SHARE PROGRAM LIBRARY AGENCY

PROGRAM NUMBER

063017

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) 1130-06.3.017

(2) System Type (machine) IBM-1130

(3) Search Key ..... HASP, 1130, WORKSTATION, RTP1130

(4) Programming Language ..... 360 BAL

(5) Author's Name and Address ..... WILLIAM F. DECKER
UNIVERSITY OF IOWA
COMPUTER CENTER - LCM
IOWA CITY, IOWA 52240

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

(7) Title of Program ..... ENHANCED HASP RTP1130 WORKSTATION FOR DISK I/O

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

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

(10) Primary Subject Code.....

(11) Operating or Monitor System Required 1130 DMS-II AND HASP

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

(13) Year Completed..... 1973

(14) Date of Submittal..... 26 FEB 1974

(15) Documentation (number of original pages submitted)..... 11 PAGES

(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.

SHARE PROGRAM LIBRARY SUBMITTAL FORM

Subject Guide:

a. Purpose
b. Programming Language used
c. Version and modification level or release number
d. Field of application
e. Type of routine (main program, subroutine, etc.)
f. Specific description of machine requirements

ABSTRACT

ALLOWS CURRENT USERS OF THE HASP 1130 WORKSTATION (RTP1130) TO ADD DISK INPUT AND OUTPUT ACCESS. WHILE ONLINE TO HASP, DMS-II DISK FILES MAY BE TRANSMITTED TO HASP OR WRITTEN WITH DATA RETRIEVED FROM HASP. NO MODIFICATIONS TO HASP ARE REQUIRED. SUPPORTS ANY AND ALL IBM DISKS FOR THE 1130. ANY NUMBER OF DISKS MAY BE ONLINE CONCURRENTLY. DISKS MAY BE LOADED AND UNLOADED WHILE ONLINE.

ENCLOSED TAPE CONTAINS SOURCE.
IT IS 9 TRACK, 800 BPI, UNLABELED.
IT IS BLOCKED 10 CARD IMAGES PER BLOCK.

(Please attach additional pages if necessary).....Total pages attached____

Permission to Publish

"I hereby give the SHARE Program Library Agency permission to reprint, reproduce, and distribute this program."

(17) Signature of Submitter and Date William T. Decker 26 Feb. 1974

(18) Signature of Installation Addressee [signature]

HASP REMOTE TERMINAL PROCESSOR
EXTENDED VERSION
MOBIL OIL CORPORATION
EXPLORATION SERVICES CENTER
DALLAS, TEXAS

1130 OPERATOR'S AND USER'S GUIDE

As an extension of the original HASP/RTP1130 program, 1130 disk storage devices may be used in conjunction with and in addition to the unit record devices already available at the remote site. With this new capability, the remote 1130 user will have access to his Disk Monitor System II disk cartridges. These extensions provide the following functions not normally available with HASP/RTP1130:

A. INPUT

Reads data from any of the disk storage devices (1131, 2310, or 2311) attached to the 1130 system and prepares this data for transmission to HASP. This new capability is closely coupled to card reader processing. Data may be retrieved from disk and logically inserted into the card input data stream. Thus, the data may be simply and easily presented to HASP for further processing.

B. OUTPUT

Punch data which is received and queued for output may optionally be written to a disk cartridge. When this option is in use, the 1442 reader function will remain available (this function is obviously not available when the 1442 punch is in use). Disk output and input functions may be active concurrently.

C. GENERAL

Commands are available to allow the 1130 operator to reconfigure his remote workstation with different disk cartridges. Thus, even while the workstation is active, disk cartridges may be loaded and unloaded as required for the input and output functions.

This guide is intended for use as a supplement to the IBM supplied HASP/RTP 1130 operator's guide. Information in this document deals only with the use of the extensions and enhancements to RTP1130 for disk processing.

As of March, 1973, the following restrictions apply to this implementation:

1. The described enhancements were done only for the 1442 reader version of HASP/RTP1130. There is no problem in doing a 2501 version, but there was no requirement for it at the time of the original implementation.

