DOCUMENT RESUME

ED 049 613	24	EM 008 860

AUTHOR	Fitzhugh, Robert J.; Chadwick, Martin M.
TITLE	IMP: The LRDC Integrated Macro Package.
INSTITUTION	Pittsburgh Univ., Pa. Learning Research and Development Center.
SPONS AGENCY	Office of Education (DHEW), Washington, D.C. Bureau of Research.
BUREAU NO	BR-5-0253
PUB DATE	Dec 70
CONTRACT	OEC-4-10-158(010)
NOTE	112p.
EDRS PRICE	EDRS Price MF-$0.65 HC-$6.58
DESCRIPTORS	Computer Assisted Instruction, *Computer Based Laboratories, *Computer Programs, *Guides, Input Output, *Programing, *Programing Languages, Time Sharing
IDENTIFIERS	IMP, *Integrated Macro Package

ABSTRACT

The Learning Research and Development Center Time-Sharing System (LRDC/TSS) supports numerous non-standard devices and terminals and provides a variety of powerful programing options, enabling the researcher to maintain close control over the experimental environment. To achieve this degree of flexibility, it was necessary to write programs exclusively in assembly language, which made program development time consuming and produced programs that were difficult to "debug." The integrated Macro Package (IMP) was developed to provide a programing aid which does not become involved in the problems of compiler writing. It provides a programing structure and a body of debugged and documented routines to programers who write for the LRDC/TSS. Although it is used largely for computer-assisted instruction and on-line laboratory application, most of the routines are general purpose. IMP has a conditional assembly feature, which permits the programer to identify sections of the code that should be assembled only if a specified condition is met. This guide defines the conventions governing memory allocation, subroutine cells, input/output, and some miscellaneous function routines. This IMP-type solution would seem to be appropriate for laboratory installations with smaller computers and applications for which there are no suitable, higher level languages available. (Author/JY)

1

UNIVERSITY OF PITTSBURGH - LEARNING R&D CENTER

1971/2

IMP THE LRDC INTEGRATED MACRO PACKAGE

ROBERT J. FITZHUGH AND MARTIN M. CHADWICK

IMP

THE LRDC INTEGRATED MACRO PACKAGE

Robert J. Fitzhugh and Martin M. Chadwick

Learning Research and Development Center

University of Pittsburgh

December 1970

The research reported herein was performed pursuant to Contract N00014-67-A-0402-0006 (NR 154-262) with the Personnel and Training Branch Psychological Sciences Division, Office of Naval Research. The document is a publication of the Learning Research and Development Center, supported in part as a research and development center by funds from the United States Office of Education, Department of Health, Education and Welfare.

PREFACE

This document describes the LRDC Integrated Macro Package (IMP). IMP operates with the Digital Equipment Corporation's MACRO-9 Assembler, and knowledge of that system is assumed.

The authors wish to thank the members of the Applications Programming Group, particularly Mr. William Schmiedlin and Mr. Raymond McKnight, for their contributions to the development of IMP.

TABLE OF CONTENTS

1.0

INTRODUCTION

1.0 INTRODUCTION

The Computer Facility of the Learning Research and Development Center of the University of Pittsburgh supports an on-line behavioral science laboratory as well as computer-assisted instruction (CAI) research and development. The computer system is built around a 32K Digital Equipment Corporation PDP-7, a single-address, 18-bit word machine with a 1.75 microseconds cycle time. Sixteen of the 32K in the PDP-7 are obtained from a PDP-9 whose memory the PDP-7 is able to access through a special interface developed at LRDC. A time-sharing system called the LRDC Time-Sharing System (LRDC/TSS) has been developed for this system and has been in operation for several years.

The LRDC/TSS supports numerous non-standard devices and terminals and provides a variety of powerful programming options, enabling the researcher to maintain close control over the experimental environment. The cost of this flexibility, however, is programming complexity. Available higher level languages such as FORTRAN were found to be unsuitable, forcing programmers to write exclusively in assembly language. Program development time was typically long and programs were difficult to debug. The Integrated Macro Package (IMP) was developed as a partial solution to these problems.

Although some of the features of IMP were designed to meet requirements specific to the LRDC configuration, the package has a number of characteristics which should be of interest to those facing similar problems. Given a good macro assembler of the type available on many computers, IMP illustrates how it is possible to provide a significant programming aid without becoming involved in the problems of compiler writing. A package of this sort can be developed and refined over time by an assembly language programmer with no prior experience in programming language development. It can be as extensive or as limited as is desired and can be tailored to meet specific application or configuration requirements. This

IMP solution would seem to be appropriate for many laboratory installations with smaller computers and applications for which there are no suitable higher level languages available.

IMP is the simplified, linear descendent of an earlier effort called SKOOLBOL (Nemitz). SKOOLBOL incorporated a number of clever design notions and was an early attempt at a computer-assisted instruction and psychological experimentation language. The language had a "COBOL-like" appearance and was designed to provide programmers with a natural, English-like command set.

A full compiler was never developed for SKOOLBOL, although one was discussed and partially designed. The actual compilation procedure used involved a two-step process of pre-assembly and assembly. The pre-assembler was a separate program which did little more than restructure the SKOOLBOL source lines into a form acceptable to the manufacturer's MACRO-9 Assembler. The pre-assembler punched the restructured SKOOLBOL program on cards which were then assembled much like a normal assembly language program.

Although these procedures were cumbersome and time-consuming, SKOOLBOL failed to gain wide acceptance for other, more fundamental, reasons. Viewed historically, the desire to develop an "English-like" language was probably premature and perhaps unnecessary. Considerable programming power and flexibility was lost in return for a concise, English-like structure, and programmers faced with complex problems were often forced to return to assembly language. In addition, SKOOLBOL programs required a core-resident and non-relocatable subroutine package which was of fixed size and which included all routines, whether or not they were referenced. Because of these difficulties, the decision was made to abandon SKOOLBOL in favor of the Integrated Macro Package, an extensive package of macro functions which can be called by programmers writing in assembly language. The version of IMP described in this document has been in active use for over one year and has proved to be a practical and satisfactory compromise between a higher level language and pure assembly language programming.

2.0

GENERAL DESCRIPTION

2.0 GENERAL DESCRIPTION

The Integrated Macro Package (IMP) provides a programming structure and a body of commonly-used routines to application programmers who write in PDP-9 assembly language for the LRDC Time-Sharing System. IMP is a programming aid rather than a "language," although a number of higher-level services are provided. Nonetheless, programmers do not write "in" IMP but rather use it as a programming tool. A structure or framework is provided since IMP handles subroutine calls, memory allocation, variable storage, and input/output requests. Because the programmer working with IMP can call upon a set of well-debugged and documented routines, both programming and debugging time is reduced. Most of these routines are general-purpose, and to that extent, IMP should not be considered a specialized package intended only for computer-assisted instruction (CAI) or on-line laboratory applications. Its primary purpose is to reduce programming overhead without sacrificing the power and flexibility of assembly language-programming.

The full Integrated Macro Package occupies approximately 1.5K, or less than 20% of the memory available to each applications program operating under the LRDC Time-Sharing System. Since most applications programs use less than 50% of the available IMP routines, only those routines required are loaded with each applications program. This selective loading is accomplished through the use of the conditional assembly option of the PDP-9 assembler. This option permits the programmer to identify sections of code that should only be assembled if a specified condition is met. One of these conditions is whether or not a symbolic name has been encountered and defined earlier in the assembly process. This is specified by preceding the conditional portion of the program with the statement ".IFDEF" and following it with the statement ".ENDC." If the variable name is defined when the assembler encounters the ".IFDEF" statement, the program up to the ".ENDC" statement is assembled in a normal fashion. If the name is undefined, the conditional portions appear on the listing but are not assembled.

All IMP routines are conditionally defined in the IMP source code with the condition being the prior definition of the routine name. Since the entire IMP source deck is placed behind each program to be assembled, only those IMP routines referenced are assembled with the program. Figures 2.1 and 2.2 illustrate the conditional assembly of the IMP routine MOVE. The original and unassembled source code for the routine is shown in Figure 2.1. The conditional that must be satisfied to generate an assembly of the routine is the statement ".IFDEF MOVE." If an applications programmer referenced the routine, as in the following example:

```
- - -
- - -
CALL MOVE    /REFERENCE TO IMP ROUTINE
- - -
- - -
```

the condition would be satisfied and the routine would be assembled with the program, as in Figure 2.2.

IMP itself uses this conditional assembly feature. Many IMP routines reference other IMP routines as well as a number of secondary routines internal to the package. To fully eliminate unused portions, all routines are conditionally coded and are assembled only when required. This is illustrated in Figures 2.3 through 2.6 with the routines "TYPE" and "SK11NT". When the applications programmer uses the routine "TYPE", the conditional statement ".IFDEF TYPE" is satisfied, and the unassembled source code in Figure 2.3 is assembled, as shown in Figure 2.5. When "TYPE" is assembled, the symbol "SK11NT" is defined, and that routine, shown in Figure 2.4, is conditionally assembled and included in the program, shown in Figure 2.6. This nesting of conditionals within IMP will, in many instances, extend as far as four routines deep.

```
        .IF DEF MOVE
/MOVE                A ROUTINE TO MOVE ONE OR MORE CELLS FROM
/SK48           ONE SPECIFIED STORAGE AREA TO ANOTHER.  THE DATA
/               TO BE MOVED MUST BE TERMINATED WITH AN ASTERISK IN
/               THE LOW ORDER HALF OF A WORD IN EITHER PACKED
/               (2 PER WORD) OR UNPACKED ASCII
/                    CALLING SEQUENCE;
/                         ON INPUT SOURCE = ADDRESS OF SENDING FIELD
/                                  RECEVE = ADDRESS OF RECEIVING FIELD
/                              CALL MOVE
/                              RETURNS HERE
/
MOVE    LAM                   /LOAD A MINUS ONE
        TAD     SOURCE        /ADD ADDRESS OF SENDING FIELD
        DAC     10            /STORE IN AI-10
        LAM                   /LOAD A MINUS ONE
        TAD     RECEVE        /ADD ADDRESS OF RECEIVING FIELD
        DAC     11            /STORE IN AI-11
SK48A   LAC*    10            /LOAD VALUE TO BE MOVED
        DAC*    11            /STORE IN RECEIVING FIELD
        AND     (777)         /MASK OFF HIGH ORDER BITS
        SAD     (.AST)        /TERMINATING AST?
        JMP     POPJ          /YES, RETURN TO CALLER
        JMP     SK48A         /NO, GO BACK TO MOVE MORE
        .ENDC
        .EJECT
```

Figure 2.1

```
                                              .IF DEF MOVE
                                       /MOVE               A ROUTINE TO MOVE ONE OR MORE CELLS FROM
                                       /SK48          ONE SPECIFIED STORAGE AREA TO ANOTHER.  THE DATA
                                       /              TO BE MOVED MUST BE TERMINATED WITH AN ASTERISK IN
                                       /              THE LOW ORDER HALF OF A WORD IN EITHER PACKED
                                       /              (2 PER WORD) OR UNPACKED ASCII
                                       /                   CALLING SEQUENCE:
                                       /                        ON INPUT SOURCE = ADDRESS OF SENDING FIELD
                                       /                                 RECEVE = ADDRESS OF RECEIVING FIELD
                                       /                             CALL MOVE
                                       /                             RETURNS HERE
                                       /
51355         777777                   MOVE   LAM               /LOAD A MINUS ONE
51356         340416                          TAD     SOURCE    /ADD ADDRESS OF SENDING FIELD
51357         040010                          DAC     10        /STORE IN AI-10
51360         777777                          LAM               /LOAD A MINUS ONE
51361         340417                          TAD     RECEVE    /ADD ADDRESS OF RECEIVING FIELD
51362         040011                          DAC     11        /STORE IN AI-11
51363         220010                   SK48A  LAC*    10        /LOAD VALUE TO BE MOVED
51364         060011                          DAC*    11        /STORE IN RECEIVING FIELD
51365         511573                          AND     (777)     /MASK OFF HIGH ORDER BITS
51366         551536                          SAD     (.AST)    /TERMINATING AST?
51367         611421                          JMP     POPJ      /YES, RETURN TO CALLER
51370         611363                          JMP     SK48A     /NO, GO BACK TO MOVE MORE
                                              .ENDC
                                              .EJECT
```

