File No. S370-32
Form GC28-2038-5

Systems Reference Library

IBM Time Sharing System (TSS)

Independent Utilities

The Utility Support System (USS) is a set of functions accessible via macros which enable the systems programmer to write utilities intended to run on a dedicated CPU interfacing with a single user. Supplied with USS are four utilities which use the functions of USS -- DASDI (Initialize a DASD volume for TSS), Dump/Restore (Transfer TSS-formatted volumes), VAM2UT (Patch and/or display data on TSS-formatted volumes) and Startup (Initialize TSS).

CONTENTS

Sixth Edition (December 1977)

This is a revision of, and makes obsolete, GC28-2038-4.

This edition is current with Release 3.0 of the IBM Time Sharing System/370 (TSS/370), and remains in effect for all subsequent versions or modifications unless otherwise noted. Changes or additions to this publication will be provided in Technical Newsletters or, if changes are significant, in a new edition.

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

A form is provided at the back of this publication for reader's comments. If the form has been removed, comments may be addressed to IBM Corporation, Time Sharing System--Department 80M, 1133 Westchester Avenue, White Plains, New York 10604.

INTRODUCTION

The TSS Utility Support System (USS) is a group of functions packaged as a standalone subsystem which interfaces to system-supplied utilities or user-developed utilities. It is required for such activities as system maintenance, data set maintenance and device formatting. It is not time shared and therefore supports only a single user at a time. The operation of USS requires a dedicated CPU with only one user.

USS is normally used by system programmers. The design of USS permits the system programmer to develop utilities necessary to accomplish desired maintenance tasks that would be impractical to do under TSS, such as patching a data set that is not known to (catalogued on) the system.

The functions available to the programmer under USS are all accessible via macros. It is anticipated that, optionally, the systems programmer will code new utilities under TSS. The functional macros to develop these new utilities are accessible through a separate systems programmer macro library. The assembled code is then a new utility that will operate under USS.

Sections 1 through 3 of this manual cover the functions available to the systems programmer in USS via macros available to him in TSS. Also included in these sections is a description of USS tools for debugging interfaces with USS during standalone testing.

Section 4 covers the common commands of USS as opposed to commands which are specific to each utility.

The last section covers the commands specific to each of the utilities supplied by IBM. Since it is not expected that the systems programmer will be interfacing code with, or modifying, these utilities, this section covers only the commands required to use these utilities. For those programmers who must modify these utilities, adequate documentation exists in the respective program listings.

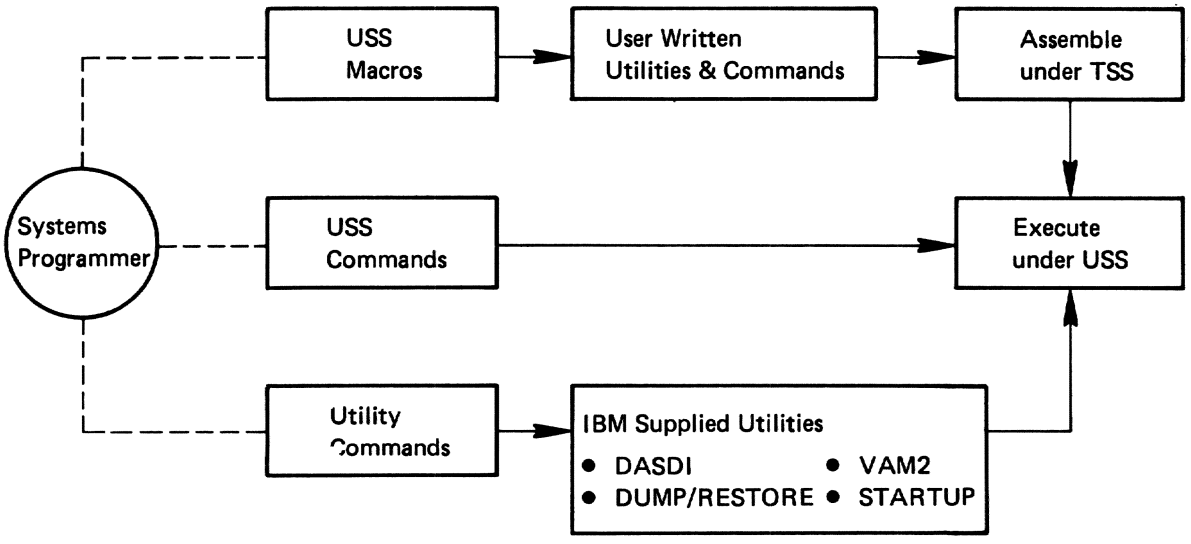

Figure 1. Utility support system overview

SECTION 1: I/O MACROS

There are four I/O macros in USS -- IAMIO, IAMOPER, IAMDEAD and IAMDAIO. IAMIO performs I/O to any device. IAMOPER which has more functions than IAMIO performs I/O to the operations console or to SYSIN or SYSOUT. (IAMOPER contains the ability to clear attentions by asking whether to continue, or abort to the calling utility.) IAMDAIO performs disk or tape I/O.

IAMDAIO -- DASD & VAM Tape I/O

IAMDAIO is used to link to a routine to perform direct access and VAM tape I/O.

Name	Operation	Operands
[symbol]	IAMDAIO	

Note: There are no operands.

Initialization: Register 1 must contain a pointer to a control block defined by the DSECT CHADIO.

Return Data: Register 1 contains a pointer to the I/O data area.

