SHARE PROGRAM LIBRARY AGENCY

PROGRAM NUMBER

051021

University of Miami

1365 MEMORIAL DRIVE - CORAL GABLES, FLORIDA
(305) - 284-6257

SHARE PROGRAM LIBRARY SUBMITTAL FORM

SHARE PROGRAM LIBRARY AGENCY
Triangle Universities Computation Center
Post Office Box 12076
Research Triangle Park, North Carolina
27709 USA
Attention: Mr. Joe Ragland

SPLA CONTROL NUMBER:

This form should be completed and submitted with the program package to the SHARE Program Library Agency at the address shown above. Standards and instructions for submitting programs are in the "SHARE Program Library Standards Manual".

(1) Program Number (to be filled in by SPLA) 360D-05.1.021

(2) System Type (machine) 360/370

(3) Search Key ..... Remote HASP to HASP

(4) Programming Language ..... OS ASSEMBLER

(5) Author's Name and Address ..... James F. Walker
Triangle Universities Computation Center
Post Office Box 12175
Research Triangle Park, N. C. 27709

(6) Direct Inquiries to Name and Address ..... (if different than Author)

(7) Title of Program ..... REMOTE HASP TO HASP

(8) Submitter's Installation Membership Code..... TUC

(9) Submitter's Own Program Identification and Suffix(Optional)...

(10) Primary Subject Code..... 5 1

(11) Operating or Monitor System Required OS HASP

(12) New or Revision Code (if revision, show prior Program Number in Item 1)..

(13) Year Completed..... 1971

(14) Date of Submittal..... Dec. 1, 1972

(15) Documentation (number of original pages submitted)..... 17

(16) Abstract (should contain sufficient information for a reader to determine the value of the program). Listed on the reverse side of this form are subjects which may serve as a guide for a descriptive abstract.

TUCC/IOWA REMOTE HASP TO HASP SYSTEM

prepared by
James F. Walker, TUCC Systems
William F. Decker, University of Iowa

December, 1971

Triangle Universities Computation Center
Research Triangle Park, North Carolina 27709

TUCC/IOWA Remote HASP to HASP Distribution Tape

The distribution tape is non-label, [redacted] and contains three files.

FILE 1. Version 3.0 HASP to HASP modifications. Approximately 1400 card images.

DCB=(LRECL=80,RECFM=FB,BLKSIZE=8000)

FILE 2. Documentation. 17 pages of documentation on the system.

DCB=(RECFM=UA,BLKSIZE=133)

FILE 3. Version 3.1 HASP to HASP modifications. Approximately 1400 card images.

DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000)

NOTE: The documentation was written for version 3.0 of HASP. A few minor changes are needed when used with version 3.1 of HASP:

Page 8	Paragraph 1	Comments on PTF's apply to 3.1 PTF's.
Page 8	Paragraph 2 #4	Terminal type has been changed to 8 from 7 since 7 is used in version 3.1 for 3780.
Page 14	Paragraph 3	Disregard comments about the $FREUNIT routine. HASP version 3.1 RTAM was changed to eliminate this problem.
Page 16	PTF's applied applies to version 3.0 only. All 3.1 PTF's available September 1, 1972 have been applied.	

TUC/IOWA HASP to HASP FIX UF102
SUBMITTED BY TOM SPRINGER, SECURITY PACIFIC BANK
APRIL 8, 1974

PROBLEM: HASP-HASP error for routed commands if HASP is used with MCS console support.

SYMPTOM: HASP Abend or loss of CMB's.

CAUSE: When HASP is used with MCS console support, (&Numcons=Ø), HASPCON may overlay CMB's if an operator enters a command which only contains the HASP-HASP command routine ID (@xxx). In this case, the SVC 34 data length is 4 characters. After HASPCON subtracts 4 from the command length, when the routing prefix is removed, a zero length results. The EXECUTE to move the remaining command into the CMB results in an executed length of 255 and 256 bytes of data is moved. The area past the end of the CMB is then overlayed with random core. This usually results in an invalid chain field being set in the CMB which follows the one in core to which data is to be moved.

CORRECTION: Add the following card to HASPCON -

```
./      CHANGE NAME=HASPCON                                    W0000000
        BCR  NP,R14              IGNORE IF NO DATA FOLLOWS     W3998610
```

This fix is applicable to both HASP V3.0 and HASP V3.1.

Contents

BACKGROUND

Development of Remote HASP to HASP was begun in December 1970 by Jim Walker, Triangle Universities Computation Center. By March of 1971, a HASP version 2.3 system was functioning, but it was somewhat dependent on other independent TUCC modifications.

