITIAL is coded as follows:

```
UTSPROC   SET      0
S69PROC   SET      1
          SYSTEM   UTS
```

CONTENT

The following procs are described:

UTSPROC

BUMP, PUSH, PULL
ENABLE, DISABLE
HD, BD
OVERLAY, OVERTO
REMEMBER
DESTRUCT

MAP, UNMAP
OB, OBAL, OBSR4
SCREECH
SET, RSET
T:PUSHE
WORTAB, HAFTAB, BYTAB

S69PROC

LOAD, STORE, MODTST, COMPARE
LDMAP

ANSPROC

BIL, BOL
ANSB, ANSNB, ANSBAL
ERRABNCD
DAYS
SXP

MONPROC

CFU, DCB, Master Index Definitions
CLEAR
LIF

DESCRIPTION

Some of the procs in SYSTEM UTS were used in other SYSTEM proc decks in earlier versions of UTS. The procs generated when UTSPROC is set to 1 were formerly the complete SYSTEM UTS. The procs generated when MONPROC is set to 1 were formerly known as SYSTEM F00MON or SYSTEM MON. The procs generated when S69PROC is set to 1 are new in D00-UTS and are used to provide parameterized code for Sigma 7/9 compatability. The procs generated when ANSPROC is set to 1 provide parameterized code for ANS/ NONANS tape systems.

The following procs are defined when UTSPROC is set to 1.

BUMP, PUSH, PULL

These procs are used to modify TSTACK. BUMP modifies the TSTACK SPD by the amount specified by the first argument, using the register specified by the second argument

BUMP #, R

PUSH and PULL push or pull N words into TSTACK, as specified by the first argument, into or from registers starting at the second argument. Acceptable forms are

```
PUSH    R       PULL    R       (default N=1)
PUSH  1,R       PULL  1,R
PUSH  N,R       PULL  N,R       1<N<17
```

ENABLE, DISABLE

ENABLE generates a Write Direct instruction to reset the interrupt inhibit bits (C, I, and E) in the current PSD. DISABLE sets the bits.

HD, BD

HD and BD are function type procs that return the halfword and byte address of the argument, respectively. The HD proc generates the address AF(1)+AF(1); the BD proc generates HD(AF(1))+HD(AF(1)).

OVERLAY, OVERTO

OVERLAY and OVERTO are used to call a Monitor overlay and enter it at a specified entry point. OVERLAY remembers the calling segment, if any, to allow returning to that segment. OVERTO assumes that the segment called will not return, (BAL versus B). OVERLAY generates a BAL using register 11. Register 2 is loaded with the segment number and, if two arguments are specified, register 0 is used to contain an entry point index. Only one argument is required if register 0 is initialized by the calling program.

```
OVERLAY   Segment #, entry point index
OVERTO    Segment #, entry point index
```

These procs use subroutines T:OVERLAY and T:OVER in the T:OV module (see the UTS System and Memory Management Technical Manual, 90 19 86, for a description of these subroutines and a general overview of UTS Monitor overlays.)

REMEMBER

REMEMBER records the current overlay number and the contents of register 11 for use in returning from an overlay, but does not cause an overlay to be associated. REMEMBER is called by

```
REMEMBER
```

and generates a BAL, 14 to T:REMEMBER.

DESTRUCT

The DESTRUCT proc generates a calling sequence to the T:SELFDESTRUCT subroutine, in the T:OV module, to delete the user's current Monitor overlay.

MAP, UNMAP

The MAP and UNMAP procs generate calls to the MAP and UNMAP subroutine, which sets and resets the MAP bit in the current PSD by executing an LPSD. Register 1 is destroyed by the BAL to the subroutine. The MAP and UNMAP subroutines are contained in the ERHNDLR module.

OB, OBAL, OBSR4

These were the segload CALs for Monitor overlays in BPM/BTM. Segloading is no longer implemented in this way. They now generate respectively a B, a BAL, 11, and a B*11.