Register 15 contains a return code:

0	-	Successful
4	-	Unsuccessful
8	-	End of File
12	-	Datacheck
16	-	End of Volume

Note: Return codes 8, 12 and 16 apply to VAM tape only.

IAMIO -- Start I/O

The IAMIO macro causes a specified I/O operation to occur on a device at a specified address.

Name	Operation	Operand
[symbol]	IAMIO	PDA=,DEV=,ACTION=,[CSW=,SENSE=,RPN=,ACTADD=,EOF=, FOLD=,VERIFY=,ALARM=,PRINTER=,HEADER=,DENSITY=, MESS=,MESSLGN=,CCW=,MF=,ALTR1=,ALTR14=,ALTR15=]

PDA=
physical device address.

Specified as: either a hard coded two-byte hexadecimal self-defining term, or a label of such a term.

Examples:

```
(1)         IAMIO   PDA=X'000E'

(2)         IAMIO   PDA=PDACON
                      •
                      •
                      •
   PDACON   DC  X'000E'
```

DEV=
device type.

Specified as: either the device type number, or a label of a field containing the two-byte device type number, or the character "?".

(DEV=? is a special case and is permitted only if ACTION=AMIHERE. It means that the caller wants IAMIO to determine the device type class at the PDA specified and to return device type class to the caller. If the device type is a tape, IAMIO returns DEV=3420 regardless of the actual tape type at that PDA. Similarly, DEV=3211 is returned for any printer and DEV=3504 is returned for any card reader. For terminals and direct access devices the return will be the actual device type (not device class) at the PDA specified.)

CAUTION: The error recovery routine differs slightly for each device of a device class. If DEV=? is used and the device at the specified PDA is a tape drive, USS assumes it is a 3420. If in fact it is a 2400, some I/O errors may be incorrectly processed. If at all possible, do not use DEV=?. See the VERIFY operand.

If the device is not ready when the device type is ascertained, IAMIO may return incorrect results.

Examples:

```
(1)         IAMIO   DEV=2540

(2)         IAMIO   DEV=DEVCON
                      •
                      •
                      •
   DEVCON   DC  X'2540'
```

Permitted device types: 2301, 2305, 2311, 2314, 3330, 333B, 3350, 3215, 3066, 3270, 2400, 3420, 1402, 2540, 3504, 1403, 3211, 140A, 140B, 321A, and 321B. (1403, 140A and 140B are all 1403 printers; 3211, 321A and 321B are 3211 printers. The 1403 and 3211 printers are assumed to be printing on paper 132 characters wide; the 140A and 321A on paper 100 characters wide; the 140B and 321B on paper 45 characters wide. For all these cases, IAMIO will print 54 lines per page.)

ACTION=
the action requested of IAMIO.

Permitted actions:
ACTION=READ
ACTION=WRITE
ACTION=READCKD (Read, count, key, data)

```
ACTION=WRITECKD (Write, count, key, data)
ACTION=AMIHERE  (Is the specified device online?)
ACTION=SIOONLY  (Start I/O on the CCW list provided by the
                 calling utility)
ACTION=REWIND
ACTION=RUNLOAD  (Rewind and unload)
ACTION=FSF      (Forward space file)
ACTION=FSB      (Forward space block)
ACTION=BSF      (Backspace block)
ACTION=WTM      (Write tape mark)
```

Additional Required Operands: If ACTION=READ, WRITE, READCKD, or WRITECKD, the MESS= and MESSLEN= operands are also required.

MESS=
address of the input or output area.

Specified as: a label.

Examples:

```
              IAMIO   MESS=MESSAREA
                  •
                  •
                  •
   MESSAREA   DS   CL50
```

MESSLEN=
the length of the output or input area.

Specified as: either a decimal number of a pointer to a four-byte binary value.

Examples:

```
(1)           IAMIO   MESS=M,MESSLEN=50

(2)           IAMIO   MESS=M,MESSLEN=MLEN
                  •
                  •
                  •
   MLEN         DC   F'50'
   M            DS   CL50
```

Note: Either RPN or ACTADD must be specified if the READ, WRITE, READCKD, or WRITECKD is to a direct access device.

RPN=
the relative page number of the input or output.

Specified as: either a decimal number or a pointer to a four-byte binary value.

Examples:

```
(1)           IAMIO   RPN=3

(2)           IAMIO   RPN=RPNCON
                  •
                  •
   RPNCON       DC   F'3'
```

ACTADD=
The actual disk address (bbcchhr) of the input or output area.

Specified as: either a decimal number or a pointer to a seven-byte actual address.

Example:

```
(1)         IAMIO   ACTADD=3

(2)         IAMIO   ACTADD=BBCCHHR
                      •
                      •
                      •
   BBCCHHR    DC    X'0000003'
```

Additional Required Operands: If ACTION=SIOONLY the CCW= operand is required:

CCW=
the address of a CCW list provided by the calling utility.

Specified as: a label.

Example:

```
            IAMIO   ACTION=SIOONLY,CCW=CCWLIST
                      •
                      •
                      •
   CCWLIST   CCW    X'01',WRITEAR,X'00',6
   WRITEAR    DC    CL6'A'
```

Other Operands:

CSW=
a CSW mask specified by the user. If at any point in its processing of the current IAMIO request, USS should encounter a synchronous interrupt with any of the same bits on that were on in the CSW mask, USS will immediately return to the caller without attempting further error recovery.

Specified as: a two-byte mask.

Example: IAMIO CSW=8020

(If USS encounters either an attention or a unit check, the caller wishes USS to do no further processing. USS will, however, perform a sense operation to clear any unit check. The CSW and sense data will be returned to the caller in IMOOUT.)

