IBM Field Engineering Education
Supplementary Course Material

OS/HASP
Volume 2

PREFACE

This document is intended for the use of IBM FE Programming System Representatives enrolled in course 10191.

PRELIMINARY EDITION (May 1971)
This publication has been printed in a preliminary format so that it would be available to the intended users in time for training on this course. This preliminary manual may contain typographical errors that would normally be corrected before publication. This edition is not eligible for suggestion awards, however, your comments will be appreciated.

Address any comments concerning the contents of this publication to: IBM, Field Engineering Education Media Development Center, Dept 927, Rochester, Minnesota 55901.

H A S P

Introduction

The information contained in Volume 1 and 2 of the HASP System Supplementary Course Material was originally distributed as a one-volume document.

Volume 1 contains pages 1 through 590, Section 1 through 8.
Volume 2 contains pages 1 through 594, Section 9 through 12.

A Contents has been included in each volume for your convenience.

9.0 HASP EXECUTOR SERVICES

The HASP Control Service Programs provide a comprehensive set of services which aid the HASP Processors in performing their respective tasks in an efficient manner without burdening the processor programmer down with endless detail. These services are requested by the processor through the use of HASP macro instructions. The services are subdivided in this publication, as follows:

- Buffer Services, which provide for the acquisition and release of HASP buffers.
- Unit Services, which provide for the acquisition and release of HASP Input/Output units.
- Job Queue Services, which provide the processors with an interface with the HASP Job Queue.
- Direct Access Space Services, which provide for the allocation and de-allocation of HASP direct-access storage space.
- Input/Output Services, which provide all communication with the Operating System Input/Output Supervisor.
- Time Services, which provide for the setting and interrogation of the interval timer.
- Overlay Services, which provide the capability to define and utilize sections of HASP that may optionally be made resident on direct-access storage and fetched into a dynamic area within HASP whenever required.
- Synchronization Services, which provide synchronization and communication between HASP processors, the HASP dispatcher, and the Operating System.
- Debug Services, which provide facilities for aid in debugging HASP.
- Error Services, which provide a uniform way of processing detected errors.
- Coding Aid Services, which provide the HASP programmer with coding aids not usually available in the Operating System, but useful in coding HASP routines.

Some of the above services are provided by "in-line" code expansion wherever the macro instruction is used. The remainder of the services are provided by routines which are integral parts of the Control Service Programs. For more information about these routines refer to Section 5. These routines are "linked to" by code generated wherever the macro instruction is used.

At execution time, the macro-expansion passes information to the control program routine to specify the exact nature of the service to be performed. This information is broken down into parameters and, in general, is passed to the routine through general purpose registers called parameter registers.

The macro-expansion can contain load instructions (LA,L,LH,etc.) that form parameters in parameter registers, and/or it can contain instructions which load parameter registers from registers loaded by the processor. The processor can also load parameters directly. Registers "R1" and "R0" are generally used as parameter registers.

Each parameter resulting from the expansion of a macro-instruction is either an address or a value.

ADDRESS PARAMETER: An address parameter is a standard 24-bit address. It is always located in the three low-order bytes of a parameter register. The high-order byte in the parameter register should contain all zeros. Any exception to this rule will be stated in the individual macro-instruction description.

An address parameter is always an effective address. The Control Service Programs is never given a 16-bit or 20-bit explicit address of the form D(B) or D(B,X) and then required to form an effective address. When an effective address is to be resolved, it is formed either by the macro-expansion or before the macro-instruction is issued.

VALUE PARAMETER: A value parameter is a field of data other than an address. It is of variable length, and is usually in the low-order bits of a parameter register. The value parameter will always have a binary format. The high-order unused bits in the parameter register should contain all zeros. Any exception to this rule will be stated in the individual macro-instruction description.

Certain value parameters can be placed in a register along with another parameter, which can either be an address or a value parameter. In this case, a value parameter will be in other than the low-order bits. Two or more parameters in the same register are called packed parameters.

OPERANDS: Parameters are specified by operands in the macro-instruction. An address parameter can result from a relocatable expression or, in certain macro-instructions, from an implied or explicit address. A value parameter can result from an absolute expression or a specific character string. Address and value parameters can both be specified by operands written as an absolute expression enclosed in parentheses. This operand form is called register notation. The value of the expression designates a register into which the specified parameter must be loaded by the processor before macro-instruction is issued. The contents of this register are then placed in a parameter register by the macro-expansion.

Types of Macro-Instruction Operands

The processor programmer writes operands in a HASP macro-instruction to specify the exact nature of the service to be performed. Operands are of two types: positional and keyword.

POSITIONAL OPERANDS: A positional operand is written as a string of characters. This character string can be an expression, an implied or explicit address, or some special operand form allowed in a particular macro-instruction.

Positional operands must be written in a specific order. If a positional operand is omitted and another positional operand is written to the right of it, the comma that would normally have preceded the omitted operand must be written. This comma should be written only if followed by a positional operand; it need not be written if it would be followed by a keyword operand or a blank.

In the following examples, EX1 has three positional operands. In EX2, the second of three positional operands is omitted, but must still be delimited by commas. In EX3, the first and third operands are omitted; no comma need be written to the right of the second operand.

```
EX1  $EXAMP   A,B,C

EX2  $EXAMP   A,,C

EX3  $EXAMP   ,B
```

KEYWORD OPERANDS: A keyword operand is written as a keyword immediately followed by an equal sign and an optional value.

A keyword consists of one through seven letters and digits, the first of which must be a letter. It must be written exactly as shown in the macro-instruciton description.

An optional value is written as a character string in the same way as a positional operand.

Keyword operands can be written in any order, but they must be written to the right of any positional operands in the macro-instruction.

In the following examples, EX1 shows two keyword operands. EX2 shows the keyword operands written in a different order and to the right of any positional operands. In EX3, the second and third positional operands are omitted; they need not be delimited by commas, because they are not followed by any positional operands.

```
EX1   $EXAMP    KW1=X,KW2=Y

EX2   $EXAMP    A,B,C,KW2=Y,KW1=X

EX3   $EXAMP    A,KW1=X,KW2=Y
```

REQUIRED AND OPTIONAL OPERANDS: Certain operands are required in a macro-instruction, if the macro-instruction is to make a meaningful request for a HASP executor service. Other operands are optional, and can be omitted. Whether an operand is required or optional is indicated in the macro-instruction descriptions.

9.0.1 BASIC NOTATION USED TO DESCRIBE MACRO-INSTRUCTIONS

HASP macro-instructions are presented in this section by means of macro-instruction descriptions, each of which contains an illustration of the macro-instruction format. This illustration is called a format description. An example of a format description is as follows:

```
[symbol]    $EXAMP    name1-value mnemonic,name2-CODED VALUE,

                      KEYWD1=value mnemonic,KEYWD2=CODED VALUE
```

Operand representations in format descriptions contain the following elements:

- An operand name, which is a single mnemonic word used to refer to the operand. In the case of a keyword operand, the keyword is the name. In the case of a positional operand, the name is merely a reference. In the above format description, name1, name2, KEYWD1, and KEYWD2 are operand names.
- A value mnemonic, which is a mnemonic used to indicate how the operand should be written, if it is not written as a coded value. For example, addr is a value mnemonic that specified that an operand or optional value is to be written as either a relocatable expression or register notation.
- A coded value, which is a character string that is to be written exactly as it is shown. For example, RDR is a coded value.

The format description also specifies when single operands and combinations of operands should be written. This information is indicated by notational elements called metasymbols. For example, in the preceding format description, the brackets around "symbol" indicate that a symbol in this field is optional.

Operand Representation

Positional operands are represented in format descriptions in one of two ways:

- By a three-part structure consisting of an operand name, a hyphen, and a value mnemonic. For example: name1-addr.
- By a three-part structure consisting of an operand name, a hyphen, and a coded value. For example: name1-RDR.

Keyword operands are represented in format descriptions in one of two ways:

- By a three-part structure consisting of a keyword, an equal sign, and a value mnemonic. For example: KEYWD1=addr.
- By a three-part structure consisting of a keyword, an equal sign, and a coded value. For example: KEYWD1=RDR.

The most significant characteristic of an operand representation is whether a value mnemonic or coded value is used; these two cases are discussed below.

Operands with Value Mnemonics

When a keyword operand is represented by:

KEYWORD=value mnemonic

the programmer first writes the keyword and the equal sign, and then a value of one of the forms specified by the value mnemonic.

When a positional operand is represented by:

name-value mnemonic

the programmer writes only a value of one of the forms specified by the value mnemonic. The operand name is merely a means of referring to the operand in the format description; the hyphen simply separates the name from the value mnemonic. Neither is written.

The following general rule applies to the interpretation of operand representations in a format description; anything shown in upper-case letters must be written exactly as shown; anything shown in lower-case letters is to be replaced with a value provided by the programmer. Thus, in the case of a keyword operand, the keyword and equal sign are written as shown, and the value mnemonic is replaced. In the case of a positional operand, the entire representation is replaced.

VALUE MNEMONICS: The value mnemonics listed below specify most of the allowable operand forms that can be written in HASP macro-instructions. Other value mnemonics, which are rarely used, are defined in individual macro-instruction descriptions.

- symbol -- the operand can be written as a symbol.
- relexp -- the operand can be written as a relocatable expression.
- addr -- the operand can be written as (1) a relocatable expression, or (2) register notation designating a register that contains an address in its three low order bytes. The designated register must be one of the registers 2 through 12, unless special register notation is used. (Refer to Section 9.0.2: Special Register Notation.)
- addrx -- the operand can be written as (1) an indexed or nonindexed implied or explicit address, or (2) register notation designating a register that contains an address in its three low-order bytes. An explicit address must be written as in the RX form of an assembler language instruction.
- adval -- the operand can be written as (1) an indexed or nonindexed implied or explicit address, or (2) register notation designating a register that contains a value. An explicit address must be written as in the RX form of an assembler language instruction.
- absexp -- the operand can be written as an absolute expression.
- value -- the operand can be written as (1) an absolute expression, or (2) register notation designating a register that contains a value in its three low-order bytes.
- text -- the operand can be written as a character constant as in a DC data definition instruction. The format description shows explicitly if the character constant is to be enclosed in single quotation marks.
- code -- the operand can be written as one of a large set of coded values; these values are defined in the macro-instruction description.

Coded Value Operands

Some operands are not represented in format descriptions by value mnemonics. Instead, they are represented by one or more upper-case character strings that show exactly how the operand should be written. These character strings are called coded values, and the operands for which they are written are called coded value operands.

A coded value operand results in either a specific value parameter or a specific sequence of executable instructions.

If a positional operand can be written as any one of two or more coded values, all possible coded values are listed and are separated by vertical stroke indicating that only one of the values is to be used.

Metasymbols

Metasymbols are symbols that convey information to the programmer, but are not written by him. They assist in showing the programmer how and when an operand should be written. The metasymbols used in this section are:

- | This is a vertical stroke and means "or". For example, A|B means either the character A or the character B. Alternatives are also indicated by being aligned vertically (as shown in the next paragraph).

- { } These are braces and denote grouping. They are used most often to indicate alternative operands. For example:

  $\{\text{YES}|\text{NO}\}$ or $\left\{\begin{array}{c}\text{YES}\\\text{NO}\end{array}\right\}$

  The two examples above are equivalent; either YES or NO must be written.

- [ ] These are brackets and denote options. Anything enclosed in brackets can either be omitted or written once in the macro instruction. For example:

  $[\underline{\text{YES}}|\text{NO}]$ or $\left[\begin{array}{c}\underline{\text{YES}}\\\text{NO}\end{array}\right]$

  The two examples above are equivalent; YES, or NO, or neither can be written. The underlining indicates that, if neither is written, YES is assumed. Braces used for grouping inside brackets are redundant.

9.0.2 SPECIAL REGISTER NOTATION

If an operand of a HASP macro-instruction is written using register notation, the resulting macro-expansion loads the parameter contained in the designated register into either parameter register "R1" or parameter register "R0".

For example, if an operand is written as (R15), and if the corresponding parameter is to be passed to the control program in register "R1", the macro-expansion could contain the instruction:

```
LR   R1,R15
```

The processor can load parameter registers directly, before the execution of the macro-expansion; this is called preloading. The programmer specifies that preloading will occur by writing an operand as either "(R1)" or "(R0)"; this is called special register notation. This notation is special for two reasons:

- The register notation designation of registers "R1" and "R0" is generally not allowed.

- The designation must be made by the specific four characters "(R1)" or "(R0)", rather than by the general form of an absolute expression enclosed in parentheses. For example, even though the absolute symbol RONE could be equated to R1, "(RONE)" must not be written instead of "(R1)" if special register notation is intended. If this were done, the macro-expression would contain a useless instruction:

  ```
  LR   R1,RONE.
  ```

The format description shows whether special register notation can be used, and for which operands. This is demonstrated by the following example:

```
                    {abc-addrx}   {def-addrx}
[symbol]   $EXAMP   {(R1)     } , {(R0)     }
```

Both operands can be written in the addrx form, and therefore can be written using register notation. Ordinary register notation indicates that the parameter register should be loaded from the designated register by the macro-expansion. The format description also shows that the abc operand can be written as "(R1)", and the def operand can be written as "(R0)". If either of these special register notations is used, the processor must have loaded the designated parameter register before the execution of the macro-instruction.

9.0.3 REGISTER TRANSPARENCY

In general, the following registers cannot be considered transparent across a HASP macro expansion and the associated link to the Control Service Program:

- LINK
- R14
- R15
- R0
- R1

All other registers will be transparent unless specifically stated in the individual macro-instruction description.

9.1 BUFFER SERVICES

9.1.1 $GETBUF - Acquire a HASP Buffer from the HASP Buffer Pool or RJE Buffer from the RJE Buffer Pool

The $GETBUF macro-instruction obtains a buffer from the HASP or RJE buffer pool and returns the address of this buffer in register "R1".

Format Description:

```
[symbol]   $GETBUF   [none-relexp] [,TYPE=TP] [,OLAY=YES]
```

none
specifies a location to which control will be returned if there are no buffers available.

If this operand is omitted, the condition code will be set to reflect the availability of a buffer as follows:
CC=0 - no buffer is available.
CC≠0 - "R1" contains the address of the buffer.

TYPE=TP
specifies that the buffer is to be obtained from the RJE buffer pool rather than the HASP buffer pool.

OLAY=YES
must be specified if the $GETBUF macro-instruction is coded physically within an overlay segment.

9.1.2 $FREEBUF - Return a HASP Buffer to the HASP Buffer Pool or RJE Buffer to the RJE Buffer Pool

The $FREEBUF macro-instruction is used to return a HASP buffer to the HASP buffer pool or RJE buffer to the RJE buffer pool.

Format Description:

```
[symbol]   $FREEBUF   {buffer-addrx}   [,OLAY=YES]
                      {(R1)        }
```

buffer
: specifies either a pointer to a buffer or the address of a buffer to be returned to the buffer pool as follows:

  If "buffer" is written as an address, then it represents the address of a full word which contains the address of the buffer to be returned in its three low order bytes. This word must be located on a full-word boundary in core.

  If "buffer" is written using register notation (either regular or special register notation), then it represents the address of the buffer to be returned.

  If register notation is used, the address must have been loaded into the designated register before the execution of this macro-instruction.

OLAY=YES
: must be specified if the $FREEBUF macro-instruction is coded physically within an overlay segment.

CAUTION: The specified buffer must have been obtained by a $GETBUF macro-instruction. The action of the macro-instruction as well as future $GETBUF and $FREEBUF macro-instructions is unpredictable in other cases.

9.2 UNIT SERVICES

9.2.1 $GETUNIT - Acquire a Unit Device Control Table (DCT)

The $GETUNIT macro-instruction obtains a Device Control Table (DCT) for a specified type of unit, and returns the address of this DCT in register "R1".

Format Description:

```
[symbol]   $GETUNIT   type-code [,none-relexp] [,OLAY=YES]
```

type
: specifies the type of unit for which a DCT is to be obtained. The values for this operand and their meanings are:
  - DA - Direct Access DCT
  - LNE - Line DCT
  - RDR - Card Reader DCT
  - TPE - Input Tape DCT
  - RJR - Remote Reader DCT
  - INR - Internal Reader DCT
  - PRT - Printer DCT
  - RPR - Remote Printer DCT
  - PUN - Punch DCT
  - RPU - Remote Punch DCT
  - CON - Console DCT

none
: specifies a location to which control will be returned if there are no available Device Control Tables for the specified device. If this operand is omitted, the condition code will be set to reflect the availability of a DCT as follows:
  - CC=0 - no DCT is available.
  - CC≠0 - "R1" contains the address of a DCT of the specified type.

OLAY=YES
: must be specified if the $GETUNIT macro-instruction is coded physically within an overlay segment.

9.2.2 $FREUNIT - Release a Unit Device Control Table (DCT)

The $FREUNIT macro-instruction is used to release a Device Control Table (DCT).

Format Description:

```
[symbol]    $FREUNIT    {dct-addrx}    [,OLAY=YES]
                        {(R1)     }
```

dct

specifies either a pointer to a DCT or the address of a DCT to be released as follows:

If "dct" is written as an address, then it represents the address of a full word which contains the address of the DCT to be released in its three low order bytes. This word must be located on a full-word boundary in core.

If "dct" is written using register notation (either regular or special register notation), then it represents the address of the DCT to be released.

If register notation is used, the address must have been loaded into the designated register before the execution of this macro-instruction.

OLAY=YES

must be specified if the $FREUNIT macro-instruction is coded physically within an overlay segment.

CAUTION: The specified DCT must have been obtained by a $GETUNIT macro-instruction. The action of the macro-instruction is unpredictable in other cases.

9.3 JOB QUEUE SERVICES

The HASP Job Queue consists of a chain of Job Queue Elements and can be divided into five logical queues. These five logical queues are represented by the following symbolic names:

Table 9.3.1 - Symbolic Representation of the Logical Job Queues

Symbolic Name	Logical Job Queue
$INPUT	Queue of jobs in input processing
$XEQ	Queue of jobs awaiting O/S Execution phase
$PRINT	Queue of jobs awaiting Print phase
$PUNCH	Queue of jobs awaiting Punch phase
$PURGE	Queue of jobs awaiting Purge phase

For more information concerning the formats of the HASP Job Queue Element and the HASP Job Information Table Element, refer to sections 8.6 and 8.7 of this manual.

9.3.1 $QADD - Add Job Queue Element to the HASP Job Queue

The $QADD macro-instruction adds an element to the HASP Job Queue, placing it in the specified logical queue. The address of the associated Job Information Table Entry is returned in register "R0".

Format Description:

```
                                 {element-addrx}     {queue-value}
[symbol]    $QADD                {(R1)         }  ,  {(R0)       }

                                 [,full-relexp] [,OLAY=YES]
```

element

specifies the address of an Element which is to be added to the HASP Job Queue.

If register notation is used, the address must have been loaded into the designated register before the execution of this macro-instruction.

queue

specifies the logical queue in which the Job Queue Element is to be placed. This value must always be one of the values listed in table 9.3.1.

If register notation is used, one of these values must have been loaded into the designated register before the execution of this macro-instruction.

full

specifies a location to which control will be returned if the HASP Job Queue is full.

If this operand is omitted, the condition code will be set to reflect the status of the HASP Job Queue as follows:

- CC=0 - the queue is full and the element cannot be accepted.
- CC≠0 - the Element was successfully added to the queue. "R0" contains the address of the associated JIT Entry.

OLAY=YES

must be specified if the $QADD macro-instruction is coded physically within an overlay segment.

9.3.2 $QGET - Obtain Job Queue Element from the HASP Job Queue

The $QGET macro-instruction obtains a Job Queue Element from the specified logical queue of the HASP Job Queue and returns the address of this element in register "R1". The address of the associated Job Information Table Entry is returned in register "R0".

Format Description:

```
[symbol]    $QGET    {queue-value}   [,none-relexp]
                     {(R1)       }
                   [,PRROUTE=YES] [,PUROUTE=YES]
                   [,CLASS=YES] [,FORMS=YES] [,OLAY=YES]
```

queue

specifies the logical queue from which the Job Queue Element is to be obtained. This value must always be one of the values listed in table 9.3.1.

If register notation is used, one of these values must have been loaded into the designated register before the execution of this macro-instruction.

none

specifies a location to which control will be returned if the specified logical queue is empty.

If this operand is omitted, the condition code will be set as follows:

CC=0 - the specified logical queue is empty.
CC≠0 - "R1" contains the address of a Queue Element from the specified logical queue and "R0" contains the address of the associated JIT Entry.

PRROUTE=YES

specifies that bits 0-7 of register "R0" contain a route code which must match the route code (QUEPRTRT) of the Job Queue Element obtained.

PUROUTE=YES

specifies that bits 8-15 of register "R0" contain a route code which must match the route code (QUEPUNRT) of the Job Queue Element obtained.

CLASS=YES

specifies that bits 16-23 of register "R0" contain a class code which must match the class code (QUECLASS) of the Job Queue Element obtained.

FORMS=YES

specifies that bits 16-31 of register "R0" contain a forms type which must match the forms type (QUEFORMS) of the Job Queue Element obtained.

If no job is found which meets all of the requirements specified, and one or more jobs are found which meet all of the requirements except for the forms specification, then the address of the highest priority Job Queue Element which meets all of the requirements except for the forms specification is returned in register "R0". If no job is found in the specified queue which meets the routing and class requirements alone, then register "R0" is zero.

OLAY=YES

must be specified if the $QGET macro-instruction is coded physically within an overlay segment.

9.3.3 $QPUT - Return Job Queue Element to the HASP Job Queue

The $QPUT macro-instruction returns a Job Queue Element to the HASP Job Queue, placing it in the specified logical queue. The address of the associated Job Information Table Entry is returned in register "R0".

Format Description:

```
[symbol]   $QPUT   {element-addrx}  ,  {queue-value}
                   {(R1)         }     {(R0)       }
                   [,OLAY=YES]
```

element
: specifies the address of an Element which is to be returned to the HASP Job Queue.

  If register notation is used, the address must have been loaded into the designated register before the execution of this macro-instruction.

queue
: specifies the logical queue in which the Job Queue Element is to be placed. This value must always be one of the values listed in table 9.3.1.

  If register notation is used, one of these values must have been loaded into the designated register before the execution of this macro-instruction.

OLAY=YES
: must be specified if the $QPUT macro-instruction is coded physically within an overlay segment.

CAUTION: The specified Job Queue Element must have been previously obtained with a $QGET macro-instruction or the action of the $QPUT macro-instruction is unpredictable.

PROGRAMMING NOTE: The $QPUT macro-instruction cannot be used to change the priority of a Job Queue Element. If a change of priority is desired, the $QREM and $QADD macro-instructions must be used.

9.3.4 $QREM - Remove Job Queue Element from the HASP Job Queue

The $QREM macro-instruction removes a specified Job Queue Element from the HASP Job Queue.

Format Description:

```
[symbol]    $QREM  {element-addrx}  [,OLAY=YES]
                   {(R1)         }
```

element
: specifies the address of an Element which is to be removed from the HASP Job Queue.

  If register notation is used, the address must have been loaded into the designated register before the execution of this macro-instruction.

OLAY=YES
: must be specified if the $QREM macro-instruction is coded physically within an overlay segment.

CAUTION: The specified Job Queue Element must have been previously obtained with a $QGET macro-instruction or the action of the $QREM macro-instruction is unpredictable.

9.3.5 $QSIZ - Determine Number of Elements in a Logical Queue

The $QSIZ macro-instruction determines the number of Job Queue Elements in a specified logical queue of the HASP Job Queue and returns this value in register "R1".

Format Description:

```
[symbol]   $QSIZ   {queue-value}   [,none-relexp]
                   {(R1)       }
                   [,PRROUTE=YES] [,PUROUTE=YES]
                   [,CLASS=YES] [,FORMS=YES] [,OLAY=YES]
```

queue

specifies the logical queue which is to be counted. This value must always be one of the values listed in table 9.3.1.

If register notation is used, one of these values must have been loaded into the designated register before the execution of this macro-instruction.

none

specifies a location to which control will be returned if the specified logical queue is empty.

If this operand is omitted, the condition code will be set to reflect the status of the specified logical queue as follows:

CC=0 - the specified queue is empty (R1=0).
CC≠0 - the specified queue contains at least one Job Queue Element (R1 = number of Elements in queue).

PRROUTE=YES

specifies that bits 0-7 of register "R0" contain a route code which must match the route code (QUEPRTRT) of all jobs counted.

PUROUTE=YES

specifies that bits 8-15 of register "R0" contain a route code which must match the route code (QUEPUNRT) of all jobs counted.

CLASS=YES

specifies that bits 16-23 of register "R0" contain a class code which must match the class code (QUECLASS) of all jobs counted.

FORMS=YES
specifies that bits 16-31 of register "R0" contain a forms type which must match the forms type (QUEFORMS) of all jobs counted.

OLAY=YES
must be specified if the $QSIZ macro-instruction is coded physically within an overlay segment.

9.3.6 $QLOC - Locate Job Queue Element for Specific Job

The $QLOC macro-instruction locates the Job Queue Element associated with the job with the specified job number and returns the address of this Element in register "R1". The address of the associated Job Information Table Entry is returned in register "R0".

Format Description:

```
[symbol]    $QLOC    {jobno-adval}  [,none-relexp]
                     {(R1)       }
                 [,OLAY=YES]
```

jobno

specifies the binary job number associated with the job for which the Job Queue Element is being searched.

If an address is used it specifies the address of a half-word that contains the binary job number. This half-word must be located on a half-word boundary.

If register notation is used, the binary job number must have been loaded into the designated register before the execution of this macro-instruction.

none

specifies a location to which control will be returned if the specified job number is not locatable in the HASP Job Queue.

If this operand is omitted, the condition code will be set to reflect the status of register "R1" as follows:

CC=0 - the specified job is not locatable.

CC≠0 - the specified job is locatable and "R1" contains the address of the associated Job Queue Element, and "R0" contains the address of the associated JIT Entry.

OLAY=YES

must be specified if the $QLOC macro-instruction is coded physically within an overlay segment.

9.4 DIRECT ACCESS SPACE SERVICES

9.4.1 $TRACK - Acquire a Direct-Access Track Address

The $TRACK macro-instruction obtains a track address on a HASP committed direct access device and returns this track address in register "R1".

Format Description:

```
[symbol]   $TRACK   [OLAY=YES]
```

OLAY=YES
: must be specified if the $TRACK macro-instruction is coded physically within an overlay segment.

CAUTION: The JCT register must be loaded with the address of a Job Control Table before the execution of this macro-instruction or the action of the macro-instruction will be unpredictable.

9.4.2 $PURGE - Return Direct-Access Space

The $PURGE macro-instruction is used to return the direct-access space which has been allocated for a given job.

Format Description:

```
[symbol]    $PURGE    {allocmap-addrx}   [,OLAY=YES]
                      {(R1)          }
```

allocmap
specifies the address of a track allocation map containing the direct-access space to be returned.

If register notation is used, the address must have been loaded into the designated register before the execution of this macro-instruction.

OLAY=YES
must be specified if the $PURGE macro-instruction is coded physically within an overlay segment.

9.5 INPUT/OUTPUT SERVICES

9.5.1 $EXCP - Execute HASP Channel Program

The $EXCP macro-instruction initiates HASP Input/Output activity.

Format Description:

```
[symbol]    $EXCP    {dct-addrx}   [,OLAY=YES]
                     {(R1)     }
```

dct

specifies either a pointer to a Device Control Table (DCT) or the address of a DCT which represents a device upon which Input/Output activity is to be initiated.

If "dct" is written as an address, then it represents the address of a full word which contains the address of the DCT in its three low-order bytes. This word must be located on a full-word boundary.

If "dct" is written using register notation (either regular or special register notation), then it represents the address of the DCT.

If register notation is used, the address must have been loaded into the designated register before the execution of this macro-instruction.

OLAY=YES

must be specified if the $EXCP macro-instruction is coded physically within an overlay segment.

9.5.2 $EXTP - Initiate Remote Terminal Input/Output Operation

The $EXTP macro-instruction initiates an Input/Output Action or Operation.

Format Description:

```
[symbol]    $EXTP    type-code, {dct-addrx}  ,  [loc-addrx]
                                {(R1)     }     [(R0)     ]
                     [,OLAY=YES]
```

type

specifies the type of operation as follows:
- OPEN - Initiate Remote Terminal processing.
- GET - Receive one record from the Remote Terminal.
- PUT - Send one record to the Remote Terminal.
- CLOSE - Terminate Remote Terminal processing.

dct

specifies either a pointer to a DCT or the address of a DCT which represents the Remote Terminal Device.

If "dct" is written as an address, then it represents the address of a full word which contains the address of the Remote Terminal Device DCT in its three low-order bytes. This word must be located on a full-word boundary in core.

If "dct" is written using register notation (either regular or special register notation), then it represents the address of the Remote Terminal Device DCT.

If register notation is used, the address must have been loaded into the designated register before the execution of this macro-instruction.

loc

If "type" specifies either "OPEN" or "CLOSE" this parameter should not be specified.

If "type" specifies "GET" this parameter specifies the address of an area into which the input record will be placed. The input area must be defined large enough to contain the largest record to be received.

If "type" specifies "PUT" this parameter specifies the address of a CCW which contains the carriage control (or stacker select), address, and length of the record to be written.

If register notation is used, the appropriate address must have been loaded into the designated register before the execution of this macro-instruction.

OLAY=YES
must be specified if the $EXTP macro-instruction is coded physically within an overlay segment.

9.5.3 $WTO - HASP Write to Operator

The $WTO macro-instruction initiates output activity on one or more of the devices designated as operator consoles.

Format Description - Standard Form:

```
[symbol]    $WTO    {message-addrx}   ,   {length-value}
                    {(R1)         }       {(R0)        }
                    [,JOB=YES/NO] [,WAIT=YES/NO] [,CONVERT=YES/NO]
                    [,ROUTE=code] [,CLASS=code] [,PRI=code]
```

Format Description - Execute Form:

```
[symbol]    $WTO    {message-addrx}  [, length-value]  ,MF=(E,name)
                    {(R1)         }  [  (R0)        ]
```

Format Description - List Form:

```
[name]      $WTO   [,length-value,] MF=L
                   [,JOB=YES/NO] [,WAIT=YES/NO] [,CONVERT=YES/NO]
                   [,ROUTE=code] [,CLASS=code] [,PRI=code]
```

message
: specifies the address of a message which is to be written on the designated console(s).

  If register notation is used, the address must have been loaded into the designated register before the execution of this macro-instruction.

length
: specifies the length of the above message.

  If register notation is used, the value must have been loaded into the low-order byte of the designated register before the execution of the macro-instruction. The rest of the register must be zero unless the message is being sent to a remote terminal (see below).

  NOTE: When using the Execute and List forms of the macro-instruction, the length can be specified on either form but must not be specified on both.

JOB

specifies whether the characters "JOB nnn" will be appended to the start of the message as follows:

- YES - The job number will be appended to the start of the message.
- NO - The job number will not be appended to the message.

If this operand is omitted, JOB=YES will be assumed.

<u>CAUTION</u>: Unless JOB=NO is specified, the JCT register must be loaded with the address of the Job Control Table before the execution of this macro-instruction or the job number printed will be unpredictable.

WAIT

specifies the action to be taken in the event no Console Message Buffers are available as follows:

- YES - Return will not be made until a Console Message Buffer has become available and the message has been queued.
- NO - An immediate return will always be made with the condition code set as follows:
  - CC=0 - No Console Message Buffers were available. The message was not accepted and the macro-instruction must be re-issued.
  - CC≠0 - The message was accepted.

If this operand is omitted, WAIT=YES will be assumed.

<u>NOTE</u>: Unless WAIT=NO is specified, the message to be issued must be constructed in a re-enterable area of storage.

<u>CAUTION</u>: WAIT=NO must be specified if the $WTO macro-instruction is coded physically within an overlay segment.

CONVERT

specifies the type of consoles indicated as follows:

- YES - Logical Consoles have been specified (e.g., $LOG) and these must be converted to physical consoles by the Control Service Program.
- NO - Physical Consoles have been specified and no conversion is necessary.

If this operand is omitted, CONVERT=YES will be assumed.

ROUTE

specifies the console or consoles on which the above message is to be written. The code consists of the absolute sum of one or more of the Logical Console designations in the following list:

Designation	Console Specified
$LOG	System Log Console(s)
$ERR	Error Console(s)
$UR	Unit Record operations area
$TP	Teleprocessing operations area
$TAPE	Tape operations area
$MAIN	Chief Operator's area
$OS	OS Message Console(s)
$ALL	All of the above Consoles
$REMOTE	Remote Terminal Console

NOTE: If "$REMOTE" is specified, no other consoles should be specified, the register form of "length" must be specified, and the remote terminal number must be loaded into bits 16-23 of the register used to specify the length before the execution of the macro-instruction. Bits 0-15 of this register must be zero.

If no ROUTE is specified, the "$LOG" console will be assumed.

CAUTION: The designation "$ALL" should not be used in conjunction with any other console but should be specified alone. Failure to observe this rule will give unpredictable results.

CLASS

specifies the class of the message as one of the following:

- $ALWAYS - The message should always be written.
- $ACTION - The message requires operator action.
- $NORMAL - The message is considered essential to normal computer operations.
- $TRIVIA - The message is considered non-essential to normal computer operations.

If no CLASS is specified, $NORMAL will be assumed.

JOB

specifies whether the characters "JOB nnn" will be appended to the start of the message as follows:

- YES - The job number will be appended to the start of the message.
- NO - The job number will not be appended to the message.

If this operand is omitted, JOB=YES will be assumed.

CAUTION: Unless JOB=NO is specified, the JCT register must be loaded with the address of the Job Control Table before the execution of this macro-instruction or the job number printed will be unpredictable.

WAIT

specifies the action to be taken in the event no Console Message Buffers are available as follows:

- YES - Return will not be made until a Console Message Buffer has become available and the message has been queued.
- NO - An immediate return will always be made with the condition code set as follows:
  - CC=0 - No Console Message Buffers were available. The message was not accepted and the macro-instruction must be re-issued.
  - CC≠0 - The message was accepted.

If this operand is omitted, WAIT=YES will be assumed.

NOTE: Unless WAIT=NO is specified, the message to be issued must be constructed in a re-enterable area of storage.

CAUTION: WAIT=NO must be specified if the $WTO macro-instruction is coded physically within an overlay segment.

CONVERT

specifies the type of consoles indicated as follows:

- YES - Logical Consoles have been specified (e.g., $LOG) and these must be converted to physical consoles by the Control Service Program.
- NO - Physical Consoles have been specified and no conversion is necessary.

If this operand is omitted, CONVERT=YES will be assumed.

ROUTE

specifies the console or consoles on which the above message is to be written. The code consists of the absolute sum of one or more of the Logical Console designations in the following list:

Designation	Console Specified
$LOG	System Log Console(s)
$ERR	Error Console(s)
$UR	Unit Record operations area
$TP	Teleprocessing operations area
$TAPE	Tape operations area
$MAIN	Chief Operator's area
$OS	OS Message Console(s)
$ALL	All of the above Consoles
$REMOTE	Remote Terminal Console

NOTE: If "$REMOTE" is specified, no other consoles should be specified, the register form of "length" must be specified, and the remote terminal number must be loaded into bits 16-23 of the register used to specify the length before the execution of the macro-instruction. Bits 0-15 of this register must be zero.

If no ROUTE is specified, the "$LOG" console will be assumed.

CAUTION: The designation "$ALL" should not be used in conjunction with any other console but should be specified alone. Failure to observe this rule will give unpredictable results.

CLASS

specifies the class of the message as one of the following:

$ALWAYS - The message should always be written.
$ACTION - The message requires operator action.
$NORMAL - The message is considered essential to normal computer operations.
$TRIVIA - The message is considered non-essential to normal computer operations.

If no CLASS is specified, $NORMAL will be assumed.

PRI

specifies the priority of the message as one of the following:

- $HI - High Priority.
- $ST - Standard Priority.
- $LO - Low Priority.

If no PRI is specified, $ST priority will be assumed.

9.6 TIME SERVICES

9.6.1 $TIME - Request Time of Day

The $TIME macro-instruction obtains the time of day and returns this time in register "R0". The time is returned as an unsigned 32-bit binary number in which the least significant bit has a value of 0.01 second.

Format Description:

```
[symbol]   $TIME   [OLAY=YES]
```

OLAY=YES
: must be specified if the $TIME macro-instruction is coded physically within an overlay segment.

The time returned is the time of day based on a 24-hour clock.

9.6.2 $STIMER - Set Interval Timer

The $STIMER macro-instruction sets an interval into a programmed interval timer.

Format Description:

```
[symbol]    $STIMER    {loc-addrx}   [,OLAY=YES]
                       {(R1)     }
```

loc

specifies the address of a HASP Timer Queue Element. Before this macro-instruction is executed, the Timer Queue Element must be initialized as follows:

- ITIME must be initialized with the interval to be set in the following manner:
  - If "x" seconds are desired, then ITIME should be set to "x"; OR
  - If "y" hundredth-seconds (0.01 seconds) are desired, then ITIME should be set to the two's complement of "y".
- IPOST must be initialized with the address of the Event Wait Field to be posted.

If register notation is used, the address must have been loaded into the designated register before the execution of this macro-instruction.

For more information, refer to section 8.10: HASP Timer Queue Element Format.

OLAY=YES

must be specified if the $STIMER macro-instruction is coded physically within an overlay segment.

PROGRAMMING NOTE: An unlimited number of independent $STIMER time intervals can be active at any time provided that each has been furnished with a unique HASP Timer Queue Element.

9.6.3 $TTIMER - Test Interval Timer

The $TTIMER macro-instruction obtains the time remaining in the associated time interval that was previously set with a $STIMER macro-instruction. The value of the time interval remainder is returned in register "R0" in seconds (rounded to the nearest second). The $TTIMER macro-instruction can also be used to cancel the associated time interval.

Format Description:

```
[symbol]    $TTIMER     {loc-addrx}   [,CANCEL] [,OLAY=YES]
                        {(R1)     }
```

loc
: specifies the address of the timer queue element.

  If register notation is to be used, the address must have been loaded into the designated register before the execution of this macro-instruction.

CANCEL
: specifies that the interval in effect should be cancelled.

  If this operand is omitted, processing continues with the unexpired portion of the interval still in effect.

  If the interval expired before the $TTIMER macro-instruction was executed, the CANCEL operand has no effect.

OLAY=YES
: must be specified if the $TTIMER macro-instruction is coded physically within an overlay segment.

9.7 OVERLAY SERVICES

9.7.1 $OVERLAY - Define Overlay Segment

The $OVERLAY macro-instruction defines the instructions which follow it as an overlay segment and defines the name, priority, and residence susceptibility factor of this overlay segment.

Format Description:

```
HASPname-symbol   $OVERLAY   prio-value [,resfact-value]
```

HASPname
specifies the name to be assigned to the overlay segment. The first four characters must be the characters "HASP". The last four characters can be any unique combination of alphameric characters.

prio
specifies the priority of the overlay segment as follows:

```
0     - Lowest Priority
&LOW  - Low Priority
&MED  - Medium Priority
&HIGH - High Priority
```

resfact
specifies the residence susceptibility factor of the overlay segment as follows:

```
0     - Never Resident
&LOW  - Resident only if &OLAYLEV<4
&MED  - Resident only if &OLAYLEV<8
&HIGH - Resident only if &OLAYLEV<12
```

If this parameter is omitted, a residence factor of 0 will be used.

NOTE: This parameter may be overridden at the time that the overlay library is built.

9.7.2 $OCON - Define Overlay Constant

The $OCON macro-instruction defines an overlay constant (OCON) for use in conjunction with other overlay macro-instructions.

Format Description:

```
[symbol]    $OCON    HASPname-symbol
```

HASPname
    specifies the name of an overlay segment.

9.7.3 $LINK - Link to an Overlay Segment

The $LINK macro-instruction is used to link to an overlay segment from a non-overlay segment.

Format Description:

```
     [symbol]    $LINK    {HASPname-symbol}
                          {(register)     }
```

HASPname
: specifies the name of the overlay segment to which control is to be transferred.

  If register notation is used, the register specified must be loaded with the address of an overlay constant (OCON) which represents the overlay segment to which control is to be transferred.

9.7.4 $XCTL - Transfer Control to Another Overlay Segment

The $XCTL macro-instruction is used to transfer control from one overlay segment to another.

Format Description:

```
[symbol]    $XCTL    {HASPname-symbol}
                     {(register)     }
```

HASPname
specifies the name of the overlay segment to which control is to be transferred.

If register notation is used, the register specified must be loaded with the address of an overlay constant (OCON) which represents the overlay segment to which control is to be transferred.

9.7.5 $RETURN - Return from an Overlay Segment

The $RETURN macro-instruction is used to return control from an overlay segment to a non-overlay segment.

Format Description:

```
[symbol]   $RETURN
```

9.7.6 $LOAD - Load an Overlay Segment

The $LOAD macro-instruction is used to load an overlay segment from a non-overlay segment. The address of the overlay area into which the overlay segment has been loaded is returned in register "BASE3".

Format Description:

```
[symbol]    $LOAD    { HASPname-symbol }
                     { (register)      }
```

HASPname
: specifies the name of the overlay segment to be loaded.

  If register notation is used, the register specified must be loaded with the address of an overlay constant (OCON) which represents the overlay segment to be loaded.

9.7.7 $DELETE - Delete a Loaded Overlay Segment

The $DELETE macro-instruction is used to delete an overlay segment which has been loaded with a $LOAD macro-instruction.

Format Description:

```
[symbol]    $DELETE
```

9.8 SYNCHRONIZATION SERVICES

9.8.1 $ACTIVE - Specify Processor is Active

The $ACTIVE macro-instruction indicates to the HASP Dispatcher that the associated processor is processing a job or task.

Format Description:

```
[symbol]   $ACTIVE   [R=register]
```

R

specifies the register which is to be used by the $ACTIVE macro-instruction.

if R is omitted, register "R1" will be used.

9.8.2 $DORMANT - Specify Processor is Inactive

The $DORMANT macro-instruction indicates to the HASP Dispatcher that the associated processor has completed the processing of a job or task and is now going into a "dormant" state.

Format Description:

```
[symbol]   $DORMANT   [R=register]
```

R

specifies the register which is to be used by the $DORMANT macro-instruction.

If R is omitted, register "R1" will be used.

CAUTION: The $DORMANT macro-instruction should never be executed unless a corresponding $ACTIVE has been executed for the same processor.

9.8.3 $WAIT - Wait for a HASP Event

The $WAIT macro-instruction places the associated processor in a HASP wait condition and specifies the event upon which the processor is waiting in the Processor Control Element Event Wait Field.

Format Description:

```
[symbol]    $WAIT   event-code [,ENABLE] [,OLAY=YES]
```

event
specifies the event upon which the processor is waiting as one of the following:

```
BUF  - waiting for a HASP Buffer.
TRAK - waiting for a direct-access track address.
JOB  - waiting for a job.
UNIT - waiting for a Device Control Table.
CKPT - waiting for the completion of a HASP checkpoint.
CMB  - waiting for a Console Message Buffer.
OPER - waiting for an operator response.
IO   - waiting for the completion of an Input/Output
       operation.
WORK - waiting to be re-directed.
HOLD - waiting for a $S operator command.
DDB  - waiting for a Device Definition Table or Unit
       Control Block.
ABIT - waiting for the next HASP dispatch.
```

ENABLE
specifies that the system mask in the PSW should be set to all ones prior to returning to the HASP Dispatcher.

OLAY=YES
must be specified if the $WAIT macro-instruction is coded physically within an overlay segment.

9.8.4 $POST - Post a HASP Event Complete

The $POST macro-instruction indicates a HASP event is complete by turning off the specified bit in the indicated Event Wait Field.

Format Description:

```
[symbol]   $POST   ewf-relexp,event-code
```

ewf

specifies the address of the event wait field which is to be posted. This operand can also be written in the form D(B).

event

specifies the event which is to be posted as one of the following:

- BUF - a HASP Buffer has been returned.
- TRAK - direct-access space has been released.
- JOB - a HASP Job Queue Element has changed status.
- UNIT - a Device Control Table has been released.
- CKPT - a HASP checkpoint has completed.
- CMB - a Console Message Buffer has been returned.
- OPER - an operator has responded.
- IO - an Input/Output operation has completed.
- WORK - a processor has been re-directed.
- HOLD - an operator has entered a $S command.
- DDB - a Device Definition Table or a Unit Control Block has been released.

CAUTION: The $POST macro-instruction should not be executed unless addressability to the HASP Communication Table (HCT) has been established.

9.8.5 $ENABLE - Enable Interrupts

The $ENABLE macro-instruction causes the specified interrupts to be enabled.

Format Description:

```
     [symbol]   $ENABLE   mask-code [,OLAY=YES]
```

mask

specifies the interrupts to be enabled as follows:

- ALL - Enable all interrupts.
- JCL - Enable the interrupts which were enabled when the Reader/Interpreter appendage was entered. NOTE: This code can be specified only in the Reader/Interpreter appendage.

OLAY=YES

must be specified if the $ENABLE macro-instruction is coded physically within an overlay segment.

9.8.6 $DISABLE - Disable Interrupts

The $DISABLE macro-instruction causes the specified interrupts to be disabled.

Format Description:

```
     [symbol]    $DISABLE   mask-code [,OLAY=YES]
```

mask
: specifies the interrupts to be disabled as follows:
  ALL - Disable all interrupts.
  INT - Disable Interval Timer Interrupt.

OLAY=YES
: must be specified if the $DISABLE macro-instruction is coded physically within an overlay segment.

9.9 DEBUG SERVICES

9.9.1 $TRACE - Make Entry in the HASP Trace Table

The $TRACE macro-instruction makes an entry in the HASP trace table if the &TRACE option is set non-zero. If the &TRACE option is set to zero, this macro-instruction does not generate any code.

Format Description:

    [symbol] $TRACE

PROGRAMMING NOTE: The $TRACE macro-expansion and associated Control Service Program preserve all registers and the condition code. For more information concerning the HASP trace table, refer to Section 5.11.

9.9.2 $COUNT - Count Selected Occurrences

The $COUNT macro-instruction increments a counter every time the macro-instruction is executed and can be used to determine the number of times a particular event occurs or a particular section of code is entered. The counter is a half-word counter (modulo 65,536) which is located fourteen bytes deep in the macro expansion (symbol+14).

Format Description:

```
[symbol]   $COUNT   [R=register]
```

R

specifies a register to be used in performing the counting operation.

If this parameter is omitted, register "R1" will be used.

9.10 ERROR SERVICES

9.10.1 $ERROR - Indicate Catastrophic Error

The $ERROR macro-instruction is used to indicate that a catastrophic error has occurred, one that prevents any further processing by HASP. The macro-instruction causes the following message to be printed out on the console specified by HASPGEN parameter $PRICONA:

    $ HASP SYSTEM CATASTROPHIC ERROR. CODE = symbol

Format Description:

    symbol $ERROR

symbol
consists of a four-character symbol indicating the type of error which occurred.

This operand usually consists of a letter, two digits, and trailing blanks, and will be printed as the error code in the message which is printed.

NOTE: This operand must be present.

9.10.2 $DISTERR - Indicate Disastrous Error

The $DISTERR macro-instruction is used to indicate that a disastrous error has occurred. The macro-instruction causes the following message to be printed out on the $ERR and $LOG consoles:

DISASTROUS ERROR - COLD START SYSTEM ASAP

Format Description:

[symbol] $DISTERR [OLAY=YES]

OLAY=YES
must be specified if the $DISTERR macro-instruction is coded physically within an overlay segment.

9.10.3 $IOERROR - Log Input/Output Error

The $IOERROR macro-instruction is used to log an Input/Output Error on the operator's console.

Format Description:

```
[symbol]   $IOERROR   {buffer-addrx}   [,OLAY=YES]
                      {(R1)        }
```

buffer
specifies either a pointer to a HASP buffer or the address of a buffer which has been associated with a HASP Input/Output error.

If "buffer" is written as an address then it represents the address of a full-word which contains the address of the buffer in error in its three low order bytes. This word must be located on a full-word boundary in core.

If "buffer" is written using register notation (either regular or special register notation), then it represents the address of the buffer in error.

If register notation is used, the address must have been loaded into the designated register before the execution of the macro-instruction.

OLAY=YES
must be specified if the $IOERROR macro-instruction is coded physically within an overlay segment.

9.11 CODING AID SERVICES

9.11.1 $GLOBAL - Define GLOBAL Symbols

The $GLOBAL argument on a COPY instruction causes all HASP GLOBAL Symbols to be defined. This COPY instruction must be the first instruction in an assembly (except for TITLE, EJECT, and SPACE operations) to function correctly.

Format Description:

```
COPY   $GLOBAL
```

9.11.2 $HASPGEN - Define HASPGEN Parameters

The $HASPGEN argument on a COPY instruction causes all general HASPGEN parameter values to be defined. This COPY instruction may be placed anywhere in an assembly but must follow the COPY $GLOBAL instruction.

Format Description:

```
COPY   $HASPGEN
```

9.11.3 NULL - Define a Symbol

The NULL macro-instruction defines the symbol in the name field, if any, as having the current value of the location counter rounded up, if necessary, to a half-word boundary.

Format Description:

```
[symbol]   NULL
```

9.11.4 $HASPCB - Generate HASP Control Blocks

The $HASPCB macro-instruction causes the specified HASP Control Block definitions and, optionally, documentation for those control blocks to be generated.

Format Description:

```
$HASPCB   cb1-code [,cb2-code]...[,cb24-code] [,DOC=YES]
```

cb1-cb24

specifies the control block definitions to be generated as follows:

```
HCT     - HASP Communication Table DSECT (or CSECT)
PCE     - HASP Processor Control Element DSECT
BUFFER  - HASP Buffer DSECT
CMB     - HASP Console Message Buffer DSECT
DCT     - HASP Device Control Table DSECT
JQE     - HASP Job Queue Element Definitions
JIT     - HASP Job Information Table Definitions
JCT     - HASP Job Control Table DSECT
TED     - HASP Track Extent Data Table DSECT
TQE     - HASP Timer Queue Element Definitions
OTB     - HASP Overlay Table DSECT
DDT     - HASP Data Definition Table DSECT
PIT     - HASP Partition Information Table Definitions
PRC     - HASP Print Checkpoint Element Definitions
MSA     - HASP Message Allocation Control Block DSECT
CVT     - OS Communication Vector Table DSECT
TCB     - OS Task Control Block DSECT
RB      - OS Request Block DSECT
DCB     - OS Data Control Block DSECT
DEB     - OS Data Extent Block DSECT
UCB     - OS Unit Control Block DSECT
RDRWORK - HASP Input Processor PCE Work Area DSECT
XEQWORK - HASP Execution Processor PCE Work Area DSECT
PPPWORK - HASP Print/Punch PCE Work Area DSECT
```

These arguments can be specified in any combination with the following exceptions:

1) If JCT is specified, BUFFER must be specified as a prior argument.
2) If RDRWORK, XEQWORK, or PPPWORK is specified, PCE must be specified as a prior argument.

DOC=YES

specifies that documentation of the control blocks is desired.

9.11.5 $XXC - Variable Core to Core Operation

The $XXC macro-instruction generates a variable number of core-to-core operations such that there is virtually no restriction on the length of such an operation. The $XXC is especially useful when the length of a core-to-core operation is dependent upon the value of an assembly parameter which may cause the number of operations needed to vary.

Format Description:

```
[symbol]    $XXC    op-code,to-relexp,from-relexp
                    [,length-integer]
```

op

specifies the core-to-core operation as one of the following:

```
NC  - AND
XC  - Exclusive OR
MVC - Move
MVN - Move Numerics
MVZ - Move Zones
OC  - OR
TR  - Translate
```

to

specifies the address of the first field.

This operand may optionally be written as two absolute expressions separated by a comma and enclosed in parentheses. The first expression will be interpreted as a displacement and the second as a base register.

from

specifies the address of the second field.

This operand may optionally be written as two absolute expressions separated by a comma and enclosed in parentheses. The first expression will be interpreted as a displacement and the second as a base register.

length

specifies the total number of bytes in the field.

If this operand is omitted, the length attribute of the first field will be used.

9.11.6 $PATCHSP - Generate Patch Space

The $PATCHSP macro-instruction causes a specified number of bytes of patch space to be generated. This patch space will be divided into half words and listed in the assembly in such a way that both the assembly location (for REPing and SUPERZAPing) and the Base-Displacement (in the form BDDD) will be printed for each half word.

Format Description:

```
[symbol]   $PATCHSP   length-number
```

length
    specifies the length of the patch space in bytes.

CAUTION: Local addressability is required for this macro-instruction to assemble correctly.

9.11.7 $DLENGTH - Compute Decimal Length

The $DLENGTH macro-instruction causes the length of a CSECT (or DSECT) to be computed and that length to be printed in decimal.

Format Description:

```
     symbol   $DLENGTH   [HEADER=character]
```

symbol
: specifies a name to which the decimal length of the CSECT (or DSECT) will be assigned. This must be unique for each use of the $DLENGTH macro-instruction.

HEADER
: specifies a one-character header which will insure unique internally generated symbols. This must be specified differently for each use of the $DLENGTH macro-instruction.

  If this operand is omitted, the character "L" will be used.

9.11.8 $RTAMDEF - Remote Terminal Access Method Definitions

The $RTAMDEF argument on a COPY instruction causes certain Remote Terminal Access Method Symbols to be defined.

Format Description:

```
COPY   $RTAMDEF
```

10.0 HASP MAINTENANCE PROCEDURES

This section describes various maintenance procedures for the HASP System and is intended primarily for use by systems programmers.

10.1 GENERATING A HASP SYSTEM (HASPGEN)

To generate a HASP System which conforms to the needs of a particular installation, it is necessary to allocate and catalog several data sets, build a tailored version of the HASP source coding in one of the data sets, assemble several of the HASP source modules, and do a few other utility functions.

10.1.1 Data Set Requirements for HASPGEN

Table 10.1.1 lists the data sets required for HASPGEN and their contents at the end of the full HASPGEN process.

Figure 10.1.2 shows a sample job which will allocate and catalog the required data sets on two 2311 disk volumes. UNIT and SPACE parameters should be changed as appropriate if other direct-access devices are used. VOLUME parameters may be changed as desired. Data sets SYS1.UT1 and/or SYS1.UT2 may be assigned to labeled tape(s) if desired.

Table 10.1.1 - HASPGEN Data Set Description

Data Set Name	Member Names	Description
SYS1.HASPSRC (HASP Source Coding)	$ACTIVE thru $XXC	74 HASP Macros
	CVT	OS CVT Macro
	HASPACCT	Accounting Routine
	HASPBR1	Return Module
	HASPCOMM	Command Processor
	HASPCON	Console Support
	HASPINIT	Initialization Routine
	HASPJCL	Sample Install Jobs
	HASPMISC	Miscellaneous Routines
	HASPNUC	HASP Nucleus
	HASPOBLD	Overlay Build Utility
	HASPPRPU	Print/Punch Processor
	HASPRDR	Input Processor
	HASPRTAM	Remote Support
	HASPSVC	SVC Routine
	HASPWTR	SMB Writer
	HASPXEQ	Execution Processors
	HRTPB360	360 and M20 BSC Remote Program
	HRTPLOAD	1130 Loader Program
	HASPOPTS	RMTGEN Standard Option Lists
	HASPSM20	M20 STR Remote Program
	HASPSYS3	System/3 Remote Program
	HASP1130	1130 Remote Program
	IEFUCBOB	OS UCB Macro
	NULL	HASP Macro
SYS1.HASPOBJ (HASP Object Decks)	HASPBR1	same as SYS1.HASPSRC
	HASPNUC	
	HASPRDR	
	HASPXEQ	
	HASPPRPU	
	HASPACCT	
	HASPMISC	
	HASPCON	
	HASPRTAM	
	HASPCOMM	
	HASPINIT	
	HASPSVC	
	HASPWTR	
	HASPOBLD	

Data Set Name	Member Name	Description
SYS1.HASPMOD (HASP Load Modules)	HASPGEN	HASPGEN Program
	EXRMTGEN	Initial RMTGEN Program
	RMTGEN	RMTGEN Control Program
	GENRMT	RMTGEN Effector Program
	LETRRIP	1130 RMTGEN Post-Processor
	SYS3CNVT	System/3 RMTGEN Post-Processor
	HASPOBLD	Overlay Build Utility
SYS1.UT1		
SYS1.UT2		
SYS1.UT3 (Sequential Scratch Data Sets)		

Figure 10.1.2 - Sample Job to Catalog Data Sets for HASPGEN

```
//CATALOG  JOB (0000,0000),'HASP DATA SETS',MSGLEVEL=1
//SCRATCH EXEC PGM=IEHPROGM
//TWOSPACK  DD UNIT=2311,VOLUME=SER=222222,DISP=OLD
//HASP      DD UNIT=2311,VOLUME=SER=HASP,DISP=OLD
//SYSPRINT  DD SYSOUT=A
//SYSIN     DD *
   UNCATLG     DSNAME=SYS1.HASPSRC
   UNCATLG     DSNAME=SYS1.HASPOBJ
   UNCATLG     DSNAME=SYS1.HASPMOD
   UNCATLG     DSNAME=SYS1.UT1
   UNCATLG     DSNAME=SYS1.UT2
   UNCATLG     DSNAME=SYS1.UT3
   SCRATCH     VTOC,VOL=2311=222222,PURGE
   SCRATCH     VTOC,VOL=2311=HASP,PURGE
/*
//ALLOCAT EXEC PGM=IEHPROGM
//SYSIN     DD DUMMY
//SYSPRINT  DD DUMMY
//HASPSRC   DD DSNAME=SYS1.HASPSRC,UNIT=2311,VOLUME=SER=HASP,
//             DISP=(,CATLG),SPACE=(CYL,(30,5,5)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3360)
//HASPOBJ   DD DSNAME=SYS1.HASPOBJ,UNIT=2311,VOLUME=SER=HASP,
//             DISP=(,CATLG),SPACE=(CYL,(5,5,5)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)
//HASPMOD   DD DSNAME=SYS1.HASPMOD,UNIT=2311,VOLUME=SER=HASP,
//             DISP=(,CATLG),SPACE=(CYL,(5,5,5))
//UT1       DD DSNAME=SYS1.UT1,UNIT=2311,VOLUME=SER=222222,
//             DISP=(,CATLG),SPACE=(CYL,(50,5))
//UT2       DD DSNAME=SYS1.UT2,UNIT=2311,VOLUME=SER=HASP,
//             DISP=(,CATLG),SPACE=(CYL,(50,5))
//UT3       DD DSNAME=SYS1.UT3,UNIT=2311,VOLUME=SER=222222,
//             DISP=(,CATLG),SPACE=(CYL,(50,5))
```

10.1.2 HASPGEN Parameter Cards

All HASPGEN parameters and their default values are discussed in Section 7. After the desired value for each parameter has been determined, the values of those which are to be changed from the default values are usually punched into cards, to be read by the HASPGEN utility program.

Each parameter should be punched in the format: "option=value", beginning in column 1 of a card, where "option" represents a HASPGEN parameter and "value" represents a permissible value for that parameter, as described in Section 7. The above format must not contain embedded blanks. The first blank terminates the "value" field and the rest of the card may contain comments.

HASPGEN parameter cards may occur in a deck in any order. If the same parameter occurs more than once, the last occurrence determines the parameter's value. A deck of one or more HASPGEN parameter cards is usually terminated by a card with "END" punched in columns 1-3. If symbolic updates (PTFs or user modifications) are to be applied, then the "END" card should be replaced by an "UPDATE" card (see 10.1.3). Alternate methods of entering HASPGEN parameters are discussed in 10.1.5.

10.1.3 HASPGEN Update Cards

Source coding of any member in SYS1.HASPSRC (see Table 10.1.1) may be updated by cards punched according to a subset of the formats acceptable to the IEBUPDAT OS utility program. This is the method used to apply Official HASP Maintenance Changes (PTFs, etc.) and user modifications to HASP, if any. Updates are placed following the HASPGEN parameter deck, immediately after a card with "UPDATE" punched in columns 1-6 (see 10.1.2).

Only the ./ CHNGE ... and ./ DELET ... control cards are defined for use with HASPGEN Update. Fields following the module (member) name on the CHNGE card are ignored, if present. Other control cards defined for use with IEBUPDAT should not be used.

A card without "./" in columns 1 and 2 replaces an existing source card (if columns 73-80 match an existing card in the member) or is inserted between existing source cards, according to ascending collating sequence based on columns 73-80. Cards which are blank in columns 73-80 (or which do not maintain the ascending collating sequence) are inserted immediately following the last modification card which was in ascending collating sequence.

All PTFs (and user modifications, if any) which apply to one source module must be integrated into a single deck, beginning with a CHNGE card naming that module, in ascending sequence number order. If more than one module is updated, the decks must be placed together so that the module names on CHNGE cards are in ascending collating sequence, as listed in Table 10.1.1 under SYS1.HASPSRC.

The last source update card must be followed by a ./ ENDUP control card and a /* delimiter card. Figure 10.1.3 shows a composite deck of HASPGEN parameters and source updates in correct order.

Figure 10.1.3 - Sample HASPGEN Parameter and Update Deck

```
Columns
1          10    16                                      73    80
&NUMLNES=1
&BSCCPU=YES
LINE01=02011
RMT01=01010100153643
UPDATE                       (END if no source updates follow)
./         CHNGE HASPMISC
           (modifications to module HASPMISC)      nnnnnnnn
           .
           .
           .
./         CHNGE HASPWTR
           (modifications to module HASPWTR)       mmmmmmmm
           .
           .
           .
./         ENDUP
/*
```

10.1.4 Standard Complete HASPGEN Process

For most installations, a complete standard HASPGEN may be performed (if the required data sets are allocated and cataloged) simply by using the first file of the distributed HASPGEN tape as an OS input stream and executing, in order, all the jobs it contains. Table 10.1.4 lists the jobs, steps, and functions of each, in the order they occur in the first file of the tape.

The first file of the tape may be executed directly, under HASP with MFT or MVT, by starting a HASP input tape (TPEn) using a tape drive as the input unit.

If PCP or PCP Starter System is used, the first file of the tape must be punched or copied to another tape, then read as a job stream. This is because the first job will read the second file of the tape which contains the entire HASP source coding. It would not be possible to read the first and second files from a single tape simultaneously, which is what a PCP system would attempt to do.

If MFT or MVT without HASP is used, then the first file of the tape must be punched and the first job (HASPGEN) run to completion before other jobs are read by the OS Reader/Interpreter. During subsequent generations with the same OS system, the first file may be processed directly by the OS RDR.

During the first job (HASPGEN) the HASPGEN utility program will write the following WTOR message on the console:

```
nn ENTER HASPGEN OPTION CHANGES (option=value), CARDS,
   UPDATE, OR END.
```

The composite HASPGEN parameter and update deck (example Figure 10.1.3) should be placed in the 2540 card reader and the following reply should be entered:

```
REPLY nn,'cards'
```

The listing output of the HASPGEN job includes:

All HASPGEN parameters with their default values
User changes to HASPGEN parameters
Source changes made to modules by HASPGEN Update

In multi-programming systems, care should be taken that the jobs as listed in Table 10.1.4 execute in sequential order under a single initiator.

If HASPGEN parameters (&BSCCPU or &STRCPU) are set to include programmable Remote Job Entry support, then job HRMTGEN will issue another WTOR console message, which allows optional generation of Remote Terminal Programs as part of the full HASPGEN process. Refer to Section 10.3.2 for further details.

If all jobs in the first file of the HASPGEN tape are executed successfully, all data sets and members as listed in Table 10.1.1 will be completed and the punched card output will contain:

Any Remote Terminal Programs created by HRMTGEN (optional, see 10.3.2)

HASPJCL, the deck of sample jobs to install HASP (described in 10.2.2)

Table 10.1.4 - HASPGEN Tape First File Job Description

Job	Step (if multi-step)	Function
HASPGEN	LNK	Link Edits object decks for HASPGEN, EXRMTGEN, RMTGEN, GENRMT, LETRRIP, and SYS3CNVT into SYS1.HASPMOD
	HASPGEN	Executes HASPGEN program which reads all source code from second file of tape, applies user HASPGEN parameter modifications and (optionally) source code modifications, and builds each source member in SYS1.HASPSRC
	PROCS	Adds procedures ASMHASP, HASPGEN, and RMTGEN to SYS1.PROCLIB, if not already there
HASMBR1		Assembles source module HASPBR1
HASMNUC		Assembles source module HASPNUC
HASMRDR		Assembles source module HASPRDR
HASMXEQ		Assembles source module HASPXEQ
HASMPRPU		Assembles source module HASPPRPU
HASMACCT		Assembles source module HASPACCT
HASMMISC		Assembles source module HASPMISC
HASMCON		Assembles source module HASPCON
HASMRTAM		Assembles source module HASPRTAM
HASMCOMM		Assembles source module HASPCOMM
HASMINIT		Assembles source module HASPINIT
HASMSVC		Assembles source module HASPSVC
HASMWTR		Assembles source module HASPWTR
HASMOBLD	OBLD	Assembles source module HASPOBLD
	LNKOBLD	Link Edits object deck HASPOBLD into SYS1.HASPMOD
HRMTGEN		Performs optional initial RMTGEN for one or more HASP Remote Terminal Programs (see 10.3.2)
HASPJCL	PRINT	Prints source member HASPJCL (sample jobs to install HASP, see 10.2.2)
	PUNCH	Punches source member HASPJCL

I want to change [illegible]
rerun HASPGEN [illegible]
assembly [illegible]
HASPOBLD

10.1.5 Some HASPGEN Variations

An installation may find it necessary or desirable to vary some of the standard HASPGEN process described previously. A few of the possibilities are given below.