Figure 2.2

```
           .IF DEF TYPE
/TYPE              A ROUTINE TO HANDLE THE PRINTING OF MESSAGES ON THE
/SK11          TELETYPES.  THE MESSAGE MUST CONTAIN THE NUMBER OF CHARACTERS
/              FOR PRINTING IN THE FIRST BUFFER WORD.  PARAMETER LIST
/              DEFAULTS TO A SUSPENDED PRINT
/                  CALLING SEQUENCE:
/                      SOURCE=BUFFER ADDRESS-15 BITS
/                      CALL   TYPE
/                      TTY1 TTY2 OR TTY3-TELETYPE NUMBER EQUIVALENCE
/                      RETURNS HERE
/
TYPE       ISZ*    STKPTR     /INCREMENT RETURN ADDRESS
           LAC*    PUSHJ3     /LOAD TELETYPE NO.
           JMP     SK11NT
           .ENDC
```

Figure 2.3

```
/
/
         .IF DEF SK11NT
SK11NT   TAD     (-1)         /SUBTRACT ONE
         DAC     SKC27        /STORE OFFSET IN SKC27
         TAD     SKC25        /ADD MEMAL BASE ADDRESS
         TAD     (30)         /ADD OFFSET TO FIRST PRINT PARAMETER WORD
         DAC     SKC28        /STORE POINTER IN SKC28
         LAC     (33)         /LOAD OFFSET TO BAGTEL PARAMETER LIST
         TAD     SKC25        /ADD MEMAL BASE ADDRESS
         DAC     SKC29        /STORE IN SKC29
         DAC     SKC30        /STORE IN SKC30
         LAC*    SKC28        /LOAD PARAMETER WORD 0
         DAC*    SKC29        /STORE IN PARAMETER LIST
         ISZ     SKC29        /INCREMENT POINTER
         DZM*    SKC29        /ZERO PARAMETER WORD 1
         ISZ     SKC29        /INCREMENT POINTER
         LAC     SOURCE       /LOAD MESSAGE ADDRESS
         DAC*    SKC29        /STORE IN PARAMETER WORD 2
         LAC     SKC30        /RELOAD PARAMETER LIST BASE ADDRESS
         TAD     (500000)     /ADD PRINT COMMAND
         EEM                  /ENABLE EXTEND MODE
         DPI                  /DISABLE API
         JMS*    TRNVEC       /TRANSFER TO SYSTEM ROUTINE - BAGTEL/D-PHONE
         SZL                  /REQUEST ERROR
         JMP     SK11A        /YES-GO PROCESS
         JMP     POPJ         /NO-RETURN TO CALLER
SK11A    LAW     146          /LOAD BASIC ERROR CODE
         TAD     SKC27        /ADD TELETYPE NUMBER OFFSET
         JMP     ERROR        /GO TO ERROR ROUTINE
         .ENDC
         .EJECT
```

Figure 2.4

```
                                        .IF DEF TYPE
                                /TYPE          A ROUTINE TO HANDLE THE PRINTING OF MESSAGES ON THE
                                /SK11        TELETYPES.  THE MESSAGE MUST CONTAIN THE NUMBER OF CHARACTER
                                /            FOR PRINTING IN THE FIRST BUFFER WORD.  PARAMETER LIST
                                /            DEFAULTS TO A SUSPENDED PRINT
                                /               CALLING SEQUENCE:
                                /                   SOURCE=BUFFER ADDRESS-15 BITS
                                /                   CALL   TYPE
                                /                   TTY1 TTY2 OR TTY3-TELETYPE NUMBER EQUIVALENCE
                                /                   RETURNS HERE
                                /
50761         460401            TYPE    ISZ*    STKPTR      /INCREMENT RETURN ADDRESS
50762         220463                    LAC*    PUSHJ3      /LOAD TELETYPE NO.
50763         611023                    JMP     SK11NT
                                        .ENDC
                                /
                                /
```

Figure 2.5

```
                                      /
                                      /
                                              .IF DEF SK11NT
51023          351550             SK11NT  TAD     (-1)        /SUBTRACT ONE
51024          040432                     DAC     SKC27       /STORE OFFSET IN SKC27
51025          340430                     TAD     SKC25       /ADD MEMAL BASE ADDRESS
51026          351535                     TAD     (30)        /ADD OFFSET TO FIRST PRINT PARAMETER WORD
51027          040433                     DAC     SKC28       /STORE POINTER IN SKC28
51030          211715                     LAC     (33)        /LOAD OFFSET TO BAGTEL PARAMETER LIST
51031          340430                     TAD     SKC25       /ADD MEMAL BASE ADDRESS
51032          040434                     DAC     SKC29       /STORE IN SKC29
51033          040435                     DAC     SKC30       /STORE IN SKC30
51034          220433                     LAC*    SKC28       /LOAD PARAMETER WORD 0
51035          060434                     DAC*    SKC29       /STORE IN PARAMETER LIST
51036          440434                     ISZ     SKC29       /INCREMENT POINTER
51037          160434                     DZM*    SKC29       /ZERO PARAMETER WORD 1
51040          440434                     ISZ     SKC29       /INCREMENT POINTER
51041          200416                     LAC     SOURCE      /LOAD MESSAGE ADDRESS
51042          060434                     DAC*    SKC29       /STORE IN PARAMETER WORD 2
51043          200435                     LAC     SKC30       /RELOAD PARAMETER LIST BASE ADDRESS
51044          351652                     TAD     (500000)    /ADD PRINT COMMAND
51045          707702                     EEM                 /ENABLE EXTEND MODE
51046          700001                     DPI                 /DISABLE API
51047          120443                     JMS*    TRNVEC      /TRANSFER TO SYSTEM ROUTINE - BAGTEL/D-PHONE
51050          741400                     SZL                 /REQUEST ERROR
51051          611053                     JMP     SK11A       /YES-GO PROCESS
51052          611421                     JMP     POPJ        /NO-RETURN TO CALLER
51053          760146             SK11A   LAW     146         /LOAD BASIC ERROR CODE
51054          340432                     TAD     SKC27       /ADD TELETYPE NUMBER OFFSET
51055          610754                     JMP     ERROR       /GO TO ERROR ROUTINE
                                          .ENDC
                                          .EJECT
```

Figure 2.6

3.0

USE AND OPERATION OF IMP

3.0 USE AND OPERATION OF IMP

Assembly language programs using IMP must obey a set of programming conventions governing memory allocation, subroutine calls, communication with IMP, the naming of routines and variables, and error handling. These conventions define the interface between the user program and IMP as well as ensure that user programs are fully reentrant. Since the LRDC Time-Sharing System is not a swapping system, memory is used most efficiently if user programs are reentrant, and many of the features of IMP were developed to meet this requirement.

In a reentrant program, pure executable code is shared and executed by more than one user. However, each user's program variables, subroutine returns, and user-specific data must be isolated and protected from that of other users. Two memory allocation schemes are provided by the LRDC/TSS for this purpose. In the first, a block of memory called COMMON is shuffled to and from lower core by the time-sharing system. Each user has his own copy of COMMON, and this is restored to its proper place by the system immediately prior to the user's execution. The COMMON block begins at location 400_8 of each 8K field and must not exceed 1000 words in length. Since this area is protected by the system, program variables stored in COMMON may be directly addressed by reentrant programs, simplifying programming.

The second scheme is called MEMAL, or MEMory ALlocation. A portion of each field of memory is reserved as a pool of available space which may be allocated on demand to any user executing within that field. A user who requires space calls the system routine MEMAL, specifying the number of contiguous locations desired. If space is available, MEMAL returns to the user with the address of the allocated memory block. Since each user executing the reentrant program makes a separate call upon MEMAL, a different memory block is allocated to each user, which ensures that information stored in MEMAL-obtained space is protected from inter-user interference. Because the address of the MEMAL block is not known when a program is written, the address must be stored as a protected variable in COMMON. To access a particular location in the MEMAL space, the reentrant program must add a displacement value to the MEMAL base address in order to compute the actual address.

Both COMMON and MEMAL are supported and used by IMP. To obtain COMMON space, the user calls the general initialization routine, LOGON (see 4.2.1 LOGON for details). Of the total space requested, locations 400_8 through 470_8 always are reserved for use by IMP for the storage of subroutine returns and special IMP variables.

Subroutine returns are stored in COMMON in a push-down stack maintained by the IMP routines PUSHJ and POPJ. Programs using IMP do not use the normal JMS instruction, but call PUSHJ, specifying the address of the subroutine:

```
---
---
JMS PUSHJ
SUBR
---
---
```

PUSHJ computes the subroutine return address, adds it to the top of the push-down stack, and transfers control to the user subroutine. To return to the caller, the subroutine transfers control to the IMP routine POPJ:

```
----
----
----
JMP POPJ     /EXIT FROM SUBROUTINE
```

POPJ removes the top-most entry from the subroutine stack, stores it as the return address, "pops" the stack upward, and transfers control to the return address. Subroutines may be nested as deeply as there are available entries in the push-down stack. At LRDC, a maximum stack size of 10 has proved to be adequate for even the most complex CAI program.

To simplify programming, IMP provides two macros which expand into calls upon PUSHJ and POPJ. To call a subroutine, the user writes:

```
         ---
         ---
         CALL SUBR     /CALL TO SUBROUTINE
         ---           /POINT OF RETURN
         ---
```

This is expanded by the assembler into:

```
         ----
         ----
         CALL SUBR     /CALL TO SUBROUTINE
   GEN*  JMS PUSHJ
   GEN*  SUBR
         ----          /POINT OF RETURN
         ----
```

To return from a subroutine, the user writes RETURN:

```
         ----
         ----
         RETURN       /EXIT FROM SUBROUTINE
```

This is expanded into:

```
         ----
         ----
         RETURN      /EXIT FROM SUBROUTINE
  GEN*   JMP    POPJ
```

These macros are provided for convenience only, and direct calls to PUSHJ and POPJ are permissable.

The PUSHJ/POPJ routines provide two primary benefits. The writing of reentrant programs is facilitated since subroutine returns are automatically stored and protected in COMMON. Secondly, the push-down stack provides a partial trace of program flow which is useful when debugging. Through an examination of the contents of the push-down stack, programmers are often able to quickly locate bugs or points of difficulty.

Two other stack manipulation routines are available in the IMP package. The routine PARAM allows a subroutine to retrieve a parameter located immediately after the call, and also, to properly adjust the return address in the push-down stack. STEPUP merely advances the return address a specified number of locations. Full details on these routines can be found in sections 4.1.2 PARAM and 4.1.3 STEPUP.

In addition to subroutine returns, a number of key IMP variables are located in COMMON where they may be directly addressed. These variables are the primary means of communication between a user program and IMP and have been assigned unique names which identify their function or purpose. For example, the IMP routine DIVIDE expects to find the dividend and divisor in the variable locations called SOURCE and RECEVE. DIVIDE returns to the caller with the quotient and remainder in the variable locations ANSWER and REMAIN.

Users are discouraged from using these IMP variables for any purpose other than to communicate with IMP. IMP makes internal use of many of these variables as temporary word space to reduce its COMMON requirement. The current version of IMP reserves only 10% of the available COMMON space so that users should not be forced to use these IMP variables because of space limitations.