SENSE=
a sense mask. If at any point in its processing of the current IAMIO request, USS should encounter sense data with any of the same bits on that were on in the caller's sense mask, USS will immediately return to the caller without attempting further error recovery.

Specified as: a sense mask less than or equal to 128 bytes in length. (The IAMIO macro will pad the user's sense mask on the right with zeroes so that it is 128 bytes long.)

Example: IAMIO SENSE=40

(The caller itself wishes to process any intervention required.)

EOF=
the address of an end-of-file routine. (If EOF= is omitted, USS will treat it as either an I/O error or as an intervention required, whichever is appropriate for the device. When using the MF=E format of IAMIO it is important to remember that specifying EOF causes a permanent change to the associated IAMIO MF=L parameter list. Hence, if you next want to ignore a previously defined end-of-file routine, specify EOF=*IOERR on all later I/O operations.

Specified as: a label.

Example:

```
(a)         IAMIO   EOF=EOFROUT
                      •
                      •
                      •
(b)         IAMIO   EOF=*IOERR
                      •
                      •
                      •
   EOFROUT   STM    REG14,REG12,REGSAVE
```

At (a), if EOF is encountered on the I/O operation the caller wishes IAMIO to branch to EOFROUT. At (b) the caller wishes to specify no end-of-file routine.

VERIFY=
indicates whether or not to verify the device type.

Specified as: Y or N (Y may be specified only if ACTION=AMIHERE is specified.) VERIFY=Y implies that a device type is specified (within this macro) and the caller wants to know if the actual device (at the PDA specified) is of the same device type for terminals and direct access devices, or of the same device class for printers, tapes, and card readers. For printers, tapes and card readers, IAMIO returns to the caller the device type specified in DEV=xxxx even though the actual device at the specified PDA may be a different device type of the same device class; for example, if DEV=2400 is specified and the actual device at the PDA specified is a 3420, the device is assumed to be a 2400 and is verified to the caller as a 2400. However, if DEV=3350 is specified and a 3420 or a 2314 is at the specified PDA, the device is verified as incorrect.

CAUTION: If a device is not ready when device type is ascertained, incorrect results may occur.

ALARM=
specifies whether or not the alarm will be sounded on the operator's console.

Specified as: Y or N.

Default: N

PRINTER=
for the 3270 only, whether or not to use the hard copy printer as well as the display for input and output.

Specified as: Y or N

Default: N

HEADER=
specifies what action is taken with respect to the header.

Specified as:
STOP -- stop printing a header
ADD -- add the input line to the header
CONTINUE -- no change

Default: CONTINUE

Notes: 1) A header is maintained, if desired, on all terminals and printers. HEADER=ADD will cause USS to begin a new page or frame with the specified header line. All subsequent pages or frames will begin with this header. HEADER=STOP will cause USS to begin a new page or frame without a header.

2) If the constant, "PAGEbbb" is found in the header, USS will maintain a running page count beginning with PAGE 1.

FOLD=
specifies whether or not to translate lower case alphabetic data into upper case.

Specified as: Y or N

Default: Y

DENSITY=
specifies the tape density.

Specified as: a three-byte value or a pointer to such a value.

Examples:

```
(1)         IAMIO   DENSITY=9D3 (9-track tape, 1600 bpi)

(2)         IAMIO   DENSITY=D
                      •
                      •
                      •
          D         DC  C'9D3'
```

Note: While density is an optional parameter, its use is advised for all tape operations. If its use is incorrect for the tape drive (for example, reading a 9-track 2400 tape drive), IAMIO will disregard it. If not specified, IAMIO assumes the tape is 9-track.

ALTR1=
ALTR14=
ALTR15=
specifies alternate registers for IAMIO to use instead of Registers 1, 14 and 15 respectively. IAMIO uses these registers exclusively unless alternates are specified.

Programming Notes: IAMIO builds a control block, CHAIMO, which passes information to IAM. IAM returns information in IMOOUT, a section of CHAIMO. IMOWHA is the area within CHAIMO in which IAM returns descriptive information about the device on ACTION=AMIHERE, VERIFY=Y or ACTION=AMIHERE,DEV=?

Return Codes: IAMIO returns with a return code, as follows, in register 15:

Code		Meaning
0	-	I/O went OK.
4	-	Unrecoverable I/O error.
8	-	The input message was truncated.
12	-	The input message was padded.

From the "AMIHERE" function only:

Code		Meaning
16	-	Not ready
20	-	Offline

From 3215, 3016, and 3270 I/O operations:

Code		Meaning
24	-	Attention

Note: No explanatory messages are written to the operator console on IAMIO error returns. If IAMDEAD is called for error information appended (see the IAMDEAD macro), IAMDEAD will print the PDA, CSW, and any other necessary information concerning the last I/O operation.

Debugging Aids

PSASAV0 contains the caller's register contents upon entry into USS.

IAMMAST contains information on the most recent I/O operation. If the last I/O operation was the printing of an IAMDEAD error message, MASTBACK contains the previous I/O operation information.

IOINLOG contains an entry for the last 256 I/O operations.

IOINLOGW contains the most recent entry in IOINLOG.

SIOLOGE1 contains an entry for the last 256 START I/Os.

SIOLOGWE contains the most recent entry in SIOLOGE1.

IAMMAST, IOINLOG, and SIOLOGE1 are found in the module IAMIO. PSASAV0 is found in the DSECT CHAPS1.