2. Only DMS II cartridge formats are supported.

3. A HASP modification is desirable to prevent the problem of blank separator cards. Its purpose is to prevent the use of the blank trailer card which HASP normally punches at the end of a deck. For HASP version 3.1 the following modification might be used:

```
CLI   DCTDEVTP,DCTPUN   LOCAL PUNCH?      P1776000
BNE   PJCTWR            BRANCH IF RMT     P1778000
```

All physical drives on a particular 1130 system can be supported by this program. The operator has the ability to load and unload these drives with various cartridges as required for processing. When the program is loaded and initialized, the then active and available cartridges and the drive numbers will be displayed as a console message. This message may also be displayed by operator request with this command:

.DD

which asks that the current disk configuration be displayed. The following message will be the response:

ONLINE DISKS dd-iiii dd-iiii .....

For each available cartridge, one entry "dd-iiii" will appear. The dd is the physical drive number, and the iiii is the cartridge id. If no entries are shown, no cartridges are available.

Two commands are available to allow the operator to change the configuration. This is done as follows. Insure that no disk input or output functions are active. Then enter the command:

.DU

which tells the program that you wish to unload one or more disks. You will receive one of these two responses:

OK!
SORRY! NO GO!

The second response indicates that some disk I/O is still active. Wait until later then try again. If OK! is typed, you may unload any drives and mount new cartridges. When your new configuration is ready, enter the command:

.DM

which tells the program to mount the new configuration. You will receive the response:

OK!

You may then use the .DD command to verify the configuration. Note that it is possible to mount two drives having cartridges with identical id's. If this is done, either cartridge may be accessed by its respective drive number, but only the cartridge on the highest numbered drive will be accessible by cartridge id.

If at any time, a hardware error is detected on any disk drive, that drive in error will be removed from the configuration and any I/O activity associated with it will be aborted. If an error occurs during disk configuration (.DM or program initialization), the entire configuration will remain unavailable, and a new .DM will be required.

Disk hardware errors are reported with this message:

DISK ERROR DRIVE xx--yy

where xx is the physical drive number, and yy is one of:

NR - not ready
DE - data error
PU - power unsafe or write select

With this program, data may be read from any available cartridge and transmitted to HASP. The following special control card, if present in the 1442 card input, will be logically replaced by the designated disk data. (Data is transmitted 8 cards per sector):

```
..DATA,d,f,s,c
```

where --

d	is	1) a physical drive number 2) a cartridge id of the form /xxxx where xxxx is a hexadecimal cartridge id
f	is	1) an asterisk indicating that s and c are present (beginning sector address and number of cards) 2) a LET/FLET file name (in this case s is an optional number of cards).
s	is	1) beginning sector address in hexadecimal (no slash) 2) a specification of the number of cards to be transmitted
c	is	the number of cards to be transmitted (required for f=*)

Consider these examples:

```
..DATA,0,*,032D,37
```

37 cards (4 full sectors and 5 cards) will be read starting at sector 32D and transmitted. Physical drive 0 is accessed.

..DATA,/011D,*,032D,37

Same as above except cartridge is designated by id rather than drive number.

..DATA,8,DATAX

The LET or FLET file named DATAX on physical drive 8 will be transmitted in its entirety.

..DATA,8,DATAX,17

Same as above, except only 17 cards will be transmitted.

Special rules:

1) Incorrect statements will be ignored and this message will be displayed:

   IMPROPER ..DATA ON 1442!

   Statements may be improper because of:

   a) bad syntax (statements must start in column 1 and must contain no imbedded blanks)

   b) drive not available

   c) cartridge not available

   d) file not found

   e) * specified but s or c omitted

2) If a file name is used and a count is specified, the count will be used only if less than or equal to the file size.

3) If a disk error occurs, the transmission will be aborted.

4) If a card is encountered in the disk data which has the characters "..END" in the first five positions, the disk transmission will be terminated. "..END" is thus used for forced end file conditions.

When disk reading is complete (file limit, card limit, or ..END), 1442 card reading will resume.

Disk output is accomplished by optionally directing punched output to the disk. If this option is used, punch data is written 8 cards per sector into the specified area.