The IMP variables and their major functions are:

ADRESS -- returns addresses calculated by IMP

ANSWER -- returns subject responses or the most significant digits of a numeric answer

CONTEN -- returns the contents of a specified MEMAL location

RECEVE -- passes to IMP the second value to be used in a calculation or the second parameter for an I/O routine

REMAIN -- returns the least significant digits of a calculation

SLIDE -- passes a projector slide number to the projector routine LOCATE

SOURCE -- used as an input parameter to most IMP routines for a variety of purposes

TIME -- returns time-of-day or response latency

In addition to COMMON, IMP supports MEMAL and provides routines to acquire, store data in, and retrieve data from MEMAL space. The IMP routine DEFINE acquires and names blocks of space up to the total amount available in the MEMAL pool. (See 4.2.2 DEFINE for full details.) The routine FIND locates a named location in MEMAL space and returns its address and contents to the caller. STORE allows the user to deposit a value in a named MEMAL location. These routines greatly simplify programming by computing the actual address from a MEMAL block base address and a displacement value. (See 4.2.3 FIND and 4.2.4 STORE for full details.)

Program error detection and handling is another feature of the IMP package. With the one exception of the routine DISK, all errors detected by IMP are passed to the IMP error handler, ERROR. ERROR calls the system error routine SYSERR with a unique error code which is printed on the operator's console. The user's program is then placed in an inactive state pending operator intervention. Errors detected by the routine DISK often indicate "normal" conditions such as end-of-file and are returned to the user program rather than passed to ERROR. Error codes returned by the DISK routine are described in section 4.3.24 DISK. All IMP error codes are listed in Appendix C.

4.0

IMP ROUTINES

4.1

SUBROUTINE CALL ROUTINES

4.0 IMP ROUTINES

4.1 SUBROUTINE CALL ROUTINES

4.1.1 PUSHJ/POPJ

4.1.1.1 FUNCTION

PUSHJ and POPJ are used to call subroutines and return from them. Instead of directly calling a subroutine, the user passes the subroutine address to PUSHJ. PUSHJ determines the origin of the call, stores the return address in a push-down stack, and passes control to the subroutine. When the user desires to return from the subroutine, POPJ is called, which 'pops-up' the push-down stack and passes control to the address at the top of the stack. The push-down stack is currently 10 deep so subroutines may be nested to that depth.

Within IMP, two macros using PUSHJ and POPJ have been defined to simplify programming. The macro 'CALL' has the subroutine address as a single argument as follows:

```
CALL SUBROUTINE
```

This macro expands in the following form:

```
JMS PUSHJ
SUBROUTINE
```

The macro 'RETURN' has no arguments and expands into:

```
JMP POPJ
```

4.1.1.2 CALLING SEQUENCE

4.1.1.2.1 PARAMETERS

The subroutine address is the only parameter required.

4.1.1.2.2 CALL

CALL ROUTINE

4.1.1.2.3 NORMAL RETURN

Control is returned to the location following the call.

4.1.1.2.4 ERROR RETURN

Type 1 SYSERR codes:

163 -- Stack overflow - PUSHJ

164 -- Stack underflow - POPJ

4.1.1.2.5 RESTRICTIONS

a. The PUSHJ 'STACK' must be initialized with the 'RESET' macro.

b. The user may not nest more than 10 routines deep.

c. Every 'CALLed' routine must be exited by a 'RETURN' statement.

d. User macro definition must be present in the user's card deck before 'RESET', 'CALL' or 'RETURN' macros are encountered.

4.1.1.3 EXAMPLE

To set up the PUSHJ STACK and call a routine, 'SUBRTN'

```
RESET    /MACRO TO SETUP STACK
         /ASSEMBLER GENERATED CODE
         /ASSEMBLER GENERATED CODE
         /ASSEMBLER GENERATED CODE
```

```
      CALL SUBRTN   /CALL USER SUBRTN
*GEN* JMS PUSHJ     /ASSEMBLER GENERATED CODE
*GEN* SUBRTN        /ASSEMBLER GENERATED CODE
                    /RETURN IS MADE HERE PROVIDED
                    /THAT SUBRTN IS EXITED BY
                    /'JMP POPJ'
```

4.1.2 PARAM

4.1.2.1 FUNCTION

PARAM enables the user to retrieve one in-line parameter from the point of a subroutine call.

4.1.2.2 CALLING SEQUENCE

4.1.2.2.1 PARAMETERS

None

4.1.2.2.2 CALL

Call PARAM

4.1.2.2.3 NORMAL RETURN

PARAM returns to the next location after the call with the parameter: CONTEN = In-line parameter from calling routine.

4.1.2.2.4 ERROR RETURN

None

4.1.2.3 EXAMPLE

To pick up an in-line parameter needed by a subroutine called SUBRTN:

```
               .   /
      CALL SUBRTN  /CALL 'SUBRTN'
               3   /IN-LINE PARAMETER
               .   /
               .   /
SUBRTN CALL PARAM  /'SUBRTN' CALLS PARAM TO GET
                   /IN-LINE PARAMETER
       LAC CONTEN  /CONTEN CONTAINS IN-LINE PARAMETER
               .   /
```

4.1.3 STEPUP

4.1.3.1 FUNCTION

STEPUP enables the user to modify or 'STEPUP' a subroutine return address up to five locations.

4.1.3.2 CALLING SEQUENCE

4.1.3.2.1 PARAMETERS

AC = Number of locations to STEPUP return

4.1.3.2.2 CALL

Call STEPUP

4.1.3.2.3 NORMAL RETURN

STEPUP returns to the location following the call.

4.1.3.2.4 ERROR RETURN

Type 1 SYSERR code:

172 -- STEPUP value outside
range 1 - 5

4.1.3.3 EXAMPLE

To return three locations beyond the normal return for a subroutine, code:

```
1. LAC (3)
2. CALL STEPUP
3. RETURN
```

4.2

MEMORY ALLOCATION ROUTINES

4.2 MEMORY ALLOCATION ROUTINES

4.2.1 LOGON

4.2.1.1 FUNCTION

LOGON is responsible for program initialization and does the following:

a. Acquires COMMON space.

b. Sets a transfer vector for Teletype or Dataphone.

c. Zeros MEMAL base pointers.

d. Initializes PUSHJ-POPJ.

e. Initializes key variables for DISK and DO routines.

4.2.1.2 CALLING SEQUENCE

4.2.1.2.1 PARAMETERS

A cell labeled COMMON must contain the number of user COMMON cells desired. (The user's COMMON should start at FIELD + 470).

4.2.1.2.2 CALL

LOGON or LOGONT (for Teletype)
LOGOND (for Dataphone)

4.2.1.2.3 NORMAL RETURN

LOGON returns to the location immediately following COMMON cell (COMMON + 1).

4.2.1.2.4 ERROR RETURNS

None

4.2.1.3 EXAMPLE

To LOGON using Dataphone and obtaining six COMMON locations:

```
          LOGOND
COMMON ENDCOM - BEGCOM + 1    /6 CELLS COMMON
          .LOC FLD + 470
```

```
BEGCOM 0                     /FIRST COMMON CELL
PTR1   0
CTR1   0
TEMP1  0
SAVE1  0
ENDCOM 0                     /LAST COMMON CELL
          .LOC COMMON + 1
          .                  /RETURNS
          .
          .
```

4.2.2 DEFINE

4.2.2.1 FUNCTION

DEFINE enables the user to obtain MEMAL space.

4.2.2.2 CALLING SEQUENCE

4.2.2.2.1 PARAMETERS

One or more in-line parameters must follow the call on DEFINE. These parameters terminate with a FINISH statement. Each in-line parameter consists of a name tag (col. 1) and the block length in octal (col. 8). In-line parameters specifying block names and their lengths plus the FINISH parameter are required. Any reasonable number of blocks may be requested, provided the total number of available MEMAL cells are not exceeded.

4.2.2.2.2 CALL

```
                    CALL DEFINE
NAME1   N    (N = Length of block in
              octal)
NAME2   N
NAME3   N
  .
  .
NAMEX   N
      FINISH
```

4.2.2.2.3 NORMAL RETURN

DEFINE returns to the location following the FINISH statement. The "N" values indicated in the parameters above are changed by DEFINE to offset values, which give the actual address when added to the MEMAL base address. The FIND routine calculates addresses of named MEMAL buffers.

4.2.2.2.4 ERROR RETURN

Type 0 SYSERR codes:

101 -- MEMAL request size zero

177 -- MEMAL space exhausted.

4.2.2.3 EXAMPLE

To obtain a 300_8 cell MEMAL area consisting of three named buffers of 100_8 cells each:

```
         CALL DEFINE   /CALL DEFINE ROUTINE
  BUFF1  100           /BUFFER 1, 100 CELLS
  BUFF2  100           /BUFFER 2, 100 CELLS
  BUFF3  100           /BUFFER 3, 100 CELLS
         FINISH        /END OF PARAMETERS
                       /RETURNS HERE
```

4.2.3 FIND

4.2.3.1 FUNCTION

FIND locates a named cell in MEMAL and returns its address and content to the caller.

4.2.3.2 CALLING SEQUENCE

4.2.3.2.1 PARAMETERS

The name of the desired cell is passed as an in-line parameter.

4.2.3.2.2 CALL

```
CALL FIND
NAME    /NAME OF A CELL
          DEFINED IN MEMAL
```

4.2.3.2.3 NORMAL RETURN

FIND returns to the location following the call with these parameters:

```
ADRESS = Address of the named cell
         in MEMAL
CONTEN = Contents of the named cell
         in MEMAL
```

4.2.3.2.4 ERROR RETURN

None

4.2.3.3 EXAMPLE

To obtain the contents of a MEMAL cell named VARBL:

```
CALL FIND      /CALL FIND ROUTINE
VARBL          /MEMAL CELL NAME
LAC CONTEN     /CONTINUE
```

4.2.4 STORE

4.2.4.1 FUNCTION

STORE enables the user to deposit a value in a named MEMAL cell. STORE is used only on MEMAL obtained through the DEFINE routine.

4.2.4.2 CALLING SEQUENCE

4.2.4.2.1 PARAMETERS

SOURCE = Value to be stored

In-line parameter following the call naming the cell within MEMAL.

4.2.4.2.2 CALL

```
CALL STORE
NAME
```

4.2.4.2.3 NORMAL RETURN

STORE returns to the location following the MEMAL name parameter.

4.2.4.2.4 ERROR RETURN

Not applicable.

4.2.4.3 EXAMPLE

To store an asterisk (ASCII 252) in a MEMAL cell called 'ENDBUF':

```
LAC (.AST)     /LOAD AN ASTERISK
DAC SOURCE     /STORE IN SOURCE
CALL STORE     /GO TO STORE ROUTINE
ENDBUF         /DEFINED CELL NAME
               /RETURNS HERE
```

4.0

INPUT/OUTPUT ROUTINES

4.3 INPUT/OUTPUT ROUTINES

4.3.1 OBTAIN

4.3.1.1 FUNCTION

OBTAIN is used to GRAB I/O devices. No more than 12 device types with seven units per type can be GRABbed in a single call on OBTAIN.

4.3.1.2 CALLING SEQUENCE

4.3.1.2.1 PARAMETERS

Two 6-digit in-line parameters are used to indicate the device types and the number of units per type to be GRABbed. These parameter words are broken down as follows:

	Digit	Device
Word 1	1	Screen
	2	Keyboard
	3	Touch
	4	RA 950 Projector
	5	Teletype/Dataphone
	6	Crow
Word 2	1	Punch
	2	Line Printer
	3	To be assigned
	4	To be assigned
	5	To be assigned
	6	To be assigned

4.3.1.2.2 CALL

CALL OBTAIN (IMPORTANT: Both parameter words must be present, even if they are zero.)

Parameter 1

Parameter 2

4.3.1.2.3 NORMAL RETURN

OBTAIN returns to the location following the second parameter in the call.