IAMOPER -- I/O to Operator

The IAMOPER macro is used to perform reads and writes to the operator's console or SYSIN/SYSOUT.

```
┌--------┬---------┬-------------------------------------------------------┐
|Name    |Operation|Operand                                                |
|--------|---------|-------------------------------------------------------|
|[symbol]|IAMOPER  |OP=,[MSGIN=,LENIN=,MSGOUT=,LENOUT=,SOURCE=,ATTN=,      |
|        |         |PFKEY=,FOLD=,MF=]                                      |
|        |         |                                                       |
└--------┴---------┴-------------------------------------------------------┘
```

OP=
the function requested of IAMOPER.

Specified as: READ, WRITE, PRMPT, or POLL.
READ - read operator console or SYSIN, as directed in SOURCE=
WRITE - write to operator console or SYSOUT, as directed in SOURCE=
PRMPT - write with response (WRITE followed by READ)
POLL - poll for attention at operator console

MSGIN=
the address of the input area.

Specified as: a label.

LENIN=
the address of a fullword containing the length of the input area.

Specified as: a label.

MSGOUT=
the address of the output area.

Specified as: a label.

LENOUT=
the address of a fullword containing the length of the output area.

Specified as: a label.

SOURCE=
specifies whether to use the operator's console or SYSIN/SYSOUT.

Specified as:
SYSIO for SYSIN/SYSOUT
SYSOPR for operator's console

Default: SYSIO

Note: SYSIN/SYSOUT are initialized to the operator's console and remain so unless explicitly changed by the SYSIN and/or SYSOUT commands.

ATTN=
specifies whether to return attentions to the caller or to clear them.

Specified as:

Y -- the caller will be notified of operator console attentions via a return code.
N -- IAMOPER will clear attentions by asking the user if he wishes to continue or to abort the current utility.

Default: N

PFKEY=
specifies whether or not to return PF key interrupts to the caller as attentions.

Specified as:

Y -- the caller will be notified of PF key interrupts via a return code.
N -- IAMOPER will clear PF key interrupts by asking the user if he wishes to continue or not.

FOLD=
specifies whether or not to transfer lower case alphabetic input to upper case.

Specified as: Y or N

Default: Y

IAMDEAD -- System Error Message

The IAMDEAD macro puts out a message even though USS itself may have been "broken". (See EXECUTION.)

Name	Operation	Operand
[symbol]	IAMDEAD	

Note: There are no operands.

Initialization: IAMDEAD calls a routine that requires Registers 0 and 1 be pre-set as follows:

Register 1 -- points to the desired message; the message is preceded by one byte containing the message length.

Register 0 -- low order byte = C'U' -- append CSW and SENSE, etc., information about the status of the I/O operation which immediately preceded the call to IAMDEAD.

low order byte = C'E' -- append 'error detected at xxxxxx', where:

xxxxxx = the address contained in the high-order 24 bits of Register 0.

low order byte = C'B' -- append both the C'U' and the C'E' messages.

low order byte = C'S' -- if writing the message would require a SIGP to the CPU address in the high-order halfword of Register 0, simply return to the caller

without attempting to write the message.

Output:

Register 15=0 The message was written.
Register 15=4 The message was not written. (This return code is possible only if the low-order byte of input Register 0 was C'S'.)

Execution: If the operator's console is a display tube, it would be preferable, if possible, to append the new message to the display image being maintained by USS to utilize the USS scrolling logic as well as the USS logic for requiring the user to acknowledge having read a screen full of writes. This is true not only because the technique should yield a more pleasing display but because losing control of the scrolled image may deprive the user of the ability to understand IAMDEAD's message in context with the preceding messages and, because of this loss of context, to debug even the most obvious procedural errors.

For this reason, IAMDEAD's first attempt is to use USS itself to write the message. If this succeeds, it returns to its caller, allowing the caller to continue processing. If this does not succeed, IAMDEAD will abort, writing the message with its own CCWs.

SECTION 2: MACROS FOR ISOLATING COMMAND PARAMETERS

The Utility Support System contains subroutines that isolate the parameters of a command. The subroutines are accessible via the following macros:

IAMGETC -- reads the next command and returns to the caller a pointer to the command operation.

IAMGETP -- returns to the caller a pointer to the first (or next) parameter of a command. It gives the caller a means of defining and handling a positional parameter sequence for the commands. Users of this command must use IAMGETC.

IAMGETS -- returns to the caller a pointer to the first (or next) subparameter. It gives the caller a means of isolating a subscript parameter; that is, a parameter of the form '(subparm1,subparm2,...)'. The user need not check first to determine that a parameter is in subparameter form; if used where subparameters do not exist, it acts like IAMGETP. IAMGETS handles the last parameter that was outputted by IAMGETP. Users must first code IAMGETC to read the command and IAMGETP to isolate this parameter.

For all of these three macros, all pointers returned to the caller point to operations and operands that are in upper case letters.

IAMGETC -- Get a Command

The IAMGETC macro causes a pointer to the next command to be returned to the caller.

Name	Operation	Operand
[symbol]	IAMGETC	input

input
specifies whether to read the command from SYSIN or from the operator's console.

Specified as: *SYSIN or *SYSOPR

Default: *SYSIN

Examples:

(1) IAMGETC *SYSIN (read a command from SYSIN)

(2) IAMGETC *SYSOPR (read a command from the operator's console)

Return Data: Register 1 contains the address of the command operation. The byte preceding the command operation contains the length of the command operation. Diagramatically:

```
                          AL1(L'COMMAND OPERATION')
Register 1  --------->    C'COMMAND OPERATION'
```

Register 15=0 means there is a command

=4 means an error has been detected which requires prompting the user for a fresh command.