When punch output is received by the program, it will cause the following message to be issued:

PUNCH DATA FOR JOB xxx WAITING FOR 1442 OR DISK

where xxx is the HASP job number or ??? if this can not be determined. Your options at this point are to follow the procedure defined in the IBM HASP 1130 operator's guide or to specify disk output. That is, you may use .DP and punch the data or enter the ..DATA command explained above and write the data to disk. In this case the ..DATA command is typed on the console and is not entered through the card reader. After the command has been analyzed, you will receive one of these responses:

WHAT?
SORRY! NO GO!
CONTINUE?

If WHAT? is typed, the command was not recognized as valid in any way. If SORRY! NO GO! is typed, the command was recognized, but it was improperly coded or requested disks not available. If CONTINUE? is typed, the command has been properly handled and the program is requesting verification that it should proceed. You may then respecify the ..DATA or decide to use .DP or enter:

.GO

which tells the program to proceed with disk output. The following rules apply:

1) counts specified on ..DATA will limit the output

2) if a file name is specified, its limit may not be exceeded

3) partially filled sectors will be padded with zeros

4) the 1442 card reader may be used while disk output is active

When either all output is processed or a limit is reached, the following message will be issued:

xxxxx RECORDS ON iiii(bbbb--eeee)

where --

```
xxxxx =     number of card images written
 iiii =     cartridge id
 bbbb =     beginning sector address
 eeee =     ending sector address
```

If a limit is reached before all incoming data is processed, the following messages will also be received:

```
FILE LIMIT PRECEDES PUNCH EOF
PUNCH DATA FOR JOB xxx WAITING FOR 1442 OR DISK
```

You may then specify either .DP or a new ..DATA to process remaining data.

One idiosyncracy of processing punched data under HASP and OS should be known. The problem, if not taken into consideration, may result in your receiving some unwanted blank punch cards with your punch output. When writing this data to disk, these cards can be a nuisance. At the end of a deck of punched output, OS will often punch two trailing blank cards. By using the following JCL convention, this can be avoided.

Punch output is normally produced with a JCL statement which specifies SYSOUT=B. If the DCB parameter BUFNO=1 is added, the blank cards will never be produced. Thus --

```
//SYSPUNCH DD SYSOUT=B,DCB=BUFNO=1

//FT07F001 DD SYSOUT=B,DCB=BUFNO=1
```

are examples of this technique.

PROGRAM LOGIC AND GENERATION GUIDE

HASP RTP1130 PROGRAM

DMS-II DISK SUPPORT MODIFICATIONS

MOBIL OIL CORPORATION

DALLAS, TEXAS

The programming described herein is a set of modifications to the standard IBM-supplied HASP/RTP1130 remote workstation program. These modifications provide for support of 1130 disks in addition to the support for unit record devices. Exact details on the use of the program are contained in a seperate user's and programmer's guide. In this document, the internal logic of the programming is described. Also, some generation guidelines are provided.

This set of modifications is restricted to a 1442 configuration. Support for the 2501 reader could easily be added, however. In addition, only disk cartridges for the DMS-II system can be supported.

The program changes are in the form of updates to the standard RTP1130 program. Therefore, instructions conform to the 360 assembler macro set used for 1130 coding. Since nearly 2000 source statement changes and additions were made, it is advisable to include these updates in a standard HASPGEN and then to generate each required program with RMTGEN. (This is clearly not a valid approach if any 2501 configurations will be generated.)

In doing a RMTGEN, use existing generation parameters. Certain other generation variables may also be set using the $.UPDATE capability:

&DEBUG This variable, if set to one, will cause a core dump module to be assembeled. Because of memory used by the modifications, if the 1132 printer is in use, this variable must be 0 or the program can not be handled by RTPLOAD.

&DEBUGT This variable isolates the SCA trace feature formerly generated with &DEBUG. In this way memory utilization in a debug environment may be better controlled.

&NUMDRIV This variable should be set to the highest numbered physical disk drive to be supported. For example, if only an 1131 cartridge is present, then &NUMDRIV should be 0. The legal range of &NUMDRIV is 0-10. (Specifying &NUMDRIV too large wastes memory, too small prevents access to some drives.)

Before describing each module in detail, a discussion of general program structure is appropriate.

