Program Library
IBM Japan, Ltd.
Systems Engineering Dept.
14, 1 Chome Nagata-cho
Chiyoda-ku
Tokyo, Japan

Canadian Program Library
IBM Canada Ltd.
Department 960
5 Yorkland Boulevard
Willowdale, Ontario
Canada

European Program Library
IBM France
23, Allée-Maillasson
F.92-Boulogne-Billancourt
France

Société Anonyme Au Capital de
620.256.000 F-R.C.
(Seine 55B-11 846)

Program Information Dept.
IBM Corporation
40 Saw Mill River Road
Hawthorne, New York 10532
United States

South American
Program Library
IBM do Brasil, Ltda.
Avenida Presidente
Vargas 642, 4 Andar
Caixa Postal 1830-ZC-00
Rio de Janeiro, Brazil

South Pacific
Program Library
IBM Australia, Ltd.
Box 3318 G.P.O.
Sydney, N.S.W.
Australia

June 26, 1972

Memorandum to: Users of the HASP II System (360D-05.1.014)

Subject: Transmittal of Version 3, Modification Level 1 of 360D-05.1.014

This letter transmits Version 3, Modification Level 1 of the HASP II System, 360D-05.1.014.

The program materials needed to update this program to Version 3, Modification Level 1 are enclosed.

The Basic materials distributed with this letter are:

1. An update to HASP II System Manual (replacement pages)
2. A replacement Memorandum to Users for HASP II System Manual
3. A complete replacement of the Basic machine readable material on one Distribution Tape Reel (DTR) recorded at 9-track 800 or 1600 bpi, or one 7-track 800 cpi (Data Conversion Feature required)

If you are a user of the Optional program material, it consists of:

1. A complete replacement of the Optional machine readable material consisting of 138 - 96 column cards as a "starter system" for a remote System/3.

This release of HASP corrects virtually every known problem in Version 3, Modification Level 0 of HASP. This sytem includes all PTFs applicable to HASP Version 3.0 through DPA5427. Following is a list of the more significant maintenance items.

The restrictions prohibiting any extended use of the 3211 Forms Control Buffer have been removed.

Rotational Position Sensing for 3330 and 2305 devices is now a HASPGEN option.

Previously announced support for the 3505 Card Reader and the 3525 Card Punch is included.

The restriction prohibiting operator control of HASP job flow based on OS jobnames has been removed.

Multi-tasking jobs are no longer excluded from the execution dynamic priority group.

The restriction prohibiting restart of the HASP execution phase of a job has been removed.

The operator is no longer prohibited from changing a printer's FCB or UCS when the printer is stopped for forms mounting.

For OS controlled console support, the operator's replies to WTORs are no longer omitted from the HASP System Log.

Optionally, the user can now define names other than SPOOLn for HASP spooling volumes.

HASPGEN efficiently accepts both the IEBUPDTE and IEBUPDAT formats of modification cards and sequence checks them.

HASP command authority has been extended to local input devices and is controlled by the central operator.

An optional backspace character can now be defined to support OS controlled consoles with no backspace key.

HASP Remote Job Entry maintenance items:

The 3780 Data Communications Terminal is now supported by HASP.

The Space Compression/Expansion feature of 2770 and 3780 terminals is supported.

Any of the three optional buffer sizes for 2770 terminals may now be specified. Current 2770 users must respecify RMTnn parameters prior to HASPGEN, according to the new definitions on page 388.

HASP RMTGEN will optionally punch an 80-column card deck for System/3 remote terminals without 5424 card readers.

This modification is supported on current OS releases.

This memorandum should be added to your program package for future reference.

The HASP II System has Programming Service Classification A.

Any discrepancy between the material received and the material listed above, or any errors in reproduction, should be reported to the Manager of the Program Library providing your programming systems.

cc: IBM Systems Engineering Managers (no enclosures)
IBM Field Engineering Managers (no encloures)

Program Information Department

Japan Program Library
IBM Japan, Ltd.
Systems Engineering Dept.
14, 1 Chome Nagata-cho
Chiyoda-ku
Tokyo, Japan

Canadian Program Library
IBM Canada Ltd.
Department 960
5 Yorkland Boulevard
Willowdale, Ontario
Canada

European Program Library
IBM France
23, Allée-Maillasson
F.92-Boulogne-Billancourt
France

Société Anonyme Au Capital de
620.256.000 F-R.C.
(Seine 55B-11 846)

Program Information Dept.
IBM Corporation
40 Saw Mill River Road
Hawthorne, New York 10532
United States

South American
Program Library
IBM do Brasil, Ltda.
Avenida Presidente
Vargas 642, 4 Andar
Caixa Postal 1830-ZC-00
Rio de Janeiro, Brazil

South Pacific
Program Library
IBM Australia, Ltd.
Box 3318 G.P.O.
Sydney, N.S.W.
Australia

June 26, 1972

Memorandum to: Recipients of HASP II (360D-05. 1. 014)

Subject Replacement Memorandum for HASP II System Manual

The attached memorandum replaces the memorandum currently attached to the HASP II System Manual.

Attachment

Program Information Department

a Program Library
M Japan, Ltd.
Systems Engineering Dept.
14, 1 Chome Nagata-cho
Chiyoda-ku
Tokyo, Japan

Canadian Program Library
IBM Canada Ltd.
Department 960
5 Yorkland Boulevard
Willowdale, Ontario
Canada

European Program Library
IBM France
23, Allée-Maillasson
F.92-Boulogne-Billancourt
France

Société Anonyme Au Capital de
620.256.000 F-R.C.
(Seine 55B-11 846)

Program Information Dept.
IBM Corporation
40 Saw Mill River Road
Hawthorne, New York 10532
United States

South American
Program Library
IBM do Brasil, Ltda.
Avenida Presidente
Vargas 642, 4 Andar
Caixa Postal 1830-ZC-00
Rio de Janeiro, Brazil

South Pacific
Program Library
IBM Australia, Ltd.
Box 3318 G.P.O.
Sydney, N.S.W.
Australia

June 26, 1972

Memorandum to: Recipients of HASP II (360D-05.1.014)

Subject: Transmittal of Version 3, Modification Level 1 of 360D-05.1.014

The program materials you have ordered are enclosed. The following describes the contents of the Basic and Optional program packages.

Basic program material consists of:

1. The enclosed HASP II System Manual.

2. Machine-readable material on one 9-track 800 or 1600 bpi, or 7-track 800 cpi (Data Conversion Feature required) Distribution Tape Reel (DTR). For a description of the tape contents see the HASP II System Manual.

If you have ordered the Optional program material, it consists of:

1. A deck of 138 - 96 column cards. This deck is a "starter system" for the HASP MULTI-LEAVING Remote Job Entry support for the IBM System/3 to allow a customized workstation program to be transmitted to the Remote System/3. See the HASP Remote Terminal Operator's Guide for the System/3 for instruction on the use of this deck.

Installations ordering this program to obtain the HASP MULTI-LEAVING workstation programs for use with non-HASP systems should refer to Section 10.4 of the HASP SYSTEM Manual for procedural information.

The HASP-II SYSTEM has Programming Service Classification A. If, in the future, a new release is made available for this program, the period that Version 3, Modification Level 1 will remain "current" for programming service purposes will be specified at the time of the new release.

HASP, Version 3 Modification Level 1 incorporates all PTFs applicable to HASP Version 3 Modification Level 0 through DPA5427. No PTF before and including DPA5427 should be applied to Version 3, Modification Level 1. Your local IBM representative will discuss the standard error reporting (APAR) procedure.

This program has been registered by system type and is listed under the name and address shown on your order. Program modifications, as and when made by IBM will be sent to this same address. Should there be a change in your system type or in your address, we would appreciate your notifying your local IBM branch office.

Any discrepancy between the material received and the material ordered, or any errors in reproduction should be reported to the Manager of the Program Library providing your programming systems.

Enclosures

Program Information Department

a Program Library
IBM Japan, Ltd.
Systems Engineering Dept.
14, 1 Chome Nagata-cho
Chiyoda-ku
Tokyo, Japan

Canadian Program Library
IBM Canada Ltd.
Department 960
5 Yorkland Boulevard
Willowdale, Ontario
Canada

European Program Library
IBM France
23, Allée-Maillasson
F.92-Boulogne-Billancourt
France

Société Anonyme Au Capital de
620.256.000 F-R.C.
(Seine 55B-11 846)

Program Information Dept.
IBM Corporation
40 Saw Mill River Road
Hawthorne, New York 10532
United States

South American
Program Library
IBM do Brasil, Ltda.
Avenida Presidente
Vargas 642, 4 Andar
Caixa Postal 1830-ZC-00
Rio de Janeiro, Brazil

South Pacific
Program Library
IBM Australia, Ltd.
Box 3318 G.P.O.
Sydney, N.S.W.
Australia

June 26, 1972

Memorandum to: Recipients of the HASP II System (360D-05.1.014) Version 3, Modification Level 1

Subject: New and Replacement pages for the HASP II System Manual

Attached are new and replacement pages for the HASP II System Manual. These pages have been reproduced in such a manner as to easily replace their corresponding pages in the HASP II System Manual.

Program Information Department

T H E

H A S P

S Y S T E M

FEBRUARY 26, 1971

TYPE III PROGRAMS WITH SERVICE A CLASSIFICATION

Type III programs which were given Service A Classification, perform functions which may be fundamental to the operation and maintainance of the user's system. These programs have not been subjected to formal test by IBM.

Until reclassified, IBM will provide for these Type III programs the following: (a) Central Programming Service including design error correction and automatic distribution of corrections; (b) Field Engineering Programming Service including design error verification, Authorized Programming Analysis Report (APAR) documentation and submission, and application of Program Temporary Fixes or development of an emergency by-pass when required.

IBM does not guarantee service results or represent or warrant that all errors will be corrected. The user is expected to make the final evaluation as to the usefulness of these programs in his own environment.

THE FOREGOING IS IN LIEU OF ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

PREFACE

This document contains complete instructions for the generation and use of the HASP SYSTEM. Also included is detailed information concerning the internal structure and implementation techniques of HASP to serve as an aid to systems programmers. The various HASP OPERATOR'S GUIDES, intended for use as removable sections, are included as Section 11 of this manual.

MAGNETIC TAPE KEY

BASIC

This volume contains two files as described below.

File 1 - Assembled object decks and JCL necessary to perform a HASPGEN (refer to Section 10 of this manual for information concerning use of this tape).
Records - 428; Characters/block - 80;
Records/block - 1; Blocks/file - 428.

File 2 - Source decks for HASP-II, Version 3.1.
Records - 52,705; Characters/block - 1600;
Records/block - 20; Blocks/file - 2636.

*Optional

System/3 users only - 138 96-column cards. This deck is a "starter system" for the HASP MULTI-LEAVING Remote Job Entry support.

*Optional material will be forwarded only when specifically requested.

TABLE OF CONTENTS

1.0 INTRODUCTION

The HASP SYSTEM operates as a compatible extension to the MFT or MVT options of the Operating System for System/360 and System/370 to provide specialized supplementary support in the areas of job management, data management, and task management.

HASP appears as a transparent "front-end" processor to OS to, via the SPOOLing functions normally associated with OS input readers and output writers, act as an automatic scheduler and operator of OS. Because of this relationship between HASP and the Operating System, various other functional, performance and operational benefits can be included in HASP.

The use of HASP offers an installation the following advantages:

- IMPROVED PERFORMANCE - In many cases, because of the singular, specialized use of resources by HASP, system performance may be improved. Any improvement is dependent upon the configuration and job mix and can only be determined by actual measurement. (See Section 2 of this manual for additional details.)

- IMPROVED OPERATIONAL PROCEDURES - HASP acts as an automatic interface between the operator and OS, to perform various OS control functions previously done directly by the operator. Readers, Writers and Initiators in OS are started and scheduled automatically by HASP. Also, many additional operator commands for controlling job flow and device operation are provided by HASP. (See Section 11 of this manual for additional details.)

- INCREASED SYSTEM FUNCTION - The use of HASP provides certain functions which are not otherwise available. These include dynamic task ordering based upon CPU - I/O characteristics (see Section 2 for additional details); the inclusion of relevant console messages in each job's output (see Section 7 for additional details); the capability of any job to introduce another job into the HASP queue via an internal reader (see Section 12.10 for additional details); an execution batching facility to pass jobs directly to a processing program such as a one-step monitor (see Section 12.13 for additional details); many additional operational control functions (see Section 11 for additional details); a priority aging technique (see Section 4.22 for additional details); a pre-execution volume fetch facility (see Section 11 for additional details); and various other functional enhancements.

- RESOURCE REDUCTION - Because of the dynamic direct-access allocation techniques utilized by HASP, installations may, in general, reduce the number of direct-access volumes required

for SPOOLing functions as compared with a non-HASP SYSTEM. The size of the OS SYS1.SYSJOBQE data set may also be reduced since all job queueing is performed by HASP.

Certain installations may actually reduce system main storage requirements (increase problem program space available) by adding HASP to their system because of the OS functions replaced by HASP. In any case, the space required for the HASP partition or region will be at least partially compensated for by the elimination of duplicate functions.

- LOW-ENTRY, HIGH-PERFORMANCE REMOTE JOB ENTRY - For a nominal increase in the size of HASP, an installation can utilize the HASP RJE support for a wide variety of workstation devices. Support for Binary-Synchronous, CPU workstations employs an advanced technique called MULTI-LEAVING which provides for simultaneous operation of all devices on a remote workstation. A subset of the HASP operator command language is provided to all remote sites. Workstation programs are supplied for all supported CPU workstations. (See Section 12.11 for additional details.)

- TRANSPARENT OPERATIONS - HASP is, in general, transparent to both the Operating System and to user programs. Although a special SYSGEN is required, no actual modifications to OS are required to utilize HASP. Thus, the same generation of OS may be interchangeably used with or without HASP. Because of this transparency, HASP is generally independent of the OS release level or options selected and can be used as a stable base for local modifications to customize for local operational requirements.

  Most standard jobs which operate under OS can be run with absolutely no change in a HASP environment. Most installations can, therefore, implement HASP with little or no changes to current user programs.

2.0 GENERAL DESCRIPTION

HASP is a specialized program which operates in the same CPU with OS/360 to perform the peripheral functions associated with batch job processing.

HASP is loaded as a normal OS/360 program and upon gaining control enters the supervisor mode via a special SVC routine. Control of all on-line unit record devices is assumed, the designated intermediate storage direct-access device(s) are initialized and job processing begins. The basic interface between HASP and OS/360 is through the Input-Output Supervisor (IOS). The entry point of IOS is modified so that Input-Output requests to unit record devices are diverted to HASP rather than being physically executed by IOS. Jobs which have been previously read from physical input devices by HASP can now be passed to OS by simulating a successful completion of the intercepted I/O request. In a similar manner, print and punch output from jobs being processed by OS/360 can be intercepted and queued on intermediate storage for later transcription to unit record devices.

HASP has four major processing stages which account for its four major external functions. These are:

1. INPUT STAGE - This stage reads jobs simultaneously from an essentially unlimited number of various types of on-line card readers, tapes and remote terminals into the system. These jobs are then entered into a priority queue by job class to await processing by the next stage.

2. EXECUTION STAGE - This stage removes jobs based upon priority and class from the queue established by the Input stage and passes those jobs to OS/360 for processing. Input cards are supplied as required to the executing program and print and punch records are received and written onto HASP intermediate storage. This stage can simultaneously control an essentially unlimited number of jobs being processed by OS/360. At the completion of a job, it is placed in a queue to await processing by the next stage.

3. PRINT STAGE - The purpose of this stage is to transcribe the printed output generated by jobs in the previous stage to printers. An essentially unlimited number of various types of printers and remote terminals can be operated simultaneously.

4. PUNCH STAGE - This stage transcribes the punch output generated by jobs in the execution phase to punches. An essentially unlimited number of various types of punches and remote terminals can be operated simultaneously.

All of the these processes are controlled by re-entrable code so that no additional code is required to support multiple, simultaneous functions. Since all of the above functions can occur simultaneously and asynchronously, a continuous flow of jobs may pass through the system.

Following are some of the more significant algorithms employed by HASP to improve function and performance:

- SPECIALIZED DIRECT-ACCESS STORAGE ALLOCATION

  HASP, through the use of an allocation bit map in main storage, dynamically allocates space for intermediate storage on a record basis, within definable track groups, for jobs. The use of this technique offers the following advantages:

  1. Disk-arm motion and interference is minimized by dynamically allocating space based upon the position of the access mechanism.

  2. Disk area fragmentation is automatically eliminated by allocation of the smallest possible increment of space.

  3. The data for a single data set can be spread across multiple direct-access volumes. In addition to further optimizing arm motion, this capability allows for the simultaneous use of multiple selector channels to increase the data rate for a given job.

  4. Since space is allocated only when required, there will be no unused space as a result of over estimated output requirements.

  5. The release of previously used space is accomplished by a simple algorithm which requires no I/O operations.

- UNIT RECORD DEVICE COMMAND CHAINING

  While operating any reader, printer or punch, rather than handling each record separately, HASP constructs a chained sequence of channel command words to pass to the channel. Thus, instead of the overhead of an EXCP and the ensuing interrupts for each record transmitted, only one EXCP and associated interrupt is required for a series of records. For example, when reading a job into the system, HASP might chain 40 commands together to instruct a card reader. This

would cause the next 40 cards to be read into memory without requiring the execution of any CPU instructions.

- TRANSPARENT BLOCKING

All input, print and punch for every job is automatically blocked by HASP to improve performance. Since all deblocking is also done by HASP, any program even if designed to operate with unblocked records can benefit from the blocking. Also, because all blocking and deblocking is done by HASP, problem programs require buffers only the size of a single card or line. This can reduce a program's partition or region requirement by several thousand bytes over normal full track blocking.

- DYNAMIC BUFFER POOL

HASP maintains a dynamic area of memory which is allocated as required. This technique allows not only multiple data sets of a job, but multiple jobs to share this area, thereby insuring optimum use of storage.

- EXECUTION TASK MONITOR

A significant item contributing to system performance is the correct ordering of dispatching priorities of jobs in relation to their CPU-I/O utilization ratios. It is obviously straightforward to manually set the dispatching priorities of two jobs, one of which is completely I/O-bound and the other completely CPU-bound. It becomes more difficult to determine relative priorities of multiple jobs with varying degrees of CPU-I/O ratios and impossible to determine priorites for multiple jobs which constantly change from CPU to I/O bound or vice versa.

HASP provides a feature which, at frequent intervals, examines each eligible job and dynamically re-orders the OS dispatching chain based upon the measured CPU-I/O characteristics of the jobs during the previous interval. This capability relieves an installation of the responsibility of attempting to assign job dispatching priorities while insuring the optimum ordering of jobs being processed by the Operating System.

(The remainder of this page intentionally left blank.)

3.0 HASP STRUCTURE

The primary goal in the design of any execution support system such as HASP must be the efficient manipulation of the various resources required for processing. The first design steps must then include the determination of what resources will be required and the careful application of sound programming design techniques to achieve an efficient and consistent solution to the allocation of these resources.

A study would reveal that HASP requires the following resources:

1. Main Storage
2. Direct-Access Space
3. Input/Output Units
4. Central Processing Unit Time
5. Input/Output Channel and Unit Time
6. Programs
7. Jobs
8. Interval Timer

Since these resources are essentially the basic facilities provided by the Operating System, it would at first seem that these facilities would be sufficient to meet the requirements of HASP. Further studies show, however, that the philosophies of the Operating System's services are not always consistent with the design requirements of a system such as HASP.

For instance, the main storage services provided by the Operating System are very flexible and comprehensive but fail to meet the requirements of HASP in the following areas:

- As requests for main storage are serviced, memory becomes fragmented in such a way that eventually a request for storage cannot be serviced for lack of contiguous memory even though the total amount of storage available far exceeds the requested quantity.

- As the amount of available storage decreases, the requestor becomes more susceptible to being placed in an OS WAIT state or being ABENDed. These conditions are both intolerable to HASP.

- The primary use of main storage in HASP is for buffering space for input/output purposes. These input/output purposes require that an Input/Output Block be associated with each segment of main storage which the Operating System Main Storage Supervisor, only naturally, does not provide. This means that HASP would have to construct such a block for each main storage segment it required.

In a similar fashion the Direct-Access Device Space Manager (DADSM) provides flexible and comprehensive services for normal job processing requirements but fails to meet the requirements of HASP in the following areas:

- Because of the data set concept employed by DADSM, the "hashing" or "fragmentation" problem described above also impacts the allocation of direct-access space.

- The data set concept complicates the simultaneous allocation of storage across many volumes (for selector channel overlap).

- The DADSM limit of extents per volume tends to cause volume switching, and the associated time delays are intolerable to HASP.

- DADSM consists of non-resident routines which must be loaded for each direct-access space allocation service. Because of the frequent allocation requirements, the associated overhead involved in the loading of these routines would degrade the performance of HASP to a certain extent.

Since the unit-record input/output units which the scheduler allocates to the jobs being processed in other partitions must be available for use by HASP, HASP must be responsible for the allocation of its own input/output units.

The Operating System Task Supervisor is responsible for the allocation of Central Processing Unit (CPU) time to all tasks in the system. The different functions of HASP (reading cards, printing, punching, etc.) could be defined as individual OS tasks except for the following considerations:

- Defining each function as a separate task would prohibit HASP from being used with anything other than a variable-task system.

- Inter-task communication and synchronization is many times more complex than intra-task communication and synchronization.

The Operating System Input/Output Supervisor is responsible for the allocation of all input/output channel and unit time. It completely meets all requirements and is used by HASP for all input/output scheduling.

The Operating System Interval Timer Supervisor provides complete interval timer management services but limits these services to one user per task. Since HASP has many functions which have simultaneous interval timer requirements, an interface must be provided which will grant unlimited access to the OS Interval Timer Supervisor.

The following sections describe, in detail, the allocation techniques and algorithms used in HASP to provide the allocation of the resources listed above.

3.1 ALLOCATION OF MAIN STORAGE

The main storage requirements of HASP are as follows:

- Storage space for buffering card images and print lines between intermediate direct-access storage devices and unit-record devices.
- Storage space for normally non-resident control tables during times when they are resident in main storage.
- Storage for console messages which have been queued for output to or input from one or more operator consoles.
- Storage for elements which reflect the status of all jobs which are queued for any stage of processing by HASP.
- Storage space for non-resident processing routines (overlays) during times when they are in main storage.

The HASP Buffer Pool

The first two requirements for main storage are provided for by the HASP Buffer Pool, a group of buffers with the following basic format:

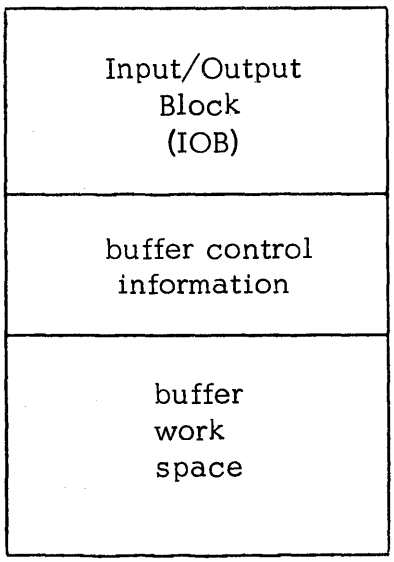


Figure 3.1.1 - The HASP Buffer

Since the use of this buffer always involves some input/output activity, a standard Operating System Input/Output Block (IOB) is provided with each buffer for the purpose of being used to initiate this input/output activity.

The "buffer control information" area is an extension of the IOB used by HASP for input/output synchronization.

The "buffer work space" is a fixed-length (set by HASPGEN) area into which data is read and/or out of which data is written.

In addition to a fixed number of buffers (set in accordance with region or partition size), the buffer pool contains a one-word control field called the Buffer Pool Control Block which contains the address of the first available buffer in the buffer pool. Each available buffer contains the address

of the next available buffer with the last available buffer containing a zero address. If no buffers are available, the Buffer Pool Control Block contains zero.

The above technique is called "chaining," the buffers are said to be "chained," and the field containing the address of the next element in the chain is referred to as the "chain field." Chaining is used throughout HASP for the maintenance of resources.

To obtain an available buffer from the buffer pool, the Buffer Pool Control Block is tested for an available buffer. If one exists it is removed from the available chain by moving its chain address into the pool control block.

To release a buffer to the available chain, the contents of the pool control block are moved to the chain field of the buffer, and the address of the buffer is placed in the pool control block.

The Console Message Buffer Pool

The third requirement for main storage is provided for by the Console Message Buffer Pool. This buffer pool is organized similarly to the HASP Buffer Pool except for the format of the buffers which is as follows:

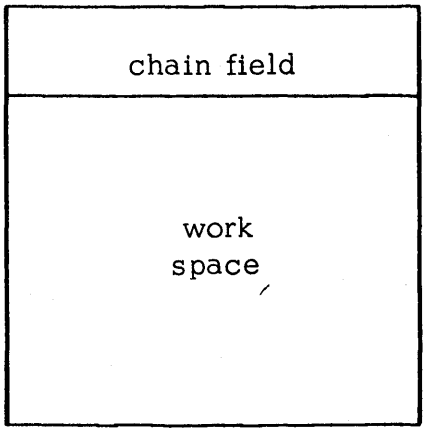


Figure 3.1.2 — The Console Message Buffer

Since IOB's are provided for each console, it is not necessary to provide such a control block with each buffer.

The length of the work space is consistent with the maximum length of a console message.

Buffers in this buffer pool are obtained and released by exactly the same procedure used in the HASP Buffer Pool.

The HASP Job Queue

The fourth requirement is provided for by the HASP Job Queue. For more information about this facility see section 3.6.

The HASP Overlay Area Pool

The HASP Overlay Area is similar to the HASP Buffer in format; however, the size of the "work space" is set to accommodate the largest non-resident HASP control-section (CSECT). Although the fixed number of overlay areas (set by HASPGEN) are chained together, control fields indicate the area status and contents for the purpose of sharing areas containing the same CSECT or for selecting an area to overlay with a new CSECT.

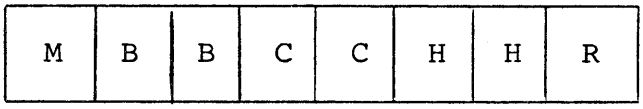
3.2 ALLOCATION OF DIRECT-ACCESS SPACE

The direct-access allocation technique employed by HASP must meet the following requirements:

- It must use a minimum of CPU time.
- It must not use an excessive amount of main storage.
- It must not be susceptible to the "hashing" or "fragmentation" problem.
- It must be capable of allocating for any direct-access device which is supported by Operating System/360.
- It must be device transparent to the user.
- It must be consistent with the checkpoint/restart technique used by HASP.

The HASP Track Address

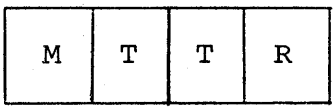
The standard Operating System track address is defined to be an eight-byte field with the following format:



where:
M = Module
BB = Bin
CC = Cylinder
HH = Head
R = Record

Figure 3.2.1 - The Operating System Track Address Format

For the purpose of HASP, this track address can be reduced to a four-byte field with the following format:



where:
M = Module (DEB extent number)
TT = True Track Number
R = Record

Figure 3.2.2 - The HASP Track Address Format

The reduction in the length of the track address permits it to be kept in a single word of storage or in a general purpose register simplifying the handling of the track address.

The HASP Master Track Group Map

The HASP Master Track Group Map is a table which represents the sum total of all track groups or logical cylinders available on all HASP direct-access SPOOL volumes. (A track group contains one or more tracks which are considered a single resource.) Each bit in the HASP Master Track Group Map represents a single track group on one direct-access volume. If the bit is one, it indicates that the corresponding logical cylinder is available for allocation; if the bit is zero, the logical cylinder is not available to HASP or has already been allocated by HASP.

The HASP Job Track Group Map

The HASP Job Track Group Map is identical to the HASP Master Track Group Map except that one word has been added to the front to save the last track address which was allocated to the particular job with which the map is associated. The bits in the Job Track Group Map represent the same track groups as the bits in the Master Track Group Map except that a one bit indicates that the respective track group has been allocated to the associated job and a zero indicates that the group has not been allocated to the job.

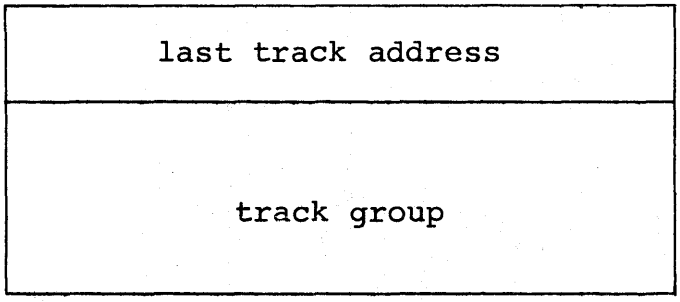


Figure 3.2.3 - The HASP Job Track Group

Two Job Track Group Maps are associated with each job. One represents the track groups used to contain the input data (SYSIN), and the other represents the groups used to contain the output data (SYSPRINT and SYSPUNCH).

Direct-Access Space Allocation Procedures

When the direct-access space allocation subroutine is entered, it first examines the first four bytes of the appropriate Job Track Group Map to determine if a new track group is required. A new group is required whenever no tracks have been allocated to this job (the last track address is zero) or if all of the tracks in the last group allocated have been used.

If a new track group is not required, the record or head field of the last track address is incremented to provide a new track address.

If a new track group must be allocated, the Master Track Group Map is scanned for an available group. When the next group to be allocated is determined, the appropriate bit in the Master Track Group Map is set to zero, and the corresponding bit in the Job Track Group Map is set to one. A track address is then constructed to represent the first track in the new group, and this track address is saved in the first four bytes of the Job Track Group Map.

When any direct-access input/output operation is initiated by HASP, the HASP I/O interface saves the cylinder which was referenced by module. When a new track group must be allocated, the allocation routine first tries to allocate a group corresponding to the last cylinder referenced on each module. If these groups are not available, the routine attempts to allocate within one cylinder of the last references. If track groups within these cylinders are not available, the routine tries to allocate a group within two cylinders, and so on, until the entire track group map has been examined.

Direct-Access Space De-Allocation Procedure

To de-allocate the direct-access space allocated to a particular function, it is necessary only to "OR" the track group map portion of the Job Track Group Map associated with the particular function into the Master Track Group Map. This will reset to one all bits in the Master Track Group Map which correspond to the track groups which have been allocated to the particular function.

3.3 ALLOCATION OF INPUT/OUTPUT UNITS

The HASP Device Control Table (DCT) is used by HASP to allocate all input/output units. It has the following basic format:

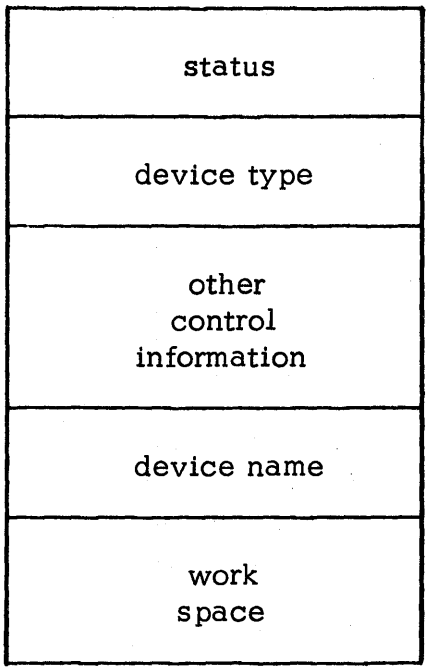


Figure 3.3.1 — The HASP Device Control Table (DCT)

The "status" field is used to indicate whether the device is available and whether it is in use.

The "device type" field specifies whether this DCT represents a card reader, printer, punch, or other type of I/O device.

The "other control information" field contains such information as the Data Control Block (DCB) address, the chain address, indications of operator commands, and other fields for synchronization purposes.

The "device name" field contains an eight-byte EBCDIC device name (such as READER1) which is primarily used for console messages.

The "work space" is a device dependent area used by some devices for extended control of the device.

All DCT's are chained together for allocation purposes. They are initialized by the HASP initialization phase if the associated devices are attached to the system.

Input/Output Device allocation consists of "running" the DCT chain and looking for a DCT of the specified type which is available and which has not been allocated. If one is found, the "in use" bit is set to one to indicate that the device has been allocated.

De-allocation consists of setting the "in use" bit to zero.

The Device Control Table is also used as a parameter list whenever Input/Output activity is initiated through the HASP I/O interface.

3.4 ALLOCATION OF CENTRAL PROCESSING UNIT TIME

The Operating System controls the allocation of Central Processing Unit (CPU) time to different tasks through the means of a Task Control Block (TCB) chain. In a similar fashion, HASP controls the allocation of CPU time to the different functions within HASP through the means of a Processor Control Element (PCE) chain. The basic format of the Processor Control Element is as follows:

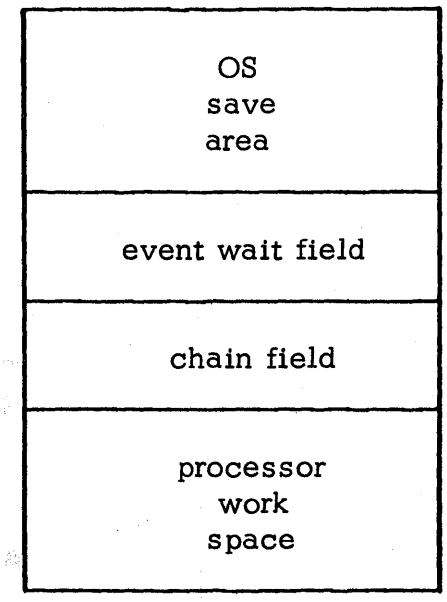


Figure 3.4.1 — HASP Processor Control Element (PCE)

Whenever a particular function is being processed, general purpose register 13 always contains the address of the Processor Control Element which is allocating the time to that function. For this reason the first eighteen words of the PCE are a standard OS register save area.

The "event wait field" is a two-byte field which describes the dispatchability of the function under the control of this PCE. If this field is zero, the function is dispatchable. If this field is non-zero, the function is not dispatchable and the bit which is one specifies upon what event the function is "waiting".

The "chain field" contains the address of the next PCE in the PCE chain.

The "processor work space" is a variable length area which is used by the program processing the function as a scratch area.

HASP searches the PCE chain looking for a PCE which is dispatchable. When a dispatchable PCE is located, the general purpose registers are loaded from the PCE/OS save area and control is passed to the location specified in register 15.

When control is returned to the dispatching program, the general purpose registers are saved in the PCE and the search for dispatchable PCEs continues. If a notable event occurred since the last PCE dispatch such as the freeing of a common resource or the "posting" of a specific event, the search starts at the beginning of the PCE chain; otherwise, it starts with the PCE following the last dispatched. The program returning control to the dispatching program must set the return address in register 15 before returning.

When no PCEs are found to be dispatchable, the HASP task enters an OS WAIT state to allow the Operating System to allocate CPU time to other tasks.

3.5 ALLOCATION OF PROGRAMS

The programs of which HASP is composed can be divided into the following classifications:

- The Dispatcher
- Processors
- Control Service Programs
- Miscellaneous Programs

The Dispatcher is the dispatching program described in Section 3.4. Its function is to distribute CPU time among the various processors described below.

Processors are programs which control the execution of various HASP functions such as reading cards, printing, punching, etc. With each processor is always associated at least one Processor Control Element which causes the dispatcher to give control to the processor and allows the processor to synchronize with various HASP events. The PCE work space also permits the processors to be written re-enterably such that by defining more than one PCE for a given processor, the processor can control an essentially unlimited number of functions simultaneously. For instance, by defining ten PCEs for the Print Processor, up to ten printers can be serviced simultaneously utilizing and requiring only one copy of the processing program.

The Control Service Programs are subroutines used by the processors in accomplishing their functions. By using the PCE/OS save area, the control service programs can maintain the re-enterability of the processors.

Miscellaneous Programs are those special purpose programs which do not fall into any of the other three categories, such as the HASP Initialization Program. They are executed only once and need not be considered in the normal HASP job flow.

3.6 ALLOCATION OF JOBS

HASP maintains its job pointers in the HASP Job Queue, a table of elements with the following basic format:

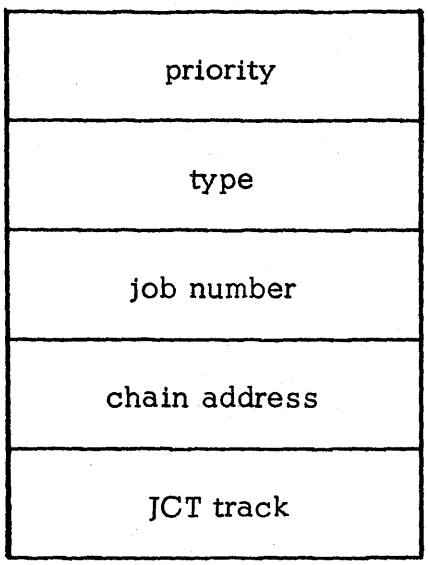


Figure 3.6.1 — The HASP Job Queue Element

The "priority" represents the dynamic priority of the job within the HASP system.

The "type" represents the function for which the element is queued or the function in which the job is currently being processed.

The "job number" is the number sequentially assigned to each job by HASP as it enters the system.

The "chain address" is the address of the next element in the chain.

The "JCT track" is the track address of the HASP Job Control Table described below.

Two chains are maintained in the Job Queue. The first chain represents those jobs which are currently awaiting processing or being processed. Elements in this chain are chained in the order of their priority. The second chain represents the inactive or unused queue elements.

To add a job to the job queue, a queue element is obtained from the inactive chain, initialized with the information shown in figure 3.6.1, and inserted into the active chain according to its priority.

To obtain a job from the job queue, the active chain is searched for an element of the specified type. When found, the "type" field is modified to reflect the fact that the job is now being processed.

To return a job to the job queue, the element is moved from the active chain to the inactive chain. Since the priority is of no concern here, the element is placed at the head of the chain.

The HASP Job Control Table (JCT)

The HASP Job Control Table contains all of the information necessary to process the associated job in the following basic format:

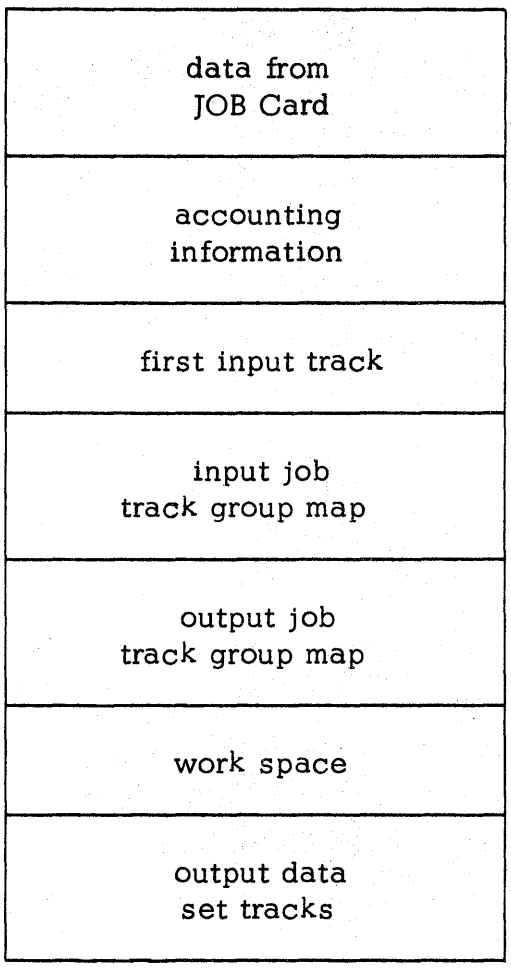


Figure 3.6.2 - The HASP Job Control Table (JCT)

The HASP Job Control Table is normally resident on a direct-access intermediate storage device. Once the HASP Job Queue Element is obtained, the "JCT track" in the element can be used to initiate a read into a HASP Buffer. Once this read has been completed, all information necessary to process the job can then be obtained.

3.7 ALLOCATION OF OVERLAY AREAS AND NON-RESIDENT CONTROL SECTIONS

Portions of the various programs of which HASP is composed are organized into non-resident control sections (CSECTs) and stored in an overlay library (OLAYLIB) on a direct-access volume. These control sections contain HASP re-entrant subroutines and/or data which may be requested for use by a Processor.

The user obtains an Overlay Area by requesting from the overlay control service program for use of a non-resident CSECT. If the CSECT requested is in main storage, the user is allowed to use the Overlay Area for processing. If, however, the CSECT is not already in an area, an area must be selected to hold the requested CSECT. The requesting Processor is made to "wait" until the requested CSECT is read from direct-access into main storage.

The algorithms for Overlay Area allocation cause multiple users of the same CSECT to use only one area, into which that CSECT is read. Competition for areas is resolved partially by the priority associated with each overlay CSECT. However, a "pre-empting" (roll) algorithm prevents any Processor from being indefinitely delayed, even if the system has only one Overlay Area.

The user releases an Overlay Area by requesting that overlay services remove his PCE from association with the area.

(The remainder of this page intentionally left blank.)

4.0 HASP PROCESSORS

This section contains detailed internal information about each of the HASP Processors and is intended primarily for use by system programmers.

4.1 INPUT SERVICE PROCESSOR

4.1.1 INPUT SERVICE PROCESSOR - GENERAL DESCRIPTION

The functions of the Input Service Processor are as follows:

- To read card images from an input device.
- To detect and scan JOB cards, extracting parameters for job accounting, job control, and print and punch identification.
- To detect and process other control cards such as the PRIORITY, MESSAGE, ROUTE, SETUP, COMMAND, DD*, and DD DATA cards.
- To assign a unique HASP job number to each job.
- To log jobs into the HASP System.
- To assign job priority based upon PRIORITY card or JOB card parameters.
- To generate, from cards read, a JCL file and input data files, and to record these files on direct-access storage device(s) for later use by the Execution Control Processor (see Section 4.2).
- To generate HASP Job Control Tables, Job Queue Entries, and other HASP control blocks required for later job processing.
- To queue jobs for processing by the Execution Control Processor.

The Input Service Processor is coded re-enterably in such a way that it can accept jobs from a number of different input devices (with different hardware characteristics) simultaneously. The re-enterability is attained by retaining all storage unique to a job in the Processor Control Element (see figure 4.1.1) which must be unique for each input device.

4.1.2 INPUT SERVICE PROCESSOR - PROGRAM LOGIC

The Input Service Processor is divided into three phases, 13 subroutines, and three non-process exits. This section will give a functional description of each of these phases, subroutines, and exits to aid the System Programmer in gaining a working knowledge of the processor.

PHASES

Phase 1 - Processor Initialization

The Initialization Phase, which is written as an overlay segment, begins by attempting to acquire an input device. If no input device is available, the processor is placed in a HASP $WAIT state until a device is made available; whereupon the entire procedure is repeated until an input device is available. Upon acquiring an available input device the processor continues by acquiring a Device Control Table (DCT) for the direct-access device(s) and a HASP buffer for use as an input buffer.

If the input device is not a remote terminal, a chain of Channel Control Words (CCW's) is then constructed in the input buffer which will be used to read 80-byte records from the input device into the rest of the input buffer. These CCWs are constructed in such a way that the input records will be read into adjacent areas in the input buffer with as many cards being read as the buffer will hold. The initialization of the PCE Work Area is then completed and control is transferred to Phase 2.

If the input device is a remote terminal, transmission is initiated by calling upon the Remote Terminal Access Method to open the Remote Terminal Device Control Table. Control is then passed to Phase 2.

Phase 2 - Main Processor

The Main Processor Phase reads cards from the input device, scans each card to detect HASP control cards and processes these cards as follows:

/*control card--The control card scan routine (HASPRCCS) is called to process the control card and take any appropriate action.

Job Card--The JOB card scan routine (HASPRJCS) is called to terminate the previous job (if any), to scan the JOB Card, and to initialize the PCE work area for the processing of the following job.

DD* or DD DATA--A track address is obtained for the first data block of the input data set. A dummy card is added to the JCL file which contains the track address in columns 1-4.

This card is differentiated from other cards by setting the control byte (see figure 8.15.1). The DD* or DD DATA statement is then added to the JCL file in normal fashion. Control is subsequently turned back to the main processor to process the input data.

When a hardware end-of-file is detected on the input device, or when "$DRAIN input device" command is entered by the operator, control is given to Phase 3.

Phase 3 - Processor Termination

Upon receiving control from the Main Processor, the Processor Termination Phase, which is written as an overlay segment, terminates the last job (if any), issues a rewind and unload command to the input device if it is tape, frees the input buffer, closes the input DCT if it is a Remote Device, releases the input and direct-access devices, and returns control to Phase 1.

SUBROUTINES

HASPRCCS -- Subroutine to Process HASP /* Control Cards

The HASPRCCS subroutine, which is written as an overlay segment, is called whenever the Main Processor Phase encounters a /* control card. The control card type is first determined and then processing continues as follows:

> /*COMMAND Card -- The command is listed on the operator's console and then added to the Command Processor's input command queue.
>
> /*PRIORITY Card -- The previous job (if any) is terminated, the priority specified is converted to binary and saved, and the scan is continued with the next card. If the following card is not a JOB card, the message, "device SKIPPING FOR JOB CARD", is written on the operator's console, the effect of the /*PRIORITY Card is nullified, and the input stream is scanned for another /*PRIORITY or JOB card.
>
> /*ROUTE Card -- The appropriate routing byte is set to the value associated with the destination indicated. If an invalid field is encountered, an appropriate message is issued, both to the operator and to the programmer, and further job processing is bypassed.

/*SETUP Card -- The volumes to be mounted are listed on the operator's console and the job is placed in "hold" status.

/*MESSAGE Card -- Leading and trailing blanks are removed and the message is routed to the operator's console.

If the control card type is not recognized, the card is ignored and treated like any other /* card.

HASPRJCS--Subroutine to Scan and Initialize Job Control Information

The HASPRJCS subroutine, which is written as an overlay segment, is called whenever the Main Processor Phase encounters a JOB card. The previous job (if any) is terminated by calling the RJOBEND subroutine. The master job number is incremented and its new value is assigned to the current job. The job control information in the PCE Work Area (see figure 4.1.1) is initialized by scanning the JOB card and extracting parameters relative to job control. The first JCL block is initiated, and control is passed to the Job Initialization Subroutine: HASPRJBI.

RSCAN - RSCANA -- Subroutine to Scan Parameters from JOB Card

This subroutine has two entry points; the entry point: "RSCAN" is used to scan numeric parameters from the JOB card, while the entry point: "RSCANA" is used to scan alphameric parameters from the JOB card. There are also two returns from the subroutine. If return is made to the first byte following the Branch and Link (the call) instruction, it indicates that the final parameter on the JOB card was returned on the previous call and that there are no more parameters. If return is made to the fourth byte following the Branch and Link instruction, it indicates that parameter register "R1" contains the next parameter, right-adjusted with leading binary zeroes. If the parameter was a "null" parameter, "R1" will be zero. If this subroutine detects an illegal character (such as a non-numeric character in a numeric field) or more than four characters in a parameter, control is transferred to the RBADJOBC subroutine.

RCONTNUE -- Subroutine to Validate Continuation Cards

This subroutine validates JCL continuation cards by ensuring that columns 1 and 2 are punched with slashes and that column 3 is blank. The start of the continuation card is located and

control is returned to the caller. If an invalid continuation card is discovered, control is passed to the illegal job card subroutine for further processing.

REBCDBIN -- Subroutine to Convert from EBCDIC to Binary

This subroutine expects to find numeric EBCDIC characters with leading binary zeroes in parameter register "R1". There are two returns from the subroutine. If return is made to the first byte following the Branch and Link (the call) instruction, it indicates that the parameter register now contains the binary equivalent of the EBCDIC input. If return is made to the fourth byte following the Branch and Link instruction, it indicates that the parameter register was zero (null parameter) and contained no EBCDIC to translate.

HASPRJBI -- Subroutine to Initialize Job Processing

This subroutine, which is written as an overlay segment, receives control from the JOB Card Scan Routine (HASPRJCS) and completes the initialization of the various control blocks for input job processing. A "job on" message is issued to the operator, the job's priority is assigned based upon JOB card or /*PRIORITY card parameters, and the job is queued in the active input queue. Control is then returned to the Main Processor Phase.

RBADJOBC -- HASPRIJC -- Subroutine to Process Illegal Job Cards

This subroutine notifies the operator of an illegal JOB card, calls the subroutine: "RJOBKILL" to delete the job, and returns control to the Main Processor Phase.

RJOBEND -- Subroutine to Complete Job Input Processing

This subroutine tests whether the Input Processor is currently processing a job, and if it is not, returns control immediately. The RJOBTERM subroutine is called to terminate the input processing of the job, and the job is queued for the Execution Control Processor in the logical queue associated with the job's JOB CLASS. Control is then returned to the calling program.

RGET -- Subroutine to Get Next Card from Input Buffer

This subroutine returns the address of the next card to be processed by the Input Service Processor in register "RPI". If the input buffer is empty or if all the cards in the input buffer have been processed, an IOS read is staged from the input device and the subroutine places the processor in a HASP $WAIT state until the input buffer has been filled. If the input device is a remote terminal, a "call" is made on the Remote Terminal Access Method to procure the next card. If a permanent error is detected on the input device, no action is taken until after the last card has been processed and then the JOB currently being processed is deleted with appropriate comments to the operator. Processing then continues by scanning the input stream for the next JOB card.

This subroutine also processes the operator commands "$STOP input device" and "$DELETE input device" by entering the HASP $WAIT state and calling the subroutine RJOBKILL to delete the job, respectively.

There are two returns from the subroutine. If return is made to the first byte following the Branch and Link (the call) instruction, it indicates that the last card has been processed and that an end-of-file has been sensed on the input device. If return is made to the fourth byte following the Branch and Link, it indicates that register "RPI" contains the address of the next card.

RPUT -- RPUTOLAY -- Subroutine to Add Card to Output Buffer

This subroutine accepts 80-byte card images and blocks them into standard HASP Data Blocks (see section 8.15). If the current output buffer is full, it is truncated and scheduled for output, and a new HASP buffer is acquired and used as the next output buffer. If no output buffer exists upon entry, it indicates that the processor is skipping for a JOB card and the subroutine returns without taking any action.

RJOBKILL -- Subroutine to Delete Current Job

This subroutine tests whether the input processor is currently processing a job, and if it is not, returns control immediately. If a job is being processed, the operator is notified that the job is being deleted, the RJOBTERM subroutine is called to terminate the input processing of the job, and the job is placed in the Print Processor Queue for subsequent processing. Control is then returned to the calling program.

RJOBTERM -- Subroutine to Terminate Job

This subroutine terminates the last output buffer and schedules it for output. It then acquires a HASP buffer, and from information kept in the PCE Work Area (see figure 4.1.1) constructs the Job Control Table (JCT) and schedules it for output. Control is then returned to the calling program.

RGETBUF -- Subroutine to Initialize Output Buffers

This subroutine acquires a HASP buffer for an output buffer and returns with the address of the buffer in register "R1".

NON-PROCESS EXITS

The following routines are used to put the Input Service Processor in a HASP $WAIT state if a HASP resource is not available. In all cases Reader Link Register 2 ("RL2") must have been set to the restart address before the routine is entered.

- RNOUNIT -- A HASP Unit was not available.
- RNOCMB -- A HASP Console Message Buffer was not available.
- RNOJOB -- The HASP Job Queue was full and a new entry could not be added.

When the respective resource is available, the processor is $POSTed and another attempt is made to acquire the resource.

Figure 4.1.1 -- INPUT SERVICE PCE WORK AREA FORMAT

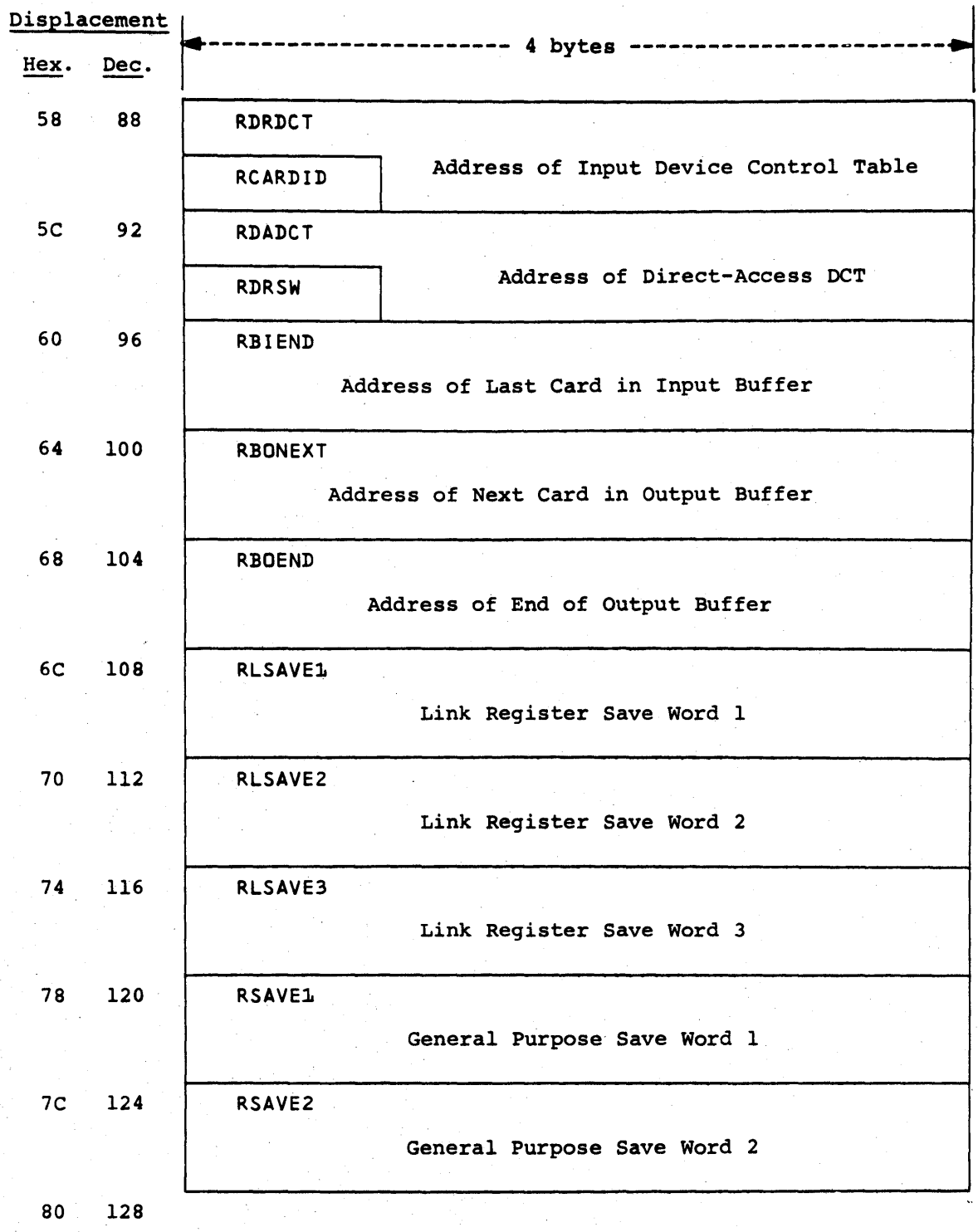


Figure 4.1.1 -- INPUT SERVICE PCE WORK AREA FORMAT (CONTINUED)

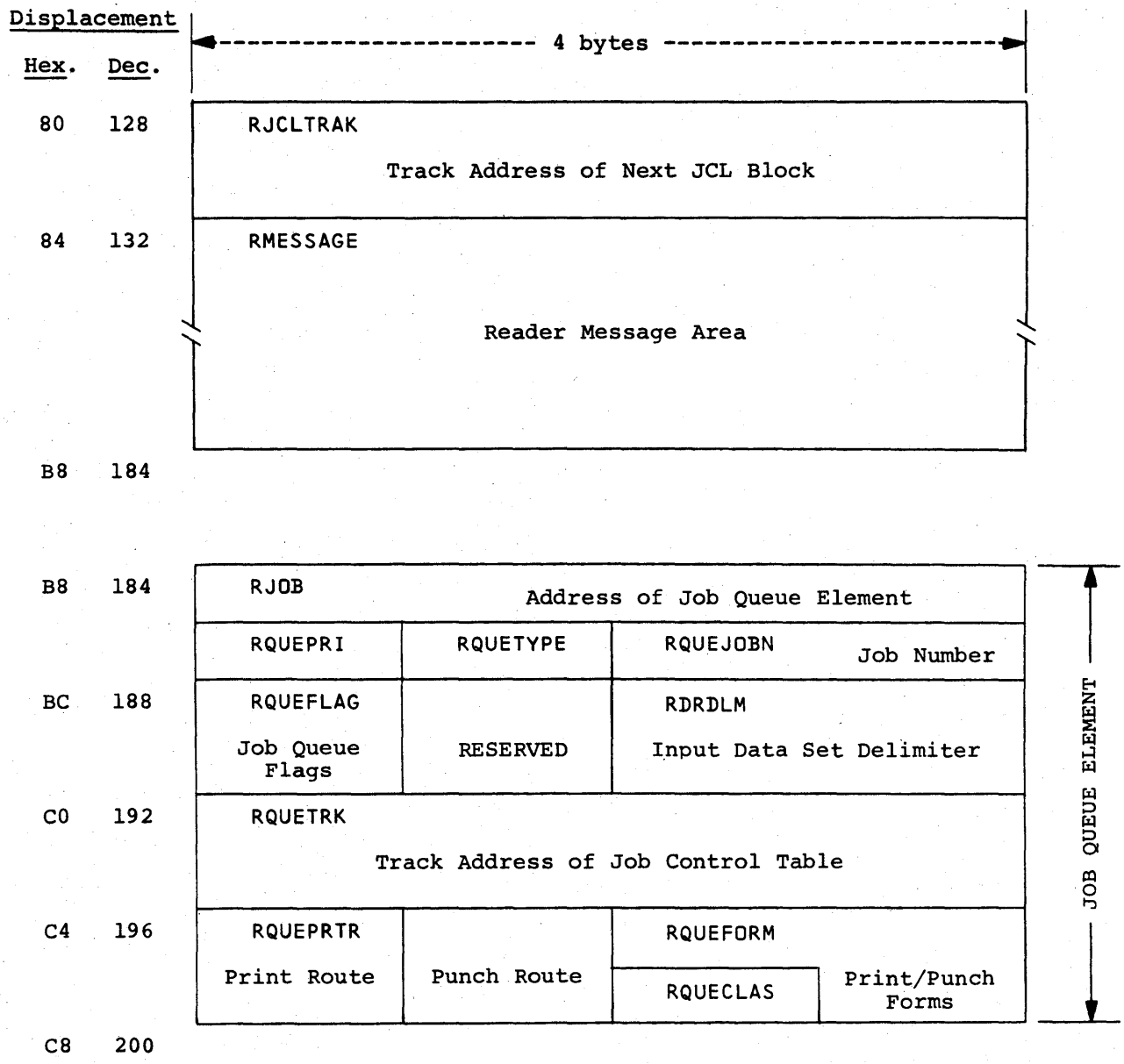


Figure 4.1.1 -- INPUT SERVICE PCE WORK AREA FORMAT (CONTINUED)

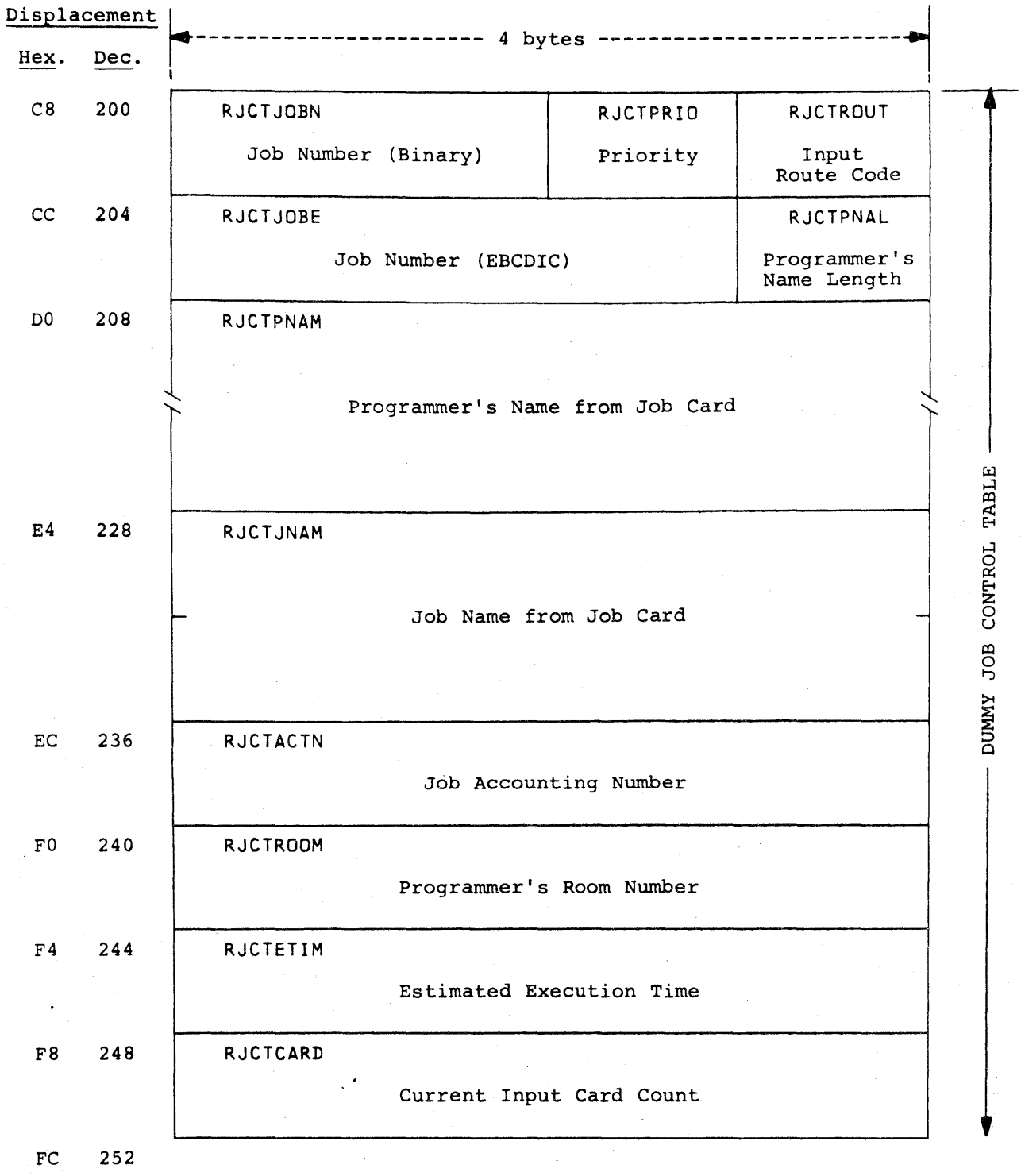


Figure 4.1.1 -- INPUT SERVICE PCE WORK AREA FORMAT (CONTINUED)

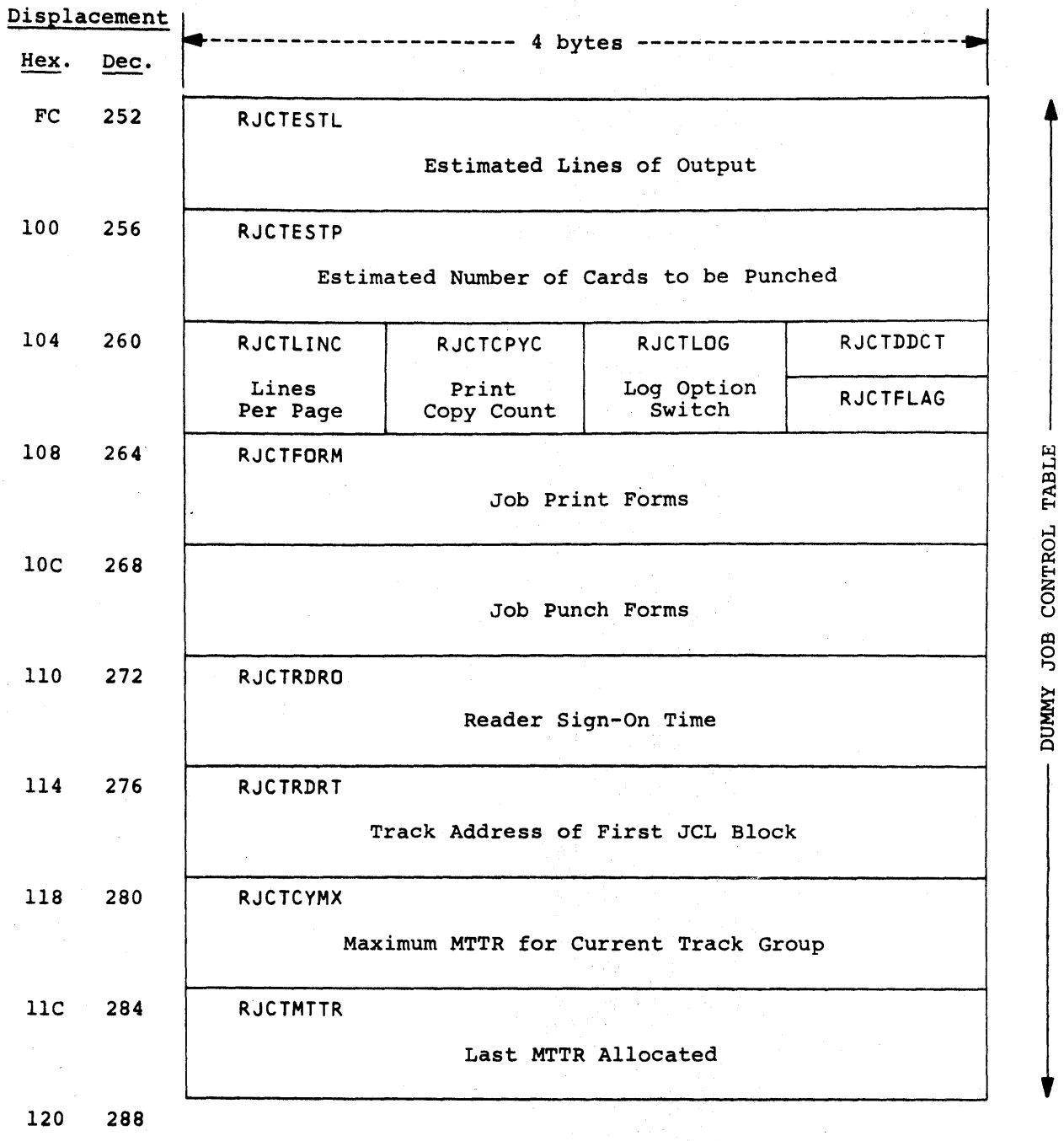


Figure 4.1.1 -- INPUT SERVICE PCE WORK AREA FORMAT (CONTINUED)

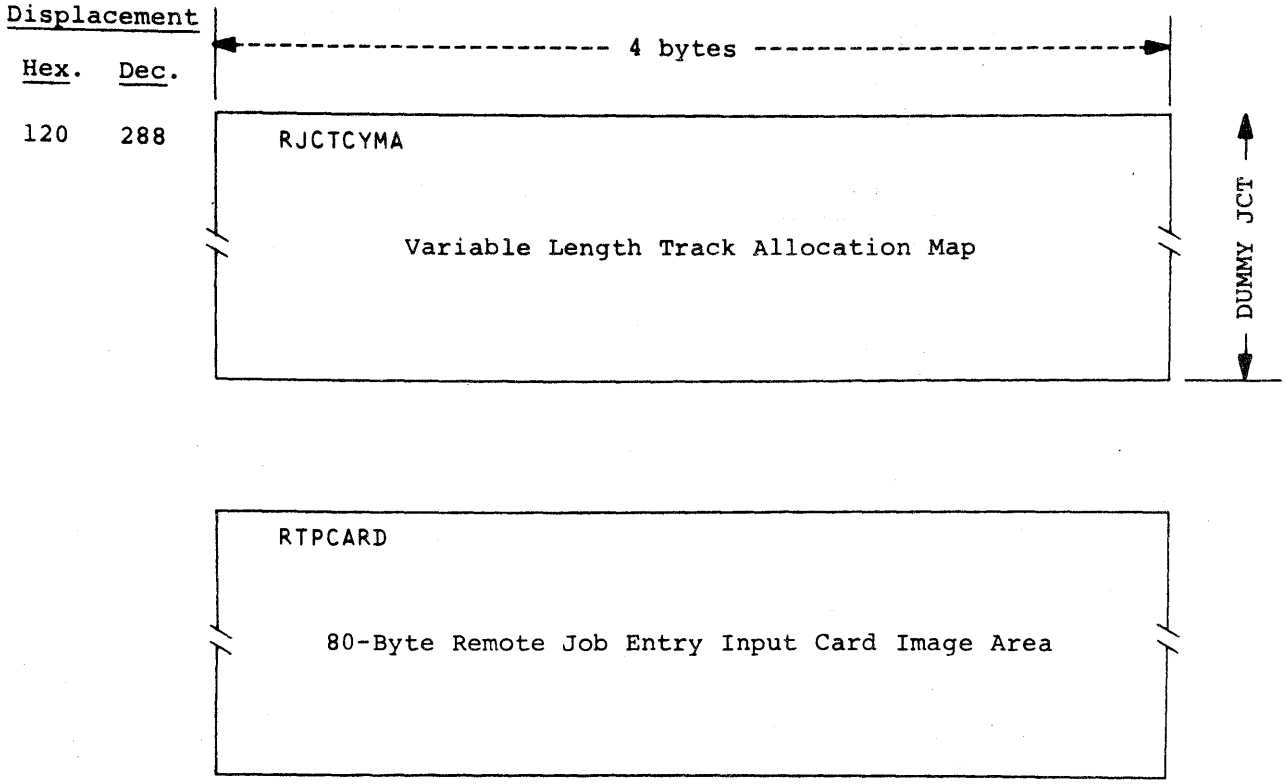


Figure 4.1.1 -- INPUT SERVICE PCE WORK AREA FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
58	88	RCARDID	1	Type of card being processed --

Hex. Value	Meaning
00	Normal Card.
03	Internally Generated Card.
04	HASP Control Card.
13	Illegal Control Card.
19	Last JCL Card.
73	Dummy Track Address Record.

58	88	RDRDCT	4	Address of Reader, Tape, Internal Reader, or Remote Device Control Table.
5C	92	RDRSW	1	Reader Switches --

Bit	Name	Meaning
0	RJOBQUED	Job has been Queued.
1	RSYSINSW	Processing Internally Generated DD * Card.
2	RXBJOBSW	Processing XEQ Batch Class Job.
3	ROSINSW	Processing O/S Input Data Set.
4	RJCLSW	Processing JCL.
5	RDREOFSW	End of File Indication.
6	RNOSCAN	Not Scanning JCL (DD DATA).
7	RJFLUSH	Job Flush Message has not been issued.

5C	92	RDADCT	4	Address of Direct-Access Device Control Table.
60	96	RBIEND	4	Address of Last Card in Input Buffer.
64	100	RBONEXT	4	Address of Next Card in Output Buffer.
68	104	RBOEND	4	Address of End of Output Buffer.
6C	108	RLSAVE1	4	Link Register Save Word 1.
70	112	RLSAVE2	4	Link Register Save Word 2.
74	116	RLSAVE3	4	Link Register Save Word 3.

Figure 4.1.1 -- INPUT SERVICE PCE WORK AREA FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
78	120	RSAVE1	4	General Purpose Save Word 1.
7C	124	RSAVE2	4	General Purpose Save Word 2.
80	128	RJCLTRAK	4	Track Address of Next JCL Block.
84	132	RMESSAGE	52	Reader Message Area.
B8	184	RJOB	4	Address of Job Queue Element (when Job has been Queued).
B8	184	RQUEPRI	1	Job Queue Priority (Before Queueing) --
			Bits 0-3	Priority (0-15).
			Bits 4-7	Zero.
B9	185	RQUETYPE	1	Job Class - X'80' (Before Queueing).
BA	186	RQUEJOBN	2	Job Number (Before Queueing).
BC	188	RQUEFLAG	1	Job Queue Flags --
				Bits Name Meaning
				0 QUEHOLD1 Job Held: TYPRUN=HOLD or Input Device Held.
				1-7 Reserved.
BD	189		1	Reserved.
BE	190	RDRDLM	2	Input Data Set Delimiter
C0	192	RQUETRK	4	Track Address of Job Control Table.
C4	196	RQUEPRTR	1	Print Routing: 0 = Local. n = Remote n.
C5	197		1	Punch Routing: 0 = Local. n = Remote n.
C6	198	RQUECLAS	1	Job Class - X'80' (After Queueing).
C6	198	RQUEFORM	2	Job Print Forms (Before Queueing).
C8	200	RJCTJOBN	2	Job Number (Binary).
CA	202	RJCTPRIO	1	Priority from /*PRIORITY Card.

Figure 4.1.1 -- INPUT SERVICE PCE WORK AREA FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
CB	203	RJCTROUT	1	Input Route Code: 0 = Local. n = Remote n.
CC	204	RJCTJOBE	3	Job Number (EBCDIC).
CF	207	RJCTPNAL	1	Programmer's Name Length.
D0	208	RJCTPNAM	20	Programmer's Name from Job Card.
E4	228	RJCTJNAM	8	Job Name from Job Card.
EC	236	RJCTACTN	4	Job Accounting Number.
F0	240	RJCTROOM	4	Programmer's Room Number.
F4	244	RJCTETIM	4	Estimated Execution Time.
F8	248	RJCTCARD	4	Current Input Card Count.
FC	252	RJCTESTL	4	Estimated Lines of Output.
100	256	RJCTESTP	4	Estimated Number of Cards to be Punched.
104	260	RJCTLINC	1	Lines per Page.
105	261	RJCTCPYC	1	Number of Copies of Print.
106	262	RJCTLOG	1	Log Option Switch.
107	263	RJCTDDCT	1	Count of Input Data Sets SPOOLed by HASP.
107	263	RJCTFLAG	1	JCT Flags.
108	264	RJCTFORM	4	Job Print Forms.
10C	268		4	Job Punch Forms.
110	272	RJCTRDRO	4	Reader Sign-On Time.
114	276	RJCTRDRT	4	Track Address of First JCL Block.
118	280	RJCTCYMX	4	Maximum MTTR for Current Track Group.
11C	284	RJCTMTTR	4	Last MTTR Allocated.

Figure 4.1.1 -- INPUT SERVICE PCE WORK AREA FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
120	288	RJCTCYMA		Variable Length Track Allocation Map.
		RTPCARD	80	Remote Job Entry Input Card Image Area.

4.2 EXECUTION CONTROL PROCESSOR

4.2.1 Execution Control Processor — General Description

The Execution Control Processor is responsible for the interface between HASP and OS/360. It presents jobs to the Operating System for execution and communicates with the I/O supervisor to supply SYSIN data for a job and to accept SYSPRINT and SYSPUNCH from a job for later printing and punching.

This Processor is re-enterably coded and has the capability to present any number of jobs to OS/360 for simultaneous execution by maintaining unique INPUT/OUTPUT streams for each job. All storage unique to a job is retained in the Processor Control Element (see Figure 4.2.2) to provide re-enterability.

The Execution Control Processor is also responsible for monitoring job limit excessions (such as time, line, or punched card estimates). Jobs are selected for OS processing based on a logical partition structure defined by HASPGEN. Each logical partition is controlled by a partition information table (PIT) which indicates the eligibility of jobs for execution by that logical partition. There is a direct correlation between the HASP logical partition and the number of initiators active in the system. Jobs thus selected for OS processing are passed to a single OS/360 Reader/Interpreter which remains constantly STARTed to a

HASP pseudo device which appears as a 2540 card reader. Only the Job Control Language statements of a job are passed to the R/I. Input stream data sets, defined by DD * or DD DATA cards have been previously transcribed to a SPOOL disk by the HASP input service processor. The JCL for a job is dynamically modified by HASP to assign pseudo unit record devices to all SYSIN and SYSOUT data sets to permit interception by HASP. The job is interpreted by the R/I and is placed in the OS job queue for immediate selection by an initiator. At the completion of a job's execution, it is placed in the OS SYSOUT queue to be processed by an output writer. Because of the assignment of pseudo unit record devices to all SYSOUT files, the output writer is required only to "print" the System Message Blocks from SYS1.SYSJOBQE. These SMB's are intercepted by HASP and are stored on the SPOOL disks as another print data set. After receiving the last SMB, HASP terminates the XEQ phase, queues the job for the HASP output processors and indicates that the logical partition requires another job. All information concerning SYSIN and SYSOUT files assigned to HASP pseudo devices is kept in Data Definition Tables (DDT). There is a DDT for each active file of a job which indicates buffer addresses, file status, record count, etc. and is correlated with the proper file through the HASP pseudo device address.

4.2.2 Execution Control Processor — Program Logic

The Execution Control Processor (XEQ) consists of the three following logical phases:

PHASE 1 — Job Control — Initiates and terminates job processing.

PHASE 2 — Asynchronous I/O Handler — Interfaces with OS/360 via the Input/Output Supervisor (IOS) to perform SYSIN/SYSPRINT/SYSPUNCH I/O requests.

PHASE 3 — Synchronous I/O Handler — Performs the SPOOL I/O required by Phase 2.

Figure 4.2.1 indicates the relationship between these three phases and OS/360.

An OS execution is initiated by Phase 1 by obtaining a suitable job from the HASP job queue and reading its Job Control Table from disk. Job limit parameters are initialized and the status of the OS/360 R/I is interrogated. If the R/I is currently processing the input for another job, Phase 1 $WAITs until it has completed. A DDT describing the JCL file for the selected job is constructed and associated with the HASP pseudo 2540 used by the R/I. The dormant R/I is then POSTed to signal the availability of

a job and control is transferred to Phase 3 to await I/O requirements from Phase 2. The OS/360 Supervisor call table has been modified by HASP initialization so that all I/O requests are diverted to Phase 2 of the XEQ processor. If the I/O request thusly intercepted refers to a HASP pseudo device, it is processed by HASP; otherwise it is passed to the Operating System Input-Output Supervisor for normal processing. Since XEQ has the capability to control the simultaneous execution of many jobs, the PCE for the job issuing the I/O request to a pseudo device must be identifiable. This is done by using a combination of the JOB name and the TCB address (Job Step TCB for MVT). Once the PCE is located, the DDT for this particular pseudo device is found by the pseudo device address from the UCB. Phase 2 verifies that there is a buffer still associated with the file and simulates the I/O request. Each channel command word in the request is examined and, when a data select type is recognized, the I/O operation is simulated by a MOVE CHARACTERS to or from the current HASP buffer for that file. Input requests are serviced by stripping (deblocking) the next card image from the HASP buffer and moving it as indicated by the CCW. These moves (only) are done while operating under the requesting program's protect key to prevent an undetected protect violation by HASP, which normally operates under protect key zero. Requirements for I/O

activities associated with Phase 2 processing are indicated by a series of status bits in each DDT. Requests to get buffers, read buffers and write buffers, are indicated in the appropriate DDT, Phase 3 of the XEQ processor is $POSTed and the HASP task is POSTed. If the requested activity must be completed before an I/O request can be satisfied by Phase 2, the I/C requestor is made to WAIT. This is done by saving the current CCW location and using the OS WAIT routine to hold the requestor. When the required I/O activity is complete, the WAITing task is POSTed and the pseudo device I/O request is re-issued. At the end of all successful I/O operations, the appropriate user appendage (channel-end appendage, etc.) is entered, the I/O is POSTed complete if required and a CSW is constructed to indicate the normal I/O completion.

When Phase 3 of the Execution Processor is entered after initiation of the job it immediately enters a HASP $WAIT state to await direction from Phase 2. Upon being activated via a $POST from Phase 2 or by a timer interrupt, this PHASE examines various status bits in the PCE and DDT's to determine the required action. This action may be either the priming of an input buffer, writing and re-initialization of an output buffer, or the notification to the operator of expiration of the estimated time of the job. An input buffer is primed by obtaining the track address

of the next buffer from the current buffer and issuing a read for the record. Status bits are set in the DDT to indicate that a read is in progress on this buffer and are reset at channel end time to indicate that the record is available. A full output buffer from Phase 2 is scheduled for transcription to disk and a new buffer is immediately obtained and initialized for use. When the buffer is initialized a track address is acquired and inserted as a forward chain in the buffer to be written. If Phase 3 is for any reason unable to get a HASP buffer, a special service called Buffer-roll is invoked. The function of Buffer-roll is to make a HASP buffer currently being utilized by another file (in this or another job) available to the requestor. This is done by selecting a low frequency DDT which owns a buffer and forcing a "free" of that buffer. To free a primary or secondary input buffer, a switch is set in the DDT to force a re-read of the buffer when the input file is next required. Output buffers are freed by terminating and writing the buffer to the SPOOL disk. Future references to this output file will cause a new buffer to be obtained and chained to the partial buffer.

A count of the number of logical records contained in each output buffer is maintained by the Phase 2 routine and is used by Phase 3 upon writing a buffer to maintain the total line and card count for this job. This accumulated figure is also compared, after each write, to

the estimated number of output records with the operator being notified on its excession. If a job exceeds either cards, lines, or time, the operator is so advised and a HASPGEN value is added to the original estimate which will cause repeated excession messages as this new estimate is reached. The job continues through normal OS/360 processing until the end of execution is reached. The job, as part of normal OS job termination, is then placed in the OS SYSOUT queue for processing by an output writer. Because of the dynamic modification of all SYSOUT= cards to pseudo devices, the only data set to be processed by the output writer is that containing the System Message Blocks. The Output Writer therefore "prints" the SMB's to a HASP pseudo device. When the last SMB is received, Phase 3 is notified (via an OS POST) to return control to Phase 1 for HASP job termination.

The job termination section of Phase 1 must now prepare the job for passage to the print queue. First, all partially filled output buffers are truncated and written out, and all input buffers are freed. The timer interval for the job is cancelled and all job execution statistics are added to the JCT. At this point the areas of the SPOOL disks used to store the job input information are made available to be re-allocated by HASP (Purged), the JCT is written to disk and the job is passed to

the print queue for printing. If no priority card was present, the job priority is recalculated as a function of the number of lines of print generated before the job is placed in the print queue.

A branch is then made to the beginning of XEQ to begin another job if available, or to display a message indicating that the logical partition is idle.

The process of dynamic examination and modification of selected JCL statements is accomplished by invoking the standard OS Reader/ Interpreter exit list option which allows users to inspect all JCL encoded by the reader. A detailed discussion of the HASP processing algorithm is contained in Appendix 12.8: HASP JCL Processing.

Figure 4.2.1 -- Execution Control - OS/360 Relationship

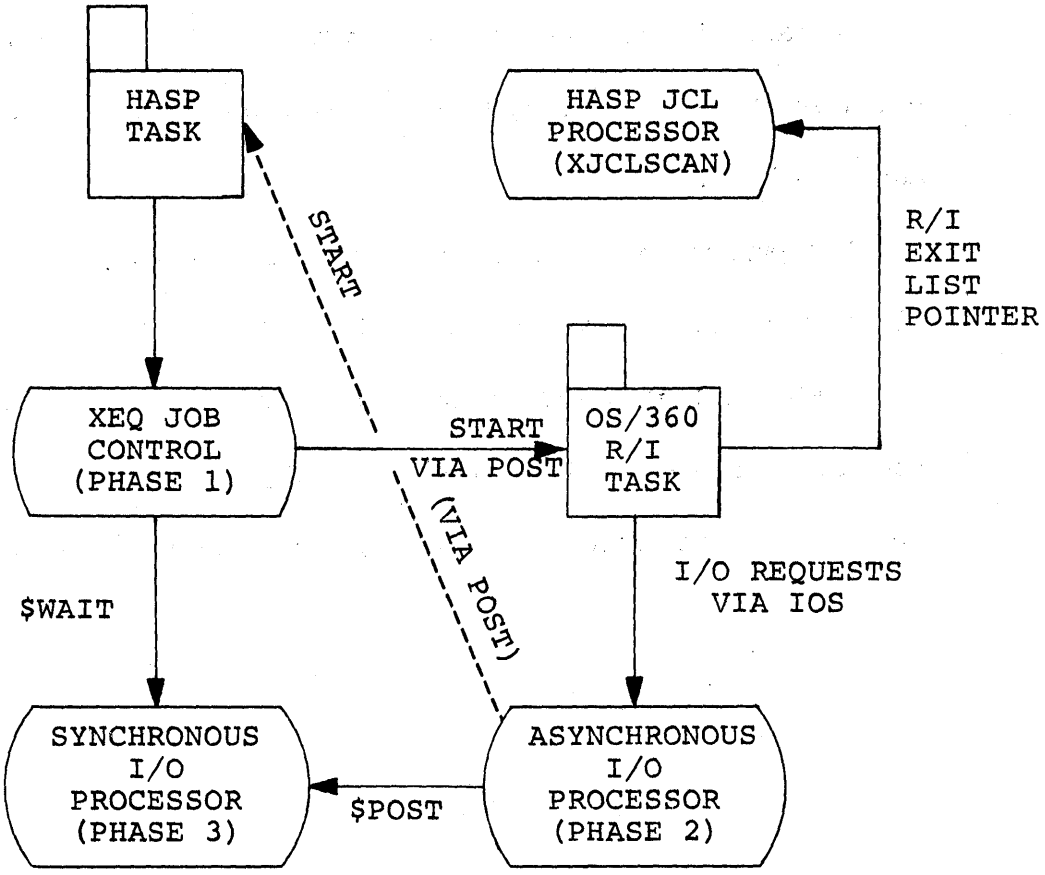


Figure 4.2.2 -- EXECUTION CONTROL PCE WORK AREA FORMAT

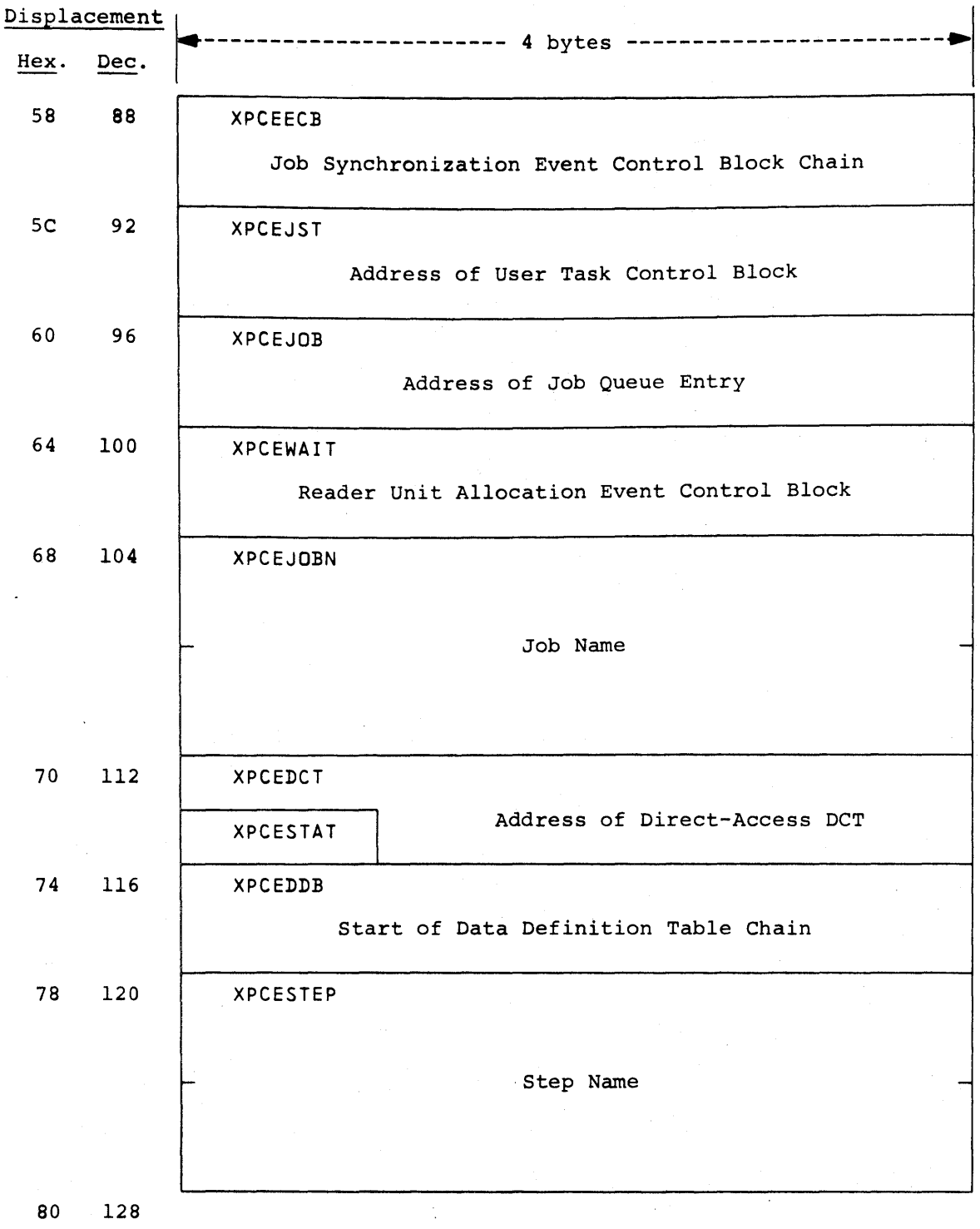


Figure 4.2.2 -- EXECUTION CONTROL PCE WORK AREA FORMAT (CONTINUED)

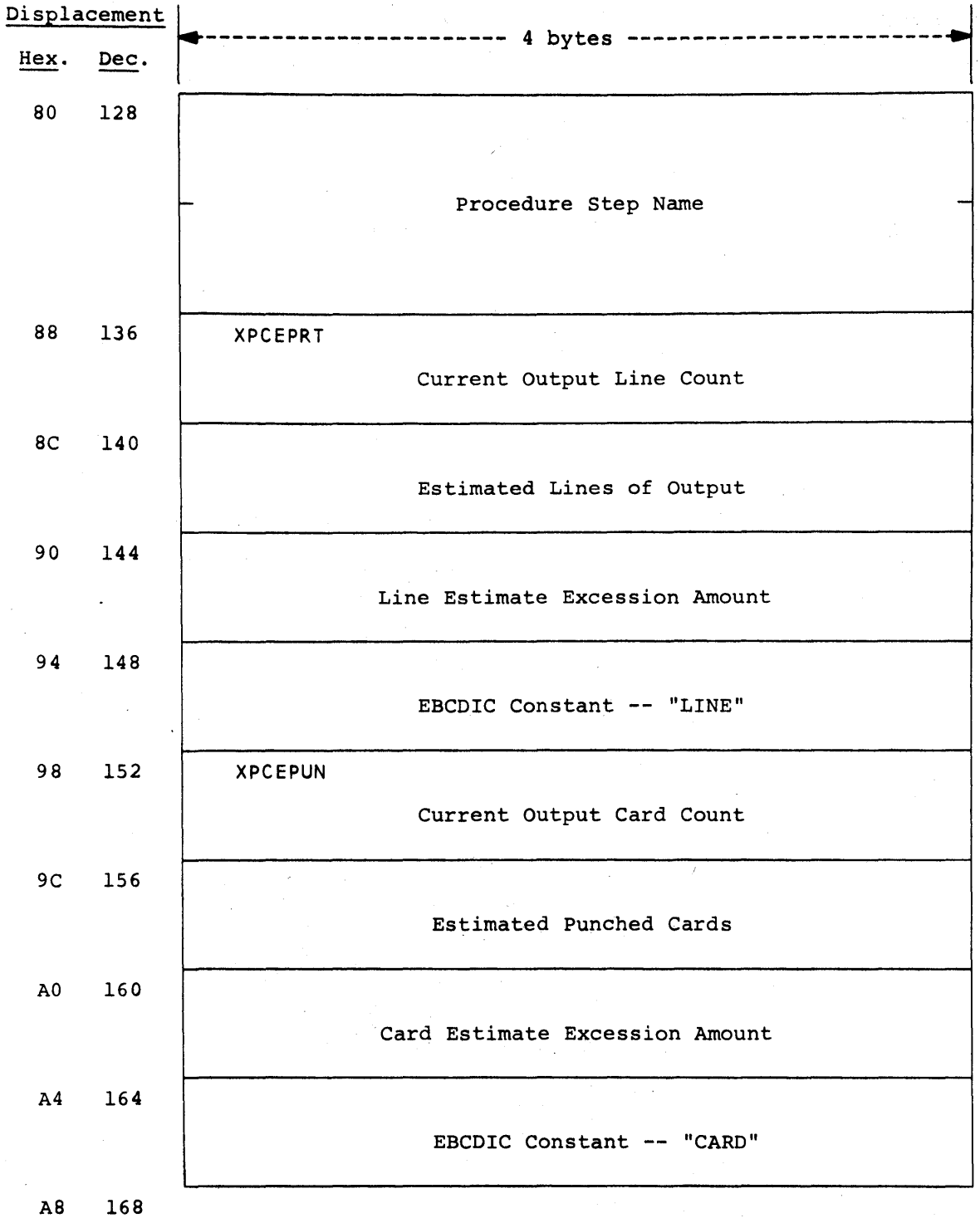


Figure 4.2.2 -- EXECUTION CONTROL PCE WORK AREA FORMAT (CONTINUED)

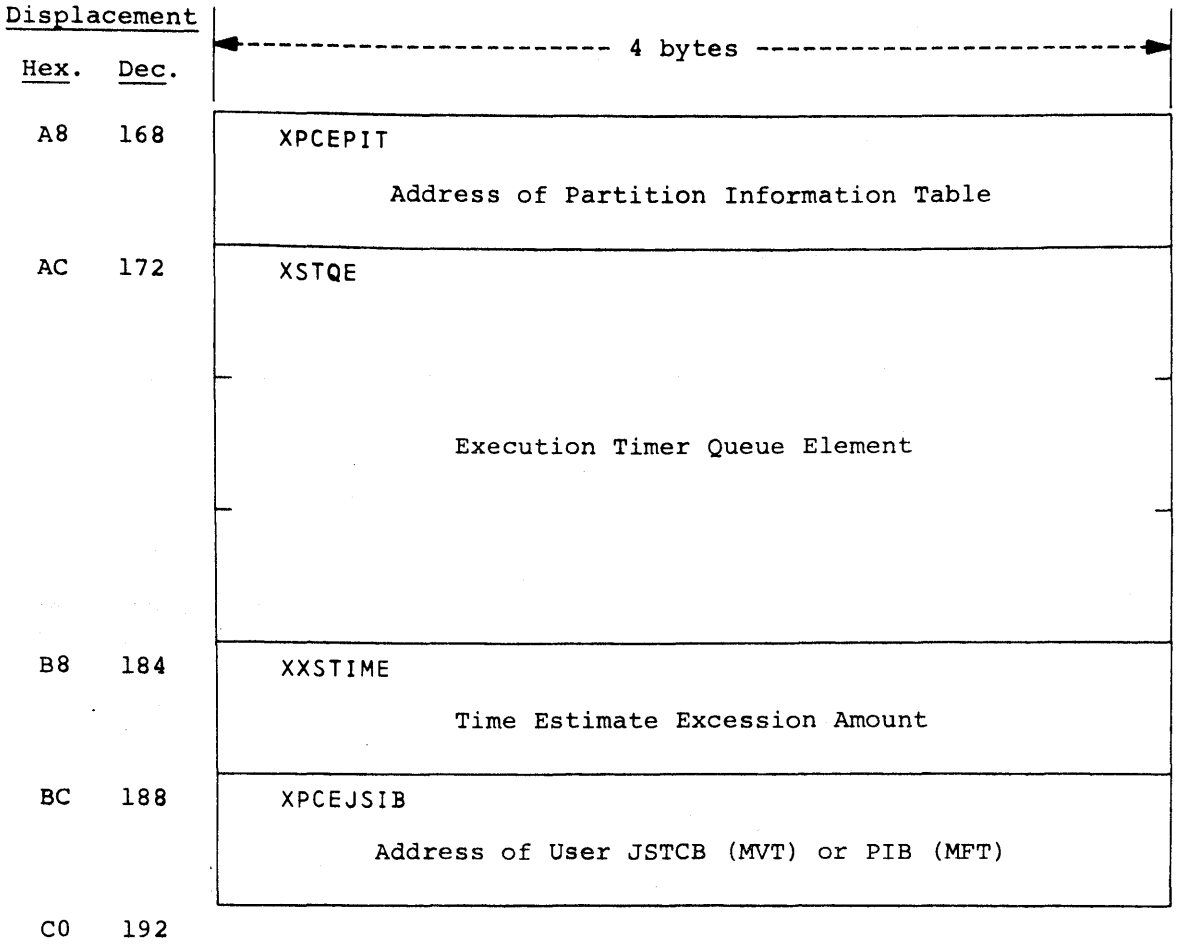


Figure 4.2.2 -- EXECUTION CONTROL PCE WORK AREA FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
58	88	XPCEECB	4	Job Synchronization Event Control Block Chain.
5C	92	XPCEJST	4	Address of User Task Control Block.
60	96	XPCEJOB	4	Address of Job Queue Entry.
64	100	XPCEWAIT	4	Reader Unit Allocation Event Control Block.
68	104	XPCEJOBN	8	Job Name.
70	112	XPCESTAT	1	Status --

Bit	Name	Meaning
0-1		Reserved.
2	XPOSTBIT	POST Request for XTHAW.
3	XRDRACT	Reserved.
4	XEOJMES	End Execution Message Sent.
5	XDUPBIT	Job with Duplicate Job Name Waiting.
6	XUCBDDB	UCB/DDT Required by Execution Interface.
7	XEOJBIT	End of Job Flag.

70	112	XPCEDCT	4	Address of Direct-Access DCT.
74	116	XPCEDDB	4	Start of Data Definition Table Chain.
78	120	XPCESTEP	8	Step Name.
80	128		8	Procedure Step Name.
88	136	XPCEPRT	4	Current Output Line Count.
8C	140		4	Estimated Lines of Output.
90	144		4	Line Estimate Excession Amount.
94	148		4	EBCDIC Constant -- "LINE".
98	152	XPCEPUN	4	Current Output Card Count.
9C	156		4	Estimated Punched Cards.

Figure 4.2.2 -- EXECUTION CONTROL PCE WORK AREA FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
A0	160		4	Card Estimate Excession Amount.
A4	164		4	EBCDIC Constant -- "CARD".
A8	168	XPCEPIT	4	Address of Partition Information Table.
AC	172	XSTQE	12	Execution Timer Queue Element.
B8	184	XXSTIME	4	Time Estimate Excession Amount.
BC	188	XPCEJSIB	4	MVT -- Address of Job Step Task Control Block. MFT -- Address of Partition Information Block.

4.3 OUTPUT SERVICE PROCESSOR (PRINT AND PUNCH)

4.3.1 OUTPUT SERVICE PROCESSOR - GENERAL DESCRIPTION

The functions of the Output Service Processor are as follows:

- To convert the print and punch output generated by the Execution Control Processor to hard copy.
- To provide for the unique identification of both print and punch output to facilitate collection and delivery.
- To provide for the routing of special data sets to printers and punches reserved for special forms processing.
- To produce multiple copies of print output upon request.
- To count print lines and produce automatic page overflow.
- To translate all illegal print characters to blanks (optional).
- To load the Universal Character Set Buffer (optional).
- To load the Forms Control Buffer (optional).
- To provide additional information for checkpoint which allows print to continue in the event of a "warm start".
- To punch a Job Accounting Card (optional).
- To process all printer and punch I/O errors with automatic error recovery (no operator intervention).
- To respond to all operator commands directed toward any printer or punch.
- To queue jobs for the next stage of processing when the current print/punch function has been completed.

The Output Service Processor is coded re-enterably in such a way that it can deliver output to a number of different output devices simultaneously. The re-enterability is attained by retaining all storage unique to a job in the Processor Control Element (see figure 4.3.1) which must be unique for each output device.

4.3.2 OUTPUT SERVICE PROCESSOR - PROGRAM LOGIC

The Output Service Processor is divided into three phases, nine subroutines, and two non-process exits. This section will give a functional description of each of these phases, subroutines, and exits to aid the system programmer in gaining a working knowledge of the processor.

PHASES

Phase 1 -- Processor Initialization

The Initialization Phase begins by attempting to get an output unit. If an output unit is not available, the processor enters a HASP $WAIT state until a device is made available and then the process is repeated.

Next, an output function is determined. If the device acquired is a remote printer, the appropriate entry in the Remote Message Table is examined to determine if any remote messages have been queued, and if so processing continues. The general purpose register: "JCT" is set to zero to indicate that remote messages are being processed.

If the device is not a remote printer, or if there are no messages queued, an attempt is made to obtain a job from the Job Queue which matches the type, routing and special forms of the device obtained. If no jobs are queued which fit these qualifications, the special forms processing type is checked to see if the forms requirement can be dropped. If so, another attempt is made to obtain a job from the Job Queue which matches the type and routing specifications only.

If a job cannot be found, then the output unit is released and control is returned to the start of the Initialization Phase.

If the output device is a remote terminal, output activity is initiated by calling upon the Remote Terminal Access Method (RTAM) to "open" the Remote Device Control Table.

The processor then acquires a direct-access Device Control Table (DCT) and a HASP buffer into which the Job Control Table (JCT) is then read. A message is sent to the operator notifying him that a particular job is now on the respective device and the initialization of the Processor Control Element Work Area (see figure 4.3.1) is completed.

If the processor is processing print output, and if the output is not a data set which has been routed for special forms, the PRINTID subroutine is called to generate the print identification header and control is transferred to Phase 2.

If the processor is processing punch output, and if the output is not a data set which has been routed for special forms, the Punch ID Card is generated for later punching, and control is transferred to Phase 2.

Phase 2 - Main Processor

The function of the Main Processor is to read the data blocks which are produced by the Execution Control Processor and build a channel program to print or punch the data. The PRDBUF and PRDCHK subroutines are used to read the data blocks, the PPPUT subroutine is used to construct the channel program and the PPWRITE and PPCHECK subroutines are used to initiate and check the execution of the channel program.

If the processor is processing print output, the "Control Byte" fields of the Data Block (see figure 8.15.1) are used to build the CCW operation codes. These control bytes are also used to count the actual lines of paper spaced and when this line count exceeds the parameter JCTLINCT, an eject is inserted to force a new page and the count is restarted. If an illegal control byte is encountered, or if the operator has entered a "$T PRTn,C=1" command, a single-space CCW is generated and used rather than the one provided in the data block. In such cases line counting continues and automatic page overflow is still provided.

If the processor is processing punch output, a "Punch, Feed, and Select Stacker P2" command is generated.

When the last data block has been printed or punched, control is transferred to Phase 3.

Phase 3 - Processor Termination

The Processor Termination Phase first reads the Job Control Table and scans the Peripheral Data Description Blocks (see figure 8.8.1) for the next data set to be processed. If another data set is encountered, control is returned to Phase 2 for processing. If no more data sets are to be processed, the termination phase then proceeds depending upon the type of output which is being processed.

If the processor is processing print output, the "Print Copy Count" field in the JCT (see figure 8.8.1) is compared with the current number of copies which have been printed. If more copies are needed, control is transferred to Phase 1 for the production of another copy. If no more copies are required, the PRINTID subroutine is called to generate the print identification trailer.

If the processor is processing punch output, the job accounting subroutine is called, and the accounting card is punched followed by a blank card to clear the punch and check the punching of the Job Accounting Card.

The Job Control Table is then re-written, the Job Queue Element is passed to the next processor queue, the Device Control Tables are released, and control is transferred to the start of Phase 1.

SUBROUTINES

PLOADUCS -- Subroutine to Load the UCSB and FCB

This subroutine determines the Universal Character Set Type from the Printer Device Control Table. The UCSB Table is then searched and the corresponding UCS image (if one is found) is $LOADed and moved into a HASP buffer. The UCS Buffer is then loaded using the PPPUT, PPWRITE, and PPCHECK subroutines.

If the output device type specifies a 3211 printer, then the Forms Control Buffer is loaded in a manner similar to the UCS Buffer. After loading the FCB, the FCB type is reset so that no more FCB loads will occur until the operator specifies that the buffer should be re-loaded.

PRINTID -- Subroutine to Generate Print Identification

This subroutine builds up the line image which is used to produce the Print Identification Page from information in the Job Control Table and information passed to the subroutine at the time it is called. This line image is built up in the "Job Accounting Storage" section of the Job Control Table (see figure 8.8.1). The subroutine then builds a channel program which starts with an eject command and follows with enough print commands to completely fill a page with print identification lines. The channel program is then executed and checked and control is returned to the calling program. The PPPUT subroutine is used to construct the channel program, and the PPWRITE and PPCHECK subroutines are used to initiate and check the execution of the channel program.

PPFORMCK -- Subroutine to Mount Forms

This subroutine compares the forms being requested with the forms currently mounted on the associated device. If a match is found, the subroutine returns immediately. Otherwise, a forms mount message is issued to the operator and the subroutine $WAITs for a "$Sdevice" command to be entered. The DCT Forms field is then set to reflect the new forms type and processing continues.

PRCOMENT -- Subroutine to Add Comment to Printer Output

This subroutine constructs and adds to the printer output (using the PPPUT, PPWRITE, and PPCHECK subroutines) a comment of the form:

PRINT xxxxxxxxx BY OPERATOR.

"xxxxxxxxx" is specified at subroutine entry by parameter register "R1" and will be one of the following:

DELETED
RESTARTED
REPEATED
BACKSPACED
FWD-SPACED
SUSPENDED

PRDBUF -- Subroutine to Initiate Read from Direct Access Storage

This subroutine initiates a read from the track address specified by register "PNP" into the appropriate HASP buffer.

PRDCHK -- Subroutine to Check Read from Direct Access Storage

This subroutine checks the read initiated by the PRDBUF subroutine. If the read is not complete, the processor is placed into a HASP $WAIT state until the read is completed. If an I/O error is detected, a "$IOERROR" macro-instruction is issued and the processing of the rest of the data set is deleted.

This subroutine also checks for any operator command which would cause the Main Processing Phase to be completed and forces any indicated completion by zeroing the chain track in the data block just read.