IAMGETP -- Get a Parameter

The IAMGETP macro returns to the caller a pointer to the first (or next) parameter in the operand portion of a command.

Name	Operation	Operand
[symbol]	IAMGETP	

Note: There are no operands.

Initialization: Register 1 must point to a string of the form 'keyword=parameter'. Preceding this string must be one byte containing the length of the keyword including its equal sign. Preceding the length byte must be still another byte containing the length of the parameter.

If a default value is not permitted for a positional or keyword parameter, the length byte of the parameter in the string must be zero. If keywords are required, the string pointed to must be C'*='.

If the command entered by the user omitted a parameter value, IAMGETP will return to the caller the parameter value pointed to by Register 1. If the command entered by the user specified a parameter value, IAMGETP will return to the caller this value instead of the value pointed to by Register 1. Similarly, if the command entered by the user omitted the keyword, IAMGETP will return to the caller the keyword (including the equal sign) pointed to by Register 1.

Form 1: If the user may omit the parameter from the command, the calling program must contain:

```
                          AL1[L'PARAMETER VALUE']
                          AL1[L'KEYWORD=']
Register 1  --------->    C'keyword=parameter'
```

Form 2: If the user must specify the parameter value in the command:

```
                          AL1[0]
                          AL1[L'KEYWORD=']
Register 1  --------->    C'KEYWORD='
```

Form 3: If the user must specify both the keyword (including the equal sign) and the parameter value:

```
                          AL1[0]
                          AL1[2]
Register 1  --------->    C'*='
```

Programming Notes: Calling programs should be coded so that the user can enter command parameters positionally or with keywords, or both. If

a user codes a keyword but in the right position, he should still be able to continue entering parameters positionally thereafter. However, once a keyword is used 'out of position', all subsequent parameter values must be entered with keywords. This can readily be accomplished by using the first two forms above to define a positional parameter sequence, and the third form above after the positional parameter sequence has been broken.

Return Data: Upon return to the caller, IAMGETP loads Register 1 with a pointer to a string having the exact same format as Form 1 under Initialization. The string itself contains values (parameters and keywords) supplied by the user; if the user did not supply either of these values, the string reflects the corresponding values that were supplied to IAMGETP by the calling program.

For all IAMGETPs after the last parameter (value) have been returned to the calling program (by IAMGETP), Register 1 contains a pointer to a string identical to Form 3 under Initialization.

Register 15=0 means the parameter sequence has not been broken (i.e., either the caller's keyword was C'*=' or it was the same keyword now being returned).

Register 15=4 means the positional parameter sequence has just been broken (i.e., the calling program's keyword was something other than C'*=' and a different.

Register 15=8 means an error has been detected which requires prompting for a fresh command.

IAMGETS -- Get a Subparameter

The IAMGETS macro returns to the caller a pointer to the first (or next) subparameter.

Name	Operation	Operand
[symbol]	IAMGETS	

Note: There are no operands. IAMGETS handles the last parameter value returned by IAMGETP.

Return Data: Register 1 points to the subparameter preceded by one byte containing the length of the subparameter. Diagramatically:

```
                        AL1[L'subparameter']
Register 1  --------->  C'subparameter'
```

Register 15=0 means there is a subparameter.

Register 15=4 means there are no more subparameters.

Register 15=8 means an error has been detected that requires prompting for a fresh command.

Programming Notes: IAMGETS may be coded whenever a parameter may be in subparameter form (that is, '(value1, value2, ...)') and the caller does not wish to check to determine that it actually is in subparameter form.

SECTION 3: OTHER MACROS AND COMMANDS

IAMLINK -- Link to a USS Function

The IAMLINK macro must be used when linking to USS functions because there is no capability in USS to resolve EXTRNs. All linkages to USS functions is via a table of adcons (CHAUVT) which is pointed to in the PSA. IAMLINK is an internal macro for the other macros in this manual.

IAMSIGP -- Change CPU

The IAMSIGP macro causes execution to take place in another CPU.

Name	Operation	Operands
[symbol]	IAMSIGP	

Note: There are no operands.

IAMSIGP puts the existing CPU in a wait state and starts execution in the new CPU. It also initializes the PSA in the new CPU if necessary. After executing IAMSIGP, the calling program continues execution in the new CPU beginning with the next sequential instruction in the calling program.

Initialization:

Register 0 -- normally contains all zeros. If it does not, it must be loaded with the address of the CPU status table.

Register 1, byte 0 -- If equal to X'00' no explanatory messages will be written to the operator's console if IAMSIGP has failed to change to a new CPU.

If equal to X'01', explanatory messages will be written.

byte 1 -- If bit 0=0, bits 1-7 of the PSW in the new CPU will be the same as in the old CPU. If bit 0=1, bits 1-7 (of this byte) replace bits 1-7 of the PSW in the new CPU.

bytes 2&3 -- the CPU address of the new CPU.

Return Data:

Register 15=0 the CPU change was successful.

Register 15=4 not successful

IAMTRAN -- Link to Translate Routine

IAMTRAN is used to link to a routine which translates 1 to 4 unpacked hexadecimal characters to right-adjusted halfword hexadecimal numbers.

Name	Operation	Operand
[symbol]	IAMTRAN	

Note: There are no operands.

Initialization: Register 1 points to the character string to be translated; the byte immediately preceding this string contains the length of the string. Diagramatically:

```
                        AL1(L'hexadecimal characters')
Register 1  --------->  C'hexadecimal characters'
```

Return Data:

Register 1 -- contains translated hexadecimal number right-adjusted in the register; for example, if input is C'12B', output will be X'0000012B'.