A disk I/O subroutine (DZ000) and associated FLIP provide all disk I/O. The code is patterned after DISKZ, a DMS-II subroutine. A routine activated by DZ000 and dispatched from the commutator (DERRTLT) is used to report disk errors to the operator.

A disk configuration module (DZINIT) is used during program initialization and by console processing of .DM commands to find and prepare ready disk drives for use.

Three modules (DECODER, DZNAMCH, and DZLETSR) are used to analyze ..DATA statements and to build the needed parameters for disk input or output.

A set of console changes provide for new commands.

A set of reader changes allow for disk input as signaled by ..DATA control cards.

Some reader/punch control module changes and associated TPGET modifications are used to inform the user of punch data availability.

Disk output (DOFFTLT) is done by a new commutator routine.

Various other changes were made for register saving, code conversion, etc.

A control block called the DCB (Disk Control Block) is defined for each drive to be supported. The purpose of the block is to retain information pertinent to the particular drive and to allow for multiple active I/O requests. The format of the block is described by a series of EQU source statements (sequence numbers E6607660-E6607930). A DCB look-up table is maintained. Negative entries indicate not ready drives. Positive entries are available drives. A disk activity control word (DZACTIVE) counts current disk activities. Routines using a disk are required to increment this word when entering disk activity and to decrement it at completion. The purpose of the word is to synchronize operator requests for disk loading and unloading with active I/O.

The level 2 first level interrupt handler (FLIP) saves appropriate data (registers, etc.), senses the ILSW and loads the appropriate DCB address. For a currently undefined disk, the interrupt is cleared. Otherwise the disk SLIP is entered (part of DZ000). In either case, after processing, the saved registers are restored and the interrupt is cleared via return (BOSC).

DZ000 is modeled after DMS-II DISKZ except that the DCB allows multiple uses for multiple active disks. DZ000 can be entered by BSI to DZ000 (program call) or by interrupt (SLIP) at DZ010. XR3 addresses the active DCB throughout.

For a program call, the busy indicator is checked and a "+0 if busy" return can be made. (This is possible if both input and output are active concurrently on the same drive.) Otherwise parameters are moved to the DCB, the busy indicator is set, and I/O preparation is begun.

I/O preparation includes saving and setting word count and sector data. The defective cylinder table is scanned and the sector's cylinder may be adjusted as appropriate. Then variable IOCC's are built (read, write, seek, and read check). Next, hardware is sensed for its state. Power, write select, or not ready conditions will cause an error exit (see below). The current arm position and the needed position are checked for a seek requirement.

In the initiation of all I/O operations, the IAR location is saved in the DCB so that the SLIP can return to the proper point in order to initiate more I/O or flag termination of the requested I/O. If a seek is needed, it is performed. Then one word is read to check for a correct seek. (The busy indicator will allow for 16 seek attempts.) Assuming a good seek, or if none is needed, the read or write is performed. For a write, a read-check is also performed. Then an exit is made.

At any time I/O posting is done, the post data is checked. If negative, the error reporter will be unloosed in the commutator. This commutator routine (DERRTLT) searches for DCB's with error indicators and reports the error type to the operator.

This code examines all disk hardware for ready drives and fills in the DCB look-up table appropriately. For ready drives, the DCB is filled with the defective cylinder table and other data needed for LET/FLET searches. This information is drawn from sectors @IDAD and @DCOM. A message is also built which reflects the configuration. The routine is used by the program initialization section as well as by the .DM command routine.

"DECODER" is the primary ..DATA handler. This subroutine inspects an input record for the sequence of characters "..DATA". If these are not the first 6 data characters, a "+2" return is made. Otherwise, the record is analyzed for a file specification. If the correct syntax is violated (missing comma, etc.), a "+1" return is made. In other cases a "+0" return is made. The original input record is overlaid with the decoded results which indicate the drive number, type of file (named or absolute), sector address and count (if absolute) and file name (if named).

The DZNAMCH subroutine converts a name to internal name code for DECODER.

Upon return, the caller will inspect for a named file specification. In this case, DZLETSR will be entered to perform a LET/FLET search. This module will return "+1" (disk error) or "+0" (file not found) if any problems arise. A "+2" return indicates that the file is found. DZLETSR is serially reuseable. Callers must insure that the entry point is zero before calling.