PPPUT -- PPUTOLAY -- Subroutine to Build a Channel Program

This subroutine accepts a CCW from the calling program and, if the output device is not a remote terminal, constructs a channel program in the Processor Control Element Work Area (see figure 4.3.1). Each command is examined and if it is an immediate printer space or skip, and if the previous command was a "Write, No Space", the two commands are combined into one. When the channel program storage area is full, this subroutine calls the PPWRITE subroutine to initiate the execution of the channel program. Upon the next entry, the execution of the channel program is checked by calling the PPCHECK subroutine.

If the output device is a remote terminal, the Remote Terminal Access Method is "called" to process the output line or card. Control is then given to the PPCHECK subroutine to test for operator commands.

PPWRITE--Subroutine to Initiate Execution of the Channel Program

If the output processor is being deleted by operator action, this subroutine returns immediately. Otherwise a write is initiated on the respective output device, using the channel program developed by the PPPUT subroutine.

PPCHECK--Subroutine to Check the Execution of the Channel Program

This subroutine checks for the successful completion of the channel program execution initiated by the PPWRITE subroutine. If the execution has not yet completed, the subroutine enters the processor into a $WAIT condition until the output has been completed. If an unsuccessful completion is detected, the subroutine performs the error recovery described in the paragraph below. This subroutine also interprets all operator commands directed at the processor and initiates appropriate action.

NON-PROCESS EXITS

The following routines are used to place the Output Service Processor into a HASP $WAIT state if a HASP resource is not available. In both cases the non-process register ("PNP") must have been set to the restart address before the routine is entered.

- PNOUNIT -- A HASP unit was not available.
- PNOBUF -- A HASP buffer was not available.

When the respective resource is made available, the processor is $POSTed and another attempt is made to acquire the resource.

PRINTER "WARM START" LOGIC

When the Output Service Processor is successful in acquiring a job from the print queue, the print checkpoint area is searched for an available Print Checkpoint Element (see figure 4.3.2). This element is thereafter used to record the job number, copy count, and line and page counts.

In the event of a "warm start", the elements are searched and each Print Checkpoint element is moved into the Job Control Table for the job which it represents.

When the job is printed, the JCT is examined, and if the Print Checkpoint Element is present, the processor continues printing from the point when the last checkpoint was taken.

OUTPUT PROCESSOR BUFFER LOGIC

The buffer logic that the output processor employs is determined by the HASPGEN parameters: $PRTBOPT, $PUNBOPT, $RPRBOPT, and $RPUBOPT.

Buffer Option = 1

One buffer will be obtained at the beginning of output processing and will be used through the entire processing of a job's output. A read for the following data block will not be initiated until the current data block has completed its output. Periods of high Input/Output activity could cause the printers and punches to operate at less than their maximum rate when this option is used.

Buffer Option = 2

Two buffers will be obtained at the beginning of output processing and will be used through the entire processing of a job's output. A read for the following data block will be

initiated as soon as the previous data block has completed its output and will be performed while the current data block is completing its output. This option represents the most efficient utilization of the output devices.

PRINT AND PUNCH ERROR RECOVERY

Print Errors

The operator will be informed of all printer errors, but they will be ignored by the Output Service Processor.

Punch Errors

The card which causes a punch check and the card following this card are selected automatically into the reject stacker. The Output Service Processor will attempt to punch these two cards correctly until no error occurs or the operator deletes the job. Since all normal punch output is selected to another stacker, no operator intervention will be required to clear the punch. Every error will be recorded on the operator's console.

Figure 4.3.1 -- OUTPUT SERVICE PCE WORK AREA FORMAT

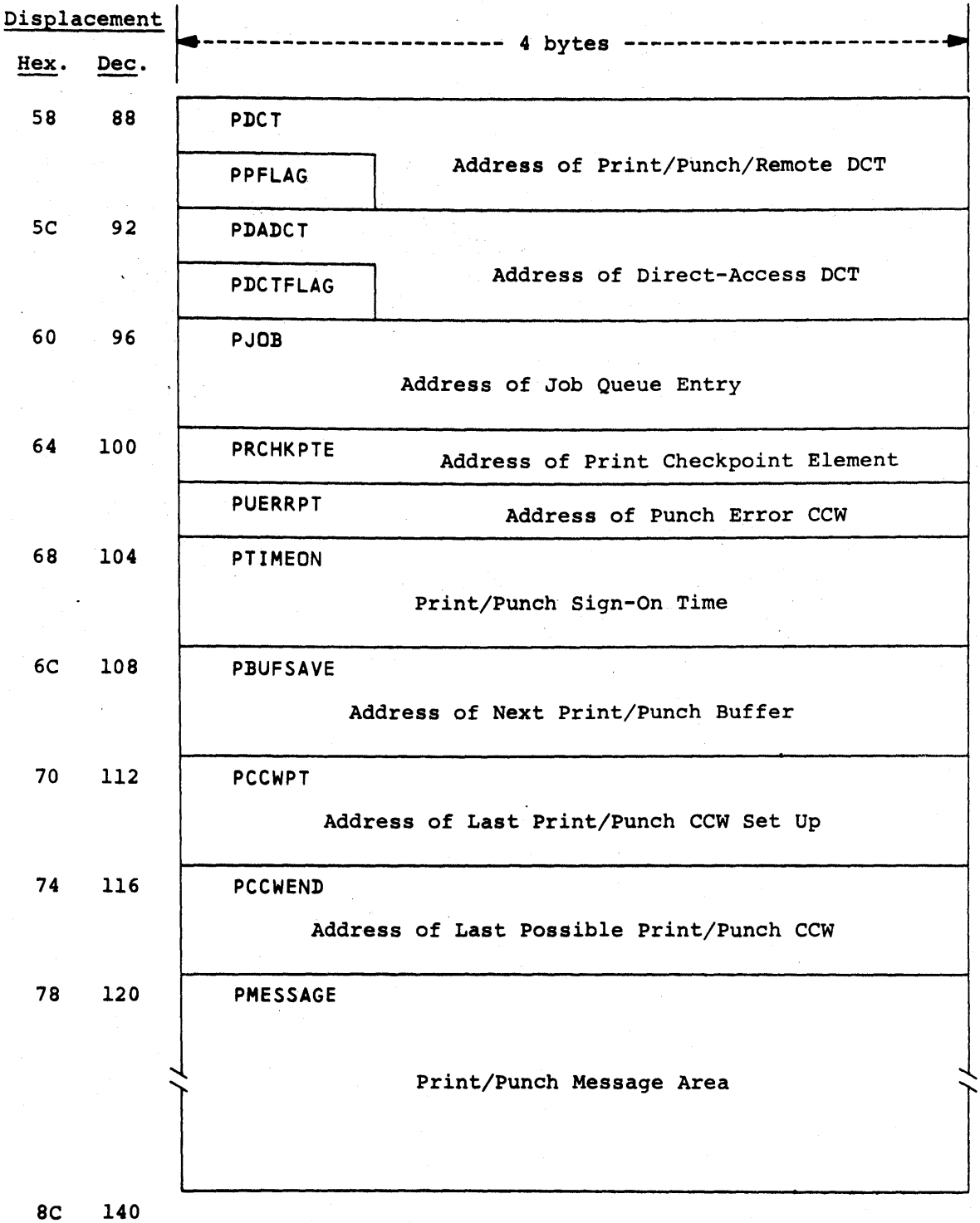


Figure 4.3.1 -- OUTPUT SERVICE PCE WORK AREA FORMAT (CONTINUED)

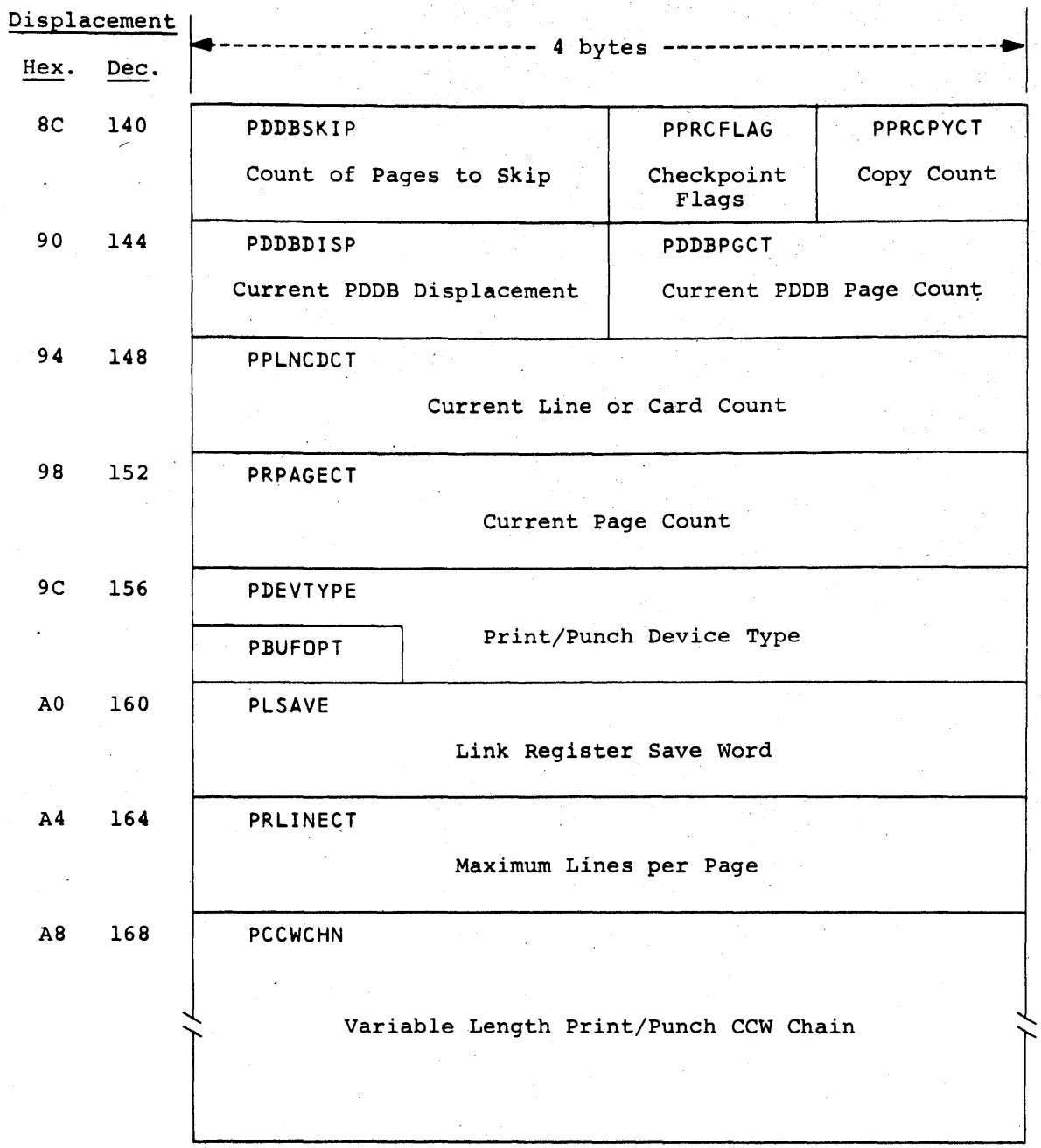


Figure 4.3.1 -- OUTPUT SERVICE PCE WORK AREA FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
58	88	PPFLAG	1	Print/Punch Synchronization Flags --

Bit	Name	Meaning
0	PPWSW	Write has been Initiated.
1	PPDELSW	Function has been Deleted.
2	PPNOJOB	No Job is Active.
3	PRDELSW	Print was Deleted by Operator.
4	PRRSTSW	Print was Restarted by Operator.
5	PPRDERR	Function Terminated by Read Error.
6-7		Reserved for Future Use.

58	88	PDCT	4	Address of Print/Punch/Remote Device Control Table.
5C	92	PDCTFLAG	1	Print/Punch/Remote Operator Commands --

Bit	Name	Meaning
0	DCTSTOP	$Z ($STOP) Command.
1	DCTDELET	$C ($DELETE) Command.
2	DCTRSTRT	$E ($RESTART) Command.
3	DCTRPT	$N ($REPEAT) Command.
4	DCTBKSP	$B ($BACKSPACE) Command.
5	DCTSPACE	$T...,C=1 Command.
2+4		$I Command.
6-7		Reserved.

5C	92	PDADCT	4	Address of Direct-Access Device Control Table.
60	96	PJOB	4	Address of Job Queue Entry.
64	100	PRCHKPTE	4	Print Only: Address of Print Checkpoint Element.
64	100	PUERRPT	4	Punch Only: Address of Punch Error CCW.
68	104	PTIMEON	4	Print/Punch Sign-On Time.
6C	108	PBUFSAVE	4	Address of Next Print/Punch Buffer.

Figure 4.3.1 -- OUTPUT SERVICE PCE WORK AREA FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
70	112	PCCWPT	4	Address of Last Print/Punch CCW Set Up.
74	116	PCCWEND	4	Address of Last Possible Print/Punch CCW.
78	120	PMESSAGE	20	Print/Punch Message Area.
8C	140	PDDBSKIP	2	Count of Pages to Skip.
8E	142	PPRCFLAG	1	Checkpoint Flags --

Bit	Name	Meaning
0	PRCHKUSE	Checkpoint Element Assigned.
1	PRCHKJOB	Job Active Indication.
2-7		Reserved.

8F	143	PPRCPYCT	1	Current Copy Count.
90	144	PDDBDISP	2	Current PDDB Displacement.
92	146	PDDBPGCT	2	Current PDDB Page Count.
94	148	PPLNCDCT	4	Current Line or Card Count.
98	152	PRPAGECT	4	Current Page Count.
9C	156	PBUFOPT	1	Buffering Option --

Value	Meaning
1	Single Buffering.
2	Double Buffering.

9C	156	PDEVTYPE	4	Device Type from UCB (UCBTYP).
A0	160	PLSAVE	4	Link Register Save Word.
A4	164	PRLINECT	4	Maximum Lines per Page.
A8	168	PCCWCHN		Variable Length Print/Punch CCW Chain.

Figure 4.3.2 -- PRINT CHECKPOINT ELEMENT FORMAT

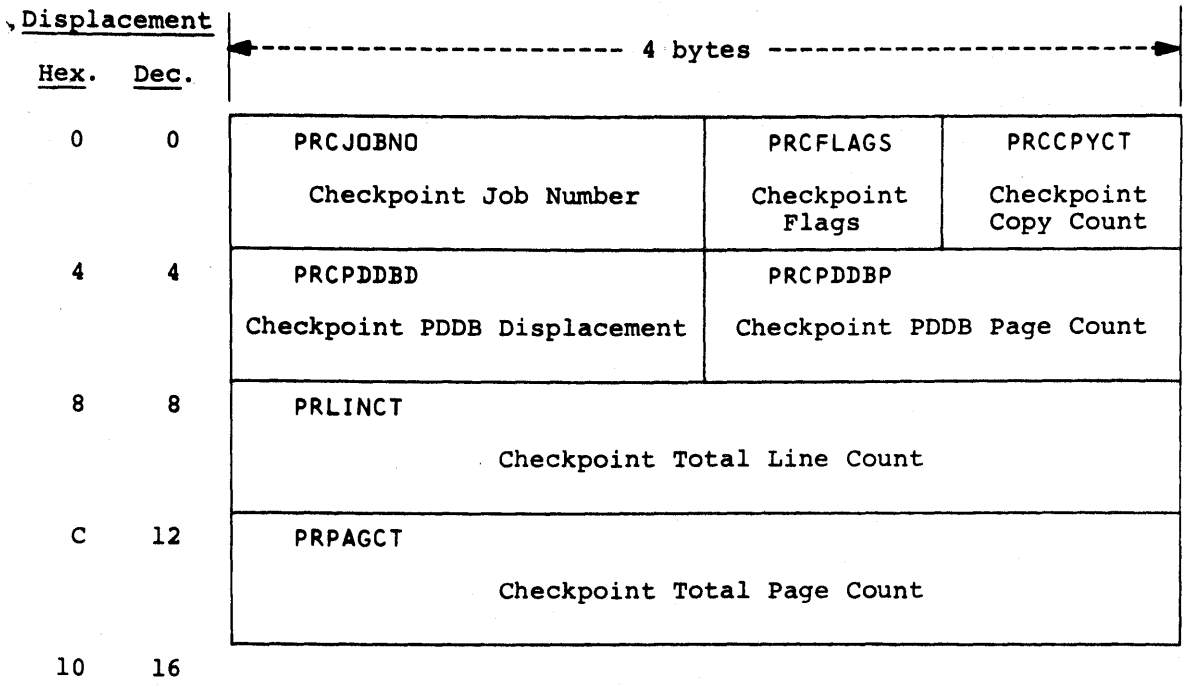


Figure 4.3.2 -- PRINT CHECKPOINT ELEMENT FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
0	0	PRCJOBNO	2	Job Number.
2	2	PRCFLAGS	1	Checkpoint Flags --

Bit	Name	Meaning
0	PRCHKUSE	Checkpoint Element in Use.
1	PRCHKJOB	Job Active Indication.
2-7		Reserved for Future Use.

3	3	PRCCPYCT	1	Current Copy Count.
4	4	PRCPDDBD	2	Current PDDB Displacement.
6	6	PRCPDDBP	2	Current PDDB Page Count.
8	8	PRLINCT	4	Total Line Count.
C	12	PRPAGCT	4	Total Page Count.

4.4. PURGE PROCESSOR

4.4.1 PURGE PROCESSOR - GENERAL DESCRIPTION

The Purge processor frees the job's acquired HASP direct-access space and removes the Job Queue Element from the system.

4.4.2 PURGE PROCESSOR - PROGRAM LOGIC

The processor first acquires a Job Queue Element and issues the $ACTIVE macro to inform the HASP Dispatcher that the processor is active. Then a direct-access Device Control Table (DCT) and a HASP buffer are acquired and initialized so that the job's Job Control Table (JCT) may be read into the buffer from the SPOOL disk. If a DCT or buffer is not available this processor will be placed in a HASP $WAIT state until a DCT or buffer can be acquired. If no permanent I/O errors occur while reading the JCT, a $PURGE macro instruction is then issued to return the job's direct access tracks. If a permanent I/O error occurs while the JCT is being read, the DISASTROUS error routine is called and the $PURGE macro instruction is not executed. Next, the Job Queue Element is removed from the HASP Job Queue and the following message is issued to the operator:

```
JOB xxx IS PURGED
```

Finally, the buffer and DCT are freed, and the $DORMANT macro instruction is issued to indicate to the HASP Dispatcher that the processor is inactive and control is returned to the start of the routine for the processing of the next job to be purged.

4.5 HASP COMMAND PROCESSOR

4.5.1 HASP Command Processor - General Description

The HASP Command Processor receives all HASP commands entered from acceptable local or remote HASP input sources. The Processor is responsible for decoding each command and performing the processing necessary to cause appropriate action to the operator's request.

4.5.2 HASP Command Processor - Program Logic

The HASP Command Processor is initially entered at the beginning of the Control Section (CSECT) HASPCOMM which is a part of the resident portion of HASP. Subsequent re-entries are returns from the various command sub-processors with optional requests for the displaying of the "OK" message or other message contained in the COMMAND area of the PCE. After displaying any requested replies the HASP Console Message Buffer queue $COMMQUE is examined for the presence of the next command to process. If no buffer is queued, the Command Processor waits on WORK. When $POSTed or if a buffer is present upon entry, the Command Edit Routine is entered via $LINK macro.

Command Edit Routine - HASPCOME

VERB CONVERSION - The Command Edit Routine converts the command text from the long form to the standard single character verb form. The data portion of the Console Message Buffer up to the first comma (,) or apostrophe (') is made upper case and non-blank characters are shifted to the left. The resulting text is compared against arguments in the VERB CONVERSION TABLE. If a match is found, the corresponding standard form of the command is substituted.

COMMAND EDIT AND BREAK OUT - The information in the HASP Console Message Buffer is moved to the COMMAND field in the PCE work area. The two bytes CMBFLAGS and CMBCONS of the buffer are moved to the COMFLAGS and COMROUTE fields of the PCE workarea. These two bytes when combined with the two succeeding bytes in the PCE form the list form of the $WTO used for all responses to the operator from the Command Processor.

The COMMAND area of the PCE is primed with blanks and the buffer is scanned. Solid characters are ORed (moved with upper casing) into the COMMAND area. Blanks encountered in the buffer will

normally be skipped (blank elimination); however, if an apostrophe is encountered, blanks will not be skipped until the next apostrophe. Double apostrophe characters will cause the blank compression status to remain as previously set; however, the second apostrophe of the pair will be eliminated.

As each comma is encountered an entry of the next available character position is made in the COMPNTER area of the PCE. (The first entry is the address of the character after the verb. The second is the address of the second operand, etc.) When the COMPNTER area is full, recording is discontinued. Upon completion of the scan, the buffer is released, the COMNULOP field in the PCE is set to the address of the second character beyond the last solid character (null operand), and the operand pointers are shifted down adjacent to the COMMULOP field (see Figures 4.5.1 to 4.5.3). Control registers are set as follows:

WD = address of the first operand pointer in the COMPNTER field
WE = 4
WF = address of the last operand pointer in the COMPNTER field

SELECTING THE COMMAND SUB-PROCESSOR - The SELECTION TABLE is used to determine the appropriate command sub-processor which must be entered. Starting with the first element, the SELECTION TABLE is scanned for a matching verb. When the verb is located, the first character of the first operand is then used for comparing. If a match is found on the operand or if the table entry contains an X'FF' for operand argument, the table entry for the command is considered "located". If the end of the entries is encountered for the verb or table, the command is considered invalid and the edit routine returns to the main processor with INVALID COMMAND message in the COMMAND area for display. (See $COMTAB macro in Section 4.5.4 for format of the SELECTION TABLE element.)

VALIDATING THE SOURCE AND ENTERING THE SUB-PROCESSOR - Each entry of the SELECTION TABLE may have restriction indicators as follows:

COMRMT = 1 - Reject remote sources
COMS = 1 - Reject consoles which are restricted from entering SYSTEM COMMANDS
COMD = 1 - Reject consoles which are restricted from entering DEVICE COMMANDS
COMJ = 1 - Reject consoles which are restricted from entering JOB COMMANDS

The restriction indicators correspond with the restriction indicators which appear in the COMFLAGS field. The COMFLAGS indicator is previously set from the CMBFLAGS field of the HASP Console Message Buffer which in turn is set by other HASP processors as follows:

1. CMBFLAGS when set by the remote console processor or remote reader processors will contain the remote indicator. This indicator corresponds to COMRMT bit in the SELECTION TABLE.

2. CMBFLAGS when set by the local console support routines will contain the restriction flags assigned to the Console Device Control Table. (Restriction is the opposite of authority which is set by the operator command $TCONn,A=authority or by the system programmer.)
3. CMBFLAGS when set by the OS console interface is the OS authority indicators inverted with the Exclusive Or Immediate (XI) instruction.

The restriction indicators are used as the second operand of a Test Under Mask (TM) instruction. If any restriction indicator in the COMFLAGS field corresponds to any restriction indicator in the SELECTION table entry, the command is rejected as invalid. Otherwise Register 1 is set with the value in the SELECTION TABLE entry COMTOFF field and control is passed to the CSECT indicated by the Overlay Constant ($OCON) field of the SELECTION TABLE element via the $XCTL macro.

Command Sub-Processor Control Sections

The Entry routine of each command sub-processor control section will, if applicable, use the offset value in register 1 (set by the edit routine) to determine the "relative" entry point for the designated sub-processor. Normally the sub-processor is entered directly by the special Command Processor macro: "Branch Relative Register" on R1 ($BRR R1). However, some control section entry routines will pre-process the operands of the command prior to entering the sub-processor. Each sub-processor performs the desired functions and returns to the main command processor for the next command.

4.5.3 HASP Command Processor Organization

The HASP Command Processor is created by a single assembly with multiple Control Sections (CSECT). The main CSECT HASPCOMM is the only portion of the Command Processor that is part of the HASP resident load module. It contains all V type address constants required by the sub-command processors and all "BASE2" service routines. The Command Edit Routine HASPCOME receives control from the main processor and determines which COMMAND SUB-PROCESSOR CSECT to enter for processing of the command entered. One or more of the various COMMAND SUB-PROCESSOR CSECTs are used in processing each HASP operator command. Although the physical CSECTs are organized in accordance with the size of the overlay work area , the logical organizational grouping is as follows:

JOB QUEUE COMMANDS
JOB LIST COMMANDS
MISCELLANEOUS JOB COMMANDS
DEVICE LIST COMMANDS
SYSTEM COMMANDS
MISCELLANEOUS DISPLAY COMMANDS
REMOTE JOB ENTRY COMMANDS

HASP Command Processor Workarea

The HASP Command Processor PCE workarea shown in Figure 4.5.1 is the primary workarea for the processor and is the only area which may be used to save information in the event a $WAIT is issued by the processor or any of the "BASE1" service routines on behalf of the processor. The fields are generally used as described in the following paragraphs.

COMFLAGS to COMCLASS - This field contains a list form of the $WTO macro. The $WTO is referred to by a single execute form of the $WTO located within the resident portion of the Command Processor which is used for all operator messages generated by any routine within the processor. The CMBFLAGS and CMBCONS fields of the HASP Console Message Buffer for each command is inserted into the COMFLAGS and COMROUTE cells and are used to provide correct route codes for replies. The three low order bits of COMFLAGS are restriction indicators and are set to zero prior to each $WTO reply.

COMEWORK - This field is used as a workarea and by function routines identified by the macro instructions as follows:

macro	contents upon exit from routine
$CFCVE	last character is blank
$CFDCTL	first four characters of requested device name
$CFJDCT	address of HASP job queue element for requested job
$CFJMSG	same as $CFCVE

COMDWORK - This field is aligned on a double word boundary and is used as a workarea and by function routines identified by the macro instructions as follows:

macro	contents upon exit from routine
$CFCVE	five character number in EBCDIC with leading blanks
$CFDCTL	last four characters of requested device name
$CFJMSG	same as $CFCVE

COMMAND - This field contains the compressed form of the operator command with trailing blanks at the time each command sub-processor is entered. The command is overlayed by the reply message text for all $WTO messages issued by any Command Processor routine. Some command sub-processors use the area as scratch areas and in some cases the right end for storage of critical information while message replies are generated in the left end of the area.

COMPNTER-COMNULOP - These fields are set by the Command Edit Routine and are used to locate the beginning of each of the specified operands in the command currently being processed. COMNULOP contains a pointer to the second character beyond the last operand specified, i.e., points to a non-existant or "null" operand. Operand 1 through n pointers are right adjusted in COMPNTER so that operand n pointer is adjacent to the "null" pointer (see Figures 4.5.2 and 4.5.3 for illustrations). Command sub-processors use these areas for additional workspace after the operand pointers are no longer needed. Examples of other uses are listed as follows:

1. Job queue command $DN and $DQ commands place queue scanning control elements in the COMPNTER area.
2. Job list commands place the job range numbers (j-jj) in the corresponding operand pointer element area.
3. $DR uses the right end of the COMMAND area and COMPNTER-COMNULOP area to hold the reply ID numbers.

Figure 4.5.1 -- HASP COMMAND PCE WORK AREA FORMAT

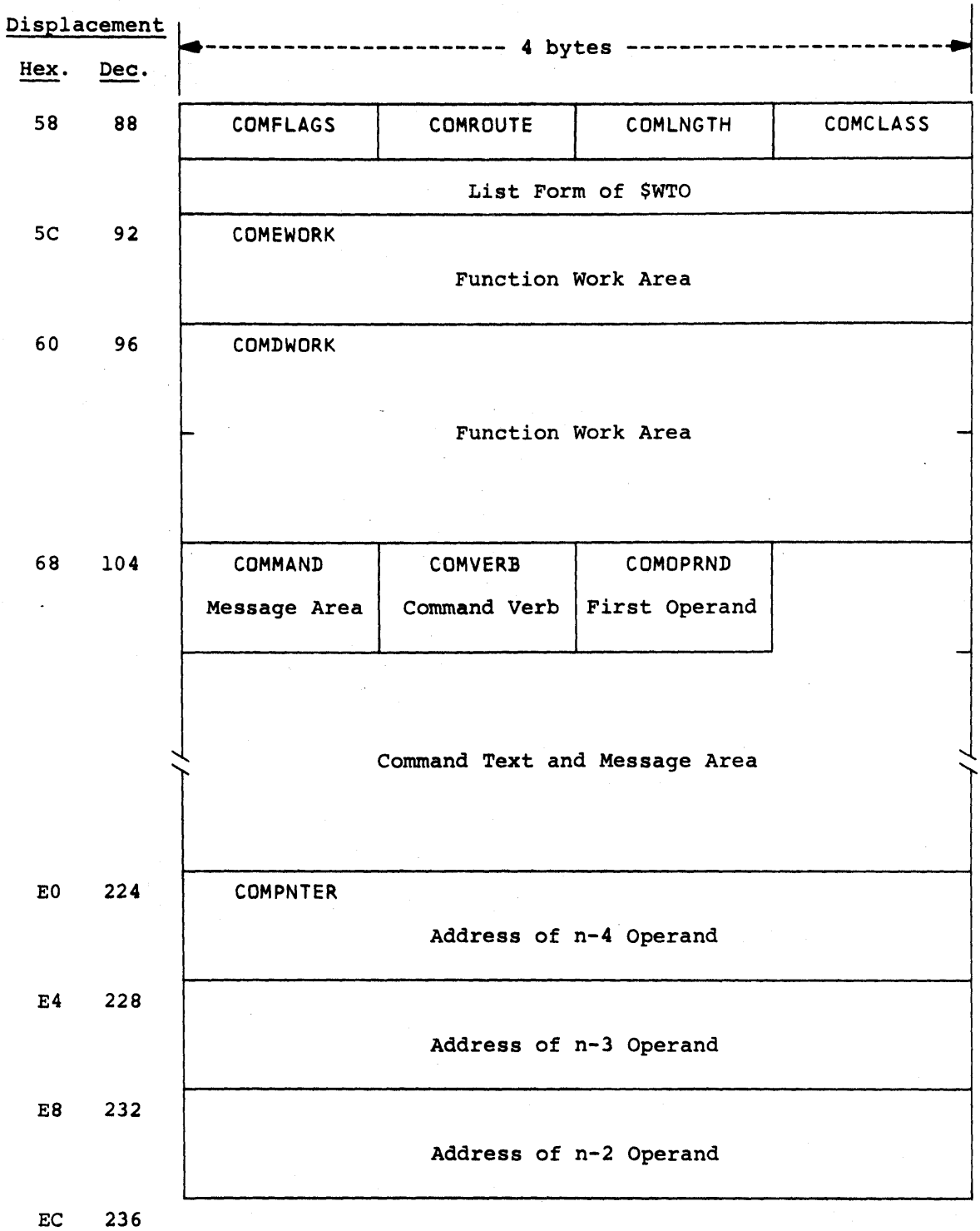


Figure 4.5.1 -- HASP COMMAND PCE WORK AREA FORMAT (CONTINUED)

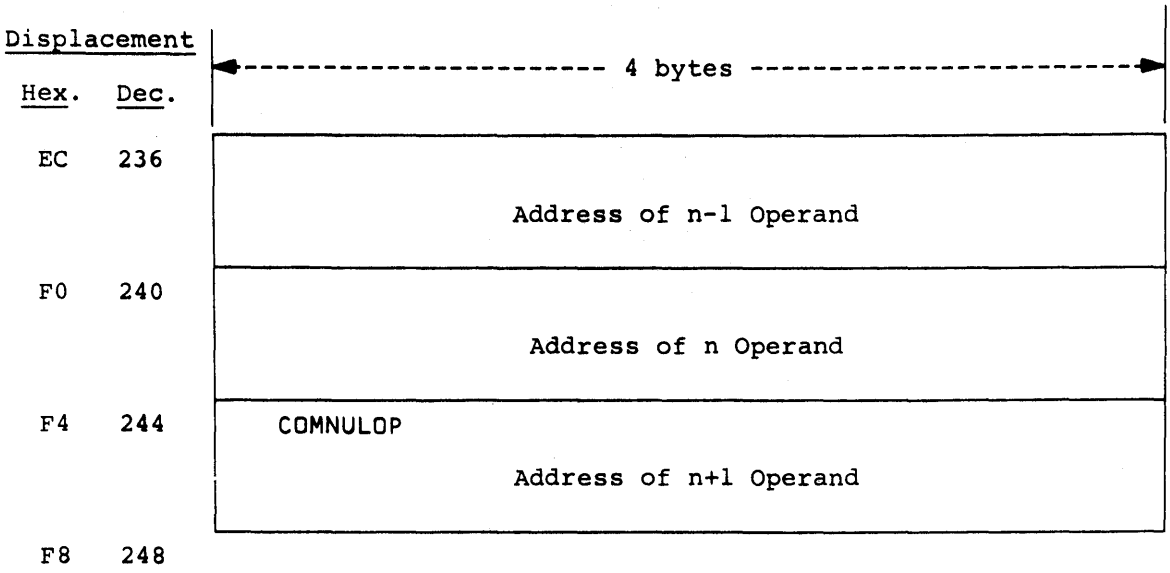
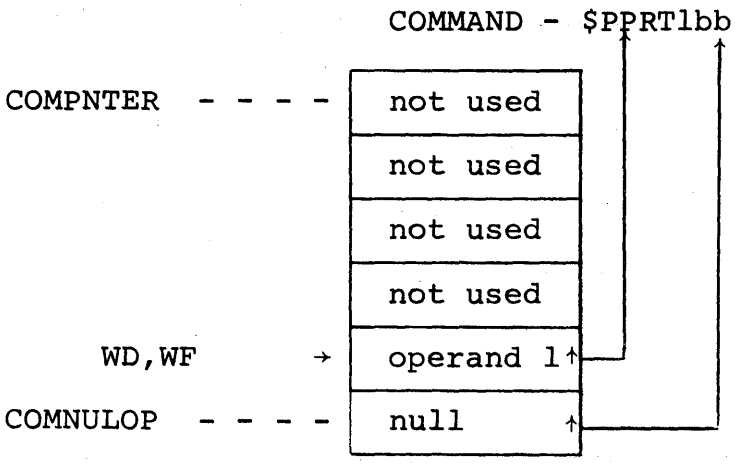


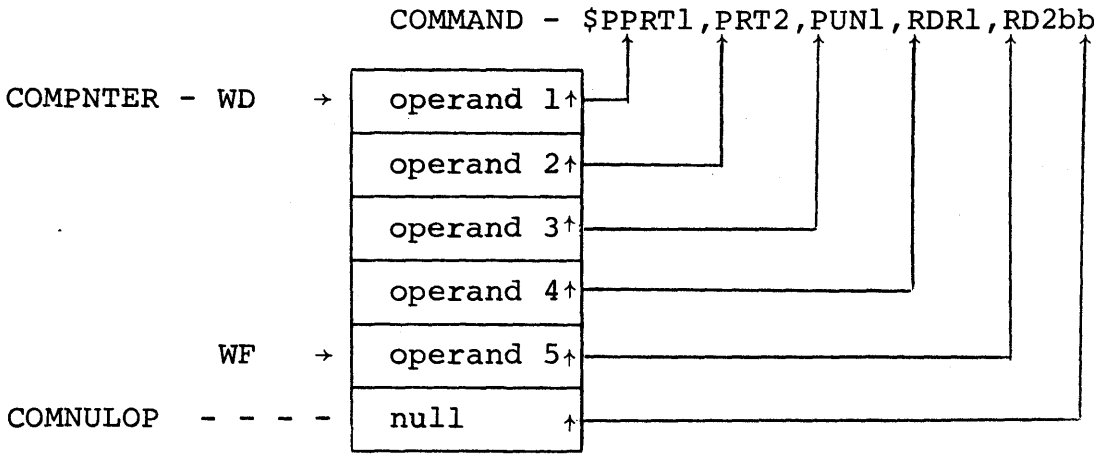
Figure 4.5.2 COMMAND - COMNULOP Areas With Single Operand Command



Upon exit of Edit Routine, Registers WD, WE and WF are set for testing instructions:

BXLE WD,WE,loop
(for next operand)
BXH WD,WE,exit
(if no more)

Figure 4.5.3 COMPNTER -COMNULOP Areas With Five Operand Commands



NOTE: b = blank character

Upon exit of Edit Routine, Registers WD,WE and WF are set for testing instructions:

BXLE WD,WE,loop
(for next operand)
BXH WD,WE,exit
(if no more)

Coding Conventions

The symbols with the command processor conform to the following conventions:

1. All main processor, Edit Routine, and PCE workarea symbols start with the characters "COM".

2. All Function macro generated symbols start with "COF".

3. All command sub-processors have entry point symbols of the following form:

form	example	command	comments
Cvo	CDN	$DN	v = the verb of the command o = the first operand character
Cv	CB	$B device	single character identifier
Cvxx	CD7D	$D'jobname'	apostrophe is hexadecimal 7D

4. All symbols created for the support of the command will start with characters which identify the entry point (CDNxxxxx identifies a location which was originally written for the $DN command). Commands with no unique operand character symbol have the character "x" as the third character. (CBX..... identifies a location which was originally written for the $B device command.) These conventions may be altered in cases where the command identification characters are redefined after original development.

5. The main processor CSECT is HASPCOMM, all other CSECTs are defined via the symbol field of the $COMGRUP macro; specified starting with the characters "HASPC".

Register Conventions

The Command Edit Routine passes control to the control section (CSECT) which contains the appropriate command sub-processor. At the point the Command group entry routine receives control, the registers will contain the following:

reg	contents
R0	unpredictable
R1	entry offset from the Command entry offset
WA	unpredictable
WB	unpredictable
WC	unpredictable
WD	first operand pointer (zero if no operand)
WE	4
WF	last operand pointer
BASE3	base for CSECT
BASE1	HCTDSECT address
BASE2	beginning of main Command Processor
SAVE	PCE address
LINK	unpredictable
R15	unpredictable

If more than one command appears within the group, the value of register R1 will be set by the $COMGRUP entry routine to a value so that a $BRR R1 will enter the command sub-processor.

4.5.4 HASP COMMAND PROCESSOR MACROS

To facilitate flexibility in the development and possible modification of the Command Processor a macro package is included within the assembly source deck. This section is intended to supplement the HASP Command Processor Source listings obtainable from the HASP generation and assembly process in assisting the user to understand the generated code as specifically used in the current HASP as distributed.

Each HASP Command Processor macro may be dependent upon the definitions contained within the Command Processor source deck as well as other members of the HASP source library. These macros are catagorized as follows:

ORGANIZATIONAL - Macros which provide basic definitions and are closely associated with the organization of the processor.

BASE2 SERVICES - Macros which call upon the main Command Processor to perform a service (display a reply).

CONDITIONAL IN-LINE FUNCTIONS - Macros which perform the function in-line or links to a routine which performs the desired function.

RELOCATABILITY AIDS - Macros which assist in keeping the overlay CSECT relocatable around $WAIT or implied $WAIT situations.

The macros which are supplied under each category are summarized in Table 4.5.4. The following conventions are used in specifying parameter requirements:

"parameter=** -" - keyword parameter is required

"parameter=text -" - the assumed value if the keyword parameter is not specified

"parameter -" - the parameter is an optional positional parameter

"parameter - Required" - the parameter is a required positional parameter.

Table 4.5.4 Command Processor Macro Summary

Op-Code	Definition
ORGANIZATIONAL:	
$COMWORK	COMMAND PROCESSOR WORKAREA (symbolic definitions)
$COMGRUP	DEFINE GROUP OF COMMAND SUB-PROCESSORS
$COMTAB	DEFINE COMMAND TABLE ELEMENT
BASE2 SERVICES:	
$CRET	RETURN TO MAIN COMMAND PROCESSOR
$CWTO	WRITE TO OPERATOR
CONDITIONAL IN-LINE FUNCTIONS:	
$CFCVB	CONVERT TO BINARY
$CFCVE	CONVERT TO EBCDIC
$CFDCTD	DEVICE CONTROL TABLE DISPLAY
$CFDCTL	DEVICE CONTROL TABLE LOCATE
$CFINVC	REPLY INVALID COMMAND
$CFINVO	REPLY INVALID OPERAND
$CFJDCT	FIND JOB'S DEVICE CONTROL TABLE
$CFJMSG	DISPLAY JOB INFORMATION MESSAGE
$CFJSCAN	SCAN JOB QUEUE ASSISTANCE
$CFSEL	SELECT A ROUTINE BASED ON CHARACTER
$CFVQE	VERIFY CONSOLE CONTROL OVER JOB
RELOCATIBILITY AIDS:	
$ARR	ADD RELATIVE REGISTER
$BRR	BRANCH RELATIVE REGISTER
$SRR	SUBTRACT RELATIVE REGISTER

Organizational Macros

$COMWORK - COMMAND PROCESSOR WORKAREA (symbolic definitions)
This macro adds to the PCEDSECT definitions for fields located in the Command Processor PCE workarea. Additional symbolic constants for BASE2 services and some externally defined parameters are defined.

$COMGRUP - DEFINE GROUP OF COMMAND SUB-PROCESSORS
This macro defines the Command Processor overlay control section via the $OVERLAY macro. It provides an optional entry point routine which locates the command sub-processor for the commands which belong to the group and sets register R1 to the relative address. (The symbol field must be specified for this macro.)

n positionals - Each positional specifies the command identification characters for the corresponding command sub-processor located within the group. Example:

specification	command	sub-processor entry point name
AA	$AA	CAA
DA	$DA	CDA
B	$B device	CB
C	$C device	CC
P40	$P	CP40
S40	$S	CS40
D7D	$D'jobname'	CD7D

PRTY=** - Priority of the HASP overlay defined by the macro.

DELAY=NO - The sub-processor will be entered via $BRR R1 macro instruction. If "YES" is specified R1 will contain the appropriate relative entry point address and control will be given to the statement following the macrc statement. (More than one positional must be specified if R1 is to be set or the branch is to be executed.)

$COMTAB - DEFINE COMMAND TABLE ELEMENT

This macro defines an element in the command SELECTION TABLE which is used by the Command Edit Routine for identifying legal commands, eliminating unauthorized input sources, and entering the correct command group CSECT.

verb - Required - The command identification character(s) corresponding to the $COMGRUP positional parameter specification for the command. No two $COMTAB macro statements may specify the same identification character string. All macro statements creating entries for the same command verb will appear in consecutive statements with the statement which specifies a single identification character last.

group - Required - The exact characters used in the specification in the symbol field of the appropriate $COMGRUP macro statement.

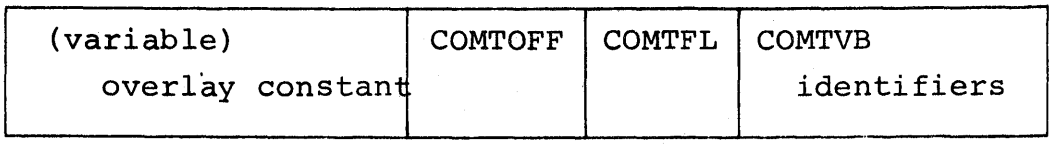
REJECT= - The command source rejection mask. One or more of the following symbols may be specified as follows:

"COMRMT"	- reject command if entered from a remote
"COMS"	- reject command if entered from a console not authorized for SYSTEM control
"COMD"	- reject command if entered from a console not authorized for DEVICE control
"COMJ"	- reject command if entered from a console not authorized for JOB control

Rejection of either a remote or a console not authorized for SYSTEM appears as follows:

"REJECT=COMRMT+COMS"

Figure 4.5.5 - Selection Table Element



COMTOFF = Offset for the overlay control section to locate the command sub-processor entry point.

COMTFL = Rejection flags.

COMTVB = Command identification characters. Verb with:

1. First character of the first operand.

2. X'FF'
   If X'FF' is specified all commands which have not been specified by the previous entries in the table will be considered "selected".

BASE2 Services

$CRET - RETURN TO MAIN COMMAND PROCESSOR

MSG= - "Address" of the message to be moved to COMMAND area for display. (L=operand of a non-register form is required.) "MSG=OK" indicates that the main processor is to display the OK message.

L= - "Value" representing the length of the message that is to be moved or has already been moved.

$CWTO - WRITE TO OPERATOR
REGISTERS USED: R0, R1, WA, LINK, R15

MSG= - "Address" of the message to be moved to COMMAND area and displayed. (L=operand of a non-register form is required.)

L=** - "Value" representing the length of the message that is to be moved or has already been moved.

Conditional In-Line Functions

The HASP Command Processor as distributed provides for the ability of the author of the command sub-processor to specify whether or not the code which performs the function is in-line or out of line. If an out of line routine is used the name and location of the subroutine must be defined. This is accomplished with parameters standard for all function macro instructions with the exception of $CFJSCAN as follows:

TYPE=CALL - The macro statement is not a definition form of the macro. "TYPE=DEF", the macro statement defines the subroutine form of the function and return linkage must be provided.

SYMBOL=address - The address of the "TYPE=DEF" version of the macro instruction. This indicates that only linkage to the "TYPE=DEF" version is to be provided. If neither "TYPE=DEF" or "SYMBOL=" parameters are specified the code will be generated in-line with no return linkage.

$CFCVB - CONVERT TO BINARY
This macro converts the numeric portion of a command operand to one or two numeric values.
REGISTERS USED: R0, R1, LINK, R15
R0 - contains the last number converted.
R1 - contains the next to last number converted (last number if the only one or the last is smaller than the previous).

POINTER=(R1) - The address of the COMPNTR field which addresses the operand containing one or more numerical values separated by dash (-).

NUM=2 - return two values. "NUM=1", one value is sufficient (R1 will be unpredictable on return).

NOK=** - Address of the error exit routine if the operand does not contain a number or if the number is too large.

$CFCVE - CONVERT TO EBCDIC
This macro converts the number in register (R0) to printable EBCDIC and sets the five resulting digits in the first five characters of the PCE area COMDWORK.
REGISTERS USED: R0, LINK

VALUE=(R0) - The positive binary half-word value to convert to EBCDIC. If the register form is not used, the value is contained within the addressed half-word.

$CFDCTD - DEVICE CONTROL TABLE DISPLAY
This macro displays the device name, unit address, and status of the DCT requested.
REGISTERS USED: R0, R1, WA, LINK, R15

DCT=(R1) - The address of the DCT to display.

$CFDCTL - DEVICE CONTROL TABLE LOCATE
This macro converts the abbreviated form of the device name to the long form (if abbreviated form is specified) and searches the DCT chain for a matching device.
REGISTERS USED: R0, R1, R15, LINK
R1 - contains the address of the DCT found or zero if no DCT found.

POINTER =(R1) - The address of the COMPNTER field which addresses the operand containing the device name (abbreviated).

$CFINVC - REPLY INVALID COMMAND
This macro returns to the Main Command Processor and causes the display "INVALID COMMAND".

$CFINVO - REPLY INVALID OPERAND
This macro moves eight characters, starting with the first character of the "current" operand to the COMMAND area and returns to the Main Command Processor causing the display of "operand INVALID OPERAND"

OPERAND=(R1) - The address of the operand to display.

$DFJDCT - FIND JOB'S DEVICE CONTROL TABLE
This macro searches the DCT chain for an active printer, punch, or reader DCT which is assigned to a procesor whose PCE contains a pointer to the HASP job queue entry belonging to the desired job. If the device is not found exit will be to the instruction immediately following the $CFJDCT statement (in-line code version); otherwise, exit will be to the instruction plus 4 (NSI+4).
REGISTERS USED: R1, LINK, R15

JOBQE=(R1) - The address of the HASP job queue entry for the desired job.

$CFJMSG - DISPLAY JOB INFORMATION MESSAGE
This macro sets into the COMMAND area of the PCE the information required for the JOB INFORMATION MESSAGE and displays the message.
REGISTERS USED: R0, R1, WA, LINK, R15

JOBQE=(R1) - The address of the HASP job queue entry for the desired job.

JDCT= - The address of the $CFJDCT TYPE=DEF macro which may be used to locate the job's DCT. Register form is prohibited.

CVE= - The address of the $CFCVE TYPE=DEF macro which may be used to convert numeric information to EBCDIC. Register form is prohibited.

JOB= - This parameter may be ignored by the macro; however, if specified as "JOB=SET" the text "JOBj" is assumed by the expanded routine to have been set in the COMMAND area for the desired job.

OPT= - This parameter may be ignored by the macro; however, if specified as "JOB=Q" all jobs given to the macro expansion are queued (not active) or if specified as "JOB=A" all jobs given to the expansion are active.

$CFJSCAN - SCAN JOB QUEUE ASSISTANCE

This macro is used to assist in scanning the job queue. As each entry is located the user's PROCESS routine is entered. The user examines the entry, performs whatever function desired on the entry, and returns to the symbol specified by the "NEXT=" operand. When the end of the queue is encountered, control is given to the instruction following the macro instruction. An optional feature of the macro is to allow the PROCESS routine an "IGNORE' entry to the generated code to indicate the current job entry is not acceptable to the PROCESS routine. If the "IGNORE=" option is specified the corresponding "EMPTY=", option is required. Register 1 is the scan register and is assumed to be unaltered by the user PROCESS routine. The "TYPE=DEF" option is not permitted for this macro.
REGISTERS USED: R1, BASE2
R1 - scan register
BASE2 - found/not found switch (in addition to processor base.

PROCESS=** - Address of the user's job queue element processing routine. Register form prohibited.

IGNORE= - Symbol to be used to define the entry to continue scan when the current job entry is not of the desired type.

NEXT=** - The symbol to be used to define the entry to continue scan when the current job entry is of the desired type.

EMPTY= - The name of the user exit routine desired to be entered when the job queue is found to be empty of jobs of the desired type. Register form is prohibited.

$CFSEL - SELECT A ROUTINE BASED ON CHARACTER
This macro matches the designated input character against a list of arguments and transfers control to the routine designated by the corresponding address. If no match is found, the next sequential instruction is entered.
REGISTERS USED: R1, LINK, R15

n positionals of form: (character, address) - Each positional "character" sub-parameter specifies an argument to compare against. The corresponding address sub-parameter indicates the address of the desired routine to enter if the character matches the argument. Register form is prohibited.

OPERAND=(R1) - The address of the designated input character to examine.

$CFVQE - VERIFY CONSOLE CONTROL OVER JOB
This macro tests COMFLAGS field of the PCE to determine if the input source is a remote. If the source is a remote, the not OK routine will be entered unless either the print or punch route codes for the indicated job specify the remote. Otherwise the OK routine will be entered.
REGISTERS USED: R1, LINK

JOBQE=(R1) - The address of the HASP job queue entry for the desired job.

OK= - Address of the routine desired to be entered if the console has control over the job. The address may be the symbolic register containing the address if specified as "OK=(register,BCR)" or "OK=(relative register,$BRR).

NOK= - Address of the routine desired to be entered if the console does not have control over the job. The address may be the symbolic register containing the address if specified as "NOK=(Register,BCR)" or "NOK=(relative register,$BRR). Either "OK=" or "NOK=" parameters must be specified.

Relocatability Aids

$ARR - ADD RELATIVE REGISTER
This macro instruction is used in conjunction with $SRR to restore the specified register to refer to the true address of relocated information.

register - Required - The symbolic register containing the address to be made true.

$BRR - BRANCH RELATIVE REGISTER
This macro instruction is used in conjunction with $COMGRUP to enter a sub-processor routine using the offset provided by the $COMGRUP routine.

condition - Condition required to be met in order to branch. If this parameter is omitted, no comma should be written to signify its omission. "Condition code" may be specified by the character strings: (E, NE, H, L, NH, NL, Z, NZ, P, M, NP, NM, O or NO).

Register - Required - The symbolic register containing the offset.

$SRR - SUBTRACT RELATIVE REGISTER
This macro instruction is used to make an address pointer relative for possible relocation before next referral to the information contained at the address.

register - Required - The symbolic register containing the address to be made relative.

4.6 OPERATOR CONSOLE ATTENTION PROCESSOR

This processor is included in HASP only if the value of the HASPGEN variable &NUMCONS is greater than 0 (see Section 7.1). The HASP interface to OS Console Support if &NUMCONS=0, is described in Appendix 12.15.

4.6.1 Operator Console Attention Processor - General Description

The function of this processor is to stage a read on a console whenever an attention is received from that console.

4.6.2 Operator Console Attention Processor - Program Logic

During HASP initialization, the first three words of the OS Console Attention Routine (IEEBA1) are overlayed with instructions which cause IOS to enter the HASPATTN routine of this processor whenever an attention interrupt occurs.

When an attention request is signalled by a console device, HASPATTN saves the device address in the processor's PCE workarea, $POSTs the PCE, and POSTs HASP.

When the Attention Processor is dispatched, it locates the physical console whose address is in the processor's PCE workarea and links to the $WTO Processing Routine (see Section 5.7) to queue a read on that console.

4.7 CHECKPOINT PROCESSOR

4.7.1 CHECKPOINT PROCESSOR - GENERAL DESCRIPTION

The purpose of this processor is to write the necessary information onto disk to affect a subsequent restart of the system. This processor will write the information at a predefined time increment and at the completion of each stage of each job.

4.7.2 CHECKPOINT PROCESSOR - PROGRAM LOGIC

The first entry into the Checkpoint Processor is into a section which initializes the processor. This section issues a &GETUNIT macro-instruction to obtain a DCT for a disk and completes this DCT by inserting the event wait field address, track to be written, and the buffer address.

The information to be checkpointed consists of the Job Queue which contains the status of each job in the system, the track allocation map which indicates the track groups of each disk that have been assigned, a save area which contains added information as to the status of the system, the print checkpoint table which is used to effect a warm start of the jobs being printed, and (if generated) the Job Information Table which contains additional information concerning each job in the system. The Job Queue and the Job Information Table reside within the checkpoint buffers, but the remaining fields must be moved into these buffers.

The track allocation map is the first to be moved and the track groups that have been reserved for the jobs that are currently executing and reading in are returned to the track allocation map to avoid loss of tracks in case of an emergency restart. Next the write buffers are completed by moving the save area and the print checkpoint tables. An $EXCP is issued to write the checkpoint buffers and a $WAIT on I/O is initiated.

The Job Information Table (if generated) is written with CCW's which are chained to the CCW's used to write the rest of the checkpoint information. The Job Information Table is not written with each checkpoint but only when the processor which requests the checkpoint indicates that he wishes the JIT to be written. This indication is made by setting the "JITJCKPT" bit in the "$JITSTAT" field to one.

At the completion of the I/O operation, the HASP ECB is posted and the timer is reset to a predefined time increment that was specified as a HASPGEN parameter. A test is now executed to determine if the previous write was successful and if so, a $WAIT macro-instruction is issued to place the processor into an inactive state until the time increment has expired or a stage of a job is completed.

If the previous write was unsuccessful, a message is issued to indicate to the operator that a restart is needed and a permanent HASP $WAIT state is entered so that no further checkpoint will be attempted.

4.8 ASYNCHRONOUS INPUT/OUTPUT PROCESSOR

4.8.1 ASYNCHRONOUS INPUT/OUTPUT PROCESSOR - GENERAL DESCRIPTION

Since the completion of all HASP I/O operations are signalled asynchronously with HASP operation via IOS channel-end appendages, these completions must be queued by the appendage until all HASP processors can be synchronized to receive the notification. The purpose, then, of the Asynchronous Input/Output Processor ($ASYNC) is to, at non-interrupt time, notify all processors of their I/O completions which were indicated by the OS I/O supervisor at interrupt time.

4.8.2 ASYNCHRONOUS INPUT/OUTPUT PROCESSOR - PROGRAM LOGIC

The buffers (and respective IOBs) associated with I/O channel-ends are chained, by the HASP channel-end appendages, for later processing by $ASYNC. In addition to the POST of the HASP task by IOS on any I/O completion, the channel-end appendages also $POST the Asynchronous Input/Output Processor to initiate its processing when the HASP task receives control. When $ASYNC receives control, it dequeues the first buffer from its chain of work (operating disabled, for this operation only, since its chain is updated at interrupt time). The Device Control Table entry (DCT) associated with this buffer is located and the active I/O count for the device is reduced by one. Next the user's EWF address is extracted from the buffer and interrogated, and action is taken according to the following algorithm:

EWF = 0 — User does not want notification of completion of I/O operation (always a write). The buffer will be returned to the HASP buffer pool by $ASYNC.

EWF > 0 — $POST the "I/O" bit in the EWF specified and take no further action.

EWF < 0 — Enter a user provided routine at the address specified by the absolute value of the EWF field. Addressability for the processor routine is established and the address given is entered via the Branch and Link instruction with the buffer address in register "R1." No further action is taken upon return by the processor.

After performing the indicated action, $ASYNC returns to dequeue the next buffer from its chain and the above procedure is repeated. When the end of the chain is reached, $ASYNC enters the $WAIT state until additional I/O completions occur.

4.9 HASP LOG PROCESSOR

4.9.1 HASP Log Processor - General Description

The function of the HASP Log Processor is to construct output buffers for eventual processing as part of each Job's printed output. Input to the Log Processor is through a queue of CMBs associated with the queue pointer $LOGQUE which is defined in the HCT. The nature of the information in the input queue, and consequently the printed output, varies as a function of the HASPGEN Parameters &NUMCONS and &WTLOPT.

4.9.2 HASP Log Processor - Program Logic

Log processing of a message buffer is started by locating the corresponding execution PCE. PCEs for output buffers are found by using the job number in the buffer, and "reply" message PCEs are located by using the TCB address which is placed into bytes six through eight of the buffer by the Operator Console Input/ Output Processor's asynchronous exit. Reply message processing is valid only for &NUMCONS>0.

A test is made to ascertain if the message will fit in the HASP buffer currently being used by the job for log output. If space is available, the message is placed in the HASP buffer and the CMB is processed as follows: If the CMB status bits indicate a "read" or a "log only" condition, then the CMB is returned to the free queue via the routine $FREEMSG. The "log only" condition is used when &NUMCONS=0. "Read" and "Write" have meaning only when &NUMCONS>0. If the status bits indicate a "write" condition, then the CMB is queued for display via the $WQUEBUF subroutine.

4.10 OPERATOR CONSOLE INPUT/OUTPUT PROCESSOR

This processor and associated routines are included in HASP only if the value of &NUMCONS is greater than 0 (see Section 7). The HASP interface to the OS console support which is included if &NUMCONS=0, is described in Appendix 12.15.

4.10.1 Operator Console Input/Output Processor - General Description

The function of the Operator Console Input/Output Processor is to process all I/O activity on all operator consoles. The processor also processes all console errors, making a number of retries. If the error continues, the message is ignored.

4.10.2 Operator Console Input/Output Processor - Program Logic

The Operator Console Input/Output Processor examines each entry in the console message buffer I/O queue, **$BUSYQUE**. Each bit in the console byte is tested for an available console. If one is found the appropriate operation is initiated with a $EXCP macro-instruction and testing of the queue is resumed. When all available consoles have been processed, the processor enters a $WAIT condition until an I/O interrupt is received on one of the consoles, or until another console message is added to the queue.

4.10.3 Operator Console Input/Output Appendage - Program Logic

The Operator Console Input/Output Processor's asynchronous exit is entered from the Asynchronous Post Processor following the completion of an I/O operation on a console device. The IOB completion code is tested for abnormal end, and if an error exists, an error routine is entered to retry the operation.

If the completion is normal the appropriate physical console bit is shut off and the console byte is tested to see if the operation is complete on all consoles. If any bits are still on the Operator Console Input/Output Processor is $POSTed and an exit is taken.

If all bits are now off and the operation code is a write, a link is made to $FREEMSG, the Input/Output Processor is $POSTed and an exit is taken.

If the completed operation is a read, the response is processed according to type. If the buffer contains a HASP command (i.e., an input message whose first character is a dollar sign ($)), it is chained to the end of a queue for the Command Processor ($COMMQUE), the processor is $POSTed, and an exit is made with a $POST of the Input/Output Processor.

If the message is a "reply", the reply number is converted to binary and the corresponding entry in $WTORQUE is located. Using the information in the entry, the message is moved to the WTOR's reply area and the WTOR's ECB is POSTed. The reply queue entry is merged into the free queue ($WTORFRE), and a link is made to the Log Queuing Routine. The Input/Output Processor is $POSTed and exit is made.

If the message is not a "reply" or a HASP command, it is assumed to be an OS command. The message buffer is set to the proper format for the Master Command Routine and an SVC 34 is issued. When control is returned from the Master Command Routine, the buffer is released, the Input/Output Processor is $POSTed and an exit is made.

4.11 TIMER PROCESSOR

4.11.1 TIMER PROCESSOR - GENERAL DESCRIPTION

The function of this processor is to reset the OS interval timer after a timer interrupt has occurred.

4.11.2 TIMER PROCESSOR - PROGRAM LOGIC

This processor calls the IPOSTIT and ISETINT subroutines in the $STIMER/$TTIMER Control Service Routine (see Section 5.6), which causes the expired TQEs to be POSTed and the time interval specified in the first TQE in the TQE chain to be set into the OS interval timer. The processor then waits for another timer interrupt to occur. When the next timer interrupt is processed, the asynchronous exit routine $POSTs this processor and the above procedure is repeated.

4.12 REMOTE TERMINAL PROCESSOR (360/20-STR)

4.12.1 Remote Terminal Processor (360/20) - General Description

The Remote Terminal Processor (RTP), although not a part of HASP proper, can be considered in the same catagory as other HASP processors. RTP is created by HASPGEN to operate as an extension of HASP on a System 360 Model 20 used as a remote terminal to HASP. RTP, in the Model 20, maintains constant communications with HASP at the central computer site via several classes of telephone lines to 1) encode and transmit jobs submitted at the remote site to HASP for execution on the central computer, and 2) print and/or punch the output from jobs thus submitted as the output becomes available. Various techniques are utilized by RTP and HASP to obtain maximum performance of both the Model 20 devices and the communication lines used. RTP currently requires an 8K Model 20 with any reader and printer attached. The program can be made to operate in a 4K environment at a somewhat degraded performance with reduced ease of operation.

RTP has been designed to allow the addition of "background" functions to operate in a multiprogrammed environment with normal remote terminal processing.

4.12.2 Remote Terminal Processor (STR Model 20) - Program Logic

Upon completion of the loading of the RTP program deck, control is transferred to the initialization phase of the program to prepare for job processing. Initialization first checks the card reader for the presence of patch (REP) cards and, if present, makes the appropriate patches (the RTP REP card format is identical to the HASP REP card as described in Section 6.4). Encountering a /*SIGNON card within the REP cards, will cause initialization to replace the default remote SIGN-ON identification and password by the contents of the card. After loading REPs, or if no REP cards are present, the dynamic configuration card (which follows REPs if present) is decoded and appropriate commands for the system punch selected are established. (The formats of the SIGN-ON and dynamic configuration card are given in the Model 20 Operator's Guide-Section 11.2). The final process of initialization is the dynamic construction of the buffer pool. Buffers are built, according to the HASPGEN parameter &TPBFSIZ until the memory size of the machine is reached or the assembly parameter &NUMBUFS is reached. Construction of the buffer pool overlays the complete initialization routine. Control is then passed to the processing section of RTP.

The processing phase of the program consists of four principal processors and a communications adapter (CA) I/O supervisor. Allocation of CPU time to the various processors is accomplished via a commutator. A processor is entered into contention for CPU time by changing its commutator entry from a NOP to a BRANCH command. Through the use of the WAIT macro, a processor may await the occurrence of a certain event and be entered, via the commutator, below the wait instruction upon completion of the event.

PROCESSORS

Card Read Processor

Upon initial entry, this processor checks the system card reader for ready status. If the device is not ready, HASP is notified, via a SEND EOT, of the lack of jobs to transmit, the CA receive processor is activated, the card read processor is deactivated, and entry is made to the commutator. If the card reader was ready, the transmission phase is immediately begun. Cards are read (double buffered) and are passed to the ENCODE subroutine which compresses and translates the card for transmission. The encoded card images are blocked in a buffer obtained from the dynamic buffer pool until the capacity of the buffer is reached. The buffer is then chained into a queue of buffers awaiting transmission by the CA transmission processor to HASP in the central computer. Another buffer, if available, is obtained from the buffer pool and is processed in a like manner. When, and if, the supply of buffers is exhausted, the reader processor enters a WAIT state to await the freeing of a transmitted buffer by the CA transmission processor. When the last card of the job stack has been read, a SEND EOT (zero word count buffer) is queued for transmission and the steps described previously are done to terminate transmission and activate reception. In order to

minimize CPU utilization, the card read processor-compression routine only compresses "n" or more blank characters (where "n" is the value of the assembly parameter &CCT). The format of transmission records to HASP is described in Section 12.9.3.

Communications Adapter Transmission Processor

The CA Transmission processor removes buffers from an ordered queue, dynamically being built by the Card Read Processor, and transmits their contents to HASP in the central computer. All transmissions are via the Communications Adapter-I/O Supervisor (CAIOS) which provides for line re-instruct at interrupt time to make optimum use of the line (See CAIOS description). As posting of successfully completed writes occurs, the buffers are returned to the free buffer chain for reuse by another processor. This processor continues to dequeue and transmit buffers, as they become available, until a buffer with a transmission word count of zero is encountered. An EOT is then sent to HASP to indicate the end of the input stream, the CA Transmission Processor is deactivated and return is made to the commutator.

Communications Adapter-Receive Processor

The CA Receive Processor is activated by the Card Read Processor when it is determined that no jobs are available to transmit to the central computer. Upon being entered, CA Receive establishes communication with HASP in the central computer to await the output of a previously submitted job. The lack of jobs to transmit is indicated by HASP with an immediate EOT signal to the Model 20. When this EOT is received, the CA Receive Processor deactivates itself and activates the Card Read Processor to again check for the presence of jobs to send to HASP.

If a job is avialable to be printed or punched, the CA Receive Processor activates the Print/Punch processor and immediately begins reading transmittal records into buffers obtained from the dynamic buffer pool. Buffers, thus filled, are placed in an ordered queue to await processing by the Print/Punch Processor. All CA reads are via the Communications Adapter I/O Supervisor (CAIOS) which provides for line re-instruct at interrupt time to make optimum use of the line (see CAIOS description). Processing continues, as buffers and/or transmittal records become available, until an EOT signal is received from HASP indicating end-of-job. A buffer with a word count of zero is added to the queue to inform the Print/Punch processor of the end-of-job.

Communication is then, once again, established with HASP to ascertain if additional output for this job is available (i.e. the punch output of the job which has just completed printing). After the additional output has been processed, or if none existed, the CA Receive Processor is deactivated, the Card Read Processor is activated, and return is made to the commutator. Note that this logic, of activating the Card Read Processor prior to beginning processing output from the next job, allows the Model 20 Operator to interrupt print/punch processing, at a job boundary, to transmit a job to the central computer.

Print/Punch Processor

When activated, the Print/Punch Processor begins dequeuing and processing buffers from the queue (being) created by the CA Receive Processor. Records to be punched are indicated by "carriage control" characters of X'0F0F' and are routed to the punch section of the processor. In order to minimize CPU requirements, the print processor does not provide for 1-7/8 encoding of print characters (see Section 12.9). The 16 4 of 8 characters normally reserved for 1-7/8 encoding are redefined for print records only, as additional print characters, thus yielding a 64 character print set.

After reconstructing and printing or punching all records in a buffer, that buffer is returned to the buffer pool for use by another processor. When a buffer with a zero word count is encountered in the queue (indicating end-of-job), the Print/Punch Processor is deactivated, unless records from the next job have already been queued, and return is made to the commutator.

If a dynamic configuration card described the system punch unit as DUMMY, the punch section of the processor is dynamically altered (by initialization) to immediately free all punch buffer encountered in the Print/Punch buffer queue. This results in eliminating punched output; however, punch records are still transmitted to the Model 20.

By setting the assembly parameter &PUNCH to 0, all code concerned with processing punched output will be eliminated from RTP. The appropriate HASPGEN must be done on the central system to force all punch output for a "punchless" terminal to be processed locally.

Communications Adapter I/O Supervisor

The primary purpose of CAIOS is to assure the maximum possible communication line utilization by re-instructing the line at the earliest possible moment after completion of a previous generation.

All requests to read and/or write the communication line are passed to CAIOS for execution by the CA processors. Upon receipt of an I/O request, CAIOS immediately initiates the operation if the line is dormant, or queues the request to await completion of the currently active operation.

The completion of a CA I/O operation causes an interrupt which immediately transfers control to CAIOS. If the operation indicated as complete by the interrupt was successful (error free), any queued I/O request is immediately initiated. The Event Control Block of the requestor of the just completed I/O operation is posted (with a X'7F') to indicate the successful completion of the request. Return is then made to the interrupted processor. CAIOS recognizes and attempts to correct all transmission errors encountered on any CA I/O operation. Since both CA processors are designed to double buffer I/O requests, CAIOS insures virtually total line utilization during transmission periods.

4.12.3 Remote Terminal Processor (360/20)-Assembly Parameters

The following indicates the variable name and function of certain RTP assembly parameters which can be of general use.

&TPBFSIZ - defines the size of the buffers used for transmission to and from the HASP system. (Since this variable must exactly agree with the corresponding variables in the HASP system, the values of both are automatically set at HASPGEN time.

&NUMBUFS - limits the number of CA buffers created dynamically at initialization time. Initialization will create buffers until the capacity of memory, or the value of &NUMBUFS is reached. It is suggested that this value be made large enough to allow sufficient buffering (hence line load-leveling) to occur.

&CCT - represents the minimum number of consecutive blank characters which will be compressed by the Card Read Processor. This value should never be less than 4 and, significantly

reduces CPU requirements by the Card Read Processor as it is increased. A value of 80 will effectively prevent all blank compression (except on totally blank cards). The value of &CCT may never exceed 80.

&PUNCH - controls the existance of code within RTP to process punch output received from HASP.

If &PUNCH=1, punching capabilities will exist in RTP.

&PUNCH=0, no punching capabilities will be created in RTP (NOTE: the HASPGEN of the central computer system must agree with this option.)

&MACHINE - defines the Model of SYSTEM/360 on which RTP is to operate. This value must presently be set to 20. This option can subsequently be used to assemble RTP, at HASPGEN time, for any Model of SYSTEM/360 being utilized as a HASP remote terminal. Although certain parts of this feature are currently in RTP, it is incomplete and totally untested.

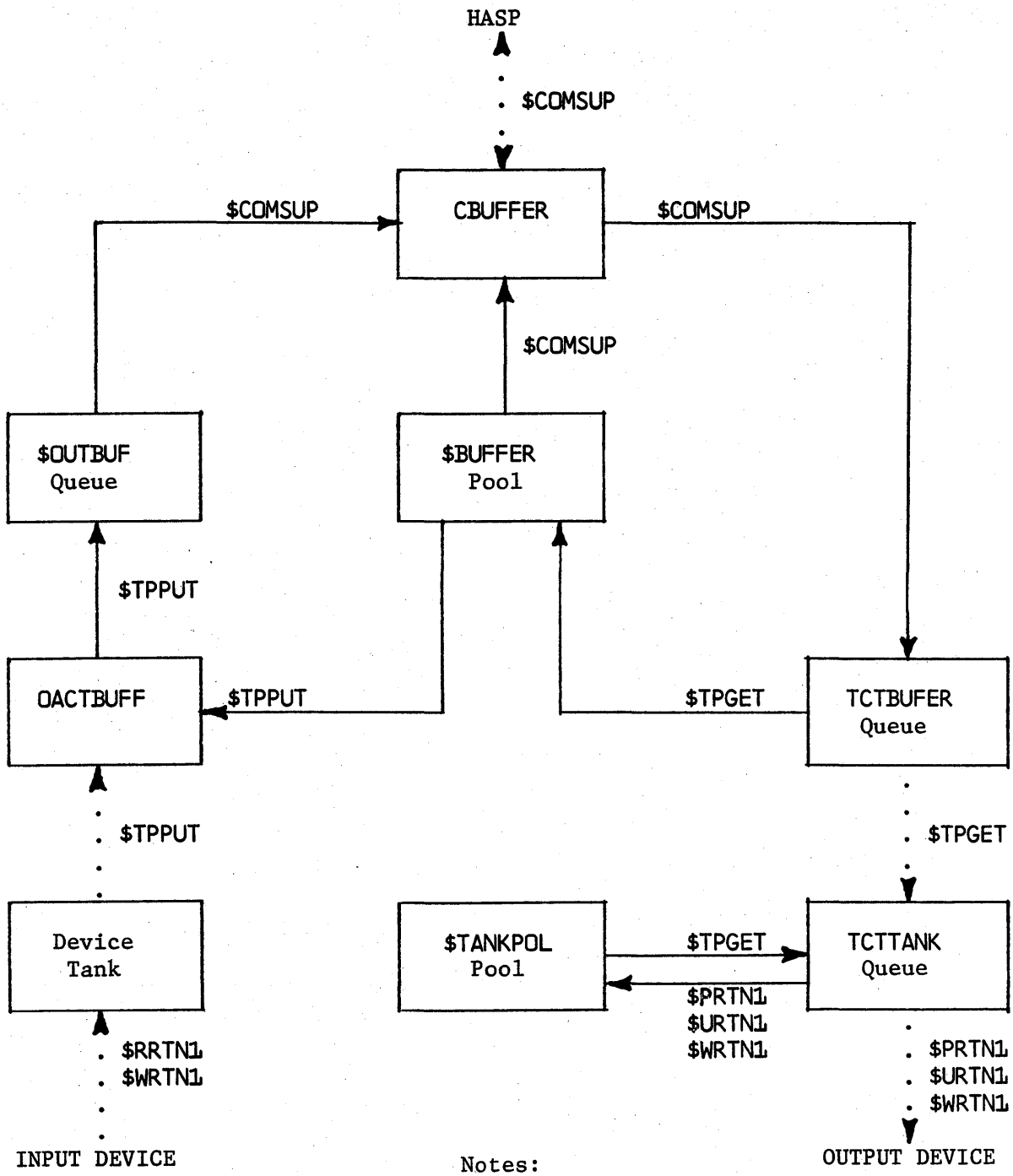
4.13 REMOTE TERMINAL PROCESSOR (SYSTEM/360-BSC)

The following sections outline the basic logic flow of the MULTI-LEAVING Remote Terminal Processor program for System/360 (including Model 20) workstations utilizing Binary Synchronous communications devices. The same workstation program is utilized for both the Model 20 and System/360 workstations with generation parameters for the machine type.

4.13.1 General Description

The MULTI-LEAVING Remote Terminal Processor program is created by HASPGEN to operate as an extension of HASP on any Model of SYSTEM/360 used as a remote workstation for HASP. This terminal program maintains constant communications with HASP at the central site via several classes of telephone lines to (1) encode and transmit jobs submitted at the remote site for OS/360 processing on the central computer, and (2) print and/or punch the output from jobs thus submitted as the output becomes available. Optionally, if an operator console is attached to the remote system, informational and control facilities are provided. All of the above functions may occur simultaneously. Various techniques are utilized by HASP and the workstation program to obtain maximum performance of the remote devices and the communications line. Figure 4.13.1 indicates the basic information flow through the system.

Figure 4.13.1 MULTI-LEAVING Information Flow Diagram



Notes:

Solid lines indicate buffer or decompression tank flow with or without data.

Broken lines indicate data flow only.

Line comments indicate processor responsible.

4.13.2 Program Logic

The MULTI-LEAVING Remote Terminal Processor consists of an initialization section, four principal processors, three communications interface processors and a communications INPUT/OUTPUT supervisor. Allocation of CPU time to the various processors is accomplished through a basic program commutator. A processor is entered into contention for CPU time by changing its commutator entry from a NOP to a BRANCH command. A single control block, the Total Control Table (TCT) is utilized by all processors to provide for synchronization of concurrent operations, processor status information, re-enterability and both inter and intra processor communication.

The following sections discuss the basic logic flow of the various components of the program.

Communications Interface Processor - Output ($TPPUT)

This processor serves as the interface between the various input processors and the communications INPUT/OUTPUT supervisor. Its function is to compress and encode records for subsequent transmission to HASP at the central site. $TPPUT is utilized as a subroutine by the various input processors and relieves the input routines of the responsibility of data compression and transmission buffer management. As records are submitted for transmission, $TPPUT compresses the records according to a compression type generation parameter

(&CMPTYPE) and add the encoded record to its current output buffer. When the current buffer is filled or terminated, it is chained in an ordered queue for transmission to HASP by the communications INPUT/OUTPUT supervisor and a new buffer obtained. Details of the compression and encoding technique utilized by $TPPUT are included as an appendix to this manual.

Communications Interface Processor - Input ($TPGET)

This processor serves as the interface between the various output processors (Print, Punch, Console, etc.) and the Communications INPUT/OUTPUT processor. Its function is to decode and uncompress transmission buffers received from HASP and to queue the decompressed records to the appropriate processor for processing. $TPGET is entered from the commutator and processes buffers from a ordered queue of received buffers established by the Communications INPUT/OUTPUT supervisor. Records received are deblocked into "decompression tanks" and passed to the appropriate processor. Synchronization and passage of the tanks to the processors is accomplished through the Total Control Table (TCT) for each processor. $TPGET additionally is responsible for metering the flow of each type of record from HASP. This also is accomplished by utilizing the various buffer and tank limits indicated in the TCT for each processor.

Control Record Processor ($CONTROL)

This processor provides synchronization between the various processing

functions at the workstation and the HASP SYSTEM at the central site. Control Records from HASP (i.e. Request to start a function, etc) are queued on this processor by the $TPGET processor. $CONTROL then processes the control record, transmits a response, if required,through $TPPUT and initializes the required functional processor.

Communications INPUT/OUTPUT Supervisor (COMSUP)

COMSUP maintains communications with HASP in the central CPU at all times and is responsible for the transmission of all data to and from the remote site. The data processed by COMSUP is always in compressed buffer form and passes to and from COMSUP via ordered queues established by $TPPUT and for $TPGET.

The communications I/O is primarily interrupt driven and is completely maintained by COMSUP (i.e. COMSUP is both the initiator and executor of communications I/O). During periods requiring no data transmission, COMSUP maintains a "handshaking" cycle with HASP at approximately 2 second intervals to insure full bi-directional capabilities and to avoid unprogrammed "time-outs" of the adapter.

In addition COMSUP maintains, verifies and corrects (if necessary) the MULTI-LEAVING block sequence checking feature and detects, logs and retries all communications errors.

Initialization Processor

The Initialization Processor receives control from the loader and initializes the remote terminal program as follows:

1. If the CPU is not Model 20, general registers 1, 2, and 3 are loaded to establish 16 K addressability.
2. Replacement (REP) cards are read from READER 1 for possible modifications to the program. The format of the REP card is as follows:

   Col. 2-4 REP

   Col. 9-12 Replacement address - hexadecimal address of the first half word of storage to replace (if blank the previous REP card is continued)

   Col. 17-n xxxx,xxxx,...xxxx replacement data - one or more half word groups of hexadecimal data separated by commas

   Col. n+1 blank - terminator for the replacement data

   Col. n+2-80 comments - any text

Each REP card is printed on PRINTER 1 when read as a record of program modification. REP reading is terminated when either a blank card (blank in Col. 1-5) or a /*SIGNON card is encountered.

3. The HASP ENVIRONMENT RECORDING ERROR PRINTOUT (HEREP) is printed if the recording table is intact from the last execution

of the program; otherwise, a new table is created for future recording and print out.

4. Interrupt PSW's are set for non Model 20 CPU's.
5. The communication adapter is enabled and communications established with HASP as follows:
   a. Write SOH-ENQ to HASP
   b. Read for DLE-ACK0 from HASP

   If I/O errors occur or HASP responses do not match the expected sequence, the sequence is repeated.
6. The processor constructs a buffer pool over itself and queues the SIGN-ON record for transmission to HASP.
7. I/O PSW's are set (I/O old points to commutator) and control is passed to the communication adapter interrupt routine.

Print Service Processor - $PRTN1

The Print Service Processor's major functions are dequeuing decompression tanks containing print information from the printer Total Control Table, examining the sub-record control byte for carriage control information, performing required carriage control, printing the information on the designated printer, and releasing the used decompression tank to the pool. The processor also provides event control upon dequeuing and releasing the "tanks". If no console typewriter is attached to the system and the value of the user option &PRTCONS is not zero, the processor will set status information

at the end of each print data set which allows the console processor to queue operator messages for printing.

Input Service Processor - $RRTN1

The Input Service Processor supports various card readers used for the purpose of submitting job streams to HASP and in the case of Model 20 DUAL 2560 MFCM serves the functions of punch service processor. The processor provides error analysis and recovery for supported devices. Execution begins with the initial read routine which continuously attempts to read cards from the designated card reader. In the case of a DUAL 2560 control is passed to the punch routine if the primary feed is empty. If reader is a DUAL 2520 or 1442 the routine will check the first card for blank and if so pass control to the punch preparation routine; otherwise subroutine $TPOPEN is called which sends a request to send a job stream to HASP. When permission is received the job stream submission routine is entered which reads cards into one of two decompression tanks calling the $TPPUT processor which compresses the data and schedules transmission to HASP. At end-of-file $TPPUT is used to signal HASP and control is passed to the initial read routine.

The DUAL 2560 punch routine attempts to dequeue a decompression tank from the Total Control Table. If successful the card image is punched and the used "tank" is released to the pool. The routine continues to dequeue and punch for a maximum of 100 cards; this time tests are made to determine

the existance of cards in the primary feed. The tests are also made in the event of no tanks available for dequeuing. If the tests are negative the processor continues to punch cards; otherwise control is passed to the read routine following the initial read. The processor provides event control upon dequeuing and releasing decompression tanks.

DUAL 2520/1442 punch preparation routine tests for:

1. Operator signal - changing of the data dials, .SR1 command, or unsolicited device end. (Depends upon configuration).
2. Presence of Decompression tanks for punching.

If the operator signals, the routine passes control to the initial read routine. If a "tank" is queued to the device Total Control Table control is passed to the Punch Service Processor ($URTN1).

Punch Service Processor - $URTN1

The Punch Service Processor's major functions are dequeuing decompression tanks containing print information from the punch Total Control Table, punching the information into cards on the designated punch, and releasing the used "tanks" to the pool. The processor also provide event control upon dequeuing and releasing the "tanks" in addition to error recovery upon erroneous punching of data. If the device is a DUAL 2520 or 1442 control is passed to the Input Service Processor ($RRTN1) after servicing output "tank".

Console Service Processor - $WRTN1

If the remote terminal has an attached operator printer keyboard, the console processor performs the following functions:

1. Reads operator commands from the console keyboard.
2. Examines the input for local commands (Model 20 only) passing local commands to the command processor and passing all other commands to HASP.
3. Type operator messages contained in decompression tanks queued to the console Total Control Table.
4. Convert codes in the error message log table to readable form and type the resulting messages.

Execution begins with the processor testing for an operator command in the console input "tank" waiting to be transmitted to HASP. If so the console read in function is skipped and an attempt is made to send the command to HASP. Control is passed to the console output routine which tests for output messages. If so, the processor dequeues the tank, types the message, and releases the tank. Control is then passed to the beginning of the processor. If no output messages are pending the console logging routine is entered which converts, types the message, and passes control to the beginning of the processor. The console read routine tests for operator requests and if so, reads the command from the keyboard, calls the $TPPUT processor to compress the data and transmit the command to HASP, and passes control to the console output routine. If the remote

terminal is a Model 20 the read routine tests for local commands and calls the command processor which in case of ".S" command , posts the appropriate Service Processor and returns. Local commands are not transmitted to HASP.

The Console Service Processor without a console keyboard exists only when the value of the user option &PRTCONS is not zero. Execution begins with a test for printer availability. If available, any console messages are removed from the console output queue by the dequeue routine and attached to the printer queue, allowing the Print Service Processor to print the message. If no console messages are queued the processor will convert any log messages into readable form, move the resulting message into a "tank" obtained from the pool, queue it to the console output queue and pass control to the console dequeue routine. If the value of &PRTCONS is one and the printer is not available console messages are allowed to accumulate to a maximum queue limit. If the limit is reached prior to the printer becoming otherwise available the printer is forced available and the messages are queued to the printer with the sub-record control byte of the first message set to skip to channel 1 before print. If the value of &PRTCONS is two and the printer is not available to the console the processor will dequeue console tanks and release them to the pool.

Total Control Table (TCT)

The Total Control Table is the major working storage area for the unit record processors and is customized for each configuration and device supported by the remote terminal program. Each basic TCT field may be referred to by using symbols defined in the DSECT named TCTDSECT, however, each processor has the option of uniquely referring to the fields directly by using the alternate three character prefix to each field name as follows:

TCT = General TCT prefix

CCT = Control record TCT

PCT = Printer TCT

RCT = Reader TCT

UCT = Punch TCT

WCT = Console TCT

Appropriate DSECT's are provided by generation macros in the event more than one TCT of a given type is supported by the system. Basic control fields appearing only in systems with model numbers above the Model 20 are as follows:

NAME	DESCRIPTION
$pCTCOMn	TCT addressability field - The commutator branches to this field to give control to the appropriate processor - the field contains a BALR R7,0 instruction which sets up TCT

NAME	DESCRIPTION
	addressability for the processor - symbol characters "p" and "n" uniquely identify the TCT for the commutator
TCTSTRT	First two characters of unconditional branch instruction
TCTENTY	"S" type address constant pointing to the appropriate processor - the field completes the branch instruction which passes control to the processor at the desired entry point
TCTRTN	Return to next entry in commutator - each processor waits by branching to this field of the TCT which in turn branches to the commutator
TCTCCW	Actual CCW op-code used in last I/O on the device - set by the processor and unit record IOS
TCTDATA	Address of data area used for last I/O transfer or address of input "tank" currently being

NAME	DESCRIPTION
	compressed for transmission to HASP
TCTFLAG	CCW flags
TCTOPCOD	Op-code which will be inserted into the TCTCCW field upon normal entry to unit record IOS
TCTCCWCT	CCW count field - length of data last transferred or to be transferred
TCTSENSE	Sense information - set by unit record IOS for error diagnostic purposes
TCTUCB	Device Address - contains hexadecimal device address for SIO and interrupt recognition purposes - the high order bit of the field is set on by the processor when waiting for HASP to authorize job submission
TCTECB	Event Control Block - contains all bits stored in CSW byte 4 since the last SIO instruction for the device - busy bit is set at SIO and when the processor desires to wait for unsolicited

NAME	DESCRIPTION
	device end - busy bit is reset at device end
TCTALTOP	Alternate op-code for DUAL reader/punch devices - processors requiring alternate op-codes have the option of setting the TCTCCW field with the contents of this field prior to entry to unit record IOS
TCTSAV1	Save area for the processor subroutine LINK register

Basic fields which may appear in remote terminal programs for all 360 models are as follows:

TCTNEXT	Next TCT in the chain of TCTs
TCTFCS	Function Control Sequence Mask - used by $TPGET processor to setup the FCS transmitted to HASP for backlog control
TCTRCB	Record Control Byte - records from HASP which have RCB bytes identical to this field will be queued for output on the corresponding device

NAME	DESCRIPTION
TCTSTAT	Status Flags - each bit has one or more meanings which are dependant upon the processor involved:

bit 0 = TCTOPEN - always off indicating device is in use by HASP output (as appropriate)

bit 1 = TCTACT - used by $TPGET to determine which output devices need more data - processors set bit 1 when dequeuing output "tanks"

bit 2 = TCTSTOP - device has been stopped and is awaiting a start command.

bit 3 = TCT1052, TCT2152 - console device identifier

bit 4 - PCT only = TCT1403, TCT1443, TCT2203, TCTPRTSW - indicates the status of the corresponding printer - if set the printer is available for printing operator messages

bit 4 - WCT only = TCTREQ - console request - operator desires to enter a command

NAME	DESCRIPTION
	bit 4 - UCT only = TCT1442 - the device is a 1442 with single stacker pocket
	bit 5 - RCT or UCT = TCT2540 - TCT is for a 2540
	bit 5 - WCT only = TCTREL - release requested - an unsuccessful attempt has been made to obtain a buffer for command transmission to HASP - the command is in compressed form in the consoles "tank" waiting for a free buffer
	bit 6 - RCT/UCT = TCT14420, TCT25600 - TCT is for a DUAL 1442 Reader Punch or DUAL 2560 MFCM
	bit 7 - RCT/UCT = TCT25200 - TCT is for a DUAL 2520 Reader Punch device
TCTCOM	Pointer to corresponding commutator entry
TCTID	Optional field - two character identification for local command processors
TCTINRCB	Optional field - exists when DUAL devices are attached to the system - identifies the Input

NAME	DESCRIPTION
	Service Processor function as opposed to the Punch Service Processor function identified by TCTRCB - TCTINRCB is equated to TCTRCB if no DUAL devices are attached

The following fields are normal device extensions and do not exist for card reader devices when DUAL devices are not attached to the remote terminal:

TCTTANK	Beginning of output "tank" queue - output records appear in unit record image form
TCTBUFER	Beginning of output buffer queue - contains records in compressed form waiting for de-compression into tanks
TCTTNKLM	Tank limit - maximum number of "tanks" which may be placed in the "TCTTANK queue
TCTTNKCT	Tank count - actual number of "tanks" queued to the TCT
TCTBUFLM	Buffer limit - maximum number of output buffers which may be placed in the TCTBUFER queue

NAME	DESCRIPTION
	before signalling HASP to suspend sending the streams - limit is ignored for WCT
TCTBUFCT	Buffer count - actual number of buffers queued to the TCT

Reader and console TCT's have extensions which are used as "tanks" for records which are transmitted to HASP. These "tanks" belong to the device (2 for readers and 1 for the console) and are not released to the "tank" pool. The following field symbols are only defined for the TCT's with prefix designators. RCT, WCT, and with DUAL devices UCT:

RCTTANK1, RCTTANK2	"Tank" origin and working storage
RCTTRCB1, RCTTRCB2	Input RCB for HASP identification
RCTTSRC1, RCTTSRC2	Sub-record control byte = X'80'
RCTTCT1, RCTTCT2	Count field - length of data portion
RCTTDTA1, RCTTDTA2	Data area - input card or operator command - will be blank for the DUAL 2520 and 1442 while in output status

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

4.14 REMOTE TERMINAL PROGRAMS (1130)

Introduction

The 1130 MULTI-LEAVING terminal program is designed to operate on a system with 8K words which contains the standard Binary Synchronous Communications Adapter.

The unit-record equipment supported may include any or all of the following devices:

- 1442 Reader/Punch or Punch
- 2501 Reader
- 1132 Printer
- 1403 Printer
- Console keyboard/Printer

Programs developed for the 1130 in conjunction with the HASP Remote Job Entry feature are assembled using the OS/360 Assembler. The 1130 instruction set is generated thru the use of macro instructions (See Section 14.4.5) corresponding to the actual 1130 hardware commands. Additionally, pseudo (assembler) operations are available to aid in the development of 1130 programs on the System 360.

The object decks produced by the OS Assembler are subjected to further processing by a program (LETRRIP) which condenses and changes the format of the EBCDIC decks to facilitate 1130 loading.

The remote terminal system for the 1130 is composed of several programs briefly described in the following paragraphs:

RTPBOOT - A bootstrap loader consisting of a single "load mode" format card and several column binary and EBCDIC program cards. The function of RTPBOOT is to "bootstrap" an EBCDIC format loader (RTPLOAD) into 1130 core. RTPBOOT will load from either a 1442 or a 2501 card reader.

RTPLOAD - Loads into the upper segment of defined 1130 core and then loads the main terminal program (RTP1130) into the lower extent of 1130 core. RTPLOAD also processes REP cards and performs the initial processing of /*SIGNON control cards.

RTP1130 - The main terminal processing program which provides the MULTI-LEAVING support for the 1130.

The following sections provide more detailed information on the design and implementation of the above programs.

4.14.1 Remote Terminal Processor (RTP1130)

Introduction

The subsequent sections present the basic structure of the terminal program for the 1130. Included, are descriptions of the commutator logic and associated processors; system subroutines; processor subroutines; control block formats and data block general formats.

The documentation presented is intended to be introductory in nature. The user intending to modify the system should use the documentation in conjunction with a program listing which contains commentary in much greater detail.

Commutator Processors

Distribution of CPU time to the processors concerned with the functions necessary to support terminal devices is through programmed commutator logic. Each processor which needs CPU time and is dependent on external I/O device rates is represented by a commutator entry. The commutator entry consists of the following basic elements:

- A named commutator "gate" which takes the form of a branch to the next commutator entry (gate closed) or a "NOP" if the entry is active (gate open).
- A long form branch to the active commutator main routine used if the gate is open.
- A named return point for reference by the main commutator routine.
- A named end to the commutator entry which is the address of the next commutator entry.

The basic structure as defined may also contain register save-restore sequences to be used for each entry-exit cycle through the commutator.

The processors entry from the commutator (gate open) usually provides for a method of setting a variable entry to the segments of the processor which are involved with waiting for I/O to complete or some system resource to become available.

The general operation of the commutator involves the opening and closing of processor gates, the setting of variable entry points within the processors, the initiation and associated wait period for I/O operations and the return to the commutator to "share" the CPU during wait periods. The last instruction in the commutator is a branch to the "top" or first instruction in the commutator which initiates the next cycle. The current system does not provide for a priority relationship among commutator processors.

The main commutator processors contained in the RTP1130 system and briefly described in the following sections.

TPIOX - SCA Input/Output Control Processor

Controls the transmission of data and/or control records between HASP and RTP1130 via the SCA. All adapter I/O is initiated using the SCA I/O Supervisor - BSXIOS.

TPGET - Processor for TP Buffers From HASP

Processes data received from HASP in the form of TP buffers or control records preprocessed by TPIOX. Control record processing is in the form of "Request to start" or "Permission to send" functions.

Data buffers are deblocked, decompressed, converted to appropriate codes (1403 printer, 1442 punch, etc.) and queued for the specified commutator I/O processors.

Control information pertinent to the unique requirements of each data type is provided through the associated UFCB.

TPPUT - Processor For Data Destined For HASP

Acquires a TP buffer from the free chain and collects data from defined sources (card reader(s), console keyboard, etc.) to be processed (converted, truncated, compressed, etc.) and inserted into the buffer which is queued for TPIOX transmission to HASP.

RDTFO - 2501 Card Reader Processor

A conditionally assembled processor which supports the 2501 card reader as a job entry device. The functions of monitoring for a 2501 "ready" condition; reading cards; requesting permission to transmit to HASP; waiting for permission to send; queueing data for TPPUT; transmitting "end-of-file" conditions and device error recovery are contained in this processor.

RPFFT - 1442 Reader And/Or Punch Processor

A conditionally assembled processor which supports the 1442 - 5, 6 or 7 as a card reader, card reader/punch or as a card punch only. The functions to be performed are controlled by the assembly variables chosen and the use of local operator commands, when applicable. The reader sections of code monitor for a "ready" condition; reads cards for transmission to HASP via TPPUT; processes "end-of-file" communications and provide error recovery. The punch sections of code wait for data to be punched through interrogation of a queue developed by the TPGET processor and provide error recovery and and punch termination procedures.

PRFOT - 1403 Printer Processor

A conditionally assembled processor which supports the 1403 printer as a terminal output device. The functions of monitoring for input to be printed; simulating carriage control operations; processing "end-of-file"

conditions; setting UFCB status information and error recovery are included in this processor.

PRETT - 1132 Printer Processor

A conditionally assembled processor which supports the 1132 printer as a terminal output device. The functions of monitoring for input to be printed; initialization of interrupt processing routines for the 1132 print scan operations; simulation of carriage control operations; processing "end-of-file" conditions; setting UFCB status information and error recovery are contained in this processor.

CONSL - Console Keyboard/Printer Processor

Processes console keyboard input and prints on the typewriter messages originating from HASP or internal sources.

Keyboard input is initiated by activation of the "INT REQ" key and by the interrupt routine which sets a flag and opens the console routine gate. Note: The position of the "keyboard/console" switch is not interrogated and input is assumed to be from the keyboard. The value of the console entry keys is read every communtator cycle and, if key o is on, stored in location $ENTKEYS. All non-control character input is printed and the card code value stored for investigation at EOF time. If the first character of input is "." (period) then the data is assumed to be a local command. All other data is transmitted to HASP for action as a HASP operator command.

Print input is obtained from a queue which originates locally and/or from HASP. Data to be printed may be EBCDIC or tilt-rotate code and black or red ribbon.

RTPET - Initialization Processor

This special commutator processor is responsible for the initialization functions necessary for the commencement of the 1130 terminal operation in conjunction with HASP. The major functions performed are:

- Sets the interrupt transfer vectors for RTP1130 operation.
- Dynamically builds the TP buffer pool using the defined extent of 1130 core; the end of the 1130 program and the defined TP buffer size.
- Builds a TP buffer containing the sign-on information processed by RTPLOAD for transmission to HASP.
- Establishes SCA communications with HASP and prepares TPIOX for "sign-on".
- Opens the commutator gates for all SCA and input processors.
- Disconnects initialization from the commutator.
- Branches to commutator which initiates MULTI-LEAVING operation.

System Subroutines

The following are brief descriptions of the major subroutines contained in the RTP1130 program. These subroutines are available for use by any system commutator processor with the restriction that they may not be used at interrupt time. Detailed information concerning the calling sequences, input values, etc. may be found in the listing of the RTP1130 program.

SGETQEL - Dequeue An Element From a Chained List

Given the address of a chained list, SGETQEL returns the address of the first element available in the list and removes the element and rechains the list. The chain field of the dequeued element is set to zero before returning. If the chain is null, an indication is returned to the user.

SPUTFQL - Enqueue An Element In A Free Element Chain

Given the address of a free element chain pointer and the address of an element to be returned to the free chain, the element is returned to the free chain. The construction of the free chain is in random order depending on system processor utilization of the free element chain.

SPUTAQL - Enqueue An Element In An Active Chained List

The address of an element supplied by the caller is used to build a chained list in first-in, first-out order.

STPOPEN - Initiate Control Record Transmission

Control record communications with HASP in the form of "Request to start" and "Permission to send" sequences is the function of this routine. Input includes an indication of the control record type and a pointer to the UFCB for the device being processed.

SSRCHB - Search UFCB Chain For Matching RCB

The RCB code supplied by the user is used to search the UFCB chain for a UFCB with a matching RCB code. An indication of the status of the search is returned to the caller.

SWTOPR - Type Message On Console Typewriter

The caller supplies the address of a message in EBCDIC and with control information indicating red or black ribbon and the number of characters to be typed. The address of a routine to be given control in the event that the message cannot be processed immediately must also be supplied.

The message is queued for processing by the console typewriter commutator routine.

SLOGSCA - Log SCA Error Messages On Console Typewriter

Error conditions associated with the SCA operation are logged on the console typewriter for information and possible remedial purposes. The format of the message logged is:

SCA LOG XXXXXXXX

Where the value of "XXXXXXXX" is determined by the caller and is in fact the contents of the ACC and EXT on entry to the routine.

An indication of the status of the request to log is returned to the caller.

SMOVE - Move A Variable Number Of Words

This routine provides for the moving of a specified number of words from a source block to a target block.

SXPRESS - Convert Card Code To EBCDIC

The card code (12 bit) input is converted to EBCDIC using a high speed conversion algorithm in conjunction with a minimal conversion table. Special consideration is given to "blank" conversion under the assumption that most cards are dense with "blank" data.

SXCPRNT - EBCDIC To Console Printer Code Conversion

Converts a single EBCDIC character to the equivalent console printer Tilt-Rotate code using a table look-up method.

SXPPRNT - EBCDIC To 1403 Printer Code Conversion

Converts a single EBCDIC character to the equivalent 1403 printer 6 bit with parity code using a table look-up method.

SXCPNCH - EBCDIC To Card Code Conversion

Converts a single EBCDIC character to the equivalent 12 bit card code using a table look-up method and conversion algorithm.

STRACE - Trace Machine Registers

Stores the information shown below in a table of variable length. Each entry is the result of the execution of the linkage created by the STRACE macro. The trace table created at assembly time is circular.

Trace table entry :

Word	Description
1	Count of the number of entries for this $TRACE
2	Location +1 of caller to $TRACE
3	Contents of ACC
4	Contents of EXT
5	Contents of XR1
6	Contents of XR2
7	Contents of XR3

The count of the number of entries is also stored in the STRACE macro linkage.

The assembly of STRACE is a function of the variable &TRACE.

SSDUMP - System Core Dump

A conditionally assembled subroutine which allows post-mortem or dynamic dumps on either the 1132 or 1403 printer. SSDUMP is assembled if &DEBUG SETA 1 is included in the RTP1130 source deck. Linkage to SSDUMP

via location 0 is also established so that a post-mortem dump may be taken by pressing system reset and start.

The linkage to use this subroutine dynamically is contained in the system listing. Note: The logic of the subroutine does not allow concurrent operation of the selected printer and other devices.

Processor Subroutines

The following are brief descriptions of the major subroutines which may be used by commutator processors subject to the restrictions that these routines are processor dependent in their operation. For example, the SCA I/O Supervisor (BSXIOS) is used at initialization time and by the TP buffer manager but cannot be simultaneously used by these commutator processors.

BSXIOS - Low Speed BSCA Input/Output Supervisor

Processes requests for transmit, receive or program timer functions on the low speed binary synchronous communications adapter. BSXIOS initiates the requested function and prepares the interrupt programs for the associated interrupt processing of the desired functions.

The status of the function performed by BSXIOS is contained in a communication cell which is addressed by a variable pointer word. A communication cell is defined for both read (receive) and write (transmit) operations. Various completion codes stored in the cells provide the status of the function with respect to normal or abnormal termination.

BSXIOS expects the caller to provide the address of an appendage routine to be entered at the termination (interrupt time) of every write operation. The purpose of the write end-of-operation appendage is to allow re-instruct (read operation) of the communications adapter as soon as possible after the write completion.

DBLOCK - Deblock, Decompress, Convert and Store Data From HASP

Locates a record (defined by RCB) in a TP buffer as specified by a given UFCB, decompresses, edits and moves data to a selected target area. The target area must have the same format as described under "Output Element (Tank) Description".

The operation of DBLOCK includes the priming of the output tank with an initialization value supplied by the user (usually the value of a blank for the associated device); the updating of control information in the UFCB; the setting of control information in appropriate fields of the output tank; the automatic entry to conversion and store routines unique to the device associated with the UFCB supplied and the communication of the status of the buffer being processed (end-of-file, end-of-block conditions).

TPCOMPR - Construct Records For Insertion In TP Buffers

Constructs a logical record consisting of a physical input record attached 1130 devices (card reader(s), console, etc.). The logical record constructed consists of the original input after code translation, data truncation and/or compression (optionally) and attachment of the control bytes necessary for HASP processing. The control bytes are per the standard HASP MULTI-LEAVING conventions.

The options listed below are set at assembly time to generate the supporting code.

- No compression or truncation
- Trailing blank elimination only (truncation)
- Blank and duplicate compression and blank truncation

The current version of TPCOMPR assumes card code input.

DBUGSCAL - Trace Routine For Low Speed SCA

This routine is conditionally assembled as a function of "&DEBUG" and provides a trace of all SCA interrupts in the form shown below. Entry is from BSXIOS interrupt processing routines. External disabling of the SCA trace function is provided through the entry keys. The trace table limits are preset to use the upper 8K of a 16K 1130 and must be changed either by assembly or by the appropriate "REP". See the program listing and refer to locations DBUGSTRT and DBUGSTND.

The trace table format is:

Word	Description
1	Operation type (BSXIOPT)
2	DSW at interrupt time
3	BSXIOS Completion Code (BSXOPF)
4	Location of interrupt
5	Data received/transmitted
6	Data transfer count
7	Read or write sequence index
8	Spare word

TPBUILD - Constructs TP Buffers

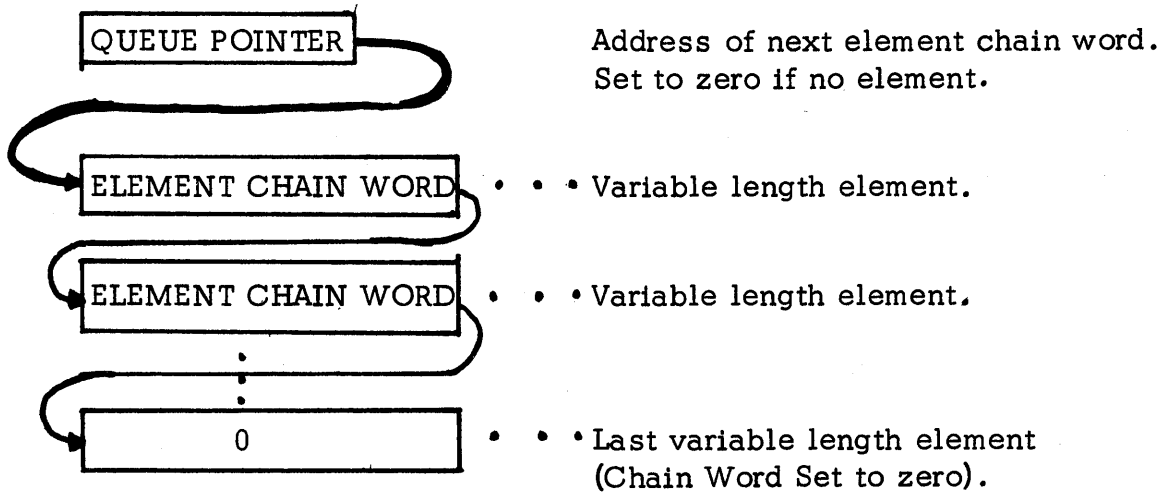
Constructs TP buffers for TPIOX transmission to HASP. Data to be inserted and length of insert are provided by user. TPPUT initializes this routine by providing the buffer to be used and setting pointers and variables.

The data to be inserted is usually in the form a logical record as constructed by TPCOMPR.

RTP1130 Control Block And Data Formats

Chained List General Format

All queues maintained within RTP1130 are of the chained list form and consist of free queues and free queue pointers and active queues and active queue pointers. Free queues are chained in a random fashion while active queues are maintained in a first-in, first-out order. The general form of a queue is:



Examples of chained lists are: TP buffers, console message tanks, printer data tanks, punch data tanks. The size and number of elements in the queue is variable according to the nature of the queue.

UFCB - Unit-Function Control Block Description

Each device which transmits data to or from HASP via the communications adapter processors must be represented by a unit-function control block. The general format of a UFCB is:

REFERENCE	WORD	DESCRIPTION
UFCBCNW	0	Chain word to next UFCB
UFCBNFO	1	Information word...
		Input: Byte 0 = Reserved
		Byte 1 = Input Code
		= 0 for IBM Card
		= 1 for PTTC/8
		= 2 for EBCDIC
UFCBSAR	2	Status and RCB Code...
		Byte 0 = Status of unit-function
		= X'90' if request to start sent from input unit-function or if request to start received for output unit-function

= X'A0' If permission to start received for input unit-function or if permission to start sent for output unit-function.