SCREECH

The SCREECH proc is used to load register 15 with a specified software check code number and branch to RECOVER. Nearly all occurrences of SCREECH in UTS have been replaced by in-line code. The proc is called by

SCREECH software check code number

SET, RSET

The SET and RSET procs encompass a large set of procs whose names are indicative of their function and which are used to manipulate user flag bits in the UH:FLG table. The general format of the SET, RESET proc is

$$[L]\ \begin{Bmatrix} \text{SET}[R] \\ \text{RSET}[S] \end{Bmatrix}[ST] \quad \text{FLAG, FR, IR}$$

where

L indicates that the flag is to be loaded first.

SET assumes the flag is reset and sets it with an AI instruction.

SETR makes no assumption and sets the flag with an OR instruction.

RSET assumes the flag is set and resets it with an AI instruction.

RSETS makes no assumption and resets the flag with an AND instruction.

ST causes the result to be stored back into UH:FLG.

FLAG is the flag to be set or reset. Flag names are defined in section VD.

FR is the register to contain the flag. The default value for FR is 15.

IR is the index register containing the user number. The default value for IR is 4.

An exception to the above is that the name SETT is used to generate the proc that, by convention, would normally be called SET.

SYSTEM UTS also contains REFs to the necessary data words for setting and resetting flags with AND or OR i.e.,

X1, X2, X4, X8, X10, X20, . . . , X8000

and

XFFFE, XFFFD, XFFFB, XFFF7, XFFEF, . . . , X7FFF

T:PUSHE

This proc is used to save an interrupted environment. T:PUSHE pushes six registers, beginning with register 13, into TSTACK to develop work space. It then loads 0 and 1 with the interrupted PSD and calls T:SAVE in the ERHNDLR module, using register 2, to complete the saving of the 19-word environment in TSTACK. The proc is called by

T:PUSHE PSD address

WORTAB, HAFTAB, BYTAB

These procs generate word, halfword, and byte tables, respectively. The procs are called by

WORTAB #, data
HAFTAB #, data
BYTAB #, data

where the first argument is the table length and the second argument is the data value to initialize the table.

The following procs are defined when S69PROC is set to 1:

LOAD, STORE, MODTST, COMPARE

These procs generate instructions for use in accessing data whose format is to be determined at load time. The instructions generated are of either byte or halfword

type, depending on the value of :9, which is an external reference. The format of the proc reference call is identical to that of standard Sigma 7 machine instructions. The procs and the generated instructions are

Proc reference		:9 = 0		:9 = 1	
LOAD,r	*a,x	LB,r	*a,x	LH,r	*a,x
STORE,r	*a,x	STB,r	*a,x	STH,r	*a,x
MODTST,r	*a,x	MTB,r	*a,x	MTH,r	*a,x
COMPARE,r	*a,x	CB,r	*a,x	CH,r	*a,x

These procs are generally used in UTS assemblies to access tables that contain physical page numbers that may vary between byte or halfword format, depending on the type of system. The value of :9 is defined during PASS2 of SYSGEN as a function of the BIG9 option.

LDMAP

The LDMAP proc generates a Load Memory Map version of the MMC instruction to load the memory map in either the Sigma 7 (byte) or Sigma 9 (halfword) type map, depending on the value of :9. The value of :9 is used to set bit 14 of the MMC instruction. The format of the LDMAP proc reference call is identical to the format of the MMC machine instruction. It is

```
LDMAP,r     a
```

The following procs are defined when ANSPROC is set to 1:

BIL, BOL

Following a CLM or CLR instruction, BIL will branch if effective word was within limits, BOL will branch if effective word was out of limits.

ANSB, ANSNB, ANSNOP, ANSBAL

These procs provide conditional branches or BALs depending on whether the system is ANS or not. A branch or NOP type instruction is generated based on the value, at load time, of the value of ANS, which is an external reference generated by PASS2 of SYSGEN.