When punch output becomes available, it may be directed to the disk output processor (DOFFTLT). This module is driven from the commutator, and uses the UFCB normally associated with the 1442 punch. It packs 8 records into a disk sector, and writes them into the disk area defined to it by the ..DATA command (see console changes). At end-of-file, the last block (possibly a short block) is written, then a message identifying the output area is sent to the console. During this process, if the file limit is reached, the operator is informed, and the remaining data is left for processing by the 1442 punch or a new disk output operation.

Each card read is inspected for "..DATA" by DECODER. If such a card is encountered, disk input is initiated, and the designated file is transmitted. At the end of the file, card reading is resumed.

A new subroutine (RPFFDISK) analyzes all cards read. It returns "+1" if no disk I/O is performed and "+0" if disk I/O is done. Data is queued for transmission by the subroutine RPFFQUE.

RPFFDISK uses DECODER to check each card. A "+1" return is made if DECODER indicates that the card is not a ..DATA command. For any file specification, DZLETSR may also be used. If a file is identified, disk I/O is initiated. DECODER parameters are used to do the actual I/O. The reader UFCB code is changed to EBCDIC to prevent conversion from card code by TPCOMPR. Each sector is deblocked into 8 card images, and each image is unpacked into the 80 words of the card transmit buffers which are passed to RPFFQUE. If a ..END card is encountered prior to the limit set by the ..DATA specification, the transmission is terminated. In all cases, the last card must be processed by TPPUT before RPFFDISK will return. This is to prevent the code type from being reset to "card code" prematurely.

Several new console commands were added. Many are handled as new ".D" commands. Two others, .GO and ..DATA, are identified separately.

The .DD command will print the configuration message prepared by DZINIT.

A .DU command is used to check the DZACTIVE control word for disk activity. Any activity will be mentioned to the operator. If no disks are active, DZACTIVE is set negative, and the operator is permitted to dismount disks.

.DM can be used to force DZINIT to reconfigure. A previous .DU must have been done or the state of DZACTIVE will yield an error message.

The ..DATA command will, if punch output is available, provide parameters to DOFFTLT. This command handler will not process the command if RPCNLOCK is not set. This software lock is set by RPCNTLT. This technique prevents .DP or ..DATA from interfering with each other. It also keeps ..DATA from changing DOFFTLT parameters if DOFFTLT is already active. ..DATA uses DECODER and friends to process the command. The results are set in DOFFTLT parameter areas for its later use. Then a "CONTINUE?" message is issued. Until .GO is used, ..DATA can change these parameters.

.GO will activiate DOFFTLT if the operator is satisfied with his ..DATA command. At this time RPCNLOCK is cleared to inhibit both .DP and another ..DATA.

.DP also respects RPCNLOCK!

These modules were modified to report the job number of available punch data and to discard the HASP seperator card if present.

In TPIOX, after a buffer has been received and its UFCB located, a check is made to see if the buffer belongs to the punch. If so, a flag maintained in TPGET (see below) is inspected. If this flag is set, then the buffer just received is the first for a given job, and it is so marked.

TPGET will set the flag for TPIOX whenever a request to initiate a transmission for the punch data stream is received from HASP.

When TPGET is processing data for the 1442 punch, it checks each buffer for the flag identifying it as the first (TPIOX does this as indicated above). For a "first buffer" the first record is inspected for a HASP separator card. If one is found, the job number is stored for RPCNTLT to use in its message, and the separator card is then discarded. If the card is not a separator card, it is kept. In either case, the RPCNTLT module is unloosed in the commutator.

Various other changes were made:

1. SXPRESS saves all registers.

2. Occasional "long" instructions were required.

3. TPCOMPR was modified to inhibit card code conversion during disk transmissions.

4. Several new commutator entries were made and many have added register saving/restoring.

5. The RTPINIT module calls DZINIT.

6. The RPCNTLT module was modified for disk/punch interlock and a new message.

7. The 1442 reader/punch module when active as a punch will always revert to the reader state at the end of a punch operation. This allows the disk option to be selected.