4.3.1.2.4 ERROR RETURN

Type 1 SYSERR codes:

100 -- GRAB error - Screen
101 -- GRAB error - Keyboard
102 -- GRAB error - Touch
103 -- GRAB error - Projector
104 -- GRAB error - Teletype/Dataphone
105 -- GRAB error - Crow
106 -- GRAB error - Punch
107 -- GRAB error - Printer
110 -- To be assigned
111 -- To be assigned
112 -- To be assigned
113 -- To be assigned

4.3.1.3 EXAMPLE

To grab 1 touch-sensitive, 2 projectors, 1 teletype, and 1 crow:

```
CALL OBTAIN     /CALL OBTAIN ROUTINE FOR
001211          /1 TOUCH, 2 PROJECTORS
                 1 TELETYPE
000000          /AND 1 CROW
                /RETURNS HERE
```

4.3.2 RELESE

4.3.2.1 FUNCTION

RELESE returns to the Executive systems those I/O devices which have been "GRABbed" by the job (see OBTAIN for device GRABs). Up to 12 device types and seven units of each type may be released.

4.3.2.2 CALLING SEQUENCE

4.3.2.2.1 PARAMETERS

As in the OBTAIN routine, two 6-digit octal in-line parameter words indicate which devices are to be released. The parameter word structure is as follows:

	Digit	Associated Device
Word 1	1	-CRT Screen
	2	Keyboard
	3	Touch Sensitive
	4	Projector
	5	Teletype/Dataphone
	6	Crow
Word 2	1	Paper Tape Punch
	2	Line Printer
	3	Presently unassigned
	4	Presently unassigned
	5	Presently unassigned
	6	Presently unassigned

4.3.2.2.2 CALL

```
CALL RELESE
WORD 1
WORD 2
```

(IMPORTANT: Both parameter words must be present, even if they are zero.)

4.3.2.2.3 NORMAL RETURN

RELESE returns to the location following parameter Word 2.

4.3.2.2.4 ERROR RETURN

Type 1 SYSTERR codes:

114 -- Release Error - Screen
115 -- Release Error - Keyboard
116 -- Release Error - Touch
117 -- Release Error - Projector
120 -- Release Error - TTY/DATAPHONE
121 -- Release Error - Crow
122 -- Release Error - Punch
123 -- Release Error - Printer
124 -- To be assigned
125 -- To be assigned
126 -- To be assigned
127 -- To be assigned

4.3.2.3 EXAMPLE

To release 1 Touch, 2 Projectors, 1 Teletype, 1 Crow, and 1 Paper Tape Punch:

```
    CALL RELESE    /GO TO RELEASE ROUTINE
    001211         /PARAMETER WORD 1
    100000         /PARAMETER WORD 2
                   /RETURNS HERE
```

4.3.3 SETUP

4.3.3.1 FUNCTION

SETUP creates parameter lists for all the devices that can be GRABbed by the OBTAIN routine. The parameter lists are set up in the most commonly used format. However, the user can alter the parameter lists if necessary (see EXCEPT routine documentation).

4.3.3.2 CALLING SEQUENCE

4.3.3.2.1 PARAMETERS

FINISH

NOTE: Detailed information concerning parameter lists can be found in the documentation of peripheral equipment routines.

4.3.3.2.2 CALL

```
CALL SETUP
FINISH
```

4.3.3.2.3 NORMAL RETURN

SETUP returns to the location following the parameter FINISH.

4.3.3.2.4 ERROR RETURN

Type 1 SYSERR code:

171 -- no "FINISH" parameter

4.3.3.3 EXAMPLE

To create standard parameter lists:

```
CALL SETUP     /ESTABLISH PARAMETER LISTS
FINISH         /WITH NO CHANGES
               /RETURNS HERE
```

4.3.4 EXCEPT

4.3.4.1 FUNCTION

EXCEPT modifies the parameter lists created by the SETUP routine.

4.3.4.2 CALLING SEQUENCE

4.3.4.2.1 PARAMETERS

Three parameters are required for each change.

Parameter 1 -- Device number *

Parameter 2 -- Parameter list word number to be replaced

Parameter 3 -- The parameter word to be inserted

*Device numbers are the same as those shown in the CENTRAL EXECUTIVE documentation.

NOTE: Detailed information concerning parameter lists can be found in the documentation of peripheral equipment routines.

4.3.4.2.2 CALL

The call on EXCEPT must appear between the call on SETUP and its parameter FINISH (see Example).

```
CALL SETUP
EXCEPT
A
B
C
FINISH
```

4.3.4.2.3 NORMAL RETURN

EXCEPT returns to the location following the word FINISH.

4.3.4.2.4 ERROR RETURN

Type 1 SYSERR code:

165 -- Unknown device number

171 -- No FINISH parameter

4.3.4.3 EXAMPLE

To set up parameter lists and alter the teletype list to enable the unit to be treated as a half-duplex unit:

```
CALL SETUP     /FIRST, CREATE PARAMETER LISTS
EXCEPT         /CHANGE FOLLOWS
23             /TELETYPE DEVICE NUMBER
1              /FIRST WORD IN PARAMETER LIST
200001         /NEW PARAMETER
FINISH         /ALL PARAMETER WORD DONE
               /CONTINUE
```

4.3.5 DISPLA

4.3.5.1 FUNCTION

DISPLA enables the user to project a text string on the CRT SCREEN.

4.3.5.2 CALLING SEQUENCE

4.3.5.2.1 PARAMETERS

The message address and the character size must be specified. The first word of the text must contain the length (number of characters in octal), followed by the message text in packed ASCII, two characters per word.

SOURCE = text address

An in-line parameter following the call must indicate the character size to be displayed. (see "SCREEN CHARACTER SIZE CHART, APPENDIX E, for details)

4.3.5.2.2 CALL

```
SOURCE = TEXT ADDRESS
CALL DISPLA
CHARACTER SIZE
```

4.3.5.2.3 NORMAL RETURN

DISPLA returns to the location following the character size parameter. NOTE: The first cell of the text string (length cell) is altered prior to return.

4.3.5.2.4 ERROR RETURN

Type 1 SYSERR code:

130 -- request error -
SCREEN

4.3.5.3 EXAMPLE

To display a message labeled MSG1 on the SCREEN using character size 4:

```
      LAC (MSG1)      /LOAD ADDRESS OF MSG1
      DAC SOURCE      /STORE IN SOURCE
      CALL DISPLA     /GO TO DISPLA ROUTINE
      4               /CHARACTER SIZE 4
                      /RETURNS HERE

MSG1  14              /MSG LENGTH (OCTAL)
      323303          /S - C
      322305          /R - E
      305316          /E - N
      240324          /  - T
      305323          /E - S
      324256          /T - .
```

4.3.6 SHOLET

4.3.6.1 FUNCTION

SHOLET displays a single character on the screen.

4.3.6.2 CALLING SEQUENCE

4.3.6.2.1 PARAMETERS

SOURCE = Character to be displayed.

Character size (see SCREEN CHARACTER SIZE CHART)

4.3.6.2.2 CALL

CALL SHOLET
CHARACTER SIZE

4.3.6.2.3 NORMAL RETURN

SHOLET returns to next location following the call.

4.3.6.2.4 ERROR RETURN

Type 1 SYSERR code:
130 -- Screen request error

4.3.6.3 EXAMPLE

To display the letter "B" on the screen:

```
LAC (302)     /LOAD THE ASCII CODE FOR "B"
DAC SOURCE    /PUT IT IN SOURCE
CALL SHOLET   /CALL SHOLET ROUTINE
4             /SCREEN SIZE 4
              /CONTINUE
              /
```

4.3.7 ERASE

4.3.7.1 FUNCTION

ERASE enables the user to erase the CRT SCREEN.

4.3.7.2 CALLING SEQUENCE

4.3.7.2.1 PARAMETER

None

4.3.7.2.2 CALL

CALL ERASE

4.3.7.2.3 NORMAL RETURN

ERASE returns to the location following the call.

4.3.7.2.4 ERROR RETURN

Type 1 SYSERR code:

130 -- request error - SCREEN

4.3.7.3 EXAMPLE

To erase a CRT SCREEN:

```
CALL ERASE      /GO TO ERASE SCREEN
                /RETURNS HERE
```

4.3.8 SKIP

4.3.8.1 FUNCTION

SKIP enables the user to skip down a specified number of lines on the CRT SCREEN.

4.3.8.2 CALLING SEQUENCE

4.3.8.2.1 PARAMETERS

VARBL = number of lines to skip (octal). An in-line parameter specifies the address of the user's VARBL.

4.3.8.2.2 CALL

```
CALL SKIP
VARBL
```

4.3.8.2.3 NORMAL RETURN

SKIP returns to the location following the in-line VARBL parameter.

4.3.8.2.4 ERROR RETURN

Type 1 SYSERR code:
130 -- request error -
SCREEN

4.3.8.3 EXAMPLE

To skip down six lines from the current position:

```
LAC (6)      /LOAD LINE COUNT (6)
DAC VARBL    /STORE IN USER VARIABLE
CALL SKIP    /GO TO SKIP ROUTINE
VARBL        /PTR TO REPETITION CELL
             /RETURNS HERE
```

4.3.9 BACKUP

4.3.9.1 FUNCTION

BACKUP returns the cursor to the left-most side of the screen.

4.3.9.2 CALLING SEQUENCE

4.3.9.2.1 PARAMETERS

None

4.3.9.2.2 CALL

CALL BACKUP

4.3.9.2.3 NORMAL RETURN

BACKUP returns to the location following the call.

4.3.9.2.4 ERROR RETURN

Type 1 SYSERR code:

130 -- Screen request error

4.3.9.3 EXAMPLE

To return the cursor to the left-most side of the screen:

```
   CALL BACKUP     /CALL BACKUP ROUTINE
                   /RETURNS HERE
```

4.3.10 SPACE

4.3.10.1 FUNCTION

SPACE gives the user the facility to display one or more spaces on the screen in a single call.

4.3.10.2 CALLING SEQUENCE

4.3.10.2.1 PARAMETERS

Desired number of spaces in some user-specified cell.

4.3.10.2.2 CALL

CALL SPACE

VARBL -- Address of cell containing number of spaces.

4.3.10.2.3 NORMAL RETURN

SPACE returns to the location following the call.

4.3.10.2.4 ERROR RETURN

Type 1 SYSERR code:

130 -- Screen request error

4.3.10.3 EXAMPLE

To display four spaces on the screen:

```
LAC (4)    /LOAD A 4
DAC VARBL  /PUT IT IN USER'S CELL
CALL SPACE /CALL SPACE ROUTINE
VARBL      /ADDRESS OF NUMBER OF SPACES
           /RETURNS HERE
```

4.3.11 REAKDY

4.3.11.1 FUNCTION

READKY monitors a keyboard for a subject's response. The monitoring time limit is defined by the user.

4.3.11.2 CALLING SEQUENCE

4.3.11.2.1 PARAMETERS

Monitoring time-limit in seconds in a user-specified cell.

4.3.11.2.2 CALL

CALL READKY

VARBL -- Address of time limit cell.

4.3.11.2.3 NORMAL RETURN

READKY returns to the location following the call. The following parameter is returned:

```
ANSWER = ZERO -- NO RESPONSE
       400000 -- PARITY ERROR
        OTHER -- ASCII RESPONSE
```

4.3.11.2.4 ERROR RETURN

Type 1 SYSERR code:

131 -- Keyboard request error

4.3.11.3 EXAMPLE

To monitor the keyboard for 5 seconds:

```
LAC (5)       /LOAD TIME LIMIT
DAC VARBL     /STORE IN TIME LIMIT CELL
CALL READKY   /CALL READKY ROUTINE
VARBL         /ADDRESS OF TIME LIMIT CELL
LAC ANSWER    /RETURNS HERE
```

4.3.12 TUCH

4.3.12.1 FUNCTION

TUCH monitors the touch-sensitive surface for a subject's response. The user defines the monitoring time-limit.