Byte 1 = RCB code associated with this UFCB

UFCBFCS	3	Function control sequence bit associated with this UFCB (and RCB)
UFCBCOM	4	Address of commutator processor gate address for processor associated with this UFCB
UFCBFQP	5	Tank free queue pointer for output devices or address of input element for input devices
UFCBBFP	6	Queue pointer for active TP buffers for output devices or end-of-file flag for input devices
UFCBBFC	7	Count of active TP buffers for associated device
UFCBBFL	8	Limit of active TP buffers for associated device
UFCBPBP	9	Buffer address of current buffer being processed by TPGET processor
UFCBPBA	10	Address of next RCB in buffer being processed

UFCBPBS	11	Position indicator for next RCB in buffer being processed. Set to 0 if RCB right justified. Set to 1 if RCB left justified.
UFCBPWD	12	Output device width = 2*W/P where W = actual width in characters and P = 2 for packed output tanks or P = 1 for unpacked output tanks.
UFCBPRO	13	Address of data processing routine (usually a conversion program) for each character processed by $DEBLOCK.
UFCBSTO	14	Address of routine to store data processed by "UFCBPRO" program.

TPBUF - TP Buffer Element Description

All data transmitted to or from HASP is contained in variable length buffers (variable at generation time) with the following general format:

REFERENCE	WORD	DESCRIPTION
TPBUFCW	0	Chain word to next TP buffer
TPBUFST	1	Reserved
TPBUFCB	2	Buffer control word
		Byte 0 = 0 (Reserved)
		Transmit function...
		Byte 1 = Number of bytes to be transmitted minus 2 for end sequence which is inserted by BSXIOS.
		Receive function...
		Byte 1 = Number of bytes received
		Timer function...
		Byte 1 = Number of program time interrupts processed before ending timer operation
TPBUFDT	3	Start of data area of length defined by "&TPBUFSZE" which includes...
TPBUFHD	3	BSC header value indicating the function (Read, write, timer) to be performed as defined by SCA function indicators

TPBUFBF	4	Control sequence...
		Byte 0 = BCB
		Byte 1 = first byte of FCS
TPBUFFR	5	Control sequence...
		Byte 0 = Second byte of FCS
		Byte 1 = RCB
TPBUFSR	6	Control sequence...
		Byte 0 = SRCB
		Byte 1 = SCB

Output Element (Tank) Description

Local terminal output devices (printers, punch, etc.) receive data via elements or tanks which are built by the commutator routine responsible for processing TP buffers transmitted by HASP. The general format of these tanks is described below.

REFERENCE	WORD	DESCRIPTION
TANKWRDA	0	Chain word to next tank
TANKWRDB	1	Reserved
TANKWRDC	2	Control word Byte 0 = Reserved for device use Byte 1 = SRCB from record received
TANKWRDD	3	Control word Byte 0 = Reserved for device use Byte 1 = Actual tank data count
TANKWRDE	4	Start of variable length data area determined at generation time

Note: The element chain word and the data area must start on even 1130 word boundaries.

Object Deck Format

The following is the format of the object decks (RPT1130, RTPLOAD) produced from OS/360 assembler output by LETRRIP.

Text Card

Column(s)	Description
1	'T' for text card identification
2-3	Absolute 1130 load address
4	Word count of data field
5-72	Data field (maximum of 34 words)
73-74	Checksum of columns 1-72
75-76	Identification
77-80	Sequence number

End Card

Column(s)	Description
1	'E' for end card identification
2-3	Entry point to program loaded
4-72	Reserved
73-74	Checksum of columns 1-72
75-76	Identification
77-80	Sequence number

REP Card Format

Column(s)	Description
1	Any legal EBCDIC punch
2-4	"REP"
5	Blank
6	Load address format field: "L" for listing option where the specified load address corresponds to the OS/360 assembler listing. "X" for absolute 1130 core address
7	Currently unused but usually punched "0" for continuity
8-11	Load address for first data word and is incremented by 1 for each additional data word. REP cards may be continued by leaving this field blank
12	Blank
13	Format field for data following. Subject to same definition as column 6.
14-17	Data field to be loaded in the location computed as a function of columns 8-11
18	","
•	
•	
•	

Columns 19 through 78 in the same format as columns 13-18 with the exception of column 78 which must be blank. A blank in columns 18, 24,...72 terminates the scan of the card.

Note: The "L" option causes the specified data to be divided by 2 for conversion from 360 byte data to 1130 word data.

Examples of REP Cards

1. The following cards:

```
Col

0   00      11
1   56      23

RREP L02208 X4C00,L004E,X4400,X000F

RREP        X74FF,X0000,X7101
```

Would result in the code represented below starting in 1130 core location 1104 (Hex):

```
1104    $B      39,,L
1106    $TSL    15
1108    $MDM    0,-1
110A    $MDX    1,1
```

2. The following card:

```
Col

0   00      11
1   56      23

RRER L01772 X4C18,X1FF8
```

Would be ignored because columns 2-4 not equal to "REP"

4.14.2 Remote Terminal Main Loader (RTPLOAD)

RTPLOAD is an EBCDIC format loader which is loaded by RTPBOOT into the upper part of defined 1130 core. The 1130 core definition (which is a RMTGEN variable) is used to specify the origin of RTPLOAD. The format of RTPLOAD (and RTP1130) is given in Section 4.14.1 under Control Blocks and Data Formats.

RTPLOAD also reads and processes "REP" cards as well as the optional /*SIGNON control card.

The major functions of RTPLOAD are:

- Clears core from location 0 to "&RTPLORG-1"
- Tests for a 2501 or 1442 card reader and initializes the card read routine for the appropriate device.
- Reads RTP1130 program cards, performing the conversion from card code to EBCDIC and loading the data into the specified locations.
- Sets up the entry to RTP1130 when the end card is processed.
- Reads and processes REP cards, if they exist.
- Reads, converts and stores /*SIGNON and sets indicator for RTP1130 signalling existance if /*SIGNON encountered.
- Transfers control to RTP1130

4.14.3 Remote Terminal Bootstrap (RTPBOOT)

The bootstrap loader distributed in object form as shown in the subsequent pages is specifically constructed to "bootstrap" the EBCDIC main loader (RTPLOAD) into the core locations defined by "&RTPLORG" at RMTGEN time. RTPBOOT loads into lower 1130 core via the load-mode format first card and following binary program cards and EBCDIC conversion table cards. RTPBOOT will load from a 2501 or 1442 card reader which is wired for the load-mode sequence initiated by the console "LOAD" button.

Figure 4.14.3 - Remote Terminal Bootstrap Card Format

Card Col.	Card No. 1	Card No. 2	Card No. 3	Card No. 4
1	12-11-7	12	12-11-1-2-3-4-5	12-11-1
2	1-2-9	11-0-3-5	blank	12
3	12-11-1-8	11	5	12-11-2-3-4-5
4	12-11-7-8-9	blank	11-0-1-5	12-11-1
5	11-0-1-6-9	5	4	5
6	0-2-6	11-0-1-5	11-0-1-5	11-0-1-5
7	4-7-8-9	12-11-0-1-2-4-5	0-1-2-3-4	12-11-0-2-3-4-5
8	blank	12-11	11	11-0-1
9	4-6	blank	blank	blank
10	0-1-2	12-11-1-5	12-11-1-4	11-0-3-5
11	blank	5	5	0-3
12	11-2-5	1-2	11-0-1-4	5
13	4-5-9	12-11-0-1-4-5	12-11-0-1-2-3-4-5	blank
14	12-0-1-2-5-6	12-11-1	11-0-1-4-5	12-1-5
15	1-2-8	blank	12-11-0-1-2-4	5
16	12-11-1-3-4-5-6-8-9	12-11-3	11-0-1	12-1-5
17	12-11-3-4-5-6-7	5	12-3-4-5	12-11-2
18	1-2-8-9	blank	12-11	12-11-1
19	12-11-1-3-4-5-6-8	12-11-0-1-2-3-4-5	12-0-3	1-5
20	12-3-4-5-7-9	11-0-1-3	11-0-1	11
21	12-11-1-3-4-5-7	11-2-4-5	blank	12-11-3-4
22	12-11-4-7	blank	12-11-3	12-11-0-1
23	1-6	12-11-3-4-5	12-11-1-2-3	1-2
24	12-11-1-4-8	11-0-1	blank	11-2-3
25	12-4-7-8	2-3-4-5	blank	11-0-2-3-4-5
26	12-11-1-4-7-9	11-0-1-3	11-0-3-4-5	blank
27	12-4-8	11-2-4	2-3-4-5	12-11-0-1-2-3-4-5
28	12-11-1-4-9	blank	4	11-0-1
29	12-11-3-4-6-9	12-11-3	0-2-4	5
30	1-6-9	11-0-1	11	11-0-1-4-5
31	12-11-1-3-4-6	3-5	5	12-11-0-1-2-3-4-5
32	1-2-6	11-0-5	11-0-1	11-0-1-4
33	12-11-1-4-6-7-9	3-4-5	3-4	12-11-0-2
34	12-11-1-5-6-7-8	11-0-1-3	11-0-1	11-0-1
35	12-11-1-5-6-8-9	11-2-4	blank	12-11-0-1-2-3-4-5
36	12-11-1-3-4-8	blank	11-0-3-4-5	11-0-1
37	12-11-1-3-4-7-9	blank	12-11-0-1-5	5
38	2-3-5-6-7-8	11-0-3-4	5	blank
39	2-3-5-6-7-8-9	11-0-2-4-5	0-3-5	blank
40	11-0-1-3-4-5-6-7-8-9	4	11	12-11-0-1-4-5

Figure 4.14.3(CONT) - Remote Terminal Bootstrap Card Format

Card Col.	Card No. 1	Card No. 2	Card No. 3	Card No. 4
41	9	1-3	12-11-0-1-4-5	0
42	2-3-4-8	11-0-1-3	11-0-1	11-2-3
43	12-11-3-5-6-7-8-9	2-3	11-2-3-4-5	12-0-1-3-5
44	12-8-9	blank	11-0-1-3	blank
45	12-11-1-3-5-6-7-9	12-0-1	12-0-2-5	5
46	2-3-5-6-7	11-0-1-4	blank	11-0-1-3
47	11-2-3-4-5-6	12-0-4-5	1	12-0-2-3-4-5
48	9	11-0-2-4	1-2	blank
49	11-0-1-3-4-5-6-7-8-9	12-1-2-3-4	12-11-1-2-3-4-5	blank
50	9	11-0-2-4	12-11-1	12-11-0-1-4-5
51	12-11-6-7-9	11-2-3-4-5	12-0-1-3-4	12
52	12-3-4-5-6-8-9	12-11	11-0-5	11-2-3
53	12-11-1-6-8-9	blank	blank	12-0-2-3-4-5
54	12-11-6	12-11-1-4	12-11-3-5	blank
55	12-3-4-5-6	12-11-0-1-2-3-4-5	0-3-4	11-1
56	12-11-1-8-9	11-0-1-5	5	blank
57	12-3-4-5-7-8	12-0-1-2-5	blank	blank
58	12-11-1-7	11-0-1	11-0-3-4-5	12-11-5
59	3-7	1-2-4-5	0-2-3	2
60	blank	11-0-1-3	5	1
61	1-2-6	11-2-4	blank	5
62	1-2	blank	11-0-3-4	12-11-0-2-5
63	blank	12-0-1-3-4	blank	12
64	1	11-0-1	5	11-2-3
65	blank	blank	1-2	12-0-1-2
66	12-11-7-8-9	11-0-3-5	11	blank
67	12-1-3-4-6-7	12-11-1-2-3-5	1-4-5	blank
68	12-11-1-7-9	blank	11-0-1	11-0-3-5
69	11-2-4	11-3-4	blank	12-11-1-2-3-5
70	11-0-1-3-4-6-7	11	12-11-3	blank
71	2-3-7-9	blank	4-5	12-11-0-1-3-4-5
72	2-3	12-11-1-5	blank	11
73	11-2-3-4-5-6	12	12-11-0-1-2	5
74	4-7-8-9	11-0-3-4	12-1	12-11-1
75	11-0-1-7	12-11-1-2-3-5	blank	blank
76	8	blank	12-11-1-3-5	11-2-3
77	blank	12	0-3-4	blank
78	blank	11-0-3-4-5	5	blank
79	0	0	0	0
80	1	2	3	4

Figure 4.14.3 (CONT) - Remote Terminal Bootstrap Card Format

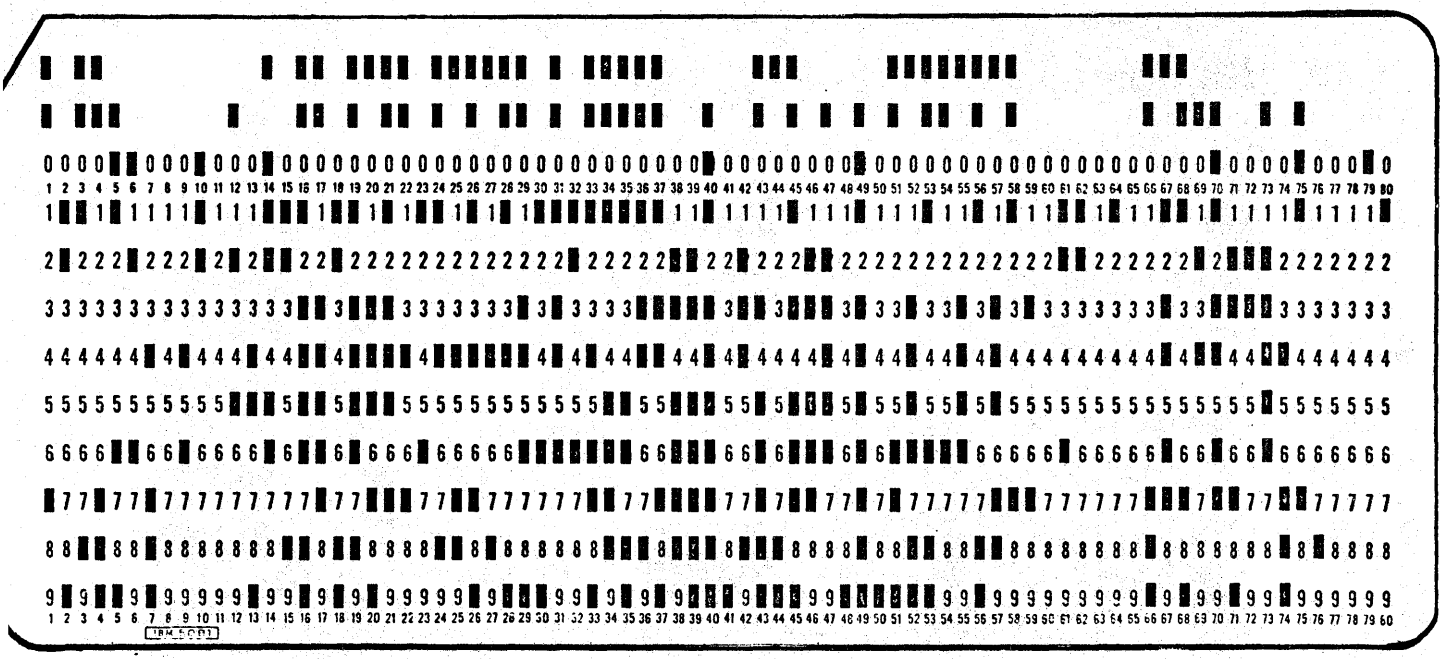
Card Col.	Card No. 5	Card No. 6	Card No. 7	Card No. 8
1	0	11-0-1-8	12	12-0-1-8-9
2	1	11-0-1	12-11-1-9	12-1-9
3	2	11-0-2	12-11-2-9	12-2-9
4	3	11-0-3	12-11-3-9	12-3-9
5	4	11-0-4	12-11-4-9	12-4-9
6	5	11-0-5	12-11-5-9	12-5-9
7	6	11-0-6	12-11-6-9	12-6-9
8	7	11-0-7	12-11-7-9	12-7-9
9	8	11-0-8	12-11-8-9	12-8-9
10	9	11-0-9	11-1-8	12-1-8-9
11	12-11-0-2-8-9	11-0-2-8	11-2-8	12-2-8-9
12	12-11-0-3-8-9	11-0-3-8	11-3-8	12-3-8-9
13	12-11-0-4-8-9	11-0-4-8	11-4-8	12-4-8-9
14	12-11-0-5-8-9	11-0-5-8	11-5-8	12-5-8-9
15	12-11-0-6-8-9	11-0-6-8	11-6-8	12-6-8-9
16	12-11-0-7-8-9	11-0-7-8	11-7-8	12-7-8-9
17	blank	12-11-0-1-8	11	12-11-1-8-9
18	.	12-11-0-1	0-1	11-1-9
19	.	12-11-0-2	11-0-2-9	11-2-9
20	.	12-11-0-3	11-0-3-9	11-3-9
21	.	12-11-0-4	11-0-4-9	11-4-9
22	.	12-11-0-5	11-0-5-9	11-5-9
23	.	12-11-0-6	11-0-6-9	11-6-9
24	.	12-11-0-7	11-0-7-9	11-7-9
25	.	12-11-0-8	11-0-8-9	11-8-9
26	.	12-11-0-9	0-1-8	11-1-8-9
27	.	12-11-0-2-8	12-11	11-2-8-9
28	.	12-11-0-3-8	0-3-8	11-3-8-9
29	.	12-11-0-4-8	0-4-8	11-4-8-9
30	.	12-11-0-5-8	0-5-8	11-5-8-9
31	.	12-11-0-6-8	0-6-8	11-6-8-9
32	.	12-11-0-7-8	0-7-8	11-7-8-9
33	.	12-0	12-11-0	11-0-1-8-9
34	.	12-1	12-11-0-1-9	0-1-9
35	.	12-2	12-11-0-2-9	0-2-9
36	.	12-3	12-11-0-3-9	0-3-9
37	.	12-4	12-11-0-4-9	0-4-9
38	.	12-5	12-11-0-5-9	0-5-9
39	.	12-6	12-11-0-6-9	0-6-9
40	.	12-7	12-11-0-7-9	0-7-9

Figure 4.14.3 (CONT) - Remote Terminal Bootstrap Card Format

Card Col.	Card No. 5	Card No. 6	Card No. 7	Card No. 8
41	blank	12-8	12-11-0-8-9	0-8-9
42	.	12-9	1-8	0-1-8-9
43	.	12-0-2-8-9	2-8	0-2-8-9
44	.	12-0-3-8-9	3-8	0-3-8-9
45	.	12-0-4-8-9	4-8	0-4-8-9
46	.	12-0-5-8-9	5-8	0-5-8-9
47	.	12-0-6-8-9	6-8	0-6-8-9
48	.	12-0-7-8-9	7-8	0-7-8-9
49	.	11-0	12-0-1-8	12-11-0-1-8-9
50	.	11-1	12-0-1	1-9
51	.	11-2	12-0-2	2-9
52	.	11-3	12-0-3	3-9
53	.	11-4	12-0-4	4-9
54	.	11-5	12-0-5	5-9
55	.	11-6	12-0-6	6-9
56	.	11-7	12-0-7	7-9
57	.	11-8	12-0-8	8-9
58	.	11-9	12-0-9	1-8-9
59	.	12-11-2-8-9	12-0-2-8	2-8-9
60	.	12-11-3-8-9	12-0-3-8	3-8-9
61	.	12-11-4-8-9	12-0-4-8	4-8-9
62	.	12-11-5-8-9	12-0-5-8	5-8-9
63	.	12-11-6-8-9	12-0-6-8	6-8-9
64	.	12-11-7-8-9	12-0-7-8	7-8-9
65	.	0-2-8	12-11-1-8	blank
66	.	11-0-1-9	12-11-1	12-0-1-9
67	.	0-2	12-11-2	12-0-2-9
68	.	0-3	12-11-3	12-0-3-9
69	.	0-4	12-11-4	12-0-4-9
70	.	0-5	12-11-5	12-0-5-9
71	.	0-6	12-11-6	12-0-6-9
72	.	0-7	12-11-7	12-0-7-9
73	.	0-8	12-11-8	12-0-8-9
74	.	0-9	12-11-9	12-1-8
75	.	11-0-2-8-9	12-11-2-8	12-2-8
76	.	11-0-3-8-9	12-11-3-8	12-3-8
77	.	11-0-4-8-9	12-11-4-8	12-4-8
78	.	11-0-5-8-9	12-11-5-8	12-5-8
79	.	11-0-6-8-9	12-11-6-8	12-6-8
80	blank	11-0-7-8-9	12-11-7-8	12-7-8

Figure 4.14.3 (CONT) - Remote Terminal Bootstrap Card Format

CARD 1



CARD 2

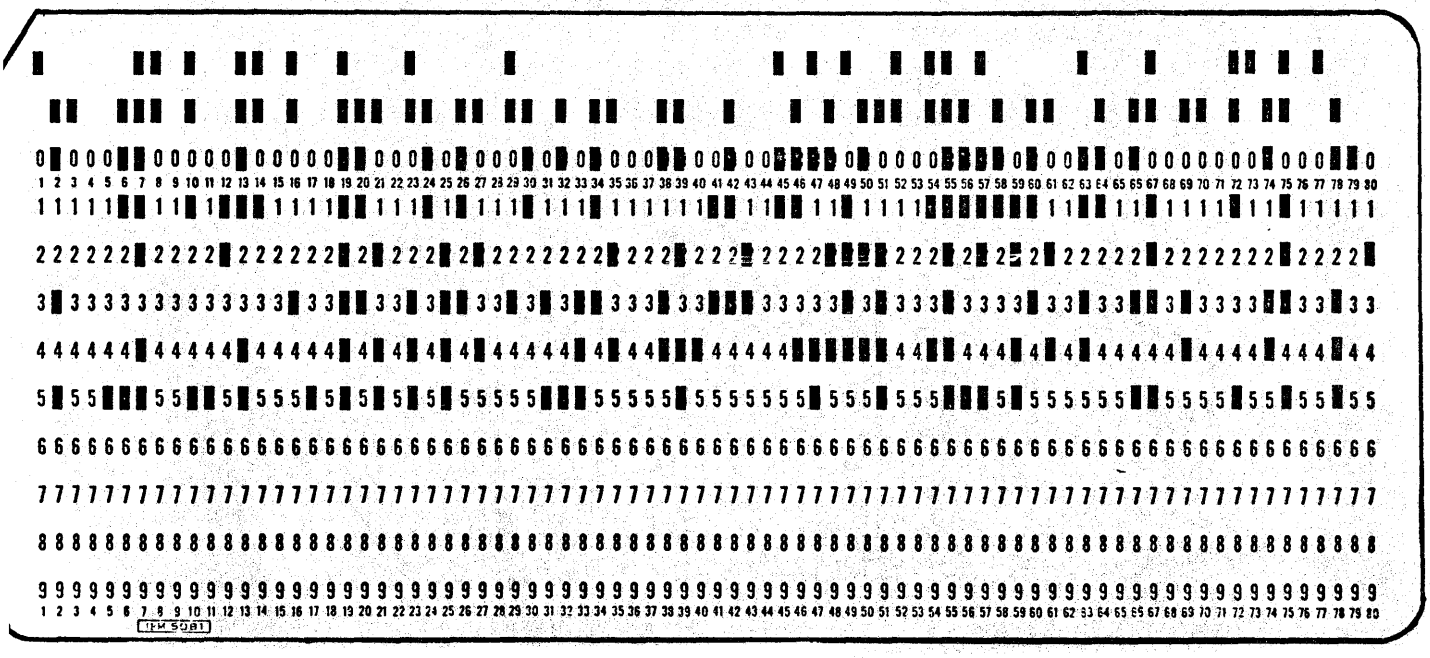
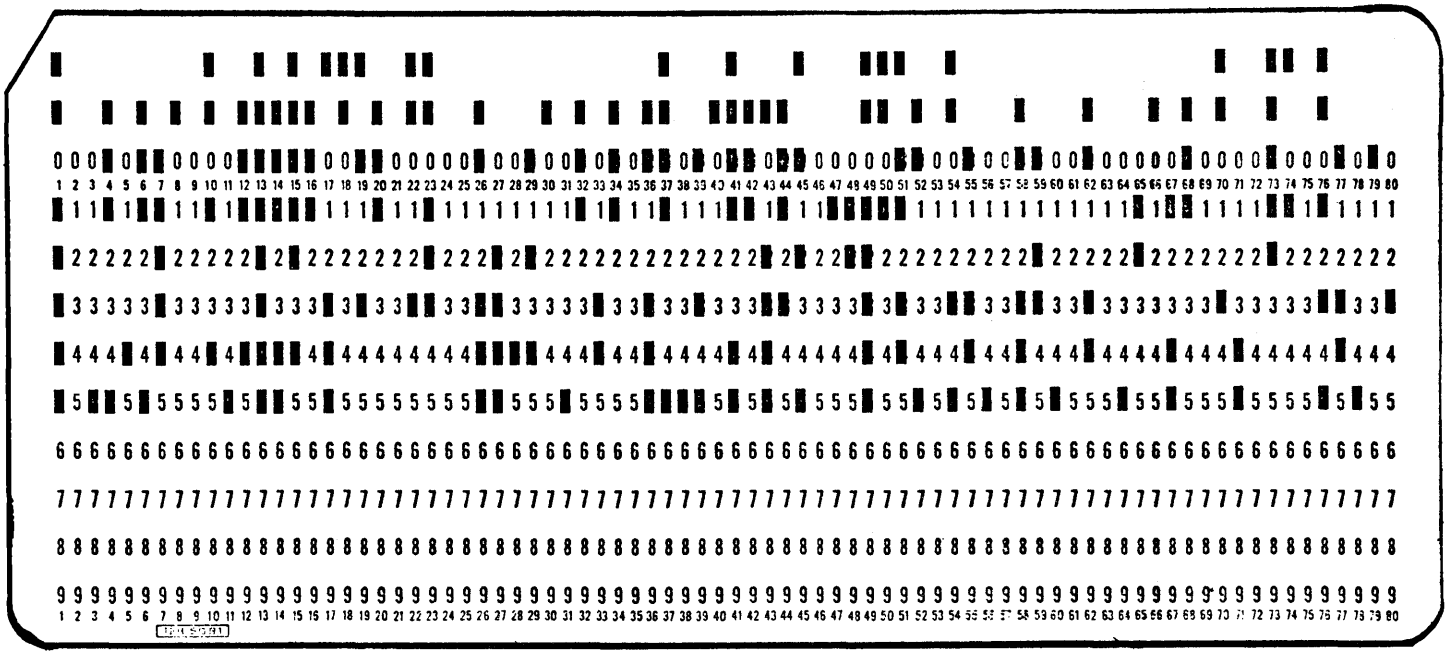


Figure 4.14.3 (CONT) - Remote Terminal Bootstrap Card Format

CARD 3



CARD 4

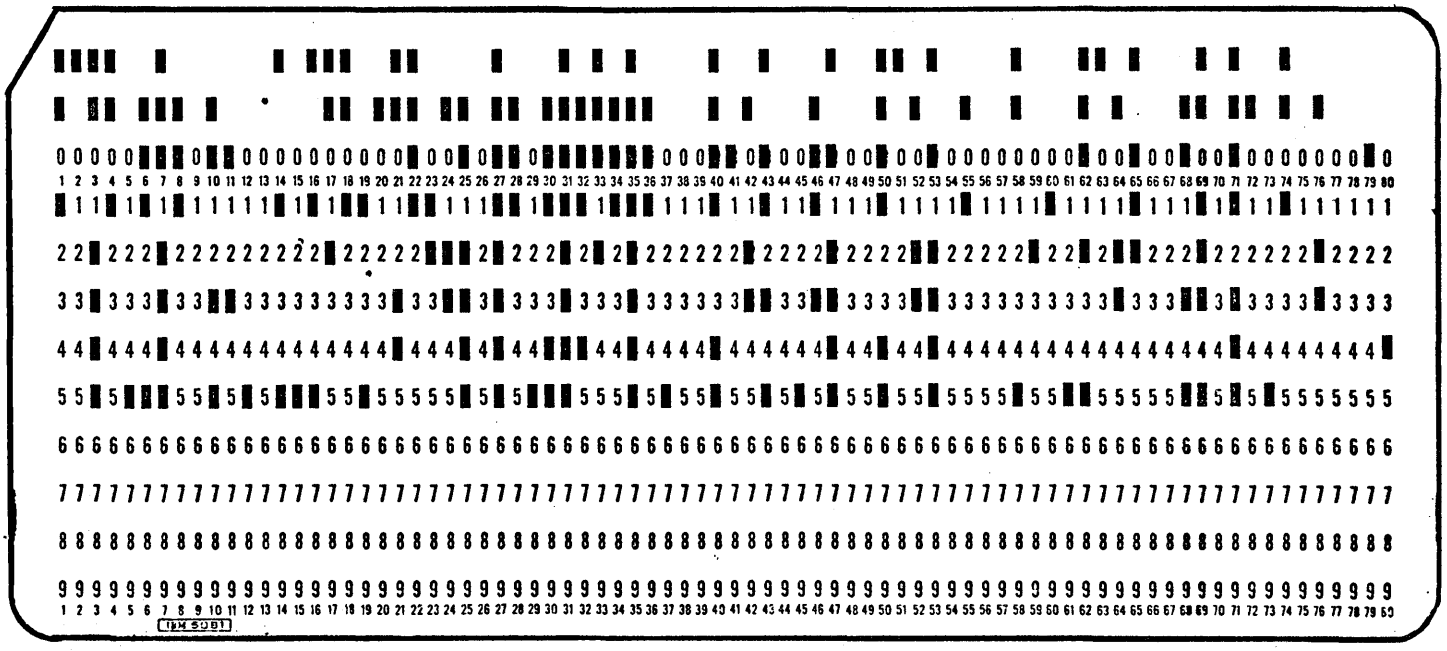
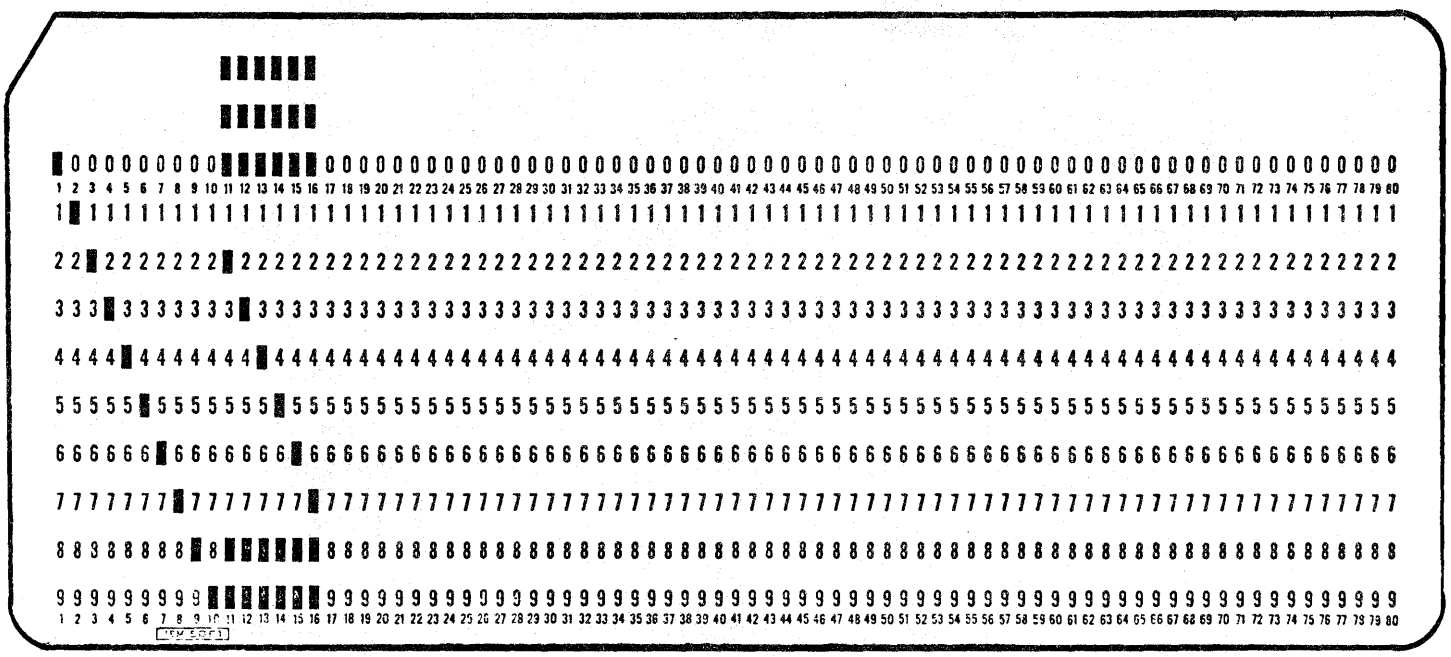


Figure 4.14.3 (CONT) - Remote Terminal Bootstrap Card Format

CARD 5



CARD 6

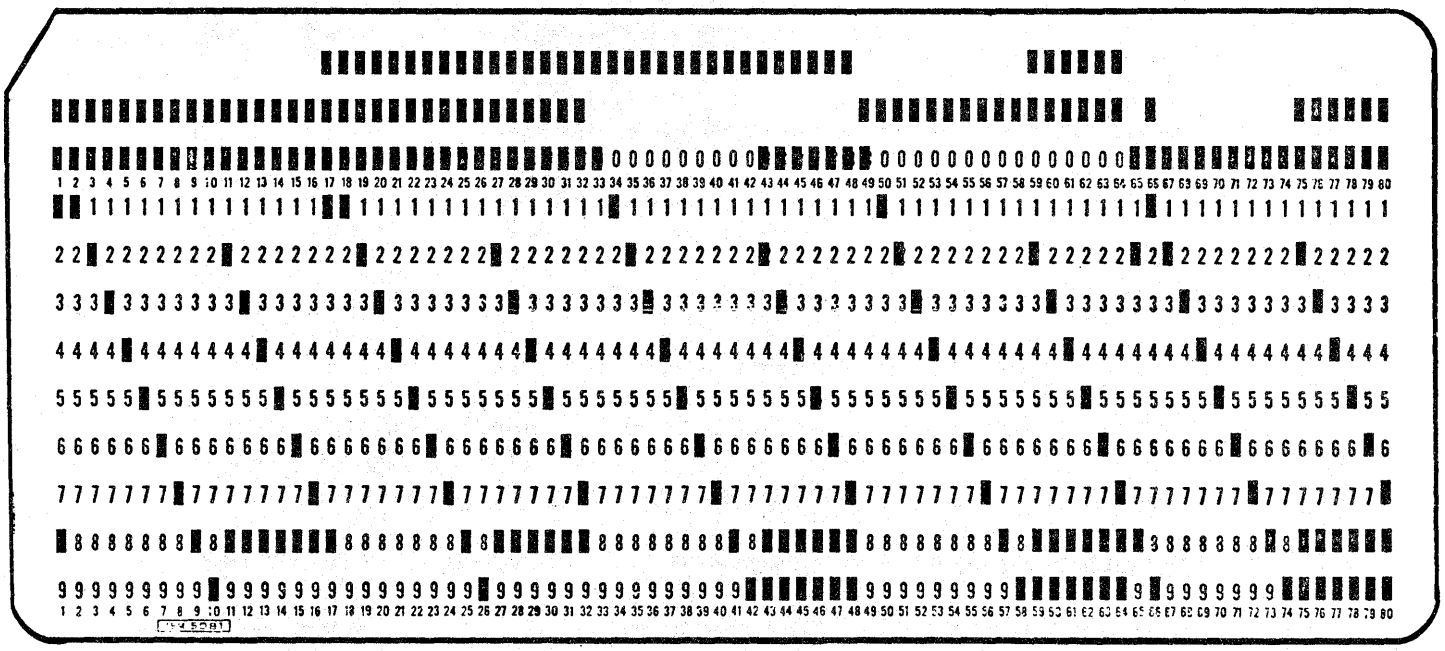
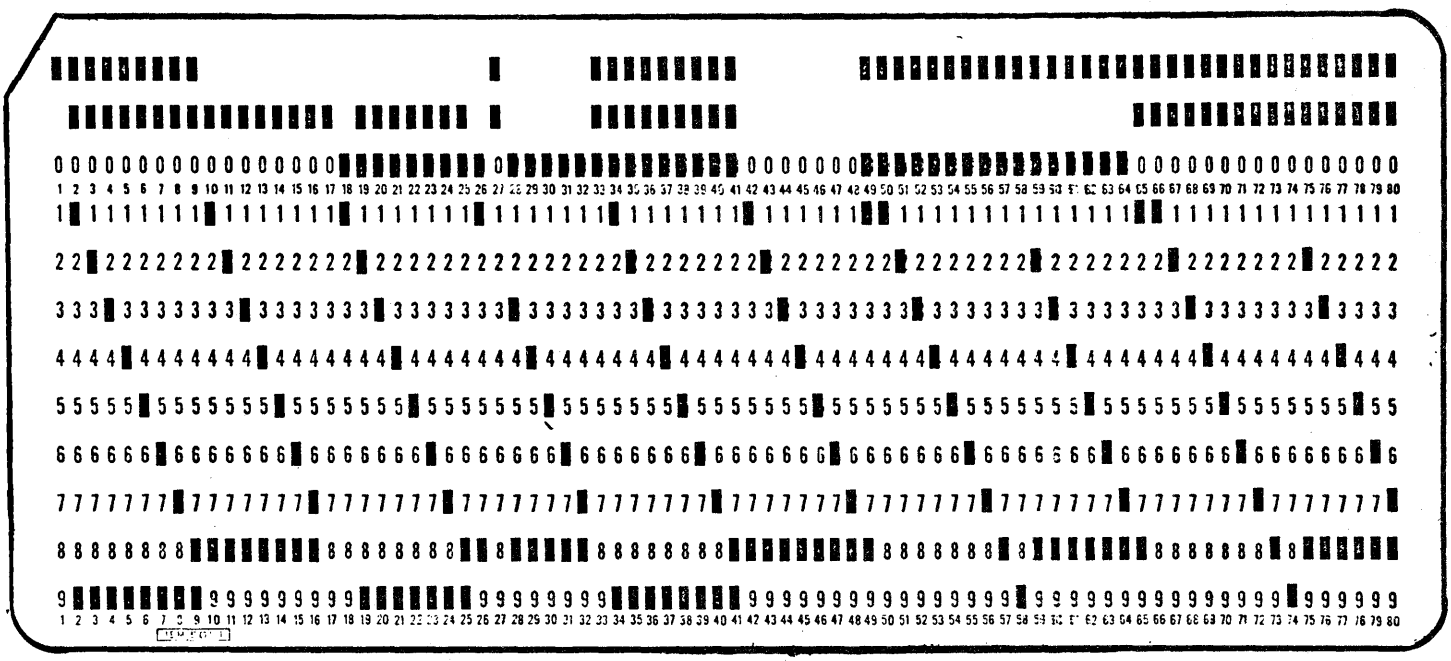
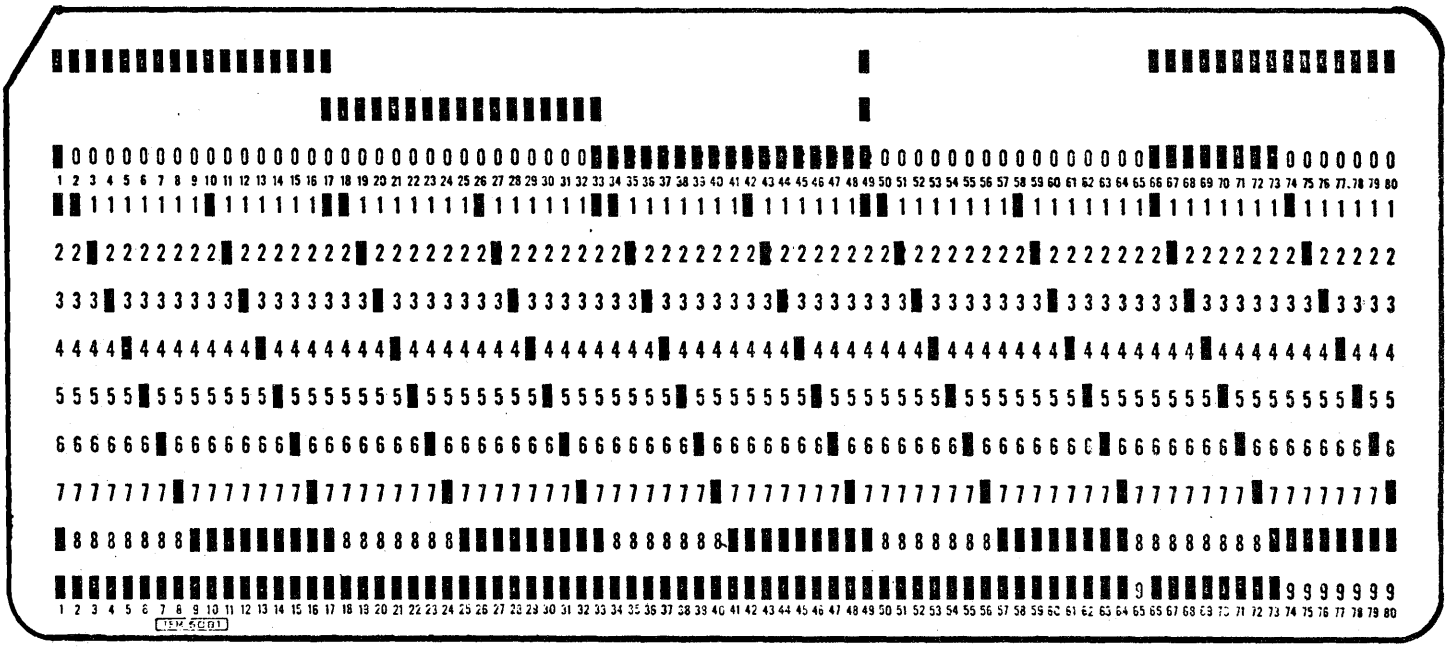


Figure 4.14.3 (CONT) - Remote Terminal Bootstrap Card Format

CARD 7



CARD 8



4.14.4 Remote Terminal Program 360 Processing (LETRRIP)

LETRRIP (Loader for Eleven-Thirty Relocatable Remote Interleaving Processor) is a 360 program executed under OS/360 as part of the RMTGEN procedure. The purpose of this program is to condense the object deck produced by the 360 assembler; relocate address constants according to the requirements of the 1130 and to produce a new object deck in the format as described in Section 4.14.1.

4.14.5 1130 Instruction Macros

The OS/360 Assembler Macro instructions listed on the following pages are used to assemble the RTP1130 and RTPLOAD programs as a part of the RMTGEN process necessary to create the 1130 workstation program.

The general format of the instructions to be assembled with the macros is:

```
LABEL $OP ADDR,TAG,FMT,MOD
```

Where:

"LABEL" is the statement label subject to the OS/360 assembler rules and restrictions.

"$OP" is a macro from the set listed at the end of this section.

"ADDR" is the address field of the 1130 instruction.

"TAG" is the index register (TAG) field of the 1130 instruction.

"FMT" is the format indicator for the 1130 instruction:

```
FMT=L for long form
FMT=I for long form indirect address
FMT=X for short form absolute address
FMT='blank' for short form relative address
```

"MOD" is the modifier bits field required for some 1130 instructions.

Listed below are some of the conventions which must be followed to successfully use the macro package in producing a program for operation on an 1130.

1. All symbols starting with the character "$" are deemed to be absolute in value.

2. The symbols WA, WB and WC are assumed to define absolute values. Note: WA, WB and WC cannot be used as the first two characters of any relocatable symbols.

3. All other symbols are assumed to be relocatable as defined by the OS/360 assembler SRL.

4. Parenthetical expressions are considered to be relocatable if contained in an instruction, e.g.,

```
     $AXT (*-*),WA,L
```

   is considered relocatable, where

```
     $AXT *-*,WA,L
```

   is considered absolute.

1130 Instruction Macros

Macro Form		Description And Notes
$LD	ADD,TAG,FMT	Load ACC
$LDD	ADD,TAG,FMT	Load double (ACC,EXT)
$STO	ADD,TAG, FMT	Store ACC
$STD	ADD,TAG,FMT	Store double (ACC,EXT)
$LDX	ADD,TAG,FMT	Load index
$LXA	ADD,TAG	Load index from address. A variation of $LDX with F = 1 and IA = 1.
$AXT	ADD,TAG,FMT	Address to index true. Identical to $LDX.
$STX	ADD,TAG,FMT	Store index
$STS	ADD,TAG,FMT	Store status
$LDS	ADD,TAG	Load status
$A	ADD,TAG,FMT	Add
$AD	ADD,TAG,FMT	Add double
$S	ADD,TAG,FMT	Subtract
$SD	ADD,TAG,FMT	Subtract double
$M	ADD,TAG,FMT	Multiply
$D	ADD,TAG,FMT	Divide
$AND	ADD,TAG,FMT	Logical AND
$OR	ADD,TAG,FMT	Logical OR
$EOR	ADD,TAG,FMT	Logical Exclusive OR

Macro Form		Description And Notes
$SLA	ADD,TAG	Shift left ACC
$SLCA	ADD,TAG	Shift left and count ACC
$SLC	ADD,TAG	Shift left and count ACC and EXT
$SRA	ADD,TAG	Shift right ACC
$SRT	ADD,TAG	Shift right ACC and EXT
$RTE	ADD,TAG	Rotate right ACC and EXT
$BSC	ADD,TAG,FMT,MOD	Branch/Skip on condition
$BOSC	ADD,TAG,FMT,MOD	Branch/Skip and reset interrupt
$BP	ADD,TAG,FMT	Branch ACC positive (long)
$BNP	ADD,TAG,FMT	Branch ACC not positive (long)
$BN	ADD,TAG,FMT	Branch ACC negative (long)
$BNN	ADD,TAG,FMT	Branch ACC not negative (long)
$BZ	ADD,TAG,FMT	Branch ACC zero (long)
$BNZ	ADD,TAG,FMT	Branch ACC not zero (long)
$BC	ADD,TAG,FMT	Branch on carry (long)
$BO	ADD,TAG,FMT	Branch on overflow (long)
$BOD	ADD,TAG,FMT	Branch ACC odd (long)
$SKPP		Skip ACC positive (short)
$SKPN		Skip ACC non-zero (short)
$SKPZ		Skip ACC zero (short)
$SKPO		Skip overflow off (short)

Macro Form		Description And Notes
$SKPC		Skip carry off (short)
$SKPX		Skip ACC not equal zero and carry off (short)
$B	ADD,TAG,FMT	Branch unconditionally. FMT = L or I generates long form $BSC with MOD = 0. FMT = X or blank generates $MDX ADD,TAG,FMT
$BSI	ADD,TAG,FMT,MOD	Branch conditionally and store IAR
$TSL	ADD,TAG,FMT	Transfer and store location counter. Assembled as a $BSI with FMT = L,MOD = 0 (long form unconditional branch and store IAR).
$MDX	ADD,TAG,FMT	Modify index and skip
$STL	ADD,FMT	Store location counter. Assembles as $STX ADD,0,FMT.
$MDM	ADD,VALUE	Modify memory.
$WAIT		Wait for interrupt
$XIO	ADD,TAG,FMT	Execute I/O
$BSS	N,X	Block started by symbol N = number of words. X = E for even storage.
$BES	N,X	Block ended by symbol N = number of words X = E for even storage

Macro Form		Description and Notes
$NULL		Null operation for symbol definition
$ADCON	ADDR	Address constant. Assembles as an absolute 1130 address. "ADDR" must be a relocatable symbol by the OS assembler definition.
$NOP		No operation. Assembles as $SLA 0
$ZAC		Clear ACC. Assembles as $SRA 16

4.14.6 GENERAL INFORMATION

OS/360 ASSEMBLY OUTPUT

If the value of &FULLIST is set to 1 at the time of generation of RTP1130 or RTPLOAD then the listing produced by the OS/360 Assembler will contain the following information:

1. The location counter value for each 1130 instruction or storage location in terms of bytes. The actual 1130 location in terms of words can be determined by dividing the displayed value by 2. The REP facility allows a specification of either byte or word form.

2. The 1130 instruction is printed in 1130 format. The long form address is in terms of 1130 words and the short form is true relative format.

VARIABLE INTERNAL PARAMETERS

The generation of the RTP1130 program using RMTGEN provides the user with a simple and flexible means of changing common parameters germane to the configuration of the 1130. Additional internal parameters may be varied by using the source file update feature of the RMTGEN program.

Listed below are the major parameters, with a brief description of each, which the user might consider altering as a function of hardware and software performance considerations.

VARIABLE	DESCRIPTION
&DEBUG	Conditionally assembles the RTP1130 internal core dump program ($SDUMP) and the BSC adapter trace routine (DBUGSCAL). Default value inhibits the assembly of these debugging programs.
&CNPSIZE	Maximum console printer message size. Default value is 120 bytes per message.
&CONINSZ	Maximum console keyboard input buffer size. Default value is 120 characters per command.
&PRFOTKL	Number of 1403 printer buffers (tanks) provided at assembly time. Default value is 2. The TPGET processor will build up to the value of &PRFOTKL and then suspend operation for the 1403 until the count of buffers falls below &PRFOTKL

VARIABLE	DESCRIPTION
&PRETTKL	Number of 1132 printer buffers (tanks) provided at assembly time. Default value is 2. See &PRFOTKL for TPGET action.
&PUNFTKL	Number of 1442 punch buffers (tanks) provided at assembly time. Default value is 2. See &PRFOTKL for TPGET action.
&CONSTKL	Number of console printer buffers (tanks) provided at assembly time. Default value is 5. See &PRFOTKL for TPGET action.
&PRFOBFL	Maximum number of TP buffers containing data destined for the 1403 printer which will be accepted by TPIOX before setting the transmission suspension bit defined in the FCS for the 1403. HASP will suspend transmission of 1403 print data until the FCS bit is reset when the number of 1403 TP buffers becomes less than the value of &PRFOBFL. Default value is 2.
&PRETBFL	Same definition as &PRFOBFL except it applies to the 1132 printer. Default value is 2.
&PUNFBFL	Same definition as &PRFOBFL except it applies to the 1442 punch. Default value is 2.
&CNSPBFL	Same definition as &PRFOBFL except it applies to the console printer. Default value is 1.
&NPTFBFL	Maximum number of TP buffers allotted to input devices collecting data to be sent to HASP. Default value is one greater than the number of card readers defined for RTP1130.

4.15 EXECUTION TASK MONITOR

4.15.1 Execution Task Monitor - General Description

The Execution Task Monitor is a processor which periodically examines the CPU utilization of user tasks within a dynamic priority group and rearranges the OS/360 task dispatching chain giving higher priority to those tasks, within the group, which use the least amount of CPU time. Tasks above and below the dynamic priority group are not affected by the rearrangement of the dispatching chain. Tasks with all of the following characteristics are included within the dynamic priority group:

1. The task belongs to a job scheduled by HASP.

2. The current dispatching priority of the task is

   a. equal to priority of the dynamic group as specified by the value of the &XZPRTY parameter for MVT or

   b. not greater than the value &XZMFTH and not less than the value of &XZMFTL for MFT.

3. The HASPGEN parameter &XZMULT is set to "YES" or if not "YES" the task is a job step with no daughters.

The interval between the periodic examinations is controlled by the value of the &MONINTV parameter. Setting &MONINTV value to a positive integer will cause the processor to be generated. OS/360 must support Job Step Timing and must not have a Time Slicing Group Defined at the priority level(s) corresponding to the priority range of the dynamic group. Users must not use TIME=1440 on Job or Execute cards for jobs to be correctly adjusted within the dynamic group.

4.15.2 Execution Task Monitor - Algorithm

The Execution Task Monitor determines the CPU utilization history ($h_{t,n}$) for each task within the dynamic priority group using the following formula:

$$h_{t,n}=cpu_{t,n}+h_{t-1,n}-H_t/N$$

where: $H_t = cpu_{t,1}+h_{t-1,1}+cpu_{t,2}+h_{t-1,2}+\ldots cpu_{t,N}+h_{t-1,N}$

= Total CPU counts observed for the N tasks being monitored plus the sum of the previous history values.

N = The number of tasks being monitored at the end of the time interval.

$h_{t,n}$ = The history of CPU utilization for task (n) during the current time interval.

$h_{t-1,n}$ = The history of CPU utilization for task (n) taken at the previous time interval.

New tasks, entering the monitored group, will be assigned a history value of zero and temporarily placed at the low priority end of the group. Task with continuous low values of CPU counts will have (h) values which become increasingly negative. The (h) values will be prevented from falling below the range of one time interval; thus providing responsiveness to erratic changes in the corresponding task's CPU utilization.

Low values of h indicate the task (1) has not been able to utilize the CPU time given to it because of waiting for events such as I/O or (2)

has not been given the opportunity to utilize the CPU. High values of h indicate the task has had the opportunity and has utilized the CPU. The Execution Task Monitor performs a partial sort and rechains the monitored tasks, insuring that the task with the largest history of CPU utilization will have the lowest effective priority within the dynamic priority group during the next time interval. This will by default raise the effective priority of other tasks in the group.

When HASPGEN parameter &XZMULT is set to "YES" all tasks for a jobstep falling within the priority group are ordered as a single unit allowing each task to maintain the same relative priority with other monitored tasks within the jobstep. In MVT systems, a task within the group that changes OS dispatching priority is removed from the group.

In MFT systems, all monitored tasks are assigned an OS dispatching priority specified by the &XZMFTL parameter. A CHAP or ATTACH that specifies a change in priority will have the effect of changing relative priority; however, as long as the task remains within the range &XZMFTH and &XZMFTL the assigned priority will be changed to &XZMFTL at the end of the monitor interval &MONINTV. A CHAP in an MFT system may therefore not produce the intended result.

4.16 INTERNAL READER

4.16.1 Internal Reader - General Description

The Internal Reader Processor is an Input Service Processor which reads card images from any system or user task running under OS/360. The Internal Reader recognizes, through the use of Execution Control Processor interface routines, an attempt by other tasks running under OS/360 to punch information into "cards" on pseudo 2520 punch devices, performs the function of the Input Service Processor on each card, and via OS/360 POST macro signals completion of I/O to the submitting task.

4.16.2 Internal Reader - Program Logic

The Internal Reader uses the code of the Input Service Processor with modifications in the following areas.

1. Processor Initialization - The Internal Reader attempts to obtain an internal reader device control table (DCT) which contains an 80 byte buffer area rather than a normal reader DCT. When a device is received the processor continues by acquiring a direct-access DCT and passes control to the main processor.
2. Main Processor - The Internal Reader RGET routine tests for the existance of a submitting task punch channel program.

If no channel program exists the Processor will wait for WORK. If a channel program exists RGET will simulate the punching of one card into the 80 byte DCT buffer area (no data chaining). If the channel command represents the end of the channel program RGET posts completion of I/O and resets channel program indicators. RGET returns to the main processor passing the card image for processing or in the event the "card" has "/*EOF" in columns 1 to 5 returns indicating end of file.

3. Processor Termination - Termination of the Internal Reader involves terminating the last job (if any), releasing the direct-access and internal reader device DCT's, and passing control to the processor initialization routines.

The Internal Reader requires supporting routines in the Execution Control Processor Asynchronous I/O Handler which perform the following functions:

1. Recognize EXCP macro references to designated internal readers as setup by HASP initialization.
2. Make the internal reader available to the Input Service Processor on first use.
3. Set up the first channel command word and IOB pointers.
4. Force the submitting task in the wait state if required.
5. Post the Input Service Processor for WORK.

4.17 MULTI-LEAVING LINE MANAGER

4.17.1 MULTI-LEAVING Line Manager — General Description

The function of this processor is to control all line activity with remote terminals. This includes line initiation/termination, remote terminal synchronization, line error recovery, and sign-on/sign-off processing. This processor interfaces very closely with the Remote Terminal Access Method described in section 5.15.

4.17.2 MULTI-LEAVING Line Manager — Program Logic

When this processor receives control from the dispatcher it first determines whether an I/O operation has completed. If not, it then scans each line (via the line Device Control Tables) to check for requested processing. When all processing has been completed the processor then returns control to the dispatcher ($WAIT's) until such time as more work becomes available.

When a channel end is detected, the channel end routine determines the sequence type of the Channel Command Word chain and branches to the appropriate section to analyze the channel end and initiate any error recovery procedures required.

The line Device Control Tables (DCT's) are scanned and when one is found to be available the Line Initiation routine is entered which acquires

the DCT, acquires a TP buffer, constructs an initial CCW chain, and initiates I/O on the line.

A single timer queue element is maintained by the Line Manager to initiate delays in line processing. This facility provides the capability of delaying a null response to a remote terminal and decreases the associated degradation. Various other timer queue elements are maintained by individual line processors to initiate other delays of varying intervals.

The code in this processor is assembled conditionally such that only the instructions required to process a given configuration will be generated.

4.18 REMOTE CONSOLE PROCESSOR

4.18.1 Remote Console Processor - General Description

The function of this processor is to process all console messages to and from remote terminals. This routine optionally saves messages to remotes which are not "signed on" MULTI-LEAVING terminals for later printing on the remote terminal printer.

4.18.2 Remote Console Processor - Program Logic

This processor receives control whenever a console message is queued for a remote terminal or whenever a console message is received from a remote terminal. The processor first examines the output queue of messages and upon encountering a message queued for a remote terminal examines the current status of the terminal. If the terminal is not an active BSC MULTI-LEAVING terminal the message is purged (the console message buffer is returned to the available queue) or the message is saved on the SPOOL1 volume if operator message SPOOLING is requested.

If the message is to be written, a Remote Console Device Control Table is constructed for the specific remote terminal, the DCT is chained onto the other DCTs for this remote, the DCT is "OPENed" by calling the Remote Terminal Access Method, all messages which are queued are written to the terminal, and the DCT is "CLOSEd" and unchained. If the message to be written is for a currently inactive or for a non-MULTI-LEAVING active remote and HASP operator message SPOOLING space is specified (&SPOLMSG ≠ 0), an attempt to save the message on the SPOOL1 volume for later printing at the remote by printer support routines is made. The remote MESSAGE SPOOLING QUEUE ($MSPOOLQ) element for the designated remote is examined for a queue HEADER entry of zero. If zero, a record is allocated from the MESSAGE ALLOCATION ($MSALLOC) Table, and the corresponding MTTR for the record is placed in both HEADER and TRAILER entries for the remote. (Non-zero but equal HEADER and TRAILER entries signify that the queue exists; however, since the last record of each remote element is always empty no data is currently queued). A record is allocated from the $MSALLOC table to represent the new end of message queue and the associated MTTR is placed in the chain field of the current HASP buffer. The HASP buffer is then filled with the operator message and any more messages for the same remote currently queued and written on the SPOOL1 volume

at the record location designated by the TRAILER MTTR for the remote. Upon completion of I/O the chain field replaces the TRAILER MTTR. The above process is repeated for additional buffers as required to empty the console message queue for the remote.

In the process of allocating message records the $MSALLOC table bit map is used. Each bit in the map when on represents a free record on the SPOOL1 volume. Allocation consists of finding the highest numbered bit that is on, turning the bit off, and converting to a corresponding MTTR. When all bits in the map are off indicating that no records are available, the messages are purged.

If an input message is to be read, a Remote Console Device Control Table is constructed and the Remote Terminal Access Method is utilized to "GET" the message. The message is written to the local console and then queued for the Command Processor.

4.19 EXECUTION THAW PROCESSOR

4.19.1 EXECUTION THAW PROCESSOR - GENERAL DESCRIPTION

XTHAW is a companion to the main Execution processor IOS interface routine called XFREEZE. XTHAW is responsible for discovering which tasks have been forcibly placed in an OS WAIT state by XFREEZE (frozen) and should now be activated (thawed) thru the OS POST ECB mechanism.

4.19.2 EXECUTION THAW PROCESSOR - PROGRAM LOGIC

XTHAW uses an IOB (HASP buffer) chain constructed thru the XTHAW PCE or the Execution Processor PCE(s) in the XPCEECB field of the PCE work area. The chain is constructed using the XTHAW or an Execution PCE depending on the reason for invoking XFREEZE. If the IOS interface section is entered while an Execution processor is active, then the XTHAW PCE is used. If an I/O request cannot be processed and an Execution processor is not active at the time of the request, then the PCE controlling the caller is used to build the chain.

XTHAW is activated ($POSTed) by the Execution Processor whenever a Job or the HASP controlled OS Reader/Interpreter is active and just prior to $WAITing for work. A special status bit (XPOSTBIT) in the XPCESTAT field of an Execution PCE is used as the primary test for processing the IOB chain. This bit is not turned on when the OS Reader/Interpreter is active and assigned to a PCE but does not have a job to process. This prevents unnecessary activation (thawing) of the Reader/Interpreter when no Jobs are available for initiation.

XTHAW performs the following major functions:

(1) Examines the XPCEECB field of the XTHAW processor. If this field is non-zero, it is used as the pointer to a chain of IOBs (HASP buffers) which contain ECBs to be POSTed (thawed) and the HASP/OS POST subroutine (WPOSTECB) is used to perform the POST. The chain address for the IOBs is contained in the IOBCSW field which is set to zero for the last IOB.
(2) Next, the Execution PCEs are searched for the XPOSTBIT condition and the XPCEECB field is processed as described in Step 1, if the XPOSTBIT is on.
(3) XTHAW $WAITs for work after processing all PCEs as described.

4.20 OVERLAY ROLL PROCESSOR

4.20.1 Overlay Roll - General Description

This Processor operates in conjunction with the Overlay Service Routines. Description of them in Section 5.16 should be read to provide proper background to understanding of Overlay Roll. The objective of this Processor is to prevent system lockout due to $WAITs in overlay routine coding.

4.20.2 Overlay Roll - Program Logic

This Processor's PCE is placed lowest on the HASP Dispatcher chain and it $WAITs on ABIT when idle. This means that all Processors with their requested overlay routine in an Overlay Area will have at least one chance to execute code or otherwise use their overlay routine before the Overlay Area containing that routine is taken for other use. Overlay Roll does not receive control unless all other HASP Processors are in a $WAIT state, i.e., HASP is ready to relinquish control to OS by WAIT. Overlay Roll always receives control, just before WAIT is executed.

Overlay Roll has local addressability provided in BASE2 and also establishes the base address for Overlay Service in register WC so that its subroutines and tables may be used. On each entry, the Queue beginning with $WAITACE (see 5.16.2) of PCEs waiting for an Overlay Area is tested. If empty, $WAIT ABIT is used to exit. Otherwise, the following attempts are made to secure one or more Overlay Areas and begin reading a requested routine into them.

For each group of one or more waiting PCEs requesting the same overlay routine, all Overlay Areas are searched to find a suitable one. If a read operation to load an overlay routine is in process, the area is never taken. Users of that routine are allowed at least one chance to execute after read completion is processed by Overlay $ASYNC Exit (see 5.16.9).

For each Overlay Area which does not have read in process, the OACEPRIO field is examined and the chain of all current users (beginning at OACEPCE) is searched to determine if any user is $WAITing on I/O. This would be I/O other than an overlay read operation, would be expected to complete "soon", and would, therefore, make it less desirable to pre-empt that area. The lowest priority area with no user $WAITing I/O is chosen, if any, otherwise the lowest priority area is chosen.

Since an overlay routine is "refreshable" at $WAIT time, it is not necessary to literally "roll", i.e., write to disk, a pre-empted Overlay Area. Each PCE on the chain of current users (OACEPCE) is

processed to prevent further use of the pre-empted area by it. The re-entry address (PCER15) is "sized" to determine if it points into the Overlay Area and if so is relativized by subtracting the Overlay Area address. The PCE is forced into a $WAIT OROL state, in addition to the other $WAIT conditions present. When other $WAIT conditions have been $POSTed, the Dispatcher (see 5.1.2) detects the PCE $WAITing OROL only and sets it to call on Overlay Service. OLOD subroutine (see 5.16.8) is eventually called to refresh the routine, either directly, or if the PCE gets into the $WAITACE Queue, by OEXIT subroutine (see 5.16.7) or by Overlay Roll.

The area thus pre-empted is used to read in a new overlay routine, to be used by the highest priority PCE group on $WAITACE. The OLOD subroutine (see 5.16.8) is called to begin the read operation.

If there are more PCE groups on the $WAITACE Queue, the above actions are repeated. When Overlay Roll finally exits by $WAIT ABIT, the $WAITACE Queue is either empty or all Overlay Areas have an overlay read operation started, to be posted by Overlay $ASYNC Exit.

4.21 HASP SMB WRITER (HASPWTR)

4.21.1 HASP SMB Writer - General Description

The primary function of this program is to read System Message Blocks (SMBs) from the data set SYS1.SYSJOBQE and "print" them to HASP. The process signals the end of the OS execution phase of a job's processing and makes the messages (JCL, JCL diagnosis, allocation/disposition, SMF, etc.) available to HASP, to be later printed with print data sets of the job previously SPOOLed by HASP.

This program is used as an attached task, in the HASP region or partition, if the HASPGEN parameter &WTRPART is set to "*". Otherwise, the standard OS Output Writer is used to fulfill the same functions and is started by HASP in a separate partition, using a procedure named HOSWTR. The requeueing feature described below is only available when using HASPWTR.

The program HASPWTR depends upon OS Queue Management structures (QCR, LTH, SMB, no-work ECB) as documented in OS/360 MVT Job Management PLM. Functions such as enqueue, dequeue or delete of a job; ENQ/DEQ to control access to Queue Management resources; conversion of record addresses between NN, TTR0, and MBBCCHHR forms; and computation of sector numbers when SYS1.SYSJOBQE is on an RPS direct access device are all performed in a manner consistent with that described for the standard OS Job Management modules.

Microfiche listings for IEFQDELQ, IEFQMDQQ, and IEFSD086 were consulted as examples during the development of HASPWTR. However, no actual Job Management modules are executed by HASPWTR.

4.21.2 HASP SMB Writer - Program Logic

On initial entry after being ATTACHed, the program saves three addresses passed to it by HASP Initialization: memory address of the pseudo 1403 UCB designated by the HASPGEN parameter &WTR, address of a HASP subroutine to be called to signal end-of-job, and address of an ECB which will be posted by HASP if the operator enters the command $P HASP. After signalling HASP (via a POST) that ATTACH was successful and setting up addressability to the OS Queue Manager resident DCB and DEB for SYS1.SYSJOBQE, the program enters its major processing loop.

The major processing loop is driven by inspection of a list of ECBs. One is the $P HASP ECB which, if posted, causes the program to terminate as described below. All other ECBs are each part of an eight byte no-work element. One such element is present for each SYSOUT (MSGCLASS) to be processed, as indicated by the list of

classes assigned to the HASPGEN parameter &WTRCLAS. If an ECB is posted, the Queue Control Record (QCR) for that class is read and a job is dequeued, if present. The dequeued job's last Logical Track Header (LTH) must be read to perform the dequeue. The updated QCR is re-written. If there were no jobs to dequeue or the one dequeued was the only one, the class ECB is cleared and the no-work element is chained from the QCR before re-writing.

If a job was dequeued, its SMBs are read, messages are formatted into print lines, and the lines are "printed" to HASP using the pseudo 1403 UCB. If non-SMB blocks such as Data Set Blocks (DSBs) are encountered, they are simply skipped. The data sets they represent are not printed or scratched. When the end of the job is reached, a small subroutine in HASP is called to signal end-of-job to HASP.

The HASPGEN parameter &WCLSREQ controls the disposition of the job after processing. If the position in the list &WCLSREQ, corresponding to the position of the job's original class in the list &WTRCLAS, is a valid SYSOUT class then the job is re-queued in the QCR for that new class. Any tasks (e.g., other system writers for perhaps CRBE, CRJE, TSO, CPS, etc.) whose no-work elements are chained from that QCR are POSTed. The re-queue action always places the job in the new QCR chains at highest priority.

If &WCLSREQ does not indicate re-queuing ("*" in a list position instead of a class), the job's tracks in SYS1.SYSJOBQE are released by chaining them to the chain of free space beginning in the Master QCR, POSTing any tasks waiting for Job Queue space, and re-writing the Master QCR.

The major processing loop is repeated until no ECBs are found posted. An OS multiple WAIT is executed and when any ECB is posted by another task (usually an OS Job Terminator), the major processing loop is resumed.

If the operator enters $P HASP, HASP will POST an ECB to signal termination actions to this program. All QCRs for processed classes (&WTRCLAS) are read, the no-work chain of each is zeroed, then the QCR is re-written. HASPWTR exits with a zero completion code.

If permanent I/O errors occur during any I/O on the SYS1.SYSJOBQE data set, an error message is always written to the operator. For write operations, no further special action is taken and processing continues. For read operations, an attempt is made to minimize system damage. No input from an incorrect read is ever used for processing. If the error occurs in reading a QCR or LTH while attempting to de-queue a job, the ECB is set so that no further processing of that class will be attempted. If there is an SMB read error, end-of-job is signalled to HASP and no further blocks on that job's chain are read. If a QCR read error occurs during a re-queue attempt, the job is deleted (tracks are released).

4.22 PRIORITY AGING PROCESSOR

4.22.1 Priority Aging Processor -- General Description

The function of the Priority Aging Processor is to regularly increase the priority of a job in such a way that its position in the HASP Job Queue is enhanced with the passage of time. This is accomplished by regularly passing through the HASP Job Queue and incrementing the priority field of all Job Queue Elements whose priority falls between upper and lower limits. These limits, as well as the time interval, are HASPGEN parameters and can be specified to fit the needs of an installation.

4.22.2 Priority Aging Processor -- Program Logic

When this processor is dispatched, it searches through the HASP Job Queue until it encounters a Job Queue Element whose priority field "QUEPRIO" (see figure 8.6.1) is less than the HASPGEN parameter: &PRIHIGH. For that Job Queue Element and every Job Queue Element after that (until the HASPGEN parameter &PRILOW is reached), the priority field is incremented by one. The Interval Timer is then reset and the processor enters a HASP $WAIT until the timer interval expires.

Since the priority of the Job Queue Element is represented by the four high-order bits of "QUEPRIO", adding one to this field has no immediate effect on the priority. After repeating this operation sixteen times, however, the actual value of the priority will be increased by one. The value of the time interval is actually only one-sixteenth of the interval implied by the HASPGEN parameter: &PRIRATE. This effect tends to smooth out the process of priority aging by creating less impact when an interval expires.

In order to minimize CPU utilization, this processor discontinues operation whenever the HASP Job Queue is empty and does not continue until a new job enters the system.

4.23 SYSTEM/3 REMOTE TERMINAL PROCESSOR PROGRAM LOGIC

The HASP System/3 Remote Terminal Program is assembled on a System/360 or System/370 computer under OS, using Assembler F (IEUASM). The advantages of assembling under OS are: the System/3 program can be assembled as part of a standard HASPGEN or RMTGEN; a System/3 program can be customized to the particular System/3 configuration and HASP System being generated, since Assembler F can handle conditional assembly statements; and macros can be used.

To allow assembly of System/3 code, a set of macros is included as part of the System/3 source code, HRTPSYS3. Most of these macros are designed to generate machine language code for the System/3; a few additional macros, such as $WAIT and $FB, provide for in-line functions and control blocks. The former macros will be discussed first; they are called the machine-language macros.

The machine-language macros consist of a set of macros whose names correspond to the mnemonic System/3 operation codes defined in the publication "Card and Disk System Components Reference Manual" (Order Number GA21-9103) and the extended System/3 assembler mnemonics defined in the publication "Disk System Basic Assembler Program Reference Manual" (Order Number SC21-7509), with the following exceptions: each mnemonic operation code is prefixed by a dollar sign; no macros are provided for the instructions ZAZ, AZ, and SZ; additional extended mnemonics $NOPB and $NOPJ are provided; and the form and order of the operands is such as to be convenient to Assembler F.

When a machine-language macro refers to a location in core, the operand is coded either "address" or "(displacement,register)". Thus one might write "$MVC X'1234',(0,REG2),LENGTH" to move LENGTH bytes to core location X'1234' (and succeeding lower-addressed bytes) from the core location pointed to by REG2 (and succeeding lower-addressed bytes).

There are ten forms of machine-language outer macros. These are:

1. The two-address form exemplified by "$MVC adr1,adr2,length". The operands "adr1" and "adr2" are as explained above. The operand "length" is assembled as "length-1" unless it is omitted or is literally "*-*" (in which case it is assembled as zero) or the opcode is $MVX, in which case it is assembled as "length". The opcodes $MVC, $ALC, $ED, $ITC, $CLC, and $MVX belong to this form. The extended mnemonics $MZZ, $MZN, $MNZ, and $MNN may be used.

2. The one-address form exemplified by "$L reg,adr" and including $L, $A, $LA, and $ST.

3. The one-address form exemplified by "$MVI adr,immediate" and including $MVI, $CLI, $SBN, $SBF, $TBN, and $TBF.

4. The Jump instruction, written as either "$JC adr,cc" or "$Jxxx adr", where $Jxxx is one of the extended mnemonics. In this case, "adr" may not be specified as "(displacement,register)" and must be within a positive displacement of 256 bytes from the last byte of the Jump instruction.

5. The Branch instruction, written as either "$BC adr,cc" or "$Bxxx adr" where $Bxxx is one of the extended mnemonics.

6. The one-address I/O forms, exemplified by "$LIO da,m,n,adr" and including $LIO, $TIO, and $SNS.

7. The instruction $SIO, written as "$SIO da,m,n,cc".

8. The instruction $APL, written as "$APL da,m,n".

9. The instruction $HPL, written as "$HPL cc", where each "c" is either the actual character to be displayed as a halt code or the character "*", indicating a byte of zeros. For example, one might write "$HPL EJ".

10. The assembler instructions $DC and $DS, where the statement label (if any) is assigned the address of the last byte of the last operand specified.

In addition to the machine-language macros, a $USING and a $DROP macro are provided to enable Assembler F DSECTs to be used more easily. The form of the $USING macro is "$USING expression,register" where "expression" is a one-to-eight-character expression with the location counter reference symbol "*" either not used or used as the first character, and "register" is a one-to-eight-character absolute expression. No more than two different $USINGs (two $USINGs with different arguments "register") may be outstanding at any time. $USING works as follows: from the time the $USING is issued, for any address-type machine-language macro which contains an address specification of "(displ,reg)", the character string "reg" is compared with the string "register" of each outstanding $USING. If no match is found, the displacement is assembled as YL1(displ). If a match is found, the displacement is assembled as YL1(displ-(expression)), where "expression" is taken from the corresponding $USING.

The form of $DROP is "$DROP register" where "register" is a character string that appeared as the second operand of a previous and outstanding $USING. The form "$DROP register,register" is also allowable.

The assembly listing generated by Assembler F contains the macro-expansion for each macro used, in order to provide a printed copy of the generated text of each machine instruction and the address at which it will be loaded in System/3 storage. The expansion of each of the machine-instruction macros is typically contained in one print line, and the text of the generated instruction is

always contained in hexadecimal on one print line.

The object deck produced by Assembler F is used as input to the translation program SYS3CNVT, called automatically by RMTGEN. SYS3CNVT reads the object deck via either ddname SYSLIN, or ddname SYSGO if SYSLIN is absent. First, SYS3CNVT punches on SYSPUNCH a System/3 one-card loader. Then it reads from SYSLIN or SYSGO, ignoring all but TXT cards and the END card. For each TXT card, SYS3CNVT creates one System/3 96-column load-mode card image, suitable for reading by the System/3 one-card loader. Each such 96-column card image contains 64 bytes of information as follows:

- bytes 1-5 contain a System/3 $MVC instruction of the form "$MVC load-adr,(column-number,1),length-1" where load-adr is the absolute load address of the rightmost byte of text on the corresponding 80-column Assembler F object deck TXT card, column-number is the number minus one of the 96-card column in which appear the low-order six bits of the rightmost byte of text, the digit "1" refers to the System/3's register 1, and length is the number of bytes of text on the card;
- bytes 6-61 contain a maximum of 56 bytes of text, starting in column 6; and
- bytes 62-64 contain a three-digit card sequence number.

When the object deck's END card is detected, or when a TXT card appears that was generated by the $END macro (whose optional keyword operand START= specifies the starting execution address of a segment of text), a 96-column load-mode card image is constructed whose 64 bytes are as follows:

- bytes 1-4 contain a System/3 $B instruction of the form "$B address" where address is either the first byte of the text segment just loaded (if the $END macro does not specify START=, or if the END card of the assembly has no operand) or the address specified in the START= parameter of the $END macro or the operand field of the END card;
- bytes 62-64 contain a three-digit card sequence number.

After the object deck's END card has been processed, SYS3CNVT creates a 96-column card image of which columns 2-4 are "EOR" (this is the rep terminator card, End-of-reps) and columns 62-64 contain a three-digit card sequence number.

Certain of these 96-column card images contain descriptive information in bytes 33-64: these are the one-card loader, which is captioned "FIRST CARD"; the card created from a $END macro, which is captioned "PSEUDO-END"; and the card created from an END card, which is captioned "LAST CARD".

After it has created each 96-column card image (including that for the one-card loader), SYS3CNVT breaks the image in half and punches two 80-column cards from it. Each 80-column card punched by

SYS3CNVT contains the following fields:

- columns 1-2 are blank;
- columns 3-50 contain the first (if column 80 is odd) or the last (if column 80 is even) 48 bytes of a System/3 card image;
- columns 51-72 are blank;
- column 73 contains the punch combination for X'80', an indicator to any System/3 Remote Terminal Program generated with &S396COL=1 that two 80-column cards are to be combined and punched as one 96-column card (the System/3 Starter System is generated with &S396COL=1);
- columns 74-80 contain the remote terminal identifier and card sequence number, in the form "Rmmnnnn", where nnnn is 0001 on the first card punched.

The punched output of SYS3CNVT may be routed directly to a System/3 which is running the Starter System or other suitable System/3 Remote Terminal Program; the resulting 96-column punched deck of cards is immediately ready for loading into a System/3 of the proper configuration. Alternatively, SYS3CNVT's punched output may be punched on 80-column cards for later transmittal to a System/3. Each 80-column card is suitable for data transmission in either transparent or non-transparent mode.

The following pages constitute the Program Logic manual for the System/3 Remote Terminal program.

The program consists of processors, interrupt routines, and system subroutines. There is a processor for each logical function to be performed by the program; each processor is controlled by a Function Block (somewhat analogous to a TCB in OS). Interrupt routines are provided for those devices (BSCA, 5471, and 5475) which are capable of interrupting the CPU; other devices are operated by processors. For example, the MFCU processor operates a hopper of the 5424 MFCU; it becomes associated with either a logical reader processor or a logical punch processor, depending upon the state of the hopper.

The various routines of the System/3 Remote Terminal program are described in the order in which they appear in the listing.

IHEREP - HASP Environmental Recording and Error Processor

IHEREP prints at program load time the error statistics gathered from the previous running of the System/3 Remote Terminal program. IHEREP is then overlaid and the Remote Terminal program continues to load.

First, IHEREP loads the 5203 forms length register and selects the correct print chain image according to the printer's status information. Then it checks the log area for validity. If the log area is valid, the characters 'HASP' will appear immediately before the log area. If these characters do not appear, IHEREP prints the message

HEREP COUNTERS HAVE BEEN ALTERED

and branches to zero to cause program loading to resume.

If the log area is intact, it contains eight two-byte counters for each status byte which can contain unit check information for a device. IHEREP prints a title line and then, for each status byte, a subtitle line and as many as eight detail lines. A subtitle line contains device description and status byte number. A detail line contains status bit description, bit number, and count of bit occurrences in decimal.

Control of IHEREP resides in the table of subtitles and detail descriptors, and control of the two-byte bit counters is by a bit string (starting at symbol IHBIT1) containing one-bits for the counters to which correspond detail descriptors. The table of subtitles and detail descriptors is made up of $IHMSG macros; if the first operand of this macro is 'T', the macro defines a subtitle, and if the first operand is an integer between 0 and 7, it specifies a detail descriptor for the bit whose bit number is the first operand. The table entries are used in order, and a byte of zero defines the end of the table.

When the HASP Environmental Recording and Error Printout is complete, the counters are zeroed out and IHEREP branches to zero to continue program loading.

$COM - Commutator

The Commutator gives control in turn to the various processors which comprise the System/3 Remote Terminal program, based upon the status of the various Function Blocks (FBs).

If the Event Wait Field (FBEWF) of an FB has zeroes in the bit positions defined by EWFALL, the function is said to be dispatchable. $COM loads register one from field FBREG1 of the FB (register two points to the FB) and gives control to the associated processor by loading the Instruction Address Register (IAR) from field FBENT.

When the processor has completed its work, it returns to the commutator with register two pointing to its FB. It may return to $COMRET, where $COM will save both the Address Recall Register (ARR) as the processor's next entry point and the value of register one; $COMRETA, where $COM will save the value of register one; or $COMRETB, where $COM will assume that both the value FBENT and the value FBREG1 are correct.

Then $COM chains to the next FB (or starts again with the first FB if the chain field FBNEXT is zero) and repeats the above process.

$MFCU - 5424 MFCU Processor

$MFCU operates under two FBs and two Hopper Control Areas (HCAs) - one for each MFCU hopper. The routine contains four levels of subroutines.

$MFCU begins by calling first-level subroutine HREAD to read a card. HREAD sets up a read $SIO instruction from information in the HCA and calls second-level subroutine HEXCP. HEXCP calls fourth-level subroutine HTIO, which returns condition code equal if the hopper described by the HCA is ready and condition code unequal if it is not. If condition code unequal is returned, HEXCP returns to the commutator; it will regain control again at the call to HTIO.

If the hopper is ready, HEXCP calls third-level subroutine HSIO to perform I/O on the hopper. HSIO first checks for various exceptional conditions. If error recovery is in progress (for the other hopper), HSIO returns immediately with condition code unequal. It returns similarly if the MFCU is busy reading, printing, or punching. If error recovery is not in progress and the MFCU is not busy, HSIO tests the "hurry" switch (which is set if one hopper is active and the other hopper becomes ready with a read $SIO pending for it). If the hurry switch is set and the current $SIO is not a read-only $SIO, HSIO returns condition code false.

If all the above tests are passed, HSIO checks the stacker request associated with the current $SIO. If the stacker request is different from that for the previous $SIO, the feed path is checked to make sure it is clear. If the feed path is not clear, HSIO returns condition false; in addition, if the $SIO is read-only, it sets the "hurry" switch. But if the feed path is clear, HSIO resets the hurry switch, sets the new stacker number, and proceeds as if the stacker request for the current $SIO were the same as that for the previous $SIO.

If no stacker change is indicated, HSIO moves the current $SIO to an in-line position from the HCA and examines it. If the $SIO indicates print (interpreting), HSIO attempts to select one of two print buffers into which to move the punch information for the $SIO. If unsuccessful, HSIO returns condition code unequal. But if one of the print buffers is free (as indicated by the MFCU print-buffer-busy status bits) HSIO copies the punch data into the print buffer and modifies the $SIO instruction to indicate the print buffer being used. Then, or if the $SIO is read-only, HSIO loads the MFCU's read and punch data address registers. After a call to HTIO to insure that the hopper is still ready, HSIO issues the $SIO instruction, sets condition code equal, and returns to its caller, HEXCP.

HEXCP examines the condition code returned to it. If the condition code is unequal, HEXCP non-process exits, exactly as it did for HTIO above. But if the condition code is equal, HEXCP non-process exits to be entered again at a $TIO which continues to non-process exit until the MFCU ceases being busy; then HEXCP calls third-level subroutine HSNS to determine the completion of the I/O operation.

HSNS calls HTIO to see if a unit check condition exists. If that is the case, HSNS reads the MFCU status bytes. If all status bits in the error status byte are off (or if no unit check condition existed) HSNS returns condition code equal; if only the no-op status bit is on, HSNS returns condition code unequal.

If other error status bits are on, HSNS calls system subroutines $MSG and $LOG to add a message to the error trace table and to count the error bits for HEREP, respectively. Then HSNS checks the error bits further. If the only error bits on are punch invalid or print check, HSNS returns condition code equal; these are regarded as user data errors (punch invalid) or trivial errors (print check).

But if other error bits are on, HSNS sets the error-recovery-in-progress flag in HSIO (to prevent other $SIO instructions from resetting the error bits) and non-process exits until a SNS instruction shows that all error bits (except no-op) have been reset by the operator (who must do a non-process run-out on the MFCU). Then HSNS returns condition code unequal.

HEXCP returns to its caller (which was HREAD in this case) the condition code it received from HSNS.

HREAD examines the condition code returned to it by HEXCP. If unequal was returned, HREAD again calls HEXCP; otherwise first-level subroutine HREAD returns control to mainline $MFCU (in this case, at its second instruction).

Having read the first card from its hopper, $MFCU now tests that card for blanks, via first-level subroutine HBLANK. If the card is blank, the hopper is assumed to contain blank cards to be punched. Otherwise, the hopper is assumed to contain a job stream and the MFCU awaiting-read routine HAR attempts to associate the hopper with a free logical reader FB, using subroutine HGET. HGET returns condition code equal if it succeeds (it also posts the logical reader's FB for UNIT), and condition code unequal if the hopper becomes not ready (and therefore dormant rather than awaiting-read); otherwise, HGET non-process exits until one of the above two conditions happens.

If HGET returns condition code equal, the MFCU reading routine, HRD, signals to the now-associated logical reader that the read buffer for the associated hopper is busy; then HRD non-process

exits until the logical reader frees the read buffer. When the read buffer is free, HRD checks the EOF flag, set by the logical reader when it encounters a /*EOF control card. If the EOF flag is on, HRD makes the hopper dormant by branching to the first instruction of $MFCU; otherwise HRD calls first-level subroutine HREAD as above to read the next card and, on return, again sets the read buffer busy.

If on the other hand $MFCU finds a blank card in a dormant hopper it gives control to HAP, the awaiting-punch routine, which tries to find (via HGET) a logical punch FB of which HASP has requested permission to send a punch stream. Having found such a logical punch, HAP gives control to HPU, the MFCU punch routine.

HPU non-process exits until the associated logical punch processor sets either the EOF flag or the punch-buffer-busy flag in the flag byte of its hopper control area. If the EOF flag is set, HPU makes the hopper dormant.

But if the punch-buffer-busy flag is set, HPU punches and prints a card and reads the next card (to insure that only blank cards are punched). HPU sets up a read-punch-print $SIO and calls second-level subroutine HEXCP. If HEXCP returns condition code unequal and the MFCU status indicates any of the errors no-op, punch check, hopper check, or feed check, the punch buffer is not marked free; otherwise it is marked free and set to blanks. The MFCU status is checked again; if neither read check nor no-op is indicated, the card is examined to determine if it is completely blank. Otherwise, or if the card now in the wait station is not blank, another card is read (via subroutine HREAD). When a blank card has been read successfully, HPU again checks for punch-buffer-busy as above.

$1442 - 1442 Card Reader - Punch Processor

The $1442 processor is assembled if RMTGEN parameter &S31442 has been set to 1. Its logic is similar to that of $MFCU but simpler, since only one hopper need be controlled. $1442 uses some of the subroutines of $MFCU; for this reason, and since its interface to the logical reader and logical punch is the same, the 1442 hopper control area is similar to (but not identical with) the HCAs of the MFCU.

$1442 starts by reading a card from the 1442 via entry point GSIORD of subroutine GSIO. If the card is blank, GAP (awaiting-punch) calls HGET just as does HAP in $MFCU; if the card is non-blank, GAR (awaiting-read) calls HGET just as does HAR in $MFCU.

When a logical reader or logical punch has been associated with the 1442, GRD or GPU gains control and proceeds with I/O as indicated by the read-buffer-busy and punch-buffer-busy flags. In addition to recognizing the EOF flag set by the logical reader, GRD also recognizes the last-card status bit from the 1442 and sets the last-card flag, recognized by the logical reader.

Subroutine GSIO performs I/O on the 1442. Entry point GSIORU sets a feed command in the $SIO and branches to common code. Entry point GSIORD sets a read-EBCDIC command in the $SIO and loads the data address register; it branches to common code. Entry point GSIOPU sets up a punch-and-feed command, loads the data address register and the punch count register, and falls through to common code.

GSIO's common code non-process exits on a $TIO until the hopper is ready. Then it issues the constructed $SIO and non-process exits until the 1442 is not busy. If entry was from GSIORU, GSIO returns condition code equal; otherwise it tests for unit check (via subroutine HTIO) and reads the 1442 status bytes. If no unit check occurred, GSIO returns condition code equal.

But if the 1442 had a unit check or otherwise became not ready, GSIO uses subroutines $MSG and $LOG to add a message to the error trace table and to count the error bits for HEREP, respectively; then it checks the status bytes. If no error bit is on, GSIO returns condition code equal; otherwise GSIO returns condition code unequal.

$5203 - 5203 Printer Processor

The 5203 Printer Processor non-process exits until another processor has marked the printer data area busy. Then it completes the Q-byte and CC-byte of a $SIO instruction from an SRCB furnished it by either $PRINTER or $CONP. After a $TIO shows that the 5203 is ready, $5203 loads the printer image address register and the printer data address register and issues the $SIO. $5203 then non-process exits until the printer is not busy.

When the printer operation has ended, $5203 checks for errors. If any of the error incrementer failure check, hammer echo check, or any hammer on check has occurred, $5203 attempts to reprint the line. Otherwise it clears the print line to blanks, shows the print buffer free, and again non-process exits until a processor sets the print buffer busy.

Additionally, whenever a unit check occurs, $5203 calls subroutines $MSG and $LOG to produce an error message and to count the one-bits in the printer status bytes.

$READER - Logical Reader Processor

$READER waits for one of the physical reader routines to post it for UNIT. When posted, it sends to HASP a request-permission control sequence (via subroutine $REQ) and waits to be posted for PERM by $BSCA when the system receives from HASP the appropriate permission-granted sequence.

When it has received permission, $READER non-process exits unless the read-buffer-busy flag is on, indicating a card is ready to be processed. Then it examines the card. If the card's columns 1-5 are "/*EOF", $READER sends to HASP an end-of-file control sequence (via subroutine $LEOF), which is merely a zero-length record. It then waits again for UNIT, and continues as above when posted. The same end-of-file processing occurs if the reader is a 1442 and the last-card flag was set by the 1442 physical reader routine. 1442 code is absent unless &S31442=1.

If there is no end-of-file indication, $READER processes the card further. If object deck processing was not specified at RMTGEN time, $READER transmits the first 80 columns of the card to HASP by calling subroutine $CMPR. On return, $READER resets the read-buffer-busy flag of the appropriate hopper control area and non-process exits until the read-buffer-busy flag is again set by the physical reader routine. Then it continues as above.

However, if object deck processing was indicated at RMTGEN time by the specification &S30BJDK=1 and if the physical reader device is a 5424, $READER first checks column 81 of the 96-column card image for the character "1". If the comparison is unequal, $READER processes the card normally, as above. But if column 81 equals "1", the card is the first card of a two-card hexadecimal image of a full-EBCDIC 80-column card. In this case, $READER compresses the first 80 columns of the card into the first 40 bytes of the same device's punch buffer, shows the read buffer free, and non-process exits until the read buffer is again busy. Then it checks the new card image for a "2" in column 81. If column 81 does not contain a "2", $READER treats the newly-read card as a normal card, and the previous card is lost. If the new card contains a "2" in column 81, $READER compresses its first 80 columns to the second 40 bytes of the same device's punch buffer and transmits the constructed card image to HASP, using subroutine $CMPR. Then it resets the read-buffer-busy bit and non-process exits as above.

Subroutine RDSQUEZE performs the above-mentioned compression. It creates a single sink byte from a pair of source bytes each of which is assumed, without validity-checking, to contain the EBCDIC representation of one of the sixteen hexadecimal characters. For example, it would compress the byte pair "F0C6" to the byte "0F".

$PRINTER - Logical Printer Processor

$PRINTER waits for HASP to send a request-permission control sequence. When $BSCA finds such a sequence, it posts $PRINTER for permission. $PRINTER then checks the printer availability flag. It non-process exits until this flag becomes zero; then it sets this same flag to show that the printer is in use. It sends a permission-granted control record to HASP (via subroutine $PERM) and then, if the print buffer is free, calls subroutine $DCOM to request a print line be decompressed into the print buffer.

On return from $DCOM, $PRINTER recognizes two or three conditions: normal return, end-of-file return, and (optionally) forms mount message.

For the forms mount message case, the SRCB (carriage-control byte, in the case of print records) will be X'8E'. $PRINTER makes the carriage control byte a print-and-space-three, shows the print buffer busy, and non-process exits until the print buffer becomes free; then it sets a carriage-control byte of space-three-immediate (so that the forms mount message will be visible on the printer without operator intervention) and continues as in the normal case. This code is assembled only if &S3547l=0.

For the normal-return case, $PRINTER moves the SRCB returned by $DCOM to the printer control area as the carriage control byte, sets the print-buffer-busy bit, and non-process exits until the print-buffer-busy bit is off. Then it again calls $DCOM for the next print line.

For the end-of-file case, $PRINTER resets the printer availability flag and checks to see if HASP had again sent a request-permission. If so, $PRINTER again sets the printer availability flag, sends to HASP permission-granted (via subroutine $PERM) and continues as above. Otherwise $PRINTER waits for HASP to send request-permission.

$PUNCH - Logical Punch Processor

$PUNCH waits for HASP to send a request-permission control sequence. When $BSCA finds such a sequence, it posts $PUNCH for PERM, whereupon $PUNCH waits for UNIT. When posted for UNIT by a physical device routine, $PUNCH sends a permission-granted control record to HASP (via subroutine $PERM) and non-process exits until the appropriate punch buffer is free. Then it calls subroutine $DCOM to decompress a card image into the punch buffer.

If $DCOM returned a card image (rather than end-of-file) the image is processed in various ways, depending upon the type of the punch device and options selected at RMTGEN time. If the punch is a 1442, $PUNCH calculates the number of bytes to punch, subtracts it from 128, places the difference in the 1442 hopper control area, and shows the punch buffer busy. It then non-process exits, as above, until the punch buffer becomes free.

If the device is a 5424, $PUNCH first checks column 1 of the card image.

If column 1 is X'6A', the card image is assumed to be a HASP job separator card. $PUNCH extracts the job number from columns 52, 62 and 72, ignores the rest of the image, and punches a card of which columns 1-32 are:

```
**********  JOB  nnn  **********
```

It causes this card to be punched as usual, that is, by marking the punch buffer busy; then it non-process exits until the punch buffer becomes free.

If the device is a 5424 and RMTGEN specified &S396COL=1, $PUNCH checks column 73 of the card image. If that column is X'80', $PUNCH checks column 80. If column 80 is odd, $PUNCH saves in a work area in its Function Block the 48 columns starting at column 3 and again calls $DCOM to get the next card, as above. If column 80 is even, $PUNCH moves columns 3-50 of the card image to columns 49-96, moves the first 48 bytes from its work area to columns 1-48, and causes the card to be punched.

If the device is a 5424 and RMTGEN specified &S3OBJDK=1, $PUNCH checks column 1. If that column is X'02', $PUNCH saves the rightmost 40 columns of the 80-column card image in its work area and expands the leftmost 40 columns to 80 columns by substituting for each byte two EBCDIC characters; for example X'02' becomes C'02'. It sets the character "1" in column 81 and causes the card to be punched. $PUNCH then repeats this process for the saved 40 columns, sets the character "2" in column 81, and causes the card

to be punched.

If none of the above situations apply, $PUNCH merely marks the punch buffer busy, non-process exits until it becomes free again, and then calls $DCOM to get the next card.

$DCOM may return an end-of-file indication rather than a card image. $PUNCH sets the end-of-file flag in the hopper control area and checks for a subsequent request-permission from HASP. If HASP has requested permission again, $PUNCH waits again for UNIT, as above; otherwise $PUNCH waits for PERM, as above.

5471 Console Interrupt Routine

CINT, the 5471 console interrupt routine, gains control upon an interrupt from either the 5471 printer or the 5471 keyboard. A keyboard interrupt may occur due to the End key, the Return key, the Cancel key, the Request key, or a Data key. A printer interrupt may occur either after completion of printing a character or after a carriage return.

At an End key interrupt CINT starts a carriage return, posts the console processor, and exits by starting the keyboard. If a request is pending, the Start-I/O instruction sets the request light on and disables interrupts from all keys; otherwise it sets both lights off and enables interrupts from the request key.

A Return key interrupt causes the same functions as an End key interrupt.

A Cancel key interrupt causes CINT to print an asterisk and set a flag which will cause a carriage return at the next printer interrupt. CINT then resets the buffer pointer to point to the first byte of the buffer and exits by issuing a SIO which leaves the same lights on and interrupts enabled as before the interrupt.

At a Request key interrupt, CINT posts the console processor if an inspection of the console status byte CCFLG shows that neither input nor output is currently in process. In any case, it sets the request-pending bit and exits by issuing a SIO which turns on the request-pending light, disables request key interrupts, and leaves the proceed light and other key interrupt indicators as they were before the interrupt.

For a Data key interrupt, CINT saves the keyed character in the buffer byte pointed to by the buffer pointer; then it increments the buffer pointer by one. It issues a SIO to the printer so that the keyed character will be printed. If the buffer pointer now falls outside the buffer, CINT turns on the carriage-return request bit and performs all the functions of the End key except for issuing a carriage return. Otherwise it exits by issuing to the keyboard a SIO which leaves the same lights on and interrupts enabled as before the interrupt.

On a printer interrupt due to end of either printing or carriage return, CINT tests the carriage return request bit. If that bit is on, CINT resets it and exits by issuing a SIO for carriage return.

If there is no carriage return request pending, CINT tests the output-in-process bit. If output is not in process, CINT exits by disabling printer interrupts. But if output is in process, CINT checks whether the final output character has been printed. If so it resets the output-in-process flag, posts the console processor, and exits by starting a carriage return. If not, it selects and

loads the next character to print and exits by issuing a SIO to print that character.

Whenever CINT posts the console processor, it also turns on the action-required flag, CFACT. This flag is tested and reset by the console processor.

5471 Console Processor

The 5471 console processor, $CON, non-process exits until posted; then it checks to find what caused it to be posted.

If input is complete, $CON replaces in the MULTI-LEAVING buffer pool the buffer it stole when it acknowledged the request key. Then it sends the operator command to HASP by calling subroutine $CMPR, unless the input length is zero. In any case, it continues by checking for request-pending.

If a keyboard request is pending, $CON first steals a buffer from the MULTI-LEAVING buffer pool, to avoid a potential buffer lock-out problem. If no buffers are available, it leaves the request pending and checks for queued buffers containing messages to print on the 5471 printer. But if the MULTI-LEAVING buffer steal was successful $CON resets the 5471 buffer pointer, resets the action-required and request-pending flags, sets the input-in-process flag, and issues a SIO which turns on the proceed light and enables all keyboard interrupts. Then it non-process exits until posted.

If $CON was not posted for the above reasons, it investigates output possibilities. If either input or output is in process, it cannot start output; it again non-process exits until posted. But if neither input nor output is in process, and if there is no end-of-forms indication from the 5471, $CON checks for output. First it checks the error message table, a circular table, to see if any error messages are outstanding. If so, it expands a four-byte coded error message to the equivalent eight-character hexadecimal representation in the 5471 buffer, sets the output-in-process flag, and issues a SIO to start printing the first character; then it non-process exits until posted, while CINT prints the remaining characters.

If no error messages are outstanding, $CON checks for messages from HASP. If there are some, $CON calls subroutine $DCOM to decompress a message. In order not to be forced into a wait condition on subsequent calls to $DCOM, $CON then checks whether the MULTI-LEAVING buffer from which the message was decompressed contains more messages; if not, $CON frees it by calling subroutine $FREEBUF. Then $CON initiates printing of the message by setting the output-in-process flag and issuing a SIO to print the message's first character. Then $CON non-process exits until posted.

5475 Console Interrupt Routine

Upon an interrupt from the 5475 Data Entry Keyboard, the 5475 Console Interrupt Routine (CINT) checks the cause of the interrupt. An interrupt may be caused by a data key, the field-erase function key, the release function key, the error-reset function key, any other function key or switch, or the multipunch key. A multipunch key interrupt is treated as an error and requires the operator to depress the error-reset key; all function keys and switches other than those mentioned are treated as no-operation keys.

A data key interrupt causes CINT to place the keyed character in the 5475 buffer. CINT then increments the buffer pointer by one; if the buffer pointer now points outside the buffer, CINT performs the release key function. Otherwise CINT adds one to the column indicated and exits. The exit process consists of issuing a LIO for the column indicators and a SIO for the keyboard.

An interrupt from the field-erase key causes CINT to reset the buffer pointer, set the column indicators to display "01", and exit.

An interrupt from the release key causes CINT to post the 5475 console processor for work, set the SIO in CINT to disable the keyboard, and exit.

Any of several error situations causes CINT to turn on the error light. It does this by setting its SIO to X'23', which also locks all data keys. When an interrupt other than from the error-reset key occurs and the error light is on, CINT exits without further processing. But if the interrupt was from the error-reset key, CINT resets the SIO to its normal value of X'4F' and exits. Conditions which cause the error light to come on are a multipunch interrupt indication, no interrupt indication, or two or more of the interrupt conditions data key, function key, and multipunch key.

5475 Input Console Processor

When posted for WORK by CINT, the 5475 Input Console Processor ($CON) sends the operator command to HASP by calling subroutine $CMPR, unless the input length is zero. In any case, it resets the column indicator save area to "01", resets the 5475 buffer pointer, and sets to X'4F' the SIO in CINT. Then $CON turns off the column indicator display (to avoid burning out the lights), issues a SIO to unlock the keyboard and enable interrupts, and again waits for WORK.

$CONP - 5203 Output Console Processor

When posted for WORK, $CONP checks the printer-availability flag. This flag is on if $PRINTER is currently printing a job. If the flag is on and RMTGEN specified &PRTCONS=2, $CONP frees all MULTI-LEAVING buffers currently queued on its Function Block (using subroutine $FREEBUF) and again waits for WORK. But if RMTGEN specified &PRTCONS=1, $CONP checks to see if it should force messages to be printed on the 5203. It does this by comparing the number of MULTI-LEAVING buffers currently queued on its Function Block with a maximum number. If the comparison is low, it non-process exits until either the comparison is not low or the printer-availability flag is off; if the comparison is not low, it performs a page eject before starting to print messages.

To print messages, $CONP first prevents the logical printer routine $PRINTER from using the 5203 simultaneously; to prevent this, it sets the UNIT wait bit in $PRINTER's Function Block. Then $CONP attempts to find an outstanding four-byte coded error message; if it finds one, it expands the message to eight bytes and causes it to be printed.

If no error messages are outstanding, $CONP checks for messages from HASP. If there are some, it calls $DCOM to decompress a message. In order not to be forced into a wait condition on subsequent calls to $DCOM, $CONP then checks whether the MULTI-LEAVING buffer from which the message was decompressed contains more messages; if not, it calls $FREEBUF to free the buffer. Then $CONP causes the message to be printed, by marking the print buffer busy and non-process exiting until it again becomes free. All messages printed by $CONP are single-spaced.

Finally, if no messages remain to be printed, $CONP examines the printer-available bit to determine if it interrupted a job to print messages. If so, $CONP does a page eject. In any case, $CONP resets the UNIT wait bit to unlock $PRINTER and waits for work again.

BSCINT - BSCA Interrupt Routine

The BSCA Interrupt Routine, BSCINT, processes all interrupts and performs all error recovery for the Binary Synchronous Communications Adapter. Processing is always initiated by one of three types of op-end interrupts - end-of-transmit, end-of-receive, and 2-second-timeout.

For an end-of-transmit interrupt, BSCINT gains control at BSXOPE. If no hardware errors have occurred, it starts a receive operation; otherwise it uses subroutine BIDISCON to recover from a possible disconnect and, on return, attempts to re-transmit.

For an end-of-receive interrupt, a great deal more is done. After having computed the number of received bytes, BIRCV checks for hardware errors; if any occurred, it uses subroutine BIDISCON and then transmits a negative acknowledgment (NAK) to HASP.

If no hardware error occurred, the starting sequence is checked at BCROK - it is valid if it is a NAK or a DLE-ACK0 or if its second byte is STX and the last byte received is ETB. If none of these is the case, BCROK sets up an error message of 03SSSS00 (where SSSS is the starting sequence) and then transmits a NAK to HASP.

The section of code responsible for transmitting a NAK first checks whether the wait-a-bit (WAB) sequence had been transmitted most recently; if so, it transmits the WAB sequence again rather than a NAK. If not, it determines if more than five bytes had been received. Since the buffer used for a receive is the same as that used for a transmit, the receive operation may have overlaid some or all of the transmitted data; since the starting or ending sequence was incorrect or a hardware error occurred, BSCINT has not yet received a positive acknowledgment for the transmitted data. To alleviate this problem, the first five bytes of the transmit data were saved before the buffer was transmitted. If the receive operation overlaid more than these bytes, the buffer cannot again be transmitted; the first two saved bytes are replaced with a DLE-ACKO and the transmit ending address is set to the starting address plus two. Then the routine transmits a NAK to HASP.

If the received starting sequence was a NAK, the interrupt routine sets up an error message of 02000000 (NAK received), refreshes the first five bytes of the buffer and the transmit ending address, and re-transmits the buffer to HASP.

If the received sequence was DLE-ACK0, BSCINT sets flags to show $BSCA that a transmit-receive operation has completed; then it exits by starting a two-second timeout. If the two-second timeout completes before $BSCA has cancelled it, BSCINT sets the two-second-timeout-complete flag and exits by disabling BSCA interrupts.

If the second byte of the received starting sequence was STX and the ending byte was ETB, BSCINT validates the Block Control Byte (a HASP control byte which contains a modulo-16 received-block count) and saves the two-byte HASP Function Control Sequence. If the BCB is as expected, interrupt processing concludes as for DLE-ACK0. Otherwise the STX is changed to X'FF' as a signal to $BSCA to throw the buffer away and the difference between the received BCB and the expected BCB is examined. If the modulo-16 difference is -2 or -1, BSCINT tolerates the error; otherwise it sets up an error message of 02rree00 to display the received and expected BCB's, and it builds and transmits to HASP a BCB-error control sequence.

$BSCA - Communications Adapter Processor

$BSCA non-process exits until BSCINT posts it with an indication that either an error message awaits synchronous processing, a receive operation has completed without error, or a two-second timeout has occurred.