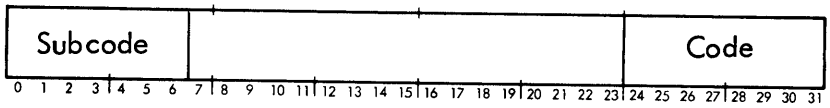
ANSB	AF	Branch if system is ANS; NOP otherwise.
ANSNB	AF	Branch if system is not ANS; NOP otherwise.
ANSNOP	AF	Branch if system is not ANS; NOP otherwise.
ANSBAL,R	AF	BAL if system is ANS; NOP otherwise.

ERRABNCD

The ERRABNCD proc produces a data word containing the specified error code and subcode for use in generating error messages. The proc reference call is

ERRABNCD code, subcode

The format of the generated data word is



DAYS

The DAYS proc generates a word table containing the day of the year each month begins on (e.g., March begins on the 60th day of the year, April on the 92nd day, etc.).

SXP

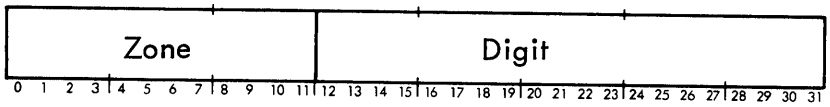
The SXP function converts a string of up to six EBCDIC characters into a 32-bit binary quantity. SXP is a function type procedure called as below:

DATA SXP('STRING')

where STRING is any string of up to six EBCDIC characters. The SXP function examines each of the six input characters as follows: Bits 0 and 1 of each character are discarded; bits 2 and 3 of each character are considered zone bits and are collected into the first 12 bits of the output value; bits 4 through 7 of each character are considered digits and are packed into bits 12 through 31 of the output value. For example, consider

DATA SXP('TAPE74')

The resultant data word will have the format



The EBCDIC representation of this character string is

E3C1D7C5F7F4

The right half of each byte is considered a decimal number and the DIGIT field formed by SXP is the binary representation of that number.

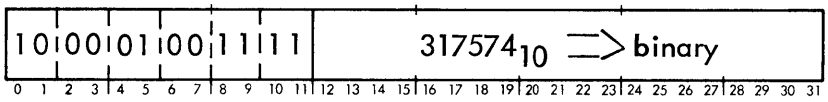
DIGIT = 317574_{10} expressed in binary.

The Zone field is formed by the two low-order bits of the leading half-byte of each character,

E = 11/10 Zone = 10
C = 11/00 Zone = 00

and so on.

In the example, the generated word would be



The following procs are defined when MONPROC is set to 1:

CFU, DCB and Master Index definitions:

Each major field item in these tables is defined by an EQU directive with the value of the items word displacement into the table. Additional selected items are defined with byte and halfword displacement by preceding the item name with BA or HA respectively. These field definitions are utilized by IOQ and the file management routines.

CLEAR

The CLEAR proc is used to zero registers 8 and 10. It generates the instructions

LI, 8 0
LI, 10 0

LIF

LIF merely generates a LI,15 AF(1) This proc is used by CALPROC to generate a table which is the target of an EXU instruction as part of the branch vector mechanism.

ID

SYSTEM BPM (for UTS)

PURPOSE

BPM procedures enable the programmer to invoke a number of monitor functions without the necessity of hand coding CALs and PLISTs.

System BPM for UTS was formed by merging E 02 SYSTEM BPM with the new PROCs required for UTS-specific monitor functions; BTM - specific PROCs were eliminated.

REFERENCE

All of the PROCs listed on the cover page of the UTS/BP Reference Manual (901769) are included in SYSTEM BPM. Note that the PROC name begins with 'M:'; this is a universal convention.

USAGE

The source file must contain the Metasymbol directive SYSTEM BPM in addition to SYSTEM SIG7 [F] [D] [P](or SYSTEM UTS which involves SYSTEM SIG7FDP).