The necessity of punching or copying the first file of the HASPGEN tape, in order to generate under a system without HASP, is discussed in 10.1.4. The installation's requirements for particular job card accounting fields or classes, or the absence of a 2540 card reader, may also require the first file to be punched, listed, and used as an input stream after appropriate modifications to the JCL.

During the execution of the HASPGEN utility, responses to the WTOR message other than 'cards' may be used. Individual HASPGEN parameters may be entered by using a reply text of 'option=value', where these terms have the same meaning as described for HASPGEN parameter cards in 10.1.2. Lower case may be used, but no blanks or comments are allowable. Each HASPGEN parameter entered from the console is acknowledged by a message if correct or else by a diagnostic, with opportunity to re-enter a correct form. The same parameter may be entered repeatedly; only the last value entered will be used. The 'cards' reply may be entered at any time to cause further parameter reading from the 2540 card reader. If all parameters are entered from the console, a reply text of 'update' may be entered to cause reading of an update deck only (all cards after UPDATE in Figure 10.1.3) from the 2540 card reader. If all parameters are entered from the console and there are no updates, a reply text of 'end' may be used to terminate all entry to HASPGEN.

If all the actions of the HASPGEN job (Table 10.1.4) are performed once and the three partitioned data sets SYS1.HASPSRC, OBJ, MOD are preserved on a disk pack, then later full or partial HASPGENs may be performed under a production batch system by using jobs such as the examples given in Figure 10.1.5. Execution of the HASPGEN proc invokes only the HASPGEN utility, with a PARM field causing the WTOR and reply to be omitted so that parameters and updates are read directly from the input stream. The data set SYS1.HASPSRC would normally be scratched and re-allocated prior to running this job. If all 14 assemblies (Table 10.1.4) are to be done, SYS1.HASPOBJ should also be scratched and re-allocated. Figure 10.1.5 shows how to use the ASMHASP proc to do assemblies. If HASPOBLD is assembled, a step should be added to link edit it from SYS1.HASPOBJ into SYS1.HASPMOD.

Partial HASPGEN may be used to save processing time, if only minor changes are made to HASPGEN parameters or only a small number of modules are changed by updates. The recommended process is to scratch and re-allocate SYS1.HASPSRC only, then to use the HASPGEN proc and full parameter/update deck to re-create SYS1.HASPSRC. Only required assemblies are performed, using ASMHASP proc, with decks replacing those of same name in SYS1.HASPOBJ.

A module must be re-assembled if a HASPGEN parameter(s) is changed, compared to the previous HASPGEN, and Table 10.1.6 indicates that the module depends upon the parameter(s). A change in the update portion of the deck for a module, compared to the previous HASPGEN, also requires that the module be re-assembled. If in any case re-assembly requirements are doubtful (e.g., changes in update deck for any member of SYS1.HASPSRC other than one of the 14 assembly modules), all 14 modules must be re-assembled.

The module HASPBR1 does not actually depend on any generation parameter. However, it contains the most complete commented documentation of all HASP Control Blocks which does depend on various HASPGEN parameters. Therefore HASPBR1 should be re-assembled periodically to provide listing documentation current with operational HASP.

Table 10.1.6 refers to the assembly modules by using a single alphabetic character for each, according to the following equivalences.

H = HASPNUC
R = HASPRDR
X = HASPXEQ
P = HASPPRPU
A = HASPACCT
V = HASPMISC
W = HASPCON
M = HASPRTAM
C = HASPCOMM
N = HASPINIT
S = HASPSVC
T = HASPWTR
O = HASPOBLD

Figure 10.1.5 - Sample Batch HASPGEN Jobs

```
//HASPGEN  JOB  ...
//JOBLIB     DD  DSN=SYS1.HASPMOD,DISP=SHR
//GEN      EXEC  HASPGEN
//HASPGEN.OPTIONS DD *
  (deck as in Figure 10.1.3)
/*
//HASMNUC  JOB  ...
//NUC      EXEC  ASMHASP,MODULE=HASPNUC
//HASPINIT JOB  ...
//INIT     EXEC  ASMHASP,MODULE=HASPINIT
```

Table 10.1.6 - Module Dependencies on HASPGEN Parameters

&ACCTNG -HPA	&NUMPUNS-HPVN	&SPD2260-W
&AUTORDR-WCN	&NUMRDRS-HWN	&SPOLMSG-PMN
&BSCCPU -PM	&NUMRJE -MCN	&STRCPU -M
&BSC2770-M	&NUMTGV -HRXPAVWCN	&STR1978-M
&BSC2780-M	&NUMTPBF-N	&TIMEOPT-X
&BSHPRSU-M	&NUMTPES-HN	$TIMEXS -X
&BSVBOPT-M	&NUMTPPR-HPVN	&TPBFSIZ-HMN
&BUFHICH-N	&NUMTPPU-HPMN	$TPIDCT -P
&BUFSIZE-HRXPMN	&NUMTPRD-H	&TRACE -HXVWN
$CKPTIME-V	&NUMWTOQ-HN	&USASCII-M
&CLS(n) -X	&OLAYLEV-RXPAVMCN	$WAITIME-M
&CONAUTH-N	&OREPSIZ-HN	&WCLSREQ-T
&DEBUG -HXVCNO	&OSC(n) -X	&WTLOPT -CN
$DELAYCT-M	&OSINOPT-R	&WTR -XWCN
&DMPTAPE-N	&OUTPOPT-X	&WTRCLAS-XNT
$ESTIME -R	$OUTXS -X	&WTRPART-WCN
$ESTLNCT-R	&PID(n) -X	&XBATCHC-RXWCN
$ESTPUN -R	&PRI(n) -X	&XBATCHN-RXWC
&INITSVC-XNS	$PRICONA-H	&XLIN(n)-RX
&JITSIZE-HRXVCN	$PRIDCT -P	&XPRI(n)-X
$LINECT -R	&PRIHIGH-V	&XZMFTH -V
LINEmm -N	&PRILOW -V	&XZMFTL -V
&LOGOPT -X	&PRIRATE-HV	&XZPRTY -XV
&MAXCLAS-XCN	$PRTBOPT-P	$$x -X
&MAXJOBS-VN	&PRTRANS-PM	
&MAXPART-X	&PRTUCS -N	
&MAXXEQS-HXVWN	$PUNBOPT-P	
&MINBUF -N	&RDR -XN	
&MLBFSIZ-M	&RDRPART-N	
&MONINTV-HXV	$REPRDR -N	
&NOPRCCW-HPC	$REPWTR -N	
&NOPUCCW-HPC	&RESCORE-N	
&NUMBUF -N	&RJOBOPT-R	
&NUMCONS-HXWCN	RMTnn -N	
&NUMDA -HRXPAVWMCN	$RPRBOPT-P	
&NUMDDT -RX	&RPRI(n)-R	
&NUMINRS-HRXN	&RPRT(n)-R	
&NUMLNES-HRPWMN	$RPUBOPT-P	
&NUMOACE-N	&RQENUM -W	
&NUMPRTS-HPVN	&SIZ2260-WN	

10.2.1 OS SYSGEN REQUIREMENTS FOR HASP

In order to utilize HASP, the following additions should be made to the standard installation OS SYSGEN STAGE 1 input deck.

10.2.1.1 Pseudo Devices

Pseudo readers, printers and punches should be generated according to the following formulas.

```
Number of pseudo 2540 readers   = INDD*&MAXXEQS+1
Number of pseudo 1403 printers  = PRDD*&MAXXEQS+1
Number of pseudo 2540 punches   = PUDD*&MAXXEQS
Number of pseudo 1443 printers  = SFPRDD*&MAXXEQS
Number of pseudo 1442 punches   = SFPUDD*&MAXXEQS
Number of pseudo 2520 punches   = &NUMINRS
```

Where:

INDD	= maximum number of DD * (or DD DATA) cards <u>per job</u>
PRDD	= maximum number of print data sets <u>per step</u>
PUDD	= maximum number of punch data sets <u>per step</u>
SFPRDD	= maximum number of print data sets requiring special forms or special routing <u>per job</u>
SFPUDD	= maximum number of punch data sets requiring special forms or special routing <u>per job</u>
&MAXXEQS	= maximum number of simultaneous Job executions
&NUMINRS	= number of Internal Reader interfaces

It should be noted that the term "Pseudo Device" implies a physically non-existent device. The addresses chosen for pseudo devices may be any hexadecimal values from 000 to 6FF; they need not be contiguous and they <u>must not match the address of any existent device or other pseudo device</u>.

10.2.1.2 Additional Symbolic Unit Names

The symbolic unit name "A" should be assigned to all pseudo 1403 printers, <u>except the one identified by the HASPGEN parameter &WTR</u>. The symbolic unit name "B" should be assigned to all pseudo 2540 punches.

The Pseudo Device and Symbolic Unit Name requirements are satisfied by using the SYSGEN macros CHANNEL, IOCONTRL, IODEVICE, and UNITNAME. The following examples give a simple method of generating the required devices and names for OS Release 18 and later releases.

```
Pseudo 2540 Reader
     IODEVICE  UNIT=DUMMY,ADDRESS=xxx,DEVTYPE=10000801

Pseudo 1403 Printer
     IODEVICE  UNIT=DUMMY,ADDRESS=xxx,DEVTYPE=10000808
     UNITNAME  NAME=A,UNIT=xxx  (omit if xxx=&WTR)

Pseudo 2540 Punch
     IODEVICE  UNIT=DUMMY,ADDRESS=xxx,DEVTYPE=10000802
     UNITNAME  NAME=B,UNIT=xxx

Pseudo 1443 Printer
     IODEVICE  UNIT=DUMMY,ADDRESS=xxx,DEVTYPE=1000080A

Pseudo 1442 Punch
     IODEVICE  UNIT=DUMMY,ADDRESS=xxx,DEVTYPE=51800803

Pseudo 2520 PUNCH
     IODEVICE  UNIT=DUMMY,ADDRESS=xxx,DEVTYPE=11000805
```

Because UNIT=DUMMY is used, control unit macros are not required. However, for hardware reasons, the channel and control unit digits in the addresses used for pseudo devices should not match an existent channel <u>and</u> control unit. For example, if the system has a 2314 using addresses 130 through 137, then no pseudo device should be generated with an address 13x.

The pseudo 2520 punches may be given a descriptive symbolic unit name, as in the following example. This will make allocation easier for programmers using the Internal Reader feature of HASP.

```
UNITNAME  UNIT=(301,302,...),NAME=INTRDR
```

10.2.1.3 <u>Position for the HASP Type I SVC</u>

The following card must be included in SYSGEN input to reserve a position for installation of the HASP Type I SVC. For "nnn", the value assigned to the HASPGEN parameter &INITSVC is used.

```
SVCTABLE  SVC-nnn-T1-S0
```

10.2.1.4 Installation of the HASP SVC at SYSGEN Time

If HASPGEN has been completed prior to SYSGEN, it is possible to cause the installation of the HASP SVC in the OS Nucleus during STAGE 2 of SYSGEN. The data set SYS1.HASPOBJ must be cataloged in the generating system and the following card must be included in the STAGE 1 SYSGEN input.

```
RESMODS    PDS=SYS1.HASPOBJ,MEMBERS=HASPSVC
```

10.2.1.5 MFT Partitions

Consideration should be given when generating MFT Systems to setting partition sizes and classes properly. This will minimize required operator actions when HASP is invoked.

HASP will normally reside in P0, but this is not mandatory. Size of P0 (or partition in which HASP resides) should be sufficient to contain the HASP load modules whose size may be determined from the link edit described under Section 10.1.4.3.

If the &WTRPART HASPGEN parameter is not set to "*", a partition (normally P1) will be needed for the OS Writer.

Other job processing partitions should be generated, each with only one eligible job class. These classes should be unique and match the classes assigned to the HASPGEN parameters &OSC(n). One OS partition should be eligible for Class A jobs, to allow processing of any job with a JCL error so severe that the OS R/I defaults it to Class A.

The following is an example of compatible HASPGEN parameters and SYSGEN PARTITNS macro.

```
&OSC(1)=A      &PID(1)=2       &WTRPART=P1
&OSC(2)=B      &PID(2)=3
&OSC(3)=C      &PID(3)=4

PARTITNS       P0(C-H,S-50K),P1(C-W,S-10K),P2(C-A,S-100K),
               P3(C-B,S-100K),P4(C-C,S-100K)
```

The &PID(n) parameters are only used to make HASP operator messages correspond to actual physical partitions. If &WTRPART=*, then P1 may be eliminated or allocated zero storage, which will allow later optional use of the OS Writer.

10.2.1.6 MFT Features

Certain features of HASP, when used with an MFT System, require that the MFT System be OS Release 19 or a later release and that certain optional MFT features be specified in the SYSGEN.

If HASPGEN parameters are specified (or defaulted) so that &WTRPART=* and/or &NUMCONS=0 and/or &MONINTV is greater than zero, then the MFT System must include the multitasking capability. This is specified by including "ATTACH" in the OPTIONS parameter of the SUPRVSOR macro. See OS System Generation documentation for further details. It is not necessary or beneficial for HASP's purposes to make ATTACH resident, which would require additional fixed storage.

If &NUMCONS=0, the MFT System must also have a four (4) byte SVC table. This is specified by including "TRSVCTBL" in the OPTIONS parameter of the SUPRVSOR macro.

10.2.1.7 Timer Requirement

Use of HASP requires that OS have certain software support for the hardware interval timer. The TIMER parameter of the SUPRVSOR macro must specify "INTERVAL" or "JOBSTEP". The specification of "JOBSTEP" is required if the HASPGEN parameter &MONINTV is greater than zero.

OS SYSGEN requirements as stated above in Sections 10.2.1.1, 2, 3 and 7 are mandatory if the System is to be used with HASP. Other requirements stated above may be satisfied at a later time (SVC may be installed later if position has been reserved, partitions may be set at IPL) or are optional depending upon use of optional features in HASP.

10.2.2 INSTALLING HASP IN AN MFT OR MVT SYSTEM

see Sec 12 for JCL

To install HASP, it is necessary to perform some or all of the following four processes, after HASPGEN has been completed. Four sample jobs, one for each process, are printed and punched from the source member HASPJCL when HASPGEN is performed as described previously in Section 10.1.4. These jobs are also listed in Section 12.1 for reference.

It must be emphasized that the sample jobs are just samples. If run exactly as punched, they will probably produce incorrect results. Each process is discussed below with comments about what modifications to the sample job may be necessary.

10.2.2.1 Install HASP SVC

12.1-2

This process is not necessary if the SVC was installed during OS SYSGEN as described previously under Section 10.1.3.4. If not done then, the sample job HASPSVC may be used.

The three step job HASPSVC scratches a second OS Nucleus data set named SYS1.OLDNUC or SYS1.NEWNUC, link edits the standard Nucleus with the HASP SVC into a newly created SYS1.NEWNUC, then performs renaming so that the new Nucleus becomes the standard SYS1.NUCLEUS data set.

All references in the sample job to volume YYYYYY should be changed to the volume serial of the system residence volume. The unit and space allocation for SYS1.NEWNUC should be made to agree with that used for SYS1.NUCLEUS during SYSGEN. Only two of the three INSERT cards should be used, as indicated by comments on the cards. INSERT cards other than those shown may be required if the OS Nucleus contains special features, such as Channel Check Handler. INSERT cards actually used should match those shown on the listing of the OS Nucleus link edit during SYSGEN.

Alternative procedures may be used to install the HASP SVC, including use of alternate members within the single data set SYS1.NUCLEUS if space permits. Naming of these members and IPL procedures are described in appropriate OS documentation.

10.2.2.2 Install HASP Procs

The sample job HASPROCS should be used to add necessary cataloged procedures (members) to the system's SYS1.PROCLIB data set. The members are described below.

Region specifications in all of the members may be modified up or down to fit actual minimum storage required in a particular MVT System, as determined by OS Storage Estimates or actual experience with a particular OS Release. Values given in the samples are appropriate for Release 19 with blocked proclib.

Member HASP - HASP is invoked when the operator types the OS START command, as described in the Section 11.1 Operator's Guide, paragraph 2.2. This starts an OS Reader/Interpreter which reads the member STRTHASP that, in turn, invokes the HASP System. The BLKSIZE parameter on card 00900000 should be changed, if necessary, to agree with the blocking of SYS1.PROCLIB.

Member STRTHASP - The member STRTHASP is an OS job which, when read and executed, invokes the HASP System. For MFT Systems, the partition specified on card 01060000 should be changed if HASP will not reside in partition zero (P0). For MVT Systems, the region size on card 01020000 should be changed to a size sufficient to contain the HASP load modules, whose size is given by the link edit described under 10.1.4.3.

The DD card 01040000 should refer to the cataloged HASP overlay data set produced during the build step described under 10.1.4.3. A STEPLIB DD card may be added to STRTHASP, if the HASP load modules do not reside in SYS1.LINKLIB.

Others versions of the STRTHASP member may be constructed which invoke alternate HASP Systems, for purposes of changing device configuration, HASP features, etc. If these are placed in SYS1.PROCLIB under other member names, they may be invoked by using the keyword ",JOB=membername" in the initial operator START HASP command.

Member HOSRDR - The member HOSRDR is used by HASP to invoke the single OS Reader/Interpreter necessary to send jobs to OS for execution. Two versions are given in the sample job, but only one should be installed. Either version may be used with MVT but only the one identified by comments as STD RDR may be used with MFT.

In both versions of HOSRDR, the EXEC PARM field may be modified if desired, however, the "SSSSSSSS" field must not be modified from the specification "SPOOL   ". Also, the DCB field of the IEFRDER DD statement must not be modified. The IEFDATA statement may be modified to fit installation requirements, but this will have effect only if the HASPGEN parameter &OSINOPT=YES and a DD * or DD DATA card with DCB parameters is encountered in an input stream read by HASP.

If the ASB version of HOSRDR is used in an MVT System, the fixed core requirement for the OS R/I is reduced from approximately 50K to 16K. However, the ASB Interpreter will dynamically acquire a region (82K in the sample) when HASP sends it a job for OS execution. The last character in the ASB Reader EXEC PARM field, called "K", must be removed if using OS Release 18.

Member HOSWTR - The member HOSWTR is needed only if the HASPGEN parameter &WTRPART is not set to "*". However, it should be installed even if unused so that &WTRPART can be later changed without requiring installation then.

HASP uses its own module HASPWTR as an attached task, or it uses an OS Output Writer invoked by the member HOSWTR, to retrieve OS System Messages from SYS1.SYSJOBQE at the end of job execution.

10.2.2.3 Install HASP Program

The HASP Program consists of one primary load module made up of resident CSECTs from each of ten object modules, two other smaller load modules each from a single object module, and several overlay CSECTs taken from some of the above object modules. Each overlay CSECT exists as a single record in a sequential data set on a direct access device, during HASP operation.

The three step sample job HASPHASP shows how the above components of the HASP Program are constructed from the object decks produced by HASPGEN. The first step simply scratches the overlay data set to be later allocated and built.

The second step executes a utility called HASPOBLD whose primary input is ten object modules from SYS1.HASPOBJ as shown. The overlay csects are written to SYS1.HASPOLIB and all references to them in other overlays or in resident CSECTs are resolved. Resident CSECTs are written to the SYSLIN DD temporary data set as input to the third step.

The third step uses the OS Linkage Editor to resolve all external references between resident CSECTs and produce the primary load module, HASP. The two smaller load modules, HASPBR1 and HASPWTR, are also produced from their respective object modules.

It must be remembered that the three load modules and the overlay data set produced by this job belong together and should be invoked as a single entity by the proclib member STRTHASP, as described under 10.1.4.2. Load modules <u>must not</u> be used with overlay data sets produced by different executions of this job, etc.

All uses of ZZZZZZ in the sample job as a volume label should be changed to the volume of the overlay data set, which may be any direct access volume including one of the SPOOL volumes. The data set should be considered a high activity system data set just like SYS1.SVCLIB and placed accordingly for optimum performance. Space allocation must be a single extent. The example shows space for 50 records of 1024 bytes, a comfortable quantity for an unmodified HASP System with HASPGEN parameter &OLAYLEV set for maximum overlay.

If it is desired to execute HASP in a hierarchy storage environment, appropriate changes should be made to the LKED step. "HIAR" should be added to the PARM field and HIARCHY control cards should be added to the input prior to the first NAME card, to control the location of various resident CSECTs. Consult documentation of the OS Linkage Editor for more details.

Any CSECT which is programmed for overlay (third character of name is a "$") may be changed from resident to overlay or vice versa during execution of HASPOBLD, by reading control cards from the SYSIN DD file (shown as empty in the sample). The CSECT name is punched in column 1 of a control card, beginning with "HA$". The fourth character is punched "O" to make the csect overlay, or "P" to make it resident. Fifth and following characters are taken from the CSECT name as given in the appropriate assembly External Symbol listing. If a CSECT is being made overlay, a priority number in the range 0-15 may be punched beginning in column 16, to change the priority.

An information listing is produced by HASPOBLD. Any control cards are listed first. Then each "HA$" CSECT name is listed, with its OCON or relative position in the overlay supervisor reference table. For actual overlay CSECTs, the relative and absolute record address is given, and the priority for use of overlay resources. The CCHHR is especially useful when using IMASPZAP to inspect or change a particular overlay CSECT on direct access.

Self-explanatory error messages "TOO LONG", "DUPLICATE", or "UNDEFINED" may be produced with any listed CSECT name. They should not occur unless erroneous user modifications to HASP have been made. Too long CSECTs are truncated to 1024 bytes. This condition may be temporarily circumvented by making the CSECT resident by use of a control card as described above. Duplicate CSECTs are ignored. The first copy encountered in HASPOBLD input is used.

If object module input to HASPOBLD causes overflow of any internal tables, the program will terminate with a U0101 ABEND after printing the last card read.

10.2.2.4 Allocate SPOOL Direct Access Space

For direct access space, HASP requires one or more volumes whose volume serial numbers begin with the characters SPOOL. One and only one of these volumes must be labeled SPOOL1. Each SPOOL volume must have a data set named SYS1.HASPACE; HASP will use the first extent of this data set for SPOOLing space. SPOOL volumes may reside on any combination of direct-access device types except 2321. HASP sets up an individual parameter list for each SPOOL volume, thus insuring full use of all allocated space.

It is strongly recommended that each SPOOL volume be entirely devoted to HASP usage. To allocate other, frequently-referenced data sets on a SPOOL volume would degrade the efficiency of HASP's direct-access allocation algorithm. The sample job HASPOOLS shows full-volume allocation; it assumes one-track VTOCs on cylinder 0, track 1. If full-volume allocation is used, the following comments in this section may be ignored.

If the installation requires that other data sets be allocated on a SPOOL volume, a simple example will show how to allocate the SYS1.HASPACE data set so it contains no dead space. HASP's unit of direct-access allocation is the track group; the number of tracks in a track group is obtained by dividing the total number of tracks on a volume by the number &NUMTGV (number of track groups per volume). For example, the number of tracks for a 2311 volume is 2000 (regardless of the size of the SYS1.HASPACE allocation); if &NUMTGV was set to 500 at HASPGEN time, the number of tracks per track group is 2000/500 = 4. HASP will use only those track groups that fall completely within the SYS1.HASPACE allocation; therefore, an improperly allocated SYS1.HASPACE could have dead space at its beginning and end.

For allocation, use the JCL specification

SPACE=(ABSTR,(quantity,address)).

To allocate any SPOOL volume but SPOOL1, use both "quantity" and "address" as integral multiples of number of tracks per track group. For example, specify SPACE=(ABSTR,(1000,20)) if number of tracks per group is 4.

To allocate SPOOL1, follow the above procedure, but add 2 to "quantity" and subtract 2 from "address". HASP uses the first two tracks of the SPOOL1 allocation for checkpoint information. For example, specify SPACE=(ABSTR,(1002,18)). This would allocate the 1002 tracks beginning with track 18 and ending with track 999. HASP would use tracks 18 and 19 for checkpoint information; it would use the 245 track groups beginning with track group 5 (which starts on track 20 and extends through track 23) and ending with track group 249 (which starts on track 996 and extends through track 999). It would mark the other 255 track groups on this 2311 as permanently unavailable for allocation.

10.3 GENERATING HASP REMOTE TERMINAL PROGRAMS (RMTGEN)

This section describes the process of generating the HASP remote terminal programs described in the HASP Remote Terminal Operator's Guides.

10.3.1 HASPGEN Preparations For RMTGEN

HASPGEN inserts the RMTGEN procedure into the central operating system's SYS1.PROCLIB and builds appropriate members of the HASP libraries SYS1.HASPMOD and SYS1.HASPSRC. These data sets along with the procedure required for RMTGEN should be retained in the system for (1) the initial HASP Remote Terminal Program generation run, and (2) subsequent Batch HASP Remote Terminal Program generation runs. Table 10.3.1 lists the data sets and members required for the above generation runs.

Each new HASPGEN will recreate the HASP libraries and will require that new Remote Terminal Programs be regenerated when any one of the following conditions exist:

1. Official HASP modifications are used in updating the remote terminal program source decks on SYS1.HASPSRC (see Section 10.2 - Modifying the HASP SYSTEM).
2. Installation HASPGEN parameters are changed which affect the HASP remote terminal interface (see Section 7).
3. Local modifications are made to HASP and/or the Remote source programs which affect the remote terminals.

Table 10.3.1 - RMTGEN Data Sets

DSNAME	DSORG	MEMBERS	DESCRIPTION
SYS1.PROCLIB	PO		Systems Procedure Library
		RMTGEN	RMTGEN procedure
SYS1.HASPMOD	PO		HASP Load Module Library
		RMTGEN	RMTGEN main module
		GENRMT	RMTGEN source deck preparation and update module
		EXRMTGEN	HASPGEN RMTGEN executor module
		LETRRIP	Post-processor for 1130 remote terminal programs
		SYS3CNVT	Post-processor for System/3 remote terminal programs
SYS1.HASPSRC	PO		HASP System Source Library
		HRTPOPTS	HASP Remote Terminal Standard Options
		HRTPB360	Source deck for HASP 360 and M20 BSC Remote Terminal Programs
		HRTPSM20	Source deck for HASP M20 STR Remote Terminal Programs
		HRTPLOAD	Source deck for HASP 1130 BSC loader
		HRTP1130	Source deck for HASP 1130 BSC Remote Terminal Programs
		HRTPSYS3	Source deck for HASP System/3 BSC Remote Terminal Programs

All named data sets must be cataloged in the System Catalog. The initial RMTGEN run will use data sets SYS1.UT1,UT2,UT3 allocated for HASPGEN.

Installations should create and maintain RMTGEN option decks for the purpose of recreating the revised remote terminal programs when necessary after each new HASPGEN. (Note RMTGEN runs may be required even though no changes to the RMTGEN option decks are required.)

10.3.2 Initial HASP Remote Terminal Program Geneneration Run (performed as part of HASPGEN)

If CPU remote terminals are indicated in the HASPGEN parameters, the job named HRMTGEN will type the message "PLACE RMTGEN OPTIONS IN UNIT XXX AND REPLY 'GO', OR REPLY 'CANCEL' ". XXX is the address of the OS allocated 2540 card reader attached to the system. The operator should make sure the named 2540 card reader is not being used for any other function, i.e., HASP reader; clear any cards remaining in the reader; load the reader with RMTGEN options for all desired remotes; and reply, "GO" using the OS/360 reply format. If no remote generations are desired initially the operator should reply, "CANCEL".

10.3.3 Batch HASP Remote Terminal Program Generation Run

RMTGEN runs may be made as a normal batch stream job. Figure 10.3.2 shows an example job stream for a Batch RMTGEN. The user options and control cards are the same as for an initial RMTGEN run.

10.3.4 RMTGEN PROGRAM EXECUTION

RMTGEN expects its input stream to contain one or more remote terminal program descriptions. Each terminal program is described by card entries in the following order:

1. HASP Remote terminal program identification card.
2. User RMTGEN option cards.
3. $.UPDATE control card (optional).
4. Update cards if $.UPDATE card is used.
5. $.RMTEND end of remote description.

The above description format is repeated for each terminal to be generated. Descriptions do not affect any following descriptions either in the current run or succeeding runs.

The following procedures are followed in the generation of each HASP Remote terminal program.

1. RMTGEN reads the card input stream for the remote terminal program identification, selects the appropriate STANDARD OPTIONS list for the desired remote terminal program, and prints the default values on the SYSOUT=A device.

Figure 10.3.2 Example of Batch RMTGEN Run

```
//RMTGENJB JOB (0000,0000),'GEN REMOTE PROGRAMS',MSGLEVEL=1
//JOBLIB    DD DSNAME=SYS1.HASPMOD,DISP=SHR
//RMTGEN  EXEC RMTGEN
//RMTGEN.OPTIONS DD *
$.RMTM20,2
&RDEV(1)=2560
&RADR(1)=2
&UDEV(1)=2560
&UADR(1)=2
&WDEV(1)=2152
&NUMTANK=5
$.RMTEND
$.RMT360,3
&CMPTYPE=3
&PDEV(2)=1403
&ADAPT=030
&WADR=009
&NUMTANK=7
&CORESIZ=16
$.RMTEND
/*
```

2. RMTGEN reads the overriding options from the card input stream and changes the current values. Overriding options are printed on the SYSOUT=A device as they are encountered. (See Section 7 for RMTGEN option specifications.)

3. When \$.UPDATE or \$.RMTEND is encountered, the remote terminal program source deck is copied to a scratch data set (ddname= SYSIN) for the assembler. During the transfer the final options as specified are used to update the source. If update is specified, data from the card input stream will be used to modify the source deck.

4. After the update the assembler is invoked to assemble the remote terminal program and, except for 1130 and System/3 programs, punch self loading object decks on the SYSOUT=B data set. 1130 or System/3 assembly places the object deck on a scratch data set.

5. On return from the assembler, if the program is for the 1130 or System/3, RMTGEN invokes a post-processor (LETRRIP or SYS3CNVT) which creates a load deck image on the SYSOUT=B data set. See 10.3.6 for further actions necessary for System/3.

6. If more cards are in the card input stream RMTGEN repeats the above procedures.

All listings produced by RMTGEN and the assebler will have the remote terminal SIGN-ON identification number at the top of each page. With the exception of loader bootstrap cards, all object deck cards will have the identification number punched in columns 75-76.

10.3.5 RMTGEN Input Card Specifications

RMTGEN accepts four basic input card groups. (1) RMTGEN control cards, (2) User options, (3) Update control cards, (4) Update cards.

RMTGEN Control Cards

CARD FORMAT:	Col.	1- 2	$.	control card identification
	Col.	3-71	operands	variable length separated by comma with no blanks allowed. (last operand must be followed by blank)
	Col.	73-80	ignored	

The first card of a Remote terminal program description is the HASP Remote terminal program identification card. It serves two functions:

1. Selects the appropriate standard options group and source member from the library.
2. Sets the remote terminal identification number.

CARD FORMAT	$.name,n	where:	name=the name specified in Table 10.3.3 for the remote terminal program to be generated.
			n=1 or 2 digit terminal number followed by blank.

RMTGEN has two additional control cards:

$.UPDATE which sets the update mode and causes following cards to be used to modify the remote program source deck for the current generation description.

$.RMTEND which signals the end of the remote generation description.

USER OPTIONS

CARD FORMAT:	Col. 1-n Name=value	where:	name = a legal option specified in the appropriate remote terminal program options section. (see section 7).
			value = a character string of up to 17 characters - ending in blank. Blanks must not appear anywhere on the card except after the value.

User options may appear in any order after the Remote terminal program identification card.Each option may occur more than once. The last value for each option overrides previous values and is used in generating the remote terminal program. See Section 7 for default option values.

UPDATE CONTROL CARDS

CARD FORMAT:	Col. 1-2	./	control identification
	Col. 10-14	verb	DELET for delete source cards indicated ENDUP for terminate update

Col.	16-23	serial 1	starting card serial number (DELET only)
	24	,	(DELET only)
	25-32	serial 2	ending card serial number (DELET only)

Update control cards may be used only during an update run i.e. after $.UPDATE card. The DELET card is used to delete one or more source cards from the source deck for the described remote terminal program as it is being prepared for the assembler. The DELET card may be mixed with insertion and replacement cards containing new source statements for the assembler. All library source cards starting with serial 1 through and including serial 2 will be omitted from the assembler input source. ENDUP terminates the remote terminal program description. It may be replaced by $.RMTEND which also serves this function.

UPDATE CARDS

Update cards are assembly language source cards and follow the format described in the OS/360 assembler manuals. Each card may be serialized in cols. 73-80 or may have all blanks in 73-80. Cards with blank serials will be inserted immediately in the source deck after the last serialized input card or, if following a DELET control card, in place of the deleted source cards. Serialized cards will replace current source program cards if the serial numbers are equal to existing source cards or will be inserted in the source deck in the appropriate location based on the serial number.

All serialized input (including update DELET cards) must indicate ascending order serial numbers.

10.3.6 System/3 96-Column Card RMTGEN Output

As described under 10.3.4, RMTGEN for System/3 invokes the post-processor SYS3CNVT to produce the System/3 load deck image on the SYSOUT=B data set. The cards thus created are 80-column cards which, if routed (by use of a /*ROUTE card or the $R operator command) to a System/3 Remote Terminal utilizing the System/3 Starter System, will be punched as full 96-column System/3 load mode cards. They may also be punched locally or remotely as 80-column cards together with the punched outputs of other RMTGENs and later be separated and routed to a System/3 Starter System as the punched output of an 80/80 card-to-punch job. The IBM data set utilities IEBPTPCH or IEBGENER might, for example, be used. See the HASP System/3 Operator's Guide for a System/3 Starter System description.

System/3 96-column load mode cards must be punched as described above in order to use the output of a RMTGEN on a System/3. 80-column cards are _not_ loadable on a System/3, even if the supported RPQ 1142 card reader is attached.

Instead of the System/3 Starter System, any HASP System/3 Remote Terminal Processor program generated with the option &S396COL set to 1 may be used to punch RMTGEN output routed to a System/3 as described above.

Table 10.3.3 - RMTGEN Terminal Program Identification Cards

HASP Remote Terminal Processor program for	Terminal Program Identification Card (1st card of each remote description)
360/20 STR	$.HRTP,n
360/20 BSC	$.RMTM20,n
360/25, 30, 40, etc.	$.RMT360,n
1130 Loader	$.RTPLOAD,n
1130	$.RTP1130,n
System/3	$.RMTSYS3,n

n= remote SIGNON number

10.4 REMOTE GENERATION FOR NON-HASP USERS

This section outlines the procedures required to generate HASP remote workstation programs without installing the complete HASP System.

PREPARATION - The remote generation (RMTGEN) process requires creation of appropriate data sets as discussed in Section 10.3.1 of this manual. The requirements may be satisfied using the following procedures:

1) Allocate and catalog the data sets:
   SYS1.HASPMOD - for HASPGEN and RMTGEN load modules
   SYS1.HASPSRC - for HASP and workstation source decks
   SYS1.UT3 - for Linkage Editor utility data set
   Refer to Figure 10.1.2 - Sample Job to Catalog Data Sets for HASPGEN.

2) Mount the HASP distribution tape on an appropriate drive and start a reader to the tape. DO NOT allow the jobs to begin executing. (The format and blocksize of the tape is listed in the front of this manual).

3) Cancel all jobs read in from the tape except the first job (job name HASPGEN).

4) Allow the HASPGEN job to execute. This will cause the required workstation source decks, RMTGEN object modules, and RMTGEN procedures to be added to the system.

5) The HASPGEN job will request that the operator enter modifications to the default options (see section 10.1.4 - Standard Complete HASPGEN Process). The remote workstation programs are dependent upon the following two HASPGEN options which are described in section 7 of this manual.

   &TPBFSIZ
   &MLBFSIZ

   The value of &MLBFSIZ is the maximum size record which may be transmitted over the communication line. This parameter must be set to the size which has been specified at the central CPU with which the workstation is to communicate.

   If official modifications are required for the remote workstation programs, these modifications should be inserted into the 2540 card reader behind the option modification cards and the UPDATE card as described in section 10.1.3 of this manual.

When the HASPGEN job completes successfully the data sets required are ready for the remote generation RMTGEN process.

EXECUTING RMTGEN - Upon completion of the HASPGEN job one or more RMTGEN jobs may be submitted in accordance with section 10.3.3.

11.0 OPERATOR'S GUIDES

This section consists of the various operator's guides needed for the efficient operation of the various HASP components. Each operator's guide is a self-contained package, capable of being separated from the rest of the documentation and used as a teaching aid for operator classes and/or for operator reference while operating the respective components.

(The remainder of this page intentionally left blank.)

T H E

H A S P

S Y S T E M

OPERATOR'S GUIDE

TABLE OF CONTENTS

PAGE

INTRODUCTION

HASP is a program which, when started by the operator, assumes control of selected devices and portions of the Operating System (OS) for the purpose of managing the subsequent flow of jobs submitted for execution. Under normal processing, jobs flow through five distinct major functions of HASP as follows:

1. INPUT - Jobs are read into the system: Each job, made up of JOB CONTROL LANGUAGE (JCL) and optional input data cards, enters the system and is saved on direct access storage (SPOOL volumes) for later high speed retrieval.

2. EXECUTION - Jobs are submitted to OS for execution: As each job is selected for execution, the JCL cards are retrieved and submitted to an OS READER/INTERPRETER for initiation by OS. During execution, each job is monitored; input data is provided and print/punch data created by the job, along with SYSTEM messages, is saved on the SPOOL volumes for later output.

3. PRINT - Print output for jobs is printed: The SYSTEM messages and print data sets created during execution are printed.

4. PUNCH - Punch output for jobs is punched: The punch data sets created during execution are punched.

5. PURGE - Jobs are removed from the system: Upon completion of all processing required for a job, the SPOOL volume space and all HASP resources associated with the job are made available for re-use.

Although each job entering the system passes sequentially through each function, one function at a time, all HASP functions may run concurrently when sufficient jobs are available for processing.

PRIORITY QUEUEING AND SCHEDULING

As each HASP function completes processing a job, the job is placed in a queue in order of HASP scheduling priority along with other jobs to wait for the next function. A new job to process is then selected from the queue of eligible jobs. Since jobs are in priority order on the queue, high priority jobs will be selected for processing in preference to lower priority jobs. The net effect is that high priority jobs will spend less time in the system than low priority jobs.

To illustrate HASP processing of jobs, the following example traces a job through the system:

Job A enters the system and is assigned HASP job number 100. Jobs 1 through 99 have entered the system previously and are being processed by other functions, queued for processing, or have been deleted from the system. Assuming that job 100 is placed in the Class A execution queue along with jobs 97, 98, and 99 and is highest priority, the HASP initiator will select job 100 for OS execution when the next Class A job is selected.

Job 100 is placed in the print queue upon completion of execution. Again assuming that the queue contains jobs 70, 71, 73, 80, and 92 and job 100 is highest priority, job 100 will be selected for printing when the printer is free from processing the previous job. After being printed, job 100 is then queued for punch. If the punch queue is empty and the punch is available, the job will be immediately selected for punching. After all punching for job 100 has completed, the job is then queued for purging and, when selected, is removed from the system.

1.0 HASP OPERATOR COMMANDS

Through the use of HASP operator commands the operator may communicate with the HASP SYSTEM for the purpose of displaying information, controlling the flow of jobs within the system, and controlling HASP SYSTEM facilities which are used in processing of jobs. Each HASP command falls into one of the following categories:

1. JOB QUEUE COMMANDS - Commands which search the HASP job queue and display or alter the status of jobs without regard for the job identity.
2. JOB LIST COMMANDS - Commands which search the HASP job queue and display or alter the status of jobs based upon the identity of the job(s).
3. MISCELLANEOUS JOB COMMANDS - Job commands which apply to a single job by identity.
4. DEVICE LIST COMMANDS - Commands which control the HASP peripheral devices.
5. SYSTEM COMMANDS - Commands which control the status of the HASP SYSTEM or the submission of jobs to OS/360 for execution.
6. MISCELLANEOUS DISPLAY COMMANDS - Commands which provide informative responses but do not belong to the other categories.
7. REMOTE JOB ENTRY COMMANDS - Commands associated almost exclusively with HASP remote job entry.

The following sections provide sufficient information for operator control of the HASP SYSTEM for that time period after the initial response to the HASP request for initialization options.

1.1 ENTERING HASP COMMANDS - GENERAL

HASP commands have the following form:

$verb operand1,operand2...,operandn

Where:

$	=	HASP command identification character--all commands to the HASP SYSTEM start with the $ character.
verb	=	HASP command verb--a single character verb which describes the general action which is to be taken (see TABLE 1.1.1). A longer form of the verb may be used which is partially compatible with former versions of the HASP SYSTEM (see TABLE 1.1.2).
operands	=	HASP command operands--operands are used to modify the verb of the command or identify the job or system facility to be acted upon. Commas are used to separate operands when more than one operand is used.
NOTE:		If more operands are entered than the command is designed to handle, the additional operands will either be ignored or be concatenated to the last acceptable operand and handled as one.

The HASP command structure allows for a great amount of flexibility in entering the text of the command. The following rules apply:

1. FOR TEXT OUTSIDE PAIRED APOSTROPHES:

   A. All alphabetic characters may be entered in upper or lower case.

   B. Blanks may be inserted at any point in the command after the initial $ for operator convenience.

   C. Apostrophes may appear in the text of the command as a text character; however, each apostrophe text character must appear in duplicate.

2. FOR TEXT INSIDE PAIRED APOSTROPHES:

   All characters must appear as required by the individual command. Text apostrophes must appear in duplicate.

3. Key words for operands may, for the most part, be misspelled. It is only necessary to enter enough information to identify the job or facility desired.

The following examples illustrate the above rules:

1. $r all, rmt 4, local
   $RALL,RMT4,LOCAL

2. $dm4,'If your job''s output is deleted, resubmit'
   $DM4,'IF YOUR JOB'S OUTPUT IS DELETED, RESUBMIT'

3. $a all or $a a
   $AA

NOTE: The first line of each example represents the operator's input. The second line represents the internal meaningful representation with the first character of each operand underlined.

TABLE 1.1.1   HASP COMMAND VERBS

COMMAND	DEFINITION	OPERAND TYPES
$A	RELEASE	ALL JOBS OR SPECIFIC JOBS
$B	BACKSPACE	PRINTERS
$C	CANCEL	DEVICE FUNCTIONS OR JOBS
$D	DISPLAY	DISK, UNITS, LINES, REMOTES, MESSAGES, JOBS, QUEUES, ACTIVITY, INITIATORS, OR OUTSTANDING REQUESTS
$E	RESTART	DEVICE FUNCTIONS
$F	FORWARD SPACE	PRINTERS
$H	HOLD	ALL JOBS OR SPECIFIC JOBS
$I	INTERRUPT	PRINTERS
$N	REPEAT	DEVICE FUNCTION
$P	STOP (AFTER CURRENT FUNCTION)	DEVICE, INITIATOR, SYSTEM, OR JOB
$R	ROUTE OUTPUT	BY ROUTING GROUP OR JOB
$S	START	DEVICE, INITIATOR, OR SYSTEM
$T	SET	DEVICE, INITIATOR, JOB, OR SYSTEM JOB NUMBER BASE
$Z	HALT (IMMEDIATE)	DEVICE

TABLE 1.1.2 ALTERNATE HASP COMMAND VERBS

ALT FORM	SHORT *	SAMPLE INPUT - comments
$ALTER	$T	$ALTER JOB4,P=+4 - up JOB 4 priority by 4
$BACKLOG	$DQ	$BACKLOG - display number of queued jobs
$BACKSPACE	$B	$BACKSPACE PRT1 - backspace printer 1
$DEFINEI	$TI	$DEFINE I1,ABC - set initiator classes
$DEFINE	$DI	$DEFINE - list all initiator status information
$DELETEJ	$PJ	$DELETE JOB 4 - purge JOB 4 after current activity
$DELETE	$C	$DELETE PRT2 - cancel current output on PRINTER 2
$DISPLAY	$D	$DISPLAY DISKS - $DISPLAY UNITS - $DISPLAY RMTS -
$DRAIN	$P	$DRAIN I - stop all further execution $DRAIN I2 - stop further execution with INITIATOR 2 $DRAIN PRT1 - stop printing on PRINTER 1 after current job
$LIST	$T	$LIST CON1,15 - all only messages classes above 15
$LOCATE	$D	$LOCATE JOB 4 - display job information about JOB 4
$HOLD	$H	$HOLD ALL - prevent all jobs from beginning activity $HOLD JOB 4 - prevent JOB 4 from beginning activity
$IDJ	$D	$IDJ JOB 3 - display job information about JOB 3 $IDJ 'ABCJOB' - display job information about all jobs with name 'ABCJOB'
$RELEASE	$A	$RELEASE ALL - release all jobs in queue if held by $HOLD ALL $RELEASE JOB 6 - release JOB 6
$REPEAT	$N	$REPEAT PRT1 - repeat the current function on PRINTER 1
$RESTART	$E	$RESTART LNE3 - abort current activity and start over
$ROUTE	$R	$ROUTE ALL,RMT3,LOCAL remote output
$SETJOBNO.TO	$TJ	$SET JOB NO. TO 4 - set system generated job number base
$SPACE	$T	$SPACE PRT1,C=1 - single space each line on printer until next job

TABLE 1.1.2 ALTERNATE HASP COMMAND VERBS (continued)

ALT FORM	SHORT *	SAMPLE INPUT - comments
$START	$S	$START - start job processing $START LNE3,QXZ3 - start line with password $START TPE1,180 - start input tape using unit 180
$STATUS	$DA	$STATUS - list current activity
$STOP	$Z	$STOP PRT1 - suspend operations until $START

* The short form listed in this table is the character string to which the ALTERNATE FORM is converted. Thus verbs such as: $IDJ, $LOCATE, $DISPLAY are all converted to $D and are therefore equivalent.

The syntax of each command is checked <u>after</u> the short form has been generated. Therefore the operator should attempt to use the short form of the command in preference to the long form.

TABLE 1.1.3 HASP COMMAND SUMMARY

COMMAND	REMOTE SOURCE	COMMENTS
JOB QUEUE		
$AA	NO	Release all jobs
$DA	OK	Display active jobs
$DF	OK	Display number of queued jobs awaiting forms
$DN	OK	Display job information on queued jobs
$DQ	OK	Display number of queued jobs
$HA	NO	Hold all jobs currently in the system
JOB LIST		
$A job list	IF OWNER	Release specified job(s)
$C job list	IF OWNER	Cancel specified job(s)
$D job list	IF OWNER	Display job information on specified job(s)
$H job list	IF OWNER	Hold specified job(s)
$P job list	IF OWNER	Stop specified job(s) after current activity
MISCELLANEOUS JOB		
$D 'job name'	OK	Display job information on job(s) with OS job name
$T Jx...j,operand	NO	Set job class or priority - c=class or p=priority
$T Jx...j	NO	Set HASP internal job number
DEVICE LIST		
$B device list	IF OWNER	Backspace device(s)
$C device list	IF OWNER	Cancel current function on device(s)
$E device list	IF OWNER	Restart current function on device(s)
$F device list	IF OWNER	Forward space device(s)
$I device list	IF OWNER	Interrupt the current function on printer(s)
$N device list	IF OWNER	Repeat current function on device(s)
$P device list	IF OWNER	Stop the device(s)
$S device list	IF OWNER	Start device(s)
$T device	IF OWNER	Set device
$Z device list	IF OWNER	Halt device(s) (suspend operation)
SYSTEM		
$DI	YES	Display initiator(s), classes and status
$PI	NO	Stop initiator(s) after current activity
$SI	NO	Start initiator(s)
$TI	NO	Set initiator classes
$P	NO	Stop system
$PHASP	NO	Terminate HASP job
$S	NO	Start system

Table 1.1.3 HASP COMMAND SUMMARY (continued)

COMMAND	REMOTE SOURCE	COMMENTS
MISCELLANEOUS DISPLAY		
$DD	YES	Display Direct Access devices
$D line n	YES	Display HASP remote job entry line
$DR	YES	Display outstanding reply identification
$DRM	YES	Display devices on remote(s)
$DU	YES	Display local unit record devices
REMOTE JOB ENTRY		
$DM	YES	Display message
$R	IF OWNER	Route output for specified job or device group to another device group

Only the characters required to recognize the uniqueness of each command are defined in this table. For complete entry format, see the individual command description.

1.2 COMMAND DESCRIPTION SYNTAX

The following conventions are used to describe the format requirements and options of the various HASP commands:

1. Upper case characters - the exact characters should be used when selecting the option

2. Lower case keyword - appropriate text should be inserted to replace the keyword

3. Braces { } - one of the options enclosed by the braces must be selected, unless part of an unselected option

4. Brackets [ ] - one of the options enclosed by the brackets may be selected

5. character string x... - the character preceeding the x is sufficient to identify the option and any alphabetic characters following are optional; i.e., Jx... indicates that the single character "J" is sufficient to identify the operand, however, "JOB", "JOBS", or any other alphabetic character strings will be accepted as long as they begin with the character "J".

6. character(s) j or jj - a job number is desired

7. character(s) r or rr - a routing code is desired (routing codes refer to local [r=0] or remote terminal [r=1 to &MAXRJE] output routing of job print or punch)

8. character n - a device number is desired

9. character(s) j-jj or r-rr - a range of numbers is desired, indicating the ability of the command to operate on one or more jobs or routing codes

1.3 STANDARD RESPONSES

It is a basic philosophy of the HASP System to display a response to each HASP command entered during normal job processing. In keeping with this philosophy the processing of each command entered into the HASP System results in one or more responses, which are displayed upon the requesting console or, in the event of card input, upon an associated console device.

OK RESPONSE

The response "OK" is used in many commands to signify that action requested has been taken or that the request has been noted and action will be taken by the system when appropriate. The "OK" response, when issued, is the last message issued as a direct response to the operator; however, many commands will cause action by components of the system which will issue information messages to the central operator console devices.

JOB INFORMATION RESPONSE

Many HASP commands will display job information as a response to the operator. The format of the response is as follows:

1. Jobs queued and waiting for processing:

$$\text{JOB j [name] AWAITING}\left\{\begin{array}{l}\text{EXEC class}\\ \text{PURGE}\\ \text{PRINT r}\\ \text{PUNCH r}\end{array}\right\}\text{PRIO priority}\left[\begin{array}{l}\text{HOLD}\\ \text{PURGE}\\ \text{DUPLICATE}\end{array}\right]$$

2. Jobs being processed (active):

$$\text{JOB j [name]}\left\{\begin{array}{l}\text{EXECUTING class}\\ \text{IS PURGING}\\ \text{ON device name}\end{array}\right\}\text{PRIO priority}\left[\begin{array}{l}\text{HOLD}\\ \text{PURGE}\end{array}\right]$$

Where:

j	=	the HASP assigned job number
name	=	the OS job name assigned by the programmer (displayed only if requested by the installation at HASPGEN time)
class	=	the job class specified on the job card or set by the operator with the $T JOB command
r	=	the remote terminal to receive the output for which the job is queued (if r=0 the job is queued for local printing)
device name	=	the device that is ready, printing or punching data associated with the job. If the operator has repeated the output of a job, the lowest numbered device will be listed.
priority	=	the HASP queueing priority
HOLD	=	the job is in HOLD status and must be released to continue to flow through the system
PURGE	=	the job has been flagged for purge and will be deleted from the system
DUPLICATE	=	the job is waiting for OS execution and another job is currently executing with the same OS job name

Examples:

```
JOB 12  JOHNSJB     EXECUTING A PRIO 9
JOB 13  JOHNSJB     AWAITING EXEC B PRIO 8 DUPLICATE
JOB 14  PUNCHJOB    ON RM1.PU1 PRIO 7
JOB 13  TESTOUT     ON PRINTER1 PRIO 8
JOB 15  ASMJOB      AWAITING PRINT 1 PRIO 6
JOB 16  UNIQUE      AWAITING PUNCH 0 PRIO 6
```

STANDARD ERROR RESPONSES

The following standard messages will be returned in response to invalid SYNTAX in command entry:

1. xxxxxxxx INVALID COMMAND - The command identified by the eight characters displayed was not found in the HASP command verb table. No action has been taken.

2. xxxxxxxx INVALID OPERAND - The input stream identified by the eight characters displayed was not recognized as a valid operand. With exception of device list commands no action has been taken. In the case of device list commands action has been taken on operands <u>preceding</u> the INVALID OPERAND.

1.4 JOB QUEUE COMMANDS

blank doesn't have to be there?

Definition:	$A Ax...	RELEASE ALL JOBS
Action:	Any jobs in the system held by the $HA command will be released and processing allowed	
Responses:	OK	- one or more jobs have been released
	QUEUE NOT HELD	- no jobs have been released

Examples:

1. user - $A ALL
   system - OK

2. user - $A A
   system - OK

3. user - $A A
   system - QUEUE NOT HELD

Definition: $D Ax... DISPLAY ACTIVE JOBS

Action: Job information for each active job in the system will be displayed.

Responses: Job information messages - (see section 1.3)

NO ACTIVE JOBS	- no active jobs were found
LIST INCOMPLETE	- the last job listed was removed from the HASP job queue while all HASP WTO buffers were in use.

Examples:

1. user - $D A
   system - JOB 3 ASSEMBLY EXECUTING A PRIO 5

2. user - $D ACTIVE
   system - JOB 20 LISTALL ON PRINTER 2 PRIO 6

Comments: The LIST INCOMPLETE response should be extremely rare when sufficient WTO buffers have been generated to handle the message traffic.

only applies to special routing

Definition: $D Fx...[,r-rr]

DISPLAY NUMBER OF JOBS QUEUED ON FORMS

Action: The number of jobs queued for special forms printers and special forms punches will be summarized and displayed for the local or remote workstations specified by the route codes (r-rr). If the route code ranges are not specified, only the local queues are displayed.

Responses:

jjj FORM ffff PRT rrr - one response for each form/route code combination with jobs queued for special forms printer output meaning: jjj jobs are queued for form ffff at a printer located at the local or remote station as indicated by rrr.

jjj FORM ffff PUN rrr - one response for each form/route code combination with jobs queued for special forms punch output.

Examples:

```
1.   user   - $D F
     system - 4 FORM 0030 PRT 0
              3 FORM 0132 PRT 0
              1 FORM 0011 PUN 0

2.   user   - $D F,3-4
     system - 2 FORM 6431 PRT 3
              1 FORM 7346 PRT 3
              3 FORM 0563 PRT 4
              1 FORM 7346 PRT 4
```

r = remote
all remotes given
numbers
0 = local

Definition: $D Nx... [,{r-rr | ,queue} [,queue]]

DISPLAY JOB INFORMATION ON QUEUED JOBS.

Where:

queue =	XEQ	- only jobs waiting for execution are to be displayed in order by class (A, B, C, etc.)
=	XEQ class	- only jobs waiting for execution in the designated class are to be displayed
=	PRT	- only jobs waiting for print are to be displayed in order by route code (0, 1, 2, etc.)
=	PUN	- only jobs waiting for punch are to be displayed in order by route code (0, 1, 2, etc.)
=	HOLD	- only jobs waiting for any activity and in hold status are to be displayed

Action: If routing and/or queue type restrictions are not specified, job information will be displayed for all jobs queued for execution (XEQ), print (PRT), and punch (PUN); destined for output at local and all remote terminal printer-punch unit record groups. If the routing restriction is specified in operand 2, only the jobs with output destined to the terminals designated will be displayed. If the queue type is specified in operand 2 or 3, only jobs in the selected queue with the appropriate routings will be displayed.

In addition to displaying job information, the percentage of spool disk utilization will be displayed following the search for queued jobs.

Responses:	Job information message	- (see Section 1.3)
	xx PERCENT SPOOL UTILIZATION	- the last response
	LIST INCOMPLETE	- the last job listed prior to this message was removed from the HASP job queue while <u>all</u> HASP WTO buffers were in use

Examples:

```
1.   user   - $D N,4,PRT
     system - JOB 6 PRINTJOB AWAITING PRINT 4 PRIO 6
              JOB 8 ASSEMBLY AWAITING PRINT 4 PRIO 5
              25 PERCENT SPOOL UTILIZATION

2.   user   - $D N,0-2,XEQ
     system - JOB 3 UNIQUE  AWAITING EXEC A PRIO 9 DUPLICATE
              JOB 6 JOHNSJB AWAITING EXEC A PRIO 9
              JOB 2 BILLSJB AWAITING EXEC A PRIO 8
              30 PERCENT SPOOL UTILIZATION

3.   user   - $D N,PUN
     system - JOB 6  XYZJOB AWAITING PUNCH 9 PRIO 9
              JOB 7  XYZJOB AWAITING PUNCH 9 PRIO 8
              JOB 12 JOBXYZ AWAITING PUNCH 1 PRIO 13
              JOB 15 JOBXYZ AWAITING PUNCH 1 PRIO 8
              JOB 5  JOBJOB AWAITING PUNCH 3 PRIO 10
              35 PERCENT SPOOL UTILIZATION
```

Comments: In example 1 the operator has requested that only jobs with output to remote 4 and waiting for print to be displayed.

In example 2 the operator has requested information on jobs waiting for execution with output to local, remote 1, or remote 2 devices.

Definition: $D Qx... [,{r-rr} [,queue] | ,queue]

DISPLAY NUMBER OF JOBS QUEUED

Where:

queue = XEQ - only jobs waiting for execution are to be counted and summarized in order of class (A, B, C, etc.)

= XEQ class - only jobs waiting for execution in the designated class are to be counted and summarized

= PRT - only jobs waiting for print are to be counted and summarized in order by route code (0, 1, 2, etc.)

= PUN - only jobs waiting for punch are to be counted and summarized in order by route code (0, 1, 2, etc.)

= HOLD - only jobs waiting for any activity and in hold status are to be displayed

Action:

If routing and/or queue type restrictions are not specified, the number of jobs queued for execution (XEQ), print (PRT), and punch (PUN); destined for output at local and all remote terminal printer-punch unit record groups will be displayed. If the routing restriction is specified in operand 2, only the count of jobs with output destined to the terminals designated will be displayed. If the queue type is specified in operand 2 or 3, only the count of jobs in the selected queue with the appropriate routings will be displayed.

In addition to displaying number of jobs queued, the percentage of spool disk utilization will be displayed following the search for queued jobs.

Responses: nn queue type - number of jobs in designated queue type--one line for each queue type

xx PERCENT SPOOL UTILIZATION - the last response

Examples:

```
1.   user   - $D Q,4,PRT
     system - 2 PRT 4
              25 PERCENT SPOOL UTILIZATION

2.   user   - $D Q,0-2,XEQ A
     system - 3 XEQ A
              30 PERCENT SPOOL UTILIZATION

3.   user   - $D Q,PUN
     system - 2 PUN 0
              2 PUN 1
              1 PUN 3
              35 PERCENT SPOOL UTILIZATION
```

Comments: In example 1 the operator has requested that the number of jobs with output to remote 4 and waiting for print to be displayed.

In example 2 the operator has requested that the number of jobs waiting for execution class A with output to local, remote 1, or remote 2 devices

In example 3 the operator has requested that the number of jobs waiting for punch be displayed. The response shows two jobs waiting for punch at remote 1 (route code 1), and one job waiting for punch at remote 3 (route code 3).

Definition: $H Ax... HOLD ALL JOBS CURRENTLY IN THE SYSTEM

Action: All jobs currently in the system will be placed in the HOLD status and further processing will be prevented. Any new jobs entering the system subsequent to $HA command will not be held. The $A ALL command may be used to negate the effect of the $HA command or the $A JOB command may be used to negate the effects for specific jobs.

Response: OK - all jobs currently in the system have been placed in the hold status

Examples:

1. user - $H ALL
   system - OK

2. user - $H A
   system - OK

1.5 JOB LIST COMMANDS

JOB LISTS

All job list commands accept requests for action for one or more jobs. The following format is used for entry of job list commands:

$verb Jx... j_1-jj_1,j_2-jj_2,...,j_m-jj_m

Each operand requests action upon a range of job numbers; i.e., if "1-300" were specified for an operand, action would be attempted on jobs 1, 2, 3,...300. If a single job is desired, the "-jj" may be omitted or entered with a value equal to the first value of the range. If the second value of the range is not greater than the first, only the job corresponding to the second value will be operated upon.

Limitations:

The maximum of five (5) range groups may be entered; any entries beyond operand five will be ignored.

Definition:	$A job list RELEASE SPECIFIED JOB(S)	
Action:	Specified jobs will be released from the HOLD status if held by $H ALL, $H JOB, or JCL TYPRUN=HOLD.	
Response:	JOBj RELEASED	- one response for each job released
	JOB(S) NOT FOUND	- none of the specified job(s) were found
	JOBj NOT HELD	- one response for each job indicated but not in the hold status

Examples:

```
1.  user   - $A JOB 3
    system - JOB 3 RELEASED

2.  user   - $A JOBS 4-6
    system - JOB 4 RELEASED
             JOB 6 NOT HELD
```

Comments: In example 2 job 5 was not found.

Definition: $C job list CANCEL SPECIFIED JOB(S)

Action: Specified jobs will be flagged for PURGE, if NOT in OS--execution will have its activity deleted and be queued for purging. If the job is queued for execution, or being read into the system it will have its JCL queued for print prior to purging.

Limitations: If the job is on an output device which has been repeated, multiple $C commands may be necessary to purge the job.

Response:

Job information response	- one response for each job cancelled
JOB(S) NOT FOUND	- none of the specified jobs were found

Examples:

1. user - $C JOB 7
   system - JOB 7 YOURJOB AWAITING PUNCH 0
   PRIO 7 PURGE

Definition: $D job list DISPLAY JOB INFORMATION ON SPECIFIED JOB(S)

Response:

Job information response	- one response for each specified job found in the system
JOB(S) NOT FOUND	- none of the specified jobs were found

Examples:

```
1.   user   - $D JOBS 1-10
     system - JOB 2 YOURJOB EXECUTING A PRIO 13
              JOB 3 YOURJOB AWAITING EXEC A PRIO 13 DUPLICATE
              JOB 6 ANOTHER ON PRINTER1  PRIO 12
              JOB 7 JOHNSJB AWAITING PRINT 0 PRIO 12 HOLD
```

Comments: In example 1 jobs 1, 4, 5, 8, 9 and 10 were not found.

If the $D job list command is entered from a remote terminal, only those jobs belonging to the remote will be displayed.

Definition:	$H job list HOLD SPECIFIED JOB(S)
Action:	Each specified job found in the system will be placed in the HOLD status.
Response:	Job information response - one response for each job held JOB(S) NOT FOUND - none of the specified jobs were found

Examples:

1. user - $H J4
   system - JOB 4 YOURJOB AWAITING PRINT 0
   PRIO 4 HOLD

Definition:	$P job list	STOP SPECIFIED JOB(S) AFTER CURRENT ACTIVITY

Action: Specified jobs will be flagged for PURGE and, if not active, will be queued for purging. Jobs which are active will be queued for purging upon completion of current activity. Jobs awaiting execution will be queued for printing of JCL prior to purging.

Response:

Job information response	- one response for each job which will be stopped
JOB(S) NOT FOUND	- none of the specified jobs were found

Examples:

1. user - $P J7
   system - JOB 7 JOHNSJB ON PRINTER2 PRIO 4 PURGE

1.6 MISCELLANEOUS JOB COMMANDS

Definition: $D'jobname' DISPLAY JOB INFORMATION ON JOB SPECIFIED BY OS JOBNAME

Where: 'jobname' = the OS job name appearing on the users job card enclosed by apostrophes. The name may be upper or lower case alphabetic characters, but must not contain blanks.

Limitations: This command is valid only if requested by the installation at HASPGEN time.

Response:

Job information response	- one response for each job in the system with the OS job name specified.
LIST INCOMPLETE	- the last job listed was removed from the HASP job queue while all HASP WTO buffers were in use.
jobname NOT FOUND	- the named job was not found

Examples:

```
user   - $D 'myjob'
system - JOB 4 MYJOB ON PRINTER 1 PRIO 13
         JOB 5 MYJOB AWAITING PRINT 0 PRIO 13
         JOB 6 MYJOB EXECUTING A PRIO 13
         JOB 7 MYJOB AWAITING EXEC A PRIO 13 DUPLICATE
```

Definition: $T Jx...j, { P=priority | P=+priority | P=-priority | C=class }   SET JOB CLASS OR PRIORITY

Where:

priority = a numeric value 0 through 15 which indicates the HASP queuing priority desired for the specified job.

+priority = a numeric value which is to be added to the present HASP queuing priority of the specified job.

-priority = a numeric value which is to be subtracted from the present HASP queuing priority of the specified job.

class = a single character (A,B,C---Z,0,1---9) representing the new execution class of the specified job. (Lower case characters will be made upper case.)

Action:

1. PRIORITY SETTING
   The specified job's priority will be adjusted as indicated; however, if the resulting priority is outside the range 0-15, the final priority is adjusted to 0 or 15 as appropriate.

2. CLASS SETTING
   The specified job's execution class will be set to the indicated class.

Limitation: No action will be taken on a job that is currently active.

Responses: Job information response--response for the job being set.