Register 15=0 -- Translation successful

Register 15=4 -- Input length less than 1 or greater than 4 characters.

Register 15=8 -- Inputted character not hexadecimal.

DSECTS

CHADIO -- the control block passed to IAMDAIO to perform disk I/O.

CHAIMO -- the control block which the IAMIO macro passes to USS to perform I/O.

CHAPS1 -- the portion of the PSA used in USS.

CHAUVT -- the utility vector table containing the addresses of all routines and control blocks referenced in USS.

CHAWHA -- the control block returned by IAMIO on ACTION=AMIHERE, VERIFY=Y requests. It is contained within CHAIMO at IMOWHA.

Debugging Aids

PSAUTA -- the address of the last utility loaded by USS.

PSALOAD -- the address of USS; that is, the address of the CEIAM module.

SECTION 4: UTILITY SUPPORT SYSTEM COMMANDS

There are 2 commands contained within USS itself -- SYSIN and SYSOUT. All other commands invoke a specific utility; they are:

QS -- invoke TSS quick startup

LS -- invoke TSS long startup

DASDI -- invoke direct access storage device initialization.

DUMP -- invoke VAM2 disk dump/restore

RESTORE -- same as DUMP above

VAM2 -- invoke VAM2 utility

The USS prompt is 'ENTER REQUEST'. The individual utilities issue their own prompt messages after they are invoked.

SYSIN -- Define Input Device

SYSIN defines the input device for any USS function or utility.

Operation	Operand
SYSIN	PDA=[,DEV=]

PDA=
physical device address.

Specified as: one to four hexadecimal characters.

DEV=
device type.

Specified as: 1052, 3215, 3066, 3270, 2400, 3420, 1402, or 2540.

Default: USS will determine the device type in the case of terminals, or the device class for all other devices.

Programming Note: Errors encountered at any point in USS operation will cause the input device to revert to the operator console.

SYSOUT -- Define Output Device

SYSOUT defines the output device for any USS function.

Operation	Operand
SYSOUT	PDA=[,DEV=]

PDA=
physical device address.

Specified as: one to four hexadecimal characters.

DEV=
device type.

Specified as: 1052, 3215, 3066, 3270, 2400, 3420, 1403, 140A, 140B, 3211, 321A, or 321B.

Default: USS will determine the device type (terminals and direct access) or the device class (tapes, printers, card reader/punch).

Programming Note: Errors encountered at any point during USS operation will cause the output device to revert to the operator console.

SECTION 5: TSS UTILITIES SUPPLIED BY IBM

INTRODUCTION

IBM supplies 4 utilities which interface with USS -- Dump/Restore, DASDI, VAM2UT (VAM2 Utility), and STARTUP. The first two provide commonly used system programmer functions necessary for creating and copying TSS volumes. VAM2UT provides facilities which enable the systems programmer to debug and alter data on TSS DASD volumes. STARTUP performs the startup/initialization function of TSS and its interface is intended to be with an operator.

DASDI -- Direct Access Storage Device Initialization

DASDI provides the capability to format direct access devices so that they may be used by TSS.

To invoke this utility issue the DASDI command with or without operands. DASDI will prompt the user with a list of additional command operations; a list of these command operations follows:

DADEF	--	describes the pack to be formatted
VLD	--	describes the volume label
PATD	--	defines the location of the PAT
END	--	begins pack formatting

Any operand may be issued on any command operation (except the END command) including the DASDI command. The various DASDI command operations differ only that they have different positional parameter sequences.

OPERANDS

To find the positional parameter sequence of any command operation in the following table, first find the command operation in question in the left-hand column. The adjacent keyword in the middle column is the first positional parameter for that command operation. The succeeding parameters continue in the order they are presented. After the last keyword in the table is reached, return to the first keyword in the table and continue until the command operation with which you started is reached. Note that the positional parameter sequence for DASDI is the same as that for DADEF.

Operation	Keyword	Definition
DASDI or DADEF	TOADDR	The physical device address of the volume to be formatted. There is no default.
	TODEV	The device-type of the volume. (Specified as: 2305, 3330, 333B or 3350.) Default: DASDI will assume TOADDR is correct and will determine the device-type.
	FORMTYPE	The format of the volume after it has been DASDI'ed. Specified as: VAM2. Default: VAM2.

Operation	Keyword	Definition
	IPL	Whether to produce an IPL-able volume or not. Specified as: YES or NO. Default: NO.
	VOLID	The present volid of the pack to be DASDI'ed. If not specified, the volid will not be checked.
	BYPASS	Whether to perform surface analysis on the pack. Specified as: YES or NO. Default: NO. (Note: Surface analysis will be performed only on a 2305.)
	VOLTYPE	The type of volume desired as the result of the DASDI. Specified as: PRIVATE, PUBLIC, or PAGING. Default: PRIVATE, unless PUBVOLNO is specified. If PUBVOLNO is specified, the desired VOLTYPE is assumed to be PUBLIC.
	FLAGTEST	Whether to mark those tracks already flagged down in record 0 as down. Specified as: YES or NO. Default: NO.
	PUBVOLNO	The public volume number of the pack after DASDI. Default: If VOLTYPE=PUBLIC, PUBVOLNO=0; otherwise, PUBVOLNO is not used.
VLD	NEWVOLID	The desired volid of the pack after DASDI. Default: The volid is left unchanged.
	VOLPASS	The desired security number of the pack after DASDI. Specified as: 0 or 1. Default: 0.
	OWNERID	The desired ownerid of the pack after DASDI. Default: ownerid is left blank.
PATD	STRTADR	The desired RPN of the first pat page. Default: 16.
	PGINGRPN	The RPN of the first and last pages of the desired paging band. For example, PGINGRPN=[5,9] states that a paging band is desired at RPN 5 through 9, including both 5 and 9. Default: no paging band.