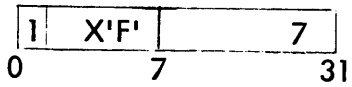
The general format for a procedure call is the following reference line:

LABEL	COMMAND	ARGUMENT
e.g.		
SLEEP, WAIT	M:WAIT	*7

In this example the programmer requires the suspension of program execution for the number of 1.2 second tics found in Register 7. SYSTEM BPM expands this reference line to generate:

```
                CAL1,8      ADD
```

where ADD points to the FPT



Two symbol addresses are available to reference the object code: SLEEP and WAIT. SLEEP references the instruction line (the CAL) and WAIT references the PLIST generated by the PROC (the ADD).

ERRORS

Error messages are printed on the LO device immediately below the object line that resulted in error notification.

Message	Description
ILLEGAL BUF/PARAM - RETURN ADDRESS	Invalid address expression specified in PROC reference line. (SL, 3)
UNRECOGNIZED KEY	Invalid Keyword specified in PROC reference line. (SL, 3)
IMPROPER # OF AFS	The number of argument fields specified in the PROC reference line was out of range. (SL, 3)
PARAMETER CONFLICT	Conflicting parameter specified in PROC reference line, e.g. 2 different file modes. (SL, 1)
ILLEGAL ARGUMENT	An alternate form of 'IMPROPER # OF AFS'. (SL, 3)
INDIRECT NOT ALLOWED	Indirect addressing is specified in PROC reference line but is not implemented. (SL, 1)
ILLEGAL AFA	Alternate form of 'INDIRECT NOT ALLOWED'. (SL, 3)
DCB MISSING	DCB is required but not specified. (SL, 7)
UNRECOGNIZED RELATIONAL	A relation other than GT, LT, EQ, GE, LE or NE was specified. (SL, 3)
TEXT PARAMETER ERROR	The argument specified could not be used to generate a text string. (SL, 3)
UNDEFINED OP LABEL	An unrecognized system operation label was specified. (SL, 3) Reference: Table 2 and Table 3 UTS/BP Reference Manual (90 17 64)
IMPROPER DEVICE TYPE	FILE or LABEL was specified but the device was incorrect. (SL, 3)
KEY CONFLICT	A keyword is repeated in the PROC reference list (SL, 3)

NOTE: SL = error severity level assignment to assembly if error encountered.

ERRORS (CONT'd.)

Message	Description
PACK SPECIFIED FOR FILE	PACK option was specified for a file DCB (SL, 3)
BIN SPECIFIED FOR FILE	BIN option was specified for a file DCB (SL, 3)
MAX RETENTION PERIOD > 999 DAYS	An improper specification was made for file retention period (SL, 2)
MAX. NO. OF DUMMY SNS EXCEEDED	SN, n specified a value of n>135 (SL, 3)

(Pages 122 and 123 deleted)

Xerox Corporation
701 South Aviation Boulevard
El Segundo, California 90245

XEROX

Reader Comment Form

We would appreciate your comments and suggestions for improving this publication.

Publication No.	Rev. Letter	Title	Current Date

How did you use this publication?

☐ Learning ☐ Installing ☐ Sales
☐ Reference ☐ Maintaining ☐ Operating

Is the material presented effectively?

☐ Fully Covered ☐ Well Illustrated ☐ Well Organized ☐ Clear

What is your overall rating of this publication?

☐ Very Good ☐ Fair ☐ Very Poor
☐ Good ☐ Poor

What is your occupation?

Your other comments may be entered here. Please be specific and give page, column, and line number references where applicable. To report errors, Please use the Xerox Software Improvement or Difficulty Report (1188) instead of this form.

Your Name & Return Address

2190(12/72)

Thank You For Your Interest. (fold & fasten as shown on back, no postage needed if mailed in U.S.A.)

Staple

Staple

Fold

First Class
Permit No. 229
El Segundo,
California

BUSINESS REPLY MAIL

No postage stamp necessary if mailed in the United States

Postage will be paid by

Xerox Corporation
701 South Aviation Boulevard
El Segundo, California 90245

Attn: Programming Publications

Fold

XEROX

701 South Aviation Boulevard
El Segundo, California 90245
213 679-4511