Examples:

1. user - $TJ4,P=14
   system - JOB 4 ANYJOB AWAITING EXEC A PRIO 14

2. user - $TJ6,C=Z
   system - JOB 6 YOURS AWAITING EXEC Z PRIO 3

Definition: $T Jx...n SET HASP INTERNAL JOB NUMBER

Where: n = the new base number for automatic job number assignments.

Action: The new base number will be set causing the next job number assignment to be "JOB n" or the first number beyond n that is not currently held by a job.

Responses: OK - indicates that the new job number base has been set.

Examples:

1. user - $TJ1
   system - OK

2. user - $TJOB100
   system - OK

Comments: In example 1 assume that jobs 1, 3 and 4 are currently in the system when the input service processors read the next job. An attempt to assign the value of "1" to the new job will fail; however, the job will be assigned the value of "2". Subsequent jobs will be assigned the value of 5, 6, 7... If, however, the jobs 1, 3 and 4 are not in the system, new jobs entering the system will receive job number 1, 2, 3, 4...

1.7 DEVICE LIST COMMANDS

Unless the format of the acceptable device list required by a command is explicitely specified, all device list commands accept entries of the following form:

$verb device$_1$,device$_2$,...,device$_n$

Each operand specifies a single device that is to be acted upon by the HASP System. The device may be specified by its full name or abbreviated name as follows:

CONSOLEn	- CONn	(abbreviation must be used)
INTRDRn	- RDIn	(abbreviation must be used)
LINEn	- LNEn	
PRINTERn	- PRTn	
PUNCHn	- PUNn	(abbreviation must be used)
READERn	- RDRn	
RMr.PRn	-	(no abbreviation)
RMr.PUn	-	(no abbreviation)
RMr.RDn	-	(no abbreviation)
TAPEn	- TPEn	

Limitations:

A maximum of five (5) operands may be specified in a single device list command. Operands which are in excess of the maximum allowed will be considered part of the fifth operand.

NOTES:

1. Device list commands generally perform operations which occur after the response to the command entered; i.e., the OK response to a device list command signifies that the command has been accepted and an attempt to perform the requested action will be made for all devices listed.

   Additional messages will be displayed on the operator's console when the action requested is either in process or has been completed as appropriate. See Messages and Codes section of this manual for the format and meanings of these messages.

2. An error response to a device list command indicates that the action requested by the previous operands will be attempted but the operand in error and all following operands will be ignored.

3. Many commands will accept operands as being valid even though the devices specified are unable to perform the function requested. Table 1.7.1 identifies the devices affected by each device list command.

TABLE 1.7.1 DEVICES AFFECTED BY DEVICE LIST COMMANDS

	$B	$C	$E	$F	$I	$N	$P	$S	$T	$Z
DIRECT ACCESS	**	**	**	**	**	**	**	**	**	**
LINE			Y				Y	Y		Y*
LOCAL READER		Y					Y	Y	Y	Y
INPUT TAPE		Y					Y	Y	Y	Y
REMOTE READER		Y					Y	Y	Y	Y
INTERNAL READER		Y					Y***	Y***	Y	Y
LOCAL PRINTER	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
REMOTE PRINTER	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
LOCAL PUNCH	Y*	Y	Y	Y*		Y	Y	Y	Y	Y
REMOTE PUNCH	Y*	Y	Y	Y*		Y	Y	Y	Y	Y
LOCAL CONSOLE								Y	Y	Y
REMOTE CONSOLE	**	**	**	**	**	**	**	**	**	**

* COMMAND ACCEPTED BUT WILL HAVE NO EFFECT
** NOT LOCATABLE
*** AUTOMATICALLY STARTED WHEN REQUIRED

Definition:	$B device [,pages / Dx...]	BACKSPACE DEVICE(S)

Where:

device	=	HASP printer device desired to perform the backspace
pages	=	the number of pages (up to 9999) to backspace (optional for the last device of the list, mandatory for the other devices)
Dx...	=	backspace to beginning of data set

Action: The designated printer will, if ACTIVE, back up the designated number of pages in the current data set and resume printing. If the beginning of the data set is encountered during the backspace process, the printer will resume printing at the beginning of the data set. If the number of pages is not specified, the count of one (1) will be assumed for the last device in the list.

Responses: OK - the specified printer(s) will be backspaced

Examples:

1. user - $B PRT1,10
   system - OK

2. user - $B PRT1
   system - OK

3. user - $B PRT1,5,PRT2

Comments: In example 1 printer 1 is to be backspaced ten pages.

In example 2 printer 1 is to be backspaced one page.

In example 3 printer 1 is to be backspaced five pages (the count for printer 1 must be specified), and printer 2 is to be backspaced one page.

Definition:	$C device list	CANCEL CURRENT ACTIVITY ON DEVICE(S)
Where:	device	= HASP reader, printer, and punch devices

Action: The current activity on the designated devices will be terminated. In the case of printer and punch devices, the highest priority job eligible for output on the device will be selected and printing or punching will resume for the new job. In case of input reader devices, the input stream will be scanned for the next valid job card and reading will continue.

Responses: OK - the activity on the specified device(s) will be cancelled.

Examples:

```
1.   user   - $C PRT1
     system - OK
```

Definition:	$E device list RESTART CURRENT ACTIVITY ON DEVICE(S)
Where:	device = HASP line, printer, and punch devices
Action:	The current activity on the designated devices will be terminated. In case of printer/punch devices the job will be returned to the appropriate print or punch queue in order of priority and made eligible for selection. In case of remote job entry lines, the HASP System will, upon completion of the current line I/O, abort all activities on the line.
Response:	OK - the specified device will be restarted
Examples:	1. user - $E PRT1,PRT2 system - OK 2. user - $E LNE2 system - OK

Definition:	$F device [,pages / Dx...]	FORWARD-SPACE PRINTER DEVICE(S)

Where:

device	= HASP printer device desired to perform the forward-space
pages	= the number of pages (up to 9999) to forward-space (optional for last device of list, mandatory for the other devices)
Dx...	= forward-space to end of data set

Action: The designated printer will, if ACTIVE, skip forward the designated number of pages and resume printing. If the end of the data set is encountered during the forward-space, printing will resume on the next data set if present. If the number of pages is not specified for the last device in the list, the count of one (1) will be assumed.

Responses: OK - the specified printer(s) will be forward-spaced

Examples:

1. user - $F PRT1,999
   system - OK

2. user - $F PRT1,DS

Definition: $I device list   INTERRUPT CURRENT ACTIVITY ON PRINTER DEVICE(S)

Where: device = HASP printer devices

Action: The current activity on the designated printer(s) will, if ACTIVE, be checkpointed and terminated. The job will be returned to the HASP SYSTEM job queue and made available for selection. Any printer selecting the job for output will resume printing the job after the backspacing of one (1) page.

Responses: OK - the specified printer(s) will be interrupted

Examples:

1. user - $I PRT1
   system - OK

Definition: $N device list REPEAT CURRENT ACTIVITY ON DEVICE(S)

Where: device = HASP printer and punch devices

Action: The current activity on the designated printer and/or punch devices will be repeated. This operation will not terminate the activity in process but will place the job back on the HASP job queue and make it available for other devices to output. Once a device has been repeated additional commands to repeat the device will be ignored until the device has completed operation on the current copy of the job output. A restart $E directed to a repeated device will have the effect of cancelling the output.

Responses: OK - the specified printer and/or punch device(s) will be repeated if the device(s) are eligible

Examples:

1. user - $N PRT1
   system - OK

DEVICE STATUS

A device controlled by the HASP System will be in one of four status conditions as follows:

ACTIVE - The device is actively performing a function.

INACTIVE - The device is available to perform a function, however, no jobs are available for the device.

DRAINING - The device is actively performing a function, but upon completion of that function will not begin a new activity.

DRAINED - The device is not performing a function and will not do so until the operator starts the device.

The operator controls the ability of a HASP device to select jobs for processing via the following commands:

$P - Stop the device
$S - Start the device

Definition:	$P device list STOP (DRAIN) DEVICE(S)
Action:	The specified devices will be prevented from starting any new activity. If a device is INACTIVE, the device will be immediately stopped (DRAINED). If a device is ACTIVE, the device status will be DRAINING and will revert to the DRAINED status upon completion of the current activity.
Responses:	OK - the device(s) have been placed in the DRAINED OR DRAINING status
Examples:	
	1. user - $P PRT1,PRT2,PUN1 system - OK
Comments:	When the device enters the DRAINED status, the HASP message "device IS DRAINED" will be displayed on the operator's console. If a VARY CPUx,OFFLINE command is to be issued to stop a CPU in the Model 65 Multiprocessor configuration, the operator must first insure that all devices accessible <u>only</u> from the CPU to be varied offline are in the HASP DRAINED status.

```
Definition:  $S ( device              ) [,additional devices] START DEVICE(S)
                ) line,password       (
                ) input tape,address  (
                ( console             )
```

Note: The explanation of this command is complex and is separated into the definitions which follow.

Definition:	$S device list	START DEVICE(S)
Where:	device = HASP card reader, printer, and punch devices	
Action:	The devices listed will be placed in the ACTIVE or INACTIVE status. If the device is INACTIVE, an attempt to select and process a job will be made. Each device started will be placed into the OS off-line status to prevent inadvertent OS allocation of an active device.	
Responses:	OK - the device(s) listed have been started	

Examples:

1. user - $S PRT1,PRT2,PUN1,RDR1
   system - OK

2. user - $S PRT1
   system - OK

Definition:	$S line,password	START DEVICE(S)

Where:

line = HASP remote job entry line device
password = 0 to 8 character security password required for remote workstation SIGNON

Action:

The specified line(s) will be started unless allocated by OS to another activity or the designated adapter is off-line. The password will be set for the line and be used to reject unauthorized terminals attempting to use the line without permission from the central installation. If the line is ACTIVE, the command has the effect of setting a new password to be used for future terminal SIGNON.

Responses:

OK	- the specified line(s) will be started
device name IN USE	- the line listed is assigned by OS
device name INVALID OPERAND	- the line listed, if spelled correctly, has not been assigned a hardware address

Examples:

1. user - $S LNE1,,LNE2
   system - OK

2. user - $S LNE1,XZQ,LNE2,XZZ
   system - OK

Definition:	$S input tape,address	START DEVICE(S)
Where:	input = HASP input tape device address = three digit hardware address for the device	
Action:	The specified HASP input tape device will be made ACTIVE and will read the input stream data from the assigned tape unit.	
Responses:	OK	- the designated tape(s) have been started
	device name IN USE	- the device is currently ACTIVE or the unit has been assigned by OS

Examples:

1. user - $S TPE1,182
   system - OK

Definition:	$S line,password	START DEVICE(S)

Where:
line = HASP remote job entry line device
password = 0 to 8 character security password required for remote workstation SIGNON

Action: The specified line(s) will be started unless allocated by OS to another activity or the designated adapter is off-line. The password will be set for the line and be used to reject unauthorized terminals attempting to use the line without permission from the central installation. If the line is ACTIVE, the command has the effect of setting a new password to be used for future terminal SIGNON.

Responses:

OK	- the specified line(s) will be started
device name IN USE	- the line listed is assigned by OS
device name INVALID OPERAND	- the line listed, if spelled correctly, has not been assigned a hardware address

Examples:

1. user - $S LNE1,,LNE2
   system - OK

2. user - $S LNE1,XZQ,LNE2,XZZ
   system - OK

Definition:	$S input tape,address		START DEVICE(S)
Where:	input	= HASP input tape device	
	address	= three digit hardware address for the device	
Action:	The specified HASP input tape device will be made ACTIVE and will read the input stream data from the assigned tape unit.		
Responses:	OK		- the designated tape(s) have been started
	device name IN USE		- the device is currently ACTIVE or the unit has been assigned by OS

Examples:

1. user - $S TPE1,182
   system - OK

Definition:	$S console	START DEVICE(S)
Where:	console = HASP console device(s)	
Action:	The specified HASP consoles will be started for all logical console classes of messages with all levels of importance (see CONSOLE SUPPORT for classes and levels of messages).	
Responses:	OK - the HASP console(s) have been started	
Examples:		

1. user - $S CON1,CON2
   system - OK

```
Definition:      $T  reader,Hx...                                    SET DEVICE
                     {printer}  ,C= {1        }
                     {punch  }      {carriage }
                                ,F= {Rx...    }
                                    {Sx...    }
                                    {Ax...    }
                                    {n        }
                                ,T= train
                                ,S= {Yx...    }
                                    {Nx...    }
                     console    {,level                }
                                {,class [,class,...]   }
                                {,Rx...                }
                                {,A=authority          }

                     CON,level,class [,class,...]
```

Note: The explanation of this command is complex and is separated into the definitions which follow.

Definition:	$T reader,Hx...	SET DEVICE
Where:	reader = HASP reader device	
Action:	The specified reader will be set to place all jobs subsequently read by the reader into the execution HOLD queue. Jobs placed into the HOLD queue may be released for execution by use of the $AJOB command. A successful $S command directed to the reader will negate the effects of the $T reader command causing the reader to revert to normal reading and queueing of jobs.	
Responses:	OK - the specified reader has been set to HOLD subsequent jobs processed by the reader.	

Examples:

1. user - $T RDR1,HOLD
   system - OK

2. user - $T RDR1,H
   system - OK

Definition:

```
$T  {printer}  ( ,C= {1        }  )  SET DEVICE
    {punch  }  (     {carriage }  )
               ( ,F= {Rx...    }  )
               (     {Sx...    }  )
               (     {Ax...    }  )
               (     {n        }  )
               ( ,T= train        )
               ( ,S= {Yx...    }  )
               (     {Nx...    }  )
```

Where:

1	= space the printer one line after each print line; i.e., single space the printer ignoring problem program carriage control
Rx...	= set the printer or punch to output jobs using standard forms (STD.) allowing changing of forms on a DEMAND basis
Sx...	= set the printer to print special forms data sets using standard (STD.) forms which the operator has loaded into the device
Ax...	= set the printer or punch to output jobs using special forms under HASP AUTOMATIC forms assignment
n	= a one to four digit number specifying the forms which the operator has loaded into the device
train	= the two character train or chain identification (AN, HN, PN, QN, RN, or UN).
carriage	= the single character 3211 carriage tape identification (6, 8, or U)
Yx...	= set the printer or punch to provide HASP separator pages or cards between data sets of different jobs.
Nx...	= set the printer or punch not to provide HASP separators and (in case of a non-console remote workstation) not to provide operator messages on the printer.

Action: The specified printer will be set to handle the carriage control (C=), forms (F=), and train/chain (T=) as specified or the specified punch will be set to handle the forms (F=) specified. If the specified printer has the UCS feature installed, the UCS buffer will be loaded with the print train/chain image prior to the printing of each job.

Notes:

1. The effect of C=1 will be negated by entry of a successful $S command directed to the printer.

2. Multiple settings directed to the same device using the same command entry are permitted.

3. The specification F=STD. will cause the printer to print the normal batch stream jobs using the installation's standard forms. However, a job requesting special forms in a data set to be printed with the rest of the job will cause a forms mount before printing the data set and a forms mount to STD. upon completion of printing this data set.

Limitations:

1. The setting of forms or train/chain is valid only when the appropriate device is not being used. It is recommended that the operator enter "$P device name" and wait for it to enter the DRAINED status.

2. Train/chain settings directed to remote printers or local printers without UCS will be ignored.

3. The (T=) operand is defined only for local printers. The remote CPU workstation operator must load UCS buffers via means other than through the HASP central system.

4. The (c=carriage) operand is defined only for local printers. The remote CPU workstation operator must load the carriage buffers via means other than through the HASP central system.

Responses:	OK	- the settings requested have been made

Examples:

```
1.  user   - $T PRT1,C=1
    system - OK

2.  user   - $T PRT1,F=AUTO,T=PN
    system - OK

3.  user   - $T PUN1,F=4732
    system - OK

4.  user   - $T PRT2,F=STD.,T=HN
    system - OK
```

Comments:

In example 1 the operator discovers the problem program is skipping to carriage tape channels which violate the installation procedures. The entry of $T PRT1,C=1 causes the printer to single space after each line printed to the end of the job.

In example 2 the operator desires to use the printer to print all jobs queued for special forms allowing HASP to select the special forms to be mounted. The printer has been loaded with a PN train to be used with the special forms.

In example 3 the operator desires to use the punch to punch only those data sets in the system which have specified forms type '4732'.

In example 4 the operator desires to use the printer to print the normal batch output using standard forms and the HN print train.

```
Definition:   $T console  (,level                  )    SET DEVICE
                          ),class[,class,...]      (
                          ),Rx...                  (
                          (,A=authority            )
```

Where:

console = HASP console device

level = a number, 0-15, which specifies the highest operator message level to eliminate for the designated console. A value of 0 will allow all messages for the designated console to be displayed. A value of 15 will eliminate all messages. The following list indicates the general levels of messages displayed by the HASP system:

1 - non-essential messages
3 - normal messages
4 - messages requiring operator action
7 - essential messages

class = the logical console class of messages the specified console is to display in addition to current classes. Specify class as follows:

```
LOG   - log console messages
ERROR - error messages
UR    - unit record messages
TP    - HASP RJE line messages
TAPE  - tape console messages
MAIN  - main operator console messages
OS    - OS WTO messages
```

Comments: When using the VARY CPUx,OFFLINE command in a Model 65 Multiprocessor system with HASP Console Support, the operator must first issue the HASP command

$T console,R

for the console on the CPU to be varied offline, and wait until the console stops printing messages.

Rx...	=	RESET--indicating the level value is to be set to 15 to eliminate all output based upon importance and the class settings are to be reset to eliminate all output based upon logical console class.
Authority	=	a number, 0-7, representing one or more command authority groups as follows: 0 - display only console 1 - system control console 2 - device control console 4 - job control console Under HASP multiple console support, console authority may be set to allow controlling commands to be entered from consoles which have the authority to control the designated function. Multiple settings are accomplished by the operator adding the authority group numbers together and entering the result; i.e., 7 indicates authority 1+2+4 (zero is assumed).

Notes:

1. See CONSOLE SUPPORT section of this manual.
2. The entry console for A=authority operands must be authorized for system control (authority 1, 3, 5 or 7)and the device specified must not be the entry console.

Action: The specified console is set to the appropriate list "level" logical console "class" and "authority" indicated by the operands. Each operand is handled individually and completely starting with the second operand. If an operand is determined to be in error, previous operands will take effect and the operand in error, along with succeeding operands will be ignored. If the RESET operand is used, it is assumed to be the last of the list.

Responses: OK - the settings requested have been made

Examples:

1. user - $T CON1,15
   system - OK

2. user - $T CON1,4
   system - OK

3. user - $T CON1,RESET
   system - OK
   user - $T CON
   system - OK

4. user - $T CON3,A=0
   system - OK

Definition: $T CON,level,class [,class,...] SET DEVICE

Where:

CON = CON--indicating that HASP is to set the level of message output for the logical console classes passed to OS consoles.

level = a number, 0-15, which specifies the highest operator message level to eliminate for the designated logical console class specified in the succeeding operands. A value of 0 will allow all HASP messages for the designated logical console class to be displayed on the OS console(s) assigned to display the message class. A value of 15 will eliminate all HASP messages of the specified class. The following list indicates the general levels of messages displayed by the HASP system:
1 - non-essential messages
3 - normal messages
4 - messages requiring operator action
7 - essential messages

class = the logical console class of the messages given to OS for display purposes. Specify as follows:
LOG - log console messages
ERROR - error messages
UR - unit record messages
TP - HASP RJE line messages
TAPE - tape console messages
MAIN - main operator console messages

Notes:

1. See CONSOLE SUPPORT section of this manual.

2. Responses to HASP commands will always be displayed at the console of entry regardless of the logical console class or level settings.

Action: The display "level" of the logical console classes will be set to the level specified. Each logical console class is set independently from the others starting with the first listed, operand 3. If an error is detected in the list, the operands preceding the operand in error will be acted upon and the operand in error and all succeeding operands will be ignored.

Responses: OK - the logical console classes have been set to the display level specified.

Examples:

1. user - $T CON,4,MAIN,LOG
   system - OK

Definition:	$Z device list	HALT (STOP) DEVICE
Where:	device = HASP reader, printer, punch, and console devices	
Action:	The specified devices will be HALTED after the current scheduled operations complete. In case of HASP consoles, all logical console classes and levels of importance will be RESET so that, except for direct responses to commands entered from the console, no new messages will be directed to the device. The effects of the $Z command may be negated by use of the $S command.	
Responses:	OK - the device(s) has been set to HALT operations	
Examples:		

1. user - $Z PRT1,PRT2,RDR1,PUN1
   system - OK

1.8 SYSTEM COMMANDS

System commands control the ability of the HASP System to process jobs through OS and may be broken down into two groups:

INITIATOR COMMANDS	- those commands which control the actual selection and submission of jobs from HASP to OS for processing.
HASP SYSTEM COMMANDS	- Those commands which control the ability of the HASP System to process jobs for any function.

In the following descriptions, a parameter "n" is referred to as the initiator identification. This identification is assigned by the systems programmer during the HASPGEN process. However, it is assumed in this manual that the initiator identifications are one or two character numeric digits 1, 2, 3, ...; in MFT the values correspond to the partition numbers.

INITIATOR STATUS CONDITIONS

An initiator's ability to process jobs depends upon the availability of jobs in the input queue of corresponding classes and the status of the initiator. These status conditions are as follows:

ACTIVE - the initiator is currently processing a job and has the ability to continue processing.

INACTIVE - the initiator has the ability to process jobs but no job of the initiator's current classes is ready for execution.

DRAINING - the initiator is currently processing a job but will not select another upon completion of the current job.

DRAINED - the initiator is not processing a job and will not attempt to select any job.

JOB SELECTION FOR OS EXECUTION

When the HASP System completes reading card images associated with an OS job, the job is placed into one of the HASP logical execution queues. The appropriate execution queue is selected based upon the job class as specified by:

1.	CLASS=class	parameter on the OS job card submitted by the programmer.
2.	$T JOBn,C=class	HASP command entered by the operator after previous queueing based upon the job card.
3.	CLASS=A	default specification in lieu of other specifications.

Each job is placed in the appropriate execution queue in order by priority so that higher priority jobs within the queue will be selected for execution before jobs of lower priority and that jobs of the same priority will be selected in order first in - first out. Job selection priority is determined from the following sources:

1. The time and line estimate in the HASP accounting field of the OS job card: Although the correlation of time estimate with priority is determined at HASPGEN time, it is normally set to give the shortest running jobs highest priority.

2. /*PRIORITY card which may appear preceding the job card. This card overrides the time and line estimate priority setting.

3. $T JOBn,P=priority HASP command entered by the operator after previous queueing.

When an initiator enters the INACTIVE status, it will attempt to select ready jobs from the HASP job queue in a manner directly controllable by the operator. An initiator will search the logical execution queues for jobs in order by class. If the operator has set the initiator to execute classes "ABX" in that order, the initiator will initiate only Class A jobs so long as there are Class A jobs ready for execution. When there are no Class A jobs ready, the initiator will initiate only Class B jobs or, if no Class B jobs, Class X jobs. The operator, therefore, by altering the initiation classes controls the selection of jobs based upon the job class. By appropriate job classing and setting of initiator class selection lists, jobs with complementary characteristics will tend to be in execution concurrently.

Definition: $D I[n] DISPLAY INITIATOR(S)

Where: n = the identification of the initiator to be displayed

Action: The status and eligible classes for the initiator(s) indicated will be displayed. If n is not specified, all initiators will be assumed.

Responses:

```
INIT n ( DRAINING ) = classes - one response for each
         DRAINED                  initiator requested
         ACTIVE
         INACTIVE
```

Examples:

```
1.   user   - $DI1
     system - INIT 1 (ACTIVE)=ABCD

2.   user   - $DI
     system - INIT 1 (ACTIVE)=ABCD
              INIT 2 (DRAINING)=BCDA
              INIT 3 (INACTIVE)=CDAB
              INIT 4 (DRAINED)=DABC
```

Definition:	$P I[n]	STOP (DRAIN) INITIATOR(S)
Where:	n	= the identification of the initiator to be stopped

Action:

The designated initiator will be prevented from selecting additional jobs for processing. If a specified initiator is actively processing a job (ACTIVE), its status will be changed to (DRAINING) until the current job terminates. If a specified initiator is not actively processing a job (INACTIVE) or upon completion of processing, the status of the initiator will be (DRAINED).

If the optional identification is <u>not</u> specified, all initiators will be stopped.

If the system contains the Execution Batch Scheduling feature (see section 12.13 of HASP Systems Manual), this command will cause a batch program(s) under control of the designated initiator(s) to be cancelled when the initiator(s) becomes DRAINED, thereby releasing memory for other processing.

Responses: OK - the specified initiator(s) are or will be stopped

Examples:

1. user - $PI3
   system - OK

2. user - $PI
   system - OK

Definition: $S I[n]   START INITIATOR(S)

Where: n = the identification of the initiator to be started

Action: The designated initiator will be allowed to select jobs of the appropriate job classes acceptable to the initiator. If the identification "n" is omitted, all initiators which were not stopped by a $PI command with the initiator specified will be started. If the initiator is DRAINING, its status will become ACTIVE; if DRAINED, its status will become INACTIVE and an immediate attempt to select a job will be made.

Responses: OK - the specified initiator(s) are started

Examples:

1. user - $SI3
   system - OK

2. user - $SI
   system - OK

Definition: $T In,list SET INITIATOR CLASSES

Where:

n	=	the identification of the initiator to be set
list	=	list of acceptable job classes for the specified initiator. Each class is listed in order of selection priority desired for the initiator. The maximum length of the list is specified by the system programmer at HASPGEN time.

Action: The new class list is inserted without inspection into the specified initiator's class selection list. All future job selection for the initiator will be done based upon the new list.

Examples:

1. user - $TI1,ABC
   system - OK

2. user - $TI2,BCA
   system - OK

3. user - $TI3,CAB
   system - OK

Definition: $P STOP SYSTEM

Action: All HASP job processing will be stopped. The HASP initiators, printers, and punches will not begin any new functions. The effects of $P under normal conditions may be negated by the $S command.

Responses: OK - current functions will be allowed to complete and the system will become dormant

Examples:

1. user - $P
   system - OK

Definition: $P HASP STOP HASP

Action: If HASP is in a dormant status, i.e., no job processing is in process, and HASP RJE lines DRAINED, the HASP SYSTEM will withdraw from control of the Operating System.

If the system contains the Execution Batch Scheduling feature (see section 12.13 of HASP Systems Manual), it is recommended (but not required) that the $P I operator command be issued and system activity be allowed to quiesce prior to issuing the $P HASP command. This will allow any batch programs to be cleared from the OS Job Queue prior to withdrawal of HASP.

Responses: HASP NOT DORMANT - response when HASP is unable to withdraw.

Note: Since HASP loses control of the system during withdrawal, a response is not issued by HASP. However, reader closed and initiator waiting for work may be issued by OS as an indicator of HASP job completion.

Examples:

1. user - $P HASP
   OS - reader closed/initiator waiting for work

Definition:	$S	START SYSTEM
Action:	All HASP job processing functions which are otherwise ready for activity will become ACTIVE. If the system is already processing jobs, it will continue to do so.	
Responses:	OK	- the HASP System functions will begin or continue.

Examples:

1. user - $S
   system - OK

1.9 MISCELLANEOUS DISPLAY COMMANDS

Definition: $D Dx... DISPLAY DIRECT ACCESS DEVICES

Action: The device addresses and volume serials of all on-line direct access storage devices will be displayed.

Limitations: The 2321 cell is not included.

Responses: aaa serial - one message for each device found

Examples:

```
1.    user   - $D DISKS
      system - 190 IPLRES
             - 191 LNKRES
             - 192 NO ID
             - 193 SPOOL1

2.    user   - $DD
      system - 190 IPLRES
             - 191 LNKRES
             - 193 SPOOL1
```

Note: Direct access devices which can be accessed through multiple channel paths will be displayed once for each active path.

Definition:	$D line	DISPLAY DEVICES ON RJE LINE
Where:	line	= HASP RJE line devices (see device list commands for specification)

Action: The status of the specified line along with the hardware device address assignment will be displayed. If no address is assigned, the address will be filled with "***". If the line is ACTIVE and associated with a HASP remote work-station, the HASP status of each device on the remote terminal will be displayed. See device list commands for status definitions.

Responses:

LINEn	aaa status	-	status of the specified line
RMr.devn	aaa status	-	one response for each device associated with the line (aaa is the address of the line)
LINEn	NOT FOUND	-	HASP has no record of the line specified

Note: Remote console devices will not be displayed.

Examples:

```
1.   user   - $D LINE1
     system - LINE1   031 ACTIVE
            - RM3.RD1 031 INACTIVE
            - RM3.PR1 031 ACTIVE
            - RM3.PU1 031 DRAINED

2.   user   - $DLNE2
     system - LINE2   032 DRAINED
```

Definition:	$D R	DISPLAY OUTSTANDING REPLY IDS

Action: All outstanding WTOR reply identification numbers will be displayed.

Note: This command is not defined if OS console support is being used.

Responses:

REPLY IDS: id,id,...id	- one line for each 10 reply ids
NO OUTSTANDING REPLY IDS	- no reply ids were found

Examples:

```
1.   user   - $D R
     system - REPLY IDS:  0,  1, 14, 11
```

Definition:	$D RMx...[r]	DISPLAY REMOTE(S)
Where:	r = the number of the remote. If r is omitted, all remotes will be assumed.	

Action: If the designated remote is currently associated with a HASP RJE line, the HASP status of the line and devices attached to the remote will be displayed. If the remote is not associated with a line, only the HASP status of the devices attached to the remote are displayed. If the remote number is not specified in the command, the HASP status of all remote devices will be displayed.

Responses:

```
LINEn    aaa status - status of the associated line
RMr.devn aaa status - status of each device on the
                      remote (aaa is the address of
                      the line)
```

Note: Remote console devices will not be displayed.

Examples:

```
1.   user   - $D RM3
     system - LINE 1  031 ACTIVE
            - RM3.RD1 031 INACTIVE
            - RM3.PR1 031 ACTIVE
            - RM3.PU1 031 DRAINED

2.   user   - $D RMTS
            - RM1.RD1 *** DRAINED
            - RM1.PR1 *** DRAINED
            - RM1.PU1 *** DRAINED
            - RM2.RD1 021 DRAINED
            - RM2.PR1 021 ACTIVE
            - RM2.PU1 021 INACTIVE
            - RM3.RD1 031 ACTIVE
            - RM3.PR1 031 ACTIVE
            - RM3.PU1 031 INACTIVE
```

Definition:	$D Ux...	DISPLAY UNITS

Action: The status of all HASP controlled, non-direct access devices attached to the local system will be displayed along with the corresponding hardware address of the device.

Note: If HASP multiple consoles are present, the ACTIVE status message will also display console authority.

Responses: device aaa status - one line for each HASP device

Examples:

```
1.   user   - $D UNITS
     system - READER1  00C INACTIVE
            - PRINTER1 00E ACTIVE
            - PRINTER2 00F DRAINED
            - PUNCH1   00D INACTIVE
            - TAPE1    *** DRAINED
            - CONSOLE  01F ACTIVE
```

1.10 REMOTE JOB ENTRY COMMANDS

Definition: $D Mr-rr,message — DISPLAY MESSAGE AT REMOTE TERMINAL(S)

Where:

r-rr = range of remote terminals--all remote terminals from r through rr are to receive the message.
= single remote number--the remote specified by r is to receive the message.

Notes:

(1) A remote specification of zero (0) indicates that the message is to be displayed at the central operator's console.

(2) If a range of remote terminals is specified by a remote terminal operator only the last remote specified will receive the message.

message = the text of the message desired to be displayed at the designated remote terminals. If the message is enclosed by apostrophes, the message will be upper cased and transmitted along with the apostrophes to the remote terminals indicated; otherwise, the text will be made upper case and blanks removed.

Action: The message will be transmitted to the indicated remote terminal if the terminal is capable of receiving the message.

Responses: OK - the message has been queued for transmission to eligible remote terminals.

Examples:

1. user - $D M4,Jobs remaining after 5PM will be purged
   at remote -0,JOBSREMAININGAFTER5PMWILLBEPURGED

2. user - $D M4,'Jobs remaining after 5PM will be purged'
   at remote -0,'JOBS REMAINING AFTER 5PM WILL BE PURGED'

Note: The value zero (0) at the beginning of the message indicates that the message originated at the central site. If the message originated from a remote the value would be the remote number.

Definition: $R type,for-id,to-id ROUTE JOB(S) OUTPUT

Where:

type	=	ALL--all output for the specified job(s) is to be routed
	=	PRT--print output for the specified job(s) is to be routed
	=	PUN--punch output for the specified job(s) is to be routed
for-id	=	JOBj--the designated output <u>for</u> job j is to be routed
	=	LOCAL--the designated output for all jobs currently in the system and routed <u>for</u> LOCAL devices is to be routed
	=	device--the designated output for all jobs currently in the system and routed <u>for</u> this device is to be routed
	=	RMx...r--the designated output for all jobs currently in the system and routed <u>for</u> remote r is to be routed
to-id	=	LOCAL--job(s) are to be routed <u>to</u> local devices
	=	device--job(s) are to be routed <u>to</u> this device
	=	RMx...r--job(s) are to be routed <u>to</u> remote r

Notes:

1. It is possible to route a job to a remote that does not exist.

2. In an unmodified HASP System device routing has no meaning and will be equivalent to specifying LOCAL or RMTr as appropriate.

3. RMT0 is equivalent to specifying LOCAL.

Action: The routing for print and punch data sets will be altered for the job specified or for all jobs currently in the system and routed for the output device group specified by the second operand to the routing as specified by the third operand.

Responses: OK - the job output specified has been routed

Examples:

1. user - $R ALL,J4,RMT6
   system - OK

2. user - $R PUN,RM3,LOCAL
   system - OK

2.0 STARTING THE HASP SYSTEM

HASP runs as a job under OS 360 in the MVT or MFT environment. Although jobs in the installation may be submitted to OS independently of HASP, it is assumed that all production jobs run by OS will be under the control of HASP and that HASP and OS have been tailored during the generation processes to minimize operator action required to start the system.

2.1 PREPARATION

The Operating System must be started and running correctly prior to any attempt to start HASP. All OS readers, writers and initiators should be stopped. If an OS "warm start" is performed, messages which indicate that the HASP System has abnormally terminated should be ignored; these messages result from the cleaning out of the OS queues from the last IPL of the system.

HASP requires that direct access volumes be mounted for the purpose of queueing JCL cards along with input data awaiting OS execution and for saving the job output for later output to the various printer and punch devices. One of these volumes will be labeled "SPOOL1". The additional volumes, if present, will be labeled "SPOOLx" where the last character "x" is an alphabetic character or numeric digit (other than 1); no two volumes may have the same volume serial. The maximum number of volumes to mount is determined by the installation during the generation of HASP. If the volumes are on-line and ready at OS IPL time and OS has not requested that they be removed, the SPOOL volumes are ready for the starting of HASP. However, if the above is not true, the operator should use the OS mount command to insure all SPOOL volumes are known to OS.

If HASP is to be "warm started", the exact physical volumes which were used during the last running of the system should be mounted. It is not necessary that the volumes be mounted on the same drives; the criteria is that all of the volumes be present and that the data set SYS1.HASPACE has not been altered. Additional SPOOL volumes may be added if desired.

All unit record and console devices which are to be used by HASP must be on-line to the CPU and should be in the ready status. If the unit record devices are not on-line at the time HASP is started, they will be unusable for any purpose until the next starting of HASP. If HASP Remote Job Entry is to be used, the line adapters should be on-line and ready with dial data sets on AUTO and non-dial data sets in the ready condition. (If HASP has been generated with knowledge of the hardware addresses of the line adapters, the adapters need not be on-line until an attempt is made to use them.)

2.2 STARTING THE HASP JOB

With the operating system otherwise dormant and ready for job processing, the direct access SPOOL volumes mounted and known to the operating system, the unit record, console, and line devices on-line, the operator starts the HASP job by entering the OS command:

```
S  HASP    - MVT start command
S  HASP.Pn - MFT start command where n is the partition number
             of the HASP partition (normally 0)
```

The start command causes OS to read the procedure "HASP" from SYS1.PROCLIB. The "HASP" procedure is an OS reader procedure which reads the HASP job from a direct access data set and starts an initiator to class H. The initiator will load the HASP executable module into storage and pass control to HASP.

HASP will issue an initial WTOR requesting directions from the operator. The WTOR message will appear as follows:

"nn $ SPECIFY HASP OPTIONS -- HASP-id VERSION x.x"

The operator should respond to this message using the standard OS reply format with the corresponding reply number "nn". The text portion of the reply must be one or more options selected from table 2.2.1. Each option may be entered in either upper or lower case. A comma must be used to separate the options. Blanks are not permitted. If two options are entered which are considered opposite, the latter option overrides the former. The FORMAT option, when used, has the effect of COLD starting regardless of the WARM/COLD specification.

WARM STARTING HASP

When HASP is "warm started", it will require that all SPOOL volumes which were up during the last execution of HASP be present and available. HASP will assume that the volumes are intact and that no FORMATTING will be required to run with the volumes. If a new volume with a "SPOOLx" label is present, HASP will make a few basic checks to determine if it has been pre-formatted, and format the volume if necessary. It is recommended, however, that only pre-formatted volumes be added at HASP warm start time.

Jobs which were in execution at the time the CPU was stopped will, on a HASP "warm start", be scheduled for execution again. For this reason, the operator should enter as a reply to the HASP WTOR:

R 00,'WARM,REQ' (assuming 00 is the current reply number)

HASP will list the activity in process at the time the CPU was stopped and wait for the operator to enter requests. The wait for HASP REQUESTS serves the following purposes:

1. It allows OS to flush the interrupted jobs from the OS queues.

2. It allows the operator to examine each job listed to determine whether or not:

    A. to allow the job to be automatically re-executed by HASP

    B. to hold the job for further investigation

    C. to cancel the job allowing it to be purged from the system

3. It allows the operator to examine the activity on the output devices to determine what action to take prior to starting normal job processing.

4. It allows the operator to change the default status of HASP initiators and devices as well as modify the status of jobs in the HASP queue.

When the operator has determined that the system is ready for job processing, he should enter "$S" on the console.

The following examples list the console messages and reply sequence expected during HASP initialization.

```
1.    user   - S HASP
      system - 00 $SPECIFY HASP OPTIONS -- HASP id VERSION x.x
      user   - R 00,'COLD,FORMAT'
      system - SPOOLi IS BEING FORMATTED
      system - ENTER HASP REQUESTS
      user   - $S

2.    user   - S HASP.P0
      system - 00 $SPECIFY HASP OPTIONS -- HASP id VERSION x.x
      user   - R 00,'U'
      system - ENTER HASP REQUESTS
      user   - $S
```

TABLE 2.2.1 HASP INITIALIZATION OPTIONS

OPTION	OPPOSITE	MEANING
FORMAT	NOFMT	All SPOOL volumes are to be formatted.
<u>NOFMT</u>	FORMAT	No SPOOL volume is to be formatted unless HASP determines necessary.
COLD	WARM	Any job data contained on the SPOOL volumes is to be ignored.
<u>WARM</u>	COLD	HASP is to continue processing where it left off during the previous IPL.
REP	NOREP	Replacement cards are to be used for temporary modifications to HASP for this IPL. This option should be specified only under the direct supervision of the system programmer responsible for the replacement cards.
<u>NOREP</u>	REP	No replacement cards are to be used.
<u>REQ</u>	NOREQ	HASP is to stop and wait for a $S command before beginning job processing.
NOREQ	REQ	HASP is to begin job processing when ready to do so.
<u>LIST</u>	NOLIST	HASP is to list on a designated printer any replacement cards read.
NOLIST	LIST	HASP is not to list replacement cards.
<u>TRACE</u>	NOTRACE	Allow tracing of HASP internal execution; this option is not active on a system generated for production.
NOTRACE	TRACE	Cut off the tracing of HASP internal execution.
NONE		Take all default options.
U		Take all default options.

Note: The options underlined are the normal default options.

3.0 ABBREVIATED WTOR REPLY

This section discusses the entry format for the Operating System "Reply to Information Request" command. When HASP is in the system the following additional formats of this command may be used:

1. The "R" or "REPLY" keyword may be omitted:

   nn,'text'

2. The comma may be omitted:

   nn'text'

3. If all alphabetic text characters may be optionally upper case, the apostrophes may be omitted:

   nntext
   nn,text

4. The numeric identifier may be one dight unless:

   a. The first text character is numeric and

   b. Neither the separating comma nor apostrophe is present:

      ntext
      n,text
      n'text'
      n,'text'

4.0 HASP MESSAGES AND CODES

The following sections list those messages originating from HASP which are not direct responses to HASP operator commands.

4.1 HASP INITIALIZATION MESSAGES

All HASP Initialization Messages are displayed by OS WTO or WTOR requests and are listed as follows:

CORRECT THE ABOVE PROBLEMS AND RESTART HASP

Explanation: This message occurs following one or more messages which describe why HASP direct-access initialization could not complete normally.

System action: The HASP job will terminate.

Operator response: Self-explanatory.

EXTENT ERROR ON SPOOLx

Explanation: The operator did a HASP warm start. HASP has found that the first extent of data set SYS1.HASPACE on SPOOLx is different from what it was previous to the warm start. This could be due to the wrong SPOOLx volume having been mounted, a different HASP system having been started, or SYS1.HASPACE having been scratched and re-allocated.

System action: After attempting to verify the remaining required SPOOL volumes, the HASP job terminates.

HASP MFT MCS SUPPORT REQUIRES RESIDENT SVC OPTION (TRSVC) SPECIFICATION AT SYSGEN TIME - HASP TERMINATED

Explanation: The MFT System does not contain the proper format SVC table (four byte entries) for use with HASP when HASPGEN variable &NUMCONS=0 was chosen. See HASP manual section 10.1.

System action: The HASP job will terminate.

Operator response: Notify System Programmer.

HASP module ATTACH ERROR - code

Explanation: HASP has attempted to attach a sub-task which is required for the running of the system. The module name indicates the ECBDIC name of the sub-task entry module and code is the OS completion code returned. If the system is allowed to continue processing, the results will be unpredictable but will cause general malfunction as follows:

Module HASPWTR - Jobs upon completion of OS execution will remain on the OS job queue and HASP will not become aware of the user job termination.

Module HASPBR1 - HASP WTO message facility will be inactive eventually causing HASP to become interlocked attempting to use the OS console interface.

System action: HASP will attempt to process jobs.

Operator response: Probable user error. Stop HASP, refer to the OS messages and completion codes manual, and correct the problem as indicated.

INVALID UNIT RECORD DEVICE CONTROL TABLES

Explanation: An inconsistency has been detected in the HASP control section HASPINIT. Unit record device control tables have been improperly generated.

System action: The HASP job will terminate.

System programmer response: Check the assembly of HASPINIT for improperly applied modifications and insure the correct HASP overlay data set corresponds with the current HASP resident module. Reassemble HASPINIT, recreate the HASP overlay data set, and LINKEDIT the HASP module as required.

JOB j WAS {READING / EXECUTING / PRINTING / PUNCHING}

Explanation : The operator did a HASP warm start. At the time of system stop, the job numbered j was in the process of reading, executing, printing, or punching.

System action: If the job was reading, it is now purged. If the job was executing, HASP will restart its execution at the first job step. If the job was printing, HASP will restart its print phase back a few pages. If the job was punching, HASP will restart its punching from the beginning.

Operator response: If the job was reading, it should be read in again. If the job was executing, printing, or punching, no operator response is necessary if the default HASP action is desired.

MAXIMUM OF n SPOOL VOLUME(S) EXCEEDED

Explanation: More direct-access volumes with labels SPOOLx have been found on-line than HASP has been generated to handle (x is any alphameric character).

System action: The HASP job will terminate.

Operator response: Probable user error. Check the volume labels of all direct-access volumes and remove all but "n" volumes. Restart HASP.

MAXIMUM OF n device type EXCEEDED

Explanation: HASP found more reader, printer, punch, or console devices physically on-line to the CPU than the installation indicated for HASP to support.

System action: The first n devices of the specified type will be used by HASP; the additional devices of the specified type will be ignored.

System programmer action: Check the OS generation to insure that the hardware devices correctly reflect the system configuration and that the additional pseudo devices generated in OS for HASP do not address a HARDWARE device or control unit on the system.

MOUNT SPOOLx ON A yyyy

Explanation: The operator did a HASP warm start. HASP has found that not all SPOOL volumes are mounted which were mounted prior to the warm start. In the message, x completes the SPOOL volume serial number and yyyy is the device type upon which the volume had been mounted.

System action: After attempting to verify the remaining required SPOOL volumes, the HASP job will terminate.

Operator response: Probable user error. Mount the required volume(s) on the required devices and do a HASP warm start, or merely a HASP cold start. This message could also mean that the wrong SPOOL1 volume was mounted.

OBTAIN FAILED ON SPOOLx WITH CC nn

Explanation: The operator did a HASP warm, cold, or format start. HASP used the OBTAIN supervisor service to get information about data set SYS1.HASPACE on volume SPOOLx, but OBTAIN did not work as expected. OBTAIN returned condition code nn to indicate the problem.

- nn = 4 - SPOOLx was not mounted. This error should not occur.
- nn = 8 - SYS1.HASPACE was not allocated on SPOOLx.
- nn = 12 - A permanent input/output error was found during OBTAIN processing.
- nn = 16 - This error should not occur.
- nn = 20 - This error should not occur.

System action: After attempting to verify the remaining SPOOL volumes, the HASP job will terminate.

Operator response: Probable user error. If nn = 8, allocate a data set named SYS1.HASPACE on SPOOLx and do a HASP warm start. If nn = 12, use the IBM utility program IEHDASDR or IBCDASDI to re-initialize the SPOOLx volume and then follow the procedure for nn = 8.

OLAYLIB DOES NOT MATCH RESIDENT HASP

Explanation: The job used to start HASP (normally in SYS1.PROCLIB when HASP is started by usual method) referenced a load module (from SYS1.LINKLIB or a JOBLIB or STEPLIB) which was not created from the output of the same execution of HASPOBLD which created the referenced OLAYLIB.

System action: The HASP job will terminate.

Operator response: Probable user error. Verify that the correct start command and direct-access volumes which contain parts of the HASP System are being used. Restart HASP. If unsuccessful, notify system programmer.

System programmer response: Verify that procedures HASP and STRTHASP (or their equivalents, see Section 10.1.4.2 of the HASP Manual) are correctly installed in SYS1.PROCLIB and that data sets they reference are cataloged and mounted, etc. If difficulty persists, re-do the install HASP program actions (sample job HASPHASP) as described in Section 10.1.4.3.

OPERATOR MESSAGE SPACE NOT AVAILABLE

Explanation: HASP has attempted to reserve tracks from SPOOL1 volume for remote operator message queuing and found:

1. The first extent of SYS1.HASPACE was not large enough for the requested number of spool records.

2. During HASP "warm start" the SPOOL1 volume was found incompatible with the loaded copy of HASP.

System action: The HASP job will terminate.

Operator response: If HASP "warm start", match the SPOOL1 volume with the HASP load module used during the "cold start". If "cold start" consult the system programmer.

System programmer response: Insure the HASP generation parameter &SPOLMSG has been correctly applied to the system and check the extents of SYS1.HASPACE on SPOOL1 for requested space.

OVERLAY REPPING ERROR

Explanation: REP card intended for resident CSECT may be mispunched or REP card intended for overlay CSECT cannot be processed because no space exists to save it. HASPGEN parameter &OREPSIZ was not set large enough to hold amount of overlay REP information currently being processed or &OREPSIZ was set to zero which eliminates capability of applying REP cards to overlay CSECTs.

System action: The HASP job will terminate.

Operator response: Probable user error. Verify that REP cards are those intended for the HASP System which was started. Restart HASP and attempt to use correct REP cards. If unsuccessful, notify system programmer.

System programmer response: Verify that REP cards are punched correctly according to format described in Section 6.4.1 of the HASP Manual and/or re-HASPGEN with parameter &OREPSIZ set larger to reserve more space for overlay REPs.

PERM I/O ERR ON SPOOLx WHILE FORMATTING

Explanation: HASP was unable to complete formatting the first extent of SYS1.HASPACE on SPOOLx. This may be because a hardware error occurred or because the SPOOL volume is not properly initialized.

System action: After attempting to process the remaining SPOOL volumes, the HASP job will terminate.

Operator response: If the message was caused by a hardware malfunction, have it corrected. If not, the SPOOL volume may need to be reinitialized; reinitialize it using the IBM utility program IEHDASDR or IBCDASDI.

PERM I/O ERR READING HASP CKPT

Explanation: The operator did a HASP warm start. HASP was unable to read the checkpoint record on SPOOL1. This may be because the wrong SPOOL1 was mounted, a different HASP System was started, or the checkpoint record had been destroyed.

System action: The HASP job will terminate.

Operator response: Probable user error. Mount the correct SPOOL1 volume and do a HASP warm start using a HASP System compatible with the old HASP checkpoint. If this fails, do a HASP cold start.

PERM I/O ERR WRITING HASP CKPT

Explanation: HASP failed to format-write correctly the HASP checkpoint record on SPOOL1. This could be because of a hardware malfunction or because the HASPGEN variables used to generate HASP created a checkpoint record too long to be written on the type of device upon which SPOOL1 is mounted.

System action: The HASP job will terminate.

Operator response: If the message was caused by a hardware malfunction, have it corrected. If the message was caused by too long a checkpoint record, and if the installation has devices which can support longer records, prepare a SPOOL1 volume for one of these devices. Otherwise it is necessary to do another HASPGEN, specifying parameters which will create a smaller checkpoint record.

SET RESTART PSW TO 0004000000aaaaaa FOR TAPE DUMP

Explanation: The special tape dump feature has been generated by the system programmer. The entry point to the dump routine is indicated by the hexadecimal address aaaaaa.

Operator Response: In the event a STAND ALONE DUMP is necessary, the operator may use the HASP tape dump feature by using the following procedures:

1. Ready the designated tape drive with a scratch tape at load point with ring in. (The device address is determined by the system programmer, but may be altered by over storing the half-word aaaaaa-4 with the new tape address.)
2. Stop the CPU and press system reset (this sets the tape mode).
3. Store the displayed PSW in location 0-7.
4. Press PSW RESTART
5. IPL a runable system and execute IMDPRDMP as prescribed by OS/360 SERVICE AIDS manual or equivalent post processor.

PREVIOUSLY-MOUNTED VOL SPOOLx IS UNFORMATTED

Explanation: The operator did a HASP warm start. HASP has found that the length of the first record of the last track of the first extent of SYS1.HASPACE on SPOOLx is incorrect. This could be due to its having been overwritten, a different HASP system having been started, or the wrong SPOOLx volume having been mounted.

System action: After attempting to verify the remaining required SPOOL volumes, the HASP job will terminate.

Operator response; Probable user error. If the wrong SPOOLx volume was mounted, mount the correct volume and do a HASP warm start. Otherwise do a HASP cold start; any SPOOL volumes that are not correctly formatted will automatically be re-formatted on a HASP cold start.

SPOOL VOLUMES HAVE DUPLICATE LABELS

Explanation: Multiple direct-access volumes have been found with identical SPOOLx labels (x is any alphameric character).

System action: The HASP job will terminate.

Operator response: Probable user error. Check the volume labels of all direct-access volumes on the system and remove the required volumes. Restart HASP.

SPOOLx IS BEING FORMATTED

Explanation: The operator did a HASP warm, cold, or format start. HASP detected an unformatted SPOOL volume which it could format and is now formatting the volume.

System action: HASP will format unformatted new SPOOL volumes on a warm start, all unformatted SPOOL volumes on a cold start, and all SPOOL volumes on a format start.

SPOOL1 IS NOT MOUNTED

Explanation: The operator did a HASP warm, cold, or format start, and HASP could not find a non-2321 direct-access UCB with volume serial SPOOL1. The SPOOL1 volume is required to be mounted and on-line when HASP is started.

System action: The HASP job will terminate.

Operator response: Probable user error. Make sure that SPOOL1 is mounted, ready, and on-line. Then restart HASP.

n BUFFERS NOT AVAILABLE

Explanation: HASP has not been able to allocate enough dynamic storage to build the minimum number of buffers required to run the system as specified by the HASP generation parameter &MINBUF.

System action: An attempt will be made to run with the available buffers.

Operator response: Probable user error. Take one of the following actions depending upon installation procedure:

1. Stop enough HASP functions to allow running with less than &MINBUF buffers.

2. Stop the HASP System, change the HASP region or partition size, and restart HASP.

nn $SPECIFY HASP OPTIONS -- HASP-id, VERSION x.x

Explanation: HASP has been given control and is requesting instructions from the operator.

System action: Wait for REPLY.

Operator response: Read the section STARTING THE HASP JOB and enter the desired options using the OS reply format.

nn $SYNTAX ERROR -- RESPECIFY OPTIONS

Explanation: HASP does not recognize one or more of the initialization options entered by the operator.

System action: Reset to default responses and wait for REPLY.

Operator response: Probable user error. Read the section STARTING THE HASP JOB and carefully enter the desired options.

4.2 HASP SYSTEM CATASTROPHIC ERROR CODES

All HASP System catastrophic errors are considered so extremely serious in nature that HASP is unable to continue processing. The message will be displayed on a single 1052 console, designated by the HASP generation parameter $PRICONA, using a hardware START I/O instruction. HASP will then go into a single instruction loop.

SYSTEM PROGRAMMER RESPONSE

A storage dump should be taken by STAND-ALONE utility and saved for later analysis. A careful check of the HASP generation process should be made to insure that HASP modules are assembled properly (modifications are correctly entered and no errors occurred during assembly), that the overlay library has been properly created, that the linkage editor created a correct HASP resident module, and that the HASP execution JCL corresponds to the data sets designated by the generation JCL. If any doubt exists that the HASP generation process is other than perfect, a new complete HASP generation should be undertaken.

A01

Explanation: HASP has detected more channel end indications for a device than expected. Only one channel end indication should be received from IOS for each Input/Output operation which HASP has initiated.

System action: Continuous Loop.

Operator response: Take a STAND-ALONE DUMP and notify system programmer.

B01

Explanation: Probable user error. Either (1) an attempt has been made to return an invalid HASP buffer to the buffer pool, or (2) the free buffer chain has been destroyed.

System action: Continuous Loop.

Operator response: Take a STAND-ALONE DUMP and notify system programmer.

E01

Explanation: The total number of channel end indications from IOS has exceeded the total number of Input/Output operations which HASP has initiated (i.e., the total number of outstanding Input/Output operations has gone negative).

System action: Continuous Loop.

Operator response: Probable user error. Take a STAND-ALONE DUMP and notify system programmer.

K01

Explanation: The Checkpoint Processor has discovered that some track groups are both free and allocated.

System action: Continuous Loop.

Operator action: Take a STAND-ALONE DUMP and notify system programmer.

M01

Explanation: HASP has detected more channel end indications for an RJE line than expected. Only one channel end indication should be received from IOS for each Input/Output operation which HASP has initiated.

System action: Continuous Loop.

Operator response: Take a STAND-ALONE DUMP notify system programmer.

M02

Explanation: An attempt has been made to initiate an Input/Output operation on an RJE line before the previous operation has completed.

System action: Continuous Loop.

Operator response: Take a STAND-ALONE DUMP and notify system programmer.

001

<u>Explanation</u>: A HASP Processor already logically executing under overlay control has issued another call ($LINK or $LOAD) to Overlay Service without exiting from overlay control ($RETURN and $DELETE). Test for this condition is performed only if the HASPGEN parameter &DEBUG is set to YES.

<u>System action</u>: Continuous Loop.

<u>Operator response</u>: Take STAND-ALONE DUMP and notify system programmer.

<u>System programmer response</u>: Probable user error. Check any local modifications to HASP for the errors described above. Consult IBM Customer Engineer if problem remains undetermined.

V01

<u>Explanation</u>: The Purge Processor has discovered that some track groups to be freed were already free.

<u>System action</u>: Continuous Loop.

<u>Operator response</u>: Take a STAND-ALONE DUMP and notify system programmer.

X03

<u>Explanation</u>: The Execution Processor routine XTERMIN8 which deallocates DDBs discovered a non-existent UCB entry in the DDB being deallocated.

<u>System action</u>: Continuous Loop.

<u>Operator response</u>: Take a STAND-ALONE DUMP and notify system programmer.

X04

<u>Explanation</u>: The Execution Processor DDB Service routine (XDDBCONT) which maintains the DDB frequency table was unable to match the action DDB with the frequency table entry.

<u>System action</u>: Continuous Loop.

<u>Operator response</u>: Take a STAND-ALONE DUMP and notify system programmer.

X05

Explanation: The HASP Reader/Interpreter appendage initialization routine (XJCLTEST) could not identify the JCL keyword value provided on the first entry to the appendage. The first entry is presumed to be a JOB statement and the keyword value must be X'65' (Release 18 and prior releases) or X'B4' (Release 19 and subsequent releases).

System action: Continuous Loop.

Operator response: Take a STAND-ALONE DUMP and notify system programmer.

H01

Explanation: A HASP Control Service Program function which was not generated was requested by a HASP processor.

System action: Continuous Loop.

Operator response: Take a STAND-ALONE DUMP and notify system programmer.

System programmer response: Probable user error. Validate HASPGEN parameters for consistency across all modules. Verify local modification dependency on HASPGEN parameters.

ABND

Explanation: The HASP abnormal exit (STAE) routine has been entered, indicating that the HASP SYSTEM has been abnormally terminated. The OS code indicating the reason for the ABEND may be found in the HASP TCB completion code field.

System action: Continuous Loop.

Operator response: Take a STAND-ALONE DUMP and notify system programmer.

4.3 HASP JOB PROCESSING MESSAGES

Messages displayed during HASP job processing reflect conditions which range from informational to serious errors and are listed as follows:

ALL AVAILABLE FUNCTIONS COMPLETE

Explanation: All HASP job processors have become dormant and no HASP RJE lines are active.

device BACKSPACED

Explanation: Output processing on the indicated printer is being backspaced.

System Action: The requested number of pages are backspaced. Then processing continues. If the start of the data set is encountered while backspacing, processing continues at the start of that data set.

device command

Explanation: The displayed command has been entered from the device indicated.

System Action: The command is passed to the command processor for further action.

device DELETED

Explanation: Output processing on the indicated device has been deleted.

System Action: The job being processed on the indicated device will be queued for the next processing phase. Output processing will be terminated.

device FWD-SPACED

Explanation: Output processing on the indicated printer is being forward-spaced.

System Action: The requested number of pages are skipped without printing. Then processing continues. If the end of a data set is encountered while skipping, processing continues with the beginning of the next data set.

device IS DRAINED

Explanation: The operator has entered a $P device command directed to the named device and the device has entered the DRAINED status.

$\left\{\begin{array}{l}\text{device}\\ \text{JOB j}\end{array}\right\}$ message

Explanation: The Input Service Processor has detected a /*MESSAGE control card in the input stream.

System Action: None

Operator Response: Observe the message and take any action which may be appropriate.

device REPEATED

Explanation: Output processing on the indicated device has been repeated.

System Action: The job being processed on the indicated device will be re-queued. Output processing will continue.

device RESTARTED

Explanation: Output processing on the indicated device has been restarted.

System Action: The job being processed on the indicated device will be re-queued. Output processing for the job will be terminated.

device SKIPPING FOR JOB CARD

Explanation: The Input Service Processor is now scanning the input stream for a Job Card.

System Action: The Input Service Processor will continue to read the input stream until a Job Card is encountered or until an end-of-data condition is recognized.

device SUSPENDED

Explanation: Output processing on the indicated printer is being interrupted.

System Action: The job being processed on the indicated printer will be re-queued in such a way that when it is processed again, printing will begin one page before the current point or at the beginning of the data set, whichever is less. Output processing will be terminated.

DISASTROUS ERROR - COLD START SYSTEM ASAP

Explanation: A critical I/O error has occurred on the SYS1.HASPACE data set. A corresponding I/O error message will accompany this message giving details of the error.

System Action: HASP will continue processing jobs using unaffected facilities.

Operator Response: Prevent new jobs from entering the system, prepare all jobs in the HASP execution queue for resubmission when HASP is restarted, allow HASP to complete all current jobs in execution and all output activity depleting the output queues, and stop HASP. The cause of the error should be determined before COLD starting HASP.

DISASTROUS ERROR DURING CHECKPOINT - RESTART ASAP

Explanation: An I/O error has occurred while attempting to write checkpoint information, thus preventing any possibility of performing a future HASP WARM start. An associated I/O error message will accompany this message.

System Action: HASP will discontinue the checkpointing of critical information on direct access.

Operator Response: Prevent new jobs from entering the system, prepare all jobs in the HASP execution queue for resubmission when HASP is restarted, allow HASP to complete all current jobs in execution and all output activity depleting the output queues, and stop HASP. The cause of the error should be determined before COLD starting HASP.

HASPWTR - PERM I/O ERR OS JOBQ

Explanation: The separate load module HASPWTR, which retrieves OS System Messages for HASP before the end of job execution, has received a permanent error indication while attempting I/O on the data set SYS1.SYSJOBQE, after all standard OS direct access error recovery actions have been attempted.

System Action: If the operation is a write, processing continues but later attempts to read the record may fail or read incorrect information. If the operation is a read, HASPWTR does not use the incorrect information. Processing of the single job, whole sysout MSGCLASS, or re-queue action is terminated (depending upon when the I/O error occurred) and other processing continues.

Operator Response: Use the $P command to prevent any new functions from starting. When all current functions have completed (except perhaps one or more jobs which may not finish execution if HASPWTR has stopped processing a MSGCLASS), re-IPL the system, cold start OS (this re-formats SYS1.SYSJOBQE by writing every record on it), and if no errors are indicated, warm start HASP to continue processing. If unsuccessful, notify system programmer.

System Programmer Response: The direct access volume containing SYS1.SYSJOBQE should be analyzed with an appropriate utility and/or the direct access device changed to localize possible machine malfunction. The IBM customer engineer should be notified if difficulties persist.

I/O ERROR ON device uuu,cc,ssss,iiii,bbcchhr

Explanation: An INPUT/OUTPUT error has occurred on the indicated HASP device where:

device	=	HASP device name or volume serial if DIRECT ACCESS
uuu	=	hardware address
cc	=	CCW op-code used at the time of error
ssss	=	CSW status code
iiii	=	sense information
bb	=	bin as appropriate
cc	=	cylinder as appropriate
hh	=	head as appropriate
r	=	record as appropriate

Associated error messages may be displayed as a result of the error. For direct access the following could be causes of the error:

1. The channel, control unit, or device is malfunctioning. This may be determinable by moving the volume (if movable) to a new drive, control unit, or channel and restarting HASP.
2. The recording surface is bad. This may be indicated by the nature of the error and distribution of the bbcchhr information (A reinitialization with assignment of alternate tracks followed by HASP FORMAT start may be desirable).
3. The data set SYS1.HASPACE may have been overwritten by improper data set assignment and protection procedures. This may be indicated by wrong length record indications. (A HASP FORMAT start is required).

System action: HASP will continue job processing and submit additional error messages indicating the severity of the error to the system.

Operator response: Determine the cause of the error and take appropriate action.

I/O ERROR ON LINEn uuu,cc,ssss,iirr,xyee

Explanation: An error has been detected on the indicated HASP RJE line or on a device attached to that line where:

```
   n = HASP RJE line number
 uuu = line adapter address
  cc = CCW op-code used at the time of error
ssss = CSW status code if no Block Sequence Check
     = 0000 - Indicator for normal channel end
              with a Block Sequence Check at the
       central CPU
     = FFFF - Indicator for normal channel end
              with a Block Sequence Check at
       the remote site
  ii = sense information if ssss=0E00
     = last character received if ssss=0C00
       and xy=94 or B4
  rr = additional sense information (if STR
       and ssss=0E00)
     = remote device first response character
       if BSC
   x = HASP CCW internal sequence identification
   y = HASP CCW internal sequence command type
  ee = expected response if BSC
     = blank if STR
```

Notes:

1. This message may also occur as an informational message when maintenance personnel have set HASP internal flags to log all channel ends on the line device.
2. The appropriate IBM Component Description System Reference Library manual describes the status and sense information in detail.

System Action: HASP will for most line errors attempt to recover and continue processing using the line.

Operator Response: The console log should be saved for maintenance personnel (even if recovery is successful). Additional responses depend upon the nature of the problem. The following discussions indicate typical errors and appropriate responses.

INTERVENTION REQUIRED - uuu,cc,0E00,40**,****
Check the corresponding line adapter to insure that the device is on-line and that the data set is ready.

DATA CHECK - uuu,cc,0E00,08**,****
- uuu,cc,0E00,**4*,** (STR write error)
If this is a continuous failure, the use of the HASP line should be discontinued (use $P and $E commands) until the cause of the failure is determined. If the error is occasional, the console log should be saved for maintenance purposes.

TIME OUT - uuu,cc,0E00,01**,****
This error represents the results of multiple time outs to indicate possible problems. If the remote system has entered the stopped state, perform one of the following procedures:

1. Wait for the remote to come back up.
2. Restart the line ($E command).

If the remote has not entered the stopped state and the remote is a MULTI-LEAVING workstation program, inform the system programmer.

BLOCK SEQUENCE CHECK (CENTRAL) - uuu,cc,0000,**rr,**ee
rr = block count modulo 16 received
ee = block count modulo 16 expected
This error indicates that one or more blocks which have been transmitted by the remote have not been received by HASP. The lost block situation will cause HASP to restart the input stream which will require that the remote operator resubmit the input which was aborted. If the block lost contained critical control signals, the input and/or output functions may be permanently suspended, thus requiring the operator to restart the line.