4.3.12.2 CALLING SEQUENCE

4.3.12.2.1 PARAMETERS

VARBL = number of seconds delay

4.3.12.2.2 CALL

```
CALL TUCH
VARBL
```

4.3.12.2.3 NORMAL RETURN

TUCH returns to the location after the call with following parameter:

```
ANSWER = ZERO -- NO RESPONSE
       400000 -- PARITY ERROR
        OTHER -- WINDOW NUMBER OF
```

TOUCH $(1\text{-}120_8)$

4.3.12.2.4 ERROR RETURN

Type 1 SYSERR code:

132 -- Executive System request error

4.3.12.3 EXAMPLE

To monitor the touch-sensitive surface for five seconds:

```
LAC (5)       /LOAD 5 SECONDS
DAC DELAY     /STORE IN DELAY CELL
CALL TUCH     /CALL TOUCH ROUTINE
DELAY         /ADDRESS OF TIME DELAY
LAC ANSWER    /LOAD RESULTS AND CONTINUE
```

4.3.13 LOCATE

4.3.13.1 FUNCTION

LOCATE is used to position a slide on a RA-950 projector.

4.3.13.2 CALLING SEQUENCE

4.3.13.2.1 PARAMETERS

SLIDE = Desired slide number (1-120_8)

Projector number = PROJ1 or PROJ2

4.3.13.2.2 CALL

```
CALL LOCATE
PROJ1 or PROJ2  /LOGICAL
                 UNIT #
```

4.3.12.2.3 NORMAL RETURN

LOCATE returns to the location following the call.

4.3.13.2.4 ERROR RETURN

Type 1 SYSERR code:

142 -- Projector 1 request error

143 -- Projector 2 request error

4.3.13.3 EXAMPLE

To position slide 4 on Projector 1:

```
LAC (4)      /LOAD SLIDE NUMBER
DAC SLIDE    /STORE IT IN THE IMP CELL
CALL LOCATE  /CALL THE LOCATE ROUTINE
PROJ1        /FOR PROJECTOR 1
             /RETURNS HERE
```

4.3.14 LITON

4.3.14.1 FUNCTION

LITON turns on the slide projector light.

4.3.14.2 CALLING SEQUENCE

4.3.14.2.1 PARAMETERS

Projector number -- PROJ1 or PROJ2

4.3.14.2.2 CALL

```
CALL LITON
PROJ1    /LOGICAL UNIT #
```

4.3.14.2.3 NORMAL RETURN

LITON returns to the location following the call.

4.3.14.2.4 ERROR RETURN

Type 1 SYSERR code:

142 -- Projector 1 request error

143 -- Projector 2 request error

4.3.14.3 EXAMPLE

To turn on the light in projector 2:

```
CALL LITON      /CALL LITON ROUTINE
PROJ2           /FOR PROJECTOR 2
                /RETURNS HERE
```

4.3.15 LITOFF

4.3.15.1 FUNCTION

LITOFF enables the user to turn off a specified projector light.

4.3.15.2 CALLING SEQUENCE

4.3.15.2.1 PARAMETERS

The projector number is specified by an in-line parameter. The IMP defined symbols PROJ1 or PROJ2 may be used.

4.3.15.2.2 CALL

```
CALL LITOFF
PROJ1 -- Projector Number
```

4.3.15.2.3 NORMAL RETURN

LITOFF returns to the location following the projector number parameter.

4.3.15.2.4 ERROR RETURN

Type 1 SYSERR codes:

142 -- Request error - Projector 1

143 -- Request error - Projector 2

4.3.15.3 EXAMPLE

To turn the light off on projector 1:

```
CALL LITOFF        /GO TO LITOFF ROUTINE
PROJ1              /REFERENCE PROJECTOR 1
                   /RETURNS HERE
```

4.3.16 TYPE

4.3.16.1 FUNCTION

TYPE prints a multi-character message on a teletype or dataphone teletype.

4.3.16.2 CALLING SEQUENCE

4.3.16.2.1 PARAMETERS

The first word of the message to be typed must contain the message length (number of characters in octal). The message text should be in packed ASCII, SOURCE = message address.

An in-line parameter following the call must indicate the unit being referenced.

4.3.16.2.2 CALL

SOURCE = Message Address
CALL TYPE
TTY1 -- Logical Unit #

4.3.16.2.3 NORMAL RETURN

TYPE returns to the location following the teletype number parameter.

4.3.16.2.4 ERROR RETURN

Type 1 SYSERR codes:

146 -- Request Error - TELETYPE 1

147 -- Request Error - TELETYPE 2

150 -- Request Error - TELETYPE 3

4.3.16.3 EXAMPLE

To type a message called MSG1 on teletype 1:

```
   LAC (MSG1)        /LOAD ADDRESS OF MSG1
   DAC SOURCE        /STORE IN SOURCE
   CALL TYPE         /GO TO TYPE ROUTINE
   TTY1              /TYPE MSG ON TTY1
                     /RETURNS HERE

MSG1  12             /MSG LENGTH (OCTAL)
   324331            /T - Y
   320305            /P - E
   240324            /  - T
   305323            /E - S
   324256            /T - .
```

4.3.17 TYPKEY

4.3.17.1 FUNCTION

TYPKEY types a single character on the teletype.

4.3.17.2 CALLING SEQUENCE

4.3.17.2.1 PARAMETERS

SOURCE = Character to be typed

Teletype number -- TTY1, TTY2, or TTY3

4.3.17.2.2 CALL

```
CALL TYPKEY
TTY 1      /LOGICAL UNIT #
```

4.3.17.2.3 NORMAL RETURN

TYPKEY returns to the location following the call.

4.3.17.2.4 ERROR RETURN

Type 1 SYSERR codes:

146 -- Teletype 1 request error

147 -- Teletype 2 request error

150 -- Teletype 3 request error

4.3.17.3 EXAMPLE

To type the letter "A" on Teletype number 2:

```
LAC (301)         /LOAD AN ASCII "A"
DAC SOURCE        /PUT IT IN SOURCE
CALL TYPKEY       /CALL TYPKEY ROUTINE
TTY2              /FOR TTY2
                  /RETURNS HERE
```

4.3.18 SPACEB

4.3.18.1 FUNCTION

SPACEB enables the user to print one or more blank spaces on the teletype with a single call.

4.3.18.2 CALLING SEQUENCE

4.3.18.2.1 PARAMETERS

The number of spaces to be printed. Teletype number -- TTY1, TTY2, or TTY3

4.3.18.2.2 CALL

```
CALL SPACEB
TTY1   /LOGICAL UNIT #
VARBL -- Address of cell
  containing number of
  spaces
```

4.3.18.2.3 NORMAL RETURN

SPACEB returns to the location following the call.

4.3.18.2.4 ERROR RETURN

Type 1 SYSERR codes:

146 -- Teletype 1 request error

147 -- Teletype 2 request error

150 -- Teletype 3 request error

4.3.18.3 EXAMPLE

To generate six spaces on Teletype 1:

```
LAC (6)        /LOAD NUMBER OF SPACES DESIRED
DAC VARBL      /STORE IT IN USER-DEFINED CELL
CALL SPACEB    /CALL SPACEBAR ROUTINE
```

```
TTY1          /TELETYPE NUMBER
VARBL         /ADDRESS OF CELL CONTAINING
               NUMBER OF REPETITIONS
              /RETURNS HERE
```

4.3.19 FEED

4.3.19.1 FUNCTION

FEED prints one or more linefeeds on a user specified teletype or dataphone teletype.

4.3.19.2 CALLING SEQUENCE

4.3.19.2.1 PARAMETERS

A user variable must contain the number of linefeeds desired. The teletype unit number must be specified.

VARBL = number of repetitions (octal). The teletype unit number 1, 2, 3. (IMP defined TTY1, TTY2, or TTY3 may be used.)

4.3.19.2.2 CALL

```
CALL FEED
TTYN   /LOGICAL UNIT #
VARBL -- Address of cell
  containing linefeeds
```

4.3.19.2.3 NORMAL RETURN

FEED returns to the location following the teletype number parameter.

4.3.19.2.4 ERROR RETURN

Type 1 SYSERR codes:

146 -- Request error - TELETYPE 1

147 -- Request error - TELETYPE 2

150 -- Request error - TELETYPE 3

4.3.19.3 EXAMPLE

To output six linefeeds on Teletype 2:

```
LAC (6)        /LOAD REPETITION COUNT
DAC VARBL      /STORE IN USER VARIABLE
CALL FEED      /GO TO FEED ROUTINE
TTY2           /TELETYPE NUMBER
VARBL          /CELL CONTAINING REPETITION
               /COUNT
               /RETURNS HERE
```

4.3.20 GETKEY

4.3.20.1 FUNCTION

GETKEY monitors a teletype for a user specified time period.

4.3.20.2 CALLING SEQUENCE

4.3.20.2.1 PARAMETERS

VARBL = Time delay in full seconds (octal)

The teletype unit number 1, 2, or 3 must be specified. (IMP defined TTY1, TTY2, or TTY3 may be used.)

4.3.20.2.2 CALL

```
            CALL GETKEY
   TTY1       /LOGICAL UNIT #
            VARBL -- Address of cell
              containing time delay
```

4.3.20.2.3 NORMAL RETURN

GETKEY returns with the parameter answer to the location following the VARBL parameter.

ANSWER = Zero indicates no response was made.

4.3.20.2.4 ERROR RETURN

Type 1 SYSERR codes:

146 -- Request error - TELETYPE 1

147 -- Request error - TELETYPE 2

150 -- Request error - TELETYPE 3

4.3.20.3 EXAMPLE

To activate Teletype 1 wait 10 seconds for a response:

```
LAC (12)        /LOAD 10 SECONDS TIME DELAY
DAC VARBL       /STORE IN USER VARIABLE
CALL GETKEY     /GO TO GETKEY ROUTINE
TTY1            /MONITOR TTY1
VARBL           /CELL CONTAINING TIME DELAY
LAC ANSWER      /LOAD RESPONSE AND CONTINUE
```

4.3.21 MUMBLE

4.3.21.1 FUNCTION

MUMBLE plays a specified message or messages on a CROW random-access audio unit.

4.3.21.2. CALLING SEQUENCE

4.3.21.2.1 PARAMETERS

SOURCE - Message stack address

NOTE: The message stack must contain no more than five message parameters. The stack must be terminated with an asterisk. (See system documentation .3.CROW for detailed information on message parameters.)

4.3.21.2.2 CALL

```
SOURCE = Message stack
           address
CALL MUMBLE
```

4.3.21.2.3 NORMAL RETURN

MUMBLE returns to the location following the call.

4.3.21.2.4 ERROR RETURN

Type 1 SYSERR codes:

133 -- SPEAK request error

156 -- Attempt to stack more than five message parameters

4.3.21.3 EXAMPLE

To play the audio message labeled

```
CROMES:
   LAC (CROMES)     /LOAD MESSAGE STACK ADDRESS
   DAC SOURCE       /STORE IT IN SOURCE
   CALL MUMBLE      /CALL MUMBLE
                    /RETURNS HERE
```

4.3.22 PUNCHO

4.3.22.1 FUNCTION

PUNCHO punches a user-specified buffer on paper tape. Unless otherwise specified, the buffer is punched in alphanumeric mode (ASCII).

4.3.22.2 CALLING SEQUENCE

4.3.22.2.1 PARAMETERS

SOURCE = Buffer address

4.3.22.2.2 CALL

CALL PUNCHO

4.3.22.2.3 NORMAL RETURN

PUNCHO returns to the location following the call.

4.3.22.2.4 ERROR RETURN

Type 1 SYSERR code:

134 -- Punch request error

4.3.22.3 EXAMPLE

To punch a buffer labeled "PBUFF":

```
LAC (PBUFF)     /LOAD THE BUFFER ADDRESS
DAC SOURCE      /STORE IT IN SOURCE
CALL PUNCHO     /CALL PUNCHOUT ROUTINE
     .          /CONTINUE
     .          /
```

4.3.23 PRINT

4.3.23.1 FUNCTION

PRINT is used to print a buffer on the line printer. A maximum of 120 characters may be printed per line.