Example 1: Format a pack on 654; default all parameters.

```
System:    ENTER REQUEST
User:      DASDI 654
System:    CEBDI002:  ENTER DADEF, VLD, PATD, OR END COMMAND
User:      END
```

Example 2: Format a pack on 654; change the VOLID to TSS001; start the PAT at RPN 100.

```
System:    ENTER REQUEST
User:      DASDI 654,NEWVOLID=TSS001,STRTADR=100
System:    CEBDI002:  ENTER DADEF, VLD, PATD, OR END COMMAND
User:      END
```

Example 3: Same as example 2.

```
System:    ENTER REQUEST
User:      DASDI 654
System:    CEBDI002:  ENTER DADEF, VLD, PATD OR END COMMAND
User:      VLD TSS001
System:    CEBDI002:  ENTER DADEF, PATD, OR END COMMAND
User:      PATD 100
System:    CEBDI002:  ENTER DADEF OR END COMMAND
User:      END
```

Note: If your installation has a card deck for an old version of DASDI and you do not desire the additional operand capability in this version, the deck is usable as follows:

(1) Issue: SYSIN (card reader PDA)

(2) Place a DASDI command (with no operands) card in front of your card deck.

(3) Read the deck. (If your deck has an IPLTEXT command which is not usable with this new version of DASDI, it will be ignored.)

DUMP/RESTORE

Dump/Restore provides a means by which a VAM2 disk may be dumped or restored. The disk may be copied either to another disk of the same or of greater capacity or to a tape. The tape may later be restored to a disk of the same or of greater capacity. Tapes generated by the time-shared DMPRST command may be restored with the stand-alone dump/restore utility and labeled tapes generated by the stand-alone utility may be restored with the time-shared DMPRST command.

Operation	Operands
DUMP or RESTORE	FROMADDR=,TOADDR=,[FROMDEV=,TODEV=,FORMTYPE=,VOLID=, MODE=,WRITCHEK=,LABEL=,IPL=,NEWVOLID=,PATBLD=,]

The parameters shown above may be issued either as positional parameters (in the sequence shown) or with keywords.

After the DUMP or RESTORE command, Dump/Restore will prompt with the message: "ENTER END OR VDRL COMMAND". The user must respond to this by entering "END".

FROMADDR=
the physical address of the 'from' device.

Specified as: one to four hexadecimal characters.

TOADDR=
the physical address of the 'to' device.

Specified as: one to four hexadecimal characters.

FROMDEV=
the device type of the 'from' device.

Specified as: 2311, 2314, 3330, 333B, 3350, 2400, or 3420.

Default: Dump/Restore will determine device type (terminals and direct access) or class (printers, tapes, reader/punch).

TODEV=
the device type of the 'to' device; must have the same or greater capacity than the 'from' device.

Specified as: 2311, 2314, 3330, 333B, 3350, 2400, or 3420.

Default: Dump/Restore will determine device type (terminals and direct access) or class (printers, tapes, reader/punch).

FORMTYPE=
the format of the pack.

Specified as: VAM2

Default: VAM2

VOLID=
the present VOLID of the 'to' pack.

Specified as: one to six alphanumeric characters.

Default: VOLID will not be checked.

MODE=
the density of the tape drive.

Specified as: 800, 1600, or 6250

Default: 1600

WRITCHEK=
specifies whether or not to perform a write check on the 'to' device.

Specified as: YES or NO

Default: YES

LABEL=
specifies whether or not to override the volume label of the 'to' device with the volume label of the 'from' device.

Specified as:
UPDATE -- override
RETAIN -- keep the 'to' device label

Default: UPDATE

IPL=
specifies which IPL records to retain.

Specified as:
UPDATE -- use IPL records on the 'from' device
RETAIN -- use IPL records on the 'to' device

Default: UPDATE

NEWVOLID=
the desired VOLID of the 'to' device.

Specified as: one to six alphanumeric characters.

Default: the VOLID on the 'from' device will be used 'as is'.

PATBLD=
specifies whether or not to rebuild the PAT.

Specified as: YES or NO

Default: NO

Example 1: Dump 654 to 653

```
System:    ENTER REQUEST
User:      DUMP 654,653
System:    CEBDR001:  ENTER END OR VDRL COMMAND
User:      END
```

Example 2: Same function as Example 1

```
System:    ENTER REQUEST
User:      DUMP FROMADDR=654,TOADDR=653
System:    CEBDR001:  ENTER END OR VDRL COMMAND
User:      END
```

STARTUP

The interface between Startup and the operator is documented in IBM TSS System Generation and Maintenance, GC28-2010.

VAM2UT

VAM2UT provides a capability to examine and modify the content of TSS DASD storage volumes.

I/O OPERATION

All I/O is performed through USS Independent Access Method (IAM) I/O services.

Input from the USS operator terminal is accomplished by a call to the IAM operator-communication function.

Output to the USS operator terminal is also accomplished by a call to the IAM operator-communication function.

Operations on an assigned card reader or line printer are accomplished by a direct call to IAM for the device.

DASD I/O is accomplished by an IAM call for the specified device.