BLOCK SEQUENCE CHECK (REMOTE) - uuu,cc,FFFF,**rr,**ee
This error indicates that the remote detected a lost block. The error is similar to BLOCK SEQUENCE CHECK (CENTRAL) and will cause all output functions to be suspended.

INVALID RESPONSE - uuu,cc,0C00,iirr,xy**
This error is due to invalid control sequences (beginning or end sequence). The operator should respond in the same manner as for data check situations.

Table 4.3.1 HASP RJE Typical Sense Information on 2701

iirr	meaning
40**	Intervention required
08**	Data check (data received is invalid)
01**	Time out
**4*	Line error on output (STR only)

Note: The display of sense information has meaning only when (sss=0E00)

Table 4.3.2 HASP BSC MULTI-LEAVING Data Stream Control Sequences

rr	sequence	comments
01	SOH-STX-data-ETB	non-transparent data transfer
02	DLE-STX-data-DLE-ETB	transparent data transfer
2D	SOH-ENQ	initial sequence (prior to SIGN ON)
3D	NAK	remote did not receive last transmission correctly
70	DLE-ACKO	last transmission received correctly but remote has no data to transmit

Note: The display of control sequences has meaning only when (ssss=0C00).

Table 4.3.3 Command Codes Utilized by HASP RJE

cc	command
01	WRITE
02	READ
03	NOP
06	PREPARE (STR only)
07	STEP COUNT (STR only)
08	TRANSFER IN CHANNEL
17	ERROR (STR only)
23	SET MODE
27	ENABLE
2F	DISABLE
33	TEST SYNCH (STR only)
37	SEND EOT (STR only)
3B	SEND INQUIRY (STR only)

Table 4.3.4 HASP RJE CCW Internal Sequence Identifiers

x	sequence identification
0	STR Hardware Remote Read Sequence
1	STR CPU Remote Read Sequence
2	STR Hardware Remote Write Sequence
3	STR CPU Remote Write Sequence
4	STR Hardware Remote Prepare Sequence
5	STR CPU Remote Prepare Sequence
8	BSC Hardware Remote Read Sequence
9	BSC CPU MULTI-LEAVING Remote Write-Read Sequence
A	BSC Hardware Remote Write Sequence
B	BSC CPU MULTI-LEAVING Remote Write-Read Sequence
C	BSC Prepare Sequence

Table 4.3.5 HASP RJE CCW Internal Sequence Command Types

y	command type
0	Disable
1	Set Mode
2	Enable
3	Test Synch
4	Read Text
5	Read Response (normal)
6	Read Response (to ENQ)
7	Prepare
8	Write Text
9	Write Response
A	Send Inquiry (Write ENQ)
B	Send EOT

$\begin{Bmatrix}\text{INIT}\\\text{PART}\end{Bmatrix}$ i IDLE-CLASS=c...

Explanation: INIT/PART "i" is idle because the Execution Processor discovered that no jobs of the class(es) identified by "c..." were available in the HASP job queue.

System Action: The execution processor will activate the INIT/PART when jobs of the class(es) become available.

JOB DELETED BY HASP OR CANCELLED BY OPERATOR BEFORE EXECUTION

Explanation: The job was either deleted by the Input Service Processor of HASP or cancelled by an operator before OS Execution Processing.

System Action: The JCL is printed and the job is purged.

Note: This message appears only in the printed output stream for the job.

JOB j -- EXCESSIVE INPUT STREAM DATA SETS

Explanation: The Input Service Processor has detected an excessive number of "DD *" or "DD DATA" JCL statements for a single job. Either the total number of Pseudo 2540 Units or the number of HASP Data Definition Tables was exceeded.
System Action: The job will be deleted.

Programmer Response: Divide the job into a number of jobs such that no one job contains too many input stream data sets.

System Programmer Response: Increase the number of Pseudo 2540 Units generated and/or increase the number of HASP Data Definition Tables generated (see HASPGEN Parameter &NUMDDT).

JOB j -- ILLEGAL JOB CARD

Explanation: The Job Card for the indicated job was found to be invalid by the Input Service Processor.

System Action: Input Service Processing is terminated for this job.

Programmer Response: Correct the Job Card and resubmit the job.

System Programmer Response: The HASPGEN Parameter &RJOBOPT, if set to "NO", will allow jobs with illegal job cards to be processed.

JOB j -- ILLEGAL /*ROUTE CARD

Explanation: The Input Service Processor has encountered an invalid /*ROUTE control card.

System Action: Input Service Processing for the job is terminated.

Programmer Response: Correct the /*ROUTE card and re-submit the job.

JOB j -- jobname -- BEGINNING EXEC - $\left\{\begin{matrix}\text{INIT}\\\text{PART}\end{matrix}\right\}$ i - CLASS c

Explanation: Job "j", named " jobname ", is beginning the execution phase in the INIT/PART "i" as a Class "c" job.

JOB j AWAITING HASP ALLOCATION

Explanation: Insufficient HASP resources are available to process the specified job (j). The Execution Processor is unable to find an available DDB or UCB needed to service the indicated job's I/O request.

System Action: Processing of the specified job will continue when a DDB or UCB becomes available. Note: If insufficient DDBs or UCBs (pseudo devices) have been defined for the system, a permanent lockout condition can occur.

Operator Response: If a single job is being processed by HASP, then a permanent lockout caused by insufficient DDBs has occurred. Notify system programmer.

If multiple jobs are being processed under control of HASP, then DDBs or UCBs can be made available by OS cancelling a job which did not cause the condition.

System Programmer Response: Frequent occurrence of this message is an indication of insufficient resources (DDBs or UCBs) for proper system performance. See the HASP Manual, Sections 7.1 and 10.1 for guidelines on DDB and UCB definitions.

JOB j BUFFER ROLL UNSUCCESSFUL ... VERIFY NUMBER OF BUFFERS

Explanation: An insufficient number of HASP buffers is available to process the specified job.

The Execution Processor BUFFER GET/ROLL routine was unable to find a DDB with a HASP buffer eligible for the buffer roll process.

System Action: Processing of the specified job will continue when a buffer becomes available from another HASP processor.

Operator Response: Notify system programmer.

System Programmer Response: If this message appears frequently, the number of buffers defined for HASP is insufficient for proper performance.

Note: This message is eligible for output only if the HASPGEN variable &DEBUG was selected at HASPGEN time.

JOB j DELETED

Explanation: The Input Service Processor has deleted the indicated job.

System Action: The job is routed to the Print phase for appropriate action; then the job is purged.

JOB j DUPLICATE JOB NAME - JOB DELAYED

Explanation: The specified job was delayed for execution because a job of the same name was already executing.

System Action: The indicated job will be executed when the job with the same name terminates execution.

JOB j END EXECUTION

Explanation: The specified job has completed execution processing.

System Action: The specified job is queued for action by the Print/Punch Processor.

JOB j ESTIMATED {LINES | CARDS} EXCEEDED BY xxxxx

Explanation: The indicated job has exceeded its estimated number of lines/cards by xxxxx lines/cards.

System Action: The action taken by the system is dependent upon the HASPGEN variable &OUTPOPT. See Section 7.2. Either the job will be cancelled (with or without a dump) or no further action will be taken.

JOB j ESTIMATED TIME EXCEEDED BY xx MINUTES

Explanation: The indicated job has exceeded its estimated real time in the HASP Execution Phase by xx minutes.

System Action: The action taken by the system is dependent upon the HASPGEN variable &TIMEOPT. See Section 7.2. The job will either be cancelled (with or without a dump) or no further action will be taken.

JOB j HELD

Explanation: The indicated job has been placed in HASP Hold Status for one of the following reasons:

1. The Job Card specified "TYPRUN=HOLD".
2. The device from which the job was read was set to hold all jobs.

System Action: None.

Operator Response: The reason why the job was placed in HASP Hold Status should be determined and the job should be released when appropriate for further processing.

JOB j HELD FOR THE FOLLOWING VOLUMES --

text

Explanation: The job indicated has been placed in HASP Hold Status pending availability of the volumes indicated by "text".

System Action: The job is placed in HASP Hold Status and input processing continues.

Operator Response: Insure that the requested volumes are available to be mounted and release the job.

JOB j IS PURGED

Explanation: HASP has completely finished processing the designated job and all HASP facilities belonging to the job are made available for reuse.

JOB j JCT OVERFLOW - OUTPUT LOST

Explanation: The indicated job generated more output data sets than were provided for by HASPGEN. See Section 7.2. The Execution Processor discovered that the JCT for the indicated job could not hold another PDDB representing additional SYSOUT data.

System Action: The job will continue to process normally except those data sets in excess of the maximum will not be printed or punched.

Operator Response: Notify system programmer of condition.

JOB j LOAD 'xxxx' FORMS IN device

Explanation: The indicated job requires that "xxxx" type forms be mounted in the indicated device before output processing can continue.

System Action: Output processing is halted on the specified device until an appropriate operator response is received.

Operator Response: The operator should load the requested forms (or verify that the requested forms are loaded) and enter a start ($S) command for the indicated device. If the operator does not wish to continue processing at this time, either the restart ($E) command, the interrupt ($I) command, or the delete ($C) command will be accepted at this time and the output processor will assume that the requested forms have not been mounted.

JOB j ON device -- jobname programmername

Explanation: A Job Card has been detected in the input stream from the indicated device and the associated job has been assigned a HASP Job Number of "j". The jobname and programmername displayed are the job name and programmer name from the Job Card.

System Action: The previous job (if any) is queued for the execution phase and input service processing is initiated for the new job.

JOB j $\left\{\begin{matrix}\text{PRINTING}\\\text{PUNCHING}\end{matrix}\right\}$ ON device

Explanation: The indicated job is now being processed by the Output Service Processor.

JOB j TERMINATED

Explanation: A permanent I/O error, while reading input from a SPOOL volume for the specified job, was encountered. The nature of the error was displayed by a previous "I/O ERROR ..." message.

System Action: The indicated job is cancelled automatically.

Operator Response: Notify system programmer.

LINEn -- INVALID PASSWORD

Explanation: A Remote attempted to sign-on the specified line with an invalid password.

System Action: The attempted sign-on is not allowed and the line is left in an inactive status.

Operator Action: The remote operator should determine the valid password and correct the sign-on card to reflect this information.

LINEn -- INVALID SIGNON

Explanation: A Remote attempted to sign-on the specified line with an invalid sign-on card. A sign-on card may be invalid if:

1. The Remote name is spelled incorrectly.
2. The Remote specified has not been generated.
3. The Remote specified is attached to another line.
4. The remote name does not begin in column 16

System Action: The attempted sign-on is not allowed and the line is left in an inactive status.

Operator Action: The remote operator should verify the spelling of the Remote. If the Remote is attached to another line, steps should be taken to correct this conflict in Remote assignments.

System Programmer Action: If the required Remote has not been generated another HASPGEN will be required to correct this situation.

r, message from operator

Explanation: The operator at a central console (r=0) or at a remote terminal identified by the value r has entered the displayed message via the $DM (display message) command.

5.0 CONSOLE SUPPORT

HASP provides the installation the option of allowing either HASP or OS to control the local operator console devices. Although the format of the entry of OS and HASP commands is the same regardless of the option selected by the installation, the physical control of the devices differs. The section HASP CONSOLE SUPPORT provides sufficient information for the operator to control HASP console devices, and the section OS CONSOLE SUPPORT provides sufficient supplementary information to the OS Operator's Guide for control of OS consoles.

5.1 HASP CONSOLE SUPPORT

Up to eight locally attached devices may be used as HASP consoles. The following devices may be used as consoles for the type of input-output listed:

1052	printer keyboard	- input and output
1053	printer (on local 2848)	- output
1443	printer	- output
2260	display (on local 2848)	- input and output

Each console device will be assigned a HASP physical device identification which is used to reference the device via HASP commands; the identifications are: CON1, CON2,....

The $DU (DISPLAY UNITS) command may be used to determine the HASP physical device with the hardware address assigned to the device.

CONTROLLING CONSOLE MESSAGE OUTPUT

When HASP is started, all local consoles will be set to display all messages generated by OS, including problem program WTO and WTOR messages, as well as those generated from within HASP. Depending upon the system, it is possible for a large volume of messages to be displayed upon the console devices. A large message volume not only makes it difficult for an operator handling a part of the operator work load to quickly identify messages intended for his use, but it tends to tie up the system waiting on the speed of the slowest console device. HASP provides a means of classifying messages so that each operator can cause only desired messages to be displayed at his console. Each message has one or more logical console classifications:

LOG	-	log console messages
ERROR	-	error messages
UR	-	unit record messages
TP	-	HASP RJE line messages
TAPE	-	tape console messages
MAIN	-	main operator's console messages
OS	-	OS WTO messages

Each message will also have an associated level of importance:

1	-	non-essential messages
3	-	normal messages
4	-	messages requiring action
7	-	essential messages

By appropriate setting of the output classifications of a given HASP console, the operator is able to select only those messages he desires to see. As an example, CON1 is a 1052 and CON2 is a 1443. Because the 1443 is a high speed device, it is allowed to display all messages generated within the system. However, the 1052 is set to display only messages to the main operator, MAIN, at a level of importance above 3. The setting for the 1052 would be accomplished using the following commands:

```
$T CON1,RESET          - turn off all output
$T CON1,3,MAIN         - set level of importance and
                         logical console class
```

If it is desired to assign a console to more than one logical console class, the following command sequences could be used:

```
1.  $T CON1,RESET          - turn off all output
    $T CON1,3,MAIN         - set level and logical class
    $T CON1,ERROR          - add an additional logical class

2.  $T CON1,RESET          - turn off all output
    $T CON1,3,MAIN,ERROR   - set level and both logical classes
```

Setting the HASP console output characteristics applies equally well with multiple or single console options. The only difference is the flexibility achievable in multiple console configurations. Resetting a console although preventing console message output will not, however, prevent responses to HASP commands from being displayed; HASP command responses will always be displayed on the console upon which the command was entered. TABLE 5.1.1 lists the classifications for each message originating from HASP.

In addition to the $T command, the following commands may be used to control console output:

```
$Z CONn    - turn off all output to console (same as RESET)
$S CONn    - turn on all output to console (same as level 0
             and specifying all of the logical console
             classes)
```

TABLE 5.1.1 HASP MESSAGE CLASSIFICATIONS

MESSAGE	LEVEL
"ERROR" CONSOLE MESSAGES	
ALL AVAILABLE FUNCTIONS COMPLETE	7
DISASTROUS ERROR - COLD START SYSTEM ASAP	7
DISASTROUS ERROR DURING CHECKPOINT - RESTART ASAP	7
I/O ERROR ON device uuu,cc,ssss,iiii,bbcchhr	7
I/O ERROR ON LINEn uuu,cc,ssss,iirr,xyee	7
"UR" CONSOLE MESSAGES	
ALL AVAILABLE FUNCTIONS COMPLETE	7
SPOOL VOLUMES ARE FULL	7
JOB j LOAD 'xxxx' FORMS IN device	5
JOB j ON device -- jobname programmername	5
device BACKSPACED	3
device command (excluding remote console devices)	3
device DELETED	3
device FWD-SPACED	3
device REPEATED	3
device RESTARTED	3
device SKIPPING FOR JOB CARD	3
device SUSPENDED	3
JOB j HELD	3
device IS DRAINED	1
JOB j -- ILLEGAL JOB CARD	1
JOB j -- ILLEGAL /*ROUTE CARD	1
JOB j DELETED	1
JOB j {PRINTING / PUNCHING} ON device	1
JOB j PURGED	1
"TP" CONSOLE MESSAGES	
ALL AVAILABLE FUNCTIONS COMPLETE	7
I/O ERROR ON device uuu,cc,ssss,iiii,bbcchhr	7
I/O ERROR ON LINEn uuu,cc,ssss,iirr,xyee	7
r, message from operator (at remote r)	7
device command	3
LINEn -- INVALID PASSWORD	3
LINEn -- INVALID SIGNON	3
REMOTEr DISCONNECTED	3
REMOTEr STARTED ON LINEn	3
device IS DRAINED	1

"TAPE" CONSOLE MESSAGES

ALL AVAILABLE FUNCTIONS COMPLETE 7
device message 5
JOB j
JOB j HELD FOR THE FOLLOWING VOLUMES -- 5

"MAIN" CONSOLE MESSAGES

ALL AVAILABLE FUNCTIONS COMPLETE 7
JOB j BUFFER ROLL UNSUCCESSFUL -- VERIFY NUMBER 7
OF BUFFERS
JOB j JCT OVERFLOW - OUTPUT LOST 7
JOB j TERMINATED 7
SPOOL VOLUMES ARE FULL 7
device message 5
JOB j
INIT i IDLE - CLASS = classes 5
PART
JOB j DUPLICATE JOB NAME - JOB DELAYED 5
JOB j ESTIMATED LINES EXCEEDED BY xxxxx 5
CARDS
JOB j ESTIMATED TIME EXCEEDED BY xx MINUTES 5
JOB j HELD FOR THE FOLLOWING VOLUMES -- 5
JOB j -- jobname -- BEGINNING EXEC - INIT i - class c 3
JOB j AWAITING HASP ALLOCATION PART 3
JOB j HELD 3
JOB j -- EXCESSIVE INPUT STREAM DATA SETS 1
JOB j END EXECUTION 1

"OS" CONSOLE MESSAGES

ALL AVAILABLE FUNCTIONS COMPLETE 7
(all OS and problem program WTO and WTOR requests) 5
HASPWTR - PERM I/O ERR OS JOBQ 5
UNREADABLE OVERLAY - REBUILD OLAYLIB AND WARM START 5

"LOG" CONSOLE MESSAGES

(All messages routed to any other console)

CONTROLLING COMMAND ENTRY

All correctly entered commands will be accepted for action when entered upon the central console of a single console system. However, when multiple local input consoles are available, some of which are accessible to large numbers or inexperienced personnel, it is desirable to limit the authority of one or more of the consoles to control the various functions of the system. HASP consoles may, therefore, be assigned one or more authority groups as follows:

- 0 - display only
- 1 - system control
- 2 - device control
- 4 - job control

Any console may be used to enter HASP display commands; these commands are not deemed to be harmful to the system. However, to control the system from a given console, that console must be authorized for entry of the command; if not, the entry will be rejected with an INVALID COMMAND response or INVALID OPERAND if the command is generally acceptable but use of the operand is unauthorized. "System Control" authorization is required for the entry of any OS command or any command which attempts to alter the authorization of a console.

At HASP initialization each console is given a default authorization; by use of the $DU (DISPLAY UNITS) command each console will be listed and if ACTIVE the sum of the authorizations will be displayed (applicable only for multiple consoles). If a console is eligible for full control, the authorization value is 7 (1+2+4). If a console is eligible for control of jobs and devices, the authorization value is 6 (2+4).

CHANGING CONSOLE AUTHORIZATION

An operator at a console authorized for system control may alter the authority of any <u>other</u> local HASP console in the system via the "$T CONn,A=value" command (value is the sum of the desired authority group numbers). As long as authorization changes are made from HASP consoles, no combination of commands can be entered which will cause all consoles to be unauthorized as a "system control" console.

HASP 2260 OPERATION

HASP 2260 consoles operate in "roll mode" such that available messages replace displayed messages at a specified predetermined rate. In order to enter commands through 2260s, the following procedure must be used:

1. Press SHIFT and ENTER

2. When MI (Manual Input Symbol) appears at the beginning of one of the display lines, the system is ready to accept commands. The console is interlocked such that no further display messages will be processed until the command is entered.

3. Enter the desired command through the 2260 keyboard.

4. To send the command to the system, press SHIFT and ENTER. The command will be read by HASP, the screen made available for display messages.

If mistakes are made during command entry, use the BKSP key and re-type over the incorrect portions, space the cursor beyond the last command character if necessary, then do step 4. To cancel a command without entering it, backspace the cursor until it is immediately to the right of the MI symbol, then do step 4.

The screen <u>should not</u> be cleared by use of the keyboard ERASE when the MI symbol is on the screen. If this is done, the usual symptom will be that the system will not respond with MI when ENTER is pressed. The following special recovery procedure should be used:

1. Re-clear the screen by pressing SHIFT and ERASE.

2. Press SHIFT and START to manually produce the MI symbol.

3. Continue as above from step 3 to enter a command.

HASP 1052 OPERATION

HASP 1052 consoles normally operate in the output mode. The system is free to print messages to the operator whenever a message is ready. In order to enter commands through 1052s, the following procedure must be used:

1. Press the REQUEST key at the right end of the keyboard.

2. When the PROCEED light located above the keyboard glows, enter the desired command using the 1052 keyboard.

3. Upon completion of command entry, enter EOB to indicate completion of entry. (Press top row keys ALTN CODING and numeric 5 simultaneously for EOB)

If mistakes are made during entry, enter CANCEL and do steps 2 and 3 again. (Press top row keys ALTN CODING and numeric 0 simultaneously for CANCEL). To cancel a command after one or more characters have been entered, enter CANCEL and then enter EOB when the proceed light glows.

5.2 OS CONSOLE SUPPORT

HASP utilizes standard OS facilities for displaying information on the OS controlled consoles and accepts HASP commands from OS by monitoring the console inputs. All devices supported by OS continue to be supported when HASP is running in the system.

CONTROLLING CONSOLE MESSAGE OUTPUT

In the process of controlling devices and jobs, HASP originates messages to be displayed on one or more OS consoles. Depending upon the system, it is possible for a large volume of messages to be displayed upon the console devices. A large message volume not only makes it difficult for an operator handling a part of the operator work load to quickly identify messages intended for his use, but tends to tie up the system waiting on the speed of the slowest device. HASP utilizes the OS Multiple Console Support and provides to OS message group routing codes for each HASP originated message (see OS Operator's Guide). TABLE 5.1.1 lists all HASP originated messages with the appropriate HASP logical console classifications. The equivalent OS routing codes are as follows:

LOG	-	MASTER CONSOLE INFORMATION
ERROR	-	SYSTEM ERROR MAINTENANCE
UR	-	UNIT RECORD POOL
TP	-	TELEPROCESSING CONTROL
TAPE	-	TAPE LIBRARY, DISK LIBRARY, TAPE POOL, DIRECT ACCESS POOL
MAIN	-	MASTER CONSOLE ACTION, MASTER CONSOLE INFORMATION

Each HASP message will also have an associated level of importance:

1	-	non-essential messages
3	-	normal messages
4	-	messages requiring action
7	-	essential messages

By appropriate setting of the output routings of the console device, the operator is able to select only the OS messages as well as HASP messages desired. The operator should refer to the OS 360 Operator's Guide for correct use of the OS "VARY unit,CONSOLE" command. The HASP "$T CON" command may be used to set the desired level of importance for HASP originated messages.

CONTROLLING COMMAND ENTRY

In a system running with OS Multiple Console Support, consoles may be physically available to unauthorized personnel. OS provides a facility by which each console is given authorization to enter selected groups of commands. HASP will, when accepting a command from OS, examine the entry console authorization and reject unauthorized entry as an INVALID COMMAND or INVALID OPERAND as appropriate. The OS command authority groups and the HASP equivalents are as follows:

OS GROUP			HASP
0	INFO	-	DISPLAY ONLY
1	SYS	-	JOB CONTROL
2	IO	-	DEVICE CONTROL
3	CONS	-	SYSTEM

The OS "VARY unit,CONSOLE,AUTH" command may be used for the control of the command entry authorization of the OS controlled consoles.

6.0 READER SUPPORT

HASP supports numerous types of devices for entry of Operating System commands, HASP commands, control cards, and user jobs to be executed under control of the HASP/OS environment. Via local attachment to the central CPU the following device types are supported:

IBM 2501 Card Reader
IBM 2540 Card Reader
IBM 24xx Tape Drive (using non-labeled tape with maximum block size set at HASP generation time--if seven track tape written with 800 BPI, odd parity, data convert on)

HASP provides an additional local reader interface enabling programs and system routines to submit commands, control cards, and jobs to HASP as though submitted through a physical reader device. This device-like interface is known as an internal reader (INTRDR) and is controllable through OS and HASP commands in a manner similar to 2540 reader devices. Devices which are connected to HASP remote work stations and supported as readers allow for entry of OS commands, user jobs, and a subset of the HASP commands.

6.1 CONTROLLING HASP READERS

Through the use of HASP operator commands the operator controls the HASP reader devices. Operators at remote work stations may control only those HASP readers which are attached to the remote work station. Commands which control HASP readers are as follows:

command		general use
$C reader	-	Cancel the current job being read on the reader thus causing the reader to skip for the next job or HASP control card.
$P reader	-	Stop HASP from using the reader device for future job streams.
$S reader	-	Start HASP use of the reader device for future job stream input.
$T reader,HOLD	-	Set the reader device to place input jobs in the HOLD status--reset by $S reader
$Z reader	-	Halt the reader device until $S reader is entered

The formal definitions of these commands may be found in the HASP OPERATOR COMMANDS section of this manual.

The following paragraphs discuss special methods of controlling local readers. The remote operator should refer to the operator's guide provided for the supported work station.

HASP LOCAL CARD READERS

Each 2540 or 2501 Card Reader on the system is assigned a HASP name at HASP initialization time; responses to the $DU command display the HASP reader names along with the corresponding hardware addresses.

STARTING HASP LOCAL CARD READERS - There are three methods of causing HASP to begin using a HASP card reader device:

1) Enter the $S reader command when the device is halted, drained, or inactive.

2) Ready the reader with cards prior to replying to the initialization WTOR. This is equivalent to entering a $S reader command.

3) If the Automatic Starting Reader feature is selected by the installation, ready the reader with cards at any time unless the $P reader command has been entered.

If OS has allocated the card reader for other functions when HASP is initialized, there will be no attempt to use the reader for reading jobs unless a $S reader command is entered. To prevent inadvertent OS allocation of the reader to other jobs, HASP simulates an OS vary off-line command prior to its initial use of the device and when each $S reader command is entered.

SHARING HASP LOCAL CARD READERS WITH OS JOBS - Because HASP is a long running job it is desirable for HASP not to prevent OS from allocating the card reader devices to other jobs within the system. The operator is then able to start OS readers to a HASP card reader or enter jobs which require direct reading from a card reader. The operator should observe the following precautionary rules when other jobs are to use HASP reader devices:

1) Enter a HASP $P command for the device and allow the device to become drained before varying the device on-line or replying to OS allocation requests.

2) Insure that the job has finished reading cards and will not attempt to read more cards prior to entering a HASP $S command for the device.

HASP INTERNAL READER

Although the HASP internal readers are not real devices on the system, they may be controlled by the operator in much the same way as real devices. If the operator desires to prevent problem program submission of jobs to HASP, he should enter the OS command:

```
VARY  unit,OFFLINE
```

once for each internal reader. Each unit specified is the three digit address for an internal reader obtainable from HASP when $DU command is entered. OS will issue an allocation request when a user job desires the unit. The operator then has the option of cancelling the job or allowing the device to be assigned.

In addition to the control of OS allocation, the operator can cause all jobs submitted via the internal reader to be placed in the HOLD status via the command:

```
$T  internal reader,HOLD
```

This allows problem programs to submit jobs to HASP but prevents the submitted jobs from executing until the operator specifically releases them.

HASP LOCAL TAPE READERS

HASP support of local tape readers differs from that of the card reader devices in that the tape drive address assignment to a HASP TAPE is specified by the operator by the command:

    $S tape reader,unit

Because of speed and characteristics of tape drives HASP does not allow sharing of the tape device with problem programs; therefore, HASP will not start a tape that is allocated to another function and will prevent OS allocation while in use by HASP.

6.2 HASP INPUT STREAM

The input job streams submitted to the Operating System via HASP follow the conventions and format described in the OS/360 Job Control Language manual. Within these conventions HASP requires that some cards be specified in a particular manner and provides for optional control cards which would appear as comments to the Operating System in systems without HASP. This section discusses the use, format, and placement of these cards.

HASP JOB CARD

The JOB card is a variable-field control card which defines the beginning of a job (and, of course, the end of the previous job if there is one) within the input stream. In addition, certain parameters are passed to HASP and to the Operating System via fields and subfields punched into the JOB card.

The format of the JOB card is basically as defined in the Job Control Language Manual. In particular, HASP requires that the accounting information field be punched in the following format:

(pano,room,time,lines,cards,forms,copies,log,linect)

where:

pano = Programmer's accounting number. This subfield MUST BE PRESENT and must consist of one to four alphameric characters. (Example: "4301")

room = Programmer's room number. This subfield MUST BE PRESENT and must consist of from one to four alphameric characters. (Example: ",E305")

time = Estimated execution time in minutes. This subfield is optional and may consist of up to four numeric digits. If omitted, a standard value will be assumed. (Example: ",30" for 30 minutes)

lines = Estimated line count in thousands of lines. This subfield is optional and may consist of up to four numeric digits. If omitted, a standard number of lines will be assumed. (Example: ",5" for 5000 lines)

cards = Estimated number of cards to be punched. This subfield is optional and may consist of up to four numeric digits. If omitted, a standard number of cards will be assumed. (Example: ",200" for 200 cards to be punched)

forms = Special forms for printing entire job. This subfield is optional and may consist of up to four numeric characters. If omitted, standard forms "STD." will be assumed. (Example: ",5" for 5-part forms)

copies = Number of times the print output is to be printed. This subfield is optional and may consist of up to two numeric digits. If omitted, one copy will be assumed. (Example: ",2" for two copies)

log = HASP System Log option. This subfield is optional and may consist of one character. If this character is an "N", the HASP System Log will not be produced. If any other character, or if omitted, the log will be produced.

linect = Lines to be printed per page. This subfield is optional and may consist of up to two numeric digits. If coded as "0" (zero) no automatic overflow will be produced. If omitted, a standard value will be assumed. (Example: ",34" for 34 lines per page)

The other fields on the JOB card are also interpreted for accounting purposes and Job control.

The job card may be continued in accordance with the Operating System Job Control Language specifications.

To omit a specific subfield, the comma normally punched following the subfield should be punched in the first column of the subfield. To omit the remainder of the subfields, the closing right parenthesis should be punched following the last subfield entered.

The following would be a typical JOB card:

```
//ORBIT    JOB (7808,E305,,2,200),         CONTINUED
//               'J. JACKSON',MSGLEVEL=1,CLASS=B
```

In this case:

pano = 7808

room = E305

time = 2 minutes (assumed value)

lines = 2000 lines

cards = 200 cards

```
forms  =  standard forms (assumed)

copies =  1 copy (assumed value)

log    =  yes (assumed value)

linect =  standard value (assumed)
```

HASP PRIORITY CARD

The PRIORITY card is a fixed-field control card used to assign a set priority to a job. The format of the card is as follows:

```
Columns    1 - 10    --    /*PRIORITY
          11 - 15    --    blank
          16 - 17    --    p(left justified)
          18 - 80    --    ignored
```

where "p" is either a number (between 0-15) or the character "*". If "p" is a number, the value of "p" will be assigned as the priority of the job following the PRIORITY card. If "p" is the character "*", or if the PRIORITY card is not present, the priority of the job will then be determined by the estimated execution time and the estimated lines on the JOB card.

The PRIORITY card must immediately precede the JOB card. If it does not, the PRIORITY card will be ignored and the input stream will be flushed until a job card (or another PRIORITY card) is found.

HASP ROUTE CARD

The ROUTE card is a fixed-field control card which allows the user to specify the location to which his output is to be printed or punched. The format of the card is as follows:

```
Columns    1 -  7    --    /*ROUTE
           8 -  9    --    blank
          10 - 14    --    PRINT or PUNCH
               15    --    blank
```

16 - 23	--	one of the following device specifications:	
		LOCAL	-- Any local device
		REMOTEn	-- Remote Terminal "n"
		PRINTERn	-- Printer "n"*
		PUNCHn	-- Punch "n"*
24 - 80	--	ignored	

A single ROUTE card can be used to direct either the print or punch routing but not both. If both print and punch are to be routed, two cards must be used.

The ROUTE cards should be placed immediately after the JOB card.

* NOTE: The PRINTERn and PUNCHn specifications are the same as LOCAL unless the specified printer or punch is subject to local print/punch routing.

HASP MESSAGE CARD

The MESSAGE card is a fixed-field control card which permits the user to send messages to the operator via the operator console at HASP job input time. The format of the card is as follows:

Columns	1 - 9	--	/*MESSAGE
	10 - 11	--	blank
	12 - 71	--	message to be written
	72 - 80	--	ignored

All leading and trailing blanks are removed from the message before writing it on the console.

If MESSAGE cards are included as part of a job they should be placed immediately following the JOB card (or after any ROUTE cards). In such cases the job number is appended on the front of the message(s).

If a MESSAGE card is not included within the boundaries of a job, the input device name is appended on the front of the message.

HASP SETUP CARD

The SETUP card is a variable-field control card which permits the user to indicate the need for certain volumes during the execution phase of his job. The format of the card is as follows:

Columns:	1 - 7	--	/*SETUP
	8 - 15	--	blank
	12 - 71	--	volume identifiers separated by commas (i.e., vvvvvv, wwwwww, xxxxxx, ...)
	72 - 80	--	ignored

The volumes required are listed on the console at the time that the job enters the system. The job is then placed in "hold" status pending subsequent release by the operator when the required volumes are available.

The SETUP card should be continued with MESSAGE cards and placed with the ROUTE and other MESSAGE cards after the JOB card.

HASP COMMAND CARD

The COMMAND card is a "variable-field" control card used to enter HASP operator commands into the system. The format of the card is as follows:

Columns:	1 - 3	--	/*$
	4 - 71	--	operator command verb and operands
	72	--	If "N" the command will not be repeated on the operator's console.
	73 - 80	--	ignored

Restrictions concerning commands which can be entered from remote terminals are listed under the HASP OPERATOR COMMANDS section of this manual.

All COMMAND cards must be placed in the input stream prior to any JOB card. COMMAND cards within jobs will be ignored.

OS COMMAND CARD

The OS command card is a variable-field control card, the format of which is described in the OS/360 Operator's Guide. This card, if submitted through the HASP input stream, must fall within a job of the input stream and is passed to OS at the time the job is submitted for OS execution. The acceptability of the OS COMMAND CARD is determined by the system programmer when creating the HASP reader procedure on the SYS1.PROCLIB data set.

6.3 LOCAL READER ERROR PROCEDURES

Unrecoverable errors encountered while reading jobs and SPOOLING the data to direct access devices will result in an error message to the operator and the deletion of the job being read. The operator should re-submit any job so deleted in its entirety to HASP.

Errors on local readers such as read checks, feed stops, etc. will be processed by the Operating System. The operator should follow the procedures described in the appropriate component description manual for the device as supplemented by the OS/360 Operator's Guide. Since HASP selects cards read by the IBM 2540 in pocket 2, cards which are non-processed run out (NPRO) will be separated from those read, the last card in pocket 2 being the card in error on data and validity checks.

7.0 PRINT AND PUNCH SUPPORT

HASP supports numerous printer and punch devices for the output of HASP System Log messages, Operating System messages and problem program SYSOUT data sets. Via local attachment to the central CPU the following devices are supported as printer or punch devices as appropriate:

IBM 1403 PRINTER
IBM 3211 PRINTER
IBM 2540 PUNCH
IBM 1442 PUNCH
IBM 2520 PUNCH

7.1 CONTROLLING HASP PRINTER AND PUNCH DEVICES

Through the use of HASP operator commands the operator controls the HASP printer/punch devices. Operators at remote work stations may control only those HASP printer/punch devices which are attached to the remote work station. Commands which are defined for direct control of HASP printer/punch devices are as follows:

command		general use
$B printer	-	Backspace the printer the designated number of pages or to the beginning of the current data set.
$C device	-	Cancel the current job output on the indicated printer or punch.
$E device	-	Restart the job output currently printing or punching on the indicated device, placing the job back on the corresponding queue for selection by the indicated device or other printer or punch, as appropriate.
$F printer	-	Forward-space the indicated printer the designated number of pages or to the end of the current data set.
$I printer	-	Interrupt the current job output on the indicated printer, allowing the output to be continued by the indicated or other printer as appropriate.
$N device	-	Repeat the job output currently printing or punching on the indicated device, placing the job back on the corresponding queue for selection by the indicated device or other printer or punch as appropriate while allowing the current job output to continue.
$P device	-	Stop the printer or punch after completion of the current job output.
$S device	-	Start the printer or punch device.
$T device	-	Set device characteristics.
$Z device	-	Halt the printer or punch device until $S device is entered.

The formal definitions of these commands may be found in the HASP OPERATOR COMMANDS section of this manual.

STR as well as non-MULTI-LEAVING BSC remote work station operators will find that for practical purposes only the $P, $S, and $T commands are available for direct control of printer or punch devices from the work station. Commands entered from these work stations can only be entered when the printer and punch devices are not ACTIVE. This is true even when the non-MULTI-LEAVING BSC work station printer is manually interrupted simulating the $I device command.

CONTROLLING HASP LOCAL PRINTER AND PUNCH DEVICES

Each printer and punch on the system is assigned a HASP name at HASP initialization time; responses to the $DU command display the HASP printer and punch device names along with the corresponding hardware addresses.

STARTING HASP LOCAL PRINTER AND PUNCH DEVICES - There are two methods of causing HASP to begin using a HASP printer or punch device:

1. Enter the $S device command when the device is halted or drained.

2. Ready the printer or punch device prior to replying to the HASP initialization WTOR. This is equivalent to entering a $S device.

If OS has allocated the device for other functions when HASP is initialized, there will be no attempt to use the device for job output unless a $S device command is entered. To prevent inadvertent OS allocation of the printer or punch to other jobs, HASP simulates an OS vary off-line command prior to its initial use of the device and when each $S device command is entered for the printer or punch.

SHARING HASP LOCAL PRINTER AND PUNCH DEVICES - Because HASP is a long running job, it is desirable for HASP not to prevent OS from allocating the printer or punch to other jobs within the system. The operator is then able to start OS writers to a HASP printer or punch device or enter jobs which require direct output. The operator should observe the following precautionary rules when other jobs are to use HASP printer or punch devices:

1. Enter a HASP $P command for the device and allow the device to become drained before varying the device on-line or replying to the OS allocation requests. This may be supplemented by the $I printer or $E device command to insure rapid termination of the current job activity.

2. Insure that the job has finished with the device and will not attempt to output more data prior to entering a HASP $S command for the device.

7.2 HASP OUTPUT ROUTING

Under the standard HASP System, output routing has meaning only when the HASP remote job entry feature is being used. Under this environment each group of printer or punch devices is considered a pool of output devices identifiable by routing codes. All local printer and punch devices are assigned route code zero (0), all printer and punch devices at work station REMOTE1 (RM1.PRn,RM1.PUn) are assigned route code one (1), etc. A job which has its print output destined to local printers will be printed on any of the local printers. Likewise, a job which has its print output destined to remote 4 will be printed on any of the printers assigned to REMOTE4 (RM4.PR1,RM4.PR2,etc.)

HASP will automatically assign print and punch output routings to each job as it enters the system. This assignment is determined by the system programmer at HASP generation time. Normally all output for jobs entering local devices will be routed to the local device pool and all output for jobs entering a remote reader will be routed to the corresponding remote output devices. This may be altered so that, for example, remotes without punch devices will have punch data routed to the local punch pool or to a remote convenient to the submitting work station.

Routing of print and punch output may be directly assigned by the programmer via /*ROUTE control cards (see READER SUPPORT) or by the operator after the job has entered the system via the $R (ROUTE) command. Although the central operator has complete routing control over jobs, the remote work station operator may only route jobs which belong to the remote, i.e., jobs which have the print or punch routings destined for output at the remote. The following sample command sequence allows the operator to redirect the print output for a job after printing of the data sets is in progress:

1. $R PRT,JOB25,LOCAL - Sets the print routing for job 25 to the central printer pool.

2. One of the following (assume job 25 is printing on remote 3 printer 1):

   A. $I RM3.PR1 - Interrupt print output and requeue for continuing the print by a LOCAL printer.

   B. $E RM3.PR1 - Restart print output and requeue for printing by a LOCAL printer.

   C. $N RM3.PR1 - Repeat the print allowing a LOCAL printer to print a copy.

7.3 HASP SPECIAL FORMS ROUTING

At HASP initialization HASP assumes that the printer and punch devices are loaded with the standard forms paper or cards as appropriate. Normal operation of the devices calls for each printer or punch device to select the highest priority job in the appropriate print or punch queue and begin outputting. Assuming that the installation selects SYSOUT Class A to be standard print output and SYSOUT Class B to be standard punch output, all "SYSOUT=A" data sets will be printed on the standard forms paper and "SYSOUT=B" data sets will be punched on the standard forms cards (see OS JOB CONTROL LANGUAGE manual for the meanings of SYSOUT=A or SYSOUT=B). Occasionally the programmer will desire to have a data set printed or punched using special forms and submits a "SYSOUT=(A,,form#)" or "SYSOUT=(B,,form#)" parameter for the Data Definition (DD) card describing the data set. When the data set is encountered during output HASP will stop the printer or punch and display a forms load message on the operator's console. This allows the operator to load the forms desired and enter a $S device command to signify that the device is ready. When output of the data set is complete, HASP will request that standard forms be loaded and wait for the operator as before. The normal mode of operation is therefore the loading of forms on a DEMAND basis.

Occasionally the programmer will decide that all print data is to be printed on special forms and instead of specifying the forms on the "SYSOUT=A" parameter of the DD card, he specifies the forms in the HASP accounting field of the JOB card (see HASP INPUT STREAM section of this manual). This causes the forms designated to be made standard for the printing of the job.

SUBMISSION OF SPECIAL FORMS DATA SETS

Processing special forms on a DEMAND basis, while convenient when occasional need for special forms exists, will cause poor printer or punch utilization when a large number of data sets require special forms. Assuming that the installation selects SYSOUT Class J to be special forms print and SYSOUT Class K to be special forms punch, all "SYSOUT=(J,,form#)" and "SYSOUT=(K,,form#)" data sets will not be printed or punched with the standard output for the job. The programmer therefore designates special forms on the appropriate DD cards in the job input stream. HASP will print the normal print data sets and queue the job for the printing of special forms data sets. After the special forms printer(s) completes all special forms printing, the job will proceed to special forms data set punching, then to standard job punching as appropriate.

ASSIGNING SPECIAL FORMS TO A PRINTER OR PUNCH

The operator may determine the number of jobs with output for special forms by entering the command:

$D F	-	Display Number of Jobs Queued on Forms

When sufficient output is awaiting special forms, the operator may choose to activate one of two types of special forms control by command as follows:

$T device,F=AUTO	-	Activate printer or punch special forms allowing HASP to determine which special forms should be loaded
$T device,F= {forms# / STD.}	-	Activate printer or punch special forms using the forms indicated and loaded by the operator (operator-controlled)

If the device is a printer, special forms jobs (forms indicated in the JOB card) along with data sets which have been disassociated from other jobs will be selected for printing. If a normal SYSOUT class (SYSOUT=A as described previously) with a special forms specification is encountered a DEMAND load for the forms is requested, requiring the operator to cancel the print or load the forms and enter $S device.

Printing and punching of output will proceed until the queue for the forms indicated by the operator or asked for by HASP is empty. If AUTO was indicated by the operator HASP will select jobs awaiting another special forms for the device, ask for the loading of the new forms, and attempt to exhaust the queue of the new forms upon receiving the appropriate $S device command.

The operator may cause the device to revert to standard output by entering:

$T device,F=RESET

NOTES: 1. The command $T device,S=YES or $T device,S=NO may be used to indicate separator pages or cards between job output on the device.

2. The non-MULTI-LEAVING remote workstation operator will find that the $S command may not be entered from the remote to signal that forms have been loaded and that no messages to the operator will be printed on a printer set to output special forms. Therefore the following rules are recommended:
    a. Use only operator-controlled special forms.
    b. Prevent users from requesting DEMAND loading of forms.
    c. After exhausting a special forms queue enter $T device, $=Y (if required), enter a $DF command and, after receiving all messages, set to the next forms type desired.

3. For safe forms changing operations when using operator controlled forms, the operator should stop the device ($P device), load the new forms, tell HASP ($T device), and then start the device ($S device).

7.4 HASP PRINT AND PUNCH OUTPUT FORMATS

HASP PRINT FORMAT

The format for standard print output for each job stream is as follows:

1. HASP START JOB SEPARATOR PAGE
2. HASP SYSTEM LOG (OPTIONAL)
3. HASP STATISTICS
4. OPERATING SYSTEM MESSAGES
5. DATA SETS CREATED BY THE JOB
6. HASP END JOB SEPARATOR PAGE

HASP START JOB and END JOB separator pages consist of a single line of information duplicated a number of lines as specified by each installation. The format of the information line is as follows:

columns	contents
1 - 17	HASP identification
18 - 22	periods (.)
23 - 31	START JOB .CONT JOB ..END JOB
32 - 35	job number assigned by HASP
36 - 40	periods (.)
41 - 51	time of printing the page in form: hh.mm.ss AM/PM
52 - 61	date of printing the page in form: day month year
62 - 65	periods (.)
66 - 69	ROOM
70 - 74	room number
75 - 78	periods (.)
79 - 86	OS jobname
87 - 90	periods (.)
91 -115	programmers name padded with trailing periods (.)
116 -132	HASP identification

The HASP statistics is a single printed line which contains the following information:

1. cards read
2. lines printed
3. cards punched
4. execution time (real time)

HASP PUNCH FORMAT

The format for standard local IBM 2540 punch output for each job stream is as follows:

1. HASP PUNCH ID CARD - in pocket 2
2. DATA SETS CREATED BY JOB - in pocket 2
3. HASP JOB ACCOUNTING CARD - in pocket 3
4. BLANK CARD - in pocket 1 (also will contain error cards)

HASP JOB ACCOUNTING CARD FORMAT

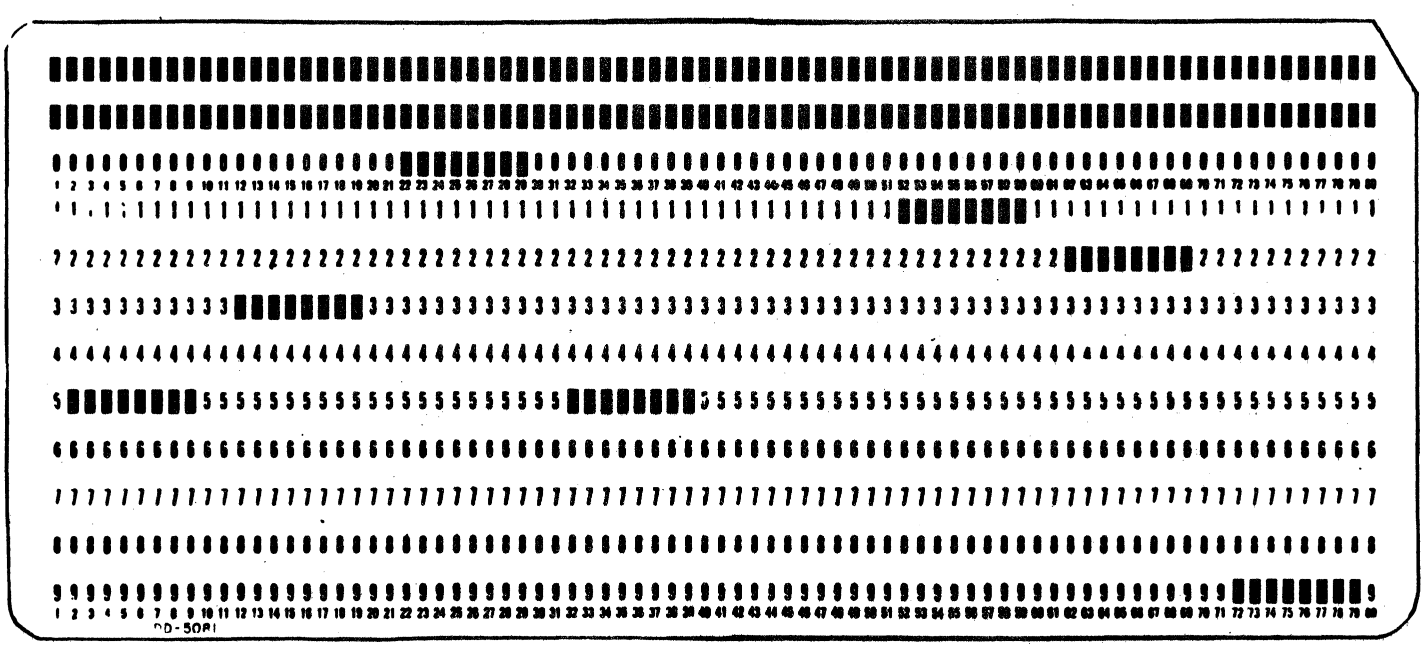
Columns	Contents	Mode
1 - 20	Programmer's name	EBCDIC
21 - 24	Room number	EBCDIC
25 - 27	Spares	N/A
28 - 31	P. A. number	EBCDIC
32	Job priority number	BINARY
33 - 35	Job input time in hundredths of a second	BINARY
36 - 38	Job output time in hundredths of a second	BINARY
39 - 40	Number of cards read in	BINARY
41 - 43	Number of output lines	BINARY
44 - 45	Number of output cards	BINARY
46 - 48	Total reader time in hundredths of a second	BINARY
49 - 51	Total execution time in hundredths of a second	BINARY
52 - 54	Total print time in hundredths of a second	BINARY
55 - 57	Total punch time in hundredths of a second	BINARY
58 - 65	Job name	EBCDIC
66 - 71	Spares	N/A
72	Identifier (X'FF')	BINARY
73 - 74	Year	EBCDIC
75 - 77	Days	EBCDIC
78 - 80	Job number	EBCDIC

HASP PUNCH ID CARD FORMAT

Each job's punch output will be preceded by an identification card containing the programmer room number and internal job number. To make the card easy to identify, it has an 11-punch and a 12-punch punched in all 80 columns. To make the room number and job number easy to read, each digit is extended over ten columns. Alphabetic characters are converted to digits as follows:

Alphabetic Characters	Numeric Punch
A or J	1
B, K, or S	2
C, L, or T	3
D, M, or U	4
E, N, or V	5
F, O, or W	6
G, P, or X	7
H, Q, or Y	8
I, R, or Z	9

Below is an example of the punch identification card which would precede a deck punched, for example, for a programmer residing in Room E305, and having an internal job number of 129.



7.5 LOCAL PRINTER AND PUNCH ERROR PROCEDURES

All job printing and punching for local devices is accomplished through Operating System facilities. OS will attempt to recover from printer and punch errors and provide appropriate error messages to the operator. In the case of permanent errors the following procedures apply:

PRINTER - Permanent errors will be ignored and output will continue. Since the accuracy of the output is determined by the presence or lack of error messages, the operator should react in accordance with the severity of the problem.

PUNCH - Error cards are dropped in pocket 1 and punching continues starting with the record detected to be in error.

(The remainder of this page intentionally left blank.)

11.2 HASP REMOTE TERMINAL PROCESSOR (MODEL 20/STR) OPERATOR'S GUIDE

The following section contains detailed instructions for operating a 360/20, equipped with a Synchronous Transmit-Receive (STR) communication adapter, as a remote terminal system under HASP. Although intended for use as a separate operational manual, it has been included into the HASP SYSTEMS Manual to achieve completeness.

(The remainder of this page intentionally left blank.)

H A S P

REMOTE TERMINAL PROCESSOR

FOR

STR COMMUNICATIONS

MODEL 20 OPERATOR'S GUIDE

TABLE OF CONTENTS

1.0 INTRODUCTION

The HASP System is an automatic spooling, priority scheduling system which, while operating in conjunction with OS/360, operates an unlimited number of peripheral devices simultaneously with normal job execution, to perform the functions normally associated with off line support computers. The function of HASP has been extended to operate, via several classes of telephone lines, peripheral devices located remotely from the central computer complex.

Through the use of the HASP Remote Job Entry feature, a user, located perhaps thousands of miles from a particular System/360 installation, can utilize the capabilities of that installation much as if it were in the local computer room. The unit record devices at a remote station are logically operated by HASP as if they were local readers, printers and punchers, so that HASP can simultaneously, while operating all local unit record devices, read jobs from several remote readers into the queue of jobs awaiting processing and upon completion of the processing, can print and punch the results at the remote site.

Although a variety of devices may be utilized as remote terminals, this document discusses only the use of a System/360 model 20 as a remote station.

A special program has been written for the model 20 which can be considered as a logical extension of the HASP SYSTEM. This program, called the HASP Remote Terminal Processor (HASP/RTP) performs the following functions:

A. INPUT

1. Reads cards from the card reader attached to the model 20.

2. Compresses, blocks and encodes the card images for transmission, over telephone lines, to the central computer facility.

3. Maintains synchronization and communication with HASP for transmission.

B. PRINT AND/OR PUNCH

1. Establishes and maintains synchronization and communication with HASP to receive transmissions of job output.

2. Decodes and decompresses print and/or punch records received.

3. Interprets and executes carriage control information in the case of print.

4. Prints and/or punches the received data.

HASP/RTP may either Read, Print or Punch but may not perform any two operations simultaneously.

Due to the use of blocking and character compression to minimize line transmission time, the speed at which the model 20 devices are operated is dependent on the data being transmitted. Certain jobs, because of their data characteristics, will enable HASP/RTP to operate the model 20 devices at full rated speed. Other jobs, with less advantageous data characteristics, may cause the devices to operate at less than full speed.

2.0 OPERATING PROCEDURES

The following pages provide sufficient information for initiating and operating the HASP/Remote Terminal Program.

2.1 INITIAL PROGRAM LOAD

1. Ready the HASP/RTP deck in a reader on the model 20. The deck should include as the last card an appropriate "configuration" card as described in Section 4.0.
2. If the model 20 has multiple readers, the reader select switch on the console should be set to indicate the reader containing the RTP deck.
3. Ready the system printer and punch (if present).
4. Set the TIME SHARING key to the on (down) position.
5. Set the BINARY/BCD switch on the communications adapter to the BINARY position.
6. Set the LINE SPEED key to the appropriate speed.
7. Set the AUTO-CALL key to the OFF position.
8. Verify the setting of the full duplex (FD) - half duplex (HD) switch inside the CE console.
9. Turn the communications adapter switch to the NORMAL position.
10. Set the DATA KEYS to 009C.
11. Set the MODE switch to PROCESS.
12. Press the I/O CHECK RESET, SYSTEM RESET and LOAD key on the model 20 console.

10. Press the STOP key on the Reader Punch and signal HASP in the central computer of the end of input by pressing the EOT key.

NOTE 1: Certain installations may, if all input card characters are of the 64 BCD character set, direct the setting of the operation mode switch to the SEND 1st CHAR position, in order to improve the transmission rate.

2.2 ESTABLISHMENT OF COMMUNICATIONS LINE

1. Advise the operator of the central computer location that job transmission is to be initiated. (The central computer operator must authorize HASP to process jobs from a given remote terminal. This authorization must be given only once per terminal.)

2a. Leased (or private transmission lines) - place the data set in the DATA mode.

2b. Dial-up transmission lines - Press the TALK button on the data switch and dial, exactly as in a normal call, the number of the appropriate data set at the central computer. After the "ringing" sound, a shrill sound indicates that the phone has been answered. Immediately upon hearing this sound, press the DATA button on the data set and hang the telephone up.

3. The CHARACTER PHASE light on the communications adapter should appear to indicate synchronization of the two computers.

NOTES on communication line establishment:

a. A "busy" signal indicates that the called data set is in use.

b. A line may be established by a call from the central computer to the remote site. To receive such a call, normal initialization procedures should be followed but rather than dialing, the AUTO button should be pressed and the phone hung up to await the call.

c. On a two-wire half-duplex telephone line, a period of several seconds may be registered to synchronize the two computers.

d. On the establishment of a connection other than the first, a period of several seconds may be required to begin transmission of print and/or punch data.

2.3 INITIATING PROCESSING

In order to allow the remote computer operator to select the function the Model 20 is to perform (i.e., input or output), the transmission of jobs to the central computer site is given priority. At the end of the print and punch for a job,HASP/RTP tests the system card reader for ready status and, upon finding it ready, immediately begins the transmission of all jobs in the reader to the HASP job queue in the central computer. If no print or punch jobs are available to be processed, the program maintains communications with HASP in the central computer to await a job. During this dormant period, the reader is tested every several seconds for the availability of a job to transmit. Thus, the Model 20 operator by merely placing a job (or jobs) in the card reader can cause transmission to the central computer at the next "end of job" (or within several seconds if no processing is active). The lack of jobs in the reader will therefore cause all print and punch output from the central computer to be processed as it becomes available.

3.0 ERROR RECOVERY

The following sections list some of the more common error conditions that may arise and indicate a solution to each.

3.1 COMMUNICATION ADAPTER ERRORS

Any errors concerned with data transmission are indicated by the communications adapter on the Model 20 by halting (stop light on), sounding the audible alarm, and displaying a combination of error indicators. Normally, an error stop indicates a line transmission failure (after three retries). By pressing start, the transmission will be retried three additional times. If a failure still results after several retries, the computing system and the telephone line should be checked. A detailed description of the meaning of various error indicators is given in the IBM Reference manual A26-5847.

NOTE: occasionally certain error lights on the communications adapter will flash on for brief periods of time. No action should be taken until the CA stop key is lighted.

3.2 UNIT RECORD DEVICE ERRORS

All error conditions on unit record devices will be indicated by the illumination of the appropriate ATTENTION light on the Model 20 console and a program stop in the Model 20 with an indicative number displayed in the ESTR Register. When the error condition has been cleared (as described subsequently), the START key on the console should be pressed to resume operation. The following sections describe the error identification halt codes and the specific actions necessary to correct an error condition on all supported devices.

3.2.1 1403/2203 Printer

SYNC CHECK/PRINT CHECK/FORMS CHECK— All printer checks will be effectively ignored by HASP/RTP. The error condition should be reset and the printer put into READY status to continue printing. If the malfunction caused the loss of print lines, the central computer operator should be contacted and advised to BACKSPACE* or RESTART* the print to recover the lost lines.

---

* See the HASP SYSTEM Operator's Guide

3.2.2 2501, 2560, 1442, 2520 CARD READERS

All card reader checks are indicated by a program halt code of 0001. To retry the faulty read, the cards should be run out of the reader and the last two cards in the receiving stacker should be placed back into the reader.* Processing may then be resumed by depressing the START keys on both the card reader and the model 20 CPU. If the read check condition persists after several retries, the validity of the card should be checked.

Feed Stops and other mechanical problems on card readers are indicated by the illumination of an error light on the reader console (along with the appropriate ATTENTION indicator on the Model 20 CPU). This condition may be corrected by running out the reader, correcting the cause of the stop, and replacing the cards into the reader. Note that no cards are removed from the receiving stacker. Pressing the START key on the card reader will cause processing to resume.

The occurrence of both types of error conditions simultaneously should be corrected by following the procedure for reader checks.

* In the case of the 2560 (primary stacker) the last THREE cards should be placed back in the read hopper.

3.2.3 2560, 1442, 2520 CARD PUNCHES

All punch checks will be indicated by a program halt code of 0002. To repunch the card in error, the cards in the punch should be run out and the last two cards in the receiving stacker discarded. Blank cards should be placed into the punch and START depressed on the punch console. The pressing of the START key on the CPU will cause processing to resume.

Punch STOPS and other mechanical problems will be indicated by an indicator light on the punch (and by the appropriate ATTENTION indicator on the CPU console). These conditions should be corrected by clearing the punch and discarding all non-processed cards. Processing will resume when the punch is re-readied. The occurrence of both types of the above errors simultaneously should be corrected by following the punch check correction procedure.

3.3 MODEL 20 RESTART

In the event of an untimely interruption of Model 20 operation such as a machine, program, communication line, or environmental failure, the following procedures should be utilized to resume processing:

A. Model 20 transmitting at failure — HASP/RTP should be reloaded with the complete job which was being sent at the time of the failure immediately behind the HASP/RTP deck. Due to the sometimes extensive buffering of cards by the Model 20, doubts concerning which job was being transmitted at the time of the failure should be resolved by contacting the operator at the Central site. The central operator should also be advised to enter the HASP RESTART RMT n command to delete this partially complete job from the HASP job queue. After normal initialization procedures, processing should resume.

B. Model 20 RECEIVING AT FAILURE — HASP/RTP should be reloaded with NO input jobs in the card reader to force it into the receive mode again. Since a system failure will normally result in the loss of some amount of data, the central computer operator should be advised to BACKSPACE or RESTART the function in progress as required by the amount of data lost.

4.0 DYNAMIC CONFIGURATION SPECIFICATION

The HASP/Remote Terminal Processor can utilize any of the types of peripheral I/O equipment that can normally be ordered with the Model 20. At program load, HASP/RTP either determines or is instructed by the operator, what devices to use. Either the 2203 or 1403 printer will automatically be used and need not be indicated. The card reader utilized will be the one from which the RTP deck is loaded. The punch to be used must be indicated by the following card which must follow the program deck:

CARD COLUMNS			
01	12	16	80
//SYSPUNCH DD		UNIT=XXXX DUMMY	

where	XXXX	=	2560S	(MFCM Secondary station)
		=	2520	
		=	1442	

If the variable field contains the word DUMMY, all punch output received from the central computer will be ignored with no indication to the Model 20 operator.

5.0 CENTRAL COMPUTER CONTROL

Certain of the control cards recognized by HASP can be introduced from the remote terminal site. Following is a list and meaning of these control cards.

```
1             12                  71
/*MESSAGE     Any Message
```

The data punched into columns 12-71 of this card will be displayed on the central computer operator's console at the time the job is being read into the system. This may be used to identify certain jobs, give special instructions, etc. The /*MESSAGE card may be placed anywhere within the input job stream. If this card appears within a job, the HASP number assigned to that job will be appended to the message before displaying it, otherwise the remote station ID will be appended.

```
1             10                  16
/*ROUTE       PRINT               LOCAL
              PUNCH
```

This card, when included anywhere within a job being submitted to the central computer, will cause the print or punch output (as indicated in column 10) to be processed on local unit-record equipment. This card may be used to divert large volumes of print or punch to local high speed devices to avoid terminal congestion. Both print and punch may be routed locally by including two /*ROUTE cards in a job.

```
1                16
/*PRIORITY       nn
```

This card may be used to force the assignment of priority "nn" to the job which immediately follows. "nn" may be any digit or digits from 0-15. This control card when read locally by HASP is interrupted as an absolute priority assignment to a job. However, when read from a remote station the card is regarded as a priority assignment to this job relative to other jobs from the same station. Thus a remote operator can, via the /*PRIORITY card order the sequence of jobs submitted from only his station, for example, a /*PRIORITY 15 (where 15 is the highest priority) would cause its job to be the next job from that remote station to be processed, although not necessarily the next job to be processed by the central computer. The relative position of the priority structure of a remote terminal with respect to the overall system priority structure is determined at HASPGEN by central computer personnel.

The /*PRIORITY card must immediately precede the OS/360 JOB card of the job to which it refers.

1	16	25
/*SIGNON	REMOTEn	Password

This card appears at the end of the HASP/RTP program deck in front of the //SYSPUNCH configuration card and is used to override the remote identification number normally assigned to the HASP/RTP program deck. For DIAL lines the /*SIGNON card may be used to submit a password which, if correct, will allow the remote terminal access to the HASP system for remote job stream processing. The value "n" must match the remote identification number assigned to the remote station by central computer personnel. The value of the "password" must match the password assigned to the line by the central computer operator when the communication line is "started".

1
/*SIGNOFF

This card is used to inform the central system that the remote terminal operator desires to terminate a remote job stream processing session. When submitted to the central system, HASP will, at the completion of the current print and/or punch streams, disconnect the terminal from the system and prepare the line for other remote stations to SIGN-ON.

1
/*command

Selected HASP commands may be submitted to the central system through the remote terminal card reader. Commands submitted in this manner must

be the first cards of a job stream (in front of the first job submitted). Commands which can be submitted are listed in the HASP operator's guide and must start in column 3 of the card, i.e. the first 3 columns will be "/*$".

```
1         12                                      71
/*SETUP   volume-ser1,volume-ser2,...,volume-sern
```

The volume serials punched in columns 12-71 of the card will be displayed on the central system console and the associated job will be placed in HOLD status (not be scheduled for execution) until released by the central operator. The /*SETUP card appears in the corresponding job input deck between the OS JOB card and the first EXEC card. To continue a /*SETUP card, a /*MESSAGE card should be used.

6.0 OPERATIONAL HINTS

1. It is suggested that the remote terminal operator become familiar with normal HASP operating procedures at the central computer site. The HASP OPERATOR'S GUIDE is contained as section 11.1 in the HASP SYSTEM MANUAL

2. During dormant periods, the Model 20 should be allowed to maintain communication with HASP at the central computer site so that printing (and/or punching) may begin as it becomes available.

3. The communications line may be disconnected at any time, which will cause HASP to hold all jobs awaiting the terminal until the line is again established. This will allow the Model 20 to be used for other purposes during long dormant periods.

11.3 HASP/REMOTE TERMINAL (1978) OPERATOR'S GUIDE

The following section contains detailed instructions for operating an IBM 1978 used as a remote terminal station with the HASP SYSTEM. Although intended for use as a separate operator's manual, it has been included in the HASP SYSTEMS Manual to achieve completeness.

(The remainder of this page intentionally left blank.)