4.3.23.2 CALLING SEQUENCE

4.3.23.2.1 PARAMETERS

SOURCE = Buffer address

One of the following forms-control parameters must be included in each call on PRINT: TOF - Top-of-form
SINGLE -- Single space after printing
DOUBLE -- Double space after printing
TRIPLE -- Triple space after printing

4.3.23.2.2 CALL

CALL PRINT
FORMS CONTROL PARAMETER

4.3.23.2.3 NORMAL RETURN

PRINT returns to the location following the call.

4.3.23.2.4 ERROR RETURN

Type 1 SYSERR code:
135 -- Printer request error

4.3.23.3 EXAMPLE

To print a buffer labeled "PRTBUF", specifying triple spacing:

```
  LAC (PRTBUF)      /LOAD BUFFER ADDRESS
  DAC SOURCE        /PUT IT IN SOURCE
  CALL PRINT        /CALL PRINT ROUTINE
  TRIPLE            /TRIPLE SPACING
     .              /CONTINUE
```

4.3.24 DISK

4.3.24.1 FUNCTION

DISK enables the user to interact with the system Disk File Management System (DISKUS). The user should become familiar with DISKUS before attempting to use the IMP DISK routine.

4.3.24.2 CALLING SEQUENCE

4.3.24.2.1 PARAMETERS

Certain operations require that the AC be initialized prior to the call. This is true in the CREATE, BACK, FORWARD, READ, WRITE and WREOF functions.

AC = (CREATE) desired record length (octal)
(BACK) number of records to backspace (octal)
(FORWRD) number of records to skip forward (octal)
(READ) address into which data is to be read
(WRITE) Address from which data is to be written
(WREOF) address from which data is to be written

In all other operations, the contents of the AC are ignored.

In-line parameters (all operations): Immediately following the call, the operation to be executed must be specified. The following IMP defined equivalences may be used:

CREATE	(1)	-- Create new file
OPEN	(2)	-- Open old file
CLOSE	(3)	-- Close file
DELETE	(4)	-- Close and delete file
BACK	(5)	-- Backspace record(s)
FORWRD	(6)	-- Skip forward record(s)
READ	(7)	-- Read one record
WRITE	(10)	-- Write one record
EOF	(11)	-- Skip to end-of-file
REWIND	(12)	-- Skip to beginning-of-file
WREOF	(13)	-- Write one record at end-of-file

The next in-line parameter must specify an address where the file name in packed ASCII is to be found: NOTE: The file name area (three cells) must contain a predetermined "name" formatted in packed ASCII (two characters per word, six characters total).

4.3.24.2.2 CALL

```
CALL DISK
OPERATION
VARBL
```

4.3.24.2.3 NORMAL RETURN

DISK returns to the location following the user's "VARBL" parameter. The AC will contain:

AC = 400000 or 400000 + -- Successful operation

OTHER = Any other AC code is indicative of an unsuccessful operation. See below.

4.3.24.2.4 ERROR RETURN

Type 1 SYSERR codes:

136 -- Invalid DISK command

137 -- Link set, SYSTEM out of MEMAL

140 -- Attempt to open fourth file

AC codes returned to user:

1 -- DISK out of file space

2 -- Attempt to create a file that already exists

3 -- Attempt to open a non-existent file

4 -- Attempt to open an open file

5 -- Attempt to operate on a file that is not open

6 -- Attempt to backspace beyond beginning-of-file

7 -- Attempt to operate beyond end-of-file

10 -- Parity/timing error on read or write

11 -- Maximum file size exceeded

12 -- Open file list full

13 -- Maximum number of files in system

4.3.24.3 EXAMPLE

To read one record from a currently "OPEN" disk file into an area called "INBUF": (IMPORTANT: It is assumed that a three cell area called "NAME" was initialized to a packed ASCII file name.)

```
LAC (NAME)     /LOAD ADDRESS OF FILE NAME AREA
DAC VARBL      /STORE IN USER'S VARIABLE
LAC (INBUF)    /LOAD ADDRESS OF USER'S BUFFER
CALL DISK      /GO TO DISK ROUTINE
READ           /OPERATION
VARBL          /POINTER TO CELL CONTAINING
               /ADDRESS OF FILE
               /NAME AREA
SMA            /SUCCESSFUL (400000)?
JMP HELP       /NO, GO EVALUATE ERROR
               /YES, CONTINUE

HELP SAD (10)  /PARITY ERROR?
JMP ERROR1     /YES, GO TO ERROR1
SAD (5)        /FILE NOT OPEN?
JMP OPEN       /YES, GO TO OPEN FILE
               /ETC.
```

4.3.25 LOGOFF

4.3.25.1 FUNCTION

LOGOFF terminates a job and does the following:

1. Releases MEMAL (if any) obtained by SETUP and/or DEFINE
2. Types a message on control teletype:
   "JOBNR - JN - HAS TERMINATED SUCCESSFULLY!"
3. Suspends the job (DEAD, 200000) for operator Intervention.

4.3.25.2 CALLING SEQUENCE

4.3.25.2.1 PARAMETERS

None

4.3.25.2.2 CALL

LOGOFF

4.3.25.2.3 NORMAL RETURN

LOGOFF does not return.

4.3.25.2.4 ERROR RETURN

None

4.3.25.3 EXAMPLE

To release MEMAL, type CTTY message and suspend the job:

```
   LOGOFF     /GO TO LOGOFF ROUTINE
```

4.4

MISCELLANEOUS FUNCTION ROUTINES

4.4 MISCELLANEOUS FUNCTION ROUTINES

4.4.1 BINDEC

4.4.1.1 FUNCTION

BINDEC converts up to 18 binary digits to decimal ASCII characters. The ASCII characters are deposited in a buffer, one per word, with a user-specified bit configuration in the left-most nine bits (i.e., 000, 240, 377, etc.). The user may specify suppression of leading zeroes.

4.4.1.2 CALLING SEQUENCE

4.4.1.2.1 PARAMETERS

SOURCE = Number to be converted

RECEVE = Buffer address (+ 400000 if zero suppression is desired).

Number of digits to convert.

Desired value in left-most nine bits.

4.4.1.2.2 CALL

CALL BINDEC

DESIRED HIGH ORDER VALUE + NUMBER OF OCTAL DIGITS TO CONVERT (1-6)

4.4.1.2.3 NORMAL RETURN

BINDEC returns to the location following the call with the ASCII characters in the user's buffer.

4.4.1.2.4 ERROR RETURN

None

4.4.1.3 EXAMPLE

To convert an 18-BIT binary number to decimal ASCII characters with an ASCII rutout in the high order of each word:

```
LAC VARBL      /LOAD SOME NUMBER TO BE
                CONVERTED
DAC SOURCE     /STORE IN SOURCE
LAC BUFFER     /LOAD BUFFER ADDRESS
DAC RECEVE     /STORE IN RECEVE
CALL BINDEC    /CALL BINDEC ROUTINE
377006         /HIGH ORDER = RUBOUT -- SIX
                OCTAL DIGITS
               /CONTINUE
```

4.4.2 CHECK1

4.4.2.1 FUNCTION

CHECK1 compares the contents of ANSWER with a specified list of data.

4.4.2.2 CALLING SEQUENCE

4.4.2.2.1 PARAMETERS

SOURCE = address of list to be searched

ANSWER = Value to seek

4.4.2.2.2 CALL

CALL CHECK1

4.4.2.2.3 NORMAL RETURN

CHECK1 returns to the location following the call with the following parameters:

REMAIN = 0, if a match for ANSWER is found

REMAIN ≠ 0, if no match is found

4.4.2.2.4 ERROR RETURNS

None

4.4.2.3 EXAMPLE

To see if a given window number from a touch-sensitive screen is in a list of valid responses called "VALRES":

```
     .         /GET A RESPONSE FROM THE TOUCH
                SENSITIVE
  LAC (VALRES) /LOAD ADDRESS OF VALID RESPONSE
                LIST
  DAC SOURCE   /STORE IN SOURCE
  CALL CHECK1  /CALL CHECK1 W/RESPONSE IN
                ANSWER
  LAC REMAIN   /EVALUATE FINDINGS
     .         /
```

4.4.3 CLEAR

4.4.3.1 FUNCTION

CLEAR enables the user to zero a specified portion of memory.

4.4.3.2 CALLING SEQUENCE

4.4.3.2.1 PARAMETERS

The area to be zeroed must be terminated with an asterisk (ASCII 252).

SOURCE = Address of area to be cleared

4.4.3.2.2 CALL

CALL CLEAR

4.4.3.2.3 NORMAL RETURN

CLEAR returns to the location following the call.

4.4.3.2.4 ERROR RETURN

None

4.4.3.3 EXAMPLE

To zero an area called "BUFFER":

```
   LAC (.AST)      /LOAD AN ASTERISK
   DAC ENDBUF      /STORE AT END OF BUFFER
   LAC (BUFFER)    /LOAD ADDRESS OF BUFFER
   DAC SOURCE      /STORE IN SOURCE
   CALL CLEAR      /GO TO CLEAR ROUTINE
                   /RETURNS HERE
BUFFER 0           /USER'S BUFFER
       0
ENDBUF 252         /TERMINATING ASTERISK
```

4.4.4 DECBIN

4.4.4.1 FUNCTION

DECBIN converts up to six ASCII digits to an 18-bit binary number.

4.4.4.2 CALLING SEQUENCE

4.4.4.2.1 PARAMETERS

The ASCII buffer to be converted may contain low order "LAMS" (777777). These are ignored.

NOTE: The high order half of each ASCII character to be converted will have no effect on the result.

SOURCE = Beginning address of ASCII buffer

The in-line parameter (1-6) indicates the number of words to be converted.

4.4.4.2.2 CALL

```
CALL DECBIN
6         (FIELD WIDTH)
```

4.4.4.2.3 NORMAL RETURN

DECBIN returns to the location following the field width with the parameter:

ANSWER = The binary result

4.4.4.2.4 ERROR RETURN

None

4.4.4.3 EXAMPLE

To convert a six word ASCII buffer to its binary equivalent:

```
LAC (BUFFER)      /LOAD ADDRESS OF BUFFER
DAC SOURCE        /STORE IN SOURCE
CALL DECBIN       /GO TO CONVERT IT
6                 /FIELD WIDTH
LAC ANSWER        /LOAD BINARY NUMBER
                  /CONTINUE

BUFFER 260        /ZERO
       260        /ZERO
       261        /ONE
       270        /SEVEN
       271        /NINE
       263        /THREE
```

4.4.5 DIVIDE

4.4.5.1 FUNCTION

DIVIDE enables the user to divide one signed integer by another.

4.4.5.2 CALLING SEQUENCE

4.4.5.2.1 PARAMETERS

SOURCE = Dividend

RECEVE = Divisor

4.4.5.2.2 CALL

CALL DIVIDE

4.4.5.2.3 NORMAL RETURN

DIVIDE returns to the location following the call with the following parameters:

ANSWER = Quotient

REMAIN = Remainder

4.4.5.2.4 ERROR RETURN

Type 1 SYSERR code:

157 -- Attempt to divide by zero

4.4.5.3 EXAMPLE

To divide 10_8 by 2_8:

```
LAC (10)         /LOAD DIVIDEND
DAC SOURCE       /STORE IN SOURCE
LAC (2)          /LOAD DIVISOR
DAC RECEVE       /STORE IN RECEVE
CALL DIVIDE      /CALL DIVIDE ROUTINE
LAC ANSWER       /EVALUATE RESULTS
```

4.4.6 DO

4.4.6.1 FUNCTION

DO executes a subroutine a variable number of times. DOs may be nested three deep but must be denested in the same order in which they are nested. Branching out of a DO is forbidden.

4.4.6.2 CALLING SEQUENCE

4.4.6.2.1 PARAMETERS

There are two in-line parameters:

(a) The subroutine address

(b) Address of cell containing number of repetitions.

NOTE: The routine to be executed must be terminated by a "RETURN" statement.