Command input, other than from the USS operator terminal, must be card images. Line continuation/concatenation is not supported.

COMMANDS

The following command operations invoke VAM2UT environmental functions:

```
SET     --  assign dump output device (UTLPRT)
CALL    --  assign command input device (UTLIN)
RUN     --  assign command input device (USSOPR)
```

The following command operations invoke VAM2UT control functions:

```
SET     --  specify DASD volume or VAM2 data set for subsequent
            utility functions
```

The following command operations invoke VAM2UT utility functions:

```
DISPLAY -- display specified data on operator terminal
PATCH   --  alter specified data, display, confirm, change and
            write to DASD
DUMP    --  display specified data on dump output device
```

COMMAND OPERANDS

The following specify VAM2UT environmental operands:

```
$DOUT   --  dump output device (UTLPRT)
```

The following specify VAM2UT control operands:

```
$DISK   --  DASD volume to be used by subsequent utility functions
$DS     --  VAM2 data set to be referenced by subsequent utility
            functions
```

The following qualify VAM2UT data operands:

```
$VL     --  DASD 'VOL1' label
$PRELUDE--  TSS 'PRELUDE' record
$VAM    --  page of VAM2 volume
$PAT    --  VAM2 volume page allocation table
$DSCB   --  VAM2 data set control block
$DS     --  VAM2 data set
$POD    --  VAM2 data set partitioned organization directory
$PMD    --  VAM2 data set member program module dictionary
$RM     --  irrelevant prefix to name of object module, control
            section, or entry point
$VM     --  irrelevant prefix to name of object module, control
            section, or entry point
```

Environmental Functions:

Operation	Operands
SET	$DOUT=X'cuu'
CALL	X'cuu'
RUN	(no operands; equivalent to CALL USSOPR)

Control Functions:

Operation	Operands
SET	$DISK=X'cuu'
SET	$DS=C'dsname'

Utility Functions:

Name	Operation	Operands
[symbol]	DISPLAY	{$VL\|$PRELUDE|$VAM([X'cuu'],rpn)| $PMD.cname\|[$VM.|$RM.]cename} .([off][,[len][,typ]])
	DISPLAY	{$VL\|$PRELUDE|$PMD.cname}
	DUMP	{$VL\|$PRELUDE|$VAM([X'cuu'],rpn)| $PAT\|$DSCB[.*ALL|.*E]|$DS| $POD\|$PMD.cname|[$VM.\|$RM.]cname}
	PATCH	{$VL\|$PRELUDE|$VAM([X'cuu'],rpn)| $PMD.cname\|[$VM.|$RM.]cename} .([off],len)={X'patch'|C'patch'}
	PATCH	$PMD.cname

cuu
: physical device address.

  Specified as: three hexadecimal digits, one for channel address and two for the unit address.

rpn
: VAM2 volume relative page number.

  Specified as: a decimal or hexadecimal number.

dsname
fully qualified VAM2 data set name.

Specified as: a string of 1 to 44 characters.

cname
control section name

Specified as: a string of 1 to 8 characters.

cename
control section name or entry point name.

Specified as: a string of 1 to 8 characters.

off
the byte offset from the beginning of the specified data operand to the field to be operated on.

Specified as: a decimal or hexadecimal number.

len
the byte length of the field to be operated on.

Specified as: a decimal or hexadecimal number.

typ
data type

Specified as: C for character; X for hexadecimal

patch
the data to overlay the specified field.

Specified as: a string

Example 1: to dump the label, IPL prelude, PAT, and format-E DSCBs of a volume:

```
SET  $DOUT=X'00B'          assign a printer
SET  $DISK=X'624'          assign a disk drive
DUMP $VL                   dumps CKD
DUMP $PRELUDE              dumps CKD
DUMP $PAT                  dumps entire PAT, page by page
DUMD $DSCB,*E              dumps each format-E DSCB
SET  $DISK                 unassign the disk
```

Example 2: to look at a volume label and then change the volume ID:

```
SET  $DISK=X'314'          assign a disk drive
DISPLAY $VL                display CKD
PATCH $VL.(,6)=C'NEWID0'   only Data area is patchable
DISPLAY $VL.(,6)           look at it to make sure its OK
```

Example 3: to dump the PAT of a volume and then change 1 byte of the PAT:

```
DUMP $PAT                  same volume as before; go look at printer
PATCH $VAM(,X'7D4').(X'3C5',1)=X'C0'
DISPLAY $VAM(,X'7D4').(X'3C0',8)
```

GC28-2038-5

IBM Time Sharing System Independent Utilities Printed in U.S.A. GC28-2038-5

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

IBM Time Sharing System
Independent Utilities
Order No. GC28-2038-5

READER'S COMMENT FORM

This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM shall have the nonexclusive right, in its discretion, to use and distribute all submitted information, in any form, for any and all purposes, without obligation of any kind to the submitter. Your interest is appreciated.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name and mailing address:

_____

_____

_____

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

What is your occupation? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

GC28-2038-5

Reader's Comment Form

Fold and tape | **Please Do Not Staple** | Fold and tape

First Class
Permit 40
Armonk
New York

Business Reply Mail
No postage stamp necessary if mailed in the U.S.A.

Postage will be paid by:

International Business Machines Corporation
Time Sharing System – Dept. 80M
1133 Westchester Avenue
White Plains, New York 10604

Fold and tape | **Please Do Not Staple** | Fold and tape

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

Cut or Fold Along Line

IBM Time Sharing System Independent Utilities Printed in U.S.A. GC28-2038-5