THE

HASP

SYSTEM

IBM 1978 REMOTE STATION OPERATOR'S GUIDE

TABLE OF CONTENTS

The HASP System is an automatic spooling, priority scheduling system which, while operating in conjunction with OS/360, operates an unlimited number of peripheral devices simultaneously with normal job execution, to perform the functions normally associated with off line support computers. The function of HASP has been extended to operate, via several classes of telephone lines, peripheral devices located remotely from the central computer complex.

Through the use of the HASP Remote Job Entry feature, a user located perhaps thousands of miles from a particular System/360 installation can utilize the capabilities of that installation much as if it were in the local computer room. The unit record devices at a remote station are logically operated by HASP as if they were local readers, printers and punches, so that HASP can simultaneously, while operating all local unit record devices, read jobs from several remote readers into the queue of jobs awaiting processing and upon completion of the processing, can print and punch the results at the remote site.

Although a variety of devices may be utilized as remote terminals, this document discusses only the use of an IBM 1978 as a remote station.

Due to the use of blocking and variable length data to minimize line transmission time, the speed at which the 1978 devices are operated is dependent on the data being transmitted. Certain jobs, because of their data characteristics, will enable the 1978 to operate unit record devices at full rated speed. Other jobs, with less advantageous data characteristics, may cause the devices to operate at less than full speed.

2.0 OPERATING PROCEDURES

The following pages provide sufficient information for initiating and operating the IBM 1978 as a HASP Remote Terminal.

2.1 INITIATING PROCESSING

2.1.1 Transmission to the Central Computer

1. Establish the communication line (see Section 2.0.2).
2. Verify that the M/D and CHAR PHASE indicator lights are on and that the REC T/P light is off.
3. Set the operation mode switch to the SEND BINARY mode (see Note 1).
4. Set the mode switch to the BINARY Position.
5. Press START on the printer panel.
6. Load the cards to be transmitted into the feed hopper.
7. Press the FEED key on the Reader/Punch.
8. Press the START key on the Reader/Punch.
9. Cards will be transmitted until the reader hopper is emptied, at which time the audible alarm will sound and the feed check light on the Reader will come on. Additional cards may be transmitted at this point by beginning with step 6 again. If there are no more cards to be transmitted . . .

10. Press the STOP key on the Reader Punch and signal HASP in the central computer of the end of input by pressing the EOT key.

NOTE 1: Certain installations may, if all input card characters are of the 64 BCD character set, direct the setting of the operation mode switch to the SEND 1st CHAR position, in order to improve the transmission rate.

2.1.2 Reception from Central Computer

1. Establish the communications line (see Section 2.2).
2. Verify that the M/D and CHAR PHASE indicator lights are on and that the REC T/P light is off.
3. Set the operation mode switch to the REC 1st CHAR position.
4. Set the mode switch to the BINARY position.
5. Load blank cards into the feed hopper of the Reader/Punch.
6. Press the FEED key twice on the Reader/Punch (only if punching is to be done).
7. Press START on the printer panel.
8. Press START on the Reader/Punch (for punch).
9. Jobs will, as available, begin printing (and optionally punching).

2.1.3 Change of Operational Mode

In order to allow the 1978 operator to select the mode of operation HASP provides the following feature:

> At the end of the output for each job (punch or print if no punching is done), HASP will automatically pause for 28 seconds to allow the 1978 to be switched to the send mode. To initiate the sending of a job during this interval, the operator should follow the procedure outlined in part 2.1.1, beginning with step 3. If the 1978 is not switched to the send mode, the printing of the next job, if available, will automatically begin after the expiration of the 28 second period. Should no job be ready for printing, the 1978 will enter a dormant state to await either the receipt of print or an operator switch to the send mode. Switching modes while a transmission is occurring can only be done by instructing the control computer operator to RESTART the remote station.

2.2 ESTABLISHMENT OF COMMUNICATION LINE

1. Advise the operator of the central computer location that job transmission is to be initiated. (The central computer operator must authorize HASP to process jobs from a given remote terminal. This authorization must be given only once per terminal.)

2a. Leased (or private transmission lines) - place the data set in the DATA mode.

2b. Dial-up transmission lines - Press the TALK button on the data switch and dial, exactly as in a normal call, the number of the appropriate data set at the central computer. After the "ringing" sound, a shrill sound indicates that the phone has been answered. Immediately upon hearing this sound, press the DATA button on the data set and hang the telephone up.

3. The CHAR PHASE light on the control panel should appear to indicate synchronization of the terminal and the central computer.

NOTES on communication line establishment:

a. A "busy" signal indicates that the called data set is in use.

b. A line may be established by a call from the central computer to the remote site. To receive such a call, normal initialization procedures should be followed but rather than dialing, the AUTO button should be pressed and the phone hung up to await the call.

c. On a two-wire half-duplex telephone line, a period of several seconds may be registered to synchronize the two computers.

d. On the establishment of a connection other than the first, a maximum time of 28 seconds may be required to begin transmission of print and/or punch data.

e. If the data set at the central computer is not in the AUTO position, the operator may answer the call and, after talking may press the DATA key. An ensuing shrill sound indicates the computer connection has been established.

3.0 ERROR RECOVERY

This section indicates most possible error conditions, their probable causes and procedures necessary for correction.

3.1 SEND OPERATION ERROR STOPS

Figure 3.1.1 illustrates possible errors which may occur while transmitting jobs from the 1978 to the central computer.

Figure 3.1.1 Send Operation Error Stops

Error and Indication	Possible Causes	Correction Procedures
Validity Check 1. Alarm sounds. 2. SRP light is on, 3. Validity light is on. 4. Card check light may be on.	1. Invalid card code character. 2. Card skew.	1. Remove cards from stacker. 2. Remove cards from hopper. 3. Press feed key and stop key simultaneously twice to clear machine of cards. First card in stacker is the error card. 4. If necessary, correct error-card or set mode switch at proper position (must be BINARY). Other records may have been sent incorrectly if this switch was not set correctly. 5. (a) If the Cd/Pnt Bfr light is on, generate a machine reset by momentarily changing the operation switch to another position. NOTE: this procedure may result in a record check at the central computer which will be ignored by HASP. (b) If the Cd/Pnt Bfr light is off, press the check start key to reset the error condition. 6. Place the error card and the card following it in front of cards removed from hopper. 7. Follow normal start procedure

Figure 3.1.1 Send Operation Error Stops (Continued)

Error and Indication	Possible Causes	Correction Procedures
Card Check 1. Alarm sounds. 2. Card light is on.	1. Buffer input address error. 2. More or fewer than 80 columns read from card (without Transmit Variable Length Records special feature).	1. Remove cards from stacker. 2. Remove cards from hopper. 3. Press feed and stop keys simultaneously twice to clear machine of cards. 4. (a) If the Cd/Pnt Bfr light is on, place the last card in the stacker into the hopper. (b) If the Cd/Pnt Bfr light is off, place both of the cards in the stacker into the hopper. 5. Replace the cards previously removed from the hopper. 6. Press check start button to reset the error indication. 7. Follow normal start procedure

Figure 3.1.1 Send Operation Error Stops (Continued)

Error and Indication	Possible Causes	Correction Procedures
Input/ Output Check 1. Alarm sounds. 2. I/O check light is on. 3. CTR light is on.	1. Buffer output address error.	1. Note card count in input counter lights to determine number of cards stored in buffer 2. Remove cards from hopper. 3. Remove cards from stacker except the number indicated by the input counter. 4. Press feed and stop keys simultaneously twice to clear machine of cards. 5. Place cards left in stacker into the hopper. 6. Replace cards removed from hopper. 7. Generate a machine reset by momentarily changing to another position with the operation switch. 8. Follow normal start procedure. NOTE: This procedure may result in a record check at the central computer, which will be ignored by HASP.

Figure 3.1.1 Send Operation Error Stops (Continued)

Error and Indication	Possible Causes	Correction Procedures
CR Check 1. Alarm sounds. 2. CR light is on.	1. Bad character out of translator.	1. Remove cards from stacker. 2. Remove cards from hopper. 3. Press feed and stop keys simultaneously twice to clear cards from machine. 4. Place second card in stacker into the hopper. 5. Replace cards removed from hopper. 6. Press check start button. 7. Follow normal start procedure. 8. If the error occurs the second time, perform steps 2 and 3 above again. 9. Generate a machine reset by momentarily changing the operation switch to another position. NOTE: This procedure may result in a record check at the central computer, which will be ignored by HASP. 10.Place both cards in the stacker into the hopper. Replace the cards removed from the hopper. 11.Follow normal start procedure.

Figure 3.1.1 Send Operation Error Stops (Continued)

Error and Indication	Possible Causes	Correction Procedure
Feed Check 1. Alarm sounds. 2. SRP light on. 3. Feed check light on.	1. Hopper empty. 2. Card jam. 3. Failure to register card at one of stations in transport. 4. Misfeed at hopper area. 5. Misfeed at stacker area.	1. Determine cause of feed check. 2. If hopper empty, put in more cards and press feed key unless at end of operation, in which case press the stop key and the EOT key. 3. If a misfeed and no cards are in machine, reshuffle cards in hopper and check for a nicked card which should be replaced. Put cards back in hopper and press the feed key. 4. If a card jam exists, follow the misfeed correction procedure to remove it. 5. Repair any damaged cards if necessary. 6. Place into hopper all cards that have not as yet passed the read station. 7. Press the feed key.
Ctr 1. Alarm sounds. 2. Ctr light is on.	1. Input check.	1. Depress card read-punch key.

3.2 RECEIVE OPERATION ERROR STOPS

Figure 3.2.1 illustrates possible error stops which may occur while receiving print (and/or punch) from the central computer.

Figure 3.2.1 Receive Operation Error Stops

Error and Indication	Possible Causes	Correction Procedure
Input/ Output Check 1. Alarm sounds. 2. I/O light is on. 3. CTR light is on. 4. Output light is on.	The 1978 requests retransmission of an error record two times from the transmitting terminal. After the three tries the machine stops and the error indicator is turned on. 1. Overflow of data (exceeding the 7 sub-record limit). 2. Loss of an address. 3. Overflow of buffer (card check light is also on). Maximum of 329 characters. 4. RM/GM Check also turns on this light (see explanation of RM/GM to indicate wrong length record.	1. Press start key on card read-punch (printer on Model 3) for three more attempts at transmission. 2. If card check light is on, follow procedure for correction of a card check. 3. If the error persists, the System should be checked.

Figure 3.2.1 Receive Operation Error Stops (Continued)

Error and Indication	Possible Causes	Correction Procedure
Punch Check (Model 2 Only) 1. Alarm sounds. 2. SRP light is on. 3. Punch Check light is on.	1. Error in punching of a card. 2. Failure of card-feed latch at the of a feed cycle.	1. Flag last card in stacker. 2. Force feed one card by pressing stop and feed key simultaneously. Operation will continue until all sub-records remaining in MBS at time of error have been punched, and then it will stop. The card in the stacker immediately after the last card stacked before stop, is the error card. If error did not occur in the last column to be punched, the error cardhas been repunched and the corrected card follows the error card in the stacker. The error card can then be thrown away. If the error occurs in the last column to be punched the entire card will have been punched (if not, the columns past the error column will be left blank), and the last column of the card should be checked and corrected if necessary, by manual methods. 3. If the error was due to a clutch failure, the entire error card will be blank and should be discarded. In this case, the flagged card should be checked for scattered punching, and if so, discarded and manually corrected. 4. Press start to resume normal operation. NOTE: The central computer operator may be notified to BACKSPACE or RESTART the punch to retry.

Figure 3.2.1 Receive Operation Error Stops (Continued)

Error and Indication	Possible Causes	Correction Procedure
Card Check 1. Alarm sounds. 2. Card light is on. 3. Other lights may be on.	1. Address overflow of buffer (transmittal record too long). 2. Address read check (loss of an address). 3. An invalid character in the translator or from the line.	Punch Operation Only (Model 2 Only) 1. (a) If Cd/Pnt Bfr indicator is on, remove all the cards from the stacker except one fewer than the number indicated by the output counters. (b) If Cd/Pnt Bfr indicator is off, remove all the cards from the stacker except the number indicated by the output counters. 2. Force feed one card by pressing the stop key and the feed key simultaneously. Discard the cards in the stacker. 3. Press the check start key. 4. If Cd/Pnt Bfr indicator was on, the first card entering the stacker must also be discarded.
Character Check 1. Alarm sounds. 2. Character check light is on. 3. CTR light is on.	1. The mode switch is not in the BINARY position. 2. Caused by the transmission line. (directly or indirectly). Three tries have been made to obtain correct information before the stop and error condition occurs.	1. Verify the setting of the mode switch. 2. Press start on card read-punch (printer on Model 3) for three more attempts at transmission. 3. If error persists, the transmission line should be changed, or the mode of the two machines should be compared.

Figure 3.2.1 Receive Operation Error Stops (Continued)

Error and Indication	Possible Causes	Correction Procedure
Record Check 1. Alarm sounds. 2. Record light is on. 3. Output light is on.	1. Loss of a record. 2. Duplication of a record.	1. Flag the deck or printed report to indicate the approximate location for future reference if necessary. 2. Press the check start button. 3. The central computer operator may be notified to BACKSPACE or RESTART the job.
CR Check 1. Alarm sounds. 2. CR light is on.	1. Bad character out of translator.	1. Follow procedure set up for correction of a card check.
RM/GM 1. Alarm sounds. 2. RM/GM light is on. 3. I/O light is on.	1. Receiving 1978 has not received a RM/GM at proper position in sub transmittal record.	1. Follow procedure for Input/Output check.

Figure 3.2.1 Receive Operation Stops (Continued)

Error and Indication	Possible Causes	Correction Procedure
Output Check 1. Alarm sounds. 2. Output light is on. 3. Other check lights may be on.	1. I/O check. 2. Card check. 3. Record check.	1. Follow the procedure indicated by the other check lights that are on. 2. If no other lights are on, press the start key to resume operation.
(Chipbox Full Model 2 Only) 1. Alarm sounds. 2. SRP light on. 3. Chipbox light on.	1. Full chipbox.	1. Open door on lower rear panel. 2. Remove chipbox and empty. 3. Replace chipbox and close cover. 4. Press start on card read punch to resume operation.
SYNC Check 1. Sync check light is on. 2. Ready light is off. 3. Run light is off.	1. Typebar is not inserted correctly. 2. Printer ribbon is out of line.	1. Determine cause of error. 2. Correct according to 1443 procedure. 3. Press reset key. 4. Press start on printer (and card read-punch on Models 1 and 2) to resume operation.

Figure 3.2.1 Receive Operation Error Stops (Continued)

Error and Indication	Possible Causes	Correction Procedure
Parity Check 1. Parity check light is on. 2. Ready light is off. 3. Run light is off.	1. Invalid character has been sent to printer.	1. Press reset key. 2. Press check start button. 3. Press start on printer (and card read-punch on Models 1 and 2) to resume operation.
Form Check 1. Form check light on. 2. Ready light off. 3. Run light is off.	1. Forms in printer are out of line.	1. Realign forms in printer according to 1443 procedures. 2. Press the reset key. 3. Press start on printer (and card read-punch on Models 1 and 2) to resume operation. 4. The print may be BACKSPACED or RESTARTed at the central computer to recover lost time.

Figure 3.2.1 Receive Operation Error Stops (Continued)

Error and Indication	Possible Causes	Correction Procedures
End of Form 1. End of form light on. 2. Ready light off. 3. Run light is off.	1. Out of printer forms.	1. Insert new printer forms according to 1443 procedures. 2. Press start on printer (and card read-punch on Models 1 and 2) to resume operation.
Carriage Interlock 1. Carriage interlock light is on. 2. Ready light is off.	1. Carriage brushes are in a raised position. 2. 6 or 8 line/inch belt cover is raised.	1. Correct condition. 2. Press start on printer (and card read-punch on Models 1 and 2) to resume operation.

Figure 3.2.1 Receive Operation Error Stops (Continued)

Error and Indication	Possible Causes	Correction Procedures
Feed Check (Model 2 Only) 1. Alarm sounds. 2. SRP light is on. 3. Feed Check light is on.	1. Hopper empty. 2. Card Jam 3. Failure to register a card at one of the stations in the transport. 4. Misfeed at hopper area. 5. Misfeed at stacker area.	Determine cause of feed check. If the hopper is empty, follow normal start procedure. Otherwise: 1. Remove cards from stacker. 2. Remove cards from hopper. 3. If a card jam, follow misfeed correction procedure to remove cards. 4. If not a card jam, press feed and stop keys simultaneously twice to clear cards from machine. 5. Examine cards cleared from machine in steps 3 or 4 and discard any damaged cards or any that are incorrectly or incompletely punched. The damaged cards from a card jam will have to be duplicated manually, the other will be repunched. (NOTE: The Central Computer Operator may be notified to BACKSPACE or RESTART the job to recover damaged cards. 6. Replace cards removed from hopper. 7. Follow normal start procedure.

4.0 CENTRAL COMPUTER CONTROL

Certain of the control cards recognized by HASP can be introduced from the remote terminal site. Following is a list and meaning of these control cards.

1	12	71
/*MESSAGE	Any Message	

The data punched into columns 12-71 of this card will be displayed on the central computer operator's console at the time the job is being read into the system. This may be used to identify certain jobs, give special instructions, etc. The /*MESSAGE card may be placed anywhere within the input job stream. If this card appears within a job, the HASP number assigned to that job will be appended to the message before displaying it, otherwise the remote station ID will be appended.

1	10	16
/*ROUTE	PUNCH PRINT	LOCAL

This card, when included anywhere within a job being submitted to the central computer, will cause the print or punch output (as indicated in col. 10) to be processed on local unit-record equipment. This card may be used to divert large volumes of print or punch to local high speed devices to avoid terminal congestion. Both print and punch may be routed locally by including two /*ROUTE cards in a job.

```
1               16
/*PRIORITY      nn
```

This card may be used to force the assignment of priority "nn" to the job which immediately follows. "nn" may be any digit or digits from 0-15. This control card when read locally by HASP is interrupted as an absolute priority assignment to a job. However, when read from a remote station the card is regarded as a priority assignment to this job relative to other jobs from the same station. Thus a remote operator can, via the /*PRIORITY card order the sequence of jobs submitted from only his station, for example, a /*PRIORITY 15 (where 15 is the highest priority) would cause its job to be the next job from that remote station to be processed, although not necessarily the next job to be processed by the central computer. The relative position of the priority structure of a remote terminal with respect to the overall system priority structure is determined at HASPGEN by central computer personnel.

The /*PRIORITY card must immediately precede the OS/360 JOB card of the job to which it refers.

5.0 OPERATIONAL HINTS

1. It is suggested that the remote terminal operators become familiar with normal HASP operating procedures at the central computer site. The HASP OPERATOR'S GUIDE is contained as Section 11.1 in the HASP SYSTEMS MANUAL.

2. While HASP allows the 1978 operator to select the mode of operation (i.e. send or receive) at the interval between jobs, a particuliar mode, once begun, <u>must</u> normally be continued to the end-of-job. Operator controls available at the central computer may, however, be utilized to avoid this restriction.

3. During dormant periods, the 1978 should be left in the receive mode so that printing (and/or punching) may begin as it becomes available.

4. The communications line may be disconnected at any time, which will cause HASP to hold all jobs awaiting the terminal until the line is again established.

11.4 HASP REMOTE TERMINAL PROCESSOR (1130) OPERATOR'S GUIDE

The following section contains detailed instructions for operating an 1130, equipped with a Binary Synchronous Communication Adapter, as a HASP MULTI-LEAVING, remote workstation. Although intended for use as a separate operational manual, it has been included in the HASP SYSTEMS manual to achieve completeness.

(The remainder of this page intentionally left blank.)

H A S P

REMOTE TERMINAL PROCESSOR

FOR

MULTI-LEAVING BINARY SYNCHRONOUS

COMMUNICATIONS

1130 OPERATOR'S GUIDE

TABLE OF CONTENTS

1.0 INTRODUCTION

The HASP SYSTEM is an automatic spooling, priority scheduling system which, while operating in conjunction with OS/360, operates an unlimited number of peripheral devices simultaneously with normal job execution, to perform the functions normally associated with off line support computers. The function of HASP has been extended to operate, via several classes of telephone lines, peripheral devices located remotely from the central computer complex.

Through the use of the HASP Remote Job Entry feature, a user, located perhaps thousands of miles from a particular System/360 installation, can utilize the capabilities of that installation much as if the central system were located at the remote site. The unit record devices at a remote station are logically operated by HASP as if they were local readers, printers, punches, and consoles, so that HASP can simultaneously, while operating all local unit record devices, read jobs from several remote readers into the queue of jobs awaiting processing; and output to several remote printers and/or punches results of previously entered jobs which have completed execution.

Although a variety of devices may be utilized as remote terminals, this document discusses only the use of the 1130 with a binary synchronous communications adapter as a remote station.

A special program has been written for the 1130 which can be considered a logical extension of the HASP System. This program referred to in the HASP documentation as HASP/RTP1130 performs the following functions:

A. INPUT

1. Reads from the attached card reader(s).
2. Recognizes operator requests and reads from the attached console.
3. Identifies, compresses, and blocks card images and commands for transmission to HASP.
4. Queues blocked records for transmission to HASP.

B. OUTPUT

1. Dequeues blocked records received from HASP.
2. Identifies the device required for output of the records.
3. Deblocks and decompresses output records, queueing the images for printing, punching, or typing.
4. Prints, punches, and types the output records as required.
5. Sets status flags indicating backlog conditions on the output devices.

C. COMMUNICATIONS

1. Establishes and maintains synchronization with HASP.
2. Dequeues blocked input records and transmits them to HASP upon request from HASP.

3. Provides backlog status flags indicating the terminal's ability to receive the various output streams from HASP.

4. Receives output from HASP and queues the blocked records for processing.

HASP/RTP1130 may read, print and punch data concurrently depending upon the options selected by the installation and the capabilities of the unit record devices.

Due to the use of blocking and character compression to minimize line transmission time, the speed at which the 1130 unit record devices will operate is dependent on the data being transmitted, and the number of concurrent functions. Certain job mixes, because of their data characteristics, will enable HASP/RTP1130 to operate the unit record devices at near full speed. Other job mixes may cause the devices to operate in short bursts because of contention on the communication line.

2.0 OPERATING PROCEDURES

The following pages provide sufficient information for initiating and operating the HASP/RTP1130 program during the remote job stream processing session.

2.1 INITIATION OF A REMOTE JOB STREAM PROCESSING SESSION

The initiation of a remote job stream processing session involves the initial program loading of the HASP/RTP1130 program deck, the establishment of the communication lines, and the exchange of initial control information between HASP and the HASP/RTP1130 program. The initial control sequence ends with the passing of the SIGN-ON remote identification information.

2.1.1 Initial Program Load (IPL)

1. Ready the RPT1130 deck in the primary card reader (do not place Jobs behind RTP1130 deck). If two card readers exist, be sure the second is not ready.

2. Ready all printers.

3. Set the STR/BSC switch to BSC.

4. Set the linespeed control to the appropriate value...1200, 2000, 2400, etc.

5. Verify that the rotary CPU control switch is set to the "RUN" position.

6. Press "IMM STOP", "RESET" and "PROGRAM LOAD" on the 1130 console.

7. After the last card has been read, the card reader will go out of ready. Ready the card reader (press start on the reader until it goes ready) and press "START" on the 1130 console. The last card should be the end card of the RTP1130 deck or a /*SIGNON card or a REP card. All unidentified cards are ignored.

8. Establish the communications line.

9. Processing should then begin in the full MULTI-LEAVING mode.

2.1.2 Establishment of Communications Line

The procedures for establishing communications with HASP are as follows:

1. Ready the data set. This will involve different actions based upon the type of data set; for non-switched lines when the BSC RDY indicator is on no action is required. Certain non-switched

lines will require the data set DATA button be pressed. To ready a dial line data set, perform the following:

A. Press the TALK button and lift the receiver on the data set.

B. Dial the assigned number for remote terminal.

C. If the HASP line is available, the control system will answer with a high pitched tone. Press DATA and hang up immediately (the data set is ready).

D. If the HASP line is in use, a busy signal will be received. Hang up and try again later or dial an alternate communications line number.

E. If the call is not answered, the central HASP operator has not given the necessary command to authorize use of that communication line.

2. When the data set is made ready, the BSC RDY indicator will be on in addition to the REC light indicating the terminal program is waiting for HASP to request a transmission. When requested, RTP1130 will begin the initial control sequence. The REC and TSM lights will alternate during normal operation.

3. When the initial sequence is complete, control information is transmitted to HASP and "handshaking" with REC and TSM alternating will continue. In addition, the message

"COMMUNICATION LINE ESTABLISHED"

is printed on the console typewriter.

NOTE: The message "DATA SET NOT READY" is printed after the execution of Step 7 in the IPL Procedure if that condition exists.

2.2 REMOTE JOB STREAM PROCESSING

During remote job stream processing the operator is concerned with operating the unit record devices, while submitting jobs and controlling the output via commands to the central system.

2.2.1 Output Processing

Except as controlled by the remote terminal operator or central system operator via commands to HASP, the printing and punching of job output is handled automatically by the HASP/RTP1130 system.

2.2.2 Input Processing

Job submission can be initiated at any time depending upon the capabilities of the card reader - punch combination attached to the 1130.

The 2501 reader allows the cards to be placed in the hopper as desired. The reader will stop after reading the last card in the hopper and the message "INTERVENTION REQUIRED ON 2501" will be printed on the console printer. The operator may press START on the reader to terminate the job stream or load more cards in the hopper, press START and continue the job stream. The intervention message described is typed any time the 2501 goes from a "ready" condition to a "not ready" condition.

The input reader to HASP/RTP1130 is considered always "HOT", that is, it is continually testing the reader and attempting to read cards.

2.2.3 Input Processing On The 1442 Reader/Punch

Operator action through the keyboard/console is required to define the function desired for the 1442 Reader/Punch. Initially, the 1442 R/P is considered to be a card reader. When punch data is transmitted to the 1130, a message is printed:

"PUNCH PROCESSOR WAITING FOR 1442"

The operator may then define the 1442 as a punch by entering the command:

.DPUNCH or .DP

which specifies the definition of the 1442 as a punch.

The above specification is necessary for each job which causes punch data to be transmitted to the terminal.

Once defined as a reader by issuing the command:

.DREADER or .DR

The 1442 remains so assigned until a .DPUNCH is given and operates in the same manner as described for the 2501 card reader.

NOTES:

1. The .DPUNCH and .DREADER Commands will result in no action if the opposite function is active at the time issued.

2. Defining the 1442 R/P as a reader with blank cards intended for punching ready in the hopper will result in a "SKIPPING for JOB CARD" message from HASP as the blank cards are read and transmitted.

3. Defining the 1442 R/P as a punch with input cards in the hopper and punch data available from HASP will result in clobbered input cards.

2.2.4 Output Processing On The 1442 Punch

A system with the 1442 defined as a punch only device requires no operator action other than blank cards and a "ready" condition.

2.3 TERMINATING A SESSION

When the remote terminal operator desires to terminate remote processing, he should send through the card reader input stream a /*SIGNOFF card. This tells HASP not to initiate the sending of any more job output and release the communication line (if DIAL) when the current print and punch streams are finished. The RDY light on the data set will go out and an SCA LOG Message Code 3 will be issued periodically. For nonswitched lines HASP will make the line available and thus send initial sequence requests to the HASP/RTP1130 program. The operator should check to see if printing and punching of output streams have successfully terminated and press STOP on the CPU. To start a new session the operator must perform the steps prescribed for the initialization of a remote job stream processing session.

2.4 COMMAND PROCESSING

Central system commands as well as local commands may be entered into the operator's console. Any message entered into the keyboard which is not recognized as a local command will be transmitted to HASP for action. Although all commands transmitted to HASP may be listed on the central system operator consoles only those designated in the HASP operator's guide as being available to the remote user will be acted upon.

2.4.1 Entering Commands

The operator should perform the following steps when entering commands:

1. Press the INT REQ button which is located to the right of the console typewriter keyboard.
2. When the K.B. Select indicator comes on, type in the command and press EOF.
3. If a typing error is noticed prior to pressing EOF, press ERASE FIELD KEY and repeat step 1.

NOTE: The "backspace key" is processed in the same manner as the erase field key.

2.4.2 Local Commands

COMMAND	MEANING/COMMENTS
.DR	Define the dual 1442 Reader/Punch as a reader. This definition remains in effect until a ".DP" command is entered and accepted.
.DP	Define the dual Reader/Punch as a Punch. This definition remains in effect for one job only. The function to be next assigned is dependent on the entering of another .DP or a .DR.

Commands must start in the first available type position and are identified by a "." period. No blanks are allowed in the body of a command. Acceptance of a console command is signalled by the message.

"OK!"

Rejection by the message:

"WHAT?"

3.0 ERROR PROCEDURES

The following sections indicate some of the more common error conditions which may arise and the necessary steps for recovery from the error.

3.1 COMMUNICATION ADAPTER ERRORS

The design of the synchronization technique for HASP remote terminals is such that no errors are expected during a processing session. The occurrence, therefore, of any error condition is an unusual condition resulting from either system or communication facility malfunction or operational conditions. In general, the displaying of error messages is informational only since the terminal processor will automatically initiate the appropriate recovery action. A statistical summary of all errors is maintained in the HASP Environmental Recording Table and a historical report is produced each time HASP/RMT360 is loaded (unless storage has been cleared). Additionally, the occurrence of any error will cause a descriptive message to be displayed immediately on the console typewriter. Table 3.1.1 indicates each of the possible communication errors which can occur, their meaning and the recovery action taken.

06DDDD00	Data overrun error. program unable to read data before next character received from transmission line.	HASP will be requested to retransmit the record.
07DDDD00	Data set not ready. Discovered at interrupt time.	RTP1130 waits for data set to become ready and then resumes operation on the line.
08DDDD00	Error on initial read. first character not SOH, DLE, ENQ or NAK...or... SOH-STX, DLE-STX, DLE-ACKO pair not found.	HASP will be requested to retransmit the record.
09DDDD00	NAK received	Last data record will be retransmitted to HASP.
0BDDDD00	Single DLE found in transparent data.	HASP will be requested to retransmit the record.
0CDDDD00	ENQ received after INITIAL SIGN-ON Sequence.	HASP will be requested to retransmit the record.
0DDDDD00	NO PAD character following NAK	HASP will be requested to retransmit the record.

DDDD = Last SCA Device Status Word received.

3.2 REMOTE TERMINAL RESTART

In the event of an untimely interruption of the remote terminal operation such as a machine, program, communications, or environmental failure, the remote terminal operator should notify appropriate maintenance personnel of the malfunction, save material which may be of use determining the source of the failure, and with the aid of the central system operator prepare for restarting the terminal as follows:

1. Notify the central system operator of the failure and, if necessary request his assistance in preparing for restart.
2. Determine the current job being transmitted to HASP. (The central system operator has a record of the current job being submitted to HASP). The job stream starting with the current job must be submitted to HASP after restart.
3. Determine the loss of data on the output devices and inform the central operator to BACKSPACE or RESTART the printer or punch as necessary. (The central system's line should be made available for a subsequent session with the remote station or other stations within the system).
4. When the remote terminal is available, perform the steps required for initiating a "Remote Job Processing Session".

3.3 LOAD PROCESS UNUSUAL CONDITIONS

The first eight cards of the 1130 remote terminal deck comprise a "bootstrap" loader (RTPBOOT) which is used to load the main loader (RTPLOAD) into upper 1130 storage. RTPLOAD then loads the main terminal deck (RTP1130), processes REP cards (if any) and the /*SIGNON card (if included).

The following tables describe the unusual conditions which may occur in conjunction with RTPBOOT and RTPLOAD.

RTPLOAD		
CONDITION INDICATION	CONDITION DESCRIPTION	OPERATOR ACTION
System wait at location '0010'. AC displays value 'FFF3'.	The last REP card read contained a format error.	Loading is terminated permanently. <u>Note:</u> card in error and notify system programmer or lead operator.
System wait at location '0010'. AC displays value 'FFF2'.	RTPLOAD computed sum (checksum) of columns 1-72 of last RTP1130 card read does not match value in columns 73-74 previously computed.	Loading may be resumed by pressing start on 1130 console. <u>UNPREDICTABLE RESULTS MAY OCCUR</u>. Best action is to note card in error and notify system programmer or lead operator.
System wait at location '0010'. AC displays value 'FFF1'.	This is not an error. The last card has been read by the 2501 or 1442 and operation action is required.	To commence RTP1130 processing, press start on card reader until ready then press start on 1130 console.

RTPBOOT		
CONDITION INDICATION	CONDITION DESCRIPTION	OPERATOR ACTION
System loop at location 'AA' with IAR displayed at location 'AB'.	RTPBOOT computed sum (checksum) of columns 1-72 of last card read does not match value in columns 73-74 previously computed during RTPLOAD generation.	Loading of RTPLOAD is permanently terminated. Note card being processed and contact system programmer or lead operator about problem.
System loop at location 'AE' with IAR displaying location 'AF'. AC contains card code value of column in error. XR2 contains 2's complement of card column number in error.	RTPBOOT detected illegal EBCDIC punch in RTPLOAD card just read <u>or</u> the last 4 cards of RTPBOOT contain an illegal EBCDIC punch.	Loading of RTPLOAD is permanently terminated. Note card being processed and contact system programmer or lead operator about problem.

4.0 SYSTEM CONTROL CARDS

Certain of the control cards recognized by HASP can be introduced from the remote terminal site. Following is a list, with meanings, of these control cards. Column numbers appear over the beginning character of each section of the card.

1	16	71
/*MESSAGE	Any Message	

The data punched into columns 16-71 of this card will be displayed on the central computer operator console at the time the job is being read into the system. This may be used to identify certain jobs, give special instructions, etc. The /*MESSAGE card may be placed anywhere within the input job stream. If this card appears within a job, the HASP number assigned to that job will be appended to the message before displaying it, otherwise the remote station ID will be appended.

1	10	16
/*ROUTE	PRINT	LOCAL
	PUNCH	REMOTEn
		PRINTERn
		PUNCHn

This card, when included anywhere within a job being submitted to the central computer, will cause the print or punch output (as indicated in column 10) to be processed on another remote workstation (REMOTEn), a

specific local printer and/or punch, or the first available local printer and/or punch (LOCAL). This card may be used to divert large volumes of print or punch to local high speed devices to avoid terminal congestion. Both print and punch may be routed locally by including two /*ROUTE cards in a job.

```
1               16
/*PRIORITY      n
```

This card may be used to force the assignment of priority "n" to the job which immediately follows. "n" may be any numerical value from 0-15. This control card when read locally by HASP is interpreted as an absolute priority assignment to a job. However, when read from a remote station the card is regarded as a priority assignment to this job relative to other jobs from the same station. Thus, a remote operator can, via the /*PRIORITY card order the sequence of jobs submitted from only his station, for example, a /*PRIORITY 15 (where is the highest priority) would cause its job to be the next job from that remote station to be processed, although not necessarily the next job to be processed by the central computer. The relative position of the priority structure of a remote terminal with respect to the overall system priority structure is determined at HASPGEN by central computer personnel.

The /*PRIORITY card must immediately precede the OS/360 JOB card for the job to which it refers.

```
1               16             25
/*SIGNON        REMOTEn        Password
```

This card appears at the end of the HASP/RTP1130 program deck and is used to override the remote identification number normally assigned to the HASP/RTP1130 program deck. For DIAL lines the /*SIGNON card may be used to submit a password which, if correct, will allow the remote terminal access to the HASP system for remote job stream processing. The value "n" must match the remote identification number assigned to the remote station by central computer personnel. The value of the "password" must match the password assigned to the line by the central computer operator when the communication lines is "started".

```
1
/*SIGNOFF
```

This card is used to inform the central system that the remote terminal operator desires to terminate a remote job stream processing session. When submitted to the central system, HASP will, at the completion of the current print and/or punch streams, disconnect the terminal from the system and prepare the line for other remote stations to SIGN-ON.

```
1
/*command
```

Selected HASP commands may be submitted to the central system through the remote terminal card reader. Commands submitted in this manner must

be the first cards of a job stream (in front of the first job submitted). Commands which can be submitted are listed in the HASP operator's guide and must start in column 3 of the card, i.e. the first 3 columns will be "/*$". (See Section 2.4.1 for entering HASP commands via the console typewriter).

```
1              16                                          71
/*SETUP        volume-ser1,volume-ser2,...,volume-sern
```

The volume serials punched in columns 16-71 of the card will be displayed on the central system console and the associated job will be placed in HOLD status (not be scheduled for execution) until released by the central operator. The /*SETUP card appears in the corresponding job input deck between the OS/360 JOB card and the first EXEC card.

5.0 MESSAGE SUMMARY

Messages which are printed on the console typewriter originate at the central HASP SYSTEM or are generated by RTP1130 in conjunction with the terminal operation. Messages from HASP may be identified by the $ character prefix and the fact that they are printed in red if the red/black typewritter ribbon is installed.

Local messages (typed in black) are listed below along with a more detailed explanation of each message.

MESSAGE	EXPLANATION/ACTION
INTERVENTION REQUIRED ON xxxx	Where xxxx=1442, 2501, 1403 or 1132. Message indicates that the indicated device has gone from a "ready" to "not ready" condition usually due to the device being manually stopped or because the device requires operator action, e.g., cards or paper. The device should be serviced as required and made ready to continue operation.
PUNCH PROCESSOR WAITING FOR 1442	Issued whenever punch data is received for a system equipped with a combination 1442 read/punch. If the 1442 is defined as a reader, it must complete the read function before it may be defined as a punch. If the 1442 is defined as a punch, no further action (other than providing blank cards and making the device ready) is necessary (see Section 2.2.3).

MESSAGE	EXPLANATION/ACTION
DATA SET NOT READY	Issued when the communications adapter signals the workstation program that the attached telephone data sets is in a "not ready" condition. The program will not attempt to use the Communication Adapter until a "ready" condition is detected. All other functions (card input, typewriter, etc.) will continue up to the point of requiring the service of the adapter. If the data set was made not ready by manual intervention, operation may be resumed by making it ready. Caution: The central HASP SYSTEM may print error messages which could cause the operator to restart the communications line. In this event, the workstation program must be re-loaded according to Section 2.1.1.
SCA LOG xxxxxx00	Indicates an unusual condition associated with the SCA (Synchronous Communications Adapter) as described in Section 3.1.
COMMUNICATION LINE ESTABLISHED	Issued at the time the workstation program is initialized and when communications have been established with the central HASP SYSTEM.
WHAT?	Response to any local command not recognized by the workstation program.
OK!	Response to any local command recognized by the workstation program.

(The remainder of this page intentionally left blank.)

11.5 HASP REMOTE TERMINAL PROCESSOR (System/360) OPERATOR'S GUIDE

The following section contains detailed instructions for operating any model of System/360 equipped with binary synchronous communication facilities, as a HASP MULTI-LEAVING, remote workstation. Although intended for use as a separate operational manual, it has been included in the HASP SYSTEMS manual to achieve completeness.

(The remainder of this page intentionally left blank.)

H A S P

REMOTE TERMINAL PROCESSOR

FOR

MULTI-LEAVING BINARY SYNCHRONOUS

COMMUNICATIONS

SYSTEM 360 OPERATOR'S GUIDE

TABLE OF CONTENTS

1.0 INTRODUCTION

The HASP SYSTEM is an automatic spooling, priority scheduling system which, while operating in conjunction with OS/360, operates an unlimited number of peripheral devices simultaneously with normal job execution, to perform the functions normally associated with off line support computers. The function of HASP has been extended to operate, via several classes of telephone lines, peripheral devices located remotely from the central computer complex.

Through the use of the HASP Remote Job Entry feature, a user, located perhaps thousands of miles from a particular System/360 installation, can utilize the capabilities of that installation much as if the central system were located at the remote site. The unit record devices at a remote station are logically operated by HASP as if they were local readers, printers, punches, and consoles, so that HASP can simultaneously, while operating all local unit record devices, read jobs from several remote readers into the queue of jobs awaiting processing; and output to several remote printers and/or punches results of previously entered jobs which have completed execution.

Although a variety of devices may be utilized as remote terminals, this document discusses only the use of a System/360 Model 25 and larger with a binary synchronous communication adapter as a remote station.

A special program has been written for the remote System/360 which can be considered a logical extension of the HASP System. This program referred

to in the HASP documentation as (HASP/RMT360) performs the following functions:

A. INPUT

1. Reads from the attached card readers.
2. Recognizes operator requests and reads from the attached console.
3. Identifies, compresses, and blocks card images and commands for transmission to HASP.
4. Queues blocked records from transmission to HASP.

B. OUTPUT

1. Dequeues blocked records received from HASP.
2. Identifies the device required for output of the records.
3. Deblocks and decompresses output records, queueing the images for printing, punching, or typing.
4. Prints, punches, and types the output records as required.
5. Sets status flags indicating backlog conditions on the output devices.

C. COMMUNICATIONS

1. Establishes and maintains synchronization with HASP.
2. Dequeues blocked input records and transmits them to HASP upon request from HASP.

1.0 INTRODUCTION

The HASP SYSTEM is an automatic spooling, priority scheduling system which, while operating in conjunction with OS/360, operates an unlimited number of peripheral devices simultaneously with normal job execution, to perform the functions normally associated with off line support computers. The function of HASP has been extended to operate, via several classes of telephone lines, peripheral devices located remotely from the central computer complex.

Through the use of the HASP Remote Job Entry feature, a user, located perhaps thousands of miles from a particular System/360 installation, can utilize the capabilities of that installation much as if the central system were located at the remote site. The unit record devices at a remote station are logically operated by HASP as if they were local readers, printers, punches, and consoles, so that HASP can simultaneously, while operating all local unit record devices, read jobs from several remote readers into the queue of jobs awaiting processing; and output to several remote printers and/or punches results of previously entered jobs which have completed execution.

Although a variety of devices may be utilized as remote terminals, this document discusses only the use of a System/360 Model 25 and larger with a binary synchronous communication adapter as a remote station.

A special program has been written for the remote System/360 which can be considered a logical extension of the HASP System. This program referred

to in the HASP documentation as (HASP/RMT360) performs the following functions:

A. INPUT

1. Reads from the attached card readers.
2. Recognizes operator requests and reads from the attached console.
3. Identifies, compresses, and blocks card images and commands for transmission to HASP.
4. Queues blocked records from transmission to HASP.

B. OUTPUT

1. Dequeues blocked records received from HASP.
2. Identifies the device required for output of the records.
3. Deblocks and decompresses output records, queueing the images for printing, punching, or typing.
4. Prints, punches, and types the output records as required.
5. Sets status flags indicating backlog conditions on the output devices.

C. COMMUNICATIONS

1. Establishes and maintains synchronization with HASP.
2. Dequeues blocked input records and transmits them to HASP upon request from HASP.

1.0 INTRODUCTION

The HASP SYSTEM is an automatic spooling, priority scheduling system which, while operating in conjunction with OS/360, operates an unlimited number of peripheral devices simultaneously with normal job execution, to perform the functions normally associated with off line support computers. The function of HASP has been extended to operate, via several classes of telephone lines, peripheral devices located remotely from the central computer complex.

Through the use of the HASP Remote Job Entry feature, a user, located perhaps thousands of miles from a particular System/360 installation, can utilize the capabilities of that installation much as if the central system were located at the remote site. The unit record devices at a remote station are logically operated by HASP as if they were local readers, printers, punches, and consoles, so that HASP can simultaneously, while operating all local unit record devices, read jobs from several remote readers into the queue of jobs awaiting processing; and output to several remote printers and/or punches results of previously entered jobs which have completed execution.

Although a variety of devices may be utilized as remote terminals, this document discusses only the use of a System/360 Model 25 and larger with a binary synchronous communication adapter as a remote station.

A special program has been written for the remote System/360 which can be considered a logical extension of the HASP System. This program referred

to in the HASP documentation as (HASP/RMT360) performs the following functions:

A. INPUT

1. Reads from the attached card readers.
2. Recognizes operator requests and reads from the attached console.
3. Identifies, compresses, and blocks card images and commands for transmission to HASP.
4. Queues blocked records from transmission to HASP.

B. OUTPUT

1. Dequeues blocked records received from HASP.
2. Identifies the device required for output of the records.
3. Deblocks and decompresses output records, queueing the images for printing, punching, or typing.
4. Prints, punches, and types the output records as required.
5. Sets status flags indicating backlog conditions on the output devices.

C. COMMUNICATIONS

1. Establishes and maintains synchronization with HASP.
2. Dequeues blocked input records and transmits them to HASP upon request from HASP.

3. Provides backlog status flags indicating the terminal's ability to receive the various output streams from HASP.

4. Receives output from HASP and queues the blocked records for processing.

HASP/RMT360 may read print and punch data concurrently depending upon the options selected by the installation and the capabilities of the unit record devices.

Due to the use of blocking and character compression to minimize line transmission time, the speed at which the remote terminal unit record devices will operate is dependent on the data being transmitted, and the number of concurrent functions. Certain job mixes, because of their data characteristics, will enable HASP/RMT360 to operate the unit record devices at full rated speed. Other job mixes may cause the devices to operate in short bursts because of contention on the communication line.

2.0 OPERATING PROCEDURES

The following pages provide sufficient information for initiating and operating the HASP/RMT360 program during the remote job stream processing session.

2.1 INITIATION OF A REMOTE JOB STREAM PROCESSING SESSION

The initiation of a remote job stream processing session involves the initial program loading of the HASP/RMT360 program deck, the establishment of the communication lines, and the exchange of initial control information between HASP and the HASP/RMT360 program. The initial control sequence ends with the passing of the SIGN-ON remote identification information.

2.1.1 Initial Program Load (IPL)

The following steps should be taken to IPL the HASP/RMT360.

1. If power is off press POWER ON.
2. Ready the HASP/RMT360 deck in READER 1 designated by central system personnel) and press START and EOF on the reader. (The last card of the deck should be a blank or /*SIGNON card as directed by the installation).
3. Ready printers, punches, and the console.
4. Set the LOAD UNIT rotary switches to the device address of READER 1.
5. Disable the interval timer if present.
6. Set the MODE (RATE) and DIAGNOSTIC (FLT) switches to PROCESS.
7. Set CHECK CONTROL to STOP.
8. Press SYSTEM reset and LOAD.
9. All cards of the HASP/RMT360 deck should be read into the reader.

10. HASP/RMT360 will print the /*SIGNON card, if present, followed by a HASP ENVIRONMENTAL RECORDING ERROR PRINTOUT (if the contents of core remain unchanged since the last running of the program).

11. The remote terminal is now ready to communicate with HASP. HASP/RMT360 will wait while communications are established with HASP.

2.1.2 Establishment Of Communication Line

The procedures for establishing communications with HASP are as follows:

1. Ready the data set. This will involve different actions based on the type of data set. Readying nonswitched lines will only require the data set DATA button be pressed (if present). To ready a dial line data set perform the following:
   a. Press the TALK button and lift the receiver on the data set.
   b. Dial the assigned number for the remote terminal.
   c. If the HASP line is available, the central system will answer with a high pitched tone. Press DATA and hang up immediately (the data set is ready).
   d. If the HASP line is in use, a busy signal will be received. Hang up and try again later or dial an alternate communication line number.

e. If the call is not answered, the central HASP operator has not given the necessary command to authorize use of that communication line.

2. When the data set is made ready for HASP, HASP/RMT360 will wait to request a transmission. When requested HASP/RMT360 will begin the initial control sequence.

3. When the initial sequence is complete the SIGN-ON is transmitted to HASP. HASP/RMT360 will "handshake" with HASP until processing of job streams actually began.

2.2 REMOTE JOB STREAM PROCESSING

During remote job stream processing the operator is concerned with operating the unit record devices, while submitting jobs and controlling the output via commands to the central system.

2.2.1 Output Processing

Except as controlled by the remote terminal operator or central system operator via commands to HASP, the printing and punching of job output is handled automatically by the HASP - HASP/RMT360 system.

2.2.2 Input Processing

With the exception of 2520 and 1442 DUAL Reader-Punch devices, job submission can be initiated at any time from any card reader supported by the HASP/RMT360 program (all readers may be running concurrently). The operator need only place the cards in the input hopper as desired and press reader START. When the last of a job stream has been loaded into the HOPPER, press reader EOF to allow the reading of the last cards to signal the program that the end of stream has been read.

The input readers to HASP/RMT360 are considered always "HOT"; that is the program is continually testing each reader and attempting to read cards. When any card reader is loaded with cards HASP/RMT360 will read and transmit them to HASP.

2.2.3 Input Processing On Dual Reader Punch

Devices with single card paths for read and punch functions are considered DUAL reader/punches if they are supported for both functions. The following are supported DUAL devices:

1. 1442 READER PUNCH
2. 2520 READER PUNCH

Operating The Dual Reader Punch Devices

Dual devices have four basic status conditions which affect the operator:

1. Neutral - Reader empty from normal program execution.
2. Input - Reading normal job stream.
3. Output - Punching normal output from HASP.
4. Output error receovery - attempting to receover from punch errors.

At IPL time the DUAL device will be in neutral status and may be treated as any reader device in that the operator is at liberty to submit multiple job streams at any time. Any blank cards mixed in the input stream will be submitted to HASP as job input. When HASP/RMT360 recognizes the end of file (EOF) the DUAL device will revert to the neutral status.

When the DUAL device is in neutral the operator may choose to ready the device with blank cards which places it in the output status. If HASP has output waiting, HASP/RMT360 will respond immediately by punching into the blank cards. However, after all punching is finished or if there is a pause due to low line speeds the operator may not run the remaining cards out of the device and ready it with job stream cards. The procedure for interrupting the output mode is as follows:

1. Press STOP on the device.
2. Remove the cards from the HOPPER. (DO NOT non-process run the cards out of the card path).
3. Place the job stream cards in the HOPPER and press reader START.
4. If the punch happens to be busy the device will continue punching until the job stream is encountered; then the device will enter the input status. (Not all blank cards need be removed from the hopper).
5. If the punch is momentarily idle, the operator can cause the device to pass through one card by pressing reader STOP and then START. If several blank cards are in the hopper in front of the job stream the operation must be repeated for each blank card.

The DUAL device is in error recovery status when a punch error occurs. HASP/RMT360 will attempt to repunch the record in error into the following card. If HASP/RMT 360 encounters a nonblank card a read error will occur (see unit record error procedures). The operator should non-process run out the job stream cards, place one or more blank cards in front and ready the device.

2.2.4 Command Processing

Any message entered into the 1052 operator's console via the keyboard will be transmitted to HASP for action. Although all commands transmitted to HASP may be listed on the central system operator consoles, only those desginated in the HASP operator's guide as being available to the remote user will be acted upon.

Entering Commands

The operator should perform the following steps when entering commands:

1. Press the REQUEST button on the right side of the keyboard. The ATTN indicator (indicator above the keyboard on 1052) will glow momentarily.
2. When the PROCD indicator comes on, type in the command and press EOB (numerical 5 key pressed while ALTN CODING key is in).
3. If a typing error is noticed prior to pressing EOB, press CANCEL (numerical 0 key pressed while ALTN CODING key is in) and repeat step 2.
4. If after receiving a proceed indicator no command is desired press EOB.

2.3 TERMINATING A SESSION

When the remote terminal operator desires to terminate remote processing, he should send through any card reader input stream a /*SIGNOFF card. (See section 4.0). This causes HASP not to initiate the sending of any more job output and to release the communication line (if DIAL) when the current print and punch streams are finished. The DATA light on the data set will go out and BSCA will enter a CHECK CONDITION. For nonswitched lines HASP will make the line available and thus send initial sequence requests to the HASP/RMT360 program. HASP/RMT360 will log on the console message device, UNIT CHECK in case of a DIAL line or INVALID RESPONSE in case of a nonswitched line. The operator should check to see if printing and punching of output streams have successfully terminated and press STOP on the CPU. To start a new session the operator must perform the steps prescribed for the initialization of a remote job stream processing session.

3.0 ERROR PROCEDURES

The following sections indicate some of the more common error conditions which may arise and the necessary steps for recovery from the error.

3.1 COMMUNICATION ADAPTER ERRORS

The design of the synchronization technique for HASP remote terminals is such that no errors are expected during a processing session. The occurrence, therefore, of any error condition is an unusual condition resulting from either system or communication facility malfunction or operational conditions. In general, the displaying of error messages is informational only since the terminal processor will automatically initiate the appropriate recovery action. A statistical summary of all errors is maintained in the HASP Environmental Recording Table and a historical report is produced each time HASP/RMT360 is loaded (unless storage has been cleared). Additionally, the occurrence of any error will cause a descriptive message to be displayed immediately on the console typewriter. Table 3.1.1 indicates each of the possible communication errors which can occur, their meaning and the recovery action taken.

Table 3.1.1 Communication Adapter Error Messages

MESSAGE	MEANING	ACTION TAKEN
01RREE00	Block sequence check- a transmission block was duplicated or lost RR = Received block number EE = Expected block number	If duplicate the received block will be ignored. If lost block, HASP will be signaled to restart the job.
02000000	Negative reply received - a transmission block was not correctly received by HASP	The bad record will be re-transmitted.
03RRRR00	Unknown response received - an unrecognizable control character was received from HASP RRRR = First two characters received (If RRRR is correct sequence, ending sequence was bad)	HASP will be requested to retransmit the record.
04000000	Unit Exception - This indicates the receipt of an "EOT" character from HASP (EOT is not utilized in MULTI-LEAVING)	HASP will be requested to retransmit the record.
05SS0000	Unit check - a check condition has occurred in the communication adapter SS = Sense byte indicating type of check	The failing operation will be retried.

Table 3.1.1 Communication Adapter Error Messages (Continued)

MESSAGE	MEANING	ACTION TAKEN
	Common Examples -	
	SS=01=overrun on write	Write retried
	SS=02=parity check on write	Write retried
	SS=81=overrun on READ	Retransmission requested
	SS=88=lost data on read	Retransmission requested
	SS=90=time out (no response received from HASP in 3 sec.)	Retransmission requested
	SS=A0=transmission error	Retransmission requested
	SS=C0=EOT received	Retransmission requested
06CC0000	Unusual end - an unusual condition has occurred in the channel or control unit interface cc=CSW byte 5	The failing operation will be retried.
07000000	SIO failure - a start I/0 instruction was rejected by the Synchronous Data Adapter	The start I/0 will be retried.

3.2 UNIT RECORD ERROR PROCEDURES

Many of the unit record device errors which may occur during processing are of such nature that HASP/RMT360 is able to continue processing without operator intervention. Errors such as, DATA check on the reader and single pocket punch devices; FEED check; END OF FORM; etc. require operator assistance before use of the device can be continued. In any event all errors occurring on unit record devices will be logged in the HASP ENVIRONMENTAL RECORDING ERROR PRINTOUT table and immediately on the 1052 operator's console.

When the error message is printed the operator should perform the following:

1. Determine which device is in error (see table 3.2.1).
2. Note the device status (If HASP/RMT360 continues to use the device, the error message is informative in nature).
3. Correct the error in accordance with procedures prescribed for the device.
4. Ready the device for resumption of operations.

Table 3.2.1 HASP/RMT360 Unit Record Error Messages

MESSAGE	DESCRIPTION	PROGRAM ACTION
05SS0AAA	Unit deck - device within address "AAA" has unit check error described by sense byte "SS" and/or indicators lights on the device console.	
	Example sense byte settings	
	40 = Intervention required	Wait for operator
	10 = Equipment check	Treat as data check
	08 = Data check - card read, card punched, or line printed incorrectly.	Depending upon device ignore, retry, or wait for operator
	01 = Carriage control tape Channel 9 encountered on printer	Ignore
06CC0AAA	Unusual End - Previous I/O came to an unusual end. IBM customer engineer should be consulted CC = CSW byte 5 AAA = device address	Treat as data check.

3.3 REMOTE TERMINAL RESTART

In the event of an untimely interruption of the remote terminal operation such as a machine, program communications, or environmental failure, the remote terminal operator should notify appropriate maintenance personnel of the malfunction, save material which may be of use in determining the source of the failure, and with the aid of the central system operator prepare for restarting the terminal as follows:

1. Notify the central system operator of the failure and, if necessary request his assistance in preparing for restart.
2. Determine the current job being transmitted to HASP. (The central system operator has a record of the current job being submitted to HASP). The job stream starting with the current job must be submitted to HASP after restart.
3. Determine the loss of data on the output devices and inform the central operator to BACKSPACE or RESTART the printer or punch as necessary. (The central system's line should be made available for a subsequent session with the remote station or other stations within the system).
4. When the remote terminal is available, perform the steps required for initiating a "Remote Job Processing Session".

4.0 SYSTEM CONTROL CARDS

Certain of the control cards recognized by HASP can be introduced from the remote terminal site. Following is a list, with meanings, of these control cards. Column numbers appear over the beginning character of each section of the card.

1	16	71
/*MESSAGE	Any Message	

The data punched into columns 16-71 of this card will be displayed on the central computer operator console at the time the job is being read into the system. This may be used to identify certain jobs, give special instructions, etc. The /*MESSAGE card may be placed anywhere within the input job stream. If this card appears within a job, the HASP number assigned to that job will be appended to the message before displaying it, otherwise the remote station ID will be appended.

1	10	16
/*ROUTE	PRINT	LOCAL
	PUNCH	REMOTEn
		PRINTERn
		PUNCHn

This card, when included anywhere within a job being submitted to the central computer, will cause the print or punch output (as indicated in column 10) to be processed on another remote workstation (REMOTEn), a

specific local printer and/or punch, or the first available local printer and/or punch (LOCAL). This card may be used to divert large volumes of print or punch to local high speed devices to avoid terminal congestion. Both print and punch may be routed locally by including two /*ROUTE cards in a job.

```
1              16
/*PRIORITY     n
```

This card may be used to force the assignment of priority "n" to the job which immediately follows. "n" may be any numerical value from 0-15. This control card when read locally by HASP is interpreted as an absolute priority assignment to a job. However, when read from a remote station the card is regarded as a priority assignment to this job relative to other jobs from the same station. Thus, a remote operator can, via the /*PRIORITY card order the sequence of jobs submitted from only his station, for example, a /*PRIORITY 15 (where 15 is the highest priority) would cause its job to be the next job from that remote station to be processed, although not necessarily the next job to be processed by the centeal computer. The relative position of the priority structure of a remote terminal with respect to the overall system priority structure is determined at HASPGEN by central computer personnel.

The /*PRIORITY card must immediately precede the OS/360 JOB card for the job to which it refers.

```
1               16          25
/*SIGNON        REMOTEn     Password
```

This card appears at the end of the HASP/RMT 360 program deck and is used to override the remote identification number normally assigned to the HASP/RMT360 program deck. For DIAL lines the /*SIGNON card may be used to submit a password which, if correct, will allow the remote terminal access to the HASP system for remote job stream processing. The value "n" must match the remote identification number assigned to the remote station by central computer personnel. The value of the "password" must match the password assigned to the line by the central computer operator when the communication line is "started".

```
1
/*SIGNOFF
```

This card is used to inform the central system that the remote terminal operator desires to terminate a remote job stream processing session. When submitted to the central system, HASP will, at the completion of the current print and/or punch streams, disconnect the terminal from the system and prepare the line for other remote stations to SIGN-ON.

```
1
/*command
```

Selected HASP commands may be submitted to the central system through the remote terminal card reader. Commands submitted in this manner must

be the first cards of a job stream (in front of the first job submitted). Commands which can be submitted are listed in the HASP operator's guide and must start in column 3 of the card, i.e. the first 3 columns will be "/*$". (See Section 2.2.4 for entering HASP commands via the console typewriter).

```
1               16                                                    71
/*SETUP         volume-ser1,volume-ser2,...,volume-sern
```

The volume serials punched in columns 16-71 of the card will be displayed on the central system console and the associated job will be placed in HOLD status (not be scheduled for execution) until released by the central operator. The /*SETUP card appears in the corresponding job input deck between the OS/360 JOB card and the first EXEC card.

(The remainder of this page intentionally left blank.)

11.6 HASP REMOTE TERMINAL PROCESSOR (BSC MODEL 20) OPERATOR'S GUIDE

The following section contains detailed instructions for operating a 360/20, equipped with a Binary Synchronous Communication Adapter, as a HASP MULTI-LEAVING, remote workstation. Although intended for use as a separate operational manual, it has been included in the HASP SYSTEMS manual to achieve completeness.

(The remainder of this page intentionally left blank.)

11.6 HASP REMOTE TERMINAL PROCESSOR (BSC MODEL 20) OPERATOR'S GUIDE

The following section contains detailed instructions for operating a 360/20, equipped with a Binary Synchronous Communication Adapter, as a HASP MULTI-LEAVING, remote workstation. Although intended for use as a separate operational manual, it has been included in the HASP SYSTEMS manual to achieve completeness.

(The remainder of this page intentionally left blank.)

H A S P

REMOTE TERMINAL PROCESSOR

FOR

MULTI-LEAVING BINARY SYNCHRONOUS

COMMUNICATIONS

MODEL 20 OPERATOR'S GUIDE

TABLE OF CONTENTS

1.0 INTRODUCTION

The HASP SYSTEM is an automatic spooling, priority scheduling system which, while operating in conjunction with OS/360, operates an unlimited number of peripheral devices simultaneously with normal job execution, to perform the functions normally associated with off line support computers. The function of HASP has been extended to operate, via several classes of telephone lines, peripheral devices located remotely from the central computer complex.

Through the use of the HASP Remote Job Entry feature, a user, located perhaps thousands of miles from a particular System/360 installation, can utilize the capabilities of that installation much as if the central system were located at the remote site. The unit record devices at a remote station are logically operated by HASP as if they were local readers, printers, punches, and consoles, so that HASP can simultaneously, while operating all local unit record devices, read jobs from several remote readers into the queue of jobs awaiting processing; and output to several remote printers and/or punches results of previously entered jobs which have completed execution.

Although a variety of devices may be utilized as remote terminals, this document discusses only the use of a System/360 Model 20 with a binary synchronous communication adapter as a remote station.

A special program has been written for the Model 20 which can be considered a logical extension of the HASP System. This program referred to in the HASP documentation as (HASP/RMTM20) performs the following functions:

A. INPUT

1. Reads from the attached card reader.

2. Recognizes operator requests and reads from the attached console.

3. Identifies, compresses, and blocks card images and commands for transmission to HASP.

4. Queues blocked records for transmission to HASP.

B. OUTPUT

1. Dequeues blocked records received from HASP.

2. Identifies the device required for output of the records.

3. Deblocks and decompresses output records, queueing the images for printing, punching, or typing.

4. Prints, punches, and types the output records as required.

5. Sets status flags indicating backlog conditions on the output devices.

C. COMMUNICATIONS

1. Establishes and maintains synchronization with HASP.

2. Dequeues blocked input records and transmits them to HASP upon request from HASP.

3. Provides backlog status flags indicating the terminal's ability to receive the various output streams from HASP.

4. Receives output from HASP and queues the blocked records for processing.

HASP/RMTM20 may read print and punch data concurrently depending upon the options selected by the installation and the capabilities of the unit record devices.

Due to the use of blocking and character compression to minimize line transmission time, the speed at which the Model 20 unit record devices will operate is dependent on the data being transmitted, and the number of concurrent functions. Certain job mixes, because of their data characteristics, will enable HASP/RMTM20 to operate the unit record devices at full rated speed. Other job mixes may cause the devices to operate in short bursts because of contention on the communication line.

2.0 OPERATING PROCEDURES

The following pages provide sufficient information for initiating and operating the HASP/RMTM20 program during the remote job stream processing session.

2.1 INITIATION OF A REMOTE JOB STREAM PROCESSING SESSION

The initiation of a remote job stream processing session involves the initial program loading of the HASP/RMTM20 program deck, the establishment of the communication lines, and the exchange of initial control information between HASP and the HASP/RMTM20 program. The initial control sequence ends with the passing of the SIGN-ON remote identification information.

2.1.1 Initial Program Load (IPL)

The following steps should be taken to IPL the HASP/RMTM20.

1. If power is off press POWER ON.
2. Ready the HASP/RMTM20 deck in the supported card reader. (The last card of the deck should be a blank or /*SIGNON card as directed by the installation).
3. Ready the printer, punch, and console (as required).
4. Set time sharing key down.
5. Set the Address/Register Data Switches to one of the following:

   1F00 - 8K storage
   2F00 - 12K storage
   3F00 - 16K storage

6. Set the mode switch to PROCESS.
7. Press LOAD.
8. All cards of the HASP/RMTM20 deck should be read except the last card. Press reader START to read the last card.
9. The IPL is complete when the last card is read. HASP/RMTM20 will print the /*SIGNON card (if the last card) followed by a HASP ENVIRONMENTAL RECORDING ERROR PRINTOUT (if the contents of core remain unchanged since the last running of the program).
10. The BSCA REC INITIAL indicator light should come on.
11. The remote terminal is now ready to communicate with HASP.
12. The toggle switch on the BSCA console may be set to WAIT STATE to stop the CPU meter until HASP communications are established.

2.1.2 Establishment of Communication Line

The procedures for establishing communications with HASP are as follows:

1. Ready the data set. This will involve different actions based on the type of data set. Readying nonswitched lines will only require the data set DATA button be pressed (if present). To ready a dial line data set perform the following:

   a. Press the TALK button and lift the receiver on the data set.
   b. Dial the assigned number for the remote terminal.
   c. If the HASP line is available, the central system will answer with a high pitched tone. Press DATA and replace the hand set in its cradle (the data set is ready).
   d. If the HASP line is in use, a busy signal will be received. Hang up and try again later or dial an alternate communication line number.
   e. If the call is not answered, the central HASP operator has not given the necessary command to authorize use of that communication line.