If an error message was produced by BSCINT, it must be placed in the circular error message trace table by a synchronous processor rather than an interrupt routine, since the $MSG subroutine is not re-entrant. $BSCA calls the $MSG subroutine to add the error message to the trace table.

If a receive operation has ended without error, $BSCA processes the received buffer, which is always the first buffer on $BSCA's buffer chain. If the buffer does not contain text, $BSCA frees it immediately. Otherwise $BSCA inspects the buffer's first RCB (or first SRCB if the RCB indicates a MULTI-LEAVING control record). If the RCB is zero (typical when HASP sends wait-a-bit) $BSCA frees the buffer. Otherwise $BSCA compares the RCB (or SRCB) with the field FBRCB in all FB's eligible to receive buffers; if there is no match, it frees the buffer. But if a match is found, $BSCA again determines if the first record in the buffer is a control record. If so, it posts the subject FB for PERM and resets its POST bit to indicate a possible early post (the POST bit is turned on by subroutine $PERM); then it frees the buffer. But if the buffer contains data records, $BSCA dequeues the buffer from its own FB and queues it onto the subject FB, in the process reducing its own buffer count by one, increasing that of the subject FB by one, and, if the subject FB's buffer count (FBBCT) becomes equal to or greater than the subject FB's maximum buffer count (FBBMX), resetting the appropriate bit in the master Function Control Sequence $FCS by using FBFCS.

If $BSCA turned off an FCS bit, it turns on flag BFCSOFF. Whether or not $BSCA turned off an FCS bit, it inspects the subject FB's flags; if flag BFCSON is on in FBFLG, $BSCA resets that flag and its own BFCSON flag. Flag BFCSON expedites transmission of a response and flag BFCSOFF delays transmission. The effect of the above manipulation is to avoid an unnecessary line turnaround when a printer or punch is temporarily at its buffer limit.

Having processed the received buffer, or if a two-second timeout occurred, $BSCA determines what and when it is to transmit. It transmits a response immediately under any of the following conditions:

- Wait-a-bit was received from HASP
- A text buffer (see Figure 4.23-1) is ready to send

- Flag BFCSON is set and there is a free buffer
- Text was received from HASP, flag BFCSOFF is not set, and there is a free buffer
- Two seconds have passed since end-of-receive.

The response transmitted is one of the following:

- Text, if a text buffer is ready to send
- A Function Control Sequence, if there is a free buffer and the FCS has changed
- DLE-ACKO, if there is a free buffer and the FCS has not changed from when it was last transmitted
- Wait-a-bit (including FCS) if there are no free buffers.

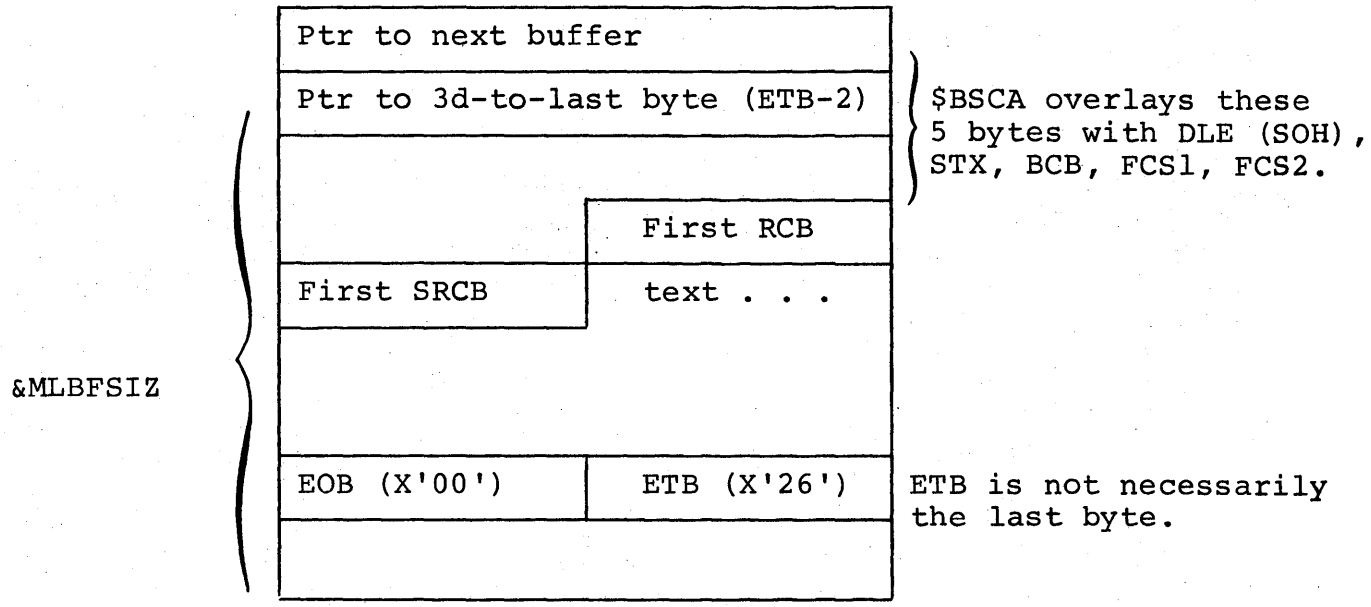


Figure 4.23-1 Multi-Leaving Buffer

To get a free buffer $BSCA uses subroutine BSGBUF, which queues the buffer on $BSCA's buffer chain (FBBUF) in last-in, first-out fashion and increments its buffer count (FBBCT) by one. Additionally, a part of BSGBUF sets up the transmit starting address, receive starting address, and receive ending address, and may be called separately from BSGBUF.

$CMDSCAN - Local Command Subroutine

If RMTGEN parameter &S3CMDS is set to 1, code is assembled to provide a local command facility. Code appears in four places:

- $CMDSCAN, to process the commands
- $CVB, used by $CMDSCAN to convert decimal command operands to binary
- $MFCU, to allow $CMDSCAN to check non-blank cards from dormant hoppers ($CMDSCAN returns condition code equal if a card contained a command; the hopper remains dormant)
- $1442, with the same functions as $MFCU

$CMDSCAN receives a pointer to a card in index register 1. It examines the card for a valid command and branches to the proper command routine, or returns to its caller with condition code not-equal.

Each command routine processes the command's operands as necessary, and exits to one of three labels:

- CMDEND (normal end) to print 'CODE0000'
- CMDSYN (syntax error) to print 'CODE0001'
- CMDOPD (operand error) to print 'CODE0002'.

A command routine may use the $CVB subroutine to convert an operand from decimal to binary. Index register 1 must point to the decimal operand's high-order byte. If this byte is not numeric, $CVB will branch to CMDSYN; otherwise, on return from $CVB, the binary result will be right-justified in bytes $CVBANS and $CVBANS-1, and index register 1 will point one byte past the low-order digit of the decimal operand.

(The remainder of this page intentionally left blank.)

$LEOF, $PERM, $REQ - Control Sequence Subroutines

These subroutines transmit to HASP certain control sequences required for proper operation of HASP MULTI-LEAVING Remote Job Entry: logical end-of-file, permission-granted, and request-permission.

$LEOF sends the sequence RCB, SRCB, SCB where RCB is taken from the FB pointed to by register 2 (FBRCB), SRCB is X'80', and SCB is X'00' (a string control byte of X'00' is an end-of-logical-record SCB; occurring immediately after an SRCB, such an SCB indicates a zero-length record).

$PERM sends the sequence RCB, SRCB, EOB where RCB is X'A0' (permission-granted for function described in SRCB), SRCB is taken from FBRCB of the FB pointed to by register 2, and EOB is X'00' (a zero RCB indicating logical-end-of-transmission-block). $PERM also sets the bit EWFPOST in the field FBEWF; this "early-post" bit is reset by $BSCA when it finds any permission-type control record whose SRCB matches FBRCB.

$REQ sends the sequence RCB, SRCB, EOB where RCB is X'90' (request-permission for function described in SRCB) and SRCB and EOB are as described for $PERM.

Code common to all three routines requests from $CKLEN three bytes of space in a MULTI-LEAVING buffer, moves the three-byte sequence, and calls $BFLUSH to truncate the buffer and queue it on $BSCA's buffer chain.

$DCOM - Decompression Subroutine

$DCOM is called by one of the output processors (such as $PRINTER) to decompress a logical record from a MULTI-LEAVING buffer into an area whose starting address is supplied by the caller. (HASP transmits all data records to MULTI-LEAVING terminals in a compressed and truncated format). If decompression is successful, $DCOM returns to the caller at an offset of three bytes; if $DCOM recognized a logical-end-of-file, it returns at an offset of zero.

To decompress a logical record, $DCOM first examines the address in FBCURL, a two-byte field in the caller's FB reserved for the use of $DCOM. If that field is non-zero, it has previously been set by $DCOM to point to the RCB following the last-decompressed logical record in the current buffer. If that RCB is not X'00', $DCOM decompresses to the caller's area (which must be two bytes longer than the maximum record length) the record following the RCB, moves the SRCB to FBSRCB, saves the address of the next RCB in FBCURL, and returns to the caller as explained above.

But if FBCURL is zero, $DCOM checks if more buffers are queued on the caller's FB. (If FBCURL is non-zero but the RCB to which it points is zero, $DCOM first frees the current buffer and then proceeds as if FBCURL were zero.)

If one or more buffers are queued, $DCOM selects the first buffer, points to its first RCB, and decompresses a logical record as above. But if no buffers are queued, $DCOM waits for WORK, to be posted by $BSCA when the next buffer for the same output device is received.

The output buffer's address is specified by the caller in field FBAREA; on return, $DCOM replaces this field by the address of the last-plus-one output byte.

$CMPR - Compression Subroutine

$CMPR compresses data from a user-specified input area to a local workarea and transmits it to HASP by calling subroutine $CKLEN.

When called, $CMPR examines the status of its local workarea. If the workarea is busy, $CMPR has been called by some other processor and has in turn called $CKLEN; $CKLEN is non-process exiting until it can find sufficient bytes in a MULTI-LEAVING buffer to allocate to $CMPR. In this case, $CMPR non-process exits until its workarea becomes free.

When the workarea is free, $CMPR compresses into it the text pointed to by FBAREA. Compression consists of either full compression and truncation, only truncation, or neither compression nor truncation, as selected by the setting of the RMTGEN variable &COMP=. Once the record is compressed, $CMPR calculates its compressed length and calls $CKLEN with a request for the number of bytes it requires in a MULTI-LEAVING buffer. When $CKLEN returns, $CMPR moves the compressed record, shows its workarea free, and returns to the caller.

$CKLEN - MULTI-LEAVING Buffer Allocation Subroutine

$CKLEN returns to its caller the address in a MULTI-LEAVING buffer of the rightmost byte of an area whose length is specified by the caller.

The caller specifies a length in register one. If $CKLEN has a current buffer, its current buffer pointer points to the last-allocated byte. It adds to this the caller's specified length. If the resultant address is lower than two bytes before the end of the buffer, $CKLEN saves this address as its current buffer pointer and returns this address to the caller in register one. But if the resultant buffer address is not lower than two bytes before the end of the current buffer, $CKLEN truncates the buffer, queues it on $BSCA's buffer chain, and posts $BSCA by turning on flag BFPOST in byte BCF1. To truncate a buffer, $CKLEN moves the current buffer pointer to its first two bytes and the sequence EOB, ETB (X'0026') to the two bytes after the byte pointed to by the current buffer pointer.

After having truncated and queued the current buffer or if on entry there was no current buffer, but not if entered via entry point $BFLUSH (in which case $CKLEN returns immediately after truncation and queuing), $CKLEN attempts to get another buffer to satisfy the caller's request; if no buffer is free, it non-process exits until one comes free. It initializes the current buffer pointer to point to what will eventually be the buffer's FCS2 byte. It initializes a pointer to the last byte available in the buffer, and it saves the address of the buffer's chain word in a third pointer. Then it allocates space for the caller and returns, as above.

$FREEBUF - MULTI-LEAVING Buffer Free Subroutine

$FREEBUF dequeues the first buffer from the buffer chain word FBBUF of the FB addressed by register two upon entry; subtracts one from FBBCT, the count of buffers enqueued upon that FB; and compares the new count with FBBMX. If the compare is low, $FREEBUF OR's the two-byte field FBFCS into the two-byte field $FCS, posts the $BSCA processor, and sets flag BFCSON in both the subject FB's flags and the BSCA flag byte. (See the $BSCA processor description for a discussion of BFCSON.)

In any case, $FREEBUF queues the just-dequeued buffer on chain word $MLPOOL in last-in, first-out sequence. If the system was generated for a 5471 console, $FREEBUF posts $CON, the console processor. Then $FREEBUF returns to its caller.

ABEND - Core Dump Subroutine

ABEND produces a core dump on the 5203 printer. The code for ABEND is assembled only if the RMTGEN specification &DEBUG=1 has been used. &DEBUG=1 also causes the generation of extra debugging code throughout the terminal program; some of the extra sequences of code generated contain conditional branches to ABEND. ABEND may also be called from the CE panel of the System/3 by setting the IAR to its address.

Each line produced by ABEND consists of a four-character address, 64 characters representing the 32 bytes starting at that address, and their printable equivalent in 32 more characters, bounded at the left and the right by a single asterisk; or four asterisks in the address position followed by blanks, to indicate that all of core up to the next line's address or the end of core would have printed the same as the previous line. The ABEND dump routine requires a printer with at least 120 print positions; if a 96-print-position printer is used, not all of the EBCDIC portion of the line will be printed.

The first six bytes of printed core contain the address recall register, register one, and register two as of the time ABEND gained control; the remainder of core is intact.

$LOG - HASP Error Recording Subroutine

$LOG is a re-entrant subroutine which maintains in-core error recording counters. Each counter is two bytes long and has a maximum count of 65535. There are eight counters for each of the following bytes:

```
1442 Status Byte 2  (if &S31442=1)
1442 Status Byte 1  (if &S31442=1)
BSCA Status Byte 2
5203 Status Byte 2
5203 Status Byte 1
5424 Status Byte 1
```

The counters are captioned, printed, and reset by IHEREP at program load time and thus form a permanent record of unit checks associated with the above devices. Only those counters which represent unusual unit checks are printed by IHEREP.

$MSG - Error Message Tracing Subroutine

$MSG adds the four-byte coded entry addressed by register one to the circular trace table of error messages. This table is examined by the 5471 console processor and under certain conditions by the 5203 output console processor; $MSG posts whichever of these processors has been generated.

The various error messages supplied to this routine by its callers are explained in the System/3 Operator's Guide.

$INIT - Initialization Routine

$INIT gains control when program loading is complete. It sets the print chain image, reads and processes REP cards, sets the 5203 forms length register, sets the 5424 print buffer register, establishes communication, sets up buffers, and exits to the commutator.

To set up the print chain image, $INIT reads the printer status bytes. If the 48-character-set bit is on, it moves the LC image to the image area; otherwise it moves the PN image. Then $INIT starts processing reps.

The format of a REP card is

```
column    2    9    17
          REP  addr rep-data
```

where "addr" must be a valid hexadecimal core address of exactly four characters (or four blanks) and "rep-data" is a sequence of one or more replacement groups with the last group terminated by a blank and all others terminated by commas. A replacement group is a string of 2n (n any integer) hexadecimal characters. The blank after the last replacement group may be followed by comments.

Starting at the address specified by "addr", the REP routine will store bytes one at a time corresponding to byte pairs of the "rep-data" taken from left to right. If the "addr" specification is blank, bytes will be stored starting at the first byte after the byte last used by the preceding REP card (or at zero if there was no preceding REP card). A REP card whose "rep-data" field contains no data is valid; its "addr" field (if any) specifies the address of the first byte to be repped if the next REP card's "addr" field is blank.

To process reps, $INIT reads a card from the primary hopper of the MFCU; a read error will give an F3 halt. If the card image contains "REP" in columns 2-4, it is processed according to the above specifications, with absolutely no validity-checking, and $INIT reads another card, as above.

If the card image contains "&MLBFSIZ=" starting in column 1, $INIT converts to binary the specified decimal buffer size (which must immediately follow the equal sign and be terminated by a blank) and substitutes the result for the default buffer size. Then $INIT reads the next card, as above.

If the card image contains "/*SIGNON" starting in column 1, $INIT overlays the default sign-on card with it and continues as if the card were an EOR card.

If the card image contains "EOR" (end-of-reps) in columns 2-4, $INIT terminates rep processing, loads the 5203 print forms length register and the 5424 print buffer address register, and establishes communications.

To establish communications, $INIT first disables and then enables the BSCA. Next, it examines the sign-on card to see if dialing information was specified. If so, it determines the starting and ending addresses for the telephone number (which is not checked for validity) and loads these values into the current and stop address registers after first ensuring that the data line is unoccupied. (If the data line is occupied, $INIT assumes the operator dialed and waits for the data set to become ready.) After starting an auto-call operation and looping until an op-end interrupt occurs, $INIT checks for timeout status; if so, the auto-call unit returned an abandon-call-and-retry signal and a CA halt (call-aborted) occurs. When the operator resets the halt, the entire logic starting with disable-BSCA will be re-executed. But if the timeout bit is off, $INIT assumes the call was successful and loops until a dataset-ready indication occurs, as above.

When the data set becomes ready, $INIT transmits the two-byte sequence SOH-ENQ, a sequence recognized by HASP as a request from a MULTI-LEAVING terminal. If the receive part of this transmit/receive command ends with timeout, the operation is repeated; if it ends with any other abnormal status, one of two things occurs. If the system was generated with &DEBUG=1 and the address knobs on the System/3 console are set to any odd address, the System/3 halts; the halt indicators display a hexadecimal image of the BSCA error status byte. Otherwise, and when the operator resets the halt, the entire logic starting with disable-BSCA will be re-executed.

If the receive operation ended normally, the two received bytes should be DLE-ACK0. If they are not, the transmit/receive operation is performed.

If DLE-ACK0 was received correctly, the message "COMMUNICATION ESTABLISHED" is printed on the 5203. If a 5471 was specified when the system was generated, its interrupts are enabled and the same message is printed on it. If a 5475 was specified, its interrupts are enabled. $INIT now performs buffer initialization.

Buffer initialization consists of three steps and overlays the initialization code with MULTI-LEAVING buffers. As the first step, the value of MULTI-LEAVING buffer size is set in the various locations throughout the program that requires it; it may have been changed by the &MLBFSIZ control card. Step two moves the actual buffer initialization code to low core, where it is executed

as step three. Execution consists of chaining together all buffers but the first buffer (which contains the sign-on record and is afterward queued to the $BSCA processor) with the chain origin at $MLPOOL.

When buffer chaining is complete, the sign-on buffer is queued as mentioned and control passes to the commutator. $COM gives control in its turn to the $BSCA processor, which as a special, first-time function transmits to HASP the buffer containing the sign-on card image.

(The remainder of this page intentionally left blank.)

5.0 HASP CONTROL SERVICE PROGRAMS

This section contains detailed internal information about each of the HASP Control Service Programs and is intended primarily for use by systems programmers.

5.1 HASP DISPATCHER

5.1.1 HASP Dispatcher - General Information

The HASP Dispatcher is responsible for the allocation of the CPU time used by the HASP Task to each of the HASP Processors.

5.1.2 HASP Dispatcher - Program Logic

The HASP Dispatcher receives control each time the HASP task is dispatched by the Operating System and utilizes an ordered queue of Processor Control Elements (PCE's) to distribute the CPU time among the HASP Processors. The Event Wait Field (EWF) in each PCE (see Figure 8.2.1) is a two byte field which is used to control the dispatchability of the Processors. Any Processor or Control Service Routine may issue a $WAIT macro-instruction at any time which turns on a particular bit in the EWF corresponding to the event $WAITed on and returns control to the HASP Dispatcher to allow other processors to be dispatched. The Processor (or Service Routine) will not be given control again until some other system function issues a $POST to its EWF for the event $WAITed on.

The events reflected by the EWF fall into two categories: the first of which is the synchronization of the use of common system resources such as buffers, direct-access space, etc. With the exception of the general $POST bit $EWFPOST, the bits in the first byte of the EWF field are used to indicate the particular resource being $WAITed on and corresponds exactly to the Event Completion Field (ECF) in the Dispatcher. The ECF is $POSTed whenever a resource becomes available and is propagated through all processor EWF's by the Dispatcher. Thus, if a track becomes available on a direct-access device, every processor (PCE) which has issued a $WAIT for a track will be put in contention for CPU time to try to obtain the track or tracks that have been released.

The second byte of the EWF is used to synchronize a processor with respect to a specific event, applicable only to that processor, such as a particular I/O completion. This section of the EWF must be $POSTed directly by the system routine performing the required function (additional details regarding $WAIT/$POST events may be found in Section 9.8).

When scanning the PCE chain, the HASP Dispatcher, upon discovering a zero EWF (no events being $WAITed on), will enter the code controlled by the PCE immediately below the prior $WAIT which had returned control to the Dispatcher. All registers of a processor which issues a $WAIT are preserved in the PCE and are reloaded prior to entering the processor (register "R15" is destroyed by the $WAIT macro to provide the address of the $WAIT, i.e., the resume point). A processor may return control to the Dispatcher

only by means of the $WAIT macro. In the event any $POST macro was executed by the processor dispatched or by any of the HASP asynchronous service routines the Dispatcher's ECF field will be altered to reflect the $POST. The general $POST bit represents a $POST of a specific processor (second byte of the EWF). If the ECF field indicates no $POST has occurred, the HASP Dispatcher continues to scan down the PCE chain starting with the next PCE. However, if the ECF field indicates $POSTs have occurred, the $POST for the general $POST is removed and scanning is resumed at the beginning of the PCE chain, after promulgating any remaining ECF $POST indicators.

Upon reaching the end of the PCE chain, the Dispatcher examines the processor active count to determine if any jobs are being processed. If an active job is in the system (active count $\neq$ 0) an OS WAIT state is entered to wait for some external event (I/O interrupt, etc.) to activate HASP again. This WAIT allows use of the CPU by other tasks in the system. If no jobs are active, the message

"ALL AVAILABLE FUNCTIONS COMPLETE"

is sent to all operator consoles and HASP is placed into the WAIT state.

When scanning the PCE chain, the Dispatcher detects the special case of a PCE which is not dispatchable (PCEEWF is not zero) but is $WAITing only on the OROL bit. This situation is created when, while the PCE was $WAITing on other event(s), the Overlay Area being used by the PCE is preempted by the Overlay Roll Processor for other use (see Section 4.20). Subsequently, the other event(s) being $WAITed on are $POSTed allowing the Dispatcher to detect the "OROL only". The Processor in such a condition is not entered but is made to call Overlay Service. The actions performed are identical to those described for $LINK Service in Section 5.16.2, beginning with the fourth paragraph describing search of Overlay Areas. The Processor will be entered by Overlay Service if the requested routine is in memory, or will be $WAITed on OLAY allowing the Dispatcher to continue its PCE scan.

5.2 INPUT/OUTPUT SUPERVISOR

5.2.1 Input/Output Supervisor - General Description

The HASP Input/Output Supervisor ($EXCP) is used to interface all HASP Input/Output requests with the Operating System Input/Output Supervisor. Through the use of $EXCP the HASP processors can remain "device independent" through the wide range and number of HASP direct-access devices. In addition, $EXCP also provides all required I/O appendages for OS IOS and for the $POSTing of I/O completions to each processor.

5.2.2 Input/Output Supervisor - Program Logic

The only interface between the HASP Input/Output Supervisor and the using processors is the Device Control Table element (DCT), which is passed via the $EXCP macro-instruction when I/O is requested. (Additional information concerning the DCT and the $EXCP macro may be found in Sections 8.5 and 9.5 respectively.) Upon entry to $EXCP the address of the buffer to be used is obtained from the DCT and the IOB (appended to every buffer) is initialized. The user's Event Wait Field (EWF) address is moved from the DCT to the buffer and a pointer to the DCT is placed in the buffer. If the DCT is a direct-access type, the coded track address from the DCT is used to compute MBBCCHHR.

The IOB is now scheduled for I/O through the use of the standard OS Execute Channel Program macro-instruction (EXCP) and immediate return is made to the caller. Each I/O request issued by HASP has an I/O appendage list specified which causes the appropriate channel end appendage in $EXCP to be entered upon termination of the I/O. Since these appendages are entered asynchronously with HASP operation, the buffer associated with the completed I/O is scheduled for synchronous HASP processing by the Asynchronous Input/Output Processor. The HASP task is POSTed, and immediate return is made to IOS. (The action taken by the Asynchronous Input/Output Processor is explained in Section 4.8.)

A separate channel end appendage is provided for remote terminal operations. This appendage correlates the channel end conditions with the commands executed and provides special processing of conditions unique to the teleprocessing.

If HASPGEN parameter &RPS was set to YES, additional code is included to support rotational position sensing.

RPS support causes each HASP EXCP to be analyzed at Start-I/O time. If the EXCP was to a direct-access device with the RPS feature, a SIO appendage will insert into the channel program a set-sector command. The sector number supplied with this command will be extracted from a table on the basis of record number, extent number, and the channel program's data length (in IOBCCW3).

Assembled into the SIO appendage are CCW sets, one for each extent of the HASP direct-access DEB. (The last extent is always for the overlay library.) Each CCW set consists of a set-sector and a transfer-in-channel. The set-sector data address points to the unused byte (byte 5) of the set-sector CCW; this byte is filled in at SIO-appendage time. The transfer-in-channel data address is filled in at SIO-appendage time, as follows: starting at the address specified by the channel address word (CAW), CCWs are inspected until a TIC is found. These CCWs constitute the direct-access start-data-transfer channel program, and the TIC is to the HASP channel program. The SIO appendage makes the start-data-transfer TIC point to the appropriate set-sector command and the set-sector TIC point to the HASP channel program.

The last four bytes of the set-sector TIC, unused by the channel, contain a pointer to set-sector values for the corresponding extent, indexable by record number. These sector number tables are built by HASPINIT at the end of direct-access initialization, using the resident sector convert routine, IEC0SCR1, in the nucleus, An extent's table is built only if that extent's UCB specifies the RPS feature. Tables for the first &NUMDA extents are built on the basis of a record length of &BUFSIZE; the last extent's table is based on a record length of &OLAYSIZ.

RPS support for SYS1.SYSJOBQE in HASPWTR is described in Section 4.21.

(The remainder of this page intentionally left blank.)

5.3 JOB QUEUE MANAGER

5.3.1 Job Queue Manager - General Information

Jobs being processed or awaiting processing by a HASP phase are represented in an ordered queue by a Job Queue Element (see Figure 8.6.1).

The Job Queue Management routines are used by the HASP Processors to insert, alter, locate, and remove Job Queue Elements. The Queue Elements are maintained in priority at all times with the highest priority element at the top of the active chain. There are six Job Queue Element routines which are called by issuing the following macros: $QADD, $QREM, $QGET, $QPUT, $QLOC, and $QSIZ (see Section 9.3). The Job Queue Elements are arranged in two chains. The active chain contains the Job Queue Elements for all the jobs in the system at a given time. The free chain contains all the Queue Elements which are not in use.

5.3.2 $QADD Routine - Program Logic

The $QADD routine is called whenever a Queue Element is to be added to the active queue. If the Checkpoint Processor is waiting for the checkpointed information to be written onto the SPOOL1 disk, this routine enters a HASP $WAIT state. Whenever the Checkpoint Processor's I/O is complete, the free queue chain is tested to see if any free Queue Elements are available. If none are available, control is returned to the caller with a condition code of zero. If a Queue Element is available, the correct slot within the active queue chain is located by comparing the priority of the element to be added with the priorities of the elements in the active chain. When the priority of the new element is higher than the priority of the element in the active chain, the free Job Queue Element is extracted from the free queue chain and inserted into the active chain. All the information for the new Job Queue Element is moved from the location pointed to by register "R1" into the new Job Queue Element. Then the HASP Dispatcher's Event Control Field is $POSTed to indicate that a Job Queue Element is available. The Checkpoint Processor's PCE is also $POSTed so that it will be given control to write the updated Job Queue onto the SPOOL1 disk. The condition code is set non-zero and control is returned to the caller. Upon return, register "R0" contains the address of the associated Job Information Table Entry.

5.3.3 $QREM Routine - Program Logic

The $QREM routine is entered to remove a Job Queue Element from the active chain. It will enter the calling Processor into a HASP $WAIT state if the Checkpoint Processor's I/O is not complete. When the Checkpoint Processor's I/O is complete, the Job Queue Element that is to be removed is located by comparing its job number with the job numbers of the queue elements in the active chain. If an equal comparison is not found, control is returned to the caller with the condition code set to zero. If a match is found, the Job Queue Element is removed from the active chain and added to the top of the free chain by updating all the chain pointers. The Checkpoint Processor's PCE is $POSTed so that it will be given control to checkpoint the Job Queue. Then control is returned to the caller with the condition code set non-zero to indicate that the Queue Element was successfully removed.

5.3.4 $QGET Routine - Program Logic

The $QGET routine is entered to acquire a Job Queue Element in a specified queue so that the job may be processed. The active queue chain is searched for a Job Queue Entry of the specified type (e.g., execution, print, punch, or purge) which is not in hold status and not presently acquired. If such a job is not present, control is returned to the caller with the condition code set to zero. If an acceptable queue element is found, the QENTBY bit is turned on in the queue element to show that the element has been acquired, and control is returned to the caller with the condition code set non-zero, register "R1" pointing to the job queue element that was acquired, and register "R0" pointing to the associated Job Information Table Entry. Whenever the system is in a drained status, this routine will be crippled such that control will always be returned to the caller with the condition code set to zero to indicate that no available Job Queue Elements are present.

5.3.5 $QPUT Routine - Program Logic

The $QPUT routine is entered to return a previously acquired Job Queue Element to the active chain, but with a new queue type. It will enter the calling Processor into a HASP $WAIT state if the Checkpoint Processor's I/O is not complete. When the Checkpoint Processor's I/O is complete, the job number of the queue element to be returned is compared with the job numbers of the queue elements in the active queue. If the job number is not found, control is returned to the caller with the condition code set to zero. If a match is found, the new queue type is set, the HASP Dispatcher's

Event Control Field is posted to indicate that a Job Queue Element is available to be acquired, and the Checkpoint Processor's PCE is $POSTed so that it will be given control to write the updated Job Queue onto the SPOOL1 disk. If the QUEPURGE bit is on in the queue element (indicating that the job has been deleted), the job queue element is placed in the punch queue by moving the punch queue type into the queue element's QUETYPE field. If the QUEPURGE bit is not on, the job queue element is placed in the queue indicated by register "R0" upon entry to this routine. The QENTBY bit is turned off to indicate that this queue entry has been returned, the condition code is set non-zero, and control is returned to the caller. Upon return, register "R1" contains the address of the Job Queue Entry just returned and register "R0" contains the address of the associated Job Information Table Entry.

5.3.6 $QLOC Routine - Program Logic

The $QLOC routine is entered to obtain the Job Queue Element address when the job number is known. The job number is compared with the job numbers in the active chain. If a match is not found, control is returned to the caller with the condition code set to zero. If a match is found, the condition code is set non-zero, and control is returned to the caller with register "R1" containing the located Job Queue Element's address and register "R0" containing the associated Job Information Table Entry address.

5.3.7 $QSIZ Routine - Program Logic

The $QSIZ routine is entered to obtain the number of Job Queue Elements in a given queue type, route, class, and forms. The number of jobs of the specified type (excluding jobs in hold status) are counted, and control is returned to the caller with register "R1" containing this count. If register "R1" is non-zero, the condition code is set non-zero, and if it is zero, the condition code is set to zero. Whenever the system is in a drained status, this routine is crippled so that control is always returned to the caller with register "R1" zeroed, and the condition code set to zero to indicate that no jobs are available in the specified job queue.

5.4 BUFFER MANAGER

5.4.1 Buffer Manager - General Description

The Buffer Management routines are responsible for the allocation of the dynamic memory area (Buffer Pool) of HASP. Fixed-size buffers in this area are allocated and de-allocated to HASP Processors and Routines via the $GETBUF and $FREEBUF macro-instructions (see Section 9.1).

5.4.2 Buffer Manager - Program Logic

The $GETBUF routine consists of two programs which allocate HASP Buffers or RJE Buffers respectively. Both programs function identically as follows: The appropriate free buffer pointer is tested, and if no buffers are available, control is returned to the caller with the condition code set to zero. If a free buffer is present, the free buffer pointer is updated to point to the next free buffer; or, if this is the last available buffer, the pointer is zeroed. Then, if the debug indicator is on, a buffer validity checking routine is entered to assure that the buffer is within the buffer chain. If it is not in the chain, the catastrophic error routine is entered; otherwise, control is returned to the $GETBUF routine. The condition code is set non-zero and control is returned to the caller with the buffer address in register "R1".

The $FREEBUF routine enters the buffer validity checking routine if the debug indicator is on, the buffer to be freed is inserted back into the appropriate free buffer chain (depending upon whether the buffer is a HASP Buffer or an RJE buffer), and the IOBSTART field is updated with the address of the buffer's channel program: IOBCCW1 (see Figure 8.3). The HASP Dispatcher's Event Control Field is $POSTed to show that a buffer is available and control is returned to the caller.

5.5 UNIT ALLOCATOR

5.5.1 Unit Allocator - General Description

The Unit Allocation routines are responsible for the allocation and de-allocation of the Input/Output units which have been assigned to HASP. Device Control Tables (DCTs) are allocated and de-allocated to HASP Processors and Routines via the $GETUNIT and $FREUNIT macro-instructions (see Section 9.2).

5.5.2 Unit Allocator - Program Logic

The $GETUNIT routine scans the Device Control Table (DCT) chain in an attempt to find an available DCT of the requested type. If none are found, control is returned to the caller with the condition code set to zero. If an available DCT of the requested type is found, it is set "not available" and control is returned to the caller with the condition code set non-zero. The address of the DCT is returned in register "R1".

The $FREUNIT routine first examines the "Active Buffer Count" field of the DCT (see Figure 8.5) to see if there are any buffers involved in active I/O with the associated unit. If the "Active Buffer Count" is non-zero, the Processor is placed in a HASP $WAIT state until this count is reduced to zero. When the count is zero, the DCT is made available and control is returned to the caller.

5.6 INTERVAL TIMER SUPERVISOR

5.6.1 Interval Timer Supervisor - General Description

The Interval Timer Supervisor is used by the various HASP Processors to record the passage of a specified period of time and to notify the requesting Processor upon expiration of the interval. This routine uses the standard OS/360 timer features (STIMER & TTIMER) but has the additional capability to simultaneously monitor an unlimited number of intervals.

5.6.2 Interval Timer Supervisor -Program Logic

All uses of the Interval Timer Supervisor are through the HASP macro-instructions $STIMER and $TTIMER which are described in Section 9.6. Each user of $STIMER is required to provide a 12-byte (three-word) HASP Timer Queue Element (TQE), passed via parameter register "R1" (see Section 8.10). $STIMER maintains a chain of all active TQEs in ascending order of interval magnitudes, with the shortest requested interval (first TQE) set on the OS STIMER queue (via a normal STIMER macro). Upon being entered with a new interval request, $STIMER first cancels the active OS timer element with a TTIMER CANCEL, and reduces the interval specified in all chained TQEs by the elapsed portion of this interval. The requestor's TQE is then, after converting the requested interval to OS timer units (26 usec units), inserted into the appropriate place on the TQE chain using the first word of the TQE as a chain field. The OS timer is now re-activated with the interval in the first TQE in the chain and return is made to the caller.

When the current OS interval elapses, the asynchronous exit routine in $STIMER is entered to record the expiration. The asynchronous routine first reduces the intervals of all queued TQEs by the size of the just-elapsed interval, then $POSTs the TIMER Processor, POSTs the HASP task, and returns to OS. The TIMER Processor, when dispatched, will $POST the appropriate Processors and reset the OS Timer to the interval specified in the first TQE in the chain by issuing an STIMER macro.

HASP Processors which have previously set an interval through $STIMER may obtain the time remaining in the interval and optionally cancel this interval through the use of the $TTIMER macro. When entered, $TTIMER cancels the active OS interval and reduces all queued TQE intervals by the elapsed portion of that interval. The requestor's TQE is then located in the queue by comparing the address of the TQE passed by the macro in register "R1" to each TQE in the chain. When the correct TQE is found, the remaining time

in the interval is loaded in register "R0" for return to the caller. The use of the CANCEL option on the $TTIMER macro, which is indicated by register "R1" containing the complement of the TQE address rather than the true address, causes the TQE to be dequeued from the chain. The OS timer is re-activated with the interval from the first TQE on queue and return is made to the caller. NOTE: A $TTIMER for a TQE which is not active has no effect and a zero value is returned in register "R0" as the time remaining.

5.7 $WTO PROCESSING ROUTINE

5.7.1 $WTO Processing Routine — General Description

This routine services the $WTO macro-instruction (see Section 9.5) by queuing the associated message for the Operator Console Input/Output Processor.

5.7.2 $WTO Processing Routine — Program Logic

This routine tests for a free message buffer. If none are available, it causes the requesting processor to be placed in a $WAIT condition until a message buffer is released. Otherwise it links to the Console Buffering Routine to process the message.

5.8 DIRECT ACCESS STORAGE ALLOCATOR

5.8.1 Direct Access Storage Allocator - General Information

This routine allocates tracks for the SPOOL volumes that were on-line at IPL time. The track information is stored in the Job Control Table (JCT) and is also returned to the caller in register "R1". The track allocation algorithm is designed to reduce seek time as much as possible.

5.8.2 Direct Access Storage Allocator - Program Logic

The status of each SPOOL volume is recorded and maintained in track group bit maps. A map is present for each module (available SPOOL volume). Each bit in the track group bit map represents a track group. If the bit is on, the track group is available to be allocated, and if the bit is off, the track group has already been allocated. Track group bit maps are also maintained in each JCT, but the bit definitions are opposite. Thus, if a bit is on in the JCT, the track group has been allocated to the JCT.

Track groups on the SPOOL volumes are allocated whenever the JCT has not previously acquired any tracks or whenever all the tracks in the current track group which is allocated to the JCT have been acquired. If the JCT has already been allocated a track group, but all the available tracks in that track group have not been acquired, the next available sequential track in the track group is allocated to the requestor. When this happens, the track information in the JCT is updated and loaded into register "R1", and control is returned to the caller with the condition code set to one. This track information is recorded in the JCT in the following format: MTTR, where M is the module number (one byte), TT is the track number relative to cylinder 0 track 0 (two bytes), and R is the record number (one byte). The JCT track group bit map is also updated whenever a new track group is acquired. The update consists of ORing in the appropriate bit for the acquired track group in the JCT track group bit map.

When a new track group has to be acquired, seek time is reduced by searching for the nearest track group + or - eight track groups from the last-used track group. The last-used track group for each track group bit map is updated each time a $EXCP is issued to the volume. Each track group bit map is searched for an available track group at the last-used track group. Then each track group bit map is searched for an available track group - one track group from the last-used track group, then + one from the last-used track group and this progression continues until an

available track group is found or the + eight track group is searched. If an available track group is found, the JCT track information is updated and loaded into register "R1", and control is returned to the caller with the condition code set to one. The JCT track group bit map is also updated. If a track group is not available within + or - eight of the last-used track group, another search routine is entered which inspects each byte of the track group maps, starting with the first byte. This search will continue until an available track group is found or until all of the active track group bit maps have been searched. If an available track group is found, the JCT track information is updated and loaded into register "R1", and control is returned to the caller with the condition code set to one. The JCT track group bit map is also updated. If an available track group is not found, the operator is notified of the out-of-track condition by the following message:

SPOOL VOLUMES ARE FULL

Then control is returned to the caller with the condition code set to zero and register "R1" zeroed.

5.8.3 Direct Access Storage Purge Routine - Program Logic

This routine frees all of the SPOOL volume tracks that the job has acquired and informs the system that these tracks are available to be re-acquired.

The track group bit map in the job's Job Control Table is ORed into the main track group bit map to return the job's tracks back to the system. Then the track group bit map in the JCT is zeroed to indicate that this job does not have any tracks allocated to it. The HASP dispatcher's Event Control Field is posted to show that tracks are available to be acquired, and control is returned to the caller.

5.9 DISASTROUS ERROR HANDLER

5.9.1 Disastrous Error Handler - General Description

This routine is entered from a Processor whenever a critical SPOOL disk error is detected. The operator is notified of the error, and processing continues, although the operator should re-IPL the system with a cold start as soon as possible.

5.9.2 Disastrous Error Handler - Program Logic

When this routine is entered, a $WTO is issued to notify the operator of the error, and control is returned to the calling Processor. The message to the operator is as follows:

```
DISASTROUS ERROR - COLD START SYSTEM ASAP
```

5.10 CATASTROPHIC ERROR HANDLER

5.10.1 Catastrophic Error Handler - General Description

This routine is entered whenever an unrecoverable error is discovered by HASP. The operator is informed of the error and given an error code, and the system enters a one instruction disabled loop. The error codes and their meanings are listed in the HASP Operator's Guide (see Section 11). For more information, refer to Section 9.10.1.

5.10.2 Catastrophic Error Handler - Program Logic

When this routine is entered, register "R0" contains the address of a four byte field containing the three character error code left justified. After the system is disabled, the four byte error code field is moved into the operator message. This message is then written on the operator's console defined by the HASPGEN parameter "$PRICONA":

```
$ HASP SYSTEM CATASTROPHIC ERROR.  CODE = xxx
```

After this message is typed, all registers are restored so that they will be intact, and a one instruction loop is executed.

5.11 TRACE EFFECTOR

5.11.1 Trace Effector — General Description

The Trace Program is a debug facility used in HASP which is completely independent of the OS trace facility. This program will insert the contents of the general purpose registers into a special trace table (assembled into the HASP module) each time it is called and thereby aid in the determination of HASP problems.

5.11.2 Trace Effector — Program Logic

The Trace Program is called by any Routine or Processor in HASP by the insertion of a $TRACE macro-instruction (see Section 9.9.1). If the HASPGEN parameter "&TRACE" is set non-zero, the macro-instruction will expand into an instruction which will cause a unique specification program interrupt. All program interrupts are fielded by the HASP Trace Program and the instruction which caused the interrupt is tested to determine if it is the unique instruction inserted by the $TRACE macro-instruction. If the interrupt was caused by a true program interrupt, the request is sent to the first level interrupt handler, to be handled in the normal way. Otherwise a sixteen word trace entry is inserted into the HASP trace table.

The sixteen word trace entry has the following format:

First Byte.....$TRACE count

First Word.....$TRACE storage location

Second Word.....Register 0

Third Word.....Register 1

Fourth Word.....Register 2

Fifth Word.....Register 3

Sixth Word.....Register 4

Seventh Word.....Register 5

Eighth Word.....Register 6

Ninth Word.....Register 7

Tenth Word.....Register 8

Eleventh Word.....Register 9

Twelfth Word.....Register 10

Thirteenth Word.....Register 12

Fourteenth Word.....Register 13

Fifteenth Word.....Register 14

Sixteenth Word.....Register 15

After the trace table entry has been inserted and the pointers updated, the count of the number of times this particular $TRACE macro-instruction has been executed is inserted into the first byte of the first word of the

the trace entry and also into the last half of the $TRACE "instruction." All registers are then restored and return is made by loading the Program Old PSW which restores the condition code to its original value before the $TRACE macro-instruction was executed.

The symbolic location "$TRACETB" in HASP identifies a three-word table with the following format: the first word is the address of the last entry which was made in the trace table; the second word is the address of the first byte of the trace table; and the third word is the address of the last byte of the trace table + 1.

5.12 WTO/WTOR PROCESSING ROUTINE

5.12.1 WTO/WTOR Processing Routine - General Description

The function of this routine is to process all OS WTO's and WTOR's. If a console buffer is not available for the message the requesting task is placed in an OS WAIT state until a buffer becomes available to process the request. This routine is not included if the HASP interface to OS Console Support is generated (&NUMCONS=0, see Appendix 12.15).

5.12.2 WTO/WTOR Processing Routine - Program Logic

The WTO/WTOR Processing Routine is entered from the Execution Control Processor whenever an SVC 35 or optionally SVC 36 is issued. The routine performs the following functions:

1. HASP is forced dispatchable and a task switch is signalled when appropriate.

2. Standard HASP $WTO parameters are set up for OS and LOG operator consoles. If the entry is for SVC 36 the OS and LOG operator console request is deleted allowing only logging to the HASP SYSTEM LOG. The number of output lines desired is set to 1.

3. If the SVC is for WTOR a check is made to insure that both a Console Message Buffer and Reply Element are available before further processing occurs. When facilities are available the parameter list is checked, the reply number assigned to the Reply Element is assigned to the message, the Reply Element filled out and queued, and normal WTO processing is resumed at step 6 below.

4. If the WTO is a multi-line WTO the format of the parameter list is determined (see Figure 5.12.3) and number of output lines desired is set as specified in the parameter list.

5. The text of the message is examined for possible elimination and/or identification of the OS jobname for use in step 6.

6. OS control blocks are searched for the purpose of associating the message with a current HASP controlled job. If an association is made the $WTO parameters will reflect a request for the job number to appear with the message and the HASP SYSTEM LOG is to contain a copy of the message.

7. A check is made to insure that a Console Message Buffer is available before continuing. (For WTOR one will be available at this point.)

8. The parameter list is altered to show request not $WTO and the Console Buffering and Queueing Routine is called upon to queue the message.

9. If the number of output lines was more than 1 additional lines are set up for each succeeding line by re-executing steps 7 to 8 above until all lines are queued.

10. Upon completion of all lines, the routine returns to OS via register 14.

A list of lines is terminated if a line length is zero, the first line is eliminated by message type elimination, or the DE or E line type parameter is encountered. If facilities are not sufficient to handle the SVC 35 or 36 request immediately and it is determined that the task can not wait or there are no WTO/WTOR Task Wait Elements (Figure 5.12.1) available, the SVC 35 or 36 request is ignored. MCS flags in the WTO/WTOR parameter lists are examined to determine the format of the parameter list only. The acceptable formats and MCS flag settings examined are listed in Figure 5.12.3.

Figure 5.12.1 -- WTO/WTOR TASK WAIT ELEMENT

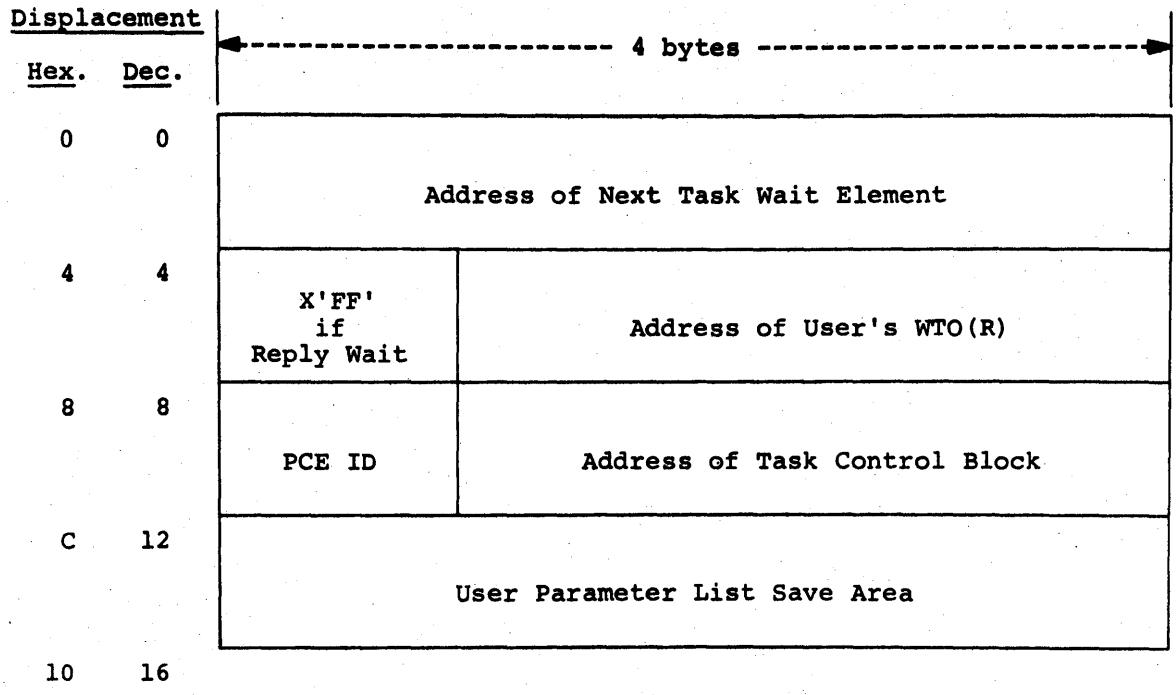


Figure 5.12.2 -- WTOR REPLY ELEMENT

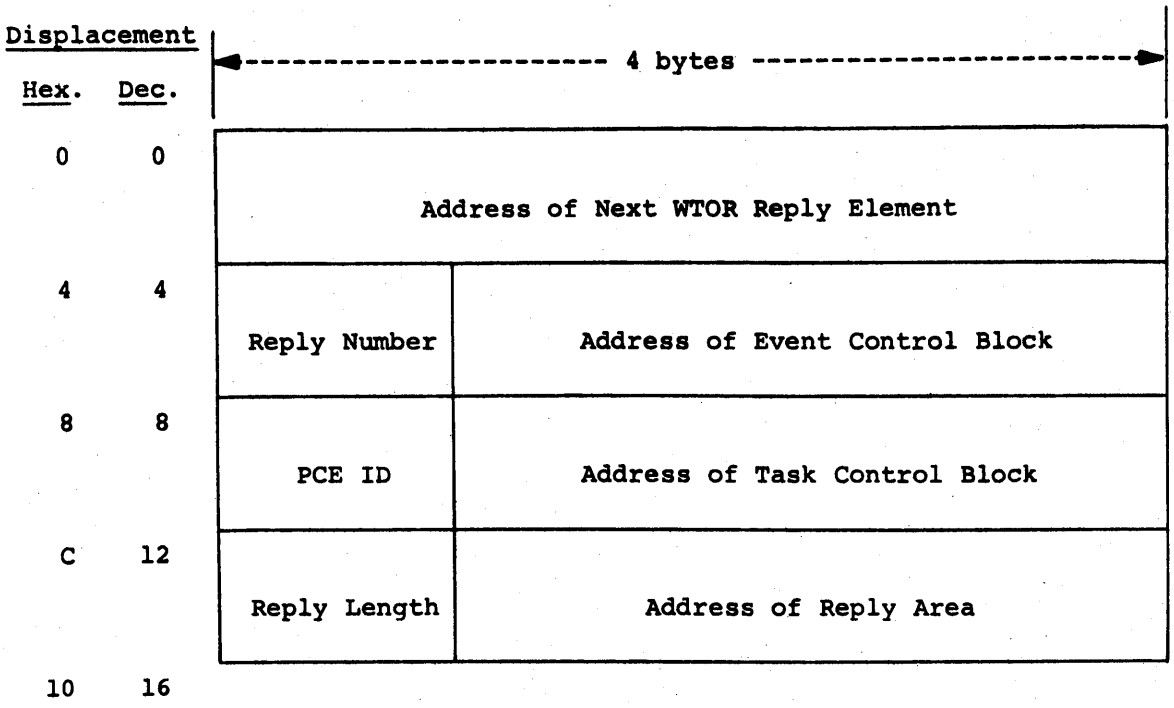


Figure 5.12.3 -- WTO/WTOR PARAMETER LIST FOR HASP CONSOLE SUPPORT

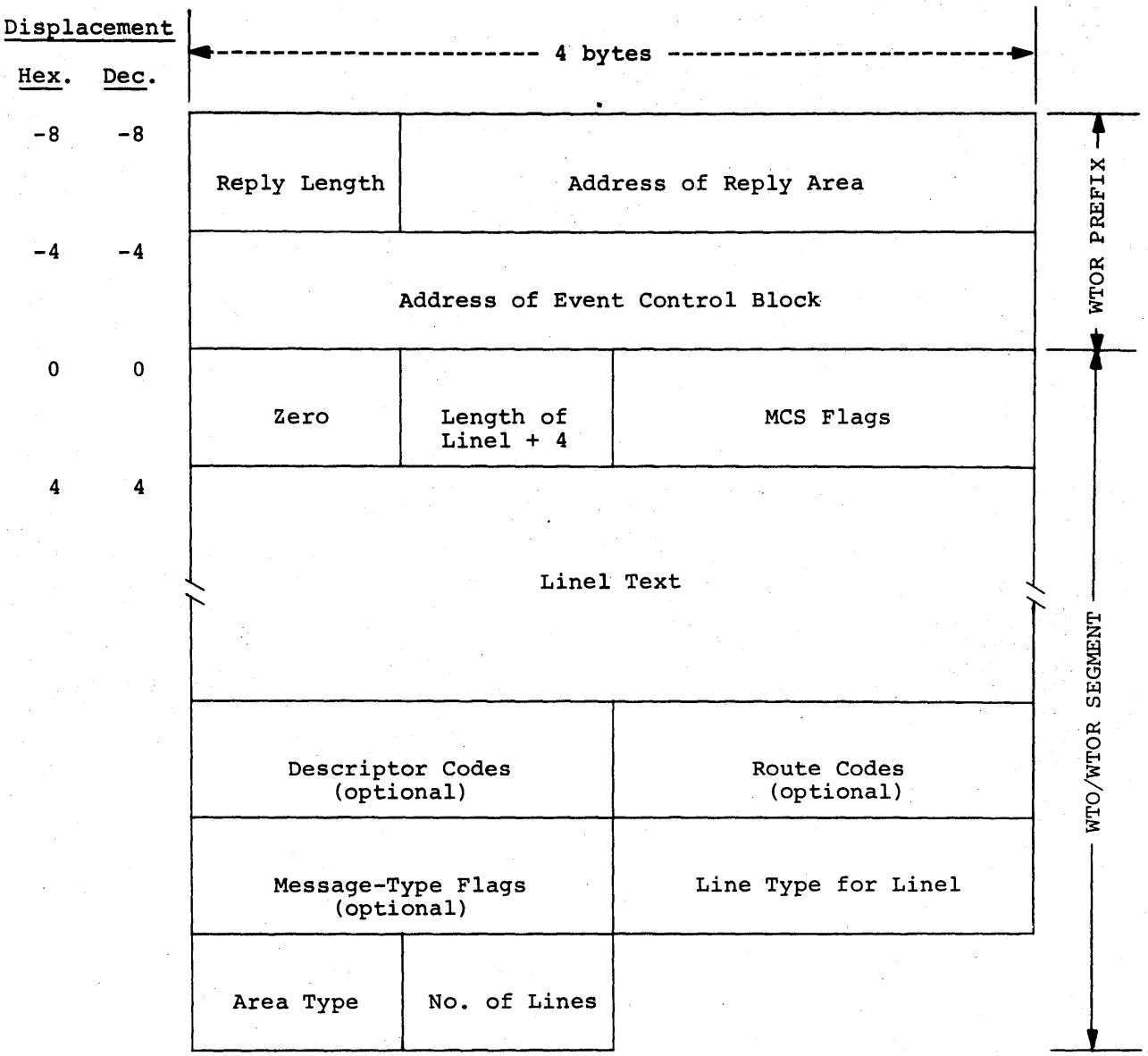
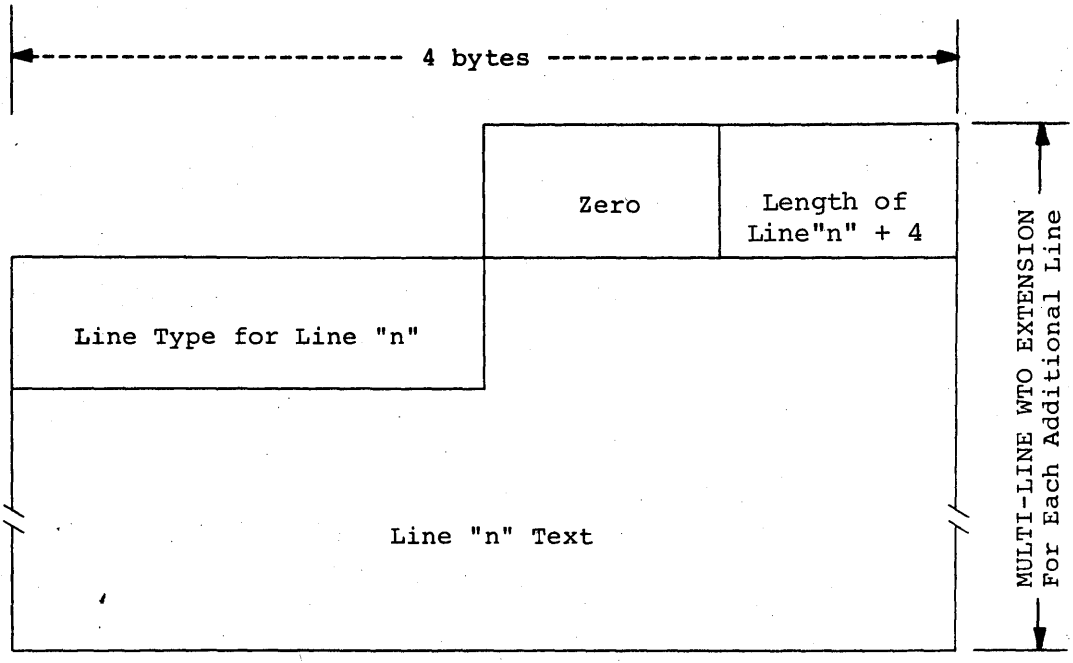


Figure 5.12.3 -- WTO/WTOR PARAMETER LIST (CONTINUED)



5.13 CONSOLE BUFFERING AND QUEUING ROUTINES

The following routines are responsible for the queuing and de-queuing of all console and log messages.

5.13.1 CONSOLE BUFFERING ROUTINE - PROGRAM LOGIC

The Console Buffering Routine is used to prepare a message buffer with the information required to process a console message. At entrance registers zero and one contain the information shown in figure 5.13.1.

The routine makes use of three tables comprised of one-byte entries. The bits in each byte specify the physical consoles which are to be used for the respective entry. In the first ($WCONTBL) each byte corresponds to one of eight consoles. The bytes are ORed for each specified symbolic console to set the physical byte for a write operation.

A second table ($WCLASTB) has an entry for each of the sixteen possible message classes. The appropriate byte is ANDed with the physical consoles byte to screen out consoles with the class set too high.

In addition to setting the console routing byte, the Console Buffering Routine supplies the other information shown in figure 8.4.1. Prior to returning to the caller, the routine places the message in the log queue (non-HASP messages with a job number), or in the queue of messages to be processed by the Console Input/Output Processor (all other output messages and all reads).

5.13.2 CONSOLE QUEUING ROUTINE - PROGRAM LOGIC

This routine places a console buffer into a queue of messages, according to priority, to be processed by the Operator Console Input/Output Processor and $POSTs that processor.

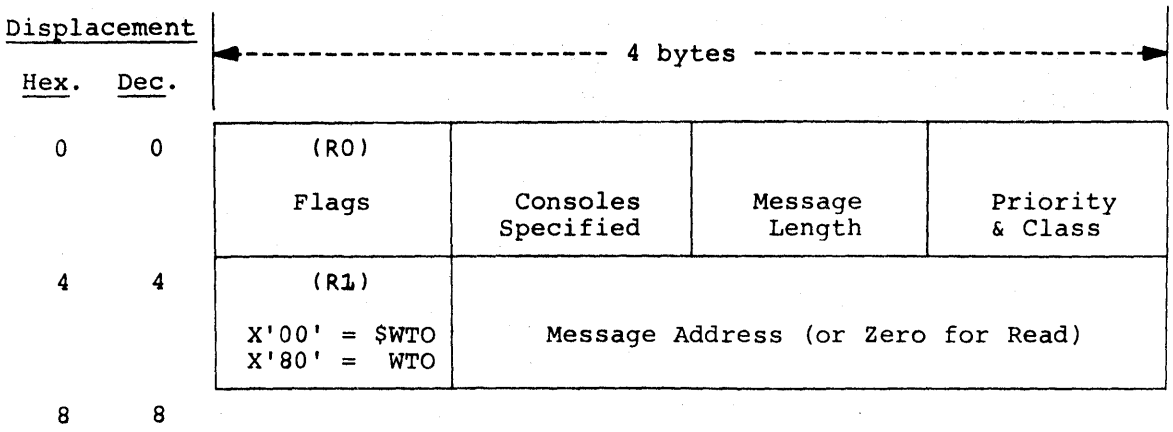
5.13.3 LOG QUEUING ROUTINE - PROGRAM LOGIC

This routine places a console buffer at the end of the queue of messages to be processed by the HASP Log Processor and $POSTs that processor.

5.13.4 CONSOLE BUFFER FREEING ROUTINE - PROGRAM LOGIC

This routine places the console buffer in the free queue. The Attention Processor's PCE is examined to determine if the Attention Processor is $WAITing for a console buffer, and if it is, the Attention Processor is $POSTed and exit is made. If the Attention Processor is not $WAITing, the $WTORQUE is tested and the first task found is POSTed. If no tasks are waiting, the HASP Event Control Field is $POSTed and exit is made.

Figure 5.13.1 -- CONSOLE BUFFERING ROUTINE PARAMETER REGISTERS



5.14 INPUT/OUTPUT ERROR LOGGING ROUTINE

5.14.1 Input/Output Error Logging Routine -- General Description

This routine is entered whenever an unrecoverable Input/Output error occurs on a HASP direct-access intermediate storage device, or whenever line errors occur which may require the attention of the operator. A message is generated describing the error and this message is routed to the operator via the operator's console. The routine then returns without taking any further action.

5.14.2 Input/Output Error Logging Routine — Program Logic

When this routine is entered, register "R1" contains the address of the Input/Output Block (IOB) which is associated with the Input/Output operation in error. The channel status, channel command code, sense information, track address, and line status are retrieved from the IOB and formatted; the unit address and volume serial are obtained from the Unit Control Block (UCB); the device name (if applicable) is acquired from the Device Control Table (DCT); and the message is written to the operator's console.

The format of the message describing a direct-access error is as follows:

I/O ERROR ON SPOOLn uuu,cc,ssss,iiii,bbcchhr

where:

n	— identifies the SPOOL disk in error
uuu	— unit address of disk
cc	— channel command code being executed
ssss	— channel status code
iiii	— unit sense information
bbcchhr	— track address as follows:

bb — bin (always zero)

cc — cylinder

hh — head

r — record

The format of the message describing a line error is as follows:

I/O ERROR ON LINEm uuu,cc,ssss,iirr,ttee

where:

m	— line number
uuu	— unit address of line
cc	— channel command code being executed
ssss	— channel status code
ii	— unit sense information

where:

rr	— STR - sense information
	BSC - terminal response
tt	— internal sequence and command code
ee	— STR - always blank
	BSC - expected response

5.15 REMOTE TERMINAL ACCESS METHOD (RTAM)

5.15.1 Remote Terminal Access Method -- General Description

The Remote Terminal Access Method provides an interface between the HASP Processor and the Remote Terminal. RTAM provides blocking/deblocking, compression/decompression, and synchronization with the remote terminal in such a way that the processor need not be concerned with the characteristics of the remote with which he is communicating. The MULTI-LEAVING Line Manager synchronizes very closely with RTAM through a series of subroutines, the more important ones, of which, are briefly described below.

5.15.2 Remote Terminal Access Method -- Program Logic

The Remote Terminal Access Method consists of four main sections and some miscellaneous subroutines. This section discusses the four main sections: OPEN, GET, PUT, and CLOSE. The primary subroutines are discussed in Section 5.15.3 below.

OPEN

The OPEN routines convert the line from an idling mode of operation to a transmit or receive mode of operation. In the case of the MULTI-LEAVING interface, this routine also generates the request or permission to begin a new function.

GET

The GET routines convert data received from the line into EBCDIC images suitable for processing by the HASP processors. This conversion includes deblocking, decompression, and conversion from line code to EBCDIC.

PUT

The PUT routines convert data from EBCDIC into a form ready to be transmitted to the remote terminal. This conversion includes compression, blocking, and conversion from EBCDIC to line code.

CLOSE

The CLOSE routines convert the line from a transmit or receive mode of operation to an idling mode of operation.

5.15.3 Remote Terminal Access Method -- Subroutines

This section describes the primary subroutines used by the Remote Terminal Access Method and the MULTI-LEAVING Line Manager.

MSIGNON -- Sign-On Card Processor

This subroutine is passed the address of a /*SIGNON card in register "R1". If the line from which the Sign-On Card was read was defined to be a "leased" line, the Sign-On Card is ignored and the subroutine returns immediately. If the line is a "dial" line, the MABORT and MDISCON subroutines are called to disconnect any other remote which may have been attached to

this line. The password is then checked and if not valid, an error message is issued and the subroutine returns. If the password is valid the specified Remote Terminal's DCT's are located and examined. If the specified remote is already attached to another line or if the specified remote is not locatable, the subroutine issues an error message and returns. Otherwise, the specified remote is attached to the line and a confirmation message is issued.

MCCWINIT -- Channel Command Word Sequence Setup Subroutine

This subroutine is passed a sequence type in bits 24-27 of register "R1". The subroutine then constructs a CCW chain based upon this value and returns. Figure 5.15.1 depicts the various CCW sequences which can be constructed by the subroutine.

MINITIO -- MULTI-LEAVING Input/Output Interface

This subroutine analyzes the status of a MULTI-LEAVING Remote Terminal and takes appropriate action to minimize degradation while insuring maximum line throughput. The subroutine first establishes the status of every processor currently active on the MULTI-LEAVING line. Then, based upon the active input processor count, the active output processor count, the status of the remote terminal, and the status of input and output buffers queued within HASP either transmits an ACK0 to the terminal, transmits a text buffer to the terminal, or initiates a one-second delay.

MEXCP -- Remote Terminal Input/Output Interface

This subroutine interfaces the Remote Terminal Access Method with the standard HASP "$EXCP" Input/Output Interface. In addition to initiating I/O, this subroutine also provides the MULTI-LEAVING Block Control Byte sequence count, and the BSC 2770/2780 parity check (ACK0-ACK1) conversion.

Figure 5.15.1 — HASP Remote Terminal CCW Sequences

STR Hardware Terminal Prepare Sequence (code=4)

CCW	COMMAND	DATA ADDRESS	FLAGS	INTERNAL CODE	BYTE COUNT
IOBCCW1	DISABLE	0	60	40	1
IOBCCW2	SET MODE	LCBMCB	60	41	2
IOBCCW3	ENABLE	0	60	42	1
IOBCCW4	TEST SYNCH	0	60	43	15
IOBCCW5	SEND INQUIRY	0	20	4A	6

STR CPU Terminal Prepare Sequence (code=5)

CCW	COMMAND	DATA ADDRESS	FLAGS	INTERNAL CODE	BYTE COUNT
IOBCCW1	DISABLE	0	60	50	1
IOBCCW2	SET MODE	LCBMCB	60	51	2
IOBCCW3	ENABLE	0	60	52	1
IOBCCW4	TEST SYNCH	0	60	53	15
IOBCCW5	SEND EOT	0	60	5B	6
IOBCCW6	PREPARE	0	60	57	1
IOBCCW7	READ	TPBUFST	20	54	&TPBFSIZ

Figure 5.15.1 (continued) — HASP Remote Terminal CCW Sequences

STR Read Sequence (code=0: Hardware; code=1: CPU)

CCW	COMMAND	DATA ADDRESS	FLAGS	INTERNAL CODE	BYTE COUNT
IOBCCW1	TEST SYNCH	0	60	03/13	15
IOBCCW2	PREPARE	0	60	07/17	1
IOBCCW3	READ	TPBUFST	20	04/14	&TPBFSIZ
IOBCCW4	STEP COUNT	0	60	00/10	1
IOBCCW5	ERROR	0	60	00/10	1
IOBCCW6	TIC	IOBCCW3	00	00/10	0

STR Write Sequence (code=2: Hardware; code=3: CPU)

CCW	COMMAND	DATA ADDRESS	FLAGS	INTERNAL CODE	BYTE COUNT
IOBCCW1	TEST SYNCH	0	60	23/33	15
IOBCCW2	SEND INQUIRY	0	60	2A/3A	6
IOBCCW3	WRITE	TPBUFST	20	28/38	*-*

Figure 5.15.1 (continued) — HASP Remote Terminal CCW Sequences

BSC Prepare Sequence (code=C)

CCW	COMMAND	DATA ADDRESS	FLAGS	INTERNAL CODE	BYTE COUNT
IOBCCW1	DISABLE	0	60	C0	1
IOBCCW2	SET MODE	LCBMCB	60	C1	1
IOBCCW3	ENABLE	0	60	C2	1
IOBCCW4	NOP	MBSCSYN	60	CA	4
IOBCCW5	NOP/WRITE	MBSCENQ/MBSCEOT	60	CA	1
IOBCCW6	READ	LCBRCB	20	C6	2

BSC MULTI-LEAVING Terminal Sequence (code=9)

CCW	COMMAND	DATA ADDRESS	FLAGS	INTERNAL CODE	BYTE COUNT
IOBCCW1	ENABLE	0	60	92	1
IOBCCW2	NOP	MBSCSYN	60	99	4
IOBCCW3	WRITE	LCBRCB	60	99	2
IOBCCW4	READ	TPBUFST	20	94	&TPBFSIZ
IOBCCW5	NOP	MBSCSYN	60	98	4
IOBCCW6	WRITE	TPBUFST	60/A0	98	*-*
IOBCCW7	WRITE	METBSEQ	60	98	2
IOBCCW8	READ	TPBUFST	20	B4	&TPBFSIZ

Figure 5.15.1 (continued) — HASP Remote Terminal CCW Sequences

BSC Hardware Terminal Read Sequence (code=8)

CCW	COMMAND	DATA ADDRESS	FLAGS	INTERNAL CODE	BYTE COUNT
IOBCCW1	ENABLE	0	60	82	1
IOBCCW2	NOP	MBSCSYN	60	89	4
IOBCCW3	WRITE	LCBRCB	60	89	2
IOBCCW4	READ	TPBUFST	20	84	&TPBFSIZ

BSC Hardware Terminal Write Sequence (code=A)

CCW	COMMAND	DATA ADDRESS	FLAGS	INTERNAL CODE	BYTE COUNT
IOBCCW1	ENABLE	0	60	A2	1
IOBCCW2	NOP	MBSCSYN	60	AA	4
IOBCCW3	WRITE	MBSCENQ	60	AA	1
IOBCCW4	READ	LCBRCB	20	A6	2
IOBCCW5	NOP	MBSCSYN	60	A8	4
IOBCCW6	WRITE	TPBUFST	60	A8	*-*
IOBCCW7	WRITE	METBSEQ	60	A8	2
IOBCCW8	READ	LCBRCB	20	A5	2

5.16 OVERLAY SERVICE ROUTINES

5.16.1 Overlay Service - General Description

These routines, together with the Overlay Roll Processor described in Section 4.20, respond to calls from other HASP Processors when the macros $LINK, $LOAD, $XCTL, $RETURN, and $DELETE are executed in HASP coding. This enables certain executable and table portions of HASP coding (assembly control sections created by use of the $OVERLAY macro) to be brought into main storage from their normal direct access residence for use during HASP execution.

Major objectives of Overlay Service and Roll logic are: to allow multiple Processors to use a single copy of the same overlay routine simultaneously, and to prevent any system lockout due to $WAITs in overlay routine coding.

The overlay data set is constructed as part of HASP installation by the HASP Overlay Build utility, described in Sections 10.2.2 and 6.3, and is referred to by the ddname OLAYLIB in the job which invokes HASP.

All Overlay Service and Roll Processor coding is located in module HASPNUC. Service entry points are addressable by register BASE1 and are referenced by macro expansions through the HASP Communication Table.

Actions necessary to initialize HASP Overlay Service are contained in module HASPINIT and are described in Section 6.1.2.

See Sections 8.3.3, 9.7, and 12.14 for descriptions of Overlay Area(s) format, macros mentioned above, and coding rules relating to use of overlay routines.

5.16.2 $LINK Service - Program Logic

On entry, register "R15" contains the address of the next instruction after $LINK and register "LINK" contains the called routine's Ocon. An Ocon is an index into the HASP Overlay Table, which is the control section HASPOTAB created by the HASP Overlay Build utility, whose individual entries are defined in OTBDSECT, created by the $OTB macro.

The calling Processor's registers "R0-WC" are saved in the caller's PCE. Overlay Service base address is established in register "WC". Register "R15" is saved in PCEORTRN. "R15" is set to the relative displacement of the called routine entry point from the beginning of an Overlay Area IOB, i.e., OACEPROG-BUFDSECT. The called routine Ocon is saved in PCEOCON, then used to compute the address of the Overlay Table entry for the called routine. If &DEBUG is set

to YES, field OTBCALLS is incremented by one. The called routine's priority is moved to PCEOPRIO.

If the Overlay Table indicates that the called routine was made a permanent part of the HASP Load Module at Overlay Build time, register BASE3 is loaded with the address of a theoretical Overlay Area containing the resident routine (BUFSTART-BUFDSECT bytes prior to the routine itself), caller's "R0-WC" are reloaded, and control is passed to the called routine at its entry point.

If the called routine is not permanently resident, a search is made of all Overlay Areas in the system. If the called routine is found in an area (PCEOCON equal to area's OACEOCON), the caller's PCE is added to the chain of all active users of the area. This chain begins at OACEPCE and continues through PCEOPCE of each PCE, if several users are on the chain, and ends with a zero chain word. A test is made for illegal nested $LINK if &DEBUG is set to YES, see Operator's Guide for error message. If the called routine is in process of being read into the area from direct-access, the calling Processor is made to $WAIT on OLAY, to be later activated by the Overlay $ASYNC Exit (see 5.16.9). Otherwise, caller's "R0-WC" are reloaded and control is passed to the called routine entry point, with register BASE3 containing the address of the Overlay Area IOB for use as the overlay routine base address.

If the called routine is not found while searching all Overlay Areas, the search attempts to find an Overlay Area which is not currently in use. It may contain an overlay routine but may not have active users (OACEPCE must be zero). The inactive area containing the routine of lowest priority (OACEPRIO) will be used, subroutine OLOD (see 5.16.8) will be called to start reading the called routine from direct-access, and the calling Processor will be $WAITed on OLAY, to be later activated by Overlay $ASYNC Exit (see 5.16.9).

If no inactive areas are found, the calling PCE is placed on a Queue waiting for an Overlay Area. The Queue begins at the word $WAITACE, continues in descending priority order by PCEOPRIO using chain word PCEBASE3, and ends with a zero chain word. If several PCEs are on the Queue requesting the same overlay routine (PCEOCONs equal), only the first PCE is on the above chain, the others are chained from it using word PCEOPCE. All PCEs in the Queue are $WAITed on OLAY. This Queue is emptied by the Overlay Roll Processor, as described in Section 4.20, or by the OEXIT subroutine, as described in 5.16.7.

5.16.3 $LOAD Service - Program Logic

$LOAD shares almost all logic with $LINK (see 5.16.2). Entry register conditions are identical to those for $LINK.

"R15" is not saved in PCEORTRN. "R15" is not set to the relative entry point of the called routine.

When the called routine is found in an Overlay Area or read into one by later system actions, "R15" still contains the address of the next instruction after $LOAD. Subsequent use of "R15" as an absolute entry point results in control being returned to the caller with the routine in an actual or theoretical area, addressable by BASE3 as with $LINK.

5.16.4 $XCTL Service - Program Logic

$XCTL logic shares almost all logic with $LINK (see 5.16.2). Entry register conditions are identical to those for $LINK.

"R15" is not saved in PCEORTRN. $XCTL is legal only when it logically follows another $XCTL or an original $LINK. Subsequent $RETURN uses PCEORTRN as stored by the original $LINK to return control from Overlay Service to the original caller.

Before doing entry actions for the new called overlay routine, the OEXIT subroutine is called (see 5.16.7) to remove the calling Processor's PCE from the chain of users of the current overlay routine.

5.16.5 $RETURN Service - Program Logic

On entry, register LINK points to the next instruction after $RETURN and also contains the condition code and program mask as set by a BAL instruction. BASE3 points to an actual or theoretical area containing the current overlay routine.

Caller's "R0-WC" are saved in the PCE. Overlay Service base address is established in WC.

The OEXIT subroutine is called (see 5.16.7) to remove caller's PCE from the chain of users of the current overlay routine.

Returned condition code is re-established using an SPM instruction. Caller's "R0-WC" are reloaded. Control is returned to the address previously saved in PCEORTRN by $LINK.

5.16.6 $DELETE Service - Program Logic

$DELETE is nearly identical to $RETURN, except that it is used to release control of an overlay routine previously $LOADed.

On entry, register LINK points to the next instruction after $DELETE. This is stored in PCEORTRN and all actions described for $RETURN are performed.

5.16.7 OEXIT Subroutine - Program Logic

This subroutine is used by service routines for $XCTL, $RETURN, and $DELETE to release use of the current overlay routine by the calling Processor. On entry, register WA contains the subroutine return address and register BASE3 contains the address of an actual or theoretical (permanently resident routine) Overlay Area containing the current overlay routine.

If the current overlay routine is permanently resident, OEXIT returns immediately. Otherwise, the chain of all users of the area (beginning at OACEPCE and continuing through PCEOPCE) is searched and the caller's PCE is removed. If other Processors are still using the area, OEXIT returns.

If the above actions result in the Overlay Area becoming inactive (OACEPCE equal zero), the $WAITACE Queue (see 5.16.2) is inspected. If PCE(s) are waiting, the top priority group of one or more requesting the same overlay routine is de-queued, the address of the first such PCE is placed in register "R1", and OEXIT simply falls through to the OLOD subroutine (5.16.8), which eventually returns to the caller of OEXIT.

5.16.8 OLOD Subroutine - Program Logic

This subroutine is used by service routines for $LINK, $LOAD, $XCTL; by the Overlay Roll Processor (see Section 4.20); and indirectly by users of the OEXIT subroutine (5.16.7). Its purpose is to start a read for a requested overlay routine from the direct-access device containing the overlay data set. On entry, register WA contains the subroutine return address, register BASE3 contains the address of an actual Overlay Area to be used, and register "R1" contains the address of the first of a group of one or more PCEs requesting the same overlay routine, chained from the first PCE by PCEOPCE.

OACEPCE of the Overlay Area is pointed to the first PCE. OACEPRIO and OACEOCON are set to indicate the routine which will reside in

the area. The Overlay Table entry for the requested routine is accessed and, if &DEBUG is set to YES, field OTBLODS is incremented by one.

The relative T and R in the overlay data set of the requested routine is obtained from the Overlay Table. The address of the Overlay DCT is loaded into register "R1". If the overlay data set is on any SPOOL volume (device type DA in the DCT), an absolute form of MTTR is computed and stored in DCTSEEK. This conforms to $EXCP requirements for SPOOL volumes (see 5.8) and allows $EXCP to remember SPOOL arm positions. If the overlay data set is on a non-SPOOL direct-access volume, the standard OS form of MBBCCHHR is computed and stored in IOBSEEK. See Section 6.1.2 for initialization of the Overlay DCT and data set.

Hardware read operation is requested by using the $EXCP macro. The Overlay DCT specifies that when the read operation is complete, Overlay $ASYNC Exit is to be entered. All PCEs chained from OACEPCE are already $WAITing OLAY, to be later activated by Overlay $ASYNC Exit (see 5.16.9). OLOD then returns to its caller or caller of OEXIT.

5.16.9 Overlay $ASYNC Exit - Program Logic

This routine is entered when under control of the Asynchronous Input/Output Processor ($ASYNC) PCE (see Section 4.8) an overlay read operation (started by OLOD subroutine, see 5.16.8) is posted complete. On entry, register "R1" points to the Overlay Area. BASE2 is set to the base value for the Overlay Roll Processor, which is used for local addressability. "R15" contains the return address to $ASYNC.

The chain of all users of the overlay routine just read (begins at OACEPCE, continues through PCEOPCE) is processed. Each PCE's re-entry address ("R15", now stored in PCER15) is absolutized by adding the address of the Overlay Area, if the value in PCER15 is determined to be relative. The address of the Overlay Area is also stored in each PCEBASE3, to provide addressability when the Dispatcher activates each Processor. The function $POST for OLAY is performed on each PCE to make it dispatchable.

If OS IOS has posted the read complete with a permanent I/O error, each PCE's (on OACEPCE chain) re-entry address (PCER15) is pointed to a routine which types the message "UNREADABLE OVERLAY - ..." and enters a permanent $WAIT. The Overlay Area is freed for other use.

If &OREPSIZ is set to zero, this Exit returns to $ASYNC. Otherwise, the Overlay REP storage area is examined to see if any REPs were read during HASP Initialization (see 6.4) which may apply to

this overlay routine. REPs whose CSECT name (last four characters) match OACENAME are applied. The assembly origin (OACEASMO) of the routine is subtracted from the REP address and the BUFSTART address of this Overlay Area is added, to determine the memory location to be patched.

Return is finally made to $ASYNC to allow other processing to continue. The Dispatcher will enter each Processor using the overlay routine just read.

(The remainder of this page intentionally left blank.)

6.0 MISCELLANEOUS

This section contains detailed internal information about miscellaneous routines imbedded in or involved with the HASP System and is intended primarily for use by systems programmers.

6.1 HASP INITIALIZATION

6.1.1 HASP Initialization - General Description

The purpose of HASP initialization is to initialize for HASP job processing. Initialization builds the required control blocks and makes modifications to the Operating System nucleus which allows HASP to monitor the execution of jobs.

HASP Initialization is designed to provide either a "cold" or "warm" starting capability. A "cold" start is one which starts the system anew. Only those jobs which are entered after a "cold" start will be processed. A "cold" start does not have any requirements as to configuration except as defined in the HASP generation parameters. A "warm" start is a restart. Checkpointed information is read from the SPOOL1 volume and the queued jobs and data from the last processing are recovered. This type of start requires, as a minimum, that the SPOOL volumes that were used during the previous execution be on-line. Extra SPOOL volumes, up to a total of &NUMDA volumes, may be added.

6.1.2 HASP Initialization - Program Logic

Initialization begins with the issuing of the HASP Supervisor Call which turns control over to the HASP Initialization SVC Routine. On return the HASP task will be in the Supervisor State with protect key of zero. Register 1 points to a list of resolved nucleus addresses and a return point for resetting HASP to problem state. These addresses are moved into the HASP Communication Table (HCT) for later use by the system.

Since HASP Initialization resides in the same area as the main HASP buffer pool as designated by the HASP parameter &NUMBUF and portions of the initialization routines are executed from overlay control sections, all HASP processors except those required for initialization and console processing are placed in the hold status. The command processor PCE is altered to refer to the Root Segment of HASP Initialization, which resides in the data portion of the first buffer used for HASP SPOOLING. The HASP Initialization WTOR is then displayed via OS WTOR facilities. Initialization then waits for the operator to respond with the desired options. The options are then compared against the Initialization Options table and the appropriate bits in the $OPTSTAT field in the HCT are set or reset in accordance with the options specified. If any option is incorrectly entered, an error message is issued and the $OPTSTAT field is set to the default option configuration. (Refer to STARTING THE HASP JOB Section of the HASP Operator's Guide.)

The HASP REP routine (described in Section 6.4) is entered for optional alteration of the resident portions of the Operating System or HASP (resident or overlay control sections).

Preparation Of Overlay Service

The Overlay DCT is prepared by indicating that it is in use, used only for reading, that Overlay $ASYNC Exit is to be entered on completion of any operation which was started by using Overlay DCT, and that Overlay Roll Processor is the owner of the DCT.

The overlay data set is described by a DD card having ddname of OLAYLIB. DEVTYPE and OPEN macros are used to determine the number of tracks/cylinder of the overlay volume and data set extent, which is placed as the last (&NUMDA+1) extent in HASP's single multi-extent direct-access DEB. The overlay data set is closed, since HASP uses its own constructed I/O control blocks.

The overlay data set UCB address is stored in a table used to withdraw or abort HASP and the UCB is made allocated, permanently resident, and private.

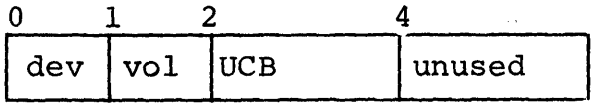
The number of tracks/cylinder and extent are used to compute a beginning absolute TT of the overlay data set, which is stored in the Overlay DCT for later use by the OLOD subroutine (see Section 5.16.8).

Locating Spool Volumes

All OS UCBs are searched via the UCB lookup table and direct-access volumes with volume serials of SPOOLx are examined for use for HASP SPOOL volumes. As each device is examined, the UCB is allocated by turning on the private, reserved, permanently resident, and allocation indicators. The UCB locations and sixth volume serial character are saved in a temporary workarea for later reference. If during the UCB search multiple volumes with the same serial or too many SPOOLx volumes are found, an error message is displayed, SPOOL volume UCBs are deallocated and the HASP job is terminated. Upon completion of a successful allocation of SPOOL volumes, control is passed to Direct-Access Initialization.

Direct-Access Initialization

Direct-Access Initialization (NGDAINIT) gains control after all Spool devices have been found by initialization; initialization has built a table of six-byte entries (NSPOOLL1) describing the direct-access devices upon which Spool disks are mounted, of which each entry appears as follows:



where:

dev is the low-order byte of the direct-access device type;

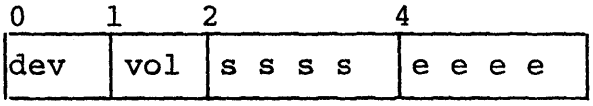
vol is the low-order byte of the volume serial number; and

UCB is the device's UCB address.

Before checking for warm start, NGDAINIT establishes where the checkpoint record is to be placed on SPOOL1. To do this, it first calls the DEB/TED setup routine to establish certain statistics about all mounted Spool volumes and then issues an OBTAIN macro-instruction for SYS1.HASPACE on SPOOL1. The checkpoint information will reside on the first track of this data set (the first two tracks if &JITSIZE is not zero); accordingly, NGDAINIT sets up the necessary channel programs using the OBTAINed information.

WARM START

If the operator requested a warm start, NGWARM reads the checkpoint information directly into the area from which the checkpoint processor will write it; the information consists of the HASP job queue, the track group map, printer checkpoint information, miscellaneous status information (including direct access checkpoint information) and, optionally, the job information table (JIT). The direct access checkpoint information, $DACKPT, consists of &NUMDA six-byte entries of the following form:



where:

dev is the low-order byte of the direct-access device type;

vol is the low-order byte of the volume serial number;

ssss is the starting absolute track number of data set SYS1.HASPACE on the indicated SPOOL volume; and

eeee is the ending absolute track number of the first extent of data set SYS1.HASPACE on the indicated SPOOL volume.

For SPOOL1, the starting track number excludes the checkpoint tracks.

NGWARM insures that each volume specified in the direct-access checkpoint is mounted and, with the help of subroutine NGALLOC, that its extents are unchanged. If not all volumes are mounted, or if any extents have been changed, or if a cursory check of a volume shows that it is not properly formatted, NGWARM writes a message and sets a quit switch to cause HASP to quiesce.

If all volumes specified by the direct-access checkpoint are correct, NGWARM checks for (and formats if necessary) newly-mounted volumes. Then it again calls subroutine NGDEBSET to allow for the possibility that the order of Spool volumes in NSPOOLL1 (by unit address) may not have been the same as in $DACKPT; the final order is that of $DACKPT.

Now NGWARM relocates the HASP job queue, if necessary. The job queue as recorded in the checkpoint record contained main storage addresses; if HASP does not now occupy the same core locations as it did before, each main storage address in the HASP job queue (and in pointers to the job queue) must be adjusted to reflect the current main storage location of the job queue.

After relocation, NGWARM scans the job queue to check the busy bit of each active entry and to reset certain flags. If a busy bit is on, NGWARM turns it off and issues a WTO to inform the operator that the job was reading, executing, printing, or punching. Additionally, if the job was reading, NGWARM uses HASP queue management routine $QREM to delete the job's queue entry.

At the end of the job queue, NGWARM gives control to NGEXIT, which assembles and format-writes the checkpoint information; restores the HASP appendage table pointer in $DADEB1, the HASP multi-extent direct access DEB; counts the number of allocated track groups (one-bits) in the track group map; and gives control to NINITWTO.

COLD/FORMAT START

If the operator specified cold or format start, NGCOLD first zeros out the track group map. Then NGCOLD processes each mounted SPOOL volume.

For each volume, NGCOLD uses subroutine NGALLOC to process the DSCB for SYS1.HASPACE. This subroutine issues the OBTAIN macro-instruction to retrieve the DSCB; if OBTAIN's return code is not zero, an appropriate error message is printed via WTO. If the return code is

zero, NGALLOC computes and saves lower and upper absolute track numbers.

If NGALLOC operated normally, NGCOLD now tests for an operator specification of COLD; if the test is positive, NGCOLD calls subroutine NGREADCT to read and validate the count field of the first record of the last track of the first extent of SYS1.HASPACE on the volume. If the count field is invalid, or if the operator specified FORMAT, NGCOLD calls NGFORMAT to format the first extent. NGFORMAT issues an unconditional GETMAIN for core in which to build a formatting channel program and data, builds them, and formats each track by calling NGEXCP, which merely issues an EXCP and a WAIT and checks the post code.

After the volume has been inspected (and formatted if necessary), NGCOLD calls NGMAP to calculate the number of track groups in this volume and the track group number of the first track group. NGCOLD increments the overall number of track groups available for allocation by the quantity returned from NGMAP and then calls NGBITMAP which turns on in the master track group map the bits corresponding to available track groups on this volume. Then NGCOLD processes the next volume.

When all volumes have been processed NGCOLD refreshes certain checkpoint information (the HASP job queue, the print checkpoint information, and some miscellaneous checkpoint information) and gives control to NGEXIT, as above.

The DEB initialization subroutine, NGDEBSET, initializes certain HASP and OS control blocks and allows a great degree of SPOOL device independence.

When called, NGDEBSET first puts into $DADEB1 the address of the HASP TCB; it also changes the DEB appendage address to point to the standard IOS appendage. (The appendage address is restored by NGEXIT.) It checks for SPOOL1 and quiesces HASP if SPOOL1 is not found. Then NGDEBSET processes the Spool volumes.

For each volume, NGDEBSET calculates number of records per track using information from the device characteristics table IECZDTAB in the OS nucleus and the formula given with the DEVTYPE macroinstruction in the OS System Programmer's Guide. Then it sets up certain information in an entry of the Table of Extent Data (TED).

Then, after setting the UCB address in $DADEB1, NGDEBSET performs the same functions for the remaining volumes and returns to the caller.

Activation of Overlay

If the overlay data set is contained on a SPOOLx volume, the Overlay Device Control Table is adjusted so that $EXCPs done by the OLOD subroutine (see 5.16.8) will use MTTR addresses and M which refers to the DEB extent for the SPOOLx volume rather than the overlay data set extent. The first Processor Control Element (PCE) in the HASP chain is connected to the OS save area chain and, with register 13 pointing to the first PCE, Initialization enters the HASP DISPATCHER as though the first processor had executed a $WAIT macro. The HASP DISPATCHER will run the PCE chain and dispatch the Initialization ROOT segment. The ROOT segment will $LINK to the first overlay control section HASPIOVA.

Unit Record Initialization -HASPIOVA

The OS UCBs are scanned for unit record devices. Devices which are on-line on a DUAL Processor Model 65 system, have OS scheduled I/O activity, or answer positively to a TIO instruction are considered real devices. Otherwise the devices are considered pseudo devices.

PSEUDO DEVICE INITIALIZATION - Pseudo devices are initialized by flagging the UCB for later identification by the HASP Execution Processor SVC 0 intercept routines and are varied on-line.

Pseudo 2540 reader, 1442 special forms punch, and 1443 special forms printer devices are especially noted and counts are maintained for the HASP Execution Processor Device Allocation Routine. Pseudo 1403 Printer UCS feature is removed from the UCB. Pseudo 2520 devices are identified and matched with an internal reader INTRDR Device Control Table which is initialized for processing.

REAL UNIT RECORD DEVICE INITIALIZATION - Each device is matched with a corresponding Device Control Table which is initialized for processing. If the device is allocated by OS, the DCT will remain in the drained status causing HASP not to use the device unless the operator starts the device by command. Automatic starting reader and (as appropriate) HASP console UCB attention index values are set to four allowing HASP to recognize the readying of the readers or the pressing of the enter key(s). (At least one HASP console device is reserved for a 1052 type of device.) If more real unit record devices of each particular type are found than available DCTs, an error message is displayed and the additional devices are ignored.

Control is then passed to the Remote Job Entry or console initialization routines as appropriate via a $XCTL macro.

Remote Job Entry Initialization - HASPIOVR

LINE INITIALIZATION - The OS UCBs are scanned for Synchronous Communication Adapter devices. The UCBs found are first matched with one or more DCT and corresponding line descriptions (LINEmm HASP Generation Parameters). Any DCT with a line description which specifically designates the UCB will be initialized for the UCB. If no line description designates the UCB, tests are made to determine if the adapter is physically on-line and, if so, a DCT with a line description with "***" specified will be located and initialized. Line devices will not automatically be started.

REMOTE DEVICE INITIALIZATION - Remote Device Control Tables are connected and initialized with information contained in the corresponding remote description (RMTnn HASP Generation Parameter). Each group of RMr.RDn,...,RMr.PRn,...,RMr.PUn,... for a given remote are chained together for control by the MULTI-LEAVING-line manager and RTAM. In addition the printer and punch DCTs are removed from the chain of all HASP DCTs and reinserted directly behind the reader DCT for the corresponding terminal. The device description is converted to internal flags and placed in each of the corresponding DCTs. If the line number is designated in the description the line DCT is located, DCTs are chained together, and flags are set to indicate non-signon remote.

The HASP Remote Job Entry Buffer Pool is initialized and control is passed to the remote console initialization routine or console (local) initialization routine as appropriate by $XCTL.

Remote Console Initialization - HASPIOVS

The Operator Message Space is allocated and control blocks are initialized. The Remote Console Processor PCE and a direct-access DCT are connected (the DCT is flagged IN USE). The origin of the first available track in the SYS1.HASPACE data set of the SPOOL1 volume and the base track address for operator message record allocation is set into the MSAMTTR field of the MESSAGE ALLOCATION ($MSALLOC) Table in the form: 0TT1 (TT is the first track available for messages). The number of records per track for the mounted SPOOL1 volume is inserted into the MSARPTRK field. If "cold" start was performed by direct-access initialization, the Cylinder map for SPOOL1 is altered to reflect the allocation of sufficient adjacent track groups starting with the group of the base track. The number of the last group is saved in the checkpoint records for future "warm" starts. If a "warm" start was performed by direct-access initialization, a check is made against the checkpoint record to insure that the space required is within the allocated space. Control is given to the console (local) initialization routine by $XCTL.

Console Initialization - HASPIOVB

OS CONSOLE INITIALIZATION - Information is extracted from the OS UCM and the Console processor is made ready for interfacing with OS.

HASP CONSOLE INITIALIZATION - The Console DCTs are initialized for HASP console support. Each DCT that was matched with a UCB by the unit record initialization routine is initialized for I/O processing. The corresponding authorization for each console is converted to console restrictions and set into the DCT. The operator command $S console is simulated.

Control is passed to the intercept initialization routine via $XCTL.

Intercepts Initialization - HASPIOVC

The following intercepts are made in accordance with the type of the HOST Operating System MVT or MFT and the HASP generation options as follows:

```
SVC  0 - EXCP interface used for control of user I/O
SVC  6 - LINK interface used to recognize events within OS
SVC  7 - XCTL interface used to recognize events within OS
         and to interface with OS console support
SVC 35 - WTO interface for console support
SVC 36 - WTL interface for write to log support.
```

Start Initiator, reader, and writer (optional) commands are issued which start the procedures contained on SYS1.PROCLIB. The reader will be directed to the pseudo device &RDR and the writer will be directed (if started) to the pseudo device &WTR.

In the event the HASP writer is selected in lieu of the OS writer, the HASP writer module "HASPWTR" is attached. If OS console support is selected, the HASP communications task is attached, via the attaching of module "HASPBR1" which enters the console processor.

Control is passed to the HASP buffer building routine via $XCTL.

Buffer Build - HASPIOVD

An OS variable GETMAIN is issued to obtain storage for the alternate buffer pool from the hierarchy as designated by the &BUFHICH generation parameter. The actual amount of core is reduced by the amount of reserved core (&RESCORE*1024) and used to determine the number of buffers which may be created in the alternate buffer pool along with the actual amount of storage needed. Extra core if any is released via OS FREEMAIN. The number of buffers which may be created in the alternate pool is compared against the expression of generation parameters:

&MINBUF - &NUMBUF where &MINBUF ≥ &NUMBUF = number of buffers in the main buffer pool

If the alternate buffer pool will not contain at least the number of buffers specified by the expression a warning is issued.

The origin of the main and alternate buffer pools are examined to determine which has the lower storage address. The pool with the lowest address is created and chained to the $BUFPOOL chain of buffers. The pool remaining is then created and chained to the end of the first.

The operator is then asked to ENTER HASP REQUESTS (optional) and the ROOT segment of HASP INITIALIZATION is entered via the $RETURN macro.

Activation Of Normal Processing

The ROOT SEGMENT returns the PCE to the Command Processor and, if the operator specified REQ in the WTOR, enters the Command Processor. If NOREQ was specified by the operator, all HASP Processors are $POSTed and the Command Processor is entered.

6.2 HASP INITIALIZATION SVC ROUTINE

6.2.1 HASP Initialization SVC Routine - General Description

This program is a Type-I SVC routine which resides in the Operating System Nucleus and provides the following basic functions:

1. For HASP:

   - To give HASP a zero storage protection key.
   - To place HASP in supervisor state.
   - To return the address of key symbols in the nucleus which are required for HASP processing.
   - To guard against recursive entries in order to prohibit multiple copies of HASP from being initiated.
   - To provide the address of an entry which will cause the SVC routine to be reset for HASP withdrawal and cause the PSW to be reset to its initial value.

2. For the HASP Reader/Interpreter Appendage:

   - To place the HASP JCL Exit routine in supervisor state.
   - To return the left half of the PSW which was in use when the SVC was invoked.

3. For the non-HASP program:

   - To give an indication to any other program as to whether HASP is currently active or not.

6.2.2 HASP Initialization SVC Routine - Program Logic

This program is a Type-I OS SVC routine. It must be link-edited with the nucleus to resolve the external address constants required for HASP processing.

Upon entry, register 1 is compared with the EBCDIC characters "HASP". If the register does not compare, a condition code is returned to the user in register 15 as follows:

R15 = 0 - HASP has not been initiated and is not currently active.

R15 ≠ 0 - HASP has been initiated and is currently active.

If register 1 contains "HASP", a test is made to determine if HASP has been invoked. If not, then this switch is set to indicate that HASP is now active and the left half of the PSW is saved for the "reset entry".

If HASP has been invoked, the protect key of the caller is interrogated. If this protect key is non-zero, the caller is ABENDed with an appropriate ABEND code.

The SVC OLD PSW is modified so that the return to HASP will place HASP in the supervisor state and give HASP a zero storage protection key. Register 1 is then loaded with the address of a table of address constants of key nucleus addresses and return is made through the OS SVC FLIH. At this time register 0 contains the left half of the PSW which was in use when the SVC was invoked.

One of the addresses in the nucleus address table is the address of the SVC reset routine. When this routine is entered, it resets the switch to indicate that HASP is no longer active. It then returns to the user by loading a PSW constructed by concatenating the left half of the original PSW with register 14.

6.3 HASP OVERLAY BUILD UTILITY

6.3.1 HASP OVERLAY BUILD - GENERAL DESCRIPTION

The purpose of this program is to process the object deck output from the ten primary HASP assemblies. Overlay CSECTs are extracted and written (each as a single record) to the sequential overlay data set (ddname OLAYLIB), all references to overlays from resident and overlay routines are resolved, and all resident CSECTs (even if programmed as overlayable) are passed to the OS Linkage Editor in a sequential data set (ddname SYSLIN). Optional control cards are processed which allow changing the status of any overlayable CSECT from actual overlay to permanently resident and vice-versa.

The use of this program to install HASP (control cards, listings produced, etc.) is described in section 10.2.2.3, which should be read as background to this description. Overlay Services and Roll logic, and Overlay Programming Rules are described in sections 5.16, 4.20, and 12.14 respectively.

6.3.2 HASP OVERLAY BUILD - PROGRAM LOGIC

On initial entry, the time is sampled. A truncated "time-like" value is saved. This value will be placed into one resident CSECT and one overlay CSECT. During HASP Initialization, if these two values do not match, an error message is produced and HASP terminates.

All data sets are OPENed and the listing title line is printed. If the control card data set is present (ddname SYSIN), cards are read, printed, and processed until end-of-file is encountered. Each card contains an overlayable CSECT name beginning in column 1, which must begin with "HA$". A SYM table entry is made for each such name. An Ocon (index into the Overlay Table, HASPOTAB) is assigned and a priority, if present in column 16 of the card, is remembered. This information is later used to override the normal processing of that CSECT, when encountered in the object decks. A listing header line is printed at the end of control card processing.

All objects decks are processed as a single sequential input data set (ddname SYSOBJ). Only the four object card types ESD, TXT, RLD, and END; as documented in OS/360 Loader PLM, Y28-6714, Figures 30-34; are processed. All other cards are written directly to SYSLIN. If an object card with a valid ESID number greater than the program's table limits (internal assembly variable &MAXESID) is encountered, the program abends with a U0101 code.

ESD card processing is essentially the construction of two Tables from ESD information. The SYM table contains the names of and information about any external names under overlay control (i.e. beginning with "HA$"). It is a global table covering all object decks together. A name is entered when a reference to it or CSECT definition of it is first encountered, or during control card processing as previously described. An overlay name in an ESD card item is first searched for in the SYM table, and if found, changes are made to the existing entry. An error message is produced for each duplicate definition of a previously defined overlay CSECT name and only the first definition is used. An Ocon is assigned to each entry. When a name becomes a defined CSECT, if the fourth character is "O", the overlay routine is actually to be made disk resident, and storage is assigned to load its text.

The ESID table is cleared at the beginning of each object deck and constructed as ESD items are encountered, under control of SYM table contents. It is a table of words, in order by ESID number. TXT and RLD card processing access this table only. It contains relocation values, Ocons, and flags controlling the disposition of text and RLD items.

ESD items, for references to overlays or for definitions of overlay CSECTs which are to be disk resident or are duplicated, are eliminated from an ESD card when processed, before the ESD card is written to SYSLIN. This elimination is done by changing them to type NULL or, if type LD, by physically removing them and compacting the card.

TXT card processing has three possible results. Text belonging to an actual overlay is loaded into memory, subject to relocation according to storage assigned by ESD processing. Text of any overlay CSECT which is a duplicate of one encountered previously is discarded. Text of non-overlay CSECTs or overlays being made permanently resident is written un-altered to SYSLIN.

RLD card processing concerns individual RLD items, as follows. If an item applies to a discarded duplicate overlay CSECT, it is eliminated. If an item references a non-overlay CSECT, it is left un-altered. An overlay reference item describes a 2 byte Q type constant assembled in the expansion of the $LINK, $LOAD, $XCTL, and $OCON macros. The reference is resolved by substituting the Ocon value assigned to the referenced overlay routine, and the item is eliminated. If the Q constant exists in an actual overlay routine, the Ocon value is simply moved to the proper address of the text already loaded in memory. If the Q constant exists in a non-overlay CSECT or overlay being made resident, a new TXT card containing the Ocon value is created and written to SYSLIN. Eliminated items are physically removed and the RLD card compacted before writing to SYSLIN.

END card processing is really end-of-object-deck processing. The card is written unchanged to SYSLIN. The entire SYM table is then scanned for selected processing. Each actual overlay whose text was loaded from the most recent object deck is written to OLAYLIB as a fixed length record of length &OLAYSIZ (internal assembly variable set to 1024 bytes in unmodified HASP). A listing line is printed for each overlay CSECT defined in the most recent deck, with its length and assigned OCON value. Priority and disk address in two forms are printed for actual overlays. An error message is printed if an actual overlay length exceeds &OLAYSIZ.

Processing of multiple object decks continues as above until end-of-file for SYSOBJ is signalled. The entire SYM table is then processed to produce the Overlay Table, which is written to SYSLIN as a new object deck (did not exist in the input) containing a single resident CSECT, HASPOTAB. An error message is printed for any name in the SYM table which is still not defined as a CSECT.

Each entry in HASPOTAB is 4 bytes or, if &DEBUG is set to YES, 12 bytes. The last 4 characters of the CSECT name are included if entries are 12 bytes, to facilitate identification in a memory dump. If a routine is actual overlay (disk resident), the TR (relative form) of disk address and the priority are placed into the table entry for that routine. If an overlay routine was written to SYSLIN by previous processing (to become permanently resident in the HASP load module), a V type constant is created in its table entry. An appropriate RLD item referencing the CSECT name is created.

When HASPOTAB is complete, an END card for it is written to SYSLIN, all data sets are CLOSEd and the program terminates. Completion code 0 is returned normally, 4 if duplicate CSECTs were encountered, and 8 if any overlays were too long or undefined.

6.4 HASP REP ROUTINE

This routine gives the systems programmer the capability of applying absolute or relocatable value patches to HASP, at absolute or relocatable memory addresses, as part of the HASP Initialization process.

6.4.1 REP Card Format

Columns	Contents
1	Any identification - ignored by REP routine
2-5	CSECT name, "REP", or "ABS"
6	Blank
7-12	Address at which to apply patch (6 hex digits) or blank
13-16	Blank
17-blank	Half word absolute value patches, 4 hex digits each, separated by commas, patch data terminated by first blank, or one full word (8 hex digit) relocatable value patch, followed by a comma and the name of the resident CSECT which defines the relocatable part of the value

The above format allows patches to be applied at any absolute memory location (by use of REP or ABS beginning in column 2) or at addresses in HASP CSECTs (resident or overlay), subject to relocation. Relocatable addresses should be taken directly from a HASP assembly listing containing the CSECT to be patched. A blank address field is interpretted as one greater than the last address patched by the previous card, but the card will be used only if columns 2-5 match those of the previous card.

The patches may be absolute values or one relocatable word per card, whose value is relative to any resident HASP CSECT. Relocatable values should be punched as if they were the assembled value of an A type constant in the CSECT which defines the referenced relocatable symbol.

Use of the term "CSECT name" in the above description means the fifth and following characters of a HASP CSECT name, as taken from the External Symbol Dictionary of a HASP assembly listing.

A deck of one or more REP cards should be terminated by a card having "/*" punched in columns 1-2.