4.4.6.2.2 CALL

```
          CALL DO
   SUBR  /ADDRESS OF SUBROUTINE
   VARBL /ADDRESS OF RFPETITIONS CELL
```

4.4.6.2.3 NORMAL RETURN

DO returns to the location following the call.

4.4.6.2.4 ERROR RETURN

Type 1 SYSERR codes:

167 -- Attempt to nest more than three deep

170 -- Attempt to denest out of order

4.4.6.3 EXAMPLE

To execute a subroutine called SUBR1 three times:

```
   LAC (3)          /LOAD NUMBER OF REPETITIONS
                     DESIRED
   DAC VARBL        /STORE IN CELL CALLED VARBL
   CALL DO          /CALL DO ROUTINE
   SUBR1            /SUBROUTINE ADDRESS
   VARBL            /ADDRESS OF REPETITIONS CELL
     .              /CONTINUE
```

4.4.7 ERROR

4.4.7.1 FUNCTION

ERROR serves as the IMP error handler. ERROR makes a call on Executive System Routine SYSERR, requesting a Type 1 SYSERR. IMP users may use ERROR, providing they restrict their error codes to the range 31_8 - 77_8.

4.4.7.2 CALLING SEQUENCE

4.4.7.2.1 PARAMETERS

AC = Error Code Number

4.4.7.2.2 CALL

JMP ERROR

4.4.7.2.3 NORMAL RETURN

ERROR does not return. Job is suspended on "DEAD" (operator intervention).

4.4.7.2.4 ERROR RETURN

Not applicable

4.4.7.3 EXAMPLE

```
LAW 31          /LOAD ERROR CODE
JMP ERROR       /GO TO ERROR ROUTINE
```

4.4.8 EXPO

4.4.8.1 FUNCTION

EXPO exponentiates any given value.

4.4.8.2 CALLING SEQUENCE

4.4.8.2.1 PARAMETERS

SOURCE = Value

RECEVE = Exponent

4.4.8.2.2 CALL

CALL EXPO

4.4.8.2.3 NORMAL RETURN

EXPO returns to the location following the call with the result in ANSWER.

4.4.8.2.4 ERROR RETURN

A set LINK upon return indicates AC overflow.

4.4.8.3 EXAMPLE

To raise the value 10_8 to the 5th power:

```
          LAC (10)       /LOAD OCTAL VALUE
          DAC SOURCE     /STORE IN SOURCE
          LAC (5)        /LOAD OCTAL EXPONENT
          DAC RECEVE     /STORE IN RECEVE
          CALL EXPO      /GO TO EXPO ROUTINE
          SZL            /OVERFLOW?
          JMP ERROR1     /YES TO PROCESS ERROR
          LAC ANSWER     /LOAD RESULT
                         /CONTINUE

ERROR 1   LAW 30         /LOAD ERROR CODE
          JMP ERROR      /GO TO ERROR ROUTINE
```

4.4.9 LOAF

4.4.9.1 FUNCTION

LOAF is a control routine which allows the user to suspend a program for a specified number of seconds.

4.4.9.2 CALLING SEQUENCE

4.4.9.2.1 PARAMETERS

In-line parameter following the call must indicate the time delay in seconds.

4.4.9.2.2 CALL

```
CALL LOAF
SECONDS (octal number)
```

4.4.9.2.3 NORMAL RETURN

LOAF returns to the location following the time delay parameter upon expiration of delay period.

4.4.9.2.4 ERROR RETURN

Type 0 SYSERR code:

104 -- Time delay was zero or greater than 143470_8 (s.144_8 > 143470_8)

4.4.9.3 EXAMPLE

To suspend a program operation for 10 seconds:

```
CALL LOAF       /GO TO LOAF
12              /NUMBER OF SECONDS (OCTAL)
                /RETURNS HERE
```

4.4.10 LOOKUP

4.4.10.1 FUNCTION

LOOKUP maintains a pointer to the current CROW segment. On request, it will update the segment pointer for each message played. Optionally, LOOKUP will add the current CROW segment to a user CROW message before updating the segment pointer.

4.4.10.2 CALLING SEQUENCE

4.4.10.2.1 PARAMETERS

SOURCE = 400000 + Track,
Length and Segment
to update the CROW
segment pointer

SOURCE = 000000 + Track,
Length to add the
current segment
and reset the
segment pointer

4.4.10.2.2 CALL

CALL LOOKUP

4 4.10.2.3 NORMAL RETURN

LOOKUP returns to the location following the call. ANSWER will contain the Crow belt segment address.

4 4.10.2.4 ERROR RETURN

Not Applicable

4.4.10.3 EXAMPLE

To set the IMP SEGPTR to reflect the Crow address indicated in CROWAD:

```
LAC CROWAD        /LOAD TLS
TAD (400000)      /ADD 400000
CALL LOOKUP       /GO TO LOOKUP ROUTINE
LAC ANSWER        /LOAD SEGMENT ADDRESS
DAC CROWAD        /RESTORE IN CROWAD
                  /CONTINUE
```

4.4.11 MEAN

4.4.11.1 FUNCTION

MEAN enables the user to determine the arithmetic mean of a series of values.

4.4.11.2 CALLING SEQUENCE

4.4.11.2.1 PARAMETERS

The series of values must be terminated by an asterisk (ASCII 252). Care must be taken to insure that the sum of the values will not cause AC overflow.

SOURCE = Starting address of series of values.

4.4.11.2.2 CALL

CALL MEAN

4.4.11.2.3 NORMAL RETURN

MEAN returns to the location immediately following the call with the parameter:

ANSWER = Arithmetic mean

4.4.11.2.4 ERROR RETURN

None

4.4.11.3 EXAMPLE

To determine the mean of a series of numbers beginning in location BUFFER:

```
LAC (.AST)     /LOAD AN ASTERISK
DAC ENDBUF     /STORE AT END OF BUFFER
LAC (BUFFER)   /LOAD BEGINNING ADDRESS
DAC SOURCE     /STORE IN SOURCE
CALL MEAN      /GO TO AVERAGE THE VALUES
               /CONTINUE
```

```
 BUFFER  1     /OCTAL VALUES TO BE AVERAGED
         2     /
         3     /
         4     /
         5     /
         6     /
         7     /
        10     /
ENDBUF 252     /TERMINATOR
```

4.4.12 MOVE

4.4.12.1 FUNCTION

MOVE moves the contents of one or more locations to another set of locations.

4.4.12.2 CALLING SEQUENCE

4.4.12.2.1 PARAMETERS

SOURCE = Address of sending area

RECEVE = Address of receiving area

The end of the area to be moved must be marked by a location containing an asterisk.

4.4.12.2.2 CALL

CALL MOVE

4.4.12.2.3 NORMAL RETURN

MOVE returns to the location following the call.

4.4.12.2.4 ERROR RETURN

None

4.4.12.3 EXAMPLE

To move data from AREA1 to AREA2:

```
LAC (AREA1)      /LOAD ADDRESS OF SENDING AREA
DAC SOURCE       /STORE IN SOURCE
LAC (AREA2)      /LOAD ADDRESS OF RECEIVING
                  AREA
DAC RECEVE       /STORE IN RECEVE
CALL MOVE        /CALL MOVE ROUTINE
    .            /CONTINUE
    .            /
    .            /
```

```
AREA 1  301     /AREA1 BEFORE AND AFTER MOVE
        302     /
        303     /
        304     /
        252     /

AREA 2    0     /AREA2 BEFORE MOVE
          0     /
          0     /
          0     /
          0     /
          0     /
          0     /

AREA 2  301     /AREA2 AFTER MOVE
        302     /
        303     /
        304     /
          0     /NOTE THAT THE ASTERISK IS
                 NOT MOVED
          0     /
          0     /
```

4.4.13 MULTIP

4.4.13.1 FUNCTION

MULTIP (MULTIPLY) multiplies one value by another.

4.4.13.2 CALLING SEQUENCE

4.4.13.2.1 PARAMETERS

SOURCE = Multiplicand

RECEVE = Multiplier

4.4.13.2.2 CALL

CALL MULTIP

4.4.13.2.3 NORMAL RETURN

MULTIP returns to the next location following the call with the following parameter:

ANSWER = Product

4.4.13.2.4 ERROR RETURN

None

4.4.13.3 EXAMPLE

To multiply 13_8 by 10_8:

```
LAC (13)        /LOAD MULTIPLICAND
DAC SOURCE      /STORE IN SOURCE
LAC (10)        /LOAD MULTIPLIER
DAC RECEVE      /STORE IN RECEVE
CALL MULTIP     /CALL MULTIPLY ROUTINE
LAC ANSWER      /CONTINUE
   .            /
```

4.4.14 RANDOM

4.4.14.1 FUNCTION

RANDOM generates a 1- to 6-digit random octal number.

4.4.14.2 CALLING SEQUENCE

4.4.14.2.1 PARAMETERS

One in-line parameter to specify the width of the random number must be included in the call on RANDOM.

4.4.14.2.2 CALL

```
CALL RANDOM
1, 2, 3, 4, 5, or 6
     /FIELD WIDTH
```

4.4.14.2.3 NORMAL RETURN

RANDOM returns to the location following the call with the random number in ANSWER.

4.4.14.2.4 ERROR RETURN

None

4.4.14.3 EXAMPLE

To generate a 4-digit random number:

```
CALL RANDOM      /CALL RANDOM ROUTINE
4                /FOR A 4-DIGIT NUMBER
LAC ANSWER       /CONTINUE
     .           /
```

4.4.15 TIME1

4.4.15.1 FUNCTION

TIME1 returns the complimented time-of-day. TIME1 is usually used in conjunction with the TIME2 routine to determine the elapsed time (latency) between one event and another.

4.4.15.2 CALLING SEQUENCE

4.4.15.2.1 PARAMETERS

None

4.4.15.2.2 CALL

CALL TIME1

4.4.15.2.3 NORMAL RETURN

TIME1 returns to the location following the call with the parameter:

TIME = Complimented TOD

4.4.15.2.4 ERROR RETURN

None

4.4.15.3 EXAMPLE

To obtain the complimented time-of-day:

CALL TIME1

(Normally, some input device is activated here for a subject or user response after which TIME2 is called.)

4.4.16 TIME2

4.4.16.1 FUNCTION

TIME2, when used in conjunction with TIME1, enables the user to determine the elapsed time between one event and another.

4.4.16.2 CALLING SEQUENCE

4.4.16.2.1 PARAMETERS

TIME = Complimented TOD obtained at same previous event by TIME1

4.4.16.2.2 CALL

CALL TIME2

4.4.16.2.3 NORMAL RETURN

TIME2 adds the complimented time-of-day in the "TIME" cell (obtained by TIME1) to the current time, deposits the result in the "TIME" cell, and returns to the user.

NOTE: Latency is returned in thousandths of seconds with the assumed decimal point between the third and fourth octal digits.

4.4.16.2.4 ERROR RETURN

None

4.4.16.3 EXAMPLE

Assuming TIME1 has been called at some previous point, a call may be made to TIME2 to obtain a latency value:

```
CALL TIME2      /GO FOR LATENCY
LAC TIME        /LOAD THE LATENCY
                /EVALUATE
```

4.4.17 TOTAL

4.4.17.1 FUNCTION

TOTAL computes the sum of a series of values.

4.4.17.2 CALLING SEQUENCE

4.4.17.2.1 PARAMETERS

SOURCE = Beginning address of the list of numbers

NOTE: The list of numbers must be terminated by an asterisk.