2. When the data set is made ready the BSCA DATA SET READY and BUSY indicators will be on in addition to alternating TRANSMIT MODE and RECEIVE MODE (RECEIVE MODE may appear to be continuous).

3. When HASP responds to the MOD 20 transmission the SIGN-ON is transmitted to HASP and CARD I/O for the card reader will come on indicating the SIGN-ON is complete. HASP/RMTM20 will "handshake" with HASP until processing of job streams actually begin. And BUSY will be on with alternating RECEIVE MODE and TRANSMIT MODE indicators).

2.2 REMOTE JOB STREAM PROCESSING

During remote job stream processing the operator is concerned with operating the unit record devices, while submitting jobs and controlling the output via commands to the central system.

2.2.1 Output Processing

Except as controlled by the remote terminal operator or central system operator via commands to HASP, the printing and punching of job output is handled automatically by the HASP - HASP/RMTM20 system.

2.2.2 Input Processing

Job submission can be initiated at any time depending upon the capabilities of the card reader - punch combination attached to the Model 20. There is no restriction on when the operator may submit a job stream with the following reader - punch combinations:

1. 2501 reader - 2560 punch (secondary feed)
2. 2501 reader - 1442 punch
3. 2501 reader - 2520 punch
4. 2560 reader (primary feed) - 1442 punch
5. 2520 reader - 1442 punch

The operator need only place the cards in the hopper as desired. The reader will stop just before reading the last card of each job stream. The operator should put more cards in the reader or press START on the reader allowing the last card to be read and the program to recognize the end of the job stream.

The input reader to HASP/RMTM20 is considered always "HOT"; that is the program is continually testing the reader and attempting to read cards. During this time the appropriate CARD I/O indicator on the CPU console will be on (see section 3.0 unit record error procedures). This condition is <u>not</u> an error but indicates that HASP/RMTM20 is ready to send the next job stream.

2.2.3 Input Processing On DUAL Reader Punches

Devices with single card paths for read and punch functions are considered DUAL reader/punch devices. When using these devices as DUAL devices the operator must concern himself with the status of the device. The following are supported DUAL devices:

1. 2520 READER PUNCH
2. 2560 MFCM (READ - PRIMARY FEED)
   (PUNCH - SECONDARY FEED)

Notice that these devices are not considered DUAL devices when used in combinations listed in section 2.2.2.

Operating The DUAL 2520

The 2520 has four basic status conditions which affect the operator:

1. Neutral - Reader empty from normal program execution.
2. Input - Reading normal job stream.
3. Output - Punching normal output from HASP.
4. Output error recovery - attempting to recover from punch errors.

At IPL time the 2520 will be in neutral status and may be treated as any reader device in that the operator is at liberty to submit multiple job streams at any time. Any blank cards mixed in the input stream will be submitted to HASP as job input. When HASP/RMTM20 recognizes the end of file (EOF) the 2520 will revert to the neutral status.

When the 2520 is in neutral the operator may choose to ready the device with blank cards which places it in the output status. If HASP has output waiting, HASP/RMTM20 will respond immediately by punching into the blank cards. However, after all punching is finished or if there is a pause due to low line speeds the operator may <u>not</u> run the remaining cards out of the 2520 and ready it with job stream cards. The procedure for interrupting the output mode is as follows:

1. Press STOP on the 2520.
2. Remove the cards from the HOPPER. (DO NOT NPRO the cards out of the card path).
3. Place the job stream cards in the HOPPER and press reader START.

4. If the punch happens to be busy the device will continue punching until the job stream is encountered; then the 2520 will enter the input status. (Not all blank cards need to be removed from the hopper).

5. If the punch is momentarily idle, it will be waiting for LOCAL COMMANDS from the console (if installed). The operator can cause the 2520 to pass through one card by typing on the console ".SR1" (start reader number 1). If several blank cards are in the hopper in front of the job stream this command must be entered for each blank card. For configurations without consoles the operator can simulate the .SR1 command by setting data dial 2 to numerical value 2 and moving data dial 1 one position in either direction. Care should be taken not to move dial 1 twice and to set data dial 2 out of position 2 upon completion of the skip function.

The 2520 is in error recovery status when a punch error occurs. HASP/RMTM20 will attempt to repunch the record in error into the following card. If HASP/RMTM20 encounters a nonblank card a read error will occur (see unit record error procedures). The operator should non-process run out the job stream cards, place one or more blank cards in front and ready the 2520.

Operating The DUAL 2560 MFCM

The 2560 has two basic status conditions:

1. Input - Submitting jobs using primary feed and hopper.

2. Output - Punching data from HASP using secondary feed.

Blank cards for punching should always be in the 2560 secondary feed hopper for punching purposes during normal processing. During idle periods and periodically while punching HASP/RMTM20 will test the primary feed for job stream cards. If job stream cards are encountered the HASP/RMTM20 will suspend the output status and submit the job stream to HASP. The operator should always press STOP on the DUAL 2560 prior to loading the job stream in the primary feed hopper. (The feed mechanism cycles when HASP/RMTM20 is testing for job stream cards).

2.2.4 Command Processing

Central system commands as well as local commands may be entered into the 2152 operator's console. Any message entered into the 2152 keyboard which is not recognized as a local command will be transmitted to HASP for action. Although all commands transmitted to HASP may be listed on the central system operator consoles, only those designated in the HASP operator's guide as being available to the remote user will be acted upon.

Local commands are available to the HASP/RMTM20 operator for the purpose of signalling the status of the unit record devices. Table 2.2.1 contains a list, with meanings, of all available local commands.

Entering Commands

The operator should perform the following steps when entering commands:

1. Press the REQ button which is located to the right of the console typewriter keyboard. The request indicator (indicator marked "R" at the right of the REQ button) will glow momentarily.
2. When the proceed indicator comes on (indicator marked "P" below the request indicator), type in the command and press EOT.
3. If a typing error is noticed prior to pressing EOT, press CAN (cancel) and repeat step 2.

4. If after receiving a proceed indicator no command entry is desired type "." and press EOT. This will be recognized as an illegal local command and be ignored.

Table 2.2.1 Local Commands

COMMAND	MEANING/COMMENTS
.SR1	"Start reader number one". This command is used to tell HASP/RMTM20 that the operator has corrected a data check condition and has made the card reader ready to continue reading the input job stream (the first card being a corrected version of the card in error). This command is also used in terminating the output status of a DUAL 2520 card reader/punch (see section 2.2.3).
.SU1	"Start punch number one". This command is used to tell HASP/RMTM20 that the operator has removed the incorrectly punched card from the punch stacker (1442) and the punch is ready for the punch of the record.

Commands start in the first available type position and are identified by the period (.). Except for the use of upper or lower case alphabetic characters, the commands must appear exactly as listed. No blanks are allowed.

2.3 TERMINATING A SESSION

When the remote terminal operator desires to terminate remote processing, he should send through the card reader input stream a /*SIGNOFF card. (See section 4.0). This tells HASP not to initiate the sending of any more job output and release the communication line (if DIAL) when the current print and punch streams are finished. The DATA light on the data set will go out and BSCA will enter a CHECK CONDITION. For nonswitched lines HASP will make the line available and thus send initial sequence requests to the HASP/RMTM20 program. Versions of the HASP/RMTM20 which support console messages will log on the console message device, UNIT CHECK in case of a DIAL line or INVALID RESPONSE in case of a nonswitched line. The operator should check to see if printing and punching of output streams have successfully terminated and press STOP on the CPU. To start a new session the operator must perform the steps prescribed for the initialization of a remote job stream processing session.

3.0 ERROR PROCEDURES

The following sections indicate some of the more common error conditions which may arise and the necessary steps for recovery from the error.

3.1 COMMUNICATION ADAPTER ERRORS

The design of the synchronization technique for HASP remote terminals is such that no errors are expected during a processing session. The occurrence, therefore, of any error condition is an unusual condition resulting from either system or communication facility malfunction or operational conditions. In general, the displaying of error messages is informational only since the terminal processor will automatically initiate the appropriate recovery action. A statistical summary of all errors is maintained in the HASP Environmental Recording Table and a historical report is produced each time the HASP/RMTM20 is loaded (unless storage has been cleared). Additionally, if an operator message device has been designated (console or printer), the occurrence of any error will cause a descriptive message to be displayed immediately. Table 3.1.1 indicates each of the possible communication errors which can occur, their meaning and the recovery action taken.

Table 3.1.1 Communication Adapter Error Messages

MESSAGE	MEANING	ACTION TAKEN
01RREE00	Block sequence check - a transmission block was duplicated or lost RR = Received block number EE = Expected block number	If duplicate the received block will be ignored. If lost block, HASP will be signaled to restart the job.
02000000	Negative reply received - a transmission block was not correctly received by HASP	The bad record will be re-transmitted
03RRRR00	Unknown response received - an unrecognizable control character was received from HASP RRRR = First two characters received (If RRRR is correct sequence, ending sequence was bad)	HASP will be requested to retransmit the record
05SS0000	Unit check - a check condition has occurred in the communication adapter SS = Sense byte indicating type of check	The failing operation will be retried.
	Common Examples -	
	SS=01=overrun on write	Write retried
	SS=02=parity check on write	Write retried
	SS=81=overrun on READ	Retransmission requested
	SS=88=lost data on read	Retransmission requested
	SS=90=time out (no response received from HASP in 3 sec.)	Retransmission requested
	SS=A0=transmission error	Retransmission requested
	SS=C0=EOT received	Retransmission requested

3.2 UNIT RECORD ERROR PROCEDURES

Unit record device errors which prevent the execution of I/O will cause HASP/RMTM20 to continuously test the device while performing other functions which are able to continue. The operator is notified of device error by the CPU indicator panel as follows:

CARD I/O 1 - 2501 Card Reader

CARD I/O 2 - 2520 Reader - Punch or 2560 MFCM

CARD I/O 3 - 1442 Card Punch

PRINTER - 1403 or 2203 Printer

Indicators on the device control panel will indicate the nature of the problem. The operator should correct the error in accordance with procedures prescribed for the device and "ready" the device. HASP/RMTM20 will resume its use of the device automatically.

Unit record errors occurring during the actual execution of I/O will result in various program action in accordance with the operator message facilities available for informing the operator and the nature of the error encountered. Table 3.2.1 indicates the program action taken for each device supported by the system. The operator should when notified of the error via the 2152 console perform the following:

1. Note the address code in the error message (see table 3.2.1).
2. Correct the error for "data check" as prescribed for the device.

Table 3.2.1 HASP/RMTM20 Action on Unit Record I/O Execution Errors

DEVICE-FUNCTION	ACTION WITH CONSOLE	ACTION WITHOUT CONSOLE
2501 2520 2460 - read	1. Type error message (note 1) 2. Wait for .SR1 command 3. Read	1. STOP with device address in ESTR register (note 2) 2. Reread when CPU STARTed
1442 - punch	1. Type error message (note 1) 2. Wait for .SU1 command 3. Repunch record in error	1. STOP with device address in ESTR register (note 2) 2. When CPU started repunch record in error
2520, 2560 - punch	1. Select out card in error 2. Repunch record in error	1. Select out card in error 2. Repunch record in error
2203, 1403 - print	Ignore error	Ignore error
2152 - write	1. Ignore first error 2. Wait on next attempt to use device (note 3)	NA
2152 - read	1. Initiate re read 2. Wait on next attempt to use device (note 3)	NA

1. Error message will be of the form: 0500000a UNIT CHECK where a is the device address of the unit in error.
2. Device addresses correspond to the CPU panel CARD I/O indicator numbers.
3. Console error indicator is cleared by pressing OFF LINE then ON LINE.

3. Ready the device for program retry of the I/O.

4. Type the appropriate command (.SR1, .SU1) to signal HASP/RMTM20 that the device is ready.

Without the 2152 the program will stop the CPU with the address of the device in the ESTR register. The operator should without delay perform the following:

1. Note the address of the device in the ESTR register.

2. Press STOP on the indicated device.

3. Press START on the CPU to allow continuation of other functions.

4. Correct the error for "data check" as prescribed for the device.

5. Ready the device for program retry of the I/O.

3.3 REMOTE TERMINAL RESTART

In the event of an untimely interruption of the remote terminal operation such as a machine, program, communications, or environmental failure, the remote terminal operator should notify appropriate maintenance personnel of the malfunction, save material which may be of use in determining the source of the failure, and with the aid of the central system operator prepare for restarting the terminal as follows:

1. Notify the central system operator of the failure and, if necessary request his assistance in preparing for restart.
2. Determine the current job being transmitted to HASP. (The central system operator has a record of the current job being submitted to HASP). The job stream starting with the current job must be submitted to HASP after restart.
3. Determine the loss of data on the output devices and inform the central operator to BACKSPACE or RESTART the printer or punch as necessary. (The central system's line should be made available for a subsequent session with the remote station or other stations within the system).
4. When the remote terminal is available, perform the steps required for initiating a "Remote Job Processing Session".

4.0 SYSTEM CONTROL CARDS

Certain of the control cards recognized by HASP can be introduced from the remote terminal site. Following is a list, with meanings, of these control cards. Column numbers appear over the beginning character of each section of the card.

1	16	71
/*MESSAGE	Any Message	

The data punched into columns 16-71 of this card will be displayed on the central computer operator console at the time the job is being read into the system. This may be used to identify certain jobs, give special instructions, etc. The /*MESSAGE card may be placed anywhere within the input job stream. If this card appears within a job, the HASP number assigned to that job will be appended to the message before displaying it, otherwise the remote station ID will be appended.

1	10	16
/*ROUTE	PRINT	LOCAL
	PUNCH	REMOTEn
		PRINTERn
		PUNCHn

This card, when included anywhere within a job being submitted to the central computer, will cause the print or punch output (as indicated in column 10) to be processed on another remote workstation (REMOTEn), a

specific local printer and/or punch, or the first available local printer and/or punch (LOCAL). This card may be used to divert large volumes of print or punch to local high speed devices to avoid terminal congestion. Both print and punch may be routed locally by including two /*ROUTE cards in a job.

```
1                  16
/*PRIORITY         n
```

This card may be used to force the assignment of priority "n" to the job which immediately follows. "n" may be any numerical value from 0-15. This control card when read locally by HASP is interpreted as an absolute priority assignment to a job. However, when read from a remote station the card is regarded as a priority assignment to this job relative to other jobs from the same station. Thus, a remote operator can, via the /*PRIORITY card order the sequence of jobs submitted from only his station, for example, a /*PRIORITY 15 (where 15 is the highest priority) would cause its job to be the next job from that remote station to be processed, although not necessarily the next job to be processed by the central computer. The relative position of the priority structure of a remote terminal with respect to the overall system priority structure is determined at HASPGEN by central computer personnel.

The /*PRIORITY card must immediately precede the OS/360 JOB card for the job to which it refers.

1	16	25
/*SIGNON	REMOTEn	Password

This card appears at the end of the HASP/RMTM20 program deck and is used to override the remote identification number normally assigned to the HASP/RMTM20 program deck. For DIAL lines the /*SIGNON card may be used to submit a password which, if correct, will allow the remote terminal access to the HASP system for remote job stream processing. The value "n" must match the remote identification number assigned to the remote station by central computer personnel. The value of the "password" must match the password assigned to the line by the central computer operator when the communication line is "started".

1
/*SIGNOFF

This card is used to inform the central system that the remote terminal operator desires to terminate a remote job stream processing session. When submitted to the central system, HASP will, at the completion of the current print and/or punch streams, disconnect the terminal from the system and prepare the line for other remote stations to SIGN-ON.

1
/*command

Selected HASP commands may be submitted to the central system through the remote terminal card reader. Commands submitted in this manner must

be the first cards of a job stream (in front of the first job submitted). Commands which can be submitted are listed in the HASP operator's guide and must start in column 3 of the card, i.e. the first 3 columns will be "/*$". (See Section 2.2.4 for entering HASP commands via the console typewriter).

```
1              16                                            71
/*SETUP        volume-ser1,volume-ser2,...,volume-sern
```

The volume serials punched in columns 16-71 of the card will be displayed on the central system console and the associated job will be placed in HOLD status (not be scheduled for execution) until released by the central operator. The /*SETUP card appears in the corresponding job input deck between the OS/360 JOB card and the first EXEC card.

11.7 HASP REMOTE TERMINAL (2780) OPERATOR'S GUIDE

The following section contains detailed instructions for operating an IBM 2780 as a HASP remote workstation. This manual is intended for use as a removable operator's guide and has been designed to serve as both a tutorial for less experienced 2780 operators and an operating guide for the more experienced.

(The remainder of this page intentionally left blank.)

THE

HASP

SYSTEM

IBM 2780 Remote Workstation Operator's Guide

TABLE OF CONTENTS

1.0 INTRODUCTION

The HASP SYSTEM is a computer program which operates in the central computer. It provides a very efficient means of gathering jobs, scheduling their execution on the bases of job priority and job class, collecting each job's printed and punched output, and returning that output to the submitter of the job. The process of gathering the card images which constitute the job, and of saving the job's output for later printing and punching, is called SPOOLing. While HASP is reading or printing or punching on your 2780, it may be simultaneously reading, printing, and punching on all of the card readers and printers next to the central computer and on all of the other 2780's, the 1130 systems, and the 360 systems to which the central computer is attached for remote job entry.

HASP Remote Job Entry (HRJE) is a feature of the HASP system whereby installations that are remote from the central computer may send jobs to the central computer for execution and receive back their printed and punched output. HRJE supports as remote terminals all models of System/360, the 1130 system, 2780's, and 1978's. A remote terminal may be at any distance from the central computer. It may be next door, or it may be thousands of miles away. The only requirement is that some means (usually telephone lines) exists to allow it to communicate with the central computer.

Jobs to be submitted from a remote terminal have exactly the same Job Control Language control cards as jobs that are submitted directly to the central computer. Their output is routed back to the terminal from whence they came, unless special HRJE control cards or operator commands specify differently.

The IBM 2780 Data Transmission Terminal can connect to a System/360 using HASP to transmit jobs to the 360 for execution and to receive the printed and punched output from those jobs. The 2780 is not a computer but rather an input/output device. HASP controls it much like any other input/output device, when connected to the central computer via telephone lines and an IBM 2701 Data Adapter Unit or an IBM 2703 Transmission Control Unit.

The 2780 consists of at most one printer and one reader-punch. The maximum speeds are 400 cards per minute for the reader, 355 cards per minute for the punch (if only column 1 of each card is punched), and about 240 lines per minute for the printer. However, the actual speeds of these devices depend heavily on the speed of the telephone line and upon the number of trailing blanks on each line to be printed and card to be punched.

Certain special features of the 2780 are of concern to you as an operator. These features are called EBCDIC Transparency and Auto Turnaround.

For System/360, EBCDIC is the character and punched-card code normally used. This code allows a column of a punched card to be punched in any of 256 different ways. Certain of these punch combinations correspond to control characters to which the 2780 will respond if it is not in transparency mode. However, some 360 programs (for example, all assemblers and compilers) punch cards using the complete set of 256 punch combinations (for example, object decks). If you intend to read these cards into a 2780, it must have the Transparency feature, and you must set the mode selection knob to TSM TRSP. As a rule of thumb, always use this setting, rather than the TSM setting, if your 2780 has the EBCDIC Transparency feature.

If your 2780 has Auto Turnaround as a feature, it has a back-lighted pushbutton to turn this feature on and off. The effect of this feature is to switch the reader-punch automatically from a reader to a punch while reading in a job; this happens when the reader reads an all-blank card. Therefore, you must be careful when using this feature that no blank cards are imbedded in your job. The advantage of auto-turnaround is that no operator intervention is required to start punching the output of a job.

The remaining sections of this manual discuss normal operating procedures for reading, printing, and punching jobs; the sign-on procedure; error recovery; and control cards.

2.0 OPERATING PROCEDURES

This section discusses procedures for transmitting jobs to the central computer and for receiving their printed and punched output. Throughout this manual, the assumption is that your 2780 has a reader-punch and a printer (that is, it is a 2780 Model 2). Refer to section 2.2 for the sign-on procedure.

At any given time, a signed-on 2780 is either reading in a job, printing a job, punching a job, or waiting for work. Often, after an operator has read in a job through the 2780, he will disconnect (sign-off) the 2780 to save telephone line charges and sign-on at a later time to receive his output. There will be printed output for every job submitted; there will be punched output only if the job requires it.

Thus, the normal cycle of a job submitted from the 2780 is: reading the job (transmitting it to the central computer), waiting for it to execute, receiving its printed output, and possibly receiving its punched output.

In the following descriptions, certain time estimates in seconds are given. These are based on the default value of a certain variable in the HASP system. Your installation may have changed this default; if so, the time estimates will be greater or less than specified.

2.0.1 POWER-ON RESET

The power-on reset operation is referred to frequently throughout this manual. Contrary to its name, a power-on reset does not involve the 2780 power on-off switch (on the right side of the 2780); this switch need be turned on only once during 2780 operations.

You do a power-on reset by merely turning the mode selection knob from its current position to any other position; this resets the 2780. If the knob is already where you want it, then turn it to some other position and back to its current position.

For example, turning the knob from REC to TSM, or from OFFLINE to TSM, or from TSM to OFFLINE and back to TSM is sufficient to do a power-on reset.

CAUTION: Do not do a power-on reset while the printer is printing, or while cards are being read or punched.

2.1 INITIATING PROCESSING

The following section contains sufficient information to allow the initiation of a remote job stream processing session.

2.1.1 TRANSMISSION TO THE CENTRAL COMPUTER

You can transmit jobs to the central computer at one of three times.

- Immediately after you have signed on.
- In the pause (about 10 seconds) after the 2780 has finished printing or punching output from a previous job.
- When the 2780 is signed on and waiting for work.

You cannot interrupt punching in the middle of a job to start transmitting a job. You **must** not do a power-on reset while a job is being printed or punched. The 2780 will not transmit or receive jobs unless you have correctly signed on.

To read in a job, take the following steps:

1. If a job is punching, wait till no cards have been punched for a period of 5 to 10 seconds. Then
   a. Do a power-on reset, leaving the knob at TSM TRSP (or TSM).
   b. Remove the blank cards from the hopper and the punched cards from the stacker.
   c. Press NPRO to run the two blank cards out of the feed mechanism.

2. Load the cards you want to read into the card hopper. Jobs may be stacked one on top of the other.

3. If a job is printing, wait till no lines have been printed for a period of 5 to 10 seconds. Then do a power-on reset, leaving the knob at TSM TRSP (or TSM).

4. If you haven't already done so, turn the mode-selection knob to TSM TRSP (or TSM).

5. Push the END OF FILE button and the START button so that the END OF FILE and READY lights come on. Cards should start reading within about 15 seconds.

If cards do not start reading, or if the reader stops before the last card has been read, see section 3.1. The reader will normally read 2 to 7 cards, pause a bit to transmit them to the central computer, and then read 2 to 7 more.

If you make the printer ready while you are transmitting (see 2.1.2), it will be able to receive after transmission is finished without further intervention.

If a job is printing and the selection knob is in either of the TSM positions (see 2.1.2), you may make the reader ready as in step 5. When the job's printing is finished, card reading and transmission should begin without further intervention.

It may be possible to interrupt printing only to begin transmitting. See Section 3.2 for details.

2.1.2 RECEPTION FROM THE CENTRAL COMPUTER

After your job has completed reading, HASP queues it for execution at the central computer. As the job executes, it may produce printed and punched output. HASP saves these outputs and at the end of the job queues the printed output for printing, usually upon the terminal from which the job was read. You may have turned off the 2780 or otherwise disconnected it from the computer; if so, you must follow the sign-on procedure before you can receive output from the job.

After a job has finished printing, HASP queues its punched output (if any) for punching, usually upon the terminal from which the job was read. When HASP finds that any output device is ready on the 2780, it inspects that remote terminal's output queues in the order punch first, then print. Thus, if you are expecting the printer to start, HASP may actually be trying to punch the output from a previously printed job. In this case, the printer's READY light would be on and the TERM ADDR light would also be on. You must let HASP punch before it will start printing.

To receive a job's printed output (assuming you have correctly signed-on), perform the following steps:

1. Make sure the mode-selection knob is at one of the positions TSM TRSP, TSM, or REC. Do not put the knob in the PRINT position.
2. Push START on the printer.

If this terminal's punch queue is empty and its print queue is non-empty, printing should start within about 15 seconds. If the TERM ADDR light comes on when the printer is ready, HASP has found a job to punch; you must ready the punch.

If the printer will not become ready, see Section 3.2. When behaving normally the printer should print from two to seven lines, pause a bit to receive more print lines, and print two to seven more lines.

To receive a job's punched output (assuming you have signed on correctly), perform the following steps.

1. Make sure the mode-selection knob is set to REC. Do not use the PUNCH setting.
2. Use the NPRO button to clear the card feed.
3. Put blank cards in the card hopper.
4. Press START on the reader-punch, and hold it in until the READY light comes on.

Punching should start within 15 seconds, if there is anything to punch. Possible problems are discussed in Section 3.2.

The audible alarm will sound at the end of each job's printed or punched output and will turn off if HASP sends more output or if you start transmitting (see 2.1.1).

2.2 ESTABLISHMENT OF COMMUNICATION LINE

Before your 2780 terminal can transmit jobs to or receive printed or punched output from the central computer, the computer must establish a path of communication to the terminal, and must recognize it as a 2780 terminal (rather than, say, a System 360 Model 20 being used as a terminal). Depending on how your 2780 is connected to the central computer, you may or may not have to use a special control card, called a sign-on card.

If your terminal is permanently connected to the central computer (probably through private lines leased from the telephone company) you need only inform the central operator, who will then issue the HASP command:

```
$START   LNEmm
```

(where mm is a one or two digit decimal number). Either you or the central operator will know the proper line number to use, depending upon your installation. Once the operator has given this command and the 2780 mainline switch is turned on, you may begin to read in a job, or to print or punch the output from a previous job.

However, if your 2780 is connected by ordinary switched telephone lines to the central computer, you will have to dial a telephone number to establish communications. You will have a sign-on card, and a telephone number to call. Carefully perform the following steps.

1. Turn on the 2780 mainline switch (it's on the right side).
2. Push the NPRO button on the card reader. Hold it in for a few seconds to make sure the card feed mechanism is clear.
3. Place the sign-on card and the card weight in the card hopper.
4. Turn the mode-selection knob to TSM TRSP (or TSM).
5. Dial the telephone number you have been given. The TALK button on the dataphone must be depressed to do this.
6. Listen for the normal sound of ringing followed by the normal sound of answering. You should then hear a high-pitched tone of about six seconds' duration, followed immediately by a short bleep.
7. When you hear the bleep, push the DATA button on the telephone and watch for the DATA SET READY light on the card reader. (You can hang up the telephone handset now.)
8. When the DATA SET READY light comes on, press the END OF FILE button and the START button, so that the END OF FILE light and the READY light come on. The reader should now read the sign-on card.

Your 2780 is now signed on, and you may start reading, printing, or punching.

If you do not hear the telephone being answered, the central operator has not issued the command

```
$START   LNEmm
```

or has issued it for the wrong line. Ask him to issue the proper command, and then re-dial if necessary. If instead of ringing you hear a busy signal, the line is of course busy. Call back in a few minutes, or try an alternate number if available.

If your sign-on card will not read, run out the sign-on card, place it in the reader again, do a power-on reset, and repeat step 8.

3.0 ERROR RECOVERY

A wide variety of problems can occur when operating almost any type of machine, including the 2780. Most problems occur only rarely, and many of those are not documented here. See the SRL "IBM 2780 Data Transmission Terminal - Component Description", form A27-3005 for a more complete description of such problems as well as how to load paper in the printer, how to fix a card jam, etc. A copy of this SRL should be near your 2780.

Most problems you encounter will result in lights appearing on the reader-punch or printer control panels. Some of these lights are not error lights. These are DATA SET READY, the two READY lights, END OF FILE, I/O BFR FULL, CTR 1, CTR 2, and CTR 4, and usually LINE. Other lights provide clues to the difficulty, and will be discussed in the following two sections. The first section describes troubles you may have in transmitting jobs to the central computer; the second section describes troubles in receiving printed or punched output.

3.1 ERROR RECOVERY WHEN TRANSMITTING

TERM ADDR - This light may come on while you are readying the card reader. If cards do not read, the READY light is on, and the TERM ADDR light is on, follow carefully these steps:

1. Remove the cards from the hopper and press NPRO to run out the two cards in the feed.
2. Put these two cards in front of the cards you removed from the hopper, and place the cards back in the hopper.
3. Do a power-on reset.
4. Wait for the TERM ADDR light to come on again. This may take as long as about 10 seconds.
5. Do another power-on reset, push END OF FILE and push START, so that the END OF FILE and READY lights come on.
6. Cards should start reading within 15 seconds. If they don't, and the TERM ADDR light comes on again, repeat the above steps.

If the above steps continue to fail, you may have interrupted printing or punching to read in a job before the printing or punching of a previous job had completed. Try readying the printer and punch as described in section 2.1.2.

DATA CHECK,

DATA CHECK and EQUIP CHECK,

DATA CHECK, EQUIP CHECK, and PARITY CHECK -

The reader could not read a card correctly. Do the following steps.

1. Remove cards from hopper (not stacker).
2. Push NPRO. Two cards will run out into the stacker. The first of these cards is the bad one.
3. Correct the bad card.
4. Put both cards back in the hopper, followed by the cards you removed from the hopper.
5. Push END OF FILE and START so that the END OF FILE and READY lights come on.

OVER RUN,

PARITY CHECK,

PARITY CHECK and EQUIP CHECK,

RECORD and LINE -

To correct errors when these lights are on, you need first to find out how many cards have been read but not yet transmitted. Add up the CTR lights to do this. For example, if CTR1 and CTR 2 are on, 3 cards have been read but not yet transmitted.

Without removing all the cards from the stacker,

1. Remove cards from hopper.
2. Press NPRO to run out the 2 cards from the feed mechanism.
3. Remove from stacker the last N+2 cards stacked, where n is the number of cards read but not yet transmitted.

4. Put these cards back in the hopper, followed by the cards you removed from the hopper. Do a power-on reset.

5. Press END OF FILE and START so that the END OF FILE and READY lights come on.

EQUIP CHECK -

A mechanical error has occurred. Use the procedure for DATA CHECK, but you shouldn't need to correct a card.

LINE -

Wait a few moments to see if the reader will start reading by itself. If it doesn't,

1. Push STOP.

2. Press END OF FILE and START so that the END OF FILE and READY lights come on.

HOPR -

No card was fed. Check the edges of the cards at the bottom of the hopper, and repair them if necessary. Then put them back, and press END OF FILE and START so that the END OF FILE and READY lights come on.

For other combinations of error lights, consult the 2780 manual. Most other errors involve misfeeds or jams.

3.2 ERROR RECOVERY WHEN RECEIVING

Some of the errors you may encounter while receiving are self-explanatory, such as END OF FORM (you need another box of paper) or FORM CHECK (the paper is jammed).

One error deserves particular attention; it is indicated by OVER RUN and INCP. If you get this error, you may have specified the wrong REMOTE number on your sign-on card, and HRJE is attempting to use features your 2780 doesn't have. You will have to sign on again, using a correct REMOTE number. If you were using the same, incorrect number when you read in a job, you should call the central computer operator and ask him to re-route the output of your job. In any case, you may have received output that is not yours; if so, you must inform the central computer operator.

Other than these, you may see the following error indicators.

TERM ADDR -

The device (printer or punch) to which HASP is trying to transmit is not ready.

1. Push STOP and CHECK RESET on the reader-punch.
2. Make the output device ready.

EQUIP CHECK -

The punch has mechanically malfunctioned. Run out and throw away the cards in the feed mechanism, and make the punch ready again.

SYNC CHECK -

The printer has erroneously printed a line.

1. Push STOP.
2. Push RESET (on the printer).
3. Push the START button on the printer.

   You will get a duplicate print line.

PARITY CHECK -

If the printer was printing, do a power-on reset and push the START button on the printer. You may get some duplicate print lines.

If the punch was punching,

1. Remove cards from hopper (not stacker).
2. Press NPRO to run out the two cards in the feed mechanism.
3. Throw away the N+K last cards stacked. N is the number represented by the CTR lights. For example, if CTR 1 and CTR 2 are lit, N is 3; if all CTR lights are off, N is 0.

   K is

   2 if all CTR lights are off and the I/O BFR FULL light is on;

   1 if some or all CTR lights are on and the I/O BFR FULL light is on;

   2 if the I/O BFR FULL light is off.

4. Reload blank cards in the hopper, and make the punch ready.

Most other errors are jams or misfeeds. Look at the 2780 manual for instructions on how to fix these problems.

Depending upon the central HASP System at your installation, actions during <u>printer</u> <u>only</u> error recovery may be somewhat different than described above. If this altered mode of printer operation is applicable to your 2780, when you make the printer ready after any of the above stops the job which was printing will be "suspended", a message and terminal separator line(s) will be printed, and the job will be re-queued in the print queue for your terminal. You may cause this "suspend" action yourself by pressing STOP while printing, then readying the printer.

Actions after the printer "suspend" depend upon the state of your terminal and the output queues. You may start transmission as in Section 2.1.1 or wait for more output as in Section 2.1.2. Print jobs of higher priority than the suspended job or any punch jobs will be received before the suspended job. When the suspended job resumes printing, it will do so at approximately one page prior to the page of interruption.

4.0 CENTRAL COMPUTER CONTROL

This section describes the control cards you may use to sign on, send a message to the central computer operator, change the destination of printed and/or punched output, and force the priority of a job.

```
1              16            25
/*SIGNON       REMOTEnn      password
```

This is the sign-on card. The number nn is a one or two digit decimal number whose purpose is to correlate this remote device with information about it in the central computer. Leave the password field blank unless you are required to give a password.

```
1
/*SIGNOFF
```

You may use the sign-off card after the last job you read in. If you use this card, the telephone circuit will disconnect after about 30 seconds.

```
1              16
/*MESSAGE      message
```

When you read in this card, the contents of columns 16-71 will immediately be printed on the central computer operator's console. You may place this card anywhere within a job; it will be deleted before the job is processed.

The typed message will automatically have the job number appended to it if it is found within a job. If it is found outside a job, the remote terminal ID will be appended.

1	10	16
/*ROUTE	PUNCH	LOCAL

This card causes punched output for the job within which it was found to be punched at the central computer instead of at the remote terminal.

1	10	16
/*ROUTE	PRINT	LOCAL

This card does the same thing for printed output that the above card does for punched output. You may use both card; a good place to put most of the cards described here is right after the // JOB card.

1	16
/*PRIORITY	nn

If you use this card, it must immediately precede the // JOB card. The number nn is some one or two digit number between 0 and 15, inclusive. It specifies the urgency with which the job should be processed relative to other jobs submitted from the same remote terminal.

Depending upon features of the central HASP System at your installation, you may use a subset of central HASP operator commands, submitted on cards as follows:

```
1
/*command
```

In place of "command", you should punch any of the commands listed in Table 1.1.3 of the central HASP Operator's Guide which are valid from a remote location. For example, "$DQ,4" punched following the "/*" causes a display of the number of jobs in various queues at the central site which are routed to the terminal REMOTE4.

A group of one or more command cards may be transmitted alone or may be placed in front of a group of jobs being transmitted.

Responses to commands from HASP are printed on the printer, after the paper is positioned at the top of a new page. Such responses are always received first after a transmission is completed, before any job's printed or punched output is received. Certain spontaneous messages (i.e., not responses) are also received. They are: messages acknowledging each job transmitted by your terminal and messages from other operators in the system to you by use of the "$DM" command.

You should read the HASP Operator's Guide to learn about the various commands you may use (Table 1.1.3), what their effects are, and how they should be constructed. Also, output device control and special forms processing are discussed in Section 7 of that Guide. However, certain properties of terminals like your 2780 require more explanation of these two topics.

Certain commands which control output devices ($B, $C, $E, $F, $I, $N, $Z) actually refer to a job currently in active processing on that device which is to be backspaced, restarted, etc. When you submit commands from your 2780, no output devices are active, therefore these commands have no effect. This is true even after you "suspend" a print job as previously described in Section 3.2. The "suspend" is functionally equivalent to $I, which includes the function of $B. To use the other commands, you must ask the central operator to enter them.

The $S, $P, and $T device commands are effective when submitted from your terminal. Furthermore, the $C command is effective when referring to a job rather than a device. The $H, $A, and $R commands may also be used effectively.

Special forms for printed or punched output can effectively be controlled from your terminal without central operator assistance, if all jobs submitted from (or routed to) your terminal follow certain conventions in requesting special forms. Programmers should be required to use only special routing output classes (J and K normally) with requests for special forms by data set. Special print forms for an entire job may be requested in the HASP accounting field of the JOB card. In no case should special forms be requested when using the ordinary output classes (A and B normally) as this will cause the system itself to request mounting of special forms at a time when you, as 2780 operator, are unable to enter the $S command to continue.

Assuming the above conventions, you should periodically submit the $DF command to determine if special forms jobs are queued for output on your terminal. If so, you should select the type of forms from those queued which you desire to process first on each output device, mount those forms, enter a command "$T device,F=forms#" for each device, and wait for printing and/or punching to occur. When jobs stop processing on a device, you should resubmit the $DF command and change to a new forms if indicated. The parameter "F=RESET" should be used to return a device to ordinary output processing. The "F=AUTO" parameter should not be used.

You may want to use the $P and $S device commands, prior to and after the $T command respectively, to prevent HASP from attempting to send an output job while you are changing forms.

11.8 HASP REMOTE TERMINAL (2770) OPERATOR'S GUIDE

The following section contains detailed instructions for operating an IBM 2770 as a HASP remote workstation. This manual is intended for use as a removable operator's guide and has been designed to serve as both a tutorial for less experienced 2770 operators and an operating guide for the more experienced.

(The remainder of this page intentionally left blank.)

THE

HASP

SYSTEM

IBM 2770 Remote Workstation Operator's Guide

TABLE OF CONTENTS

INTRODUCTION

The HASP SYSTEM is a computer program which operates in the central computer. It provides a very efficient means of gathering jobs, scheduling their execution under OS/360 based on job priority and job class, collecting each job's printed and punched output, and returning that output to the submitter of the job. The process of gathering the card images which constitute the job, and of saving the job's output for later printing and punching, is called SPOOLing. While HASP is reading or printing or punching on your 2770, it may be simultaneously reading, printing and punching on all of the other 2770s, 2780s, the 1130 systems and the 360 systems to which the central computer is attached for remote job entry.

HASP Remote Job Entry (HRJE) is a feature of the HASP system whereby installations that are remote from the central computer may send jobs to the central computer for execution and receive back their printed and punched output. HRJE supports as remote terminals all models of System/360, the 1130 system, 2770s, 2780s and 1978s. A remote terminal may be at any distance from the central computer. It may be next door, or it may be thousands of miles away. The only requirement is that some means (usually telephone lines) exists to allow it to communicate with the central computer.

Jobs to be submitted from a remote terminal have exactly the same OS/360 Job Control Language cards as jobs that are submitted directly to the central computer. Their output is routed back to the terminal from whence they came, unless special HRJE control cards or operator commands specify differently.

The IBM 2770 Data Communication System can connect to a System/360 using HASP to transmit jobs to the 360 for execution and to receive the printed and punched output from those jobs. The 2770 is not a computer but rather an input/output device. HASP controls it much like any other input/output device, when connected to the central computer via telephone lines and an IBM 2701 Data Adapter Unit or an IBM 2703 Transmission Control Unit.

The 2770 System may have a wide variety of I/O devices attached. However, when operating your 2770 with HASP, you will be concerned only with the standard keyboard and if attached, the card reader, printer and card punch. These devices (mechanical features, speeds, etc.) are described in the SRL "System Components: IBM 2770 Data Communication System", form A27-3013 which you should have on hand for reference when operating your 2770. Actual speeds at which these devices will operate when communicating with HASP depend upon the type of telephone line used and on the amount of information in each line or card to be transmitted or received, as well as the devices' mechanical speeds which are given in the SRL.

Certain special features of the 2770 are of concern to you as an operator. One of these features is called EBCDIC Transparency.

For System/360, EBCDIC is the character and punched-card code normally used. This code allows a column of a punched card to be punched in any of 256 different ways. Certain of these punch combinations correspond to control characters to which the 2770 will respond if it is not in transparency mode. However, some 360 programs (for example, all assemblers and compilers) punch cards using the complete set of 256 punch combinations (for example, object decks). If you intend to read these cards into a 2770 or receive such cards from the central computer for punching at the 2770, it must have the Transparency feature.

You should also know if your 2770 has the Buffer Expansion feature. This feature affects device performance and the amount of information which can be sent to HASP in a single transmission from the keyboard.

The Keyboard Correction feature, if present, will make it easier for you to correct errors in keyed data before transmission to HASP.

The remaining sections of this manual discuss switch setup; communications establishment; normal operating procedures for reading, printing and punching jobs; error recovery; and control cards.

SWITCH SETUP

During all 2770 operations with HASP, certain console switches should be set as follows:

JOB SELECT - VARIABLE SELECT

INPUT KEYBOARD, 2 (card reader) - both up

OUTPUT PRINTER, 2 (card punch) - both up

DIRECT DATA OUTPUT PRINTER - up

TERM MODE - LINE

SELECTION REQD - up

ANSWER - MANUAL

MONITOR PRINT - as desired by installation, normally down

Any of above which refer to devices not on your 2770 - down

All other VARIABLE SELECT switches - down

On installation, your 2770 may have been provided with the equivalent of all the above switch settings at one of the five JOB positions on the JOB SELECT switch. If so, simply set to that position and ignore all VARIABLE SELECT switches. When power is on, console lights will show the settings which are in effect.

Your 2770 card reader may be attached to the INPUT 3 position rather than INPUT 2. Simply set the INPUT 3 switch up instead of 2.

The TRANSPCY switch should be set in the down position except when used for transmitting EBCDIC card decks which use all 256 possible punch combinations.

ESTABLISHMENT OF COMMUNICATIONS

Before your 2770 terminal can transmit jobs to or receive printed or punched output from the central computer, the computer must establish a path of communication to the terminal and must recognize it as a 2770 terminal (rather than, say, a System/360 Model 20 being used as a terminal). Depending upon how your 2770 is connected to the central computer, you may or may not have to use a special control card, called a sign-on card.

If your terminal is permanently connected to the central computer (probably through private lines leased from the telephone company), you need only inform the central operator who will then issue the HASP command:

$START LNEmm

(where mm is a one or two digit decimal number). Either you or the central operator will know the proper line number to use, depending upon your installation. Once the operator has given this command and the 2770 Power On switch is turned on, you may begin to read in a job, or to print or punch the output from a previous job.

However, if your 2770 is connected by ordinary switched telephone lines to the central computer, you will have to dial a telephone number to establish communications. You will have a sign-on card and a telephone number to call. Carefully perform the following steps.

1. Turn on the 2770 Power On switch.
2. Push the STOP button then the NPRO button on the card reader. Hold it in for a few seconds to make sure the card feed mechanism is clear.
3. Place the sign-on card and the card weight in the card hopper.
4. Dial the telephone number you have been given. The TALK button on the dataphone must be depressed to do this.
5. Listen for the normal sound of ringing followed by the normal sound of answering. You should then hear a high-pitched tone of about six seconds' duration, followed immediately by a short bleep.
6. When you hear the bleep, push the DATA button on the telephone. The DATA SET READY light on the console should come on. (You can hang up the telephone handset now.)
7. Press TERM RESET on the console, turn the card reader EOF switch on, and push the card reader START button to run the sign-on card into the card feed.
8. Press the START button on the console; the BID light should come on. Momentarily, the card reader should read the sign-on card and move it into the stacker.

Your 2770 is now signed on, and you may start reading, printing or punching.

If you do not hear the telephone being answered, the central operator has not issued the command

$START LNEmm

or has issued it for the wrong line. Ask him to issue the proper command and then re-dial if necessary. If instead of ringing you hear a busy signal, the line is of course busy. Call back in a few minutes or try an alternate number if available.

If your sign-on card will not read, run out the sign-on card, place it in the reader again and repeat steps 7 and 8.

If still unsuccessful, call the central computer operator to verify that you have the correct sign-on card and telephone number, and that he has started the line correctly.

OPERATING PROCEDURES

The next two sections discuss procedures for transmitting jobs to the central computer and for receiving their printed and punched output. The 2770 will not transmit or receive jobs unless you have correctly signed on. Refer back to page 6 for the sign-on procedure.

At any given time, a signed-on 2770 is either reading in a job, printing a job, punching a job or waiting for work. Often, after an operator has read in a job through the 2770, he will disconnect (sign-off) the 2770 to save telephone line charges and sign-on at a later time to receive his output. There will be printed output for every job submitted; there will be punched output only if the job requires it.

Thus, the normal cycle of a job submitted from the 2770 is: reading the job (transmitting it to the central computer), waiting for it to execute, receiving its printed output, and possibly receiving its punched output.

In the following descriptions, certain time estimates in seconds are given. These are based upon the default value of a certain variable in the HASP System. Your installation may have changed this default; if so, the time estimates will be greater or less than specified.

TRANSMISSION TO THE CENTRAL COMPUTER

The 2770 can transmit jobs to the central computer only if signed on and not busy printing or punching. However, you may make it ready to transmit any time that it is signed-on. You cannot interrupt punching in the middle of a job to start transmitting a job. You must not press TERM RESET while a job is being printed or punched. If you accidentally do this, a line restart (described on page 14) must be done.

To transmit job(s), take the following steps:

1. Push the card reader STOP button then the NPRO button to clear the feed.
2. Place one or more jobs in the card read hopper. Jobs may be stacked one on top of the other.
3. Push card reader START to run cards into the feed. The INPUT 2 light on the console should stop blinking and come on steady indicating that the card reader is ready. Turn on the reader EOF switch if all the cards you intend to transmit fit in the hopper.
4. Turn on the console TRANSPCY switch if required by the cards to be transmitted. (See previous discussion on page 2.)
5. Press the START button on the console; the BID light should come on.
6. If the 2770 is printing or punching, it will continue until the end of the current job. Then, or as soon as START is pressed if the 2770 is idle, the 2770 will ask

permission to transmit. When the central computer answers affirmatively (within 15 seconds), the BID light should go out and cards should begin reading into the stacker.

If you add more cards, be sure to turn on the reader EOF switch when all cards you intend to transmit are in the hopper. If you allow the hopper to become empty in the middle of a job's input, you must not have the EOF switch on.

You may push STACKER UNLOAD on the card reader at any time to halt reading temporarily to facilitate removing cards from the stacker or adding more to the hopper. Push reader START to continue or wait for the reader to automatically continue in 30 seconds.

The keyboard can be used to transmit short jobs or control cards alone, or can be used to transmit typed cards in front of more cards read from the card reader in a single transmission.

To use the keyboard:

1. Wait until the current job, if any, is finished printing or punching.
2. Turn off the TRANSPCY switch.
3. Press TERM RESET on the console and KEY REQ on the keyboard. The console PROCEED light should come on.
4. Type in one or more lines as if they were cards of 80 or less columns. Use the END CARD key to end each card.
5. Press the ENTER key to transmit what you have typed. The PROCEED light should go out, the BID light should come on, then go out when transmission is complete.

The keyboard transmits letters in lower case unless you upshift. You must upshift to transmit letters as they would be transmitted if keypunched on cards.

Keyed information should appear on the printer as you type. If mistakes are made, you must repeat from step 3 and retype everything. See the SRL on page 2772-24 for better correction procedures if you have the Keyboard Correction feature.

You may use the keyboard instead of the card reader to transmit the sign-on card in the procedure previously described on page 6.

To transmit keyed cards in front of those read from the card reader, do steps 1 through 4 and then instead of step 5, follow the previously described procedure for the card reader. Keyed information is always transmitted in non-transparency, therefore cards following keyed information must also be transmitted in non-tranparency.

The maximum number of cards which can be transmitted from the keyboard in a single transmission is two, without the Buffer Expansion feature. With the feature, a variable number of cards up to the capacity of two 256 character buffers can be transmitted. In either case, when the limit is reached, the keyboard locks after the END CARD key is pressed. You must then cause transmission with ENTER, or START if you are transmitting from the card reader after keying as described above.

It may be possible to interrupt printing only to begin transmitting. See page 20 for details.

RECEPTION FROM THE CENTRAL COMPUTER

After your job has completed reading, HASP queues it for execution at the central computer. As the job executes, it may produce printed and punched output. HASP saves these outputs and at the end of the job queues the printed output for printing, usually upon the terminal from which the job was read. You may have turned off the 2770 or otherwise disconnected it from the computer; if so, you must follow the sign-on procedure before you can receive output from the job.

After a job has finished printing, HASP queues its punched output (if any) for punching, usually upon the terminal from which the job was read. When HASP finds that any output device is ready on the 2770, it inspects that remote terminal's output queues in t order punch first, then print. Thus, if you are expecting the printer to start, HASP may actually be trying to punch the output from a previously printed job.

The 2770 will receive either printed or punched output from the central computer if HASP has output to send, the terminal is not transmitting, and the output devices are ready. You should always have the printer and punch ready, even when transmitting, so that the 2770 can automatically begin receiving when transmission is finished.

The printer is ready if it is loaded with forms, has a correct carriage tape, if the carriage is engaged, and the cover is closed.

To ready the card punch, turn power on, place blank cards in the hopper, set the punch keyboard switch to KEY PCH, place a card with "D" punched in columns 2-80 on the Program Drum and lower the star wheels, press the FEED key twice and the RELEASE key once, then set the switch to AUTO PCH. The AUTO light should come on and the CHECK light on the card punch should go out. See SRL pages 545-11, 12, 18 for more details.

Blinking OUTPUT PRINTER or OUTPUT 2 lights on the console indicate that the above devices are not ready. Even after making them ready, it may be necessary to press CHECK RESET and START on the console to make the lights stop blinking and the devices ready to receive.

Printed and punched output jobs will be separated by separator pages or cards respectively, which are described in the central computer HASP Operator's Guide.

ERROR RECOVERY

A wide variety of problems can occur when operating almost any type of machine, including the 2770. Some problems occur only rarely and are not documented here. See the SRL "System Components: IBM 2770 Data Communication System," form A27-3013, for a description of any problems you encounter which are not discussed in this Guide, as well as how to load paper in the printer, how to fix a card jam, etc. A copy of this SRL should be near your 2770.

In general, there are three levels of error recovery which you may have to perform, depending upon the severity of the error. They are:

1. Fix the difficulty (a not ready I/O device, check condition, etc.) and continue. See the following two sections for the most common examples.
2. Job restart. This is done when the possibility of incorrect or lost data exists and requires the assistance of the central computer operator. Job restart procedures for both transmitting and receiving are described in the following two sections.
3. Line restart. This is done usually when job restart is unsuccessful or any time it is necessary to press TERM RESET to clear a check condition during printing or punching. You should tell the central computer operator to issue the HASP command:

   $RESTART LNEmm

then re-establish communications as previously described on pages 5 and 6. Incomplete input or output jobs are handled as described for job restart in the following two sections.

If even line restart fails to establish successful operation, you probably have a hardware and/or software problem which must be analyzed by your installation's systems personnel and IBM Customer Engineers.

Most problems you encounter will result in lights appearing on the 2770 console or the I/O devices themselves. Some of these lights are not error lights. These are DATA SET READY, CARRIER OFF, DATA IN BUFFER, LINE MODE, PROCEED, BID, SELN REQD, TRNSPCY, MANUAL ANSWER, and any of the I/O device lights when on steady. Any I/O device light which is blinking indicates that the device is not ready. Other lights provide clues to the difficulty and will be discussed in the following two sections.

ERROR RECOVERY WHEN TRANSMITTING

If job restart is required while transmitting, the OS job which is only partially read into the 2770 must be re-read from the beginning. You should ask the central computer operator to issue the HASP command:

$DELETE RMnn.RD1

to delete the partially read job. Press TERM RESET. Load the hopper beginning with the JOB card of the incompletely read job, push reader START and console START.

Any card reader trouble while transmitting is indicated by a blinking INPUT 2 or 3 light, whichever your card reader is attached to. The following lights on the card reader may further indicate the type of trouble.

FEED CHECK - The bottom card in the hopper failed to feed. Remove hopper cards. Push NPRO. Repair bottom hopper card if necessary and make sure the feed throat is clear. Reload cards. Push reader START and console START.

ATTENTION - Full stacker, empty hopper with EOF off, and cover open are possible causes. Correct, push reader START and console START.

READ CHECK or VALIDITY CHECK - Last card was incorrectly read due to invalid or off punching or read station jam. Last card in stacker (if no jam) and following card (run out by NPRO after hopper cards are removed) must be re-read. After appropriate correction, place these two cards at the front of the cards in the hopper, push then re-establish communications as previously described on pages 5 and 6. Incomplete input or output jobs are handled as described for job restart in the following two sections.

If even line restart fails to establish successful operation, you probably have a hardware and/or software problem which must be analyzed by your installation's systems personnel and IBM Customer Engineers.

Most problems you encounter will result in lights appearing on the 2770 console or the I/O devices themselves. Some of these lights are not error lights. These are DATA SET READY, CARRIER OFF, DATA IN BUFFER, LINE MODE, PROCEED, BID, SELN REQD, TRNSPCY, MANUAL ANSWER, and any of the I/O device lights when on steady. Any I/O device light which is blinking indicates that the device is not ready. Other lights provide clues to the difficulty and will be discussed in the following two sections.

ERROR RECOVERY WHEN TRANSMITTING

If job restart is required while transmitting, the OS job which is only partially read into the 2770 must be re-read from the beginning. You should ask the central computer operator to issue the HASP command:

$DELETE RMnn.RD1

to delete the partially read job. Press TERM RESET. Load the hopper beginning with the JOB card of the incompletely read job, push reader START and console START.

Any card reader trouble while transmitting is indicated by a blinking INPUT 2 or 3 light, whichever your card reader is attached to. The following lights on the card reader may further indicate the type of trouble.

FEED CHECK - The bottom card in the hopper failed to feed. Remove hopper cards. Push NPRO. Repair bottom hopper card if necessary and make sure the feed throat is clear. Reload cards. Push reader START and console START.

ATTENTION - Full stacker, empty hopper with EOF off, and cover open are possible causes. Correct, push reader START and console START.

READ CHECK or VALIDITY CHECK - Last card was incorrectly read due to invalid or off punching or read station jam. Last card in stacker (if no jam) and following card (run out by NPRO after hopper cards are removed) must be re-read. After appropriate correction, place these two cards at the front of the cards in the hopper, push reader START and console START. If a jam is so severe that the order of cards or the last card read is not clear, do a job restart.

HASP retries all transmission line errors automatically until transmission is successful, however, certain console lights may indicate necessary action on your part as follows.

TERMINAL ADDRESS - HASP is trying to send output while you are trying to start an input function. Continue input procedure (e.g., typing) until you have turned on the BID light. Then press CHECK RESET and wait for input to begin. Press CHECK RESET if TERMINAL ADDRESS comes on again. If you are not able to initiate the input function, you may have interrupted an incomplete output function. You must make your output devices ready to accept the output and wait until the next output job ending to again attempt transmission.

BID RETRY - HASP has failed to give permission to transmit. Press CHECK RESET to cause the 2770 to try again. If this fails, ask the central operator if he has temporarily prevented you from submitting jobs. If not, a line restart procedure must be done.

INPUT CHECK, BUFFER CHECK, TRNSPCY CHECK - With these serious errors you must always do a job restart. Make sure that you have turned on the TRANSPCY switch if the job contains OS object decks or other cards requiring transparent transmission.

RECORD CHECK or LINE CHECK - These lights may come on while HASP is attempting re-transmissions for line errors and will go out if recovery is successful. If they stay on and transmission does not proceed, you must do a job restart.

ERROR RECOVERY WHEN RECEIVING

If job restart is required while receiving, you must cause HASP to begin printing or punching the current partially completed job from its beginning. You should ask the central computer operator to issue the HASP command:

$RESTART RMnn.PR1 or $RESTART RMnn.PU1

to cause the restart. Make your output devices ready and press CHECK RESET in the normal manner. Discard the partially completed output beginning with the last previous separator page or separator card. For printing only you may ask the central operator to issue the HASP command:

$BACKSPACE RMnn.PR1

instead. Only the few duplicated pages should be discarded in this case. Do not press TERM RESET when doing a job restart while receiving. If TERM RESET is required to clear a check condition, a line restart must be done.

Output device trouble is indicated by blinking OUTPUT PRINTER or OUTPUT 2 lights and lights on the devices as follows.

CARRIAGE CHECK - The printer carriage, forms, or carriage tape are not ready or jammed. Correct the condition, press console CHECK RESET and START.

PRINT CHECK - The printer had a parity error. Press console CHECK RESET and START. Failure to recover indicates hardware trouble.

After any of the above printer recoveries, duplicate lines may be printed because HASP's recovery programming is designed to prevent loss of data at all costs. For most applications, these duplicate lines are obvious and may simply be crossed out or ignored. For more sensitive applications, you may use the back-space procedure described previously, which will make it easier to discard duplicate output at page or document boundaries.

CHECK light or any card punch not ready condition - Hopper empty, stacker full, or jams are possible causes. Set the keyboard switch to KEY PCH. Remove all cards from the stacker or eject station just below the stacker if any. Discard all removed cards after the last one with a column 81 punch. Clear the entire card feed path. With blank cards in the hopper, press the FEED key twice and the RELEASE key once, then set the switch to AUTO PCH. Press console CHECK RESET and START. The first card through the feed after recovery will be blank and should be discarded.

As with printing, there is a high probability of duplicate output following the punch error recovery described above. If duplicate punched output occurs, a whole 2770 internal buffer full of cards will be duplicated. The first full buffer punched after recovery consists of the cards coming into the stacker up to and including the first one with a column 81 punch. These cards (may be as few as one) should be compared with the same number of cards from the bottom of those removed from the stacker. If each card is an exact duplicate, you should discard the second group. If the application is such that a duplicated group of cards could occur as part of the intended punched output, a job restart must be done and all of the partially completed job's punched output must be discarded.

Certain console lights may require your attention while receiving, as follows.

TERMINAL ADDRESS - HASP is trying to send output but your 2770 is not ready. Make sure your switch setup is correct, ready all output devices, press console CHECK RESET.

OVERRUN - This usually indicates that features on your 2770 were not specified correctly at the central computer or that you have signed-on using the wrong remote number. You may have submitted jobs previously using this wrong number which will need to be re-routed to your correct number. You may have received output which is not yours. Ask the central operator to help you correct this confusion and do a line restart so that you can sign-on using the correct number.

BUFFER CHECK - This serious hardware error will always require you to do a line restart.

LINE CHECK - HASP is attempting re-transmissions. If they are successful, the light will go out. If the light stays on and printing or punching does not continue within a short time, you must do a job restart.

Depending on the central HASP System at your installation, actions during printer only error recovery may be somewhat different than described above. If this altered mode of printer operation is applicable to your 2770, when you make the printer ready after any of the above stops the job which was printing will be "suspended", a message and terminal separator line(s) will be printed, and the job will be re-queued in the print queue for your terminal. You may cause this "suspend" action yourself by pressing STOP while printing, then readying the printer.

Actions after the printer "suspend" depend on the state of your terminal and the output queues. You may start transmission as described on page 9 and following or you may wait for more output. Print jobs of higher priority than the suspended job or any punch jobs will be received before the suspended job. When the suspended job resumes printing, it will do so at approximately 1 page prior to the page of interruption.

CENTRAL COMPUTER CONTROL

This section describes the control cards you may use to sign on, send a message to the central computer operator, change the destination of printed and/or punched output, and force the priority of a job.

```
1               16          25
/*SIGNON        REMOTEnn    password
```

This is the sign-on card. The number nn is a one or two digit decimal number whose purpose is to correlate this remote device with information about it in the central computer. Leave the password field blank unless you are required to give a password.

```
1
/*SIGNOFF
```

You may use the sign-off card after the last job you read in. If you use this card, the telephone circuit will disconnect after about 30 seconds.

```
1               16
/*MESSAGE       message
```

When you read in this card, the contents of columns 16-71 will immediately be printed on the central computer operator's console. You may place this card anywhere within a job; it will be deleted before the job is processed.

The typed message will automatically have the job number appended to it if it is found within a job. If it is found outside a job, the remote terminal ID will be appended.

```
1          10       16
/*ROUTE    PUNCH    LOCAL
```

This card causes punched output for the job within which it was found to be punched at the central computer instead of at the remote terminal.

```
1          10       16
/*ROUTE    PRINT    LOCAL
```

This card does the same thing for printed output that the above card does for punched output. You may use both cards; a good place to put most of the cards described here is right after the //JOB card.

On either of these ROUTE cards, you may use REMOTEnn, PRINTERn, or PUNCHn in place of LOCAL, beginning in column 16. These alternate forms cause the printed or punch output for the job to go to a remote other than yours, or to a specific printer or punch at the central computer rather than any printer or punch at the central computer.

```
1                   16
/*PRIORITY          nn
```

If you use this card, it must immediately precede the //JOB card. The number nn is some one or two digit number between 0 and 15, inclusive. It specifies the urgency with which the job should be processed relative to other jobs submitted from the same remote terminal.

Depending on features of the central HASP System at your installation, you may use a subset of central HASP operator commands, submitted on cards as follows.

```
1
/*command
```

In place of "command", you should punch any of the commands listed in Table 1.1.3 of the central HASP Operator's Guide which are valid from a remote location. For example, "$DQ,4" punched following the "/*" causes a display of the number of punched jobs in various queues at the central site which are routed to the terminal REMOTE4.

A group of one or more command cards may be transmitted alone or may be placed in front of a group of jobs being transmitted. Command cards may also be transmitted from the keyboard, using lower case letters if desired.

Responses to commands from HASP are printed on the printer, after the paper is positioned at the top of a new page. Such responses are always received first after a transmission is completed, before any job's printed or punched output is received. Certain spontaneous messages (i.e. not responses) are also received. They are:

> messages acknowledging each job transmitted by your terminal and messages from other operators in the system to you by use of the "$DM" command.

You should read the HASP Operator's Guide to learn about the various commands you may use (Table 1.1.3), what their effects are, and how they should be constructed. Also, output device control and special forms processing are discussed in Section 7 of that Guide. However, certain properties of terminals like your 2770 require more explanation of these two topics.

Certain commands which control output devices ($B, $C, $E, $F, $I, $N, $Z) actually refer to a job currently in active processing on that device which is to be backspaced, restarted, etc. When you submit commands from your 2770, no output devices are active, therefore these commands have no effect. This is true even after you "suspend" a print job as previously described on page 20. The "suspend" is functionally equivalent to $I, which includes the function of $B. To use the other commands, you must ask the central operator to enter them.

The $S, $P, and $T device commands are effective when submitted from your terminal. Furthermore, the $C command is effective when referring to a job rather than a device. The $H, $A and $R commands may also be used effectively.

Special forms for printed or punched output can effectively be controlled from your terminal without central operator assistance, if all jobs submitted from (or routed to) your terminal follow certain conventions in requesting special forms. Programmers should be required to use only special routing output classes (J and K normally) with requests for special forms by data set. Special print forms for an entire job may be requested in the HASP accounting field of the JOB card. In no case should special forms be requested when using the ordinary output classes (A and B normally) as this will cause the system itself to request mounting of special forms at a time when you, as 2770 operator, are unable to enter the $S command to continue.

Assuming the above conventions, you should periodically submit the $DF command to determine if special forms jobs are queued for output on your terminal. If so, you should select the type of forms from those queued which you desire to process first on each output device, mount that forms, enter a command "$Tdevice,F=forms#" for each device, and wait for printing and/or punching to occur. When jobs stop processing on a device, you should resubmit the $DF and change to a new forms if indicated. The parameter "F=RESET" should be used to return a device to ordinary output processing. The "F=AUTO" parameter should not be used. You may want to use the $P and $S device commands, prior to and after the $T command respectively, to prevent HASP from attempting to send an output job while you are changing forms.

11.9 HASP REMOTE TERMINAL (SYSTEM/3) OPERATOR'S GUIDE

The following section contains detailed instructions for operating the IBM System/3 as a HASP MULTI-LEAVING, remote workstation. This manual is intended as a removable section for use at the remote location.

H A S P

(The remainder of this page intentionally left blank.)

H A S P

REMOTE TERMINAL PROCESSOR

FOR

MULTI-LEAVING BINARY SYNCHRONOUS

COMMUNICATIONS

SYSTEM/3 OPERATOR'S GUIDE

TABLE OF CONTENTS

1.0 INTRODUCTION

Remote Job Entry means the submission of jobs to an operating system from a terminal that is "remote" from the central computer. Ordinarily, job submission occurs from a card reader that is at most a matter of feet from the central computer, but a remote terminal may be hundreds of miles away.

The terminal communicates with the central computer over telephone lines or by similar means. If the telephone lines are permanently connected between the terminal and the central computer, they are called "point-to-point non-switched". If the lines are not permanently connected, they are called "point-to-point switched", and the remote terminal operator must dial the telephone number of the central computer, using the remote terminal's data set telephone, to connect the terminal with the computer.

HASP MULTI-LEAVING is a teleprocessing philosophy which allows the full use of all resources of the remote computer and of the communication line. A special, stand-alone terminal program in the remote computer establishes and maintains communication with HASP in the central computer. It compresses and blocks (for most efficient line usage) and transmits to HASP the card images of Operating System jobs. It receives from HASP, deblocks, and decompresses the printed and punched output of jobs. It performs similar functions for HASP operator commands and their responses, and for HASP messages to the remote terminal. The terminal program has the capability of operating all supported devices simultaneously.