6.4.2 REP Routine - Program Logic

REP cards, as described in Section 6.4.1, are read from the card reader, whose address is given by the HASPGEN parameter $REPRDR, immediately after the operator replys to HASP's initial WTOR, if the operator specifies "REP" in the reply options. Each card is listed on the printer, whose address is given by the HASPGEN parameter $REPWTR, unless the operator specifies "NOLIST" in the reply options. All I/O is performed using CPU instructions SIO and TIO with the CPU disabled for all interruptions. Cards are read and processed until a card having "/*" in columns 1 and 2 is encountered or until the card reader signals unit exception.

The value or data portion of each card is processed first. If the value is relocatable (indicated by comma in column 25), eight hex digits beginning in column 17 are converted to a binary value. The CSECT name (last four characters beginning in column 26) is located in an internal table of standard resident module names. A value is taken from this table which is the memory address at which the resident module is loaded. This value is added to the value taken from the card.

If the value portion is absolute, groups of four hex digits (separated by commas) beginning in column 17 are converted to binary values until a blank is encountered instead of an expected comma. The values are concatenated to form a single variable length binary value.

The address portion of the card is processed next. If non-blank, six hex digits beginning in column 7 are converted to a binary address. An attempt is made to locate the to-be-patched CSECT name (last four characters beginning in column 2) in the standard resident module name table. If located, the loaded memory address of the resident module is added to the address taken from the card. If the CSECT name is not in the standard resident module name table, the overlay table is searched to determine if the CSECT is an overlay which was made permanently resident. If so, the non-zero assembly origin of the overlay CSECT is subtracted from and the loaded memory address is added to the address taken from the card. In both of the above cases, the patch value as previously computed is applied by moving it to the memory address determined by one of the two methods described.

If the CSECT name is not located by either search just described, it is assumed to be an overlay CSECT which is not permanently resident. The name, unrelocated address, and value are saved in a reserved area, to be applied each time the overlay is read from direct access during HASP operation.

If the address field of the card is blank, the to-be-patched CSECT name is compared with that from the preceeding card. If they are not equal, the card is ignored. Otherwise, the card is considered

to be a continuation of the preceeding card and the patch value is applied at the next higher memory address or saved as appropriate.

If no area was reserved to save patch information for application to non-resident overlays (HASPGEN parameter &OREPSIZ=0) or if the capacity of the reserved space is exceeded, the operator message "OVERLAY REPPING ERROR" is issued and HASP operation is abortively terminated.

6.5 HASP ACCOUNTING ROUTINE

6.5.1 HASP Accounting Routine - General Description

The Accounting Routine accumulates statistics for each job at the completion of the Punch phase and produces the HASP Account Card (see Section 11) which is punched by the Punch Processor. This feature is optional and may be deleted at HASPGEN time.

6.5.2 HASP Accounting Routine - Program Logic

The HASP Accounting Routine is a separately assembled overlay segment which gains control at the end of the Punch phase. Its function is to construct an accounting card such that the Punch Processer can punch this card upon return.

Upon entry, the following registers contain the following information:

Register 1 - Address of the HASP Job Queue Entry Priority Byte.

Register 2 - Address of the Accounting Card Image Area.

Register 10 - Address of the HASP Job Control Table.

Register 14 - Return Address.

All registers must be saved and restored before return to HASP.

This routine blanks out the Accounting Card Image Area and then extracts information from the HASP Job Control Table and HASP Job Queue Entry and constructs the Accounting Card Image in the Accounting Card Image Area. Special consideration is made for the clock passing midnight. In such cases the elapsed time is negative and a correction factor (24 hours) must be added.

The accounting card which is normally punched when this routine returns to the punch processor may be deleted by setting the condition code to zero before returning.

6.6 HASP DUMP ROUTINES

HASP provides two dump routines which are optionally included as debugging aids. The HASP Dump Routine for printer formatted dumps and the HASP High Speed Dump to Tape Routine are discussed in the following sections.

6.6.1 HASP Dump Routine - General Description

The $Dump routine is available as a debugging aid (effective only if &DEBUG=YES) and will, when the console PSW RESTART key is depressed, dump memory according to specified limits.

6.6.2 HASP Dump Routine - Program Logic

This routine gains control via the PSW RESTART key on the console. Upon activation of the key, a specially formatted HASP PSW is loaded from location HEX'0'. The format is HEX'0004000F' for the first word; where 0004 is the mask that allows only a machine check interrupt, and 000E is the address of the printer that is referenced in the routine. The second word will contain the address of the $Dump routine of HASP.

Once activated, $Dump will reference the low core address of HEX'34' for its beginning limit. This limit defaults to a value that will dump all of the memory unless the operator changes the limit prior to pushing the PSW RESTART key. If a change is desired, the limit should be entered at its location in the following format: HEX'00XXXXXX'. Also, if an operator should care to change the limit while $Dump is activated, the routine will immediately note a change, will immediately stop the previous dump, and will start dumping memory from the new limit. It should be noted at this point that a machine check will destroy the limit values within their position in core. To avoid undetected machine checks, however, the dump program is, at all times, enabled for machine check interrupts.

The routine also allows the operator to route the printing to a printer with an address other than HEX'00E'. A change of the printer address in the HASP preformatted PSW, prior to activation of $Dump, will accomplish this.

At the normal end of the routine, the system will be placed in a "wait" state via the LPSW command. At this point in time, the registers will have been restored back to their values prior to $Dump and the default limits of $Dump will have been returned to their respective values.

6.6.3 HASP HIGH SPEED DUMP TO TAPE ROUTINE - GENERAL DESCRIPTION

The High Speed Dump to Tape Routine, available as an optional debugging aid, dumps all of main storage to tape for post processing by the IBM System/360 Operating System Service Aids program IMDPRDMP or an equivalent processor.

6.6.4 HASP HIGH SPEED DUMP TO TAPE ROUTINE - PROGRAM LOGIC

ENTRY TO IDMTAPE - If the system programmer sets the HASPGEN parameter &DMPTAPE to the address of an attached magnetic tape drive the High Speed Dump to Tape Routine (IDMTAPE) will be created to write on the specified drive when entered. Initialization will display on the operator's console the message "SET RESTART PSW TO 0004000000aaaaaa FOR TAPE DUMP" where "aaaaaa" is the address of the entry point IDMTAPE. This message not only verifies that the routine is present; it is sufficient information for the operator to manually activate the dump. Via the REP processing routine or via manual key entries the system programmer or operator is able to insert code within the system to detect errors and cause dumps by program entry to the routine simulating the loading of the requested PSW. (Example: set program new PSW as directed to dump on the first program check.)

IDMTAPE assumes that the device generated is an appropriate tape drive to use, that the tape is at load point ready for writing, and that the recording mode status is correctly set. If these conditions are not true unpredictable results will occur.

CHANGING THE TAPE ADDRESS - The halfword located in storage at IDMTAPE-4 contains the address of the tape drive upon which the routine will write when entered. This address may be altered manually to any other tape drive address as appropriate prior to executing the routine.

PROGRAM LOGIC - The general registers and selected fixed storage areas are saved in location 84 hexadecimal to be compatible with the OS Service Aids dump program.

HASP control section locator elements are moved into low storage adjacent to the fixed area information. These elements contain the last four characters of the "HASPxxxx" control section names of basic assembly modules. Following each CSECT identification is the address of the beginning of the identified CSECT. (HASP control sections may then be easily located in the dump after post processing.) Location 80 hexadecimal

is set to the negative of address 2048. Location 80 hexadecimal for 4 bytes is written followed by 2048 bytes of storage (first part of which contains saved data).

Each succeeding record is written by adding the address 2048 to the address in location 80 (hex), storing the memory protect key in the high byte, and writing 2052 bytes of storage (four from location 80 (hex) and 2048 from the designated address). When the address is greater than zero the program new PSW is set to provide an end of storage exit which, when entered, will cause the writing of EOF, a rewind unload, and the loading of a wait state PSW.

7.0 HASPGEN AND RMTGEN PARAMETERS

This section describes the parameters used to specify the HASP System, HASP MULTI-LEAVING Remote Terminal Programs, and the System/360 Model 20 STR Remote Terminal Program.

Generation of the HASP System is called HASPGEN, and generation of the HASP MULTI-LEAVING Remote Terminal Programs and the System/360 Model 20 STR Program for HASP Remote Job Entry is called RMTGEN. Both generation processes are described in Section 10.

7.1 HASPGEN PARAMETERS

Generation of a HASP System involves specification of certain parameters, called HASPGEN parameters. With these parameters, the installation system programmer specifies the characteristics of the System/360 or System/370 with which he will use HASP and the optional HASP features he wishes to be included in the generated HASP System.

The following pages describe the HASPGEN parameters. For each parameter there is an explanation, the default value, and frequently notes which expand upon the explanation and refer to related HASPGEN parameters.

The HASPGEN parameters are given in alphabetical order (neglecting the first character if it is & or $) except for parameter $$x, which appears last.

&ACCTNG

Explanation: Variable symbol &ACCTNG specifies the HASP job accounting option. If it is specified as YES, HASP will call the HASP accounting routine and punch a HASP accounting card for each job processed by HASP. The specification must be either YES or NO.

Default: &ACCTNG=YES

Notes:
1. The HASP accounting routine and the HASP accounting card are discussed in other sections of this manual.
2. If &NUMPUNS=0, parameter &ACCTNG should be set to NO.

&AUTORDR

Explanation: Variable symbol &AUTORDR specifies the inclusion or exclusion of code in HASP to recognize automatically when a physical card reader available to HASP becomes ready. The specification must be either YES or NO.

Default: &AUTORDR=YES

Notes:

1. If &AUTORDR=NO, HASP's physical card readers remain in the INACTIVE state when they become ready; the operator must issue a $SRDRn command to cause HASP to begin reading cards from READERn.

&BSCCPU

Explanation: Variable symbol &BSCCPU specifies inclusion or exclusion in the HASP Remote Terminal Access Method of support for HASP MULTI-LEAVING Remote Job Entry.

Default: &BSCCPU=NO

&BSC2770

Explanation: Variable symbol &BSC2770 specifies inclusion or exclusion in the HASP Remote Terminal Access Method of Remote Job Entry support for the 2770 Data Communication System. The specification must be either YES or NO.

Default: &BSC2770=NO

&BSC2780

Explanation: Variable symbol &BSC2780 specifies inclusion or exclusion in the HASP Remote Terminal Access Method of Remote Job Entry support for the 2780 Data Transmission Terminal. The specification must be either YES or NO.

Default: &BSC2780=NO

&BSC3780

Explanation: Variable symbol &BSC3780 specifies inclusion or exclusion in the HASP Remote Terminal Access Method of Remote Job Entry support for the 3780 Data Communications Terminal. The specification must be either YES or NO.

Default: &BSC3780=NO

&BSHPRES

Explanation: Variable symbol &BSHPRES specifies inclusion or exclusion in the HASP Remote Terminal Access Method of support for the Space Compression/Expansion feature of 2770 and 3780 terminals. The specification must be either YES or NO.

Default: &BSHPRES=NO

Notes:
1. This support must be included if any terminal will transmit to HASP using the Space Compression/Expansion feature.
2. Use of this support for output to any terminal is controlled by specification in the RMTnn parameter for that terminal.

&BSHPRSU

Explanation: Variable symbol &BSHPRSU specifies inclusion or exclusion of the HASP Remote Job Entry Printer Interrupt feature for binary synchronous hardware terminals. If this feature is included, the Remote Terminal operator may interrupt printing to transmit jobs or HASP commands to HASP. The specification must be either YES or NO.

Default: &BSHPRSU=YES

Notes:

1. If &BSHPRSU=YES, HASP will recognize certain control characters from the binary synchronous hardware terminal which indicate that the printer has stopped. The HASP Remote Terminal Operator's Manual for hardware terminals contains more information.

&BSHTAB

Explanation: Variable symbol &BSHTAB specifies inclusion or exclusion in the HASP Remote Terminal Access Method of support for the Printer Horizontal Format Control feature of 2770, 2780, and 3780 terminals. The specification must be either YES or NO.

Default: &BSHTAB=YES

Notes:
1. Use of this support for any terminal is controlled by specification in the RMTnn parameter for that terminal.

$BSPACE

Explanation: Ordinary symbol $BSPACE specifies the character which will be interpreted as the 360 hardware defined backspace character X'16'. The $BSPACE character when entered on any OS controlled system operator console will be removed from the command text along with the previously entered character (if any). Characters following the $BSPACE character will be shifted left to replace the removed characters. The $BSPACE edit is performed on all commands entered via OS system operator command input sources regardless of its position within the text of the data entered. $BSPACE is active only for HASP Systems using the &NUMCONS=0 option and does not apply to HASP card reader or remote workstation sources. $BSPACE is specified using the two hexadecimal digit representation of the EBCDIC character.

Default: $BSPACE=5F

Notes:
1. The default specification indicates that the EBCDIC character "¬" is to be used to backspace command entry on OS controlled system operator consoles.
2. The character selected for the backspace function must be chosen with extreme caution since it eliminates the use of that character (except as a backspace operation) in all commands and replies to WTORs.

&BSVBOPT

Explanation: Variable symbol &BSVBOPT specifies inclusion or exclusion in the HASP Remote Terminal Access Method of code to recognize an EM (End of Media) punch in card images transmitted nontransparently by the 2780 Data Transmission Terminal. The specification must be either YES or NO.

Default: &BSVBOPT=NO

&BUFHICH

Explanation: Variable symbol &BUFHICH specifies the storage hierarchy for which HASP initialization will issue a GETMAIN in an attempt to get more than &NUMBUF buffers for HASP. The specification must be either 0 or 1.

Default: &BUFHICH=1

Notes:
1. &BUFHICH has meaning only with OS storage hierarchy support.
2. See HASPGEN parameters &NUMBUF, &MINBUF and &RESCORE for additional information concerning the use of this parameter.

&BUFSIZE

Explanation: Variable symbol &BUFSIZE specifies the size in bytes of each HASP buffer. If the value specified is not a multiple of eight, HASPGEN will adjust it upward to a multiple of eight. The specification must be an integer not larger than the track size of any SPOOL device and not smaller than the number given by

300+2*&NUMDA*&NUMTGV/8+5*S+8*F

where S = maximum allowable number of SYSOUT specifications per job, including special forms requests
F = maximum allowable number of special forms requests per job.

Default: &BUFSIZE=688

Notes:

1. The above formula is the approximate size of a HASP control block called the JCT. &BUFSIZE is the data length of each physical record on the portion of a SPOOL volume used by HASP, and JCTs are written on SPOOL volumes. A JCT's initial size is about 300 + 2*&NUMDA*&NUMTGV/8; it increases in size as a job is being executed. When HASP recognizes a job step change, the JCT is increased in size by five bytes for each non-special-forms SYSOUT data set, or by thirteen bytes for each special-forms SYSOUT data set, that was written upon since the previously-recognized step change. But if an increase of five (or thirteen) bytes would cause the JCT size to become greater than &BUFSIZE, HASP instead writes to operator the message
JCT OVERFLOW--OUTPUT LOST
for the SYSOUT data set, and its output is lost.
2. The default &BUFSIZE of 688 is optimized for the 2314 track size. If &NUMTGV and &NUMDA are left at their default values of 400 and 2, the &BUFSIZE default allows a maximum of 37 (no special forms) and a minimum of 14 (all special forms) SYSOUT data sets per job. A good value of &BUFSIZE optimized for the 3330 would be 736.
3. &BUFSIZE must be 416 or greater if &XBATCHC is altered from its default null value.
4. &BUFSIZE must be 536 or greater if 3211 printers are used by HASP.

$CKPTIME

Explanation: Ordinary symbol $CKPTIME specifies the interval, in seconds, at which certain HASP information will be checkpointed for warm start purposes.

Default: $CKPTIME=60

Notes:
1. The time interval specified is a maximum. Checkpoints are also taken when a job changes its status in the HASP job queue.
2. The section of this manual describing the HASP checkpoint processor describes the checkpoint information.

&CLS(n)

Explanation: Subscripted variable symbols &CLS(n) specify HASP job classes. The nth HASP logical partition may select for OS execution a job from the HASP job queue only if the job's class (specified by the user in the CLASS=parameter of the JOB card, or defaulted to A) is one of the characters specified in the &CLS(n) parameter or specified by the operator in the set command, $T Imm,list (where &PID(n)=mm). Each specification must be a 1- to 8-character string of valid HASP job classes. The same HASP job class may be specified in two or more specifications.

Default:
```
&CLS(1)=A
&CLS(2)=BA
&CLS(3)=CBA
&CLS(4)=DCBA
&CLS(5)=EDCBA
&CLS(6)=FEDCBA
&CLS(7)=GFEDCBA
&CLS(8)=HGFEDCBA
&CLS(9)=IHGFEDCB
&CLS(10)=JIHGFEDC
&CLS(11)=KJIHGFED
&CLS(12)=LKJIHGFE
&CLS(13)=MLKJIHGF
&CLS(14)=NMLKJIHG
&CLS(15)=ONMLKJIH
```

Notes:
1. Only the first &MAXPART specifications, &CLS(1) through &CLS(&MAXPART), will be used.
2. If &MAXCLAS is specified less than 8, only the first &MAXCLAS characters of each specification &CLS(n) will be used.

&CONAUTH

Explanation: Variable symbol &CONAUTH specifies the HASP command authorization of each of the eight possible HASP physical consoles when HASP console support is included in the generated HASP system. The specification must be a string of up to eight numeric digits, each of which may range from 0 to 7 and is the sum of the desired authorizations for its respective console (leftmost digit for CONSOLE 1, next digit for CONSOLE2, etc.) as follows:

1 - System Control (including OS) Commands
2 - Device Control Commands
4 - Job Control Commands.

Default: &CONAUTH=77777777

Notes:
1. Any HASP console's command authorization may be changed from a console with System Control Command authority.
2. If &NUMCONS=0, parameter &CONAUTH is not used.
3. All HASP consoles are authorized for the issuance of Display Commands.

&DEBUG

Explanation: Variable symbol &DEBUG specifies inclusion or exclusion of debugging code in the generated HASP system. The specification must be either YES or NO.

Default: &DEBUG=NO

Notes:
1. The &DEBUG option is independent of the &TRACE option.
2. If &DEBUG is specified as YES, HASP includes, in addition to other debugging code, a core dump routine.

$DELAYCT

Explanation: Ordinary symbol $DELAYCT specifies a delay factor to be applied by the HASP Remote Terminal Access Method when transmitting to a Multi-Leaving System/360 Model 20 Submodel 2 or 4 Remote Terminal over a high-speed (19,200 baud or greater) teleprocessing line, to avoid the possibility of certain line errors. The specification must be an integer greater than zero. Recommended values for some central CPUs are:

Model 195	8000
Model 91	4000
Model 85	3000
Model 165	3000
Model 75	500
Model 65	256
Model 155	256
Model 50	100
Model 145	100
Model 135	50
Model 40	1

Values for other CPUs may be interpolated based on CPU speed.

Default: $DELAYCT=256

&DMPTAPE

Explanation: Variable symbol &DMPTAPE specifies a unit address to be used with the HASP dump program. The specification must be a valid unit address. If the specification is not 000, the address is assumed to be that of a tape drive and the generated HASP system will include a dump-to-tape program for that tape drive. The tape produced by this program may be printed using FE Service Aid IMDPRDMP.

Default: &DMPTAPE=000

Notes:

1. This parameter does not affect inclusion of the HASP dump-to-printer program, which is always assembled if &DEBUG=YES. To use the dump-to-tape program the operator must set location 0 as indicated in an operator message produced by HASP initialization, make the tape drive ready for writing, and push PSW RESTART.
2. This facility is provided as an aid to the system programmer and may not operate correctly in all environments or with other than the IMDPRDMP program provided with Release 19 of OS. No maintenance will be provided for this facility. Installations unable to utilize this support as distributed should utilize the service aid IMDSADMP to produce dump tapes.

$ESTIME

Explanation: Ordinary symbol $ESTIME specifies the default estimated execution time, in minutes, for a job. The specification must be an integer greater than zero.

Default: $ESTIME=2

Notes:
1. If a user does not specify in the accounting field of his job card a value for estimated execution time, the value $ESTIME is used.
2. All timings performed by HASP are in real time. The timing for estimated execution time begins when HASP allows its OS Reader/ Interpreter to start reading the job.

$ESTLNCT

Explanation: Ordinary symbol $ESTLNCT specifies the default estimated print line count, in thousands of lines, for a job. The specification must be an integer greater than zero.

Default: $ESTLNCT=2

Notes:
1. If a user does not specify in the accounting field of his job card a value for estimated print line count, the value $ESTLNCT is used.

$ESTPUN

Explanation: Ordinary symbol $ESTPUN specifies the default estimated punched card count, in cards, for a job. The specification must be an integer greater than zero.

Default: $ESTPUN=100

Notes:
1. If a user does not specify in the accounting field of his job card a value for estimated card count, the value $ESTPUN is used.

&FCBV

Explanation: Variable symbol &FCBV specifies inclusion or exclusion of the 3211 Variable Forms Control Buffer loading capability. If set to YES, the "V" specified FCB image is generated and the code to support the $TF... command is included. The specification must be either YES or NO.

Default: &FCBV=NO

(The remainder of this page intentionally left blank.)

&INITSVC

Explanation: Variable symbol &INITSVC specifies the SVC number of the HASP SVC. The specification must be an integer between 200 and 255, inclusive.

Default: &INITSVC=255

Notes:

1. The specification for &INITSVC must correspond to a use at SYSGEN time of the OS SYSGEN macro-instruction SVCTABLE. The HASP SVC is a type-1 SVC and must be included in the OS Nucleus before HASP can be used. After HASPGEN has completed, the object deck for the HASP SVC is member HASPSVC of partitioned data set SYS1.HASPOBJ. For further discussion, see the section on installing HASP.

&JITSIZE

Explanation: Variable symbol &JITSIZE specifies the number of bytes per entry of the HASP job information table. The specification must be an integer greater than or equal to 8, or equal to 0, but never so large that the value

&MAXJOBS*&JITSIZE

is greater than the track size of the device upon which SPOOL1 is mounted.

If &JITSIZE=0, the HASP commands with OS job names as operands will be inoperative.

Default: &JITSIZE=0

Notes:
1. The recommended specifications for &JITSIZE are 0 and 8. If &JITSIZE is set greater than 8, the additional space generated in the job information table will not be used by HASP.
2. If &JITSIZE is specified greater than zero, the job information table will be checkpointed on SPOOL1 whenever it changes.

$LINECT

Explanation: Ordinary symbol $LINECT specifies the default maximum number of lines to be printed per page of a job's printed output.

Default: $LINECT=61

Notes:

1. If a user does not specify in the accounting field of his job card a value for line count, the value $LINECT is used.

2. Setting $LINECT=0 will cause automatic page overflow, normally provided by HASP, to be suppressed unless overridden by the job card accounting parameter (see note 1).

3. If a print data set is generated without any ejects (no skips to any channel in the carriage tape), and if $LINECT=0 or if the "linect" parameter on the job card of the job producing the data set is zero, then that data set will be treated as one page when forward spaced, backspaced, interrupted, or warm started while printing.

LINEmm

Explanation: Ordinary symbols LINEmm specify the characteristics of teleprocessing lines to be used by HASP Remote Job Entry. Lines must be defined consecutively, starting with LINE01. Each specification must be a 5-character string of the form

LINEmm=aaalc

where the letters represent the following:

Code Letters	Range	Description
mm	01-99	Line Number
aaa	000-FFF	STR or BSC Adapter Address (See Note 2)
l	0-5	Line Description as follows: STR Lines 0 = Interface A - 2 wire Half-Duplex 1 = Interface A - 4 wire Half-Duplex 2 = Interface A - Full-Duplex 3 = Interface B - 2 wire Half-Duplex 4 = Interface B - 4 wire Half-Duplex 5 = Interface B - Full-Duplex
l	0-5	BSC Lines 0 = Interface A - Half-Duplex (1200-9600 baud) 1 = Interface A - Full-Duplex (1200-9600 baud) 2 = Interface A - Full Duplex (19.2-230.4 k-baud) 3 = Interface B - Half-Duplex (1200-9600 baud) 4 = Interface B - Full-Duplex (1200-9600 baud) 5 = Interface B - Full-Duplex (19.2-230.4 k-baud)
c		Clock/Code as follows:
	0-3	STR Lines 0 = Internal Clock X 1 = Internal Clock Y 2 = Internal Clock Z 3 = External Clock

Code Letters	Range	Description
	0-7	BSC Lines

0 = Code A - EBCDIC - No Transparency
1 = Code A - EBCDIC - Transparency
2 = Code A - USASCII - No Transparency
3 = Code A - USASCII - Transparency
4 = Code B - EBCDIC - No Transparency
5 = Code B - EBCDIC - Transparency
6 = Code B - USASCII - No Transparency
7 = Code B - USASCII - Transparency

Default: LINEmm=***01

Notes:

1. Parameter &NUMLNES must specify the number of specifications LINEmm to be included in the generated HASP system.
2. The unit address aaa may be specified as ***. HASP initialization will assign unit addresses to lines whose unit addresses are specified as *** by scanning the OS UCBs. A teleprocessing UCB whose device type field specifies a 2701 STR adapter, a 2701 BSC adapter, or a 2703 BSC adapter will be recognized as a UCB defining a line. If the unit address of such a UCB is not specified explicitly in any of the first &NUMLNES line definitions LINEmm, HASP initialization will assign the UCB to the first line number whose unit address is specified *** and will change the *** to the EBCDIC address specified in the UCB, unless no line definition remains whose unit address is ***, or if (except for M65MP and a 2-CPU multiprocessor) a TIO shows the line not operational, or if (for M65MP and a 2-CPU multiprocessor) the UCB is marked off-line; in that case HASP will not use the line.
3. If a line specification LINEmm designates USASCII, that line cannot be used with any but 2770 and 2780 USASCII terminals. HASP will translate each record it receives into EBCDIC, and each record it transmits into USASCII before transmission.

&LOGOPT

Explanation: Variable symbol &LOGOPT specifies inclusion or exclusion of code to support the HASP System Log feature. The specification should be either YES or NO.

Default: &LOGOPT=YES

Notes:

1. The HASP System Log is a listing included in each user's output of console messages that were produced during processing of the job and of replies to WTORs issued during processing of the job.

2. If &LOGOPT=YES, the HASP System Log may be suppressed on an individual job basis through a parameter in the accounting field of the job card.

3. If the HASP System Log is suppressed, the HASP statistics information normally printed with the job is also suppressed.

&MAXCLAS

Explanation: Variable symbol &MAXCLAS specifies the maximum number of job classes which may be specified via the HASP command $T In,list for a HASP logical partition. The specification must be an integer from 1 to 64, inclusive.

Default: &MAXCLAS=8

Notes:
1. If &MAXCLAS is specified as less than 8, then no more than &MAXCLAS characters may be specified for each of the parameters &CLS(n).

&MAXJOBS

Explanation: Variable symbol &MAXJOBS specifies the maximum number of jobs that can be in the HASP System at any given time.. The specification must be an integer greater than zero.

Default: &MAXJOBS=100

Notes:

1. This variable does not affect the range of HASP job numbers, which is 1 to 999.
2. This variable strongly influences the size of the HASP checkpoint record(s). The size of the first checkpoint record is

   ```
   16*(&MAXJOBS+&NUMPRTS+&NUMTPPR)+
   &NUMDA*((&NUMTGV+7)/8)+40.
   ```

   The size of the second checkpoint record is

   ```
   &MAXJOBS*&JITSIZE.
   ```

   If either checkpoint is longer than the track size of the device on which SPOOL1 is mounted, HASP will not warm start correctly.

&MAXPART

Explanation: Variable symbol &MAXPART specifies, for both MFT and MVT, the number of HASP logical partitions to be defined. The specification must be an integer between 1 and 15, inclusive.

Default: &MAXPART=&MAXXEQS

Notes:
1. The nth logical partition is further defined by the specifications &PRI(n), &OSC(n), and &CLS(n).

&MAXXEQS

Explanation: Variable symbol &MAXXEQS specifies the maximum number of jobs which may concurrently be active in the HASP Execution phase. The specification must be an integer greater than zero.

Default: &MAXXEQS=3

Notes:
1. See also &MAXPART, the variable which determines the number of HASP logical partitions.

&MINBUF

Explanation: This variable is provided to allow installations, which depend on the dynamic buffer construction feature of HASP, to detect the condition where sufficient buffers for proper operation cannot be obtained. The specification should be an integer value representing the minimum number of buffers determined as necessary for the installation (see &NUMBUF).

Default: &MINBUF= 3*&MAXXEQS+2*&NUMRDRS
+&NUMINRS+2*&NUMPRTS+&NUMPUNS
+&NUMTPBF

Notes:

1. HASP will automatically attempt to utilize, via a variable GETMAIN, any free space in its region or partition (hierarchy indicated by &BUFHICH) as additional buffers. If the number of buffers so obtained when added to the variable &NUMBUF is less than the value of &MINBUF the warning message
   &MINBUF BUFFERS NOT AVAILABLE
   will be issued during HASP initialization and processing will continue.
2. Since the changing of HASPGEN options, local modifications and/or OS changes can affect the number of HASP buffers, proper setting of this variable can prevent a possible undetected performance degradation.
3. See the description of HASPGEN parameters &NUMBUF, &BUFHICH and &RESCORE for related information.

&MLBFSIZ

Explanation: Variable symbol &MLBFSIZ specifies the size in bytes of each HASP Multi-Leaving buffer. The specification for &MLBFSIZ must be a positive integer no larger than &TPBFSIZ.

Default: &MLBFSIZ=400

Notes:
1. The value specified for &MLBFSIZ automatically becomes the Multi-Leaving buffer size in each HASP Multi-Leaving Remote Terminal program.
2. Satisfactory support of one device of each type (reader, printer, punch, console) on 8K terminal CPUs is based on the assumption that &MLBFSIZ=400 or less. If the supported terminals include any 8K CPUs, it is recommended that &MLBFSIZ not be increased above 400, even if support of a non-programmable terminal requires increasing &TPBFSIZ to 512.

&MONINTV

Explanation: Variable symbol &MONINTV specifies the interval in seconds at which the HASP Execution Task Monitor will examine CPU utilization characteristics and, if necessary, modify dynamically the order in the TCB chain, of all HASP-controlled job step tasks which fit Execution Task Monitor criteria. The specification should be an integer between 0 and 10 inclusive. If &MONINTV is specified as zero, the Execution Task Monitor is excluded from the generated HASP system.

Default: &MONINTV=5

Notes:
1. See also parameters &XZPRTY (for MVT), &XZMFTL, and &XZMFTH (for MFT).
2. OS Time Slice Groups must not be assigned to the priority level or partitions monitored by the Execution Task Monitor.
3. Users must not specify TIME=1440 in job or execute cards. Such specifications will cause the jobstep TQE not to be created by OS thus forcing the job HIGH in the dynamic group.

&NOPRCCW

Explanation: Variable symbol &NOPRCCW specifies the maximum number of channel command words per channel program for local printers.

Default: &NOPRCCW=15

&NOPUCCW

Explanation: Variable symbol &NOPUCCW specifies the maximum number of channel command words per channel program for local punches.

Default: &NOPUCCW=10

&NUMBUF

Explanation: This variable symbol indicates the number of INPUT/OUTPUT buffers to be included in the HASP load module and should normally be set by each installation, according to the formulae below, to reflect the total number of buffers required for proper operation of HASP. However, since HASP will automatically utilize free space in its region or partition to dynamically construct additional buffers, there are circumstances when &NUMBUF may be set to a value less than the actual number of buffers required for proper HASP operation. In this case, it is assumed that sufficient additional buffers will be dynamically obtained from free storage in the HASP region/partition to provide an adequate total number of buffers (see &MINBUF and &RESCORE). This facility could be used, for example, to allow additional buffers to reside in a storage hierarchy different from that of the HASP load module (see &BUFHICH) or to provide a HASP whose size (and performance and function) can be controlled by the setting of the region or partition size.

Default: &NUMBUF=15

Notes:

1. In order to utilize all the dynamic storage contained in the HASP load module for the initialization process, the value of &NUMBUF must never be less than the value
   1+6000/(80+&BUFSIZE)
2. Since all HASP buffers are maintained in a dynamic pool until required by an active function, installation should determine the appropriate number of buffers for HASP based on predicted simultaneity of the various functions required at the installation. The following indicates the number of buffers required for each logical function. A defined function which is inactive requires no buffers.

Each local input function	--2	
Each internal reader	--1	
Each Remote Input function	--1	
Each local print function	--2	(1 if $PRTBOPT=1)
Each remote print function	--1	(2 if $RPRBOPT=2)
Each local punch function	--1	(2 if $PUNBOPT=2)
Each remote punch function	--1	(2 if $RPUBOPT=2)
Each OS job execution	--2	(minimum value)

For performance reasons, additional buffers must be available to sustain periods of high SYSIN/SYSOUT activity by jobs being processed by OS. It is therefore recommended that additional buffers (beyond the requirements indicated above) be included corresponding to the value: 1+&MAXXEQS.

SEVERE PERFORMANCE AND/OR DEVICE DEGRADATION CAN OCCUR IN A SYSTEM CONTAINING INSUFFICIENT BUFFERS TO PERFORM THE REQUIRED FUNCTIONS.

3. To avoid a complete system failure caused by a buffer "lock-out" condition, the number of available buffers must never be less than the value

```
 &MAXXEQS+&NUMRDRS+&NUMTPES+1
+&NUMPRTS*($PRTBOPT-1)
+&NUMPUNS*($PUNBOPT-1)
+&NUMTPPR*($RPRBOPT-1)
+&NUMTPPU*($RPUBOPT-1)
```

&NUMCONS

Explanation: Variable symbol &NUMCONS specifies the type of console support to be provided by HASP. Two options are available: console support controlled totally by HASP or a HASP interface to the standard OS Console processors.

The specification &NUMCONS=n (n an integer between one and eight) causes HASP to support directly as many as n consoles. The devices which may be used as consoles are 1052, 1053, 3210, 3215, 2260 and 1443. The 1053s and 2260s must be attached locally via a 2848.

The specification &NUMCONS=0 causes HASP to interface with the OS console support, including the MCS option. All devices available with the OS console routines are supported through this interface.

Default: &NUMCONS=0

Notes:

1. If &NUMCONS=0 is specified, then all HASP functions with the exception of those listed below are provided.

   a. Non-HASP Message, e.g., problem program WTOs, will appear on the console(s) without time tags or HASP Job numbers. A copy of the message which includes the Job number and time tag is placed in the HASP System Log.

   b. WTORs issued by the problem program will be included in the HASP Log without the OS assigned reply identification number.

   c. Only the first line of multi-line WTOs issued by problem programs running under HASP will be included in the HASP Systems Log.

2. If OS Multiple Console Support or M65MP or the Time Sharing Option have been SYSGENed and are to be used with HASP, then &NUMCONS should be specified as zero.

3. If &NUMCONS greater than zero is specified and if HASP initialization finds more than &NUMCONS devices of the type supported then the message

   MAXIMUM OF &NUMCONS CONSOLE(S) EXCEEDED

   is issued and HASP uses as consoles the devices with the lowest unit addresses starting with the first 1052, 3210 or 3215 and continuing with the next &NUMCONS-1 devices.

4. 2260 and 1053 support (&NUMCONS greater than zero) is dependent on additional specifications via the variables &SIZ2260 and &SPD2260.

5. If &NUMCONS is specified greater than zero, HASP will intercept and process all WTOs and WTORs, ignoring all MCS information. In particular, HASP will ignore all routing codes. Thus, for example, a WTO with ROUTCDE=11 will be written on HASP consoles and on the HASP System Log but not in an OS System Message Block.

6. See parameter &CONAUTH, &WTLOPT and &LOGOPT for additional information.

7. See Chapter 12, Section 12.7.2, for other restrictions on &NUMCONS>0.

&NUMDA

Explanation: Variable symbol &NUMDA specifies the maximum number of direct-access volumes which may be mounted concurrently as SPOOL volumes. The specification must be an integer greater than zero.

Default: &NUMDA=2

Notes:
1. All direct-access devices except 2321s are eligible for use as SPOOL devices.
2. Specifying &NUMDA greater than the default may require increasing the value of &BUFSIZE.
3. If HASP initialization finds mounted more than &NUMDA direct-access volumes whose volume serials begin with the characters SPOOL, it will write to operator the message
   MAXIMUM OF &NUMDA SPOOL VOLUME(S) EXCEEDED
   and HASP will quiesce.
4. This variable influences the size of the HASP checkpoint record; see Note 2 of variable &MAXJOBS.
5. An associated variable is &NUMTGV.

&NUMDDT

Explanation: Variable symbol &NUMDDT specifies the number of Data Definition Tables (DDTs) to be assembled into HASP. The specification should be an integer between 3 and 256, and equal to

2 + &MAXXEQS + A + B + C + D + E

where

A = number of pseudo-2540 readers defined at SYSGEN time
B = number of pseudo-2540 punches defined at SYSGEN time
C = number of pseudo-1403 printers defined at SYSGEN time
D = number of pseudo-1442 punches defined at SYSGEN time
E = number of pseudo-1443 printers defined at SYSGEN time

Default: &NUMDDT=20

Notes:

1. Pseudo-units for &RDR and &WTR need not be counted in &NUMDDT.

&NUMINRS

Explanation: Variable symbol &NUMINRS specifies the number of 2520 pseudo-punches to be used by the generated HASP System as internal readers.

Default: &NUMINRS=0

Notes:
1. If &NUMINRS is specified as or defaulted to zero, code to support the HASP internal reader feature will be deleted from the generated system.
2. If more than &NUMINRS 2520 pseudo-punches have been specified at SYSGEN time, only the first &NUMINRS 2520 pseudo-punches can be used. It is permissible to specify &NUMINRS greater than the number of 2520 pseudo-punches specified at SYSGEN time.
3. The count of 2520 pseudo-punches is not included in HASPGEN variable &NUMDDT.

&NUMLNES

Explanation: Variable symbol &NUMLNES specifies the largest teleprocessing line identification number (mm in LINEmm) and thus the number of line definitions which are to be used by the generated HASP System. The specification must be an integer between 0 and 99 inclusive. The specification for &NUMLNES automatically becomes the specification for &NUMRJE, unless &NUMRJE is specified explicitly.

Default: &NUMLNES=0

Notes:
1. See also the HASPGEN variable LINEmm.
2. If &NUMLNES is set to or left at zero, all other Remote Job Entry parameters should be left at their default values.

&NUMOACE

Explanation: Variable symbol &NUMOACE specifies the number of overlay areas to be provided for the standard HASP Overlay feature. The specification must be an integer greater than zero.

Default: &NUMOACE=1

Notes:
1. It is judged that more than two overlay areas will benefit only a system with high performance orientation (a very fast CPU or a work load consisting of a large number of short jobs).
2. See also parameter &OLAYLEV.

&NUMPRTS

Explanation: Variable symbol &NUMPRTS specifies the maximum number of physical printers HASP may use to print the printed output of jobs. HASP supports 1403 and 3211 printers. The specification must be an integer greater than zero.

Default: &NUMPRTS=2

Notes:

1. If HASP initialization finds more than &NUMPRTS 1403 and 3211 printers, it writes to operator the message
   MAXIMUM OF &NUMPRTS PRINTER(S) EXCEEDED
   and continues normally, using as printers only the &NUMPRTS printers with lowest unit addresses.
2. Regardless of the number specified for &NUMPRTS, HASP will use only those 1403 and 3211 printers which are operational (as shown by a TIO) or on-line (for M65MP only) when HASP is started.
3. Handling of special forms by printer, the optional 1403 UCS buffer, and the 3211 UCS and Forms Control buffers are explained as part of the $T operator command.
4. This variable influences the size of the HASP checkpoint record; see Note 2 of variable &MAXJOBS.
5. &BUFSIZE must be 536 or greater if 3211 printers are used by HASP.

&NUMPUNS

Explanation: Variable symbol &NUMPUNS specifies the maximum number of physical punches which will be used by HASP to punch the punched output of jobs. HASP supports 3525, 2540, 2520, and 1442 card punches. The specification must be an integer greater than or equal to zero.

Default: &NUMPUNS=1

Notes:

1. If HASP initialization finds more than &NUMPUNS 3525, 2540, 2520 and 1442 punches, it writes to operator the message
   MAXIMUM OF &NUMPUNS PUNCH(S) EXCEEDED
   and continues normally, using as printers only the &NUMPUNS punches with lowest unit addresses.
2. Regardless of the number specified for &NUMPUNS, HASP will use only those 3525, 2540, 2520 and 1442 punches which are operational (as shown by a TIO) or on-line (for M65MP only) when HASP is started.
3. If &NUMPUNS=0, parameter &ACCTNG should be set to NO.

&NUMRDRS

Explanation: Variable symbol &NUMRDRS specifies the maximum number of physical card readers HASP may use to read OS job streams. HASP supports 3505, 2540 and 2501 card readers. The specification must be an integer greater than zero.

Default: &NUMRDRS=1

Notes:

1. If HASP initialization finds more than &NUMRDRS 3505, 2540 and 2501 card readers, it writes to operator the message

   MAXIMUM OF &NUMRDRS READER(S) EXCEEDED

   and continues normally, using as readers only the &NUMRDRS readers with lowest unit addresses.
2. Regardless of the number specified for &NUMRDRS, HASP will use only those 3505, 2540 and 2501 readers which are operational (as shown by a TIO) or on-line (for M65MP only) when HASP is started.

&NUMRJE

Explanation: Variable symbol &NUMRJE specifies the largest remote terminal identification number (nn in RMTnn) and thus the number of remote terminal definitions which are to be used by the generated HASP System. The specification must be an integer between 0 and 99 inclusive.

Default: &NUMRJE=&NUMLNES

Notes:
1. See also the HASPGEN variable RMTnn.
2. If this variable is not specified and if &NUMLNES is specified as an integer greater than zero, the first &NUMLNES remote terminal definitions RMTnn are used by the generated HASP System, whether they are specified explicitly or by default.

&NUMTGV

Explanation: Variable symbol &NUMTGV specifies the number of units (track groups) into which each SPOOL volume will be divided for HASP allocation purposes. The specification must be a positive integer no greater than the number of tracks on the SPOOL device with the fewest tracks.

Default: &NUMTGV=400

Notes:

1. The user should decide upon the number of tracks he requires in a track group and then divide by that number the total number of tracks (except alternate tracks) on a typical SPOOL device type at the installation. For example, to obtain a track group size of five tracks on a 2311, the division would yield a quotient of 400; the user would specify &NUMTGV=400. If the same installation occasionally used a 2314 as a SPOOL device, the track group size for the 2314 would automatically become ten tracks.
2. Specifying a large &NUMTGV may require increasing the value of &BUFSIZE.
3. For each SPOOL volume it finds, HASP initialization calculates number of tracks per group by dividing the total number of tracks on the volume by &NUMTGV. It then marks unavailable all track groups which lie partially or wholly outside the first extent of data set SYS1.HASPACE on that volume. HASP initialization also computes the number of HASP buffers of length &BUFSIZE which will fit on a track of the SPOOL volume for each SPOOL volume mounted.

&NUMTPBF

Explanation: Variable symbol &NUMTPBF specifies the number of HASP Teleprocessing buffers for HASP Remote Job Entry to be assembled into the generated HASP system. The specification must be an integer greater than or equal to zero.

Default: &NUMTPBF=&NUMLNES

Notes:
1. Each signed-on HASP Multi-Leaving terminal requires at least two HASP Teleprocessing buffers; each other signed-on terminal requires at least one buffer. If &NUMTPBF is specified too small, HASP RJE may not work correctly.
2. See also parameters &TPBFSIZ and &MLBFSIZ.

&NUMTPES

Explanation: Variable symbol &NUMTPES specifies the maximum number of tape drives HASP may use simultaneously to read OS job streams. The specification must be an integer greater than or equal to zero.

Default: &NUMTPES=1

Notes:

1. If &NUMTPES=0 is specified, code required to support tapes as readers is omitted from the generated HASP System.
2. Since the operator specifies a unit address when issuing the start command to a tape drive (e.g., $S TPE1,182), there is rarely a need to specify for &NUMTPES a number greater than one.
3. Tapes should be equivalent to the JCL specification LABEL=(1,NL),DCB=(RECFM=FB,LRECL=80, BLKSIZE=nnnnn) where nnnnn is not greater than (10*&BUFSIZE)/11. For seven-track tape, use the additional DCB specification DEN=2,TRTCH=C.

&NUMTPPR

Explanation: Variable symbol &NUMTPPR specifies the maximum number of HASP Remote Terminal (including Multi-Leaving) printed-output streams that can simultaneously be active. The specification must be an integer greater than or equal to zero.

Default: &NUMTPPR=&NUMLNES

Notes:
1. If any remote terminal is to receive printed output, &NUMTPPR must not be zero.

&NUMTPPU

Explanation: Variable symbol &NUMTPPU specifies the maximum number of HASP Remote Terminal (including Multi-Leaving) punched-output streams that can simultaneously be active. The specification must be an integer greater than or equal to zero.

Default: &NUMTPPU=&NUMLNES

Notes:
1. If any remote terminal is to receive punched output, &NUMTPPU must not be zero.

Explanation: Variable symbol &NUMTPRD specifies the maximum number of HASP Remote Terminal (including Multi-Leaving) input streams that can simultaneously be active. The specification must be an integer greater than or equal to zero.

Default: &NUMTPRD=&NUMLNES

Notes:
1. If any remote terminal is to read cards (including the /*SIGNON and /*SIGNOFF control cards) &NUMTPRD must not be zero.

&NUMWTOQ

Explanation: Variable symbol &NUMWTOQ specifies the number of message buffers to be provided in HASP. The specification must be an integer greater than two.

Default: &NUMWTOQ=15

Notes:

1. If &NUMCONS is specified greater than zero, additional message buffers are needed.
2. If Remote Job Entry is used, more message buffers are needed. This is especially true with console support for MULTI-LEAVING terminals.
3. Serious system degradation can be caused by specifying too few message buffers.
4. During periods of high console activity, when no message buffers are available, certain non-critical HASP messages will be discarded rather than delaying the associated process. These include certain RJE oriented messages (such as communication line error messages), execution time/line/card excession messages (continued excession will be noted when a message buffer becomes available), and certain I/O error messages on HASP-controlled devices. Additionally, when no message buffers are available in a system in which &NUMCONS is greater than zero, messages from the OS error task are queued only to a depth of one which can result in the loss of some of these messages.

&OLAYLEV

Explanation: Variable symbol &OLAYLEV specifies a HASP overlay level to be used for assembly of the various HASP control sections. Any potential overlay code defined (by the $OVERLAY macro) with a residence factor greater than &OLAYLEV will be assembled as resident code rather than overlay code. The specification for &OLAYLEV must be an integer between 0 and 15, inclusive.

Default: &OLAYLEV=15

Notes:

1. HASP uses only residence factors 4, 8, 12 and (for HASP initialization only) 0.
2. If &OLAYLEV=15, all potential overlay code will be assembled as overlay code.
3. If &OLAYLEV=0, all potential overlay code except that in HASP initialization will be assembled as resident code. HASP main storage requirements will be increased by approximately 24K over the case &OLAYLEV=15.
4. Regardless of the setting of &OLAYLEV, the installation systems programmer may use control cards for the HASP Overlay Builder after the HASPGEN process is complete to specify that a particular section of potential overlay code be made either resident code or overlay code.

&OREPSIZ

Explanation: Variable symbol &OREPSIZ specifies the size in bytes of a table in HASP to be used to hold REP data for true overlay code. The REPs associated with a particular section of true overlay code will be applied to that code every time it is brought into main storage from the HASP overlay library. The specification for &OREPSIZ must be either 0 or an integer not less than 10.

Default: &OREPSIZ=0

Notes:

1. Each entry in the HASP Overlay REP table consists of 8+n bytes ($2 \leq n \leq 256$) where n is the number of contiguous bytes to be changed in a section of overlay code.
2. The table is used only if the operator specifies to HASP initialization that REPs are to be used and if some of the REPs are for sections of true overlay code.
3. If the HASP Overlay REP table is too small to handle all true overlay REPs, HASP initialization writes to operator the message
   OVERLAY REPPING ERROR
   and HASP quiesces.
4. See also Section 6.4 of this manual.

&OSC(n)

Explanation: Subscripted variable symbols &OSC(n) specify OS job classes. A job selected by HASP logical partition n will be submitted to OS with the job class &OSC(n). Each specification must be a single unique letter between A and O, inclusive. No two specifications may be the same.

Default:
```
&OSC(1)=A
&OSC(2)=B
&OSC(3)=C
&OSC(4)=D
&OSC(5)=E
&OSC(6)=F
&OSC(7)=G
&OSC(8)=H
&OSC(9)=I
&OSC(10)=J
&OSC(11)=K
&OSC(12)=L
&OSC(13)=M
&OSC(14)=N
&OSC(15)=O
```

Notes:

1. Only the first &MAXPART specifications, &OSC(1) through &OSC(&MAXPART) will be used.
2. In an MVT system, HASP initialization issues the &MAXPART commands
```
S INIT.HOSINIT&OSC(1),,,&OSC(1)

S INIT.HOSINIT&OSC(&MAXPART),,,&OSC(&MAXPART).
```
3. In an MFT system, HASP initialization issues the single command
```
S INIT.ALL;
```
thus the classes of the MFT partitions to be controlled by HASP must have already been defined. Each such partition must be defined with only one job class; that job class must match one and only one of the &MAXPART job classes &OSC(1), &OSC(&MAXPART).

&OSINOPT

Explanation: Variable symbol &OSINOPT specifies inclusion or exclusion of the HASP OS Input Spooling option. The specification must be either YES or NO. If &OSINOPT=YES and a DD * (or DD DATA) statement specifies the DCB keyword, HASP will pass the DD statement and the data following it to the OS Reader/Interpreter; OS will perform input spooling. If &OSINOPT=NO, or if &OSINOPT=YES and no DCB parameter is specified on the DD * (or DD DATA) statement, HASP will SPOOL the input data as usual.

Default: &OSINOPT=NO

Notes:

1. If the DLM parameter is specified on a DD * (or DD DATA) statement, HASP will SPOOL the input data regardless of the setting of &OSINOPT or the inclusion of DCB parameters.

&OUTPOPT

Explanation: Variable symbol &OUTPOPT specifies the action to be taken when a job exceeds its estimated print lines or punched cards of output. The specification must be one of the integers 0, 1 or 2. For &OUTPOPT=2, output excession causes the job to be cancelled with a dump. For &OUTPOPT=1, output excession causes the job to be cancelled without a dump. For &OUTPOPT=0, output excession does not cause the job to be cancelled.

Default: &OUTPOPT=0

Notes:

1. Regardless of the specification for &OUTPOPT, output excession causes messages to be written to the operator. See also Notes 1 and 2 of $OUTXS.
2. If &OUTPOPT=2 is specified, users should use SYSUDUMP or SYSABEND DD cards if a storage dump is desired on output excession.
3. For &OUTPOPT=1 or 2, the job will not be cancelled if, at the time of output excession, the calling task is normally or abnormally terminating. Therefore, if this terminating task is not the job step task, job termination may not occur unless provided for by the program or unless the operator cancels the job.

$OUTXS

Explanation: Ordinary symbol $OUTXS specifies the interval, in print lines/punched cards, at which messages will be written to the operator informing him that a job's print line count or punch card count has been exceeded. The specification must be an integer greater than zero.

Default: $OUTXS=2000

Notes:
1. The first print line excession message will be written to the operator when the job's estimated print line count has been exceeded.
2. The first punched card excession message will be written to the operator when the job's estimated punched card count has been exceeded.

&PID(n)

Explanation: Subscripted variable symbols &PID(n) specify the identifiers to be used with the HASP logical partitions. Each specification must be a unique 1- or 2-character string.

Default:
```
&PID(1)=1
&PID(2)=2
&PID(3)=3
&PID(4)=4
&PID(5)=5
&PID(6)=6
&PID(7)=7
&PID(8)=8
&PID(9)=9
&PID(10)=10
&PID(11)=11
&PID(12)=12
&PID(13)=13
&PID(14)=14
&PID(15)=15
```

Notes:

1. Only the first &MAXPART specifications, &PID(1) through &PID(&MAXPART), will be used.
2. The identifiers &PID(n) are used in messages to and commands from the operator. For example, when an operator uses the set command $T Imm,list he is referring not to logical partition mm but to logical partition n, where &PID(n)=mm.

&PRI(n)

Explanation: Subscripted variable symbols &PRI(n) specify OS job priorities. A job selected by HASP logical partition n will be submitted to OS with the job priority &PRI(n). Each specification must be an integer between 1 and 15, inclusive.

Default:
```
&PRI(1)=7
&PRI(2)=7
&PRI(3)=7
&PRI(4)=7
&PRI(5)=7
&PRI(6)=7
&PRI(7)=7
&PRI(8)=7
&PRI(9)=7
&PRI(10)=7
&PRI(11)=7
&PRI(12)=7
&PRI(13)=7
&PRI(14)=7
&PRI(15)=7
```

Notes:

1. The defaults are all the same as &XZPRTY. This allows the HASP Execution Task Monitor to regulate all job steps under the control of HASP. See also parameters &XZPRTY and &MONINTV.
2. These parameters have no effect in MFT.
3. The priorities defined by &PRI(n) affect only OS execution. The priority of a job in the HASP Job Queue is determined by parameters &RPRT(m), &RPRI(m), &XLIN(m), and &XPRI(m).
4. Only the first &MAXPART specifications, &PRI(1) through &PRI(&MAXPART), will be used.

$PRICONA

Explanation: Ordinary symbol $PRICONA specifies the unit address of a 1052, 3210, 3215, 1443, or 1403 to which HASP will issue a SIO in the event of a catastrophic error. The message HASP writes to this device is:

HASP CATASTROPHIC ERROR. CODE=xxx.

The specification must be a valid unit address.

Default: $PRICONA=01F

Notes:

1. When HASP is operating with M65MP in a 2-CPU multiprocessor environment, the device specified by $PRICONA must be operational to both CPUs when a HASP catastrophic error occurs.

$PRIDCT

Explanation: Ordinary symbol $PRIDCT specifies the number of print lines to appear on each HASP job separator page for local printers. The specification must be an integer greater than or equal to zero. If the specification is zero, no separator page will be produced on local printers.

Default: $PRIDCT=61

Notes:
1. The equivalent HASPGEN parameter for remote terminal printers is $TPIDCT.

&PRIHIGH

Explanation: Variable symbol &PRIHIGH specifies a HASP priority to be associated with the HASP Priority Aging feature. A job will not be priority-aged if its HASP priority is (or becomes) greater than or equal to &PRIHIGH. The specification must be an integer between 0 and 15, inclusive.

Default: &PRIHIGH=10

Notes:
1. If &PRIRATE=0, parameter &PRIHIGH is not used.
2. See also parameters &PRIRATE and &PRILOW.

&PRILOW

Explanation: Variable symbol &PRILOW specifies a HASP priority to be associated with the HASP Priority Aging feature. A job will not be priority-aged by HASP unless its HASP priority is initially at least &PRILOW. The specification must be an integer between 0 and 15, inclusive.

Default: &PRILOW=5

Notes:
1. If &PRIRATE=0, parameter &PRILOW is not used.
2. See also parameters &PRIRATE and &PRIHIGH.

&PRIRATE

Explanation: Variable symbol &PRIRATE specifies the amount by which a job's HASP priority will be incremented in 24 hours by the HASP Priority Aging feature. For example if &PRIRATE=3 then a job's priority will be incremented by one for every eight hours it remains in the system. But a job's priority will not be incremented unless it is at least &PRILOW; nor will a job's priority be incremented above &PRIHIGH. The specification must be an integer greater than or equal to zero. If zero is specified, Priority Aging is excluded from the generated HASP system.

Default: &PRIRATE=0

Notes:
1. If &PRIRATE=0, parameters &PRILOW and &PRIHIGH are not used.
2. See also parameters &RPRT(n), &RPRI(n), &XLIN(n), and &XPRI(n).
3. If a job's priority is specified on the /*PRIORITY control card, the job will be priority-aged if its priority is eligible.

$PRTBOPT

Explanation: Ordinary symbol $PRTOPT specifies the printer buffering option to be used for local HASP printers. The specification must be either 1 (for single buffering) or 2 (for double buffering).

Default: $PRTBOPT=2

&PRTRANS

Explanation: Variable symbol &PRTRANS specifies translation for lines of print. The specification must be either YES or NO.

Default: &PRTRANS=YES

Notes:
1. If &PRTRANS is specified as YES, each line to be printed by HASP is first translated. Translation changes lower-case letters to upper-case letters and characters invalid on a PN train to blanks.
2. If any print train is to be used on any HASP-controlled printer which has characters not equivalent to those on a PN train or a P11 train, &PRTRANS must be specified as NO.

&PRTUCS

Explanation: Variable symbol &PRTUCS specifies the name of the print chain or print train which HASP initially assumes is mounted on every local 1403 printer SYSGENed with the UCS feature, and on every local 3211 printer. The UCS identifier can be modified by the operator individually by printer. The specification should be either AN, HN, PN, QN, RN, UN, A11, H11, P11, U11, or 0.

Default: &PRTUCS=0

Notes:
1. A specification of zero causes HASP to bypass the UCS loading procedure on all local printers until the UCS type of each printer is specified by the operator.
2. Only the first character of &PRTUCS is interrogated. That is to say, an AN specification is equivalent to an A11 specification, an H11 specification is equivalent to an HN specification, etc.
3. If a UCS specification is encountered which is not valid for the type of printer being addressed, the UCS loading procedure will be bypassed.
4. The UN and U11 specifications are provided for installation use to support other type of print chains.

$PUNBOPT

Explanation: Ordinary symbol $PUNBOPT specifies the punch buffering option to be used for local HASP punches. The specification must be either 1 (for single buffering) or 2 (for double buffering).

Default: $PUNBOPT=1

&RDR

Explanation: Variable symbol &RDR specifies the unit address of a pseudo-2540 reader to be used with HOSRDR to supply jobs to OS. The specification must be a valid unit address which has been specified at SYSGEN time as a pseudo-2540 reader.

Default: &RDR=0FC

&RDRPART

Explanation: Variable symbol &RDRPART specifies for MFT systems the identifier field of an OS START command issued by HASP initialization for the OS reader/interpreter HOSRDR. The complete START command is

```
S  HOSRDR.&RDRPART,&RDR.
```

The specification must be a valid identifier field for an OS START reader command, as described in the OS Operator's Guide.

Default: &RDRPART=S

Notes:

1. If HASP initialization detects an MVT system, this parameter is not used.

$REPRDR

Explanation: Ordinary symbol $REPRDR specifies the unit address of a physical 2540 or 2501 card reader from which HASP initialization will read REP cards if requested by the operator. The specification must be a valid unit address.

Default: $REPRDR=00C

Notes:

1. When REP cards are to be read and HASP is operating with M65MP in a 2-CPU multiprocessor environment, the device specified by $REPRDR must be operational to both CPUs.

$REPWTR

Explanation: Ordinary symbol $REPWTR specifies the unit address of a physical 1403 or 1443 on which each REP card read is to be printed, if printing of REP cards is requested by the operator. The specification must be a valid unit address.

Default: $REPWTR=00E

Notes:

1. When REP cards are to be printed and HASP is operating with M65MP in a 2-CPU multiprocessor environment, the device specified by $REPWTR must be operational to both CPUs.

&RESCORE

Explanation: Variable symbol &RESCORE specifies a storage size, in multiples of 1024 bytes. HASP will always issue a GETMAIN for additional storage for HASP buffers; all such storage but &RESCORE*1024 bytes will be used for buffers.

The specification must be an integer greater than or equal to zero.

Default: &RESCORE=0

Notes:
1. MFT users should set &RESCORE=1 or larger if 80A or 804 abends occur during HASP operation. This is normally not required for unmodified HASP with MVT, because of its 2K block storage management technics.
2. &RESCORE only prevents HASP from using all available storage for buffers. It will not cure abends due to inadequate partition size.

&RJOBOPT

Explanation: Variable symbol &RJOBOPT specifies whether or not an illegal HASP JOB card is to prevent execution of the associated job. The specification must be either YES or NO.

Default: &RJOBOPT=NO

Notes:
1. If &RJOBOPT=YES and HASP reads a JOB card whose accounting field does not match the specifications required by HASP, the job is flushed by HASP and an appropriate message is written to the operator.

RMTnn

Explanation: Ordinary symbols RMTnn specify the characteristics of remote terminals to be used with HASP Remote Job Entry. Terminals must be defined consecutively, starting with RMT01. Each specification must be a 14-character string of the form

RMTnn=mmooppiillwtdf

where the letters represent the following:

Code Letters	Range	Description
nn	01-99	Remote Number
mm	01-99	Line Number (**indicates /*SIGNON assignment)
oo	00-99	Print Routing (Remote Number)
pp	00-99	Punch Routing (Remote Number)
ii	00-15	Priority Increment for this Remote
ll	00-15	Priority Limit for this Remote
w	0-6	Printer Width as follows: 0 = 80 characters 1 = 100 characters 2 = 120 characters 3 = 132 characters 4 = 144 characters 5 = 150 characters 6 = 96 characters
t	0-6	Terminal Type as follows: 0 = 1009, 2770 1 = 1978, 2780, 7702 2 = 2922, S360/20 Sub-Model 2, 4 3 = System 360/20 Sub-Model 5, 6 4 = System 360/22, 25, 30, 40, etc. 5 = 1130 6 = System/3 7 = 3780

d 0-4 Data Format as follows:

- 0 = Unblocked fixed length
- 1 = Blocked fixed length
- 2 = Unblocked variable length (Note - use this for basic 2770 terminals.)
- 3 = Blocked variable length (Note - use this for all 3780, 2780, and 1978 terminals, and for 2770 terminals with Buffer Expansion.)
- 4 = Programmable Interface (Note - use this for all STR 20 and BSC MULTI-LEAVING interfaces.)

f Terminal Features as follows:

0-9 3780 Terminal Features

f	Compress Expand	Horizontal Format Control	Transparency
0	No	No	No
1	No	No	Yes
4	No	Yes	No
5	No	Yes	Yes
8	Yes	No	No
9	Yes	No	Yes

0-@ 2770 Terminal Features

f	Compress Expand	Horizontal Format Control	Additional Buffer Expansion	Transparency
0	No	No	No	No
1	No	No	No	Yes
2	No	No	Yes	No
3	No	No	Yes	Yes
4	No	Yes	No	No
5	No	Yes	No	Yes
6	No	Yes	Yes	No
7	No	Yes	Yes	Yes
8	Yes	No	No	No
9	Yes	No	No	Yes
#	Yes	No	Yes	No
@	Yes	No	Yes	Yes

0-7 2780 Terminal Features

f	Horizontal Format Control	Multiple Record Feature	Transparency
0	No	No	No
1	No	No	Yes
2	No	Yes	No
3	No	Yes	Yes
4	Yes	No	No
5	Yes	No	Yes
6	Yes	Yes	No
7	Yes	Yes	Yes

0-3 MULTI-LEAVING Terminal Features

f	Console Support	Transparency
0	No	No
1	No	Yes
2	Yes	No
3	Yes	Yes

Default: RMTnn=**nn0000153131

Notes:

1. Parameter &NUMRJE must specify the number of specifications RMTnn to be included in the generated HASP system.
2. No two specifications RMTnn may specify the same line number (mm). If ** is specified instead of a line number for mm, the associated remote terminal may connect to HASP via any suitable line. HASP will logically connect the terminal with the line when it recognizes the /*SIGNON control card. If line number is specified explicitly, the associated terminal need not use a /*SIGNON card.
3. The line number specification mm refers to line specification LINEmm, which in turn specifies the unit address of the line.
4. For print and punch routing, a specification of 00 causes output from jobs submitted at the Remote Terminal to be printed/punched locally, unless re-routed.

5. Priority increment is the value to be added to the priority of a job submitted from the Remote Terminal.
6. Priority limit is the maximum value of priority for any job submitted from the Remote Terminal.
7. If any Multi-Leaving workstation is to utilize more than one reader, printer or punch, see Section 12.16 for additional information.
8. For a basic 2770 (128 byte buffers), Printer Width must be specified as 120 characters or less.
9. Printed and non-transparent punched output to a basic 2770 will be variable-blocked up to the limit of the 128 byte buffers, even though the Data Format must be specified as variable-unblocked.
10. The following table gives minimum values of &TPBFSIZ required to support various non-programmable terminals:

Minimum &TPBFSIZ	Terminal Type, Features
128	2770 basic
256	2770 with Buffer Expansion
328	1978
400	2780
512	2770 with Buffer Expansion and Additional Buffer Expansion
512	3780

See also parameters &TPBFSIZ and &MLBFSIZ.

(The remainder of this page intentionally left blank.)

$RPRBOPT

Explanation: Ordinary symbol $RPRBOPT specifies the printer buffering option to be used for all printers at HASP Remote Terminals. The specification must be either 1 (for single buffering) or 2 (for double buffering).

Default: $RPRBOPT=1

Notes:
1. The specification refers to HASP regular buffers, not to HASP Teleprocessing buffers.

&RPRI(n)

Explanation: Subscripted variable symbols &RPRI(n) specify tentative job priorities corresponding to intervals defined by subscripted variable symbols &RPRT(n). If a user specifies in the accounting field of his job card an estimated execution time of t minutes, the job's tentative priority will be &RPRI(n) where

&RPRT(n-1)<t≤&RPRT(n).

Each specification must be an integer between 0 and 15 inclusive.

Default: &RPRI(1)=9
&RPRI(2)=8
&RPRI(3)=7
&RPRI(4)=6
&RPRI(5)=5
&RPRI(6)=4
&RPRI(7)=3
&RPRI(8)=2
&RPRI(9)=1

Notes:
1. The values &RPRI(n) will normally be in opposite order from the subscripts n.
2. See also Notes 1 and 2 for &RPRT(n).

&RPRT(n)

Explanation: Subscripted variable symbols &RPRT(n) specify estimated execution times in minutes. If a user specifies in the accounting field of his job card an estimated execution time of t minutes, and if t satisfies the relation

&RPRT(n-1)<t≤&RPRT(n)

then &RPRI(n) will be the tentative priority of the job. If t is less than &RPRT(1) or greater than &RPRT(9), value &RPRI(1) or zero will be the tentative priority of the job. Each specification must be an integer between 1 and X'FFFFFF'/60 inclusive.

Defaults:
```
&RPRT(1)=2
&RPRT(2)=5
&RPRT(3)=15
&RPRT(4)=X'FFFFFF'/60
&RPRT(5)=X'FFFFFF'/60
&RPRT(6)=X'FFFFFF'/60
&RPRT(7)=X'FFFFFF'/60
&RPRT(8)=X'FFFFFF'/60
&RPRT(9)=X'FFFFFF'/60
```

Notes:

1. Priority &RPRI(n) is overridden by HASP control card /*PRIORITY.
2. The tentative priority defined above is adjusted according to &XLIN(m) and the estimated print lines specified in the accounting field of the user's JOB card. If the user estimated p print lines, and if p satisfies the relation
   &XLIN(m)<p≤&XLIN(m+1)
   then the tentative priority is reduced by m.
3. The values &RPRT(n) should be in the same order as the subscripts n.

&RPS

Explanation: Variable symbol &RPS specifies inclusion or exclusion of Rotational Position Sensing for all HASP channel programs directed to 3330 and 2305 devices. The specification must be either YES or NO.

Default: &RPS=NO

Notes:
1. Regardless of the setting of &RPS, HASP will correctly operate with any supported direct-access device or combination of devices.

(The remainder of this page intentionally left blank.)

$RPUBOPT

Explanation: Ordinary symbol $RPUBOPT specifies the punch buffering option to be used for all punches at HASP Remote Terminals. The specification must be either 1 (for single buffering) or 2 (for double buffering).

Default: $RPUBOPT=1

Notes:
1. The specification refers to HASP regular buffers, not to HASP Teleprocessing buffers.

&RQENUM

Explanation: Variable symbol &RQENUM specifies the number of WTOR reply buffers to be provided in the generated HASP system. The specification must be an integer greater than zero.

Default: &RQENUM=5

Notes:
1. This parameter is ignored if &NUMCONS=0.
2. If &NUMCONS is not specified as zero, no more than &RQENUM replies can be outstanding at any time.

&SIZ2260

Explanation: Variable symbol &SIZ2260 specifies screen width in characters for local 2260s (attached via 2848) to be used as HASP operator consoles. The specification must be either 0, 40, or 80. If 0 is specified, support for local 2260s and local 1053s (attached via 2848) will be excluded from the generated HASP system.

Default: &SIZ2260=0

Notes:
1. This parameter is not used if &NUMCONS=0.
2. See also parameter &SPD2260.
3. 2260 and 1053 console errors are retried, but are not recorded by the ERP and LOGREC functions of OS.

&SPD2260

Explanation: Variable symbol &SPD2260 specifies roll rate in hundredths of a second for local 2260s (attached via 2848) to be used as HASP operator consoles. If new messages are pending for display on a HASP 2260 console, they will be displayed (and old messages will be deleted, if the screen is full) at the rate of one message every &SPD2260/100 seconds.

Default: &SPD2260=50

Notes:

1. This parameter is not used if &NUMCONS=0 or if &SIZ2260=0.

&SPOLMSG

Explanation: Variable symbol &SPOLMSG specifies the number of physical records in the first extent of SYS1.HASPACE on SPOOL1 which are to be reserved for holding operator and HASP messages for HASP Remote Terminals. Each physical record is capable of holding one or more messages for a single remote terminal. Messages are held if they are directed to:

- any terminal not signed on;
- any signed-on hardware terminal which is currently processing an input or output stream;
- any signed-on computer terminal that is not a Multi-Leaving terminal with a console.

If a message is to be held but no space is available to hold it, the message is thrown away without operator notification.

The specification for &SPOLMSG must be an integer greater than or equal to zero. If &SPOLMSG is specified as zero, no messages will be sent to hardware terminals.

Default: &SPOLMSG=10*&NUMRJE

Notes:

1. Only the $DM command can generate messages to a terminal not signed on.
2. For signed-on terminals, messages are generated for job-on-reader, by $DM, and as responses to commands from the terminal.
3. Each message to a terminal (except to a Multi-Leaving remote defined with a console) is held until it can be printed, or until HASP is restarted.

&SPOOL

Explanation: Variable symbol &SPOOL specifies the first 5 characters of the volume serial of each mounted direct-access volume to be used for spooling by HASP. The specification must be exactly five characters and valid as a volume serial.

Default: &SPOOL=SPOOL

Notes:
1. If this variable is changed from its default, certain HASP messages will vary from their documentation in Chapter 11.
2. With the default, HASP requires that at least SPOOL1 be mounted. If, for example, &SPOOL=$-#-@, then HASP would require at least $-#-@1 be mounted.
3. The requirement for "SPOOL " in the HOSRDR procedure, as documented in Section 10.2.2.2, is not affected by the setting of &SPOOL.

(The remainder of this page intentionally left blank.)

&STRCPU

Explanation: Variable symbol &STRCPU specifies inclusion or exclusion in the HASP Remote Terminal Access Method of Remote Job Entry support for the System/360 Model 20 with a Synchronous Transmit-Receive (STR) adapter and the associated HASP Remote Terminal program. The specification must be either YES or NO.

Default: &STRCPU=NO

&STR1978

Explanation: Variable symbol &STR1978 specifies inclusion or exclusion in the HASP Remote Terminal Access Method of Remote Job Entry support for Synchronous Transmit-Receive (STR) hardware terminals such as the 1978. The specification must be either YES or NO.

Default: &STR1978=NO

&TIMEOPT

Explanation: Variable symbol &TIMEOPT specifies the action to be taken when a job's estimated execution time is exceeded. The specification must be one of the integers 0, 1, 2 or 4. For &TIMEOPT=4, the job's time limits will not be monitored. For &TIMEOPT=2, time excession causes the job to be cancelled with a dump. For &TIMEOPT=1, time excession causes the job to be cancelled without a dump. For &TIMEOPT=2, &TIMEOPT=1, or &TIMEOPT=0, time excession causes messages to be written to the operator.

Default: &TIMEOPT=4

Notes:

1. See also Notes 1 and 2 of $ESTIME, which apply for &TIMEOPT=0, &TIMEOPT=1, and &TIMEOPT=2.
2. For &TIMEOPT=1 or 2, the job will not be cancelled if, when time is exceeded, the job's task which last called HASP for I/O service is normally or abnormally terminating. Therefore, if this terminating task is not the job step task, job termination may not occur unless provided for by the program or unless the operator cancels the job.

$TIMEXS

Explanation: Ordinary symbol $TIMEXS specifies the interval, in minutes, at which messages will be written to the operator informing him that a job's execution time is exceeded. The specification must be an integer greater than zero.

Default: $TIMEXS=1

Notes:
1. The first time excession message is written to the operator when the job's estimated execution time has been exceeded.
2. If &TIMEOPT is specified greater than 2, $TIMEXS is not used.
3. See also Note 2 of $ESTIME.

&TPBFSIZ

Explanation: Variable symbol &TPBFSIZ specifies the size in bytes of each HASP Teleprocessing buffer. The specification must be a positive integer.

Default: &TPBFSIZ=400

Notes:

1. The value of &TPBFSIZ is the maximum size of any HASP Teleprocessing buffer. See also parameter &MLBFSIZ, which may never be specified larger than &TPBFSIZ.
2. The HASP Remote Terminal program for the System/360 Model 20 with an STR communications adapter (HRTPSM20) uses a teleprocessing buffer size of &TPBFSIZ; all other HASP Remote Terminal programs are Multi-Leaving programs, and use &MLBFSIZ.
3. The parameter &TPBFSIZ is specified only once, at HASPGEN time; it is conveyed automatically to the requisite Remote Terminal programs by HASPGEN.
4. See Note 9 under RMTnn for minimum &TPBFSIZ required when HASP supports non-programmable terminals.

$TPIDCT

Explanation: Ordinary symbol $TPIDCT specifies the number of print lines to appear on each HASP job separator page for jobs whose printed output is directed to any HASP Remote Terminal. The specification must be an integer greater than or equal to zero. If the specification is zero, no separator page will be produced on remote printers.

Default: $TPIDCT=6

Notes:
1. The equivalent HASPGEN parameter for local printers is $PRIDCT.

&TRACE

Explanation: Variable symbol &TRACE specifies inclusion or exclusion of a facility for event-tracing and statistics-gathering in the generated HASP system. It also specifies the number of entries to be generated in the HASP trace table. The specification must be an integer greater than or equal to zero.

Default: &TRACE=0

Notes:
1. Inclusion of the HASP Trace facility causes the OS program interrupt exit (SPIE) mechanism to work incorrectly. For this reason, the HASP Trace should not be included in any generated HASP system designed for normal production.
2. The &TRACE option is independent of the &DEBUG option.

&USASCII

Explanation: Variable symbol &USASCII specifies inclusion or exclusion in the HASP Remote Terminal Access Method of the capability to use USASCII line control characters as well as EBCDIC line control characters. If any line specification LINEmm for a BSC line has value c set to 2, 3, 6 or 7, &USASCII should be set to YES; otherwise, &USASCII should be set to NO.

Default: &USASCII=NO

$WAITIME

Explanation: Ordinary symbol $WAITIME specifies a time interval, in seconds. For hardware terminals, the HASP Remote Terminal Access Method will wait $WAITIME seconds at the completion of processing of any input stream, printed output stream, or punched output stream, to allow the operator time to alter the normal sequence of Remote Job Entry operations. For example, the operator may wish to transmit another job to HASP after a previous job has finished printing rather than wait till the previous job has finished punching.

The specification for $WAITIME must be an integer greater than zero.

Default: $WAITIME=1

&WCLSREQ

Explanation: Variable symbol &WCLSREQ specifies optional requeueing for OS output classes specified by &WTRCLAS. The values assigned &WCLSREQ are effective only if &WTRPART=*.

If &WTRPART=*, then the HASP writer subtask (load module HASPWTR) processes jobs queued in the OS output queues defined by &WTRCLAS. At the end of processing a job whose output class is the nth character of &WTRCLAS, HASPWTR examines the nth character of &WCLSREQ. If the nth character of &WCLSREQ is *, HASPWTR deletes the job from the OS Job Queue. But if the nth character of &WCLSREQ is an OS Output class, HASPWTR requeues the job in the OS Output queue specified by the nth character of &WCLSREQ (which must be different from any class specified in &WTRCLAS).

The specification must be a string of one to eight characters each of which is either * or a unique valid OS output class different from any specified in &WTRCLAS. If more characters are specified than were specified for &WTRCLAS, the excess characters are unused.

Default: &WCLSREQ=********

Notes:

1. The output requeueing option is useful for providing an extra copy of a job's system messages to, for example, a conversational programming terminal.
2. A requeued job is not referenced by HASP, but must be accessed by a standard OS Output Writer or other suitable means.
3. A requeued job may contain a mixture of system messages and sysout data sets of the same class, if the sysout data sets were spooled by OS (see HASPGEN parameter $$x). The module HASPWTR does not process the sysout data sets, but requeues the entire job containing them in the new class specified by &WCLSREQ. The system messages and sysout data sets are then available to a standard OS Output Writer which is processing the new class.

4. Any DD statements in the system messages of a requeued job, which are originally coded as DD* or DD DATA and are not subject to OS spooling (see HASPGEN parameter &OSINOPT), are available to a Writer processing a &WCLSREQ class as DD$ and DD CATA respectively. They are printed as DD$ and DD CATA unless the Writer is programmed to change them to their original form.

&WTLOPT

Explanation: Variable symbol &WTLOPT specifies inclusion or exclusion of code to cause HASP to intercept the WTL SVC (SVC 36) and to add to a job's output any log messages associated with it. The messages will be written on the HASP System Log for the job. The specification for &WTLOPT must be either YES or NO.

Default: &WTLOPT=NO

Notes:
1. If &WTLOPT is set to YES and &NUMCONS is set non-zero, no messages will be recorded on the OS log data sets and the OS WRITELOG command must not be used.
2. If in addition to the conditions of Note 1 above, &LOGOPT is set to NO, all WTL messages will be thrown away.

&WTR

Explanation: Variable symbol &WTR specifies the unit address of a pseudo-1403 printer to be used by a writer to retrieve from the OS Job Queue System Message Blocks (SMBs) for jobs controlled by HASP. The specification must be a valid unit address which has been specified at SYSGEN time as a pseudo-1403 printer.

Default: &WTR=0FE

Notes:
1. The unit address assigned to this parameter must not be assigned a symbolic unit name at SYSGEN time, as described for other pseudo-1403 printers.

&WTRCLAS

Explanation: Variable symbol &WTRCLAS specifies the OS System Output classes to be processed by HASP. The output writer started by HASP initialization (and selected by the &WTRPART Parameter) is intended to process only those System Message Blocks (SMBs) created by OS jobs submitted to and controlled by HASP. If other OS writers are to be used concurrently with the writer started by HASP, none of them may process any of the output classes specified in &WTRCLAS.

The specification for &WTRCLAS must be one to eight unique characters that are valid OS output classes.

Default: &WTRCLAS=HA

Notes:

1. HASP examines the MSGCLASS parameter of every JOB card it sends to OS. If MSGCLASS is not specified or is not one of the classes specified by &WTRCLAS, HASP adds the MSGCLASS parameter to the JOB card, using as a class the leftmost character of &WTRCLAS.
2. If a job submitted to OS by HASP has certain errors on the JOB card, OS will fail the job and change its MSGCLASS to A. It is therefore recommended that class A be specified in &WTRCLAS. If class A is not specified and such an error happens, HASP may not operate correctly.
3. See also HASPGEN parameter $$x.

&WTRPART

Explanation: Variable symbol &WTRPART specifies the method HASP will use to retrieve from the OS Job Queue System Message Blocks for jobs controlled by HASP.

For &WTRPART=*, HASP initialization creates a subtask to interface directly between HASP and the OS Job Queue.

If &WTRPART is not specified as *, HASP initialization starts an OS writer (using procedure HOSWTR) to interface between HASP and the OS Job Queue. In particular, for MFT systems HASP initialization issues the OS command

```
S  HOSWTR.&WTRPART,&WTR,,&WTRCLAS
```

The specification for &WTRPART must be either * or, for MVT, any other character string of one to eight characters or, for MFT, a valid identifier for an OS START writer command, as described in the OS Operator's Guide.

Default: &WTRPART=*

Notes:

1. For an OS MFT system, the default specification requires that, during SYSGEN, the SUPRVSOR macro include ATTACH in the OPTIONS=keyword.
2. If &WTRPART is not specified as *, it is recommended for an MFT system that HOSWTR be assigned the partition immediately below HASP in priority. That is, if HASP were to be assigned P0, &WTRPART would be specified as P1.

&XBATCHC

Explanation: Variable symbol &XBATCHC specifies a list of job classes to be used with the HASP Execution Batch Scheduling feature. The specified classes are excluded from running jobs outside of Execution Batch Scheduling. The specification for &XBATCHC is a string of one to eight characters (letters and numbers) which specify valid unique HASP job classes. If &XBATCHC is left at its default, the generated HASP system will not include Execution Batch Scheduling.

Default: &XBATCHC=[null string]

Notes:
1. For further information, see the section of this manual on the Execution Batch Scheduling feature.
2. If &XBATCHC is not specified, then &XBATCHN is not used.

&XBATCHN

Explanation: Variable symbol &XBATCHN specifies the first five characters of the name of each OS job to be started internally by HASP when required for the execution of a user "job" under the HASP Execution Batch Scheduling feature. The specification must be a five-character string of which the first character is alphabetic or national and the remaining four are alphameric or national.

Default: &XBATCHN=$$$$$

Notes:
1. For further information, see the section of this manual on the Execution Batch Scheduling feature.
2. If &XBATCHC is specified, then HASP will reject all user submitted jobs whose jobnames start with the five characters &XBATCHN.

&XLIN(n)

Explanation: Subscripted variable symbols &XLIN(n) specify estimated line counts. If a user specifies in the accounting field of his job card an estimated line count of l=p/1000, then the tentative job priority computed on the basis of his estimated execution time will be reduced by n, where

&XLIN(n)<p≤&XLIN(n+1).

Each specification must be an integer between 1 and 16,777,215. &XLIN(9) must be 16,777,215.

Default:
```
&XLIN(1)=2000
&XLIN(2)=5000
&XLIN(3)=15000
&XLIN(4)=X'FFFFFF'
&XLIN(5)=X'FFFFFF'
&XLIN(6)=X'FFFFFF'
&XLIN(7)=X'FFFFFF'
&XLIN(8)=X'FFFFFF'
&XLIN(9)=X'FFFFFF'
```

Notes:

1. The values &XLIN(n) must be in the same order as the subscript n.
2. See also Note 2 for &RPRT(n).
3. These values are not used if the job uses a /*PRIORITY HASP control card.
4. See also the description of &XPRI(n), used with &XLIN(n) to determine a job's printing priority.

&XPRI(n)

Explanation: Subscripted variable symbols &XPRI(n) specify job priorities for printing which correspond to intervals defined by subscripted variable symbols &XLIN(n). If a user does not supply a /*PRIORITY control card with his job, the job's priority is recomputed after execution based upon the actual number of print lines it produced. If the job produced p print lines then its priority for printing and punching will become &XPRI(n), where n is the smallest number for which

$p \leq$ &XLIN(n).

Each specification must be an integer between 0 and 15.

Default:
&XPRI(1)=9
&XPRI(2)=8
&XPRI(3)=7
&XPRI(4)=6
&XPRI(5)=5
&XPRI(6)=4
&XPRI(7)=3
&XPRI(8)=2
&XPRI(9)=1

&XZMFTH

Explanation: Variable symbol &XZMFTH specifies the dispatching priority of the highest-priority MFT task to be included in the group of tasks analyzed by the HASP Execution Task Monitor. Each MFT HASP-controlled job step task without subtasks whose dispatching priority falls within the range &XZMFTL through &XZMFTH is examined by the HASP Execution Task Monitor every &MONINTV seconds. In order to balance the CPU utilization characteristics of these tasks, the Execution Task Monitor resets the dispatching priority of each of them to &XZMFTL and, if necessary, changes their order on the TCB ready chain. The specification for &XZMFTH must be one or two hexadecimal characters ranging from 00 to FF.

Default: &XZMFTH=FF

Notes:

1. If &XZMFTH is specified as 00, Execution Task Monitor support for MFT is excluded from the generated HASP system.
2. If &XZMFTH is specified as 00 and &XZPRTY is specified as 0-1, then &MONINTV must be specified as 0.

&XZMFTL

Explanation: Variable symbol &XZMFTL specifies the dispatching priority of the lowest-priority MFT task to be included in the group of tasks analyzed by the HASP Execution Task Monitor. Each MFT HASP-controlled job step task without subtasks whose dispatching priority falls within the range &XZMFTL through &XZMFTH is examined by the HASP Execution Task Monitor every &MONINTV seconds. In order to balance the CPU utilization characteristics of these tasks, the Execution Task Monitor resets the dispatching priority of each of them to &XZMFTL and, if necessary, changes their order on the TCB ready chain. The specification for &XZMFTL must be one or two hexadecimal characters ranging from 00 to FF.

Default: &XZMFTL=00

Notes:
1. If &XZMFTH is specified as 00, &XZMFTL is not used and Execution Task Monitor support for MFT is excluded from the generated HASP system.

&XZMULT

Explanation: Variable symbol &XZMULT specifies whether or not tasks which are part of a multi-tasking jobstep are to be included in the group of tasks analyzed by the Execution Task Monitor. A specification of YES indicates that all tasks for jobs controlled by HASP running at eligible OS dispatching priorities will be monitored. Tasks within a jobstep will maintain relative priorities as assigned by ATTACH and CHAP macros. A specification of NO indicates that multi-tasking jobstep tasks are not to be monitored. The specification must be either YES or NO.

Default: &XZMULT=YES

Notes:

1. In an MFT system all monitored tasks are assigned an OS dispatching priority specified by the &XZMFTL parameter. A CHAP or ATTACH specifying a change in priority will have the effect of changing relative priority; however, as long as the task remains within the range &XZMFTH and &XZMFTL the assigned priority will be changed to &XZMFTL at the end of the monitor interval &MONINTV.

(The remainder of this page intentionally left blank.)

&XZPRTY

Explanation: Variable symbol &XZPRTY specifies a priority value used by the HASP Execution Task Monitor for MVT systems. The Execution Task Monitor periodically analyzes each MVT HASP-controlled job step task, without subtasks, whose dispatching priority is &XZPRTY*16+11. In order to balance the CPU utilization characteristics of these tasks, the Execution Task Monitor may re-order these tasks on the TCB ready chain. The specification for &XZPRTY must be an integer between 0 and 15 inclusive, or the expression "0-1". The latter value should be used when the Execution Task Monitor is to operate for MFT but not for MVT.

Default: &XZPRTY=7

Notes:
1. If &MONINTV=0, the value of &XZPRTY is not used.
2. If &XZPRTY is specified as 0-1 and &XZMFTH is specified as 0, then &MONINTV must be set to 0.

$$x

Explanation: Ordinary symbol $$x specifies the destination for an output data set designated in the user's JCL as SYSOUT=x. The specification for each of these ordinary symbols must be one of the characters A, B, 1, 2, or *.

For $$x=A, associated SYSOUT data sets will be printed with the user's job.

For $$x=B, associated SYSOUT data sets will be punched with the user's job.

For $$x=1, associated SYSOUT data sets will be added to the HASP special forms queue, to be printed with other SYSOUT data sets requiring the same forms.

For $$x=2, associated SYSOUT data sets will be added to the HASP special forms queue, to be punched with other SYSOUT data sets requiring the same forms.

For $$x=*, associated SYSOUT data sets will be processed entirely by OS. In this case, HASP will add the specification UNIT=SYSDA to the JCL, unless the user has himself specified UNIT=information.

Default:

$$A=A	$$S=A
$$B=B	$$T=A
$$C=A	$$U=A
$$D=A	$$V=A
$$E=A	$$W=A
$$F=A	$$X=A
$$G=A	$$Y=A
$$H=A	$$Z=A
$$I=A	$$1=A
$$J=1	$$2=A
$$K=2	$$3=A
$$L=A	$$4=A
$$M=A	$$5=A
$$N=A	$$6=A
$$O=A	$$7=A
$$P=A	$$8=A
$$Q=A	$$9=A
$$R=A	$$0=A

Notes:

1. For any output class x, regardless of the value specified for $$x, a four-digit special forms number can be coded as the third positional parameter of the SYSOUT=keyword. The specification is converted to a packed number (unless $$x=*); that is, forms number 0001 is the same as forms number 01.
2. For an output class x for which $$x=*, the SYSOUT=parameter may be coded as described in the OS Job Control Language Reference manual.
3. A user SYSOUT specification which includes the second positional parameter (program name) will be processed entirely by OS, regardless of whether the associated $$ parameter was specified as *.
4. If a given output class x is one of the classes assigned to &WTRCLAS, it must not be used in a SYSOUT specification to be processed by OS (caused if $$x=*, or if the second parameter of SYSOUT is used), unless that class x is subject to requeueing as described under the parameter &WCLSREQ.

7.2 RMTGEN PARAMETERS FOR SYSTEM/360 MODEL 20 STR

This section describes the parameters used in assembly of the System/360 Model 20 STR Remote Terminal Program for HASP Remote Job Entry. The parameters are used during RMTGEN to specify hardware configuration and software options.

For each parameter there is an explanation, the default value, and frequently notes which expand upon the explanation.

The parameters are listed in alphabetical order.

&CCT

Explanation: Variable symbol &CCT specifies the degree of text compression to be provided in the program. For text transmission to HASP, the program will compress strings of &CCT or more identical characters. The specification must be an integer between 3 and 80 inclusive.

Default: &CCT=4

Notes:
1. Low values of &CCT cause creation of highly compact records, increasing effective line speed at the expense of CPU

&CORESIZ

Explanation: Variable symbol &CORESIZ specifies the amount of main storage available to the program in K bytes. The specification must be an integer greater than 0.

Default: &CORESIZ=8.

&NUMBUFS

Explanation: Variable symbol &NUMBUFS specifies the maximum number of buffers to be used by the program. The specification must be an integer greater than or equal to 2.

Default: &NUMBUFS=10

Notes:
1. The length of each buffer is given by HASPGEN parameter &TPBFSIZ.

&PUNCH

Explanation: Variable symbol &PUNCH specifies inclusion (&PUNCH=1) or exclusion (&PUNCH=0) of support for a card punch attached to the Model 20. The specification must be either 0 or 1.

Default: &PUNCH=1

Notes:
1. The program supports 1442, 2520, and 2560 card punches interchangeably.

7.3 RMTGEN PARAMETERS FOR SYSTEM/360 MODEL 20 BSC

This section describes the parameters used in assembly of the System/360 Model 20 BSC Remote Terminal Program for HASP MULTI-LEAVING Remote Job Entry. The parameters are used during RMTGEN to specify hardware configuration and software options.

For each parameter there is an explanation, the default value, and frequently notes which expand upon the explanation.

The parameters are listed in alphabetical order.

&CCT

Explanation: Variable symbol &CCT specifies for all text compression but trailing blank compression the minimum number of characters to be compressed. A duplicate character string of fewer than &CCT characters will be treated as a string of non-duplicate characters for compression purposes. The specification must be an integer between 3 and 31, inclusive.

Default: &CCT=4

Notes:
1. See also &CMPTYPE. The value of &CCT is not used if &CMPTYPE=1.
2. A smaller value of &CCT increases efficiency of communication line usage at the expense of compute time required for compression.

&CMPTYPE

Explanation: Variable symbol &CMPTYPE specifies the type of compression to be applied to all text transmitted from the Model 20 to the central computer. The specification must be either 1, 2, or 3. The value 1 specifies trailing blank compression; 2 specifies compression of leading, embedded, and trailing blanks, and 3 specifies compression of all duplicate character strings.

Default: &CMPTYPE=2

Notes:
1. See also &CCT.

&CORESIZ

Explanation: Variable symbol &CORESIZ specifies the size of Model 20 main storage in Kbytes (1 Kbyte = 1024 bytes). The specification must be an integer between 8 and 32 inclusive.

Default: &CORESIZ=8

&ERRMSGN

Explanation: Variable symbol &ERRMSGN specifies the number of four-byte entries to be assembled in the Model 20 Remote Terminal program as an error message log table. The specification must be an integer not less than 8.

Default: &ERRMSGN=10

&LINESPD

Explanation: Variable symbol &LINESPD specifies the speed, in baud, of the communication line to be used between the Model 20 and the central computer. The specification must be a positive integer.

Default: &LINESPD=2000

&NUMBUFS

Explanation: Variable symbol &NUMBUFS specifies number of teleprocessing buffers to be constructed by the Model 20 Remote Terminal program. The specification must be an integer no less than given by the formula

$$2*X+1$$

where

X=1 if either a 2520 or a 2560 is to be used as both a reader and a punch, or
0 otherwise.

Default: &NUMBUFS=8

Notes:

1. The length of each buffer is &MLBFSIZ+5 bytes (rounded up to the next full word); the value of HASPGEN parameter &MLBFSIZ is automatically propagated to RMTGEN.
2. If &NUMBUFS specifies more buffers than can be built in available storage, the Remote Terminal program will build as many buffers as it can.
3. It is recommended that at least two buffers be furnished for each output device and for the communication adapter.

&NUMTANK

Explanation: Variable symbol &NUMTANK specifies the number of decompression buffers ("decompression tanks") to be assembled in the Model 20 Remote Terminal program. The specification should be an integer not less than 2.

Default: &NUMTANK=8

Notes:
1. The length of each decompression tank is &PRTSIZE+6.
2. It is recommended that at least two tanks each be provided for the printer and the punch.
3. For an 8K Model 20, specification of &NUMTANK greater than 8 may cause the Remote Terminal program to assemble larger than X'1F00' bytes (8K-256); the resultant program will fail to load.

&PDEV(1)

Explanation: Subscripted variable symbol &PDEV(1) specifies device type for the Model 20 printer. The specification must be either 1403 or 2203.

Default: &PDEV(1)=2203

&PRTCONS

Explanation: Variable symbol &PRTCONS specifies the degree of use of the printer as an output console and is dependent upon the specifications used in the generation of the HASP System pertaining to the handling of messages for the remote as follows:

1. If HASP is told that the remote has a console via the RMTnn HASPGEN parameter, &PRTCONS has the following meanings:

   &PRTCONS=0 - Error logging and display will be suppressed and operator messages created while the remote is on-line to HASP will be discarded.
   &PRTCONS=1 - The printer will be used as an output console when sufficient operator messages from HASP have been queued for output at the remote. If the printer is busy with job stream output, that output will be interrupted for the printing of operator messages from HASP and error messages from the remote error log. When the console queue is empty, job stream output will continue.
   &PRTCONS=2 - The printer will be used as an output console but will not interrupt the printing of jobs. Operator messages received from HASP while jobs are being printed will be discarded.

2. If HASP is told that the remote does not have a console via the RMTnn HASPGEN parameter and HASP does not have message SPOOLing capability as determined by the &SPOLMSG HASPGEN parameter &PRTCONS has the following meanings:

   &PRTCONS=0 - Error logging and display will be suppressed and no operator messages will be displayed.
   &PRTCONS=1 - Error log messages will be displayed when the printer is free to print them (no job interruptions).
   &PRTCONS=2 - Same as &PRTCONS=1

3. If HASP is told that the remote does not have a console via the RMTnn HASPGEN parameter and HASP does have message SPOOLing capability as determined by &SPOLMSG HASPGEN parameter &PRTCONS takes on the same meanings as 2 above with the additional capability of printing operator messages queued for the remote by HASP and transmitted to the remote when the printer is free and is set to receive messages by the $TRMr.PR1 command.

Settings for &PRTCONS must be 0, 1, or 2.

Default: &PRTCONS=0

Notes:
1. If &WDEV(1) is not zero &PRTCONS should be set to zero.
2. See HASPGEN parameters RMTnn and &SPOLMSG.
3. Regardless of the settings of the &WDEV(1) and &PRTCONS parameters error messages resulting from loggable errors detected by the remote will be discarded when the errors occur at a rate faster than the output device can display them.

(The remainder of this page intentionally left blank.)

&PRTSIZE

Explanation: Variable symbol &PRTSIZE specifies the length in bytes of the text portion of each decompression tank. Each tank must be long enough to hold a maximum-length output record to either the printer, the punch, or the operator console. The specification must be an integer that is the largest of 80 (if &UDEV(1) is not zero), 120 (if &WDEV(1) is not zero), and the line width of the printer.

Default: &PRTSIZE=120

&RADR(1)

Explanation: Subscripted variable symbol &RADR(1) specifies the unit address of the Model 20 card reader. The specification must correspond to the specification for &RDEV(1) as follows:

&RDEV(1)	&RADR(1)
2501	1
2520	2
2560	2

Default: &RADR(1)=1

&RDEV(1)

Explanation: Subscripted variable symbol &RDEV(1) specifies device type for the Model 20 card reader. The specification must be either 2501, 2520, or 2560.

Default: &RDEV(1)=2501

Notes:
1. See also &RADR(1)

&SUBMOD

Explanation: Variable symbol &SUBMOD specifies the Submodel number of the System/360 Model 20 for the specified Remote Terminal. The specification must be a valid System/360 Model 20 Submodel number.

Default: &SUBMOD=2

&UADR(1)

Explanation: Subscripted variable symbol &UADR(1) specifies the unit address of the Model 20 card punch. The specification must correspond to the specification for &UDEV(1) as follows:

&UDEV(1)	&UADR(1)
1442	3
2520	2
2560	2
0	not used

Default: &UADR(1)=3

&UDEV(1)

Explanation: Subscripted variable symbol &UDEV(1) specifies device type for the Model 20 card punch. The specification must be either 1442, 2520, 2560, or 0. Specification 0 is used when the Model 20 does not include a card punch.

Default: &UDEV(1)=1442

Notes:
1. See also &UADR(1), unless &UDEV(1)=0.

&WDEV(1)

Explanation: Subscripted variable symbol &WDEV(1) specifies device type for the Model 20 console. The specification must be either 2152 (if a console is present) or 0 (if no console is present).

Default: &WDEV(1)=0

Notes:
1. If &WDEV(1)=1, console support must be indicated for this Remote Terminal at HASPGEN time. See HASPGEN parameter RMTnn.

&WTOSIZE

Explanation: Variable symbol &WTOSIZE specifies the maximum length in bytes of a HASP operator command to be transmitted from the Model 20 to the central computer. The specification must be a positive integer not greater than 120.

Default: &WTOSIZE=120

Notes:
1. If &WDEV(1)=0, this parameter is not used.

&XPARENT

Explanation: Variable symbol &XPARENT specifies presence or absence of the text transparency feature. If the Binary Synchronous Communication Adapters at both the Model 20 and the central computer have the text transparency feature, YES should be specified; otherwise NO should be specified.

Default: &XPARENT=YES

7.4 RMTGEN PARAMETERS FOR SYSTEM/360 (EXCEPT MODEL 20) BSC

This section describes the parameters used in assembly of the System/360 BSC Remote Terminal Program for HASP MULTI-LEAVING Remote Job Entry. The parameters are used during RMTGEN to specify hardware configuration and software options.

For each parameter there is an explanation, the default value, and frequently notes which expand upon the explanation.

The parameters are listed in alphabetical order.

&ADAPT

Explanation: Variable symbol &ADAPT specifies the unit address of the Binary Synchronous Communication Adapter to be used by the System/360 Remote Terminal to communicate with HASP at the central computer. The specification must be a valid unit address.

Default: &ADAPT=020

&CCT

Explanation: Variable symbol &CCT specifies for all text compression but trailing blank compression the minimum number of characters to be compressed. A duplicate character string of fewer than &CCT characters will be treated as a string of non-duplicate characters for compression purposes. The specification must be an integer between 3 and 31, inclusive.

Default: &CCT=4

Notes:

1. See also &CMPTYPE. The value of &CCT is not used if &CMPTYPE=1.
2. A smaller value of &CCT increases efficiency of communication line usage at the expense of compute time required for compression.

&CMPTYPE

Explanation: Variable symbol &CMPTYPE specifies type of compression to be applied to all text transmitted from the System/360 Remote Terminal to the central computer. The specification must be either 1, 2, or 3. The value 1 specifies trailing blank compression; 2 specifies compression of leading, embedded, and trailing blanks, and 3 specifies compression of all duplicate character strings.

Default: &CMPTYPE=2

Notes:
1. See also &CCT.

&CORESIZ

Explanation: Variable symbol &CORESIZ specifies the size of main storage for the System/360 Remote Terminal in K bytes (1K byte = 1024 bytes). The specification must be an integer between 8 and 32 inclusive. If the System/360 is larger than 32K bytes, &CORESIZ must be specified as 32.

Default: &CORESIZ=8

&ERRMSGN

Explanation: Variable symbol &ERRMSGN specifies the number of four-byte entries to be assembled in the System/360 Remote Terminal as an error message log table. The specification must be an integer not less than 8.

Default: &ERRMSGN=10

&LINESPD

Explanation: Variable symbol &LINESPD specifies the speed, in baud, of the communication line to be used between the System/360 Remote Terminal and the central computer. The specification must be a positive integer.

Default: &LINESPD=2000

&MACHINE

Explanation: Variable symbol &MACHINE specifies the model number of the System/360 to be used as a HASP Remote Terminal. The specification must be a valid System/360 model number for a System/360 which includes the standard instruction set and the decimal instruction set.

Default: &MACHINE=30

&NUMBUFS

Explanation: Variable symbol &NUMBUFS specifies number of teleprocessing buffers to be constructed by the System/360 Remote Terminal program. The specification must be an integer no less than given by the formula

2*X+1

where

X=1 if either a 2520 or a 1442 is to be used as both a reader and a punch, or
0 otherwise.

Default: &NUMBUFS=8

Notes:

1. The length of each buffer is &MLBFSIZ+5 bytes (rounded up to a multiple of 4); the value of HASPGEN parameter &MLBFSIZ is automatically propagated to RMTGEN.
2. If &NUMBUFS specifies more buffers than can be built in available storage, the Remote Terminal program will build as many buffers as it can.
3. It is recommended that at least two buffers be furnished for each output device and for the communication adapter.

&NUMTANK

Explanation: Variable symbol &NUMTANK specifies the number of decompression buffers ("decompression tanks") program. The specification should be an integer not less than 2.

Default: &NUMTANK=5

Notes:
1. The length of each decompression tank is &PRTSIZE+6.
2. It is recommended that at least two tanks be provided for each printer and each punch (3 for a 2540 punch).

&PADR(n)

Explanation: Subscripted variable symbols &PADR(n) specify unit addresses for the printers defined by &PDEV(n). For each &PDEV(n) not specified as zero, the corresponding symbol &PADR(n) must specify the device's 3-character hexadecimal unit address.

Default: &PADR(1)=00E
&PADR(2)=00F
&PADR(3)=FFF
&PADR(4)=FFF
&PADR(5)=FFF
&PADR(6)=FFF
&PADR(7)=FFF

&PDEV(n)

Explanation: Subscripted variable symbols &PDEV(n) specify the existence and device types of the Remote Terminal printers. Each specification must be either 1403, 1443, or 0. A specification of 0 indicates that the associated printer does not exist.

Default:
```
&PDEV(1)=1403
&PDEV(2)=0
&PDEV(3)=0
&PDEV(4)=0
&PDEV(5)=0
&PDEV(6)=0
&PDEV(7)=0
```

Notes:

1. If &PDEV(n) is specified as a device type, then &UDEV(8-n) must be specified as zero.
2. If &PDEV(n+1) is specified as a device type, then &PDEV(n) must be specified as a device type.
3. If more than one printer is specified, a Device Control Table (DCT) for each additional printer must be added to the HASP System.

&PRTSIZE

Explanation: Variable symbol &PRTSIZE specifies the length in bytes of the text portion of each decompression tank. Each tank must be long enough to hold a maximum-length output record to either a printer, a punch, or the operator console. The specification must be an integer that is the larger of 120 and the line width of the widest printer.

Default: &PRTSIZE=132

&RADR(n)

Explanation: Subscripted variable symbols &RADR(n) specify unit addresses for the readers defined by &RDEV(n). For each &RDEV(n) not specified as zero, the corresponding symbol &RADR(n) must specify the device's 3-character hexadecimal unit address.

Default: &RADR(1)=00C
&RADR(2)=FFF
&RADR(3)=FFF
&RADR(4)=FFF
&RADR(5)=FFF
&RADR(6)=FFF
&RADR(7)=FFF

&RDEV(n)

Explanation: Subscripted variable symbols &RDEV(n) specify the existence and device types of the Remote Terminal readers. Each specification must be either 2540, 2501, 2520, 1442, or 0. A specification of 0 indicates that the associated reader does not exist.

Default: &RDEV(1)=2540
&RDEV(2)=0
&RDEV(3)=0
&RDEV(4)=0
&RDEV(5)=0
&RDEV(6)=0
&RDEV(7)=0

Notes:
1. If &RDEV(n+1) is specified as a device type, then &RDEV(n) must be specified as a device type.
2. If more than one reader is specified, a Device Control Table (DCT) for each additional reader must be added to the HASP System.

&UADR(n)

Explanation: Subscripted variable symbols &UADR(n) specify unit addresses for the punches defined by &UDEV(n). For each &UDEV(n) not specified as zero, the corresponding symbol &UADR(n) must specify the device's 3-character hexadecimal unit address.

Default:
```
&UADR(1)=00D
&UADR(2)=FFF
&UADR(3)=FFF
&UARD(4)=FFF
&UARD(5)=FFF
&UARD(6)=FFF
&UARD(7)=FFF
```

&UDEV(n)

Explanation: Subscripted variable symbols &UDEV(n) specify the existence and device types of the Remote Terminal punches. Each specification must be either 2540, 2520, 1442, or 0. A specification of 0 indicates that the associated punch does not exist.

Default: &UDEV(1)=2540
&UDEV(2)=0
&UDEV(3)=0
&UDEV(4)=0
&UDEV(5)=0
&UDEV(6)=0
&UDEV(7)=0

Notes:
1. If &UDEV(n) is specified as a device type, then &PDEV(8-n) must be specified as zero.
2. If &UDEV(n+1) is specified as a device type, then &UDEV(n) must be specified as a device type.
3. If more than one punch is specified, a Device Control Table (DCT) for each additional punch must be added to the HASP System.

&WADR(1)

Explanation: Subscripted variable symbol &WADR(1) specifies the unit address of the 1052 operator console on the System/360 Remote Terminal. The specification must be a 3-character hexadecimal unit address.