4.4.17.2.2 CALL

CALL TOTAL

4.4.17.2.3 NORMAL RETURN

TOTAL returns to the location following the call with the following parameter:

ANSWER = Sum of the series of numbers

4.4.17.2.4 ERROR RETURN

None

4.4.17.3 EXAMPLE

To find the sum of a series of numbers beginning at NUMLST:

```
LAC (NUMLST)    /LOAD THE BEGINNING ADDRESS
                 OF THE NUMBERS
DAC SOURCE      /PUT IT IN SOURCE
CALL TOTAL      /CALL TOTAL ROUTINE
LAC ANSWER      /CONTINUE
     .          /
     .          /
     .          /
```

```
NUMLST  1036    /BEGINNING OF LIST OF NUMBERS
          12    /
         364    /
         721    /
       53047    /
          33    /
         252    /LIST TERMINATED BY AN
                 ASTERISK
```

REFERENCES

Judd, Wilson A. The development of an on-line laboratory for CAI and behavioral research (1964-1968). Technical Report. Pittsburgh, Pennsylvania: Learning Research and Development Center, University of Pittsburgh, 1969.

Nemitz, Bertram P. SKOOLBOL: A simplified user's language for programming the PDP-7. Working Manual. Pittsburgh, Pennsylvania: Learning Research and Development Center, University of Pittsburgh, 1968

APPENDIX A: RESERVED WORDS

LOCATION NAMES

ACSAVE
ADRESS
ANSWER
COMMON
CONTEN
ITEM
PUSHJ
PUSHJ2
PUSHJ3
RECEVE
REMAIN
RETADD
*SKCELL
*SKC1
⋮
*SKC56
*SKEND
SLIDE
SOURCE
STACK
STKBSE
STKPTR
TIME
TRNVEC

*All tags beginning with the letters "SK" are reserved for IMP.

ROUTINE NAMES

BACKUP
BINDEC
CHECK1
CLEAR
DECBIN
DEFINE
DISK
DISPLA
DIVIDE
DO
ERASE
ERROR
EXCEPT
EXPO
FEED
FIND
GETKEY
LITOFF
LITON
LOAF
LOCATE
LOGOFF
LOGON
LOGOND
LOGONT
LOOKUP
MEAN
MOVE
MULTIP
MUMBLE
OBTAIN
PARAM
POPJ
PRINT
PUNCHO
RANDOM
READKY
RELESE
SAVE
SETUP
SHOLET
SKIP
SPACE
SPACEB
STEPUP
STORE
TAPE
TIME1
TIME2
TOTAL
TYPE
TYPKEY
TUCH

APPENDIX B: I/O PARAMETER LISTS

LOCATION NUMBER	DEVICE	CONTENT	USAGE
1	SCREEN	300001	STORAGE MODE, TEXT, UNIT ONE
2		0	
3		0	
4		0	
5		0	
6		0	
7		0	
8		0	
9	KEYBOARD	200001	TIME DELAY, UNIT ONE
10		0	
11		0	
12	TOUCH	1	UNIT ONE
13		0	
14		0	
15		0	
16		0	
17		0	
18	PROJECTOR	1200	UNIT ONE, SUSPEND
19		2200	UNIT TWO, SUSPEND
20		0	
21		0	
22	TTY/DPHONE	240001	READ, UNIT ONE, TIME DELAY, ECHO
23		240002	READ, UNIT TWO, TIME DELAY, ECHO
24		240003	READ, UNIT THREE, TIME DELAY, ECHO
25		400001	PRINT, UNIT ONE, SUSPEND
26		400002	PRINT, UNIT TWO, SUSPEND
27		400003	PRINT, UNIT THREE, SUSPEND
28		0	
29		0	

LOCATION NUMBER	DEVICE	CONTENT	USAGE
30		0	
31	CROW	1	UNIT ONE
32		0	
33		0	
34		0	
35		0	
36		0	
37		0	
38	PUNCH	400000	SUSPEND, PUNCH ASCII
39		0	
40	PRINTER	400000+ COMM CELL	SUSPEND, COMM CELL ADDRESS
41		0	
42		0	
43	MAGTAPE	0	
44		0	
45	DISK	0	
46		0	
47		0	
51		0	
52		0	
53		COMM CELL	COMM CELL ADDRESS
54	COMCEL	0	COMM CELL ONE
55	COMCEL	0	COMM CELL TWO

APPENDIX C: ERROR CODES

NUMBER	SIGNIFICANCE	GENERATING ROUTINE
100	GRAB ERROR-SCREEN	OBTAIN
101	GRAB ERROR-KEYBOARD	OBTAIN
102	GRAB ERROR-TOUCH	OBTAIN
103	GRAB ERROR-HYSPRJ	OBTAIN
104	GRAB ERROR-BAGTEL/DPHONE	OBTAIN
105	GRAB ERROR-CROW	OBTAIN
106	GRAB ERROR-PUNCH	OBTAIN
107	GRAB ERROR-PRINTER	OBTAIN
110	TO BE ASSIGNED	
111	TO BE ASSIGNED	
112	TO BE ASSIGNED	
113	TO BE ASSIGNED	
114	RELEASE ERROR-SCREEN	RELESE
115	RELEASE ERROR-KEYBOARD	RELESE
116	RELEASE ERROR-TOUCH	RELESE
117	RELEASE ERROR-HYSPRJ	RELESE
120	RELEASE ERROR-BAGTEL/DPHONE	RELESE
121	RELEASE ERROR-CROW	RELESE
122	RELEASE ERROR-PUNCH	RELESE
123	RELEASE ERROR-PRINTER	RELESE
124	TO BE ASSIGNED	
125	TO BE ASSIGNED	
126	TO BE ASSIGNED	
127	TO BE ASSIGNED	
130	REQUEST ERROR-SCREEN	DISPLA
131	REQUEST ERROR-KEYBOARD	READKY
132	REQUEST ERROR-TOUCH	TUCH
133	REQUEST ERROR-SPEAK	MUMBLE
134	REQUEST ERROR-PUNCH	PUNCHO

		GENERATING
NUMBER	SIGNIFICANCE	ROUTINE
135	REQUEST ERROR-PRINTER	PRINT
136	INVALID DISK COMMAND	DISK
137	REQUEST ERROR-DISKUS	DISK
140	ATTEMPT TO OPEN FOURTH FILE	DISK
141	TO BE ASSIGNED	
142	REQUEST ERROR-PROJECTOR 1	LOCATE
143	REQUEST ERROR-PROJECTOR 2	LOCATE
144	TO BE ASSIGNED	
145	TO BE ASSIGNED	
146	REQUEST ERROR-TELETYPE 1-PRINT	TYPE
147	REQUEST ERROR-TELETYPE 2-PRINT	TYPE
150	REQUEST ERROR-TELETYPE 3-PRINT	TYPE
151	REQUEST ERROR-TELETYPE 1-READ	GETKEY
152	REQUEST ERROR-TELETYPE 2-READ	GETKEY
153	REQUEST ERROR-TELETYPE 3-READ	GETKEY
154	TO BE ASSIGNED	
155	TO BE ASSIGNED	
156	ATTEMPT TO STACK MORE THAN FIVE CROW MESSAGES	MUMBLE
157	ATTEMPT TO DIVIDE BY ZERO	DIVIDE
160	PHYSICAL TAPE DRIVE FAILURE	TAPE
161	MAGTAPE PARITY ERROR	TAPE
162	MAGTAPE OTHER ERROR	TAPE
163	PUSHJ STAKC OVERFLOW	PUSHJ
164	PUSHJ STACK UNDERFLOW	POPJ
165	ATTEMPT TO ALTER NONEXITENT PARAMETER	EXCEPT
166	TO BE ASSIGNED	
167	NESTING MORE THAN THREE DEEP	DO
170	UNNESTING MORE THAN THREE DEEP	DO
171	NO "FINISH" PARAMETER	SETUP

NUMBER	SIGNIFICANCE	GENERATING ROUTINE
172	ILLEGAL RETURN ADDRESS INCREMENT	STEPUP
173	TO BE ASSIGNED	
174	TO BE ASSIGNED	
175	TO BE ASSIGNED	
176	TO BE ASSIGNED	
177	TO BE ASSIGNED	

APPENDIX D: IMP EQUIVALENCES

ASCII CODE EQUIVALENCES

MNEMONIC	OCTAL VALUE	MEANING
.A	301	UPPER CASE A
.B	302	UPPER CASE B
.C	303	UPPER CASE C
.D	304	UPPER CASE D
.E	305	UPPER CASE E
.F	306	UPPER CASE F
.G	307	UPPER CASE G
.H	310	UPPER CASE H
.I	311	UPPER CASE I
.J	312	UPPER CASE J
.K	313	UPPER CASE K
.L	314	UPPER CASE L
.M	315	UPPER CASE M
.N	316	UPPER CASE N
.O	317	UPPER CASE O
.P	320	UPPER CASE P
.Q	321	UPPER CASE Q
.R	322	UPPER CASE R
.S	323	UPPER CASE S
.T	324	UPPER CASE T
.U	325	UPPER CASE U
.V	326	UPPER CASE V
.W	327	UPPER CASE W
.X	330	UPPER CASE X
.Y	331	UPPER CASE Y
.Z	332	UPPER CASE Z
.ZERO	260	ZERO
.ONE	261	ONE

MNEMONIC	OCTAL VALUE	MEANING
.TWO	262	TWO
.THREE	263	THREE
.FOUR	264	FOUR
.FIVE	265	FIVE
.SIX	266	SIX
.SEVEN	267	SEVEN
.EIGHT	270	EIGHT
.NINE	271	NINE
.EXC	241	EXCLAMATION
.QOT	242	QUOTE
.NOS	243	NUMBER SIGN
.DOL	244	DOLLAR SIGN
.PRC	245	PERCENT
.AMP	246	AMPERSAND
.APS	247	APOSTROPHE
.LPR	250	LEFT PARENTHESIS
.RPR	251	RIGHT PARENTHESIS
.AST	252	ASTERISK
.PLU	253	PLUS SIGN
.COM	254	COMMA
.MIN	255	MINUS SIGN
.PER	256	PERIOD
.SLH	257	SLASH
.COL	272	COLON
.SCL	273	SEMI-COLON
.LTH	274	LESS THAN
.EQU	275	EQUAL SIGN
.GTH	276	GREATER THAN
.QUS	277	QUESTION MARK
.ATS	300	AT SIGN
.LBR	333	LEFT BRACKET

MNEMONIC	OCTAL VALUE	MEANING
.BSL	334	BACKWARD SLASH
.RBR	335	RIGHT BRACKET
.UAR	336	UP ARROW
.LAR	337	LEFT ARROW
.LF	212	LINEFEED
.CR	215	CARRIAGE RETURN
.SP	240	SPACE
.RO	377	RUBOUT

INSTRUCTION EQUIVALENCES

EQUIVALENCE	VALUE	MEANING
FINISH	777777	LIST TERMINATOR
TTY1	1	LOGICAL TELETYPE OR DATAPHONE #1
TTY2	2	LOGICAL TELETYPE OR DATAPHONE #2
TTY3	3	LOGICAL TELETYPE OR DATAPHONE #3
PROJ1	1	LOGICAL PROJECTOR #1
PROJ2	2	LOGICAL PROJECTOR #2
TOF	500000	TOP-OF-FORM
SINGLE	204	SINGLE SPACING (132_{10} CHAR/LINE)
DOUBLE	604	DOUBLE SPACING (132_{10} CHAR/LINE)
TRIPLE	1204	TRIPLE SPACING (132_{10} CHAR/LINE)
CREATE	1	OPEN NEW DISK FILE
OPEN	2	OPEN OLD DISK FILE
CLOSE	3	CLOSE DISK FILE
DELETE	4	CLOSE AND DELETE DISK FILE
BACK	5	BACKSPACE RECORD(S)
FORWRD	6	SKIP RECORD(S) FORWARD
READ	7	READ ONE RECORD
WRITE	10	WRITE ONE RECORD
EOF	11	SKIP FORWARD TO END-OF-FILE
REWIND	12	SKIP BACKWARD TO BEGINNING-OF-FILE
WROF	13	WRITE ONE RECORD AT END-OF-FILE

APPENDIX E: CRT CHARACTER SIZE CHART

CHAR. SIZE	SPACES (HORIZONTAL)	LINES (VERTICAL)
3	28	13
4	21	10
5	17	8
6	14	6
7	12	5