The HASP System/3 Remote Terminal Processor program is a member of the family of HASP MULTI-LEAVING Terminal Programs. It is a stand-alone, self-loading, customized program which enables any System/3 with at least a Binary Synchronous Communications Adapter, a 5424 Multi-Function Card Unit, and a 5203 Printer to be used as a HASP MULTI-LEAVING Terminal.

This manual is the operating guide for HASP Remote Job Entry from the System/3. It contains operating procedures, error recovery procedures, and specifications for certain optional HASP Remote Job Entry and HASP-System/3 control cards. Since each System/3 Remote Terminal Processor is custom-generated, not all of the features described here may be in a particular System/3 Remote Terminal Processor.

The HASP System/3 Remote Terminal Processor supports most devices which can be attached to the System/3. Certain devices must be present:

a 5424 Multi-Function Card Unit
a 5203 Printer, with any features
a Binary Synchronous Communication Adapter
  with EBCDIC code and point-to-point
  network attachment.

The following devices need not be present, but will be supported if they are present and specified at the time of generation of the System/3 program:

a 5471 Printer-Keyboard, as an operator's
  input/output console
a 5475 Data Entry Keyboard, as an operator's
  input console
a 1442 Card Reader-Punch, an RPQ device,
  as an 80-column card reader/punch.

2.0 OPERATING PROCEDURES

This section of the HASP System/3 Remote Terminal Operator's Guide describes normal operating procedures for the System/3 as a remote job entry terminal. Operation generally consists of:

loading the Remote Terminal program
signing on
operating the various System/3 devices to
send jobs and receive their output
signing off.

Although this program does not operate under the IBM System/3 Card System, this guide refers to the IBM System/3 Card System Operator's Guide (Order Number GC21-7513) for extended information on some phases of operation. You should have a copy of the Card System Operator's Guide nearby for reference.

2.1 INITIATION OF A REMOTE JOB STREAM PROCESSING SESSION

To start a remote job entry session, you must accomplish three things: loading the HASP/Remote Terminal Processor (HASP/RTPSYS3) program deck, establishing a connection between the System/3 and the central computer, and signing on.

The HASP/RTPSYS3 program deck is a deck of System/3 cards about an inch thick. The cards are sequentially-numbered in columns 62-64, starting with card number 001, which is labeled "FIRST CARD" on its second print line, and ending with a card similarly labeled "LAST CARD". At least one control card should appear at the end of the program deck; it must be either an EOR card or a /*SIGNON card. Its function is to terminate the program load deck and enable the BSCA, and optionally to override the default /*SIGNON card, which is included in the RTPSYS3 assembly. Cards may appear before this card and after the card labeled "LAST CARD"; they have been provided by your installation systems programmer and should be left as they are.

To load the HASP/RTPSYS3 program deck, place it in the primary (rightmost) card hopper of the MFCU, make sure the last card is an EOR or /*SIGNON card, make the hopper ready, and push the Program Load Key on the console. Make the printer ready. For disk systems, the program load selector knob must point to "MFCU".

Midway through the program deck, the MFCU will stop reading and the printer will start printing the HASP Environmental recording and Editing Program (HEREP), a standard feature of RTPSYS3. The information printed is the contents of certain error counters; these counters contain a record of the unit checks which occurred during the last remote terminal session. If the counters are destroyed, one line will be printed:

HEREP COUNTERS HAVE BEEN ALTERED.

In any case, program loading will automatically resume after printing is complete.

Program loading has completed satisfactorily if when cards stop reading the console indicator "DT TERM READY" is on and the primary hopper is empty (or the first card in the primary hopper is not EOR or /*SIGNON; jobs or blank cards may be stacked behind the program deck). If "DT TERM READY" is not on, the last card of the program deck was not EOR or /*SIGNON or a card read error occurred. To correct a card read error, follow the procedure under halt code F3 in the IBM System/3 Card System Operator's Guide, make the primary hopper ready, and depress the START key (on dual-programming systems, the Program Level One Halt Reset Key) if halt code F3 is displayed.

If "DT TERM READY" is lit and the primary hopper contains an EOR or /*SIGNON card, remove the cards from the primary hopper and push STOP and then NPRO on the MFCU. The card that was stacked when you pushed NPRO is either an EOR or a /*SIGNON card. You should reload the program deck, making sure that it ends with either the correct /*SIGNON card or a single EOR card. (See section 4 for descriptions of these cards.)

Step 2 of initiating a remote session is establishing a connection between your System/3 and the central computer. The operator at the central computer should already have issued the HASP command "$START LNEnn" where LNEnn is the communication line to which your System/3 is permanently connected (point-to-point non-switched) or corresponds to the telephone number you will dial (point-to-point switched).

If your communication line is non-switched, make sure that any controls on its data set are in the "DATA" position. The System/3 will automatically establish communication with the central computer.

If your communication line is switched, pick up the data set's telephone handset and depress the data set's "TALK" button. Dial the telephone number you have been given and listen for the ring. When the ring is answered (automatically by the central computer) you will hear a high-pitched tone, followed by silence. Depress the data set's "DATA" key and hang up the handset. The System/3 will initiate communications with HASP and will automatically send it the /*SIGNON card. As the /*SIGNON card is being sent, the message

COMMUNICATION ESTABLISHED

will print on the 5471 Printer-Keyboard and on the 5203 Printer (if the 5203 Printer is ready).

If your System/3 has the Auto-Call feature and your /*SIGNON card (or the default /*SIGNON card, if not over-ridden) specifies a telephone number, leave the data set in "AUTO". The System/3 will automatically dial the required telephone number. When the number answers, the System/3 will automatically sign on.

If your call is not answered, or if the System/3 halts with halt code CA (call aborted) while trying to auto-call, the trouble is most likely that you dialed or specified on the /*SIGNON card an incorrect telephone number, or that the central operator did not start the correct line.

An auto-call halt CA can occur if the called number is busy. Depress the console start (or Program Level One Halt Reset) key to re-dial, or re-dial manually.

2.2 REMOTE JOB STREAM PROCESSING

During remote job stream processing, you are concerned with operating the unit record devices to submit jobs to the central computer and receive their printed and punched output. Each job goes through four phases - reading, execution, printing, and punching.

READING

You place into a card hopper (either 5424 or 1442 card reader) a stack of one or more jobs, and make the card hopper ready. The system reads the first card, finds it to be non-blank, and requests from HASP permission to start sending a job stream. When the system receives permission from HASP, it continues reading cards and sending them to HASP.

If you are reading from the 5424, you may use either card hopper to read from. The last card of your stack of jobs must be a /*EOF card (the characters /*EOF punched into columns 1-5); this card instructs the system to send to HASP an end-of-job-stream indicator, and to make the card hopper dormant.

If you are reading from the 1442, you end the job stream by pressing START when the hopper is empty. No special considerations apply to preparing or reading 80-column cards.

Each job you submit to HASP should be in the format of standard OS JCL. That is, it should consist of one JOB card followed by one or more EXEC and DD cards, and possibly by input stream data sets.

EXECUTION

When HASP receives the last card of a job from the System/3, it queues the job for OS execution. In due time, OS completes the job and HASP queues its printed output for transmission to the remote terminal from which it came. (However, the $ROUTE operator command or the /*ROUTE control card may be used to change the destination of printed or punched output, or both.) The execution process happens automatically, and you as an operator are not normally concerned with it.

PRINTING

You need only press START on the printer to allow print to occur; once a job has completed execution, printing starts automatically. The normal JCL specification for printed output is SYSOUT=A.

Some print data sets may require special forms; the programmer specified a 1- to 4-digit forms number on his DD card (e.g., SYSOUT=(A,,1234) is the specification for forms type 1234). When special forms are to be mounted, you will receive the message

```
LOAD TYPE mmmm FORMS IN RMnn.PR1
```

either on the 5203 or on the 5471. Mount the forms and type the command

```
$S RMnn.PR1
```

where nn is the same as in the LOAD message, or put into an available hopper the two cards

```
/*$S RMnn.PR1
/*EOF.
```

When a job's printed output is complete, HASP queues that job's punched output (if any) for processing. Though a job may not have punched output, it will always have printed output.

PUNCHING

You need only load an available hopper with blank cards and make it ready. Once a job has completed printing, punching starts automatically. The normal specification for punched output is SYSOUT=B.

Some punch data sets may require special forms; the programmer specified a 1- to 4-digit card forms number on his DD card (e.g., SYSOUT=(B,,9876) is the specification for forms type 9876). When special cards are to be loaded, you will receive the message

```
LOAD TYPE mmmm FORMS IN RMnn.PUn
```

either on the 5203 or on the 5471. Run out the card path, load cards of the type indicated, and type the command

```
$S RMnn.PUn
```

where nn and n are the same as in the LOAD message, or put into an available hopper the two cards

```
/*$S RMnn.PUn
/*EOF.
```

NOTES ON THE 5424

1. Either hopper of the 5424 can be used as either a reader or a punch. When a previously-dormant 5424 hopper reads a non-blank card, it becomes a reader. It remains a reader until it reads a /*EOF card; then it goes dormant with the /*EOF card in the wait station.

2. When a previously-dormant 5424 hopper reads a blank card, it becomes a punch. It remains a punch until it has completed punching all jobs queued for it. If no jobs are queued for it, you may make the hopper dormant by removing the blank cards from it.

3. The 5424 can read cards much faster than it can punch cards; therefore, to increase card throughput, the system performs card reading preferentially over card punching. If you are using both hoppers, one as a reader and one as a punch, punching will tend to proceed intermittently.

4. Though the 5424 has two hoppers, it has only one card path. For reasons of error recovery, the system ensures the card path is empty before switching hoppers. Therefore, if you are using both hoppers as readers, or both as punches, the system will tend to process cards from one or the other of the hoppers rather than dividing its time evenly between them.

5. Each blank card to be punched is read before it is punched, to make sure it is blank. A card that is not blank is stacked in the read stacker for the hopper from which it came.

6. Stacker selection is as follows:

Condition	Stacker
Reading from Primary	1
Punching from Primary	2
Punching from Secondary	3
Reading from Secondary	4

7. When preparing 96-column cards for the job stream, either as JCL or as data, you should avoid punching column 81, since the system makes special use of this column. In any case, the system only transmits the contents of columns 1-80; columns 82-96 are completely ignored. If the RMTGEN parameter &S3OBJDK was set to 1, the system inspects column 81. If that column contains the character "1", the system assumes that the card contains a hexadecimal image of the first 40 bytes of an 80-column card. It reads the next card, checks for a "2" in column 81, combines the cards into an 80-column card image, and transmits it. No checks are made for validity of hexadecimal characters. If a "2"-card does not follow a "1"-card, the "1"-card is lost.

8. Programmers should be aware of certain punching restrictions on the 5424. For all systems, if column 1 is X'6A' (12-11 punch on an 80-column card) the system recognizes a HASP

job separator card, extracts the job number to punch a System/3 job separator card, and ignores the rest of the card. If during RMTGEN the value of &S3OBJDK was specified as 1, then if column 1 is X'02' (12-2-9 punch on an 80-column card) the system recognizes a card image of an OS object deck and punches two 96-column cards with a hexadecimal representation of the card; see note 7 above. If during RMTGEN the value &S396COL was specified as 1, then if column 73 is X'80' (12-0-1-8 punch on an 80-column card) the system recognizes the left 48 columns (if column 80 is odd) or the right 48 columns (if column 80 is even) of a 96-column card; in this way all 96 columns of a System/3 card can be punched. This feature is used to create the System/3 Remote Terminal Program Deck, which is punched in System/3 load mode.

NOTES ON THE 1442

1. When a previously-dormant 1442 reads a nonblank card, it becomes a reader. It remains a reader until you press the START button after the hopper becomes empty (or until it reads a /*EOF card); then it goes dormant. If it became dormant because you pressed the START button with no cards in the hopper, it also runs out the cards in its feed path.

2. When a previously-dormant 1442 reads a blank card, it becomes a punch. It remains a punch until it has completed punching all jobs queued for it. Only after all queued jobs have been punched can you safely remove cards from the 1442 hopper; with the hopper empty and no more punching to do, the 1442 goes dormant. You should press the NPRO button to stack into the right stacker the two blank cards remaining in the card feed path.

3. All cards processed by the system are stacked into the left stacker.

NOTES ON THE 5203

1. At program load time, the system checks indicators of the 5203 to determine which print chain is mounted. If the indicators show a 48-character-set chain, the system assumes character arrangement LC; otherwise it assumes character arrangement PN.

2. At program-load time, the system sets number of print lines per page to 66 (this may be different for your installation). For dual-carriage printers, the system uses only the left carriage; you must not press the RIGHT CAR. RESTORE key.

3. At program-load time, the system sets line numbers for programmed page skipping. These are provided to simulate the carriage tape control normally encountered in OS. A skip to carriage channel 1 will result in a page eject; a skip to channel 12 will stop 5 lines from the bottom of the page; and a skip to any other channel will result in no paper movement. Carriage tape channels may, however, be defined differently for your installation.

2.3 TERMINATION OF A REMOTE JOB STREAM PROCESSING SESSION

When you are done using the System/3 as a Remote Job Entry terminal, put into an available hopper the two cards

```
/*SIGNOFF
/*EOF
```

and press START on the card reader.

The /*SIGNOFF card tells HASP to disconnect the communication line after it has finished sending the current print and punch streams to the System/3 and receiving the current job from the System/3. That is, HASP disconnects when all currently-operating functions are complete. If you sign off before HASP has started printing or punching some or all of your jobs, HASP will save the output for transmission to your terminal the next time you sign on with the same remote terminal identification.

Alternatively, either you or the central operator can tell HASP to route the printed or punched output of any or all jobs to the central site. See the /*ROUTE control card in Section 4 of this manual and the $ROUTE command in the HASP Operator's Manual.

When HASP finally disconnects the communication line, the System/3 Communication Adapter will get a time-out error every three seconds for about 20 seconds; then the DATA light on the data set telephone will go out. The System/3 may continue printing and punching for a short time. When the System/3 is dormant, push the STOP button on the console to stop the customer meter from running. Your RJE session is now ended.

2.4 COMMAND PROCESSING

If your System/3 includes a 5475 Data Entry Keyboard or a 5471 Printer-Keyboard, you use the keyboard to enter commands. Otherwise, you punch commands on cards and enter them through a reader, exactly as if they were jobs.

The only commands valid from a remote terminal are certain HASP commands. These commands are described in another section, the HASP Operator's Guide; you should have a copy nearby for reference.

ENTERING COMMANDS FROM THE 5471

To type a command to HASP, press the REQ key. If the system can immediately allow you to type a command, the PROCEED light will go on; otherwise the REQUEST PENDING light will go on. You may press the REQ key while you are typing a command, while the system is typing a message to you, or while the console is dormant.

When the PROCEED light comes on, start typing your command. If you make a mistake, press the CANCEL key and start typing again.

When you are done typing, press either the END key or the RETURN key; their functions are identical. Your command will be transmitted to HASP, where it will be executed (if valid) and repeated together with your remote terminal number on the central operator's console.

If you type a command of 120 characters, the system will automatically perform the END key function when you type the 120th character.

ENTERING COMMANDS FROM THE 5475

To type a command to HASP, merely start typing on the 5475 Data Entry Keyboard. The keyboard is always alive. After you have typed the first character, the column indicator will become active and display "02", the position of the character you will be typing next. If you make a mistake, depress the FLD ERASE key; the column indicator will display "01" and you may start typing again.

When you are done typing, depress the REL key to transmit the command to HASP. When the column indicators go dark, you may begin typing another command. If you type a command of 120 characters, the system will automatically perform the REL key function when you have typed the 120th character.

ENTERING COMMANDS FROM CARDS

To send a command to HASP from a card reader, you must first punch the command on a card. Starting in column 1, punch a slash, punch an asterisk, and then punch the command. Since all HASP commands start with a dollar sign, columns 1-3 will read "/*$". Then put one or more command cards,followed by a /*EOF card, into an available card hopper, and push START. Your commands will be transmitted to HASP, where they will be executed (if valid) and repeated together with your remote terminal number and reader number on the central operator's console.

3.0 ERROR RECOVERY PROCEDURES

Two general classes of errors are defined in the System/3 Remote Terminal Processor: Communication Errors and Unit Record Errors. For either type of error, the system generates an 8-character error message. If your system has a 5471 console, error messages will be typed on it as errors occur. If your system does not have a 5471 console, error messages may or may not be printed on the 5203 printer, depending upon how your Remote Terminal Program was generated. The format of all error messages is

ttxxxxuu

where tt is the message type, xxxx is additional error information, and uu is the device upon which the error occurred. The correspondence between uu and device is as follows:

Device	uu
BSCA	00
1442	05
5203	0E
5424	0F

3.1 COMMUNICATION ADAPTER ERRORS

The communication technique used by HASP is such that there should be no BSCA errors during a processing session. Therefore, any BSCA error that occurs while you are signed on is an unusual condition, resulting from system or communication facility malfunction or operational conditions. For all BSCA errors, the BSCA processor within the System/3 Remote Terminal Processor will automatically take corrective action; therefore, you should regard all BSCA error messages as only informational messages.

The following BSCA messages can be produced:

01RREE00

MEANING - A block sequence check occurred - a transmission block was duplicated or lost. RR is the received block number, and EE is the expected block number. Both RR and EE will range from X'80' to X'8F'.

ACTION - Duplicate transmission blocks will be ignored. Lost transmission blocks will cause automatic job restart.

02000000

MEANING - The System/3 received a negative acknowledgment (NAK) from HASP.

ACTION - The transmission block which was negatively acknowledged will be re-transmitted.

03RRRR00

MEANING - The transmission block received by the System/3 had an unrecognizable starting or ending sequence. The starting sequence is RRRR; if it is correct, the ending sequence is in error.

ACTION - The System/3 will send a NAK to HASP, which will then re-transmit the block.

05SSSS00

MEANING - The System/3 BSCA had a unit check. The BSCA status indicators are SSSS.

ACTION - The appropriate action will automatically be taken to continue or restore communication. Two of the most common examples of BSCA unit check are 05800000-timeout error, and 05840000-timeout with abortive disconnect. Read Section 2.3 of this manual to find out when these errors can occur normally.

3.2 UNIT RECORD ERROR PROCEDURES

Unit record error messages are provided for errors on the 1442 Card Reader/Punch (an RPQ device), the 5424 Multi-Function Card Unit, and the 5203 Printer.

5424 MFCU

The only MFCU error message is 05SSSSOF, where SSSS are the MFCU status indicators. In all cases, operator intervention is required. You should check the MFCU control panel to determine which card hopper the error message applies to. PRI means the rightmost (primary) hopper; SEC means the leftmost (secondary) hopper. The system will attempt to perform its previous operation again when you have cleared the error condition: if it was reading when an error occurred, it will try to read the same card again; or if it was punching, it will try to punch again. Therefore, if the hopper was punching, you should throw away the last card punched; if the hopper was reading, you should place the last card read in the hopper again, so the system can re-read it. First, however, lift the cards out of the indicated hopper and press the NPRO key to clear the error condition.

5203 Printer

The only 5203 error message is 05SSSSOE, where SSSS are the 5203 status indicators. If any error light is on at the 5203 control panel, correct the condition and press printer START. The system will automatically retry printing when an incrementer failure or print check occurred.

1442 Card Reader/Punch

The only 1442 error message is 05SSSS05, where SSSS are the status indicators. The system recovers from 1442 errors the same way it recovers from MFCU errors. You should perform the action indicated by the 1442 error lights; then throw away the last-punched card or place the last-read card back in the hopper and press START.

3.3 REMOTE TERMINAL RESTART

In the event of an untimely interruption of the remote terminal operation such as a machine, program communications, or environmental failure, you should notify appropriate maintenance personnel of the malfunction, save material which may be of use in determining the source of the failure, and with the aid of the central computer operator prepare for restarting the terminal as follows:

1. Notify the central computer operator of the failure and, if necessary,request his assistance in preparing for restart.

2. Determine the current job being transmitted to HASP. (The central operator has a record of the current job being submitted to HASP.) The job stream starting with the current job must be submitted to HASP after restart.

3. Determine the loss of data on the output devices and inform the central operator to BACKSPACE or RESTART the printer or punch as necessary. (The central computer's line should be made available for a subsequent session with the remote station or other stations within the system.)

4. When the remote terminal is available, perform the steps required for initiating a "Remote Job Processing Session."

4.0 SYSTEM CONTROL CARDS

You may use the same HASP control cards in submitting your job from a HASP Remote Terminal that you would use for local job submission. These cards, the /*PRIORITY, /*ROUTE, /*MESSAGE, and /*SETUP control cards, offer you a greater degree of control over jobs submitted to HASP.

By contrast, certain other control cards are fundamental to the operation of the System/3 Remote Terminal Processor: the /*EOF, /*SIGNON, EOR, and /*SIGNOFF cards.

/*EOF

The /*EOF control card consists of the characters "/*EOF" punched in columns 1-5. This control card must be the last card read by an MFCU hopper when the hopper is reading, whether jobs, commands, or just a /*SIGNOFF card is being read. This card may optionally be used on the 1442, but the recommended 1442 procedure is as stated in Section 2.2.

/*SIGNON

The /*SIGNON card consists of the characters "/*SIGNON" in columns 1-8, your remote terminal identification starting in column 16, an optional password field starting in column 25, and optional dialing information starting in column 34. You will only rarely be using this card, since a /*SIGNON card is already included in your HASP/RTPSYS3 deck.

The remote terminal identification field consists of the letters "REMOTE" followed by one or two decimal digits. If your remote number is less than ten, use (for example) REMOTE1 rather than REMOTE01.

The password field should not be used unless required by your installation systems programmer.

The dial field should be used only if your System/3 Binary Synchronous Communications Adapter has the Auto-Call feature and you want the telephone number dialed automatically. The word "DIAL" should start in column 34. It should be followed by at least one blank, and by an all-numeric telephone number of any length. No alphabetic characters, hyphens, or embedded blanks may appear in the telephone number.

See Section 2.1 for an explanation of the use of the /*SIGNON card.

EOR

The EOR control card consists of the characters "EOR" in card columns 2-4. It is used instead of the /*SIGNON card when the default /*SIGNON card, assembled into the HASP/RTPSYS3 deck, is not to be overridden. See Section 2.1 for an explanation of the use of the EOR card.

/*SIGNOFF

The /*SIGNOFF card consists of the characters "/*SIGNOFF" in columns 1-9. Its use is explained more fully in Section 2.3.

/*PRIORITY

The /*PRIORITY card consists of the characters "/*PRIORITY" punched in columns 1-10 and a decimal number from 1 to 15 punched starting in column 16. You use this control card when you want to assign to your job a specific priority relative to other jobs submitted from the same remote terminal. The placement of the /*PRIORITY card is immediately before the OS JOB card.

If you do not use the /*PRIORITY card, HASP will automatically set your job's priority to a number calculated from your JOB card's estimated execution time and estimated print lines.

/*ROUTE

The /*ROUTE control card offers a convenient way to redirect the printed and/or punched output of jobs submitted from your terminal. You may place the /*ROUTE card anywhere within a job; it is effective for all printed or punched output of that job.

The card consists of three fields: starting in column 1, the characters "/*ROUTE"; starting in column 10, either the word PRINT or the word PUNCH, depending upon which output type you are rerouting; and, starting in column 16, the destination of the output, expressed as either LOCAL, REMOTEn, PRINTERn, or PUNCHn. LOCAL routing routes the selected output to any printer or punch at the central site, whichever device is appropriate. REMOTEn routing routes the selected output to the appropriate device type at the named remote terminal. If allowed by your installation, PRINTERn and PUNCHn may be used in place of LOCAL to route your output to a selected local printer or punch.

You may reroute both your printed output and your punched output; use two /*ROUTE cards to do this.

/*MESSAGE

The /*MESSAGE control card requests HASP to give the message punched in its columns 16-71 to the central operator. The characters "/*MESSAGE" start in column 1. The operator receives the message just after the /*MESSAGE card is read. If the /*MESSAGE card appears within a job, the job's number will be appended to it. Otherwise, your remote number and reader number will be appended.

/*SETUP

The /*SETUP card consists of the characters "/*SETUP" punched in columns 1-7 and, starting in column 16, a free-form list of volume serial numbers for volumes your job requires. The list must end by column 71. This card causes your job to be placed in hold status and a message to be printed to the central computer operator listing the volumes your job requires. When the operator has located the volumes, he will issue the $RELEASE command for your job.

To continue your list of volume serial numbers, use one or more /*MESSAGE control cards after the /*SETUP control card.

5.0 THE STARTER SYSTEM

The System/3 Remote Terminal Processor Starter System is a deck of 96-column cards distributed as a part of the HASP System. You should use the Starter System to punch at the System/3 the punched output of the RMTGEN process.

To use the Starter System deck, you must add two cards at the end of the deck. The first card describes the size of the HASP MULTI-LEAVING buffers and is in exactly the same format as for the HASPGEN parameter &MLBFSIZ=. For example, if the correct size were 400 bytes, you would punch "&MLBFSIZ=400" starting in column 1.

The second card to be added is a /*SIGNON card. You punch this card according to its description in Section 4.0 of this manual.

The Starter System deck will work on any System/3 which supports HASP MULTI-LEAVING Remote Job Entry. The deck it punches will be your customized System/3 Remote Terminal Processor, as defined by your installation systems programmer. The Starter System does not include support for the 5475, 5471, or 1442.

(The remainder of this page intentionally left blank.)

12.0 APPENDICES

The following appendices are included as additional information pertaining to the current status of the HASP System.

12.1 REFERENCE LISTING OF HASPJCL

This section contains a reference listing of the source module HASPJCL which is printed and punched during a complete HASPGEN, as described in Section 10.1.4. The module contains four sample jobs for use when installing HASP, as described in Section 10.2.2.

12.1.1 Sample Job HASPSVC

```
//HASPSVC  JOB (0000,0000),'INSTALL HASP SVC',MSGLEVEL=1
//SCRATCH EXEC PGM=IEHPROGM
//SYSPRINT  DD SYSOUT=A
//SYSRES    DD UNIT=SYSDA,VOLUME=SER=YYYYYY,DISP=OLD
//SYSIN     DD *
        RENAME DSNAME=SYS1.OLDNUC,NEWNAME=SYS1.NEWNUC,VOL=SYSDA=YYYYYY
      UNCATLG DSNAME=SYS1.NEWNUC
      SCRATCH DSNAME=SYS1.NEWNUC,VOL=SYSDA=YYYYYY,PURGE
/*
//LKED     EXEC PGM=IEWL,PARM='XREF,LET,LIST,NCAL,SCTR',REGION=96K
//HASPOBJ   DD DSNAME=SYS1.HASPOBJ,DISP=SHR
//NUCLEUS   DD DSNAME=SYS1.NUCLEUS,DISP=SHR
//SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(10,5))
//SYSLMOD   DD DSNAME=SYS1.NEWNUC,UNIT=2314,VOLUME=SER=YYYYYY,          C
//             DISP=(NEW,CATLG),LABEL=EXPDT=99366,                     C
//             SPACE=(TRK,(40,,2),,CONTIG)
//SYSPRINT  DD SYSOUT=A
//SYSLIN    DD *
        INSERT IEAANIP0
        INSERT IEAAIH00                   USE ONLY FOR MFT
        INSERT IEAQFX00                   USE ONLY FOR MVT
       INCLUDE HASPOBJ(HASPSVC)
       INCLUDE NUCLEUS(IEANUC01)
          NAME IEANUC01(R)
/*
//RENAME   EXEC PGM=IEHPROGM
//SYSPRINT  DD SYSOUT=A
//SYSRES    DD UNIT=SYSDA,VOLUME=SER=YYYYYY,DISP=OLD
//SYSIN     DD *
        RENAME DSNAME=SYS1.NUCLEUS,NEWNAME=SYS1.OLDNUC,VOL=SYSDA=YYYYYY
        RENAME DSNAME=SYS1.NEWNUC,NEWNAME=SYS1.NUCLEUS,VOL=SYSDA=YYYYYY
/*
```

12.1.2 Sample Job HASPROCS

```
//HASPROCS JOB (0000,0000),'INSTALL HASP PROCS',MSGLEVEL=1
//PROCS    EXEC PGM=IEBUPDTE,PARM=NEW
//SYSPRINT  DD SYSOUT=A
//SYSUT2    DD DSNAME=SYS1.PROCLIB,DISP=SHR,DCB=LRECL=80
//SYSIN     DD DATA
./         ADD NAME=HASP,LIST=ALL
./      NUMBER NEW1=20000,INCR=20000
//HASP     PROC JOB=STRTHASP
//IEFPROC EXEC PGM=IEFIRC,REGION=50K,                                  C
//             PARM='01499900100124905100SYSDA    E000'                C
//                   BPPTTTOOOMMMIIICCCRLSSSSSSSSAAAA
//IEFRDER   DD DSNAME=SYS1.PROCLIB(&JOB),DISP=SHR,                     C
//             DCB=(BUFNO=1,RECFM=FB,LRECL=80,BLKSIZE=80)
//IEFPDSI   DD DSNAME=SYS1.PROCLIB,DISP=SHR
//IEFDATA   DD DUMMY
./         ADD NAME=STRTHASP,LIST=ALL
./      NUMBER NEW1=20000,INCR=20000
//HASP     JOB 'HASP-II','INVOKE HASP SYSTEM',CLASS=H,MSGCLASS=H
//SYSTEM  EXEC PGM=HASP,TIME=1440,REGION=51K
//OLAYLIB   DD DSNAME=SYS1.HASPOLIB,DISP=SHR
// S INIT.P0,,,H
//
./         ADD NAME=HOSRDR,LIST=ALL          ASB RDR FOR MVT ONLY
./      NUMBER NEW1=20000,INCR=20000
//IEFPROC EXEC PGM=IEFVMA,REGION=16K,                                  C
// PARM='00103000100125205011SPOOL     E00001,1011208204E000SYSDA    00'
//*      BPPTTTOOOMMMIIICCCRLSSSSSSSSAAAAEF,EJJAARRATABAAADDDDDDDDGK
//IEFRDER   DD UNIT=00C,DISP=OLD,                                      C
//             DCB=(RECFM=F,LRECL=80,BLKSIZE=80,BUFNO=1)
//IEFPDSI   DD DSNAME=SYS1.PROCLIB,DISP=SHR
//IEFDATA   DD UNIT=SYSDA,VOLUME=REF=SYS1.LINKLIB,                     C
//             SPACE=(80,(200,200),RLSE,CONTIG),DISP=OLD,              C
//             DCB=(DSORG=PS,RECFM=FB,LRECL=80,BUFL=80,BLKSIZE=80)
./         ADD NAME=HOSRDR,LIST=ALL          STD RDR FOR MFT OR MVT
./      NUMBER NEW1=20000,INCR=20000
//IEFPROC EXEC PGM=IEFIRC,REGION=50K,                                  C
//             PARM='00103000100125205011SPOOL    '                    C
//                   BPPTTTOOOMMMIIICCCRLSSSSSSSS
//IEFRDER   DD UNIT=00C,DISP=OLD,                                      C
//             DCB=(RECFM=F,LRECL=80,BLKSIZE=80,BUFNO=1)
//IEFPDSI   DD DSNAME=SYS1.PROCLIB,DISP=SHR
//IEFDATA   DD UNIT=SYSDA,VOLUME=REF=SYS1.LINKLIB,                     C
//             SPACE=(80,(200,200),RLSE,CONTIG),DISP=OLD,              C
//             DCB=(DSORG=PS,RECFM=FB,LRECL=80,BUFL=80,BLKSIZE=80)
./         ADD NAME=HOSWTR,LIST=ALL          USED ONLY IF &WTRPART NE *
./      NUMBER NEW1=20000,INCR=20000
//IEFPROC EXEC PGM=IEFSD080,PARM='PA',REGION=12K
//IEFRDER   DD UNIT=1403,DSNAME=SYSOUT,DISP=(NEW,KEEP),                C
//             DCB=(RECFM=FM,LRECL=133,BLKSIZE=133,BUFL=133,BUFNO=1)
./       ENDUP
/*
```

12.1.3 Sample Job HASPHASP

```
//HASPHASP JOB (0000,0000),'INSTALL HASP PROGRAM',MSGLEVEL=1
//SCRATCH EXEC PGM=IEHPROGM
//SYSPRINT  DD SYSOUT=A
//OLAYLIB   DD UNIT=SYSDA,VOLUME=SER=ZZZZZZ,DISP=OLD
//SYSIN     DD *
       UNCATLG DSNAME=SYS1.HASPOLIB
       SCRATCH DSNAME=SYS1.HASPOLIB,VOL=SYSDA=ZZZZZZ,PURGE
/*
//OBLD    EXEC PGM=HASPOBLD
//STEPLIB   DD DSNAME=SYS1.HASPMOD,DISP=SHR
//SYSIN     DD *,DCB=BLKSIZE=80
/*
//SYSOBJ    DD DSNAME=SYS1.HASPOBJ(HASPNUC),DISP=SHR
//          DD DSNAME=SYS1.HASPOBJ(HASPRDR),DISP=SHR
//          DD DSNAME=SYS1.HASPOBJ(HASPXEQ),DISP=SHR
//          DD DSNAME=SYS1.HASPOBJ(HASPPRPU),DISP=SHR
//          DD DSNAME=SYS1.HASPOBJ(HASPACCT),DISP=SHR
//          DD DSNAME=SYS1.HASPOBJ(HASPMISC),DISP=SHR
//          DD DSNAME=SYS1.HASPOBJ(HASPCON),DISP=SHR
//          DD DSNAME=SYS1.HASPOBJ(HASPRTAM),DISP=SHR
//          DD DSNAME=SYS1.HASPOBJ(HASPCOMM),DISP=SHR
//          DD DSNAME=SYS1.HASPOBJ(HASPINIT),DISP=SHR
//SYSLIN    DD DSNAME=&&TEMP,UNIT=SYSSQ,DISP=(NEW,PASS),
//             SPACE=(400,(400,50)),DCB=BLKSIZE=400
//OLAYLIB   DD DSNAME=SYS1.HASPOLIB,UNIT=SYSDA,VOLUME=SER=ZZZZZZ,
//             DISP=(NEW,CATLG),LABEL=EXPDT=99366,
//             SPACE=(1024,50,,CONTIG)
//SYSPRINT  DD SYSOUT=A,DCB=BLKSIZE=121
//LKED    EXEC PGM=IEWL,PARM='LIST,XREF,NCAL',REGION=96K
//HASPOBJ   DD DSNAME=SYS1.HASPOBJ,DISP=SHR
//SYSUT1    DD DSNAME=SYS1.UT3,DISP=OLD
//SYSLMOD   DD DSNAME=SYS1.LINKLIB,DISP=OLD
//SYSPRINT  DD SYSOUT=A
//SYSLIN    DD DSNAME=&&TEMP,DISP=(SHR,PASS)
//          DD *
          NAME HASP(R)
       INCLUDE HASPOBJ(HASPBR1)
          NAME HASPBR1(R)
       INCLUDE HASPOBJ(HASPWTR)
          NAME HASPWTR(R)
/*
```

12.1.4 Sample Job HASPOOLS

```
//HASPOOLS JOB (0000,0000),'ALLOCATE SPOOL SPACE',MSGLEVEL=1
//SCRATCH EXEC PGM=IEHPROGM
//SYSPRINT  DD SYSOUT=A
//SPOOL1    DD UNIT=SYSDA,VOLUME=SER=SPOOL1,DISP=OLD
//SPOOL2    DD UNIT=SYSDA,VOLUME=SER=SPOOL2,DISP=OLD
//SYSIN     DD *
       SCRATCH VTOC,VOL=SYSDA=SPOOL1,PURGE
       SCRATCH VTOC,VOL=SYSDA=SPOOL2,PURGE
/*
//ALLOCAT EXEC PGM=IEFBR14
//SPOOL1    DD DSNAME=SYS1.HASPACE,VOLUME=SER=SPOOL1,
//             DISP=(NEW,KEEP),LABEL=EXPDT=99366,
//             UNIT=2314,SPACE=(ABSTR,(3998,2))
//SPOOL2    DD DSNAME=SYS1.HASPACE,VOLUME=SER=SPOOL2,
//             DISP=(NEW,KEEP),LABEL=EXPDT=99366,
//             UNIT=3330,SPACE=(ABSTR,(7674,2))
```

12.2 HASP STORAGE REQUIREMENTS

This section is provided to allow installations to compute the size of a HASP SYSTEM based upon the HASPGEN options selected. The formula given, when properly evaluated will indicate the size of the resident HASP load module. This value may then be used in computing the proper region or partition size for HASP.

In computing the region or partion size, allowances must be made for the various control blocks and work space required in the region/partition by the Operating System for the initiation and operation of HASP. Additional space may also be required for dynamic construction of additional HASP buffers (see &NUMBUF description in section 7). In all computations, the maximum degree of HASP overlay (&OLAYLEV=15) is assumed.

12.2.1 Additional Nucleus Storage Requirements

In addition to the storage required as a region or partition, HASP also requires certain fixed space in the Nucleus of the Operating System as follows:

- The space required for the pseudo device Unit Control Blocks required by HASP.
- System Queue Space required by the Operating System to initiate a job.
- Space for the HASP initialization SVC (136 bytes).

12.2.2 HASP Module Storage Requirements

The storage requirements of the primary HASP module are expressed by the following formula:

$$
\begin{aligned}
S_{HASP} = {} & 21{,}400 + S_1 + S_2 + S_3 + S_4 + S_5 + S_6 + S_7 + S_8 + S_9 + \\
& S_{10} + S_{11} + S_{12} + S_{13} + S_{14} + S_{15} + S_{16} + S_{17} + S_{18} + \\
& S_{19} + S_{20} + S_{21} + S_{22} + S_{23} + S_{24} + S_{25} + S_{26} + S_{27} + \\
& S_{28} \text{ bytes.}
\end{aligned}
$$

where the values of S_n are defined below.

To facilitate ease in computation and simplicity of equations, the following value should first be computed:

DAMAP = &NUMDA x ((&NUMTGV+7)/8)

The values of S_n can then be computed as follows:

S_1 = &NUMRDRS x (396+DAMAP)

S_2 = &NUMTPES x (396+DAMAP)

$$S_3 = \begin{cases} 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{if \&NUMINRS} = 0 \\ 364 + \text{\&NUMINRS} \times (496+\text{DAMAP}) \ldots\ldots\ldots\ldots \text{if \&NUMINRS} \neq 0 \end{cases}$$

S_4 = &NUMPRTS x (316 + 8x&NOPRCCW)

S_5 = &NUMPUNS x (372 + 8x&NOPUCCW)

$$S_6 = \begin{cases} 0 \ldots\ldots\ldots\ldots\ldots\ldots \text{if \&NUMCONS} = 0 \text{ and \&AUTORDR} = \text{NO} \\ 120 \ldots\ldots\ldots\ldots\ldots \text{if \&NUMCONS} = 0 \text{ and \&AUTORDR} = \text{YES} \\ \left.\begin{array}{l} 1264 + 164 \times \text{\&NUMCONS} + 16 \times \text{\&RQENUM} \\ \quad + 224 \ldots\ldots\ldots\ldots \text{if \&SIZ2260} \neq 0 \\ \quad + 60 \ldots\ldots\ldots\ldots \text{if \&AUTORDR} = \text{YES} \end{array}\right\} \text{if \&NUMCONS} \neq 0 \end{cases}$$

S_7 = &NUMDA x 58 + 2 x DAMAP

S_8 = &NUMBUF x (80+&BUFSIZE)

S_9 = &NUMOACE x 1112

S_{10} = &NUMWTOQ x 140

S_{11} = &MAXJOBS x 16

$$S_{12} = \begin{cases} 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{if \&JITSIZE} = 0 \\ 132 + \text{\&JITSIZE} \times \text{\&MAXJOBS} \ldots\ldots\ldots\ldots \text{if \&JITSIZE} \neq 0 \end{cases}$$

S_{13} = &MAXXEQS x 216

S_{14} = &MAXPART x (12+&MAXCLAS)

S_{15} = &NUMDDT x 37

$$S_{16} = \begin{cases} 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{if \&MONINTV} = 0 \\ 252 + 12 \times \text{\&MAXXEQS} \ldots\ldots\ldots\ldots\text{if \&MONINTV} \neq 0 \end{cases}$$

$$S_{17} = \begin{cases} 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{if \&PRIRATE} = 0 \\ 108 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{if \&PRIRATE} \neq 0 \end{cases}$$

$$S_{18} = \begin{cases} 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{if \&WTRPART} \neq * \\ 1592 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{if \&WTRPART} = * \end{cases}$$

$$S_{19} = \begin{cases} 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{if \&OSINOPT} = \text{NO} \\ 24 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{if \&OSINOPT} = \text{YES} \end{cases}$$

$$S_{20} = \begin{cases} 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{if \&XBATCHC} = \text{null} \\ 704 + 14 \times \text{\&MAXPART} \ldots\ldots\ldots\ldots\text{if \&XPATCHC} \neq \text{null} \end{cases}$$

$$S_{21} = \begin{cases} 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{if \&TIMEOPT} = 4 \\ 180 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{if \&TIMEOPT} \neq 4 \end{cases}$$

$$S_{22} = \begin{cases} 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{if \&PRTRANS} = \text{NO} \\ 292 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{if \&PRTRANS} = \text{YES} \end{cases}$$

$$S_{23} = \begin{cases} 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{if \&ACCTNG} = \text{NO} \\ 30 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{if \&ACCTNG} = \text{YES} \end{cases}$$

$$S_{24} = \begin{cases} 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{if \&DEBUG} = \text{NO} \\ 1660 + 2 \times \text{DAMAP} \ldots\ldots\ldots\ldots\text{if \&DEBUG} = \text{YES} \end{cases}$$

$$S_{25} = \begin{cases} 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{if \&TRACE} = 0 \\ 534 + 64 \times \text{\&TRACE} \ldots\ldots\ldots\ldots\text{if \&TRACE} \neq 0 \end{cases}$$

$$S_{26} = \begin{cases} 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{if \&OREPSIZ} = 0 \\ 72 + \text{\&OREPSIZ} \ldots\ldots\ldots\ldots\ldots\text{if \&OREPSIZ} \neq 0 \end{cases}$$

$$S_{27} = \begin{cases} 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{if \&DMPTAPE} = 000 \\ 560 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{if \&DMPTAPE} \neq 000 \end{cases}$$

$$S_{28} = \begin{cases} 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{if \&NUMLNES} = 0 \\ R_1 + R_2 + R_3 + R_4 + R_5 + R_6 + R_7 + R_8 \text{ if \&NUMLNES} \neq 0 \end{cases}$$

where:

R_1 = &NUMTPBF x (144+&TPBFSIZ)

R_2 = &NUMLNES x 104

R_3 = &NUMRJE x 132

R_4 = &NUMTPRD x (368+DAMAP)

R_5 = &NUMTPPR x 200

R_6 = &NUMTPPU x 168

$$R_7 = \begin{cases} 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{if \&SPOLMSG} = 0 \\ (\text{\&SPOLMSG}+7)/8 + \text{\&NUMRJE} \times 8 \ldots\ldots \text{if \&SPOLMSG} \neq 0 \end{cases}$$

R_8 = a value selected from the following table:

&STR1978	&STRCPU	&BSC2770 or &BSC2780	&BSCCPU	R_8
YES	NO	NO	NO	6,012
NO	YES	NO	NO	5,116
YES	YES	NO	NO	7,724
NO	NO	YES	NO	5,772
YES	NO	YES	NO	8,868
NO	YES	YES	NO	7,988
YES	YES	YES	NO	10,572
NO	NO	NO	YES	5,872
YES	NO	NO	YES	9,416
NO	YES	NO	YES	8,448
YES	YES	NO	YES	11,048
NO	NO	YES	YES	8,328
YES	NO	YES	YES	11,400
NO	YES	YES	YES	10,464
YES	YES	YES	YES	13,048

12.2.3 Example I -- Storage Requirements for a Small HASP

Consider a HASP package which has been HASPGENed to be used on a small machine with limited resources. The HASPGEN parameters might be set as follows:

&NUMDA = 1	&MAXXEQS = 2
&NUMTGV = 400	&MAXPART = 2
&NUMRDRS = 1	&MAXCLAS = 8
&NUMTPES = 0	&NUMDDT = 16
&NUMINRS = 0	&MONINTV = 0
&NUMPRTS = 1	&PRIRATE = 0
&NOPRCCW = 5	&WTRPART = *
&NUMPUNS = 1	&OSINOPT = NO
&NOPUCCW = 5	&XBATCHC = null
&NUMCONS = 0	&TIMEOPT = 4
&AUTORDR = YES	&PRTRANS = NO
&NUMBUF = 11	&ACCTNG = NO
&BUFSIZE = 504	&DEBUG = NO
&NUMOACE = 1	&TRACE = 0
&NUMWTOQ = 5	&OREPSIZ = 0
&MAXJOBS = 50	&DMPTAPE = 000
&JITSIZE = 0	&NUMLNES = 0

The storage requirements would be computed as follows:

DAMAP = 1 x 50 = 50

$$S_{HASP} = 21{,}400 + 446 + 0 + 0 + 356 + 412 + 120 + 158 + 6{,}424 + 1{,}112 + 700 + 800 + 0 + 432 + 40 + 592 + 0 + 0 + 1{,}592 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0$$

= 34,584 bytes.

12.2.4 Example II -- Storage Requirements for a Typical HASP

Consider a HASP package which has been HASPGENed to be used on a large machine with Remote Job Entry capabilities. The HASPGEN parameters might be set as follows:

&NUMDA	=	2	&MAXCLAS	=	8
&NUMTGV	=	400	&OSINOPT	=	YES
&NUMRDRS	=	2	&XBATCHC	=	W
&NUMTPES	=	1	&TIMEOPT	=	4
&NUMINRS	=	1	&PRTRANS	=	YES
&NUMPRTS	=	2	&ACCTNG	=	YES
&NOPRCCW	=	15	&DEBUG	=	NO
&NUMPUNS	=	1	&TRACE	=	0
&NOPUCCW	=	5	&OREPSIZ	=	0
&NUMCONS	=	0	&DMPTAPE	=	000
&AUTORDR	=	YES	&NUMLNES	=	2
&NUMBUF	=	20	&TPBFSIZ	=	400
&BUFSIZE	=	688	&NUMTPBF	=	2
&NUMOACE	=	2	&NUMRJE	=	2
&NUMWTOQ	=	10	&NUMTPRD	=	2
&MAXJOBS	=	200	&NUMTPPR	=	2
&JITSIZE	=	8	&NUMTPPU	=	2
&MAXXEQS	=	3	&BSC2780	=	YES
&MAXPART	=	3	&BSC2770	=	YES
&NUMDDT	=	30	&STR1978	=	NO
&MONINTV	=	3	&STRCPU	=	NO
&PRIRATE	=	3	&BSCCPU	=	NO
&WTRPART	=	*	&SPOLMSG	=	20

The storage requirements would be computed as follows:

$$DAMAP = 2 \times 50 = 100$$

$$S_{HASP} = 21{,}400 + 992 + 496 + 960 + 984 + 412 + 120 + 316 +$$
$$15{,}360 + 2{,}224 + 1{,}400 + 3{,}200 + 1{,}732 + 648 + 60 +$$
$$1{,}110 + 288 + 108 + 1{,}592 + 24 + 746 + 0 + 292 +$$
$$30 + 0 + 0 + 0 + 0 + (1088+208+264+936+400+336+19+5772)$$
$$= 63{,}517 \text{ bytes.}$$

(The remainder of this page intentionally left blank.)

12.3 HASP CONTROL CARD FORMATS

12.3.1 HASP Job Card Format

The JOB card is a "variable-field" control card which defines the beginning of a job (and, of course, the end of the previous job if there is one) within the input stream. In addition, certain parameters are passed to HASP and to the Operating System via fields and subfields punched into the JOB card.

The format of the JOB card is basically as defined in the JOB CONTROL LANGUAGE Manual (Form #C28-6539).

In particular, HASP requires that the accounting information field be punched in the following format:

(pano,room,time,lines,cards,forms,copies,log,linect)

where:

pano = Programmer's accounting number. This subfield MUST BE PRESENT and must consist of one to four alphameric characters. (Example: "4301")

room = Programmer's room number. This subfield MUST BE PRESENT and must consist of from one to four alphameric characters. (Example: ",E305")

time = Estimated execution time in minutes. This subfield is optional and may consist of up to four numeric digits. If omitted, a standard value will be assumed. (Example: ",30" for 30 minutes)

lines = Estimated line count in thousands of lines. This subfield is optional and may consist of up to four numeric digits. If omitted, a standard number of lines will be assumed.
(Example: ",5" for 5000 lines)

cards = Estimated number of cards to be punched. This subfield is optional and may consist of up to four numeric digits. If omitted, a standard number of cards will be assumed. (Example: ",200" for 200 cards to be punched)

forms = Special forms for printing entire job. This subfield is optional and may consist of up to four numeric characters. If omitted, standard forms will be assumed. (Example: ",0005" for 5-part forms)

copies = Number of times the print output is to be printed. This subfield is optional and may consist of up to two numeric digits. If omitted, one copy will be assumed. (Example: ",2" for two copies)

log = HASP System Log option. This subfield is optional and may consist of one character. If this character is an "N", the

HASP System Log will not be produced. If any other character, or if omitted, the log will be produced.

linect = Lines to be printed per page. This subfield is optional and may consist of up to two numeric digits. If coded as "0" (zero) no automatic overflow will be produced. If omitted, a standard value will be assumed.
(Example: ",34" for 34 lines per page)

The other fields on the JOB card are also interpreted for accounting purposes and Job control.

The job card may be continued in accordance with the Operating System Job Control Language specifications.

To omit a specific subfield, the comma normally punched following the subfield should be punched in the first column of the subfield. To omit the remainder of the subfields, the closing right parenthesis should be punched following the last subfield entered.

The following would be a typical JOB card:

```
//ORBIT    JOB (7808,E305,,2,200),                    CONTINUED
//             'J. JACKSON',MSGLEVEL=1,CLASS=B
```

In this case:

pano	=	7808
room	=	E305
time	=	2 minutes (assumed value)

lines	=	2000 lines
cards	=	200 cards
forms	=	standard forms (assumed)
copies	=	1 copy (assumed value)
log	=	yes (assumed value)
linect	=	standard value (assumed)

12.3.2 SPOOL Priority Card Format

The PRIORITY card is a "fixed-field" control card used to assign a set priority to a job. The format of the card is as follows:

Columns	1 - 10	--	"/*PRIORITY"
	11 - 15	--	blank
	16 - 17	--	p (left justified)
	18 - 80	--	ignored

where "p" is either a number (between 0-15) or the character "*." If "p" is a number, the value of "p" will be assigned as the priority of the job following the PRIORITY card. If "p" is the character "*," or if the PRIORITY card is not present, the priority of the job will then be determined by the estimated execution time and the estimated lines on the JOB card.

The PRIORITY card must immediately precede the JOB card. If it does not, the PRIORITY card will be ignored and the input stream will be flushed until a job card (or another PRIORITY card) is found.

12.3.3 SPOOL Route Card Format

The ROUTE card is a "fixed" control card which allows the user to specify the location to which his output is to be printed or punched. The format of the card is as follows:

Columns:	1 - 7	—	"/* ROUTE"	
	8 - 9	—	blank	
	10 - 14	—	"PRINT" or "PUNCH"	
	15	—	blank	
	16 - 23	—	one of the following device specifications:	
			LOCAL	— Any local device
			REMOTEn	— Remote Terminal "n"
			PRINTERn	— Printer "n" *
			PUNCHn	— Punch "n" *
	24 - 80	—	ignored	

A single ROUTE card can be used to direct either the print or punch routing but not both. If both print and punch are to be routed, two cards must be used.

All ROUTE cards should be placed immediately after the JOB card(s).

* Note — The PRINTERn and PUNCHn specifications are the same as LOCAL unless the specified printer or punch is subject to local print/punch routing.

12.3.4 SPOOL Message Card Format

The MESSAGE card is a "fixed-field" control card which permits the user to send messages to the operator via the operator console at HASP job input time. The format of the card is as follows:

Columns	1 - 9	"/*MESSAGE"
	12 - 71	message to be written
	72 - 80	ignored

All leading and trailing blanks are removed from the message before writing it on the console.

If MESSAGE cards are included as part of a job they should be placed immediately following the JOB card(s) (or after any ROUTE cards). In such cases the job number is appended on the front of the message(s).

If a MESSAGE card is not included within the boundaries of a job, the input device name is appended on the front of the message.

12.3.5 SPOOL Setup Card Format

The SETUP card is a "variable-field" control card which permits the user to indicate the need for certain volumes during the execution phase of his job. The format of the card is as follows:

Columns:	1 - 7	—	"/*SETUP"
	8 - 15	—	blank
	16 - 72	—	volume identifiers separated by commas (i.e., vvvvvv, wwwwww, xxxxxx, ...)
	73 - 80	—	ignored

The volumes required are listed on the console at the time that the job enters the system. The job is then placed in "hold" status pending subsequent release by the operator when the required volumes are available.

All SETUP cards should be placed with the ROUTE and MESSAGE cards after the JOB card(s).

12.3.6 SPOOL Command Card Format

The COMMAND card is a "variable-field" control card used to enter HASP operator commands into the system. The format of the card is as follows:

Columns:	1 - 3	—	"/*$"
	4 - 71	—	operator command
	72	—	if "N" the command will not be repeated on the operator's console.
	73 - 80	—	ignored

Restrictions concerning commands which can be entered from remote terminals are documented in subsection 6 of the Operator's Guide (Section 11).

All COMMAND cards must be placed in the input stream prior to any JOB card. COMMAND cards within jobs will be ignored.

12.4 SPOOL ACCOUNTING CARD FORMAT

COLUMNS	CONTENTS	MODE
1-20	Programmer's name	EBCDIC
21-24	Room number •	EBCDIC
25-27	Spares	N/A
28-31	P. A. number •	EBCDIC
32	Job priority number	BINARY
33-35	Job input time in hundredths of a second	BINARY
36-38	Job output time in hundredths of a second	BINARY
39-40	Number of cards read in	BINARY
41-43	Number of output lines	BINARY
44-45	Number of output cards	BINARY
46-48	Total reader time in hundredths of a second	BINARY
49-51	Total execution time in hundredths of a second	BINARY
52-54	Total print time in hundredths of a second	BINARY
55-57	Total punch time in hundredths of a second	BINARY
58-65	Job name	EBCDIC
66-71	Spares	N/A
72	Identifier (X'FF')	BINARY
73-74	Year	EBCDIC
75-77	Days	EBCDIC
78-80	Job number	EBCDIC

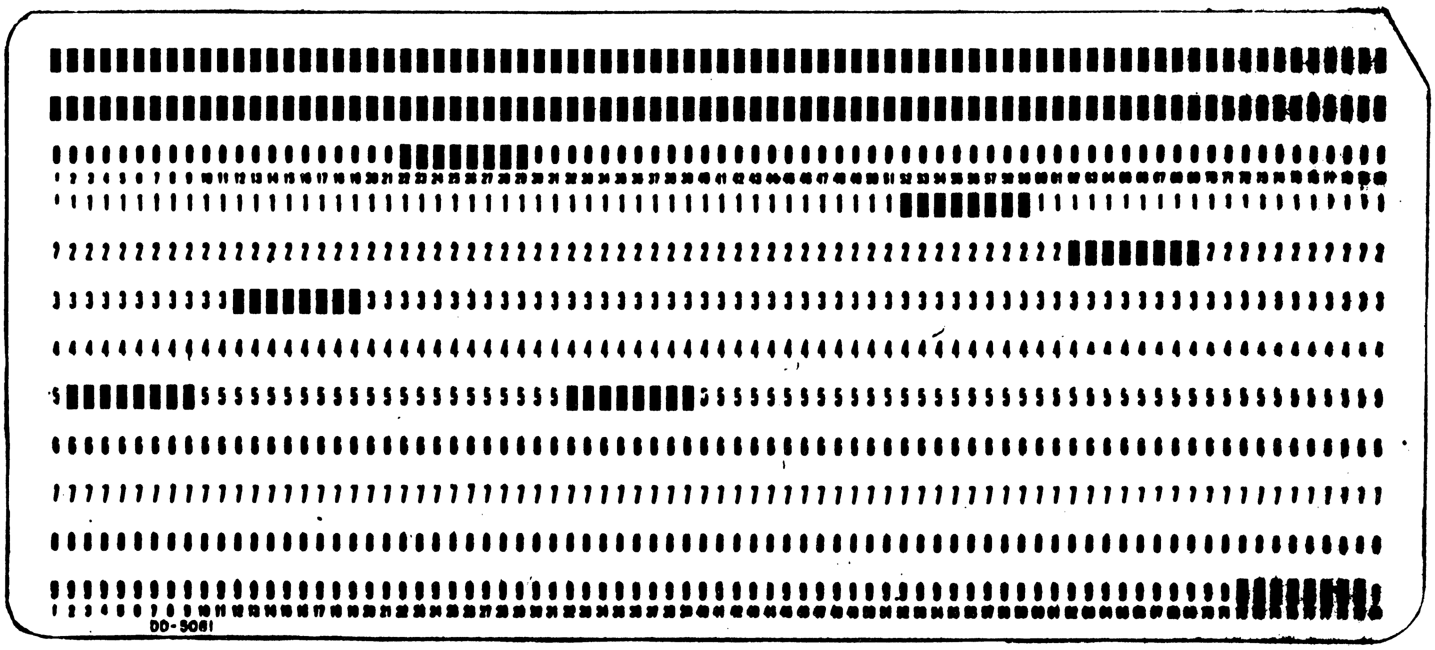
12.5 SPOOL PRINT AND PUNCH ID FORMATS

12.5.1 SPOOL Punch ID Card Format

The punch output will be preceded by an identification card containing the programmer room number and internal job number. To make the card easy to identify, it has an 11-punch and a 12-punch punched in all eighty columns. To make the room number and job number easy to read, each digit is extended over 10 columns. Alphabetic characters are converted to digits as follows:

Alphabetic Characters	Numeric Punch
A or J	1
B, K, or S	2
C, L, or T	3
D, M, or U	4
E, N, or V	5
F, O, or W	6
G, P, or X	7
H, Q, or Y	8
I, R, or Z	9

Below is an example of the punch identification card which would precede a deck punched, for example, for a programmer residing in Room E305, and having an internal job number of 129.



12.5.2 SPOOL Print ID Card Format

The print output for all jobs processed by SPOOL will be preceded and followed by special pages of job identification information. These pages will consist of one line duplicated many times so as to fill the page. This line will have the following format:

Columns	Contents
1-17	HASP identification
18-22	periods (.)
23-31	START JOB
	.CONT JOB
	..END JOB
32-35	job number assigned by HASP
36-40	periods (.)
41-51	time of printing the page in form: hh.mm.ss $\begin{Bmatrix}\text{AM}\\\text{PM}\end{Bmatrix}$
52-61	date of printing the page in form: day month year
62-65	periods (.)
66-69	ROOM
70-74	room number
75-78	periods (.)
79-86	OS jobname
87-90	periods (.)
91-115	programmers name padded with trailing periods (.)
116-132	HASP identification

12.6 HASP CODING CONVENTIONS

Each logical section of code within HASP has been assigned a unique alphabetic header character which is used as the first character of symbolic names within that section. The character "$" is reserved to preface symbolic names used for inter-routine communication. The following are the currently assigned HASP header characters:

A - Asynchronous Input/Output Processor

B - Buffer Handling Routines

C - Operator Console and Command Processor

D - Dump Routine

E - HASP I/O Supervisor

F - unassigned

G - Priority Aging Processor

H - HASP Dispatcher

I - Interval Timer Routines

J - unassigned

K - Checkpoint Processor

L - Log Processor

M - MULTI-LEAVING Line Manager

N - Initialization Routine

O - Overlay Service Routines

P - Print/Punch Processor

Q - Queue Manager Routines

R - Input Service Processor

S - unassigned

T - Direct Access Space Allocation Routines

U - Unit Allocation Routines

V - Purge Processor

W - Write To Operator Routine and Console Processor

X - OS Execution Processor

Y - unassigned

Z - unassigned

12.7 GENERAL HASP RESTRICTIONS

Because of the techniques utilized in the implementation of HASP, certain features and/or functions of the Operating System may not be available or may differ in operation in a system utilizing HASP. Additionally, certain features and functions implemented by HASP may not perform in the same manner as similar functions replaced in OS; or may be affected by various environmental or operating characteristics of a particular installation. The following sections indicate a partial list of these restrictions, excluding those restrictions which are made obvious by the general interface technique utilized by HASP.

12.7.1 Unsupported OS Features

A. All I/O operations for SYSIN/SYSOUT data SPOOLed by HASP will appear to the user as the direct use of unit record devices (which do not actually exist). A program which depends upon the physical characteristics of a particular device for processing SYSIN/SYSOUT data may, therefore, not function properly in a HASP environment.

B. All I/O requests for SYSIN/SYSOUT data files controlled by HASP must be made through the standard use of the EXCP macro instruction.

C. SYSIN/SYSOUT operations, which appear to programs as the direct use of unit record devices, are actually performed by HASP by simulating the function of the unit record device. In simulating the operation of these devices, certain functions of the actual device may not be accurately simulated by HASP. These include:

- Timing - I/O operations to the pseudo devices will not have the same timing characteristics as to an actual device.

- Data Chaining - HASP does not support the Channel Command Word data chaining feature of System/360, System/370 when simulating unit record devices. The command chaining feature is, however, fully supported.

- Input/Output Appendages - In responding to requests for I/O operations, HASP will enter, if specified, only the normal channel appendage. Because of the instantaneous nature of HASP "I/O" operations, the use of any other appendage is not applicable and will be ignored if specified.

D. The use of the Checkpoint/Restart feature of OS is, in general, inconsistent with the SPOOLing techniques utilized by HASP and, in many cases, will not function properly in a HASP environment. It is the responsibility of the user to verify the compatibility of the various features of Checkpoint/ Restart to be utilized. Jobs requiring the use of unsupported features of Checkpoint/Restart may be run, in a HASP environment (outside of the control of HASP).

E. In the processing of special forms types on SYSOUT data sets, only the numeric portion of the characters specified will be utilized by HASP for control purposes. Although alphabetic forms types may be specified, it is recommended that numeric-only types be utilized to avoid possible operational problems.

F. No provision has been made in HASP to support the ROLLOUT/ ROLLIN feature of OS. It is the responsibility of the user to evaluate the compatibility and accuracy of this feature in a HASP environment.

12.7.2 HASP - Function/Feature Restrictions

A. The capability to dynamically withdraw HASP from the system and continue operation is intended, primarily, as a programming aid for the systems programmers and is highly dependent upon individual operational environments. For these reasons, this function is not designed to (and may not) effect a complete withdrawal such that the previous presence of HASP is completely transparent to the host Operating System. Each installation utilizing this feature should individually verify the accuracy and completeness of the withdraw operation.

B. The console support capability of HASP (&NUMCONS>0) is intended primarily for standard Write-To-Operator support and may not function properly in certain unusual conditions. In particular, REPLYs to WTORs erroneously left pending by completed job steps may not be processed correctly. Installations with unusual support requirements of this type may circumvent the problem by specifying &NUMCONS=0. Also, see HASPGEN parameter &NUMCONS for other restrictions.

C. HASP will not operate correctly if two or more jobs being simultaneously processed by OS have identical job names. While HASP will protect against this circumstance for jobs under its control, it is the responsibility of the user to insure that no job, submitted outside of HASP control, has the same job name as any job being controlled by HASP.

D. Because of the HASP/OS interface techniques and the total system control status of HASP, no provision has been made to allow processing to continue after a HASP failure. Any abnormal termination of HASP is considered a system failure and requires a re-IPL.

E. All unit-record devices of the type utilized by HASP must be attached to the system (appear as physically existent) at the time HASP is invoked if they are to be subsequently utilized.

12.8 JCL PROCESSING

The HASP philosophy of providing an interface to OS/360 which is essentially transparent to the user is supplemented by the collection of HASP routines which examine and alter JCL statements during system operation. Since the HASP routines have access to every JCL statement destined for the OS scheduler, it is unnecessary to provide a special Procedure Library (SYS1.PROCLIB) for HASP operation. In addition, HASP features such as priority and job class scheduling can be easily realized by including or changing the appropriate "CLASS" and "PRTY" fields on the JOB JCL statements. HASP requirements for input stream (DD*, DD DATA) and output stream (SYSOUT) correspondence to user defined pseudo I/O units is a major function of the JCL routines.

Access to the JCL routines is provided thru the standard OS/360 Reader/Interpreter "exit list" feature which allows the user to specify a routine to be given control whenever a JCL statement has been encoded and is ready for individual statement analysis. The encoding scheme used is described in the MVT Job Management PLM.

12.8.1 JCL PROCESSING OS/360 INTERFACE

XNEXRCON - NEL Exit List Reconstruction

The Exit List Reconstruction program receives control from the HASP LINK/XCTL interface routine when a LINK to the Reader/Interpreter initialization module (IEFVH1) is intercepted. The NEL (Interpreter Entrance List) and associated Exit List is examined for a Special Access method entry which indicates that the Reader/Interpreter is being used as a subroutine to process "In-core" JCL. If the Reader/Interpreter is being used to process jobs, a test is made for the HASP Reader identified by the default SYSOUT device name of "SPOOL" in the HASPRDR Proclib Procedure. When the HASP Reader is identified, the Exit List is modified to include linkage to the main HASP JCL program (XJCLSCAN) from the Reader/Interpreter after each JCL statement is encoded.