Work began on conversion of the code to version 3 of HASP in April of 1971. Significantly, HASP to HASP was the first TUCC modification to version 3 so that there is absolutely no dependence on any other local modifications.

Bill Decker, University of Iowa, obtained the version 3 code in June 1971. He subsequently added console support, converted the retriever to use HASP overlays, generally cleaned up the code, and packaged it for distribution.

Several others who have assisted in designing, debugging or testing the system are:

Triangle Universities Computation Center
John Stephenson
Harold Jackson
Jim Scoggins

University of Iowa
Lee Shope

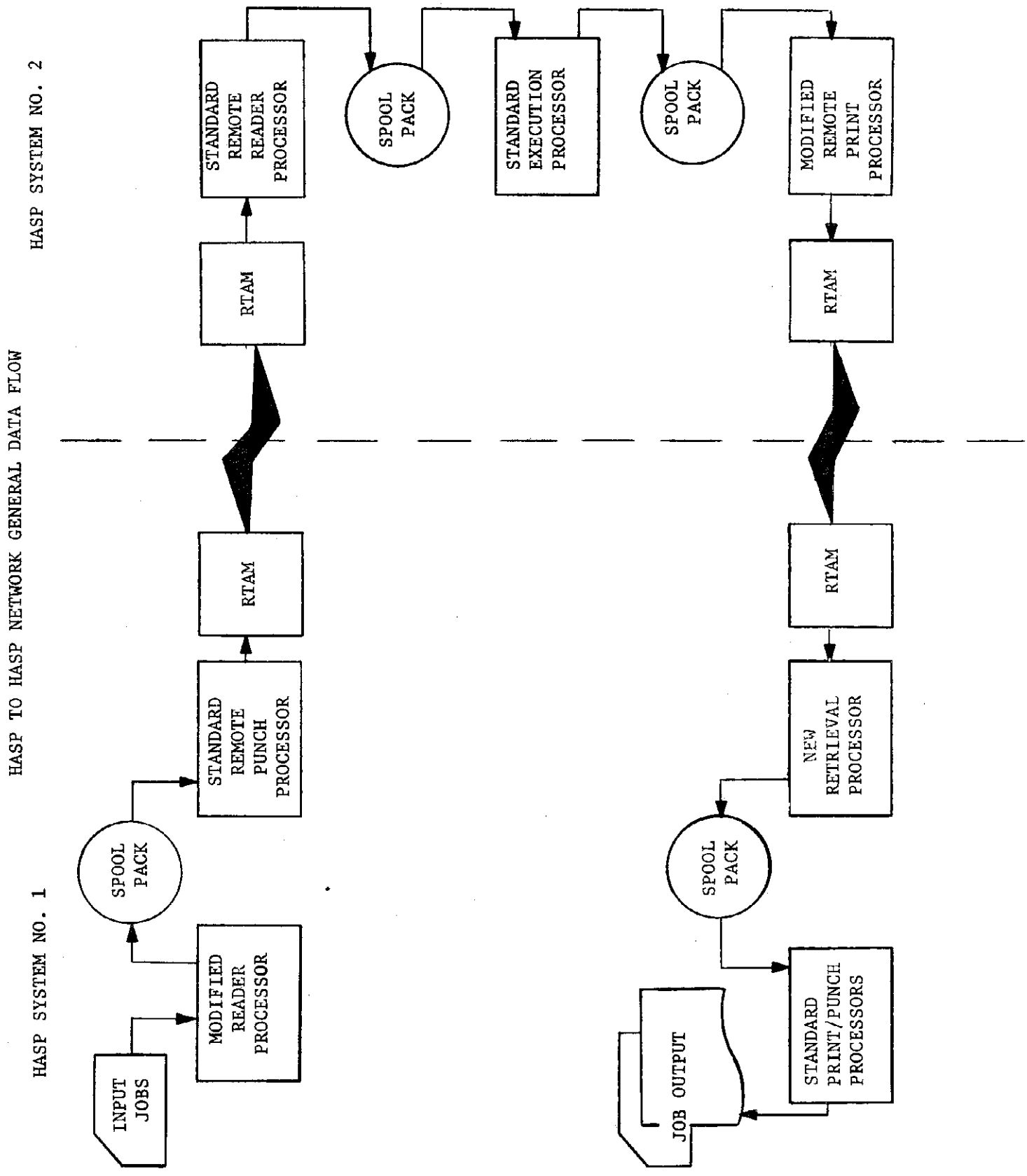
Duke University
Dave Crow
Jim Allen

University of North Carolina
Bob Daniel
Lane Ferris

North Carolina State University
Larry Roberson
Gary Funck

Iowa State University
Dana Zimmerli
Mike Bowman
Howard Jesperson

Credit should also go to the IBM HASP implementors. Their design of multi-leaving was sufficiently general and complete to allow the inclusion of HASP to HASP with a very minimum of modification.