Default: &WADR(1)=01F

&WTOSIZE

Explanation: Variable symbol &WTOSIZE specifies the maximum length in bytes of a HASP operator command to be transmitted from the System/360 Remote Terminal to the central computer. The specification must be a positive integer not greater than 120.

Default: &WTOSIZE=120

&XPARENT

Explanation: Variable symbol &XPARENT specifies presence or absence of the text transparency feature. If the Binary Synchronous Communication Adapters at both the System/360 Remote Terminal and the central computer have the text transparency feature, YES should be specified; otherwise NO should be specified.

Default: &XPARENT=YES

7.5 RMTGEN PARAMETERS FOR 1130

This section describes the parameters used in assembly of the 1130 Remote Terminal Program for HASP MULTI-LEAVING Remote Job Entry. The parameters are used during RMTGEN to specify hardware configuration and software options.

For each parameter there is an explanation, the default value, and frequently notes which expand upon the explanation.

The parameters are listed in alphabetical order.

&CLOCK

Explanation: The variable symbol &CLOCK is used to specify the type of communication adapter clocking available on the 1130 to be used by the workstation program. The specification of &CLOCK=0 is interpreted to mean that data set clocking is being used. The value &CLOCK=1 specifies internal (1130) clocking.

Default: &CLOCK=0

Notes:

1. The rate of insertion of the synchronous idle sequence in the transmitted data is determined by the variables &CLOCK, &LINESPD and &TRANPRN. The relationship of these variables to the insertion rate is:

&CLOCK	&TRANPRN	INSERTION EVERY:	
0	0	&LINESPD/8	characters
0	1	&LINESPD/8	characters
1	0	70	characters
1	1	&LINESPD/8	characters

2. The equation used for the insertion rate is:

   (&LINESPD/8)*T

   where T is 1.00 second which is the nominal 2701 timer value.

&CMPTYPE

Explanation: The variable symbol &CMPTYPE is used to specify the compression technique that is to be applied to the data transmitted to the central HASP system. The choices for &CMPTYPE are:

&CMPTYPE=0 for no compression of duplicate characters or truncation of trailing blanks.

&CMPTYPE=1 for trailing blank truncation only.

&CMPTYPE=2 for full compression: trailing blank truncation and encoding of duplicate characters.

Default: &CMPTYPE=2

Notes:

1. The process of compressing input data offers optimum performance with respect to efficient line utilization. However, the factors of line speed, CPU availability, buffer size, line turn-around time, nature of the data to be compressed, etc., are variables which contribute to the overall operation of the workstation program. Since compression and truncation require considerable CPU time, the user may decide, on the basis of the other variables, to respecify the compression technique.

&DELAY

Explanation: The variable symbol &DELAY is used to define the number of intervals of time that RTP1130 will delay in transmitting a "handshaking" sequence (DLE-ACK0) to the central HASP site. The hardware program timer clock is used to measure the delay and is assumed to be set to a nominal value of .35 seconds.

Default: &DELAY=3

Notes:
1. &DELAY=3 results in a delay of 1.05 seconds, assuming a timer interval of .35 seconds.
2. The purpose of the delay when "handshaking" is to minimize CPU processing at the central HASP computer when no data is being transmitted.
3. The value of &DELAY must not be set to such a large increment that the delay will be greater than the timeout period of the central site 2701/2703.

&FULLIST

Explanation: The variable symbol &FULLIST is used to specify the type of assembly listing which is produced by the OS/360 assembler during the RMTGEN process. If the value of &FULLIST is set to 0, then the assembly listing produced will be according to the PRINT NOGEN stipulation of the assembler. If the value of &FULLIST is set to 1, the listing will be produced according to the PRINT GEN stipulation.

Default: &FULLIST=1

Notes:

1. Since most of the code in RTP1130 and RTPLOAD is created by Macro instructions, the specification of &FULLIST=0 will essentially produce a source listing (cross referenced) without the 1130 assembled instructions. Error messages will not appear on the listing.

&LINESPD

Explanation: The variable symbol &LINESPD is used to specify the baud rate for the communication line interface to the workstation program. The value should correspond to the selected setting of the baud rate switch on the 1130 SCA control panel: 1200,2000,...,etc.

Default: &LINESPD=2000

Notes:

1. The rate of insertion of the synchronous idle sequence (DLE-SYN or SYN-SYN) in the transmitted data is determined by the variables &CLOCK, &LINESPD and &TRANPRN. See note 1 of &CLOCK description.

&MACHSIZ

Explanation: Variable symbol &MACHSIZ specifies the amount of 1130 core to be used by RTP1130. The value of &MACHSIZ is in units of 1130 words.

Default: &MACHSIZ=8192

Notes:
1. The value of &MACHSIZ is interpreted to mean that "&MACHSIZ" number of words, starting at location 0, are available for the workstation program consisting of RTPBOOT, RTPLOAD and RTP1130.
2. The same variable symbol must be defined for RTPLOAD and should have the same value.
3. The value of &MACHSIZ may be less than the actual available storage but must not be greater.

&PN1442

Explanation: The variable symbol &PN1442 is used to define a 1442 punch. If the variable is set to 1, then RTP1130 will include support for punched card output produced by jobs at the Central HASP site. If the variable is set to 0, no support for the 1442 punch will be provided. See &RD1442 for the definition of a reader function on the 1442.

Default: &PN1442=1

&PRFOTLW

Explanation: The value of the variable symbol &PRFOTLW is used to define the line width of the 1403 printer specified by &PR1403. The choices are 120 or 132 character lines.

Default: &PRFOTLW=120

Notes:
1. The definition of the line width for all printers on a particular remote is a HASPGEN requirement. See HASPGEN parameter RMTnn.

&PR1132

Explanation: The variable symbol &PR1132 is used to define an 1132 printer. If the variable is set to 1, then RTP1130 will include support for the 1132 to print job output. If the variable is set to 0, no support will be included in RTP1130 for the 1132.

Default: &PR1132=0

&PR1403

Explanation: The variable symbol &PR1403 is used to define a 1403 printer for use as an output device. If the value of &PR1403 is 1, then the 1403 function will be included in RTP1130. If the value is 0, the function is deleted from RTP1130.

Default: &PR1403=1

Notes:
1. See &PRFOTLW for specifying the line width of the 1403.

&RD1442

Explanation: The variable symbol &RD1442 is used to define a 1442 as a card reader. If the variable is set to 1, then RTP1130 will be assembled with all necessary control blocks and support routines to provide job input from the 1442. If the variable is set to 0, no support for the 1442 reader will be provided in RTP1130. See &PN1442 for a definition of the punch function on the 1442.

Default: &RD1442=1

Notes:

1. If the variable &RD1442 is set to 1 and a 1442 reader does not exist then the operation of the workstation program may be unpredictable.

&RD2501

Explanation: The variable symbol &RD2501 is used to define a 2501 card reader. If the variable is set to 1, then RTP1130 will be assembled with all necessary control block and subroutines to support the 2501 as a job input device. If the variable is set to 0, no support for the 2501 will be included in RTP1130.

Default: &RD2501=0

Notes:

1. If the variable &RD2501 is set to 1 and a 2501 does not exist then the operation of the work-station program will be unpredictable and usually unproductive.

&RTPLORG

Explanation: The variable symbol &RTPLORG defines the origin in 1130 storage of the program loader RTPLOAD which is used to load RTP1130.

Default: &RTPLORG=2*(&MACHSIZ-1024)

Notes:

1. The value of the above expression, assuming &MACHSIZ=8192, is 14336 (which is twice the actual 1130 storage address because the value is used in an ORG operation and must be in terms of bytes not 1130 words.
2. The RTPLOAD program must origin in the storage available between the end of RTP1130 (beginning of buffer pool) and the end of defined (&MACHSIZ) storage MINUS the length of RTPLOAD. The default value of &RTPLORG allows for an RTPLOAD of 1024 words in size.

&TRANPRN

Explanation: The variable symbol &TRANPRN is used to define the simulation of the Binary Synchronous Transparency feature. If the value of &TRANPRN is set to 1, then RTP1130 will simulate the transparency feature in the same manner as the 2701 SDA-II adapter equipped with the transparency feature. If the variable is set to 0, no simulation will occur and therefore data which contains transparent characters cannot be properly processed by RTP1130.

Default: &TRANPRN=1

Notes:
1. If &TRANPRN=0 is specified, the conversion of card code data is monitored and all BSC control characters are converted to hexadecimal 0. This prevents mispunched data from causing an infinite error retry if the central site does not have transparency.
2. See &LINESPD and &CLOCK for additional influence of &TRANPRN.
3. If &TRANPRN=1, the generated Remote Terminal program will communicate only with a 2701 or 2703 adapter which has the text transparency feature.

7.6 RMTGEN PARAMETERS FOR 1130 LOADER

This section describes the parameters used in assembly of RTPLOAD, the 1130 Loader Program. RTPLOAD is used to load the 1130 Remote Terminal Program. RTPLOAD's three parameters specify machine size, loader origin, and an assembler list option.

For each parameter there is an explanation, the default value, and frequently notes which expand upon the explanation.

The RMTGEN processes produce the object decks for RTPLOAD and RTP1130. The bootstrap loader (RTPBOOT) cannot be produced on a System 360 and must be punched by keypunch as indicated in Section 4.14.3.

The parameters are listed in alphabetical order.

&FULLIST

Explanation: The variable symbol &FULLIST is used to specify the type of assembly listing which is produced by the OS/360 assembler during the RMTGEN process. If the value of &FULLIST is set to 0, then the assembly listing produced will be according to the PRINT NOGEN stipulation of the assembler. If the value of &FULLIST is set to 1, the listing will be produced according to the PRINT GEN stipulation.

Default: &FULLIST=1

Notes:

1. Since most of the code in RTP1130 and RTPLOAD is created by Macro instructions, the specification of &FULLIST=0 will essentially produce a source listing (cross referenced) without the 1130 assembled instructions. Error messages will not appear on the listing.

&MACHSIZ

Explanation: Variable symbol &MACHSIZ specifies the amount of 1130 core to be used by RTPLOAD. The value of &MACHSIZ is in units of 1130 words.

Default: &MACHSIZ=8192

Notes:
1. The value of &MACHSIZ is interpreted to mean that "&MACHSIZ" number of words, starting at location 0, are available for the workstation program consisting of RTPBOOT, RTPLOAD and RTP1130.
2. The same variable symbol must be defined for RTP1130 and should have the same value.
3. The value of &MACHSIZ may be less than the actual available storage but must not be greater.

&RTPLORG

Explanation: The variable symbol &RTPLORG defines the origin in 1130 storage of the program loader RTPLOAD which is used to load RTP1130.

Default: &RTPLORG=2*(&MACHSIZ-1024)

Notes:

1. The value of the above expression, assuming &MACHSIZ=8192, is 14336 (which is twice the actual 1130 storage address because the value is used in an ORG operation and must be in terms of bytes not 1130 words.
2. The RTPLOAD program must origin in the storage available between the end of RTP1130 (beginning of buffer pool) and the end of defined (&MACHSIZ) storage MINUS the length of RTPLOAD. The default value of &RTPLORG allows for an RTPLOAD of 1024 words in size.

7.7 RMTGEN PARAMETERS FOR SYSTEM/3

This section describes the parameters used in assembly of the System/3 Remote Terminal Program for HASP MULTI-LEAVING Remote Job Entry. The parameters are used during RMTGEN to specify hardware configuration and software options.

For each parameter there is an explanation, the default value, and frequently notes which expand upon the explanation.

The parameters are listed in alphabetical order.

&COMP

Explanation: Variable symbol &COMP specifies degree of text compression to be provided for all text transmitted from the System/3 to HASP. The specification must be either 0, 1, or 2.

For &COMP=0, neither compression nor truncation is performed.

For &COMP=1, trailing blanks are truncated from each logical record before it is transmitted.

For &COMP=2, compression takes place after truncation. Strings of from two to 31 blanks are compressed to a single byte; strings of from three to 31 duplicate characters are compressed to two bytes.

Default: &COMP=2

&DEBUG

Explanation: Variable symbol &DEBUG specifies inclusion or exclusion of certain validity tests and a core dump program in the System/3 Remote Terminal Program. The specification must be either 0 or 1.

Default: &DEBUG=0

&DIAL, &DIAL1

Explanation: Variable symbols &DIAL and &DIAL1 specify the telephone number to be used during the initialization process. The values will be included on the default /*SIGNON card assembled into the System/3 Remote Terminal Program and preceded by the keyword DIAL (unless the parameters are left at their defaults). Each specification is a string of from one to eight decimal digits. If the telephone number is eight or fewer digits long, it should be specified by &DIAL. If the telephone number is longer than eight digits, its leftmost eight digits should be specified by &DIAL and the remaining digits by &DIAL1.

Default: &DIAL=[null string]
&DIAL1=[null string]

&MACHSIZ

Explanation: Variable symbol &MACHSIZ specifies the size of System/3 Core storage. The specification should be either 8192, 12288, 16384, 24576, or 32768 for core storage sizes of 8K, 12K, 16K, 24K or 32K respectively.

Default: &MACHSIZ=8192

&PASSWD

Explanation: Variable symbol &PASSWD specifies a password to be used during the SIGNON process. The value will be included on the default /*SIGNON card assembled into the System/3 Remote Terminal Program. The specification must be a character string of from one to eight characters. If blanks are desired, no specification may be made.

Default: &PASSWD=[null string]

&PC(n)

Explanation: Subscripted variable symbols &PC(n) specify skip information for the 5203 printer. The value to which &PC(n) is set will be the print line number to which paper will be skipped when the System/3 Remote Terminal Program simulates the 1403 command "Skip to Channel n". Each specification must be an integer between 0 and &S3FORML, inclusive. A specification of 0 causes no forms movement.

Default:
```
&PC(1)=1
&PC(2)=0
&PC(3)=0
&PC(4)=0
&PC(5)=0
&PC(6)=0
&PC(7)=0
&PC(8)=0
&PC(9)=0
&PC(10)=0
&PC(11)=0
&PC(12)=&S3FORML-5
```

&PRTCONS

Explanation: Variable symbol &PRTCONS specifies utilization of the 5203 printer as an operator's output console. The specification must be 0, 1, or 2.

For &PRTCONS=0, the 5203 printer will never be used as an operator's output console.

For &PRTCONS=1, the System/3 Remote Terminal Program will attempt to hold operator messages from HASP until a job has completed printing. However, if two or more MULTI-LEAVING buffers are received containing HASP operator messages, the 5203 will eject a page (skip to channel 1), print the HASP operator messages, eject another page, and resume printing its job.

For &PRTCONS=2, the System/3 Remote Terminal Program will throw away all operator messages while the 5203 is printing a job. While the 5203 is dormant, it will print any received messages.

Default: &PRTCONS=2

Notes:
1. If &S35471=1, the value of &PRTCONS is ignored and assumed to be zero.
2. Regardless of the setting of &PRTCONS, messages temporarily saved on disk for a remote terminal will be printed to the terminal as a job. Thus, they will always appear on the printer, even if another console exists. See also HASPGEN parameter &SPOLMSG.
3. If &PRTCONS is specified greater than zero, MULTI-LEAVING console support should be specified in HASPGEN parameter RMTnn for this remote.

&S3CMDS

Explanation: Variable symbol &S3CMDS specifies inclusion or exclusion, local to the System/3, of a command facility and commands to assist the System/3 operator. The specification must be either 0 or 1.

Default: &S3CMDS=0

Notes:
1. Commands available with this facility are explained in the System/3 Operator's Guide.

(The remainder of this page intentionally left blank.)

&S3FORML

Explanation: Variable symbol &S3FORML specifies the number of print lines on a page of the continuous forms which will be used in the 5203 printer. The specification must be an integer not less than 6.

Default: &S3FORML=66

&S3NPUNS

Explanation: Variable symbol &S3NPUNS specifies the maximum number of jobs that can be punching simultaneously at the System/3 Remote Terminal. The specification must be 1, 2, or 3. (A value of 3 allows simultaneous operation of both 5424 hoppers and the 1442 hopper as punches.)

Default: &S3NPUNS=1

Notes:

1. If &S3NPUNS is set to 2 or 3, extra device control tables must be added for the appropriate remote to the HASP System at HASPGEN time.

&S3NRDRS

Explanation: Variable symbol &S3NRDRS specifies the maximum number of job streams that can be reading simultaneously from the System/3 Remote Terminal. The specification must be 1, 2, or 3. (A value of 3 allows simultaneous operation of both 5424 hoppers and the 1442 hopper as readers.)

Default: &S3NRDRS=1

Notes:

1. If &S3NRDRS is set to 2 or 3, extra device control tables must be added for the appropriate remote to the HASP System at HASPGEN time.

&S3OBJDK

Explanation: Variable symbol &S3OBJDK specifies inclusion of a facility to punch Operating System object decks. Text transparency should be present. The specification should be 0 or 1.

If &S3OBJDK=1, each card of an OS object deck will be expanded and punched into two 96-column cards. These cards will be recognized when later read by a System/3 Remote Terminal program for which &S3OBJDK=1, and for each two 96-column cards read an OS object deck card image will be transmitted.

Default: &S3OBJDK=0

&S3SIP

Explanation: Variable symbol &S3SIP specifies usage of those bytes of System/3 core storage between X'100' and X'1FF', inclusive. The specification must be either 0 or 1. For &S3SIP=1, the System/3 Remote Terminal Program will not use the bytes; their values will be preserved for the use of the System/3 Card System Initialization Program.

Default: &S3SIP=0

&S3TRACE

Explanation: Variable symbol &S3TRACE specifies the number of four-byte entries in the System/3 Remote Terminal Program's internal error message table. The specification must be an integer greater than 1.

Default: &S3TRACE=10

&S3XPAR

Explanation: Variable symbol &S3XPAR specifies presence or absence of the EBCDIC text transparency feature. The specification should be 1 if both the central computer's communications adapter and the System/3 BSCA have the EBCDIC text transparency feature; otherwise the specification should be 0.

Default: &S3XPAR=0

Explanation: Variable symbol &S31442 specifies inclusion or exclusion of support for the 1442 Card Reader-Punch (RPQ). The specification must be 1 for inclusion and 0 for exclusion of 1442 support.

Default: &S31442=0

Notes:

1. If &S31442=1, the resultant System/3 Remote Terminal Program requires that a 1442 be present on the System/3.

&S35424

Explanation: Variable symbol &S35424 specifies inclusion or exclusion of support for the 5424 Multi-Function Card Unit. The specification must be 1 for inclusion or 0 for exclusion of 5424 support.

Default: &S35424=1

Notes:
1. If &S35424 is specified as 0, then &S31442 must be specified as 1.
2. See Chapter 10.3 for RMTGEN considerations for &S35424=0.
3. See the System/3 Operator's Guide in Chapter 11.9 for program loading considerations for &S35424=0.

(The remainder of this page intentionally left blank.)

&S35471

Explanation: Variable symbol &S35471 specifies presence or absence of a 5471 Printer-Keyboard on the System/3. The 5471 will be used as an operator's input/output console. The specification must be 1 if a 5471 is present; otherwise it must be 0.

Default: &S35471=0

Notes:
1. If console support is desired, HASPGEN parameter RMTnn for this remote must specify MULTI-LEAVING console support.
2. Regardless of the setting of &S35471, messages from HASP can print on the printer. See RMTGEN parameter &PRTCONS, note 2, and HASPGEN parameter &SPOLMSG.

&S35475

Explanation: Variable symbol &S35475 specifies presence or absence of a 5475 Data Entry Keyboard on the System/3. The 5475 will be used as an operator's input console. The specification must be 1 if a 5475 is present; otherwise it must be 0.

Default: &S35475=0

Notes:
1. If &S35471=1, this parameter is ignored.
2. If console support is desired, HASPGEN parameter RMTnn for this remote must specify MULTILEAVING console support.
3. For output console specification, see RMTGEN parameter &PRTCONS.

&S396COL

Explanation: Variable symbol &S396COL specifies inclusion or exclusion of the System/3 load-mode punch option. The specification must be either 0 or 1. If &S396COL is specified, the resultant System/3 Remote Terminal Program will be capable of receiving correctly the punched output of a System/3 RMTGEN.

Default: &S396COL=0

7.8 RMTGEN PARAMETERS FOR 2922

To generate a 2922 Remote Terminal Program for HASP MULTI-LEAVING RJE, the parameters and procedures documented in Sections 7.3 and 10.3 for the System/360 Model 20 BSC should be used, subject to the following discussion.

Some parameters should be specifically set. They are:

```
&PDEV(1)=1403
&PRTSIZE=132
&UDEV(1)=0
&WDEV(1)=2152, if the optional typewriter console is installed
&XPARENT=NO, if optional transparency is not installed
&LINESPD=xxxx (the actual line speed used)
```

Some parameters should not be altered from their default values. They are:

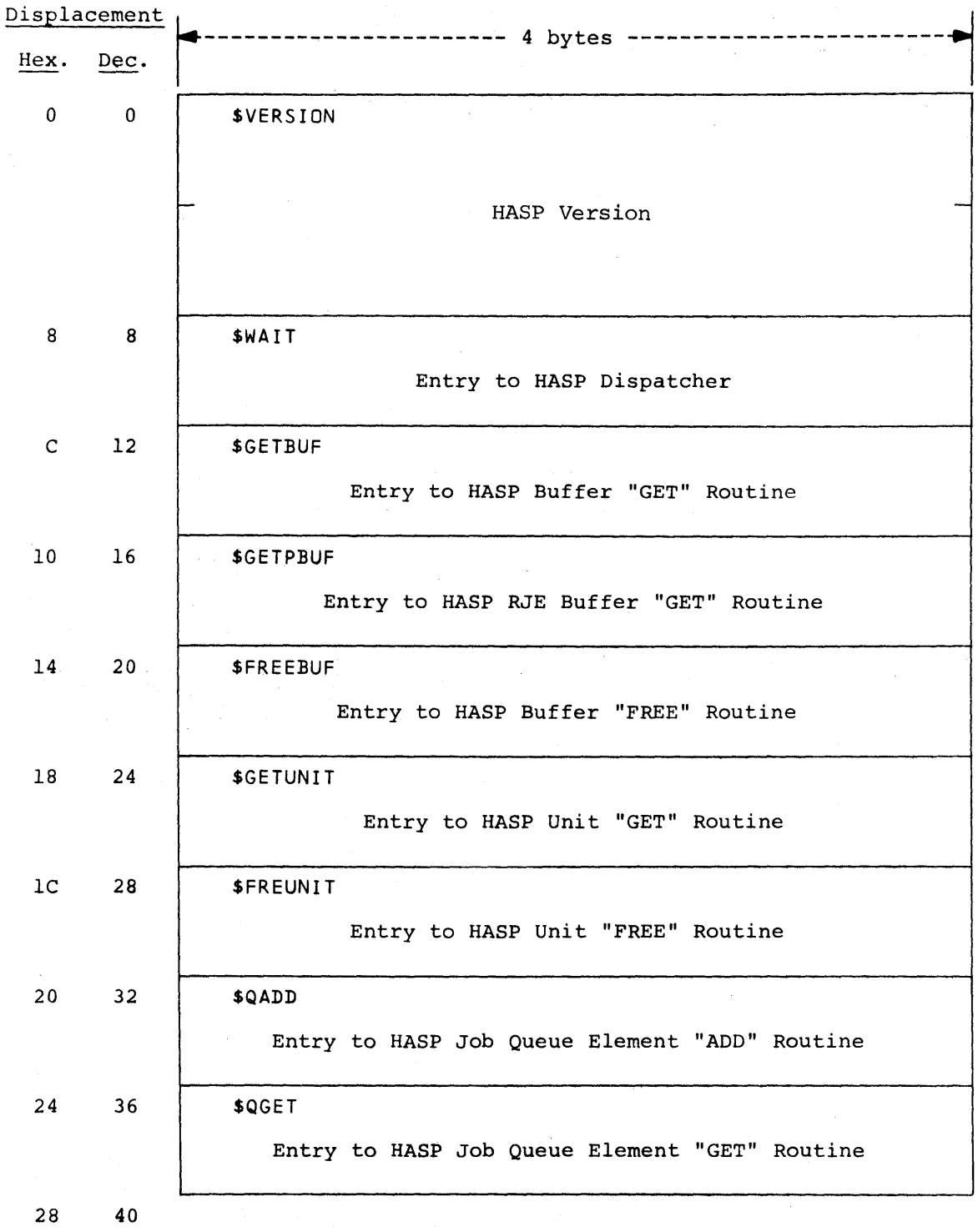
```
&CORESIZE    &RADR(1)    &RDEV(1)
&SUBMOD      &UADR(1)
```

All other Model 20 BSC parameters may be allowed to default or may be altered as desired, according to the description in Section 7.3.

8.0 HASP CONTROL TABLE FORMATS

This sections contains block diagrams which depict the formats of the HASP Control Tables which are not described in other sections of this manual.

Figure 8.1.1 -- HASP COMMUNICATION TABLE FORMAT



HASP Communication Table Format -- Page 8.1-1

Figure 8.1.1 -- HASP COMMUNICATION TABLE FORMAT (CONTINUED)

Displacement Hex.	Dec.	4 bytes
28	40	$QPUT Entry to HASP Job Queue Element "PUT" Routine
2C	44	$QREM Entry to HASP Job Queue Element "REMOVE" Routine
30	48	$QSIZ Entry to HASP Job Queue "SIZE" Routine
34	52	$QLOC Entry to HASP Job Queue Element "LOCATE" Routine
38	56	$QJITLOC Entry to HASP JIT Element "LOCATE" Routine
3C	60	$TRACK Entry to HASP Track Allocation Routine
40	64	$PURGER Entry to HASP Track Purge Routine
44	68	$EXCP Entry to HASP Input/Output Supervisor
48	72	$EXTPOPE Entry to HASP RTAM Open Routine
4C	76	$EXTPGET Entry to HASP RTAM Get Routine
50	80	

Figure 8.1.1 -- HASP COMMUNICATION TABLE FORMAT (CONTINUED)

Displacement Hex.	Dec.	4 bytes
50	80	$EXTPPUT Entry to HASP RTAM Put Routine
54	84	$EXTPCLO Entry to HASP RTAM Close Routine
58	88	$RESTORE Entry to HASP RTAM Restore Routine
5C	92	$ODEL Entry to HASP Overlay $DELETE Routine
60	96	$ORET Entry to HASP Overlay $RETURN Routine
64	100	$OLINK Entry to HASP Overlay $LINK Routine
68	104	$OXCTL Entry to HASP Overlay $XCTL Routine
6C	108	$OLOAD Entry to HASP Overlay $LOAD Routine
70	112	$WTO Entry to HASP Write-to-Operator Routine
74	116	$FREEMSG Entry to HASP Console Message Buffer Free Routine
78	120	

Figure 8.1.1 -- HASP COMMUNICATION TABLE FORMAT (CONTINUED)

Displacement Hex.	Dec.	4 bytes			
78	120	$STIMER Entry to HASP Set Interval Timer Routine			
7C	124	$TTIMER Entry to HASP Test Interval Timer Routine			
80	128	$IOERROR Entry to HASP Input/Output Error Logging Routine			
84	132	$ERROR Entry to HASP Catastrophic Error Routine			
88	136	$DISTERR Entry to HASP Disastrous Error Routine			
8C	140	$SYSTYPE System Type MFT or MVT	$OPTSTAT Initialization Options	$STATUS HASP Status	RESERVED
90	144	$HASPECF Master Event Control Field	MHASPECB RJE Event Control Field	$XEQACT O/S Execution Count	$ACTIVE Active Count
94	148	$ENBALL Enable All Mask	$DISALL Disable All Mask	$DISINT Disable Int Timer Mask	RESERVED
98	152	$PSRDRCT Pseudo Reader Count (2540)	$PSPRFCT Pseudo Printer Count (1443)	$PSPUFCT Pseudo Punch Count (1442)	RESERVED
9C	156	$EXCPCT Active I/O Count		$COMMCT Active Command Count	
A0	160				

Figure 8.1.1 -- HASP COMMUNICATION TABLE FORMAT (CONTINUED)

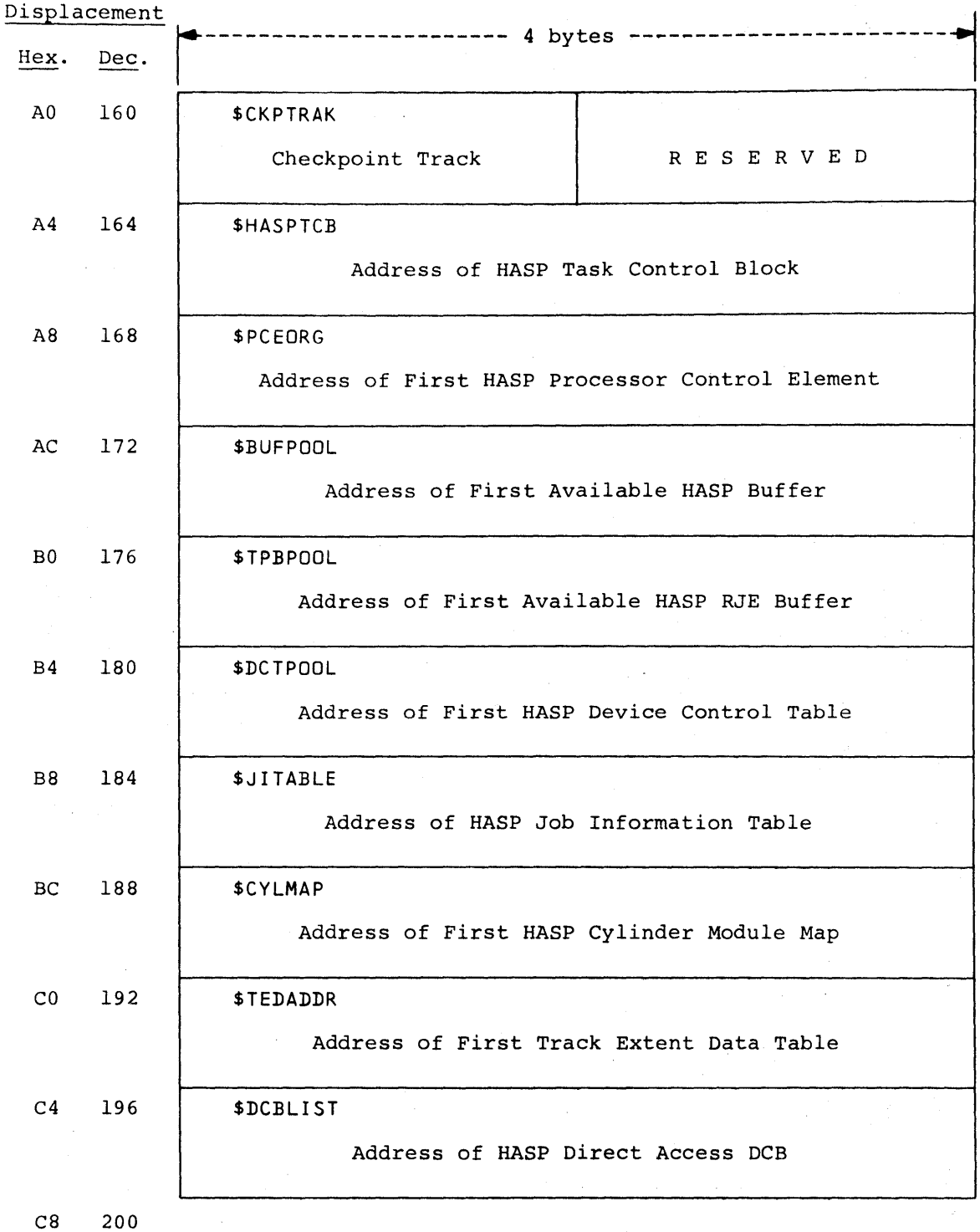


Figure 8.1.1 -- HASP COMMUNICATION TABLE FORMAT (CONTINUED)

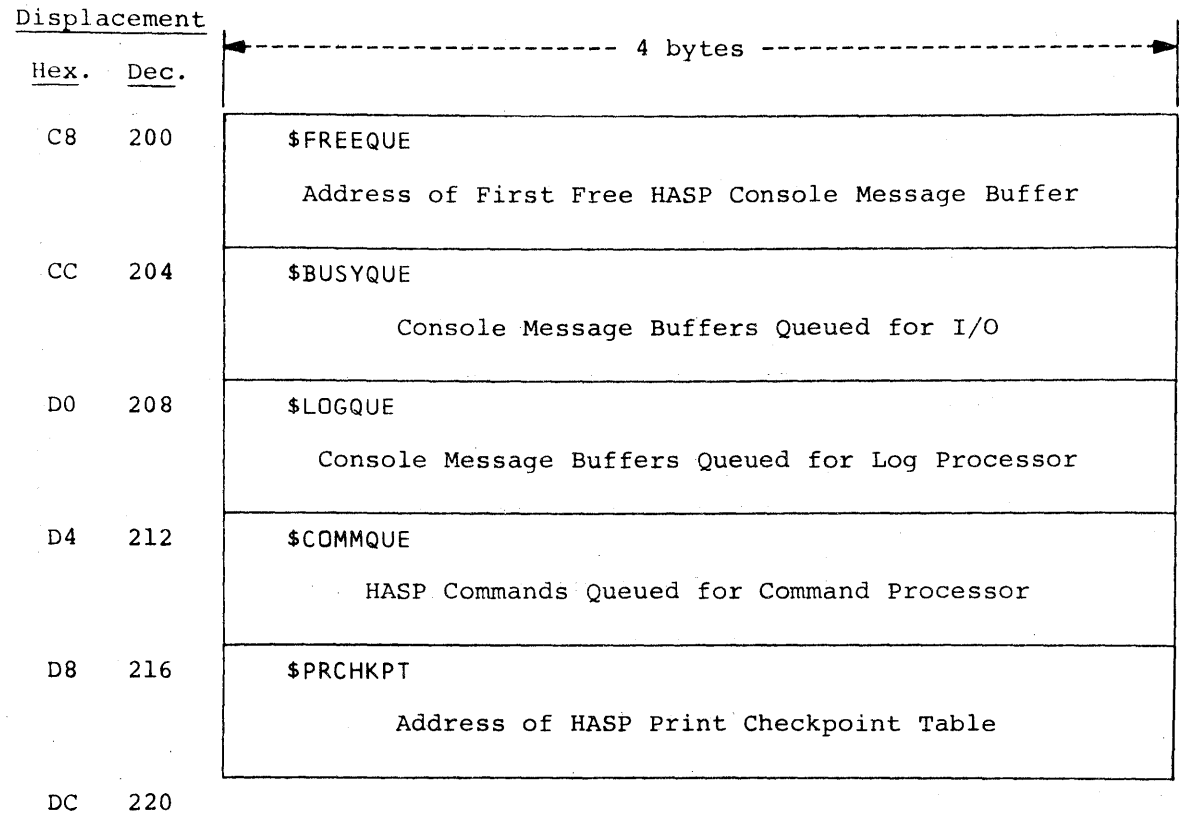


Figure 8.1.1 -- HASP COMMUNICATION TABLE FORMAT (CONTINUED)

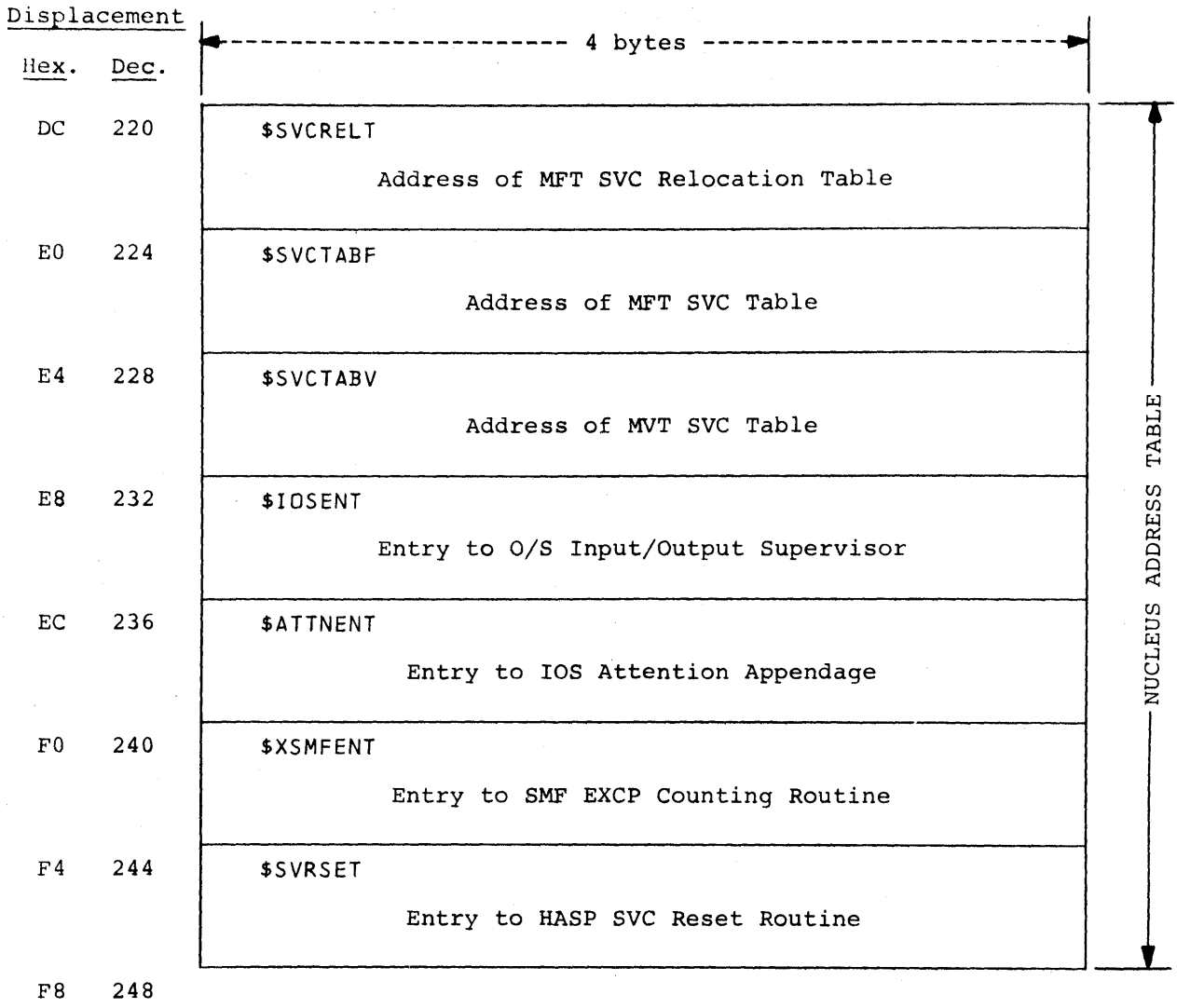


Figure 8.1.1 -- HASP COMMUNICATION TABLE FORMAT (CONTINUED)

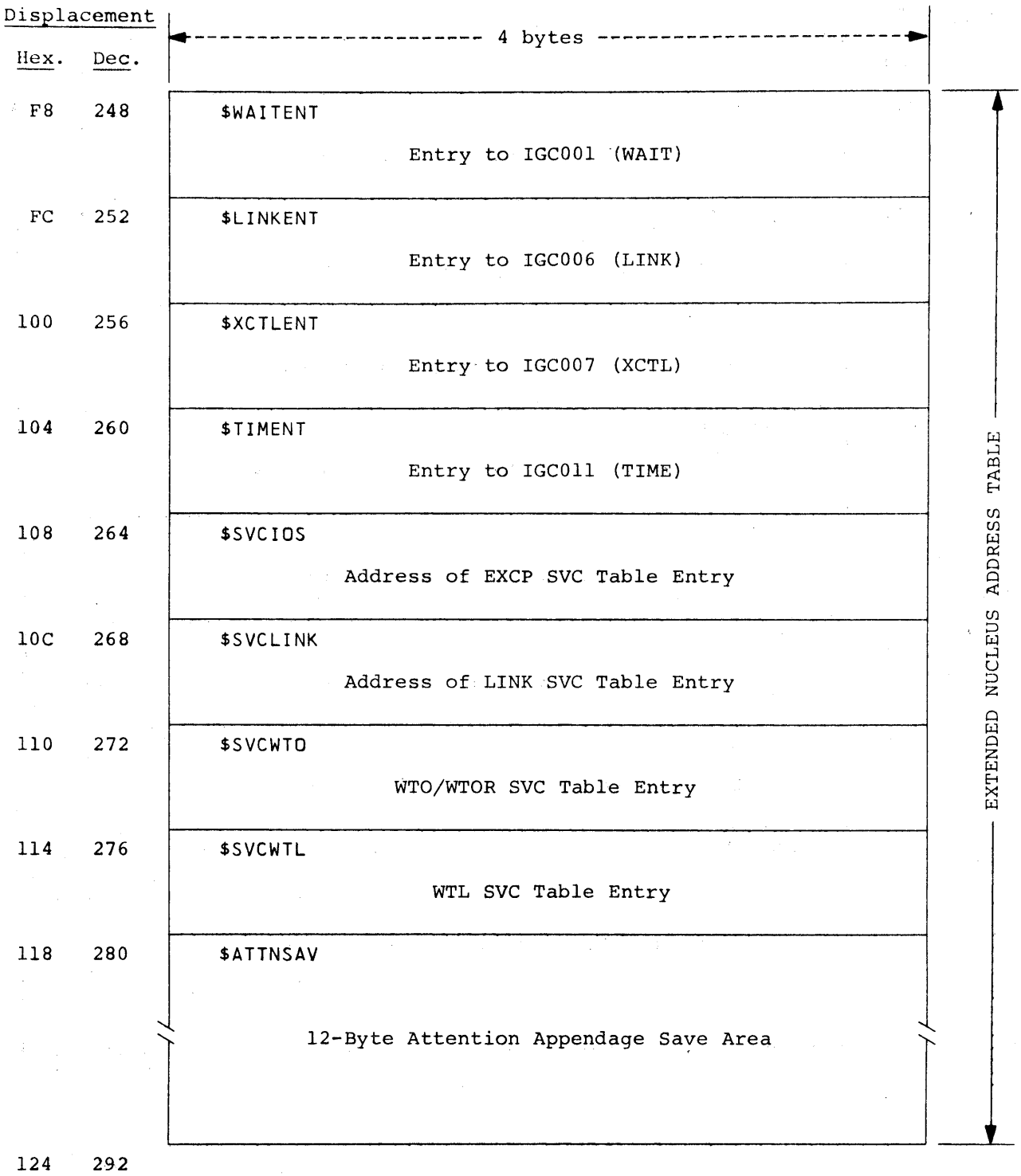


Figure 8.1.1 -- HASP COMMUNICATION TABLE FORMAT (CONTINUED)

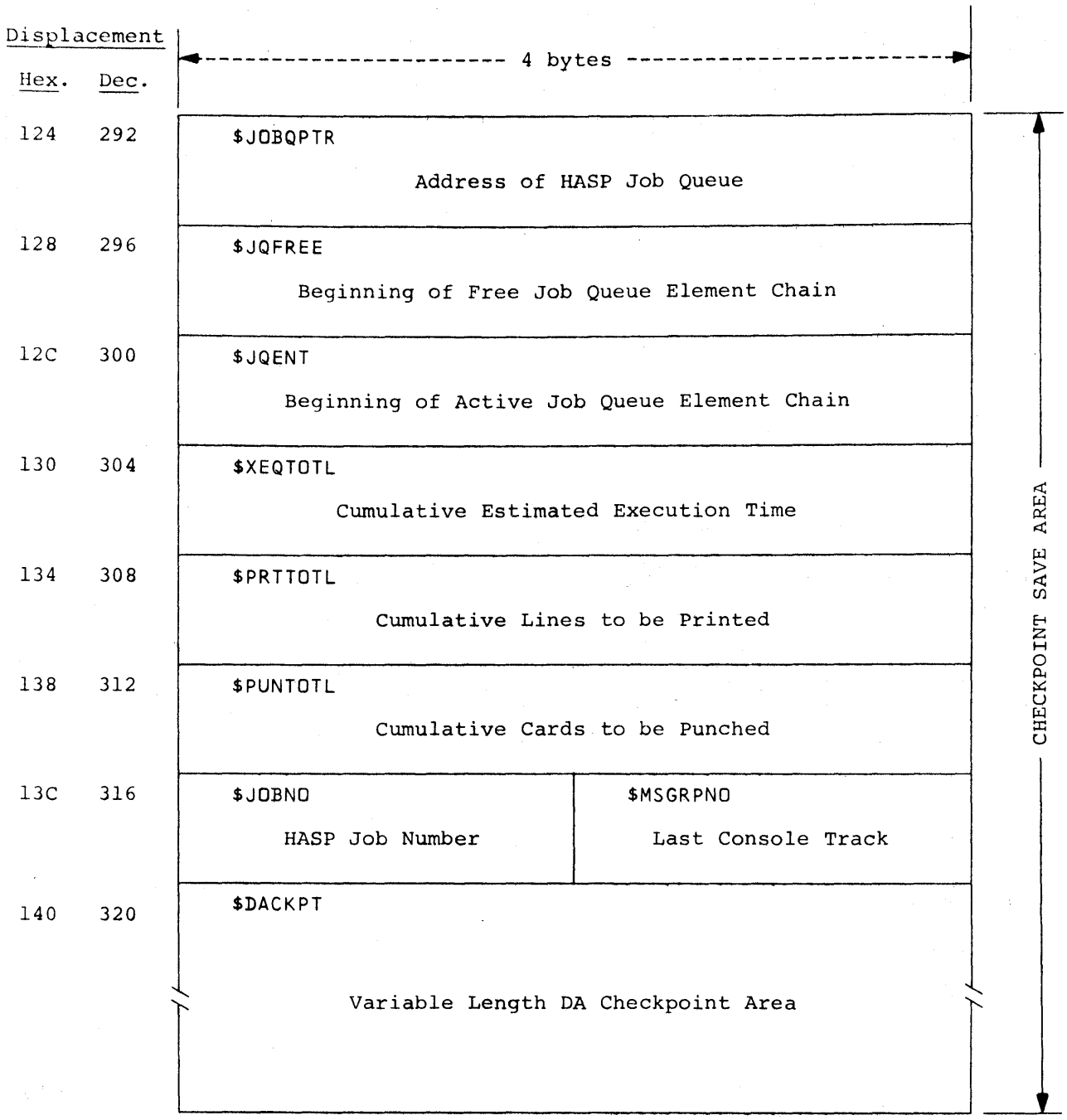


Figure 8.1.1 -- HASP COMMUNICATION TABLE FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
0	0	$VERSION	8	HASP Version (" V v.m ").
8	8	$WAIT	4	Entry to HASP Dispatcher.
C	12	$GETBUF	4	Entry to HASP Buffer "GET" Routine.
10	16	$GETPBUF	4	Entry to HASP RJE Buffer "GET" Routine.
14	20	$FREEBUF	4	Entry to HASP Buffer "FREE" Routine.
18	24	$GETUNIT	4	Entry to HASP Unit "GET" Routine.
1C	28	$FREUNIT	4	Entry to HASP Unit "FREE" Routine.
20	32	$QADD	4	Entry to HASP Job Queue Element "ADD" Routine.
24	36	$QGET	4	Entry to HASP Job Queue Element "GET" Routine.
28	40	$QPUT	4	Entry to HASP Job Queue Element "PUT" Routine.
2C	44	$QREM	4	Entry to HASP Job Queue Element "REMOVE" Routine.
30	48	$QSIZ	4	Entry to HASP Job Queue "SIZE" Routine.
34	52	$QLOC	4	Entry to HASP Job Queue Element "LOCATE" Routine.
38	56	$QJITLOC	4	Entry to HASP Job Information Table Element "LOCATE" Routine.
3C	60	$TRACK	4	Entry to HASP Track Allocation Routine.
40	64	$PURGER	4	Entry to HASP Track Purge Routine.
44	68	$EXCP	4	Entry to HASP Input/Output Supervisor.
48	72	$EXTPOPE	4	Entry to HASP RTAM Open Routine.
4C	76	$EXTPGET	4	Entry to HASP RTAM Get Routine.
50	80	$EXTPPUT	4	Entry to HASP RTAM Put Routine.
54	84	$EXTPOPE	4	Entry to HASP RTAM Close Routine.
58	88	$RESTORE	4	Entry to HASP RTAM Restore Routine.

Figure 8.1.1 -- HASP COMMUNICATION TABLE FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
5C	92	$ODEL	4	Entry to HASP Overlay $DELETE Routine.
60	96	$ORET	4	Entry to HASP Overlay $RETURN Routine.
64	100	$OLINK	4	Entry to HASP Overlay $LINK Routine.
68	104	$OXCTL	4	Entry to HASP Overlay $XCTL Routine.
6C	108	$OLOAD	4	Entry to HASP Overlay $LOAD Routine.
70	112	$WTO	4	Entry to HASP Write-to-Operator Routine.
74	116	$FREEMSG	4	Entry to HASP Console Message Buffer Free Routine.
78	120	$STIMER	4	Entry to HASP Set Interval Timer Routine.
7C	124	$TTIMER	4	Entry to HASP Test Interval Timer Routine.
80	128	$IOERROR	4	Entry to HASP Input/Output Error Logging Routine.
84	132	$ERROR	4	Entry to HASP Catastrophic Error Routine.
88	136	$DISTERR	4	Entry to HASP Disastrous Error Routine.
8C	140	$SYSTYPE	1	System Type --

Hex. Value	Meaning
10	MVT
14	MPS
20	MFT

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
8D	141	$OPTSTAT	1	Initialization Options --

Bit	Name	Meaning
0	$OPTFMT	FORMAT.
1	$OPTCOLD	COLD.
2	$OPTREQ	REQ.
3	$OPTREP	REP.
4	$OPTLIST	LIST.
5	$OPTRACE	TRACE.
6-7		Reserved for Future Use.

Figure 8.1.1 -- HASP COMMUNICATION TABLE FORMAT (CONTINUED)

Displacement Hex.	Dec.	Field Name	Bytes	Field Description
8E	142	$STATUS	1	HASP Status --

Bit	Name	Meaning
0	$RDRPEND	O/S Reader is Pending.
1	$ALMSGSW	ALL AVAILABLE FUNCTIONS COMPLETE Message has been Issued.
2	$DRAINED	System has been $DRAINed.
3	$CKPTACT	Checkpoint is in Progress.
4	$JITCKPT	Job Information Table (JIT) is to be Checkpointed.
5	$SYSEXIT	HASP System is in termination process.
6-7		Reserved for Future Use.

8F	143		1	Reserved for Future Use.
90	144	$HASPECF	1	Master Event Control Field --

Bit	Name	Meaning
0	$EWFPOST	A PCE has been $POSTed.
1	$EWFBUF	A Buffer has been Released.
2	$EWFTRAK	A Direct-Access Track has been Released.
3	$EWFJOB	A Job Queue Element has Changed Status.
4	$EWFUNIT	A HASP Unit has been Released.
5	$EWFCKPT	A HASP Checkpoint has Completed.
6	$EWFCMB	A Console Message Buffer has been Released.
7	$EWF8	Reserved for Future Use.

91	145	MHASPECF	1	Remote Job Entry Line Manager Event Control Field --

Bit	Name	Meaning
0-2		Reserved - Must be Zero.
3	$EWFJOB	A Job Queue Element has Changed Status.
4-7		Reserved - Must be Zero.

Figure 8.1.1 -- HASP COMMUNICATION TABLE FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
92	146	$XEQACT	1	Count of Jobs in O/S Execution Phase.
93	147	$ACTIVE	1	Count of Active Processors.
94	148	$ENBALL	1	$ENABLE ALL Mask (X'FF').
95	149	$DISALL	1	$DISABLE ALL Mask (X'00').
96	150	$DISINT	1	$DISABLE INT Mask (X'FE').
97	151		1	Reserved for Future Use.
98	152	$PSRDRCT	1	Count of Pseudo 2540 Readers.
99	153	$PSPRFCT	1	Count of Pseudo 1443 Printers.
9A	154	$PSPUFCT	1	Count of Pseudo 1442 Punches.
9B	155		1	Reserved for Future Use.
9C	156	$EXCPCT	2	Count of Active I/O Operations.
9E	158	$COMMCT	2	Number of Console Message Buffers which are not Queued for the HASP Command Processor.
A0	160	$CKPTRAK	2	Checkpoint Track.
A2	162		2	Reserved for Future Use.
A4	164	$HASPTCB	4	Address of HASP Task Control Block.
A8	168	$PCEORG	4	Address of First HASP Processor Control Element.
AC	172	$BUFPOOL	4	Address of First Available HASP Buffer.
B0	176	$TPBPOOL	4	Address of First Available HASP RJE Buffer.
B4	180	$DCTPOOL	4	Address of First HASP Device Control Table.
B8	184	$JITABLE	4	Address of HASP Job Information Table.
BC	188	$CYLMAP	4	Address of First HASP Track Allocation Map.
C0	192	$TEDADDR	4	Address of First Track Extent Data Table.

Figure 8.1.1 -- HASP COMMUNICATION TABLE FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
C4	196	$DCBLIST	4	Address of HASP Direct Access DCB.
C8	200	$FREEQUE	4	Address of First Free HASP Console Message Buffer.
CC	204	$BUSYQUE	4	Address of First Console Message Buffer which is Queued for I/O.
D0	208	$LOGQUE	4	Address of First Console Message Buffer which is Queued for the Log Processor.
D4	212	$COMMQUE	4	Address of First Console Message Buffer which is Queued for the Command Processor.
D8	216	$PRCHKPT	4	Address of HASP Print Checkpoint Table.
DC	220	$SVCRELT	4	Address of MFT SVC Relocation Table.
E0	224	$SVCTABF	4	Address of MFT SVC Table.
E4	228	$SVCTABV	4	Address of MVT SVC Table.
E8	232	$IOSENT	4	Address of Entry to O/S Input/Output Supervisor.
EC	236	$ATTNENT	4	Address of Entry to IOS Attention Appendage.
F0	240	$XSMFEMT	4	Address of Entry to SMF EXCP Counting Routine.
F4	244	$SVRSET	4	Address of Entry to HASP SVC Reset Routine.
F8	248	$WAITENT	4	Address of Entry to IGC001 (WAIT).
FC	252	$LINKENT	4	Address of Entry to IGC006 (LINK).
100	256	$XCTLENT	4	Address of Entry to IGC007 (XCTL).
104	260	$TIMENT	4	Address of Entry to IGC011 (TIME).
108	264	$SVCIOS	4	Address of EXCP SVC Table Entry.
10C	268	$SVCLINK	4	Address of LINK SVC Table Entry.

Figure 8.1.1 -- HASP COMMUNICATION TABLE FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
110	272	$SVCWTO	4	WTO/WTOR SVC Table Entry.
114	276	$SVCWTL	4	WTL SVC Table Entry.
118	280	$ATTNSAV	12	Attention Appendage Save Area.
124	292	$JOBQPTR	4	Address of HASP Job Queue.
128	296	$JQFREE	4	Beginning of Free Job Queue Element Chain.
12C	300	$JQENT	4	Beginning of Active Job Queue Element Chain.
130	304	$XEQTOTL	4	Cumulative Estimated Execution Time.
134	308	$PRTTOTL	4	Cumulative Lines to be Printed.
138	312	$PUNTOTL	4	Cumulative Cards to be Punched.
13C	316	$JOBNO	2	HASP Job Number.
13E	318	$MSGRPNO	2	Last Remote Console Message Queueing Track.
140	320	$DACKPT		Variable Length Direct Access Checkpoint Area.

Figure 8.2.1 -- PROCESSOR CONTROL ELEMENT FORMAT

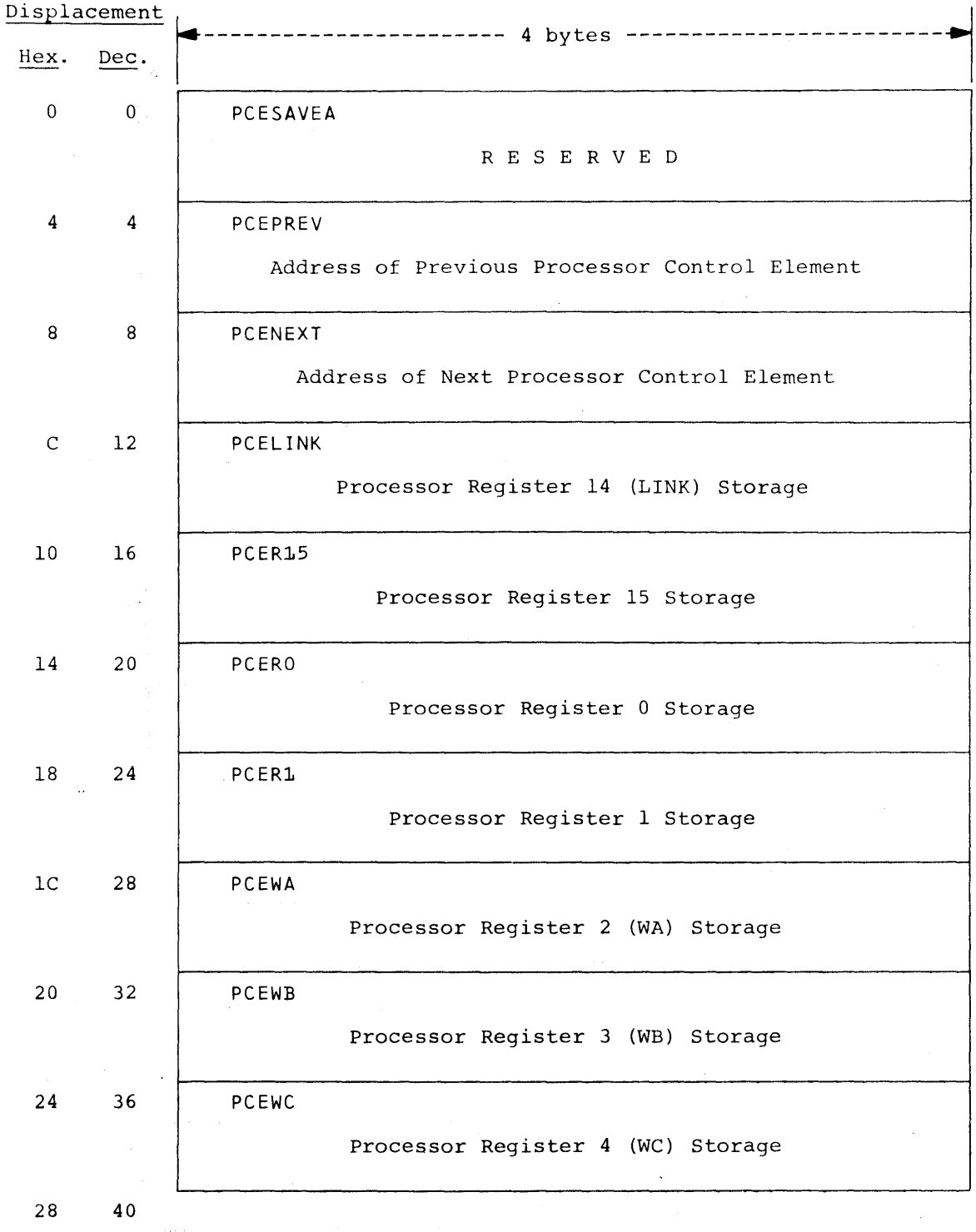


Figure 8.2.1 -- PROCESSOR CONTROL ELEMENT FORMAT (CONTINUED)

Displacement Hex.	Dec.	4 bytes
28	40	PCEWD Processor Register 5 (WD) Storage
2C	44	PCEWE Processor Register 6 (WE) Storage
30	48	PCEWF Processor Register 7 (WF) Storage
34	52	PCEWG PCEBASE3 Processor Register 8 (WG or BASE3) Storage
38	56	PCER9 Processor Register 9 Storage
3C	60	PCEJCT Processor Register 10 (JCT) Storage
40	64	PCEBASE1 Processor Register 11 (BASE1) Storage
44	68	PCEBASE2 Processor Register 12 (BASE2) Storage
48	72	PCEEWF Event Wait Field / PCEID Processor Type
4C	76	RESERVED / PCEOPRIO Overlay Priority / PCEOCON Overlay Routine OCON
50	80	

Processor Control Element Format -- Page 8.2-2

Figure 8.2.1 -- PROCESSOR CONTROL ELEMENT FORMAT (CONTINUED)

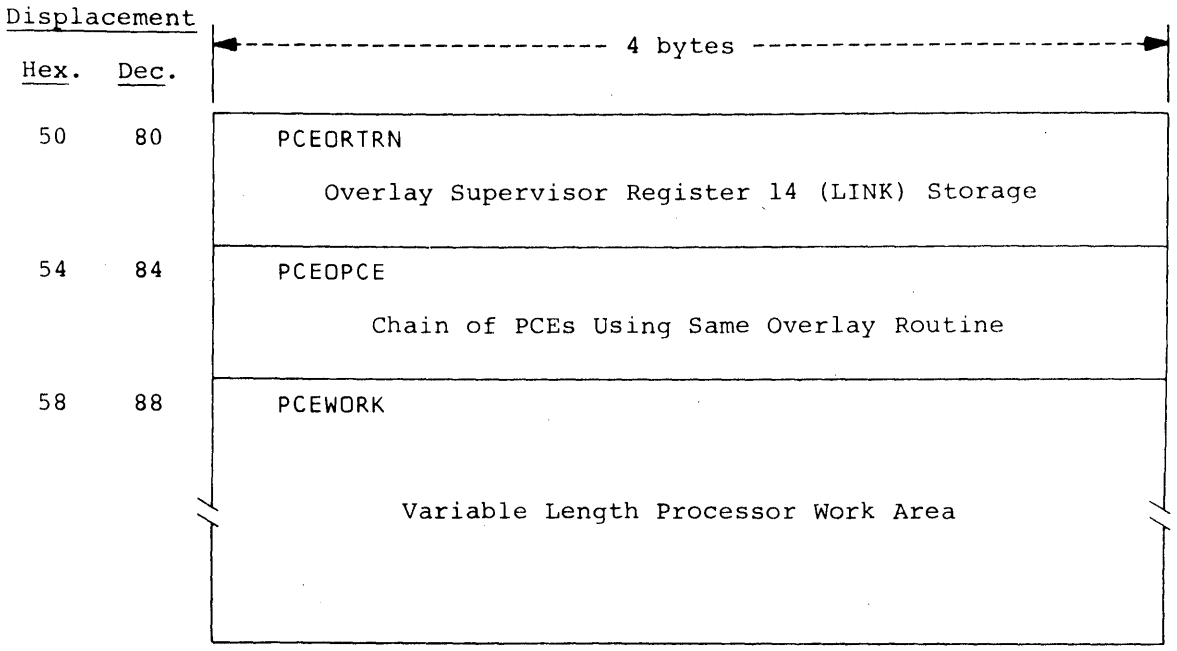


Figure 8.2.1 -- PROCESSOR CONTROL ELEMENT FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
0	0	PCESAVEA	4	Reserved.
4	4	PCEPREV	4	Address of Previous Processor Control Element.
8	8	PCENEXT	4	Address of Next Processor Control Element.
C	12	PCELINK	4	Processor Register 14 (LINK) Storage.
10	16	PCER15	4	Processor Register 15 Storage.
14	20	PCER0	4	Processor Register 0 Storage.
18	24	PCER1	4	Processor Register 1 Storage.
1C	28	PCEWA	4	Processor Register 2 (WA) Storage.
20	32	PCEWB	4	Processor Register 3 (WB) Storage.
24	36	PCEWC	4	Processor Register 4 (WC) Storage.
28	40	PCEWD	4	Processor Register 5 (WD) Storage.
2C	44	PCEWE	4	Processor Register 6 (WE) Storage.
30	48	PCEWF	4	Processor Register 7 (WF) Storage.
34	52	PCEWG PCEBASE3	4	Processor Register 8 (WG or BASE3) Storage.
38	56	PCER9	4	Processor Register 9 Storage.
3C	60	PCEJCT	4	Processor Register 10 (JCT) Storage.
40	64	PCEBASE1	4	Processor Register 11 (BASE1) Storage.
44	68	PCEBASE2	4	Processor Register 12 (BASE2) Storage.

Figure 8.2.1 -- PROCESSOR CONTROL ELEMENT FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
48	72	PCEEWF	2	Event Wait Field --

	Hex. Value	Name	Meaning
Byte 1	80	$EWFPOST	Reserved.
	40	$EWFBUF	Waiting for a Buffer.
	20	$EWFTRAK	Waiting for HASP Direct-Access Space.
	10	$EWFJOB	Waiting for a Job.
	08	$EWFUNIT	Waiting for a Unit.
	04	$EWFCKPT	Waiting for the completion of a HASP Checkpoint.
	02	$EWFCMB	Waiting for a Console Message Buffer.
	01	$EWF8	Reserved for Future Use.
Byte 2	80	$EWFOPER	Waiting for Operator Response.
	40	$EWFIO	Waiting for the Completion of I/O.
	20	$EWFWORK	Waiting to be Re-directed.
	10	$EWFHOLD	Waiting for a $S Command.
	08	$EWFDDB	Waiting for a DDT or UCB.
	04	$EWFOLAY	Waiting for an Overlay Area.
	02	$EWF15	Reserved for Future Use.
	01	$EWFOROL	Relinquished Overlay Area.

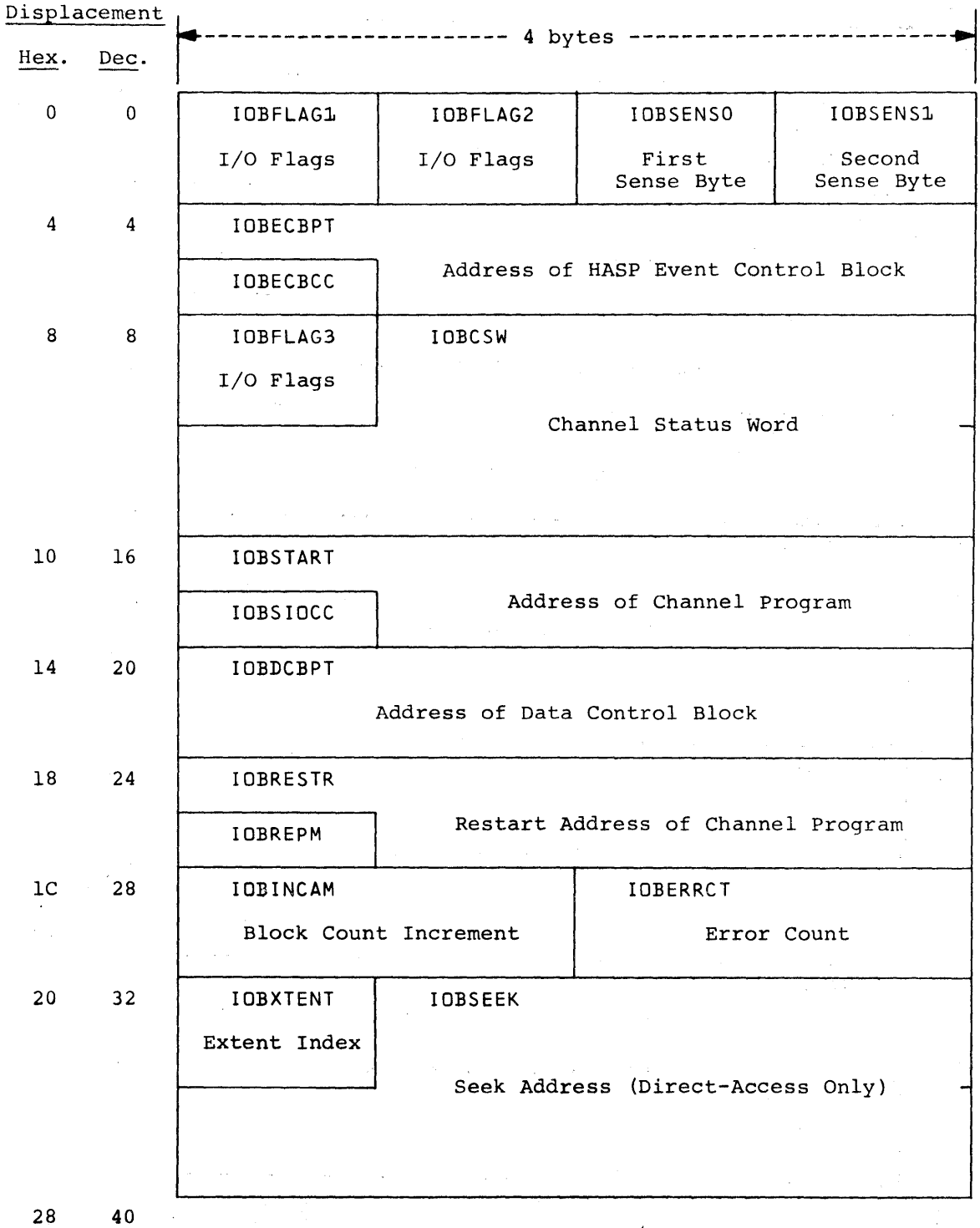
4A	74	PCEID	2	Processor Type --

	Bit	Name	Meaning
Byte 1	0	PCEPRSID	Print Processor.
	1	PCEPUSID	Punch Processor.
	2-4		Reserved for Future Use.
	5	PCEINRID	Internal Reader Processor.
	6	PCERJEID	Remote Terminal Processor.
	7	PCELCLID	Local Processor.

Figure 8.2.1 -- PROCESSOR CONTROL ELEMENT FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
				Processor Type (continued) --
				Hex. Value / Name / Meaning
			Byte 2	00 PCEASYID ASYNCH Processor.
				01 PCERDRID Input Service Processor.
				03 PCEXEQID Execution Service Processor.
				04 PCETHWID Execution Thaw Processor.
				05 PCEXZMID Execution Task Monitor.
				07 PCEPRTID Print Processor.
				08 PCEPUNID Punch Processor.
				09 PCEPRGID Purge Processor.
				0A PCECONID Console Processor.
				0B PCEMLMID Line Manager Processor.
				0C PCETIMID Timer Processor.
				0D PCECKPID Checkpoint Processor.
				0E PCEGPRID Priority Aging Processor.
				0F PCEOROID Overlay Roll Processor.
4C	76		1	Reserved for Future Use.
4D	77	PCEOPRIO	1	Priority of Current Overlay Routine.
4E	78	PCEOCON	2	Overlay Constant (OCON) of Current Overlay Routine.
50	80	PCEORTRN	4	Overlay Supervisor Register 14 (LINK) Storage.
54	84	PCEOPCE	4	Chain of PCEs Using Same Overlay Routine.
58	88	PCEWORK		Variable Length Processor Work Area.

Figure 8.3.1 -- BUFFER FORMAT



Buffer Format -- Page 8.3-1

Figure 8.3.1 -- BUFFER FORMAT (CONTINUED)

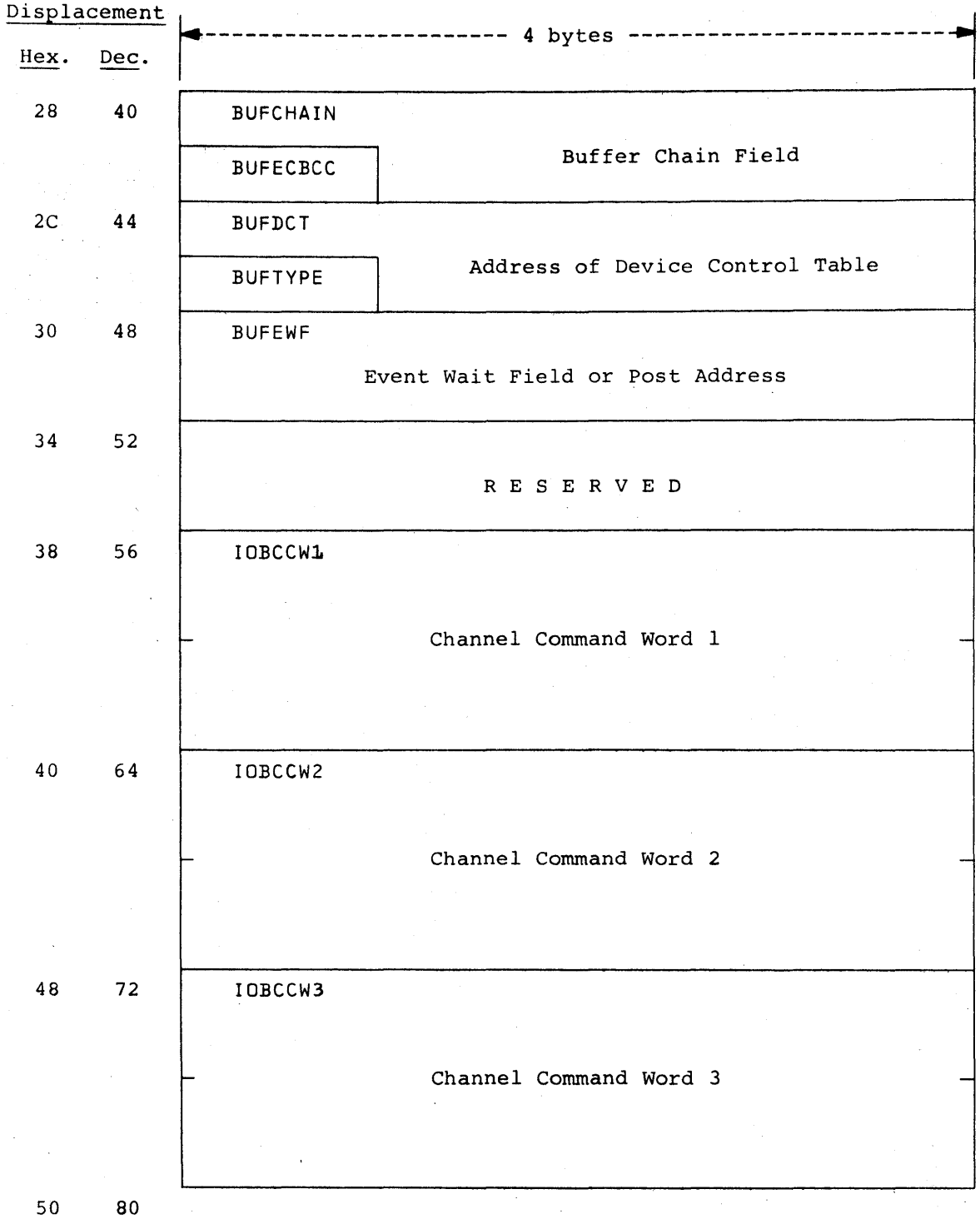


Figure 8.3.1 -- BUFFER FORMAT (CONTINUED)

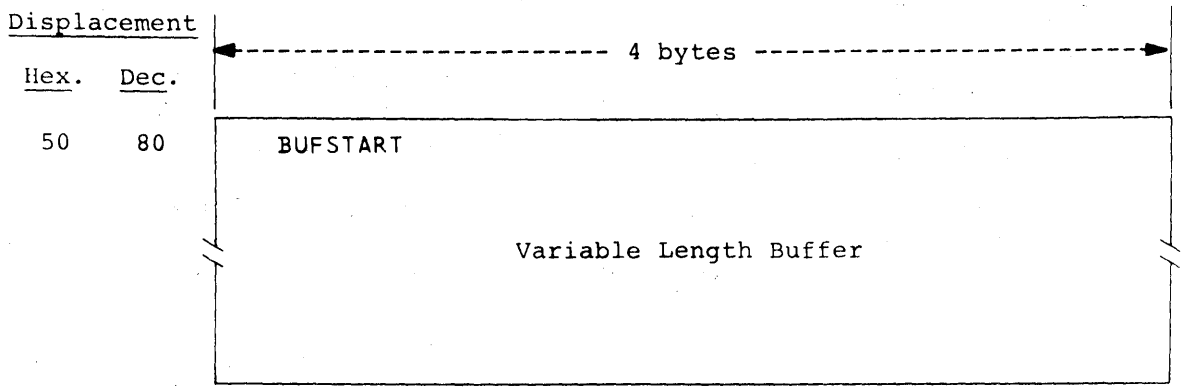


Figure 8.3.1 -- BUFFER FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
0	0	IOBFLAG1	1	Standard OS/360 IOB Flag Byte.
1	1	IOBFLAG2	1	Standard OS/360 IOB Flag Byte.
2	2	IOBSENS0	1	First Sense Byte (Device Dependent).
3	3	IOBSENS1	1	Second Sense Byte (Device Dependent).
4	4	IOBECBCC	1	Completion Code for I/O Event.
4	4	IOBECBPT	4	Address of HASP Event Control Block: $HASPECB.
8	8	IOBFLAG3	1	I/O Supervisor Error Routine Flag Byte (Device Dependent).
9	9	IOBCSW	7	Low-Order Seven Bytes of the Last CSW that Reflects the Status of the Last Request.
10	16	IOBSIOCC	1	Condition Code Returned after Execution of SIO Instruction for Last Request.
10	16	IOBSTART	4	Address of Channel Program to be Executed.
14	20	IOBDCBPT	4	Address of Data Control Block Associated with this IOB.
18	24	IOBREPM	1	Operation Code Used by I/O Supervisor Error Routines for Repositioning Procedures.
18	24	IOBRESTR	4	Restart Address of Channel Program Used by I/O Supervisor Error Routines During Error Correction.
1C	28	IOBINCAM	2	Value used to Increment Block Count Field in DCB for Magnetic Tape.
1E	30	IOBERRCT	2	Used by I/O Supervisor Error Routines to Count Temporary Errors during Retry.
20	32	IOBXTENT	1	The Number of the DEB Extent to be Used for this Request.

Figure 8.3.1 -- BUFFER FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
21	33	IOBSEEK	7	Seek Address Required for this I/O Request (Direct-Access Only).
28	40	BUFECBCC	1	Completion Code for I/O Event --

Hex. Value	Meaning
00	The I/O Event has not Completed.
7F	The I/O Event has Completed Successfully.
other	The I/O Event has Completed Unsuccessfully.

28	40	BUFCHAIN	4	Buffer Chain Field.
2C	44	BUFTYPE	1	Buffer Type --

Hex. Value	Name	Meaning
00	HASPBUF	HASP Buffer

2C	44	BUFDCT	4	Address of Device Control Table Associated with this I/O Request.
30	48	BUFEWF	4	Event Wait Field or Post Address.
34	52		4	Reserved for Future Use.
38	56	IOBCCW1	8	Channel Command Word 1.
40	64	IOBCCW2	8	Channel Command Word 2.
48	72	IOBCCW3	8	Channel Command Word 3.
50	80	BUFSTART		Variable Length Buffer.

Figure 8.3.2 -- REMOTE JOB ENTRY BUFFER FORMAT

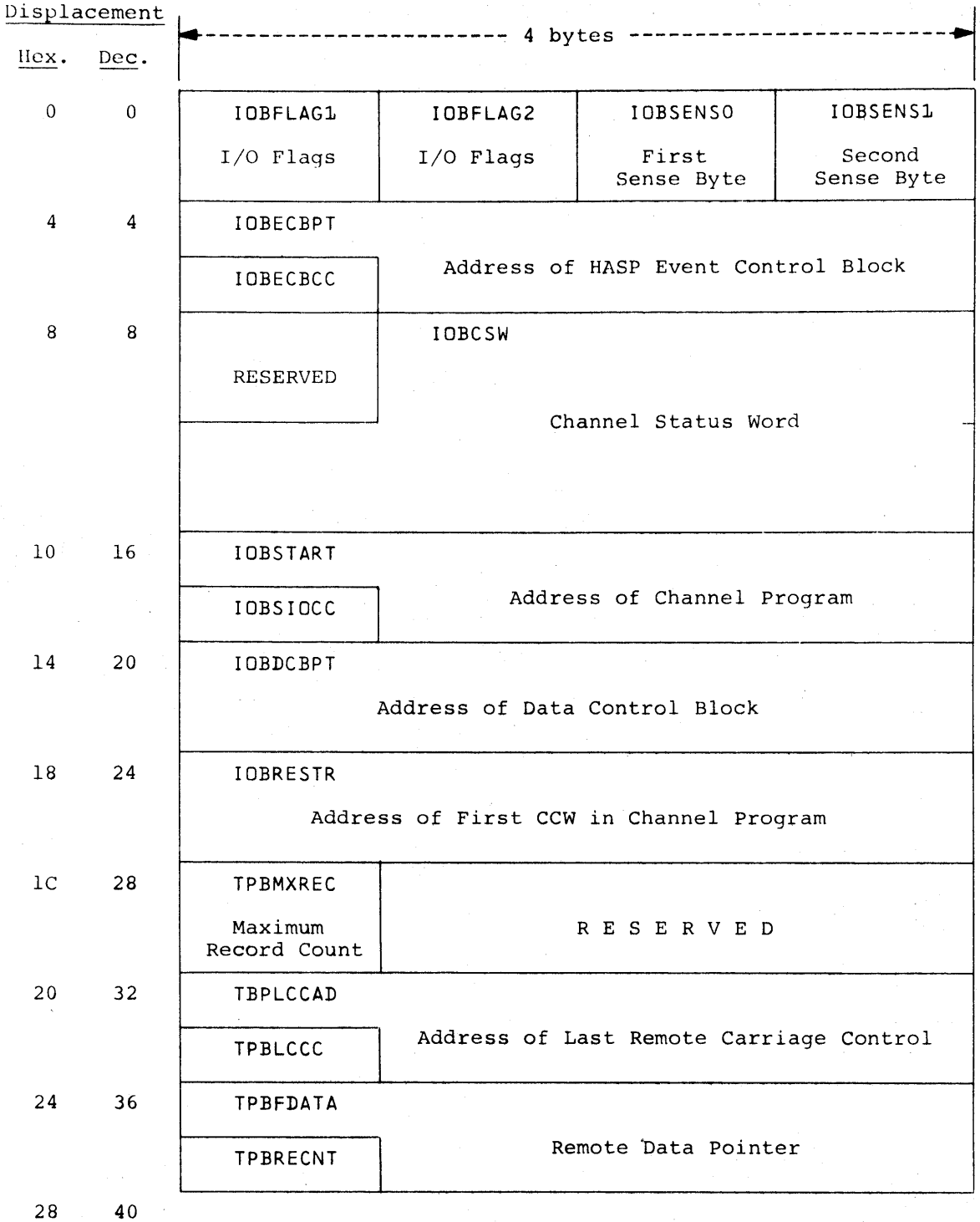


Figure 8.3.2 -- REMOTE JOB ENTRY BUFFER FORMAT (CONTINUED)

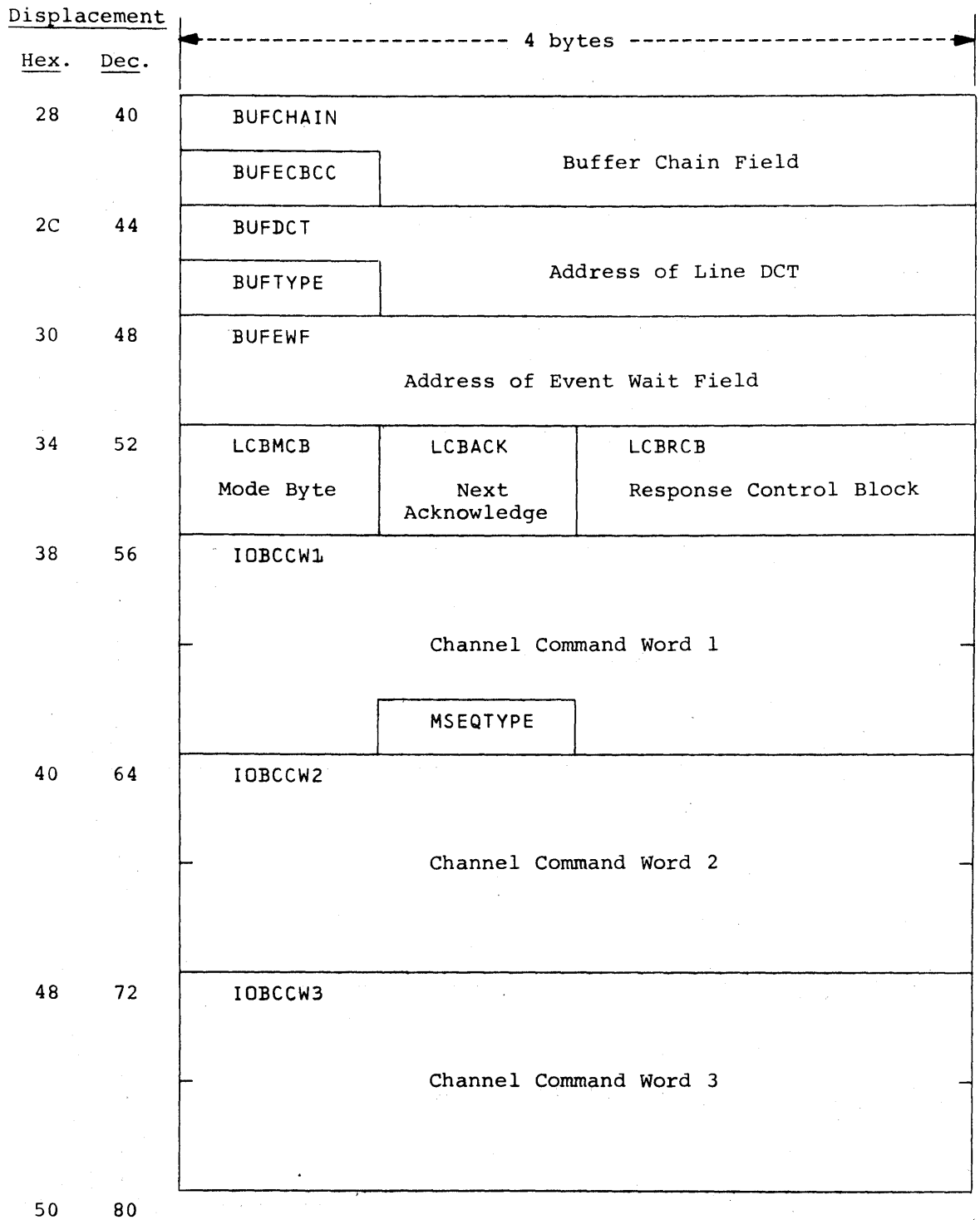


Figure 8.3.2 -- REMOTE JOB ENTRY BUFFER FORMAT (CONTINUED)

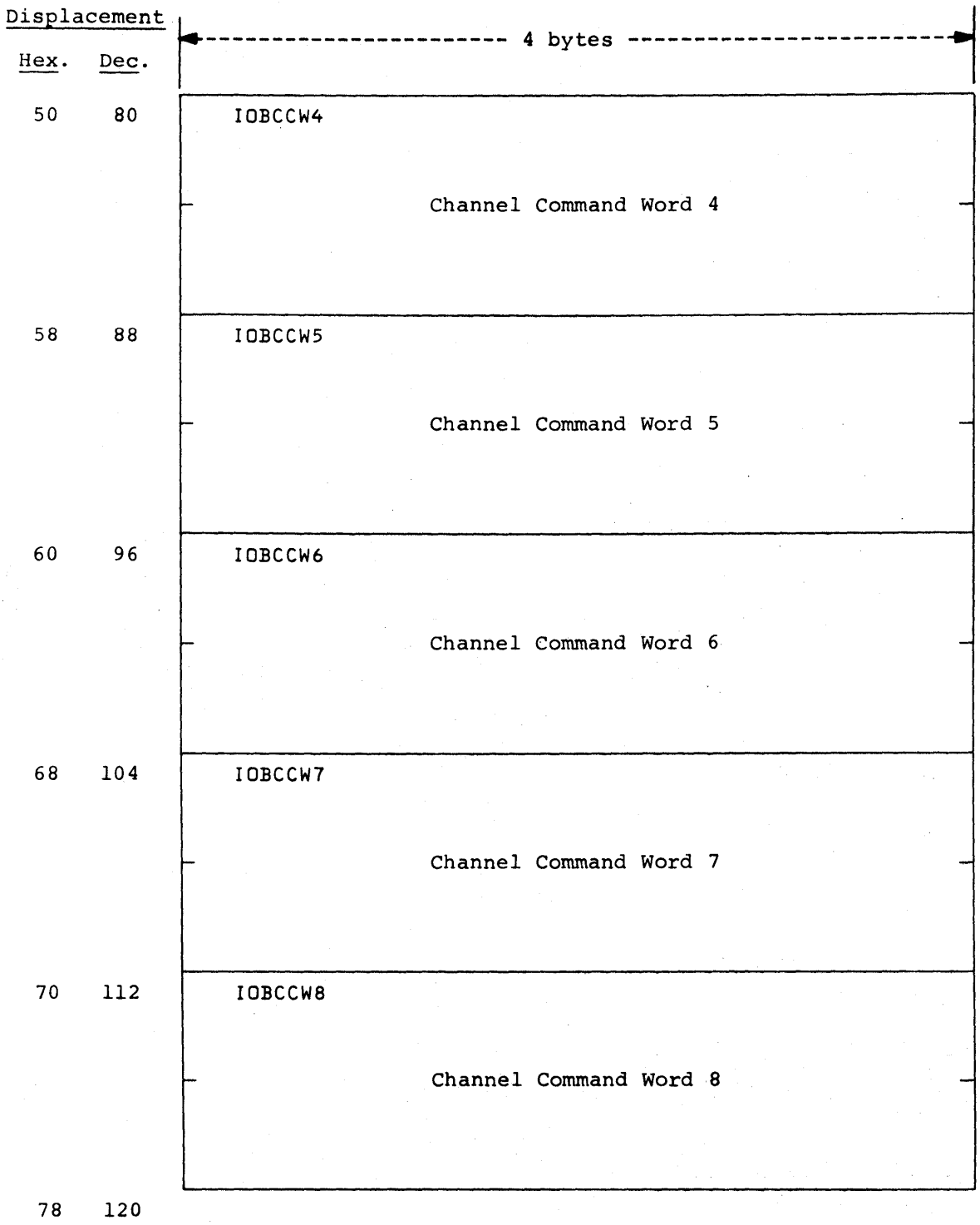


Figure 8.3.2 -- REMOTE JOB ENTRY BUFFER FORMAT (CONTINUED)

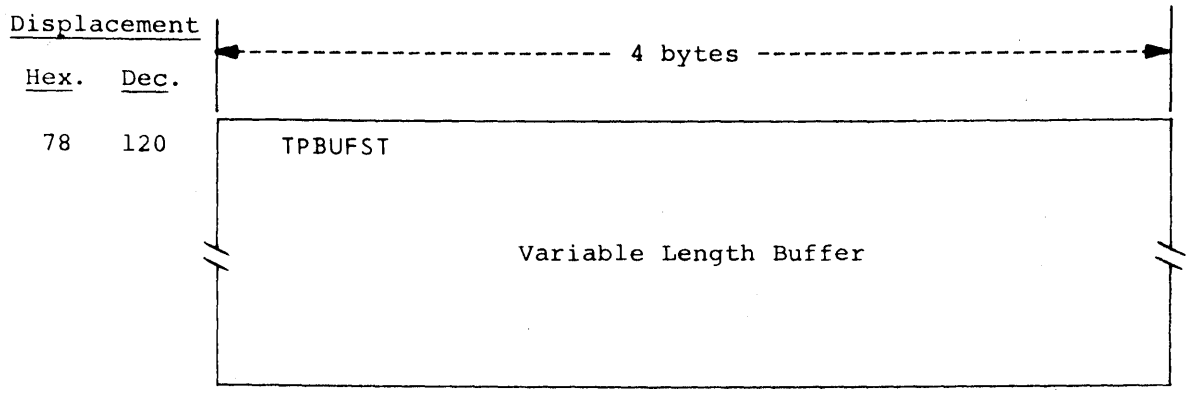


Figure 8.3.2 -- REMOTE JOB ENTRY BUFFER FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
0	0	IOBFLAG1	1	Standard OS/360 IOB Flag Byte.
1	1	IOBFLAG2	1	Standard OS/360 IOB Flag Byte.
2	2	IOBSENS0	1	First Sense Byte (Device Dependent).
3	3	IOBSENS1	1	Second Sense Byte (Device Dependent).
4	4	IOBECBCC	1	Completion Code for I/O Event.
4	4	IOBECBPT	4	Address of HASP Event Control Block: $HASPECB.
8	8		1	Reserved.
9	9	IOBCSW	7	Low Order Seven Bytes of the Last CSW that Reflects the Status of the Last Request.
10	16	IOBSIOCC	1	Condition Code Returned after Execution of SIO Instruction for Last Request
10	16	IOBSTART	4	Address of Channel Program to be Executed.
14	20	IOBDCBPT	4	Address of Data Control Block Associated with this IOB.
18	24	IOBRESTR	4	Address of Normal Channel Program to be Executed.
1C	28	TPBMXREC	1	Maximum Output Record Count.
1D	29		3	Reserved for Future Use.
20	32	TPBLCCC	1	Last Output Channel Command Operation.
20	32	TPBLCCAD	4	Address of Last Remote Carriage Control.
24	36	TPBRECNT	1	Current Output Record Count.
24	36	TPBFDATA	4	Address of Next Data in Buffer.
28	40	BUFECBCC	1	Completion Code for I/O Event.
28	40	BUFCHAIN	4	Buffer Chain Field.

Figure 8.3.2 -- REMOTE JOB ENTRY BUFFER FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
2C	44	BUFTYPE	1	Buffer Type --

Hex. Value	Name	Meaning
80	TPBUF	Remote Job Entry Buffer

2C	44	BUFDCT	4	Address of Line Device Control Table Associated with this I/O Request.
30	48	BUFEWF	4	Address of Event Wait Field.
34	52	LCBMCB	1	Mode Byte Used to Set SDA Mode.
35	53	LCBACK	1	BSC: Next Acknowledgement Character (Expected or to be Sent). STR: Second Mode Byte.
36	54	LCBRCB	2	BSC: Response Control Block. STR: Unused.
38	56	IOBCCW1	8	Channel Command Word 1.
3D	61	MSEQTYPE	1	Sequence and Command Type --

Bits 0-3 Sequence Type

Bit	Name	Value	Meaning
0	MBSCSEQ	0	STR Sequence.
		1	BSC Sequence.
1	MPREPSEQ	0	Text Sequence.
		1	Prepare Sequence.
2	MWRITSEQ	0	Read Sequence.
		1	Write Sequence.
3	MCPUSEQ	0	Hardware Sequence.
		1	CPU Sequence.

Figure 8.3.2 -- REMOTE JOB ENTRY BUFFER FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
				Sequence and Command Type (continued) --
			Bits 4-7	Command Type

Hex. Value	Name	Meaning
0	MDISCMD	Disable Command.
1	MSETMCMD	Set Mode Command.
2	MENBCMD	Enable Command.
3	MTSYNCMD	Test Synch Command.
4	MREADCMD	Read Text Command.
5	MRRSPCMD	Read Response (Normal).
6	MRREQCMD	Read Response (To ENQ).
7	MPREPCMD	Prepare Command.
8	MWRITCMD	Write Text Command.
9	MWRSPCMD	Write Response Command.
A	MWENQCMD	Send Inquiry Command.
B	MSEOTCMD	Send EOT Command.

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
40	64	IOBCCW2	8	Channel Command Word 2.
48	72	IOBCCW3	8	Channel Command Word 3.
50	80	IOBCCW4	8	Channel Command Word 4.
58	88	IOBCCW5	8	Channel Command Word 5.
60	96	IOBCCW6	8	Channel Command Word 6.
68	104	IOBCCW7	8	Channel Command Word 7.
70	112	IOBCCW8	8	Channel Command Word 8.
78	120	TPBUFST		Variable Length Buffer.

Figure 8.3.3 -- OVERLAY AREA FORMAT

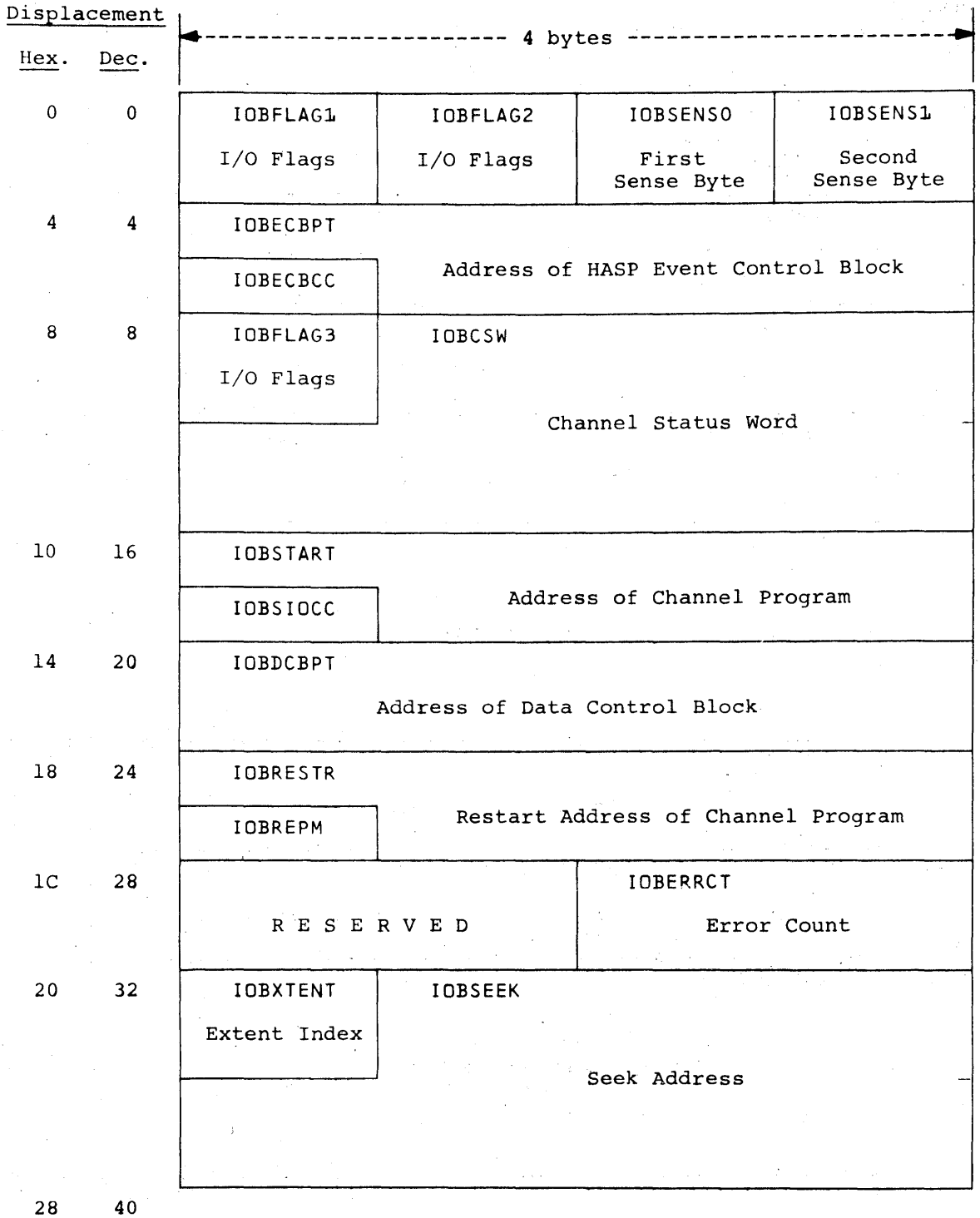


Figure 8.3.3 -- OVERLAY AREA FORMAT (CONTINUED)

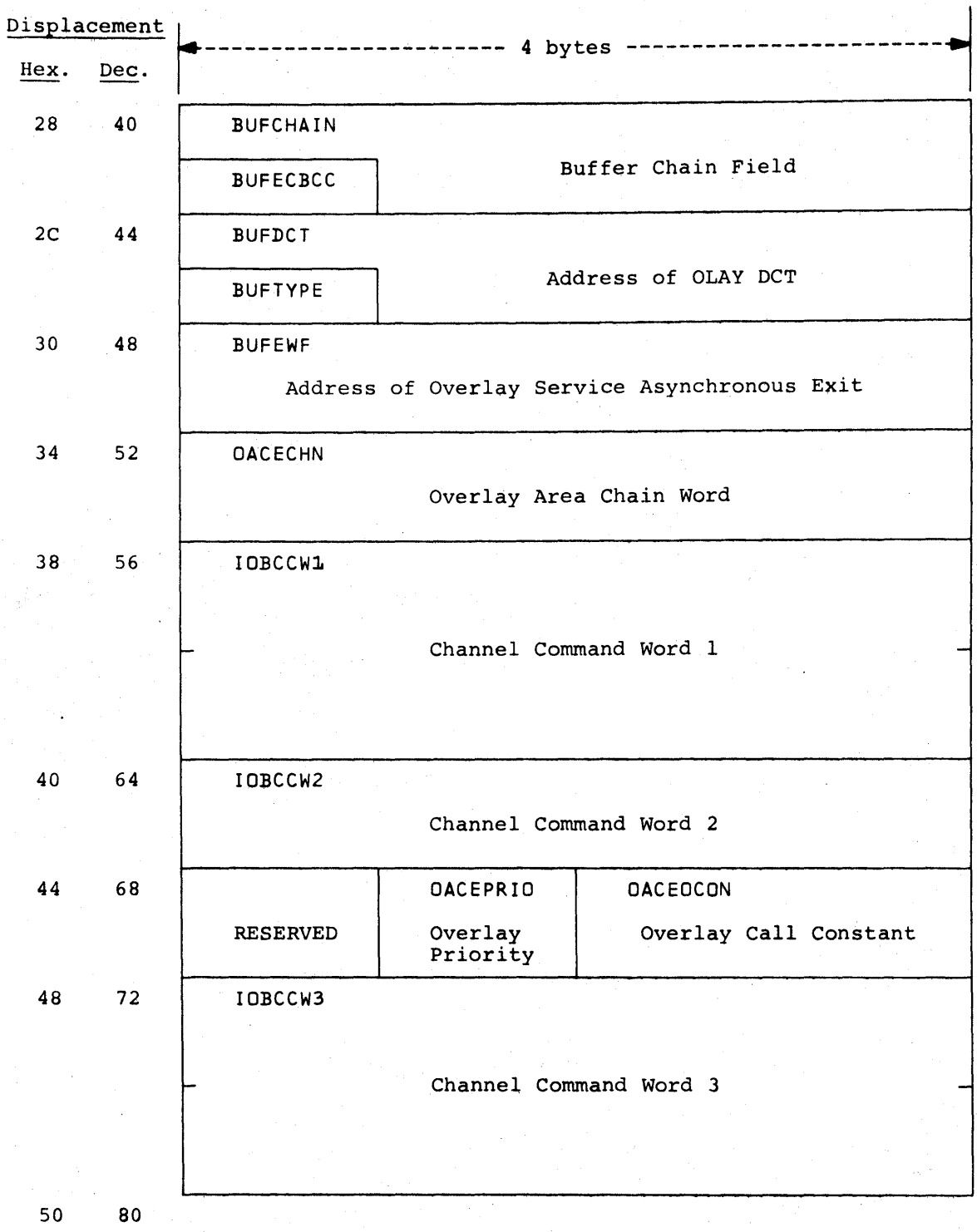


Figure 8.3.3 -- OVERLAY AREA FORMAT (CONTINUED)

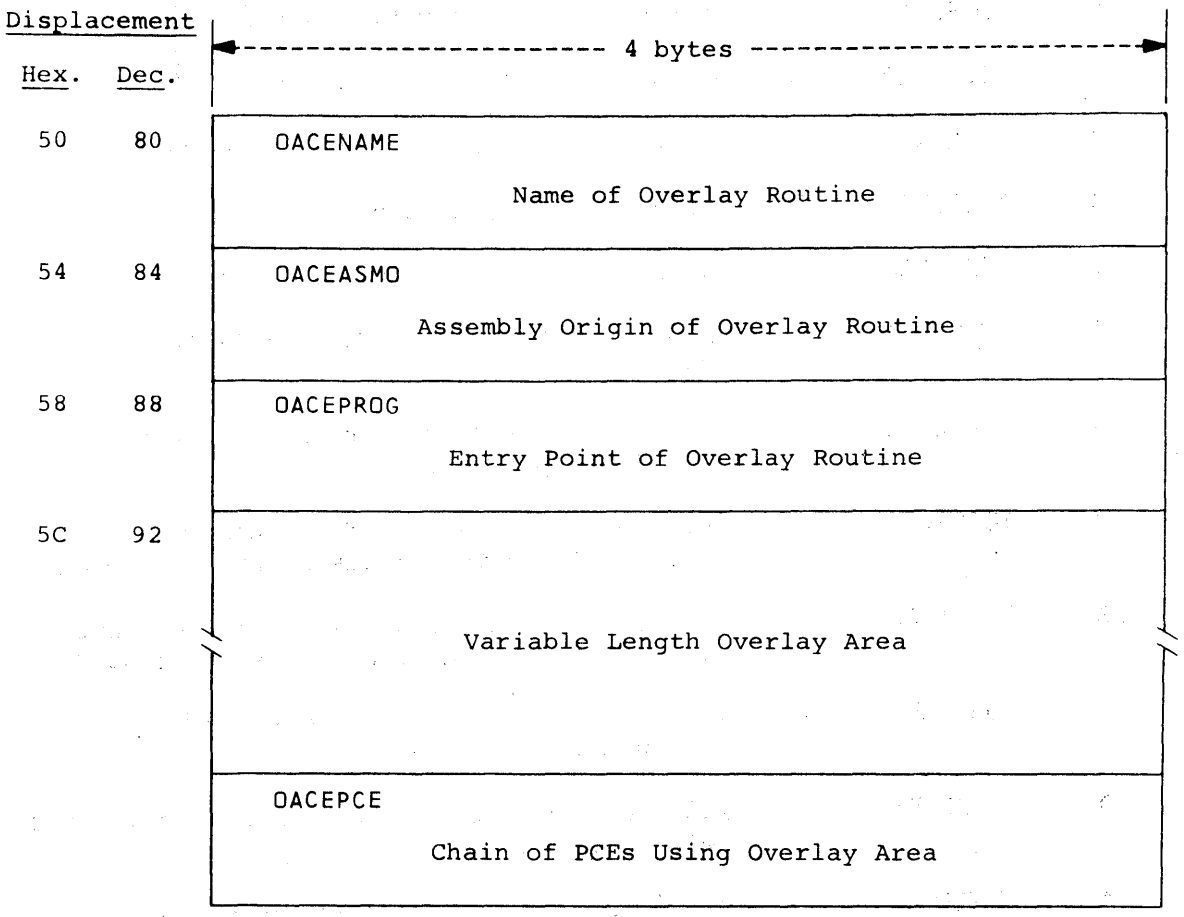


Figure 8.3.3 -- OVERLAY AREA FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
0	0	IOBFLAG1	1	Standard OS/360 IOB Flag Byte.
1	1	IOBFLAG2	1	Standard OS/360 IOB Flag Byte.
2	2	IOBSENS0	1	First Sense Byte (Device Dependent).
3	3	IOBSENS1	1	Second Sense Byte (Device Dependent).
4	4	IOBECBCC	1	Completion Code for Overlay Read.
4	4	IOBECBPT	4	Address of HASP Event Control Block: $HASPECB.
8	8	IOBFLAG3	1	I/O Supervisor Error Routine Flag Byte (Device Dependent).
9	9	IOBCSW	7	Low Order Seven Bytes of the Last CSW that Reflects the Status of the Last Read.
10	16	IOBSIOCC	1	Condition Code Returned After the Execution of the SIO Instruction for the Last Read.
10	16	IOBSTART	4	Address of the Channel Program to be Executed.
14	20	IOBDCBPT	4	Address of the Data Control Block Associated with this IOB.
18	24	IOBREPM	1	Operation Code Used by I/O Supervisor Error Routines for Repositioning Procedures.
18	24	IOBRESTR	4	Restart Address of Channel Program Used by I/O Supervisor Error Routines During Error Correction.
1C	28		2	Reserved.
1E	30	IOBERRCT	2	Used by I/O Supervisor Error Routines to Count Temporary Errors During Retry.
20	32	IOBXTENT	1	The Number of the DEB Extent to be Used for this Read.
21	33	IOBSEEK	7	The Seek Address for the Requested Overlay Routine.