12.8.2 JCL PROCESSING MAIN PROGRAMS

XJCLSCAN - JCL Scan and Control

The JCL Scan and Control Program is entered from the Reader/ Interpreter module IEFVFA as a result of the Exit List linkage established by the HASP routine XNEXRCON. The following major functions are performed:

1. A return PSW is constructed from the IEFVFA return register (R14) and the left half of the current PSW provided by entry to the HASP initialization SVC routine. This special use of the initialization SVC allows the JCL routines to operate disabled when necessary.

2. The Reader/Interpreter internal text pointer is obtained and saved for subsequent text processing. A test is made on the first word of the text to determine if an extensive JCL statement has caused overflow to SYS1.JOBQUE. If this condition exists, HASP is unable to process the statement.

3. The first JCL key in the Reader/Interpreter internal text is tested for "JOB", "EXEC" and "DD" values and control passed to the corresponding HASP processor as indicated below:

KEY VALUE	HASP PROCESSOR
JOB	XJCLJBPR
EXEC	XJCLXQPR
DD	XJCLDDPR

4. All JCL processors terminate by passing control to XJCLEXIT in the Scan and Control routine. The termination function is accomplished by restoring saved registers and issuing a LPSW using the PSW constructed when the control routine is entered. Control is returned to IEFVFA.

XJCLJBPR - JOB Statement Routine

The JOB statement routine examines and changes all JOB statements processed by the Reader/Interpreter. The major functions performed are:

1. A new JOB internal text string is constructed in a HASP work-area. The new text consists of the original text with "CLASS", "PRTY"; "TYPRUN" and "MSGCLASS" entries deleted if they were specified by the user.

2. A new CLASS entry is produced by extracting the "PITCLAS" field from the PIT (Partition Information Table) associated with the job being processed by the Reader/Interpreter.

3. A new "PRTY" entry is produced by extracting the "PITPRIO" field from the PIT.

4. A MSGCLASS value corresponding to the first class given in the parameter &WTRCLAS is entered into the text unless the originally specified MSGCLASS corresponded to any class given in &WTRCLAS.

5. This newly constructed text is then moved to the Reader/Interpreter text area.

XJCLXQPR - EXEC Statement Routine

The EXEC statement routine does not examine the associated internal text if the Execution Task Monitor has not been selected (&MONINTV=0). If the Execution Task Monitor has been selected, then XJCLXQPR performs the following functions:

1. The value of the Monitor priority represented by the &XZPRTY is compared with the PIT priority field associated with the Job being interpreted. If the values are not equal, then no further action is taken on the EXEC statement.

2. If the priorities are equal, the internal text is examined for the key value of "DPRTY=". If this parameter has not been coded, no further action is taken.

3. If "DPRTY" has been coded under the circumstances cited, it is removed from the internal text and therefore not processed by OS.

The overall purpose of the action taken by XJCLXQPR is to prevent circumvention of the function of the Execution Task Monitor by coding the "DPRTY" field.

XJCLDDPR - DD Statement Routine

The DD Statement Routine examines all DD statements for SYSOUT or DD*, DD DATA Specifications and performs the following major functions:

1. The key content of the statement is determined by use of the XINTSCAN routine.

2. If the statement does not contain a SYSOUT specification, control goes to the DD*/DD DATA test routine XJCLDDDT.

12.8.3 JCL PROCESSING INPUT STREAM PROGRAMS

XJCLDDDT - DD*, DD DATA Test Routine

This routine tests for a DD* or DD DATA specification and goes to XJCLDDDA if either is found . If the DD statement does not contain an * or DATA, control returns to the Reader/Interpreter via XJCLEXIT.

NOTE: The positional parameters "*" and "DATA" are coded as "$" and "CATA" by the HASP Card Reader Service Main Processor and are recognized in these formats by XJCLDDDT.

XJCLDDDA - DD*, DD DATA Processing Routine

This section of XJCLDDDT processes the DD* and DD DATA statements in the following manner:

1. The execution PCE for the current job is established using the XESTBPCE subroutine.

2. The DDNAME (if it exists) and the internal text for "UNIT=xxx" is constructed in the HASP text area.

3. The BLKSIZE, LRECL analysis routine XJCLSHFL is entered.

4. Control goes to the termination section of XJCLDDPR.

12.8.4 JCL PROCESSING OUTPUT STREAM PROGRAMS

XJCLDDCB - SYSOUT/DCB Routine

Output stream statements (SYSOUT) with DCB parameters are processed by this routine as follows:

1. The DCB preservation routine (XJCLDECB) is invoked to extract and then delete user BLKSIZE and LRECL specifications while preserving all other DCB parameters.

2. The HASP maximum/default output stream BLKSIZE (defined by &OBLKSZE) is moved to a test location (XDEFOBXX) for further analysis by XJCLSHFL.

3. The BLKSIZE, LRECL analysis routine XJCLHSFL is entered.

4. Control goes to the termination section of XJCLDDPR.

XJCLDDWR - SYSOUT with Special Writer Routine

DD statements with a SYSOUT specification which includes a special writer entry are processed by this routine:

1. A test is made for a UNIT specification in the statement containing the SYSOUT disposition. If UNIT has been specified, control returns to the Reader/Interpreter via XJCLEXIT.
2. If UNIT is not specified, the text for UNIT=SYSDA is added to the Reader/Interpreter text and processing is terminated via XJCLEXIT. This addition is made to overcome the HASPRDR procedure default SYSOUT unit of SPOOL.

XJCLDDFR - SYSOUT with Forms Routine

This routine processes a forms specification in conjunction with a SYSOUT disposition:

1. A test is made to determine if the current job has depleted the special forms pseudo devices (1442 for punch, 1443 for print) and the forms specification is ignored if so.
2. The Execution Processor PCE for the current job is established using the XESTBPCE subroutine.
3. The Execution Processor routine XGETDDB is used to get a DDT. If a DDT is not available, control goes to XJCLWAIT to place the Reader/Interpreter task in an OS wait state pending the availability of a DDT.
4. The Execution Processor routine XGETUCB is used to get the appropriate pseudo device UCB. If a UCB of the correct type is not available, control goes to XJCLWAIT.

5. The DDBTYPE entry in the acquired DDT is set to indicate the output type according to the SYSOUT class specified (print, punch, special routing print, or special routing punch).
6. The DDBSTAT1 entry in the acquired DDT is set to indicate no primary buffer.
7. The DDBSTAT2 entry in the acquired DDT is set to indicate an allocatable UCB exists.
8. The EBCDIC unit address from the DDBUNIT field of the DDT is moved to the UNIT=xxx internal text establishing the HASP pseudo unit for forms processing.
9. The extracted forms field is moved to the DDBFORMS field of the DDT.
10. The UNIT=xxx text is moved to the HASP text buffer and control goes to the DCB test section of XJCLDDPR.

12.8.5 JCL PROCESSING SUBROUTINES

XJCLSHFL - BLKSIZE/LRECL Subroutine

This routine is used to analyze BLKSIZE and LRECL values, if specified, for both input stream (DD*, DD DATA) and output stream (SYSOUT) statements. Processing proceeds as described below:

1. The key status word (XSTATUS) is tested for both BLKSIZE and LRECL specifications and control goes to the section (XJCLBOTH) which processes this case. XJCLBOTH compares the user LRECL with the maximum/default HASP BLKSIZE (&OBLKSZE or &IBLKSZE). If the specified LRECL is greater than the HASP maximum BLKSIZE, both LRECL and BLKSIZE are set to the value of the HASP maximum/default BLKSIZE (&OBLKSZE OR &IBLKSZE) and processing is terminated.

2. If both BLKSIZE and LRECL have not been specified, a test is made for BLKSIZE only. If BLKSIZE has not been specified, the HASP default BLKSIZE (&OBLKSZE or &IBLKSZE) is used and processing is terminated.

3. If BLKSIZE alone is specified, it is used as is and processing is terminated.

XJCLDECB - DCB Preservation Subroutine

The purpose of this routine is to investigate DCB parameters specified on input stream (SYSOUT) and output stream (DD*, DD DATA) statements and to perform the following functions:

1. Locates the position of the DCB substring in the Reader/Interpreter internal text by use of the XINTSCAN subroutine.

2. Examines all DCB subparameters in the DCB substring. BLKSIZE and LRECL specifications are extracted by the XJCLXTRC Routine for analysis by XJCLSHFL.

3. All user DCB subparameters, except BLKSIZE and LRECL, are moved to the HASP text buffer for retention.

4. When the end of the DCB substring is reached, control returns to the caller.

XJCLXTRC - BLKSIZE/LRECL Data Extractor Subroutine

This routine is used to extract the user specified BLKSIZE or LRECL data fields from the Reader/Interpreter internal text for later analysis by XJCLSHFL. Processing is as follows:

1. The address of the target field for the extracted data is contained in Register WD. This field is set to decimal zeros.

2. The maximum field length, defined by XJCLMXLT, is used to extract the low order XJCLMXLT bytes from the Reader text for insertion into the target field.

3. Control returns to the caller.

XESTBPCE - Execution Processor PCE Search Subroutine

This routine finds the Execution Processor PCE associated with the job being processed by the Reader/Interpreter.

1. The current (Reader/Interpreter) TCB address is found via the OS Communication Vector Table.

2. The Execution Processor PCE's are searched for the PCE corresponding to the current TCB.

3. The correct PCE address is returned to the caller in the SAVE Register.

XINTSCAN - Internal Text Scan Subroutine

This routine provides two major functions. The contents of the Reader/Interpreter internal text with respect to the key values encoded from the users original JCL statement can be determined. The position of a particular key value in the text string can be determined. The general program logic is:

1. The pointer to the first key in the internal text is obtained from XINTKEYS which is set when XJCLSCAN is entered for each JCL statement.

2. A test is made on the contents of Register R0. If R0 contains the value of XJSENDKE (a special key value indicating the end of the text string) then control goes to the section of XINTSCAN which determines the key contents of the text. Any other value in R0 is assumed to be a request to find that particular key in the text string.

3. When a particular key search is specified, the subroutine XFINDKEY is used to obtain successive keys from the internal text until the requested key is found or until the end of the string is reached. A successful search is flagged by setting R0 non-zero and placing the pointer to the requested key location in R1 and returning to the caller. If the search is

unsuccessful, R0 is set to zero before returning.

4. The key contents of the internal text string is reflected by the final disposition of the XSTATUS word. XSTATUS bits are set on as dictated by the key values in the text and the selected key values defined by the table XSTATDEF. Whenever an internal text key is found which matches an entry in the XSTATDEF table, a corresponding bit is turned on in the XSTATUS word. Using this scheme, the entire internal text string is examined and the selected key values, if they exist in the string, are reflected by XSTATUS.

5. After the setting of XSTATUS, the address of the end key of the text string is saved in XINTENDK and the length of the text is calculated and saved in XINTEXTL. XSTATUS is placed in R1 and control returned to the caller.

XFINDKEY - Next Key Byte Search Subroutine

This routine is used to find the position of the next key byte in the internal text given the position of the preceding key byte. R1 points to the preceding (current) key byte on entry. R1 points to the next key byte on exit.

12.9 HASP/RJE LINE TRANSMISSION TECHNIQUES

The following sections discuss, in detail, the line transmission techniques utilized by HASP to communicate with remote terminals which have programming capabilities. Transmission formats to mechanical terminals, such as the IBM 1978, follow the specifications as outlined in the manuals for those terminals.

12.9.1 1-7/8 of 8 Encoding

In order to support the entire EBCDIC character set from a remote station, with no unnecessary degradation of line transmission, a new encoding technique, inadaquately named 1-7/8 of 8, was devised. 1-7/8 of 8 is (obviously) based on the standard STR-4 of 8 encoding and operates in the following manner:

The standard 4 of 8 character encoding has been entirely re-defined such that, the logically highest 48 characters of the 4 of 8 set have been defined as the 48 most common EBCDIC (BCD) characters. The remaining 16 4 of 8 characters have been reserved for use as 1-7/8 characters to represent the remaining EBCDIC characters. These 16 characters are defined as the hexadecimal digits "0" - "F" so that any of 256 character combinations may be represented with two (2) 1-7/8 characters. Since 1-7/8 characters represent the numerically lowest characters of both the 4 of 8 character set <u>and</u> the EBCDIC character set, the receiving program may detect 1-7/8 encoding (either before or after 4 of 8 translation) by a single logical compare instruction. This, then allows for the intermixing, within a single record, of normal character encoding (1 for 1) and 1-7/8 (2 for 1) with <u>no</u> additional control information. Any character represented by 1-7/8 encoding may be reconstructed with a single MOVE WITH OFFSET instruction in the receiving program.

Since a very high percentage of all characters contained in a program source deck are of the 48 character set, normal transmission of jobs from a remote site will be in a 1 for 1 character representation with an occasional "unusual" character represented by 1-7/8 encoding.

This feature also allows the random interspersing of OS/360 object decks in an input stream. Figure 12. 9. 1 shows the 1-7/8 of 8 definitions for the 4 of 8 character set.

Figure 12.9.1 1-7/8 of 8 — 4 of 8 Conversion Table (48 characters)

Graphic	EBCDIC	4 of 8	Graphic	EBCDIC	4 of 8
(1-7/8 of 8)	00	0F	G	C7	8E
(1-7/8 of 8)	01	17	H	C8	93
(1-7/8 of 8)	02	1B	I	C9	95
(1-7/8 of 8)	03	1D	J	D1	96
(1-7/8 of 8)	04	1E	K	D2	99
(1-7/8 of 8)	05	27	L	D3	9A
(1-7/8 of 8)	06	2B	M	D4	9C
(1-7/8 of 8)	07	2D	N	D5	A3
(1-7/8 of 8)	08	2E	O	D6	A5
(1-7/8 of 8)	09	36	P	D7	A6
(1-7/8 of 8)	0A	3A	Q	D8	A9
(1-7/8 of 8)	0B	3C	R	D9	AA
(1-7/8 of 8)	0C	47	S	E2	AC
(1-7/8 of 8)	0D	4B	T	E3	B1
(1-7/8 of 8)	0E	4D	U	E4	B2
(1-7/8 of 8)	0F	4E	V	E5	B4
0	F0	56	W	E6	B8
1	F1	5A	X	E7	C3
2	F2	5C	Y	E8	C5
3	F3	63	Z	E9	C6
4	F4	65	.	4B	C9
5	F5	66	(	4D	CA
6	F6	69	+	4E	CC
7	F7	6A	&	50	D1
8	F8	6C	*	5C	D2
9	F9	71	)	5D	D4
A	C1	72	-	60	D8
B	C2	74	/	61	E1
C	C3	78	,	6B	E2
D	C4	87	'	7D	E4
E	C5	8B	=	7E	E8
F	C6	8D	(blank)	40	F0

12.9.2 Character Compression Techniques

HASP, when communicating with a remote terminal with programming capabilities (as indicated at HASPGEN), optionally utilizes a duplicate character compression algorithm to improve transmission line efficiency. This algorithm breaks each logical record into one or more character strings, prefaced by a pair of string control bytes (SCBs) to indicate the type and extent of the character string. SCBs are in the form - IJ KL where:

JL = a count describing the extent of the character string. This count is obviously derived by combining the low order digit of both SCBs. Counts on records created by HASP are one less than actual, while the count on substrings received by HASP are expected to represent the actual count.

I = a HEX digit which identifies the type of character string as follows:

I=X'O' - indicates a normal character string. (i.e.the number of characters described by 'JL' should be inserted intact when reconstructing the record image.

I=X'F' - indicates that 'JL' blank characters should appear in the reconstructed record image. Note that no additional characters are required with this type of string.

I=X'4' or X'C' - indicates that the character immediately following the SCBs should be duplicated 'JL' times and inserted into the image being reconstructed.

K = a HEX digit to additionally describe a character substring. K is presently 0 in all cases but is reserved for future expansion.

Any character in a normal character string or the sample character in non-blank compression may be encoded in a 1 for 1 representation or in 1-7/8 representation (2 for 1). The count indicated by 'JL' is a character count, rather than a byte count, so that a character represented by 2 bytes (1-7/8) causes only a count increment of 1.

A logical OUTPUT record from HASP, which may consist of several SCBs and associated character strings, is terminated by the special SCB of X'F0F0'. (This record terminator is not used on records sent to HASP since a reconstructed record length of 80 may be assumed). A physical record, which may consist of several logical records, is terminated by the special SCB of X'F0F0' immediately following the last logical record. (Again in the case of input records to HASP the X'F0F0' is <u>not</u> used to terminate the buffer)

The following example illustrates each of the HASP encoding techniques. Assume the record below is to be encoded by HASP for transmission to a remote terminal as punched output.

```
0     00          33       44     44          55          8
1     67          56       23     78          43          0
******bbbbb. . . bbAbTEST ? bbbbb? ? ? ? ? ? ? bbbb. . . . b
```

Where b=BLANK=X'40', ? =NON 48 Character Set Character=X'6F'

This record, after compression and before 4 of 8 translation, would appear as below (with SCBs underlined):

40055CF10C0006C140E3C5E2E3060FF0044006060FF0F0

Note that trailing blanks are not encoded by HASP. After translating to 4 of 8 characters the record would be:

F027D25A470F2B72F0B18BACB12B4E561EF02B2B4E5656

Note that SCB characters are encoded, as required, in 1-7/8.

12.9.3 HASP Transmission Block Encoding

The following pages describe the format of physical records transmitted by HASP to a programmable remote terminal and the format of physical records expected by HASP from such a terminal. All values are indicated in EBCDIC (rather than 4 of 8) for simplicity.

PRINT RECORD

The first two bytes of a logical print record are used to indicate carriage control requirements to the remote terminal program. The following list indicates all current carriage control characters and their meaning.

BYTE 1	=	X'04' - Space immediate "N" spaces
	=	X'05' - Skip immediate to channel "N"
	=	X'06' - Space "N" after print
	=	X'07' - Skip to channel "N" after print
	=	X'08' - Suppress space
BYTE 2	=	X'01' - X'03' - "N" as described in space commands above.
	=	X'01' - X'0F' - "N" as described in skip commands above.
	=	X'00' - if suppress space is indicated above.

Immediately after the carriage control bytes, any number of character strings constituting the line to be printed may follow, ended by a SCB of X'F0F0' to indicate the end of record. (In the case of carriage control only, the X'F0F0' may immediately follow the carriage control information). The above sequence may be repeated as many times as buffer space permits, followed finally by another X'F0F0' to indicate end-of-buffer.

NOTE: In order to minimize CPU requirements at the remote terminal, HASP does not utilize 1-7/8 encoding on print records. The 16 characters normally utilized for 1-7/8 encoding are defined as additional print characters, thus yielding a 64 character print set. An attempt to print a character which is not in this character set results in the substitution of a blank for that character. Figure 12.9.3 indicates the 4 of 8 definitions for the 64 character print set. Although data characters are not encoded in 1-7/8, carriage control and String Control bytes may be encoded in 1-7/8 as required.

Figure 12.9.3 EBCDIC/4 of 8 Conversion Table (64 characters)

Graphic	EBCDIC	4 of 8	Graphic	EBCDIC	4 of 8
¢	4A	0F	G	C7	8E
<	4C	17	H	C8	93
|	4F	1B	I	C9	95
!	5A	1D	J	D1	96
$	5B	1E	K	D2	99
;	5E	27	L	D3	9A
¬	5F	2B	M	D4	9C
%	6C	2D	N	D5	A3
_	6D	2E	O	D6	A5
>	6E	36	P	D7	A6
?	6F	3A	Q	D8	A9
:	7A	3C	R	D9	AA
#	7B	47	S	E2	AC
@	7C	4B	T	E3	B1
"	7F	4D	U	E4	B2
≠	E0	4E	V	E5	B4
0	F0	56	W	E6	B8
1	F1	5A	X	E7	C3
2	F2	5C	Y	E8	C5
3	F3	63	Z	E9	C6
4	F4	65	.	4B	C9
5	F5	66	(	4D	CA
6	F6	69	+	4E	CC
7	F7	6A	&	50	D1
8	F8	6C	*	5C	D2
9	F9	71	)	5D	D4
A	C1	72	-	60	D8
B	C2	74	/	61	E1
C	C3	78	,	6B	E2
D	C4	87	'	7D	E4
E	C5	8B	=	7E	E8
F	C6	8D	(blank)	40	F0

PUNCH RECORD

A punch record generated by HASP for transmission to a terminal has exactly the same format as a print record. The punch record is so identified by "carriage control" bytes of X'0F0F'. In order to support the punching of OS/360 object decks at the remote site, 1-7/8 encoding is utilized as required.

INPUT RECORD

Input records to HASP from a remote terminal consist only of character strings and their associated SCBs. Note that no end-of-record characters are used to separate card images. The end-of-logical-record condition is assumed by HASP upon expanding the 80th character of a card image. The end-of-buffer condition will be detected by HASP for the byte count of the data transmitted. 1-7/8 encoding is utilized when required.

12.10 HASP INTERNAL READER

A procedure exists in HASP to allow the introduction of jobs directly into the HASP job stream from any other program operating in the system. The following sections describe techniques to accomplish this.

12.10.1 PROCEDURES FOR USING THE HASP INTERNAL READER

The method of passing jobs to HASP through the internal reader is by writing "cards" to a pseudo 2520 card punch device. Standard OS/360 QSAM PUT or BSAM WRITE macros may be used to write the "cards" to the pseudo punch. The information which would be physically punched into a real 2520 card punch will be passed to the normal HASP reader for insertion into the HASP Job Queue. The last "job" must be followed by a "card" with an end of file indicator (/*EOF in columns 1-5). The end of file "card" is used to free the last job allowing it to be scheduled for execution.

12.10.2 JCL CONSIDERATIONS

Since any system or user task may utilize the HASP internal reader, the method of allocation and controlling the use of the device is via OS/360 Job Control Language. Figure 12.10.1 shows an example of an OS/360 IEBGENER utility run which reads Job Control Language card images from a disk data set passing the jobs to HASP. The program that created the data set inserted the end of file card at the end of the data.

12.10.3 OS/360 - SYSGEN CONSIDERATIONS

Pseudo 2520 punch units must be specified at OS/360 SYSGEN time. The device addresses selected as pseudo punches must be legal System/360 addresses but must <u>not</u> be recognized by the physical devices or control units attached to the System/360. One device should be generated for each internal reader allocated at any given time and should correspond to the value of the SPOOLGEN parameter &NUMINRS. The following card might be used to generate an appropriate Unit Control Block.

```
IODEVICE    UNIT=2520,ADDRESS=301,MODEL=B2
```

The devices should be descriptively named for ease in allocation. The following card might be used to name three internal readers:

```
UNITNAME    UNIT=(301,302,303),NAME=INTRDR
```

Figure 12.10.1 Sample Use of HASP Internal Reader

```
//PASSJOB  JOB (0000,0000),'PASS JOB STREAM',MSGLEVEL=1
//COPY     EXEC PGM=IEBGENER
//SYSIN      DD DUMMY
//SYSPRINT   DD SYSOUT=A
//SYSUT1     DD DSNAME=DAYSWORK,DISP=SHR
//SYSUT2     DD UNIT=INTRDR,DCB=(RECFM=F,BLKSIZE=80,LRECL=80)
```

12.11 MULTI-LEAVING

"MULTI-LEAVING" is a term which describes a computer-to-computer communication technique developed for use by the HASP SYSTEM. In a gross sense, MULTI-LEAVING can be defined as the fully synchronized, pseudo-simultaneous, bi-directional transmission of a variable number of data streams between two or more computers utilizing binary synchronous communications facilities.

The following section describes, in general terms, the basic structure of MULTI-LEAVING. Section 12.10.2 describes the detailed specifications of the MULTI-LEAVING control information and Section 12.10.3 discusses the application of MULTI-LEAVING to the **HASP** BSC/RJE support.

12.11.1 MULTI-LEAVING Philosophy

The basic element for MULTI-LEAVED transmission is the character string. One or more character strings are formed from the smallest external element of transmission - the phsyical record. These phsyical records are input to MULTI-LEAVING and may be any of the classic record types (card images, printed lines, tape records, etc). For efficiency in transmission, each of these data records is reduced to a series of character strings of two basic types. These two types are (1) a variable length nonidentical series of characters and, (2) a variable number of identical characters. Because of the high frequency occurrance of blank characters, a special case is made in 2 above when the duplicate character is a blank. An eight bit control field, termed a String Control Byte (SCB), precedes each character string to identify the type and length of the string. Thus a string as in 1 above is represented by an SCB followed by the nonduplicate characters. A string of consecutive, duplicate, nonblank characters (as in 2 above) can be represented by an SCB and a single character (the SCB indicates the duplication count and the character following indicates the character to be duplicated). In the case of an all blank character string, only an SCB is required to indicate both the type and number of blank characters. A data record to be transmitted is therefore segmented into the optimum number of character strings (to take full advantage of the identical character compression) by the transmitting program. A special SCB is utilized to indicate the grouping of character strings which compose the original physical record. The receiving program can then reconstruct the original record for processing.

In order to allow multiple physical records of various types to be grouped together in a single transmission block, an additional eight bit control field precedes the group of character strings representing the original physical record. This field, the Record Control Byte (RCB), identifies the general type and function of the physical record (input stream, print stream, data set, etc). A particular RCB type has been designated to allow the passage of control information between the various systems. Also, to provide for simultaneous transmission of similar functions (i.e. multiple input streams, etc) a stream identification code is included in the RCB. A second 8 bit control field, the Sub-Record Control Byte (SRCB) is also included immediately following the RCB. This field is utilized to supply additional information concerning the record to the receiving program. For example, in the transmission of data to be printed, the SRCB can be utilized for carriage control information.

For actual MULTI-LEAVING transmission, a variable number of records may be combined into a variable block size, as indicated previously (i.e. RCB,SRCB,SCB1,SCB2,...SCBn,RCB,SRCB,SCB1,...etc). The MULTI-LEAVING design provides for two (or more) computers to exchange transmission blocks, containing multiple data streams as described above, in an inter-leaved fashion. To allow optimum use of this capability, however, a system must have the capability to control the flow of a particular data stream while continuing normal transmission of all others. This requirement becomes obvious if one considers the case of the simultaneous transmission of two data streams to a system for immediate transcription to physical I/O devices of different speeds (such as two print streams). To provide for the metering of the flow of individual data streams, a Function Control Sequence (FCS) is added to each transmission block. The FCS is a sequence of bits, each of which represent a particular transmission stream. The receiver of several data streams can temporarily stop the transmission of a particular stream by setting the corresponding FCS bit OFF in the next transmission to the sender of that stream. The stream can subsequently be resumed by setting the bit ON.

Finally, for error detection and correction purposes, a Block Control Byte (BCB), is added as the first character of each block transmitted. The BCB, in addition to control information, contains a modulo 16, block sequence count. This count is maintained and verified by both the sending and receiving systems to exercise a positive control over lost or duplicated transmission blocks.

In addition to the normal binary synchronous text control characters (STX, ETB, etc), MULTI-LEAVING utilizes two of the BSC control characters -- ACK0 and NAK. ACK0 is utilized as a "filler" by all systems to maintain communications when data is not available for transmission. NAK is used as the only negative response and indicates that the previous transmission was not successfully received. Figure 12.11.1 indicates the format of a typical MULTI-LEAVING transmission block.

Figure 12.11.1 - Typical MULTI-LEAVING Transmission Block

bytes	
DLE	- BSC Leader (SOH if no transparency feature)
STX	- BSC START-OF-TEXT
BCB	- Block Control Byte
FCS	- Function Control Sequence
FCS	- Function Control Sequence
RCB	- Record Control Byte for record 1
SRCB	- Sub-Record Control Byte for record 1
SCB	- String Control Byte for record 1
DATA	- Character String
SCB	- String Control Byte for record 1
DATA	- Character String
SCB	- Terminating SCB for record 1
RCB	- RCB for record 2
SRCB	- SRCB for record 2
SCB	- SCB for record 2
DATA	- Character String
SCB	- Terminating SCB for record 2
RCB	- Transmission Block Terminator
DLE	- BSC Leader - (SYN if no transparency feature)
ETB	- BSC Ending Sequence

12.11.2 MULTI-LEAVING Control Specification

The following pages indicate the bit-by-bit definitions of the various MULTI-LEAVING control fields and notes concerning their utilization.

String Control Byte (SCB)

O	K	L	J	J	J	J	J

0 7

Usage: Control field for data character strings

Bit Meanings:

O = 0 = End of record (K L J J J J J = 0)

O = 1 = All Other SCB's

K = 0 = Duplicate Character String

L = 0 = Duplicate Character is blank

L = 1 = Duplicate Character is nonblank (and follows SCB)

JJJJJ = Duplication count

K = 1 = Nondupiicate Character String

LJJJJJ = Character String Length

NOTES:

1. If KLJJJJJ = 0 and O = 1, SCB indicates record is continued in next transmission block.
2. Count units are normally 1 but may be in any other units. The units utilized may be indicated as function control sign-on or dynamically in the SRCB.

Record Control Byte (RCB)

O	I	I	I	T	T	T	T

0 7

Usage: To identify each record type within a transmission block

Bit Meanings:

O = 0 = End of transmission block (I I I T T T T = 0)

O = 1 = All Other RCB's

III = Stream identifier - used to identify streams of multiple identical functions (i.e. multiple print streams to a multiple printer terminal, etc.)

III = Control information if TTTT = 0 (control record)

- = 000 = Reserved for future expansion
- = 001 = Request to initiate a function transmission (Prototype RCB for function in SRCB)
- = 010 = Permission to initiate a function transmission (RCB for function contained in SRCB).
- = 011 = Reserved
- = 100 = Reserved
- = 101 = Available for local modification
- = 110 = Available for local modification
- = 111 = General Control Record (type indicated in SRCB)

TTTT = Record type identifier

- = 0000 = Control record
- = 0001 = Operator message display request

TTTT	=	0010	=	Operator command	
	=	0011	=	Normal input record	
	=	0100	=	Print record	
	=	0101	=	Punch record	
	=	0110	=	Data set record	
	=	0111	=	Terminal message routing request	
	=	1000 - 1100	=	Reserved for future expansion	
	=	1101 - 1111	=	Available for local modifications	

Sub-Record Control Byte (SRCB)

O	S	S	S	S	S	S	S

0 7

Usage: To provide supplemental information about a record

Bit Meanings: O = 1 (Must always be ON)

SSSSSSS = Additional record information - actual content is dependant on record type. Several examples are listed below --

SRCB for General Control Record

(character)

0 7

Usage: To identify the type of generalized control record

Bit Meanings: character = A = Initial terminal SIGN-ON

= B = Final terminal SIGN-OFF

= C = Print initialization record

= D = Punch initialization record

= E = Input initialization record

= F = Data set transmission initialization

= G = System configuration status

= H = Diagnostic control record

= I - R = Reserved

= S - Z = Available for local modification

SRCB for Print Records

O	M	C	C	C	C	C	C

0 7

Usage: To provide carriage control information for print records

Bit Meanings: O = 1 (Must always be ON)

M = 0 = Normal carriage control

= 1 = Reserved for future use

CCCCCC = Carriage control information

= 1000NN = Space immediately NN spaces

= 11NNNN = Skip immediately to channel NNNN

= 0000NN = Space NN lines after print

= 01XXXX = Skip to channel NNNN after print

= 000000 = SUPPRESS SPACE

SRCB for Punch Records

O M M B R R S S

0 7

Usage: To provide additional information for punch records

Bit Meanings: O = 1 (Must always be ON)

SS = Punch stacker select information

B = 0 = Normal EBCDIC card image

= 1 = Column Binary card image

M = 00 = SCB count units = 1

= 01 = SCB count units = 2

= 10 = SCB count units = 4

= 11 = Reserved

RR = Reserved for future expansion

SRCB for Input Record

O M M B R R R R

0 7

Usage: To provide additional information for input records

Bit Meanings: O = 1 (Must always be ON)

M = 00 = SCB count units = 1

= 01 = SCB count units = 2

= 10 = SCB count units = 4

= 11 = Reserved

RRRR = Reserved

SRCB for Terminal Message Routing Record

O	T	T	T	T	T	T	T

0 7

Usage: To indicate the destination of a terminal message

Bit Meanings: O = 1 (Must always be ON)

TTTTTTT = Remote system number ($1 \leq T \leq 99$)

TTTTTTT = Remote system group, as HASPGENed ($100 \leq T \leq 127$)

TTTTTTT = 0 = Broadcast to all remote systems

Function Control Sequence (FCS)

O	S	R	R	A	B	C	D	O	T	R	R	W	X	Y	Z
0							7	8							15

Usage: To control the flow of individual function streams

Bit Meanings: O = 1 (Must always be on)

S = 1 = Suspend <u>all</u> stream transmission (WAIT-A-BIT)

= 0 = Normal state

T = Remote console stream identifier

R = Reserved for future expansion

ABCD...WXYZ = Various function stream identifiers (oriented only to recipient)

- Normal print (or input) = A,B,C,...
- Normal punch streams = Z,Y,X,...
- Other functions = ...-...

NOTE - a bit on = continue function transmission

- bit off = suspend function transmission

Block Control Byte (BCB)

O	X	X	X	C	C	C	C

0 7

Usage: Transmission block status and sequence count

Bit Meanings: O = 1 (Must always be ON)

CCCC = Modulo 16 block sequence count

XXX = Control information as follows --

= 000 = Normal Block

= 001 = Bypass sequence count validation

= 010 = Reset expected block sequence count to CCCC

= 011 = Reserved

= 100 = Reserved

= 101 = Available for user modification

= 110 = Available for user modification

= 111 = Reserved for future expansion

12.11.3 MULTI-LEAVING In BSC/RJE

The previous sections have grossly outlined the specifications of a comprehensive, MULTI-LEAVING communications system. While the HASP support for programmable BSC workstations is completely consistent with the MULTI-LEAVING design, it does not utilize certain of the features provided in MULTI-LEAVING. These features not utilized include:

1. The transmission of record types other than print, punch, input, console and control is not supported.
2. The only general control record type utilized is the terminal SIGN-ON control.
3. Only SCB count units of 1 are utilized.
4. No support is included for column binary cards.

12.12 HASP 2770 RJE SUPPORT

12.12.1 2770 Configuration

The basic 2770 with standard keyboard and either EBCDIC or USASCII code is supported.

Optional supported devices include any combination of: 2502 Card Reader - Model Al or A2, 2213 Printer - Model 2, and 545 Output Punch - Model 3 or 4. The 2770 requires fixed attachment positions for the keyboard and printer. The card reader may be attached to either INPUT 2 or 3. The card punch must be attached to OUTPUT 2.

Optional features which are supported or will function with HASP support are: Buffer Expansion, EBCDIC Transparency, Keyboard Correction, Synchronous Clock and Transmit-Receive Monitor Print. The Multipoint Data Link Control feature must not be present. All other devices and features may be attached, but will not be supported.

12.12.2 Other Devices

Although HASP formally supports only the keyboard, card, and printer I/O devices listed previously in Section 12.12.1, the basic design of the IBM 2770 (i.e., media formats independent of transmission format) may make it possible for individual installations to use other I/O devices. This must be done only after careful analysis, design and testing by the customer and local IBM Representatives to establish the feasibility of the proposed device usage in the customer's environment. Refer to the SRL A27-3013, especially pages 2772-7, CU-2,3 and appropriate device sections. Also, the following descriptions of HASP's handling of input and output transmission blocks from and to the 2770 will aid in analysis of other device usage possibilities.

Input

Input blocks to HASP from any device on the 2770 are transformed into 80 character records of an OS job stream, according to one of the following two rules:

1. If the block is non-transparent, it is interpreted as one or more records of 80 or less data characters, each ended by an IRS character which does not become part of the record processed by OS. If the block contains only a single record, the ending IRS need not be present.

2. If the block is transparent, it is interpreted as one or more records of exactly 80 data characters, however the last or only record may be less than 80 characters. No record ending characters are recognized.

Transparent and non-transparent input blocks may be mixed, in any order, in any job or series of jobs transmitted to HASP. Proper handling of transparent input blocks is not dependent upon the setting of the RMTnn HASPGEN parameter describing the particular terminal.

Therefore, to use other input devices, the input medium and device must conform to the above. The device may be connected to any INPUT position as long as that position is switched on before transmission is initiated. Input which does not conform to these rules will cause unpredictable deblocking when received by HASP and probably error messages or incorrect results when processed by OS or the user's program.

If the input medium/device cannot produce an IRS record ending character or if control characters are used as data, then the transmission must be unblocked and/or possibly transparent. The processing program must handle as data any record ending character other than IRS which the medium/device may produce.

An input medium other than cards may not be suitable for the preparation and transmission of OS JCL cards (e.g., //ANY JOB ... up to //SYSIN DD *) which are required preceeding data in an OS input job stream. The keyboard may be used to transmit such cards, followed by data from the other device, using an operational procedure similar to that described for the keyboard and card reader on page 11 of the 2770 Operator's Guide.

Output

Output from HASP to the 2770 is in two forms: one intended for printing, the other for punching cards. These outputs are produced during OS execution of jobs by using the disposition SYSOUT= on DD cards. The decision to produce printed or punched output from a given SYSOUT class is controlled for the entire system by the HASPGEN parameters $$x, as described in Section 7.

Output block maximum length is 128 bytes, or 256 bytes if Buffer Expansion feature is indicated in the RMTnn HASPGEN parameter. Output records do not span transmission block boundaries. Each printed or punched output job is ended by an EOT transmission.

Printed Output

Printed output is always sent as non-transparent blocks. All data characters less than X'40' are translated to X'00', or if the &PRTRANS parameter is set to YES, all non-printing characters are translated to X'40'. The first block of a job contains the component selection character DC1. One or more variable length records are sent in each block. Each record begins with the two character ESC x carriage control sequence, has data characters up to the maximum specified for Printer Width in the RMTnn parameter, and ends with the IRS character.

The listing content for each job is the same as for all jobs printed by HASP: beginning and ending separator pages (number of separator lines controlled for all remotes in the system by the $TPIDCT parameter), HASP System Log, OS System Messages (JCL, etc.) and any printed SYSOUT data sets.

It is probably not very practical to direct printed output to another device for output data purposes, because of the inclusion of separator pages, messages, etc. The material could be directed to another medium (e.g., paper tape) for later listing offline or on another machine, however, because only the printer can be attached to the OUTPUT PRINTER position, HASP would have to be modified to use other than DC1 for print component selection. This would be a trivial one card modification if all 2770s in the system were configured and used the same way, but more difficult if not.

Punched Output

Punched output is sent as transparent blocks if the RMTnn parameter indicates that the Transparency feature is present. In this case, the component selection character DC2 is sent alone in a non-transparent block, at the beginning of the job. All other blocks are transparent and contain one or more records of exactly 80 data characters, without any record ending characters.

If transparency is not indicated by the RMTnn parameter, all punched output data characters less than X'40' are translated to X'00'. Only non-transparent blocks are sent, with the DC2 in the first block. Each block contains one or more variable length records. Each record contains 80 or less data characters and ends with the IRS character.

Punch job content is: separator card (described in Section 12.5.1), punched SYSOUT data sets, and one blank card at the end of the job. Blank cards may be produced at the end of each SYSOUT data set by some OS access methods, but these are simply transmitted as data by HASP. A second blank card at the end of each job is produced at the 545 Output Punch by a mechanical eject when EOT is received.

Punched output, except for separator and terminal blank cards, is pure data output whose content is controlled completely by the application program execution. Therefore, it may be practical to direct punched output to another device connected to the OUTPUT 2 position, or other positions if HASP is appropriately modified to use other than DC2 for punch component selection. If the non-transparency, variable length record, form of punched output described above is considered more desirable for the output device in question, HASP may be forced to produce it by omitting Transparency in the RMTnn parameter, even if the 2770 has the Transparency feature. This will not prevent the 2770 from transmitting transparent input blocks to HASP.

12.13 HASP EXECUTION BATCH SCHEDULING

This feature is a modification of normal HASP scheduling of jobs into logical partitions for execution by OS. The purpose is to allow the system to realize performance improvement by avoiding unnecessary OS Job Management overhead between "jobs" or "transactions" processed by an appropriate batch processing program; while maintaining the flexibility of having these "jobs" or "transactions" submitted to HASP independently, coming from possibly differing input sources, having differing printed and punched output routing, and with separate accounting for each job.

12.13.1 Batch Processing Program Characteristics

The processing programs to be used with the Batch Scheduling Feature of HASP may cover a wide variety of application areas such as:

- Compile and go debugging compilers
- File inquiry programs
- Hardware or software system emulators

However, a particular program to be used in the batch scheduling mode must have certain characteristics:

- It must read all user input from a single sequential data set.

- It must recognize a standard OS JOB card or its own control card to determine the beginning of a "job".

- It must recognize a standard OS null JCL card (// followed by 78 blanks) or its own control card to determine the ending of a "job".

The batch processing program will receive an actual end-of-file condition when a card having $$ in columns 1 and 2 is read while processing a "job". The program may continue to the next logical sub-file by a variety of technics. It may simply reset appropriate bits in OS I/O control blocks and continue reading or it may CLOSE the data set. The data set may then be re-OPENed to continue reading at the card following the $$ card.

It is desirable that the program process "jobs" or "transactions" of relatively short duration. If not, the saving in OS Job Management overhead between successive jobs may not be a large enough percentage of total job execution time to justify use of this feature.

12.13.2 Submission Of Batch Jobs

To use a batch processing program under the Batch Scheduling Feature of HASP, the user simply constructs jobs as follows:

The first card of each should be a standard HASP/OS JOB card, which includes a CLASS=x parameter, where x is the class (installation defined) indicating which batch program is to process the job. The accounting field is interpreted by HASP just as for non-batch jobs.

No other JCL is used. All other cards should be control cards, source cards, data cards, etc., as required by the batch program. These will be read by the batch program just as if they had been placed in a DD * data set and the batch program had been invoked by standard JCL. If the batch program requires it, each logical sub-file should be terminated by a card having $$ in columns 1 and 2.

12.13.3 Batch Scheduling Process

Special actions take place when HASP recognizes that a batch job has been selected for execution.

If the batch program is not already active in the logical partition for which the job was selected, then HASP generates and sends to the OS R/I an internal job which uses JCL from proclib (see 12.13.4) to invoke the program. The entire user job as submitted (JOB card, all other user input) followed by two null JCL cards added by HASP is allocated as an input data set to the batch program.

If the batch program is already active and simply waiting for another job, then HASP makes the input data set allocation as above and processing begins immediately, without any use of OS Job Management.

Job termination is detected by the batch program when it reads its own ending control card or one of the null JCL cards added by HASP. After writing any remaining SYSOUT data for the completed job, the batch program simply attempts to read ahead in its input file for another job. HASP detects this condition, temporarily forces the batch program into a wait state, and does its job termination actions for the job (flush output buffers, release input SPOOL space, queue job for printing, etc.). The batch program remains in the logical partition.

When a batch program is waiting in a logical partition, HASP job selection is altered. Instead of scanning for all classes eligible to execute in that partition, HASP first tries to start another job of the same class as the batch program still in the partition. If successful, processing can begin immediately as described above.

If no more jobs of the same class are available to execute, then all other job classes of the partition are scanned in order. If a job is found, HASP internally cancels the batch program and normal scheduling takes place using OS Job Management. If no jobs of the other classes are found, then the partition remains idle awaiting availability of a job in any of its classes. If a job becomes available in the class of the batch program still in the partition, processing begins immediately.

If a batch program ends (abend or normal return to OS), HASP detects this as a non-batch termination in the partition. OS Job Management will be used to re-invoke the batch program when another job for its class is selected.

Use of the operator commands $PI or $PIn will cause HASP to cancel an idle batch program when the partition(s) becomes drained.

12.13.4 Installing Batch Scheduling

The Batching feature is included in HASP by setting the &XBATCHC HASPGEN parameter equal to a list of job classes to be processed by the rules described above. The &XBATCHN parameter should also be set (see descriptions of these two parameters in Section 7).

Each batch class should be used to represent one batch processing program. Each batch class should be made eligible to execute in one or more logical partitions, by setting the &CLS(n) HASPGEN parameters or by use of the $T operator command.

The batch processing program for each class must be available in loadable form somewhere in the system.

For each combination of batch class and logical partition in which it may execute, there must be a procedure in SYS1.PROCLIB whose name is "nnnnncid"; where nnnnn are the five characters assigned to &XBATCHN, c is the particular batch job class (one of the list assigned to &XBATCHC), and id is the one or two character logical partition identification set by the parameters &PID(n). These procedures actually call the batch processing programs for each class and define all data sets other than the user input data set.

The procedures may either be single step or may have preliminary steps before the single step which processes the user jobs. That step must have a stepname of GO. The processing program invoked by this step must read its input from a ddname SYSIN or the procedure must refer to DDNAME=SYSIN on a DD card whose name is the one used for input by the processing program. It is recommended that the DCB parameter BUFNO=1 be included on any SYSOUT data sets in a procedure. This will help to insure that HASP has actually received all output produced by the batch program for a job or transaction, when the program is suspended while trying to read ahead to the next job.

If a given batch class is eligible to execute in more than one logical partition, the requirement for a separate procedure name for each class-partition combination may be satisfied by alias names of a single procedure, or by actual separate procedures which may specify different region sizes, work files, etc.

The following example shows the internal job which HASP would generate to intially load a program to process batch class X jobs, in a partition whose &PID(n)=3, assuming the default setting for &XBATCHN.

```
//$$$$$X3  JOB 1,SYS,MSGLEVEL=1
//FAKE EXEC $$$$$X3
//GO.SYSIN DD DATA,DCB=BUFNO=1
//
```

This job would call a procedure as shown. The following is an example of a procedure which an installation might use for a simple file inquiry program which reads inquiry input from SYSIN, interrogates a file, and prints responses to SYSPRINT.

```
//GO EXEC PGM=FINDPART
//SYSPRINT DD SYSOUT=A,DCB=(BLKSIZE=121,BUFNO=1)
//PARTFILE DD DSN=PARTFILE.MASTER,DISP=SHR
//SYSUDUMP DD SYSOUT=A
```

This procedure would be placed in SYS1.PROCLIB with the name $$$$$X3.

12.14 HASP OVERLAY PROGRAMMING RULES

The following comments summarize the rules for coding and using "overlayable code" in HASP. All rules apply to use of any control sections created by use of the $OVERLAY macro, even if the code so produced is optionally made permanently resident as part of the overlay build process. HASP Overlay does not use any overlay facility defined elsewhere in OS/360 documentation. More precise details of Overlay Macros syntax, Overlay Build process, Overlay Service and Overlay Roll internal logic are given in Sections 9.7, 10.2.2.3, 4.20 and 5.16.

12.14.1 Creating Overlay Control Sections

The beginning of a portion of HASP executable coding or tables to be made overlayable is indicated by the $OVERLAY macro. By convention, the name field begins with "HASP" and continues with up to four more characters. The fifth character (first after "HASP") usually indicates the Processor of which the overlayable code is a part; e.g., R for read, X for execution, P for print/punch, etc. A specific example is "HASPXJI1", the name of the first of two overlays used by the HASP Execution Processor for job initiation actions. The name coded with $OVERLAY will be defined at the first location coded by the programmer after the $OVERLAY and will be used to derive a name for the control section created.

The operands of $OVERLAY specify the priority for use of overlay resources and, in conjunction with the HASPGEN parameter &OLAYLEV, whether the code created is to be actually disk or main memory resident during HASP operation.

The $OVERLAY macro is a functional replacement for CSECT, USING, and BALR or L when creating a HASP overlayable control section. $OVERLAY creates an actual assembly control section and indicates local addressability in register BASE3. Overlay Service and Roll functions insure that the proper base value is loaded into BASE3 when an overlay section is being used.

An overlay control section's coding may be terminated and all effects of a previous $OVERLAY cancelled in one of two ways. Another overlay may be begun by a new $OVERLAY macro. Non-overlay coding may be resumed by DROPing register BASE3 and re-establishing an appropriate CSECT.

If it is desired to add more coding to a previously terminated overlay section, the actions in the following example must be performed. &xyz is a properly declared variable symbol. HASPabcd is the overlay name chosen by the programmer. Other symbols are defined in standard HASP assemblies. The second statement must be placed after the $OVERLAY defining the overlay section to be resumed, before another $OVERLAY is used.

```
HASPabcd  $OVERLAY  12,0      (original definition)
&xyz      SETC '&OSECT'
            .
            .
            .
&xyz      CSECT            (later additional code)
          USING  HASPabcd-OACEPROG+BUFDSECT,BASE3
```

12.14.2 Calling Overlay Routines

The three executable macros $LINK, $XCTL, and $LOAD cause an overlay routine to be made available for use in addressable memory. The single operand of each of these macros gives the name of the overlay to be used, either directly or by providing (in register form) the address of a $OCON macro which gives the name. The name referenced is that used with a $OVERLAY macro to create the overlay routine. The overlay control section ($OVERLAY and following code) may be in the same or a different HASP assembly as a macro which calls it.

The $LINK and $LOAD macros must be physically placed in non-overlay CSECTs and executed only when no other overlay routine is being used, i.e., nested calling of overlays is not defined. With $LINK, program control is eventually passed to the first instruction after $OVERLAY of the called routine. The address of the caller's next instruction is saved for later return. $LOAD returns control to the next instruction after $LOAD when the routine is available in memory.

$XCTL relinquishes use of an overlay routine, previously called by $LINK or $XCTL, and calls a new overlay routine which is entered as if called by $LINK. Return address saved by the original $LINK is not altered. $XCTL must always be executed when an overlay is in use, but may physically be in an overlay routine or in non-overlay coding, subject to the requirements of 12.14.3.

$RETURN and $DELETE both relinquish use of an overlay routine, which must be in use when they are executed. These macros have no operands; the routine released is the only one in use at the time. $RETURN causes control to pass to the next instruction after the $LINK previously executed by the Processor from non-overlay code. $RETURN, like $XCTL, may physically reside anywhere. $DELETE must physically reside in non-overlay code and is valid only after a routine was previously called by $LOAD. Control continues following $DELETE, after use of the overlay routine has been released.

Overlay routines may be called only by HASP Processors operating under the primary HASP TCB, HASP Dispatcher, and PCE control (see Section 5.1). Overlay routines may not be called in exits from the Asynchronous Post Processor (see Section 4.8).

12.14.3 Coding While Using Overlay Routines

On entry to an executable overlay by $LINK or $XCTL or after loading an overlay with $LOAD, the caller's registers R0-R7 and R9-R13 are preserved. However, registers BASE3 (same as R8 or WG in unmodified HASP), LINK, R15 and the condition code are destroyed and are not later restored. While an overlay routine is being used (after the execution of $LINK or $LOAD but before the execution of $RETURN or $DELETE), the program must not alter the value of register BASE3.

Coding in an overlay routine is "covered" by local addressability provided by $OVERLAY. Coding physically outside an overlay but referring to it (usual case after a $LOAD) must be "covered" by a USING like that in the example in 12.14.1. Other addressability (e.g., BASE1, BASE2) remains in effect if not dropped and may be used.

Program control may be transferred out of or into an overlay routine and its storage may be retrieved, as long as overlay control of that routine is in effect (has not been released by $RETURN, $DELETE, or $XCTL to a new routine) and proper addressability is maintained. References to locations in an overlay routine from physically outside the overlay at any other time are illegal.

Relocatable valued A or V type constants must not be physically coded in overlay routines. Such constants may be coded in non-overlay CSECTs and referenced from overlay routines. Relocatable A or V type literals may be coded if the literal pool containing them is not physically in an overlay routine. An A or V constant or literal containing an "un-paired" (see Assembly Language SRL) reference to a symbol defined in an overlay routine is always illegal, regardless of location.

When use of an overlay routine is released by $RETURN or $DELETE, only the LINK and BASE3 registers are destroyed. All other registers and the condition code are preserved as set prior to the execution of these macros.

Total size of all coding in an overlay routine must not exceed the value of the internal assembly variable &OLAYSIZ, currently set at 1024 bytes in unmodified HASP. An error message will be produced during the Overlay Build process for each routine which violates this restriction.

12.14.4 Overlay Location Independent Coding

Whenever a HASP Processor which is using an overlay routine executes $WAIT, regardless of the physical location of the $WAIT, the Overlay Roll Processor may pre-empt the Overlay Area for other use. When control is returned to the Processor following the $WAIT, the overlay routine may have been re-read from direct access, destroying all self-modification or temporary storage in the overlay, and may be in a different Overlay Area, making all address values relative to the overlay routine's location invalid (in registers or elsewhere).

The first effect above (destruction of temporary storage) is similar to the effect on single (non-re-entrant) temporary storages in non-overlay coding used by multiple Processors when $WAIT is executed. The effect on overlay storage may take place when only one PCE is using an overlay routine. Re-entrant temporary storage (e.g., in a PCE workarea) or re-construction from known values after $WAIT will avoid errors due to this possible "re-freshing" of overlay routines.

The second effect (changing overlay location) is, of course, peculiar to use of overlay routines. System Overlay Service and Roll logic automatically makes proper adjustments to registers BASE3 (overlay routine base value) and R15 ($WAIT re-entry address), if the $WAIT is physically in the overlay routine.

Other address values relative to an overlay routine are usually created in registers by use of instructions such as LA (with BASE3 as base), BAL, or BALR (the last two if physically in an overlay routine). These registers should be "relativized" prior to $WAIT by "SLR n,BASE3" instruction(s) and "absolutized" after $WAIT by "ALR n,BASE3" instruction(s). Equivalent techniques may be created for other coding situations.

Certain HASP macros which call services subroutines represent a "hidden" possible $WAIT. They must be treated as equivalent to $WAIT in all cases previously described. Specifically, any macro for which the keyword parameter OLAY=YES is defined (see Section 9) represents a hidden $WAIT, regardless of physical location. The OLAY=YES is coded only if the macro physically exists in an overlay routine. Macro expansion and service subroutine exit coding handle possible adjustment of the LINK register. The services subroutines assume that all parameter address values (in R0, R1, or R15) are not relative to an overlay routine. Other addresses relative to an overlay routine must be adjusted before and after the service macro call by the caller.

The $WTO macro is a special case. It represents a hidden $WAIT unless WAIT=NO is coded. If coded physically in an overlay routine, WAIT=NO must be coded. It may be coded physically outside an overlay routine without WAIT=NO, but then registers must be treated as for macros which have OLAY=YES defined.

12.15 HASP WITH OS CONSOLE SUPPORT

The following sections describe the HASP routines which are provided for the OS console support interface selected by specifying the HASPGEN parameter &NUMCONS=0 (see Section 7.1).

12.15.1 General Description

The functions included in HASP to provide an interface with the OS console support are in the following major areas:

- Initialization procedure
- SVC 34 processing
- SVC 35/36 processing
- WTO subtask

Each of these areas is described in greater detail in the remaining sections of this appendix.

The combined overall functions of the interface is to allow operator commands (both OS and HASP) to be entered from any OS supported console input device without special operator action and to display the commands and associated information in accordance to a combined OS and HASP criteria. In addition, the unique HASP features of abbreviated replies to WTORs and the HASP System Log of WTO messages as part of the programs printed output are included (subject to the restrictions noted in the description of the &NUMCONS variable in Section 7.1).

12.15.2 Initialization Procedure

Preparation for the &NUMCONS=0 option at the time HASP is invoked includes the following functions in the INIT modules:

1. Information concerned with the UCM base is extracted and stored in the resident CON module. Included are: address of UCM save area; TCB address of communication task; address of UCM base fields containing address of first UCM entry, size of each UCM entry, address of last UCM entry; contents of Mode flag byte from UCM base. The source of this information is OS release dependent. See the OS MVT Supervisor PLM for additional information on the UCM.

2. The address of the HASP TCB is stored in CON to facilitate OS POSTing of the HASP task.

3. The servicing of SVC 35 and, conditional on the HASPGEN parameter &WTLOPT, SVC 36 by OS is diverted to HASP by changing the contents of the SVC table to enter the XEQ module IOS interface section and, subsequently, the CON module code $WTOSVC. If the OS System is MFT, the SVC table is changed to indicate a Type 3 resident routine. If the OS System is MVT, the SVC table is changed to indicate a Type 2 SVC. In both cases the original SVC table contents are saved in the HCT prior to the indicated changes. Note: The SVC table for MFT must contain four byte entries in order to indicate a Type 3 resident SVC.

4. An ATTACH is issued to the BR1 module which executes a branch to the address contained in register 1. Prior to the ATTACH, register 1 is set to the entry point of the HASP WTO subtask ($HASPWTO). Register WA is set to the address of an ECB ($WTOECB) which is used to coordinate the activities of HASP with the subtask. See Section 12.15.5 for further details.

5. An error message, indicating the completion code provided by the return from ATTACH, is issued if the ATTACH was unsuccessful. Control is passed to the HASPIOVD segment of INIT to continue processing.

12.15.3 SVC 34 Processing

$MGCRSVC, a section of code contained in the CON module, is entered whenever an XCTL to IGC0403D is detected by the HASP LINK/XCTL interface. The functions performed by $MGCRSVC are:

1. Immediate return to perform the XCTL if the SVC 34 was issued by HASP.

2. Tests for possible HASP format abbreviated reply to outstanding WTOR. If the first character is numeric, the abbreviated reply process is invoked to expand the HASP form to a form acceptable to OS. Control returned to process the XCTL with expanded reply.

3. If the first character is non-numeric, an explicit test is made for a "$" which identifies HASP commands. Control returned to process the XCTL if the first character is non-numeric or not a "$".

4. If the first character is a "$", a test for at least one CMB which is not being used to process HASP commands is performed using a counter ($COMMCT) maintained in the HCT. If all CMBs (except one) are being used to process commands or if no CMBs are available on the free queue ($FREEQUE), then control is returned to process the XCTL. The command is subsequently

rejected by OS as invalid.

5. If MCS is being used in the OS system, the authorization code for the device indicated by the UCMID contained in the low order byte of R0 is extracted from the UCM and converted to the HASP restriction level. Reference the HASP COMM Processor for additional information on the OS authorization code and HASP restriction level relationship.

6. The contents of the input buffer are copied to the acquired HASP CMB and the CMB is queued for the HASP command processor using the $COMMQUE pointer in the HCT. The command processor is $POSTed for work and HASP is OS POSTed.

7. The resume PSW in the SVRB of the issuer of the XCTL to IGC0403D is changed to point to CVTEXIT in the CVT. The current SVRB is terminated by issuing an SVC 3 which eventually causes the whole process to be ignored by OS.

12.15.4 SVC 35/36 Processing

$WTOSVC, a section of the CON module, is entered from the SVC SLIH via the Execution Processor IOS interface routine whenever an SVC 35 or, optionally, SVC 36 is encountered. This section of code operates as a Type 2 SVC and accomplishes the following functions:

1. XCTLs to the real first load of SVC 35 (IGC0003E) if the WTO was issued by the HASP subtask ($HASPWTO).

2. Saves registers in the current SVRB extended save area and tests input for WTO or WTOR.

3. If WTOR, adjusts input pointer (R1) to beginning of message and proceeds as follows (WTO processing):

4. An internal table is used to compare the first eight bytes of the input message and, if a match occurs, special processing is invoked through a corresponding routine. The usual function of the special processing is to bypass the display of redundant system messages.

5. A search of the Execution Processor PCEs is made to locate the JCT for the job issuing the SVC 35 or 36. The TIOT and associated job name is used for the search. If a match is not found, control goes to the real first load of the SVC 35/36.

6. If a CMB is available, the subroutine HASPCBUF is entered to copy the message to the HASP Log for the particular job; HASP is OS POSTed and control passed to the appropriate OS module: IGC0003E or IGC0003f for SVC 35 or 36 respectively.

7. If a CMB is not available, the caller is forced into an OS WAIT condition. The forced WAIT is conditional: the Communication Task, DAR and SIRB controlled routines are excluded. In addition, if no PQE (used to retain the TCB and define an ECB for the routine $FREEMSG to OS POST) is available, the caller is not forced into a WAIT condition and the message is not entered into the HASP Log.

12.15.5 HASP WTO Subtask

The HASP OS WTO interface ($HASPWTO) which operates as an ATTACHed subtask to HASP is responsible for the processing of all WTO messages generated by HASP processors as the result of $WTO macros. $HASPWTO is implemented as a subtask in order to allow the normal OS function of delaying the execution of a task due to predetermined buffer limits. The "delaying" of the task, under these circumstances, is in the form of an ENQ and WAIT procedure which causes the task to be non-dispatchable until sufficient resources (buffers) are available to process the WTO. It is undesirable for HASP proper to be forced into a WAIT state but it is tolerable for the $HASPWTO subtask to be subjected to a forced WAIT.

$HASPWTO is assembled as part of the CON module and is executed as a task via an ATTACH issued at HASP initialization. The overall logic of this task is:

1. The initial entry establishes local and HASP addressability, POSTs a synchronization ECB for INIT and then WAITs for work using a communication ECB which is POSTed by the CON module routine $WQUEBUF.

2. When the communication ECB ($WTOECB) is posted, $HASPWTO examines the CMB active queue ($BUSYQUE) for messages to be sent to the OS console routines via a WTO. All messages on the queue except those flagged for remote processing are processed by $HASPWTO. If the queue is empty, $HASPWTO WAITs for the next POST of $WTOECB.

3. If the message selected from the active queue contains a UCMID byte, then the MCSFLAGS field of the WTO calling sequence is set to indicate R0 contains the UCMID and, eventually, R0 is loaded with the UCMID. This feature allows responses to HASP commands to be returned to the indicated console.

4. The routing code field of the WTO calling sequence is used for non-UCMID CMBs. The HASP logical console bit indicators (contained in the CMB) are used to translate to the equivalent OS routing codes based on the following table:

HASP	OS Function
LOG	MASTER CONSOLE INFORMATIONAL
ERROR	SYSTEM/ERROR MAINTENANCE
UR	UNIT RECORD POOL
TP	TELEPROCESSING CONTROL
TAPE	TAPE, DISK LIBRARIES TAPE, DA POOLS
MAIN	MASTER CONSOLE, ACTION AND INFORMATIONAL

5. The message is copied from the CMB to an internal buffer corresponding to the WTO format. If routing codes are being used, the description code field is set to indicate a system status message.

6. The CMB is returned to the available CMB queue ($FREEQUE) using the $FREEMSG subroutine. If $FREEMSG returns with a condition code = 0, than an OS POST is issued to the HASP ECB ($HASPECB), otherwise no POST is given.

7. The WTO is issued for the copied CMB using either the UCMID or the routing and descriptor code options. The procedure described starting at step 2 is repeated.

12.16 MULTIPLE DEVICES ON MULTI-LEAVING REMOTES

If a HASP System includes MULTI-LEAVING RJE support (&NUMLNES > 0 and &BSCCPU=YES) and if any remote terminal to be supported has multiple devices (i.e., more than one reader, printer or punch), then the following considerations should be reviewed before performing HASPGEN and RMTGEN for that configuration.

12.16.1 RMTGEN Considerations

The appropriate parts of Section 7 describe how to specify support for a second (or third, etc.) reader, printer, or punch when performing RMTGEN for the various types of MULTI-LEAVING remote workstation programs.

12.16.2 HASP Processor Considerations

It may be necessary to increase the value(s) of the HASPGEN parameters &NUMTPPR, &NUMTPPU, and &NUMTPRD to allow concurrent operation of all remote devices in the total system.

For example, if &NUMLNES=3 and the default value &NUMTPPR=&NUMLNES is taken, then the HASP System can only support three concurrent remote print operations. If all three lines are active and one of the three active remotes has two printers, then unless &NUMTPPR is increased to four, one of the four possible concurrent remote print operations may be delayed until a print operation on another remote comes to the end of a job.

The decision to increase these parameters and by how much, depends on the total remote configuration and an estimate of how many active remotes will usually be doing the same stage of job processing.

12.16.3 HASP Remote Device Considerations

HASP generates a Device Control Table (DCT) for one of each type of device (reader, printer, and punch) on each remote terminal known to HASP (RMT01 through RMTnn where &NUMRJE=nn).

If a remote terminal has more than one of each type of device, then a DCT for each such additional device must be generated.

Each additional DCT must be specified on a card of the following format:

```
          $RMTDCT type,device                     -serial-
```

Values for the above card should be chosen from the following table:

	type	device	serial
readers	RJR	RMnn.RDm	N0730nnm
printers	RPR	RMnn.PRm	N0732nnm
punches	RPU	RMnn.PUm	N0736nnm

where "nn" is the remote number (same as in the RMTnn HASPGEN parameters but with a leading zero omitted in device) and "m" is the device number which must be 2 or greater, up to a maximum of 7.

All the above cards describing additional devices for all remotes in a system must be placed in ascending order by serial number and added to the source module HASPINIT using the HASPGEN Update facility described in Section 10.1.3. The following example shows how to generate a second printer DCT for remote 2 and a second reader DCT for remote 5.

```
Columns
1         10     16                                   73    80
---------------------------------------------------------------
./        CHNGE HASPINIT
          $RMTDCT RJR,RM5.RD2                         N0730052
          $RMTDCT RPR,RM2.PR2                         N0732022
```

(LAST PAGE)

SR27-9721-0

OS/HASP Volume 2 Supplementary Course Material Printed in U.S.A. SR27-9721-0

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]