HASP TO HASP NETWORK GENERAL DATA FLOW
HASP SYSTEM NO. 1
HASP SYSTEM NO. 2
INPUT JOBS
MODIFIED READER PROCESSOR
SPOOL PACK
STANDARD REMOTE PUNCH PROCESSOR
RTAM
RTAM
STANDARD REMOTE READER PROCESSOR
SPOOL PACK
STANDARD EXECUTION PROCESSOR
SPOOL PACK
MODIFIED REMOTE PRINT PROCESSOR
RTAM
NEW RETRIEVAL PROCESSOR
SPOOL PACK
STANDARD PRINT/PUNCH PROCESSORS
JOB OUTPUT

HASPRTAM

Modifications to RTAM fall logically into three categories -- those for access method processing (OPEN, PUT, GET, CLOSE,...), those for channel-end processing, and those for remote console message and command processing. Each of these topics will be treated separately. Note that the devices attached to a remote HASP system include two input and two output devices plus console. A new symbol, DCTIO, is defined such that the value of DCTDEVTP will indicate input or output when compared to DCTIO. Also, the order of the devices is critical. They must be chained as follows: readers, retrievers, printers, punches (i.e., input before output).

Changes to the access routines are:

OPEN: Changed to correctly process request for output or permission for input. This change is required because RCB codes are no longer absolutely unique; i.e., punch RCB matches the reader RCB; printer RCB matches the retriever RCB.

GET: For retrieval devices, length and command code (SRCB) bytes are returned in addition to the data.

PUT: Command codes are processed uniquely for console messages and header records.

CLOSE: DCTIO determines function performed.

RCB analysis: New console message types are allowed; permission and request conditions are associated with proper devices.

Other minor changes were made.

Changes in the channel end processing (Multi-Leaving Line Manager) are made to properly handle the prepare sequence processing and to properly construct the prepare CCW chain. HASP normally issues a one character initial write, but expects a two character response. Since there is no "master-slave" relationship this concept was changed slightly for HASP to HASP communications only.

Console changes were fairly extensive. It was discovered that a system deadlock potential existed in which all console buffers were in use, line input buffers were full, and all available teleprocessing buffers were queued for output. To solve this problem, console input and output functions were separated and put under control of distinct PCE's. In addition, an effort is made to test for output buffer availability, and spooling of messages is used in the event none are available. Note that &SPOLMSG must be non-zero to prevent the discarding of messages in this case. As to actual message handling, console message buffer flags are used to determine message output processing. The command code on command output (RCB = X'92') designates message routing for resultant responses. Console input saves SRCB codes and uses proper RCB codes when doing GETs from teleprocessing buffers which have been placed on the console input

```
//jobname WATFIV xxx.yyy.zzz,username,T=(,2),P=15,FORMS=1111
```

Note that JOB is replaced by WATFIV; the three account code levels are separated by periods; pages and forms keyword parameters have been added.

Many installations accept shortened JCL syntax. The RELAY control as distributed could send jobs to any such system with no syntax checking modifications in the sending system.

The logic is very simple. Upon reading the RELAY card, verifying it, and setting the proper routing code, the JCL switch (RJCLSW) is set, causing all remaining input to be written on the SPOOL packs in the JCL data set. At job termination (EOF) the JCL MTTR is used to build a punch PDDB, and the job is queued to the punch queue instead of the execution queue.

HASP commands which appear in the input stream are analyzed and queued for transmission to remote HASP systems if a proper CPUID is found.

HASPCOMM

This module was modified to provide for proper handling of the remote console ID byte if a command is being executed for a remote HASP system.

HASPCON

Changes were made to support either option of &NUMCONS. The console buffering routine and its invoker ($WTO) were modified to allow the exclusion of the time stamp and job stamp. Certain other flags (WCMBFH,WFSVR2) are allowed to remain unchanged.

Depending on the choice of &NUMCONS, either the SVC34 intercept or the input processing routine examine and queue for transmission commands of the form, @xyz$---, where xyz is a valid CPUID and $--- is a valid HASP command.

HASPPRPU

A standard punch processor punches the RELAY job. The punch DCT was modified during initialization to reset the separator indicator (DCTIDSEP), but otherwise no special code is required.

The basic idea in transmitting output jobs to another HASP system is to completely rebuild the job on the receiving system's output queue. Some relatively minor modifications were required to the PRINT part of HASPPRPU to accomplish this.