Figure 8.3.3 -- OVERLAY AREA FORMAT (CONTINUED)

Displacement Hex.	Dec.	Field Name	Bytes	Field Description
28	40	BUFECBCC	1	Completion Code for Overlay Read --

Hex. Value	Meaning
00	The Read has not Completed.
7F	The Read has Completed Successfully.
other	The Read has Completed Unsuccessfully.

28	40	BUFCHAIN	4	Buffer Chain Field.
2C	44	BUFTYPE	1	Buffer Type --

Hex. Value	Name	Meaning
40	OLAYBUF	Overlay Area.

2C	44	BUFDCT	4	Address of Overlay Device Control Table.
30	48	BUFEWF	4	Address of Overlay Service Asynchronous Exit.
34	52	OACECHN	4	Overlay Area Chain Word.
38	56	IOBCCW1	8	Channel Command Word 1.
40	64	IOBCCW2	8	Channel Command Word 2.
44	68		1	Reserved for Future Use.
45	69	OACEPRIO	1	Priority of Current Overlay Routine.
46	70	OACEOCON	2	Overlay Constant (OCON) of Current Overlay Routine.
48	72	IOBCCW3	8	Channel Command Word 3.
50	80	OACENAME	4	Name of Overlay Routine.
54	84	OACEASMO	4	Assembly Origin of Overlay Routine.
58	88	OACEPROG		Entry Point of Overlay Routine.
				Variable Length Overlay Area
		OACEPCE	4	Chain of PCEs Using Overlay Area.

Figure 8.4.1 -- CONSOLE MESSAGE BUFFER FORMAT

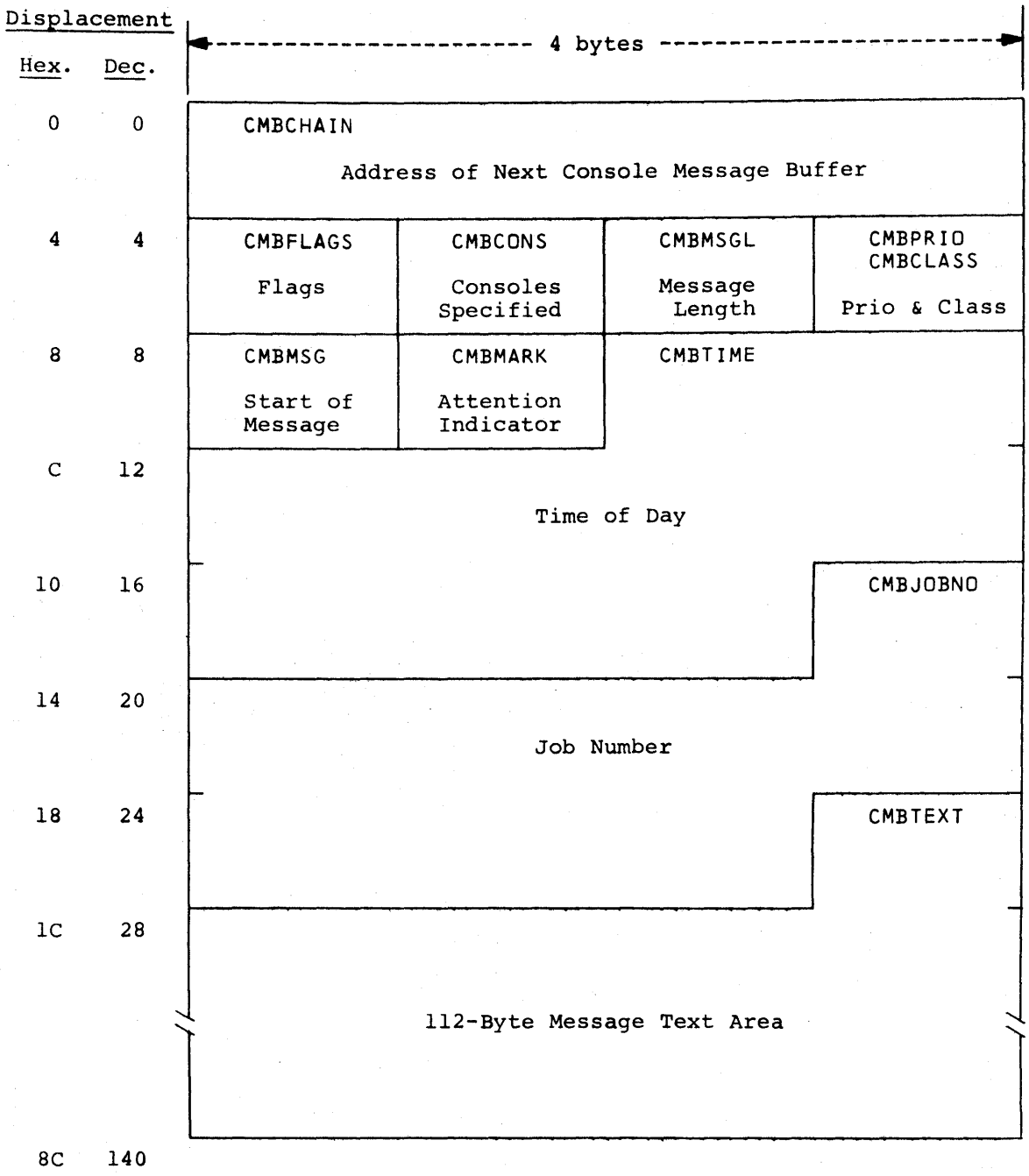


Figure 8.4.1 -- CONSOLE MESSAGE BUFFER FORMAT (CONTINUED)

Displacement Hex.	Dec.	Field Name	Bytes	Field Description
0	0	CMBCHAIN	4	Address of Next Console Message Buffer.
4	4	CMBFLAGS	1	Console Buffer Flags --

Bit	Name	Meaning
0	WCMBFD	CMBCONS Contains Physical Consoles.
1	WCMBFH	Operation Type.
2	WCMBFE	Message for HASP Log Only.
3	WCMBFF	CMBCONS Contains UCMID.
4	WCMBFG	CMBCONS Contains Remote No.
5	WCMBFA	Reserved for
6	WCMBFB	Command
7	WCMBFC	Processor.

5	5	CMBCONS	1	Console Specifications or Remote Number.
6	6	CMBMSGL	1	Message Length.
7	7	CMBCLASS	1	Message Class --

Bits 0-3

Value	Name	Meaning
1	$TRIVIA	Non-Essential Messages.
3	$NORMAL	Normal Messages.
5	$ACTION	Messages Requiring Operator Action.
7	$ALWAYS	Essential Messages.

CMBPRIO — Message Priority --

Bits 4-7

Value	Name	Meaning
1	$LO	Low Priority.
4	$ST	Standard Priority.
7	$HI	High Priority.

8	8	CMBMSG	132	Message Area.
9	9	CMBMARK	1	Asterisk (*) if Message Class is 5 or More.
A	10	CMBTIME	9	Time of Day (HH.MM.SS).
13	19	CMBJOBNO	8	Job Number (If Applicable).
1B	27	CMBTEXT	112	Message Text Area.

Figure 8.5.1 -- DIRECT-ACCESS DEVICE CONTROL TABLE FORMAT

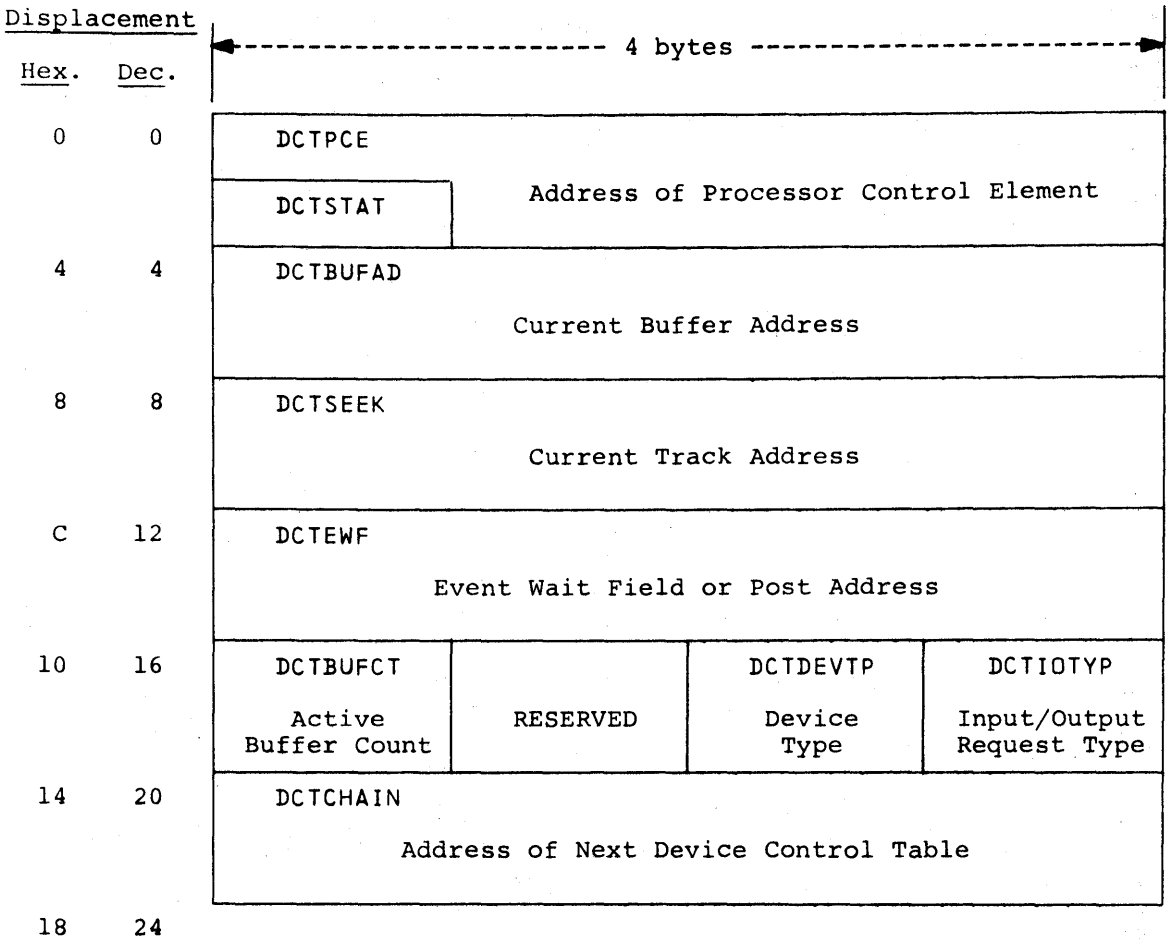


Figure 8.5.1 -- DIRECT-ACCESS DEVICE CONTROL TABLE FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
0	0	DCTSTAT	1	DCT Status --

Bit	Name	Meaning
0	DCTINUSE	DCT is In Use.
1-7		Reserved.

0	0	DCTPCE	4	Address of Processor Control Element.
4	4	DCTBUFAD	4	Address of Current Buffer.
8	8	DCTSEEK	4	Current Track Address.
C	12	DCTEWF	4	Event Wait Field or Post Address.
10	16	DCTBUFCT	1	Number of I/O Requests Outstanding.
11	17		1	Reserved for Future Use.
12	18	DCTDEVTP	1	Device Type --

Hex. Value	Name	Device Type
00	DCTDA	Direct-Access Device.

13	19	DCTIOTYP	1	Input/Output Request Type --

Bit	Name	Meaning
0	DCTREAD	Read Request.
1	DCTWRITE	Write Request.
2-7		Reserved for Future Use.

14	20	DCTCHAIN	4	Address of Next Device Control Table.

Figure 8.5.2 -- OVERLAY DEVICE CONTROL TABLE FORMAT

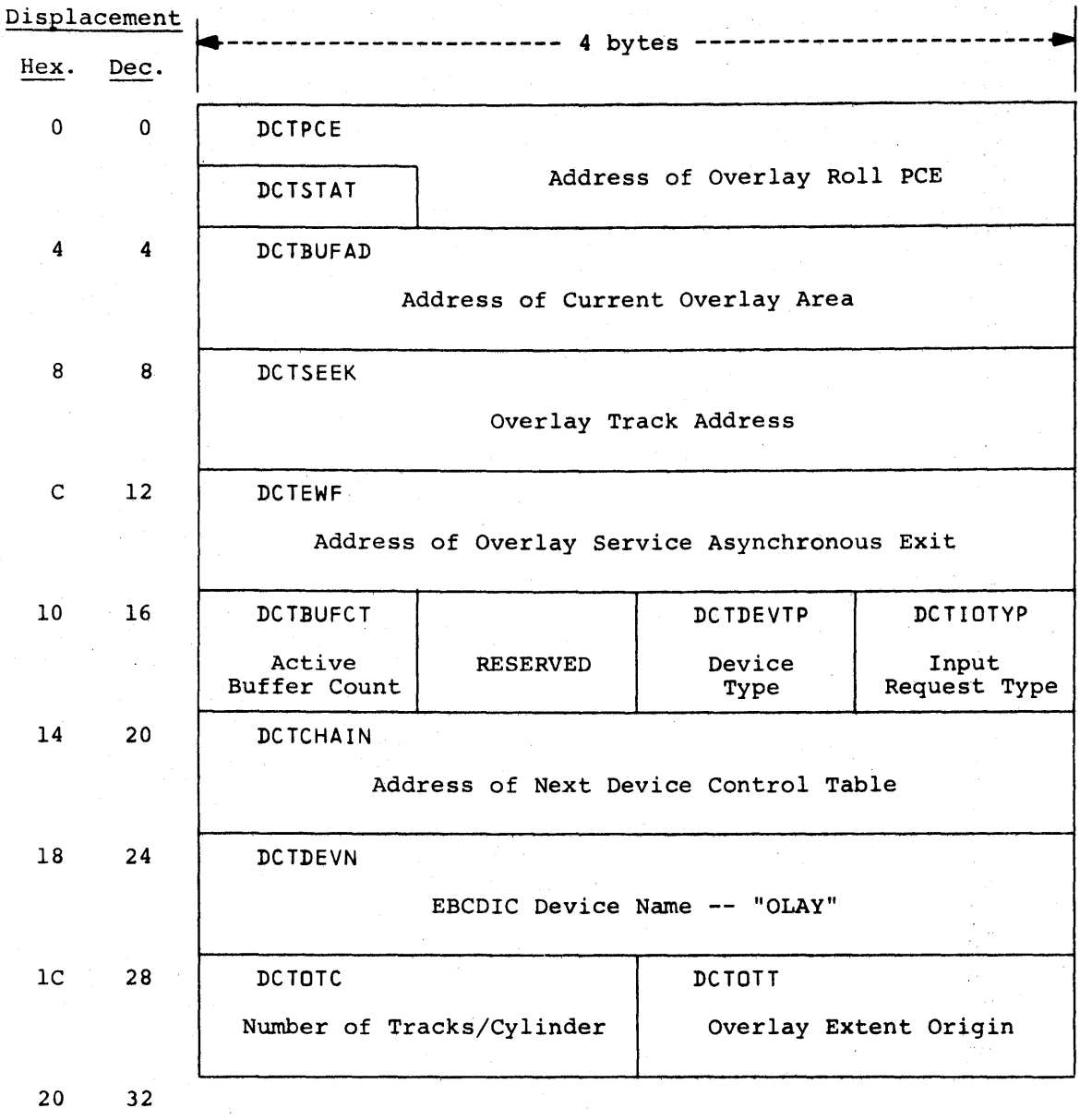


Figure 8.5.2 -- OVERLAY DEVICE CONTROL TABLE FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
0	0	DCTSTAT	1	DCT Status --

Bit	Name	Meaning
0	DCTINUSE	DCT is In Use.
1-7		Reserved.

0	0	DCTPCE	4	Address of Overlay Roll PCE.
4	4	DCTBUFAD	4	Address of Current Overlay Area.
8	8	DCTSEEK	4	Overlay Track Address.
C	12	DCTEWF	4	Address of Overlay Service Asynchronous Exit.
10	16	DCTBUFCT	1	Number of I/O Requests Outstanding.
11	17		1	Reserved for Future Use.
12	18	DCTDEVTP	1	Device Type --

Hex. Value	Name	Meaning
00	DCTDA	Overlay Data Set Resides on SPOOL Disk.
01	DCTOLAY	Overlay Data Set does not Reside on SPOOL Disk.

13	19	DCTIOTYP	1	Input Request Type --

Bit	Name	Meaning
0	DCTREAD	Read Request.
1-7		Reserved for Future Use.

14	20	DCTCHAIN	4	Address of Next Device Control Table.
18	24	DCTDEVN	4	EBCDIC Device Name -- "OLAY".
1C	28	DCTOTC	2	Number of Tracks per Cylinder on Overlay Direct-Access Device.
1E	30	DCTOTT	2	Overlay Extent Origin.

Figure 8.5.3 -- UNIT RECORD DEVICE CONTROL TABLE FORMAT

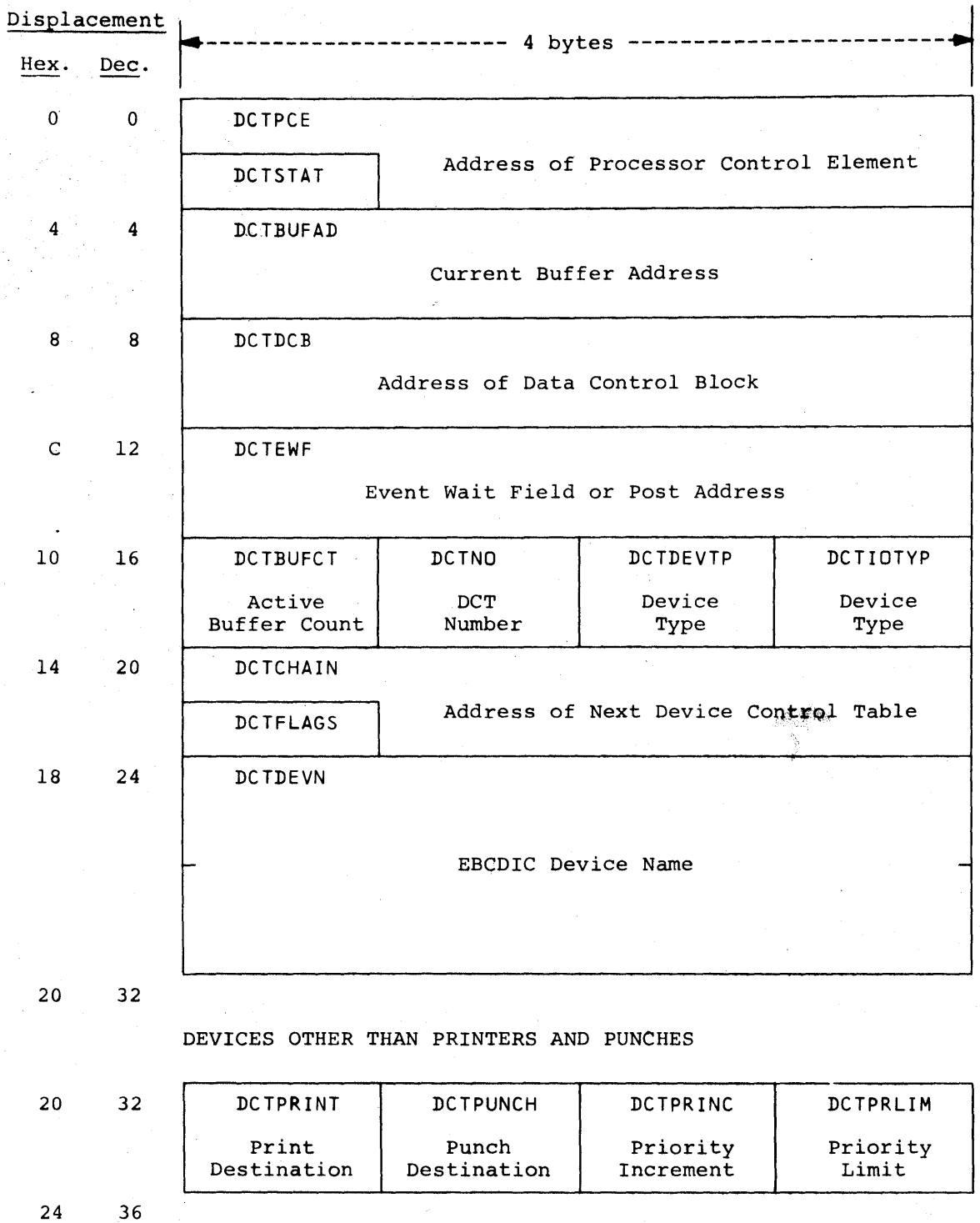


Figure 8.5.3 -- UNIT RECORD DEVICE CONTROL TABLE FORMAT (CONTINUED)

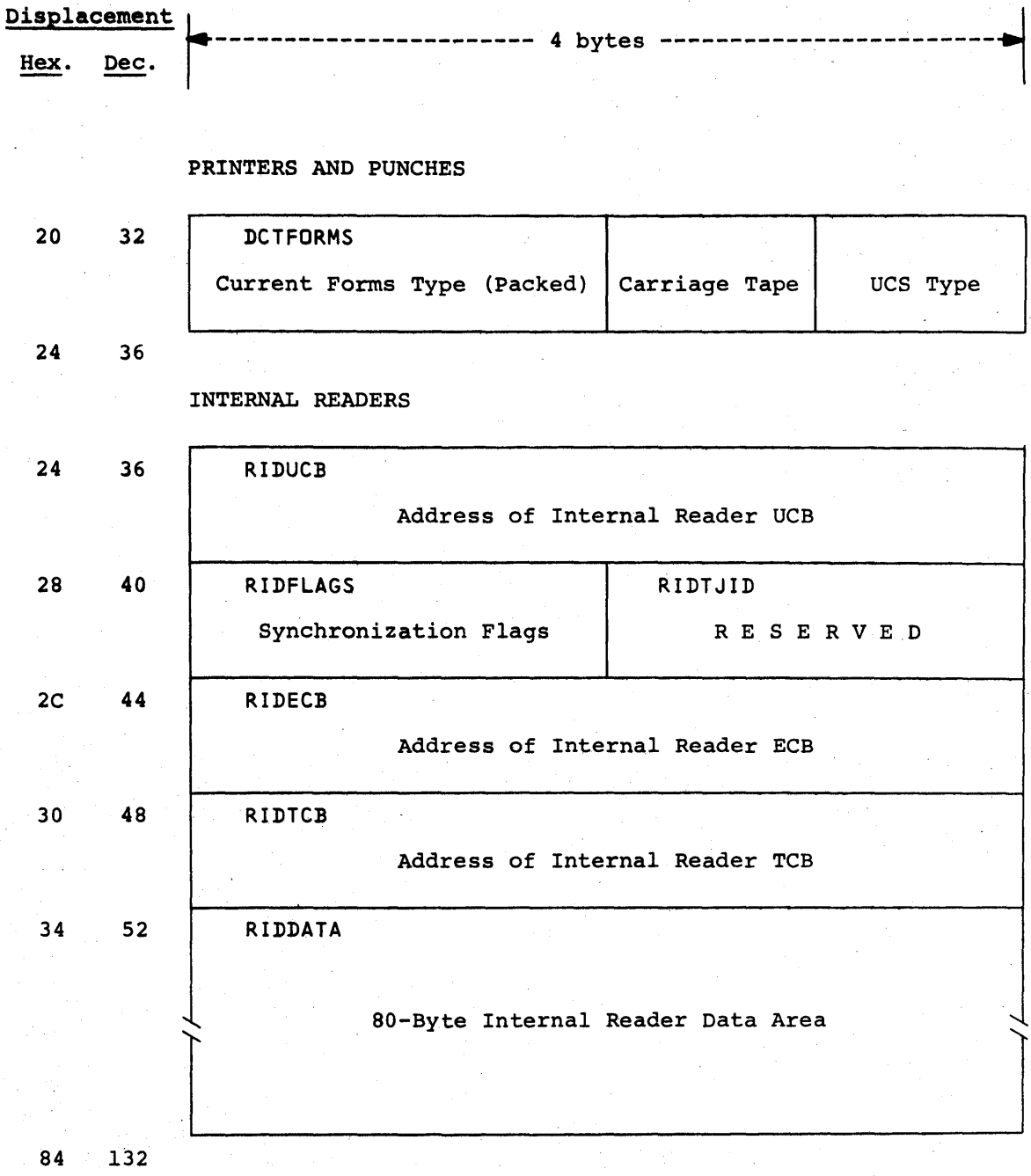


Figure 8.5.3 -- UNIT RECORD DEVICE CONTROL TABLE FORMAT (CONTINUED)

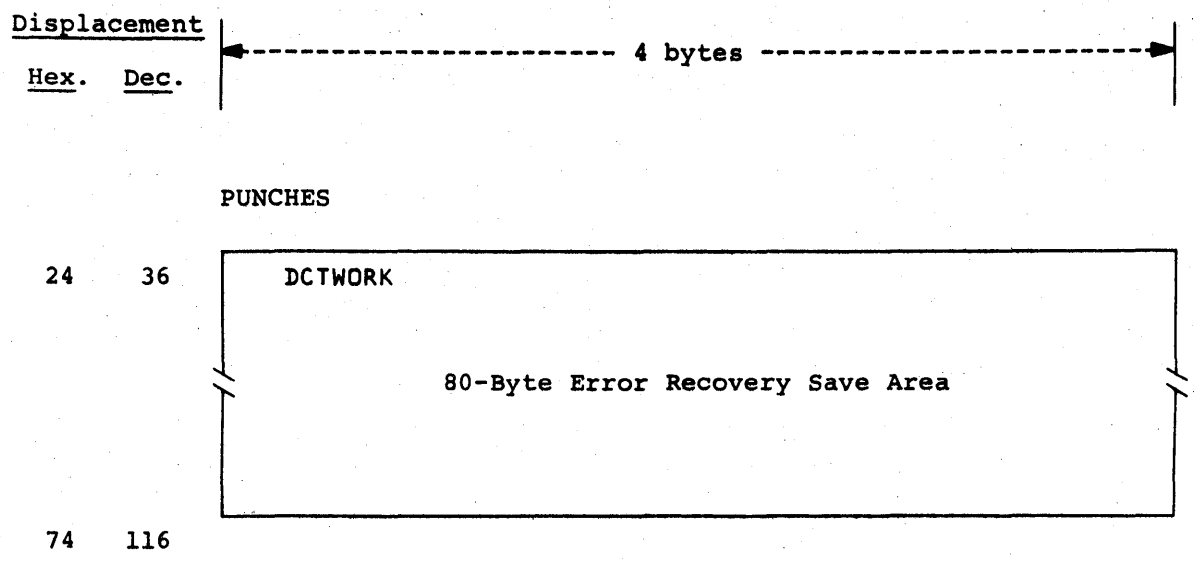


Figure 8.5.3 -- UNIT RECORD DEVICE CONTROL TABLE FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
0	0	DCTSTAT	1	DCT Status --

Bit	Name	Meaning
0	DCTINUSE	DCT is In Use.
1	DCTDRAIN	DCT is Drained.
2	DCTHOLD	DCT is Held.
3-7		Reserved.

0	0	DCTPCE	4	Address of Processor Control Element.
4	4	DCTBUFAD	4	Current Buffer Address.
8	8	DCTDCB	4	Address of Data Control Block for this Unit.
C	12	DCTEWF	4	Event Wait Field or Post Address.
10	16	DCTBUFCT	1	Number of I/O Requests Outstanding.
11	17	DCTNO	1	Device Number.
12	18	DCTDEVTP	1	Device Type --

Hex. Value	Name	Device Type
10	DCTRDR	Card Reader.
11	DCTTPE	Input Tape.
14	DCTINR	Internal Reader.
20	DCTPRT	Printer.
30	DCTPUN	Punch.
40	DCTCON	Console.

13	19	DCTIOTYP	1	Device Type and Console Restrictions --

Bit	Name	Meaning
0-1		Reserved.
2	DCT1053	1053 Console.
3	DCT2260	2260 Console.
4	DCTREJRM	Reserved.
5	DCTREJJB	Job Command Restriction.
6	DCTREJDV	Device Command Restriction.
7	DCTREJSY	System Command Restriction.

Figure 8.5.3 -- UNIT RECORD DEVICE CONTROL TABLE FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
14	20	DCTFLAGS	1	Operator Command Flags --

Bit	Name	Command
0	DCTSTOP	$Z ($STOP)
1	DCTDELET	$C ($DELETE)
2	DCTRSTRT	$E ($RESTART)
3	DCTRPT	$N ($REPEAT)
4	DCTBKSP	$B ($BACKSPACE)
		$F
5	DCTHOLDJ	$T...,H
	DCTSPACE	$T...,C=1
2+4		$I
6-7		Reserved for Future Use.

14	20	DCTCHAIN	4	Address of Next Device Control Table.
18	24	DCTDEVN	8	EBCDIC Device Name.
20	32	DCTPRINT	1	Print Destination.
21	33	DCTPUNCH	1	Punch Destination.
22	34	DCTPRINC	1	Priority Increment.
23	35	DCTPRLIM	1	Priority Limit.
20	32	DCTFORMS	2	Current Forms Type (Packed).
22	34		1	Carriage Tape --

Bit	Name	Meaning
0	DCTFSPEC	Special Forms Routing.
1	DCTFOPER	Operator Controlled Forms.
2-7		Carriage Tape Type.

23	35		1	Universal Character Set --

Bit	Name	Meaning
0	DCTIDSEP	Generate Separator Page/Card.
1	DCTOPACT	Operator Action Allowed.
2-7		UCS Type.

Figure 8.5.3 -- UNIT RECORD DEVICE CONTROL TABLE FORMAT (CONTINUED)

Displacement Hex.	Dec.	Field Name	Bytes	Field Description
24	36	RIDUCB	4	Address of Internal Reader UCB.
28	40	RIDFLAGS	2	Synchronization Flags --
			Byte 1	Bit / Name / Meaning
				0 RIDPOST User Waiting for POST.
				1 RIDBUSY I/O Simulation in Progress.
				2-7 Reserved for Future Use.
			Byte 2	Reserved for Future Use.
2A	42	RIDTJID	2	Reserved for Future Use.
2C	44	RIDECB	4	Address of Internal Reader ECB.
30	48	RIDTCB	4	Address of Internal Reader TCB.
34	52	RIDDATA	80	Internal Reader Data Area.
24	36	DCTWORK	80	Punch Error Recovery Save Area.

Figure 8.5.4 -- LINE DEVICE CONTROL TABLE FORMAT

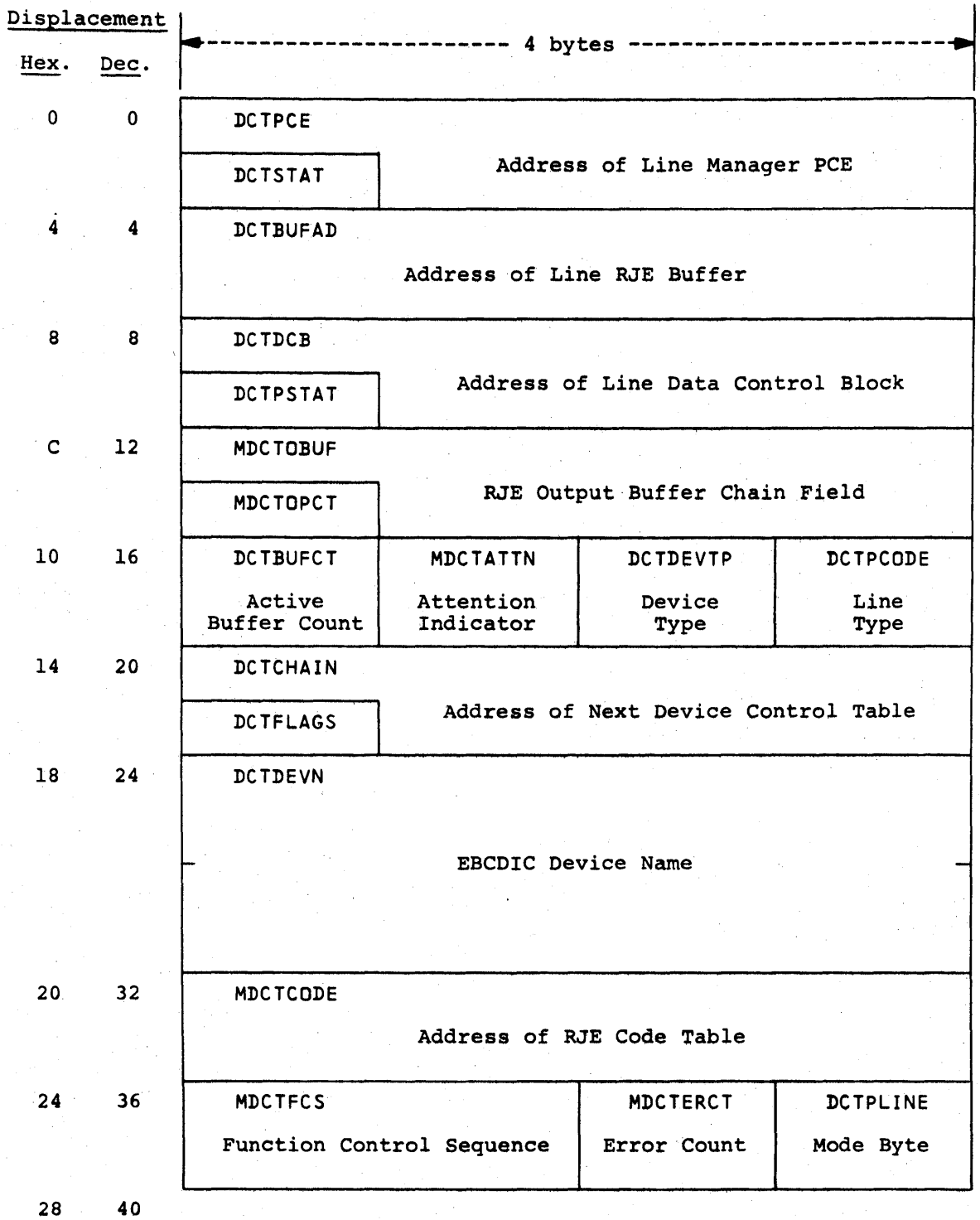


Figure 8.5.4 -- LINE DEVICE CONTROL TABLE FORMAT (CONTINUED)

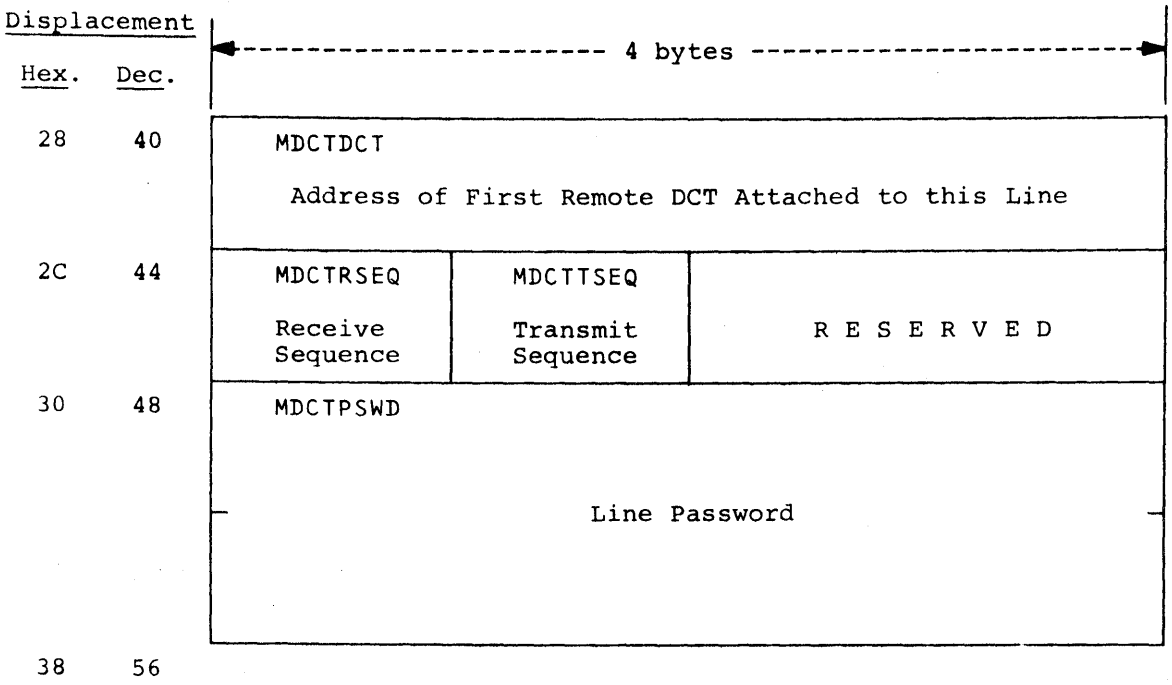


Figure 8.5.4 -- LINE DEVICE CONTROL TABLE FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
0	0	DCTSTAT	1	DCT Status --

Bit	Name	Meaning
0	DCTINUSE	DCT is In Use.
1	DCTDRAIN	DCT is Drained.
2-7		Reserved.

0	0	DCTPCE	4	Address of Line Manager PCE.
4	4	DCTBUFAD	4	Address of Line RJE Buffer.
8	8	DCTPSTAT	1	Line Flags --

Bit	Name	Meaning
0	DCTLOGAL	Log Every Channel End.
1	DCTLEASE	Leased Line.
2	DCTETX	An ETX has been Received.
3	DCTSOFF	A /*SIGNOFF Card has been Processed.
4-7		Reserved for Future Use.

8	8	DCTDCB	4	Address of Line Data Control Block.
C	12	MDCTOPCT	1	MULTI-LEAVING Terminal Open Count.
C	12	MDCTOBUF	4	RJE Output Buffer Chain Field.
10	16	DCTBUFCT	1	Number of I/O Requests Outstanding.
11	17	MDCTATTN	1	Line Attention Requests --

Bit	Name	Meaning
0	MDCTIMER	Timed Action Requested.
1	MDCTPAWS	Line Pause Requested.
2	MDCTJOB1	Job Post Indicator 1.
3	MDCTJOB2	Job Post Indicator 2.
2+3	MDCTJOB	Job Post Indication.
4-7		Reserved for Future Use.

Figure 8.5.4 -- LINE DEVICE CONTROL TABLE FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
12	18	DCTDEVTP	1	Device Type --

Hex. Value	Name	Device Type
02	DCTLNE	Line.

13	19	DCTPCODE	1

Line Type --

Bit	Name	Value	Meaning
0	DCTPBSC	0	STR Line.
		1	BSC Line.
1	DCTPTRSP	0	No Transparency.
		1	Transparency.
2	DCTPASCI	0	EBCDIC Code.
		1	USASCII Code.
3	DCTPHASP		Reserved.
4-5			Reserved.
6	DCTPWIDE	0	Low-Speed Line.
		1	Wide-Band Line.
7	DCTPHALF	0	Half-Duplex Line.
	DCTPFULL	1	Full-Duplex Line.

14	20	DCTFLAGS	1

Operator Command Flags --

Bit	Name	Command
0-1		Reserved.
2	DCTRSTRT	$E ($RESTART) -- Abort.
3-7		Reserved.

14	20	DCTCHAIN	4	Address of Next Device Control Table.
18	24	DCTDEVN	8	EBCDIC Device Name.
20	32	MDCTCODE	4	Address of RJE Code Table.
24	36	MDCTFCS	2	Last Function Control Sequence Received.
26	38	MDCTERCT	1	Line Error Count/Indicator.
27	39	DCTPLINE	1	SDA Mode Byte.
28	40	MDCTDCT	4	Address of First Remote DCT Attached to this Line.

Figure 8.5.4 -- LINE DEVICE CONTROL TABLE FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
2C	44	MDCTRSEQ	1	Receive Block Sequence Count.
2D	45	MDCTTSEQ	1	Transmit Block Sequence Count.
2E	46		2	Reserved for Future Use.
30	48	MDCTPSWD	8	Line Password.

Figure 8.5.5 -- REMOTE DEVICE CONTROL TABLE FORMAT

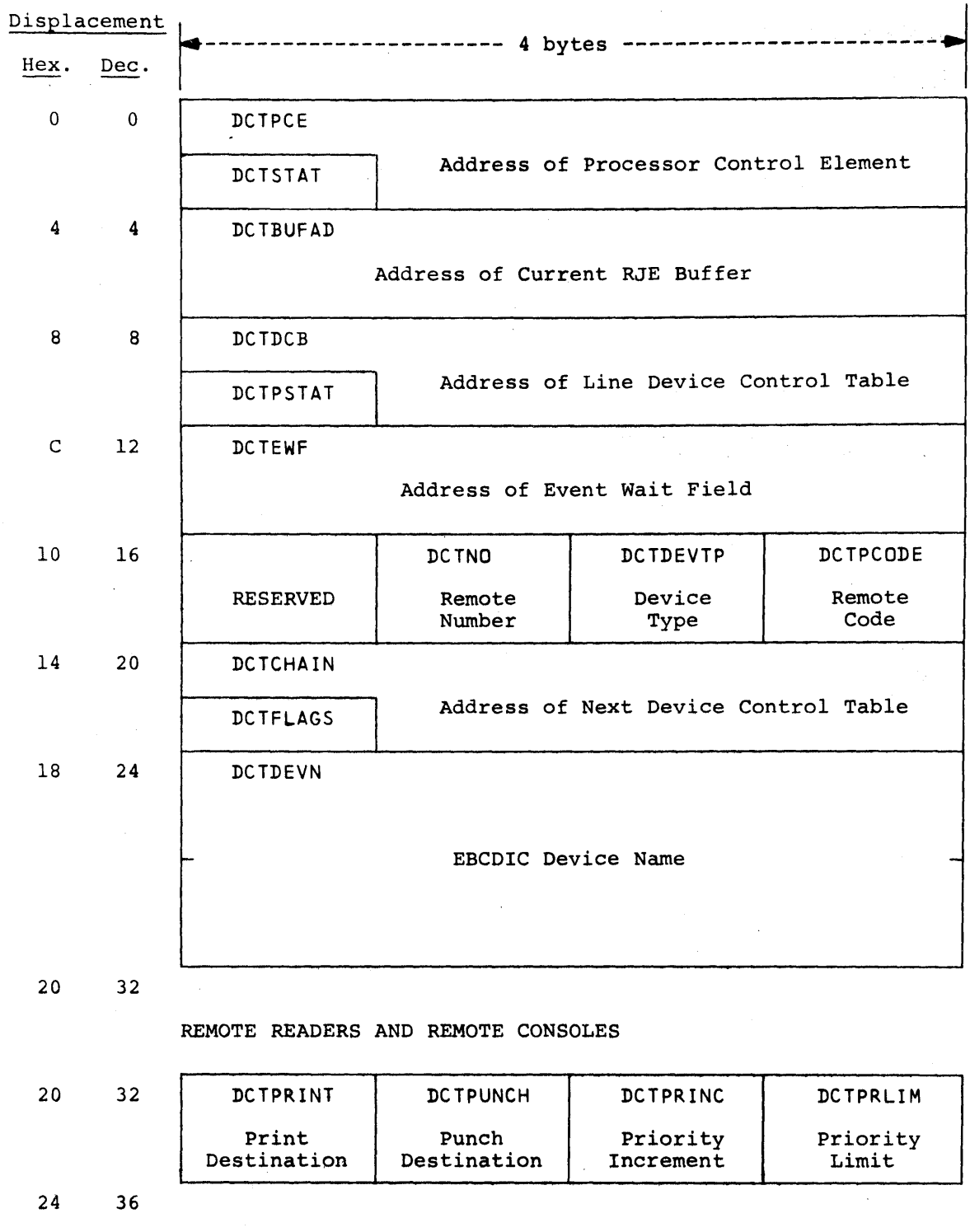


Figure 8.5.5 -- REMOTE DEVICE CONTROL TABLE FORMAT (CONTINUED)

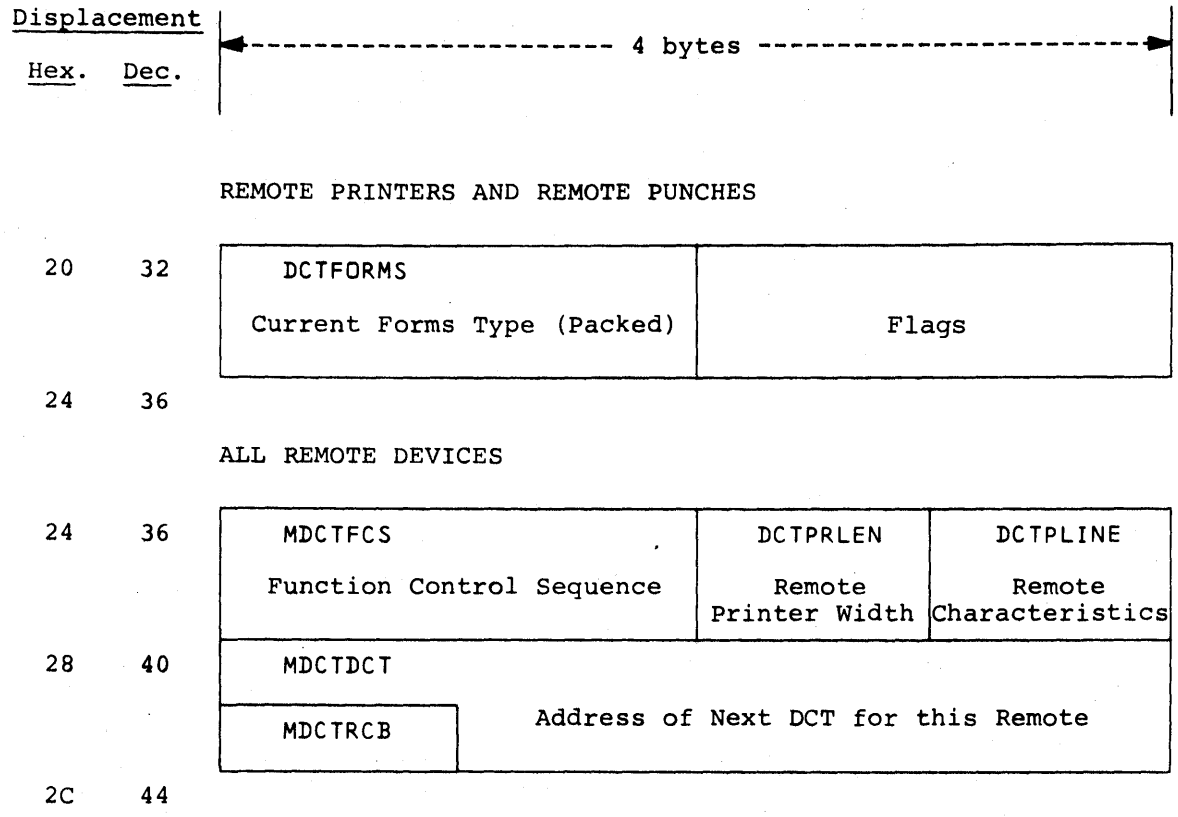


Figure 8.5.5 -- REMOTE DEVICE CONTROL TABLE FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
0	0	DCTSTAT	1	DCT Status --

Bit	Name	Meaning
0	DCTINUSE	DCT is In Use.
1	DCTDRAIN	DCT is Drained.
2	DCTHOLD	DCT is Held.
3-7		Reserved.

0	0	DCTPCE	4	Address of Processor Control Element.
4	4	DCTBUFAD	4	Address of Current RJE Buffer.
8	8	DCTPSTAT	1	Remote Flags --

Bit	Name	Meaning
0-2		Reserved for Future Use.
3	DCTEOF	An EOF has been Detected.
4	DCTSINON	DCT is Attached to Line DCT.
5	DCTPOST	I/O Complete Flag.
6	DCTABORT	Transmission was Aborted.
7	DCTPBUF	Remote has Output Buffer.

8	8	DCTDCB	4	Address of Line Device Control Table.
C	12	DCTEWF	4	Address of Event Wait Field.
10	16		1	Reserved -- Must be zero.
11	17	DCTNO	1	Remote Number.
12	18	DCTDEVTP	1	Device Type --

Hex. Value	Name	Meaning
12	DCTRJR	Remote Reader.
22	DCTRPR	Remote Printer.
32	DCTRPU	Remote Punch.
42	DCTRCON	Remote Console.

Figure 8.5.5 -- REMOTE DEVICE CONTROL TABLE FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
13	19	DCTPCODE	1	Remote Code --

Bit	Name	Meaning
0		Reserved.
1	DCTPTRSP	Terminal Transparency.
2	DCTPPRES	Hardware Compress Feature.
3	DCTPCON	Terminal Console.
	DCTPMRF	Multiple Record Feature.
		Buffer Expansion, Additional.
4	DCTPTAB	Horizontal Format Control.
5	DCTPROG	Programmable Interface.
6	DCTPVAR	Variable Length Records.
7	DCTPBLK	Blocked Records.
		Buffer Expansion Feature.

14	20	DCTFLAGS	1	Operator Command Flags --

Bit	Name	Command
0	DCTSTOP	$Z ($STOP)
1	DCTDELET	$C ($DELETE)
2	DCTRSTRT	$E ($RESTART)
3	DCTRPT	$N ($REPEAT)
4	DCTBKSP	$B ($BACKSPACE)
		$F
5	DCTHOLDJ	$T...,H
	DCTSPACE	$T...,C=1
2+4		$I
6-7		Reserved for Future Use.

14	20	DCTCHAIN	4	Address of Next Device Control Table.
18	24	DCTDEVN	8	EBCDIC Device Name.
20	32	DCTPRINT	1	Print Destination.
21	33	DCTPUNCH	1	Punch Destination.
22	34	DCTPRINC	1	Priority Increment.
23	35	DCTPRLIM	1	Priority Limit.

Figure 8.5.5 -- REMOTE DEVICE CONTROL TABLE FORMAT (CONTINUED)

Displacement Hex.	Dec.	Field Name	Bytes	Field Description
20	32	DCTFORMS	2	Current Forms Type (Packed).
22	34		2	Flags --

Byte 1

Bit	Name	Meaning
0	DCTFSPEC	Special Forms Routing.
1	DCTFOPER	Operator Controlled Forms.
2-7		Reserved for Future Use.

Byte 2

Bit	Name	Meaning
0	DCTIDSEP	Generate Separator Page/Card.
1	DCTOPACT	Operator Action Allowed.
2-7		Reserved for Future Use.

24	36	MDCTFCS	2	Function Control Sequence Mask --

	Bit	Meaning
Byte 1	0-3	Reserved.
	4	Reader 1 or Printer 1.
	5	Reader 2, Printer 2, or Punch 7.
	6	Reader 3, Printer 3, or Punch 6.
	7	Reader 4, Printer 4, or Punch 5.
Byte 2	0	Reserved.
	1	Remote Console.
	2-3	Reserved.
	4	Reader 5, Printer 5, or Punch 4.
	5	Reader 6, Printer 6, or Punch 3.
	6	Reader 7, Printer 7, or Punch 2.
	7	Punch 1.

26	38	DCTPRLEN	1	Remote Printer Width and Remote Input Size.
27	39	DCTPLINE	1	Remote Characteristics --

Bits 0-3 Adapter/Terminal Characteristics --

Bit	Name	Value	Meaning
0	DCTPBSC	0	STR Adapter.
		1	BSC Adapter.
1	DCTPTRSP	0	No Transparency.
		1	Transparency.
2	DCTPASCI	0	EBCDIC Code.
		1	USASCII Code.
3			Reserved.

Figure 8.5.5 -- REMOTE DEVICE CONTROL TABLE FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
				Remote Characteristics (continued) --
			Bits 4-7	Terminal Type

Hex. Value	Name	Terminal Type
0	DCTP2770	2770, 3780, 1009.
1	DCTPHARD	2780, 1978.
2	DCTP20	360/20 Sub-Model 5, 6.
4	DCTP360	360/22, 25, 30, 40, etc.
6	DCTP20S2	360/20 Sub-Model 2, 4.
8	DCTP1130	1130.
A	DCTPSYS3	System/3.

28	40	MDCTRCB	1	Record Control Byte --

Bits	Meaning
0	Always One.
1-3	Device Number.
4-7	Device Type --

Value	Device Type
1	Output Console.
2	Input Console.
3	Reader.
4	Printer.
5	Punch.

28	40	MDCTDCT	4	Address of Next DCT for this Remote.

Figure 8.6.1 -- JOB QUEUE ELEMENT FORMAT

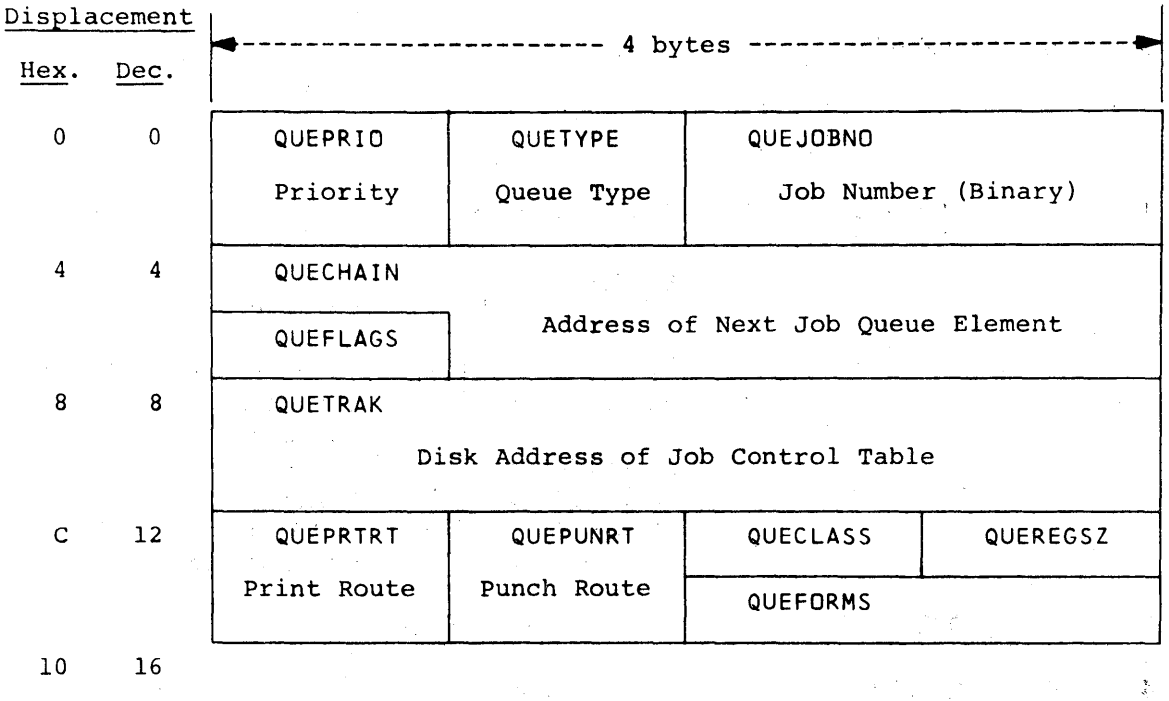


Figure 8.6.1 -- JOB QUEUE ELEMENT FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
0	0	QUEPRIO	1	Queueing Priority --
			Bits 0-3	Priority (0-15).
			Bits 4-7	Reserved.
1	1	QUETYPE	1	Queue Type --

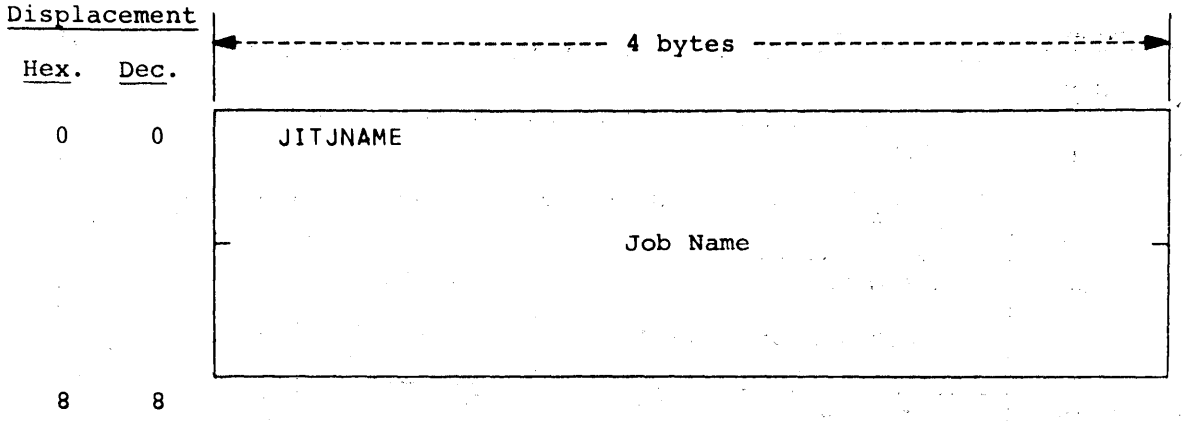
Binary Value	Name	Meaning
lxxxxxxx	QENTBY	Queue Entry is In Use.
xlcccccc	$XEQ	Execution -- cccccc = Job Class - X'C0'.
x0100000	$INPUT	Input Queue.
x0000100	$PRINT	Print Queue.
x0000010	$PUNCH	Punch Queue.
x0000000	$PURGE	Purge Queue.

2	2	QUEJOBNO	2	Job Number (Binary)
4	4	QUEFLAGS	1	Queue Flags --

Bit	Name	Meaning
0	QUEHOLDA	Job Held ($H A)
1	QUEHOLD1	Job Held (Single Job)
2	QUEHOLD2	Job Held (Duplicate Job Name).
3	QUEPURGE	Job Deleted.
4-7	QUEUSECT	Entry Use Count.

4	4	QUECHAIN	4	Address of Next Job Queue Element.
8	8	QUETRAK	4	Track Address of Job Control Table.
C	12	QUEPRTRT	1	Print Routing: 0 = Local. n = Remote n.
D	13	QUEPUNRT	1	Punch Routing: 0 = Local. n = Remote n.
E	14	QUECLASS	1	Sub-Class -- Unused.
E	14	QUEFORMS	2	Forms Code (Packed).
F	15	QUEREGSZ	1	Region Size -- Unused.

Figure 8.7.1 -- JOB INFORMATION TABLE ELEMENT FORMAT



Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
0	0	JITJNAME	8	Job Name.

Figure 8.8.1 -- JOB CONTROL TABLE FORMAT

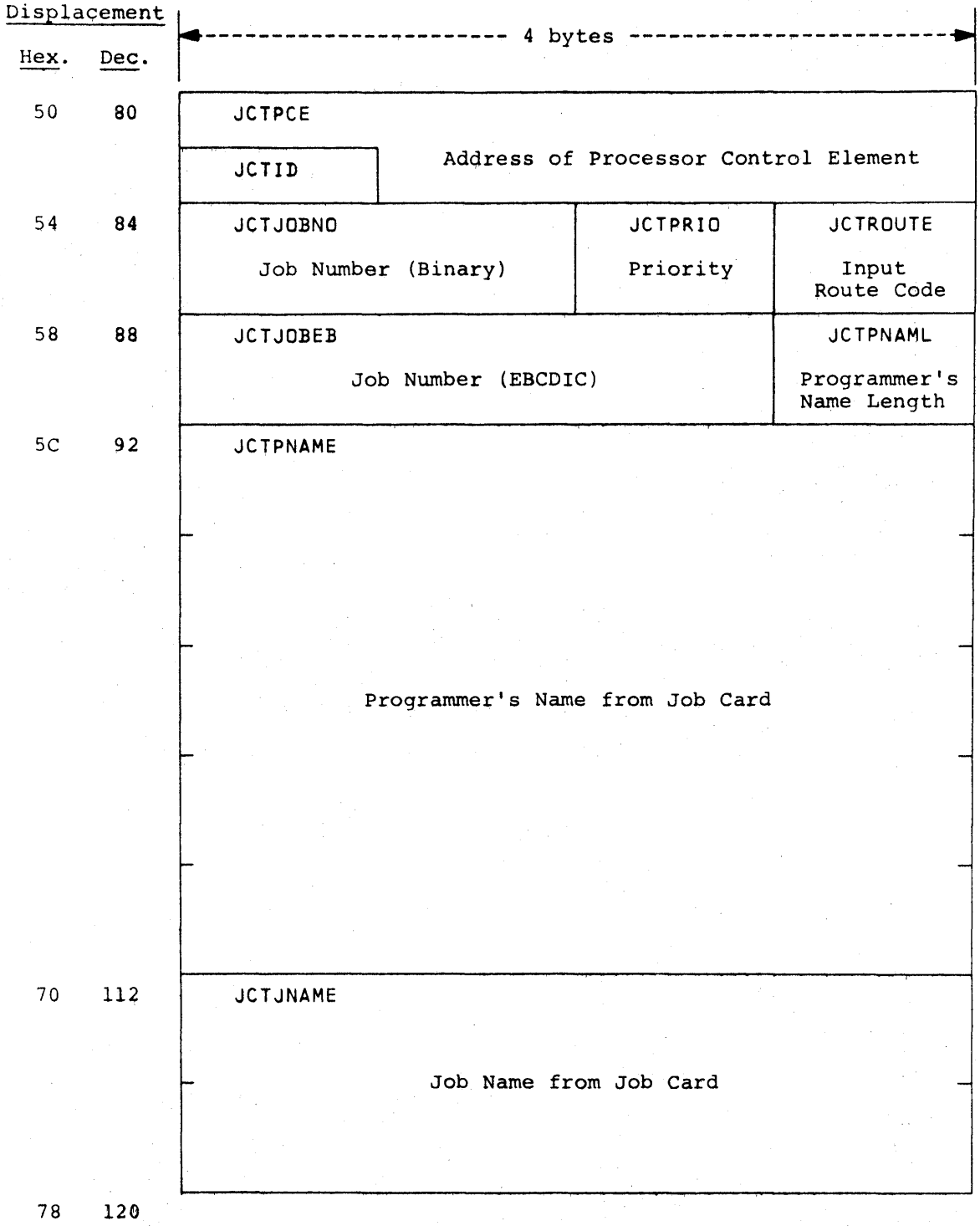


Figure 8.8.1 -- JOB CONTROL TABLE FORMAT (CONTINUED)

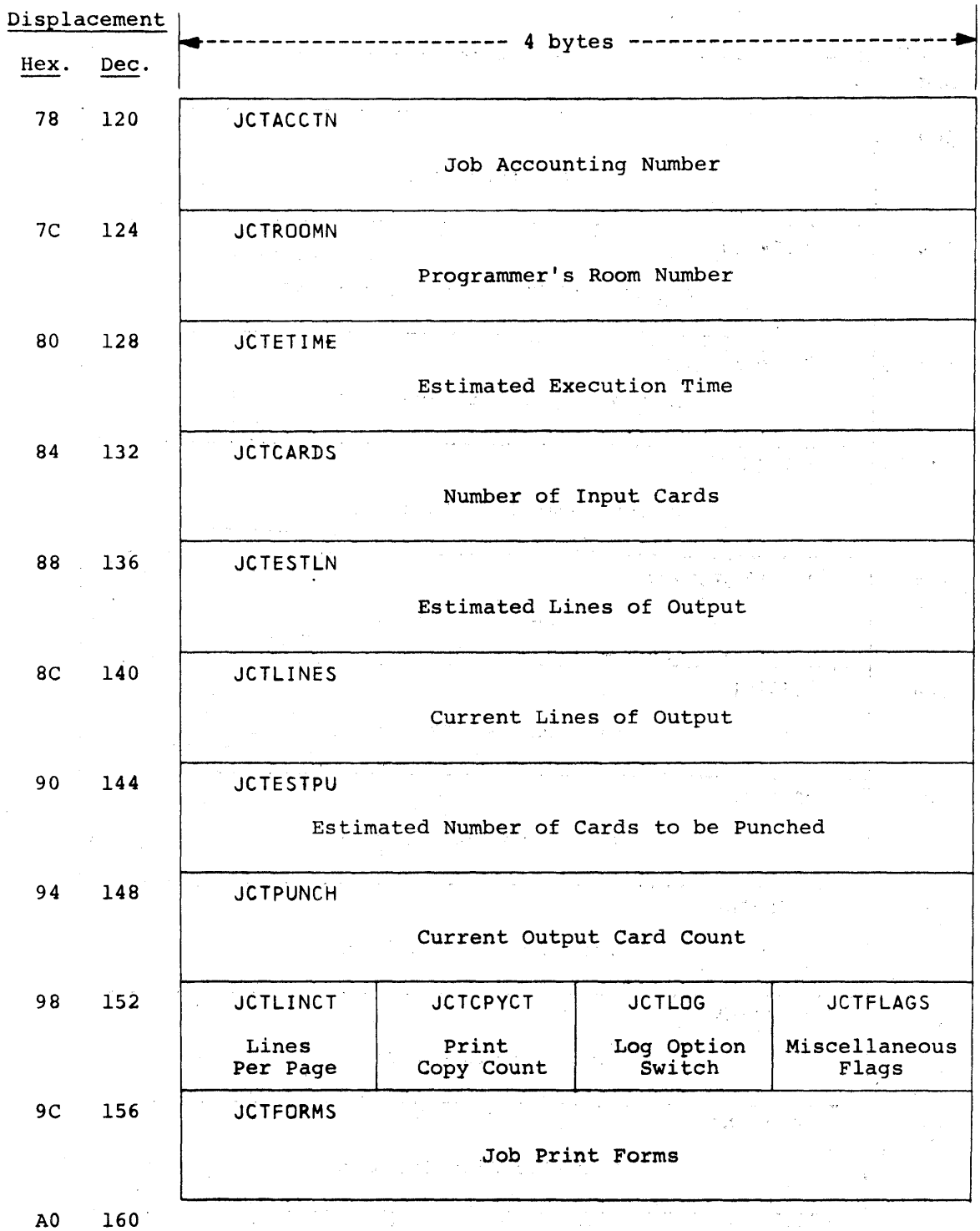


Figure 8.8.1 -- JOB CONTROL TABLE FORMAT (CONTINUED)

Displacement Hex.	Dec.	4 bytes
A0	160	Job Punch Forms
A4	164	JCTPRTCT Current Number of Lines Printed
A8	168	JCTPAGCT Current Number of Pages Printed
AC	172	JCTPUNCT Current Number of Cards Punched
B0	176	JCTRDRON Reader Sign-On Time
B4	180	JCTRDROF Reader Sign-Off Time
B8	184	JCTXEQON Execution Sign-On Time
BC	188	JCTXEQOF Execution Sign-Off Time
C0	192	JCTPRTON Printer Sign-On Time
C4	196	JCTPRTOF Printer Sign-Off Time
C8	200	

Figure 8.8.1 -- JOB CONTROL TABLE FORMAT (CONTINUED)

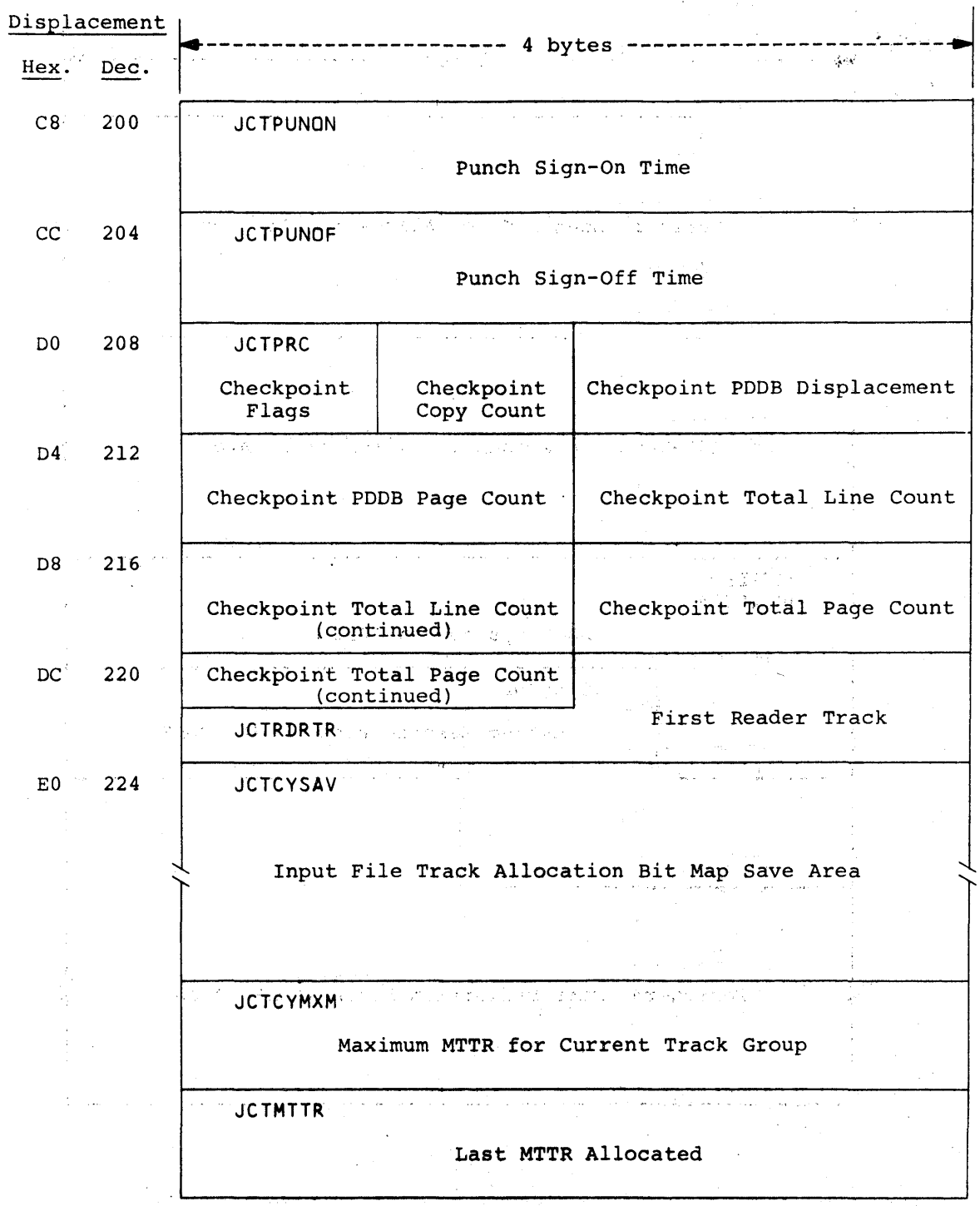


Figure 8.8.1 -- JOB CONTROL TABLE FORMAT (CONTINUED)

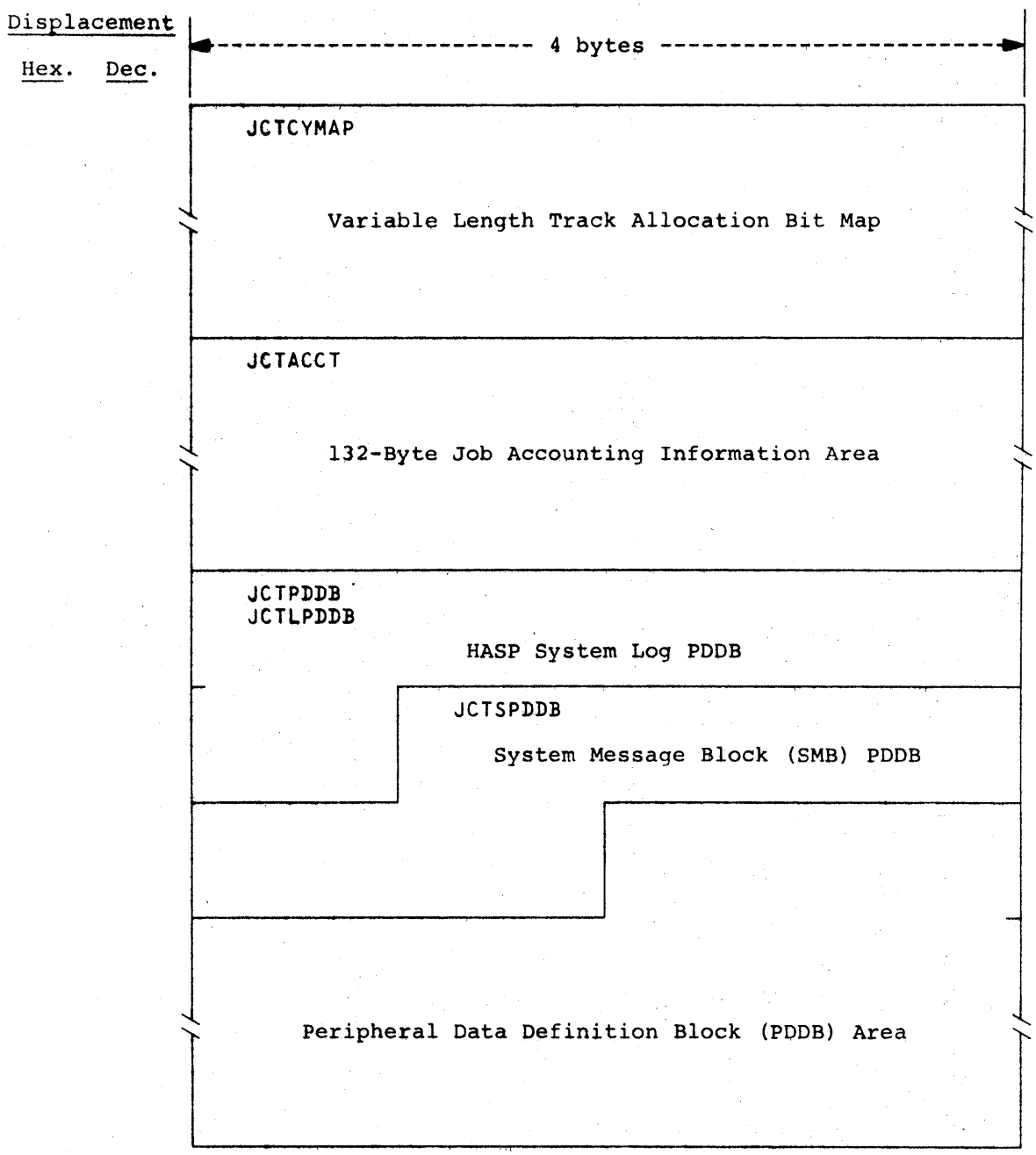


Figure 8.8.1 -- JOB CONTROL TABLE FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
50	80	JCTID	1	JCT Identification -- X'FF'.
50	80	JCTPCE	4	Address of Processor Control Element.
54	84	JCTJOBNO	2	Job Number (Binary).
56	86	JCTPRIO	1	Priority from /*PRIORITY Card.
57	87	JCTROUTE	1	Route Code of Input Device: 0 = Local. n = Remote n.
58	88	JCTJOBEB	3	Job Number (EBCDIC).
5B	91	JCTPNAML	1	Length of Programmer's Name.
5C	92	JCTPNAME	20	Programmer's Name from Job Card.
70	112	JCTJNAME	8	Job Name from Job Card.
78	120	JCTACCTN	4	Job Accounting Number.
7C	124	JCTROOMN	4	Programmer's Room Number.
80	128	JCTETIME	4	Estimated Execution Time.
84	132	JCTCARDS	4	Number of Input Cards.
88	136	JCTESTLN	4	Estimated Lines of Output.
8C	140	JCTLINES	4	Generated Lines of Output.
90	144	JCTESTPU	4	Estimated Number of Cards to be Punched.
94	148	JCTPUNCH	4	Number of Output Cards Generated.
98	152	JCTLINCT	1	Lines per Page.
99	153	JCTCPYCT	1	Number of Copies of Print.
9A	154	JCTLOG	1	Log Option Switch --

EBCDIC Value	Meaning
L	Produce HASP SYSTEM LOG.
N	Do not Produce HASP SYSTEM LOG.

Job Control Table Format -- Page 8.8-6

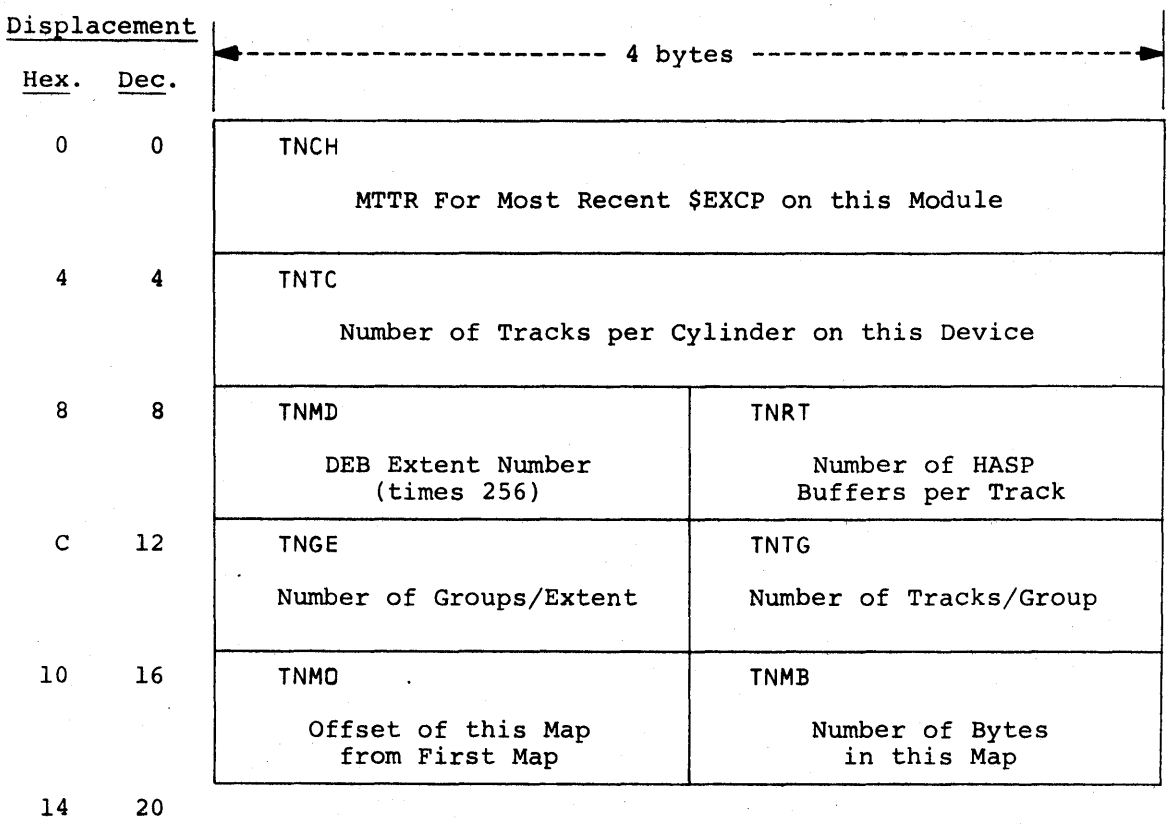
Figure 8.8.1 -- JOB CONTROL TABLE FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
9B	155	JCTFLAGS	1	Miscellaneous Flags --
				Bit 0 — JCTDSRT — Processing Special Forms.
				Bit 1-7 — Count of Input Data Sets SPOOLed by HASP.
9C	156	JCTFORMS	4	Job Print Forms.
A0	160		4	Job Punch Forms.
A4	164	JCTPRTCT	4	Number of Lines Printed.
A8	168	JCTPAGCT	4	Number of Pages Printed.
AC	172	JCTPUNCT	4	Number of Cards Punched.
B0	176	JCTRDRON	4	Reader Sign-On Time.
B4	180	JCTRDROF	4	Reader Sign-Off Time.
B8	184	JCTXEQON	4	Execution Sign-On Time.
BC	188	JCTXEQOF	4	Execution Sign-Off Time.
C0	192	JCTPRTON	4	Print Sign-On Time.
C4	196	JCTPRTOF	4	Print Sign-Off Time.
C8	200	JCTPUNON	4	Punch Sign-On Time.
CC	204	JCTPUNOF	4	Punch Sign-Off Time.
D0	208	JCTPRC	14	Print Checkpoint Element.
DC	220	JCTRDRTR	4	First Reader Track.
E0	224	JCTCYSAV		Variable Length Input File Track Allocation Bit Map Save Area.
		JCTCYMXM	4	Maximum MTTR for Current Track Group.
		JCTMTTR	4	Last MTTR Allocated.
		JCTCYMAP		Variable Length Track Allocation Bit Map.

Figure 8.8.1 -- JOB CONTROL TABLE FORMAT (CONTINUED)

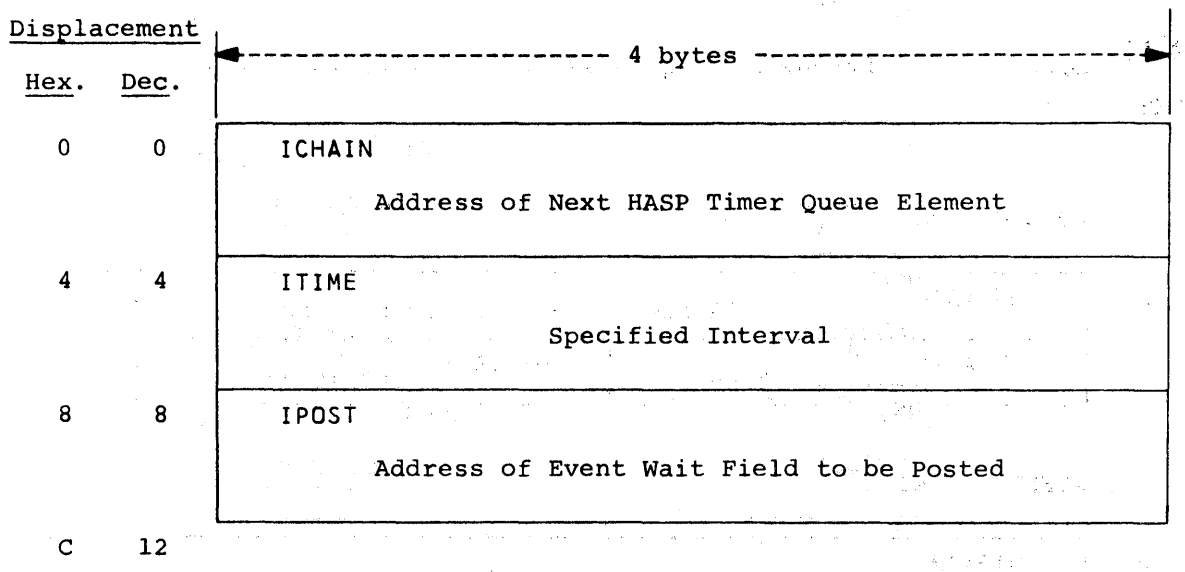
Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
		JCTACCT	132	Job Accounting Information Area.
		JCTPDDB		Peripheral Data Definition Block (PDDB) Area.
		JCTLPDDB	5	HASP System Log PDDB.
		JCTSPDDB	5	System Message Block (SMB) PDDB.

Figure 8.9.1 -- TRACK EXTENT DATA TABLE FORMAT



Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
0	0	TNCH	4	MTTR for Most Recent $EXCP on this Module.
4	4	TNTC	4	Number of Tracks per Cylinder on this Device.
8	8	TNMD	2	DEB Extent Number (times 256).
A	10	TNRT	2	Number of HASP Buffers per Track.
C	12	TNGE	2	Number of Track Groups per Extent.
E	14	TNTG	2	Number of Tracks per Track Group.
10	16	TNMO	2	Offset of This Map from First Map.
12	18	TNMB	2	Number of Bytes in This Map.

Figure 8.10.1 -- TIMER QUEUE ELEMENT



Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
0	0	ICHAIN	4	Address of Next HASP Timer Queue Element.
4	4	ITIME	4	Timer Interval.
8	8	IPOST	4	

Byte 1 Flag Byte --

Bit	Value	Meaning
0	0	Timer Interval has not Expired.
	1	Timer Interval has Expired.
1-7		Reserved.

Bytes 2-4 Address of Event Wait Field to be Posted.

Timer Queue Element Format -- Page 8.10-1

Figure 8.11.1 -- OVERLAY TABLE FORMAT

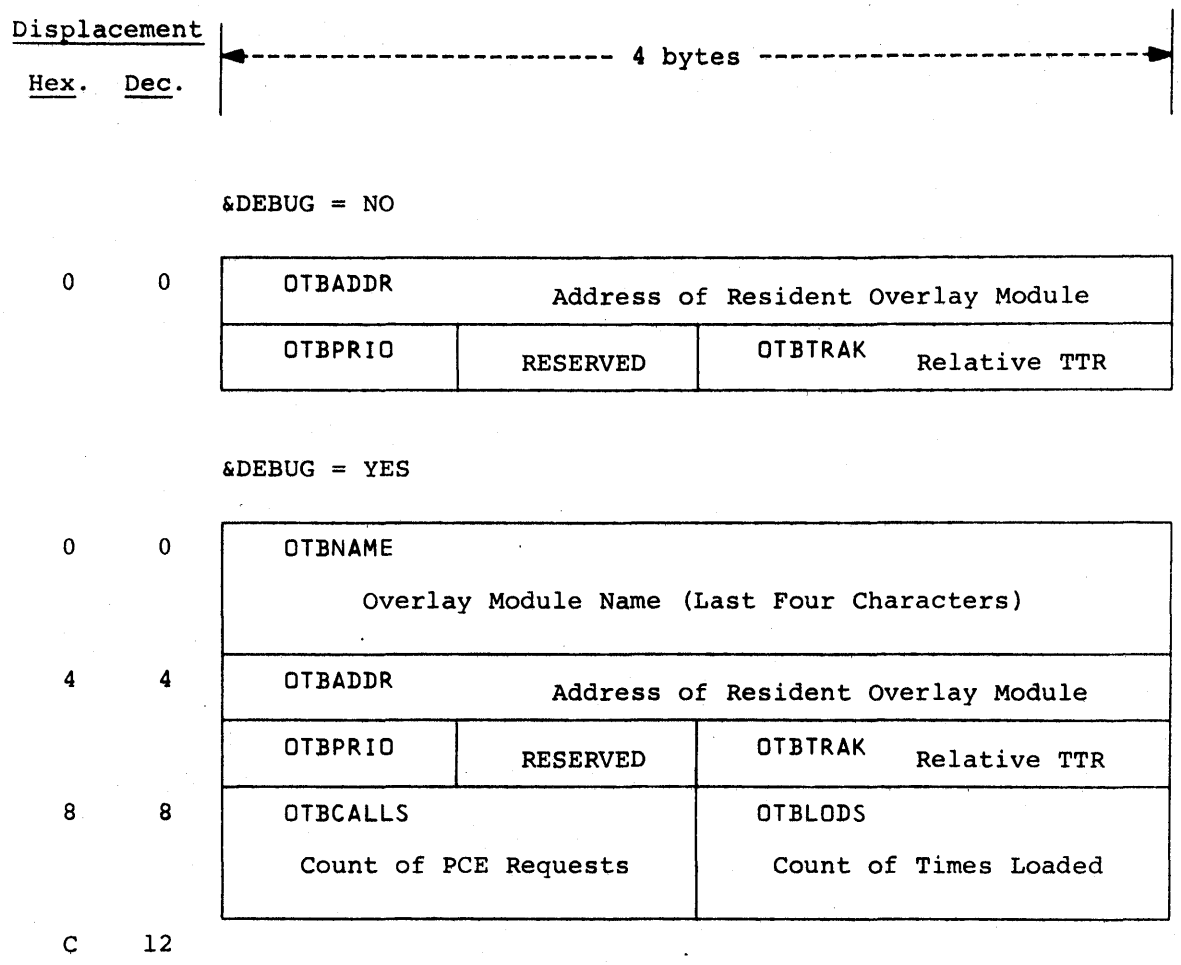


Figure 8.11.1 -- OVERLAY TABLE FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
0	0	OTBADDR	4	Address of Resident Overlay Module.
			Byte 1	X'FF'.
			Bytes 2-4	Addressability Address of Resident Overlay Module -- Assembly Origin - X'50'.
0	0	OTBPRIO	1	Priority of Overlay Module.
1	1		1	Reserved for Future Use.
2	2	OTBTRAK	2	Relative Track and Record Address of Overlay Module.
0	0	OTBNAME	4	Last Four Characters of Overlay Module Name.
8	8	OTBCALLS	2	Number of Times This Overlay Module was Requested.
A	10	OTBLODS	2	Number of Times This Overlay Module was Loaded.

Figure 8.12.1 -- DATA DEFINITION TABLE FORMAT

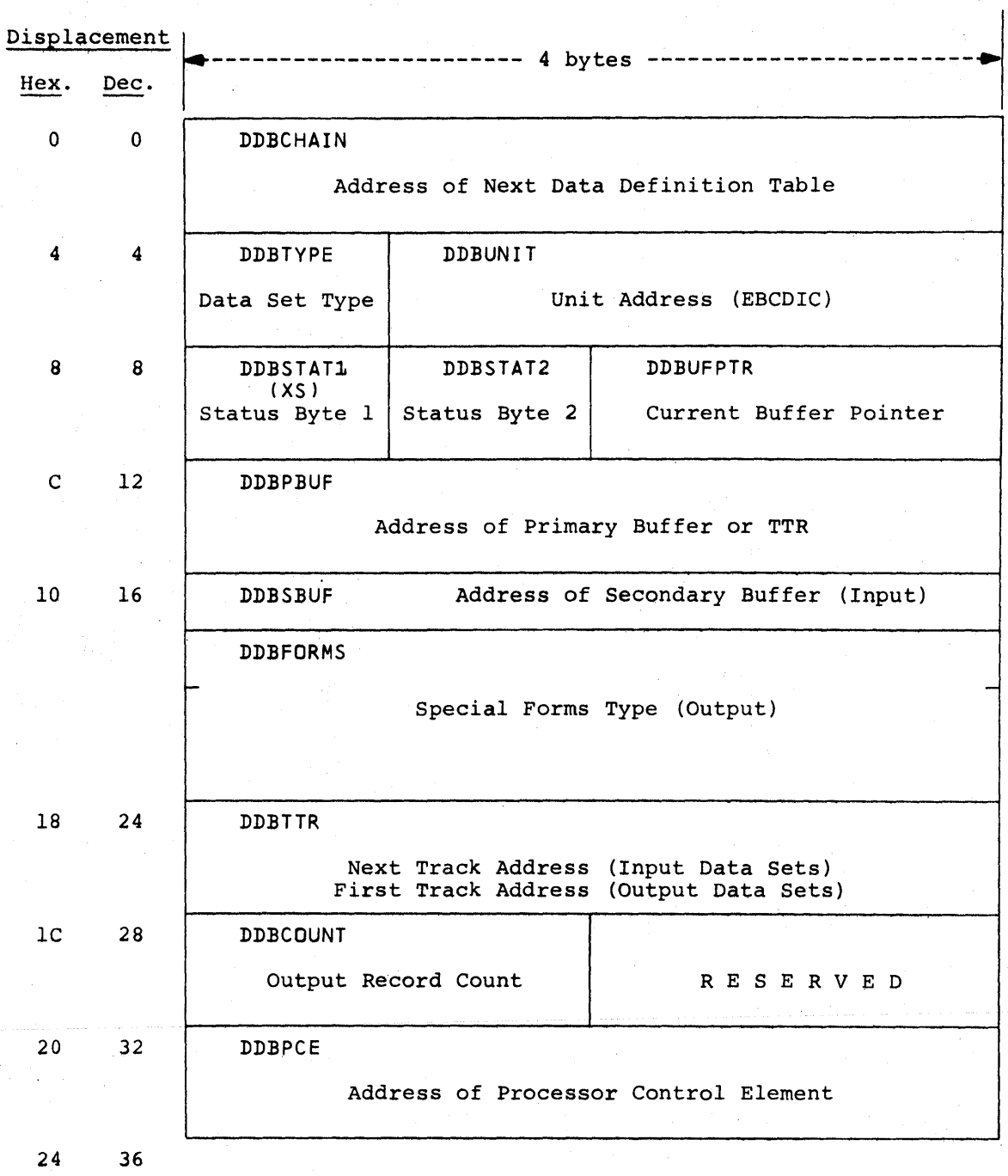


Figure 8.12.1 -- DATA DEFINITION TABLE FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Definition
0	0	DDBCHAIN	4	Address of Next Data Definition Table.
4	4	DDBTYPE	1	Data Set Type --

Hex. Value	Name	Data Set Type
01	XSPROUTE	Special Route SYSOUT.
02	XPRTDDB	Print.
04	XPUNDDB	Punch.
08	XPLOTDDB	Plot.
10	XLOGDDB	Log.
40	XNULLDDB	Dummy (Null).
80	XINDDB	Input.

5	5	DDBUNIT	3	Unit Address (EBCDIC).
8	8	DDBSTAT1 (XS)	1	Status Byte 1 --

Bit	Name	Meaning
0	XSEOD	End of Data on Secondary.
1	XSIOA	I/O Active on Secondary.
2	XSIO	I/O Required on Secondary.
3	XNSB	No Secondary Buffer.
4	XPEOD	End of Data on Primary.
5	XPIOA	I/O Active on Primary.
6	XPIO	I/O Required on Primary.
7	XNPB	No Primary Buffer.

9	9	DDBSTAT2	1	Status Byte 2 --

Bit	Name	Meaning
0	XACT	Action Required on This DDT.
1		Reserved for Future Use.
2	XLOGHEAD	Log Title Switch.
3	XOPEN	DDT has been Used.
4	XUCB	Allocatable UCB Exists.
5	XIOC	I/O Error on Read.
6	XROLL	Roll Output Buffer.
7	XTERM	Terminate DDT.

Figure 8.12.1 -- DATA DEFINITION TABLE FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
A	10	DDBUFPTR	2	Current Displacement of Data in Primary Buffer.
C	12	DDBPBUF	4	Address of Primary Buffer -- Or TTR if No Primary Buffer.
10	16	DDBSBUF	4	Address of Secondary Buffer (Input Only).
10	16	DDBFORMS	8	Special Forms Type (Output Only).
18	24	DDBTTR	4	Input: Next Track Address. Output: First Track Address.
1C	28	DDBCOUNT	2	Output Record Count.
1E	30		2	Reserved for Future Use.
20	32	DDBPCE	4	Address of Processor Control Element.

Figure 8.13.1 -- PARTITION INFORMATION TABLE FORMAT

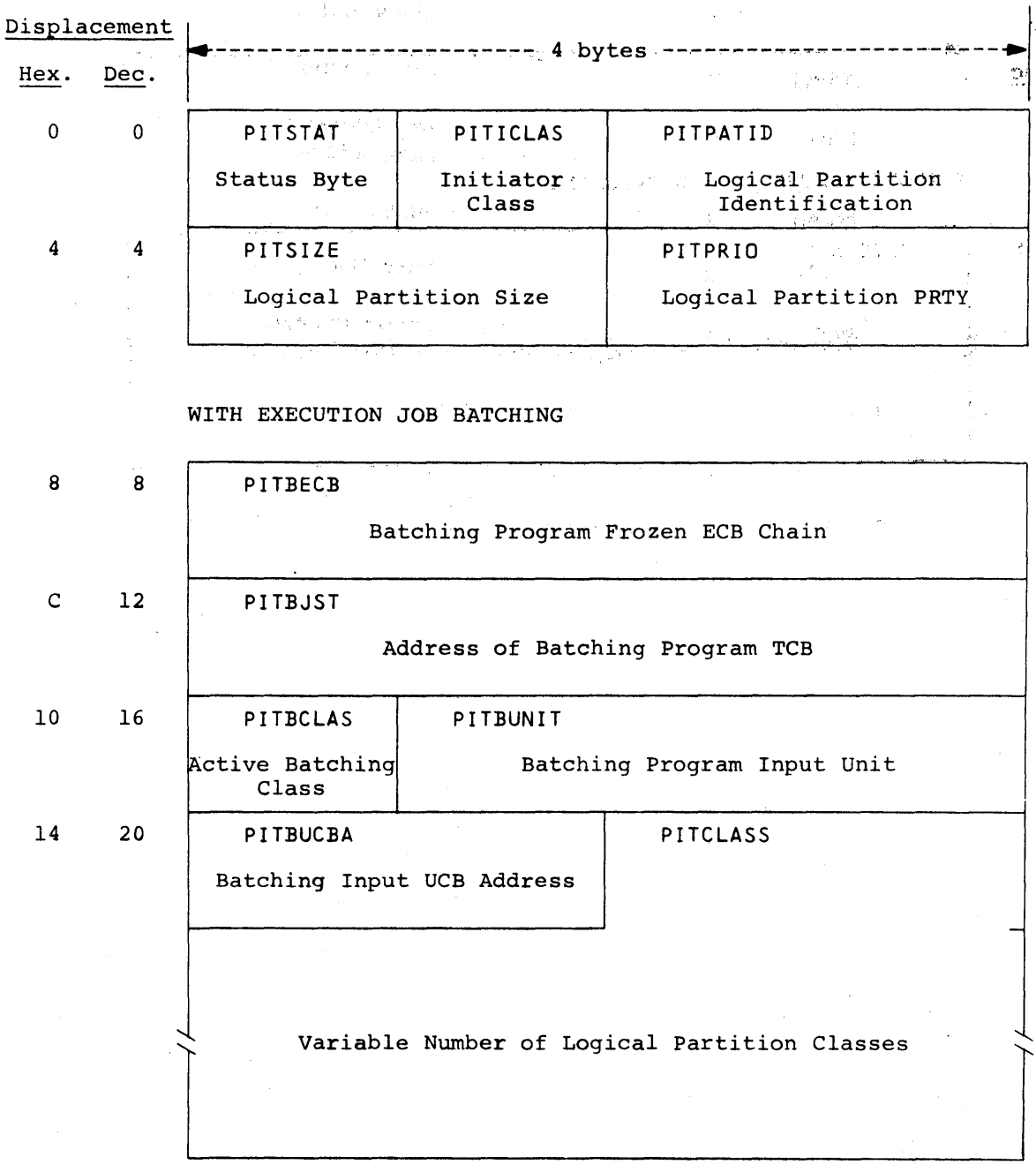


Figure 8.13.1 -- PARTITION INFORMATION TABLE FORMAT (CONTINUED)

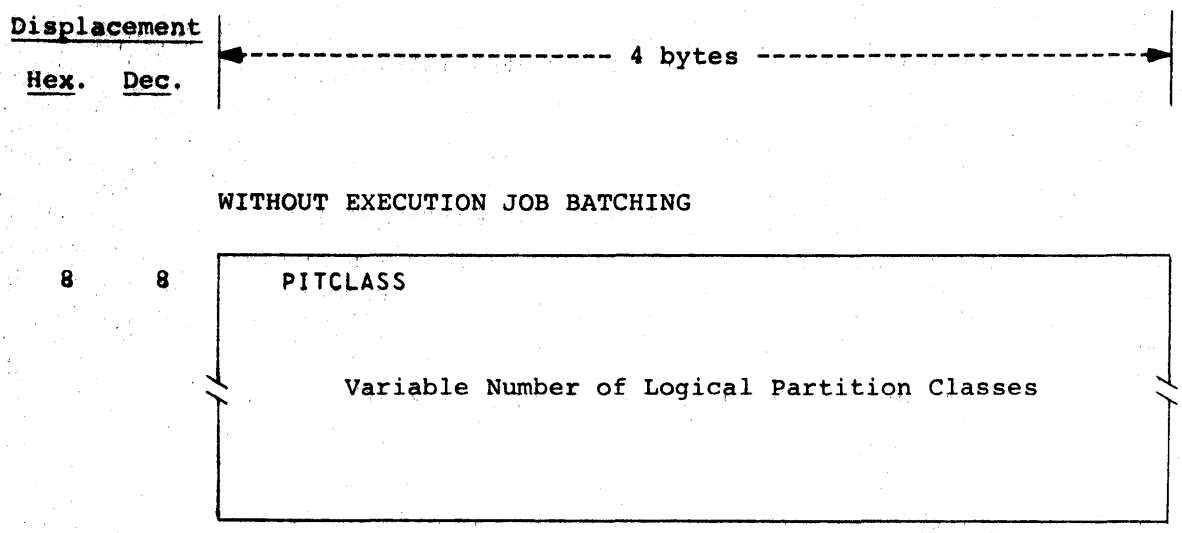


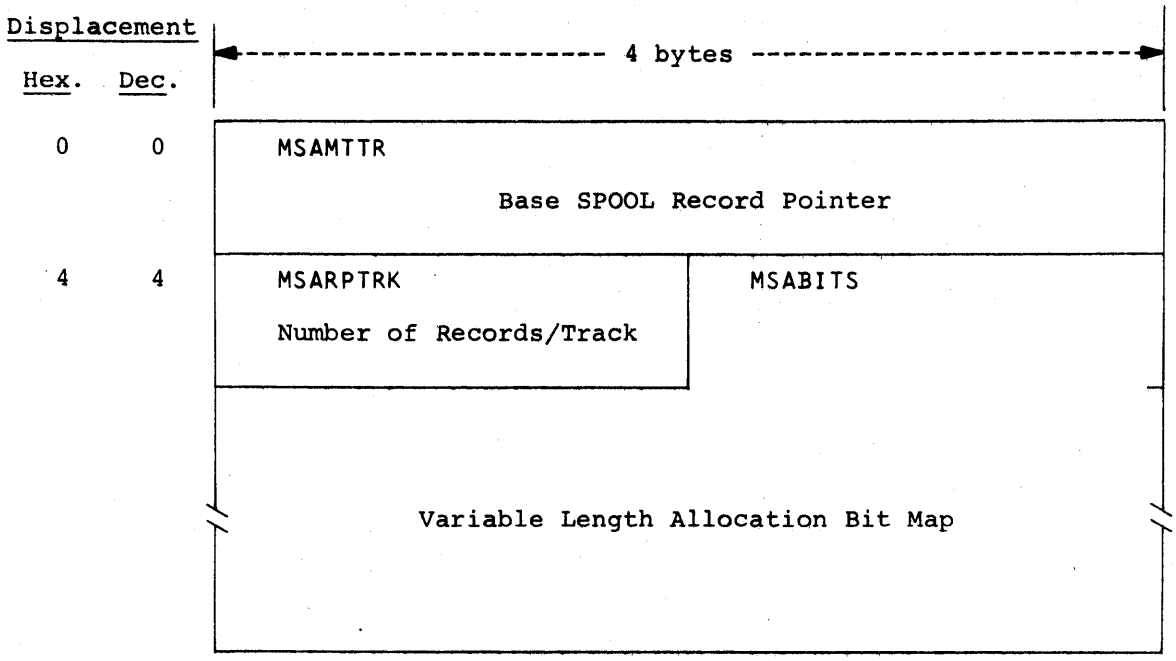
Figure 8.13.1 -- PARTITION INFORMATION TABLE FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
0	0	PITSTAT	1	Status Byte --

Bit	Name	Meaning
0	PITHOLDA	PIT is Drained ($P I).
1	PITHOLD1	PIT is Drained ($P In).
2	PITBUSY	Partition Busy Indicator.
3	PITIDLE	PIT Idle Message Switch.
4-6		Reserved for Future Use.
7	PITLAST	Last PIT Indicator.

1	1	PITICLAS	1	O/S Initiator Class.
2	2	PITPATID	2	Logical Partition Identification.
4	4	PITSIZE	2	Logical Partition Size (Unused).
6	6	PITPRIO	2	Logical Partition PRTY.
8	8	PITBECB	4	Batching Program Frozen ECB Chain.
C	12	PITBJST	4	Address of Batching Program TCB.
10	16	PITBCLAS	1	Active Batching Class.
11	17	PITBUNIT	3	Batching Program Input Unit.
14	20	PITBUCBA	2	Batching Input Unit Control Block (UCB) Address.
		PITCLASS		Variable Number of Logical Partition Classes.

Figure 8.14.1 -- MESSAGE ALLOCATION CONTROL BLOCK



Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
0	0	MSAMTTR	4	Base SPOOL Record Pointer.
4	4	MSAPTRK	2	Number of Records/Track.
6	6	MSABITS		Variable Length Allocation Bit Map.

Figure 8.15.1 -- DATA BLOCK FORMAT

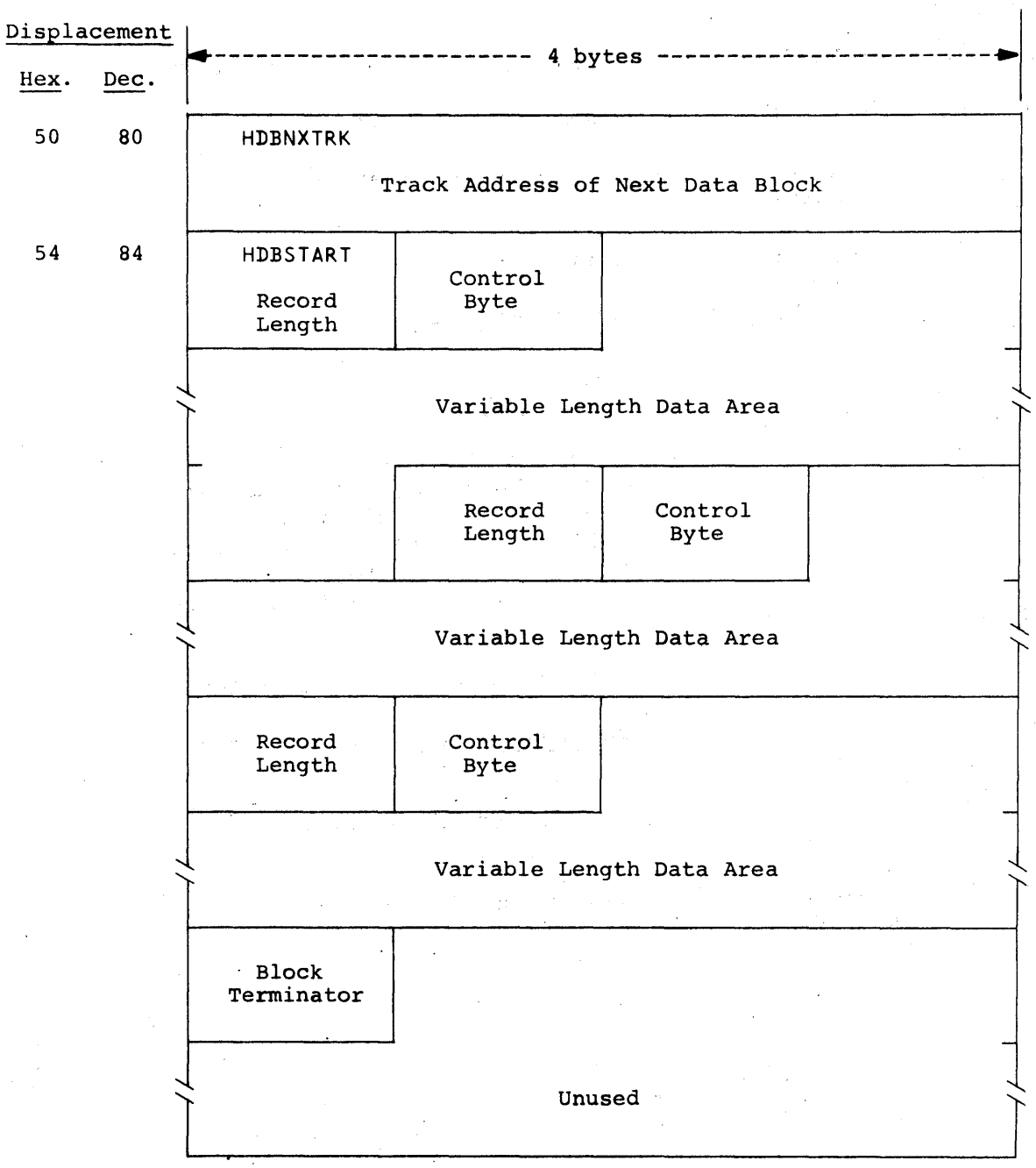


Figure 8.15.1 -- DATA BLOCK FORMAT (CONTINUED)

Displacement Hex.	Displacement Dec.	Field Name	Bytes	Field Description
50	80	HDBNXTRK	4	Track Address of Next Data Block.
54	84	HDBSTART		Start of Data Block.
		Record Length	1	Length of Data Area (0-254).
		Control Byte	1	Control Byte --

Input Data Sets

Hex. Value	Meaning
00	Normal Record.
03	Internally Generated Card.
04	HASP Control Card.
13	Illegal HASP Control Card.
19	Last JCL Card.
73	Dummy Track Address Record.

Print Data Sets -- Carriage Control.

Punch Data Sets -- Stacker Select.

		Data Area		Variable Length Data Area.
		Block Terminator		Record Length of 255 (X'FF').

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: namel-addr.
- By a three-part structure consisting of an operand name, a hyphen, and a coded value. For example: namel-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: KEYWDl=addr.
- By a three-part structure consisting of a keyword, an equal sign, and a coded value. For example: KEYWDl=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 pa. or a specific sequence of executable instructions.

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

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

  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:

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

  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".

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:

[symbol] $EXAMP {abc-addrx / (R1)} , {def-addrx / (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:

```
[symbol]   $QADD  {element-addrx}  ,  {queue-value}
                  {(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}]  [,WAIT={YES}]  [,CONVERT={YES}]
                        {NO }         {NO }          {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}]  [,WAIT={YES}]  [,CONVERT={YES}]
                        {NO }         {NO }          {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.

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]
                      {(Rl)        }
```

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
```

9.11.9 $FCB - Define 3211 Forms Control Buffer Load

The $FCB macro-instruction causes the creation of an overlay csect containing a forms control buffer load for the 3211. It may be used in CSECT HASPPRPU to create a new FCB load or, in a stand-alone assembly to be included later in HASP via the overlay-build process, to replace an existing FCB load.

Format Description:

```
FCBx   $FCB  inch,page,chan-line[,line...]
             [,chan-line][,line...]...
```

x
specifies the character by which the FCB load will be referenced by the operator, using the command $TPRTn,C=x. The character must be numeric or alphabetic and cannot be 1 or V.

inch
specifies lines per inch. It must be 6 or 8.

page
specifies lines per page. It must be 180 or less.

chan
specifies carriage channel number. It must be greater than 0, not greater than 12, and followed by a hyphen.

line
specifies line number at which the carriage channel punch is to appear. It must not be greater than lines per page.

(The remainder of this page intentionally left blank.)

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 2314 disk volumes. UNIT and SPACE parameters should be changed as appropriate if other direct-access devices are used. VOLUME parameter 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	75 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
	HRTPOPTS	RMTGEN Standard Option Lists
	HRTPSM20	M20 STR Remote Program
	HRTPSYS3	System/3 Remote Program
	HRTP1130	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=2314,VOLUME=SER=222222,DISP=OLD
//HASP      DD UNIT=2314,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=2314=222222,PURGE
   SCRATCH     VTOC,VOL=2314=HASP,PURGE
/*
//ALLOCAT EXEC PGM=IEHPROGM
//SYSIN     DD DUMMY
//SYSPRINT  DD DUMMY
//HASPSRC   DD DSNAME=SYS1.HASPSRC,UNIT=2314,VOLUME=SER=HASP,
//             DISP=(,CATLG),SPACE=(CYL,(35,5,5)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3360)
//HASPOBJ   DD DSNAME=SYS1.HASPOBJ,UNIT=2314,VOLUME=SER=HASP,
//             DISP=(,CATLG),SPACE=(CYL,(5,5,5)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)
//HASPMOD   DD DSNAME=SYS1.HASPMOD,UNIT=2314,VOLUME=SER=HASP,
//             DISP=(,CATLG),SPACE=(CYL,(5,5,5))
//UT1       DD DSNAME=SYS1.UT1,UNIT=2314,VOLUME=SER=222222,
//             DISP=(,CATLG),SPACE=(CYL,(20,5))
//UT2       DD DSNAME=SYS1.UT2,UNIT=2314,VOLUME=SER=HASP,
//             DISP=(,CATLG),SPACE=(CYL,(20,5))
//UT3       DD DSNAME=SYS1.UT3,UNIT=2314,VOLUME=SER=222222,
//             DISP=(,CATLG),SPACE=(CYL,(20,5))
```

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 the formats acceptable to the IEBUPDTE or IEBUPDAT OS utility programs. 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).

All IEBUPDTE and IEBUPDAT control cards are defined for use with the HASPGEN Update except the ./ ALIAS ... detail statement. The ./ NUMBER ... and ./ NUMBR ... detail statements will be accepted but will be ignored. Only the NAME, SEQ1, and SEQ2 keywords (and equivalent IEBUPDAT positional parameters) will be interpreted for meaning. Other keyword and positional parameters are ignored and may be omitted. IEBUPDTE and IEBUPDAT control cards may be intermixed as desired.

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 are inserted immediately following the last modification card which was in ascending collating sequence. Update cards which do not maintain an ascending collating sequence in columns 73-80 and are not blank will terminate the HASPGEN with an update error.

All PTFs (and user modifications, if any) which apply to one source module must be integrated into a single deck, beginning with a CHANGE (or 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 CHANGE (or 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)
./ CHANGE NAME=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

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 reallocate SYS1.HASPSRC only, then to use the HASPGEN proc and full parameter/update deck to recreate SYS1.HASPSRC. Only required assemblies are performed, using ASMHASP proc, with decks replacing those of same name in SYS1.HASPOBJ.

A module must be reassembled 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). If a changed parameter is used as a default value for another parameter, then that other parameter is changed also, and all modules depending on it must be reassembled. For example, if &NUMTPPR is allowed to default, then changing &NUMLNES will cause &NUMTPPR to change. A change in the update portion of the deck for a module, compared to the previous HASPGEN, also requires that the module be reassembled. If in any case reassembly 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 reassembled.

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 reassembled 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
&AUTORDR-WCN
&BSCCPU -PM
&BSC2770-M
&BSC2780-M
&BSC3780-M
&BSHPRES-M
&BSHTAB -M
&BSHPRSU-M
$BSPACE -W
&BSVBOPT-M
&BUFHICH-N
&BUFSIZE-HRXPMN
$CKPTIME-V
&CLS(n) -X
&CONAUTH-N
&DEBUG -HXVCNO
$DELAYCT-M
&DMPTAPE-N
$ESTIME -R
$ESTLNCT-R
$ESTPUN -R
&FCBV -PC
&INITSVC-XNS
&JITSIZE-HRXVCN
$LINECT -R
LINEmm -N
&LOGOPT -X
&MAXCLAS-XCN
&MAXJOBS-VN
&MAXPART-X
&MAXXEQS-HXVWCN
&MINBUF -N
&MLBFSIZ-M
&MONINTV-HXV
&NOPRCCW-HPC
&NOPUCCW-HPC
&NUMBUF -N
&NUMCONS-HXWCN
&NUMDA -HRXPAVWMCN
&NUMDDT -RX
&NUMINRS-HRXN
&NUMLNES-HRPWMN
&NUMOACE-N
&NUMPRTS-HPVCN
&NUMPUNS-HPVN
&NUMRDRS-HWN
&NUMRJE -MCN
&NUMTGV -HRXPAVWCN
&NUMTPBF-N
&NUMTPES-HN
&NUMTPPR-HPVN
&NUMTPPU-HPMN
&NUMTPRD-H
&NUMWTOQ-HN
&OLAYLEV-RXPAVMCN
&OREPSIZ-HN
&OSC(n) -X
&OSINOPT-R
&OUTPOPT-X
$OUTXS -X
&PID(n) -X
&PRI(n) -X
$PRICONA-H
$PRIDCT -P
&PRIHIGH-V
&PRILOW -V
&PRIRATE-HV
$PRTBOPT-P
&PRTRANS-PM
&PRTUCS -N
$PUNBOPT-P
&RDR -XN
&RDRPART-N
$REPRDR -N
$REPWTR -N
&RESCORE-N
&RJOBOPT-R
RMTnn -N
$RPRBOPT-P
&RPRI(n)-R
&RPRT(n)-R
&RPS -HNT
$RPUBOPT-P
&RQENUM -W
&SIZ2260-WN
&SPD2260-W
&SPOLMSG-PMN
&SPOOL -N
&STRCPU -M
&STR1978-M
&TIMEOPT-X
$TIMEXS -X
&TPBFSIZ-HMN
$TPIDCT -P
&TRACE -HXVWN
&USASCII-M
$WAITIME-M
&WCLSREQ-T
&WTLOPT -CN
&WTR -XWCN
&WTRCLAS-XNT
&WTRPART-WCN
&XBATCHC-RXWCN
&XBATCHN-RXWC
&XLIN(n)-RX
&XPRI(n)-X
&XZMFTH -V
&XZMFTL -V
&XZMULT -V
&XZPRTY -XV
$$x -X

10.1.6 Compatible HASP SPOOL Volumes

There are two levels of SPOOL volume compatiblity between two different HASP generations: the compatibility level which allows one HASP System to use another SPOOL volume without requiring complete reformatting of the SPOOL data sets; and the compatibility level which allows one HASP System to continue another system's job processing with job data contained on the SPOOL data set "warm start".

If a new release of HASP, or official HASP modification requires a reformat or prohibits a "warm start" the release documentation will inform the installation of this fact and will therefore not be discussed in this section. However, certain HASPGEN parameters when altered between HASP generations of the same release may also produce incompatible SPOOL volumes. The HASPGEN parameters that (when altered) require reformatting of the SPOOL volumes between systems are as follows:

&BUFSIZE

The HASPGEN Parameters that (when altered) prohibit "warm start" between systems are as follows:

&NUMDA	&NUMPRTS
&NUMTGV	&NUMTPPR
&MAXJOBS	&SPOLMSG
&JITSIZE	

Many of these parameters are assigned values automatically by HASPGEN based on the assignment of other parameters and must be overriden to provide compatibility. For example: &SPOLMSG depends upon the number of remote terminals capable of communicating with HASP (&NUMRJE) which in turn depends upon the number of lines (&NUMLNES) unless overriden by a user specification.

(The remainder of this page intentionally left blank.)

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 per job
PRDD     = maximum number of print data sets per step
PUDD     = maximum number of punch data sets per step
SFPRDD   = maximum number of print data sets requiring special
           forms or special routing per job
SFPUDD   = maximum number of punch data sets requiring special
           forms or special routing per job
&MAXXEQS = maximum number of simultaneous Job executions
&NUMINRS = number of Internal Reader interfaces
```

If a job requires more pseudo devices than the total number generated, it may be deleted from the system (with an appropriate message) or some of its special forms requests may be ignored.

It should be noted that the term "Pseudo Device" implies a physically non-existent device. An address chosen for a pseudo device may be any device address acceptable to OS and it must not match the address of any existent device or other pseudo device.

10.2.1.2 Additional Symbolic Unit Names

The symbolic unit name "A" should be assigned to all pseudo 1403 printers, except the one identified by the HASPGEN parameter &WTR. 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=50000801

Pseudo 1403 Printer (except &WTR)
     IODEVICE  UNIT=DUMMY,ADDRESS=xxx,DEVTYPE=50000808
     UNITNAME  NAME=A,UNIT=xxx

Pseudo 1403 Printer (for &WTR)
     IODEVICE  UNIT=DUMMY,ADDRESS=xxx,DEVTYPE=10000808

Pseudo 2540 Punch
     IODEVICE  UNIT=DUMMY,ADDRESS=xxx,DEVTYPE=50000802
     UNITNAME  NAME=B,UNIT=xxx

Pseudo 1443 Printer
     IODEVICE  UNIT=DUMMY,ADDRESS=xxx,DEVTYPE=5000080A

Pseudo 1442 Punch
     IODEVICE  UNIT=DUMMY,ADDRESS=xxx,DEVTYPE=50000803

Pseudo 2520 Punch
     IODEVICE  UNIT=DUMMY,ADDRESS=xxx,DEVTYPE=50000805
```

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 and 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 Position for the HASP Type I SVC

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 10.2.2.3.

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

Any MFT system used with HASP must have a four byte SVC table and resident SVC capability. This is specified by including "TRSVCTBL" in the OPTIONS parameter and "TRSVC" in the RESIDNT parameter of the SUPRVSOR macro. It is not necessary to make any type 3 or 4 SVCs actually resident. This may be prevented, if desired, by appropriate changes to SYS1.PARMLIB and/or operator action during OS IPL.

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. It is not necessary or beneficial for HASP's purposes to make ATTACH resident, which would require additional fixed storage.

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, 6 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

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

This process is not necessary if the SVC was installed during OS SYSGEN as described previously under Section 10.2.1.4. 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 first three INSERT cards should be used, as indicated by comments on the cards. The fourth INSERT card is required if the OS Nucleus contains the 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 20.7 with proclib blocked 400.

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 10.2.2.3.

The DD card 01040000 should refer to the cataloged HASP overlay data set produced during the build step described under 10.2.2.3. 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 52K to 18K. However, the ASB Interpreter will dynamically acquire a region (76K 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 must not 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 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 and less than 128 tracks. The example shows space for 60 records of 1024 bytes, a comfortable quantity for an unmodified HASP System with HASPGEN parameter &OLAYLEV set for maximum overlay. The overlay data set may be moved, after being originally written by HASPOBLD, but only to another volume of the same device type.

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 givn 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 length and 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.

Completion code 0 is returned normally, 4 if duplicate CSECTs were encountered, and 8 if any overlays were too long or undefined. Sample job HASPHASP prevents the link edit step if a completion code greater than 4 is returned.

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. (It is assumed that HASPGEN parameter &SPOOL has been left at the value '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 mutiples 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 1019. HASP would use tracks 18 and 19 for checkpoint information; it would use the 250 track groups beginning with track group 5 (which starts on track 20 and extends through track 23) and ending with track group 254 (which starts on track 1016 and extends through track 1019. It would make the other 250 track groups on this 2311 as permanently unavailable for HASP SPOOL data sets.

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 Genereation 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. The resultant cards are:

   - For 1130, the RTPLOAD or RTP1130 deck.
   - For System/3 without 5424, a complete load deck.
   - For System/3 with 5424, a deck to be further processed as described in Section 10.3.6.

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

All listings produced by RMTGEN and the assembler 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)

CARD FORMAT:

Col. 1-2	./	control identification
Col. 3-n	blank	0 to 44 blanks
Col. (n+1)-(n+6)	DELETE	verb for delete source cards indicated
Col. (n+7)-m	blank	1 to (45-number of previous blanks)
Col. (m+1)-(m+13)	SEQ1=serial 1	starting card serial number
Col. (m+14)	,	
Col. (m+15)-(m+27)	SEQ2=serial 2	ending card serial number

Update control cards may be used only during an update run i.e. after $.UPDATE card. The DELET/DELETE 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/DELETE 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 allow 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/DELETE 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

10.3.7 RMTGEN Completion Codes

RMTGEN determines the highest completion codes returned by any of the HASP supplied generation modules or the assembler used for object deck creation and returns that code to the system. HASP supplied RMTGEN modules detect and return completion codes as follows:

1) ABEND 20 - RMTGEN - A module read to end of data on the CARDIN data set without setting RMTGEN module's EODAD exit. The load modules on the Job or Step library are not correct and should be restored.

2) Completion Code 24 - EXRMTGEN - The operator replied cancel to the request to place cards in the CARDIN data set or the generation of remotes was surpressed based upon HASPGEN parameters (no HASP workstation decks are generated).

3) Completion Code 24 - GENRMT - An error was detected by GENRMT module and a message was displayed on the SYSPRINT data set as follows:

A. ****INVALID SELECTION CARD****

1. The program identification card named an unsupported remote.
2. The format of the identification card was incorrect.
3. The numeric field was not numeric.

The card in error is displayed preceding the error message and the generation of the requested remote is suppressed.

B. ****OPTION SPECIFICATION ERROR****

1. The specified RMTGEN paramter was misspelled.
2. The format of the card is incorrect.
3. Card sequence numbers are not in ascending order.
4. Invalid ./ card.
5. A $. card other than $.RMTEND was encountered during the update process.
6. A /* card was encountered within a remote description deck.

The card in error is displayed preceding the error message and the generation of the requested remote is suppressed.

C. ****UNEXPECTED END OF CARD INPUT****

1. The last card of the CARDIN data set was not $.RMTEND or in case of updating operations "./ ENDUP".

The generation of the requested remote is suppressed.

D. ****HASP SOURCE LIBRARY ERROR****

1. An internal control card on the GENPDS data set member HRTPOPTS is incorrect or missing.
2. An overflow of the GENRMT standard options table has occurred.

The generation of the requested remote is suppressed. The user should check the spelling on his selection card and if spelling is correct recreate the HASP source library using standard HASP generation procedures.

4) Completion Code 24 - LETRRIP - An error was detected by the 1130 post processor and appropriate message displayed as follows:

A. ***REMOTE id DECK INCOMPLETE***

B. *** REMOTE id EXCEEDS AVAILABLE 1130 STORAGE***

5) Completion Code 12 - SYS3CNVT - Unexpected or missing end-of-file indication on the input dataset. The message displayed on the operator console and in the HASP System Log is:

S3CNVT - UNEXPECTED OR MISSING END-OF-DATA

6) Completion Code 16 - SYS3CNVT - Unable to open one or both data sets. The message displayed on the operator console and in the HASP System Log is:

S3CNVT - UNABLE TO OPEN ONE OR BOTH DATASETS

(The remainder of this page intentionally left blank.)

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

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.

Normally an operator must correct command input via printer keyboard devices by cancelling the entry and re-entering the entire corrected command. During HASP job processing HASP provides backspace editing of all commands (HASP and OS) entered via OS controlled consoles. Since most printer keyboards do not have the backspace key the facility is simulated via a "substitute" backspace character, defined here for the purposes of illustration as the "not" character "¬". The installation selects the actual "substitute" backspace at HASP generation time. Rules concerning the "¬" character follows:

1. Although the entry of "¬" does not physically move the printer position backward, the command text character preceding the "¬" and the "¬" character itself are removed from the internal image of the command.

2. The backspace edit is unconditional; therefore, "¬" cannot be entered for the purpose of contributing to the text of the message regardless of HASP or OS command entry specifications.

3. Multiple "¬" entries causes a logical backspace for each "¬" entered. (Backspacing beyond the beginning of the line is prevented.)

The following examples illustrate the above rules:

Entry	Result
$DN¬Q	$DQ
$DJ4¬¬¬CJ4	$CJ4
$D'ABE¬C'	$D'ABC'
$C'¬¬¬¬C ABC	C ABC

(The remainder of this page intentionally left blank.)

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, OR JOBS IN EXECUTION
$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, FCB IMAGE, 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 after 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)
$E job list	NO	Restart execution of 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		
$A 'job name'	IF OWNER	Release job by OS job name
$C 'job name'	IF OWNER	Cancel job by OS job name
$D 'job name'	OK	Display job information on job(s)
$E 'job name'	NO	Restart execution of job by OS job name
$H 'job name'	IF OWNER	Hold job by OS job name
$P 'job name'	IF OWNER	Stop job by 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)

Table 1.1.3 HASP COMMAND SUMMARY (continued)

COMMAND	REMOTE SOURCE	COMMENTS
DEVICE LIST (continued)		
$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
$TF	NO	Set FCB image for 3211 carriage control C=V
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:

```
JOB j [name] AWAITING{EXEC class } PRIO priority [HOLD     ]
                     {PURGE       }               [PURGE    ]
                     {PRINT r     }               [DUPLICATE]
                     {PUNCH r     }
```

2. Jobs being processed (active):

```
JOB j [name] {EXECUTING class } PRIO priority [HOLD]
             {IS PURGING      }               [PURGE]
             {ON device name  }
```

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 preceding the INVALID OPERAND.

1.4 JOB QUEUE COMMANDS

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.

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
```

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 all 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: $E job list RESTART EXECUTION OF SPECIFIED JOB(S)

Action: Each specified job found in the system and currently in OS execution will have its HASP execution controller flagged to restart the job. Upon completion of OS execution (normal or abnormal) the controller will place the job back on the HASP execution queue.

Note: This command requires job and system command authority and is not available to HASP remote operators.

Response:

Job information response	- one response for each specified job in execution which has been flagged for re-execution
JOBj NOT RESTARTABLE	- one response for each specified job found in the system which cannot currently be restarted
JOB(S) NOT FOUND	- none of the specified jobs were found

Examples:

1. user - $E J6
   system - JOB 6 ANY EXECUTING A PRIO 8

2. user - $E J3
   system - JOB 3 MYJOB EXECUTING A PRIO 6
   user - C MYJOB
   OS - accepted message

Comments: In example 1 the operator desires to let job 6 run to completion before requeueing for execution.

In example 2 the operator desires to abort current execution of job 3 and requeue for execution.

(The remainder of this page intentionally left blank.)

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: $A'jobname' RELEASE JOB SPECIFIED BY OS JOBNAME

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

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

Action: The HASP job queue is searched for the single job with the specified job name and the action of the $A job list command is performed as though that command had been entered for the job. If the job is not found or if more than one job with the specified name is encountered, no action is taken and an appropriate diagnostic is displayed.

Response:

JOBj RELEASED	- the job specified has been released
jobname NOT FOUND	- the job named is not in the system
JOBj NOT HELD	- the specified job was not in hold status
MULTIPLE JOBS WITH jobname	- more than one job with the specified name is in the system
	- messages compatible with $D 'jobname' will follow

Examples:

1. user - $A'MYJOB'
   system - JOB 4 RELEASED

Definition: $C'jobname' CANCEL JOB SPECIFIED BY OS JOBNAME

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

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

Action: The HASP job queue is searched for the single job with the specified job name and the action of the $C job list command is performed as though that command had been entered for that job. If the job is not found or if more than one job with the specified name is encountered, no action is taken and an appropriate diagnostic is displayed.

Responses:

Job information response	- response listing the current status of the job after command action
jobname NOT FOUND	- the job named is not in the system
MULTIPLE JOBS WITH jobname	- more than one job with the specified name is in the system
	- messages compatible with $D'jobname' will follow

Example:

1. user - $C'YOURJOB'
   system - JOB 82 YOURJOB AWAITING PRINT 0

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

Where: 'jobname' = the OS job name appearing on the user's job card enclosed by apostrophes. The name may be upper or lower case alphameric 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 job named is not in the system

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: $E'jobname' RESTART EXECUTION OF JOB SPECIFIED BY OS JOBNAME

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

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

Action: The HASP job queue is searched for the single job with the specified job name and the action of the $E job list command is performed as though that command had been entered for that job. If the job is not found or if more than one job with the specified name is encountered, no action is taken and an appropriate diagnostic is displayed.

Note: This command requires job and system command authority and is not available to HASP remote operators.

Responses:

Job information response	- response listing the current status of the job (the job's execution controller has been flagged to restart the job)
JOBj NOT RESTARTABLE	- the specified job is not currently in execution
jobname NOT FOUND	- the job named is not in the system
MULTIPLE JOBS WITH jobname	- more than one job with the specified name is in the system
	- messages compatible with $D'jobname' will follow

Examples:

```
1.   user   - $E 'ANY'
     system - JOB 6 ANY EXECUTING A PRIO 8
2.   user   - $E 'MYJOB'
     system - JOB 3 MYJOB EXECUTING A PRIO 6
     user   - C MYJOB
     OS     - accepted message
```

Comments: In example 1 the operator desires to let job "ANY" run to completion before requeueing for execution.

In example 2 the operator desires to abort current execution of job "MYJOB" and requeue for execution.

Definition: $H'jobname' HOLD JOB SPECIFIED BY OS JOBNAME

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

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

Action: The HASP job queue is searched for the single job with the specified job name and the action of the $H job list command is performed as though that command had been entered for the job. If the job is not found or if more than one job with the specified name is encountered, no action is taken and an appropriate diagnostic is displayed.

Responses:

Job information response	- response listing the current status of the job after command action
jobname NOT FOUND	- the job named is not in the system
MULTIPLE JOBS WITH jobname	- more than one job with the specified name is in the system
	- messages compatible with $D'jobname' will follow

Example:

1. user - $H 'ANYJOB'
   system - JOB 302 ANYJOB AWAITING EXEC A PRIO 4 HOLD

Definition: $P'jobname' STOP JOB SPECIFIED BY OS JOBNAME

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

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

Action: The HASP job queue is searched for the single job with the specified job name and the action of the $P job list command is performed as though that command had been entered for the job. If the job is not found or if more than one job with the specified name is encountered, no action is taken and an appropriate diagnostic is displayed.

Responses:

Job information response	- the job specified has been stopped
jobname NOT FOUND	- the job named is not in the system
MULTIPLE JOBS WITH jobname	- more than one job with the specified name is in the system
	- messages compatible with $D'jobname' will follow

Example:

1. user - $P 'UNIQUE'
   system - JOB 31 UNIQUE ON PRINTER 2 PRIO 6 PURGE

(The remainder of this page intentionally left blank.)

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.

If a job's class is execution batching (one of the classes specified by HASPGEN parameter &XBATCHC) it should not be changed to a non-batching class. Similarly, a non-batching class should not be changed to a batching class. Such actions will cause the job to execute incorrectly.

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
PRINTRn  - PRTn (used when more than 9 printers are on the
                 system)
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
LINE			Y				Y[4]	Y	Y[7]	Y[7]
LOCAL READER		Y[2]					Y	Y	Y	Y
INPUT TAPE		Y[2]					Y	Y	Y	Y
REMOTE READER		Y[2]					Y[5]	Y	Y	Y
INTERNAL READER		Y[2,3]					Y[6]	Y	Y	Y[7]
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[1]	Y	Y	Y[1]	Y[1]	Y	Y	Y	Y	Y
REMOTE PUNCH	Y[1]	Y	Y	Y[1]	Y[1]	Y	Y	Y	Y	Y
LOCAL CONSOLE								Y	Y	Y

[1]RESTARTS FUNCTION

[2]DELETES CURRENT JOB

[3]SIMULATES EOF IF NO CURRENT EXCP

[4]DRAIN AFTER SIGNOFF OR $E IF REMOTE ACTIVE

[5]DRAIN AT EOF AS OPPOSED TO END OF FUNCTION FOR CURRENT JOB

[6]AUTOMATICALLY STARTED BY USER PROGRAM

[7]COMMAND ACCEPTED BUT WILL HAVE NO EFFECT

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.

Note: A $B directed to a punch will have the effect of restarting the current function.

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 current job will be queued for print and reading will continue. In case of internal reader devices an end of file may be simulated based upon the instantaneous status of the device.

Note: A $C directed to an internal reader should only be used when the reader has been left active by a submitting task that has terminated (ie., the job that submitted the input stream terminated without freeing the internal reader).

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.

Note: A $F directed to a punch device will have the effect of restarting the current function.

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.

Note: A $I directed to a punch device will have the effect of restarting the current function.

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 or interrupt the device will be ignored until the device has completed operation on the current copy of the 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 only from the CPU to be varied offline are in the HASP DRAINED status.

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

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.

Note: The unit address is used for the purpose of assigning a physical device to the HASP logical device. Once this assignment has been made succeeding $S commands directed to the device should be made omitting the unit specification and associated comma.

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

2. user - $S TAPE1,TAPE2
   system - OK

Comments: In example 2 the operator has previously set tapes 1 and 2 to place all jobs in the hold status and now wishes to reverse that action.

Definition: $S console START DEVICE(S)

Where: device = HASP console device(s)

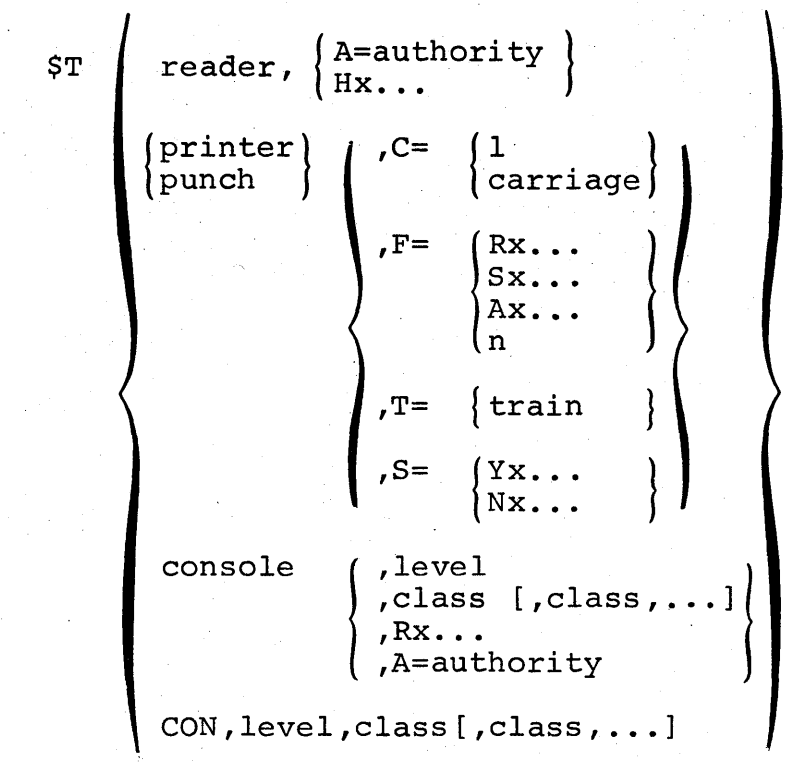
Action: The specified HASP consoles will be started for all logical console classes of messages with all levels of importance (see CONSOLE SUPPORT for classes and levels of messages).

Responses: OK - the HASP console(s) have been started

Examples:

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

Definition:



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

Definition: $T reader, {Hx... / A=authority} SET DEVICE

Where: reader = HASP reader device

Action: If Hx... is specified the designated 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,H" command causing the reader to revert to normal reading and queueing of jobs.

If A=authority is specified the HASP command authority of the designated local, tape, or internal reader is set to allow HASP command entry as follows:

0 - display only
1 - system control
2 - device control
4 - job control

Notes:

1. A reader may not be used to set the command authority of a reader device.
2. See CONSOLE SUPPORT section of this manual.
3. See $TCON and $TCONn commands.
4. The $S command does not negate the effects of the "$T reader, A=authority" command.
5. A=authority operand requires device and system authority.
6. If a specification is in error, that specification and all succeeding specifications in the command are rejected. Previous specifications will be acted upon.

Responses: OK - The specified reader has been set to HOLD subsequent jobs and/or its HASP command authority has been set.

Examples:

1. user - $T RDR1,HOLD
   system - OK
2. user - $T RDR1,H
   system - OK
3. user - $T RDR1,A=0
   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=R 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.

4. The specification C=V should be used in conjunction with the $TF command discussed in the SYSTEM COMMANDS section of this manual.

Limitations:

1. The setting of forms, train/chain, or carriage 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. When HASP has issued a forms load and is waiting for the operator to enter $S for the device, the device may be considered inactive for the purpose of setting train/chain or carriage for local printers.

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=R,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 data set.

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 | ,class[,class,...] | ,Rx... | ,A=authority } SET DEVICE

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 CON1,0,MAIN
system - OK
```

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

Comments: In example 3 all output to console 1 is turned off (console level set to 15 and class set for no logical classes). Then the console is set to display all messages for the MAIN console.

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. If an Execution Batch Processing Program is in main storage under control of a logical initiator, its OS jobname will also be displayed.

Responses: INIT n ({ DRAINING / DRAINED / ACTIVE / INACTIVE }) = classes - one response for each initiator requested

Examples:

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

2.
```
user   - $DI
system - INIT 1 (ACTIVE)=ABCD
         INIT 2 (DRAINING)=BCDA
         INIT 3 (INACTIVE)=WDAB     $$$$$W3
         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 not 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: The $P command will be simulated and, if HASP is in a dormant status (no job processing is in process, and all HASP devices are DRAINED or INACTIVE), the HASP SYSTEM will withdraw from control of the Operating System.

Note: 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 indication 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

Definition: $TF...[6|8][,L=n][,c=list]

SET FCB IMAGE FOR 3211 CARRIAGE CONTROL C=V

Where:

6 = Indicator for six lines/inch (default if six or eight omitted)

8 = Indicator for eight lines/inch

L=n = Number of lines per page (may be specified only once per command)
Acceptable numeric values for n range from 2 to 180 (default settings L=66 for 6 lines/inch and L=88 for 8 lines/inch)

c=list = Specifications for one or more printer carriage control channels. The "c" specification may take values 1 through 12 representing the channel number. List items identify the line(s) to which a corresponding printer skip command is to position the page. List items are separated by commas. No two list items within the command may designate the same line number. Acceptable values for list items range from 2 to 180.

Notes:

1) The normal operand limit specified for HASP commands does not apply with this command.
2) List items are not inspected to insure specified lines are within the L=n specification or the default if L is not specified.
3) This command is defined only when selected by the installation at HASPGEN time by &FCBV parameter.

Action:

The FCB image used for setting carriage control via $T printer,C=V is reset and created in accordance with the operands of the command. If the last character of the first operand is not "8", the image is set for six lines/inch. If the L=n operand is omitted, a default of 66 lines/page is assumed for six lines/inch forms and 88 lines/page is assumed for eight lines/inch forms.

For each channel "c=list" specified, the carriage channel is set so that a "skip to channel" command addressed to the printer will cause the page to be

positioned at the line indicated in the list items (see 3211 component description manuals for hardware details). If invalid characters, invalid lines or channel values, or mutually exclusive parameters are specified, the image is reset and six or eight lines per inch null parameters are assumed. Carriage channel 1 is always set for line 1 representing the first print line on the page. This setting must not appear as a list item for channel 1.

Starting at the bottom of the page for as many lines as possible, channels which are omitted from the specifications are assigned automatically.

Responses: OK - the settings requested have been made.

Examples:

1. user - $T FCB
   system - OK
2. user - $T FCB8,L=40,2=20,4=10,30
   system - OK

Comments: In example 1, the user desires the default six lines/inch carriage control image of channel 1 at line 1 with 66 lines/page.

In example 2, the user desires eight lines/inch with 40 lines/page. Channel 2 is to allow skipping to line 20 while channel 4 is to allow skipping to line 10 and 30 on the page.

Notes:

1. This command requires console authority for both device and system commands.
2. The null FCB image settings are as follows:

6 lines/inch		8 lines/inch	
line	channel	line	channel
1	1	1	1
:	:	:	:
56	2	78	2
57	3	79	3
58	4	80	4
59	5	81	5
60	6	82	6
61	7	83	7
62	8	84	8
63	10	85	10
64	11	86	11
65	9	87	9
66	12	88	12

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
```

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.

Note: If there are no HASP remote devices $DR command will be assumed.

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 for job j is to be routed
- = LOCAL--the designated output for all jobs currently in the system and routed for LOCAL devices is to be routed
- = device--the designated output for all jobs currently in the system and routed for this device is to be routed
- = RMx...r--the designated output for all jobs currently in the system and routed for remote r is to be routed

to-id
- = LOCAL--job(s) are to be routed to local devices
- = device--job(s) are to be routed to this device
- = RMx...r--job(s) are to be routed to 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

HASP the ready condition. (If HASP has been generated with 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)
S  HASP.S   - MFT start command if HASP partition is not large
              enough to contain the MFT RDR.
```

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 - SPOOL1 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.
NOFMT	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.
WARM	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.
NOREP	REP	No replacement cards are to be used.
REQ	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.
LIST	NOLIST	HASP is to list on a designated printer any replacement cards read.
NOLIST	LIST	HASP is not to list replacement cards.
TRACE	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 reallocated.

System action: After attempting to verify the remaining required SPOOL volumes, the HASP job terminates.

HASP MFT 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. 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.

!OUNT 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 SPOOL. 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 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.

O01

Explanation: 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.

System action: Continuous Loop.

Operator response: Take STAND-ALONE DUMP and notify system programmer.

System programmer response: Probable user error. Check any local modifications to HASP for the errors described above. Consult IBM Customer Engineer if problem remains undetermined.

V01

Explanation: The Purge Processor has discovered that some track groups to be freed were already free.

System action: Continuous Loop.

Operator response: Take a STAND-ALONE DUMP and notify system programmer.

X03

Explanation: The Execution Processor routine XTERMIN8 which deallocates DDBs discovered a non-existent UCB entry in the DDB being deallocated.

System action: Continuous Loop.

Operator response: Take a STAND-ALONE DUMP and notify system programmer.

X04

Explanation: 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.

System action: Continuous Loop.

Operator response: 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.

{device / JOB j} 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 Responses: The console log should be saved for maintenance personnel (even if recovery is successful). Additional responses depend upon the nature of the problem.

Specific Messages and Explanations: For STR terminals, the "IBM 2701 Data Adapter Unit Component Description" (GA22-6864) should be consulted for a complete discussion of status and sense bit meanings. For BSC terminals that employ the USASCII transmission code, the following substitutions should be made in response fields:

Response	EBCDIC	USASCII
EOT	37	04
NAK	3D	15
ACK1	61	31
ACK0	70	30

In the following messages, conventions observed are as follows:

UPPER CASE LETTERS AND NUMBERS--indicate fields which will appear on the error log exactly as described.

LOWER CASE LETTERS--indicate fields which have not been described in any previous example exactly the way that they appear in the error log.

ASTERISKS (**)--indicate fields which should be ignored as they do not contribute to the meaning of the message.

I/O ERROR ON LINEn uuu,02,0000,00rr,{94 / B4}ee

Explanation: A block sequence error has been detected by HASP while communicating with a MULTI-LEAVING terminal. This indicates that one or more transmission blocks have been lost. "rr" indicates the count received and "ee" indicates the count expected.

System Action: Any job reading in will be deleted and must be resubmitted.

Operator Action: This represents a very serious error. Control records may have been lost which could cause partial loss of terminal function. The line should be drained ($P) and restarted ($S) as soon as practical.

I/O ERROR ON LINEn uuu,02,0C00,0001,{84 / A6}**

Explanation: A 2770 or 2780 terminal has disconnected without signing off and a MULTI-LEAVING terminal has subsequently connected to the same line and is attempting to sign on.

System Action: The previous terminal will be signed off and the line disconnected.

Operator Action: The remote terminal operator must re-dial and attempt to sign on again.

I/O ERROR ON LINEn uuu,02,0C00,003D,{94 / A5 / A6 / B4}**

Explanation: A NAK has been received from the remote terminal indicating an error was detected at the terminal.

System Action: Normal error recovery procedures are invoked.

I/O ERROR ON LINEn uuu,02,0C00,0061,{A5 / A6}70

I/O ERROR ON LINEn uuu,02,0C00,0070,{A5 / A6}61

Explanation: HASP has received an incorrect acknowledgement from a 2770 or 2780 terminal. This may indicate that an output device (printer or punch) at the remote terminal has become not ready. It may also indicate that an output block has been lost.

System Action: The last block is retransmitted.

Operator Action: The remote terminal operator should check (to any extent possible) for missing or duplicate output and request a backspace or restart if the output looks questionable.

I/O ERROR ON LINEn uuu,02,0C00,00rr,84**

Explanation: Invalid data has been received from a 2770 or 2780. "rr" indicates the first significant byte received.

System Action: Normal error recovery procedures are invoked.

I/O ERROR ON LINEn uuu,02,0C00,00rr,****

Explanation: An invalid response has been received from the remote terminal. "rr" indicates the first significant byte of the response.

System Action: Normal error recovery procedures are invoked.

I/O ERROR ON LINEn uuu,02,0C00,ii {01 / 02} , {94 / B4} **

Explanation: An invalid termination character was received from a MULTI-LEAVING terminal. "ii" indicates the termination character received.

System Action: Normal error recovery procedures are invoked.

I/O ERROR ON LINEn uuu,02,0D00,0037,84**

Explanation: The card reader on a 2770 or 2780 has become not ready. This may be caused by a card feed error or by the failure of the remote operator to activate the END-OF-FILE switch or button.

System Action: The system waits for the reader to be made ready and transmission to continue.

Operator Action: The remote terminal operator should correct the problem and ready the card reader (insuring that the END-OF-FILE switch or button is activated).

I/O ERROR ON LINEn uuu,02,0D00,0037, {94 / B4 / C6} **

Explanation: HASP has received an unexpected EOT.

System Action: Normal error recovery procedures are invoked.

I/O ERROR ON LINEn uuu,**,0E00,ii**,****

Explanation: A Unit Check has been detected by HASP on the Communications Adapter. For more detailed information concerning the exact nature of the error, the "IBM 2701 Data Adapter Unit Component Description" (GA22-6864) and/or the "System/360 Component Description--2703 Transmission Control" (GA27-2703) publications should be consulted. Table 4.3.1 gives an explanation of the sense bits.

I/O ERROR ON LINEn uuu,02,FFFF,00rr,{94 / B4}ee

Explanation: A block sequence error has been detected by a MULTI-LEAVING terminal. This indicates that one or more transmission blocks have been lost. "rr" indicates the count that was received at the remote terminal and "ee" indicates the count that was expected.

System Action: Any job printing on the terminal will be interrupted, and any job punching on the terminal will be restarted.

Operator Action: This represents a very serious error. Control records may have been lost which could cause partial loss of terminal function. The line should be drained ($P) and restarted ($S) as soon as practical.

I/O ERROR ON LINEn uuu,cc,ssss,ii**,****

Explanation: An unusual channel end condition has been detected by HASP on the Communications Adapter. For more detailed information concerning the exact nature of the error, the appropriate hardware component description manuals should be consulted.

System Action: The line is automatically restarted.

Table 4.3.1 HASP RJE Typical Sense Information on 2701

ii	Meaning
80	Command Reject--"Abortive Disconnect" option of the 2701/2703 has been selected and the remote terminal has disconnected without signing off.
40	Intervention Required--Remote terminal has disconnected without signing off and abortive disconnect is not selected.
20	Bus Out Check--Hardware error.
10	Equipment Check--Hardware error.
08	Data Check--Line error or hardware error.
04	Overrun--Hardware error or deficiency.
02	Lost Data--Synchronization error.
01	Timeout--Expected terminal response not received by HASP.

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

{INIT / PART} 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 - {INIT / PART} 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 10.2 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 xxxxxx

Explanation: The indicated job has exceeded its estimated number of lines/cards by xxxxxx 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 cancelld (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 {PRINTING / PUNCHING} 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.

UNREADABLE OVERLAY--REBUILD OLAYLIB AND WARM START

Explanation: The HASP Overlay Service Routines have received a permanent error indication while attempting to read from the data set whose ddname is OLAYLIB, after all standard OS direct-access error recovery actions have been performed.

System Action: The HASP Processor which requested the overlay module is placed on a permanent $WAIT state. Other processors continue to function.

Operator Response: Notify the systems programmer. Enter $P to prevent new functions from starting.

System Programmer Response: Re-install HASP as described in Section 10.2.2.3. Ask the operator to warm start HASP to continue processing.

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 / JOB j} message	5
JOB j HELD FOR THE FOLLOWING VOLUMES--	5

"MAIN" CONSOLE MESSAGES

ALL AVAILABLE FUNCTIONS COMPLETE	7
JOB j BUFFER ROLL UNSUCCESSFUL--VERIFY NUMBER OF BUFFERS	7
JOB j JCT OVERFLOW - OUTPUT LOST	7
JOB j TERMINATED	7
SPOOL VOLUMES ARE FULL	7
{device / JOB j} message	5
{INIT / PART} i IDLE - CLASS = classes	5
JOB j DUPLICATE JOB NAME - JOB DELAYED	5
JOB j ESTIMATED {LINES / CARDS} EXCEEDED BY xxxxxx	5
JOB j ESTIMATED TIME EXCEEDED BY xx MINUTES	5
JOB j HELD FOR THE FOLLOWING VOLUMES--	5
JOB j -- jobname -- BEGINNING EXEC - {INIT / PART} i - class c	3
JOB j AWAITING HASP ALLOCATION	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

"REMOTE" CONSOLE MESSAGES

JOB j ON device--jobname programmer name
device SKIPPING FOR JOB CARD
JOB j LOAD 'xxxx' FORMS IN device
r, message from operator (at remote r)

"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).

Because HASP local readers (card, tape, and internal) may be used for command entry, these devices have an associated command authority with the default setting of 7 (full authority).

CHANGING CONSOLE AUTHORIZATION

An operator via a HASP local console or reader device authorized for system control may alter the authority of any other 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.

CHANGING READER AUTHORIZATION

An operator via an OS console or HASP local console with System and Device authority may change the command authority of any local HASP reader in the system via the "$T reader device, A=value" command. The operator must not use an authorized reader to turn off the System authority of a HASP console while that console has sufficient commands pending which will turn off the System authority of all HASP readers and other consoles on the system. This sequence of events will result in loss of operational control of the system.

(The remainder of this page intentionally left blank.)

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 should not 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
5 - 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
$T reader,A=a	-	Set the command authority for the local reader device
$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 initialziation 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.

To prevent the problem program from entering HASP commands via an internal reader, the operator may restrict the command authority of the device by entering:

```
$T internal reader,A=0
```

In order to prevent OS commands from taking effect the system programmer must have properly restricted the OS reader procedure used by HASP to pass jobs to OS.

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) This count applies only to data sets printed on job forms and demand forms. Only one copy of data sets indicated as specially routed data sets will be produced.

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
          16 - 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 workstation 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 workstation. Commands entered from these workstations can only be entered when the printer and punch devices are not ACTIVE. This is true even when the non-MULTI-LEAVING BSC workstation 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.

The operator should align printer forms setting printer FCB images, non-standard UCB images, and non-standard forms via the $T command prior to allowing HASP to begin job processing on printer devices.

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.

SETTING THE 3211 PRINTER FCB IMAGES - The 3211 printer carriage control tape--Forms Control Block (FCB)--is internally set by HASP. The installation system programmer creates several FCB images at HASP generation time and will inform the operator which image should be used on the various printers. The operator sets the FCB image by entering the following command:

$T printer,C=x - where x is the installation defined image id character.

Image character "V" may be designated by the installation as a variable image and, if so designated, may be changed by the operator. This single image may then be used by operators to set desired printer FCB images by the command:

$T printer,C=V

An operator may create the "V" image by entering the $TF command from an OS console, HASP local console, or reader with device and system HASP command authority.

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#)" parameters 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 the first special forms data set. When this data set has been printed the job will be queued for the printing of the next special forms data set (if any). This process will continue until all special forms data sets have been processed. The data sets will be queued in the order of the collating sequence of the special forms designations. After the special forms printer(s) complete all special forms printing, the job will proceed to special forms data set punching (again in collating sequence), and then, if appropriate, to standard job punching.

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, S=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).

4. The operator is permitted to change carriage tapes and print trains/chains while HASP is waiting for the $S command following a request for forms load. This change may be activated by entering $T device operands C=carriage or T=train as appropriate.

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

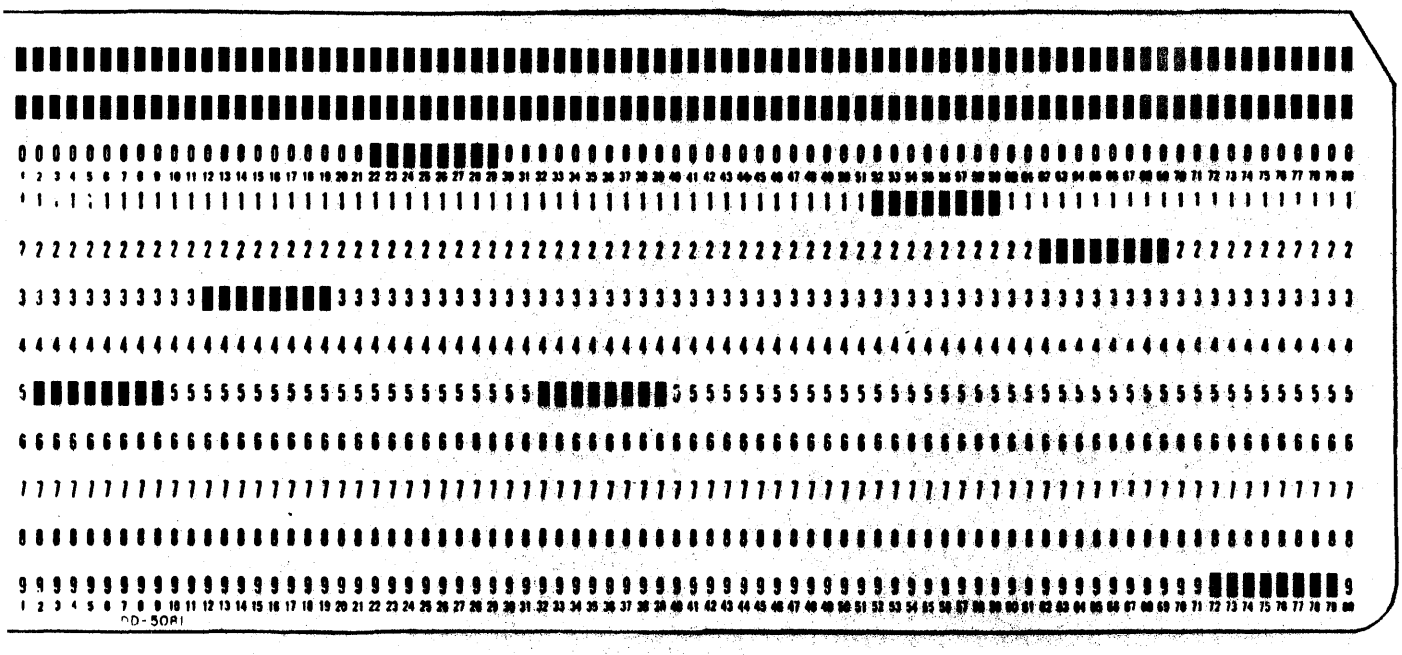
Columns	Contents	Mode
1 - 20	Programmer's name	EBCDIC
21 - 24	Room number	EBCDIC
25 - 27	Spares	N/A
28 - 31	P. A. number	EBCDIC
32	Job priority number	BINARY
33 - 35	Job input time in hundredths of a second	BINARY
36 - 38	Job output time in hundredths of a second	BINARY
39 - 40	Number of cards read in	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 appropriate hardware SRL (e.g., GA21-9033 for the 2540, GA21-9124 for the 3525) should be consulted for correct operator procedures in response to indicator lights and sense bits displayed when the punch stops and requires intervention.

(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.

13. If the Model 20 has less than 16,000 bytes of memory a program stop will occur after the loading of the first card. When this occurs, press only the LOAD key again to initiate loading.
14. Press the START key on the communications adapter while the HASP/RTP deck is being loaded into memory.
15. Establish the communications line (see Section 2.2).
16. The program should now begin processing in accordance with the algorithm given in Section 2.3.

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

```
0         1     1                                                  8
1         2     6                                                  0

//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-serl,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 subrecord 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, must 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 COMMUNICATIONS 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.

The following is a list of error messages which may be displayed on the Console Printer:

MESSAGE	MEANING	ACTION TAKEN
FFRREE00	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 signalled to restart the job again.
02DDDD00	Abnormal read complete. Number of bytes requested have been read but no end sequence was detected.	HASP will be requested to retransmit the record.
03DDDD00	Receive timeout while attempting to synchronize an initial sequence.	HASP will be requested to retransmit the record.
04DDDD00	Receive timeout while reading data.	HASP will be requested to retransmit the record.
05DDDD00	BCC compare error after normal read complete condition.	HASP will be requested to retransmit the record.

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.
0DDDD00	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. Note: 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. UNPREDICTABLE RESULTS MAY OCCUR. 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 or 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.

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 recovery - Attempting to recover 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 wait for an initial sequence request from the HASP/RMT360 program. HASP/RMT360 will log on the console message device, UNIT CHECK. 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 - Communicaton Adapter Error Messages (Continued)

MESSAGE	MEANING	ACTION TAKEN
	Sense Information byte 80 Command reject 40 Intervention required 20 Bus out check 10 Equipment check 08 Data Check 04 Overrun 02 Lost data 01 Time out	If write operation in process at time of error, the write will be reissued; otherwise, HASP will be requested to retransmit.
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/O instruction was rejected by the Synchronous Data Adapter	The start I/O 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.)

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 indicator lights should show periodic transmit and receive activity.

11. The remote terminal is now ready to communicate 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 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 begins. This will be indicated by BUSY being 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 not 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 not 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 id entified 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 will wait for an initial sequence request from the HASP/RMTM20 program. Versions of the HASP/RMTM20 which support console messages will log on the console message device, UNIT CHECK. 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	1. Type error message (note 1)	1. STOP with device address in ESTR register (note 2)
2460 - read	2. Wait for .SR1 command	2. Reread when CPU STARTed
	3. Read	
1442 - punch	1. Type error message (note 1)	1. STOP with device address in ESTR register (note 2)
	2. Wait for .SU1 command	2. When CPU started repunch record in error
	3. Repunch record in error	
2520, 2560 - punch	1. Select out card in error	1. Select out card in error
	2. Repunch record in error	2. Repunch record in error
2203, 1403 - print	Ignore error	Ignore error
2152 - write	1. Ignore first error	NA
	2. Wait on next attempt to use device (note 3)	
2152 - read	1. Initiate re read	NA
	2. Wait on next attempt to use device (note 3)	

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 <u>printing</u> <u>only</u> 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 the alarm comes on (in 15 to 45 seconds depending on 2780 wiring options) and cards are in the stacker, then a block of data will be sent twice. You must ask the central operator to cancel the partially read job. Begin transmission as in Section 2.1.1 with the JOB card of the partially read job.

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 printer only 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 or Additional Buffer Expansion features. These features affect 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 procedurs 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-transparency.

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. The Additional Buffer Expansion feature provides two 512 character buffers. 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 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.

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, the type bar properly

installed (2203 only), the cover is closed, and the printer START has been pressed (2203 only). If you have a 2203 printer, the INHIBIT IRS switch should be off when operating with HASP.

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

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 ENTRY - HASP has failed to respond to the 2770 within 15 to 45 seconds (depends on 2770 wiring option). If any cards are in the stacker, a duplicate block of data will probably be sent. Follow the job restart procedure given previously.

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 retransmissions 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 (2213), FORM CHECK (2203), END OF FORM (2203), CARRIAGE INTERLOCK (2203) - The printer carriage, forms, or carriage tape are not ready or jammed. Correct the condition, press RESET (2203), console CHECK RESET, START (2203), and console START.

PRINT CHECK (2213), other CHK lights (2203) - The printer had a parity error or other hardware malfunction. See the SRL to interpret CHK lights. Press RESET (2203), console CHECK RESET, START (2203), and console 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 requeued in the print queue for your terminal. You may cause this "suspend" action yourself by pressing STOP on the 2213 or CARRIAGE STOP on the 2203 while printing, then readying the printer.

If you have a 2203, you may press the STOP key to make minor carriage adjustments without causing a "suspend", if you ready the printer within 34 seconds or periodically press STOP, CARRIAGE SPACE, or CARRIAGE RESTORE to extend the 34 second period.

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 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 message (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.

(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 either a deck of 96-column cards or a deck of 80-column cards.

To load the 96-column load deck--

1. Put the deck in the rightmost card hopper of the MFCU.
2. Hit START on the MFCU.
3. Hit PROGRAM LOAD on the System/3.
   (For disk systems, the program load selection knob must point to MFCU.)
4. Hit START on the printer.

To load the 80-column load deck--

1. Raise the CE Controls cover on the System/3.
2. With the CE Mode Selector at PROCESS, hit SYSTEM RESET.
3. Turn the CE Mode Selector to ALTER STOR.
4. Set the data knobs to C2 and hit START once.
5. Set the data knobs to 01 and hit START once.
6. Set the data knobs to 00 and hit START once.
7. Set the data knobs to 00 and hit START once.
8. Set the data knobs to 31 and hit START once.

9. Set the data knobs to 54 and hit START once.
10. Set the data knobs to 00 and hit START once.
11. Set the data knobs to 03 and hit START once.
12. Set the data knobs to F3 and hit START once.
13. Set the data knobs to 51 and hit START once.
14. Set the data knobs to 01 and hit START once.
15. Set the data knobs to F1 and hit START once.
16. Set the data knobs to 52 and hit START once.
17. Turn the CE Mode Selector to PROCESS.
18. Hit SYSTEM RESET.
19. Close the CE Controls cover.
20. Put the 80-column load deck in the 1442.
21. Hit START on the 1442, the printer, and the System/3. Cards should begin reading.
22. When the 1442 ready light goes out, again push START on the 1442.

Midway through the program deck, the reader 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 hopper is empty (or the first card in the 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 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 hopper contains an EOR or /*SIGNON card, remove the cards from the primary hopper and push STOP and then NPRO on the reader. 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

(The remainder of this page intentionally left blank.)

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.

The 5471 will not type messages if the end-of-forms switch is on. When the 5471 runs out of forms, reload forms, hit the REQ key, and then hit the END key. The 5471 will resume typing if there are messages to be typed.

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.

2.5 SYSTEM/3 LOCAL COMMAND PROCESSING

If your System/3 Remote Terminal Program includes the local command facility, you can read in commands to be executed locally by the System/3. Place a card containing the command into any dormant card hopper and hit START. Do not use a /*EOF card.

For each command, the command name starts in column 1 and is followed by the operand field.

Command completion is indicated by one of the following messages, printed in the same place as error messages:

CODE0000 - Command completed satisfactorily
CODE0001 - Syntax error in command
CODE0002 - Operand value error in command

2.5.1 /*CARRIAGE - Define Printer Carriage Information

The operand field has the format:

[L=forms-length][,chan=line-no][,chan=line-no...]

where

forms-length	is the number of print lines on a page of forms (must not be greater than 112)
chan	is the carriage-channel number (must be between 1 and 12)
line-no	is the line number at which forms skipping will stop for the indicated carriage channel (must not be greater than forms-length)

Notes:
1. The /*CARRIAGE command is effective immediately when read in.
2. Specification of the L= operand destroys all previous carriage channel settings.

(The remainder of this page intentionally left blank.)

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 retransmitted.

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 retransmit the block.

05SSSS00

MEANING - The System/3 BSCA has 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 05SSSS0F, 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.)

11.10 HASP REMOTE TERMINAL (3780) OPERATOR'S GUIDE

The following section contains detailed instructions for operating an IBM 3780 as a HASP remote workstation. This manual is intended as a removable section for use at the remote location.

(The remainder of this page intentionally left blank.)

T H E

H A S P

S Y S T E M

IBM 3780 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 SPOOL-ing. While HASP is reading or printing on your 3780, it may be simultaneously reading, printing and punching on all of the other 3780s, 2770s, 2780s, the 1130 systems, the S/3 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 System/3 Model 10, the 1130 system, 2780s, 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 thou-sands 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 di-rectly 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 3780 Data Communications Terminal can connect to a System/360 or System/370 using HASP to transmit jobs to the 360 or 370 for execution and to receive the printed output from those jobs. The 3780 is not a computer but rather an input/output device. HASP con-trols it much like any other input/output device, when connected to the central computer via telephone lines and an IBM 2701 Data Adapter or an IBM 2703 Transmission Control Unit.

The 3780 Terminal contains a standard card reader, printer, associ-ated control circuitry, and an operator control console. These de-vices (mechanical features, speeds, etc.) are described in the SRL "IBM 3780 Data Communications Terminal", form GA27-3063 which you should have on hand for reference when operating your 3780. 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 card or line to be transmitted or received, as well as the devices' mechanical speeds which are given in the SRL.

The remaining sections of this manual discuss switch setup; communications establishment; normal operating procedures for reading and printing jobs; error recovery; and control cards.

SWITCH SETUP

During all 3780 operations with HASP, certain console switches should be set as follows:

TERM MODE - LINE

SPACE COMPRESS/EXPAND - ON

OFFLINE TEST - OFF

AUTO RESTART - OFF

ANSWER - MANUAL

INQUIRY MODE - OFF

When power is on, console lights will show some of the settings which are in effect.

The TRANSPCY switch should be set in the OFF position except when you are transmitting EBCDIC card decks which use all 256 possible punch combinations (for example, object decks).

ESTABLISHMENT OF COMMUNICATIONS

Before your 3780 terminal can transmit jobs to or receive printed output from the central computer, the computer must establish a path of communication to the terminal and must recognize it as a 3780 terminal (rather than, say, a System/360 Model 20 being used as a terminal). Depending upon how your 3780 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 3780 POWER ON key is pressed, you may begin to read in a job or to print the output from a previous job.

However, if your 3780 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. Press the 3780 POWER ON key (located on the card reader).

2. Press the STOP key then the NPRO key 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, press 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 press the card reader START key to run the sign-on card into the card feed.

8. Press the START key 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 3780 is now signed on, and you may start reading or printing.

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 output. The 3780 will not transmit or receive jobs unless you have correctly signed on. Refer back to page 4 for the sign-on procedure.

At any given time, a signed-on 3780 is either reading in a job, printing a job, or waiting for work. Thus, the normal cycle of a job submitted from the 3780 is: reading the job (transmitting it to the central computer), waiting for it to execute, and receiving its printed output. Often, after an operator has read in a job through the 3780, he will disconnect (sign-off) the 3780 to save telephone line charges and sign-on at a later time to receive his output.

Since the 3780 does not have a card punch, the central HASP system will normally route any punched output produced by jobs submitted from a 3780 to the central computer card punch or another remote. If punched output is routed to your 3780, it will be printed on the printer.

TRANSMISSION TO THE CENTRAL COMPUTER

The 3780 can transmit jobs to the central computer only if signed on and not busy printing. However, you may make it ready to transmit any time that it is signed-on. You must not press TERM RESET while a job is being printed. If you accidentally do this, a line restart (described on page 9) must be done.

To transmit job(s), take the following steps:

1. Press the card reader STOP key then the NPRO key 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. Press card reader START to run cards into the feed. The reader 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 (for example, object decks).

5. Press the START key on the console; the BID light should come on.

6. If the 3780 is printing, it will continue until the end of the current job. Then, or as soon as START is pressed if the 3780 is idle, the 3780 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 press STOP on the card reader at any time to halt reading temporarily to facilitate removing cards from the stacker or adding more to the hopper. If you press reader START within 30 seconds, transmission will continue without a reader error condition.

It may be possible to interrupt printing to begin transmitting. See page 13 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 3780 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 at the central computer if the job was submitted from a 3780.

The 3780 will receive printed output from the central computer if HASP has output to send, the terminal is not transmitting, and the printer is ready. You should always have the printer ready, even when transmitting, so that the 3780 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, the typebar properly installed, the cover closed, and the printer START key has been pressed.

The printer INHIBIT IRS switch should always be off when operating with HASP.

A blinking PRINTER light on the console indicates that the printer is not ready. Even after making it ready, it may be necessary to press CHECK RESET and START on the console to make the light stop blinking.

Printed output jobs will be separated by separator pages, 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 3780. Some problems occur only rarely and are not documented here. See the SRL "IBM 3780 Data Communications Terminal", form GA27-3063, 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 3780.

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. You should tell the central computer operator to issue the HASP command:

   $RESTART LNEmm

   then re-establish communications as previously described on pages 4 and 5. 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 3780 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, OPERATE, BID, 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 3780 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, press reader START and console START.

Any card reader trouble while transmitting is indicated by a blinking READER light. 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. Press NPRO. Repair bottom hopper card if necessary and make sure the feed throat is clear. Reload cards. Press reader START and console START.

ATTENTION - Full stacker, empty hopper with EOF off, and cover open are possible causes. Correct, press reader START and console START.

READ 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, press 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 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 printer ready to accept the output and wait until the next output job ending to again attempt transmission, or try to cause a printer "suspend" as described on page 13.

BID RETRY - HASP has failed to respond to the 3780 within 15 to 45 seconds (depends on 3780 wiring option). If any cards are in the stacker, a duplicate block of data will probably be sent. Follow the job restart procedure given previously.

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 retransmissions 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 the current partially completed job from its beginning. You should ask the central computer operator to issue the HASP command:

$RESTART RMnn.PR1

to cause the restart. Make your printer ready and press CHECK RESET. Discard the partially completed output beginning with the last previous separator page. 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 a blinking PRINTER light and lights on the printer as follows.

FORM CHECK, END OF FORM, CARRIAGE INTERLOCK - The printer carriage, forms, or carriage tape are not ready or jammed. Correct the condition, press printer RESET, console CHECK RESET, printer START, and console START.

Other printer CHK lights - The printer had a parity error or other hardware malfunction. See the SRL to interpret CHK lights. Press RESET, console CHECK RESET, START, and console 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 backspace procedure described previously, which will make it easier to discard duplicate output at page or document boundaries.

Certain console lights may require your attention while receiving, as follows.

TERMINAL ADDRESS - HASP is trying to send output but your 3780 is not ready. Make sure your switch setup is correct, ready the printer and press console CHECK RESET.

OVERRUN - This usually indicates 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 does not continue within a short time, you must do a job restart.

Depending on an option chosen for the central HASP System at your installation, actions during printer error recovery may be somewhat different than described above. If this altered mode of printer operation is applicable to your 3780, 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 requeued in the print queue for your terminal. You may cause this "suspend" action yourself by pressing CARRIAGE STOP while printing, then readying the printer. You may press the printer STOP key to make minor carriage adjustments without causing a "suspend", if you ready the printer within 34 seconds or periodically press STOP, CARRIAGE SPACE, or CARRIAGE RESTORE to extend the 34 second period.

Actions after the printer "suspend" depend on the state of your terminal and the output queue. You may start transmission as described on page 7 and following or you may wait for more output. Print jobs of higher priority than the suspended job 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 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   PRINT LOCAL
```

This card causes printed output for the job within which it was found to be printed at the central computer instead of at the remote terminal. A good place to put this card is right after the //JOB card.

On the above ROUTE card, you may use REMOTEnn or PRINTERn in place of LOCAL, beginning in column 16. These alternate forms cause the printed output for the job to go to a remote other than yours, or to a specific printer at the central computer rather than any printer 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 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 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 3780 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 3780, your printer is not active; therefore, these commands have no effect. This is true even after you "suspend" a print job as previously described on page 13. 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 printer 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 3780 operator, are unable to enter the $S command to continue.

Assuming the above conventions, you should periodically submit the $DF command to determine if special forms jobs are queued for output on your terminal. If so, you should select the type of forms from those queued which you desire to process first on your printer, mount that forms, enter a command "$T RMn.PR1,F=forms#", and wait for printing to occur. When jobs stop processing on the printer, you should resubmit the $DF and change to a new forms if indicated. The parameter "F=RESET" should be used to return the printer 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.

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                    00020000
//SCRATCH EXEC PGM=IEHPROGM                                                 00040000
//SYSPRINT  DD SYSOUT=A                                                     00060000
//SYSRES    DD UNIT=SYSDA,VOLUME=SER=YYYYYY,DISP=OLD                        00080000
//SYSIN     DD *                                                            00100000
        RENAME DSNAME=SYS1.OLDNUC,NEWNAME=SYS1.NEWNUC,VOL=SYSDA=YYYYYY      00120000
      UNCATLG DSNAME=SYS1.NEWNUC                                            00140000
      SCRATCH DSNAME=SYS1.NEWNUC,VOL=SYSDA=YYYYYY,PURGE                     00160000
/*                                                                          00180000
//LKED    EXEC PGM=IEWL,PARM='XREF,LET,LIST,NCAL,SCTR',REGION=96K           00200000
//HASPOBJ   DD DSNAME=SYS1.HASPOBJ,DISP=SHR                                 00220000
//NUCLEUS   DD DSNAME=SYS1.NUCLEUS,DISP=SHR                                 00240000
//SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(10,5))                                00260000
//SYSLMOD   DD DSNAME=SYS1.NEWNUC,UNIT=2314,VOLUME=SER=YYYYYY,              C00280000
//             DISP=(NEW,CATLG),LABEL=EXPDT=99366,                          C00300000
//             SPACE=(TRK,(40,,2),,CONTIG)                                  00320000
//SYSPRINT  DD SYSOUT=A                                                     00340000
//SYSLIN    DD *                                                            00360000
        INSERT IEAANIPO                                                     00380000
        INSERT IEAAIH00                      USE ONLY FOR MFT               00400000
        INSERT IEAQFX00                      USE ONLY FOR MVT               00420000
        INSERT IGFCCH                  USE FOR SYSTEMS WITH CCH  V03.1      00430000
       INCLUDE HASPOBJ(HASPSVC)                                             00440000
       INCLUDE NUCLEUS(IEANUC01)                                            00460000
          NAME IEANUC01(R)                                                  00480000
/*                                                                          00500000
//RENAME  EXEC PGM=IEHPROGM                                                 00520000
//SYSPRINT  DD SYSOUT=A                                                     00540000
//SYSRES    DD UNIT=SYSDA,VOLUME=SER=YYYYYY,DISP=OLD                        00560000
//SYSIN     DD *                                                            00580000
        RENAME DSNAME=SYS1.NUCLEUS,NEWNAME=SYS1.OLDNUC,VOL=SYSDA=YYYYYY     00600000
        RENAME DSNAME=SYS1.NEWNUC,NEWNAME=SYS1.NUCLEUS,VOL=SYSDA=YYYYYY     00620000
/*                                                                          00640000
```

12.1.2 Sample Job HASPROCS

```
//HASPROCS JOB (0000,0000),'INSTALL HASP PROCS',MSGLEVEL=1                  00660000
//PROCS   EXEC PGM=IEBUPDTE,PARM=NEW                                       00680000
//SYSPRINT  DD SYSOUT=A                                                     00700000
//SYSUT2    DD DSNAME=SYS1.PROCLIB,DISP=SHR,DCB=LRECL=80                    00720000
//SYSIN     DD DATA                                                         00740000
./          ADD NAME=HASP,LIST=ALL                                        00760000
./       NUMBER NEW1=20000,INCR=20000                                     00780000
//HASP    PROC JOB=STRTHASP                                               00800000
//IEFPROC EXEC PGM=IEFIRC,REGION=50K,                                     C00820000
//             PARM='0149990010012490510OSYSDA   E000'                    C00840000
//                  BPPTTTOOOMMMIIICCCRLSSSSSSSSAAAA                      00860000
//IEFRDER   DD DSNAME=SYS1.PROCLIB(&JOB),DISP=SHR,                        C00880000
//             DCB=(BUFNO=1,RECFM=FB,LRECL=80,BLKSIZE=80)                 00900000
//IEFPDSI   DD DSNAME=SYS1.PROCLIB,DISP=SHR                               00920000
//IEFDATA   DD DUMMY                                                      00940000
./          ADD NAME=STRTHASP,LIST=ALL                                    00960000
./       NUMBER NEW1=20000,INCR=20000                                     00980000
//HASP    JOB 'HASP-II','INVOKE HASP SYSTEM',CLASS=H,MSGCLASS=H           01000000
//SYSTEM  EXEC PGM=HASP,TIME=1440,REGION=51K                              01020000
//OLAYLIB   DD DSNAME=SYS1.HASPOLIB,DISP=SHR                              01040000
// S INIT.P0,,,H                                                          01060000
//                                                                        01080000
./          ADD NAME=HOSRDR,LIST=ALL          ASB RDR FOR MVT ONLY      01100000
./       NUMBER NEW1=20000,INCR=20000                                     01120000
//IEFPROC EXEC PGM=IEFVMA,REGION=18K,                                V03.1C01140000
// PARM='0010300010012520501ISPOOL   E00001,1011207604E000SYSDA   00' 1 01160000
//*       BPPTTTOOOMMMIIICCCRLSSSSSSSSAAAAEF,EJJAARRATABAAADDDDDDDDGK    01180000
//IEFRDER   DD UNIT=00C,DISP=OLD,                                         C01200000
//             DCB=(RECFM=F,LRECL=80,BLKSIZE=80,BUFNO=1)                  01220000
//IEFPDSI   DD DSNAME=SYS1.PROCLIB,DISP=SHR                               01240000
//IEFDATA   DD UNIT=SYSDA,VOLUME=REF=SYS1.LINKLIB,                        C01260000
//             SPACE=(80,(200,200),RLSE,CONTIG),DISP=OLD,                 C01280000
//             DCB=(DSORG=PS,RECFM=FB,LRECL=80,BUFL=80,BLKSIZE=80)        01300000
./          ADD NAME=HOSRDR,LIST=ALL          STD RDR FOR MFT OR MVT    01320000
./       NUMBER NEW1=20000,INCR=20000                                     01340000
//IEFPROC EXEC PGM=IEFIRC,REGION=52K,                                V03.1C01360000
//             PARM='0010300010012520501ISPOOL   '                        C01380000
//                  BPPTTTOOCMMMIIICCCRLSSSSSSSS                          01400000
//IEFRDER   DD UNIT=00C,DISP=OLD,                                         C01420000
//             DCB=(RECFM=F,LRECL=80,BLKSIZE=80,BUFNO=1)                  01440000
//IEFPDSI   DD DSNAME=SYS1.PROCLIB,DISP=SHR                               01460000
//IEFDATA   DD UNIT=SYSDA,VOLUME=REF=SYS1.LINKLIB,                        C01480000
//             SPACE=(80,(200,200),RLSE,CONTIG),DISP=OLD,                 C01500000
//             DCB=(DSORG=PS,RECFM=FB,LRECL=80,BUFL=80,BLKSIZE=80)        01520000
./          ADD NAME=HOSWTR,LIST=ALL          USED ONLY IF &WTRPART NE * 01540000
./       NUMBER NEW1=20000,INCR=20000                                     01560000
//IEFPROC EXEC PGM=IEFSD080,PARM='PA',REGION=12K                          01580000
//IEFRDER   DD UNIT=1403,DSNAME=SYSOUT,DISP=(NEW,KEEP),                   C01600000
//             DCB=(RECFM=FM,LRECL=133,BLKSIZE=133,BUFL=133,BUFNO=1)      01620000
./        ENDUP                                                         01640000
/*                                                                      01660000
```

Reference Listing of HASPJCL - Page 12.1-3

12.1.3 Sample Job HASPHASP

```
//HASPHASP JOB (0000,0000),'INSTALL HASP PROGRAM',MSGLEVEL=1                01680000
//SCRATCH EXEC PGM=IEHPROGM                                                 01700000
//SYSPRINT  DD SYSOUT=A                                                     01720000
//OLAYLIB   DD UNIT=SYSDA,VOLUME=SER=ZZZZZZ,DISP=OLD                        01740000
//SYSIN     DD *                                                            01760000
       UNCATLG DSNAME=SYS1.HASPOLIB                                         01780000
       SCRATCH DSNAME=SYS1.HASPOLIB,VOL=SYSDA=ZZZZZZ,PURGE                  01800000
/*                                                                          01820000
//OBLD    EXEC PGM=HASPOBLD                                                 01840000
//STEPLIB   DD DSNAME=SYS1.HASPMOD,DISP=SHR                                 01860000
//SYSIN     DD *,DCB=BLKSIZE=80                                             01880000
/*                                                                          01900000
//SYSOBJ    DD DSNAME=SYS1.HASPOBJ(HASPNUC),DISP=SHR                        01920000
//          DD DSNAME=SYS1.HASPOBJ(HASPRDR),DISP=SHR                        01940000
//          DD DSNAME=SYS1.HASPOBJ(HASPXEQ),DISP=SHR                        01960000
//          DD DSNAME=SYS1.HASPOBJ(HASPPRPU),DISP=SHR                       01980000
//          DD DSNAME=SYS1.HASPOBJ(HASPACCT),DISP=SHR                       02000000
//          DD DSNAME=SYS1.HASPOBJ(HASPMISC),DISP=SHR                       02020000
//          DD DSNAME=SYS1.HASPOBJ(HASPCON),DISP=SHR                        02040000
//          DD DSNAME=SYS1.HASPOBJ(HASPRTAM),DISP=SHR                       02060000
//          DD DSNAME=SYS1.HASPOBJ(HASPCOMM),DISP=SHR                       02080000
//          DD DSNAME=SYS1.HASPOBJ(HASPINIT),DISP=SHR                       02100000
//SYSLIN    DD DSNAME=&&TEMP,UNIT=SYSSQ,DISP=(NEW,PASS),                   C02120000
//             SPACE=(400,(400,50)),DCB=BLKSIZE=400                         02140000
//OLAYLIB   DD DSNAME=SYS1.HASPOLIB,UNIT=SYSDA,VOLUME=SER=ZZZZZZ,         C02160000
//             DISP=(NEW,CATLG),LABEL=EXPDT=99366,                        C02180000
//             SPACE=(1024,60,,CONTIG)                          V03.1 02200000
//SYSPRINT  DD SYSOUT=A,DCB=BLKSIZE=121                                     02220000
//LKED    EXEC PGM=IEWL,PARM='LIST,XREF',REGION=96K,COND=(4,LT,OBLD)   1 02240000
//HASPOBJ   DD DSNAME=SYS1.HASPOBJ,DISP=SHR                                 02260000
//SYSUT1    DD DSNAME=SYS1.UT3,DISP=OLD                                     02280000
//SYSLMOD   DD DSNAME=SYS1.LINKLIB,DISP=OLD                                 02300000
//SYSPRINT  DD SYSOUT=A                                                     02320000
//SYSLIN    DD DSNAME=&&TEMP,DISP=(SHR,PASS)                                02340000
//          DD *                                                            02360000
          NAME HASP(R)                                                      02380000
       INCLUDE HASPOBJ(HASPBR1)                                             02400000
          NAME HASPBR1(R)                                                   02420000
       INCLUDE HASPOBJ(HASPWTR)                                             02440000
          NAME HASPWTR(R)                                                   02460000
/*                                                                          02480000
```

12.1.4 Sample Job HASPOOLS

```
//HASPOOLS JOB (0000,0000),'ALLOCATE SPOOL SPACE',MSGLEVEL=1                02500000
//SCRATCH EXEC PGM=IEHPROGM                                                 02520000
//SYSPRINT  DD SYSOUT=A                                                     02540000
//SPOOL1    DD UNIT=SYSDA,VOLUME=SER=SPOOL1,DISP=OLD                        02560000
//SPOOL2    DD UNIT=SYSDA,VOLUME=SER=SPOOL2,DISP=OLD                        02580000
//SYSIN     DD *                                                            02600000
       SCRATCH VTOC,VOL=SYSDA=SPOOL1,PURGE                                  02620000
       SCRATCH VTOC,VOL=SYSDA=SPOOL2,PURGE                                  02640000
/*                                                                          02660000
//ALLOCAT EXEC PGM=IEFBR14                                                  02680000
//SPOOL1    DD DSNAME=SYS1.HASPACE,VOLUME=SER=SPOOL1,                      C02700000
//             DISP=(NEW,KEEP),LABEL=EXPDT=99366,                         C02720000
//             UNIT=2314,SPACE=(ABSTR,(3998,2))                            02740000
//SPOOL2    DD DSNAME=SYS1.HASPACE,VOLUME=SER=SPOOL2,                      C02760000
//             DISP=(NEW,KEEP),LABEL=EXPDT=99366,                         C02780000
//             UNIT=3330,SPACE=(ABSTR,(7674,2))                            02800000
```

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 partition 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 for 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:

$$S_{HASP} = 21{,}900 + S_1 + S_2 + S_3 + S_4 + S_5 + S_6 + S_7 + S_8 + S_9 + S_{10} + S_{11} + S_{12} + S_{13} + S_{14} + S_{15} + S_{16} + S_{17} + S_{18} + S_{19} + S_{20} + S_{21} + S_{22} + S_{23} + S_{24} + S_{25} + S_{26} + S_{27} + S_{28} + S_{29} + S_{30} \text{ bytes.}$$

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 (400+DAMAP)

S_2 = &NUMTPES x (400+DAMAP)

S_3 = { 0 . . . . . if &NUMINRS = 0
        { 398 + &NUMINRS x (496+DAMAP) . . . . if &NUMINRS ≠ 0

S_4 = &NUMPRTS x (316 + 8x&NOPRCCW)

S_5 = &NUMPUNS x (372 + 8x&NOPUCCW)

S_6 = { { 0 . . . . . if &AUTORDR = NO
          { 120 . . . . . if &AUTORDR = YES } if &NUMCONS = 0
          { 800 + 164 x &NUMCONS + 32 x &RQENUM
          { + 298 . . . . . if &SIZ2260 ≠ 0
          { + 60 . . . . . if &AUTORDR = YES } if &NUMCONS ≠ 0

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} = { 0 . . . . . if &JITSIZE = 0
           { 86 + &JITSIZE x &MAXJOBS . . . . . if &JITSIZE ≠ 0

S_{13} = &MAXXEQS x 216

S_{14} = &MAXPART x (12+&MAXCLAS)

S_{15} = &NUMDDT x 37

S_{16} =
- 0 . . . . . if &MONINTV = 0
- 252 + 12 x &MAXXEQS if &XZMULT = NO
- 252 + 16 x &MAXXEQS if &XZMULT = YES

} if &MONINTV ≠ 0

S_{17} =
- 0 . . . . . if &PRIRATE = 0
- 108 . . . . . if &PRIRATE ≠ 0

S_{18} =
- 0 . . . . . if &WTRPART ≠ *
- 1504 . . . . . .
- + 96 . . . . if &WCLSREQ ≠ ********
- + 112 . . . . . if &RPS = YES

} if &WTRPART = *

S_{19} =
- 0 . . . . . if &OSINOPT = NO
- 26 . . . . . if &OSINOPT = YES

S_{20} =
- 0 . . . . . if &XBATCHC = null
- 274 + 14 x &MAXPART if &NUMCONS = 0
- 316 + 14 + &MAXPART if &NUMCONS ≠ 0

} if &XBATCHC ≠ null

S_{21} =
- 0 . . . . . if &TIMEOPT = 4
- 134 . . . . . if &TIMEOPT ≠ 4

S_{22} =
- 0 . . . . . if &PRTRANS = NO
- 314 . . . . . if &PRTRANS = YES

S_{23} =
- 0 . . . . . if &ACCTNG = NO
- 46 . . . . . if &ACCTNG = YES

S_{24} =
- 0 . . . . . if &DEBUG = NO
- 1632 + DAMAP . . . . . if &DEBUG = YES

S_{25} =
- 0 . . . . . if &TRACE = 0
- 534 + 64 x &TRACE . . . . . if &TRACE ≠ 0

S_{26} =
- 0 . . . . . if &OREPSIZ = 0
- 72 + &OREPSIZ . . . . . if &OREPSIZ ≠ 0

S_{27} =
- 0 . . . . . if &DMPTAPE = 000
- 560 . . . . . if &DMPTAPE ≠ 000

S_{28} =
- 0 . . . . . if &FCBV = NO
- 222 . . . . . if &FCBV = YES

S_{29} =
- 0 . . . . . if &RPS = NO
- 140 + 32 * &NUMDA . . . . . if &RPS = YES

$$S_{30} = \begin{cases} 0 \text{ . . . . . if \&NUMLNES = 0} \\ R_1 + R_2 + R_3 + R_4 + R_5 + R_6 + R_7 + R_8 + R_9 + R_{10} + R_{11} \text{ if \&NUMLNES} \neq 0 \end{cases}$$

where:

R_1 = &NUMTPBF x (144+&TPBFSIZ)

R_2 = &NUMLNES x 104

R_3 = &NUMRJE x 132

R_4 = &NUMTPRD x (372+DAMAP)

R_5 = &NUMTPPR x 200

R_6 = &NUMTPPU x 168

$$R_7 = \begin{cases} 0 \text{ . . . . . if \&SPOLMSG = 0} \\ (\&SPOLMSG+7)/8 + \&NUMRJE \times 8 \text{ . if \&SPOLMSG} \neq 0 \end{cases}$$

$$R_8 = \begin{cases} 0 \text{ . . . . . if \&BSHTAB = NO} \\ 194 \text{ . . . . . if \&BSHTAB = YES} \end{cases}$$

$$R_9 = \begin{cases} 0 \text{ . . . . . if \&BSHPRES = NO} \\ 196 \text{ . . . . . if \&BSHPRES = YES} \end{cases}$$

$$R_{10} = \begin{cases} 0 \text{ . . . . . if \&USASCII = NO} \\ 548 \text{ . . . . . if \&USASCII = YES} \end{cases}$$

R_{11} = a values selected from the following table:

&STR1978	&STRCPU	&BSC2770 or &BSC2780 or &BSC3780	&BSCCPU	R_{11}
YES	NO	NO	NO	6,290
NO	YES	NO	NO	5,394
YES	YES	NO	NO	8,002
NO	NO	YES	NO	5,840
YES	NO	YES	NO	8,936
NO	YES	YES	NO	8,048
YES	YES	YES	NO	10,632
NO	NO	NO	YES	6,202
YES	NO	NO	YES	9,770
NO	YES	NO	YES	8,802
YES	YES	NO	YES	11,402
NO	NO	YES	YES	8,460
YES	NO	YES	YES	11,540
NO	YES	YES	YES	10,588
YES	YES	YES	YES	13,172

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          &MAXCLAS = 8
&NUMTGV  = 400        &NUMDDT  = 16
&NUMRDRS = 1          &MONINTV = 0
&NUMTPES = 0          &PRIRATE = 0
&NUMINRS = 0          &WTRPART = *
&NUMPRTS = 1          &WCLSREQ = ********
&NOPRCCW = 5          &RPS     = NO
&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          &FCBV    = NO
&MAXXEQS = 2          &NUMLNES = 0
&MAXPART = 2
```

The storage requirements would be computed as follows:

DAMAP = 1 x 50 = 50

$$S_{HASP} = 21,900 + 450 + 0 + 0\ 356 + 412 + 120 + 158 + 6,424 + 1,112 + 700 + 800 + 0 + 432 + 40 + 592 + 0 + 0 + 1,504 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0$$

$$= 35,000 \text{ bytes.}$$

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          &RPS     = NO
&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        &FCBV    = YES
&NUMBUF  = 20         &NUMLNES = 2
&BUFSIZE = 688        &NUMTPBF = 2
&NUMOACE = 2          &TPBFSIZ = 400
&NUMWTOQ = 10         &NUMRJE  = 2
&MAXJOBS = 200        &NUMTPRD = 2
&JITSIZE = 8          &NUMTPPR = 2
&MAXXEQS = 3          &NUMTPPU = 2
&MAXPART = 3          &SPOLMSG = 20
&MAXCLAS = 8          &BSHTAB  = YES
&NUMDDT  = 30         &BSHPRES = NO
&MONINTV = 3          &USASCII = NO
&XZMULT  = YES        &STR1978 = NO
&PRIRATE = 3          &STRCPU  = NO
&WTRPART = *          &BSC2770 = YES
&WTRCLAS = HAQ        &BSC2780 = YES
&WCLSREQ = **R        &BSCCPU  = NO
```

The storage requirements would be computed as follows:

DAMAP = 2 x 50 = 100

S_{HASP} = 21,900 + 1,000 + 500 + 994 + 872 + 412 + 120 + 316 +
15,360 + 2,224 + 1,400 + 3,200 + 1,686 + 648 + 60 +
1,110 + 300 + 108 + 1,600 + 26 + 316 + 0 + 314 +
46 + 0 + 0 + 0 + 0 + 222 + 0 +
(1088+208+264+944+400+336+19+194+0+0+5840)
= 64,027 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) This count applies only to data sets printed on job forms and demand forms. Only one copy of data sets indicated as specially routed data sets will be produced.

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

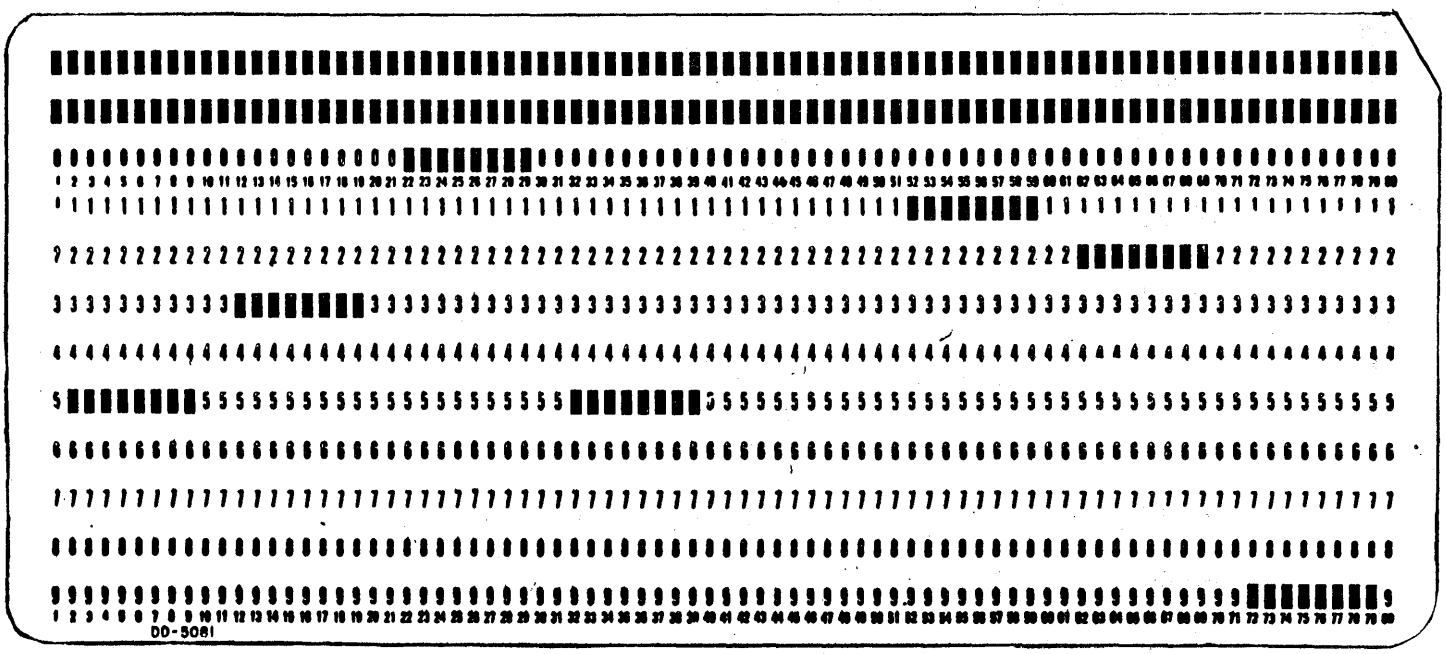
12.5 SPOOL PRINT AND PUNCH ID FORMATS

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 {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

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 may 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.

G. No provision has been made to support the interpreter restart function when using the ASB Reader with OS MVT. If message IEF336I occurs, the job must be resubmitted and the space for SYS1.SYSJOBQE must be increased.

H. HASP does not support the continuation of the DD * or DD DATA JCL statements under any conditions.

I. The HASP support of the DLM parameter on DD * and DD DATA JCL statements is compatible with the OS support of this parameter with the following exceptions:

- The OS Input SPOOLing Option will not be activated for any input data set specifying the DLM parameter.
- If DCB parameters are specified, they must be specified physically before the DLM specification (i.e., the DLM specification must be the last parameter on the DD statement).
- The apostrophe (') cannot be used as a delimiter character.
- HASP Control Cards (/*MESSAGE, /*SETUP, /*ROUTE, etc.) will not be recognized if the DLM specification is other than "/*".

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 WTO and WTOR support and may not function properly under certain conditions. For example, HASP console support may not function properly when:

- REPLYs to WTORs are erroneously left pending.
- A non-HASP (e.g., Z EOD) command issues a WTO or a WTOR, unless it is operating under a separate task.

See HASPGEN parameter &NUMCONS for other restrictions. These functional restrictions may be removed by using OS console support (&NUMCONS=0).

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 responsiblity 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.

F. If any SYSOUT data set contains more than 65,535 pages (or skips to any channel in the carriage tape), then print positioning (either forward/backward spacing or warm start spacing) will not function after the 65,535 page (or skip) is reached.

G. While HASP makes an attempt to enter every WTO and WTOR in the HASP System Log of the job associated with the message, there are certain messages (e.g., DDR and MFT I/O Error messages) which are not readily associated with any given job and may not be logged.

H. While HASP is programmed to recover from most catastrophic Input/Output errors in such a way that the impact on the installation will be minimal, it is conceivable that multiple unusual errors might occur in a time relationship such that loss of data is inevitable and complete recovery by HASP is impossible.

(The remainder of this page intentionally left blank.)

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.

3. If a SYSOUT entry exists, the class is saved for possible translation to a HASP pseudo unit. The class-pseudo unit translation table (XTRTABLE) entry corresponding to the SYSOUT class is examined for the values "*", "1", or "2". If "*" is found, HASP processing is terminated for the DD card. If "1" or "2" is found (indicating special routing), the card will be processed as if special forms were specified. If the card was a simple SYSOUT=x, control goes to the new internal text string build processor.

4. If the SYSOUT entry includes a special output writer specification, control goes to XJCLDDWR.

5. If the SYSOUT specified includes a forms field but not a special writer, the forms field is extracted for later processing.

6. The DDNAME, if it exists, is moved to the HASP text buffer where internal text for UNIT=x is constructed.

7. If the forms field was detected, then control goes to XJCLDDFR, otherwise the SYSOUT class is used to translate to a HASP pseudo unit and the completed UNIT=x text is moved to the Reader/Interpreter text buffer. Control returns to the Reader/Interpreter via XJCLEXIT.

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 execution DDT chain is searched for an unassigned DDT. The EBCDIC unit is extracted from the first available DDT and inserted in the "xxx" subfield of the "UNIT=xxx" text.

4. If DCB parameters exist, control goes to XJCLINDB, otherwise the HASP default BLKSIZE (defined by &IBLKSZE) and RECFM=F are added to the constructed text.

5. Control goes to the termination section of XJCLDDPR where the new internal text is moved to the reader text buffer and processing is completed.

XJCLINDB - DD*, DD DATA and DCB Processing Routine

Input stream statements (DD*, DD DATA) with DCB parameters are processed by this routine in the following manner:

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 input stream BLKSIZE (defined by &IBLKSZE) is moved to a test location (XDEFOBXX) for further analysis by XJCLSHFL.

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 and 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 no 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 not 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 PROCEDURE 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 resident (non-swappable) 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 not 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 HASPGEN 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 of allocation. The following card might be used to name three internal readers:

```
UNITNAME        unit=(301,302,303),NAME=INTRDR
```

12.10.4 DELETION OF CURRENT JOB ON READER

If the submitting task determines that the previous JCL and/or data of the job currently being punched into the internal reader is incorrect a deletion card (/*DEL in columns 1-5) may be punched. This will cause the job currently on the device to be deleted as though cancelled by the operator.

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 interleaved 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)

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 = Nonduplicate 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)

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

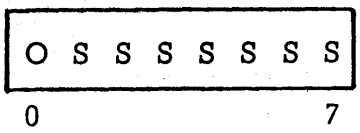
= 0110 = Data set record

= 0111 = Terminal message routing request

= 1000 - 1100 = Reserved for future expansion

= 1101 - 1111 = Available for local modifications

Sub-Record Control Byte (SRCB)

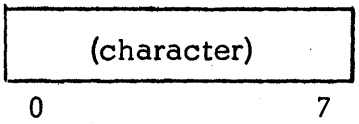


Usage: To provide supplemental information about a record

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



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

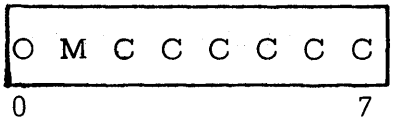
= G = System configuration status

= H = Diagnostic control record

= I - R = Reserved

= S - Z = Available for local modification

SRCB for Print Records



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

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

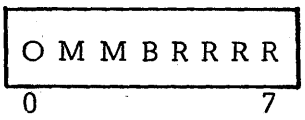
= 01 = SCB count units = 2

= 10 = SCB count units = 4

= 11 = Reserved

RR = Reserved for future expansion

SRCB for Input Record



Usage: To provide additional information for input records

Bit Meanings: O = 1 (Must always be ON)

M = 00 = SCB count units = 1

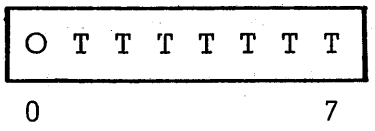
= 01 = SCB count units = 2

= 10 = SCB count units = 4

= 11 = Reserved

RRRR = Reserved

SRCB for Terminal Message Routing Record



Usage: To indicate the destination of a terminal message

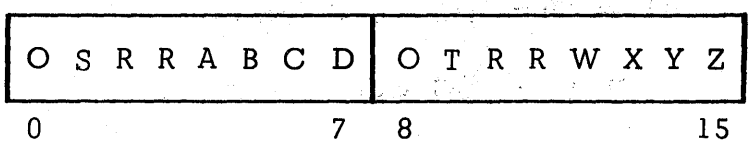
Bit Meanings: O = 1 (Must always be ON)

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)



Usage: To control the flow of individual function streams

Bit Meanings: O = 1 (Must always be on)

S = 1 = Suspend all stream transmission (WAIT-A-BIT)

= 0 = Normal state

T = Remote console stream identifier

R = Reserved for future expansion

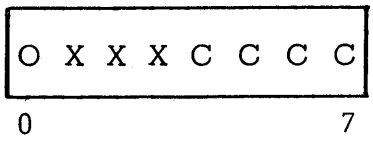
ABCD...WXYZ = Various function stream identifiers (oriented only to recipient)

- 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)



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 AND 3780 RJE SUPPORT

12.12.1 2770 Configuration

The basic 2770 with standard keyboard and either EBCDIC or USASCII code is supported.

Optional supported devices are the 2502 Card Reader, 2213 Printer, 2203 Printer, and 545 Output Punch. The printer must be attached to OUTPUT PRINTER. The card punch must be attached to OUTPUT 2.

EBCDIC Transparency, Printer Horizontal Format Control, Space Compression/Expansion, and any of the three buffer sizes (128, 256, 512) are supported by HASP programming. The Multipoint Data Link Control feature and Identification features must not be present. All other devices and features may be attached, but are either not affected by programming or not supported.

12.12.2 I/O Formats

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 Representative to establish the feasibility of the proposed device usage in the customer's environment. Refer to the SRL GA27-3013, especially pages 2772-9, 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. Compressed blanks, indicated by the IGS characters, are detected and expanded prior to processing by OS, if the HASPGEN parameter &BSHPRES=YES.

2. If the block is transparent, it is interpreted as one or more records of exactly 80 data 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 compressed blanks in non-transparent input blocks and proper handling of transparent input blocks are 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 preceding 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, 256, or 512 bytes, as indicated 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.

If indicated by parameter RMTnn (and supporting settings of &BSHTAB and &BSHPRES), blanks are compressed and encoded using either the HT or IGS characters. Encoding by HT sets electronic tabs, every 10 columns beginning at column 11. This can be changed by altering internal assembly variable &HTDIST.

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. Blanks are compressed and encoded using the IGS character, if indicated by RMTnn (and supporting &BSHPRES). 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.

12.12.3 3780 Support

The previous description of 2770 support applies to the 3780 also, with minor exceptions: The 3780 is assumed to have standard 512 byte buffers, card reader, printer, but no keyboard or card punch. Component selection characters are not sent to the 3780. Although features Transparency, Horizontal Format, and Compression are standard; use of them for output is controlled by the RMTnn parameter, as with 2770.

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".

If it is necessary that the batch processing program have a WTOR outstanding past the ending of one of its "jobs", the HASPGEN option &NUMCONS=0 must be used.

The batch processing program will receive an actual end-of-file condition when a card having $$ in columns 1 and 2 is read while processing a "job". The program may continue to the next logical subfile by a variety of techniques. It may simply reset appropriate bits in OS I/O control blocks and continue reading or it may CLOSE the data set. The data set may then be re-OPENed to continue reading at the card following the $$ 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.

The special forms field in SYSOUT must not be used in any batching procedure. If OS output spooling is used with any SYSOUT (see HASPGEN parameter $$X), the output is not queued for the OS writer until the batch processing program terminates, which is not necessarily when any batch job terminates.

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 initially 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 modules 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.

12.5.3 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. Performs backspace editing of the command in accordance with the HASPGEN parameter "$BSPACE". 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. The SVC 35/36 Processing routine is entered for possible logging of the expanded reply on the HASP SYSTEM LOG. Control is returned to process the XCTL with expanded reply.

3. If the first character is non-numeric, an explicit test is made for a "$" which identifies HASP commands. Control is returned to process the XCTL if the first character is non-numeric or not a "$".

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, 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
```

12.17 3211 FORMS CONTROL BUFFER ADDITIONAL LOADS

Installations using HASP with 3211 printers may want to add carriage tape images to HASP in addition to the images provided. Each such image is named by a letter or a number, but the number '1' is reserved to allow the operator to force single-spacing and the letter 'V' is reserved for the operator-variable FCB load. Of the 34 remaining alphanumerics, HASP supplies images for '6', '8', and 'U'.

12.17.1 Adding and Changing FCB Loads

The mechanism for defining Forms Control Buffer loads in HASP is the $FCB macro, defined in Chapter 9. To add or change an FCB image, the installation system programmer

- codes a $FCB macro for each image
- assigns it a card sequence number from the numbers shown in Section 12.17.3.
- includes it in his HASPGEN modification deck
- does a HASPGEN to create new source for HASPPRPU
- re-assembles HASPPRPU
- executes HASPOBLD to create a new HASP load module and overlay library.

12.17.2 FCB Loads Provided by HASP

HASP provides three FCB loads, callable by the characters '6', '8', and 'U'.

The '6' image is designed for 11-inch-long forms. Channel 1 is punched at line 1, channel 2 at line 7, channel 3 at line 13, and so on to channel 8. Channel 10 is at line 49, channel 11 at line 55, channel 12 at line 61, and channel 9 at line 63. The $FCB macro is

```
FCB6 $FCB 6,66,1-1,2-7,3-13,4-19,5-25,6-31,7-37,8-43,10-49,
          11-55,12-61,9-63
```

The '8' image is designed for 8-1/2-inch-long forms, at 8 lines per inch. The punches are the same as for the '6' image, except that channel 9 is at line 64. The $FCB macro is

```
FCB8 $FCB 8,68,1-1,2-7,3-13,4-19,5-25,6-31,7-37,8-43,10-49,
          11-55,12-61,9-64
```

The 'U' image specifies only carriage channel 1 at line 1; other carriage channels are filled in by the $FCB macro to prevent forms runaway. The $FCB macro is

```
FCBU $FCB 6,66,1-1
```

12.17.3 Recommended Card Sequence Numbers

It is recommended that additional FCB images be assembled using the following card sequence numbers:

Image Name	First Sequence Number	Continuation Sequence Numbers
A	P3458000	P3458100-P3459900
B	P3460000	P3460100-P3461900
C	P3462000	P3462100-P3463900
D	P3464000	P3464100-P3465900
E	P3466000	P3466100-P3467900
F	P3468000	P3468100-P3469900
G	P3470000	P3470100-P3471900
H	P3472000	P3472100-P3473900
I	P3474000	P3474100-P3475900
J	P3476000	P3476100-P3477900
K	P3478000	P3478100-P3479900
L	P3480000	P3480100-P3481900
M	P3482000	P3482100-P3483900
N	P3484000	P3484100-P3485900
O	P3486000	P3486100-P3487900
P	P3488000	P3488100-P3489900
Q	P3490000	P3490100-P3491900
R	P3492000	P3492100-P3493900
S	P3494000	P3494100-P3495900
T	P3496000	P3496100-P3497900
U	P3540000	P3540100-P3541900
W	P3502000	P3502100-P3503900
X	P3504000	P3504100-P3505900
Y	P3506000	P3506100-P3507900
Z	P3508000	P3508100-P3509900
0	P3510000	P3510100-P3511900
2	P3514000	P3514100-P3515900
3	P3516000	P3516100-P3517900
4	P3518000	P3518100-P3519900
5	P3520000	P3520100-P3521900
7	P3524000	P3524100-P3525900
9	P3528000	P3528100-P3529900

(The remainder of this page intentionally left blank.)