Job header records, data set header records, job trailer records, and optionally an additional accounting record are generated and transmitted to remote HASP systems. Illegal CCW command codes are used to identify the control records. (Bit 5 or X'04' must be on. In addition, bit 1 or X'40' must be on for retrieval inspection and

The main processor then reads records from the line until it receives a job header. Note that if any other data were received before the job header it would be thrown away. This situation apparently does not occur in a properly functioning system however. A dummy job could be created! This might be desirable if communicating with an unmodified HASP.

When a job header is received, the job initialization overlay (HASPFJHD) is invoked. It saves the JCT in the PCE, assigns a job number, adds the job to the input queue, and performs several other housekeeping chores.

The main processor then searches for a data set header. Should there be any data received before the first data set header, it also would be discarded. This situation would exist if page separators were generated by the transmitting system's PRINT processor.

When a data set header is received, the previous data set, if any, is terminated by the FENDDS routine. There will, of course, be no previous data set following a job header, but there will always be one after the first data set commences.

The data set initialization overlay (HASPFDSH) is then invoked to generate a new PDDB, to assign the first track to the new data set, and to get the first buffer.

The main processor then GETs and PUTs data records until another header of some type is read or a restart condition occurs. If a data set header is received, the action described above is initiated. If a job trailer is read, the job termination routine (FJOBTERM) is called to queue the job for output. If a job header is received, job termination is called to queue the incomplete job to PURGE, then job initialization begins for the new job. If end-of-file or a line restart occurs the job is queued to PURGE and the processor terminated.

The FGET routine performs all input functions for the processor. It issues the $EXTP GET to read the next input record from RTAM into FCURREC, which is in the processor PCE work area. RTAM was modified to provide the record length and machine control character in the first two bytes of FCURREC. FGET checks for delete or restart conditions, and deletes the input job if found. Otherwise it analyses the machine control character to determine if a header record was just received. It returns to the caller at specified offsets from the link register to indicate its findings.

The FPUT routine writes output records to disk. It gets buffers when required, assigns and chains tracks, builds buffers, and issues the $EXCP to write full buffers to the SPOOL pack.

The FJOBTERM routine terminates a job and queues it to either PURGE or PRINT. After terminating the last data set, it gets a track for the JCT. It builds a JCT by moving the job header from the PCE to the JCT buffer. It moves the cylinder map from the PCE to both JCT cylinder map areas since it is not clear which area will be used by

RETRIEVAL INPUT RECORD FORMATS

Job Header	llC4(JCT defined by $RTVJHD DSECT)
DATA Set Header	llC5ppfffffffffffffff
JOB Trailer	llC6(JCT defined by $RTVJHD DSECT)
Accounting record	llC7(Installation generated and defined, see note)
DATA record	llmm(Print line or card image)
Where:	ll is the length of the record pp is the PDDB type fffffffffffffff is PDDB forms information mm is the print or punch machine control character encoded as described in the HASP manual (page 12.11-11)

Byte 2 of above is SRCB constructed by HASPPRPU and HASPRTAM for transmission of header records. Details are:

Bit
0 - Turned on by HASPRTAM in all cases
1 - Defines a header record to HASPRTVL (See page 1149 or 12.11-11 in HASP manual. M = 1 = header record.)
2 - identifies header type
3 - identifies header type
4 - identifies header type
5 - Defines a header record to HASPRTAM by HASPPRPU. (This bit forces illegal CCW type.)
6 - identifies header type
7 - identities header type

Thus possible valid header types are:

C4,C5,C6,C7,CC,CD,CE,CF
D4,D5,_____
E4,E5,_____
F4,F5,_____

NOTE: It is suggested that installation defined accounting records always begin with the three character CPUID so that they can be uniquely identified by other systems.

APPENDIX B: INSTALLATION REPORT

We would like to maintain accurate data about the number and types of systems running HASP to HASP in order to provide feedback to IBM. We request that a written report be provided to us from each network running the system whenever a major change in the network takes place. The data should include for each installation in the network a minimum of:

1. Installation name and location
2. Description of equipment
3. Type lines used
4. Some indication of the data volume through each HASP to HASP link (Jobs per day or cards and print lines per day)
5. Status (Testing or production)
6. Major modifications and/or additions made to U0032 locally

In addition, we would like a written report of any bugs along with any fixes applied. This does not imply that we will necessarly correct reported problems, but we will forward bug reports to other users who have filed installation reports.

Please forward the Reports to:

Jim Walker
Triangle Universities Computation Center
Box 12175
Research Triangle Park, N.C. 27709